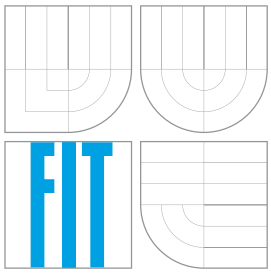
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# INTERAKTIVNÍ NÁSTROJ PRO ÚPRAVU 3D POLYGONÁLNÍCH MODELŮ

INTERACTIVE TOOL FOR 3D POLYGONAL MODELS MODIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR VONDRÁŠEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PŘEMYSL KRŠEK, Ph.D.
SUPERVISOR

BRNO 2007

# Zadání bakalářské práce

## Interaktivní nástroj pro úpravu 3D polygonálních modelů

**Interactive Tool for 3D Polygonal Models Modification**

**Vedoucí:**
Kršek Přemysl, Ing., Ph.D., UPGM FIT VUT
**Oponent:**
Herout Adam, Ing., Ph.D., UPGM FIT VUT
**Přihlášen:**
Vondrášek Petr
**Zadání:**

1. Seznamte se s problematikou 3D polygonálních modelů
2. Analyzujte problematiku interaktivní modifikace 3D polygonálních modelů
3. Navrhněte systém pro interaktivní modikaci 3D polygonálních modelů
4. Implementujte navrčžený systém v jazyce C/C++.
5. Zhodnoťte dosažené výsledky a stanovte další vývoj projektu.

**Kategorie:**
Počítačová grafika
**Implementační jazyk:**
C/C++
**Operační systém:**
MS Windows, Linux, Unix
**Literatura:**

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

# Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

# Abstrakt

Práce se zabývá problematikou úpravy 3D polygonálních modelů pomocí konkrétní metody, která spočívá v lokální deformaci povrchu modelu ve směru normály.

# Klíčová slova

3D, Polygonální model, 2.5D koncept, úprava polygonalních modelů, deformace ve směru normály, OpenSceneGraph

# Abstract

This work aims at 3D polygonal models editation in alternative way by local surface deformation in direction of the surface normal.

# Keywords

3D, Polygonal model modification, 2.5D concept, deformation in normal direction, OpenSceneGraph

# Citace

Petr Vondrášek: Interaktivní nástroj pro úpravu 3D polygonálních modelů, bakalářská práce, Brno, FIT VUT v Brně, 2007

# Interaktivní nástroj pro úpravu 3D polygonálních modelů

## 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.

.....
Petr Vondrášek
15. května 2007

## Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Přemyslu Krškovi, Ph.D. za konzultace k projektu a odbornou pomoc při řešení, dále za poskytnutí knihovny VectorEntity a úvodního tutoriálu k OpenSceneGraphu.

# Obsah

# Kapitola 1

# Úvod

Tato práce se zabývá řešením úpravy 3D polygonálních modelů konkrétním přístupem, který spočívá v lokální deformaci povrchu modelu ve směru normály. Jedná se o experimentální projekt, jehož cílem bylo vytvořit nástroj, který by uživateli bez jakýchkoliv zkušeností s touto problematikou umožňoval intuitivně upravovat již existující 3D polygonální modely. Úvodní inspirace pro tento projekt vychází z principu, který používá například program ZBrush, který je na poli profesionálních nástrojů pro práci s 3D polygonálními modely hlavním reprezentantem zmíněného přístupu.

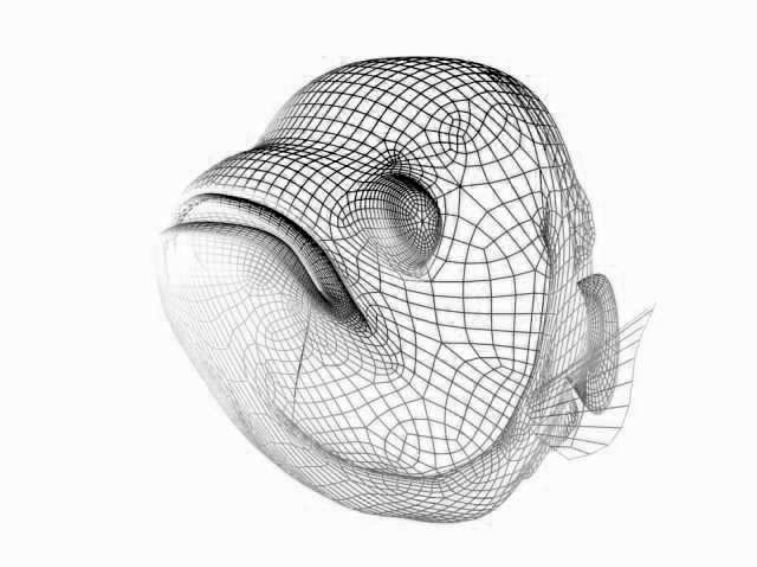
Projekt byl rozdělen do dvou fází. V první fázi bylo třeba seznámit se s problematikou 3D polygonálních modelů obecně a analyzovat tuto problematiku. Obojí bylo provedeno v rámci semestrálního projektu a je podrobně rozebráno v kapitole 2, která zahrnuje teoretickou část práce a následně v kapitole 3, která se zabývá analýzou konkrétního problému. V závěru první fáze byl vytvořen objektový návrh aplikace, který je představen v kapitole 4.

