

# Masarykova univerzita Fakulta informatiky



## Modelování pomocí lokálních deformací

Diplomová práce

Podzim 2005

Václav Samec

## **Prohlášení**

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

---

Rád bych poděkoval svému vedoucímu, doc. Ing. Jiřímu Sochorovi, CSc., za pomoc a vedení při tvorbě této diplomové práce.

## **Shrnutí**

Cílem práce bylo prostudovat metody modelování těles a povrchů pomocí deformací prostorů vymezených bodovými svazy. Vybrané metody popsat a implementovat jako součást experimentální aplikace.

## **Klíčová slova**

Volné deformace, Rozšířené volné deformace, FFD, EFFD, mřížka, kontrolní bod, lokální deformace

# OBSAH

|  |           |
|--|-----------|
| <b>OBSAH</b> .....                             | <b>4</b>  |
| <b>1 ÚVOD</b> .....                            | <b>5</b>  |
| <b>2 MODELOVÁNÍ POMOCÍ DEFORMACÍ</b> .....     | <b>6</b>  |
| <b>3 GLOBÁLNÍ DEFORMACE</b> .....              | <b>8</b>  |
| <b>4 LOKÁLNÍ DEFORMACE</b> .....               | <b>10</b> |
| 4.1 VOLNÉ DEFORMACE .....                      | 10        |
| 4.1.1 <i>Lokální souřadný systém</i> .....     | 11        |
| 4.1.2 <i>Prostorová mřížka</i> .....           | 12        |
| 4.1.3 <i>Deformace FFD</i> .....               | 13        |
| 4.1.4 <i>Algoritmus FFD</i> .....              | 15        |
| 4.1.5 <i>Omezení</i> .....                     | 15        |
| 4.1.6 <i>Shrnutí</i> .....                     | 15        |
| 4.1.7 <i>Implementace FFD</i> .....            | 16        |
| 4.2 ROZŠÍŘENÉ VOLNÉ DEFORMACE .....            | 17        |
| 4.2.1 <i>Prostorová mřížka EFFD</i> .....      | 17        |
| 4.2.2 <i>Výpočet lokálních souřadnic</i> ..... | 21        |
| 4.2.3 <i>Problematické případy</i> .....       | 26        |
| 4.2.4 <i>Deformace EFFD</i> .....              | 27        |
| 4.2.5 <i>Algoritmus EFFD</i> .....             | 27        |
| 4.2.6 <i>Shrnutí</i> .....                     | 28        |
| 4.2.7 <i>Implementace EFFD</i> .....           | 29        |
| <b>5 EXPERIMENTÁLNÍ APLIKACE</b> .....         | <b>31</b> |
| 5.1 ROZHRAŇÍ APLIKACE .....                    | 31        |
| 5.2 PŘEHLED NÁSTROJŮ .....                     | 32        |
| 5.3 DEFORMACE.....                             | 33        |
| <b>6 ZÁVĚR</b> .....                           | <b>34</b> |
| <b>7 GALERIE</b> .....                         | <b>35</b> |
| <b>8 SEZNAM POUŽITÉ LITERATURY</b> .....       | <b>40</b> |

# 1 Úvod

Geometrické modelování těles bylo vždy jednou z hlavních oblastí výzkumu v počítačové grafice. Modelování zahrnuje vytváření geometrické reprezentace tělesa a vývoj nástrojů pro jeho úpravu. Moderní modelovací systémy obsahují řadu nástrojů pro tvorbu a úpravu geometrických těles, které nám umožňují vytvářet modely ze všech oblastí lidské činnosti či lidské fantazie. Většinou nabízejí modelovací systémy nástroje pro úpravy tělesa podle typu jeho geometrické reprezentace, což může být efektivní řešení pro konkrétní model. Snahou moderních systémů je zobecnit tento přístup, tedy oddělit typ geometrické reprezentace tělesa od nástrojů pro jeho úpravu. Tím se stane typ geometrické reprezentace transparentním pro uživatele a usnadní se tak jeho práce.

Tato práce popisuje deformační techniky užívané v moderní počítačové grafice pro vytváření a úpravu všech geometrických těles. Detailně představí dvě techniky a návod pro jejich implementaci.

Práce se skládá z těchto kapitol:

- *Modelování pomocí deformací* stručně popisuje problematiku modelování těles a uplatnění deformačních technik.
- *Globální deformace* vysvětluje princip deformačních metod aplikovaných na celé těleso a problémy s tím spojené.
- *Lokální deformace* se zabývá v podrobném měřítku dvěma deformačními technikami.
- *Experimentální aplikace* popisuje vlastní implementaci popsaných metod pro modelování těles.

Budeme-li v textu hovořit o bodech modelu, na které se aplikují transformace, máme na mysli klíčové body, kterými je model určen. Je-li například model reprezentován trojúhelníkovou sítí, za klíčové body považujeme pozice krajních bodů trojúhelníků.

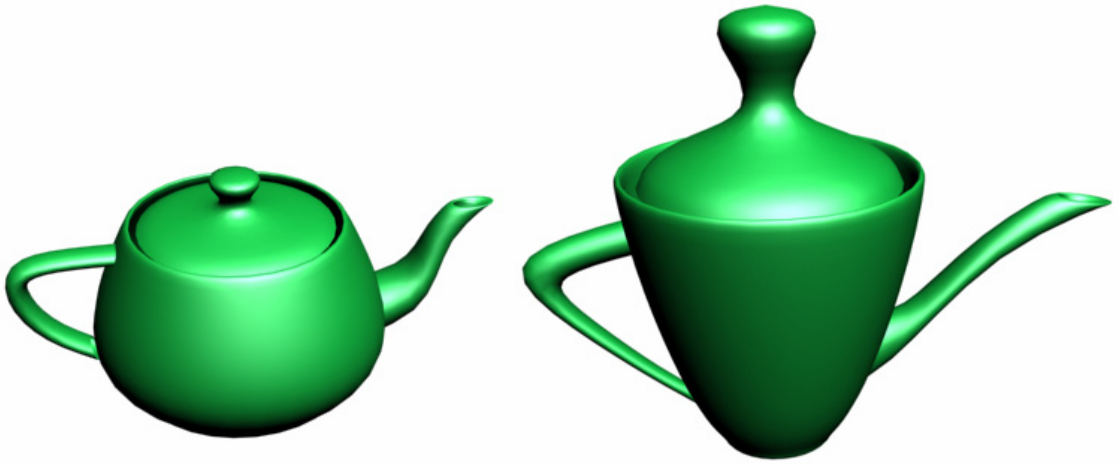
## 2 Modelování pomocí deformací

Modelovací techniky vycházejí z několika přístupů. Závisí většinou na geometrické reprezentaci tělesa. Pokud je těleso definováno seznamem polygonů, modelování spočívá v úpravách jednotlivých plošek. Na ně lze aplikovat řadu operací, jako vytažení, rozdělení, zjemňování, apod. Těleso může být také zadáno jako parametrický povrch, což je výhodné především pro přesnou specifikaci modelu. Těleso je pak sestaveno z částí, kterým říkáme pláty. Ty jsou zpravidla definovány množinou kontrolních bodů a jejich manipulací v prostoru upravujeme povrch. Před vykreslováním se však parametrický povrch aproximuje pomocí polygonů.

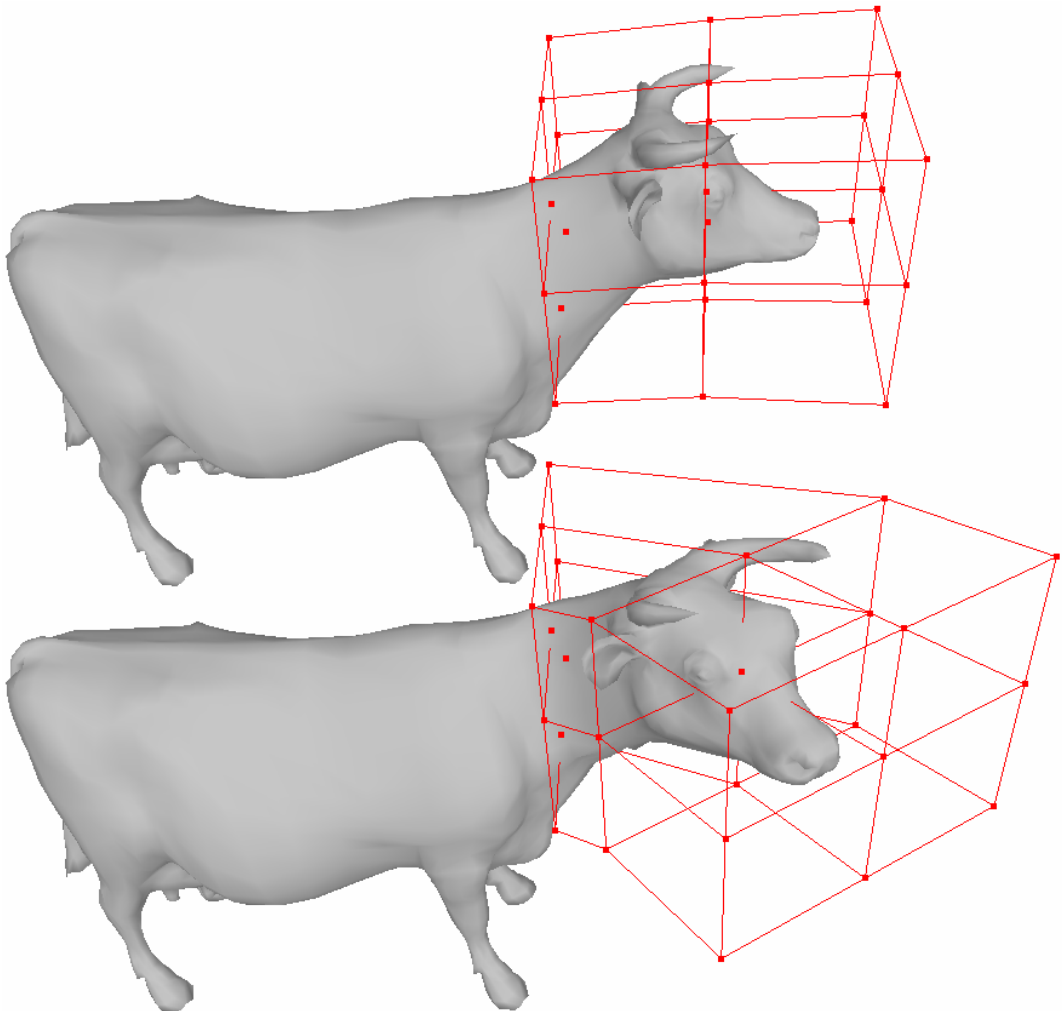


Obrázek 1: Těleso z polygonů a těleso z plátů.

Vytváření složitějších povrchů z malých částí může být velmi náročné a nepřehledné, proto byly vyvinuty metody dodatečného tvarování, které obvykle označujeme jako *deformace*. Podle toho, zda chceme objekt tvarovat v celém rozsahu nebo pouze v jeho části, rozdělujeme *deformace* na *globální* a *lokální*. V prvním případě se transformace aplikuje na všechny body tělesa. V případě *lokálních deformací* se nejprve vytvoří lokální souřadný systém v požadované oblasti modelu, a v něm se těleso deformuje. Následující obrázky prezentují oba typy deformací. Konvice na obrázku 2 byla deformována globálně operací „squeeze“. Na dalším obrázku s krávou je mřížka, která vymezuje lokální souřadný systém v oblasti hlavy. Manipulací bodů na mřížce se provádí deformace a můžeme tak otáčet s hlavou krávy.



Obrázek 2: Globální deformace čajové konvice.



Obrázek 3: Lokální deformace krávy.

### 3 Globální deformace

Nejznámějším zástupcem této třídy modelovacích technik jsou *Barrovy globální deformace* [5]. Jedná se o transformace, které jsou aplikovány na prostorové těleso a mění tvar tohoto objektu v celém rozsahu. Obecný tvar deformace je:

$$X = F_x(x), Y = F_y(y), Z = F_z(z),$$

kde  $[x, y, z]$  je původní bod modelu,  $[X, Y, Z]$  je bod po aplikaci deformace a  $F_x, F_y, F_z$  jsou deformační funkce pro jednotlivé osy. Ukážeme si definice tří nejznámějších *Barrových deformací*:

1. Změna měřítek jednotlivých os (scaling):

$$X = a_1x,$$

$$Y = a_2y,$$

$$Z = a_3z,$$

kde  $a_1, a_2, a_3$  jsou koeficienty pro míru stlačení či natáhnutí ve směru souřadných os.

