Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

DIPLOMOVÁ PRÁCE

Plzeň, 2009

Zdeněk Prokop

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Simulace povětrnostních vlivů na výškové mapy

Plzeň, 2009 Zdeněk Prokop

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne …… Zdeněk Prokop, ……

Abstrakt

Simulation of climatic influences on height fields

This diploma thesis is focused on a simulation of erosion. Information about some erosion techniques has been gathered and these techniques are described in detail. Library with chosen erosion methods was created. This library contents classes for terrain creation, terrain representation and terrain visualization. This work also includes a program for library demonstration. I present here a new method for visual wind erosion.

Obsah

1. Úvod ... 1
2. Základní datové struktury ... 3
3. Existující metody ... 6
3.1. Tepelná eroze ... 7
3.1.1. F. Kenton Musgrave, Craig E. Kolb a Robert S. Mace ... 7
3.1.2. Bedřich Beneš, Rafael Forsbach ... 9
3.1.3. Jacob Olsen ... 9
3.2. Vodní eroze ... 11
3.2.1. F. Kenton Musgrave, Craig E. Kolb a Robert S. Mace ... 11
3.2.2. Bedřich Beneš, Rafael Forsbach ... 12
3.2.3. B. Neidhold, M. Wacker, O. Deussen ... 16
3.2.4. X. Mei, P. Decaudin, B.-G. Hu ... 20
3.3. Větrná eroze ... 24
3.3.1. Bedřich Beneš, Toney Roa ... 25
3.4. Shrnutí ... 27
4. Implementace ... 29
4.1. Datové struktury ... 30
4.2. Vytváření terénu ... 32
4.3. Reprezentace terénu a erozní metody ... 33
4.3.1. Tepelná eroze ... 34
4.3.2. Vodní eroze ... 35
4.3.3. Větrná eroze ... 38
4.4. Vizualizace terénu ... 42
5. Testování ... 45
6. Závěr ... 49
Literatura ... 50
Příloha ... 52

1. Úvod

Modelování realisticky vypadajícího terénu je předmětem zkoumání počítačové grafiky již spoustu let. Vzniklo mnoho různých algoritmů, jak terén generovat, téměř vždy se však skládají ze dvou kroků. Nejprve se vygeneruje hrubý tvar krajiny a poté se upraví různými erozními algoritmy.

Pro generování hrubého tvaru krajiny se poměrně často využívá fraktální geometrie, například náhodná procházka (*random walk*), která vychází z Brownova pohybu, dále metoda náhodného přesouvání středního bodu (*random midpoint dispacement*), jejíž podstata spočívá v rekurzivním dělení úsečky na polovinu a posouvání prostředního bodu o nějaké náhodné číslo, nebo metoda náhodných poruch (*random faults metod*), kde se v cyklu provádí rozdělení mapy na dvě části (nejčastěji přetnutím přímkou) a následné vyvýšení jedné části a snížení druhé o nějakou stejnou náhodnou hodnotu. Samozřejmě existuje i spousta dalších metod. Podrobnější informace o zmíněných metodách a odkazy na další naleznete v knize [ZBSF04].

Formování skutečného terénu je ale ovlivňováno obrovským množstvím klimatických vlivů, jako jsou vítr, voda, teplota nebo různé chemické procesy. Proto se pro dodání vyšší věrohodnosti generované krajině používají na vygenerovaný terén erozní algoritmy. Právě těmito algoritmy se zabývá tato práce. Erozní algoritmy lze rozdělit do tří základních kategorií – tepelná, vodní a větrná eroze.

Tepelná eroze (termoeroze) je proces, při kterém vlivem velkých teplotních rozdílů dochází k opadávání částeček materiálu, čímž dochází k vyhlazování povrchu a tvorbě svahů, které mají určitý sklon závisejícím na vlastnostech erodovaného materiálu, na úpatí hor.

Vodní erozi můžeme rozdělit na dva druhy – globální a lokální. Globální vodní eroze je způsobená deštěm, kdy dopadem dešťových kapek dochází k oddělování malých půdních částic. Pokud množství srážek převýší infiltraci půdy, dojde ke spláchnutí částeček půdy tekoucí vodou z vyšších míst do nižších. Lokální vodní eroze je způsobena proudící vodou, která způsobuje drolení částeček materiálu a může se projevovat jako vyschlé řečiště, meandry nebo delta.

Větrnou erozi lze rozdělit na dvě fáze. První je obrus způsobený třením větrem transportovaného materiálu, je závislý na síle a úhlu dopadajícího větru a na množství a hrubosti unášeného materiálu. Druhou je pak odnos sypkého zvětralého materiálu. Větrná eroze se z těchto tří typů erozí používá v počítačové grafice nejméně, slouží především pro modelování písečných dun.

Cílem této práce bylo prostudovat a porovnat existující erozní algoritmy, z nich vybrat podmnožinu s co nejširším spektrem pokrytí a vytvořit modulární programové vybavení. Programové vybavení a vybrané metody byly implementovány v prostředí .NET, konkrétně v jazyce C#.

2. Základní datové struktury

Před tím, než začneme s popisem a podrobnějším rozebíráním jednotlivých erozních algoritmů, by asi bylo dobré se zmínit o základních datových strukturách, které se používají pro reprezentaci terénu a se kterými pak tedy musí každý erozní algoritmus pracovat.

Asi nejčastěji používanou datovou strukturou pro reprezentaci modelovaného terénu je pravidelné výškové pole (*regular heigh field*) někdy také označované jako výšková mapa. V podstatě se jedná o dvourozměrné pole, jehož prvky představují konkrétní výšku na dané pozici, odkud vznikl název. Tato reprezentace má několik výhod, ale také spoustu nevýhod. Asi největší výhodou výškových map je jejich malá náročnost na paměť, naopak velkou nevýhodou je, že tato struktura není plně trojrozměrná, umožňuje reprezentovat pouze spojitou plochu, proto se také někdy označuje jako 2,5 rozměrná. Pomocí výškových polí není například možné vytvořit různé jeskyně, skalní převisy apod. Pro zobrazení se obvykle tato pole převádějí na sítě trojúhelníků. Z pohledu erozních algoritmů má tato struktura výhodu v tom, že algoritmy s ní pracující jsou velmi rychlé oproti jiným datovým strukturám. Na druhou stranu se ale zase předpokládá, že se buď celý terén skládá z jednoho homogenního materiálu, nebo každý prvek ještě dále obsahuje celou řadu další parametrů, jako například množství zde držené vody, schopnost materiál rozpouštět se ve vodě, množství minerálů, sílí atd., a předpokládá se, že se z jednoho homogenního materiálu skládá celý konkrétní sloupec terénu. Obě možnosti ale přinášejí značné omezení, protože se ztrácí informace o podložních strukturách daného materiálu.

Další možnou reprezentací terénu jsou voxely, což jsou prvky pravidelné třídimenzionální mřížky v prostoru. Voxely jsou obdobou pixelů v dvourozměrné grafice. Umožňují reprezentaci plně trojrozměrných terénů, mohou zachytit libovolné povrchy a členitosti. Každý voxel představuje malou 3D krychličku, která si udržuje informace o určitých svých vlastnostech podobným těm z výškových polí – druh materiálu, množství držené vody, odolnost v prostředí atd. Techniky založené na voxelech dávají lepší výsledky a mají vyšší přesnost. Jsou velmi silným nástrojem, protože umožňují náhlé změny struktury. Kvůli tomu jsou ale také velmi pomalé. Také veliká datová náročnost této datové struktury a její pomalé zobrazování jsou značnými nevýhodami.

Terén lze reprezentovat také pomocí sítě trojúhelníků. Takováto reprezentace umožňuje vyjádřit plně trojrozměrné objekty, ale pouze jako plochy, o vnitřní struktuře se nic nedozvíme. Největší výhodou této reprezentace je bezpochyby rychlost jejího zobrazování,

existují dokonce adaptivní zobrazovací algoritmy, které umožňují podle zadané úrovně detailu dynamicky vygenerovat zjednodušené sítě trojúhelníků (viz [ZBSF04]).

Jak jsem se už výše zmínil, výhodou výškových polí je jejich malá datová náročnost a velká rychlost algoritmů s nimi pracujícími, nevýhodou pak, že neumožňují uchovávat informace o podložních strukturách. Na druhou stranu voxelová reprezentace informace o podložních strukturách uchovávat umí, ale je příliš náročná na paměťové prostředky, práce s ní je pomalá a taktéž je pomalé její zobrazování. Beneš a Forsbach proto v [BF01] navrhli tzv. vrstvenou datovou reprezentaci, která je kompromisem mezi výškovými poli a voxelovými reprezentacemi. Inspirovali se geologickými průzkumy půdy. Podzemní struktura půdy se skládá z různých vrstev materiálů o různé tloušťce, která je ale obvykle poměrně veliká a pro reprezentaci pomocí voxelů je to plýtvání daty. Proto se rozhodli použít nějaký druh intervalové struktury, kterou lze chápat jako RLE kompresi sloupců voxelové struktury. Konkrétně vstupní terén uložili do dvourozměrného pole, jehož každý prvek se skládá z výšky povrchu terénu na daných souřadnicích a jednorozměrného pole, které obsahuje tloušťky jednotlivých podložních vrstev a informace o materiálech, ze kterých jsou tyto vrstvy složeny. Struktura tohoto jednoho prvku pole vypadá podle [BF01] takto:

```
Typedef struct {
   PropertiesT data [MAX_LEVEL];
   float height;
} ElmT; //one element of the array
```

MAX_LEVEL je konstanta určující velikost pole a tím pádem také omezuje maximální počet vrstev. Pokud bychom se chtěli tomuto omezení vyhnout, můžeme použít místo pole spojový seznam. V každém prvku pole *data* se uchovávají všechny parametry daného materiálu potřebné pro erozi, jako jsou například hustota materiálu, jeho nasákavost, schopnost se usazovat a další.

Můžeme si všimnout, že pokud by byl maximální počet vrstev této datové struktury omezen pouze na jednu, získáme tím klasickou výškovou mapu. Proto můžeme brát tuto strukturu jenom jakési rozšíření výškové mapy, které nám umožní díky informaci o podložní struktuře půdy terén lépe erodovat, proto jsem si tuto strukturu vybral pro reprezentaci terénu, se kterým budu dále pracovat a nadále při řešení budu uvažovat pouze ji.

Beneš a Forsbach ovšem tuto svoji strukturu ještě dále doplnili o to, že jednotlivé vrstvy se mohou skládat nejen ze zeminy, ale také z vody nebo plynu, což umožňuje reprezentovat

jeskyně, převisy nebo díry, což je další výhoda této reprezentace zděděná z voxelů. Mým úkolem ale bylo zabývat se erozními algoritmy výškových map, které tyto útvary zobrazit neumožňují, proto jsem se tímto vylepšením příliš nezabýval a předpokládal jsem, že se jednotlivé vrstvy budou skládat pouze z pevných materiálů.

3. Existující metody

Modelování terénu je předmětem počítačové grafiky už dlouhou dobu. Zpočátku se terén generoval hlavně pomocí metod využívajících fraktály. Například v roce 1988 publikovali Kelley a kolektiv v článku [KMN88] metodu, která využívá fraktálovou interpolaci ke spojení předdefinovaných řek a údolí. Na tuto metodu později navázali i Belhadj a Audiber v [BA05].

Terény vygenerované pomocí fraktálních metod ale měly tu nevýhodu, že pokud bychom je otočily tak, aby vrcholky hor tvořily údolí a naopak údolí tvořila vrcholky hor, terén by vypadal pořád stejně. Ve skutečné přírodě takovouto vlastnost mají možná jen velmi mladá pohoří, u většiny ostatních hor se ale tvar údolí a tvar vrcholků hor podstatně liší a to díky působení erozních procesů.

Pravděpodobně asi první algoritmus pro vizuální simulaci eroze představil Musgrave a kolektiv v roce 1989 v [MKM89], kde uvedl algoritmy pro tepelnou a vodní erozi, které použil na terén vygenerovaný pomocí fraktálové techniky. Oběma těmto algoritmům se budeme podrobněji věnovat dále v kapitolách 3.1.1. a 3.2.1.

Algoritmus tepelné eroze pak v roce 2001 dále rozšířili Beneš a Forsbach v [BF01] tak, že původní algoritmus pro výškovou mapu upravili pro svojí vlastní datovou strukturu, která dokáže uchovávat informace i o podložních vrstvách terénu. Jejich metoda je popsána v kapitole 3.1.2.

Kapitola 3.1.3. popisuje optimalizaci algoritmu tepelné eroze, kterou navrhnul v roce 2004 Jacob Olsen v [Ols04].

I na algoritmus vodní eroze z [MKM89] Beneš s Forsbachem navázali. A to v roce 2002 v článku [BF02], kde nově přidali k procesu eroze i krok vypařování a celý proces rozdělili na kroky, které jsou na sobě nezávislé a umožňují tak paralelní zpracovávání. Takto upravený algoritmus je popsán v kapitole 3.2.2.

Vodní erozi založenou na silách, kterými působí tekoucí voda, popsal v roce 1998 Chiba s kolektivem v článku [CMF98]. Na tuto práci navázala spousta dalších autorů, například B. Neidhold, M. Wacker a O. Deussen v článku [NWD05] z roku 2005 tuto erozní metodu rozšířili o vrstvenou datovou strukturu a také tuto metodu zrychlili zjednodušením výpočtu rychlosti vody. Podrobnějším popisem této metody se zabývá kapitola 3.2.3. Dalšími, kdo navázali na [CMF98], byli v roce 2007 Xing Mei, Philippe Decaudin a Bao-Gang Hu, kteří v článku [MDH07] pro transport vody využili tzv. *pipe model* a výpočet eroze převedli z CPU na GPU, čímž ho také poměrně výrazně urychlili. I tato metoda je zde podrobněji popsána a to v kapitole 3.2.4. Implementace vodní eroze, která je založená na silách, pro GPU je

popsána také v [ASA07], nevyužívá ale pro transport vody *pipe model* a pracuje s trojrozměrným vektorem rychlostí vody.

Článek, který by se přímo zabýval větrnou erozí, jsem bohužel nenašel. Existuje ale pár takových, které se zabývají modelováním pouštních krajin, zejména pak písečných dun. Jedná se tedy spíše jen o přesun písku větrem, což je pouze součást větrné eroze. Chybí zde obrušování hornin tímto unášeným pískem, aby se jednalo o plnohodnotnou větrnou erozi.

Jedním z článků zabývajících se modelováním pouštní krajiny je [BR04]. Způsob simulace transportu písku v této práci je popsán v kapitole 3.3.1. Další práce, které se touto tématikou zabývají, jsou například [ON00] a [ON03].

Jak již bylo zmíněno v úvodu, erozní algoritmy můžeme rozdělit do tří základních skupin na tepelnou, vodní a větrnou erozi. Každou skupinou zvlášť se budou zabývat jednotlivé podkapitoly a v každé budou uvedeny některé již existující erozní algoritmy.

Erozní algoritmy mohou být popsány přesunem materiálu mezi jednotlivými prvky terénu. To vyjadřuje následující diferenciální rovnice:

$$Q_{in} - Q_{out} = \frac{dS}{dt} \qquad (3.1)$$

kde Q_{in} je celkové množství materiálu, které daný prvek získá od ostatních prvků, a Q_{out} je celkové množství materiálu, které daný prvek předá ostatním prvkům. S je objem materiálu uvnitř tohoto prvku a t je čas. Různé algoritmy používají různé přístupy k řešení této rovnice, tok materiálu pak závisí na jevech, které algoritmus simulují.

3.1. **Tepelná eroze**

Tepelná eroze je proces, při kterém vlivem velkých teplotních rozdílů dochází k opadávání částeček materiálu, čímž dochází k vyhlazování povrchu a tvorbě svahů, které mají určitý sklon závisejícím na vlastnostech erodovaného materiálu, na úpatí hor.

3.1.1. **F. Kenton Musgrave, Craig E. Kolb a Robert S. Mace**

Tato podkapitola vychází z článku [MKM89]. Algoritmus, který Musgrave a kolektiv navrhli, simuluje rozrušování materiálu, který poté padá a hromadí se na úpatí hor. Vytváří svahy se stejným sklonem, je rychlý a má jednoduchou implementaci.

Algoritmus funguje následovně: v každém časovém kroku $t + 1$ se porovnává rozdíl mezi výškou a_t^v z předchozího kroku t každého vrcholu v a každým sousedním vrcholem u

s (globálním) svahovým úhlem T (*talus angle*). Když je spočítaný svah větší než svahový úhel, přesune se nějaká fixní část c_t rozdílu na souseda. To vyjadřuje následující vztah:

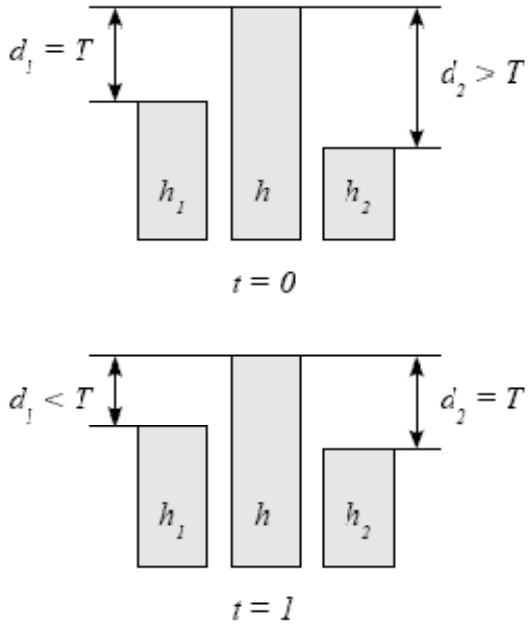
$$a_{t+1}^u = \begin{cases} a_t^v - a_t^u > T : a_t^u + c_t\left(a_t^v - a_t^u - T\right) \\ a_t^v - a_t^u \le T : a_t^u \end{cases} \qquad (3.2)$$

Erodovaný materiál je důležité rozmístit spravedlivě na všechny sousední prvky, které jsou menší, tedy podle poměrů rozdílů výšky erodovaného prvku a výšky sousedů. Výška souseda se pak spočítá podle následujícího vztahu:

$$a_{t+1}^u = a_t^u + c_t\left(d_{\max} - T\right) \cdot \frac{a_t^v - a_t^u}{d_{total}} \qquad (3.3)$$

kde d_{max} je rozdíl mezi výškou právě erodovaného prvku a výškou jeho nejnižšího souseda a d_{total} je celkový součet rozdílů výšky daného prvku a výšky jeho nižších sousedů.

Obrázek 3.1 znázorňuje zjednodušený příklad přesunu materiálu uvedený v [Ols04], materiál potřebuje přemístit pouze do jedné sousední buňky. V prvním kroku $d_1 = h - h_1 = a_0^v - a_0^1 = T$ a $d_2 = h - h_2 = a_0^v - a_0^2 > T$, to znamená, že materiál se bude přesouvat z a_0^v na a_0^2, pokud $c_t = 1$, bude $a_1^1 = h_1 = a_0^1$ a $a_1^2 = a_0^v - T$.



Obrázek 3.1: Příklad přesunu materiálu (obrázek pochází z [Ols04])

3.1.2. **Bedřich Beneš, Rafael Forsbach**

Bedřich Beneš a Rafael Forsbach naimplementovali v [BF01] předchozí algoritmus teplotní eroze, který byl navržen pro výškové mapy, pro jejich vrstvenou datovou strukturu, která byla popsána ve 2. kapitole. Tím ho obohatili o některé nové věci, které si nyní popíšeme.

Oproti původnímu algoritmu se liší výpočet gradientu, kdy se musí nejprve vypočítat celková výška všech vrstev v daném prvku a až pak ji porovnávat se sousedy. Při erozi daného prvku se množství materiálu k přenesení počítá z nejvrchnější vrstvy. Pokud se přenáší více materiálu, než vrstva obsahuje, nastavíme jí v případě reprezentace vrstev pomocí pole nulovou výšku, nebo při použití spojového seznamu ji odstraníme. Dále pokračujeme vrstvou, která se nachází pod touto odstraněnou vrstvou.

