VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

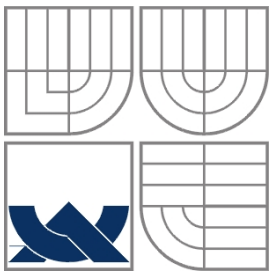
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PŘEVOD TROJÚHELNÍKOVÝCH POLYGONÁLNÍCH 3D SÍTÍ NA 3D SPLINE PLOCHY

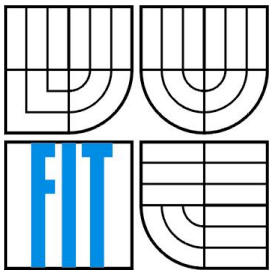
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ZDENĚK JAHN
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PŘEVOD TROJÚHELNÍKOVÝCH POLYGONÁLNÍCH 3D SÍTÍ NA 3D SPLINE PLOCHY

TRINANGULAR POLYGONAL 3D MESHES TO 3D SPLINE SURFACES REMESHING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

Zdeněk Jahn

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Přemysl Kršek, PhD.

BRNO 2007

Zadáni bakalářské práce/4022/2006/xjahnz00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Jahn Zdeněk**

Obor: Informační technologie

Téma: **Převod trojúhelníkových polygonálních 3D sítí na 3D spline plochy**

Kategorie: Počítačová grafika

Pokyny:

1. Seznamte se s problematikou trojúhelníkových polygonálních 3D sítí a 3D spline ploch.
2. Analyzujte základní možnosti převodu trojúhelníkových polygonálních 3D sítí na 3D spline plochy.
3. Navrhněte algoritmus (vhodně modifikujte existující) pro převodu trojúhelníkových polygonálních 3D sítí na 3D spline plochy.
4. Implementujte vybranou část navrženého algoritmu ve vybraném jazyce (C/C++, Java, Python, C#).
5. Zhodnoťte dosažené výsledky a stanovte další vývoj projektu

Literatura:

- Žara J., Beneš B., Felkel P.: Moderní počítačová grafika. 1. vyd. Praha, Computer press 1998, 448 s., ISBN 80-7226-049-9

Při obhajobě semestrální části projektu je požadováno:

- Provedení prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kršek Přemysl, Ing., Ph.D.,** UPGM FIT VUT

Datum zadání: 1. listopadu 2006

Datum odevzdání: 15. května 2007

L.S.

---

doc. Dr. Ing. Pavel Zemčík

vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Zdeněk Jahn**
Id studenta: 84348
Bytem: Krokova 22, 796 01 Prostějov
Narozen: 23. 03. 1985, Prostějov
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

……
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP:	Převod trojúhelníkových polygonálních 3D sítí na 3D spline plochy
Vedoucí/školitel VŠKP:	Kršek Přemysl, Ing., Ph.D.
Ústav:	Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP:	……

VŠKP odevzdal autor nabyvateli v:

tištěné formě	počet exemplářů: 1
elektronické formě	počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ■ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: ……

……..……	……..……
Nabyvatel	Autor

Abstrakt

Tato práce se zabývá problematikou převodu *nestrukturovaných trojúhelníkových 3D sítí* na vhodnější reprezentace (*quadrilaterální sítě nebo spline plochy*). Vysvětluje základní problémy spojené s nestrukturovanými sítěmi a důvody k jejich řešení. Klasifikuje použitelné metody, stručně popisuje nejvhodnější kandidáty. Detailně se věnuje vybrané metodě, jak teoretickému základu, tak konkrétní implementaci.

Klíčová slova

Trojúhelník, polygon, quadrilaterál, 3D síť, 3D spline plocha, NURBS, T-Spline, harmonická funkce, vektorové pole, trasování, zjednodušení, vyhlazování, přesíťování.

Abstract

This bachelor's thesis deals with the problem of the remeshing of unstructured triangular 3D meshes to more suitable representations (*quadrilateral meshes or spline surfaces*). It explains the basic problems related with the unstructured meshes and the reasons for its solution. It classifies the usable methods, describes the most suitable candidates briefly. It follows the chosen method in detail - both the theoretical matter and the specific implementation.

Keywords

Triangle, polygon, quadrilateral, 3D mesh, 3D spline surface, NURBS, T-Spline, harmonic function, vector field, tracing, simplification, smoothing, remeshing.

Citace

[Jahn 2007] Jahn Z.: *Převod trojúhelníkových polygonálních 3D sítí na 3D spline plochy*.
Bakalářská práce, Brno, FIT VUT v Brně, 2007

Převod trojúhelníkových polygonálních 3D sítí na 3D spline plochy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením
Ing. Přemysla Krška, PhD.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Chtěl bych poděkovat Ing. Přemyslu Krškovi, PhD. za odbornou pomoc, vstřícnost a ochotu při řešení tohoto projektu.

Obsah

1 Úvod

3D tělesa můžeme v počítači reprezentovat různými způsoby. Každý způsob modelování nabízí různé možnosti a nalezne uplatnění ve specifických aplikacích. Na druhé straně každý způsob přináší určité nevýhody nebo problémy, které se musí řešit. Nejběžnější reprezentační modely jsou CSG, šablonování, dekompoziční modely, hraniční reprezentace a implicitní plochy.

V *CSG* nebo-li konstruktivní geometrii se 3D model skládá z geometrických primitiv pomocí transformací a logických operací. K popisu stavby modelu se používá stromová struktura. Konstruktivní geometrie nalezne široké uplatnění, např. při modelování různých mechanických součástek. Zobrazení modelu však není triviální a obvykle se převádí na hraniční reprezentaci. Ve spojení s CSG může být využito *šablonování*. Modely se vytvářejí pohybem 2D profilu v prostoru. Profil se buďto pohybuje po obecné křivce nebo rotuje kolem libovolné osy. Díky šablonování můžeme jednoduše vytvářet jednoduchá tělesa, která by se jiným způsobem modelovala mnohem složitěji.

Další variantou jsou *dekompoziční modely*. Jedná se o diskrétní popis objemu tělesa. Těleso je rozděleno 3D mřížkou a nositelem informace jsou její jednotlivé objemové jednotky, tzv. *voxely*[1]. Podobně jako pixely[2] na monitoru, akorát ve 3D. Mřížka může být uložena v podobě sekvenčního pole, což vede k velké paměťové náročnosti. Obvykle se proto používá oktalový strom *(angl. octree)*, který dělí objem postupně a to jen v případě, že se daná oblast liší svými vlastnostmi (odstraňuje se tak redundance[3]). Dekompoziční modely jsou využitelné pro uložení naměřených 3D dat (např. v geologii, počítačové tomografii, magnetické rezonanci apod.). Pro zobrazení se obvykle také převádějí na hraniční reprezentaci.

Velice zajímavým modelovacím způsobem jsou *implicitní plochy*. Původně se používaly pro fyzikální modelování elementárních částic, mohou být ale využitelné i pro modelování reálných těles. Model tělesa je popsán určitým potenciálním polem kolem modelové kostry. Povrch se nachází v oblasti, kde má toto pole nulovou hodnotu. Velkou výhodou takto modelovaných těles je tzv. *morphing* – mohou být plynule transformovány na jiný tvar.

Všechny předchozí modely dnes nalézají uplatnění ve specifických aplikacích. Pro zobrazení se vesměs vždy převádějí na *hraniční reprezentaci*. Hraniční reprezentace popisuje povrch tělesa, ale neukládá informace o jeho vnitřku. Nejjednodušší způsobem je *drátový model* (popis pomocí vrcholů a hran), ten však může sloužit pouze pro rychlé orientační zobrazení, protože neposkytuje úplnou informaci o tvaru tělesa. Pro úplný popis je nutné použít buďto polygonální nebo spline model.

---

[1] Voxel je 3D ekvivalent pro 2D pixel.

[2] 2D obrazový bod.

[3] Opakování stejných dat.

Polygonální model používá pro popis povrchu tělesa *síť vrcholů, hran* a *stěn* (polygony, trojúhelníky nebo quadrilaterály). Není to zcela přesný popis, nýbrž lineární aproximace. Je nutné hlídat regulérnost sítě – model musí být *manifold*[1]. Polygonální sítě mohou být zobrazeny v reálném čase, přičemž zobrazení bývá hardwarově akcelerováno. Je relativné jednoduché modely stínovat, nanášet textury, vytvářet různé efekty apod. V současnosti se jedná asi o nejběžnější způsob zobrazení 3D dat na grafický výstup.

Spline plochy mají hodně blízko k polygonálnímu modelu. Povrch tělesa popisují podobně pomocí *sítě vrcholů, hran* a *stěn*, narozdíl od polygonálního modelu se však jedná o matematický popis. Přesnost modelu se blíží skutečnosti mnohem více a je dána přesností aproximace. Pro zobrazení se spline plochy opět převádějí na polygonální model, popř. se dají využít další metody jako např. ray-casting. Definicí spline ploch existuje nepřeberné množství a obvykle vycházejí z popisu 2D křivek. Příkladem mohou být *bikubické plochy* ze skupiny interpolačních metod nebo *beziérovy, NURBS a T-Spline plochy*, které povrch aproximují. Bikubické plochy dělí plochu na jednotlivé pláty, přičemž je nutné hlídat jejich spojitost. NURBS patří k těm způsobům, které popisují plochu jako celek.

V současnosti existují různé způsoby získání 3D reprezentace reálného tělesa (3D skenování, počítačová tomografie, magnetická rezonance, geologické údaje apod.). Ne vždy má taková reprezentace požadované vlastnosti. Může to být způsobeno metodou získání, špatným nastavením, převodem z jiného modelu apod.

Tato práce se konkrétně zabývá transformací nestrukturovaných trojúhelníkových 3D sítí na vhodnější reprezentace. Problémem takových sítí je malá možnost dále s nimi pracovat. Jejich úprava v modelovacích nástrojích není snadná a převod na ekvivalentní model také není triviální záležitostí. V této práci vysvětlím, jak takovou síť převést na *quadrilaterální síť*[2] a jaké možnosti nám tento převod poskytne. Popíšu návrh a implementaci aplikace demonstrující řešení tohoto problému a uvedu varianty použitelné pro další rozvíjení.

1.1 Struktura práce

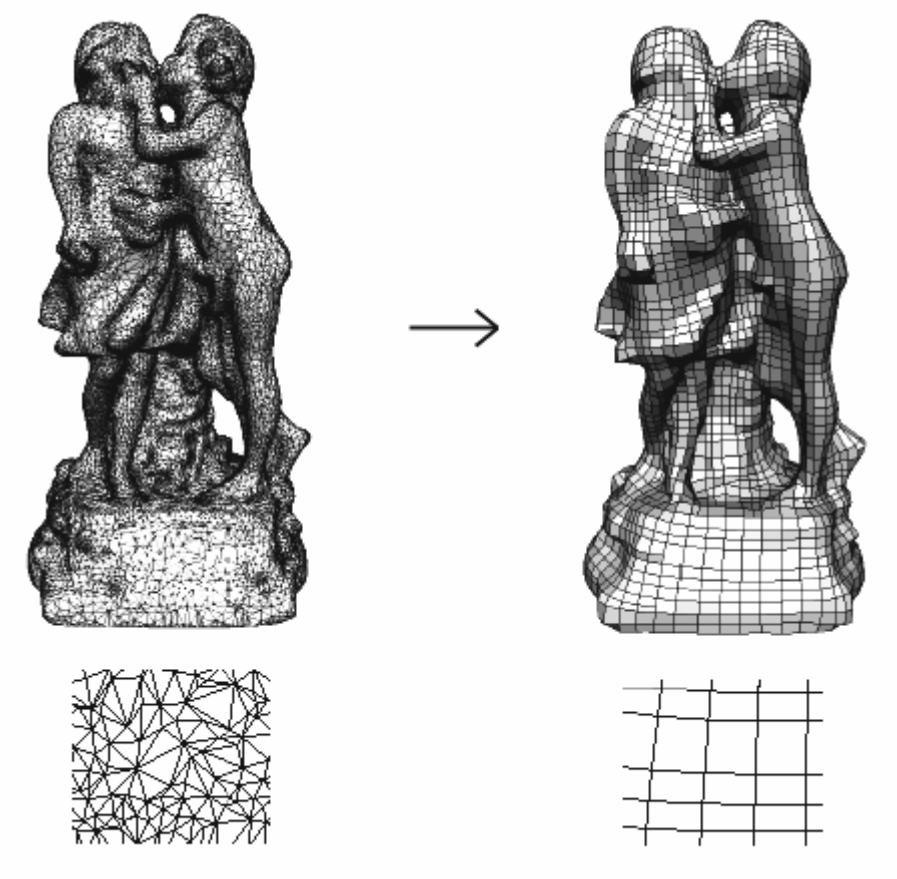
V kapitole 2 nastíním problematiku nestrukturovaných trojúhelníkových 3D sítí, klasifikuji použitelné metody pro řešení problému a navrhnu nejvhodnější kandidáty. Následně si jednu z těchto metod vyberu a podrobně ji rozeberu. V kapitole 3 navrhnu aplikaci pro implementaci vybrané metody. Popis samotné implementace a konkrétní výběr algoritmů a datových struktur je vysvětlen v kapitole 3. V kapitole 5 popíšu výslednou aplikaci a ukážu obrázky, které budou demonstrovat její funkčnost. V závěrečné kapitole 6 celkově shrnu problematiku, můj přínos a přínos pro mě.