2. Zešpičatění (tapering) podle zvolené osy. Vybereme osu zešpičatění a plynule měníme měřítko na zbývajících osách. Např. pro zešpičatění objektu ve směru osy  $z$  použijeme transformace

$$X = r_x x,$$

$$Y = r_y y,$$

$$Z = z,$$

kde  $r_x = f(z)$ ,  $r_y = g(z)$  jsou lineární nebo nelineární funkce.

3. Zkroucení (twisting) podle jedné osy se provede obdobně jako u zešpičatění. Zkroucení okolo osy  $z$  realizuje transformace

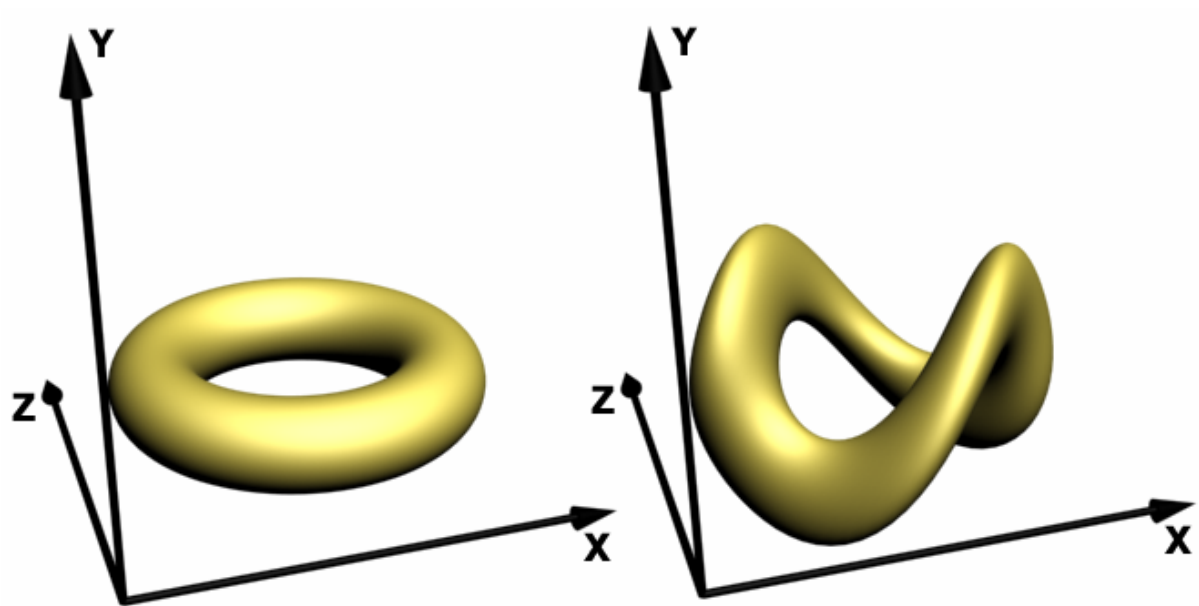
$$X = xC_\theta - yS_\theta,$$

$$Y = xS_\theta + yC_\theta,$$

$$Z = z,$$

kde  $C_\theta = \cos(\theta)$ ,  $S_\theta = \sin(\theta)$  a  $\theta = f(z)$  je funkce mající podobný význam jako u zešpičatění.





Obrázek 4: Zkroucený torus podle osy  $x$ .

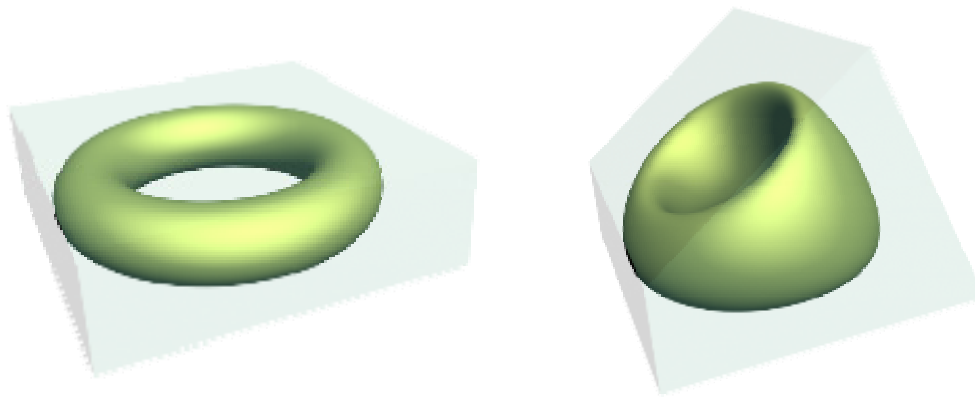
Všechny tyto transformace mají jedno společné. Mění objekt v celém jeho rozsahu a na každý bod se aplikuje stejná transformace. Tyto deformace jsou velmi jednoduché a snadné pro implementaci, ale mají svá omezení. Deformační operace jsou obecně nelineární a nelze je proto skládat s jinými transformacemi. Navíc jsou tyto deformace použitelné pouze pro celý objekt a nelze jimi, bez újmy na spojitosti, měnit lokální část objektu.

## 4 Lokální deformace

I když jsou *Barrovy globální deformace* užitečným a běžně používaným nástrojem pro modelování, nelze je použít pro libovolné deformační operace. Každý typ *Barrovy* deformace umožňuje pouze jednu operaci v rámci celého tělesa. Sederberg a Parry přišli s metodou nazvanou *Free-Form Deformation (FFD)* [1], v češtině nazývané *volné deformace*. Tato technika umožňuje deformovat tělesa libovolným (volným) způsobem. Může být použita na modely CSG nebo na modely popsané hraniční reprezentací. Lze ji aplikovat na tělesa omezená libovolnými analytickými povrchy, jako jsou roviny, kvadriky, parametrické plochy nebo implicitně definované povrchy. *Volné deformace* lze aplikovat na těleso lokálně i globálně, ale protože se tato metoda používá většinou pro lokální úpravy těles, byla zařazena do kapitoly *lokálních deformací*.

### 4.1 Volné deformace

Pro snadné pochopení *FFD* si představme průhledný elastický materiál ve tvaru 4-bokého hranolu, ve kterém je ponořen předmět našeho modelování. Následně měníme tvar tohoto hranolu a s ním se mění i jeho obsah.

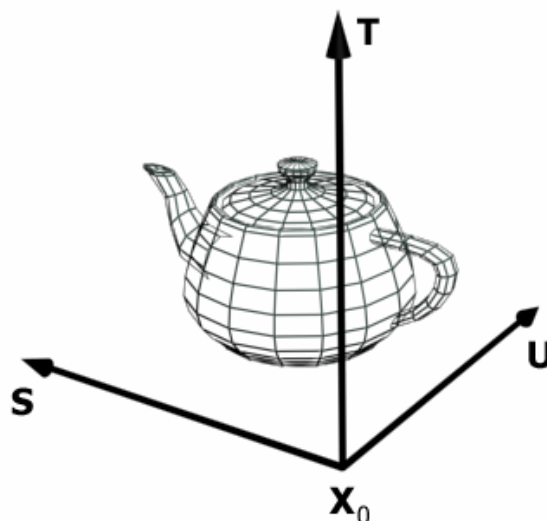


Obrázek 5: Aplikace FFD transformací krajních bodů hranolu.

To znamená, že se vytvoří uzavřená oblast pro účely deformace. Tuto oblast určitým způsobem měníme a vše, co je uvnitř, je podle toho deformováno. Na obrázku 5 je torus uzavřený v průhledném hranolu, který reprezentuje deformační oblast. Vytažením dvou krajních bodů hranolu deformujeme i torus. Je to velmi intuitivní technika a velmi oblíbená mezi designéry.

### 4.1.1 Lokální souřadný systém

Uzavřenou deformační oblast reprezentuje lokální souřadný systém s počátkem v bodě  $X_0$  ve světových souřadnicích a trojicí lineárně nezávislých vektorů  $S$ ,  $T$ ,  $U$ , které svojí orientací a velikostí vymezují rovnoběžnostěn.



Obrázek 6: Lokální souřadný systém.

Každý bod  $X$  deformovaného tělesa se světovými souřadnicemi  $[x, y, z]$  má v tomto lokálním systému souřadnice  $[s, t, u]$  pro které platí:

$$X = X_0 + sS + tT + uU.$$

Tyto lokální souřadnice jsou důležité pro výpočet deformace. S pomocí lineární algebry nalezneme  $[s, t, u]$  řešením rovnic:

$$s = \frac{T \times U (X - X_0)}{T \times U \cdot S}, \quad t = \frac{S \times U (X - X_0)}{S \times U \cdot S}, \quad u = \frac{S \times T (X - X_0)}{S \times T \cdot U}. \quad (1)$$

Pro zjednodušení a bez újmy na obecnosti budeme předpokládat, že  $0 < s, t, u < 1$ . Body, které nesplňují tuto podmínku, leží mimo lokální souřadný systém. Za tohoto předpokladu můžeme rovnice zjednodušit do tvaru:

$$s = \frac{(x - X_0^x)}{|S|}, \quad t = \frac{(y - X_0^y)}{|T|}, \quad u = \frac{(z - X_0^z)}{|U|}, \quad (2)$$

kde  $|\cdot|$  je norma vektoru a  $X_0^x, X_0^y, X_0^z$  jsou hodnoty bodu  $X_0$ . Tento tvar rovnice lze použít v případě, že vektory  $[S, T, U]$  jsou souhlasně rovnoběžné s osami  $[X, Y, Z]$  v našem kartézském systému souřadnic. V případě nesouhlasně rovnoběžného vektoru lze pro něj upravit rovnici (2) otočením výrazů v čitateli. Například za předpokladu, že vektor  $S$  je nesouhlasně rovnoběžný s osou  $X$ , získáme lokální souřadnici  $s$  jako:

$$s = \frac{(X_0^x - x)}{|S|},$$

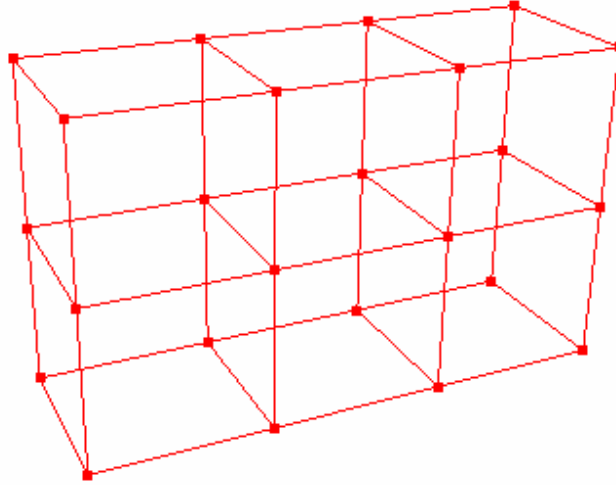
analogicky můžeme upravit rovnice pro ostatní vektory.

#### 4.1.2 Prostorová mřížka

Pro účely deformace potřebujeme pružný nástroj, který nám umožní měnit strukturu lokálního souřadného systému. K tomu slouží pravidelná prostorová mřížka kontrolních bodů  $P_{i,j,k}$ , kde  $i, j, k$  jsou indexy. Tyto body definujeme trojicí  $(l \times m \times n)$ , kde  $l+1$  je počet kontrolních bodů při vektoru  $S$ ,  $m+1$  při vektoru  $T$  a  $n+1$  při  $U$ . Tyto body jsou uniformně rozloženy po celém objemu lokálního prostoru. To znamená, že formují  $l+1$  rovin ve směru  $S$ ,  $m+1$  rovin ve směru  $T$  a  $n+1$  rovin ve směru  $U$  a platí  $l > 0; m > 0; n > 0; i = 0, \dots, l; j = 0, \dots, m; k = 0, \dots, n$ . Pozice kontrolních bodů ve světových souřadnicích lze vypočítat součtem vektorů  $[S, T, U]$  s váhou příslušného indexu:

$$P_{i,j,k} = X_0 + \frac{i}{l}S + \frac{j}{m}T + \frac{k}{n}U.$$

Pro minimální hodnoty  $l, m, n$ , což odpovídá trojici  $(1 \times 1 \times 1)$ , má prostorová mřížka celkem 8 kontrolních bodů, což odpovídá kvádru s řídicími body na jeho rozích. Na následujícím obrázku je znázorněna prostorová mřížka pro  $l = 1, m = 2$  a  $n = 3$ . Je vidět, že mřížka formuje celkem 9 rovin, 2 ve směru vektoru  $S$ , 3 ve směru  $T$  a 4 pro  $U$ .



