VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZOBRAZENÍ ZÁVODNÍ DRÁHY S ÚROVNÍ DETAILU

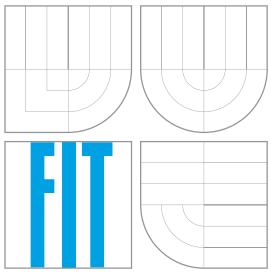
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR MOHELNÍK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZOBRAZENÍ ZÁVODNÍ DRÁHY S ÚROVNÍ DETAILU

LEVEL OF DETAIL FOR RACE TRACK RENDERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR MOHELNÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ POLOK
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce se zabývá efektivním zobrazením závodní dráhy pomocí OpenGL. K reprezentaci terénu se používá výšková mapa. Dráha je zobrazena jako polygonální model zřetězený na křivce s využitím techniky level of detail. Je navržen algoritmus pro použití této techniky v reálném čase i na mobilních zařízeních. Mezi popsanými algoritmy jsou dart throwing, Catmull-Rom spline, kontrakce hran, view frustum culling. Součástí práce je demonstrační aplikace implementující navržený algoritmus.

Abstract

This work describes efficient race track rendering using OpenGL. It uses height map for terrain representation. The race track is rendered as polygonal model instanced on a curve using level of detail. An algorithm is proposed for using level of detail in real time even on mobile devices. Among described algorithms are dart throwing, Catmull-Rom spline, edge collapse simplification, view frustum culling. Part of this work is an application demonstrating the designed algorithm.

Klíčová slova

závodní dráha, výšková mapa, OpenGL, úrověn detailu, LOD

Keywords

race track, heightmap, OpenGL, level of detail, LOD

Citace

Petr Mohelník: Zobrazení závodní dráhy s úrovní detailu, bakalářská práce, Brno, FIT VUT v Brně, 2014

Zobrazení závodní dráhy s úrovní detailu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáše Poloka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Mohelník
19. května 2014

Poděkování

Děkuji svému vedoucímu práce Ing. Lukáši Polokovi za rady při řešení projektu.

Obsah

1 Úvod **2**

2 Reprezentace 3D objektů a terénu **3**
2.1 Konstruktivní geometrie - CSG . . . 3
2.2 Šablonování . . . 3
2.3 Dekompoziční modely . . . 3
2.4 Hraniční modely . . . 4
2.5 3D plochy . . . 4
2.6 Implicitní plochy . . . 4
2.7 Digitální model terénu . . . 5

3 Level of Detail **6**
3.1 Zjednodušující operátory . . . 6
3.2 Chybové metriky . . . 8
3.3 Výběr úrovně detailu . . . 9
3.4 Diskrétní LOD . . . 9
3.5 Spojité LOD . . . 9
3.6 View-Dependent LOD . . . 10
3.7 LOD terénu . . . 10

4 OpenGL **12**

5 Návrh a realizace řešení **14**
5.1 Generování terénu . . . 14
5.2 Definovaní dráhy . . . 15
5.3 Level of Detail . . . 19
5.4 Tvorba nižších úrovní detailu modelu . . . 21
5.5 Osvětlení . . . 23
5.6 Obloha . . . 23
5.7 Model . . . 24

6 Výsledky a měření **25**

7 Závěr **28**

Kapitola 1

Úvod

V počítačové grafice je snaha o co nejrealističtější zobrazení světa a objektů v něm. K tomu ale nemusí být dostatečný výpočetní výkon. Proto se vyvíjejí různé techniky, které se snaží o zobrazování komplexních scén, při využití menšího množství výpočetního výkonu a s minimálním dopadem na výsledný obraz. Existuje množství způsobů, které tento problém řeší. Jednoduchým způsobem může být při zobrazení rozsáhlých scén nahrazení vzdálených horizontů plochou texturou. Složitější techniky mohou být vyčlenění objektů mimo zorné pole kamery, objektů zakrytých jinými objekty nebo částí objektů odvrácených pozorovateli před posláním grafické kartě k zobrazení. Další technikou je level of detail. Může být diskrétní, který rozděluje prostor na regiony a v každém používá jinou předem vytvořenou úroveň detailu. Nebo může být spojitý, který vytváří modely v reálném čase za běhu programu.

Cílem této práce je navrhnutí systému pro efektivní zobrazení závodní dráhy a jeho demonstrace v aplikaci. Závodní dráha se zobrazuje pomocí polygonálních modelů za použití diskrétního level of detail. Pro účely vytvoření dráhy je vygenerován terén. Terén je reprezentován jako výšková mapa. Existuje množství způsobů vytvoření výškové mapy, v této práci je jeden popsán a použit.

Trasa dráhy je určena křivkou vedoucí po terénu. Křivka je rozdělena na segmenty, které jsou zobrazeny pomocí jednoho modelu. Modely na segmentech jsou transformovány tak, aby kopírovaly křivku. Vstupem aplikace je jeden model, ze kterého jsou algoritmicky vytvořeny nižší úrovně detailu tak, aby byla vytvořena návaznost mezi jednotlivými úrovněmi detailu. K implementaci je použit jazyk C++ a programové rozhraní grafického hardwaru OpenGL, které se často používá např. i pro tvorbu počítačových her.

V druhé kapitole práce seznamuje s různými reprezentacemi 3D objektů a terénu. V další kapitole jsou shrnuty různé techniky level of detail. Ve čtvrté kapitole následuje stručný popis rozhraní OpenGL. Pátá kapitola obsahuje návrh výsledné aplikace a popis použitých algoritmů a jejich realizaci. V šesté kapitole je změřena a zhodnocena funkčnost aplikace.

Kapitola 2

Reprezentace 3D objektů a terénu

Kapitola popisuje různé způsoby reprezentace objektů v tří-dimenzionálním prostoru a reprezentaci terénu pomocí výškové mapy.

2.1 Konstruktivní geometrie - CSG

K reprezentaci konstruktivní geometrie se používá strom. Listy jsou primitivní tělesa (kvádr, válec, koule). Uzly stromu obsahují operace (sjednocení, průnik, rozdíl) a transformace, které se nad danými primitivy provádějí. Po provedení každé operace musí proběhnout regenerace stromu. CSG neobsahuje informace o povrchu objektu. K urychlení regenerace a zobrazování se prostor rozděluje oktalovým stromem. Pro účely zobrazení je možné převést CSG model na model polygonální. Zpravidla jsou nachystané polygonální modely používaných primitivních těles a operace, které s nimi umí pracovat. Další metody zobrazení jsou převod na spline hraniční model, ray-casting, ray-tracing. Využití nachází převážně ve strojírenství a architektuře.

2.2 Šablonování

Šablonování definuje povrch tělesa pomocí křivek. Jedna křivka definuje trajektorii, po které se druhá (profilová) křivka nebo plocha pohybuje. Profilová křivka se může v průběhu měnit, to se nazývá potahování. Kromě translačního šablonování existuje také rotační, kde se profilová křivka otáčí. Časté je použití NURBS křivek. Pro zobrazení je nutné převést na polygonální model.

2.3 Dekompoziční modely

Dekompoziční modely popisují objekt diskrétně rozkladem na elementární objemové jednotky (krychle, hranoly). Nejčastěji se používá pravidelná kartézská mřížka. Částice objemu, která představuje hodnotu v této mřížce se nazývá voxel. Je výhodný pokud potřebujeme znát informace o vnitřní struktuře objektu např. k vzorkování objektu nebo vyčíslení obsazeného objemu. Pro převedení na polygonální model se používá algoritmus Marching cubes.

2.4 Hraniční modely

Hraniční modely popisují pouze povrch tělesa. Objekt je definován pomocí vrcholů, hran a stěn.

2.4.1 Polygonální model

Polygonální model (viz obrázek 5.14) popisuje objekt jednoznačně, ale s malou přesností. Má hardwarovou podporu zobrazení a je vhodný pro interaktivní zobrazení. Základním objektem při vytváření modelu je bod v trojrozměrném prostoru, vrchol. Dva spojené vrcholy tvoří hranu. Tři spojené vrcholy třemi hranami tvoří trojúhelník. Z těchto trojúhelníků mohou být vytvářeny složitější polygony. Nejčastěji používané polygony jsou trojúhelníky a čtverce. Trojúhelníky vždy tvoří jednu rovinu, proto je jednoduché zjistit jejich povrchovou normálu. Povrchová normála je trojrozměrný vektor kolmý k povrchu trojúhelníku používaný například k zobrazení osvětlení. Skupina vzájemně propojených polygonů společnými body tvoří síť. Síť může být vytvořená manuálně nebo pomocí nějakého z široké nabídky nástrojů. Pro použití modelu v animacích, hrách aj. je vhodné texturování a přidání kostry pro animování. Texturování umožňuje určit barvu a případně další optické vlastnosti v určitém bodě povrchu modelu. Algoritmus k nanášení textur se označuje pixel shader a je realizován grafickou kartou. Textury mohou být rastrové, které jsou uloženy v předem připraveném rastrovém obrázku nebo procedurální, které jsou vyjádřeny matematickou funkcí. Pro zobrazení na počítači se využívá OpenGL a Direct3D. Hlavní výhodou zobrazení modelů pomocí polygonů je rychlost. Největší nevýhodou je neschopnost přesně zobrazit zakřivené povrchy, k jejich aproximaci je použito velké množství polygonů.

Základní datová struktura pro reprezentaci polygonálního modelu je okřídlená hrana. Obsahuje tři provázané lineární seznamy vrcholů, hran a stěn. Seznam hran obsahuje datovou strukturu, která obsahuje odkazy do seznamu vrcholů na hrany, které ji tvoří. V seznamu vrcholů jsou pro každý vrchol uloženy souřadnice ve scéně. Dále seznam hran obsahuje odkazy do seznamu stěn na stěny, které hranu svírají. A odkazy na hrany, které ohraničují stěny svírající hranu a jsou k této hraně připojeny. Jako odkazy se pro efektivitu používají indexy do polí.

2.5 3D plochy

3D plocha je definována bázovou maticí a sítí řídících bodů. Plocha se může spojovat ze segmentů. Pro zobrazení se používá převod na polygonální model nebo ray-casting. Např. Beziérovy plochy a NURBS plochy, které obsahují 4x4 řídících bodů.