[1] Vyrobitelné těleso. Každou hranu sítě sdílí právě dvě stěny.

[2] Síť složená z quadrilaterálů (tj. čtyřúhelníkových plošek).

2 Teoretický rozbor

Jak již bylo zmíněno v úvodu, existují nestrukturované trojúhelníkové 3D sítě (dále jen nestrukturované sítě), které není možné pro svoji složitost dále upravovat. Nejvýraznějším problémem takových sítí je chaotičnost rozložení trojúhelníků, tj. v okolí každého vrcholu se může vyskytovat libovolný počet různě velkých trojúhelníků. Dalším problémem může být příliš velké množství trojúhelníků, větší než je nutné k uchování požadované informace. Pro další zpracování takových sítí je vhodné mít možnost převést je na vhodnější reprezentaci. Takovou reprezentací může být síť složená z *quadrilaterálů*. Přestože quadrilaterální síť je možné chápat jako síť složenou z trojúhelníků (každý quadrilaterál lze opět rozdělit na dva trojúhelníky), trojúhelníky v této síti jsou již dobře strukturovány. Vrcholy jednotlivých *quadrilaterálů* leží v souběžných linkách (křivkách), které mají určitý počátek a konec, resp. mohou tvořit uzavřenou smyčku. Již na první pohled na takovou síť je vidět určitý řád, viz obrázek 2-1.



Obrázek 2-1 : *Převzato z [Garland et al. 2005] a upraveno. Vlevo je vidět nestrukturovaná trojúhelníková 3D síť, vpravo reprezentace stejného tělesa, tentokrát složená převážně z quadrilaterálů. Je vidět, že síť vpravo, bude mnohem vhodnější pro další zpracování.*

2.1 Metody

Pro převod *nestrukturované sítě* na *síť quadrilaterální* existují různé přístupy s různým uplatněním. Prvním způsobem může být *vyhlazování (angl. smoothing).* Primárním cílem vyhlazování je odstranit ze sítě nadbytečné fragmenty vzniklé jako šum nebo nepodstatné pro další uchování. Jak lze z názvu poznat, výsledná síť získá oblé tvary a ztrácí tak některé charakteristické vlastnosti. Z tohoto důvodu vyhlazování nalezne své uplatnění někde jinde. Dalším přístupem je *zjednodušování (angl. simplification).* Zjednodušování v podstatě dělá přesně to, co potřebujeme. Z nestrukturované sítě získáme síť strukturovanou quadrilaterální. Zjednodušování má však stejně jako vyhlazování velkou nevýhodu. Neumožňuje cíleně uchovat detaily. Jako nejlepší alternativa se jeví *přesíťování (angl. remeshing)*. Svým způsobem se podobá zjednodušování, s tím rozdílem, že bere zřetel na drobné prvky či ostré hrany, které se v síti nacházejí.

Významným faktorem metod pro převod sítě, stejně jako asi většiny počítačových algoritmů, je paměťová a výpočetní náročnost. Pokud budeme mít k dispozici metodu s ideálními výsledky, která bude vyžadovat několikahodinové výpočty nebo bude mít gigabajtové paměťové nároky, těžko bude využitelná v běžných aplikacích. Přesto musíme počítat s tím, že převod sítě s miliony vrcholů náš hardware trochu potrápí i v případě použití solidní metody.

Pro podrobnější rozbor a následnou implementaci jsem zvažoval jednu ze dvou metod, které stručně popíšu v následujících bodech této kapitoly.

2.1.1 Harmonic Functions for Quadrilateral Remeshing of Arbitrary Manifolds

Tuto přesíťovací metodu vyvinul tým v čele s M. Garlandem. Informace jsem čerpal z dokumentu [Garland et al. 2005]. Český název zní *Harmonické funkce pro quadrilaterální přesíťování libovolného manifoldu*. Dále v textu ji zkráceně označuji jako HFQRAM.

Tato metoda využívá konstrukce *hladkého harmonického skalárního pole* nad vstupní sítí. Pro toto pole se následně počítají dva *vektorové toky*, *gradientní* a *isoparametrický*, vzájemně ortogonální ve všech bodech. Následně se oběma vektorovými toky *trasují (angl. tracing) integrální linky*. Na průsečících těchto linek vznikají vrcholy nové sítě, které zároveň leží na povrchu původní sítě. Triangulací vzniklých vrcholů se vytvoří *quadrilaterální* síť, která s jistou odchylkou reprezentuje stejné těleso jako síť původní.

Metoda umožňuje vytvářet jak *isotropní* tak *anisotropní sítě*[1]. Vzdálenost linek a tedy velikost výsledných plošek lze korigovat pomocí *vzorkovacích distančních funkcí* s uživatelsky definovanými parametry (počáteční vzdálenost linek a stupeň *anisotropie*). Kromě toho může být uživateli

[1] Velikost dílčích plošek anisotropní sítě závisí na lokální křivosti na rozdíl od sítě isotropní.

umožněno definovat libovolné množství *omezujících vrcholů,* viz kapitola 2.2.3 a tím ovlivnit tvar výsledné sítě.

2.1.2 Anisotropic Polygonal Remeshing

Druhá přesíťovací metoda jejíž použití jsem zvažoval. Vyvinul ji tým vedený P. Alliezem, viz [Alliez et al. 2003]. Její název se do češtiny překládá jako *Anisotropní polygonální přesíťování*. Dále v textu se na ni odkazuji jako na metodu APR.

Tato metoda se částečně podobá metodě HFQRAM. Na rozdíl od ní se zde nepočítá *skalární pole*, ale rovnou se přistupuje ke konstrukci *tensorového pole*[1]. Toto pole se nejdříve předběžně odhaduje na základě určitého okolí každého vrcholu a určí se jeho hlavní směrové toky. Následně se iterativně vyhlazuje. V další fázi se trasují křivostní linky, jejichž hustota závisí na lokální křivostí (díky tomu je možné dosáhnout *anisotropie*). Na průsečících sítě těchto linek vnikají vrcholy nové sítě. Např. pomocí algoritmu CDT *(constrained Delaunay triangulation)* se nad těmito vrcholy vytvoří nová síť, složená převážně z quadrilaterálů.

Tvar výsledné sítě lze ovlivnit definováním kostry vlastností, které chceme uchovat. Hladkost tensorového pole, tudíž kvalitu výsledné sítě, lze ovlivnit dvěma způsoby. Definováním velikosti okolí vrcholů pro počáteční odhad tensorů a nastavením počtu iterací pro vyhlazování pole.

2.1.3 Výběr metody

Obě metody mají společné rysy, ale také své význačné charakteristiky. Obě nabízejí elegantní koncepci k řešení daného problému. Metoda APR dosahuje kvalitativně lepších výsledků. Toho je nejspíš dosaženo iterativním vyhlazováním tensorového pole za cenu vyšší výpočetní náročnosti. Výpočetní náročnost jsem však neměl možnost jak porovnat. V kapitole 9.2 dokumentu [Garland et al. 2005] je uvedeno, že podobný model ruky umožňuje jeho metoda převést za 20s, zatímco metoda APR za 60s. Pro směrodatné porovnání výkonnosti by však bylo třeba mít k dispozici implementace obou metod a podrobit je podrobným testům. Výsledky mohou být ovlivněny různými aspekty, nejen počtem vrcholů a hran v síti, a mohou se také lišit podle implementace. Podle údajů, které mám k dispozici, se rychlost reálných implementací obou metod řádově shoduje. Převod jednoduchých sítí s několika tisíci vrcholy trvá okamžik, větších sítí se stovkami tisíc vrcholů několik minut. Na základě předem dostupných informací jsem se rozhodl podrobněji rozebrat a implementovat metodu HFQRAM.

[1] Tensorové pole je matematické zobecnění pole vektorového.

2.2 Popis vybrané metody

Vybraná metoda HFQRAM pracuje s *manifold sítěmi libovolných tříd (angl. genus)*, je zcela obecná, efektivní a flexibilní. To znamená, že si poradí se sítí jakékoliv topologie a vždy podá uspokojivý výsledek v relativně krátkém čase. Převádí nestrukturovanou síť na strukturovanou quadrilaterální, přičemž umožňuje kontrolovat hustotu linek a tvar výsledné sítě. Převod probíhá v několika krocích. Každý krok funguje jako oddělená operace s konkrétním výsledkem, závislá na výsledcích předchozích kroků. Nic nebrání tomu, aby byl převod znovu spuštěn od libovolného kroku s upravenými parametry. Díky tomu můžeme opakovat některou fázi a dolaďovat výslednou síť.

V následujících bodech této kapitoly vysvětlím princip této metody.

2.2.1 Algoritmus

Uvádím přehled celého algoritmu, částečně převzatý z [Garland et al. 2005]. Předpokládejme, že máme danou vstupní trojúhelníkovou manifold síť $M = (V, F)$, složenou z množiny vrcholů V a množiny trojúhelníků F. Každému vrcholu i je přiřazena souřadnice $x_i \in R^3$ ve 3D euklidovském prostoru. Požadujeme, aby byla síť *manifold*, jinak může být libovolných druhů a nepředpokládáme žádná omezení co se počtu hraničních křivek týče. Na nejvyšší úrovni se tento přesíťovací algoritmus skládá z následujících základních kroků:

1. Výpočet po částech lineárního skalárního pole $u : V \to R$ nad vrcholy M.
2. Na základě u odvození dvou ortogonálních po částech konstantních tangentních vektorových polí $g1, g2 : F \to R^3$ nad ploškami *(angl. faces)* M.
3. Sestavení sítě polygonů nad povrchem trasováním integrálních linek vektorových polí g_1 a g_2.
4. Eliminace tzv. T-zakončení.

Výsledkem je nesouhlasná *(angl. non uniform)* převážně quadrilaterální síť pokrývající vstupní manifold M.

2.2.2 Skalární pole

Prvním krokem je konstrukce *harmonického skalárního pole*. Toto pole reprezentuje jakýsi tok podél povrchu sítě, které konverguje k předem definovaným extrémům. Tyto extrémy nejsou nic jiného než vrcholy, kterým je přiřazena nenulová hodnota skalárního pole. Mohou být také nazývány jako omezující vrcholy. Ostatní vrcholy mají nulovou hodnotu, která je během výpočtu nahrazena výslednou hodnotou skalárního pole. Hodnota extrémů zůstává výpočtem nezměněna. Jediným omezením, co se počtu omezujících vrcholů týče, je $C \subset V$. Pro relevantní výsledek je však nutné

definovat alespoň dva. Vhodným rozmístěním extrémů se upravuje tvar skalárního pole, tudíž i výsledné sítě, což bude podrobněji rozebráno v kapitole 2.2.3.