Obrázek 7: Prostorová mřížka.

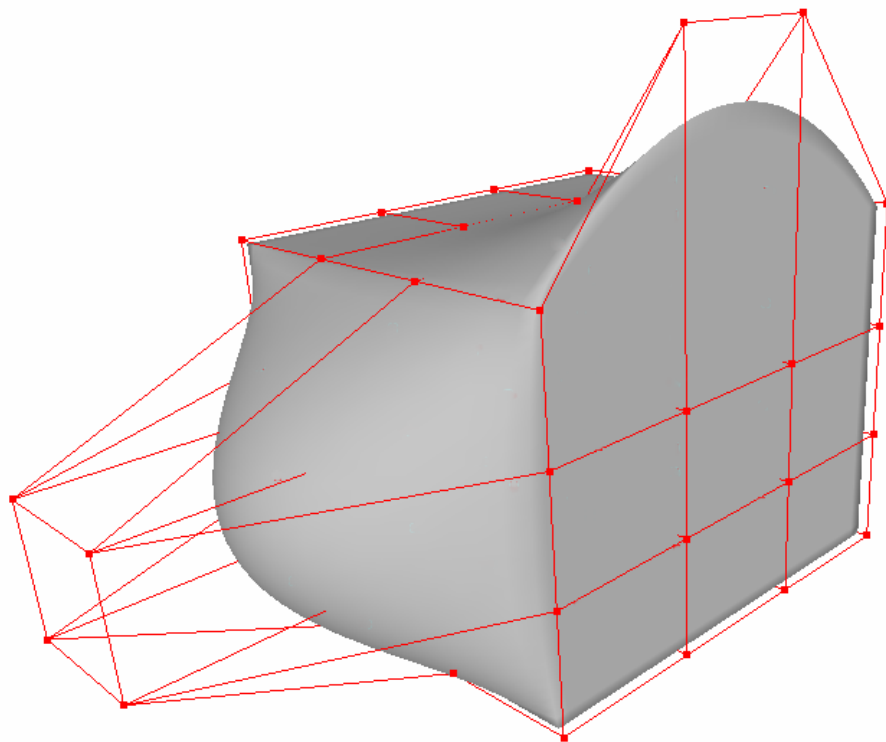
### 4.1.3 Deformace FFD

Minimální kvádr, který obsahuje mřížku, vymezuje lokální souřadný prostor. Princip deformace spočívá v přemístování kontrolních bodů  $P_{i,j,k}$  na mřížce. Před začátkem deformace musí mít mřížka strukturu rovnoběžnostěnu, neboť v tomto případě, jak uvidíme dále, nedochází ke změně polohy bodů, tj. deformační kvádr „nedeformuje“. Poté začneme deformovat, tj. přemísťovat řídicí body na mřížce. Po deformaci, tedy po přemístění jednoho a více řídicích bodů  $P_{i,j,k}$ , je pro každý bod tělesa uvnitř prostorové mřížky určena jeho nová pozice  $X_{ffd}$  ve dvou krocích. Nejprve se podle vzorce (1) vyjádří jeho lokální souřadnice  $[s, t, u]$  a podle něj se určí nová deformovaná poloha bodu ve světových souřadnicích. Každý kontrolní bod  $P_{i,j,k}$  působí určitou vahou na body modelu. To znamená, že pro body tělesa ležící v blízkém okolí řídicího bodu jsou váhy největší a vzdáleností postupně klesají. Pro zajištění požadované spojitosti povrchu jsou nejvhodnější různé polynomiální báze. Sederberg a Parry použili pro výpočet deformace systém pro vyjádření *Bézierova objemu*:

$$\begin{aligned}
 X_{ffd}(s, t, u) &= \sum_{i=0}^l \sum_{j=0}^m \sum_{k=0}^n P_{i,j,k} B_{i,l}(s) B_{j,m}(t) B_{k,n}(u) = \\
 &= \sum_{i=0}^l \binom{l}{i} (1-s)^{l-i} s^i \left[ \sum_{j=0}^m \binom{m}{j} (1-t)^{m-j} t^j \left[ \sum_{k=0}^n \binom{n}{k} (1-u)^{n-k} u^k P_{i,j,k} \right] \right], \quad (3)
 \end{aligned}$$

kde  $B_{i,l}(s)$  označuje  $i$ -tý *Bernsteinův polynom* stupně  $l$  a  $P_{i,j,k}$  jsou nové pozice kontrolních bodů ve světových souřadnicích. Samozřejmě lze použít i jinou polynomiální bázi, jako je *uniformní kubický B-spline* či *NURBS*. Nově získaný bod  $X_{ffd}$  je tedy vypočten dosazením jeho lokálních souřadnic  $[s,t,u]$  do rovnice pro vyjádření *Bézierova objemu*, kde  $P_{i,j,k}$  slouží jako koeficient *Bernsteinových polynomů*.

Při deformaci se hrany šestistěnu vymežující deformační prostor chovají jako *Bézierovy křivky* určené podle kontrolních bodů ležících na příslušných hranách a obdobně všechny stěny jsou transformovány podle předpisu *Bézierových plátů* vzhledem ke kontrolním bodům ležících na těchto stěnách. Tento vztah vychází z toho, že *Bézierův objem* je prostým rozšířením *Bézierových plátů* o jednu dimenzi, stejně jako *Bézierův plát* vychází z *Bézierovy křivky*. Následující obrázek prezentuje tyto vlastnosti.



Obrázek 8: Deformovaný kvádr.

Při volbě mřížky musíme zvážit počet jejích dělicích ploch (dimenzi) nastavením správných hodnot  $l, m, n$ . Ty určují počet kontrolních bodů, které ovlivňují stupeň spojitosti (hladkosti) povrchu. Je zřejmé, že pro  $l = m = n = 1$  bude celkem osm kontrolních bodů, každý na rohu hranolu, a deformace bude lineární, viz. obrázek 5. Vhodnou volbou kontrolních bodů zaručíme požadovaný stupeň hladkosti povrchu.

#### 4.1.4 Algoritmus FFD

Shrneme si předchozí postup do přehledného algoritmu.

##### Algoritmus FFD:

1. Připravíme těleso pro deformaci.
2. Vytvoříme prostorovou mřížku a určíme její dimenzi  $(l, m, n)$ .
3. Manipulací v prostoru přesuneme mřížku do požadované oblasti deformace.
4. Pro každý bod tělesa, který je uvnitř mřížky, vypočítáme jeho lokální souřadnice  $[s, t, u]$  podle rovnice (1) a uložíme je do pole.
5. Cyklus:
  - 5.1 Měníme polohu kontrolních bodů na mřížce.
  - 5.2 Pro každý klíčový bod tělesa, který byl uvnitř mřížky před bodem 5. spočítáme jeho novou pozici dosazením jeho lokální souřadnice z bodu 4. do rovnice (3).
  - 5.3 Uložíme novou pozici tělesa.
6. Konec.

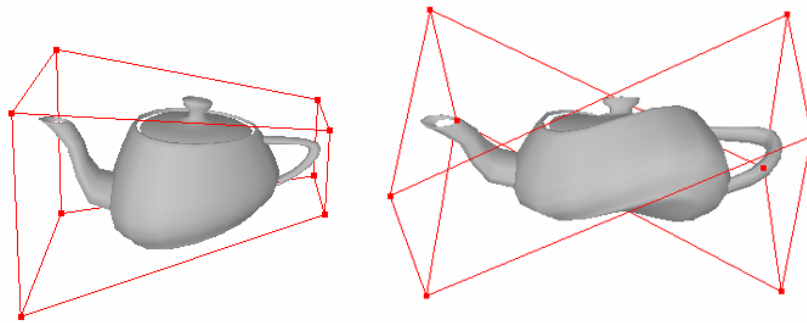
#### 4.1.5 Omezení

Omezující podmínkou původního algoritmu *volných deformací* je skutečnost, že mřížka musí mít před transformací strukturu pravidelného 4-bokého hranolu. Klíčové pro výpočet je nalezení lokálních souřadnic, při této struktuře nám stačí dosadit příslušné hodnoty do lineární rovnice. Je dobré si uvědomit, že výpočet lokálních souřadnic  $[s, t, u]$  může proběhnout jenom jednou před začátkem deformace. Při opakovaných změnách pozic kontrolních bodů vycházíme stále ze stejných předpočítaných souřadnic  $[s, t, u]$ . Nebylo by ani možné spočítat nové lokální souřadnice, neboť po změně kontrolních bodů dojde k narušení struktury mřížky a to znemožní nalezení nových souřadnic pomocí (1). Vzhledem k tomu, že nalezení  $[s, t, u]$  je velmi levná operace (v konstantním čase), může být výhodnější ušetřit paměť pro předpočítané souřadnice a při každé nové transformaci je znovu vyčíslit.

#### 4.1.6 Shrnutí

Doménou *volných deformací* je jejich všestrannost. *FFD* můžeme aplikovat na libovolný geometrický objekt zadaný parametrickou rovnicí či při implicitním vyjádření. Pro *FFD* platí

jedna důležitá vlastnost, a to, že deformací parametrického povrchu získáme opět parametrický povrch. Jestliže je parametrický povrch definován jako  $x = f(\alpha, \beta)$ ,  $y = g(\alpha, \beta)$ ,  $z = h(\alpha, \beta)$  a FFD je dáno  $X_{ffd} = X(x, y, z)$ , potom je deformovaný parametrický povrch dán jako  $X_{ffd}(\alpha, \beta) = X(f(\alpha, \beta), g(\alpha, \beta), h(\alpha, \beta))$ . FFD můžeme aplikovat na těleso globálně a snadno simulovat *Barrovy globální deformace*.



Obrázek 9: Zešpičatění a zkroucení pomocí FFD.

#### 4.1.7 Implementace FFD

V této kapitole se budeme věnovat možné implementaci *volných deformací* s ohledem na optimalizaci.

##### Prostorová mřížka

Vhodnou datovou strukturou se jeví trojrozměrné pole o velikosti  $(l+1 \times m+1 \times n+1)$ , kde každou položku bude reprezentovat jeden kontrolní bod s údaji o pozici ve světových souřadnicích a dalšími atributy vhodné pro vizualizaci bodu. Mřížku tedy indexujeme stejně jako dle definice pomocí  $i, j, k$ , kde  $i = 0, \dots, l$ ;  $j = 0, \dots, m$ ;  $k = 0, \dots, n$ .

##### Lokální souřadnice

Po vytvoření mřížky, tedy nastavení její dimenze, pozice a velikosti, vypočteme převrácené hodnoty délky vektorů  $S, T, U$ . Tedy  $1/|S| = 1/\sqrt{S_x^2 + S_y^2 + S_z^2}$ , analogicky pro  $T$  a  $U$ . Jak již bylo zmíněno, musíme se rozhodnout, kdy budeme souřadnice  $[s, t, u]$  počítat. Jejich předpočítání vyžaduje paměť o velikosti bodů deformovaného tělesa. V tomto případě procházíme body modelu a podle rovnice (1) nebo (2) vyčíslíme  $[s, t, u]$ . V paměťově šetrnější variantě počítáme lokální souřadnice bodu při deformaci před dosazením do (3), k tomu



využijeme převrácenou hodnotu vektorů  $S$ ,  $T$ ,  $U$  a počítáme pomocí (2). To jsou celkem tři aritmetické operace na vektorech a pohodlně jsme se vyhnuli dělení.