Druhá fáze navazuje na semestrální projekt a představuje vlastní implementaci programu, kterou detailně popisuje kapitola 5. Kapitola 6 popisuje ovládání aplikace a kapitola 7 je stručným souhrnem použitých nástrojů. Závěr představuje kapitola 8, která obsahuje celkové zhodnocení projektu a stanovuje jeho další možný vývoj.

# Kapitola 2

# Teoretický rozbor

Tato kapitola zahrnuje teoretickou část práce. Podkapitola 2.2 objasňuje základní pojmy z řešené problematiky, které budou v dalším textu využívány. Podkapitola 2.3 se zabývá základními přístupy, které se používají v běžně používaných nástrojích pro práci s 3D polygonálními modely z hlediska uživatelského rozhraní a navazující podkapitola 2.4 následně na základě zvoleného přístupu formuluje konkrétní cíl projektu.



Obrázek 2.1: Polygonální model

## 2.1 Polygonální reprezentace

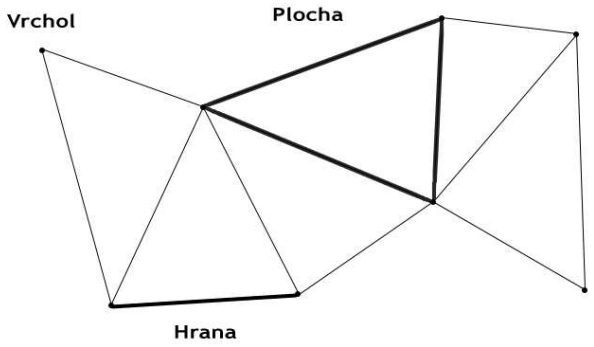
Polygonální reprezentace představuje jednu z nejstarších, nejjednodušších, ale zároveň i nejpopulárnějších metod reprezentace 3D modelu. Nachází velké uplatnění v aplikacích pro real-time 3D zobrazení, především díky relativně jednoduché hardwarové implementaci. Pro vyplňování trojúhelníků existují velmi rychlé algoritmy implementované v grafických procesorech. Díky tomu se jedná o v současnosti nejrychejší možnou reprezentaci. Mezi další výhody polygonální reprezentace patří, že umožňuje popis prakticky jakkoli komplexních

tvarů. Hlavní nevýhodou je nemožnost zobrazit zakřivené plochy. Tento problém je řešitelný pouze aproximací velkým množstvím polygonů, což má zpětně negativní dopad na rychlost vykreslování.

K alternativním metodám k polygonální reprezentaci se řadí například reprezentace pomocí NURBS, která se v dnešní době využívá především v oblasti CAD.

## 2.2 Základní teorie polygonálních modelů

Tato podkapitola popisuje základní geometrické objekty (dále primitiva), které utváří polygonální model. Pro podrobnější informace k tomuto odkazuji na materiály [2] a [3].




Obrázek 2.2: Základní primitiva

**Vrchol** je bod ve 3D prostoru a představuje nejzákladnější stavební komponentu polygonálního modelu. Počet vrcholů, které se mohou nacházet na stejném místě není omezen.

**Hrana** vznikne přímým propojením dvou vrcholů úsečkou a tvoří jednu stranu polygonu. Manipulací s hranou dochází i k manipulaci s vrcholy, které ji tvoří.

**Polygon (stěna)** vzniká vzájemným propojením minimálně tří hran. Výhodným typem polygonu je trojúhelník, který představuje nejjednodušší plochu, kterou lze vytvořit. Komplexnější polygony se vytváří z trojúhelníků nebo jako jeden objekt z více než tří vrcholů. Čtyřstranný polygon a trojúhelník jsou nejběžnější používaná primitiva. Jedna z hlavních výhod trojúhelníku spočívá v tom, že trojúhelník vždy představuje právě jednu rovinu, proto není problém spočítat normálu povrchu.

**Normála** je jednotkový vektor, který ukazuje ve směru kolmém k povrchu. Normálu lze získat vektorovým součinem vektorů kterýchkoliv dvou hran trojúhelníka. Pro rovný povrch je jeden směr normály pro všechny body, pro obecný tvar povrchu se může normálový vektor v každém bodě povrchu lišit. Využití normál je například v osvětlovacích modelech (phong shading model, gouraud shading model). Normála také určuje orientaci povrchu v prostoru. Na využití normál je založen základní princip fungování aplikace, který je popsán v závěru této kapitoly.

**3D Polygonální model** je aproximace povrchu 3D objektu pomocí polygonů. Podle [1] je to kolekce hran, vrcholů a polygonů, které jsou propojeny tak, že každý vrchol je sdílen nejméně dvěma hranami a každá hrana je sdílena nejvýše dvěma polygony. Hrana propojuje dva vrcholy a polygon je uzavřená sekvence hran.

## 2.3 Hlavní používané přístupy při úpravě 3D polygonálních modelů z pohledu uživatelského rozhraní

Úprava polygonálního modelu obecně spočívá v manipulaci s primitivy, ze kterých se tento model skládá. Přidáváním, odebíráním nebo změnou pozice jeho vrcholů případně hran nebo polygonů. Cílem této podkapitoly je charakterizovat problematiku úpravy 3D polygonálních modelů z pohledu uživatelského rozhraní, tedy způsobu interakce mezi uživatelem a programem tak, aby byl jasně zřejmý základní rozdíl mezi přístupem, který byl použit pro řešení a přístupem, který má hlavní využití ve většině dnešních nástrojů pro 3D modelování. Následující dva přístupy představují dva různé pohledy na jeden problém.

