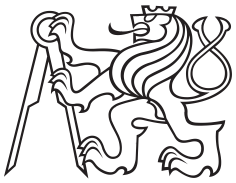
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Modely 3D scén pro jízdní simulátor

Petr Brachaczek

Školitel: Ing. Jiří Bittner, Ph.D.
Obor: Softwarové technologie a management
Zaměření: Web a multimédia
Květen 2017

Poděkování

Děkuji vedoucímu semestrálního projektu Ing. Jiřímu Bittnerovi, Ph.D., za odborné vedení spolu s laskavým přístupem. Také děkuji svým rodičům, kteří mě ve studiu silně podporují.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze 26. května 2017

Abstrakt

Tato bakalářská práce se věnuje modelové úpravě rozsáhlé scény exteriéru pro zobrazování v aplikaci VRUT. Mimo jiné také popisuje teorii zobrazování 3D scén v reálném čase a zabývá se reprezentací modelů a jejich materiálů.

Klíčová slova: VRUT; 3D modelování; scéna rozsáhlého exteriéru; zobrazování v reálném čase

Školitel: Ing. Jiří Bittner, Ph.D.

Abstract

This bachelor thesis presents rebuilding of large exterior scene and its rendering in VRUT application. In addition the thesis describes theory about real time rendering and representation of models and materials.

Keywords: VRUT; 3D modeling; large exterior scene; real time rendering

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V roce 2008 zpracoval Václav Kyba v rámci své diplomové práce základ aplikace Virtual Reality Universal Toolkit (VRUT) [Kyb08]. Tato silně modulární aplikace se postupem času rozšiřovala a hojně se používá dodnes ve společnosti Škoda Auto pro nejrůznější vizualizace a simulace. Součástí je také řada zobrazovacích modulů, které mají za úkol zobrazovat prostorová data.

Společnost Škoda Auto aplikaci VRUT využívá mimo jiné také k testům s uživateli a k simulacím provozu. Pro tyto účely vzniklo také malé množství scén se silniční sítí. Kvalita těchto scén je z vizuálního hlediska poměrně omezená. Jejich úpravy jsou velmi náročné jak časově, tak z hlediska provedení, protože je potřeba zajistit, aby dosahovaly dostatečné snímkové frekvence při jejich vykreslování v aplikaci VRUT.

Náplní této práce se stalo provedení úprav rozsáhlého modelu scény s dálničním vedením s důrazem na vizuální stránku. Pro dosažení potřebné kvality bylo potřeba zvolit vhodné postupy jak pro tvorbu jednotlivých modelů, tak pro tvorbu textur. Rozsáhlé scény je navíc potřeba dobře optimalizovat, aby se daly zobrazovat v reálném čase. Mimo optimalizace dané scény bylo také stěžejní správné nastavení zobrazovacích modulů aplikace VRUT.

Bakalářskou práci s podobnou tématikou napsal v roce 2014 Tomáš Kraus [Kra14], ve které se však nezabývá budováním scény rozsáhlého exteriéru, ale tvorbě scény reálného interiéru. Rovněž používá aplikaci VRUT, ale scénu zobrazuje odlišnými moduly.

Kapitola 2

Zobrazování 3D scén v reálném čase

Vykreslování virtuálních prostorových scén v reálném čase je komplexní problematika, která vyvíjí tlak nejen na technické možnosti hardwaru, ale také na jejich tvůrce. K vykreslování v reálném čase se obvykle používá grafická karta. Tato kapitola stručně popisuje postup, jakým je tak učiněno, a doplňuje ho o technické informace ohledně reprezentace virtuálních světů.

2.1 Zobrazovací řetězec

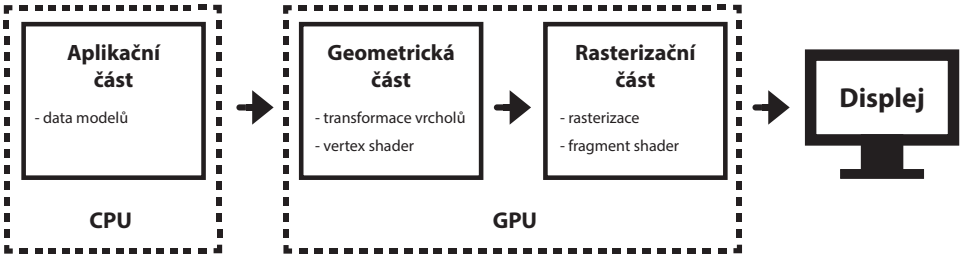
Zobrazovací řetězec pro vykreslování scén v reálném čase se v základu dělí na části aplikační, geometrickou a následně rasterizační [AMHH08]. Každá z těchto částí se může dělit do několika dalších.

Aplikační část je podle názvu prováděna na straně aplikace na CPU. V aplikační části se běžně počítají například kolize, animace, fyzikální výpočty a podobně. Jedním z nejdůležitějších úkolů aplikační části je sycení geometrické fáze geometrickými primitivy (body, úsečky, případně trojúhelníky) [AMHH08].

V geometrické části se běžně provádí výpočty projekce a různých transformačních operací. Zjednodušeně řečeno řídí jak, kde a jaké objekty se vykreslí. Tato část se typicky provádí na straně grafického procesoru. V rámci této fáze se zmíněné transformační operace provádí na jednotlivých vrcholech gometrických primitiv, kterými geometrickou část nasytila fáze aplikační. Tyto operace se provádí prostřednictvím vertex shaderu, programu, který se spouští na grafické kartě a je v režii programátora [AMHH08].

Rasterizační fáze nakonec vykreslí výsledný obrázek do pixelové mříže. Rasterizována jsou postupně všechna primitiva z předchozí fáze a na každý rasterizovaný pixel se spouští takzvaný fragment shader. V tomto shaderu se typicky provádí výpočty nasvícení daného bodu. Fragment shader je jedním z nejmocnějších nástrojů programátora grafiky pro ovlivnění podoby samotného výstupu. Výsledná barevná hodnota pro každý zpracovaný pixel je následně uložena do snímkového bufferu. Po kompletním zpracování dat je buffer zobrazen na monitoru [AMHH08].

Celý proces zobrazovacího řetězce ilustruje obrázek 2.1.



Obrázek 2.1: Zobrazovací řetězec

2.2 Reprezentace geometrických dat

Chceme-li vyrobit a zobrazit prostorové virtuální modely, je potřeba vědět, pro jaké účely budou použity. Tvar těchto objektů lze reprezentovat v základu dvěma způsoby: hraničně, nebo objemově.

2.2.1 Hraniční reprezentace

Nejpoužívanějším typem reprezentace prostorových těles pro účely syntézy obrazu v reálném čase je reprezentace pomocí popisu povrchu. Síla tohoto způsobu spočívá ve snadném a rychlém zpracování samotných dat během vykreslování.

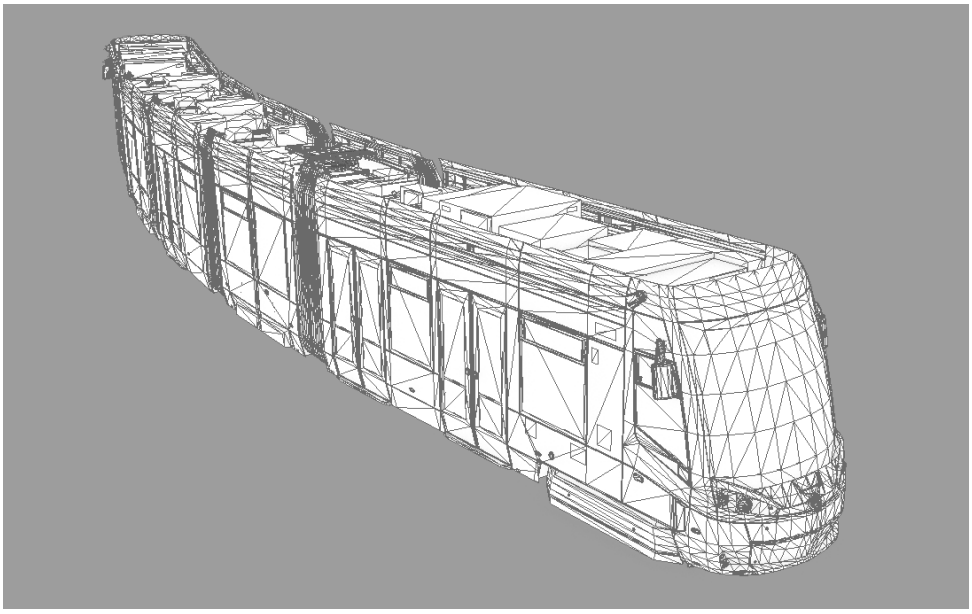
Nejčastěji se setkáváme s polygonálním modelováním, kdy je povrch tělesa aproximován pomocí trojúhelníkové geometrie. Tento systém profituje hlavně ze své jednoduchosti samotných povrchových dat. Ta jsou popsána pomocí setu souřadnic pro pozici, normálu a případně mapovacích souřadnic pro textury. Všechna tato data pak mohou přímo sloužit jako vstup do zobrazovacího řetězce a není potřeba je dále upravovat. Nevýhodou tohoto systému je ovšem velký objem dat. Příklad takto vytvořeného modelu prezentuje obrázek 2.2.

Velikost dat částečně řeší metoda modelování pomocí NURBS ploch. Tyto křivkové plochy jsou reprezentovány pomocí řídících bodů, které rovněž mohou sloužit jako přímý vstup do zobrazovacího řetězce. Plochy NURBS však vyžadují další zpracování během samotného vykreslování[1], které se tím pádem zpomaluje. Problematičtější je také mapování textur. Oblé plochy jsou ale popsány matematicky, čímž dosahuje tato metoda vyšší přesnosti než polygonální modelování. Další výhodou je možnost řídit kvalitu polygonální aproximace. Můžeme tak pružně řídit jemnost sítě na základě výkonu výpočetního stroje.

2.2.2 Objemová reprezentace

Dalším typem reprezentace je reprezentace pomocí objemu. Ten lze vytvářet například pomocí konstruktivní geometrie, která s využitím stromu, jenž

[1] V zásadě jsou NURBS plochy za běhu dynamicky aproximovány polygonální sítí a následně se vykreslují stejně jako klasická polygonální geometrie.



Obrázek 2.2: Příklad hraničně definovaného polygonálního modelu

je tvořen primitivy, transformacemi a eukleidovskými operacemi, postupně popisuje celý objekt.

Metoda objemové reprezentace není příliš vhodná pro zobrazování v reálném čase. Vykreslování takové geometrie je nejčastěji řešeno metodou ray-tracingu, která sice nabízí snadné řešení mnoha grafických problémů, pro svou omezenou rychlost je však v plné podobě nevhodná pro účely zobrazování v reálném čase.

Ačkoliv není způsob objemové reprezentace optimální pro účely rychlého vykreslování, velmi dobře se chová například k matematickým výpočtům kolizí. Proto jsou často polygonální modely doplněny o hrubou aproximaci pomocí konstruktivní geometrie.

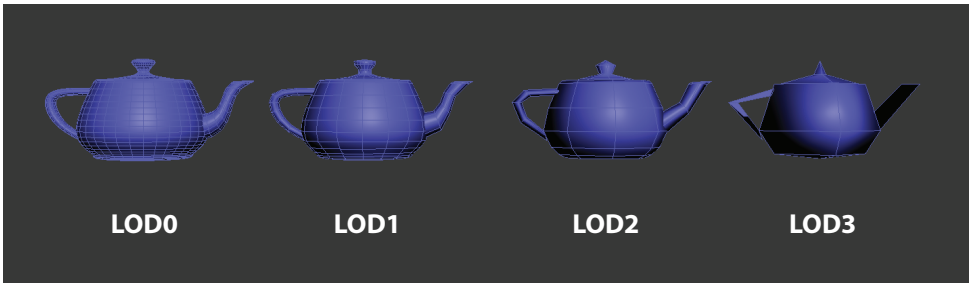
2.2.3 Řízení detailu

Množství geometrických dat vstupujících do geometrické fáze zobrazovacího řetězce (viz kapitola 2.1) není neomezené. Množství polygonů významně ovlivňuje snímkovou frekvenci celého zobrazovacího procesu. Kvůli tomuto omezení musíme při výrobě modelů určených pro zobrazování v reálném čase dodržovat střídmý počet polygonů ve scéně. U rozsáhlých scén nám ovšem plošně nízký počet polygonů u zobrazovaných objektů přestává stačit. Další plošná optimalizace sítí by sice přinesla výkonnostní benefit, výrazně by však poškodila vizuální kvalitu scény.

Elegantním řešením je takzvané LOD, neboli level of detail (úroveň detailu). Tato technika zaměňuje objekty vzdálenější od kamery za jejich zjednodušené verze. LOD počítá s faktem, že pozorovatel nebude schopen u vzdálenějších objektů rozeznat rozdíl mezi dvěma úrovněmi.

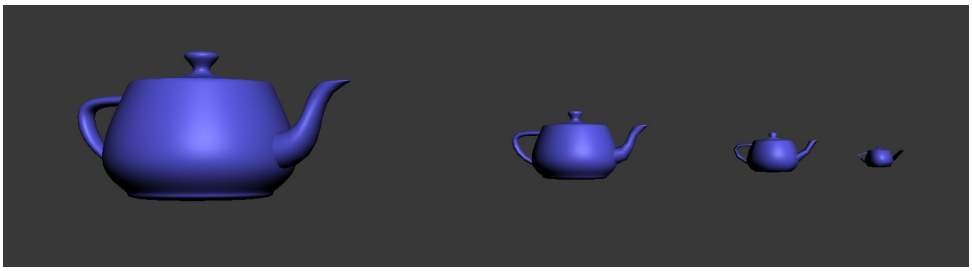
Hendikepem techniky LOD je však nutnost zpracování modelů v několika

variantách (množinu úrovní názorně prezentuje obrázek 2.3), což může být časově náročné. Existují sice dynamické metody řízení detailu (například teselace NURBS ploch), tyto metody ovšem nejsou všemocné a mnohdy dochází u složitých modelů k nežádoucím artefaktům [AMHH08].



Obrázek 2.3: LOD úrovně polygonálního modelu

Použitím LOD se však výrazně zvyšuje vizuální kvalita scény, protože můžeme věnovat více výkonu blízkým objektům, a tak zlepšit jejich propracovanost. Použití LOD úrovní ve scéně ukazuje obrázek 2.4. Další výhodou je minimalizace vykreslování vzdálených polygonů, které jsou v rastrové síti po provedení projekce velmi drobné. Jejich dopad na výkonnost dobře dokumentoval Christophe Riccio [Ric14].



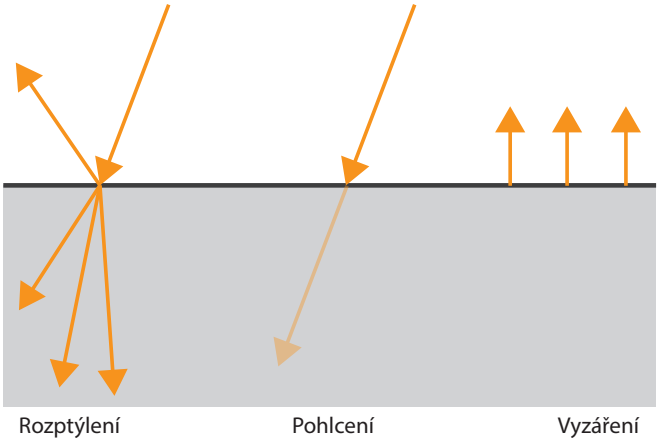
Obrázek 2.4: Příklad použití LOD úrovní ve scéně

2.3 Reprezentace materiálů

Pro popis vzhledu povrchu těles využíváme materiály. Každý materiál může být definován celou řadou parametrů, například: shader programy, texturami, parametry hladkosti povrchu a jinými. Tyto parametry pak přímo ovlivňují způsob, jakým povrch interaguje se světlem, a tím i výsledný vzhled vykresleného obrazu.

Interakci materiálu se světlem můžeme rozdělit v zásadě na 3 typy: rozptýlení, pohlcení a vyzáření [AMHH08]. Všechny 3 typy ilustruje obrázek 2.5.

K rozptýlení světla dochází, když světlo narazí na rozdíl v prostředí (například prostup vzduchu do skla). Rozptýlení v zásadě nemění množství světla, pouze jeho směr [AMHH08].

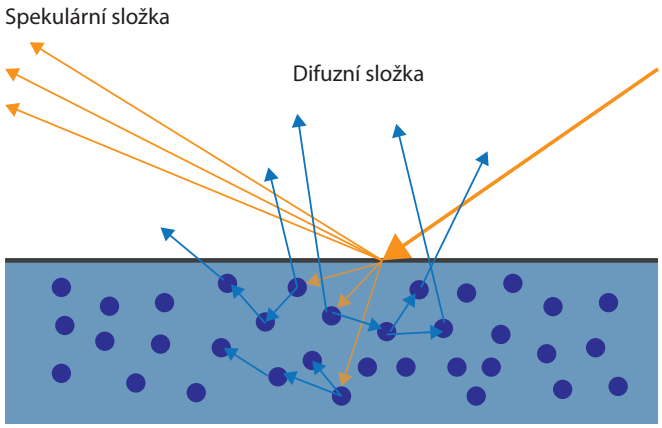


Obrázek 2.5: Typy světelných interakcí

Pohlcení světla je jev, kdy materiálem projde pouze část světelných paprsků. Pohlcení ovlivňuje pouze množství světla, nikoliv jeho směr [AMHH08].

Vyzáření, jak už je z názvu patrné, naopak světlo generuje.

Mluvíme-li o chování světla na rozhraní vzduchu a povrchu neprůhledného objektu, můžeme logicky interakci světla dále rozdělit na světlo odražené směrem zpět do vzduchu a na světlo rozptýlené směrem do objektu. Oba typy chování mají diametrálně odlišný charakter. Světlo rozptýlené směrem do objektu má například odlišný směr a šířku rozptýlení, dále se pak liší v barvě odraženého světla, protože světlo je materiálem i částečně pohlceno, než je následně odraženo zpět směrem ven z objektu. Dále budeme světlo odražené na povrchu označovat jako spekulární složku a světlo, které prošlo objektem, jako složku difuzní [AMHH08]. Charakter obou složek odrazu lze vidět na obrázku 2.6.



