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

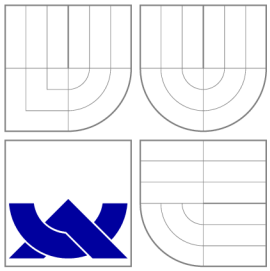
DETEKCE ZUBŮ NA 3D POČÍTAČOVÉM POLYGONÁLNÍM MODELU ČELISTI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

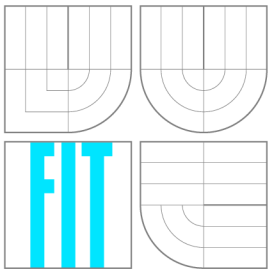
AUTOR PRÁCE
AUTHOR

JAN FILIP

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

DETEKCE ZUBŮ NA 3D POČÍTAČOVÉM POLYGONÁLNÍM MODELU ČELISTI

TOOTH DETECTION ON JAW 3D COMPUTER POLYGONAL MODEL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN FILIP

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PŘEMYSL KRŠEK, Ph.D.

BRNO 2007

Zadání bakalářské práce/6187/2006/xfilip20

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Filip Jan**
Obor: Informační technologie
Téma: **Detekce zubů na 3D počítačovém polygonálním modelu čelisti**
Kategorie: Počítačová grafika

Pokyny:

1. Stručně prostudujte problematiku počítačového 3D modelování.
2. Prostudujte problematiku detekce objektů a tvarů na 3D polygonálním modelu.
3. Navrhněte metodu pro detekci zubů na 3D počítačovém polygonálním modelu čelisti.
4. Implementujte a otestujte navrženou metodu.
5. Zhodnoťte dosažené výsledky a stanovte další vývoj projektu.

Literatura:

1. 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:

- Bez požadavků.

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 nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), 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: 15. června 2007
Datum odevzdání: 31. července 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

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: **Jan Filip**
Id studenta: 84404
Bytem: Lukavská 851, 564 01 Žamberk
Narozen: 27. 02. 1985, Ústí nad Orlicí
(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: Detekce zubů na 3D počítačovém polygonálním modelu čelisti
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: .....

.....	..... Filip .....
Nabyvatel	Autor

Abstrakt

Cílem této bakalářské práce je návrh a implementace nástroje pro detekci zubů na 3D počítačovém polygonálním modelu čelisti, který umožní uživateli získat tvary zubů na čelisti. Tento nástroj by měl pomoci ve stomatologii. V případě jakékoli nepřesnosti při detekci lze měnit některé parametry a metody, kterými bude možné ovlivnit přesnost detekce. Program bude importovat model ve formátu STL a následně uživateli umožní v prohlížeči prohlédnout výsledek detekce. Detekované modely zubů bude možno exportovat ve formátu STL a to jednotlivě, či dohromady. Implementačním jazykem bude C++, kompilátorem GNU C v distribuci MinGW, toolkitem pro práci s 3D modelem bude MDSTk a zejména jeho část VectorEntity. Pro zobrazení 3D scény bude užit toolkit OpenSceneGraph. Aplikace bude vyvíjená hlavně pro OS MS Windows, ale díky užití GNU nástrojů by měla být na úrovni zdrojových kódů dobře přenositelná na jiné OS.

Klíčová slova

Polygonální modely, Gaussova křivost, MDSTk, OpenSceneGraph, MinGW, reprezentace 3D modelů, STL.

Abstract

Aim of this bachelor thesis is a proposal and an implementation of a tool for the tooth detection on a 3D computer polygonal jaw model, which will allow to a user to obtain the shape of teeth on the jaw. This tool should to help in the stomatology. In the event of any inaccurary in the detection it will be possible to change certain parameters and methods, by which will be possible to influence the accuracy of detection. Program will import models in the STL format and subsequently will allow to the user to see the result of the detection in a browser. Teeth detection models will be possible to export single or together, in the STL format. The implementation language will be C++, compiler GNU C in distribution MinGW, toolkit to work with the 3D model will MDSTk and primarily its part VectorEntity. For imaging of thr 3D scene will be used OpenSceneGraph toolkit. Application will be developed primarily for MS Windows operating system but thanks to using GNU tools should be portable at the level of source codes to anothers operating systems.

Keywords

Polygonal models, Gauss curvature, MDSTk, OpenSceneGraph, MinGW, representation of 3D models, STL.

Citace

Jan Filip: Detekce zubů na 3D počítačovém polygonálním modelu čelisti, bakalářská práce, Brno, FIT VUT v Brně, 2007

Detekce zubů na 3D počítačovém polygonálním modelu čelisti

Prohlášení

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

.....
Jan Filip
9. srpna 2007

Poděkování

Rád bych poděkoval Ing. Přemyslu Krškovi, Ph.D. za jeho pomoc, ochotu, trpělivost, odborné vedení a připomínky týkající se této práce.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zabývá detekcí zubů na polygonálním modelu čelisti a s tím související Gaussovou křivostí, která se při detekci využívá jako jeden z nástrojů při analýze křivosti povrchu. Tento polygonální 3D model je získáván skenováním reálné čelisti.

Hlavní uplatnění programu by tedy mělo být v medicíně v oboru stomatologie. Program by měl zpracovat importovaný model tak, aby bylo možné si importovaný model prohlédnout a následně na něm detekovat zuby a to jednotlivě nebo zvlášť. Výsledek pak může být exportován. Program není koncipován tak, aby poskytoval všechny možnosti editoru 3D objektu. Mělo by jít spíše o specializovaný pomocný nástroj.

Teoretická část práce je rozdělená na několik podkapitol a v převážné míře se zabývá reprezentací 3D modelů. Zaměření na tuto oblast je z toho důvodu, že každý základní editor a prohlížeč musí nějakým způsobem 3D model reprezentovat. Zabývá se tedy takovými metodami jako jsou konstruktivní geometrie, dekompoziční modely a hlavně hranovou reprezentací nebo-li tzv. *B-Rep*. Vše je doprovázeno spoustou ilustračních obrázků.

Dále je zde uvedena analýza řešeného problému, která se snaží nalézt záchytné body problému, na základě kterých je možné problém vyřešit.

Další kapitolou je návrh implementace. Zde je popsána struktura aplikace. Dále jak by mělo vypadat uživatelské rozhraní a nakonec je zde popsána metoda detekce zubů pomocí analýzy křivosti povrchu a následné dodatečné optimalizační metody.

Další částí je implementace. Zde je ve popsáno jak jsou jednotlivé třídy a důležité funkce tvořeny, jak fungují a obsahuje take náčrty nejduležitějších algoritmů.

V další části jsou pak ukázány výsledky detekce a na závěr nalezneme shrnutí práce a možnosti dalšího vývoje. Na úplném konci pak ještě nalezneme použitou literaturu, stručný návod k ovládání aplikace a ukázky výsledků aplikace.

Kapitola 2

Teoretický rozbor problematiky

V rozboru se nejprve zaměřuji na formulaci cíle této bakalářské práce. Dále je zde trocha teorie z oblastí počítačové grafiky, které se týká tato práce, zejména technik reprezentace 3D modelů.

2.1 Cíl práce

Výsledkem této bakalářské práce by měl být program pro detekci zubů na 3D polygonálním počítačovém modelu čelisti.

2.2 Reprezentace 3D modelů

Svět je trojrozměrný, proto chceme-li být schopní ho popsat, musíme ho popsat pomocí trojrozměrných objektů, ze kterých se svět skládá. Pokud chceme reprezentovat 3D model pomocí libovolné techniky, tak existují základní požadavky na tuto techniku jak píše [7]. Měla by tedy být maximálně obecná, aby šlo pomocí ní popsat co nejrozsájhlejší třídu objektů. Dále musí být schopná úplně popsat daný objekt a to jednoznačně, pro vyhodnocení jediným způsobem. Model musí být unikátní a jedinečný, takže jednomu tělesu odpovídá jeden model. Samozřejmostí je též přesnost popisu, jeho regulérnost, tedy to že není možné vytvořit nereálný objekt. Také je důležitá konzistentnost modelu tělesa stejné třídy vůči vybraným operacím a v neposlední řadě jeho kompatnost a možnost efektivního zpracování. Nakonec se stejně většina typů převádí pro samotné zobrazení na polygonální model. Grafickému procesoru se pracuje totiž většinou nejlepe s polygonálnímy modely. Dále si uvedeme několik různých reprezentací 3D modelů. Toto a více o reprezentaci 3D modelů lze nalést v [7].

2.2.1 Konstruktivní geometrie(CSG)

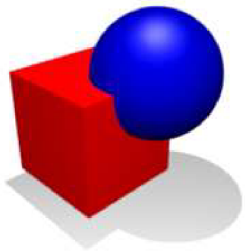
Konstruktivní geometrie nebo-li *CSG - constructiv solid geometry*, je reprezentací tělesa, která je založená na reprezentaci tělesa stromovou strukturou - takzvaným CSG stromem, která uchovává historii dílčích kroků. Tato technika, jak píše [7], popisu tělesa a je často používáná v CAD systémech, protože odráží postupy používané konstruktérem při navrhování těles.

Z jednoduchých geometrických objektů, takzvaných CSG primitiv, je pomocí množinových operací a prostorových transformací vytvořen výsledný objekt. Často tak vypadá prezento-

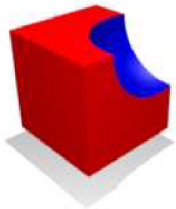
vaný CSG model nebo povrch, tak že vypadá vizuálně komplexně, ale jde přece jen o chytře zkombinované a dekombinované objekty.

Jako primitiva často slouží jednoduchá tělesa jako kvádr, koule, válec, kužel, jehlan či toroid. Seznam použitelných primitiv je však podle [10] často limitován softwarovým balíkem. Některé dovolují i abstraktnější entity jako jsou poloprostor nebo plocha NURBS.

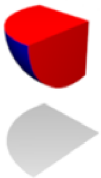
Bylo řečeno, že objekt je vytvářen pomocí množinových operací odpovídajících konstrukčním postupům a mohou být prováděny jak nad jednotlivými CSG primitivy[7], ale také nad celými CSG stromy[7]. Prováděné operace jsou sjednocení(obr.2.1), rozdíl(obr.2.2) a průnik(obr.2.3).



Obrázek 2.1: Operace sjednocení primitiv (Obrázek od [10]).



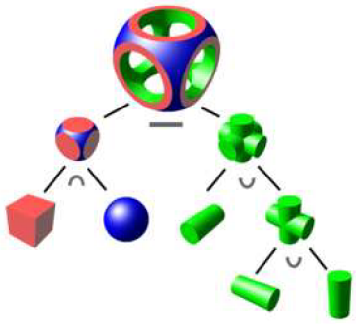
Obrázek 2.2: Operace rozdílu primitiv (Obrázek od [10]).



Obrázek 2.3: Operace průniku primitiv (Obrázek od [10]).

Příklad CSG stromu je na obrázku 2.4. Vnitřní uzly CSG stromu obsahují operace a v listech jsou zapsány údaje o primitivech. Transformace mohou být chápany jako operace

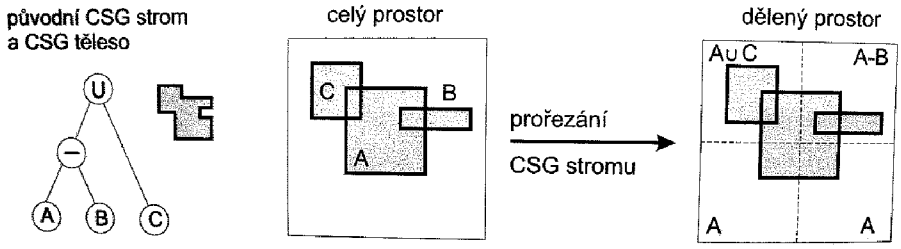
nebo mohou být zaznamenávány ke každému primitivu [7], takže vnitřními uzly budou pouze množinové operace.



Obrázek 2.4: Ukázka CSG stromu (Obrázek od [10]).

Reprezentace tělesa CSG stromem není příliš vhodná pro zobrazení, protože neobsahuje přímo vykreslitelné prvky, jako jsou hrany a plochy [7]. Proto je lepší převést CSG strom do jiné lépe zobrazitelné reprezentace, třeba hraniční.

Vyhodnocení CSG stromu je složitý proces, proto se využívá techniky *prořezávání CSG stromu*. Princip je dle [7] jednoduchý, pokud se v dané části prostoru nějaké primitivní těleso nevyskytuje, odstraní se nevyužitá větev CSG stromu. Viz. obr. 2.5. Proces *prořezávání* tedy zásadně zjednodušuje původní CSG strom.



Obrázek 2.5: Ukázka prořezávání CSG stromu umístěného do rozděleného prostoru(2D pohled) (Obrázek od [7]).

Více lze najít v [10].

2.2.2 Šablonování

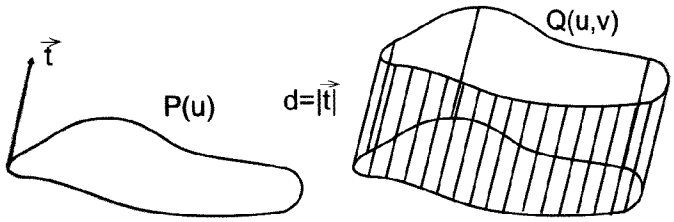
Šablonování je metoda, při které získáváme plochu tažením dvourozměrného obrysu(tzv. profilu) po trojrozměrné křivce - tzv. *páteři*(*spine curve*)[7]. Šablonování se rozděluje do tří kategorií a dále budou uvedeny.

Translační šablonování

Obrys u této metody je libovolný a páteř je úsečka. Touto technikou získáváme přímkovou plochu. Přímkové plochy se v jednom ze směrů skládají z úseček a lze je reprezentovat

pomocí NURBS. Přímkovou plochu můžeme tedy získat vytažením nebo spojením a to třemi způsoby[7]:

1. **S profilovou křivkou, která nemění svůj směr.** Vytažená plocha (*extruded surface*) se v tomto případě získá z křivky $P(u)$ vytažením ve směru $\vec{t}$ o vzdálenost $d = |\vec{t}|$ (obr.2.6).

