VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

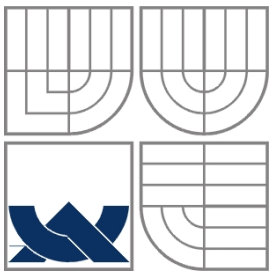
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DYNAMICKÉ ZMĚNY V TERÉNU

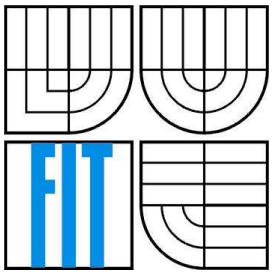
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. RADIM DVOŘÁK
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DYNAMICKÉ ZMĚNY V TERÉNU

DYNAMIC CHANGES IN THE TERRAIN

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADIM DVOŘÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN DRAHANSKÝ, Ph.D.

BRNO 2007

Abstrakt

Tato práce se zabývá návrhem, realizací a analýzou modelu pro dynamickou deformaci terénu. Je popsán současný stav problematiky dynamických změn terénu v prostředí OpenSceneGraph a je uveden dostupný relevantní software TDS, který umožňuje adaptovat terén podle nově přidaných objektů. Zvláštní důraz je kladen na návrh modelu pro fyzikální deformace terénu, které jsou způsobeny pohybujícím se tělesem nebo explozí výbušniny. Výsledky simulačních testů jsou prezentovány a na základě analýzy modelu jsou navrženy a realizovány optimalizace, které mnohonásobně zefektivňují výsledný algoritmus.

Klíčová slova

Deformace terénu, fyzikální simulace, deformovatelný model, OpenSceneGraph

Abstract

This thesis deals with design, implementation and analysis of the model for dynamic changes in the terrain. Present state of terrain deformation in OpenSceneGraph environment is described and available relevant software called TDS, which allows terrain adaptation to new inserted objects is presented. Special emphasis is placed on design of model for physically based terrain deformations that are caused by moving object or by bomb explosion. The results of simulation tests are presented and on the base of model analysis, the optimizations, which significantly improve final algorithm, are designed and realized.

Keywords

Terrain deformations, physically based simulation, deformable model, OpenSceneGraph

Citace

Radim Dvořák: Dynamické změny v terénu, diplomová práce, Brno, FIT VUT v Brně, 2007

Dynamické změny v terénu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing., Dipl.-Ing. Martina Drahanského, Ph.D.
Další informace mi poskytl Václav Bílek ze společnosti E-COM s.r.o.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radim Dvořák
22.5.2007

Poděkování

Rád bych poděkoval svému vedoucímu, Ing., Dipl.-Ing. Martinu Drahanskému, Ph.D., za četné konzultace a pomoc při tvorbě diplomové práce. Mé poděkování patří také Václavu Bílkovi ze zadavatelské firmy E-COM s.r.o. za ochotu a poskytování informací důležitých pro zahájení a vývoj projektu.

Obsah

Obsah 1
1 Úvod 3
2 Reprezentace scény 5
2.1 Graf scény 5
2.2 OpenSceneGraph 7
2.2.1 Architektura 7
3 Deformovatelný terén 9
3.1 Projekt „Terrain Deformation Software“ 9
4 Fyzikálně deformovatelné modely 11
4.1 Fyzikální pozadí 11
4.2 Spojité modely 12
4.3 Částicové systémy 13
4.4 Plynoucí částice a bez-síťové metody 14
4.5 Volně vázané částicové systémy 14
5 Numerická integrace 17
5.1 Eulerova metoda 17
5.2 Runge-Kuttova metoda druhého řádu 18
5.3 Runge-Kuttova metoda čtvrtého řádu 19
5.4 Integrační metoda „leapfrog“ 19
5.5 Adaptivní velikost integračního kroku 20
6 Návrh 21
6.1 Fyzikálně deformovatelný terén 21
6.2 Detekce kolizí 22
6.2.1 Hierarchické dělení prostoru 22
6.2.2 Prořezání obdélníků (box pruning) 24
6.2.3 Detekce kolizí s terénem 24
6.3 Deformace pohybem po terénu 25
6.4 Deformace terénu explozí výbušniny 26
6.5 Optimalizace částicového systému 28
7 Realizace 29
7.1 Integrace do toolkitu OpenSceneGraph 29
7.2 Speciální datové struktury 30

7.3 Algoritmus 32
8 Výsledky a analýzy 34
8.1 Časová analýza algoritmu 37
8.2 Efektivita struktury kd-tree 38
8.3 Optimalizace algoritmu 40
9 Závěr 43
Literatura 45

1 Úvod

Dynamické změny v terénu spadají v oblasti počítačové vědy do mnoha kategorií. Jmenovitě to je počítačová grafika, modelování a simulace, které využívají poznatky z fyziky. Z hlediska počítačové grafiky je to zejména způsob reprezentace a vizualizace samotného terénu a schopnost dynamické změny této reprezentace podle aktuálních potřeb. Z pohledu fyzikálního modelování je terén vyjádřen jako model, který je zjednodušenou napodobeninou skutečnosti a má určité fyzikální vlastnosti, které jej charakterizují. Simulace potom znázorňuje chování modelu v čase.

Dynamiku terénu nebo jiného grafického objektu lze čistě z hlediska počítačové grafiky chápat jako jeho vytváření či úpravu, což je řízeno vnějším systémem nebo přímo člověkem. Pro snadnou tvorbu nebo změnu charakteru grafického objektu je vhodné mít k dispozici patřičné nástroje, které tuto činnost usnadní nebo do jisté míry zautomatizují. Tyto nástroje zefektivňují práci a manipulaci s grafickým objektem a umožňují jej podle požadavků dynamicky měnit.

Terén jako fyzikální model lze dynamicky měnit podle aktuálního působení vnějších vlivů. Mezi tyto vlivy patří zejména působení sil na terén, pod jejichž vlivem se mění jeho vlastnosti. Tyto síly mohou být přirozeného nebo i umělého charakteru. Mezi přirozené síly lze počítat např. gravitaci, odpor prostředí nebo třecí síly. Umělé síly jsou vyvolány vnějšími zdroji, které se během procesu simulace náhle objevují a zanikají. Jejich účel může být například manipulace či úprava již existujícího modelu na fyzikální bázi.

O tato fakta se opírá také samotný cíl projektu, kterým je pomocí dostupné literatury navrhnout model pro dynamické změny (deformace) v terénu, který splňuje stanovené požadavky a který bude pracovat nad toolkitem OpenSceneGraph. Součástí řešení je také zjištění, otestování a prezentace současných možností, které OpenSceneGraph v této oblasti poskytuje.

Prvním požadavkem je možnost úpravy terénu přidáváním nových objektů, tzn. upravit terén podle jejich podstavy (footprintu) včetně úpravy terénu podle definované křivky. Druhým požadavkem je schopnost modelovat deformace terénu na fyzikální bázi, tzn. pod působením určitých sil. Tyto dva požadavky jsou doplněny nutností provádět změny, v případě úpravy terénu interaktivně a v případě fyzikálních deformací v reálném čase (real-time). Samotný návrh a implementace modelu je provedena s ohledem na prostředí OpenSceneGraph.

Problematikou vizualizace scény a její reprezentace v souvislosti s prostředím OpenSceneGraph se zabývá kapitola 2, Reprezentace scény, ve které je obecně popsané, jak se v počítačové grafice reprezentuje grafická scéna a je uvedena charakteristika grafického toolkitu OpenSceneGraph, který takovouto reprezentaci scény obsahuje, a jehož bude tento projekt využívat.

V kapitole 3, Deformovatelný terén, je uveden stručný úvod do oblasti deformace terénu a dále v jeho podkapitole 3.1, Projekt „Terrain Deformation Software“, je uveden současný stav možností deformace terénu v prostředí OpenSceneGraph v relevantním projektu, jež jsou založené na nefyzikálním principu.

Kapitola 4, Fyzikálně deformovatelné modely, se zabývá problematikou a současným stavem tvorby fyzikálně deformovatelných modelů. Konkrétně kapitola 4.1, Fyzikální pozadí, obsahuje stručný úvod týkající se fyziky deformace těles. V následujících kapitolách jsou popsány hlavní přístupy pro simulaci deformací v počítačové grafice. Jsou to spojité modely popsané v kapitole 4.2,

částicové systémy popsané v kapitole 4.3, plynoucí částice a bez-síťové metody popsané v kapitole 4.4 a volně vázané částicové systémy popsané v kapitole 4.5. Ke všem metodám je v každé kapitole uvedeno stručné hodnocení podle jejich možností a schopností.

V téměř každé fyzikální simulaci je třeba řešit lineární či nelineární diferenciální rovnici, ovšem ne vždy je snadné nebo vůbec možné použít analytické řešení a je nutné ji řešit numericky. Této problematice je věnována kapitola 5, Numerická integrace, kde jsou v podkapitolách 5.1, Eulerova metoda, 5.2, Runge-Kuttova metoda druhého řádu, 5.3, Runge-Kuttova metoda čtvrtého řádu, a 5.4, Integrační metoda „leapfrog“, popsány jmenované často používané integrační metody, vysvětleny jejich výhody a nevýhody. V poslední podkapitole 5.5, Adaptivní velikost integračního kroku, je popsáno, jak dynamicky měnit velikost integračního kroku při numerické integraci podle aktuální potřeby v průběhu simulace.

V kapitole 6, Návrh, je pak navržen způsob, jak modelovat a simulovat deformovatelný terén reprezentovaný sítí trojúhelníků. Konkrétně to je v podkapitole 6.1, Fyzikálně deformovatelný terén, způsob reprezentace fyzikálně deformovatelného terénu se zřetelem na požadavky projektu. V podkapitole 6.2, Detekce kolizí, a je navržen efektivní způsob detekce kolizí mezi tělesy a terénem. V následujících dvou podkapitolách 6.3, Deformace pohybem po terénu, a 6.4, Deformace terénu explozí výbušniny, jsou navrženy způsoby jmenovaných typů deformací. V poslední podkapitole 6.5, Optimalizace částicového systému, je pak uveden návrh, jak optimalizovat aktualizaci částicového systému.

V následující kapitole 7, Realizace, a jejích podkapitolách je popsán způsob implementace navrženého simulátoru, jeho začlenění do toolkitu OpenSceneGraph a použité speciální datové struktury.

V kapitole 8, Výsledky a analýzy, jsou prezentovány výsledky projektu jak pro deformaci terénu pohybem tělesa, tak pro deformaci terénu způsobenou explozí výbušniny. V návaznosti na zjištěnou skutečnost, že algoritmus je možné mnohokráte urychlit, byl proveden časový rozbor algoritmu, jenž je uveden v podkapitole 8.1, Časová analýza algoritmu. Dále byla z návrhu optimalizace částicového systému změřena efektivita použití struktury kd-tree pro pohybující se částicový systém, což uvádí podkapitola 8.2, Efektivita struktury kd-tree. V poslední podkapitole 8.3, Optimalizace algoritmu, je pak navržena a vyhodnocena celková optimalizace algoritmu vycházející ze skutečností z předchozích podkapitol.

V poslední kapitole 9, Závěr, je obsaženo celkové zhodnocení projektu a uveden možný způsob jeho rozšíření do budoucna.

2 Reprezentace scény

Scénou se rozumí množina prostorových objektů doplněná dalšími informacemi důležitými pro jejich zobrazení. Mezi tyto informace patří zejména transformace jednotlivých těles, tj. zaznamenání především orientace, posunutí a změny měřítka těles ve scéně, dále definice osvětlení, definice kamery, vytváření skupin těles apod.

Transformace ve scéně by měly být strukturovaně definované, aby bylo možné snadno manipulovat s jednotlivými tělesy.

Popis osvětlení scény se skládá z definic světelných zdrojů ve scéně. Tyto světelné zdroje mohou být definovány obecně jako bodová světla, kdy se světlo šíří všemi směry nebo jen určitou oblastí, nebo jako plošná světla, která jsou definována vyzařovací plochou. Světelné zdroje jsou virtuálními tělesy ve scéně a nemají obecně svou geometrickou reprezentaci.

Kamera ve scéně je definována jako pohledová transformace, podle které autor scény ví, odkud se dívá pozorovatel, jaký je jeho směr pohledu a jaké použít promítání scény. Ve scéně může být definováno více kamer, mezi kterými se lze určitým způsobem přepínat, a které mohou mít odlišné vlastnosti.

Definicí skupin těles se rozumí tvorba logicky oddělených bloků objektů scény, které mají podobné vlastnosti. Skupiny těles jsou důležité zejména pro zpřehlednění reprezentace scény a pro zvýšení efektivity při práci s jednotlivými tělesy.

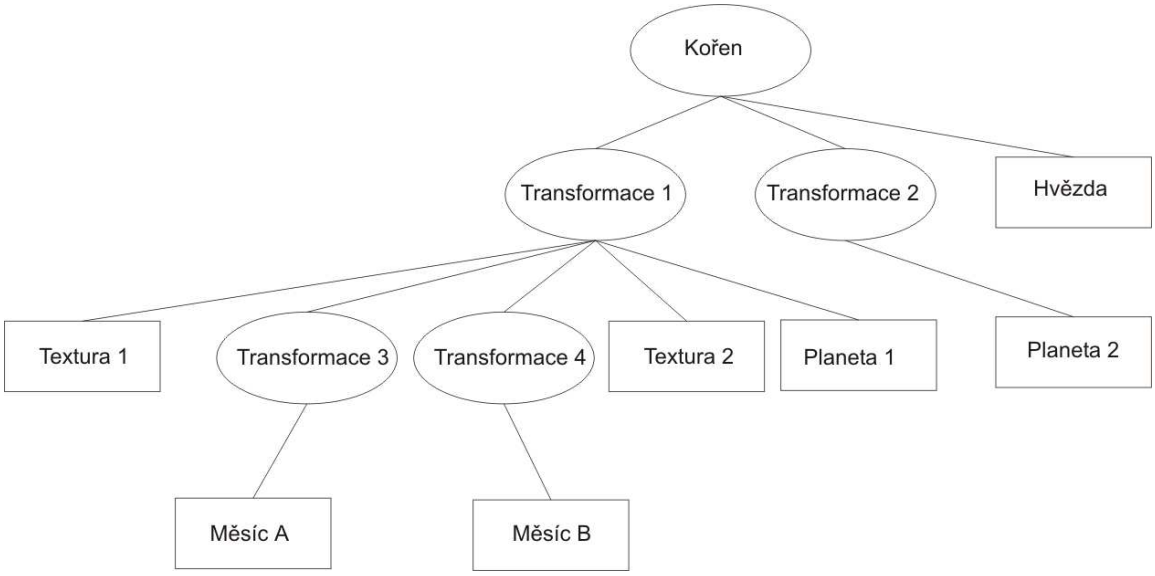
Všechny výše popsané vlastnosti scény vedou k použití efektivní struktury, grafu scény, která umožňuje efektivní práci s transformacemi, efektivní zobrazování scény, přehledný popis těles, vkládání a odebírání těles a objektů ze scény a logické a hierarchické dělení scény [Zar04].

2.1 Graf scény

Graf scény je acyklický n-ární strom, tj. graf neobsahující cykly, kde každý uzel, vyjma kořene, stromu má právě jednoho předchůdce. Graf scény je tedy hierarchická struktura dělící objekty scény do prostorových skupin, obsahující veškeré informace spojené se scénou ve svých uzlech a definující vztahy mezi jednotlivými objekty scény.

Nelistové uzly grafu obsahují informace jako je transformace, které jsou platné pro celý podstrom daného uzlu. Listy grafu obsahují informace o grafických tělesech, jejich geometrii, barvě, textuře apod.

Na Obr. 1 je znázorněn příklad, jak může jednoduchý graf scény vypadat.



Obr. 1: Příklad grafu scény

Vnitřní uzly stromu označené elipsou určují transformace svých podstromů. Listy stromu označují buď grafická tělesa, nebo ve výše uvedeném grafu také textury. Interpretace zobrazeného grafu muže být následující.

Na všechny grafické objekty jsou aplikované předcházející transformace směrem od kořene. Například pro měsíc A jsou aplikovány transformace 1 a následně 3. Grafické objekty budou mít texturu, která je umístěna vlevo od nich a je zároveň nejpravější v dané hierarchii. To znamená například, že měsíc B bude mít texturu 1 a planeta 1 bude mít texturu 2. Hvězda a planeta 2 budou mít nastavené implicitní textury pro hvězdu a pro planetu.

Mezi hlavní příčiny toho, proč je graf scény jedním z nejpoužívanějších reprezentací scény v počítačové grafice jsou podle [Sce06] tyto:

Výkon (performance)

Graf scény je vhodný pro operace typu prořezávání stromu, kdy se odstraní ze zpracování ty části stromu, které nejsou vidět danou kamerou.

Důležité z hlediska výkonu je také setřídění stavů vlastností objektů, tj. seskupování objektů, které mají stejné či podobné vlastnosti a vykreslování těchto objektů v dané skupině bezprostředně po sobě. Toto má za následek menší zatížení zejména sběrnice, kdy se posílá do grafické karty méně dat. Této vlastnosti lze pomocí grafu scény dosáhnout díky stromové struktuře poměrně snadno.

Produktivita (productivity)

Grafy scény pracují nad grafickým rozhraním (např. OpenGL) a šetří psaní nízko-úrovňového kódu týkajícího se daného grafického aplikačního programového rozhraní (angl. API).

Navíc při použití objektově orientovaného paradigmatu ve spojení se strukturou stromového grafu je dosaženo velké efektivity v psaní a rozšiřování programu.

Přenositelnost (portability)

Jestliže graf scény pracuje nad OpenGL, je jeho přenositelnost často pouze otázkou překompilování konkrétního programu.

Rozšiřitelnost (scalability)

Dnešní moderní grafy scény obsahují vykreslující framework, který se stará o efektivní vykreslení scény na různých hardwarových konfiguracích s jedním či více procesorů (CPU), s jednou či více vykreslovacími stanicemi či grafickými jednotkami (GPU). Uživatel či vývojář se tak může soustředit zejména na vlastní práci spojenou s vývojem konkrétní aplikace pracující nad grafem scény.