2.6 Implicitní plochy

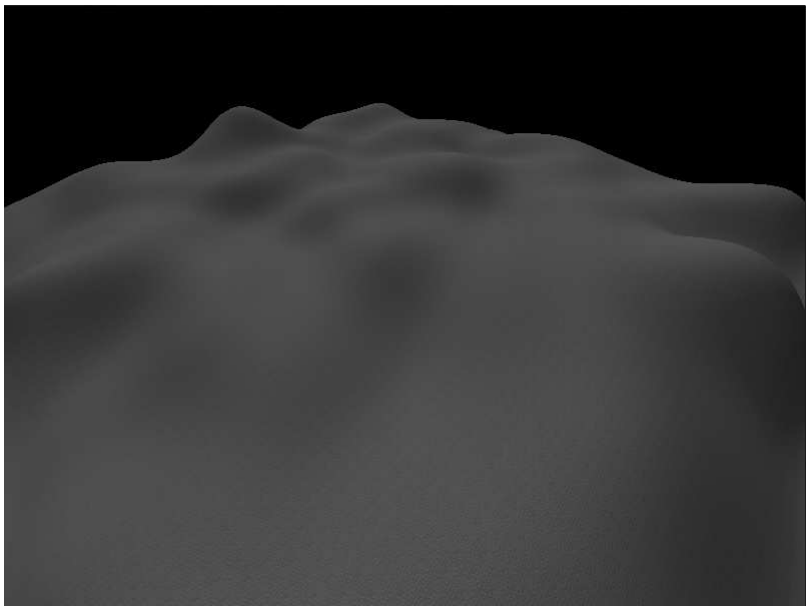
Implicitní plochy jsou tvořeny kostrou objektu, která je tvořena pomocí primitiv (bod, přímka). Kolem každého prvku kostry je potenciální pole. Povrch objektu je v místě, kde je intenzita pole 0. Intenzita pole se zjistí směšovací funkcí, která počítá s potenciálními poli všech prvků kostry. Pro zobrazení pomocí polygonů je nutné převést např. pomocí marching cubes. Při zobrazování ray-castingem nebo ray-tracingem není nutné převádět. Jsou snadno kombinovatelné s CSG stromy.

2.7 Digitální model terénu

Digitální model terénu je model, který reprezentuje holý zemský povrch bez jakýchkoliv objektů jako jsou květiny nebo budovy. Může být reprezentován jako rastr tedy mřížka hodnot. Pokud tyto hodnoty reprezentují výšku nazývá se model výšková mapa. Také může být reprezentován jako triangulated irregular network (TIN). To je vektorově založená reprezentace fyzického povrchu, vytvořená z nepravidelně rozmístěných bodů a čar v trojrozměrném prostoru, které jsou uspořádány v síti nepřekrývajících se trojúhelníků. TIN je často získáván z rastrové reprezentace, jeho výhodou je proměnné uspořádání bodů a proto stačí jejich menší množství.

2.7.1 Výšková mapa

Výšková mapa je dvourozměrná rovina s rovnoměrně rozloženými body, kde každý bod má přiřazenou výšku povrchu od určité roviny. Často je vizualizována jako černobílý obrázek, kde černá reprezentuje minimální výšku a bílá maximální. Při renderování mapy může být měněn kontrast obrázku specifikováním rozdílu výšek mezi dvěma hodnotami výškového kanálu. Při použití černobílého obrázku může být použito pouze 256 stupňů šedi tedy 256 výšek. K ukládání většího množství výšek lze využít barevný obrázek. Výšková mapa může být vytvořena v klasickém malovacím programu, specializovaném programu pro vytváření výškových map nebo může být vygenerována algoritmem pro generování terénu. Výškové mapy umožňují uložení vysokého detailu za použití malého množství paměti. Používají se např. v geografii a počítačových hrách. K zobrazení se často převádí na polygonální síť, viz obrázek 2.1.

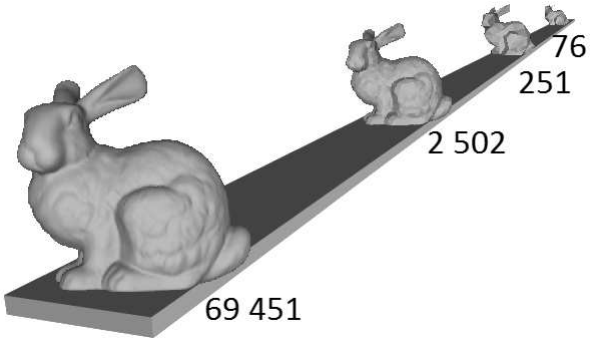


Obrázek 2.1: Výšková mapa, výstup aplikace

Kapitola 3

Level of Detail

Level of detail [7] (dále jen LOD) jsou techniky, které mění úroveň detailu objektu v závislosti na jeho vizuální důležitosti. Počátky sahají do roku 1976, kdy James Clark popsal výhody reprezentace objektů ve scéně v několika rozlišeních, objekty zabírající malý prostor na obrazovce mohou být zobrazeny pomocí jednodušších verzí objektu [4]. LOD je stále aktuální a velmi důležitý v počítačové grafice, protože detailnost objektů se zvyšuje rychleji než výkonnost hardwaru. V začátcích byly nižší úrovně detailu vytvářeny ručně, ale postupem času bylo vyvinuto množství algoritmů, které tyto verze objektu vytvářejí automaticky. Nestačí pouze mít různé verze objektu, je také nutné správně určit, kdy jakou verzi použít. LOD nachází využití především při zobrazování komplexních scén obsahujících množství malých a detailních objektů. Princip LOD je ukázán na obrázku 3.1.



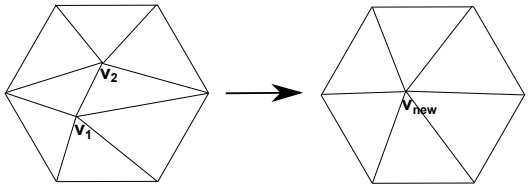
Obrázek 3.1: Objekty ve větší vzdálenosti jsou zobrazeny v menším rozlišení. Čísla udávají počty trojúhelníků modelů, převzato z [7].

3.1 Zjednodušující operátory

K vytvoření jednodušších verzí polygonálních modelu se aplikují zjednodušující operátory [7]. Existují různé chybové metriky pro určení v jakém místě modelu se operátor aplikuje tak, aby model byl co nejméně ovlivněn.

3.1.1 Edge collapse

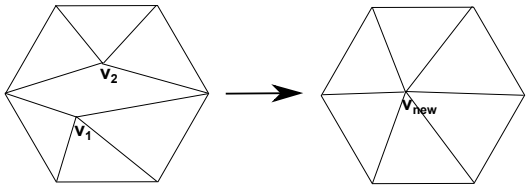
Operátor odstraní hranu a nahradí ji jedním vrcholem. Odstraní se také trojúhelníky obsahující tuto hranu, viz obrázek 3.2.



Obrázek 3.2: Edge collapse operátor

3.1.2 Vertex pair collapse

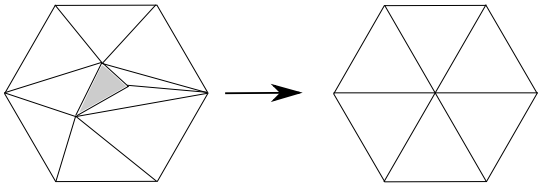
Operátor nahradí dva vrcholy pouze jedním vrcholem. Tyto dva vrcholy nemusí tvořit hranu. Umožňuje vyplnění děr v modelu, viz obrázek 3.3.



Obrázek 3.3: Vertex pair collapse operátor

3.1.3 Triangle collapse

Operátor odstraní celý trojúhelník a nahradí ho jedním vrcholem. Spolu s trojúhelníkem jsou odstraněny i přilehlé trojúhelníky, viz obrázek obrázek 3.4.



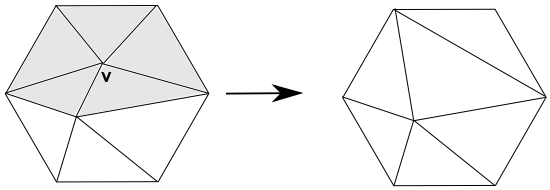
Obrázek 3.4: Triangle collapse operátor

3.1.4 Cell collapse

Sloučí množinu vrcholů v okolí do jednoho vrcholu.

3.1.5 Vertex removal

Operátor odstraní jeden vrchol spolu se všemi připojenými trojúhelníky, výsledná díra je poté zaslepena novými trojúhelníky, viz obrázek 3.5.



Obrázek 3.5: Vertex removal operátor

3.2 Chybové metriky

Chybové metriky [7] slouží k řízení zjednodušování modelů. Chybové metriky se také mohou použít k výběru úrovně detailu, která se použije za běhu aplikace. Chybová metrika může být geometrická, která může být určena jako vzdálenost mezi body nebo povrchy v 3D prostoru nebo na obrazovce. Mnoho moderních algoritmů také určuje chybu z atributu modelu např. z barvy, souřadnic textury nebo normály.

3.2.1 Vrchol – vrchol vzdálenost

Nejjednodušší přístup k měření chyby zjednodušeného modelu je měřit vzdálenost původních vrcholů a zjednodušených vrcholů. Způsob měření této vzdálenosti se liší podle toho, který zjednodušující operátor je aplikován. Tento přístup nemusí zjistit chybu mezi povrchy, například pokud vrcholy zůstanou nezměněné a změní se pouze trojúhelníky mezi nimi. Použitelnost vzdáleností mezi vrcholy je především u operací slučujících vrcholy (edge collapse, vertex collapse).

3.2.2 Vrchol – rovina vzdálenost

Vzdálenost mezi rovinou a vrcholem se počítá efektivněji než vzdálenost mezi dvěma vrcholy. Zjednodušující metody používající tuto vzdálenost jsou rychlé a produkují modely s nízkou chybou.

