Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Morphing 3D geometrických objektů

Plzeň, 2003 Jindřich Parus

Morphing of 3D geometrical objects

A morphing or metamorphosis is the process of continuously transforming one geometric object into another. This technique is used mainly in animation (special effects) and design. Our work aims at 3D geometric objects given in boundary representation. The goal of our work is to develop a method which works more or less automatically (i.e. there is no need to manually specify correspondence between the source and target objects) and which produces realistic animations.

Poděkování

Na tomto místě bych chtěl poděkovat vedoucí diplomové práce Doc. Dr. Ing. Ivaně Kolingerové za odbornou pomoc při zpracovávání této práce. Dále patří poděkování rodičům, příbuzným a přátelům, kteří mě po dobu studia podporovali.

Obsah

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne ..... ……..……

1. Úvod

Oblast počítačové animace je dynamicky se rozvíjející odvětví filmového průmyslu. V současné době je množství filmových scén kompletně tvořeno na počítači, skuteční herci často dodávají pouze své hlasy a v některých případech i své pohyby (v podobě motion capture dat). Takovéto scény by bylo často velmi složité nafilmovat, ať už z důvodů nedostupnosti prostředí (vesmírné scény apod.) nebo z důvodů ekonomických.

Jednou z klíčových vlastností počítačové animace je dynamika. Objekty se mohou různě pohybovat, vzájemně interagovat a v některých případech i měnit svůj tvar. Změna tvaru je přirozeným fenoménem i v přírodě – rostliny a živočichové během svého života rostou a mění tak svůj tvar, stromy se rozvětvují, květiny rozkvétají, živočichové nabírají sílu. Pokud chceme tento jev zachytit i v počítačové animaci, používáme animační techniku známou jako morphing.

Morphing samozřejmě nemá uplatnění pouze v animaci přírodních jevů (růstové simulace), ale používá se zejména pro tvorbu nejrůznějších speciálních efektů nebo v oblasti designu, kde lze kombinacemi existujících tvarů vytvářet tvary nové. Dále bezesporu v oblasti herního průmyslu, ale i například pro interpolaci mezi jednotlivými LOD[1] v různých GIS aplikacích.

1.1. Nástin problematiky

Úlohu morphingu lze formalizovat následovně. Máme dány dva zdrojové objekty – výchozí a cílový. Úkolem je nalézt transformaci mezi výchozím a cílovým objektem. Takových transformací existuje jistě velké množství, ale pro nás je důležitá ta třída transformací, která působí reálným dojmem. Zatímco například v úlohách triangulace jsou naprosto přesně definovány požadavky na výstupní trojúhelníkovou síť, pojem reálný dojem není žádná exaktní matematická definice. Je tedy třeba empiricky odvodit podmínky a omezení, která musí animace splňovat, aby přechod působil reálně. Intuitivně můžeme vysledovat například následující podmínky: pokud těleso neprotíná samo sebe, nemělo by k tomu docházet ani během přechodu; neměly by se objevovat díry v objektech, pokud tomu tak není v zdrojových objektech; poloha vrcholů by se měla měnit spojitě atd.

Práce se zabývá objekty danými hraniční reprezentací. Hraniční reprezentace je poměrně efektivním popisem geometrických objektů a navíc v ní pracuje většina grafických systémů. Na druhé straně je ovšem jen pouhou diskrétní aproximací reálného objektu a výpočet chybějících bodů je řešen interpolací. Spojitým popisem objektu může být například implicitní reprezentace, kde je ovšem komplikovanější modelování.

Cílem práce je metoda, fungující automaticky, případně s malým ručním zásahem. To je zásadní rozdíl od řešení morphingu v současných profesionálních animačních

[1] LOD – Level Of Detail (úroveň detailu).

nástrojích[2], kde je možné morphovat jen objekty velmi podobné a pokud tomu tak není, je třeba vyvinout značné úsilí v podobě manuální editace zdrojových těles.

Je zřejmé, že v hraniční reprezentaci je možné vyjádřit celou škálu objektů, nicméně dosud známé metody nepracují zcela univerzálně a často se omezují pouze na určitou třídu zdrojových objektů.

1.2. Rozvržení textu

Teoretická část práce se nachází v kapitolách 2 a 3. V kapitole 2 jsou definovány dále používané důležité pojmy, kapitola 3 teoreticky popisuje známé metody.

Na základě teoretického rozboru je v kapitole 4 podrobněji rozebrána vlastní metoda založená na článcích [1], [2] , [3] a [4]. V kapitole 5 je popsána implementace.

V 6. kapitole se nachází rozbor výsledků a v 7. kapitole závěr.

1.3. Matematické značení

$\vec{v}$ vektor, pokud není explicitně uvedeno jinak, jedná se o vektor třísložkový

x bod

$|\vec{v}|$ velikost vektoru

$|F|$ počet prvků množiny F

$\|\vec{v}\|$ normalizovaný vektor, tj. $\|\vec{v}\| = \frac{\vec{v}}{|\vec{v}|}$

$\vec{a} \times \vec{b}$ vektorový součin vektorů $\vec{a}$ a $\vec{b}$

$\vec{a}.\vec{b}$ skalární součin vektorů $\vec{a}$ a $\vec{b}$

[2] Více viz oddíl 3.1.