2.2 OpenSceneGraph

Jedním ze současných nejvíce rozšířených grafů reprezentujících scénu je OpenSceneGraph. OpenSceneGraph je (převzato z [Int06]) otevřený softwarový (open source) grafický toolkit, který umožňuje vývoj vysoce výkonných grafických aplikací, jako jsou letecké simulátory, hry, virtuální realita či vizualizace ve vědě.

OpenSceneGraph je napsán ve standardu C++, maximálně využívá knihovnu STL a jeho jednotlivé části jsou navrženy pomocí návrhových vzorů. Poskytuje objektově orientovaný systém postavený nad OpenGL osvobozující vývojáře od implementace a optimalizace nízko-úrovňových grafických volání a umožňuje skrze mnoho dalších utilit rychlý vývoj grafických aplikací. Díky těmto vlastnostem se jedná o vysoce produktivní systém.

Z hlediska výkonu obsahuje OpenSceneGraph prvky popsané v kapitole 2.1 Graf scény a navíc umožňuje například urychlování vykreslování pomocí Level of Detail, využívá vertexová pole či display listy.

OpenSceneGraph se začal vyvíjet na platformě IRIX, následně byl přenesen na Linux, pak na Windows, FreeBSD, Mac OSX, Solaris, HP-UX a dokonce na PlayStation2. Přenositelnost, jako jeden z požadavků na graf scény, je tedy do značné míry splněna.

OpenSceneGraph lze také provozovat na rozličných hardwarových konfiguracích, jako je více CPU s více grafickými subsystémy. Více informací o tomto tématu lze najít v [Int06].

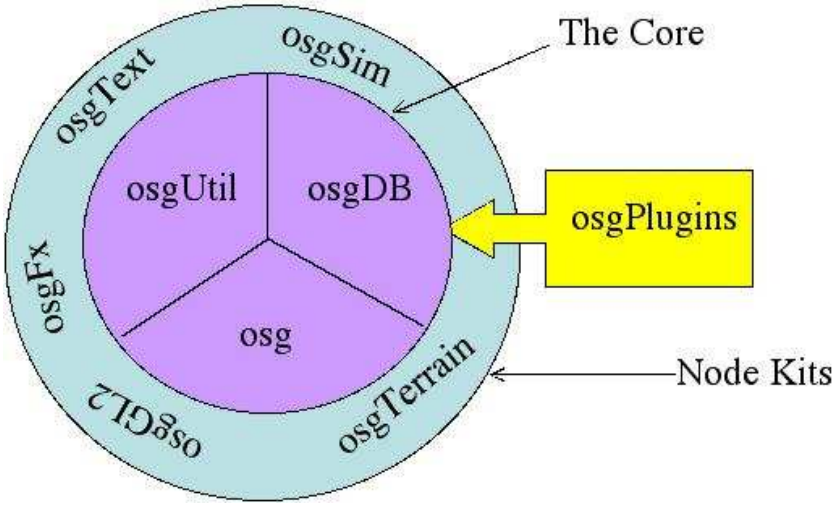
2.2.1 Architektura

OpenSceneGraph se skládá ze tří jádrových modulů, implementovaných jako dynamické knihovny: osg, osgUtil a osgDB (viz Obr. 2).

Modul osg obsahuje vlastní graf scény a s ním spojené grafické geometrie objektů, dále stavové proměnné OpenGL, funkce s podporou lineární algebry, jako je práce s vektory, maticemi a quaterniony a další podpůrné funkce. Jedná se o pasivní sadu struktur a algoritmů pro přímou práci s geometriemi objektů a stavovými proměnnými.

Modul osgUtil poskytuje operace pro práci s vlastním grafem scény. Mezi tyto operace patří zejména správa grafu a jeho aktualizace, optimalizace, prořezávání grafu, tesalace a jiné speciální funkce.

Modul osgDB obsahuje registr pluginů, abstrakci pro čtení a zápis grafické databáze, utility souborových systémů apod. Tento modul zabezpečuje ukládání či načítání grafické databáze do/ze souboru a poskytuje pro tyto účely základní funkce a rozhraní.



Obr. 2: Schéma architektury OpenSceneGraph [HLD05]

Pro načítání rozličných formátů grafických databází slouží tzv. pluginy. Tyto pluginy dědí rozhraní od modulu osgDB. Pluginy jsou v OpenSceneGraph rovněž implementovány jako dynamické knihovny, které jsou načteny, když aplikace vyžaduje jejich použití. Všechny pluginy jsou automaticky registrovány pomocí osgDB registru během svého načítání a mohou využívat funkce poskytnuté modulem osgDB např. pro načítání, zápis, nalezení souborů apod.

Pluginy v OpenSceneGraph zahrnují práci jak s databázovými grafickými formáty jako jsou OpenFlight, 3DS, X nebo VRML 1.0, tak s obrazovými grafickými formáty jako jsou rgb, tiff, jpg, bmp, png nebo dds.

Vedle pluginů obsahuje OpenSceneGraph také sadu tzv. node-kitů, které jsou samostatnými knihovnami poskytujícími další funkce pro práci se scénou. Příkladem mohou být osgParticle, která přidává práci s částicovými systémy, osgText pro práci s textem, osgTerrain pro generování terénu a další.

3 Deformovatelný terén

Deformace terénu v grafické databázi zahrnuje širokou oblast v počítačové grafice. Jedná se o změnu terénu působením vnějších vlivů. Terénem se v dalším výkladu rozumí grafická databáze reprezentovaná toolkitem OpenSceneGraph a deformace terénu probíhá v reálném čase.

Terén se skládá z grafických objektů, jež jsou reprezentovány sítí trojúhelníků a tedy úprava neboli deformace terénu se děje úpravou geometrií, trojúhelníků, jednotlivých grafických objektů ve scéně. Vnější účinky, které působí na terén, lze rozdělit do dvou základních skupin.

První skupinou jsou deformace založené na nefyzikálním principu. Těmito deformacemi lze upravit model tělesa podle předem zadaných parametrů. V počítačové grafice se jedná zejména o různé modely křivek a splinů, jako jsou Beziérovy křivky, B-spliny, racionální B-spliny, neuniformní racionální B-spliny (NURBS) a jejich přirozené rozšíření do 3D, kde se jedná o plochy.

V oblasti modifikace terénu se jedná zejména o deformaci terénu podle tvarů grafických objektů, které jsou do databáze přidávány. Může se jednat o přidání grafického objektu s danou podstavou, jako je například dům. Tento nový grafický objekt, respektive jeho podstava či otisk (footprint), mění geometrii původního terénu, který se jemu přizpůsobí. Druhým příkladem je deformace terénu přidáním křivky či pásu reprezentovaným například nově vkládanou silnicí či řekou. Oba tyto příklady vedou k umělé deformaci terénu, která slouží zejména k realistickému modelování scény.

Druhou skupinou jsou deformace založené na fyzikálním principu. Jedná se o deformace způsobené okolnostmi, které nastaly během existence scény a které nemusejí být explicitně uživatelsky specifikovány. Příkladem může být deformace terénu způsobená výbuchem střely, dopadem objektu na terén nebo deformace typu zanechání stopy po průjezdu motorového vozidla. Fyzikálně deformovatelné modely jsou popsány v kapitole 4, Fyzikálně deformovatelné modely.

3.1 Projekt „Terrain Deformation Software“

Projekt „Terrain Deformation Software“ (TDS) je open source software vyvíjený jako plugin pro OpenSceneGraph, který umožňuje deformaci terénu v místech nově vložených grafických objektů. Projekt je vyvíjen pod záštitou sdružení Computer Graphics Systems Development Corporation (CGSD) a jeho vývoj byl rozdělen do dvou fází.

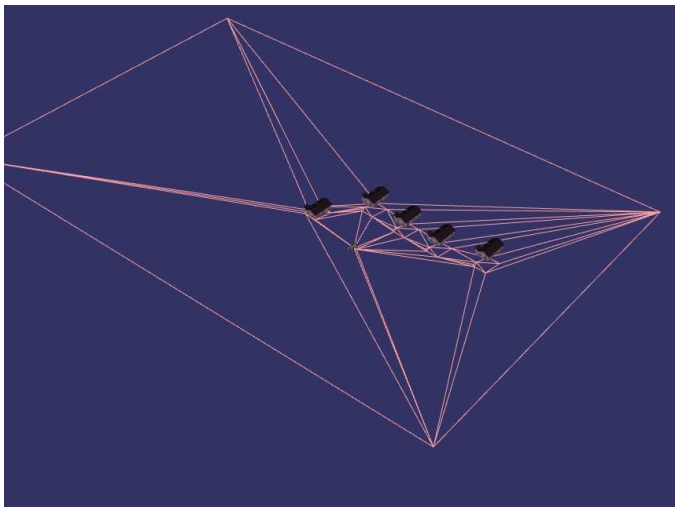
V první fázi, započaté v červnu 2004, bylo úkolem prostudovat stávající a navrhnout, implementovat a otestovat nové algoritmy pro interaktivní deformaci terénu. První fáze se pak měla stát východiskem pro druhou, realizační fázi. Na konci této etapy byly vyvinuty a ověřeny nové postupy pro změnu terénu dynamickým vkládáním nových objektů. Více o výsledcích první fáze se lze dočíst v [CSB05].

Druhá fáze byla započata v dubnu roku 2005. Její vývoj ještě sice zcela neskončil, nicméně v dubnu 2007 byla vypuštěna první oficiální verze 1.0.

Protože je TDS implementován jako plugin toolkitu OpenSceneGraph, je součástí návrhu i nový typ souboru s příponou tds. Struktura tds souboru je ve formě XML a soubor obsahuje značky

odkazující na soubory s definicí terénu a soubory s definicí přidávaných objektů, tzv. targets. Samotný algoritmus je rozdělen do tří částí.

V první části jsou do terénu vloženy vertexy přidávaných objektů ležící na konvexní obálce jejich footprintů. Tyto vertexy jsou pak pomocí triangulace propojeny s terénem. Tím se vytvoří tzv. Correction Space Mesh, což je znázorněno na Obr. 3.



Obr. 3: Ukázka Correction Space Mesh [Pro06]

Ve druhé části se provádí adaptivní dělení vzniklé trojúhelníkové sítě. Toto dělení se provádí s ohledem na to, jak takto vzniklá trojúhelníková síť opisuje původní terén. Jestliže se nově vzniklý trojúhelník, respektive jeho alespoň jedna strana, po korekci příliš odklání od původního terénu, je trojúhelník rozdělen na čtyři menší v polovině svých stran. Tento proces pokračuje dále rekurzivně.

V poslední, třetí, části dochází k aplikaci korekční funkce na každý nově vzniklý bod v terénu. Tato korekční funkce upraví výšku daného bodu podle Gaussova rozložení a pomocných vah. Tím se dosáhne hladkého napojení nově vzniklých trojúhelníků na původní terén. Více o použitém algoritmu se lze dočíst v [CSR05]. Jako ukázka dosažených výsledků slouží Obr. 4 a Obr. 5, převzaté z [Pro06].

Obr. 4: Scéna před aplikací TDS [Pro06]

Obr. 5: Scéna po aplikaci TDS [Pro06]

4 Fyzikálně deformovatelné modely

Modelování fyzikálně deformovatelných objektů se vyvíjí v oblasti počítačové grafiky již dvě desetiletí [Nea05]. Aplikace vytvořené pomocí fyzikálně deformovatelných modelů se pohybují od simulace elastických těles přes simulace textilií až po simulaci dynamiky tekutin. Na vývoji této oblasti počítačové vědy se podílelo a podílí mnoho výzkumných pracovníků a jedná se o bouřlivě se vyvíjející vědní obor, který kombinuje zejména newtonovskou dynamiku, spojitou mechaniku, numerickou matematiku, vektorový počet a počítačovou grafiku.

Během své existence se fyzikálně deformovatelné modely pouštěly různými cestami vývoje. Je to zejména z důvodu různorodého uplatnění této vědní disciplíny a různých nároků a požadavků kladených na konkrétní model.

Existuje mnoho pohledů na klasifikaci těchto modelů, z nichž nejčastějším je dělení uvedené v [GM97], kde se dělí fyzikálně deformovatelné modely na spojité a částicové modely.

V [Nea05] se objevilo v důsledku vývoje jiné, podrobnější dělení. Do Lagrangeovských síťových modelů patří výše zmíněné spojité a částicové modely. Do Lagrangeovských bez-síťových modelů náleží volně vázané částicové systémy, plynulá částicová hydrodynamika a tzv. bez-síťové metody. Zvláštní oblastí jsou redukované deformovatelné modely a modální analýza. Do čtvrté a poslední skupiny nazvané Eulerovské a semi-Lagrangeovské metody patří tekutiny, plyny a taveniny.

4.1 Fyzikální pozadí

Deformací a tokem hmoty se zabývá vědní obor reologie. V souladu s reologií mohou být všechny materiály modelovány od čistě elastických až po čistě viskózní. Žádné reálné materiály nevykazují čistě elastické nebo čistě viskózní chování, existují ovšem materiály, které se přibližují těmto extrémům nebo projevují částečně elastické a částečně plastické chování.

Těleso z materiálu vykazujícího elastické chování se deformuje pod účinkem vnější síly a vrací se do původního tvaru po jejím uvolnění. Vlastnost materiálu odolávat deformacím se vyjadřuje Youngovým modulem elasticity E a velikost odpovídající síly F je [Ton98]:

$$F = E\varepsilon \tag{1}$$

kde ε je relativní prodloužení, dáno vztahem [Ton98]:

$$\varepsilon = \frac{l - l_0}{l_0} \tag{2}$$

kde l je nová délka tělesa a l_0 je původní délka tělesa.

Pro lineárně elastické materiály platí tzv. Hookův zákon, podle kterého je míra deformace přímo úměrná síle způsobující deformace [You06]:

$$F = \left(\frac{EA_0}{l_0} \right) \Delta l = kx \qquad (3)$$

kde A_0 je plocha, skrze kterou působí síla F, $\Delta l = l - l_0$.

U plastických materiálů je situace odlišná. Existuje zde limitní velikost síly, po jejímž překročení dochází k permanentní deformaci. Jedním z modelů, který toto chování materiálu simuluje je Voightův model, který vyjadřuje [Ton98]:

$$F = E\varepsilon + \eta \frac{\partial \varepsilon}{\partial t} \qquad (4)$$

kde F je síla a η je viskózní koeficient.

4.2 Spojité modely

Spojité modely předpokládají energetickou rovnováhu obecného tělesa, na které působí vnější síly. Deformace tělesa je pak funkce těchto působících sil a materiálových vlastností tělesa. Těleso docílí rovnováhy, když je jeho potenciální energie minimální. Celková potenciální energie tělesa Π je dána [GM97]:

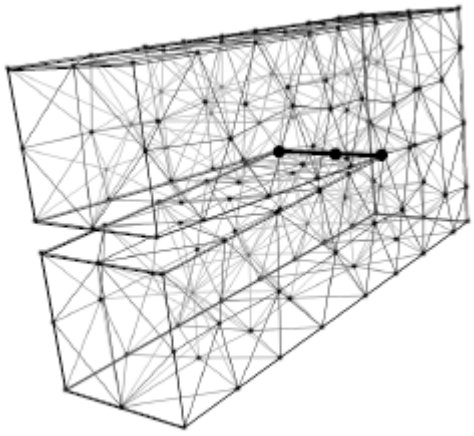
$$\Pi = \Lambda - W \qquad (5)$$

kde Λ je celková energie napětí tělesa a W je práce vykonaná vnějšími silami.

Energie napětí tělesa je energie uložená v tělese jako deformace materiálu. Práce vykonaná vnějšími vlivy může pocházet ze tří zdrojů. Jsou to koncentrované síly působící v diskrétních bodech, síly působící na celé těleso, jako je gravitace, a síly působící na povrch tělesa, jako jsou tlakové síly.

K určení rovnovážného tvaru tělesa jsou obě energie Λ a W vyjádřeny vzhledem k deformaci tělesa, které je vyjádřeno jako materiálové přesunutí. Potenciální energie systému dosahuje minima, pokud je derivace Π rovna nule. Tento přístup vede k řešení spojité diferenciální rovnovážné rovnice pro materiálové přesunutí.

Není vždy možné najít analytické řešení rovnovážné rovnice, proto se hledá určitá aproximace řešení užitím různých numerických metod. Pro spojité modely se často využívá tzv. FEM (Finite Elements Methods), které rozdělí těleso do konečného počtu elementů (viz Obr. 6) a aproximují rovnovážnou rovnici přes každý element. Více informací o FEM je obsaženo v [GM97] a [Nea05].



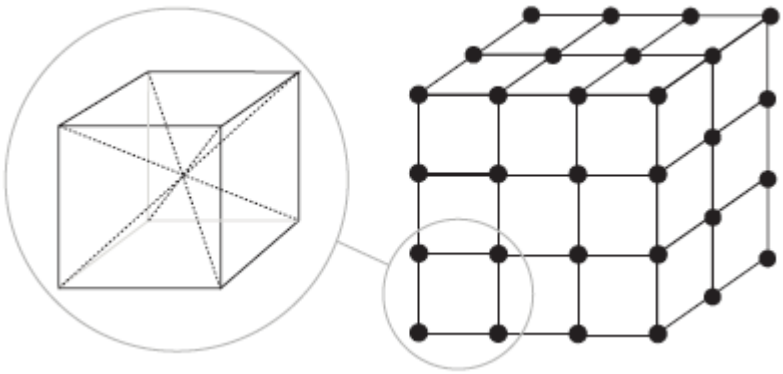
Obr. 6: Ukázka dělení tělesa na tetraedry [Nea05]

Spojité modely obecně předpokládají malé změny v deformaci tělesa. Pokud je toto splněno během simulace, jsou výsledky dané řešením soustavy diferenciálních rovnic poměrně přesné.

Z podstaty FEM či alternativních numerických metod, používaných pro řešení soustav diferenciálních rovnic, jsou tyto metody ovšem výpočetně náročnější.

4.3 Částicové systémy

Jedná se pravděpodobně o nejjednodušší a nejintuitivnější deformovatelný model. Částicové systémy (mass-spring models) jsou oproti spojitým modelům již ze své podstaty diskrétními modely. Model tělesa je aproximován pomocí pravidelné mřížky částic v prostoru s pevně definovanými vazbami. Obvyklá struktura je ukázána na Obr. 7.



