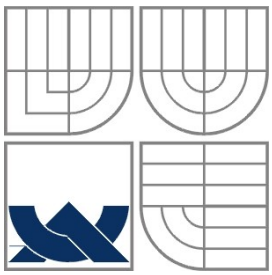
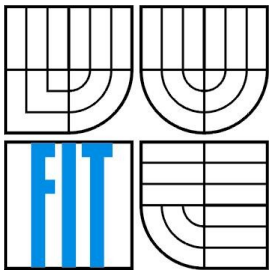


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REALISTICKÁ KRAJINA S VEGETACÍ

REALISTIC LANDSCAPE WITH VEGETATION

DIPLOMOVÁ PRÁCE

AUTOR PRÁCE — BC. JAN ZELENÝ
AUTHOR

VEDOUCÍ PRÁCE — ING. MICHAL HRADIŠ
SUPERVISOR

BRNO 2009

Realistická krajina s vegetací

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Hradiše
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Chtěl bych na tomto místě poděkovat mému vedoucímu za věcné rady, konstruktivní připomínky
a obětavou pomoc při konzultacích.

Abstrakt

V dnešní době již grafický výkon počítačů stačí na mnohem více než jen strohé interiéry a může nabídnout velmi realistické zobrazení krajiny a vegetace na ní. Spolu s tím se objevují čím dál sofistikovanější metody pro generování takové krajiny a simulaci ekosystému rostlin, které na ni žijí. Tato práce popisuje mnohé z algoritmů pro generování i metod pro interaktivní vykreslování krajiny a vegetace.

Klíčová slova

Realistická krajina, Vegetace, Procedurální generování, Vykreslování, Úroveň detailů, Hardwarový Geomorphing, Geo MipMap, Impostor, L-systém, Simulace ekosystému, Výpočet řek, Výpočet jezer

Abstract

There is enough rendering power to draw more than only simple indoor scenes today and it can produce very realistic images of landscape with vegetation. Moreover, there are new sophisticated methods for generating of such landscape and simulation of plants ecosystem. This text explains few algorithms for generating and methods for interactive rendering of landscape and vegetation.

Keywords

Realistic landscape, Vegetation, Procedural generation, Rendering, Level of details, Hardware Geomorphing, Geo MipMap, Impostor, L-system, Ecosystem simulation, River drainage, computation of Lakes

Citace

Jan Zelený: Realistická krajina s vegetací, diplomová práce, Brno, FIT VUT v Brně, 2009

Obsah

1 Úvod

Ve svých počátcích stačila počítačová grafika jen na omezené interiéry, později se ale čím dál častěji objevovaly rozsáhlé exteriéry spolu s mnoha přírodními útvary, zachovávajícími si dostatečnou míru detailů a pro přírodu tolik typickou a téměř nevyčerpatelnou variabilitu. S tím vzrůstaly i nároky na vytváření, uchovávání a zobrazování takové scény.

K řešení problému s výpočtem, paměťovou reprezentací a vykreslováním složitých scén existuje hned několik přístupů, které můžeme podle jejich přístupu rozdělit do 3 skupin: první skupinu tvoří metody zaměřené na vědecké využití, jakými jsou simulace eroze terénu, pravděpodobnost záplav apod., kde je větší důraz kladen na samotné generovaní a simulaci terénu. Jiné jsou využity v herním průmyslu např. v The Hunter [34] (obrázek 1.), ve vizualizačních nástrojích a také např. pro výcvik pilotů, kde je zase větší důraz kladen na samotné vykreslování. Třetí skupinu tvoří metody, zaměřující se na způsob uložení a efektivního načítání dat. Mnoho aplikací totiž nemá data uložena na stejném počítači a musí je streamovat či efektivně pracovat s omezenou pamětí, Google Earth [17].

Také zobrazování rostlin je v počítačové grafice věnována větší pozornost. Díky výkonům grafických karet je možná čím dál větší různorodost, a v některým případech dokonce generování rostlin za běhu aplikace pomocí programovatelného vykreslovacího řetězce, PipesGS [22]. S tím vzrůstají nároky na metody generování a efektivního vykreslování vegetace v počítačové grafice.

Obrázek 1: The Hunter, počítačová hra ve virtuálním prostředí zalesněné krajiny.

Tato práce se věnuje jak generování terénu a šíření rostlin na něm, tak i zobrazování realistického terénu a vegetace. Klade si za cíl navrhnout systém pro efektivní práci s terénem a popsat metody, jakými budou jednotlivé dílčí úkoly řešeny. Hlavní využití mého přístupu bude v herní aplikaci/simulátoru, proto jednotlivé generovací metody nebudou vybrány s ohledem na fyzikální korektnost, ale na rychlost a realistický vzhled.

V práci jsou popsány způsoby získávání základní výškové mapy terénu a její modifikace pomocí erozí. Je zde definována a popsána nová metoda, sloužící k výpočtu řek a jezer na terénu. Poté se práce věnuje rozmístění rostlin a simulaci jejich ekosystému (tedy růstu, šíření, uhynutí v důsledku nepříznivých lokálních podmínek, apod). Následně je popsána metoda generování těl rostlin a nakonec i metody sloužící k efektivnímu vykreslení všech předchozích výsledků, jakými jsou výšková mapa terénu, spolu s jezery a řekami, vegetace, definované rozmístěním jednotlivých rostlin a modely jejich těl.

Všechny metody, které jsou zde zmíněny, zachovávají základní požadavky na systém schopný výpočtu a vykreslení terénu s vegetací. Tyto pravidla popsali autoři práce Realistic Modeling and Rendering of Plant Ecosystems [27] a uvedu zde jejich výčet a volný překlad:

- **Otevřená architektura systému** - Jasnou specifikací formátu vstupů a výstupů každé jednotlivé fáze generovacího řetězce vytvoříme podmínky pro spolupráci nezávisle vyvíjených modulů. Otevřená architektura umožňuje větší komplexnost modelovaných scén díky množství různých modulů a poskytuje příležitosti k experimentování s odlišnými přístupy a algoritmy.
- **Procedurální modely** - Procedurální modely jsou často charakterizovány výrazným znásobením dat, což znamená, že dokáží generovat komplexní geometrické struktury z relativně malého množství vstupních dat. Je dokonce možné využít tento fenomén ve všech fázích generování terénu i rostlin.
- **Instancing** - Jako hlavní formu redukování geometrické reprezentace je vhodné použít instancing. Ke zefektivnění se jednotlivé komponenty (rostliny a jejich části) shlukují v jejich parametrickém prostoru a zprůměrovávají v každém takovém shluku na jeden reprezentující objekt.
- **Efektivní vykreslování** - K lepší práci s pamětí a efektivnímu vykreslování je vhodné scénu rozdělit na několik podscén a při vykreslení je opět spojit do výsledného obrázku.

Tato pravidla byla vytvořena pro ray-tracing metodu vykreslování. Můj přístup je zaměřen na vykreslování v real-time aplikacích. Oba přístupy jsou přesto velmi podobné a v zásadě se bude tato práce uvedených pravidel držet.

2 Generování terénu

Prvním krokem, vyžadovaným jako vstup pro ostatní metody, je vygenerování terénu. Jedná se o metodu, při které výpočtem získáme výškovou mapu zvoleného dílu krajiny (výškovou mapou zde rozumíme uloženou nadmořskou výšku každého bodu mřížky, rovnoměrně rozprostřené po zvoleném dílu mapy).

V následujících kapitolách jsou popsány způsoby, jak výpočtem získat výškovou mapu, která bude popisovat podobné krajinné útvary, jaké známe z přírody. Poté následuje popis erozí, přizpůsobujících výsledek ještě více geologickým a hydrologickým pochodům v přírodě.

Na závěr je uveden popis nové metody pro výpočet řek a jezer na získaném terénu a také doplňujících informací pro další metody (především terén doplníme informacemi o lokálních vlhkostech, sklonu svahů apod. Ty jsou vstupem nejen pro metodu simulace ekosystému rostlin (kde se na jejich základě rozhoduje o přežití či třeba rozmnožení rostliny), ale i například pro metodu vykreslování terénu (zde lokální atributy udávají použitou texturu, jakou je skála na prudkých svazích, či tráva v místech větší vlhkosti).

2.1 Způsoby získání výškové mapy

Existují v podstatě tři základní způsoby získání výškové mapy. Je možné ji získat jako skutečná data (např. z družic, které z oběžné dráhy mapují zemský povrch). Výhodou takového přístupu je reálnost a realističnost dat, která se ale na druhou stranu musejí uchovávat uložena na disku a ani jejich získávání není obecně snadné.

Oproti tomu výšková mapa získaná od uživatele za použití interaktivního grafického editoru může být vytvořena až za běhu aplikace, její realističnost je však silně omezena představivostí a schopnostmi uživatele a veskrze nedosahuje dobrých výsledků.

Třetím a pro účely této práce nejvhodnějším způsobem je generování. Nabízí v podstatě nekonečnou variabilitu s možností vytvoření výškových dat až za běhu aplikace a s přijatelnou realističností. Také zachovává princip procedurálního modelu (jak byl popsán v kapitole 1. Úvod). Nevýhodou při pokročilejších metodách je velké zatížení procesoru a delší výpočetní doba potřebná k získání výsledných dat. Nyní následuje popis algoritmů, které jsou pro generování výškové mapy použity.

Obrázek 2: Fault formation: a) schéma rozdělení plochy a přidání malého čísla vždy k jedné polovině, b) metoda po jedné iteraci, c) metoda po 128 iteracích.

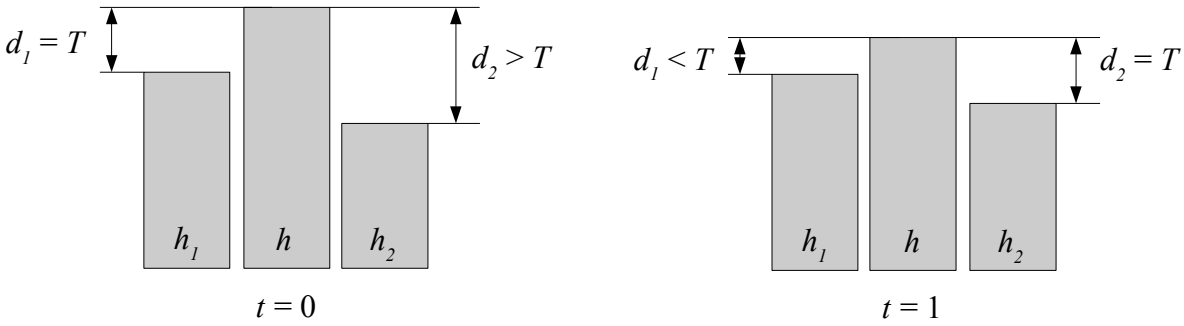
2.2 Způsoby generování výškové mapy

K vytváření výškové mapy je nejjednodušší použít šum Perlin noise [21], v mnoha textech je také popsáno generování s fraktály A Fractal Model of Mountains with Rivers [2], The Synthesis and Rendering of Eroded Fractal Terrains [36]. Přesto, že se tyto metody svojí jednoduchostí jen stěží blíží reálným výškovým mapám, jsou vhodné jako doplňující úpravy výškové mapy, které zvyšují variabilitu a komplexnost terénu, a proto (zvláště perlin noise) jsou v práci použity.

Vhodnější způsob generování nabízí Realtime Procedural Terrain Generation [28]. Zavádí tzv. Diamong sqare algoritmus, který je v podstatě Midpoint displacement algoritmus a můžeme se o něm mnohé dozvědět i z práce Random Midpoint Displacement Method [25]. Navíc výše zmíněná práce Realtime Procedural Terrain Generation [28] zavádí několik úprav, založených mimo jiné na Voronoi diagramu, a rozšiřuje tak množinu filtrů pro úpravu terénu.

V knihách Game Programming Gems [15] je popsán výkonný a svými výsledky dobrý algoritmus Fault Formation, který spočívá v dělení plochy na dvě poloroviny a v přičtení malé hodnoty vždy k jedné z nich. Přehledně to zobrazuje obrázek 2.

V mém řešení se ještě tato hodnota postupně snižuje a tím docílíme výraznějších terénních uskupení (souvislého hřebenu hor). Tento algoritmus odstraňuje problémy např. u výše zmíněného Perlin noise, kde se vrcholy hor a dna údolí (tedy lokální maxima a minima) vyskytovaly při špatné volbě parametrů na přímkách osově rovnoběžných. U metody Fault formation se s tím nesetkáme, zato je zde nutnost výslednou výškovou mapu nějakým vhodným filtrem zahladit, neboť výstupem tohoto algoritmu je výšková mapa obsahující jen množství geologických zlomů, podle kterých se i metoda jmenuje. V knize Game Programming Gems [15] je na zahlazení použit FIR filtr, který je sice jednoduchý, ale vhodnější je použít nějakou sofistikovanější metodu z kategorie Thermal erosion, volně přeloženo jako zvětrávání, která je v následující přiblížena podrobněji.



Obrázek 3: Jednoduchý příklad sesuvu půdy: d_2 je větší než mezní hodnota a proto je přesunut materiál z *h* do h_2 až je d_2 zpět alespoň na mezní hodnotě. Zde je konstanta *c = 1*.

2.3 Zvětrávání

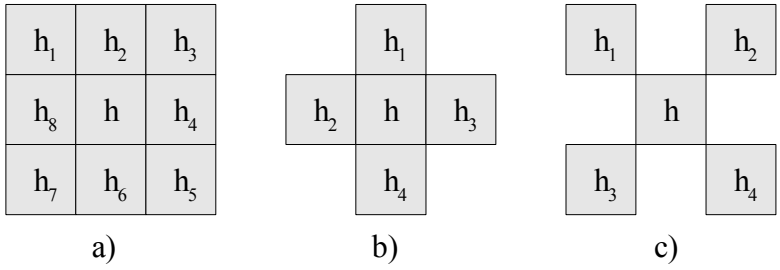
První eroze, kterou si popíšeme je v anglické literatuře označovaná jako thermal, do češtiny si ji můžeme volně přeložit jako sesuv půdy či zvětrávání. V nejjednodušším případě se nahrazuje prostým FIR filtrem. Ten sice postihuje nejdůležitější vlastnosti thermal erosion, jakými jsou snížení sklonu prudkých svahů a zaplnění dna ostrých údolí materiálem, přesto svojí jednoduchostí neposkytuje téměř žádné možnosti ke specifikaci lokálních vlastností terénu a navíc mimo zahlazení dna údolí snižuje ostrost i hřebenů hor.

Sofistikovanější metodu, která daleko přesněji vystihuje podstatu sesuvů půdy v přírodě, popisuje dílo Realtime Procedural Terrain Generation [28], princip tohoto algoritmu je následující: Výšková mapa se projde a pro každý bod se provede nejprve zjištění množství materiálu, kterým je již překročen jistý limit maximálního sklonu svahu. Tento materiál je posunut po svahu tak, aby sklon kopce dosáhl hraniční hodnoty. Vzorec pro tuto událost je $d_i = h - h_i$, kde *h* je výška aktuálního bodu a h_i je výška sousedního (viz vzorec 1.), d_i je rozdíl výšek (u nižšího souseda je tato hodnota pozitivní).

$$h_i = \begin{cases} d_i > T: & h_i + c(d_i - T) \\ d_i \leq T: & h_i \end{cases} \qquad \textbf{(1)}$$

Zde je *T* právě mezní hodnota sklonu svahu (rozdílu výšek sousedních bodů), po které již dochází k sesuvu. Výsledná výška souseda je v případě překročení sklonu svahu d_i zvětšena o část nebo celé množství materiálu, kterým byl sklon překročen. Právě *c* udává, zda se přesune celá část, nebo jen něco. Můžeme si tuto situaci prohlédnout na obrázku 3.

Tento jednoduchý příklad předpokládá pouze jednoho souseda. V případě 2D výškové mapy musíme zvolit sousedů více a máme hned několik možností. V práci Realtime Procedural Terrain Generation [28] jsou popsány 3 verze sousedností. Jednou z nich je sousednost Moorových



Obrázek 4: Druhy sousedností u celulárních automatů: a) Moore, b) Von Neumann a c) pootočená Von Neumann

celulárních automatů. Rychlejších výsledků je možné docílit sousedností Von Neumanových celulárních automatů a pomocí pootočené sousednosti těchto automatů se docílí ještě lepších výstupů, viz obrázek 4.

V práci je dále popsán způsob, jak vypočítat množství přesunutého materiálu. Zavádějí vztah (vzorec 2), kde d_{max} je největší d_i a d_{tot} je suma všech d_i větších něž úhel *T*. Každému sousedu se distribuuje takové množství materiálu, které odpovídá d_i / d_{total}.

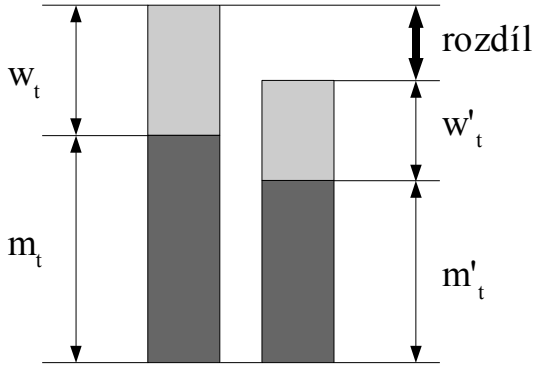
$$h_i = h_i + c(d_{max} - T) \times \frac{d_i}{d_{total}} \qquad \textbf{(2)}$$

Pro konstantu *c* je třeba zvolit přiměřenou hodnotu. Nejlepší je 0.5, neboť vyšší hodnota by způsobila nestabilitu eroze a nižší by zase generovala uspokojivý výstup po příliš mnoha krocích. Pro volbu úhlu T doporučují autoři $T = 4/N$.

V mém řešení jsem zvolil hodnotu proměnlivou, pro lepší popis lokálních vlastností. Hodnota se pohybuje v rozsahu *<0, 0.1>* podle normalizované výšky terénu. Tím se ve vyšších polohách umožní ostřejší svahy a nížiny se dostatečně zaoblí. Také se tím částečně stabilizuje systém a k ustálení potřebujeme méně kroků. Mimo to se také lépe zachovají ostré hřebeny skutečně vysokých pohoří.