### 2.3.1 Klasický přístup

Tento přístup nazývám jako klasický pouze z důvodu, že je daleko více rozšířený. Manipulace s jednotlivými primitivy 3D polygonálního modelu probíhá v kterékoli z os x, y, z souřadného systému, přičemž výběr jak těchto os tak jednotlivých primitiv může uživatel volit zcela libovolně. Pracovní plocha může být k tomuto účelu rozdělena na několik pohledů na scénu, klasicky na pohled shora, z leva, z prava a perspektivní pohled.

Tento přístup je velice silnou zbraní v nástrojích jako je například 3D Studio MAX nebo Maya, které se hodně využívají například na poli vytváření modelů pro hry a obecně real-time nasazení, kde je třeba neustále brát ohled na celkový počet polygonů modelu. Nachází uplatnění všude tam, kde je potřeba mít maximální kontrolu nad modelem.

Podstata práce, kdy je s modelem často manipulováno přímo na úrovni jednotlivých vrcholů, hran, případně polygonů předurčuje skutečnost, že uživatel musí být s danou problematikou relativně dobře obeznámen a chce-li rychle dosáhnout rozumných výsledků, musí mít určité zkušenosti.

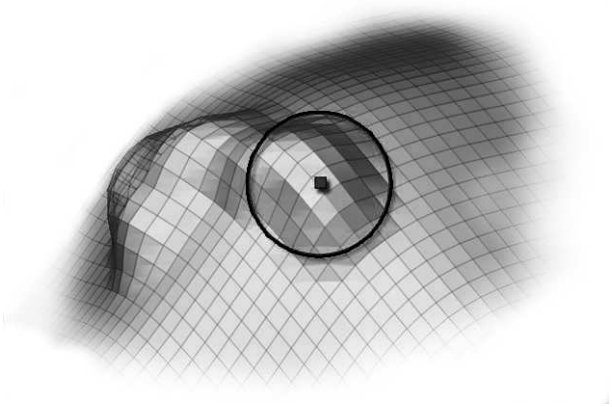
### 2.3.2 Alternativní přístup

Oproti předchozímu přístupu se jedná o značné zjednodušení, protože jakákoliv manipulace s primitivy se odehrává pouze ve směru, který udává normála k povrchu, nikoliv ve směru os souřadného systému. Uživatel určuje pouze směr ve smyslu kladné nebo záporné normály. Výběr primitiv bývá navíc zjednodušen pouze na možnost vybrat určitou oblast v rámci daného poloměru. Uživatel se tím pádem nemusí starat o to, co je nebo z čeho se skládá polygonální model, protože nemanipuluje s konkrétními vrcholy, hranami ani polygony jako tomu bylo v předchozím případě.

Tento přístup nabízí výrazné zjednodušení ovládání, díky čemuž je způsob práce daleko více intuitivní. Prakticky tak dochází k oddělení podstaty práce s 3D polygonálním modelem od práce s modelem jako takové pomocí další vrstvy, která se nachází mezi modelem a uživatelem, čímž se celý proces úpravy modelu stává pro uživatele snazším. Tento přístup nabízí ideální řešení pro lokální deformace, nevýhodou je, že v této základní podobě zdaleka neumožňuje takovou kontrolu nad modelem.

Tento přístup využívá právě program ZBrush zmíněný v úvodu, ale i jiné, nekomerční programy, jako například Blender. V souvislosti s tímto lze narazit na pojem koncept 2.5D, což je podle [5] spíše marketingový výraz, nicméně hlavní myšlenka spočívá v tom, že pracovní plocha programu představuje na první pohled klasické 2D plátno, známé například z běžných bitmapových editorů, se kterým se i podobně pracuje. Rozdíl je v tom, že výsledek práce se projevuje na 3D modelu.

Lépe tuto problematiku vysvětluje obrázek 2.3



Obrázek 2.3: Princip alterativní metody

Poloměr nástroje je nastavitelný a na obrázku je vyznačen kružnicí. Prostřední bod představuje střed operace. Funkce nástroje – způsob jakým je model deformován – mohou být různé. Intenzita provádění deformace se například může měnit se vzdáleností od středu. Deformace je prováděna v kladném nebo v záporném směru normály, čímž může docházet například k vytahování nebo zatlačování povrchu.

## 2.4 Použitý způsob řešení a formulace konkrétního cíle projektu

Základní rozdíl mezi dvěma výše zmíněnými přístupy spočívá v tom, že klasický přístup dává uživateli větší kontrolu nad modelem, což je vyváženo o něco složitějším způsobem prace. Alternativní přístup naopak poskytuje výhodu relativně jednoduchého ovládání která je však vyvážena menší flexibilitou.

Cílem projektu bylo vytvořit nástroj s intuitivním ovládáním, jinak řečeno minimálními nároky na znalosti a předchozí zkušenosti uživatele s problematikou 3D polygonálních modelů. Realizace založená na klasickém přístupu by patrně neposkytovala ideální řešení v tomto ohledu. Lepší řešení daného problému tak nabízí koncept 2.5D.

