České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Model zámeckého parku ve Vlašimi

Jan Zimandl

Vedoucí práce: prof. Ing. Jiří Žára, CSc.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Web a multimedia

10. června 2009

Poděkování

Děkuji tajemníkovi Městského úřadu ve Vlašimi Ing. Karlu Balíkovi za pomoc při hledání materiálů, řediteli Podblanického muzea ve Vlašimi Mgr. Radovanu Cáderovi za poskytnutí architektonických nákresů vlašimského zámku. Především mé poděkování patří panu prof. Jiřímu Žárovi, vedoucímu mé bakalářské práce, za rychlou pomoc při řešení problémů.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Ve Vlašimi dne 10. 6. 2009 .....

Abstract

This bachelor thesis is dealing with formation of virtual sight-seeing tour of the Vlašim manor park in VRML and its integration into the website. The text itself is mostly attending to the way of creating simple 3D models, which are optimized and simplified for VRML, and also to the way of creating textures by the help of curves. This thesis is especially focused on optimizing data transfer via net and also optimizing imaging of VRML objects.

Abstrakt

Tato bakalářská práce se zabývá tvorbou virtuální prohlídky zámeckého parku ve Vlašimi ve VRML a její začlenění do webové stránky. Text se věnuje převážně způsobu modelování jednoduchých 3D modelů, které jsou optimalizovány a zjednodušeny pro VRML, a tvorbě textur pomocí křivek. Práce je zaměřena především na optimalizaci přenosu dat po síti a optimalizaci zobrazování VRML objektů.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

1.1 Výběr tématu

Téma „Model zámeckého parku ve Vlašimi“ ve VRML jsem si vybral na základě zhlédnutí obdobného projektu „Virtuální Stará Praha“ [4]. S jazykem VRML jsem se dostal do styku již na gymnáziu, kde jsme stručně prozkoumali jednotlivé statické uzly v tradičním VRMLPadu.

Vždy mě imponovala 3D grafika. Možnost pohybovat se ve vlastnoručně vymodelovaném 3D světě mě tak nadchla, že jsem si již tehdy vytvořil model našeho bytu, především model svého pokoje. Různé informace a ukázky použití VRML jsem nalezl na internetu, také jsem narazil na „Laskavého průvodce virtuálními světy“ [6], z jehož příkladů jsem právě čerpal nejvíce.

1.2 Vlašimský park

Vlašim leží 60 km jihovýchodně od Prahy, nedaleko města se tyčí bájná hora Blaník. Ve městě se nachází romantický park [5], jenž byl vytvořen ze zámecké obory. Koncem 18. století byl park považován za nejkrásnější v Čechách. Stálo zde přes 20 staveb, které byly inspirovány antikou, gotikou, orientem, ale i přírodou. Napříč celým parkem protéká řeka Blanice.

1.2.1 Hlavní stavby v parku

Dnes je zde pouze několik staveb, které stojí za povšimnutí. Největší stavbou parku je zámek. Vystavěn byl ve 14. století původně jako hrad, až později byl přestaven na zámek. Nyní je v zámku umístěno Podblanické muzeum, restaurace U Blanických rytířů a v celém jednom křídle sídlí Střední odborné učiliště Vlašim. V muzeu se konají různé výstavy, koncerty, ale jsou zde také stále expozice.

Hned poblíž zámku se nachází jedna ze tří hlavních bran, které byly postaveny v roce 1846. Tato brána spojuje park s centrem Vlašimi, proto Vlašimská brána. V této bráně má svůj ateliér vlašimský malíř Stanislav Příhoda.

Další dvě brány, Znosimská a Domašínská, slouží jako restaurace. Za hezkého počasí nabízejí venkovní posezení v přírodě. Obě brány prošly rekonstrukcemi, Domašínská v letech

2002 – 2003 a Znosimská v roce 2005 díky projektu s finanční podporou Evropské unie a spolufinancování Města Vlašimi.

Zajímavou stavbou je Starý hrad, tyčící se na skále nad řekou Blanicí. Dříve se mu říkalo Kouzelný hrad, protože skrýval tajemný mechanismus, který zasouval všechen nábytek pod podlahu. Tato stavba byla dlouhá léta v polorozpadlém stavu, až do roku 2005, kdy byla opravena v rámci stejného projektu jako Znosimská brána.

Bezpochyby nejzajímavější stavbou ve Vlašimském zámeckém parku je Čínský pavilon. Tato romantická stavba čekala na svou rekonstrukci nejdéle, až do roku 2008, kdy dostala nové barvy a byl návštěvníkům parku umožněn vstup do interiéru v rámci prohlídkových okruhů a při významných dnech.

Protože je v parku řeka, je zde mimo tyto významné stavby několik mostů a menších mostků. Všechny tyto objekty jsou zahrnuty v mém projektu.

V parku dříve existovalo více zajímavých staveb, ale ty se nedochovaly. Město Vlašim však investuje své finanční prostředky a také prostředky z fondů Evropské Unie na další rekonstrukce staveb a obnovování již zaniklých romantických zákoutí parku.

1.3 Podobné projekty

Vlašim nedávno uskutečnila projekt zvaný „Virtuální prohlídka města Vlašim“ [1], který je složen z několika desítek pozorovacích bodů, ve kterých si může uživatel prohlédnout okolí za pomoci panoramatické fotografie (viz obrázek 1.1). Několik bodů je také v parku, ale pohyb návštěvníka je omezen právě pozorovacími body.

Obrázek 1.1: Náhled z projektu Virtuální průchod městem Vlašim

Toto nabídne můj projekt, kde se bude moci uživatel procházet ve 3D modelu zámeckým parkem, popřípadě si může prohlédnout významné objekty. Konkrétně u modelu vlašimského zámku, všech tří bran, Čínského pavilonu a Starého hradu se budou v XHTML stránce zobrazovat základní informace o aktuálně prohlížené stavbě a také její fotografie.

Kapitola 2

Získávání materiálů

Při získávání potřebných materiálů jsem narazil hned na několik problémů, které více či méně zbrzdily začátek mé práce na modelu.

Prvním problémem bylo nalezení odpovědné organizace, kde by mi byly poskytnuty mapy a architektonické výkresy staveb v parku. V muzeu ani ve vlašimském infocentru u Českého svazu ochránců přírody, neměli žádné podklady k dispozici. Proto jsem zavítal na městský úřad ve Vlašimi, kde jsem kontaktoval tajemníka, pana **Ing. Karla Balíka**. Ten mi přislíbil architektonické výkresy staveb z roku 1983, které jsem získal k zapůjčení až 1. prosince.

Mezitím jsem v parku pořídil fotodokumentaci vybraných staveb, také jsem získal turistickou mapu Vlašimi a okolí s vrstevnicemi z místní základní školy Sídliště. Některé menší stavby jsem vlastnoručně přeměřoval metrem. Díky těmto záložním řešení jsem mohl začít vytvářet alespoň základy modelů.

Posledním materiálem, který chyběl, byly plány zámku. Ředitel Podblanického muzea ve Vlašimi, **Mgr. Radovan Cáder**, mi vyhledal plány větší části zámku, které ještě zůstaly zachovány. Bohužel mi tyto podklady nemohl zapůjčit, tudíž jsem si musel potřebné rozměry z plánů opsat a přeměřit. Navíc některé části zámku úplně chyběly. Naštěstí pro model, který nemusí být tolik detailní, to nebyl zásadní problém.

2.1 Fotografování

Pro vytvoření přesné představy reálné podoby objektů bylo třeba pořídit několik desítek fotografií z různých zákoutí parku. Fotografie napomohly k zachycení aktuální podoby staveb po rekonstrukcích, které samozřejmě v materiálech z roku 1983 ještě nebyly zaneseny, také zachytily určité detaily, které ve výkresech nebyly zobrazeny.

Fotografie byly též potřebné pro tvorbu textur. Ty jsem nevytvářel přímo z vyfotografovaných obrázků, ale sloužily mi jako podklady pro jejich tvorbu ve vektorovém formátu. Pouze na textury stromů, mraků, kamene, dlažby a pohyblivou texturu jezu jsem použil upravené fotky.

2.2 Nahrávání zvuků

Pro oživení statického modelu jsem v parku nahrál zvuky ptáků a hlavně šumění jezu. Zvuky jsem zaznamenával za pomoci notebooku ASUS A6RP, externí USB zvukové karty Lexicon ALPHA a mikrofonu Shure PG57. Bohužel jsem neměl prostředky pro zapojení kondenzátorového mikrofonu, který by byl pro tyto účely nejlepší. Použil jsem tedy alespoň dynamický nástrojový mikrofon značky Shure, který má kardioidní směrovou charakteristiku. Nahrané zvuky na dynamický mikrofon byly přesto relativně dost kvalitní.

Zvuk jezu jsem nahrával z rozdílných vzdáleností, úhlů a míst nad jezem. Nejlepší se ovšem jevil zvuk nahraný ze vzdálenosti přibližně 1 m od tekoucí vody, kdy byl mikrofon nasměrován napříč na celý splav. Bylo slyšet jak šumění tekoucí vody po jezu, tak bublání pod spodní hranou.

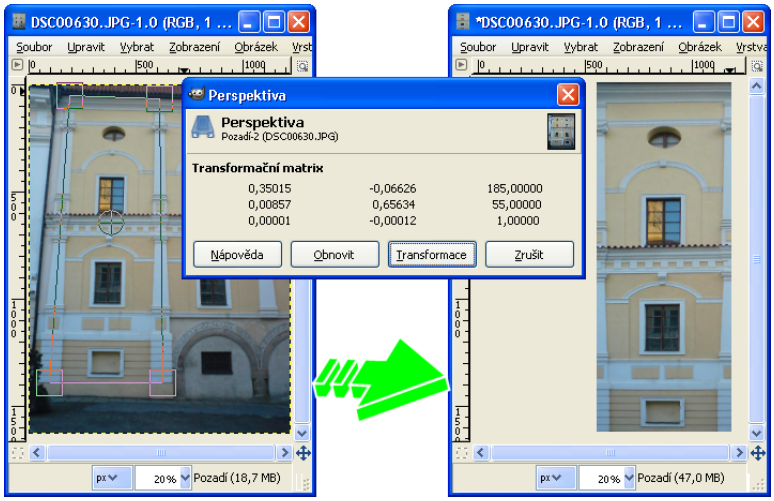
Nahrávky jezu a ptáků jsem upravoval v programu Nero Wave Editor, zbavil jsem je nechtěného šumu, upravil equalizérem zvuková pásma a zvýšil hlasitost. Výsledné soubory jsem uložil ve formátu `.wav` v několika různých stupních vzorkování. Jako optimální poměr kvality a velikosti souboru se ukázalo nastavení vzorkování na 22 kHz a 16 b hloubky záznamu.