Erozi je možné aplikovat nejen na povrch terénu, ale také na podpovrchové jeskyně, u nichž umožňuje také simulovat padání materiálu ze stropu.

Oproti výškovým mapám také umožňuje nastavit vlastnosti erodovaného materiálu různé od vlastností původního materiálu. Obvykle neusazený materiál je totiž velmi hustý, když se ale eroduje, tak je pohyby změněn na prach a ten má úplně jiné vlastnosti, než původní materiál.

3.1.3. **Jacob Olsen**

Jacob Olsen vycházel z předchozího algoritmu tepelné eroze z podkapitoly 3.1.1. a v [Ols04] navrhl jeho optimalizaci. Původní implementace pracuje s hodnotami d_{max} a d_{total}, které se určují následujícím způsobem:

```
di = h - hi;
if (di > T)
{
     dtotal = dtotal + di;
     if (di > dmax)
          dmax = di;
}
```

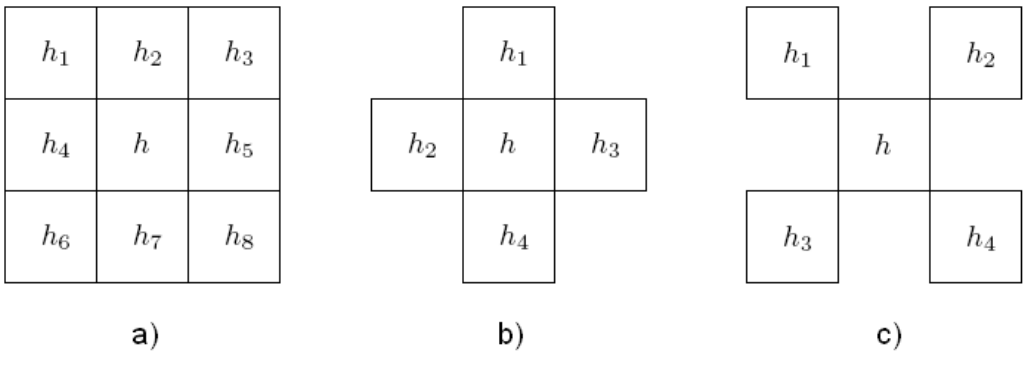
Pokud ve výše uvedeném postupu použijeme místo Mooreova okolí Von Neumannovo okolí, sníží se počet podmínek na polovinu. Protože je množství přenášeného materiálu úměrné d_{max}, přesune se díky Von Neumannově okolí stejné množství materiálu za poloviční čas.

Algoritmus může být dále urychlen tím, že velmi zjednodušíme přenos erodovaného materiálu. Místo toho, abychom přenášeli na každého nižšího souseda část materiálu úměrnou k $\frac{d_i}{d_{total}}$ (kde d_i je rozdíl výšky erodovaného prvku a výšky jeho i-tého souseda, $d_i < T$ a d_{total} je součet všech d_i větších než T), přeneseme všechen erodovaný materiál pouze na nejnižšího souseda. Nevýhodou tohoto zjednodušení je, že může být přenášeno méně materiálu na každý prvek. To je ale částečně kompenzováno tím, že se přenáší maximální množství materiálu, jaké je jen možné, tedy:

$$\Delta h = \frac{d_{\max}}{2} \qquad (3.4)$$

Přenesení tohoto množství způsobí to, že se výšky erodovaného prvku a jeho nejnižšího souseda vyrovnají. V původní implementaci z předchozí kapitoly se nepřesouvá celé množství materiálu určeného k přesunu, ale pouze jeho část, abychom se vyhnuli oscilacím. Podobný problém by nastal i zde, pokud by například čtyři vysoké hodnoty obklopovaly hlubokou díru, nedošlo by pouze k jejímu vyplnění, ale vytvořila by se zde vysoká špička. To je způsobeno tím, že výšková mapa zůstává nezměněna, dokud nejsou zpracovány všechny její prvky v daném čase. Proto, abychom se těmto oscilacím vyhnuli, musíme změny výškové mapy provádět okamžitě po dokončení výpočtu výšky každého konkrétního prvku, což nám i velmi mírně zvýší výpočetní rychlost.

Dále ještě Olsen zjistil při experimentech s různými druhy okolí, že upravená verze Von Neumannova okolí, zobrazená na obrázku 3.2 c), dává výsledky o trošku podobnější původní implementaci.



Obrázek 3.2: a) Mooreovo okolí, b) Von Neumannovo okolí, c) upravené Von Neumannovo okolí (obrázek pochází z [Ols04])

3.2. **Vodní eroze**

Vodní erozi můžeme rozdělit na dva druhy – globální a lokální. Globální vodní eroze je způsobená deštěm, kdy dopadem dešťových kapek dochází k oddělování malých půdních částic. Pokud množství srážek převýší infiltraci půdy, dojde ke spláchnutí částeček půdy tekoucí vodou z vyšších míst do nižších. Lokální vodní eroze je způsobena proudící vodou, která způsobuje drolení částeček materiálu a může se projevovat jako vyschlé řečiště, meandry nebo delta.

3.2.1. **F. Kenton Musgrave, Craig E. Kolb a Robert S. Mace**

Jak již jsem uvedl na začátku této kapitoly, Musgrave a kolektiv v článku [MKM89] navrhli jak algoritmus pro tepelnou erozi, tak i algoritmus pro vodní erozi. Tento algoritmus vodní eroze simuluje nános vody na prvky výškové mapy, což simuluje déšť. Síla eroze vody je funkce velikosti jejího objemu a množství právě unášeného sedimentu. Dále umožňuje pohyb vody i se zachyceným erodovaným materiálem na nižší sousední prvky.

Každý prvek v výškové mapy obsahuje výšku a_t, objem vody w_t a množství sedimentu rozpuštěného ve vodě s_t v čase t. V každém časovém kroku se přelije nadbytek vody se sedimentem z prvku v do každého souseda u. Množství vody, které se přelije, Δw je definováno takto:

$$\Delta w = \min\left(w_t^v, \left(w_t^v + a_t^v\right) - \left(w_t^u + a_t^u\right)\right) \qquad (3.5)$$

Pokud $\Delta w \leq 0$ usadí se část sedimentu rozpuštěného ve vodě v prvku v:

$$\begin{aligned} a_{t+1}^v &= a_t^v + K_d s_t^v \\ s_{t+1}^v &= (1 - K_d) s_t^v \end{aligned} \qquad (3.6)$$

jinak:

$$\begin{aligned} w_{t+1}^v &= w_t^v - \Delta w \\ w_{t+1}^u &= w_t^u + \Delta w \\ c_s &= K_c \Delta w \end{aligned} \qquad (3.7)$$

kde c_s je kapacita přemisťované vody Δw, konstanta K_c je kapacita sedimentu, která určuje maximální množství sedimentu rozpustné v jednotce vody, a K_d je konstanta nánosu, která rychlost, s jakou se rozpuštěný sediment usazuje.

Když se přelije sediment z v do u, odstraníme nejvýše toto množství sedimentu z s_t^v a přidáme ho do s_{t+1}^u. Když je c_s větší než s_t^v, část rozdílu je odečtena od a_t^v a je přidána k s_{t+1}^u,

což představuje erozi zeminy z bodu *v*. Nakonec se část sedimentu zbývající v prvku *v* usadí tak, jak již bylo uvedeno výše. Tedy pokud $s_t^v \geq c_s$, provedeme tyto operace:

$$\begin{aligned} s_{t+1}^u &= s_t^u + c_s \\ a_{t+1}^v &= a_t^v + K_d\left(s_t^v - c_s\right) \\ s_{t+1}^v &= \left(1 - K_d\right)\left(s_t^v - c_s\right) \end{aligned} \qquad (3.8)$$

jinak:

$$\begin{aligned} s_{t+1}^u &= s_t^u + s_t^v + K_s\left(c_s - s_t^v\right) \\ a_{t+1}^v &= a_t^v - K_s\left(c_s - s_t^v\right) \\ s_{t+1}^v &= 0 \end{aligned} \qquad (3.9)$$

K_s je konstanta, která určuje měkkost zeminy a umožňuje tak ovládání rychlosti, jakou je zemina přeměňována na sediment. Usazený sediment je přičten k výšce daného prvku.

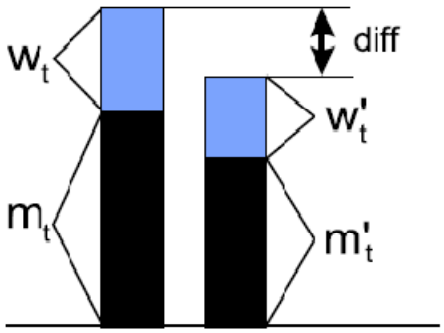
Během výše popsaného procesu se voda s rozpuštěnou zeminou přemisťuje z vyšších poloh terénu do nižších, kde se postupně usazuje. V plně dvourozměrné implementaci musíme stejně jako u tepelné eroze v podkapitole 3.1.1. distribuovat vodu s rozpuštěným sedimentem do všech nižších sousedních prvků v takovém poměru, který je úměrný rozdílu výšek.

3.2.2. **Bedřich Beneš, Rafael Forsbach**

Bedřich Beneš a Rafael Forsbach představili v článku [BF02] algoritmus vodní eroze, který se skládá ze čtyř různých, na sobě nezávislých kroků, což může být značná výhoda například při paralelizaci. Například v předešlé implementaci z podkapitoly 3.1.1. byly tyto procesy simulovány jako související, kdy usazování materiálu bylo funkcí přenosu vody.

Prvním krokem je výskyt nové vody, kdy se voda objeví na nějakém místě nebo místech například vlivem deště nebo z nějakého vodního zdroje. Druhým krokem je, že voda rozrušuje terén pod sebou a zachytává jej. To může být způsobeno například prouděním vody nebo rozpouštěním. Třetím krokem je pak přemístění vody se zachyceným sedimentem podle vnitřních a vnějších sil, kdy asi nejdůležitějším faktorem přispívajícím k vodnímu toku je gravitace. Ačkoliv vnitřní síly hrají také důležitou roli, v tomto přístupu zachyceny nejsou. Nánosový proces, při kterém voda nechává unášený materiál v určitých místech, je závislý na dvou hlavních faktorech. Prvním je ten, že voda zpomaluje, takže těžké částečky materiálu nemohou být přenášeny na libovolné místo. Druhým faktorem je pak nadbytek objemu vodního sedimentu, který je způsoben vypařováním vody, což je poslední čtvrtý krok.

Pro implementaci algoritmu Beneš s Forsbachem použili vrstvenou strukturu dat (blíže je popsána ve 2. kapitole), ale pro zjednodušení popisují algoritmus na výškové mapě, proto si nyní také uvedeme tento popis algoritmu pouze pro výškovou mapu.



Obrázek 3.3: Zobrazení vrcholů a materiálu. Levý sloupec je prozkoumávaný vrchol a pravý je představitel sousedního vrcholu. (Obrázek pochází z [BF02])

Označme si tedy výšku materiálu na dané pozici písmenem m_t, objem vody tohoto prvku w_t a množství ve vodě rozpuštěné zeminy s_t v daném čase t. Čas $t + 1$ pak znamená libovolný časový krok $t + \Delta t$ (nebo jednoduše jiný stav systému), nikoliv zvýšení času o jednu sekundu nebo hodinu. Dále pro jednoduchost budeme předpokládat pouze jeden sousední prvek, jehož výšku materiálu označíme m'_t, objem vody w'_t a množství ve vodě rozpuštěné zeminy s'_t. Příklad značení je uveden na obrázku 3.3.

V prvním kroku, což je výskyt nové vody, předpokládáme, že se voda může objevit buď v případě vodního zdroje lokálně na nějaké pozici, anebo může postihnout velké oblasti terénu ve formě deště. V obou těchto případech je úroveň vody v prvku jednoduše zvětšena o koeficient K_r, který udává množství nové vody, které přibývá v jednom časovém kroku. To je popsáno následující rovnicí:

$$w_{t+1} = w_t + K_r \qquad (3.10)$$

K_r sice může být konstanta odpovídající konstantnímu přítoku vody, ale k dosažení lepších výsledků simulace by měl tento koeficient respektovat čas i pozici P v prostoru, tedy být funkcí $K_r(t, P)$. Například v přírodě odpovídá intenzita deště v daném bodě v závislosti n čase funkci ve tvaru zvonu.

Když se teď podíváme na čtvrtý krok, jímž je vypařování vody, zjistíme, že množství vypařené vody závisí na ploše vodní hladiny, která je pro reprezentaci výškovou mapou konstantní, a na teplotě, kterou budeme pokládat také za konstantní a bude vyjadřovat průměr teplot ve velkém časovém intervalu. Vypařování vody je popsáno rovnicí:

$$\frac{dw}{dt} = -K_e w \qquad (3.11)$$

kde K_e je koeficient vypařování a odpovídá rychlosti vypařování vody v daném časovém rozsahu. Z řešení rovnice (3.11) vyplývá, že množství vody v daném vrcholu závisí pouze na počátečním objemu vody w_0 a na čase t. Voda v čase ubývá exponenciálně. To vyjadřuje vztah (3.12).

$$w_t = w_0 e^{-K_e t} \qquad (3.12)$$

V praktické implementaci by úroveň vody nikdy nedosáhla úplné nuly, proto se zavádí prahová hodnota T. Když množství vody klesne pod tuto prahovou hodnotu, nastavíme ji na nulu, což nám umožní simulovat vysychání některých oblastí. Upravený vztah (3.12) má formu:

$$w_t = w_0 e^{-K_e t} - T \qquad (3.13)$$

Beneš a Forsbach se nezabývají erozí založenou na silách, ale pouze nánosem materiálu, který je ve vodě rozpuštěný. Také nesimulují usazování materiálu způsobené tím, že částečky materiálu dosáhnou dna a nemůžou být dále přenášeny, ale používají jinou techniku usazování.

Množství zeminy rozpuštěné v určitém objemu vody nemůže přesáhnout úroveň nasycení. Úroveň nasycení S_t je vyjádřena tímto vztahem:

$$S_t = K_s w_t \qquad (3.14)$$

kde K_s je koeficient nasycení v kilogramech na litr a říká, že litr vody může nést K_s kilogramů materiálu. Pokud je voda nasycena a dojde k vypaření nějakého objemu vody, úroveň nasycení se překročí, takže odpovídající množství materiálu musí být usazeno na dané pozici. Pomocí rovnice (3.14) zjistíme úroveň nasycení S_t, porovnáme ji s množstvím materiálu na dané pozici s_t, a pokud bude $s_t > S_t$, usadíme $s_t - S_t$. Výsledná výška pak tedy bude:

$$m_{t+1} = m_t + (s_t - S_t) \qquad (3.15)$$

Pro přemístění vody při pomalém pohybu můžeme použít jednoduchý difúzní model, ve kterém se pro každou pozici ve výškové mapě zjistí, u kterých prvků výška vody přesahuje sousedy a přesune se k sousedům, kteří jsou pod nimi.

Výška, která dosahuje úrovně vody v aktuální pozici, je součet všech přispívajících množství, tedy výšky materiálu a objemu vody:

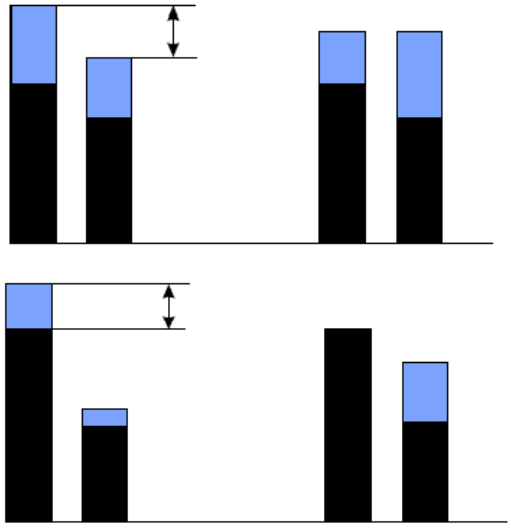
$$h = m + w \qquad (3.16)$$

samozřejmě pro výšku souseda obdobně platí:

$$h' = m' + w' \qquad (3.17)$$

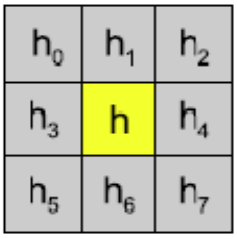
Pokud je rozdíl $h - h'$ menší než nula, aktuální prvek je v díře a musíme část nebo všechnu vodu přesunout.

Obrázek 3.4 ukazuje dva důležité případy, které mohou nastat při přesunu vody. V levé části je znázorněna situace před přesunem a v pravé části po přesunu vody. V horní části je naznačen případ, kdy je nutné přesunout pouze část vody, v dolní části je pak případ, kdy je potřeba přesunou z vyššího prvku veškerou vodu.



Obrázek 3.4: Dva případy pohybu vody. Levé kresby znázorňují situaci před a pravé po přesunu vody. (Obrázek pochází z [BF02])

V plně dvourozměrné implementaci musí být množství vody rozděleno úměrně k relativním rozdílům výšek nižších sousedních vrcholů. Označme výšku aktuálního vrcholu písmenem h a výšky sousedních vrcholů h_i pro $i = 0, 1, \ldots, 7$ (viz Obrázek 3.5).



Obrázek 3.5: Indexování sousedních vrcholů (Obrázek pochází z [BF02])

Množství vody, které by mělo být odstraněno Δw je rozděleno podle rovnice:

$$\Delta w_i = \Delta w \frac{d_i}{sum} \qquad (3.18)$$

kde Δw_i je množství vody, které se přesune do i-tého vrcholu, $d_i = h - h_i$ a zároveň $h_i < h$ a $sum = \sum d_i$.

Dosažení rovnováhy vody ještě neznamená, že je systém stabilní. Sousedící prvky mohou obsahovat úrovně rozpuštěných sedimentů, které se velmi liší. Ty by měli být také rozloženy rovnoměrně, proto se používá řešení, které je obdobou výše popsaného vodního přesunu, jen se rozděluje koncentrace sedimentu ve vodě.

3.2.3. B. Neidhold, M. Wacker, O. Deussen

Neidhold a kolektiv v [NWD05] popisují další algoritmus vodní eroze, který bere v potaz i rychlost proudící vody. Původní algoritmus vodní eroze založený na silách tekoucí vody poprvé představil Chiba v [CMF98]. Neidhold s kolektivem tento původní algoritmus pouze o něco rozšířili, především o vrstvenou datovou strukturu. Proto zde budu popisovat až tento rozšířený algoritmus místo toho původního.

Jak již bylo řečeno, algoritmus pracuje opět s vrstvenou datovou strukturou a data, která se uchovávají pro každý prvek mapy, jsou: trojrozměrný vektor zrychlení $\vec{a}$ a trojrozměrný vektor rychlosti vody $\vec{v}$, množství vody F v daném prvku, množství rozpuštěného materiálu S a výška terénu H.

Pro simulaci pohybu vody využívají zjednodušené Navier-Stokesovy rovnice. Výsledky jsou pak o něco méně fyzikálně přesné, ale výpočet je o mnoho rychlejší. Pohyb vody je závislý na rychlosti $\vec{v}$ a zrychlení $\vec{a}$ a rovnice mají následující tvar:

$$\begin{aligned} \dot{\vec{v}} &= \vec{a} - K_A.\vec{v} = \frac{\vec{F}}{m} - K_A.\vec{v} \\ \dot{\vec{x}} &= \vec{v} \end{aligned} \qquad (3.19)$$

Parametr $K_A \in \langle 0,1 \rangle$ umožňuje ovlivňovat velikost tření mezi vodou a materiálem. Například pokud použijeme K_A = 0,3, napodobujeme tím velmi tvrdý povrch, který rychlost klouzající vody snižuje o 30 procent za jednotku času.