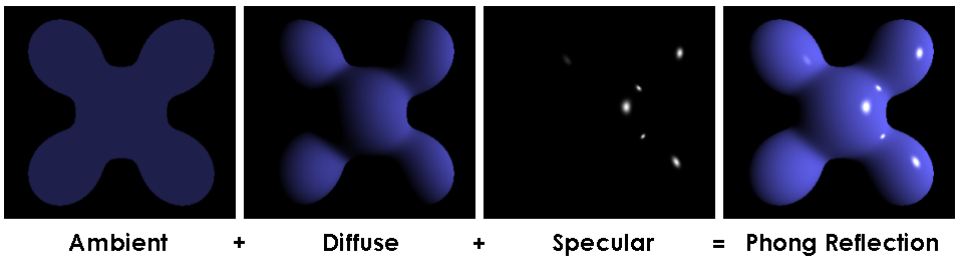
Obrázek 2.6: Difuzní a spekulární složka odraženého světla

2.3.1 Světelné modely

Údělem shaderů je implementovat ono výše zmíněné chování světla na povrchu daného materiálu. K tomu slouží rovnice světelných modelů, které na základě vstupního světla řeší sílu a podobu spekulární a difuzní složky světla odraženého. V souvislosti se světelnými modely můžeme narazit také na pojem BRDF, neboli bidirectional reflectance distribution function. Funkce řeší způsob, jakým je světlo odraženo na základě dvou vektorů (bidirectional): vektoru směřujícího ke světlu a vektoru směru odraženého světla [AMHH08].

Phongův osvětlovací model

Phongův osvětlovací model je jeden z nejzákladnějších a nejjednodušších modelů. Skládá dohromady kromě spekulární a difuzní složky také složku ambientní. Ambientní složkou můžeme rozumět světlo o nízké intenzitě, které se několikrát odrazilo od okolního prostředí a nakonec dopadlo na zobrazovaný povrch. Ambientní složka není závislá na směru, díky ní nejsou plochy odvrácené od světelných zdrojů zcela černé, ale jsou nasvíceny právě barvou ambientního světla. Je však potřeba zdůraznit, že ambientní složka je pouze hrubou aproximací reality. Jednotlivé složky a jejich sestavení dobře ilustruje obrázek 2.7.



Obrázek 2.7: Složky a sestavení Phongova osvětlovacího modelu (autorem obrázku je Brad Smith)

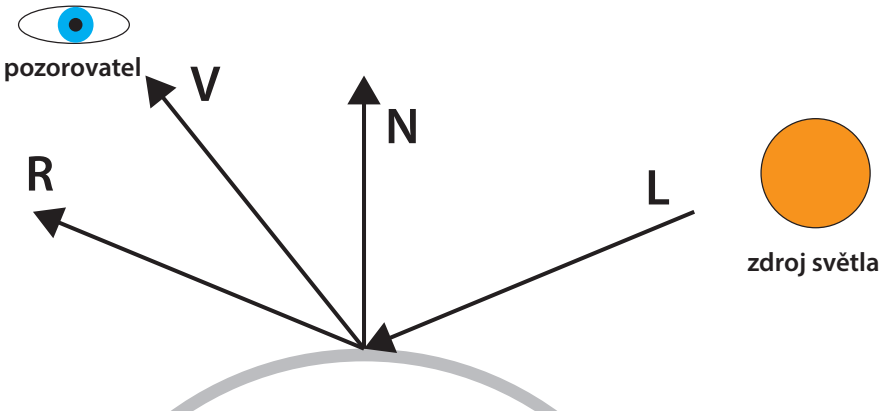
Ambientní složku lze vyjádřit jako součin ambientní hodnoty materiálů s ambientní hodnotou světelného zdroje $m_a l_a$.

Složka difuzní je dána skalárním součinem vektorů $\mathbf{L_l} \cdot \mathbf{N}$ vynásobený difuzní barvou materiálu a světelného zdroje $m_{diff} l_{diff}$.

Složka spekulární je tvořena pomocí skalárního součinu vektorů $\mathbf{R_l} \cdot \mathbf{V}$ umocněného sílou lesku m_{shine}, který je dále vynásoben spekulární barvou světelného zdroje a materiálu $m_{spec} l_{spec}$.

Význam uvedených vektorů dobře prezentuje obrázek 2.8. Konečná rovnice Phongova modelu má tvar:

$$l_{out} = m_a l_a + \sum_{l \in lights} ((\mathbf{L_l} \cdot \mathbf{N}) m_{diff} l_{diff} + (\mathbf{R_l} \cdot \mathbf{V})^{m_{shine}} m_{spec} l_{spec})$$



Obrázek 2.8: Významné vektory ve Phongově osvětlovacím modelu

BRDF modely

Přesná definice BRDF modelu je podíl diferenciální odražené záře $dL_0(\mathbf{v})$ a diferenciálu ozáření povrchu $dE\mathbf{l}$ 2.1 [AMHH08]. Vstupem funkce je vždy vektor od ozařovaného bodu k pozorovateli $\mathbf{v}$ a vektor směřující od tohoto bodu ke světlu $\mathbf{l}$.

$$f(\mathbf{l}, \mathbf{v}) = \frac{dL_0(\mathbf{v})}{dE\mathbf{l}} \tag{2.1}$$

Pro BRDF modely je typické, že respektují (na rozdíl od původního Phongova osvětlovacího modelu) zákon o zachování energie a reciprocitu. Teorie BRDF modelů je obsáhlá a dále se jí v této práci zabývat nebudeme. Více informací lze najít ve zdroji [AMHH08].

2.3.2 Průhledné materiály

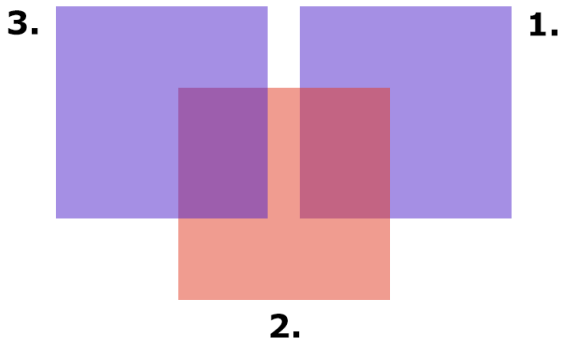
V reálném světě neexistují pouze neprůhledné povrchy. Zobrazování tabulek skla a jiných průhledných materiálů s sebou nese řadu problémů. Základem pro výpočet barevné hodnoty pixelu s příspěvkem poloprůhledného materiálu je míchací funkce [AMHH08]:

$$c_0 = \alpha_s c_s + (1 - \alpha_s) c_d \tag{2.2}$$

V našem případě hodnota c_s definuje barvu průhledného materiálu, α_s je pak parametr průhlednosti tohoto materiálu[2]. c_d charakterizuje barvu pozadí. Z uvedeného vztahu je patrné, že záleží na pořadí uvedených pixelů. Do problému se dostáváme, pokud vykreslujeme 3 objekty A, B, C, kde A je neprůhledný objekt a B a C jsou objekty průhledné. Pokud například provedeme vykreslení v pořadí A, C, B, bude výsledek jiný než v případě pořadí A, B, C (viz obrázek 2.9). Abychom provedli vykreslení takových

[2] 1 = zcela neprůhledný, 0 = zcela průhledný

objektů správně, je potřeba je vykreslovat od nejvzdálenějšího po nejbližší. Takovéto seřazení je však nutné provést ještě v aplikační fázi zobrazovacího řetězce (viz kapitola 2.1) na straně procesoru. Kromě potřebných operací navíc pro míchání pixelů je ono seřazení dalším výkonnostním zatížením samotného zobrazování.



Obrázek 2.9: Odlišné pořadí vykreslování průhledné geometrie generuje rozdílné výsledky. Čtverce vykreslené v kroku 1 a 3 mají stejnou barvu. Všechny 3 čtverce mají alfa kanál hodnoty 0.5.

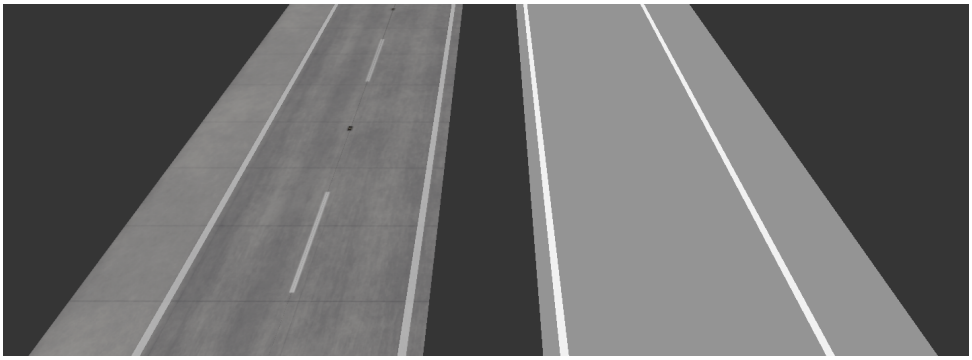
Parametr α je normalizovaný, proto ho lze snadno uchovat jako součást bitmapových textur. Hodnotu alfa kanálu však lze zpracovat i bez potřeby použití míchací funkce. Učinit tak můžeme ale pouze v případě, kdy jsou hodnoty alfa kanálu extrémní (obsahují pouze hodnoty 0 nebo 1). Takové materiály označujeme jako maskované. V případě nulové alfy je zobrazovaný fragment jednoduše zahozen. Díky tomu není potřeba geometrii dále řadit. Maskované materiály lze vhodně využít například při tvorbě modelů zeleně nebo modelu zábradlí, kdy není potřeba jednotlivé mříže modelovat polygonálně, a tím ušetřit výkon.

2.3.3 Textury

Definice materiálů pomocí konstantních hodnot je velmi omezujícím faktorem. V reálném světě není povrch objektů uniformní. Abychom byli schopni definovat proměnlivý charakter povrchu objektů pomocí konstantních hodnot, museli bychom objekt rozdělit na mnoho částí a každé přiřadit originální materiál. Logicky by takové řešení bylo velmi neefektivní. Elegantním řešením je použití obrázků v podobě textur místo konstantních hodnot (viz obrázek 2.10).

Projekce textur na geometrii

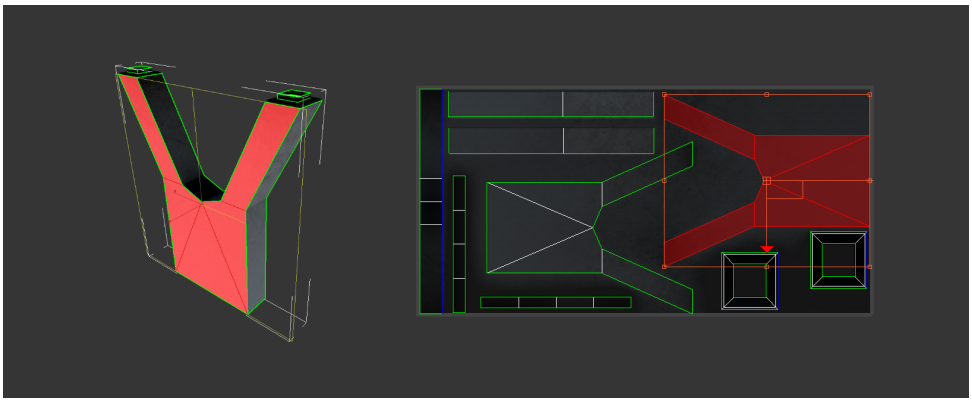
Abychom mohli použít dvourozměrná obrazová data na povrchu trojrozměrné geometrie, musíme definovat takzvanou mapovací funkci. Tu lze definovat jako projekci z 3D prostoru geometrie do 2D prostoru textury. Výsledkem



Obrázek 2.10: Příklad využití textur (napravo bez textur)

je vektor o dvou složkách, který bodu v prostoru přiřazuje bod textury. 2D mapovací souřadnice nesou typicky označení *uv* a jejich báze je ortonormální.

U polygonálních modelů je běžnou praxí tyto projekce propočítat předem v podobě uv seznamů, kdy je každému vrcholu geometrie přiřazena hodnota uv souřadnic. Toto řešení nabízí perfektní kontrolu nad projekcí a urychluje zobrazovací proces. Při něm se předpočítané uv souřadnice už pouze interpolují mezi jednotlivými vrcholy povrchu. Objekt může disponovat i více než jedním seznamem uv souřadnic. Příklad mapování dobře ilustruje obrázek 2.11.



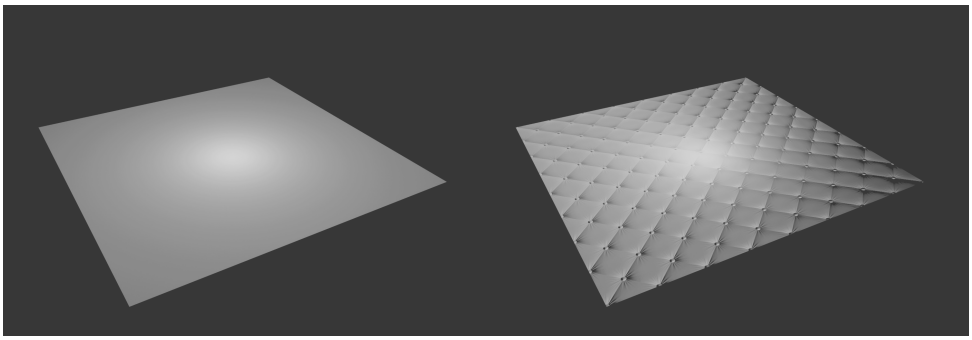
Obrázek 2.11: Mapovací souřadnice objektu mostního pilíře

Další možnosti textur

Textury nemusí být zdaleka použity pouze pro specifikaci difuzní barvy povrchu. Pomocí textury lze například řídit i sílu spekulárního odrazu. Je však třeba připomenout, že hodnoty RGB složek souboru obrázku jsou omezeny maximální hodnotou, proto lze do textur uložit pouze normalizovanou veličinu.

Častým příkladem, kdy jsou do textur uložena i neobrazová data, jsou takzvané normálové mapy. Normálová mapa využívá tří RGB složek k uložení normálového vektoru. Připomeňme si, že normálový vektor je stěžejním parametrem, který ovlivňuje nasvícení daného bodu. Díky normálovým mapám můžeme tedy vizuálně modelovat větší nerovnosti povrchu, které by byly

příliš drahé pro geometrické vyjádření pomocí polygonální sítě. Díky tomu získáváme mnohem detailnější výsledek bez navýšení složitosti geometrie. Příklad efektu normálové mapy prezentuje obrázek 2.12.



Obrázek 2.12: Iluze nerovností pomocí normálové mapy

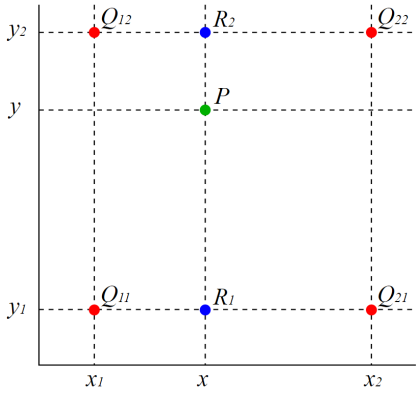
Setkat se můžeme také se světelnými mapami. Tyto textury drží předpočítané informace o osvětlení a zásadním způsobem urychlují vykreslování. Světelné mapy jsou obvykle mapovány svým vlastním seznamem uv souřadnic.

Filtrování textur

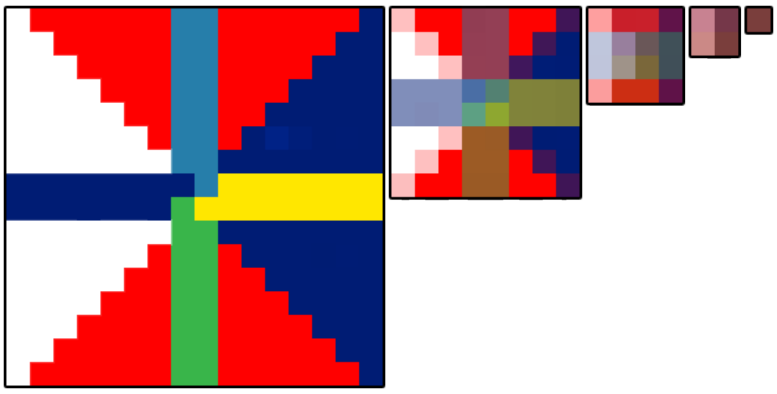
Jelikož jsou obrazová data spojitým signálem, nemusíme být omezeni pouze na hodnotu jednotlivých pixelů. Jestliže souřadnice uv zasáhne prostor mezi 2 pixely, lze hodnotu vypočítat pomocí těchto sousedících pixelů. Tento proces se nazývá filtrování a je velmi výhodným, pokud texturujeme plochu o větší pixelové velikosti než je rozlišení samotné textury (projekční velikost pixelu je větší než velikost pixelu textury) [AMHH08]. Metod filtrování je celá řada, pro názornost si popíšeme 2 přístupy:

- nejbližší soused - nefiltruje se, pouze se přečte hodnota nejbližšího pixelu
- bilineární interpolace - interpoluje se v okolí bodu (v zásadě mezi čtyřmi hodnotami pixelů). Proces ilustruje obrázek 2.13.

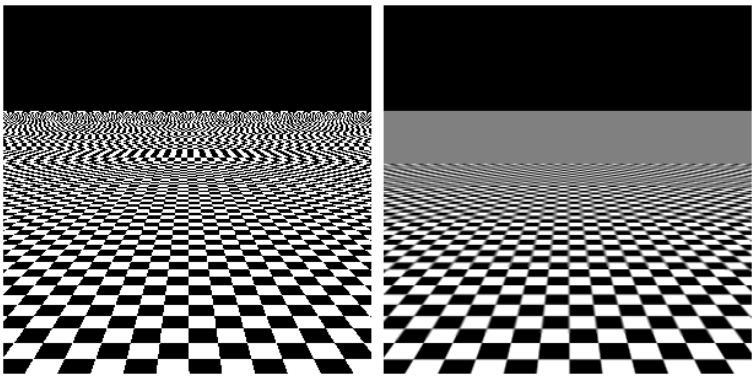
Pokud však do projekční velikosti pixelu spadá více pixelů textury, může docházet k nepříjemnému zrnění, protože pohyb o zlomek pixelu v prostoru obrazovky může radikálně změnit výsledný vzorek textury. Tento jev se projevuje zrněním vzdálenějších objektů. Řešením jsou takzvané mipmapové úrovně. Každé textuře je vytvořen set zmenšených kopií, u kterých je postupně textura zmenšována na poloviční velikost strany, dokud není velikost textury pouze jediný pixel. Charakter mipmap ilustruje obrázek 2.14. Díky přítomnosti mipmap lze následně u vzdálenějších objektů číst texturu z její nižší mipmapové úrovně, a tak se vyvarovat zmíněnému zrnění jako na obrázku 2.15 [AMHH08].