Kvadratická vzdálenost

Vzdálenost vrcholu od každé roviny se dá spočítat pomocí symetrické matice určující danou rovinu. Díky symetričnosti stačí k uložení této 4x4 matice pouze 10 čísel. Vzdálenost vrcholu od více rovin zároveň lze určit pomocí součtu matic těchto rovin. Každý vrchol modelu má přiřazen matici získanou součtem rovin všech přilehlých trojúhelníků. Pří slučování dvou vrcholů do jednoho se nová pozice vrcholu určí tak, aby jeho vzdálenost od součtu matic těchto dvou vrcholů byla co nejmenší. Novému vrcholu se přiřadí sečtená matice.

3.2.3 Vrchol – povrch vzdálenost

V tomto případě jsou vrcholy originálního modelu namapovány na nejbližší vrcholy polygonů zjednodušeného povrchu. Tento přistup je vhodný pro modely které vznikly triangulací množiny vrcholů. Povrch může být změněn na jiné rozlišení, aby se optimalizovala vzdálenost mezi dvěma povrchy.

3.2.4 Povrch – povrch vzdálenost

Tato metrika uvažuje všechny body z originálního a zjednodušeného povrchu ke zjištění chyby v daném kroku zjednodušování. Tyto metody minimalizují maximální chybu. Využití nachází obzvláště v medicíně a vědecké vizualizaci.

3.3 Výběr úrovně detailu

Za běhu aplikace je potřeba nějaké kritérium výběru úrovně detailu, která se použije pro zobrazení objektu. Při použití view-independent kritéria je rovnoměrně zjednodušen celý objekt, poté může být úroveň detailu určena např. podle vzdálenosti od pozorovatele, množství zabíraného prostoru na obrazovce nebo tak, aby byla udržována konstantní frekvence zobrazování. V posledním příkladu může být použito reaktivní nebo prediktivní plánování. Při použití view-dependent kritéria je model hierarchicky rozdělen na části, např. do stromu. Podle kritéria se vybere část modelu, která se zjednoduší [7].

3.4 Diskrétní LOD

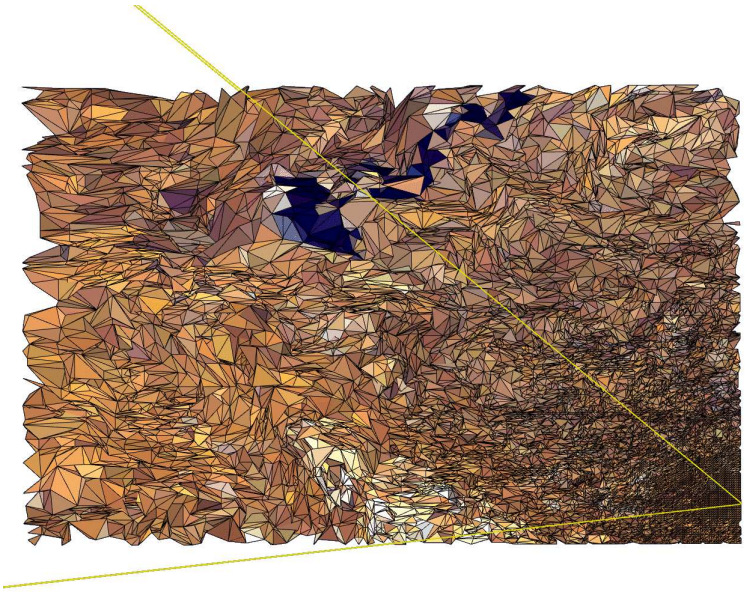
Základním konceptem diskrétního LOD [7] je poskytnutí různých verzí modelu pro reprezentaci jednoho objektu. Tyto verze modelu mohou být vytvořeny ručně nebo algoritmicky. Diskrétní LOD je převážně používán v aplikacích, kde je prioritou co nejvyšší výkon a zároveň obsahují dostatečně málo dat, aby se všechny verze modelu vešly do paměti. Model, který se použije se vybírá v závislosti na vzdálenosti objektu od pozorovatele nebo podobného kritéria. Výhodami tohoto přístupu je jednoduchost na implementaci a oddělení zjednodušování modelů od renderování. Tedy rychlost tvorby modelů nemusí být tak rychlá, aby se modely stíhaly tvořit během renderování a během renderování se pouze určuje jaký model bude použit. Další výhodou může být možnost převodu jednotlivých modelů na reprezentaci pomocí pruhů trojúhelníků, které se renderují mnohem rychleji než neorganizované trojúhelníky. Nevýhodou je nevhodnost při použití u příliš velkých modelů, které se musí rozdělit na části nebo malých modelů, které se používají ke skládání větších, kdy nemusí být dodržena návaznost sousedních částí.

3.5 Spojité LOD

Konceptem spojitého LOD [7] je vytvoření datové struktury, ze které může být výsledný model vytvořen za běhu aplikace. Výhodami je tvorba úrovně detailu modelu přesně v závislosti na potřebách aplikace a objekty tedy nepoužívají více polygonů než je potřeba a tím uvolňují místo pro další objekty. Oproti diskrétnímu LOD odpadá problém skokových přechodů mezi modely. Nevýhoda je vyšší složitost a výpočetní náročnost.

3.6 View-Dependent LOD

Rozšíření spojitého LOD, které zobrazuje části modelu blíže ke kameře s vyšším detailem, než části dále. V rámci jednoho objektu tedy může být použito více úrovní detailu. Nachází využití u velkých modelů, které mají velký rozsah vzdálenosti. Příkladem může být výšková mapa na obrázku 3.6. Může se také použít menší úroveň detailu modelu, na který se uživatel pravděpodobně dívá periferně. Výhoda je ještě lepší využití polygonů a vyšší přesnost, než u spojitého LOD [7].



Obrázek 3.6: View dependent LOD aplikován na výškovou mapu, převzato z [7].

3.7 LOD terénu

Terén může být jednodušším případem než jiné 3D modely, protože má omezenou geometrii a skládá se z rovnoměrně rozložených bodů. Na druhou stranu spojitost terénu umožňuje velké množství terénu viditelné v jeden okamžik. Proto se používá především view-dependent LOD [7].

3.7.1 Spojitý LOD pro výškové mapy

Algoritmus začíná s modelem v nejvyšším rozlišení a postupně snižuje počet trojúhelníků. Síť je rozdělena do obdélníkových bloků, kde se odstraňují vrcholy. Když je vrchol odstraněn provedou se spojení nebo rozdělení trojúhelníků. Který vrchol bude odstraněn je určeno podle vzdálenosti na obrazovce dvou povrchů spojených tímto bodem.

3.7.2 ROAM algoritmus

ROAM algoritmus používá přístup se zvyšující se prioritou a binárním stromem trojúhelníků. Výsledná síť je vytvořena aplikováním série slučovacích a rozdělovacích operací nad

dvojicemi trojúhelníků. Algoritmus používá dvě prioritní fronty pro slučující a rozdělující operace. Fronta slučujících operací slouží k zjednodušování terénu, fronta s rozdělujícími operacemi k vylepšování terénu. K řazení těchto front je použita velikost na obrazovce.

3.7.3 Progressive meshes

Progressive mesh je struktura, která je vytvořena při zjednodušování původního modelu na model s minimální úrovní detailu. Výsledný model je model s minimálním detailem, na který je aplikována sekvence inverzních zjednodušovacích operací. Model je zjednodušován operátorem edge collapse, při kterém se ukládají informace o odstraněných hranách. Inverzní operace poté umožňuje tuto hranu znovu vytvořit.

Kapitola 4

OpenGL

OpenGL [8] je programové rozhraní grafického hardwaru. Bylo vytvořeno firmou Silicon Graphics v roce 1992. Příkazy tohoto rozhraní se používají pro určení objektů a operací potřebných k vytvoření interaktivních trojrozměrných aplikací. OpenGL je nezávislé na programovacím jazyce a hardwaru, je tedy použitelné na mnoha platformách. Pracuje pouze se základními tvary (body, přímky, polygony), ze kterých se vytvářejí tvary složitější. OpenGL je založeno na architektuře klient-server. Program (klient) vydává příkazy, které grafický adaptér (server) vykonává. Díky tomu je možné, aby program běžel na jiném počítači, než na kterém se příkazy vykonávají.

Posloupnost operací vedoucí k zobrazení objektu je následující:

1. **Popis objektu**. Objekt je popsán pomocí vertex buffer objektů (VBO), které umožňují ukládat pole vrcholů na paměť grafické karty. VBO popisující jeden objekt se sdružují do vertex array objektů (VAO). Například pro čtverec v 3D prostoru budou za sebe do pole uloženy souřadnice jeho čtyř vrcholů. Každý vrchol určují x, y a z souřadnice celkem tedy 12 čísel. K těmto souřadnicím ale mohou být přidány např. souřadnice textury t_x a t_y. Je nutné pro každou složku určit, který vstup vertex shaderu ji bude přijímat. Do VAO se také ukládá VBO s indexy, které jsou uloženy v závislosti na tom co budeme zobrazovat (např. úsečky nebo trojúhelníky). Pokud budeme zobrazovat čtverec pomocí trojúhelníků, uložíme do VBO indexy vrcholů, které určují dva trojúhelníky tvořící čtverec, např. 0, 1, 2, 0, 2, 3.

2. **Transformace**. Transformace se provádí ve vertex shaderu. Shadery jsou programy napsané v jazyku GLSL a běžící na grafické kartě. Program je proveden pro každý vrchol objektu. Vertex shader dostává jako vstup vrcholy z VAO a transformační matice. Jeho výstupem je pozice bodu ve světových souřadnicích.

3. **Rasterizace**. Rasterizace je převedení primitiv na dvojrozměrný obraz. Určí se jaká primitiva budou viditelná a jaké pixely jimi budou zabrány.

4. **Výpočet barev**. Barvy se počítají ve fragment shaderu. Fragment shader dostává od vertex shaderu informace o vrcholech primitiv, jako je barva nebo normála. Tyto informace využívá k výpočtu barvy v konkrétním pixelu.

OpenGL zobrazuje pouze to, co je mu předáno. Neposkytuje tedy žádný mechanismus pro změnu úrovně detailu. Režie úrovně detailu musí být prováděna mimo OpenGL. OpenGL je poté předán různý model v závislosti na tom, s jakou úrovní detailu chceme objekt zobrazit.