### Deformace

Dosazení  $[s, t, u]$  do (3) realizujeme snadno procházením indexů  $i, j, k$  ve vnořených cyklech a přiřítáním násobků *Bernsteiových polynomů* s pozicemi kontrolních bodů. Zde využijeme toho, že je mřížka reprezentována trojrozměrnou maticí a  $P_{i,j,k}$  indexujeme podle toho. Nejslabší částí algoritmu je výpočet *Bernsteinových polynomů*. Ty lze před průchodem cyklu vypočítat jednou pro hodnoty  $[s, t, u]$  a  $l, m, n$  a vytvořit si pro ně tabulku. Podle definice

$$B_{i,l}(s) = \binom{l}{i} (1-s)^{l-i} s^i$$

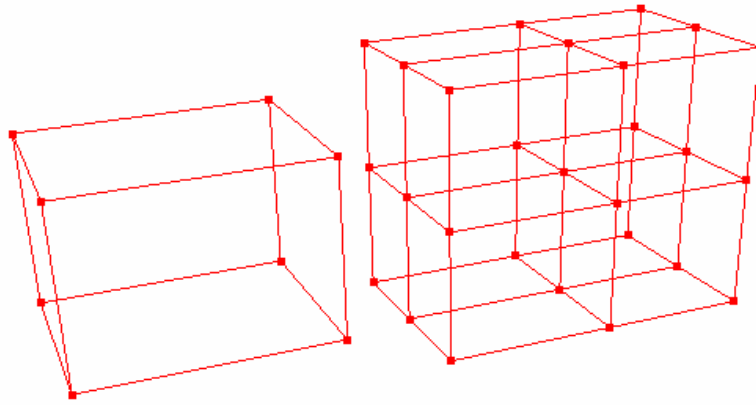
neobsahuje proměnnou hodnotu  $s$ , ale pouze konstanty. Proto je výhodné při inicializaci vytvořit tabulku binomických koeficientů.

## 4.2 Rozšířené volné deformace

*Volné deformace* jsou efektivním nástrojem v rukou designéra, avšak hlavním problémem je počáteční struktura mřížky. Ta musí mít tvar rovnoběžnostěny a tím jsme omezeni vytvářet mřížky různých topologií. V roce 1990 přišla Sabine Coquillart s metodou nazvanou Extended Free-Form Deformation (EFFD) [2], neboli *Rozšířené volné deformace*, která nám dovoluje používat mřížku, až na jistá omezení, libovolného tvaru. Princip deformace zůstává stejný jako u *FFD*, klíčovou úlohu hraje definování mřížky a výpočet lokálních souřadnic.

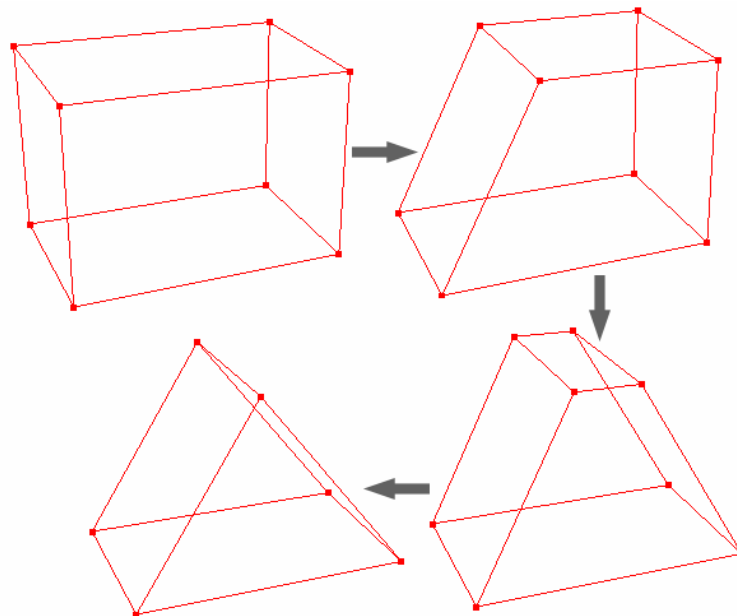
### 4.2.1 Prostorová mřížka EFFD

V klasických *volných deformacích* je mřížka definována trojicí  $(l \times m \times n)$  kontrolních bodů ležících uniformně rozděleny uvnitř lokálního souřadného systému. Mřížka *EFFD* vychází z podobného principu. Coquillart zde zavedla pojem *chunk*, který označuje nejmenší dílčí element mřížky. Ten je formován z osmi kontrolních bodů jako u *FFD* pro  $l = m = n = 1$ .



Obrázek 10: Samostatný *chunk* a mřížka sestavená z osmi *chunků*.

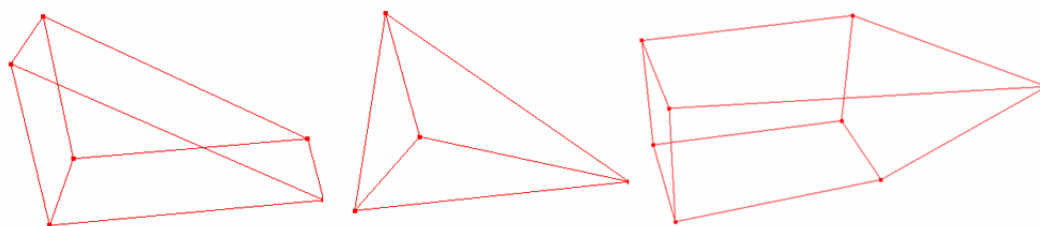
Na rozdíl od *FFD*, kde každý element musí mít tvar rovnoběžnostěnu, na *chunk* toto omezení neklademe. Mřížka *EFFD* je sestavena z *chunků*, které mohou být transformovány svými řídicími body do požadované pozice. Mezi legální transformace patří pouze přesun kontrolních bodů. Nemůžeme bod smazat ani přidat. Tedy každý *chunk* mřížky *EFFD* má topologii jako na obrázku 10. Příklad transformace ilustruje následující obrázek.



Obrázek 11: Transformace *chunku*.

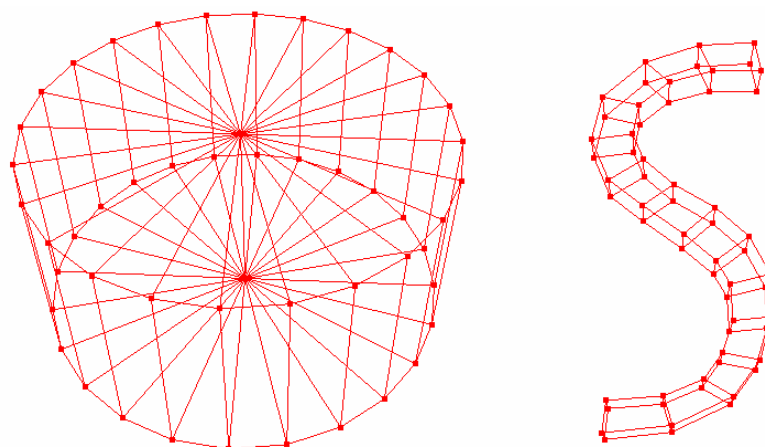
Topologie zůstane zachována, dvojice horních kontrolních bodů byly pouze přesunuty do stejné pozice.

Obrázek 12 prezentuje správně utvořené *chunky*, první dva vznikly „spojením“ několika kontrolních bodů, poslední představuje nepravidelný šestistěn.



Obrázek 12: Správně utvořené *chunky*.

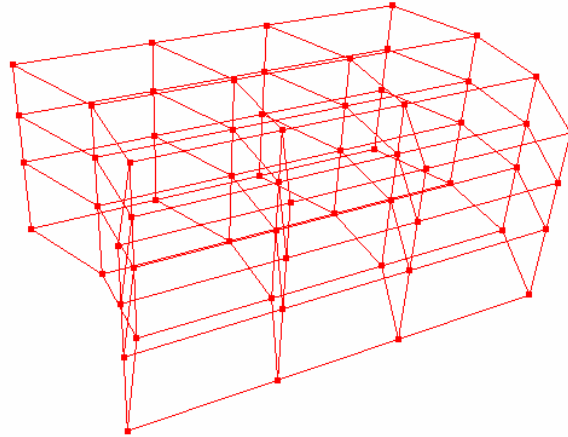
Samotná mřížka je pak sestavena z těchto stavebních elementů. Mřížka je správně utvořena, jestliže všechny *chunky* jsou správně utvořeny. Sousední *chunky* jsou na sebe napojeny svými kontrolními body. Tímto způsobem můžeme snadno vytvořit mřížku rozličných tvarů, např. válcového tvaru či podobnou písmenu *S*. S dostatečně podpůrnými nástroji záleží jen na fantazii designéra.



Obrázek 13: Mřížky *EFFD*.

Každý jednotlivý *chunk* vymezuje v sobě lokální souřadný systém, proto výpočet souřadnic provádíme vždy v rámci jednoho elementu. Coquillart definovala *chunk* jako *Bézierův objem*, který je vymezen 64 kontrolními body:

$$Chunk(s, t, u) = \sum_{i=0}^3 \sum_{j=0}^3 \sum_{k=0}^3 B_{i,3}(s) B_{j,3}(t) B_{k,3}(u) P_{i,j,k}.$$

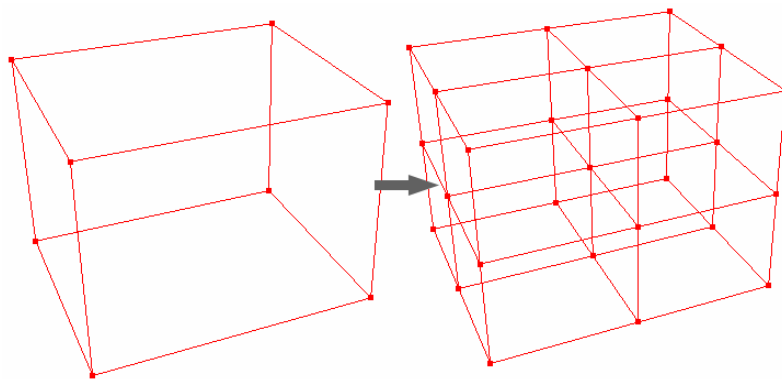


Obrázek 14: *Chunk* dle Coquillart.

Při vlastní implementaci byl zvolen model *chunku* jako *Bézierův objem* prvního stupně s 8 kontrolními body:

$$Chunk(s, t, u) = \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k}. \quad (4)$$

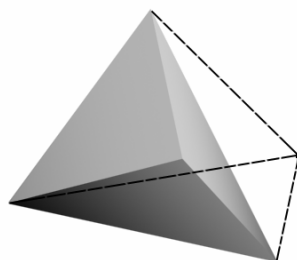
Tento model sice neumožňuje spojitost povrchu takového stupně jako v případě modelu Coquillart, ale jednoduchým trikem lze zvýšit počet kontrolních bodů *chunku* a uspokojit tak požadavky na křivost povrchu. Ten spočívá v pravidelném rozdělení *chunku* na 8 dílčích objemů, viz. obrázek 15.



Obrázek 15: Dělení *chunku*.

### 4.2.2 Výpočet lokálních souřadnic

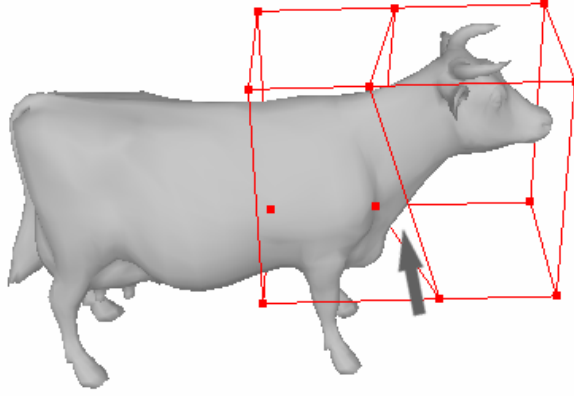
Ten se skládá ze dvou částí. Nejprve je třeba zjistit pro každý bod tělesa, ve kterém *chunku* leží. Tedy každý bod uvnitř prostorové mřížky musí mít určen jeden jednoznačný *chunk*. Tento požadavek splníme vypočtením konvexního obalu tohoto elementu. Při implementaci byl použit inkrementální algoritmus IncrementalHull3D [12,13]. Na jeho vstupu je množina osmi kontrolních bodů *chunku*. Tento algoritmus v prvním kroku najde 4 body a vytvoří 4-stěn, v dalším vybere nový kontrolní bod a spustí z něj hrany na body aktuálního konvexního obalu, které jsou v horizontu viditelnosti, a zruší hrany pod ním.



