VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

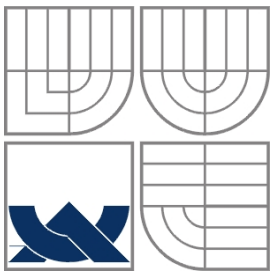
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MANIPULACE S 3D MODELY IMPLANTÁTŮ PRO PLÁNOVÁNÍ CHIRURGICKÝCH OPERACÍ V MEDICÍNĚ

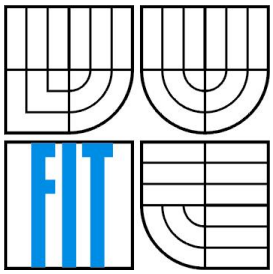
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MARTIN BIŠOF
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

MANIPULACE S 3D MODELY IMPLANTÁTŮ PRO PLÁNOVÁNÍ CHIRURGICKÝCH OPERACÍ V MEDICÍNĚ

MANIPULATION WITH IMPLANTS 3D MODELS FOR MEDICAL SURGERY PLANNING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN BIŠOF

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. KRŠEK PŘEMYSL, Ph.D.

BRNO 2007

Abstrakt

Tato práce popisuje vývoj i samotné řešení aplikace pro plánování a simulaci chirurgických operací v medicíně. Dokumentační část začíná teoretickým základem, jež se zaměřuje nejprve na reprezentaci objektů v počítačové grafice s bližším poznáním reprezentace hraniční. Následuje teoretický rozbor transformací a manipulátorů, které jsou v programu využity. Návrhová část stručně představí zvolená systematická řešení, a na abstraktní úrovni nás seznámí s problematikou zadaného projektu. Implementace pak plní úkoly dané zadáním od volby implantátů přes transformace modelů po manipulaci s kamerami. Praktická část je elementárně dokumentována s návazností na teoretické znalosti a je vytvářena s ohledem na fakta určená návrhem.

Klíčová slova

Hraniční reprezentace, transformace, manipulátory, OpenSceneGraph, graf scény, chirurgické operace, implantát.

Abstract

This thesis describes development and solution of application for planning and simulation of medical surgeries. The documentary part begins with the theoretical principles, which are focused on representation of objects in computer graphics with closer look at the boundary representation. The following part focuses on the theoretical analysis of transformations and manipulators, which will be implemented in the program. The design part of the thesis briefly introduces chosen systematic solutions and informs us on the abstract level with the issues of the whole project. Implementation includes solved tasks according to the assignment from the choice of the implants through transformation of the models to manipulation with cameras. The practical part is documented in the connection to the theoretical knowledge and is developed with the view of the given proposal.

Keywords

Boundary representation, transformation, manipulators, OpenSceneGraph, scene graph, surgical operations, implant.

Citace

Martin Bišof: Manipulace s 3D modely implantátů pro plánování chirurgických operací v medicíně, bakalářská práce, Brno, FIT VUT v Brně, 2007

MANIPULACE S 3D MODELY IMPLANTÁTŮ PRO PLÁNOVÁNÍ CHIRURGICKÝCH OPERACÍ V MEDICÍNĚ

Prohlášení

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

.....
Martin Bišof
8.5.2007

Poděkování

Rád bych poděkoval vedoucímu své práce Ing. Přemyslu Krškovi, Ph.D za odbornou pomoc a rady při řešení tohoto projektu. Dále pak celému týmu laboratoře počítačové grafiky pro medicínu za dosavadní zpracování informací a dokumentace ke knihovně *OpenSceneGraph*.

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Martin Bišof

Bytem: Polní 18, 58601, Jihlava

Narozen/a (datum a místo): 3.5. 1985, Jihlava

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta …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“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - ■ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako ……
   (dále jen VŠKP nebo dílo)

Název VŠKP:	Manipulace s 3D modely implantátu pro plánování chirurgických operací
Vedoucí/ školitel VŠKP:	Kršek Přemysl, Ing., Ph.D.
Ústav:	UPGM
Datum obhajoby VŠKP:	

VŠKP odevzdal autor nabyvateli v*:

- ■ tištěné formě – počet exemplářů …1……..
- ■ elektronické formě – počet exemplářů …2……..

* hodící se zaškrtněte

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 pořizovaní 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

Vysoké učení technické v Brně – Fakulta informačních technologií

Akademický rok 2006/2007

Zadání bakalářské práce

Téma:

Manipulace s 3D modely implantatu pro plánování chirurgických operací v medicíně

Vedoucí:

Kršek Přemysl, Ing., Ph.D., UPGM FIT VUT

Řešitel:

Bišof Martin

Pokyny:

1. Stručně prostudujte problematiku manipulace s 3D objekty ve 3D virtuálním prostoru prostřednictvím transformací
2. Prostudujte problematiku plánování chirurgických operací
3. Navrhněte interaktivní systém pro 3D plánování chirurugických opetací v medicíně
4. Implementujte a otestujte navržený systém
5. Zhodnoťte dosažené výsledky a stanovte další vývoj projektu

Kategorie:

Počítačová grafika

Implementační jazyk:

C++

Operační systém:

MS Windows, Linux, Unix

Návrhová metodologie:

OOP

Datum zadání:

1.listopadu 2006

Datum odevzdání:

15.května 2007

Literatura:

1. Žara J., Beneš B., Felkel P.: Moderní počítačová grafika. 1. vyd. Praha, Computer press 1998, 448 s., ISBN 80-7226-049-9
2. Drastich A.: Zobrazovací systémy v lékařství. 1. vyd. Brno, Rektorát VUT v Brně 1990, 512 s., ISBN 80-214-0220-2

Obsah

1 Úvod

Ve světe kolem nás je jen málo míst, kam počítačová grafika dosud nevkročila. Mocně se rozvíjí a doprovází nás ve nejrůznějších odvětvích, jako je filmová tvorba, virtuální realita či třeba herní průmysl. Tady všude můžeme pozorovat sílu, jež roztáčí vrtule milionovým projektům. V důsledku toho se ve stínu stále nových technologií a nástrojů přibližuje počítačová grafika více realitě a zasahuje do různých sfér lidského života. Je zřejmé, že své místo má i v oborech, jako je školství nebo třeba lékařství, kde je v různých formách využívána na celém světě. Právě lékařství a využití počítačové grafiky při plánování chirurgických operací se tato práce dotýká.

Předoperační plánování je oblast, ve které může právě počítačová grafika zastat nemalou část prostředků, mimo jiné zprůhlednit a zrychlit samotný zákrok. Každý člověk je jedinečný, proto je při každém plánování potřeba individuálního přístupu k problému. Počítačovou tomografií nebo magnetickou rezonancí (CT/MR) je možné získat data pro trojrozměrný model lidské tkáně konkrétního případu. Nabízí se celé spektrum dalšího využití.

Cílem této práce je vytvořit prostředí, ve kterém bude možné nasimulovat použití zvoleného implantátu na daný případ. Tedy vytvořit sadu implantátů různých velikostí, jež bude možno v aplikaci měnit dle potřeby a vzhledem k nehybné tkáni s nimi manipulovat. Tímto způsobem si potom chirurg může vyzkoušet celou řadu řešení nanečisto a zlepšit tím průběh a dobu trvání samotné operace. Implantáty v prostředí bude možno přidávat podle potřeby a také po výběru odstraňovat. Pro větší přehlednost je vhodné využít několika pohledů z různých úhlů. Ty budou také ovladatelné, bude možno se pomocí nich zaměřit na detail a sledovat tím incident tkáň-implantát zároveň z více stran pohledu.

Návrh i implementace aplikace je v této publikaci dokumentována. Nejprve je stručně rozebrána teorie oblastí, do kterých práce zasahuje, tedy reprezentace trojrozměrných objektů v počítačové grafice, transformace a manipulátory. Následuje kompletní návrh programu, rozebírající například strategii načítání a volbu implantátů či třeba rozvržení oken v uživatelském prostředí. V části implementace je pak nastíněno konkrétní řešení a realizace dílčích problémů včetně samotné tvorby implantátů.

Ve všech částech dokumentu jsem se snažil o názornost, proto je publikace provázena obrázky, jež jsem k tématům ilustroval. V závěru práce jsem zhodnotil dosažené výsledky s ukázkami a uvedl několik možností, jakým směrem by se měla aplikace v budoucnu dále vyvíjet.

2 Teoretický rozbor

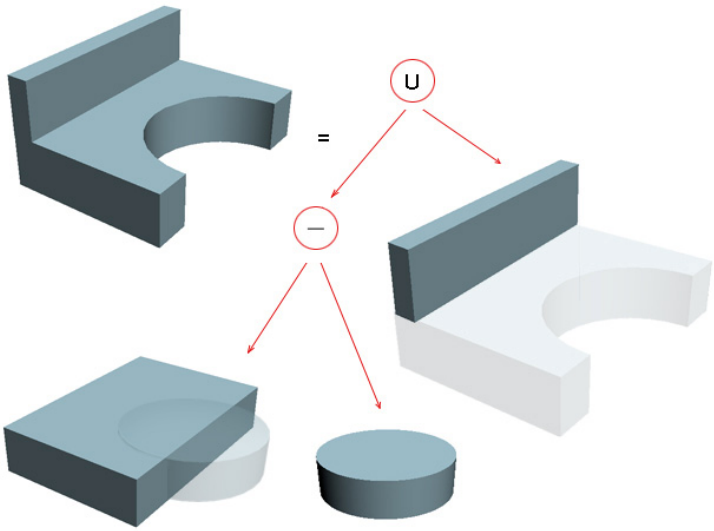
V první části bakalářské práce je nezbytné teoreticky uvést jednotlivé elementy problematiky, kterých se celý projekt dotýká. V první řadě jde o samotnou reprezentaci objektů a možnosti jejich zobrazování. Ukážeme si zde několik způsobů, přičemž se zaměříme především na reprezentaci hraniční. Teoretická část dále rozebírá kapitolu transformací v počítačové grafice a uvádí typické příklady, jako je posunutí, rotace objektu, změna měřítka a ve stručnosti se zmíní i o projekční transformaci. Téma volně přechází k látce s tím spojené, a sice k manipulátorům. Zde si povíme něco o typických představitelích pro jednotlivé manipulace a jejich nadstandardních možnostech.

2.1 3D reprezentace

První část teoretického rozboru stručně popisuje problematiku, jak mohou být objekty v imaginárním světě 3D grafiky reprezentovány. Jedná se o několik způsobů, jež se již ujaly a do jisté míry mají své zastoupení v různých odvětvích dle svého zaměření, potažmo jejich možností použití.

V dnešním světě, plném zdokonalujících se multimédií a grafiky vůbec, lze říci, že monopolní zastoupení má pravděpodobně hraniční reprezentace (*boundary representation, B-rep*). Zde, jak již z názvu vyplývá, je objekt definován svými hranami. Jedná se o reprezentaci pomocí vrcholů, křivek či například množinou trojúhelníků. Metoda je to úsporná, nevýhodou je však pomíjení způsobu vzniku objektu. Více je uvedeno v další části kapitoly.

Opakem hraniční reprezentace je, z hlediska uchování způsobu vzniku objektu, konstruktivní geometrie. Konstruktivní geometrie těles (*CSG*) se zakládá na reprezentaci objektu stromovou strukturou, kde jsou zaznamenány jednotlivé konstrukční kroky.



Obrázek 2.1 Ukázka konstruktivní geometrie CSG

Jedná se o vytváření modelu z jednoduchých geometrických primitiv za pomoci množinových operací a prostorových transformací. Za primitiva můžeme označit např. kouli, kvádr, válec, kužel, jehlan i toroid. Množinové operace potom zastupuje sjednocení, průnik nebo rozdíl. Význam má tato metoda především pro samotné konstruktéry, často se potom kvůli zobrazení převádí například na hraniční reprezentaci.

Za další oblast můžeme považovat objemovou reprezentaci. Využívá se především v případech, kdy nemáme k dispozici geometrický popis tělesa, ale pouze sadu vzorků v určitém místě povrchu nebo objemu. Jinde může být sada strukturovaná do podoby pravidelných či nepravidelných mřížek. Důležitá je pak metodika pro převody do hraniční reprezentace.

V neposlední řadě se můžeme setkat s pojmem procedurální modelování. To je založeno na generování objektů na základě definovaného algoritmu. Jednak jde o generování např. ploch, křivek nebo šablonování, ale především se jedná o metody automatického generování objektů, které se vizuálně či chováním podobají objektům, s nimiž se setkáváme v přírodě apod.

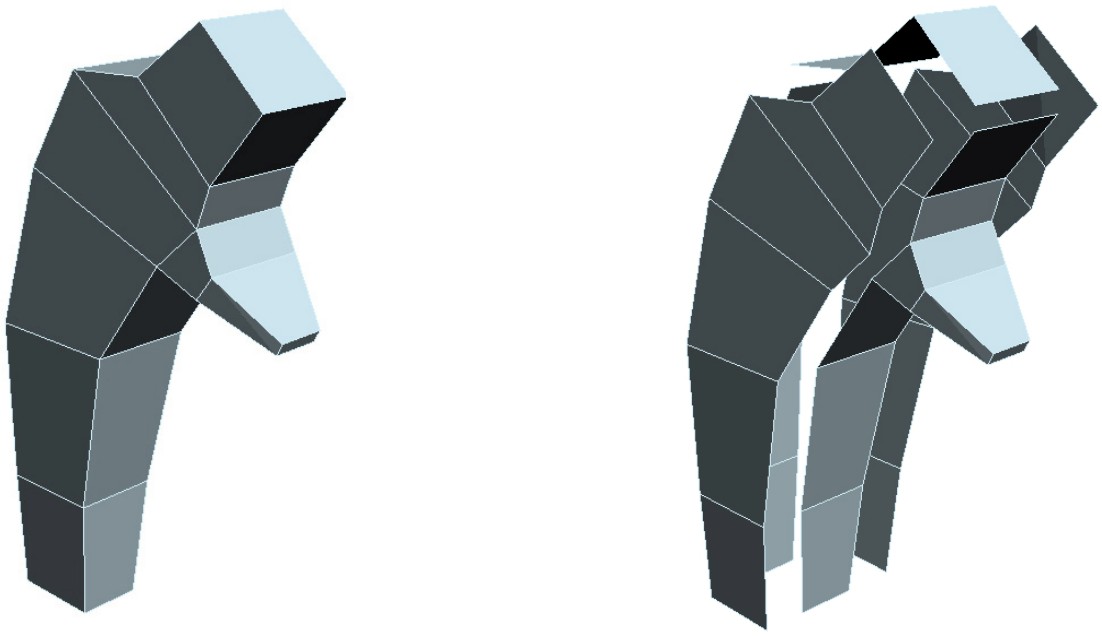
Ve stručnosti bych se zmínil o pojmu implicitní plochy. Podstata tkví v modelování objektů pomocí kostry, která je jakoby obalena hmotou podle intenzity pole v každém bodě. Pro zobrazení se pak model převádí například opět na hraniční reprezentaci.

