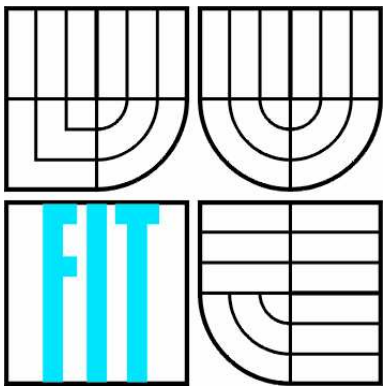
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Zobrazování terénu

Bakalářská práce

2005 Roman Schulz

Zobrazování terénu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Kadlece a uvedl v ní veškerou literaturu a zdroje, ze kterých jsem čerpal.

V Letovicích, dne 20. 4. 2006

……..

Abstrakt

Cílem této práce je navrhnout a implementovat systém pro zobrazení krajiny na počítači ve 3D prostoru s použitím rozhraní OpenGL. Systém by měl umožňovat zobrazení terénu, vegetace a dalších objetů, jako jsou částicové systémy, budovy, vozidla atd. Uživatel by měl mít možnost terén editovat a po spuštění programu se v něm libovolně pohybovat. Dále je potřeba mít možnost pro demonstraci zobrazit drátěný model terénu a objektů. Čtenář by měl být schopen porozumět principům zobrazení reálně vypadajícího terénu a měl by se seznámit s algoritmy pro zobrazení 3D scény a povrchu z výškových map.

Klíčová slova

Počítačová grafika, terén, generování terénu, ROAM, binární strom, zobrazení terénu, zobrazení objektů, výšková mapa, textury, detailní textury, GIS, OpenGL, šumová funkce, interpolace.

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu bakalářské práce Ing. Jaroslavu Kadlecovi, za jeho rady a připomínky k bakalářské práci.

Abstract

The purpose of this work is to design and implement system for display of landscape on personal computer in 3D space using OpenGL interface. This system should allow to display terrain, vegetation and other objects, like particle systems, buildings, cars etc. and have option to show wire model of terrain and objects for demonstration purpose. User should have option to edit terrain and while running of the program, support free movement. Reader should be able to understand principles of display realistic landscape and should learn algorithms for display of 3D scene and surface from height maps.

Keywords

Computer graphics, terrain, terrain generating, ROAM, binary tree, display of terrain, display of models, height map, textures, detail textures, GIS, OpenGL, noise function, interpolation.

Obsah

1. Úvod

Počítačová grafika je obor informatiky, který využívá počítačovou techniku na vytváření umělých snímků (tzv. renderování). Tento obor je možné dělit na několik oblastí: 3D rendering v reálném čase (často využívaný v počítačových hrách), počítačová animace, video, střih speciálních efektů (často využívané ve filmu a televizi), oditování obrázků a modelování.

V grafických aplikacích, jako jsou například letecké simulátory nebo počítačové hry, je často potřeba zobrazit terén. Ten může být vytvořen z dat o skutečné krajině nebo může být kompletně celý vygenerován na počítači. Při zobrazení rozsáhlého terénu se dříve či později narazí na problém spočívající v tom, že terén obsahuje obrovské množství dat a jejich zobrazení v reálném čase (při vysokém počtu zobrazených snímků za jednu sekundu) je v současných podmínkách téměř nemožné. Proto bylo vyvinuto několik algoritmů, které tento problém řeší tím, že různé části terénu zobrazí v různém stupni detailu.

Hlavním cílem této bakalářské práce je vytvořit aplikaci která umožní věrohodné zobrazení terénu. S tímto souvisí jednak implementace algoritmu pro vykreslení terénu, tak i zobrazení dalších objektů, které podpoří reálný vzhled, jako jsou např. budovy, vozidla, vegetace atd. Vstupní data bude možno zadávat v editoru, který bude součástí odevzdaného projektu.

Dalším cílem je zpracování teoretické části, která by měla poskytnout čtenáři ucelený přehled o možnostech zobrazení realistického terénu ve 3D grafice a popis principu nejčastěji používaných algoritmů. Naopak si tato bakalářská práce neklade za cíl shromáždit vyčerpávající teoretickou studii, která by se zabývala podrobným popisem všech algoritmů a jejich variantami, deformacemi terénu v reálném čase atd. Samozřejmostí je popis důležitých částí zdrojového kódu, použitých tříd, metod, funkcí a struktur.

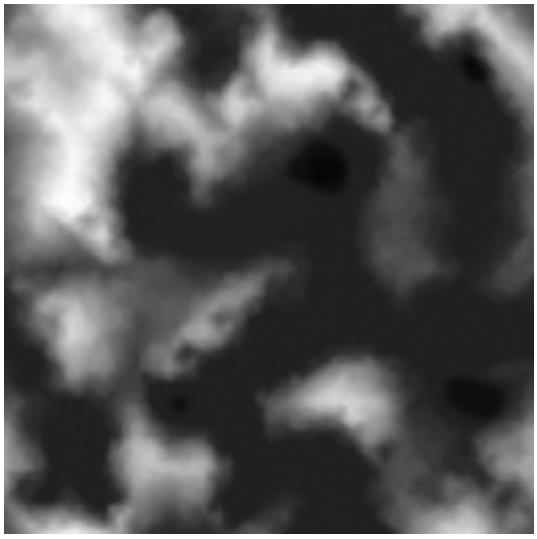
V první kapitole se čtenář seznámí s reprezentací terénu v počítačové grafice. Ve druhé kapitole jsou popsány nejpoužívanější algoritmy používané při zobrazení terénu a jeho generování. Ve třetí kapitole je seznámen s možnostmi vylepšení vzhledu terénu po vizuální stránce. Ve čtvrté kapitole je podrobněji rozepsána implementace a struktura programu. Následuje závěr, vyhodnocení výsledků a popis možného využití aplikace. Za těmito kapitolami následuje seznam použité literatury a příloha.

1.1. Rozhraní pro zobrazení 3D scény

Zobrazení objektů ve 3D je výpočetně velmi náročné, proto je vhodné využít některé již existující rozhraní. Rozšířená jsou v dnešní době rozhraní DirectX a OpenGL. Obě dvě jsou ve většině dnešních počítačích hardwarově akcelerovaná a tudíž velmi rychlá. Já jsem se pro zobrazení scény použil rozhraní OpenGL. Hlavním důvodem k tomuto výběru bylo to, že jej dobře znám a díky tomu jej mohu plně využít, narozdíl od konkurenčního DirectX. Program byl vytvořen pro operační systém MS Windows, ale není problém jej upravit tak aby fungoval i v dalších operačních systémech, protože implementace OpenGL existují pro prakticky všechny platformy.

1.2. Reprezentace terénu jako výškové mapy

Nejčastěji se v počítačové grafice na terén pohlíží jako na dvojrozměrné pole hodnot, které udávají výšku terénu v bodě daném dvěma souřednicemi. Tomuto poli se říká výšková mapa a jedná se vlastně o pravidelnou matici hodnot. Výhodou je možnost velmi rychle zjistit výšku terénu ve kterémkoliv bodě. Výšková mapa slouží jako vstupní data pro naprostou většinu algoritmů určených pro zobrazení terénu. Její hodnoty je možné pro lepší názornost také považovat za barvy šedi od černé až po bílou, kde černá barva jsou nejnižší místa terénu a bílá barva jsou nejvyšší místa terénu. Jedná se tedy vlastně o bitmapu a pro její editaci je možné využít libovolný dostupný grafický editor. Z výškové mapy lze snadno vytvořit trojúhelníkovou síť a zobrazit ve 3D.