Kapitola 3

Modelování

3.1 Použité programy

Pro přípravu materiálů a grafických prvků jsem použil poloprofesionální editor **Paint Shop Pro 7**, se kterým pracuji už několik let. Jako vektorový editor, jenž byl použit při tvorbě většiny textur, jsem využil volně šiřitelný nástroj **Inkscape 0.46**, který ve velké míře nahradí profesionální alternativy. Pro úpravu perspektivy jsem používal open source program **Gimp 2.6.3**, kde se dá perspektiva upravovat korektivně, neboli vpřed. Stačí pouze na fotografii označit čtyřúhelníkem část, která je v reálu pravoúhlá a Gimp provede danou transformaci (viz obrázek 3.1).



Obrázek 3.1: Úprava perspektivy s programem Gimp a transformační matice

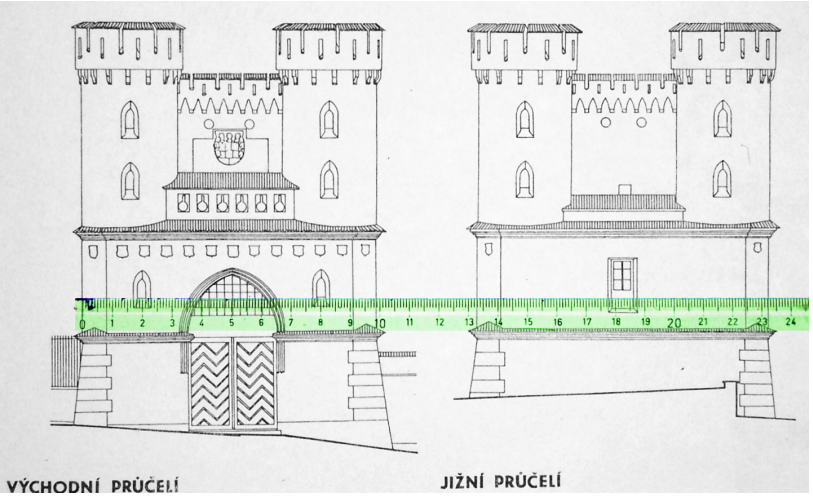
Program **Cinema 4D 9.6** je 3D modelář, ve kterém jsem modeloval převážnou většinu objektů, protože umí exportovat do VRML a navíc obsahuje modul **BodyPaint** pro mapování textur. Cinema 4D exportuje všechny 3D objekty do VRML pouze jako uzel typu `IndexedFaceSet` a metodu `DEF/USE` umí uplatnit pouze u materiálů, proto pro optimalizaci a lepší přehlednost kódu bylo nutné primitivní tělesa ručně převést v editoru **VRML-**

Pad 2.1 na základní VRML objekty a opakovaně používané objekty zadefinovat a následně pouze zobrazit kopii příkazem `USE` s potřebnou transformací.

Testování VRML scény probíhalo na internetových prohlížečích **Internet Explorer 6** a **Internet Explorer 7** za pomoci prohlížeče **Cortona 4.2**. Cortona existuje jako plugin také do internetového prohlížeče Mozilla Firefox, ovšem s tímto pluginem nemám dobré zkušenosti. Vykreslování VRML je znatelně pomalejší, počet zobrazovaných okének za vteřinu (fps) je také nižší a navíc některé verze prohlížeče Cortona nechtěly s Firefoxem 3 vůbec pracovat. Internetový prohlížeč se po načtení scény vypnul bez jakéhokoli ohlášení chyby.

3.2 Příprava materiálů

Veškeré výkresy a plány byly na velkých formátech papíru, tudíž bylo nemyslitelné je jakkoli skenovat na klasickém A4 skeneru, který mám k dispozici. Zvolil jsem metodu focení podkladů digitálním fotoaparátem, pro zachování poměrů a následné získávání rozměrů z výkresů jsem vždy přiložil pravítko, které jsem pak v počítači používal jako měřítko (viz obrázek 3.2).



Obrázek 3.2: Vyfocený architektonický výkres s pravítkem

Protože bylo vyobrazení parku na zapůjčené mapě Vlašimska přesně na řezu dvou archů, musel jsem ji v počítači ořezat a upravit, abych dostal jednu celistvou mapu. Tu jsem následně použil jako podklad pro tvorbu terénu podle vrstevnic ve 3D modeláři.

Pro vytváření textur jsem použil jako podklady fotky, jejichž perspektivu jsem upravil pomocí programu Gimp. Grafický program Paint Shop Pro umí také pracovat s perspektivou, ovšem nástroj, který nabízí Gimp je velice jednoduše ovladatelný a perspektivu převede do pravoúhlého promítání sám, bez zdlouhavého zkoušení, zdali již obrázek zachovává pravé úhly.

3.3 Tvorba textur

Textury se dají vytvářet z fotografií nebo se musí celé nakreslit v počítači podle podkladových fotek. Kreslit se mohou také dvěma způsoby, rastrově nebo vektorově. Při vytváření textur prvním způsobem, tedy z reálných fotografií, by bylo zapotřebí používat kvalitní fotoaparát a následně textury zdlouhavě upravovat v grafickém editoru. Při kreslení textur rastrově je zde omezení velikosti textury. Pokud chceme texturu zvětšit, nelze to již provést bez znatelné ztráty kvality. Proto jsem se rozhodl vytvářet veškeré textury vektorově v bezplatném editoru Inkscape, jehož primárním formátem souborů je `.svg`.

3.3.1 Vektorové textury

Vektorová grafika není omezena rozměry obrázku, protože je definována křivkami. Na křivky mohou být aplikovány různé efekty jako například rozmazání, průhlednost nebo výplň jako barevný přechod. Takovýto vektorový obrázek je možné vyexportovat do souboru `.png` v jakémkoli rozlišení a velikosti textury. Díky této výhodě jsem mohl testovat jak velká textura bude pro daný model optimální bez jakékoli ztráty kvality.

Pro vlastní tvorbu textur jsem si vždy připravil podkladový obrázek z fotky pomocí změny perspektivy v editoru Gimp. Tento obrázek jsem vložil v programu Inkscape do vrstvy „pozadí“. Do dalších vrstev jsem kreslil křivky a geometrické objekty tak, abych se přiblížil co nejvíce podkladové fotografii a skutečnosti.

Ve VRML se nejdéle načítají textury, protože jsou to řádově větší soubory, nežli vlastní modely. Zvláště při nahrávání modelu z internetu se zobrazí nejdříve datově méně náročný model a následně se objeví textury, poté, co se načtou. Aby toto zpoždění nebylo příliš rušivé, vždy jsem nastavil také barvu částí modelů, která odpovídá průměrné barvě textury a také reálné podobě. Nejdříve se tedy zobrazí barevný model a následně se objeví textury.

Jelikož Inkscape používá hexadecimální zápis barev, potřeboval jsem často převod na jednotkovou interpretaci. Pro tyto případy jsem si vytvořil velice jednoduchý program v jazyce PHP a umístil ho na své stránky `http://zimandl-j.wz.cz/programky/barvy/`[2]. Tato pomůcka převádí šestnáctkovou reprezentaci barvy jednak na 3 desítková čísla pro jednotlivé barvy RGB, kde je rozmezí hodnot od 0 do 255, dále pak na procentuální reprezentaci zaokrouhlenou na celá procenta, hodnota 255 odpovídá 100%, a konečně také na jednotkovou interpretaci zaokrouhlenou na tisíciny, která se využívá právě ve VRML.

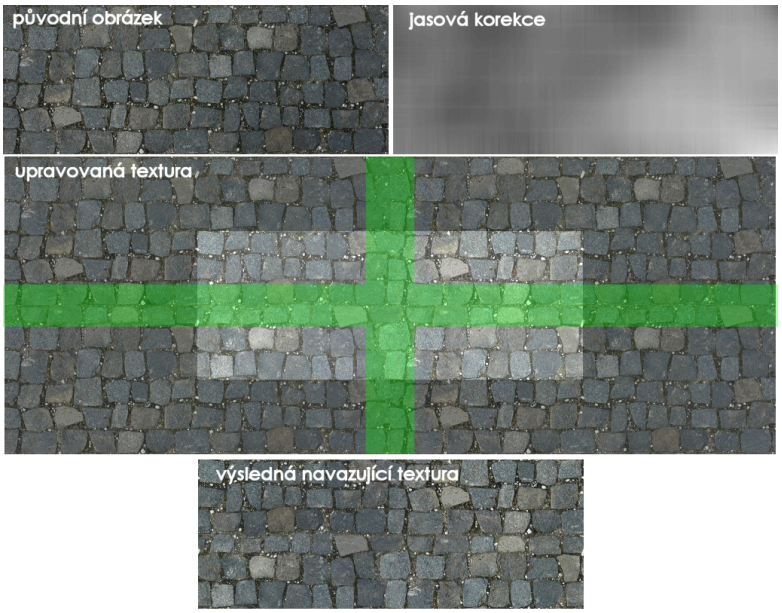
Některé textury bylo nutné vytvořit přímo z fotografií. Například textury stromů a stromořadí jsou tvořeny z upravených fotek zeleně v parku. Ale tento postup si nevyžádala jen příroda. Čínský pavilon má po obvodu čínské znaky, u kterých bylo jednodušší přetvořit vyfotografované obrázky stěn s těmito znaky.

3.3.2 Navazující textura z fotografie

Pro vytvoření textury kamene, dlažby, štěrkové cesty nebo stromořadí bylo potřeba vyrobit texturu, která sama na sebe při zopakování navazuje. V grafickém editoru Gimp jsem tedy otevřel fotku opracovaného kamene, ořízl jsem okolí a vyrovnal perspektivu. Výsledný obrázek jsem dále upravoval v programu Paint Shop Pro.

Bylo nezbytné upravit barevné rozdíly mezi jednotlivými místy. Obrázek jsem si zkopíroval, převedl barvy do stupňů šedi, rozmazal pomocí metody Gaussian, invertoval barvy a ještě zvýšil kontrast. Tím jsem docílil, že tmavá místa na původním obrázku odpovídají světlým místům v právě vytvořeném korekčním obrázku, naopak světlá korespondují s tmavými. Tento černobílý obrázek (viz obrázek 3.3) jsem následně vložil jako novou hladinu nad původní upravený obrázek kamene a změnil metodu mísení vrstev na jas. Nyní jsem pouze upravoval, jak moc by se měl jas původního obrázku ovlivňovat a tím se vyrovnala barevnost.