Dále se budeme výhradně zabývat reprezentací spojenou s praktickou částí této práce, tj. hraniční reprezentací.

2.1.1 B-rep

Jak již bylo výše zmíněno, relevantní pro účely zadané bakalářské práce je reprezentace hraniční. Způsob je to vhodný nejen díky své úspoře, ale také dostupnosti i jednoduchosti. Jeho další nesporné přednosti můžeme najít i z hlediska dalšího zpracování, zobrazování hraniční reprezentace je snadno proveditelné v grafických procesorech.

Ukážeme si metodu přímo na rozpracovaném polygonálním modelu implantátu, který jsem pro účely tohoto projektu vytvářel.

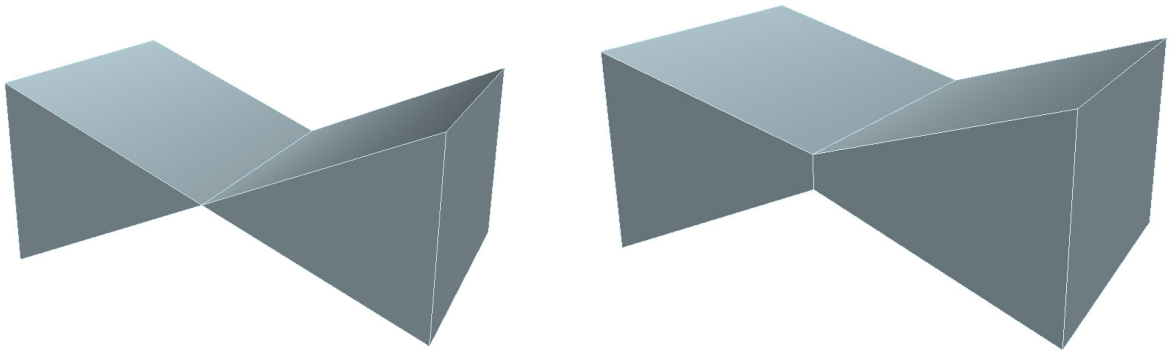


Obrázek 2.2 Hraniční reprezentace

Jak můžeme vidět, model na obrázku vlevo působí celistvě, pokud ovšem dojde k přeskupení některých ploch pláště, zjistíme, že informace o vnitřních bodech tělesa se neuchovávají (popřípadě je lze odvodit z popisu hranice). Hranice tělesa jsou tedy jeho přirozenou reprezentací, model se tváří velmi úsporně. Je tento způsob ale vždy ten správný?

2.1.2 Manifold objekty

Můžeme říci, že metoda definice tělesa, jak jsme si ji popsali výše, nás zcela jistě neuchrání před možností vymodelovat a popsat objekt, který by nebylo v reálném světě možné vytvořit. Skutečnost, že počítačový popis může určovat i nevyrobitelné těleso, pramení z použití matematické a geometrické abstrakce. Pro názornost uvedu příklad nonmanifoldního (nevyrobitelného) a manifoldního (vyrobitelného) objektu.



Obrázek 2.3 Nonmanifold a manifold objekt

Sestrojit těleso, jehož části se budou např. dotýkat pouze v jednom vrcholu je asi stejná utopie, jako vyrobit absolutně rovnou plochu nebo malovat absolutně černou barvou. Těchto možností je v počítačové grafice nespočet. Vlevo je jasný příklad nonmanifoldního objektu. Dvě části objektu se stýkají v místech nekonečně tenké hrany, která inciduje se čtyřmi plochami. Naopak vedle vidíme objekt, jehož obdobu bude možné reálně zkonstruovat. Za manifold můžeme z praktického hlediska považovat takové těleso, jehož každá z hran inciduje právě se dvěma ploškami a jehož hrany neprotínají žádné jiné plochy. Obdobně pak platí, že jeden vrchol nespojuje dvě a více částí tělesa.

2.1.3 Eulerova rovnost

V nemalém zastoupení používaných těles v trojrozměrné grafice jsou mnohostěny (těleso ohraničené množinou mnohoúhelníků). Pro mnohostěn musí platit, že každou z hran sdílí vždy sudý počet stěn. Mnohostěn, který neobsahuje díry a deformací lze převést na kouli, se nazývá jednoduchý mnohostěn. Eulerova rovnost zajišťuje, že množina vrcholů, stěn a hran tvoří jednoduchý mnohostěn, jestliže platí, že každá hrana propojuje právě dva vrcholy a stěny se navzájem neprotínají.

$$F + V - E = 2.$$

Vzorec udává vztah mezi počtem stěn (F), vrcholů (V) a hran(E) pro jednoduchý mnohostěn.

Pro mnohostěny, které jsou prolnuty otvory, platí obecnější Eulerova rovnost, kde jsou navíc další operátory.

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

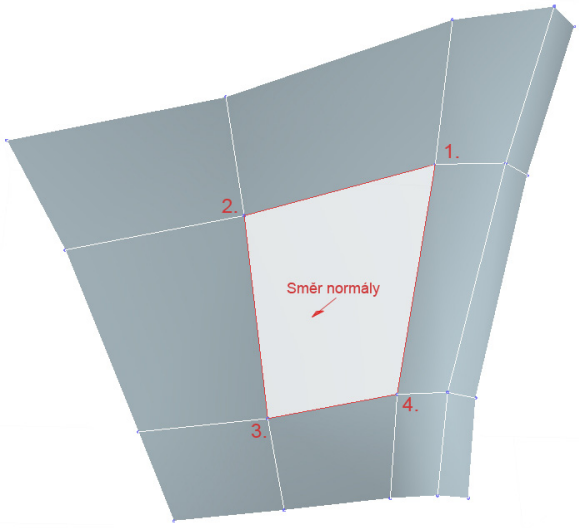
Člen R označuje počet vnitřních smyček hran, C počet oblastí nebo samostatných komponent tělesa a H počet otvorů v tělese.

2.1.4 Reprezentace pomocí hran a vrcholů

Existuje několik možností jak reprezentovat objekt pomocí prvků, jako jsou výše zmíněné vrcholy, hrany či plochy. Nejjednodušší metoda popisu tělesa vychází pouze z prvních dvou uvedených, tedy z vrcholů a hran. Je předem jisté, že takový objekt nebude pokryt žádným pláštěm, který by tvořil samotné plochy. Vytvoříme tím efekt tzv. drátový model, který však nebude jednoznačně interpretovatelný. Ke každé hraně modelu náleží vždy dva vrcholy, jejichž souřadnice jsou vždy definovány. Toto je velmi jednoduchá struktura zabírající minimum paměti, neobsahuje už ale žádné další informace o topologii jednotlivých elementů modelu.

2.1.5 Reprezentace pomocí triviálních ploch

Drátový model má jistě celou řadu uplatnění, avšak pro realističtější zobrazení je potřeba podat o objektu více informací. Především je potřeba pokrýt tuto „kostru“ ploškami, které budou simulovat povrch. Plochy můžou být uspořádány do sítě trojúhelníků nebo polygonů. Důležité je však uchovávat vždy informace o počtu a pořadí vrcholů.

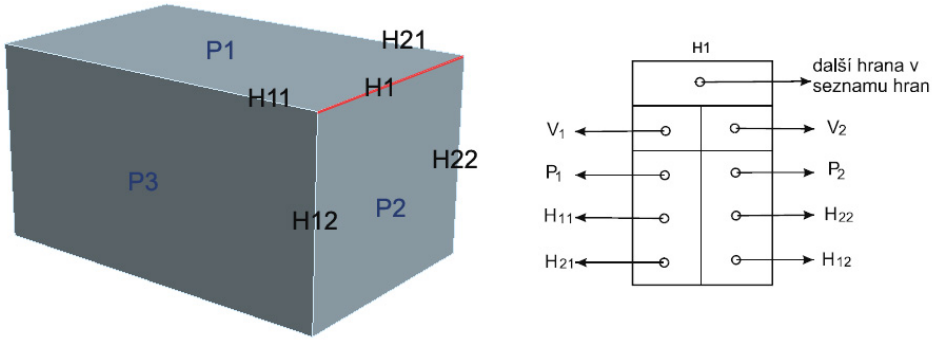


Obrázek 2.4 Pořadí vrcholů při vytváření plochy

Jak ukazuje obrázek, je relevantní, s jakým pořadím vrcholů bude plocha vložena. Závisí na něm směr normály, který určuje, bude-li plocha viditelná zvenku či zevnitř objektu. Tímto způsobem tedy zajistíme, že objekt bude viditelný od pozorovatele. Není to však pořád komplexní přístup, který by říkal například něco o vzájemném sousedství jednotlivých plošek.

2.1.6 Okřídlená hrana

Za opravdu strukturovanou reprezentací, složenou z vrcholů, hran a ploch, lze označit metodu navrženou B.G. Baumgartem. Metoda vytváří takovou strukturu, která je vhodná pro popis



Obrázek 2.5 Okřídlená hrana

polygonálního modelu a je určena v základu pouze pro manifold objekty. Název si vysloužila díky podobnosti hrany, vždy sousedící se dvěma ploškami. Každá hrana nese informace o sousedních elementech, konkrétně ukazuje vždy na koncové vrcholy (V1, V2), sousední plochy (P1, P2) a nese

informace o dalších čtyřech hranách (H11,H21,H12, H22). V levé polovině se nacházejí hrany sousedící s levou plochou (H11, H21), v pravé související s plochou pravou (H12, H22). Ukazatel *p* má pak hodnotu adresy další hrany ve zřetězeném seznamu. Není pak složité z takovéto struktury určit důležité informace o topologii, jako např. všechny sousedící plochy s danou plochou, plošky incidující s danou hranou, vrcholy a hrany dané stěny či plochy, stýkající se v daném vrcholu.

Pomocí podobné struktury lze reprezentovat i nonmanifold objekty. Označuje se jako půlhrana a smysl této metody tkví v myšlence rozdělení hrany tělesa na skupiny půlhran spojujících vždy tutéž dvojici vrcholů. Taková půlhrana pak představuje dvojici stěna-hrana.

2.1.7 Reprezentace pomocí bodů

Jak již z názvu vyplývá, půjde o reprezentaci pouze pomocí povrchových bodů. Taková množina bodů může být získána například digitálním snímáním objektu či příslušným algoritmem. Tyto body ve své struktuře nenesou pouze informaci o poloze, ale i směru normálového vektoru, barvě a dalších parametrů týkajících se materiálu tělesa. Taková reprezentace je určena třeba k archivačním účelům reálných architektonických památek, kde je kladen důraz na zachování přesných dat. Paměťové nároky pak u velmi detailních objektů budou mimořádně vysoké.

Při zobrazování této reprezentace však můžeme narazit na problém, že mohou vznikat nechtěné nespojitosti. U modelu složeného z trojúhelníků umíme přesně určit hranice každého z nich, u množiny izolovaných bodů toto nelze. Podle lokální hustoty bodů jsme však schopni určit, jak by měly být jednotlivé body velké, aby se navzájem překrývaly. Metoda není vhodná pro ploché a jednoduché objekty a je nepraktická k aplikaci na ostré hrany. Byly proto vyvinuty hybridní techniky, které kombinují vlastnosti bodů s vlastnostmi trojúhelníků.

2.2 Transformace

V kapitole 2.1 jsme si rozvedli teorie, jakým způsobem lze objekty v počítačové grafice zobrazovat. To ovšem samo o sobě nestačí. Nyní je důležité zamyslet se, jak budou modely ve scéně umístěny, jakým způsobem budou natočeny, či v jakém měřítku se na každý z nich budeme dívat. Volit místo pozorovatele a vzdálenost je také určující. Vždy totiž zabíráme právě tu oblast, která je pro nás důležitá. Toto vše se týká jedné velké kapitoly, které říkáme transformace. Můžeme je pomyslně rozdělit do tří skupin.

- Lineární transformace
- Nelineární transformace
- Projekční transformace

První skupinu tvoří lineární transformace. Počítáme do ní například posunutí, rotace a změnu měřítka. Složité operace, týkající se deformací a wrapingu, označujeme jako nelineární. Zvláštním případem je potom transformace projekční, která se týká převodu trojrozměrných scén na úroveň dvojrozměrného obrazu. Můžeme říci, že geometrické transformace jsou určitě jedny z nejčastěji využívaných operací v počítačové grafice.

Objekty jsou popsány svými souřadnicemi, které jsou vztaženy ke zvolenému souřadnicovému systému, transformace pak můžeme použít přímo na jednotlivé souřadnice. Objekt potom bude vzhledem k souřadnicovému systému měnit svou polohu. Jinou možností je aplikovat transformace na celý souřadnicový systém, což může být výhodné pro další zpracování objektu, jako je například výpočet objemu apod.

Lineární transformace aplikujeme na všechny body objektu. Takový bod *P* má v kartézské soustavě pro tři rozměry souřadnice [x, y, z], pro dva rozměry souřadnice [x, y] . Potom transformací tohoto bodu *P* získáme nově bod *P'* o souřadnicích [x', y', z'], obdobně pro dva rozměry [x', y'].

2.2.1 Homogenní souřadnice

Pro práci s transformacemi je výhodný převod na nějakou jednoduchou strukturu, ve které pak nebude problém skládat jednotlivé transformace a dále s nimi pracovat. Homogenní souřadnice jsou využívány tak, aby umožnily vyjádření lineárních transformací ve tvaru jedné matice, což v nehomogenních kartézských souřadnicích není možné.

Homogenní souřadnice bodu P s kartézskými souřadnicemi [x, y, z] zapíšeme ve tvaru [x, y, z, w], kde hodnota *w* představuje váhu bodu a velmi často se volí s jedničkovou hodnotou. Ostatní souřadnice bodu se potom vyjádří následovně:

$$x = \frac{x}{w}, y = \frac{y}{w}, z = \frac{z}{w}, w \neq 0.$$

Ze zápisu vyplývá, že budeme-li mít bod s parametry souřadnic [2,4,6,6] a druhý bod se souřadnicemi [4,8,12,12], půjde stále o ten stejný bod.

Jestliže potom bod P s homogenními souřadnicemi [x, y, z, w] transformujeme, dostaneme nový bod P' s hodnotami souřadnic [x', y', z', w']. Lineární transformaci zmiňovaného bodu P budeme reprezentovat pomocí matice M o velikosti 4x4 pro tři rozměry. Pro dvourozměrné transformace se potom využívá matice 3x3.

