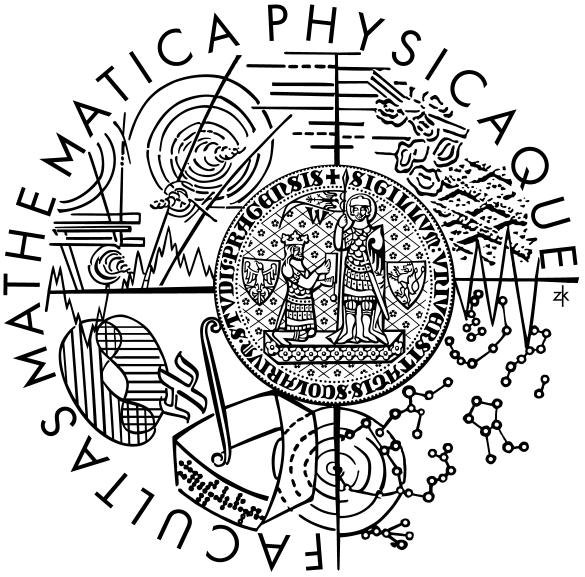
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

# DIPLOMOVÁ PRÁCE




Erik Kratochvíl

# Modelování deformací geometrických objektů

Kabinet software a výuky informatiky

Vedoucí práce: **Doc.Dr.Ing. Ivana Kolingerová**
Studijní program: **Informatika**

Srpen 2007

Děkuji vedoucí své práce Doc.Dr.Ing. Ivaně Kolingerové za poskytnuté připomínky, rady a obětavost. Své rodině děkuji za všechnu podporu, kterou mi poskytla nejen během psaní této práce, a své přítelkyni děkuji za shovívavost a trpělivost.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 10. srpna 2007 Erik Kratochvíl

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam algoritmů

| | |
|---|---|
| **Název práce:** | Modelování deformací geometrických objektů |
| **Autor:** | Erik Kratochvíl |
| **Katedra:** | Kabinet software a výuky informatiky |
| **Vedoucí práce:** | Doc.Dr.Ing. Ivana Kolingerová |

**Abstrakt.** Problematika deformovatelných modelů je v počítačové grafice studována již více než dvě desetiletí. Mnoho souvisejících témat muselo být pokryto a mnoho překážek muselo být překonáno, než se podařilo věrohodně modelovat deformovatelné objekty.

Cílem této práce je simulovat vzájemné interakce mezi několika deformovatelnými objekty v reálném čase. Nejprve studujeme základní principy několika vybraných deformovatelných modelů pevných těles, která jsou reprezentována povrchovým nebo objemovým meshem. Soustředíme se především na fyzikálně založené techniky, které dávají věrohodnější výsledky. V úvahách se omezujeme pouze na modelování elastických materiálů. Dále krátce pojednáváme o tématu detekce kolizí pro deformovatelné modely a jeho specifických aspektech. Zvláštní pozornost věnujeme problematice řešení kolizí, protože zásadně ovlivňuje celkový dojem ze simulace.

Výsledkem práce je návrh algoritmu, detailní popis jeho částí a jeho implementace. Na závěr provádíme i několik měření dokazujících použitelnost navržené metody ve virtuálním prostředí a schopnost pracovat v reálném čase.

**Klíčová slova:** deformovatelné modely, fyzikálně založené modelování, detekce kolizí, řešení kolizí

| | |
|---|---|
| **Title:** | Modelling of Deformations of Geometric Objects |
| **Author:** | Erik Kratochvíl |
| **Department:** | Department of Software and Computer Science Education |
| **Supervisor:** | Doc.Dr.Ing. Ivana Kolingerová |

**Abstract.** Deformable models have been widely studied by the CG community for more than two decades. Many issues had to be addressed and many problems had to be solved before the quality of the deformable models reached an acceptable level.

The aim of the work is to simulate interactions of several deformable bodies in realtime. First, we unveil the basic ideas behind several deformable models created for solids represented by a surface or volumetric mesh. We prefer the physically-based approaches as they tend to yield more convincing results. We consider elastic materials only. We also briefly discuss the topic of collision detection for deformable models with its specific aspects. A special attention is paid to contact resolution because it greatly influences the final impression.

The result of this thesis is an overview of the proposed algorithm, detailed description of its parts, and its implementation. We also perform several benchmarks to prove its applicability in virtual environments and its capability to run in realtime.

**Keywords:** deformable solids, physically-based modeling, collision detection, contact resolution

# Kapitola 1

# Úvod

Počítače se v průběhu času staly neocenitelným pomocníkem v oblasti modelování objektů a scén reálného světa. S narůstajícím výpočetním výkonem začaly nacházet uplatnění i při modelování fenoménů, které můžeme každodenně pozorovat, a při vizualizaci chování vzájemně interagujících objektů.

Sledujeme-li proud vody, vidíme, že se neustále snaží přizpůsobovat povrchu, jímž protéká, vyhýbá se překážkám, tvoří se v něm víry a v různých částech plyne různě rychle. Interaguje nejen s okolním prostředím, ale i sám se sebou. Malý gumový míček se při styku s podlahou deformuje, odráží a snaží se vrátit do svého původního tvaru; pokud ho pevně uchopíme a hodně roztáhneme, můžeme jej roztrhnout. Dlouhé vlasy se ve větru ohýbají a vlní a mají tendenci se spojovat do pramenů, zatímco kratší vlasy a chlupy lépe zachovávají svůj tvar.

Z výše popsaných příkladů je zřejmé, že velkou část reálného světa nelze zjednodušeně namodelovat jako dokonale tuhá tělesa, proto se velmi rychle vynořila potřeba vyvinout speciální modely deformovatelných objektů. Modelování takových objektů v počítačové grafice je složité hlavně díky dvěma protichůdným požadavkům, které jsou na něj kladeny: interaktivita a věrnost.

Maximální fyzikální věrnost simulace je přirozený požadavek, který je kladen na většinu modelů v počítačové grafice, neboť konečným cílem je věrně předvídat chování objektů ve scéně za daných podmínek. Cílem je také obsáhnout různé typy deformací, s nimiž se v reálném světě setkáváme; pro příklad vezměme pevná tělesa vystavená vnějšímu silovému působení. Tělesa se různě ohýbají, stlačují a natahují, někdy se vrací do původního tvaru úplně a někdy jenom z části. Za určitých okolností je možné je rozlomit nebo roztříštit. Na pevná tělesa ovšem nemusíme působit pouze vnější silou, ale i jinak. Dodáváme-li pevnému tělesu teplo, mění se jeho lokální vlastnosti a tvar, některé jeho části se odpařují nebo tají. Všechny nastíněné úvahy platí pouze pro pevná tělesa, kapalná a plynná skupenství se chovají v zásadě jinak a podřizují se jiným zákonům. Například kapaliny nemají samy o sobě žádný tvar a jsou téměř nestlačitelné, predikce chování je vedena jinými rovnicemi než je tomu u pevných těles. Lze tedy předpokládat, že různá látková skupenství budou vyžadovat různé typy deformovatelných modelů.

Fyzikální věrnost umožňuje uživateli chovat se k deformovatelnému objektu jako k *černé skříňce* a zaměřit se výhradně na to, co s modelem zamýšlí provést. Třeba při animaci dovoluje animátorovi soustředit se výhradně na požadovanou akci a ne na dolaďování tvaru povrchu. Při virtuálním chirurgickém zákroku musí být chování objektu i silová odezva natolik adekvátní, aby bylo možné považovat simulaci za trénink skutečného zákroku a nikoli za novou videohru.

Proti fyzikální věrnosti simulace stojí požadavky na maximální rychlost celého systému. Zde je cílem zobrazovat změny stavu simulovaných těles v reálném čase. Současně musí být zahrnuta i odezva na zásahy uživatele do simulovaného prostředí, které mu poskytují zpětnou vazbu. Uživatel může manipulovat se zkoumanými objekty a působit na ně, čímž si rychle a intuitivně osvojí vlastnosti a zákonitosti související se studovaným problémem.

Počátky studia deformovatelných modelů sahají přibližně dvě desetiletí zpátky. První techniky byly relativně omezené nízkým výpočetním výkonem dostupných počítačů a k modelování deformací využívaly převážně geometrických vztahů. Později se vydělily, zdokonalily a utvořily samostatnou skupinu deformovatelných modelů, které se využívají především ke snadné manipulaci s částmi geometrických modelů prostřednictvím zástupných objektů. Zástupným objektem je například speciálně sestrojená kostra objektu; změny poloh jednotlivých kostí jsou zpětně promítány na přilehlé části objektu.

Jen o něco málo později byly navrženy techniky založené na fyzikálních principech nebo vycházející přímo ze zákonů, které jsou předmětem zkoumání mnoha fyzikálních oborů, např. mechaniky kontinua. Tyto modely měly sloužit celé řadě účelů: studiu vlastností známých i nových materiálů vystavených vnějším vlivům a studiu propagace těchto vlivů v materiálu nebo v celém systému. Díky tomu je možné zkoumat vibrace budov, rozložení zátěže u konstrukcí mostů, namáhání částí letadla nebo simulovat tzv. crash-testy automobilů. Později se objevily i úspěšné pokusy simulovat komplexnější objekty živé přírody jako jsou svaly, tkáně, pokožka a dokonce celé vnitřní orgány. Takové modely se uplatňují při počítačem generované animaci výrazů obličeje nebo realistické syntéze pohybu lidí a zvířat. Při nich je nejprve vymodelována kostra živočicha a na ni jsou připevněny deformovatelné objekty reprezentující svaly. Při pohybu kostry se umělé svaly stahují nebo natahují a tím ovlivňují chování další vrstvy — pokožky. Podobné využití nachází deformovatelné modely i v oblasti interaktivní virtuální chirurgie, kde si uživatel může vyzkoušet chování orgánů a tkání a jejich reakce při řezání skalpelem nebo při vpichu jehly.

Deformovatelné modely byly nasazeny i v původně nezamýšlených oblastech. Jednou z nich je analýza obrazu. Zde se deformovatelné modely uplatňují při segmentaci obrazu, což je proces, při němž je obraz rozdělen na několik částí s cílem extrahovat důležité informace, například hranice objektů. Další využití nalézají při rekonstrukci povrchu objektů vzniklých digitalizací 3D skenerem. Deformovatelné modely se postupně přibližují navzorkovaným datům a obepínají povrch určený těmito vzorky, podobně jako punčocha obepíná nohu.

Deformovatelné modely tedy nepředstavují jen zajímavý teoretický problém, ale nachází široké uplatnění v mnoha oblastech — od videoher a filmů až po analýzu materiálu a trénink chirurgických zákroků.

## 1.1 Cíle práce

Kvůli nepřebernému množství typů objektů, které se v přírodě nacházejí, je potřeba se omezit na zkoumání pouze určité skupiny objektů a deformací. Podobně jako ve fyzice ani na poli deformovatelných objektů neexistuje jednotný všeobjímající postup, který dává výborné výsledky za všech okolností.

Cílem této práce je nejprve prostudovat současný stav problematiky modelování deformací pevných těles vlivem mechanického působení jiných pevných těles a vlivem působení vnějšího silového pole. Jiné typy vlivů zanedbáme. Kvůli rozsáhlému množství typů deformací se omezíme pouze na elastické a plastické deformace objektů. Dalším cílem je vybrat a implementovat vhodný postup pro dané typy deformací, který by byl použitelný v oblasti virtuální reality. Vybraný algoritmus by měl umožňovat snadné nastavení vlastností deformovatelných modelů pomocí malého počtu srozumitelných parametrů. Současně musí navržený algoritmus dbát na konzistenci reprezentací simulovaných objektů. Primárním kriteriem kvality algoritmu je rychlost a interaktivita dosažitelná i na běžně dostupných počítačových sestavách. Posledním cílem je implementovaný algoritmus otestovat na vhodných datech, zhodnotit dosažené výsledky a celkovou použitelnost.

## 1.2 Struktura textu

Celá práce je logicky rozdělena na tři části. První část práce se věnuje úvodu do problematiky deformovatelných modelů. Protože se budeme zabývat převážně zjednodušenými modely, které jsou s to pracovat v reálném čase, oslabíme nejprve v kapitole 2 význam fyzikální věrnosti. Dříve než přistoupíme ke konkrétním deformovatelným modelům, krátce připomeneme nejběžnější způsoby reprezentace pevných těles spolu s jejich klady a zápory, protože pro konkrétní typy deformací se hodí různé druhy reprezentací. Zbytek kapitoly je věnován vymezení základních pojmů, nastínění matematicko-fyzikálního popisu deformací pevných těles a kategorizaci deformovatelných modelů.

V druhé části práce se soustředíme především na detailní popis vlastních deformovatelných modelů pro pevná tělesa (kapitola 3). Diskutujeme jejich přednosti a nedostatky a zmiňujeme nejběžnější způsoby využití. Protože cílem práce je simulovat vzájemné interakce dvou pevných deformovatelných těles, musíme se zabývat i studiem detekce kolizí pro deformovatelné objekty, které stručně pokrývá kapitola 4. V kapitole 5 představujeme nejběžnější techniky, které mají za úkol nastalou kolizi vyřešit a případně dopočíst vhodnou silovou odezvu pro zpětnou vazbu.

Třetí část rozdělená mezi kapitoly 6 a 7 obsahuje detailní popis navrhovaného řešení včetně celkového schematu algoritmu, hlavní přednosti, nastavení a chování na testovacích datech.

# Kapitola 2

# Základy modelování deformací

## 2.1 Věrnost a věrohodnost

Přesně zjistit, co by se skutečně stalo v nějaké přesně popsané situaci reálného světa — to je nejvyšší cíl každé dynamické simulace, která usiluje o modelování fyzikálně korektního chování nějakého objektu. Tato kapitola poukazuje na fakt, že z pohledu diváka nehraje přesnost zásadní roli; mnohem důležitější je naopak vlastnost, kterou budeme nazývat *věrohodnost* [BHW96].

V reálném světě existují simulace, které mají své vlastní potřeby. Například při počítačových animacích určených pro zábavní průmysl není zapotřebí kvalitní prediktivní model, který přesně spočte, co se v dané situaci stane. Je to totiž animátor, kdo už předem rozhodl, co se má stát. Smysl simulace tkví v poskytnutí nástrojů, které umožní situaci zobrazit tak, aby divák uvěřil, že se scéna mohla zobrazeným způsobem ve skutečnosti odehrát.

Většina běžných simulací zpravidla vypadá *sterilně*. Jedním z důvodů je nedostatek variability, který je zapříčiněn zanedbanými detaily v matematickém modelu. Mnohé detaily se z rovnic a upravených fyzikálních modelů vypouští, protože by jejich zahrnutí výrazně zesložitilo výpočet celé simulace. Z toho důvodu má smysl v některých případech usilovat spíše o věrohodnost simulace. Věrohodnost zde znamená, že určitý scénář chování může nastat za daných znalostí, které máme o systému. Pro konkrétní podmínky může existovat několik věrohodných scénářů.

Ilustrujme celý problém na simulaci pohybu míče. Uvažme klasický případ, kdy je koule z klidového stavu vypuštěna na vodorovnou rovinu. Skáče nahoru a dolů právě nad jedním bodem této roviny. Uvážíme-li však reálný míč, který je spuštěn na podlahu, odrazí se pokaždé do jiného směru. Přestože je při opakování experimentu jeho trajektorie vždy jiná, povaha pohybu míče bude pořád stejná. Během reálného experimentu totiž působí celá řada zanedbaných faktorů: nepatrná vlastní rotace míče udělená při jeho vypuštění, nepravidelnosti na povrchu míče, nehomogenity v hustotě, nerovnosti podlahy způsobené nedokonalostmi při stavbě i dalšími částečkami jako prach a další faktory.

Dokonalá simulace nabízená klasickou počítačovou grafikou je tedy v jistém smyslu příliš mechanická a působí umělým dojmem. Pro naprosto přesnou reálnou simulaci je však nemožné uvážit úplně všechny faktory, které působí během simulace. Pro pozorovatele ale stejně není podstatné, kterou cestou se míč po vypuštění přesně vydá, dokud se bude zdát, že se pohybuje realisticky.

Nadále budeme vycházet z následujícího pozorování: ne vždy (vlastně prakticky málokdy) jsou lidé schopni absolutně přesných odhadů, které se týkají pohybu a v některých případech jsou lidé schopni přijmout i nepravděpodobné chování spoléhajíce se na neviditelné síly jako působení větru nebo vlastní rotaci objektu. Experimenty při sledování pohybujících se navzájem kolidujících objektů naznačují, že lidé jsou schopni se plně soustředit pouze na jednu informaci zahrnutou v simulaci — například na odhadování rychlosti těles po srážce. Jsou-li však zahrnuty další jevy jako rotace objektů po srážce, dělá lidem potíže správně identifikovat fyzikálně nekorektní chování [OD01].

Přímo v samotných počítačových simulacích dochází k nežádoucí variabilitě v důsledku mnoha jevů:

- numerické chyby při výpočtech — žádné počítače nejsou schopny reprezentovat všechna reálná čísla přesně
- aproximace daná abstrakcí problému — ve skutečnosti neexistuje žádné dokonale tuhé nebo dokonale elastické těleso
- nepřesnost vstupních dat — jak přesně známe počáteční rychlosti, jak přesně známe hustotu objektu ve všech jeho částech, ...
- chybějící detaily modelů — míče nejsou dokonale kulaté ani plochy nejsou dokonale rovné
- nestabilita systému — divergence zapříčiněná např. velkou časovou diskretizací

Simulaci lze považovat za *fyzikálně věrnou* (physically correct), pokud leží uvnitř prostoru vymezeném pouze výše uvedenými variabilitami. Simulace je *vizuálně věrohodná* (visually plausible), pokud vypadá dostatečně přesvědčivě. Nejde samosebou o přesnou definici, jelikož závisí na řadě kognitivních faktorů — kdo se dívá, odkud se dívá a jaké má zkušenosti s daným jevem.

V této práci obětujeme absolutní fyzikální věrnost, abychom byli schopni dosáhnout simulací, které běží v reálném čase. Budeme místo toho usilovat o vizuální věrohodnost, tedy o stav, kdy se bude uživateli jevit, že jde o přesnou simulaci.

## 2.2 Reprezentace objektů

V námi vnímaném světě se vyskytuje nepřeberné množství různých tvarů: od jednoduchých euklidovských obrazců jako jsou trojúhelníky, čtverce nebo koule,

přes hladké křivky lodních trupů až po komplexní fraktální strukturu mraků, hor nebo stromů. Reprezentace schopná obsáhnout všechny typy tvarů samosebou neexistuje. Tato část práce zběžně představuje nejpoužívanější způsoby reprezentace pevných těles v počítačové grafice. Soustředí se na to, jakým způsobem daná reprezentace objekt popisuje, bez udání konkrétních datových struktur. Jednotlivé reprezentace se liší mezi sebou množstvím informací, které udržují o objektech a které se týkají především tvaru povrchu a celkové topologie. U reprezentací jsou uvedeny jejich výhody a nevýhody zejména z pohledu složitosti jejich vizualizace a složitosti detekce kolizí.

### 2.2.1 Bodová reprezentace

*Bodová reprezentace* (Point-Based Representation, PBR) popisuje objekt konečnou množinou bodů, které leží na jeho povrchu [ABCO+01]. Pro tyto body se přirozeně nabízí termín *surfel* — surface element [PZvBG00]. Každý surfel krom prostorové informace může nést informace o normále, barvě povrchu nebo texturovacích souřadnicích. PBR lze rozšířit o objemovou informaci přidáním konečné množiny bodů, které vzorkují vnitřek objektu. Tyto body se občas nazývají *phyxely* [MKN+04] a usnadňují propagaci vlivů z jedné strany povrchu na druhou.

PBR zaznamenává v dnešní době rostoucí množství pozornosti zejména díky cenově dostupné, kvalitní a přesné technice, která je schopná generovat velmi husté množiny surfelů. Bodově reprezentovaný objekt vyžaduje velké množství primitiv, ta však — při běžném přiblížení objektu — na zobrazovacím rastrovém zařízení zabírají méně než jeden pixel a stávají se díky tomu efektivními zobrazovacími primitivy, protože oproti povrchovému meshi není zapotřebí ořezávání polygonů ani rasterizace. Zobrazování objektů reprezentovaných PBR přináší problémy až při velkém přiblížení objektu, kdy surfely nedostatečně pokrývají povrch a na rastrovém zobrazovacím zařízení se mezi nimi objevují mezery.

Vzrůstající popularitu této reprezentace dokládá řada článků, které se věnují modelování deformací bodově reprezentovaných objektů, za všechny uveďme např. [PKKG03]. Velmi efektivně a snadno lze provádět velké změny tvaru či topologie objektu. Simulovat lze rozsáhlé deformace i tání objektů. Přesto se PBR ujala spíše pro reprezentaci kapalin a zrnitých materiálů. Efektivní detekce kolizí bodově reprezentovaných objektů je studována [KZ04].

### 2.2.2 Mesh

*Mesh* $M$ je dvojice $(K, V)$ [HDD+94]. $V$ je množina vrcholů, což jsou body $N$-rozměrného euklidovského prostoru $E_N$. $K$ je *simpliciální komplex*, který popisuje vzájemné geometrické vztahy vrcholů, tzv. *konektivitu*. Nechť $v_0, \ldots, v_n \in V$ je $n+1$ afinně nezávislých bodů, potom *$n$-simplex* $s(v_0, \ldots, v_n)$ je konvexní obal těchto bodů. 0-simplex je bod, 1-simplex je úsečka (hrana), 2-simplex je trojúhelník, 3-simplex je čtyřstěn. *Stěna* $f$ simplexu $s(v_0, \ldots, v_n)$ je konvexní obal neprázdné podmnožiny

množiny $\{v_0, \ldots, v_n\}$. *Simpliciální komplex* $K$ je množina simplexů, které splňují následující dvě podmínky:

1. všechny stěny $f$ simplexů $s \in K$ jsou též v $K$,
2. jsou-li $s_1, s_2$ simplexy v $K$, pak i jejich průnik je v simplexu $K$.

Reprezentace meshem (mesh representation, MR) je nejběžnější typ reprezentace v počítačové grafice. *Povrchový mesh* je mesh, který obsahuje 2-simplexy a všechny simplexy jsou nejvýše 2-simplexy. Používá se k po částech rovinné interpolaci povrchu reálných objektů. *Objemový mesh* je mesh, který obsahuje 3-simplexy a všechny simplexy jsou nejvýše 3-simplexy. Povrchové meshe a povrch objemových meshů lze snadno a rychle zobrazovat, zejména díky podpoře dnešních grafických akcelerátorů. Pro objekty reprezentované povrchovými nebo objemovými meshi bylo odvozeno množství metod pro simulaci jejich deformací. Simulovat lze prakticky jakýkoli jev — trvalé i vratné deformace, lámání i praskání objektů nebo jejich tavení. Detekce kolizí je dobře prostudována pro povrchové i objemové meshe. Detailně se těmto tématům věnuje Kapitola 3 a 4.

### 2.2.3 Alternativní reprezentace

MR se nehodí k reprezentaci objektů, které obsahují velké detaily. Takové objekty vznikají typicky při digitalizaci objektů reálného světa. Proto byly vytvořeny speciální reprezentace, které separují obecné informace o tvaru objektu od detailů. Část reprezentující obecný tvar objektu se nazývá *řídící model* nebo *základní doména*. Řídící model je zpravidla uložen jako povrchový nebo objemový mesh. Na podobném principu pracuje i známá technika bump mapping. Mezi nejznámější zástupce těchto reprezentací patří lineární diferenciální souřadnice [LSCO+04] nebo normálové meshe [GVSS00].

Deformace se pak provádějí buď na řídícím modelu, který bývá jednodušší, přičemž detaily se nemění a použijí se až při vizualizaci objektu [Ale06], nebo se naopak úpravy provádějí jenom s detaily, čímž lze velmi rychle simulovat lokální deformace jako třeba vrypy do povrchu.

Nevýhodou těchto reprezentací je netriviální konverze z původní reprezentace. Složitá je i vizualizace objektů v těchto reprezentacích; lze ji však zvládnout s použitím moderního grafického akcelerátoru. Nepříliš prozkoumanou oblastí je detekce kolizí, protože je závislá na konkrétním typu reprezentace.

### 2.2.4 Shrnutí

Pevná tělesa reálného světa lze reprezentovat mnoha způsoby, nejběžnější pro modelování deformací jsou reprezentace povrchovým nebo objemovým meshem. Pro tento typ reprezentací jsou velmi dobře prostudovány různé techniky simulace deformací

i detekce kolizí. Modely lze v této reprezentaci snadno vizualizovat. Méně vhodné jsou pouze v případech, které vyžadují generování nových částí povrchu. V dalším textu se soustředíme především na techniky pracující s touto reprezentací objektů.

Bodová reprezentace je novější způsob, nicméně většinu technik používaných s MR lze pro ni adaptovat. Změny topologie jsou oproti MR velmi snadné. Detekce kolizí pro bodově reprezentované objekty využívá podobné principy jako u MR.

Alternativní reprezentace se zpravidla hodí pro uživatelem prováděné deformace; za tímto účelem byly primárně stvořeny. S reprezentovanými modely umožňují řadu operací, které jsou jinak v MR i PBR složité. Nevýhodou je složitější vizualizace a nutně specializovaná detekce kolizí.

## 2.3 Modelování deformací

Vnější síla, která působí na těleso skutečného světa, způsobuje změnu tvaru tohoto tělesa. Tuto změnu nazýváme *deformací*. V některých případech jsou deformace tak malé, že mohou být zanedbány; pevné těleso se v takovém případě chová jako (dokonale) *tuhé těleso*. Tuhé těleso je model tělesa, jehož rozměry nelze zanedbat, nicméně působením sil se téměř nedeformuje. V jiných případech je naopak deformace tělesa jeho význačnou vlastností.

Přístupy k modelování deformací lze rozlišit podle jejich základu na čistě geometrické a na fyzikálně založené. Vzájemně se liší dosažitelnou kvalitou, věrohodností a rychlostí simulace, numerickou stabilitou výpočtů a nároky na reprezentaci objektů. Liší se i zamýšleným nasazením — mnohé metody byly vyvinuty pro modelování konkrétních typů deformací (elastické, plastické, praskání, tání) nebo pro modelování deformací konkrétních objektů (šatů).