Aby se nemusely provádět výpočty pro objekty, které nebudou vidět, poskytuje OpenGL mechanismus occlusion query. Ten umožňuje aplikaci zjistit kolik pixelů na obrazovce primitivum nebo skupina primitiv zabírá. Typické použití je vykreslení velkých objektů, které budou pravděpodobně zakrývat ostatní, menší objekty. Poté se pro ohraničující boxy menších objektů zjistí kolik pixelů zabírají. Objekt je vykreslen pouze pokud jeho ohraničující box zabírá alespoň jeden pixel.

Kapitola 5

Návrh a realizace řešení

V této kapitole bude popsáno řešení práce a budou vysvětleny použité algoritmy. Na začátku bude popsána tvorba terénu, následně vytvoření dráhy a nakonec řešení level of detail. K implementaci je použit jazyk C++, OpenGL 3.0 a GLSL verze 330.

5.1 Generování terénu

Na začátku se vytvoří jednoduchá výšková mapa v 3D prostoru. Všechny body v této mapě mají nulovou výšku. Mapa se bude zobrazovat jako polygonální model, proto se musí mezi body nadefinovat hrany a vytvořit mezi nimi trojúhelníky. Podstatou generování terénu bude nastavování výškové souřadnice jednotlivým bodům mapy pomocí kruhového algoritmu, viz kapitola 5.1.1.

Po vygenerování může terén obsahovat příliš ostré hrany, k jejich vyhlazení je možné použít vyhlazovací filtr. Filtr v každé iteraci projde všechny řádky a sloupce v obou směrech. Pro každý bod se jeho nová výška vypočítá pomocí výšky souseda. Soused se bere podle toho, jakým směrem se prochází. Nová výška h_{new} se vypočítá následovně. Nechť aktuální výška je h a výška souseda je h_n, potom $h_{new} = h * (1 - k) + h_n * k$, kde k je konstanta nabývající hodnot 0 až 1. Čím je konstanta vyšší, tím menší změny filtr provede [2].

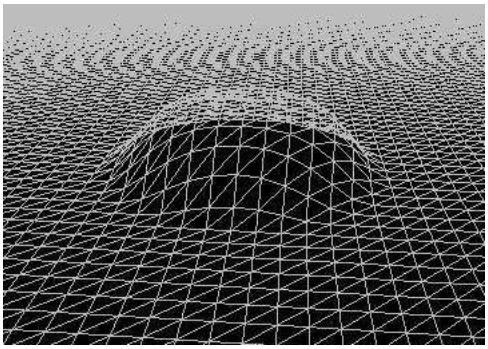
Je nutné vypočítat normály výškové mapy pro použití při výpočtu osvětlení. Pro každý vrchol výškové mapy se spočítají normály všech přilehlých trojúhelníků. Normála trojúhelníku se spočítá jako kartézský součin dvou vektorů vycházejících z jednoho bodu trojúhelníku do bodů sousedních. Normály se poté znormalizují, sečtou a výsledek se znovu znormalizuje. Díky započítání všech trojúhelníků okolo vrcholu bude osvětlení plynulé. Také se vrcholům výškové mapy přidají souřadnice textury tak, aby na dva trojúhelníky tvořící čtverec byla namapována jedna textura. Výsledný vygenerovaný terén je vidět na obrázku 2.1.

5.1.1 Kruhový algoritmus

Kruhový algoritmus v každé iteraci zvýší body v určité oblasti o určitou hodnotu. Oblast je kruh s náhodně vygenerovaným středem a daným průměrem. Bod ve středu kruhu je zvýšen o maximální hodnotu, body dále od středu směrem k okraji kruhu jsou zvýšeny o menší hodnotu podle kosinové funkce. Body mimo kruh zůstávají nezměněné. Je-li bod uvnitř kruhu a jeho vzdálenost od středu je d, poloměr kruhu je r a $disp$ je maximální hodnota zvýšení, potom $p = d/r$ a výška $h = h + disp/2 + \cos(p * \pi) * disp/2$. Průměr kruhu

a maximální hodnota zvýšení se zmenšují s každou iterací. Místo kosinové funkce je možné použít i jiné funkce např. sinus [2].

Protože procházet všechny body výškové mapy a porovnávat jejich vzdálenost se středem kruhu je příliš pomalé, prochází se pouze body, které jsou v rozsahu kruhu a jejich výška bude změněna. Spočítá se index bodu, jehož souřadnice jsou nejblíže středu kruhu a poté se spočítá počet řádků a sloupců, které se budou od středu každým směrem procházet, tak aby vzniklý čtverec obsahoval vygenerovaný kruh.



Obrázek 5.1: Jedna iterace algoritmu, převzato z [2]

5.2 Definovaní dráhy

Když máme vygenerovaný terén, můžeme určit kudy dráha povede. Cílem je vytvořit kruhovou dráhu, tak aby vedla vhodným terénem. Dráha bude určena jako Catmull-Rom křivka. K určení křivky je potřeba mít vhodné kontrolní body, ty se vygenerují algoritmem dart throwing.

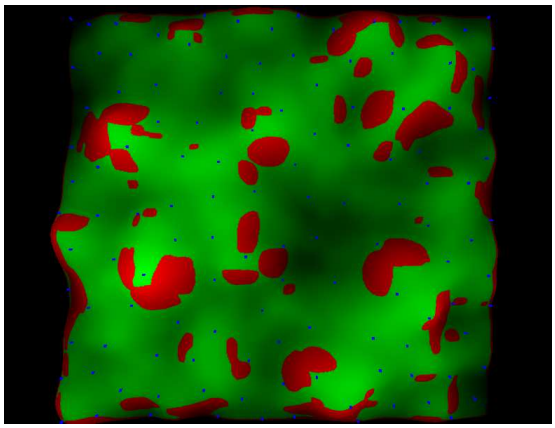
5.2.1 Sjízdnost terénu

K vygenerování závodní dráhy je nutné klasifikovat sjízdné a nesjízdné oblasti terénu. Pro každý vrchol mapy se prochází okolí s definovanou velikostí. V daném okolí se zkontroluje sklon mezi všemi sousedy. Sklon mezi dvěma body se určí podle úhlu trojúhelníku, který tvoří výškové souřadnice dvou porovnávaných bodů. Proto aby byl vrchol mapy označen jako sjízdný musí být úhel mezi všemi sousedy v jeho okolí menší než hraniční úhel. Je-li sklon mezi některými sousedy v okolí větší než maximální sjízdný sklon, je bod označen jako nesjízdný. Důvod pro zavedení tohoto okolí je, že pozice modelu na mapě je určena jeho středem a je proto vhodné zajistit, že i okraje modelu budou na sjízdné oblasti. Sjízdné a nesjízdné oblasti jsou zobrazeny na obrázku 5.2.

5.2.2 Dart Throwing

Dart throwing algoritmus generuje rovnoměrně rozložené body a odmítá body, které nesplňují minimální vzdálenost od již vygenerovaných bodů. Tento proces pokračuje dokud již nemůže být přidán žádný další bod. Dart throwing je široce používán a je jednoduchý na implementaci. Nicméně je pomalý a obtížný na kontrolu - místo specifikování počtu bodů je algoritmu poskytnuta minimální vzdálenost mezi body. Ale pro použití v tomto případě je vhodný [6].

Na začátku se vytvoří pole všech sjízdných vrcholů. Poté se vybere náhodný bod z pole a uloží se. Tento bod se následně odstraní z pole spolu se všemi body, jejichž vzdálenost od tohoto bodu je menší než zadaná minimální vzdálenost mezi dvěma body. Takto se pokračuje dokud není pole prázdné. Vygenerované body jsou vidět na obrázku 5.2.



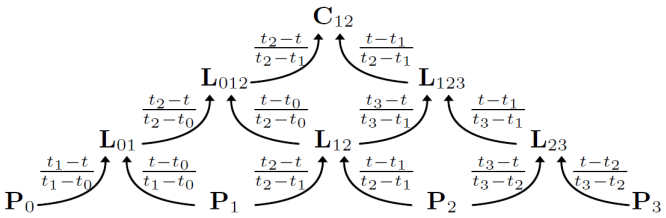
Obrázek 5.2: Výšková mapa s vygenerovanými body (modře) na sjízdné oblasti (zeleně), výstup aplikace

5.2.3 Catmull-Rom spline

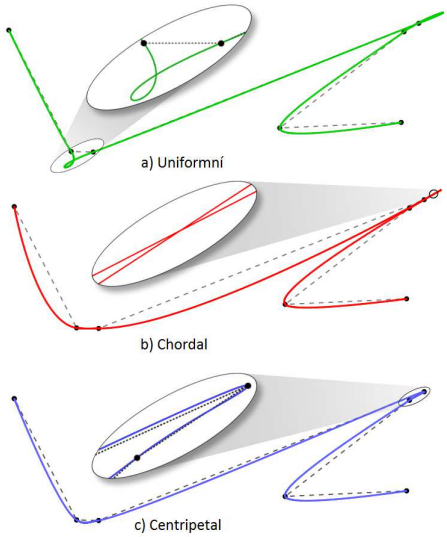
Catmull-Rom křivky jsou interpolační křivky (procházejí svými kontrolními body) široce používané v grafice při modelování nebo animaci. Jejich každý kontrolní bod ovlivňuje pouze malé okolí, to umožňuje vysokou kontrolu nad tvarem křivky, která je potřeba pro zajištění průchodu dráhy přes sjízdné oblasti. Křivka je reprezentována po částech jako množiny bodů. V této práci je použit kubický spline, tedy body jsou čtyři. Spojitost křivky se zajistí opakováním koncových bodů. Pro Catmull-Rom spline existují tři základní parametrizace, které se liší způsobem jakým pracují s časem: uniformní, chordal a centripetal. Asi nejčastěji používanou parametrizací je uniformní. Při použití této parametrizace docházení ke křížení křivky, pokud jsou různě dlouhé segmenty. Protože jsou kontrolní body generovány náhodně, k tomuto křížení docházet může. V této práci je použita centripetal, u které je dokázáno, že u ní jako jediné ke křížení nedochází [9].