Obrázek 1. Výšková mapa jako bitmapa

2. Algoritmy pro zobrazení terénu

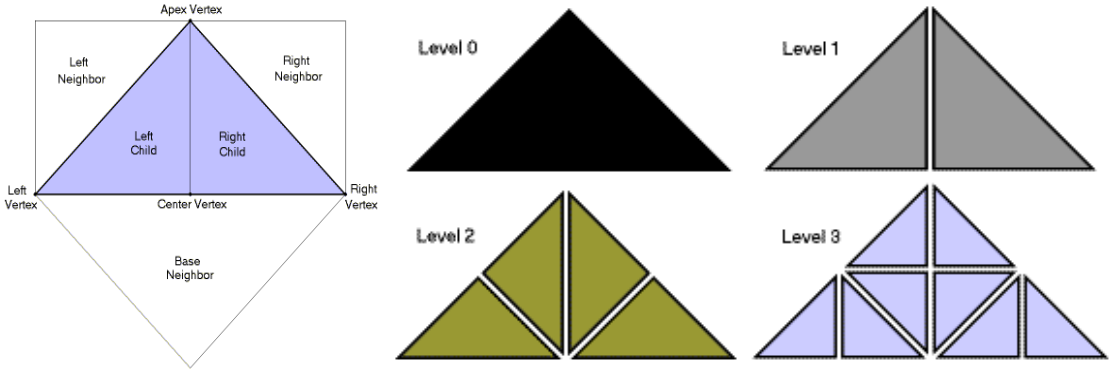
Algoritmů pro zobrazení terénu existuje celá řada, velice se ale liší efektivitou, množstvím zabrané paměti, škálovatelností atd. Algoritmus hrubé síly neřeší vůbec nic a vykreslí všechny trojúhelníky, ze kterých se terén skládá. Jméno tohoto algoritmu přesně vystihuje jeho nevýhodu. V případě rozsáhlého terénu je tento přístup velice nevhodný, protože je ve skutečnosti vidět jen malá část toho, co algoritmus vykresloval. Proto je vhodný jen pro testovací účely a nebude zde dále popisován.

2.1. Algoritmus ROAM

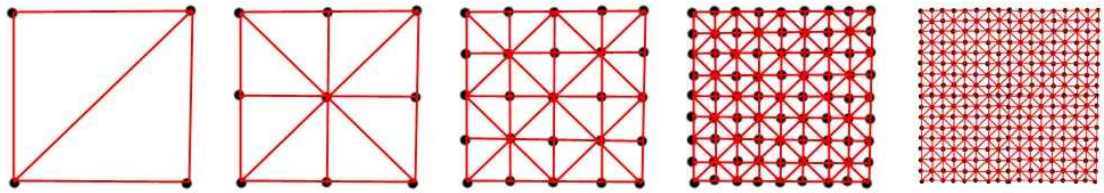
Algoritmus ROAM (Real-time Optimally Adapting Meshes) byl vyvinut v roce 1997 pro zobrazení terénu v leteckých simulátorech. Je použit v simulátoru T^3SIM při simulaci vzdušných soubojů stíhacího letadla JAS 39 Gripen. Hlavní výhodou tohoto algoritmu je možnost přizpůsobení členitosti terénu na základě výkonu PC. Algoritmus je založen na struktuře binárního stromu.

2.1.1. Příprava binárního stromu

Tento strom se vyznačuje tím, že každý uzel má právě dva potomky. Každý uzel tohoto stromu musí obsahovat ukazatel na dva jeho potomky, jednoho předka a tři jeho sousedy (viz. obrázek 6, který je převzat z literatury [4]). Strom je vytvořen tak, že se z celé výškové mapy vytvoří dva základní pravoúhlé trojúhelníky. Tyto trojúhelníky jsou rekurzivně půleny do té doby, než se při půlení dostane na nejmenší možný trojúhelník. Průběh vytváření binárního trojúhelníkové stromu je znázorněn na obrázku 6 (převzato z literatury [10])



Obrázek 6. Struktura uzlu binárního stromu(vlevo) a půlení trojúhelníků (vpravo).



Obrázek 7. Kroky při tvorbě binární trojhelníkové sítě

Po vytvoření binárního trojúhelníkového stromu je potřeba tento strom celý projít metodou post-order a nějakou funkcí ohodnotit důležitost každého trojúhelníku. Pro výpočet je vytvořena funkce, která spočítá odchylku ve středu přepony každého trojúhelníku mezi případem, že je trojúhelník rozpůlený a nerozpůleným trojúhelníkem. Každý rodič obsahuje součet hodnot obou jeho potomků. Tuto hodnotu je vhodné v každém snímku doladit podle vzdálenosti daného trojúhelníku od pozorovatele.

2.1.2. Rozložení na optimální trojúhelníkovou síť

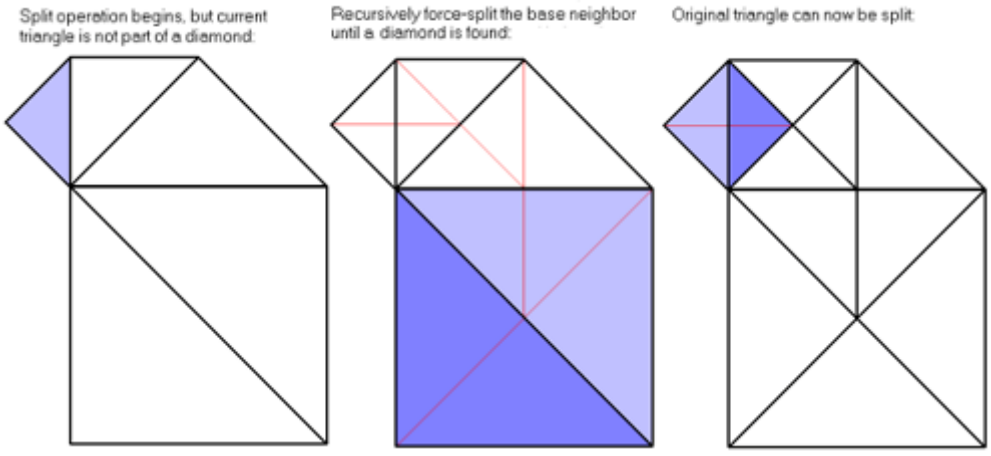
Rozložením terénu na optimální trojúhelníkovou síť se získá jen tolik trojúhelníků, kolik je třeba, přičemž se vyberou jen ty které jsou nejdůležitější z daného pohledu na terén. S tím vzniká problém, že se musí v každém snímku optimální rozložení přepočítat. Důvod je ten, že se pozorovatel po terénu pohybuje, rozhlíží atd a tím se mění důležitost jednotlivých trojúhelníků. Pro rozložení terénu je třeba zvolit hraniční hodnotu, která určuje, kdy se má trojúhelník dále rozpůlit nebo naopak zůstat nerozpůlený. Postupuje se od hlavního největšího trojúhelníku, a pokud je jeho důležitost tohoto trojúhelníku nižší než zvolená hodnota, trojúhelník již dále není půlen. Jinak dojde k rozpůlení a rekurzivně se otestují oba dva jeho potomci. Tímto vznikne optimálně vytvořená trojúhelníková síť.

2.1.3. Úprava hraniční hodnoty

Hraniční hodnotu, která udává v jakém případě se má trojúhelník dále rozdělovat, lze měnit pro každý snímek a udržovat tím plynulý počet snímků za jednu sekundu i při velmi odlišných scénách. To znamená že když je vykreslení scény pomalé a je zobrazeno příliš mnoho trojúhelníků, je možné snížit hraniční hodnotu a v příštím snímku tak vykreslit méně trojúhelníků. Nebo naopak, pokud je vykreslení rychlé, je možné zobrazit více trojúhelníků. Proto může být ze začátku hodnota zvolena zcela libovolně, po několika zobrazených snímcích se ustálí na optimální hodnotě (požadovaný počet snímků za sekundu nebo požadovaný počet vykreslených trojúhelníků) a při pohybu pozorovatele se automaticky přizpůsobí. V tom je obrovská výhoda tohoto algoritmu oproti ostatním. Tento algoritmus může být dále optimalizován tak, aby se při pohybu nebo rozhlížení pozorovatele v terénu přepočítala jen změněná data.