Protože pracujeme s diskrétním časem, budeme rovnice (3.19) používat v následujícím tvaru:

$$\begin{aligned} \vec{v}_{t+\Delta t} &= \vec{v}_t + \vec{a}_t.\Delta t - K_A.\vec{v}_t.\Delta t \\ \vec{x}_{t+\Delta t} &= \vec{x}_t + \vec{v}_{t+\Delta t}.\Delta t \end{aligned} \qquad (3.20)$$

Nejprve musíme vypočítat vektor aktuálního zrychlení $\vec{a}_t$, jehož velikost je určena následujícím vzorcem:

$$|\vec{a}| = \sin\alpha.g \qquad (3.21)$$

Kde g je velikost gravitačního zrychlení ($g \approx 9{,}81\,\mathrm{m.s^{-2}}$) a α je úhel největšího spádu. Směr zrychlení získáme pomocí gradientu $\nabla I(x, y)$, který určuje směr největšího spádu.

$$\nabla I(x,y) = \left(\frac{\Delta I}{\Delta x}, \frac{\Delta I}{\Delta y} \right)^T \qquad (3.22)$$

ΔI je rozdíl výšek dvou počítaných prvků a Δx, Δy určují jejich vzdálenost. Pro jednoduchou aproximaci gradientu můžeme vzít pouze dva prvky ve směru osy x a dva prvky ve směru osy y, je ale lepší pro větší přesnost simulace počítat derivace ve směru x a y pro všechny sousední prvky, které jsou nižší než aktuální prvek. Gradienty pak vypadají následujícím způsobem:

$$\nabla I_{1..8} = \left\{ \begin{array}{lll} \begin{pmatrix} \Delta I_7 \\ -\Delta I_7 \end{pmatrix}, & \begin{pmatrix} 0 \\ -\Delta I_6 \end{pmatrix}, & \begin{pmatrix} -\Delta I_5 \\ -\Delta I_5 \end{pmatrix}, \\ \begin{pmatrix} \Delta I_8 \\ 0 \end{pmatrix}, & & \begin{pmatrix} -\Delta I_4 \\ 0 \end{pmatrix}, \\ \begin{pmatrix} \Delta I_1 \\ \Delta I_1 \end{pmatrix}, & \begin{pmatrix} 0 \\ \Delta I_2 \end{pmatrix}, & \begin{pmatrix} -\Delta I_3 \\ \Delta I_3 \end{pmatrix} \end{array} \right\} \qquad (3.23)$$

kde $\Delta I_n = I(x,y) - I_n$ pro n = 1…8. *I(x, y)* vypočteme podle vzorce (3.24) a I_n značí $I(x_n, y_n)$, kde x_n a y_n jsou souřadnice n-tého souseda počítaného prvku.

$$I(x,y) = H(x,y) + K_F.F(x,y) \qquad (3.24)$$

Parametr $K_F \in \langle 0,1 \rangle$ říká, jaké množství vody na daném prvku bude ovlivňovat výpočet zrychlení. Například v referovaném článku se používal pro výpočet K_F následující vztah:

$$K_F = \max(1 - 0{,}05.|\vec{v}|, 0) \qquad (3.25)$$

Výsledný gradient se spočítá jako průměr jednotlivých derivací, jak naznačuje vztah (3.26).

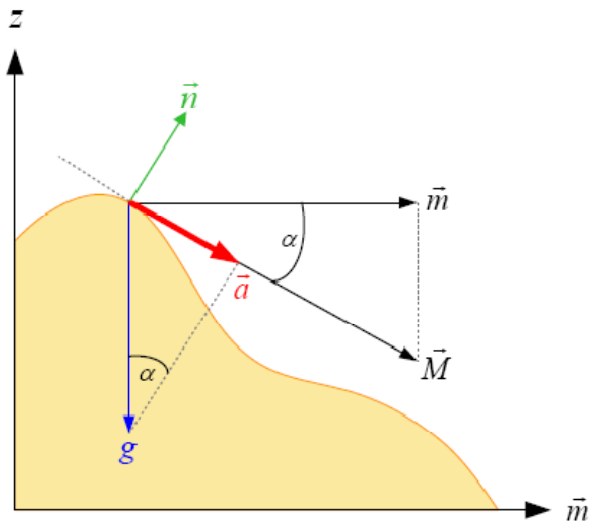
$$\nabla I(x,y) = avg(\nabla I_{1..8}) \qquad (3.26)$$

Tento způsob výpočtu gradientu je poměrně rychlý, je však vhodný, pouze pokud má voda velikou rychlost a gradient v konkrétním místě tok příliš neovlivňuje. Pro pomalý tok vody se ale příliš nehodí, protože umožňuje urychlení vody pouze v jednom směru a to ve směru gradientu. U malé rychlosti vody musíme tok rozdělit a akcelerovat v jednotlivých směrech všech nižších sousedů. To, který způsob výpočtu použijeme, rozhodneme podle určité vhodně zvolené prahové rychlosti.

Nyní máme vypočítaný gradient a tedy i určený směr zrychlení $\vec{m}$ v rovině *xy*:

$$\vec{m} = -\nabla I(x,y) \qquad (3.27)$$

Pro získání trojrozměrného směru zrychlení $\vec{M}$ ještě musíme započíst osu z. Názorná situace je vidět na obrázku 3.6.



Obrázek 3.6: Výpočet zrychlení (obrázek pochází z [NWD05])

Vektor $\vec{M}$ je projekcí vektoru $\vec{m}$ do roviny kolmé k normálovému vektoru terénu a procházející počítaným bodem. Platí následující vztahy:

$$\vec{n} = \left(\frac{\Delta I}{\Delta x}, \frac{\Delta I}{\Delta y}, -1 \right)^T$$

$$\vec{n}.\vec{M} = 0 \qquad (3.28)$$

$$\vec{M} = \left(-\frac{\Delta I}{\Delta x}, -\frac{\Delta I}{\Delta y}, -\frac{\Delta I^2}{\Delta x^2} - \frac{\Delta I^2}{\Delta y^2} \right)^T$$

Vektor zrychlení tedy určíme následovně:

$$\vec{a} = \sin\alpha . g . \frac{\vec{M}}{|\vec{M}|} \qquad (3.29)$$

Díky výpočtu vektoru $\vec{M}$ můžeme ještě *sin α* nahradit podílem protilehlé strany a odvěsny znázorněného trojúhelníku, tedy:

$$\sin\alpha = \frac{|M_z|}{|\vec{M}|} \qquad (3.30)$$

kde M_z je souřadnice z vektoru $\vec{M}$. Výsledný vzorec pro výpočet zrychlení v daném bodě tedy je:

$$\vec{a} = \frac{|M_z|}{|\vec{M}|} . g . \frac{\vec{M}}{|\vec{M}|} \qquad (3.31)$$

Nyní máme spočítané zrychlení a můžeme se vrátit k rovnicím (3.20), kde už můžeme dopočítat rychlost $\vec{v}_{t+\Delta t}$. Když známe $\vec{v}_{t+\Delta t}$, můžeme také dopočítat $\vec{x}_{t+\Delta t}$, tedy oblast, kam se

unášený materiál dostane. Tato oblast nemusí vyjít přesně na jeden konkrétní prvek, v tom případě se pak materiál pomocí bilineární interpolace rozdělí na čtyři nejbližší prvky.

Pokud označíme původní množství vody v cílovém prvku F_{dest} a přenášené množství vody F_{add}, dostaneme nové množství vody v cílovém prvku F_{new} pouhým sečtením těchto dvou množství:

$$F_{new} = F_{dest} + F_{add} \qquad (3.32)$$

Pro novou rychlost vody $\vec{v}_{new}$ v cílovém prvku je výpočet podobný, jen s tím rozdílem, že se rychlosti váží podle množství přenášené vody.

$$\vec{v}_{new} = k.\vec{v}_{dest} + (1-k)\vec{v}_{add}$$
$$k = \frac{F_{dest}}{F_{new}} \qquad (3.33)$$

$\vec{v}_{dest}$ je původní rychlost vody v cílovém prvku, $\vec{v}_{add}$ je rychlost přenášeného množství vody a k je váhový koeficient.

Je také důležité po každém kroku simulace přidat difúzní krok, ve kterém se rychlosti a množství unášeného materiálu mírně promíchají a vyrovnají. To provedeme tak, že aktuální hodnotu daného prvku distribuujeme do jeho čtyř přímých sousedů. Tento difúzní krok je důležitý kvůli tomu, aby nevznikaly oscilace.

To by bylo vše o transportu vody, nyní se podíváme na samotnou erozi, která se skládá z rozrušování materiálu vodou a jeho usazování. Abychom zjistili, jaké maximální množství materiálu může být rozpuštěno ve vodě na daném prku, spočítáme si kapacitu vody c_S:

$$c_S = \frac{K_C}{\Delta t}.\Delta F.|\vec{v}| \qquad (3.34)$$

K_C je konstanta kapacity, která určuje, jak mnoho materiálu může být rozpuštěno v jednotkovém objemu vody tekoucí jednotkovou rychlostí a ΔF je množství přenesené vody.

Abychom rozlišili, zda budeme materiál rozpouštět nebo usazovat, porovnáme kapacitu c_S s aktuálním množstvím rozpuštěného sedimentu ΔS. Pokud je $\Delta S > c_S$, usazujeme následujícím způsobem:

$$H = H + \frac{K_D}{\Delta t}.(\Delta S - c_S)$$
$$S = S + \Delta S - \frac{K_D}{\Delta t}.(\Delta S - c_S) \qquad (3.35)$$

Pokud je $\Delta S \le c_S$, rozpouštíme podle následujícího předpisu:

$$H = H + \frac{K_S}{\Delta t}.(\Delta S - c_S)$$
$$S = S + \Delta S - \frac{K_S}{\Delta t}.(\Delta S - c_S) \qquad (3.36)$$

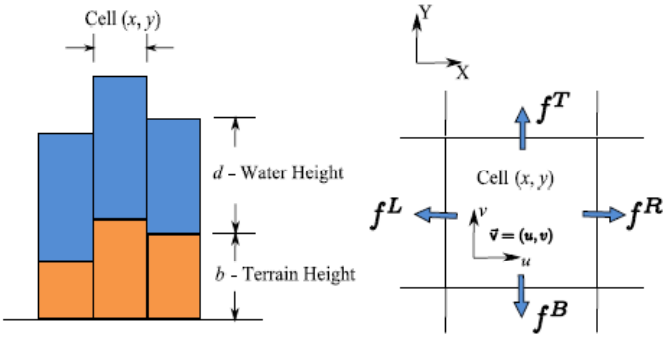
$K_D \in \langle 0,1 \rangle$ je usazovací konstanta, pomocí které můžeme kontrolovat rychlost usazování materiálu, a $K_S \in \langle 0,1 \rangle$ je rozpouštěcí konstanta, pomocí které můžeme kontrolovat rychlost rozpouštění materiálu.

3.2.4. X. Mei, P. Decaudin, B.-G. Hu

V článku [MDH07] je popsán opět algoritmus vodní eroze založený na silách. Ten vychází z erozního algoritmu, který je popsaný v předchozí kapitole 3.2.3. (tedy z článků [CMF98] a [NWD05]). Pro transport vody je zde ale použit tzv. *pipe model*, který popsali James F. O´Brien a Jessica K. Hodgins v [OH95].

Celý proces eroze se skládá z následujících kroků. Nejprve se na mapě objeví nějaká voda. Ta může být buď z vodních zdrojů anebo z deště. Pomocí *pipe modelu* se pak spočítá tok vody a upraví se vodní hladina a rychlost vody v každém prvku výškové mapy. Upravené rychlosti vody se dále využijí při erodování nebo usazování materiálu a následném přenosu materiálu rozpuštěného ve vodě. Nakonec dojde k odpaření určité části vody.

Pro každý prvek výškové mapy je potřeba si zapamatovat následující údaje: výšku terénu, která bude dále označena *b*, výšku vody nad terénem označenou *d*, množství rozpuštěného materiálu *s*, odtok vody v jednotlivých směrech $f = \left(f^L, f^R, f^T, f^B\right)$ a vektor horizontální rychlosti vody $\vec{v} = (u, v)$. Obrázek 3.7 tyto hodnoty potřebné pro simulaci znázorňuje. Dále budou u jednotlivých proměnných používány indexy pro rozlišení v jakém čase se nacházejí. Například d_t vyjadřuje výšku vody v čase *t* a $d_{t+\Delta t}$ vyjadřuje výšku vody v dalším časovém kroku. Pokud bude index číselný, například d_1 nebo d_2, znamená to, že se jedná o mezivýpočty výšky vody někde mezi časy *t* a $t + \Delta t$.



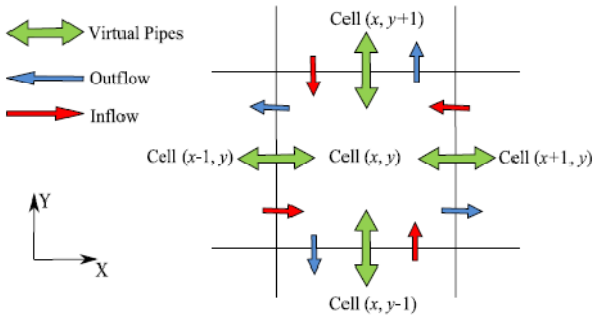
Obrázek 3.7: Hodnoty potřebné pro simulaci (obrázek pochází z [MDH07])

Pro výskyt nové vody v daném časovém kroku je použit následující jednoduchý vztah:

$$d_1(x,y) = d_t(x,y) + \Delta t.r_t(x,y) \qquad (3.37)$$

kde r_t je množství vody, které přibude na daném místě za jednotku času díky některému vodnímu zdroji nebo dešti.

Dalším krokem je transport vody. Tento krok je z celé simulace asi nejsložitější. Jak již bylo řečeno na začátku, využívá se zde tzv. *pipe model* znázorněný na obrázku 3.8. Jednotlivé buňky si se svými přímými sousedy předávají vodu skrz virtuální rouru a pamatují si velikost tohoto toku.



Obrázek 3.8: Pipe model (obrázek pochází z [MDH07])

V každém kroku se tok mění podle rozdílu hydrostatických tlaků uvnitř vody v buňkách připojených k virtuální rouře. Poté se výška hladiny vody upraví podle součtu všech toků ze všech virtuálních rour.

Může nastat situace, kdy vyjde výška hladiny záporná, což nelze. V tomto případě by se musela část chybějící vody stáhnout zpět od sousedů, kteří vodu odebrali. Autoři článku tento problém řeší jiným způsobem. U každé buňky pracují pouze s výstupními toky, které uchovávají v proměnné $f = \left(f^L, f^R, f^T, f^B\right)$, kde f^L znamená odtok vody z aktuální buňky (x, y) do levé sousední buňky o souřadnicích (x – 1, y), podobně f^R je odtok z aktuální buňky do pravé soudní buňky o souřadnicích (x + 1, y), f^T je odtok do horní sousední buňky

(x, y + 1) a f^B je odtok do spodní sousední buňky (x, y – 1). Pokud nastane situace, kdy součet všech výstupních toků převyšuje množství vody v dané buňce, je každý výstupní tok úměrně zmenšen tak, aby součet všech výstupních toků byl roven nule.

Pro výpočet jednotlivých toků je nejprve potřeba spočítat rozdíly výšek mezi danou buňkou a jejími sousedy podle následujících vztahů:

$$\begin{aligned}
\Delta h_t^L(x,y) &= b_t(x,y) + d_1(x,y) - b_t(x-1,y) - d_1(x-1,y) \\
\Delta h_t^R(x,y) &= b_t(x,y) + d_1(x,y) - b_t(x+1,y) - d_1(x+1,y) \\
\Delta h_t^T(x,y) &= b_t(x,y) + d_1(x,y) - b_t(x,y+1) - d_1(x,y+1) \\
\Delta h_t^B(x,y) &= b_t(x,y) + d_1(x,y) - b_t(x,y-1) - d_1(x,y-1)
\end{aligned} \qquad (3.38)$$

Toky pro danou buňku se pak spočítají podle vzorce (3.39), kde *i = {L, R, T, B}*, *A* je plocha příčného řezu virtuální roury, *g* je gravitační zrychlení ($g \doteq 9{,}81 m.s^{-2}$) a *l* je délka virtuální roury.

$$f_{t+\Delta t}^i(x,y) = \max\left(0, f_t^i(x,y) + \Delta t.A.\frac{g.\Delta h_t^i(x,y)}{l}\right) \qquad (3.39)$$

Po spočítání všech čtyř výstupních toků v dané buňce musíme ještě zjistit, zda existuje pro odtok dostatek vody a případně výstupní toky zmenšit. Spočítáme proto faktor zmenšení *K* podle vztahu (3.40).

$$K = \min\left(1, \frac{d_1.l_X.l_Y}{(f^L + f^R + f^T + f^B).\Delta t}\right) \qquad (3.40)$$

Konstanty l_X a l_Y znamenají vzdálenosti bodů mřížky ve směru osy x a ve směru osy y. Pokud je K menší než jedna, musíme ještě všechny čtyři výstupní toky tímto faktorem vynásobit, čímž dostaneme celkový výsledný tok, který je roven množství vody v dané buňce. Konečný výstupní tok tedy vyjadřuje tento vztah:

$$f_{t+\Delta t}^i(x,y) = K.f_{t+\Delta t}^i(x,y) \qquad (3.41)$$

Pro buňky nacházející se na okrajích výškové mapy předpokládáme, že voda nebude odtékat mimo mapu, proto by se měl výstupní tok vody v těchto buňkách, který směřuje na neexistujícího souseda mimo mapu, nastavit na nulu.

Nyní, když jsou spočítány všechny výstupní toky všech buněk, musíme upravit výšku vodní hladiny a rychlost vody. Změna vodní hladiny *Δd* je dána rozdílem celkového přítoku do dané buňky od všech sousedů a celkovým odtokem z dané buňky. Výpočet vyjadřuje následující vzorec:

$$\Delta d(x,y) = \frac{\Delta t}{l_X l_Y} \cdot \left(f^R_{t+\Delta t}(x-1,y) + f^L_{t+\Delta t}(x+1,y) + f^B_{t+\Delta t}(x,y+1) + f^T_{t+\Delta t}(x,y-1) - \sum_{i=L,R,T,B} f^i_{t+\Delta t}(x,y) \right) \quad (3.42)$$

Celková výška vody d_2 je pak dána součtem původní výšky hladiny d_1 a změny vodní hladiny Δd:

$$d_2(x,y) = d_1(x,y) + \Delta d(x,y) \quad (3.43)$$

Rychlost vody $\vec{v} = (u,v)$ můžeme spočítat z průtoku vody danou buňkou. Průměrné množství vody, které projde danou buňkou za jednotku času ve směru osy x, je:

$$\Delta W_X(x,y) = \frac{f^R_{t+\Delta t}(x-1,y) - f^L_{t+\Delta t}(x,y) + f^R_{t+\Delta t}(x,y) - f^L_{t+\Delta t}(x+1,y)}{2} \quad (3.44)$$

Průměrné množství vody, které projde danou buňkou za jednotku času ve směru osy y, je:

$$\Delta W_Y(x,y) = \frac{f^T_{t+\Delta t}(x,y-1) - f^B_{t+\Delta t}(x,y) + f^T_{t+\Delta t}(x,y) - f^B_{t+\Delta t}(x,y+1)}{2} \quad (3.45)$$

Pomocí těchto průměrných množství vody už snadno dopočítáme výslednou rychlost podle vzorce (3.46), kde $\bar{d}(x,y) = \frac{d_1(x,y) + d_2(x,y)}{2}$ je průměrná výška vody během prvních dvou kroků.

$$\vec{v}_{t+\Delta t}(x,y) = \left(\frac{\Delta W_X(x,y)}{l_X . \bar{d}(x,y)}, \frac{\Delta W_Y(x,y)}{l_Y . \bar{d}(x,y)} \right) \quad (3.46)$$

Po dokončení transportu vody začíná proces eroze a usazování. Tento proces je velmi podobný tomu z předcházející kapitoly, liší se jen nepatrně ve výpočtu kapacity vody pro rozpuštěný sediment. Opět probíhá tak, že tekoucí voda pod sebou rozrušuje půdu, zachytává její částečky a unáší je na jiné místo, kde dojde k jejich usazení.