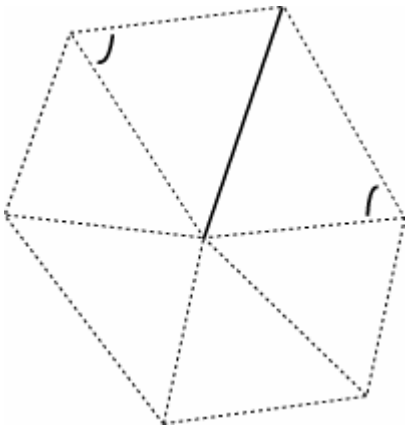
Z matematického hlediska je skalární pole definováno *harmonickou funkcí* [Wikipedia 2007a], která přiřazuje každému vrcholu sítě konkrétní skalární hodnotu ($u : V \to R$). Základním požadavkem takové harmonické funkce je dodržení Laplaceho rovnosti $\Delta u = 0$ [Wikipedia 2007b]. Skalární pole je spojité, tato metoda však předkládá způsob, jak toto pole řešit diskrétně. Laplaceho rovnost se převede do tvaru $\Delta u = -L\vec{u}$, kde $\vec{u}$ představuje vektor hodnot skalárního pole odpovídajících jednotlivým vrcholům sítě a L je následující matice:

$$L_{ij} = \begin{cases} \Sigma_{<i,k>\in M} w_{ik} & i = j \\ -w_{ij} & <i,j> \in M \\ 0 & jinak \end{cases}. \qquad (1)$$

Váhu hrany w_{ij} lze definovat různými způsoby, např. kombinatoricky v závislosti na počtu hran vedoucích k danému vrcholu, propagováním střední hodnoty nebo váženou hodnotou, viz [Garland et al. 2004]. V mojí implementaci jsem se řídil dokumentem [Garland et al. 2005] a použil vztah:

$$w_{ij} = -\tfrac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right), \qquad (2)$$

kde α_{ij} a β_{ij} jsou protilehlé úhly oproti hraně $v_i v_j$.



Obrázek 2-2 : *Výřez sítě – příslušné úhly pro určení váhy zvýrazněné hrany.*

Řešením lineární soustavy 2-1 však nezískáme potřebný výsledek. Do soustavy potřebujeme vnést extrémy, ke kterým pole konverguje, tak aby bylo následně možné určit vektorové pole. Požadovanou soustavu tedy definujeme takto:

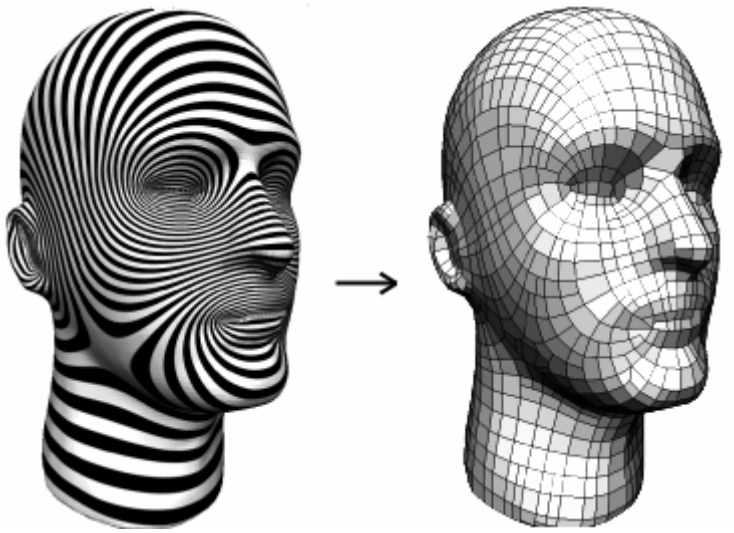
$$A\vec{u} = \vec{b} \qquad A_{ij} = \begin{cases} \delta_{ij} & i \in C \\ L_{ij} & jinak \end{cases} \qquad b_i = \begin{cases} c_i & i \in C \\ 0 & jinak \end{cases}. \qquad (3)$$

V tomto okamžiku máme lineární soustavu pro výpočet skalárního pole. Kroneckerova delta funkce δ_{ij}, viz [Wolfram 2002], zařídí, že hodnoty všech extrémů zůstanou výpočtem nezměněny. Výsledné pole k těmto extrémům konverguje.

Výhoda tohoto postupu spočívá v tom, že není potřeba síť parametrizovat. Díky tomu může být metoda zcela obecná pro všechny manifold sítě a navíc se vyhneme komplikacím se složitými integracemi. Skalární pole představuje *harmonickou, po částech lineární funkci*. Hodnoty ve vrcholech se dají spočítat diskrétně, řešením lineární soustavy 2-3, v ostatních bodech potom lineární interpolací hodnot vrcholů odpovídajícího trojúhelníka.

2.2.3 Rozmisťování extrémů

Vhodné rozmístění extrémů záleží na konkrétní topologii sítě. Skalární pole lze spočítat pro každý manifold, je ale nutné přizpůsobit jeho tvar tak, aby jeho tok byl co nejhladší, aby ve výsledné síti nevznikaly různé deformace. Obvykle se extrémy umisťují na *výčnělky* do bodu s největší křivostí. Některé sítě ale nemají žádné výčnělky, např. *geometrická primitiva* jako koule, anuloid apod. Někdy stačí umístit maximum na jednom konci sítě a minimum na protilehlém. V síti se ale mohou vyskytovat prvky, které se takovými volbami degradují. Tyto situace sice nejsou příliš časté, ale musí se s nimi počítat. Jejich řešení spočívá v určení skupiny extrémů seřazených v řadě, která může tvořit křivku nebo uzavřenou smyčku.



Obrázek 2-3 : *Obrázek je převzatý z [Garland et al. 2004]. Zobrazuje vhodnou volbu extrémů na modelu hlavy. Jsou definovány skupiny extrémů, v oblasti očí minima, v oblasti úst a uší maxima.*

U většiny sítí může člověk tvar skalárního pole relativně dobře přizpůsobit intuitivním způsobem. Navíc někdy je potřebné nějakým způsobem proces přesíťování ovlivnit, pokud není výsledek zcela podle představy. Na druhou stranu nevhodná volba hodnot extrémů může vést k lokálním deformacím pole a následnému zborcení *(angl. warping)* částí výsledné sítě. Možnou alternativou je poloautomatické rozmístění extrémů. Uživatel určí několik málo extrémů na jejichž základě se

spočítá počáteční *neharmonické pole*, z něhož se následně určí nové extrémy. Ve druhé fázi se již spočítá výsledné pole. Během tohoto postupu mohou vznikat nevhodné skupiny lokálních extrémů v oblastech s malou křivostí. To lze ale jednoduše vyřešit shlukováním *(angl. clustering)* blízkých extrémů a výběrem toho nejdůležitějšího.

2.2.4 Kritické body

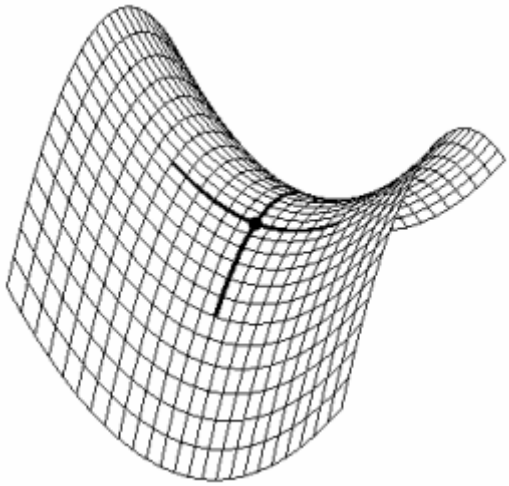
Eulerova charakteristika popisuje třídu manifoldu v závislosti na počtu jeho vrcholů, hran a plošek *(angl. vertices, edges, faces)*:

$$\chi = 2 - 2g = |V| - |E| + |F| . \quad (4)$$

Pomocí Morseho teorie lze z Eulerovy charakteristiky odvodit vztah popisující počet kritických bodů, tj. počet minim, maxim a tzv. *sedlových bodů (angl. saddle points)* skalárního pole:

$$\chi = \eta_{\min} + \eta_{\max} - \eta_{saddle} . \quad (5)$$

Sedlové body jsou místa, ve kterých se první i druhá derivace funkce libovolného počtu proměnných rovná nule. To znamená, že to jsou zároveň stacionární i inflexní body, ale nejedná se o extrémy. Neformálně řečeno, křivost funkce tady v některých směrech roste do kladných hodnot, zatímco v jiných směrech klesá do záporných, viz [Wikipedia 2006] a obrázek 2-4. Z rovnice 2-5 vyplívá, že čím více extrémů uživatel definuje, tím bude více sedlových bodů. Počet kritických bodů je vhodné omezit na minimum. Jejich přítomnosti se však neubráníme, proto na ně musí být brán zvláštní zřetel při trasování plovoucích linek a vytváření nové sítě, viz kapitoly 2.2.6 a 2.2.11.



Obrázek 2-4 : *Sedlový bod na povrchu funkce dvou proměnných.*

2.2.5 Vektorové pole

Před tím, než se dostaneme k trasování tzv. *plovoucích linek gradientního* a *isoparametrického toku*, potřebujeme určit vektory skalárního pole pro každý trojúhelník sítě. Gradientní a isoparametrický tok jsou vzájemně *ortogonální*, tedy i vektory těchto toků pro libovolný trojúhelník jsou ortogonální.

Gradientní vektor pro trojúhelník (i, j, k), jehož vrcholy mají souřadnice $x_i, x_j, x_k \in R^3$ a $\vec{n}$ je jeho normálový vektor, určíme jako $g_1 = \nabla u$ řešením lineární soustavy:

$$\begin{bmatrix} x_j - x_i \\ x_k - x_j \\ \vec{n} \end{bmatrix} \begin{bmatrix} \\ \vec{g}_1 \\ \\ \end{bmatrix} = \begin{bmatrix} u_j - u_i \\ u_k - u_j \\ 0 \end{bmatrix}. \quad (6)$$

Protože *isoparametrický vektor* je kolmý na gradientní i normálový vektor, určí se následovně:

$$\vec{g}_2 = \vec{n} \times \vec{g}_1. \quad (7)$$

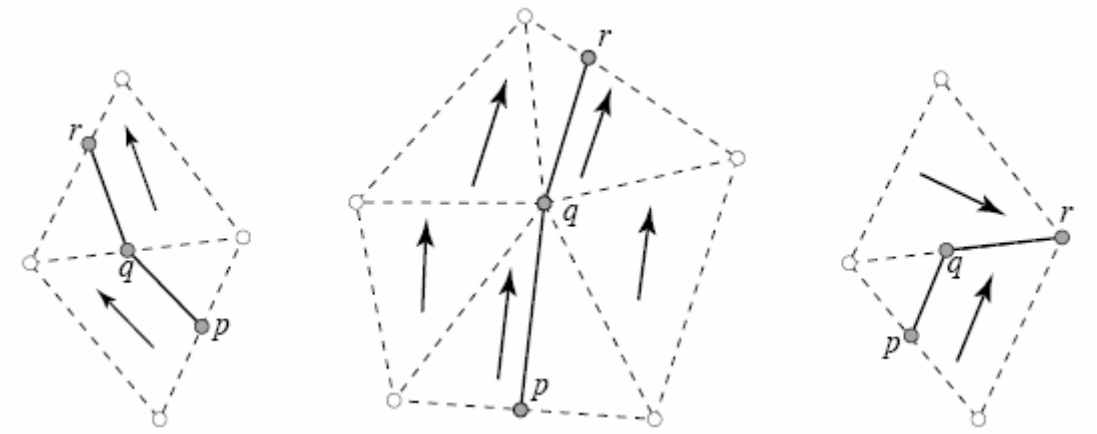
2.2.6 Trasování plovoucích linek

Potom co máme spočítáno skalární pole a určen jeho gradientní a isoparametrický vektorový tok (máme definovány vektory těchto toků pro každý trojúhelník sítě), dostáváme se k trasování *plovoucích linek*. Plovoucí linka je diskrétním ekvivalentem k *integrální lince skalárního pole*. Pro její výpočet však není nutné parametrizovat povrch sítě, čímž se vyhneme náročným integracím.

