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

SVĚTELNÉ MAPY VE 3D POČÍTAČOVÉ GRAFICE

LIGHTS MAPS IN 3D COMPUTER GRAPHICS

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE JAROSLAV VRÁNA
AUTHOR

VEDOUCÍ PRÁCE JIŘÍ VENERA, Ing.
SUPERVISOR

BRNO 2008

Abstrakt

Bakalářská práce popisuje techniky využívané při tvorbě virtuálního terénu. První část práce je zaměřena na techniky používané pro generování výškové mapy, vytvoření textury terénu a následné renderování a texturovaní terénu. Dále se práce zabývá tvorbou a aplikací světelných map. Jsou popsány tři algoritmy pro vytvoření světelných map, které jsou aplikovány na terén a hladinu vody. Jako rozšíření je popsána technika pro urychlení vykreslování scény.

Klíčová slova

Světelné mapy, výškové mapy, OpenGL, multitexturing, generování textur, Level of Detail, View frustum culling

Abstract

Bachelor´s thesis describes techniques that are used in virtual terrain creation. First part describes generating of the height maps and the textures, after that is shown the terrain rendering and texturing. And next that is described the light maps generating. I show tree algorithms for the light maps generating, two for the terrain a one for the water. At the end I describe extension for the acceleration of the scene.

Keywords

Light Maps, height maps, OpenGL, multitexturing, texture generating, Level Of Detail, view frustum culling

Citace

Jaroslav Vrána: Světelné mapy ve 3D počítačové grafice, bakalářská práce,
Brno, FIT VUT v Brně, 2008

Světelné mapy ve 3D počítačové grafice

Prohlášení

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

……
Jaroslav Vrána
7. 5. 2008

© Jaroslav Vrána, 2008.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

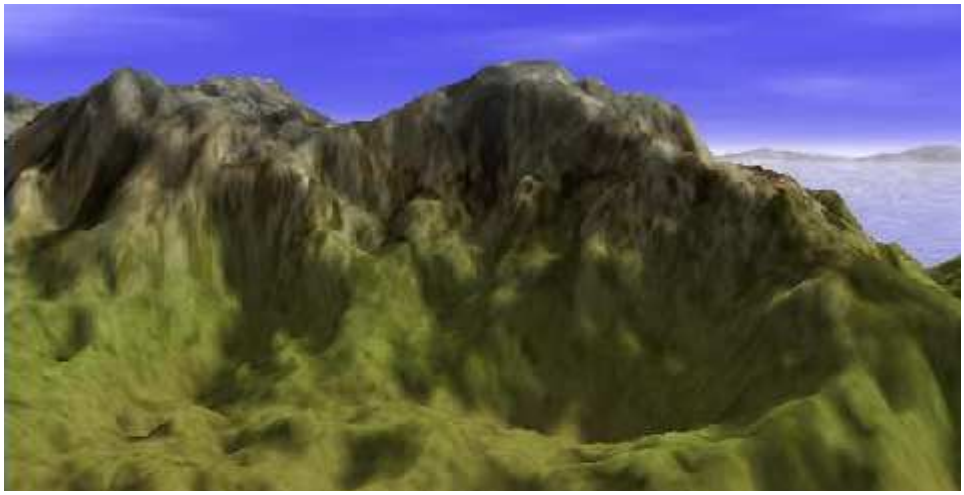
Obsah

Obsah....1
1 Úvod....2
2 Terén....3
2.1 Generování výškové mapy....3
2.2 Zobrazení terénu....4
3 Texturovaní terénu....5
3.1 Generování textury terénu....5
3.2 Zobrazení terénu s texturou....6
3.3 Detailní mapy....7
3.4 Multitexturing....7
3.5 Zobrazení terénu s detailní mapy....8
4 SkyBox....8
4.1 Voda....9
5 Světelné mapy....11
5.1 Co jsou to světelné mapy....11
5.2 Generování světelné mapy....12
5.3 Světelná mapa založená na výšce terénu....12
5.4 Světelná mapa založena na algoritmu Slope lighting....14
5.5 Světelná mapa založena na stochastickém fraktálu....16
6 Urychlení zobrazování scény....17
6.1 Level Of Detail – GeoMipMapping....17
6.2 View frustum culling....21
6.3 Přínos optimalizace vykreslování....23
7 Vlastní implementace....24
8 Možná rozšíření....25
8.1 Dokonalejší světelné mapy....25
8.2 Realistický vzhled vodní hladiny....25
8.3 Kolize s terénem....25
8.4 Přidání dalších objektů....25
9 Závěr....26
Příloha – ovládání programu....27
Literatura....28
Seznam příloh....29

1 Úvod

Počítačová 3D grafika je dnes využívána v mnoha odvětvích. Lze ji najít například v interaktivních aplikacích typu počítačových her, či v rozsáhlých trikových scénách v nejednom filmu. Při tvorbě počítačových scén, ať se jedná o počítačovou hru či filmový trik chceme, aby výsledná scéna vypadala realisticky a ne jako plastické objekty v prostoru. Důležitým faktorem je osvětlení. Existují různé techniky, které vypadají více, či méně realisticky a mají různé nároky na výpočetní výkon. Některé jsou počítány v reálném čase, některé předpočítány a za běhu aplikace již pouze aplikovány na objekt. Zajímavou metodou jsou světelné mapy. Před vlastním vykreslováním scény se vhodným algoritmem spočítá intenzita světla dopadající na jednotlivé pixely objektu a tato intenzita je zaznamenána právě do světelné mapy. Následně při vykreslování scény je nanesena spolu s texturou. Tato práce se věnuje technice aplikace světelných map na 3D teren.

V první části se dočteme o možných způsobech generování reliéfu krajiny a jeho následným vykreslením v prostoru pomocí pravidelné mřížky trojúhelníků. Další kapitola se věnuje tvorbě textury terénu a jejímu vylepšení pomocí detailní mapy. Terén nechceme mít zasazen do černého prostoru, proto se další část věnuje tvorbě Skyboxu a následně vodní hladině, která obklopuje terén a realisticky ho začleňuje do scény. Je vysvětleno generování a aplikace světelných map, které se používají jak pro terén, tak pro vodní hladinu. Poslední kapitola se věnuje mému rozšíření, je zde popsán použitý LOD algoritmus a ořezávání neviditelných objektů pomocí pohledového ořezávacího kuželu(view frustum culling).



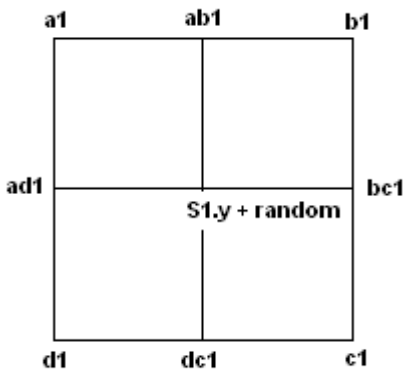
Obr. 1.1: Ukázka kompletní scény

2 Terén

Digitální terén může být v počítačové grafice reprezentován různými způsoby. Např. pro GIS systémy je vhodné použití vektorů ve formě vrstevnic. Často se také používají výškové mapy, které tato práce také popisuje. Dalším krokem je vykreslení terénu do podoby, na kterou je člověk zvyklí z reálného světa. Používají se např. metody vrhání paprsků, obdoby raycastingu. Ovšem nejpoužívanější metodou je zobrazení terénu pomocí polygonové sítě.