Obrázek 2.13: Získání barevné hodnoty bodu pomocí interpolace sousedních pixelů. Body Q určují hodnotu čtyř nejbližších pixelů v okolí bodu P. Body R vznikly interpolací příslušných dvojic bodů Q. Bod P vzniká lineární interpolací bodů R. (převzato z wikipedia.org)



Obrázek 2.14: Mipmapové úrovně textur (převzato z webglfundamentals.org)



Obrázek 2.15: Zrnění vzdálených ploch bez přítomnosti filtrování a napravo s využitím mipmap (autorem obrázku je Wojciech Mula)

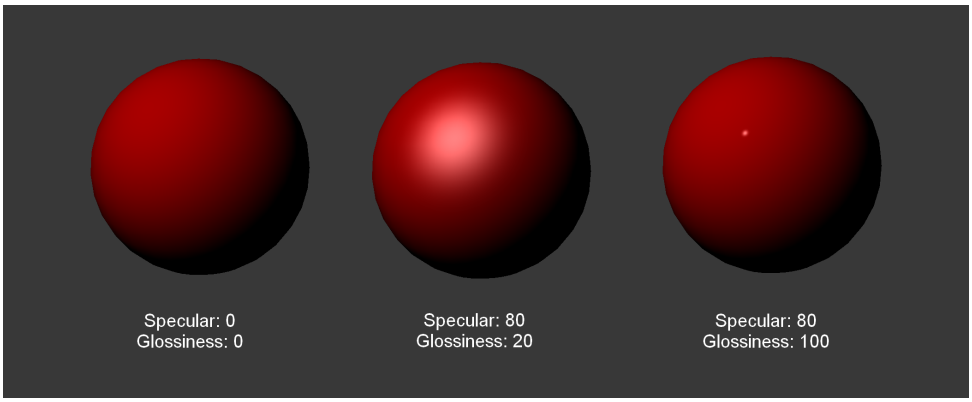
Složky materiálů z pohledu vzhledu

Aby mohly světelné modely produkovat výstup, je potřeba jim přiřadit vhodná vstupní data. Například k tomu, abychom dokázali sytit phongův model,

potřebujeme o materiálu vědět následující parametry:

- difuzní barvu (barva povrchu, kdy je osvětlen 100% bílým světlem [Gam17a])
- barvu spekulárního odrazu (ve skutečnosti nemá většina povrchů barevný odraz, výjimkou jsou například kovy)
- sílu lesku
- barvu emisní složky (barva, kterou povrch vyzařuje).

Ačkoliv jsou parametry jasně definovány, jejich použití je poměrně abstraktní a velmi snadno bude výsledek vizuálně nepřirozený. Často je obtížné nastavit fyzikálně korektní barvu a sílu spekulární odrazu. Efekt rozdílných hodnot materiálu ilustruje obrázek 2.16.



Obrázek 2.16: Příklady různých materiálů pod Phongovým osvětlovacím modelem

Složky materiálů z pohledu fyzikálního charakteru

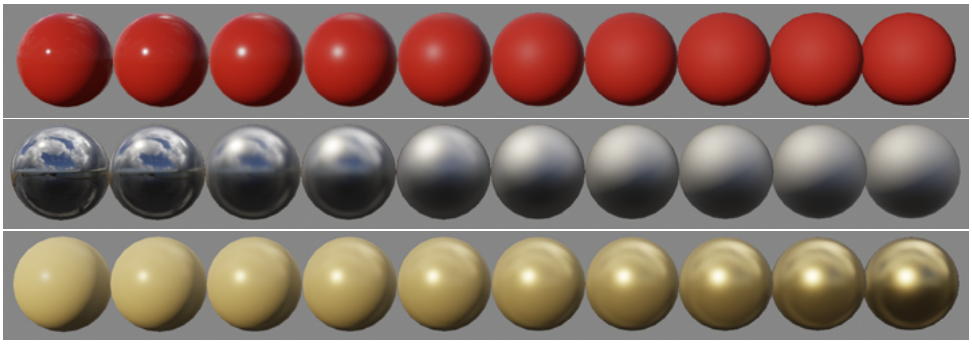
Nejnovějším přístupem vykreslování v reálném čase je PBR, neboli Physically-Based Rendering. Tento způsob zobrazování přinesl velký posun k fotorealismu, protože změnil způsob, jakým je na materiály nahlíženo. Způsob definice je v porovnání s předchozí kapitolou mírně odlišný. Složky materiálu jsou zaměřeny spíše na fyzikální charakter materiálu než na jeho vizuální vlastnosti, které z jeho fyzikálního charakteru plynou. Teorie ohledně PBR je rozsáhlá a použití fyzikálně založených materiálů je svázáno s použitím dané BRDF.

Unreal Engine 4 například definuje materiál těmito složkami [Gam17b]:

- base color jakožto barvy povrchu
- roughness, která popisuje hrubost povrchu na úrovni mikrogeometrie
- metallness, která specifikuje, zda je materiálem kov či nekov. Hodnoty tohoto kanálu jsou pro většinu materiálů extrémní (0 či 1).

- specular, který určuje sílu reflexe materiálů u dielektrik.

Efekt složek ilustruje obrázek 2.17.



Obrázek 2.17: Vliv materiálových složek ve fyzikálně definovaných materiálech Unreal enginu 4. Shora: proměnná hodnota roughness u nekovů, proměnná hodnota roughness u kovů, proměnná hodnota metallness; vždy v intervalu 0 až 1 (převzato z docs.unrealengine.com)

Z pohledu definice materiálů můžeme narazit na četné odlišnosti napříč enginy. Unity engine například definuje místo roughnes její převrácenou hodnotu specular smoothness, base color pak pojmenovává jako albedo [Kö15].

Velkou výhodou tohoto systému je jeho intuitivnost. Tento způsob z velké částí minimalizuje riziko vytvoření fyzikálně nekorektních materiálů. Všechny materiálové vlastnosti jsou navíc normalizované, takže se dají velmi dobře reprezentovat pomocí jedné z barevných složek bitmapových textur. Tento fakt umožňuje výhodně spojit hodnoty metalness a roughness do jedné textury a snížit tak objem dat v paměti. Další výhodou je konstantní charakter napříč různými světelnými podmínkami.

2.3.4 Zapékání dat do textur

Zapékání je v 3D grafice označován proces, kdy jsou povrchová data objektu zapsána do textur. Máme-li například objekt v komplexní scéně, můžeme využít pokročilého rendereru, který vypočítá osvětlení pro povrch objektu a jeho hodnotu zapíše do textur[3]. Zapékat však můžeme celou řadu informací. Motivací tohoto způsobu tvorby textur je jeho rychlost v porovnání s ručním kreslením.

V praxi se setkáme i s technikou, kdy je vytvořen detailní model s velmi jemnou polygonální sítí, která by byla pro zobrazování v reálném čase neoptimální. K této geometrii se vytvoří sekundární verze modelu, která hrubě aproximuje její povrch. Následně lze data detailní geometrie projekčně zapéct do textur geometrie zjednodušené. Takovým postupem se běžně generují například normálové mapy.

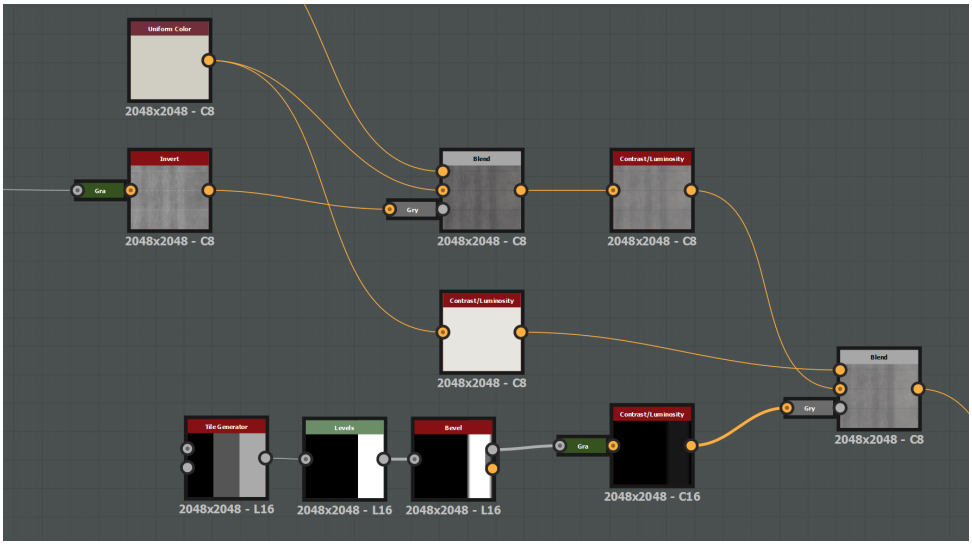
[3]Textuře, která nese předpočítané informace o osvětlení, říkáme světelné mapy.

2.3.5 Procedurálně generované materiály

Poměrně novým přístupem jsou procedurální materiály. Často se setkáme se situací, kdy je ve scéně obsaženo mnoho materiálů podobného charakteru, například několik druhů obkladových dlaždic. Vytvoření originálních textur pro každý druh dlaždic je časově i paměťově náročné. Procedurální materiály tento problém řeší.

Procedurálně generovaným materiálem můžeme rozumět takový materiál, který je definován posloupností matematických operací z určeného základu. Použijeme-li náš příklad dlaždic, základem můžeme rozumět generátor čtvercové sítě, na kterou je následně aplikována barva pomocí operace prolnutí. Chceme-li vygenerovat texturu různě barevných materiálů, stačí nám změnit pouze jediný parametr.

V praxi jsou procedurální materiály mnohdy sestavovány pomocí značně rozvětvených grafů. Ukázkou takového grafu je obrázek 2.18.



Obrázek 2.18: Výřez grafu procedurálně generovaného materiálu betonové vozovky v programu Substance Designer

Kapitola 3

Zmapování softwarových nástrojů

3.1 VRUT

Aplikace virtual reality universal toolkit vznikla v rámci diplomové práce Václava Kyby [Kyb08], ve které navrhl samotné jádro a systém této silně modulární aplikace. Jádro spravuje moduly aplikace a grafická data. Řídí také tok událostí a dohlíží na správný běh programu. Samotné jádro je ovšem bez modulů nepoužitelným nástrojem.

3.1.1 Moduly aplikace VRUT

Jak již bylo zmíněno, VRUT je silně modulární aplikace. Moduly se funkčně dělí do čtyř kategorií:

- moduly pro import a export geometrických dat
- zobrazovací moduly
- manipulační moduly
- ostatní moduly.

V rámci této práce jsem využil širokou škálu nabízených modulů. Seznam použitých modulů a jejich členění ilustruje obrázek 4.9.

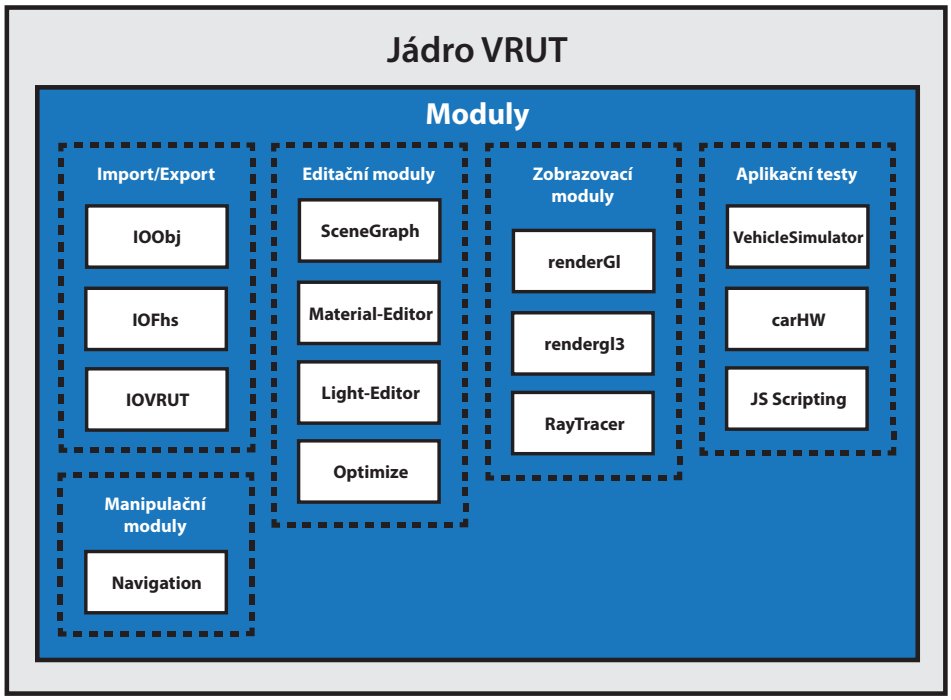
Moduly pro import a export

Pro převod scény jsem využil funkce modulů IOobj, IOfhs a IOvrut.

První jmenovaný provádí převod do formátu Wavefront OBJ, který jsem si zvolil jako hlavní tunel mezi aplikací VRUT a 3Ds Max, kterou využívám jako hlavní editační software pro tvorbu geometrie (viz kapitola 4.2).

Modul IOfhs překládá grafický formát fhs, ve kterém se původně scéna dálnice nacházela. Formát fhs rovněž nabízí o něco širší možnosti popisu geometrie než obj, ale jeho neexistující podpora pro 3Ds Max znemožnila jeho přímé použití.

V neposlední řadě IOvrut obstarává import a export nativního formátu aplikace. Jeho načítání je nejrychlejší a nabízí plnou podporu pro reprezentaci



Obrázek 3.1: Schéma použitých modulů aplikace VRUT

scén v aplikaci VRUT (bavíme se například o kompletní podpoře veškerých uzlů grafu scény, kompletní množiny parametrů materiálů a podobně), proto byl zvolen jako konečný formát, do kterého bude scéna v závěru vyexportována.

Zobrazovací moduly

V současné době aplikace VRUT nabízí několik zobrazovacích modulů. V rámci této práce budu pracovat s moduly renderGl, RayTracer a renderGl3.

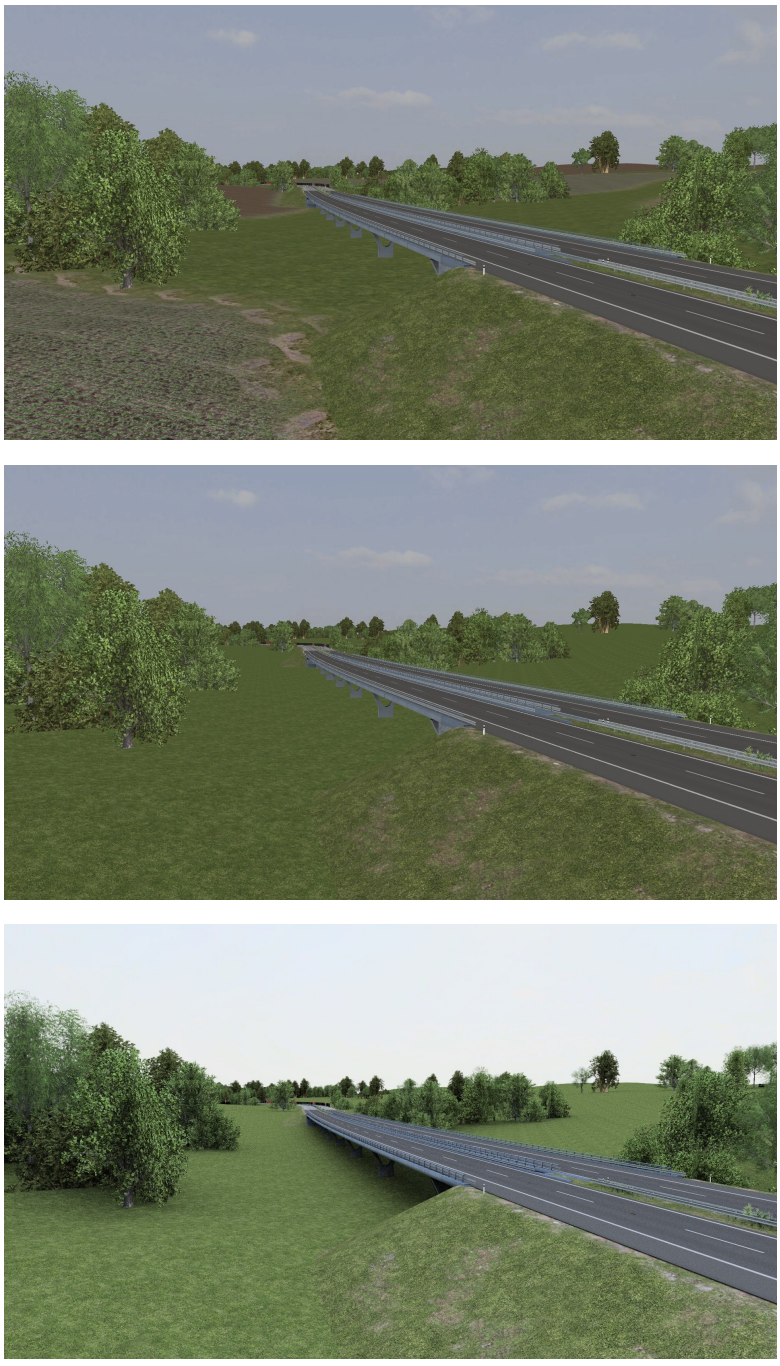
Modul renderGl vznikl spolu s jádrem jako jeden z prvních modulů v rámci diplomové práce Václava Kyby [Kyb08], jedná se proto o nejstarší zobrazovací modul ze tří jmenovaných. Pro vykreslování využívá OpenGL API. Jeho vizuální schopnosti jsou ovšem velmi omezené. Podporuje pouze základní dopředné vykreslování[1]. Nemá implementovány žádné průchody pro vykreslování stínů.

RayTracer je modul, který zobrazuje scénu pomocí metody raytracing a pathtracing. Tato skutečnost ho činí vizuálně nejpokročilejším. Implementačně se jedná o velmi rychlý renderer, který dokáže běžet v omezené kvalitě i v reálném čase.

RenderGl3 je dalším modulem, který podporuje vykreslování v reálném čase. Jeho vývoj však stále trvá. V současné době sice disponuje podporou pro víceprůchodové vykreslování, stále však nepodporuje například vykreslování dynamických stínů. V této chvíli produkuje téměř shodný výstup jako

[1] Vykreslovací postup, při kterém je postupně celá scéna vykreslena přímo do obrazového bufferu.

modul renderGl. Výsledky naměřené tímto modulem jsou proto převážně experimentálního charakteru.



Obrázek 3.2: Porovnání výstupů zobrazovacích modulů (odshora: rendergl, rendergl3, raytracer)

V minulosti vznikl také modul RenderGl2, jehož autorem je Daniel Šimek [Ši13]. Tento modul podporoval víceprůchodové vykreslování a implementoval

techniku odloženého vykreslování[2]. Následný vývoj jádra aplikace VRUT však zapříčinil nekorektní chování tohoto modulu, proto nebyl do testů zařazen.

Další použité moduly

Mimo výše uvedené moduly jsem používal manipulační modul Navigation, který obstarává pohyby kamery a nastavení projekčních rovin.

Z obecných modulů jsem využil editačních modulů SceneGraph, který umožňuje měnit a strukturovat graf scény, Material-Editor, který nabízí funkce pro nastavení materiálů ve scéně, dále pak modul Light-Editor umožňující nastavení světelných zdrojů a modul Optimize, který jsem využil při závěrečné optimalizaci scény za účelem dosažení co nejrychlejšího vykreslování.

Pro testování jsem využil modulu Vehicle Simulator a carHW, které se zabývají simulací jízdy. V neposlední řadě jsem pak upotřebil také modul JS Scripting, který pro aplikaci VRUT přidává možnost psaní scriptů v jazyce javascript.