2.4 Vodní eroze

Druhou erozí, která je an terén použita, je eroze vodní. Algoritmy, které ji popisují, jsou o něco složitější, ale její aplikace je pro realistický vzhled výsledného terénu neméně důležitá. Opět existuje mnoho prací, které se jí zabývají, jmenujme alespoň Terrain Simulation Using a Model of Stream Erosion [31], která se zabývá více statistickým rozmístěním řek a jejich povodí na neerodované mapě, než simulací vody na již modelovaném terénu. Novějším a vhodnějším přístupem je Visual



Obrázek 5: Značení materiálů. Levý sloupec představuje zkoumaný bod a pravý jeho souseda.

Simulation of Hydraulic Erosion [39] a pozdější práce Terrain Simulation Using a Model of Stream Erosion [31], kterou jsem v návrhu použil a bude v této práci popsána. Nakonec ještě zmíním velmi perspektivní práci Real-Time Erosion Using Shallow Water Simulation [26], používající pro výpočet povrchové vody 2D zjednodušení Navier-Stokes rovnic, které se podle slov autorů osvědčily nejen pro rychlost výpočtu a vhodnost použití v real-time aplikacích, ale i pro vizuální kvalitu výsledků.

Zmíněná metoda Terrain Simulation Using a Model of Stream Erosion [31] dekomponovala problém vodní eroze do čtyř oblastí. Jsou to:

- Objevení se nové vody (dešťové srážky apod.)
- Eroze podkladu a odnášení materiálu
- Přesouvání vody a suspenze v ní po svahu do údolí
- Sáknutí či vypařování vody a ukládání unášeného materiálu na nové místo

Autoři zavádějí následující sémantiku použitých symbolů: Nechť m_t je výška materiálu, w_t je výška vodního sloupce v daném místě terénu a s_t je množství materiálu rozptýleného ve vodě (suspenze) v čase *t*. Sémantika je zobrazena na obrázku 5. V následujícím časovém okamžiku budou hodnoty m_{t+1} , w_{t+1} a s_{t+1} . Koeficient $t+1$ znamená změnu času o Δt , tedy $t+\Delta t$. Pro jednoduchost předpokládejme, že bod má pouze jednoho souseda, jeho hodnoty označme *m′*, *w′* a *s′*. Jak řešit případ 2D výškové mapy je popsáno hned poté.

Zdroj vody

Prvním krokem je přidání vody na terén. Může to být buď bodový zdroj (pramen) nebo plošný (déšť). Pro jednoduchost předpokládejme vodní přírůstek na celé mapě o K_r , tedy že $w_{t+1}=w_t+K_r$. Tento koeficient volí autoři konstantní, ale pro lepší simulaci je možné doplnit funkci času a místa, tedy $K_r(t, P)$, kde *P* je místo na mapě.

Vypařování vody

Míra vypařené vody závisí na teplotě a objemu. Vzhledem k tomu, že používáme výškovou mapu a máme tedy konstantní plochy, bereme v potaz pouze výšku vodního sloupce. Autoři dále předpokládají i konstantní teplotu. Pro vypaření poté zavádějí vztah 3.

$$\frac{dw}{dt} = -K_e * w \qquad (3)$$

kde K_e je koeficient odpaření či sáknutí vody do země. Míra množství vody tedy klesá exponenciálně podle vztahu $w_t = w_0 * e^{-K_e * t}$. Pro úsporu výpočetního času je zavedena mezní hodnota *T*, pod kterou když výška vodního sloupce klesne, je považována za nulovou a dále se s ní nepočítá.

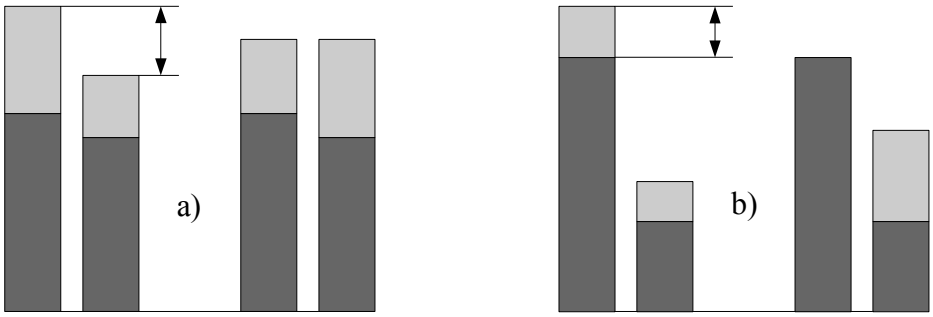
Ukládání sedimentu

V přírodě se při pomalu tekoucí vodě vyskytuje více sedimentů (většinou na okrajích řek, v zákrytu překážek v řece apod.). To je způsobeno tím, že sediment v pomalu tekoucí vodě tolik nevíří a klesá ke dnu, kde je zachycován a ukládán.

Zmíněná práce Terrain Simulation Using a Model of Stream Erosion [31] k tomuto poznatku přidává ještě maximální míru S_t sedimentu v daném množství vody w_t danou vztahem $S_t = K_s * w_t$, kde K_s je koeficient udávající kolik kilogramů sedimentu může být maximálně v jednom litru vody. Pokud při odpařování poklesne objem vody natolik, že míra sedimentu v ní přesáhne danou maximální hodnotu, usadí se přebývající část materiálu na dno. Zavádějí proto vztah $m_{t+1} = m_t + (s_t - S_t)$, kde při překročení množství sedimentu s_t přes mez S_t dojde k sedimentaci.

Transport vody se sedimentem

V přírodě se voda snaží dosáhnout určité rovnováhy a to má za následek vyrovnávání hladiny a stékání vody do údolí. Tato práce se nesnaží řešit simulaci proudění vody, spolu s její setrvačností a vnitřní dynamikou. Bude pouze předpokládat pomalu tekoucí vodu a výše zmíněnou notaci, kde se bere v potaz pouze jeden sousední sloupec. K notaci se přidává další symbol *h*, který je součtem výšky materiálu a vodního sloupce na něm. $h=m+w$. Pro sousední sloupec je to $h'=m'+w'$. Pokud je rozdíl $h-h'$ menší než nula, znamená to, že aktuální bod má hladinu níže a nelze odebírat nic. Pokud je naopak rozdíl kladné číslo, jeho hodnota říká, o kolik je jeho hladina výše než u souseda. Poté obě hladiny srovnáme přesunem polovičního množství k sousedovi, maximálně však výšku vodního sloupce na aktuálním místě.



Obrázek 6: Dva důležité případy přesunu vody. a) vyrovnají se hladiny, b) veškerá voda se přemístí.

V případě 2D výškové mapy pak algoritmus upravíme. Zmíněná práce Terrain Simulation Using a Model of Stream Erosion [31] přináší řešení v podobě vztahu 4:

$$\Delta w_i = \Delta w \frac{h_i}{sum} \qquad (4)$$

kde Δw_i je množství vody přesunuté k *i*-tému sousedovi a h_i je rozdíl souseda (bereme v potaz pouze nižší sousední body), *sum* je suma takových rozdílů.

Nakonec práce zmiňuje několik dalších rozšíření, jako je řešení nejen pohybu vody, ale i sedimentu v ní. Tedy pokud jeden vodní sloupec obsahuje velkou hustotu sedimentu a sousední (byť stejné výšky hladiny) nízkou, pak se míra sedimentů v obou rovnoměrně rozprostře.

Eroze materiálu

Velmi důležitou součástí vodní eroze je samotná eroze materiálu, tedy výpočet množství materiálu, které proudící voda mění ve vodní suspenzi, jež je pak odnášena do údolí.

V práci Terrain Simulation Using a Model of Stream Erosion [31] není eroze materiálu vůbec řešena. Algoritmus nabízí práce Fun with erosion [14], kde je i příklad implementace. Přesto ani tato práce neobsahuje metodu eroze založenou na silách proudící vody a zavádí pouze aproximaci takových vlivů. Pro účel této práce ale postačuje a je použita i v demonstrační aplikaci.

2.5 Získání doplňujících lokálních atributů výškové mapy

Samotná výšková mapa často nepostačuje jako vstup pro další metody, jako jsou generování a simulace šíření rostlin a nanesení textur na terén, a proto zde bude popsáno, jak ji doplnit o další lokální atributy. Je třeba získat informace o sklonu svahu, neboli první derivaci výškové mapy (později je tento údaj použit pro nanesení textu skal v místech velmi prudkých svahů apod.). Další

možný doplňující údaj je druhá derivace výškové mapy, neboli do jaké míry je místo na mapě konkávní či konvexní (ve vydutých místech mapy se např. lépe uchytí semena rostlin atd.). K metodě šíření a simulaci rostlin je ještě třeba získat vlhkost půdy a k tomu účelu poslouží metoda počítání plochy povodí spadajících do jednotlivých míst mapy. Tato metoda je popsána v následující kapitole 2.6 - Řeky a jezera a vhodnou aproximací vlhkosti terénu v konkrétním místě je údaj právě o velikosti povodí, spadajícího do daného bodu.

2.6 Řeky a jezera

Dalším krokem, který výrazně zvýší míru realistického vzhledu terénu je přidání řek a jezer. Velká část současných terénů v počítačové grafice má řeky řešeny ručně (což znemožňuje náhodné generování terénu), nebo nemají řeky a jezera vůbec. Chceme-li tedy počítat řeky algoritmicky, existuje jen několik přístupů a metod, které ale minimálně berou v potaz již připravenou výškovou mapu. Jedním z nich je obsahem práce Adding Realistic Rivers to Random Terrain [4]. Zde autor popisuje metodu, při které je náhodně terén „přetnutý“ křivkou řeky. Ta je poté mírně upravena vlastnostmi terénu pro lepší zakřivení a tím i reálnější vzhled. Na závěr je terén okolo řeky snížen, aby vizuálně odpovídal korytu řeky.

Dalším řešení bývá použití fraktálů: A Fractal Model of Mountains with Rivers [2]. Zmíněná práce zavádí generování jak výškové mapy, tak i tvaru řek. Nesnaží se tedy simulovat řeku na terénu, ale rekurzivním algoritmem vypočítat trajektorii řeky tak, aby se nikde nepřetínala a dalo se generovat i více úrovní detailů. Přináší také možnost výpočtu řeky i s přítoky. Okolní terén se opět generuje až po vytvoření řek.

Zcela jiný přístup nabízí práce Terrain simulation using a model of stream erosion [31]. Ta vychází ze statistických poznatků vysledovaných v přírodě a řečiště generuje podle těchto údajů. Vstupem je tedy např. úhel, kterým se odděluje přítok, informace, jak časté jsou přítoky a kolik mívají rozlohu povodí atp. Výstupem této práce je realisticky vypadající systém řek a soutoků. Ani tento přístup ale nepočítá s již připravenou výškovou mapou a terén se tvoří až na základě řek.

Hledáme-li větší míru vlivu již generovaného terénu, musíme nahlédnout do metod pro vodní erozi a pro zjištění toků řek tyto algoritmy přizpůsobit. Je tedy možné např. zjistit nejsilnější proudy unášející materiál do údolí a vést jejich středem křivku řeky. Z této myšlenky vychází nová metoda výpočtu řek podle povodí, kterou zde popíši.

Metoda výpočtu řek podle povodí

Nová metoda výpočtu řek a jezer podle velikosti povodí, kterou zde definuji, sestává ze dvou hlavních kroků. Prvním je výpočet tzv. "Primárních řek" tedy řek, které pramení v horách a jejich

trajektorie končí dnem údolí. Výstup této metody je pak vstupem druhé fáze, která počítá jezera a "Sekundární řeky". Těmi nechť jsou takové řeky, které vznikají jako odtokové z jezer a plní jiná jezera.

Výpočet primárních řek

Metoda výpočtu primárních řek nejprve vyčíslí velikosti povodí pro každý bod výškové mapy. Na začátku uloží do každého bodu *a* číselnou konstantu *c*, která udává rozlohu právě jednoho dílku mapy. Tzn. rozloha *S(a)* každého bodu a je na počátku *S(a)=c*. Poté prochází několikrát celou mapu a aktualizuje hodnotu v každém bodu *a* následujícím způsobem: Pro každé *a* ověří všechny jeho sousedy a_i a sčítá uloženou rozlohu povodí $S(a_i)$ každého takového souseda, jehož vlastním nejnižším sousedem je právě *a*, tedy z něhož stéká voda právě do zkoumaného bodu *a*. Na závěr k této sumě přičte opět konstantu *c* a pokud je výsledkem větší rozloha, než je uložena v daném bodě *a*, uloží do něho tento nový údaj. Algoritmus pro tento výpočet je popsán v Algoritmu 1. a obrázek 6 ilustruje několik iterací pro jeden bod.

Na získanou mapu velikostí povodí aplikujeme Cannyho hranový detektor, popsaný v práci A computational approach to edge detection [1], který nalezne trajektorie řek. Na závěr je třeba aplikovat filtr, který obnoví soutoky řek, protože Cannyho hranový detektor ve své podstatě negeneruje větvící se křivky. Tento filtr může být například kontrola všech bodů řek a zjištění jejich nejnižšího souseda a označení všech takových také za bod řeky.

Pro každý bod *a* mapy {

S(a) = c

}

Dělej (dokud *změna*) {

změna = false;

Pro každý bod *a* mapy {

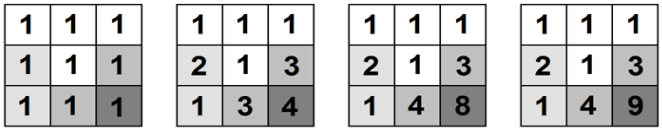
$$S(a) = argmax\left(S(a), c + \sum_{pro\ každého\ souseda\ i} \begin{cases} a\ je\ nejnižší\ soused\ pro\ a_i: & S(a_i) \\ jinak: & 0 \end{cases} \right)$$

změna = false;

}

}

Algoritmus 1: Výpočet povodí pro každý bod.



Obrázek 6: Několik iterací algoritmu výpočtu povodí řek pro jeden bod mapy. *c=1*.

Výstupem z první fáze (výpočtu primárních řek) jsou trajektorie řek. Pokud chceme získat průtok v konkrétním místě řeky, můžeme jako aproximaci použít právě údaj o velikosti povodí řeky. Rychlost řeky je zase úměrná sklonu kopce ve zkoumaném místě (ten již byl získán z metody popsané v kapitole 2.5 - Získání doplňujících lokálních atributů výškové mapy). Obsah průřezu řeky je pak výsledkem podělení průtoku řeky a její rychlosti.

Výpočet jezer a sekundárních řek

Vstupem tohoto kroku je seznam bodů, tvořících dna údolí, a velikostí povodí v těchto bodech. Velikost povodí (při zanedbání odpařování a sáknutí do půdy) násobená průměrnými dešťovými srážkami je přímo úměrná k objemu vody, který musíme do údolí umístit. $V(a) = S(a) * K_r$, kde *S(a)* je velikost povodí nejnižšího bodu údolí *a* a K_r je konstanta udávající dešťové srážky. K umístění daného objemu vody do údolí slouží algoritmus plnění údolí vodou, který (opět iterativně) přidává hraniční body jezera do seřazeného seznamu (začíná se dnem údolí a jeho sousedními body). Vždy vezme ze seznamu nejnižší nezatopený hraniční bod a zvýší hladinu jezera do jeho výšky (pokud objem vody již postačil, hladina se nemusí zvýšit až k tomuto bodu, v takovém případě algoritmus končí). Poté zatopíme tento bod a opět přidáme jeho sousední body.

Kromě případu, kdy objem jezera již postačuje k umístění požadovaného objemu vody do údolí, může nastat ještě jedna ze dvou událostí. Buď jezero přeteče a pak se tvoří sekundární řeka, která přebytečný objem vody distribuuje do jiného dna údolí, nebo jezero přeteče do jiného jezera a pak se obě jezera spojí a dále se s nimi počítá jako s jedním jediným. Celý algoritmus je zapsán v Algoritmus 2. a plnění jezer je znázorněno na obrázku 7.

Obrázek 7: Plnění jezer vodou. a) Dno údolí, b) Plnění jezera,
c) Přetečení jezera a tvorba sekundární řeky, d) Spojení dvou sousedních jezer

```
Pro každé dno údolí a {
    V (a) = S(a) * Kr
    Přidej do seřazeného seznamu hranice (a) všechny sousedy ai bodu a
    plocha_jezera (a) = c
}
Dělej (dokud změna)  {
    změna = false;
    Pro každé dno údolí a {
        konec = false
        Dělej (dokud not konec)  {
            nová_výška_hladiny = nejnižší nezaplavený bod hranice (a)
            Pokud ( jezero(a) s hladinou ve výšce nová_výška_hladiny  má už
                objem > V(a) ) {
                hladina jezera(a) =  zvyš výšku hladiny pro zbylý objem;
                konec = true
            }
            Jinak pokud (nová_výška_hladiny < hladina jezera(a) ) {
                Sleduj sekundární řeku do nového údolí b
                Přidej přebytečný objem vody do V(b)
                Pokud (b přímo sousedí s a) {
                    Spoj jezera (a, b)
                }
                konec = true
                změna = true
            }
            Zvyš hladinu jezera do nová_výška_hladiny
            plocha_jezera (a) = plocha_jezera (a) + c
            Zaplav nejnižší bod hranice(a) a přidej jeho sousední body do hranice(a)
        }
    }
}
```

Algoritmus 2: Plnění jezer vodou a tvorba sekundárních řek

3 Simulace šíření rostlin

Šíření rostlin po terénu můžeme rozdělit do dvou hlavních úloh. Prvním krokem je výchozí umístění rostlin na terén a získání parametrů jednotlivých rostlinných druhů a terénu. Druhou fází je pak simulace života takových rostlin, jejich růst, šíření a odumírání rostlin, které byly zastíněny vitálnější vegetací, které jsou přestárlé, nebo které rostou na nevhodných místech. Nyní budou popsány podrobně jednotlivé fáze šíření rostlin. Výstupem simulace šíření rostlin je sada informací o druhu a pozici každé jednotlivé rostliny spolu s její velikostí a vitálností.