2.1.4. Zamezení vzniku chyb

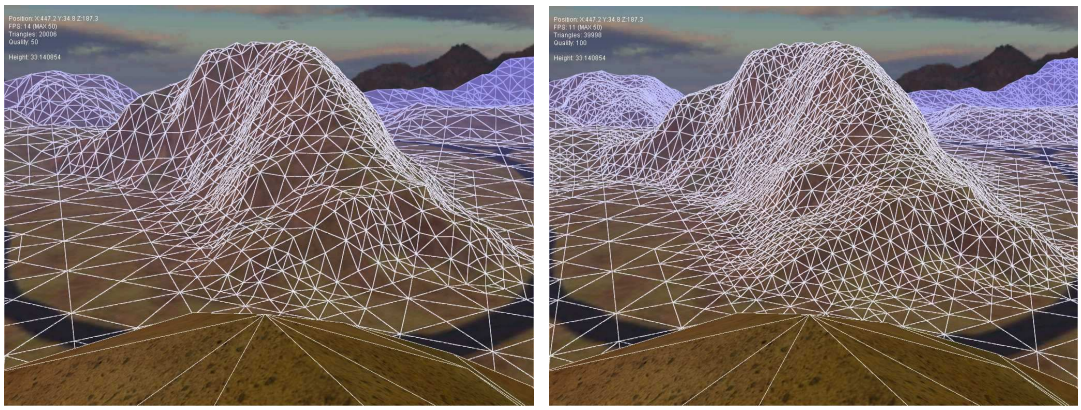
S rekurzivním dělením binární trojúhelníkové síťě na optimálně rozloženou síť souvisí problém s nuceným půlením trojúhelníků. Vždy, když je trojúhelník půlen, musí být půlen i jeho soused se kterým sdílí přeponu. Pokud by se tak nestalo, v terénu by byly vidět trojúhelníky, které na sebe nenavazují. Důvodem je to, že se střed přepony trojúhelníku nevyskytuje přesně v původním bodě (může být umístěn výš nebo níž než původní bod). Na obrázku 8 je znázorněn příklad nuceného rozpůlení několika trojúhelníků tak, aby nedocházelo k výskytu chyb. Obrázek je převzat z literatury [4].



Obrázek 8. Zamezení vzniku chyb při půlení trojúhelníků

2.1.5. Vykreslení optimální síťě

Nyní jsou připravena veškerá data pro zobrazení terénu. Jsou známy všechny trojúhelníky ze kterých se terén složí a nic nebrání jej předat rozhraní OpenGL, které je zobrazí. Na obrázku 9 jsou zvýrazněny trojúhelníky, které byly použity při vykreslení terénu v různém stupni detailu.



Obrázek 9. Stejný terén v různém stupni detailu

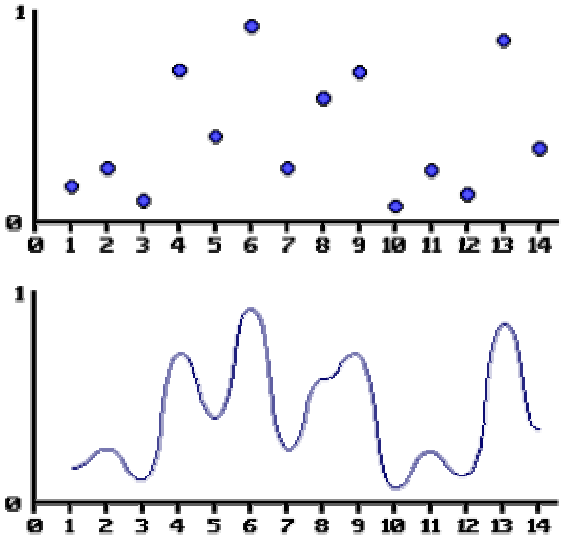
2.2. Generování výškové mapy

Vytvořit výškovou mapu je možné v libovolném grafickém editoru, jak bylo napsáno v úvodní kapitole. Mnohem lepší možností ale je nechat si výškovou mapu vygenerovat některým algoritmem pro generování šumu a výsledek jenom lehce upravit v grafickém editoru. K dispozici jich je velké množství, jedním z nejpoužívanějších je algoritmus Perlin Noise, který podává poměrně kvalitní výsledky generování. Další často používané algoritmy pro generování výškové mapy jsou Rided Perlin, Multi Perlin, Fractal Generation, Midpoint Displacement, Fault Formation, Spectral Synthesis Noise atd.

2.2.1. Popis algoritmu Perlin noise

Na tomto místě je popsáno z důvodu lepší srozumitelnosti pouze vytvoření jednorozměrného šumu. Pro víc rozměrů je princip úplně stejný, jen je potřeba 1 rozměr přidat do všech následujících funkcí. K vytvoření generátoru Perlin Noise šumu je potřeba v podstatě jen funkce diskrétního šumu a interpolační funkce. Funkce diskrétního šumu je v podstatě generátor náhodných čísel, který má jediný parametr, a vrací náhodné číslo v rozsahu 0-1, které závisí na parametru. Funkce musí vrátit stejné hodnoty v případě, že se zavolá dvakrát se

stejným parametrem. Interpolační funkce je kompromisem mezi rychlostí a kvalitou výstupu. Vygenerovaná šumová funkce je znázorněna na obrázku 2 (obrázky 2, 3, 4, 5 jsou převzaty z literatury [9]). Je možné sice použít lineární interpolaci, ale mnohem lepší výsledky lze dosáhnout použitím bilineární nebo kosinové interpolace. Dále je možné použít například kubickou nebo kvadratickou interpolaci.



Obrázek 2. Generátor náhodných čísel a interpolovaná funkce

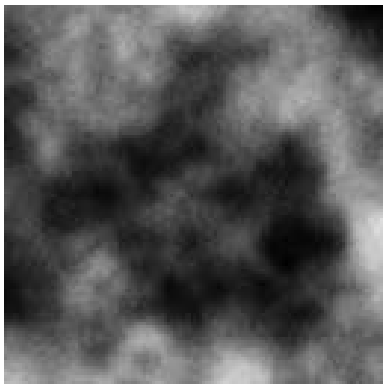
Zde se jedná o základní frekvenci. Frekvence této interpolované funkce se dá změnit velice jednoduše násobením požadovaného bodu na vodorovné ose (osa x). V případě že je potřeba 2 krát vyšší frekvence, stačí pouze najít bod s 2krát vyšší x-ovou souřadnicí. Šum s různými frekvencemi je znázorněn na obrázku:

Obrázek 3. Šum se základní, dvou a čtyř-násobnou frekvencí

Obrázek 4. Šum s osmi, 16-ti a 32-ti násobnou frekvencí

Každá následující frekvence se nazývá oktáva. Nyní je možné vypočítat hodnotu Perlin Noise šumu. Samotná funkce pro výpočet Perlin Noise šumu má 2 parametry, počet oktáv a počáteční frekvenci. Výpočet je velice jednoduchý, k šumu s počáteční frekvencí se přičte 1/2 hodnoty šumu s dvojnásobnou frekvencí. K výsledku se přičte 1/4 hodnoty šumu se čtyřnásobnou frekvencí a celý cyklus se opakuje pro každou další oktávu.

Podrobnější informace viz reference [9]. Všechny metody pro generování tohoto šumu jsou umístěny v souborech *CNoiseGenerator.cpp* a *CNoiseGenerator.h*.



Obrázek 5. Perlin noise pro 6 oktáv a frekvenci 1.0

3. Dosažení reálnosti zobrazení

Jelikož se samotná trojúhelníková síť moc nepodobá skutečnému terénu a pozorovatel si stěží dokáže představit terén, je nutné zobrazit několik dalších detailů. Každá mapa, která se na terén nanese mu mnohonásobně přidá na detailu. Nanesená může být například textura, bump mapa, detailní mapa atd.