Obr. 7: Běžná struktura částicového systému [Nea05]

Pro každý bod v mřížce je aplikován druhý Newtonův zákon [GM97]:

$$m_i \ddot{x}_i = -\gamma_i \dot{x}_i + \sum_j g_{ij} + f_i \qquad (6)$$

kde m_i je hmotnost částice, x_i poloha částice, γ_i tlumící koeficient, g_{ij} síla způsobená vazbou mezi částicemi i a j, a f_i je suma vnějších sil působících na částici i.

Druhý člen pravé strany rovnice (6) vyjadřuje součet sil způsobených definovanými vazbami mezi částicemi i a j.

Nejlevější člen pravé strany rovnice (6) je označován jako tlumící síla, která působí proti směru pohybu částice. Tato tlumící síla je důležitá zejména pro stabilitu systému. Jestliže na systém nepůsobí žádná vnější síla, je systém díky tlumící síle uveden do klidového stavu.

Rovnice (6) je převedena na soustavu dvou diferenciálních rovnic prvního řádu a je vyřešena pro každou částici v modelu vybranou numerickou metodou.

Pomocí částicových modelů je možné simulovat velké deformace jak elastických tak plastických těles. Na druhou stranu, nevýhodou částicových systémů je na rozdíl od spojitých modelů nutnost ladění parametrů diferenciální rovnice, protože ty přímo neodpovídají měřitelným fyzikálním vlastnostem materiálů. U částicových systémů je nutné též volit sofistikovanější numerické metody, neboť se může stát, že výsledek nebude konvergovat ke správnému řešení vlivem příliš velkého integračního kroku.

4.4 Plynoucí částice a bez-síťové metody

Tento způsob simulace deformovatelných těles patří do bez-síťových deformovatelných metod. Na rozdíl od síťových modelů je hmota tělesa navzorkována částicemi, které mají kromě hmotnosti a pozice definovanou také hustotu a objem. Dalším rozdílem je, že hmota částic je distribuována do prostoru pomocí tzv. vyhlazovacího jádra, které je normalizováno [DG96, Nea05].

Modelem plynoucích částic lze aproximovat okamžitou hodnotu silového pole <*f(r)*> kdekoliv uvnitř či vně tělesa či tekutiny [DG96]:

$$\langle f(r) \rangle = \sum_j m_j \frac{f_j}{\rho_j} W_h(r - r_j) \tag{7}$$

kde m_j je hmotnost částice j, f_j je velikost síly v místě částice j, ρ_j je hustota částice j, r je aktuální poloha, r_j je poloha částice j a W_h je vyhlazovací jádro.

Pokud je znám tlak v poloze každé částice, je možné aproximovat působící síly díky změnám tlaku. Tento tlak je možné díky informacím u každé částice vyjádřit pro tekutiny pomocí rovnice pro ideální plyn [DG96]:

$$PV = k \tag{8}$$

kde P je tlak uvnitř tekutiny, V je objem částice a k je daná konstanta.

Pro materiál s konstantní hustotou ρ_0 lze použít vztah [DG96]:

$$P = k(\rho - \rho_0) \tag{9}$$

kde P je tlak v místě částice, k je daná konstanta a ρ je hustota v místě částice.

U částicově založených animací, které podle [Nea05] také patří do skupiny bez-síťových metod, se působící síly na částici určí z derivace pole přesunutí materiálu (displacement field) a tlakové energie v místě částice. Podrobnosti o této metodě lze nalézt v [Mue04] a [Pau05].

Oběma popsanými metodami lze modelovat také elasto-plastické materiály přidáním tlumící síly do pohybové rovnice částice.

4.5 Volně vázané částicové systémy

Stejně jako v klasických částicových systémech, je i u volně vázaných částicových systémů využívána klasická fyzika pro pohyb částic v prostoru, který je vyjádřen pomocí rovnice (6). Rozdíl je v tom, že nejsou pevně definované vazby mezi jednotlivými částicemi. Namísto toho se definuje potenciální energie systému, která určuje síly mezi částicemi na základě jejich polohy.

Pro potenciální energii Φ je definována síla f_i působící na částici i takto [Ton98]:

$$f_i = -\nabla_{x_i} \Phi \tag{10}$$

kde $\nabla_{xi}\Phi$ značí gradient potenciální energie Φ podle polohy částice x_i.

Potenciální energie Φ_i částice i se pro systém s N částicemi definuje takto [Ton98]:

$$\Phi_i = \sum_{j \neq i}^{N} \Phi_{ij} \tag{11}$$

kde Φ_{ij} je potenciální energie mezi částicemi *i* a *j*.

Výpočet potenciálních energií pro všechny částice v systému podle vzorce (11) má výpočetní složitost $O(N^2)$ a je proto pro praktické účely nevhodný. Mezi vzdálenými částicemi je potenciální energie zanedbatelná, je možné vypočítat potenciální energie částice jako součet potenciálních energií se sousedícími částicemi bližšími než stanovená vzdálenost:

$$\Phi_i = \sum_{j \in N_i} \Phi_{ij} \tag{12}$$

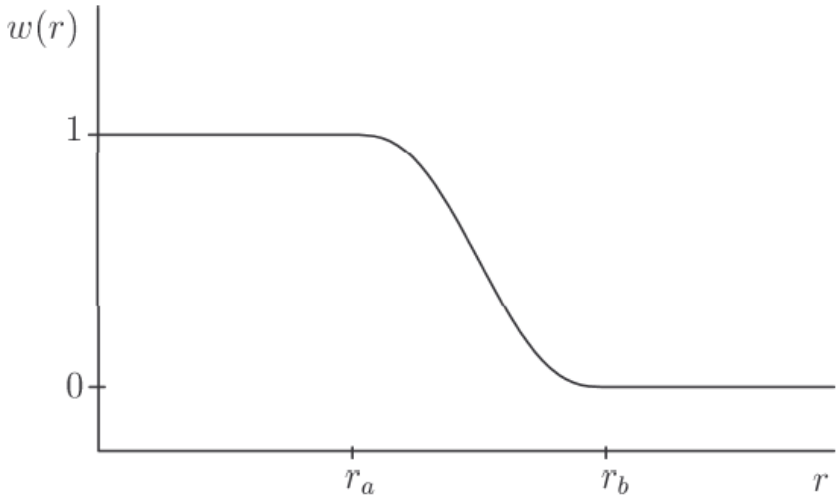
kde N_i je množina částic sousedících s částicí *i*.

Prosté ignorování vzdálených částic při výpočtu potenciální energie částice *i* podle vztahu (12) může vést vlivem nespojitosti potenciální energie v systému k numerické nestabilitě či ke vzniku nežádoucích vizuálních artefaktů. Proto je vhodné velikost potenciální energie u hranice sousednosti vést spojitě k nule. Toto se dá zajistit pomocí váhování potenciální energie.

Vhodnou váhovací funkcí *w(r)* je například [Ton98]:

$$w(r) = \begin{cases} 1 & r < r_a \\ g(s) & r_a \leq r \leq r_b, \quad s = \dfrac{r - r_a}{r_b - r_a} \\ 0 & r > r_b \end{cases} \tag{13}$$

kde *r* je vzdálenost, $0 < r_a < r_b$, a funkce $g(s) = -6s^5+15s^4-10s^3+1$, která má spojitou první i druhou derivaci pro $s=0$ a $s=1$. Obr. 8 přejatý z [Ton98] ukazuje průběh funkce *w(r)*.



Obr. 8: Průběh funkce *w(r)* s označenými body r_a a r_b [Ton98]

Výpočet potenciální energie částice *i* se potom změní ze vztahu (12) takto:

$$\Phi_i = \sum_{j \in N_i} w(r_{ij}) \Phi(r_{ij}) \tag{14}$$

kde r_{ij} je vzdálenost mezi částicemi i a j.

Vhodná potenciální funkce $\Phi(r_{ij})$, která určuje energii mezi dvěma částicemi, jejichž vzdálenost je r_{ij}, a která umožňuje modelovat širokou škálu deformací, je tzv. Lennard-Jonesova energetická funkce. Tato funkce je pouze funkcí vzdálenosti a tudíž není potřeba explicitně definovat vazby mezi částicemi jako je tomu u klasických částicových systémů. Lennard-Jonesova funkce $\Phi_{LJ}(r)$ má obecný tvar [Ton98, Nea05]:

$$\Phi_{LJ}(r) = \frac{B}{r^n} - \frac{A}{r^m} \tag{15}$$

kde r je vzdálenost částic, n, m, A a B jsou konstanty.

Alternativní a vhodnější reprezentací Lennard-Jonesovy funkce je [Ton98, Nea05]:

$$\Phi_{LJ}(r) = \frac{-e_0}{m-n}\left(m\left(\frac{r_0}{r}\right)^n - n\left(\frac{r_0}{r}\right)^m \right) \tag{16}$$

kde r je vzdálenost částic, r_0 je rovnovážná vzdálenost částic, e_0 je tzv. disociační energie, tj. energie potřebná k oddělení částic. Odvození je v [Ton98].

Změnou parametrů m a n lze dosáhnout různorodých vlastností materiálu. Při nižších hodnotách jsou materiály více deformovatelné a naopak. Je také možné nastavovat rovnovážnou vzdálenost r_0, při které je potenciální energie dvou částic minimální a síla, kterou na sebe částice působí je nulová.

Pro simulace plasticity lze přidat, jako je tomu u klasických částicových systémů, ke každé částici i tlumící sílu f_{di}, která působí proti relativnímu pohybu částic i a j [Nea05]:

$$f_{di} = k_d (v_j - v_i) \tag{17}$$

kde k_d je tlumící konstanta, v_j je rychlost částice j a v_i je rychlost částice i.

V [WTT95] a [Ton98] byly pro popis deformovatelných těles použity tzv. povrchové částice nebo částicové systémy. V tomto případě je těleso popsáno pouze svým povrchem a jeho vlastnostmi potenciálních energií v rámci povrchu. I zde byla použita tlumící síla f_{di}, dána vztahem (17) pro modelování plastických deformací. V [Ton98] je definován nový model orientovaných částicových systémů, který používá kromě Lennard-Jonesové potenciální funkce také jiné funkce popisující jiné potenciální energie zajišťující hladký povrch tělesa.

Volně vázané částicové systémy lze použít pro velkou škálu deformací výběrem vhodné aproximace potenciální funkce a použití tlumící síly podobně jako tomu je u klasických částicových systémů k modelování plasticity. Navíc je možné popsat deformovatelné těleso pouze částicemi na jeho povrchu, které se přímo mohou využít k vizualizaci tělesa.

5 Numerická integrace

Všechny fyzikálně deformovatelné modely popsané v kapitole 4, Fyzikálně deformovatelné modely, potřebují pro řešení diferenciálních rovnic, jako je rovnice (6), užít nějaký způsob integrace pro výpočet nového polohového vektoru a vektoru rychlosti. Analytické řešení není možné vždy určit, a proto se používá řešení numerické.

Při řešení diferenciálních rovnic numerickými metodami se téměř vždy objevuje ve výsledcích numerická chyba. Je to způsobeno tím, že každá numerická metoda vychází z Taylorova rozvoje [WB97] a všechny metody ignorují členy vyšších řádů.

S chybou metody či přesností výpočtu souvisí i numerická stabilita metody. U jednoduchých metod se může stát, že výsledek se vzdaluje od řešení, neboli diverguje. Metoda je pak ve spojení s touto konkrétní diferenciální rovnicí numericky nestabilní a tato rovnice je z hlediska numerické matematiky označována jako tuhá rovnice.

Metody, které počítají výsledek v dalším integračním kroku pouze z předchozích výsledků, se označují jako explicitní. Mezi tyto metody patří např. Eulerova metoda, midpoint metoda či Runge-Kutta čtvrtého řádu. Obecně nelze stabilitu u těchto metod zaručit [Bar97] a je často nutné volit velmi malé integrační kroky, čímž roste výpočetní náročnost.

Stabilitu lze do jisté míry zaručit metodami tzv. implicitními, kde se neznámá proměnná (vektor) objevuje na obou stranách rovnice. Implicitní metody zaručují pro lineární systémy stabilitu s libovolně velkým integračním krokem a pro nelineární systémy je stabilita větší než u explicitních metod. Řešení implicitními metodami ovšem vede k řešení soustavy lineárních nebo nelineárních rovnic, což je výpočetně mnohem náročnější než použití explicitních metod. Konkrétnější vysvětlení implicitních metod je popsáno např. v [Bar97] a [Kas97].

5.1 Eulerova metoda

Nejjednodušší integrační metodou je Eulerova integrační metoda. Nechť je počáteční hodnota vektoru x v čase t dána $x_0 = x(t_0)$ a odhad vektoru x v čase t_0+h dán $x(t_0+h)$, kde h je velikost integračního kroku. Vzorec explicitní Eulerovy metody je pak následující [WB97]:

$$x(t_0 + h) = x_0 + h\,\dot{x}(t_0) \qquad (18)$$

Lze říci, že výsledek je odhadnut pouze z první derivace funkce vektoru x v čase t_0. Tento příliš jednoduchý a hrubý odhad vede často, jak je napsáno výše, k problémům nepřesnosti a nestability, a proto je v těchto případech nutné volit mnohdy neprakticky malé integrační kroky h.

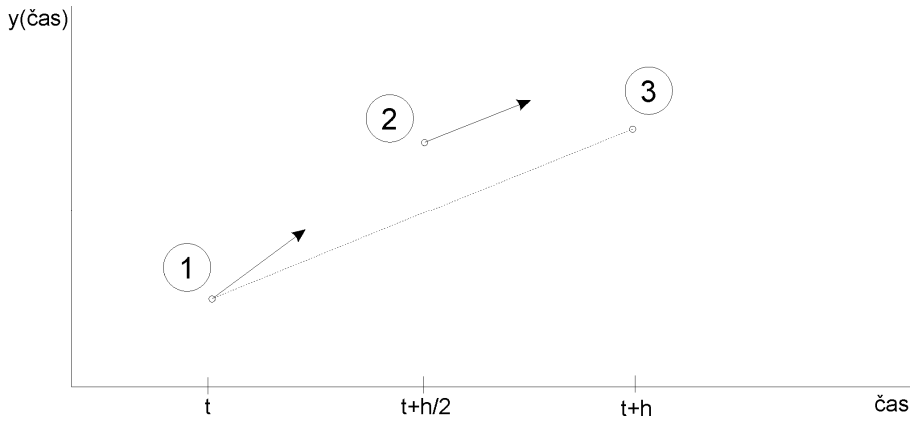
Eulerovu integrační metodu lze pro druhý Newtonův zákon ($F=ma$, F je součet sil působících na těleso o hmotnosti m, které mu udávají zrychlení a), pro vypočtení nové rychlosti v^{t+h} v čase $t+h$ a nové polohy x^{t+h} v čase $t+h$, použít takto:

$$a^t = \frac{F^t}{m}$$
$$v^{t+h} = v^t + ha^t \qquad (19)$$
$$x^{t+h} = x^t + hv^{t+h}$$

Díky své implementační jednoduchosti je tato metoda poměrně využívána, přičemž stabilita je dosažena patřičným laděním systému a stanovením různých omezujících podmínek.

5.2 Runge-Kuttova metoda druhého řádu

Runge-Kuttova metoda druhého řádu vychází z Eulerovy integrační metody. Prakticky využívá Eulerovu integraci k odhadnutí hodnoty funkce v polovině intervalu integračního kroku *h*, kde se následně spočítá nová derivace proměnné. Odtud také pramení druhý název pro tuto metodu – midpoint metoda. Nově vypočítaná derivace pak určuje směr, kde bude ležet hodnota proměnné v čase *t*+*h*, kde *t* vyjadřuje stávající hodnotu času. Tento postup výpočtu znázorňuje Obr. 9.



Obr. 9: Výpočet nové hodnoty funkce *y* metodou Runge-Kutta druhého řádu. V čase *t* se nejprve vypočítá derivace funkce *y* (směr šipky), poté následuje výpočet hodnoty funkce *y* v čase *t*+*h*/2, následuje výpočet derivace v čase *t*+*h*/2, která se použije pro konečný výpočet hodnoty 3 v čase *t*+*h*.

Matematický zápis pro výše uvedený postup vypadá takto [PTWF97]:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \qquad (20)$$
$$y_{n+1} = y_n + k_2$$

kde k_1 a k_2 jsou proměnné, h je velikost integračního kroku, x_n je aktuální čas, y_n je předchozí výstup integrátoru (stav), f je funkce pro výpočet derivace v daném čase a stavu a y_{n+1} je konečný výstup z integrátoru.

Z rovnice (20) je zřejmé, že výpočet derivací (zrychlení a rychlostí) je nutné v jednom kroku provádět dvakrát. První pro současný čas druhý pro čas v polovině intervalu integračního kroku. Zejména z tohoto důvodu je Runge-Kuttova metoda druhého řádu teoreticky dvakrát pomalejší než

Eulerova metoda při zachování stejné velikosti kroku. Na druhou stranu vykazuje větší stabilitu, a proto ji lze použít s delším integračním krokem.

5.3 Runge-Kuttova metoda čtvrtého řádu

Runge-Kuttova metoda čtvrtého řádu patří k pokročilejším jednokrokovým metodám a je běžně používána tam, kde je potřeba přesnějších výsledků. Tato metoda je opět odvozena z Taylorova rozvoje [WB97]. Její typický obecný zápis vypadá takto [PTWF97]:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \end{aligned} \qquad (21)$$

kde k_1, k_2, k_3 a k_4 jsou proměnné, h je velikost integračního kroku, x_n je aktuální čas, y_n předchozí výstup integrátoru (stav), f je funkce pro výpočet derivace v daném čase a stavu, a y_{n+1} je konečný výstup (stav) integrátoru.

Z rovnice (21) je zřejmé, že při výpočtu výstupu z integrátoru y_{n+1} je nutné počítat velikosti zrychlení (derivace rychlosti) a rychlostí (derivace dráhy) celkem čtyřikrát. To se samozřejmě projeví na době výpočtu, která je oproti Eulerově metodě čtyřnásobná. Z důvodu zejména stability je ovšem použití této metody nebo podobných nezbytné v případech, kdy se musí zvětšit integrační krok nebo kde jsou velké rozdíly hodnot u derivací (působících sil, rychlostí) apod.