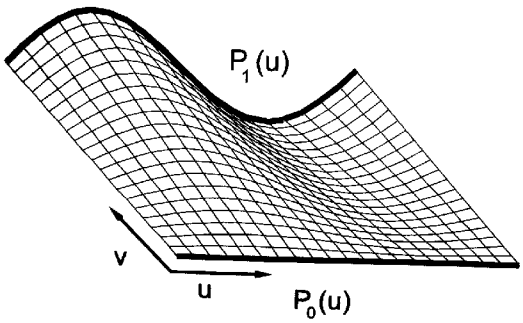


Obrázek 2.6: Plocha vpravo je získaná vytažením z profilové křivky $P(u)$ vlevo a to ve směru $\vec{t}$ o vzdálenost $d = |\vec{t}|$ (Obrázek od [7]).

S obrázku 2.6 je vidět, že například posunutím kružnice lze vytvořit válcovou plochu nebo posunem čtverce hranolovou plochu. Výsledkem je ale vždy plocha, jejíž profil v řezu kolmém na trajektorii se nemění.

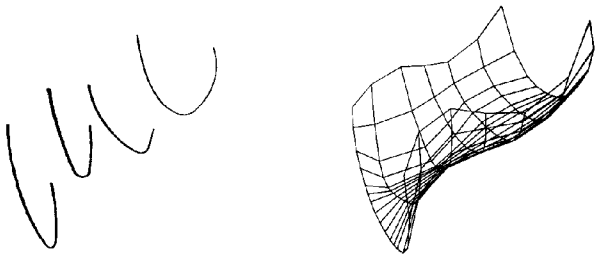
2. **S měnící se profilovou křivkou.** Tato technika propojuje dva profily (okrajové křivky) úsečkami. Přímková plocha (*ruled surface*) tedy vznikne spojením dvou křivek $P_0(u)$ a $P_1(u)$ pomocí úseček (obr.2.7) a je popsáná vztahem

$$Q(u, v) = (1 - v) \cdot P_0(u) + v \cdot P_1(u)$$



Obrázek 2.7: Přímková plocha získaná spojením dvou okrajových křivek (Obrázek od [7]).

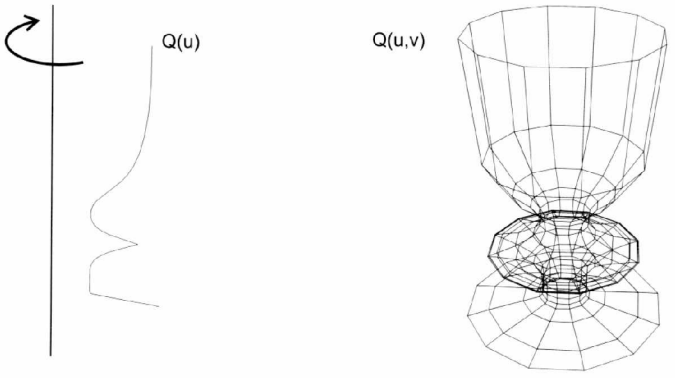
3. **Kombinací předchozích dvou způsobů.** Tedy po obecné křivce s měnící se profilovou křivkou, tzv. *potahování*(*skinning*). Potahování je vlastně interpolace křivek. Viz obrázek 2.8.



Obrázek 2.8: NURBS plocha vpravo vzniklá interpolací profilových křivek vlevo. (Obrázek od [7])

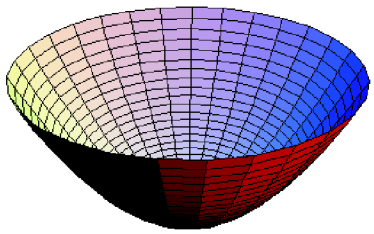
Rotační šablonování

U této metody je obrys libovolný. Trajektorie, po které je obrys tažen, je kružnice či její část. Těleso lze tedy získat rotací rovinné profilové křivky okolo osy (obr. 2.9). Výsledkem operace je tedy rotační plocha (*surface of revolution*)[7].



Obrázek 2.9: NURBS plocha vpravo vzniklá rotací rovinné křivek vlevo. (Obrázek od [4])

Bude-li kupříkladu profilovou křivkou úsečka rovnoběžná s osou z a od ní o určitou vzdálenost posunutá, vznikne rotací válec. Rotací různoběžné úsečky by to byl otevřený kužel. Rotací kružnice posunuté o dostatečnou vzdálenost od osy z vznikne anuloid nebo-li prstenec a v případě , že zůstane v základní poloze a při rotaci o 180 stupňů, získáme kouli[7]. Rotací paraboly například vznikne objekt na obrázku 2.10 atd.



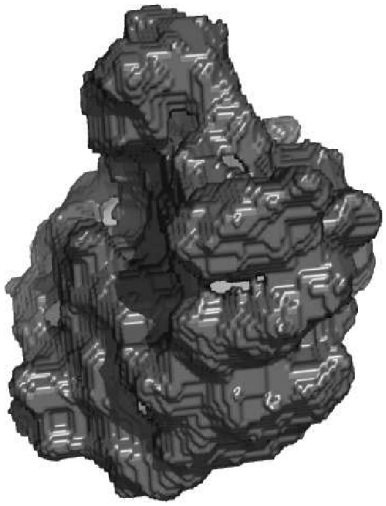
Obrázek 2.10: Parabola $y = x^2$ rotující kolem osy y. (Obrázek od [13])

Obecné šablonování

Zde je obrys i trajektorie obecná. Touto operací tedy získáme zobecněnou válcovou plochu (*generalized cylinder*).

2.2.3 Dekompoziční modely

Jedná se o metodu, kdy je diskrétní popis objektu dekompozicí jím obsazeného objemu na elementární objemové jednotky(viz. obrázek 2.11)[4]. Podle [7] tento model může obsahovat mimo jiné *rozptýlená data*, kde každý vzorek nese údaje nejen o jasu či hustotě apd., ale také své souřadnice. Diskrétní objekty, pro jejíž popis se využíva dekompoziční model, jsou například mlha, mraky, oheň, ale také data získaná z počítačových tomografů, nepravidelná data ze simulace proudění kapalin či data jež jsou výsledkem meterologických měření teploty a tlaku vzduchu.



Obrázek 2.11: Makromolekula sestavená z elementárních objemových jednotek - voxelů. (Obrázek od [14])

V aplikacích je model strukturován do podoby pravidelných, nebo nepravidelných *mřížek*. Nejčastěji však jde o pravidelnou kartézskou mřížku. Existují ale i jiné mřížky. Dělení geometrického tvaru mřížky navrhli Spear a Kennon, kteří rozdělili obvyklé tvary do sedmi tříd. Ty lze dále rozdělit na uspořádané do tvaru mřížky, kdy její geometrický tvar je různým způsobem deformován. Jsou to kartézská, pravidelná, pravoúhlá a strukturovaná mřížka. Další třídou je nestrukturovaná mřížka, jejíž topologie je uložená v poli buněk. Zbylé dva typy, blokově strukturovaná a hybridní mřížka, jsou kombinací ostatních. Více též [7].

Jedním z problému diskrétních objemových dat je jejich velké množství a tím pádem velké nároky na paměť a výkon[7]. Vzhledem k tomu, že se uchovávají jen diskrétní vzorky, je problematické objemová data natáčet o nepravé úhly, zvětšovat je či zmenšovat. Při převzorkování totiž dochází také ke ztrátě informací o příslušnosti buňky k objektu.

Objemová data mají ale i své výhody. A to snadnou práci s naměřenými daty, snadné provádění blokových a logických operací či zpracování dat jako celku[7]. Pro zobrazení libovolným zůsobem zobrazených těles pak postačuje jediný algoritmus. Pokud jde o ukládání

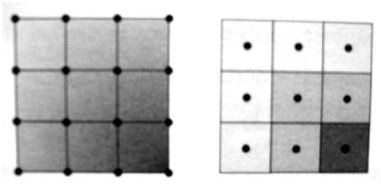
vícerozměrných dat, tak to není žádný velký problém. Horší je se v nich dokázat orientovat, zobrazit je a analyzovat.

Voxel a buňka

Základním objemovým elementem je voxel, což je vlastně analogie k dvojrozměrnému pixelu, a označuje nejmenší element ve trojrozměrném diskrétním prostoru. Má tvar kvádru či krychle.

Voxel je tedy vyplněný kvádr mající v celém svém objemu konstantní hodnotu (obr. 2.12 vpravo)[7]. Jako hodnota mezi středy voxelů je obvykle brána hodnota nejbližšího voxelu. Tento způsob se nazývá interpolací hodnotou nejbližšího souseda[7].

Avšak takové vzorkování je pro řadu algoritmů příliš hrubé, proto jsou hodnoty v uzlech mřížky častěji chápány jako bodové vzorky spojitého prostoru, přičemž osmice vzorků vytváří jednu *buňku (cell)*(obr. 2.12 vlevo)[7]. Prostorová buňka mívá různý tvar jako je čtyřstěn, kostička či n-stěn. Hodnoty uvnitř buňky se pak vypočítají pomocí lineární interpolace.

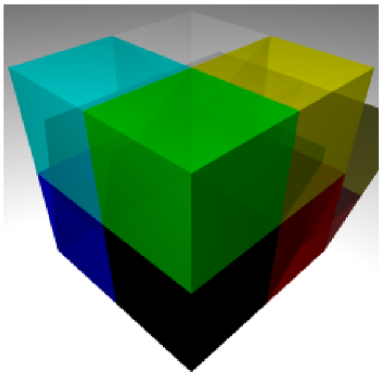


Obrázek 2.12: Základní elementy: buňky a plné krychličky. (Obrázek od [7])

Uložení dat

Data dekompozičního modelu lze ukládat třemi způsoby:

1. **Oktalový strom**(octree) je dle [4] metoda rekurzivního dělení hranolu na osm částí (obr. 2.13). Tato metoda je vhodná pro malou hustotu dat, avšak procházení stromem je zde problematické.



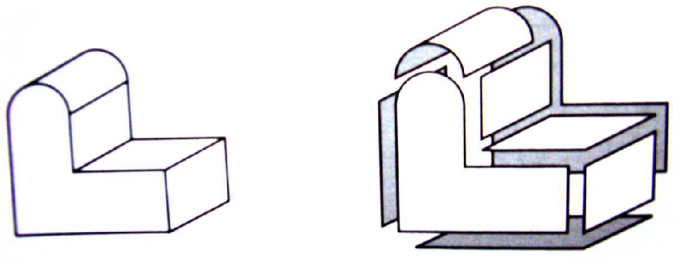
Obrázek 2.13: Uzel oktalového stromu znázorněný jako rozdělený barevný prostor (Obrázek od [11]).

2. **3D pole diskrétních hodnot** je podle [4] prosté sekvenční pole hodnot. Jeho nevýhodou je velká pamětová náročnost. Naproti tomu je zde vysoká přístupová rychlost.

3. **Subvoxely** jsou kombinací předchozích dvou metod ve snaze využít jejich výhody[4]. Jsou zde tedy různé možnosti kombinace předchozích možnosti. Navíc je tato metoda vhodná pro paralelizaci.

2.2.4 Hraniční reprezentace(B-rep)

Jedná se o jeden z nejběžnějších způsobu reprezentace těles spočívající v popisu hranice (*boundary representation = B-rep*). Tedy v popisu množiny hraničních bodů(viz obrázek 2.14). Informace o vnitřních bodech tělesa se neuchovávají a lze je odvodit. Objekty jsou tedy definovány pomocí vrcholů, hran a stěn. Základní metoda byla vyvinuta začátkem 70-tých let minulého století Ianem Braidem na Cambridge. Toto a další také na [9]



Obrázek 2.14: Popis tělesa převedený na popis pláště (Obrázek od [7]).

Manifoldy

Výše uvedená definice tělesa je např. vhodná pro definici tělesa v systémech CAD. Pro praxi je však příliš široká, dovoluje totiž popsat i nevyrobitelné objekty[7]. *Manifold* nebo i *2-manifold* je tedy model tělesa, který odpovídá skutečnému tělesu. Je vyrobitelný.

Nonmanifold je oproti tomu nevyrobytelné těleso. Vychází ze skutečnosti, že pro počítačový popis tělesa je užita matematická a geometrická abstrakce typu nekonečně tenké přímky[7].

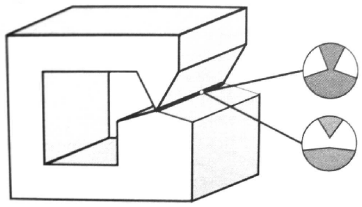
Na obrázku 2.15 je příklad *nonmanifoldu*. Zvýrazněná hrana je zde z geometrického hlediska nekonečně tenká úsečka, jež je průsečnicí čtyř ploch. V reálu zde však musí být dvě hrany a těleso je tedy propojené, či rozpojené. Za manifold tedy lze považovat každé těleso jehož každá hrana inciduje právě s dvěma plochami a jehož hrany neprotínají jiné plochy[7].

Eulerova rovnost

Eulerova rovnost udává vztah mezi počtem vrcholů (*V, vertex*), hran (*E, edge*) a stěn (*F, face*) *mnohostěnu*[7]:

$$F + V = E + 2. \tag{2.1}$$

Přičemž hraniční reprezentace jednoduchého mnohostěnu splňuje Eulerovu rovnost.



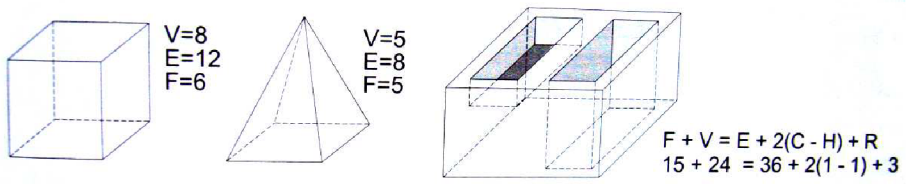
Obrázek 2.15: Nonmanifold a dvě možnosti jak ho převést na manifold (Obrázek od [7]).

Platnost Eulerovy rovnosti však sama nedokazuje, že libovolná množina vrcholů, hran a stěn tvoří mnohostěn, jenž je hranicí uzavřeného objemu. Podle [7] totiž musí též platit, že každá hrana propojuje dva vrcholy a stěny se s hranami nesmí protínat.

Pro manifoldy mající otvory platí obecná Eulerova rovnost (2.2). Mezi otvory se slepé prohlubně nepočítají. Jsou zde přidány další členy, a to počet vnitřních smyček hran (*R, ring*), počet oblastí objektu *C, component* a počet děr procházejících tělesem *H, hole*[7]:

$$F + V = E + 2 \cdot (C - H) + R. \tag{2.2}$$

Viz obrázek 2.16.



Obrázek 2.16: Mnohostěny a jejich charaktreristické prvky splňující Eulerovu rovnost (Obrázek od [7]).

