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

Diplomová práce

Modelování písčitého terénu pro virtuální realitu

Plzeň, 2009 Václav Purchart

Poděkování

Rád bych poděkoval všem, kteří se na projektu účastnili ať už v minulosti nebo v současné fázi projektu, za jejich nápady, inspiraci a pomoc. Můj dík patří těmto osobám: Doc. Dr. Ing. Ivana Kolingerová, Bc. Jan Kadlec, Ing. Jiří Sedmihradský, Ing. Bedřich Beneš, Ph.D., RNDr. Jan Flasar, Ph.D., Ing. Petr Vaněček, Ph.D. a Enkhtuvshin Dorjgotov.

Tato práce je podporována z projektu Triangularizované modely pro haptiku a virtuální realitu č. GA201/09/0097 a projektu Ministerstva školství, mládeže a tělovýchovy Modelování přírodních jevů s užitím výpočetní geometrie č. ME09051.

Zadání

1. Prostudujte další prameny k problematice modelování písku na terénu a analyzujte, v čem jsou lepší a v čem horší než stávající řešení na domovském pracovišti.
2. Navrhněte vhodná vylepšení a zahrňte je stávajícího řešení.
3. Navrhněte a vyzkoušejte jiné modely pro fyzikální vrstvu.
4. Otestujte a zhodnoťte dosažené výsledky.
5. Analyzujte možnosti propojení daného programového vybavení s haptickým zařízením, proveďte případně potřebné úpravy a vyzkoušejte.

Prohlášení

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

V Plzni dne ..... Václav Purchart .....

Abstract

This master thesis aims to improve our method for the modeling of sandy terrain. Our solution allows deforming terrain by a set of virtual tools and erosion simulation. All these simulations run in real time. Our goal is to try to represent the terrain model as a triangulated irregular network while existing solutions are based on a regular grid. We evaluate the properties of our solution and compare it. As a weak point of the triangulated irregular network solution is its visual appearance, we concentrate also on its improvement. The last goal of this thesis is to use haptic device in combination with our application.

Obsah

1 Úvod....8
2 Existující metody....10
3 Přehled původního řešení....15
3.1 Struktura řešení....16
3.2 Geometrický model....17
3.2.1 Odstraňování vrcholu....18
3.2.2 Lokace trojúhelníku....19
3.3 Modelování eroze....19
3.3.1 Metody eroze....21
3.4 Virtuální nástroje....21
3.4.1 Vnitřní část VR nástroje....22
3.4.2 Vnější část VR nástroje....22
3.4.3 Pořadí vynucování....22
3.4.4 Vytlačování materiálu....23
4 Problémy řešené v rámci této diplomové práce....24
4.1 Zlepšení vzhledu....24
4.1.1 Shadery v OpenGL....24
4.2 Haptická zařízení....25
4.3 Porovnání s existující metodou....26
4.4 Automatické odstraňování vrcholů....26
5 Podrobnosti řešení....28
5.1 Prostředky použité k realizaci....28
5.1.1 Knihovna OpenGL....28
5.1.2 Knihovna SDL....29
5.1.3 Knihovna GLEE....29
5.1.4 Knihovna OpenHaptics....30
5.2 Zlepšení vzhledu....31
5.2.1 Náhodná textura....31
5.2.2 Bump Mapping....32
5.2.3 Subsurface Scattering....33
5.2.4 Prahování....34
5.3 Použití haptických zařízení....35
5.3.1 Zrychlení aplikace....36
5.3.2 Silové složky....36
5.3.3 Techniky pro modelování materiálů....38
5.3.4 Zjištění množství materiálu ve směru pohybu....38
5.3.5 Omezení chvění nástroje....39
5.3.6 Rozhraní pro haptické zařízení....40
5.3.7 Interpolace zpětné silové vazby....40
5.3.8 Shrnutí....40
5.4 Porovnání s existující metodou....41
5.4.1 Měření řešení s pravidelnou sítí....41
5.4.2 Měření paměti....44
5.4.3 Měření času....46
5.4.4 Vizuální porovnání....49
5.5 Opravené chyby a drobné úpravy....49
5.6 Kompletní algoritmus výpočtu....50
6 Závěr....53
7 Použité pojmy a zkraty....55
8 Literatura a zdroje....56

8.1 Literatura.....56
8.2 Elektronické zdroje.....57

1 Úvod

Bylo vydáno již mnoho publikací zabývajících se počítačovým modelováním terénu. V naší práci se zaměříme na modelování terénu tvořeného sypkým materiálem, tedy například pískem, štěrkem apod. Model terénu je na rozdíl od publikovaných prací reprezentován nepravidelnou trojúhelníkovou sítí. To přináší některé výhody, ale i mnohá úskalí. Naše metoda umožňuje simulaci eroze s ohledem na vlhkost materiálu a přímou deformaci terénu uživatelem pomocí virtuálních nástrojů. Virtuální nástroje je možné ovládat pomocí haptických zařízení, což navodí dojem, jako by se uživatel simulovaného materiálu skutečně dotýkal. Díky reprezentaci terénu nepravidelnou trojúhelníkovou sítí je možné libovolně měnit úroveň detailů modelu, a snížit tak spotřebu paměti. Na druhou stranu je nutné použít mnohem komplikovanější algoritmy.

Tato práce navazuje na metodu navrženou Doc. Dr. Ing. Ivanou Kolingerovou, Ing. Bedřichem Benešem, Ph.D., mnou, J. Kadlecem a J. Sedmihradským, kterou budeme dále zdokonalovat. Vycházeli jsme tedy z již implemetované aplikace, která umožňovala do sítě vynutit různé tvary pomocí virtuálních nástrojů, obsahovala jednoduchý částicový systém simulující padání písku a také umožňovala simulaci eroze.

Po zadání diplomové práce byl přijat projekt Triangularizované modely pro haptiku a virtuální realitu č. GA201/09/0097 a projekt Modelování přírodních jevů s užitím výpočetní geometrie č. ME09051. Po dohodě s vedoucí práce jsme zaměřili především na část zadání věnující se haptickým zařízením a propagaci práce a nevěnovali jsme se jiným fyzikálním modelům.

V rámci této práce jsme ve spolupráci s RNDr. Janem Flasarem, Ph.D. propojili naši aplikaci s haptickým zařízením umístěným na Masarykově univerzitě v Brně. To umožnilo navodit hmatový vjem, jako by se uživatel dotýkal skutečného materiálu. Navíc tento výzkum umožní propojení dalších aplikacích vyvíjených v rámci Centra počítačové grafiky a vizualizace na ZČU s haptickými zařízeními.

Protože je navržená metoda v poměrně pokročilém stádiu vývoje, provedli jsme měření a porovnání naší metody s řešením založeným na pravidelné síti, poskytnuté Ing. Bedřichem Benešem, Ph.D. z University of Purdue, USA.

Dále jsme se pokusili zlepšit vizualizaci terénu, protože v našem řešení byla vizualizace modelu horší v porovnání s řešením používajícím pravidelnou síť. Zlepšení vizualizace jsou

inspirována radami Ing. Petra Vaněčka, Ph.D. a Ing. Bedřicha Beneše, Ph.D.

Tato metoda byla prezentována na Studentské vědecké konferenci na FAV v roce 2008. Prezentace a článek v soutěži ACM Student Research Competition 2008 v Praze získal čtvrté místo ve finále ČR + SR. Dále pak byla metoda prezentována na konferenci CESCG 2009 na Slovensku. Všechny zmíněné články lze nalézt v příloze na konci tohoto textu.

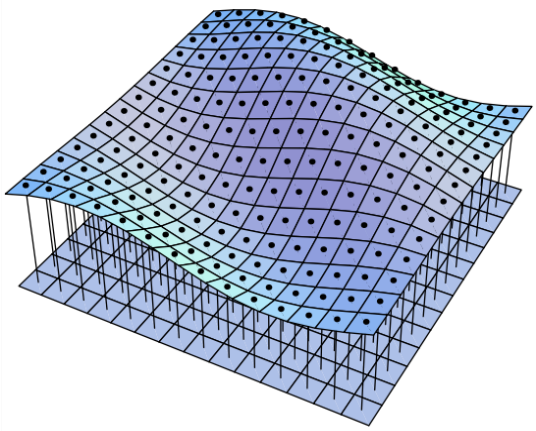
Paralelně se mnou pracoval na rozšíření této práce i kolega Kolařík (viz [Kol09]). Jeho úkolem bylo pokusit se manipulaci s nástroji a terénem rozšířit z 2D do 3D. Dále pak měl vyzkoušet ovládat aplikaci pomocí jiných zařízení za pomoci navrženého rozhraní.

V následujících kapitolách nejdříve shrneme existující práce. Kapitola Existující metody obsahuje popis publikovaných prací zabývajících se modelováním sypkých materiálů. Shrneme zde základní principy publikovaných metod, jejich výhody a nevýhody. V kapitole Přehled původního řešení jsou popsány jednotlivé hlavní části naší metody, které byly implementovány před zahájením diplomové práce. Kapitola Přehled mé části obsahuje obecné poznámky týkající se problémů řešených v rámci této práce. Kapitola Podrobnosti řešení popisuje podrobně řešení problémů nastíněných předcházející kapitole na implementační úrovni včetně měření, které bylo jedním z bodů zadání. Následuje Závěr, kde zhodnotíme dosažené výsledky a nastíníme další možnost rozvoje projektu. Na konci práce jsou přiloženy články publikované na toto téma.

2 Existující metody

V této kapitole si stručně představíme existující práce, které se zabývají simulací povrchů tvořených sypkými materiály. Existující práce lze rozdělit do dvou hlavních směrů. První skupinu tvoří metody založené na fyzikálně přesných modelech. Druhá skupina jsou aplikace pracující interaktivně v reálném čase, což ovšem (zatím) vylučuje fyzikálně přesné modely.

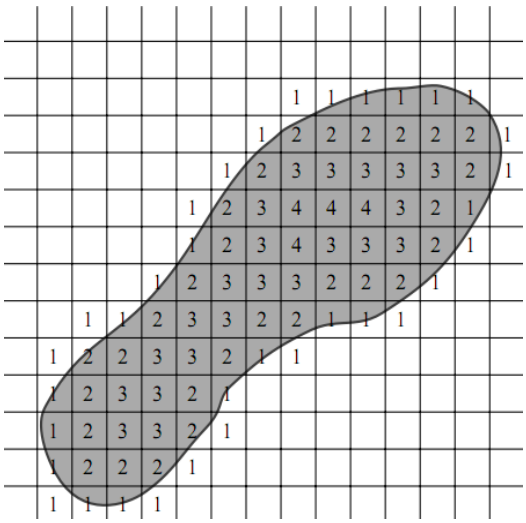
V článku [SBH98] je popsán princip povrchového terénního modelu, který je schopný deformací a umožňuje simulaci různých materiálů, např. písku, bahna nebo sněhu. Povrchový model je takový, kde uvažujeme pouze povrch modelovaného materiálu a nezabýváme se tím, co se nachází pod ním. Toto řešení využívá pravidelnou čtvercovou síť – matici, která obsahuje v každé buňce údaj o výšce terénu v daném místě (viz obrázek 2.1). Takové matici říkáme výšková mapa.



Obrázek 2.1: Ukázka výškové mapy použité k reprezentaci terénu. Převzato ze [SBH98].

Deformace terénu jsou simulovány tak, že v každé iteraci spočítáme průsečík pevného objektu deformujícího terén (virtuálního nástroje) s výškovou mapou. Výška míst ve výškové mapě, představující část terénu kolidující s virtuálním nástrojem, je poté postupně redukována do té doby, kdy už nedochází k průniku nástroje (objektu) s povrchem terénu. Materiál ovlivněný nástrojem je pak buď stlačen dolů nebo vytlačen do stran na základě vlastností modelovaného materiálu a tvaru nástroje. Na každou takovouto deformaci pak také působí eroze, která redukuje převýšení mezi buňkami matice a vyhlazuje terén. Materiál může být při kontaktu s pevným objektem buď stlačen v závislosti na koeficientu stlačitelnosti daného materiálu nebo vytlačen do nejbližší buňky, která nekoliduje s nástrojem (viz obrázek 2.2).

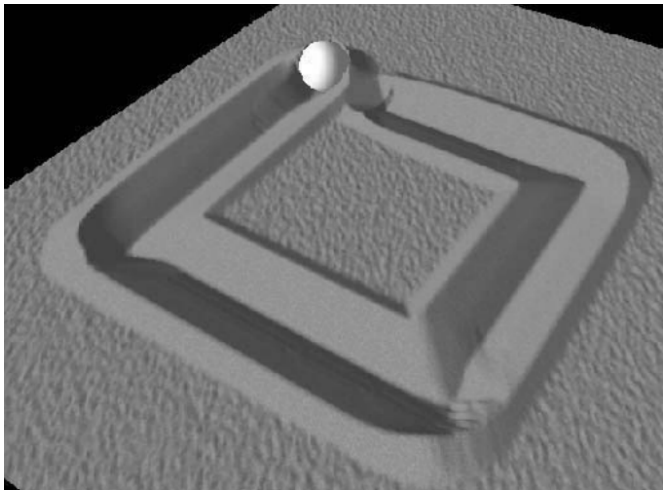
Čísla uvnitř buněk určují, přes kolik sousedních buněk se bude materiál přemisťovat. V tomto článku je modelován písek, bahno a sníh. Vlastnosti modelovaného materiálu lze určit pomocí šesti parametrů (např. již zmíněná stlačitelnost). Tato práce se zabývá co nejpřesnějším modelováním, které ale neprobíhá v reálném čase. Některé náročnější výpočty jsou prováděny paralelně.



Obrázek 2.2: Schéma přesunu materiálu vlivem eroze. Převzato ze [SBH98].

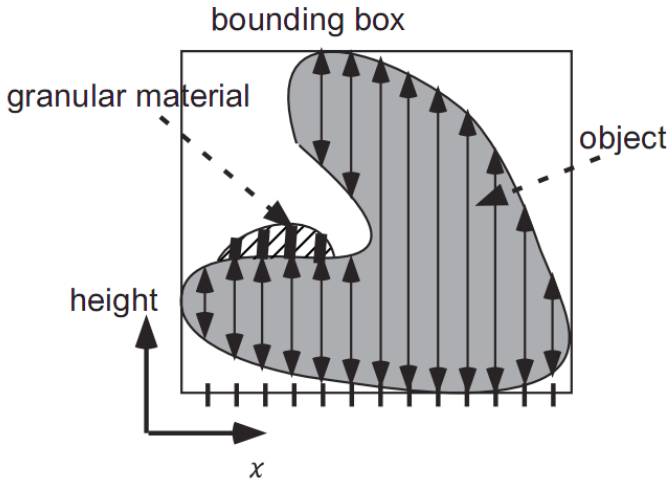
Největší nevýhodou toho přístupu, jak je zmíněno v [SBH98], je spotřeba paměti celé aplikace. Rozlišení matice totiž musí odpovídat těm nejmenším detailům, které chceme simulovat. Naopak výhodou je například snadný přístup k jednotlivým prvkům matice. Pravidelnou síť lze také poměrně snadno otexturovat.

Další práce [Ono03] se opět zabývá modelováním terénu, tvořeného sypkým materiálem. Ukázku programu vidíme na obrázku 2.3. Aplikace umožňuje deformaci terénu pomocí virtuálních nástrojů a simulaci eroze. Při kontaktu virtuálního nástroje s terénem dochází k navýšení vrcholů ležících v okolí nástroje, čímž je simulováno vytlačení materiálu pomocí nástroje do stran. Pro detekci kolize nástroje a terénu jsou využívány bounding boxy a také height spans (HS) mapy. HS mapy jsou matice obsahující výškové rozložení daného objektu. Autoři této práce se zaměřili především na dosažení co nejvíce věrohodné vizualizace písku. Po prozkoumaní původního modelu autoři zjistili, že při přesunu písku mezi vrcholy nepůsobí vizualizace věrohodně. Ve vizualizaci písku využili tzv. „sliding textures“. Ze znalostí toho, kam se má materiál přesunout, pak můžeme zavést tyto pohyblivé textury, které zlepší realističnost celého modelu. Terén je popsán pomocí pravidelné výškové mapy podobně jako v předchozím řešení, což přináší stejné výhody i nevýhody. Tato metoda velmi věrohodně vizualizuje písek.



Obrázek 2.3: Vizualizace písčitého terénu [Ono03].

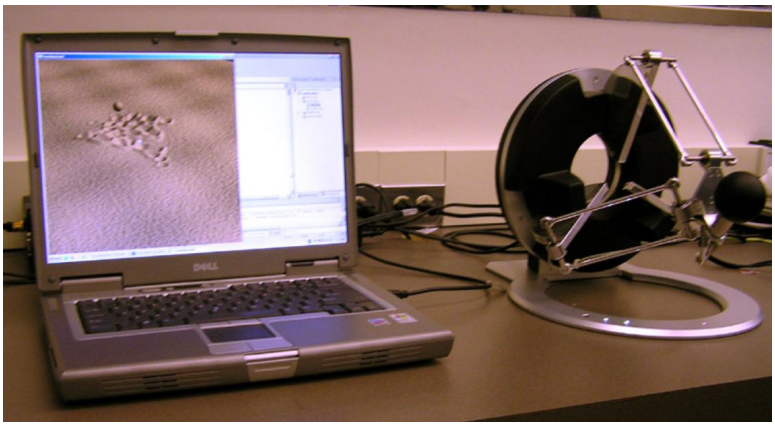
Oproti předchozí práci je možné reprezentovat sypký materiál i na objektech, což samotná výšková mapa neumožňuje. Mapa může obsahovat v každém místě pouze jednu hodnotu. Pro umožnění reprezentace materiálu na objektech obsahuje řešení částicový systém, kde každá částice reprezentuje určitou část objemu písku nad rámec výškové mapy (viz obrázek 2.4). Po dopadu částice na terén pak dojde v příslušném místě k navýšení výškové mapy a zániku částice. Tato práce patří mezi aplikace pracující v reálném čase.



Obrázek 2.4: Reprezentace materiálu na objektech pomocí částicového systému. Převzato z [Ono03].

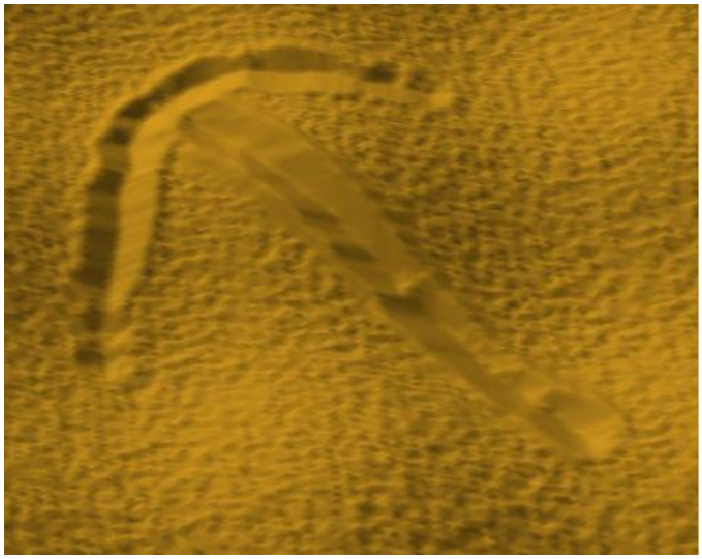
V článku [Ben06] je popsán způsob eroze materiálu s využitím výškové mapy a deformace terénu pomocí virtuálních nástrojů za použití haptických zařízení, což umožňuje uživateli skutečně cítit odpor písku. Zpětná silová vazba haptického zařízení (viz obrázek 2.5) se skládá ze síly způsobené třením nástroje o povrch terénu a ze síly způsobené pronikáním nástroje do písku. Je zde mimo jiné zmíněno, že po delší práci s haptickým zařízením dochází

ke značné únavě uživatele. Jako virtuální nástroj je zde použita koule, díky její invariantnosti vůči transformacím. K vizualizaci jsou použity pohyblivé textury podobně jako v předchozí práci, jejichž účelem je docílit věrohodnější vizualizace.



Obrázek 2.5: Propojení aplikace s haptickým zařízením. Převzato z [Ben06].

Pohyb písku vlivem eroze závisí především na jeho vlhkosti a tzv. „sutinovém úhlu“ (talus angle). Autoři tento úhel experimentálně určili na 30°. Pokud písčitá rovina dosáhne tohoto sklonu, nedochází už k přesunu materiálu z výše položených míst na místa níže položená a eroze v daném místě končí až do dalšího zásahu do terénu.

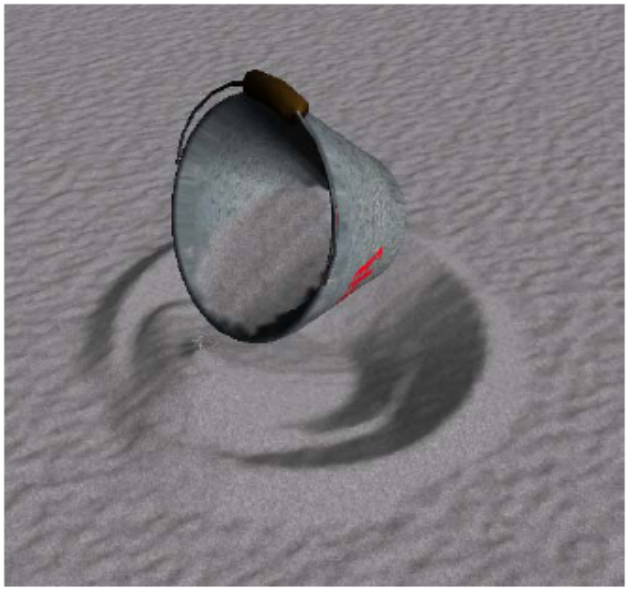


Obrázek 2.6: Ukázka vyhlazení vrypu po nástroji vlivem eroze. Převzato z [Ben06].

Metoda spadá do skupiny interaktivních aplikací. Reprezentace terénu výškovou mapou s sebou přináší stejné výhody i nevýhody jako v předchozích případech.

Shrnutí

Uvedené metody přinášejí několik zajímavých myšlenek. Především je to samotná reprezentace písčitého terénu pomocí výškové mapy. Dále jsou zde uvedeny různé algoritmy pro výpočet eroze. Tyto metody přesouvají materiál z výše položených míst na místa níže položená. Zajímavou zmíněnou vizualizační technikou jsou pohyblivé textury, pomocí kterých lze dosáhnout věrohodnější simulace eroze. Byly zde také uvedeny různé postupy výpočtu kolize virtuálních nástrojů s terénem. Tyto nástroje mohou mít i velmi komplikovaný tvar, jak je vidět na obrázku 2.7. Všechna výše zmíněná řešení používají k reprezentaci terénu výškovou mapu uloženou v pravidelné čtvercové síti. Hlavní výhodou toho přístupu je jednoduchost použití pravidelné výškové mapy. Snadno lze zjistit místo v libovolném místě terénu. To *velmi* zjednodušuje práci s terénním modelem, jak už jeho vizualizaci nebo simulaci fyzikálních procesů. Pravidelné sítě lze také poměrně snadno otexturovat. Ovšem velkou nevýhodou, jak je zmíněno v [SBH98], je rozsáhlost této datové struktury. Čím větší detaily potřebujeme vyjádřit, tím větší musí být rozlišení výškové mapy a to po *celé* její ploše. Velikost výškové mapy je kvadraticky závislá na nejmenší úrovní detailů, kterou chceme v modelu vyjádřit. Pokud například chceme v nějakém místě terénu o rozměrech 10m^2 dosáhnout úroveň detailu 0.1cm, musí být rozlišení výškové mapy 10^8 bodů. A to i v případě, že zbytek terénu je naprosto plochý. Nelze tedy simulovat rozsáhlejší plochy nebo velké úrovně detailů. Tento problém se pak ještě umocňuje ve vícevrstvých terénních modelech.



Obrázek 2.7: Ukázka vyhlazení vrypu po nástroji vlivem eroze. Převzato z [Ono03].

3 Přehled původního řešení

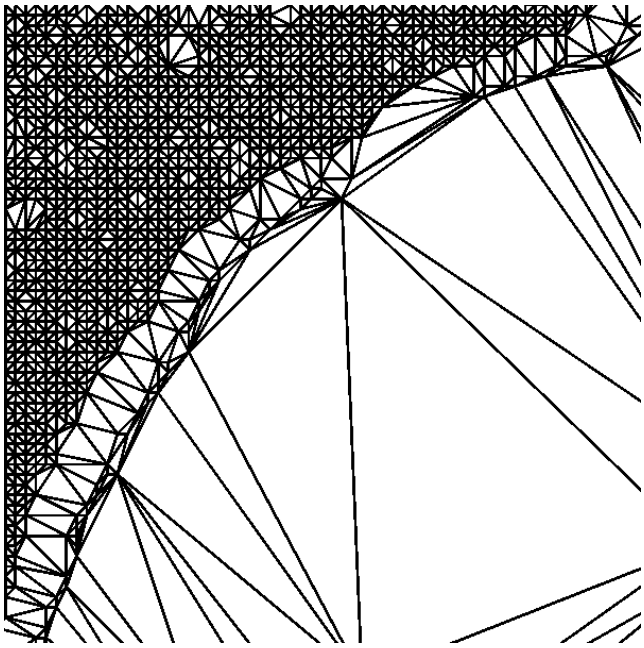
Naše metoda umožňuje simulovat jakési virtuální pískoviště. Písčitý terén lze deformovat pomocí různých virtuálních nástrojů. Na tyto a jiné změny pak terén reaguje jako skutečný písek tak, že se sesypává z výše položených míst na níže položená místa. Vlivem této eroze se tak postupně zasypou vzniklé díry a důlky, až celý terén působí hladce. Je nutné brát v úvahu řadu parametrů modelovaného materiálu, jako je například jeho vlhkost. Virtuální nástroje umožňují jak jednoduché otisknutí svého tvaru do terénu, tak tažení nástrojem napříč terénem. Toto tažení je simulováno obtiskováním nástroje v pevně daných časových intervalech.

Metodu jsme se snažili navrhnout tak, aby odstranila největší nedostatky modelů založených na pravidelné síti, tj. pevnou úroveň detailů a vysokou paměťovou náročnost. Pro reprezentaci terénu bylo nutné zvolit síť, která nemá pevně danou velikost a vzdálenost svých buněk. Po důkladném prozkoumání dané problematiky jsme se rozhodli použít pro reprezentaci terénu nepravidelnou trojúhelníkovou síť (TIN, triangulated irregular network). Taková síť se skládá z vrcholů, hran a trojúhelníků. Trojúhelníky obsahují informaci o svých sousedních trojúhelnících.