### 2.3.1 Geometrické metody

Přestože většina metod je v součastnosti založena na nějakém fyzikálním základě, existuje početná skupina metod využívající čistě geometrické vztahy mezi primitivy reprezentací. Tyto techniky bývají zpravidla výpočetně nenáročné, pracují v interaktivních rychlostech a spoléhají na schopnost uživatele dosáhnout požadovaného výsledku. Díky tomu jsou limitovány zkušenostmi, zručností a trpělivostí uživatele, který deformace provádí; systém nezahrnuje žádné znalosti o povaze objektů, s nimiž je manipulováno. To je ostatně důvod obliby fyzikálně založených přístupů. Geometricky založené metody nejsou zpravidla určeny pro deformace vzniklé mechanickým působením jiného objektu nebo působením sil, ale pro intuitivní editaci uživatelem. Existuje však i výjimka, kterou je Shape Matching, metoda založená na porovnávání aktuálního deformovaného tvaru tělesa s původním klidovým tvarem. Tato technika je velmi rychlá, nicméně ze své podstaty je omezená pouze na simulace malého počtu deformací.

### 2.3.2 Metody založené na fyzikálních principech

Čistě geometrické přístupy neobsahují žádné znalosti o povaze simulovaného systému, z toho důvodu byly vyvinuty modely, které vychází alespoň částečně z fyzikálního základu. Tím lze dosáhnout přesvědčivějších výsledků zejména při modelování vzájemných deformací dvou těles v kontaktu. V této práci se soustředíme především na tyto přístupy.

Konkrétní techniky často vyžadují konkrétní typy reprezentací simulovaných objektů, nejčastěji objemových nebo povrchových meshů. V poslední době získávají stále větší oblibu i PBR, které se výborně hodí pro deformace zahrnující změny topologie.

### 2.3.3 Fyzikální základ

Abychom byli schopni přesně popsat, co je deformace a jaký vztah k nim mají vnější síly, připomeneme nejprve fyzikální základ daný mechanikou kontinua [BSS05]. Součástí je vymezení pojmů, na které bude dále v textu odkazováno.

**Vnitřní a vnější síly.** Vyšetřování účinků sil na skutečná tělesa zkoumá mechanika kontinua. Mechanika kontinua předpokládá, že určitá oblast prostoru je spojitě vyplněna látkou neboli *kontinuem*, které se působením vnějších sil deformuje. Síly působící na kontinuum jsou dvojího druhu: vnitřní a vnější. *Vnější síly* mají charakter zatížení. Jsou projevem působení silového pole a projevují se buď uvnitř celého kontinua jako *objemové síly*, kde působí na každý objemový element kontinua přímo úměrně jeho hmotnosti, nebo jsou plošně rozdělené na povrchu kontinua (tj. ve stykové ploše kontinua s okolním kontinuem) a nazývají se *plošné síly*. Všechny vnější síly působí nezávisle na silách působících na sousední objemové/plošné elementy. Příkladem objemových sil je síla gravitační nebo setrvačná, příkladem plošné síly je třecí síla. *Vnitřní síly* vznikají na povrchu kontinua a z těchto míst se přenášejí do vnitřní části kontinua, tzn. z jedné části kontinua na druhou. Účinek těchto sil se přenáší po ploše; silové působení z jedné strany libovolné myšlené plošky uvnitř kontinua je vyrovnáno působením síly stejného charakteru, ale opačného směru z druhé strany plošky —- podle principu akce a reakce. Tyto síly brání vzájemnému oddělení obou částí kontinua.

**Vektor napětí.** *Vektor napětí* $\vec{\sigma}$ je síla $d\mathbf{F}$ vztažená na jednotku plochy $d\mathbf{S}$ uvnitř nebo na povrchu kontinua. Vyjadřuje vzájemné silové působení na plošce $d\mathbf{S}$ dvou částí uvažovaného kontinua, které z obou stran k této plošce přiléhají. Ploškou $d\mathbf{S}$ myslíme část uzavřené plochy $\mathbf{S}$, která vyděluje z kontinua určitý konečný objem. Tím je jednoznačně určen směr vnější normály plošky $d\mathbf{S}$. Napětí tedy závisí nejen na poloze plošného elementu v kontinuu, ale i na jeho orientaci. Pravoúhlý průmět vektoru napětí do směru normály $\vec{\mathbf{n}}$ nazýváme *normálovým napětím* a do roviny kolmé k normále *napětím tečným*.

**Tenzor napětí.** Nechť $\vec{\sigma}$ je vektor napětí působící na infinitezimální plošku $d\mathbf{S}$, jejíž vnější normála $\vec{\mathbf{n}}$ je rovnoběžná s osou $\mathbf{x}_i$. Označme $\tau_{ij}$ průmět $\vec{\sigma}$ do osy $\mathbf{x}_j$. $\tau_{ij}$ jsou složky *tenzoru napětí* $\sigma$ (stress tenzor), první index určuje směr vnější normály plošky $d\mathbf{S}$ a druhý index směr napětí. $\tau_{ii}$ jsou normálová napětí. $\tau_{ii}$ kladné znamená dilataci (protažení) materiálu, záporné kompresi (stlačení). Zbylá $\tau_{ij}$ jsou napětí smyková.

**Podmínky rovnováhy.** Nepůsobí-li na kontinuum žádné vnější síly, je kontinuum v *přirozeném (klidovém) stavu*. Kontinuum je v *rovnováze*, pokud výslednice i výsledný moment všech sil na něj působících je roven nule.

**Materiálové a světové souřadnice.** Chápeme-li klidový tvar pevného tělesa jako souvislou spojitou podmnožinu $M$ prostoru $\mathbf{R}^3$, potom souřadnice $\mathbf{m}$ bodu tohoto tělesa jsou jeho *materiálové souřadnice*. Působením sil dochází k deformaci objektu a bod se souřadnicemi $\mathbf{m} = (x_1^0, x_2^0, x_3^0)$ se dostane na novou *světovou souřadnici* $\mathbf{x}(\mathbf{m}) = (x_1, x_2, x_3)$. $\mathbf{x}$ je vektorové pole definované na množině $M$. Alternativně lze na deformaci pohlížet jako na vektorové pole posunutí $\mathbf{u}(\mathbf{m}) = \mathbf{x}(\mathbf{m}) - \mathbf{m}$.

**Tenzor deformace.** Nové světové souřadnice tělesa zahrnují nejen transformace, při nichž se těleso otáčí nebo posouvá jako tuhý celek (rigid body transformation, RBT), ale i vlastní deformace tělesa, jejichž projevem jsou změny tvaru tělesa. Vlastní deformace, tj. změny vzájemné polohy jednotlivých bodů tvořících kontinuum, popisují veličiny $\varepsilon_{ij}$, což jsou složky *tenzoru deformace* (strain tenzor). $\varepsilon_{ii}$ charakterizují relativní prodloužení přímkových elementů, které byly před deformací rovnoběžné se souřadnými osami. Vztah těchto složek k posunutí bodu je

$$\varepsilon_{ii} = \frac{\partial u_i}{\partial x_i}$$

Zbylé složky $\varepsilon_{ij}$ (někdy značeny $\gamma_{ij}$) charakterizují změnu pravého úhlu, který spolu svíraly dva přímkové elementy, z nichž jeden byl před deformací rovnoběžný s osou $\mathbf{x}_i$ a druhý s $\mathbf{x}_j$. Vztah, který se používá v případě malých deformací je

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{x_j}\right)$$

Výsledný tenzor $\varepsilon$ se nazývá *Cauchyho lineární tenzor* $\varepsilon_C$. Pro popis velkých deformací je vhodnější *Greenův nelineární tenzor* $\varepsilon_G$, který má tvar

$$\varepsilon_{ij} = \left(\frac{\partial \mathbf{u}}{\partial x_i} \cdot \frac{\partial \mathbf{u}}{\partial x_j}\right) - \delta_{ij}$$

$\delta_{ij}$ je Kroneckerovo delta.

**Vztah mezi $\varepsilon$ a $\sigma$.** Závislost mezi napětím a deformací má několik fází [Kaf04]

1. *elastická fáze* (vratná deformace)

   Přestane-li vnější síla působit, látka se vrací do klidového stavu. Největší napětí, při kterém jsou deformace ještě elastické, nazýváme *mezí pružnosti*. Podle Hookova zákona je relativní prodloužení většiny pevných látek při působení napětí menšího než mez pružnosti přímo úměrné tomuto napětí. Koeficient úměrnosti se označuje jako *Youngův modul pružnosti* nebo *koeficienty elasticity*. Obecně to ovšem není pravda a Hookův zákon platí pouze do tzv. *meze úměrnosti*. V oblasti mezi oběma mezemi je vztah deformace a napětí nelineární. Příkladem materiálů, které vykazují převážně lineární chování je ocel, uhlíkové vlákno nebo sklo. Mezi materiály, jejichž chování je — mimo oblast velmi malých deformací — nelineární, patří například přírodní guma.

2. *plastická fáze* (nevratná deformace)

   Pokud napětí překoná mez pružnosti, klidový tvar objektu je trvale změněn. Část deformace vymizí a část deformace je trvalé povahy — tu nazýváme plastickou deformací. Dojde-li k dalšímu namáhání, dosáhneme *meze pevnosti*, po jejímž překročení dochází k porušení souvislosti kontinua.

Vztah mezi tenzorem napětí a tenzorem deformace pro lineárně elastické materiály upravuje zobecněný Hookův zákon

$$\sigma_{ij} = \sum_{k,l} E_{ijkl}\varepsilon_{kl} \tag{2.1}$$

$E$ je tenzor řádu 4 a závisí na konkrétním materiálu. Obecné tenzory čtvrtého řádu mají 81 koeficientů. Za předpokladu, že $\sigma$ i $\varepsilon$ jsou symetrické, je nezávislých jen 36 složek.

**Potenciální energie deformace.** Elastická deformace způsobuje ukládání potenciální energie v látce; tato energie způsobí po odtížení návrat látky do původního tvaru. Deformační potenciální energie $E_P$ látky se vypočítá z *hustoty elastického potenciálu* $\eta$ definovaného vztahem

$$\eta(\mathbf{m}) = \sum_{k,l} \sigma_{kl}\varepsilon_{kl}$$

integrací přes celý objem $V$ dané látky

$$E_P = \int_V \eta(\mathbf{m}) d\mathbf{m}$$

**Tenzor rychlosti deformace.** Při popisu deformace se dále zavádí tenzor $\nu$ *rychlosti deformace* (strain rate), který je definován jako

$$\nu_{ij} = \left(\frac{\partial \mathbf{u}}{\partial x_i} \cdot \frac{\partial \dot{\mathbf{u}}}{\partial x_j}\right) + \left(\frac{\partial \dot{\mathbf{u}}}{\partial x_i} \cdot \frac{\partial \mathbf{u}}{\partial x_j}\right)$$

$\dot{\mathbf{u}}$ je derivace $\mathbf{u}$ podle času, tedy rychlost bodu ve světových souřadnicích.

**Viskoelasticita.** V praxi se setkáváme nejen s čistě elastickými látkami, ale i s tzv. *viskoelastickými látkami*. U těchto látek nedochází k okamžité reakci na změnu v zatížení, naopak se deformace projeví s určitým zpožděním, tzv. *jev hystereze*. Tento jev je způsoben ztrátou energie při změně zatížení. V případě viskoelastických materiálů má tenzor napětí $\sigma$ dvě složky, a to tenzor elastického napětí $\sigma^\varepsilon$ a tenzor viskózního napětí $\sigma^\nu$. Celkové napětí $\sigma$ je součtem obou složek, tj. $\sigma = \sigma^\varepsilon + \sigma^\nu$. Vztahy mezi těmito napětími, deformací a rychlostí deformace jsou v případě lineárních viskoelastických materiálů

$$\sigma^\varepsilon_{ij} = \sum_{k,l} E_{ijkl}\varepsilon_{kl} \qquad \text{a} \qquad \sigma^\nu_{ij} = \sum_{k,l} D_{ijkl}\nu_{kl}$$

$E$ má týž význam, jako v (2.1). $D$ je tenzor 4. řádu, který definuje tlumicí vlastnosti materiálu.

**Potenciál tlumení.** *Hustota potenciálu tlumení* $\kappa$ je zavedena vztahem

$$\kappa(\mathbf{m}) = \sum_{k,l} \sigma^\nu_{kl}\nu_{kl}$$

a integrací přes celý objem $V$ látky dostaneme *potenciál tlumení* $E^\nu_P$

$$E^\nu_P = \int_V \kappa(\mathbf{m}) d\mathbf{m}$$

Tlumicí potenciál se vztahuje ke kinetické energii materiálu po odečtení energie spojené s RBT[1] a po její normalizaci vzhledem k hustotě.

### 2.3.4 Integrační schémata

Pro simulaci dynamicky se deformujících těles je zapotřebí znát světové souřadnice $\mathbf{x}(\mathbf{m})$ tělesa v čase, tj. $\mathbf{x}(\mathbf{m}, t)$. Jsou-li tyto souřadnice známy, zobrazujeme postupně $\mathbf{x}(\mathbf{m}, t_0), \mathbf{x}(\mathbf{m}, t_1), \mathbf{x}(\mathbf{m}, t_2), \ldots$ čímž dosáhneme animace deformace objektu. Neznámé vektorové pole $\mathbf{x}(\mathbf{m}, t)$, které zjednodušeně označujeme $\mathbf{x}(t)$ nebo jenom $\mathbf{x}$, je však zpravidla dáno implicitně jako řešení diferenciální rovnice tvaru

$$\ddot{\mathbf{x}} = F(\dot{\mathbf{x}}, \mathbf{x}, t) \tag{2.2}$$

[1] Rigid Body Transformations, transformace objektu jako tuhého celku

$\dot{\mathbf{x}}$ je derivace podle času $t$. $F$ je obecná algebraická funkce daná konkrétním modelem deformovaného tělesa. Kupříkladu pro čistě elastické materiály má rovnice (2.2) tvar

$$\rho\ddot{\mathbf{x}} = \nabla \cdot \sigma + \mathbf{f}^e$$

$\nabla \cdot \sigma$ popisuje vnitřní síly, $\rho$ je hustota materiálu a $\mathbf{f}^e$ jsou vnější síly. Rovnice (2.2) se přepisuje do formy diferenciálních rovnic prvního řádu

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= F(\mathbf{v}, \mathbf{x}, t) \end{aligned} \tag{2.3}$$

**Explicitní Eulerova metoda.** Řešení soustavy (2.3) lze najít numerickou integrací [WB93]. Nejjednodušší způsob řešení je *explicitní (dopředná) Eulerova metoda*, která aproximuje derivace následujícím způsobem

$$\dot{\mathbf{x}}(t) = \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} \tag{2.4}$$

Z (2.4) lze snadno vyjádřit $\mathbf{x}(t+h)$, které nás zajímá

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\dot{\mathbf{x}}(t) \tag{2.5}$$

Rovnice (2.3) tak nabydou (časově) diskretizovaného tvaru

$$\begin{aligned} \mathbf{x}(t+h) &= \mathbf{x}(t) + h\mathbf{v}(t) \\ \mathbf{v}(t+h) &= \mathbf{v}(t) + hF(\mathbf{v}(t), \mathbf{x}(t), t) \end{aligned}$$

Tato metoda je velmi jednoduchá, její velká slabina je numerická stabilita. Stabilní je pouze v případě, že krok $h$ je dostatečně malý; je-li nastaven špatně, řešení diverguje. Důvod je ten, že místo toho, abychom sledovali trajektorii $\mathbf{x}(t)$ přesně, se posuneme ve směru tečny předpokládajíce, že v obou případech dojdeme do stejného bodu. Podstatu problému lze pozorovat při rozepsání Taylorova rozvoje

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\dot{\mathbf{x}}(t) + \frac{1}{2}h^2\ddot{\mathbf{x}}(t) + \frac{1}{6}h^3\dddot{\mathbf{x}}(t) + O(h^4) \tag{2.6}$$

Je zřejmé, že aproximace (2.5) vznikla oříznutím Taylorova rozvoje a je přesná jen v případě, že všechny další derivace řádu většího než jedna jsou nulové, neboli že $\mathbf{x}(t)$ je lineární funkce.

**Verletovo schéma.** Protože explicitní Eulerova metoda je přesná pouze v řádu $O(h^2)$, objevila se přesnější explicitní schémata aproximující derivace. Jedním z nich je *Verletovo schéma* [THMG04]. Sečtením rozvoje (2.6) s rozvojem

$$\mathbf{x}(t-h) = \mathbf{x}(t) - h\dot{\mathbf{x}}(t) + \frac{1}{2}h^2\ddot{\mathbf{x}}(t) - \frac{1}{6}h^3\dddot{\mathbf{x}}(t) + O(h^4) \tag{2.7}$$

dostaneme vztah

$$\mathbf{x}(t+h) = 2\mathbf{x}(t) - \mathbf{x}(t-h) + h^2\ddot{\mathbf{x}}(t) + O(h^4) \tag{2.8}$$

S použitím vztahu (2.2) lze rovnici (2.8) přepsat do tvaru

$$\mathbf{x}(t+h) = 2\mathbf{x}(t) - \mathbf{x}(t-h) + h^2 F(\mathbf{v}(t), \mathbf{x}(t), t)$$

Pro rychlost lze odvodit vztah odečtením rozvojů (2.6) a (2.7)

$$\mathbf{v}(t+h) = \frac{\mathbf{x}(t+h) - \mathbf{x}(t-h)}{2h}$$

Jedná se však stále o explicitní schéma, proto trpí stejnými nedostatky, jako klasická explicitní Eulerova metoda. Drobnou nevýhodou je i nutnost mít k dispozici polohy bodů ze dvou předcházejících časových kroků $t$ a $t-h$.

**Implicitní Eulerova metoda.** Úplně jiný způsob integrace představuje *implicitní (zpětná) Eulerova metoda* která využívá hodnoty ze stejného časového kroku na obou stranách rovnice [Bar01]. Časově diskretizovaná soustava (2.3) má tvar

$$\begin{aligned} \mathbf{x}(t+h) &= \mathbf{x}(t) + h\mathbf{v}(t+h) \\ \mathbf{v}(t+h) &= \mathbf{v}(t) + hF(\mathbf{v}(t+h), \mathbf{x}(t+h), t) \end{aligned}$$

Tato metoda je numericky stabilní pro mnohem větší kroky $h$, než které dovolují explicitní metody. Cenou za tuto výhodu je nutnost řešit soustavu (obecně nelineárních) algebraických rovnic. Změna oproti explicitnímu Eulerovu schématu spočívá v tom, že místo vyhodnocení funkce $F$ v bodě, ve kterém se zrovna nacházíme, ji vyhodnocujeme v bodě, kam se teprve dostaneme. Jméno zpětná si vysloužila díky tomu, že perfektní smysl rovnice dávají v případě, že by svět běžel pozpátku. Jsme-li totiž v čase $t+h$ v bodě $\mathbf{x}(t+h)$ a provedeme-li krok $-h\mathbf{v}(t+h)$, skončíme v bodě $\mathbf{x}(t)$.

## 2.4 Shrnutí

V kapitole jsme nejprve diskutovali význam fyzikální věrnosti a rozhodli se omezit na vizuálně věrohodné simulace. Druhé rozhodnutí se týkalo volby reprezentace simulovaných objektů, kterou budeme používat. Zde jsme zvolili klasickou reprezentaci povrchovým a objemovým meshem.

Ve zbylé části kapitoly jsme se věnovali základům modelování deformací. Nejprve jsme metody kategorizovali podle toho, zda mají alespoň částečný fyzikální základ, či nikoli. Protože geometrické metody jsou pro modelování interakcí dvou pevných těles v kontaktu použitelné pouze v omezené míře, budeme se zabývat zejména technikami, které mají fyzikální základ a produkují jednoduché diferenciální rovnice. Z toho důvodu jsme se stručně věnovali i matematicko-fyzikálnímu jádru deformací. Drželi jsme se především [NMK+06], detaily jsme čerpali hlavně z [BSS05] a [Kaf04].

Pro časovou diskretizaci rovnic vznikajících ve fyzikálně založených metodách je pak nutné použít některé z představených integračních schémat. Implicitní Eulerova metoda vyžaduje časově náročné řešení obecně nelineárních algebraických rovnic. Při jejich linearizaci dochází k artefaktům spojených s tím, že lineární zobrazení popisuje pouze omezenou množinu deformací. Tyto nevýhody nemůže vyvážit ani

dosažitelný velký časový krok. Proto ideálním integračním schématem je pro nás Verletovo schéma, které lze v porovnání s jinými metodami vyčíslit velice rychle a nabízí přijatelnou chybu.

# Kapitola 3

# Deformovatelné modely

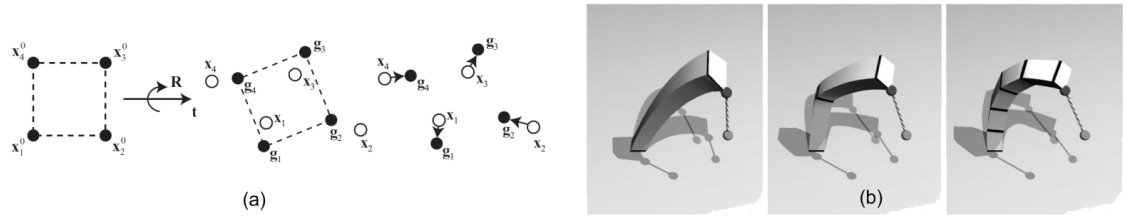
Úkol deformovatelných modelů je zejména vyhodnotit celkové síly, které působí na těleso, a podniknout takové kroky, které povedou k dosažení rovnovážného stavu nebo ke snížení deformační energie.

V této kapitole podrobně popíšeme vybrané modely, zhodnotíme jejich výhody, identifikujeme slabiny a diskutujeme jejich použitelné nasazení. Z kategorie geometrických metod se věnujeme jedinému zástupci, kterým je Shape Matching. Z oblasti fyzikálně založených metod se zabýváme zejména Mass-Spring Systémy a metodami konečných prvků. Není-li výslovně uvedeno jinak, předpokládáme, že tělesa jsou reprezentována objemovým meshem.

Na závěr kapitoly provedeme stručné zhodnocení vlastností modelů, jeden z nich zvolíme a v kapitole 6 provedeme odvození všech načrtnutých vztahů do konečných podob.

## 3.1 Shape Matching

Deformace objektů vlivem mechanického působení jiného objektu lze řešit s explicitním použitím geometrických vztahů mezi původním nedeformovaným tvarem objektu a jeho aktuálním tvarem [MHTG05].




Obrázek 3.1: Technika Shape Matching. Obrázky převzaty z [MHTG05]. (a) Základní myšlenka — najít cílový tvar (přerušovanou čarou), do kterého budou body přitahovány. (b) Vylepšení s použitím clusterů.

### 3.1.1 Základní myšlenka

S body reprezentace pracujeme, jako by šlo o jednoduchou PBR. Stav systému je dán polohami $\mathbf{x}_i(t)$ a rychlostmi $\mathbf{v}_i(t)$ bodů $i$ v čase $t$. V každém kroku se nejprve vypočítá tzv. *cílový tvar* objektu. Cílový tvar vznikne posunutím a rotací původního klidového tvaru objektu tak, aby co nejlépe vystihoval aktuální deformovaný tvar. Každá částice je pak přitahována do své cílové polohy. Vzdálenost částice od cílové polohy ovlivňuje rychlost přitahování. Celý objekt se tak snaží vrátit do svého původního tvaru. Myšlenka je zachycena na obrázku 3.1 (a).

Výchozí klidovou polohu bodu $i$ v čase 0 označne $\mathbf{x}_i^0$ a výchozí rychlost bodu $i$ definujme $\mathbf{v}_i(0) = \vec{0}$. V každém kroku simulace hledáme rotační matici $\mathbf{R}$ typu $3 \times 3$ a vektory posunutí $\mathbf{t}$ a $\mathbf{t}_0$ tak, aby transformované souřadnice

$$\mathbf{g}_i = \mathbf{R}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} \tag{3.1}$$

odpovídaly co nejlépe ve smyslu nejmenších čtverců aktuálním souřadnicím $\mathbf{x}_i$. Formálně jde o minimalizaci funkce

$$\sum_i w_i(\mathbf{R}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} - \mathbf{x}_i)^2 \tag{3.2}$$

$w_i$ jsou váhové koeficienty jednotlivých bodů. Mohou jimi být například jejich hmotnosti $m_i$. Lze ukázat, že optimální translační vektory $\mathbf{t}_0$ a $\mathbf{t}$ míří do těžiště původního resp. aktuálního objektu, tedy

$$\mathbf{t}_0 = \frac{\sum_i w_i \mathbf{x}_i^0}{\sum_i w_i} \quad \text{a} \quad \mathbf{t} = \frac{\sum_i w_i \mathbf{x}_i}{\sum_i w_i}$$

K výpočtu optimální rotace použijeme následující úvahu s relativními souřadnicemi. Označme $\mathbf{q}_i = \mathbf{x}_i^0 - \mathbf{t}_0$ a $\mathbf{p}_i = \mathbf{x}_i - \mathbf{t}$. Rovnici (3.2) převedeme na hledání optimální lineární transformace $\mathbf{A}$ minimalizující výraz

$$\sum_i w_i(\mathbf{A}\mathbf{q}_i - \mathbf{p}_i)^2 \tag{3.3}$$

Požadujeme-li, aby derivace funkce v rovnici (3.3) vzhledem ke všem $a_{ij}$ byly rovny nule, dostaneme výsledek

$$\mathbf{A} = \left(\sum_i w_i \mathbf{p}_i \mathbf{q}_i^T\right)\left(\sum_i w_i \mathbf{q}_i \mathbf{q}_i^T\right)^{-1}$$

Hledaná matice $\mathbf{A}$ typu $3 \times 3$ má tvar součinu dvou matic $\mathbf{A} = \mathbf{A}_{pq}\mathbf{A}_{qq}$. Matice $\mathbf{A}_{qq}$ je symetrická, to znamená, že popisuje pouze změnu měřítka, ne rotaci. Optimální rotaci $\mathbf{R}$ lze extrahovat z matice $\mathbf{A}_{pq}$ polární dekompozicí $\mathbf{A}_{pq} = \mathbf{R}\mathbf{S}$, kde symetrická část $\mathbf{S}$ a rotační část $\mathbf{R}$ je pak rovna

$$\mathbf{S} = \sqrt{\mathbf{A}_{pq}^T \mathbf{A}_{pq}} \qquad \text{a} \qquad \mathbf{R} = \mathbf{A}_{pq}\mathbf{S}^{-1}$$

Numericky nejnáročnější je výpočet matice $\mathbf{S}^{-1}$. Ta je ale pouze typu $3 \times 3$, to znamená, že počet operací potřebných k výpočtu $\mathbf{R}$ z $\mathbf{A}$ je konstantní, nezávisí na počtu bodů.