Obrázek 16: Spouštění hran při vytváření konvexního obalu.

Pro zbytek bodů pokračujeme stejným způsobem, na konci dostaneme konvexní obal pro celou množinu. Pro hrany, které tvoří nejmenší cyklus, vytvoříme stěny z trojúhelníků. Pro testování příslušnosti bodu uvnitř konvexního obalu musíme určit roviny pro všechny trojúhelníky a zjistit, zda bod leží v jejich průniku. Rovnici roviny, na které leží trojúhelník, lze určit vypočtením jeho *normálového vektoru*. Je-li trojúhelník definován třemi body  $(p_1, p_2, p_3)$ , *normálový vektor*  $\vec{N}$  získáme vektorovým součinem vektorů určených těmito body, tedy  $\vec{N} = (p_2 - p_1) \times (p_3 - p_2)$ . Rovnice roviny je pak dána  $ax + by + cz + d = 0$ , kde  $\vec{N} = (a, b, c)$ . Bod tělesa  $p = (x, y, z)$  je dosazen do této rovnice, a je-li  $d < 0$  pro všechny roviny trojúhelníků konvexního obalu, je uvnitř.

Pro sousední *chunky* může vzniknout problém, kdy se jejich konvexní obaly překrývají. Proto musíme uvážit případ bodů ležících v jejich průniku. Tento problém lze jednoduše vyřešit. Pokud pro daný bod modelu narazíme na první *chunk*, v jehož konvexním obalu leží, dále tento bod již netestujeme.



Obrázek 17: Sousední *chunky* a překrývající se konvexní obaly.

Když máme pro každý bod modelu uvnitř mřížky určen jednoznačný *chunk*, spočítáme jeho lokální souřadnice  $[s, t, u]$ . Bohužel výpočet není tak přímočarý jako v případě *FFD*, neboť je zde povolena nerovnoběžná struktura mřížky. U mřížky válcového tvaru lze například použít převodu mezi kartézskými a cylindrickými souřadnicemi [6], ale obecně to neřeší náš problém. Další metodou pro nalezení souřadnic je rekurzivní dělení *chunku* podle středového bodu. Aritmetickým průměrem se zjistí střed osmi kontrolních bodů a pro bod tělesa, který je v nejbližším okolí středu se položí  $[s, t, u] = [0.5, 0.5, 0.5]$ . Původní *chunk* rozdělíme podle středu na osm vnitřních elementů a rekurzivně pokračujeme, dokud neobsadíme všechny hodnoty  $[s, t, u]$  nebo nedosáhneme maximální hloubky rekurze. Tato metoda je sice velmi snadná, ale není přesná a vyžaduje rekurzi, což nemusí být vždy žádoucí. Coquillart vyřešila tento problém aproximací souřadnic z rovnice pro výpočet *Bézierova objemu*. Pro každý bod  $X$  deformovaného tělesa platí vztah podle rovnice (4), tedy:

$$X = \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k},$$

kde  $P_{i,j,k}$  jsou kontrolní body *chunku* ve světových souřadnicích, ke kterému je bod  $X$  přiřazen, hodnoty souřadnic  $[s, t, u]$  jsou neznámé. Převodem levé strany na pravou obdržíme nelineární rovnici rovnou nule:

$$\sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k} - X = 0.$$

Protože ale hledáme tři neznámé hodnoty  $[s, t, u]$ , lze využít faktu, že pracujeme ve třech dimenzích a rovnici proto můžeme přepsat na soustavu tří nelineárních rovnic:

$$\begin{aligned} f_x &= \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k}^x - X_x = 0 \\ f_y &= \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k}^y - X_y = 0 \\ f_z &= \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k}^z - X_z = 0, \end{aligned} \quad (5)$$

kde  $P_{i,j,k}^x$ ,  $P_{i,j,k}^y$  a  $P_{i,j,k}^z$  jsou souřadnice kontrolních bodů *chunku* pro jednotlivé osy a  $X = (X_x, X_y, X_z)$ . Tento systém už lze řešit pomocí numerické metody. Existuje několik numerických metod pro řešení soustavy nelineárních rovnic, nejznámější je však *Newtonova iterační metoda* [9]. Vyžaduje sice řešení parciálních derivací, ale v našem případě to nepředstavuje problém.

### Newtonova iterační metoda pro řešení systému nelineárních rovnic:

Je dáno  $m$  nelineárních rovnic o  $m$  neznámých:

$$\begin{aligned} f_1 &= (x_1, \dots, x_m) = 0 \\ &\quad \vdots \\ f_m &= (x_1, \dots, x_m) = 0 \end{aligned} \quad (6)$$

Kořenem systému (6) rozumíme každou uspořádanou  $m$ -tici reálných čísel  $(\xi_1, \dots, \xi_m)$ , která tomuto systému vyhovuje. Systém (6) lze také zapsat ve vektorovém tvaru:

$$F(x) = 0, \quad x \in \mathbb{R}^m, \quad 0 = (0, \dots, 0)^T \in \mathbb{R}^m.$$

Kořen této rovnice značíme  $\xi = (\xi_1, \dots, \xi_m)^T$ . Tuto rovnici ve vektorovém tvaru převedeme na rovnici ekvivalentní:

$$x = G(x), \quad x \in \mathbb{R}^m$$

a budeme hledat pevný bod zobrazení  $G: \mathbb{R}^m \rightarrow \mathbb{R}^m$ . Pro *Newtonovu iterační metodu* platí:

$$G(x) = x - J_F^{-1}(x)F(x),$$

kde

$$J_F(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \dots & \frac{\partial f_m(x)}{\partial x_m} \end{pmatrix}.$$

Matice  $J_F(x)$  je *Jacobiova matice* funkce  $F$ . Necht'  $J_F(x)$  je regulární matice se spojitými prvky v okolí bodu  $\xi$ . *Newtonova iterační metoda* pro systém  $F(x) = 0$  je:

$$x^{k+1} = x^k - J_F^{-1}(x^k)F(x^k), \quad k = 0, 1, 2, \dots \quad (7)$$

V našem případě máme tři rovnice o třech neznámých. *Jacobiova matice* je ve tvaru:

$$J_F(x) = \begin{pmatrix} \frac{\partial f_x}{\partial s} & \frac{\partial f_x}{\partial t} & \frac{\partial f_x}{\partial u} \\ \frac{\partial f_y}{\partial s} & \frac{\partial f_y}{\partial t} & \frac{\partial f_y}{\partial u} \\ \frac{\partial f_z}{\partial s} & \frac{\partial f_z}{\partial t} & \frac{\partial f_z}{\partial u} \end{pmatrix},$$

kde  $f_x, f_y, f_z$  jsou rovnice ze systému (5). Výpočet parciální derivace lze upravit do jednoduchého výrazu, ukážeme si to na příkladu pro  $\frac{\partial f_x}{\partial s}$ .



$$\begin{aligned}
\frac{\partial f_x}{\partial s} &= \frac{\partial \left( \sum_{i=0}^1 \binom{1}{i} (1-s)^{1-i} s^i \left[ \sum_{j=0}^1 \binom{1}{j} (1-t)^{1-j} t^j \left[ \sum_{k=0}^1 \binom{1}{k} (1-u)^{1-k} u^k P_{i,j,k} \right] - X_x \right] \right)}{\partial s} = \\
&= \frac{\left( \sum_{i=0}^1 \left( -\binom{1}{i} \left( \sum_{j=0}^1 \binom{1}{j} \right) \left( \sum_{k=0}^1 \binom{1}{k} P_{i,j,k}^x \left( -\frac{-1+u}{u} \right)^{(-k)} \right) \left( -\frac{-1+t}{t} \right)^{(-j)} \right) * \right. \\
&\quad \left. -(-1+t)(-1+u) \sum_{i=0}^1 \left( \binom{1}{i} \left( -\frac{-1+s}{s} \right)^{(-i)} - i \left( -\frac{-1+s}{s} \right)^{(-i)} s + \left( -\frac{s}{-1+s} \right)^i s i - \left( -\frac{s}{-1+s} \right)^i s \right) \right)}{s} = \\
&= -P_{0,0,0}^x + P_{0,0,0}^x u - u P_{0,0,1}^x + t P_{0,0,0}^x - t P_{0,0,0}^x u + t u P_{0,0,1}^x - t P_{0,1,0}^x + t P_{0,1,0}^x u - t u P_{0,1,1}^x + P_{1,0,0}^x - \\
&\quad - P_{1,0,0}^x u + u P_{1,0,1}^x - t P_{1,0,0}^x + t P_{1,0,0}^x u - t u P_{1,0,1}^x + t P_{1,1,0}^x - t u P_{1,1,0}^x + t u P_{1,1,1}^x.
\end{aligned}$$

Podobně se dají zjednodušit i ostatní parciální derivace. Výraz  $F(x)$  vypočítáme dosazením aktuální aproximace  $x^k$  do rovnic (5).

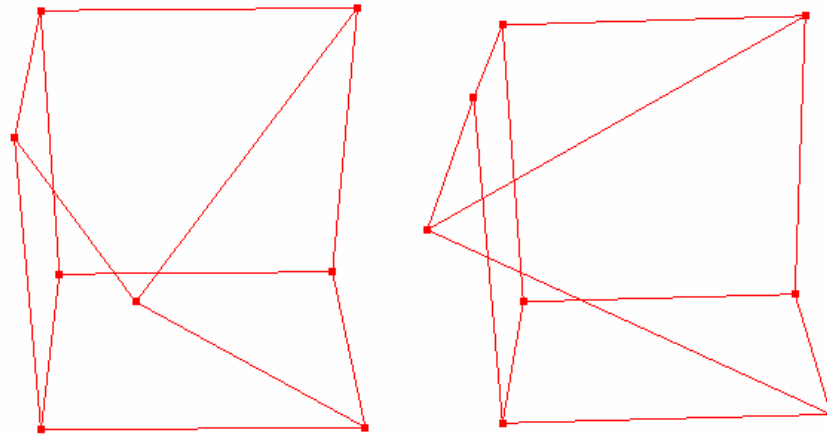
Tato iterační metoda sice předpokládá, že počáteční aproximace  $x^0$  leží dostatečně blízko pevného bodu  $\xi$ , ale podle zkušeností se ukázalo, že počáteční hodnota  $x^0 = (0.5, 0.5, 0.5)$  vede k velmi dobré konvergenci kromě případů s degenerativní strukturou mřížky, které představíme v další kapitole. Při výpočtu tedy začneme s hodnotou  $x^0 = (0.5, 0.5, 0.5)$  a další iteraci  $x^1$  obdržíme aplikací *Newtonovy iterační metody*:

$$x^1 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} - J_F^{-1} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} F \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}.$$

V iteraci pokračujeme dokud  $F(x^k) < \varepsilon$  pro dostatečně malé  $\varepsilon$ , zpravidla pokládáme  $\varepsilon = (0.001, 0.001, 0.001)^T$ . Tato metoda konverguje velmi rychle, výsledky ukázaly, že průměrný počet iterací k nalezení  $[s, t, u]$  je tři.

### 4.2.3 Problematické případy

Nyní zvážíme některé případy, které mohou vést ke špatnému výsledku iterační metody. V prvním případě problém divergence může nastat při degenerativní struktuře *chunku*. Za degenerativní *chunk* lze považovat ten, jehož kontrolní bod je dostatečně vzdálený od své počáteční pozice uvnitř konvexního obalu nebo protíná některou jeho stěnu, viz. obrázek 18:



Obrázek 18: Degenerativní *chunky*.

Tento problém lze dodatečně řešit prostým dělením *chunku* a opakovanému hledání lokálních souřadnic. Lze však s velkou jistotou tvrdit, že při praktickém použití není potřeba takto *chunk* degenerovat. Ve vlastní implementaci byla divergence odhalena pouze při experimentálních pokusech s mřížkou, tedy při úmyslném hledání těchto případů. Dalším problémem k nutnému uvážení je výpočet inverze *Jacobiové matice*. Jak je známo, matice má inverzi, právě když je *regulární* a její *determinant* je nenulový. V naší implementaci byl případ *singulární Jacobiové matice* nalezen pouze při „ošklivě“ degenerované mřížce, kdy kontrolní body protínaly protější stěny konvexního obalu, tedy opět v extrémním případě. Tento problém lze ošetřit aproximací inverze, a to použitím *pseudo-inverzní matice*  $J^+ = (J^T J)^{-1} J^T$ , kde  $J^T$  je transponovaná matice. Další možností je samozřejmě použít jiné numerické metody, která nevyžaduje inverzi matice. Nelze však zaručit, že takto nalezené souřadnice povedou k očekávaným výsledkům při deformaci.