3.1. Textura terénu

Ve většině grafických rozhraních může být povrch trojúhelníku popsán nejen barvou, ale také rastrovým obrazem. Takový obrázek se nazývá textura a udává, jak budou trojúhelníky nakonec vypadat. Většina dnešních grafických karet má omezenou maximální velikost textury, proto je nutné zobrazit buď menší části terénu s různými texturami, nebo zobrazit jednu hrubou texturu, a k ní přičíst detailní mapu. Z důvodu nižší paměťové náročnosti byl zvolen druhý způsob. Navíc je detailní mapa na terén aplikována tak, že není roztáhnuta po celém terénu jako hlavní textura, ale pravidelně se opakuje. Výhodou tohoto řešení jsou malé rozměry detailní textura.

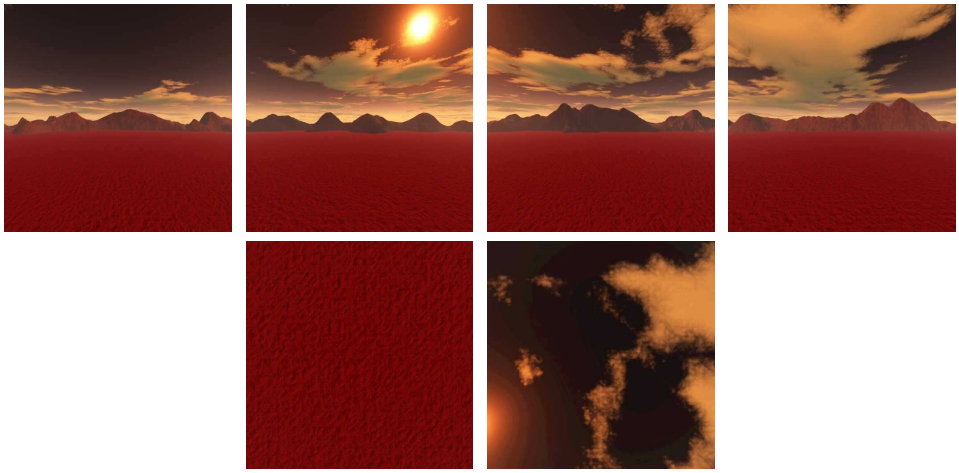
Obrázek 10. Terén bez textury (vlevo) a s texturou (vpravo)

Na obrázku 10. vlevo je zobrazen terén bez textury jen se stínováním a v pravo je zobrazena ta samá část terénu s nanesenou texturou. Rozdíl je patrný hned na první pohled a výhoda je v možnosti barevného rozlišení jednotlivých částí terénu (hory, pole, lesy atd.). Na obrázku 11 je k textuře přidána detailní mapa, která opět terénu přidala na reálném vzhledu.

Obrázek 11. Terén s detailní texturou

3.2. Textura okolí

Dalším prvkem, kterým lze zlepšit vizuální dojem, je zobrazit texturu okolí. Jiste si každý dokáže představit jak by vapadal terén, kdyby na pozadí nebyla obloha, ale jen černá barva? Vše spočívá v tom, že se vytvoří krychle, na jejíž strany se zobrazí textura. Tato textura zpravidla obsahuje buď oblohu, nebo oblohu a vzdálenou část krajiny. Příklad textur, které se zobrazí na krychli je na obrázku 12. Střed této krychle se umístí do pozice o souřadnicích kamery.



Obrázek 12. Textury okolí

Výhoda tohoto řešení je ve velmi snadné změně textury okolí v závislosti na denní době a počasí. Například za svítání je vhodné zobrazit texturu s vycházejícím sluncem, naopak v noci je možné zobrazit hvězdy a měsíc.

3.3. Stromy

Zobrazení stromů je nedílnou součástí zobrazení terénu. Proto byl navržen systém, kde každý strom může být reprezenovaný několika objekty v různém stupni detailu. Pokud je tento strom zobrazen v dálce, je vykreslen jako 2 obdelníky pokryté texturou. Naopak pokud je zobrazen těstně před pozorovatelem (kamerou), může být zobrazen jako objekt složený z několika tisíc trojúhelníků (tím pádem má mnohem vyšší detail). Přepínání stupně detailu závisí na důležitosti stromu (malé stromky a keře mají menší důležitost než obrovské stromy) a vzdálenosti stromu od pozorovatele (vzdálenější stromy jsou zobrazeny v menším detailu). Přepínání stupně detailu probíhá automaticky, pokud jsou definovány objekty pro různé stupně detailu. Jelikož nejsou k dispozici volně dostupné modely stromů ve vyšším detailu, jsou součástí projektu jen stromy v nižším stupni detailu.

Obrázek 13. Zobrazení terénu bez stromů (vlevo) a se stromy (vpravo)

Stromy obohatí terén o další důležité detaily a pro pozorovatele zajímavá místa. Jejich zobrazení mnohonásobně zlepší dojem samotného terénu, jak lze vidět na obrázku 13.

3.4. Další objekty

Zlepšení reálného vzhledu je možné dosáhnout také přidáním dalších objektů. Takové objekty mohou být například auta, budovy, zemědělské stroje, mosty, lávky, zdi atd. Tyto objekty by měly vtáhnout pozorovatele do děje a do doby ve které se scéna odehrává. Například je možné zobrazit historické vozy nebo futuristické zbraně.

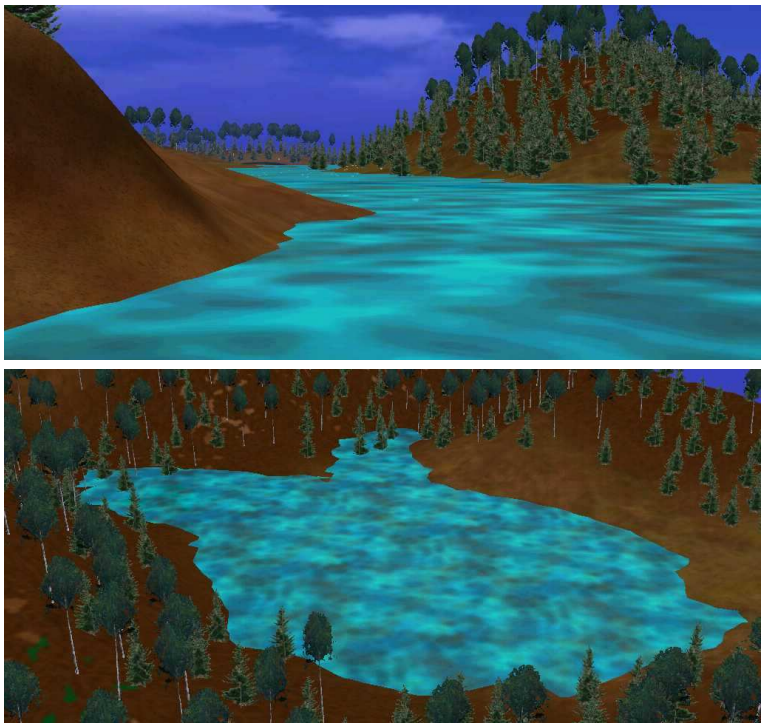
Objekty mohou být buď světla, částicové systémy, modely, kamera nebo skupina objektů. Všechny objekty jsou ve scéně hierarchicky uspořádány, tzn. je možné vytvořit skupinu objektů a při změně polohy této skupiny se změní poloha všech objektů, které skupina obsahuje. Pro bezproblémové vykreslení všech objektů je potřeba rozdělit části objektu na části neprůhledné a průhledné. To je nutné z principu vykreslování rozhraním OpenGL. Nejdříve je nutné nastavit všechna světla, potom kreslit pevné objekty a nakonec objekty s definovanou průhledností (například částicové systémy, průhledné části objektů - sklo atd.).

Obrázek 14. Objekty umístěné na terénu. Hořící tanker (vlevo) a dům (vpravo)

Objetky umístěné ve scéně mohou být nadefinovány v několika různých stupních detailu a při jejich vykreslení na obrazovku se vybere optimální stupeň detailu. Tímto lze opět dosáhnout zrychlení vykreslování objektů které jsou umístěny daleko od pozorovatele. Na obrázku 14 je uveden příklad dvou objektů umístěných na terénu.