3.2 Nástroje geometrického modelování

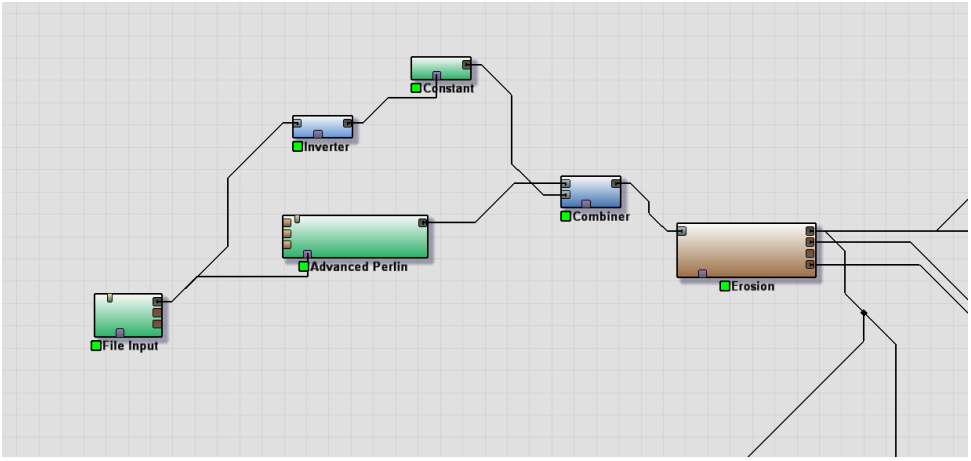
Velkou částí samotné práce je úprava modelu scény. Pro takto rozsáhlý model bylo potřeba dostatečně robustního nástroje.

Jako hlavní editační nástroj geometrie byl zvolen 3Ds Max od firmy Autodesk [Aut17], který nabízí také bezplatnou studentskou licenci. Jedná se o aplikaci s širokou škálou funkcí a dlouhou historií. Disponuje také více než solidní základnou podporovaných formátů pro import, mezi které mimo jiné patří také formát Wavefront obj. Byl zvolen také z důvodu bohatých zkušeností, které s tímto nástrojem mám.

3.2.1 Geometrie terénu

V 3Ds Maxu byl vytvořen prakticky celý model scény s výjimkou terénu. Možnosti tvorby terénu jsou v 3Ds Maxu velmi omezené a pomalé, proto jsem zmapoval možné specializované nástroje. Jedním z nich je Terragen [sof17]. Tento nástroj umožňuje procedurálně generovat reliéf terénu a následně ho zobrazit. Zvolil jsem však alternativní software v podobě programu World Machine [Sch17], který je dostupný také v bezplatné verzi s omezenými funkcemi. Pro mé potřeby se hodila hlavně díky snazší integraci do procesu přestavby scény. Oba nástroje pracují na bázi tvorby grafu, ve kterém jednotlivé uzly představují funkční operátory. Jejich vhodným pospojováním a nastavením pak docílíme potřebného výsledku 3.3.

[2] Vykreslovací postup, při kterém jsou syrová vstupní geometrická data nejdříve vykreslena do bufferů a jejich stínování je odloženo až do doby, kdy celá scéna projde rasterizací. Popis modulu a jeho funkcí je možné najít v diplomové práci Daniela Šimka [Ši13].



Obrázek 3.3: Ukázka grafového přístupu v rozhraní nástroje World Machine

3.3 Nástroje pro tvorbu textur

Důležitým faktorem vizuální kvality jsou z velké části textury. Jejich tvorba je náročná, proto jsem zmapoval dostupné nástroje na poli softwaru, které by mi tvorbu ulehčily a zároveň urychlily.

3.3.1 Procedurálně generované materiály

Průlom nastal s příchodem procedurálně generovaných materiálů, které zásadním způsobem ovlivnily přístup k tvorbě textur. O procedurálních materiálech se lze více dočíst v kapitole 2.3.5. Jejich tvorbu zprostředkovává software Substance Designer firmy Allegorithmic [All17], který je možné získat také pod studentskou licencí. Substance Designer navíc disponuje několika kvalitními bakery, kterými lze zapékat například ambientní zastínění.

3.3.2 Texturování 3D geometrie

Způsob aplikace textur na model je další náročnou disciplínou. Tento problém řeší hned několik nástrojů.

Firma Foundry disponuje nástrojem Mari [Fou17], který umožňuje importovat vlastní model a následně ho pokreslit zvolenými materiály. Tento postup zásadním způsobem usnadňuje tvorbu textur, protože zcela řeší problém návaznosti švů uv mapování u geometrie. Mari je však komerční software a nenabízí se s bezplatnou studentskou licencí.

Pro dokreslování textur jsem proto zvolil software Substance Painter 2 od firmy Allegorithmic [All17]. Ten nejen že je k dispozici s bezplatnou studentskou licencí, ale navíc dobře doplňuje spolu se Substance Designerem ekosystém pro texturování objektů. Jeho funkce jsou stejně bohaté jako u nástroje Mari.

3.3.3 Ladění textur

Ačkoliv jsou výše uvedené nástroje velmi užitečné, stále bylo potřeba místy zasáhnout do samotného souboru textur pomocí některého z široké řady rastrových editorů. Pro drobné doladění textur jsem použil bezplatný PhotoFiltre editor a Photoshop ve zkušební verzi.

Kapitola 4

Realizace úprav scény

Originální model scény zachycuje dálniční vedení dlouhé 38 kilometrů v jednom směru. Okolí bylo zpracováno velmi řídce a vizuálně nepříliš kvalitně. Podmínkou úprav bylo respektování vedení silnice, která měla zůstat nezměněna, aby bylo možné porovnávat dosavadní data z testování na staré scéně s těmi novými. V této kapitole je popsán postup modelové přestavby.

4.1 Postup řešení

Pro modifikaci tak rozsáhlé scény jsem sestavil postup, kterým se budu řídit.

Původní scéna dálnice byla ve formátu fhs. Abych byl schopen použít další programy, bylo potřeba scénu převést do mnohem rozšířenějšího formátu Wavefront OBJ. K převodu jsem použil příslušné moduly aplikace VRUT. Využití a úpravy exportních modulů popisuje sekce 4.2.

Pro samotnou přestavbu scény jsem využil 3Ds Max, ve kterém byla postavena většina geometrie modelů ve scéně s výjimkou větší části stromů. Kompletace scény byla provedena také v programu 3Ds Max.

Materiály jsem vytvářel pomocí nástroje Substance Designer, který jsem využil při tvorbě procedurálních materiálů a zapékání map s informací o ambientním zastínení. Část textur hotových objektů byla vytvořena aplikací materiálů na modely v aplikaci Substance Painter.

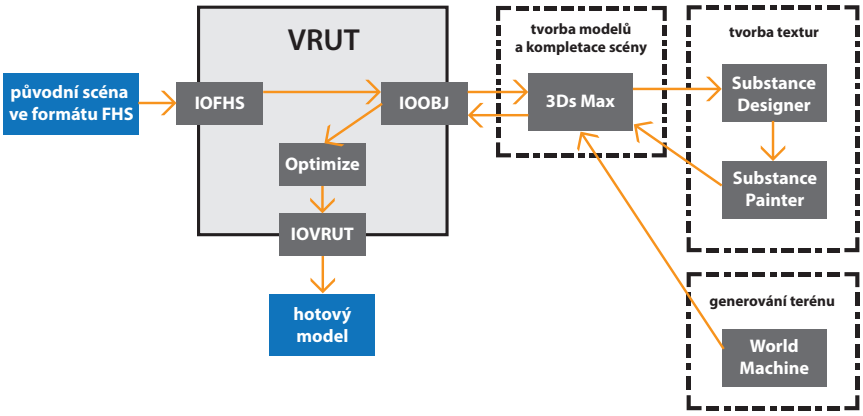
Geometrie terénu byla procedurálně vygenerována v nástroji World Machine. Tvorbě terénu se podrobně věnuje sekce 4.3.

Proces přestavby byl zakončen optimalizací scény pomocí modulu Optimize aplikace VRUT. Následně byla scéna vyexportována do nativního formátu vrut.

Popsaný průběh prací ilustruje obrázek 4.1. Zmíněné nástroje jsou shrnuty v kapitole 3.

4.2 Přenos dat mezi 3D studiem a aplikací VRUT

Aplikace VRUT nabízí širokou škálu exportních modulů. Řada z nich je však implementačně neúplná, případně chybí podpora importu na straně 3Ds Maxu. Optimální volba se jevila v podobě formátu Wavefront OBJ, který



Obrázek 4.1: Postup při úpravě scény

je podporován jak u 3D studia, tak na straně aplikace VRUT. Struktura tohoto formátu navíc podporuje zápis většiny informací, které budou potřeba. Výjimkou je například chybějící podpora pro strukturování scény pomocí grafu.

Modul pro import a export scény s využitím obj formátu byl z velké části funkční a pro naše potřeby použitelný. Chyběla však implementace exportu materiálů, jejíž absence se stala klíčovým nedostatkem celého modulu. Bez exportu materiálů by je bylo nutné redefinovat uvnitř 3D studia a následně je propojit s náležícími modely ve scéně. Manuální řešení by však bylo pracné a časově náročné.

4.2.1 Implementace exportu materiálů

Formát Wavefront OBJ definuje podobu scény pomocí dvou souborů: *.obj, který nese geometrická data, a *.mtl, který popisuje vlastnosti materiálů. Na základě specifikace formátu MTL [RRT95] jsem provedl implementaci zápisu materiálů. Definice jednoduchého materiálu může vypadat například takto:

```
newmtl myMaterial                # material name
    Ka 1.0000 1.0000 1.0000      # ambient
    Kd 1.0000 1.0000 1.0000      # diffuse
    Ks 0.0000 0.0000 0.0000      # specular
    d 1.0000                     # alpha
    illum 2                      # illumination model index
    map_Kd texture.png           # diffuse texture
```

Samotné přiřazení materiálu ke geometrii se specifikuje v souboru *.obj [Wav95]. Zápis vypadá takto:

```
o objectName
g objectName
usemtl myMaterial                # material name
```

4.2.2 Úprava pojmenování polygonálních sítí

S nynější implementací modulu IOObj vyvstal také problém, který se týkal jmen jednotlivých geometrických celků. Původní modul používal k pojmenování názvy uzlů z grafu scény systému VRUT. V něm ovšem mnoho uzlů nedisponovalo vlastním jménem, případně více geometrických uzlů sdílelo shodné pojmenování. Opětovnou definicí geometrie stejného jména dojde při následném importu do aplikace 3Ds Max ke spojení těchto jednotlivých částí do jednoho celku. V našem případě docházelo například k nežádoucímu spojení terénu a objektů, které na něm ležely.

Problém se podařilo vyřešit úpravou zápisu jména geometrie tak, že za vlastní název bylo připojeno i unikátní ID geometrického uzlu ze systému VRUT.

4.3 Tvorba terénu

Scéna svou rozlohou zaujímá poměrně rozsáhlou oblast (přibližně 30 kilometrů na délku a 15 kilometrů na šířku). Vytvořit manuálně terén na takové rozloze by bylo časově velmi náročné a vizuální výsledek by nemusel dosahovat dostatečné věrohodnosti. Proto jsem terén vygeneroval procedurálně pomocí softwaru World Machine [Sch17].

4.3.1 Generování terénu pomocí World Machine

Jak již bylo popsáno v sekci 3.2.1, nástroj World Machine umožňuje procedurální generování terénu a výsledku se dosahuje sestavením grafu.

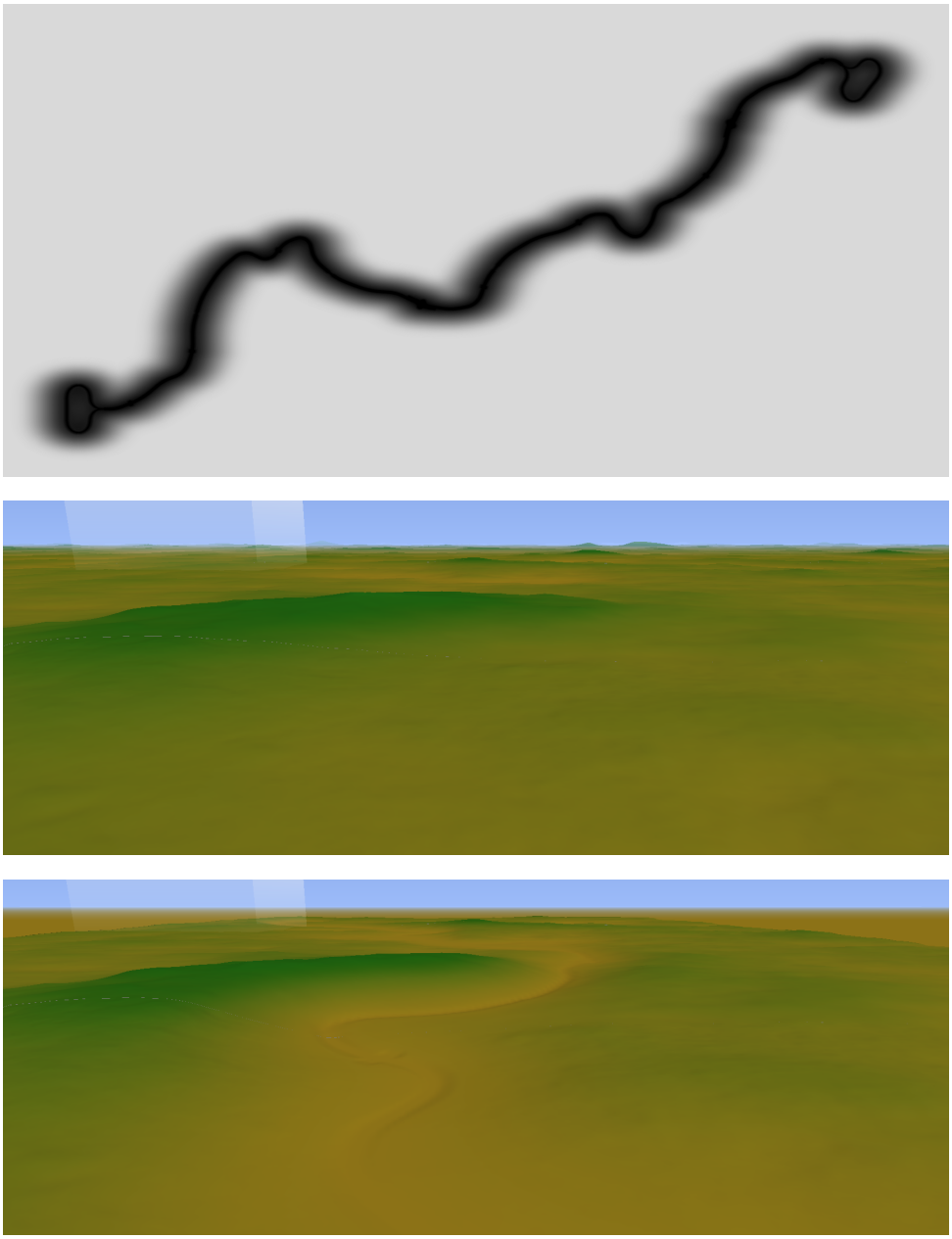
Prvním z hlavních uzlů je generátor samotného terénu. Ten je založen na Perlinově šumu[1] a generuje hrubou podobu terénu. V tomto uzlu jsem nastavil také charakter terénu (jako je například výška hor a jejich sklon) tak, aby přibližně korespondovala s charakterem české krajiny. V dalším kroku jsem využil uzlu eroze, který simuluje zvětrávací procesy a přidává na realističnosti reliéfu.

Výsledek operací jsem vyexportoval pomocí uzlů pro výstup geometrie a uzlů pro výstup obrazových map. Výstupní geometrická data jsou ve formátu obj, pro obrazová data jsem pak zvolil formát png. Pro další zpracování jsem si vyexportoval obrazovou mapu znázorňující sklon povrchu, výšku povrchu a také průběh eroze.

Poslední fází bylo zakomponování vedení silnice do terénu. Vozovka dálnice je koncipována tak, že je v celé své délce v jedné rovině. Tohoto faktu jsem využil a vytvořil si masku pomocí ortogonálního snímku silnice z ptačí perspektivy. Okraje masky jsem rozmazal směrem od silnice. Výsledkem byl obrázek v odstínech šedi, kde v místech vedení vozovky byla černá barva, která postupně přecházela v bílou (viz obrázek 4.2). Masku jsem následně importoval do World Machine, kde jsem ji připojil ke generátoru Perlinova

[1] Autorem je Ken Perlin a algoritmus pro generování šumu publikoval v roce 1985 na konferenci Siggraph [Per85].

šumu. V této chvíli se terén generoval pouze tam, kde byla na masce bílá barva, díky plynulému přechodu do černé pak vznikl také plynulý přechod mezi generovaným terénem a rovinou vozovky (viz obrázek 4.2).



Obrázek 4.2: Maska silničního vedení a její efekt na generovaný terén

World machine ve své bezplatné verzi dovoluje maximální rozlišení obrazových map 513 x 513 pixelů. Abych udržel potřebnou kvalitu terénu, rozdělil jsem jej do šesti oblastí, které jsem následně složil v 3Ds Maxu. Až v této chvíli jsem v 3Ds Maxu zkontroloval vedení vozovky terénem a doladil nepřesnosti, kdy terén nechtěně zasahoval do silnice, případně jsem terén snížil v místech mostů. K tomu jsem využil funkce paint deformation.

4.3.2 Materiál terénu

Aplikace World Machine umožňuje mimo jiné také vygenerování textury terénu na základě povrchových dat (například sklonu). Zde však narážíme na problém omezeného rozlišení jedné textury na 513 x 513 pixelů. I kdybychom chtěli například roztříštit terén na dlaždice o délce strany 100 metrů, bude rozlišení textury stále velmi nízké, kompletace takto roztříštěného terénu by bylo zdlouhavé a objem textur příliš velký. Vzhledem k úplné absenci streamování dat scény aplikací VRUT jsem musel přijít se zcela jiným řešením.

Možná řešení

Jednou možností je potáhnout terén jednou opakující se texturou. Rozlišení při pohledu z blízka nebude rušivé, ale jakákoli variace v barvě je tímto vyloučena. Pokud bychom chtěli například oblasti s ornou půdou potáhnout jiným typem textury, museli bychom problém řešit na úrovni geometrie, kdy bychom terén rozřezali na části a těm přiřadili různé materiály. Toto řešení pak ale vypouští plynulé přechody mezi různými typy povrchu. Ty by šly řešit přechodovými zónami pomocí dalšího geometrického dělení. Jak vidíme, geometrické řešení rychle narůstá na komplexitě a jeho manuální provedení v celé rozloze scény se jeví z časového hlediska jako nereálné.