Nechť $\mathbf{P}_i$ jsou kontrolní body s asociovanou parametrickou hodnotou t_i. Pro kubický spline $i = 0, 1, 2, 3$, kde mezi $\mathbf{P}_1$ a $\mathbf{P}_2$ se počítají body křivky. Výsledný bod pro parametr t se spočítá podle vzorce na obrázku 5.3. V pyramidě se vynásobí každý bod koeficientem na šipce a sečte se s bodem vedle, tento součet je bod v další úrovni pyramidy. Nejvyšší bod $\mathbf{C}_{12}$ je výsledný bod. Parametry t se spočítají podle vzorce: $t_{i+1} = \|\mathbf{P}_{i+1} - \mathbf{P}_i\|^{\alpha} + t_i$, kde $\alpha = 0$ pro uniformní, $\alpha = 0.5$ pro centripetal a $\alpha = 1$ pro chordal [9].

Mapa je rozdělená do $4 * \mathbf{N}$ oblastí, kde $\mathbf{N}$ je počet oblastí v každém kvadrantu, např. viz obrázek 5.5. Je-li bod příliš blízko okraje ignoruje se, není vhodné aby dráha vedla na okraji mapy. Z každé oblasti se vybere jeden náhodný bod. Každý úsek křivky vedoucí mezi dvěma sousedními oblastmi je definován dvěma body z těchto oblastí a dalšíma dvěma body z jim přilehlých oblastí. Mezi dvěma body, kde se počítá úsek, se určí vzdálenost a podle ní se určí hodnoty parametru t tak, aby vznikl jeden bod na každou délku modelu, který se na zobrazení dráhy použije. Pokud výsledný úsek křivky vede přes nesjízdnou



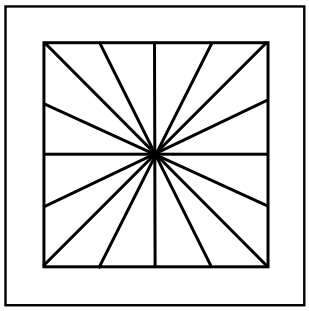
Obrázek 5.3: Vzorec pro výpočet kubické Catmull-Rom křivky, převzato z [9]



Obrázek 5.4: Unifromní, chordal a centripetal Catmull-Rom spline, převzato z [9]

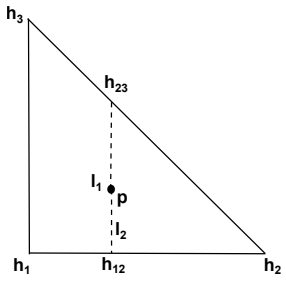
oblast, vybere se jiný bod ze stejné oblasti. Pokud se vyzkouší všechny kombinace ze čtyř sousedních oblastí a žádná jimi definovaná křivka nevede pouze přes sjízdnou oblast, použije se poslední kombinace a křivka tedy povede přes nesjízdnou oblast. Tento průchod slouží pouze k určení kontrolních bodů. Křivka bude obsahovat více bodů a kontrolovat a zkoušet kombinace kontrolních bodů by bylo pomalé, proto se kontrolní body určí při výpočtu křivky s většími rozestupy t. Nyní se vytvoří křivka s použitím získaných kontrolních bodů s vysokým počtem bodů na každou délku modelu.

Tímto způsobem je získána dráha v 2D prostoru, je nutné dopočítat výškové souřadnice tak, aby křivka kopírovala povrch výškové mapy. Mapa je tvořena trojúhelníky, které jsou oproti vzdálenosti mezi sousedními body křivky velké. Pokud by se dopočítala pouze výška trojúhelníku v místě, které je nad bodem, křivka by obsahovala ostré hrany. Proto je potřeba spočítat výškové souřadnice pomocí Catmull-Rom splinu. Jako kontrolní body se vyberou body z již spočítané 2D křivky tak, aby jejich rozestup odpovídal rozestupu vrcholů mapy a spočítá se jejich výšková souřadnice podle povrchu mapy. Pokud by rozestup byl menší, vznikaly by pořád ostré hrany. Jestliže by byl větší, křivka by nekopírovala terén dostatečně přesně. Najdou se tři body mapy, které tvoří trojúhelník nad kontrolním bodem a určí se výška, viz obrázek 5.6. Spočítá se výška h_{12} a h_{23} a s pomocí vzdáleností l_1 a l_2 se dopočítá výška v bodě p. Pro každý bod mezi kontrolními body $\mathbf{P}_1$ a $\mathbf{P}_2$ se spočítá vzdálenost od



Obrázek 5.5: Rozdělení výškové mapy na oblasti, ze kterých se vybírají body pro definici křivky.

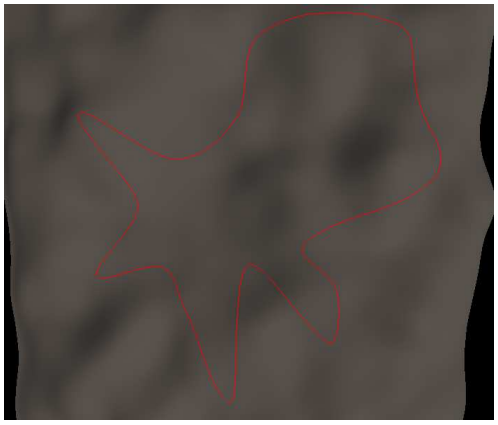
$\mathbf{P}_1$ a hledá se parametr t takový, aby výsledný bod křivky této vzdálenosti odpovídal. Je-li vzdálenost bodu pro který chceme spočítat výškovou souřadnici různá od vzdálenosti bodu, který dostaneme od Catmull-Rom křivky, vynásobí se t podílem těchto vzdáleností. Tímto způsobem se přibližujeme odpovídajícímu t, dokud vzdálenosti neodpovídají. Poté se t použije pro výpočet výškové souřadnice jednoho bodu. Vytvořená křivka na výškové mapě je zobrazena na obrázku 5.7.



Obrázek 5.6: Výpočet výšky bodu nacházajícího se na mapě.

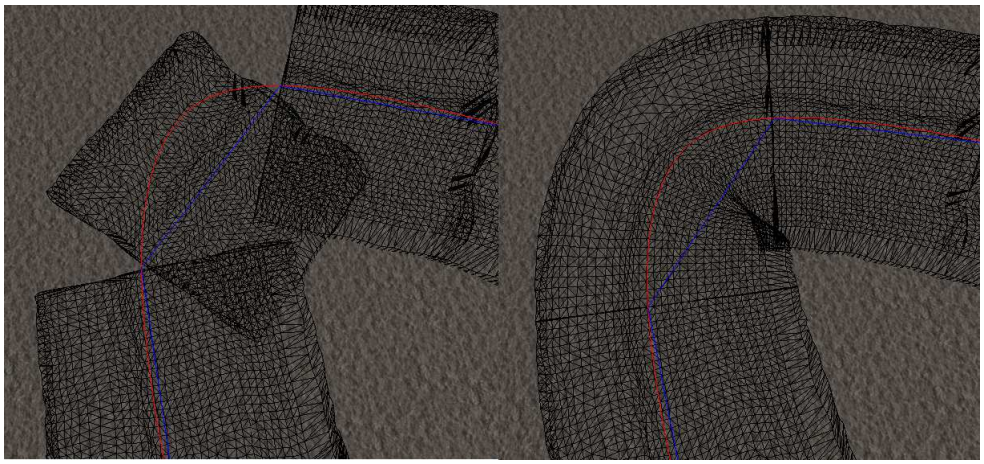
5.2.4 Namapování modelu na křivku

Aby se dal jeden model použít k zobrazení celé dráhy, je nutné ho zakřivit podél křivky. Zakřivení se provádí ve vertex shaderu. Předtím se křivka musí rozdělit na úseky, jejichž vzdálenost prvního a posledního bodu je délka d modelu podél osy Z. Na začátku je začátečním bodem úseku první bod křivky. Prochází se další body křivky a porovnává se jejich vzdálenost od začátečního bodu. Je-li vzdálenost větší než d, určí se bod mezi posledními dvěma body takový, aby jeho vzdálenost od začátečního bodu byla d. Tento koncový bod daného úseku bude začátečním bodem dalšího úseku. Každému úseku dráhy je přiřazena tato část křivky, která se předá jako parametr vertex shaderu, když se daný úsek zobrazuje. Tato křivka se ještě před tím upraví tak, aby vzdálenost mezi sousedními body na křivce byla rovnoměrná. Catmull-Rom křivka nemá rovnoměrně rozložené body. Při větším zakřivení jsou body hustější než při menším. Dva body úseku křivky, které určují úsečku, na kterou se bude konkrétní vrchol modelu mapovat se zjistí jednoduchým přepočítáním pozice na ose Z modelu na indexy bodů křivky, to je možné, protože body na křivce jsou rovnoměrně rozložené.



Obrázek 5.7: Křiva definující dráhu na vygenerované výškové mapě, výstup aplikace

Nová pozice vrcholu modelu se určí jako otočení kolmice k ose Z podle úhlu, který svírá osa Z s určenou úsečkou křivky. Rotace se provádí pomocí kvaternionů, jeden kvaternion je použit pro otočení okolo osy Y a druhý okolo osy X. Po otočení se vrchol posune o vzdálenost mezi bodem na ose Z a odpovídajícím bodem na určené úsečce křivky. Na obrázku 5.8 je původní model a model po namapování na křivku. Je nutné také přepočítat normály, aby odpovídaly změnám v modelu. Nová normála se získá otočením staré normály pomocí stejných kvaternionů jako se otáčel vrchol. Normála se nemusí posunovat, pouze se otočí.



Obrázek 5.8: Část dráhy vytvořená z původních modelů. Modře jsou označeny segmenty, které se zobrazí jedním modelem, červeně je křivka.

5.3 Level of Detail

Pokud je funkční zobrazení dráhy s jedním modelem, můžeme přidat techniku level of detail. Je použito diskrétní LOD. Budou se používat čtyři úrovně detailu, z nichž nejvyšší je původní model. Modely se zjednodušují na začátku programu a uloží se do paměti. Podle množství aktuálně zobrazovaných objektů se vyberou vhodné modely tak, aby s přibýva-

jící vzdáleností od pozorovatele ubývala úroveň detailu. Množství aktuálně zobrazovaných objektů se zjistí technikou view frustum culling, viz kapitola 5.3.1. Počet trojúhelníků použitých k zobrazení dráhy je omezen maximální hodnotou **max**. Tato maximální hodnota může být překročena, jestliže jsou všechny objekty zobrazeny nejnižší úrovní detailu a je jich tak mnoho, že počet trojúhelníků je vyšší. Každá úroveň detailu je použita minimálně jednou (pro 4 a více objektů), tzn. že nejbližší objekt (nad kterým se nachází kamera) bude vždy zobrazen nejvyšší úrovní detailu a každý následující bude mít úroveň detailu maximálně o jednu úroveň nižší.