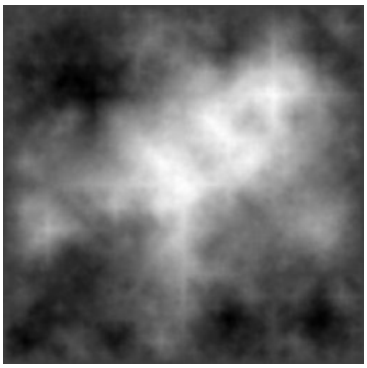
2.1 Generování výškové mapy

Výšková mapa udává ráz krajiny. Jedná se dvourozměrné pole hodnot, kde každá hodnota pole vyjadřuje výšku terénu v daném bodě. Vstupem tohoto algoritmu jsou výšky čtyř krajních bodů čtverce. V každém kroku dojde k výpočtu výšek středů hran a následně výšky středu čtverce, která je následně posunuta o náhodnou hodnotu v ose y. Tento výpočet probíhá dokud velikost hrany čtverce není rovna jedné.



Obr. 2.1: Princip rekurzivního dělení čtverce

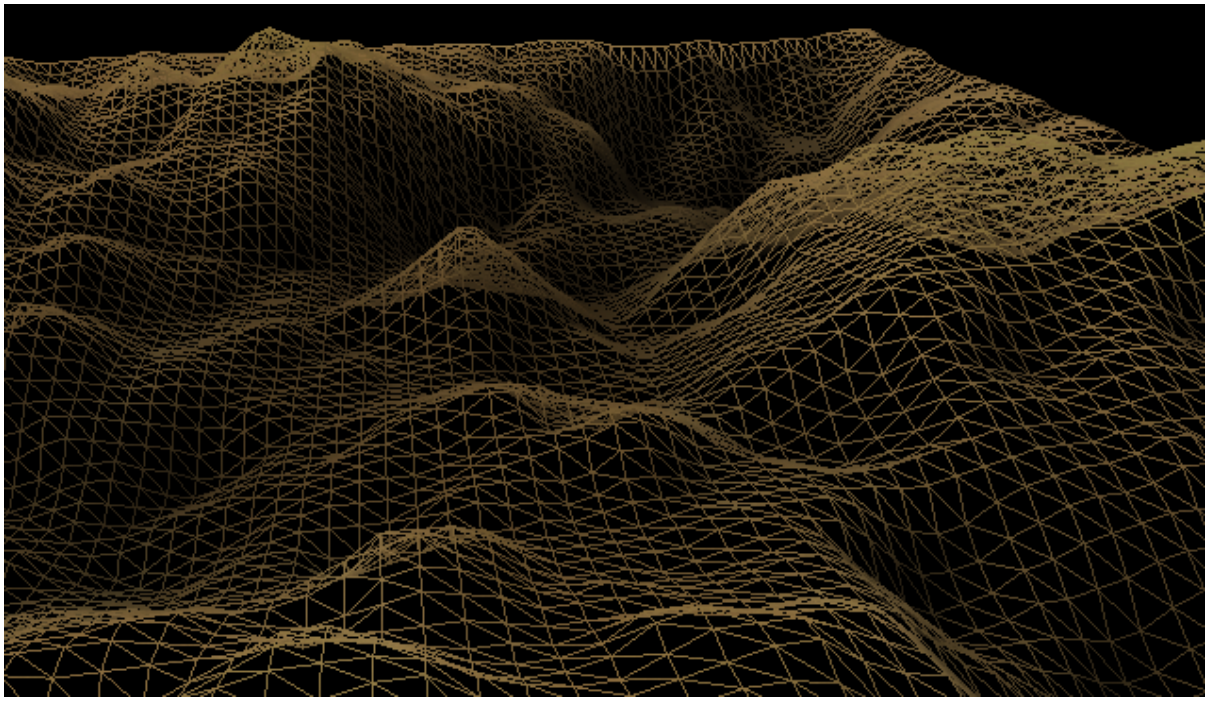
Výškovou mapu lze reprezentovat jako šedo-tónový obrázek. Následující obrázek tuto možnost znázorňuje.



Obr. 2.2: Výšková mapa reprezentovaná šedo-tonový obrázkem

2.2 Zobrazení terénu

Pro zobrazení terénu z výškové mapy se ve většině případů používá pravidelná trojúhelníková síť, která ušetří jak objem dat přenášený do grafické karty, tak umožňuje snadné zobrazení terénu. V nejjednodušším případě, který se v anglické terminologií nazývá Brutal force, vezmeme všechny hodnoty výškové mapy a vytvoříme pravidelnou trojúhelníkovou síť. Důležitým faktorem je, že jsou všechny vrcholy stejně daleko, vzdálenost se volí podle plánované rozlohy terénu. Existují však postupy, kdy se hodnoty výškové mapy vynechávají například podle vzdálenosti od pozorovatele. Tato technika se nazývá LOD a bude popsána dále.



Obr. 2.3: Část terénu vykreslená pomocí pravidelné trojúhelníkové sítě

3 Texturovaní terénu

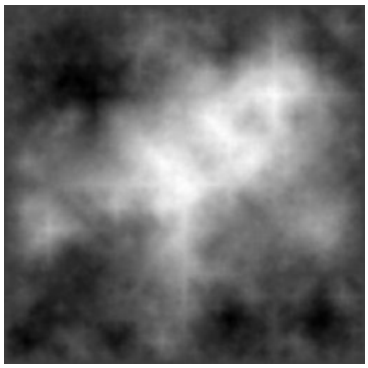
Texturování je důležitým krokem při tvorbě téměř každé počítačové scény. Textura přidává objektu větší stupeň detailu. S tímto souvisí pojem mapování textur, což je přiřazení správné části textury na odpovídající polygon objektu. Následující kapitola se zabývá generováním textury terénu a jejím vylepšením za pomocí detailní mapy.

3.1 Generování textury terénu

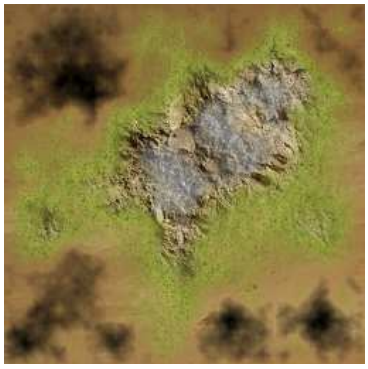
Myšlenka je jednoduchá. Je použita kolekce čtyř textur, kde každá textura znázorňuje jiné prostředí. První textura zasněžené vrcholky hor, následující skály atd. Každé textuře přiřadíme interval, na kterém má být nanášena. Dále podle výšky bodu ve výškové mapě se rozhodneme jakou texturu pro daný bod použijeme. Ovšem tento jednoduchý přístup neřeší plynulé přechody mezi jednotlivými prostředími, proto se zavádí rozšíření. Pro každé prostředí se počítá procentuální náležitost pro zkoumanou výšku a pomocí těchto percentilů se počítá výsledná barva texelu:

$$
\begin{aligned}
texturaTerenu[x, y] = \; & textura1[x, y] * percentil1 + \\
& textura2[x, y] * percentil2 + \\
& textura3[x, y] * percentil3 + \\
& textura4[x, y] * percentil4
\end{aligned}
$$

kde percentilN je procentuální náležitost daného prostředí pro zkoumanou výšku.