### 3.1.2 Pohyb bodů v čase

Pro každý bod $i$ je udržována rychlost $\mathbf{v}_i$, jakou je do cílové polohy $\mathbf{g}_i$ přitahován. Pro zjištění poloh bodů objektu v čase $t+h$ použijeme vzorce

$$\begin{aligned} \mathbf{v}_i(t+h) &= \mathbf{v}_i(t) + \frac{\alpha}{h}\left(\mathbf{g}_i(t) - \mathbf{x}_i(t)\right) + h\frac{\mathbf{f}_i^e}{m_i} \\ \mathbf{x}_i(t+h) &= \mathbf{x}_i(t) + h\mathbf{v}_i(t+h) \end{aligned} \tag{3.4}$$

$0 < \alpha \leq 1$ je parametr ovlivňující tuhost objektu. Pro vylepšení chování se nepoužívá konstantní $\alpha$, ale $\alpha(h)$ závislé na velikosti kroku. $\mathbf{f}_i^e$ jsou vnější síly, které působí na bod $i$ o hmotnosti $m_i$. Lze ukázat, že celá simulace je bezpodmínečně stabilní (unconditionally stable) pro libovolně velký krok $h$.

### 3.1.3 Lineární a kvadratické deformace

Pomocí výše popsané techniky lze simulovat pouze malé odchylky od původního tvaru. Pokud místo matice $\mathbf{R}$ v rovnici (3.1) použijeme lineární kombinaci $\beta\mathbf{A} + (1-\beta)\mathbf{R}$, kde $\beta$ je uživatelem specifikovatelný kontrolní parametr, může cílový tvar navíc prodělat částečnou lineární transformaci. Pro zachování objemu stačí zajistit, aby $|\mathbf{A}| = 1$.

Celý model lze rozšířit o schopnost cílového tvaru prodělávat kvadratické deformace. Místo původních 3D vektorů $\mathbf{q} = (q_x, q_y, q_z) = \mathbf{x}^0 - \mathbf{t}_0$ použijeme vektory $\tilde{\mathbf{q}} = (q_x, q_y, q_z, q_x^2, q_y^2, q_z^2, q_xq_y, q_yq_z, q_zq_x)$. Matice $\tilde{\mathbf{A}}$ bude mít v tomto případě tři bloky $\tilde{\mathbf{A}} = [\mathbf{A}|\mathbf{Q}|\mathbf{M}] \in \mathbf{R}^{3\times 9}$. $\mathbf{A}$ popisuje koeficienty pro lineární složky, $\mathbf{Q}$ pro kvadratické a $\mathbf{M}$ pro smíšené členy. Minimalizovaná funkce (3.3) má tvar

$$\sum_i w_i(\tilde{\mathbf{A}}\tilde{\mathbf{q}}_i - \mathbf{p}_i)^2$$

a optimální kvadratická transformace je v tomto případě

$$\tilde{\mathbf{A}} = \left(\sum_i w_i\mathbf{p}_i\tilde{\mathbf{q}}_i^T\right)\left(\sum_i w_i\tilde{\mathbf{q}}_i\tilde{\mathbf{q}}_i^T\right)^{-1} = \tilde{\mathbf{A}}_{pq}\tilde{\mathbf{A}}_{qq}$$

Z $\tilde{\mathbf{A}}$ dopočteme $\tilde{\mathbf{R}}$ polární dekompozicí analogicky jako v základní technice a podobně jako v lineárním případě použijeme dodatečný kontrolní koeficient $\beta$ pro úpravu matice $\tilde{\mathbf{R}}$ v rovnici (3.1) — transformace vrcholů do cílového tvaru je popsána maticí $\beta\tilde{\mathbf{A}} + (1-\beta)\tilde{\mathbf{R}}$. Nesmíme zapomenout, že tuto transformaci aplikujeme na body $\tilde{\mathbf{q}}$.

### 3.1.4 Clustery

Celou simulaci lze dále vylepšit rozdělením objektu na *clustery*. Pokud používáme objemový mesh, přirozeně se nabízí použít jako cluster každý čtyřstěn, avšak obecně mohou mít clustery složitější tvary. V každém kroku simulace se pro každý cluster $c$

snažíme najít jeho cílový tvar $\mathbf{g}^c$. Ten vznikne z původního tvaru clusteru $c$ analogickou úvahou, jako předtím pro celý objekt. Označme $\mathbf{g}_i^c$ cílovou polohu bodu $i$ vzhledem ke clusteru $c$. Každý cluster $c$ pak přidá člen

$$\mathbf{v}_i^c = \frac{\alpha}{h}\left(\mathbf{g}_i^c(t) - \mathbf{x}_i(t)\right)$$

všem bodům $i$, které obsahuje. $\alpha$ je týž parametr, jako v (3.4). Na obrázku 3.1 (b) je vidět, jak použití clusterů ovlivňuje výsledný tvar. Čím více clusterů je použito, tím detailnější deformace je, a celkový dojem je vylepšen.

### 3.1.5 Nasazení

I přes všechna popsaná rozšíření má algoritmus stále několik nedostatků. Menší problém představuje fakt, že je navržen především pro modelování elastických deformací. Nemožnost simulovat plastické deformace by bylo možné odstranit pomocí jednoduché úvahy. Tu část deformace, která je trvalého charakteru, využijeme k tomu, aby ovlivnila původní klidový tvar objektu. Cílový tvar, který vychází z původního tvaru, potom bude už ovlivněný touto deformací a celý objekt se bude vracet k deformovanému tvaru.

Největší výhody jsou jednoduchost, rychlost a hlavně bezpodmínečná stabilita algoritmu. Na běžných počítačových sestavách lze tímto způsobem simulovat objekty obsahující zhruba $10^4$ bodů a $10^3$ clusterů.

Další výhodou je snadné nastavení algoritmu. To lze provádět pouze parametrem $\alpha$, který ovlivňuje rychlost přitahování do cílového tvaru a reprezentuje tak tuhost objektu, a parametrem $\beta$, který v jistém smyslu ohraničuje maximální deformovatelnost.

## 3.2 Mass-Spring System

Mass-Spring System je velmi jednoduchý a intuitivní model. Jak napovídá jméno, model se skládá ze sítě hmotných bodů spojených nehmotnými pružinkami. Tím, jak se hmotné body pohybují, natahují nebo stlačují připojené pružinky a ty následně působí silami proti pohybu bodů.

### 3.2.1 Základní myšlenka

Stav systému v čase $t$ je dán polohami $\mathbf{x}_i(t)$ a rychlostmi $\mathbf{v}_i(t)$ uvažovaných hmotných bodů. Uspořádejme polohy $\mathbf{x}_i$ bodů sítě a jejich rychlostí $\mathbf{v}_i$ do vektorů $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ a $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$. V každém kroku se spočte celková síla $\mathbf{f}_i(\mathbf{x}, \mathbf{v})$ působící na hmotný bod $i$ o hmotnosti $m_i$. Tato síla zahrnuje působení všech pružinek, které zde představují vnitřní síly, i vnějších sil. Pohyb každého hmotného bodu $i$ zachovává druhý Newtonův zákon

$$\mathbf{f}_i = m_i \ddot{\mathbf{x}}_i \tag{3.5}$$

Tvar obecné funkce $F_i$, která vystupuje v rovnici 2.2, je v tomto případě

$$F_i(\mathbf{v}, \mathbf{x}, t) = \frac{\mathbf{f}_i(\mathbf{x}, \mathbf{v})}{m_i}$$

Převod rovnice (3.5) na diferenciální rovnice 1. řádu dává každý pro uvažovaný bod $i$ soustavu dvou rovnic

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{v}_i \\ \dot{\mathbf{v}}_i &= \frac{\mathbf{f}_i(\mathbf{x}, \mathbf{v})}{m_i} \end{aligned}$$

Celková síla $\mathbf{f}_i$ působící na bod $i$ má, jak bylo zmíněno, dvě složky, vnější sílu $\mathbf{f}_i^e$ a vnitřní síly $\mathbf{f}_{ij}$ vznikající působením pružinek, které spojují bod $i$ se sousedními body $j$

$$\mathbf{f}_i = \mathbf{f}_i^e + \sum_j \mathbf{f}_{ij}$$

Oproti metodám řešícím rovnice mechaniky kontinua zde máme již přímo soustavy lineárních diferenciálních rovnic. Ty samosebou zbývá vyřešit některým z integračních schemat.

Mass-Spring systémy jsou používány nejčastěji s povrchovými meshi, avšak není vyloučeno použití objemových meshů nebo meshů s komplexními vztahy mezi uzly sítě; síť může obsahovat jednak strukturální pružinky a jednak tzv. *shear springs*, které zabraňují podélným deformacím a zkosení, viz obrázek 3.2 (a). Klidové délky pružinek odpovídají vzdálenostem bodů v nedeformovaném tělesu. Pružinky jsou nejčastěji modelovány podle lineárně elastické teorie. Síla $\mathbf{f}_{ij}$ působící na bod $i$ spojený s bodem $j$ pružinkou, jejíž klidová délka je $l_{ij}$, je dána vzorcem

$$\mathbf{f}_{ij} = k_s \frac{|\vec{\mathbf{x}}_{ij}| - l_{ij}}{|\vec{\mathbf{x}}_{ij}|} \vec{\mathbf{x}}_{ij} \quad \text{kde} \quad \vec{\mathbf{x}}_{ij} = \mathbf{x}_j - \mathbf{x}_i$$

$k_s$ je konstanta tuhosti (stiffness coefficient) pružinky. Jiné modely pružinek jsou používány při modelování tkání např. lidské kůže, protože vykazuje nelineární chování. Síla $\mathbf{f}_{ij}$ zřejmě působí proti změně délky pružinky. Snadno nahlédneme, že $\mathbf{f}_{ij} = -\mathbf{f}_{ji}$.

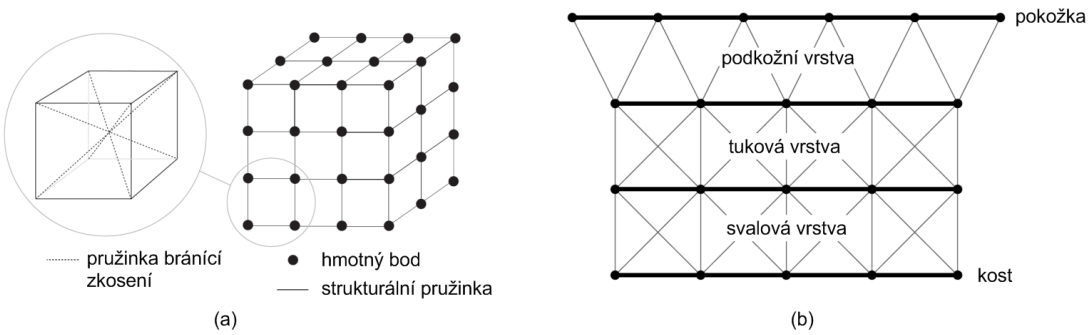
**Činitel útlumu.** Reálná tělesa nejsou dokonale elastická — část energie je při deformaci přeměněna na jiné formy. Proto se zavádí *činitel útlumu* (damping coefficient) pro relativní rychlost. Nejběžnější model přidává k síle $\mathbf{f}_{ij}$ ještě útlum $\mathbf{f}_{ij}^d$ v podobě

$$\mathbf{f}_{ij}^d = k_d \vec{\mathbf{v}}_{ij} \quad \text{kde} \quad \vec{\mathbf{v}}_{ij} = \mathbf{v}_j - \mathbf{v}_i$$

$k_d$ je konstanta ovlivňující míru útlumu. Tento model ale utlumuje i běžné transformace objektu jako tuhého tělesa (RBT) a hůř, při modelování šatů působí i proti ohýbání, což je klíčová vlastnost těchto objektů. Proto je preferováno používat sílu

$$\mathbf{f}_{ij}^d = k_d \frac{\vec{\mathbf{v}}_{ij} \cdot \vec{\mathbf{x}}_{ij}}{\vec{\mathbf{x}}_{ij} \cdot \vec{\mathbf{x}}_{ij}} \vec{\mathbf{x}}_{ij}$$

která korektně promítne rozdíl rychlosti do vektoru mezi oběma body.




Obrázek 3.2: Ukázka různých typů sítí. (a) Pravidelná síť se strukturálními pružinkami a pružinkami, které brání zkosení (*shear springs*). Převzato z [NMK+06]. (b) Síť určená k modelování svalů a pokožky vytvořená z několika vrstev. Struktura sítě i vlastnosti pružinek v jednotlivých vrstvách se liší. Převzato z [GM97].

### 3.2.2 Zobecnění pohledu na pružinky

Na klasický Mass-Spring Systém, kde pružinky spojují vždy pouze dva hmotné body, lze pohlížet obecněji [THMG04]. Zavedeme nové typy vztahů, tzv. *omezující podmínky*, které spojují dva a více hmotných bodů současně. Deformační model je založen na deformačních energiích, které kvantifikují penále za porušení omezujících podmínek. Tento přístup se dobře hodí pro povrchové a objemové meshe; obecné vztahy mohou být nastaveny tak, aby bránily nejen změně vzdáleností mezi body, ale i změně povrchu trojúhelníků a/nebo úhlů, které spolu trojúhelníky svírají, nebo změně objemu čtyřstěnů.

Nechť vektor $\mathbf{i}$ označuje $m$-tici bodů $(i_1, \ldots, i_m)$ a vektor $\mathbf{y} = (\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_m})$ jejich polohy. Řekneme, že $m$-tice bodů $\mathbf{i}$ tvoří *element* $\mathbf{i}$. Pro každý typ elementu, který závisí na počtu bodů, mějme dánu skalární funkci $C(\mathbf{y})$. Funkce $C$, kterou nazýváme *omezující podmínkou*, je nulová pouze v případě, že body elementu jsou navzájem v původní relativní klidové poloze. Potenciální deformační energie $E$ asociovaná s porušením této podmínky — s odchýlením vrcholů od původních relativních klidových poloh — je dána vztahem

$$E(\mathbf{y}) = \frac{1}{2} k_C C^2(\mathbf{y}) \tag{3.6}$$

Koeficient $k_C$ upravuje sílu významu konkrétní podmínky $C$. Čím větší $k_C$, tím větší penalizace za porušení této podmínky.

Protože integrační schéma nepracuje přímo s energiemi, ale se silami, je potřeba odvodit vztah mezi deformační energií a silami, kterými deformovaný element působí na své jednotlivé vrcholy. Vnitřní síla $\mathbf{f}^{\mathbf{i}}_j(\mathbf{y})$ působící na bod $j \in \mathbf{i}$ vlivem deformační energie $E$ elementu $\mathbf{i}$ je

$$\mathbf{f}^{\mathbf{i}}_j(\mathbf{y}) = -\frac{\partial E}{\partial \mathbf{x}_j}(\mathbf{y}) = -kC(\mathbf{y})\frac{\partial C}{\partial \mathbf{x}_j}(\mathbf{y}) \tag{3.7}$$

Směr síly $\mathbf{f}_j^{\mathbf{i}}$ působí proti gradientu energie $E$, což znamená, že simulace inkorporující tyto síly usiluje o snížení deformační energie. Celková vnitřní síla $\mathbf{f}_j$ působící na bod $j$ je součtem všech sil $\mathbf{f}_j^{\mathbf{i}}$ přes všechny elementy $\mathbf{i}$ tž. $j \in \mathbf{i}$. Je tedy součtem silových příspěvků všech elementů, jejichž částí bod $j$ je. Výsledná síla navíc zachovává hybnost a moment hybnosti (linear and angular momentum).

**Činitel útlumu.** Pro zvýšení robustnosti modelu se zavádí útlum síly. Jestliže $\mathbf{v}_j$ značí rychlost bodu $j$, potom vnitřní síla $\mathbf{f}_j^{\mathbf{i}}$ nemá tvar (3.7), ale

$$\mathbf{f}_j^{\mathbf{i}}(\mathbf{y}) = \left( -k_C C - d_C \sum_k \mathbf{v}_k \cdot \frac{\partial C}{\partial \mathbf{x}_k} \right) \frac{\partial C}{\partial \mathbf{x}_j} \tag{3.8}$$

$d_C$ je koeficient útlumu.

**Zachování vzdálenosti.** Konkrétní podmínka, která zaručuje zachování původní vzdálenosti $l_{ij}$ dvou hmotných bodů $i$ a $j$, má tvar

$$C_D(\mathbf{x}_i, \mathbf{x}_j) = \frac{|\vec{\mathbf{x}}_{ij}| - l_{ij}}{l_{ij}} \tag{3.9}$$

Značení $\vec{\mathbf{x}}_{ij}$ pro vektor $\mathbf{x}_j - \mathbf{x}_i$ budeme používat i v dalších podmínkách.

**Zachování plochy.** Podmínka, která zaručuje zachování velikosti povrchu trojúhelníku tvořeného body $i$, $j$ a $k$, který má klidovou plochu velikosti $A_{ijk}$, má tvar

$$C_A(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) = \frac{\frac{1}{2}(\vec{\mathbf{x}}_{ij} \times \vec{\mathbf{x}}_{ik}) - A_{ijk}}{A_{ijk}} \tag{3.10}$$

Tuto podmínku je výhodné zahrnout při simulacích tenkých plátů. Naopak, pokud pracujeme s objemovými meshi a usilujeme spíš o zachování objemu, je její vliv zanedbatelný. Zatímco útlum síly odvozené z podmínky (3.9) má pozitivní vliv na stabilitu simulace, při praktických experimentech se ukazuje, že útlum sil odvozených z jiných podmínek jako třeba (3.10) naopak téměř žádný vliv nemá. Proto se pro odvození síly z (3.10) používá pouze vztah (3.7).

**Zachování objemu.** Analogicky lze zkonstruovat podmínku pro zachování objemu čtyřstěnu tvořeného body $i$, $j$, $k$, $l$ s klidovým objemem $V_{ijkl}$. Ta má tvar

$$C_V(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) = \frac{\frac{1}{6}\left(\vec{\mathbf{x}}_{ij} \cdot (\vec{\mathbf{x}}_{ik} \times \vec{\mathbf{x}}_{il})\right) - V_{ijkl}}{V_{ijkl}} \tag{3.11}$$

V případě, že používáme tuto podmínku při simulaci tenkých plátů, doporučuje se pro dosažení lepších výsledků použít mírně odlišný tvar předešlé rovnice

$$C_V(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) = \frac{\frac{1}{6}\left(\vec{\mathbf{x}}_{ij} \cdot (\vec{\mathbf{x}}_{ik} \times \vec{\mathbf{x}}_{il})\right) - V_{ijkl}}{\tilde{V}_{ijkl}}$$

kde $\tilde{V}_{ijkl} = \frac{\sqrt{2}}{12}\tilde{l}$, přičemž $\tilde{l}$ označuje průměrnou délku hrany čtyřstěnu. $\tilde{V}_{ijkl}$ je pak objem pravidelného čtyřstěnu s hranou délky $\tilde{l}$. Za povšimnutí stojí, že síly odvozené z (3.11) se snaží zachovat orientaci čtyřstěnu, brání jeho invertování.

### 3.2.3 Nasazení

Původně byl Mass-Spring System určen pro modelování výrazů obličeje a byl představen pod názvem *tension nets*. Později byly vyvinuty modely kůže, tuku, svalů. Ukázka sítě, která se k těmto účelům používala, je na obrázku 3.2 (b). Klidové délky pružinek se v tomto případě vždy dynamicky měnily podle situace např. podle stahu svalu. Velké obliby dosáhla tato metoda zejména při modelování šatů. Především proto, že šaty jsou tvořeny sítí vláken a představa sítě pružinek jí velmi dobře přiléhá.

Přestože se Mass-Spring System jeví jako velmi intuitivní a je snadno implementovatelný, má několik nedostatků. Především není fyzikálně věrný, neboť nestaví přímo na teorii elasticity; simulace nekonverguje k reálnému stavu, tj. k tomu, jak by to bylo ve skutečnosti. Nehodí se tudíž k aplikaci v oblastech, kde je vyžadována maximální fyzikální realističnost. Chování modelu je silně závislé jednak na použitých typech pružinek a jednak na hustotě sítě. Čím jemnější síť, tím přesvědčivější výsledky. Problém s hustotou sítě lze řešit pomocí adaptivního zjemňování v oblastech s vysokou křivostí, zde ale narážíme na jiná úskalí týkající se toho, jak korektně rozdělit danou oblast a jak získané výsledky konsolidovat se zbylou, nezjemněnou částí sítě.

Další závažný problém souvisí s koeficienty tuhosti. Je složité najít korespondenci mezi nimi a materiálem, který je simulován. Navíc se objevují problémy s numerickou stabilitou, zvlášť pokud se jednotlivé koeficienty tuhosti hodně liší. Velké koeficienty jsou potřeba v oblastech, kde je materiál téměř rigidní nebo kde je zapotřebí namodelovat nějaká omezení; například tenké pláty (thin shells), které jsou odolné vůči ohybu, je složitější namodelovat pomocí Mass-Spring Systému. Koeficienty se mohou lišit i v důsledku snahy o modelování anizotropního materiálu. V takových případech je potřeba simulovat pouze s malým časovým krokem, což se negativně promítá do rychlosti simulace. Tyto problémy jsou intenzivně studovány.

Pro některé aplikace je však dosažitelná přesnost postačující a poskytuje dostatečně přesvědčivé výsledky — zejména pro filmový nebo herní průmysl. V mnoha případech simulace běží interaktivně i na dnešních stolních počítačích, což s jinými fyzikálně založenými metodami je obtížný úkol. Právě díky rychlosti, přestože neposkytují dokonale přesné výsledky, jsou stále využívány i při simulacích chirurgických zákroků.

Díky své povaze lze v Mass-Spring Systémech využít paralelního zpracování, které zvlášť v dnešní době dostupných vícejádrových procesorů nebo programovatelných grafických akcelerátorů nabírá na významu, neboť jejich výkon rychle roste a jsou přímo navrženy pro rychlé paralelní zpracování dat [MHS05]. Interaktivně lze simulovat sítě s řádově $10^4$ hmotných bodů, což je dostačující množství i pro vnitřní orgány, jako např. srdce.

## 3.3 Metoda konečných prvků

Metoda konečných prvků (Finite Element Method, FEM) je populární přístup k diskretizaci objektu při řešení diferenciálních rovnic v numerické matematice. Objekt původně chápaný jako spojitá souvislá část prostoru — kontinuum — je aproximován konečným počtem disjunktních elementů, čímž vznikne objemová síť s uzlovými body $\mathbf{m}_i$. Nemusí jít přímo o objemový mesh — tvar elementů závisí na požadavcích kladených na rychlost a přesnost konvergence, počet stupňů volnosti či přesnost aproximace původního modelu. Obecně platí, že aproximace elementy s větším počtem uzlových bodů vyžaduje celkově méně elementů pro dosažení stejného stupně přesnosti. Nejpoužívanější typy elementů jsou čtyřstěny a kvádry. Na obrázku 3.3 (b) jsou zobrazeny dva různé druhy elementů.

### 3.3.1 Základní myšlenka

Diferenciální rovnice (2.2) se upraví následujícím způsobem. Místo hledání spojitého řešení $\mathbf{x}(\mathbf{m}, t)$ se řešení hledá pro konečnou množinu uzlových bodů $\mathbf{x}_i(t) = \mathbf{x}(\mathbf{m}_i, t)$ dané sítě. Funkce $\mathbf{x}(\mathbf{m}, t)$ je hodnotami v uzlových bodech aproximována použitím tzv. *uzlových bázových funkcí* $b_i(\mathbf{m})$

$$\tilde{\mathbf{x}}(\mathbf{m}, t) \approx \sum_i \mathbf{x}_i(t) b_i(\mathbf{m}) \tag{3.12}$$

Předem zvolené uzlové bázové funkce $b_i(\mathbf{m})$ (nodal basis functions, též shape functions), jsou závislé na tvaru elementů a zpravidla splňují následující tři podmínky

$$\begin{aligned} \sum_i b_i(\mathbf{m}) &= 1 \qquad \forall \mathbf{m} \\ b_i(\mathbf{m}) &\geq 0 \qquad \forall \mathbf{m} \\ b_i(\mathbf{x}_j) &= \delta_{ij} \qquad \forall \mathbf{x}_j \end{aligned}$$

Dosazením aproximace (3.12) do (2.2) dostaneme diferenciální rovnici o neznámých $\mathbf{x}_i(t)$

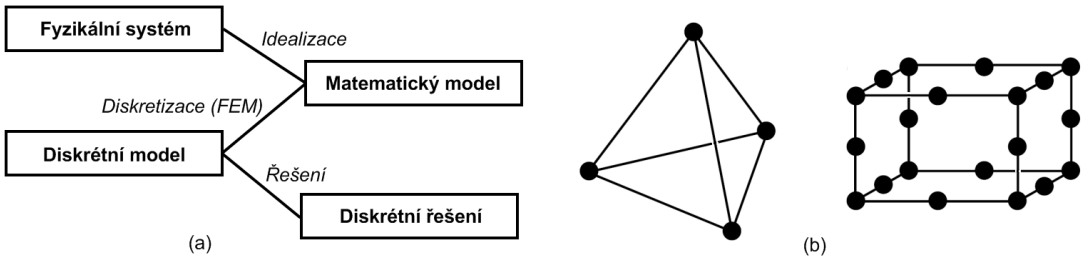
$$\sum_i \ddot{\mathbf{x}}_i(t) b_i(\mathbf{m}) = F\left( \sum_i \dot{\mathbf{x}}_i(t) b_i(\mathbf{m}), \sum_i \mathbf{x}_i(t) b_i(\mathbf{m}), t \right)$$

Substitucí funkce $\tilde{\mathbf{x}}$ za hledanou funkci $\mathbf{x}$ je nekonečně dimenzionální prostor řešení redukován na konečně dimenzionální podprostor. Obecně žádná funkce z tohoto podprostoru neřeší původní soustavu přesně. Odchylka vzniklá dosazením aproximace $\tilde{\mathbf{x}}$ do parciální diferenciální rovnice se nazývá reziduum $R(\tilde{\mathbf{x}})$

$$\ddot{\tilde{\mathbf{x}}} = F(\dot{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}, t) + R(\tilde{\mathbf{x}})$$

Na hledání neznámých $\mathbf{x}_i$ lze nahlížet jako na optimalizační proces, jehož cílem je nalézt takové polohy, které reziduum minimalizují. Tento proces je ovšem časově velmi náročný.