VRUT umožňuje u materiálů specifikovat vlastní fragment a vertex shader. Možnosti shaderů jsou však úzce spjaty s podporou zobrazovacího modulu. Například modul raytracer a rendergl3 vlastní shadery nepodporuje, zde se tedy musíme spokojit s uniformní texturou v celé rozloze terénu. Modul rendergl však jistou podporu vlastních shaderů má. Omezení spočívá v nemožnosti připojení více než jedné textury v celém programu shaderu[2].

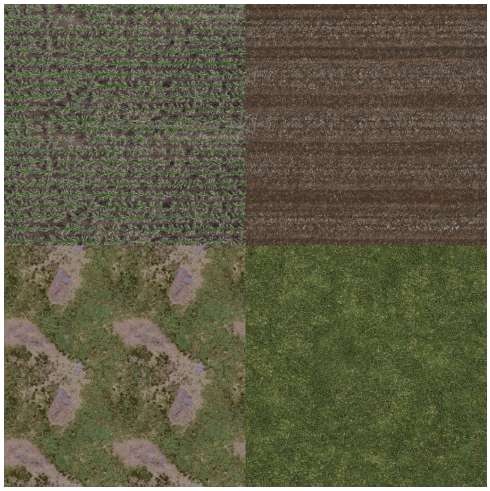
Realizace shaderu

S důrazem na výše uvedená omezení jsem tedy navrhl řešení. K dispozici jsem měl jednu texturu o čtyřech kanálech (3 složky barev a 1 složka průhlednosti). RGB složky ponesou barevnou informaci o povrchu. Složku alfy jsem v našem případě využil pro zakódování typu povrchu. Vzhledem k tomu, že jsem mohl připojit do shaderu pouze jeden soubor textury, seřadil jsem všechny potřebné druhy povrchu do jedné textury tabulkovým způsobem. Systém seřazení ilustruje obrázek 4.3.

Geometrie terénu disponuje mapovacími souřadnicemi UV. Tyto souřadnice mohu pomocí fragment shaderu upravit. V prvé řadě jsem hodnoty vynásobil konstantou 500. Výsledkem této operace bylo zopakování textury v celé ploše terénu na potřebnou velikost. Pomocí vzorce 4.1 pak můžu omezit opakovanou část textury na potřebnou zónu. Parametr a určuje počet druhů povrchů v jednom směru tabulky (v našem případě 2).

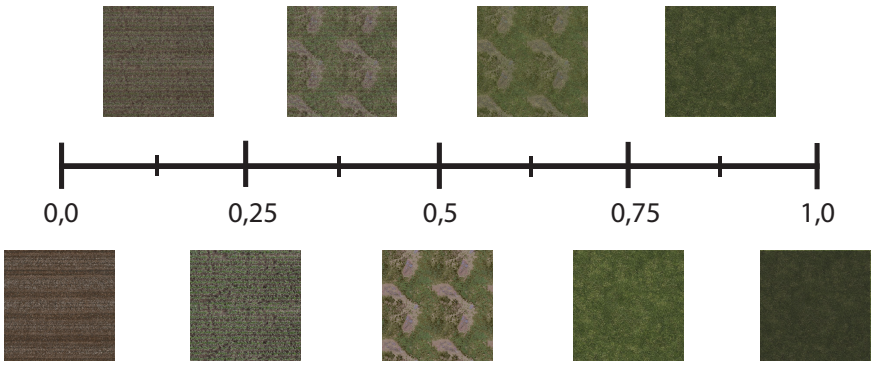
$$\mathbf{uv_{out}} = \frac{(\mathbf{uv_{in}} * scale) \bmod 1}{a} + \mathbf{offset} \tag{4.1}$$

[2]Shader program je v našem případě celá dvojice vertex shader a fragment shader.



Obrázek 4.3: Seřazení typů terénu do mřížky

Normalizovanou souřadnici alfa kanálu jsem logicky rozdělil na 4 intervaly s krajními hodnotami v bodech 0.0, 0.25, 0.5, 0.75 a 1.0. Každému z těchto bodů s výjimkou 1.0 pak náleží určitý typ povrchu. Z pohledu fragment shaderu jsem na základě hodnoty určil potřebný offset při výpočtu mapovacích souřadnic, jak ilustruje rovnice 4.1. Toto řešení nám umožňuje řídit i plynulé přechody mezi typy povrchů. Je-li například alfa složka pixelu textury rovna hodnotě 0.2, bude výsledkem mísení dvou typů povrchů z bodů 0.0 a 0.25. Mísení jsem realizoval pomocí lineární interpolace. Interval 0.75 až 1.0 jsem využil pro řízení světlosti povrchu. Krajnímu bodu 0.75 jsem z tohoto důvodu přiřadil travnatý typ povrchu. Díky tomu jsem mohl použít tmavší odstíny zelené v oblasti lesů (viz obrázek 4.4).

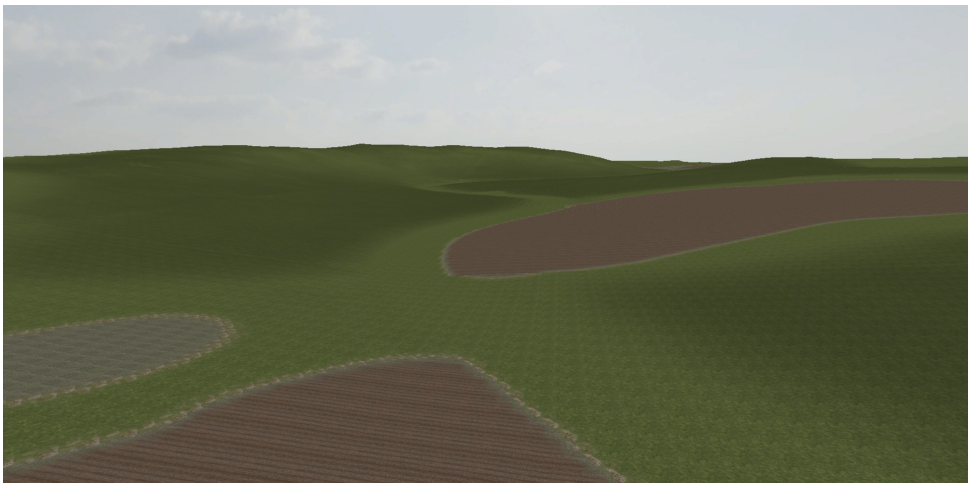


Obrázek 4.4: Přiřazení intervalu alfa kanálu k materiálům

Vytvoření textur

Posledním krokem bylo vytvořit potřebnou podobu alfa kanálu. Oblasti polí a lesů jsem nakreslil manuálně pomocí grafického editoru. Pro drobné nuance

zelené jsem použil vyexportovaná data o erozních pochodech a sklonu terénu. Hotový terén ilustruje obrázek 4.5.



Obrázek 4.5: Výsledek terénu v aplikaci VRUT (modul renderGl)

4.4 Modely scény

Tvorba součástí pro scénu kopírovala postup, který se skládal z kroků:

- vytvoření detailní geometrie
- vytvoření mapování pro textury modelu
- tvorby textur
- tvorby LOD úrovní
- umístění do scény.

4.4.1 Vytvoření geometrie

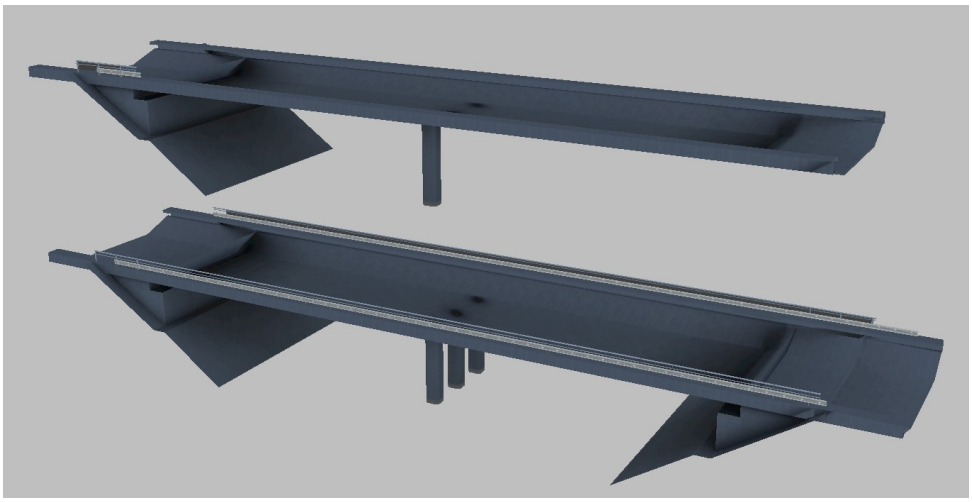
Všechny modely scény byly vytvořeny pomocí polygonální povrchové prezentace. Jako základní strategii tvorby geometrické sítě jsem zvolil volnou editaci vrcholů, ploch a hran. Pro modelování jsem využil 3D studio Max.

Již během modelování jsem pracoval s důrazem na následné použití pro zobrazování v reálném čase. Tím jsem časově odlehčil fázi zjednodušování geometrie a tvorby LOD úrovní.

4.4.2 Mapování geometrie

Vytvoření uv mapovacích souřadnic pro aplikování textur je důležitým krokem ve stavbě modelu. Častou praxí je vytvoření hned dvou setů uv souřadnic, kdy je jedna použita pro aplikaci povrchové textury a druhá určuje mapování lightmap [AMHH08]. Aplikace VRUT však v současnosti dvojí mapování

nepodporuje. Pro navýšení kvality jsem se rozhodl u části objektů obětovat objem textur na úkor rozloženějšího mapování. Použité mapování se blíží podobě světelných map, ale v našem případě bylo možné použít stejné zóny textury pro opakující se geometrii, kde by změny v nasvícení byly zanedbatelné. Jako příklad uvedu 2 koncovky mostu na obrázku 4.6, které sdílí jednu oblast textury. Pro nejrychlejší rozložení uv map jsem použil integrovanou funkci 3D studia, která automaticky generuje mapování podobné světelným mapám. Výsledek jsem pak manuálně doladil podle potřeb modelu.



Obrázek 4.6: Nahoře geometrie s vlastním místem na textuře, dole kompletní geometrie s duplikovanými částmi, které byly v modelu znovu použity

4.4.3 Tvorba textur

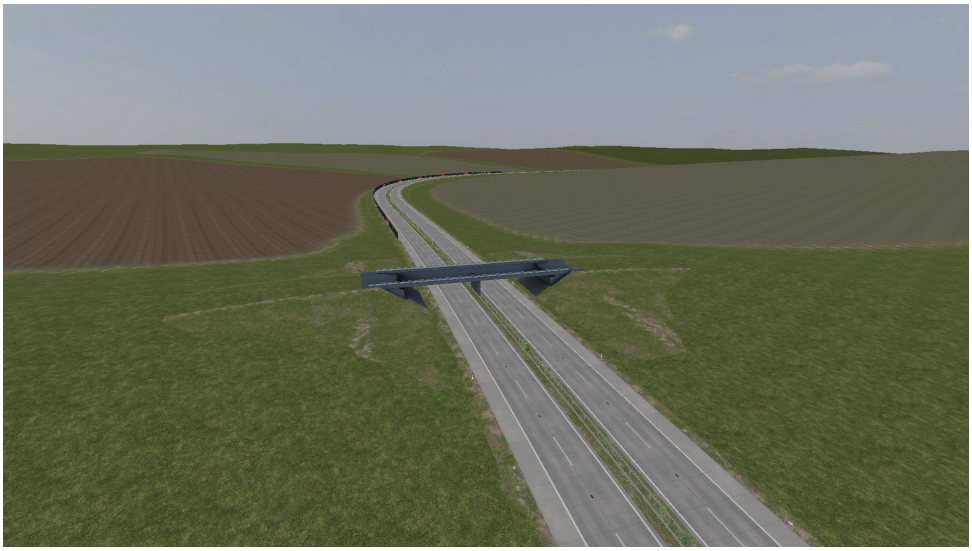
Při tvorbě textur jsem plně využíval potenciálu procedurálních materiálů. Geometrii modelu jsem importoval do Substance Designeru, kde jsem zapekl hodnoty ambientního zastínění, hodnotu normál, pozici bodů a dalších map, které jsem použil jako vstup do generátorů prachu a špíny. Pomocí grafu jsem pak dosáhl potřebné vizuální podoby.

4.4.4 LOD úrovně

Pro generování nižších stupňů LOD nebylo využito optimalizačních metod 3Ds Maxu, protože ty často degenerovaly síť mapovacích uv souřadnic. Z tohoto důvodu by pak nebylo možné použít stejné textury. Proto jsem LOD úrovně vytvářel ručně odmazáváním drobných detailů na povrchu objektů, případně vhodným vymodelováním geometrie o nižší složitosti, na které bylo stále možno namapovat stejné textury.

4.4.5 Umístění objektů do scény

Hotové objekty jsem následně manuálně umisťoval do scény. Výjimkou jsou objekty křivkového charakteru (například svodidla, mosty, protihlukové zdi a



Obrázek 4.7: Usazení statických a křivkových objektů do scény

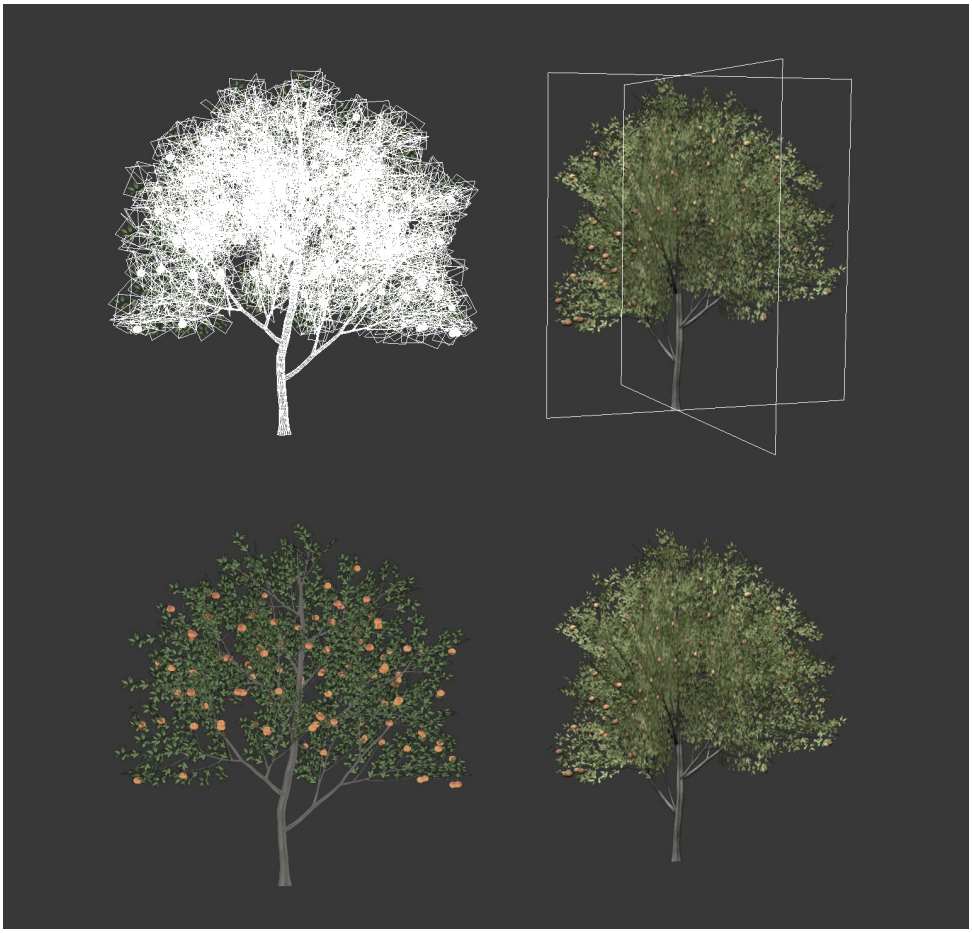
podobně). Křivkové modely jsem pokládal podél definovaných křivek uvnitř 3Ds Maxu. K tomu jsem využil funkce scénového modifikátoru pro deformaci geometrie podél křivek. Pro urychlení procesu jsem často vytvořil komplexnější bloky geometrie a až následně jsem ji ohýbal podél křivek. Tento postup jsem uplatnil třeba u náspu, jehož součástí byla například i svodidla středního pásu, vegetace středního pásu nebo reflexní sloupky.

4.5 Vegetace

Vegetace je jedna z nejdůležitějších částí venkovních scén a výrazně formuje jejich podobu. Zeleň je ovšem také poměrně náročnou disciplínou z hlediska vykreslování v reálném čase. Moderní enginy mnohdy používají sofistikované postupy při jejím vykreslování. Aplikace VRUT má však z tohoto pohledu velmi omezené možnosti, proto byl přístup k provedení vegetace značně konzervativní.

4.5.1 Modely stromů a travin

Protože modul renderGl v současnosti nepodporuje LOD, nemůžeme si dovolit používat ve scéně plnohodnotné modely stromů a keřů, u kterých není výjimkou, že se model skládá i z desetitisíců polygonů. Jako vhodná alternativa se pro nás jeví využití přístupu, kdy je strom složen ze dvou křížících se polygonů. Na polygony je následně aplikována textura s alfa kanálem. Příklad takového modelu prezentuje obrázek 4.8. Stromy modelované pomocí dvou ploch jsou sice nenáročné na vykreslování, chovají se však nekorektně vzhledem ke světlům a stínům. Z blízké vzdálenosti pak může být použití takových modelů rušivé, proto je potřeba vyvážit tuto nevýhodu kvalitními texturami.

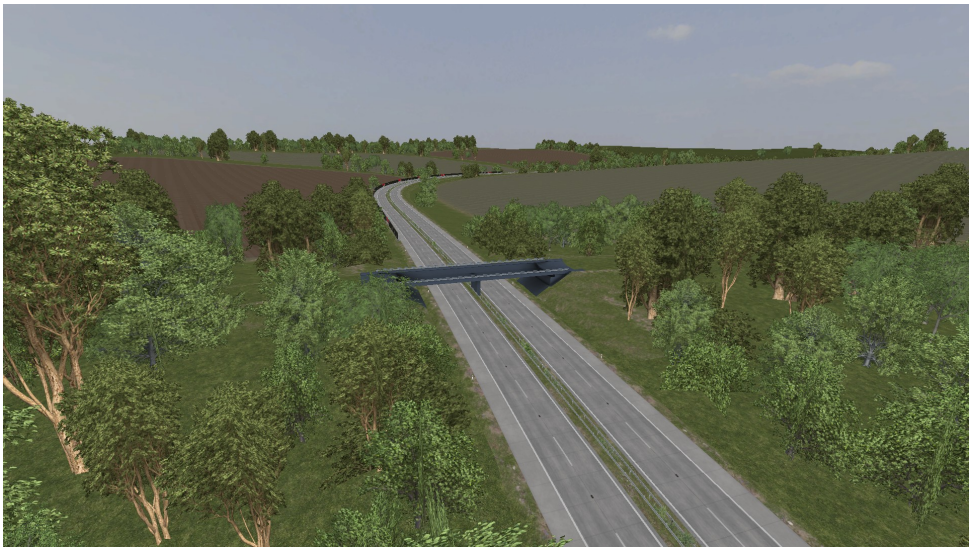


Obrázek 4.8: Příklad modelu stromu Xfrog, vlevo pomocí husté polygonální sítě (117675 polygonů), vpravo pomocí bilboardového kříže (4 polygony)

Vytvořit textury lze pro plochou vegetaci dvěma základními způsoby: pomocí fotografií, nebo rederingem počítačově vytvořených modelů. Focení vegetace není triviální z hlediska vhodného nasvícení nebo barevné kalibrace mezi jednotlivými stromy. Proto jsem zvolil druhou možnost.