5.4 Integrační metoda „leapfrog“

Integrační metoda leapfrog je, podobně jako metoda midpoint, numerická integrační metoda druhého řádu. U metod vyšších řádů se s výhodou definují rychlosti ve středech intervalů $t \pm h/2$ [McM05].

Obecně se tato integrační metoda vyjadřuje vztahy [McM05]:

$$\begin{aligned} x^{t+h} &= x^t + hv^{t+h/2} \\ v^{t+3/2h} &= v^t + hf(x^{t+h}) \end{aligned} \qquad (22)$$

kde $f(x^{t+h})$ je derivace x v čase $t+h$.

Integrační metodu leapfrog lze pro druhý Newtonův zákon ($F=ma$, F je součet sil působících na těleso o hmotnosti m, a které mu udávají zrychlení a), pro vypočtení nové rychlosti $v^{t+h/2}$ v čase $t+h/2$ a nové polohy x^{t+h} v čase $t+h$, použít takto [Tan98]:

$$
\begin{aligned}
a^t &= \frac{F^t}{m} \\
v^{t+h/2} &= v^{t-h/2} + ha^t \\
x^{t+h} &= x^t + hv^{t+h/2}
\end{aligned}
\tag{23}
$$

Rovnice (23) se téměř neliší od rovnice (19), přičemž výpočetní složitost je stejná a navíc je v [Tan98] empiricky ukázáno, že tato metoda je na rozdíl od Eulerovy metody mnohem přesnější a stabilnější při stejně velkém integračním kroku.

Nevýhodou je absence velikosti rychlosti v^{t+h} v čase $t+h$, která je nutná, vzhledem k částicovým systémům, pro výpočet tlumících sil (nejlevější člen pravé strany rovnice (6) nebo rovnice (17)). Rychlost v^{t+h} je proto nutno aproximovat např. lineární extrapolací z rychlostí $v^{t-h/2}$ a $v^{t+h/2}$ [Tan98]:

$$
v^{t+h} = \frac{3v^{t+h/2} - v^{t-h/2}}{2} \tag{24}
$$

5.5 Adaptivní velikost integračního kroku

Ke zvýšení přesnosti integrační metody a tím i její stability, se často dosahuje adaptivní velikostí integračního kroku. Základní myšlenkou tohoto postupu je nastavit integrační krok podle aktuální potřeby. Jestliže je současná přesnost výpočtu větší, než je potřeba, dojde k prodloužení integračního kroku. Naopak, jestliže je nynější přesnost výpočtu menší, než je požadováno, dojde k patřičnému zmenšení integračního kroku.

Ve [WB97] je uveden postup, jak adaptovat velikost integračního kroku pro Eulerovu metodu. Nechť je v současném čase t dán integrační krok h. Základní otázka zní: na jakou velikost je třeba změnit integrační krok h, aby byla dosažena potřebná přesnost?

Provede se dvojí odhad pro $x(t+h)$, a to zaprvé odhad x_a jednou integrací s krokem h, a zadruhé odhad x_b dvěma po sobě jdoucími integracemi s krokem $h/2$. Oba odhady se liší pro Eulerovu integrační metodu, která je prvního řádu, od skutečné hodnoty $x(t+h)$ řádovou chybou $O(h^2)$ [WB97], což znamená, že i oba odhady se liší o $O(h^2)$. Velikost chyby e je potom odhadnuta:

$$
e = \left| x_a - x_b \right| \tag{25}
$$

Výsledné nastavení nového integračního kroku h_{new} potom je:

$$
h_{new} = \left(\frac{e_0}{e} \right)^{1/2} \tag{26}
$$

kde e_0 je požadovaná přesnost.

6 Návrh

Pro návrh vhodného způsobu fyzikální deformace terénu je nutné specifikovat požadavky, které jsou na výsledný model určeny.

Prvním závazným požadavkem projektu je fyzikální podstata deformace terénu, která je přítomna u všech modelů uvedených v kapitole 4, Fyzikálně deformovatelné modely. Dalším požadavkem je real-time simulace deformace terénu. To znamená, že požadovaná rychlost simulace musí být minimálně okolo třiceti snímků/kroků za vteřinu. Z tohoto hlediska musí být vybrán fyzikálně deformovatelný model i numerická integrační metoda jako kompromis mezi přesností a výkonem. Třetím hlavním požadavkem je implementace výsledného systému v prostředí OpenSceneGraph.

Co se týká druhu deformace terénu, byla uvažována během návrhu a implementace deformace způsobená pohybem objektu po povrchu terénu, čímž vzniká stopa (rýha) v terénu a deformace terénu způsobená explozí výbušniny, kdy se síla šíří do prostoru z jednoho bodu, a tím vznikne v terénu kráter.

Výše zmíněné požadavky byly brány v potaz při vlastním návrhu a realizaci fyzikálně deformovatelného terénu, což zahrnuje několik oblastí problému, které je potřeba řešit. Tyto oblasti jsou popsány v následujících kapitolách.

6.1 Fyzikálně deformovatelný terén

Terén v OpenSceneGraph je definován sítí trojúhelníků, které představují jeho povrch. Protože deformace jsou způsobené takto definovanému terénu, byl vybrán povrchový částicový model pro reprezentaci terénu.

Jednotlivé částice tohoto částicového systému mají stejnou inicializační pozici jako vrcholy již existujících trojúhelníků stávajícího terénu. Dále jsou tyto částice svázány s danou grafickou reprezentací terénu, což znamená, že po jejich vychýlení dojde také k vychýlení vrcholů trojúhelníků a tím vznikne vizuální deformace terénu.

Pro soudržnost terénu byla vybrána Lennard-Jonesova potenciální funkce, podle které lze definovat přitažlivé síly držící částice pohromadě. Výhodou této reprezentace je možnost určení disociační energie, možnost určení rovnovážné vzdálenosti mezi částicemi a možnost měnit tuhost materiálu (asfalt × půda). Rozdílem reprezentace Lennard-Jonesovy funkce je oproti [Ton98] v tom, že vzdálenosti sousedních částic se mohou lišit (trojúhelníky mohou mít různou velikost). Proto se klidové vzdálenosti mezi jednotlivými částicemi určují až při inicializaci každého částicového systému podle aktuálních vzdáleností.

Pro vyjádření velikosti působící síly je nutné vypočítat gradient z potenciálního energetického pole, jak je uvedeno v rovnici (10). Pro Lennard-Jonesovu potenciální funkci danou vztahem (16) se výpočet gradientu rovná derivaci potenciálu podle vzdálenosti (jedná se o funkci o jedné neznámé):

$$\frac{\partial \Phi_{LJ}}{\partial r} = \frac{-e_0 mn}{(n-m)r}\left(\left(\frac{r_0}{r}\right)^n - \left(\frac{r_0}{r}\right)^m\right) \tag{27}$$

kde význam všech symbolů je stejný jako v rovnici (16).

Pro výpočet stavu částicových systémů je nutné vyzkoušet několik integračních metod zejména v případech deformací způsobených explozemi, kde působí na částice velké síly, čímž kvůli přesnosti vzrůstají nároky na integrátor.

6.2 Detekce kolizí

Pro detekci kolizí je primárním cílem efektivita a samozřejmě přesnost. Jak již bylo řečeno, terén je sestaven z velkého počtu trojúhelníků (řádově desítky až stovky tisíc) a není tudíž efektivní detekovat kolize bez použití sofistikovaného přístupu jako je použití hierarchického dělení prostoru (Bounding Volume Hierarchy - BVH).

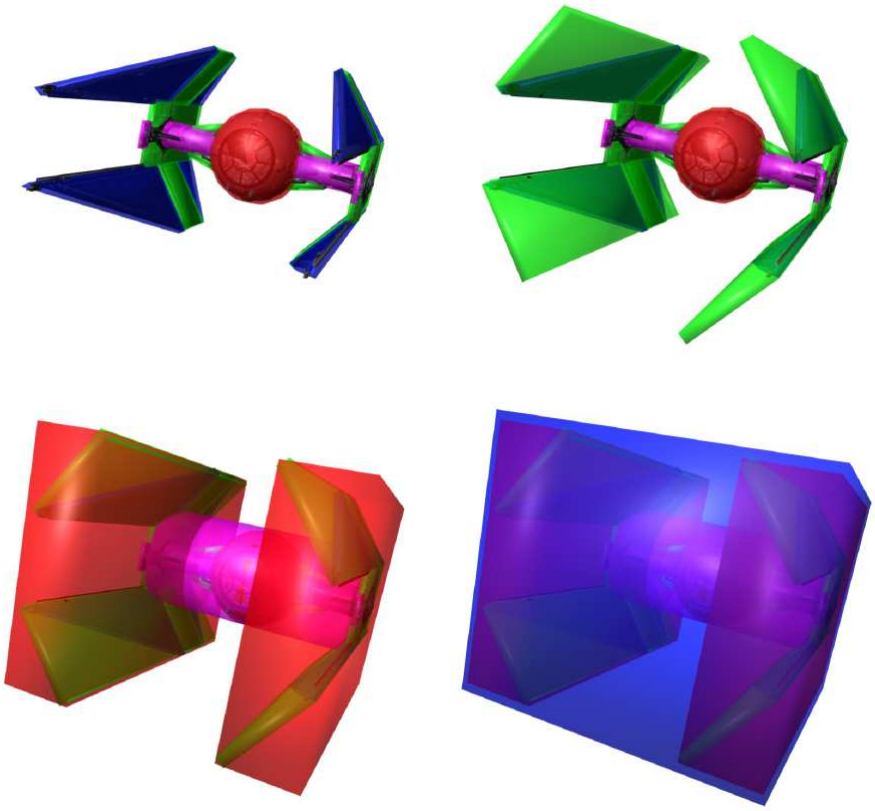
6.2.1 Hierarchické dělení prostoru

V běžných případech detekcí kolizí se pro každý objekt ve scéně sestaví zpravidla strom obálek, jenž představuje různé úrovně popisu daného objektu (Obr. 10). Nejhrubším popisem je obálka v kořeni stromu, nejjemnějším je pak sada obálek uložených v listech stromu obalující jednotlivá grafická primitiva (trojúhelníky, obecné polygony, úsečky apod.) daného objektu.

Na Obr. 11 je uveden algoritmus pro detekce kolize. Vstupem funkce Detekce_kolize jsou kořeny obalujících stromů objektů A a B. Jako první se otestuje kolize mezi obálkami v kořeni. Pokud není detekována kolize, funkce skončí. Pokud je detekována kolize mezi obálkami A a B, detekuje se kolize pro oba podstromy obou kořenů rekurzivně, až se algoritmus dostane k listům obou stromů. Teprve zde dochází k testování kolizí mezi grafickými primitivy a určují se potřebné kontaktní informace, jako je bod nebo normála kolize, potřebných pro další výpočet. Během algoritmu pro detekci kolizí používajícího BVH se efektivně odstraňují celé podstromy hierarchické struktury, které nejsou navzájem v kontaktu, a tím se značně zefektivní vlastní detekce.

Co se týká obálek, ty mohou být obecně různého charakteru, přičemž mezi používané patří zejména obalové koule, kvádry, hranoly, válce atd. Pro detekci kolizí u objektů složených z trojúhelníků, což je i případ terénu, se nejvíce rozšířily obálky ve tvaru kvádru.

Kvádry mohou být přitom dvojího druhu. Jsou to orientované obalové kvádry (Oriented Bounding Box - OBB) nebo to jsou kvádry zarovnané se souřadnicovými osami (Axis Aligned Bounding Box - AABB).



Obr. 10: Úrovně obálek popisující objekt s použitím heterogenního stromu [ED04]

Obě reprezentace pomocí OBB nebo AABB obálek mají své výhody a nevýhody a hodí se na různé druhy problémů. OBB stromy přesněji popisují tvar tělesa než AABB stromy, a proto je prořezávání stromu efektivnější. Na druhou stranu je detekce kolizí mezi dvěma OBB kvádry výpočetně náročnější, než detekce mezi dvěma AABB kvádry. Navíc pro deformovatelné objekty je použití OBB nevhodné, protože určení nového OBB pro dané body v prostoru je výpočetně mnohem náročnější než výpočet AABB.

```
Detekce_kolize( BVH: A, BVH: B)
{
  if not kolize(A, B)
    return
  if not list(A)
    pro všechny potomky C uzlu A
      Detekce_kolize(C, B)
  if not list(B)
    pro všechny potomky C uzlu B
      Detekce_kolize(A, C)
  //if list(A) and list(B)
  Zjistit_kontakt(A,B)
}
```

Obr. 11: Rekurzivní algoritmus detekce kolizí dvou objektů A a B s definovanými BVH

6.2.2 Prořezání obdélníků (box pruning)

Použití samotného hierarchického dělení prostoru značně urychluje detekci kolizí několikanásobně. Je ovšem nutné detekovat kolize mezi všemi možnými páry potenciálně kolidujících objektů, což znamená asymptotickou složitost $O(n^2)$. Toto řešení je tudíž nepraktické už pro počet objektů v řádu stovek natožpak tisíců a výše. Z tohoto důvodu se provádí prořezávání AABB kvádrů (angl. box pruning), které efektivně eliminuje ty objekty, jejichž obalové obdélníky na nejvyšší úrovni s ničím nekolidují.

Každý AABB obdélník je definován dvojicí bodů. Počáteční bod je ten, který obsahuje souřadnice s menšími hodnotami (lexikograficky x > y > z), než má druhý, koncový, bod. To znamená např. pro dvojici bodů se souřadnicemi (10, 20, 30) a (10, 19, 30), že počáteční bod bude mít souřadnice (10, 19, 30) a koncový (10, 20, 30).

Na Obr. 12 je znázorněn pseudokód algoritmu box pruning.

```
Box_pruning(AABBset : aabbs, PAIRset pairs)
{
  seradit_vsechny_pocatecni_body_obdelniku(aabbs)

  aktivni_obdelnik = prvni_obdelnik z aabbs
  i = 0
  dokud plati (i < pocet_obdelniku(aabbs) )
    j = i
    dokud plati (aktivni_obdelnik.maxCoords > aabbs[j].minCoords
                 and j < pocet_obdelniku(aabbs))
      // detekovano prekryti
      pairs.push(index_of(aktivni_obdelnik), index_of(aabbs[j]))
      j++

    aktivni_obdelnik++
    i++

}
```

Obr. 12: Algoritmus box pruning pro rychlou počáteční detekci kolizí

Efektivita algoritmu box pruning záleží na řadící metodě (bubblesort, quicksort, radixsort, atd.) a také na způsobu detekce překrytí. Podle algoritmu na Obr. 12 je celková výpočetní složitost s použitím quicksortu ($O(n \cdot log(n))$) a uvedeného způsobu detekce překrytí ($O(n+k)$, kde k je počet párů překrytých obdélníků) dána $O(n \cdot log(n)+k)$, což je veliká úspora oproti metodě silou ($O(n^2)$) zejména při velkém počtu těles ve scéně.

6.2.3 Detekce kolizí s terénem

Na základě skutečnosti, že model terénu je obecně rozlehlý a že detekce kolizí s terénem musí být efektivní, je vhodné terén lépe strukturovat, než s pouhým použitím jednoduchého hierarchického stromu. Navíc geometrie terénu je v OpenSceneGraphu definovaná jako množina koncových uzlů v grafu scény, čehož lze dále při návrhu a implementaci hierarchické struktury terénu využít.

Hlavní myšlenkou je rozdělit terén na regiony, kde každý region odpovídá jednomu koncovému uzlu terénu v grafu scény. Každý tento region přitom může být libovolného tvaru – např. tvar silnice, řeky, pole, lesa apod. Tyto regiony, skládající se ze sítě trojúhelníků, mají definovanou svou hierarchickou stromovou strukturu. Terén je potom definován celkovou stromovou strukturou, jejíž koncové uzly tvoří jednotlivé regiony. Jedná se tedy o strom se stromy ve svých koncových uzlech. Protože je terén deformován účinkem vnějších sil, je vhodné pro obalové obdélníky zvolit typ AABB, kvůli nastávajícím změnám geometrie terénu.

Algoritmus detekce kolizí mezi tělesy a terénem potom probíhá ve dvou fázích. V první fázi se detekují použitím algoritmu box pruning (Obr. 12) ty páry region-těleso, jejichž vnější AABB se překrývají, čímž se efektivně eliminují regiony terénu a tělesa, která s ničím nekolidují. Ve druhé fázi se potom vypočítají přesné kontaktní trojúhelníky a body mezi skutečně kolidujícími tělesy a regiony algoritmem uvedeným na Obr. 11.

6.3 Deformace pohybem po terénu

Během pohybu tělesa po terénu dochází mezi těmito objekty k četným drobnějším kolizím, které pak jako celek způsobí celistvou deformaci terénu. Při tomto procesu je třeba analyzovat a spočítat odezvu u každého kontaktu, neboť ne v každé kolizi vznikají odezvy v podobě např. reakčních impulsů či tlakových sil.

Obecně, pokud dvě tělesa kolidují, mohou nastat tři případy kontaktu. V prvním případě, pokud se tělesa od sebe vzdalují, se nepočítá žádná odezva kolize. Druhým případem je, že se tělesa k sobě přibližují. V tom případě se vypočítá na základě jejich relativní rychlosti a hybnosti impuls (zpětný ráz), tělesa skokově změní svůj směr pohybu a zabrání se jejím vzájemným vniknutím. Ve třetím případě se tělesa od sebe ani nevzdalují ani se nepřibližují, ale jsou v klidovém kontaktu. V tomto posledním případě může a nemusí mezi tělesy působit kontaktní síla, a proto musí být tento stav dále vyšetřen.

Nechť jsou dána dvě tělesa *A* a *B*, která jsou v kontaktu v bodě *p*. Pro určení správného kontaktního případu, se musí spočítat relativní rychlost v_{rel} těles *A* a *B* v aktuálním čase t_0 [B97]:

$$v_{rel} = \hat{n}(t_0) \cdot (\dot{p}_a(t_0) - \dot{p}_b(t_0)) \tag{28}$$

kde $\hat{n}(t_0)$ je jednotkový normálový vektor směřují k tělesu *A*, $\dot{p}_a(t_0)$ je rychlost bodu *p* vzhledem k tělesu *A* a $\dot{p}_b(t_0)$ je rychlost bodu *p* vzhledem k tělesu *B*.