2.2.2 Dvourozměrné transformace

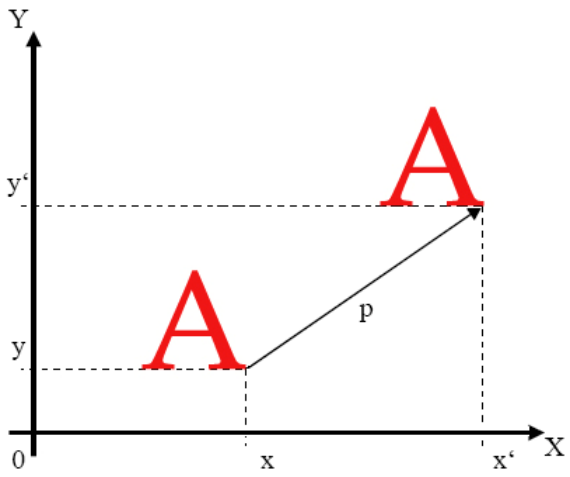
Po uvážení výše zmíněného textu a algebraických pravidel můžeme napsat odpovídající rovnost matic ve tvaru:

$$P' = M \cdot P = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}.$$

Je předem zřejmé, že pro případy trojrozměrných transformací budeme používat obdobný zápis, který bude pouze jakýmsi zobecněním. Nejprve si pro jednoduchost představíme transformace bodu pro dva rozměry a hned si uvedeme typické případy lineárních transformací, jež se v počítačové grafice hojně používají.

2.2.3 Translace ve 2D

Posun dvourozměrného bodu určíme vektorem posunutí $\vec{p} = (x_t, y_t) = (x' - x, y' - y)$. Matice transformace posunu T má tvar $T(x_t, y_t) = \begin{bmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{bmatrix}$ a inverzní matice k matici T bude ve tvaru $T^{-1}(x_t, y_t) = T(-x_t, -y_t) = \begin{bmatrix} 1 & 0 & -x_t \\ 0 & 1 & -y_t \\ 0 & 0 & 1 \end{bmatrix}$. Případ nazvaný posunutí nebo také translace ve 2D situuje následující obrázek.

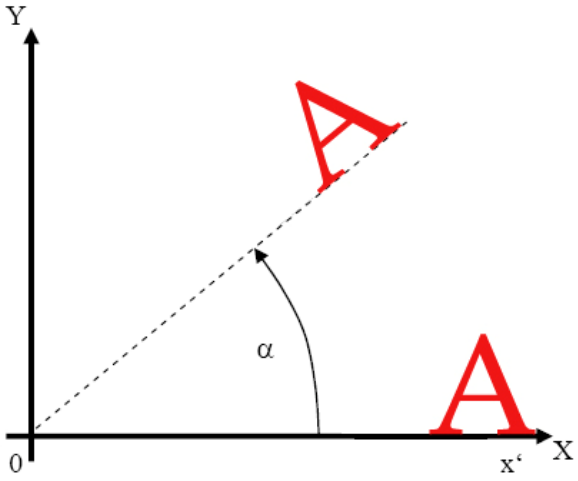


Obrázek 2.6 Translace

Po transformaci jsou nově získané souřadnice bodu P' $x' = x + x_t$, $y' = y + y_t$.

2.2.4 Rotace ve 2D

Transformace rotace je označení pro otočení bodu P kolem počátku soustavy souřadnic O=[0,0] o úhel α.



Obrázek 2.7 Rotace

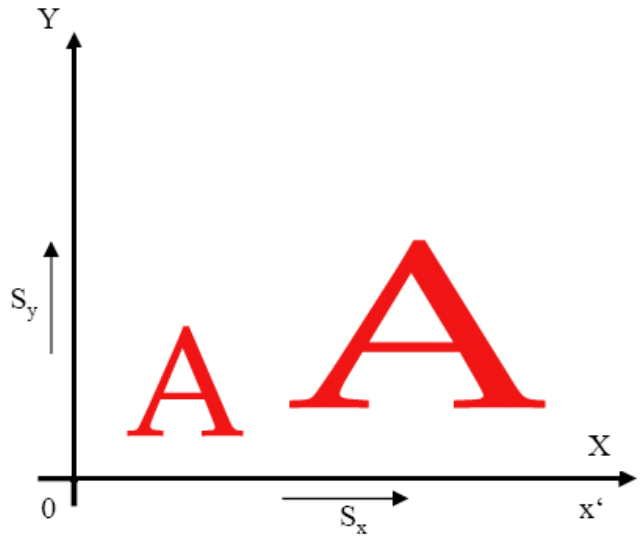
Matice transformace rotace R a její inverzní matice R^{-1} jsou ve tvaru

$$R(\alpha)=\begin{bmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix},\ R^{-1}(\alpha)=R(-\alpha)=\begin{bmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}.$$

Po transformaci jsou nově získané souřadnice bodu P' $x'=x\cdot\cos\alpha-y\cdot\sin\alpha$, $y'=x\cdot\sin\alpha+y\cdot\cos\alpha$.

2.2.5 Změna měřítka ve 2D

První dva případy transformací mají relativně jednoduchý charakter v tom směru, že ovlivňují pouze jednu preferovanou veličinu. Naproti tomu, zamyslíme-li se například nad potřebou zvětšovat či zmenšovat objekt, měli bychom vzít v úvahu, že budeme současně ovlivňovat polohu i velikost objektu ve směru souřadnicových os. Koeficient určující míru zvětšení nebo zmenšení je pro horizontální osu S_x a pro vertikální S_y. V intervalu od nuly do jedné dochází ke zmenšování a posunu objektu směrem k počátku souřadnicových os. Naopak hodnoty větší jak jedna objekt zvětší a od počátku se oddálí. Záporné hodnoty budou měnit velikost v záporném směru.



Obrázek 2.8 Změna měřítka

Matice transformace změna měřítka S a její inverzní matice S^{-1} jsou ve tvaru

$$S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; S^{-1}(s_x, s_y) = S(\frac{1}{s_x}, \frac{1}{s_y}) = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Po transformaci jsou nově získané souřadnice bodu $P' \;\; x' = x \cdot s_x, \;\; y' = y \cdot s_y$.

2.2.6 Zkosení ve 2D

Příklad zkosení v ose x je na následujícím obrázku. V maticích definujeme jeho míru koeficientem sh_x pro horizontální zkosení ve směru osy x a sh_y pro zkosení ve vertikální ose y. Maticové vyjádření je následující:

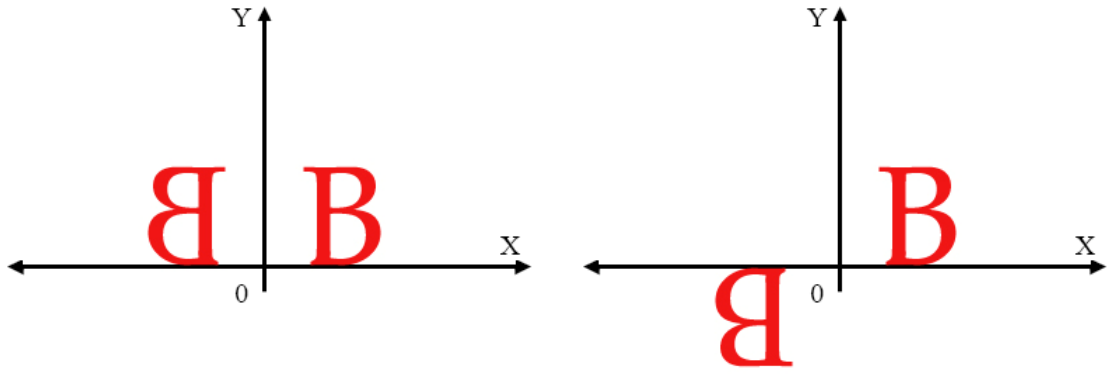
$$Sh(sh_x, sh_y) = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; Sh^{-1}(sh_x, sh_y) = \begin{bmatrix} 1 & -sh_x & 0 \\ -sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Po transformaci jsou nově získané souřadnice bodu $P' \;\; x' = x + sh_x \cdot y, \;\; y' = sh_y \cdot x + y$.

2.2.7 Zrcadlení ve 2D

Zrcadlení je speciálním případem změny měřítka, kdy absolutní hodnota koeficientu pro změnu je jedna, ostatní členy jsou zachovány. Na následujících obrázcích můžeme sledovat dva typy

souměrností. Příklad vlevo prezentuje, jak by mohla vypadat osová souměrnost podle *y*, stejně tak si jistě dovedeme představit odlišnost transformovaného písmene pro souměrnost podle osy x.



Obrázek 2.9 Zrcadlení ve 2D (osová a středová souměrnost)

Klíčové členy S_x a S_y jsou pro osovou souměrnost podle osy *y* rovny 1 resp. -1, u souměrnosti podle osy *x* jsou hodnoty $S_x = -1$ a $S_y = 1$. Obrázek vpravo znázorňuje souměrnost středovou. U zrcadlení podle osy jsme převraceli body pouze podle jedné z os, u středové souměrnosti převracíme podle obou. Hodnoty jsou tedy $S_x = -1$ a $S_y = -1$.

2.2.8 Skládání transformací

Jak jsme již výše naznačili, práce s transformacemi bude skutečně efektivní v případě, že budeme mít možnost jednotlivé transformace kombinovat. Toto samozřejmě lze a je to i jeden z důvodů, proč jsme si na začátku rozebírali převod na homogenní souřadnice. Při aplikaci jednotlivých transformací je zřejmě, že bude záležet na pořadí jednotlivých úkonů. Výsledek bude rozdílný, jestliže objekt nejprve posuneme a teprve potom s ním budeme rotovat okolo počátku souřadnicového systému, nebo zda body nejprve otočíme a následně posuneme. Používáme-li zápis $P' = M \cdot P$, budeme pozdější transformace přidávat na začátek zleva. Pokud máme tedy nejprve bod P posunout maticí *T* a následně rotovat maticí *R*, bude výsledný bod dán vztahem $P' = R \cdot T \cdot P$.

2.2.9 Trojrozměrné transformace

Pro transformace ve třech rozměrech můžeme analogicky využít fakta, jež platí a které jsme si uvedli pro transformace ve 2D. Rozšíříme matice o hodnoty osy *z*, tedy používáme typ matice 4x4 a bude obecně platit následující:

$$P' = M \cdot P = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{14} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}.$$

2.2.10 Posun ve 3D

Ve třech rozměrech je posunutí dáno vektorem $\vec{p} = (x_t, y_t, z_t)$. Matice transformace translace T a její inverzní matice T^{-1} jsou ve tvaru

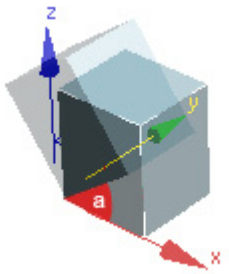
$$T(x_t, y_t, z_{tt}) = \begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T^{-1} = T(-x_t, -y_t, -z_{tt}) = \begin{bmatrix} 1 & 0 & 0 & -x_t \\ 0 & 1 & 0 & -y_t \\ 0 & 0 & 1 & -z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2.2.11 Rotace ve 3D

Transformaci rotace ve 3D o úhel α si pojmeme nejdříve jako rotaci kolem jedné z os. Příslušné vyjádření pro R_x, R_x^{-1}, R_y a R_z je následující:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_x^{-1}(\alpha) = R_x(-\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$



Obrázek 2.10 Ukázka rotace ve 3D o úhel $a \approx \alpha$

Rotaci kolem obecné osy realizujeme složením několika rotací kolem osy x, y a z. Výsledná osa rotace je dána vektorem a bodem umístění. Nalezení příslušných transformací otočení není triviální. Při řešení volíme bod $P = \lfloor P_x, P_y, P_z, 1 \rfloor$ na ose rotace, vypočteme jednotkový vektor $\vec{v}$ ve směru osy otáčení, tento bod posuneme do středu souřadnicového systému, provedeme příslušné rotace a posuneme bod zpět do původních souřadnic. Výsledná matice M bude tedy dána vynásobením tří transformačních matic

$$A = T(P_x, P_y, P_z) \cdot R(\vec{v}, \alpha) \cdot T(-P_x, -P_y, -P_z).$$

2.2.12 Zkosení ve 3D

Zkosení se aplikuje na body v trojrozměrné grafice podobně jako u 2D s tím rozdílem, že samotné zkosení není podle jedné osy, ale podle roviny, kterou dvojice os tvoří. Míra zkosení je dána opět členem Sh, pro jednotlivé roviny platí následující vyjádření:

$$Sh_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Sh_x & Sh_y & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; Sh_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_x & 1 & Sh_z & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; Shy_{yz} = \begin{bmatrix} 1 & Sh_x & Sh_x & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

U změny měřítka ve 3D platí stejná pravidla jako u dvojrozměrné transformace zase s rozšířením na matici 4x4. Souměrnost je řešena podle rovin, tedy například souměrnost podle roviny xy se vytvoří nastavením koeficientů hodnotami S_x=1, S_y=1, S_z= -1.

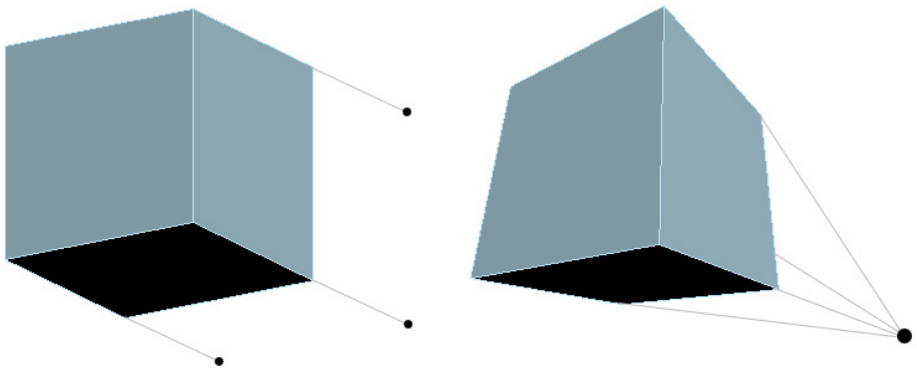
2.2.13 Projekční transformace

Reprezentace trojrozměrných objektů, jak jsme si je doteď definovali, obsahují informaci o prostoru, kdežto na obrazovku počítače či na plátno jej musíme zobrazit dvojrozměrně a tuto informaci tím ztratit. Tento proces převodu nazveme projekcí nebo také promítáním. Z prostorového bodu ve scéně vychází projekční paprsek, který dopadá na plochu (tzv. průmětnu). Směr paprsků je dán způsobem, který chceme na 3D obraz aplikovat. Promítání můžeme v zásadě rozdělit na dvě třídy:

- paralelní
- perspektivní.

Paralelní nebo-li rovnoběžná metoda se vyznačuje tím, že její projekční paprsky jsou vyslány všechny rovnoběžně a mají stejný směr. Její typický příklad můžeme vidět na krychli na obrázku vlevo.