3.5. Vodní hladina

Nedílnou součástí terénu by také měla být možnost zobrazení menší vodní hladiny. Pro jednoduchou vodní hladinu bez vlnek stačí zobrazit rovnou plochu s poloprůhlednou texturou. Pro lepší dojem jsem nastavil texturu tak, aby se pohybovala ve směru větru.



Obrázek 15. Zaplavené nížiny (nahoře) a malé jezírko (dole)

3.5. Částicové systémy

Částicové systémy jsou zásobníky mnoha malých částic zobrazených jako čtverce s částečně průhlednou texturou. Díky nim je možné snadno naprogramovat kouř, exploze, efekty zásahů střel, vodní fontány a jiné efekty. Tyto systémy se proto využívají ve většině dnešních her. Mnou navržený částicový systém se skládá ze třídy CEmitor, která se stará o správu částic (jejich vytváření, rušení a aktualizaci v čase), a struktury TParticle, která uchovává informace o každé částici (pozice, barva, rychlost, směr atd.)

Částic mohou být zobrazeny tisíce a velmi často dochází k vytváření nových a rušení starých. Alokování paměti pro každou zvlášť by zabralo příliš mnoho času. Proto jsem použil datovou stukturu dvojitého lineárního seznamu. Pri vytvoření nové částice se alokuje paměť (pokud je seznam s nepoužívanými částicemi prázdný) a uloží se na počátek lineárního seznamu s aktivními částicemi. Jakmile vyprší platnost částice, nedojde k uvolnění paměti ale částice se pouze přesune do druhého lineárního seznamu, ve kterém jsou uloženy struktury nepoužívaných částic. Pokud je v tomto seznamu uložena alespoň jedna částice tak se při vytváření nové částice

použije (tj. pro novou částici se použije dříve alokovaná paměť). Výhoda tohoto řešení je efektivita částicového systému při práci s pamětí.

Obrázek 16. Příklady částicových systémů (fontána, hořící dům, sníh, ohnivý kruh)

4. Implementace

4.1. Zobrazení terénu

Program, který zobrazuje scénu ve 3D rozměru se jmenuje 3Dengine-release.exe. Tento program se ovládá pomocí konfiguračního souboru. V něm lze nastavit výšku a šířku výsledného okna, barevnou hloubku, hloubku Z-Bufferu, zda se má program spustit v režimu celé obrazovky, kvalitu zobrazení objektů, reakce na pohyby myši a soubor se skriptem. Jako skriptovací jazyk jsem použil jazyk LUA, který je poměrně rozšířený. Vybral jsem jej proto, že jej lze snadno integrovat s programem C++. Příklad použití skriptu uvedu později v textu.

Hlavní funkce programu je umístěna v souboru main.cpp. Nejdříve je vytvořen soubor pro ukládání kontrolních výpisů (log.txt). Pro snadnou práci s ním byla implementována třída *CLog* v sobuoru *CLog.cpp* a *CLog.h*. Poté jsou načteny údaje z konfiguračního souboru. Třída pro načítání *.ini souboru se jmenuje *CIniFile* a je umístěna v souborech *CIniFile.cpp* a *CIniFile.h*.

Následně je vytvořeno hlavní okno aplikace podle požavků v nastavení. Pro tento účel byla vytvořena třída *CWindow* v souborech *CMyWindow.cpp* a *CMyWindow.h*. Okno je vytvořeno pomocí Win API funkcí a je možno nastavit hodnotu gamma, která určuje křivku jasu.

Dále je inicializováno rozhraní OpenGL. Celé nastavení bylo zahrnuto do jedné třídy *CRenderDevice* která je uložena v souborech *CRenderDevice.cpp* a *CRenderDevice.h*. Tato funkce vytvoří rendering context potřebný pro OpenGL a inicializuje toto rozhraní. Třída dále obsahuje metody pro nastavení rozšíření OpenGL - multitexturingu, kompresi textur a anizotropního filtrování. Dále jsou implementovány metody pro uložení aktuálního snímku a testování chyb.

Poté je vytvořen objekt typu *CDirectInput* pro ovládání vstupu (myši a klávesnice) a *CSoundManager* pro zvukový výstup. Další bod inicializace je vytvoření rozhraní pro spouštění skriptů LUA. Toto rozhraní je umístěno ve třídě *CScriptManager* v souborech *CScriptManager.cpp* a *CScriptManager.h* a právě zde jsou definovány funkce, které lze z jazyka LUA volat. Poté se spustí funkce *Init* ze skriptovacího souboru, tím dojde k načtení

všech objektů do hierarchické struktury scény (kamery, modely, světla, skybox, terén, emitory atd.).

Následuje vytvoření dialogů pro nastavení některých hodnot přímo z programu bez nutnosti spouštět program znovu. Tyto dialogy mohou obsahovat okna, tlačítka, checkboxy, listboxy, panely, posuvníky a další prvky. Dialogy jsou zobrazeny přímo rozhraním OpenGL. Pro správu dialogů slouží všechny soubory a třídy v adresáři source/GUI.

Poté následuje smyčka, která se opakuje pro každý vykreslený snímek do té doby, než program skončí. V první části této smyčky dojde ke zpracování zpráv windows, aktualizaci vstupu od uživatele a vygenerování některých fyzikálních veličin (vítr). Následuje aktualizace scény v závislosti na pohybu uživatele. Poté dojde k zjištění viditelnosti všech objektů z pohledu uživatele. a vypočítá se detekce kolizí. Následuje rozložení terénu na optimální trojúhelníkovou síť a poté začne vykreslování. Nejdříve je vykreslena textura pozadí, poté jsou nastavena všechna světla a dojde postupně k vykreslení terénu, všech pevných objektů, průhledných objektů a částicových systémů. Poté jsou vykresleny v případě potřeby i další data (drátěné modely, obálky objektů, normály trojúhelníků atd.). Následně dojde k vykreslení dialogů a informací na obrazovku ve 2D režimu zobrazení. Toto celé se opakuje do té doby, než uživatel klikne na tlačítko konec.

Poté dojde k uvolnění veškeré alokované paměti. Pro testování zda všechna paměť uvolněna byla do projektu zahrnuta knihovna Memory manager (*MMGR*) od Fluid Studios.

Implementace terénu je umístěna ve třídách *CTerrain* a *CPatch* v souborech *CTerrain.h* a *CTerrain.cpp*. Třída *CTerrain* má za úkol spravovat terén jako celek, narozdíl od třídy *CPatch*, která souvisí s částmi terénu o předem dané velikosti (například 32x32 bodů).

CTerrain obsahuje metody pro inicializaci terénu, načtení dat ze souboru vygenerovaného editorem, výpočet normál, správu všech dlaždic, ze kterých se terén skládá a vykreslení terénu, stromů a vody. Pro načtení terénu ze souboru vygenerovaného editorem slouží metoda *Load*, která nastaví všechny vlastnosti terénu, jako je velikost rastru, načte textury, výškovou mapu a všechny stromy, následně dojde k výpočtu normálových vektorů a inicializaci dlaždic, ze kterých se struktura terénu skládá.