Výše zmíněná numerická metoda se aplikuje na každý bod tělesa uvnitř mřížky, není snad nutno dodávat, že se jedná o velmi drahou operaci. Ale stejně jako u *FFD* stačí, aby výpočet  $[s, t, u]$  proběhl jenom jednou a uložil se v paměti. Při optimalizaci je proto vhodnější se soustředit na výpočet samotné deformace.

#### 4.2.4 Deformace EFFD

Výpočet deformace se provede stejně jako v případě *FFD*. Po správném nastavení mřížky se vypočítají lokální souřadnice  $[s, t, u]$ . Manipulací kontrolních bodů měníme pozice  $P_{i,j,k}$  a počítáme deformaci dosazením do rovnice (4):

$$X_{effd}(s, t, u) = \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 B_{i,1}(s) B_{j,1}(t) B_{k,1}(u) P_{i,j,k},$$

kde  $X_{effd}$  je nově vypočtená pozice bodu ve světových souřadnicích,  $[s, t, u]$  jsou lokální souřadnice bodu původního a  $P_{i,j,k}$  jsou kontrolní body *chunku*, ve kterém původní bod ležel.

#### 4.2.5 Algoritmus EFFD

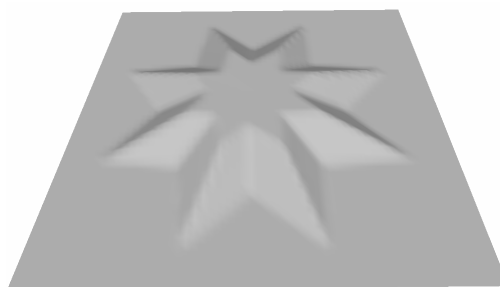
Shrneme si opět předchozí postup do přehledného algoritmu.

##### Algoritmus EFFD:

1. Připravíme těleso pro deformaci.
2. Vytvoříme prostorovou mřížku skládající se z jednotlivých *chunků*.
3. Manipulací v prostoru přesuneme mřížku do požadované oblasti deformace.
4. Vypočítáme konvexní obaly všech *chunků* a každému bodu tělesa uvnitř mřížky přidělíme jednoznačný *chunk*, ve kterém leží.
5. Newtonovou iterační metodou (7) spočítáme lokální souřadnice  $[s, t, u]$ , jako počáteční aproximaci volíme vektor  $(0.5, 0.5, 0.5)$ , výsledky ukládáme do pole.
6. Cyklus:
  - 6.1 Provádíme manipulaci kontrolních bodů na mřížce.
  - 6.2 Pro každý bod tělesa, který byl uvnitř *chunku* před bodem 6., spočítáme jeho novou pozici dosazením jeho lokální souřadnice z bodu 5. do rovnice (4).
  - 6.3 Uložíme novou pozici tělesa.
- 6.7 Konec.

#### 4.2.6 Shrnutí

Rozšířené volné deformace jsou velmi efektivním nástrojem při modelování těles. Deformování pomocí prostorové mřížky je velice intuitivní technika a lze tak měnit povrchy těles způsobem, který se dá jinými technikami nahradit jen velmi těžko. *EFFD* lze považovat jako sochařský nástroj pro modelování těles. Můžeme jím měnit libovolný povrch do tvarů určených prostorovou mřížkou. Hvězda na následujícím obrázku byla vytvořena definováním mřížky válcového tvaru a kontrolními body byla „vtažena“ z povrchu rovinného tvaru.



Obrázek 19: Hvězda.

Jednoduše lze také vymodelovat například ubrus na stole, který by byl modelován jinými nástroji velmi pracně.



Obrázek 20: Ubrus.

Při implementaci animačních technik by bylo možné simulovat například tkaninu vlající ve větru. Rozšiřující možnosti této deformace jsou opravdu velké.

Implementace samotné deformace je poměrně snadná, důležité je se zaměřit na dostatek podpůrných nástrojů pro definování mřížky a usnadnit tak práci designérům. Topologické omezení mřížky se nedá považovat za velkou překážku, neboť dělením *chunků* či spojováním

kontrolních bodů lze vytvářet mřížky, které musí uspokojit každého náročného designéra. Avšak existuje již algoritmus *volných deformací pro mřížky libovolné topologie* [3]. Ten spočívá v rekurzivním dělení struktury na jednodušší topologie a následný výpočet deformace je analogický jako u *EFFD*.

#### 4.2.7 Implementace EFFD

V této části si ukážeme postup pro možnou implementaci *EFFD*. Hlavně se zaměříme na výpočet lokálních souřadnic  $[s, t, u]$ . Výpočet deformace a definice mřížky je obdobná jako u *FFD*.

##### Prostorová mřížka

*Chunk* může být implementován jako matice  $(2 \times 2 \times 2)$ , kde význam jednotlivých položek je stejný jako v případě *FFD*. Protože prostorová mřížka je složena z jednotlivých *chunků*, pro její implementaci se jeví nejvhodnější udržování seznamu *chunků*, kde veškeré operace jsou realizovány přes tyto stavební elementy.

##### Výpočet lokálních souřadnic

Tento výpočet je časově velmi náročný, ale výhodou je, že ho provádíme pouze jednou na začátku deformačního procesu. Rozdělíme ho do dvou částí. V prvním bodě výpočet konvexního obalu realizujeme některým známým algoritmem, například inkrementálním algoritmem Hull3D, viz. kapitola 4.2.2. Pro hodnoty  $[s, t, u]$  si vytvoříme paměťové místo o velikosti bodů modelu. Dále procházíme seznam bodů tělesa a testujeme jejich příslušnost uvnitř konvexních obalů jednotlivých *chunků* a zaznamenáváme si výskyty. Poté procházíme všechny body tělesa uvnitř mřížky a počítáme lokální souřadnice pomocí *Newtonovy iterační metody*. Jako iterační chybu je vhodné položit  $\mathcal{E} = (0.001, 0.001, 0.001)$ . Výpočet  $F(x^k)$  lze zjednodušit do jednoho výrazu pro každou dimenzi a urychlit tak výpočet. *Jacobiovu matici* nalezneme výpočtem parciálních derivací, všechny derivace se dají zjednodušit do výrazu popsaném v kapitole 4.2.2. Existenci inverze matice lze testovat určením jejího determinantu a porovnáním s nulou. Samotnou inverzi lze realizovat přímým výpočtem vyjádřením jednotlivých prvků matice. Iteraci provádíme v cyklu a pokaždé testujeme  $F(x^k)$  pro aktuální aproximaci oproti  $\mathcal{E}$ . Příklad divergence detekujeme nastavením horní meze pro počet iterací, v našem případě hodnotou 1000.

### **Deformace**

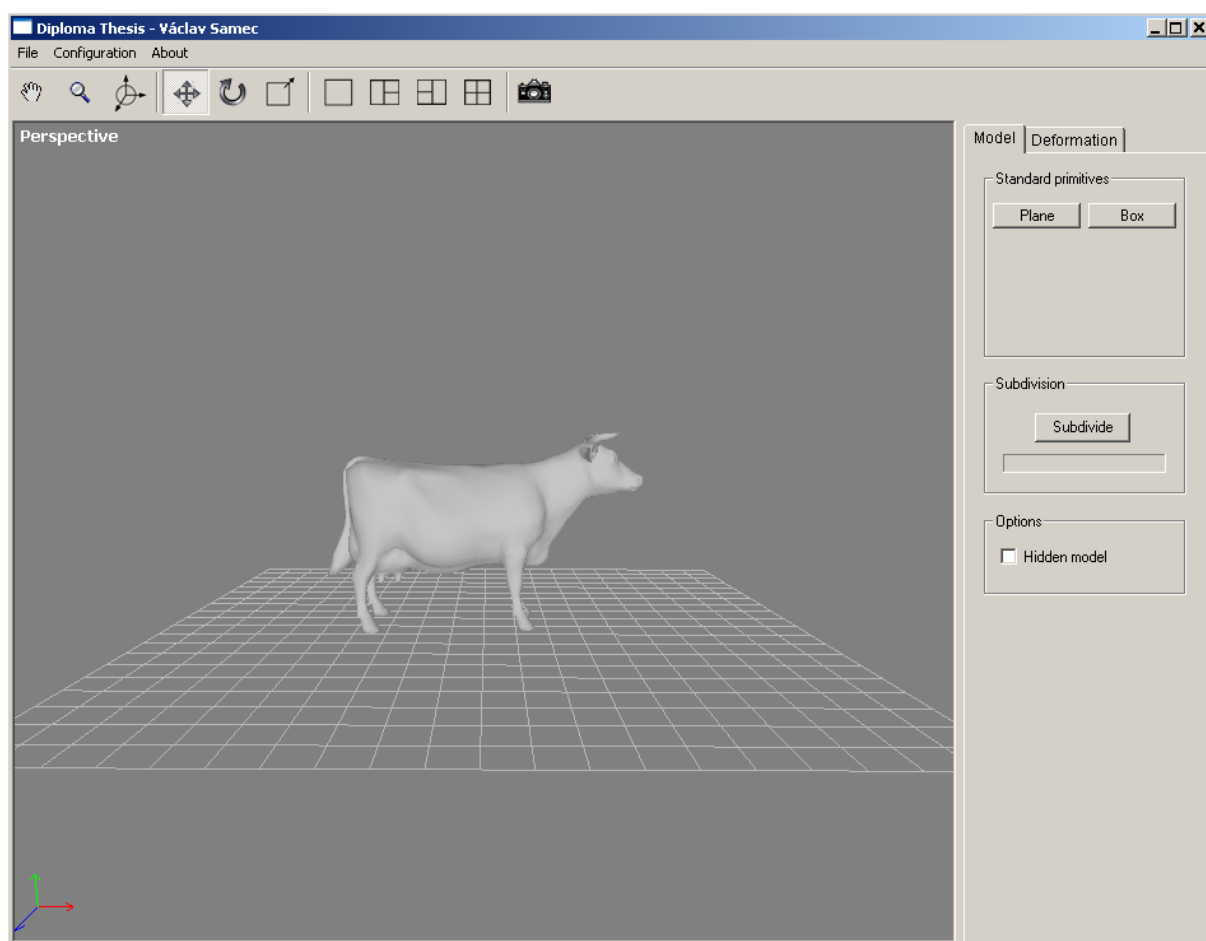
Výpočet deformace realizujeme dosazením hodnot  $[s, t, u]$  do (4). V tomto případě je fixní dimenze mřížky pro  $l = m = n = 1$  a výpočet lze opět zjednodušit do jednoho výrazu pro každou dimenzi.

## 5 Experimentální aplikace

V praktické části této práce byla vytvořena experimentální aplikace, kde byly implementovány výše zmíněné techniky lokálních deformací. Aplikace byla naprogramována v jazyce C++ s použitím knihovny OpenGL. Algoritmy *volných deformací* jsou platformově nezávislé, neboť neobsahují volání systémových funkcí. Program byl vytvořen pro prostředí Windows, podpůrné funkce rozhraní byly implementovány v čistém Win32 API ve vývojovém prostředí Visual C++ 6.0. Rovnice pro výpočet deformací byly odvozeny a zjednodušeny pomocí matematického systému Maple 6.0.

### 5.1 Rozhraní aplikace

Tvorba aplikace byla inspirována modelovacím systémem 3D Studio Max od firmy Descreet. Rozhraní programu je rozčleněno do tří částí: vykreslovací okno, panel nástrojů a panel pro editaci modelu a deformací.



Obrázek 21: Rozhraní aplikace.