2. Důležité pojmy

Nechť nejsou všechny pojmy chápány jako exaktní matematické definice, ale hlavně jako vysvětlení termínu a uvedení do kontextu práce. Některé termíny jsou ponechány ve svém anglickém znění, neboť se jedná o ustálené pojmy a jejich překlad by byl matoucí, u některých termínů je uveden jejich anglický ekvivalent pro snadnější orientaci v literatuře.

Zdrojová tělesa – dvě vstupní tělesa mezi nimiž je prováděna transformace morphingu, je třeba odlišovat výchozí těleso (označované M_1) a cílové těleso (označované M_2).

Objekt – v kontextu práce se jedná o jakékoliv trojrozměrné těleso.

Ohraničený objekt – v objektu lze najít hraniční linii, tj. objekt je tvořen plátem.

Neohraničený objekt – objekt, který nemá žádnou hranici, tj. objekt je uzavřený.

Tvar objektu (shape) – množina bodů tvořící povrch objektu.

Model – model objektu, nebo také kompletní popis tvaru objektu. Lze formálně zapsat jako množinu $M=(V, F)$, kde $V=(v_1, v_2, \ldots, v_n)$ je množina vrcholů a $F=(f_1, f_2, \ldots, f_m)$ je množina trojúhelníků. Každý vrchol $v_i=(x, y, z)$ je dán uspořádanou trojicí souřadnic v Euklidovském prostoru R^3. Každý trojúhelník $f_j=(k, l, m)$ je dán trojicí indexů vrcholů, přičemž pro účely práce s trojúhelníky (zobrazování, rozdělování) je vhodné dodržovat konzistentní pořadí vrcholů, tj. buď ve směru hodinových ručiček nebo proti směru hodinových ručiček.

Topologie – relace mezi jednotlivými vrcholy, tj. pospojování jednotlivých vrcholů hranami.

Geometrie – konkrétní instance topologie s definovanými souřadnicemi vrcholů.

Mnohostěn (polyhedron) – model, jehož každá hrana je součástí sudého počtu polygonů.

Jednoduchý mnohostěn – počet vrcholů, hran a trojúhelníků vyhovuje Eulerově formuli $V-E+F=2$ (Eulerova formule pro jednoduchý mnohostěn). Je nutné poznamenat, že Eulerova formule je nutná, nikoli však postačující podmínka jednoduchého mnohostěnu. Jednoduchý mnohostěn musí splňovat ještě další podmínky:

- každá hrana spojuje právě dva vrcholy,
- každá hrana je sdílená právě dvěma trojúhelníky, tj. mnohostěn musí být uzavřený objekt,
- nejméně tři hrany vycházejí z jednoho vrcholu.

Genus 0 objekty – objekty vyhovující Eulerově formuli pro jednoduchý mnohostěn, navíc objekty nesmějí mít díru.

Supermesh – těleso, které je kombinací výchozího a cílového tělesa; často označováno, že „sdílí“ topologii výchozího a cílového tělesa.

Barycentrické souřadnice – vyjádření polohy bodu jako lineární kombinace vrcholů simplexu. V našem případě je simplexem trojúhelník, barycentrické souřadnice mají tedy tři složky. Pro vyjádření bodu x vzhledem k trojúhelníku daném vrcholy v_1, v_2, v_3 platí:

$$\begin{aligned} v_1 t + v_2 u + v_3 w &= x \\ t + u + w &= 1 \end{aligned} \qquad (2.1)$$

, kde *t, u, v* jsou barycentrické souřadnice. Navíc pro bod uvnitř trojúhelníku platí $t, u, w \leq 1$. Jsou-li například dvě složky barycentrických souřadnic nulové (a třetí je rovna jedné), leží bod *x* ve vrcholu; je-li pouze jedna složka nulová, pak bod leží na jedné z hran, atd.

Viditelnost – body AB jsou vzájemně viditelné, pokud úsečka AB (kde A, B jsou vnitřní body tělesa) se nachází kompletně uvnitř tělesa. Zúžením termínu je jasná (zřetelná) viditelnost, kde každý bod úsečky musí být uvnitř tělesa a navíc nesmí koincidovat s žádným bodem na hranici tělesa; vyjma případu, kdy by úsečka AB byla hranou tělesa.

Konvexní tělesa – konvexní tělesa mají tu vlastnost, že úsečka spojující libovolné dva vrcholy tělesa se nachází kompletně uvnitř tělesa.

Star-shaped tělesa – třída těles, ve kterých existuje alespoň jeden bod, z kterého je vidět do všech vrcholů tělesa. Podtřídou star-shaped těles jsou konvexní tělesa.

Star-point – vnitřní bod tělesa, ze kterého je vidět do všech ostatních vrcholů tělesa.

Jádro star-shaped tělesa (kernel) – množina všech star-point bodů.

Zobrazení – zobrazením rozumíme předpis:

$$f : A \rightarrow B \qquad (2.2)$$

, který každému bodu x z množiny *A* (tzv. vzoru) přiřazuje jeden bod y = f(x) z množiny *B* (tzv. obraz).

Vzorový prostor – prostor bodů x z množiny *A* (definiční obor).

Obrazový prostor – prostor bodů y z množiny *B* (obor hodnot).

Homeomorfismus – dva objekty se nazývají homeomorfní, pokud existuje vzájemně jednoznačné zobrazení mezi jednotlivými body povrchu těles.

Další pojmy jsou přímo vysvětleny v textu za účelem lepší čitelnosti.