Metoda *LoadHeightMap* načte výškovou mapu z obrazového souboru, alokuje paměť pro všechny trojúhelníky a nastaví správné souřadnice všech bodů, ze kterých je terén složen. Alokování veškeré paměti je uloženo v metodě *Init*. V každém snímku je volána nejdříve metoda *Reset* a následně metoda *Render*. Metoda *Reset* otestuje viditelnost všech dlaždic z

pohledu pozorovatele, rozloží terén na optimálně vytvořenou trojúhelníkovou síť a vypočítá úroveň detailu všech viditelných stromů. V metodě *Render* je z optimálně rozložené sítě vygenerováno pole trojúhelníků, které je následně vykresleno na obrazovku. Zobrazení textury a detailní mapy současně na každém trojúhelníku je provedenou technikou tzv. multitexturingu, která spočívá v zobrazení několika textur současně. Tuto techniku musí podporovat grafická karta počítače. Pokud ji počítač nepodporuje, je každý trojúhelník vykreslen 2krát po sobě, přičemž dojde k sečtení hodnot obou textur. Tato technika se nazývá míchání barev (z angl. blending). Výsledek je v tomto případě mírně pomalejší, protože tato operace není hardwarově akcelerovaná, ale lze ji použít na všech grafických kartách. Navíc oba dva algoritmy poskytují naprosto stejný výsledek. Po vykreslení terénu je volitelná možnost vykreslení drátového modelu terénu a normálových vektorů v každém bodě. Následně je podle požadavku na počet zobrazených trojúhelníků upravena hodnota, která udává hranici zda se má každý trojúhelník při vytváření optimální sítě dále půlit nebo už nikoliv. Dále jsou v této třídě k dispozici metody *RenderTrees* a *RenderWater*, které na obrazovku vykreslí stromy a vodu. Třída ještě obsahuje metodu *GetHeight*, která vypočítá interpolací výšku terénu v libovolném bodě mezi známými hodnotami.

Třída *CPatch* obsahuje metody pro výpočet důležitosti jednotlivých trojúhelníků, rozložení na optimální trojúhelníkovou síť a připravení trojúhelníků do pole pro vykreslení. Metoda *Reset* provede inicializaci dlaždice, vytvoří dva její hlavní trojúhelníky a provede napojení těchto trojúhelníků s okolními dlaždicemi. Metoda *Split* provede rozpůlení libovolného trojúhelníku a v případě potřeby se postará o rozpůlení sousedních trojúhelníků tak, aby nedocházelo k žádným chybám (viz. kapitola 2.1.4). Metoda *ComputeBoundingBox* vypočítá obálku dané dlaždice, podle které se počítá viditelnost celé dlaždice. Metody *ComputeVariance(Recurse)* jsou vytvořeny pro výpočet důležitosti všech trojúhelníků. Metody *Tesselate(Recurse)* slouží k rozložení terénu na optimální trojúhelníkovou síť v závislosti na poloze pozorovatele a členitosti terénu. Metody *Render(Recurse)* provedou transformaci vytvořené trojúhelníkové síťě na jedno velké pole trojúhelníků, které na dnešních grafických kartách možné vykreslit mnohonásobně rychleji než každý trojúhelník samostatně.

4.2. Editor terénu

Editor terénu je program, který umožňuje vizuální návrh terénu. Program umí načítání a ukládání výškové mapy a textury terénu z obrazového souboru, zobrazení mapy v nastavitelném měřítku, snadé vkládání předdefinovaných objektů na libovolné místo (stromy, keře atd.),

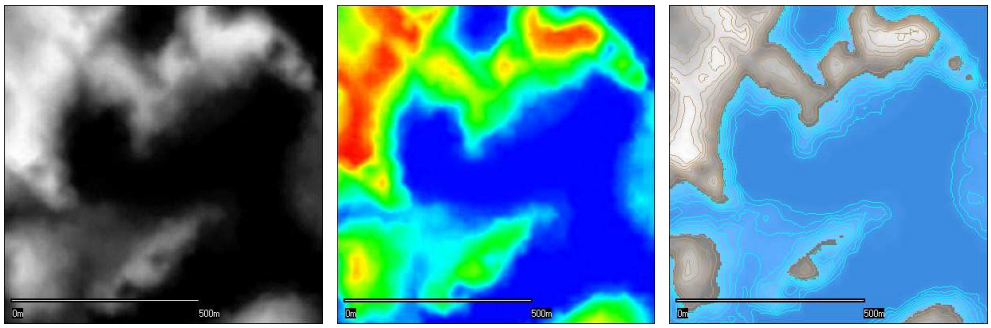
nastavení základních parametrů terénu (vzdálenost rastru, převýšení atd.) a vizuální nastavení výšky vodní hladiny. Výslednou práci je možné uložit (např. do XML souboru) a zpětně načíst.

Pro vytvoření programu jsem se rozhodl využít rozhraní zvané WinAPI. Vypočítání vrstevnic terénu je operace hodně náročná na výkon počítače a při pohybu v mapě či změně měřítka je nutné vrstevnice vždy přepočítat. Toto se mi díky rychlosti WinAPI podařilo implementovat bez zbytečného čekání. Pro vstup a výstup dat jsem se rozhodl použít informace uložené v XML souboru, protože je přehlednější než obyčejný textový nebo binární soubor a lze jej ručně editovat, což bylo nutné při ladění výsledného formátu tohoto souboru.

Editor využívá části kódu napsané pro samotnou bakalářskou práci, využil jsem například kód pro načítání textur různých druhů formátů (JPG, BMP, TGA). Toto je nutné pro sladění možností editoru a samotného programu, který zobrazuje terén ve 3D.

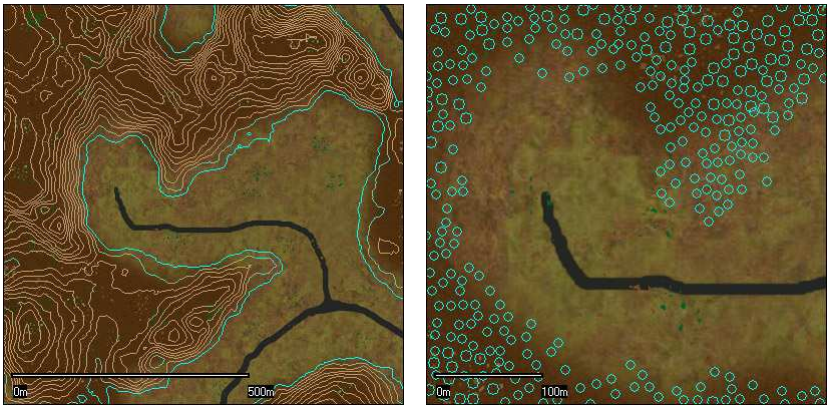
4.2.1. Příklad práce s programem

Po spuštění programu je vhodné nejdříve načíst ze souboru výškovou mapu nebo ji vygenerovat. Načtenou výškovou mapu je možné zobrazit černobíle (tmavé body znamenají nižší místa a světlé body znamenají vyšší místa) nebo pro lepší názornost obarveně. Zároveň je možné zapnout zobrazení vrstevnic a vodní hladiny. Vrstevnice nad vodní hladinou jsou zobrazeny hnědě a vrstevnice pod vodní hladinou jsou zobrazeny modře.



Obráezk 17. Zobrazení výškové mapy, obarvené výškové mapy a mapy s vrstevnicemi

Po načtení výškové mapy je vhodné načíst texturu terénu a potom vygenerovat nějaké stromy



Obrázek 18. Mapa s načtenou texturou a mapa se stromy

Poté stačí terén uložit do XML souboru. Výsledek je zobrazen na obrázku níže:

Obrázek 19. Výsledná scéna

4.2.2. Vlastnosti programu

Editor terénu je program, který je určen pro vizuální návrh terénu. Hlavní vlastnosti programu jsou:

- Načítání výškové mapy a textur ze souborů BMP, TGA a JPG
- Umístění jednoho nebo více stromů pouhým kliknutím myši
- Zobrazení vrstevnic z výškových map
- Nastavení výšky vodní hladiny
- Generování výškové mapy algoritmem fBm (fractional Brownian motion)
- Import a export všech dat z/do XML souboru
- Nastavení vlastností terénu: Převýšení, velikost rastru atd.
- Přibližování mapy a pohyb v ní

5. Závěr

Velká výhoda v použitém algoritmu ROAM je v jeho obrovské škálovatelnosti podle počítače na kterém běží. Není žádný problém zobrazit terén na počítači se slabým CPU a integrovanou grafickou kartou v nízkém detailu a na jiném počítači s rychlým CPU a moderní grafickou kartou ve velmi vysokém detailu.