2.2.7 Trasování samostatné linky

Plovoucí linka je po částech lineární křivka, která kopíruje síť podél jednoho z vektorových toků. Skládá se z přímočarých segmentů, které jsou vzájemně napojeny v bodech, ležících na hranách sítě. Počáteční a koncový bod každého segmentu leží na hranách téhož trojúhelníka, vyjma počátečního bodu plovoucí linky, který může ležet uvnitř trojúhelníka. Směrový vektor segmentu je určen aktuálním vektorem příslušného toku, tedy jedním z vektorů příslušného trojúhelníka vypočítaných v kapitole 2.2.5. Při určování libovolného segmentu plovoucí linky mohou nastat tři případy:

1. Nejjednodušší situace nastává, když se od jednoho průsečíku na hraně pokračuje k dalšímu průsečíku na jiné hraně téhož trojúhelníka, viz obrázek 2-5A.
2. Situace se komplikuje, pokud linka prochází přes vrchol. Potom má potenciálně na výběr z několika možných cest. Pokud se jedná o gradientní tok, musí se vyzkoušet všechny varianty a vybírá se z nich ta, ve které se postoupí nejdál, viz obrázek 2-5B.
3. Poslední varianta nastává, když není možné najít průsečík s jednou ze dvou dalších hran příslušného trojúhelníka, protože vektorový tok konverguje k výchozí hraně. Nezbývá než vybrat jeden z vrcholů hrany ve směru konvergence, viz obrázek 2-5C.



(A) *(B)* *(C)*

Obrázek 2-5 : *Tento obrázek je převzatý z [Garland et al. 2005]. Ukazuje tři možné varianty průchodu plovoucí linky přes trojúhelník sítě. A) normální průchod, B) plovoucí linka prochází přes vrchol, C) linka se zalomí podél hrany, ke které konverguje vektorový tok.*

Trasování gradientních a isoparametrických linek je velice podobné. Liší se pouze v tom, že gradientní linky mají počátek a konec, zatímco isoparametrické linky vesměs vždy tvoří uzavřené smyčky. Dalo by se říct, že gradientní linky sledují tok skalárního pole, zatímco isoparametrické tvoří jeho vrstevnice.

Počátek trasování jednotlivých linek určují tzv. *semena (angl. seeds)*. To bude rozebráno v kapitole 2.2.9. Konec gradientních linek lze určit tím, že dosáhly opačného extrému skalárního pole, než ve kterém začínaly (nelze již dál pokračovat). To ale nelze aplikovat na linky isoparametrického toku. Kromě toho je potřeba kontrolovat vzdálenost linek stejného toku, aby byla síť vhodně pokryta. Obecným způsobem, jak řešit tuto situaci, je vzorkovací distanční funkce, viz kapitola 2.2.8.

2.2.8 Vzorkovací distanční funkce

V kapitole 2.2.7 byly vysvětleny důvody, pro kontrolování vzdálenosti linek téhož vektorového toku. V případě generování isotropní sítě vystačíme se vzorkovací distanční funkcí h, která bývá obvykle konstantní. Pokud chceme generovat anisotropní síť, použijeme distanční funkce, h_1 pro gradientní vektorový tok, h_2 pro isoparametrický:

$$h_1 = \frac{h}{1+\alpha \log_{10}\left(1+\kappa_n^2\right)}, \qquad h_2 = \frac{h}{1+\alpha \log_{10}\left(1+\kappa_n^1\right)}. \qquad (8)$$

κ^1 resp. κ^2 je normálová křivost povrchu ve směru g_1 resp. g_2. Hodnota α bývá obvykle menší než 20, ale může být i mnohem větší. Pokud se $\alpha = 0$, je výsledná síť isotropní ($h_1 = h_2 = h$).

2.2.9 Rozmisťování plovoucích linek

Jak bylo zmíněno dříve, počátek linky je určen tzv. *semenem*. Ukončení linky zajišťuje vzorkovací distanční funkce. Tedy pokud je vzdálenost mezi dvěma linkami menší, tj. vzdálenost počítaná podél ortogonálního toku, než hodnota τh_1 resp. τh_2, je linka ukončena. Hodnota τ bývá obvykle 0.5, ale může být ponechána jako uživatelsky nastavitelný parametr. Otázka zní, jak správně rozmístit semena?

Nejdříve umístíme počáteční *semena* do každého *extrému* a na *roh*[1] každé *vlastnosti*[2] (viz kapitola 2.2.10). V místech tzv. *sedlových bodů* (viz kapitola 2.2.4) může docházet k problémům při trasování linek, proto je také z obou stran obou vektorových toků ohraničíme semeny (celkem 4 semena). Po dokončení trasování každé linky se vygenerují další semena v pravidelných intervalech podél této plovoucí linky, po jejích obou stranách. Semena pro každý vektorový tok se ukládají do samostatné prioritní fronty. Nejvyšší prioritu mají ta semena, která jsou nejdále od plovoucí linky téhož toku. Důvodem je snaha trasovat co nejdelší linky. Pokud leží semeno blízko plovoucí linky, nová linka bude pravděpodobně ukončena dříve, než když semeno leží dál.

2.2.10 Vlastnosti

V sítích bývají často různé ostré prvky, které je potřeba zachovat. Nazýváme je *vlastnosti*. Množinu sousedících hran dané vlastnosti nazýváme *řetězy (angl. chains)*. Pokud z některého vrcholu dané vlastnosti vycházejí nejméně tři hrany, říkáme mu *roh (angl. corner) vlastnosti*. Pro správné pokrytí vlastností plovoucími linkami se musí každý *roh* umístit mezi počáteční semena (viz kapitola 2.2.9). Pro detekci hran doporučuje tato metoda *značkování hran klínových úhlů*, odkazuje se ale také na alternativní způsoby jako např. [Watanabe and Belyaev 2001; Page et al. 2002]. Takto označkované hrany se musí přidat do výsledné sítě, viz kapitola 2.2.11.

Toto je okamžik, ve kterém se tato metoda odlišuje od metod zjednodušovacích a řadí se k metodám přesíťovacím. Právě tímto postupem se zachovají klíčové detaily.

2.2.11 Konstrukce nové sítě

Potom co určíme síť plovoucích linek gradientního a isoparametrického toku, přistoupíme ke konstrukci výsledné sítě. Její vrcholy leží na průsečících sítě plovoucích linek, z nich se tvoří nové hrany a polygony.

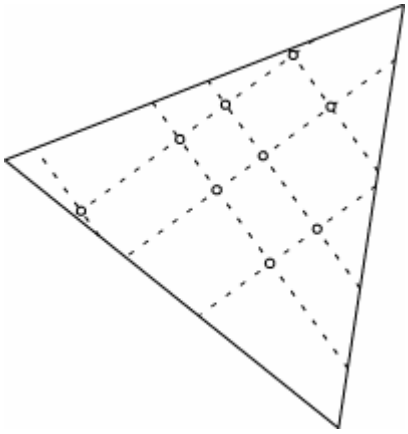
---

[1] Vrchol vlastnosti, ve kterém se zbíhají alespoň 3 hrany této vlastnosti.

[2] Oblast sítě s velkou křivostí, kterou je potřeba uchovat.

2.2.12 Určení vrcholů

Segmenty plovoucích linek uvnitř každého trojúhelníka původní sítě tvoří *rovinnou pravoúhlou mřížku*. Vrcholy se potom určí jednoduchým výpočtem průsečíků uvnitř každého trojúhelníka zvlášť. Do množiny křížení nezahrnujeme kritické body, viz kapitola 2.2.4. Naopak přidáme vrcholy hran, které byly označeny jako vlastnost, viz kapitola 2.2.10, přičemž není podmínkou, aby tyto vrcholy ležely na průsečících plovoucích linek. Každý průsečík má obvykle 4 sousední, 2 podél gradientního a 2 podél isoparametrického toku. Pokud má pouze jednoho souseda, je vyřazen. Průsečíky blízké k průsečíku vlastnosti se také eliminují, sloučením s průsečíkem této vlastnosti. Pro správné napojení vrcholů je nutné ukládat si informace o okolních průsečících.



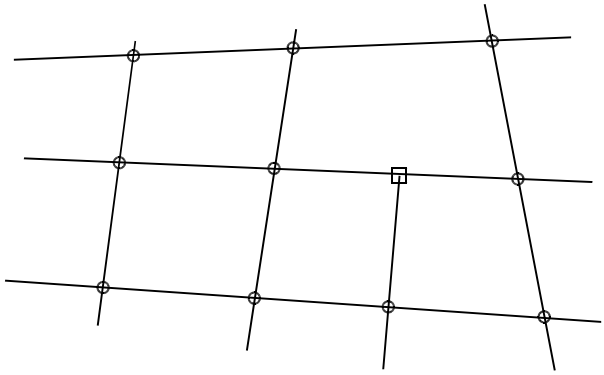
Obrázek 2-6 : *Rovinná mřížka plovoucích linek uvnitř trojúhelníka.*

2.2.13 Vytvoření polygonů

V tomto okamžiku máme vrcholy nové sítě a graf jejich propojení. Jednotlivé polygony získáme pravotočivým kruhovým průchodem každé plošky *(angl. face)* tohoto grafu. Protože jsou extrémní body vyřazeny ze seznamu průsečíků, musíme vytvořit dodatečné polygony v jejich okolí, aby byla síť kompletně uzavřená. K tomuto účelu projdeme vrcholy nejbližší isoparametrické plovoucí linky a z jejich sousledných dvojic a extrémního bodu vytvoříme vějíř trojúhelníků. V tomto okamžiku máme kompletní výslednou síť.

2.2.14 Postprocessing

Ve výsledné síti mohou vznikat tzv. *T-zakončení (angl. T-Junctions).* Jsou to oblasti, ve kterých se plovoucí linky příliš přiblížily a jedna z nich zde končí. V tomto místě vznikají polygony s více jak 4 stranami, což je nežádoucí. Takto defektní polygony se odstraní jednoduchým rozdělením na několik dílčích trojúhelníků.



Obrázek 2-7 : *Ilustrace T-zakončení (čtvereček).*

2.3 3D Spline plochy

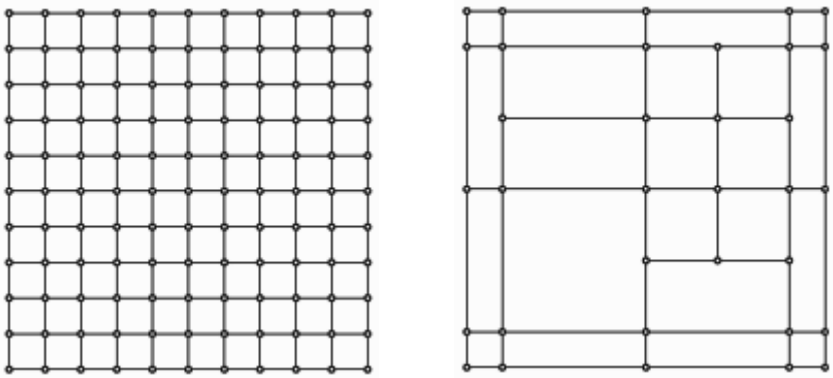
Původní nestrukturovaná trojúhelníková síť neposkytuje příliš možností pro další zpracování. V předchozích bodech této kapitoly jsem popsal převod takové sítě na síť quadrilaterální, kterou je možné dále zpracovávat. V tomto okamžiku by bylo možné ukončit další postup. Quadrilaterální síť je jedna z mnoha dnes používaných reprezentací 3D těles. Jako základ pro další použití postačuje. Pro zpracování ve 3D editorech se dnes ale používají alternativní vyjádření, která poskytují více uživatelské volnosti a komfortu. Asi nejznámější variantou jsou NURBS plochy, viz [Wikipedia 2007d]. Jako nadějný moderní nástupce NURBS se jeví T-Spline plochy, viz [Sederberg et al. 2004].