3.1 Parametry pro simulaci

Seznam vlastností rostlin, které bude následující metoda simulovat, je dán především dostupnými parametry terénu (známe-li na terénu vlhkost, můžeme mezi vlastnosti simulovaných rostlin zařadit vlhkomilnost).

Inspirací pro tuto metodu byl popis parametrů v práci Realistic Modeling and Rendering of Plant Ecosystems [27], kde autoři pro parametry rostlin zavádějí dvě skupiny: Parametry daného botanického druhu a parametry konkrétní rostliny. První skupinu (tedy rostlinný druh) tvoří tyto vlastnosti:

- počet rostlin přidaných do simulace za každý simulovaný krok
- maximální velikost rostlin
- průměrná rychlost růstu
- pravděpodobnost přežití dominance (můžeme chápat jako zastínění) jinou rostlinou
- preference pro vlhká nebo suchá místa.

Druhou skupinou, tedy vlastnosti pro konkrétní rostlinu jsou:

- rostlinný druh
- velikost
- jakási vitalita (na jak vhodném roste místě apod.)

Konkrétní metoda v této práci přidává ještě několik dalších parametrů, daných především získanými doplňujícími lokálními vlastnostmi terénu. Je odstraněn počet rostlin přidaný do simulace každým krokem (tento parametr je zavádějící a záleží na velikosti terénu apod.). Namísto toho jsou rostlinné druhy popsány šancí vysemenit a vzdáleností, do jakých se v průměru semena dostanou (to mimo jiné podporuje shlukování rostlin). Také byly pro větší možnost experimentování doplněny parametry jako relativní míra stínění rostliny a také její tolerance na stínění ostatních, svah kopce,

na jakém se ještě udrží, střední nadmořská výška a rozptyl a také pravděpodobnost uvadnutí přestárlých rostlin.

3.2 Metody rozmístění rostlin

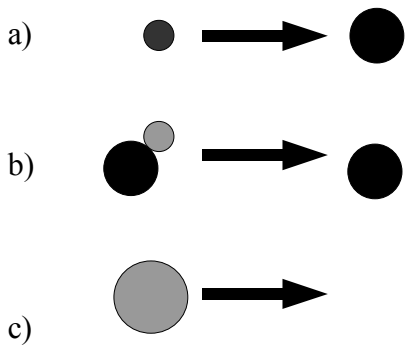
V práci Realistic Modeling and Rendering of Plant Ecosystems [27] autoři popisují dva způsoby, jak počáteční polohu rostlin pro simulaci určit. Je mezi nimi explicitní určení uživatelem a procedurální umístění algoritmem.

Co se týče prvního způsobu, dále ho dělí na dva rozdílné přístupy. Uživatel může pozice rostlin určit sám a to konkrétně pro každou rostlinu, nebo může interaktivním grafickým editorem určit hustotu každého rostlinného druhu na terénu (černou barvou místa, kde se rostlina nevyskytuje a světlejší až nakonec bílá místa s její největší hustotou, nebo načtením předem vytvořeného obrázku se stupni šedi). Výsledné konkrétní pozice rostlin jsou pak vypočítány programem, např. half-tone algoritmem, zmámým z počítačové grafiky. Autoři použili Floyd-Steinberg, popsaný blíže v práci An Adaptive Algorithm for Spatial Grey Scale [7]. Navíc uvádějí možnost využití explicitního rozmístění již jako výstup této fáze generování a jako vstup pro vykreslování.

3.3 Simulace ekosystému

Samotnou simulací ekosystému se zabývá několik prací, které ale vesměs používají stejný přístup (tzv. self-thinning), jmenujme za všechny Realistic modeling and rendering of plant ecosystems [27], kde autoři zavádějí model založený na individuálních rostlinách, které jsou reprezentovány pozicí rostliny a kruhovým okolím kolem ní, tzv. sféře vlivu. V práci s úspěchem využívají Open L-Systém schopný větvení struktur. Rostliny jsou tedy organizovány do stromových struktur, kde nejvýše je rodičovská rostlina a jednotlivé větve struktury představují potomky. K simulaci soupeření rostlin použili v angl. literatuře nazývaný self-thinning (volně přeloženo jako samo sebe regulující) model. Podobný přístup má i práce Generating Spatial Distributions for Multilevel Models of Plant Communities [16], která obsahuje i příklady L-systémů pro šíření rostlin a detailně do nich čtenáře zasvěcuje. Jistým rozšířením se pak dá chápat práce A stable modeling of large plant ecosystems [3], kde se autor podrobně zabývá stabilitou simulovaného systému (zamezuje jedné dominantnější rostlině, aby zaplavila celý terén).

Samotný self-thinning algoritmus je použit i v této práci a jeho popis je následující: Nejprve jsou rozmístěny rostliny metodami, které jsou popsány v předchozí kapitole 3.2 - Metody rozmístění rostlin. Každá je reprezentována polohou a poloměrem sféry vlivu. Ten se volí náhodně z daného rozsahu. Každým iteračním krokem rostliny povyrostou a zároveň s určitou pravděpodobností vytvoří



Obrázek 8: Základní pravidla pro simulaci ekosystému se self-thinning a) pravidlo pro růst rostliny, b) pokud se sféry vlivu protnou, přežije jen dominantní rostlina a c) přestárlá rostlina uvadá.

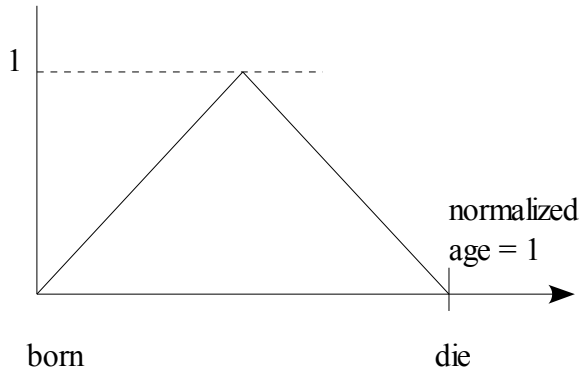
novou rostlinu stejného druhu a nulového poloměru na pozici dané náhodným nevelkým vektorem od rodičovské rostliny. Tím je docíleno shlukování rostlin podobného druhu. Dále platí pravidlo, že pokud se dvě sféry vlivu překrývají, menší rostlina hyne a je ze systému odstraněna. Dále rostliny, které dosáhnou jistého stáří (poloměru sféry vlivu), jsou označeny za přestárlé a opět hynou za současného odstranění ze simulace. Na obrázku 8 můžeme vidět základní pravidla této metody.

V systému tak na počátku bývá velmi mnoho rostlin (v závislosti na jejich rozmístění a hustotě) a po čase se jejich počet ustálí na nějaké hodnotě. Ta je mimo jiné dána plochami sfér vlivu (čím větší plochy, tím méně je v simulaci rostlin a naopak) a tím je dána tzv. self-thinning curve, neboli křivka samo-regulace, podrobně popsaná ve jmenované práci Generating Spatial Distributions for Multilevel Models of Plant Communities [16].

3.4 Stabilita systému

Popsaná metoda v kapitole 3.3 - Simulace ekosystému produkuje výsledky, které velmi citlivě závisí na vyvážení šancí pro jednotlivé rostlinné druhy. Při neuváženém volení těchto hodnot se tak může snadno stát, že jedna dominantní rostlina se po několika krocích simulace rozšíří po celé mapě na úkor ostatních druhů. Tímto nežádoucím jevem se zabývá práce A stable modeling of large plant ecosystems [3], kde autor zavádí upravené parametry dominance dvou překrývajících se sfér vlivu. Nevyhrává rostlina s největším věkem, ale ta s věkem kolem střední hodnoty. Rostliny jsou tedy penalizovány nejen pro svůj malý věk, ale i za stáří. Autor zavádí jakousi vitálnost *v(t)* (viz vzorec 5.)

$$v(t)=\begin{cases} \overline{age} & if\ age<\frac{1}{2} \\ 1-\overline{age} & otherwise \end{cases} \qquad \textbf{(5)}$$



Obrázek 9: Funkce vitálnosti rostliny podle normalizovaného věku.

$$\overline{age} = \frac{age}{die - born} \qquad (6)$$

kde $\overline{age}$ je normalizovaný věk vypočítaný z aktuálního věku *age*, pomocí vzorce 6, tedy podle postupu od narození k smrti. Zmíněný vztah tedy vede ke grafu vitálnosti rostliny zobrazenému na obrázku 9.

Po vyřešení problému s dominancí jednoho druhu zavádí penalizaci rostliny za přílišnou hustotu jejího druhu v okolí. Autor to ospravedlňuje tím, že v přírodě takové přemnožení jednoho druhu vyčerpá z půdy konkrétní živiny, které daný druh potřebuje a umožní se tím přežití ostatních typů vegetace. Dále zavádí vliv samotného prostředí jako další způsob, jak zabránit dominanci jedné rostliny. Například jeden druh je spíše vlhkomilný a pokud bude růst na sušším místě, bude mít opět postih při soutěži s jinou rostlinou. Přístup s vitálností a vlivem prostředí je použit i v této práci.

4 Modelování těl rostlin

V aplikacích zobrazujících terén s vegetací vzniká potřeba 3D modelů těl rostlin jako vstupu pro metodu vykreslování. Rostliny jsou svojí podstatou velmi složité 3D modely a je potřeba nejen jejich velké míry detailů, ale i dostatečné variability (v přírodě se od jednoho druhu objevuje nespočet variant a tvarů). Existují nástroje, kterými se dají modely vytvářet ručně, jako 3D studio Max [9], ale jejich získávání bez pomoci generování procedurálně je složité a ani dlouhodobou snahou nedocílíme dostatečné variability a míry detailů.

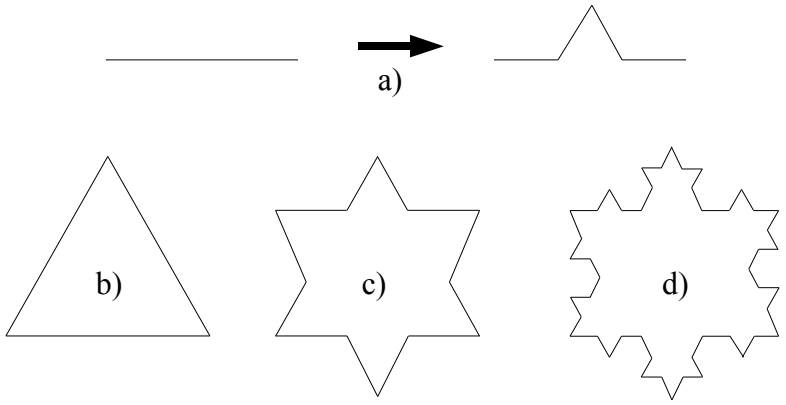
Vhodnějším přístupem, který je použit v této práci i kvůli dodržení stanovené zásady procedurálního modelu z kapitoly 1 - Úvod, je modelování rostlin čistě algoritmicky, což není triviální, přesto existuje několik přístupů, které poskytují vizuálně uspokojivé výstupy již při několika málo informacích na vstupu.

Jedním z těchto algoritmů je modelování rostlin podle vlivu světla, fototropismu rostliny - Visual Model of Plant Development with Respect to Influence of Light [38]. Jiným a o mnoho častěji využívaným přístupem je modelování rostlin fraktály. Přístup se zakládá na představě, že kmen a od něho se větvící hlavní větve u stromů jsou velmi podobné těm nejtenčím rozeklaným větvičkám, jen ve zvětšeném měřítku. Můžeme tedy popsat větvení od hlavní větve (resp. kmene) a poté již tento přístup rekurzivně aplikovat na menší a menší větévky. Jazyk, který pro tento přístup existuje se nazývá L-systém, je popsaný na stránkách Fractint L-Systems Definition [13] a pro stromy pak konkrétně Fractint L-Systems Plants [13]. Tento přístup je dále rozveden v práci Parametric l-systems and their application to the modelling and visualization of plants [20]. Z mého pohledu nejlepší a nejrozsáhlejší prací je The algorithmic beauty of plants [33] na stránkách Algorithmic botany [6], kde se autoři zabývají nejen modelováním těl rostlin pomocí Lindenmayerovy gramatiky (L-systému), ale popisují také procedurální vytváření listové žilnatiny, květů a souplodí, růstu rostlin, mušlí, praskání kůry a suché země a další spoustu přírodních úkazů, které se jim podařilo fraktály popsat. V následující kapitole je podrobněji popsána právě Lindenmayerova gramatika.

4.1 L-systém

Lidenmayerovy systémy, neboli L-systémy, byly vytvořeny jako matematická teorie růstu rostlin. Původně neobsahovala dostatek detailů pro modelování samotných rostlin, ale byla určena pouze pro popis šíření rostlin s jejich vzájemnými vztahy. Později se ale objevilo mnoho geometrických interpretací L-systémů a ty se díky nim staly univerzálním nástrojem na modelování rostlin.

Základem L-systémů jsou přepisovací pravidla. Pro ilustraci poslouží příklad se sněhovou vločkou, viz obrázek 10.



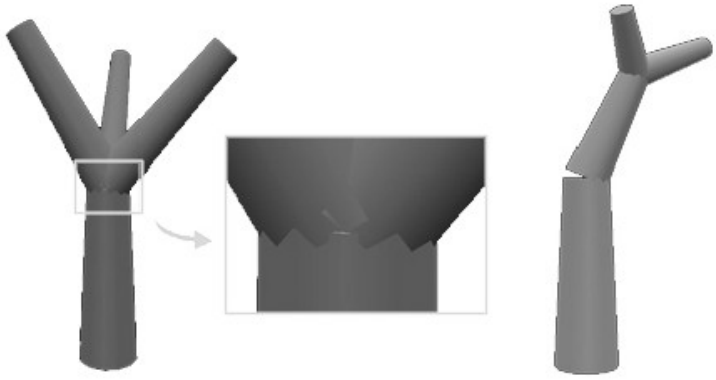
Obrázek 10: Příklad L-systémů: a) Přepisovací pravidlo, b) výchozí stav, c) po prvním kroku, d) po druhém kroku.

Nejjednodušší a čitelný systém pro taková pravidla je řetězec znaků. Popis tohoto systému přinesl Aristid Lindenmayer a podobá se Chomského gramatice s tím rozdílem, že přepisovací pravidla jsou v Lindenmayerově gramatice aplikována paralelně na celý řetězec, zatímco v Chomského sekvenčně. Jen doplním, že v případě této práce je použita verze DOL-systémů, ve které je gramatika deterministická a bezkontextová.

4.2 Převod do polygonů

Popsané L-systémy jsou dobrým nástrojem pro vytvoření jakési „kostry“ výsledného objektu. Jejich výstupem je řetězec znaků, který se dá jednoduše interpretovat do orientovaného, hierarchického grafu, který popisuje přímky v prostoru, jejich větvení, otáčení apod. Taková reprezentace ale pro účel modelování rostlin nestačí a i když může taková „kostra“ modelu obsahovat u každé části i její tloušťku, je pro realistický vzhled potřeba nějakého robustního algoritmu na převod do polygonální reprezentace. Otevřené L-systémy sice nabízejí možnost popisu povrchu (polygonů) modelu, to ale musí být obsaženo již v přepisovacích pravidlech a pro objekty snadněji definovatelné jejich kostrou (orientovaným acyklickým grafem, jako právě u modelů stromů), je takový přístup nevhodný a značně komplikuje popis rostliny.

Nejjednodušším způsobem převodu orientovaného acyklického grafu do polygonální reprezentace je nahrazení jednotlivých úseků kostry hranolem, nejčastěji čtyřbokým kvádrem, jehož rozměry korespondují s uloženou šířku v daném místě. Takový přístup nabízí i implementace L-parseru Laurens Lapre's Project:Lparser [19]. Počtem polygonů a složitostí výpočtu patří sice mezi úspornější metody, přesto nenabízí žádné řešení návazností úseků, natož realistického větvení.



a) b)

Obrázek 11: Vizuální problémy způsobené primitivním spojením. a) Problém s viditelností a b) se spojitostí.

Pokročilejším algoritmem je použití kosých kuželů, jejichž dva poloměry nabízejí možnost lepší návaznosti dvou částí kostry. Vizuální výsledek je o mnoho blíže ideálnímu, přesto se vyskytují chyby s viditelností a u horších algoritmů i s návazností (viz obrázek 11).

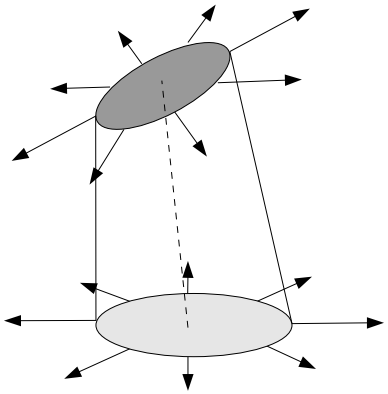
Tento algoritmus je také součástí implementace L-parseru Laurens Lapre's Project:Lparser [19] a v této práci jej budu používat.

Existuje ještě několik pokročilejších metod, např. The Modeling of Branched Structures Using a Single Polygonal Mesh [32], která zavádí do míst spojů několik sub-kontur, kterými se snaží řešit návaznost v místech větvení. Řešení ale není vhodné ve všech případech (např. pro kolmo se oddělující větev) a jeho výpočet je náročnější. Také počet polygonů znatelně narůstá.

Vhodnějším řešením je Polygon Mesh Generation of Branching Structures [23]. Tato metoda předpokládá jako vstup acyklické orientované grafy, které L-systém vytváří, a za využití SMART algoritmu (vytvořeného pro vizualizaci lidských jater) vytváří polygonální mesh. Poloměry v místech větvení aproximuje pomocí metody, kterou prezentoval Leonardo da Vinci pro odhad poloměrů větvení v krevním řečišti. Tato metoda je jistě perspektivní a s využitím Calmull-Clark dělení polygonů poskytuje dostatečně kontinuální a správně se větvící struktury. Pro účely modelování stromů však bohatě postačí metoda navazujících kosých kuželů, která je jak výkonnější, tak i úspornější na počet polygonů.