Způsob práce s nástrojem vypadá tak, že uživatel volně přejíždí kurzorem myši v okně scény, která obsahuje načtený polygonální model a ve zvoleném místě na modelu provádí úpravu jeho povrchu v kladném nebo záporném směru normály. Uživatel si při tom může vybrat z několika různých nástrojů s nastavitelným poloměrem, z nichž každý provádí určitý typ deformace. Může se jednat například o posouvání vybrané oblasti určitým způsobem, nebo vyhlazování, případně nějaký druh šumu atd.

Následující analýza problému a z ní vyplývající řešení vycházejí z tohoto konceptu.

# Kapitola 3

# Analýza problému

Problém lze rozložit do několika dílčích podproblémů. Prvním z nich je zajištění reprezentace dat 3D modelu. Touto problematikou se zabývá podkapitola 3.1. Dalším je zajištění vykreslování modelu, čemuž se věnuje podkapitola 3.2. Třetí podproblém představuje řešení úpravy modelu a je podrobně rozebrán v podkapitole 3.3.

## 3.1 Reprezentace dat 3D polygonálního modelu

Data polygonálního modelu mohou být reprezentována několika různými způsoby z nichž každý má své výhody a nevýhody. V jedné aplikaci může být využito více reprezentací zároveň. Vychází se při tom z toho, jaké operace mají být s uloženými daty prováděny. Běžnými operacemi vykonávanými nad daty polygonálního modelu jsou například:

Nalezení všech hran, které jsou napojeny na daný vrchol.

Nalezení všech polygonů, které sdílejí jednu hranu nebo vrchol.

Nalezení vrcholů, které tvoří danou hranu.

Aby bylo možné takové operace efektivné provádět, je třeba zvolit takový způsob uložení dat, který to umožní. Platí, že čím přesněji jsou vztahy mezi polygony, hranami a vrcholy reprezentovány v paměti (tedy čím více paměti využijeme), tím rychlejší je následně provádění těchto operací a obráceně.

Podle [2] datová struktura popisující polygonální síť bývá rozdělena do dvou logických částí a to na část geometrickou a topologickou. Geometrická část zaznamenává souřadnice vrcholů trojúhelníků, topologická část udržuje informace o tom, které vrcholy tvoří které trojúhelníky, případně o tom, které trojúhelníky spolu sousedí atd.

Pro řešení problému vnitřní reprezentace dat polygonálního modelu byla využita knihovna Vector Entity, kterou poskytl Ing. Přemysl Kršek, Ph.D. a která je součástí toolkitu MDSTk, (Medical Data Segmentation Toolkit) vyvíjeného na FIT VUT v Brně Ing. Michalem Španělem.
Následuje přehled základních řešení podle [1], některé informace jsem čerpal též z [2].

### 3.1.1 Explicitní reprezentace

Explicitní reprezentace znamená, že každý polygon je popsán seznamem souřadnic vrcholů, které ho tvoří.

$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n))$$

Jedná se o nejjednodušší řešení, které je efektivní pro uložení jednoho polygonu, ne však pro uložení celého modelu. Zásadní nevýhoda tohoto řešení spočívá ve zbytečném opakování definic již definovaných vrcholů, které jsou společné pro sousední polygony. Jak uvádí [3], dobrým příkladem je obyčejná krychle, která má šest stěn a osm sdílených vrcholů. Pokud bude tento objekt reprezentován explicitní metodou, každý vrchol bude specifikován třikrát: pro každou stěnu, která jej používá jednou. Muselo by se tedy definovat 24 vrcholů, i když by teoreticky stačilo pouze osm. Navíc i přilehlé hrany k těmto vrcholům by byly použity 2x.

Dalším problémem je, že kdybychom chtěli například interaktivně posunout vybraný vrchol tak, aby se s ním posunuly i přilehlé hrany, museli bychom nejprve najít všechny polygony, které sdílejí daný vrchol, což by obnášelo porovnat všechny vrcholy jednoho polygonu se všemi vrcholy všech ostatních polygonů. Taková operace by byla krajně neefektivní.

### 3.1.2 Ukazatele na seznam vrcholů

Tato metoda umožňuje mít každý vrchol uložen pouze jednou v seznamu vrcholů.

$$V = ((x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n))$$

Princip spočívá v ukládání seznamu ukazatelů na seznam vrcholů z nichž každý je uložen pouze jednou. Každý vrchol má unikátní index, který jej identifikuje v rámci tohoto seznamu. Každý polygon je pak definován jako seznam ukazatelů do seznamu vrcholů.

Výhoda této metody je v tom, že každý vrchol je uložen pouze jednou. Není tak problém jeho polohu kdykoliv změnit. Na druhou stranu stále bude problém nalézt jednoduše polygony, které sdílejí stejnou hranu, navíc tyto hrany budou pak nalezeny 2x. Tento problém lze řešit rozšířením o pole ukazatelů na seznam hran.