2.3.1 NURBS plochy

NURBS znamená *non uniform rational B-Spline*, česky *neuniformní racionální B-spline*. Jedná se o matematické vyjádření křivek resp. ploch pomocí *vážených kontrolních bodů* a *uzlového vektoru*. NURBS jsou oblíbené díky svojí flexibilitě. Umožňují lokální změnu tvaru křivky, přidávání kontrolních bodů bez změny tvaru, jsou invariantní vůči lokálním transformacím atd. V současnosti se hojně využívají v modelovacích nástrojích.

2.3.2 T-Spline plochy

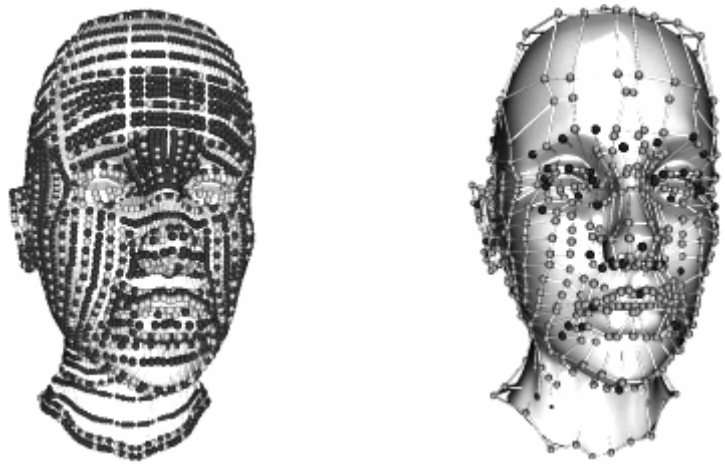
Přes všechny výhody NURBS ploch, najdou se i nedostatky. Síť kontrolních bodů musí tvořit kompletní mřížku. Komplexnější sítě potom potřebují definovat velké množství nadbytečných kontrolních bodů, což může vést k nepřehlednosti. T-Spline plochy jsou zobecněním NURBS, takže mají hodně společného a také existuje jednoduchá konverze. Ve skutečnosti jsou NURBS krajním případem T-Spline, ve kterém je mřížka kontrolních bodů zcela kompletní.



(A) *(B)*

Obrázek 2-8 : *Srovnání mřížky kontrolních bodů (A) NURBS a (B) T-Spline.*

T-Spline redukuje množství kontrolních bodů. T-zakončení, která jsou u NURBS nebo quadrilaterální sítě nežádoucí, zde slouží jako klíčový element. Díky nim mohou být do sítě přidávány další detaily lokálního charakteru, bez nutnosti vytvářet řady nadbytečných kontrolních bodů. T-Spline poskytuje všechny výhody NURBS a přidává další. Jeví se jako moderní alternativa, která dle mého názoru NURBS brzy zcela nahradí.



(A) *(B)*

Obrázek 2-9 : *Obrázek je převzatý z [Sederberg et al. 2004]. Srovnání počtu kontrolních bodů modelu reprezentovaného pomocí (A) NURBS, (B) T-Spline.*

3 Návrh

Jednotlivé kroky, které je nutné provést ke zdárnému přesíťování zdrojové nestrukturované sítě na quadrilaterální síť byly specifikovány v teoretickém základu v kapitole 2.2 a jsou neměnné. V této kapitole popíšu podklady pro implementaci funkční aplikace. Mohou zde být jisté odlišnosti v použitých algoritmech oproti metodě HFQRAM, ta totiž není striktní a ponechává dost tvůrčího prostoru.

3.1 Přípravná fáze

Na počátku se načte vstupní síť ze zdrojového souboru. Data mohou být uložena např. v hojně používaném formátu STL (*Standard Tessellation Language;* viz [Wikipedia 2007c]). Pro výpočet skalárního pole je potřeba sestavit vektor omezení, viz rovnice 2-3. Aby bylo možné jeho jednotlivé hodnoty asociovat s vrcholy, je nutné pro vrcholy vygenerovat indexy – lineární posloupnost s počátkem v nule. Vrcholy nastavené jako extrém mají nenulovou hodnotu, ostatní nulovou.

3.2 Skalární pole

Pro výpočet skalárního pole je nutné sestavit a vyřešit rozsáhlou matici M. Musí se počítat s variantou, že se bude řešit síť s miliony vrcholy. Lze využít toho, že je matice řídká, a není tedy nutné ukládat všechny hodnoty. Pro řešení je vhodné použít nějakou optimalizovanou metodu. Pro další výpočet je potřeba uchovat výsledné hodnoty skalárního pole. Ty mohou být přiřazeny jednotlivým vrcholům sítě nebo mohou zůstat v poli, které je výsledkem řešení matice, a následně být přístupné přes indexy vrcholů.

3.3 Vektorové pole

Po určení skalárního pole je nutné stanovit jeho gradientní a isoparametrické vektory v každém trojúhelníku. Lineární soustava pro řešení gradientních vektorů je definovaná zvlášť pro každý trojúhelník. Řešení gradientních vektorů tedy spočívá v iterativním dosazování odpovídajících hodnot do rovnice 2-6. Vektorovým součinem gradientního a normálového vektoru trojúhelníka se určí vektor isoparametrický. Výsledné vektory se musí ukládat tak, aby byly při trasování plovoucích linek jednoduše dostupné pro každý trojúhelník.

3.4 Trasování plovoucích linek

Trasování plovoucích linek je asi nejkomplexnější krok celého procesu. Musí být ošetřeny všechny situace, aby byla síť co nejlépe pokryta, ale aby nevznikaly různé defekty.

Průchod jedné plovoucí linky se jeví jako ukázkový příklad rekurze. Následující uzel linky se určí jako průnik hrany sítě a polopřímky, jejíž počátek leží v aktuálním uzlu a směrový vektor odpovídá vektorovému toku v aktuálním trojúhelníku. Ze získaného uzlu se pokračuje stejným způsobem rekurzivně, dokud není dosažen konec. Protože se ale plovoucí linka může skládat z takřka neomezeného počtu uzlů, taková rekurze by nemusela končit úspěchem. Z tohoto důvodu je lepší řešit problém v cyklu. Každý uzel plovoucí linky musí mít možnost správně určit a vrátit následující uzel v závislosti na své poloze. Polohy uzlu vůči trojúhelníku mohou být následující:

1. *Ve vrcholu* – určuje se průnik s protilehlou hranou všech okolních trojúhelníků.
2. *Na hraně* – určuje se průnik se dvěma zbylými hranami trojúhelníka.
3. *Uvnitř trojúhelníka* – průnik se určuje se všemi hranami trojúhelníka.

Po nalezení každého následujícího uzlu se musí ukládat informace o průchodu linky přes konkrétní trojúhelník. Pro jednoduchost stačí, aby každý trojúhelník obsahoval seznam segmentů linek, které přes něj procházejí.

Vzdálenost linek zajišťuje distanční vzorkovací funkce, viz 2.2.8. Pro zjednodušení implementace stačí, aby plovoucí linka skončila, pokud se přiblíží k jiné na menší vzdálenost, než je uživatelsky definovaná konstanta. Výsledná síť potom bude isotropní. Vzdálenost se určí podél ortogonálního vektorového toku vůči toku aktuální linky a to v kladném i záporném smyslu. Vyhledávání blízkých linek téhož toku se provede trasováním virtuální linky. Pokud není v definované vzdálenosti nalezena jiná plovoucí linka, ukončí se trasování virtuální linky a pokračuje se v trasování vlastní linky.

Po vytvoření každého segmentu se podél něho v pravidelných intervalech určí počáteční body dalších plovoucích linek ortogonálního toku. Počáteční body se vloží do jedné ze dvou front, podle toho, ke kterému toku bod přísluší. Pokud se obě fronty vyprázdní, trasování linek končí.

3.5 Konstrukce nové sítě

Každý trojúhelník obsahuje seznam segmentů, které přes něj procházejí. Jednotlivá křížení plovoucích linek lze vyřešit výpočtem průsečíků segmentů v rámci každého trojúhelníka. Tyto segmenty tvoří rovinnou pravoúhlou mřížku, čímž se výpočet zjednodušuje. Důležitá je datová struktura pro ukládání těchto průsečíků. Musí totiž existovat efektivní způsob pro určení sousedních průsečíků ve všech čtyřech směrech. Toto je nutná podmínka pro efektivní triangulaci sítě. Pro tento účel navrhuji pro každou plovoucí linku vytvořit datovou strukturu, která bude umožňovat efektivně tyto operace:

1. Zařadit prvek na správnou pozici podle vzdálenosti od počátku linky.
2. Iterativně procházet prvky od počátku ke konci.
3. Umožnit určit předchůdce a následníka libovolného prvku.

Každý průsečík leží na gradientní a zároveň na isoparametrické lince, tudíž se bude vkládat do dvou takových struktur.

Jednotlivé polygony sítě se určí z grafu křížení a následně se rozloží na trojúhelníky. Většina polygonů se skládá ze 3 nebo 4 vrcholů, takže jejich rozklad je jednoduchý. Mohou však vznikat i složitější polygony. Pro rozklad konvexních polygonů do velikosti kolem 10 vrcholů lze sestavit sadu šablon, které řeší konkrétní konfigurace. Tento způsob by měl pokrýt asi 95% případů včetně jednoduchých T-zakončení (T-Junctions). Toto však není obecné řešení a složitější polygony není možné řešit šablonovým způsobem. Pro ty je nutné použít nějaký obecný triangulační algoritmus, např. Delaunay triangulation.

4 Implementace

4.1 Vývojové prostředí

Pro implementaci jsem si z nabízených programovacích jazyků vybral C++. Přestože se jazyky jako Java a C# těší veliké oblíbenosti a také bych k nim rád sáhnul, v oblasti počítačové 3D grafiky dle mého názoru stále kraluje C++. K tomu obě knihovny, které používám, jsou psány právě v C++. Existují sice způsoby, jak kód psaný v C++ použít v kombinaci se zmiňovanými pokročilejšími jazyky, tím bychom se však ochudili o velkou výhodu C++, kterou je kompilace do strojového kódu a tím vysoká výkonnost, kterou u 3D grafiky potřebujeme.

Jako implementační prostředí jsem zvolil Microsoft Visual Studio 2003. Přestože tento programovací nástroj funguje ryze pod operačním systémem Microsoft Windows, měl by být můj program plně přenositelný na jiné platformy, protože nevyužívá žádné platformě závislé knihovny.

Pro zobrazování grafických výstupů používám knihovnu OSG (Open Scene Graph; viz http://www.openscenegraph.org). Tato knihovna slouží jako objektově orientovaná nadstavba nad OpenGL. Na jedné straně mi její architektura přijde dost nestandardní a velice komplexní, tudíž je nutné se učit něco úplně nového. Na druhé straně umožňuje řešit komplexní problémy mnohem jednodušeji než v klasickém OpenGL.

Druhou nosnou knihovnou mého projektu je MDSTk (Medical Data Segmentation Toolkit; viz http://www.fit.vutbr.cz/~spanel/mdstk/). Tato knihovna, která je vyvíjena v rámci školy (FIT VUT v Brně), zahrnuje řadu modulů, využitelných pro 2D a 3D grafické aplikace. Pro mě jsou důležité zejména součásti LAPACK a VectorEntity. Knihovna LAPACK slouží k výpočtům rozsáhlých lineárních soustav. Je vysoce optimalizována, původně napsána ve Fortranu77, později zkompilována s rozhraním pro jazyk C. VectorEntity je navrženo pro strukturované ukládání elementů 3D sítě (vrcholy, hrany, trojúhelníky, apod.) a efektivní iterační operace nad těmito elementy. Díky knihovně VectorEntity jsem se mohl odchýlit od některých implementačních postupů navrhovaných v metodě HFQRAM a navrhnout vlastní.