U perspektivního nebo také středového promítání vychází paprsky pouze z jednoho bodu, vytváříme tím realističtější pohled, který je bližší pohledu člověka. Při středovém promítání jsou vzdálenější objekty ve výsledném vyobrazení menší, naproti tomu paralelní projekce proporce striktně zachová. Metoda rovnoběžné projekce má však nesporné výhody v použitelnosti. Její uplatnění zasahuje např. do stavebnictví, konstruktérství typu CAD apod.



Obrázek 2.11 Příklad paralelní projekce vlevo a perspektivní vpravo.

2.3 Manipulátory

V předcházející kapitole jsme si řekli něco o základních transformacích v počítačové grafice. Jejich definice pomocí matic je jistě velkým přínosem. Uživatel softwaru však jistě nechce zadávat hodnoty koeficientů těchto matic ručně. Má-li být práce s aplikací efektivní a dostupná i pro méně zkušené jedince, je potřeba uživateli samotné transformace zpříjemnit pomocí nějakého rozhraní. Samotná problematika veškerých výše zmíněných transformací zůstane uživateli skryta a k modifikacím dotyčných matic bude docházet právě skrze toto rozhraní.

Řešíme-li problematiku návrhu rozhraní pro manipulaci, pokládáme si otázku týkající se cílové skupiny, pro kterou bude aplikace určena a celkový přínos tohoto „rozšíření“. Klademe zde důraz na dva rozhodující aspekty:

- přesnost
- uživatelská přívětivost.

V jistých odvětvích využívání grafického softwaru bude nezbytně důležité, aby transformace byly naprosto přesné. Když chce konstruktér umístit objekt na osu x do vzdálenosti 5,11 od středu souřadnicového systému, musí dotyčná transformační matice tuto translaci skutečně přesně provést. Jedním z řešení může být zadávání konkrétních hodnot do připravených polí, což požadavek na přesnost určitě splňovat bude. Další možností je realizovat transformace na podobné bázi, ovšem pohybovat se bude objektem vždy po stisku tlačítka apod. o předem stanovený krok. Zachováme tím přesnost a celkový vykonaný pohyb je potom zpětně přepočítatelný podle počtu kroků. Metodě se někdy také říká offset transformací, jelikož dochází pouze k přičtení hodnoty k aktuálnímu stavu.

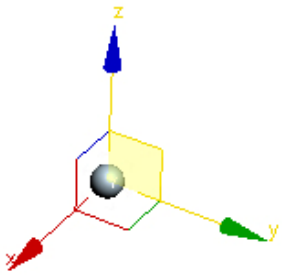
2.3.1 Grafické manipulátory

Pro aplikace jiného typu, kde na přesnosti zase tak docela nezáleží, je spíše rozhodující uživatelská přívětivost. Ta lze docílit grafickou úpravou, jež bude vnášet do samotných transformací průhlednost a jednoduchost. Představme si objekt, který chceme transformovat. Nejpraktičtější by bylo u něj, či nejlépe přímo ve středu jeho souřadnicového systému, zobrazit jednoduché geometrické objekty,

které by intuitivně uživatele svým tvarem informovaly o aktuálních možnostech transformace. Tyto objekty by nebyly přímo součástí scény, i když by byly ve scéně zobrazovány, nýbrž obrazně nad nimi. Jejich vzhled by se měnil vždy s typem transformace a toto by bylo možné aplikovat i v rámci skupiny objektů. Je samozřejmě na nás, jakou strategii si při vlastní implementaci zvolíme, tuto teorii si ale můžeme ukázat na příkladech z komerčních aplikací.

2.3.2 Manipulátor translace

U manipulátorů, jež mají provádět pohyb, jsou nejdůležitější osy, po kterých bude k transformacím docházet. Tyto osy by měly být reprezentovány úsečkami nejlépe s rozdílnými barvami. Šipky na jejich koncích naznačují směr a možnost posunu, je také dobré jednotlivé osy nadepsat. Samotná akce se pak provádí způsobem „chyť a táhni“.

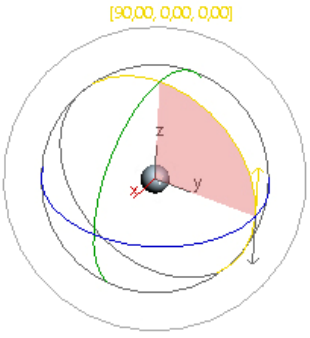


Obrázek 2.12 Manipulátor posunu

Takový manipulátor se přímo nabízí k možnosti transformovat objekt v rovinách, tedy měnit souřadnice dvou os zároveň. V aplikaci *3ds max* je to vyřešeno naznačením plochy, jež je ohraničena právě dvěma dotyčnými osami. Je jasné, že posunutí bude pouze přibližné. Je však možné vylepšit manipulátor o ukazatel hodnoty, jež bude samotný posun uživateli upřesňovat.

2.3.3 Manipulátor rotace

Pro manipulátory otáčení bude tvar objektu takový, aby opět intuitivně naznačoval svou činnost. Rotaci po osách x, y a z nebudou reprezentovat úsečky, ale kružnice.

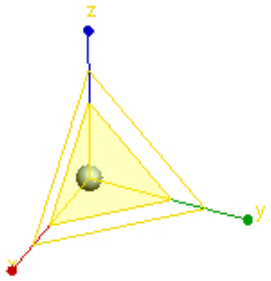


Obrázek 2.13 Manipulátor rotace

Na obrázku můžeme vidět vnější kružnici, pomocí které je možné rotovat kolem pohledového vektoru kamery. Tato kružnice vytváří plochu, jež bude vždy rovnoběžná s průmětnou kamery. Pro zvýšení názornosti je zde celá řada vylepšení. Jmenujme například barevné označení výseku úhlu, o který se objekt otáčí, zvýraznění směru rotace a celkem v tomto případě důležitá textová informace o velikosti úhlu. Jestliže budeme chtít rotovat se skupinou modelů, je potřeba umístit střed rotace na místo, odpovídající rozmístění objektů. Jsou-li tyto umístěny například do tvaru trojúhelníka, vypočítají se těžnice tohoto trojúhelníka a střed otáčení se umístí do jeho těžiště.

2.3.4 Manipulátor změny měřítka

Pro změnu měřítka objektu bude platit podobná soustava manipulátorů jako pro posun. Opět budou hrát roli příslušné osy, podle kterých budeme chtít zvětšovat či zmenšovat těleso. Takové nerovnoměrné změně se říká neuniformní. Naopak změna měřítka ve všech osách zároveň, jež se provede rovnoměrně na celém tělese, se nazývá uniformní.



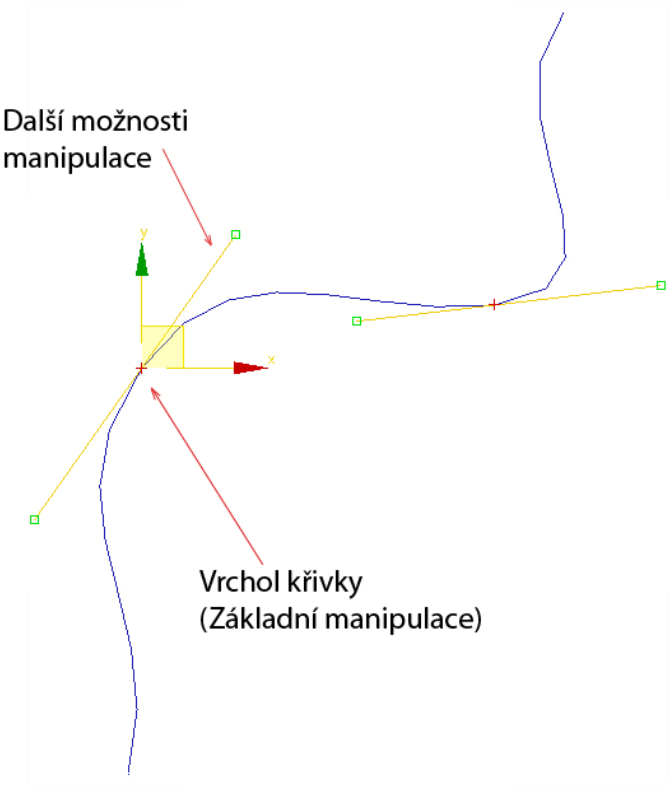
Obrázek 2.14 Manipulátor změny měřítka

Opět je zde možná změna měřítka v celé rovině *xy*, *yz* nebo *xz*, jež je naznačena spojením dvou příslušných os. Pro změnu velikosti ve všech osách je určen celý střed manipulátoru. I zde stejně jako u rotace záleží na umístění středu. Jak jsme si řekli v teorii transformací, při změně měřítka může docházet ke zvětšení či zmenšení, současně s tím k posunu v závislosti právě na umístění středu. Při aplikaci manipulace na více objektů se střed určuje stejným způsobem jako u případu rotace.

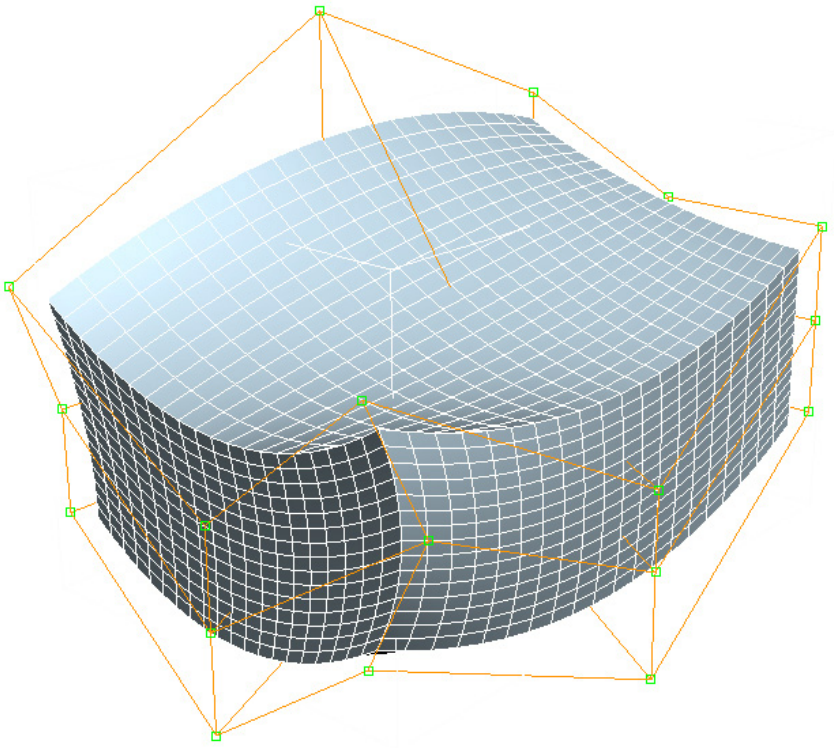
2.3.5 Další možnosti manipulátorů

Popsali jsem si nejzákladnější manipulátory a jejich možné tvary, samozřejmě při návrhu vlastních se bude tvůrce bude snažit o jejich použitelnost vzhledem k zaměření jeho práce. Můžeme se tedy setkat s manipulátory, jež z části využívají první tři deklarované typy nebo jsou navrženy rozdílně. Nakonec uvedu pár příkladů, kam manipulátory také zasahují. Na prvním obrázku můžeme vidět Beziérovu křivku. Manipulátor obsahuje jednak prvky pro posun celých vrcholů, navíc je vybaven pomocnými body, jež určují zakřivení apod. Jde stále o využití jedné potřeby. Pro definici tvaru tohoto druhu křivky je však takový manipulátor výhodou. Na dalším obrázku je klasický kvádr, na němž je

namapován další pomocný box. Aplikací manipulátorů na jednotlivé body pomocného objektu deformujeme i objekt uvnitř, ovšem za použití několikanásobně menšího počtu vrcholů.



Obrázek 2.15 Beziérova křivka s pomocnými vrcholy



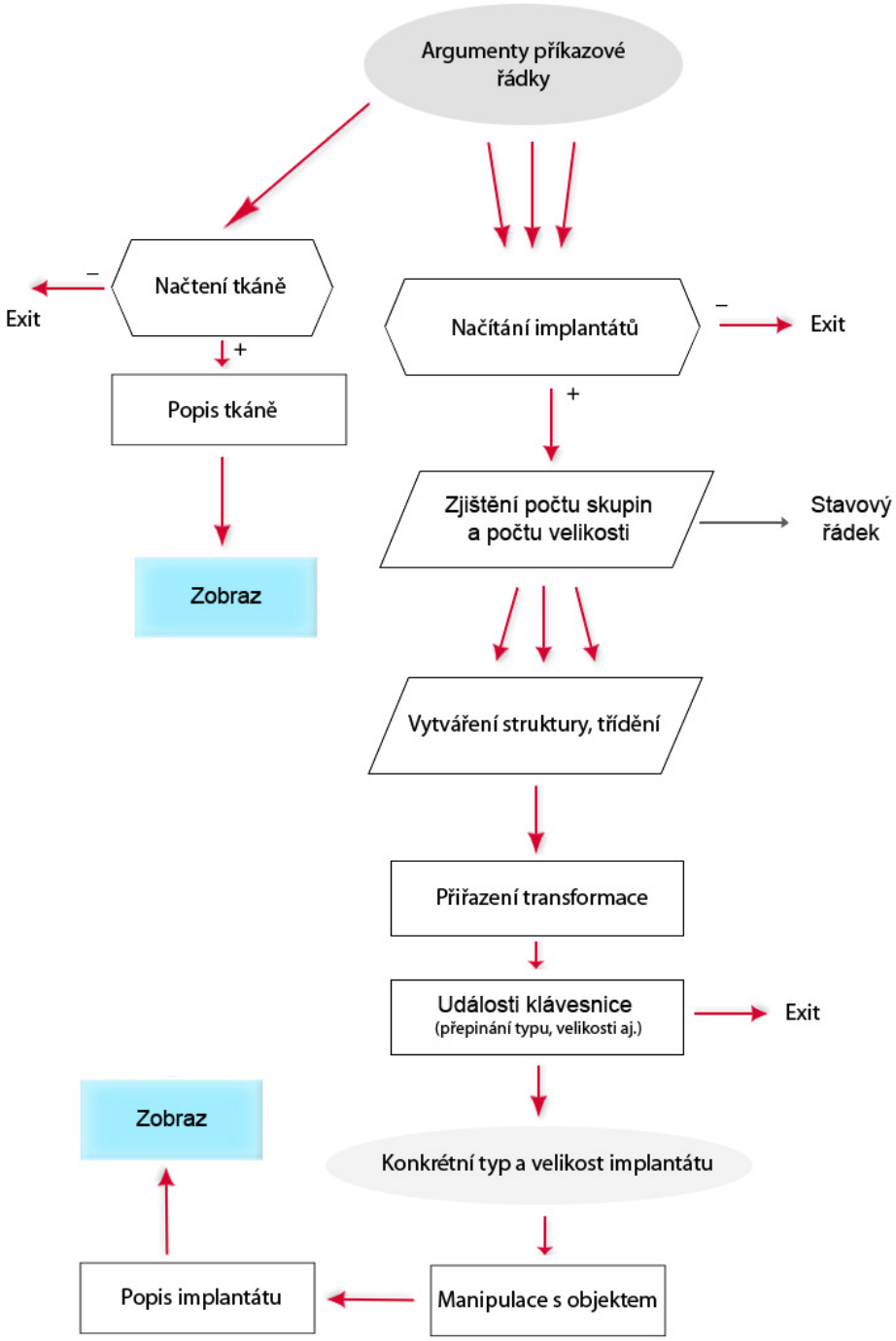
Obrázek 2.16 FFD box 3x3 manipulátor

3 Návrh

V kapitole návrhu je popsána problematiku zadaného projektu a jeho systematická řešení na obecné úrovni. První částí je vytvoření vývojového diagramu, jež nám pomůže pochopit fungování celého projektu. Principielně nám osvětlí sekvenci úkolů, které se v programu po sobě vykonávají. Následuje návrh uživatelského prostředí, kde rozeberu například volbu rozmístění oken v obraze a jejich smysl. Další podkapitolou je návrh grafu scény. Ten nám řekne něco o závislostech mezi jednotlivými prvky v projektu. Zde je důležité promyslet strategii vkládání a odebírání implantátů či třeba seskupování prvků grafu, jež spolu souvisí. Poslední část je zaměřena na návrh tříd a přidělení úkolů každé z nich. O všechny body návrhu se opírá kapitola následující, kde dochází k realizaci projektu podle bodů návrhu.