Tato práce mi byla přínosem v tom, že jsem si prakticky například vyzkoušel práci se strukturami dvojitého lineárního seznamu, binárního a obecného stromu a jejich návrhu. Také jsem se díky ní velmi zdokonalil v používání rozhraní OpenGL a blíže jsem se seznámil s funkcemi Win API. Díky zvolenému typu aplikace (zobrazení dat v reálném čase) jsem mnohokrát musel kód přepisovat tak aby byly funkce, metody a struktury efektivní. Poznal jsem, že programování není jen zábava ale i řešení různě složitých problémů.

Návaznost programu na další projekty podobného typu je možná, například jej lze využít pro zobrazení výškových map získaných z vytištěné mapy (práce Vytvoření výškové mapy z vrstevnic v rastrové mapě z roku 2006).

5.1. Možné využití aplikace a další vývoj projektu

S rostoucím výkonem počítačů se zlepšují možnosti při zobrazení velkého množství dat. V současné době je možné zobrazit stále více dat v reálném čase, a tudíž lze dosáhnout velmi kvalitních výsledků. Tím pádem je dobré uvažovat na smysluplném využití tohoto výkonu.

Zobrazení terénu je možné využít v průmyslu, například v leteckých simulátorech při výuce pilotů. V tomto případě je zobrazení na počítači jedinou možností, jak bezpečně naučit pilota ovládat daný stroj na zemi, ještě před tím než poprvé vzlétne. V poslední době dochází velmi často k záplavám. Při návrhu protipovodňových opatření by mohlo pomoci zobrazení výsledků simulace záplavových oblastí v blízkosti vodních toků.

Další možné využití zobrazení terénu by mohlo být v kartografii a GIS. Při spojení programu s reálnými mapami by jejich zobrazení ve 3D by mohlo být být využito například při sledování polohy dopravních prostředků, polohy chráněných živočichů (která by se získala přes vysílačku), navigaci a plánování trasy při procházení turistických cest a cyklistických stezek.

Aktuální poloha se získá přes GPS a mapa by byla zobrazená na přenosném počítači nebo pocket PC (popřípadě v telefonu s větším displejem).

Obrázek 20. Porovnání letecké fotografie a výsledné scény

Zobrazení terénu může být využito i v zábavném průmyslu v počítačových hrách nebo například při točení nákladných filmových scén (vytvořit počítačový model terénu může být mnohem levnější varianta než výprava přes půl světa na požadované místo nebo například do vesmíru - v případě sci-fi terénu). Další možností je využití v reklamě, například při prodeji objektů nebo stavebních pozemků. Zájemce si může stáhnout data a po pozemku se projít, aniž by musel odejít od počítače. Aplikaci lze využít dále například pro armádní účely, při zobrazení rozmístění vojenských jednotek.

Myslím si, že program který jsem vytvořil je pro demonstraci postupů zobrazení terénu ve 3D grafice plně vyhovující. Snažil jsem se o co nejvyšší možnost budoucí rozšiřitelnosti, efektivní provádění operací a co nejjednoduší ovládání. Podařilo se mi implementovat algoritmus ROAM a poměrně efektivně zobrazovat terén o velikosti 2x2km s rozlišením 5m. Díky editoru lze terén snadno vytvářet a modifikovat. Pro reálnější vzhled jsem přidal možnost načítat další objekty, modely a přidal jsem různé speciální efekty, jako jsou oheň, voda, dým atd. Terén by do budoucna bylo možné vylepšit v kvalitnějším zobrazení vodní hladiny, zobrazením vržených stínů pod objekty, podporováním moderních pixel a vertex shaderů (to jsou programy které neběží na CPU ale na grafickém procesoru). Dále by bylo možné implementovat knihovnu pro fyzikální výpočty a lépe počítat kolize objektů mezi sebou. Nakonec by bylo dobré uvažovat o implementaci prostorového zvuku a deformaci terénu (při výbuchu atd.)

6. Literatura

[1] Mark Duchaineau, Murray Wolinsky, David E.Sigeti, Mark C. Miller, Charles Aldrich, Mark B.Mineev-Weinstein, ROAMing Terrain: Realtime Optimally Adapting Meshes, IEEE Visualization '97 Proceedings, 1997

[2] Trent Polack,Game Development Series: Focus on 3D terrain programming, Premier Press, USA, Ohio, 2003, 239 stran, ISBN 1-59200-028-2

[3] Tom Davis, Jackie Neider, Dave Shreiner, Mason Woo, OpenGL Programming Guide, Addision Wesley, 1997, 410 stran, ISBN: 0-201-63274-8

[4] Bryan Turner, Real-Time Dynamic Level of Detail Terrain Rendering with ROAM, Gamasutra, 2000, dostupné na adrese http://www.gamasutra.com/features/20000403/turner_01.htm, 17.4.2006

[5] Yordan Gyurchev, ROAM Implementation Optimizations, Flipcode, 2001, dostupné na adrese http://www.flipcode.com/articles/article_roamopt.shtml, 17.4.2006

[6] Microsoft software development network, Microsoft Corporation, 2006, dostupné na adrese http://msdn.microsoft.com

[7] Encyklopedie Wikipedia, 2006, dostupné na adrese http://en.wikipedia.org, http://cs.wikipedia.org

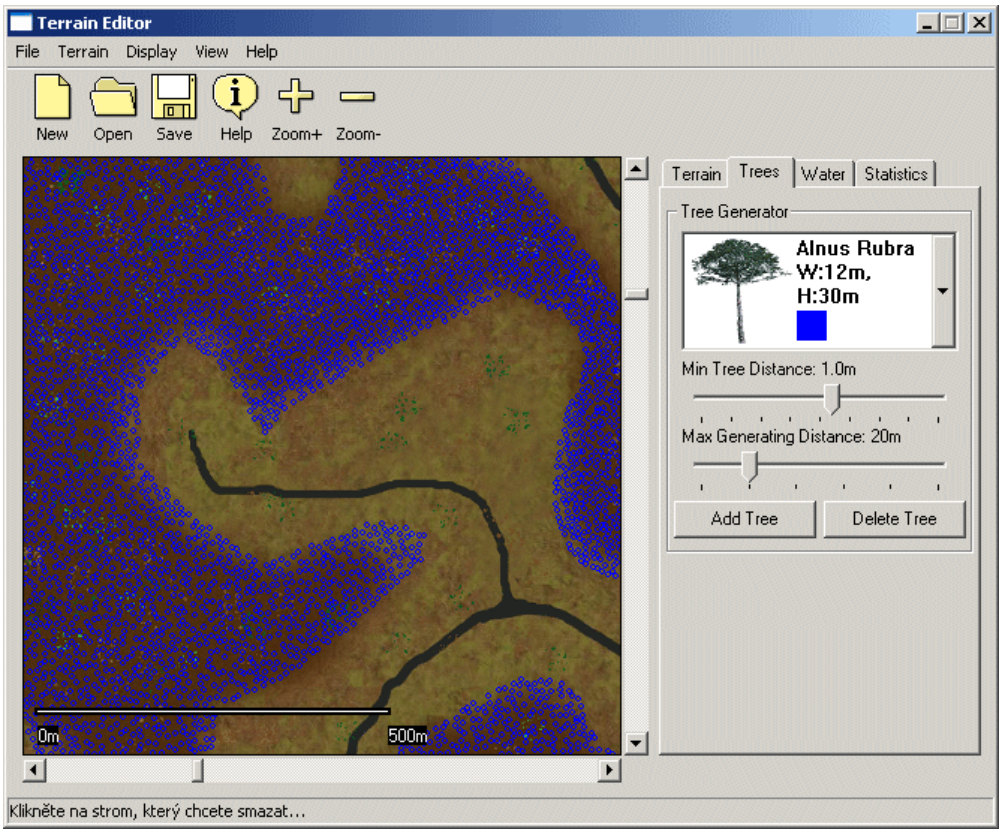
[9] Hugo Elias, Perlin Noise, 2003, dostupné na adrese http://freespace.virgin.net/hugo.elias/models/m_perlin.htm, 17.4.2006

