INFINITE TERRAIN RENDERING IN REAL TIME

Tomáš MRKVIČKA, Bachelor Degree Programme (3)
Dept. of Computer Graphics, FIT, BUT
E-mail: xmrkvi03@stud.fit.vutbr.cz

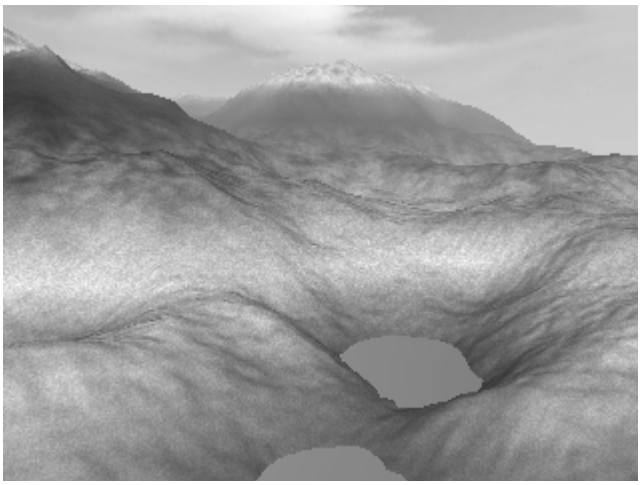
Supervised by: Dr. Adam Herout

ABSTRACT

This paper contains information about infinite terrain rendering based on the tile terrain algorithm. The main goal of this work is to create usable terrain for a computer game that includes terrain and texture generation, algorithms for visibility determining, level of detail algorithms and terrain interaction with others objects in the game.

1 ÚVOD

Terén a jeho zobrazení jsou v dnešní době součástí velkého množství grafických aplikací. Nejčastěji se pochopitelně jedná o počítačové hry, ale využití lze hledat i mimo odvětví zábavního průmyslu – např. v profesionálních leteckých nebo vojenských simulátorech. Většina dnešních her (a často i ostatní aplikace) bohužel implementuje použitý terén pouze jako konečnou plochu, což snižuje realističnost výsledné aplikace. Vytvoření skutečně nekonečného terénu je totiž stále příliš náročné na výpočetní systém dnešních osobních počítačů. Cílem této práce je pokusit se navrhnout a implementovat systém, který umožňuje zobrazování nekonečného terénu v reálném čase a zároveň příliš nezatěžuje výpočetní systém. Výsledkem práce je nekonečný terén pro hru TankGame vyvíjenou na ÚPGM FIT VUT v Brně.



Obr. 1: *Terén ve hře TankGame*

2 ROZBOR

Cílem práce je vytvoření nekonečného terénu pro počítačovou hru a z toho vyplývá základní požadavek – terén se musí nejen zobrazovat, ale být také plně interaktivní. Dalším požadavkem na terén ve hře je minimální vytížení procesoru, protože celá hra obsahuje velké

množství objektů, které také spotřebují velké množství procesorového času (umělá inteligence, výpočty fyziky,…).

Nekonečnosti terénu je dosaženo neustálým opakováním konečného čtvercového terénu. Z toho vyplývá, že požadovaný konečný terén musí navazovat ve svých protilehlých hranách. Použitý přístup je velmi podobný zemskému povrchu, kdy po cestě jedním směrem dojdeme časem na stejné místo. Dalším možností by bylo neustále generovat nový terén za běhu aplikace. Tento způsob je zatím příliš náročný na výpočetní systém, nicméně řešení popsané v následujících odstavcích lze drobnými úpravami změnit na tento způsob při zachování všech základních mechanismů.

Samotnou práci lze rozdělit do několika kategorií, které budou rozebrány v následujících odstavcích.

2.1 PŘÍPRAVA TERÉNU

Jak bylo uvedeno výše, vytvářený terén musí splňovat podmínku návaznosti jednotlivých protilehlých hran. Pro vygenerování takového terénu je použit algoritmus Diamond-Square [2]. Pro lepší výsledek je celý algoritmus upraven tak, že nejprve generuje základní tvar krajiny a naposledy generuje drobné nerovnosti, jejichž velikost lze ovlivňovat zadanými parametry.

Speciálním požadavkem pro počítačovou hru je možnost vytvořit při generování terénu na zadaných místech rovné plochy, na které bude možné umístit objekty, které rovný podklad vyžadují (např. domy).

2.2 GENEROVANÍ TEXTUR

Po vygenerování terénu je nutné vygenerovat texturu, která bude terén pokrývat a zlepší tak vizuální kvalitu zobrazení. Vzhledem k tomu, že tvorba textur není prováděna syntézou již připravených textur, je celé generování poměrně složité. Je založeno na procedurálních celulárních texturách a jejich skládání funkcí velmi podobnou Perlinovu šumu.