3.1 Vývojový diagram

Při práci na větším projektu je velmi důležité se v první řadě, než autor začne se samotnou implementací, nejdříve nad celou problematikou zamyslet a navrhnout logický postup. I já jsem se držel tohoto pravidla. Jak jsem již dříve poznal, je mimořádně žádoucí vytvořit v prvotní fázi jakýsi postup práce, říkejme mu vývojový diagram, kde může nejen autor, ale další kolegové vidět jednotlivé fáze programu, logiku celého návrhu a nad těmito fakty na abstraktní úrovni polemizovat. Takový diagram je často jen náčrtem na papíře, není třeba striktně dodržovat konvence pro kreslení vývojových diagramů, jde přirozeně pouze o pochopení myšlenkového pochodu. Vytvořený diagram na obrázku pod textem je pouze informativní z hlediska návrhu a návazností, komplexní řešení dílčích problémů s vazbou na použitý grafický toolkit si uvedeme později v části implementace.



Obrázek 3.1 Vývojový diagram

Aplikace je navržena tak, aby načítala libovolné množství implantátů, mezi kterými bude možno listovat přímo za chodu programu. Načtení se provede při startu z příkazové řádky, kdy podle jednotlivých parametrů dojde k rozeznání rolí souborů. Skupiny implantátů jsou odděleny parametrem *–i* a načítaná tkáň parametrem *–t*. Pokud něco z tohoto nelze načíst, protože třeba zadané soubory neexistují, aplikace končí. Jestliže vše proběhne bez komplikací, provede se zobrazení tkáně ze zadaného zdrojového souboru a soubory implantátů se načítají do struktur programu. Zde se

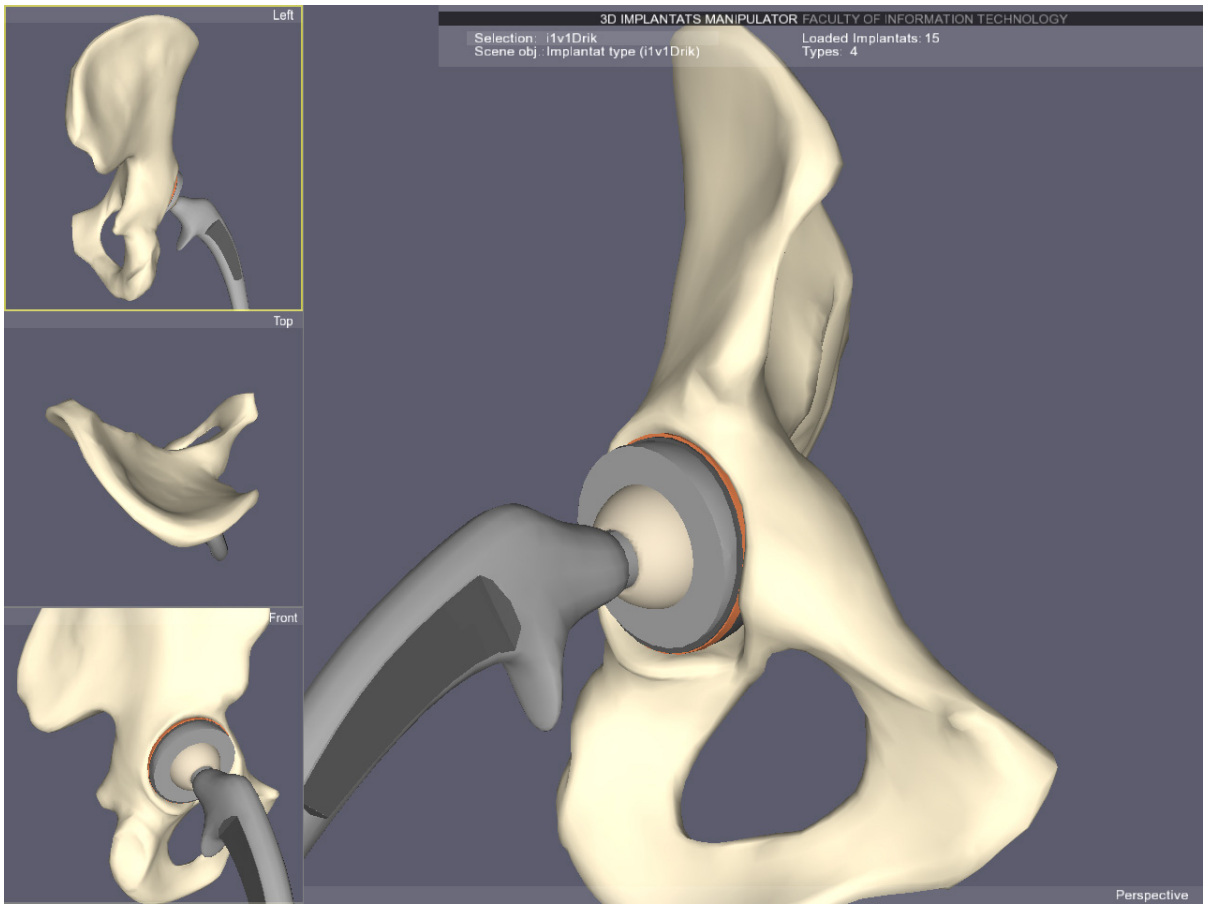
vyhodnotí informace typu kolik bylo regulérně načteno souborů, kolik typů implantátů apod. Tyto informace jsou relevantní pro další práci a zobrazujeme je na obrazovku.

Každému typu, tedy všem velikostem daného implantátu, je přiřazena transformace. Chtěl jsem totiž, aby při změně typu nebo velikosti implantátu byla zachována transformace (poloha a rotace) tak, jak jsme si je nastavili. Znovuustavení těchto parametrů při každé změně typu či velikosti by byla nežádoucí. Ačkoli je možné do scény přidávat libovolné množství implantátů, vybrán je vždy pouze jeden, na který se vztahuje manipulace. Po vložení dalšího implantátu se vztahuje automaticky manipulace na tento nově přidaný objekt. Změny objektu a transformace se neprodleně promítají na obrazovce.

3.2 Návrh uživatelského prostředí

Uživatelské prostředí programu pro manipulaci s 3D modely implantátů bude jistým způsobem kombinovat dvě možnosti. Jednak půjde především o grafické rozhraní, kde se odehrává podstata projektu. Na druhé straně bude velmi důležité rozhraní jednoduchého příkazového řádku, kde definujeme veškeré vstupní soubory, a podle něhož dojde také k roztřídění do příslušných skupin dle parametrů příkazu.

Samotný návrh uspořádání jednotlivých pohledů najde inspiraci v 3D aplikacích typu *3ds max*, *SoftImage* a jiných. Je koncipován s ohledem na použitelnost. Centrem dění bývá zpravidla perspektivní pohled, ve kterém se snadno přibližuje nebo rotuje kolem objektu. Důležité jsou však i doplňující okna. Typicky to bývá okno půdorysu, nárysu a bokorysu, kde může uživatel sledovat detail změny současně ve všech oknech. V návrhu jsem volil 3 tyto doplňkové pohledy na levou stranu, dominantní zůstává perspektiva. Jednotlivé pohledy je možno klikem myši přepínat a následně manipulovat kamerou pomocí kláves.



Obrázek 3.2 Ukázka uživatelského prostředí aplikace

Co se týče grafické prezentace, je volena s ohledem na zaměření decentně. Celé prostředí bylo tvořeno standardní cestou, tedy nejdříve ručně navrženo grafickým editorem, v tomto případě to byl Adobe Photoshop. Poté byly teprve vloženy jednotlivé prvky do samotné aplikace. Byly využity schopnosti knihovnen, jež dovolují uplatnit průhlednost, volby fontu a jiné. V horní části můžeme vidět panel nesoucí informace o označeném objektu, možnosti výběru, počtu načtených implantátů a počtu typů, do kterých se implantáty řadí.

3.3 Návrh grafu scény

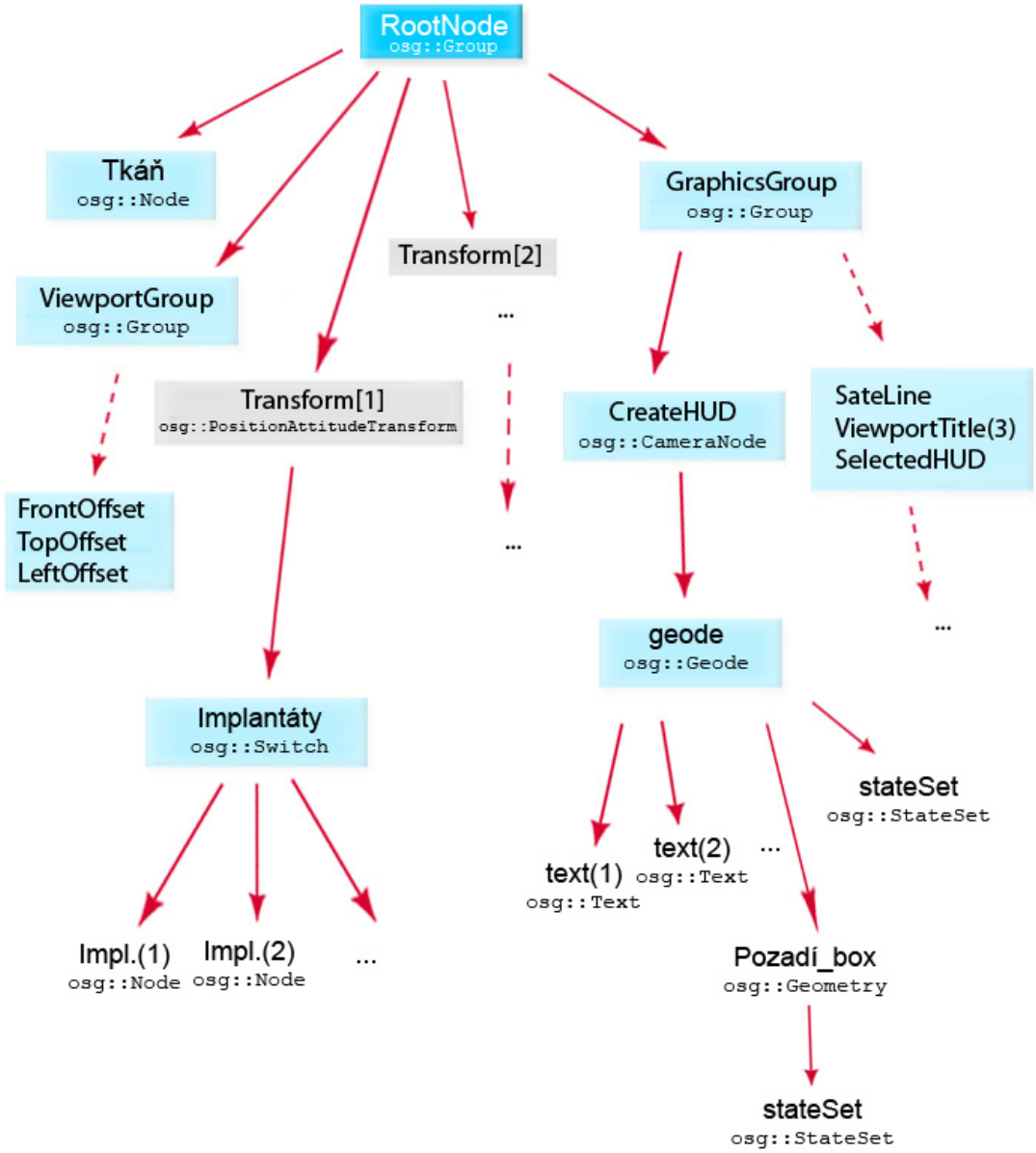
Při tvorbě takovéto aplikace je důležité, aby jednotlivé prvky v programu byly umístěny v nějaké hierarchii, kterou sice nevidíme, ale která řeší koncepci fungování celého programu. Taková hierarchie nám potom napoví něco o smyslu a závislostech jednoho prvku na druhém.

Vše začíná u kořene, ze kterého vychází několik větví a každá větev zapouzdřuje jednu skupinu prvků, jež se zase mohou větvit, už ale v rámci svého zaměření. Například větev *GraphicsGroup* podle obrázku pod textem zaštiťuje veškeré záležitosti grafické stavby prostředí a výpisů na obrazovku. Větví se na jednotlivé elementy viditelné na obrazovce až k listům, které popisují

jejich samotnou geometrickou stavbu. Další větev *ViewportGroup* bude orientována na nastavení kamer jednotlivých pohledů.

Hlavní částí grafu jsou větve nesoucí implantáty. Od kořene se zavěšují nejdříve jejich transformace, které mění pozici a rotaci všech svých potomků. Díky této strategii bude možné přepínat implantáty v prostředí s nezměněnou pozicí a rotací na místě původního. Nebylo by totiž žádoucí, aby po nastavení implantátu přímo k požadované tkání došlo při změně velikosti k resetu všech souřadnic. Potomka zavěšeného na transformaci si představme jako kontejner, ve kterém jsou uloženy implantáty jednoho typu, tedy všechny jeho velikosti.