Rychlost $\dot{p}_a(t_0)$ bodu *p* vzhledem k tělesu *A* v čase t_0 lze vypočítat takto [B97]:

$$\dot{p}_a(t_0) = v_a(t_0) + \omega_a(t_0) \times (p - x_a(t_0)) \tag{29}$$

kde v_a je rychlost tělesa *A*, ω_a je úhlová (rotační) rychlost tělesa *A*, x_a je poloha tělesa *A*. Symbol × určuje vektorový součin.

Protože se terén nepohybuje, je velikost všech jeho bodů nulová a tudíž relativní rychlost v_{trel} tělesa *A* a terénu je dána (všechny ostatní symboly jsou stejné jako v rovnici (28)):

$$v_{trel} = \hat{n}(t_0) \cdot \dot{p}_a(t_0) \tag{30}$$

Pokud v_{trel} je kladná, nastává první případ – těleso se v daném bodě oddaluje od terénu a žádná odezva na kolizi v daném bodě se nepočítá. Pokud v_{trel} je záporná, je nutné vypočítat impuls, který působí jak na terén, čímž vzniká jeho deformace, tak na těleso, které se od terénu následně odrazí. Velikost impulsu J lze určit takto [B97]:

$$J = \frac{-(1+e) \cdot v_{trel}}{\frac{1}{m_a} + \hat{n}(t_0) \cdot \left[I^{-1}(t_0)\left((p - x_a(t_0)) \times \hat{n}(t_0)\right) \times (p - x_a(t_0))\right]} \tag{31}$$

kde v_{trel} je relativní rychlost tělesa A a terénu před výpočtem odezvy kontaktu, m_a je hmotnost tělesa A, I je moment setrvačnosti (tensor) tělesa A, a e je empirická konstanta určující odrazivost tělesa v daném terénu (0 – neodrazí se, 1 – odrazí se bez ztráty energie) a zároveň míru deformace terénu. Všechny ostatní symboly jsou ekvivalentní odpovídajícím symbolům z rovnic (28), (29) a (30).

Pokud je velikost v_{trel} nulová (s určitou numerickou tolerancí), jedná se o případ klidového kontaktu. V tomto případě je nutné určit velikost klidové síly působící jak na terén, tak v opačném směru na těleso. Protože na těleso působí na rozdíl od terénu gravitace, bude klidová síla nenulová. Pro určení její přesné velikosti je nutné znát v daném kontaktním bodě p aktuální relativní zrychlení terénu a tělesa. Protože na terén nepůsobí žádná vnější síla, bude relativní zrychlení tudíž dáno pouze zrychlením tělesa v čase t_0 v daném kontaktním bodě p [B97]:

$$\ddot{p}(t_0) = \omega(t_0) \times (p - x(t_0)) + \omega(t_0) \times (\omega(t_0) \times (p - x(t_0))) + \dot{v}(t_0) \tag{32}$$

kde ω je úhlová rychlost tělesa, x je pozice tělesa a v je rychlost tělesa.

Kontaktní sílu F_r v čase t_0 a bodě p lze pak na základě rovnice (32) a druhého Newtonova zákona vypočítat takto:

$$F_r(t_0) = m_a \ddot{p}(t_0) \tag{33}$$

kde m_a je hmotnost kolidujícího tělesa A.

Pokud existuje při kolizi tělesa s terénem více kontaktních bodů, jsou všechny dílčí impulzy a kontaktní síly (pro jednoduchost a rychlý výpočet) rovnoměrně zmenšeny.

6.4 Deformace terénu explozí výbušniny

Zvláštním případem deformace terénu jsou ty, které vzniknou působením tlakové vlny vycházející z malého prostoru. Jedná se především o explozi bomby na povrchu terénu nebo v určité výšce nad ním. Každou explozi v daném bodě v prostoru následuje vznik masivní tlakové vlny, která se šíří do všech směrů stejnou počáteční rychlostí. Vlivem postupu tlakové vlny z bodu exploze dochází k odstranění velké části zeminy, která je v blízkosti místa exploze – vzniká kráter.

Aby bylo možné určitým způsobem popsat charakter poškození (deformace) blízkého terénu vlivem exploze, zavádí se referenční výbuch o síle jedné tuny trhaviny TNT (trinitrotoluen) [Ham04]. Všechny ostatní velikosti výbuchů lze od této trhaviny odvodit. Mezi důležité parametry výbušniny

patří zejména velikost šokové (shock) vlny v okamžiku vzniku exploze, velikost proudové (blast) vlny a šířka vzniklého kráteru.

Tlak P je definován jako množství síly F působící na jednotku plochy [HRW00]:

$$P = \frac{F}{S} \tag{34}$$

kde S je velikost plochy.

S výhodou lze tlak vyjádřit v jednotkách decibelů a to z důvodu přímé návaznosti na další výpočty v souvislosti s výbušninami [Ham04]:

$$P_{db} = 20\log(P_{Newton}/(2\cdot 10^{-5})) \tag{35}$$

kde P_{db} je tlak vyjádřený v decibelech a P_{Newton} je tlak v jednotkách N/m^2.

Jedna tuna TNT má počáteční šokový tlak o velikosti P_{s1TNT} = 210.6 db a šířka kráteru činí W_{1TNT} = 7.13232 m. Velikost šokové vlny P_{sNTNT} odlišně silné trhaviny, jejíž velikost je N-násobek velikosti jedné tuny TNT, lze vypočítat takto [Ham04]:

$$P_{sNTNT} = P_{s1TNT} + 6.67\log(N) \tag{36}$$

Šířku kráteru W_{NTNT} obecně silné trhaviny o N-násobku velikosti jedné tuny TNT lze od šířky referenční trhaviny W_{1TNT} vypočítat následovně [Ham04]:

$$W_{NTNT} = W_{1TNT} N^{1/3}$$

Výbuch bomby doprovází kromě šokové vlny také vlna proudová (blast), která se projeví zejména u silnějších bomb, které mají velikost šokové vlny vyšší než je 207.46 db. U takto velkých výbušnin je velikost proudové vlny větší než je velikost samotné šokové vlny, proto je vhodné ji pro simulaci explozí brát v potaz. Velikost proudové vlny P_Q se odvodí od šokové vlny $P_{sNewton}$ zadané v jednotkách N/m^2 následovně [Ham04]:

$$P_Q = 2.5\cdot P_{sNewton}^2/(7\cdot P_{atm} + P_{sNewton}) \tag{37}$$

kde P_{atm} je velikost atmosférického tlaku (P_{atm} = 101325 N/m^2).

Celkový tlak P_{total} vzniklý v místě výbuchu bezpotřebně po explozi je dán součtem šokového tlaku P_s a proudového tlaku P_Q:

$$P_{total} = P_s + P_Q \tag{38}$$

Velikost tlaku P_x v jiném místě, než je epicentrum výbuchu, klesá s třetí mocninou vzdálenosti x [Ham04]:

$$P_x = P_{total}/(\frac{4}{3}\pi\cdot x^3) \tag{39}$$

Podle definice tlaku rovnicí (34) a výpočtu hodnoty tlaku v místě vzdáleném o x metrů od epicentra výbuchu daném rovnicí (39), lze ke každé částici i vypočítat aktuální tlakovou sílu F_{Pi} působící na danou částici bezprostředně po explozi:

$$F_{Pi} = P_{total} / (\frac{4}{3}\pi \cdot x^3) \cdot (\frac{\pi \cdot W^2}{4C}) \qquad (40)$$

kde P_{total} je celkový tlak v epicentru výbuchu bezprostředně po explozi, W je šířka kráteru a C je počet částic povrchového částicového systému. Nejpravější část rovnice (40) představuje plochu, která připadá na jednu částici v rámci povrchu terénu před vznikem kráteru.

6.5 Optimalizace částicového systému

Jak je uvedeno v kapitole 4.5, Volně vázané částicové systémy, kde jsou popsány vlastnosti částicového systému při použití potenciální energie pro definici mezičásticových sil, je možné ignorovat síly vzdálenějších částic. Toto je klíčové pro efektivitu simulace. Proto je v [Ton98] použita efektivní datová struktura u částicových systémů pro výpočet nejbližších sousedů k dané částici *i*.

Touto datovou strukturou je struktura kd-tree, která hierarchicky dělí prostor v daných bodech. Tím se dynamicky vytvoří stromová struktura obsahující ve všech svých uzlech právě jeden prostorový bod. Sestavení kd-tree struktury trvá $O(n \cdot \log(n))$ času [Kdt06], kde n je počet bodů, a je časově nejnáročnější operací, která je potřebná z hlediska použití pro částicové systémy.

Druhou operací je vyhledávání nejbližších sousedů k danému bodu. Průměrná doba vyhledávání je $O(F+\log(n))$, kde F je počet vrácených nejbližších bodů k danému bodu a n je počet všech bodů. Pro výpočet mezičásticových sil je potřeba najít nejbližší body ke všem částicím, což trvá $O(n \cdot \log(n))$ času při zanedbání počtu vrácených bodů. V porovnání s výpočtem sil mezi všemi možnými páry částic, kde je doba výpočtu rovna $O(n^2)$, se jedná teoreticky o velkou časovou úsporu.

Obecně ovšem při relativně malém počtu částic je složitost díky režii při manipulaci s datovou strukturou výhodnější použít raději metodu silou a projít všechny páry částic. Navíc pokud dochází k deformacím je potřeba poměrně často kd-tree opětovně sestavovat. Analýza efektivity a vhodnosti použití struktury kd-tree pro povrchové částicové systémy je uvedena v kapitole 8.2, Efektivita struktury kd-tree.

7 Realizace

Grafický toolkit OpenSceneGraph poskytuje řadu užitečných funkcí a nástrojů pro efektivní organizaci a vizualizaci obrazových dat. Tyto podpůrné mechanismy jsou primárně určeny zejména pro práci s rozsáhlou scénou sestávající se z mnoha grafických objektů. Může to být podpora level of detail (LOD) – vykreslování různé úrovně detailů, efektivní prořezávání grafu scény nebo podpora vytváření animací.

Z hlediska modifikace terénu, kdy je potřeba efektivně počítat detekci kolizí s terénem a její odezvu, není nástroj OpenSceneGraph příliš tímto směrem vybaven. Existuje zde sice základní podpora pro detekci kolizí ve formě průsečíku paprsku s grafickým objektem, ta ovšem není vhodná pro řešený problém, a proto je nutné podporu vhodné detekce kolizí do OpenSceneGraphu začlenit.

7.1 Integrace do toolkitu OpenSceneGraph

Protože deformace terénu probíhá v reálném čase za běhu programu, je software pro deformaci terénu (s názvem osgTDM) implementován jako node-kit, tedy dynamická knihovna pracující nad uzly daného grafu scény. Koncept komunikace node-kitu osgTDM s grafickou scénou vychází z toho, aby bylo možné kdykoliv během chodu aplikace přidávat a odebírat tělesa interagující s terénem.

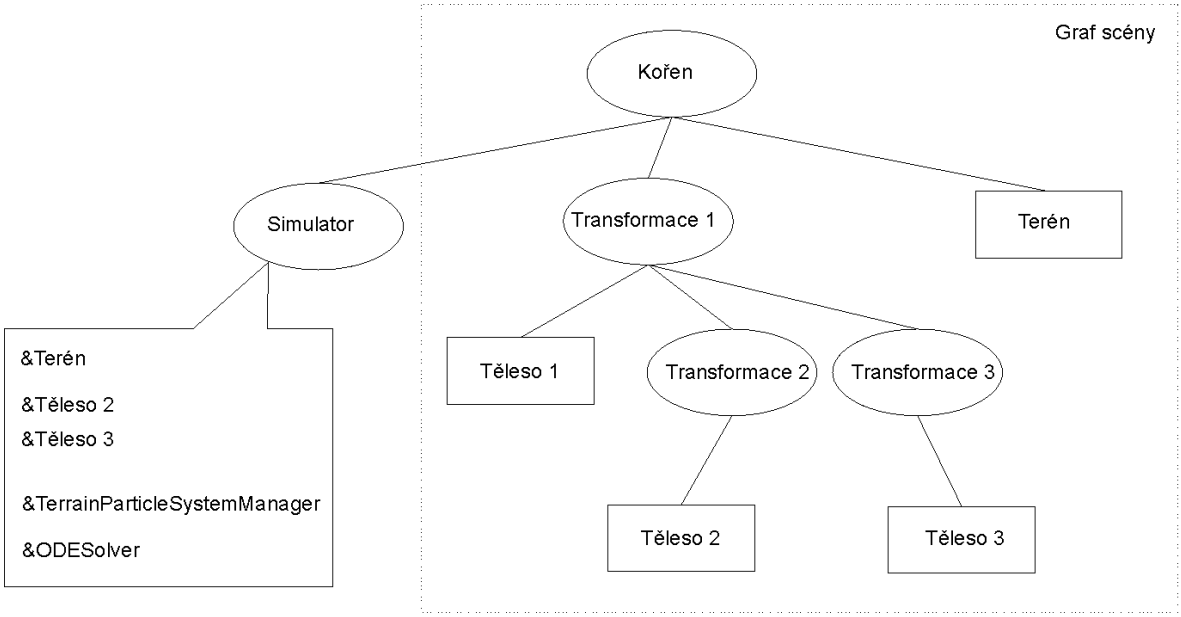
Hlavní objekt, Simulator, node-kitu osgTDM je proto přidán do scény jako jeden z uzlů grafu a je tudíž aktivován v každém snímku, kdy se překresluje scéna. Z důvodu toho, že dochází při výpočtu ke změnám geometrie terénu, je nutné dávat hlavní objekt jako prvního potomka kořene stromu. Po aktivaci spočítá hlavní objekt všechny nové změny ve scéně a v případně nutnosti aktualizuje geometrii terénu.

Objekt Simulator obsahuje odkaz na podstrom scény reprezentující terén a seznam těles, která mohou přijít s terénem do kontaktu. Do tohoto seznamu lze kdykoliv během chodu aplikace přidávat nebo odebírat tělesa. U všech těles se předpokládá definice základních fyzikálních vlastností jako je hmotnost, moment setrvačnosti (tensor) nebo koeficient odrazivosti, aby bylo možné vypočítat případné impulsy či kontaktní síly v případě interakce s terénem jak je popsáno v kapitole 6.3, Deformace pohybem po terénu. Speciálními tělesy jsou tělesa výbušná, která jsou odvozená od obecných těles, mají ovšem definovánu výbušnou sílu v jednotkách tun TNT. Tato tělesa zanikají po své explozi a jsou poté implicitně odstraněna ze seznamu těles. Příklad začlenění objektu Simulator do grafu scény je znázorněn na Obr. 13.

Aby bylo možné měnit během simulace polohu a orientaci těles umístěných v seznamu objektu Simulator, je nutné definovat jako předchůdce každého takového tělesa transformační maticový uzel (osg::MatrixTransform), který obsahuje transformační matici pro své potomky. Simulator potom dynamicky mění tyto transformační matice těles a tím i jejich polohu a orientaci ve scéně.

Pokud dojde k interakci libovolného tělesa s terénem, vytvoří se dynamicky částicový systém, který kopíruje povrch terénu. Velikost tohoto částicového systému je odvozena od rozměrů kontaktního tělesa s terénem v případě obyčejných těles, nebo od síly výbušniny v případě výbušných těles. Každý částicový systém také automaticky zaniká v případě, že se těleso přestane pohybovat

nebo dojde k rozplynutí tlakové vlny po explozi. O chování jednotlivých částicových systémů a tím i o deformaci terénu se stará objekt zvaný TerrainParticleSystemManager.



Obr. 13: Příklad začlenění knihovny osgTDM do scény toolkitu OpenSceneGraph. Objekt Simulator je umístěn jako první potomek kořene grafu scény a obsahuje ukazatel na terén. Dále má v seznamu uloženy ukazatele na Těleso 2 a Těleso 3, které mohou interagovat s terénem, a obsahuje také ukazatele na manažera č. systémů a ukazatele na objekt řešící ODE rovnice.

Objekt Simulator obsahuje také odkaz na objekt odvozený z abstraktní třídy ODESolver, který řeší obyčejné diferenciální rovnice. V současné době lze vybírat z Eulerovy integrační metody, metody Runge-Kutta druhého řádu a metody Runge-Kutta čtvrtého řádu. Pro použití integrátoru k řešení soustavy obyčejných diferenciálních rovnic, je nutné specifikovat objekt, jenž implementuje funkci, která vrátí derivace v libovolném čase k aktuálním stavům. Jinými slovy tato funkce vypočítá aktuální hodnoty působících sil a momentů a velikosti rychlostí a úhlových rychlostí těles. Tuto funkci implementuje jak objekt Simulator, pro pohyb těles v prostoru, tak objekt TerrainParticleSystemManager, pro dynamiku (deformace) částicových systémů.

7.2 Speciální datové struktury

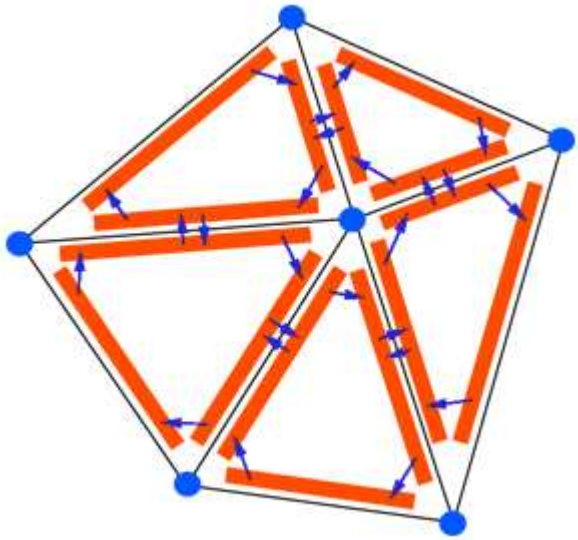
Pro efektivní výpočet detekce kolizí a jejich odezvy je implementován systém pro detekci kolizí popsaný v kapitole 6.2.3, Detekce kolizí s terénem. Jedná se o datovou strukturu, která obsahuje strom terénu, v jehož listech jsou obsaženy jednotlivé regiony terénu. Tyto regiony se odvozují od počátečního stromu terénu definovaného jako podstrom scény, kdy jednotlivé geometrie terénu v listech tohoto stromu byly označeny jako regiony a následně byly obaleny datovou strukturou AABB stromem s využitím knihovny Gimpact pro detekci kolizí [Leo06].