Rychlost tohoto procesu ovlivňuje hlavně velikost kapacity vody C. Ta se zde vypočítá následujícím způsobem:

$$C(x,y) = K_C \cdot \sin(\alpha(x,y)) \cdot |\vec{v}(x,y)| \quad (3.47)$$

kde K_C je konstanta kapacity sedimentu, kterou definuje uživatel, a $\alpha(x,y)$ je úhel sklonu terénu v daném místě. Nevýhodou tohoto výpočtu je, že pokud je terén rovný a úhel sklonu se blíží k nule, je eroze minimální a tím pádem neznatelná. Tento problém se dá vyřešit tak, že se nastaví prahová hodnota minimálního úhlu sklonu, takže pokud je úhel sklonu menší, než tato prahová hodnota, je nastaven na tuto hodnotu.

Pro rozhodnutí, zda se bude erodovat půda nebo usazovat rozpuštěný sediment, se porovná vypočítaná kapacita $C(x,y)$ s aktuálním množstvím ve vodě rozpuštěného

sedimentu $s_t(x,y)$. Pokud $C(x,y)$ je větší než $s_t(x,y)$, dojde k rozpuštění půdy ve vodě následujícím způsobem:

$$\begin{aligned} b_{t+\Delta t}(x,y) &= b_t - K_S\big(C(x,y) - s_t(x,y)\big) \\ s_1(x,y) &= s_t(x,y) + K_S\big(C(x,y) - s_t(x,y)\big) \end{aligned} \qquad (3.48)$$

V opačném případě dojde k usazení rozpuštěného sedimentu:

$$\begin{aligned} b_{t+\Delta t}(x,y) &= b_t - K_D\big(C(x,y) - s_t(x,y)\big) \\ s_1(x,y) &= s_t(x,y) + K_D\big(C(x,y) - s_t(x,y)\big) \end{aligned} \qquad (3.49)$$

K_S je konstanta rozpouštění a K_D je konstanta usazování. Obě tyto konstanty si definuje sám uživatel.

Poté, co se provede eroze nebo usazení pro každý prvek mapy, je nově vzniklý rozpuštěný sediment přemístěn vodou podle tohoto vztahu:

$$s_{t+\Delta t}(x,y) = s_1(x - u.\Delta t, y - v.\Delta t) \qquad (3.50)$$

Pokud souřadnice pro s_1 nevycházejí celočíselně, ale ukazují někam mezi prvky mřížky, získáme výslednou hodnotu pomocí lineární interpolace čtyř nejbližších prvků od této pozice.

Posledním krokem je vypařování vody. To je vyjádřeno následovně:

$$d_{t+\Delta t}(x,y) = d_2(x,y) \cdot (1 - K_e.\Delta t) \qquad (3.51)$$

K_e je konstanta vypařování.

3.3. **Větrná eroze**

Větrnou erozi lze rozdělit na dvě fáze. První je obrus způsobený třením větrem transportovaného materiálu, je závislý na síle a úhlu dopadajícího větru a na množství a hrubosti unášeného materiálu. Druhou je pak odnos sypkého zvětralého materiálu. Větrná eroze se z těchto tří typů erozí používá v počítačové grafice nejméně, slouží především pro modelování písečných dun a pouštních krajin.

Pouště jsou totiž oblasti s minimálním množstvím srážek, a tak zde vodní eroze hraje pouze nepatrnou roli v utváření krajiny. Hlavní je zde především tepelná eroze, protože v pouštích dochází k mnohem většímu kolísání teplot, než v oblastech s vegetací. Díky tepelné erozi se postupně skalnaté části povrchu drolí a mění na písek. Vítr pak zrnka písku zachycuje a přenáší na jiné místo. Unášená zrnka písku mohou ještě dále obrušovat skalnaté části, tím dochází k větrné erozi.

3.3.1. **Bedřich Beneš, Toney Roa**

Bedřich Beneš a Toney Roa představili v [BR04] algoritmus pro simulaci pouštní krajiny. Nejedná se přímo o simulaci větrné eroze, ale pouze o přemisťování písku větrem. Tento transport materiálu je ale důležitou součástí větrné eroze. V algoritmu simulují přemisťování lehkých zrnek písku, které vítr snadno zachytí a přenese na nějaké jiné místo, kde se usadí do energeticky nejvýhodnější polohy. Vítr tak v písku vytváří vlnky, které jsou pro pouště charakteristické.

Určité nejmenší množství písku, které může být zachyceno větrem, může být předmětem tří procesů: poskakování, suspenze a plazení.

Tzv. poskakování vzniká, když vítr zachytí částečky písku a přemístí je na jiné místo. Tento proces vyjadřují následující rovnice:

$$\begin{aligned} hf'(p) &= hf(p) - q_s(p) \\ hf'(p+l) &= hf(p+l) + q_s(p) \end{aligned} \qquad (3.37)$$

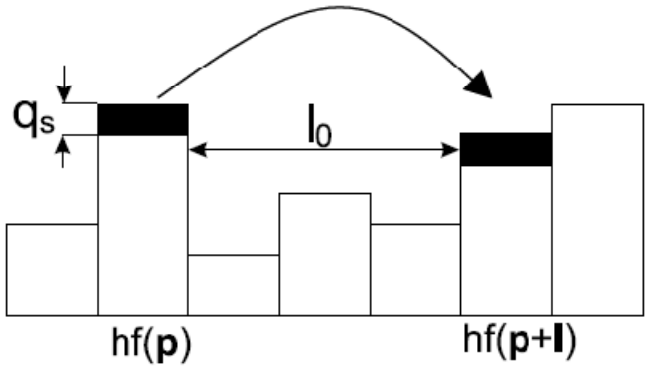
kde *hf* je označení pro výškovou mapu v čase *t*, *hf´* označuje výškovou mapu v čase $t + \Delta t$ (kde Δt je časový krok), *p* značí pozici, kde je písek zachycen, q_s je množství zachyceného písku, které se vypočítá podle vztahu:

$$q_s(p) = q_0\left(1 + \tanh(\nabla hf)\right) \qquad (3.38)$$

kde q_0 je průměrné množství přeneseného písku. Vektor $l = (l_i, l_j)$ je vektor horizontálního posunutí se složkami:

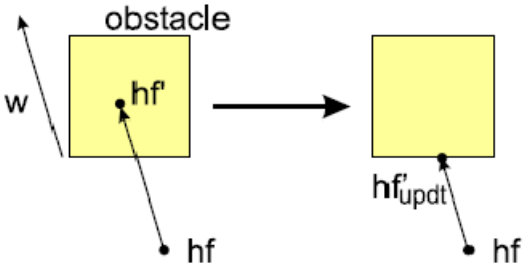
$$\begin{aligned} l_i &= \left(l_{0i} + w_i hf\right)\left(1 - \tanh\frac{\partial hf}{\partial i}\right) \\ l_j &= \left(l_{0j} + w_j hf\right)\left(1 - \tanh\frac{\partial hf}{\partial j}\right) \end{aligned} \qquad (3.39)$$

kde $w = (w_i, w_j)$ je vektor, který reprezentuje vítr (velikost vektoru odpovídá intenzitě větru a směr vektoru odpovídá směru větru), $l_0 = (l_{0i}, l_{0j})$ je vektor který určuje průměrný skok a spolu s vektorem w ovlivňují frekvenci vlnek. Parametry q_0, *w* a l_0 si určuje sám uživatel, musejí ale být z intervalu $\langle 0,1 \rangle$, jinak by byl systém nestabilní a vlnky by se netvořily. Obrázek 3.9 schematicky znázorňuje jeden skok tohoto procesu.



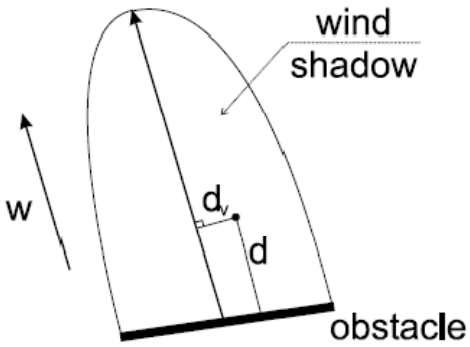
Obrázek 3.9: Znázornění procesu poskakování (obrázek pochází z [BR04])

Při poskakování může nastat problém, pokud narazíme na nějakou překážku. Písek nemůže být přenesen dovnitř objektu, pokud by takováto situace nastala, musí být výsledná pozice přesunu upravena tak, že se písek usadí těsně před objektem (myšleno ve směru větru). Situaci naznačuje obrázek 3.10.



Obrázek 3.10: Případ přesunu písku dovnitř materiálu (obrázek pochází z [BR04])

Další problém, který musíme řešit při výskytu nějaké překážky, je větrný stín, který za sebou překážka vytváří. Beneš a Roa navrhli čistě *ad hoc* techniku, která je rychlá a dává věrohodné vizuální výsledky. K určení větrného stínu musíme nejprve objekt promítnout do roviny kolmé ke směru větru a procházející středem objektu. Střed objektu a směr větru určují hlavní osu větrného stínu. Délka větrného stínu je závislá na výšce objektu a na intenzitě větru. Šířka větrného stínu je závislá pouze na šířce promítnuté oblasti. Pokud se bod výškové mapy nachází ve větrném stínu a je předmětem procesu poskakování, je intenzita větru lineárně snížena v závislosti na vzdálenosti od objektu a na jeho výšce. Intenzita závisí nejen na vzdálenosti od překážky, ale taká na vzdálenosti od hlavní osy větrného stínu. Příklad větrného stínu můžete vidět na obrázku 3.11.



Obrázek 3.11: Větrný stín (obrázek pochází z [BR04])

Proces suspenze vzniká, když vítr zachytí částice prachu a přenáší je neurčitě dlouhou dobu. Blíženi tento proces autoři článku nepopisují.

Posledním procesem je tzv. plazení. Pokud se písek přesune na nestabilní místo, začne se sesouvat ze svahu dolů, dokud nedosáhne energeticky nejvýhodnější pozice. Tento proces se provádí pomocí modifikovaného algoritmu tepelné eroze popsaným kapitole 3.1.1. Nejprve se pro každý prvek výškové mapy vypočítá lokální gradient a pak se tento gradient každého prvku porovnává se svahovým úhlem T (*talus angle*). V případě potřeby se přesune maximální možné množství písku a toto množství je na sousedy rozděleno úměrně k velikosti gradientu. Tento proces se opakuje tak dlouho, dokud je potřeba nějaký materiál přemisťovat.

3.4. **Shrnutí**

Bylo zde představeno několik algoritmů různých druhů eroze. Základní algoritmus tepelné i vodní eroze navrhli Musgrave a kolektiv v roce 1989. Většina dalších autorů na tyto algoritmy ať již přímo či nepřímo navazuje a něčím novým je obohacuje. Dá se říci, že bychom mohli předpokládat, že čím novější algoritmus je, tím zpravidla bývá lepší než algoritmy předcházející.

Zásadním vylepšením, které je možné aplikovat na libovolné druhy erozí, je podle mého názoru navržení vrstvené datové struktury. Díky této struktuře už nemusejí erozní algoritmy pracovat pouze s výškovou mapou s jedním homogenním materiálem, čímž se mohou více přiblížit skutečné přírodě.

Dále asi obecně platí, že tzv. *ad hoc* erozní algoritmy, jejichž výstupy jsou založeny pouze na vizuální podobnosti s realitou a nikoliv na fyzikální korektnosti, bývají zpravidla rychlejší než algoritmy založené na fyzice. Obvykle ale vyžadují nastavování velkého množství různých koeficientů, práce s nimi není tolik intuitivní a pro efektivní práci je potřeba mít s daným algoritmem určité zkušenosti. Také tyto algoritmy nejsou příliš použitelné

v aplikacích, které vyžadují fyzikální přesnost, jako jsou například různé geografické informační systémy apod. Proto si myslím, že jsou lepší algoritmy založené na fyzice, které kromě větší výpočetní náročnosti mají oproti vizuálně založeným algoritmům jenom samé výhody.

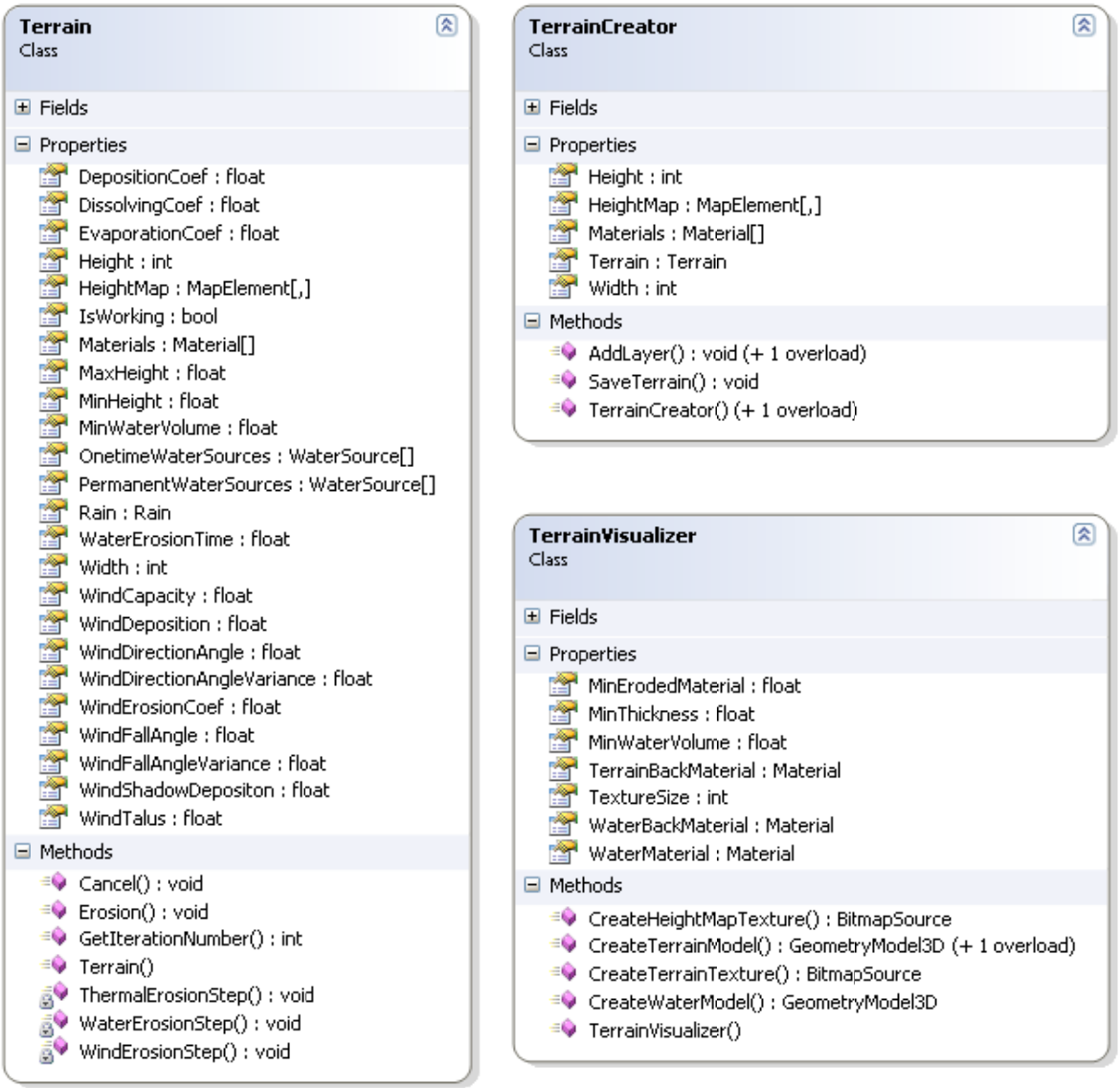
Pokud se zaměříme jen na tepelnou erozi, zjistíme, že od původní verze z roku 1989 se příliš nezměnilo. Beneš a Forsbach tento algoritmus rozšířili pouze o krok vypařování a jednotlivé kroky od sebe oddělili. Olsen poté navrhnul optimalizaci, kde místo Mooreova okolí použil upravené Von Neumannovo okolí a tím snížil počet vyhodnocovaných podmínek na polovinu a množství přenášeného materiálu zvýšil na maximum. Pro implementaci jsem si proto zvolil tepelnou erozi, kterou navrhli Beneš a Forsbach rozšířenou o Olsenovy optimalizace.

U vodní eroze je situace trochu komplikovanější, protože se toto téma těší většímu zájmu autorů. Postupem času se vodní eroze rozdělila na dva podtypy: vodní erozi založenou na silách a vodní erozi založenou na rozpustnosti. Oběma typy se zabývalo několik autorů a například Šťava, Beneš, Brisbin a Křivánek v [SBBK08] oba tyto typy spojili. Osobně si ale myslím, že vodní eroze založená na silách sama o sobě dává velmi dobré výsledky a ve spojení s erozí založenou na rozpustnosti se tyto výsledky už příliš nezmění, zato naroste výpočetní náročnost algoritmu. U vodní eroze je velmi důležitý transport vody, protože pokud se simulovaná voda nechová jako skutečná voda, nemůže ani simulovaná eroze vypadat jako skutečná. Během mé práce jsem vyzkoušel několik způsobů transportu vody, z nich nejvíce podobný chování skutečné vody byl způsob popsaný v [OH95]. Vodní erozi založenou na silách a využívající tento způsob transportu vody popsali Mei, Decaudin a Hu v článku [MDH07], proto jsem při implementaci vycházel hlavně z něj.

Oproti předchozím dvěma typům erozí jsem se nesetkal s článkem, který by se zabýval čistě větrnou erozí. Existují pouze články, které se zabývají tvorbou písečných dun a vlnek v písku vytvořených větrem. Proto jsem se rozhodnul vytvořit vlastní algoritmus větrné eroze, který ale nebude jako předešlé dva algoritmy založený na fyzice, ale spíše na vizuální podobnosti.

4. Implementace

Pro implementaci byl použit programovací jazyk C#. Byla vytvořena knihovna *ErosionLibrary.dll*, která obsahuje třídu pro vytváření terénu *TerrainCreator*, třídu pro reprezentaci terénu *Terrain*, která také umožňuje erozi, a třídu pro vizualizaci terénu *TerrainVisualizer*. Diagram těchto tříd je znázorněn na obrázku 4.1, kde z úsporných důvodů nejsou zobrazeny soukromé proměnné. Většině těchto soukromých proměnných ale odpovídají veřejné vlastnosti, které v diagramu znázorněny jsou. Dále tato knihovna obsahuje ještě interní třídu *Generator* sloužící pro generování náhodných čísel s normálním rozdělením, která má pouze jednu metodu bez parametrů *NextNumber* vracející náhodné číslo, a veřejné datové struktury používané v ostatních třídách, které jsou znázorněny na obrázku 4.2 a popsány v následující kapitole 4.1.



Obrázek 4.1: Diagram veřejných tříd knihovny ErosionLibrary.dll

Tato knihovna byla vytvořena pro WPF (Windows Presentation Foundation). V dalších podkapitolách si stručně popíšeme používané datové struktury v této knihovně a základní tři třídy *TerrainCreator*, *Terrain* a *TerrainVisualizer*. U třídy *Terrain* se pak detailněji zaměříme na popis jednotlivých erozních metod, které jsou hlavním tématem této práce.