Obr. 3.1: Výšková mapa

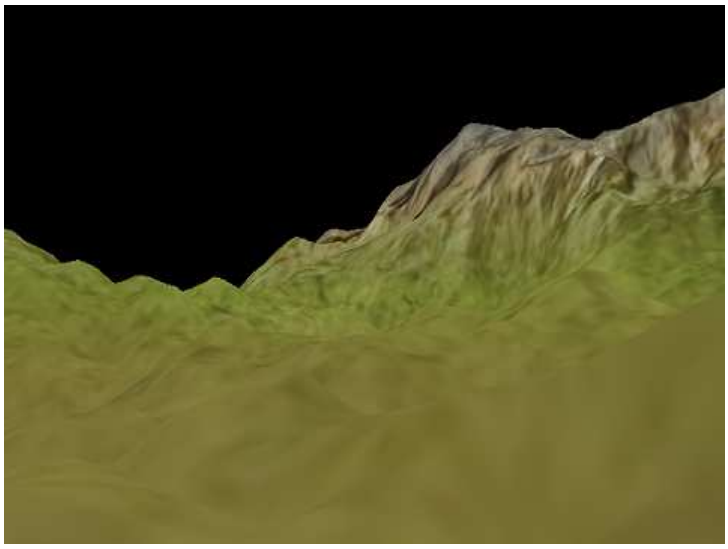


Obr. 3.2: Odpovídající textura

Jak je patrné z obrázků nejnižší místa výškové mapy jsou na výsledné textuře terénu tmavá. Tento negativní jev je způsoben zvolenou metodou pro počítání percentilů. Zajímavá je funkce publikovaná v článku Terrain texture generation[1]. Jedná se o jednoduchou a efektivní funkci, kde pro výšku rovnou nule se do výsledné textury nezahrne žádná z podkladových textur. V mém případě jsem celý ostrov zasadil do Skyboxu s vodou, takže se daný negativní jev neprojevuje. Další možností je dodatečná korekce textury, například interpolací tmavých bodů s okolními.

3.2 Zobrazení terénu s texturou

Jedná se o namapování textury na již vytvořenou pravidelnou trojúhelníkovou síť. Jediným úskalím, ovšem ne příliš komplikovaným, je výpočet koordinál pro otexturování jednotlivých trojúhelníků tvořící terén.



Obr. 3.3: Pravidelná trojúhelníková síť s nanesenou texturou

3.3 Detailní mapy

Nyní je povrch terénu hladký a vypadá až plasticky. Detail mapa je šedo-tónová textura nanášená opakovaně na terén. Tímto dojde k efektu zvrásnění a terén již nevypadá tak plasticky.

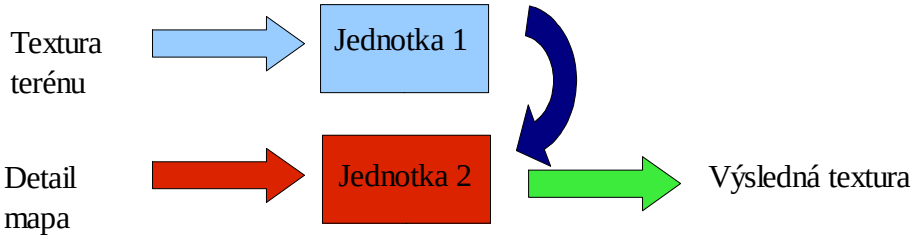
Obr. 3.4: Ukázka detail mapy

Ovšem nastává problém. Nyní chceme relativně rychlým způsobem nanést na terén více než jednu texturu. Toto umožňuje technika nazývaná multitexturing.

3.4 Multitexturing

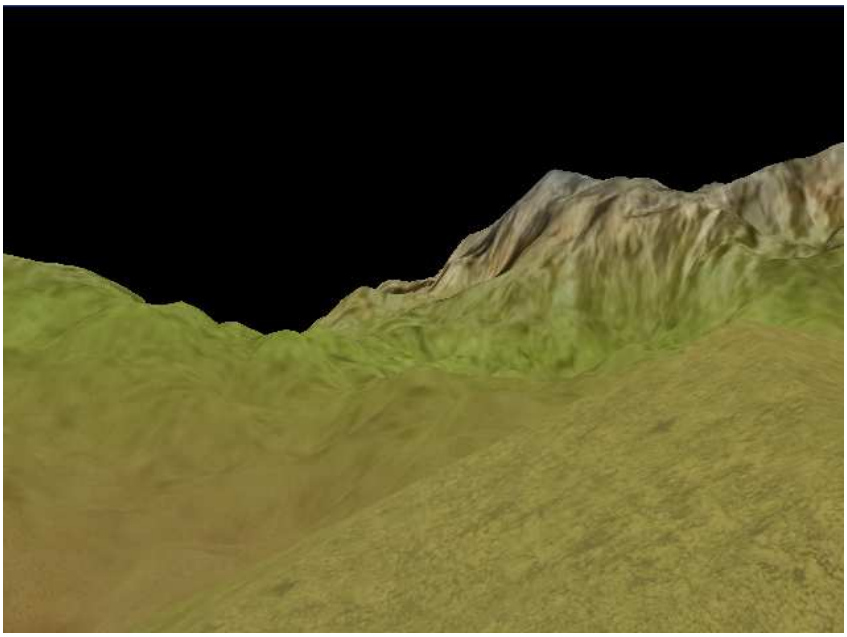
Jedná se o rozšíření grafických karet, musíme tedy před použitím multitexturingu otestovat grafický akcelerátor, zda je daná funkce podporována. Současná generace grafických karet má k dispozici až 32 texturovacích jednotek. Problém může nastat při použití na starších kartách.

Multitexturing umožňuje zpracovat více textur najednou. Umožňuje to přítomnost více texturovacích jednotek na grafické kartě. Každé texturovací jednotce přiřadíme její vlastní texturu a o vykreslení se již stará samotná grafická karta. Alternativou by mohlo být využití blendingu, které by ovšem bylo řádově pomalejší.



3.5 Zobrazení terénu s detailní mapy

Jak již bylo uvedeno výše, tak před použitím detail map je nejdříve potřeba otestovat přítomnost podpory multitexturingu a po té nastavit dostatečný počet opakovaní na povrchu terénů, aby došlo k požadovanému efektu.

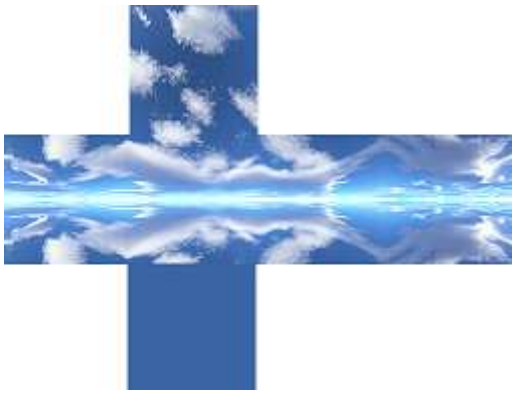


Obr. 3.6: Terén zobrazený s texturou terénu a detail mapu

Z obrázků 3.6 je patrné, že terén s přidanou detail mapou již není tak plastický, jako byl bez detail mapy na obrázku 3.3. Na textuře terénu jsou patrné linie detail mapy, které povrchu přidávají detail a přispívají k celkové realističnosti scény.

4 SkyBox

Nyní je náš terén situován do černého pozadí. Tato situace nepřidává příliš realistickému vzezření scény. Ve 3D počítačové grafice se používá technika, která scénu uzavře do krychle. Na stěny této krychle jsou naneseny textury okolní krajiny, oblohy, moře, atd. SkyBox musí být do scény vykreslen jako první. Dále musí být vypnuté testování hloubky, to nám umožní vykreslit SkyBox jako krychli o malé délce hrany a i přesto se tváří jako největší objekt ve scéně. Dále by mělo být vypnuto osvětlení, které by mohlo odhalit skutečný tvar SkyBoxu.