Dále jsem obrázek zvětšil tak, aby se mi vedle sebe vešla tato textura dvakrát. Mezi těmito dvěma stejnými obrázky vznikl ostrý přechod, kterého se zbavíme vhodným použitím nástroje nazvaného klonování nebo též klonovací razítko. Zbavil jsem se ostrého přechodu a docílil tak návaznosti mezi těmito konkrétními místy. Nyní jsem zvětšil obrázek znovu na dvojnásobnou velikost tentokrát ve vertikálním směru a zkopíroval pod sebe podobně jako předtím v horizontálním směru. Znovu jsem použil klonování tak, aby byl přechod hladký. Nakonec stačilo výsledný obrázek oříznout na poloviční rozměry, tedy na velikost původního obrázku, aby výřez zahrnoval prostřední část a nedotýkal se hran, takže vytvořené přechody jsou uprostřed obrázku a hrany musí nutně navazovat.



Obrázek 3.3: Fáze postupu tvorby navazujících textur

Na obrázku 3.3 jsou zobrazeny fáze tvorby navazující textury kamenné dlažby. Původní fotografie na sebe nenavazovala, na některých místech světlejší než na jiných. K vyrovnání posloužila černobílá jasová korekce. Zelená plocha v obrázku znázorňuje místa, kde jsem eliminoval ostrý přechod mezi texturami. V tomto případě bylo také nutné zachovat strukturu dlažby. Světlý obdélník uprostřed znázorňuje výsledný výřez navazující textury a na posledním ze série obrázků je výsledná podoba textury po drobných úpravách kontrastu a jasu barev.

3.4 Vlastní modelování staveb

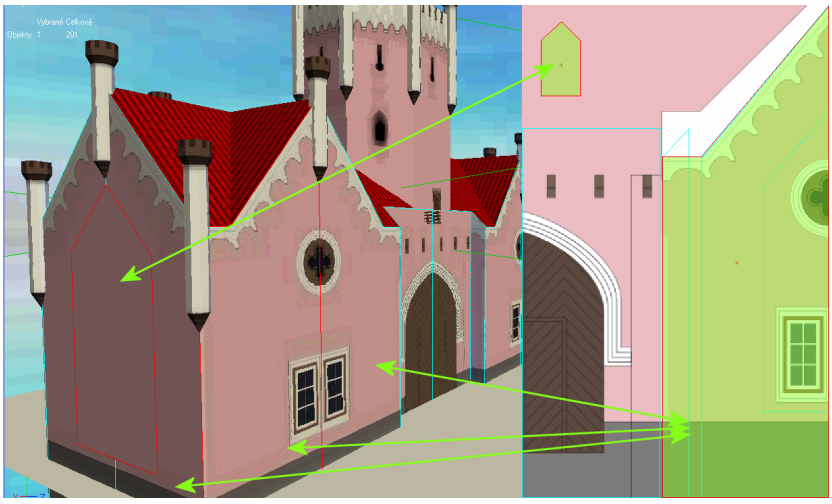
Upravené podklady se dají vložit do programu Cinema 4D jako pozadí pohledů, například půdorysu nebo bokorysu, a následně toto pozadí poslouží jako šablona pro modelování objektů. Pro přesnou velikost určitých partií jsem detaily přeměřoval podle originálních podkladů a počítal reálnou velikost pomocí uvedeného měřítka na výkresech. Ve VRML pak odpovídá 1 jednotka na modelu jednomu metru ve skutečnosti.

Při modelování jsem redukoval počet polygonů na co nejmenší, aby nebylo zobrazování náročné na výpočet, ale přitom jsem dbal na estetiku, aby nebyly na modelu nějaké rušivé hrany, kde má být například hladký oblouk. Detaily, které se dají nahradit texturou, jsem zcela vypustil.

Prvotní modely staveb byly jednoduché s minimálním počtem plošek, ovšem při texturování se ukázalo, že v určitých případech je lepším způsobem optimalizace, zvláště z hlediska velikosti souborů textur, přizpůsobit model a rozvržení polygonů právě texturám.

Modely jsou rozděleny do různých částí, kterým jsou přiděleny textury, jež jsou na tyto části namapovány pomocí texturovacích souřadnic UV. K mapování textur slouží v Cinemě 4D modul BodyPaint, kde se dají na jednotlivé polygony a skupiny polygonů aplikovat různé projekce, jako kubická, cylindrická či sférická, také je možné polygony i jejich body jednotlivě posunovat, či je různě transformovat. Takto se docílí toho, že se textura zobrazí na modelu jak potřebujeme.

3.4.1 Model Domašínské brány



Obrázek 3.4: Model Domašínské brány a namapování textur

Při tvorbě Domašínské brány jsem rozdělil plochy stěn podle různých os souměrností a oblastí, kde se motiv textury opakuje. Na obrázku 3.4 je zobrazen model brány a je zde názorně vidět, že všechny spodní stěny mají bílý dekor pod střechou, ovšem na boku nejsou okna. Toto jsem vyřešil tak, že jsem místo, kde jsou na textuře okna, nahradil dalším polygonem, na který jsem namapoval pouze barvu zdi. Stěny jsou souměrné, proto jsou rozděleny

podle osy souměrnosti. Tím se ušetřila polovina místa v textuře, protože by se jinak druhá polovina zrcadlově opakovala. Analogicky je toto řešeno na všech zdech včetně zdi s vraty. Také podobně byla vytvářena osmihranná věž brány, kde nejsou okénka na každé stěně. Po obvodu vrcholu věže je cimbuří, na které by se při modelování použilo relativně mnoho polygonů, proto jsem ho vytvořil pouze pomocí textury.

Nápadné dekorace v podobě věžiček se nedaly nijak nahradit texturami, proto jsou jednoduše namodelovány. Ve VRML jsem použil nadefinování věžičky pomocí příkazu `DEF` a následné používání s různými transformacemi. Tím se sníží zátěž na grafickou kartu. V tomto postupu nebránilo ani to, že věžičky nejsou všechny stejně velké. Na bráně jsou totiž 3 velikosti těchto dekorací, takže jsem tento problém vyřešil pouhým zvětšováním a zmenšováním měřítka nadefinované geometrie věžičky příkazem `scale`.

3.4.2 Model Čínského pavilonu

Složité střechy pavilonu jsou tvořeny z minimálního počtu polygonů, který ještě zachovává tvar reálné podoby střech. Pro obě střechy je vytvořena pouze jedna textura, jelikož jsou barevně totožné. Osmihranné patro má ze dvou stran dveře a z ostatních okna. Všechny stěny zdobí podobné motivy, pouze čínské znaky jsou různé. Patro tedy má texturu, kde je podoba stěny s dveřmi a podoba stěny s oknem. Znaky jsou doplněny pomocí osmi obdélníkových polygonů, které jsou nepatrně odsazeny od stěn, aby se zobrazily před nimi a na každém z nic jsou jednotlivé znaky jako textura s průhledností.

Zábradlí v patře je taktéž tvořeno osmi samostatnými polygony, které mají nastaven parametr `solid` na `FALSE`, což způsobí, že se zobrazí ploška z obou stran, nejen ze strany kam míří normálový vektor. Na sobě mají texturu s průhledností, která připomíná reálné zábradlí.

Ve spodní části je na sloupech složitá kresba s květinami, a tak jsem tuto texturu vytvořil z fotografie, kterou jsem upravoval v programech Gimp a Paint Shop Pro. Kolem dokola je osm sloupů. Mmezi nimi je dřevěný trám a kamenný oblouk, jehož textura je také tvořena z fotografie. Toto seskupení, tedy sloup s patkou, trám a oblouk, jsem zadefinoval ve VRML příkazem `DEF` a následně pouze sedmkrát zopakoval s jinou transformací.

Na špici je ještě umístěn čínský drak, který je reprezentován jako `Billboard`, otáčí se tedy k pozorovateli stále čelem, neboli plochou polygonu s texturou s průhledností.

3.4.3 Model Znosimské brány

3D model byl znovu podřízen texturám a polygony byly rozděleny tak, aby mohla být textura co nejmenší. Na střeše brány je použita stejná, mnohokrát zopakovaná textura střechy jako je tomu u Domašínské brány. Znovu bylo využíváno souměrnosti stavby, takže bylo zapotřebí pouze jedné poloviny textury, druhá je pouze zrcadlovým odrazem.

Čtyři věže Znosimské brány jsou stejné, bylo tedy nasnadě použít metodu `DEF/USE`. Cimbuří věžiček bylo pro zjednodušení taktéž vytvořeno pomocí textury. Na zdech směřujících do parku jsou vybudována okna, ale na opačné straně vně parku jsou místo oken střílny, jinak jsou zdi totožné. Tato situace znovu vyžadovala vytvoření polygonů, které jsem umístil nepatrně před zdi, konkrétně 1 cm ve VRML měřítku, z jedné strany jsem použil průhledné textury oken a z druhé textury střílen.

3.4.4 Model zámku

Objekt zámek je složitější jednak rozměrem, jednak rozmanitostí různých motivů na fasádách. Kupříkladu na nádvoří zámku má každé křídlo rozdílné zdi, levá strana je navíc rozdělená ještě na dvě různé budovy. Další stěny zámku jsou také jiné.

Při tvorbě textur ve vektorech bylo potřeba vytvořit mnoho detailů a mnoho dekorativních výlisků fasád. Některé motivy se potom opakovaly, takže již bylo možno je využít ze zdrojových souborů textur ostatních křídel. Určité odlišnosti, jako jsou vrata, okna v přízemí, či dveře, bylo nutno znovu vytvořit jako samostatné textury na předsazených polygonech před vlastními zdmi.

Na nádvoří se nachází vstup do Podblanického muzea, jehož zdi mají podobné motivy jako ostatní zdi okolo nádvoří, ovšem s tím rozdílem, že sloupy nejsou znázorněny pouze jako výlisky na fasádě, ale jsou plastické (viz obrázek 3.5). Nebylo tedy možné sloupy pouze nakreslit do textury, ale musel jsem je vymodelovat. Všechny čtyři jsou stejné, takže bylo samozřejmostí použít metodu `DEF/USE`. Na střechy je znovu použita již existující textura střešních tašek.

Obrázek 3.5: Náhled reálného a vymodelovaného nádvoří

3.4.5 Další modely staveb

Při modelování ostatních staveb, tedy Starého hradu, Vlašimské brány a mostů, jsem využíval stále stejné postupy, jako u předešlých staveb. Cimbuří a drobné detaily jsem vždy nahradil pouze texturou. Na mosty jsem pak využil vytvořených navazujících textur štěrkové cesty, dlažby a kamene.