Pro vykreslovací okno lze měnit způsob promítání na perspektivní nebo rovnoběžné z pohledu tří stran. Program umožňuje načítání modelů z několika známých grafických formátů pro popis geometrických těles: 3DS (3D Studio Max), PLY (Polygon file), OBJ (Alias Wavefront) a modely z počítačové hry Quake (MDL, MD2, MD3). V rámci aplikace lze vytvořit rovinu nebo kvádr složenou z trojúhelníků. U každého modelu lze zjemnit trojúhelníkovou síť pro lepší výsledky aplikovaných deformací. Vytvořené modely můžeme ukládat do dvou známých textových formátů OBJ a PLY, s kterými umí pracovat většina modelovacích systémů. Snímek obsahu vykreslovacího okna lze ukládat do obrazového formátu PNG.

## 5.2 Přehled nástrojů

- **Pan tool** 


Tímto nástrojem provádíme manipulaci s kamerou do stran ve vykreslovacím okně.

- **Zoom tool** 


Lupou přibližujeme a oddalujeme kameru k modelovanému tělesu.

- **Arcball tool** 

Tento nástroj slouží pro natáčení kamery ve vykreslovacím okně.

- **Move tool** 

Manipulační nástroj slouží k přesunu kontrolních bodů na mřížce. Lze je posouvat ve směru osy  $X$ ,  $Y$  nebo  $Z$ . Alternativou je volný posun bodů, kdy měníme pozice bodů všemi směry podle aktuálního promítání.

- **Rotate tool** 

Opět slouží k manipulaci kontrolních bodů na mřížce. S body lze provádět rotaci okolo jejich středu ve směru souřadných os.



- **Scale tool**



Tento nástroj mění pozice bodů změnou měřítka ve směru souřadných os.

### 5.3 Deformace

V aplikaci jsou implementované obě metody *volných deformací*. Na těleso lze aplikovat jednu nebo více deformací současně. Vybereme deformaci ze seznamu a uložíme ji na zásobník. Na těleso se pak aplikují všechny deformace v pořadí od jeho dna.

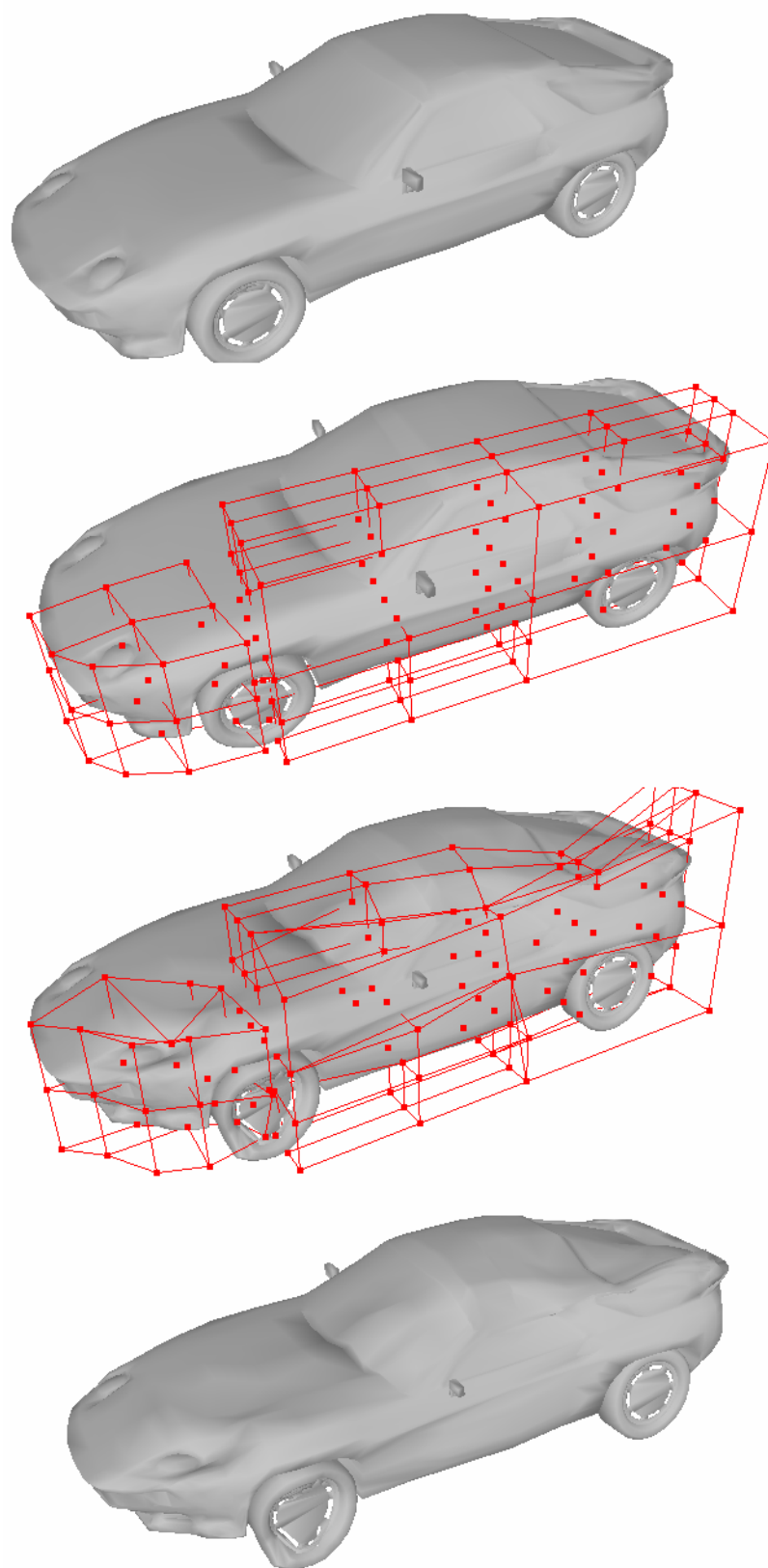
V případě *FFD* specifikujeme na liště dimenzi mřížky a umístíme ji na těleso. Přepneme se do deformačního módu a manipulací kontrolních bodů ho deformujeme.

U *EFFD* jsou dvě předdefinované mřížky. První je klasická mřížka se strukturou rovnoběžnostěny, kde specifikujeme dimenzi stejně jako u *FFD*. Druhá mřížka je válcové topologie, pro kterou lze nastavit počet *chunků* na výšku, šířku a ve směru poloměru. Pro vytvoření jiné struktury mřížky lze využít již předdefinovaných konstrukcí a spojovat je kontrolními body nebo sestavovat mřížku od základů přidáváním jednotlivých *chunků*. Deformační proces zahájíme přepnutím do deformačního módu a manipulujeme kontrolními body. Při změně kontrolních bodů lze s výhodou využít funkce *Freeze*, která znemožní další pohyb označených řídicích bodů. Pro bližší popis všech funkcí odkazují na manuál, který je přiložen spolu s aplikací.

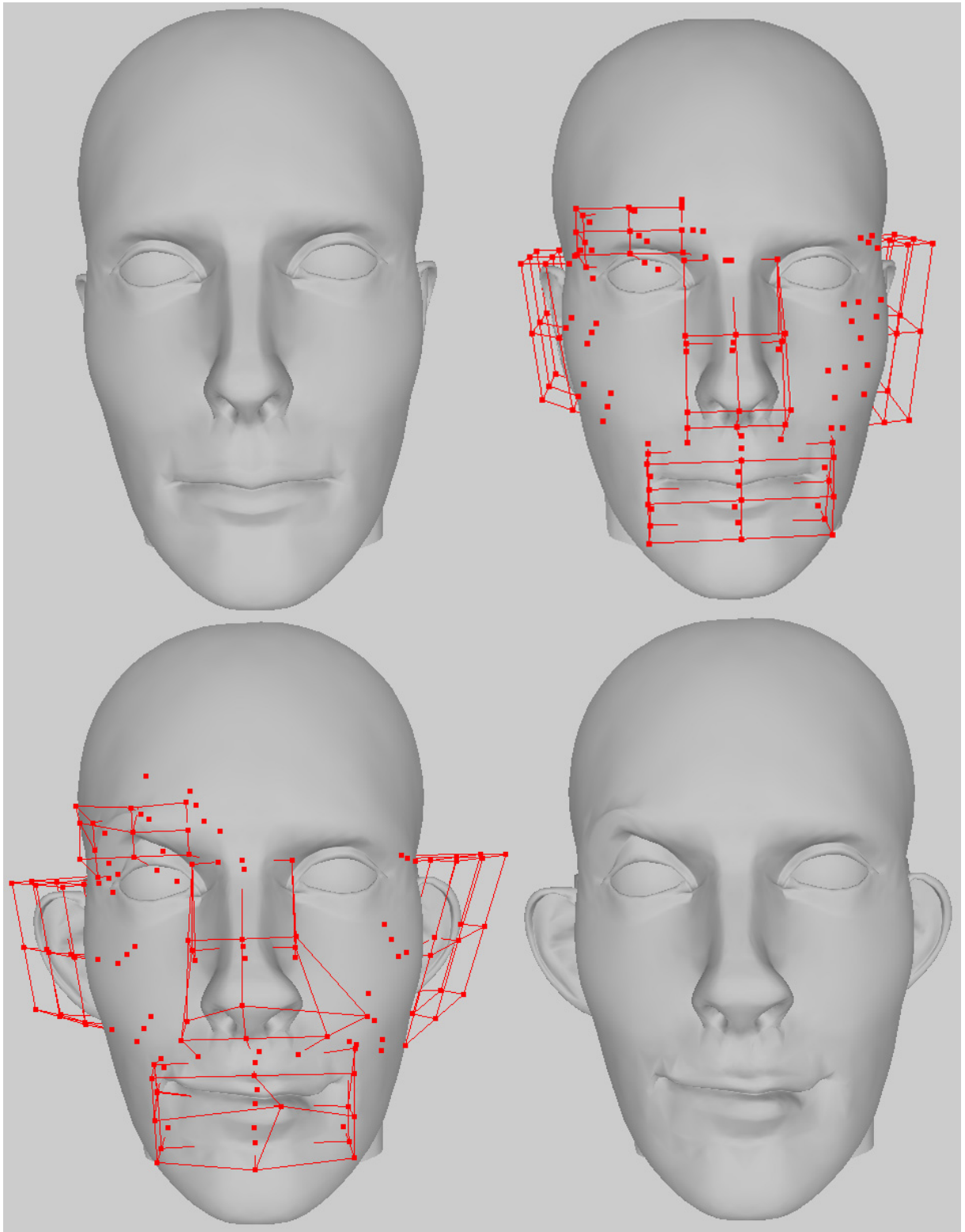
## 6 Závěr

Modelování pomocí lokálních deformací je efektivní metoda při vytváření geometrických těles. Ukázali jsme, že pomocí klasických *volných deformací* lze nahradit většinu známých globálních deformačních technik. Pouhou manipulací kontrolních bodů měníme povrch tělesa, což je velice snadná a intuitivní technika. *Rozšířené volné deformace* jsou komplexním modelovacím nástrojem pro editaci povrchu. Podle struktury prostorové mřížky lze měnit povrch tělesa do rozličných forem, záleží jen na fantazii designéra. Přestože je definování mřížky jistým způsobem omezené, s podpůrnými nástroji, jako je rozdělování elementů či spojování bodů, lze vytvářet struktury všech tvarů a uspokojit tak každého. Experimentální aplikace může být použita jako doplněk k modelovacím systémům, které nemají tyto deformační techniky implementovány. Stejně tak může sloužit jako demonstrační nástroj k výukovým účelům modelovacích technik.