AABB strom byl vybrán, protože se lépe hodí pro neformovatelná tělesa než jeho konkurent, OBB strom. Pokud dojde k deformaci terénu, a tím pádem také ke změně trojúhelníkové sítě jednoho nebo více regionů terénu, je nutné přepočítat u každého regionu jeho AABB strom a tím pádem

přepočítat také celkovou strukturu celého terénu (strom s AABB stromy ve svých listech). Zde se ovšem odkrývá prostor pro optimalizaci kódu, protože není vždy nutné měnit struktury AABB stromů, protože změny geometrie terénu jsou v drtivé většině lokální.

V části výpočtu, kdy dochází k dynamickému vytváření povrchových částicových systémů, je potřeba efektivně určovat nejbližší okolní vrcholy kolidujících trojúhelníků terénu. K tomuto účelu je potřeba svázat okolní elementy vhodnými vazbami tak, aby bylo možné v případě změny geometrie trojúhelníkové sítě a to včetně případu vzniku nových vrcholů, efektivně tyto vazby aktualizovat. Protože je terén vyjádřen sítí trojúhelníků a zároveň se jedná o tzv. manifold povrch, kdy každá hrana polygonu je hranou nanejvýše dvou polygonů, je s výhodou využita struktura half-edge pro popis struktury a topologie terénu.

Struktura half-edge umožňuje efektivní prohledávání okolní k libovolnému druhu grafického elementu, ať už se jedná o bod, hranu nebo polygon. Navíc je tato struktura oproti jiným, např. struktuře okřídlená hrana (winged edge), paměťově méně náročná. Její struktura je uvedena na Obr. 14.



Obr. 14: Grafická reprezentace datové struktury half-edge pro síť trojúhelníků [McG00]. Modré body značí vrcholy trojúhelníků, červené obdélníky jsou jednotlivé orientované hrany a modré šipky reprezentují ukazatele.

Každá hrana ve struktuře half-edge je reprezentována dvojicí orientovaných hran, z nichž každá ukazuje opačným směrem, a obě obsahují ukazatele na tu druhou – své dvojče. Každá z těchto orientovaných hran obsahuje také ukazatel na svůj počáteční vrchol, ovšem ne na svůj koncový. Pozici koncového bodu k dané orientované hraně lze zjistit přes ukazatel na její dvojče, jejíž počáteční vrchol je zároveň koncovým vrcholem tohoto dvojčete.

Orientovaná hrana také obsahuje ukazatel na „svůj“ trojúhelník (polygon) a na orientovanou hranu, která je zároveň druhou hranou tohoto trojúhelníka. Pro zpětnou vazbu z trojúhelníků na orientované hrany existuje ještě ukazatel z každého trojúhelníka na libovolnou „svou“ orientovanou hranu. Směr všech orientovaných hran musí být v rámci jednoho modelu stejný, je ovšem lhostejné zda je proti směru hodinových ručiček či opačný.

V případě knihovny osgTDM je struktura half-edge doplněna o nekonečný polygon, na který ukazují orientované hrany na hranicích terénu. Použitím nekonečného polygonu se lze vyhnout ošetřování okrajů sítě trojúhelníků a tím zefektivnit tvorbu a úpravu této sítě za chodu programu.

Použití dvojích struktur na popis terénu by se mohlo zdát neefektivní a to zejména z důvodu paměťových nároků. Toto je ovšem v případě potřeby rychlých výpočtů, což je případ real-time simulace deformace terénu, akceptovatelné. Navíc v dnešní době není použití větší systémové paměti zase až takový problém, což o výpočetním výkonu potřebného pro účely deformace terénu v reálném čase říci nelze.

7.3 Algoritmus

V této chvíli je již definováno vše potřebné pro uvedení základního algoritmu. Na Obr. 15 je uveden pseudokód základního algoritmu knihovny osgTDM.

```
inicializace
  vytvoření hierarchických obalových struktur
  vytvoření struktury half-edge pro terén
  čas = 0
  spuštění simulace
simulační smyčka (vstup: krok)
  detekce kolizí
    provedení box prunning
    určení kolidujících párů regionTerénu-těleso
    určení kolizních bodů u každého páru
  odezva kolize
    určení kolidujících a klidových bodů
    výpočet impulsů v kolidujících bodech
    výpočet kontaktních sil v klidových bodech
    určení tlakové velikosti tlakové vlny u explozí
    určení velikosti kráteru u explozí
  zpracování povrchových částicových systémů
    vytvoření nových případných č. systémů
    určení tělesy ovlivněných částic každého č. systému
  aktualizace (integrace) částicových systémů o krok
  aktualizace (integrace) těles o krok
  případná aktualizace geometrie a obalové hierarchie terénu
  čas += krok
```

Obr. 15: Pseudokód základního algoritmu knihovny osgTDM

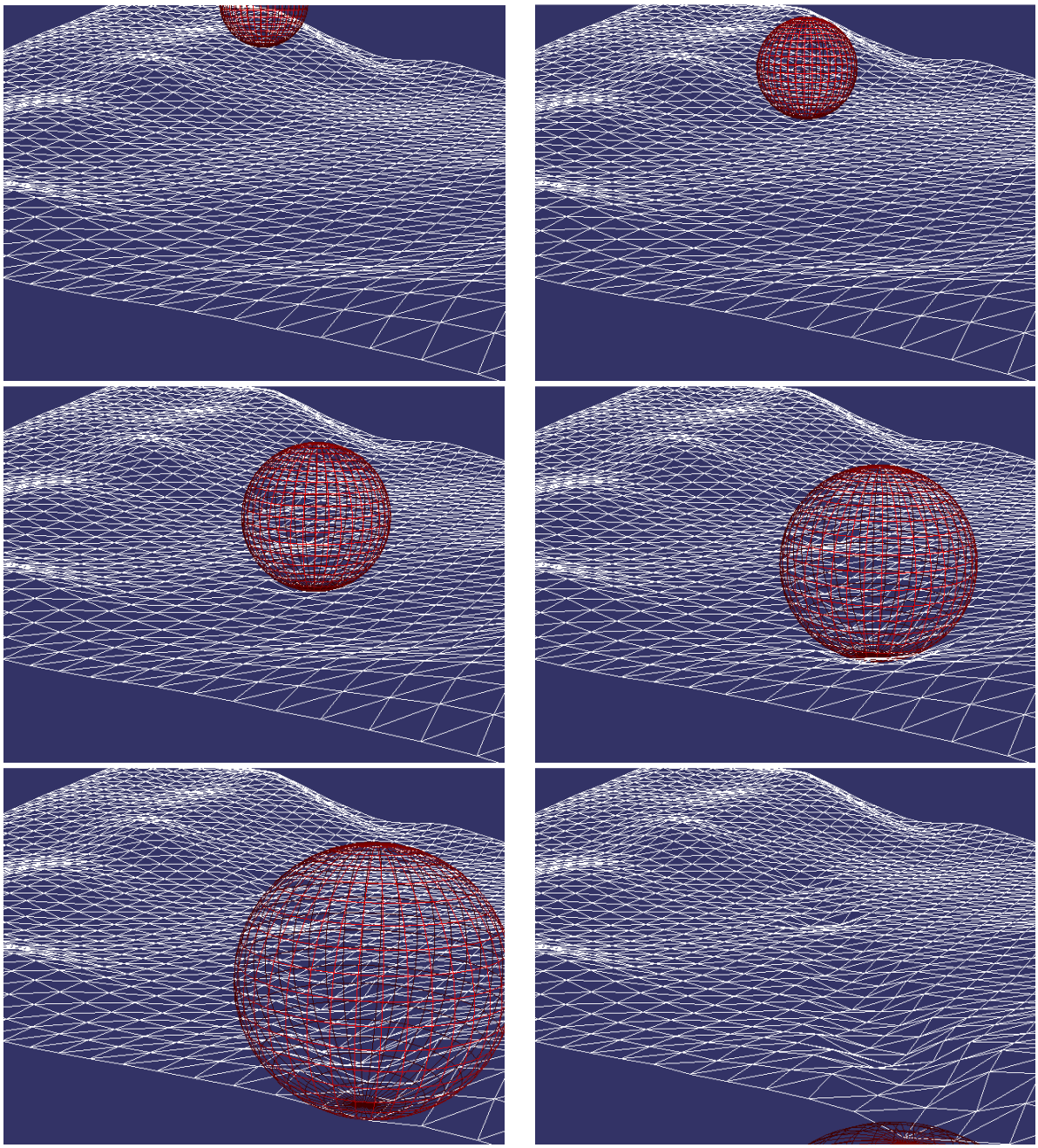
Nejprve je provedena inicializace, kdy se vytvoří všechny pomocné datové struktury pro všechna tělesa a terén, potřebné pro výpočty během simulace. Dále se inicializuje čas na počáteční hodnotu a spouští se simulační cyklus. Tato fáze algoritmu je vzhledem k sestavování potřebných struktur tou časově nejnáročnější.

Vlastní simulace probíhá v rámci smyčky, která v každé iteraci aktualizuje scénu a simulační čas. Protože se jedná o real-time simulaci, simulační čas se rovná reálnému, resp. simulační krok má stejnou délku jako doba uplynulá v reálném čase. V rámci jedné smyčky dochází k jevům popsaným

v dřívějších kapitolách – detekce kolizí, odezva kolize, zpracování částicových systémů, integrace a aktualizace geometrie a struktur terénu.

8 Výsledky a analýzy

Pro demonstraci a testování zvoleného způsobu deformace terénu, byl vytvořen vedle samotné knihovny osgTDM také program osgTDMdemo. Jedná se o konzolovou aplikaci, která používá k vizualizaci a manipulaci se scénou knihovnu osgProducer, jež je součástí jádra toolkitu OpenSceneGraph. Jako parametr programu se udává scéna ve formátu OSG, která představuje terén scény. Jednotlivá tělesa ve scéně jsou generována automaticky během chodu programu.

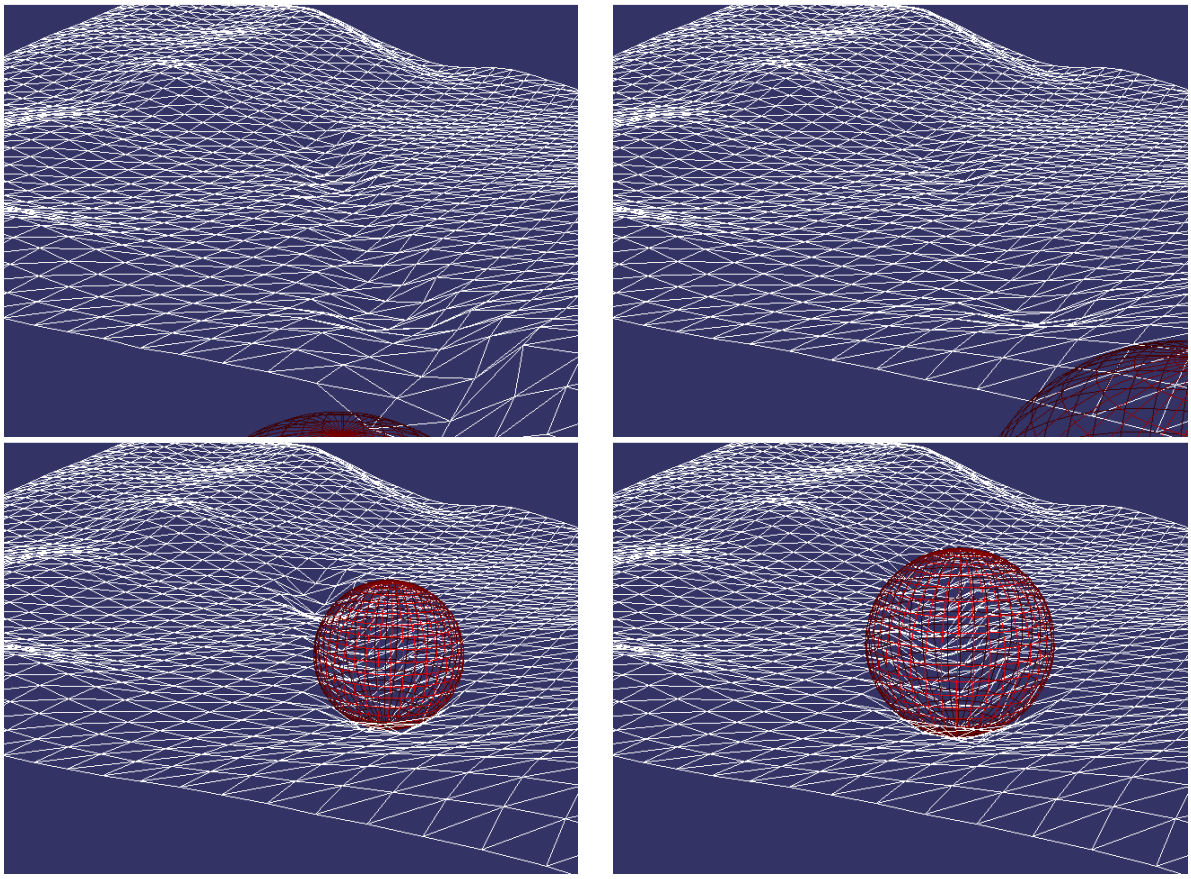


Obr. 16: Průběh simulace deformace terénu pohybem koule

První simulace je scéna s koulí padající na terén z určité výšky a mající nenulovou počáteční rychlost. Výsledkem simulace je stopa (rýha) v terénu, kterou v něm zanechala pohybující se koule. V modelu deformace terénu nebylo zahrnuto tření, proto nedochází k rotaci koule.

Na Obr. 16 jsou zobrazeny screenshoty z průběhu první simulace. Terén je v tomto případě složen z 2808 trojúhelníků, koule má průměr 75m a hmotnost koule je 70kg. Částicový systém vznikající při nárazu má následující parametry: disociační energie je nastavena na 300J, parametry *m* a *n* Lennard-Jonesovy potenciální funkce mají hodnotu 2 a 4. Klidová vzdálenost mezi částicemi je dynamicky odvozována podle aktuálního rozmístění částic. Pro integraci byla zvolena Eulerova integrační metoda, přičemž simulace běžela v reálném čase s velikostí integračního kroku danou frekvencí překreslování, což bylo 1/60s.

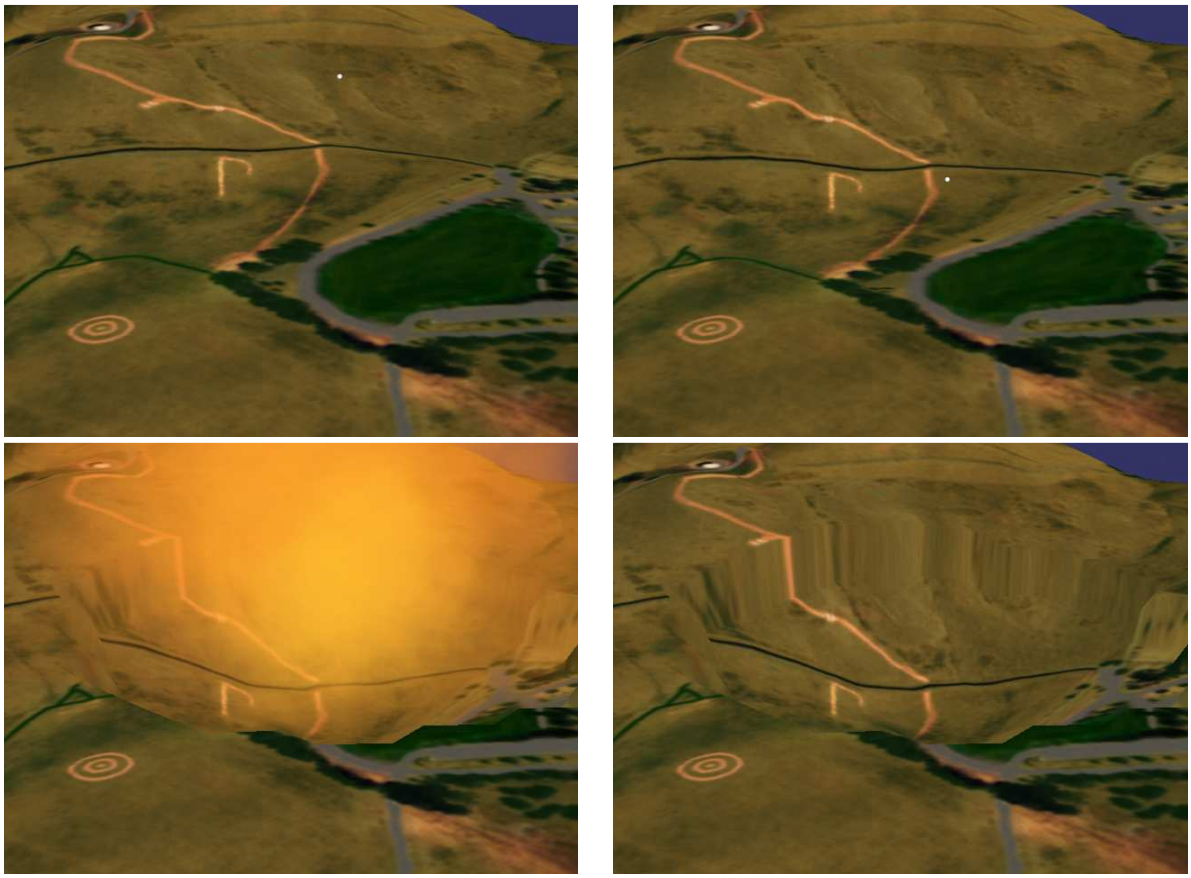
Úkolem prvních simulací bylo ověřit schopnost navrženého způsobu řešení simulovat zanechávání stop (např. v blátivém terénu, v poli s plodinami apod.) za pohybujícím se tělesem. Během provádění dalších testů byly průběžně měněny parametry částicového systému, čímž se dosáhlo odlišného způsobu deformací – hlubší či méně hluboká rýha apod. Hloubka rýhy se mění také v závislosti na velikosti povrchu styčné plochy tělesa s terénem a na hmotnosti tělesa, přičemž má vliv na dráhu po terénu pohybujícího se tělesa. Čím větší hloubka, tím větší zpomalení tělesa a naopak.