### 3.1.3 Ukazatele na seznam hran

V této metodě máme opět seznam vrcholů, ale polygon reprezentujeme nikoliv jako seznam ukazatelů do seznamu vrcholů, ale jako seznam ukazatelů do seznamu hran ve kterém je každá hrana pouze jednou. Každá hrana v seznamu hran pak dále ukazuje do seznamu vrcholů na vrcholy, kterými je tvořena a také na jeden nebo dva polygony ke kterým hrana náleží. Tedy:

$$P = (E_1, ..., E_n)$$
$$E = (V_1, V_2, P_1, P_2)$$

## 3.2 Vykreslování 3D polygonálního modelu

Pro vykreslování modelu byl využit toolkit OpenSceneGraph. Tato podkapitola představuje stručné seznámení s tímto toolkitem a jeho základními vlastnostmi. Následující definice pochází z [4].

### 3.2.1 OpenSceneGraph

OpenSceneGraph (OSG) je multiplatformní OpenSource grafický toolkit určený pro vývoj výkonných grafických aplikací. Je založen na konceptu grafu scény a poskytuje objektově orientovaný framework nad OpenGL, který osvobozuje programátora od nutnosti implementovat a optimalizovat nízkoúrovňová volání funkcí OpenGL. OSG je napsán kompletně v C++ a OpenGL.

Graf scény je kolekce uzlů uspořádaných ve stromové struktuře. Listy stromu reprezentují fyzické objekty scény, jejich geometrii a jejich vlastnosti jako materiál atd. Scéna je složena z různých typů uzlů. Základní typ uzlu v OSG nese informace o geometrii těles, další typy pak nesou informace o atributech jako je například barva nebo textury. Dále existuje speciální typ uzlu, který umožňuje v sobě uchovat další typy uzlů, což umožňuje organizovat ostatní uzly do hierarchických struktur zvaných grafy. Takovýto graf pak reprezentuje celou scénu. Matematicky se jedná o acyklicky orientovaný graf.

OSG je jen jedno z řešení na bázi grafu scény. Podobných toolkitů dnes existuje celá řada. Zdaleka ne všechny jsou však OpenSource.

## 3.3 Řešení úpravy 3D polygonálního modelu

Jak bylo stanoveno v závěru kapitoly 2, úprava 3D polygonálního modelu probíhá v rámci řešení tohoto projektu formou lokální deformace ve směru normály k povrchu. Tato podkapitola analyzuje tento problém.

**Úprava modelu představuje 3 základní kroky:**

1. Výběr bodu na povrchu modelu – výchozí bod deformace, který zároveň představuje střed nástroje.
2. Výběr vrcholů v oblasti definované aktuálně nastaveným poloměrem nástroje.
3. Řešení funkce, která manipuluje s vybranými vrcholy – deformace.

V případě, že se kurzor myši v okně pracovní plochy nachází nad modelem, výchozí bod deformace je na modelu vždy přímo pod kurzorem. Výběr oblasti funguje na principu průchodu jednotlivými vrcholy směrem od zvoleného středu a porovnávání jejich vzdáleností od tohoto středu s aktuálním poloměrem nástroje. Deformace spočívá v dosazení hodnot vzdálenosti jednotlivých vrcholů z vybrané oblasti do funkčního předpisu některé z celé řady použitelných funkcí. Funkční hodnoty pak udávají míru posunu konkrétního vrcholu po normále. Zde se nabízí velký prostor pro experimentování s nejrůznějšími funkcemi. Následuje analýza řešení základních druhů úprav, které by aplikace neměla postrádat.

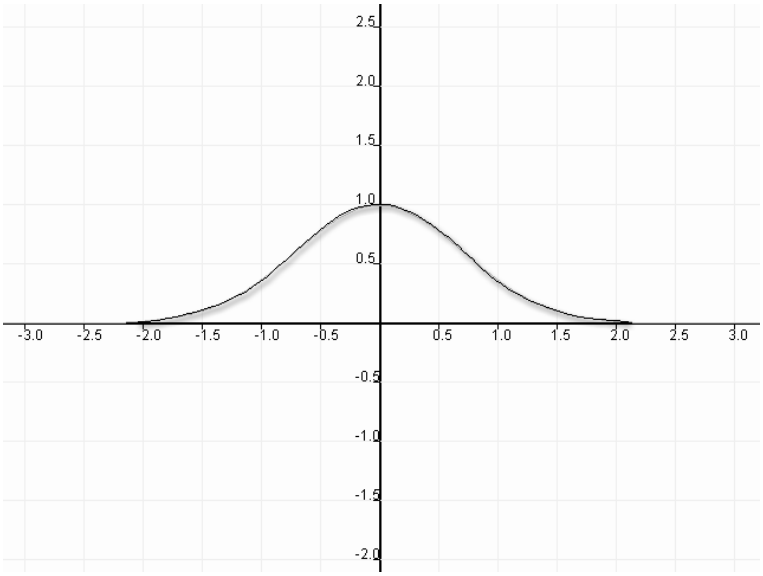
### 3.3.1 Deformace pomocí Gaussovy funkce normálního rozdělení

Základní funkcí zajišťující deformaci modelu je Gaussova funkce normálního rozdělení. Využitím této funkce dochází k realizaci chování, které umožňuje plynulé vytahování vybrané oblasti tak, že nejvíce jsou deformací ovlivněny vrcholy, které se nacházejí nejblíže středu nástroje, nejméně naopak vrcholy které se nacházejí blízko hranice poloměru. Při řešení tohoto problému jsem vycházel z [6].