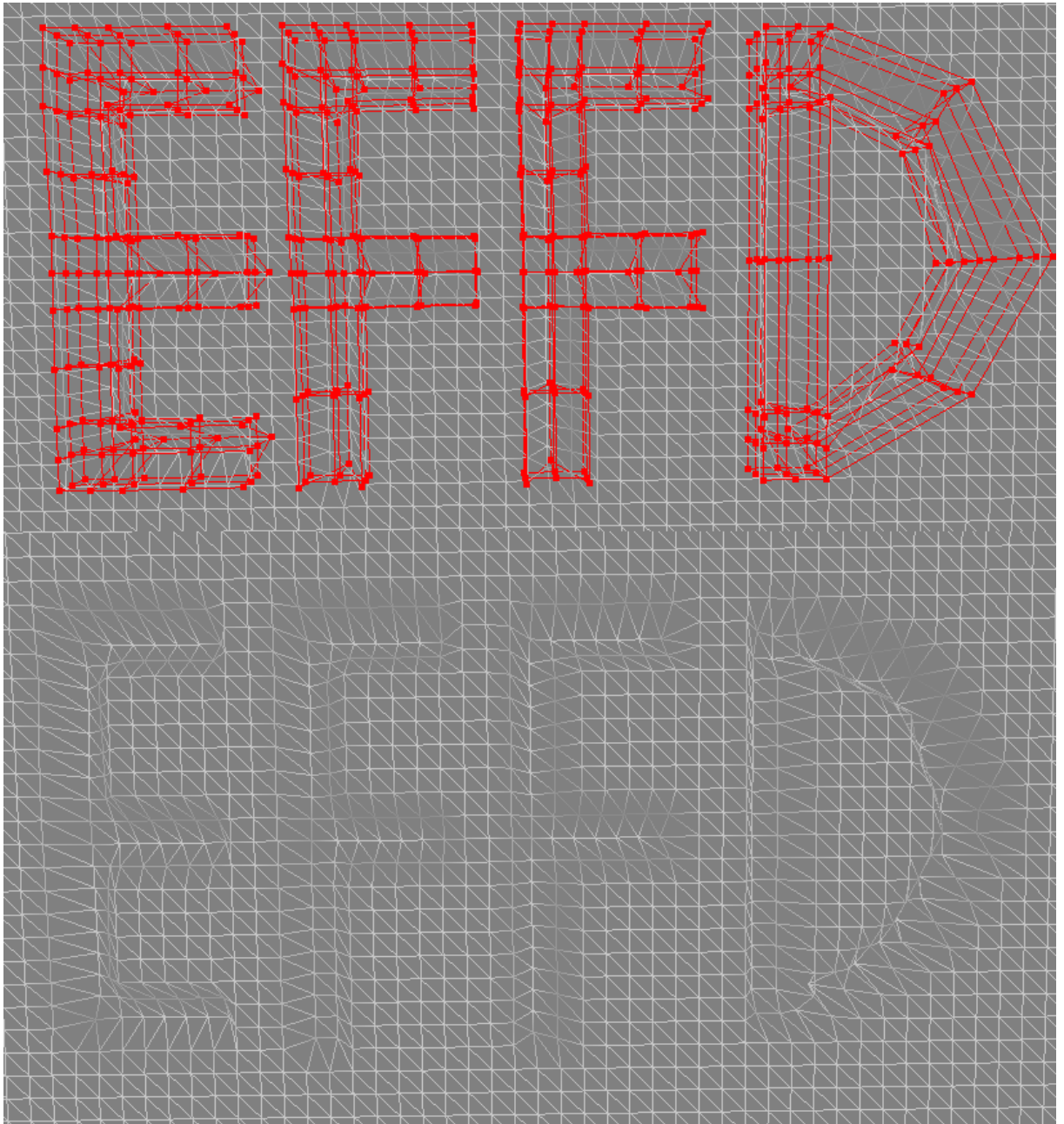
## 7 Galerie



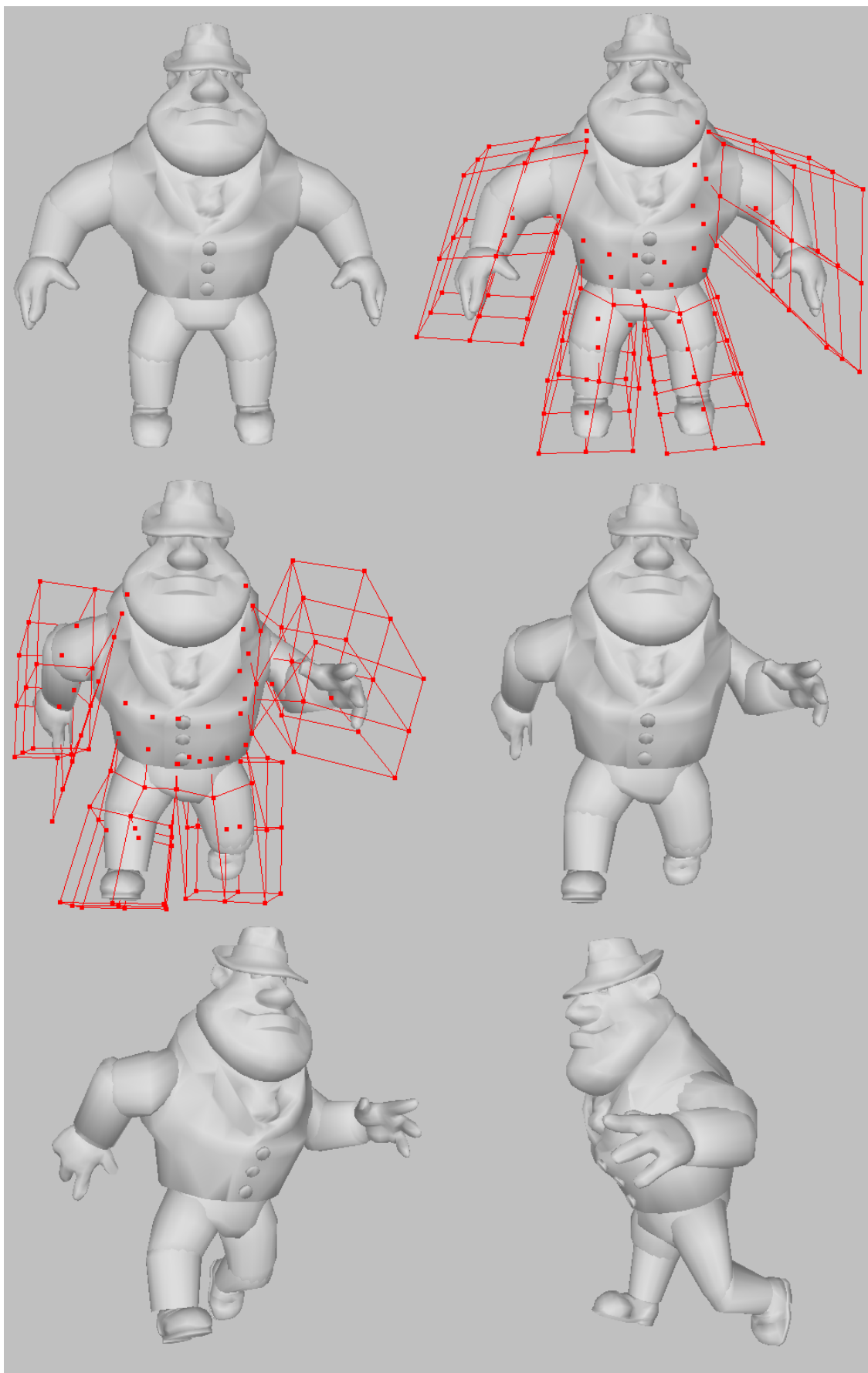
Obrázek 22: Deformace porsche.



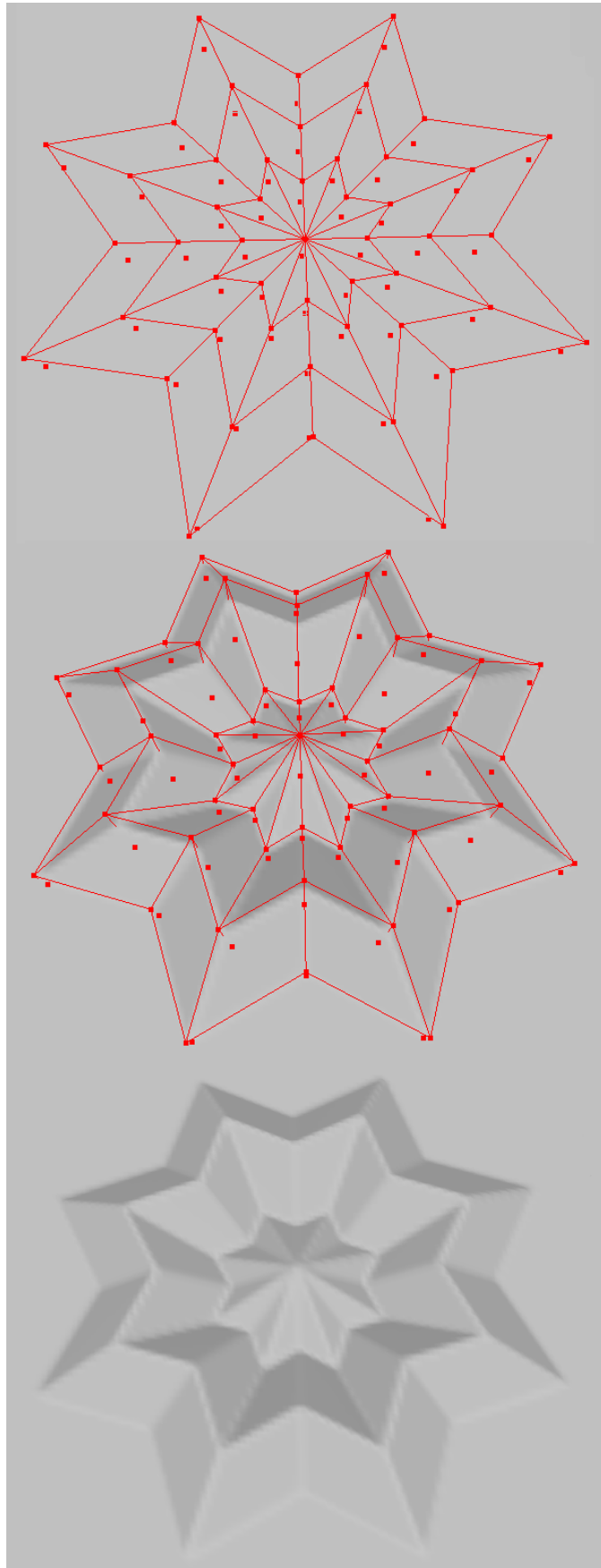
Obrázek 23: Mimika tváře.



Obrázek 24: Nápis v síti.



Obrázek 25: Běžící gangster.



Obrázek 26: Hvězda.

## 8 Seznam použité literatury

- [1] T. W. Sederberg and S. R. Parry. Free-Form Deformation of Solid Geometric Models. *SIGGRAPH 1986*, volume 20, pages 151-160.
- [2] S. Coquillart. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. *SIGGRAPH 1990*, volume 24, pages 187-196.
- [3] R. MacCracken and K. I. Joy. Free-Form Deformation With Lattices of Arbitrary Topology. *SIGGRAPH 1996*, pages 181-188.
- [4] Clint Chua and Ulrich Neumann. Hardware-Accelerated Free-Form Deformation, *SIGGRAPH 2000*.
- [5] A. H. Barr. Global and Local Deformation of Solid Primitives, *SIGGRAPH 1984*, pages 21-30.
- [6] C. Blanc and C. Shlick. Easy Transformations between Cartesian, Cylindrical and Spherical Coordinates, *LaBRI 1996*
- [7] J. Griessmair and W. Purgathofer. Deformation of solids with trivariate B-Splines, *Eurographics 1989*, pages 137-148
- [8] B. Hamann, D. Wu and R. J. Moorhead II. On particle path generation based on quadrilinear interpolation and Bernstein-Bézier polynomials. *IEEE Transactions on Visualization and Computer Graphics 1995*.
- [9] I. Horová. Numerické metody, *Přírodovědecká fakulta Masarykovy university, Brno 1999*, první vydání.
- [10] J. Žára, B. Beneš, J. Sochor, P. Felkel. Moderní počítačová grafika, *Computer Press, Brno, 2004*.



- [11] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. Numerical Recipes in C On-line Software Store, July 2005, <http://www.library.cornell.edu/nr/bookcpdf.html>.
- [12] J. Slovák. Skripta Geometrické algoritmy, *Přírodovědecká fakulta Masarykovy univerzity*, Brno, 2004.
- [13] J. O'Rourke. Computational Geometry in C, *Press Syndicate of the University of Cambridge*, 1998, <http://maven.smith.edu/~orourke/books/ftp.html>.
- [14] Real-time Soft-object Animation using Free-Form Deformation, *Gamasutra*, October 2004, [www.gamasutra.com/features/19990827/deformation\\_01.htm](http://www.gamasutra.com/features/19990827/deformation_01.htm).
- [15] Mathworld, 2005, <http://mathworld.wolfram.com>.
- [16] OpenGL, Programming Guide, *Addison-Wesley Publishing Company*, 2002.