2.3 ALGORITMUS ZOBRAZENÍ

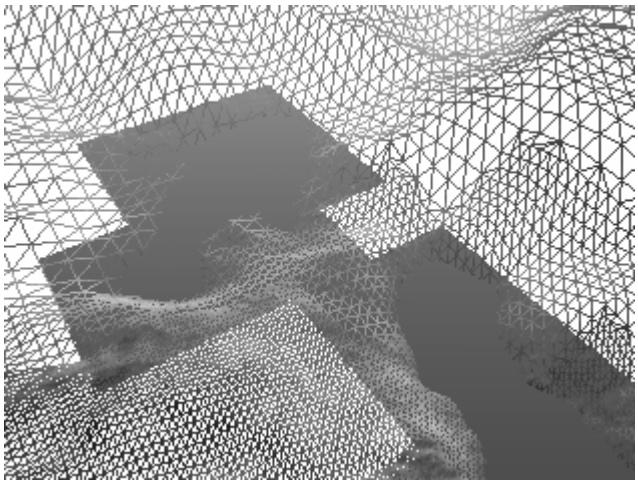
Vlastní zobrazení je založeno na dlaždicovém algoritmu (*tile-based terrain algorithm*). Vygenerovaný konečný terén je nejprve rozdělen na dlaždice, potom jsou každé dlaždici přiděleny odpovídající body tvořící terén a také příslušná část vygenerované textury. Dlaždicový algoritmus sám o sobě ale nekonečné terény zobrazovat neumí, proto byl tento algoritmus v rámci řešení rozšířen o tuto schopnost. Každá dlaždice obsahuje své jedinečné číslo a strukturu, která informuje o tom, které dlaždice jsou v její blízkosti a v jaké poloze jsou k této dlaždici. Velikost této struktury okolních dlaždic je volitelná a může se za běhu programu měnit, čímž lze dynamicky měnit rozsah viditelnost. Při zobrazení se nejprve zjistí pozice zadané pohledové kamery, která je ve světových souřadnicích. Tyto souřadnice se následně operací modulo převedou na odpovídající místo v konečném (vygenerovaném) terénu. Z těchto nových souřadnic je pak možné určit dlaždici, na které je kamera umístěna. Zároveň také můžeme zjistit, které dlaždice jsou v jejím okolí a jak jsou umístěny. Z těchto informací už můžeme provést zobrazení správně umístěných dlaždic.

Vlastní zobrazení je dále rozděleno na dvě podstatné části – určení viditelných částí terénu a určení úrovně detailů (*level of detail, LOD*) jednotlivých dlaždic.

Určení viditelných částí je řešeno pomocí metod *QuadTree* a *OctTree*. Každá dlaždice je tedy testována ve 2D průmětu a následně ve 3D. V případě úspěšného testu viditelnosti jsou testovány její části stejným způsobem.

Určování úrovně detailů je založeno na nejjednodušší technice – statické úrovně detailů závislé na vzdálenosti dlaždice od kamery. Úroveň detailů je určována vždy až pro nejmenší dlaždici.

Součástí zobrazení terénu je i zobrazení vodní hladiny. Tento způsob byl zaveden s ohledem na to, že pokud je někde zobrazen terén, pak zde nemůže být zobrazena vodní hladina a naopak. Po vykreslení terénu jsou tedy dostupné informace o tom, kde všude je potřeba vykreslit vodní hladinu a ta může být vykreslena velmi efektivně.



Obr. 2: *Efektivní vykreslení vodní hladiny*

2.4 INTERAKCE S TERENÉM

Hra TankGame má na terén dva základní požadavky. Terén musí být schopen velmi rychle zjistit výšku na libovolně zadaných souřadnicích a musí být také schopen velmi rychle určit průsečík mezi zadanou přímkou a povrchem.

Oba požadavky jsou s ohledem na rychlost provádění řešeny pomocí technik *QuadTree* a *OctTree* (pouze v případě zjišťování průsečíku s přímkou) podobně jako při zobrazování.

3 ZÁVĚR

Pro počítačovou hru je terén jednou z nejdůležitějších součástí. Proto bylo při řešení nutné docílit především jednoduchého a zároveň efektivního rozhraní. Zároveň bylo nutné zajistit možnost snadné budoucí náhrady některých algoritmů jinými – např. jiná metoda pro LOD.

Implementované řešení využívá moderních přístupů používaných v dnešních grafických aplikacích. Z nich nejdůležitější jsou dávkové zpracování a maximální přenesení zátěže z centrálního procesoru na grafický čip. Tím bylo dosaženo uspokojujících výsledků s ohledem na poměr mezi zatížením CPU a zatížení GPU.

LITERATURA

[1] Sánchez, D., Dalmau, C.: Core Techniques and Algorithms in Game Programming, New Riders, Publishing, Indianapolis, 2003

[2] Martz, P.: Generating Random Fractal Terrain, 1996 web: http://www.gameprogrammer.com/fractal.html, (únor 2006)