Při vložení nového modelu se vloží celá tato větev včetně nové transformace a kontejneru s implantáty. Kontejner (typ implantátu) i jeho obsah (jednotlivé velikosti) se potom bude měnit na stávající transformaci. Při odstranění konečného implantátu je potom důležité odstranit celou odnož, která mu přísluší. Obrázek pod textem obsahuje doplňující informace, které budou kompletně osvětleny v části implementace.



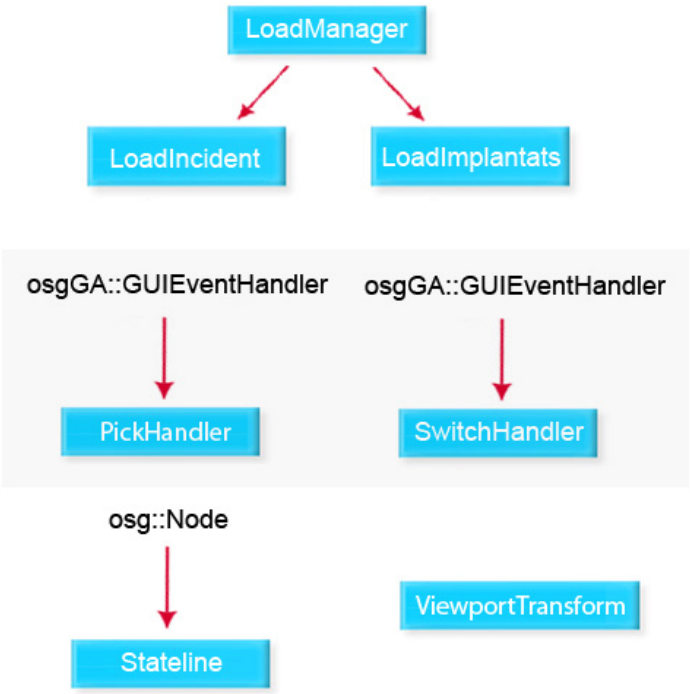
Obrázek 3.3 Graf scény

3.4 Diagram tříd

Program je vhodné rozdělit do několika tříd podle svého zaměření a přidělených úkolů. Jde-li o načítání souborů do programu, starají se o tento problém moduly *LoadImplantats* a *LoadIncident*, které jsou zděděny od mateřské třídy *LoadManager*. První z nich má za úkol vše spojené s načením implantátů, druhý provádí podobný proces s modelem tkáně.

Ošetření událostí bude předmětem řešení třídy *SwitchHandler*, která se stará také o veškerou manipulaci s objekty i kamerami. Doplňovat ji bude druhá třída, koncipovaná spíše na události od myši. Modul *StateLine* zaštiťuje záležitosti, spojené s grafikou a informativními výpisy. Pomocná

třída *ViewportTransform* pak provádí nezbytné operace pro nastavení kamer. Dohromady je tedy navrženo sedm tříd. Implementaci každé z nich se budeme podrobněji věnovat v následující kapitole.



Obrázek 3.4 Diagram tříd

4 Implementace

V této části dokumentace se budu věnovat jednotlivým implementačním segmentům v programu. Popíši zde smysl a úkoly všech navržených tříd a dílčí řešení jednotlivých problémů po návrhu. Nejdříve zde ale rozeberu výběr jazyka a knihovny, které při implementaci využiji. Alespoň okrajově nastíním použití některých elementů vybraného modulu, jež celý program provází. Samotná implementace programu byla napsána v univerzální programovací aplikaci Microsoft Visual Studio 2005. Umožnila mi využít veškeré potřebné nástroje programovacího softwaru včetně propracovaného debug módu a přehledného popisu veškerých zdrojových kódů.

4.1 Jazyk C/C++

Pro implementaci praktické části bakalářské práce byl vybrán univerzální programovací jazyk C++, jež umožňuje objektové programování. Je předem jisté, že projekt bude objektově koncipován. Velmi se nám zde hodí vlastnosti a schopnosti, které samo objektové programování zastřešuje. Při tvorbě je snadné využít znalostí získaných z programování v jazyce C. Objektově pojatý kolega zaštiťuje celý tento jazyk, k tomu nám nabízí vlastnosti jako zapouzdření, dědičnost a polymorfizmus. Z hlediska tvorby většího programu je pro nás důležité například zapouzdření jednotlivých tříd, jež dekomponuje projekt na menší celky, které budou mít jasně definované rozhraní. Využití si zde najdou i jiné schopnosti tohoto velmi rozšířeného jazyka včetně dědičnosti apod., s tím souvisí i možnost obohacení právě jazyka C++ některým *toolkitem* pro samotnou práci s grafikou.

4.2 OpenSceneGraph

OpenSceneGraph (OSG) je volně přístupná rozsáhlá knihovna, jež je primárně určena pro 3D vizuální simulace, virtuální realitu či kupříkladu vývoj her. Svoje uplatnění dokazuje mohutně rostoucím zájmem z řad vývojářů 3D aplikací a při realizaci projektu jsem ho volil i já ve své práci. Toolkit je nadstavbou vývojářům dobře známé knihovny *OpenGL*, kterou sám zastřešuje. Není tudíž problém při potřebě programátora, sáhnout po konstrukcích právě z tohoto *API*. Knihovna je multiplatformní a je podporována na systémech Microsoft Windows, OSX, GNU/Linux, IRIX, Solaris a FreeBSD.

OpenSceneGraph, jak už z názvu vyplývá, reprezentuje 3D svět jako graf složený z jednotlivých komponent ve scéně. Tento graf je vždy orientovaný a acyklický. Můžeme si ho představit jako n-nární strom, který má kořen nahoře. Při cestě od vrchu pak procházíme přes slučující prvky, podgrafy a transformace k listům, kde jsou uloženy geometrie jednotlivých elementů scény a některá nastavení.

Na obrázku 3.3 v kapitole návrhu můžeme vidět konkrétní graf scény naší aplikace. Nelze jednoznačně tvrdit, v jakých částech programu se celý tento strom vytváří a přestavuje. Dá se říci, že do jeho struktury zasahují všechny deklarované třídy, centrálně je ale konstruován v hlavní části programu *main*. Kořen grafu scény lze předávat jako parametr do různých částí programu, což umožňuje jednoduchou manipulaci s celým obsahem. Zvláště pak v části přepínání typů a velikostí implantátů dochází k přeměně jeho struktury velmi často.

Knihovna OSG dále obsahuje velmi rozsáhlou hierarchii tříd a metod, které jsou programátorovi k dispozici. Já jsem se při svém návrhu dotkl jen malé části možností, které toolkit OpenSceneGraph nabízí.

4.3 Prvky OSG

Než budu pokračovat v popisu jednotlivých tříd, je nezbytné alespoň ve stručnosti představit základní elementy knihovny OpenSceneGraph, které byly při implementaci využity. Celým programem jsou propleteny tzv. chytré ukazatele `osg::ref_ptr<>`. Využívám jich z prostého důvodu, a sice že veškeré alokované entity svázané s tímto ukazatelem se samy při ukončení programu dealokují a odstraní. Kdybych tuto konstrukci nepoužíval, bylo by potřeba činit tak ručně a to také pro veškeré výjimky, které můžou za běhu programu nastat.

Třída `osg::Group` slouží mimo jiné pro napojování konečných uzlů `osg::Node`. Vytváří tak jakýsi kontejner, na který jsou navěšeny konečné uzly či další kontejnery atd. Podobně tak lze přidávat do kontejneru `osg::Geode` (Geometric Node), který je současně sám listem, objekty, jež jsou viditelné. Jeho metody jsou schopny například vytvořit kolem objektu ohraničující kvádr (bounding box) nebo kouli (bounding sphere).

Další entitou, která se nám hodí při řešení úlohy je přepínač `osg::Switch`. Slouží v této aplikaci jako kontejner pro implantáty. Má totiž výhodu v jednoduché obsluze, kdy ovládáme jednou metodou aktivitu, jinou zobrazení apod. Jednou z použitých tříd je `osg::Text`, pomocí kterého řešíme výpisy na obrazovku. Ten se volí obvykle jako potomek instance třídy `osg::Geode`, která je zase potomkem třídy `osg::CameraNode`. Řešení grafu scény bude věnována jedna podkapitola, názorně pak budou vidět návaznosti jednotlivých prvků celého grafu.

Důležité jsou i třídy pro nastavení a manipulaci s kamerami a samotné zobrazení. Je zde vytvořen zase jakýsi kontejner `Producer::CameraConfig`, jež zastřešuje použití a nastavení jednotlivých kamer `Producer::Camera`, které vykreslují svůj podgraf. Kontejner `CameraConfig` se pak vkládá jako parametr pro nastavení samotného vieweru programu `osgProducer::Viewer`.

Implantáty a kamery je potřeba transformovat, OpenSceneGraph je založen na principech OpenGL, tudíž lze k popisu transformace použít matici 4x4 prezentovanou třídou `Osg::Matrixd`. Jiný způsob de definovat změnu polohy či rotace přímo parametrem. Pro posun se využívá definice

parametrů `osg::Vec3` pro třídu `osg::PositionAttitudeTransform`, jež se kompletně stará o translaci i rotaci. V případě otočení je potom dobré využít k definici rotace quaternion `osg::Quat`. Metody třídy `osg::PositionAttitudeTransform` jsou přirozeně přetížitelné, zmínil jsem pouze některé možností použití parametrů. Představili jsme si alespoň elementárně některé prvky modulu OSG, které v této práci byly využity, nutno však dodat, že knihovna obsahuje velmi širokou škálu tříd a metod pro různá využití.

4.4 Části programu a popis tříd

Nyní, když už máme přibližnou představu o modulu OpenSceneGraph, si rozebereme úkoly jednotlivých tříd, jejich odvození a řešení dílčích problémů projektu. Kapitola se zabývá oblastmi, jako je načtení modelů, vytvoření grafického prostředí či zpracování událostí od myši a klávesnice, uvedu zde ale také něco o nástrojích, které jsem využíval k dokumentaci kódu a verzování. Poslední částí je tvorba implantátů, což není ryzí téma této kapitoly. Při realizaci aplikace však hraje velmi důležitou roli.

4.4.1 Načítání

První kapitolou, kterou jsem se zabýval, a byla elementární z hlediska dalšího vývoje aplikace, bylo načítání jednotlivých souborů do prostředí. Připomeňme, že knihovně OSG nečiní problém pracovat s velkým množstvím formátů (až na 45 druhů) souboru, mezi něž patří např. *.3ds, .stl, .obj, .wrl, .osg*. Problematiku načítání řeší podle návrhu třída *LoadManager*, která má dva potomky, jež dědí nastavení cest do příslušných adresářů. Později by byla třída využita pro další možnosti, např. volba souborů z průzkumníku apod.

Zděděná třída *LoadIncident* slouží pouze k načtení modelu tkáně, která má být ze zásady statická. Ukládá se takový soubor, jež následuje za tagem *–t* v příkazové řádce. Metoda jež se stará o načtení, vrací do hlavního segmentu programu ukazatel na uzel *Node*, který je následně přidáván do grafu scény. Druhá zděděná třída *LoadImplantats* má trochu zajímavější úkol. Dochází zde k načtení veškerých implantátů z příkazové řádky, tedy všech velikostí každého typu. Ukládání se provádí do připravené struktury `osg::Switch`, kterou také metoda vrací do hlavní funkce. Současně s tím se provádí informativní výpis o skutečně načtených souborech a evidují se hodnoty o počtech typů a velikostí v každém z nich. Tyto informace budou pak důležité pro jejich volbu a přepínání v simulaci.

4.4.2 Grafické prostředí

Kromě promítání modelů implantátů a tkáně se na obrazovce objevují i další prvky, jež souhrnně vytvářejí samotné grafické prostředí aplikace. Při pohledu na obrázek uživatelského prostředí

v kapitole návrhu zjistíme, že jsou zde tři pomocné pohledy a jeden majoritní. Každé z pomocných oken je vždy ohraničeno poloprůhledným rámcem a při změně aktivního pohledu se zobrazí žlutý rámeček, jež informuje o aktivitě. Mimo to všechny čtyři pohledy obsahují popisek (Left, Top, Front, Perspective). V okně perspective se potom ještě nachází informační panel, nesoucí údaje o načtených implantátech a možnostech selekce.

O celou tuto reprezentaci se stará třída *StateLine*. Je zřejmé, že zmíněné rámečky a pozadí pod popisky jsou tvořeny zase geometrickými útvary, které vychází z možností knihovny OSG.

Obrázek 4.1 Horní informační panel

V případě pozadí textových popisků se jen zjednodušeně jedná o definici polygonu, jež má nastaveny pozice vrcholů právě tak, aby s přesahem ohraničovaly text či námi zvolenou oblast. Nejdříve se deklaruje instance OSG třídy `osg::Geometry`, která využívá pole nastavených pozic vrcholů. Následuje nastavení normály, jež je určena tak, aby byl polygon vidět od pozorovatele, tedy směřovala proti nám. Procedura vytváření statického pozadí pokračuje volbou barev, průhlednosti a několika dalších stavů. Důležitou roli zde hraje kamera vykreslující právě svůj podgraf, jež určuje celkovou matici o rozměrech okna. U textu je situace obdobná s rozdílem, že nedefinujeme jednotlivé vrcholy, ale modul OSG nám dovolí přímo nastavit font, pozici, zarovnání, rozlišení, barvu a další parametry. Takto jsou volány jednotlivé metody podobného charakteru starající se např. o celý horní panel, jiná o popisky oken atd. Každá metoda pak vrací jednu kameru, které máme možnost přiřadit masku.

4.4.3 Maskování

Aplikace pro manipulaci s 3D modely implantátů je tvořena, jak již bylo řečeno, pouze rozhraním OSG. Jestliže používáme více pohledů (viewportů) ve scéně, je potřeba říci, co v jakém výřezu vidět bude a co ne. Standardně je každý nový pohled jakousi kopií hlavního okna, kde je viditelné vše. Je však na nás upravit si jeho velikost, úhel pohledu a to, co vlastně má zobrazit. Nejde samozřejmě o samotné modely implantátu a tkáně, u nichž je v našem zájmu, aby byly viditelné všude. Na mysli máme popisky určené pro konkrétní okno, ohraničení výřezů apod. Toto se řeší v toolkitu OSG pomocí masek, jež jsou přiřazovány jednotlivým uzlům. Masky chápeme jako binární, jestliže chceme v pohledu uzel vidět, nastavujeme na příslušné pozici jedničku. Podle toho, kolik bude zapotřebí různých možností a kombinací, je vhodné udělat si tabulku, kde pak není problém masky kombinovat.