Metoda konečných prvků diskretizuje diferenciální rovnice pouze *prostorově* — to znamená, že nekonečný počet proměnných symbolizujících polohu kontinua aproximuje pouze konečným počtem. K *časové* diskretizaci, k diskrétnímu řešení úlohy, je samosebou zapotřebí použít některého integračního schématu. Roli FEM v procesu modelování fyzikálních problémů zachycuje obrázek 3.3 (a).




Obrázek 3.3: Metoda konečných prvků. (a) Role FEM při modelování fyzikálních problémů. (b) Různé druhy elementů — vlevo nejpoužívanější čtyřstěny, vpravo 20ti-uzlový kvádr. Převzato z [GM97].

### 3.3.2 Explicitní metoda konečných prvků

Kvůli výpočetním nárokům FEM se objevily zjednodušující techniky. Jednou z nich je *explicitní FEM* [OH99]. Nejde však o FEM integrovanou explicitními schématy — explicitní FEM lze integrovat jak implicitně, tak explicitně. Vnitřní síly působící v uzlových bodech se spočtou lokálně pouze pomocí hodnot sousedních uzlů. Zjednodušeně lze říct, že uzly sítě jsou chápány jako hmotné body, obdobně jako u Mass-Spring Systému. Každý element sítě se pak chová jako *zobecněná pružinka* spojující uzlové body; při deformaci pak element na každý svůj uzlový bod působí silou.

Předpokládejme, že kontinuum v klidovém stavu je aproximováno sítí čtyřstěnů. Vyberme jeden čtyřstěn $T$ s uzly $\mathbf{m}_1, \ldots, \mathbf{m}_4$. Pro bod $\mathbf{m} = (x_1, x_2, x_3)$, který se nachází v elementu $T$, určeme jeho barycentrické souřadnice $\beta(\mathbf{m}) = (\beta_1, \beta_2, \beta_3, \beta_4)$. Lze tedy psát $(x_1, x_2, x_3, 1) = \mathbf{M} \cdot \beta(\mathbf{m})$. $\mathbf{M}$ je matice tvořená sloupcovými vektory $(\mathbf{m}_i, 1)$. Označíme-li $\mathbf{B} = \mathbf{M}^{-1}$, pak lze pro bod $\mathbf{m}$ uvnitř elementu $T$ psát $\beta(\mathbf{m}) = \mathbf{B} \cdot (x_1, x_2, x_3, 1)$.

Předpokládejme, že známe světové souřadnice uzlových bodů $\mathbf{x}_i = \mathbf{x}(\mathbf{m}_i)$. Označme $\mathbf{P}$ matici tvořenou vektory $(\mathbf{x}_i, 1)$ a $\mathbf{V}$ matici tvořenou vektory $(\mathbf{v}_i, 1)$. Barycentrické souřadnice lze použít k interpolaci světových souřadnic $\mathbf{x}(\mathbf{m})$ a rychlosti $\mathbf{v}(\mathbf{m})$ bodů $\mathbf{m}$ uvnitř $T$.

$$\mathbf{x}(\mathbf{m}) = \mathbf{P} \cdot \beta(\mathbf{m}) \qquad \text{a} \qquad \mathbf{v}(\mathbf{m}) = \mathbf{V} \cdot \beta(\mathbf{m}) \tag{3.13}$$

Výpočet složek tenzorů $\varepsilon$ a $\nu$ lze provést s použitím aproximací (3.13)

$$\frac{\partial \mathbf{x}}{\partial x_i} = \mathbf{P} \cdot \mathbf{B} \cdot \delta_i \qquad \text{a} \qquad \frac{\partial \dot{\mathbf{x}}}{\partial x_i} = \mathbf{V} \cdot \mathbf{B} \cdot \delta_i$$

kde $\delta_i$ je vektor $[\delta_{i1}, \delta_{i2}, \delta_{i3}, 0]$. Z tenzoru deformace $\varepsilon$ a tenzoru změny deformace $\nu$ dopočteme i tenzor napětí $\sigma^\varepsilon$ resp. $\sigma^\nu$. Složky všech tenzorů jsou konstantní v celém elementu, protože byla použita lineární interpolace barycentrickými souřadnicemi.

Každý element vlivem deformace na své uzlové body působí elastickou a tlumicí silou. Elastická síla $\mathbf{f}_i^\varepsilon$ a tlumicí síla $\mathbf{f}_i^\nu$ působící na uzel $i$ je odvozena z elastické deformační energie $E_P$ elementu resp. z potenciálu tlumení $E_P^\nu$ elementu podle rovnic

$$\mathbf{f}_i^\varepsilon = -\frac{\partial E_P}{\partial \mathbf{x}_i} \qquad \text{a} \qquad \mathbf{f}_i^\nu = -\frac{\partial E_P^\nu}{\partial \mathbf{x}_i}$$

a díky linearitě interpolačních funkcí lze snadno dojít ke tvaru

$$\begin{aligned} \mathbf{f}_i^\varepsilon &= -\frac{1}{2} V_T \cdot \sum_j \mathbf{x}_j \sum_{k,l} B_{jl} B_{ik} \sigma_{kl}^\varepsilon \\ \mathbf{f}_i^\nu &= -\frac{1}{2} V_T \cdot \sum_j \mathbf{x}_j \sum_{k,l} B_{jl} B_{ik} \sigma_{kl}^\nu \end{aligned}$$

$V_T$ označuje původní objem elementu $T$.

Vnitřní sílu $\mathbf{f}_i^T$, kterou deformovaný element $T$ působí na uzel $i$, dostaneme jako součet $\mathbf{f}_i^T = \mathbf{f}_i^\varepsilon + \mathbf{f}_i^\nu$. Celkovou vnitřní sílu $\mathbf{f}_i$, která působí na uzel $i$ sítě, spočteme jako součet příspěvků od všech elementů, které jsou s daným uzlem spojeny.

### 3.3.3 Nasazení

Využití FEM v počítačové grafice bylo zpočátku limitováno její výpočetní složitostí. Jevilo se, že její nasazení v interaktivních aplikacích je většinou nemožné. Postupně byly však předvedeny techniky, které za určitých dodatečných předpokladů urychlují potřebné výpočty, zejména jsou-li modelované objekty dostatečně jednoduché nebo je působení vnějších sil relativně lokalizované. Mezi tyto techniky patří časově a prostorově adaptivní zjemňování modelů, které pro některé oblasti, kde aktuálně nedochází k velkým změnám, používá hrubší reprezentaci.

Pro ještě větší urychlení výpočtů se objevují úspěšné pokusy akcelerovat FEM na programovatelném grafickém hardware použitém jako koprocesor [GST05]. Tímto způsobem lze dosáhnout až pětinásobného urychlení, nebo zvýšení realističnosti simulace (snížením kroku $h$). Jedinou nevýhodou grafických akcelerátorů je podpora pouze half (s10e5) a single (s23e8) precision floating point formátů.

Spolu s výpočetní náročností je dalším omezujícím faktorem konstrukce plně objemové diskretizace objektu; FEM generují pro výpočty objemový mesh, tudíž je zapotřebí vetší počet uzlů oproti jiným metodám. V současnosti lze v reálném čase modelovat tělesa obsahující řádově $10^3$ elementů.

FEM produkuje realistické a vizuálně velmi přesvědčivé výsledky. Uvádí se, že pro dosažení kvalitativně srovnatelných výsledků s Mass-Spring Systémem je potřeba menší počet uzlových bodů. Použitím vhodné rovnice (a případného dalšího rozšíření) lze modelovat nejen elastické a plastické objekty, ale i složitější jevy jako lámání a praskání objektů. Díky fyzikálnímu základu lze chování materiálů ovlivňovat prostřednictvím experimentálně naměřených konstant, jako jsou hustota, Youngův modul pružnosti nebo Poissonův poměr (Poisson Ratio).

FEM se uplatňují především v oblastech, kde je zapotřebí fyzikální věrnosti. Objevují se v medicínských aplikacích při modelování svalů, pokožky nebo vnitřních orgánů [WDGT01], které vykazují komplexní nelineární viskoelastické chování, nebo při tréninku chirurgických zákroků a při analýze materiálu.

## 3.4 Shrnutí

Představené techniky rozhodně netvoří kompletní výčet deformovatelných modelů pevných těles. Celá problematika je enormně obsáhlá, pro přehled nejpoužívanějších metod odkazujeme na [GM97] a zejména na novější a úplnější [NMK+06]. Existují deformovatelné modely, které se specializují podle typu

- deformace — plastické, elastické, praskání
- reprezentace — MR, PBR, parametrické plochy
- objektu — šaty, tkáně, automobily
- materiálu — zrnité, elastické, viskózní
- vnějších vlivů — mechanické, elektromagnetické, tepelné
- nasazení — virtuální prostředí, segmentace obrazu, rekonstrukce povrchu

Všechny modely se navzájem více či méně liší a dávají různě dobré výsledky.

Vývoj v oblasti deformovatelných modelů má dlouhou historii a velmi pravděpodobně bude mít i velkou budoucnost. Jen z představených technik je zřejmé, že mají mnoho nedostatků a jsou omezeny na relativně jednoduché objekty, které obsahují řádově tisíce primitiv.

Z představených technik lze extrahovat schéma, které poskytuje zjednodušený, nicméně výstižný pohled na konkrétní kroky během simulace.

1. pro daný objekt vypočti vnitřní síly $\mathbf{f}_i$, které působí uvnitř tělesa
2. vyhodnoť externí síly $\mathbf{f}_i^e$, které na těleso působí
3. použij integrační schéma pro aktualizaci poloh $\mathbf{x}_i$ bodů deformovatelného modelu při uvážení sil $\mathbf{f}_i + \mathbf{f}_i^e$.

Konkrétní volba jedné z popsaných technik je poměrně těžká, protože všechny splňují požadavky kladené na rychlost a věrohodnost. Rozumným kompromisem mezi dosažitelnou rychlostí, nastavitelností, robustností, věrohodností a složitostí se jeví zobecněný Mass-Spring System. Přesný popis nastavení deformovatelného modelu, odvození konečných podob použitých vzorců a strukturu sítí simulovaných těles rozebereme v kapitole 6.

# Kapitola 4

# Detekce kolizí

Zabýváme-li se simulacemi deformací pevných těles vlivem mechanického působení jiného objektu, je zapotřebí rozšířit schopnosti simulátoru — musí být schopen poznat, zda jsou objekty v kontaktu, a pokud ano, kde přesně se stýkají. *Detekce kolizí* (collision detection, často i contact determination, CD), je proces, jehož cílem je najít a ohlásit geometrický kontakt mezi dvěma nebo více tělesy, která se střetnula. V mnoha aplikacích jde o časově nejnáročnější část výpočtů (computation bottleneck).

Detekce kolizí je studována pro všechny typy reprezentací pevných těles, omezíme se pouze na objekty reprezentované meshem. Detekce kolizí musí být obecně schopna zodpovídat různé typy dotazů, nás bude především zajímat, zda dva modely kolidují (zda mají neprázdný průnik), a pokud ano, které přesně jejich části to jsou a jak hluboko se nacházejí uvnitř druhého tělesa.

Většina algoritmů byla navržena pro dokonale tuhá tělesa a soustředila se na rychlé provádění tzv. *rejection testů*. Jde o testy, jejichž cílem je identifikovat dvojice objektů nebo jejich částí, které se vzájemně neprotínají. Zatímco detekce kolizí pro tuhá tělesa je dnes dobře prostudována, detekce kolizí pro deformovatelné objekty musí navíc brát v potaz následující aspekty:

- kolize objektů se sebou samými (self-collision)

  aby bylo možné realisticky simulovat deformovatelné objekty, musí být řešeny nejenom kolize s dalšími objekty scény, ale i s objektem samotným

- omezené možnosti předzpracování scény (limited preprocessing)

  speciální datové struktury používané při práci algoritmu musí umožňovat své rychlé aktualizace, protože zpracovávaná data se často mění

- detailnější informace o kolizi

  algoritmy pro simulaci deformovatelných objektů musí být schopny adekvátně a realisticky reagovat na kolize, a k tomu mohou vyžadovat daleko více informací o kolizi (např. o hloubce penetrace objektu ve všech klíčových bodech)

Ve zbytku kapitoly stručně představíme nejznámější algoritmy pro detekci kolizí mezi deformovatelnými objekty a budeme se zabývat jejich výhodami a omezeními. Podle nich vybereme jeden algoritmus, který použijeme v navrhovaném řešení (kapitola 6), kde bude podrobně rozebrán.

## 4.1 Hierarchie obalových těles

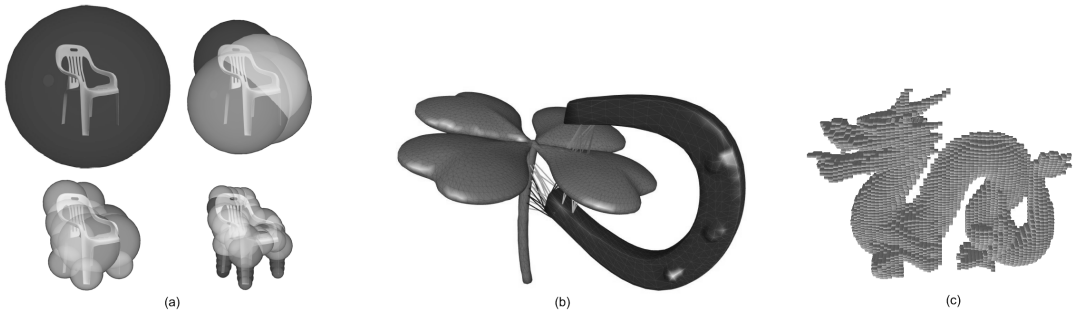
*Hierarchie obalových těles* (Bounding Volume Hieararchy, BVH) se osvědčila jako jedna z nejefektivnějších datových struktur pro detekci kolizí mezi tuhými tělesy. BVH se konstruuje pro každý objekt scény zvlášť během fáze předzpracování. Základní myšlenka je dělit objekt (respektive množinu primitiv objektu) rekurzivně na menší části a pro ně konstruovat obalová tělesa, která je dobře aproximují, dokud není dosaženo určité koncové podmínky.

Každý uzel hierarchie je tedy asociován s určitou podmnožinou primitiv zpracovávaného objektu a sadou *obalových těles* (bounding volumes, BV), která tuto podmnožinu uzavírají v tom smyslu, že nějaký předem zvolený tvar všechna primitiva obsahuje a má přitom nejmenší možný objem. Jedním z rozhodnutí při návrhu BVH je volba třídy tvarů. Nejpoužívanější jsou koule, kvádry s hranami rovnoběžnými s osami souřadného systému (Axis-Aligned Bounding Box, AABB), obecné kvádry (Oriented Bounding Box, OBB) nebo diskrétní orientované polytopy (Discrete Oriented Polytope, DOP). Ukázka několika úrovní hierarchie obalových koulí je na obrázku 4.1 (a).

Průchod vytvořenou hierarchií při detekci kolizí mezi dvěma objekty probíhá shora dolů. Dvojice uzlů jsou vzájemně rekurzivně testovány na překrytí svých obalových těles, dokud oba dva porovnávané uzly nejsou listy, porovnávají se mezi sebou jejich děti navzájem. V případě, že v obou hierarchiích dospějeme k listům, testují se navzájem všechna primitiva v nich uložená.

Při tvorbě BVH pro tuhá tělesa je cílem zkonstruovat hierarchii tak, aby všechny dotazy, zda tělesa kolidují, byly zodpovězeny, jak nejrychleji je to možné. Pro deformovatelné objekty je situace poněkud jiná; cílem je zkonstruovat takovou BVH, kterou lze snadno aktualizovat během toho, jak se objekt deformuje. Během předzpracování se zkonstruuje BVH pro deformovatelný objekt, jako kdyby šlo o tuhé těleso, a během simulace se struktura stromu udržuje, mění se většinou jenom vlastnosti obalových těles v zasažených větvích. Alternativně můžeme buď celou strukturu vystavět znova, což je extrémně nákladné, nebo přestavět pouze zasažené větve. Aktualizační techniku je zapotřebí volit s ohledem na rozsah deformací, protože výsledná hierarchie může primitiva obalovat mnohem hůře a díky tomu se začne snižovat výkon.

BVH provádí velmi efektivně rejection testy, kdykoli jsou objekty poměrně vzdáleny. Jakmile se však objekty ocitnou v těsné blízkosti, výkon BVH klesá, protože velké množství BV musí být testováno mezi sebou navzájem. BVH se pro deformovatelné objekty vyplatí pouze v případě, že je deformace pouze lokálního charakteru. Globální deformace mohou vyžadovat kompletní rekonstrukci hierarchie.




Obrázek 4.1: Ukázka různých datových struktur detektorů kolizí. (a) Několik různých úrovní BVH tvořené koulemi [JP04]. (b) Množina aktivních dvojic při stochastické detekci kolizí [GD04]. (c) LDI reprezentace vypočtená pro standfordského draka [HTG04].

## 4.2 Stochastické metody

Nasazení pravděpodobnostních metod je motivováno několika pozorováními. Za prvé, MR je typicky pouze aproximací skutečné geometrie reprezentovaného tělesa. Za druhé, vnímaná kvalita většiny interaktivních 3D aplikací nezávisí na naprosto přesné simulaci, ale rychlé odezvě na kolize. Lidský mozek nedokáže často odlišit simulaci fyzikálně korektní od fyzikálně věrohodné, jak zmiňuje sekce 2.1. Můžeme tedy tolerovat zrychlení výkonu detektoru kolizí na úkor drobných nepřesností. Nastíníme princip jednoho z takových algoritmů — detekce kolizí Monte Carlo [GD04].

Během simulace se pro každou dvojici objektů ve scéně vytváří a udržuje množina dvojic tzv. *vzorků* (samples) nebo *příznaků* (features), viz obrázek 4.1 (b). Příznakem může být prakticky kterákoli část objektu — hrany, vrcholy nebo povrchové polygony. Nová dvojice $(q, r)$ vzorků mezi objekty $A$ a $B$ vznikne tak, že se náhodně vybere příznak $p \in A$, najde se příznak $q \in B$, který je příznaku $p$ nejblíž, a pro příznak $p$ se najde nejbližší příznak $r \in A$.

V každém kroku simulace se tímto způsobem vytváří pouze omezený počet nových dvojic vzorků. Tento počet lze průběžně dolaďovat během simulace — větší počet párů snižuje rychlost, ale zvyšuje pravděpodobnost odhalení kolize. Kvůli efektivitě celého procesu je zapotřebí seznam aktivních dvojic prořezávat a udržovat pouze ty nejzajímavější. Při aktualizaci dvojic během dvou po sobě jdoucích časových kroků $t_1$ a $t_2$ lze efektivně využít tzv. *časové koherence* (temporal coherence): jsou-li dva vzorky blízko v čase $t_1$, budou si pravděpodobně blízko i v čase $t_2$. Při této aktualizaci se zpravidla používá pouze bezprostřední okolí vzorků.

Pro rychlou stochastickou detekci kolizí se ukazuje, že je velmi výhodné využívat hierarchii příznaků. Jsou-li objekty v některých oblastech hodně vzdáleny, používá se nejhrubší úroveň příznaků. Jak se objekty postupně přibližují, využívá se v oblastech, které jsou si blízko, jemnější úroveň, čímž se zpřesňuje detekce případných kolizí.

Zásadní výhodou tohoto algoritmu jsou především mizivé nároky na paměť a dynamicky ovlivnitelná složitost detekce kolizí. Nevýhodou je zejména fakt, že pro efektivní nasazení je potřeba hierarchie příznaků. Tu ale není nutno během simulace přestavovat.

## 4.3 Dělení prostoru

Celý prostor je rozdělen na množinu vzájemně se nepřekrývajících buněk tvořících mřížku. Algoritmus klasifikuje vybraná primitiva reprezentace podle dané mřížky — každá buňka obsahuje seznam primitiv, která do ní patří a musí být vzájemně testována na průnik. Jestliže buňka obsahuje více primitiv téhož objektu, mohou se testovat i spolu navzájem, čímž je elegantně vyřešen problém detekce kolizí objektu sama se sebou.V případě dynamických a deformovatelných objektů je potřeba obsahy buněk v každém kroku aktualizovat.

Rozdělení prostoru pro detekci kolizí mezi tuhými tělesy lze realizovat mnoha způsoby — nejen pomocí pravidelných mřížek, ale i octrees, BSP stromů, k-D stromů; tyto datové struktury snižují paměťové nároky oproti pravidelné mřížce, ale za cenu nárůstu složitosti aktualizace pro dynamické a deformovatelné objekty, protože jsou velmi závislé na tvaru objektu. Oproti tomu dělení prostoru mřížkou na objektu vůbec závislé není a tudíž se lépe hodí pro deformovatelné objekty. Speciální způsob uložení 3D mřížky je pomocí hashovací tabulky [THM+03]. Indexy jednotlivých buněk mohou být chápány jako klíče. Výhodou tohoto přístupu je ušetření paměti oproti celé pevné mřížce a absence složitých datových struktur v porovnání s ostatními technikami.

**Optimalizované prostorové hashování.** Algoritmus využívající optimalizovaného prostorového hashování postupuje poněkud odlišně, protože netestuje na průnik stejné typy primitiv reprezentace. Navíc je přímo navržen pro práci s objemovými meshi. Detekce kolizí v každém kroku probíhá ve dvou fázích. Nejprve jsou všechny vrcholy objektů klasifikovány podle dané mřížky a uloženy do jednodimenzionální hash tabulky. V druhé fázi jsou klasifikovány všechny čtyřstěny objektů podle téže mřížky a následně testovány na průnik s body, které se nachází v čtyřstěnem zasažených buňkách. Pro rychlé nalezení seznamu buněk, které čtyřstěn protíná, se používá jeho AABB obal. Tento obal lze pak navíc použít pro urychlení testu, zda se bod buňky nachází uvnitř čtyřstěnu nebo ne.

Tato technika netestuje vzájemný průnik čtyřstěnů, tedy může dojít k zanedbání těch stavů, kdy se dva čtyřstěny protínají stěnami, ale žádný jejich vrchol se nenachází uvnitř druhého. Z hlediska korektnosti je to problém, ale uvážíme-li, že výpočty zahrnující tuto situaci jsou zpravidla velmi drahé a výrazně snižují výkon algoritmu, nestojí za to je v případě rozumně hustě vzorkovaných objektů uvažovat.

## 4.4 Detekce kolizí v prostoru obrazu

Rostoucí výkon grafických akcelerátorů se snaží zužitkovat speciální techniky pracující *v prostoru obrazu* (image space), které zpracovávají projekce objektů pro urychlení detekce kolizí. Nevyžadují zpravidla žádné předzpracování a jeví se tedy vhodné pro deformovatelné objekty.

**LDI technika.** Původní myšlenka byla omezena na konvexní tělesa. Každé těleso je vyrenderováno do z-bufferu (depth bufferu); používá se vždy stejná ortogonální projekce. Výsledné z-buffery se po dvou porovnávají pixel po pixelu a interval od menší z-hodnoty k větší se dá chápat jako aproximace vzdálenosti obou objektů. Pokročilejší techniky využívají *Layered Depth Image* (LDI), což je datová struktura, která uchovává v každém pixelu setříděný seznam všech hodnot, které do něj byly zapsány, nejen tu aktuálně nejbližší. LDI klasifikuje prostor na vnitřní a vnější oblasti vzhledem k reprezentovanému objektu. Tato paměťová struktura je relativně malá. Ukázka této struktury je na obrázku 4.1 (c).

Algoritmus pro hledání kolizí pracuje tak, že na grafickém akcelerátoru generuje LDI [HTG04]. Generování LDI je časově náročný úkol, objekt se musí renderovat několikrát podle maximálního počtu polygonů, které přispívají v jednom pixelu (depth complexity). Zásadní problém představuje fakt, že fragmenty jsou renderovány v předem neznámém pořadí a je nutné výsledky z LDI správně uspořádat. Průnik objektů se detekuje průchodem jejich LDI reprezentací pixel po pixelu. Při experimentech se ale ukazuje, že čisté CPU implementace mohou poskytovat za určitých okolností mnohem větší výkon.

**Technika z-fail.** Protože propustnost sběrnice mezi grafickým akcelerátorem a zbytkem systému byla relativně nízká a navíc GPU nebyl projektován pro zpětné čtení informací z depth bufferu, objevily se postupy překonávající tuto překážku [GRLM03]. Navíc, techniky pracující s LDI předpokládají, že objekty jsou vodotěsné; v následujícím algoritmu tomu tak být nemusí. Zásadním vylepšením je odstranění zdlouhavého kopírování depth bufferu. Místo toho jsou použity tzv. *z-fail testy*, které jsou naopak velmi rychlé a dobře podporované. Při předzpracování je každý objekt rozložen na několik částí nazývaných subobjekty (například povrchové trojúhelníky). Tyto subobjekty se postupně renderují a ze z-fail testů se pozná, zda je nutné provést přesnou detekci kolizí. GPU je v tomto případě použito pro rychlé prořezávání ve stromě subobjekt testů — rychle odstraňuje ty větve, které nejsou perspektivní.