Obr. 17: Ukázky výsledků simulací pro různá nastavení parametrů modelu. Vlevo nahoře (referenční hodnoty) byly nastaveny parametry částicového systému na e_0=70, m=2, n=4, hmotnost koule m_s=70kg a poloměr koule r_s=75m. Vpravo nahoře byl změněn viskózní parametr z 0.1 na 0.2. Vlevo dole byl zmenšen poloměr koule na 60m. Vpravo dole byla zvětšena hmotnost koule na 150kg.

Případy výsledků po změně parametrů simulace jsou uvedeny na Obr. 17. Výsledek simulace zobrazený vlevo nahoře je nominálním výsledkem, který slouží jako vzorový. V tomto případě by se mohlo jednat o pohyb tělesa po bahnitém terénu, kde vznikají hluboké stopy. Druhý případ vpravo nahoře mění terén na méně deformovatelný, např. na vyschlejší půdu. Vlevo dole se zvětšila hustota koule, vlastnosti terénu jsou stejné jako v nominálním případě. Zde došlo k zaboření tělesa do terénu a těleso zůstalo stát. Obdobný případ je i v poslední ukázce výsledku simulace na obrázku vpravo dole, kdy byla zvětšena hmotnost koule při zachování poloměru. Těleso se v tomto případě také zarylo do země, ovšem směr jeho dráhy zůstal díky větší hybnosti přímější.

Druhá simulace reprezentuje případ, kdy na terén dopadá bomba, která exploduje v čase kontaktu s povrchem. Sekvence snímků na Obr. 18 ukazuje průběh simulace, kde svržená bomba ve tvaru koule má sílu šestnácti kilotun TNT.



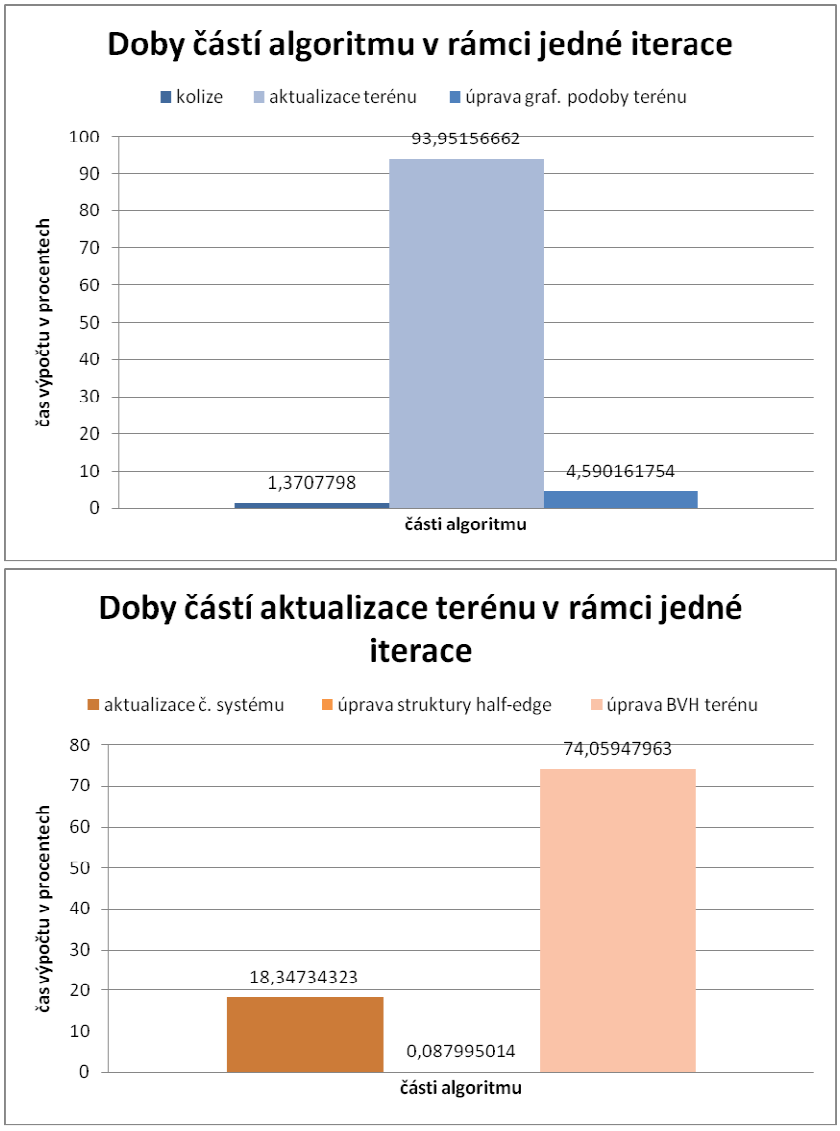
Obr. 18: Průběh simulace výbuchu bomby o síle šestnácti kilotun TNT.

Parametry částicového systému jsou nastaveny takto: disociační energie e_0=200J, m=2, n=4. Velikost kráteru a síly působící na částicový systém byly vypočítány podle postupu napsaném v kapitole 6.4, Deformace terénu explozí výbušniny. Bylo ovšem nutné mírně uzpůsobit šíření tlakové vlny velikosti kráteru tak, aby hloubka odpovídala zhruba jedné sedmině šířky kráteru [Ham04].

8.1 Časová analýza algoritmu

Implementovaný algoritmus knihovny osgTDM není zcela optimalizován. Aby bylo možné zvýšit efektivitu výpočtu během simulace, je třeba najít úzká místa algoritmu, která mají nejdelší výpočetní čas.

Jednu iteraci algoritmu knihovny osgTDM lze rozdělit na tři části. První část se zabývá kolizemi, tj. řeší detekci kolizí a jejich odezvu mezi pohybujícími se tělesy s terénem. Druhá část algoritmu upravuje terén. To zahrnuje výpočet nových poloh částic u č. systému, aktualizaci struktury half-edge pro terén a aktualizaci objemové hierarchické struktury (BVH) terénu potřebné pro detekci kolizí. Třetí částí je aktualizace grafické reprezentace terénu, tedy aktualizace informací v grafu scény toolkitu OpenSceneGraph.



Obr. 19: Podíl doby výpočtů jednotlivých kroků algoritmu v rámci jedné iterace.

Na základě tohoto rozdělení byly provedeny testy s deseti různými scénami, kde bylo těleso ve tvaru koule, podobně jako je tomu na Obr. 16, vrženo nenulovou rychlostí na povrch terénu. Ve všech scénách testu měla koule stejný poloměr r=75m, aby vznikající částicový systém měl opakovaně stejný počet částic. Velikost integračního kroku byla nastavena na pevnou hodnotu 1/60s. Před začátkem každého měření byly měněny počáteční rychlost koule, směr pohybu koule a hmotnost koule. Všechny simulace končily buď úplným zastavením koule, nebo v okamžiku, kdy koule opustila terén. V průběhu simulace byly měřeny v rámci jedné iterace výše popsané části algoritmu, jež byly v průběhu algoritmu akumulovány pouze v případě, že byla provedena také aktualizace částicového systému. Jinými slovy, měřilo se pouze v případech, kdy se plně počítaly všechny části algoritmu (neměřilo se například, když nebyla koule ve styku s terénem).

Na Obr. 19 jsou zobrazeny grafy znázorňující výše zmíněné části algoritmu. Na prvním grafu je zobrazen podíl výpočtů detekce kolize a její odezvy, aktualizace terénu a aktualizace grafické reprezentace terénu. Z grafu je jasně patrné, že aktualizace terénu potřebuje na svůj výpočet nejvíce času (více jak 90%). Proto byla u této části algoritmu provedena hlubší analýza, jejíž výsledek znázorňuje druhý graf na Obr. 19.

Dříve než bude okomentován tento graf, je nutné uvést podrobnější informace o testovaných scénách. Celý terén byl tvořen jedním regionem o počtu 2808 trojúhelníků a 1443 vrcholech. To znamená, že hierarchická struktura terénu byla tvořena pouze jedním AABB stromem. Dále počet částic částicového systému byl nastaven v každé scéně na 98, aby nedošlo k ovlivnění doby výpočtu algoritmu jinými počty částic.

Aktualizace samotného částicového systému trvá přibližně 20% celkové doby potřebné na výpočet v rámci jedné iterace. Aktualizace datové struktury half-edge je díky přímému spojení s částicovým systémem zanedbatelná (bylo naměřeno 0.09% doby iterace). Největší podíl na výpočetní čas má poslední část a tou je úprava objemové hierarchie terénu. To je zapříčiněno tím, že se po deformaci aktualizuje celá hierarchie AABB stromu všech změněných regionů, což je ovšem v tomto případě celý terén.

8.2 Efektivita struktury kd-tree

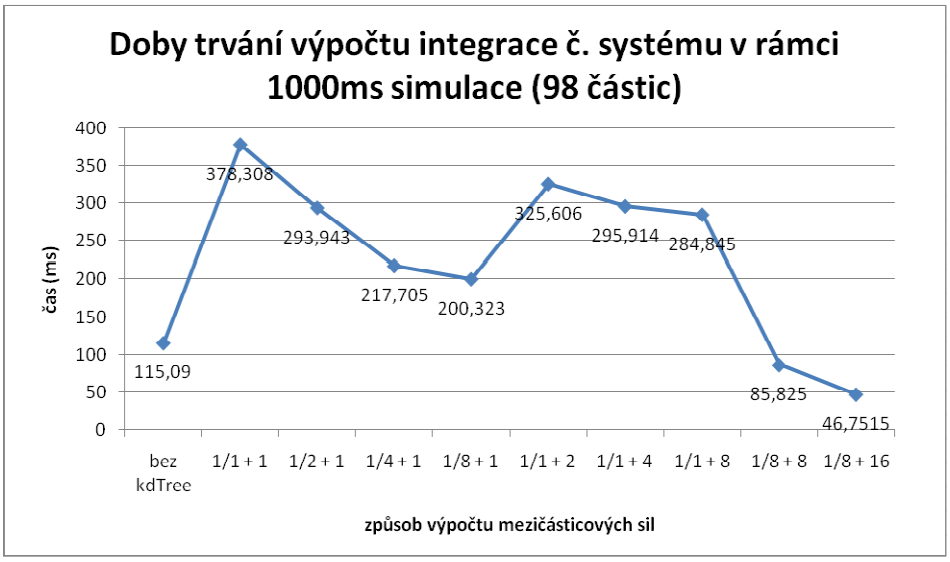
Pro optimalizaci výpočtu sil mezi částicemi lze použít pomocné datové struktury kd-tree, která efektivně počítá nejbližší body v prostoru k zadaným souřadnicím. Struktura kd-tree byla úspěšně aplikována pro částicový systém v [Ton98], kde se částicový systém skládal z předem daného počtu částic. Ovšem v případě deformace terénu pohybujícím se tělesem se polohy a počet částic mění v čase, a proto je důležité zjistit, zda je struktura kd-tree vhodná pro tento nový typ částicového systému. V případě deformace terénu explozí výbušniny nemá smysl díky krátkému trvání částicového systému kd-tree použít, proto se budou zkoumat pouze případy částicových systémů u deformace terénu pohybem tělesa.

Pro účely vyhodnocení vhodnosti struktury kd-tree pro algoritmus knihovny osgTDM byly testovány scény s různě se pohybujícím tělesem po terénu. Částicový systém vzniklý tímto pohybem měl nastaven ve scénách různý počet částic a těleso mělo pokaždé jinou hustotu. Krok simulace byl nastaven na pevnou hodnotu 1/60s a bylo zvoleno Eulerovo integrační schéma. Všechna měření

probíhala pouze v době, kdy bylo těleso v kontaktu s terénem. Struktura kd-tree byla reprezentována odpovídající třídou programové knihovny ANN [MA06].

V prvním testu byly měřeny doby integrace částicového systému bez použití a s použitím struktury kd-tree, u níž byly testovány různé způsoby a stupně optimalizace. Kd-tree umožňuje v zásadě optimalizaci dvojího druhu. Zaprvé to je již zmíněný způsob ignorování vzdálených částic a tím zmenšení počtu výpočtů mezičásticových sil, což vede teoreticky ke zmenšení výpočetních nároků. Druhou optimalizací je omezení počtu přepočítávání struktury kd-tree v čase, tzn. konstruovat kd-tree jen v určitých iteracích algoritmu (např. každou druhou iteraci), čímž dochází k nemalé časové úspoře.

Oba způsoby optimalizace se navzájem nevylučují, a proto je lze použít zároveň. Oba způsoby urychlení výpočtu mají ovšem své hranice, protože u obou dvou dochází k narůstání chyb ve výpočtu tím, čím je optimalizace větší. Tyto chyby se dají poměrně snadno kompenzovat úpravou v nastavení globálních a viskózních tlumících koeficientů u částicových systémů.



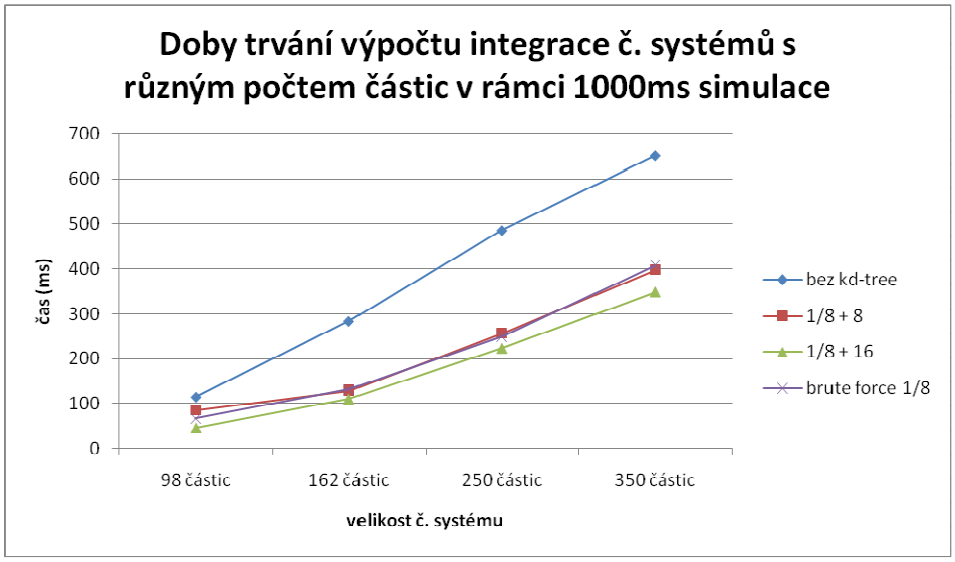
Obr. 20: Výsledky měření doby integrace č. systému. První člen (zlomek) v popisu způsobu výpočtu značí poměr všech částic k nejbližším částicím (sousedům), které byly k jednotlivým částicím vypočteny. Druhý člen značí frekvenci výpočtu nové struktury kd-tree.

Různé kombinace výše popsaných optimalizací byly zavedeny i v simulačních testech. Výsledky prvního testu znázorňuje Obr. 20. Použití samotné struktury kd-tree pro výpočet nejbližších sousedů, jejichž počet se rovná počtu částic, způsobil více jak třínásobný nárůst doby výpočtu algoritmu oproti případu bez použití kd-tree (378,308ms × 115,09ms). Optimalizací výpočtu nejbližších sousedů na 1/8 počtu částic došlo k urychlení téměř o polovinu oproti výpočtu všech sousedů (200,323ms). Opětovným konstruováním struktury kd-tree v každé osmé iteraci s výpočtem sousedů o počtu rovném počtu částic klesla výpočetní doba z 378,308ms na 284,845ms. Když se obě optimalizace zkombinovaly, byla výsledná doba rovna 85,825ms, což je zmenšení doby výpočtu oproti případu bez kd-tree na necelých 75%. Když se navíc prodloužila doba sestavení kd-tree na šestnáct iterací, došlo k časové úspoře na necelých 41%, což je nezanedbatelná hodnota.

Z provedeného testu je zřejmé, že správné použití struktury kd-tree je i v případě měnícího se částicového systému výhodné. Výjimkou je pohybující se těleso, jež má vysoký poměr rychlosti

k velikosti částicového systému, což ovšem není případ reálně se pohybujících těles (vozidel) po terénu. Nabízí se ovšem otázka, zda není výhodnější prozkoumat vzdálenosti všech částic a na jejich základě vyloučit ty vzdálenější. Výhodou tohoto přístupu je fakt, že není potřeba uchovávat v paměti zvláštní datovou strukturu, jejíž sestavení navíc netrvá nezanedbatelnou dobu.

Pro zodpovězení této otázky byl proveden další test, ve kterém byly testovány částicové systémy o různém počtu částic s a bez použití kd-tree a s použitím prostého ignorování vzdálenějších částic podle vypočtené vzdálenosti. Na Obr. 21 je zobrazen graf s výsledky simulačních testů, z něhož vyplývá, že použití optimalizací u různých částicových systémů s různým počtem částic vedlo ke snížení výpočetního času integrace minimálně na polovinu. Vítězem testu se stal způsob s použitím kd-tree ve variantě 1/8 + 16 následovaný metodou s jednoduchým ignorováním stejného počtu (1/8) vzdálených částic. Z výsledků testu lze odvodit závěr, že použít kd-tree má smysl i pro případ pohybujícího se částicového systému po povrchu terénu.



Obr. 21: Výsledky testů integrace č. systému při aplikaci různých druhů optimalizací. Způsob bez kd-tree neobsahuje optimalizace. Způsob označený 1/8 + 8 resp. 1/8 + 16 značí optimalizace pomocí kd-tree s počtem sousedů rovným jedné osmině a opětovným sestavením kd-tree každou osmou iteraci resp. s počtem sousedů rovným jedné osmině a opětovným sestavením kd-tree každou šestnáctou iteraci. Způsob označený brute force 1/8 vypočítá osminu sousedů při výpočtu vzdáleností mezi všemi páry částic.