3.5 Vytváření krajiny

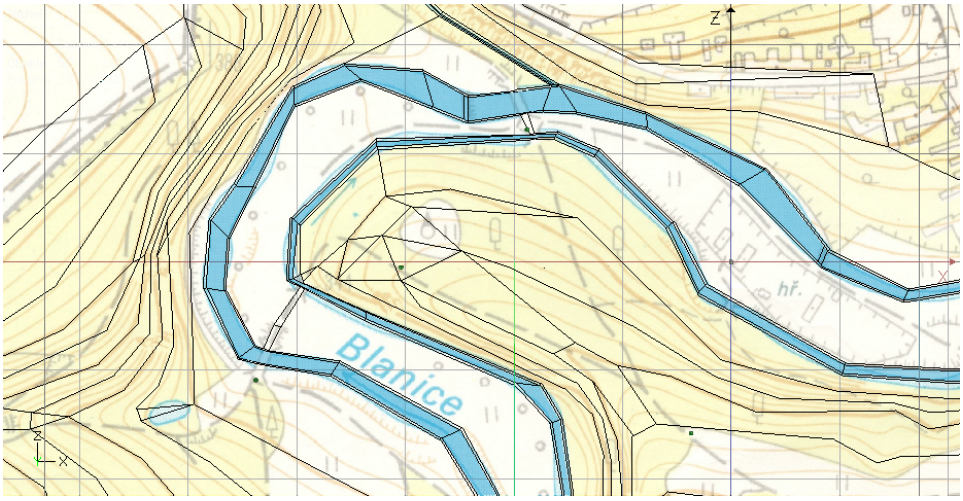
Hlavním a největším objektem, který bylo třeba vymodelovat, byl samotný terén. Parkem také protéká řeka Blanice tvořící v krajině údolí. Na řece jsou 3 splavy a přes ni spojují břehy 3 velké mosty, domašínský, znosimský a železný, a pak 3 malé mosty přes mlýnský náhon.

V parku je mnoho stezek a cest, také velké množství stromů, protože Vlašimský zámecký park lze klasifikovat jako lesopark. Bylo tedy nutné vytvořit dojem lesa se zachováním co nejmenší náročnosti na grafickou kartu a na objem přenášených dat.

3.5.1 Vymodelování terénu podle podkladu

Při modelování terénu jsem jako podklad použil upravenou turistickou mapu s vrstevnicemi. Hledal jsem nejlepší způsob, jak vymodelovat členitý terén. Použití křivek, jako vrstevnic v jednotlivých výškách, se ukázalo neproveditelné. Funkce potažení křivek, která by měla potáhnout křivky NURBS pláštěm, nevytvořila obal křivek podle mých představ a záměrů ani při pokusech s různým nastavení.

Terén jsem tedy vytvářel pomocí polygonů s několika hranami a body. Vždy jsem vytvořil polygon jako pruh mezi sousedními vrstevnicemi, počet bodů jsem tedy mohl ovlivňovat podle potřeby. Jednotlivé pásy jsem poté spojoval mezi sebou do jednoho objektu (viz obrázek 3.6). Stejným způsobem jsem také vymodeloval koryto řeky Blanice.



Obrázek 3.6: Objekt terén s řekou a mapovým podkladem

Místa, kde jsou situovány stavby, bylo nutné upravit, aby modely do zjednodušeného terénu zapadly. Toto bylo klíčové zvláště u Čínského pavilonu, který má kruhovou betonovou podstavu, jež musela nutně navazovat na terén. Důležitá návaznost terénu na objekty připadá v úvahu také u již zmíněných mostů. Zde bylo třeba body terénu parku posunout tak, aby seděly na okraje mostů.

Okolo terénu jsem v místech reálných hranic parku vytvořil jednoduchý plot z polygonů, aby nemohl avatar, tedy virtuální návštěvník, utéct z parku mimo namodelovanou krajinu. Tento plot je exportován odděleně do vlastního `.wrl` souboru a je na něm umístěna opakovaná textura bílé hrubé fasády. Před samotným exportováním celého terénu do VRML jsem od sebe oddělil plochy, které znázorňují hladinu řeky a potoků. Také plošky, které představují tři jezy, jsou exportovány zvlášť.

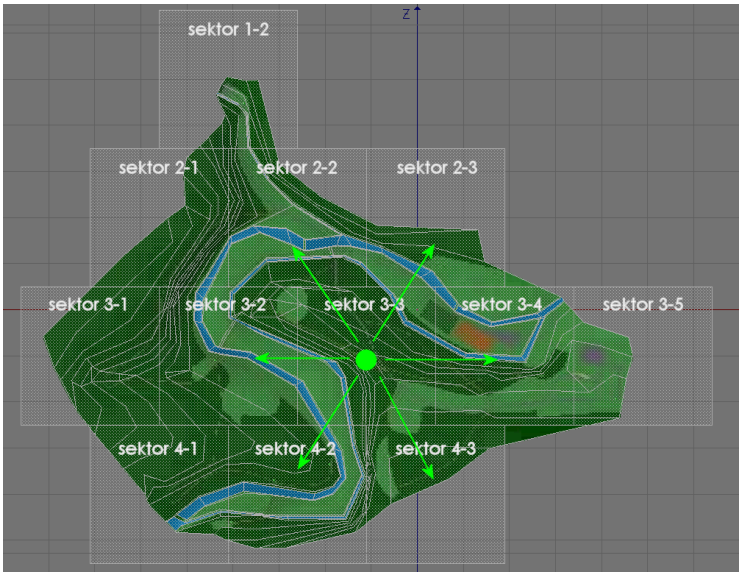
Takto převedený terén do VRML byl nepoužitelný, protože se avatar každou chvíli propadl pod úroveň terénu a zůstal stát na místě. Různé nastavování parametru `convex` na

`FALSE` nebo `TURE` ve VRML nepomohlo. Řešením tohoto problému bylo až převedení polygonů na trojúhelníky už při modelování jednoduchou funkcí, kterou Cinema 4D disponuje.

Textura terénu se skládá pouze z barev, které znázorňují les, louky a cesty. Je vytvořena ve formátu `.svg`, tedy křivkami, jejichž přednosti jsem již uvedl výše. Abych vyvolal dojem tekoucí vody po jezu, vyfotografoval jsem sérii za sebou jdoucích snímků splavu, které jsem v editoru Paint Shop Pro vyrovnal, vybral nejlepší čtyři a převedl do obrázkového formátu `GIF`, jenž umožňuje vytvářet animace ze sekvence několika okének.

3.5.2 Rozdělení terénu do sektorů

Pro následné dynamické načítání terénu bylo nutné vytvořit mřížku, která celý park rozdělí do několika segmentů. Bylo potřeba rozdělit také objekt terén parku. Určil jsem optimální velikost sektorů na 300m x 300m. Při takovémto rozměru bude sektorů dostatečný počet, v tomto případě 12, absence zatím nenačtených částí by neměla uživatele vůbec rušit. Mřížka rozdělení není pravidelná, protože při tomto způsobu návrhu mřížky, jako je znázorněno na obrázku 3.7, se zobrazí maximálně 6 okolních sektorů a jeden, ve kterém se avatar právě nachází, přitom okolí tohoto sektoru je do dostačující vzdálenosti načteno. Pokud se avatar nachází v krajním sektoru, zobrazuje se vždy již méně sektorů.



Obrázek 3.7: Rozdělení terénu na sektory a jejich načítání

Vymodelovaný terén jsem rozdělil v modeláři pomocí nástroje zvaného booleovské modelování. Použil jsem booleovskou operaci průnik objektů terén a krychle o rozměrech x, y, z 300m x 200m x 300m. Kvůli ulehčení mapování textur jsem do objektu terén ještě přidal jeden veliký polygon, který se nacházel pod celým modelem terénu a přesahoval i přes hrany krychlí obvodových sektorů. Texturové souřadnice se daly následně jednoduše namapovat pomocí čelní projekce z pohledu shora. Potom bylo možné tento pomocný polygon z každé z dvanácti částí odstranit. Jednotlivé rozdělené segmenty jsem uložil do souborů `.wrl`, které pro přehlednost nesou název sektoru.

Samotnou texturu terénu bylo nutné také rozdělit a to podobným způsobem jako 3D model. V programu Inkscape jsem ze čtverců vytvořil totožnou mřížku a exportoval jednotlivé bitmapové textury. Toto lze jednoduše udělat tak, že Inscape umožňuje exportovat pouze určitý vybraný výřez obrázku. Krychle jsem tedy udělal průhledné, vždy jsem jednu z nich vybral a následně exportoval bitmapu. Tento postup zajistil, že textury na sebe bez problémů navazují i ve VRML modelu, tudíž není třeba nahrávat na úvod hned celou na datový přenos náročnou texturu podkladu terénu.

3.6 Realizace lesa a stromů

Vlašimský park je z převážné většiny pokryt listnatým a místy smíšeným lesním porostem. Nejlepším způsobem, jak vytvořit ve VRML virtuální strom, je pomocí uzlu `Billboard`, v němž se nachází jeden obdélníkový polygon s průhlednou texturou stromu. `Billboard` se k návštěvníkovi virtuálního světa stále otáčí, takže uživatel vždy vidí strom zepředu. Při testování takto vytvořených stromů ve VRML scéně jsem narazil na zásadní problém. Aby byl les dost hustý, bylo potřeba vysadit alespoň 1 strom do prostoru o velikosti 3m x 3m. Při vygenerování takovéto scény s náhodně umístěnými stromy na plochu 3m x 500m s hustotou stromů 0,2 na 1 metr pomocí mnou naprogramovaného PHP skriptu se ukázalo, že těchto 300 textttBillboardů s texturou stromů přestává grafická karta plynule zobrazovat. Rychlost snímků byla relativně vysoká, pohybovala se mezi 30 až 50 snímky za sekundu, ale pohyb ve scéně nebyl plynulý a reakce na ovládání byly znatelně zpomaleny.

Rozhodl jsem se tedy vytvořit stromořadí jako kulisy z polygonů a do prostoru postavit pouze několik stromů jako `Billboard`, které kryjí ostré hrany stromořadí nebo vyplňují prázdný prostor, kam se stromořadí jako kulisa nehodí.

Rozmístění jednotlivých kulis v části parku je zobrazeno na obrázku 3.8, kde je možno také pozorovat rozestavení několika jednotlivých stromů. Celkově jsem v parku rozmístil 151 samostatných stromů a pro jejich zobrazení jsem využil 8 různých textur stromů, 7 listnatých a 1 jehličnatý.