Vrcholy trojúhelníkové sítě popisují výšku terénu v daném místě. Změny písčitého terénu jsou simulovány buď změnou výšky stávajících vrcholů, nebo přidáním nových vrcholů do sítě. Požadavkem na trojúhelníkovou síť je, abychom do ní byly schopni vynutit určité tvary dané tvarem používaného virtuálního nástroje. To zajistíme vynucením skupiny takových hran, které mají požadovaný tvar. Eroze pracuje na principu částicového systému. Vrcholy a hrany trojúhelníkové sítě tvoří graf. Hmota modelovaného materiálu se pak po hranách tohoto grafu „přelévá“ z vrcholu do vrcholu tak, až postupně zaniknou všechny ostré přechody v terénu. Výše ležícím vrcholům je výška vlivem eroze snížena, naopak níže ležícím je zvýšena. Eroze je závislá na vlastnostech modelovaného materiálu (např. vlhkost). Kvůli tomuto částicovému systému jsme zavedli ještě sousednost vrcholů. To znamená, že každý vrchol nese informaci o tom, se kterými jinými vrcholy sousedí.

Použitím TIN odstraníme jakékoli problémy vyplývající z pravidelného tvaru buněk nebo škálovatelnosti úrovně detailů. V místech, kde je terén hodně členitý, obsahuje nepravidelná trojúhelníková síť mnoho trojúhelníků. Naopak v místech, kde je terén rovný, nám stačí k jeho popisu několik málo trojúhelníků. V extrémním případě stačí pro popis takové oblasti dva trojúhelníky a čtyři vrcholy. Ukázka nepravidelné trojúhelníkové sítě je na

obrázku 3.1. Vidíme, že méně členité plochy (např. vodní hladina) jsou tvořeny menším počtem trojúhelníků než okolní terén. Některé části následujícího textu jsou převzaty z [Pur08].



Obrázek 3.1: Ukázka nepravidelné trojúhelníkové sítě (část jezera Crater Lake použitá v našem programu).

3.1 Struktura řešení

Vstupem navržené je množina vrcholů, které popisují výšku terénu v daném místě. Body jsou nerovnoměrně a nepravidelně rozmístěny podle potřebné úrovně detailů požadované v daných místech.

Definice:

Terén je definován jako množina bodů $P:\left\{x_i \in \Re^3,\ x_i=(x,y,z),\ i=1,\ldots,n\right\}$. Souřadnice x a y určují pozici vrcholu v terénu a platí $x,y \in [0;1]$. Souřadnice z určují výšku terénu v daném místě. Výšku terénu v ostatních místech lze spočítat interpolací vrcholů. Nad množinou vrcholů P vytvoříme nepravidelnou trojúhelníkovou síť, triangulaci. Každý trojúhelník obsahuje odkazy na sousední trojúhelníky. Navíc obsahuje každý vrchol x_i odkazy na sousední vrcholy, tj. takové vrcholy, do kterých vede z x_i hrana.

Z toho vyplývá, že tvar terénu musí být vyjádřitelný jako funkce dvou proměnných $f(x,y)=z$. Model tedy neumožní modelování převisů a tunelů. Což ovšem není nijak omezující, protože písčitý terén lze v ustáleném stavu popsat právě takto.

3.2 Geometrický model

Nejdříve přesně definujeme, jak je model reprezentován. Dále pak popíšeme základní operace, jako je přidání/odebrání vrcholu, vynucení hrany do sítě, zjištění a změna výšky v libovolném místě modelu a nalezení trojúhelníku na určené pozici.

Triangulací v prostoru $\Re^2$ neboli rovinnou triangulací na množině bodu *P*, rozumíme maximální možné rozdělení konvexní obálky množiny *P* na trojúhelníky (rovinné simplexy), které splňuje následující podmínky (viz [Kad07], [BKO97]):

- každé dva trojúhelníky se buď neprotínají vůbec, nebo mají společný jeden bod, anebo jednu hranu,
- množina bodů *V* tvořená vrcholy všech trojúhelníků triangulace je shodná s množinou *P*.

Kvůli vizualizaci a simulaci písku je potřeba, aby hrany každého trojúhelníku v síti byly pokud možno podobně dlouhé. Tedy aby se trojúhelníky co nejvíce blížily trojúhelníkům rovnostranným. Přesouvání materiálu pak působí přirozeněji. Delaunayova triangulace omezuje vznik nám nevyhovujících trojúhelníků.

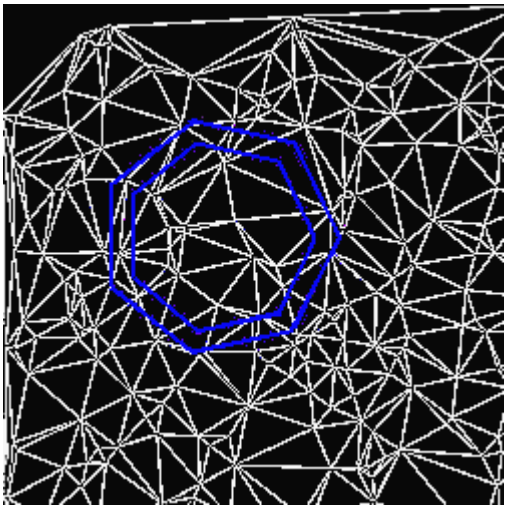
Základní Delaunayova triangulace v $\Re^2$ na množině bodů *P* je taková triangulace *DT(P)*, v níž každý trojúhelník, definovaný vrcholy $A=[x_1, y_1]$, $B=[x_2, y_2]$ a $C=[x_3, y_3]$, kde platí $A, B, C \in P$, splňuje tzv. *Delaunayovo kritérium prázdné kružnice* pro každý bod $D=[x_4, y_4]$, kde $D \in P \wedge D \notin \{A, B, C\}$ (vztah 3.2.1). Bod *D* neleží v kružnici opsané trojúhelníku *ABC* právě tehdy, když $d < 0$ (kritérium je splněno).

$$d = \begin{vmatrix} x_1 - x_4 & y_1 - y_4 & (x_1 - x_4)^2 + (y_1 - y_4)^2 \\ x_2 - x_4 & y_2 - y_4 & (x_2 - x_4)^2 + (y_2 - y_4)^2 \\ x_3 - x_4 & y_3 - y_4 & (x_3 - x_4)^2 + (y_3 - y_4)^2 \end{vmatrix} \qquad (3.2.1)$$

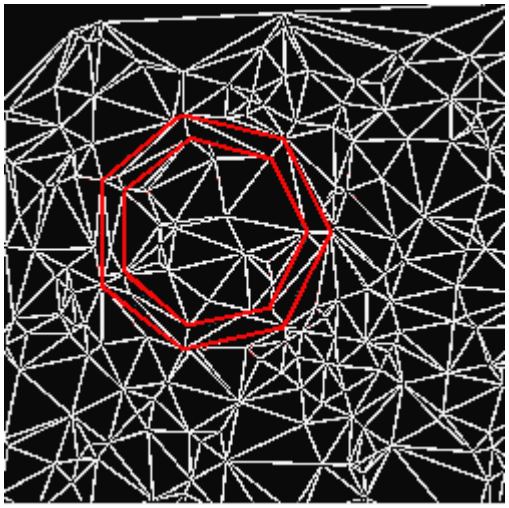
Delaunayova triangulace je konstruována algoritmem inkrementálního vkládání bodů (podrobnější popis viz [Kad07]). Při vložení nového vrcholu do sítě najdeme trojúhelník, který nový vrchol obsahuje, a rozdělíme ho na několik nových. Zbytek sítě upravíme prohazováním hran tak, aby vyhověl triangulačnímu kritériu.

Pro další práci je nutné, aby síť umožňovala přímo vynutit některé hrany. Pro vynucení hran používáme Delaunayovu triangulaci s omezením (dále jen CDT). Tato triangulační metoda vychází ze základní Delaunayovy triangulace, oproti té se v CDT navíc zavádí tzv.

„vynucené hrany“. Tyto hrany jsou v síti vyžadovány i v případě, že nesplňují Delaunayovo kritérium prázdné kružnice (vztah 3.2.1). Umožňují tak přímo v terénu vynutit požadované rysy. Ukázku vynucení hran vidíme na obrázku 3.2.1 (kam chceme hrany vynutit) a obrázku 3.2.2 (výsledná síť s vynucenými hranami).



Obrázek 3.2.1: Původní trojúhelníková síť. Modře jsou hrany, které chceme vynutit.



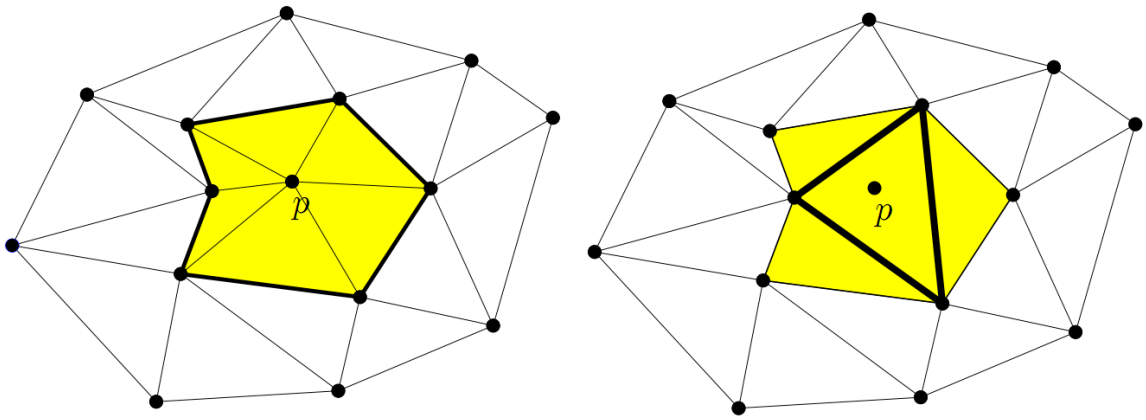
Obrázek 3.2.2: Síť s vynucenými hranami – vynucené hrany jsou zobrazeny červeně.

Algoritmy pro vynucení množiny hran jsou vedeny například v [Ang97], [Che87] nebo [Slo93]. Námi použitý algoritmus pro vynucení množiny hran vychází ze [Slo93]. Nejdříve zkonstruujeme základní DT a poté v ní vynutíme požadovanou množinu hran. Pro jednu hranu postupujeme následujícím způsobem: Postupně prohazujeme existující hrany, které se protínají s vynucovanou hranou, až lze do sítě vložit požadovanou hranu. Tato vynucená hrana ignoruje požadavky na prohození (viz výše). Výsledná triangulace tak už nemusí splňovat kritérium 3.2.1, ovšem síť obsahuje požadovaný tvar.

3.2.1 Odstraňování vrcholu

Odebírání existujících vrcholů ze sítě je nezbytné, protože pouhým přidáváním se model stává stále komplikovanějším a to zpomaluje probíhající výpočty. Vrcholy jsou přidávány všude tam, kde je nutné dosáhnout vyšší úrovně detailů, to jsou například místa deformovaná virtuálním nástrojem. Odstraňování vrcholů využívá algoritmus [Dev98]. Ten nejdříve odstraní požadovaný vrchol ze sítě a poté retriangularizuje vzniklou díru tak, aby vznikla opět Delaunayova triangulace. Po odstranění vrcholu vznikne v nejhorším případě hvězdicový polygon. Ten postupným ořezáváním eliminujeme. Ukázku odstraňování vrcholu ze sítě vidíme na obrázku 3.2.3. V oblasti ohraničené hranami polygonu přibudou nové hrany.

Naproti tomu za touto hranicí se už síť měnit nebude, protože odebráním vrcholu neporušíme Delaunayovo kritérium. Všechny změny se tedy budou odehrávat uvnitř vzniklého polygonu. V článku [Dev98] je uveden postup v jakém pořadí vytvořit nové hrany tak, aby výsledná triangulace byla opět Delaunyovská. Využívá se zde prioritní fronta zabezpečující správné pořadí ořezávání. Podrobně je celý implementovaný algoritmus popsán v [Pur07b].



Obrázek 3.2.3: Vlevo síť před odstraněním vrcholu p. Vpravo po odstranění p. Převzato z [Dev98].

3.2.2 Lokace trojúhelníku

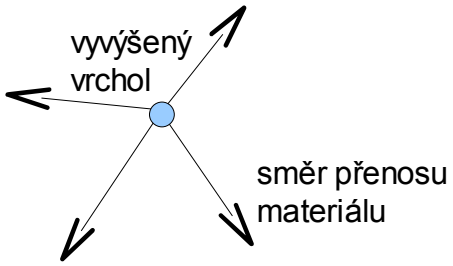
Nejpoužívanější geometrickou operací je nalezení trojúhelníku, v němž leží zadaný (cílový) bod. Je proto nutné, aby byla doba lokace trojúhelníku co nejkratší. Síť reprezentující terén se ovšem neustále, někdy i poměrně drasticky, mění. Použití složitých vyhledávacích struktur tedy není příliš efektivní. Na doporučení kolegy Soukala jsme pro lokaci trojúhelníku zvolili algoritmus pravoúhlé procházky [Sou08]. Tento algoritmus je suboptimální s očekávanou složitostí přibližně od $O(n^{1/3})$ do $O(n^{1/2})$. V porovnání se složitostí $O(\log(n))$ jiných vyhledávacích algoritmů, jako je například DAG (directed acyclic graphs), skip-list, quadtree apod. Nepotřebujeme ovšem žádné speciální datové struktury nebo informace z předchozích hledání, pouze informaci o sousedních trojúhelnících každého trojúhelníku. Při rozsáhlosti a dynamičnosti našeho modelu terénu je tato vlastnost klíčová. Proto jsme zvolili „procházkové algoritmy“ pro lokaci trojúhelníku.

3.3 Modelování eroze

V přírodě existuje mnoho druhů eroze – například vodní eroze, eroze způsobená roztahováním a smršťováním materiálu při teplotních změnách a jiné. Výsledkem společného působení těchto procesů je, že se materiál přesouvá z vyšších míst na místa níže položená.

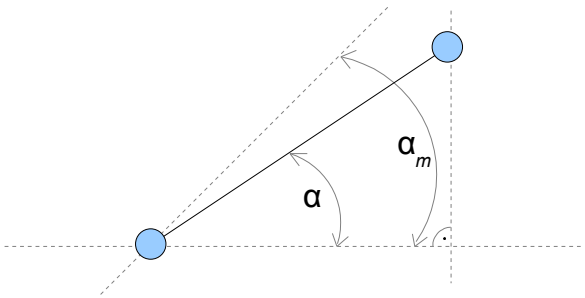
Přesun je tím razantnější, čím je rozdíl výšek větší. Tímto typem eroze se budeme zabývat. Simulace neprobíhá na základě přesného fyzikálního modelu z důvodu interaktivity celého procesu.

Vrcholy P trojúhelníkové sítě *CDT(P)* a množina hran E tvoří graf. Simulace eroze probíhá iteračně. Pro každý vrchol $P_k \in P$ zjistíme všechny vrcholy P_j, do kterých vede z P_k hrana $E_i \in E$ (viz obrázek 3.3.1). Pro každou takto vzniklou dvojici spočítáme úhel, který svírá vodorovná přímka procházející P_k s hranou mezi P_k a P_j. Pokud je tento úhel větší než tzv. „kritický úhel“ materiálu, dojde k přesunu materiálu z bodu P_k do každého sousedního bodu P_j ležícího níže než P_k. Pro suchý písek je tento úhel přibližně 30°, viz [Sed07]. Ukázku kritického úhlu vidíme na obrázku 3.3.2.



Obrázek 3.3.1: Směr šíření materiálu z vyvýšeného vrcholu.

Procházení a erodování všech vrcholů sítě by bylo výpočetně neúnosné. Eroze se zúčastní jen ty vrcholy, u kterých může dojít k přesunu materiálu. Vrcholy uložíme do zvláštního seznamu L_e. Vrchol je nutné přidat do L_e tehdy, pokud se výška vrcholu v předchozí iteraci změnila. Změna výšky je vyvolána deformací terénu virtuálním nástrojem a také samotným procesem eroze. Vrchol ze seznamu odstraníme v případě, že u něj opakovaně nedošlo ke změně výšky působením eroze. Podrobněji viz níže. Popis celého algoritmu je v kapitole 5.6 .



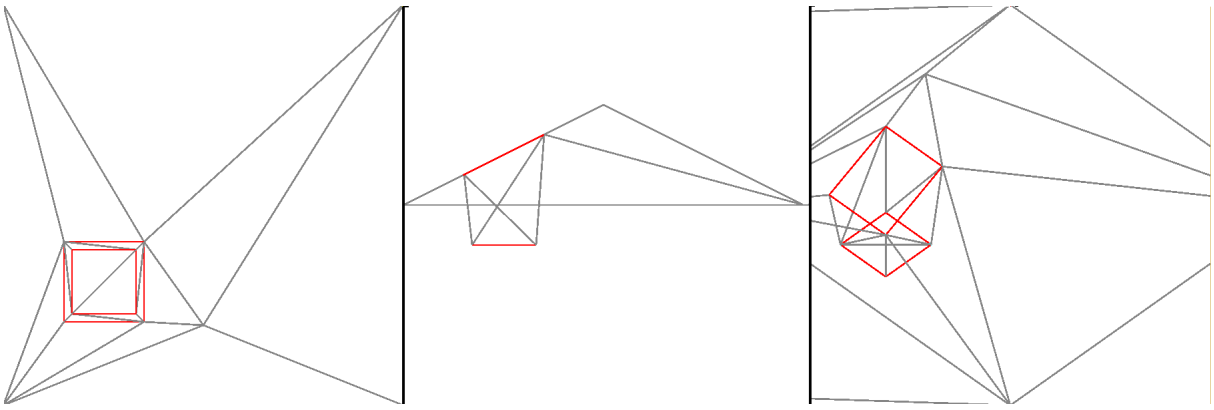
Obrázek 3.3.2: Porovnání úhlu (α) mezi dvěma vrcholy s mezním úhlem pro daný materiál (α_m).

3.3.1 Metody eroze

V druhé fázi algoritmu dochází ke změně výšky vrcholů. Jednomu vrcholu je výška snížena a některým jeho sousedům naopak zvýšena. Podstatným údajem z hlediska věrohodnosti průběhu eroze je právě poměr těchto hodnot. Z fyzikálního hlediska je podstatnou vlastností eroze zachovávání objemu, a naším cílem by tedy mělo být určení daného poměru výšek tak, aby „objem sítě“ před a po erozi zůstal stejný. Trojúhelníková síť samozřejmě nemá objem, ale lze si ji představit jako jednu ze „stěn“ tělesa, které vznikne, pokud z okrajů sítě spustíme svislé stěny a ty uzavřeme v určité vzdálenosti rovinou. Množství materiálu distribuovaného ze středového vrcholu pak závisí právě na poměru těchto objemů. Podrobněji viz [Sed07], kde jsou uvedeny i různé modifikace zmíněné metody. Například lze objem hranolů prohlásit za přibližně konstantní, čímž ušetříme velké množství výpočetního času. Dále je možné materiál distribuovat na základě rozdílů výšek a vzdáleností obou vrcholů.

3.4 Virtuální nástroje

Aplikace obsahuje sadu virtuálních nástrojů, pomocí kterých lze do terénu přímo vynucovat změny. Tyto virtuální nástroje umožňují jak jednoduché otisknutí nástroje do terénu, tak tažení nástrojem v materiálu. Vynucování změn v síti je založeno na *CDT*. Otisknutí nástroje simulujeme řízeným vynucováním skupiny hran. Tažení nástrojem simulujeme vzorkováním otisků nástroje v pravidelných časových intervalech. Každý virtuální nástroj (resp. jeho otisk) se dělí na dvě důležité části – vnější a vnitřní část. Tyto části jsou od sebe nepatrně vzdáleny, aby se nepřekrývaly. Aby platilo, že lze povrch terénu vyjádřit jako funkci dvou proměnných. Ukázka hloubícího razítka je vidět na obrázku 3.4.1. Podrobněji viz [Pur07].



Obrázek 3.4.1: Ukázka otisku nástroje. Vlevo – půdorys, uprostřed – bokorys, vpravo – pohled ze strany.

3.4.1 Vnitřní část VR nástroje

Vnitřní část nástroje je definována jako skupina hran *I*, tvořících požadovaný tvar. Účelem vnitřní části je deformovat povrch terénu požadovaným způsobem, tj., změnit výšku povrchu na úroveň hloubky zarytí nástroje. Tím se oblast uzavřená množinou hran *I* stává ve většině případů rovinou. Body, které jsou v této oblasti, je proto možné ze sítě odstranit. Odstraňování je také možné přenechat automatickému zjednodušování sítě, které je ale výpočetně náročnější. Procházíme všechny trojúhelníky vnitřní oblasti a kontrolujeme jejich vrcholy. Vrchol P_i uzavřený ve vnitřní oblasti nástroje množinou hran *I* je redundantní, pokud zároveň platí:

- hloubka bodu je přibližně stejná jako hloubka razítka,
- odstraňovaný bod není bodem tvořícím hraniční oblast *I*,
- bod nesousedí s bodem, jehož hloubka je větší než hloubka vrypu nástroje.

V síti mohou být i hrany z předchozích vrypů. Každá množina vynucovaných hran má jednoznačný identifikátor, podle kterého lze každou jí náležící hranu identifikovat. Pokud při procházení vnitřní oblasti narazíme na vynucenou hranu, která není součásti zpracovávané množiny, ignorujeme ji, protože se jedná o starý vryp.

3.4.2 Vnější část VR nástroje

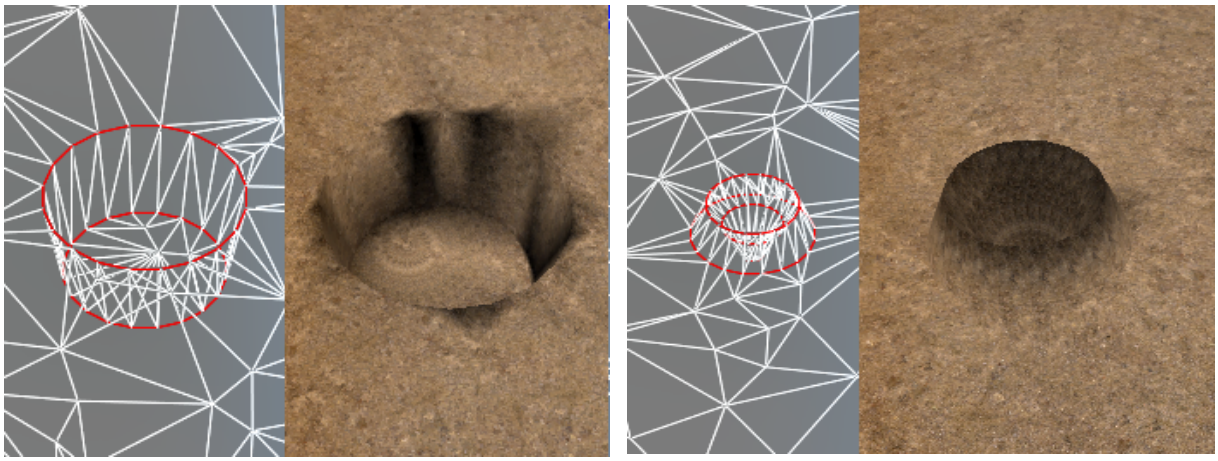
Vnější část nástroje je definována jako skupina hran *O*. Tato část se přizpůsobuje tvaru okolního terénu tak, aby okraj vrypu působil co nejpřirozeněji. Lineární interpolací zjistíme výšku terénu ve vrcholech tvořících vnější oblast *O*. Poté do sítě vynutíme všechny hrany z *O*. Tím dosáhneme ve většině případů relativně věrohodného výsledku. Pro kvalitnější otisknutí je nutné přesně přizpůsobit vnější okraj okolnímu tvaru terénu. Přibližného řešení dosáhneme rozdělením hran z *O* na menší části, které se opět přizpůsobí výšce terénu.

3.4.3 Pořadí vynucování

Je důležité, v jakém pořadí do terénu vynutíme jednotlivé části nástroje. Nejdříve je nutné otisknout vnější část, která se přizpůsobí tvaru terénu, a teprve poté vnitřní část, která „násilně“ deformuje terén podle vlastností nástroje. V opačném případě by se totiž některé hrany mohly při výpočtu triangulace po vložení vnitřní části zrušit. Vnější okraj razítka by se pak sice přizpůsobil tvaru terénu, ale ten už by neodpovídal situaci před otiskem razítka.

3.4.4 Vytlačování materiálu

Ze základních nástrojů lze odvozovat libovolné jiné. Pro ještě realističtější simulaci hrabání je nutné vytlačovat materiál o objemu nástrojem vyhloubené díry do okolí vrypu. Jinak by se materiál z terénu postupně ztrácel. Podrobnosti o přibližném řešení nalezneme v [Sed07]. Ukázku různých nástrojů vidíme na obrázku 3.4.2. Tento virtuální nástroj rozšiřuje nejjednodušší typ nástroje zmíněného v předchozích kapitolách o další dvě množiny hran. Tyto hrany vytvoří jakýsi násyp, jak vidíme na obrázku 3.4.2. Vnější okraj násypu se opět musí interpolací co nejvíce přiblížit původnímu tvaru terénu. Výška vnitřního okraj násypu se pak řídí podle množství vytlačeného materiálu v daném směru. Řešení implementované v [Sed07] je velmi hrubou aproximací používající konstantní výšku a tvar násypu nezohledňující vlastnosti materiálu ani směr tažení. Ovšem i přesto je vizuální výsledek mnohem lepší než v případě jednoduchého nástroje.



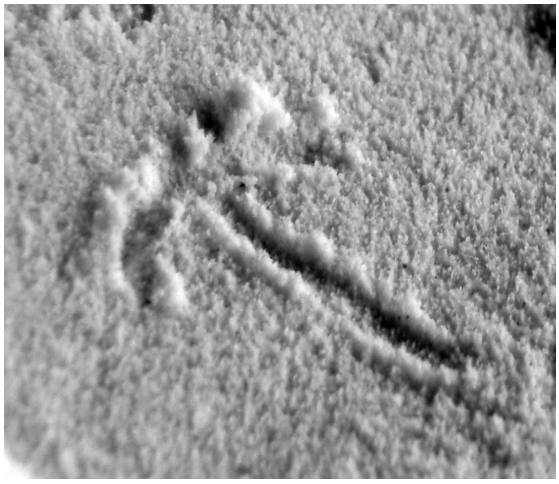
Obrázek 3.4.2: Ukázka vrypu jedno z nástrojů bez vytlačování materiálu (vlevo) a vrypu nástroje s vytlačováním materiálu (vpravo). Model bez eroze. Převzato z [Sed07]

4 Problémy řešené v rámci této diplomové práce

V této kapitole stručně popíšeme problémy řešené v rámci této diplomové práce. Konkrétní implementační detaily jsou uvedeny v kapitole 5 . Na závěr kapitoly přehledně shrneme celý algoritmus pro geomerii a výpočet eroze.

4.1 Zlepšení vzhledu

Stávající vizualizaci implementoval v rámci své diplomové práce Jiří Sedmihradský (viz [Sed07]). Z obrázků 4.1.1 a 4.1.2 vidíme, že vzhled příliš neodpovídá chování skutečného písku. Je proto nutné nějak stávající vizualizaci vylepšit. Na tomto úkolu jsme pracoval s pomocí Ing. Petra Vaněčka, PhD. (ZČU Plzeň) a Ing. Bedřicha Beneše, PhD. (Univerisity of Purudue, Indiana, USA). Pro implementaci většiny moderních grafických efektů je nutné použit shadery grafické karty. Tímto směrem zaměříme i naší snahu, protože výpočet na GPU nebude výrazně omezovat výkon aplikace i při složitějších vizualizačních efektech.