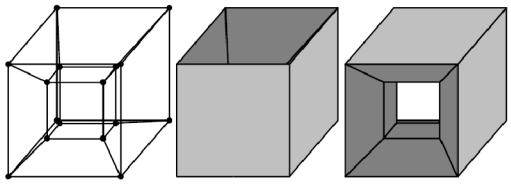
Drátový model

Drátový model (*wire-frame model*) je nejstarší a nejjednodušší metodu reprezentace povrchu tělesa. Metoda je založena na zápisu hran a vrcholů(obrázek 2.17).

Obrázek 2.17: Drátový model krychle, ikosaedru a koule (Obrázek od [15]).

Při implementaci drátového modelu jsou vytvářeny dva seznamy, jeden vrcholů, kde jsou ukládány souřadnice, a druhý hran, kde každá položka má právě dva ukazatele do seznamu vrcholů. Vzniká tedy úsporná struktura obsahující málo topologických informací a proto je

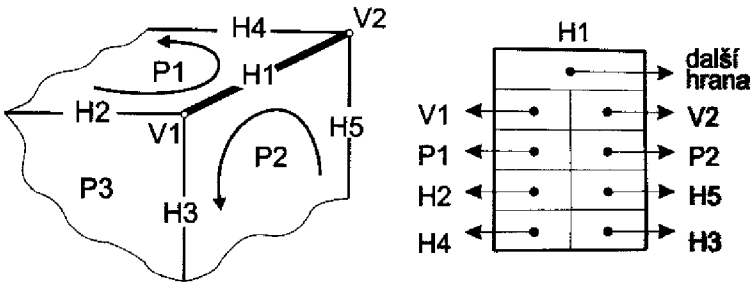
drátový model nejednoznačný. Může být totiž interpretován jako několik různých těles(viz obrázek 2.18). Pouhé vykreslení hran je však vhodné pro rychlé orientační zobrazení nebo pro průzkum jinak skrytého vnitřku tělesa. Viz také [15].



Obrázek 2.18: Nejednoznačnost drátového modelu (Obrázek od [4]).

Okřídlená hrana

Okřídlená hrana je datová struktura pro B-rep, kterou dle [7] navrhl Baumgart. Její jméno je odvozeno od toho, že grafické znázornění jedné hrany společně s prvky, které s ní sousedí připomíná křidélka(obrázek 2.19 vlevo). Datový záznam hrany obsahuje ukazetele na sousední vrcholy, hrany ohraničující stěny a sousední stěny. Všechny hrany jsou dále zřetězeny a jedno těleso je tvořeno třemi seznamy, které v hierarchickém uspořádání obsahují lineární seznamy vrcholů, hran a stěn. Přičemž na nejnižší úrovní je seznam vrcholů, na střední seznam okřídlených hran a na nejvyšší seznam ploch.



Obrázek 2.19: Okřídlená hrana a její datový záznam (Obrázek od [7]).

Obrázek 2.19 vpravo ukazuje schéma záznamu okřídlené hrany. Schéma obsahuje odkazy na oba koncové vrcholy (V1, V2), dále ukazatele na vedlejší stěny (P1, P2) a nakonec odkazy na další čtyři hrany. Vlevo jsou hrany sousedící s levou stěnou (H2, H4) a vpravé s pravou (H3, H5) a to s ohledem na orientaci těchto ploch. Horní ukazatel pak ukazuje na další hranu ve zřetězeném seznamu.

Oproti seznamu hran jsou seznami vrcholů a stěn podstatně jednodušší. Záznam každé stěny obsahuje ukazatele na všechny její hrany.

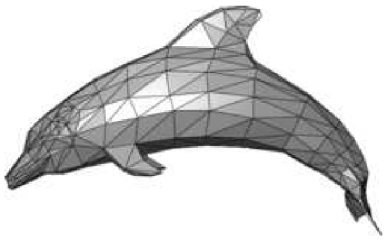
Dále si můžeme všimnout, že tato datová struktura využívá jen asi přibližně 25% paměti na geometrické údaje. Zbytek struktury je využit k popisu topologií.

Dále je třeba říci, že datová struktura okřídlené hrany je určena pouze pro manifoldy a to proto, že každá okřídlená hrana sousedí právě s dvěma stěnami. Pro reprezentaci nonmanifoldů se tedy dle [7] využívá odvozená datová struktura *půlhrana*.

Polygonální model

Polygonální model jsou díky své snadné reprezentaci nejoblibenějším typem reprezentace všech modelů.

Nejčastěji se používají sítě trojúhelníků(viz obr. 2.20). Trojúhelníky mají totiž řadu dobrých vlastností. Na rozdíl od obecných mnohoúhelníku jsou totiž konvexní a vždy leží v jedné rovině. Dále existují algorimy pro jeho rychlé vyplňování a je podporován grafickými procesory. Také mnoho geometrických operací lze nad trojúhelníkem optimalizovat.

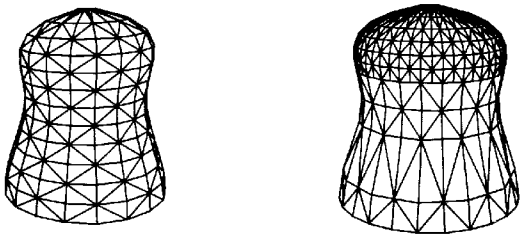


Obrázek 2.20: Ukázka modelu vytvořeného pomocí sítě trojúhelníků (Obrázek od [12]).

Skupiny trojúhelníků se nazývají sítě(*triangle mesh*) a sdílejí své hrany. Datová struktura popisující síť bývá rozdělena na *geometrickou* a *topologickou* část[7]. V geometrii se zaznamenávají souřadnice vrcholů trojúhelníků. Oproti tomu topologická část obsahuje informace o tom, které vrcholy tvoří trojúhelník, popřípadě ty, které spolu sousedí. To má své výhody například při transformaci, kdy se pouze vypočítají nové vrcholy.

Při vytváření sítí se sledují tato kritéria:

1. **přesné a úsporné vyjádření tvaru, jenž síť reprezentuje.** Toto kritérium se bere nejčastěji v úvahu při převodu z jiné reprezentace, kdy je třeba síť optimalizovat tak, aby co nejúsporněji a nejpřesněji reprezentovala vymodelovaný tvar (viz obr. 2.21).



Obrázek 2.21: Síť trojúhelníků pokrývající plochu pravidelně vlevo a jemněji v místech s větší křivostí vpravo (Obrázek od [7]).

2. **uspořádání vhodné pro další práci se sítí.** Toto kritérium podle [7] souvisí s její topologií, protože její oddělení od geometrické datové struktury je pro některé úlohy nevyhovující. Při zpracování dat v grafickém procesoru je nutné zpracovat síť jednou lineární strukturou a to při minimalizaci operací prováděných s jednotlivými vrcholy sítě. Toho lze dosáhnout uspořádáním trojúhelníků do *pruhu trojúhelníků* (*triangle strip*). To zajišťuje zpracování každého vrcholu právě jednou. Podobně lze uspořádat

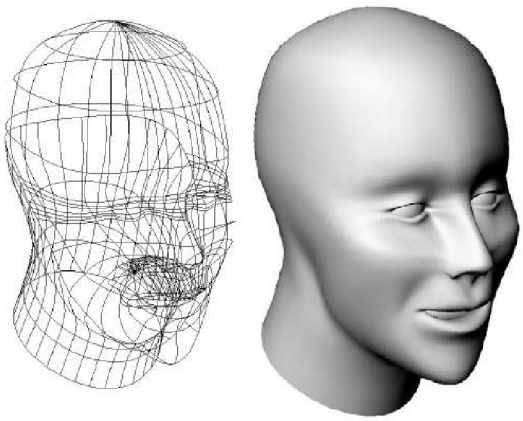
trojúhelníky do *vějíře trojúhelníků*(*triangle fan*). Ten vzniká např. triangulací konvexních polygonů.

Nalezení optimální množiny pruhů či vějířů však není jednoduché a jde o zajímavý výpočetní problém.

Trojúhelníkové sítě mají ale i své nevýhody. Mimo jiné jde o nesnadné mapování textur nebo tzv. *geometrický alias*, který se projevuje při změně měřítka[7].

Hraniční spline model

Hraniční spline model je definován pomocí vrcholů, stěn nebo-li spline ploch a někdy i hran. Tato metoda poskytuje úplné informace pro popis objektu. Přesnost modelu je samozřejmě dána přesností aproximace spline ploch (NURBS). Dále je nutnost u této metody hlídat regulérnost a uzavřenost modelu. Hraniční spline model je vhodný pro přesné geometrické modelování, ale pro zobrazení se většinou převádí na polygonální model (obrázek 2.22). Viz. [4].



Obrázek 2.22: Hraniční spline model (Obrázek od [4]).

2.2.5 Implicitní plochy

Většina 3D modelu se sice skládá ze základních primitiv jako jsou hrany a stěny. Avšak pro mnoho hladkých a deformovatelných objektů to jsou náročné a neefektivní reprezentace. A to i za použití takových primitiv jako jsou spline plochy. Praktické aplikace tedy používají i modelování založené na implicitním vyjádření objektu.

Tato myšlenka není nijak nová. Již v roce 1982 J.F. Blinn navrhl chápat izoplochy vzniklé při modelování elektrického potenciálu jako objekty - *Blobby objects*. Pak v roce 1986 G. Wyvill *Soft objects* a dále *Meta Balls*[7]. Implicitní plochy zobecnil v roce 1988 J. Bloomenthal. Implicitní plocha je tedy množina bodů P ve třírozměrném prostoru, pro jenž platí

$$Q(P) = konst.$$

Příkladem takové množiny může být kulová plocha zadaná implicitní rovnicí

$$Q(P) = x^2 + y^2 + z^2 = r^2.$$

Volbou konstanty r^2 se volí v prostoru jednotlivé izoplochy, což jsou místa, kde implicitní funkce nabývá zvolené konstantní hodnoty. Modelovací možnosti jednoduchých ploch jsou omezené, ale jejich kombinací se dají vytvářet tvarově bohaté modely. Základem modelování je tzv. *kostra* (skeleton). Prvky kostry, ze kterých se kostra skládá jsou tzv. *generátory*. Pro všechny body P v okolí generátoru je definována funkce $d = d(P)$, která určuje vzdálenost bodu P od generátoru, pro který je dále definována *potenciálová funkce* $F(d)$, určující vliv generátoru na místa ve shodné vzdálenosti d. Principem modelování tedy je používání jednoduchých prvků kostry jako jsou body, úsečky, polygony či části křivek. Poté se vyhodnotí jednotlivé potenciálové funkce(např. jednoduchá potenciální funkce 2.3) a kombinací jednotlivých vlivů generátorů vznikne výsledná izoplocha. Viz. [7].

$$F_a(d) = (1 - fracd^2R^2), d \leq R, \tag{2.3}$$

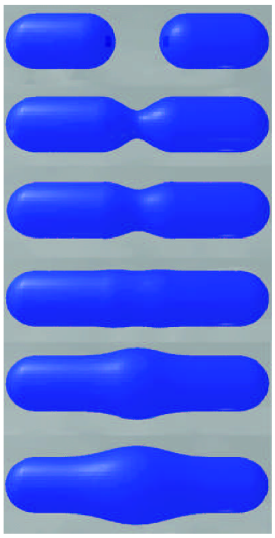
kde d je vzdálenost bodu od generátoru a R poloměr vlivu generátoru.

Potenciálová funkce závisí jen na vzdálenosti v prostoru a prvek kostry určuje tvar izoploch. Každý prvek kostry vyplňuje prostor skalárním polem $F^i(d^i)$, jenž přiřazuje každému bodu P v prostoru hodnotu $F^i(d^i(P))$ a celková intenzita v bodě P tedy je:

$$F(P) = \sum_{i=1}^{n} c_i F_i(d_i), \tag{2.4}$$

kde c_i je skalární hodnota určující vliv i-tého prvku kostry v bodě P v prostoru na celkovou hodnotu funkce.

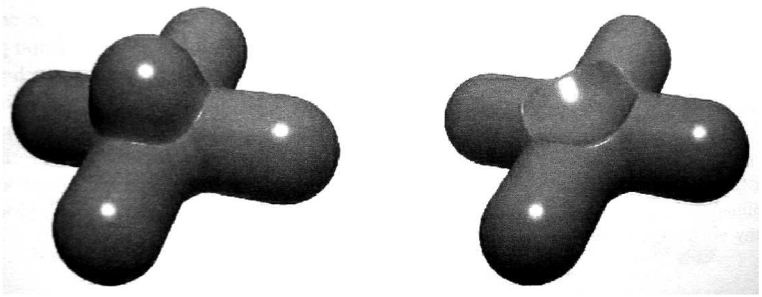
Izoplocha samozřejmě nemusí být souvislá a může se jednat o více objektů(či naopak) v prostoru jak lze vidět na obrázku 2.23.



Obrázek 2.23: Postupné zvětšování parametru, který určí rozsah vlivu polí kolem dvou úseček (Obrázek od [1]).

Znaménko koeficient c_i určuje, jestli se má vliv jeho prvku přičíst či odečíst. Viz obrázek 2.24 ukazuje vliv kladného a záporného c_i na tvar plochy.

Určování vzdálenosti bodu P od prvků kostry se řídí pravidly, jež jsou definovaná pro zvolený druh generátoru. Pravidla se tedy mění pro různé prvky kostry.



Obrázek 2.24: Dvě úsečky v jejichž středu je koule s parametrem c_i kladným vlevo a záporným vpravo(Obrázek od [7]).

Závěrem lze říci, že modelování pomocí kostry má mnoho nesporných výhod. Kostra je značně intuitivní aproximací spousty objektů reálného světa. Je snadno zobrazitelná. Tedy modelování je přehledné. Kostra není plnohodnotná reprezentace, ale na jejím základě lze získat přibližnou představu o modelovaném objektu. A stejně jako u CSG složité těleso vzniká skládáním těles jednoduchých. Kostru je též možné sdružovat do bloků a určovat jejich vzájemné působení. Viz. [7].