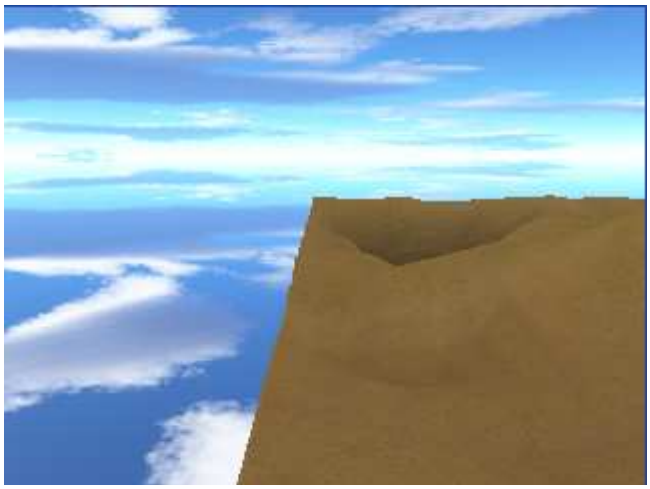
Obr. 4.1: Textura SkyBoxu

Obrázek 4.1 znázorňuje, jak je komponována textura SkyBoxu. Není vždy jednoduché požadovanou texturu upravit tak, aby pěkně navazovala a tvořila realistický dojem ve scéně.

Obr. 4.2: Scéna vykreslená ve SkyBoxu

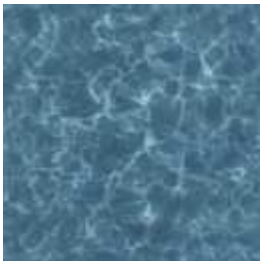
4.1 Voda

Nyní, když máme terén pěkně vykreslený ve SkyBoxu je potřeba vyřešit ještě jeden malý problém. Stále jsou jasně patrné okraje terénu jako objektu situovaného do „nekonečného“ SkyBoxu. Situaci znázorňuje obrázek 4.3.



Obr. 4.3: Patrné okraje terénu

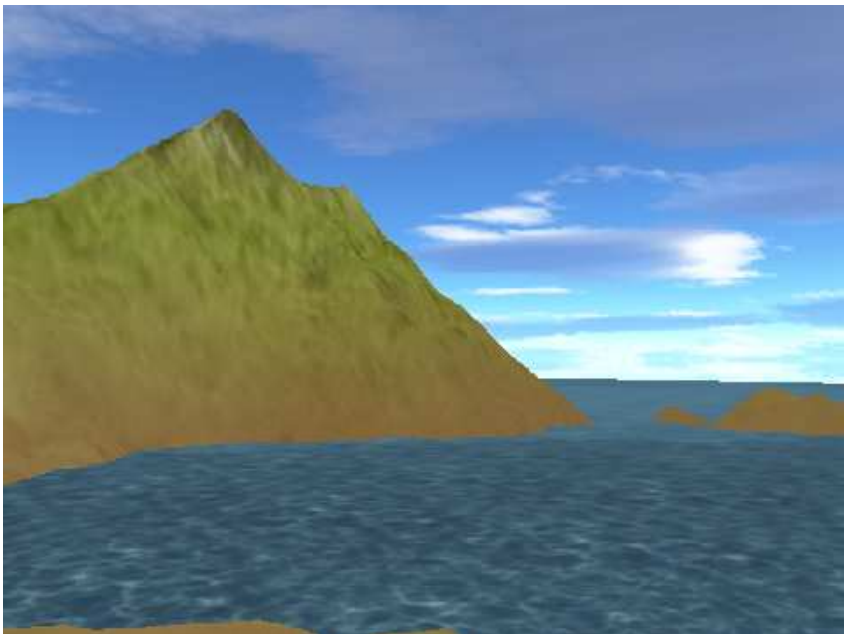
Zajímavým východiskem z této situace je použití vodní hladiny. Část terénu se „potopí“ pod vodu, která bude přesahovat okraje terénu. Tímto docílíme lepšího vizuálního dojmu celého pojetí scény. Vodní hladina není nic jiného než čtverec vykreslený s nanesenou texturou v určité výšce.



Obr. 4.4: Textura vodní hladiny

Pěkného dojmu lze dosáhnout s texturou vody v nízkém rozlišení a jejího několikanásobného opakovaní, podobně jako u detail mapy.

Dalším zajímavým rozšířením může být pohyb vody. Pouhý pohyb vody po ose Y v rozsahu několika pixelů v několika stupních nahoru a následně opět v několika stupních dolu s analogickým opakováním vede k pěknému efektu narážení vln na břeh terénu.



Obr. 4.5: Scéna s vykreslenou vodou

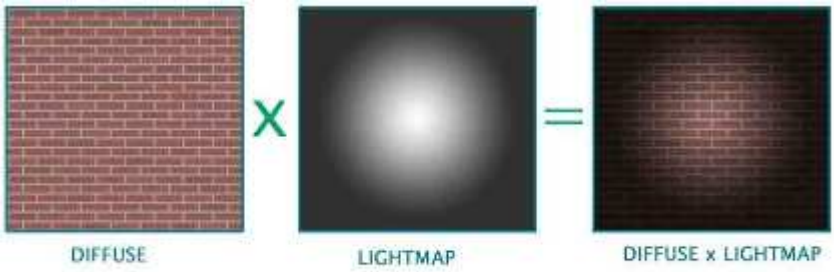
5 Světelné mapy

Následující kapitola popisuje generování a aplikaci světelných map. Přidání světelných map do scény vede k velkému posunu vizuálního zpracování scény vpřed.

5.1 Co jsou to světelné mapy

Osvětlení lze počítat dynamicky za chodu aplikace nebo předpočítat a za chodu aplikace již pouze aplikovat výsledek na daný objekt. Pro dynamický výpočet osvětlení dnes již existují různé hardwarové implementace. Výsledek vypadá velmi realisticky. Ovšem výpočetní výkon potřebný pro implementaci v interaktivní aplikaci je obrovský a především starší hardware proto není vhodný. Z tohoto důvodu se používají světelné mapy(v anglické terminologii Light maps), které jsou předpočítány při startu aplikace a za běhu již pouze aplikovány.

Ve většině případů se jedná o šedo-tónovou texturu, kde odstíny šedi odpovídají intenzitě dopadajícího světla. Pro generování světelných map lze použít různé algoritmy, kde se může jednat o zcela náhodné generování fraktálů, či systematické generování dle přesného umístění světelného zdroje.



Obr. 5.1: Ukázka aplikace světelné mapy

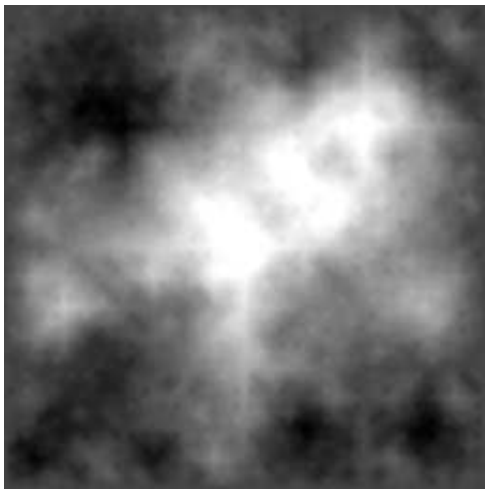
Pro představu funkce světelných map slouží obrázek 5.1, který je převzat z článku Light Mapping - Theory and Implementation [2].