Obrázek 4.1.1: Skutečný písek, velmi jemný, mírně vlhký. Jako nástroj bylo použito kulaté dřívko.

Obrázek 4.1.2: Simulovaný písek, suchý, působení eroze. Byl použit nástroj s kulatým koncem.

4.1.1 Shadery v OpenGL

V OpenGL se postupem času ustálily dva vysokoúrovňové jazyky pro programování GPU. Prvním je GLSL, které je od verze OpenGL 2.0 součástí standardu. Dalším jazykem je Cg od firmy NVIDIA. Tento jazyk lze využít pomocí tzv. Extensions (rozšíření). Jazyk Cg je velice podobný konkurenčnímu řešení od firmy Microsoft – jazyku HLSL. HLSL je ovšem pouze pro rozhraní DirectX. Pro podrobnější informace o vysokoúrovňových jazycích pro programování GPU viz [Glsl], [Cg], [Hlsl]. Z důvodu přímé podpory jsme se rozhodli pro programování shaderů využít jazyk GLSL. Velkou výhodou použití vysokoúrovňového jazyka je možnost kompilace GPU programu za běhu aplikace. Tím se značně zjednoduší

ladění aplikace, protože nejsme nuceni při každé změně shaderu program znovu překládat. Tím se alespoň částečně kompenzuje problematické ladění GLSL a OpenGL aplikací obecně.

4.2 Haptická zařízení

Haptické zařízení je zařízení působící na uživatele silovou zpětnou vazbou. Společnost SensAble [Sens09] dodává haptická zařízení Phantom. Tato zařízení umožňují uživatelům manipulovat s virtuálními objekty a přímo je díky hmatu cítit. SensAble dodává několik typů těchto zařízení lišících se velikostí pracovní zóny, velikostí zpětné vazby apod. Mezi nejlepší modely patří Phantom Premium, dále pak Phantom Desktop, až k nejlevnějšímu Phantom Omni. Pro využití těchto zařízení v aplikacích lze využít buď Ghost SDK [Ghost] nebo toolkit OpenHaptics [OpenH].

Práce je součástí již zmíněných projektů, na kterých spolupracujeme s Masarykovou univerzitou v Brně. V době tvorby této práce ještě Západočeská univerzita nevlastnila žádné z těchto zařízení, a tak jsme se rozhodli využít zařízení umístěné na MU v Brně. Bylo nutné dohodnout se s kolegy z Brna na způsobu spolupráce a požadavcích, které musí naše aplikace splňovat.

Po první schůzce v únoru s kolegou RNDr. Janem Flasarem, Ph.D. z MU Brno na konferenci WSCG 2009 jsme stanovili první požadavky, které naše aplikace musí splňovat, aby v ní bylo možné využít haptická zařízení typu Phantom.

Požadavky na aplikaci:

- Obnovovací frekvence hlavní smyčky programu, která komunikuje se zařízením, musí být minimálně 1000Hz, jinak není odpor zařízení přirozený a působí příliš vláčně.
- Je nutné získat „množství materiálu ve směru pohybu“. (Opět minimálně 1000x za sekundu.)

Obnovovací frekvence naší aplikace je cca 100Hz a VR nástroj pracuje pouze lokálně bez znalosti okolního terénu. Tyto zásadní problémy je tedy nutné nějak obejít. Podrobnosti řešení tohoto bodu zadání si dále popíšeme v kapitole 5.3 .

V březnu jsme podnikli dvoudenní cestu do Brna na Masarykovu univerzitu, kde jsme spolu s kolegou Flasarem navrženou metodu pro interakci s uživatelem otestovali a doladili na reálném zařízení. Na Masarykově univerzitě v Brně používají Virtual Reality Framework – VRECKO (viz [Vrec]). VRECKO je navrženo pro rychlé prototypování technik interakce člověka s počítačem. Virtuální prostředí a jeho komponenty mohou být konfigurovány za

běhu. V budoucnu by bylo vhodné naší aplikaci propojit s tímto systémem.

V průběhu dubna bylo do laboratoří Západočeské univerzity zakoupeno haptické zařízení typu Phantom. Dříve vytvořené rozhraní a ovládání implementované v Brně poté ještě testoval a zdokonaloval kolega Kolařík (viz [Kol09]).

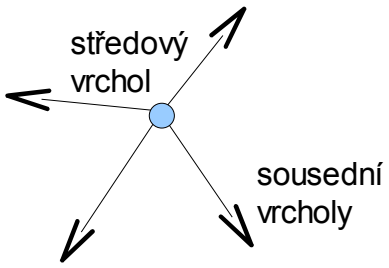
4.3 Porovnání s existující metodou

Na tomto projektu spolupracujeme s kolegou Ing. Bedřichem Benešem, Ph.D, který celou práci kdysi inspiroval. Nyní nám půjčil implementaci svého řešení publikovaného v [Ben06] pro porovnání naší metody založené na nepravidelné trojúhelníkové síti a jeho řešení, které vytvořili společně s E. Dojgotovem, využívající pravidelnou síť.

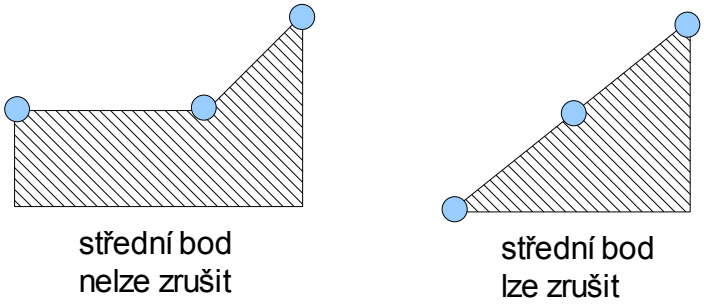
Získaný program také využívá haptické zařízení typu Phantom. Ovšem pro obsluhu zařízení je využita knihovna CHAI 3D (viz [Chai]). Jedná se o open-source knihovnu, jejíž primárním účelem je obsluha haptických zařízení. Knihovna umožňuje obsluhovat různá komerční zařízení se třemi a šesti stupni volnosti. Kromě výše zmíněného obsahuje knihovna také část určenou vizualizaci scény (mono a stereo) a systém pro detekci kolizí mezi nástroji a modelem. Podrobnosti řešení lze nalézt v kapitole 5.4 .

4.4 Automatické odstraňování vrcholů

Mechanismus pro automatickou detekci redundatních vrcholů navržený v [Pur07b] jsme na začátku tohoto akademického roku dále zdokonalovali. Nepotřebné vrcholy lze totiž poměrně snadno detekovat. Pokud vrchol leží vzhledem ke svým sousedním vrcholům (přibližně) v rovině, jeho odstraněním se síť příliš nezmění (viz obrázky 4.4.1 a 4.4.2). Pokud je maximální úhel, o který se vychyluje spojnice zkoumaného vrcholu a každého jeho přímého sousedního vrcholu (tj. takových vrcholů, do kterých vede ze středového vrcholu hrana), vhledem k vodorovné přímce vycházející ze zkoumaného vrcholu blízký nule, je vrchol nadbytečný a lze ho ze sítě odstranit.

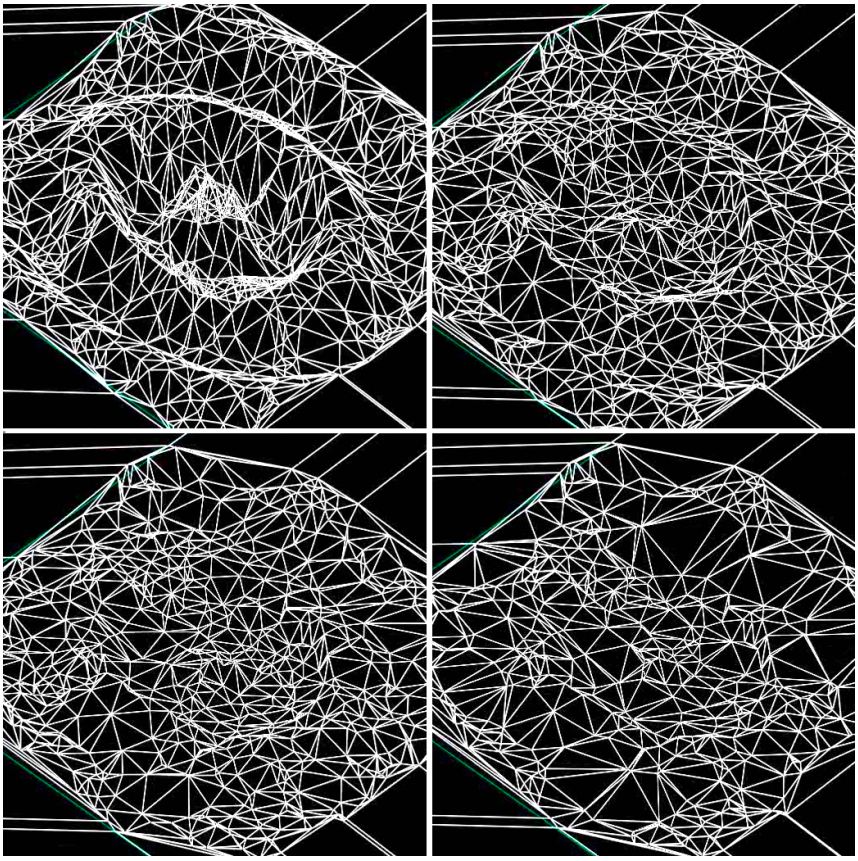


Obrázek 4.4.1: Procházení sousedních vrcholů.



Obrázek 4.4.2: Detekce nadbytečného bodu (vlevo nelze zrušit, vpravo lze smazat bez ztráty informace).

Nelze ovšem procházet všechny vrcholy sítě a kontrolovat u nich výše uvedené kritérium. Vrcholů může být velké množství. Používáme frontu, ve které se hromadí body, které by mohly být nadbytečné. Tj. body, kterým se nějakým způsobem v poslední iteraci změnila výška. Pokud je výše uvedené kritérium splněno, je vrchol označen jako redundantní a před dokončením iterace je ze sítě odstraněn. Ukázku postupného zjednodušování sítě vlivem eroze vidíme na obrázku 4.4.3. Pro přehlednost ještě uvedeme souhrnný algoritmus celého výpočtu, který je uveden níže v kapitole 5.6 .



Obrázek 4.4.3: Postupné zjednodušování sítě vlivem eroze.

5 Podrobnosti řešení

V této kapitole je popsán podobněji způsob řešení jednotlivých bodů zadaní. Popis místy souvisí přímo se zdrojovým kódem programu a API používaných knihoven.

5.1 Prostředky použité k realizaci

Nejdříve si stručně popíšeme možnosti a základní API použitých knihoven.

5.1.1 Knihovna OpenGL

Část textu této kapitoly byla převzata z [Tis03]. Knihovna OpenGL (Open Graphics Library) byla navržena firmou SGI (Silicon Graphics Inc.) jako aplikační programové rozhraní k akcelerovaným grafickým kartám, resp. celým grafickým subsystémům. OpenGL byla navržena s důrazem na to, aby byla použitelná na různých typech grafických akcelerátorů a aby ji bylo možno použít i v případě, že na určité platformě žádný grafický akcelerátor není nainstalován – v tom případě se použije softwarová simulace.

Na některých platformách je možné rozdělení aplikace na dvě relativně samostatné části – serverovou a klientskou. Při vykreslování se potom jednotlivé příkazy (což jsou většinou parametry funkcí OpenGL) přenášejí přes síťové rozhraní. Knihovna OpenGL byla vytvořena tak, aby byla nezávislá na použitém operačním systému, grafických ovladačích a správcích oken. Proto také neobsahuje žádné funkce pro práci s okny, pro vytváření grafického uživatelského rozhraní ani pro zpracování událostí. Tyto funkce lze zajistit buď přímo voláním funkcí příslušného správce oken, nebo lze použít některou z nadstaveb. V našem případě použijeme níže popsanou knihovnu SDL. Pro dosažení co největší nezávislosti na použité platformě zavádí knihovna OpenGL vlastní primitivní datové typy, například GLbyte, GLint nebo GLdouble.

Z programátorského hlediska se OpenGL chová jako stavový automat. To znamená, že během zadávání příkazů pro vykreslování lze průběžně měnit vlastnosti vykreslovaných primitiv (barva, průhlednost) nebo celé scény (volba způsobu vykreslování, transformace) a toto nastavení zůstane zachováno do té doby, než ho explicitně změníme. Výhoda tohoto přístupu spočívá především v tom, že funkce pro vykreslování mají menší počet parametrů a že jedním příkazem lze globálně změnit způsob vykreslení celé scény. Vykreslování scény se provádí procedurálně - voláním funkcí OpenGL se vykreslí výsledný rastrový obrázek. Výsledkem volání těchto funkcí je rastrový obrázek uložený v tzv. framebufferu, kde je každému pixelu přiřazena barva, hloubka, alfa složka popř. i další atributy. Z framebufferu lze

získat pouze barevnou informaci a tu je možné následně zobrazit na obrazovce.

5.1.2 Knihovna SDL

Části této kapitoly byly převzaty z [Tur05]. Knihovna SDL Simple DirectMedia Layer byla navržena jako obecné nízkoúrovňové API (aplikační programové rozhraní) pro tvorbu her a obecně multimediálních aplikací. Z velké části zastřešuje funkce operačních systémů, a tím umožňuje téměř stoprocentní přenositelnost zdrojového kódu.

SDL obsahuje funkce pro vytvoření okna a správu událostí. Dvourozměrná grafika je zahrnuta přímo, 3D grafika je realizována pomocí OpenGL, které má přímou podporu. SDL dále umožňuje práci s audiem, CD-ROM, časovači, obsahuje také podporu vícevláknového programování a mnoho dalšího.

SDL poskytuje jen základní funkce. Například je možné načítat jen obrázky ve formátu BMP. Pokud chceme použít i jiné formáty (v našem případě pro načítání textur), je nutné použít rozšiřující knihovny. Pro načítání textur jsme využili knihovnu SDL_image, která podporuje formáty BMP, GIF, JPEG, LBM, PCX, PNG, PNM, TGA, TIFF, XCF, XPM, XV, JPEG, PNG a TIFF.

V rámci této práce jsme využili i část API pro paralelní programování. Jedná se o funkce pro vytvoření a obsluhu semaforů `SDL_CreateMutex`, `SDL_mutexP`, `SDL_mutexV` a funkce pro práci s vlákny `SDL_CreateThread`, `SDL_DestroyMutex`. Použití těchto funkcí je podrobně popsáno v kapitole 5.3.1 .

5.1.3 Knihovna GLEE

GLEE (OpenGL Easy Extension Library, viz [Glee]) je multiplatformní knihovna, která zpřístupňuje rozšíření OpenGL a funkce zahrnuté ve standardu OpenGL do verze 3.0. Knihovna podporuje operační systémy Windows, Linux, OS-X a FreeBSD. V naší práci využijeme především funkce pro kompilování zdrojových programu pro programování GPU a tím související akce jako nastavování textur, různých parametrů apod.

Pro kompilaci shaderů je potřeba následující kód:

```
const char *fragSource, *vertSource;
GLuint prog, frag, vert;
vert = glCreateShaderObjectARB(GL_VERTEX_SHADER);
frag = glCreateShaderObjectARB(GL_FRAGMENT_SHADER);
```

Funkcím musíme předat přímo zdrojový kód programů:

```
glShaderSourceARB(frag, 1, &fragSource, NULL);
glShaderSourceARB(vert, 1, &vertSource, NULL);
```

Nyní již můžeme program pro vertex shader i pixel shader zkompilovat:

```
glCompileShaderARB(vert);
glCompileShaderARB(frag);
```

Nakonec propojíme oba programy, nahrajeme do grafické karty a spustíme:

```
prog = glCreateProgramObjectARB();
glAttachObjectARB(prog, vert);
glAttachObjectARB(prog, frag);
glLinkProgramARB(prog);
glUseProgramObjectARB(prog);
```

Pokud při tomto postupu vznikne jakákoli chyba, lze ji získat pomocí funkce `glGetProgramiv`. Tato funkce vrátí ukazatel na pole s textem, kde je popis chyb. Pokud se vyskytne nějaká chyba, je použita vestavěná fixní pipeline pro zpracování geometrie a rasterizaci.

Při vykreslování pak použijeme `glGetUniformLocation`, `glActiveTexture`, `glUniform1XYZ` (kde XYZ určuje datový typ parametru, který chceme nastavit).

5.1.4 Knihovna OpenHaptics

Knihovna OpenHaptics [OpenH] je dodávána firmou SensAble (viz dále). Tato knihovna poskytuje jednoduché rozhraní jak pro ovládání jednoho i více haptických zařízení, tak například pro OpenGL grafiku, načítání populárních grafických 3D formátů, řízení scény, mapování silové zpětné vazby na objekty ve scéně, apod. V našem případě se omezíme pouze na silovou zpětnou vazbu a načítání trojrozměrné pozice nástroje v prostoru.

V našem případě ovšem využijeme jen přímé načtení pozice nástroje v prostoru a nastavení silové zpětné vazby zařízení, protože zbytek aplikace máme již hotový.

Inicializaci haptického zařízení provedeme následovně (pro jednoduchost výpisu zde není uvedeno ošetření chybových stavů):

```
HHD hHD = hdInitDevice(HD_DEFAULT_DEVICE);
hGravityWell = hdScheduleAsynchronous(gravityWellCallback,
        this->dataMutex, HD_MAX_SCHEDULER_PRIORITY);
hdEnable(HD_FORCE_OUTPUT);
hdStartScheduler();
```

Poté musíme v obslužné metodě vždy vybrat zařízení se kterým pracujeme a zahájit prováděnou itraci:

```
HHD hHD = hdGetCurrentDevice();
hdBeginFrame(hHD);
```

Získání pozice zařízení v prostoru získáme následovně:

```
hduVector3Dd position;
hdGetDoublev(HD_CURRENT_POSITION, position);
```

Nastavení silové zpětné vazby v Newtonech:

```
hduVector3Dd force;
hdSetDoublev(HD_CURRENT_FORCE, force);
```

Ukončení aktuální iterace:

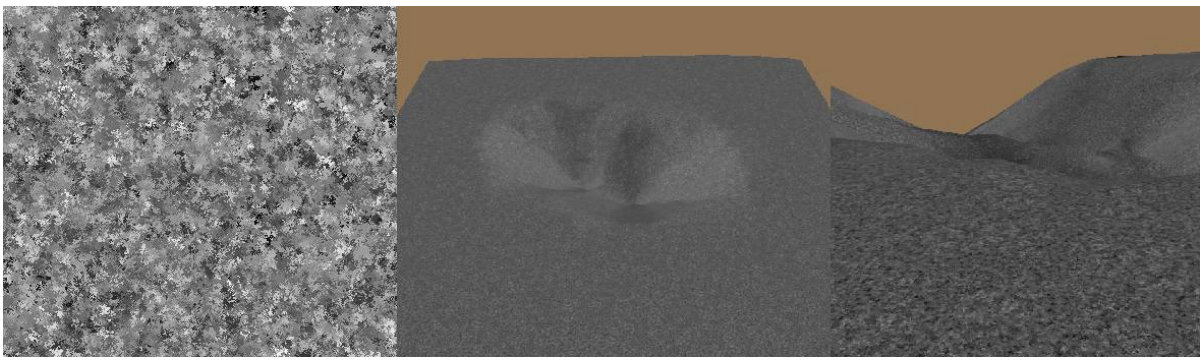
```
hdEndFrame(hHD);
```

5.2 Zlepšení vzhledu

Obecně lze říci, že vizualizace písku v počítačové grafice není dosud vyřešeným problémem. Mým původním návrhem pro zlepšení vzhledu bylo použití Bump Mapy, aby povrch terénu působil více členitě, ale pan inženýr Vaněček mi doporučil vůbec se nezabývat texturou písku a použít náhodnou texturu, protože členitost písčitého povrchu je opravdu značná, a dále mi doporučil použití Subsurface scatteringu pro dosažení reálnějšího vzhledu.

5.2.1 Náhodná textura

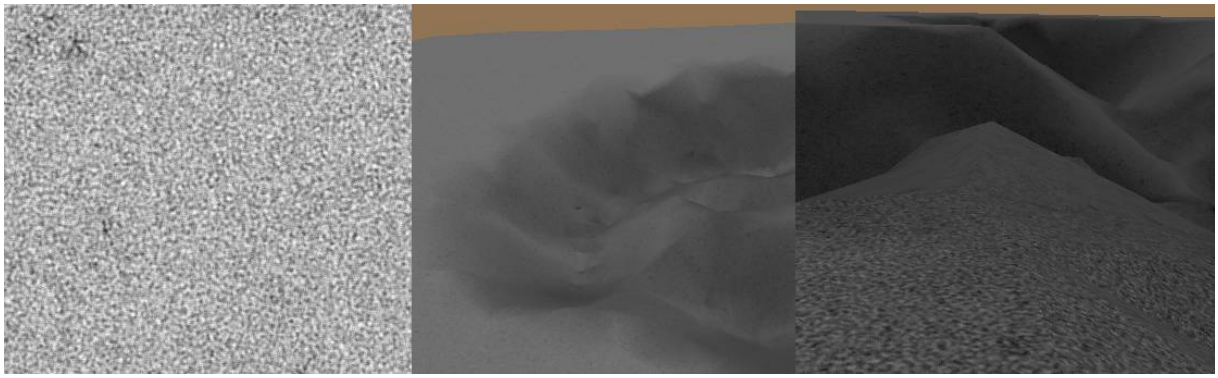
V programu GIMP (viz [Gimp]) jsme vytvořili několik náhodných textur obsahujících šum. Tímto postupem se nám ovšem nepodařilo věrohodně napodobit písek. Ukázku první textury s velkými náhodnými skvrnami vidíme na obrázku 5.2.1.



Obrázek 5.2.1: Textura s poměrně velkými náhodnými skvrnami. Vlevo vidíme část textury, uprostřed pohled na terén z dálky, vpravo detail terénu.

Z dálky vypadají velké skvrny poměrně dobře, ovšem zblízka už moc jako písek

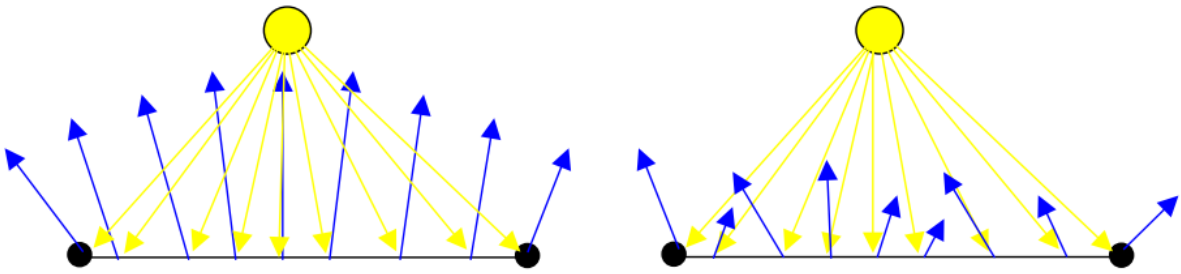
nepůsobí. Pokud použijeme jemnější texturu, je problém opačný. Z dálky celý terén působí příliš jednolitě a blízka pak s použitím bump mappingu vypadá přijatelně.



Obrázek 5.2.2: Textura s jemným šumem. Vlevo vidíme část textury, uprostřed pohled na terén z dálky, vpravo detail terénu.

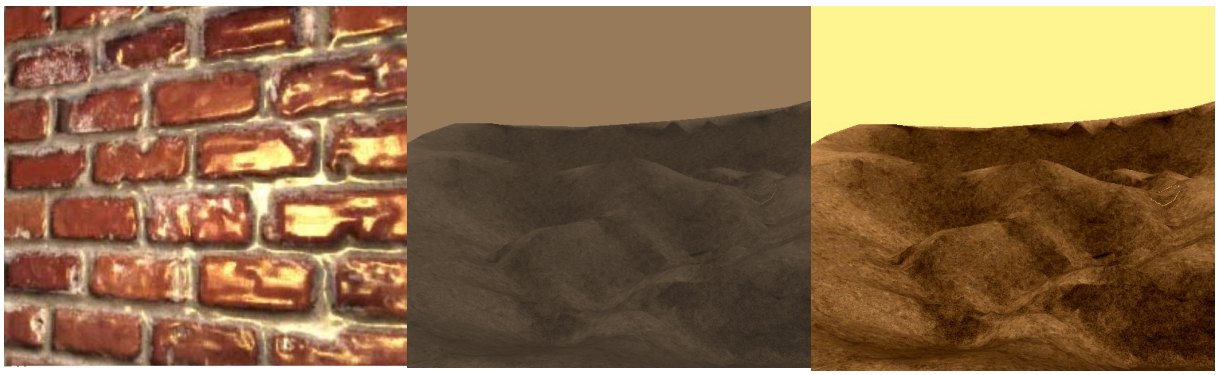
5.2.2 Bump Mapping

Částečně převzato z [Wik1]. Bump mapping je technika texturování, která vytváří iluzi nerovnosti povrchu bez změny jeho geometrie. Iluze nerovnosti povrchu se dosahuje úpravou normály v každém pixelu plochy (viz obrázek 5.2.3). Modifikovaná normála pak ovlivní výpočet osvětlení plochy. Tato technika nedokáže simulovat silně zvlněný povrch, který by zakrýval sám sebe, proto je nutno volit přiměřenou míru zvlnění. Pro uložení informace o hrbolatosti povrchu se používá *bump mapa*. Obvykle je to textura s osmibitovou barevnou hloubkou.



Obrázek 5.2.3: Na obrázku vlevo vidíme standardní mapování normál Phongovým stínováním. Na obrázku vpravo jsou tyto normály nahrazeny předpočítanými normálami z BumpMapy. Obrázek je převzat z [Vil09]

Na obrázku 5.2.4 vidíme výsledný efekt takové modifikace normály. Jedná se o model složený pouze ze dvou trojúhelníků, na které je namapována textura zdi.



Obrázek 5.2.4: Vlevo výsledný efekt modifikace normály na cihlové zdi. Převzato z [Vil09]. Uprostřed je stejná technika aplikována na náš model. Vpravo modifikace kontrastu prostředního obrázku v programu GIMP.

Bump mapping sám o sobě na písku příliš nevynikne, protože členitost písčitého terénu je poměrně velká, ovšem pokud se podíváme na obrázek 5.2.4 vpravo, vypadá už poměrně dobře.