Zobrazení

Nejjednodušší metodou je zobrazovat implicitní tělesa pomocí metody sledování paprsku. Tato metoda je však časově značně náročná. Jinou metodou může být převedení implicitní plochy na jinou reprezentaci. Nejčastěji jde o převod na rovinné plošky, tedy o *polygonizaci implicitních ploch.*

Kapitola 3

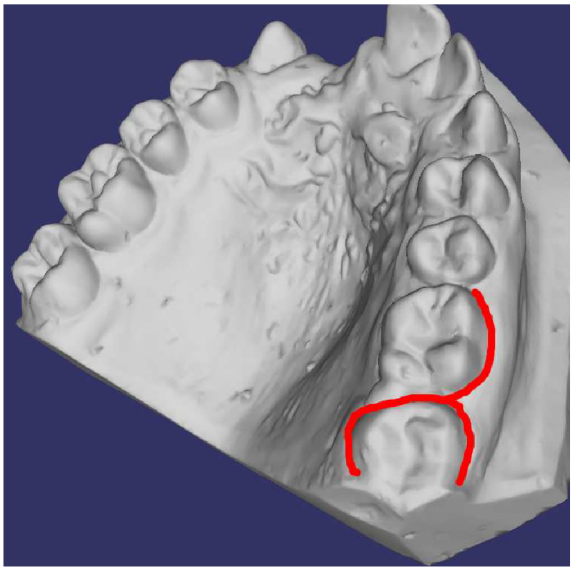
Návrh implementace

V této části bych se rád zabýval teoretickou stránkou implementace. Nejprve je zde analyzován řešený problém. Potom je zde nastíněna základní struktura aplikace. Dále zde popisujit jak by se měl vstupní model načítat a výsledek ukládat. Potom popisuji, jak by mělo vypadat uživatelské rozhraní a na závěr uvedu metody pomocí, kterých by se měly detekovat zuby.

3.1 Analýza řešeného problému

Jak již víme, cílem této práce je detekovat jednotlivé zuby na 3D polygonálním modelu čelisti.

Abych věděl jak na to, je třeba se nejprve podívat na daný model čelisti a jeho charakteristické znaky. Na první pohled je vidět, že zuby vystupují z čelisti jako samostané celky, které jsou od čelisti, jak je zvýrazněno na obrázku 3.1, odděleny změnou křivosti povrchu. Různě zakřivený povrch je samozřejmě po celém povrchu čelisti, ale zuby jsou takzvané největší *výstupky* čelisti.



Obrázek 3.1: 3D polygonální model čelistí s názorným zvýrazněním kde začínají zuby.

Na základě toho by se tedy daly podle křivosti povrchu vyhledat hrany kolem zubů a určit podle toho, kde zuby jsou.

Nebo podíváme-li se na model, vidíme, že jednotlivé druhy zubů (stoličky, tesáky, atd.) si jsou vlastně podobné. Je tedy možné vzít ukázkové vzory jednotlivých druhů zubů a porovnat je s jednotlivými částmi modelu a ty části, které se přibližně rovnají označit za zuby.

Nebo také můžeme říci, že model je členěn na jednotlivé samostatné části, které lze detekovat na základě členění modelu čelisti do smysluplných částí. Toho lze samozřejmě jen dosáhnout na základě analýzy křivosti povrchu modelu. A touto metodou se bude dále zabývat má práce.

3.2 Struktura aplikace

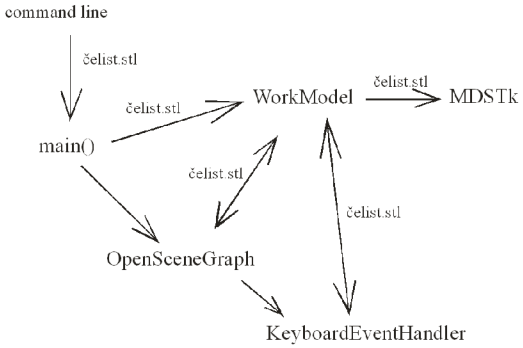
Základem aplikace bude spouštěcí funkce main. Ta bude mít za úkol příjmout argument z příkazové řádky. Tento argument by měl obsahovat jméno STL souboru, který bude dále načítán a zpracováván.

STL soubor, který obsahuje model s čelistí, se tedy načte pomoci třídy *WorkModel*, která bude mít dále na starost veškerou práci s modelem. K tomu by jí měl pomoci toolkit MDSTk a hlavně jeho balík VectorEntity, který obsahuje veškeré metody pro načtení a práci s polygonálními modely. Výsledné detekované zuby bude možné ukládat zpět v STL formatu a to buď jednotlivě, nebo dohromady.

Main funkce tedy vytvoří třídu *WorkModel* a potom pomocí toolkitu OpenSceneGraph vytvoří prohlížeč, ve kterém se bude zobrazovat model z třídy *WorkModel*.

K práci s modelem a jeho manipulací se bude dále využívat třídy *KeyboardEventHandler*, pomocí které se budou zobrazovat specifická nápověda aplikace a dále bude zracovávat jednotlivé specifické příkazi klávesových zkratek. Tato třída je potomkem třídy *osgGA::GUIEventHandler* z balíku OpenSceneGraph.

Přibližná struktura aplikace je graficky znázorněná na obrázku 3.2.



Obrázek 3.2: Grafický návrh struktury aplikace.

3.3 Načítání modelu

Jak již bylo uvedeno výše, model bude načítán a výsledky ukládány do formátu souborů STL. A to také proto, že tento formát je velmi jednoduchý. Formát STL obsahuje ideální

trojúhelníkovou polygonální reprezentaci jednoho modelu bez zbytečných definic materiálu, barvy atd. Navíc je formát STL kompaktibilní s velkým množstvím standardních 3D editorů

3.3.1 Popis formátu STL

Jedná se o formát binárních STL souborů. Skládá se z 84 bytové hlavičky a obsahuje záznamy jednotlivých polygonů. Každý záznam je 50-ti bytový.

Pro ověření správnosti modelu existuje více technik, ale tou základní je zjištění velikosti souboru, která by měla být rovna $84 + 50 \cdot n$ bytů, přičemž n je počet polygonů uložených v souboru. STL formát má i svojí ASCII variantu. Datá zde uložená jsou tedy čitelná i obyčejným textovým editorem. Velikost výsledného souboru je ale několikanásobně větší než v případě binární verze.

3.3.2 Způsob načítání a uložení výsledků modelu

Načtení modelu ze souboru a následné uložení výsledků obstarají metody balíku MDSTk. Hlavně se využijí metody tříd *mds::mod::CFileChannel* a *MCTriS*. První jmenovaná nastaví spojení se souborem a druhá daný model pomocí metod LoadSTL a SaveSTL načte nebo uloží. Před uložením souboru bude třeba, aby uživatel zadal na vyzvání do příkazové řádky jméno ukládaného souboru.

3.4 Uživatelské rozhraní

Pro praktické použití programu je potřeba vytvořit jednoduché uživatelské rozhraní, které by umožnilo potřebnou manipulaci se scénou.

Základní ovládání programu tedy bude realizováno pomocí myši, jejich tlačítek, několika dalších klávesových zkratek a příkazové řádky, kam se za prvé budou zadávat vlastní parametry, které ovlivňují detekci a za druhé zde bude program vypisovat svůj stav a případné běhové chyby.

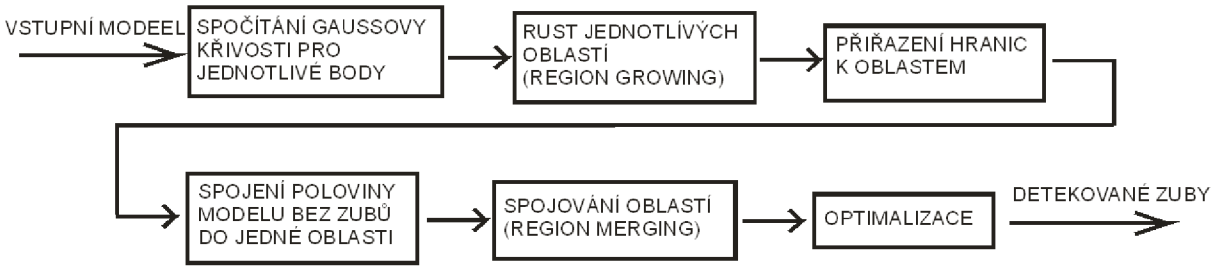
Ovládání scény pomocí myši zabezbečuje třída *osgProducer::viewer* z balíku OpenSceneGraph. Ta by měla zaručovat na základě pohybu myši adekvátní transformace scény. Dále má vlastní klávesové zkratky, které zobrazují nápovědu či nějakým způsobem mění mód zobrazení modelu. V nápovědě se lze potom dozvědět vše o funkcích jednotlivých kláves.

Mé specifické klávesové zkratky se budou zachytávat a vykonávat za pomocí třídy *KeyboardEventHandler*. Bude se jednat o několik klávesových zkratek pro zobrazení vlastní nápovědy, změny parametrů a klávesové zkratky spouštějící samotnou detekci.

3.5 Metoda detekce zubů

Má navrhovaná metoda detekce zubů je založená na segmentaci modelu na smysluplné části na základě analýzy křivosti povrchu. Metoda je kombinací metod od [16] a [2]. Z metod zmíněných v článcích [16] a [2] vybírám jednotlivé části metod, které se mi nejvíce hodí do mé koncepce a zejména z pohledu snadné a srozumitelné implementace. Obě metody mají společné to, že základem analýzy je Gaussova křivost, na základě které se rozdělí body modelu na hraniční vrcholy a semínka, z kterých se vytvoří jednotlivé části.

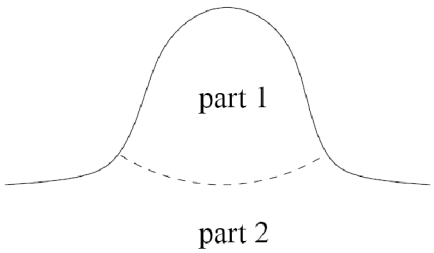
Metodu detekce shrnuje následující graf3.3:



Obrázek 3.3: Grafické znázornění metody detekce zubů.

3.5.1 Detekce hranic jednotlivých částí modelu pomocí Gaussovy křivosti

Cílem této metody je určit pomocí analýzy křivosti, které vrcholy trojúhelníkové sítě modelu jsou a které nejsou hraniční. Na základě hraničních vrcholů se pak model rozdělí na části jak je znázorněno na obrázku 3.4



Obrázek 3.4: Nalezení hranice mezi dvěma částmi modelu.

V modelu se tedy počítá pro každý vrchol proměná K(rovnice 3.1), která je diskrétní hodnotou Gaussovy křivosti. Je-li potom $K > 0$ pro daný vrchol p na povrchu S, je povrch eliptický. Pro $K = 0$ parabolický a pro $K < 0$ hyperbolický. Jak píše [2], povrch každé individuální části modelu má eliptické nebo parabolické chování. Proto můžeme rozdělit model na disjunktní části pomocí detekce hraničních vrcholů s hyperbolickým chováním. To vše je možné pouze u modelu typu emphmanifold.

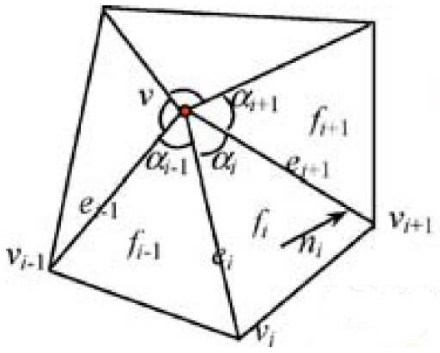
Diskrétní hodnota Gaussovy křivosti K se vypočítá pro vrchol v v polygonální síti pomocí následujícího vztahu 3.1. Kde α_i je úhel u v (viz. obrázek) a A_i je obsah příléhají plochy kolem v.

$$K(v) = \frac{3(2\pi - \sum_{i=1}^{n} \alpha_i)}{\sum_{i=1}^{n} A_i}, \tag{3.1}$$

Lze tedy jednoduše určit zda je vrchol hyperbolický nebo ne.

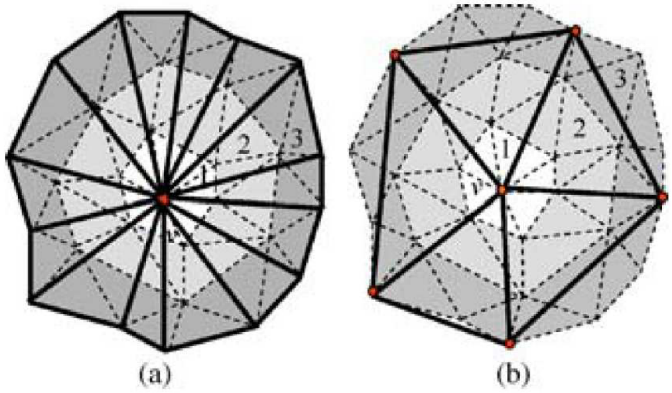
Potom je potřeba určit pro jak moc negativní hodnotu $K(v)$ určí práh, zda je či není vrchol hraniční. Práh se určuje zvlášť pro každý 3D model a to podle typu modelu a jeho hustoty sítě. Poté se izolované vrcholy, které mají celé okolí jiné(hraniční či semínka) změní na okolí.

Takto lze hraniční body získat efektivně pouze pro menší modely s malou hustotou polygonální sítě. U větších 3D modelů pak nastane problém, poněvadž se metoda stane



Obrázek 3.5: Vrchol v a jeho atributy pro sousedy v jednom okruhu.

nepřesnou a model může být rozdělen příliš. Proto [2] zavádí rozšířené multiokruhové sousedy, tzv. *eXtended multi-ring (XMR) neighborhood*. Jde o rozšíření okolí kolem vrcholu v, aby se mohli získat jeho přesnější atributy. Obrázek to objasní lépe 3.6(a).



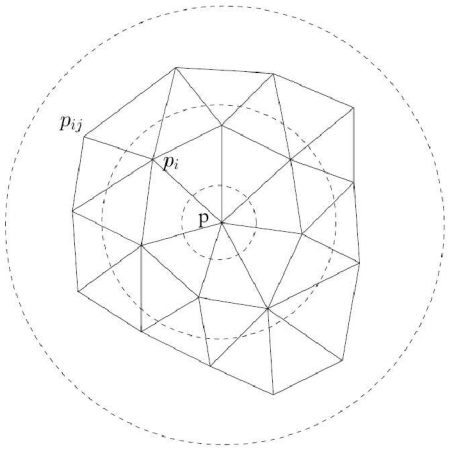
Obrázek 3.6: (a)XMR neighborhood 3 úrovně pro vrchol v; (b) zjednodušený XMR neighborhood pro vrchol v.

Problémem *(*XMR neighborhood) je ale jeho zřetelná vyšší výpočetní náročnost. Proto se daná metoda zjednodušuje(obr.3.6(b)). Já budu počítat okolí vrcholu v maximálně do 3. úrovně a nebudu tedy toto zjednodušení používat. Vice se o tom lze dozvědět na [2].