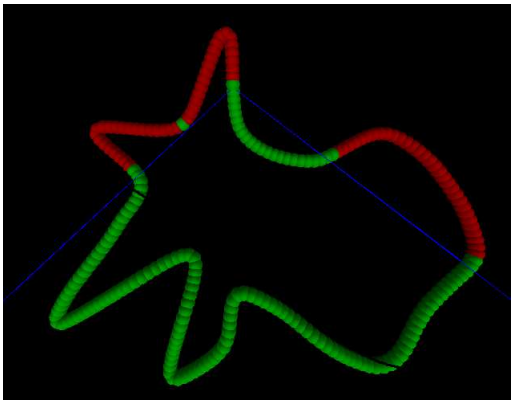
Pro určení úrovní detailu objektů se spočítá vzdálenost každého zobrazovaného objektu od pozice kamery a podle této vzdálenosti se objekty seřadí. K řazení se využívá binární vkládání do seřazeného pole. Vzdálenosti jsou rozděleny do čtyř intervalů, které jsou určeny $lodDist1$, $lodDist2$ a $lodDist3$. Tyto vzdálenosti se iniciují na $lodDist1 = \mathbf{d}$, $lodDist2 = 2 * \mathbf{d}$ a $lodDist3 = 3 * \mathbf{d}$, kde $\mathbf{d}$ je délka jednoho segmentu dráhy. Jsou definovány tři inkrementy **inc1**, **inc2** a **inc3**, které zvyšují tyto vzdálenosti. Počet objektů v těchto čtyřech intervalech se definuje jako $o1$, $o2$, $o3$ a $o4$, počet trojúhelníků čtyř úrovní detailu modelu jako $t1$, $t2$, $t3$ a $t4$. Inkrementy se aplikují dokud platí $o1*t1+o2*t2+o3*t3+o4*t4 < \mathbf{max}$.

Mezi jednotlivými segmenty dráhy musí být zajištěna návaznost. Nechť f_i a b_i jsou přední a zadní strana modelu a i je číslo segmentu dráhy. Aby segmenty navazovaly, musí platit $f_i = b_{i+1}$. Původní model je vytvořen tak, aby se f rovnalo b. Při vytváření nižších úrovní se toto zajistí tím, že při zjednodušování f se provede ekvivalentní operace i na b a naopak. Tím je zajištěna návaznost pokud je pro zobrazení celé dráhy použita jedna úroveň detailu. Při použití více úrovní detailu nenavazují přechody mezi úrovněmi. Přechody mohou být tři různé z vyšší úrovně na nižší a tři z nižší na vyšší. Pro každou úroveň detailu kromě té nejvyšší budou vytvořeny další dva modely. Jeden bude mít shodnou f s danou úrovní a b s vyšší úrovní, druhý bude mít opačné f a b. Celkem tedy bude uloženo v paměti 10 modelů, z toho 6 přechodových. Přechody mezi úrovněmi detailu jsou zobrazeny na obrázku 5.10.

5.3.1 View frustum culling

View frustum culling je jedna z technik, které se používají k urychlení zobrazení scény tím, že se k renderování předávají pouze viditelné části scény. Další techniky jsou ořezávání částí modelu odvrácených od pozorovatele a Occlusion culling, který ořezává modely zakryté jinými modely. V této práci je použit pouze View frustum culling, jehož cílem je vybrání pouze modelů, které se nachází uvnitř komolého jehlanu určeného projekční maticí. Na obrázku 5.9 jsou zeleně objekty uvnitř a červeně vně komolého jehlanu. Tato technika má smysl pro větší množství malých modelů. Pro velké modely, které jsou skoro vždy vidět urychlení poskytnout nemusí [3].

Existuje několik přístupů jak modely vybírat. Zde je použit přístup, který určí rovnice všech šesti stran komolého jehlanu z projekční matice. Každý objekt má určený střed S a kouli se středem S a poloměrem r, která ho celý obklopuje. Pro každý objekt se porovná pozice středu vzhledem ke každé rovině. Je-li střed na vnější straně všech rovin a jeho vzdálenost od nich je větší než poloměr koule r, je objekt vně komolého jehlanu a tedy nebude renderován. Je-li střed na vnitřní straně některé roviny nebo vzdálenost na vnější straně je menší než r, zasahuje objekt do jehlanu a bude tedy renderován [3].



Obrázek 5.9: View frustum culling. Modrou barvou je vyznačeno view frustum. Zelenou sféru mají objekty uvnitř, červenou vně. Výstup aplikace

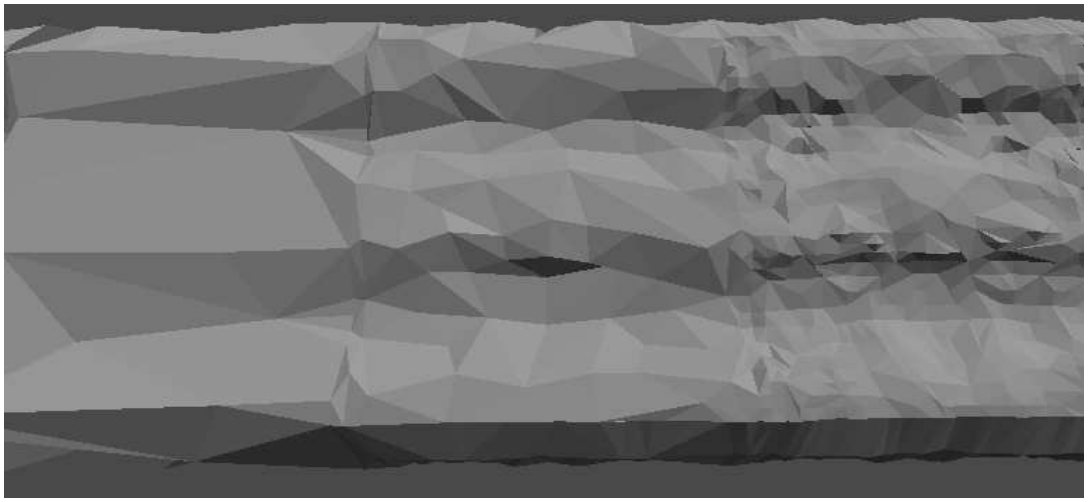
5.4 Tvorba nižších úrovní detailu modelu

Jednodušší úrovně detailu se vytvoří algoritmem kontrakce hran s určením chyby pomocí kvadratické vzdálenosti, viz kapitola 5.4.1. Spolu s tvorbou nižší úrovně detailu modelu se paralelně zjednodušují i jeho dvě přechodové verze. Jestliže se odstraňuje hrana v místě spoje provede se ekvivalentní operace i na druhé straně modelu. Když se odstraní hrana na spoji modelu, je nutné dodržet návaznost přechodových modelů. U jednoho se provede ekvivalentní operace na přední straně a u druhého na zadní. U těchto modelů bude jedna strana nezměněna, tudíž bude navazovat na vyšší úroveň detailu a druhá bude navazovat na nižší, právě vytvářenou verzi modelu. Při každém odstranění hrany uprostřed modelu se inkrementuje počítadlo odstranění hran. Po dosáhnutí požadovaného počtu trojúhelníků modelu se stejný počet hran uprostřed odstraní i u přechodových modelů, tyto modely potom budou mít skoro stejný počet trojúhelníků, akorát budou mít více detailu na přední nebo zadní straně.

Takto vytvořené nové modely nemají určené normály, ty je nutno dopočítat tak, aby na sebe navazovaly segmenty dráhy. K tomu se spočítají normály všech trojúhelníků v modelu a přičtou se ke všem jeho vrcholům. Pokud trojúhelník obsahuje vrchol na předním nebo zadním okraji, přičte se jeho normála i k odpovídajícímu vrcholu na druhé straně. Po znormalizování normály každého vrcholu vznikne průměrná normála, osvětlení bude potom plynulé. K vypočítání normály se použije kartézský součin dvou vektorů směřujících z jednoho vrcholu trojúhelníku k dalším dvěma. Podle pořadí vektorů při výpočtu kartézského součinu může normála směřovat jedním nebo opačným směrem. Aby normála směřovala správným směrem porovnají se obě normály s normálou původního modelu trojúhelníku na stejném místě a vybere se ta, která s ní svírá menší úhel. U přechodových modelů se nahradí normály vrcholů na předním a zadním spoji normálami jim odpovídajících modelů.

5.4.1 Kontrakce hran s určením chyby pomocí kvadratické vzdálenosti

Kontrakce hran s určením chyby pomocí kvadratické vzdálenosti iterativně odstraňuje vybrané hrany operátorem edge collapse, viz kapitola 3.1.1. Pro výběr hrany k odstranění je potřeba zjistit chybu kontrakce a vybrat hranu s nejnižší chybou. K tomu se použije kvadratická vzdálenost. Ke každému vrcholu se přiřadí symetrická 4x4 matice $\mathbf{Q}$. Pro kaž-



Obrázek 5.10: Tři segmenty dráhy. Modely jsou vytvořené z modelu vlevo na obrázku 5.14. Zprava doleva mají modely 1700, 600 a 100 trojúhelníků. Není zobrazena textura a osvětlení není plynulé.

dou hranu definovanou vrcholy $\mathbf{v}_1$ a $\mathbf{v}_2$ se vypočítá nová matice $\overline{\mathbf{Q}} = \mathbf{Q}_1 + \mathbf{Q}_2$. Chyba při nahrazení hrany vrcholem $\mathbf{v}$ se spočítá jako $\Delta(\mathbf{v}) = \mathbf{v}^\intercal \overline{\mathbf{Q}} \mathbf{v}$ [5].