Obrázek 3.8: Zobrazení polohy stromů a stromořadí v části parku

Jako pomůcku při rozmisťování stromů a stromořadí jsem využíval podkladovou turistickou mapu a fotografickou mapu parku z internetových map na stránkách http://www.mapy.cz [3].

3.6.1 Jednotlivé stromy

Průhledné textury stromů vznikaly z fotografií reálných stromů, které jsou v parku. Bylo potřeba vytvořit masku ve stupních šedi, aby bylo pozadí okolo stromu průhledné. Nejdříve jsem vytvářel masku manuálně, pomocí štětce v Paint Shop Pro s možností nastavování hustoty nánosu. Pokud se hustota sníží, vzniká šum celé stopy nástroje. Toto se dá využít jako imitace masky listů na periférii stromu.

Posléze jsem přišel na zlepšení, vytvářet masky z původních obrázků. Nejlepším postupem bylo rozložit obrázek na jednotlivé kanály barev RGB a právě modrý kanál vypadal již téměř jako hotová maska s inverzními barvami. Stromy na sobě většinou nemají barvy, které by obsahovaly větší množství modré složky, naopak nebe je čistě modrá barva. Výslednému zobrazení modré složky jsem zvýšil barevný kontrast a dále upravil štětcem tak, abych odstranil části travnatého povrchu, okolních stromů a budov v pozadí, nebo naopak lépe zviditelnil kmen a světlé listy. Takto upravený obrázek jsem použil jako masku původní fotografie stromu a výsledek byl hotov. Postupné fáze tvorby takovýchto textur můžete vidět na obrázku 3.9.

Obrázek 3.9: Postupné fáze tvorby průhledných textur stromů

Pro lepší přehlednost kódu jsem tyto jednotlivé stromy vytvořil ve VRML jako prototyp nazvaný „Strom“, který v sobě obsahuje uzel `Billboard` zahrnující jeden obdélníkový polygon o velikosti 1m x 1m. Obdélník protíná uprostřed lokální osa X, spodní body mají souřadnice Y rovny 0 a horní body jsou právě ve výšce jednoho metru. Takto je možno výslednou velikost lehce ovlivňovat pomocí transformace `scale`, tedy měřítka. Prototyp dále zahrnuje uzel zvuku zpěvu ptáků. Parametry prototypu jsou následující:

```
PROTO Strom [
   exposedField SFVec3f poloha 0 0 0      # souřadnice paty stromu
   exposedField SFVec3f rozmer 3 5 0      # rozměr stromu pouze v ose X a Y
   exposedField MFString textura [""]     # jaký obrázek stromu se použije
```

```
   exposedField SFInt32 ptaci 0          # zapnutí zvuku ptaci.wav
]{ ... }
```

Prototypy byly definovány v externím souboru, aby k nim byl zajištěn snadný přístup z více `.wrl` souborů. Stromy jsou totiž rozmístěny pouze do těch sektorů terénu, ve kterých se fyzicky nacházejí a jsou tedy zobrazovány a nahrávány dynamicky společně s okolním terénem.

3.6.2 Stromořadí

Texturu stromořadí jsem tvořil ze tří fotografií, které zachycovaly souvislou řadu stromů. Fotografie na sebe navazovaly, bylo tedy nutné je spojit do jednoho panoramatického záběru. Takovéto spojení panoramatických fotek se dá udělat pomocí jednoduchých bezplatných programů. Já jsem to ovšem provedl ručně v editoru Paint Shop Pro, protože jsem potřeboval zachovat rovnou spodní linii stromů. Ve vrstvách jsem tedy měl 3 fotografie, změnil jsem jim perspektivu tak, aby na sebe správně navazovaly a vytvořil přechody do ztracena.

Aby se textura mohla na polygonech opakovat, použil jsem postup, jakým jsem vytvářel navazující textury kamene, dlažby nebo štěrku. Zde bylo nutné zajistit opakování pouze v jednom směru. Na výsledné fotografii jsou vrcholky stromů vyfoceny tak, že je za nimi v pozadí pouze modré nebe. Bylo tedy velice jednoduché vytvořit masku průhlednosti z modrého kanálu obrázku stejně jako tomu bylo u textur samostatných stromů.

Mapování takto vyrobené a upravené textury na polygony jsem prováděl hlavně pomocí přednastavené kubické projekce. Požadavek byl takový, aby se textura na polygonech zobrazila rovnoměrně a stromy nebyly někde roztažené a někde zúžené. Mezi všemi projekcemi, cylindrickou, sférickou, plošnou, generovala tato kubická projekce nejlepší výsledek. Bylo nutno texturovací souřadnice upravovat ručně, ale korekce byly relativně minimální v porovnání s ostatními výsledky automatických projekcí.

Všechna stromořadí nejsou na rozdíl od jednotlivých stromů rozdělena pouze do svých segmentů. Jsou zobrazena vždy. Takovéto řešení je výhodné, protože stromy kopírují terén, takže vzdálený terén ještě nemusí být načten, ale je vidět silueta stromů. Navíc stromořadí kryje ještě nenačtené nebo nezobrazené části parku.

3.7 Orientace v parku

Důležitou součástí virtuální 3D prohlídky je určitý systém orientace, aby se návštěvník mohl v neznámém prostředí pohybovat a byly mu představeny hlavní zajímavá místa. V blízkosti takovýchto míst, tedy u všech staveb a různých rozcestí, jsem ve VRML umístil pozorovací místa `Viewpointy`, celkem jich je 12 a nesou názvy, které popisují jejich lokaci:

1. „vstup do parku“ – úvodní poloha uživatele u vstupu do parku z centra Vlašimi
2. „Vlašimská brána“ – pohled na Vlašimskou bránu od zámku
3. „zámek“ – pohled na zámek
4. „velká louka“ – místo na centrální louce, kde se cesta rozděluje

5. „Starý hrad“ – před Starým hradem
6. „Znosimská brána“ – před Znosimskou bránou
7. „znosimský most“ – u rozcestí za znosimským mostem
8. „pod Starým hradem“ – cestou od Znosimské brány pohled na Starý hrad pod skálou
9. „železný most“ – rozcestí u železného mostu
10. „Čínský pavilon“ – na cestě u Čínského pavilonu
11. „domašínský most“ – rozcestí u mostu – směr Domašín
12. „Domašínská brána“ – před Domašínskou bránou

Každý VRML prohlížeč umí určitým způsobem mezi těmito místy přepínat, a tak přesouvat návštěvníka mezi předem určenými místy. Toto je ovšem omezující v tom smyslu, že funkci přepínání mezi pohledy mají různé prohlížeče různě implementovanou. Z tohoto důvodu jsem vytvořil vlastní přepínání mezi pozorovacími místy.

Poblíž každého `Viewpointu` jsem umístil rozcestník (viz obrázek 3.10), který ukazuje směr, kudy je možno dojít k dalšímu místu. Na jednotlivých cedulích rozcestníku jsou napsány názvy míst. Každý ukazatel slouží jako interaktivní odkaz, na nějž stačí kliknout a avatar se přesune do dalšího pozorovacího bodu.

Obrázek 3.10: Podoba orientačního rozcestníku a názorně vybraný směr přesunu

Rozcestník je vytvořen ze dvou prototypů. První zahrnuje pouze tyč, na kterou se umisťují směrové cedule a nese v sobě informaci o poloze. Jednotlivé cedule jsou vytvořeny jako druhý prototyp, jenž funguje jako odkaz na další místo pomocí uzlu `Anchor` a objekt cedule je jeden polygon o pěti bodech. Nápis názvu další lokace je tvořen `PNG` texturou na dvou polygonech z každé strany cedule. Hlavičky prototypů vypadají takto:

```
PROTO Rozcestnik [
   exposedField SFVec3f poloha 0 0 0   # poloha paty rozcestníku
   exposedField MFNode cedule []       # pole cedulí umístěných na sloupu
```

```
]{ ... }

PROTO Cedule [
   exposedField MFString textura [""]     # textura s názvem místa
   exposedField SFVec3f poloha 0 0 0      # poloha cedule na sloupu
   exposedField SFRotation smer 0 1 0 0   # úhel, o který je cedule otočena
   exposedField MFString odkaz [""]       # odkaz na Viewpoint
   exposedField SFString popis ""         # textový popis odkazu
]{ ... }
```

Aby bylo jasné na kterou ceduli uživatel právě ukazuje a hodlá ji použít, rozsvítí se tato cedule změnou emisní barvy. Uzel `Anchor` ovšem negeneruje událost `isOver` jako například uzel `TouchSensor`. Do uzlu `Anchor` jsem tedy ještě vložil `TouchSensor`, který posílá do skriptu událost `isOver`, jež detekuje přejetí kurzoru nad sousedním uzlem. Skript už jen rozhodne, zdali je splněna podmínka `isOver` a podle tohoto nastaví emisní barvu textu a ceduli.

3.8 Struktura VRML projektu

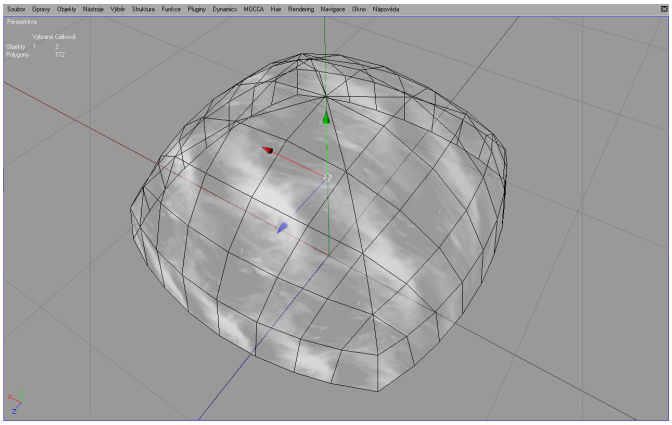
Hlavním souborem celého projektu je `vpvp.wrl`, který kromě uzlu `WorldInfo` s informacemi o práci a uzlu `NavigationInfo`, nesoucí nastavení avatara, obsahuje další uzly, které definují již konkrétní podobu modelu.

3.8.1 Světla

Osvětlení je tvořeno třemi směrovými světly. Hlavní světlo má plnou intenzitu a směřuje shora šikmo dolů. Jeho směrový vektor je nastaven na `1 -0.7 1`. Toto světlo simuluje slunce. Další dvě jsou doplňková, která mají za úkol zvýšit jas stínů, aby nebyly uměle tmavé, proto mají tyto uzly `DirectionalLight` nastavenu intenzitu pouze na hodnotu 0,3 a jejich vektory `direction` mají hodnoty `-3 -1 -1` a `3 1 -1`, takže jeden směřuje nahoru a druhý dolů.