4.3 Získání lokálních parametrů L-systémů

Na závěr této kapitoly ještě připojuji několik slov o lokálních parametrech vytvořeného 3D modelu stromu. Je třeba na něho správně nanést texturu, vypočítat normály v jeho vrcholech a příp. přidat další lokální atributy, jako je barva, id materiálu apod.



Obrázek 12: Normálové vektory pro jeden úsek kostry rostliny. Horní a dolní podstavou navazuje na další díly. Čárkovaná čára představuje linii kostry a jednotlivé šipky značí normálové vektory.

Normálové vektory se vypočítávají již v průběhu generování modelu. Je to dáno tím, že algoritmus přidává do modelu vždy jeden díl kostry rostliny (v našem případě kosý kužel, viz kapitola 4.2 - Převod do polygonů) a normálové vektory jsou pro celý takový díl snadno zjistitelné. V každém vrcholu směřuje normálový vektor od linie kostry pro daný díl. Navíc vždy leží normálový vektor v rovině základny kosého kuželu pro daný úsek (buď dolní nebo horní). Lépe to ilustruje obrázek 12.

Dalším lokálním atributem je pozice textury. Tu je možné nanášet na každý díl kostry (v našem případě kosý kužel) zvlášť. Vertikální pozice textury (V) je 0 pro horní podstavu a 1 pro dolní. Horizontální pozice (U) se pak pohybuje v intervalu <0,1> postupně pro všechny vrcholy jedné podstavy.

5 Vykreslování

Další rozsáhlou kapitolou je samotné vykreslovaní získaných dat. Cílem této kapitoly je navrhnout dostatečně výkonný systém, který bude interaktivně zobrazovat rozsáhlé zalesněné plochy terénu včetně velmi vzdálených lokalit a zároveň velmi detailně vykreslovat blízké rostliny spolu s jejich listy, větvemi, květy apod. To vše s takovým výkonem, aby bylo možné interaktivně pohybovat s pohledem a prohlížet si tak jednotlivé lokální i globální vlastnosti vygenerované krajiny. Je-li cílem vytvoření takového interaktivního vykreslování, již se neobejdeme s prostým vykreslením všech objektů a musíme zavést několik optimalizací. Jednotlivé přístupy a algoritmy jsou zde rozděleny do skupin podle komponentu krajiny, pro který jsou určeny. Jiné budou pro samotný terén, jiné zase pro vykreslované rostliny.

Vstupem pro tuto fázi budou výstupy fází generování, ať už terénu, těl a pozic jednotlivých rostlin, trajektorií řek či pozic jezer. Výstupem bude rychlá a realistická rasterizace.

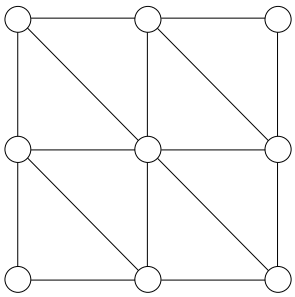
5.1 Terén

V této kapitole je popsán efektivní způsob pro vykreslení realistického terénu, který jsem pro tuto práci zvolil. Je zde popsána reprezentace terénu v počítačové grafice a několik optimalizací, sloužících k urychlení vykreslení rozsáhlé krajiny.

Optimalizacemi vykreslování terénu se podrobně zabývá velká spousta publikací. Jejich přístupy se dají rozdělit do několika druhů, od těch základních, začínajících s jednoduchým ořezáváním části mimo zorné pole pomocí quad-tree, až po sofistikované metody využívajících např. occlusion culling (zakrývaní z pohledu pozorovatele např. jednoho kopce jiným) a vykreslujících části terénu s téměř spojitým Lod (úrovní detailů). Všechny metody mají společný záměr a tím je minimalizace počtu polygonů, které je třeba pro realistický vzhled terénu vykreslit.

Jen doplním, že se mnoho z těchto metod vyvinulo ještě předtím, než se hardwarové renderování stalo naprostou samozřejmostí, a proto se nehodí všechny pro současnou 3D interaktivní grafiku. Důvodem je odlišný přístup pro optimalizaci vykreslování grafickými kartami od přístupu čistě softwarového.

Pro účel této práce jsem proto vybral metodu Geomipmap popsanou v práci Fast Terrain Rendering Using Geometrical MipMapping [12], která se pro současné grafické karty hodí více, a i když neposkytuje minimální počet polygonů, je i přes to velmi rychlá. Je to dáno minimálními výpočetními nároky na určení, které polygony kreslit a které ne. V následujících kapitolách je tedy i metoda popsaná v práci Fast Terrain Rendering Using Geometrical MipMapping [12] vysvětlena podrobněji.



Obrázek 13: Příklad reprezentace terénu pro metodu Geomipmap. Počet čtverců na každé straně musí být mocninou 2.

Reprezentace terénu

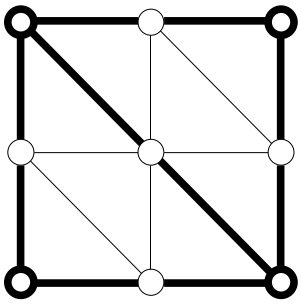
Pro metodu Geomipmap je třeba jisté konkrétní reprezentace terénu. Celá výšková mapa musí být rozdělena do dílů předem dané velikosti N. Ta musí splňovat tu vlastnost, že $N = 2^k + 1$, kde *k* je celé číslo a $k \geq 1$. Tím zaručíme, že počet čtverců v takovém dílu bude mocninou 2. Okrajové vrcholy jsou vždy pro dvě sousední oblasti společné. Na obrázku 13 vidíme oblast 3x3 vrcholů, která sestává z 2x2 čtverců.

Důvodem k tomuto shlukování polygonů je využití většího výkonu grafických akcelerátorů při vykreslení velkého počtu polygonů v jednom kroku. Zároveň, jak je popsáno v dalších kapitolách, má tato reprezentace i výhody při zobrazování různých stupňů úrovně detailů.

Ořezání podle zorného pole

Pro co největší výkon při vykreslování terénu je třeba nevykreslovat v ideálním případě žádný polygon, který není z pohledu uživatele vidět. Rozhodování o viditelnosti každého jednotlivého polygonu ale zabírá příliš výpočetního času, a tak se v praxi setkáme spíše s rozhodováním, zda vykreslit či nevykreslit celý shluk polygonů, který byl popsán v předešlé kapitole - Reprezentace terénu. Pro ještě větší urychlení se navíc data organizují do stromové struktury, tzv quad-tree (viz práce Quad Trees: A Data Structure for Retrieval on Composite Keys [24]), kde nejvyšším uzlem je celá mapa a každý uzel ve struktuře má 4 potomky (odpovídající 4 jeho čtvrtinám). V listech stromu pak leží výše popsané shluky polygonů. Ke každému uzlu stromu je poté dopočítán tzv. obalový kvádr (v anglické literatuře Bounding box). Při vykreslování se pak začne u nejvyššího uzlu stromu (celé mapy) a vždy se zjišťuje, která ze 3 možností nastala:

- Celý obalový kvádr je v zorném poli: Vykreslíme celý jeho podstrom bez dalšího testování.
- Obalový kvádr je částečně v zorném poli: Zanoříme se a otestujeme všechny 4 potomky.
- Obalový kvádr není v zorném poli: Celý jeho podstrom nevykreslíme (přeskočíme).



Obrázek 14: Geo MipMap. Silněji je zobrazena menší úroveň detailů.

Geo MipMaps

Pokud jsou díly mapy více vzdáleny od kamery, mohou se bez větší vizuální újmy vykreslovat v menším počtu polygonů. Tomu se říká Lod (úroveň detailů). Pro hardwarovou akceleraci na grafických kartách se ale nehodí přístup založený na zvážení každého jednotlivého polygonu a lepším přístupem je vygenerování několika geometrických rozlišení dílu mapy, ze kterých v době vykreslování už jen vybereme ten nejvhodnější.

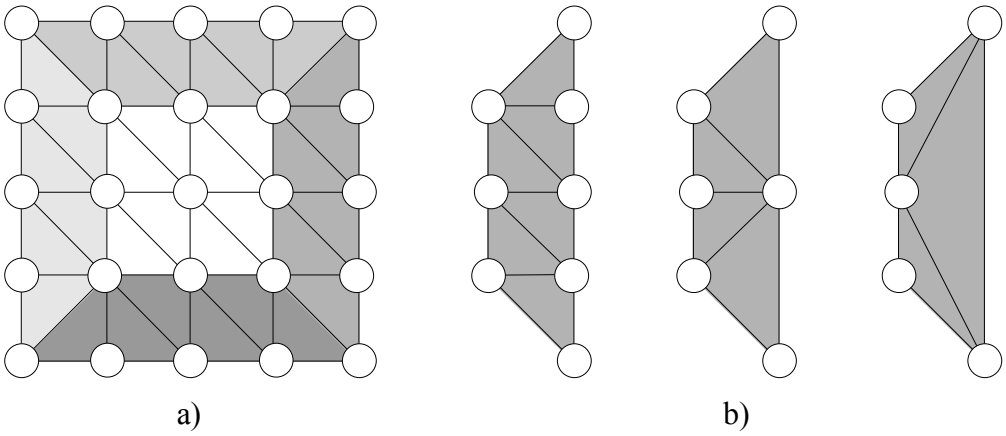
První takovou úrovní je kompletní díl, jako bychom vykreslovali původně, další úrovní je pak použití každého druhého vrcholu jak horizontálně, tak vertikálně, atd. až do vytvoření potřebného počtu rozlišení. Na obrázku 14 je zobrazena druhá úroveň (pouze zvýrazněné body se použijí).

Vhodný přístup pro použití v hardwaru je vygenerování kompletního pole vrcholů pro každý díl mapy a pro celou mapu pak jednoho pole indexů pro každé rozlišení. Kombinací pole vrcholů pro daný díl mapy a pole indexů pro dané rozlišení můžeme vykreslit jakýkoli díl mapy s jakýmkoli rozlišením v jednom kroku. Tento způsob je také díky jednomu seznamu indexů pro každé rozlišení a jednomu seznamu vrcholů pro každý díl mapy úsporný na paměť grafické karty.

Vyřešení návaznosti dílů

Pokud použijeme různé rozlišení pro dva sousední díly mapy, vzniknou na jejich společné hranici nespojitosti, které způsobují vizuální chybu (v angl. literatuře označovanou jako T-crack). V takovém případě nabízejí práce Fast Terrain Rendering Using Geometrical MipMapping [12] a ještě lépe popis geomipmap metody v Simplified Terrain Using Interlocking Tiles [30] elegantní a funkční řešení, které je použito i pro tuto práci a nyní bude následovat jeho popis.

Díl mapy se rozdělí na oblast středu a čtyři oblasti hraniční. Pro každou hraniční oblast se pak vygeneruje samostatné pole indexů pro každý případ nižší sousední úrovně detailů. Výsledné pole indexů pro daný díl mapy se pak získá složením pole indexů pro střed v daném rozlišení a 4 pole indexů pro dané rozlišení a dané rozlišení souseda. Přehledněji to opět můžeme vidět na obrázku 15.



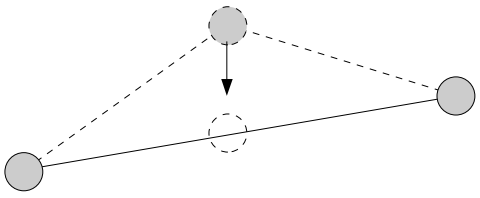
Obrázek 15: Interlocking tiles metoda. a) Výsledné polygony pro díl vzniknou složením středu a okrajů s ohledem na Lod souseda. b) Jednotlivé verze okrajové oblasti podle Lod souseda.

Hardware geomorphing

Při změnách úrovně detailů se často setkáme s viditelným „bliknutím“, způsobeným skokovou změnou jednoho geometrického rozlišení do druhého a tím zmizením, resp. objevením se, některých vrcholů. Povrch terénu se v takovém místě náhle posune do průměrné pozice sousedních vrcholů, které zůstaly. Na obrázku 16 je situace zobrazena.

Takové chybě se dá předejít. Pokud se bude blížit odstranění nějakého vrcholu, je třeba jej předem plynule posouvat do budoucího místa povrchu. Tomuto přístupu se říká geomorphing a zabývá se jím i práce Fast Terrain Rendering Using Geometrical MipMapping [12].

V dnešní době je výhodnější geomorphing počítat pomocí hardwaru a informace o tomto přístupu zase poskytuje článek Terrain geomorphing in the Vertex Shader [32]. Autor v něm popisuje interpolaci mezních pozic vrcholu pomocí devíti parametrů, které k němu připojí. Takový přístup ale vyžaduje více parametrů, než je nezbytně nutné pro účely jednoduššího geomorphingu na výškové mapě a pro tyto účely dokonce stačí k vrcholu připojit jen parametry dva. Nyní bude tento upravený algoritmus popsán a ukázáno, jak jej použít.



Obrázek 16: Posunutí povrchu terénu při odstranění jednoho vrcholu. Viditelnému bliknutí se předchází postupným posunutím vrcholu před jeho odstraněním.

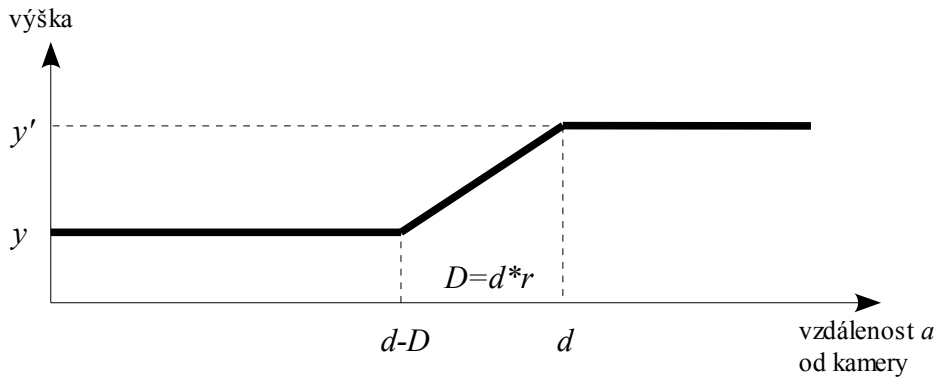
Předpokládejme, že vrchol již obsahuje informace o své poloze, tedy *x*, *y* a *z*. Prvním atributem navíc bude vzdálenost *d*, kde musí být daný vrchol již v nové pozici a brzo tedy bude eliminován a samotná nová pozice *y′* (pokud potřebujeme další parametry, do kterým interpolujeme, jsou přidány právě na tomto místě, stejně jako *y′*. Pro účely prosté výškové mapy však skutečně stačí pouze nová výška *y′*). Při výpočtu pomocí grafické karty se pak volí konstanta *r*, která představuje relativní šířku pásma, ve kterém k přizpůsobování vrcholu dochází. Skutečná šířka tohoto pásma *D*, závisí na definované vzdálenosti změny (tedy *d*) následujícím způsobem: $D = d * r$. Důvodem je fakt, že vzdálenější oblasti s nižší úrovní detailů jsou mnohem větší (exponenciálně), než detailní oblasti a proto ke změně pomocí geomorphingu nemusí docházet tak rychle.

Pokud je tedy vrchol ve vzdálenosti <*d-D, d*>, jeho vzdálenost je *a*, pak jeho výška je vypočítána lineární interpolací pozic *y* a *y'* podle parametru *t* podle vztahu 7.

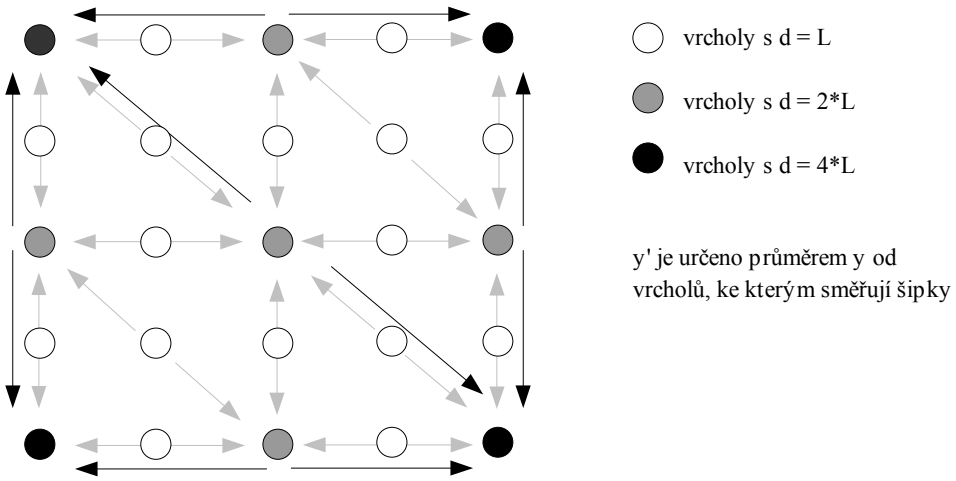
$$t = \frac{a-(d-D)}{D} \qquad (7)$$

Pokud je vzdálenost menší, resp. větší než meze tohoto intervalu, je pozice *y* resp. y*'*. Pro lepší představu se můžeme podívat na obrázek 17. Nyní zde následuje popis, jak vypočítat pro každý vrchol oba parametry: pozici *y′* a vzdálenost *d*.

Nejprve vzdálenost *d*: Ta je odvozena od informace, v jaké úrovni detailů se daný vrchol odstraní a tedy máme-li vzdálenost *L* změny detailů (vzdálenost, do které je terén v maximálních detailech a za ní v nižších), vypadá situace pro použití Geo MipMap jako na obrázku 17.



Obrázek 17: Zobrazení změny výšky bodu od *y* do *y′* v závislosti na vzdálenosti *a* bodu od kamery. Je zřejmá i šíře pásma *D*, v němž se pozice bodu mění.