5.2 Generování světelné mapy

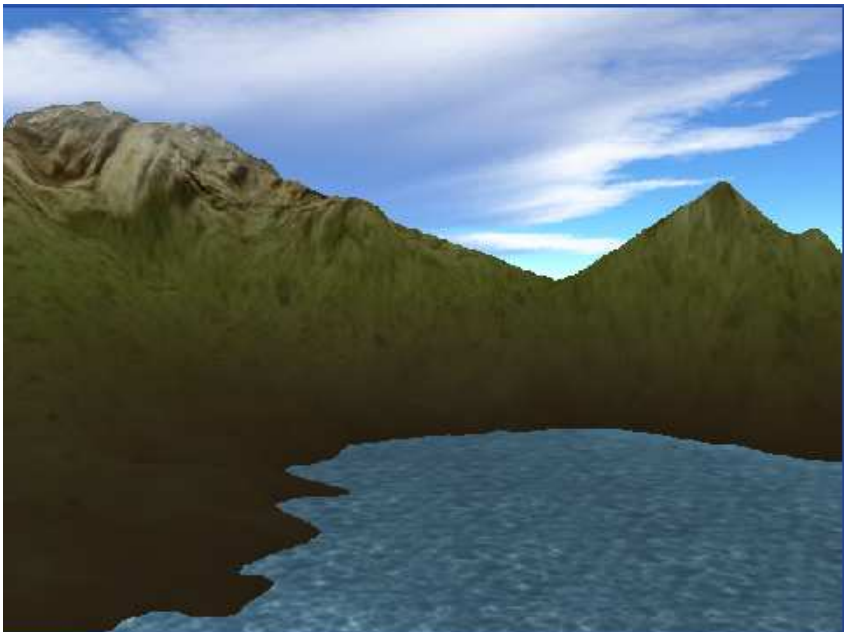
Generování světelných map je předvedeno na třech algoritmech. Následující kapitoly popisují jednotlivé techniky.

5.3 Světelná mapa založená na výšce terénu

Jedná se o jednu z nejjednodušších metod. V anglické terminologii označována jako heigh-based lighting. Světelná mapa je nejsvětlejší v bodech, které jsou na výškové mapě umístěny nejvýše a naopak body umístěné ve výškové mapě v nižších polohách jsou ve světelné mapě tmavé. V nejjednodušším případě se tedy může výšková mapa rovnat světelné. Ovšem můžou být implementovány rozšíření, kdy například definujeme barvu světelného zdroje.



Obr. 5.2 Světelná mapa vygenerovaná pomocí Height-based algoritmu

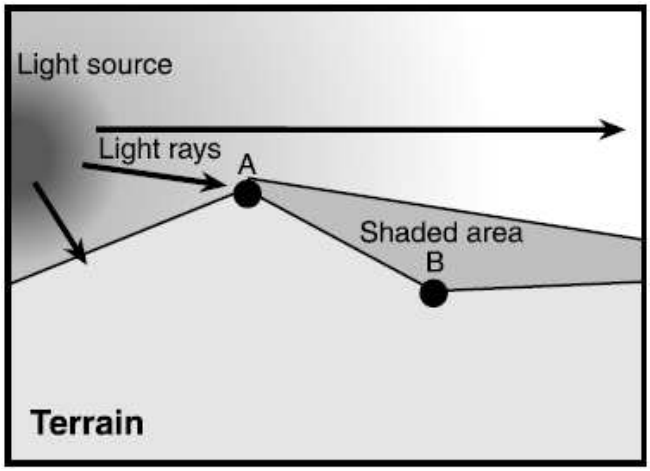


Obr. 5.3 Scéna se světelnou mapou vygenerovanou pomocí Heigh-based

Jak je vidět z obrázku 5.3 tato technika se pro daný účel příliš nehodí. Nejspíše by se našli specifické situace, kdy by se dala daná technika použít, ale v tomto případě nevypadá výsledek nijak realisticky. Jako zajímavé by se mohlo ukázat přiřadit světelnému zdroji barvu. Toto ovšem není v této práci implementováno.

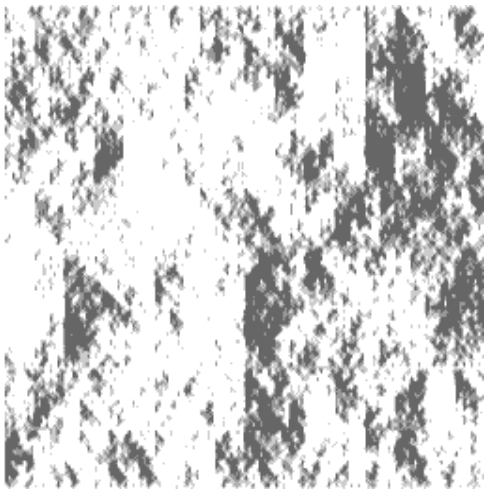
5.4 Světelná mapa založena na algoritmu Slope lighting

Tato technika je velice často využívána v interaktivních aplikacích. Generování světelné mapy je velice rychlé a implementace není příliš komplikovaná.



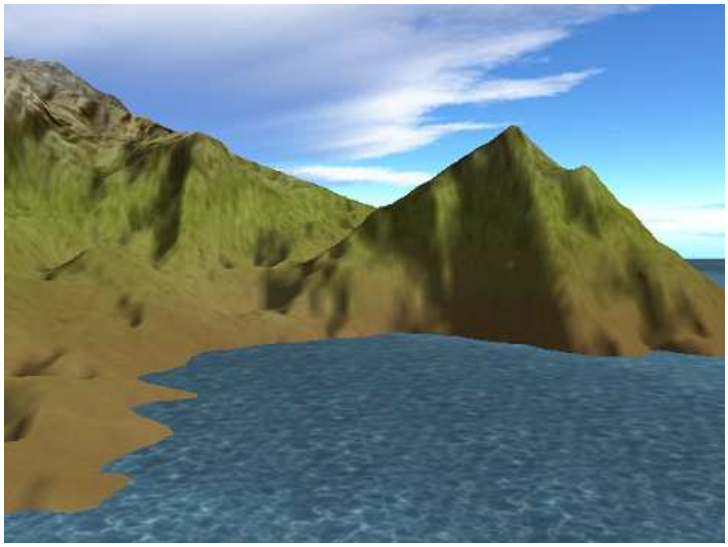
Obr. 5.4 Princip algoritmu Slope lighting

Obrázek 5.4 znázorňuje požadované chování při výpočtu světelné mapy pomocí algoritmu Slope lighting. Nejdříve se stanoví pozice světelného zdroje, následuje procházení výškovou mapou a dle směru šíření světla se zkoumají vztahy sousedních bodů. Pokud se světlo šíří zleva jako na obrázku 5.4 a zkoumáme osvětlení bodu B, podíváme se ve výškové mapě na předchozí bod ve směru šíření světla, v tomto případě na bod A, vypočítáme který bod se na mapě nachází výš a o jakou hodnotu výš a podle tohoto poznatku nastavíme hodnotu osvětlení zkoumaného bodu.



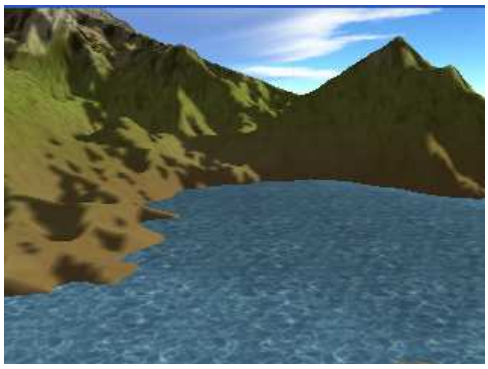
0br. 5.5: Světelná mapa vygenerovaná pomocí Slope lighting

Tento algoritmus sebou nese jednu nevýhodu. Polohu zdroje světla lze nastavovat pouze skokově a to po 45 stupních. Toto omezení je limitující v případě, že chceme dosáhnout přesných a velice kvalitních výsledků. Pro účely zobrazení interaktivního dema zcela postačuje a přináší kvalitní vizuální výsledky.