Modelovat vegetaci manuálně je obtížné a časově náročné. Ruční modelování také nemusí bez větších zkušeností přinést věrohodný výsledek. Existují nástroje pro procedurální generování vegetace jako například speedTree [IDV17] nebo Xfrog [Inc17]. Jedná se však o komerční software. Xfrog však na svých webových stránkách nabízí několik kusů vygenerované vegetace, které je možné bezplatně stáhnout. Součástí balíku byly již zpracované ploché snímky včetně alfa kanálu, což mi velmi ulehčilo práci. Vytvořit potřebný kříž o dvou polygonech v 3D studiu už bylo triviální.

Pro modely travin a nižších stromů, které jsem využil zejména v oblasti svodidel v tělese vozovky, jsem vytvořil z vyfotografovaných textur ze serveru CGtextures [Tex17]. Odstíny jednotlivých textur jsem pak sladil pomocí grafického editoru.



Obrázek 4.9: Usazení modelů vegetace do scény

4.5.2 Umístění zeleně do scény

Umisťování vegetace po kusech vzhledem k rozloze scény nepřipadá v úvahu. Aplikace VRUT také nedisponuje žádným systémem pro procedurální rozmisťování zeleně jako například Unreal Engine 4 [Gam17c].

Jako alternativní způsob se nabízí funkce object painting v 3D studiu. Ta umožňuje umístit pomocí tažením štětce určitou množinu objektů, v našem případě seznam stromů, na zvolený povrch (terén). Funkci lze pomocí parametrů detailně nastavit a ovlivnit tak výsledek. Stěžejní bylo například nastavení náhodného výběru stromů z daného seznamu, náhodná velikost ve zvoleném intervalu, náhodné natočení a podobně.

Vegetaci jsem umisťoval s důrazem na co největší snímkovací frekvenci, proto se objevuje ve scéně pouze v bezprostřední blízkosti silnice. Geometrie s alfovanými materiály jsou také dražší na vykreslení a velkou roli hraje také časté překreslování fragmentů (pixel overdaw), protože stromy často tvoří polygonální vrstvy z horizontálního pohledu. Více o pixel overdraw se lze dočíst v [AMHH08]. Abych zakryl fakt, že je vegetace v oblastech lesů položená pouze do několika metrů od vozovky, umístil jsem na přelomu silnice a kraje lesa nižší a hustší stromy. V místech středního pruhu svodidel je pak jediný prostor, na kterém najdeme modely trávy.

4.6 Korektnost dopravní stavby

Abych zajistil věrohodnost silničního tělesa dálnice, snažil jsem se při přestavbě respektovat řadu norem. Zde je však třeba upozornit na skutečnost, že původní scéna z mnoha úhlů nerespektuje normy pro budování silničních komunikací. Náplní mého studia není projektování dopravních staveb, proto je využití norem pouze snahou o přiblížení se realitě, nikoliv snahou o vytvoření

modelu absolutně odpovídající platným zákonům. Vybereme-li ty nejdůležitější dokumenty, využil jsem:

- normu zásady pro dopravní značení na pozemních komunikacích [dopk13] - upotřebeno pro svislé značení dálničních sjezdů
- vzorový list staveb pozemních komunikací o silničních tělesech [dopk95]. - upotřebeno pro korektní zpracování náspu dálničního tělesa
- vzorový list o dopravním značení [dopk17] - upotřebeno pro zpracování reflexních prvků
- vzorový list staveb pozemních komunikací o mostech [dopk15b] - upotřebeno pro zpracování tělesa a koncovek mostů
- dokument zásady pro vodorovné dopravní značení na pozemních komunikacích [dopk12] - upotřebeno pro korektní podobu vodorovného značení
- dokument o technických podmínkách svodidel na pozemních komunikacích [dopk15a] - upotřebeno pro zpracování betonových a ocelových svodidel.

Při stavbě dálničních sjezdů jsem mnohdy využíval i satelitních snímků, abych udržel uvěřitelný charakter oblouků vozovky.

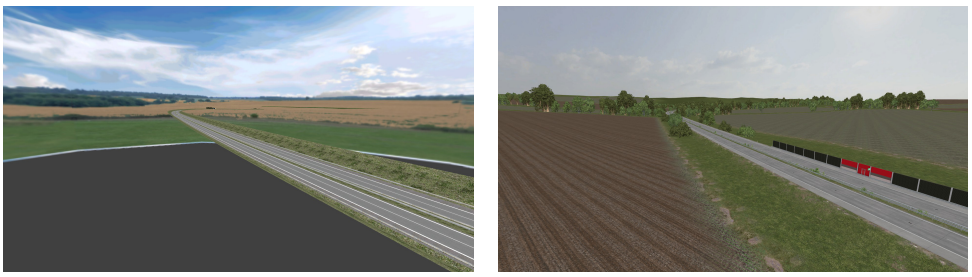
4.7 Optimalizace scény

Pro optimalizaci scény jsem využil modulu optimize. Postupným voláním funkcí: unifyMaterials, dereferenceGeos, iqMerge, mergePrimitives a minimizeACMR bylo dosaženo výrazného zlepšení ve snímkové frekvenci (v řádu až 100 FPS).

Největší vliv mělo použití optimalizační funkce iqMerge, která spojila geometrii všech objektů, které sdílely stejný materiál. Díky tomu se snížil počet vykreslovacích volání, kterých je nyní méně, ale každý z nich vykresluje větší počet primitiv.

Kapitola 5
Výsledky

Předmětem testování bude porovnání původní a přepracované scény. Bude proveden test výkonnosti a prezentovány budou také vizuální výsledky přestavby.



Obrázek 5.1: Porovnání původní a upravené scény

5.1 Technika sběru dat

Výkonnost scény bude měřena v modulu renderGl. Modul má mimo jiné implementovaný čítač měření snímkové frekvence. Loguje také řadu dalších dat, například počet vykreslovacích volání nebo čas strávený vykreslováním průhledné geometrie, neprůhledné geometrie, případně čas strávený ořezáváním zadních stran polygonů. Modul rendergl3 má také implementovaný profilovací nástroj, množina sbíraných dat je však omezenější než u modulu renderGl.

Abych logováním dat ovlivnil výkon zobrazovacího modulu co nejméně, určil jsem frekvenci 250 milisekund, kterou se budou logovat aktuální data. Do modulu renderGl jsem následně naprogramoval zápis logovaných dat do souboru. Vzhledem k charakteru scény (obsahuje velké množství vegetace) budu kromě snímkové frekvence zachytávat také data o času potřebném k vykreslení průhledné a neprůhledné geometrie.

U modulu rendergl3 jsem také implementoval zápis logovaných dat do souboru. Tento modul ale v současné době neeviduje čas strávený vykreslováním průhledné a neprůhledné geometrie, proto budeme evidovat pouze snímkovou frekvenci a čas, který modul rendergl3 strávil generováním jednoho snímku.

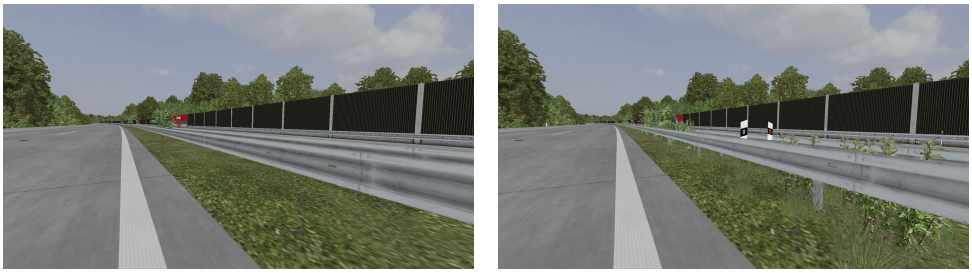
Nutno podotknout, že modul rendergl3 je stále ve vývoji, proto mají výsledky na tomto modulu pouze experimentální charakter.

5.2 Vytvoření testovacího skriptu

Modul vehicleSimulator nabízí možnost nahrát jízdu a následně záznam znova pustit. Této funkce jsem využil a připravil záznam průjezdu scénou v jednom směru v délce přibližně 22,5 minuty. Díky záznamu jízdy byl průjezd ve všech testech shodný. Pro snazší provedení testu jsem vytvořil javascript, který mi umožnil spustit jeden ze zvolených testů (viz kapitola 5.3) a následně automaticky nastavil potřebné parametry. Abych zefektivnil spouštění testů, rozdělil jsem je do skupin. Pomocí javascriptu jsem tyto testy následně volal sekvenčně, takže dokázaly běžet bez dozoru.

5.3 Přehled testů

Absence implementace LOD v modulu renderGl mi znemožnila měřit dopad této techniky na výkonnost, proto jsem LOD úrovně vypínal a zapínal uniformně v celé scéně. Výsledek přímo neodpovídá dynamickému řízení složitosti geometrie, nabízí nám však alespoň přibližnou představu o možném výkonnostním benefitu.



Obrázek 5.2: Příklad rozdílné úrovně detailu u středního pásu dálnice

V našem případě se hlavním parametrem, který ovlivňuje výkonnost, stala hodnota vzdálené roviny ořezového hranolu (vybrány byly vzdálenosti 30 km, u které je viditelná celá scéna, 3 km, která je ve většině případů nerozeznatelná od dohledu 30 km, a 0,5 km). Díky jejímu posunu můžeme řídit dohledovou vzdálenost a množství vykreslené geometrie. Hodnota blízké roviny byla nastavena na hodnotu 40 pro všechny testy.

Pro srovnání byla provedena i trojice testů s původní scénou. Všechny testy byly měřeny v režimu celé obrazovky v rozlišení 1920x1080 bodů.

Bylo sestaveno celkem 11 testů, které dokumentuje tabulka 5.1.

Nastavení modulu rendergl dokumentuje tabulka 5.2 (neuvedené parametry byly deaktivovány).

Nastavení modulu rendergl3 dokumentuje tabulka 5.3 (neuvedené parametry byly deaktivovány).

test	modul	dohled [km]	scéna	LOD
1	rendergl	30	nová	0
2	rendergl	3	nová	0
3	rendergl	0,5	nová	0
4	rendergl	3	nová	1
5	rendergl	3	nová	2
6	rendergl3	3	nová	0
7	rendergl3	3	nová	1
8	rendergl3	3	nová	2
9	rendergl	30	původní	-
10	rendergl	3	původní	-
11	rendergl	0,5	původní	-

Tabulka 5.1: Charakteristika jednotlivých testů (LOD0 označuje úroveň detailu geometrie v nejvyšším detailu, LOD2 pak označuje geometrii nejmenší složitosti)

parametr	nastavení
drawUsingFBO	zapnuto
showStats	zapnuto
useVBO	zapnuto
texturing	zapnuto
runRenderLoop	zapnuto
clearColorBuffer	zapnuto
antiAliasingMethod	None
sortGeometries	0-None
lighting	vypnuto

Tabulka 5.2: Nastavení modulu rendergl

Pro testování byly použity 2 sestavy 5.4, na kterých bylo provedeno všech 11 testů.

5.4 Výsledky

Testy ze sekce 5.3 byly rozděleny do scénářů po trojicích:

1. nová scéna na modulu renderGl s proměnnou hodnotou ořezové roviny (LOD0) - testy 1, 2 a 3
2. nová scéna na modulu renderGl s proměnnou úrovní složitosti geometrie (dohled 3 km) - testy 3, 4 a 5
3. nová scéna na modulu rendergl3 s proměnnou úrovní složitosti geometrie (dohled 3 km) - testy 6, 7 a 8
4. původní scéna na modulu rendergl s proměnnou hodnotou ořezové roviny - testy 9, 10 a 11.

parametr	nastavení
Enable lighting	vypnuto
Render with VBO	zapnuto
Back-face culling	vypnuto
Looping rendering	zapnuto
Render with vertex arrays	zapnuto
Show profiling	zapnuto
FBO Multisampling Method	None
Sorting	No Sorting

Tabulka 5.3: Nastavení modulu rendergl3

	sestava A	sestava B
procesor	i5-3210M (2,5 GHz)	i7-4770S (3,1 GHz)
grafická karta	Nvidia 650M (2 GB)	Nvidia 970 (4 GB)
množství RAM	8 GB	16 GB
operační systém	Windows 8.1 64bit	Windows 7 64bit

Tabulka 5.4: Parametry testovacích sestav

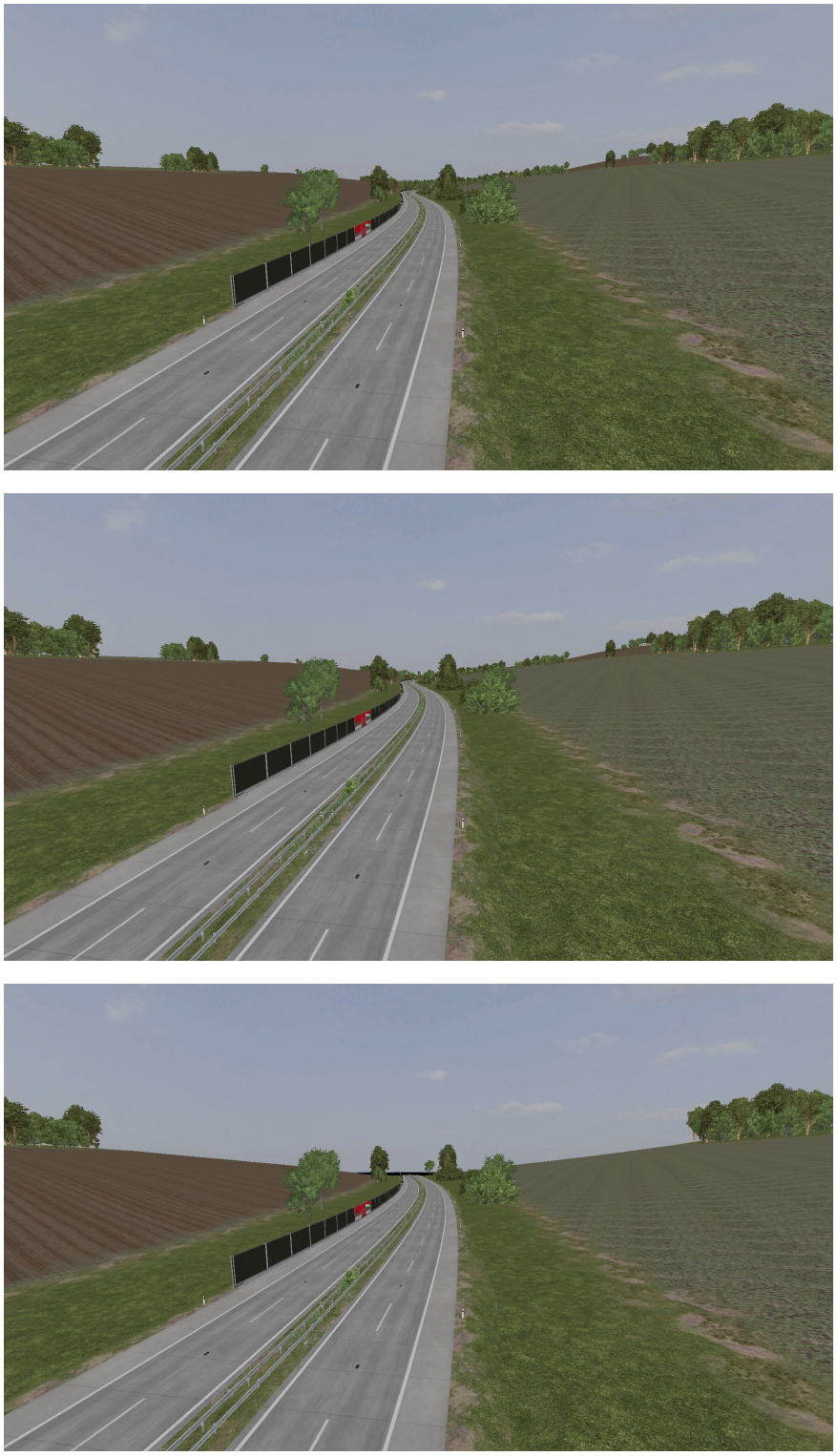
5.4.1 Výsledky scénáře 1

Potřebný čas k vykreslení snímku na grafu 5.4 naznačuje to, že vliv dohledové vzdálenosti je pro zkoumané testy poměrně malý. Znatelně se odlišuje pouze linka s dohledem 0,5 km. Výkon dohledových vzdáleností 30 km a 3 km se v grafu prakticky slučuje.

Výkyvy v čase na snímek jsou prakticky shodné mezi oběma sestavami. V časech přibližně 400 a 900 sekund si můžeme všimnout, že se k sobě všechny linky grafu přibližují. Tento fakt je způsoben vedením vozovky, kdy se auto nachází v zatáčce a velká část scény není v zorném poli. V těchto místech se vytrácí benefit nižší dohledové vzdálenosti.

První tři sloupce grafu 5.9 dokumentují složení průměrných časů na snímek. Barevně jsou vyznačeny časy fází, kdy se vykreslovala průhledná, respektive neprůhledná geometrie. Jako zajímavá zkutečnost se jeví, že u sestavy B zabralo vykreslování větší část z celkového času. Tento jev je pravděpodobně způsoben mnohem výkonnějším procesorem sestavy B. Z toho vyplývá, že aplikační část zobrazovacího řetězce (viz kapitola 2.1) byla provedena mnohem rychleji než v případě sestavy A, kde aplikační část zabrala mnohem více času[1]. Tento fakt potvrzuje procesor sestavy B, který pracuje s vyšším taktem než u sestavy A.

[1]Součástí sloupce *ostatní* grafu 5.9 jsou i výpočty, které nepatří do aplikační části zobrazovacího řetězce jako například ořezávání zadních stran polygonů. Tyto části však zabírají pouze malé procento času, a proto si je dovoluji zanedbat.



Obrázek 5.3: Dohledová vzdálenost (odshora: 30 km, 3 km, 0,5 km)

5.4.2 Výsledky scénáře 2

Graf 5.5 dokumentuje dopad na výkonnost z pohledu složitosti použité geometrie. Z grafů je patrné, že snímkové časy se škálují dle předpokladu, že čím jednodušší geometrie je, tím rychleji se ji podaří vykreslit. Díky použití dynamického řízení LOD bychom pak mohli dosáhnout zvýšení výkonu až o 30% u sestavy A, respektive 12% u sestavy B.