Obrázek 18: Zobrazení parametru *d* (odstínem šedé) a *y′* (směr šipek) pro hardwarový geomorphing modelu terénu s Geo MipMap.

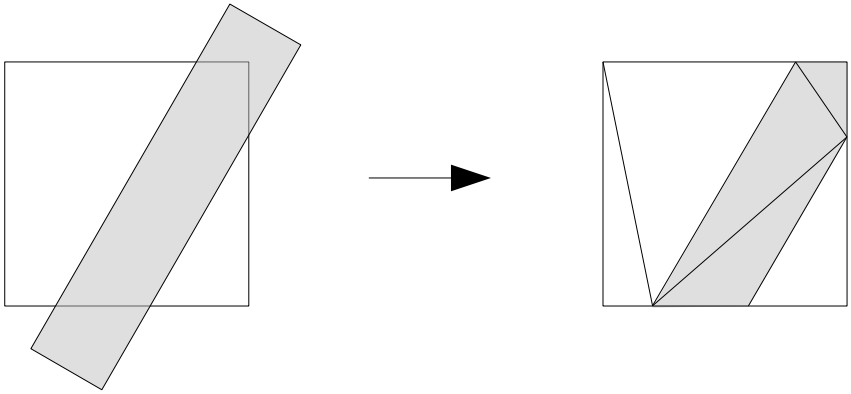
Na obrázku 18 můžeme vidět parametr *d* a závislost pozice *y′* na okolních bodech pro metodu geomorphing. Stupni šedi jsou znázorněny body podle parametru *d* (je vidět, že nejprve jsou v metodě Geo MipMap odstraněny bílé vrcholy a mají proto vzdálenost *d* již rovnu *L*, další jsou odstraněny šedé body a mají proto *d=2*L* atd.). Parametr *y′* je pozice vrcholu těsně před tím, než je odstraněn, a proto musí být ve stejné výšce jako terén v daném místě bez tohoto vrcholu. Na obrázku jsou vidět šipkami vrcholy, na kterých pozice *y′* odstraňovaného vrcholu závisí (pozice *y′* je průměrnou hodnotou parametrů *y* obou sousedních vrcholů, ke kterým šipky míří).

Pozor je třeba si dát na diagonální šipky. Diagonála je totiž dána uskupením dvou polygonů, které tvoří čtverec. V případě metody Interlocking Tiles je směr diagonály často různí z důvodu návazností hraničních oblastí apod.

5.2 Řeky

Tato kapitola se věnuje zobrazování řek na terénu. Vstupem jsou trajektorie řek a jejich šířky. Pro zjištění řek na generovaném terénu slouží nová metoda popsaná v kapitole 2.6 - Řeky a jezera. Samotné vykreslení řeky lze řešit jedním ze dvou základních principů: Buď vykreslíme polygonální reprezentaci takové řeky, nebo pouze upravíme či přidáme texturu nanášenou na terén.

První přístup je zkomplikován složitostí vytvoření polygonální reprezentace, která musí korespondovat s reprezentací samotného terénu. Nelze tedy pouze převést trajektorii a šířku řeky do polygonů, musíme tyto polygony ještě přizpůsobit terénu. Celou situaci ilustruje obrázek 19. Tím se zamezí vizuálním chybám, jakými jsou nezobrazení řeky, jejíž polygony jsou pod terénem, zobrazení řeky nad terén a problikávání řeky v případě konfliktů z-bufferu.

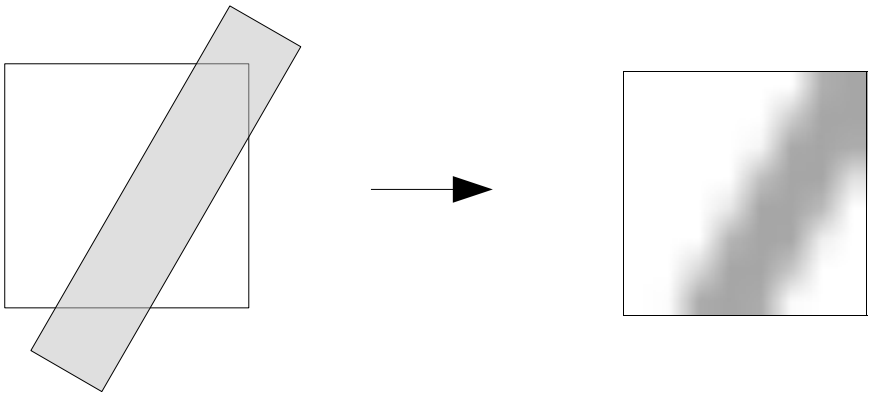


Obrázek 19: Způsob zobrazení řek: Převod řek do polygonů.
Šedou je znázorněna řeka a bílým čtvercem díl mapy.

Jak je z předchozího výčtu zřejmé, není první metoda (tedy převod řeky do polygonální reprezentace) jednoduchá a krom zmíněných komplikací je i náročnější na počet polygonů, tedy i rychlost vykreslení. Pro tuto práci jsem proto vybral metodu druhou, tzn. úpravu textury nanášené na terén. Jejím principem je aplikace speciální před-počítané textury (resp. textur) pokrývající buď celý terén, nebo alespoň z pohledu uživatele viditelnou část.

Metoda spočívá ve dvou krocích. Prvním je vytvoření textury s potřebným rozlišením, na kterou vykreslíme řeky. Zde se sice opět používá polygonální reprezentace trajektorie a šířky řek, nemusíme ale brát v potaz terén s výškovou mapou (jedná se tedy o jednodušší případ, znázorněný na obrázku 20). Navíc je tato textura před-počítána a nepřidává tedy jediný polygon do fáze vykreslování. Její nevýhodou je zvýšení nároků na paměť grafické karty a potřeba při vykreslování aplikovat na terén ještě tuto texturu. Nevýhoda paměťových nároků se dá snížit využitím dynamického překreslování této textury pouze pro místa, která se dostala do blízkosti uživatele.

V kapitole 6.4 - Rozšíření - Animace řek jsou popsány rozšíření vykreslení řek, jako je animace proudící vody a zobrazení dna řeky.



Obrázek 20: Druhý způsob zobrazení řek: Převod řek do textury.
Šedou je znázorněna řeka a bílý čtverec představuje díl mapy.

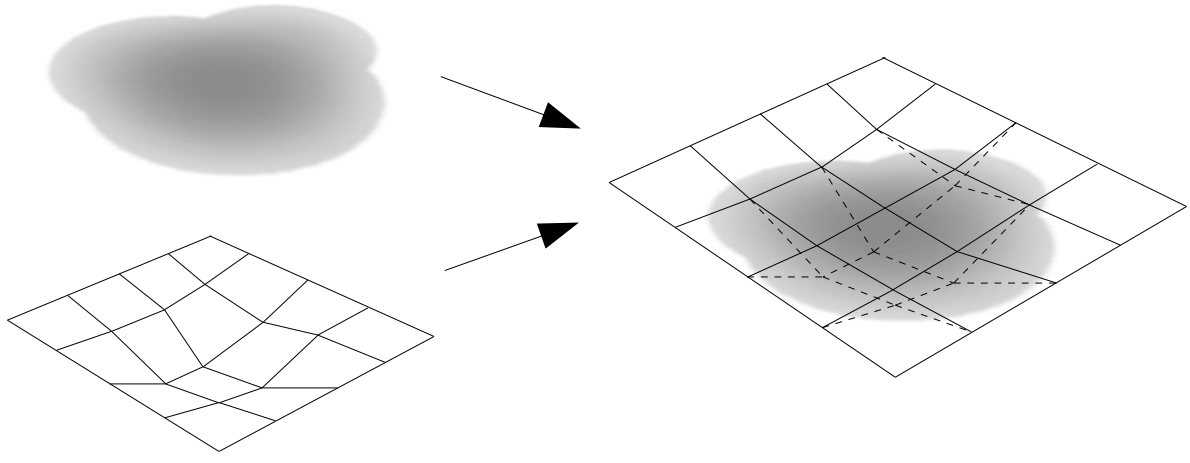
5.3 Jezera

V této kapitole se práce věnuje vykreslování jezer. Jejím vstupem jsou definované tvary a výšky hladin jezer na terénu, pro jejichž počítání může sloužit metoda z kapitoly 2.6 - Řeky a jezera.

Při vykreslování jezer je třeba vyřešit téměř stejné problémy, jaké jsou obsahem předchozí kapitoly 5.2 Řeky. I zde je třeba vykreslovat téměř obecné tvary, které se nedají snadno aplikovat na pravidelnou mřížku výškové mapy a potažmo její polygonální reprezentaci. Také zde se tedy možné přístupy mohou rozdělit na převod do polygonální reprezentace beroucí v potaz terén a přístup s před-počítáním či úpravou textury terénu.

Pro účely této práce jsem i zde zvolil druhou metodu, tedy před-počítání speciální textury s jezery (textura pro řeky a pro jezera se může kombinovat a tedy opět snížit paměťové nároky a čas na vykreslení). Jedinou změnou oproti textuře pro řeky je nutnost zobrazovat nejen plochu jezera, ale i jeho objem (tedy nejen pokrýt jeho dno texturou jezera, ale zobrazovat i hladinu). Možným řešením je přidání dalších polygonů pro hladinu jezera a tu s jistou průhledností vykreslovat. Tímto ale opět vzniká problém s vhodnou polygonální reprezentací hladiny a vzrůstá i počet polygonů.

Můj přístup spočívá v úpravě speciální před-počítané textury tak, aby kromě samotné rozlohy jezera ukládala i jeho hloubku. Při vykreslování terénu se podle před-počítané textury nejen aplikuje textura hladiny, ale posouvají se podle ní i vrcholy terénu do úrovně hladiny (to je znázorněno na obrázku 21). Výsledná textura na terénu je poté kombinací textury dna zatemněné barvou hluboké vody podle hloubky jezera a textury hladiny. V kapitole 6.4 - Rozšíření - Zrcadlení jezer jsou popsány další rozšíření vykreslování, jako je Fresnelův teorém, zrcadlení a animace hladiny jezer.



Obrázek 21: Kombinace terénu a před-počítané textury jezera. Vrcholy terénu se posunují do pozice hladiny, čárkovaně je naznačena jejich původní poloha.

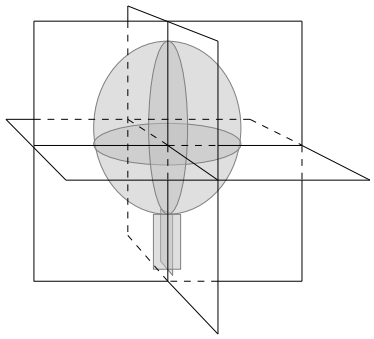
5.4 Rostliny

V této kapitole jsou popsány způsoby, jak reprezentovat a především efektivně vykreslovat vegetaci na terénu. Zobrazování rostlin se od vykreslení terénu podstatně liší a je třeba použít odlišné optimalizace a metody. Jsou komplikovanější nejen větší geometrickou komplexností a tedy nutností vykreslení většího počtu polygonů, ale i větší variabilitou a složitější strukturou. Oproti terénu mají ale výhodu ve snazší separaci jednotlivých částí a v lokálním efektu příp. změn (tento fakt také využívá většina optimalizací - pokud část z vegetace nevykreslíme nebo změníme její úroveň detailů, není třeba okolní rostliny přizpůsobit). Následuje popis jednotlivých optimalizací, které byly pro účely této práce použity.

Aproximace křížem

V dřívějších dobách se rostliny vykreslovaly pomocí tzv. sprite, který představoval obrázek umístěný do scény (při pohybu kamery vytvářel dojem, že se natáčí vždy kolmo ke kameře). Aby se odstranil tento jev, byl jediný sprite často nahrazen 2 svislými plochami (4 polygony), které byly umístěny do scény a orientovány do kříže. Tento přístup je ze současného pohledu vhodný pro středně vzdálené rostliny. Pro blízké již představuje příliš hrubý model a pro vzdálenější jsou vhodnější jiné přístupy, které dokáží jedním polygonem vykreslit více stromů (viz následující kapitola - Impostor).

Metoda, kterou jsem si vybral, používá 6 čtverců (tedy 12 polygonů) orientovaných kolmo ke 3 osám *x, y, z* vždy v pozitivním a negativním směru (obrázek 22). Textury, které je pokrývají, jsou předkresleny v době vytváření rostliny (stačí tedy jedna sada pro všechny instance daného druhu rostliny) a zachycují objekt ze šesti směrů. Navíc jsou vykresleny do jedné velké textury, což má další pozitivní vliv na rychlost vykreslování.



Obrázek 22: Aproximace křížem. 6 čtverců (vždy dva na opačnou stranu) orientovaných do kříže.

Impostor

V nejmenší vzdálenosti jsou rostliny reprezentovány jejich 3D modelem, ve středních vzdálenostech aproximací křížem. Jak ale efektivně vykreslovat objekty ve velkých vzdálenostech, kde je jich o několik řádů více než v blízkosti kamery? Vhodnou a v dnešní době často využívanou metodou je v anglické literatuře tzv. impostor či impostoring.

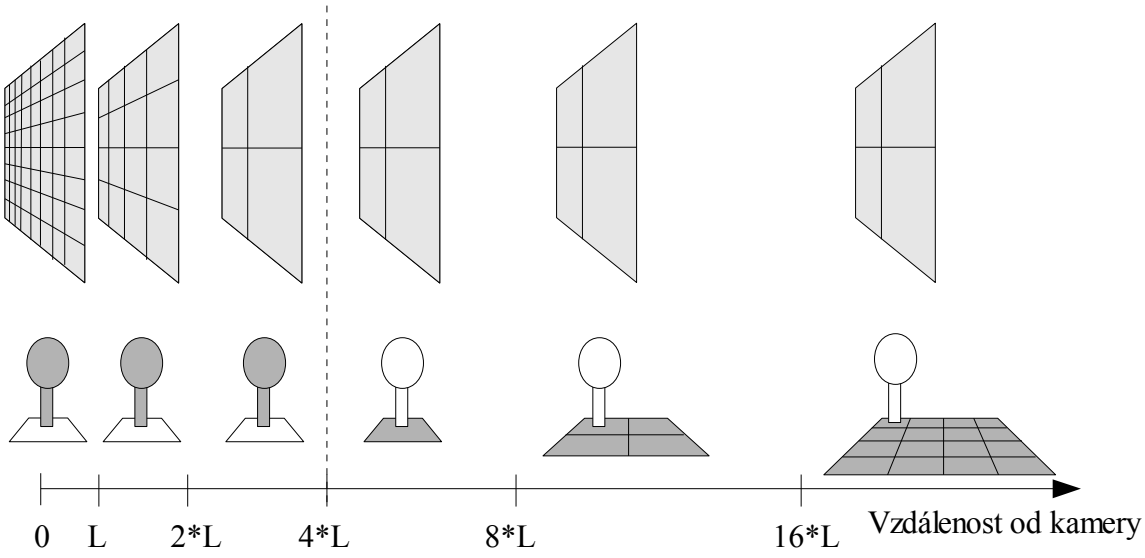
Spočívá v nahrazení rostliny či její části jedním nebo více texturovanými polygony. Práce An Image-Based Multiresolution Model for Interactive Foliage Rendering [8] uvádí dvě základní verze imostoru a to buď jeden polygon otáčející se ke kameře (v angl. literatuře již zmíněný sprite či billboarding) a nebo dva polygony orientované do kříže. Metoda se v některých ohledech velmi podobá aproximaci křížem. Hlavním rozdílem je ale vykreslování textur pro impostory dynamicky za běhu aplikace, k čemuž se vztahuje celá množina pravidel a událostí, kdy se s texturou manipuluje. Také je jedna textura přiřazena pouze jednomu objektu a zachycuje ho z aktuálního směru od uživatele. U aproximace křížem je textura jen jedna pro všechny rostliny stejného druhu a za běhu aplikace se nemění.

Práce An Image-Based Multiresolution Model for Interactive Foliage Rendering [8] se dále zabývá generováním těchto impostorů pro jednotlivé větve a možností převádět L-systémy přímo na textury. Jiný přístup nabízí práce View-Dependent Multiresolution Model for Foliage [37], která není založena na textuře ale na zjednodušování geometrie rostliny.

Pro účely této práce jsem zvolil konkrétní metodu impostorů pomocí jednoho polygonu otáčejícího se za kamerou. Dále je třeba vykreslovat velké množství rostlin, ideálně pomocí jediného společného polygonu a s detaily závislými na vzdálenosti od uživatele. Proto zde popíši metodu pro změnu rozlišení impostorů za běhu aplikace a novou metodu pro dynamické shlukování objektů do jediného impostoru.

Impostory se vykreslují do dílů větších textur, tzv katalogů textur. Do jedné jsou vykreslovány impostory se stejným rozlišením, tím je tato textura využita efektivně a v následující kapitole (Instancing) je uvedena ještě výhoda tohoto přístupu ve vykreslení všech impostorů jedné textury pomocí jednoho kroku vykreslení.

Rozlišení impostorů je dáno jejich vzdáleností od uživatele, a proto se v průběhu času dynamicky mění spolu s přibližováním nebo vzdalováním uživatele. Vzdálenost, ve které dochází ke změně rozlišení impostoru, je dána exponenciálně (první snížení rozlišení je ve vzdálenosti *d*, další poté *2*d*, pak *4*d* atd.) a od definované vzdálenosti se již nesnižuje. Naopak se do jednoho impostoru vykresluje více objektů a exponenciálně se zvětšuje oblast, ze které se všechny objekty kreslí do jediného impostoru. Situace je znázorněna na obrázku 23.



Obrázek 23: Shlukování impostorů.V levé části se snižuje rozlišení impostoru pro každou rostlinu. V pravé části se rozlišení již nesnižuje, ale jeden impostor sdílí více rostlin, resp. všechny rostliny z čím dál většího dílu mapy.