3.8.2 Obloha

V přírodě musí být samozřejmě obloha s mraky. Zde je tvořena dvěma částmi. Uzel pozadí je reprezentován jejím barevným přechodem od modrého nadhlavníku až po horizont, kde je barevný přechod ukončen bílou barvou. Atribut `groundColor`, barva spodní polokoule pozadí, je tvořena pouze jedinou zelenou barvou. Pro imitaci plujících mraků jsem v programu Cinema 4D vymodeloval polokouli, která má čtvercovou topologii polygonů, aby se textura mraků při posouvání nedeformovala (viz obrázek 3.11). Tato textura je vytvořena z fotografie mraků, kterou jsem upravil tak, aby na sebe navazovala, a následně jsem ji barevně upravil. Potom jsem ji nastavil jako masku průhlednosti bílého obrázku. Tím jsem docílil toho, že výsledná textura mraků je průhledná tam, kde je modrá obloha a neprůhledná v místě mraků. Tuto texturu jsem uložil do formátu `PNG` s podporou 8 bitové průhlednosti a namapoval ji na objekt polokoule čtyřikrát zopakovanou v obou směrech. Pohyb mraků je pak vytvořen animací polohy textury.



Obrázek 3.11: Čtvercová topologie polokoule s namapovanou texturou

3.8.3 Viewpointy

V tomto kořenovém souboru jsou také nadefinovány `Viewpointy`, protože jakmile byly nahrávány z jiného souboru pomocí uzlu `Inline`, při otevření prohlídky se vždy nejdříve zobrazil implicitní pohled z pozice `0 0 10` a po načtení souboru s místy se teprve pohled přesunul. Takto žádné nechtěné přesuny nevznikají. Všechny `Viewpointy` mají nastaven shodný zorný úhel na 0,9 rad, což je v přepočtu přibližně 52°, aby nedocházelo k rušivému přepínání mezi různými úhly.

3.8.4 Vlastní modely

Dále se zde nahrává soubor `teren.wrl`, ve kterém se načítají stromořadí, plot, a také se zde nachází skript se senzory přítomnosti, který nahrává jednotlivé sektory s dalšími objekty, jako jsou stavby, jednotlivé stromy a terén s řekou. Tomuto se budu věnovat více v kapitole 4.4 o dynamickém načítání.

3.8.5 HUD – Head Up Display

Další věcí, která je v tomto souboru zahrnuta, je Head Up Display, neboli náhlavní displej. Slouží k zobrazení ovládacích prvků na displeji uživatele. Využil jsem ho pro zobrazení tlačítek pro přepínání mezi nocí a dnem, dále pro tlačítko zapínání avatarovy svítilny a také pro záchranné tlačítko, které přesune uživatele k zámku, pokud se v parku ztratí. Tlačítka jsou tvořena jedním čtvercovým polygonem, je na nich grafická ikona znázorňující jejich funkci. Jsou umístěny těsně před avatarem. Také se s avatarem pohybují a otáčejí, tudíž jsou na displeji stále na jednom místě.

Následují skripty, které ovládají plynulé přepínání mezi nocí a dnem. Hlavní skript na řízení přechodu si v proměnné den pamatuje, je-li noc nebo den. Podle toho změní při aktivaci ikonu na HUD a pomocí přidání a vymazání `ROUTE` zajistí zapnutí správných animací, které se následně spustí. Tyto animace proběhnou jednou a mění parametry světel, slábnutí a zesilování intenzity též přeměnu barev z nažloutlé na odstín modré pomocí `ColorInterpolatoru` a naopak. Modrou barvu světla jsem použil pro imitaci noční krajiny a měsíčního svitu. Tato

metoda modrého světla je používána i ve filmu. Všechny tyto animace přeměn jsou ještě upraveny funkcemi tak, že neprobíhají lineárně, ale první čtvrtinou funkce sinus, což zajistí plynulý přechod s pomalým začátkem, rychlejším středem a zpomalením na konci.

Kapitola 4

Optimalizace

Protože je virtuální prohlídka určena hlavně pro prohlížení na internetu, bylo třeba se věnovat nejen optimalizaci zobrazování 3D objektů, ale především optimalizaci přenosu dat, tedy velikosti souborů, které musí uživatel stáhnout, aby se mu virtuální park zobrazil.

To platí také v případě použití zvuků. VRML podporuje pouze formát `.wav`, který není komprimovaný například jako `.mp3`. Zvukové soubory jezu a zpěvu ptáků se sníženou kvalitou mají velikosti 119 kB a 481 kB. Zabírají tedy hodně místa v porovnání s celým modelem ve VRML, který díky optimalizaci zabírá pouze 875 kB. Jejich načítání zpomaluje stahování objektů a textur, proto je zvuk ve formátu `.wav` na internetu prakticky nepoužitelný, a tak jsem zvuk zařadil pouze na lokální verzi projektu, tedy i na CD.

4.1 Optimalizace modelů ve VRML

Jak jsem se již zmínil v kapitole 3.4 o modelování jednotlivých staveb, optimalizace objektů probíhala již při vlastním modelování v programu Cinema 4D. Detaily nejsou vymodelovány, jsou znázorněny pouze v texturách. Tím se zmenšila náročnost na výpočet zobrazení 3D objektů, protože je na modely použito minimum trojúhelníků.

Exportované objekty z 3D modeláře byly pro modely ve VRML nepoužitelné, protože všechny části modelů byly vytvořeny jako uzly `IndexedFaceSet` a to včetně primitivních těles a opakujících se částí modelu. Bylo tedy třeba primitivní tělesa převést. Již při modelování jsem vytvářel stromovou strukturu modelu, která reprezentuje reálnou hierarchii částí staveb. Díky tomuto uspořádání jsem mohl ve VRML jednoduše používat metodu `DEF-USE`, kdy jsem zopakované části nahradil pouze použitím nadefinovaných uzlů s jinými transformacemi.

Další nepříjemnou vlastností exportu je nepřesnost bodů. Při určitých operacích, například průniku nebo odečítání těles, se přesně zadané body zaokrouhlené na 2 desetinná místa exportovaly jako číslo se šesti desetinnými místy. Stejně tak tomu bylo u bodů oblouku. Vždy byla většina bodů takto zbytečně nepřesně exportována. Podobné to bylo u texturovacích souřadnic, které se v modulu BodyPaint ani přesně zadávat nedají. Pouze s pomocí transformací a projekcí.

Zaokrouhlovat všechny body ručně bylo nemyslitelné. Vytvořil jsem tedy další skript v PHP, který část VRML kódu se zadanými souřadnicemi tyto souřadnice zaokrouhlí na

nastavený počet desetinných míst. Po převodu vrátí výsledný kód a zobrazí kolik procent původního kódu bylo zbytečných.

Původní exportované texturovací souřadnice:

```
texCoord TextureCoordinate {
   point [ 0.564 0.832,0.563999 0.832,0.564 0.009984,
      0.011279 0.832,0.01128 0.832,0.563999 0.009984,
      0.011279 0.009984,0.01128 0.009984
   ]
}
```

Zaokrouhlené souřadnice:

```
texCoord TextureCoordinate {
   point [ 0.56 0.83,0.56 0.83,0.56 0.01,
      0.01 0.83,0.01 0.83,0.56 0.01,
      0.01 0.01,0.01 0.01
   ]
}
```

V tomto případě program vypočítal, že bylo 49% kódu zbytečných. Zaokrouhlený model na 2 desetinná místa není nijak degradován, navíc jsem se zbavil typických exportních nepřesností jako souřadnicové body typu `-0.000001, 1.799999`. Texturovací souřadnice jsou ovšem na takovéto zaokrouhlování velice citlivé, zvláště pak u velkých objektů, například jednotlivých sektorů terénu. Tyto souřadnice bylo nutné zaokrouhlit na 3 desetinná místa, pak byl výsledek ideální a upravení bodů nebylo nijak patrné. Výsledná velikost souboru však byla znatelně menší. Program jsem umístil na své webové stránky http://zimandl-j.wz.cz/programky/round/[2] a používal ho buď odsud, nebo z lokálního serveru na osobním počítači.

optimalizace	nekomprimovaný	komprimovaný
přímý export	**281 778 B**	33 631 B
upravené VRML	52 851 B	9 942 B
zaokrouhlené body	43 031 B	**8 348 B**

Tabulka 4.1: Velikosti souboru modelu Vlašimské brány s různými stupni optimalizace

Dalšího zmenšení souboru jsem docílil ukládáním do komprimovaného formátu `.wrl`, který program VRMLPad podporuje. Porovnání velikostí souborů můžete pozorovat v tabulce 4.1. Z exportovaného souboru s nepřehledným kódem o velikosti 282 kB se mi podařilo vyrobit soubor zabírající 8 kB, jehož výsledek vypadá naprosto stejně jako původní export, navíc je jednodušší pro zobrazování.

4.2 Optimalizace textur

V kapitole 3.3 jsem se již zmínil o vytváření textur. Převážnou většinu texturovacích obrázků jsem kreslil pomocí vektorového editoru Inkscape a díky tomu jsem je mohl exportovat v jakémkoli rozlišení a velikosti. Zde se tak vyskytl prostor k experimentování s velikostmi textur. Při zkoušení jsem dospěl k závěru, že určujícím parametrem velikosti textury bude velikost samotné plochy objektu, na němž je textura nanesena. Jako dostačující minimální poměr jsem stanovil toto: 2 cm plochy na objektu budou odpovídat nejméně 1 px na obrázku textury. Podle tohoto kritéria jsem ukládal výsledné obrázky z Inkscapu.

4.2.1 Formáty textur

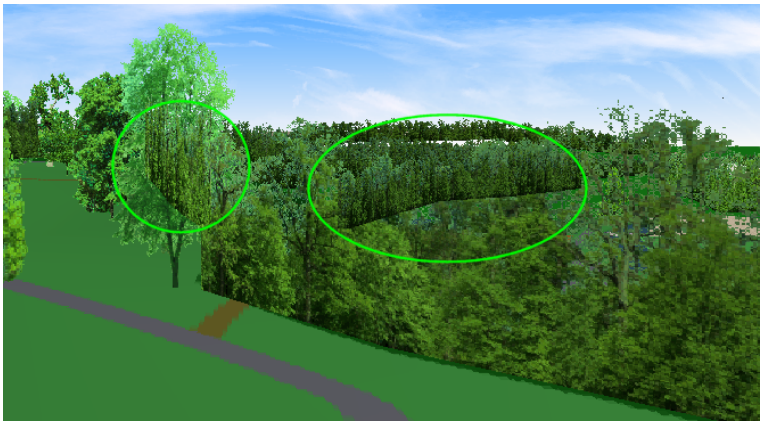
Pro každou texturu jsem vybíral ten nejlepší výsledek jako poměr velikosti souboru a kvality zobrazení barev. Vždy jsem obrázky převedl v editoru Paint Shop Pro do formátů `.jpg`, `.gif` s omezenou paletou na 50 – 100 barev podle míry barevnosti samotné textury a do formátu `.png`. Jelikož jsou textury tvořeny z křivek, nemají tak velkou barevnou hloubku. Bylo tedy možné snížit barevnou hloubku například až na 50 barev bez velké ztráty kvality, ovšem se znatelným snížením velikosti souboru.