Výhodou z-fail algoritmu je schopnost zvládnout úplně libovolný typ povrchového meshe, neexistují žádné předpoklady týkající se vstupního modelu. Nepotřebuje komplexní předzpracování ani složité datové struktury. Obecnou nevýhodou všech image space technik je, že pracují jen v určitém rozlišení, neposkytují přesnou detekci kolizí, ale pouze s jistou aproximační chybou.

## 4.5 Diskuse

Podobně jako v kapitole 3 i zde platí, že celá oblast detekce kolizí je mnohem a mnohem bohatší. Pro kompletní výčet používaných metod v naší práci nezbývá místo, natož pro jejich detailnější popis. Zájemce o povolanější úvod do problematiky odkazujeme na práce [TKZ+04] a [Kim05], které mapují oblast detekce kolizí pro tuhá tělesa i pro deformovatelné objekty, a uvádí množství příkladů jejich nasazení.

Protože deformovatelné modely vnáší do detekce kolizí řadu specifických problémů, vyplatí se uvažovat ty z nich, které vyžadují minimální předzpracování, nespoléhají na komplexní datové struktury a podporují i testy na kolizi objektu sama se sebou. Pro účely této práce se jako bezkonkurenčně nejlepší jeví algoritmus prostorového hashování, poněvadž je přímo navržen pro práci s čtyřstěnovými meshi a elegantně zvládá i detekce kolizí objektu sama se sebou. Navíc se dá výhodně použít při výpočtu penetračních hloubek, o němž budeme blíže hovořit v kapitole 5 a 6.

# Kapitola 5

# Řešení kolizí

Jsou-li dvě pevná tělesa v kontaktu, působí na sebe vzájemně silami, které jsou v každém bodě stejně velké, ale opačného směru. To je Newtonův třetí zákon, zákon akce a reakce. Při simulaci vzájemné interakce mezi dynamicky se pohybujícími tuhými tělesy lze zobrazovat a uvažovat pouze konkrétní stavy v diskrétních krocích, což s sebou přináší řadu problémů. Tělesa se mohou začít během simulace protínat. Takové stavy nejsou fyzikálně obhajitelné a je potřeba je nějakým způsobem řešit.

Fyzikálně korektní řešení je odkrokovat celou simulaci v čase zpět do stavu, kdy se tělesa ještě neprotínala, ale byla pouze v kontaktu. V takovém případě lze silovou odezvu spočítat na základě zákonů zachování hybnosti a momentu hybnosti a Newtonových zákonů. Tento postup je v aplikacích pracujících v reálném čase příliš nákladný, protože vyžaduje opakování testů na kolizi.

V této kapitole se budeme zabývat technikami, které jsou nepřesné a fyzikálně nepodložitelné, ale pracují v reálném čase a dávají rozumné výsledky. Tyto techniky hledají výslednou *kontaktní plochu* (contact surface) a/nebo počítají *silovou odezvu* (collision response). Výpočet kontaktní plochy je proces, při němž je vyřešen problém geometrického prolínání obou objektů. Kolidující části těles jsou posunuty tak, aby se vzájemně neprotínaly. Aby se uživateli jevila simulace realisticky, musí být ještě dopočtena vhodná silová odezva na kolizi. Ta pak dál ovlivňuje pohyb a tvar těles po srážce — je zahrnuta do zvoleného fyzikálního modelu jako zpětná vazba v podobě vnějších sil.

Nejprve se budeme krátce věnovat nejpoužívanější informaci o kolidujících bodech, kterou metody při řešení kolizí používají. Tou je tzv. *penetrační vektor*. Dále stručně popíšeme některé konkrétní způsoby řešení kolizí, shrneme jejich vlastnosti a dosažitelnou kvalitu výsledků a zvolíme nejvhodnější techniku pro naše potřeby.

## 5.1 Penetrační vektor

Hledání kontaktní plochy a výpočet silové odezvy se nejčastěji opírá o charakteristiku bodů, která je známá jako *penetrační vektor*. Máme-li dvě kolidující tělesa $A$ a $B$ a kolidující bod $i \in A$, označme $\mathbf{y}_i$ místo na povrchu tělesa $B$, kterým bod $i$ do tělesa $B$

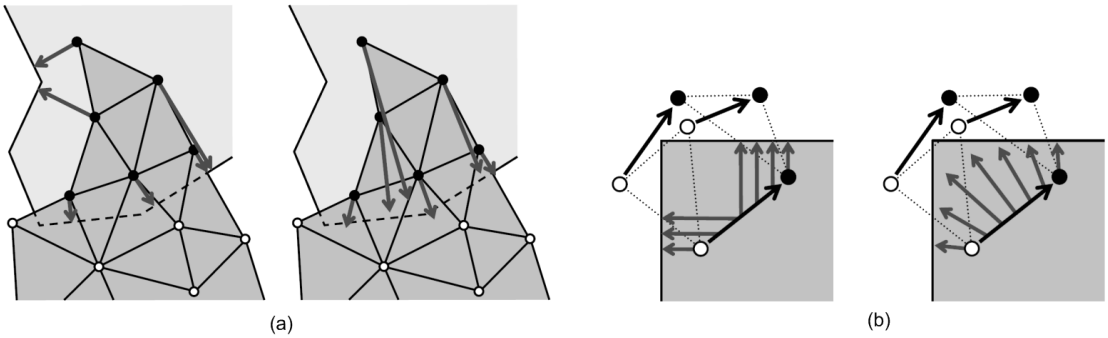
pravděpodobně vniknul. *Penetrační vektor* $\mathbf{d}_i$ *bodu* $i$ je potom

$$\mathbf{d}_i = \mathbf{y}_i - \mathbf{x}_i$$

Velikosti vektoru $\mathbf{d}_i$ se říká *penetrační hloubka*. Otázkou zůstává, jak zjistit, kterým místem $\mathbf{y}$ na povrchu tělesa $B$ daný bod $i$ pronikl.

Postupně se objevilo mnoho algoritmů, které penetrační vektory počítají. Většina z nich dává rozumné výsledky, pokud jsou použity pro hustě vzorkované povrchy a v případě malých penetrací. Naneštěstí je v interaktivních simulacích těžké splnit byť jednu z těchto podmínek.

Mnohé algoritmy vycházejí z nesprávného předpokladu, že penetrační hloubka je rovna nejkratší vzdálenosti kolidujícího bodu k povrchu. Díky tomu se objevují nekonzistence v odhadech, jak ukazuje obrázek 5.1 (a). Dalším často opomíjeným hlediskem je konzistence odhadu penetračního vektoru v čase. Tělesa se ve scénách dynamicky pohybují a u většiny algoritmů díky tomu dochází k nespojitostem v odhadech směrů penetračních vektorů. Ty mohou vést k nežádoucím artefaktům. Problém dobře vystihuje obrázek 5.1 (b).




Obrázek 5.1: Problémy s odhadem penetračních vektorů, které vedou na nežádoucí artefakty [HTK+04]. (a) Vlevo jsou červeně zobrazeny nejkratší vzdálenosti k povrchu. Vpravo jsou věrohodné konzistentní odhady penetračních hloubek. (b) Posouvá-li se bílý trojúhelník do černé koncové polohy, dojde v jednom okamžiku k nespojité změně penetračního vektoru.

Vhodná metoda aproximuje penetrační vektory v tzv. *hraničních bodech*, což jsou kolidující body, které sousedí alespoň s jedním nekolidujícím bodem, a propaguje tyto hodnoty do zbylých kolidujících bodů tak, aby byly v celém kolidujícím objemu konzistentní. Detailně se tomuto tématu věnujeme v kapitole 6.

## 5.2 Výpočet silové odezvy

Fyzikálně nejvěrnější simulace zahrnují jak výpočet kontaktní plochy tak silové odezvy. Techniky pracující v reálném čase v rámci urychlení výpočtů nehledají kontaktní plochu a místo toho se soustředí pouze na silovou odezvu, kterou se záměrně

snaží nastavit tak, aby v dalším kroku integrace bylo zaručeno, že se tělesa přestanou prolínat. Pokud je však kolizní odezva nastavena příliš malá, může dojít k tomu, že tělesa nebudou plně separována. Naopak v případě, že je odezva příliš velká, vynutí poskakování jednoho tělesa po druhém, což působí rušivě v případě, kdy by tělesa měla na sobě pouze v klidu spočívat (resting contacts). Tento problém je velmi dobře patrný obzvlášť tehdy, má-li na sobě v klidu spočívat hned několik těles (object stacking).

### 5.2.1 Penalizační metody

Penalizační metody představují nejstarší způsob generování silové odezvy pro kolidující objekty [MW88]. Jsou oblíbeny zejména díky svojí jednoduchosti a snadné rozšiřitelnosti o další charakteristiky povrchu jako je tření nebo hystereze. Nejčastěji se používá odezva přímo úměrná penetrační hloubce. Pokud se dva objekty protnou, vloží se mezi ně dočasně sada myšlených pružin. Každá pružinka spojuje kolidující vrchol $i$ s pravděpodobně protnutým místem $\mathbf{y}_i$ na povrchu druhého tělesa a působí proti tomu, aby se objekty dál protínaly.

První problém při penalizaci je numerická stabilita, protože velmi tuhé pružiny, které mohou být zapotřebí v případě hlubokých penetrací, vyžadují při integraci krátké časové kroky. Zásadnější problém je vlastní tuhost pružinky — ta musí být dostatečně velká, aby dokázala úplně vyřešit oddělení těles, aniž by zbytečně *přestřelovala*, tj. separovala tělesa mnohem víc, než je nutné. Právě z toho důvodu se používá tato technika iterativně, tzn. pro úspěšné vyřešení kolize se v kolidujícím regionu několikrát zopakuje.

Popsaná technika je založena na penalizaci *point-in-volume*, to znamená, že silová odezva je spočtena pouze pro kolidující body. Ta ale ne vždy generuje plně validní odezvu — totiž pokud se čtyřstěny protínají, aniž by kolidoval některý z jejich vrcholů. Pro uspokojivé vyřešení se zavádí rozšíření zvané penalizace *segment-in-volume*, které je výpočetně náročnější, protože vyžaduje testy na průniky trojúhelníků [MAC04].

### 5.2.2 Neiterativní výpočet kontaktních sil

Protože penalizační metody založené na proměnných parametrech jsou špatně nastavitelné, neflexibilní a často je zapotřebí je iterovat, vzniká potřeba najít metodu, která jednak zaručí, že kolize bude vyřešena ihned, a jednak nebude využívat žádné zbytečné a nesrozumitelné parametry.

Za určitých zjednodušujících předpokladů lze zkonstruovat soustavy rovnic, které svazují nové, opravené polohy a původní polohy bodů a které na sobě navzájem nezávisí a díky tomu je možné je vyřešit analyticky [SBT07]. Cílem je spočíst pravděpodobnou polohu kolidujícího bodu $i$ na kontaktním povrchu a aplikovat na něj *kontaktní sílu* $\mathbf{f}_i^c$, která jej okamžitě akceleruje do této pozice. Tím bude zaručeno, že se v žádném časovém kroku simulace nebudou žádná dvě tělesa protínat.

Metoda vychází z předpokladu, že povrch objektů je tvořen trojúhelníky. Pro výpočet kontaktních sil $\mathbf{f}_i^c$ použijeme následující úvahu. *Kolize* $\ell$ je dvojice $(i, T_i)$, kde $i$ je kolidující vrchol a $T_i$ je jeho *kontaktní trojúhelník*. Kontaktní trojúhelník $T_i$ bodu $i$ je ten trojúhelník na povrchu penetrovaného tělesa, který byl při kolizi pravděpodobně protnut vrcholem $i$. Pro jednoduchost předpokládejme, že všechny vrcholy $\{j, k, l\}$ trojúhelníku $T_i$ patří do množiny kolidujících vrcholů $C$. Každá kolize indukuje vznik lokálních sil — síly $\mathbf{f}_i^\ell$ působící na bod $i$ a sil $\mathbf{f}_{T_i}^\ell$ působících na vrcholy trojúhelníku $T_i$. Na každý bod $i$ působí celková kontaktní síla $\mathbf{f}_i^c$, která je součtem lokálních sil ze všech kolizí, jichž se bod $i$ účastní (tedy buď $\mathbf{f}_i^\ell$ nebo $\mathbf{f}_T^\ell$, kde $i \in T$).

Lokální síly $\mathbf{f}_i^\ell$ a $\mathbf{f}_{T_i}^\ell$, které vznikají během kolize $\ell = (i, T_i)$, jsou odvozeny za předpokladu, že prostor řešení je pro tuto kolizi vymezen penetračním vektorem $\mathbf{d}_i$ a má tedy jen jednu dimenzi. Výpočet bezkolizní polohy $\tilde{\mathbf{x}}_i(t+h)$ bodu $i$ se redukuje na nalezení vhodného skaláru $\alpha_i \in [0, 1]$. Bezkolizní polohu pak spočteme

$$\tilde{\mathbf{x}}_i(t+h) = \mathbf{x}_i(t+h) + \alpha_i \mathbf{d}_i$$

Síla, kterou je potřeba aplikovat na bod $i$ s polohou $\mathbf{x}_i(t)$, aby se dostal na vypočtenou bezkolizní polohu $\tilde{\mathbf{x}}_i(t+h)$, závisí na vybraném integračním schématu. Kupříkladu pro Verletovo schéma je to

$$\mathbf{f}_i^c = \frac{m_i}{h^2} \alpha_i \mathbf{d}_i$$

Tento vztah nezávisí na zvoleném fyzikálním modelu.

## 5.3 Výpočet kontaktní plochy

Kontaktní plochu i silovou odezvu je zapotřebí odvozovat společně. Nejsou-li obě techniky pečlivě skloubeny, může být negativně ovlivněna stabilita fyzikálního systému, protože dojde k porušení zákonů dynamiky.

Vypočítat korektní změny tvaru povrchu je náročné a obecně má velmi mnoho stupňů volnosti. Nejsložitější metody vychází z teorie, jež pro výsledné povrchy formuluje soustavy *omezujících podmínek* (constraints) a vede na optimalizační problémy, které jsou typicky NP těžké. Tyto metody jsou označovány jako metody založené na omezujících podmínkách (constraint methods).

### 5.3.1 Technika porovnávání tlaků

Popíšeme metodu, která porovnává vzájemné tlaky mezi dvěma tělesy $A$ a $B$. Jejím cílem je iterativně hledat bezkolizní polohy vrcholů. Celý postup lze provést v reálném čase a navíc je vhodný i pro řídce vzorkované objekty [ST05].

Nechť $C$ je množina kolidujících bodů. Pro každý kolidující bod $i$ je spočten penetrační vektor $\mathbf{d}_i$ a *kontaktní trojúhelník* $T_i$, stejně jako v sekci 5.2.2. Ke kolidujícím bodům se přidají i vrcholy, které patří kontaktním trojúhelníkům, čímž vznikne *deformační region* $D$.

Pro každý uzel $i \in D$ spočteme vektor posunutí $\mathbf{s}_i$, který bude použit pro vyřešení kolize. Tento vektor spočteme jako vážený průměr penetračních vektorů $\mathbf{d}_j$ vrcholů $j$ druhého objektu, jejichž kontaktní trojúhelníky obsahují uzel $i$. Formálně jde o množinu $J_i = \{j \in D | i \in T_j\}$.

Poloha $\mathbf{x}_i$ vrcholu $i$ je iterována podle vzorce

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} \pm \frac{1}{2^{(k+1)}} \mathbf{s}_i$$

Směr postupu záleží na rozdílu tlaků, které jsou v bodě $\mathbf{x}_i$ vykazovány jedním a druhým povrchem. Po každém kroku této iterace totiž dojde v důsledku změn v polohách bodů deformačního regionu ke změně rozložení vnitřního silového působení uvnitř tělesa. Vnitřní síly je potřeba vyhodnotit a přepočítat na tlaky, které působí v okolí bodu $i$. V závislosti na tom, které z těles $A$ a $B$ vyvíjí větší tlak, se bod $i$ posune ve směru nebo proti směru vektoru $\mathbf{s}_i$. Cílem je dosáhnout rovnováhy ve vzájemném silovém působení obou těles.

Protože popsané schéma konverguje velmi rychle, v praxi postačují 3-4 iterace. Po vyhodnocení výsledného tvaru povrchu jsou upraveny rychlosti kolidujících vrcholů a dopočteny případné další síly, například tření.

## 5.4 Diskuse

Penalizační metoda je nejstarší technika, která je sice jednoduchá, ale špatně nastavitelná. Dlouhou dobu dominovala na poli simulace deformovatelných modelů v reálném čase, zejména díky svojí nenáročnosti na implementaci. Metody, které poskytují mnohem kvalitnější výsledky byly představeny postupně teprve nedávno. Z nich vybíráme dvě nejzajímavější. Algoritmus využívající neiterativní výpočet kontaktních sil a technika porovnávání tlaků slibují stejně dobré výsledky, protože používají podobné prostředky a oběma jim jde v podstatě o nalezení kontaktní plochy. Liší se pouze způsobem, kterým body na kontaktní plochu dopraví. Porovnávání tlaků sice vyžaduje (malý) počet opakování, ale má intuitivnější koncept a je přímo navrženo pro práci s různě hustě vzorkovanými objekty, a proto se přikloníme k tomuto řešení. Celou techniku spolu s algoritmem pro odhad penetračních vektorů podrobně rozebereme v kapitole 6.

# Kapitola 6

# Navrhované řešení

V předchozích kapitolách jsme pokryli všechna témata, kterými je třeba se při simulaci deformovatelných modelů zabývat. V této kapitole představíme celkové schéma, které kombinuje zmiňované techniky. Konkrétní techniky pro toto schéma byly vybrány tak, aby zaručovaly

1. schopnost pracovat v reálném čase s rozumně komplexními objekty
2. snadnou konfigurovatelnost pomocí jednoduchých parametrů
3. věrohodné výsledky

Detailnímu popisu jednotlivých částí, z nichž se celkové schéma skládá, se budeme věnovat v druhé části této kapitoly. Zhodnocení dosažených výsledků lze najít v kapitole 7.

## 6.1 Celkové schéma

Navrhované schéma je postaveno tak, aby pracovalo s objemovými meshi. To přináší řadu nevýhod a omezení především proto, že objemové meshe jsou poměrně obsáhlé. Například krychle složená z $10 \times 10 \times 10$ krychliček, které jsou převedeny na čtyřstěny, má 1331 uzlů a 5000 čtyřstěnů. Velmi snadno tedy dosahujeme maximálního počtu bodů a čtyřstěnů, které lze simulovat v reálném čase. Problém není jenom v deformovatelných modelech, musíme zohlednit i fakt, že použité objekty podstupují ještě řadu dalších, výpočetně náročnějších kroků, jmenovitě detekci kolizí a řešení kolizí.

Zmíněný nedostatek je však vyvážen lepšími a věrohodnějšími výsledky, kterých lze s objemovými meshi dosáhnout. Důvodem je hlavně konzistentní odhad penetračních hloubek, z něhož plyne i větší kvalita odvozeného kontaktního povrchu. Objemové meshe navíc mnohem lépe zachovávají svůj tvar než pouhé povrchové meshe.

Budeme-li chtít simulovat komplexní povrchové meshe pomocí navrženého algoritmu, použijeme techniku známou z oblasti *free-form deformací*. Komplexní povrchový mesh vnoříme do hrubší pravidelné prostorové sítě (lattice), která jej bude

aproximovat. Celou simulaci pak budeme provádět s hrubou sítí, kterou lze snadno rozdělit na pravidelné čtyřstěny, a deformace těchto čtyřstěnů zpětně promítneme na jemný povrchový model. Pokud se ukáže, že je vizualizace zdlouhavá, můžeme ji v některých krocích vynechat a jemný model zobrazovat třeba jenom každý druhý krok simulace.

Ze všech deformací se omezíme pouze na elastické. Budeme předpokládat, že během simulace nemůže dojít k porušení souvislosti modelu. Spolu s vnitřními silami vznikajícími v tělesech je potřeba uvážit ještě vnější síly, které na těleso mohou působit. V našem řešení půjde především o gravitační sílu, která je v simulovaném prostředí přítomna neustále, a liniovou sílu, kterou může uživatel působit na těleso ve vybraném bodě. Další vnější síla, kterou budeme uvažovat, je třecí síla, která vzniká v oblasti, kde se dvě tělesa navzájem stýkají.

Kostra navrženého algoritmu má tři základní body

1. simulace deformovatelného modelu
   - výpočet sil působících v každém bodě
   - aktualizace poloh bodů v čase
2. detekce kolizí
   - nalezení seznamu kolidujících bodů
   - a seznamu kolidujících (povrchových) trojúhelníků
3. řešení kolizí
   - výpočet penetračních hloubek a bezkolizních poloh
   - aktualizace poloh kolidujících bodů

Kostra je dostatečně obecná, aby nás neomezovala v použití různých technik v jednotlivých částech.

## 6.2 Deformovatelný model

Všechny tři deformovatelné modely popsané v kapitole 3 jsou z hlediska vytýčených kriterií stejně vhodnými kandidáty. Všechny jsou schopny pracovat v interaktivních rychlostech s objemovými meshi a podporují jak elastické, tak plastické deformace a chování lze ovládat prostřednictvím malého počtu parametrů.

Jako nejvhodnější kompromis mezi rychlostí, intuitivitou parametrů a věrohodností vybereme zobecněný Mass-Spring System.

### 6.2.1 Detaily modelu

Deformovatelný model jsme již poměrně podrobně představili v kapitole 3. Zde se budeme věnovat jeho konkrétnímu nasazení a odvodíme finální podoby vzorců.

Zobecněný Mass-Spring System, který má pracovat s objemovými meshi, bude zahrnovat dva různé typy vnitřních sil. Jedním budou tlumené síly vycházející z podmínky zachování délek hran a druhým typem budou síly zaručující zachování objemu čtyřstěnů. Nebudeme se zde zabývat silami bránícími změně obsahu povrchových trojúhelníků, které by v tomto případě měly pouze zanedbatelný vliv.

**Zachování vzdálenosti.** Síly, které zaručují zachování vzdálenosti, se musí vypočítat pro každou hranu a distribuovat jejím koncovým bodům. Tlumení těchto sil je zásadní pro stabilitu celé simulace.

Mějme hranu $e = (i, j)$, jejíž klidová délka je $l_0$. Tlumená síla, která působí proti změně délky $e$ v bodě $i$, je odvozena z podmínky (3.9) za použití vztahu (3.8) a má tvar

$$\mathbf{f}_i^e(\mathbf{x}_i, \mathbf{x}_j) = \left( -k_D \cdot \frac{|\vec{\mathbf{x}}_{ij}| - l_0}{l_0} + d_C \cdot \frac{\vec{\mathbf{x}}_{ji} \cdot \mathbf{v}_i + \vec{\mathbf{x}}_{ij} \cdot \mathbf{v}_j}{|\vec{\mathbf{x}}_{ij}| \cdot l_0} \right) \cdot \frac{\vec{\mathbf{x}}_{ji}}{|\vec{\mathbf{x}}_{ij}| \cdot l_0} \tag{6.1}$$

$\vec{\mathbf{x}}_{ij}$ označuje vektor $\mathbf{x}_j - \mathbf{x}_i$; analogické značení používáme i v rovnicích níže. Parametr $k_D$ upravuje význam této síly; čím je větší, tím větší je i síla působící proti změně. Parametr $d_C$ je činitel útlumu a v závislosti na rychlostech koncových bodů provádí útlum.

**Zachování objemu.** Síly, které brání změně objemu, se vypočítají z (3.11), ale nepoužívá se u nich tlumení, tedy jsou odvozeny podle (3.7).

Mějme čtyřstěn $T = (i, j, k, l)$ s klidovým objemem $V_0$. Síla, která vlivem jeho deformace působí na vrchol $m$, je

$$\mathbf{f}_m^T(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) = -k_V \cdot \left( \frac{\frac{1}{6}(\vec{\mathbf{x}}_{ij} \cdot (\vec{\mathbf{x}}_{ik} \times \vec{\mathbf{x}}_{il})) - V_0}{V_0} \right) \cdot \frac{\partial C}{\partial \mathbf{x}_m} \tag{6.2}$$

$k_V$ je koeficient vyjadřující význam této síly. Poslední člen rovnice (6.2) je pro $i$ roven

$$\frac{\partial C}{\partial \mathbf{x}_i} = \vec{\mathbf{x}}_{il} \times \vec{\mathbf{x}}_{ik} + \vec{\mathbf{x}}_{ik} \times \vec{\mathbf{x}}_{ij} + \vec{\mathbf{x}}_{ij} \times \vec{\mathbf{x}}_{il}$$

a pro $j$, $k$ a $l$

$$\frac{\partial C}{\partial \mathbf{x}_j} = \vec{\mathbf{x}}_{ik} \times \vec{\mathbf{x}}_{il}, \qquad \frac{\partial C}{\partial \mathbf{x}_k} = \vec{\mathbf{x}}_{il} \times \vec{\mathbf{x}}_{ij}, \qquad \frac{\partial C}{\partial \mathbf{x}_l} = \vec{\mathbf{x}}_{ij} \times \vec{\mathbf{x}}_{ik}$$

**Tření.** Celkový dojem ze simulace lze velmi vylepšit, použijeme-li alespoň zjednodušený model dynamického tření. Třecí síly $\mathbf{f}_i^r$ vznikají ve stykových plochách mezi dvěma tělesy; my je budeme počítat v kolidujících povrchových bodech. Tyto síly brání pohybu objektů v kontaktu, jsou přímo úměrné celkové síle, která v uvažovaném bodě působí, a působí proti směru relativní rychlosti.

Odvodit kvalitní model tření není triviální. Mnoho simulací implementuje tzv. Coulombův model (Coulomb friction) [MC95]. V naší zjednodušené implementaci nebudeme uvažovat relativní rychlost bodu $i$ vůči tělesu, s nímž je v kontaktu, ale pouze jeho vlastní rychlost $\mathbf{v}_i$:

$$\mathbf{f}_i^r = -\mu \cdot |\mathbf{f}_i| \cdot \frac{\mathbf{v}_i}{|\mathbf{v}_i|} \tag{6.3}$$

$\mathbf{f}_i$ je celková síla působící v bodě $i$ a $\mu$ je koeficient upravující velikost výsledné třecí síly.