4.2 Implementace vybrané metody

Implementace se hodně blíží navrhovanému postupu podle metody HFQRAM. V kapitole 2.2 byl uveden teoretický základ, v kapitole 3 jsem popsal návrh aplikace, tady se budu věnovat použitým algoritmům a konkrétním datovým strukturám. Konkrétní programová dokumentace je k nalezení na přiloženém CD.

Přehledově uvádím architekturu aplikace. Základem je třída `Mesh`, která rozšiřuje funkčnost třídy z knihovny OSG a díky tomu slouží pro zobrazení a převod trojúhelníkové sítě. Pro uložení

zdrojové sítě obsahuje kontejner trojúhelníků z knihovny VectorEntity. Ke každému trojúhelníku toho kontejneru je připojen objekt `Tri`, který poskytuje prostředky nutné pro převod sítě. Třída `Point` a od ní odvozené slouží jako uzly plovoucích linek. Křížení plovoucích linek představuje třída `Crossing`. `CrossingTable` je tabulka pro ukládání a rychlý přístup k těmto křížením. Třída `PTemplate` umožňuje dělení polygonů na trojúhelníky podle definované šablony. Kompletní hierarchie tříd je vidět na obrázku 4-1. V následujících kapitolách uvádím průřez klíčovými body převodu sítě.

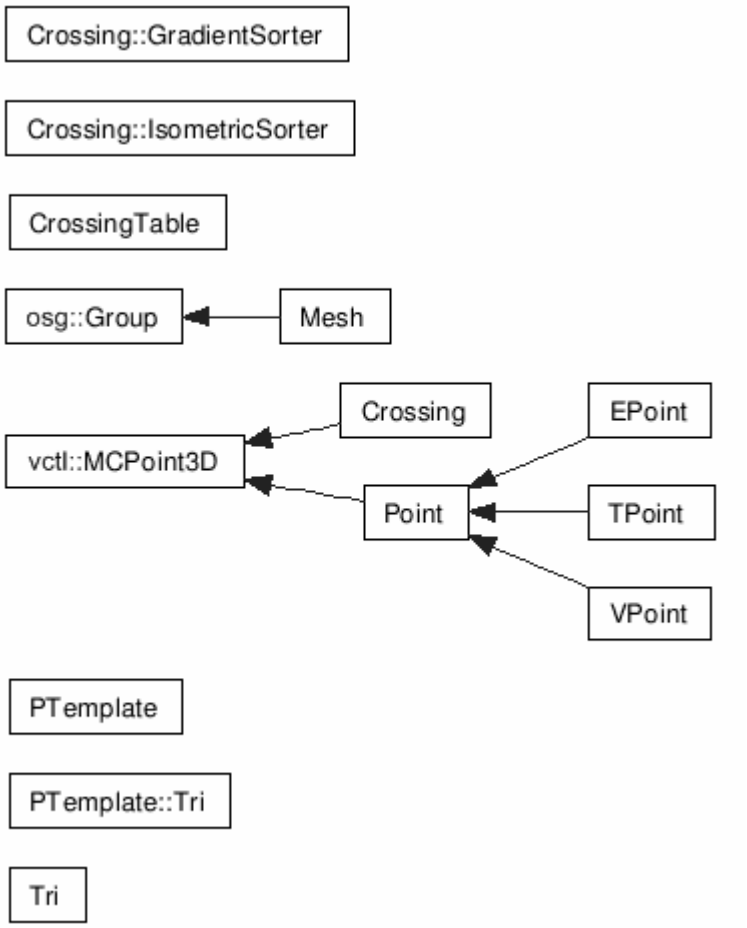


Diagram 4-1 : *Hierarchie tříd.*

4.2.1 Přípravná fáze

Načtení vstupní sítě ze zdrojového souboru ve formátu STL zajišťuje knihovna VectorEntity. Extrémy se určí automatickým způsobem jako nejmenší a největší x-souřadnice. Tuto fázi je nutné dokončit tak, aby mohl uživatel definovat extrémy libovolně. Hodnoty extrémů se uloží do pole omezení na příslušné pozice podle indexů odpovídajících vrcholů.

4.2.2 Skalární pole

Vyřešení matice M zajišťuje výkonná funkce z knihovny LAPACK. Sestavení probíhá ve dvou průchodech sítí. V prvním se spočítají vnitřní úhly každého trojúhelníka, ve druhém se stanoví váhy jednotlivých vrcholů a sestaví se matice M v požadovaném tvaru. Mohlo by se zdát zbytečné dělit tuto operaci do dvou průchodů. Ve skutečnosti se tak řešení problému zjednodušuje. V knihovně VectorEntity jsou vrcholy a trojúhelníky uloženy v samostatných kontejnerech. Jednotlivé úhly je jednodušší určit průchodem kontejneru trojúhelníků, zatímco váhy vrcholů průchodem kontejneru vrcholů. V okamžiku potřeby hodnoty velikosti úhlu, je tato hodnota dostupná přímo přes ukazatel a není nutné dohledávat odpovídající úhel v síti.

Tento postup je efektivní, vyvstává ale problém, kam ukládat velikosti úhlů. Využil jsem toho, že pro každý úhel lze určit unikátní dvojici indexů ij, ty odpovídají indexům vrcholů protilehlé hrany v daném pořadí, a tento úhel je možné zapsat přímo do matice M. Při sestavování výsledné matice se výpočet odkazuje přímo na tyto předpočítané hodnoty a následně je nahradí výslednými. Díky tomuto postupu není nutné alokovat velký rozsah další paměti.

4.2.3 Vektorové pole

Gradientní vektory se spočítají průchodem kontejneru trojúhelníků a dosazením odpovídajících hodnot do jednoduché lineární soustavy. Její vyřešení opět zajišťuje funkce z knihovny LAPACK. Isoparametrické vektory se určí jako vektorový součin gradientního a normálového vektoru. Vypočítané vektory se ukládají do instance třídy `Tri` propojené přes ukazatel s odpovídajícím trojúhelníkem. Díky tomu jsou následně přístupné přes rozhraní knihovny VectorEntity.

4.2.4 Trasování plovoucích linek

Plovoucí linky musí mít určitý počátek. Podle doporučení metody HFQRAM se jako první počáteční body zvolí všechny kritické body a vloží se do fronty. V současné implementaci se fronty `std::queue` zatím vkládají jen extrémy, protože jsem ještě nevyřešil detekci sedlových bodů. Pro každý vektorový tok je vytvořena jedna fronta. Po určení každého segmentu plovoucí linky se vygenerují v pravidelných intervalech nové počáteční body a vloží se do fronty ortogonálního toku.

Pro správné určení následujícího uzlu plovoucí linky je každý uzel odpovídajícího typu s příslušnou metodou (viz diagram 4-1 – třídy odvozené od `Point`), která řeší následující průsečík v závislosti na poloze uzlu. Jednotlivé uzly se řetězí do lineárního seznamu. Kromě vzájemného zřetězení se každý segment ještě vloží do seznamu příslušného trojúhelníka, přes který prochází. Toho se využívá při detekci vzdálenosti od jiných linek. Pro další použití je nutné, aby každý uzel obsahoval jedinečný identifikátor linky, na které se nachází, a informaci o vzdálenosti od počátku linky.

Vzdálenost mezi linkami se kontroluje trasováním virtuální linky ortogonálního toku, tak jak jsem popisoval v návrhu v kapitole 3.4. Tato linka kontroluje seznamy segmentů v trojúhelnících, přes které prochází. Pokud v určité vzdálenosti nenalezne žádný segment, může se pokračovat v trasování vlastní linky.

4.2.5 Konstrukce nové sítě

Jednotlivá křížení plovoucích linek se řeší výpočtem průsečíků segmentů v rámci každého trojúhelníka. Pro ukládání křížení se mi nejvíce osvědčilo vytvořit pro každou plovoucí linku množinu křížení, konkrétně `std::set` ze standardní výbavy C++, která umožňuje křížení třídit podle pozice na plovoucí lince. Každé křížení je umístěno ve dvou takových množinách (jedna pro gradientní tok, druhá pro isoparametrický). Při následné triangulaci sítě potom není problém určit pro každé křížení jeho předchůdce či následníka v gradientním i isoparametrickém toku.

Generované polygony dělím přímo na trojúhelníky. Tyto trojúhelníky potom vkládám do kontejneru trojúhelníků nové sítě. Pro dělení nejběžnějších konvexních polygonů, které se v síti mohou vyskytovat, jsem implementoval šablony. Tímto způsobem je síť z velké části pokryta. Pro komplikovanější polygony mi zbývá implementovat nějaký obecný algoritmus, např. Delaunay triangulation.

4.3 Další postup

Implementoval jsem všechny kroky popisované metody, takže program produkuje smysluplné výsledky. Nelze ale říci, že by byl zcela kompletní. Některé části programu jsou pouze ve zjednodušené verzi, buďto po funkční stránce nebo z hlediska efektivity provedení. Některá možná vylepšení se týkají uživatelského prostředí. Dalším významným krokem mohou být rozšíření v podobě převodu quadrilaterální sítě na NURBS nebo T-Spline plochu.

4.3.1 Optimalizace

Přestože jsem se celou dobu snažil psát čistý a efektivní kód, některé nedostatky se projeví až při komplexní analýze již fungujícího programu. Některé postupy navíc nebylo možné z časových důvodů použít, vhodnější bylo implementovat fungující program.

Nejdůležitější optimalizace se týká urychlení řešení matice pro výpočet skalárního pole. V současné verzi se používá optimalizovaná funkce z knihovny LAPACK. Ta je ale navržena pro řešení obecných matic, zatímco matice, která se řeší zde, je řídká. Knihovnu LAPACK používám jako dočasné řešení a od začátku počítám s použitím alternativní metody.

Dost možná by šlo nějakým způsobem vylepšit trasování plovoucích linek. Stávající algoritmus je dost složitý a šel by zjednodušit. Neočekávám ale, že by se časová náročnost rapidně změnila. Jak můžete vidět v tabulce 5-3, nejedná se o kritickou část programu.

Kritická část aplikace je procházení grafu křížení. Současná implementace využívá množinu `std::set`, což mi z počátku přišlo jako efektivní řešení. Naměřené hodnoty v tabulce 5-3 tomu ale moc neodpovídají. Věřím tomu, že existuje výkonnější datová struktura, která by mohla znatelně urychlit přesíťování rozsáhlých sítí.

4.3.2 Uživatelské prostředí

V současné verzi program nenabízí příliš uživatelské volnosti ani neoplývá neotřelým designem. Nastavení zobrazení se provádějí z příkazové řádky nebo přepínacími klávesami. Pro reálné aplikace je nutné vyřešit uživatelské zadávání extrémů, umožnit vytvoření výstupu použitelného pro další použití apod. Se všemi těmito rozšířeními moje implementace počítá.

4.3.3 Rozšíření

Aktuální verze programu umožňuje vytvářet pouze izotropní sítě. Doplnění vzorkovací distanční funkce by umožnilo produkovat i sítě anisotropní.

Počáteční nestrukturovaná trojúhelníková síť neskýtá příliš možností. Výsledná quadrilaterální síť je použitelná pro další zpracování. Po exportování dat do souboru, např. ve formátu STL, by bylo možné síť otevřít a upravovat v libovolném 3D editoru, který pracuje s touto reprezentací. Jedná se o hodně rozšířený způsob vyjádření sítě, existují však i vhodnější. Převod na quadrilaterální síť otevírá dveře pro převod např. na NURBS nebo T-Spline plochu, viz kapitola 2.3, které nabízejí více uživatelského komfortu a možností, tím i širší využití.

5 Výsledky

5.1 Program

5.1.1 Prerekvizity

Pro spuštění programu je nutné mít nainstalované prostředí OSG. To je ke stažení na domovské stránce http://www.openscenegraph.com. Ne nezbytným, každopádně vhodným požadavkem je výkonný počítač s grafickou kartou, která korektně podporuje moderní standardy.