Objekty jako jednotlivé stromy, stromořadí nebo mříže a ploty, které potřebovaly průhledné textury, byly omezeny pouze na formáty GIF a PNG. Formát PNG navíc nabízí 8 bitovou průhlednost, je tedy možné vytvořit poloprůhlednou texturu na rozdíl od GIFu, kde je textura v bodě buď průhledná nebo ne. Ukládání do PNG z programu Paint Shop Pro má více možností nastavení. Je možné omezit barevnou paletu na různé počty barev jako je tomu u GIFu, ale dá se potom použít už jen 1 bitová průhlednost. Výsledný soubor s obrázkem textury je však menší než stejný výsledek ve formátu `.gif`. Většinu textur jsem tedy převedl do formátu PNG, kromě textur s potřebou větší barevné hloubky. Musel jsem tedy ponechat textury složitých kreseb na Čínském pavilonu, textury kamenů, štěrku a dlažby ve ztrátovém formátu JPG. V takových případech byla velikost souborů vždy menší než velikost ostatních formátů.

Na textury stromů jsem použil snížení počtu barev a tedy i 1 bitovou průhlednost. Lépe by se zde hodilo využití poloprůhlednosti PNG, ale soubor byl značně velký. Textura listnatého stromu s 8 bitovou průhledností zabírá 75 kB a stejný soubor s 1 bitovou průhledností, kde jsou poloprůhledné části nahrazeny nastavenou barvou, v případě stromu odstínem zelené, zabírá 18 kB. Takovýto strom s jednoduchou průhledností nijak neruší celkový dojem z celého modelu a znatelně zmenší objem přenesených dat. Stromy, kterým obecně nebylo možné tolik snížit barevnou hloubku, jsem velikosti textur minimalizoval do únosné míry a to tak, že přibližně 10 cm odpovídá jednomu pixelu textury.

Při převedení většiny textur na formát PNG se objevily chyby v zobrazování viditelných a skrytých částí jednotlivých objektů ve scéně. Na obrázku 4.1 je vidět, jak se před strom a stromořadí dostalo další vzdálené stromořadí.

Z tohoto důvodu jsem veškeré textury s 1 bitovou průhledností nahradil zpět o trochu většími obrázky ve formátu GIF. Ostatní textury, které nepotřebovaly průhlednost, jsem převáděl znovu do PNG se sníženým počtem barev v paletě, ale tentokráte s vypnutou průhledností v nastavení exportu. Pak se už tato chyba nikde nevyskytla.



Obrázek 4.1: Chyby v zobrazování PNG s 1 bitovou průhledností

4.3 LOD – úroveň detailů

Pro optimalizaci počtu zobrazených trojúhelníků ve scéně se používá metoda LOD, Level of Detail neboli česky úroveň detailů. Zde se nadefinuje vzdálenost od modelu, při které už není potřeba zobrazovat všechny detaily a proto je zobrazen jen zjednodušený model. Tuto metodu jsem použil u všech modelů staveb.

Jako druhý stupeň LOD jsem vždy namodeloval velice zjednodušený tvar stavby, převážně také ze základních primitivních těles jako je válec, krychle, kužel nebo koule. V tomto druhém stupni nejsou už použity textury, pouze materiály s barvami jednotlivých částí. Tomu se tedy podřizovalo rozdělení částí těchto zjednodušených modelů.

U zámku se například jako druhý stupeň detailů zobrazují jen některé jeho stěny. Toto je možné jen proto, že je zámek situován na začátku parku, tedy blízko oplocení, které omezuje pohyb ve virtuálním parku. Vzdálenost, při níž se přepne model do méně propracovanější verze, lze docílit pouze při pokračování do dalších částí parku, ze kterých je však vidět zámek jen ze tří stran. Nádvoří zámku pak může být naprosto vypuštěno. Zde byly textury ponechány, protože zobrazují okna, která by z dálky chyběla a model by tak byl neúplný.

Jako třetí stupeň je použit prázdný uzel `Group`, takže se model vůbec nezobrazí. Vzdálenosti, při kterých se LOD přepíná jsou nastaveny pro každou stavbu různě. Většinou se první vzdálenost pohybuje okolo 100 m a druhá vzdálenost, která přepíná druhý a třetí stupeň LOD, je 400 m.

4.4 Dynamické načítání

Dynamické načítání jsem použil pro minimalizaci stahovaných dat při prvním načtení celé prohlídky. Při spuštění virtuální prohlídky se načtou pouze okolní sektory, které jsem určil již při modelování terénu. Více je o tom v kapitole 3.5.2 o rozdělení terénu do sektorů. V mém případě se při prvním načtení nahraje pouze sektor s označením 3-5, ve kterém se avatar na začátku nachází, a sektor 3-4, který je jediný sousedící se sektorem 3-5. Nahrají se pouze 2 sektory ze 12. O to se stará skript ve spolupráci s 12 senzory přítomnosti, které jsou v souboru `teren.wrl`, jenž je nahrán v kořenovém souboru projektu `vpvp.wrl`. Uzly

`ProximitySensor` reprezentují krychli, do které když avatar vstoupí, respektive z ní vystoupí, vyšle události `enterTime`, respektive `exitTime`. Pro každý sektor je jeden senzor, jenž má stejné rozměry jako navržené dělení terénu, tedy 300m x 200m x 300m. Tyto senzorické krychle jsou umístěny tak, aby každá pokrývala právě jeden celý sektor.

Události ze senzorů jsou odchytávány skriptem, který přidává do uzlu `Group` uzlem `Inline` soubory reprezentující jednotlivé sektory. Každý sektor je pak vytvořen jako LOD se dvěma úrovněmi detailů. V první úrovni je obsah celého sektoru, tedy terén, řeka a případné stromy nebo stavby. Druhá úroveň je prázdný uzel `Group`, což je prázdná geometrie. Není pak potřeba mazat načtené sektory, které nemusí být zrovna zobrazeny, protože je avatar v jiné části parku. Již načtené sektory tedy zůstávají v paměti a LOD jednoduše zajistí, že jsou dané objekty náležící sektoru zobrazeny či nikoliv. Vzdálenost, kdy sektor zmizí, je nastavena na 450 m, je tedy větší nežli úhlopříčka čtvercového sektoru. Tím je zajištěno, že se požadované sektory včas objeví.

Hlavička skriptu zahrnuje 12 `eventIn` událostí pro každý senzor sektoru, kam se posílá čas vstupu do sektoru a tím se aktivuje jedna ze 12ti funkcí. Tyto funkce mají na starost přidávání uzlů `Inline` se sektory, které jsou v okolí aktuálně aktivovaného senzoru včetně sektoru, který k němu patří. Funkce také kontrolují, jestli je již sektor nahrán, aby ho nenahrávaly znovu. K tomu slouží 12 proměnných typu `bool`, již jsou nastaveny na FALSE, pokud ještě nebyl daný sektor načten.

Kapitola 5

Začlenění modelu do XHTML stránky

Aby mohl být projekt prezentován na internetu, je třeba ho začlenit do webové stránky. Stránka uživateli ukáže nejen 3D model parku, ve kterém se může pohybovat, ale také fotografie a informace o celém parku a stavbách v něm. Grafické zpracování a barevné ladění je inspirováno oficiálními stránkami vlašimského parku [5].

Celá webová prezentace obsahuje sekci „Virtuální prohlídka“, což je hlavní stránka se zobrazeným VRML modelem. Sekce „Mapa parku“ s renderovaným obrázkem mapy se stavbami a rozcestníky slouží pro lepší orientaci ve 3D modelu i v reálném parku. Další sekce se nazývá „Hlavní stavby“, kde jsou prezentovány modely šesti hlavních staveb, a konečně poslední sekce je nepostradatelná „Nápověda“, jež uživateli poradí, jak virtuální prohlídku ovládat.

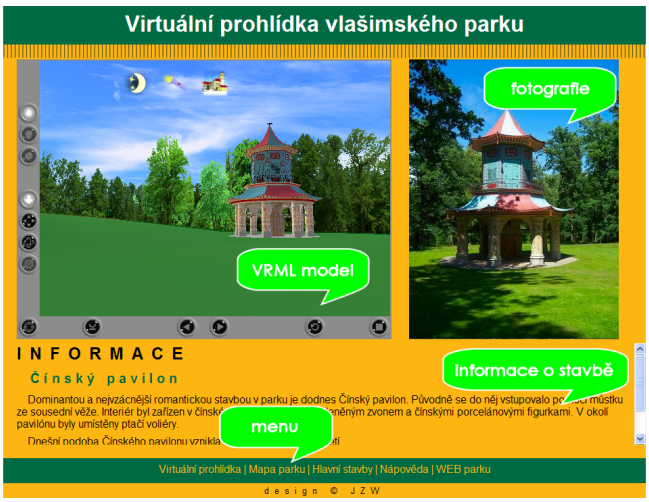
5.1 Rozvržení hlavní stránky

Po vstupu na stránky projektu Virtuální prohlídka vlašimského parku se objeví jednoduchá XHTML stránka, ve které je VRML model vložen jako element `<object>`. Pokud internetový prohlížeč nepodporuje zobrazování `.wrl` souborů, je návštěvníkovi sdělena informace s odkazem na stažení prohlížeče Cortona.

```
<object id="obj_vrml" type="x-world/x-vrml" data="vpvp.wrl"
width="550" height="410">
<param id="param_vrml" name="src" value="vpvp.wrl" />
Váš prohlížeč nepodporuje zobrazování VRML. Prosím stáhněte si prohlížeč
souborů .wrl. Například na WEBu <a href="http://www.cortona3d.com/cortona"
title="web firmy Parallel Graphics">www.cortona3d.com</a> prohlížeč Cortona.
</object>
```

Vpravo se nachází fotografie, která se mění podle polohy avatara, stejně jako textové informace ve spodní části. Aby se dal jednoduše měnit obsah těchto buněk bez nutnosti používat server na spuštění PHP, jsou vytvořeny samostatné stránky s obrázkem a stránky

s textem pro každou ze šesti hlavních staveb plus jedna stránka s malbou, zobrazující starý park, a jedna stránka s textem o celém parku. Tyto XHTML stránky jsou ve hlavním okně zobrazovány pomocí rámců pojmenovaných „foto“ a „info“.