3. Popis známých metod

Při studiu známých metod, bylo čerpáno především z výborného přehledu [5] a [2]. Článek [5] rovněž zahrnuje popis metod řešících morphing nejen v hraniční reprezentaci, ale i v reprezentaci objemové. Článek [2] se zabývá výhradně morphingem hraniční reprezentace, ale zde uvedené metody jsou podrobněji rozebrány, článek je také novějšího data než [5], takže lépe zachycuje aktuální stav v problematice morphingu. Dále byly prostudovány některé materiály ([6], [7], [8]) týkající se problematiky 2D morphingu polygonů. Ve 2D je situace o něco jednodušší, bohužel však nelze metody jednoduše zobecnit do 3D. Dříve než začne být podrobně rozebírána problematika morphingu, je vhodné pro srovnání nastínit současný stav této techniky v profesionálních animačních produktech typu 3ds max.

3.1. Morphing v profesionálních aplikacích

Morphing v 3ds max r3[3] klade na morphované objekty dosti omezující podmínky. Oba objekty musí mít totiž stejný počet vrcholů, přičemž jejich korespondence je dána apriori, tj. vrchol 0 modelu A koresponduje s vrcholem 0 modelu B, atd. Je-li známa korespondence, nezbývá než prostá interpolace mezi extrémními polohami vrcholů. V 3ds max r3 je navíc možné morphovat více objektů najednou, resp. míchat polohy vrcholů více těles. Morphing v této podobě se často používá pro animace mimiky. Práce pak vypadá zhruba takto. Je vymodelován nějaký základní výraz obličeje, tento model je zkopírován a v každé kopii modelu je možné nastavením jiných poloh vrcholů docílil různých výrazů tváře. Každý výraz je pak uložen do samostatného kanálu. Pro jednotlivé kanály lze pak nastavovat míru, s jakou budou obsaženy ve finální animaci. Tato situace je podobná i v produktu Lightwave 6[4].

Morphing geometricky a topologicky naprosto odlišných objektů není dost dobře možný. Přesto je možné morphing sledovat například ve filmovém průmyslu v různých speciálních efektech, je to však často výsledek velmi usilovné manuální práce.

3.2. Problém morphingu

Problém morphingu je často formalizován: jsou dána dvě zdrojová tělesa $M_1=(V_1, F_1)$ a $M_2=(V_2, F_2)$. Cílem je vygenerovat model $M(t)=(V, F)$, kde $t \in \langle 0; 1 \rangle$ tak, že tvar $M(0)$ je totožný s tvarem M_1 a podobně tvar $M(1)$ totožný s tvarem M_2. Model M musí být tedy kombinací obou zdrojových těles a to takovou, aby mohl reprezentovat jak tvar M_1, tak tvar M_2. Jednotlivé fáze morphingu jsou pak realizovány interpolací poloh vrcholů modelu M mezi jejich extrémními polohami. Model M bude v následujícím textu označován jako supermesh.

[3] 3ds max r3 je produktem firmy Discreet (www.discreet.com).
[4] Lightwave [7] je produktem firmy Newtek (www.newtek.com).

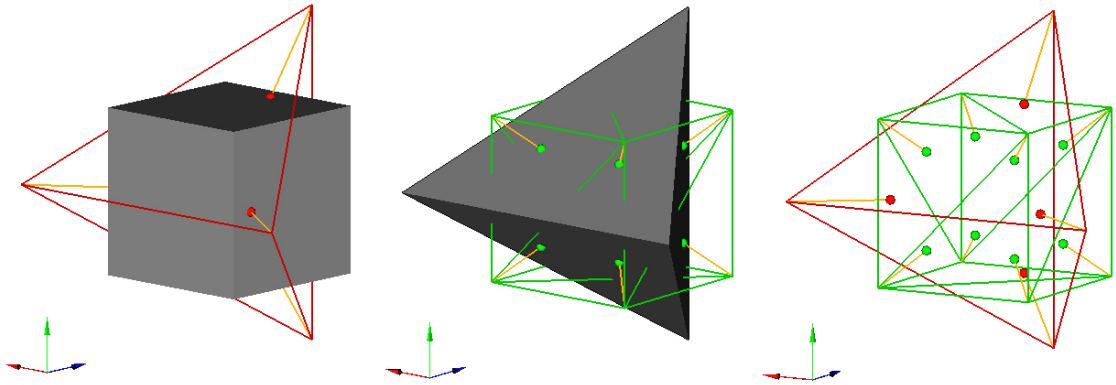
Výpočet morphingu bývá často rozdělován do tří základních kroků:

1. Nalezení korespondence mezi výchozím a cílovým objektem. Pro účely morphingu je nejvhodnější hledat korespondenci vrcholů, tj. kterému vrcholu z výchozího modelu odpovídá který vrchol z cílového modelu.
2. Vygenerování supermeshe. Na základě informací z předchozího kroku lze vygenerovat model, který „sdílí“ topologii obou zdrojových těles.
3. Interpolace poloh vrcholů supermeshe. Způsob interpolace a nalezení korektních trajektorií je sám o sobě dosti obsáhlý problém a v této práci bude uvažována pouze prostá lineární interpolace.

V následujícím textu jsou podrobněji popsány výše uvedené kroky.