Na začátku algoritmus zjistí matice $\mathbf{Q}$ všech bodů následovně. Ke každému bodu přiřadí všechny trojúhelníky, kterých je součástí. Dále určí rovnici plochy pro každý trojúhelník. Rovnice se určí tak, aby platilo $ax + by + cz + d = 0$, kde $a^2 + b^2 + c^2 = 1$. Tedy a, b, c obsahuje souřadnice normály a d je $-(n \cdot V_0)$, kde V_0 je bod ležící na ploše, tedy libovolný bod trojúhelníku. Chyba se spočítá jako suma kvadratické vzdálenosti vrcholu od všech ploch (trojúhelníků). Vzdálenost od plochy může být zapsána jako matice $\mathbf{K}_p = \mathbf{p}\mathbf{p}^\intercal$, kde $\mathbf{p}$ je plocha. Suma matic $\mathbf{K}_p$ odpovídá matici $\mathbf{Q}$, která umožňuje určit vzdálenost bodu od všech ploch dohromady jedním výpočtem [5].

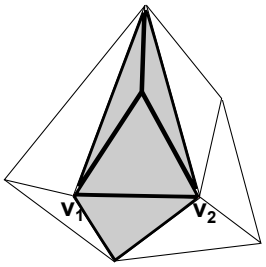
Dále se prochází všechny trojúhelníky modelu a pro každou z jeho hran se vybere nový vrchol $\mathbf{v}$ z $\mathbf{v}_1$, $\mathbf{v}_2$ a $(\mathbf{v}_1 + \mathbf{v}_2)/2$ podle toho, který má nejmenší chybu. Pro vrchol $\mathbf{v}$ se určí také nová souřadnice textury. Hrana se poté uloží do seřazeného seznamu, kde klíčem je chyba kontrakce. K vyhledávání se používá algoritmus binární vyhledávání v seřazeném poli, jehož složitost je v nejhorším případě logaritmická. Pokud se hrana v seznamu již nachází, přidají se k ní druhé souřadnice textury. Jedna hrana sdílí dva trojúhelníky a může tedy mít různé souřadnice textury v závislosti na tom, v rámci kterého trojúhelníku se uvažuje.

Poté se iterativně odstraňuje hrana s nejmenší chybou kontrakce. Při odstranění se vrcholu $\mathbf{v}_1$ dají nové souřadnice a všechny výskyty vrcholu $\mathbf{v}_2$ se nahradí vrcholem $\mathbf{v}_1$. Každá hrana obsahuje dva páry souřadnic textur, které mohou být různé. Pro každý pár je spočítána souřadnice pro nový vrchol. Je nutné vybrat tu souřadnici, která je spočítána z páru, jenž má jednu souřadnici textury shodnou s měněným vrcholem. Také se aktualizuje chyba všech hran obsahujících vrchol $\mathbf{v}_1$.

Při vybírání hran k odstranění se ignorují hrany, které mají jeden vrchol v místě spoje a druhý uprostřed, i když mají nejnižší cenu kontrakce. Důvodem je, že takovéto hrany mohou při určitém rozložení sousedních hran, které nastává asi v polovině těchto případů, vytvářet nové hrany na místě spoje. Takto může na jedné straně vzniknout hrana, která na druhé straně nevznikne a modely na sebe nebudou navazovat.

Dále je vhodné odstranit přebytečné trojúhelníky, které mohou vzniknout pokud je rozložení trojúhelníků jako na obrázku 5.11 a je odstraňována hrana mezi vrcholy $\mathbf{v}_1$ a $\mathbf{v}_2$. V tomto případě zůstanou dva shodné trojúhelníky, které sdílí hranu s dalšími dvěma trojúhelníky a tvoří tedy nemanifoldní geometrii.

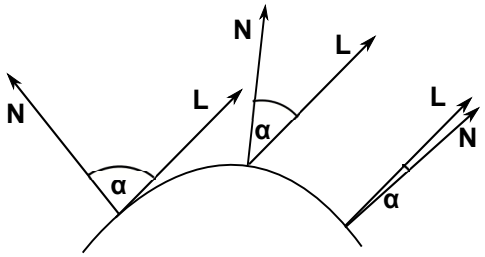
Modely vytvořené tímto algoritmem jsou na obrázku 5.10.



Obrázek 5.11: Vznik přebytečných trojúhelníků. Je potřeba odstranit čtyři a ne pouze dva trojúhelníky.

5.5 Osvětlení

Osvětlení je spočítáno jako součet světel ze čtyř zdrojů s různou intenzitou. Zdroj světla je určen vektorem směru. Barva odraženého světla se určí pomocí Lambertovského osvětlovacího modelu [1]. Odražené světlo závisí na kosinu úhlu mezi normálovým vektorem povrchu $\mathbf{N}$ a paprskem příchozího světla $\mathbf{L}$. Normálový vektor povrchu je kolmý k povrchu v místě dopadu světla, je spočítaný pro konkrétní pixel ve fragment shaderu. Kosinus úhlu mezi normalizovanými vektory $\mathbf{N}$ a $\mathbf{L}$ se zjistí jako skalární součin $\mathbf{N} \cdot \mathbf{L} = \|\mathbf{N}\|\|\mathbf{L}\| \cos \alpha$, kde α je úhel mezi $\mathbf{N}$ a $\mathbf{L}$, viz obrázek 5.12. Jsou-li vektory normalizované $\|\mathbf{N}\|\|\mathbf{L}\|$ je 1. Když je skalární součin záporný, světlo přichází z opačné strany povrchu a odraz je 0. Dále odražené světlo závisí na vektoru intenzity příchozího světla $\mathbf{I}$ a barvě povrchu $\mathbf{C}$. Odražené světlo $L_{diffuse}$ se spočítá podle vzorce $L_{diffuse} = \mathbf{ICN} \cdot \mathbf{L}$.

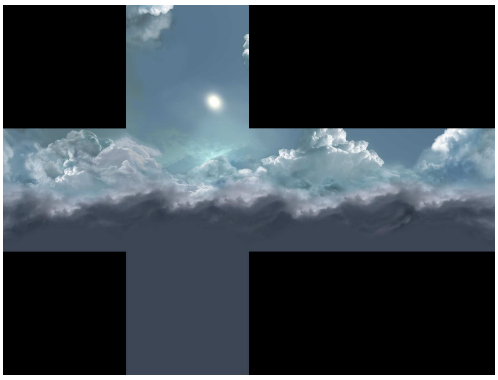


Obrázek 5.12: Odraz světla se vypočítá z vektoru světla a povrchové normály.

5.6 Obloha

Pro zobrazení oblohy je implementován skybox. Skybox je krychle, jejíž střed je umístěn v místě pozice kamery a pohybuje se spolu s ní. Krychle se nikdy nerotuje, pouze posunuje.

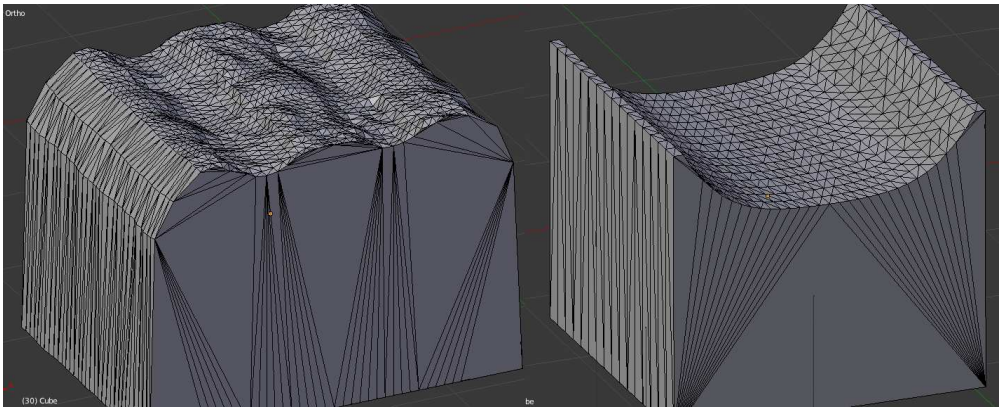
Na vnitřní stranu krychle je namapována speciální textura, viz obrázek 5.13, jejíž přechody mezi stranami krychle jsou plynulé. OpenGL umožňuje vytvoření cubemap textury k namapování na krychli. Díky tomu nemusí vrcholy krychle obsahovat souřadnice textury. Jako souřadnice textury se poté fragment shaderu předávají souřadnice bodů krychle. Skybox se vykresluje jako první s vypnutým depth bufferem. Tím je zajištěno, že obloha je zobrazena vždy až za ostatními objekty a je vytvořena iluze nekonečně vzdálené nehybné oblohy.



Obrázek 5.13: Textura pro vytvoření skyboxu

5.7 Model

Modely jsou pro aplikaci vytvořeny dva, viz obrázek 5.14. Jsou vytvořeny v programu Blender. Osa Z modelů vede ve směru dráhy a ve výsledné aplikaci po ní vede kamera, proto se nachází nad modelem. Osa Y vede nahoru. Modely musí mít osově symetrické přední a zadní okraje s osou X a tyto okraje musí mít na ose Z souřadnici 1.0 resp. -1.0. Aby mohl být model dobře namapován na křivku, je vhodné aby měl dostatek vrcholů podél osy Z.



Obrázek 5.14: Modely vytvořené v Blenderu

Kapitola 6

Výsledky a měření

Měření jsou prováděna na notebooku Asus K53SV.

- Operační systém: Windows 7
- Procesor: Intel Core i5-2410M, 2.30GHz
- Operační paměť: 4096MB, 1333MHz
- Grafická karta: NVIDIA GeForce GT 540M, 2048MB

Je prováděn normovaný průchod s různými nastaveními parametrů. Normovaný průchod pro vygenerovanou dráhu vytvoří rovnoměrně vzdálené kamery po celém okruhu. Pro každou kameru zobrazí jeden snímek a měří celkový a průměrný čas pro zobrazení snímků.