4.1. **Datové struktury**

Knihovna pracuje s následujícími šesti datovými strukturami: *WaterSource*, *Rain*, *IntPoint*, *Material*, *MapElement*, *Layer*. Tyto struktury znázorňuje diagram na obrázku 4.2. Všechny struktury obsahují veřejné proměnné a kromě struktury *MapElement* také konstruktor, který tyto proměnné naplní hodnotami. Všechny proměnné jsou veřejné z toho důvodu, aby práce s nimi byla rychlejší, i přestože to z programátorského hlediska není úplně korektní. Přístup k proměnným přes vlastnosti je totiž pomalý a způsoboval by výrazné prodloužení výpočetní doby, protože se k těmto proměnným během výpočtu přistupuje poměrně často.

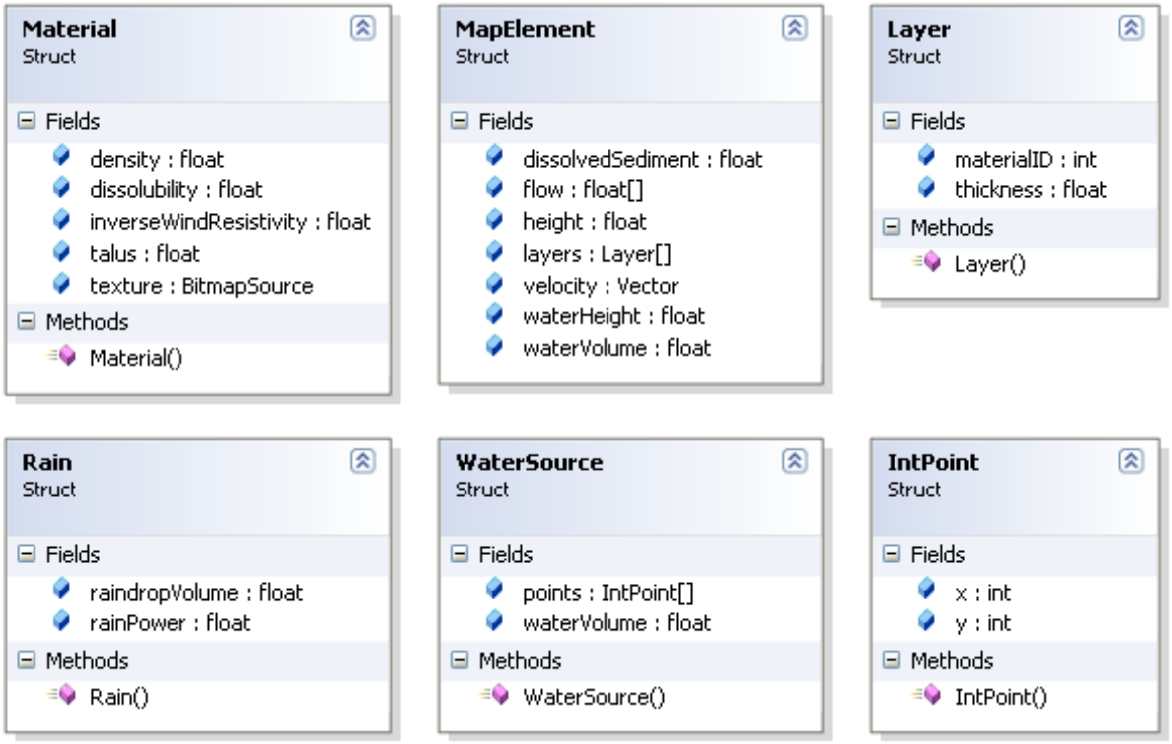
Nejdůležitějšími strukturami jsou *Material* a *MapElement*. Struktura *Material* slouží pro uchovávání informací o jednotlivých materiálech použitých ve vrstvách terénu. Každý materiál má svou denzitu (*density*) z intervalu od nuly do jedné, která určuje jeho odolnost vůči tepelné erozi. Nulová denzita znamená, že tepelná eroze materiál vůbec neovlivní, naopak jednotková denzita způsobí maximální rozpad materiálu při tepelné erozi. S tepelnou erozí je ještě spojen úhel maximálního sklonu svahu *talus* ve stupních. Dále má každý materiál svou rozpustnost (*dissolubility*) také z intervalu od nuly do jedné. Tato rozpustnost udává míru rozpustnosti materiálu ve vodě. Pokud je rozpustnost nulová, materiál je zcela odolný vůči vodní erozi, pokud je jednotková, je maximálně rozpustný ve vodě. Koeficientem pro ovlivňování míry větrné eroze je *inverseWindResistivity* opět z intervalu od nuly do jedné, kde nula znamená úplnou odolnost proti větru a jedna minimální odolnost proti větru. Poslední informací, kterou si materiál uchovává a kterou budeme potřebovat pro jeho vizualizaci, je textura materiálu *texture*.

Struktura *MapElemet* slouží k reprezentaci jednoho prvku výškové mapy. Každý tento prvek obsahuje výšku terénu *height* v tomto bodě, pole vrstev terénu *layers*, kde jsou jednotlivé vrstvy terénu seřazeny od nejvyšší po nejnižší (základní) vrstvu. Tyto vrstvy jsou reprezentovány strukturou *Layer*, která obsahuje pouze tloušťku *thickness* dané vrstvy v daném místě a identifikační číslo materiálu dané vrstvy *materialID*. Toto identifikační číslo odpovídá indexu materiálu v poli všech použitých materiálů. Při vytvoření terénu pole *layers* obsahuje pouze vrstvy s kladnou tloušťkou, tedy nikoliv s nulovou. Dále každý prvek výškové

mapy obsahuje výšku vody *waterHeight* a objem vody *waterVolume*. Objem vody udává výšku vody nad povrchem terénu (rozměry buněk terénu se předpokládají jednotkové, proto výška vody nad povrchem číselně odpovídá objemu vody na dané buňce). Výška vody *waterHeight* pak udává absolutní výšku vodní hladiny, tedy součet výšky terénu *height* a objemu vody *waterVolume*. Dále si každý prvek pamatuje množství ve vodě rozpuštěného sedimentu *dissolvedSediment*, rychlost vody *velocity* v daném místě a čtyři výstupní toky v poli *flow*. Tyto toky byly použity pro simulaci transportu vody v kapitole 3.2.4. a budou ještě dále podrobněji popsány v kapitole 4.3.2.

Dalšími strukturami jsou *Rain* a *WaterSource*, které slouží pro uchovávání údajů o zdrojích vody pro vodní erozi. Struktura *Rain* obsahuje informace o dešti a to velikost objemu kapky *raindropVolume*, která udává množství (tedy výšku) vody, které dopadne na povrch terénu v jedné buňce jako jedna dešťová kapka, a sílu deště *rainPower* z intervalu od nuly do jedné, která ovlivňuje počet kapek spadlých v jednom kroku vodní eroze. Pokud je síla nulová, nespadne žádná dešťová kapka, pokud je jednotková, spadne v jednom kroku počet kapek rovný celkovému počtu prvků výškové mapy. Neznamená to ale, že na každý prvek dopadne jedna kapka. Kapky dopadají náhodně a může se stát, že na některý prvek dopadne více kapek v jednom kroku a na některý jiný žádná.

Struktura *WaterSource* slouží pro reprezentaci vodního zdroje, kterým může být buď voda, která se jednorázově na začátku simulace eroze objeví v nějaké oblasti, nebo voda, která se v určité oblasti objevuje pravidelně v každém kroku. Oblast výskytu představuje pole prvků typu *IntPoint*, kde struktura *IntPoint* obsahuje pouze x-ovou a y-ovou souřadnici jednoho prvku výškové mapy. V každém prvku této oblasti pak přibude množství vody definované v proměnné *waterVolume*.



Obrázek 4.2: Diagram veřejných struktur knihovny ErosionLibrary.dll

4.2. **Vytváření terénu**

Třída pro vytváření terénu *TerrainCreator* obsahuje konstruktor, který slouží k vytvoření základní výškové mapy a k určení vlastností erodovaného materiálu. Výšková mapa základního terénu se načte buď ze zadané bitmapy, nebo ze zadaného *raw* souboru, kde jsou navíc dalšími parametry potřebnými pro vytvoření výškové mapy její rozměry, tedy výška a šířka, protože pokud nebudeme předpokládat pouze čtvercové mapy, nedokážeme pouze z velikosti souboru určit tyto rozměry. Další parametry jsou minimální a maximální hodnota výšky výškové mapy, podle kterých se výška jednotlivých prvků mapy interpoluje, materiál, ze kterého se základní terén skládá a materiál, a materiál erodované vrstvy. Pro vrstvu základního terénu je nastavena nekonečně veliká tloušťka.

Dále třída *TerrainCreator* obsahuje jednu veřejnou metodu *AddLayer*, která umožňuje k základnímu terénu přidávat nové vrstvy s odlišnými vlastnostmi. Pro vytvoření takovéto nové vrstvy je zapotřebí zadat buď bitmapu, nebo už jen název *raw* souboru, ze kterého se bude výšková mapa načítat. Rozměry výškové mapy se předpokládají stejné, jako byly rozměry základního terénu. Dále je opět nutné zadat ještě minimální a maximální hodnotu výšky této mapy a materiál, ze kterého se bude nová vrstva skládat. Šířka vrstvy je nastavena pro každý prvek individuálně podle rozdílu výšky nové vrstvy a výšky dosavadní nejvyšší

vrstvy terénu. Pokud je tento rozdíl záporný, tedy výška nové vrstvy je pod úrovní dosavadního terénu, vrstva se pro daný prvek nepřidá.

Poslední metodou, kterou třída *TerrainCreator* obsahuje, je statická metoda *SaveTerrain*, která uloží výškovou mapu zadaného terénu do *raw* souboru. Parametry jsou pouze ukládaný terén a název souboru, do kterého se výšková mapa uloží.

4.3. **Reprezentace terénu a erozní metody**

Stěžejní třídou celé knihovny je třída *Terrain*. Ta slouží pro reprezentaci terénu, a protože vznikla za účelem provádění různých druhů erozí, obsahuje velké množství vlastností sloužících pro nastavování těchto erozí. Tyto vlastnosti si popíšeme až u jednotlivých druhů erozí. Mimo tyto vlastnosti třída obsahuje ještě několik dalších vlastností a metod, které si popíšeme nyní.

Nejprve začneme konstruktorem třídy. Pro vytvoření nového terénu je zapotřebí zadat rozměr výškové mapy, tedy její šířku *width* a výšku *height*, dvourozměrné pole *heightMap* obsahující struktury *MapElement* a reprezentující tuto výškovou mapu a jednorozměrné pole *materials* skládající se ze struktur *Material*, které obsahuje veškeré materiály použité ve vrstvách terénu. Prvek tohoto pole s indexem nula představuje materiál použitý pro erodovaný materiál. Všechny tyto zadané parametry pak můžeme pomocí příslušných vlastností této třídy kdykoliv přečíst.

Dále tato třída obsahuje vlastnosti *MinHeight* a *MaxHeight*, které umožňují přečíst minimální a maximální výšku terénu. Poslední vlastností, která se netýká přímo některého druhu eroze, je vlastnost pouze pro čtení *IsWorking*. Tato vlastnost vrací hodnotu *true*, pokud probíhá výpočet eroze, v opačném případě vrací hodnotu *false*.

Třída *Terrain* obsahuje tři veřejné metody. Jednou z nich je metoda *GetIterationNumber*, která vrací, v případě že probíhá výpočet eroze, aktuální číslo iterace (neboli počet již provedených kroků simulace). V případě že výpočet eroze neprobíhá, vrací nulu.

Další veřejnou metodou je metoda *Cancel*. Tato metoda slouží k přerušení probíhajícího výpočtu. Výpočet se nepřeruší okamžitě, ale až po dokončení aktuálně počítaného kroku eroze.

Poslední a nejdůležitější metodou celé této třídy je metoda *Erosion*. Jejími parametry jsou celkový počet kroků eroze *stepCount* a proporce jednotlivých erozí, tedy *waterErosionProportion* pro vodní erozi, *windErosionProportion* pro větrnou erozi a *thermalErosionProportion* pro tepelnou erozi. Tyto proporce udávají, v jakém poměru se

jednotlivé kroky erozí budou generovat. Například pokud bude poměr vodní, větrné a tepelné eroze 100:50:1, znamená to, že v každém novém kroku eroze bude pravděpodobnost provedení vodní eroze $\frac{100}{151}$, pravděpodobnost provedení větrné eroze $\frac{50}{151}$ a pravděpodobnost provedení tepelné eroze $\frac{1}{151}$. Jak již je asi patrné z toho příkladu, hlavním úkolem této metody je provést simulaci zadaného počtu kroků eroze a v každém kroku rozhodnout podle zadaného poměru jednotlivých erozí, který typ eroze se v něm provede. Po tomto rozhodnutí se zavolá jedna ze tří soukromých metod *WaterErosionStep*, *WindErosionStep*, *ThermalErosionStep* a provede se krok příslušné eroze. Po každém kroku se ještě zaktualizuje výška vodní hladiny kvůli vizualizaci a výšková mapa se zkopíruje do pomocné proměnné, kde je připravena na předání během výpočtu dalšího kroku, aby se nemohlo stát, že by se předala jen částečně zerodovaná mapa.

V následujících podkapitolách si popíšeme jeden krok každého druhu eroze.

4.3.1. Tepelná eroze

Krok tepelné eroze je poměrně jednoduchý. Pro každý prvek výškové mapy se nejprve určí všichni sousedé z upraveného Von Neumannova okolí (viz obrázek 3.2 v kapitole 3.1.3.), jejichž výška terénu je nižší než výška terénu daného prvku snížená o rozdíl výšek sousedních prvků při maximálním možném svahu. Označme tento rozdíl výšek při maximálním možném svahu ΔT. Vypočítáme ho z úhlu *talus*, který určuje maximální velikost úhlu svahu, vrchního materiálu počítaného prvku takto:

$$\Delta T = \Delta l . \tan(talus) \qquad (4.1)$$

kde Δl je vzdálenost mezi dvěma sousedními prvky a předpokládáme zde, že je jednotková.

U všech nižších sousedů pak sečteme rozdíly výšek terénu počítaného prvku a těchto sousedů. Tento součet označíme *sum* a největší rozdíl označíme d_{max}. Množství materiálu ΔS, které se přenese z daného prvku, spočítáme podle vzorce (4.2).

$$\Delta S = rnd . 0{,}5 . D . d_{\max} \qquad (4.2)$$

kde D je denzita materiálu vrchní vrstvy daného prvku a *rnd* je náhodné číslo, které ve výpočtu být může i nemusí. Číslo *rnd* je do výpočtu zahrnuto z toho důvodu, že v reálné přírodě se vše chová s určitou náhodností. Je generováno s normálním rozdělením se střední hodnotou 1 a rozptylem 0,2. Pokud je jeho hodnota větší než 1, je nastavena na 1, aby nemohlo být přeneseno více materiálu, než je dovoleno.

Pokud přenášené množství materiálu ΔS je větší než tloušťka vrchní vrstvy, pouze ΔS snížíme na tuto tloušťku a vrchní vrstvu smažeme. V původní implementaci jsem sice použil postup stejný jako u ostatních erozí, tedy že jsem místo pouhého snížení ΔS pokračoval erozí další vrstvy, jenže během testování programu jsem objevil chybu, kterou tento způsob řešení způsobuje. Tento způsob totiž při změně vrstvy zohlední pouze měnící se denzitu, jenže s vrstvou se změní i její úhel maximálního svahu (*talus*). Proto, abychom zohlednili i změnu tohoto úhlu, bychom museli provádět znovu srovnávání se všemi sousedy a tedy celý výpočet, z toho důvodu je sníženo množství přenášeného materiálu na tloušťku vrstvy, aby byla pouze odstraněna, a eroze vrstvy pod ní se provede až v následujícím kroku. Toto řešení ale samozřejmě zpomalí průběh eroze, nicméně dává lepší výsledky.

Poté, co se odečte přenášený materiál ΔS od tloušťky vrstvy, sníží se o tuto hodnotu i celková výška terénu. Nyní už zbývá jen materiál ΔS spravedlivě rozdělit všem sousedům, kteří přispěli do součtu *sum* a to následujícím způsobem:

$$h_i = h_i + \frac{\Delta S \cdot (h - h_i)}{sum} \qquad (4.3)$$

kde h_i je výška terénu souseda a h je výška terénu počítaného prvku. Nesmíme také zapomenout o přidělené množství materiálu zvýšit tloušťku vrstvy erodovaného materiálu. Všechny změny provádíme ihned, nikoliv až na konci kroku, pro všechny prvky najednou.

Krok tepelné eroze je tímto hotov, nyní už je poznámka k hledání sousedů okrajových prvků mapy. Zde je několik možností, jak toto řešit, například zvětšením mapy a zdvojením okrajových prvků, nebo ošetřením podmínkami. Zdvojení okrajových prvků sice sníží množství vyhodnocovaných podmínek při výpočtu, ale na druhou stranu zase vyžaduje velké množství paměti navíc (hlavně u větších map). Proto jsem se rozhodl použít ošetření podmínkami.

4.3.2. **Vodní eroze**

Krok vodní eroze se skládá z několika částí. Nejprve se na prvky mapy nanese nová voda, což je řešeno mimo metodu kroku vodní eroze kvůli vodním zdrojům, které přidávají novou vodu pouze před začátkem eroze. Nanášení vody je jednoduché, u prvků určených vodním zdrojem se zvýší objem vody o stanovené množství. U deště je to stejné, jen prvky nejsou určeny přímo. Jsou určeny pouze procentem pokrytí mapy, ze kterého se vypočítá počet prvků, na které se bude voda nanášet. Před každým krokem vodní eroze se pak vygeneruje počet pozic odpovídající určenému počtu prvků a na tyto pozice se voda nanese.

Další částí je transport vody. Ten byl detailně popsán v kapitole 3.2.4. a protože je velmi důležitý, uvedu zde část zjednodušeného kódu bez deklarací proměnných a bez ošetřených okrajových prvků:

```
for (int y = 0; y < height; y++) {
  for (int x = 0; x < width; x++) {
    h = heightMap[x, y].waterHeight; //výška vodní hladiny
    flow = heightMap[x, y].flow; //výstupní toky
    flow[0] += t * G * (h - heightMap[x - 1, y].waterHeight);
    flow[1] += t * G * (h - heightMap[x + 1, y].waterHeight);
    flow[2] += t * G * (h - heightMap[x, y - 1].waterHeight);
    flow[3] += t * G * (h - heightMap[x, y + 1].waterHeight);
    if (flow[0] < 0) //udržujeme pouze výstupní toky, vstupní nulujeme
      flow[0] = 0f;
    if (flow[1] < 0)
      flow[1] = 0f;
    if (flow[2] < 0)
      flow[2] = 0f;
    if (flow[3] < 0)
      flow[3] = 0f;
    outV = t * (flow[0] + flow[1] + flow[2] + flow[3]);
    wVolume = heightMap[x, y].waterVolume; //objem vody
    if (outV > wVolume) { //kontrola dostatečného objemu vody
      coef = wVolume / outV; //faktor zmenšení
      flow[0] *= coef;
      flow[1] *= coef;
      flow[2] *= coef;
      flow[3] *= coef;
    }
  }
}

for (int y = 0; y < height; y++) {
  for (int x = 0; x < width; x++) {
    flow = heightMap[x, y].flow; //výstupní toky
    sum = heightMap[x + 1, y].flow[0] - flow[0];
        + heightMap[x - 1, y].flow[1] - flow[1];
        + heightMap[x, y + 1].flow[2] - flow[2];
        + heightMap[x, y - 1].flow[3] - flow[3];
    //zapamatování starého objemu vody potřebného pro výpočet rychlosti
    oldWaterVolume[x, y] = heightMap[x, y].waterVolume;
    newWaterVolume = oldWaterVolume[x, y] + t * sum; //nový objem vody
    if (newWaterVolume < 0f)
      newWaterVolume = 0f;
    heightMap[x, y].waterVolume = newWaterVolume;
    //aktualizace výšky vodní hladiny
    heightMap[x, y].waterHeight = heightMap[x, y].height + newWaterVolume;
  }
}
```

Je zde opravdu důležité nulovat záporné (vstupní) toky, protože jinak by bylo velmi problematické ošetřit případy, kdy je potřeba z buňky odvést více vody, než obsahuje.