Překreslení impostoru nastává při splnění jedné z definovaných podmínek. První z nich je, že byl impostor právě vytvořen a je třeba naplnit pro něho texturu. Druhý případ nastává, pokud se změnila vzdálenost impostoru od kamery natolik, že již spadá do jiné úrovně detailů (resp. rozlišení impostoru či velikosti oblasti shlukované do tohoto impostoru). Poslední případ nastává tehdy, když se vektor od impostoru ke kameře změnil o definovanou mezní hodnotu úhlu. Je možné doplňujících podmínek, např. v případě animace rostliny i po uplynutí jisté doby, apod.

Ve velké vzdálenosti je třeba při obnovení impostoru vykreslit celou řadu často detailních modelů rostlin (všechny, které spadají do tohoto impostoru). Pro udržení dostatečného vykreslovacího výkonu je možné od jisté vzdálenosti namísto modelů rostlin použít jejich křížovou aproximaci.

Nakonec ještě doplním metodu pro výpočet rozměrů impostoru, tedy jak velké budou polygony impostoru ve scéně. Nejvhodnější metodou je tzv. Bounding sphere, tedy obalová koule. Výpočtem se z modelu rostliny zjistí její střed S a poloměr r a podle nich se vytvoří polygon impostoru (se stejným středem S a rozměry $2*r$). V případě shlukování objektů se zvětšuje i velikost impostoru a to použitím opět obalové koule na všechny obalové koule jednotlivých objektů.

Instancing

Další metodou, kterou je možné zvýšit vykreslovací výkon, je Instancing. Jedná se o metodu, pomocí které se vykresluje více stejných objektů jedním krokem vykreslení. Právě tím se hodí pro zobrazování rostlin, protože jak již bylo zmíněno v úvodu, pro efektivní vykreslování je třeba

objekty shlukovat v parametrickém prostoru a zprůměrovat na jeden reprezentující objekt, který se pak vykresluje pro všechny takové rostliny stejný. Jedním krokem vykreslení je tak možné zobrazit více stejných rostlin a urychlit tak jejich vykreslování.

Samotný princip metody spočívá v tom, že spolu s modelem se do grafické karty zasílají i informace o jednotlivých instancích (pro každou rostlinu např. pozici, natočení apod.). Všechny instance daného modelu se pak vykreslí naráz za použití těch několika drobných doplňujících změn pro každou jednotlivou rostlinu.

Pokud se jedná o model impostoru (dva polygony), mají dokonce všechny rostliny stejný model a liší se pouze použitou texturou. Díky využití katalogů textur je ale do jedné textury uloženo více impostorů (jejich maximální počet je dán velikostí textury a rozlišením impostorů) a díky tomu je možné vykreslit pomocí instancingu všechny impostory, uložené v jedné textuře.

Hranice grafické paměti

Při optimalizacích se často setkáme s rozhodováním mezi větším výkonem a menší spotřebou paměti, kdy je pro optimální řešení třeba využití maximální velikosti paměti a tím umožnit co největší výkon. Stejně tomu je i v případě impostorů, kde hledáme takový stav, ve kterém jsou impostory v co největším rozlišení do co největší vzdálenosti a za využití právě tolika paměti, kolik jí je k dispozici.

Moderní grafické akcelerátory již umožňují získávat za běhu aplikace informace o tom, kolik je alokováno paměti a kolik jí grafická karta maximálně disponuje, a proto je možné na základě těchto údajů s pamětí efektivně hospodařit. Metoda, jak hospodařit s pamětí impostorů, spočívá v dynamické alokaci katalogů textur, v překreslování těchto textur tak, aby se maximálně využívaly (impostory z textur, které jsou téměř nevyužity se překreslují do volných dílků na jiných texturách) a v omezení počtu katalogů textur v závislosti na zbývající paměti.

Pokud například zobrazujeme nový impostor, pro který již není volný dílek v katalogu textur a nová textura se již do paměti nevejde, použijeme namísto impostoru křížovou aproximaci, či blízko kamery celý 3D model rostliny. Tím nedojde k výpadku objektu z vykreslované scény a paměť se nezvýší nad maximální mez.

5.5 Adaptive lod

Distribuce rostlin po mapě nebývá zpravidla příliš pravidelná a vegetace se vyskytuje spíše v uzavřených shlucích. Stejně tak terén může být komplexní v pohledu do dálky a velmi prostý z pohledu na nebe do země či kopce před uživatelem. Díky tomu i rychlost vykreslování při určování úrovní detailů čistě podle předem nastavených hraničních vzdáleností může velmi kolísat. Jiné rychlosti vykreslení dosáhneme v hustém porostu a jiné zase na otevřených prostranstvích s velkým výhledem.

Způsob, jakým dosáhneme vyrovnanějších rychlostí, příp. garantované minimální rychlosti vykreslení scény, může být adaptivní přizpůsobování hranic pro Lod. O této metodě se zmiňuje publikace Game programming gems [15]. Spočívá na principu, kterým se nová hraniční hodnota pro Lod nastaví podle předchozí a podle toho, jak se daří dodržovat stanovenou dobu pro vykreslení jednoho snímku. Základní vztah je ve vzorci 5.

$$d' = d * (\frac{t_{des}}{t}) \qquad \textbf{(5)}$$

Kde d' je nová vzdálenost pro změnu Lod, t_{des} je požadovaný čas vykreslení a t je čas posledního vykreslení. Pro klouzavý odhad můžeme t průměrovat z více snímků, popř. přidat konstantu k, představující rychlost změny hraniční hodnoty d. Viz vztah 6.

$$d' = d * ((1 - (\frac{t_{des}}{t})) * k + 1) \qquad \textbf{(6)}$$

Díky tomuto přístupu je tak možné nastavit, kolik času má aplikace trávit vykreslováním rostlin, kolik terénu a ve výsledku pak dochází k dynamickým změnám, díky nimž je vykreslení podobně časově náročné (pokud je např. uživatel v hustém lese, dohledová vzdálenost se sníží za zachování detailů u blízkých rostlin, zatímco je-li uživatel na holé pláni či vyhlídce, vzdálenost se opět zvýší a umožní zobrazení vzdálených horizontů).

5.6 Hystereze

V případě, že od jisté vzdálenosti skokově změníme rozlišení impostoru, mohou nastat v podstatě dva problémy. Za prvé se může stát, že bude impostor kolísáním velmi blízko této hranice rychle měnit rozlišení a tím vynucovat příliš často své překreslení. Řešením tohoto problému je tzv. hystereze, tedy úprava algoritmu, při které namísto jedné hraniční hodnoty použijeme dvě. Jednu pro zvětšení úrovně detailů a druhou pro snížení. Pokud je vzdálenost impostoru mezi těmito hodnotami, ponecháme jeho rozlišení beze změn.

Druhým problémem je vizuální „bliknutí“ impostoru při skokové změně jeho rozlišení. Řešením je použití, podobně jako u geomorphingu, jistého pásma, ve kterém budou vykresleny dvě textury pro jeden impostor. Jedna s původním a druhá s novým rozlišením. Obě textury jsou pak při vykreslování prolínány podle vzdálenosti impostoru od kamery. Použití dvou textur pro každý impostor ale velmi zvyšuje paměťové nároky a v demonstrační aplikaci nebyla tato metoda použita.

6 Demonstrační aplikace

K této práci vzniklo jako příloha i programové dílo, které implementuje a demonstruje popsané metody generování a vykreslovaní. V následujících kapitolách jsou popsány detaily tohoto programu, jednotlivá rozšíření generování či vykreslování, která jsou nad rámec popsaných metod, a také jsou zde výsledky a výkon aplikace.

Nechybí popis znovupoužitelnosti částí programu (díky dynamickým knihovnám s dokumentovaným rozhraním), modifikovatelnost programu pomocí xml souborů s nastavením a možnost rozšíření programu například o podporu jiných renderovacích knihoven (rozšíření na OpenGl) či jiných operačních systémů (Linux, Mac)

6.1 Návrh a implementovaný systém

Aplikace demonstruje popsané metody a jedná se o 3D aplikaci s vhodným ovládáním pro interaktivní prohlížení vygenerovaného terénu, možností měnit parametry generování a jednoduše tak experimentovat s vlastnostmi použitých algoritmů.

Pro dodržení zásad stanovených v kapitole 1 – Úvod, bylo třeba navrhnout systém s otevřenou architekturou a dobře definovanými vstupy a výstupy jednotlivých generovacích fází a umožnit tak doplnění o nové metody. Zároveň využívat maximálně princip procedurálního generování a umožnit tak uživateli s minimem vstupů získat komplexní a variabilní výsledky.

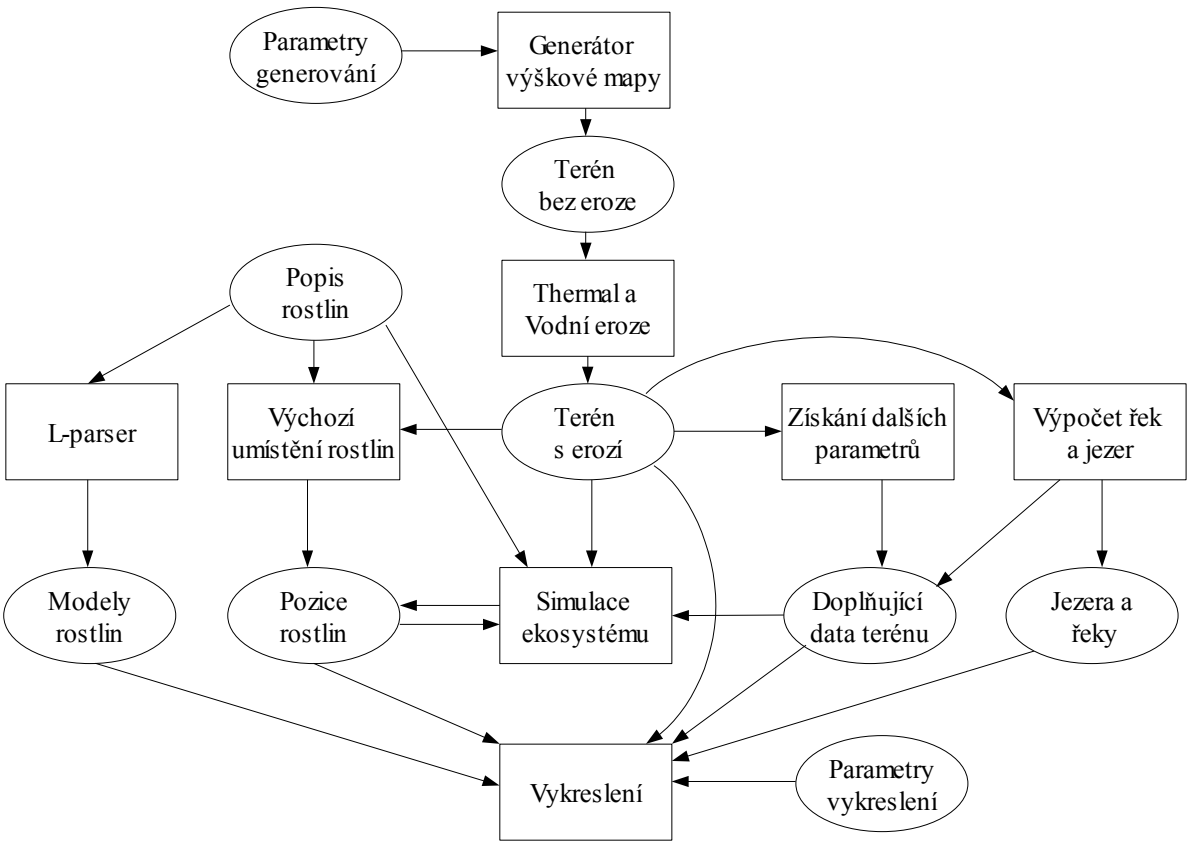
Aplikace pracuje s diskrétním popisem terénu pomocí výškové mapy a implementuje zmíněné eroze (viz kapitola 2 - Generování terénu). Dále obsahuje popsané nové metody pro výpočet řek a jezer na terénu a výpočet doplňujících atributů terénu.

V aplikaci je obsaženo rozmístění a simulace ekosystému rostlin (viz kapitola 3 - Simulace šíření rostlin) včetně popsané metody pro stabilitu simulace.

Pro rostliny jsou využity L-systémy s převodem na polygonální reprezentaci (viz kapitola 4 - Modelování těl rostlin). Dále jsou těla rostlin doplněna lokálními atributy (jsou naneseny textury, vypočteny normálové vektory apod.)

Vykreslování je urychleno pomocí popsaných metod (viz kapitola 5 - Vykreslování) včetně metody interlocking tiles, harwarového geomorphingu (s novou metodou využívající pouze 2 parametry), před-počítaných textur pro řeky a jezera, impostoru s novou metodou změny rozlišení a shlukování objektů do jediného impostoru, instancingu a omezení alokace grafické paměti. Na terén i rostliny je aplikován adaptivní lod s klouzavým odhadem a hysterezí.

Další rozšíření aplikace (především o metody vykreslení) jsou uvedeny v kapitole 6.3 - Rozšíření a detailní architekturu systému popisuje schéma generovacích fází na obrázku 24.



Obrázek 24: Graf architektury spolu se zobrazenými vstupy a výstupy jednotlivých fází generování a vykreslení.

6.2 Variabilita pomocí XML

Aplikace je navržena s ohledem na pravidla stanovená v kapitole 1 - Úvod a tedy obsahuje otevřený dokumentovaný systém pro snadné experimentování s parametry generování. Jejich struktura a množství již neumožňuje nastavení např. pomocí příkazové řádky při spuštění programu a proto jsou řešeny přehledným značkovacím jazykem formátu XML a načítány ze souborů. Pro uživatele je tento přístup zjednodušen i tím, že např. pro přidání nové rostliny do aplikace stačí přidat soubor do specifického adresáře, odkud bude aplikací automaticky načten. Nyní následuje přehled použitých xml souborů či adresářů a jejich struktura a nastavitelné parametry:

- ***setup.xml*** - hlavní nastavení aplikace. Obsah je uzavřen tagem *params* a rozdělen do několika skupin:
  - *heightmap* - obsahuje parametry pro generování výškové mapy, eroze
  - *render* - obsahuje maximální velikost alokované paměti, zapíná/vypíná adaptivní lod pro terén nebo rostliny, stejně tak jako požadovanou rychlost vykreslení terénu či rostlin, nastavuje pravidla pro překreslení impostoru, jeho rozlišení atd.

- *lakes* - zapíná/vypíná výpočet a vykreslení jezer a nastavuje jejich míru (hodně velkých jezer, několik malých)
- *rivers* - zapíná/vypíná výpočet a vykreslení řek, nastavuje jejich míru (hodně velkých řek, jen několik drobných toků)
- *grass* - zapíná/vypíná zobrazení trávy
- *plants* - zapíná/vypíná výpočet a vykreslení rostlin, nastavuje jejich počáteční hustotu rizmístění, věk, upravuje pro všechny vliv nadmořské výšky a nastavuje počet kroků simulace ekosystému

- ***Plants*** - adresář do něhož se umísťují xml soubory s nastavením rostlin. Ty jsou automaticky načteny při spuštění aplikace, proto je jejich rozšíření či odebrání konkrétních druhů velmi snadné.
- ***rostlina.xml*** - každá rostlina je definována jedním xml souborem s nastavením, který mimo její parametry obsahuje i předpis pro generování a další data, viz následující výčet:
  - *name* - jméno rostliny, slouží pro přehlednější organizaci rostlin a navíc se uvnitř enginu dá rostlina vyhledat a změnit ji parametry právě podle tohoto jména
  - *params* - parametry rostliny pro simulaci ekosystému. Obsahují informace např. o vlhkomilnosti rostliny, rychlosti růstu, velikosti, průměrné nadm. výšce výskytu, rozptylu nadm. výšky, šanci vysemenit a vzdálenosti doletu semen, citlivosti na stín a míře stínění apod.
  - *texture_file* - cesta k textuře pro rostlinu. Díky této vlastnosti je možné mezi rostlinami sdílet texturu, či ji uživatelem snadno změnit.
  - *lsystem* - předpis pro generování těla rostliny. Jedná se o L-systém, začínající třemi parametry, následuje axiom a sada pravidel. Jejich přesný popis bude uveden dále v kapitole 6.3 - Gramatika L-systémů
  - *fxeffect_file* - cesta k souboru s pixel a vertex shadery pro rostlinu. Může se nahradit parametrem *fxeffect*
  - *fxeffect* - text shaderů pro rostlinu, možno zaměnit s *fxeffect_file*

6.3 Gramatika L-sytémů

Použité L-sytémy jsou definovány strukturou a pořadím zadávaných parametrů, použitými znaky, jejich významem a syntaxí zápisu. Nyní následuje jejich popis v pořadí, v jakém jsou uvedeny v předpisu pro L-systémy v aplikaci.

Nejprve jsou uvedeny 3 základní parametry L systému vždy odděleny koncem řádku. Představují po sobě: počet iterací růstu (rostlina se generuje v několika verzích podle stáří, počet

a číslo iterace je dán parametry *lsystem_max_iter* a *lsystem_min_iter* v souboru xml rostliny a parametrem *lsystem_step_iter* v aplikaci), úhel větvení či otáčení a posledním parametrem je šířka základny.

Následuje axiom opět oddělený koncem řádku a sada pravidel. Axiom je zapsán *nonterminály* a *terminály* l-systému a pravidla vždy *nonterminál* = *sada_terminálů* a *nonterminálů*. Každé pravidlo je odděleno koncem řádku a předpis je ukončen symbolem @ na samostatném řádku. Význam nejpoužívanějších symbolů (resp. terminálů) je následující:

- **F** ... vykreslení dílu
- **g** ... posunutí bez kreslení
- **f** .. vykreslení bodu v polygonu
- **{}** .. uzavření definice polygonu
- **[]** ... větvení
- **+ -** ... otočení doprava, doleva
- **<>^&** ... ohnutí +x, -x, +z, -z
- **~** ... náhodné ohnutí
- **c** ... změna barvy
- **' !** ... zesílení, zeslabení resp. zvětšení, zmenšení

K většině parametrů (vyjma závorek) lze připojit mezi symboly '(' a ')' číselně hodnotu pro danou operaci (např '(0.8) udává zúžení na 0.8 - násobek). Použitý L-parser Laurens Lapre's Project:Lparser [19], který byl v této práci použit, využívá celou řadu dalších znaků, už jen uvedu jejich celkový výčet: @, +, -, ~, t, $, &, ^, <, >, %, |, !, ?, :, ;, ', ", Z, F, [,], {, }, f, ., g, z, c.