Graf funkce je na obrázku 3.1

$$f(x) = ae^{-(x-b)^2/c^2}$$

kde $a > 0$, $b \in \mathbb{R}$, $c \in \mathbb{R}$




Obrázek 3.1: Gaussova funkce normálního rozložení

Funkce má tři parametry $a$, $b$, $c$. Parametr $b$ ovlivňuje posun grafu po ose $x$, což v rámci aplikace nemá žádné uplatnění, proto není třeba se jím dále zabývat. Jeho hodnota bude vždy 0. Parametr $a$ ovlivňuje globální maximum funkce neboli funkční hodnotu v bodě 0 (je-li parametr $b$ roven 0). Parametr $c$ ovlivňuje roztažení grafu do šířky. S parametry $a$ a $c$ uživatel může manipulovat a tím přesněji definovat chování tohoto druhu deformace.

### 3.3.2 Vyhlazování povrchu pomocí algoritmu Laplacian smooth

Další ze základních operací, které by aplikace neměla postrádat je lokální vyhlazování povrchu.

Pro řešení tohoto problému byl zvolen algoritmus Laplacian smooth, který je jedním z nejjednodušších. Jeho základní princip spočívá v nastavení nové pozice každého vrcholu ve vybrané oblasti na pozici danou průměrem pozic vrcholů se kterými tento sousedí.

$$p' = p + \frac{\lambda}{n} \sum_{i=0}^{n=1} q_i - p$$

Nová pozice vrcholu $p'$ vychází z původní pozice $p$ a pozic sousedních vrcholů $q$. $\lambda$ představuje faktor váhy, která reguluje vliv přímých sousedních vrcholů na novou pozici vrcholu v každé iteraci.

### 3.3.3 Další funkce

Použitelné jsou například i základní goniometrické funkce jako sin nebo cos, ale najde se celá řada dalších. Pro účely testování funkce vyhlazování bylo třeba vytvořit funkci, která by fungovala opačně a povrch naopak zvrásňovala. K tomu jsem použil funkci $sinx$. Jak bylo zmíněno výše, tato oblast nabízí široký prostor pro další experimentování.

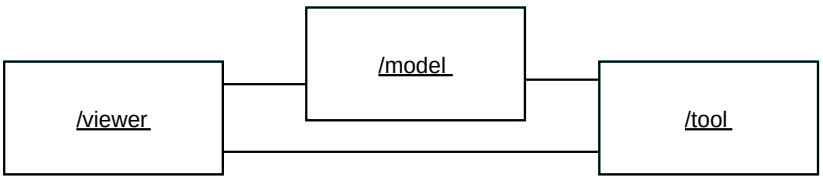
# Kapitola 4

# Návrh řešení

Podkapitola 4.1 představuje základní koncept, ze kterého vychází objektový návrh, který je představen v podkapitole 4.2.

## 4.1 Způsob řešení – základní koncept

Jak vyplývá z analýzy problému, základní koncept vychází z použití třech hlavních částí a představuje jej obrázek 4.1.

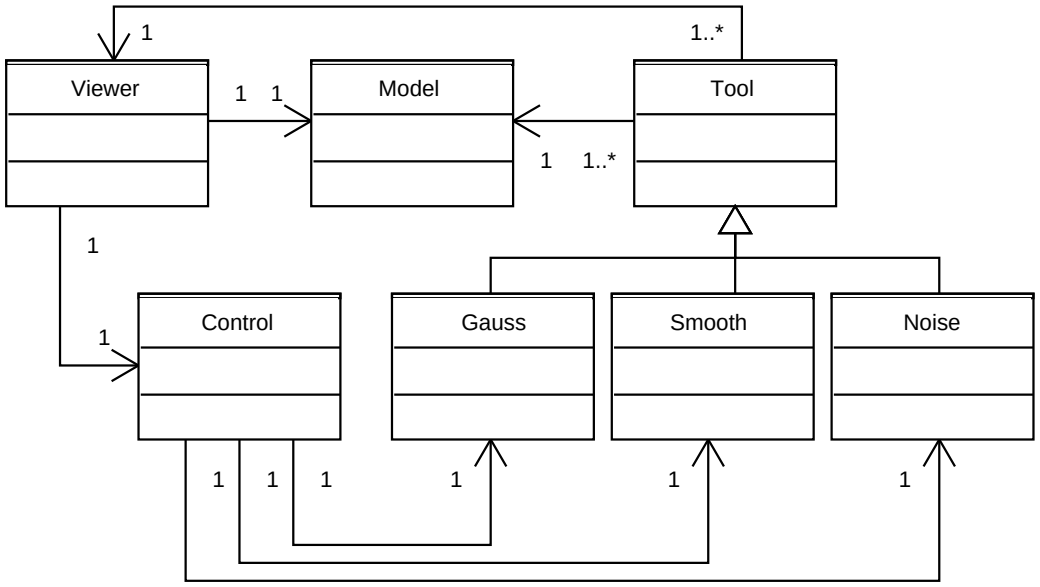



Obrázek 4.1: Základní koncept

Jedná se o digram spolupráce, na které se podílejí tři hlavní objekty, z nichž každý řeší konkrétní podproblém tak, jak bylo popsáno v kapitole 3. Model zajišťuje vnitřní reprezentaci dat, viewer tvoří frontend aplikace a stará se o vykreslování. Tool zajišťuje úpravy modelu.