3.3. Hledání korespondence

Výsledkem kroku hledání korespondence je určení mapování, kterému vrcholu z výchozího modelu odpovídá který vrchol z cílového modelu a obráceně. V obecném případě se vrchol jednoho modelu mapuje kamsi na povrch modelu druhého. Je tedy nutné určit, kam se vrchol mapuje (resp. na jaký trojúhelník), a určit jeho polohu. Geometrickou polohu mapovaného vrcholu lze výhodně vyjádřit pomocí barycentrických souřadnic vzhledem k vrcholům trojúhelníku na který se vrchol mapuje. Situace je vystižena na obrázku 3-1.



Obrázek 3-1: Mapování vrcholů modelu M_1 na povrch modelu M_2 a obráceně

Výsledkem hledání korespondence je tedy množina barycentrických souřadnic B_1, B_2, tj. udání relativní polohy V_1 vzhledem k F_2 a V_2 vzhledem k F_1, kde V_1, V_2 je množina vrcholů modelu M_1, M_2 a F_1, F_2 je množina trojúhelníků modelu M_1, M_2.

Otázkou zůstává, jak najít trojúhelník, na který se daný bod mapuje. Pravděpodobně by bylo možné trojúhelníky hledat protínáním paprsku vedeného bodem a nějakým středem promítání se všemi trojúhelníky modelu. Tento problém je však v úlohách morphingu typicky řešen jinak. Pro oba zdrojové modely je nalezena společná parametrická doména D a modely jsou do této domény zobrazeny. Pro toto zobrazení je vzorovým prostorem E^3 a obrazovým prostorem parametrická doména D.

Důležitou podmínkou je, že zobrazení musí být bijektivní, z čehož mimo jiné plyne, že se hrany v zobrazení nesmějí protínat.

Poznámka:

Výhodou vyjádření mapování bodu na trojúhelník pomocí barycentrických souřadnic je, že barycentrické souřadnice jsou invariantní vůči použitému zobrazení, tj. jsou-li vypočítány barycentrické souřadnice bodu vůči trojúhelníku v obrazovém prostoru, budou stejné i v prostoru vzorovém.

Pro objekty homeomorfické s koulí (vyhovují Eulerově formuli pro jednoduchý mnohostěn, tj. jedná se o jednoduché uzavřené mnohostěny typu 2-manifold) je přirozenou parametrickou doménou jednotková koule. Výsledkem zobrazení je množina vrcholů $S = \{s_1, s_2, \ldots s_n\}$, přičemž platí, že $s_i \in R^3$ a $|s_i|=1$. Množina S se nazývá parametrizací tělesa na jednotkovou kouli.

Alexa ve svém článku [2] uvádí jako parametrickou doménu jednotkový disk. Tato parametrická doména je vhodná pro jednoduché neuzavřené objekty (jednoduchý neuzavřený objekt nevyhovuje Eulerově formuli, neboť hraniční hrany jsou součástí pouze jednoho polygonu). Třída jednoduchých neuzavřených objektů je často označována jako topologické disky.

V následujícím textu budou popsány metody zobrazení pro objekty homeomorfické s koulí (oddíly 3.3.1. a 3.3.2.) a pro objekty homeorfické s diskem (oddíl 3.3.4.).

3.3.1. Star-shaped tělesa

Pokud se jedná o star-shaped tělesa, může nám pro zobrazení do společné parametrické domény posloužit prostá projekce bodů na jednotkovou kouli se středem promítání ve star-pointu tělesa. Středem promítání musí být star-point proto, aby bylo zobrazení bijektivní. Je-li navíc star-point v počátku souřadné soustavy[5], stačí pouze normalizovat všechny souřadnice vrcholů tělesa.

Jediným problémem je, jak nalézt adekvátní star-point. Star-point leží v jádře star-shaped mnohostěnu, je tedy třeba nalézt toto jádro. Jádro je průnikem všech poloprostorů tvořených ploškami tělesa. Jádro je konvexní útvar a je-li neprázdné, jakýkoliv jeho bod může být star-pointem.

3.3.2. Relaxace

Článek [1] a [2] navrhuje další metodu zobrazení. Zobrazení není omezeno pouze na star-shaped tělesa, ale dovoluje bijektivně zobrazit libovolný genus 0 mnohostěn. Alexa ([1], [2]) tuto techniku označuje jako spring embedding, což je technika používaná pro zobrazení 3D grafů takové, aby se hrany grafu neprotínaly. Základní myšlenka techniky spring embedding je následující. Graf je modelován jako soustava pružin (reprezentovaných hranami grafu) spojujících jednotlivé uzly grafu. Cílem je najít konfiguraci pružin a uzlů takovou, aby byla minimalizována celková energie systému pružin ([URL1]). Často je také tato technika označována jako relaxace. Toto označení budeme používat i v této práci.

Základní idea relaxace je následující: umístnit každý vrchol do středu jeho sousedních vrcholů. Sousedními vrcholy se zde rozumí vrcholy, které jsou s daným vrcholem spojeny hranou. Modifikace relaxace spočívá v tom, že po umístnění vrcholu do středu

[5] Není-li tomu tak, lze těleso samozřejmě transformovat, tak aby jeho star point ležel v počátku souřadného systému.

sousedních vrcholů je navíc poloha vrcholu normalizována, čímž se docílí toho, že vrchol leží na parametrické doméně (tj. na povrchu koule). V relaxačních úlohách je dále třeba identifikovat množinu vrcholů, které zůstanou pevné, tj. nebudou se umisťovat do středu svých sousedů. Bez těchto vrcholů by relaxace po dostatečném počtu iterací konvergovalala k triviálnímu řešení – všechny vrcholy koincidentní, což je samozřejmě pro naše potřeby nevhodné.