V tabulce 6.1 jsou nastavené parametry pro jednotlivá měření. Na každý segment se vygeneruje 10 kamer. V tabulce jsou počty trojúhelníků pro jednotlivé úrovně detailu, inkrementy k určování intervalů pro použití různých úrovní detailu a maximální počet trojúhelníků. Na obrázku 6.1 je vidět dráha s parametry 1. měření.

	segmentů	**lod1**	**lod2**	**lod3**	**lod4**	**inc1**	**inc2**	**inc3**	**max**
1.	225	2694	1600	800	300	0.25	0.75	1.75	25 000
2.	125	2694	1300	600	100	0.25	0.75	1.75	20 000
3.	597	2694	2000	1500	1000	0.3	0.6	0.9	100 000
3.	286	2694	1500	700	200	0.1	0.5	1.5	30 000

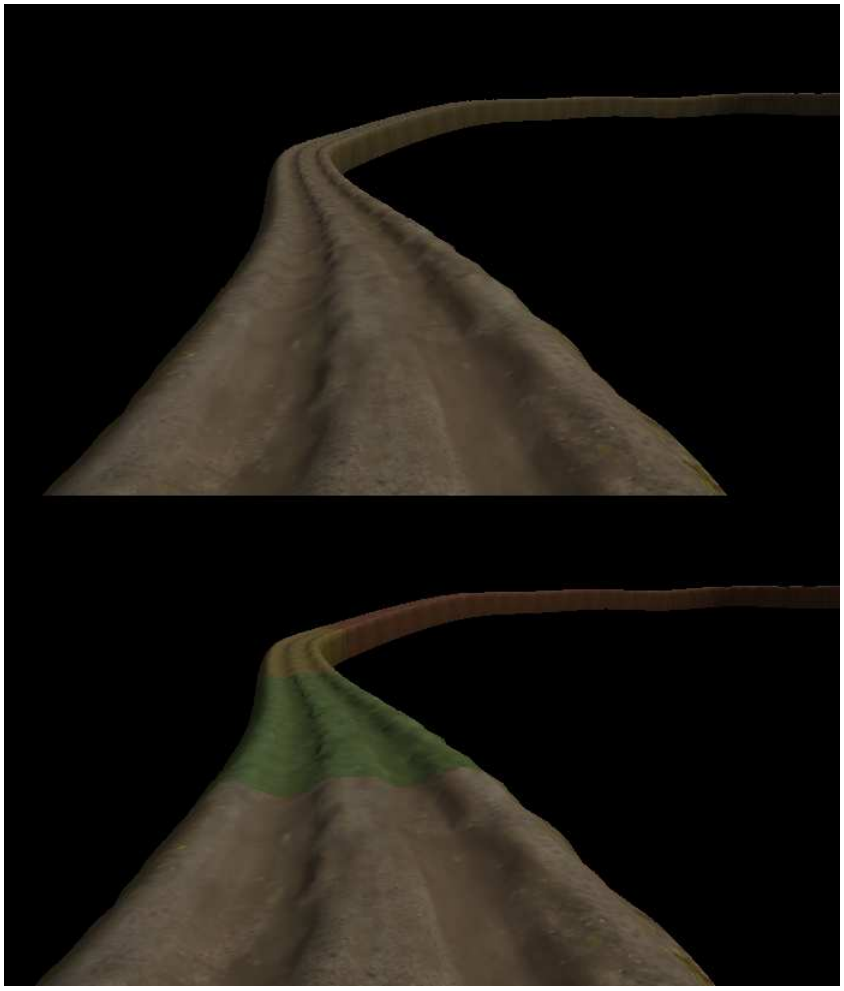
Tabulka 6.1: Parametry jednotlivých měření

Z naměřených hodnot v tabulce 6.2 je vidět, že view frustum culling zabírá jenom zanedbatelné množství z celkového času pro zobrazení. Při zapnutém LOD je doba vykreslování mnohonásobně zkrácena oproti vypnutému LOD. Čas strávený výběrem vhodného modelu je také zanedbatelný. Doba výběru LOD a view frustum culling se zvyšuje spolu s počtem segmentů dráhy. V 1. a 3. měřením je průměrný počet trojúhelníků na snímek vyšší než maximální hranice. To je dáno velkým množstvím zobrazovaných segmentů a počtem trojúhelníků nejnižšího LOD. Celkové zrychlení je vyšší než dvojnásobné pro všechna čtyři měření, pro 1. měření je více než čtyřnásobné. Množství zobrazených trojúhelníků je průměrně asi pětkrát nižší.

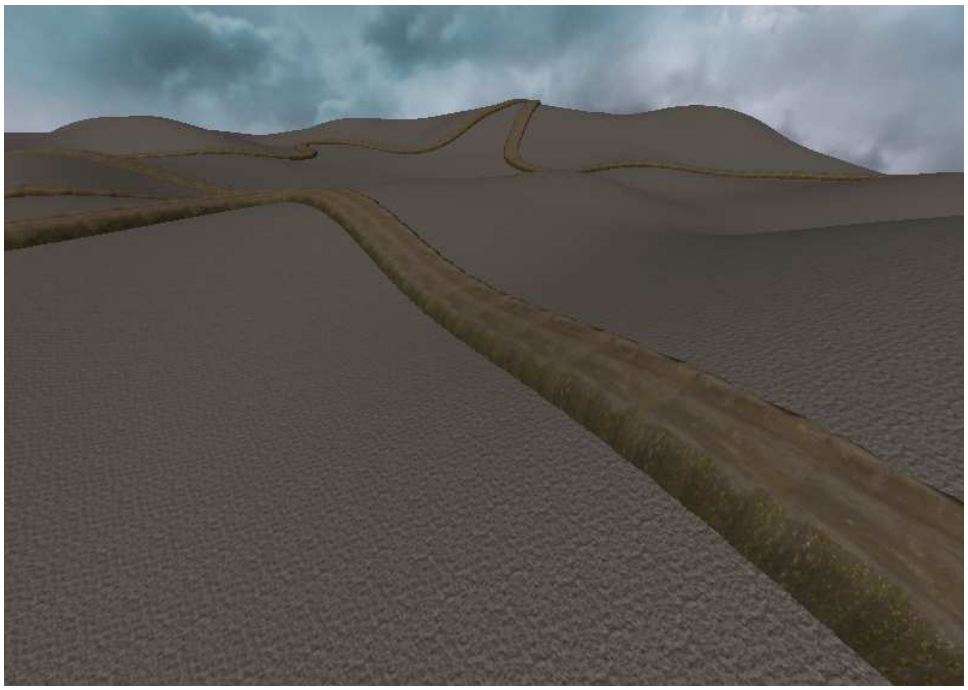
Snímek z výsledné aplikace se zapnutým terénem a skyboxem je vidět na obrázku 6.2.

		LOD zapnuto				LOD vypnuto		
	FC	**LOD**	**kreslení**	**celkem**	**troj.**	**kreslení**	**celkem**	**troj.**
1.	0.016	0.020	1.508	1.561	32.8	7.002	7.028	201.8
2.	0.010	0.013	1.562	1.603	20.0	4.705	4.733	126.4
3.	0.024	0.037	6.850	6.935	204.4	16.976	17.043	516.2
4.	0.020	0.022	1.593	1.652	31.8	5.593	5.688	247.9

Tabulka 6.2: Naměřené průměrné hodnoty na jeden snímek, všechny časy jsou v milisekundách, počty trojúhelníků jsou v tisících



Obrázek 6.1: Nahoře je LOD vypnut a je zobrazeno 105 066 trojúhelníků, dole je LOD zapnut s parametry 1. měření a je zobrazeno 25 228 trojúhelníků, jednotlivé úrovně detailu jsou odlišeny barevně.



Obrázek 6.2: Výsledná aplikace, zapnut terén a skybox

Kapitola 7

Závěr

V rámci bakalářské práce bylo potřeba nastudovat způsoby reprezentace 3D objektů a terénu. Byly popsány použité metody a vytvořen návrh aplikace. Aplikace byla implementována a popsána. Bylo provedeno měření a zhodnoceny výsledky.

Byl vytvořen systém, který je schopen z jednoho modelu úseku dráhy vytvořit celou dráhu. Vytvoření takovéto dráhy je jednoduché a paměťově nenáročné. Pro efektivní zobrazení byla použita úroveň detailu, která urychlila zobrazení až čtyřnásobně. Urychlení při zachování vizuální podobnosti je značně závislé na použitém modelu – na detailnosti jeho textury a topologii. Dále na použité dráze, např. kolik segmentů dráhy je průměrně vidět. Množství trojúhelníků modelů jednotlivých úrovní detailu a velikosti intervalů jejich použití je nutno těmto podmínkám přizpůsobit a najít vhodný kompromis mezi rychlostí a vizuální kvalitou.

Pro další vývoj práce by bylo vhodné implementovat podporu pro vytvoření dráhy z více modelů, occlusion culling k určení segmentů zakrytých terénem nebo jinými objekty a zvýšit vizuální přitažlivost např. přidáním vody nebo vegetace.

Literatura

[1] GLSL Programming/GLUT/Diffuse Reflection [online]. http://en.wikibooks.org/wiki/GLSL_Programming/GLUT/Diffuse_Reflection, [rev. 2012-10-18].

[2] OpenGL Terrain Tutorial [online]. http://www.lighthouse3d.com/opengl/terrain.

[3] View Frustum Culling [online]. http://www.lighthouse3d.com/tutorials/view-frustum-culling.

[4] CLARK, J.: Hierarchical Geometric Models for Visible Surface Algorithms. *Communications of the ACM*, ročník 19, č. 10, 1946: s. 547–554.

[5] GARLAND, M.; HECKBERT, P. S.: Surface Simplification Using Quadric Error Metrics. 1997.

[6] LAGAE, A.; DUTRÉ, P.: A Comparison of Methods for Generating Poisson Disk Distributions. *COMPUTER GRAPHICS forum*, ročník 27, č. 1, 2008: s. 114–129.

[7] LUEBKE, D.; REDDY, M.; COHEN, J. D.; aj.: *Level of Detail for 3D Graphics*. Morgan Kaufmann Publishers, 2003, ISBN 1-55860-838-9, 390 s.

[8] SCHREINER, D.; WOO, M.; NEIDER, J.; aj.: *OpenGL: průvodce programátora*. Computer Press, 2006, ISBN 80-251-1275-6.

[9] YUKSEL, C.; SCHAEFER, S.; KEYSER, J.: On the parameterization of Catmull-Rom curves. In *Proceedings of the 2009 SIAM/ACM joint conference on geometric and physical modeling*, ACM.