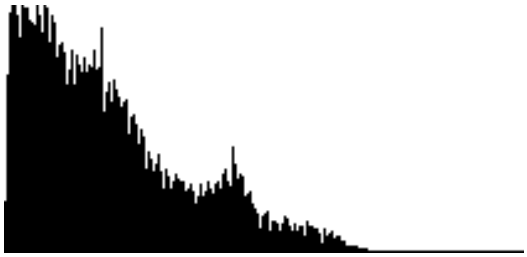
5.2.3 Subsurface Scattering

Částečně převzato z [Wik2]. Subsurface scattering je založen na principu šíření světla, které proniká povrchem materiálu průsvitných objektů. Světlo je rozptýleno a opouští materiál v úplně jiném bodě. Uvnitř materiálu se odráží pod různými vesměs náhodnými úhly. Tato technika je nezbytná pro renderování některých materiálů, jako je například mramor, kůže nebo mléko. Ukázku šíření světla v materiálu vidíme na obrázku 5.2.5. Tato technika kvůli své složitosti nebyla implementována.

Obrázek 5.2.5: Ukázka subsurface scatteringu. Vídíme, jak se světlo šíří pod povrchem materiálu. Převzato z [Tom09].

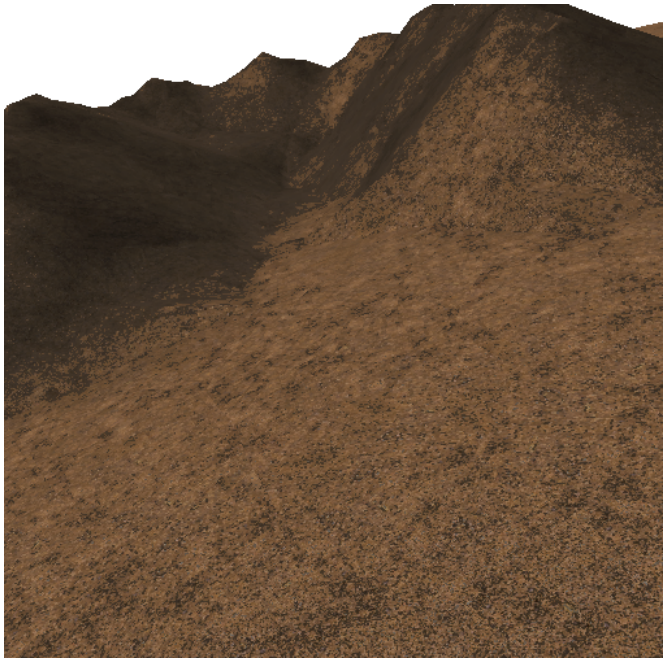
5.2.4 Prahování

Vycházíme-li z obrázku 5.2.4, potřebujeme zvýšit kontrast celé scény, aby působila barevněji a vynikly v ní detaily. Ve zpracování obrazu pro zvýšení barevnosti musíme vytvořit histogram četnosti jednotlivých barev a jeho tvar pak upravit tak, aby co nejlépe využil celé spektrum. Ukázku histogramu pro jasovou složku vidíme na obrázku .



Obrázek 5.2.6: Demonstrační histogram pro jasovou složku obrazu.

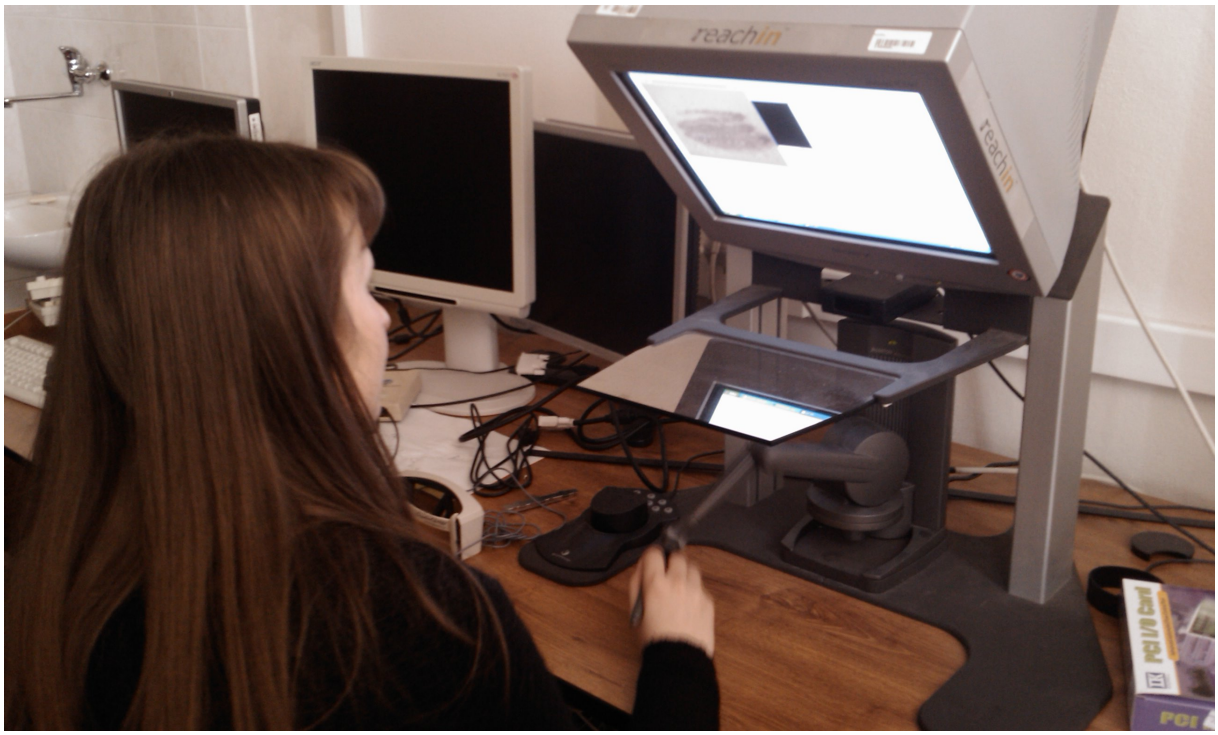
S výhodou můžeme využít již zmíněné shadery a výsledný obraz počítat přímo v pixel shaderu grafické karty. Protože vizualizace scény je v podstatě pořád stejná (podobný odstín), můžeme si dovolit stanovit meze prahování v histogramu napevno. Pro naši vizualizaci jsem experimentálně stanovil konstantu 0.3 na obou koncích histogramu z celkového rozsahu 1. Tyto konce jsou pak utlumeny na úkor zbylé (prostřední části histogramu). Tímto postupem dostaneme výstup uvedený v obrázku .



Obrázek 5.2.7: Výsledek po aplikaci bump mapy, zvýšení konstrastu a barevnosti.

5.3 Použití haptických zařízení

Haptická zařízení typu Phantom jsou schopná snímat trojrozměrnou polohu VR nástroje v pracovní oblasti zařízení. Dále pak je možné snímat natočení nástroje ve směru *X*, *Y* a *Z*. Nástroj má tvar pera s jedním tlačítkem, které lze naprogramovat pro libovolné účely aplikace. Jako vstup může být zadána síla o třech složkách působící na uživatele. Velikost této síly se liší v závislosti na typu zařízení. Námi používaného zařízení v Brně (viz obrázek 5.3.1) může působit maximální silou 10N. Tato síla není nijak veliká, ovšem pokud se skokově změní z 0N na 10N v průběhu jedné iterace (0.001s) může to uživatele překvapit natolik, že není schopen udržet nástroj v ruce.



Obrázek 5.3.1: Demonstrace našeho snažení na MU v Brně. Ing. Barbora Kozlíková (doktorandka zdejší fakulty informatiky) právě spustila naši simulaci virtuálního pískoviště. Virtuální nástroj je ovládán pomocí haptického zařízení a působí na uživatele silovou zpětnou vazbou.

Souřadný systém zařízení se od našeho liší. Námi používané $[X, Y, Z]$ je nutné přemapovat na $[X, Z, -Y]$ (vzorec 5.3.2). Dále je nutné přemapovat rozsah zařízení z našeho $X, Y \in [0;1] \wedge Z \in [-1;1]$ do $X, Z, Y \in [-100;100]$ a opačně. Pokud po stranách uvažujeme ještě bezpečnostní rezervu 50 jednotek souřadného systému zařízení, bude převod souřadnic zařízení na souřadnice naší aplikace vypadat následovně (vzorec 5.3.1):

$$
\begin{aligned}
X_{app} &= (X_{dev} + 150)/200 \\
Y_{app} &= (-Z_{dev} + 150)/200 \\
Z_{app} &= Y_{dev}/200
\end{aligned}
\tag{5.3.1}
$$

kde index *app* symbolizuje souřadný systém aplikace a index *dev* symbolizuje souřadný systém zařízení. Přepočet souřadného systému z aplikačního na souřadný systém zařízení 5.3.2.

$$
[X, Y, Z]_{app} \rightarrow [X, Z, -Y]_{dev} \tag{5.3.2}
$$

5.3.1 Zrychlení aplikace

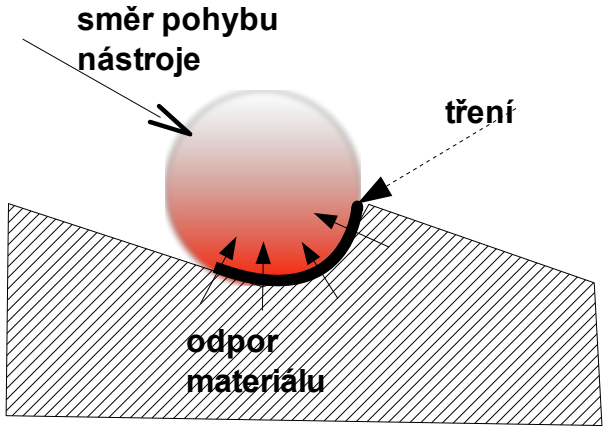
Jako nejnáročnější se jevilo dosáhnout obnovovací frekvence aplikace 1000Hz. Náročnost geometrických a fyzikálních procesů běžících na pozadí aplikace je značná, a tak této rychlosti aplikace nedosahovala. Jediným řešením tedy bylo provádět obsluhu zařízení v jiném vlákně. Knihovna SDL poskytuje poměrně jednoduché API pro paralelní výpočty. Jedinou podmínkou je, že vykreslování musí probíhat v hlavním vlákně, což pro nás není nijak omezující. Pro vytvoření vlákna slouží funkce `SDL_CreateThread`.

Pro zjišťování množství materiálu ve směru pohybu nástroje nebo výšky nástroje nad terénem ovšem potřebujeme přistupovat ke složkám trojúhelníkové sítě a virtuálnímu nástroji. Obsah těchto objektů se ale v průběhu iterace i poměrně razantním způsobem mění a v konzistentním stavu se nachází až těsně před dokončením iterace. Byla navržena jednoduchá vyrovnávací paměť, která uchovává hodnoty potřebné pro výpočet silové zpětné vazby (viz dále) pro „rychlé“ vlákno obsluhující zařízení. Tato paměť se vždy, když se používané datové struktury hlavního vlákna nacházejí v konzistentním stavu aktualizuje, a tak mohou obě vlákna pracovat odděleně nad svými daty. Rychlé vlákno provádí výpočet silové zpětné vazby jen na základě dat uložených ve vyrovnávací paměti. Pro synchronizaci vláken jsme použili jednoduchý binární semafor, který poskytuje knihovna SDL. Semafor deklarujeme `SDL_mutex *dataMutex`. K vytvoření semaforu použijeme funkci `SDL_CreateMutex()`. Ke zrušení semaforu slouží funkce `SDL_DestroyMutex()`. Zamknutí semaforu (operace *P*) provedeme funkcí `SDL_mutexP()`, k odemknutí slouží funkce `SDL_mutexV()`.

5.3.2 Silové složky

V naší aplikaci potřebujeme simulovat silovou zpětnou vazbu složenou ze síly způsobené třením VR nástroje o písek a síly způsobené pronikáním nástroje do materiálu. Třecí síla působí proti směru pohybu nástroje a je závislá na velikosti styčných ploch nástroje

a materiálu. Druhá silová složka (odpor materiálu) závisí na množství materiálu, které se nástrojem snažíme vytlačit. Ukázku působení sil vidíme na obrázku 5.3.2. Musíme dát pozor na to, že i silové složky je třeba přepočítat do souřadného systému podle vztahu 5.3.2.



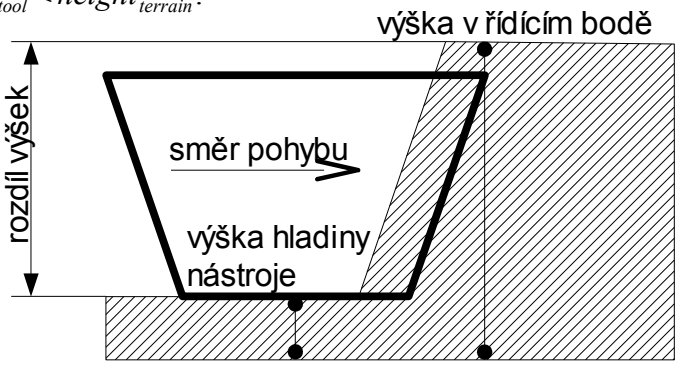
Obrázek 5.3.2: Ilustrační obrázek působení sil na nástroj.

Třecí síla

Třecí sílu spočítáme jako $F_t = f \cdot F_n \quad [N]$, kde f je součinitel smykového tření a F_n je kolmá tlaková síla mezi tělesy. Tlaková síla je závislá na velikosti styčných ploch obou povrchů a koeficient f není snadné určit, proto třecí sílu simulujeme pouze přibližně pomocí techniky gelu. To znamená, že pokud dojde k zaboření nástroje do materiálu, cítí uživatel lehký odpor závislý na velikosti rychlosti nástroje.

Odpor materiálu

Odpor materiálu je způsoben jeho stlačováním. Písek se nerozestoupí ihned po dotyku nástroje do stran a je nejprve nutné vyvinout určitou sílu. Tato síla je přímo úměrná rozdílu výšky hladiny nástroje a výšky povrchu terénu. Odporovou sílu jsme experimentálně stanovili na $F_v = (height_{terrain} - height_{tool}) * 10 \quad [N]$, kde první člen je výška terénu ve středovém bodě nástroje a druhý člen je výška spodní hladiny nástroje (viz obrázek 5.3.3). Tato síla působí jen v případě, že došlo k zanoření nástroje do materiálu, tedy jen pokud je splněna podmínka $height_{tool} < height_{terrain}$.



Obrázek 5.3.3: Rozdíl výšek pro výpočet odporové síly.

5.3.3 Techniky pro modelování materiálů

Předčasné působení síly

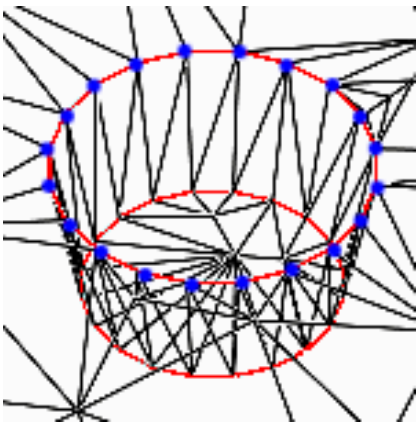
Není vhodné začít působit silou až při bezprostředním dotyku nástroje a materiálu. Zaprvé do sítě vynucujeme zbytečné vrcholy, protože hladina nástroje je shodná s výškou terénu. Zadruhé taková zpětná vazba nepůsobí na uživatele přirozeně. Mnohem lepší je začít působit malou silou těsně předtím než dojde k dotyku obou materiálů. Tuto hodnotu jsme experimentálně stanovili na 0.05 (v souřadném systému aplikace). Výslednou odporovou sílu pak tedy spočítáme jako $F_v = (0.05 + height_{terrain} - height_{tool}) * 10$. Tato síla působí za podmínky $height_{tool} - 0.05 < height_{terrain}$.

Simulace gelu

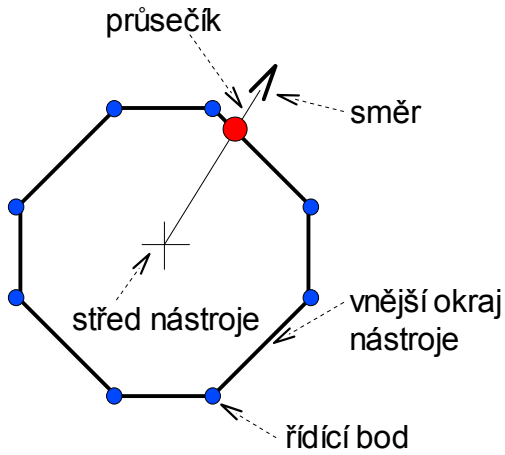
Technikou gelu aproximujeme simulaci smykového tření. Znamená to, že když dojde ke kontaktu nástroje a materiálu a působící síla je pod prahovou hranicí, změníme působící sílu na velikost tohoto prahu. Práh jsme experimentálně stanovili na 0.2N. Odpor reálného gelu není vždy stejný, ale je závislý na rychlosti pohybu předmětu v něm. Tuto vlastnost zanedbáváme.

5.3.4 Zjištění množství materiálu ve směru pohybu

Pro simulaci odporu materiálu ve směru pohybu nástroje potřebujeme znát množství materiálu v síti ve směru pohybu. Opět ovšem nelze pracovat přímo s trojúhelníkovou sítí, protože se v průběhu iterace mění a není v konzistentním stavu. Využijeme opět vyrovnávací paměť. Při aktualizaci rychlého vlákna (hlavní vlákno je v konzistentním stavu) provedeme interpolaci výšky terénu v řídících bodech vnější části VR nástroje. Hodnotu pro každý řídící bod zkopírujeme do vyrovnávací paměti. Zjišťování množství materiálu ve směru pohybu nástroje pak provedeme tak, že vyšleme paprsek ze středu nástroje v určeném směru. Řídící body spojíme fiktivními hranami a tam, kde se paprsek protne s nějakou z těchto hran, pak počítáme výšku terénu. Pokud známe souřadnice průsečíku, lze výšku na jeho souřadnicích spočítat prostou lineární interpolací výšky bodů tvořících fiktivní hranu. Vše je pro lepší pochopení znázorněno na obrázcích 5.3.4 a 5.3.5.



Obrázek 5.3.4: Otisk nástroje. Modré vrcholy znázorňují řídící body vnější oblasti VR nástroje.



Obrázek 5.3.5: Schéma zjišťování množství materiálu v daném směru.

Tímto způsobem lze zjišťovat množství materiálu v libovolném směru. Zjištění ve směru pohybu pak provedeme jednoduchým odečtením aktuální pozice nástroje od předchozí pozice nástroje. Tento přístup ovšem nedává postačující výsledky v případě, že se nástroj téměř nehýbe. Směr pohybu nástroje je pak blízko nuly a snadno se vlivem působení síly mění na opačný. Tím dochází k jakémusi chvění nástroje. Pro odstranění toho nežádoucího efektu jsme navrhli jednoduchý způsob odstranění chvění (viz kapitola 5.3.5).

5.3.5 Omezení chvění nástroje

Ke chvění nástroje dochází, pokud je rozdíl mezi polohami nástroje blízký nule. Pokud se síla počítá z množství materiálu ve směru pohybu nástroje a směr se neustále mění, síla se také často mění. Pro odstranění tohoto jevu jsme navrhli jednoduché řešení:

- pro každý řídící bod vnější části VR nástroje zjistíme množství materiálu ve směru určeném středem nástroje a daným řídícím bodem podle postupu v kapitole 5.3.4 a spočítáme silovou zpětnou vazbu v tomto směru,
- poté všechny výsledné síly sečteme a spočítáme průměrnou sílu působící na uživatele.

Tímto postupem eliminujeme chvění a zároveň zachováme dostatečně věrohodnou silovou zpětnou vazbu. Celková síla tedy bude:

$$F = F_V + F_H,$$

kde $F_v = (0.05 + height_{terrain} - height_{tool}) * 10,$ (viz výše)

a $F_H = \frac{1}{N} \cdot \sum_{i=1}^{N} F_i,$ kde N je počet řídících bodů nástroje a F_i je síla působící proti směru od středu nástroje ke každému z řídících bodů nástroje v závislosti na množství materiálu.

5.3.6 Rozhraní pro haptické zařízení

Aby haptické nebo jiné zařízení mohl používat i člověk, který nerozumí vnitřní struktuře naší aplikace, vytvořili jsme zapouzdřující třídu HapticInterface, která se nachází v souborech HapticInterface.cpp a HapticInterface.h. Tato třída automaticky vytváří obslužné vlákno a stará se o synchronizaci a aktualizaci dat pro obě vlákna. Navíc obsahuje některé pomocné metody, jako například zjištění množství materiálu ve směru pohybu nástroje. Uživatel pak tedy jen dopíše tělo metody HapticHandler (ve stejnojmeném souboru) podle svých požadavků a typu používaného zařízení. Pro aktivaci tohoto rozhraní je nutné definovat makro `ENABLE_EXTERNAL_TOOL_POSITION_CONTROL`. Tím dojde k připojení knihoven pro obsluhu haptického zařízení a VR nástroj je ovládán pomocí jiného zařízení než je myš. (Kolega Kolařík měl demonstrovat tento přístup na použití školního tabletu. Po nákupu haptického zařízení na ZČU pak toto rozhraní využil spolu se silovou zpětnou vazbou implementovanou v Brně přímo na haptickém zařízení.)

5.3.7 Interpolace zpětné silové vazby

Synchronizací vláken jsme sice vyřešili problém s obnovovací frekvencí, ovšem pokud se údaje aktualizují s nižší frekvencí, nastává při aktualizaci více či méně citelný skok, protože síla se počítá z údajů, které nejsou úplně aktuální. Tento problém jsme vyřešili tím, že se velikost síly interpoluje mezi jednotlivými aktualizacemi. Tím dosáhneme spojitého silového vjemu. To, že došlo k aktualizaci vyrovnávací paměti hlavním vláknem, indikuje proměnná `updatePerformed`. Při každé aktualizaci si uložíme, kolika iteracemi proběhlo rychlé vlákno. Z nových údajů spočítáme výslednou sílu a velikost této síly pak spojitě měníme podle aktuální iterace. Předpokládáme, že se počet iterací nijak výrazně skokově nemění.

5.3.8 Shrnutí

Nutno podotknout, že ovládání aplikace pomocí haptických zařízení ještě není ve své konečné fázi. Navržené rozhraní a omezení chvění bylo možné použít pro ovládání dvou typů zařízení. Na propojení s haptickým zařízením v Brně a výpočtu silové zpětné vazby jsme pracovali společně s kolegou Flasarem. Ovládání funguje podle očekávání, ovšem je nutné vylepšit silovou zpětnou vazbu tak, aby působila víc realisticky. V současné době uživatel cítí odpor, ovšem výsledný vjem příliš nepřipomíná písek. Cílem našeho snažení ovšem bylo vyzkoušet, zda je možné vůbec aplikaci propojit s haptickým zařízením na MU v Brně, což se povedlo. Výsledky tohoto výzkumu jsou použity například v práci kolegy Kolaříka [Kol09].

5.4 Porovnání s existující metodou

Kolega B. Beneš nám zapůjčil pro porovnání zdrojové kódy programu popsaného v článku [Ben06]. Díky němu a několika emailů od jeho studenta jménem Enkhtuvshin Dorjgotov se mi podařilo program přeložit. Tento program je kompatibilní s Visual C++ verze 6.00. Podle instrukcí (osobní komunikace s E. Dorjgotovovem, 10. 11. 2008) E. Dorjgotovova je k jeho kompilaci potřeba ještě knihovna CHAI 3D verze 1.51. Po zběžném prozkoumání zdrojových kódů objevíme závislost na knihovně GLUT (viz [Glut]). K programu bohužel nebyla žádná dokumentace.

5.4.1 Měření řešení s pravidelnou sítí

Získaný projekt jsme se nejdříve pokoušeli přeložit kompilátorem GCC, který byl používán i k vývoji naší aplikace – bohužel neúspěšně. Postupně jsme vyzkoušeli všechna veřejná vydání knihovny Chai3D (1.25, 1.41, 1.42, 1.51 a 1.61). Od ZČU jsme získali Microsoft Visual C++ verze 2008 a dále jsme pokračovali s tímto vývojovým nástrojem. Bohužel ani ve Visual Studiu se nepodařilo s žádnou verzí Chai3D program úspěšně přeložit a provést tak potřebná měření. Program bylo nutné ručně poměrně dlouho upravovat. Celou práci navíc komplikuje to, že většina kódu je vepsaná přímo do Chai3D a tak je orientace v kódu poměrně nepřehledná. Po otevření projektu a spuštění kompilace dostaneme následující chybová hlášení (obrázek 5.4.1).

50	error C2039: 'ForcesOFF' : is not a member of 'cGeneric3dofPointer'	hapticsandserver.cpp	78:		HapticSandServer
49	error C2039: 'ForcesON' : is not a member of 'cGeneric3dofPointer'	hapticsandserver.cpp	73		HapticSandServer
31	error C2039: 'getDevicePosition' : is not a member of 'cGeneric3dofPointe	hapticsandserver.cpp	57(		HapticSandServer
33	error C2039: 'getDevRotMatrix' : is not a member of 'cGeneric3dofPointer'	hapticsandserver.cpp	57:		HapticSandServer
70	error C2039: 'getDynamicFriction' : is not a member of 'cMesh'	hapticsandserverdlg.c	48:		HapticSandServer
30	error C2039: 'getProxyPosition' : is not a member of 'cGeneric3dofPointer	hapticsandserver.cpp	52!		HapticSandServer

Obrázek 5.4.1: Chyby při kompilaci programu prezentovaného v [Ben06].

Tyto chyby jsou poměrně matoucí – např. funkce `getDevicePosition` využívaná v aplikaci byla dostupná jen do verze 1.42, jenže jsou zde využívány některé funkce, které jsou dostupné až ve verzi 1.51. Postupně jsem tedy vyzkoušel všechna zveřejněná vydání knihovny Chai3D. Bohužel s žádnou verzí se nepodařilo program úspěšně přeložit, a provést tak potřebná měření. Podle B. Beneše už není E. Dorjgotov nadále dostupný, a tak nelze využít jeho pomoci.

Po poměrně dlouhé době se podařilo pomocí různých úprav program přeložit a spustit. Pro úspěšnou kompilaci je potřeba provést následující úpravy. Je potřeba upravit proměnné typu *register* na *int*. To se týká například souborů CsandSurface.cpp. Dále je potřeba

v souboru HapticSandServerDlg.cpp změnit 174. řádku z `const nSize` na `const int nSize`.

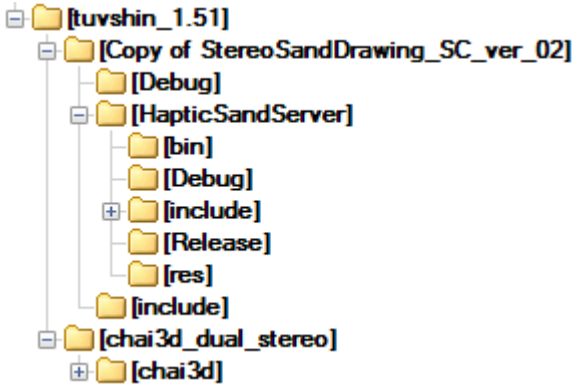
V souboru HapticSandServer.cpp je potřeba odstranit koncovku `.h` u C++ hlavičkových souborů (to se týká knihoven iostream, fstream, iomanip). Ve stejném souboru musíme proměnnou `float depth` inicializovat na nulu.