Postup zobrazení je pak následující. Je nalezen jakýkoli vnitřní bod tělesa a všechny vrcholy jsou promítnuty na kouli se středem promítání v nalezeném bodě.

Poznámka:

Pokud nebude těleso star-shaped a nalezený bod nebude star-pointem, pak v zobrazení jistě bude docházet k protínání některých hran. Cílem relaxace je posunout vrcholy, aby se hrany neprotínaly.

Relaxační algoritmus v každé iteraci umístní vrcholy, které nejsou pevné, do středu jejich sousedů. Středem se zde rozumí těžiště. Střed samozřejmě nemusí ležet přímo na kouli (parametrické doméně), vypočítaný střed je tedy dále normalizován. Matematický zápis relaxační metody je následující:

$$v_i^{k+1} = \frac{\frac{1}{|N_i|}\sum_{j \in N_i} v_j^k}{\left\| \frac{1}{|N_i|}\sum_{j \in N_i} v_j^k \right\|} \qquad (3.1)$$

, kde N_i je množina sousedních vrcholů k vrcholu v_i, $|N_i|$ je počet sousedních vrcholů a v_i^{k+1} je poloha vrcholu v_i v k-té iteraci relaxační metody.

Jediným problémem zůstávají chybějící pevné vrcholy, tj. vrcholy, které se nebudou centrovat na své sousední vrcholy. Bez dostatečného počtu správně vybraných (z hlediska polohy) pevných vrcholů (v literatuře označovány jako anchors – kotvy) se zobrazení zhroutí. Zhroucením se zde rozumí fakt, že zobrazení nepokrývá celou kouli, tj. existuje hemisféra taková, kde se nenachází žádný z vrcholů s_i. Příklad zhroucení zobrazení je uveden na obrázku 3-2 b). V relaxačních úlohách v rovině obvykle stačí zafixovat jeden trojúhelník (tři vrcholy) a tím je zaručeno, že se zobrazení nezhroutí. Zafixováním třech vrcholů v prostoru se však docílí toho, že všechny vrcholy leží uvnitř trojúhelníku tvořeného zafixovanými vrcholy (viz obrázek 3-2 a)). Je tedy nutné zafixovat nejméně čtyři vrcholy a to navíc tak, aby každá hemisféra obsahovala nejméně jeden pevný bod. Vhodným rozmístněním pevných vrcholů na kouli jsou vrcholy pravidelného tetrahedronu, který má opsanou jednotkovou kouli (parametrickou doménu).

Poznámka:

Ve skutečnosti je třeba najít vrcholy modelu takové, které jsou vrcholům vepsaného tetrahedronu nejblíže.

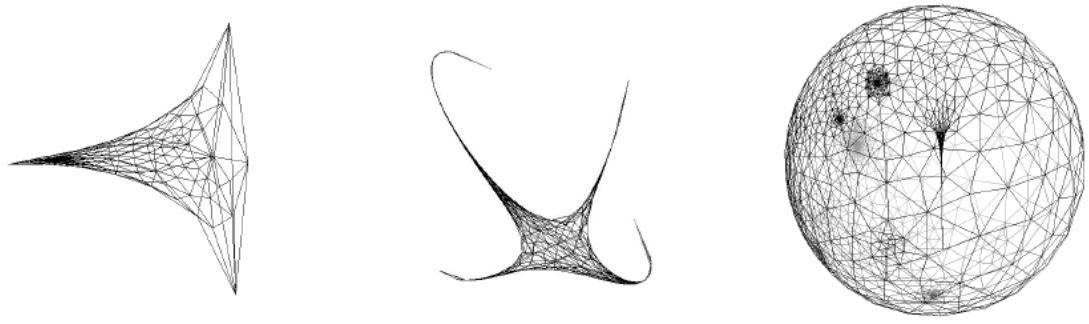
Přesto však může docházet ke zhroucení (viz obrázek 3-2 b)), což je v [1] řešeno pomocí heuristiky. Relaxace je tedy citlivá na počáteční umístnění pevných vrcholů, eventuální zhroucení zobrazení je třeba identifikovat a spustit znovu relaxační proces s jiným počátečním nastavením.

Každý zafixovaný vrchol však může eventuelně působit další problémy. Jak již bylo uvedeno, zafixované vrcholy nejsou během relaxace umístňovány do středu vrcholů sousedů, což může způsobit, že v zobrazení se budou hrany protínat (viz obrázek 3-2 c)). Tento problém je poměrně elegantně řešen v [1] následovně. Nejdříve jsou výše popsaným způsobem vybrány pevné vrcholy. Po určitém počtu iterací relaxace bude zobrazení ve stavu, že vrcholy daleko od pevných vrcholů budou všechny ideálně umístněny. Ideálním umístněním se zde rozumí fakt, že vrcholy se nacházejí ve středu svých sousedů.

Poznámka:

Pokud si opět představíme model jako systém pružin, pak ideálním umístněním vrcholu se rozumí, že vrchol dosáhl stavu, kdy je energie pružin ukotvených v daném vrcholu minimální.