Obrázek 5.1: Vzhled WEBové stránky s informacemi, fotografií a VRML světem

Ve spodní části je menu s dalšími sekcemi a odkazem na oficiální webové stránky o vlašimském parku [5]. Rozdělení všech elementů hlavní stránky je zobrazeno na obrázku 5.1.

5.2 Zobrazení fotky a informací

Aby se mohly ve stránce zobrazovat informace o blízkém objektu, bylo potřeba odhalit přítomnost avatara u jedné ze 6ti hlavních staveb. Každá z těchto staveb má okolo sebe různě velkou krychli uzlu `ProximitySensor`, která vysílá události `enterTime` a `exitTime` do připraveného skriptu. Při vstupu do jednoho ze senzorů přítomnosti se pošle `eventOut` do skriptu, zde se spustí funkce příslušící aktivnímu senzoru. Každá funkce obsluhuje jeden senzor a příkaz `Browser.loadURL(MFString url, MFString parameter)` s parametrem `target=foto` nebo `target=info` zobrazí zadanou url adresu v požadovaném rámci podle hodnoty parametru `target`. Při opuštění aktivního prostoru a vyslání `exitTime` se v rámcích pomocí stejného příkazu zobrazí úvodní malba a obecné informace o parku.

V sektorech, kde se nacházejí stavby, musí být definován obsluhující skript a spojení se senzorem. Aby byl skript pro všechny sektory jednotný, nadefinoval jsem ho jako prototyp. Pak se jen načte definice prototypu v souborech sektorů, ve kterých je nutná, a prototyp se použije pro propojení příkazem `ROUTE` se senzory.

Textové informace jsou převzaty z oficiálních stránek parku [5] a aktualizovány. Protože jsou texty uloženy jako klasické XHTML stránky, není problém je v budoucnu rozšířit o nové informace, odkazy, či obrázky.

5.3 Samostatné hlavní stavby

Jelikož jsem si dal záležet na modelování hlavních staveb, tedy zámku, všech tří bran, Čínského pavilonu, Starého hradu, a vytvořil jsem objekty ze všech stran, rozhodl jsem se je ukázat v plné kráse i v projektu. Většina z nich není ze všech stran ve virtuální prohlídce přístupná.

V sekci s hlavními stavbami je seznam těchto šesti památek s obrázky, též okno rámce, které zobrazuje další XHTML stránku, v níž je vložena VRML scéna pomocí tagu `<object>`. Tato VRML scéna zobrazuje vždy jednu otáčející se stavbu na kulaté ploše. Kliknutím na další stavby v seznamu se ve vloženém rámci zobrazí jiná stránka s vybraným točícím se VRML modelem.

V každém adresáři s hlavní stavbou jsou dva soubory s názvy `show.wrl` a `show.html`. V souboru `show.html` je stránka s jedním tagem typu `<object>`, jenž zobrazuje VRML scénu ze souboru `show.wrl`. Tato jednoduchá VRML scéna obsahuje pouze vypnutí všech navigací, takže se uživatel nemůže ve 3D scéně pohybovat, jeden `Viewpoint`, definici externího prototypu `Show` a následné použití prototypu, jemuž je přiřazen objekt, který se má zobrazit. Dále je zde možno nastavit dobu otočení, která je implicitně nastavena na 8 vteřin. Je možné také ovlivnit velikost kruhu (implicitně má poloměr 50 m), kterou bylo nutné u zámku změnit, protože je rozměrnější než ostatní stavby. Dalšími možnostmi nastavení je také určení osy otáčení a posun objektu, což jsem využil k vycentrování modelů zámku a Starého hradu, které nemají svou osu objektu uprostřed modelu.

Prototyp je znovu použit kvůli sjednocení VRML scén otáčejících se prezentací. V souboru `showProto.wrl` je uvnitř prototypu nadefinována animace, která otáčí s přiřazeným objektem. Dále je zde pozadí s přechodem tmavě modré barvy do světlejší, šedivá plocha a uzel `Transform`, jemuž se přiřazuje zobrazovaný objekt.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo vytvořit virtuální prohlídku vlašimského parku ve VRML, která bude začleněna do XHTML stránky, a tím pádem přístupná na internetu. Optimalizace přenosu dat zajistí, že si mohou 3D model vychutnat i ti, kteří mají pomalejší připojení k internetu.

Vytvořil jsem kompletní 3D park, který odráží skutečnou podobu reálného parku s přihlédnutím na nutné zjednodušení 3D modelů. Porovnání můžete vidět v příloze na obrázcích B.1, B.3, B.2, B.4 a B.5.

Díky optimalizacím textur, zjednodušování modelů, zaokrouhlování bodů a ukládání do komprimovaného formátu se mi podařilo vytvořit plnohodnotnou virtuální prohlídku krajiny ve VRML, která zabírá pouze 875 kB. Dynamické načítání navíc zajistí, že se při prvním nahrání modelu načte pouze nezbytně nutná část a ostatní části se stahují v průběhu podle polohy avatara ve 3D světě.

6.1 Uplatnění projektu

Tento projekt by našel uplatnění u široké veřejnosti, zvláště pak v turistickém ruchu. Trochu omezující a odrazující může být skutečnost, že je k prohlížení potřeba nainstalovat prohlížeč VRML, ale pro ty, kteří tak učiní bude virtuální procházka parkem zábavou i poučením.

6.2 Další možné rozšíření

Textové informace u hlavních staveb se dají jednoduše rozšířit. Text je zapsán v samostatných XHTML stránkách, kam se mohou nové části doplnit. Celá prohlídka může být také rozšířena o další nová zákoutí, kterým jsem nevěnoval v této práci takovou pozornost. V tomto případě by bylo třeba přepracovat části kódů, případně skripty na zobrazování informací a fotografie. Nemusí se ovšem předělávat celý projekt. Terén a jeho dynamické načítání je již hotov a připraven například na přidání dalších objektů pomocí uzlů `Inline`.

Literatura

[1] Život ve městě Vlašim.
http://www.nasevlasim.cz.

[2] Jan Zimandl ml. - osobní stránky.
http://zimandl-j.wz.cz.

[3] Mapy.cz.
http://www.mapy.cz.

[4] Virtuální Stará Praha.
http://www.cgg.cvut.cz/vsp.

[5] Vlašimský zámecký park.
http://www.vlasimskypark.cz/.

[6] J. Žára. *Laskavý průvodce virtuálními světy.* Computer Press, 2000.

Příloha A

Seznam použitých zkratek

3D Three-Dimensional - trojrozměrný

fps frames per second - počet okének za vteřinu

NURBS Non-Uniform Rational B-Splines

VRML Virtual Reality Modelling Language

XHTML Extensible Hypertext Markup Language

PHP PHP (= Personal Home Page) Hypertext Preprocessor

LOD Level of Detail - úrovně detailů

HUD Head Up Display - náhlavní displej

RGB red, green, blue - červená, zelená, modrá

PNG Portable Network Graphics

GIF Graphics Interchange Format

JPEG Joint Photographic Experts Group

⋮

Příloha B

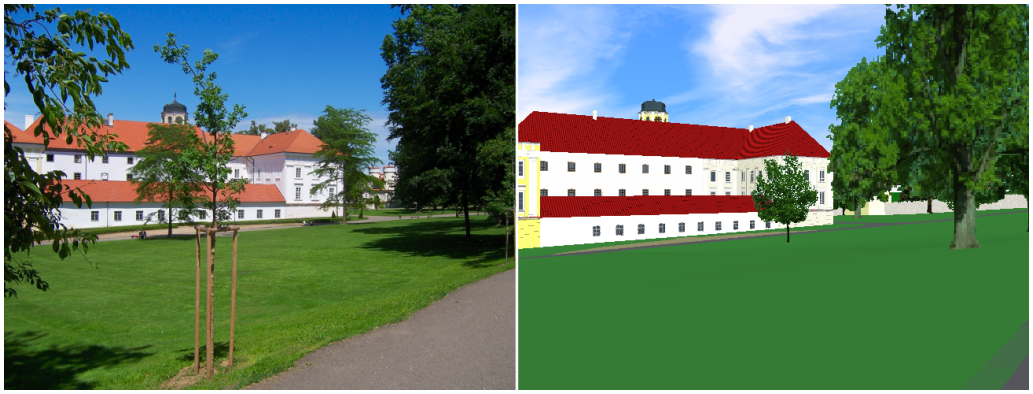
Ukázky výsledků

Jako motivaci pro potencionální uživatele zde uvádím náhledy z VRML modelu parku a fotografie z odpovídajících míst v reálné přírodě.

Obrázek B.1: Pohled na Starý hrad na skále

Obrázek B.2: Porovnání podobnosti nádvoří zámku

Obrázek B.3: Na louce před Znosimskou branou



Obrázek B.4: Celkový pohled na zámek z cesty

Obrázek B.5: Panoramatický záběr na velké louce

Příloha C

Instalační a uživatelská příručka

Pro prohlížení virtuální prohlídky zámeckého parku ve Vlašimi je třeba mít nainstalovaný prohlížeč souborů `.wrl`, tedy VRML prohlížeč. Takovýchto prohlížečů existuje celá řada a vetšina z nich je bezplatná. Celá prohlídka je integrovaná v XHTML stránce a na CD se nachází ve složce `VPVP`. Virtuální prohlídka vlašimského parku je spustitelná přímo z CD souborem `VPVP/index.html`.

Příloha D

Obsah přiloženého CD

```
CD Bakalářské práce
|
|   index.html                  - výchozí webová stránka
|   README.txt                  - obsah CD
|
|___text
|       zimanj1-bach2009.pdf    - text bakalářské práce v PDF
|
|___text_src                    - zdrojové soubory textu v LaTeXu
|   |   zimanj1-bach2009.tex    - hlavní zdrojový soubor LaTeXu
|   |___figures                 - adresář s logem ČVUT
|   |___obr                     - obrázky použité v bakalářské práci
|   |___texty                   - texty kapitol
|
|___VPVP                        - kořenový adresář projektu
    |   vpvp.wrl                - kořenový soubor projektu
    |___data                    - adresář s nápovědou v PDF
    |___html                    - adresář s informacemi o stavbách v HTML
    |   |___foto                - adresář s fotografiemi staveb
    |___obrazky                 - obrázky použité ve stránce a VRML
    |___vrml                    - všechny objekty ve VRML s texturami
```