V souboru HapticSandServer.cpp je potřeba změnit způsob práce s haptickým zařízením, protože od Chai3D verze 1.51 jsou zavedené tzv. proxy pro jednotlivá zařízení. Navíc se změnil název některých dalších funkcí (např. pro ovládání síly). Všechny níže zmíněné funkce je potřeba přepsat podle tabulky 5.4.1.

Řádka	Původní kód programu	Nový kód programu
170	`app->tool->getDevicePosition()`	`app->tool->getProxy()->getDeviceGlobalPosition()`
171	`app->tool->getProxyPosition()`	`app->tool->getProxy()->getProxyGlobalPosition()`
172	`app->tool->getDevRotMatrix()`	`app->tool->getGlobalRot()`
734	`tool->ForcesON()`	`tool->setForcesON()`
782	`tool->ForcesOFF()`	`tool->setForcesOFF();`
846	`radius = edge / 2.0`	`radius = edge / 2.0f`

Tabulka 5.4.1: Úprava souboru HapticSandServer.cpp.

Knihovnu Chai3D je potřeba umístit na stejnou adresářovou úroveň, jako je adresář s projektem tohoto programu, a to do složky `chai3d_dual_stereo`. Potřebnou adresářovou strukturu vidíme na obrázku 5.4.2.



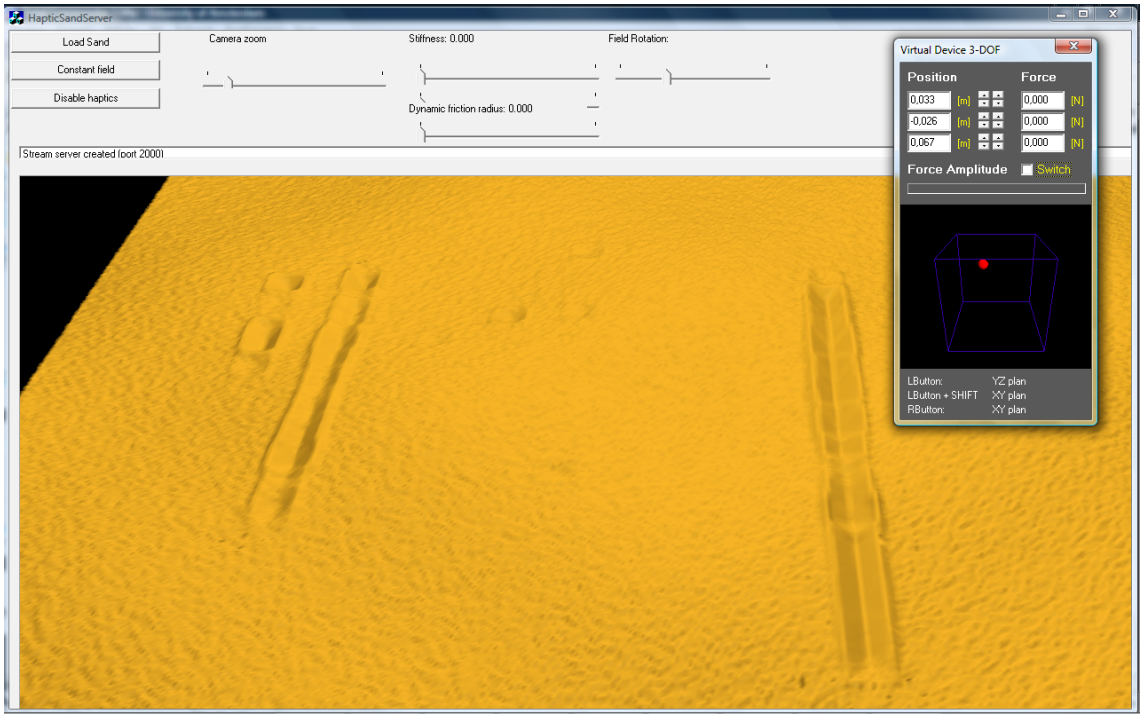
Obrázek 5.4.2:Struktura adresářů vyžadovaná aplikací s pravidelnou sítí.

Po těchto a jiných drobných změnách lze spustit linkování programu. Zde se nám zobrazí chybové hlášky o tom, že knihovny `PhantomDriver.lib`, `dhdms.lib` nelze

připojit. Po prozkoumání dokumentace k Chai3D zjistíme, že knihovnu `PhantomDriver.dll` od verze 1.51 nemusíme linkovat vůči `PhantomDriver.lib`. Potřebná volání jsou přímo v Chai3D. Stačí tedy ve Visual C++ otevřít nastavení projektu a v záložce *Linker* odebrat `PhantomDriver.lib`. O druhé knihovně jsem nic nenašel. Metodou pokus a omyl jsme přišli na to, že ji také stačí odebrat z nastavení linkeru.

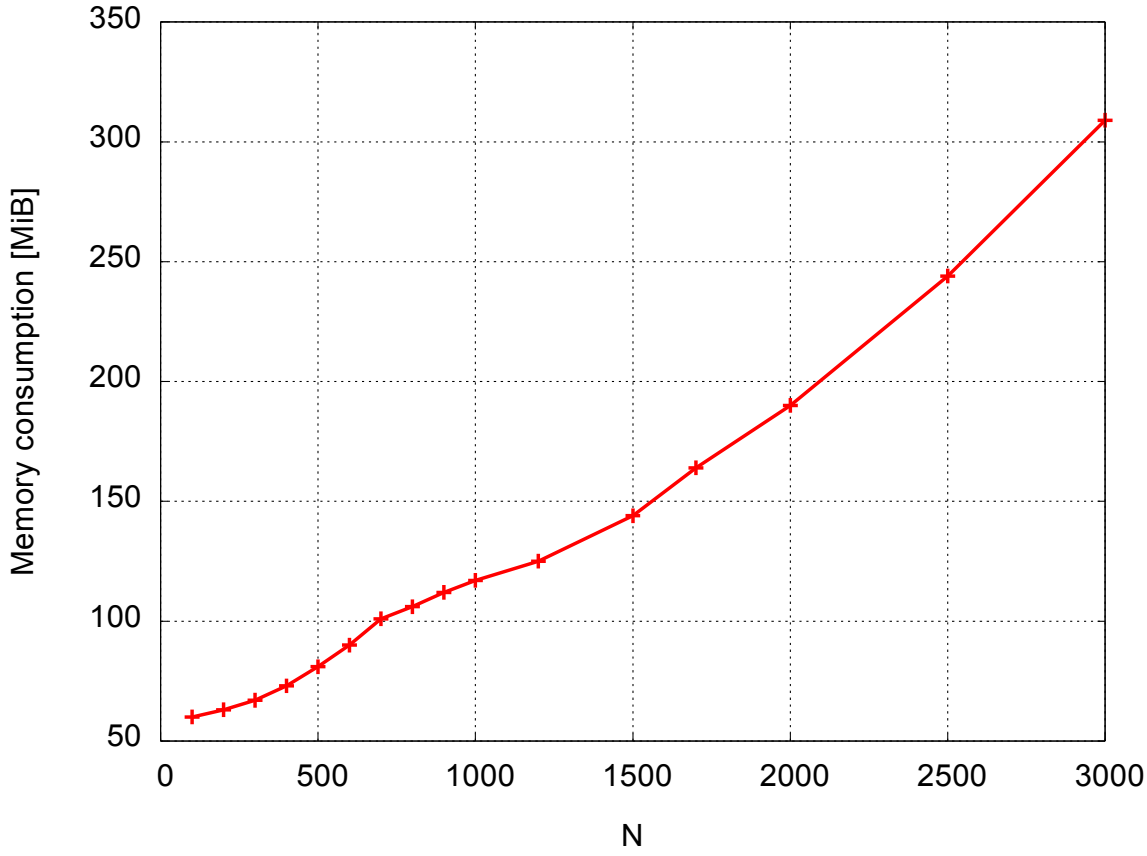
Nyní již kompilace proběhne v pořádku. Po spuštění aplikace se ale objeví jen černé okno s několika červenými čárkami. To je způsobeno tím, že má vykreslovací okno příliš velké rozlišení. Starší verze Visual Studia používala pro tvorbu GUI knihovnu MFC. Pokud vytvoříme formulář v návrháři, vytvoří se také soubor s příponou `.rc`. Nová verze Visual Studia už bohužel neobsahuje grafický návrhář těchto aplikací, a tak musíme soubor upravit ručně. U každého GUI prvku máme několik parametrů. První dvě čísla v seznamu udávají vlastnosti *Top* a *Left*, další pak *Width* a *Height*. Úpravou těchto vlastností lze alespoň přibližně okno zmenšit a posunout ovládací prvky tak, aby byly vidět.

Knihovna Chai3D obsahuje jakési virtuální haptické zařízení pro testovací účely. Po úspěšném spuštění aplikace jsem ho využil k rýpání do písku. Souřadné osy nejsou přepočítány, do souřadnic jaké používá OpenGL, to je trochu matoucí, ale aplikace se dá i tak ovládat. Výstup programu po několika vrypech do písku vidíme na obrázku 5.4.3 (včetně virtuálního haptického zařízení).



Obrázek 5.4.3: Výstup zkompilovaného programu založeného na pravidelné síti.

Aplikace je interaktivní, pokud je rozlišení výškové mapy pod 1000x1000. Z grafu na obrázku 5.4.4 vidíme, jak roste spotřeba paměti s rostoucím rozlišením sítě. Rozměr výškové mapy lze nastavit v souboru CSandSurface.cpp pomocí konstanty `WIDTH`.



Obrázek 5.4.4: Spotřeba paměti řešení využívajícího pravidelnou síť. N udává rozměr matice výškového pole použitého k reprezentaci terénu který je NxN. ***Celkový počet bodů v síti je tedy N²***.

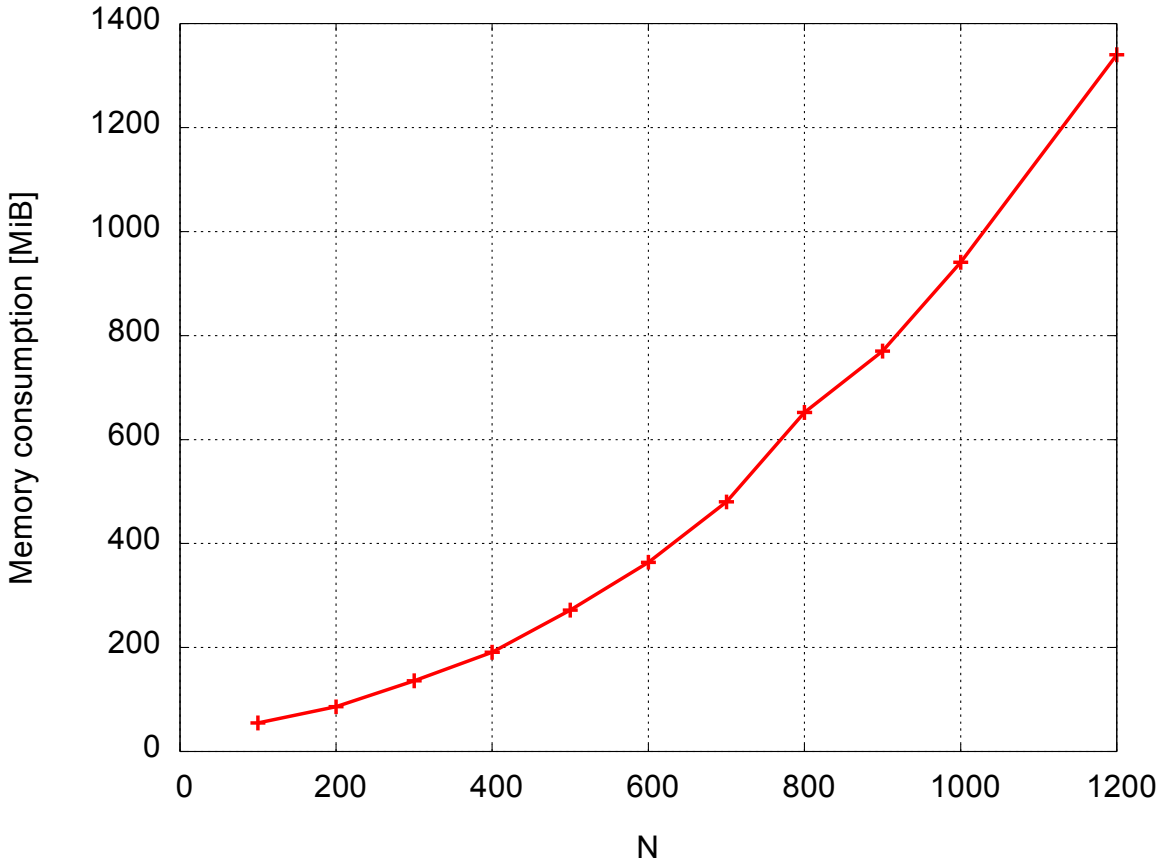
V budoucnu by bylo možné pro testovací účely napsat vlastní program, založený na pravidelné síti, a obě řešení detailně porovnat. To ale přesahuje rámec této práce.

5.4.2 Měření paměti

Do našeho programu byla implementována třída `TimeCounter`, s pomocí které lze pohodlně měřit všechna potřebná data. Náš model má mnoho parametrů, které lze měřit, ať už jsou to závislosti na rozměru, tvaru a geometrii sítě, nebo parametry virtuálního nástroje, nebo vlastnosti eroze. Vybral jsem několik parametrů, které jsou z mého pohledu důležité a mezi nimi jsem měřil různé závislosti. Těmito parametry jsou rozměr sítě, délka rýhy nástroje po diagonále (čtverce, který tvoří naše virtuální pískoviště), časy výpočtu různých dílčích algoritmů a počet ovlivněných elementů v síti. V našem pokusu je plochý terén tvořený *N*

rovnoměrně rozloženými body. Nástrojem postupně ryjeme rýhu na diagonále. Nástroj má poloměr 0.04, hloubku zanoření -0.1 (hladina terénu je 0) a počet řídících bodů 12. Pokud je délka rýhy v průběhu nějakého pokusu neměnná (fixní parametr), je to 80% délky diagonály. Výsledky těchto měření byly publikovány v [Pur09]. Byl napsán jednoduchý skript v jazyce PERL, který provede příslušná měření a sestaví tabulky pro program GNU Plot (viz [Gplot]).

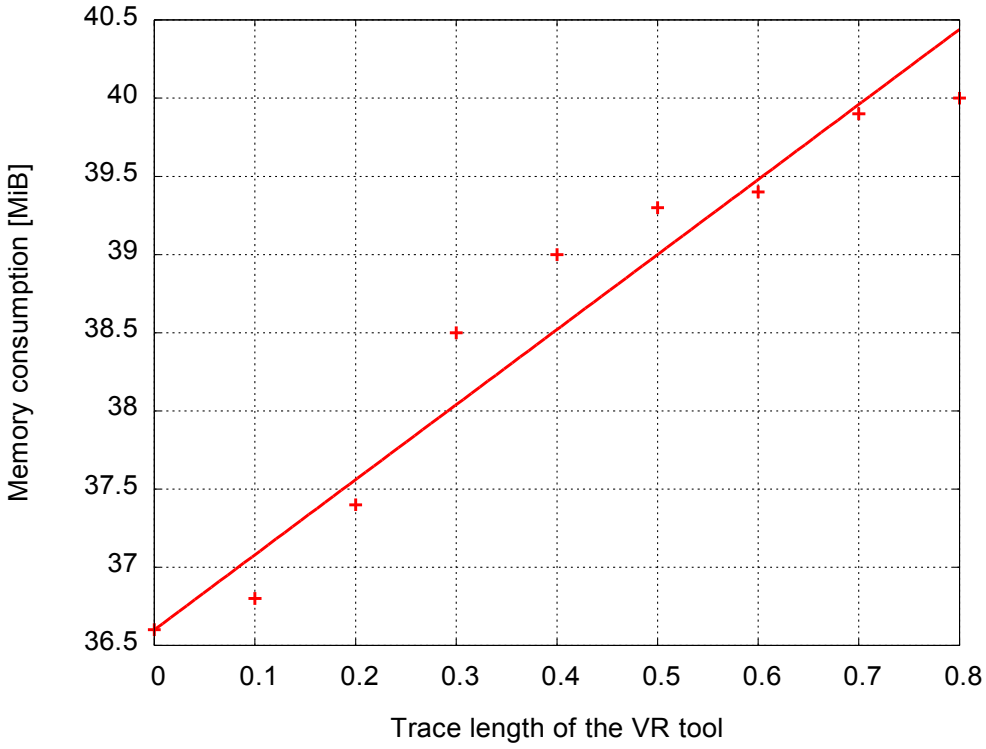
Jako u předchozího programu jsem provedl měření spotřeby paměti (viz obrázek 5.4.5). Tento test ovšem není zcela objektivní, protože pomocí nepravidelné sítě o velikosti např. 10 000 vyjádříme mnohem více detailů než pomocí pravidelné sítě o rozměrech 100x100 buněk.



Obrázek 5.4.5: V grafu vidíme spotřebu paměti aktuální verze našeho programu. Pro zachování měřítka s grafem 5.4.4 je zde N jako druhá odmocnina z počtu vrcholů v síti. ***Celkový počet bodů v síti je tedy N²****, tzn. 1 440 000 vrcholů pro nejvyšší hodnotu.*

Nutno podotknout, že aplikace je interaktivní přibližně do 30 000 bodů, pokud má síť nad 100 000, přestává postačovat stávající vizualizační metoda. Oproti měření v práci [Kad07] se zvýšila spotřeba paměti oproti 300MiB na téměř 1GiB! To je způsobeno seznamem trojúhelníků pro každý vrchol, které pro vizualizační účely zavedl J. Sedmihradský.

Paměťovou náročnost v závislosti na délce rýhy podél diagonály čtverce, který reprezentuje model terénu, vidíme na obrázku 5.4.6. Celková délka diagonály je 1 (100%). Vidíme, že tato závislost je u naší metody přibližně lineární.

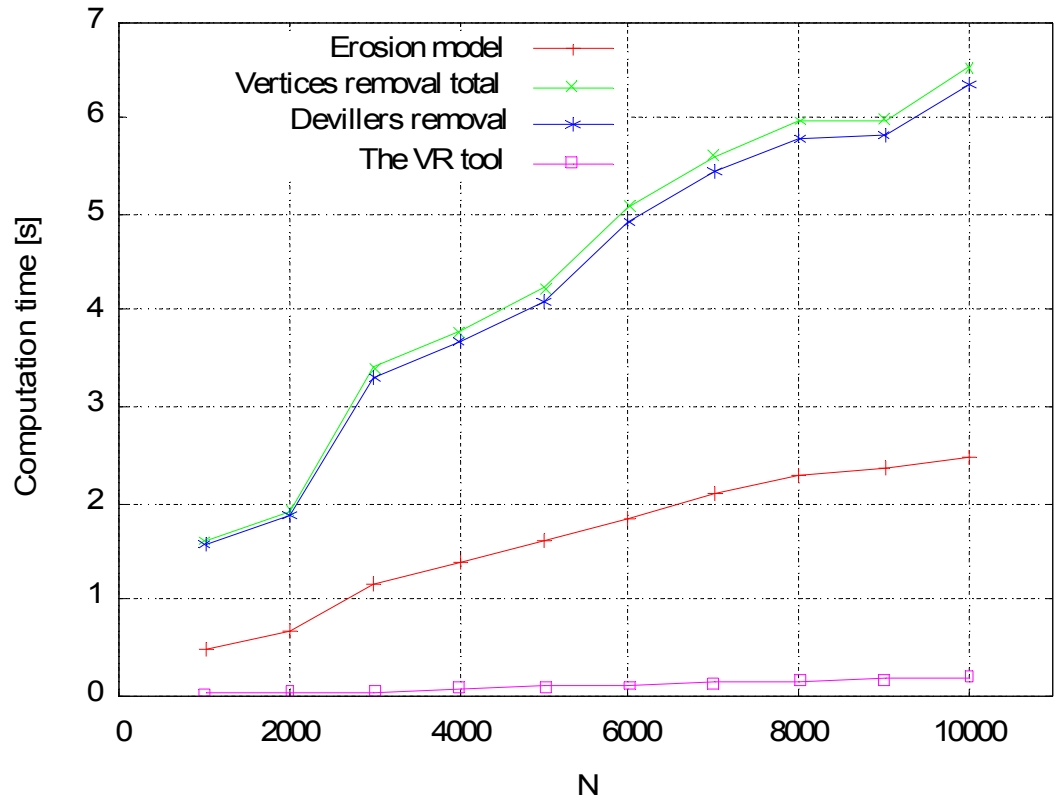


Obrázek 5.4.6:Paměťová náročnost v závislosti na délce rýhy.

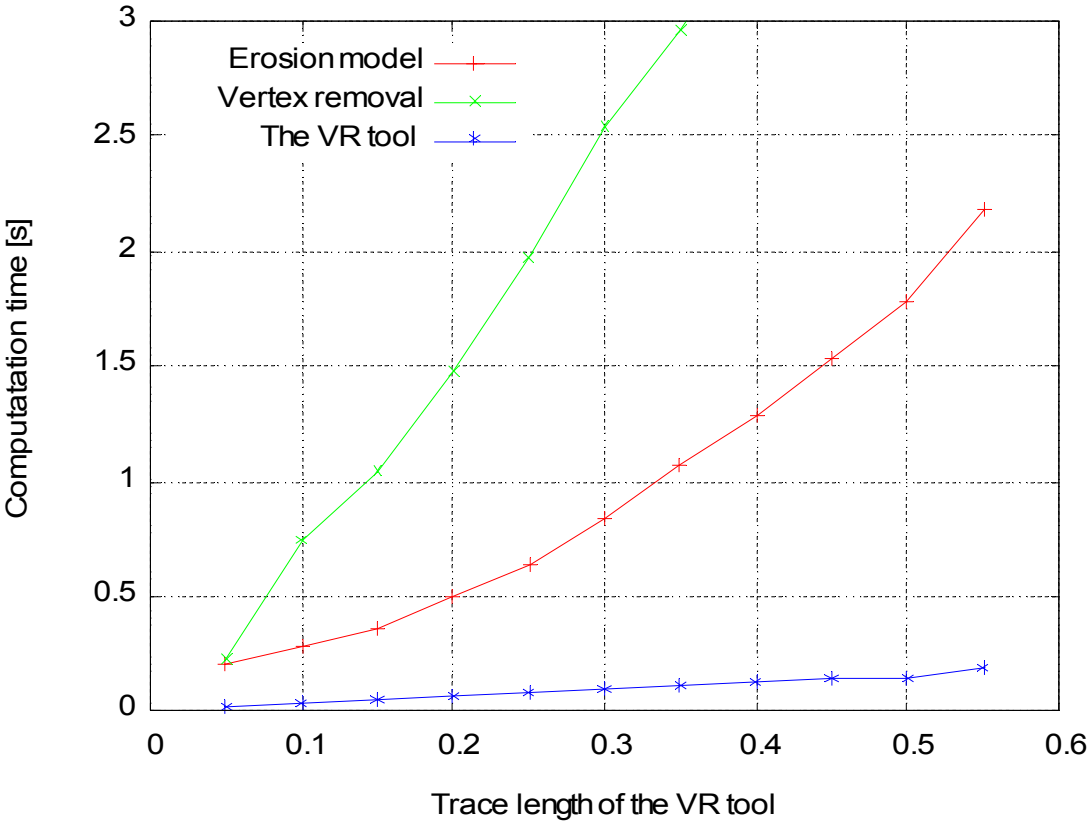
Pro konkurenční program [Ben06] kolegy Beneše využívajícího pravidelnou síť je paměťová náročnost přibližně konstantní pro všechny délky rýhy a pohybuje se okolo 75MiB pro referenční výškovou mapu o rozměrech 500x500 buněk. V tomto testu jsme tedy dosáhli poloviční úspory paměti.

5.4.3 Měření času

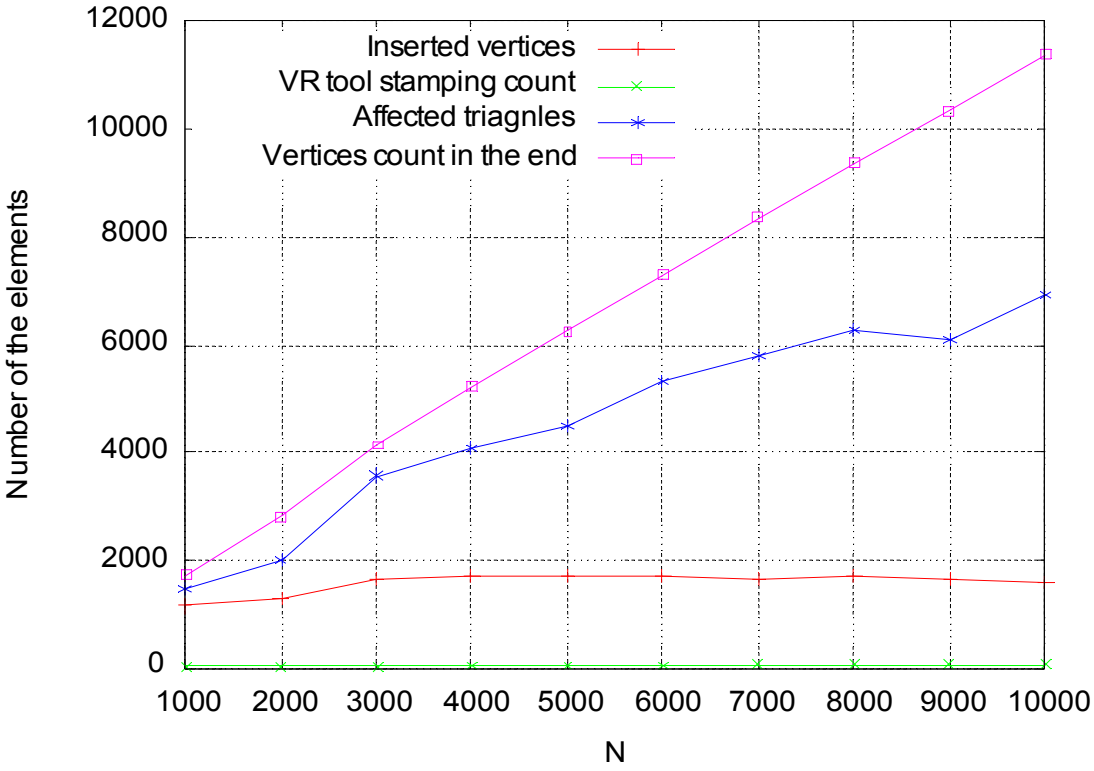
V následujících testech byly měřeny výše popsané parametry. Výsledky testů spotřeby času jsou vidět na obrázku 5.4.7, kde měříme čas výpočtu v závislosti na na velikosti sítě. Je zřejmé, že časově nejnáročnější je odebírání nepotřebných vrcholů ze sítě. Naměřená složitost $O(n^{0.5})$ naznačuje použití algoritmu procházky. Odhad složitosti pro výpočet eroze je $O(n^{0.5})$. Na obrázku 5.4.8 vidíme závislost doby výpočtu na délce rýhy nástroje. Odhad složitosti je $O(n)$ pro simulaci eroze a $O(n^{1/4})$ pro odebírání vrcholů. Výsledky testů počtu ovlivněných elementů v síti vidíme na obrázcích 5.4.9 a 5.4.10. Závislost počtu ovlivněných elementů na velikosti sítě a na délce rýhy nástroje je přibližně lineární.