**Vztah parametrů.** Největší význam pro celou simulaci má parametr $k_D$. Je-li velký, těleso se chová, jako by bylo z tuhého materiálu. Při menších hodnotách chování tělesa připomíná rosolovitou látku.

Oproti tomu parametr $k_V$ má význam spíš pro zachování orientace čtyřstěnů. Pokud je hodně velký, zatímco $k_D$ je malý, těleso vypadá, jako by se rozpouštělo.

Dalším nastavitelným parametrem je činitel útlumu $d_C$. Na jeho nastavení je třeba dát pozor, protože kdyby byl příliš velký, docházelo by k nežádoucímu jevu — k přidávání energie do systému, což by okamžitě vyústilo ve ztrátu stability.

Koeficient tření ovlivňuje, jak moc budou objekty po sobě sklouzávat. Malé koeficienty vytvářejí dojem hladkých kluzkých povrchů, jako je například led, zatímco použití větších koeficientů připomíná hrubé materiály.

**Význam pravidelnosti meshe.** Nezanedbatelný vliv na kvalitu simulace má pravidelnost objemového meshe. Je známo, že adaptivní metody, které jsou schopny pracovat s různě hustě vzorkovanými meshi, provádějí složitou konsolidaci atributů (polohy, hmotnosti, rychlosti) mezi jemnými a hrubými oblastmi [DDCB01]. Různé velikosti čtyřstěnů a příliš velký rozptyl v délkách hran mohou negativně ovlivnit stabilitu celého systému.

### 6.2.2 Algoritmus

Postup je schematicky popsán v algoritmu 0.1. Celková síla $\mathbf{f}_i$, která působí na bod $i$, se během každého kroku počítá znovu. Pro aktualizaci poloh v čase je použito Verletovo schéma. Snadno nahlédneme, že složitost algoritmu je lineární vzhledem k počtu vrcholů, hran a čtyřstěnů.

## 6.3 Detektor kolizí

Pro detekci kolizí využijeme optimalizovaného prostorového hashování, které jsme vybrali v kapitole 4. Algoritmus byl publikován teprve nedávno, ale výsledky měření ukazují, že je schopen pracovat v reálném čase až s $10^4$ čtyřstěny.

**Algoritmus 0.1** Výpočet sil a integrace poloh bodů v čase.

**AktualizujPolohy**(časový krok $h$)

1. pro každou hranu $e = (i, j)$
   - pro $m = i, j$ spočti $\mathbf{f}_m^e$ podle (6.1)
2. pro každý čtyřstěn $T = (i, j, k, l)$
   - pro $m = i, j, k, l$ spočti $\mathbf{f}_m^T$ podle (6.2)
3. pro každý bod $i$
   - spočti liniovou sílu $\mathbf{f}_i^u$, kterou působí uživatel
   - $\mathbf{f}_i = \sum_e \mathbf{f}_i^e + \sum_T \mathbf{f}_i^T + m_i \cdot \mathbf{g} + \mathbf{f}_i^u$; $\mathbf{g}$ je gravitační zrychlení
   - pokud je $i$ kolidující povrchový bod, spočti $\mathbf{f}_i^r$ podle (6.3)
   - $\mathbf{x}_i(t+h) = 2 \cdot \mathbf{x}_i(t) - \mathbf{x}_i(t-h) + \frac{h^2}{m_i}(\mathbf{f}_i + \mathbf{f}_i^r)$
   - $\mathbf{v}_i(t+h) = \frac{1}{2h}\left(\mathbf{x}_i(t+h) - \mathbf{x}_i(t-h)\right)$

### 6.3.1 Mřížka

Algoritmus využívá myšlenou pravidelnou mřížku, která prostor dělí na množinu disjunktních buněk. Ty slouží ke klasifikaci primitiv objektů nacházejících se v uvažovaném prostoru.

Prostorová mřížka je nastavena jediným parametrem, kterým je velikost buňky $l$. Tento parametr v podstatě ovlivňuje počet primitiv v buňce. Jsou-li buňky příliš malé, budou sice pravděpodobně obsahovat méně uložených bodů, ale čtyřstěny jich budou protínat více. Jsou-li naopak buňky příliš velké, obsahují velké množství bodů, s nimiž pak čtyřstěny musí být testovány. Experimenty a měření ukazují, že optimální velikost buňky by měla být přibližně rovna průměrné délce hran čtyřstěnů uvažovaných objektů.

### 6.3.2 Kolidující body

Samotná detekce kolizí probíhá ve dvou fázích. V první fázi jsou zpracovány všechny body všech objektů. Pro každý bod $\mathbf{x} = (x, y, z)$ se zjistí, do které buňky padl. Index této buňky je použit jako klíč do hashovací tabulky. Pro bod $\mathbf{x}$ je klíčem trojice $(\lfloor x/l \rfloor, \lfloor y/l \rfloor, \lfloor z/l \rfloor)$, $l$ je délka hrany buňky.

Hashovací funkce, která převádí klíče na indexy do tabulky velikosti $n$, by měla být především rychle a snadno vyčíslitelná. V naší implementaci použijeme jednoduchou funkci

$$h(\mathbf{x}) = h(x, y, z) = (x \cdot p_1 \,\mathbf{xor}\, y \cdot p_2 \,\mathbf{xor}\, z \cdot p_3) \,\mathbf{mod}\, n$$

$p_1$, $p_2$ a $p_3$ jsou vybraná velká konstantní prvočísla, konkrétně 73856093, 19349663 a 83492791.

Ve druhé fázi jsou zpracovány všechny čtyřstěny objektů. Pro každý čtyřstěn $T$ se nejprve zkonstruuje jeho obalový kvádr, který má stěny rovnoběžné se souřadnými osami. Zjistí se, které buňky dané myšlené mřížky obalový kvádr protíná, a testuje se, zda body obsažené v těchto buňkách nejsou uvnitř čtyřstěnu $T$. Pokud ano, jde o kolidující body.

Test, zda se bod $\mathbf{x}$ nachází uvnitř $T$, lze urychlit tak, že $\mathbf{x}$ nejprve testujeme proti obalovému kvádru. Pro vlastní test bod–čtyřstěn autoři metody navrhují vypočítat barycentrické souřadnice $\mathbf{x}$ vzhledem k $T$. Výhodou tohoto testu je, že nezávisí na orientaci čtyřstěnu.

### 6.3.3 Hashovací tabulka

Vkládání záznamů do hashovací tabulky realizujeme technikou *hashování s přesuny*. Do tabulky jsou ukládány *řetězce* kolidujících prvků. Řetězec je vždy tvořen množinou bodů, které padnou do téže buňky myšlené mřížky. Provázání v řetězci zajišťuje ukazatel *vzad* na předchozí a ukazatel *vpřed* na následující prvek. Začátek, resp. konec řetězce má prázdný ukazatel vzad, resp. vpřed.

Pokud je při vkládání nového prvku $\mathbf{x}$ pozice $h(\mathbf{x})$ volná, znamená to, že buňka zatím žádné body neobsahuje. Proto prvek jednoduše vložíme. Dojde-li však ke kolizi, musíme zjistit, zda řetězec, do kterého $\mathbf{x}$ patří, opravdu na pozici $h(\mathbf{x})$ začíná. Jinými slovy musíme ověřit, zda $\mathbf{x}$ do tohoto řetězce patří. Pokud ano, vložíme $\mathbf{x}$ na konec tohoto řetězce. Pokud ne, pak prvek, který kolidující pozici okupuje, přesuneme na jiné volné místo, aktualizujeme příslušné ukazatele vpřed a vzad v jeho řetězci, a na nyní volnou polohu $h(\mathbf{x})$ uložíme $\mathbf{x}$.

Na rychlost vkládání má vliv velikost tabulky $n$. Čím je větší, tím menší je riziko, že dojde ke kolizi. Z toho důvodu zajistíme, aby tabulka byla zaplněna maximálně na 75%. Kvůli zajištění kvality hashovací funkce se navíc doporučuje, aby $n$ bylo prvočíslo. V naší implementaci tedy bude $n$ první prvočíslo, které je větší než $4/3$ počtu bodů ukládaných do tabulky.

### 6.3.4 Kolidující trojúhelníky

Algoritmus pro výpočet penetračních hloubek pro svou práci potřebuje znát i seznam *kolidujících trojúhelníků*. To jsou ty trojúhelníky na povrchu tělesa, kterými mohly kolidující body proniknout do tělesa. Mezi ně patří ty trojúhelníky, které mají alespoň jeden vrchol kolidující, ale také ty, které patří zasaženým čtyřstěnům. Prakticky je lze dohledat velmi snadno. Při detekci kolizí označíme nejenom kolidující body, ale navíc všem vrcholům kolidujícího čtyřstěnu nastavíme speciální příznak, který znamená, že se účastní nějaké kolize. Na konci projdeme všechny povrchové trojúhelníky a sesbíráme všechny, které mají alespoň jeden kolidující nebo speciální vrchol. To lze provést v lineárním čase vzhledem k počtu všech povrchových trojúhelníků scény.

### 6.3.5 Algoritmus

Princip prostorového hashování určeného k nalezení seznamu kolidujících bodů shrnuje algoritmus 0.2. Zpracování všech bodů v první fázi probíhá v lineárním čase vzhledem k počtu bodů; vkládání prvků do hash tabulky má očekávaný konstantní čas. Druhá fáze je lineární vzhledem k počtu $n$ čtyřstěnů ve scéně. Jestliže $p$ značí průměrný počet buněk mřížky, které protne jeden čtyřstěn, a $q$ průměrný počet bodů v buňce, pak časová složitost je řádu $O(q \cdot p \cdot n)$. Protože velikost mřížky byla volena relativně vzhledem k průměrné délce hrany čtyřstěnu, je $p$ konstantní.

---

**Algoritmus 0.2** Algoritmus prostorového hašování.

---

**DetekujKolize**(objekty)

1. pro každý bod $i$ každého objektu
   - najdi klíč $(x, y, z)$ a vlož $i$ podle něj do hash tabulky
2. pro každý čtyřstěn $T$ každého objektu
   - pro každou buňku $C$ dané myšlené mřížky, kterou čtyřstěn $T$ protíná
     - pro každý bod $i$ buňky $C$
       - zjisti, zda $i$ neleží uvnitř $T$
       - pokud ano, vlož $i$ do seznamu kolidujících vrcholů
       - vrcholům $T$ nastav speciální příznak, že se účastní kolize
3. vrať seznam kolidujících vrcholů

---

## 6.4 Řešení kolizí

Řešení kolizí založíme na technice porovnávání tlaků, pro niž jsme se rozhodli v kapitole 5. Nejprve je zapotřebí spočítat penetrační vektory a poté dohledat kontaktní trojúhelníky kolidujících bodů; tyto dvě informace využijeme k výpočtu vektorů posunutí, které slouží k nalezení kontaktního povrchu.

### 6.4.1 Penetrační vektory

Penetrační vektory budeme počítat podle schématu posaného v algoritmu 0.3. Pro lepší představu jednotlivé kroky zachycuje i obrázek 6.1. Do algoritmu zbývá doplnit způsob, jak efektivně najít průsečíky hran, které spojují kolidující a nekolidující body a které budeme označovat jako *průnikové hrany* (intersection edges), jak v těchto průsečících interpolovat normálu penetrovaného povrchu a jak z těchto informací aproximovat penetrační vektory hraničních bodů. Dále je potřeba vyřešit konkrétní odvození penetračních vektorů zbylých kolidujících bodů v propagační fázi.

Časová složitost tohoto algoritmu závisí na rychlosti kroku 2. Procedura, kterou používáme a kterou popíšeme o kousek níže, v nejhorším případě pracuje v čase $O(m \cdot t)$, kde $m$ je počet průnikových hran a $t$ je počet kolidujících trojúhelníků. Ve většině případů však pracuje s lineární složitostí $O(m)$. Propagační fázi lze provést v lineárním čase vzhledem k počtu kolidujících vrcholů.

---

**Algoritmus 0.3** Odhad penetračních vektorů.

---

**PenetračníVektory**(množina $C$ kolidujících bodů, množina $D$ nekolidujících bodů)

1. najdi množinu $B$ *hraničních bodů* = kolidujících bodů, které sousedí alespoň s jedním nekolidujícím bodem
2. pro každou *průnikovou hranu* $e$ spočti její průsečík $\mathbf{x}_e$ s povrchem protnutého tělesa a jednotkovou povrchovou normálu $\mathbf{n}_e$ v tomto bodě
3. aproximuj penetrační vektory $\mathbf{d}_i$ hraničních bodů $i$ s použitím přilehlých průsečíků $\mathbf{x}_e$ a normál $\mathbf{n}_e$.
4. *propagační fáze*
   - všechny hraniční body z $B$ označ jako *zpracované*
   - najdi množinu $B$ nových hraničních bodů = všech nezpracovaných kolidujících bodů, které sousedí s alespoň jedním zpracovaným bodem
   - je-li $B$ prázdná, algoritmus končí
   - pro každý nový hraniční bod $i$ vypočti $\mathbf{d}_i$ s použitím hodnot ze všech zpracovaných sousedních bodů $j$
   - opakuj propagační fázi

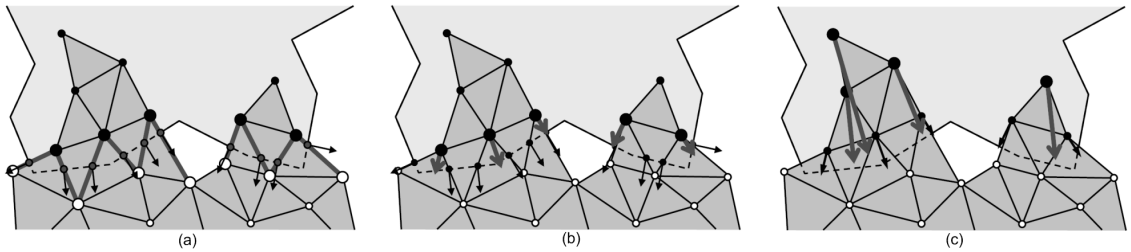
---

**Průnikové hrany.** Ve scénách, kde se nachází více než dvě tělesa, se může stát, že oba konce nějaké hrany $e = (a, b)$ proniknou do různých těles. Taková hrana je samosebou stále chápána jako průniková, přestože spojuje dva kolidující body. Hranu $e$ v takovém případě chápeme jako dvojici hran $e_a = (a, b)$, kde $a$ koliduje a $b$ nikoli, a $e_b = (a, b)$, kde $b$ koliduje a $a$ nikoli.

Opatrní musíme být pak i při propagaci penetračních hloubek, aby kolidující body přispívaly pouze těm sousedům, kteří pronikli do stejného objektu.

**Nalezení průsečíků.** Jak již bylo předesláno na konci kapitoly 4, algoritmus prostorového hashování lze použít pro nalezení průsečíků průnikových hran s povrchem penetrovaného tělesa.

Místo bodů jsou do tabulky vkládány průnikové hrany. Ty mohou protínat více buněk současně, proto je potřeba je vkládat podle klíče každé zasažené buňky. K tomu použijeme *DDA* algoritmus (Digital Differential Analyzer) pro průchod mřížkou. Každá buňka tak má v hash tabulce asociován řetězec hran, které do ní zasahují.




Obrázek 6.1: Výpočet penetračních vektorů. Převzato z [HTK+04]. (a) Nalezení průnikových hran, jejich průsečíků s povrchem a normál v těchto bodech. (b) Aproximace penetračních vektorů v hraničních bodech. (c) Propagace penetračních vektorů do zbylých kolidujících bodů.

V druhé fázi prostorového hashování pracujeme s kolidujícími trojúhelníky $T$. Ty, podobně jako předtím čtyřstěny, postupně testujeme se všemi hranami obsaženými ve všech buňkách, které $T$ protne.

V praxi se při měřeních ukázalo, že taková technika je zbytečně složitá, protože vkládání hran a zjišťování, které buňky trojúhelník protne, zabere dost času. Lepší výsledky dostaneme, pokud využijeme časové koherence a budeme si pamatovat, který trojúhelník hrana naposledy protnula. V případě, že je informace neaktuální, nový trojúhelník najdeme lineárním průchodem přes všechny *kolidující trojúhelníky* toho tělesa, do kterého hrana zasahuje. Toto těleso snadno poznáme, pokud si během detekce kolizí u všech kolidujících bodů poznamenáme, do kterých těles padly.

**Aproximace penetračních vektorů.** Známe-li všechny průsečíky $\mathbf{x}_e$ hran $e$ s povrchem penetrovaného tělesa i jednotkové normály $\mathbf{n}_e$ v těchto bodech, můžeme v hraničních bodech $i$ aproximovat penetrační hloubku $h_i$ a jednotkový penetrační směr $\mathbf{r}_i$ pomocí vážených průměrů. Použitá váhová funkce má tvar

$$w_i(\mathbf{x}) = \frac{1}{|\mathbf{x} - \mathbf{x}_i|^2} \tag{6.4}$$

Tato funkce výrazně preferuje body, které jsou si hodně blízko.

Součtem přes všechny průnikové hrany $e$ incidentní s hraničním bodem $i$ dostaneme hledané aproximace

$$\begin{aligned} h_i &= \frac{\sum_e w_i(\mathbf{x}_e) \cdot (\mathbf{x}_e - \mathbf{x}_i) \cdot \mathbf{n}_e}{\sum_e w_i(\mathbf{x}_e)} \\ \tilde{\mathbf{r}}_i &= \frac{\sum_e w_i(\mathbf{x}_e) \cdot \mathbf{n}_e}{\sum_e w_i(\mathbf{x}_e)} \\ \mathbf{r}_i &= \frac{\tilde{\mathbf{r}}_i}{|\tilde{\mathbf{r}}_i|} \end{aligned}$$

a kýžený odhad vlastního penetračního vektoru $\mathbf{d}_i$ bodu $i$ je

$$\mathbf{d}_i = h_i \cdot \mathbf{r}_i$$

**Propagace penetračních vektorů.** Ve fázi propagace se penetrační hloubky a směry zpracovaných bodů $j$ použijí k odvození hloubek a směrů nových hraničních bodů $i$, se kterými sousedí, s použitím následujících vztahů

$$\begin{aligned} h_i &= \frac{\sum_j w_i(\mathbf{x}_j) \cdot ((\mathbf{x}_j - \mathbf{x}_i) \cdot \mathbf{r}_j + h_j)}{\sum_j w_i(\mathbf{x}_j)} \\ \tilde{\mathbf{r}}_i &= \frac{\sum_j w_i(\mathbf{x}_j) \cdot \mathbf{r}_j}{\sum_j w_i(\mathbf{x}_j)}, \quad \mathbf{r}_i = \frac{\tilde{\mathbf{r}}_i}{|\tilde{\mathbf{r}}_i|} \end{aligned}$$

### 6.4.2 Kontaktní trojúhelníky

Pro identifikaci kontaktních trojúhelníků $T_i$ kolidujících bodů $i$ můžeme využít podobný přístup, jako při hledání průsečíků průnikových hran s povrchem objektu. Místo průnikových hran použijeme polopřímky, které budou vycházet z bodu $i$ ve směru penetračního vektoru $\mathbf{d}_i$. Protože penetrační vektor míří k povrchu, lze očekávat, že příslušný kontaktní trojúhelník bude protnut pro $t \approx 1$ a z toho důvodu nemusíme do hash tabulky vkládat přímo polopřímky, ale jenom jejich části, například pro $t \in [0, 2]$.

I přesto se může stát, že průsečíků s kolidujícími trojúhelníky bude několik. Pak vybereme ten, který je bodu $i$ nejblíž.

Kandidáty na kontaktní trojúhelníky budeme hledat mezi kolidujícími trojúhelníky. To není plně korektní předpoklad, protože v případě komplexních povrchů se může stát, že penetrační vektor bude protínat některý z nekolidujících povrchových trojúhelníků. Vzhledem k tomu, že se v naší implementaci omezujeme na pravidelné objemové meshe, je tato situace krajně nepravděpodobná. Pokud se přece jen u některého bodu nepodaří najít kontaktní trojúhelník, ponecháme v rámci daného kroku tento vrchol na svém místě.

Při měřeních se opět ukazuje, že rychlejší je si u každého bodu pamatovat poslední kontaktní trojúhelník a v případě, že není aktuální, jednoduše projít seznam všech kolidujících trojúhelníků proniknutého objektu.

### 6.4.3 Výpočet vektorů posunutí

V této části popíšeme, jak najít vektory posunutí $\mathbf{s}_i$ pro každý bod $i$ *deformačního regionu* $D$. Deformační region je množina všech kolidujících bodů spolu s vrcholy jejich kontaktních trojúhelníků. Vrcholy kontaktních trojúhelníků nemusí samy o sobě kolidovat, nicméně jsou zahrnuty do deformačního regionu proto, že jejich posunutí pomůže úspěšně vyřešit kolizi.

Abychom byli schopni zaručit rozumné a v čase spojité vektory posunutí $\mathbf{s}_i$, použijeme při výpočtu interpolaci, která uvažuje nejen samotný penetrační vektor $\mathbf{d}_i$ bodu $i$, ale i vážený průměr penetračních vektorů bodů $j$, jejichž kontaktní trojúhelníky $T_j$ jsou incidentní s $i$. Tedy

$$\mathbf{s}_i = \frac{\sum_{j \in J_i} w_j \cdot (-\mathbf{d}_j) + \mathbf{d}_i}{\sum_{j \in J_i} w_i + 1}, \quad J_i = \{j \in D : i \in T_j\}$$

Pokud vrchol $i$ není kolidující a tudíž nemá penetrační vektor, je při výpočtu $\mathbf{s}_i$ použita hodnota $\mathbf{d}_i = \mathbf{0}$. Váha $w_j$ penetračního vektoru $\mathbf{d}_j$, kterým bod $j$ s kontaktním trojúhelníkem $T_j$ přispívá bodu $i$, se vypočte následovně

$$w_j = \frac{\mathbf{area}(j,k,l)}{\mathbf{area}\, T_j}, \quad T_j = \{i,k,l\}$$

### 6.4.4 Kontaktní povrch

Výsledný kontaktní povrch mezi dvěma tělesy $A$ a $B$ spočteme iterativně s použitím vektorů posunutí. V prvním kroku posuneme všechny body v deformačním regionu přesně do poloviny

$$\mathbf{x}_i^{(1)} = \mathbf{x}_i + \frac{1}{2}\mathbf{s}_i$$

Pokud by obě tělesa byla v daném místě stejně elastická, kolize by byla vyřešena, kontaktní povrch by se nacházel přesně mezi nimi. V každém dalším kroku iterace musíme nejprve porovnat tlaky, které působí v okolí bodu $i$, a podle toho určíme, zda je třeba bod $i$ posunout ve směru nebo proti směru spočteného vektoru $\mathbf{s}_i$.

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} \pm \frac{1}{2^{k+1}}\mathbf{s}_i \tag{6.5}$$

**Tlaky v okolí bodů.** Tím, že v každé iteraci dojde ke změně poloh bodů deformačního regionu, dojde i ke změně rozložení vnitřních sil $\mathbf{f}_i$ působících v bodech $i$ deformovatelných těles. Tyto síly je potřeba znovu vyhodnotit.

Uvažme nyní bod $i$ z deformačního regionu tělesa $A$. Tlak, který působí v bodě $i$, budeme počítat v malé oblasti $O_i$ kolem tohoto bodu, která je tvořena sjednocením oblastí všech povrchových trojúhelníků incidentních s bodem $i$. Z tělesa $A$ se v $O_i$ nachází jenom bod $i$ a proto výpočet tlaku $\mathbf{p}_i^A$, který vzniká deformací tělesa $A$, je snadný

$$\mathbf{p}_i^A = \frac{\mathbf{f}_i}{\mathbf{area}\, O_i}$$

Z tělesa $B$ se v oblasti $O_i$ nachází všechny body $j \in J_i \cap B = \{j \in B \cap D : i \in T_j\}$, $D$ je deformační region. Aby tlak $\mathbf{p}_i^B$ byl spojitý, použijeme podobnou interpolaci jako při výpočtu vektorů posunutí $\mathbf{s}_i$.

$$\mathbf{p}_i^B = \frac{\sum_{j\in J_i} w_j \mathbf{f}_j}{\mathbf{area}\, O_i \cdot \sum_{j\in J_i} w_i}$$

Spočtené tlaky $\mathbf{p}_i^A$ a $\mathbf{p}_i^B$ promítneme do vektoru posunutí $\mathbf{s}_i$ a porovnáme jejich velikost: pokud $(\mathbf{p}_i^A \cdot \mathbf{s}_i < \mathbf{p}_i^B \cdot \mathbf{s}_i)$, postupujeme ve směru $\mathbf{s}_i$ a naopak.

V případě, že povrch tělesa $B$ je mnohem méně hustě vzorkován, může dojít k tomu, že množina $J_i \cap B$ je prázdná, tzn. že žádný kontaktní trojúhelník žádného bodu tělesa $B$ neobsahuje bod $i$. Tehdy se uvažovaná oblast rozšíří a tlak $\mathbf{p}_i^B$ se spočte pomocí vnitřních sil působících ve vrcholech kontaktního trojúhelníku $T_i$.

### 6.4.5 Opakovaná iterace

Iterační schéma zachycené rovnicí (6.5) konverguje velmi rychle a pro praktické použití stačí jen několik iterací. Ukazuje se, že tři iterace poskytují dostatečně dobré výsledky.

## 6.5 Shrnutí

V kapitole jsme vytvořili obecné schéma pro simulaci deformovatelných objektů, které neomezuje v použití konkrétních technik. Na základě informací a závěrů z předešlých kapitol jsem vybrali nejvhodnější kandidáty řešící úkoly v jednotlivých krocích a podrobně jsme se věnovali jejich popisu, odvození a nastavení.

Představené schéma se soustředí na objekty reprezentované objemovým meshem a při jejich deformaci zahrnuje jak působení vnějších silových polí, tak vzájemné mechanické působení. Velká pozornost byla věnována kvalitě řešení kolizí, protože na ní velmi závisí celkový dojem ze simulace.