Po transportu vody přichází na řadu eroze. Nejprve potřebujeme určit vektor rychlosti vody v horizontálním směru. Jeho jednotlivé složky určíme z průtoků vody danou buňkou v příslušných směrech podle vzorce (3.46) z kapitoly 3.2.4. Zde je potřeba dávat pozor na průměrnou výšku vody $\overline{d}(x,y)$, která může být nulová. V tomto případě pak vzorec pro výpočet rychlosti nepoužijeme, ale rovnou rychlosti přiřadíme vektor s nulovou délkou.

Dále potřebujeme určit úhel sklonu terénu v daném místě $\alpha(x,y)$, nebo lépe sinus tohoto úhlu. Ten určíme následujícím vztahem:

$$\sin\alpha(x,y) = \max\left(\frac{d_{\max}}{\sqrt{1+d_{\max}^2}}, prah\right) \qquad (4.3)$$

kde d_{max} je rozdíl výšky terénu daného prvku a výšky terénu jeho nejnižšího souseda a *prah* je prahová hodnota určující minimální hodnotu $\sin\alpha(x,y)$, aby i u rovného terénu docházelo k erozi (v našem případě byl zvolen $prah = 0{,}01$).

Nyní, když známe všechny proměnné, spočítáme velikost kapacity vody *C* podle vztahu (3.47), porovnáme je s rozpuštěným sedimentem a rozhodneme, zda se bude materiál na daném prvku usazovat, nebo rozpouštět.

Vrstvy jsou u každého prvku uloženy v poli *layers*, kde prvek s indexem 0 je vyhrazen pro vrstvu erodovaného materiálu a ostatní vrstvy terénu jsou uloženy v dalších prvcích seřazeny od nejvyšší po nejnižší, kde nejnižší (a tudíž poslední) vrstva je vždy základní vrstva s nekonečnou tloušťkou. Pokud dojde k rozpouštění a rozpuštěné množství materiálu překročí tloušťku vrstvy, tak pokud se nejedná o vrstvu erodovaného materiálu, je vrstva smazána a tudíž vyřazena z pole *layers*. Tato vrstva už nemůže být v budoucích krocích nikdy obnovena, protože jediná vrstva, která se přenáší, je vrstva erodovaného materiálu. Ostatní vrstvy mohou být pouze na erodovaný materiál přeměněny, ten se ale už na jiné vrstvy nepřemění. Pokud je mazána vrstva erodovaného materiálu, je pouze její tloušťka nastavena na nulu, ale z pole se neodstraňuje, protože v budoucnu se do ní může nový materiál opět přidat. Jako vrchní vrstva je pak nastavena vrstva, která je těsně pod smazanou vrstvou, a s jejími parametry se znovu spočítá množství rozpouštěného materiálu. Toto množství se ještě navíc vynásobí poměrem neuspokojené části rozpuštěného materiálu z předchozí vrstvy.

Například pokud je v původní horní vrstvě spočítáno, že se musí rozpustit 10 cm materiálu, ale tloušťka vrstvy je pouze 6 cm, rozpustí se 6 cm a vrstva se odstraní. Tím se uspokojí 60 % požadavku na rozpuštění a zbylých 40 % se vezme z množství určeného

k rozpuštění při výpočtu s novou horní vrstvou. Jak vidíme, nerozpustí se pouze zbylé 4 cm z následující vrstvy, ale 40 % z nově rozpuštěného množství. Pokud by nová vrstva měla stejnou rozpustnost jako vrstva původní, opět by vyšlo k rozpuštění 10 cm materiálu a tudíž by se rozpustily zbylé 4 cm. Pokud by ale například nová vrstva byla zcela nerozpustná, bude určeno k rozpuštění 0 cm a z nich se vezme 40 %, což je 0 cm. Žádný další materiál se tedy z nové horní vrstvy nerozpustí. Pokud by ani tloušťka této nové vrstvy nepostačovala k celkovému uspokojení rozpouštěného materiálu, je opět odstraněna a počítá se stejným způsobem dále, dokud není požadavek uspokojen (vzhledem k tomu, že tloušťka poslední vrstvy je nekonečná, bude uspokojen vždy). Tedy pokud budeme předpokládat, že měla stejné parametry jako předešlá vrstva a požadavek byl opět 10 cm, z nichž se měly rozpustit už jen čtyři. Jenže tloušťka této vrstvy je jenom 2 cm, což je 50 % požadavku, celkově je tedy uspokojeno 80 % a zbylých 20 % se zkusí uspokojit další vrstvou.

Co se týká usazování, zde žádný problém není, usazuje se vždy jen erodovaný materiál, tedy vrstva s indexem nula. Nesmíme zapomenout usazený materiál odečíst od rozpuštěného a také vždy když změníme šířku vrstvy, musíme změnit výšku terénu a obráceně.

Poté, co dokončíme rozpouštění nebo usazování, musíme přesunout také sediment podle (3.50). Jedná se pouze o bilineární interpolaci mezi čtyřmi prvky výškové mapy. Zde žádný problém nevidím.

Poslední částí je vypaření vody. To provedeme podle vztahu (3.51). Pokud objem vody klesne pod prahovou hodnotu *minWaterVolume*, nastavíme objem vody na nulu. Tím končí jeden krok vodní eroze.

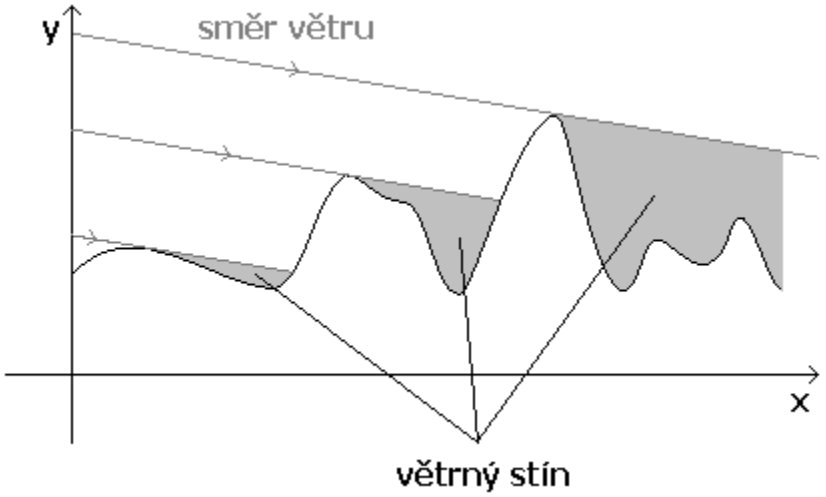
4.3.3. Větrná eroze

Poslední erozní metodou, kterou jsme si ještě nepředstavili, je krok větrné eroze. Jedná se o můj vlastní algoritmus, který nikde v předešlém textu popsán nebyl, proto jeho popis bude podrobnější, než byl popis předchozích metod, které vycházely z teoretické části práce.

Jedná se o algoritmus, který není fyzikálně založený, snaží se pouze reálné větrné erozi co nejvíce vizuálně přiblížit. Pracuje opět jako předešlé dva algoritmy s vrstvenou datovou strukturou.

Pro zjednodušení si představme terén ve dvourozměrném prostoru, kde na ose x je znázorněn index výškové mapy a na ose y je zobrazena výška terénu prvku s daným indexem (viz obrázek 4.3). Pokud budeme předpokládat, že vítr vane stále jedním směrem a pokud narazí na nějakou překážku (například horu), postupuje po jejím povrchu stále ve směru svého

horizontálního směru vátí tak dlouho, dokud mu překážka brání foukat ve svém přirozeném směru. Proto, abychom mohli jednoduše určit, zda se některý objekt nebo prvek výškové mapy nachází ve větrném stínu, můžeme postupně procházet terén po směru větru a pamatovat si minimální výšku, do které vítr fouká. Na začátku nastavíme tuto minimální výšku větru na výšku terénu. Při postupu na další prvek výškové mapy vždy snížíme minimální výšku větru o hodnotu, která udává, o jakou výšku vítr klesne mezi dvěma prvky při přirozeném foukání. Pokud se budeme nacházet na prvku, jehož výška terénu je vyšší než minimální výška větru, prvek se nachází na návětrné části a minimální výšku větru musíme zvýšit na výšku terénu tohoto prvku, protože se vítr pohybuje po povrchu terénu. Pokud se budeme nacházet na prvku, jehož výška je nižší než minimální výška větru, nacházíme se ve větrném stínu.



Obrázek 4.3: Znázornění větrného stínu ve 2D

Ve větrném stínu vítr nijak nepůsobí, může sem pouze zanášet materiál, který zachytil cestou. Na návětrné straně vítr rozrušuje svrchní materiál a zachytává jej. Některé částice se mohou rovněž na návětrné straně usadit, například těžké částice, které nemohou být přenášeny příliš daleko. Množství erodovaného materiálu závisí převážně na odolnosti materiálu proti větru a na ploše, na kterou vítr dopadá. Čím je výškový rozdíl mezi současným a následujícím prvkem větší, tím je i větší plocha a tím silnější je eroze. Přesný vztah pro výpočet množství erodovaného materiálu ΔS_e vypadá následovně:

$$\Delta S_e = \min\left(R \cdot E \cdot (\Delta h_{wind} - \Delta h_{barrier}), C - s\right) \quad (4.4)$$

kde R je koeficient odolnosti materiálu proti větru z intervalu od 0 do 1 (0 znamená zcela odolný), E je koeficient eroze (*windErosionCoef*), který určuje sílu větrné eroze (opět

z intervalu od 0 do 1), Δh_{wind} je rozdíl aktuální výšky terénu h_{akt} a minimální výšky větru, $\Delta h_{barrier}$ se spočítá takto:

$$\Delta h_{barrier} = \max(0, \min(h_{next} - h_{akt}, \Delta h_{wind})) \quad (4.5)$$

kde h_{next} je výška následujícího prvku (tedy prvku, na který se přejde po dopočítání aktuálního). Člen $C - s$ v (4.4) určuje maximální množství materiálu, které ještě může vítr zachytit, než dojde k jeho nasycení. C zde určuje kapacitu větru (*windCapacity*) a s množství již ve větru zachyceného materiálu.

Spočítané množství erodovaného materiálu se odečte od tloušťky vrchní vrstvy daného prvku. Pokud vrstva neobsahuje dostatečné množství materiálu, provedou se stejné operace jako u vodní eroze v předchozí kapitole. Nakonec se množství erodovaného materiálu odečte i od výšky daného prvku.

Po kroku eroze přichází krok usazování, které zde rozdělíme na dva druhy: na usazování ve větrném stínu a usazování na návětrné straně. Ve větrném stínu je usazování jednoduché, usadíme část ΔS_{sd} materiálu zachyceného ve vzduchu. Jak velký díl unášeného materiálu bude usazen, určuje koeficient *windShadowDeposition* z intervalu od 0 do 1:

$$\Delta S_{sd} = s \cdot windShadowDeposition \quad (4.6)$$

Usazování na návětrné straně je komplikovanější, protože se zde snažíme zachytit návany materiálu před nějakou překážkou. Množství usazeného materiálu ΔS_{wd} spočítáme takto:

$$\Delta S_{wd} = s \cdot \min\left(\frac{h_{next} - h_{akt}}{windTalus} + windDeposition, 1\right) \quad (4.7)$$

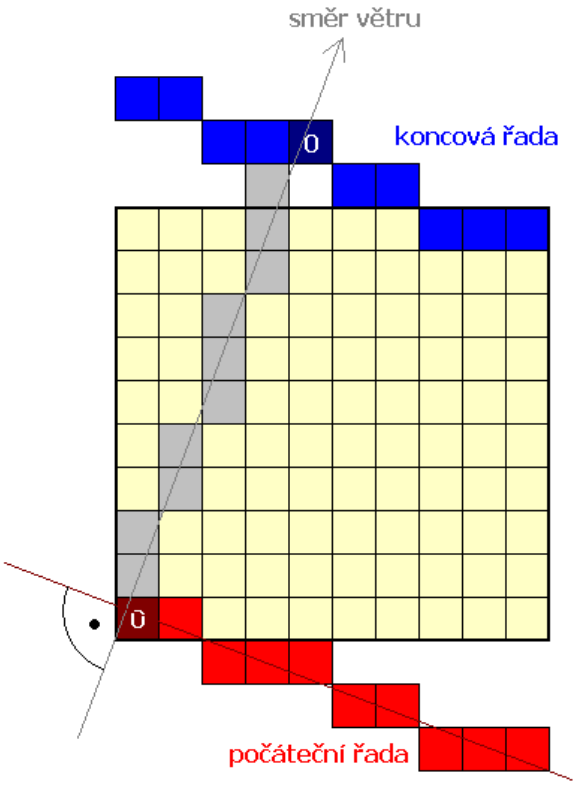
Kde *windDeposition* je koeficient z intervalu od 0 do 1, který ovlivňuje míru usazování materiálu na návětrné straně, a *windTalus* určuje rozdíl výšek sousedních prvků při maximálním svahu. Tento rozdíl výšek můžeme ovlivnit vlastností *WindTalus*, která se zadává ve stupních a určuje úhel maximálního svahu.

Nyní jsme si vysvětlili funkci navržené větrné eroze na zjednodušeném příkladu. U dvourozměrné výškové mapy ale můžeme takového zjednodušení dosáhnout také a to řezem terénu vertikální rovinou. Pokud se zaměříme pouze na indexy výškové mapy, jedná se o řez úsečkou. A jelikož máme výškovou mapu, která se skládá z jednotlivých buněk a je tedy diskrétní, vytvoříme řez pomocí rasterizované úsečky (viz [ZBSF04]) rovnoběžné s horizontálním směrem větru.

Pro pokrytí celé výškové mapy o rozměrech $n \times n$ potřebujeme minimálně n a maximálně $2n$ takovýchto řezů. Hlavní věc, kterou od rasterizovaných úseček vyžadujeme, je, aby

pokryly úplně celou výškovou mapu a aby se nikde nepřekrývaly, což může být problém, pokud chceme, aby řezy byly rovnoběžné se směrem větru a také abychom postupovali po řadách kolmých ke směru větru. Pokud bychom nepostupovali po kolmých řadách, ale například po řadách vždy ve směru jedné z os, tvořila by se místa z části zerodovaná ze směru větru a z části zarovnaná s jednou z os.

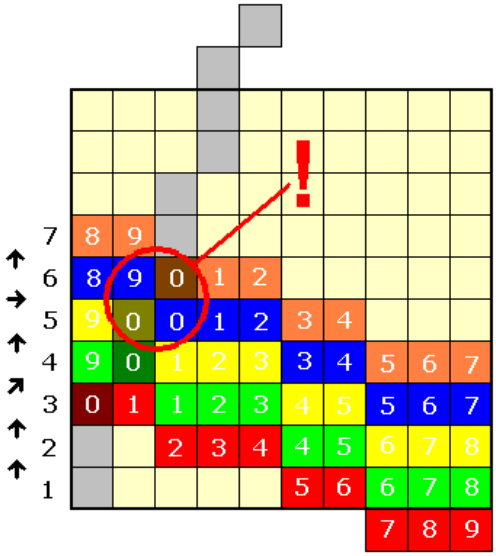
Na začátku kroku si tedy vytvoříme počáteční řadu, která je kolmá ke směru větru, je široká stejně jako výšková mapa a alespoň jeden její prvek se nachází na okrajovém prvku výškové mapy, ostatní mimo mapu. Pro každý její prvek si pamatujeme minimální výšku větru a množství materiálu zachyceného větrem. Tvar této řady se během kroku nemění a řada se celá postupně posouvá ve směru řídící osy směru větru, dokud se neprojdou všechny prvky výškové mapy, jak naznačuje obrázek 4.4.



Obrázek 4.4: Procházení prvků výškové mapy – začátek a konec

Posun této řady mapou ale není jediný posun, ke kterému dochází. Uvnitř řady se ještě posouvají prvky tak, jak se posouvá rasterizovaná úsečka znázorňující směr větru. Na obrázku 4.4 je naznačena pozice indexu 0 na začátku a na konci procházení a vidíme, jak kopíruje směr větru. Ještě lépe je posouvání indexů v poli vidět na obrázku 4.5, kde je znázorněno několik po sobě jdoucích posunů (jsou odděleny barevně). Je zde zvýrazněna situace, na kterou si musíme dávat pozor. Pokud se zaměříme pouze na index nula (ostatní indexy se

automaticky posunují s ním), tak při každém sudém posunu větru do strany se index neposune o jednu pozici vpřed jako obvykle, ale pouze do strany, čímž „vybočí“ z úsečky směru větru a vrátí se na ní až v nadcházejícím posunu.



Obrázek 4.5: Procházení výškové mapy – znázornění posunu indexů

Při posouvání indexů dochází k situacím, kdy je prvek řady na jedné straně z výškové mapy vysunut a na druhé straně vložen. Zde si můžeme zvolit, jak se terén bude chovat. Můžeme například tuto situaci ignorovat a veškerý materiál se bude přenášet z jedné strany mapy na druhou, čímž sice nedojde ke ztrátě materiálu, ale nemusí to vypadat příliš dobře. Další možností je třeba materiál vynášený z mapy distribuovat rovnoměrně nebo náhodně. Vyzkoušel jsem spoustu možností, ale nakonec jsem se rozhodl materiál vynesený z mapy už z novu do mapy nevracet, protože toto řešení podle mého názoru dává nejlepší vizuální výsledky.

4.4. **Vizualizace terénu**

Pro vizualizaci terénu byla navržena třída *TerrainVisualizer*. Vizualizace ale nebyla hlavním cílem této práce a slouží spíše k vizuálnímu ověření správnosti erozních metod. Celý proces vizualizace je poměrně pomalý a nehodí se pro zobrazování v reálném čase.

Tato třída obsahuje čtyři metody, dvě pro vytváření textury a dvě pro vytváření geometrických modelů. První metoda *CreateHeightMapTexture* není určena přímo pro vizualizaci. Slouží pro vytvoření bitmapového obrázku výškové mapy daného terénu, kde černá barva představuje nejnižší hodnoty výškové mapy a bílá barva nejvyšší. Tato metoda

může být využita například pro nástroj pro zadávání vodních zdrojů, jak je tomu u ukázkové aplikace, nebo pro uložení vytvořené výškové mapy jako obrázku.

Druhou metodou pro vytváření textury je metoda *CreateTerrainTexture*. Tato metoda vytvoří texturu celého povrchu daného terénu, která pak následně může být použita při vizualizaci. Velikost této textury v bytech je možné nastavit vlastností *TextureSize*. Podmínkou ale je, že tato velikost nemůže být menší, než je čtyřnásobek velikosti výškové mapy. Pokud bude velikost nastavena menší, automaticky se zvětší. Výsledná textura se pak spočítá kombinací jednotlivých textur materiálů. Nezáleží pouze na tom, který materiál je na povrchu terénu, ale také na šířce vrstvy s tímto materiálem. Uživatel si může sám ovlivnit dvě prahové hodnoty vlastnostmi *MinErodedMaterial* a *MinThickness*. První určuje minimální tloušťku vrstvy erodovaného materiálu, která je ještě viditelná. Pokud je tloušťka erodovaného materiálu menší než tato prahová hodnota, jeho textura se vůbec nezobrazí a místo ní se zobrazí textura vrstvy ležící těsně pod ní. Druhá prahová hodnota *MinThickness* určuje minimální tloušťku vrstvy, kdy je zobrazena pouze textura materiálu vrchní vrstvy. Pokud je tloušťka vrchní vrstvy menší než tato hodnota, je vrchní vrstva částečně průhledná, tedy míchá se textura této vrstvy s texturou vrstvy ležící těsně pod ní. Míra viditelnosti horní vrstvy závisí na poměru tloušťky této vrstvy a prahové hodnoty *MinThickness*.