Zajímavým jevem je složení snímkových časů, které dokumentuje graf 5.8. Z grafu sestavy A je patrné, že snížení času snímků nepřišlo ve fázi vykreslování průhledné ani neprůhledné geometrie. Opakování testu však nepřineslo odlišné výsledky. Vzhledem k omezení profilovacích nástrojů modulu renderGl se mi nepodařilo odhalit, jaké akce modulu stojí za zrychlením snímkového času. Naopak graf sestavy B dokumentuje mnohem očekávanější výsledky, kdy se složky snímkového času škálují podle složitosti geometrie.

5.4.3 Výsledky scénáře 3

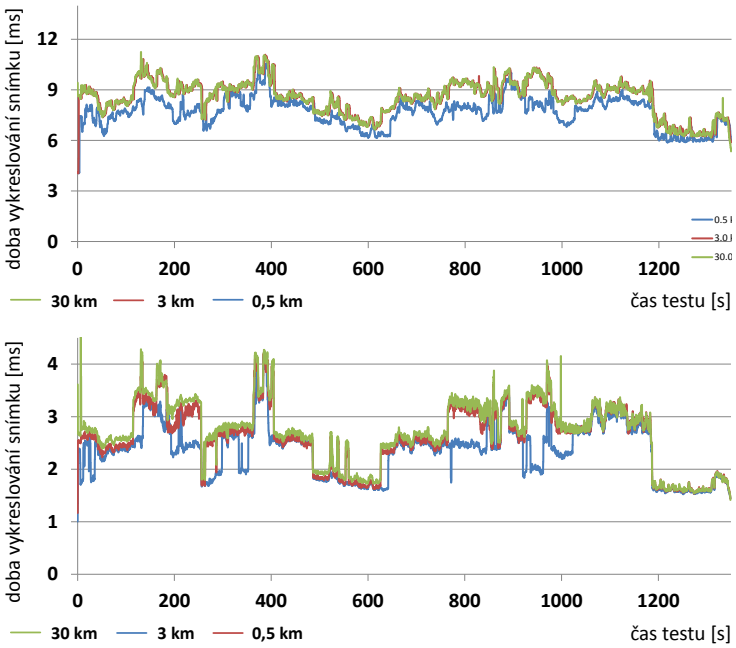
Graf 5.6 dokumentuje vývoj snímkové frekvence na modulu rendergl3. Když porovnáme grafy 5.8, zjistíme, že modul rendergl3 dosahuje přibližně o 21%, respektive 52% vyšší průměrné snímkové frekvence. Tento výsledek je však zatížen systematickou chybou. V sekci 4.3.2 je popsána výroba shaderu pro terén. Modul rendergl3 však v této chvíli nemá implementaci pro vlastní shadery, proto byl použit terén s uniformní texturou. Použití shaderu zpomalilo snímkové frekvence u modulu rendergl přibližně o 15% a tím zvýhodnilo výkon modulu rendergl3.

Graf 5.8 dokazuje, že modul rendergl3 těží více z výkonnější sestavy než modul renderGl.

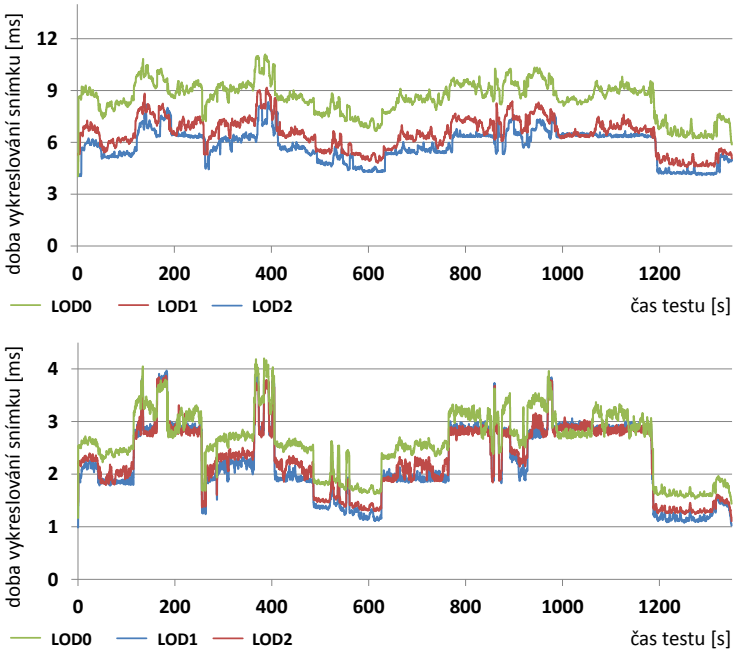
5.4.4 Výsledky scénáře 4

Pro porovnání byla provedena trojice testů s původní scénou. Z tohoto testu je zajímavé srovnání složení jednotlivých snímkových časů, které dokumentuje graf 5.9. Vykreslení původní scény bylo podle očekávání mnohem rychlejší. Rozdíl byl znatelnější na výkonnější sestavě B. Původní scéna byla geometricky mnohem chudší. Zajímavým detailem se jeví čas, který byl věnován vykreslování průhledné geometrie. U původní scény je tento díl v grafu téměř neznatelný. Tento fakt je způsoben hlavně hustotou vegetace po přestavbě scény.

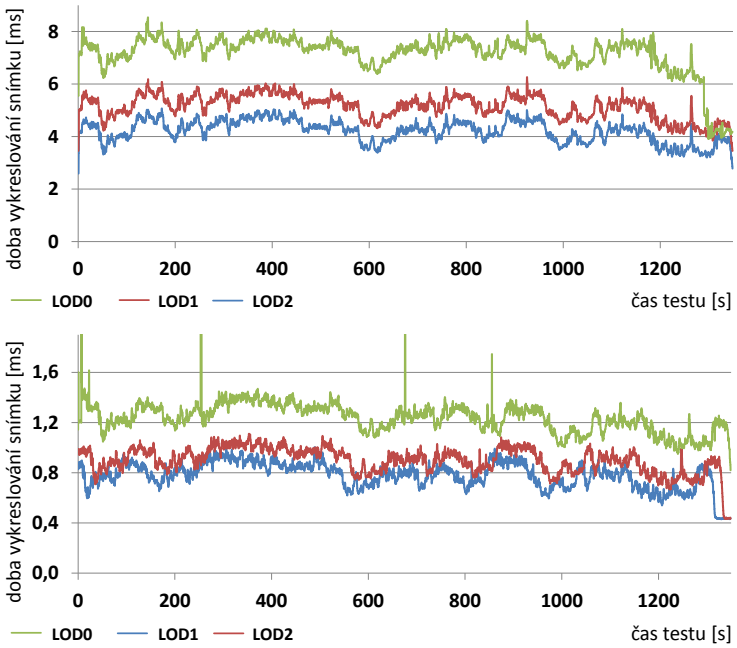
Vykreslovací čas se s postupným průjezdem scény snižuje (viditelné hlavně od 1200. sekundy dále). Tato snižující se tendence je dána částí scény, která je viditelná z pohledu automobilu. Při startu je v zorném poli při dohledu 30 km celá scéna. Postupným průjezdem je stále větší díl za autem a menší díl scény před ním.



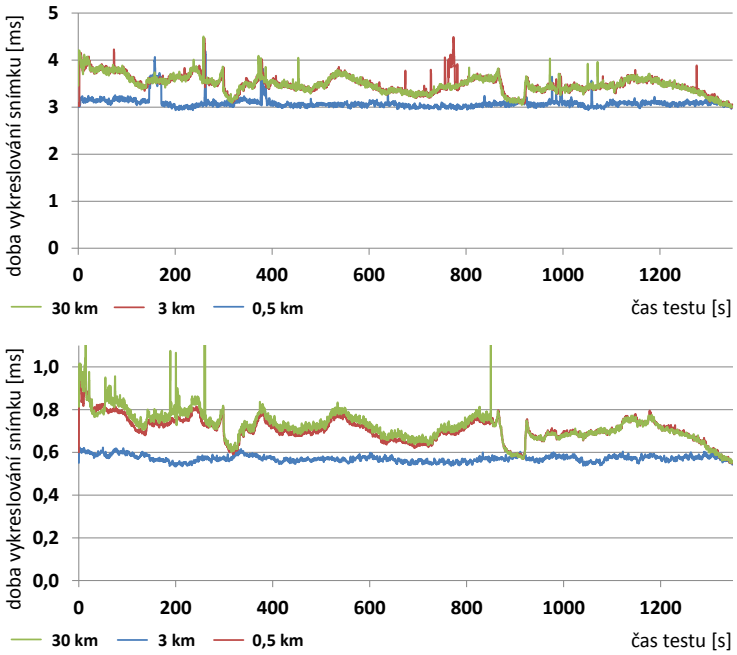
Obrázek 5.4: Vývoj času potřebného k vykreslení jednoho snímku při průjezdu scénou s různou dohledovou vzdáleností pro modul rendergl (v pořadí: sestava A, sestava B)



Obrázek 5.5: Vývoj času potřebného k vykreslení jednoho snímku při průjezdu scénou s různou složitostí geometrie pro modul rendergl (v pořadí: sestava A, sestava B)



Obrázek 5.6: Vývoj času potřebného k vykreslení jednoho snímku při průjezdu scénou s různou složitostí geometrie pro modul rendergl3 (v pořadí: sestava A, sestava B)



Obrázek 5.7: Vývoj času potřebného k vykreslení jednoho snímku při průjezdu původní scénou s různou hodnotou dohledu na modulu renderGl (v pořadí: sestava A, sestava B)

test	modul	dohled [km]	LOD	FPS A	FPS B
1	rendergl	30	0	88,9	190,8
2	rendergl	3	0	90,2	238,8
3	rendergl	0,5	0	94,7	240,6
4	rendergl	3	1	109,1	258,4
5	rendergl	3	2	120,1	247,0
6	rendergl3	3	0	117,1	335,1
7	rendergl3	3	1	159,7	900,2
8	rendergl3	3	2	197,2	996,0
9	rendergl	30	0	222,2	868,0
10	rendergl	3	0	222,7	1063,9
11	rendergl	0,5	0	238,6	1606,8

Tabulka 5.5: Přehled minimálních naměřených hodnot snímkové frekvence napříč testy pro sestavy A a B

	původní scéna	nová scéna
počet objektů	917	140
počet materiálů	936	55
počet polygonů	821 843	2 280 528/1 483 918/1 035 061
objem dat	47,7Mb	486Mb

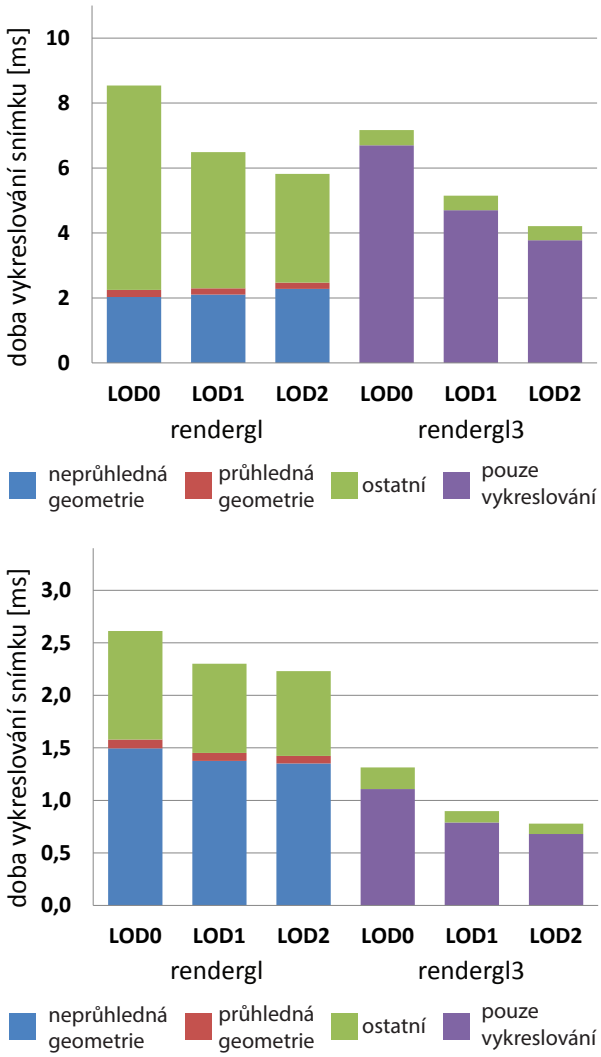
Tabulka 5.6: Statistický přehled dat původní a upravené scény (trojice čísel u polygonální položky značí jednotlivé LOD úrovně od nejsložitější po nejjednodušší)

5.5 Grafický výstup modulu RayTracer

Výstup modulu RayTracer je značně ovlivněn jeho nastavením. Vykreslování bude prováděno s vypnutým reflektorem z pohledu kamery a s aktivovaným vykreslovacím postupem PathTracing.

Z důvodu použití bilboardové vegetace byly deaktivovány světelné zdroje a barva reflektoru byla nastavena na černou. Veškeré osvětlení bylo provedeno pomocí globálního nasvícení pomocí skyboxu. Abych zvedl jas obrazu, nastavil jsem hodnotu keyValue na 440. Přepálení obrazu jsem korigoval nastavením parametru whitePoint na maximum (2000). Nastavení modulu RayTracer prezentuje tabulka 5.7 (neuvedené parametry byly ponechány s výchozí hodnotou).

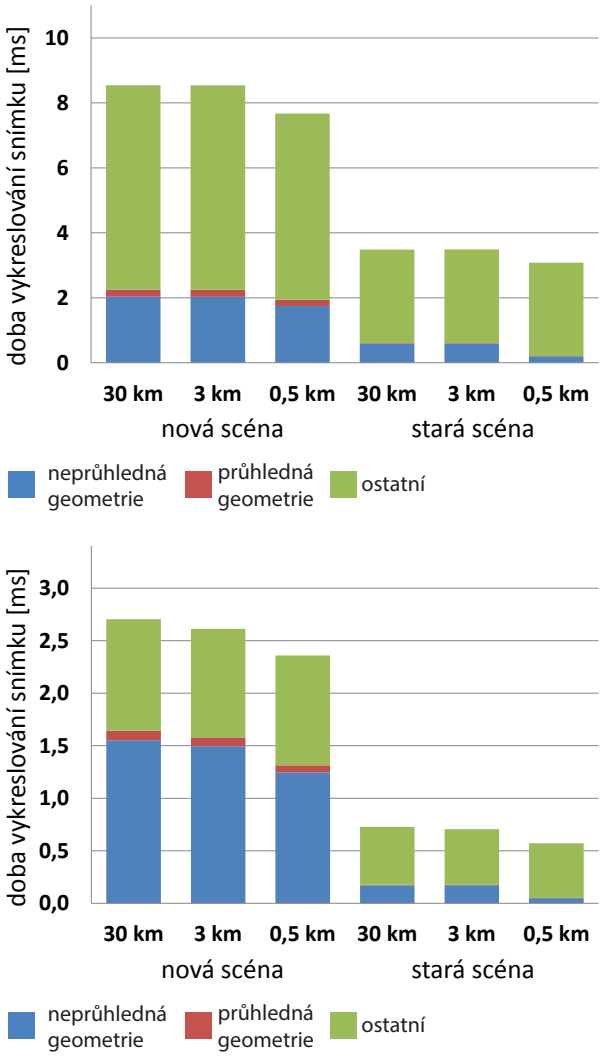
Současné počítače při nastavení takovýchto hodnot zatím nedokáží zobrazovat scénu v reálném čase. S optimálním nastavením však lze přepracovanou scénu pomocí modulu RayTracer zobrazit s poměrně kvalitním grafickým výstupem.



Obrázek 5.8: Porovnání časů vykreslování snímku a jeho složení (v pořadí: sestava A, sestava B)

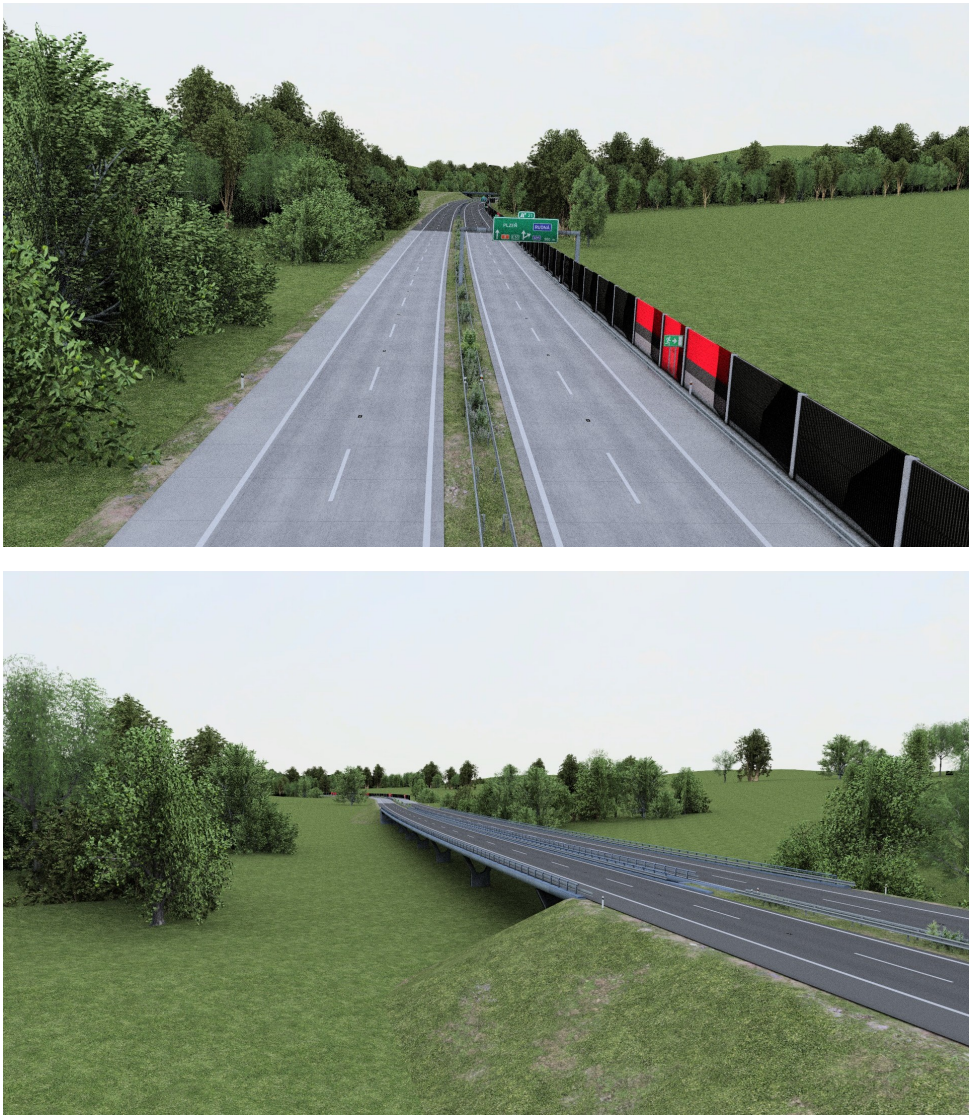
parametr	nastavení
maxLights	0
maxrecursionDepth	20
rayOffset	10
usePathTracing	zapnuto
whitePoint	2000
keyValue	440

Tabulka 5.7: Nastavení modulu rendergl3



Obrázek 5.9: Porovnání časů vykreslování snímku a jeho složení (v pořadí: sestava A, sestava B)

Obrázek 5.10: Výstup modulu RayTracer



Obrázek 5.11: Výstup modulu RayTracer

Kapitola 6

Závěr

V rámci této závěrečné práce byla úspěšně přepracována rozsáhlá scéna dálničního vedení. Výsledného modelu bylo dosaženo za použití sofistikovaných nástrojů pro procedurální generování terénu (World Machine) či textur (Substance Painter 2, Substance Designer). K modelování byl hojně využit program 3Ds Max.