Od schématu očekáváme, že bude schopno v reálném čase simulovat objekty obsahující řádově $10^3$ čtyřstěnů a nebude trpět klasickými neduhy, zejména pokud bude mít jeden objekt spočívat v klidu na jiném.

Musíme si však uvědomit, že jednak díky tomu, že penetrační hloubky jsou pouze aproximovány, a jednak proto, že zanedbáváme při detekci kolizí průniky dvou stěn, se nelze úplně vyhnout menšímu prolínání kolidujících těles. I přesto však schéma slibuje rozumné výsledky.

# Kapitola 7

# Výsledky a měření

Algoritmus navržený v kapitole 6 jsme implementovali na platformě Win32 a otestovali na sadě dat. V této kapitole rozebereme nastavení a omezení jeho jednotlivých částí a problémy, se nimiž se mohou potýkat. Tyto problémy jsou však koncepčního charakteru, proto vysvětlíme důvody jejich vzniku a případně navrhneme adekvátní opatření. Na konci kapitoly učiníme závěr ze získaných měření a prezentujeme ukázky z výstupu programu.

## 7.1 Použitá data

Předběžné testy deformovatelného modelu probíhaly na různě kvalitních objemových meshích, které byly sestrojeny z povrchových meshů použitím volně dostupného programu `TetGen`[1]. Tyto modely obsahovaly 100–7000 čtyřstěnů a 200–10000 hran. Zde se ovšem potvrdilo, že různá lokální hustota uzlů a především velká variabilita délek hran (až několik řádů) velmi negativně ovlivňuje stabilitu. Tyto modely byly vloženy do scény a ponechány bez jakýchkoli vnějších vlivů, přesto se nastřádané numerické chyby vzniklé při vyhodnocování vnitřních sil rychle projevily a celý model znehodnotily i při extrémně malém časovém kroku 1ms.

Ukázalo se, že pro rozumné použití je zapotřebí pracovat s převážně pravidelnými meshi, kde vzdálenosti mezi body jsou přibližně stejné a všechny čtyřstěny mají podobné objemy. Komplexní meshe simulujeme tak, že je vnoříme do hrubé pravidelné sítě a s ní provádíme simulaci. Změny v polohách bodů jsou promítnuty zpětně na jemný model pomocí barycentrických souřadnic — každý bod jemného meshe má uložen čtyřstěn, do kterého patří, a barycentrické souřadnice v tomto čtyřstěnu.

Takto spojené meshe mají dvě výhody. Jednak je simulace mnohem stabilnější a jednak se tímto způsobem omezí artefakty vznikající při prolínání dvou objektů. Jak jsme zmínili v předchozích kapitolách, prolínání nelze úplně zabránit, ale pokud se částečně protínají dva hrubé meshe, obalené jemné meshe se ještě protínat nemusí. Zde však hodně závisí na přesnosti, s jakou hrubý mesh obaluje vnořený objekt,

[1] Domácí stránka projektu TetGen je `http://tetgen.berlios.de/`

protože může docházet i k opačnému efektu — modely se od sebe odrážejí, přestože se mezi nimi nachází dostatek místa, nebo se nad sebou jakoby vznášejí.

V našem případě jsme jako základní hrubý mesh použili pravidelnou mřížku $10 \times 10 \times 10$ krychliček. Každá krychlička byla pravidelně rozdělena na pět čtyřstěnů. Tabulka 7.1 popisuje složitost jednotlivých objektů měřenou počtem jejich primitiv.

| **Model** | **Uzlů** | **Hran** | **Troj.** | **Čtyřstěnů** |
|---|---|---|---|---|
| Kostka | 216 | 990 | 300 | 625 |
| | 2402 | — | 4800 | — |
| Pes | 113 | 451 | 216 | 232 |
| | 4004 | — | 8004 | — |
| Prstence | 322 | 1460 | 572 | 860 |
| | 6000 | — | 12000 | — |

Tabulka 7.1: Použité modely. Uveden je celkový počet uzlů, hran, povrchových trojúhelníků a čtyřstěnů hrubých meshů a pod nimi počet uzlů a povrchových trojúhelníků jemných vnořených meshů.

## 7.2 Deformovatelný model

**Koeficienty.** Není překvapením, že zvolený deformovatelný model založený na zobecněných pružinkách má poměrně velkou tendenci oscilovat. Z toho důvodu je prakticky nemožné se obejít bez útlumu. Koeficienty útlumu se však velmi liší v závislosti na struktuře použitého modelu a hmotnostech jeho uzlových bodů. V zahrnutých scénách používáme koeficienty tlumení v rozsahu 0,01–2. Čím větší tlumení je aplikováno, tím méně model osciluje; je-li však překročena určitá hranice, začne model velmi rychle kmitat nebo ztratí stabilitu úplně.

Oscilace je nepříjemná zejména kvůli problémům, které způsobuje při detekci a řešení kolizí. Těm se věnujeme později.

Koeficienty vynucující zachování vzdáleností se též promítají do kmitání modelu. Tvrdší materiály ($k_d$ = 40–100) kmitají s větší frekvencí a jejich celkové chování připomíná gumová tělesa — relativně málo se deformují a dobře se odrážejí. Materiály s menšími konstantami ($k_d$ = 1–40) připomínají spíše rosolovité látky, poměrně hodně se deformují a nemají takovou tendenci se vracet k původnímu tvaru.

Koeficienty zaručující zachování objemu používáme především proto, abychom zabránili invertování čtyřstěnů. Ve většině simulací je $k_v$ = 10–20.

**Hmotnosti.** Pro většinu simulací se ukazuje, že hmotnosti 50–100 gramů dávají nejlepší výsledky. V případě, že je hmotnost hodně malá, udělují uvažované síly bodům příliš velká zrychlení, která destabilizují celý systém. Pokud je naopak hmotnost velká, získávají nad vnitřními silami převahu vnější síly, zejména gravitace, která model velmi snadno slisuje.

**Časový krok.** Integrační schema je ovlivněno velikostí časového kroku $h$. Protože simulace může v závislosti na rozsahu kolizí běžet různě rychle, používáme adaptivní krok, který ovšem v rámci zajištění stability shora omezujeme. Maximální povolený krok je 15ms. Zobrazování scény provádíme pouze každý druhý krok simulace, tudíž maximální simulovaný krok mezi dvěma snímky je 30ms. Měření ukazují, že algoritmus pracuje i na dnes průměrném PC s přibližně 40–60 snímky za sekundu, což je víc než dostatečná hodnota pro odpovídající simulaci. I kdyby počet snímků klesl na 30, pořád máme simulaci přibližně 1:1, za 1s totiž odsimulujeme 900ms.

## 7.3 Detekce kolizí

Ukazuje se, že algoritmus pro detekci kolizí je dostatečně rychlý pro nasazení v našich scénách s řádově $10^3$ čtyřstěnů a $10^2$ bodů; nejedná se dokonce ani o nejpomalejší část algoritmu, prvenství v tomto ohledu drží řešení kolizí a vykreslování scény.

Krom již zmíněného handicapu, kterým je neschopnost detekovat průnik dvou čtyřstěnů, které se protínají pouze stěnami, se objevuje další problém, kterým je detekce kolizí více těles současně. Může se stát, že jeden bod se bude nacházet uvnitř dvou různých těles. K této situaci dochází například v okamžiku, kdy dvě tělesa proniknou do sebe navzájem a penetrují i podlahu. Detektor kolizí je omezen pouze na dvě současně se protínající tělesa a preferuje řešení kolize s podlahou. Problémem naopak není, pokud oba dva koncové body jedné hrany penetrují různá tělesa; tento stav je v implementaci korektně ošetřen.

Díky tomuto nedostatku se rozšiřuje okruh situací, při kterých tělesa mohou zůstat protnuta (i přesto, že je kolize detekována). Korektní řešení by si vyžádalo rozšíření schopností detektoru kolizí; každý bod by musel mít uloženu množinu všech čtyřstěnů, do kterých padnul. Mnohem náročnější by pak bylo řešení kolizí, pro každý bod by se muselo hledat několik penetračních hloubek a vektorů posunutí a tyto výsledky rozumně konsolidovat.

Protože v praxi k takovým situacím nedochází příliš často (pokud se záměrně nesnažíme dosáhnout tohoto stavu) a protože jsou zpravidla odstraněny díky vyřešení kolizí zbývajících kolidujích vrcholů, budeme tento nedostatek ignorovat.

## 7.4 Řešení kolizí

Nejsložitější proces, který během simulace probíhá, je řešení kolizí. Ten se opírá o odhady penetračních vektorů, které nejsou vždy snadné. Zásadním problémem je kmitání modelů. Než se modely ustálí, vyžaduje to určitý čas. Během tohoto času se přes jakoukoli snahu aproximovat penetrační vektory spojitě nedá zabránit mírnému poskakování jednoho objektu po druhém, i když by na sobě měly spočívat klidněji. Tento jev lze pozorovat zejména při simulacích geometricky pravidelných objektů, například dvou krychlí. Naopak ve scénách s jemnými povrchovými meshi jsou tyto artefakty nepozorovatelné.

Na druhou stranu, v době, kdy jsou modely ustáleny, získávají sílu numerické chyby. Penetrační hloubky mezi dvěma na sobě spočívajícími tělesy jsou velmi malé

a díky druhé mocnině ve váhové funkci (6.4) se ještě zmenšují. V některých případech díky zaokrouhlovacím chybám byla celková váha vyhodnocena $+\infty$. Tento problém byl snadno odstraněn tak, že bodům s penetračními hloubkami nedosahujícími určitého prahu byla uměle nastavena nulová penetrační hloubka i nulový penetrační vektor.

Díky nedokonalému detektoru kolizí a výpočtu odezvy se může stát, že se jeden bod objektu do jiného objektu zanoří náhle. Velmi náchylné jsou v tomto ohledu hraniční body v okamžiku, kdy jedno těleso pomalu sklouzává z druhého. Může dojít k tomu, že penetrační hloubka odhadnutá na základě popsaného algoritmu (přestože se jeví rozumně) je ve skutečnosti nevhodná. Problém názorně ilustruje obrázek 7.1. Kvůli nerozumnému odhadu penetračního vektoru se generuje i špatná odezva — zasažený bod se vzdaluje od správné polohy, sklouzávající těleso jej *tlačí* před sebou místo toho, aby pod něj bod *zajel*. Ještě výrazněji se problém projevuje v případě složitějších hrubých meshů, kde může vyústit v rušivé *třepání se* objektů. Je nutné si však uvědomit, že nejde o chybu v algoritmu, ale o nedostatek, který plyne z vágní definice penetračního vektoru. Není totiž vždy jasné, jak správně poznat, kterým místem na povrchu tělesa kolidující bod proniknul. Více viz sekce 6.4.1.

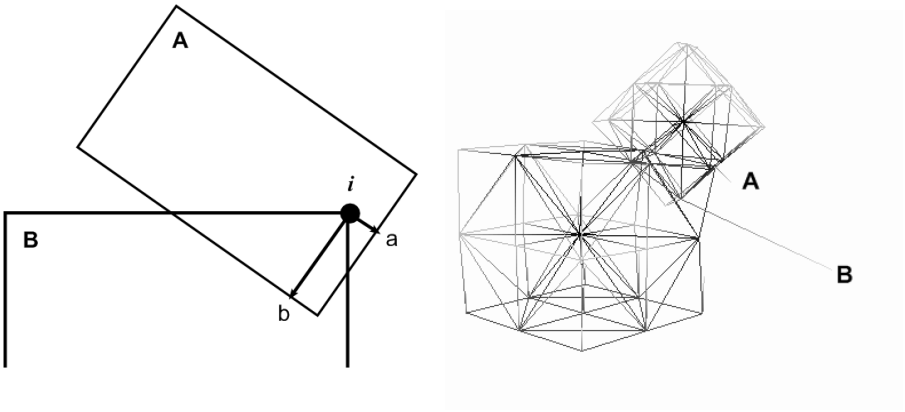
Až na popsaný koncepční nedostatek, dává proces řešení kolizí velmi dobré výsledky a plně se podařilo dosáhnout vytyčeného cíle — omezit poskakování těles po sobě na minimum.

Nepodařilo se však úplně uspokojivě vyřešit problém spočívání dvou těles na sobě v klidu (resting contacts), tělesa mají tendenci ze sebe po čase sklouznout. Vinu nese nepříliš dobrý model tření spolu s malými vibracemi modelu, které vznikají při velmi mělkém zanoření.

## 7.5 Zobrazování

Výsledky měření naznačují, že zobrazování může být paradoxně nejdražší operací celého programu. Zde jsou na vině dva faktory. První je velmi slabý grafický akcelerátor, který byl při měření použit; dnes lze jej klasifikovat jako low-end GPU. Na druhou stranu je nutné podotknout, že pro vykreslování vytvořených scén bohatě dostačuje, limitující je v tomto případě spíš rychlost sběrnice, která souvisí s druhým, zásadnějším faktorem, jímž je fakt, že se v každém kroku musí na grafickou kartu posílat všechna data znova. O přepočet nových poloh bodů jemných povrchových meshů přes barycentrické souřadnice a odhad normál trojúhelníků se stará procesor v hlavní paměti.

První zmíněný nedostatek by bylo možné částečně eliminovat použitím dnes běžně dostupných výkonnějších akcelerátorů ve sběrnici PCI-E. V případě druhého faktoru by bylo možné zvážit využití přímo programovatelného GPU k výpočtům nových opravených poloh jemných meshů. Takové úvahy však přesahují rámec této práce.




Obrázek 7.1: Ukázka vzniku chybného odhadu při výpočtu penetrační hloubky. Nalevo zobrazeno schematicky; $A$ sklouzlo z $B$, bod $i$ se *náhle* ocitnul uvnitř $A$. Odhadovaná penetrační hloubka je $a$, přitom bychom preferovali (mnohem větší) $b$. Napravo je celá situace zachycena přímo v programu, penetrační hloubky jsou záměrně stonásobně zvětšeny. Pravý horní roh spodní krychle má penetrační hloubku $A$ mířící ven a nově zanořený prostřední bod z dolní podstavy menší krychle má podobně špatně odhadnutou hloubku $B$.

## 7.6 Měření

Měření rychlosti algoritmu bylo provedeno na stroji s procesorem Intel Pentium 4 s taktem 1,8GHz a 1GB RAM. Pro vykreslování bylo použito grafického akcelerátoru NVIDIA GeForce4 MX 440 s 64MB VRAM osazeném v AGP. Scéna se renderovala v rozlišení 800×600 pixelů s 32bitovou barevnou hloubkou.

Program byl implementován v prostředí MS Visual Studio C++ verze 8.0 a zkompilován s optimalizacemi pro rychlost `/O2 /Ot` (Maximize Speed, Favor Fast Code).

### 7.6.1 Scény

Pro měření byly sestaveny dvě scény. První scéna sestávala ze tří modelů krychlí nad sebou, cílem bylo demonstrovat schopnost v reálném čase zvládnout výpočty s řádově tisíci čtyřstěny.

Druhá scéna obsahovala celkem čtyři hrubé objemové a čtyři jemné povrchové meshe, které byly do objemových vnořeny. Hlavní záměr byl otestovat vzájemné chování hrubých meshů a celkový dojem ze simulace, jsou-li zobrazovány jemné meshe. Přesné parametry scén zachycuje tabulka 7.2.

| **Scéna** | **Uzlů** | **Hran** | **Trojúhelníků** | **Čtyřstěnů** |
|---|---|---|---|---|
| 3 × Kostka | 648 | 2970 | 900 | 1875 |
| 4 × Pes | 452 | 1804 | 864 | 928 |
| | 16000 | — | 32000 | — |

Tabulka 7.2: Popis scén a jejich rozsah.

### 7.6.2 Technika měření

Uvažovaná scéna byla vždy 8× spuštěna a testována. Z každého měření byla dvě vypuštěna (nejlepší a nejhorší) a ze zbylých byl spočten aritmetický průměr. Sledovány byly především průměrný a minimální počet snímků za sekundu, průměrný a maximální počet kolidujících bodů, průměrný a maximální počet trojúhelníků, z nichž byly voleny kontaktní trojúhelníky. Výsledky jsou obsaženy v tabulce 7.3. Za povšimnutí stojí, že se ve scénách při kolizích uvažuje průměrně 40–50% povrchových trojúhelníků.

| **Scéna** | **FPS prům.** | **FPS min** | **PKU** | **MKU** | **PKT** | **MKT** |
|---|---|---|---|---|---|---|
| 3 × Kostka | 31 | 29 | 92 | 147 | 370 | 575 |
| 4 × Pes | 30 | 25 | 105 | 142 | 451 | 596 |

Tabulka 7.3: Výsledná měření. PKU a MKU značí průměrný resp. maximální počet kolidujících uzlů. PKT a MKT značí průměrný resp. maximální počet kolidujících trojúhelníků.

Měření zahrnovala i průměrné časy potřebné k vyhodnocení jednotlivých kroků algoritmu, ty však byly *příliš malé* a proto je neuvádíme. Místo nich uvádíme v tabulce 7.4 nejdelší naměřené časy. Je dobré mít na paměti, že nejhorší časy vznikají v okamžiku prvního doteku objektů, jelikož je nutné hledat průsečíky s povrchem a kontaktní trojúhelníky bez využití časové koherence, slovo tedy mají nejhorší časové odhady.

| **Scéna** | **NF** | **UI** | **CD** | **TC** | **PD** | **CR** | **FI** | **R** |
|---|---|---|---|---|---|---|---|---|
| 3 × Krychle 5×5×5 | 16 | 32 | 32 | 16 | 16 | 47 | 16 | 47 |
| 4 × Pes | 16 | 16 | 16 | 16 | 16 | 16 | — | 63 |

Tabulka 7.4: Výsledná měření nejhorších časů potřebných k provedení jednotlivých kroků, údaje jsou v milisekundách: NF vyhodnocení interních sil, UI integrace v čase, CD detekce kolizí, TC sbírání trojúhelníků, PD aproximace penetračních vektorů, CR vyřešení kolize, FI uložení poloh v čase (pro Verletovo schema), R renderování scény.

## 7.7 Výsledky

Ukázali jsme, že námi implementovaný algoritmus je schopen pracovat v interaktivních rychlostech s řádově tisíci čtyřstěny i na dnes spíše podprůměrném stroji. Přes všechny nedostatky, které jsme podrobně rozebrali a pro něž jsme navrhli rozumná řešení, jsou vizuální výsledky experimentů velmi dobré a srovnatelné s výsledky, kterých dosáhli autoři článků, z nichž jsme vycházeli. Čtenář sám může toto tvrzení posoudit z obrázků v příloze C.

Na obrázcích se nesnažíme zakrývat nejmarkantnější nedostatky našeho řešení. Tím je jednak *vznášení se* objektů nad sebou zapříčiněné poměrně hrubou aproximací jemného meshe. Na tomto místě bychom rádi připomněli, že se nejedná o chybu algoritmu, ale že jde čistě o problém týkající se kvality simulovaných dat. Druhým problémem, který lze na obrázcích těžko zachytit, ale v simulovaných scénách je dobře patrný, jsou *náhle* detekované průniky těles vedoucí k rychle se měnící odezvě, což se projevuje jako *třepání* objektu a působí velmi rušivě.

# Kapitola 8

# Závěr

V diplomové práci jsme se soustředili na modelování deformací pevných těles. Nejprve jsme zmapovali aktuální situaci na poli deformovatelných modelů. Tato oblast je velmi bohatá a rozvinutá. Existuje mnoho modelů, které se navzájem velmi liší mnoha aspekty — typem používané reprezentace objektů, kvalitou a rychlostí simulace, zamýšleným nasazením a mnoha dalšími. Z celého odvětví jsme se zaměřili na ty deformační modely, které podporují deformace pevných těles reprezentovaných povrchovými nebo objemovými meshi v reálném čase. V práci popisujeme tři vybraná schemata. Všechna prezentovaná schemata jsou rozumně nastavitelná prostřednictvím relativně malého množství parametrů, které ovlivňují chování simulových modelů, a všechna dbají na konzistenci reprezentace objektů, nevyžadují žádné lokální úpravy jejich kvality.

V další části jsme se zabývali detekcí kolizí a jejich řešením. I tomuto tématu jsme věnovali velkou pozornost, protože jsme usilovali o co možná nejvěrohodnější výsledky. Důsledkem je, že implementovaný algoritmus řešící kolize nevyžaduje žádné složité nastavení a chová se realisticky i v situaci, kdy na sobě v klidu spočívají dvě tělesa.

Součástí práce bylo i ověření chování navrženého algoritmu v praxi. Prokázalo se, že algoritmus je schopen pracovat v reálném čase s řádově $10^3$ čtyřstěny i na průměrné výkonném stolním počítači. Deformovatelný model vykazuje rozumné chování, nicméně má mnohdy zbytečně velkou tendenci oscilovat, což snižuje věrohodnost simulace. Též nastavení parametrů, zejména tlumení, není vůbec intuitivní a je potřeba jej provést metodou pokus–omyl. Řešení kolizí poskytuje velmi dobré výsledky, až na jednu speciální situaci, kterou jsme popsali a vysvětlili.

Ve všech úvahách, které jsme v práci vedli, jsme zanedbali celou řadu jevů. Případná další rozšíření se týkají hned několika oblastí.

Do celého modelu by bylo možné zahrnout další vlivy a proměnné. Jedná se zejména o rozumný model dynamického tření, síly, která vzniká na kontaktní ploše mezi dvěma objekty a brání pohybu. Vycházet bychom mohli např. ze zmíněného Coulombova modelu a relativní rychlosti aproximovat s použitím kontaktních trojúhelníků.

Další rozšíření by se mohlo týkat tepelných a teplotních vlivů, protože chování všech látek se mění v závislosti na teplotě. Nejblíže má k tomuto rozšíření metoda konečných prvků, protože vychází z mechaniky kontinua, která tyto vlivy přímo zahrnuje.

Deformovatelný model by bylo možno dále obohatit o větší spektrum podporovaných deformací, především o plastické deformace. Existují práce, které se tímto problémem zabývají a na kterých by bylo možné stavět [OBH02]. Princip spočívá v tom, že trvalá část deformace se uchovává v objektu, například ve formě speciálních tenzorů deformace. Při odvozování sil se pak uvažované tenzory rozloží na elastickou a plastickou část a výpočty se dále provedou jen na základě elastického příspěvku.

Dalším typem deformací, které bychom mohli zahrnout, je praskání a lámání. To je však poměrně náročný úkol, techniky se liší v závislosti na použitém deformovatelném modelu a mnohé není možné simulovat v reálném čase. Důvodem je zejména fakt, že je potřeba měnit konektivitu meshe, generovat nová primitiva a případně rušit stávající.

Celkový dojem ze simulace by dále prohloubilo přidání dynamických stínů, z nichž by uživatel mohl lépe odhadovat polohu těles ve scéně.

# Literatura


[ABCO+01] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[Ale06] Marc Alexa. Mesh editing based on discrete laplace and poisson models. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 51–59, New York, NY, USA, 2006. ACM Press.

[Bar01] David Baraff. Implicit methods for differential equations. In *SIGGRAPH 2001 Course Notes*, 2001.

[BHW96] Ronen Barzel, John F. Hughes, and Daniel N. Wood. Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation '96*, pages 183–197, 1996.

[BSS05] Miroslav Brdička, Ladislav Samek, and Bruno Sopko. *Mechanika kontinua*. Academia, 2005.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space - time adaptive sampling. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 31–36. ACM Press / ACM SIGGRAPH, 2001.

[GD04] Stéphane Guy and Gilles Debunne. Monte-carlo collision detection. Technical Report RR-5136, INRIA, March 2004.

[GM97] Sarah F.F. Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Lab., November 1997.

[GRLM03] Naga K. Govindaraju, Stephane Redon, Ming C. Lin, and Dinesh Manocha. Cullide: Interactive collision detection between complex models in large environments using graphics hardware. In *Proceedings of the Eurographics/SIGGRAPH Graphics Hardware Workshop*, 2003.

[GST05] D. Göddeke, R. Strzodka, and S. Turek. Accelerating double precision FEM simulations with GPUs. In F. Hülsemann, M. Kowarschik, and U. Rüde, editors, *Simulationstechnique 18th Symposium in Erlangen,*

*September 2005*, volume Frontiers in Simulation, pages 139–144. SCS Publishing House e.V., 2005. ASIM 2005.

[GVSS00] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. In *Computer Graphics Proceedings (SIGGRAPH 2000)*, pages 95–102, 2000.

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994.

[HTG04] Bruno Heidelberger, Matthias Teschner, and Markus Gross. Detection of collisions and self-collisions using image-space techniques. In *WSCG*, pages 145–152, 2004.

[HTK+04] Bruno Heidelberger, Matthias Teschner, Richar Keiser, Matthias Müller, and Markus Gross. Consistent penetration depth estimation for deformable collision response. In *Proc. Vision, Modeling, Visualization VMV '04*, pages 339–346, 2004.

[JP04] Doug L. James and Dinesh K. Pai. Bd-tree: Output-sensitive collision detection for reduced deformable models. In *ACM Transactions on Graphics (SIGGRAPH 2004)*, August 2004.

[Kaf04] Martin Kafka. Vizualizace praskání a rozlamování objektů. Master's thesis, České Vysoké Učení Technické v Praze, 2004.

[Kim05] Stefan Kimmerle. *Collision Detection and Post-Processing for Physical Cloth Simulation*. PhD thesis, Eberhard-Karls-Universität Tübingen, 2005.

[KZ04] Jan Klein and Gabriel Zachmann. Point cloud collision detection, 2004.

[LSCO+04] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[MAC04] Damien Marchal, Fabrice Aubert, and Christophe Chaillou. Collision between deformable objects using fast-marching on tetrahedral models. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–129, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[MC95] Brian Mirtich and John Canny. Impulse-based dynamic simulation. In *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, pages 407–418, Natick, MA, USA, 1995. A. K. Peters, Ltd.