3.5.2 Růst oblastí ohraničených hraničními vrcholy

Tato metoda je převzatá od [16]. Metoda růst oblastí, tzv. *region growing*, je aplikována na každý označený vrchol, který není hraniční. Pro ilustraci obrázek 3.7 znázorňuje trojúhelníkovou síť druhé úrovně kolem bodu p.

Region growing pracuje následujícím způsobem. Začína se ve vrcholu p označeném jako semínko. Nejprve je k tomuto vrcholu přiřazeno nejbližší okolí první úrovně. Poté jsou všichni sousedi p_i, kteří jsou označeni jako semínka, označeny jako oblast, která je stejná jako p. Stejný proces označení pak pokračuje pro vrcholy p_i a jejich sousedy p_ij. Proces je ukončen, když je rostoucí oblast obklopena hraničními vrcholy. Proces se opakuje pro



Obrázek 3.7: Dvou úrovňová trojúhelníková síť kolem bodu p, který je vrcholem označen jako semínko. p_i reprezentuje sousedící vrchol bodu p a p_ij reprezentuje sousedící vrchol bodu p_i.

všechny vrcholy označené jako semínka. Ne však ty vrcholy, které už byly přiřazeny k nějaké oblasti. Tato metoda je tedy podobná metodě semínkového vyplňování.

3.5.3 Přiřazení hraničních vrcholů k oblastem

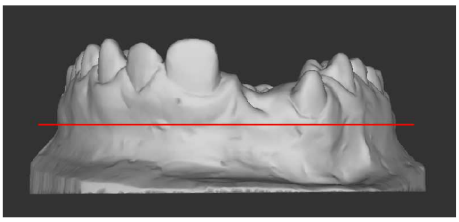
Spíše optimalizační metodou popsanou v [16] je odstranění hraničních vrcholů a jejich přiřazení k jednotlivým oblastem. Máme hraniční vrchol x a jeho sousedy x_i, které jsou seřazeny podle Euklidovské vzdálenosti k vrcholu x. Sousedící vrchol x_i, který je nejblíže k vrcholu x a je již zahrnut v nějaké oblasti, se vezme a vrcholu x se přiřadí stejná oblast jako má právě nejbližší x_i. Tak eliminujeme všechny hraniční vrcholy a měly by zbýt pouze vrcholy, které jsou přiřazeny k různým oblastem.

3.5.4 Zjednodušení 3D modelu pomocí jeho rozdělení na dvě části

Cílem mé metody je zjednodušit 3D model a zbavit se oblastí, ve kterých se nemohou nacházet zuby. Model je, jak je ilustračně znázorněno na obrázku 3.8, rozdělen na dvě části. Přičemž 3D model může být rozdělen v libovolném poměru dle potřeby, abych eliminoval co nejvíce oblastí kolem zubů, ale ne samostatné zuby. Všechny oblasti, které alespoň částečně zasahují do jedné poloviny modelu jsou potom sloučeny do jedné velké oblasti. Samozřejmě v případě určení špatné části modelu, kde se slučuje, půjde pomocí klávesové zkratky změnit tuto část na opačnou. Metoda by totiž měla určovat část oblasti, která se zjednodušuje jako tu, kde je méně bodů. Nastaví-li se ale poměr dělení, tak že na část, kde jsou zuby, zbyde příliš velká část, tak se určí oblast na zjednodušení špatně.

3.5.5 Spojování oblastí

Spojování oblastí, tzv. *region merging*, je metoda popsaná v [2], která je určená k optimalizaci příliš rozděleného povrchu. Použije se jednoduchý algoritmus spojování oblastí, který je založený pouze na dvouch kritérijích.

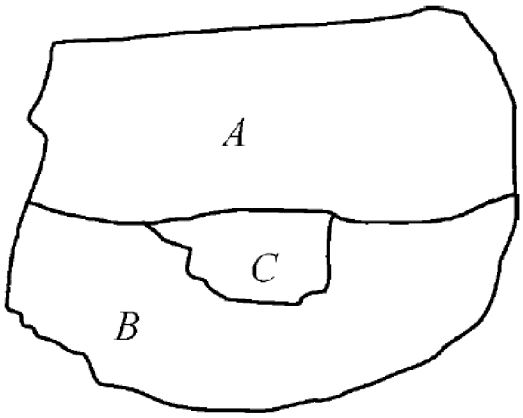


Obrázek 3.8: Rozdělení modelu na dvě části.

Prvním kritériem je velikost oblasti, je totiž jasné, že větši oblast má větší váhu než ta menší. Určení přibližné velikosti oblasti je v mém případě triviální. Počítá se jako počet vrcholů v dané oblasti. Více vrcholů znamená v mém případě větši oblast a obráceně. Výpočty jednotlivých ploch by totiž bylo výpočetně náročnější. Potom se už jen nastaví práh, který určuje podle počtů vrcholů v oblasti, zda se oblast spojí s větší oblastí.

Druhým kritériem je kritérium, které určuje, ke které oblasti se spojí kandidát na spojení. To se urči podle vzdálenosti hranice z okolními oblastmi. Vybere se potom ta oblast, se kterou mám spojovaná oblast nejdelší společnou hranici. V mém případě se tato metoda aplikuje opět pouze na základě počtu sousedících vrcholů, které patří do jiné oblasti.

Obrázek 3.9 ukazuje názorný příklad metody.



Obrázek 3.9: Malá oblast bude spojena s tou sousedící oblastí, se kterou má nejdelší společnou hranici. V tomto případě oblast C bude připojena k oblasti B.

3.5.6 Optimalizace

Nejduležítější optimalizací je opakování metody *region merging*. Lze totiž určit práh minimální velikosti připojované oblasti. Ta se na začátku určí o něco menší a potom se celková nejmenší velikost oblasti dožene postupným opakováním metody a násobením počátečního prahu minimální velikosti oblasti. Na počátku operace spojování se totiž nejprve zjistí velikosti oblastí, které se už později neaktualizují. To má za následek, že než dojde u oblasti na test zda má být spojena a podle testu je pod prahem minimální velikostí, tak ve skutečnosti už k této oblasti byla připojena oblast jiná a prah minimální velikosti je překrocen. Opakováním metody *region merging* s nižším počátečním prahem se

tato chyba minimalizuje. Je však vykoupena vyšší početní náročností, která závisí právě na počtu opakování.

Další optimalizace se týka způsobu zahrnování hraničních vrcholů do oblastí. Jde o to, že standartní metoda je navržena tak, aby byla co nejrychlejší. Vzdálenost mezi vyšetřovaným vrcholem x a jeho okolními body počítá i s okolními body, které byli právě přiřazeny k nějaké oblasti. Optimalizace však s čerstvě přiřazenými body nepočítá a zařazuje je mezi body oblastí, se kterými se počítá až v dalším cyklu. Metoda se tím ale značně zpomaluje a výsledek podle testů není nijak zvlášť působiví.

3.5.7 Křivost povrchu

Křivostí povrchu nebo také *Gaussova křivost povrchu* nijak nesouvisí s křivostí křivky, ale jde o vnitřní geometrickou vlastnost. Říká, že je jedno jak moc je daný povrch ohnutý a to dokavaď není překroucený, napjatý či stlačený.

K získání lepší představy o tom jak křivost funguje uvedu několi příkladů.

Povrch, který je celý vypoulen, jako například koule je *zkřiven pozitivně*. Hrubým otestováním, že daný povrch je křivý pozitivně, je že ať se vezme jakýkoli bod na povrchu, tak jakákoli plocha dotýkající se v daném bodě povrchu, se ho dotýká tak, že celý povrch leží kromě daného bodu dotyku na jedné straně plochy.

Kus papíru nebo povrch válce má naproti tomu křivost nulovou.

A sedlový tvar povrchu má křivost negativní, protože každá plocha jdoucí skrz bod protíná sedlový tvar ve dvou nebo více místech.

Viz také na [3].

Kapitola 4

Implementace

Úvodem implementace bych rád uvedl použité technologie, které mi usnadnily práci s aplikací. Dále popíšu implementaci jednotlivých nezbytných částí, které byly užity a uvedu důležité části některých algoritmů.

4.1 Užité technologie

Dnes existuje mnoho prostředků pro vytvoření aplikace pracující s 3D modely. Liší se však svojí kvalitou, cenou a službami, které nabízejí. V akademickém prostředí je tedy nejvhodnější použít volně šiřitelné open source aplikace, které zaručují i díky své licenci GPL neomezené použití. Akorát je po té povinnost také zveřejnit zdrojové kódy výsledné aplikace spolu se spustitelným programem. Tyto nástroje mají mnoho výhod. Je možné je dle potřeby upravovat a rozšiřovat, dále většinou nebývají závislé na konkrétním operačním systému, takže zaručují dobrou kompatibility. Stačí tedy pod jiným operačním systémem, který je podporován těmito užitými nástroji, zkompilovat zdrojový kód a program by měl pracovat stejně jako v prostředí operačního systému, pod kterým byl původně vyvíjen.

4.1.1 Programovací jazyk C++

Užití tohoto programovacího jazyka vyplývá již ze zadání, ikdyž zde byla možnost, užít i jiný programovací jazyk.

Pro C++ však hovoří jeho nesporné výhody. Hlavně jeho masové rozšíření i mezi mnohými profesionálním systémy. Dále jeho dostupnost a to mnoha jeho implementací a také jeho dokumentace. Nesmí se ani opomenout na jeho dobrou přenositelnost mezi různými operačními systémy a architekturami.

Protože já pracuji hlavně v operačním systému MS Windows, zvolil jsem si tedy distribuci MinGW, která je pro tento operační systém ideální. Pro zjednodušení práce a automatizaci překladu je užit MSYS(Minimal SYStem), který umožňuje používání skriptů a tedy užití programu MAKE, který umožňuje vytváření inteligentních překladových skriptů.

4.1.2 Toolkit MDSTk

Medical **D**ata **S**egmentation **T**ool**k**it je kolekcí zobrazení a množství data zpracovávajících nástrojů zaměřených na lékařské segmentační zobrazení. To je důvod, proč se toolkit jmenuje MDSTk. Základní část toolkitu je veřejná a zdarma. Hlavními tvůrci toolkitu jsou Ing. Michal Španěl a Ing. Přemysl Kršek, Ph.D.

MDSTk je navržen jako vysoce modulární nástroj. Obsahuje mnoho oddělených modulů, přičemž každý zajišťuje nějakou funkci či algoritmus. Jednotlivé moduly jsou propojeny použitím jednoduchých kanálů.

MDSTk dále obsahuje několik souborů knihoven. V mé aplikaci se pracuje hlavne se souborem knihoven VectorEntity, který usnadňuje práci s 3D počítačovými polygonálními modely. Obsahuje knihovny, které zajišťují práci s modely ve formátu STL, a to jejich import i export. Dále obsahuje množství funkcí pro práci s jednotlivými polygony, hranami a body.

Více také na [6].

4.1.3 OpenSceneGraph

OpenSceneGraph je open source na všech platformách běžící toolkit, používaný pro vývoj vysoce výkonných grafických aplikací jako jsou letecké simulátory, hry, virtuální reality a vědecké vizualizace. OpenSceneGraph je objektově orientovaný systém založený na OpenGl, tím odprošťuje vývojáře od nutnosti implementovat a optimalizovat grafické příkazy nižšího stupně, a zajišťuje mnoho dalších nástrojů pro rychlý vývoj grafických aplikací.

Má aplikace právě využívá služeb OpenSceneGraphu pro zobrazení scény s 3D modelem a ukázky detekce hran na načteném 3D počítačovém polygonálním modelu čelisti. K práci s modelem slouží jednoduché rozhraní klávesových zkratek a užití myši.

Více také na [8].

4.2 Spouštění aplikace z main() a její funkce

Jak již bylo řečeno dříve , program se spouští z příkazové řádky, kde se zadává jediný parametr a to buď jmeno STL souboru s načítaným modelem anebo parametr -h či -help. Parametry příkazové řádky dále zpracovává třída *osg::ArgumentParser* z balíku OpenSceneGraph(OSG). Pak se vytváří třída *osgProducer::viewer*, též z balíku OSG. Tato třída má poté na starosti okno, ve kterém se zobrazuje model. Model je do *osgProducer::viewer* uložen jako objekt třídy *osg::Node*. Předtím je, ale ješte model pomocí metody třídy *osgUtil::Optimizer* optimalizován, aby se odstranili případné redundantní uzly a stavy. Poté třída *osgProducer::viewer* vytváří okno, ve kterém je zobrazen model, a na závěr spouští smyčku, ve které se scéna animuje na základě příkazu myši či příkazové řádky.

4.2.1 Funkce model()

Tato funkce vytváří prohlížený model. V atributech obsahuje všechny volitelné parametry a řídí tak vlastně celou práci. Model je načítán za pomocí balíku MDSTk a vrácen je jako objekt třídy *osg::Group*, která je potomkem třídy *osg::Node*, jež ukládá model do okna.

Pomocí metody *loadTooth()* objektu *WorkModel* se model načte. Dále metoda *getTooth()* té samé třídy načte polygonální model do objektu *vctl::MCTriS* z balíku VectorEntity. Dále se za pomocí metod balíku VectorEntity indexují jednotlivé vrcholy polygonálního modelu. Pak se vytvoří vektory třídy *osg::Vec3Array* pro uložení bodů polygonu a jeho normál. Potom se vytvoří třída *osg::DrawElementsUInt*, u které se nastaví metoda vykreslování na trojúhelníky a podle toho se do ní uloží indexy jednotlivých trojúhelníků. Pak se vytvoří objekt *osg::Geometry* a u něj se nastaví pole vykreslovaných bodů, normál a indexů. Ten se přes objekt *osg::Geode* uloží do *osg::Group*.

Ve funkci je oddělená část kódu podmínkou *if*. Tato část je vstupem k samotné detekci zubů. Je zde speciální část, která vytváří na základě prvního počtu samostatných oblastí vytvořené metodou *region growing* vektorové pole barev třídy *osg::Vec4Array*. Barva je pak podle atributu *flag* přidělována jednotlivým trojúhelníkům a hranice mezi oblastmi je vyznačena pro přehlednost řadou bílých trojúhelníků. Hlavně se zde podle atributu *gauss* určuje jaká se použije funkce pro analýzu křivosti povrchu. Je zde výběr mezí výpočtem diskrétní Gaussovy křivosti vrcholu v se zahrnutím okolí 1. úrovně, 2. úrovně nebo 3. úrovně. Dále se zde spouští metoda *RegionGrowing()* patřící třídě *WorkModel* stejně jako předchozí metoda vypočítávající Gaussovu křivost. V metodě *RegionGrowing()* se potom spouští všechny ostatní funkce, které aplikace obsahuje, například ukládání.