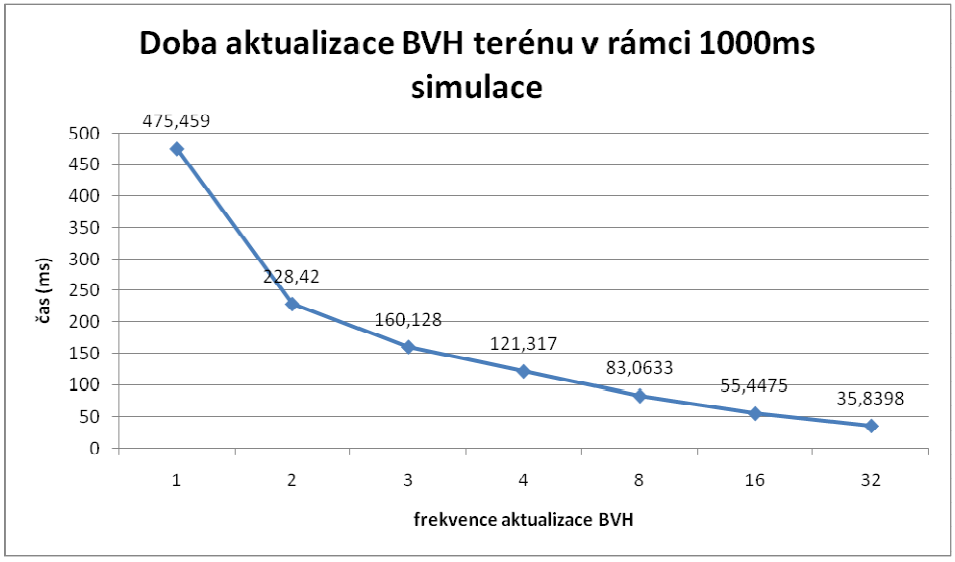
8.3 Optimalizace algoritmu

Vedle optimalizace výpočtu mezičásticových sil je vhodné pro efektivitu algoritmu optimalizovat také sestavení BVH terénu, což vyplývá z kapitoly 8.1, Časová analýza algoritmu. Protože detekce kolizí se provádí pomocí BVH terénu, je potřeba pro korektní výpočty přepočítávat BVH terénu po každé jeho deformaci. To pro případ pohybujícího se tělesa po terénu znamená aktualizovat hierarchii terénu v každé iteraci algoritmu, což vede ke zbytečně velkému zatížení algoritmu.

Řešením je buď rozdělit terén na menší regiony, což může být z hlediska vytváření terénu nepraktické, nebo nechat vypočítat BVH po uběhnutí určitého časového intervalu nebo po několika iteracích. Detekci kolizí je potom nutné přenechat např. pro strukturu half-edge. To může být na první

pohled málo efektivní, protože tato struktura nepoužívá žádné dělení prostoru a je obecně nevhodná pro detekování kolizí. Není to ale úplně pravda, protože pro pohybující se těleso jsou známy kontaktní trojúhelníky terénu i směr pohybu tělesa, což lze s výhodou použít pro podstatné omezení počtu potenciálních kolidujících trojúhelníků. Navíc aktualizace struktury half-edge trvá zanedbatelnou dobu oproti opětovnému sestavení BVH terénu.

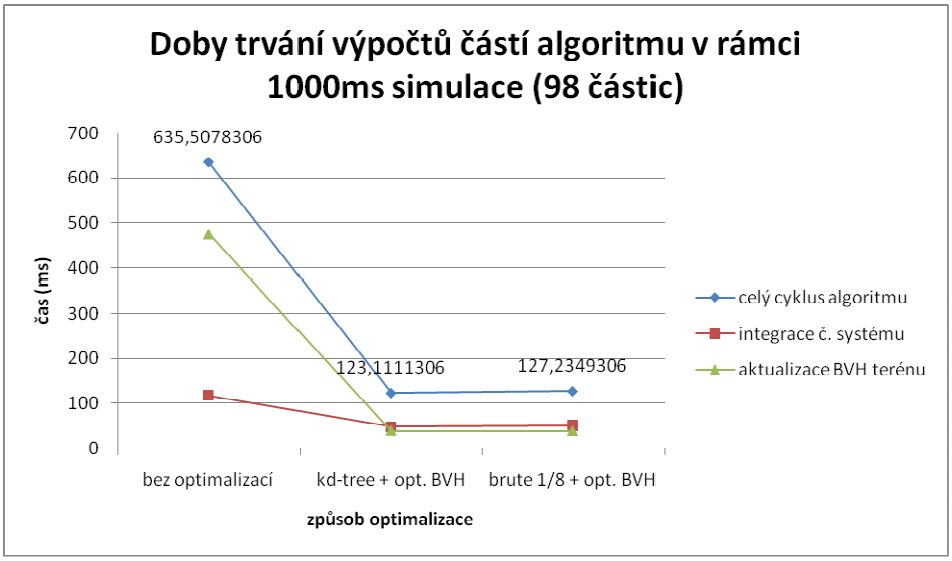
Pro potvrzení těchto předpokladů byly provedeny testy pro pohybující se těleso s různou časovou prodlevou mezi aktualizacemi hierarchií terénu. Na Obr. 22 jsou znázorněny výsledky testu, které dopadly podle očekávání. Tedy podíl doby výpočtu BVH se na jednu sekundu algoritmu s prodloužením doby aktualizace BVH úměrně snižuje.



Obr. 22: Výsledky testu aktualizace BVH terénu po různém počtu iterací algoritmu.

Z výsledků testů optimalizace výpočtu sil mezi částicemi v kapitole 8.2, Efektivita struktury kd-tree, a výše provedených testů optimalizací sestavení BVH terénu, je nasnadě tyto dva druhy optimalizací zkombinovat a vyčíslit konečnou míru zefektivnění algoritmu.

Byly proto provedeny testy scén, jejichž výsledky jsou zobrazeny na Obr. 23, s různými kombinacemi výše uvedených optimalizací. Důležitá je zejména křivka pro celý cyklus algoritmu, která ukazuje, kolik času procesoru bylo potřeba pro chod celé simulace v rámci jedné sekundy simulace. I v případě neoptimalizovaného algoritmu je možné simulovat scénu s jedním částicovým systémem v reálném čase. Pokud se ovšem přidá byť jen jeden další stejně velký částicový systém, už by nebylo možné provádět výpočet simulace na stejném počítači v reálném čase (s nastaveným integračním krokem na 1/60s).



Obr. 23: Výsledky celkové optimalizace algoritmu.

Oproti tomu v případech použití optimalizace, ať už pomocí kd-tree nebo prostým ignorováním vzdálených částic, je možné provést simulaci v reálném čase s až osmi stejně velkými částicovými systémy zároveň.

9 Závěr

Byl navržen a implementován originální způsob tvorby změn terénu v reálném čase s ohledem na toolkit OpenSceneGraph. V souvislosti s tímto nástrojem byl také popsán současný stav problematiky deformace terénu, který v této fázi vývoje již umožňuje deformovat terén podle vkládaných objektů do grafické databáze. Bylo zjištěno, že touto problematikou se zabývá projekt Terrain Deformation Software, který již přešel z fáze vývojové a testovací do své konečné podoby. Jeho funkčnost byla otestována na modelech terénu, které byly k softwaru přiloženy. Popis funkčnosti a ukázka výsledku práce TDS jsou uvedeny v kapitole 3.1, Projekt „Terrain Deformation Software“.

Pro účely dynamických změn v terénu založeném na fyzikálním základě byly prostudovány relevantní fyzikálně deformovatelné modely. Každý model byl popsán a byly shrnuty jeho výhody a nevýhody. V průběhu studie byly modely porovnávány v souvislosti možné aplikace pro deformovatelný terén, což vyústilo v návrh deformovatelného terénu pro OpenSceneGraph. Tímto modelem je povrchový částicový systém popsaný v kapitolách 4.5, Volně vázané částicové systémy, a 6, Návrh.

Součástí každého fyzikálního modelu je numerický integrátor, který řeší pohybové diferenciální rovnice. Proto byly také prostudovány a implementovány metody, které se u fyzikálních modelů používají. U numerické integrace je obecně doporučován postup adaptivního nastavení integračního kroku, který byl nastudován a popsán. S ohledem na požadavek na fyzikální deformace terénu v reálném čase byla pro pohybující se těleso implementována Eulerova integrační metoda, která je sice nejméně přesná zato ale nejefektivnější. Byly zkoušeny i Runge-Kuttovy numerické metody, které se ukázaly být nezbytné pro částicový systém pro deformace explozí výbušniny, kde dochází k velkým změnám rychlostí během simulace.

Pro chod simulace v reálném čase je důležitá efektivní detekce kolizí. Pro tento účel byl terén rozdělen na regiony, které se deformují během simulace na sobě nezávisle. Celý terén je pak zabalen do hierarchické obalové struktury, která efektivně během probíhající detekce kolize nachází kolizní trojúhelníky a body mezi tělesem a terénem. Datová struktura half-edge byla použita pro efektivní lokální vyhledávání a změnu souřadnic vrcholů v terénu, což výrazným způsobem urychlilo běh simulace.

K dalším výrazným úsporám došlo při aplikaci různých druhů optimalizací, které vycházejí z vnějších zdrojů i ze zkušeností, které byly během práce získány. První typ optimalizace a to ignorování vzdálených částic, s použitím nebo bez použití kd-tree, mělo za následek zrychlení integrace o více než 50%. Druhý způsob optimalizace vycházel z toho, že není potřeba při každé změně terénu vlivem pohybujícího se tělesa znovu sestavovat hierarchickou strukturu terénu, protože deformace probíhá lokálně. To se samozřejmě projevilo na značném snížení výpočetní doby průměrné iterace algoritmu. Obě optimalizace použité současně snížily výpočetní dobu simulačního cyklu v testovacích scénách více jak pětkrát.

V další fázi projektu je vhodné z hlediska použitelnosti rozšířit možnosti deformace na terén s nepravidelnou sítí trojúhelníků. To je případ terénu, ve kterém mají trojúhelníky různou velikost, a proto by částicový systém mající částice ve vrcholech trojúhelníku nebyl použitelný. Rozšíření by potřebovalo možnost přidávat nové body do již existující sítě trojúhelníků, což není pro datovou

strukturu half-edge problém, a také použít triangulační algoritmus pro vytvoření nových trojúhelníků v terénu. Jeden z efektivních triangulačních algoritmů, Delaunayův triangulátor, je použit v samotném toolkitu OpenSceneGraph. Aktualizace BVH terénu by probíhala podobně jako je tomu v této fázi projektu, tedy jednou za určitý počet iterací.

Alternativou pro numerickou integraci, která nebyla v rámci projektu implementována je použití metody leapfrog, která je obecně přesnější a dosahuje větší stability a není o mnoho výpočetně náročnější, než je Eulerova integrační metoda. V dalším pokračování projektu se také nabízí možnost adaptivně měnit integrační krok u jednotlivých částic částicového systému, protože ty se nepohybují všechny stejnou rychlostí, což je popsáno v kapitole 5.5, Adaptivní velikost integračního kroku.

Z důvodu současného rozmachu technologií GPU, je vhodné zvážit použití této platformy pro výpočet některé části algoritmu, čímž by došlo k uvolnění výpočetních prostředků hlavního procesoru (CPU) a následnému zefektivnění algoritmu.

Literatura

[B97] Baraff, D.: An Introduction to Physically Based Modeling: Rigid Body Simulation II – Nonpenetration Constraints. *Siggraph Course Notes* [online]. 1997 [cit. 2007-05-16]. <http://www.cs.cmu.edu/~baraff/pbm/rigid2.pdf>.

[Bar97] Baraff, D.: Physically Based Modeling: Principles and Practice : Implicit Methods for Differential Equations. *Siggraph Course Notes* [online]. 1997 [cit. 2007-05-16]. <http://www.cs.cmu.edu/~baraff/sigcourse/notese.pdf>.

[CSB05] Computer Graphics Systems Development. *Algorithms for Run-Time Terrain Deformation : SBIR Phase II Proposal* [online]. [2005] [cit. 2007-05-16]. <http://www.andesengineering.com/Projects/TDS/Docs/AF04-064_PhaseII-fnc.pdf>.

[CSR05] Computer Graphics Systems Development. *Algorithms for Run-Time Terrain Deformation : status report* [online]. [2005] [cit. 2007-05-16]. <http://www.andesengineering.com/Projects/TDS/Docs/AF04-064_II_Status-Dec_05.pdf>.

[DG96] Desbrun, M., Gascuel, M.-P.: *Smoothed particles: A new paradigm for animating highly deformable bodies* [online]. 1996 [cit. 2007-05-16]. <http://www.geometry.caltech.edu/pubs/DC_EW96.pdf>.

[ED04] Kenny, E.: *Bounding Volume Hierarchies, Collision Detection of Deformable Objects*. Copenhagen University [online]. 2004 [cit. 2007-05-16]. <http://image.diku.dk/kenny/cisp04.pdf>.

[GM97] Gibson, S. F. F., Mirtich, B.: A Survey of Deformable Modeling in Computer Graphics. *MERL Technical Report* [online]. 1997 [cit. 2007-05-16]. <http://www.merl.com/people/frisken/deformationSurvey.pdf>.

[Ham04] Hamby, W.: *BOMB SHOCK WAVE ESTIMATION* [online]. 2004 July 13 [cit. 2007-05-16]. <http://www.makeitlouder.com/document_bombshockwaveestimation.html>.

[HLD05] *Terrain Deformation Software : high level design* [online]. [2005] [cit. 2007-05-16]. <http://www.andesengineering.com/Projects/TDS/HighLevelDesign/>.

[HRW00] Halliday, D., Resnick, R., Walker, J.: *Fyzika*. Brno : VUTIUM, 2000. ISBN 80-214-1868-0.

[Int06] *OSGWiki : introduction.Introduction* [online]. c2004 , March 21, 2006 [cit. 2007-05-16]. <http://www.openscenegraph.org/osgwiki/pmwiki.php/Introduction/Introduction>.

[Kas97] Kass, M.: An Introduction to Physically Based Modeling : An Introduction to Continuum Dynamics for Computer Graphics. *Siggraph Course Notes* [online]. 1997 [cit. 2007-05-16]. <http://www.cs.cmu.edu/~baraff/pbm/continuators.pdf>.

[Kdt06] *Kd-tree* [online]. 2005 , 21 December 2006 [cit. 2007-05-16]. <http://en.wikipedia.org/wiki/Kd-tree>.

[Leo06] Leon, F: *GIMPACT* [online]. 2006 [cit. 2007-05-16]. <http://gimpact.sourceforge.net/>.

[MA06] Mount, D. M., Arya, S.: *ANN - Approximate Nearest Neighbor Library* [online]. Aug 4, 2006 [cit. 2007-05-16]. <http://www.cs.umd.edu/~mount/ANN/>.

[McM05] McMillan, S.: *The Leapfrog Integrator* [online]. 2005 [cit. 2007-05-16]. <http://einstein.drexel.edu/courses/CompPhys/Integrators/leapfrog/>.

[McG00] McGuire, M.: *The Half-Edge Data Structure* [online]. 07 August 2000 [cit. 2007-05-16]. <http://www.flipcode.com/articles/article_halfedge.shtml>.

[Mue04] Müeller, M., et al.: *Point Based Animation of Elastic, Plastic and Melting Objects* [online]. 2004 [cit. 2007-05-16]. <http://www.inf.ethz.ch/personal/giesen/tch/AGCGsem_SS05/mueller.pdf>.

[Nea05] Nealen, A., et al.: *Physically Based Deformable Models in Computer Graphics* [online]. The Eurographics Association, c2005 [cit. 2007-05-16]. <http://graphics.ethz.ch/~mattmuel/publications/egstar2005.pdf>.

[Pau05] Pauly, M., et al.: Meshless Animation of Fracturing Solids. *ACM SIGGRAPH (2005)* [online]. 2005 [cit. 2007-05-16]. <http://www.cs.kuleuven.ac.be/~graphics/CGRG.PUBLICATIONS/MAFS/meshless.pdf>.

[Pro06] *Terrain Deformation Software : progress report for Feb 8, 2006* [online]. Feb 8, 2006 , Feb 8, 2006 [cit. 2007-05-16]. <http://www.andesengineering.com/Projects/TDS/TDSReport_Feb_2006/>.

[PTWF97] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P.: *Numerical Recipes in C : The Art of Scientific Computing*. Second Edition. Cambridge : Cambridge University Press, 2002. ISBN 0-521-43108-5. Chapter 16., Integration of Ordinary Differential Equations, s. 710-714.

[Sce06] *OSGWiki : knowledgeBase.SceneGraph* [online]. c2004 , May 16, 2006 [cit. 2007-05-16]. <http://www.openscenegraph.org/osgwiki/pmwiki.php/KnowledgeBase/SceneGraph>.

[Ton98] Tonnesen, D.: *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation*. [s.l.], c1998. xiv, 188 s. University of Toronto. PhD thesis. <http://www.dgp.toronto.edu/~davet/phd/tonnesen-thesis-ps/tonnesen-0.ps>.

[WB97] Witkin, A., Baraff, D.: Physically Based Modeling: Principles and Practice : Differential Equation Basics. *Siggraph Course Notes* [online]. 1997 [cit. 2007-05-16]. <http://www.cs.cmu.edu/~baraff/sigcourse/notesb.pdf>.

[Wit97] Witkin, A.: Physically Based Modeling: Principles and Practice : Particle System Dynamics. *Siggraph Course Notes* [online]. 1997 [cit. 2007-05-16]. <http://www.cs.cmu.edu/~baraff/sigcourse/notesc.pdf>.

[WTT95] Wu, Y., Thalmann, D., Thalmann, N. M.: *Deformable Surfaces using Physically-Based Particle Systems* [online]. [1995] [cit. 2007-05-16]. <http://vrlab.epfl.ch/Publications/pdf/Wu_Thalmann_Magnenat_Thalmann_CGI_95.pdf>.

[You06] *Young's modulus* [online]. 2003 , 11 December 2006 [cit. 2007-05-16]. <http://en.wikipedia.org/wiki/Young%27s_modulus>.

[Zar04] Žára, J., et al.: *Moderní počítačová grafika : kompletní průvodce metodami 2D a 3D grafiky*. 2. přepracované a rozšířené vyd. Brno : Computer Press, 2004. ISBN 80-251-0454-0. Kapitola 14, Reprezentace scény, s. 397-401.