[MHS05] Jesper Mosegaard, Peder Herborg, and Thomas S. Sorensen. A gpu accelerated spring-mass system for surgical simulation. In *13th Medicine Meets Virtual Reality (MMVR) Studies in Health Technology and Informatics 2005*, pages 342–348, 2005.

[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 471–478, New York, NY, USA, 2005. ACM Press.

[MKN+04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[MW88] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animationr3. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1988. ACM Press.

[NMK+06] Anrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. In *Computer Graphics Forum, Vol. 25, issue 4*, pages 809–836, 2006.

[OBH02] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 291–294, New York, NY, USA, 2002. ACM Press.

[OD01] Carol O'Sullivan and John Dingliana. Real vs. approximate collisions: When can we tell the difference. In *SIGGRAPH 2001 Sketches Program*, page 249, 2001.

[OH99] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[PKKG03] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 641–650, New York, NY, USA, 2003. ACM Press.

[PZvBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 335–342. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[SBT07] Jonas Spillmann, M. Becker, and Matthias Teschner. Non-iterative computation of contact forces for deformable objects. In *Journal of WSCG*, volume 15, pages 33–40, 2007.

[ST05] Jonas Spillmann and Matthias Teschner. Contact surface computation for coarsely sampled deformable objects. In *Proc. Vision, Modeling, Visualization VMV '05*, pages 289–296, 2005.

[THM+03] Matthias Teschner, Bruno Heidelberger, Matthias Mueller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization*, pages 47–54, 2003.

[THMG04] Matthias Teschner, Bruno Heidelberger, Matthias Müller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *Computer Graphics International, 2004. Proceedings*, pages 312–319, 2004.

[TKZ+04] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, Laks Raghupathi, A. Fuhrmann, Marie-Paule Cani, François Faure, N. Magnetat-Thalmann, and W. Strasser. Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)*, pages 119–139. Eurographics Association, Eurographics Association, 2004.

[WB93] Andrew Witkin and David Baraff. Differential equations basics. In *SIGGRAPH93 20th International Conference on Computer Graphics and Interactive Techniques*, pages B1–B8, 1993.

[WDGT01] Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshe. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 349–358. Blackwell Publishing, 2001.

# Dodatek A

# Obsah CD

## A.1 Struktura adresářů

| Adresář | Obsah |
|---|---|
| `/src` | Zdrojové kódy aplikace |
| `/inc` | Hlavičkové a zdrojové soubory třetích stran |
| `/bin` | Přeložená aplikace |
| `/bin/data` | Ukázková data |
| `/doc` | Text diplomové práce |
| `/lib` | Knihovny třetích stran |
| `/media` | Ukázky výstupu programu |

Tabulka A.1: Struktura adresářů na přiloženém CD a jejich obsah.

## A.2 Knihovny třetích stran

Program využívá dvou volně dostupných knihoven

1. **Boost**[1] verze 1.34.1. Knihovna obsahuje řadu pomocných konstrukcí, algoritmů, datových struktur a maker, které usnadňují programování v C++. Z knihovny využíváme konstrukci `BOOST_FOREACH`.

2. **Microsoft DirectX SDK**[2] vydání February 2006. Z knihovny používáme část Direct3D pro vykreslování scény a DirectInput pro zpracování vstupu z klávesnice a myši.

[1] `http://www.boost.org/`
[2] `http://msdn.microsoft.com/directx/`

# Dodatek B

# Ovládání programu

## B.1 Systémové požadavky

Doporučená HW konfigurace počítače: procesor Intel Pentium 4 1.8GHz nebo lepší, 512MB RAM, grafický akcelerátor schopný provádět transformaci a osvětlení (HW T&L), například NVIDIA řady GeForce 4 nebo vyšší.

Aplikace pro běh potřebuje nainstalovaný **Microsoft DirectX Runtime**[1] alespoň verze 9.0.

## B.2 Spuštění a ovládání programu

Program se spouští z příkazové řádky s jedním parametrem, kterým je název konfiguračního souboru, jenž obsahuje popis scény, a ovládá se pomocí klávesnice a myši.

| Klávesa | Význam |
|---|---|
| **ESC** | Ukončení simulace a konec programu. |
| **P** | Pozastavení nebo spuštění simulace (pause). |
| **Tab** | Přepínání mezi objekty ve scéně.<br>Na aktivní objekt se vztahují klávesy **V**, **F**, **D**, **1**–**4**. |
| **V** | Skryje nebo zobrazí aktivní objekt. |
| **F** | Zobrazí nebo skryje celkové síly působící v bodech hrubého meshe. |
| **D** | Zobrazí nebo skryje penetrační hloubky kolidujících bodů.<br>Úsečky jsou stonásobně zvětšeny. |
| **1–4** | Přepíná renderovací režimy.<br>První tři režimy zobrazují jemný povrchový mesh.<br>čtvrtý režim zobrazuje drátěný hrubý mesh (wireframe). |
| šipky | Šipky slouží k ovládání polohy kamery. |
| +/− | Zoom kamery. |
| **LMB** | Levé tlačítko myši aplikuje liniovou sílu na nejbližší bod aktivního objektu. |
| **RMB** | Držením pravého tlačítka a pohybem myši lze ovládat polohu kamery. |

[1] Volně ke stažení na stránkách `http://msdn.microsoft.com/directx/`

## B.3 Struktura konfiguračního souboru

Konfigurační soubor obsahuje seznam deformovatelných modelů, jejich nastavení a umístění ve scéně. Struktura souboru vypadá následovně

```
# Komentář
DeformableModels

DeformableModel {
  Parametr    Hodnota
  ...
}

Deformable Model {
  Parametr    Hodnota
  ...
}

...
```

Každý deformovatelný model ve scéně musí mít svůj vlastní záznam, v němž jsou specifikovány jeho parametry. Seznam přípustných parametrů je uveden v tabulce B.1. Řádky, které jsou uvozeny znakem # (mříž) jsou chápány jako komentáře a přeskakovány.

| **Parametr** | **Význam** | **Impl.** |
|---|---|---|
| Name | Jméno tělesa, které se zobrazí ve scéně. | "N/A" |
| *Path* | Cesta k meshi. | |
| Immersed | Cesta k jemnému povrchovému meshi. | "" |
| Color | Barva tělesa. | 0xc8c864 |
| Mass | Hmotnost bodů. | 1 |
| Scale | Změna měřítka. | $(1, 1, 1)$ |
| Translate | Posunutí. | $(0, 0, 0)$ |
| Rotate | Otočení. | $(0, 0, 0)$ |
| Damping | Koeficient útlumu. | 1 |
| DistancePreservation | Koeficient ovlivňující zachování vzdáleností. | 20 |
| VolumePreservation | Koeficient ovlivňující zachování objemů. | 10 |

Tabulka B.1: Nastavitelné parametry deformovatelného modelu, jejich význam a implicitní hodnoty.

Povinný je pouze parametr *Path*. Zbylé parametry jsou nepovinné a nejsou-li specifikovány, nastaví se na implicitní hodnoty. Je-li nastaven jemný povrchový mesh, zobrazuje se v renderovacích režimech 1–3; pokud ne, vykresluje se povrch hrubého meshe. Transformace jsou prováděny v následujícím pořadí: rotace, změna měřítka, posunutí.

Podlaha je tvořena rovinou $y = 0$. Tělesa by ji na začátku simulace neměla protínat, navíc by se neměla protínat ani mezi sebou navzájem.

Konkrétní nastavení scény obsahující jeden model provedeme následovně (pro názornost jsou použity všechny parametry)

```
# Scéna obsahující psa.
DeformableModels

DeformableModel {
  Name      Pes
  Path      Dog.do
  Immersed  Dog.s
  Color     255
  Mass      0.050
  Scale     1.0 1.0 1.0
  Translate 0.0 8.0 0.0
  Rotate    -1.57 0.0 0.0
  Damping   2
  DistancePreservation  60
  VolumePreservation    20
}
```

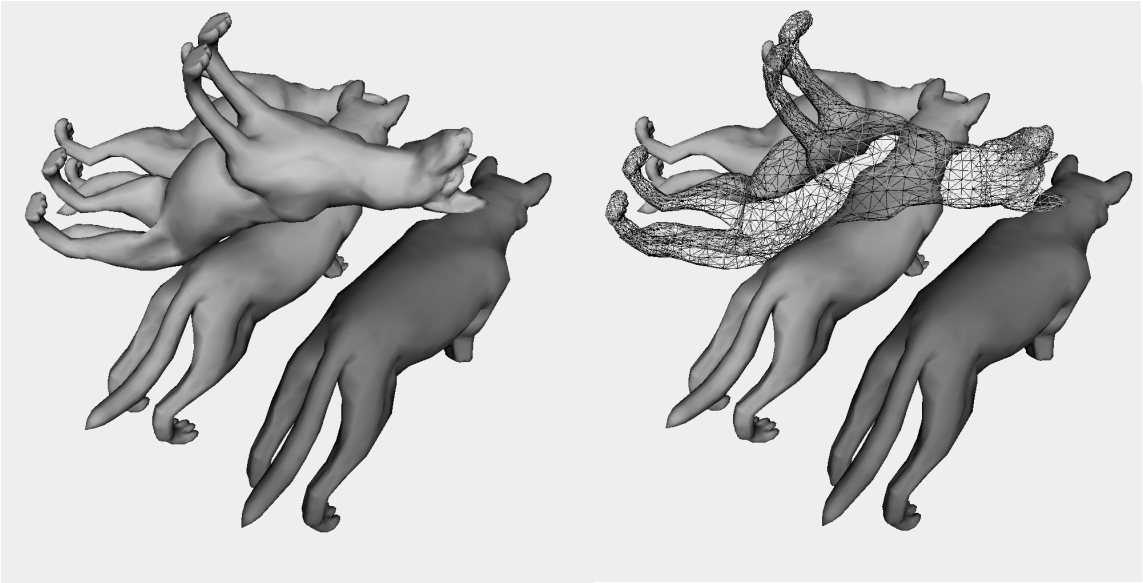
# Dodatek C

# Ukázky

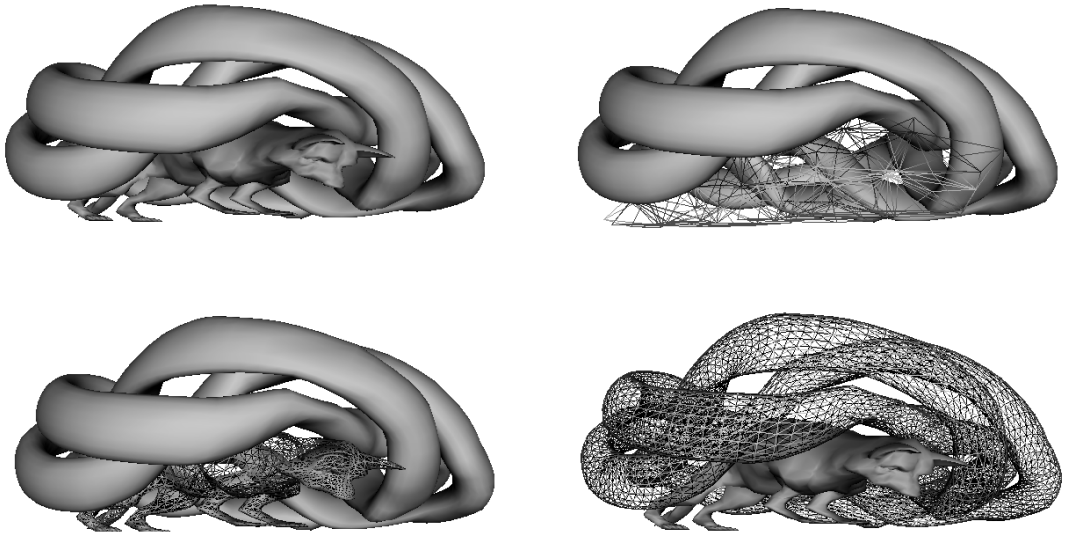
Z modelů, jež jsou uvedeny v tabulce 7.1, jsme vytvořili řadu scén, na nichž demonstrujeme chování navrženého algoritmu. V této příloze prezentujeme dosažené výsledky, jak dobré, tak špatné.

Rozumné chování dokazují obrázky C.1, C.2, C.3, C.4, C.5. Problém *vznášení se* dvou objektů nad sebou ilustruje obrázek C.6. Vliv parametrů $k_D$ a $k_V$ na chování modelu zachycují obrázky C.7 a C.8.

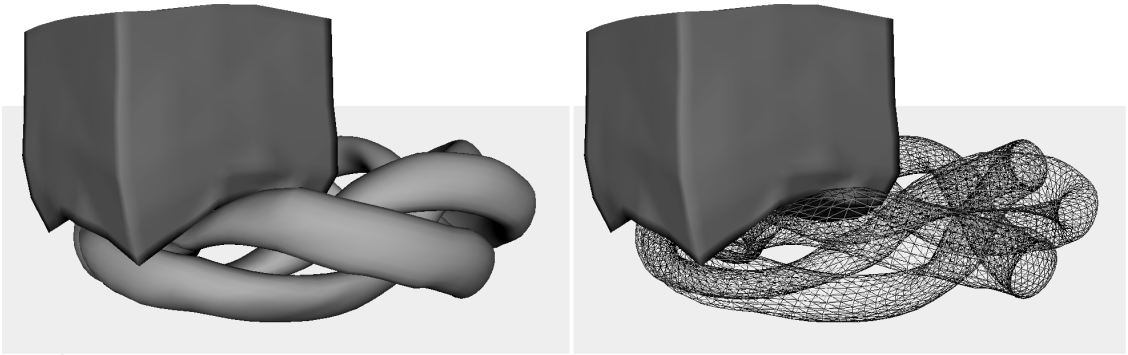
Vysloveně špatné chování je popsáno na obrázcích C.9 a C.10.



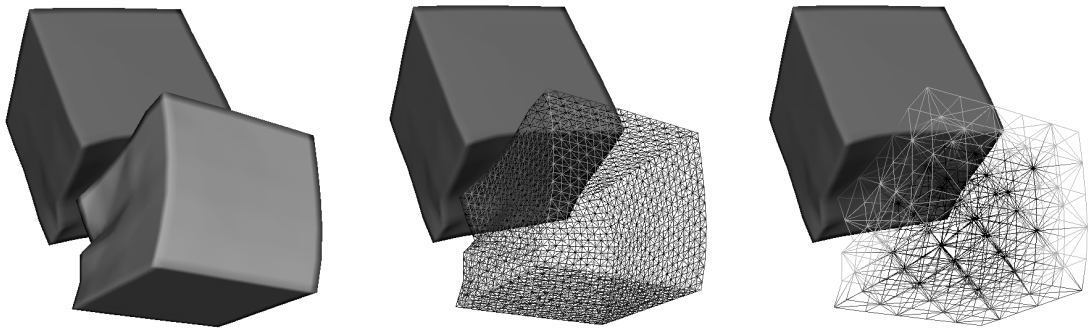
Obrázek C.1: Pes padající na tři další. Žádné pozorovatelné prolínání. Všichni psi mají $k_D = 60$, $k_V = 20$, $d_C = 2$, $m = 50$g.



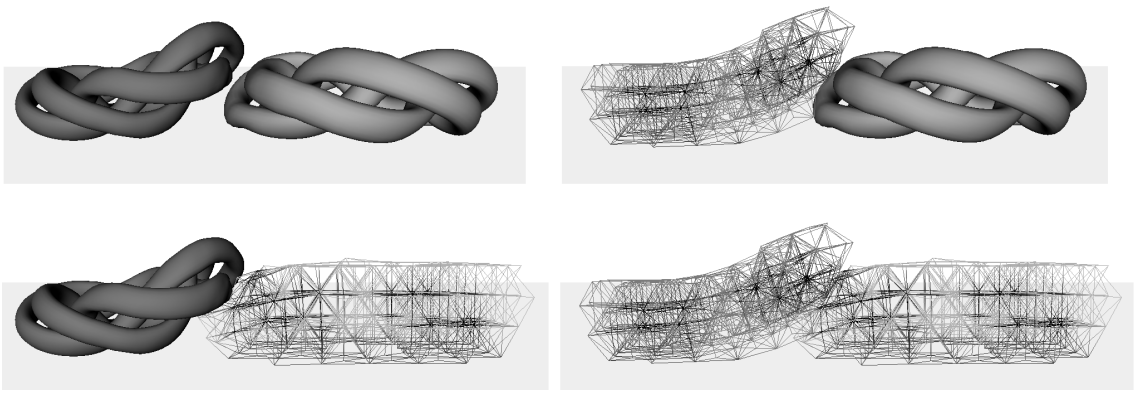
Obrázek C.2: Pes lisovaný prstenci. Žádné pozorovatelné prolínání, jediným artefaktem jsou *zlomené nohy*, což je zapříčiněno lineární interpolací barycentrickými souřadnicemi, protože zaručují pouze $C^0$ spojitost.



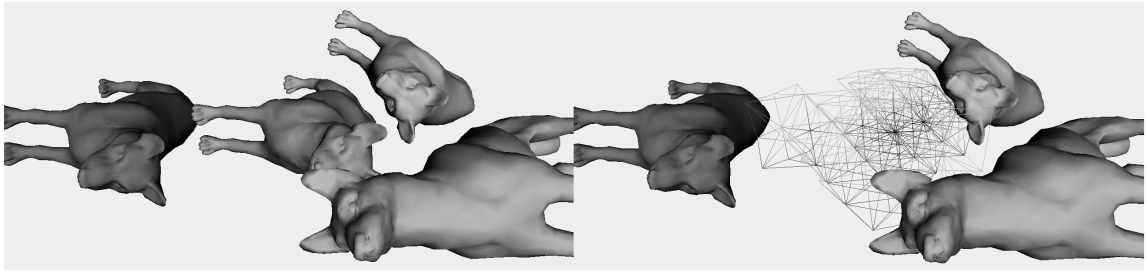
Obrázek C.3: Krychle ležící na prstencích. Žádné pozorovatelné prolínání.



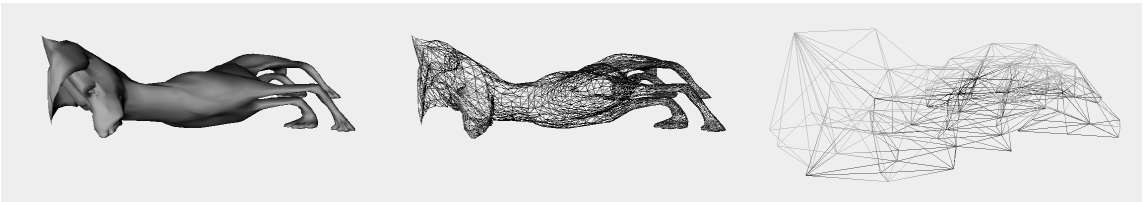
Obrázek C.4: Krychle sklouzávající po druhé krychli. Žádné pozorovatelné prolínání. Jediným artefaktem jsou příliš ostré zlomy na levé hraně spodní krychle. Na vině je kvalita hrubého meshe, který má jen 5×5×5 krychliček. Celkový dojem nemůže vylepšit ani vnořená jemná krychle 20×20×20.



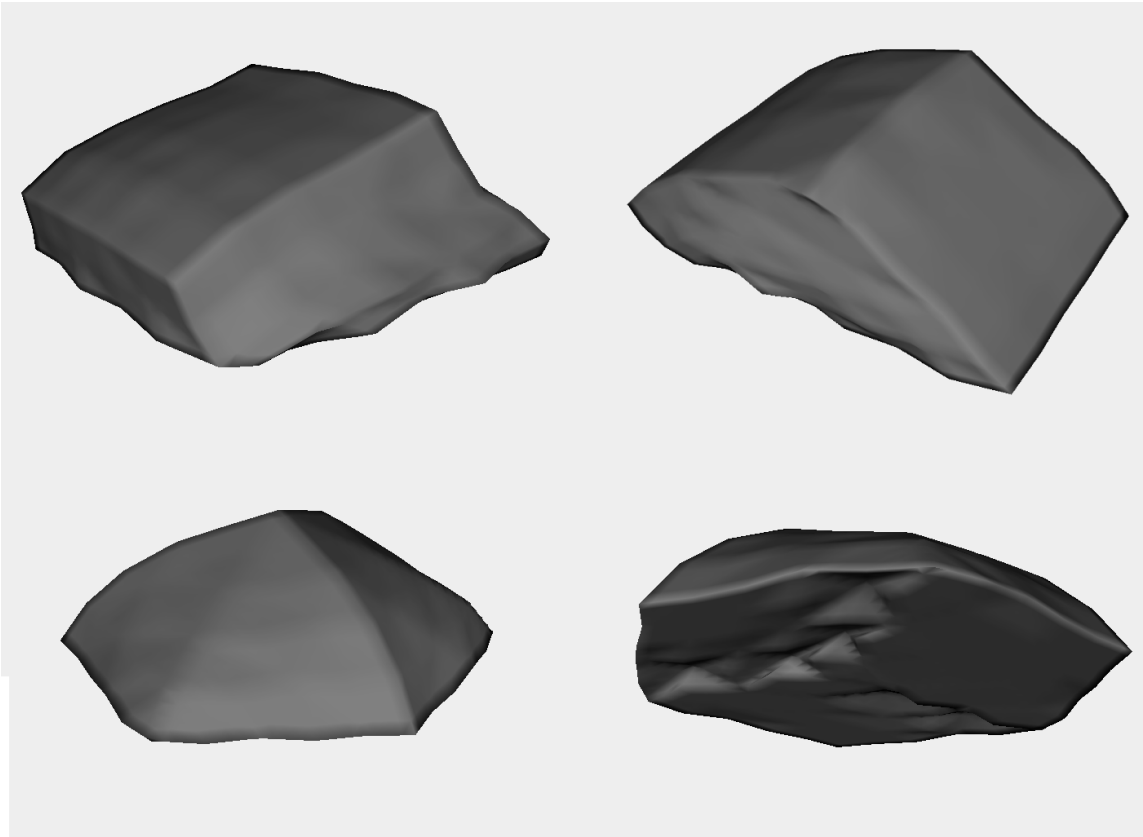
Obrázek C.5: Dva prstence sklouzávající po sobě. Jemné meshe jsou mírně vzdáleny, nicméně kontakt hrubých meshů je vyřešen uspokojivě.



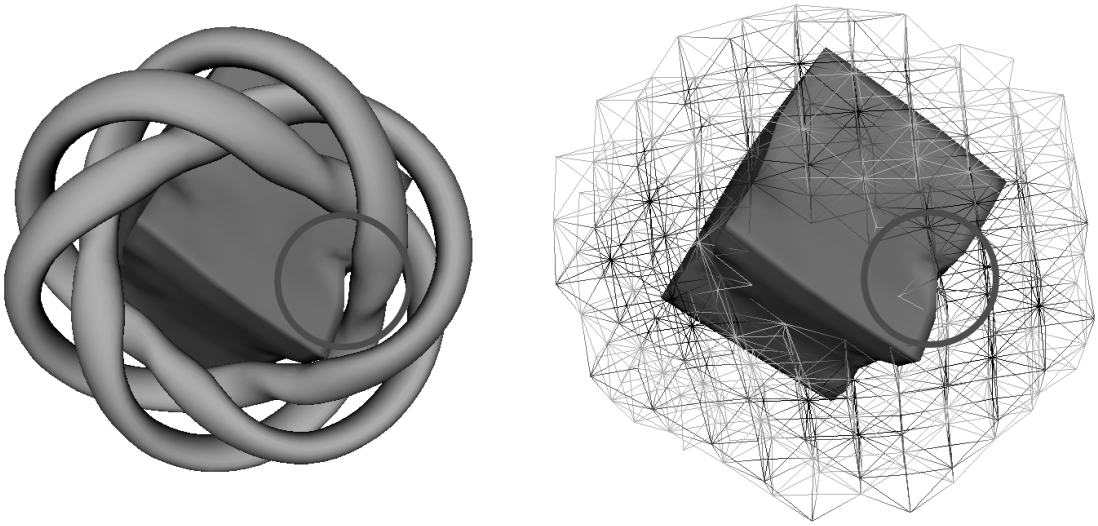
Obrázek C.6: Sklouzávání v případě scény se čtyřmi psy poodhaluje nedostatky aproximace hrubým meshem. Horní pes *se vznáší* nad ostatními, přestože by se jich měl dotýkat.



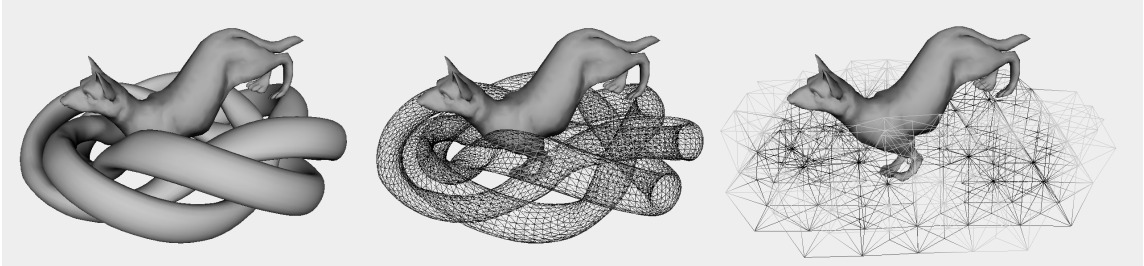
Obrázek C.7: Ukázka vlivu parametru $k_D$. Na psa je navíc aplikována liniová síla, která ho táhne za uši směrem vlevo. $k_D = 2$, $k_V = 20$, $d_C = 2$, $m = 50$g



Obrázek C.8: Ukázka vlivu parametru $k_V$. Horní dvě krychle mají $k_V = 10$, spodní dvě $k_V = 5$. Zbylé parametry jsou stejné; $k_D = 2$, $d_C = 2$, $m = 50$g.



Obrázek C.9: Zakroužkovaná oblast ukazuje problematické místo. Dochází zde ke komplikovanému průniku čtyřstěnů, výsledkem je hodně proměnlivá odezva. Tělesa se pak nepřirozeně *třepou*.



Obrázek C.10: Stejný problém jako na obrázku C.9 lze pozorovat i zde. Hrubý mesh psa v oblasti předních tlap se nepravidelně zanořuje do hrubého meshe prstenců, což vyvolává nepříjemné *třepání*.