Do objektu *osg::Group* se pak podle nastavených parametrů přidávají objekty třídy *osg::Node* jako jsou nápověda nebo zobrazení vrcholů, které byly na základě analýzy křivosti určeny jako hraniční.

4.2.2 Funkce makeVertices()

Funkce zpracovává objekt *vctl::MCVerticeS* balíku VectorEntity, který je kontejnerem vrcholů. Zobrazují se pak pouze ty vrcholy, které jsou označeny atributem *flag* jako hraniční. Vrací pak uzel objektu *osg::Node* s načtenými vrcholy.

Tato funkce funguje stejně jako funkce *model()*. Akorát se ve třídě *osg::DrawElementsUInt* nastaví metoda vykreslování na body a tomu se přizpůsobí indexování vrcholů. Navíc se ještě vytvoří vektor třídy *osg::Vec4Array* pro uložení barvy a ta se aplikuje stejně pro všechny vrcholy.

4.2.3 Funkce createHUD()

Tato funkce vrací uzel objektu *osg::Node* a pomocí tohoto objektu se zobrazuje nápověda. Ta vysvětluje význam vlastních příkazu. Aktivuje se a deaktivuje se klávesovou zkratkou "j".

Pomocí objektu *osg::StateSet* se nastaví, aby se objekt *osg::Node* zobrazoval nad všemi objekty. Do objektu *osgText::Text* se poté nastaví text nápovědy, který se bude zobrazovat. Do tohoto objektu se také nastaví barva a pozice nápovědy.

Na závěr se vrací přes objekty *osg::Geode* a *osg::MatrixTransform* objekt *osg::Projection*, jenž je potomkem třídy *osg::Node*.

4.3 Třída WorkModel

Tato třída obsahuje vlastnosti a metody pro práci s polygonálním modelem jako jsou načítání 3D modelu, detekce zubů a ukládání výsledků. Třída využívá pro práci s polygonálními modely toolkitu MDSTk a zvláště jeho balíku VectorEntity.

Konstruktor přebíra parametr názvu STL souboru s modelem a inicializuje na něj vlastnost jména souboru. Mezi důležitější metody, které třída obsahuje, patří metoda *loadTooth()*, která načítá modelu. Další důležité metody jsou uvedeny níže.

4.3.1 GaussCurvature()

Tato metoda vypočítává diskrétní hodnotu Gaussovy křivosti pro vrchol v, přičemž se do výpočtu zahrnuje nejblížší okolí, tzv. okolí 1. stupně. Metoda má dále své modifikace

pro okolí 2. a 3. stupně. Aktivuje se klávesovou zkratkou "g" a zároveň se tak lze mezi jednotlivými modifikacemi metody přepínat.

Je zde implementován vzorec 3.1 pro výpočet diskrétní hodnoty Gaussovy křivosti. Poté se dle hodnoty $K(v)$ a volitelného prahu určí zda vrchol je hraniční. Je-li hraniční, označí se jeho atribut *flag* na hodnotu, která určuje, že je hraniční, jinak na hodnotu semínka.

Samotný výpočet je triviální. Postupně se prochází kontejner s vrcholy. U každého vrcholu se pak získají trojúhelníky, do kterých je zahrnut. Využijí se metody, které jsou definované balíkem VectorEntity, pro výpočet obsahu jednotlivých trojúhelníků a získání vrcholů naproti vyšetřovanému vrcholu pro výpočet úhlu. Všechny úhly a obsahy se u vyšetřovaného vrcholu postupně sčítají dlé vzorce 3.1. Poté se vypočte samotná diskrétní hodnota $K(v)$ a vrcholu se podle prahu přiřadí odpovídající hodnota *flagu*. Tak se postupuje dále s ostatními vrcholy, než se všechny projdou.

Modifikacé analýzy křivosti pro okolí 2. a 3. stupně jsou o něco složitější, protože se nejprve musí dopracovat k sousedům 2. a 3. stupně vyšetřovaného vrcholu. V případě 2. stupně se nejprve získají okolní vrcholy, potom se teprve z nich určují dvojíce vrcholů, které společně s vyšetřovaným vrcholem tvoří trojúhelník, u kterého se znovu vypočítá obsah a úhel. U 3. stupně je vše ještě o řád složitější.

Nakonec se jěště volá metoda *filter()*, která mění izolované vrcholy na jejich okolí. A to hraniční i semínka. Pro každý typ vrcholu se prochází kontejner s vrcholy zvlášť a u každého vrcholu, buď hraničního nebo semínka, se prochází jeho okolí a nemá-li žádný okolní vrchol stejný, je změněn na opačný typ vrcholu.

4.3.2 RegionGrowing()

Samotná metoda *region growing* je triviální. Její algoritmus se totiž podobá algoritmu semínkového vyplňování.

Prochází se znovu kontejner s vrcholy a ty vrcholy, jenž mají nastavenou hodnotu *flagu* na hodnotu odpovídající semínku, jsou zahrnovány do vyplňovaných oblasti. Využívá se zde rekurze a rekurzivní metody *segmentation()*, které se na začátku vždy zadá počáteční vrchol, od kterého danou oblast vyplňovat, a číslo oblasti. Metoda vždy změní hodnotu *flagu* na hodnotu odpovídající dané oblasti. Potom se zavolá pro všechny okolní vrcholy, které jsou semínka. Tedy vrcholy, ktere nejsou ještě přiřazeny žádné oblasti ani nejsou hraniční. Na konci jsou tedy všechny vrcholy rozděleny do oblastí nebo do hranic.

Tato metoda však v rámci funkce celé aplikace zajišťuje mnohem víc. Zastřešuje totiž celý zbytek detekce zubů až po uložení výsledků. Spouští se odtud další metody, které zpřesňují a optimalizují detekci zubů a na závěr se zde na základě podmínky *if* rozhoduje zda uložit jednotlivé detekované oblasti zvlášť nebo dohromady. Většina těchto oblastí by přitom měla obsahovat detekovaný zub, ale závisí na parametrech, jak přesně je zub detekován.

4.3.3 BoundarySeg()

Jde o další metodu, která je spouštěna metodou *RegionGrowing*. Má za úkol přiřadit již nepotřebné hraniční vrcholy k jednotlivým oblastem. Děje se tak ve smyčce, která probíhá tak dlouho dokuď nejsou všechny hraniční vrcholy přiřazeny. U každého vrcholu se vždy najdou jeho sousedi a zjistí se co jsou zač. Potom se vrchol přiřadí k nejbližšímu sousedovi, který už je v nějaké oblasti. Přiřazení probíha prostou změnou *flagu* na hodnotu odpovídající oblasti nebližšího souseda.

Jistou optimalizací zde je to, že lze pomocí klávesové zkratky "d" pozměnit způsob přiřazení hraničního vrcholu k oblasti. V každé smyčce se totiž ukládají v dané smyčce změněné vrcholy do pole a s těmito vrcholy není již možno počítat jako s vrcholy, ke kterým lze přiřadit hraniční vrchol a to i přesto, že už v nějaké oblasti zahrnuty jsou. Lze je použít až v dalším cyklu. Je to tak vyřešeno kvůli tomu, že se často detekuje třeba i polovina všech bodů modelu jako hraniční vrcholy a vznikájí tak rozsáhlé plochy tvořené pouze těmito vrcholy. A tato optimalizace by k jednotlivým oblastem měla tyto vrcholy přiřadit věrohodněji. Je ale podstatně výpočetně náročnější, protože se u každého vrcholu, který je již přiřazen k nějaké oblasti - to určí hodnota jeho *flagu*, v okolí vyšetřovaného hraničního vrcholu prochazí sekvenčně pole s již změněnými vrcholy v daném cyklu.

4.3.4 HalfRegionMerging()

Jde o jednoduchou metodu, která rozdělí pomoci metody *SimpleSide()* model na dvě části. U té se dá pomocí klávesové zkratky "n" měnit poměr dělení 3D modelu a pomocí "k" otočit stranu, se kterou se pracuje. Potom prochází v kontejneru s vrcholy, ty vrcholy, které jsou v té polovině modelu, v které nejsou zuby a mají se tedy spojit do jedné oblasti. Zde se vždy uloží do vektorového pole hodnota *flagu* oblasti, která je přiřazena k danému vrcholu, přičemž se v tomto poli sekvenčně kontroluje, aby se každá hodnota uložila pouze jednou. Po projití kontejneru s vrcholy se projde vektorové pole s uloženými *flagy* oblastí a všechny oblasti v něm obsažené se změní změnou hodnoty *flagu* vrcholu na oblast, která byla do pole uložena jako první.

4.3.5 RegionMerging()

Je další klíčovou metodou při detekci. Jejím cílem je rozumně spojit příliš roztříštěny 3D model do větších oblastí, které by hlavně obsahovaly celé zuby a nejen jejich části.

Na začátku metoda obdrží parametr určující minimální počet vrcholů v jedné oblasti. Nejprve metoda prochází postupně jednotlivé oblasti a u každé z nich prohledává celý kontejner s vrcholy a podle označení *flagem* počíta počet vrcholů, ten je poté pro každou oblast uložen ve vektorovém poli. Když je tato první část hotova, prochází se v cyklu postupně znovu všechny oblasti. Nejprve se vždy určí jestli je daná oblast větší či menší než dané minimum. Je-li tedy menší, tak se inicializuje a vynuluje pole, kam se bude ukládat počet sousedních vrcholů ze sousedících oblastí. Poté se prochází pro každou oblast kontejner s vrcholy a u těch, které patří do vyšetřované oblasti, se vyšetřují okolní vrcholy a podle toho, ke které oblasti patří, se to zaznamená do pole, kde se ukládají počty vrcholů sousedních oblastí. Dále se toto pole projde a najde se ta sousedící oblast k vyšetřované oblasti, jenž obsahuje nejvyšší počet sousedních vrcholů. Na závěr se u právě spojované oblasti znovu prochází kontejner s vrcholy a všechny *flagy* označující spojovanou oblast se změní na hodnotu náležící oblasti jež měla se spojovanou oblastí nejvíce sousedních vrcholů. To vše se pak celé provádí znovu pro další oblasti.

Tato metoda se může volat s postupným zvyšováním minimálního počtu vrcholů v oblastí zdůvodu optimalizace výsledků detekce. Minimální počet vrcholů lze měnit pomocí klávesové zkratky "m" a klávesová zkratka "x" umožnuje nastavit počet opakování metody. Opakování je zajištěno v metodě *RegionGrowing()* pomocí cyklu *for* a ukaždého dalšího cyklu se minimální počet vrcholů oblasti vynásobí aktuální iterací cyklu.

4.3.6 TriSeg()

Tato metoda má nastarost převést *flagy* vrcholů, které určují přislušnost vrcholů k jednotlivým oblastem, na *flagy* trojúhelníků, podle kterých je pak možné v prohlížeči graficky odlišit jednotlivě detekované oblasti, tedy zuby.

Nejprve se tedy projde kontejner s trojúhelníky a jejich *flagy* se pro jistotu nastaví na výchozí hodnotu. Potom se v cyklu projdou všechny oblasti a ukaždé se prochází kontejner s vrcholy patřící dané oblasti. U každého vrcholu se získají trojúhelníky, které obsahuje. Pak každý trojúhelník, který obsahuje zbylé dva vrcholy z téže oblasti, je této oblasti přiřazenen odpovídajícím nastavením svého *flagu*.

Výše popsaný algoritmus má za následek, že trojúhelníky, které mají sve vrcholy z různých oblastí, nebudou zařazeny do žádné oblasti a v prohlížeči budou zobrazeny jako bílé hranice, které oddělují jednotlivé oblasti.

4.3.7 Ukládání

Jak již bylo psáno víše, aplikace umožnuje ukládat výsledky dohromady nebo po jednotlivých zubech. To zajišťují metody *saveToothTogether()* a *saveToothSingle()*.

Ukládání funguje tak, že se nejprve určí, která oblast je nejvetší - to je čelist a neukládá se. Potom se v případě metody *saveToothTogether()* postupně prochází kontejner s vrcholy a u všech vrcholů, které nejsou podle *flagu* součástí největší oblasti, se získají trojúhelníky, kterých je součástí. Vrcholy těchto trojúhelníku se pak uloží do nového kontejneru na vrcholy a na něm se postaví nový kontejner trojúhelníku. Pak už se vezme uživatelem zadané jméno souboru kam se bude ukládat, připojí se k názvu připona typu souboru ".stl". Na závěr se použijí metody balíku VectorEntity pro ukládání do STL formátu.

U ukládání po jednotlivých zubech se postupuje podobně, akorát se ukládají jednotlivě po oblastech a tedy v cyklech, kdy na konci každého jména souboru je číslovka určující jeho pořadí.

Ukládání se aktivuje klávesovými zkratkami "e" - dohromady, a "r" - jednotlivě.

4.4 Třída KeyboardEventHandler

Tato třída zajišťuje vykonávání specifických klávesových zkratek aplikace. Je inicializována třídou *osgProducer::viewer* v základní funkci *main()*. Všechny volitelné parametry aplikace, které lze měnit se na počátku inicializují do výchozího stavu a potom se na základě klávesových zkratek zapínají či vypínají. Objekt *osgGA::GUIEventAdapter::KEYDOWN* detekuje stisknutí klávesy a na základě toho jakou klávesu vrátí metoda *getEventType()* třídy *osgGA::GUIEventAdapter*, tak takovou reakci vybere podmínka *if*. Zde se nejprve v okně příkazového řádku zobrazí aktuální nastavení všech parametru a poté se upraví parametr, kterého se klávesová zkratka týká. Ostatní parametry zůstanou nastaveny v takovém stavu, v jakém zrovna byly, a zavolá se funkce *model()* pro přepočítání modelu i se všemi aktuálně nastavenými parametry.

Kapitola 5

Testování implemetované metody

Zde uvedu ukázky detekce zubů. Z příkladů je vidět, že různé parametry mají vliv na rychlost a přesnost detekce. Šikovným nastavením parametrů pak lze v podstatě získat relativně přesné tvary všech zubů.