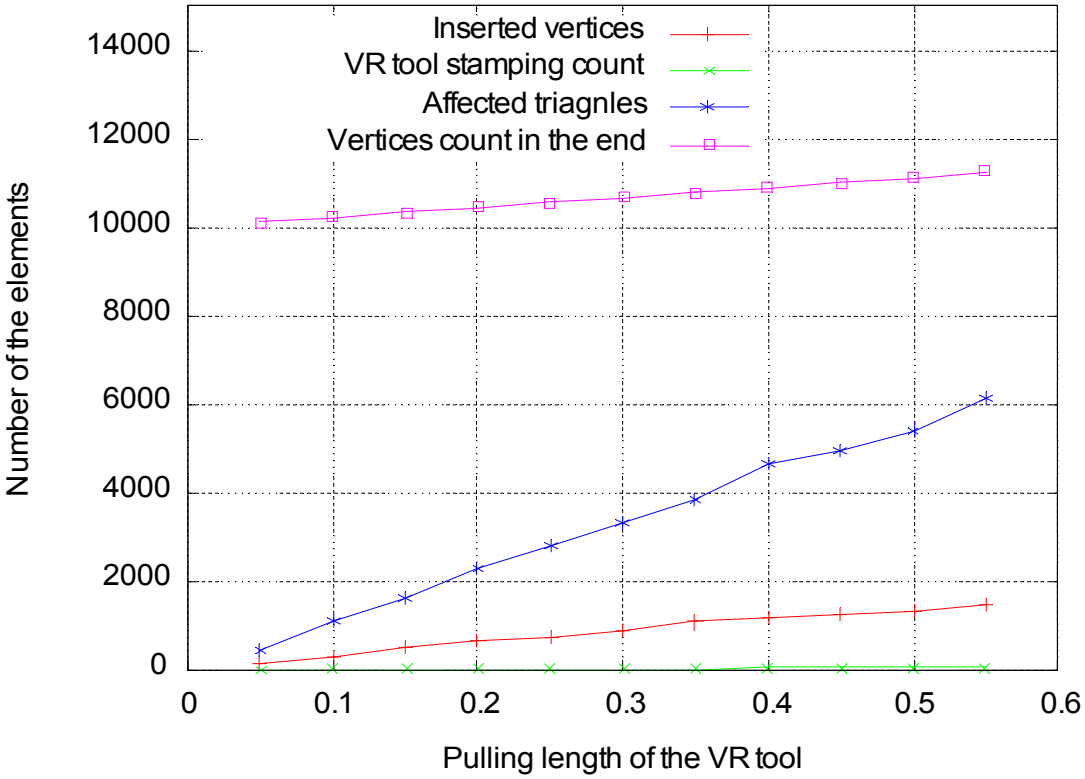
Obrázek 5.4.7: Závislost doby výpočtu důležitých částí programu na velikosti trojúhelníkové sítě.



Obrázek 5.4.8: Závislost doby výpočtu důležitých částí programu na délce rýhy nástroje.



Obrázek 5.4.9: Počet ovlivněných entit během tažení nástroje v závislosti na velikosti sítě.

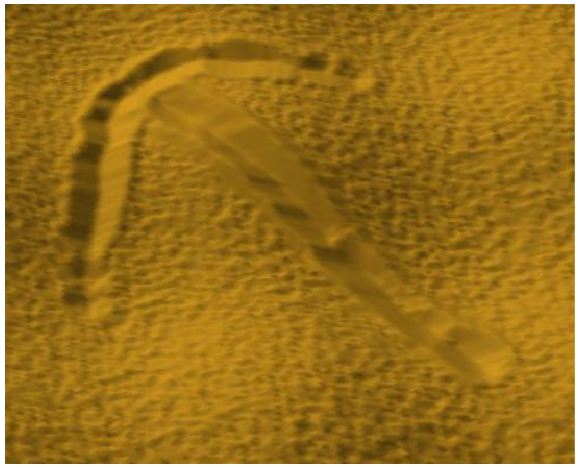


Obrázek 5.4.10: Počet ovlivněných entit během tažení nástroje v závislosti na délce rýhy.

5.4.4 Vizuální porovnání

Provedli jsme také vizuální porovnání obou programů. Toto porovnání bylo prezentováno na konferenci CESCG 2009 a na ACM Student Research Competition 2008. Jedná se o jednoduchý test, kdy jsem jedním z nástrojů vytvořil stejnou šipku, jaká je prezentována v článku [Ben06]. Porovnání výstupů obou programů vidíme na obrázcích 5.4.11 a 5.4.12.

Obrázek 5.4.11: Výstup našeho programu založeného na nepravidelné trojúhelníkové síti (TIN).



Obrázek 5.4.12: Výstup programu prezentovaného v článku [Ben06] založeného na pravidelné síti.

Z tohoto porovnání vidíme, že řešení na pravidelné síti má propracovanější vizualizaci.

5.5 Opravené chyby a drobné úpravy

V průběhu řešení výše uvedených problémů vzniklo několik požadavků například na sofistikovanější konfigurační soubory. Konfigurace aplikace umístěna J. Sedmihradským v hlavním souboru main.cpp se přesunula do pro tento účel nově vytvořené třídy Config. Konfigurace může obsahovat čísla i řetězce, komentáře a snadno lze přidávat další parametry.

Dále došlo ke zjednodušení API pro fyzikální vrstvu. Událost pro výpočet eroze, původně bez parametrů, obsahovala v době předání programu šest parametrů. Tyto parametry jsou užitečné pro aktuálně používaný systém eroze, ale pro jiné modely komplikují použití rozhraní pro fyzikální vrstvy. Všechny parametry byly nahrazeny jediným, kterým je výše zmíněná konfigurace.

Při spolupráci s kolegou Kolaříkem jsme objevili několik chyb. Jedna se týkala virtuálních hran použitých při výpočtu *CDT* (to jsou takové hrany, které nenastavují vrcholům sousednosti). Konstruktor virtuální hrany neinicializoval některé proměnné, což vedlo k občasným náhodným chybám. Tato chyba je opravena. Několik chyb se vyskytlo také

v odstraňování jednotlivých vrcholů. Některé chyby byly opraveny, ovšem některé se nepodařilo úplně eliminovat, protože se vyskytují velmi ojediněle.

5.6 Kompletní algoritmus výpočtu

Nejdříve popíšeme symboly použité v popisu algoritmu:

- L_e je seznam vrcholů, které budou erodovány,
- L_r je seznam vrcholů použitý k detekci redundantních vrcholů,
- α_{flat} je hodnota mezního úhlu, který určuje, jak dalece se mohou vrcholy odchylovat od přibližné roviny, kterou vzájemně tvoří, aby je ještě bylo možné ze sítě odstranit. Pokud je α_{flat} větší, více vrcholů je odstraněno ze sítě a naopak. Správná volba α_{flat} je uvedena v kapitole 4.4 ,
- α_m je mezní úhel materiálu pro výpočet eroze. Pokud je úhel mezi výše položeným vrcholem a níže položeným vrcholem menší než α_m, nedojde již k přesunu materiálu vlivem eroze. Experimentální volba α_m je uvedena v kapitole 3.3 ,
- „rychlé“ vlákno je vlákno programu obsluhující haptické zařízení. Toto vlákno provádí výpočet silové zpětné vazby a běží paralelně s hlavním programem.

Protože je celý algoritmus výpočtu velmi komplikovaný, uvedeme celý postup jen zjednodušeně::

- Výpočet Delaunayovy triangulace na počáteční množině vrcholů.
- V každé iteraci:

1. Vynucení deformací na současné pozici VR nástroje:

   a) Interpolace výšky terénu v kontrolních bodech vnější části VR nástroje.

   b) Vynucení množiny hran vnější části nástroje za použití *CDT*.

   c) Vynucení hran tvořících vnitřní část nástroje.

   d) Redukce výšky bodů ležících ve vnitřní části nástroje.

   e) Naplánování odstranění přebytečných vrcholů.

   f) Přidání změněných vrcholů do seznamů L_e a L_r.

2. Eroze materiálu; pro každý vrchol P_k v seznamu L_e:

a) Výpočet rozdílu výšky mezi vrcholem P_k a každým jeho sousedem P_i (z P_k do P_i vede hrana). Vybrání nejvhodnější metody pro erozi.

b) Výpočet přírůstku materiálu pro vrcholy P_i ležící níže než P_k. Musí platit, že úhel mezi hranou tvořenou vrcholem P_k a každým jeho sousedem P_i a vodorovnou hranou procházející bodem P_k je větší než α_m.

c) Redukce výšky vrcholu P_k, navýšení vrcholů P_i ležících níže než P_k.

d) Vyloučení vrcholů, u kterých nedošlo k přesunu materiálu, ze seznamu L_e.

3. Pro každý vrchol P_l ze seznamu L_r:

a) Test úhlu α_i mezi hranou tvořenou vrcholem P_l a každým jeho sousedem P_i a vodorovnou hranou procházející bodem P_l.

I. Pokud $\alpha_i > \alpha_{flat}$ odebereme vrchol P_l ze seznamu L_r.

Pokud pro všechny vrcholy P_i platí $\alpha_i \leq \alpha_{flat}$, odebereme vrchol P_k ze seznamu L_r, odebereme vrchol ze seznamu L_e, odebereme vrchol z trojúhelníkové sítě.

4. Aktualizace vyrovnávací paměti pro haptiku:

a) Pokud „rychlé“ vlákno přistupuje k vyrovnávací paměti, čekáme na uvolnění semaforu.

b) Aktualizace pozice VR nástroje podle polohy haptického zařízení.

c) Přidání řídicích bodů vnější části VR nástroje (s interpolovanou výškou terénu v dané pozici) do vyrovnávací paměti pro haptiku.

d) Interpolace výšky terénu ve středovém bodě vnitřní oblasti VR nástroje (uložení do vyrovnávací paměti).

- „Rychlé“ vlákno:

1. Pokud hlavní vlákno přistupuje do vyrovnávací paměti, čekáme na uvolnění semaforu.

2. Načtení aktuální pozice haptického zařízení. Přepočet do souřadného systému aplikace. Pokud je středový bod nástroje nad terénem, nastavíme sílu na 0N, konec iterace.

3. Interpolace odporové síly. Podle zanoření (spočítaného z rozdílu výšek) nástroje určíme odporovou sílu. Výpočet tření průměrováním sil v různých směrech.
4. Přepočet silové zpětné vazby do souřadného systému zařízení. Nastavení silové zpětné vazby do haptického zařízení.

6 Závěr

Úkolem této práce bylo dále zdokonalit dříve navrženou metodu pro modelování písčitého terénu pro virtuální realitu, která umožňuje deformovat terén pomocí virtuálních nástrojů a obsahuje simulaci eroze, a provést její porovnání s existující metodou založenou na pravidelné výškové mapě.

Z porovnání vidíme, že po vizuální stránce je zatím i přes provedená vylepšení lepší řešení využívající pravidelnou síť. Je to způsobeno zejména samotnou členitostí písčitého terénu, další příčinou je nespojité otexturování sítě (jak je zmíněno v [Sed07]). Ovšem po paměťové stránce jsme dosáhli zlepšení oproti pravidelné síti.

V aplikaci se nám podařilo použít pro ovládání virtuálního nástroje haptické zařízení od firmy SensAble typu Phantom, a to jak na Masarykově univerzitě v Brně, tak později na Západočeské univerzitě v Plzni. Silová zpětná vazba simulovaná programem navozuje uživateli dojem, jako by se dotýkal skutečného materiálu. Hmatový vjem ovšem ještě zdaleka nepřipomíná písek a na realističnosti zpětné vazby je ještě potřeba pracovat. Tento výzkum umožní propojení aplikací vyvíjených v rámci Centra počítačové grafiky a vizualizace na ZČU s haptickými zařízeními.

Je mnoho směrů, kterými se vydat při rozvoji tohoto projektu. Jako zajímavá se jeví například integrace se systémem VRECKO vyvíjeným na MU v Brně. To by mimo jiné umožnilo použít stereo stěnu pro výstup programu a snadnější použití haptických zařízení. Navíc by byly implementované techniky snáze využitelné v ostatních projektech.

Dále by bylo vhodné ještě více vylepšit realističnost celé aplikace. Toho lze dosáhnout například dalšími vylepšeními vizualizace modelu nebo důmyslnějšími virtuálními nástroji, které vytlačují materiál na základě směru pohybu a rychlosti nástroje. Výška „násypu“ (vytlačeného materiálu) by se pak řídila těmito parametry.

I když jsme v průběhu této práce nalezli a opravili řadu chyb, program i tak obsahuje některé chyby, které jsme zatím nebyli schopni odstranit. Pokud například používáme automatickou detekci redundantních vrcholů spolu s modelem eroze, dojde občas k pádu programu. To se stane jen tehdy, pokud pracují obě části zároveň. Každá zvlášť funguje bez problémů. Chyba spočívá v tom, že i když z odebíraného vrcholu odstraníme všechny sousednosti, tvořené hranami okolních trojúhelníků, občas zde jedna sousednost zbude, i když už vrchol není připojen žádnou hranou do sítě. Příčinu této chyby je třeba nalézt a odstranit.

Bylo by také vhodné implementovat důmyslnější fyzikální model, který při erodování materiálu nepřesýpá materiál na příliš dlouhé vzdálenosti a místo toho (v závislosti na vlastnostech materiálu) tyto hrany dělí, a tvoří tak realističtější srázy. Toto dělení je popsáno v [Pur08b].

Užitečné by bylo provést další podrobná měření programu [Ben06]. To je ovšem bez dokumentace velmi obtížné. Jedná se především o měření časové složitosti použitých algoritmů.

Haptická zařízení spolu s různými modely materiálů by bylo možné v budoucnu využít ve zdravotnictví, pro rehabilitaci osob s poraněním ruky. Vytvořením trénovacího programu by pak mohl pacient v podstatě sám provádět rehabilitaci s přesně daným průběhem a pravidly. Rozhraní pro haptická zařízení má využití v mnoha jiných oblastech.

Pro složitější fyzikální simulace by bylo zajímavé vyzkoušet materiál složený z různých vrstev, kde každá vrstva může obsahovat odlišný fyzikální model. Vzájemnou interakcí by tyto vrstvy mohly simulovat složité procesy probíhající nejen na povrchu, ale i uvnitř materiálu.

7 Použité pojmy a zkraty

API	Application Programming Interface
CCW	Proti směru hodinových ručiček (counter-clockwise)
CDT	Constrained Delaunay triangulation (Delaunyova triangulace s ohraničeními)
Cg	C for Graphics (jazyk pro programování GPU od firmy NVIDIA)
CHAI 3D	The Open Source Haptic Project (knihovna)
DAG	Directed Acyclic Graphs
DT	Delaunay triangulation (Delaunyova triangulace)
GCC	GCC, the GNU Compiler Collection
GLEE	OpenGL Easy Extension Library (knihovna)
GLSL	OpenGL Shading Language
GLUT	OpenGL Utility Toolkit
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HLSL	High Level Shader Language (jazyk pro programování GPU od firmy Microsoft)
MFC	Microsoft Foundation Classes
OpenGL	Open Graphics Library (knihovna)
SDL	Simple DirectMedia Layer (knihovna)
TIN	Triangulated Irregular Network (nepravidelná trojúhelníková síť)
VRECKO	Systém pro navrhování a tvorbu technik sloužících k interakci člověka s počítačem

8 Literatura a zdroje

8.1 Literatura

[Ang97] Anglada, M., V.: *An Improved Incremental Algorithm For Constructing Restricted Delaunay Triangulations*. Computers & Graphics, Vol. 21, No.2, pp. 215-223, 1997.

[Ben06] Beneš, B. - Dorjgotov, E. - Arns, L. - Bertoline, G.: *Granular material interactive manipulation: Touching sand with haptic feedback*. In Proceedings of the 14-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006, pp. 295–304, 2006.

[BKO97] de Berg, M. - van Kreveld, M. - Overmans, M. - Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag, 1997.

[Che87] Chew, L., P.: *Constrained Delaunay Triangulations*. Proceedings of the third annual symposium on Computational geometry, pp. 215–222, 1987.

[Dev98] Devillers, O.: *On Deletion in Delaunay Triangulation*, Annual Symposium on Computational Geometry, pp. 181-188, 1998.

[Fer04] Fernando, R.: *GPU Gems*. NVIDIA Corporation. One Lake Street Upper Saddle River, NJ 07458, USA. ISBN-0-321-22832-4, 2004.

[Kad07] Kadlec, J.: *Deformace terénu pro virtuální realitu – geometrická část modelu*. Bakalářská práce KIV/ZČU, 2007.

[Kol09] Kolařík, L.: *Využití existujícího modelu písčitého terénu pro modifikace jiných typů povrchů ve virtuální realitě*. Diplomová práce. Západočeská univerzita v Plzni, 2008.

[Ono03] Onoue, N. - Nishita, T.: *Virtual Sandbox*, Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG'03), pp 252–259, 2003.

[Pur07] Purchart, V.: *Deformace terénu pro virtuální realitu – datová, logická a vizualizační část modelu*. Bakalářská práce KIV/ZČU, 2007.

[Pur08a] Purchart, V. - Kolingerová, I.: *Modelování eroze a deformací na terénu pro virtuální realitu*. Studentská vědecká konference FAV/ZČU, 2008.

[Pur08b] Purchart, V.: *Využití Constrained Delaunayovy triangulace pro reprezentaci rytí do písku – etapa II*. Dokumentace k oborovému projektu. ZČU, 2008.

[Pur08] Purchart, V.: *Modelování písčitého terénu pro virtuální realitu*. ACM Student Research Competition 2008, pp. 41-48, Prague, 2008.

[Pur09] Purchart, V.: *Interactive manipulation of sand on a TIN terrain model for virtual reality*. CECSG 2009, pp 105-111. Burmerice, Slovakia, 2009.

[SBH98] Summer, R., W. - O'Brien, J., F. - Hodgins, J., K.: *Animating Sand, Mud, and Snow*, Graphics Interface '98, 1998.

[Sou08] Soukal, R.: *Aplikace algoritmu procházky v počítačové grafice*. Diplomová práce. Západočeská univerzita v Plzni, 2008.

[Sed07] Sedmihradský, J.: *Modelování eroze a deformací terénu*. Diplomová práce. Západočeská univerzita v Plzni, 2008.

[Slo93] Sloan, S.W.: *A Fast Algorithm for Generating Constrained Delaunay Triangulations*, Computers and Structures, Pergammon Press Ltd., Vol 47, No. 3, pp 441–450, 1993.

8.2 Elektronické zdroje

[Cg] *Cg - The Language for High-Performance Realtime Graphics*: http://developer.nvidia.com/page/cg_main.html

[Chai] *CHAI 3D – an open source, freely available set of C++ libraries for computer haptics, visualization and interactive real-time simulation*: www.chai3d.org

[Ghost] *Ghost SDK – haptic development libraries*: http://www.sensable.com/support-ghost-sdk.htm

[Gimp] *GIMP – The GNU Image Manipulation Program*: www.gimp.org

[GLee] *GLee – GL Easy Extension library*: http://elf-stone.com/glee.php

[Glsl] *OpenGL Shading Language*: www.opengl.org/documentation/glsl/

[Glut] *GLUT – The OpenGL Utility Toolkit*: www.opengl.org/resources/libraries/glut

[Gplot] *GNU Plot – plotting utility*: www.gnuplot.info

[Hlsl] *High Level Shading Language for DirectX*: http://msdn.microsoft.com/en-us/library/bb509561(VS.85).aspx

[OpenH] *OpenHaptics Toolkit*: www.sensable.com/products-openhaptics-toolkit.htm

[Vil09] Villar, J., R.: *OpenGL Shading Language Course – Advanced Shaders* http://www.typhoonlabs.com/tutorials/glsl/Chapter_4.pdf

[Sens09] *SensAble Technologies*: www.sensable.com

[Tis03] Tišnovský P.: *Grafická knihovna OpenGL*. Praha 2003, ISSN 1212-8309, http://www.fit.vutbr.cz/research/view_pub.php?id=7245

[Tom09] *Review Tom's Hardware: Performance Leap*, Bestofmedia Network: http://www.tomshardware.com/reviews/performance-leap,789-16.html

[Vrec] Flasar, J.: *VRECKO – Systém pro tvorbu virtuálních světů*: http://cpg.felk.cvut.cz/tiki-download_file.php?fileId=10

[Wik1] *Bump mapping – Wikipedia, free encyclopedia*: http://cs.wikipedia.org/wiki/Bump_mapping

[Wik2] *Subsurface scattering – Wikipedia, free encyclopedia*: http://en.wikipedia.org/wiki/Subsurface_scattering

[Wik3] *OpenGL (Open Graphics Library)* – Wikipedia, free encyclopedia: http://cs.wikipedia.org/wiki/OpenGL

Přílohy

Příloha A

Purchart, V.: *Interactive manipulation of sand on a TIN terrain model for virtual reality*. CECSG 2009, pp 105-111. Burmerice, Slovakia, 2009.

Interactive manipulation of sand on a TIN terrain model for virtual reality

Václav Purchart*
Center of Computer Graphics and Data Visualization,
Department of Computer Science and Engineering
University of West Bohemia
Plzen / Czech Republic

Abstract

This paper describes a new approach in modeling of sand surface. The terrain model is based on a triangulated irregular network while existing solutions have been based on a regular grid. Memory consumption is significantly reduced without a regular grid at the expense of more complicated algorithms. Our solution allows deforming terrain by a set of virtual tools and erosion simulation. All these simulations run in real time. We want to use haptic devices with our application in the future.

Keywords: Virtual Reality, Terrain Erosion Modeling, VR Tool, TIN Terrain Model

1 Introduction

Our project has the following goals. We would like to induce the same visual and touch feeling as if the user directly touches the sand in real world with the respect of the sand humidity and the thrust.

At present, we have started to work on the haptic part of this project (together with the Masaryk University in Brno, Czech Republic), however, this paper contains only the geometric, physical simulation and visualization part of our solution.

Our method aims to handle computer modeling of a terrain formed by some granular material such as sand, salt, etc. Our goal was to propose a terrain model in which it will be possible to make direct deformations using virtual tools. The terrain model responds on these and other changes by a material movement – it fills up holes and pits. Let us present a concept of our solution.

The whole terrain surface is represented by a triangulated irregular network (TIN), a triangulation. Triangle vertices describe the terrain height in the given position. The enforcement of terrain changes is done by setting vertex height or adding new vertices to the TIN. We also need to enforce some shape to the TIN because of virtual tools. It is done by constraining edges to the TIN. A tool stroke is simulated by continual enforcement of virtual tool samples.

Erosion modeling is based on a particle system principle. Material moves over the edges from one vertex to another to imitate the behavior of real sand. Depending on the characteristics of the modeled material (e.g. humidity of sand), the material is moving from the above-lying position into the low-lying positions. This causes continuous terrain smoothing.

Model based on an irregular network allows setting any necessary level of detail. The places where the terrain is flat can be described using a few triangles. By contrast it is not necessary to set a fixed level of detail for complex shapes. This approach has significantly reduced the consumption of memory needed for the terrain model. The disadvantage is that more complicated algorithms are needed than for a regular grid. The main novelty of our approach is the use of TIN instead of regular grid. Preliminary version of the algorithm has been published in [10].

2 Related work

In the area of terrain computer modeling many papers have been published. The following work deals with the modeling of granular materials, too. The existing work can be divided into two main streams – the first one focuses primarily on physical correctness. The second stream concerns interactive applications operating in real time.

In the publication [14] the terrain surface is represented as a regular grid. Grid cells represent the terrain height in each point. Working with this data structure is relatively simple and very fast. A major disadvantage, as described in [14], is the size of this data structure. The number of elements in the matrix must correspond to the smallest detail which we want to display. Memory consumption is rapidly growing with the resolution. This method does not work in real time and focuses more on creating the best possible model visualization.

Other publication dealing with the modeling of sand is [8]. It is an interactive application which simulates terrain erosion and interactive manipulation with virtual tools. The terrain is represented as a regular grid. The application contains a particle system allowing to simulate sand pouring.

By contrast, the paper [2] focused more on the use of haptical devices. These devices have force feedback and

*lipop@students.zcu.cz

induce a feeling that the user directly touches the material in the real world. Simulated feedback contains two forces: the first is the penetration resistance of the material to the tool and the second is the friction that is caused by tool movement.

3 The proposed method

In this section we describe in detail the proposed method – the manipulation of sand on a TIN terrain model. The described solution has been created within the bachelor thesis of V. Purchart [9], J. Kadlec [7] and master thesis of J. Sedmihradský [11]. The method is further improved by the author of this paper.

The model is based on a triangulated irregular network, a triangulation which represents the terrain surface. It is formed by a list of vertices coordinates, a list of triangle vertices and a list of pointers to the triangle neighbors. Each vertex *X* has real coordinates *x*, *y*, *z*, where *x* and *y* is the location in the terrain model and *z* is the height in the given location. It follows that the terrain shape must be a function of two variables $z = f(x, y)$. The model does not allow modeling of overlapping and tunnels. For sandy terrain this model is sufficient.

3.1 Geometric part

At first, we define how the model is represented exactly. Then we describe the basic operations such as adding / removing the vertex, constraining the edge, getting and setting a height in any location and finding a position of the triangle in the TIN.

A triangulation *T(P)* of a set *P* of *N* points in the Euclidean plane is a maximum set of edges *E* such that:

- no two edges in *E* intersect at a point not in *P*,
- the edges in *E* divide the convex hull of *P* into triangles.

The edges of each triangle in the TIN are required to be similarly long. Triangles should be as much as possible close to equiangular triangles. Delaunay triangulation provides such triangles. Material movement looks more realistic on such a triangulation.

The Delaunay triangulation in $\Re^2$ of the set of vertices *P* is a triangulation *DT(P)* in which each triangle defined by the vertices $A, B, C \in P$ meets *The Delaunay empty circle criterion* for each vertex *D*, where $D \in P \wedge D \notin \{A, B, C\}$ (Equation 1). Vertex *D* is not in the circle defined by the triangle *ABC*, when $d < 0$ (the criterion is met).

$$d = \begin{vmatrix} A_x - D_x & A_y - D_y & (A_x - D_x)^2 + (A_y - D_y)^2 \\ B_x - D_x & B_y - D_y & (B_x - D_x)^2 + (B_y - D_y)^2 \\ C_x - D_x & C_y - D_y & (C_x - D_x)^2 + (C_y - D_y)^2 \end{vmatrix} \quad (1)$$

There are many Delaunay triangulation algorithms. Our choice is the incremental insertion algorithm because we need to modify the completed TIN (a more detailed description see in [3]). When we add a new vertex to the TIN, the triangle which includes the new vertex must be found. We divide this triangle into new ones. We adjust the rest of the TIN by edge flipping to meet the triangulation criterion.