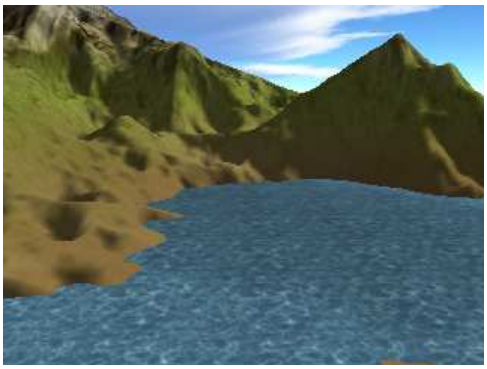


Obr. 5.6 Scéna se světelnou mapou vygenerovanou pomocí Slope lighting

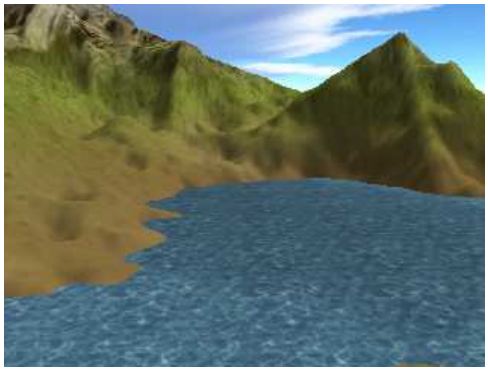
Snadno lze měnit jemnost stín. Nnásledující obrázky zobrazují porovnání různých nastavení jemnosti.



Obr. 5.7: Agresivní nastavení



Obr. 5.8: Střední nastavení



Obr. 5.9: Jemné nastavení

5.5 Světelná mapa založena na stochastickém fraktálu

Tento přístup nezohledňuje konkrétní umístění světla. Uplatní se v případě, kdy chceme nějakým náhodným způsobem vypočítat osvětlení.

Zde je tento přístup využit pro simulaci stínu mraků na vodní hladině. Jedná se o jednoduchý způsob jak oživit monotonní hladinu vody. Pro generování náhodného(stochastického) fraktálu se dá použít algoritmus, který byl již využit dříve pro generování výškové mapy terénu. Jedná se o rekurzivní dělení čtverce.

Obr. 5.10: Světelná mapa vygenerovaná metodou rekurzivní dělení čtverce

Obrázek 5.10 znázorňuje světelnou mapu reprezentující stíny mraků na vodní hladině. Metoda rekurzivní dělení čtverce vyprodukovala pěkný výsledek reprezentující mraky. Následující obrázek 5.11 znázorňuje aplikaci světelné mapy 5.10 na vodní hladinu.

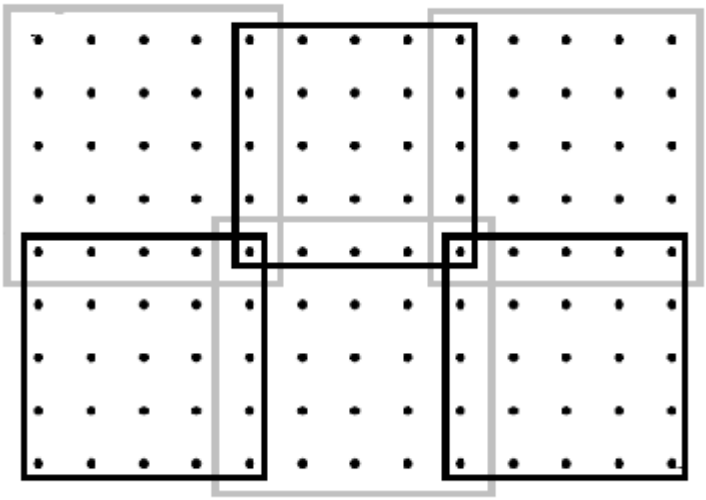
Obr. 5.11 Stíny mraků na vodní hladině

6 Urychlení zobrazování scény

Nyní je kompletně hotový celý virtuální svět, ale celá scéna se nevykresluje příliš rychle. Z tohoto důvodu je implementováno opatření pro urychlení vykreslování celé scény. Tato kapitola popisuje nejdříve Level Of Detail algoritmus a následně algoritmus pro ořezávání objektů(tj. jejich neposílání grafické kartě), které nejsou na vykreslené scéně přímo vidět.

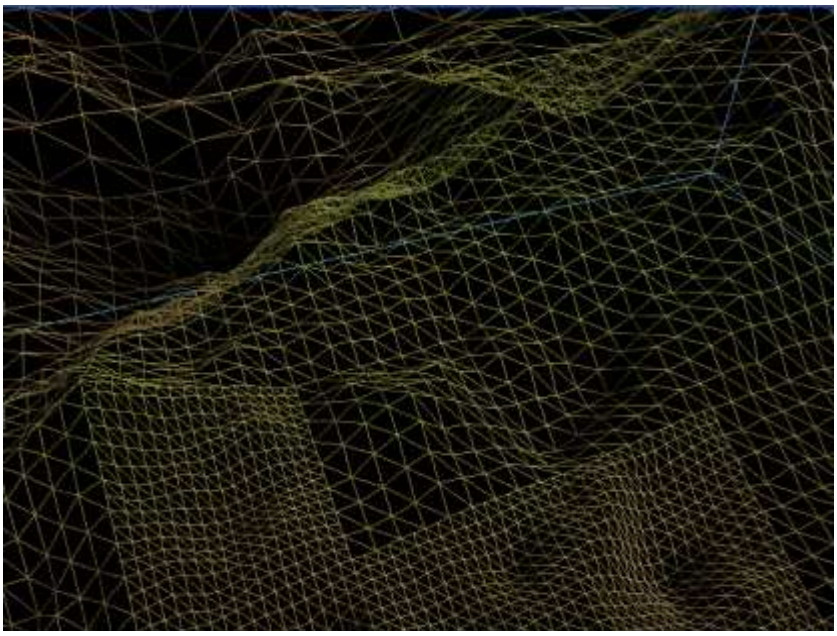
6.1 Level Of Detail – GeoMipMapping

Principem této techniky je rozdělení terénu do vzorů o velikosti 2^N+1 na 2^N+1 bodů, jednotlivé vzory se navzájem překrývají krajními body.



Obr. 6.1: Kompozice jednotlivých vzorů terénu

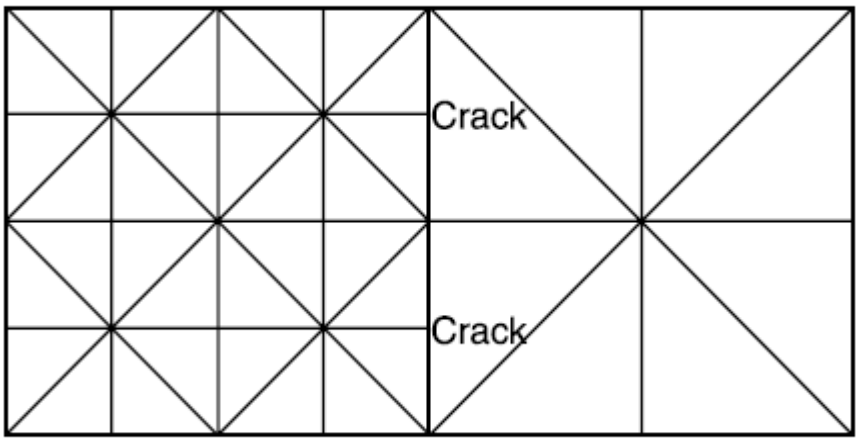
Poté co je terén takto rozdělen do vzorů se pro každý vzor zjistí jeho vzdálenost od aktuální polohy kamery, v potaz se bere vzdálenost kamera - střed vzoru. Pomocí vypočítaných vzdáleností se již nastavuje úroveň detailů pro jednotlivé vzory. Například můžeme říct, že vzdálenost menší než 1000 jednotek má úroveň detailů jedna, vzdálenost větší nebo rovna 1000 a menší než 2000 jednotek má úroveň detailů dva atd. Následně podle nastavené úrovně detailů se určuje jak se fyzicky bude vykreslovat každý vzor. Typickým postupem je, že při úrovni jedna se vykresluje každý bod vzoru, při úrovni dva každý druhý, následně každý čtvrtý atd. Vždy se jedná o mocninou dvou.