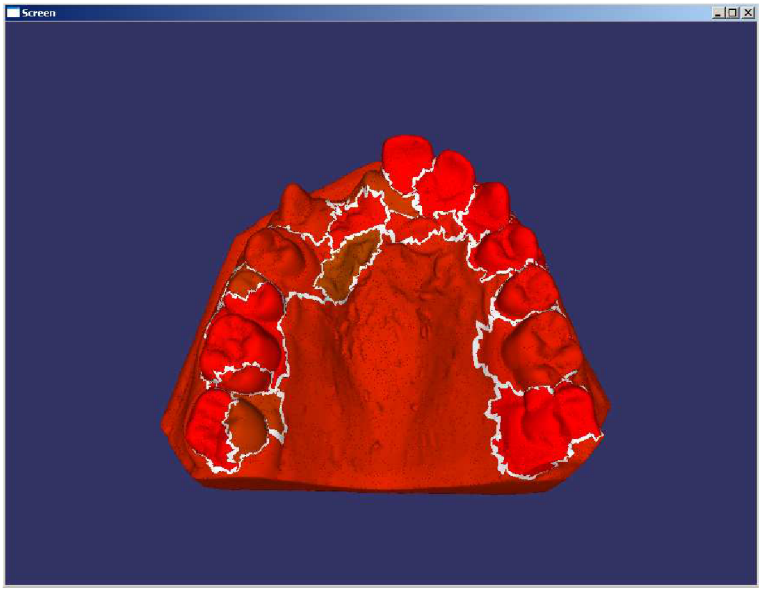
Jak již bylo výše psáno, tak k analýze křivosti 3D modelu je možno použít tři metody výpočtu diskrétní Gaussovy křivosti. Ta se pro každý vrchol vypočítává z okolí 1., 2. či 3. stupně. Níže uvedené výsledky ukázali, že asi nejlepši výsledky jsou dosahovány u vypočtu Gaussovy křivosti při zahrnutí okolí 2. stupně, viz. obrázky 5.1, 5.2, 5.3, 5.4, 5.5 a 5.6. Červené body, které jsou zobrazeny v modelu, jsou zobrazené hraniční vrcholy. Problém však je s prahém u Gaussovy křivosti, je-li totiž příliš málo bodů označeno jako hraniční vrcholy, tak metoda *region growing* obsahuje příliš mnoho rekurzivních volání a to vede až k pádu aplikace. Nejnižší možné prahy, kdy aplikace ještě funguje jsou pro první model -0.2684 pro 1. úroveň Gausse, -0.6026 pro 2. úroveň Gausse a -1.5352 pro 3. úroveň Gausse, a pro druhy model -0.1271 pro 1. úroveň Gausse, -0.2407 pro 2. úroveň Gausse a -0.8828 pro 3. úroveň Gausse. Dle testu je pro optimální výsledek detekce běžně 50 až 65 % vrcholů 3D modelu určeno jako hraniční vrcholy. Dle časové náročnosti se dá ještě říci, že čím vyšší stupeň okolí je zahrnut do výpočtu Gaussovy křivosti, tím vyšší časová náročnost, ale jde řádově o vteřiny.

Aplikace dále umožňuje například měnit rozdělení modelu na dvě části, jako ukazuje obrázek 5.7. Toho lze využít hlavně tak, že model je rozdělen těsně pod zuby, jako je tomu u 5.5 a 5.6.

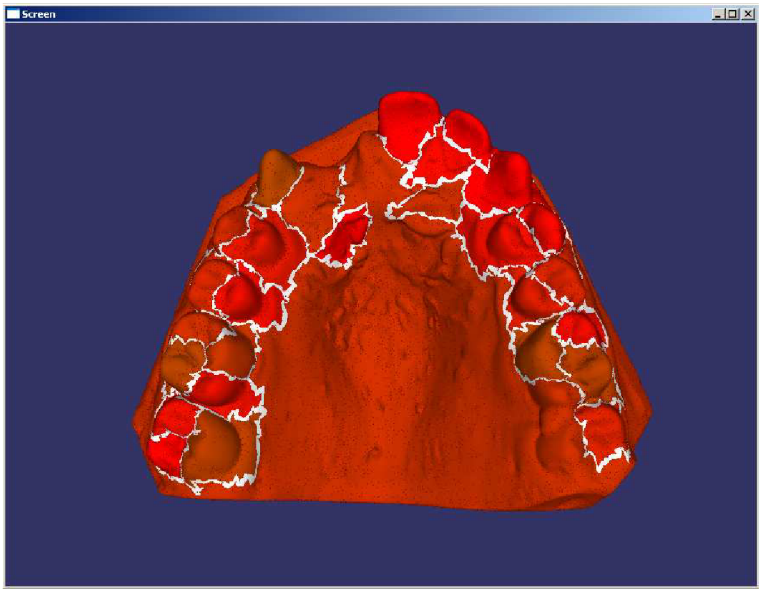
Parametry, které značně ovlivňují detekci, jsou potom minimální počet vrcholů v oblasti a počet opakování metody *region merging*. Přičemž čím vyšší minimální počet vrcholů, o to je třeba méně opakování. Dá se říci, že násobek těchto dvou parametrů by měl být pro testované 3D modely optimálně někde mezi hodnotami 200 až 300. Přičemž opakování aplikaci zpomalují více.

Poslední optimalizační parametr, tedy způsob zahrnutí hraničních vrcholů, prokázal, že někdy složitější výpočet výsledek vylepší, jindy ne(obr. 5.8 a 5.5). Avšak výpočet je vždy minimálně o 1 minutu pomalejší. Záleží tedy na 3D modelu a nastavení ostatních parametrů.

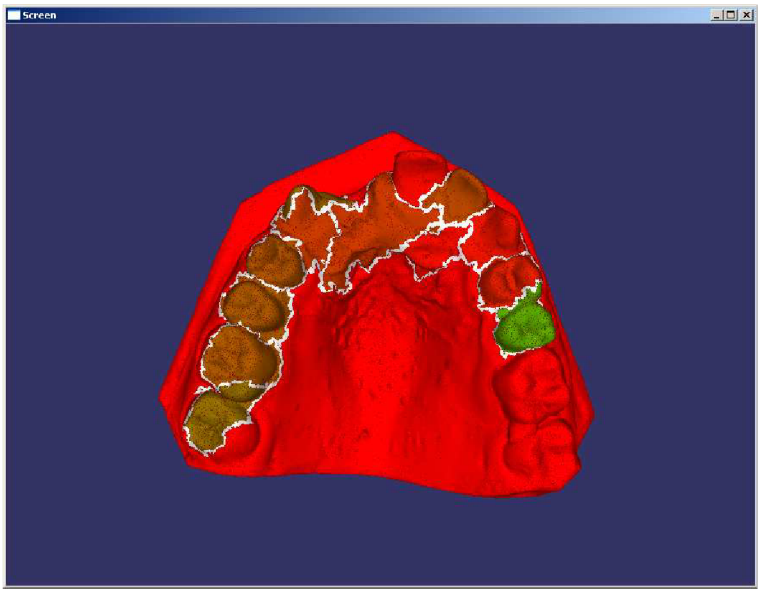
Nejlépe je ale vydět úspěšnost detekce na samostatně znovu načteném výsledku detekce - zvlášť či dohromady. To ukazují obrázky 5.9, 5.10, 5.11 a 5.12.



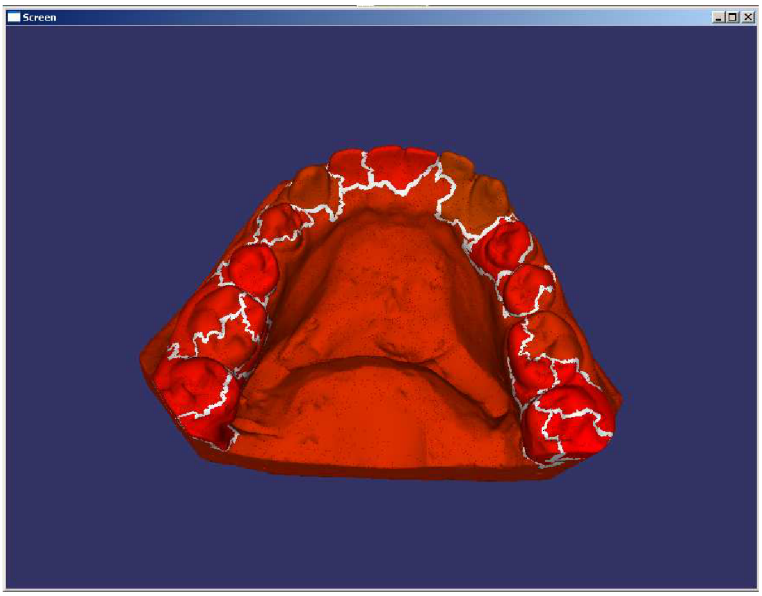
Obrázek 5.1: Ukázka detekce zubů, Gauss 1. stupně, práh nastaven na 0, dělení modelu 0.5, minimální počet vrcholů v oblasti 5, počet opakování *region mergingu* 50 a složitější přiřazení hraničních vrcholů. Doba výpočtu 2min 59s.



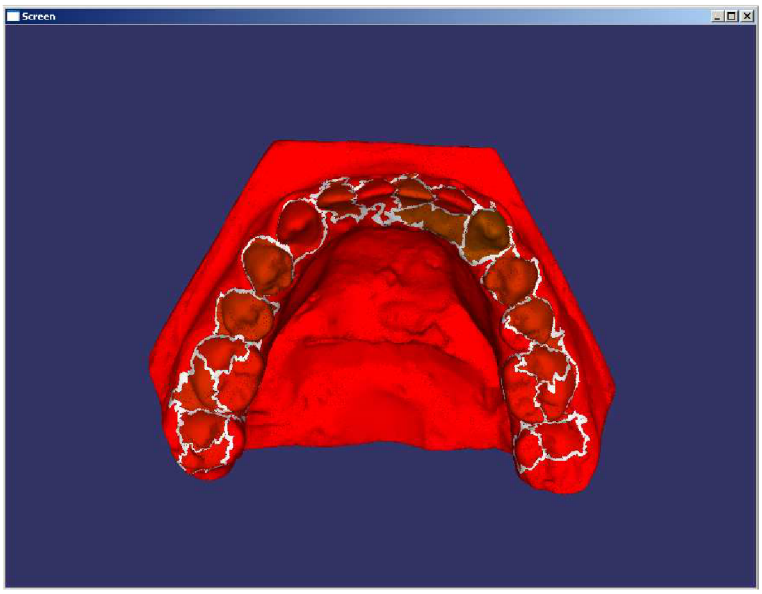
Obrázek 5.2: Ukázka detekce zubů, Gauss 2. stupně, práh nastaven na -0.015, dělení modelu 0.55, minimální počet vrcholů v oblasti 27, počet opakování *region mergingu* 15 a jednodužší přiřazení hraničních vrcholů. Doba výpočtu 17s.



Obrázek 5.3: Ukázka detekce zubů, Gauss 3. stupně, práh nastaven na -0.6, dělení modelu 0.55, minimální počet vrcholů v oblasti 15, počet opakování *region mergingu* 25 a jednodužší přiřazení hraničních vrcholů. Doba výpočtu 39s.

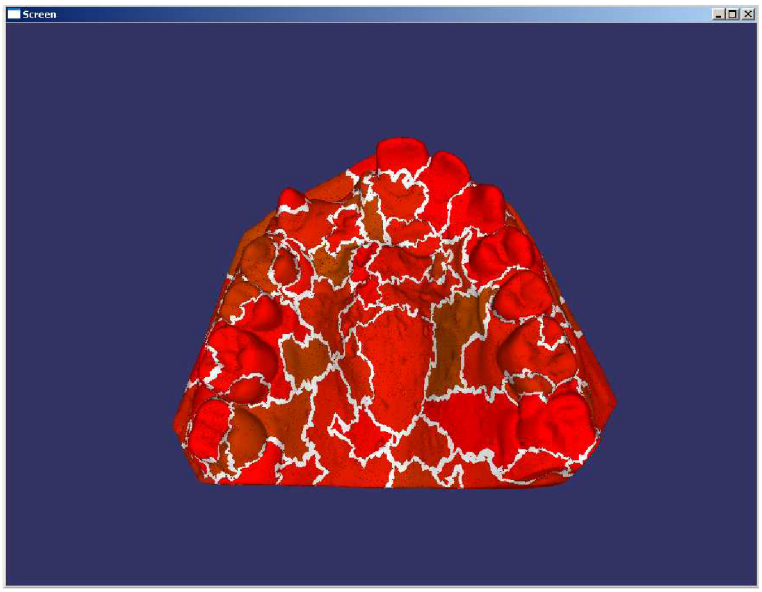


Obrázek 5.4: Ukázka detekce zubů, Gauss 1. stupně, práh nastaven na 0, dělení modelu 0.55, minimální počet vrcholů v oblasti 15, počet opakování *region mergingu* 20 a jednodužší přiřazení hraničních vrcholů. Doba výpočtu 48s.

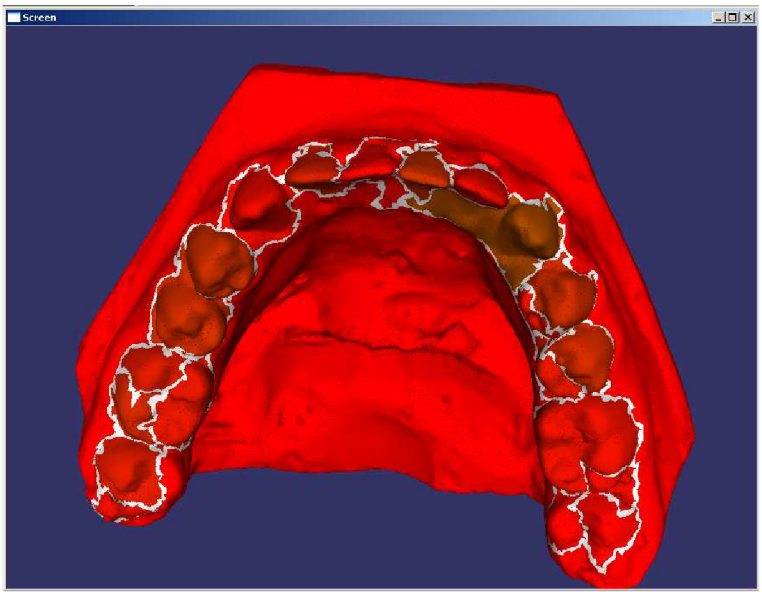


Obrázek 5.5: Ukázka detekce zubů, Gauss 2. stupně, práh nastaven na -0.02, dělení modelu 0.65, minimální počet vrcholů v oblasti 5, počet opakování *region mergingu* 30 a jednodužší přiřazení hraničních vrcholů. Doba výpočtu 23s.