For further application it is necessary that the TIN allows direct enforcement of some edges. For edge constraining we use the Constrained Delaunay triangulation (*CDT*). This triangulation method is based on the basic Delaunay triangulation, but, moreover, there are "constrained edges". These edges are required in the TIN even if they do not meet the criterion of Delaunay empty circle (Equation 1). This allows enforcing direct changes into the model (see Figure 1).

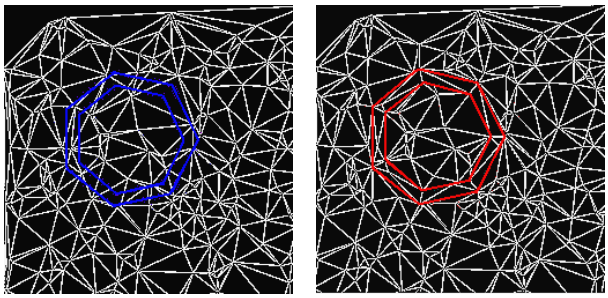


Figure 1: CDT in action. On the left we can see the proposed edges which should be constrained. On the right is the same TIN with constrained edges included.

Algorithms for the constraining of the set of edges are described in [1], [4] and [12]. Our algorithm is based on [12]. At first we construct the basic *DT* and than we constrain the required set of edges. Continuously we flip existing edges which intersect with the constrained edge. Finally we can insert the constrained edge to the TIN without an intersection with any existing edge.

Removal of vertices

Removal of vertices is necessary due to erosion, some parts of the terrain model may become flat and unnecessary vertices in such places are not needed any more (see Figure 2). They do not bring a substantial shape information but slow down the computation. Vertices removal is based on the algorithm [5]. At first, the vertex is removed from the TIN and then the resulting hole is retriangulated. So it will become again the Delaunay triangulation. After vertex removal we get a star-shape polygon in the worst case. By cutting ears of this polygon we eliminate the hole.

Unnecessary vertices can be relatively easy to detect. If the vertex is, due to its neighbor vertices, (approximately) in the plane, we do not loose any detail from the model by removing the vertex. At first we get the maximum angle between each join from the removed (center) vertex to each of its neighbor vertex (i.e. vertices which are connected by the edge) and to the horizontal line defined by the analyzed vertex. If the maximum angle is close to zero, the center vertex is redundant and can be removed from the TIN. It is impossible to check all vertices in the TIN for the criterion above, because

there can be a huge amount of vertices and the time needed for it would be prohibitive. We use a queue, which accumulates vertices that could be redundant, such as the vertices which have changed the height in the last iteration somehow. If the above criterion is met, the vertex is marked as redundant and before the completion of iteration it is removed from the TIN.

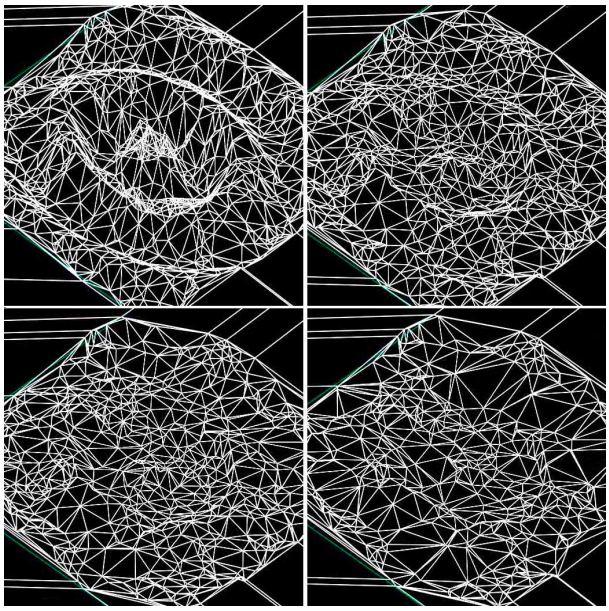


Figure 2: The automated simplification of the TIN. The top-left image shows the original TIN. The bottom-right image shows the simplified TIN which is more flat than the original due to erosion.

Point location

The most used geometric operation is a location of the triangle in which the given (target) point lies. Because of this the location of the triangle has to be effective. The TIN is permanently changing. Due to permanent changes, the use of complex search structures is not too efficient and is memory demanding. We used the algorithm of rectangular walk [13] which is suboptimal, but it does not need any special data structure, only information about neighboring triangles of each triangle.

3.2 Erosion modeling

In real life there are many types of erosion – such as water erosion, erosion caused by material temperature changes and others. The result of these processes is that the material is moved from higher-lying places to lower-lying places. The amount of the moved material depends on the height difference. We simulate this kind of erosion. The accurate physical models require more computation time. Therefore the simulation is not based on an accurate physical model because of interactivity of the whole process.

The set of vertices P of the triangulation *CDT(P)* and the set of edges E define a graph. The erosion simulation works iteratively. For each vertex $P_k \in P$ we find all vertices P_j, where the edge $E_i \in E$ from P_k exists. For each pair P_k, P_j we compute the angle. This angle is formed by a horizontal line going through P_k with an edge between P_k, P_j. If this angle is lower than the "critical angle" of the simulated material then the material transfer is realized from the vertex P_k to each neighbor vertex P_j, which lies lower than P_k (see Figure 3). For a dusty sand the critical angle is about 30°, see [11].

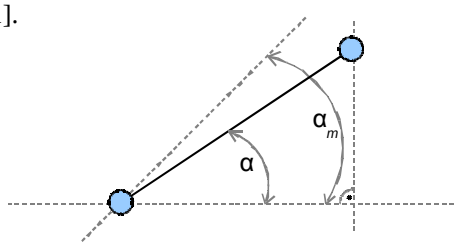


Figure 3: The material movement is stopped if an angle α is less than the critical angle α_m.

Browsing and eroding of all vertices in the TIN would be infeasible. Only the changed vertices are eroded where a material moving is possible. These vertices are saved to a special list L_e. A vertex must be added to the L_e if its height has been changed in the last iteration. The vertex height can be changed by a virtual tool or the erosion algorithm. The vertex can be removed from the L_e if its height had not been altered by erosion in a few last iterations.

Erosion methods

In the second phase of the algorithm we change the height of vertices (details in [11]). The height of the vertex is reduced and the height of some of its neighbors is increased. From the physical point of view, the essential erosion characteristic is the volume of the terrain. After the material movement the volume of the terrain should be the same as before. TIN has no "volume", but we can imagine a solid object which will be created by a terrain surface and some bottom plane.

3.3 VR deformations

Terrain deformations are done using a set of virtual tools [9]. These tools allow both simple contour constraining and pulling. VR tools are based on the *CDT* principles. We achieve a real sand deformation sensation by a managed constraining set of edges. Tool pulling is done as a periodical stamping of tool contour at fixed time intervals.

The basic VR tool consists of two parts – the set of outer edges (*O*) and the set of inner edges (*I*). The inner and outer edges cannot be at the same *(x, y)* position as we cannot represent vertical edges in our model. Therefore, the inner and outer edges are in a small mutual distance and are never completely identical in the projection, see Figure 4. By a proper constraining of both sets of edges in the TIN we achieve one tool stamping.

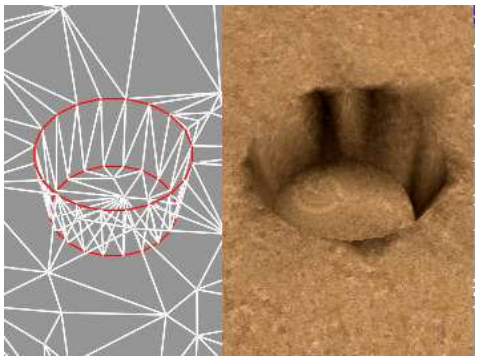


Figure 4: Stamping of a VR tool. On the left we can see the wire-frame model (red lines are constrained edges). On the right we can se the textured model. Both are without erosion; taken from [11].

The inner part of the VR tool

The purpose of the inner part of the VR tool is to deform the terrain in a required way. Every stamping by the VR tool changes the height of the terrain according to the depth level of the virtual tool. Because of this, the inner part is a simple plane in most cases. Vertices located in this area are redundant and can be deleted from the TIN. Removal of these vertices can be done by an automatic TIN simplification which is described below. This requires a little bit more computational time. We walk through all triangles inside the area and check these vertices. The vertex P_i is redundant if:

- the vertex depth is close to the tool depth level
  and
- the removed vertex does not belong to the set of border vertices
  and
- the vertex has no neighbor, whose depth is greater than the tool depth level.

The TIN can also contain edges from the previous stamping. Each set of edges has a unique number, which can determine the "generation" of each constrained edge. If a constrained edge is discovered during walking through the inner triangles, it can be deleted because it belongs to some old stamping.

The outer part of the VR tool

The outer part of the VR tool adapts to the contour of the surrounding terrain. This affects how real the stamp looks. We interpolate the height of vertices in the outer set O by a simple linear interpolation of the existing vertices. After this we constrain all edges from the set O. This gets a good result in most cases. For more realistic stamping it is necessary to adapt the outer part accurately. A good approximation is obtained by the subdivision of edges to smaller pieces which are again adapted to the height of a terrain.

It is important, in which order we constrain the parts of the VR tool. At first the outer part must be constrained into the terrain. This part adapts to the terrain. After that the inner part is constrained and makes changes "by force" in the terrain according to tool properties.

Extensions

More complex tools can be derived from the above basic tools. For a realistic appearance it is necessary to push away the material. Volume of this material should be the same as a volume of the created hole. Details of the approximate solution can be found in [11].

4 The complete algorithm

Let us first explain the meaning of the symbols in our algorithm:

L_e is the list of vertices which will be eroded.

L_r is the list of vertices used for the detection of the redundant vertices.

α_{flat} is the value of a limit angle to what extent the vertices of the nearly-flat triangles are removed from the terrain model. If α_{flat} is higher then more vertices are removed and vice versa. A proper choice of α_{flat} is addressed in Section 5.

The whole algorithm can be simply described as follows:

- Computing of the Delaunay triangulation of the initial set of vertices.
- For each iteration:
  1. Constraining deformations into the terrain on the current location of the VR tool:
     a) The height interpolation of the control points which are in the outer part of the VR tool.
     b) Constraining the set of edges of the outer tool part using *CDT*.
     c) Constraining the set of edges in the inner tool part.
     d) The height reduction in the inner VR tool part.
     e) Schedule removal of redundant vertices.
     f) Adding the changed vertices to the lists L_e and L_r.
  2. Material erosion; for each vertex P_k in the list L_e:
     a) Computation of the height difference between the vertex P_k and all its neighbors P_i (there is an edge from P_k to P_i) and choosing of the best erosion method.
     b) Computation of the material gain for the lower-lying vertices.
     c) Height reduction of the vertex P_k, height increase for the lower-lying vertices P_i.
     d) Adding changed vertices to the lists L_e and L_r,
     e) The exclusion of the vertices from L_e for which there was no material transfer.
  3. For each vertex P_l from the list L_r:
     a) The test of the angle α_i between the edge of the vertex P_l and each its neighbor P_i and a horizontal line passing through the point P_l.
        I. If $\alpha_i > \alpha_{flat}$ then removal of the vertex from the list L_r.
        II. If $\alpha_i \leq \alpha_{flat}$ then for all vertices P_i, removal of the vertex P_i from the list L_r, from the list L_e and from the TIN.

5 Experiments and results

To verify the approach with a triangulated irregular network, we have created an application in C++. In the next phase it is planned to use haptical devices, so the application must be portable to other operating systems. A model visualization is done using the OpenGL library, the graphical user interface and system communication uses the multiplatform SDL library. The application is operated on Windows XP, Windows Vista and Linux. All measured data are related to a computer with Windows Vista, 2GB RAM, Intel Core 2 Duo 1.8 GHz, NVIDIA GeForce 8600M GS, 256MB.

We experimentally set the angle α_{flat} to value 0.4 (on the basis of the experiments in [11]). This brings the best results. If the angle is lower, then the terrain model is too complex, otherwise the terrain model is close to the flat surface and a lot of details have been lost.

In Figure 5 and Figure 6, we can see a comparison of solutions using regular grid and TIN. We tried to imitate the experiment and illustration from [2] to compare visual quality of our solution and [2].

Figure 5: Our method based on a TIN

Figure 6: Result on regular grid; taken from [2].

From the figures we see that due to the long edges and probably not the optimal shading model our result in Figure 5 looks a little worse. One of the future challenges is to achieve a more realistic appearance. Possibilities are as follows:

- a better shading model,
- a subdivision of the long edges which have negative impact on shading,
- a use of randomization in the sand movement simulation to achieve less regular appearance.

At this moment, we are unable to compare with [2] as to the performance as the program obtained thanks to the authors of [6] do not work on our OS but we suppose to solve it in the near future. We at least measured the performance of our solution without comparison.

In the first experiment, we measured the computation time of the main algorithm parts dependency on the TIN size or the length of tool trace. The graph in Figure 7 shows that the vertex removal is the most time-consuming part of the whole algorithm. This time consists mainly of the Devillers vertex removal algorithm [5]. The rest of time is spent by the redundant vertices detection. The used triangle walking algorithm implies the measured complexity $O(n^{0.5})$. It is one of the future work on the project to reduce this time. Calculation of the erosion model has complexity around $O(n^{0.5})$.

In the second experiment, we measured the total time consumption in dependence on the length of tool trace. The graph in Figure 8 shows the time consumption in dependence on the length of tool trace for TIN with 10000 vertices. Complexity estimate is $O(n)$ for the erosion model and $O(n^{1/4})$ for the vertex removal. Time complexity is getting better with the size of the TIN.

In all experiments we set the fixed TIN size or the length of tool trace and measure the other parameters of the model. The trace was created along the diagonal of the modeled terrain. Time data represent the accumulated time during the whole tool pulling.

We also analyzed the number of vertices after the simulation dependency on the trace length and the triangle count affected by the VR tool dependency on the trace length (Figure 9). These dependencies are nearly linear.

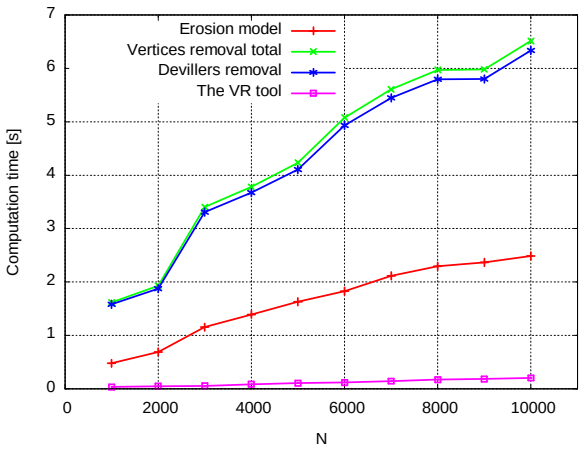


Figure 7: Time consumed by each part of computation in dependency on the TIN size (the trace length is 0.55; the whole model diagonal length is 1).

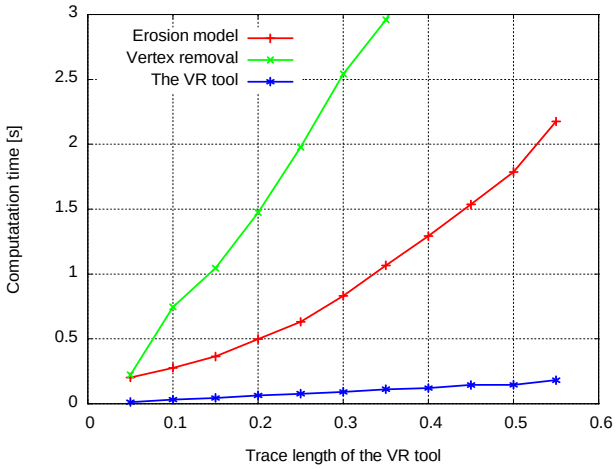


Figure 8: Time consumed by each part of computation in dependency on the trace length (the TIN size is 10000 vertices).

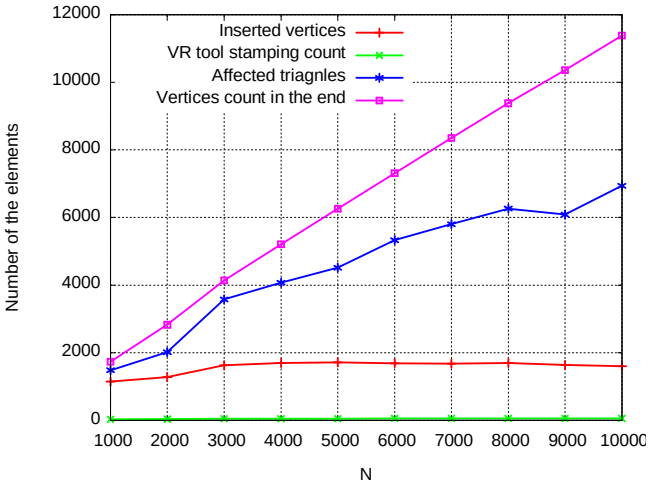


Figure 9: Number of elements in the TIN in dependency on the TIN size.

The graph in Figure 10 shows the number of elements in the TIN dependency on the trace length. Also these dependencies are nearly linear.

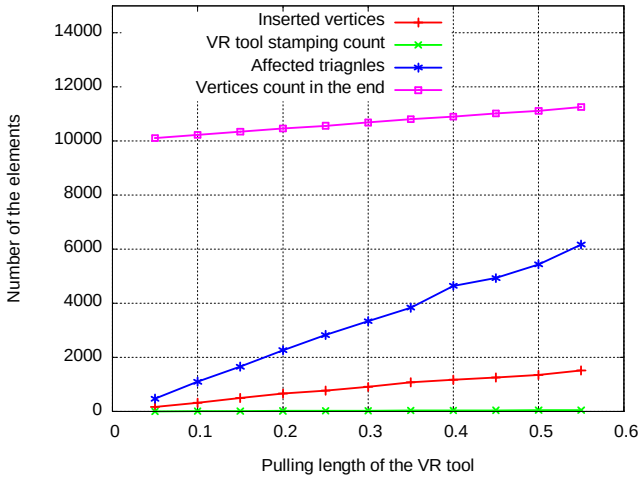


Figure 10: Number of the elements in the TIN dependency on the trace length.

6 Conclusion

Our method based on the triangulated irregular network as a terrain model can be used in practical applications. Further work will aim at the appearance improvement of the model and optimization of the vertices removal. At present we started to collaborate with Masaryk University in Brno in using haptical devices with our application. This will allow a real touching feeling of the granular material. It will be possible to create a sort of virtual sandbox. The application will be extended with more accurate physical models. There is also an effort to map the whole TIN onto 3D objects. Other possibilities would be to simulate various physical processes. The implemented erosion algorithm can be easily replaced by another physical model, which will operate on the existing geometric model of the terrain. There is a future possibility that the application would be used for e.g. visually impaired people because of simulation of touch sensation.

Acknowledgements

I would like to thank to Doc. Dr. Ing. Ivana Kolingerová (University of West Bohemia, Czech Republic) for inspiration and help with solving of this problem. Next I would like to thank to my colleagues Jan Kadlec, who created the geometric model used, and Jiří Sedmihradský, who is the author of the erosion simulation and model visualization. Both participated in the development of the first phase of the project. My thanks go also to Ing. Bedřich Beneš, Ph.D. (University of Purdue, USA) for his help with visualization and erosion modeling. I thank also to anonymous reviewers for valuable comments which enabled to improve the paper and brought inspiration for future work.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, project Kontakt No. ME09051.

References

[1] Anglada, M., V.: *An Improved Incremental Algorithm For Constructing Restricted Delaunay Triangulations*. Computers & Graphics, Vol. 21, No.2, pp. 215-223, 1997.

[2] Beneš, B., Dorjgotov, E., Arns, L., Bertoline, G.: *Granular material interactive manipulation: Touching sand with haptic feedback*. In Proceedings of the 14-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006, pp. 295–304, 2006.

[3] de Berg, M., van Kreveld, M., Overmans, M., Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag, 1997.

[4] Chew, L., P.: *Constrained Delaunay triangulations*. Proceedings of the third annual symposium on Computational geometry, pp. 215–222, 1987.

[5] Devillers, O.: *On Deletion in Delaunay Triangulation*, Annual Symposium on Computational Geometry, pp. 181-188, 1998.

[6] Enkhtuvshin, D., Beneš, B.: *Stereo Sand Drawing*. The sandy terrain simulation program based on regular grid. Received on the 10th November 2008.

[7] Kadlec, J.: *Terrain deformations for virtiual reality – geometric model part.* Bachelor thesis. University of West Bohemia, Pilsen, 2007 (in Czech).

[8] Onoue, K., Nishita, T.: *Virtual Sandbox*, 11th Pacific Conference on Computer Graphics and Applications (PG'03), pp. 252–259, 2003.

[9] Purchart, V.: *Terrain deformations for virtual reality – basic structures, control layer and model visualization*. Bachelor thesis. University of West Bohemia, Pilsen, 2007 (in Czech).

[10] Purchart, V.: *The sandy terrain modeling for virtual reality*. ACM Student Research Competition. Prague, 2008 (in Czech).

[11] Sedmihradský, J.: *Terrain erosion and deformations modeling*. Master thesis. University of West Bohemia, Pilsen, 2007 (in Czech).

[12] Sloan, S.W.: *A Fast Algorithm for Generating Constrained Delaunay Triangulations*, Computers and Structures, Pergammon Press Ltd., Vol 47, No. 3, pp. 441–450, 1993.

[13] Soukal, R.: *Application of triangle walking algorithm in computer graphics*. Master thesis. University of West Bohemia, Pilsen, 2008 (in Czech).

[14] Sumner, R. W., O'Brien, J. F., Hodgins, J. K.: *Animating Sand, Mud, and Snow*. Proceedings of Graphics Interface '98, Vancouver, B.C., Canada, June 17-21, pp. 125-132, 1998.

Příloha B

Purchart, V.: *Modelování písčitého terénu pro virtuální realitu*. ACM Student Research Competition 2008, pp 41-48. 42-50, Prague, 2008.

Modelování písčitého terénu pro virtuální realitu

Václav PURCHART*

FAV, Západočeská univerzita v Plzni,
Univerzitní 22, 306 14 Plzeň
lipop@students.zcu.cz

Abstrakt. Tento článek popisuje nový přístup k modelování písčitého terénu pomocí nepravidelné trojúhelníkové sítě. Naše řešení umožňuje deformovat terén pomocí virtuálních nástrojů a modelování eroze. Všechny tyto simulace probíhají v reálném čase.

1 Úvod

Naše metoda se zabývá počítačovým modelováním terénu tvořeného sypkými materiály, jako je například písek, sůl apod. Naším cílem bylo navrhnout model terénu, do kterého bude možné provádět cílené zásahy pomocí virtuálního nástroje. Model pak na tyto a jiné změny reaguje tak, že jako skutečný písek postupně zanáší prohlubně materiálem.

Celý povrch je popsán pomocí nepravidelné trojúhelníkové sítě. Vrcholy trojúhelníků popisují výšku terénu na dané pozici. Vynucení změn v terénu pak provádíme nastavením výšky příslušných vrcholů, popř. jejich přidáním do sítě. Požadovaného tvaru docílíme vynucením takových hran v síti, které popisují vynucovaný objekt. Tažení nástroje je simulováno postupným vynucováním otisků nástroje.

Modelování eroze pracuje na principu částicového systému. Hmota se po hranách trojúhelníků „přelévá“ z vrcholu do vrcholu tak, aby napodobila chování písku. V závislosti na vlastnostech modelovaného materiálu (např. vlhkost písku) se přesouvá materiál z výše položených míst na místa níže položená. Tím dochází k postupnému vyhlazování terénu.

Použití nepravidelné sítě umožňuje libovolně volit úroveň detailů podle potřeby. Místa, která popisují jednoduchý terén, lze popsat pomocí několika málo trojúhelníků. Naproti tomu u komplikovaných tvarů se nemusíme omezovat pevnou úrovní detailů.

---

* Vedoucí práce: Doc. Dr. Ing. Ivana Kolingerová, FAV, Katedra informatiky a výpočetní techniky, Západočeská univerzita v Plzni.

ACM Student Research Competition 2008, Prague, 2008, pp. 1-8.

Tímto přístupem se *výrazně* sníží spotřeba paměti potřebná pro vyjádření popisovaného terénu. Nevýhodou je ovšem větší časová složitost používaných algoritmů.

1.1 Existující metody

V oblasti počítačového modelování terénů bylo publikováno již mnoho článků. Následující práce se zabývaly právě modelováním sypkých materiálů. Existující práce je možné rozdělit do dvou hlavních proudů – první kladou důraz hlavně na fyzikální přesnost. Druhým proudem jsou pak interaktivní aplikace pracující v reálném čase.

V publikaci [9] je povrch terénu popsán pravidelnou čtvercovou sítí reprezentující výšku terénu v jednotlivých místech. Práce s touto datovou strukturou je poměrně snadná a velice rychlá. Zásadní nevýhodou, jak je popsáno v [9], je rozsáhlost této struktury. Počet prvků v matici totiž musí odpovídat těm nejmenším detailům, které chceme vyjádřit. Tím velmi roste spotřeba paměti. Tato metoda nepracuje v reálném čase a zaměřuje se spíše na vytvoření co nejkvalitnější vizualizace modelu.

Další publikací zabývající se modelováním písku je [4]. Jedná se o interaktivní aplikaci simulující erozi a zásahy virtuálním nástrojem do terénu reprezentovaného opět výškovou mapou na čtvercové síti. Práce obsahuje částicový systém, který umožňuje simulovat například sypání materiálu z výšky.

Naproti tomu se publikace [1] zaměřila spíše na použití haptických zařízení. To jsou zařízení se silovou zpětnou vazbou, která slouží k navození pocitu, že uživatel přímo manipuluje s materiálem v reálném světě. Simulujeme sílu skládající se ze dvou složek, první je odpor materiálu při pronikání nástroje do něj a druhou je tření, které vzniká při pohybu nástroje.

2 Navržené řešení

V této kapitole si podrobněji popíšeme navrhovanou metodu pro modelování písčitého terénu. Popisované řešení bylo vytvořeno v rámci bakalářských prací V. Purcharta [5], J. Kadlece [3] a diplomové práce J. Sedmihradského [6] a dále je zdokonalováno autorem tohoto článku.

Model je vyjádřen pomocí nepravidelné trojúhelníkové sítě, která popisuje povrch terénu. Vrcholy sítě náleží prostoru $\Re^3$, trojúhelníková síť je v prostoru $\Re^2$. Každý vrchol *X* má souřadnice *x, y, z,* kde *x* a *y* určuji pozici v terénu a *z* výšku v daném místě. Z toho vyplývá, že tvar terénu musí být vyjádřitelný jako funkce dvou proměnných $f(x, y) = z$. Model tedy neumožní modelování převisů a tunelů. Což ovšem není nijak omezující, protože písčitý terén lze v ustáleném stavu popsat takto.

2.1 Geometrická část řešení

Nejdříve přesně definujeme, jak je model reprezentován. Dále pak popíšeme základní operace, jako je přidání/odebrání vrcholu, vynucení hrany do sítě, zjištění a změna výšky v libovolném místě modelu a nalezení trojúhelníku na určené pozici.