Nastavení masky	Viditelné ...
0000 0001	Pouze v okně Perspective
0000 0010	V okně Top, Fron, Left
0000 1000	Pouze v okně Front
0001 0000	Pouze v okně Top
0010 0000	Pouze v okně Left
0000 0100	Aktivní viewport

Obrázek 4.2 Tabulka zvolených masek

V tabulce je znázorněno, jak byly nastaveny masky při řešení tohoto problému. Jednotlivé výřezy si nastaví hodnotu, jež reprezentuje objekty, které chceme vidět. Jestliže je mezi sebou potřebujeme kombinovat, vytváříme logický součet binárních hodnot. Chceme-li například zobrazit okno *Top* a má být současně aktivní, čili bude-li ohraničeno žlutým rámcem, výsledná hodnota je dána logickým součtem hodnot z tabulky 00010000 a 0000100, tedy 00010100. Podobným způsobem je to řešeno i pro ostatní případy.

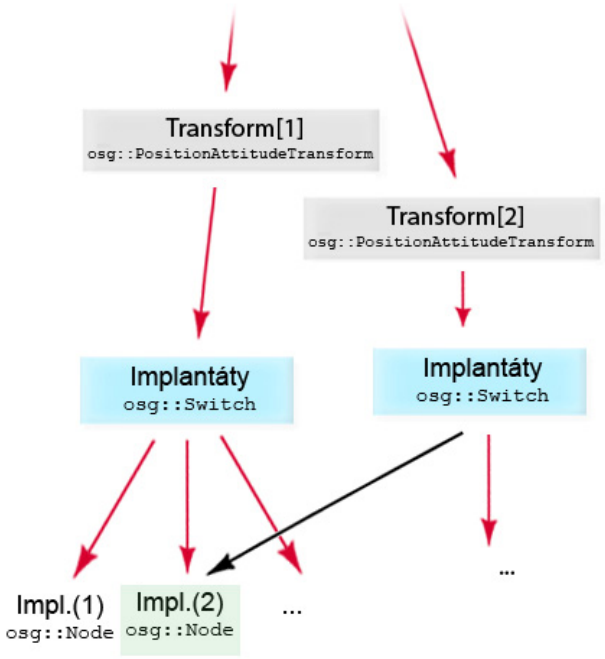
4.4.4 Zpracování událostí

Ke zpracování událostí programu slouží dvě třídy s různým zaměřením. Třída *SwitchHandler* je orientována výhradně na práci s klávesnicí a zastřešuje přidávání a přepínání implantátů, manipulaci s nimi a pohyb kamer. Druhá třída *PickHandler* je orientována spíše na události vyvolané myší. Nejdůležitějším jejím úkolem je vypočítávat průsečíky scény a myši, zjišťovat objekty a vyvolávat výběr modelu a pohledu kamery.

První zmíněná třída řeší problematiku stisku kláves, jež vyvolávají události. Jako první věc je přepínání jednotlivých modelů. Strategie řešení je volena s ohledem na možné vysoké hardwarové nároky. Modely implantátů mohou být velice detailní a tedy i náročné na paměť, proto je velmi důležité, nad problematikou popřemýšlet. Načtené implantáty v hlavním programu, jež nás informují o počtech v příslušných skupinách, se setřídí v této třídě co nejúsporněji tak, aby v programu figurovala pouze ta skupina implantátů, se kterou chceme pracovat. Ostatní načtené objekty není potřeba v grafu scény uchovávat. Při přepnutí skupiny pak dochází k odstranění staré skupiny se všemi náležitostmi a navázání nově zvolené. Je však v našem zájmu zachovávat transformace, kterých jsme už předtím docílili, odstranění skupiny tak dochází až za ní. Přepínat modely lze samozřejmě pouze v mezích hodnot, jaké jsme získali ze třídy *LoadImplantats* po načtení z příkazové řádky.

Problematika přidávání dalších implantátů není tak docela triviální, musíme se smířit s několika fakty ohledně jedinečnosti modelů a oprostit se z části od paměťově minimálně náročné

aplikace. Původní řešení spočívalo ve vkládání nových modelů, jež reprezentoval vždy jeden objekt. Ukazatele všech přepínačů `osg::Switch`, pokud byly například všechny nastaveny na první typ a první velikost, ukazovaly současně na jeden originální model s jednou adresou. Transformovat lze každý model zvlášť. Problém však nastal ve chvíli výběru objektu, kdy se označí vlastně všechny vložené objekty, jež mají právě adresu modelu. Černá šipka na obrázku značí ukazatel na stejné místo, na které již má nastavenou adresu jiný přepínač implantátů.



Obrázek 4.3 Nesprávné řešení problému přidávání implantátů

Řešením bylo, při každém vložení skupiny implantátů, vytvořit kopie původně načtených entit, které mají však vždy různou hodnotu ukazatele. Jinými slovy se do paměti nahrává model znova, docílili jsme tím ale jedinečnost každého z modelů.

Při přidání nového modelu se vkládá také nová transformace s hodnotami nastavenými v počátku a nulovou rotací. Jestliže chceme s objektem manipulovat, je toto řešeno pomocí klávesnice, kdy pro změnu pozice se nastavují parametry OSG metody *setPosition*. Pohybovat lze objektem ve třech osách po předem definovaných krocích. Podobně tak je využita metoda *setAttitude* pro nastavení rotace.

Dalším úkolem pro třídu *SwitchHandler* je pohyb kamer v pomocných pohledech. Děje se tak vždy v právě označeném výřezu, o jehož aktivitě nás informuje parametr `ActiveViewportFlag` ze třídy *PickHandler*. Posuv se provádí zase o předem definovaný krok a pohybujeme se opět ve třech osách. Kompletní manipulace je zatím věcí pouze klávesnice, je však předmětem dalšího vývoje, přiblížit se pohodlnosti práce s myší za použití jiných manipulátorů. O možnostech manipulace všech

pohyblivých elementů a kompletní obsluze programu se může uživatel dočíst v nápovědě stiskem klávesy *h*.

Sesterská třída *PickHandler* má za úkol vše, co je nějakým způsobem spojeno s myší. Jedná se především o možnost výběru libovolného objektu, kterou provádí uživatelská metoda *Pick*. K samotnému výpočtu intersekce myší se scénou slouží knihovní třída `osgUtil::PickVisitor`, které se předává množství parametrů včetně přepočítaných koordinát a projekční matice. Nejprve jsem průsečíky řešil jiným způsobem, a to ručně s aktuálními koordináty a seznamem protnutých objektů. Narazil jsem však na problém, kdy aplikace padala na některých procesorech, ač celý zápis byl podle norem. To bylo však samozřejmě nepřípustné a tak jsem využil metody výše zmíněné.

Označit nebo odznačit objekt je možné pouze v případě, nezmění-li se během stlačení a puštění tlačítka koordináty. Jestliže dojde k posunu myši během této doby, tedy proměnné zapsané při stisku se nerovnají proměnným při uvolnění, výběr zůstává nezměněn. Jde pouze o rotaci kamery kolem scény.

Možnost přepínat jednotlivé výřezy oken je také řešeno třídou *PickHandler*. Při kliknu v obrazovce se vyhodnocuje, zda-li nespadají souřadnice kliku do oblastí určených pomocným oknům. Jestliže ano, dojde k aktualizaci hodnot masek a nastavení příslušného bitu pole `ActiveViewportFlag`, jež ve třídě *StateLine* způsobí „rozsvícení“ okna, které jsme požadovali. Odstraňování modelů je také jedna ze schopností této třídy. Děje se tak pouze nad modely implantátu, ostatní objekty jsou rozpoznány, není však v zájmu je označovat. Samotné odstranění se děje poměrně složitou konstrukcí, kdy je potřeba z rozpoznaného objektu určit příslušnou větev grafu scény, na jejímž konci je model zavěšen. Není rozhodně obratné a úsporné, odstraňovat pouze listový prvek, je potřeba odalokovat veškerou paměť určenou pro transformace a přepínače samotných implantátů.

4.5 Doxygen a Subversion

Při samotné implementaci je třeba dbát na určité zásady, které by měly být programátorovi vlastní. Ve výsledku je sice vidět především aplikace, samotný vývoj ale doprovází několik náležitostí. Z hlediska dalšího rozvoje aplikace je velmi důležité samotnou implementaci řádně komentovat a vytvářet současně programovou dokumentaci. Ušetříme si tím nemálo času a dodáme do samotného kódu dávku přehlednosti a porozumění. Projekt je navíc pro další vývoj určen, je tedy důležitý jeho komplexní popis a komentář implementace. Při vývoji byl použit volně dostupný software pro dokumentaci kódu *Doxygen.* Je to poměrně rozsáhlý produkt, který je určen pro jazyky C, C++, Javu, IDL, PHP a C#. Program zcela automaticky vytváří HTML prezentaci, která v několika sekcích zobrazuje jednotlivé segmenty kódu, jejich provázání s grafy a samotné komentáře, zapsané v implementaci.

Další věcí, kterou je vhodné brát v úvahu, je správa zdrojových kódů a verzování. Je to jeden z pilířů týmové spolupráce. Za tímto účelem byl využíván volně dostupný program *Subversion*, který pracuje na principu *CVS*, jeho možnosti rozšiřuje a upravuje. Vytváří centrální repozitář, který je dostupný týmovým pracovníkům. Nástroj umožňuje například větvení, sledování změn v čase, poznámkování a mnoho dalších funkcí použitelných pro komplexní správu zdrojových kódů.

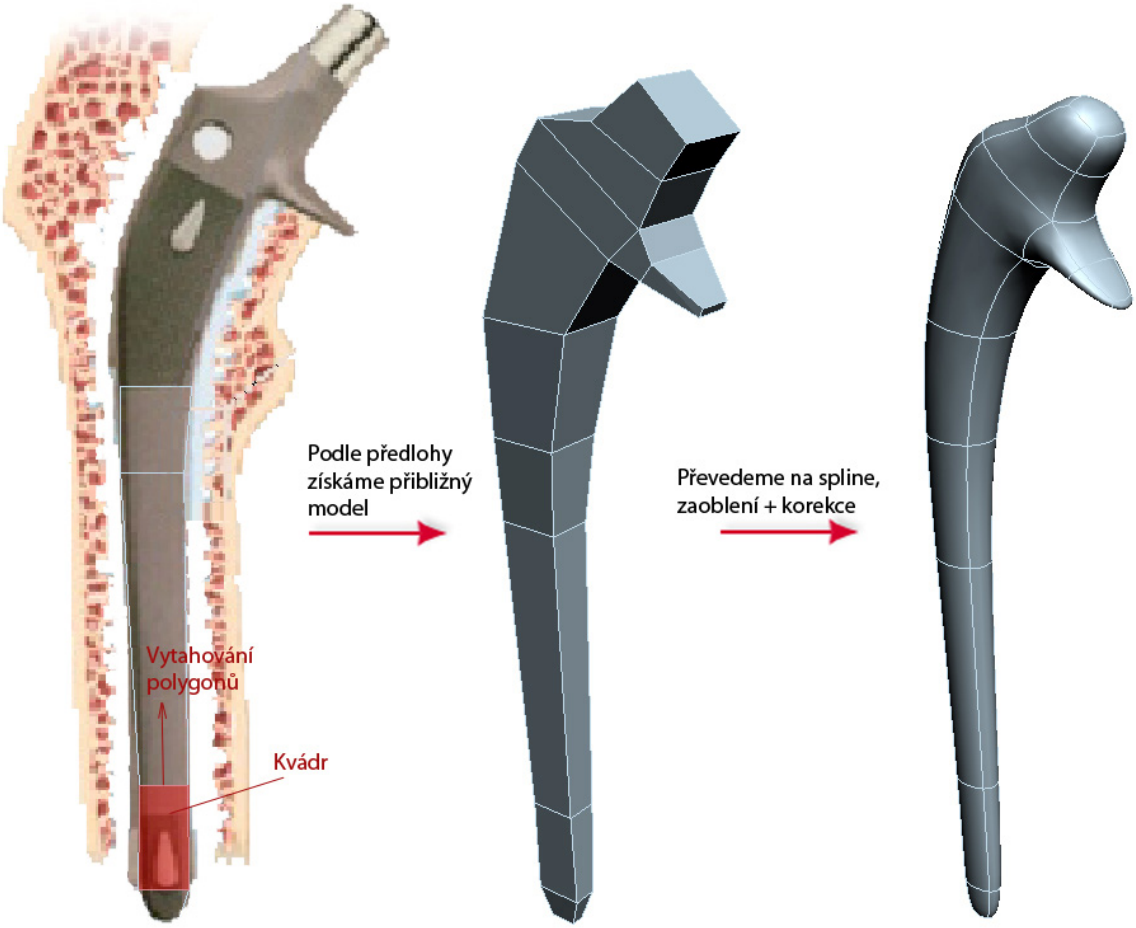
4.6 Tvorba implantátů

Jedním z bodů zadání je i samotné vytvoření implantátů, jež budou aplikaci prezentovat. Program je samozřejmě univerzální k použití na různých částech lidské kostry, je však použitelný i v úplně odlišných kruzích. Fantazii se meze nekladou, a tedy není problém aplikaci nasměrovat třeba pro konstruktérství automobilu atd.

Primárně je aplikace určena pro plánování chirurgických operací. K simulaci jsem volil a následně nastudoval problematiku implantace kyčelního kloubu, u něhož je předoperační plánování velmi důležité. Při návrhu implantátů jsem byl nucen sáhnout po některých dokumentacích a fotografiích reálných nástrojů. Podotýkám, že vytvořené modely jsou navrženy podle skutečných implantátů, jejich rozměry a parametry však nejsou zcela přesné. Slouží zde pouze jako prezentace a při pozdějším použití by nečinilo žádný problém reálné předměty v počítači dovést k dokonalosti, rozšířit jejich nabídku apod.

Při návrhu implantátů jsem využil mě důvěrně známou aplikaci *3ds max*, která svým zaměřením a rozsahem schopností rozhodně neomezuje grafika ve vytvoření potřebných modelů. Ba naopak je pro jakýkoliv návrh a modelování 3D objektů více než vhodná. Jen ve stručnosti zde poukážu, jakým způsobem se modely vytvářely. Tato látka se nám zde hodí, jelikož můžeme mimo jiné sledovat návaznost v praxi na veškeré okruhy teoretické části této práce.

Existuje celá řada modelovacích technik, kterými lze zdárně dojít k výsledku. Já jsem použil nejjednodušší z nich, tedy vytvoření modelu z pouhého kvádru a následným vytahováním polygonů a vrcholů do příslušných pozic. Za pomocí manipulátorů dostupných v aplikaci transformujeme objekt do výsledného tvaru, čímž vytvoříme typický příklad hraniční reprezentace polygonového modelu. Když je hrubý model vytvořen, zbývá převést ho na prezentaci pomocí křivek. Docílíme tím zaoblení a realističtějšího vzhledu, nezapomínáme však, že i tento model je ve výsledku zase tvořen trojúhelníky. Další postup při modelování hlavice je podobný, využíváme dále booleovské operace, pomocí nichž odečítáme z modelu jiné objekty apod. Počáteční postup je velmi zkráceně ukázán na následujícím obrázku. Modely implantátů jsou následně exportovány do formátu *3ds*, se kterým si modul OpenSceneGraph bez problému poradí.