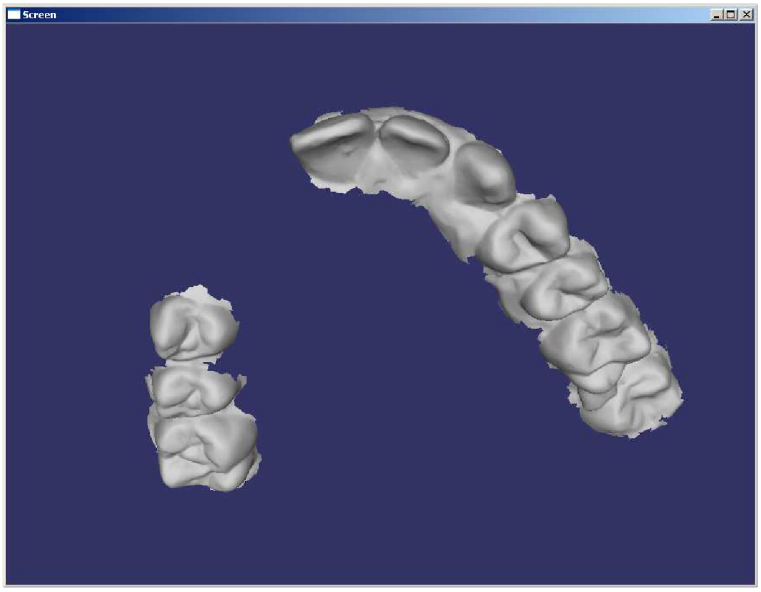
Obrázek 5.6: Ukázka detekce zubů, Gauss 3. stupně, práh nastaven na -0.45, dělení modelu 0.65, minimální počet vrcholů v oblasti 25, počet opakování *region mergingu* 10 a jednodužší přiřazení hraničních vrcholů. Doba výpočtu 18s.



Obrázek 5.7: Ukázka detekce zubů, Gauss 1. stupně, práh nastaven na 0, dělení modelu 1, minimální počet vrcholů v oblasti 70, počet opakování *region mergingu* 1 a jednodužší přiřazení hraničních vrcholů. Doba výpočtu 7s.

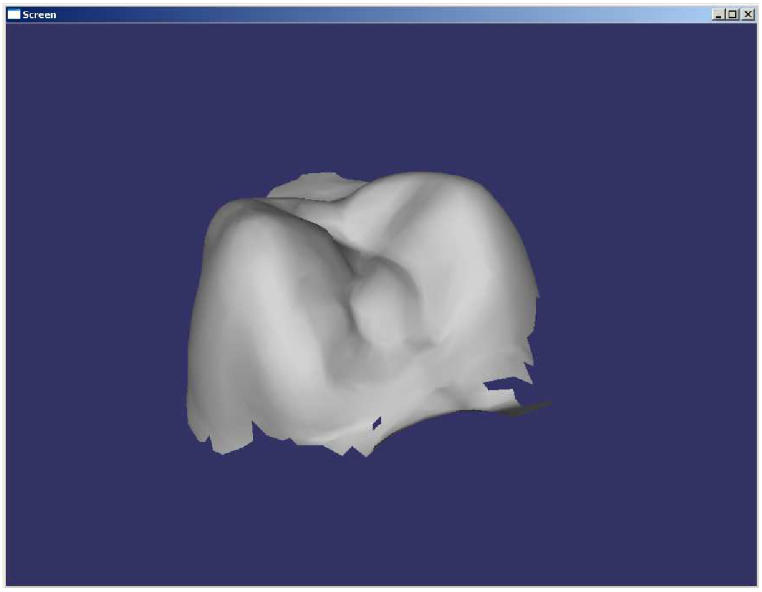


Obrázek 5.8: Ukázka detekce zubů, Gauss 2. stupně, práh nastaven na -0.02, dělení modelu 0.65, minimální počet vrcholů v oblasti 5, počet opakování *region mergingu* 30 a složitější přiřazení hraničních vrcholů. Doba výpočtu 1min 23s.

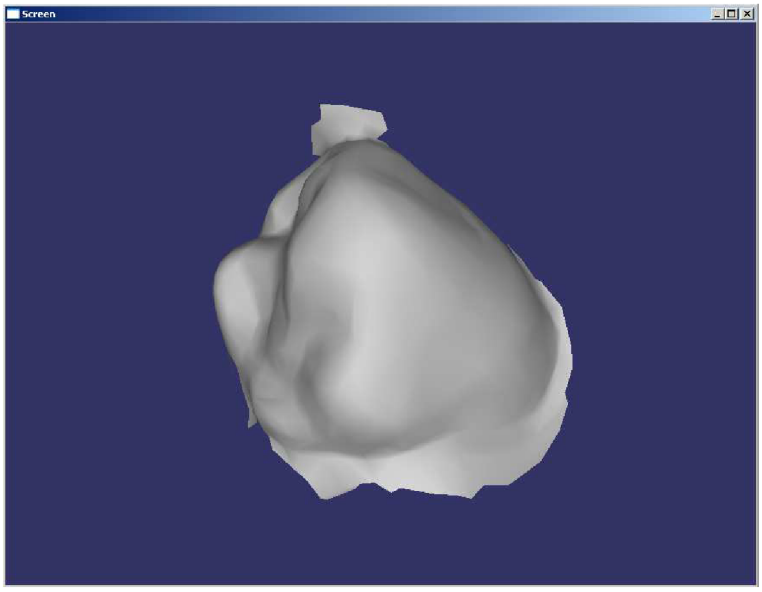


Obrázek 5.9: Výsledek detekce zubů vzniklí uložením 5.2 dohromady.

Obrázek 5.10: Výsledek detekce zubů vzniklí uložením 5.5 dohromady.



Obrázek 5.11: Jeden z výsledků detekce zubů vzniklí uložením 5.2 zvlášť.



Obrázek 5.12: Jeden z výsledků detekce zubů vzniklí uložením 5.5 zvlášť.

Kapitola 6

Závěr

6.1 Shrnutí stavu

Tato práce se zabývá detekcí zubů na 3D počítačovém polygonálním modelu. K problematice týkající se detekce samostatných objektů jsem na polygonálním modelu jsem našel na internetu několik publikací. Dvě z nich, které se mi zdáli nejvhodnější jsem použil jako základ aplikace. Jde o již zmíněné [16] a [2]. Kromě těchto článků jsem ještě našel práci [5], kde se tímto tématem zabývá podrobněji David Lon Page.

Při analýze problému bylo tedy nutné zvolit na základě poznatků vhodnou metodu pro detekci zubů. Po neúspěšné vlstní metodě založené na detekci ostrých hran jsem se rozhodl využít kombinace metod popsaných v [16] a [2], které jsou založeny na analýze křivosti povrchu. Nejsložitější bylo pochopit některé aspekty těchto metod, proto jsem aplikoval z každé metody jen to čemu jsem porozuměl a zkombinoval to dohromady. Výsledkem je hybrid obou metod doplněný o mojí invenci.

U výpočtu Gaussovy křivosti jsem se rozhodl, když jsem shlédl do jisté míry slabších výsledku - větší nepřesnosti detekce křivosti způsobené příliš velkým dělením povrchu, zahrnout do vyšetřování křivosti vrcholu větší okolí. Ukázalo se nakonec, že počítání s okolím 2. stupně je nejlepší. Okolí 3. stupně zase některá křivá místa příliš přehlíží.

Další použitá metoda *region growing*, která má za úkol dělit na základě hraničních vrcholů model do oblastí, díky algoritmu s příliš hluboko zanořenou rekurzí způsobuje při příliš nízkých hodnotách prahu křivosti pád aplikace. Jde ale o takové hodnoty prahu, při kterém stejně nelze žádné zuby detekovat. Hrozí ale, že při přepínání metod výpočtu Gaussovy křivosti z vyšší na nižší, dojde zdůvodu vyššího prahu křivosti u okolí 3. stupně k pádu aplikace u výpočtu 1. stupně. Je tedy silně doporučeno nastavit ve fázi, kdy není aktivovaná detekce pomoci žádného stupně Gausse, nastavit práh křivosti nejlépe na 0.

Zajímává optimalizace se však dozajista podařila u metody *region merging* a jejího možného opakováni, kdy se zároveň mění počet minimálních vrcholů nejmenší povolené oblasti. Díky tomu může aplikace produkovat různorodé výsledky.

Výsledkem práce je tedy jednoduchá aplikace, která umožňuje vcelku rychlou a relativně přesnou detekci zubů. Navíc je možno aplikaci použít i jako jednoduchý prohlížeč 3D modelů uložených v STL souboru. Aplikace se spouští buď pomocí dávky nebo z příkazového řádku. Ovládácím nástrojem je myš, která umožňuje manipulovat se scénou a několik klávesových zkratek. Soubor s modelem, ze kterého se načítá je binární STL soubor. Výsledek detekce lze uložit zpět do STL souboru a to dohromady nebo zvlášť.

6.2 Stanovení dalšího vývoje

Plně si uvědomuji, že má aplikace by pro lepší uplatnění snesla další vývoj.

Může se začít přesnější detekcí zubů, které by bylo potřeba detekovat přesněji a čistěji, tak aby k nim nebyly přidávány menší části dásní a naopak, aby menší kusy zubů nechyběly. Chtělo by to tedy již uvedené metody maximálně optimalizovat a vylepšit. Třeba i zkombinovat s jinou metodou detekce, kupříkladu porovnávat se šablonami jednolivých typů zubů.

Dále by mohla být aplikace schopná načítat model ve více formátech a také ho v nich ukládat. S tím souvisí i možnost vylepšení celkového uživatelského rozhraní, které by mohlo být více uživatelsky přátelské a k práci s aplikací poskytovat i jiné nástroje než klávesové zkratky. Např. nějaká barevná tlačítka se symboly naznačujícími jejich funkci.

Závěrem lze říci, že vylepšení a možností dalšího vývoje by se našlo mnoho a záleží pouze na lidské fantazii, jakým směrem by se aplikace měla dále ubírat.

Literatura

[1] Paul Bourke. Implicit surfaces. http://local.wasp.uwa.edu.au/ pbourke/modelling_rendering/implicitsurf/, Červen 1997.

[2] Lijun Chen and Nicolas D. Georganas. An efficient and robust algorithm for 3d mesh segmentation. *Multimedia Tools Appl.*, 29(2):109–125, 2006.

[3] Peter Doyle. Curvature of surfaces. http://www.geom.uiuc.edu/docs/doyle/mpls/handouts/node21.html, 2007.

[4] Přemysl Kršek. Přednášky kurzu izg. http://www.fit.vutbr.cz/study/course/IZG/private, Březen 2006.

[5] David Lon Page. *Part decomposition of 3d surfaces.* PhD thesis, 2003. Major Professor-Mongi A. Abidi.

[6] Michal Španěl. Medical Data Segmentation Toolkit - MDSTk. http://www.fit.vutbr.cz/ spanel/mdstk/home.html.en, Únor 2005.

[7] Jiří Žára. *Moderní počítačová grafika.* Computer Press, Brno, Česká republika, 2004. ISBN 80-251-0454-0.

[8] WWW strákny. Openscenegraph. http://www.openscenegraph.com/index.php, 2004.

[9] WWW strákny. Boundary representation. http://en.wikipedia.org/wiki/B-rep, Květen 2007.

[10] WWW strákny. Constructive solid geometry. http://en.wikipedia.org/wiki/Constructiv_Solid_Geometry, Květen 2007.

[11] WWW strákny. Octree. http://en.wikipedia.org/wiki/Octree, Březen 2007.

[12] WWW strákny. Polygon mesh. http://en.wikipedia.org/wiki/Polygon_mesh, Březen 2007.

[13] WWW strákny. Surface of revolution. http://en.wikipedia.org/wiki/Surface_of_revolution, Březen 2007.

[14] WWW strákny. Voxel. http://en.wikipedia.org/wiki/Voxel, Duben 2007.

[15] WWW strákny. Wire frame model. http://en.wikipedia.org/wiki/Wire_frame_model, Květen 2007.

[16] Y. Zhang, J. PAIK, and A. Koschan. *A simple and efficient algorithm for part decomposition of 3D triangulated models based on curvature analysis*, volume 3. IEEE International Conference on Image Processing, Rochester, NY, USA, 2002.

Dodatek A

Ovládání aplikace

Aplikace se spouští z příkazového okna příkazem:

main.exe -h

nebo:

main.exe –help

nebo:

main.exe model.stl

V prvních dvou případech se zobrazí nápověda prohlížeče OSG s paratmetry, které lze zadat při spuštění.

Ve třetím případě se načte a zobrazí daný model ze souboru.

V samotném prohlížeči má myš při výchozím nastavení při zmáčknutí jednoho z tlačítek tyto funkce:

- **levé tlačítko spolu s táhnutím myši** - v daném směru rotuje scéna kolem středu
- **prostřední tlačítko spolu s táhnutím myši** - v daném směru mění střed scény
- **pravé tlačítko spolu s táhnutím myši** - v daném směru oddaluje či přibližuje střed scény

Klávesové zkratky prohlížeče

Zde uvedu několik základních klávesových zkratek, významy dalších si lze vyhledat v nápovědě prohlížeče:

- **h** - zobrazí nápovědu prohlížeče
- **f** - přepínání prohlížeče mezi fullscreenem a oknem
- **esc** - ukončení aplikace
- **w** - přepínání mezi zobrazením vyplněných polygonů, hran a body

Specifické klávesové zkratky pro práci s modelem

Zde jsou uvedeny klávesové zkratky, které pracují s modelem:

- **j** - zobrazí nápovědu aplikace
- **e** - v příkazové řádce požádá uživatele o zadání jména souboru kam uloží výsledně detekované zuby dohromady
- **r** - v příkazové řádce požádá uživatele o zadání jména souboru kam uloží výsledně detekované zuby zvlášť
- **g** - přepíná mezi jednotlivými způsoby detekce zubů na základě tří analýz křivosti pomocí Gaussovy diskrétní křivosti, přičemž se počítá s okolím 1., 2. nebo 3. stupně
- **x** - v příkazové řádce požádá uživatele o zadání počtu opakování metody spojování oblastí
- **d** - přepíná mezi dvěma typy zahrnutí hraničních vrcholů do oblastí
- **k** - přepíná část modelu na, které se detekují zuby
- **p** - v příkazové řádce požádá uživatele o zadání nového prahu křivosti
- **n** - v příkazové řádce požádá uživatele o zadání poměru dělení 3D modelu na dvě části
- **m** - v příkazové řádce požádá uživatele o zadání minimálního počtu vrcholů v nejmenší oblasti