Čím blíže je vrchol k nějakému pevnému vrcholu, tím více je vzdálen od své ideální polohy. Po jistém počtu iterací je tedy možné zafixovat jiné vrcholy a odstranit tak možné protínání hran v místech původně zafixovaných vrcholů. Nejvhodnější oblastí pro umístnění nových pevných vrcholů je oblast, kde jsou vrcholy již ve svých ideálních polohách a není je třeba dále relaxovat. Vhodným kandidátem na takovou oblast je oblast protilehlá k původním pevným vrcholům – to je právě oblast nejvíce vzdálená od původního pevného vrcholu, v níž jsou všechny vrcholy ve svých ideálních polohách. Na následujícím obrázku jsou shrnuty problémy spojené s relaxací.



a) **b)** **c)**

Obrázek 3-2: Problémy zobrazení pomocí relaxace a) Zhroucené zobrazení se třemi zafixovanými vrcholy b) Zhroucené zobrazení se čtyřmi zafixovanými vrcholy umístněnými ve vrcholech pravidelného tetrahedronu c) Zobrazení, kde dochází k protínání hran (obrázek převzat ze [1])

Dále je vhodné zvážit zastavovací podmínku relaxace, není totiž možné nastavit pevný počet iterací a doufat, že bude postačující pro různé modely. Vhodné je sledovat maximální posun vrcholu v aktuální iteraci a pokud bude menší než zadané ε, proces relaxace zastavit. Matematicky zapsáno:

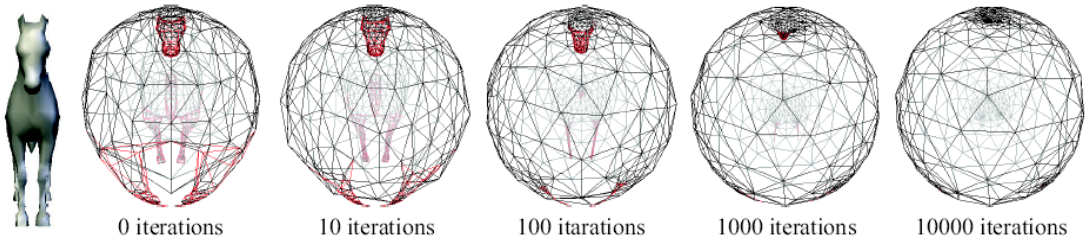
$$\max\left|v_i^k - v_i^{k+1}\right| < \varepsilon \qquad (3.2)$$

, kde v_i^k , v_i^{k+1} jsou polohy bodů v_i v k-té, resp. k+1-té iteraci a ε maximální rozdíl poloh vrcholů mezi k-tou a k+1-ní iterací.

Sledování ε není však postačující, podstatné je, aby zobrazené trojúhelníky měly stejnou orientaci jako trojúhelníky modelu, tj.:

$$\mathrm{sgn}((v_k \times v_l).v_m) = \mathrm{sgn}((s_k \times s_l).s_m) \qquad (3.3)$$

, kde sgn je znaménková funkce; v_k, v_l, v_m jsou vrcholy trojúhelníku a s_k, s_l, s_m jejich zobrazení (na parametrickou doménu). Toto ovšem vyžaduje dodržovat v modelu konzistentní orientaci vrcholů trojúhelníků (buď ve směru hodinových ručiček nebo proti směru hodinových ručiček). Na obrázku 3-3 je znázorněn postup relaxace po určitém počtu iterací. Trojúhelníky, které nemají správnou orientaci, jsou znázorněny červeně.



Obrázek 3-3: Relaxace modelu koně; červeně jsou znázorněny trojúhelníky, které nemají správnou orientaci (převzato ze [2])

3.3.3. Harmonické mapy

V článku [4] je pro zobrazení použito harmonických map (harmonical map). Na rozdíl od předchozí metody není společnou parametrickou doménou jednotková koule nýbrž jednotkový disk. Ačkoli se jednotkové disky hodí zejména pro neuzavřené objekty, mohou být použity i pro objekty uzavřené s tím, že je třeba specifikovat nějakou hraniční smyčku (viz dále). Zobrazení h lze zapsat:

$$h : M \rightarrow H \qquad (3.4)$$

, kde M je vzorový prostor v R^3 a H je obrazový prostor v R^2. Zobrazení je bijektivní a navíc minimalizuje metrický rozptyl. Metoda zobrazení funguje následujícím způsobem.

Je nalezeno n vrcholů, které na modelu tvoří uzavřenou smyčku. V zobrazení jsou tyto vrcholy umístněny na hranici jednotkového disku, přičemž jednotkový disk má střed v počátku souřadné soustavy. Zbylé vrcholy jsou zobrazeny tak, aby byla minimalizována celková energie E_{harm}. Podobně jako v případě relaxační techniky[6] je opět objekt modelován jako soustava pružin. E_{harm} tedy může být vyjádřena jako suma energie soustavy pružin, přičemž pružiny jsou umístněny podél jednotlivých hran. Celková energie E_{harm} je pak vyjádřena takto:

$$E_{harm} = \frac{1}{2} \sum_{\{i,j\} \in Edges(H)} k_{i,j} \left| v_i - v_j \right|^2 \qquad (3.5)$$

, kde i, j jsou indexy vrcholů, v_i, v_j jejich polohy v zobrazení a $k_{i,j}$ je materiálová konstanta pružiny (resp. hrany) dané vrcholy i, j. Materiálová konstanta je stanovena následujícím způsobem. Nechť $l_{i,j}$ je délka hrany mezi vrcholy i,j měřená ve vzorovém prostoru M, $A_{i,j,k}$ je plocha trojúhelníka daného vrcholy i,j,k. Pak pro každou vnitřní hranu (i, j), s kterou incidují dva trojúhelníky (i, j, k_1) a (i, j, k_2):