## 4.2 Vlastní návrh aplikace – diagram tříd

Následující diagram tříd staví na předchozím konceptu.




Obrázek 4.2: Diagram tříd

Aplikace je navržena tak, aby nebyl problem jí rozšířit o další funkce zajišťující deformaci. Každá taková funkce představuje objekt, který dědí základní vlastnosti z třídy Tool. Jádrem systému je třída Control, která zajišťuje správu těchto objektů. Pro doimplementování dalších nástrojů stačí vytvořit novou třídu, která zdědí základní vlastnosti ze třídy Tool a následně implementovat její metodu Apply(), která zajišťuje konkrétní deformaci. Využívá se zde principu polymorfismu.
Detaily řešení popisuje následující kapitola.

# Kapitola 5

# Implementace

Tato kapitola popisuje praktickou část práce – implementaci programu v jazyce C++, která představovala druhou fázi projektu. V podkapitole 5.1 se nachází popis jednotlivých tříd implementovaných v rámci tohoto projektu, podkapitola 5.2 objasňuje nejpodstatnější části implementace.

Celý proces implementace měl iterativní charakter. Nejprve byl řešen daný podproblém především s ohledem na co nejrychlejší dosažení požadované funkčnosti. Hledání a implementace optimálního řešení následovala v následující iteraci. Během práce bylo třeba se neustále seznamovat s možnostmi použitého toolkitu, díky čemuž byly postupně objevovány způsoby, jak některé již vyřešené problémy vyřešit efektivněji.

## 5.1 Popis implementovaných tříd

Součástí odevzdaného balíku je systémem Doxygen vygenerovaná dokumentace ve formátu .chm, která poskytuje detailní přehled implementovaných tříd, jejich vlastností a metod.

**třída Viewer:** Je součástí OSG a jako jediná ze zde zmíněných nebyla implementována v rámci tohoto projektu. Pro celkovou srozumitelnost a ucelenost řešení však bylo třeba ji zde uvést, protože právě možnost využití této třídy byla jedním z hlavních důvodů použití OSG. Úkolem objektu, který je instancí této třídy, je řešení prezentační vrstvy aplikace. Jedná se o vytvoření hlavního okna a zajištění vykreslování objektů scény. Tato třída zároveň poskytuje základní uživatelské rozhraní.

**třída Control:** Představuje jádro systému, které zajišťuje správu objektů typu Tool, které implementují konkrétní nástroje. K tomu je použit stl kontejner typu vector obsahující ukazatele na objekty typu Tool. Naplnění kontejneru se provádí v konstruktoru tohoto objektu. Další hlavní funkcí této třídy je implementace metod reagujících na uživatelovy požadavky.

**třída Model:** Reprezentuje 3D polygonální model se kterým uživatel pracuje. Obsahuje metody pro načítání zvoleného modelu pro jeho správné zobrazení pomocí OSG, které zajišťuje instance zmíněné třídy Viewer.

**třída Tool:** Z této třídy dále dědí konkrétní nástroje reprezentované například třídami Gauss nebo Smooth případě dalšími, které mohou být přidány v budoucnu. Třída Tool implementuje základní vlastnosti a metody, které jsou všem nástrojům společné.

**třída Gauss : Tool:** Implementuje základní funkci programu, která do předpisu Gaussovy funkce normálního rozdělení dosazuje hodnoty vzdáleností vrcholů nacházejících se ve vybrané oblasti dané poloměrem zvoleného nástroje od středu. Funkční hodnota udává míru posunutí daného vrcholu ve směru určeném kladnou nebo zápornou normálou.

**třída Smooth : Tool:** Implementuje nástroj využívající vyhlazovací algoritmus Laplacian smooth.

**třída Noise : Tool:** Implementuje nástroj využívající k deformaci funkci $\sin x$.

## 5.2 Průvodce programem

### 5.2.1 Inicializace programu

Po spuštění aplikace nejprve dojde k vytvoření kořenového uzlu grafu scény. Následně je vytvořen objekt, který je instancí třídy Model a je zavolána jeho metoda, která zajistí načtení souboru s modelem ve formátu stl do VectorEntity. Cesta k tomuto souboru se zadává do konzole jako první argument programu. Po té je zavolána metoda objektu Model, která zajistí vytvoření geometrie modelu pro OpenSceneGraph, která se následně přiřadí kořenovému uzlu. Dalším krokem je vytvoření objektu viewer a nastavení jeho vlastností včetně zaregistrování speciální metody, která zpracovává reakce na události vyvolané uživatelem. Vieweru je předán kořenový uzel grafu a úvodní inicializace programu je ukončena vstupem do hlavní vykreslovací smyčky.