Triangulací v prostoru $\Re^2$ neboli rovinnou triangulací na množině bodu *P*, rozumíme maximální možné rozdělení konvexní obálky množiny *P* na trojúhelníky (rovinné simplexy), které splňuje následující podmínky (převzato z [3]):

- každé dva trojúhelníky se buď neprotínají vůbec, nebo mají společný jeden bod, anebo jednu hranu,
- množina bodů *R* tvořená vrcholy všech trojúhelníků triangulace je shodná s množinou *P*.

Kvůli použitému částicovému systému je potřeba, aby hrany každého trojúhelníku v síti byly pokud možno podobně dlouhé. Tedy aby se trojúhelníky co nejvíce blížily trojúhelníkům rovnostranným. Přesouvání materiálu pak působí přirozeněji. Delaunayova triangulace omezuje vznik nám nevyhovujících trojúhelníků.

Základní Delaunayova triangulace v $\Re^2$ na množině bodů *P* je taková triangulace *DT(P)*, v níž každý trojúhelník definovaný vrcholy $A,B,C \in P$ splňuje tzv. *Delaunayovo kritérium prázdné kružnice* pro každý bod *D*, kde $D \in P \wedge D \notin \{A,B,C\}$ (vztah 1). Bod *D* neleží v kružnici opsané trojúhelníku *ABC* právě tehdy, když $d < 0$ (kritérium je splněno).

$$d = \begin{vmatrix} A_x - D_x & A_y - D_y & (A_x - D_x)^2 + (A_y - D_y)^2 \\ B_x - D_x & B_y - D_y & (B_x - D_x)^2 + (B_y - D_y)^2 \\ C_x - D_x & C_y - D_y & (C_x - D_x)^2 + (C_y - D_y)^2 \end{vmatrix} \quad (1)$$

Delaunayova triangulace je konstruována algoritmem inkrementálního vkládání bodů (podrobnější popis viz [3]). Při vložení nového vrcholu do sítě najdeme trojúhelník, který nový vrchol obsahuje, a rozdělíme ho na několik nových. Zbytek sítě upravíme prohazováním hran tak, aby vyhověl triangulačnímu kritériu.

Pro další práci je nutné, aby síť umožňovala přímo vynutit některé hrany. Pro vynucení hran používáme Delaunayovu triangulaci s omezeními (dále jen CDT). Tato triangulační metoda vychází ze základní Delaunayovy triangulace, oproti té se v CDT navíc zavádí tzv. „vynucené hrany“. Tyto hrany jsou v síti vyžadovány i v případě, že nesplňují Delaunayovo kritérium prázdné kružnice (vztah 1). Umožňují tak přímo v terénu vynutit požadované rysy.

Algoritmus pro vynucení množiny hran vychází ze [7]. Nejdříve zkonstruujeme základní DT a poté v ní vynutíme požadovanou množinu hran. Postupně prohazujeme existující hrany, které se protínají s vynucovanou hranou, až lze do sítě vložit požadovanou hranu. Tato hrana ignoruje požadavky na prohození.

Odstranění vrcholu

Odebírání existujících vrcholů ze sítě je nezbytné, protože pouhým přidáváním se model stává stále komplikovanějším a to zpomaluje probíhající výpočty. Odstraňování vrcholů využívá algoritmus [4]. Ten nejdříve odstraní požadovaný vrchol ze sítě a poté retriangularizuje vzniklou díru tak, aby vznikla opět Delaunayova triangulace. Po odstranění vrcholu vznikne v nejhorším případě hvězdicový polygon. Ten postupným ořezáváním eliminujeme.

Nepotřebné vrcholy lze poměrně snadno detekovat. Pokud vrchol leží vzhledem ke svým sousedním vrcholům (přibližně) v rovině, jeho odstraněním se síť příliš nezmění. Pokud je maximální úhel, o který se vychyluje spojnice zkoumaného vrcholu a každého jeho přímého sousedního vrcholu (tj. takových vrcholů, do kterých vede ze středového vrcholu hrana), vhledem k vodorovné přímce vycházející ze zkoumaného vrcholu, blízký nule, je vrchol nadbytečný a lze ho ze sítě odstranit. Nelze ovšem procházet všechny vrcholy sítě a kontrolovat u nich výše uvedené kritérium. Vrcholů může být velké množství. Používáme frontu, ve které se hromadí body, které by mohly být nadbytečné. Tj. body, kterým se nějakým způsobem v poslední iteraci změnila výška. Pokud je výše uvedené kritérium splněno, je vrchol označen jako redundantní a před dokončením iterace je ze sítě odstraněn.

Lokace trojúhelníku

Nejpoužívanější geometrickou operací je nalezení trojúhelníku, v němž leží zadaný (cílový) bod. Je proto nutné, aby byla co nejefektivnější. Síť se neustále mění. Použití složitých vyhledávacích struktur tedy není příliš efektivní. Námi použitý algoritmus pravoúhlé procházky [8] je suboptimální. Nepotřebuje ovšem žádné speciální datové struktury, pouze informaci o sousedních trojúhelnících každého trojúhelníku.

2.2 Modelování eroze

V přírodě existuje mnoho druhů eroze – například vodní eroze, eroze způsobená roztahováním a smršťováním materiálu při teplotních změnách a jiné. Výsledkem společného působení těchto procesů je, že se materiál přesouvá z vyšších míst na místa níže položená. Přesun je tím razantnější, čím je rozdíl výšek větší. Tímto typem eroze se budeme zabývat. Simulace neprobíhá na základě přesného fyzikálního modelu z důvodu interaktivity celého procesu.

Vrcholy P trojúhelníkové sítě CDT(P) a množina hran E tvoří graf. Simulace eroze probíhá iteračně. Pro každý vrchol $P_x \in P$ zjistíme všechny vrcholy P_j, do kterých vede z P_x hrana $E_i \in E$. Pro každou takto vzniklou dvojici spočítáme úhel, který svírá vodorovná přímka procházející P_x s hranou mezi P_x a P_j. Pokud je tento úhel větší, než tzv. „kritický úhel“ materiálu, dojde k přesunu materiálu z bodu P_x do každého sousedního bodu P_j ležícího níže než P_x. Pro suchý písek je tento úhel přibližně 30°, viz [6].

Procházení a erodování všech vrcholů sítě by bylo výpočetně neúnosné. Eroze se zúčastní jen ty vrcholy, u kterých může dojít k přesunu materiálu. Vrcholy uložíme do zvláštního seznamu *R1*. Vrchol je nutné přidat do *R1* tehdy, pokud se výška vrcholu v předchozí iteraci změnila. Změna výšky je vyvolána deformací terénu virtuálním nástrojem a také samotným procesem eroze. Vrchol ze seznamu odstraníme v případě, že u něj opakovaně nedošlo ke změně výšky působením eroze.

Metody eroze

V druhé fázi algoritmu dochází ke změně výšky vrcholů (podrobněji v [6]). Jednomu vrcholu je výška snížena a některým jeho sousedům naopak zvýšena. Podstatným

údajem z hlediska věrohodnosti průběhu eroze je právě poměr těchto hodnot. Z fyzikálního hlediska je podstatnou vlastností eroze zachovávání objemu, a naším cílem by tedy mělo být určení daného poměru výšek tak, aby „objem sítě“ před a po erozi zůstal stejný. Trojúhelníková síť samozřejmě nemá objem, ale lze si ji představit jako jednu ze „stěn“ tělesa, které vznikne, pokud z okrajů sítě spustíme svislé stěny a ty uzavřeme v určité vzdálenosti rovinou.

2.3 VR deformace

Deformace písčitého terénu provádíme sadou virtuálních nástrojů [5]. Tyto nástroje umožňují jak jednoduché otisknutí do terénu, tak tažení (rýpání). VR nástroj využívá vlastnosti CDT. Správným vynucením skupiny hran v síti lze písek deformovat tak, jako bychom do něj otiskli skutečný nástroj. Tažení nástrojem docílíme pravidelným vzorkováním těchto otisků v pevně stanovených časových intervalech.

Základní VR nástroj se skládá ze dvou částí – množiny vnějších hran (*O*) a množiny vnitřních hran (*I*). Obě části jsou od sebe nepatrně vzdáleny, aby stále platilo, že lze povrch terénu popsat jako funkci dvou proměnných. Správným vynucením obou množin hran v síti docílíme jednoho otisknutí nástroje (obrázek 2).

Vnitřní část VR nástroje

Účelem vnitřní části je deformovat povrch terénu požadovaným způsobem, tj., změnit výšku povrchu na úroveň hloubky zarytí nástroje. Tím se oblast uzavřená množinou hran *I* stává ve většině případů rovinou. Body, které jsou v této oblasti, je proto možné ze sítě odstranit. Odstraňování je také možné přenechat automatickému zjednodušování sítě, které je ale výpočetně náročnější. Procházíme všechny trojúhelníky vnitřní oblasti a kontrolujeme jejich vrcholy. Vrchol P_i je redundantní, pokud platí:

- je hloubka bodu přibližně stejná, jako hloubka razítka,
- odstraňovaný bod není bodem tvořícím hraniční oblast *I*,
- bod nesousedí s bodem, jehož hloubka je větší než hloubka vrypu nástroje.

V síti mohou být i hrany z předchozích vrypů. Každá množina vynucovaných hran má unikátní identifikátor, podle kterého lze každou jí náležící hranu identifikovat. Pokud při procházení vnitřní oblasti narazíme na vynucenou hranu, která není součásti zpracovávané množiny, ignorujeme ji, protože se jedná o starý vryp.

Vnější část VR nástroje

Vnější část se přizpůsobuje tvaru okolního terénu tak, aby okraj vrypu působil co nejpřirozeněji. Lineární interpolací zjistíme výšku terénu ve vrcholech tvořících vnější oblast *O*. Poté do sítě vynutíme všechny hrany z *O*. Tím dosáhneme ve většině případů relativně věrohodného výsledku. Pro kvalitnější otisknutí je nutné přesně přizpůsobit vnější okraj okolnímu tvaru terénu. Přibližného řešení dosáhneme rozdělením hran z *O* na menší části, které se opět přizpůsobí výšce terénu.

Je důležité, v jakém pořadí do terénu vynutíme jednotlivé části nástroje. Nejdříve je nutné otisknout vnější část, která se přizpůsobí tvaru terénu, a teprve poté vnitřní část, která „násilně“ deformuje terén podle vlastností nástroje.

Rozšíření

Ze základních nástrojů lze odvozovat libovolné jiné. Pro ještě realističtější simulaci hrabání je nutné vytlačovat materiál o objemu nástrojem vyhloubené díry do okolí vrypu. Jinak by se materiál z terénu postupně ztrácel. Podrobnosti o přibližném řešení nalezneme v [6].

3 Shrnutí

Celý algoritmus lze tedy zjednodušeně popsat následovně:

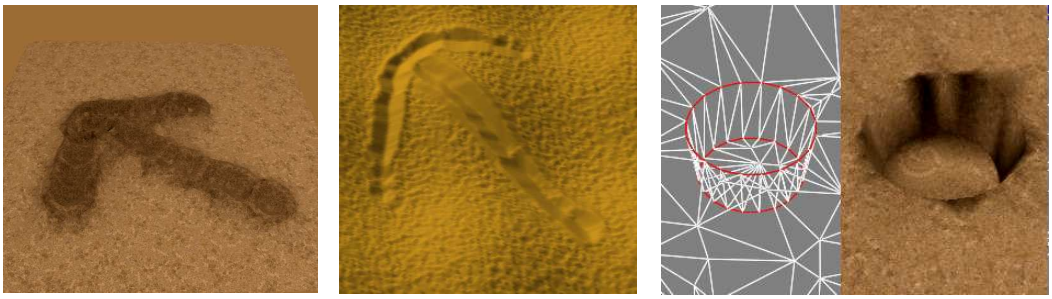
- inicializace seznamu erodovaných vrcholů *R1*, inicializace seznamu *R2* pro detekci vrcholů které je nutné odstranit, úhel ε vyjadřuje požadovanou úroveň detailů,
- výpočet Delaunayovy triangulace z počáteční množiny bodů,
- pro každou iteraci:
  1. vynucení deformací v terénu pomocí VR nástroje na aktuální pozici nástroje:
     a) interpolace výšky terénu v řídících bodech vnějšího okraje VR nástroje,
     b) vynucení skupiny hran vnějšího okraje VR nástroje pomocí CDT,
     c) vynucení skupiny hran tvořící vnitřní část VR nástroje,
     d) redukce výšky terénu ve vnitřní části VR nástroje,
     e) naplánování odstranění redundantních bodů,
     f) přidání změněných bodů do seznamu *R1* a *R2* (viz dále),
  2. eroze písku; pro každý *P* vrchol v seznamu *R1*:
     a) výpočet převýšení mezi vrcholem *P* a každým jeho sousedem P_i (z *P* do P_i vede hrana) a volba vhodné metody eroze,
     b) výpočet přírůstku materiálu pro níže položené vrcholy,
     c) redukce výšky vrcholu *P*, navýšení níže položených bodů P_i,
     d) zařazení změněných vrcholů do seznamu *R2* a *R1*,
     e) vyloučení vrcholů z *R1*, u kterých nedošlo k přesunu materiálu,
  3. pro každý *P* vrchol v seznamu *R2*:
     a) kontrola úhlu α_i mezi spojnicí E_i mezi vrcholem *P* a každým jeho sousedem P_i a vodorovnou přímkou procházející bodem *P*,
        i. pokud je pro nějaký bod P_i $\alpha_i > \varepsilon$, odstraníme bod ze seznamu *R2*,
        ii. pokud je pro všechny body P_i $\alpha_i \leq \varepsilon$, odstraníme bod ze seznamu *R2*, odstraníme bod ze seznamu *R1*, odstraníme bod ze sítě,

4 Experimenty a výsledky

Pro ověření přístupu s nepravidelnou trojúhelníkovou sítí jsme vytvořili aplikaci v jazyce C++. V další fázi je plánováno použití haptických zařízení, proto musí být celá aplikace přenositelná na jiné platformy. Vizualizace modelu je realizována prostřednictvím knihovny OpenGL, grafické uživatelské rozhraní a komunikaci se systémem obstarává muliplatformní knihovna SDL. Aplikace je provozována na operačních systémech Windows XP, Windows Vista a Linux.

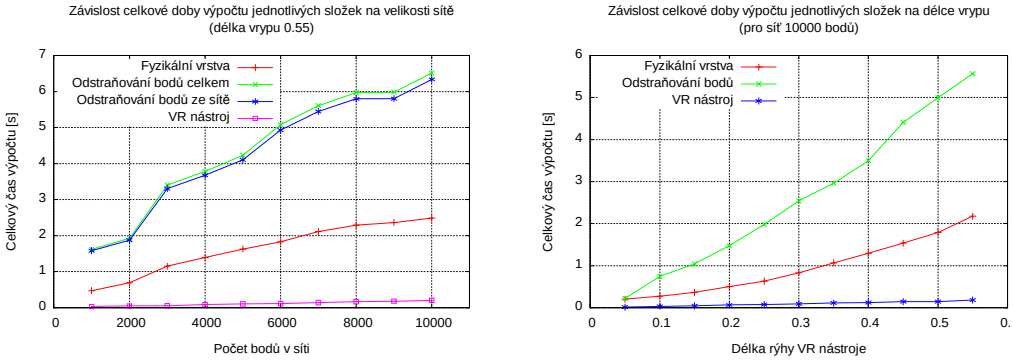
Všechna naměřená data se vztahují k počítači s Windows Vista, 2GB RAM, Intel Core 2 Duo 1,8GHz.

Na obrázku 1 vidíme porovnání řešení využívající pravidelnou a nepravidelnou síť. Z obrázku vidíme, že dlouhé hrany trochu zhoršují vzhled oproti pravidelné síti. Jedním z budoucích úkolů je dosažení realističtějšího vzhledu.



Obrázek 1: Vlevo naše metoda používající nepravidelnou síť. Uprostřed výsledek na pravidelné síti (převzato z [1]). Vpravo ukázka vrypu jednoho z používaných nástrojů (převzato z [6]). Model je bez eroze. V první části je drátěný model (červeně jsou vynucené hrany), v druhé je otexturovaný model.

Graf vlevo na obrázku 2 ukazuje, že doba odstraňování vrcholů je časově nejnáročnější. Z dílčích časů odstraňování převládá doba na fyzické odstranění redundantních vrcholů ze sítě. Naměřená složitost $O(n^{0.5})$ ukazuje použití algoritmu procházky, budeme se snažit o zrychlení této části. Doba výpočtu fyzikálního modelu má složitost přibližně $O(n^{0.5})$. Provedli jsme také měření závislosti doby výpočtu na délce rýhy. Graf vpravo na obrázku 2 ukazuje časové závislosti na délce rýhy nástroje pro síť o velikosti 10 000 bodů. Odhad složitosti je $O(n)$ pro fyzikální model a $O(n^{4/3})$ pro odstraňování bodů. S růstem velikosti sítě se složitost zlepšuje.



Obrázek 2: Vlevo závislost celkové doby výpočtu na počtu bodů sítě. Vpravo doba výpočtu v závislosti na délce rýhy pro síť s 10 000 body.

Studovali jsme také závislosti konečného počtu bodů v síti a počet nástrojem ovlivněných trojúhelníků na délce rýhy. Tyto závislosti jsou přibližně lineární.

5 Závěr

Naše metoda používající pro reprezentaci terénu nepravidelnou trojúhelníkovou síť je v praxi použitelná. Další práci budeme směrovat na zlepšení vzhledu a urychlení

odstraňování vrcholů. V blízké době se počítá ve spolupráci s MU v Brně s propojením aplikace s haptickými zařízeními, což umožní reálný vjem doteku sypkého materiálu a bude tak možné simulovat jakési virtuální pískoviště. Aplikace bude rozšířena o kvalitnější fyzikální modely a automatické vyhlazovaní terénu. Díky haptickým zařízením bude možné využít aplikaci například v lékařství pro rehabilitaci osob po vážných zraněních ruky, protože správným nastavením parametrů modelu je možné simulovat v podstatě libovolný materiál. Další využití by mohlo být pro simulaci různých fyzikálních procesů. Implementovanou erozi lze snadno nahradit jiným fyzikálním procesem, který bude operovat na stávajícím geometrickém modelu terénu.

Poděkování

Rád bych poděkoval především Doc. Dr. Ing. Ivaně Kolingerové (Západočeská univerzita v Plzni) za její nápady a podporu při řešení tohoto problému, dále Janu Kadlecovi, který vytvořil používaný geometrický model, a Jiřímu Sedmihradskému, který je autorem simulace eroze a vizualizace celého modelu. Ti se velkou částí podíleli na vývoji první fáze projektu. Můj dík patří také Ing. Bedřichu Benešovi, Ph.D. (University of Purdue, USA) za jeho pomoc s vizualizací a modelováním eroze.

Tato práce byla podporována z projektu MŠMT LC06008.

Reference

[1] Beneš, B., Dorjgotov, E., Arns, L., Bertoline, G.: *Granular material interactive manipulation: Touching sand with haptic feedback*. In Proceedings of the 14-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006, pp. 295–304, 2006.

[2] Devillers, O.: *On Deletion in Delaunay Triangulation*, Symposium on Computational Geometry, pp 181-188, 1998.

[3] Kadlec, J.: *Deformace terénu pro virtuální realitu – geometrická část modelu*. Bakalářská práce 2007. Západočeská univerzita v Plzni.

[4] Onoue, K., Nishita, T.: *Virtual Sandbox*, 11th Pacific Conference on Computer Graphics and Applications (PG'03), pp 252–259, 2003.

[5] Purchart, V.: *Deformace terénu pro virtuální realitu – datová, logická a vizualizační část modelu*. Bakalářská práce 2007. Západočeská univerzita v Plzni.

[6] Sedmihradský, J.: *Modelování eroze a deformací terénu*. Diplomová práce 2007. Západočeská univerzita v Plzni.

[7] Sloan, S.W.: *A Fast Algorithm for Generating Constrained Delaunay Triangulations*, Computers and Structures, Pergammon Press Ltd., Vol 47, No. 3, pp 441–450, 1993.

[8] Soukal, R.: *Aplikace algoritmu procházky v počítačové grafice*. Diplomová práce 2008. Západočeská univerzita v Plzni.

[9] Sumner, R.W., O'Brien, J.F., Hodgins J.K.: *Animating Sand, Mud, and Snow*. Graphics Interface '98, pp 125-132, 1998.

Příloha C

Purchart, V. - Kolingerová, I.: *Modelování eroze a deformací na terénu pro virtuální realitu*. Studentská vědecká konference FAV/ZČU, 2008.

Studentská Vědecká Konference 2008

MODELOVÁNÍ EROZE A DEFORMACÍ NA TERÉNU PRO VIRTUÁLNÍ REALITU

Václav Purchart[1], Ivana Kolingerová[2]

1 ÚVOD

Pro počítačové modelování terénu ve 3D se používají nejčastěji povrchové modely. To jsou takové modely, které popisují pouze povrch terénu a nezabývají se tím, co se nachází pod ním.

Existují dva základní přístupy pro vyjádření tvaru povrchu terénu. Prvním jsou plně 3D modely, které umožnují modelovat libovolný obecně konkávní objekt. Práce s nimi je implementačně i výpočetně velice náročná. Druhým přístupem je popsat terén pomocí výškové mapy. Toto zjednodušení předpokládá, že lze povrch terénu popsat pomocí funkce dvou proměnných.

V praxi se pro modelování terénu používají většinou pravidelné čtvercové sítě, které dobře aproximují výškovou mapu. Každá buňka uchovává diskretizovanou hodnotu výšky terénu konkrétní oblasti. Práce s touto strukturou je rychlá a jednoduchá. Pravidelná čtvercová síť má ale i několik nevýhod – pro zvýšení detailů musíme zvětšit rozlišení celé vzorkovací matice. Platí, že celkový počet bodů v síti je kvadraticky závislý na jednotce vzdálenosti. Tím *velmi* rostou nároky na paměť.

Opačným přístupem je použití nepravidelné (například trojúhelníkové) sítě. Vrcholy sítě nesou informaci o výšce terénu v daném místě. Výšku ostatních částí snadno zjistíme interpolací výšky sousedních buněk. Tento přístup nám umožní škálovat úroveň detailů v různých místech terénu podle potřeby. Nezajímavé části lze vyjádřit pomocí několika málo trojúhelníků. Složité útvary pak můžeme libovolně zjemňovat aniž bychom ovlivnili ostatní části krajiny. Cenou za to je vyšší složitost používaných algoritmů. Dosáhneme ovšem výrazně nižších nároků na paměť.

Naše řešení využívá nepravidelné trojúhelníkové sítě v kombinaci s modelem eroze založeném na částicovém systému. Řešení bylo vyvinuto a implementováno studenty ZČU V. Purcharem, J. Kadlecem a J. Sedmihradským v rámci bakalářských a diplomové práce ve spolubráci s Ing. Bedřichem Benešem, Ph.D. z University of Purdue, USA.

2 EROZE

V přírodě existuje mnoho druhů eroze – například vodní eroze, eroze způsobená roztahováním a smršťováním materiálu při teplotních změnách a jiné. Výsledkem společného působení těchto procesů je, že se materiál přesouvá z vyšších míst na místa níže položená. Přesun je tím razantnější, čím je rozdíl výšek větší. Tímto typem eroze se budeme zabývat.

Modelování eroze pracuje na principu částicového systému. Hmota se po hranách trojúhelníků „přelévá“ z jednoho vrcholu do druhého. Pro tento účel by bylo nejlepší, kdyby se síť skládala jen z rovnostranných trojúhelníků. To ovšem v nepravidelné síti není možné

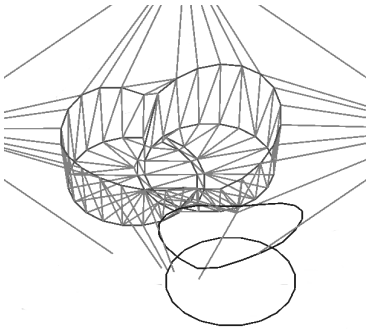
[1] Václav Purchart, student navazujícího programu studijního programu Aplikované vědy a informatika, obor Počítačová grafika a výpočetní systémy, e-mail: lipop@students.zcu.cz

[2] Doc. Dr. Ing. Ivana Kolingerová, ZČU v Plzni, FAV, Katedra informatiky a výpočetní techniky, Univerzitní 22, 306 14 Plzeň, tel.: +420 377632433, email: kolinger@kiv.zcu.cz (vedoucí práce)

splnit. Proto se snažíme, aby se trojúhelníky v síti co nejvíce rovnostranným blížily. Kritérium Delaunayovy triangulace v síti vytváří právě takové trojúhelníky. Modelovím eroze se podrobně zabývá J. Sedmihradský ve své diplomové práci (2007).

3 VIRTUÁLNÍ NÁSTROJ

Pro zajištění interaktivity s uživatelem obsahuje program ještě jakýsi virtuální nástroj, který do terénu přímo vynucuje změny za běhu programu a umožňuje tak simulovat například hrabání (viz obr. 1). Na nástrojem deformovný terén pak opět působí eroze. Změny v terénu provádíme vynucováním hran v trojúhelníkové síti. Pro vynucování hran je použit algoritmus CDT popsaný J. Kadlecem (2007).



Obr. 1: Otisk virtuálního nástroje

Virtuální nástroj se skládá minimálně ze dvou částí – vnějšího a vnitřního okraje. Vnitřní část deformuje terén podle požadavků uživatele a vnější část tvoří rozhraní mezi otiskem nástroje a zbykem terénu. Obě části tvoří uzavřenou oblast. Body obsažené ve vnitřní části je možné ze sítě odstranit – stejně jako všechny ostatní, které leží v rovině. Tím se síť postupně zjednodušuje na místech, kde je to možné. Simulace virtuálního nástroje je popsána V. Purchartem (2007).

4 ZÁVĚR

Aplikace má výrazně nižší spotřebu paměti než řešení využívající pravidelné sítě. V budoucnu se počítá s využitím aplikace na haptických zařízeních, které ještě více zlepší vjem hrabání, a bude tak možné simulovat jakési virtuální pískoviště.

Další využití by mohlo být pro simulaci různých fyzikálních procesů. Implementovnou erozi lze snadno nahradit jiným fyzikálním procesem, který bude operovat na stávajícím geometrickém modelu terénu.

Podle výsledků experimentů a odezvy zahraničních partnerů se očekává další rozšiřování projektu v rámci oborových a diplomových prací. Animace ukazující výsledky práce byly prezentovány I. Kolingerovou v rámci přednášky o triangulacích na konferenci CESCG 2007 na Slovensku.

5 LITERATURA

Kadlec, J., 2007. Deformace terénu pro virtuální realitu – geometrická část modelu. Bakalářská práce. Západočeská univerzita v Plzni.

Purchart, V., 2007. Deformace terénu pro virtuální realitu – datová, logická a vizualizační část modelu. Bakalářská práce. Západočeská univerzita v Plzni.

Sedmihradský, J., 2007. Modelování eroze a deformací terénu. Diplomová práce. Západočeská univerzita v Plzni.