[6] Viz oddíl 3.3.2.

$$k_{i,j} = \frac{l_{i,k1}^2 + l_{j,k1}^2 - l_{i,j}^2}{A_{i,j,k1}} + \frac{l_{i,k2}^2 + l_{j,k2}^2 - l_{i,j}^2}{A_{i,j,k2}} \qquad (3.6)$$

Jediné řešení, které minimalizuje E_{harm}, je řešení soustavy lineárních rovnic $\nabla E_{harm} = 0$, kde ∇E_{harm} je gradient E_{harm}. Pro výpočet gradientu je třeba uspořádat neznámé (polohy vrcholů v obrazovém prostoru H) do $2N$ dimenzionálního vektoru.

$$V = (v_{1x}, v_{1y}, v_{2x}, v_{2y}, ..., v_{Nx}, v_{Ny}) \qquad (3.7)$$

, kde N je počet vrcholů. E_{harm} je navíc kvadratické forma, lze tedy psát $E_{harm}=V^T HV$. Gradient lze pak vyjádřit jako $\nabla E_{harm}=\partial E_{harm}/\partial V$.

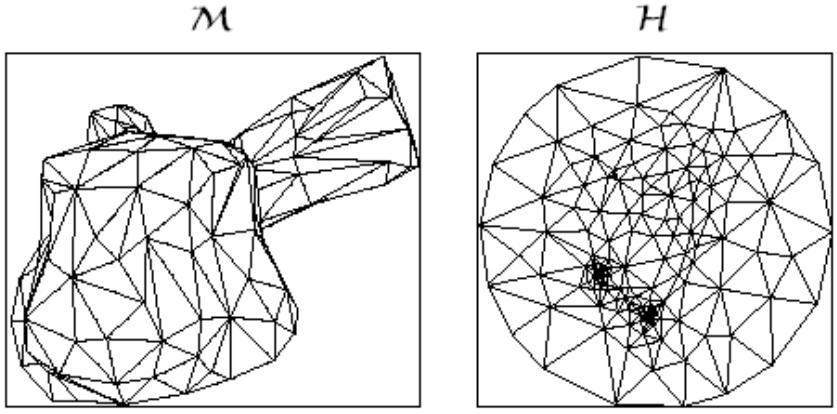
Vrcholy na hranici jednotkového disku jsou zafixované (jejich poloha se nemění). Vektor V je tedy možné rozdělit na dvě části, část proměnných V^α a pevnou část V^β. Pak je třeba také rozdělit matici koeficientů H.

$$E_{harm} = \begin{bmatrix} V^{\alpha T} V^{\beta T} \end{bmatrix} \begin{bmatrix} H^{\alpha\alpha} & H^{\alpha\beta} \\ H^{\beta\alpha} & H^{\beta\beta} \end{bmatrix} \begin{bmatrix} V^\alpha \\ V^\beta \end{bmatrix} \qquad (3.8)$$

Pro pevnou část je funkce konstantní, stačí tedy pouze vyřešit soustavu pro část vektoru neznámých V^α.

$$\nabla E_{harm} = \frac{\partial E_{harm}}{\partial V^\alpha} = 2H^{\alpha\alpha} V^\alpha + 2H^{\alpha\beta} V^\beta = 0 \qquad (3.9)$$

Obrázek 3-4 ukazuje zobrazení modelu M na jednotkový disk (převzato z [4]).



Obrázek 3-4: Zobrazení modelu hlavy zajíce do jednotkového disku. Jako hraniční smyčka byla použita sekvence hran okolo krku zajíce (převzato z [4]).

3.3.4. Zobrazení topologických disků

Pro zobrazení topologických disků je třeba nalézt bijektivní zobrazení ohraničeného objektu do roviny. Podobně jako v předchozím případě (harmonické mapy) je nutné identifikovat množinu vrcholů, které budou pevné, a množinu volných vrcholů, jejichž poloha bude určena vzhledem k pevným. V případě ohraničených objektů jsou přirozenými kandidáty na pevné vrcholy vrcholy hraniční (tj. vrcholy, které tvoří okrajové hrany). Volné vrcholy jsou pak všechny ostatní.

Poznámka:

Alexa ([2]) uvádí, že přirozenou parametrickou doménou pro neohraničené objekty je koule a pro ohraničené objekty jednotkový disk. Naproti tomu však v předchozím oddílu byl popsán způsob zobrazení uzavřeného objektu na topologický disk. Jediný rozdíl je ve výběru pevných vrcholů. V případě neuzavřených objektů jsou pevnými vrcholy hraniční vrcholy, v případě uzavřených objektů je třeba pevné vrcholy specifikovat nalezením uzavřené smyčky hran.