Metoda, která slouží k vytvoření modelu terénu, se jmenuje *CreateTerrainModel*. Tato metoda vytvoří z výškové mapy daného terénu množinu trojúhelníků a přiřadí jí materiál pro přivrácenou i odvrácenou stranu (myšleno podle orientace trojúhelníků). Pro odvrácenou stranu je použit materiál, který uživatel může nastavit vlastností *TerrainBackMaterial*. Materiál pro přivrácenou stranu je možné nastavit v parametru metody. Pokud nastaven není, je použit difúzní materiál s aktualizovanou texturou terénu.

Poslední metoda se jmenuje *CreateWaterModel* a slouží pro vytvoření modelu vody. Opět jako u předchozí metody se vytvoří množina trojúhelníků, tentokráte z výšky vodní hladiny, a přiřadí se jí materiál přivrácené strany *WaterMaterial* a materiál odvrácené strany *WaterBackMaterial*. Vrcholy trojúhelníků se nemusejí vyskytovat v každém prvku výškové mapy, jako tomu bylo u vytváření terénu, vyskytují se pouze tam, kde se vyskytuje voda a zároveň její objem je větší než prahová hodnota určená vlastností *MinWaterVolume*. Pokud by objem vody byl menší než tato prahová hodnota, je s prvkem zacházeno, jako by zde žádná voda nebyla a tím pádem se zde nezobrazí. Vytváření modelu vody je poměrně jednoduché a v některých situacích může dojít k chybnému zobrazení. Například pokud se voda vyskytuje v prvku, který má pouze jednoho souseda s vodou, voda se zde nezobrazí, protože nevytvoří trojúhelník. To samé platí i pro prvky s vodou ležící za sebou v jedné přímce, které okolo sebe

mají jen prvky bez vody. Jak ale již bylo uvedeno dříve, tato vizualizace slouží spíše jen pro orientaci, proto si zde takovéto drobné chyby zobrazení můžeme dovolit. Je to cena za jednoduchost implementace a nepříliš velikou náročnost výpočtu.

5. Testování

V této kapitole se pokusíme ověřit správnost funkcionality vytvořené knihovny vizuální kontrolou terénu a porovnat doby výpočtu pro různě veliké výškové mapy a také tyto doby srovnat s jinými implementacemi z referovaných článků. Testy byly prováděny na počítači s procesorem Intel Core 2 Duo E8400 (3.0 GHz), operační pamětí 2 GB a operačním systémem Microsoft Windows XP SP2.

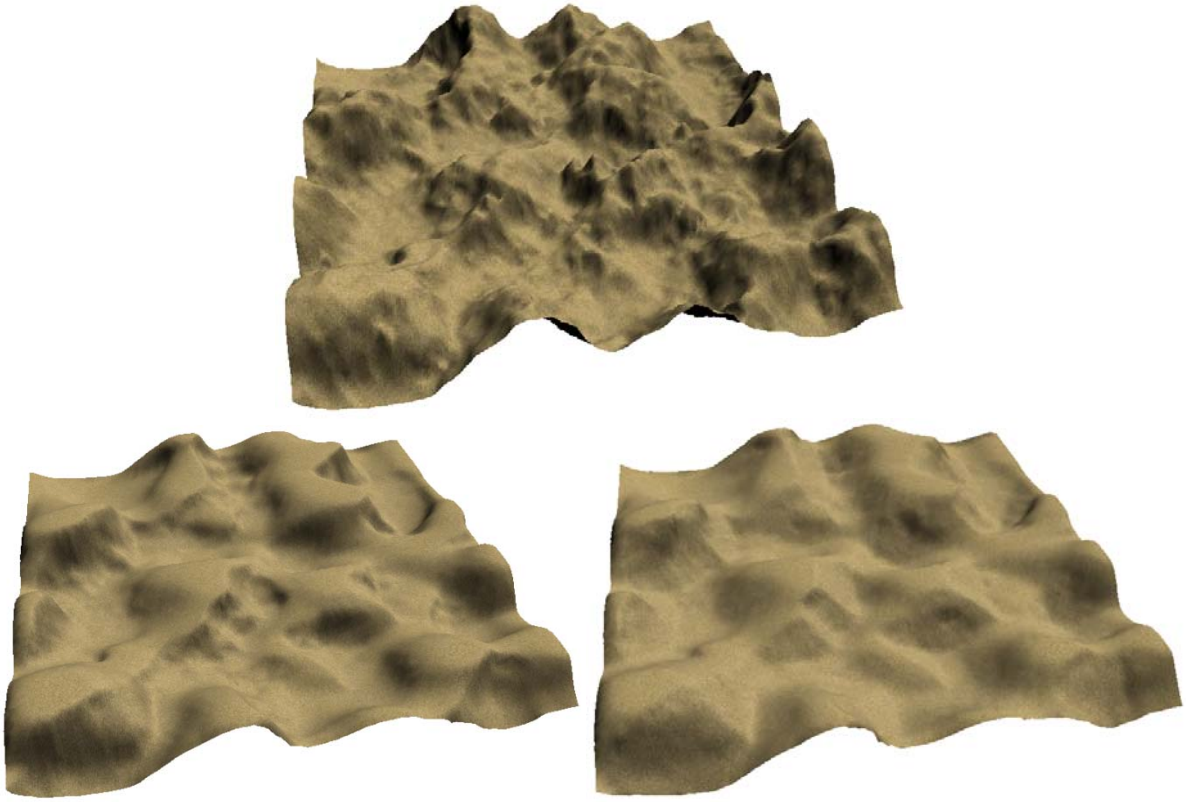
Nejprve začneme s určením doby výpočtu. Měření bylo provedeno pro výškové mapy s rozlišením 256 x 256, 512 x 512 a 1024 x 1024. Výsledky znázorňuje tabulka 5.1.

	Doba kroku eroze v [ms]		
rozlišení	tepelná	větrná	vodní
256 x 256	23,4	31,2	48,2
512 x 512	171,8	210,9	262,5
1024 x 1024	1132,8	1087,5	1853,9

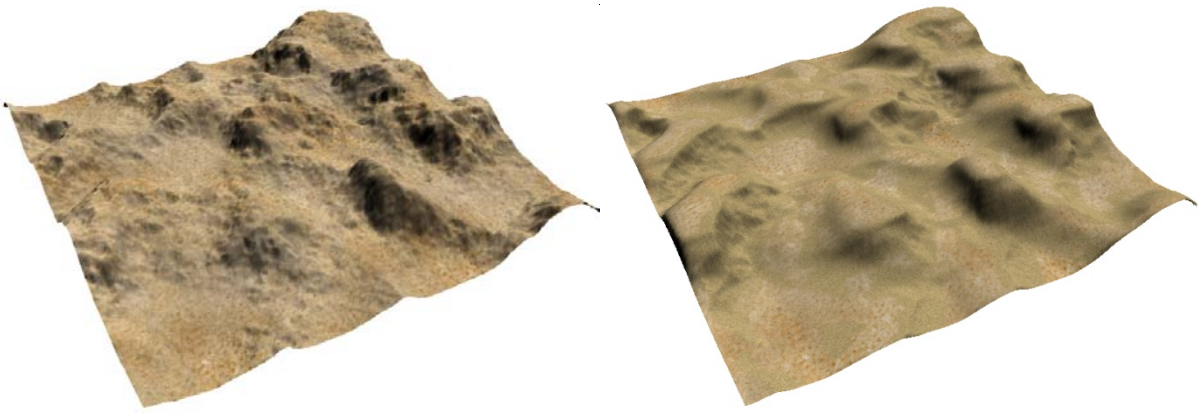
Tabulka 5.1: Doba kroku eroze

Je vidět, že s rostoucí velikostí výškové mapy roste doba výpočtu jednoho kroku nelineárně a pro mapy s rozlišením od 512 x 512 přestává být eroze proveditelná v reálném čase, protože obvykle potřebujeme provést velké množství kroků. Pokud srovnáme doby vodní eroze s [SBBK08] nebo s [MDH07] jsou naše doby výrazně delší. To je způsobeno tím, že v obou těchto pracích je algoritmus naimplementován pro GPU, zatímco my využíváme k výpočtu CPU.

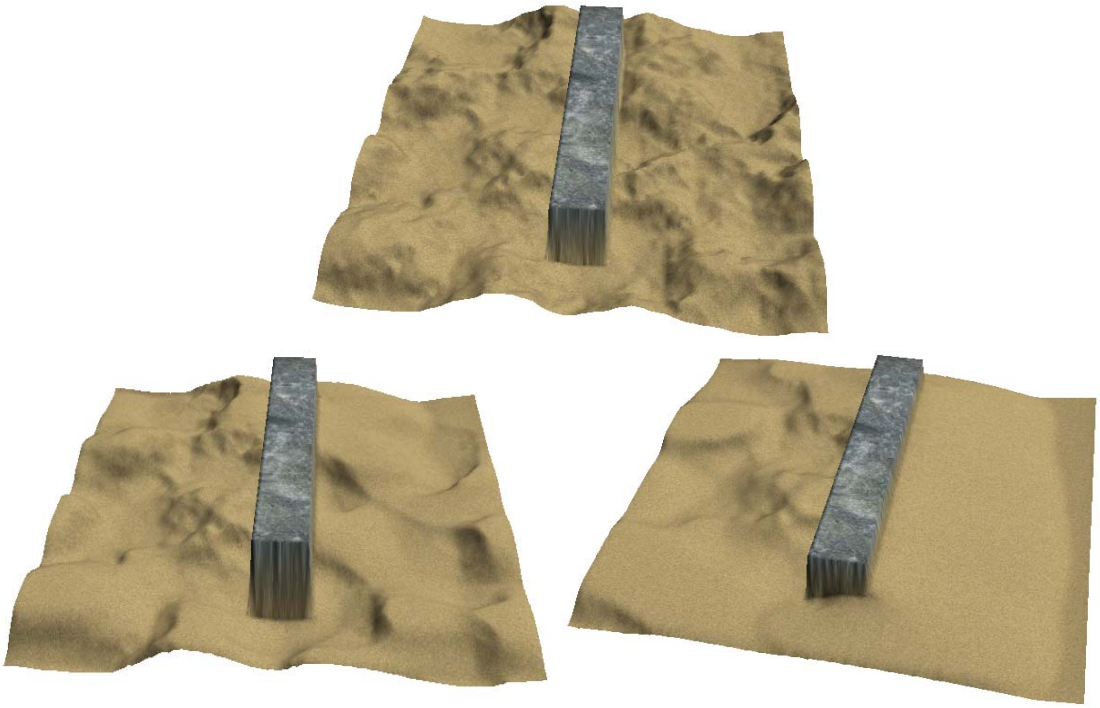
Nyní si uvedeme několik příkladů úprav terénu v našem demonstračním programu. Jako první načteme vygenerovaný terén a aplikujeme na něj větrnou erozi. Výsledek můžeme vidět na obrázku 5.1. Veškeré nastavení bylo ponecháno tak, jak je přednastavené. Na obrázku 5.2 je můžeme vidět srovnání před a po větrné erozi na jiném terénu. Na obrázku 5.3 je pak ukázka, jak to vypadá, pokud se větru do cesty postaví odolná překážka.



Obrázek 5.1: nahoře – původní terén; vlevo dole – terén po aplikaci 600 kroků větrné eroze; vpravo dole – větrná eroze kombinovaná s tepelnou v poměru 50:1



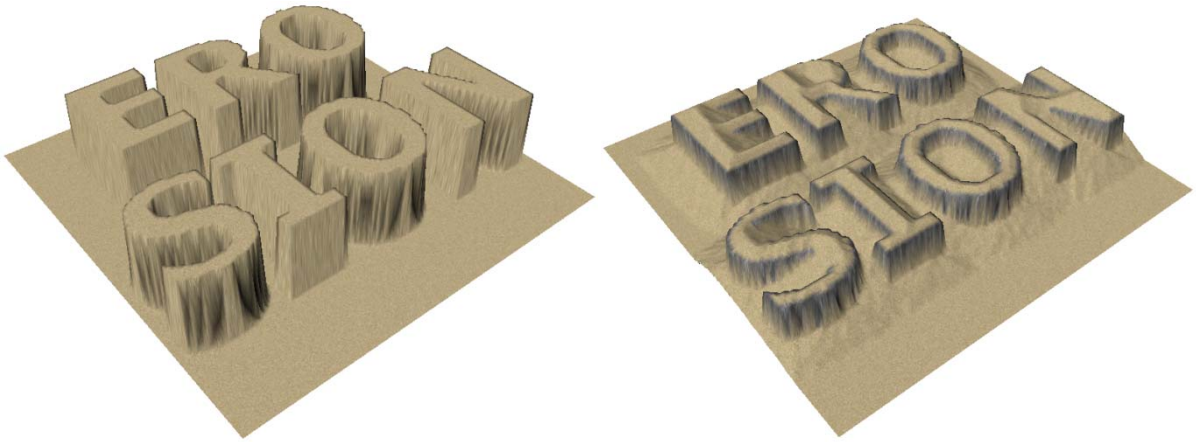
Obrázek 5.2: vlevo – původní terén; vpravo – terén po aplikaci 500 kroků větrné eroze



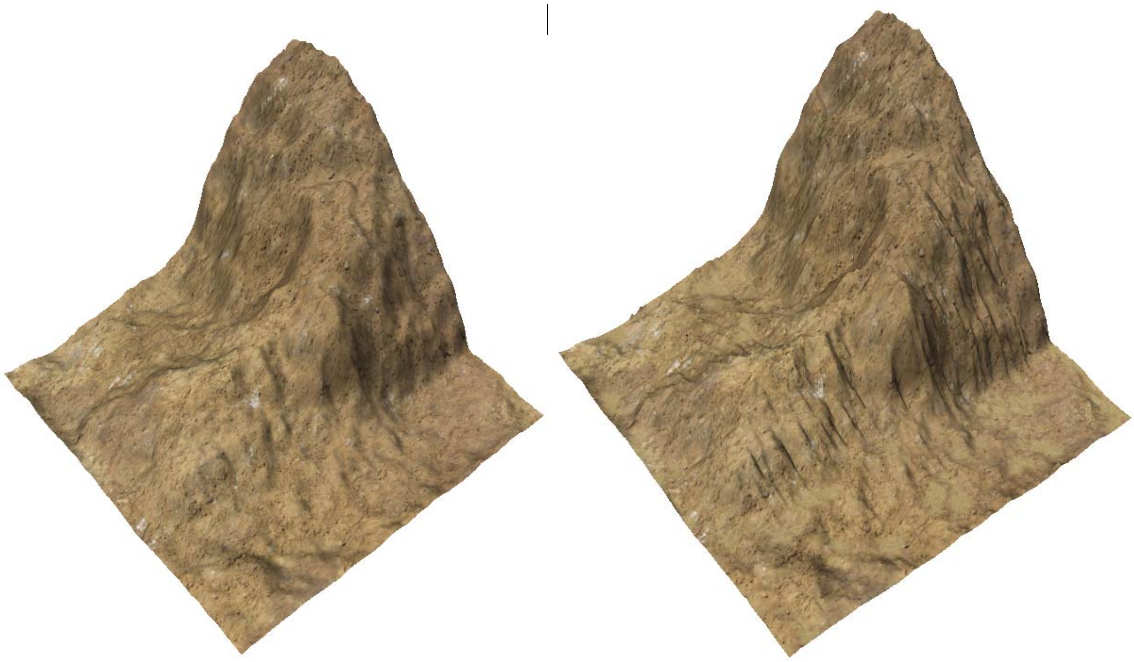
Obrázek 5.3: nahoře původní terén, vlevo dole terén po několika krocích a vpravo dole terén po větším množství kroků.

Příklad na tepelnou erozi si uvedeme pouze na umělém terénu (obrázek 5.4), kde je tato eroze lépe vidět. Dolní části písmen slova „EROSION“ se skládají z materiálu, který je zcela odolný vůči tepelné erozi.

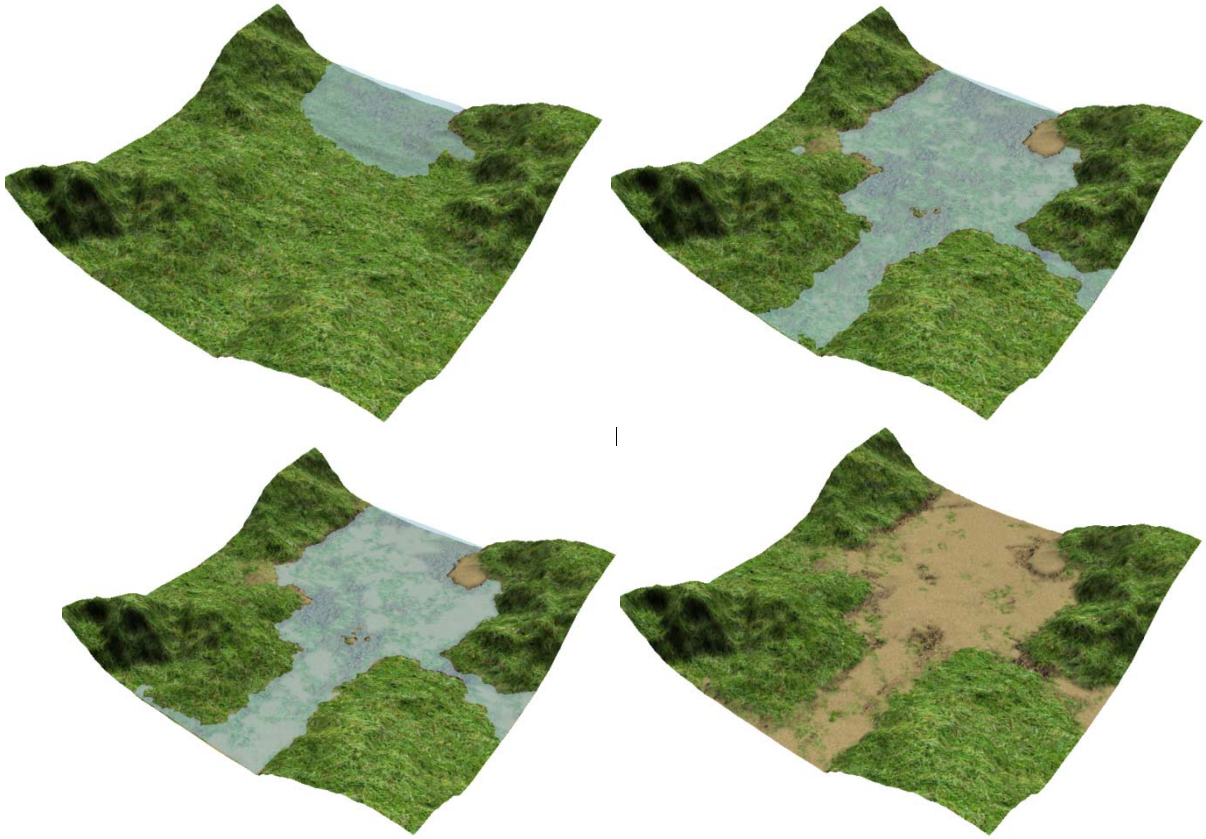
Obrázek 5.5 ukazuje erozi hory způsobenou deštěm. Pokud chceme, aby déšť vytvářel ve svazích hory úzké ostré rýhy, musíme nastavit déšť na hodně silný s malými kapičkami a také zvýšit odolnost materiálu hory proti vodní erozi. Během testování jsem zjistil, že déšť vytváří rýhy vždy rovnoběžné s některou z os, což asi není úplně správné chování. Nepodařilo se mi ale zjistit, jak tuto chybu nějak jednoduše opravit, proto zde na ní alespoň upozorňuji.