I když byla upravená scéna složena z více než dvojnásobku polygonů u nejdetailnější verze než scéna původní (spolu s více než desetkrát větší paměťovou náročností), dosahovala v aplikaci VRUT více než uspokojivých výsledků. Dle testů bylo dosaženo vysokých snímkových frekvencí, které v řadě sestavených testů překračovala poměrně vysokou hodnotu 100 snímků za sekundu a v žádném z nich na pomalejší sestavě neklesla pod hodnotu 80 snímků za sekundu.

S dalším vývojem modulu rendergl3 může dojít ke zkvalitnění vizuálního výstupu, ale v případě implementace dynamického řízení LOD i výkonnostních zlepšení.

Rovněž bylo dosaženo vysoké vizuální kvality u statických obrázků, které byly vygenerovány pomocí modulu RayTracer. S dalším rozvojem techniky by mohl modul RayTracer v budoucnu běžet v rozumné kvalitě i v reálném čase.

Příloha A

Literatura

[All17] Allegorithmic, *Substance tools*, www.allegorithmic.com, 2017.

[AMHH08] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman, *Real-time rendering*, A K Peters, 2008.

[Aut17] Autodesk, *3ds max*, www.autodesk.com/products/3ds-max/overview, 2017.

[dopk95] Ministerstvo dopravy odbor pozemních komunikací, *Vzorové listy staveb pozemních komunikací, vl 2 - silniční těleso, md Čr č.j. 18864/95-230*, http://www.pjpk.cz/data/USR_001_2_10_VL/VL2_Silnicni_teleso__199505_.pdf, 1995.

[dopk12] _____, *Zásady pro vodorovné dopravní značení na pozemních komunikacích, tp 133, md Čr č.j. 538/2013-120-stsp/1*, http://www.pjpk.cz/viewFile.asp?file=1583, 2012.

[dopk13] _____, *zásady pro dopravní značení na pozemních komunikacích, tp 65, md Čr č.j. 532/2013-120-stsp/1*, http://www.pjpk.cz/viewFile.asp?file=1529, 2013.

[dopk15a] _____, *Svodidla na pozemních komunikacích, tp 114, md Čr č.j. 58/2015-120-tn/2*, http://www.pjpk.cz/viewFile.asp?file=1570, 2015.

[dopk15b] _____, *Vzorové listy staveb pozemních komunikací, vl 4 – mosty, md Čr č.j. 50/2015-120-tn/1*, http://www.pjpk.cz/viewFile.asp?file=1764, 2015.

[dopk17] _____, *Vzorové listy staveb pozemních komunikací, vl 6.3 – dopravní zařízení, md Čr č.j. 44/2017-120-tn/1*, http://www.pjpk.cz/data/USR_001_2_10_VL/VL_6.3_2017.pdf, 2017.

[Fou17] Foundry, *Mari*, www.foundry.com/products/mari, 2017.

[Gam17a] Epic Games, *Epic games texturing guidelines*, docs.unrealengine.com/udk/Three/TexturingGuidelines.html, 2017.

[Gam17b] _____, *Physically based materials*, docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/, 2017.

[Gam17c] _____, *Procedural foliage tool quick start*, docs.unrealengine.com/latest/INT/Engine/OpenWorldTools/ProceduralFoliage/QuickStart/, 2017.

[Ši13] Daniel Šimek, *Rozšiřitelný zobrazovací řetězec založený na odloženém stínování.*, Master's thesis, ČVUT, 2013.

[IDV17] Inc. Interactive Data Visualization, *Speedtree*, www.speedtree.com, 2017.

[Inc17] Xfrog Inc., *Xfrog*, xfrog.com, 2017.

[Kö15] Erland Körner, *Working with physically-based shading: a practical approach*, blogs.unity3d.com/2015/02/18/working-with-physically-based-shading-a-practical-approach/, 2015.

[Kra14] Tomáš Kraus, *Přesný 3d model společných prostor dcgi*, Bachelor's thesis, ČVUT, 2014.

[Kyb08] Václav Kyba, *Modulární 3d prohlížeč*, Master's thesis, ČVUT, 2008.

[Per85] Ken Perlin, *An image synthesizer*, SIGGRAPH Comput. Graph. **19** (1985), no. 3, 287–296.

[Ric14] Christophe Riccio, *How bad are small triangles on gpu and why?*, www.g-truc.net/post-0662.html, 2014.

[RRT95] Diane Rame, Linda Rose, and Lisa Tyerman, *Mtl material format (lightwave, obj)*, http://paulbourke.net/dataformats/mtl/, 1995.

[Sch17] Stephen Schmitt, *World machine*, www.world-machine.com/about.php, 2017.

[sof17] Planetside software, *Terragen*, planetside.co.uk/terragen-showcase/, 2017.

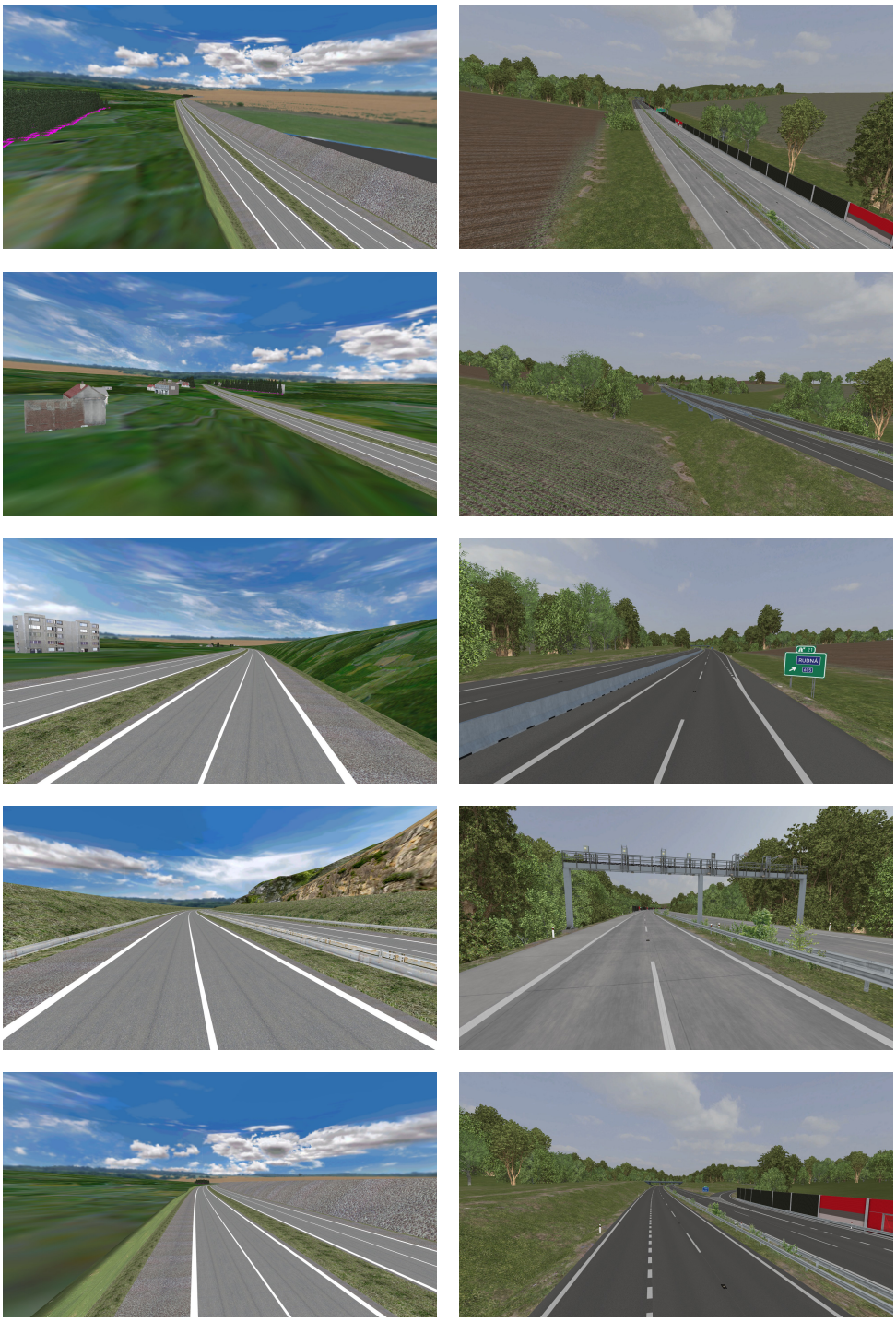
[Tex17] Textures.com, *Textures.com*, www.textures.com/, 2017.

[Wav95] Inc. Wavefront, *Object files (.obj)*, http://paulbourke.net/dataformats/obj/, 1995.

Příloha B

Porovnání původní a přepracované scény

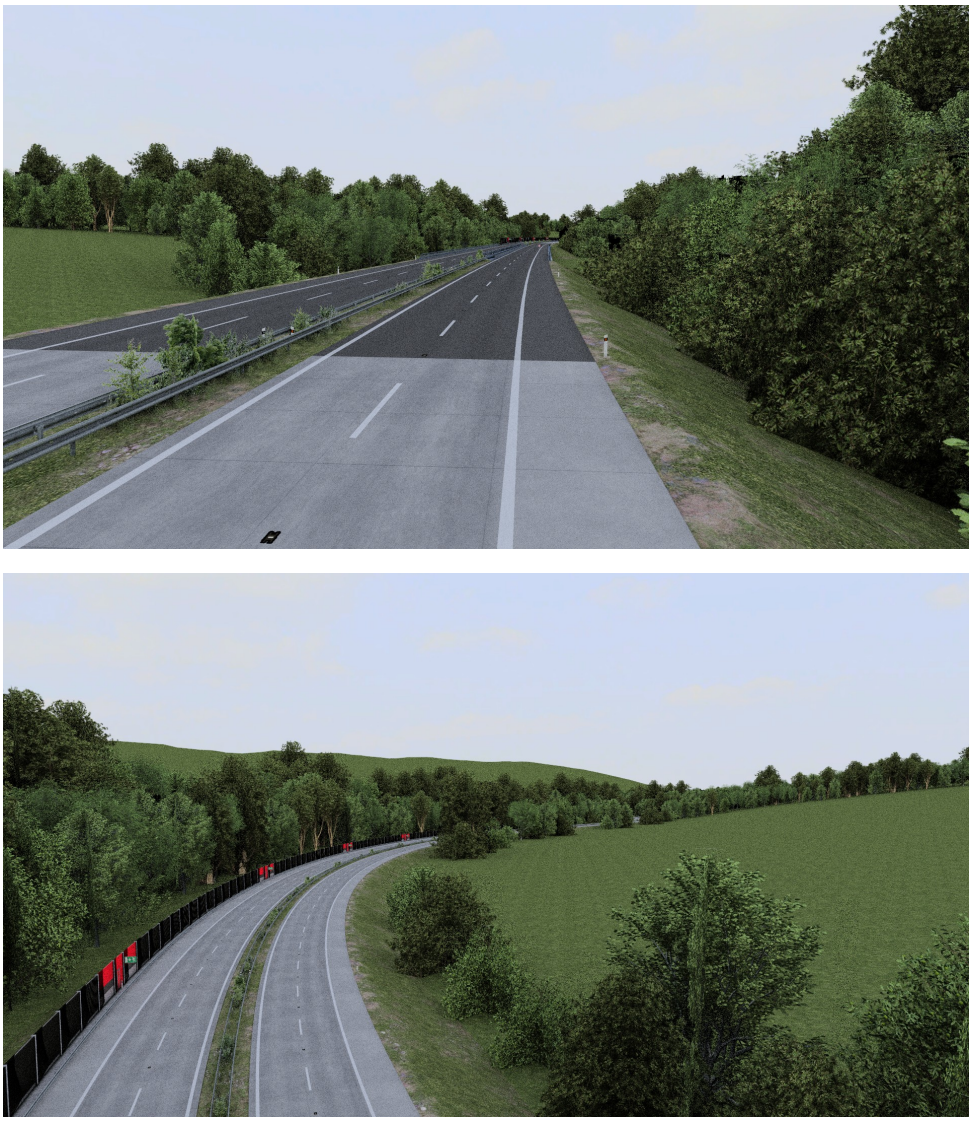
Obrázek B.1: Porovnání původní a přepracované scény



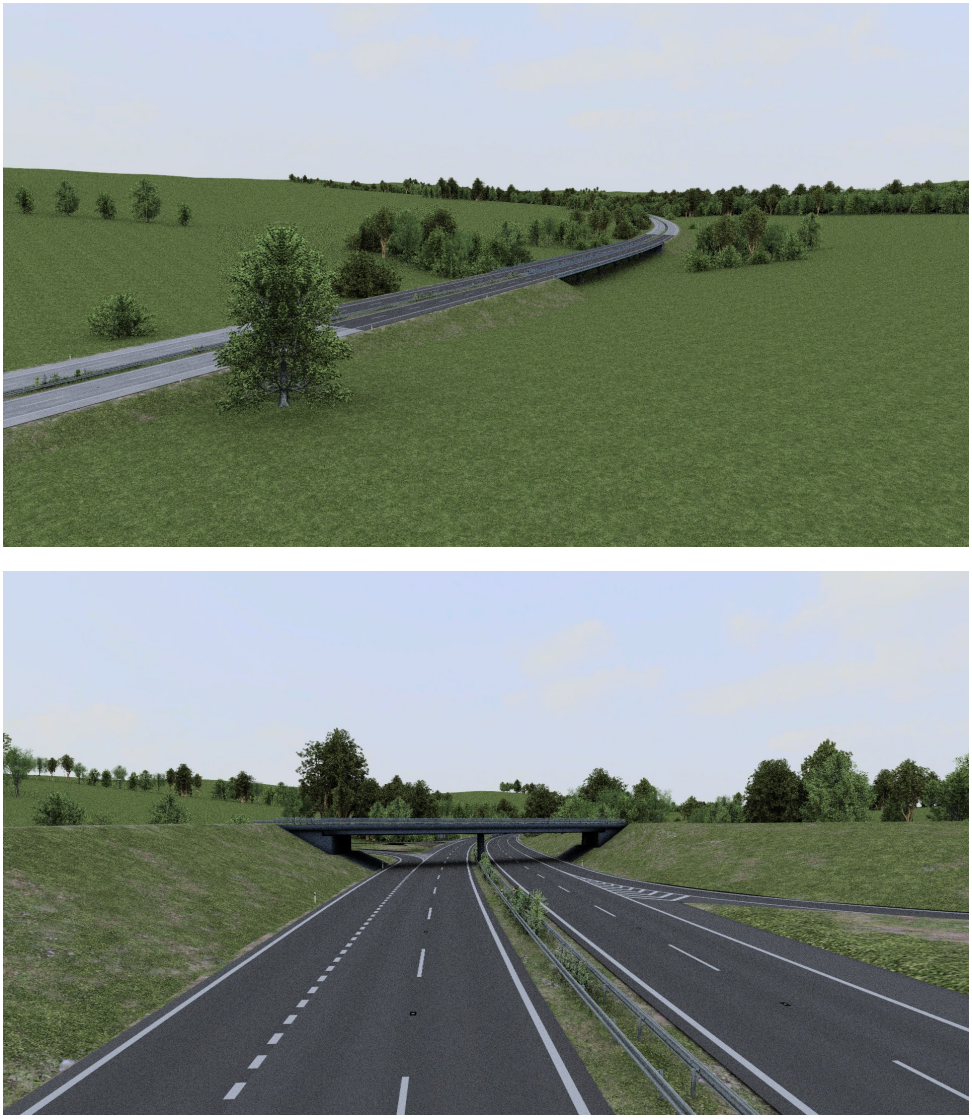
Obrázek B.2: Porovnání původní a přepracované scény

Příloha C

Výstup modulu RayTracer



Obrázek C.1: Výstup modulu RayTracer



Obrázek C.2: Výstup modulu RayTracer

Příloha D

Soubory projektu

- src/ - zdrojové soubory scény a kódových úprav aplikace VRUT
- latex/ - zdrojové soubory textu
- images/ - obrázky práce
- bin/data/Tracks/dalnice2 - soubory modelu scény
- bin/ - spustitelná aplikace VRUT s nastavenými testy
- text.pdf - text bakalářské práce
- readme.txt - popis struktury přílohy

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Petr Brachaczek

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Modely 3D scén pro jízdní simulátor

Pokyny pro vypracování:

Prostudujte existující modely 3D scén používané v jízdním simulátoru v systému Virtual Reality Universal Toolkit (VRUT). Vytvořte nové varianty těchto scén s vyšším množstvím detailů a atraktivnější vizuální podobou. Změny koncipujte tak, aby silniční síť existujících scén zůstala zachována i v nově vytvořených scénách. Zmapujte možnosti stávajících zobrazovacích modulů v systému VRUT (moduly rendergl, rendergl3 a raytracer).
Maximalizujte využití možností těchto modulů s cílem dosažení vyšší vizuální kvality při zachování dostatečné rychlosti zobrazování. Vytvořené scény vyexportujte ve variantách, které v budoucnu umožní efektivní využití nově vytvářených komponent zobrazovacích nástrojů (např. export scény včetně osvětlovacích map s předpočítaným globálním osvětlením).
Zobrazování nově vytvořených scén důkladně otestujte na nejméně dvou hardwarových platformách. Zmapujte závislosti rychlosti zobrazování na nastavení parametrů zobrazovacích modulů.

Seznam odborné literatury:

[1] Václav Kyba. Modulární 3D prohlížeč. Diplomová práce ČVUT FEL, 2008.
[2] Daniel Šimek. Rozšiřitelný zobrazovací řetězec založený na odloženém stínování. Diplomová práce ČVUT FEL, 2013.
[3] Tomas Akenine-Moller, Eric Haines, Naty Hoffman. Real-Time Rendering. A K Peters, 2008.

Vedoucí: doc. Ing. Jiří Bittner, Ph.D.

Platnost zadání: do konce zimního semestru 2018/2019

V Praze dne 4.4.2017