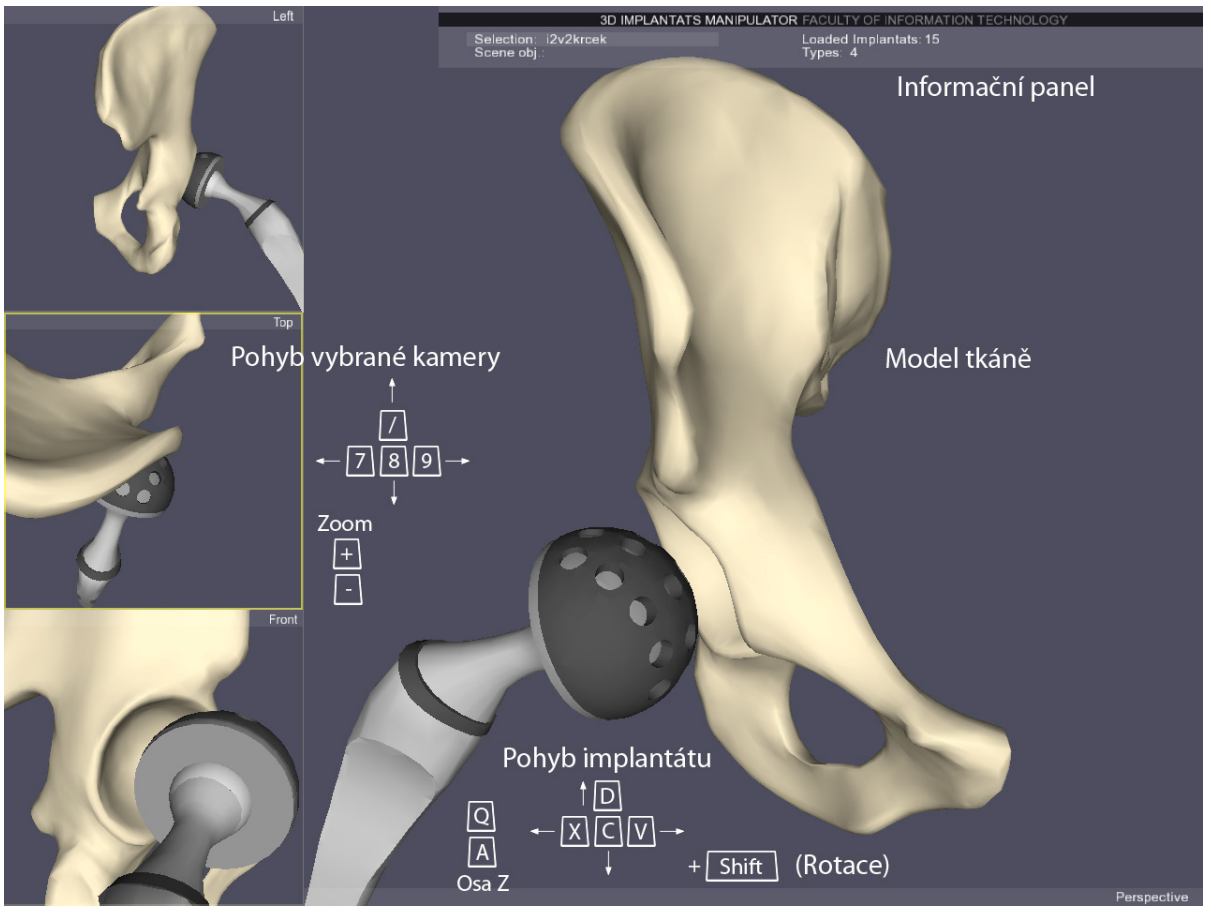


Obrázek 4.4 Postup při modelování implantátů

5 Výsledky

Výsledkem celého návrhu a implementace je program, který je schopen nasimulovat použití vybraného implantátu. Tyto implantáty jsou vymodelovány v externí aplikaci a do programu jsou načítány z příkazové řádky. Podle parametrů jsou roztříděny do příslušných skupin podle typů a velikostí. Po spuštění aplikace je možné přepínat šipkami jednotlivé skupiny či velikosti modelů, a dále s nimi manipulovat. Posun se pak děje pomocí kláves ve třech osách. U rotace jde o stejná tlačítka, pouze se současným stiskem klávesy *Shift*. Do scény lze přidávat libovolné množství implantátů a po výběru myší lze také libovolný z nich odstranit.

Situaci ve scéně je možno pozorovat ve čtyřech nezávislých pohledech. Ve třech pomocných, jež lze přepínat, můžeme pohybovat kamerou pomocí kláves a v hlavním perspektivním pohledu prohlížíme scénu myší. V horní části grafického prostředí je umístěn průhledný panel, jež informuje uživatele o počtu načtených implantátů a počtu typů. Dále nese informace o rozpoznaném objektu a jeho konkrétní části, na které právě leží ukazatel myši a vypisuje označený objekt. Uživatelské prostředí je zobrazeno na ilustraci 3.2 v kapitole návrhu. Pod tímto textem můžeme vidět také ukázku z aplikace, je však obohacena o jednoduché popisky, jež názorně vysvětlují možnosti transformace implantátu i pomocné kamery.



Obrázek 5.1 Ukázka aplikace s možnostmi pohybu

6 Závěr

Aplikace pro manipulaci s 3D modely implantátů prošla typickým vývojem, jež navrhlo moderní softwarové inženýrství. Během vytváření projektu bylo třeba nejprve stanovit reálné cíle vzhledem k možnostem a okolnostem. Tyto cíle byly, podle mého názoru, splněny, je však ještě poměrně dlouhá cesta k vytvoření komplexní aplikace, jež bude prakticky využitelná přímo před samotným operačním aktem. Proces vývoje celé práce jsem se snažil zachytit v této dokumentaci, která by měla postihovat veškeré etapy návrhu i implementace s návazností na teoretické základy rozebírané na samotném začátku publikace.

K realizaci úlohy byla zvolena knihovna *OpenSceneGraph* pro jazyk *C++*, jež pro mě byla dříve velkou neznámou. Po praktických zkušenostech s tímto modulem, kdy bylo třeba využít různé techniky k docílení požadovaného efektu, můžu říci, že pro mě samotné poznání bylo velkým přínosem. Ověřil jsem si v programátorské praxi svoji znalost o 3D grafice, a i samotný objektový návrh programu mě pro další práci velmi obohatil.

Další etapy práce na této aplikaci jsou věcí budoucna. Můžu zde ale poukázat, jakým směrem by se měl projekt dále vyvíjet. První důležitou věcí je řešení manipulace s objekty, která by využívala klasické manipulátory ovládané myší. Toto není zatím implementováno, jelikož byla očekávána nová verze knihovny *OpenSceneGraph*, která by problematiku manipulátorů měla řešit. Ta vychází současně s termínem pro odevzdání bakalářské práce, nebude tedy problém rozšíření provést v zápětí. Své výhody bude mít jistě i začlenění celého prostředí do nějakého toolkitu typu *wxWidgets*, jehož použití už je ale, mimo jiné, věcí cílové platformy. Další úloha pro nadcházející vývoj se týká samotných implantátů. Jestliže by měl být software použitelný v praxi při plánování chirurgických operací, je nezbytná absolutní přesnost modelů a rozšíření jejich nabídky.

Dovedu si představit ale i jiná vylepšení, jež by člověku při plánování operace mohla velmi napovědět, a která by mohla do praxe tato aplikace přinést. Jde o možnost zprůhlednit objekty či provést profilový řez, jež by informoval chirurga o množství a tloušťce materiálu, což za běžných okolností není viditelné. Tato myšlenka mě zaujala a její návrh si můžete prohlédnout na obrázku v příloze 1. Vylepšení je možné i u pomocných pohledů, kde je praktické umístit mřížku (tzv. grid), která informuje o měřítku a skutečné velikosti objektů. Možnost přepínání dalších pohledů zezdola a zprava je pak už pouze detail.

Domnívám se, že po realizaci výše zmíněných úprav by aplikace byla celkem schopným nástrojem pro předoperační plánování. Mohla by tak být využita v celé řadě případů nejen v oblastech medicíny, ale i v úplně odlišných sférách počítačové simulace.

Literatura

[1] ŽÁRA, J. – BENEŠ, B. – FELKEL, P.: *Moderní počítačová grafika*. 1. vyd. Praha, Computer press 1998, 448 s., ISBN 80-7226-049-9.

[2] DRASTICH, A.: *Zobrazovací systémy v lékařství*. 1. vyd. Brno, Rektorát VUT v Brně 1990, 512 s., ISBN 80-214-0220-2.

[3] VIRIUS, M.: *Od C k C++*. 2. Vyd., Kopp nakladatelství České Budějovice 2004, 217 s., ISBN 80-7232-110-2.

[4] SHREINER, D. – WOO, M. – NEIDER, J. – DAVIS, T.: *OpenGL Průvodce programátora.* 1. vyd. Brno, Computer Press, 680 s., ISBN 80-251-1275-6.

[5] Tým počítačové grafiky pro medicínu: *OpenSceneGraph tutoriál*. Fakulta informačních technologií v Brně, VUT, 2006.

[6] KRŠEK, P.: *Reprezentace 3D objektů*. Materiál k přednášce kurzu IZG, FIT VUT v Brně, https://www.fit.vutbr.cz/study/courses/IZG/private/lecture03/izg_3d_objekty_1.pdf,

[7] KRŠEK, P.: *Geometrické transformace ve 2D a 3D*. Materiál k přednášce kurzu IZG, FIT VUT v Brně, https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/ izg_slide_transformace_ print.pdf.

[8] KRŠEK, P.: *Reprezentace 3D objektů, 2.část*. Materiál k přednášce kurzu IZG, FIT VUT, https://www.fit.vutbr.cz/study/courses/IZG/private/lecture03/izg_3d_objekty_2.pdf.

[9] BioVendor Klinické produkty: *Operační postup ANA.NOVA*, Implantace kyčelního kloubu. www: http://kp.biovendor.cz/core/get_file.php?file=Operacni_postup_ ANA.NOVA.pdf.

[10] MOCEK, T: *Efektivní manipulace s objekty ve 3D*. [Diplomová práce]. Fakulta informačních technologií v Brně, VUT, 2006.

[11] Wikipedia encyklopedie. www.wikipedia.org

Seznam příloh

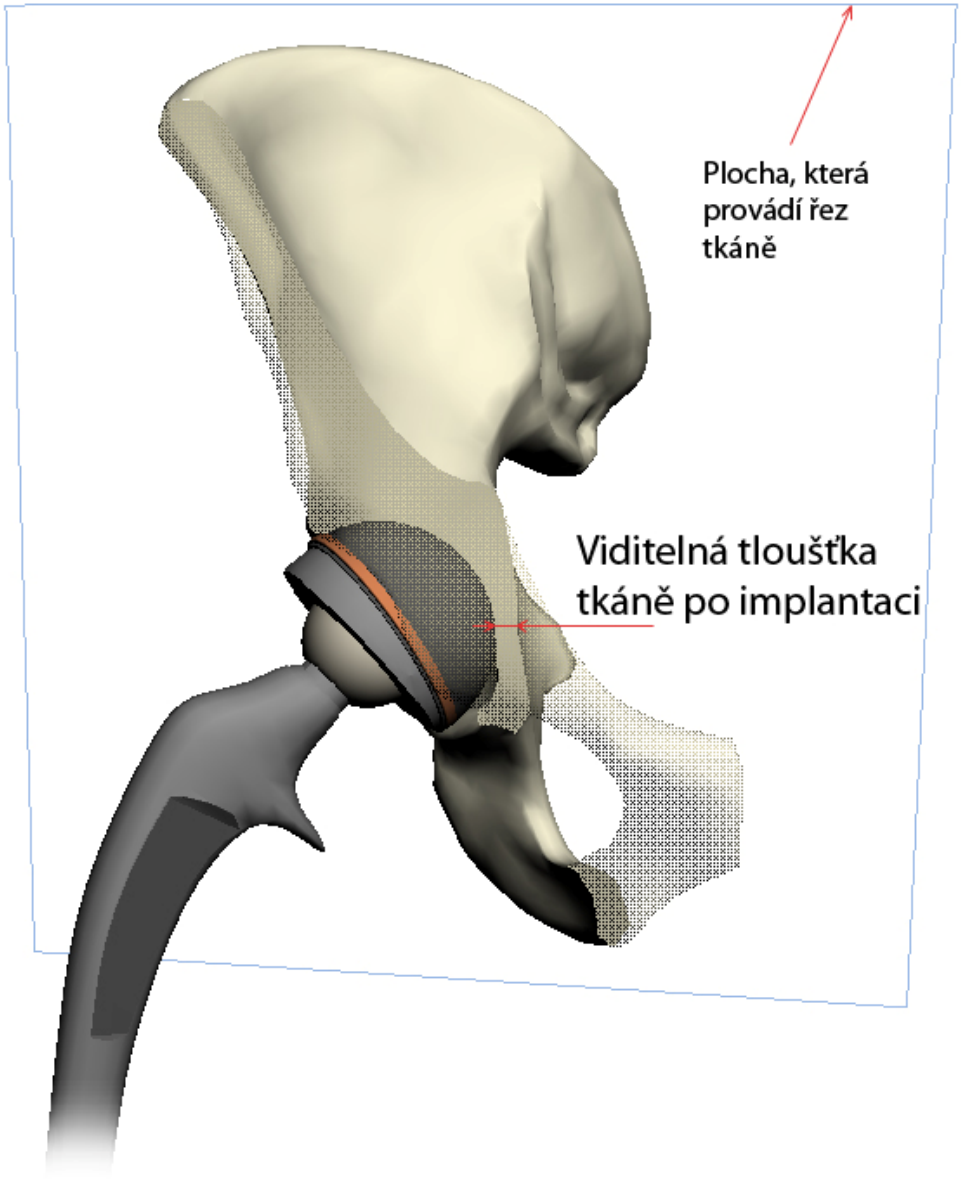
Příloha 1. Návrh na budoucí rozšíření aplikace:

- Profilový řez tkáně a zprůhlednění.

Příloha 2. CD/DVD obsahuje

- bakalářskou práci ve formátu pdf s licencí a zadáním,
- programovou dokumentaci ve formátu HTML, LaTex,
- vytvořené obrázky provázející publikaci ve formátu JPG,
- zdrojové texty programu včetně potřebných knihoven pro kompilaci,
- zkompilovanou verzi výsledného programu pro operační systém Microsoft Windows.

Příloha 1



Ukázka možnosti rozšíření aplikace. Návrh byl vytvořen v aplikaci 3ds max.