Obrázek 5.4: vlevo – původní terén; vpravo – terén po aplikaci 100 kroků tepelné eroze



Obrázek 5.5: vlevo – původní terén; vpravo – terén erodovaný deštěm



Obrázek 5.6: Průběh vodní eroze

Na obrázku 5.6 je vidět průběh vodní eroze řekou. Na prvním obrázku do údolí přitéká voda, další obrázky pak byly postupně pořizovány po 500 krocích vodní eroze.

6. Závěr

Hlavní náplní teoretické části této práce bylo prozkoumávání širokého spektra různých erozních algoritmů, jejich porovnávání a výběr těch nejvhodnějších pro pokrytí co nejširší škály terénních úprav. Mou snahou bylo především vybrat zástupce tří základních typů erozí, jimiž jsou vodní, větrná a tepelná eroze.

Zástupce vodní a tepelné eroze jsem vytvořil kombinací a úpravou několika dosavadních prací. O větrné erozi se mi ale příliš informací najít nepodařilo, proto jsem navrhnul vlastní erozní algoritmus, který stejně jako oba předchozí nepracuje pouze s výškovou mapou, ale obohacuje ji o podložní vrstvy terénu, které mohou mít nastaveny různé vlastnosti a lépe tak přiblížit skutečný terén.

Vytvořil jsem knihovnu *ErosionLibrary.dll*, která umožňuje z výškových map vytvářet vrstvený terén, na ten pak aplikovat erozi a terén následně zobrazit. U eroze je možné nastavit proporce jednotlivých typů erozí, podle kterých se pak na terén aplikují. Knihovna byla vytvořena pro WPF (Windows Presentation Foundation). Ke knihovně byl vytvořen i ukázkový program, který demonstruje všechny její funkce.

Hlavním směrem vylepšení této práce by mohlo být urychlení výpočtu jednotlivých erozí implementací pro GPU, dále také vylepšení nebo předělání vizualizační třídy, která není příliš vhodná pro zobrazování v reálném čase. Možným rozšířením by mohlo být doimplementování vodní eroze založené na rozpustnosti nebo přidání některého dalšího, ne tolik běžného, typu eroze. Také by se mohlo umožnit do terénu přidávat i vrstvy, které neobsahují pevné látky, ale například vodu nebo plyn, čímž by přibyla možnost simulovat i různé převisy a jeskyně, které klasické výškové mapy nejsou schopny zobrazit.

Literatura

[ASA07] Anh N. H., Sourin A., Aswani P. *Physically based hydraulic erosion simulation on graphics processing unit*. Proceedings of the 5th international Conference on Computer Graphics and interactive Techniques in Australia and Southeast Asia. GRAPHITE '07. ACM, New York, NY, USA, 257-264, 2007.

[BA05] Belhadj F., Audibert P. *Modeling landscapes with ridges and rivers*. VRST ’05: Proceedings of the ACM symposium on Virtual reality software and technology, ACM Press, New York, NY, USA, 151–154, 2005.

[Bel07] Belhadj F. *Terrain Modeling: A Constrained Fractal Model*. Afrigraph 2007, Grahamstown, South Africa, October 29–31, pp. 197-204, 2007.

[Ben07] Beneš B. *Real-time erosion using shallow water simulation*. VRIPHYS’07: 4th Workshop in Virtual Reality Interactions and Physical Simulation, 2007.

[BF01] Beneš B., Forsbach R. *Layered data representation for visual simulation of terrain erosion.* SCCG’01: Proceedings of the 17th Spring conference on Computer graphics, IEEE Computer Society, p. 80, 2001.

[BFo01] Beneš B., Forsbach R. *Parallel implementation of terrain erosion applied to the surface of Mars*. AFRIGRAPH’01: Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation, ACM Press, New York, NY, USA, 53–57, 2001.

[BF02] Beneš B., Forsbach R. *Visual Simulation of Hydraulic Erosion*. In Journal of WSCG 2002, vol. 10. N., 2002.

[BR04] Beneš B., Roa T. *Simulating Desert Scenery*. WSCG’2004, Plzen, Czech Republic, February 26, 2004.

[BTHB06] Beneš B., Těšínský V., Hornyš J., Bhatia S. K.. *Hydraulic erosion*. Computer animation and virtual worlds, 2006.

[CMF98] Chiba N., Muraoka K., Fujita K. *An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain scenery*. The Journal of Visualization and Computer Animation, 9:185–194, 1998.

[DEJP99] Dorsey J., Edelman A., Jensen H. W., Pedersen H. K. *Modeling and rendering of weathered stone*. Proc. of SIGGRAPH ’99, pp. 225–234, 1999.

[KM90] Kass M., Miller G. *Rapid, stable fluid dynamics for computer graphics.* Proc. of SIGGRAPH ’90, 1990.

[KMN88] Kelley A. D., Malin M. C., Nielson G. M. *Terrain Simulation Using a Model of Stream Erosion*. Computer Graphics, 22(4), pp. 263–268, 1988.

[LM93] Li X., Moshell M. *Modeling soil: Realtime dynamic models for soil slippage and manipulation*. Proc. of SIGGRAPH ’93, pp. 361–368, 1993.

[MBS97] Marák I., Beneš B., Slavík P. *Terrain Erosion Based on Rewriting of Matrices*. Proceedings of The Fifth International Conference in Central Europe on Computer Graphics and Visualization - WSCG, pp.341-351, 1997.

[MDH07] Mei X., Decaudin P., Hu B.-G. *Fast hydraulic erosion simulation and visualization on GPU*. Proc. of Pacific Graphics, pp. 47–56, 2007.

[MFC06] Maes M. M., Fujimoto T., Chiba N. *Efficient animation of water flow on irregular terrains*. Proc. of GRAPHITE ’06, pp. 107–115, 2006.

[MKM89] Musgrave F. K., Kolb C. E., Mace R. S. *The Synthesis and Rendering of Eroded Fractal Terrains*. Computer Graphics, 23(3):11–1–11–9, 1989.

[NWD05] Neidhold B., Wacker M., Deussen O. *Interactive physically based Fluid and Erosion Simulation*. Proceedings of Eurographics Workshop on Natural Phenomena 2005, vol. 1, pp. 25–32, 2005.

[OH95] O’Brien J., Hodgins J. K. *Dynamic simulation of splashing fluids*. Proceedings of Computer Animation’95, pp. 198–205, 1995.

[ON00] Onoue K., Nishita T. *A Method for Modeling and Rendering Dunes with Wind-ripples*. Proceedings of Pacific Graphics’00, pp. 427–428, 2000.

[ON03] Onoue K., Nishita T. *Virtual sandbox*. Proc. of Pacific Graphics, pp. 252–260, 2003.

[Ols04] Olsen, J. *Realtime Procedural Terrain Generation*. 31. 10. 2004.

[SBBK08] Šťava O., Beneš B., Brisbin M., Křivánek J. *Interactive Terrain Modeling Using Hydraulic Erosion*. Eurographics/ACM SIGGRAPH Symposium on Computer Animation, 2008.

[Sed07] Sedmihradský J. *Modelování eroze a deformací terénu*. Diplomová práce, KIV/ZCU 2007.

[SOH99] Sumner R. W., O’Brien J. F., Hodgins J. K. *Animating sand, mud, and snow*. Comp. Graph. Forum 18, 1999.

[WCMT07] Wojtan C., Carlson M., Mucha P. J., Turk G. *Animating corrosion and erosion.* Eurographics Workshop on Natural Phenomena, 2007.

[ZBSF04] Žára J., Beneš B., Sochor J., Felkel P. *Moderní počítačová grafika*. 1. vyd. Brno: Computer Press, 2004. ISBN 80-251-0454-0.

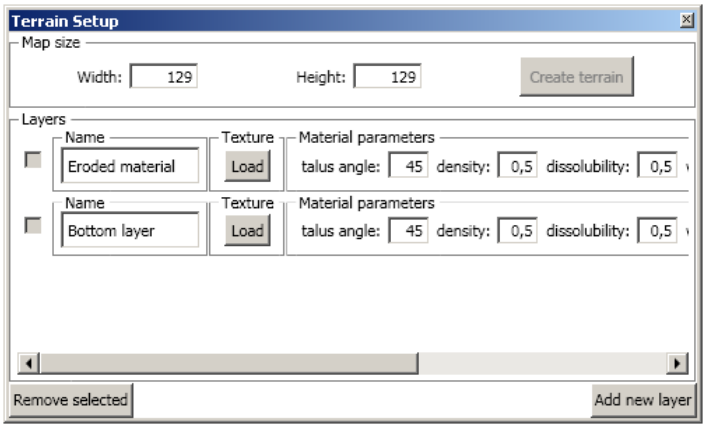
Příloha

Uživatelská příručka

Toto je uživatelská příručka k ukázkovému programu *Erosion.exe* demonstrujícímu funkcionalitu knihovny *ErosionLibrary.dll*. Program je napsán ve WPF (Windows Presentation Foundation) a ke svému spuštění vyžaduje operační systém Microsoft Windows Vista a novější. Program je možné spustit i pod Microsoft Windows XP, pokud je nainstalován Microsoft .NET Framework 3.5, zde se ale může v některých případech vyskytnout problém při zavírání oken aplikace.

Program se spouští souborem *Erosion.exe* a pro spuštění vyžaduje, aby byl ve stejné složce jako knihovna *ErosionLibrary.dll*. Po spuštění programu se zobrazí okno *Erosion*.

Nejprve, abychom mohli erozi použít, musíme vytvořit nějaký terén. To uděláme kliknutím na tlačítko *Load terrain…* Zobrazí se nám okno *Terrain Setup* (viz obrázek U.1).



Obrázek U.1: Okno Terrain Setup pro vytvoření terénu

V části *Map size* můžeme nastavit velikost vytvořeného terénu. V části *Layers* pak definujeme jednotlivé vrstvy terénu. Terén musí vždy obsahovat vrstvu *Eroded material*, která určuje vlastnosti erodovaného materiálu. Po vytvoření terénu se v něm tato vrstva nevyskytuje, proto se pro ni nenastavuje výšková mapa. Tato vrstva se v terénu objeví až po aplikaci eroze. Další povinná vrstva, kterou musí terén obsahovat, je vrstva *Bottom Layer*. Tato vrstva je základní vrstvou terénu (její rozměr udává rozměr celého terénu). Tloušťka této vrstvy je nekonečná a při použití více vrstev je tato vrstva vždy ta nejspodnější.

Pokud chceme k terénu přidat novou vrstvu, přidáme ji tlačítkem *Add new layer* vpravo dole. Nová vrstva se vždy přidá těsně pod vrstvu *Eroded material*, stane se z ní tedy vrchní

vrstva. Pokud bychom chtěli nějakou přidanou vrstvu odstranit, musíme ji označit zaškrtávacím políčkem vlevo od názvu vrstvy a stisknout tlačítko *Remove selected*.

Nyní nastavíme parametry jednotlivých vrstev. Jméno vrstvy slouží jen pro naši orientaci, z hlediska tvorby terénu je nepodstatné. Tlačítkem *Load* v části *Texture* načteme obrázek, který bude použit jako textura při vizualizaci dané vrstvy. Proto, aby mohl být terén vytvořen, musí mít každá vrstva nastavenu nějakou texturu.

V části *Material parameters* můžeme nastavit jednotlivé parametry materiálu vrstvy:

talus angle - určuje úhel maximálního sklonu svahu ve stupních při tepelné erozi

density - určuje odolnost materiálu vůči tepelné erozi z intervalu $\langle 0,1 \rangle$
(0 – eroze materiál vůbec neovlivní, 1 – maximální rozpad materiálu)

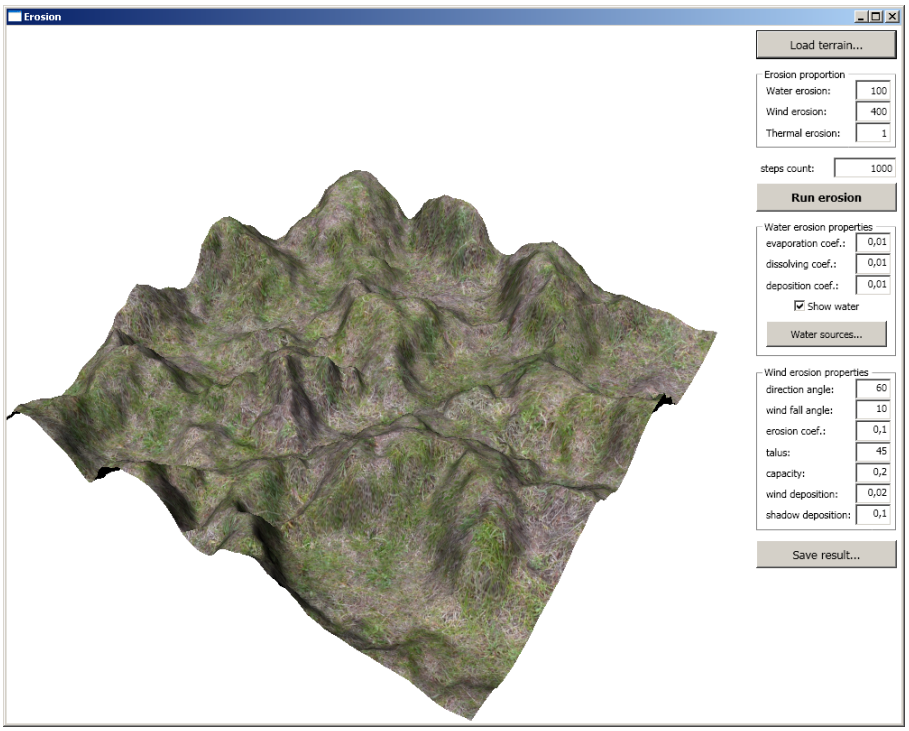
dissolubility - udává míru rozpustnosti materiálu ve vodě z intervalu $\langle 0,1 \rangle$
(0 – materiál je zcela odolný vůči vodní erozi, 1 - maximálně rozpustný)

wind resistivity - určuje odolnost vůči větru z intervalu $\langle 0,1 \rangle$
(0 – materiál je minimálně odolný vůči větru, 1 – vůči větru zcela odolný)

V části *Height map* nastavíme výškovou mapu pro každou vrstvu. Tlačítkem *Load* načteme soubor s výškovou mapou, který může být buď ve formátu **.raw*, nebo to může být jakýkoliv běžný obrázek, kde černá barva reprezentuje nejnižší výšku terénu a bílá barva nejvyšší. V případě obrázku se použijí jako rozměry mapy rozměry obrázku a nastavené hodnoty se ignorují. Proto, aby mohl být vytvořen terén, je nutné nastavit výškovou mapu pro každou vrstvu (kromě erodovaného materiálu).

Po nastavení všech parametrů můžeme terén vytvořit tlačítkem *Create terrain*.

Nyní by se měl v okně *Erosion* objevit vytvořený terén. Okno by mělo vypadat podobně jako na obrázku U.2. Kolečkem myši můžeme terén přibližovat nebo oddalovat. Stiskem levého tlačítkem na ploše, kde je terén zobrazený, a tažením myši terénem otáčíme, podobně pravým tlačítkem můžeme terén posouvat a tím měnit střed otáčení (ten je vždy uprostřed zobrazovací plochy a při vygenerování terénu se vždy nastaví na střed terénu).



Obrázek U.2: Okno Erosion, hlavní okno programu, slouží pro vizualizaci terénu

Nyní si popíšeme jednotlivá políčka v okně *Erosion*:

Water erosion - proporce vodní eroze

Wind erosion - proporce větrné eroze

Thermal erosion - proporce tepelné eroze

steps count - počet kroků eroze

evaporation coef. - vypařovací koeficient (0 – voda se nevypařuje, 1 – v jednom kroku se vypaří 100% vody)

dissolving coef. - koeficient rozpustnosti (0 – voda žádný materiál nerozpouští, 1 – rozpouštění je maximální)

deposition coef. - koeficient usazování (0 – sediment se neusazuje, 1 – usazování je maximální)

Show water - určuje, zda se zobrazí voda

directon angle - úhel směru větru ve stupních (0 – směr osy x)

wind fall angle - úhel klesání větru ve stupních v intervalu od 0 do 45

erosion coef. - koeficient větrné eroze (0 – žádná eroze, 1 – maximální eroze)

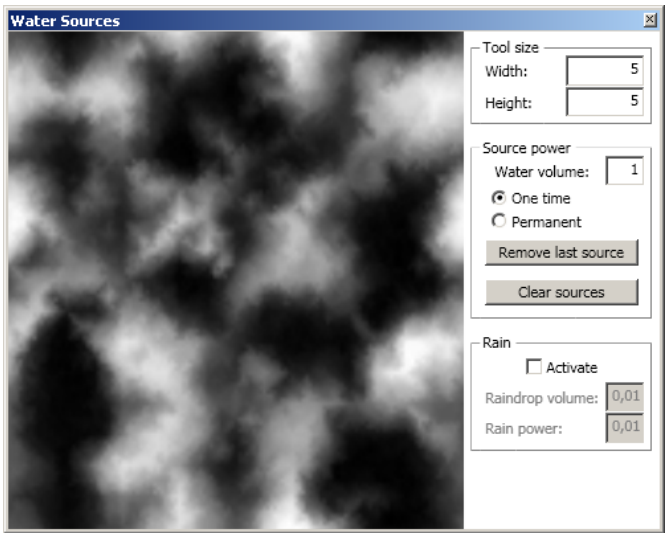
talus - úhel maximálního svahu (ve stupních) způsobeného návanem větru

capacity - kapacita větru (kolik maximálně materiálu může být ve vzduchu)

wind deposition - koeficient usazování na větru (0 – žádné, 1 – maximální)

shadow deposition - koeficient usazování ve větrném stínu (0 – žádné, 1 – maximální)

Pro nastavení vodních zdrojů pro vodní erozi slouží zvláštní okno *Water Sources*, které zobrazíme tlačítkem *Water sources*…. Toto okno je zobrazeno na obrázku U.3.



Obrázek U.3: Okno Water Sources pro nastavení vodních zdrojů

V levé části okna se zobrazí výšková mapa terénu, kam můžeme kreslit vodní zdroje. V pravé části okna si můžeme v části *Tool size* nastavit velikost kreslícího nástroje. V části *Source power* pak v políčku *Water volume* určujeme množství vody, které bude přibývat u kresleného zdroje (čím větší množství, tím tmavší barva kreslení). Dále si v této části můžeme zvolit, zda kreslený zdroj přidá dané množství vody pouze na začátku eroze (*One time*), nebo toto množství bude přidávat v každém kroku (*Permanent*). Tlačítkem *Remove last source* můžeme smazat naposledy nakreslený vodní zdroj (dalším stiskem pak zdroj kreslený předposlední atd.). Tlačítkem *Clear sources* smažeme všechny nakreslené vodní zdroje.

V části *Rain* můžeme zaškrtnutím políčka *Activate* přidat také déšť. Políčko *Raindrop volume* určuje velikost objemu kapky, která udává množství vody, které dopadne na povrch terénu v jedné buňce jako jedna dešťová kapka. Políčko *Rain power* určuje sílu deště z intervalu od nuly do jedné, která ovlivňuje počet kapek spadlých v jednom kroku vodní eroze. Pokud je síla nulová, nespadne žádná dešťová kapka, pokud je jednotková, spadne v jednom kroku počet kapek rovný celkovému počtu prvků výškové mapy. Neznamená to ale, že na každý prvek dopadne jedna kapka. Kapky dopadají náhodně a může se stát, že na některý prvek dopadne více kapek v jednom kroku a na některý žádná.

Simulaci eroze můžeme spustit tlačítkem *Run erosion*. Během výpočtu se zobrazí progress bar, který informuje o jeho průběhu. Výpočet lze přerušit stiskem tlačítka *Cancel* vedle tohoto progress baru.

Pokud se nám výsledný zerodovaný terén líbí, můžeme si ho pomocí tlačítka *Save result…* uložit jako výškovou mapu texturu (ztratí se tím ale informace o vrstvách terénu).