Obr. 6.2: Část scény zobrazená pomocí GeoMipMappingu

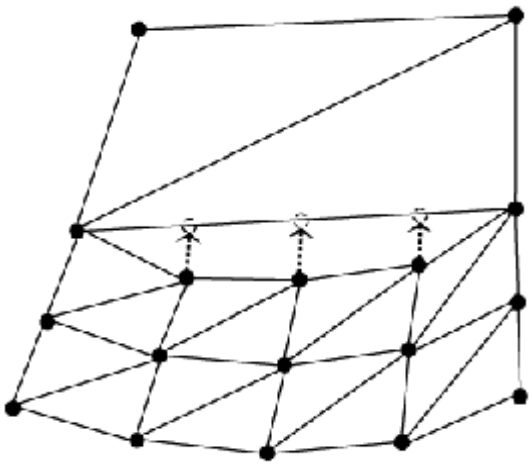
Na obrázku 6.2 je krásně vidět jak se detaily směrem od kamery snižují. Výše zmíněné výpočty se samozřejmě dějí při každém pohybu kamery. Úroveň detailů jednotlivých vzorů se dynamicky mění, což může být pozorovatelem postřehnutelné, mění se ráz krajiny. Ovšem při rozumném nastavení se nejedná o dramatické změny a výměnou za velkou úsporu výpočetního výkonu je tento jev snesitelný.

Při implementaci geomipmappingu se projevuje problém. Navazování vzorů s různou úrovní detailů vznikají trhliny v zobrazeném terénu.



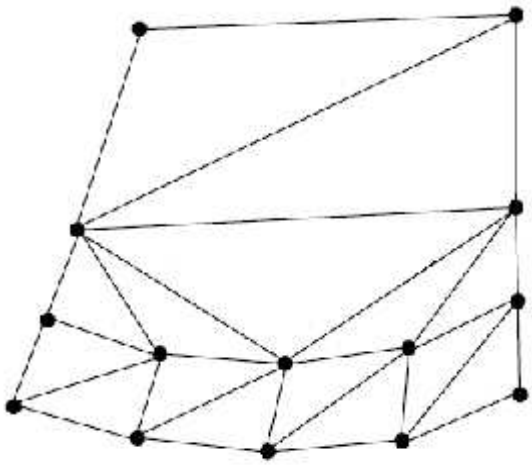
Obr. 6.3: Návaznost plátů s různým detailem

Obrázek 6.3 naznačuje proč se v terénu vyskytovaly trhliny. Vzor s vyšším detailem má na okraji více vrcholů než plát s nižším detailem a proto věrněji kopíruje reliéf terénu než druhý vzor. Tímto vznikají trhliny v terénu. Tento problém lze vyřešit dvěma způsoby. První z nich je pouhá interpolace bodů. V této práci je tento způsob dostačující, ovšem při použití hardwarového osvětlení by byly patrné ostré hrany na místech, kde došlo k interpolaci bodů.



Obr. 6.4: Princip interpolace

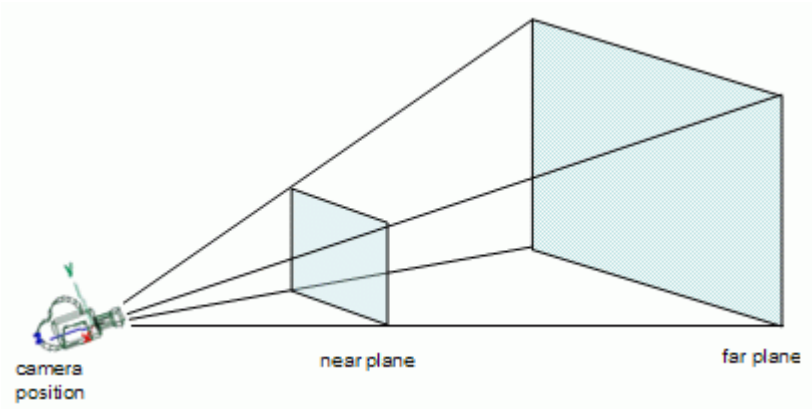
Druhý způsob je komplikovanější na implementaci, ovšem výsledek je velmi pěkný. Princip popisuje následující obrázek 6.5.



Obr. 6.5: Princip přesunutí vrcholů

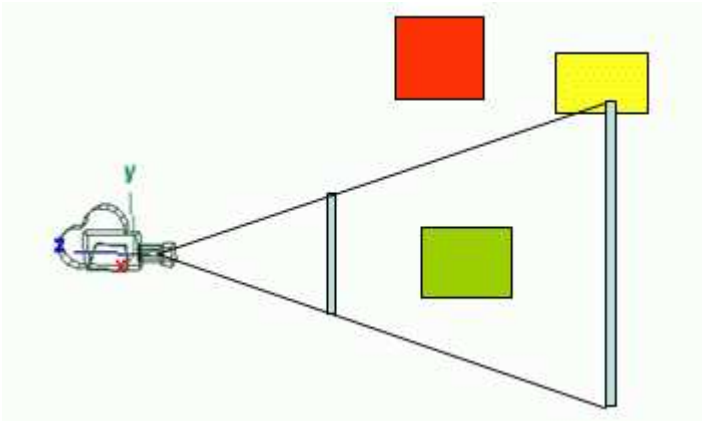
6.2 View frustum culling

Nyní, když je terén rozdělen do vzorů nabízí se implementovat další postup pro urychlení vykreslování scény. Tímto postupem je View frustum culling, jedná se o vytvoření komolého jehlanu, který je definovaný přední a zadní ořezávací plochou.



Obr. 6.6: Komolý ořezávací kužel

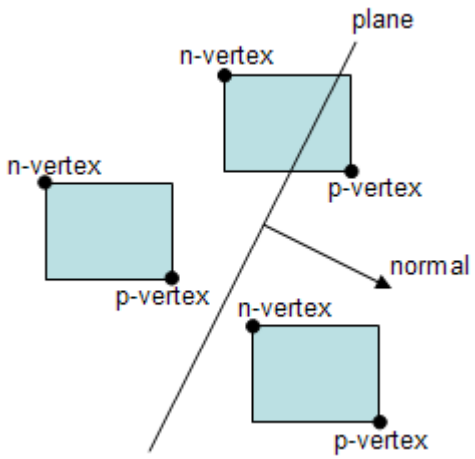
Po definování tělesa, které nám určuje objem pohledu, můžeme začít testovat, zda se hledaný objekt nachází celý vně, či venku tělesa nebo se částečně nachází v tělese. Princip zobrazuje následující obrázek 6.7.



Obr. 6.7: Testování náležitosti těles v pohledovém objemu

Následuje se zamyslet, jak složitou členitost každého ze vzoru terénu snadno otestovat, zda patří do pohledového objemu. Řešení může poskytnout anglickou terminologií pojmenovaný Axis Aligned Bounded Boxes(AABB). Jedná se o myšlenou krychli, která ohraničuje námi zkoumaný vzor tak, že se celý nachází uvnitř krychle. Veškeré následující testy probíhají nad touto krychlí.

Po vytvoření AAB krychle nad požadovaným vzorem chceme zjistit zda daný vzor patří do scény nebo ne, tj. jestli leží celý nebo alespoň částečně v pohledovém objemu. K tomuto testování se ještě musí určit dva body, tzv. positivní a negativní. Positivní je ten bod AAB krychle, který leží dále ve směru normály ořezávací roviny a negativní naopak, princip naznačuje obrázek 6.8.