### 5.2.2 Editační mód

Základem je metoda, která zjišťuje, který bod na povrchu modelu se nachází pod aktuální pozicí kurzoru myši (picking). K tomuto účelu je využíváno funkce zmíněné třídy viewer. Základní princip spočívá ve využití přímky, která je kolmá na rovinu projekce a vychází z bodu aktuální pozice kurzoru směrem do scény. Tato přímka protíná model v určitých bodech, které OSG ukládá do seznamu (tzv. hitlist). První bod v tomto seznamu který je vždy nejblíže k pozorovateli je hledaný bod, ze kterého se dále vychází. Tato operace se v editačním módu provádí v každém snímku. Následně je vybrán trojúhelník modelu, který obsahuje tento bod. Z tohoto trojúhelníku je dále vybrán kterýkoli jeho vrchol a následně přichází na řadu rekurzivní algoritmus, který na základě znalosti výchozího bodu pro operaci, poloměru nástroje a tohoto vrcholu zajišťuje selekci všech vrcholů v oblasti. Ukazatele na vybrané vrcholy se ukládají do stl kontejneru typu vector kterýnásledně slouží jako zdroj dat pro konkrétní metodu Apply() vybraného nástroje která provádí danou deformaci.

# Kapitola 6

# Ovládání programu

Program má dva režimy funkce: režim prohlížení a režim úprav. Mezi těmito režimy se přepíná klávesou SPACE. Výchozí režim prohlížení je aktivní po spuštění a umožňuje prohlížení modelu, jeho posouvání, otáčení, zoom pomocí tlačítek myši. Slouží ke zvolení oblasti modelu, která má být deformována. Přepnutí do režimu úprav aktivuje naposledy vybraný nástroj. V případě prvního spuštění je to nástroj využívající Gaussovu funkci. Aktivování nástroje znamená, že se automaticky provádí výběr oblasti dané poloměrem nástroje, což je vyznačeno změnou barvy vybrané oblasti na barvu konkrétního nástroje.

Červená barva vybrané oblasti značí použití nástroje, který využívá Gaussovu funkci. V případě použití tohoto nástroje lze navíc ovlivňovat vlastnosti parametrů ovlivňujících tvar funkce pomocí kláves...

Zelená barva vybrané oblasti představuje nástroj, který vyhlazuje základním algoritmem Laplacian smooth. Žlutá barva pak představuje experimentální nástroj - opak vyhlazování.

Poloměr nástroje lze zvětšovat pomocí klávesy A a zmenšovat pomocí kláves Z a Y.

Následně lze provádět úpravy modelu pomocí levého tlačítka myši. Klávesa SHIFT přepíná mezi kladným a záporným směrem úprav. Klávesou alt se cyklicky přepíná mezi jednotlivými nástroji.

Program pracuje s modely ve formátu stl. Cestu k souboru je třeba zadat jako první argument při spuštění z konzole.

# Kapitola 7

# Použité nástroje

Vývoj probíhal na operačním systému MS Windows XP v jazyce C++ a v prostředí Visual Studio 2005. Pro správu verzí byl využíván nástroj Subversion. Jako Subversion klient ve Windows byl používán TortoiseSVN. Proces psaní kódu byl průběžně dokumentován systémem Doxygen. Jako toolkit pro vykreslování byl použit zmíněný OpenSceneGraph. Pro reprezentaci dat polygonálního modelu byla použita poskytnutá knihovna Vector Entity.

# Kapitola 8

# Výsledky a závěr

Počáteční ambicí tohoto projektu zdaleka nebylo překonat již existující nástroje, ale vyzkoušet implementaci řešení založeného na popsaném přístupu, což se dle mého názoru podařilo úspěšně splnit. Toolkit OpenSceneGraph se k tomuto účelu jeví jako použitelný. Aplikace umožňuje základní úpravy 3D polygonálních modelů, nicméně pro její případné širší nasazení je nezbytné doimplementovat další funkce. Zatím se jedná spíše o základní jádro systému představující řešení první fáze, po které by měly následovat další.

Tento projekt sám o sobě na žádný předchozí nenavazuje, v budoucnu by ale měly vzniknout projekty, které by navazovaly na tento. Je potřeba implementovat grafické uživatelské rozhraní, což je jedna z věcí, která již nyní začíná být omezující při testování funkcí především z důvodů nepřehlednosti spojené s nedostatkem použitelných kláves a s nutností si je pamatovat. Dalším rozšířením by měla být možnost úpravy topologie modelu například přidáním nebo odebráním vrcholů v situacích, které by toto mohly vyžadovat. Tento problém aplikace v této fázi zatím neřeší. Do budoucna se dále plánuje podpora využití funkcí tabletu což by mělo výrazně zvýšit efektivitu práce s programem.

Za hlavní přínos tohoto projektu pro mojí osobu považuji vedle získání řady nových zkušeností s programováním především seznámení se s OSG a s technologií grafu scény, v čemž spatřuji velké možnosti využití. V budoucnu se hodlám této problematice dále věnovat.

# Literatura


[1] Feiner S. K. Foley J. D., van Dam A. *Computer Gaphics: Principles and Practice*. USA.

[2] Felkel P. Žára J., Beneš B. *Moderní počítačová grafika*. Computer press, Praha, 1998. ISBN 80-7226-049-9.

[3] Neider J. Shreiner D., Woo M. *OpenGL Průvodce programátora*. Computer Press, Brno, 2006. ISBN 80-251-1275-6.

[4] Webové stránky. Osg. `http://www.openscenegraph.org`.

[5] Webové stránky. Wikipedia – 2.5d. `http://en.wikipedia.org/wiki/2.5D`.

[6] Webové stránky. Wikipedia – gaussian function. `http://en.wikipedia.org/wiki/Gaussian_function`.