5.1.2 Spuštění

Program se spouští z příkazové řádky s jedním povinným parametrem, který specifikuje zdrojový soubor ve formátu STL. Všechny parametry se zadávají ve tvaru `<parametr>:<hodnota>`, na jejich pořadí nezáleží. Pokud zadáte některý parametr vícekrát, aplikuje se hodnota jeho posledního výskytu. Při vynechání volitelného parametru se použije implicitní hodnota.

Tabulka 5-1 : *Parametry pro spuštění programu. Nezáleží na pořadí zadání, parametry vyznačené (!) jsou povinné, při zadávání není nutné dodržet velikost (case-insensitive), implicitní hodnoty jsou podtrženy.*

Parametr	**Hodnoty**
FILE (!) Zdrojový soubor.	Cesta k souboru, povinný parametr.
SHADING Stínovací model.	FLAT – ploché stínování SMOOTH – hladké stínování
PLOT Způsob znázornění skalárního pole.	NA – bez znázornění GRADIENT – barevný přechod STRIPES – barevné pruhy
VECTORS Zobrazení daných vektorů (gradientní, isoparametrické a normálové vektory).	Variace počátečních znaků názvů vektorů (GIN) v libovolném pořadí.
DARKCOLOR Barva tmavého pruhu pro znázornění skalárního pole způsobem PLOT:STRIPES.	Hexadecimální vyjádření barvy ve formátu RRGGBB (404040).

LIGHTCOLOR Barva světlého pruhu pro znázornění skalárního pole způsobem PLOT:STRIPES.	Hexadecimální vyjádření barvy ve formátu RRGGBB (c0c0c0).
STRIPECOUNT Počet pruhů pro znázornění skalárního pole způsobem PLOT:STRIPES.	<1..1024>

Příklad spuštění se zdrojovým souborem `plane1.stl`, nastavené hladké stínování a zobrazení skalárního pole pomocí pruhů:

```
> IBP.EXE FILE:PLANE1.STL SHADING:SMOOTH SCALARFIELD:STRIPES
```

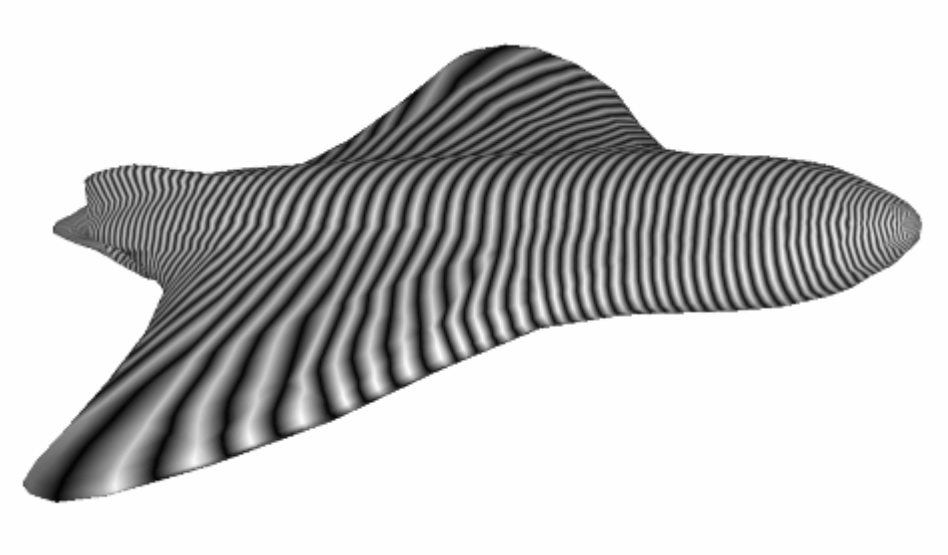
5.1.3 Ovládání

Pomocí myši lze s modelem otáčet (držení levého tlačítka), přibližovat, oddalovat (držení pravého tlačítka) a posouvat pohled (držení obou tlačítek). Pro komfortnější ovládání je možné většinu nastavení, která byla popsána v kapitole 5.1.2, přepínat přímo v programu pomocí přepínacích kláves, viz tabulka 5-2.

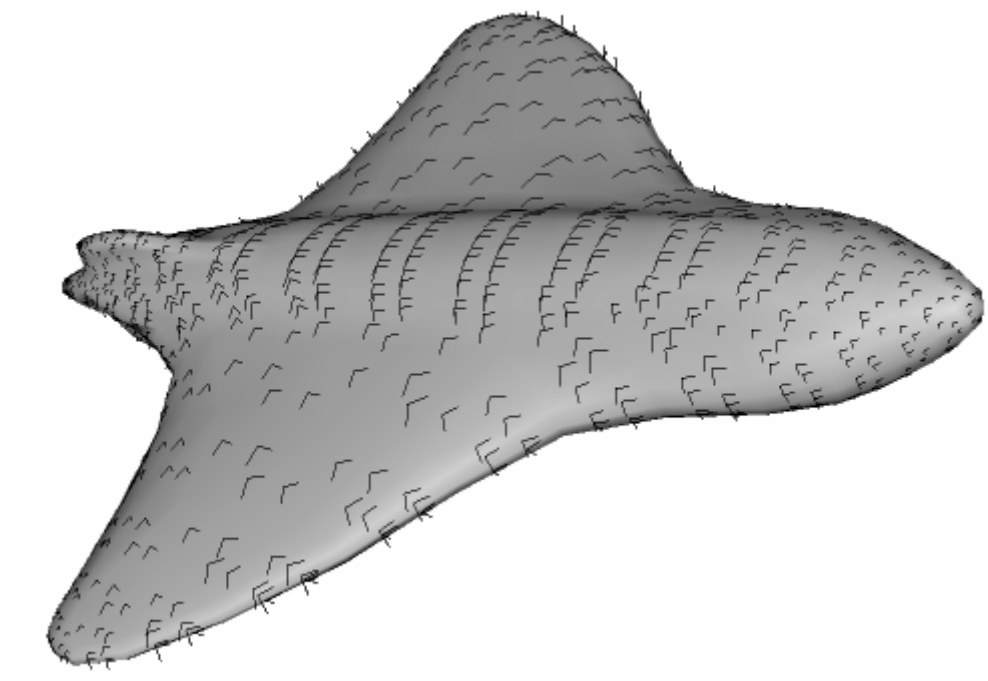
Tabulka 5-2 : *Tlačítka pro ovládání programu. Pro přehlednost jsou uvedena velkými písmeny, v programu se však používají bez přidržení klávesy SHIFT.*

Klávesa	Funkce
F	FULLSCREEN – přepíná zobrazení přes celou obrazovku.
S	STATISTIC – přepíná zobrazení statistik.
W	WIREFRAME – přepíná zobrazení polygonů, sítě hran, vrcholů.
B	BACK – přepíná zobrazení zadní části tělesa.
L	LIGHT – přepíná aktivitu osvětlení.
SPACE	Resetuje nastavení.
P	PLOT – přepíná módy zobrazení skalárního pole.
G	GRADIENT VECTORS – přepíná zobrazení gradientních vektorů.
I	ISOPARAMETRIC VECTORS – přepíná zobrazení isoparametrických vektorů.
N	NORMAL VECTORS – přepíná zobrazení normálových vektorů.
C	CROSSINGS – přepíná zobrazení křížení plovoucích linek.
R	TRACED LINES – přepíná zobrazení trasovaných linek.

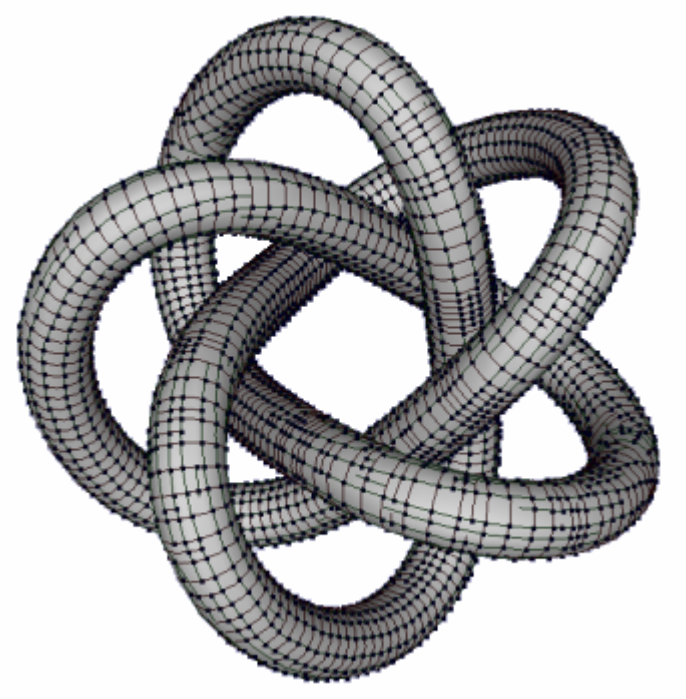
5.2 Ukázky



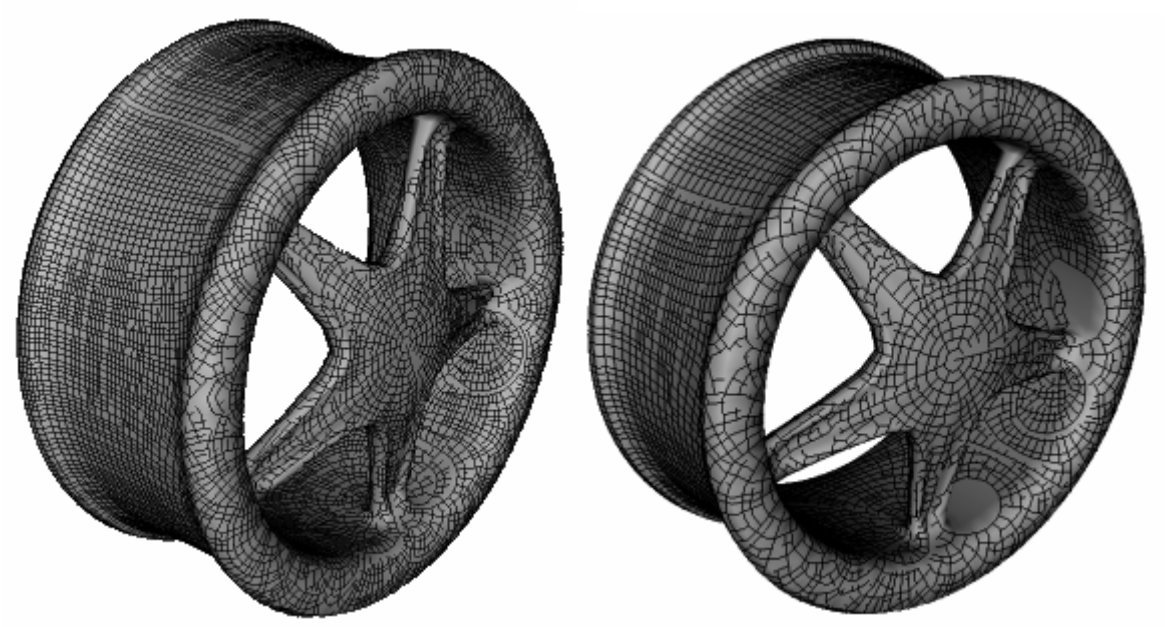
Obrázek 5-1 : *Znázornění toku skalárního pole na síti modelu letadla*