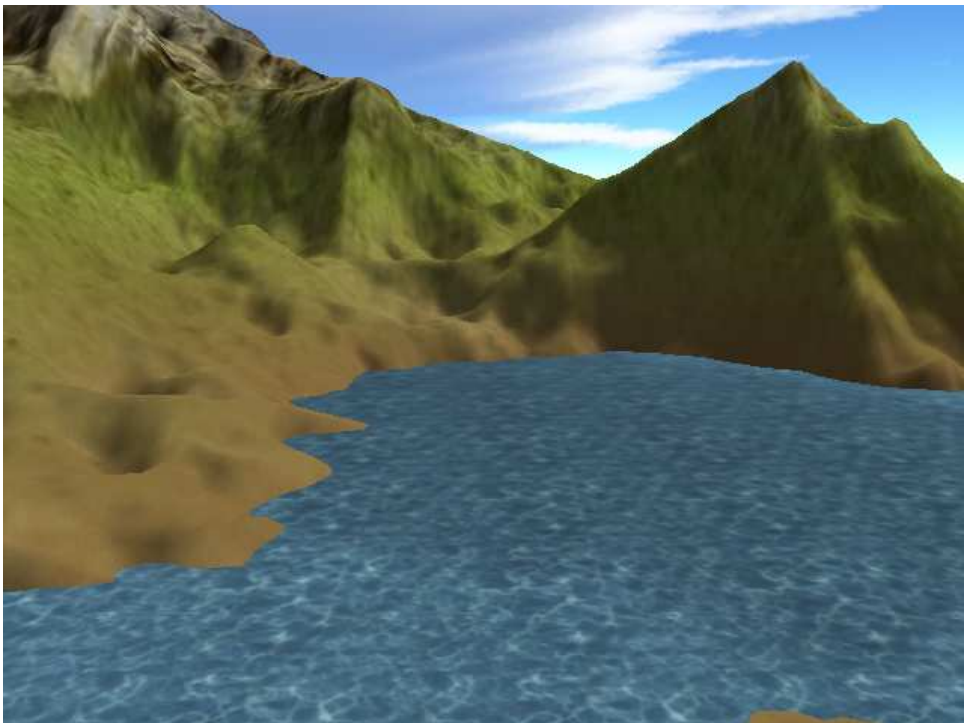


Obr. 6.8: Určení positivního a negativního bodu

Po získání pozitivního a negativního bodu se již zjišťuje vlastní poloha AAB krychle vzhledem k ořezávací rovině. Počítá se vzdálenost bodu od roviny, pokud je vzdálenost menší než nula, bod leží na opačné straně roviny než kam směřuje normála, tudíž do pohledového objemu nepatří. Nejdříve se testuje positivní bod, pokud do pohledového objemu nepatří, celá AAB krychle tam nepatří. Pokud pohledovému objemu náleží, musíme daný vzor vykreslit, protože minimálně částečně do pohledového objemu naleží.

6.3 Přínos optimalizace vykreslování

Tato kapitola shrnuje přínos optimalizačních technik, které byly použity pro urychlení vykreslování scény. Testování proběhlo s referenčním umístěním kamery, které je znázorněno na obrázku 6.9. Poprvé vykreslení proběhlo pomocí Brutal force algoritmu a poté pomocí zapnutého geomipmapingu a zároveň zapnutého View frustum cullingu. Celou situaci shrnuje tabulka 6.1.



Obr. 6.9: Referenční umístění kamery

Metoda vykreslení	Brutal force	GeoMipMap + View frustum culling
Počet trojúhelníků	130 050	9 584
Snímky za sekundu	13	28

Tab.: 6.1

Přínos optimalizačních metod je z tabulky 6.1 zřejmý. Testování proběhlo na notebooku s procesorem Intel Celeron 1.5GHz, 512MB RAM a grafickou kartou Intel Extreme Graphics 2.

7 Vlastní implementace

Celý projekt byl implementován v programovacím jazyce C++. Využil jsem objektově orientovaných možností jazyka C++ a celý návrh aplikace tomu podřídil. Snažil jsem se důkladně dekomponovat každý dílčí problém do vlastní třídy a tu následně do vlastního souboru, tímto jsem se dostal ke konečnému číslu 36 zdrojových souborů v projektu.

Použil jsem grafické API OpenGL. Toto prostředí jsem si vybral především z důvodu přenositelnosti aplikací mezi různými platformami. Dále se mi líbila jeho relativní jednoduchost a koncepční uspořádání celé OpenGL knihovny. Také jsem použil nadstavbovou knihovnu GLUT, která je taktéž přenositelná mezi platformami. MUSÍ být přítomna v systému, aby byl umožněn chod této aplikace.

Do vlastní aplikace jsem implementoval všechny výše popsané techniky. Snažil jsem se je implementovat tak, jak jsem výše popsal jejich princip.

8 Možná rozšíření

V této kapitole uvádím letmý přehled možných rozšíření, která bych rád přidal do tohoto projektu.

8.1 Dokonalejší světelné mapy

Rozhodně bych chtěl implementovat dokonalejší generovaní světelných map, kde bych nebyl omezen umisťováním světelných zdrojů v prostoru. Jako zajímavá možnost se mi jeví některá z implementací algoritmu Per-pixel lighting.

8.2 Realistický vzhled vodní hladiny

Jedna z věcí, které bych chtěl vyzkoušet především je zlepšení vzhledu vodní hladiny. Tohoto bych velice rád dosáhl za využití dnes hojně využívaných Pixel a Vertex shaderů. Techniky efektů jako vlnící se hladina, či odraz okolí na vodní hladině mě velice lákají prostudovat a následně implementovat.

8.3 Kolize s terénem

Dalším zajímavým rozšířením může být detekce kolize kamery s terénem. Nyní je možné se po scéně „prohánět“ zcela volně, takže je snadné se dostat pod povrch vykresleného terénu. Zcela určitě by toto mělo být v další fázi vývoje odstraněno.

8.4 Přidání dalších objektů

Velkou měrou by se na lepším dojmu z celého dema určitě podepsalo včlenění dalších objektů na povrch ostrova. Různé druhy stromů, keřů, či travin. Nejprve bych chtěl implementovat jednoduchý přístup pomocí billboardingu a následně implementovat načítání externích objektů typu 3DS.

9 Závěr

Cílem práce bylo nastudovat a implementovat algoritmy pro generovaní a následné vykreslení krajiny a generování světelných map. Tento úkol jsem dle mého názoru splnil. Kvalita zobrazené scény není na vysoké úrovni, ale myslím si, že přiměřeným způsobem předvádí postupy používané při tvorbě digitálního modelu terénu. Dále jsem přinesl rozšíření v podobě optimalizace vykreslování scény, která nemalou měrou urychlila zobrazování scény.

Tuto práci jsem si vybral, protože mě 3D počítačová grafika vždy přitahovala. Z tohoto důvodu jsem se návrh implementace snažil vytvářet obecně. Snažil sem navrhnout kvalitní objektově orientovaný návrh, abych mohl bez větších problému na tomto projektu dále pracovat. Má nejočekávanější rozšíření jsem uvedl v předchozí kapitole.

Na počátku jsem čekal, že práce bude mít větší rozsah a kvalitu, což se samozřejmě nesplnilo. Přesto jsem nyní spokojen s odvedenou prací i s vybraným zadáním.

Příloha – ovládání programu

Stiskem levého tlačítka myši a jejím pohybem dochází k rotaci scény kolem osy pozorovatele.

Použité klávesy:

w - Pohyb vpřed

s - Pohyb vzad

a - Úkrok doleva

d - Úkrok doprava

m - Drátový model

l - Zapnutí urychlovacích technik pro vykreslování terénu

Literatura

[1] http://www.flipcode.com/archives/Terrain_Texture_Generation.shtml

[2] http://www.flipcode.com/archives/Light_Mapping_Theory_and_Implementation.shtml

[3] Trent Polack, 3D Terrain programming

[4] http://nehe.ceske-hry.cz

[5] http://www.root.cz/serialy/graficka-knihovna-opengl/

[6] http://www.root.cz/serialy/fraktaly-v-pocitacove-grafice/

[7] http://www.cgtextures.com/

Seznam příloh

Příloha 1. CD