Uvedený L-Parser umožňuje export generovaných rostlin do formátů VOL, POV object, POV blob, multiple POV blob, DXF, RAW, BLB, VRML, v této práci byl rozšířen i na X (directX mesh). Díky implementovaným mutacím umožňuje tvorby téměř nekonečných variant z jednoho L-systém předpisu. Umožňuje definici jednotlivých polygonů, čehož je s výhodou využito pro tvorbu listů stromů, umožňuje ale i např. fraktály popsané skály, kameny apod.

6.4 Rozšíření

Při implementaci demonstrační aplikace bylo třeba vyřešit některé dílčí úkoly pro správnou funkčnost a zobrazování popsaných metod a jejich výsledků. Vzniklo tak několik rozšíření, které jsou často z kategorie optimalizací a metod vykreslování a nejsou uvedeny v teorii, protože nemění podstatu metod nebo slouží čistě pro lepší vizuální výsledek. V této kapitole jsou změny oproti základním algoritmům popsány.

Prerendering lokálních atributů

Na terén je aplikována řada textur, jejichž mapování je podmíněno lokálními parametry, viz kapitola 2.5 - Získání doplňujících lokálních atributů výškové mapy, tedy např. v místech větší vlhkosti použije aplikace texturu pro svěží trávu, zatímco v suchých místech pro trávu suchou apod. Samotná kombinace textur je prováděna pomocí Multitexturingu (aplikace více textur v jednom vykreslovacím kroku) a před-připravené masky Perlin Noise [21]. Díky tomu se docílí dojmu náhodnějšího a realističtějšího rozmístění.

Tyto parametry se vztahují k jednomu bodu výškové mapy a proto při nižší úrovni detailů pomocí metody Geo MipMap (viz kapitola 5.1 - Terén), kdy ubývá i počet zobrazovaným polygonů, dochází k viditelným ztrátám detailů. Navíc při snížení či zvýšení úrovně detailů terén v místě změny "blikne". Řešením je před-počítání lokálních atributů výškové mapy do speciální textury (v rozlišení 1 bod výškové mapy na 1 bod textury) a mapování na celý terén, či alespoň na okolí uživatele. Textura se principem podobá texturám pro uložení řek a jezer (5.2 - Řeky, 5.3 - Jezera) a může být s nimi kombinována (nabízí se možnost ukládání těchto dat do jednotlivých složek RGBA textury a pomocí pixel shaderů je opět dekomprimovat a použít).

Spolu s lineární interpolací bodů textury získáme dostatek spojitých parametrů pro např. detailní stínování či texturování i v low-polygonálních částí terénu, navíc zpomalení použitím těchto textur je vzhledem k jejich před-počítávání minimální.

V aplikaci byly speciální textury pro terén použity dvě, přičemž první uchovává v každé své složce (r, g, b, a) informace o vlhkosti, míře osvětlení, sklonu svahu a nadm. výšce. Druhá pak v prvních třech složkách uchovává souřadnice normálového vektoru, díky kterému je na terén aplikována normálová mapa a je docíleno vizuálně větších detailů a variability.

Animace řek

Pro zobrazení řek je již popsán způsob před-počítání speciální textury (viz kapitola 5.2 - Řeky). Pro lepší realističnost však nestačí vykreslit pouze speciální texturu, ale je třeba na řečiště nanést texturu dna a vody. Pro rozpohybování textury vody nelze použít snadnější způsob, ve kterém by byly do speciální textury ukládány souřadnice pro mapování tekoucí vody (textura je před-počítána a v době běhu aplikace se již nemění). Nelze ani ukládat směr toku vody a podle něj texturu vody posouvat (na místech soutoků se díky lineární interpolaci tvoří nežádoucí grafické artefakty). Mé řešení tak spočívá v ukládání směru proudění (r složka textury obsahuje směr toku od -x do x a g složka směr od -y do y) a pomocí pixel shaderu je pak nanášeno podle počítaných vah 8 textur posouvaných v 8 základních směrech . Tím i na místech soutoků vypadá proudění vody reálněji.

Výsledný efekt byl ještě zlepšen aplikací masky Perlin Noise [21], která v řečišti vytváří dojem nestejnorodého proudění, obtékání kamenů apod.

Zrcadlení jezer

V kapitole 5.3 - Jezera bylo popsáno zobrazení jezer včetně posunutí povrchu terénu do výšky hladiny. Zobrazení dna jezera je prováděno nanesením textury dna se vzrůstající průhledností v místech, kde je uložena malá hloubka. Textura přechází do temně modré v místech, kde je naopak hloubka největší (tím je docíleno nejen neostrých okrajů jezera, ale také vlivu hluboké vody na horší viditelnost dna).

Zrcadlení je z důvodů velkých nároků na vykreslování omezeno na odraz nebe. Díky tomu také není třeba počítat zrcadlení pro každé jezero v různých nadmořských výškách, ale pouze jednou a použít texturu odrazu bez újmy na realističnosti pro všechna jezera.

Hladina každého jezera je opatřena 4 normálovými mapami v různém měřítku a každá se i různě posouvá v čase po rovině x, y. Jejich kombinací vzniká dojem rozvlněné hladiny. Pomocí závislosti úhlu mezi vektory k uživateli a kolmicí k hladině na míru odrazu (Fresnelův teorém) je dosaženo ještě větší realističnosti (dívá-li se uživatel kolmo do hladiny, vidí zřetelněji dno, zatímco při pohledu do dálky nízko nad hladinou dochází k téměř úplnému odrazu).

Tráva

Vykreslování krajiny je doplněno zobrazením trávy. Děje se tak pouze do omezené vzdálenosti od uživatele (vykreslování trávy do velké vzdálenosti by bylo nejen časově náročné, ale ani by nepřinášelo - vzhledem k malým rozměrům trsů trávy - zamýšleného výsledku).

Polygony trávy jsou předem vytvořeny s ohledem na rovnoměrné ale nepravidelné rozložení přes definovanou plochu. Opakují se vždy v každém čtverci o zvolených rozměrech $d * d$. Když se kamera přesune do nového čtverce, před-připravený model trávy se skokově posune o vzdálenost d. Tím je s minimem paměťových nároků docíleno prakticky nekonečných rozměrů zatravněné plochy.

Program v pixel shaderu trávu při vzdalování se od uživatele postupně zprůhledňuje. Trsy trávy jsou navíc obarvovány do zelených odstínů na vlhčích místech mapy a temnějších na odvrácených svazích od slunce.

Osvětlování rostlin

Zobrazení modelů stromů a květin také nezůstalo v demonstrační aplikaci beze změny. Přidáno bylo stínování koruny a zatemnění rostlin na zastíněných svazích. Již při generování těla rostliny jsou podle principů v kapitole 4.3 - Získání lokálních parametrů L-systému vypočteny normálové vektory.

Těmi je umožněno osvětlování kmene a větví. Listy mají normálový vektor, směřující od středu obalové koule (viz konec kapitoly 5.4 - Rostliny - Impostor). Tím je dosaženo osvětlování listů podobně, jako by celou korunu tvořila jedna koule (odvrácené části koruny se celé zatemní, části přivrácené ke slunci jsou naopak osvětleny).

6.5 Použité prostředí a knihovny

Tato aplikace byla vyvíjena v jazyce C++ a prostředí Visual Studio 2005 spolu s DirectX SDK (ve verzi listopad 2008). Pro 3D zobrazení využívá DirectX9.0c, ale je navržena s ohledem na abstrakci vykreslovací API a proto je možné snadno rozšířit pro např. OpenGl či jinou verzi knihoven DirectX.

Součástí aplikace je i nástroj na parzování XML souborů (soubory nastavení a soubory rostlin) a nástroj na parzování L-systémů LParser.

Samotná aplikace je rozdělena na část enginu s nástroji a část aplikace. Při kompilaci tak vytváří řadu svých dynamických knihoven, sloužících pro práci s okny, pro logiku enginu (graf scény), pro vykreslování a snadné načítání modelů a programů v shaderech.

6.6 Znovupoužitelnost

Demonstrační aplikace sestává z několika dobře oddělitelných dílů, díky čemuž je možné je použít jednotlivě (nebo i všechny) v jiné aplikaci. Při návrhu byla tato možnost zvážena a vzniklo několik dynamických knihoven. Uvedu zde stručně jejich popis, rozhraní a funkci:

- ***LParser.dll*** - knihovna na interpretaci L-systémů a na jejich převádění do 3D modelů či export do souboru.
  - *Generate* (in *počet_parametrů*, in *seznam_parametrů*, out *data_v_paměti*, out *vrcholy*, out *indexy*) - funkce sloužící k převádění L-systémů (*data_v_paměti*, příp načtených ze souboru uvedeného v parametrech), **nápověda** se zobrazí při neuvedení parametrů
- ***LandscapeGenerator.dll*** - knihovna zapouzdřující práci s výškovou mapou a simulací ekosystému rostlin (viz kapitola 2 - Generování terénu, 3 - simulace šíření rostlin). Nastavení parametrů jednotlivých funkcí je provedeno již při konstrukci instance třídy. Samotná třída obsahuje mnoho funkcí a já zde uvedu jen ty nejdůležitější:
  - *FaultFormation*() - provede na výškové mapě metodu tektonických zlomů.
  - *Thermal*(in *počet_iterací*, in *maximální_sklon_svahu*) - Sesuv půdy
  - *FaultFormationErosion*() - FIR filter nad výškovou mapou
  - *FunWithErosion*() - metoda vodní eroze

- *Canny*() - metoda pro výpočet řek
- *FillAllLakes*() - výpočet jezer
- *InitTrees*() - rozmístění stromů
- *StepTrees*() - jeden krok simulace ekosystému
- *GetHeight*(), *GetHumidity*(), *GetTree*(), *GetSlope*(), ... - způsob získání generovaných dat

- ***Engine.dll***, ***Dx9Renderer.dll***, ***WinXPWindow.dll*** - knihovny zapouzdřující práci enginu s grafem scény a abstrakcí vykreslovače a práce s okny, konkrétní verzi vykreslovacích API a knihovny na správu oken.

6.7 Třídní dekompozice

Pro přehledný návrh, lepší udržovatelnost a rozšiřitelnost zdrojového kódu je objektově orientovaný návrh nutností. Rozdělení do dynamických knihoven s třídní dekompozicí úzce souvisí, přesto zde uvedu i rozdělení do tříd a funkce jednotlivých tříd.

- ***BoundingSphere*** - soubor metod pro výpočet obalové koule
- ***Dx9Mesh*** - zapouzdření práce s 3D modelem v DirectX9.0c
- ***Dx9Renderer*** - konkrétní vykreslovací API založené na DirectX9.0c
- ***Engine*** - třída spravující graf scény, abstrakci vykreslovačů apod.
- ***Grass*** - třída obstarávající vykreslní trávy
- ***Impostor*** - třída obsahující metody a optimalizace pro vykreslení vegetace
- ***LandscapeGenerator*** - třída zapouzdřující práci s výškovou mapou a simulací ekosystému rostlin
- ***LParser*** - třída pro interpretaci L-systému a převod do 3D polygonální reprezentace
- ***Mapa*** - třída zapouzdřující práci s výškovou mapou, řekami, jezery a stromy (obsahuje *LandscapeGenerator*)
- ***Object*** - objekt enginu pro zařazení do grafu scény
- ***Renderer*** - abstrakce vykreslovacích API
- ***Teren*** - třída starající se o vykreslení terénu, řek a jezer
- ***Water*** - třída starající se o před-počítanou texturu řek
- ***Window*** - abstrakce správy oken
- ***WinXPWindow*** - konkrétní správa oken pro WindowsXP
- ***XmlParser*** - načítání a parsování xml souborů

6.8 Testy a výkon

Na výsledné aplikaci je třeba hodnotit nejen čas generování jednotlivých komponentů terénu, ale i rychlost jejich vykreslování v závislosti na míře detailů a dohledové vzdálenosti. Rychlost vykreslování je ovlivněna uživatelským nastavením a může se pohybovat od několika milisekund až po vteřiny. Časy generování jsou uvedeny v tabulce 1, časy zobrazování v tabulce 2.

Výsledky byly naměřeny na testovací sestavě v konfiguraci: AMD Athlon™ XP 2200+, 1.5 GB RAM, nVidia GeForce 6600 GT, pro výškovou mapu velikosti 256*256.

Metoda/fáze generování	**Doba výpočtu**
Fault formation	26 - 33 ms
FIR filter	3 - 4 ms
Thermal eroze (sesuv půdy)	10 - 12 ms
Vodní eroze	1778 - 1782 ms
Výpočet řek	25 - 27 ms
Výpočet jezer	17 - 27 ms
Umístění 50 000 stromů	79 - 81 ms
Simulace ekosystému (1 krok)	442 ms (50k stromů) - 1884 ms (100k stromů)

Tabulka 1: Naměřené časy potřebné k vygenerování komponentů mapy.

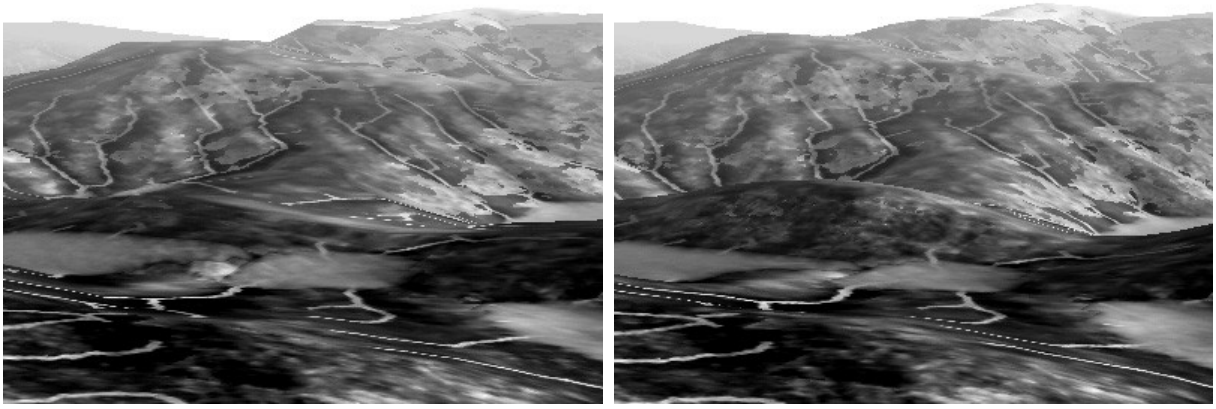
Úroveň detailů rostlin	**Doba vykreslení**
lod 1.0	250 ms
lod 1.33	313 ms
lod 1.66	391 ms
lod 2.0	501 ms

Tabulka 2: Naměřených časy potřebné k vykreslení rostlin.

Úroveň detailů terénu	**Doba vykreslení**
lod 10	44 ms
lod 32	54 ms
lod 64	64 ms

Tabulka 3: Naměřených časy potřebné k vykreslení terénu.

Obrázek 25: Srovnání úrovně detailů u stromů. Vlevo Lod = 1.0, vpravo Lod = 2.0



Obrázek 26: Srovnání úrovně detailů u terénu. Vlevo Lod = 10 (731 polygonů), vpravo Lod = 64 (12 818 polygonů). Je patrné, že díky před-počítaným texturám lokálních atributů se tolik neztrácejí při nižších Lod detaily.

Metriky kódu

Zdrojový text je rozdělen do několika souborů *.cpp a několika hlavičkových souborů *.h. Jejich velikosti, počty řádků a funkcí jsou uvedeny v tabulce 4.

Soubor	Funkcí	Řádků	Bytů	Soubor	Funkcí	Řádků	Bytů
XmlParser.h	5	38	985	DxRenderer.cpp	25	999	33621
App.h	26	435	15159	Engine.h	110	391	20015
XmlParser.cpp	2	102	2643	Types.h	21	279	11223
App.cpp	6	626	20563	Engine.cpp	17	441	14208
Grass.cpp	3	282	10405	Object.cpp	15	208	7989
Impostor.cpp	16	1884	76836	LandscapeGenerator.h	31	350	12727
L-system.cpp	4	376	14914	LandscapeGenerator.cpp	28	1841	60030
Mapa.cpp	3	151	4418	LParser.h	1	450	13732
Teren.cpp	11	1917	76290	LParser	56	3205	104859
Water.cpp	3	240	10491	Window.h	5	47	1451
Dx9Mesh.h	3	138	3951	WinXPWindow.h	2	65	1816
DxRenderer.h	25	120	7409	WinXPWindow.cpp	6	177	5121
Dx9Mesh.cpp	15	555	17732	**CELKEM**	**439**	**15317**	**548588**

Tabulka 4: Metriky kódu

6.9 Ovládání programu

Aplikace se spustí souborem *app.exe*. Jako parametr lze uvést preferované nastavení (preset), tedy například *app.exe setup.xml* a tím je umožněno uživateli vytvořit několik nastavení a jednoduše si z nich poté vybírat. Po spuštění jsou načteny soubory rostlin ze složky Plants a zahájeno generování terénu, poté výpočet řek a jezer a nakonec rozmístění a simulace ekosystému rostlin. O těchto událostech je uživatel informován prostřednictvím konzole.