Obrázek 5-2 : *Zobrazení vektorového pole na modelu letadla.*

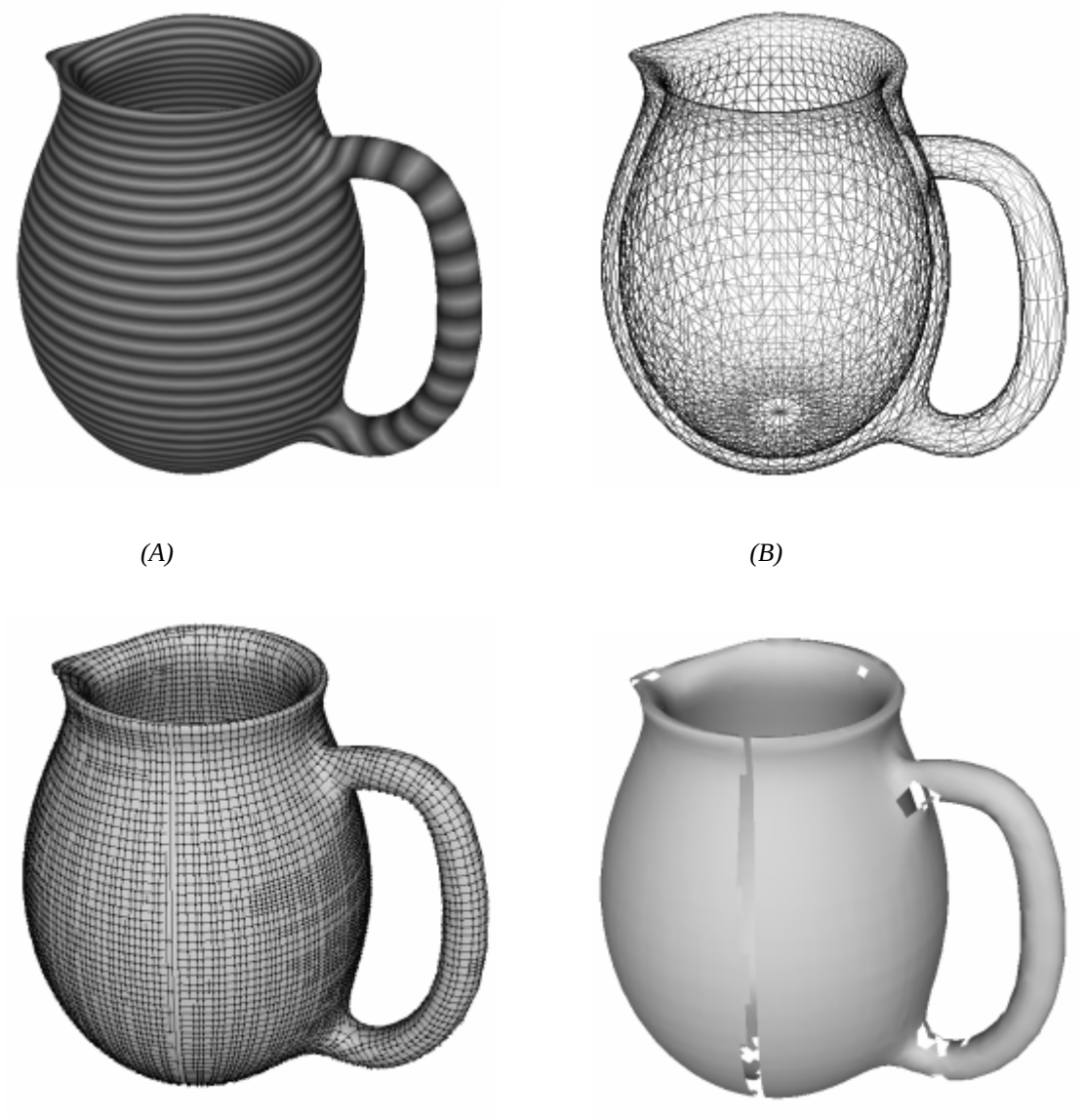


Obrázek 5-3 : *Síť plovoucích linek obou toků a jejich průsečíky na modelu uzlu.*



(A) (B)

Obrázek 5-4 : *Zobrazení plovoucích linek na modelu disku litého kola ve dvou nastaveních hustoty.*



(A) *(B)*

(C) *(D)*

Obrázek 5-5 : *Model džbánu. Na podstavě džbánu je umístěno minimum skalárního pole, na dně maximum. (A) znázornění toku skalárního pole, (B) trojúhelníková síť modelu, (C) síť plovoucích linek, (D) výsledný model ve fázi dolaďování triangulace.*

5.3 Testy

Z tabulky 5-3 je vidět, že počet vrcholů a trojúhelníků v síti hodně ovlivňuje rychlost výpočtu skalárního pole. Vhledem k tomu, že se počítá čtvercová matice obecnou metodou, jedná se o kvadratickou složitost. Změna parametru hustoty plovoucích linek ovlivňuje rychlost trasování a zejména triangulaci polygonů. Vhledem k rapidnímu zpomalení triangulace při větší hustotě čar lze usuzovat, že nebyl zvolen zcela vhodný algoritmus. Některé převody jsem opakoval vícekrát, pro ilustraci toho, jak může být měření zkresleno aktuálním stavem operačního systému.

Tabulka 5-3 : *Tabulka rychlostí v sekundách jednotlivých fází převodu různých sítí. Jednotlivá měření jsou seřazena vzestupně podle počtu vrcholů v síti.*

Skalární pole	Vektorové pole	Trasování linek	Výpočet průsečíků	Triangulace	Celkem
Letadlo, 738 vrcholů, 1472 trojúhelníků, h = 5					
0.23	0.01	0.05	0.01	0.311	0.611
0.28	0.01	0.06	0.001	0.281	0.632
Džbán, 1028 vrcholů, 2056 trojúhelníků, h = 5					
0.831	0.01	0.2	0.07	4.998	6.109
Džbán, 1028 vrcholů, 2056 trojúhelníků, h = 2					
0.58	0.01	0.972	0.47	68.499	70.531
3D uzel, 2400 vrcholů, 4800 trojúhelníků, h = 5					
6.739	0.02	0.1	0.02	8.352	15.231
7.621	0.02	0.12	0.221	9.113	17.095
7.42	0.02	0.11	0.02	9.064	16.634
Letadlo, 2946 vrcholů, 5888 trojúhelníků, h = 5					
13.609	0.03	0.301	0.06	2.173	16.173
15.212	0.04	0.32	0.07	2.564	18.206
Džbán, 3960 vrcholů, 7920 trojúhelníků, h = 5					
30.153	0.04	0.33	0.08	4.446	35.049
Litý disk, 4022 vrcholů, 8060 trojúhelníků, h = 7					
37.213	0.04	0.31	0.05	1.332	38.945
Litý disk, 4022 vrcholů, 8060 trojúhelníků, h = 5					
31.745	0.04	0.411	0.12	3.235	35.551
Litý disk, 4022 vrcholů, 8060 trojúhelníků, h = 3					
40.097	0.04	0.922	0.28	20.48	61.819

6 Závěr

Tento projekt mi zabral asi 8 měsíců průběžné práce. Přes všechny komplikace, které s sebou přináší každý větší projekt, jsem došel ke zdárnému konci. Celý postup jsem shrnul v této práci. Popsal jsem zde problematiku modelování 3D těles v počítači. Vysvětlil jsem problémy spojené s nestrukturovanými trojúhelníkovými 3D sítěmi a důvody pro hledání metod pro převod na vhodnější reprezentace. Jednou z alternativ je převod na quadrilaterální síť, k čemuž slouží řada metod. Z metod použitelných pro řešení tohoto problému jsem si jednu vybral, podrobně jsem prozkoumal princip a následně jsem tuto metodu implementoval. Výsledný program v současné verzi není zcela kompletní, ale již teď koncepčně zvládá celý postup. Již teď produkuje smysluplné výsledky a nadále je možné ho rozšiřovat. Po dodělání některých nedostatků a po optimalizacích by mohl program posloužit pro reálné využití. Dalšími vlastnostmi, kterými by mohl být program rozšířen, je převod výsledné quadrilaterální sítě na další reprezentace jako třeba NURBS nebo T-Spline plochu, které by umožnily komfortní a komplexní zpracování v moderních modelovacích nástrojích.

Co se týče toho, co jsem se naučil. Musel jsem pochopit různé matematické záležitosti, které jsem dříve neznal. Používal jsem několik programových knihoven s odlišnými architekturami. Zjistil jsem, jak může být C++ nevyzpytatelné. Vyhledával jsem různé informace, především na internetu, často v angličtině. Všechno toto mě hodně obohatilo.

Ohledně mého přínosu pro řešení této problematiky nemohu říct, že bych přišel s nějakým zásadním objevem. Hlavním cílem bylo zorientovat se v problematice a navrhnout možné řešení. Přestože metoda, podle které jsem prováděl implementaci, dosti podrobně specifikuje celý postup, poskytuje také dostatek prostoru pro tvůrčí nápady. Např. pro trasování plovoucích linek jsem využil výhod knihovny VectorEntity a navrhnul jsem vlastní postup. V současnosti je aplikace pouze takovým polotovarem reálné aplikace, do budoucna ale počítám s tím, že pod vedením Ing. Přemysla Krška, PhD. dovedu aplikaci do reálně použitelné verze.

Literatura

[Alliez et al. 2003] Alliez P., Cohen-Steiner D., Devillers O., Levy B., Desbrun M.: *Anisotropic Polygonal Remeshing.* ACM Transactions on Graphics, 2003, str. 485–493. [online], [cit. 17.04.2007], URL: ftp://ftp-sop.inria.fr/prisme/alliez/anisotropic.pdf

[Garland et al. 2004] Ni X., Garland M., Hart J. C.: *Fair Morse Functions for Extracting the Topological Structure of a Surface Mesh.* ACM Transactions on Graphics, 2004, 23(3), str. 613–622. [online], [cit. 17.04.2007], URL: http://graphics.cs.uiuc.edu/~jch/papers/morsecut.pdf

[Garland et al. 2005] Dong S., Kircher S., Garland M.: *Harmonic Functions for Quadrilateral Remeshing of Arbitrary Manifolds.* Computer Aided Geometry Design, Special Issue on Geometry Processing, 2005, 22(5), str. 392–423, 2005. [online], [cit. 17.04.2007], URL: http://graphics.cs.uiuc.edu/%7Egarland/papers/harmonic-preprint.pdf

[Žara 1998] Žara J., Beneš B., Felkel P.: *Moderní počítačová grafika.* 1. vyd., Praha, Computer press, 1998, 448 s., ISBN 80-7226-049-9

[Wikipedia 2006] *Saddle Point*. Wikipedia, 06.11.2006. [online], [cit. 25.04.2007], URL: http://en.wikipedia.org/wiki/Saddle_point.

[Wikipedia 2007a] *Harmonic function*. Wikipedia, 11.04.2007. [online], [cit. 17.04.2007] , URL: http://en.wikipedia.org/wiki/Harmonic_function

[Wikipedia 2007b] *Laplace's equation.* Wikipedia, 11.04.2007. [online], [cit. 17.04.2007], URL: http://en.wikipedia.org/wiki/Laplace's_equation

[Wikipedia 2007c] *STL (file format)*. Wikipedia, 20.04.2007. [online], [cit. 01.05.2007], URL: http://en.wikipedia.org/wiki/STL_(file_format)

[Wikipedia 2007d] *Non uniform rational B-Spline.* Wikipedia, 21.04.2007. [online], [cit. 04.05.2007], URL: http://en.wikipedia.org/wiki/Nonuniform_rational_B-spline

[Sederberg et al. 2004] Sederberg T. W. et al.: *T-spline Simplification and Local Refinement.* ACM Transactions on Graphics, 23(3), 2004. [online], [cit. 09.05.2007], URL: http://cagd.cs.byu.edu/~tspline/innovation/papers

[Wolfram 2002] *Kronecker Delta*. Wolfram MathWorld, 09.06.2002, [online], [cit. 04.05.2007], URL: http://mathworld.wolfram.com/KroneckerDelta.html

[Watanabe and Belyaev 2001] Watanabe K., Belyaev A. G.: *Detection of salient curvature features on polygonal surfaces.* Computer Graphics Forum, 2001, 20(3), str. 385–392

[Page et al. 2002] Page D. L. et.al.: *Normal vector voting: crease detection and curvature estimation on large, noisy meshes.* Graph. Models, 2002, 64(3/4), str. 199–229. ISSN 1524-0703

Seznam příloh

Přiložené CD obsahuje:

1. Zdrojové kódy programu.
2. Dokumentace zdrojových kódů.
3. Testovací modely.
4. Manuál programu ve formátu PDF.
5. Elektronická verze technické zprávy ve formátu PDF.