[10] Virtual Terrain Project, dostupné na adrese http://www.vterrain.org, 17.4.2006

7. Příloha 1 - Editor terénu

7.1. Hlavní okno editoru

Aplikaci tvoří okno, které je rozděleno na několik částí: Menu, Nástrojová lišta, Záložky z editačními funkcemi, Mapa s posuvníky, Zobrazení stavu úlohy (je viditelné jen když je třeba) a Stavový řádek



Obrázek 21. Hlavní okno editoru terénu

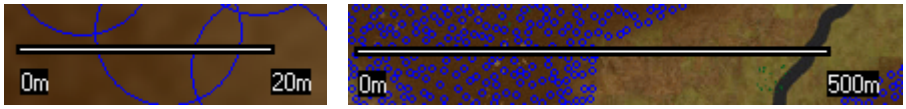
7.2. Menu

Menu programu se skládá z 5 položek (File, Terrain, Display, View, Help). Popis jednotlivých položek:

- **File/New** Začne editovat nový terén. Všechna dříve vytvořená data budou smazána
- **File/Open...** Otevře z XML souboru dříve terén
- **File/Save As...** Uloží do XML souboru právě vytvořený terén
- **Terrain/Size** Nastaví velikost terénu a potřebné rozlišení výškové mapy. Možnosti jsou 128x128, 256x256, 512x512 a 1024x1024
- **Terrain/Height** Nastaví převýšení terénu (rozdíl mezi nejnižším a nejvyšším bodem). Možnosti jsou 20m, 50m, 100m, 200m. Tato položka má vliv pouze pro export dat a je využívána až při zobrazení v 3D enginu.
- **Terrain/Grid Size** Nastaví vzdálenost dvou sousedních bodů v rastru. Možnosti jsou 1m, 2m, 5m, 10m. Tato položka má vliv pouze pro export dat a je využívána až při zobrazení v 3D enginu.
- **Terrain/Contour Height** Nastavení této položky má vliv na zobrazení vrstevnic. Ty mohou zobrazovat převýšení 1m, 2m, 5m, 10m nebo 20m
- **Display/Water** Nastaví, zda se má v 3D enginu zobrazovat vodní hladina
- **Display/Trees** Nastaví, zda se mají v editoru zobrazovat stromy
- **Display/Contour** Nastaví, zda se mají v editoru zobrazovat vrstevnice
- **Display/Height Map** Nastaví, zda se má v editoru zobrazovat výšková mapa
- **Display/Texture** Nastaví, zda se má v editoru zobrazovat textura terénu
- **Display/Color Map** Nastaví, zda se má v editoru zobrazovat obarvená výšková mapa
- **View/Put Near** Přiblížení mapy
- **View/Take Away** Oddálení mapy
- **View/Take Max Away** Maximální oddálení mapy
- **Help/About** Zobrazí informace o programu

7.3. Mapa

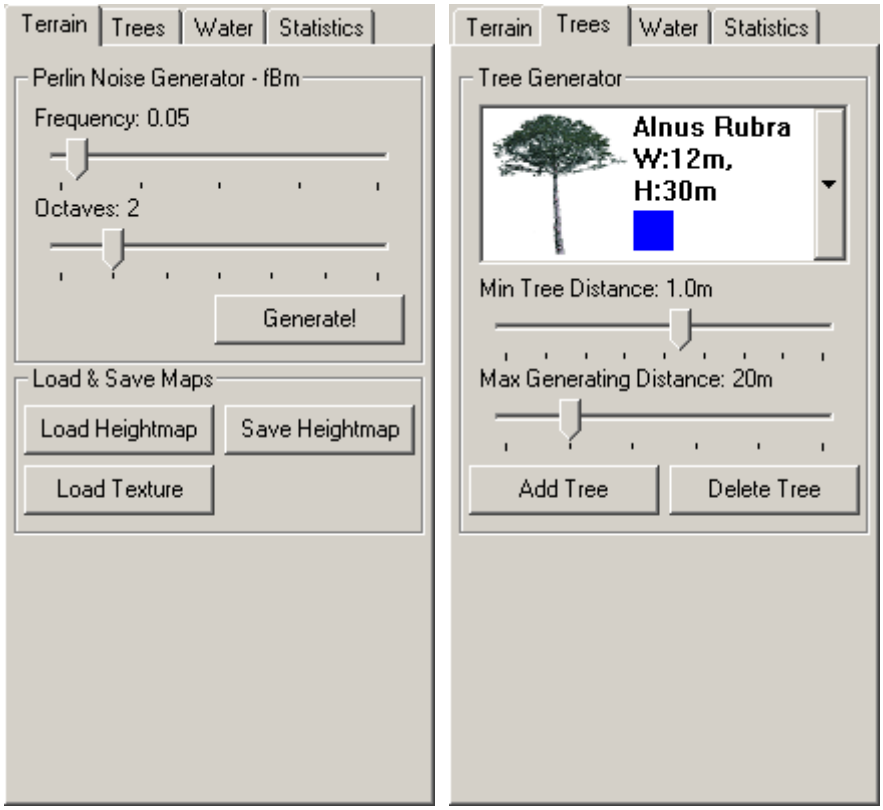
Ovládání mapy je intuitivní. Pro posun v mapě slouží posuvníky po stranách mapy. Mapa se dá přiblížit nebo oddálit tlačítky „+“ a „-“ v nástrojové liště nebo otáčením kolečkem myši při současně stisknuté klavese CTRL. V levém spodním rohu mapy se zobrazuje měřítko, které se automaticky přizpůsobuje přiblížení mapy (a také v závislosti na velikosti rastru):



Obrázek 22. Měřítko s automatickým přizpůsobením přiblížení

7.4. Záložky s editačními funkcemi

Program obsahuje 4 záložky pro editaci a nastavení různých vlastností terénu. Patří mezi ně záložka Terrain, Trees, Water a Statistics.



Obrázek 23. Záložky Terrain, Trees

Záložka Terrain - V této záložce jsou v sekci **Perlin Noise Generator** parametry pro vygenerování výškové mapy algoritmem fractional Brownian motion. Položka **Frequency** udává členitost terénu a položka **Octaves** udává vrásčitost terénu. Tlačítkem Generate! se spustí generování terénu, které může podle velikosti terénu trvat několik desítek sekund. V sekci Load & Save Maps jsou umístěny tlačítka pro načítání a ukládání výškové mapy a textury z/do obrazového souboru.

Záložka Trees – V této záložce je jediná sekce **Tree Generator**. Tato sekce je určena pro nastavení vkládání stromů do mapy. Je nutné nejdříve ze seznamu vybrat typ stromu, který se bude do mapy vkládat. V současné době je k dispozici kolem 40 druhů stromů. Potom se v položce **Min Tree Distance** nastaví minimální vzdálenost stromů od sebe. Tato položka může nabývat kladných i záporných hodnot. V případě kladných hondot udává jak nejblíže budou generované stromy od sebe. V případě záporných hodnot mohou být generované stromy částečně prolnuty, například korunami. Položka **Max Generating Distance** udává, jak daleko budou stromy generovány od pozice kliknutí myší. Po stisknutí tlačítka **Add Tree** je možno generovat stromy kliknutím do mapy nebo tažením myši se stisknutým levým tlačítkem. Nové stromy se generují do vzdálenosti Max Generating Distance od pozice myši a s minimální vzdáleností od ostatních stromů Min Tree Distance. Po stisknutí **tlačítka Delete Tree** je možné mazat stromy do vzdálenosti Max Generating Distance.

Záložka Water – Tato záložka obsahuje jedinou sekci **Water Settings** ve které je posuvník **Water Height**, kerým se nastavuje výška vodní hladiny. Změna hodnoty posuvníku se okamžitě projeví v mapě za předpokladu že je v menu zapnuta položka **Display/Water.**

Záložka Statistics – Tato záložka je prozatím bez významu.