Po dokončení přípravy terénu a provedení rozmístění rostlin a přípravy jejich polygonální reprezentace se otevře okno aplikace, do kterého se interaktivně vykresluje terén s vegetací. Uživatel se pohybuje pomocí kláves a rozhlíží pomocí myši se stisknutým pravým tlačítkem. Volbou F1 lze zobrazit nápověda, přehled informací o výkonu, stupních detailů, spotřebované paměti a možnostech ovládání. K použití jsou následující klávesy:

- **F1** ... Zobrazení / skrytí nápovědy a testovacích informací
- **←, →, ↑, ↓** ... Pohyb uživatele
- **Tlačítko myši** + **pohyb** ... Otáčení kamerou
- **J**, **K** ... snížení resp. zvýšení času pro vykreslování terénu (pokud je nastaven adaptivní lod pro terén)
- **N**, **M** ... snížení resp. zvýšení času pro vykreslování impostorů (pokud je nastaven adaptivní lod pro rostliny)
- **W** ... zobrazení polygonálního modelu terénu
- **O**, **L** ... zapnutí/ vypnutí křížové aproximace a optimalizace
- **F** ... zapnutí / vypnutí optimalizace ořezávání podle zorného pole

7 Závěr

Tato práce poskytuje široký pohled na problematiku generování a zobrazování terénu s rostlinami a přináší podrobné popisy metod vhodných pro řešení těchto problémů, především s důrazem na realistický vzhled a variabilitu, ale zároveň zachování dostatečné rychlosti pro využití v interaktivních aplikacích. Popisuje několik optimalizačních přístupů spolu s algoritmy pro větší urychlení generování a vykreslování.

Tato práce obsahuje novou robustní metodu pro výpočet řek a jezer s ohledem na již generovaný terén. Definovaná metoda tak úzce souvisí se vstupními výškovými daty a není třeba opačného postupu, kdy se nejprve vytvořil systém řek a poté se teprve vytvářel okolní terén. Algoritmus je variabilní a nabízí i možnosti rozšíření (např o nehomogenní dešťové srážky, zpětné ovlivnění výškové mapy terénu, apod.)

Zároveň zavádí novou zjednodušenou metodu pro hardware geomorphing s využitím pouze dvou předpřipravených doplňkových parametrů pro vrcholy a tím docílení větší rychlosti při výpočtu v grafických kartách. Navíc přináší úpravu metody optimalizace vykreslování vegetace impostory, do které zavádí (po existující dynamické změně rozlišení impostoru) další optimalizaci v podobě shlukování impostorů. Také definuje rychlou metodu pro vykreslování řek a jezer, která nevyžaduje přidání dalších polygonů do modelu terénu. U řek, jezer, terénu i stromů přináší práce některé úpravy a vylepšení pro real-time vykreslování na grafických kartách.

Nakonec je práce doplněna vyhodnocením implementovaných algoritmů, přehledem jejich výpočetní náročnosti a detaily o demonstrační aplikaci, která uvedené postupy prezentuje a zároveň slouží jako prostor pro experimentování s parametry uvedených metod. Přínosem je i možnost použití uvedených algoritmů v jiných aplikacích pomocí dynamických knihoven.

Tato práce otevírá některé možnosti pro budoucí vývoj. Nabízí například metody pro generování terénu a rostlin pomocí algoritmů, které se inspirují přírodními pochody. Hloubka jejich zkoumání je však velmi malá a např. jediný materiál terénu popsaný výškovou mapou nepředstavuje dostatečný popis ani pro sesuv půdy, natož pro vodní erozi. Tato práce se nesnaží stavět na roveň specializovaným botanických či geologických aplikacím a svým zaměřením směřuje čistě do oblastí her a výukových simulací. Nabízí metody, které jsou schopny vytvářet vizuálně přijatelný terén s rostlinami a díky jednodušší simulaci to provádí v krátkém čase.

Další možné rozšíření je doplnění některých zpětných vazeb. Popsaný systém je čistě dopřednou architekturou. Pro lepší simulaci pochodů v simulované krajině je možné práci rozšířit o zpětné vazby, díky kterým např. distribuce rostlin ovlivní sesuv půdy (zarostlý břeh je odolnější

k zvětrávání i k erozi vodou). Nebo řeky radikálně zvýší míru a rychlost eroze v místech, kudy protékají.

V neposlední řadě se dá práce rozšířit v růstu rostlin, který v popsaných metodách nebere v potaz distribuci světla (např. stromy v lesích mívají koruny výše, zatímco osamocené stromy jsou nižší s větvemi téměř k zemi).

Celkově však práce nabízí ucelený text, který čtenáře provede od prvních fází tvorby virtuální krajiny s vegetací, až po její realistickou a rychlou vizualizaci na moderních grafických procesorech.

Literatura

[1] Canny, J.F.: A computational approach to edge detection. IEEE: 679-698, 1986

[2] Prusinkiewicz, P., Hammel, M., A Fractal Model of Mountains with Rivers, From Proceeding of Graphics Interface ’93, pages 174–180, 1993

[3] Beneš, B., *A Stable Modeling of Large Plant Ecosystems*. In Proceedings of the International Conference on Computer Vision and Graphics, pages 94–101. Associoation for Image Processing, 2002

[4] Clyde, D., *Adding Realistic Rivers to Random Terrain*, dostupné z www: <http://www.gamedev.net/reference/articles/article2065.asp>

[5] *Adobe Photoshop*, digital image editor, dostupné z www 6.1.2008: <http://www.adobe.com/>

[6] Prusinkiewicz, P., *Algorithmic Botany*, the website of the Biological Modeling and Visualization research group, dostupné z www 2.1.2008: <http://algorithmicbotany.org/>

[7] Floyd, R. W., Steinberg, L., *An adaptive algorithm for spatial grey scale*, 1976

[8] Lluch, J., Camahort, E., Vivó, R., *An Image-Based Multiresolution Model for Interactive Foliage Rendering*, Programa de Incentivo a la Investigación 2003

[9] *Autodesk 3ds Max*, 3D modelling, animation and rendering solution, dostupné z www 6.1.2008 : <http://usa.autodesk.com>

[10] *Bounding Sphere*, Wikipedia, Free encyklopedia, dostupné z www 6.1.2008: <http://en.wikipedia.org/wiki/Bounding_sphere>

[11] Dutch, S., *Erosion and Landscape evolution*, Natural and Applied Sciences, 1997, dostupné z www 2.1.2008: <http://www.uwgb.edu/DutchS/EarthSC202Notes/erosion.htm>

[12] Boer, W. H. de, *Fast Terrain Rendering Using Geometrical MipMapping*, E-mersion Project, *2000*

[13] *Fractint L-Systems Definition, Fractint L-System Plants*, dostupné z www 2.1.2008: <http://spanky.triumf.ca/www/fractint/lsys>

[14] Shearer, P., *Fun with erosion*, Introduction to SIO Computing, 2005

[15] *Game programming gems*, volume 1-4, Charler River Media 2002-2007

[16] Lane, B., Prusinkiewicz, P., *Generating Spatial Distribution for Multilevel Models of Plant Communities*. In Proceedings of Graphics Interface'02, volume I, 2002

[17] *Google earth*, 3D Earth model viewer, dostupné z www 6.1.2008: <http://earth.google.com/intl/cs/>

[18] Tucker, G. E., Bras, R. L., *Hillslope processes, drainage density, and landscape morphology*, Water Resources Research, 1998

[19] Lapre L., *Lparser*, dostupné z www 2.1.2008 : <http://members.ziggo.nl/laurens.lapre/lparser.html>

[20] Hanan, J. S., Saskatchewan, R., *Parametric l-systems and their application to the modelling and visualization of plants*, June 1992

[21] Perlin,K., *Perlin Noise*, dostupné z www 2.1.2008: <http://en.wikipedia.org/wiki/Perlin_noise>

[22] *PipesGS sample*, Microsoft, dostupné z www 22.4.2009 <http://msdn.microsoft.com/en-us/library/bb205332(VS.85).aspx>

[23] Skjermo, J., Eidheim, O. Ch., *Polygon Mesh Generation of Branching Structures*, SCIA 2005, LNCS 3540, pp. 750–759, 2005

[24] Finkel, R., Bentley, J. L., Quad Trees: A Data Structure for Retrieval on Composite Keys, Acta informatica 4 (1): 1-9, 1974

[25] Fournier, A. D., Fussel, Carpenter, *Random Midpoint Displacement Method*, Computer Rendering of Stochastic Models , 1982

[26] Beneš, B., *Real-Time Erosion Using ShallowWater Simulation*, 4*th* Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS", 2007

[27] Deussen, O., Hanrahan, P., Lintermann, B., Mech, R., Pharr, M., Prusinkiewicz, P., *Realistic modeling and rendering of plant ecosystems*. In Proceedings of SIGGRAPH'98, Annual Conference Series 1998, pages 275.286, 1998

[28] Olsen, J., *Realtime Procedural Terrain Generation*, Department of Mathematics And Computer Science (IMADA), 2004

[29] Bornik, A., Reitinger, B., Beichel, R., *Simplex-Mesh based Surface Reconstruction and Representation of Tubular Structures*, Austrian Science Foundation (FWF), 2004

[30] *Simplified Terrain Using Interlocking Tiles*, Game Programming Gems 2, p. 377-383

[31] Kelley, A. D., Malin, M. C., Nielson., G. M., *Terrain simulation using a model of stream erosion*. Computer Graphics (SIGGRAPH 88 Proceedings), 22(4):263–268, 1988

[32] Wagner, D., Terrain Geomorphing in the Vertex Shader, Appeared in Shader-X 2, 2001

[33] Prusinkiewicz, P., Lindenmayer, A., *The algorithmic beauty of plants*. Springer-Verlag, New York, With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer. 1990

[34] *The Hunter*, 3D hunting game, dostupné z www 16.5.2008: <http://www.thehunter.com>

[35] Lluch, J., Vicent, M. J., Fernandes, S., Minserrat, C., Vivo, R., *The Modelling of Branched Structures Using a Single Polygonal Mesh*, TIC1999-0510-C02-01, 1999

[36] Musgrave, F. K., Kolb, C. E., Mace, R. S., *The synthesis and rendering of eroded fractal terrains*. Computer Graphics (SIGGRAPH 89 Proceedings), 23(3):41–50, 1989

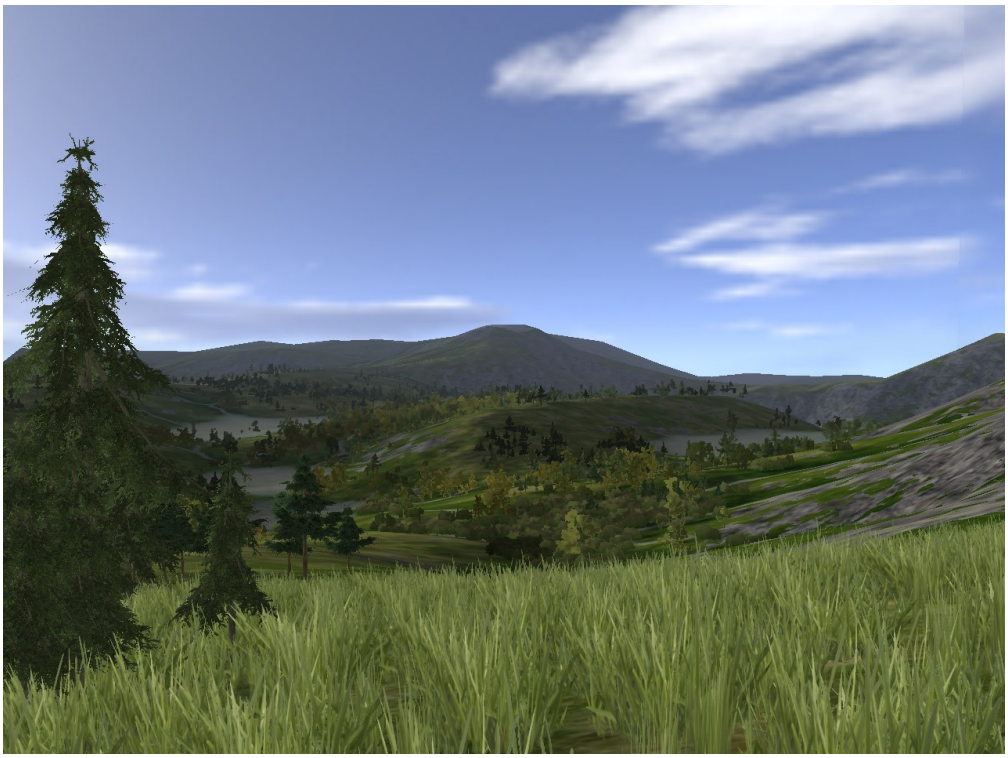
[37] Remolar, I., Chover, M., Ribelles, J., Belmonte, O., *View-Dependent Multiresolution Model for Foliage*, Journal of WSCG, Vol.11, No.1., ISSN 1213-6972 WSCG'2003, 2003

[38] Beneš, B., *Visual Model of Plant Development with Respect to Influence of Light*, Prague, 1997

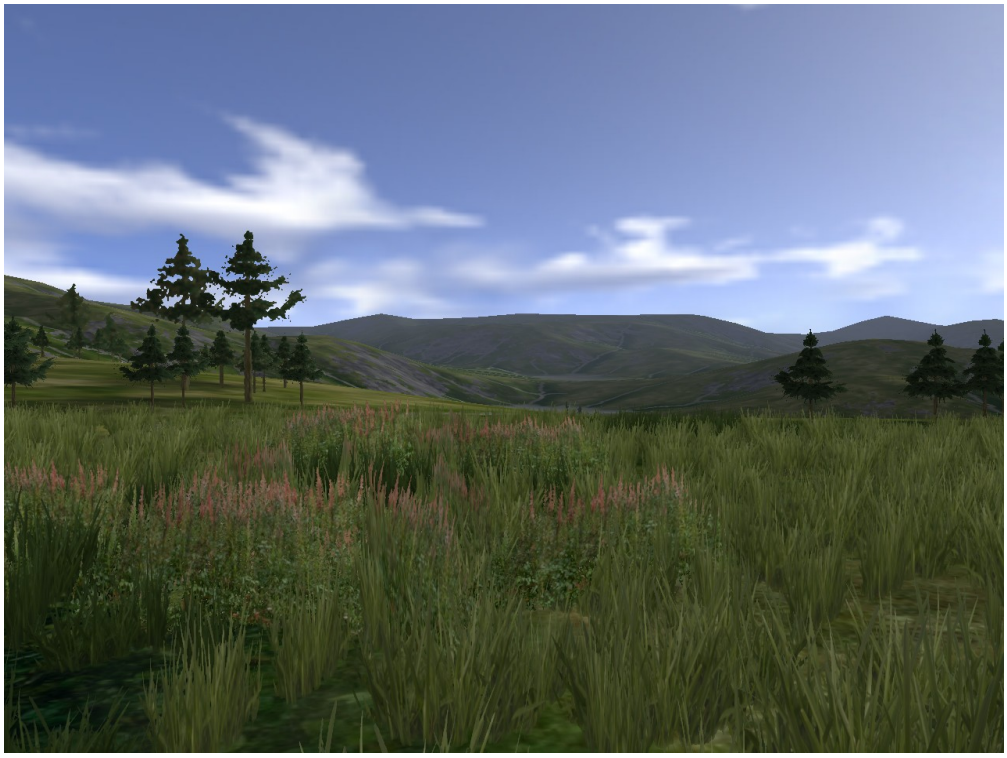
[39] Beneš, B., Forsbach, R., Visual Simulation of Hydraulic Erosion. Department of Computer Science, ITESM, Campus Ciudad de M´exico

Seznam příloh

Obrazové přílohy

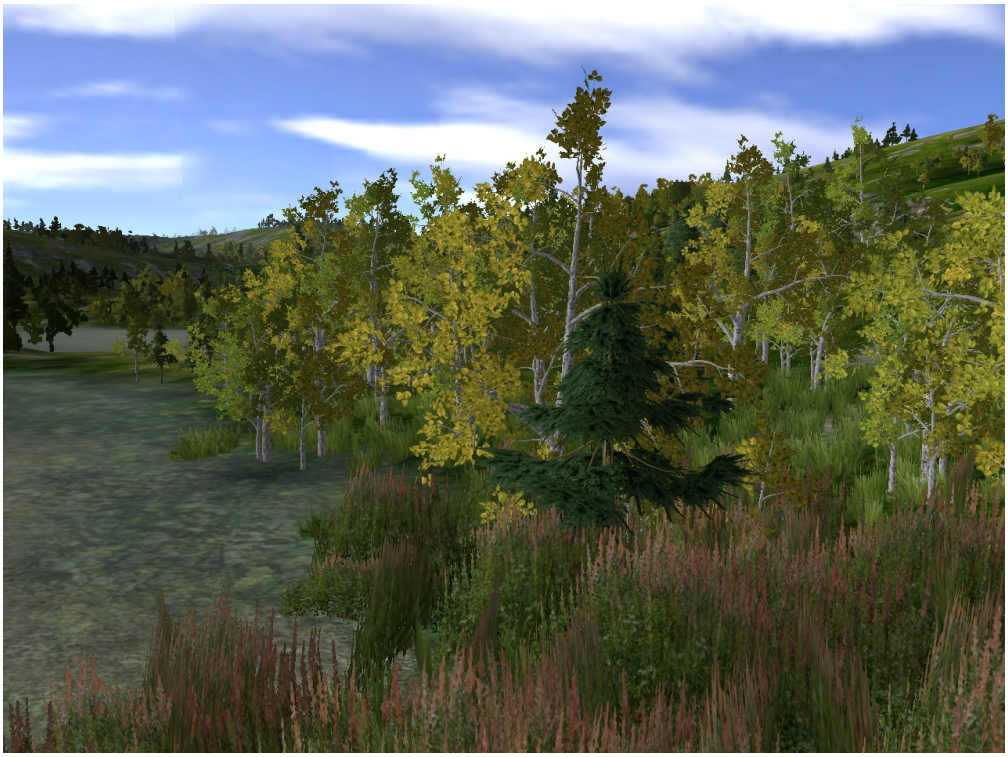


Obrázek 27: Ukázka real-time vykreslování vygenerovaného terénu s vegetací v demonstrační aplikaci.



Obrázek 28: Další ukázka z demonstrační aplikace.

Obrázek 29: Bližší pohled na fraktální stromy. Jejich těla ovlivňují i mutace.



Obrázek 30: Ve vlhčích místech se daří rostlinám lépe.

Obrázek 31: Zobrazení bez stromů. Zde je lépe vidět systém řek.

Obrázek 32: Další pohled na terén z velké výšky. Zde je vidět množství stromů, které se může vykreslovat.