V prvním kroku jsou tři pevné vrcholy zafixovány na okraji jednotkového disku tak, že tvoří rovnostranný trojúhelník. Zbylé vrcholy jsou umístněny na okraj jednotkového disku tak, že délka oblouků je úměrná skutečné délce hran zobrazovaného modelu. Zbylé vrcholy (tj. vnitřní) jsou volné a jejich poloha je určena vzhledem k zafixovaným. Nechť tedy množina vrcholů zobrazovaného modelu je $V = \{v_i\}, i \in <0, M-1>$, kde M je počet vrcholů. Množina vnitřních volných vrcholů je $V^{\alpha} = \{v_j\}, j \in <0, N-1>$, kde N je počet vnitřních volných vrcholů a množina pevných vrcholů je $V^{\beta} = \{v_k\}, k \in <N, M-1>$. Úkolem je tedy najít zobrazení vrcholů z množiny V^{α}, přičemž pro vrcholy z množiny V^{β} platí, že $|v_k| = 1$, kde $k \in <N, M-1>$.

V článku [2] je popsáno tzv. barycentrické zobrazení. Základní idea je umístnit vrchol do středu svých sousedů. Podobnou ideu navrhuje stejný autor v [1] s tím rozdílem, že v [1] je toto řešení je založeno na iteračním výpočtu s užitím relaxace. V tomto případě se jedná o řešení soustavy lineárních rovnic. Polohu volného vrcholu určíme ze vztahu:

$$w_j = \frac{1}{|N_j|} \sum_{l \in N_j} w_l \qquad (3.10)$$

, w_j je poloha j-tého vrcholu po zobrazení na jednotkový disk, $|N_j|$ je počet sousedních vrcholů k vrchol w_j. Zapíšeme-li hodnoty $\frac{1}{|N_j|}$ do matice Λ tak, že:

$$\lambda_{m,n} = \begin{cases} |N_m|^{-1} & \{m,n\} \in E \\ 0 & \{m,n\} \notin E \end{cases} \qquad (3.11)$$

, kde zápis $\{m, n\} \in E$, resp. $\{m, n\} \notin E$, znamená, zda existuje, resp. neexistuje, hrana mezi vrcholy s indexy m, n. Nyní je možné zapsat soustavu lineárních rovnic pro neznámé w_j:

$$(I - \Lambda) \begin{pmatrix} w_0 \\ w_1 \\ \dots \\ w_{M-1} \end{pmatrix} = \begin{pmatrix} \sum_{k=N}^{M-1} \lambda_{0,k} w_k \\ \sum_{k=N}^{M-1} \lambda_{1,k} w_k \\ \dots \\ \sum_{k=N}^{M-1} \lambda_{N-1,k} w_k \end{pmatrix} \qquad (3.12)$$

, kde I je jednotková matice. V [2] je uvedeno, že tato rovnice má jediné řešení a pro řešení lze využít faktu, že Λ je řídká.

Alexa ([2]) dále uvádí hlavní problém zobrazení modelů do parametrické domény topologický disk. Tím je fakt, že vnitřní trojúhelníky (tj. trojúhelníky tvořené vnitřními vrcholy) mají mnohem menší plochu, než hraniční trojúhelníky (tj. trojúhelníky, jejichž některé vrcholy jsou hraničními vrcholy). Tímto může vznikat problém kvůli limitované přesnosti reprezentace reálných čísel v počítači. Empiricky lze odvodit, že pro vyrovnané poměry velikostí vnitřních a vnějších trojúhelníků je vhodné, aby objekt měl větší počet pokud možno rovnoměrně rozložených vrcholů.

3.3.5. Další zobrazení

Článek [3] udává další metody zobrazení, které jsou ovšem použitelné pouze pro určitou třídu objektů (objekty vytvořené tažením profilu po nějaké trajektorii a objekty vytvořené rotací profilu okolo nějaké osy). Dále navrhuje tzv. simulační metody, založené na fyzikálních principech. Základní představa je taková, že uvnitř objektu je plyn, který se rozpíná a tím deformuje objekt do tvaru koule. Nicméně autor uvádí, že žádná simulační metoda není dostatečně obecná pro zobrazení libovolného genus 0 mnohostěnu.

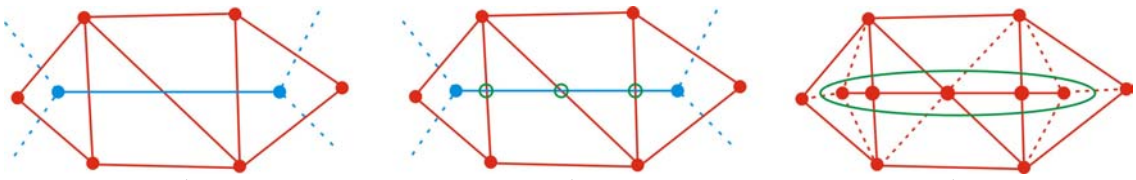
3.4. Generování supermeshe

Jak již bylo výše uvedeno, supermesh je model, který je kombinací obou zdrojových těles. Kombinací v tom smyslu, že má topologii jak výchozího modelu, tak modelu cílového – často se označuje jako „sdílená“ topologie (shared topology). V předchozím kroku bylo nalezeno mapování vrcholů modelu M_1 na povrch modelu M_2 (množina barycentrických souřadnic B_1) a obráceně (B_2). Rovněž bylo stanoveno zobrazení, které promítá objekt do parametrické domény (v případě genus 0 mnohostěnů se jedná o jednotkovou kouli).

V kroku generování supermeshe jsou zobrazení obou zdrojových těles sloučena. Pokud chápeme zobrazení modelu jako graf, pak se sloučením rozumí výpočet překrytí dvou grafů (v literatuře označováno jako graph overlay). Ze sloučení grafů je možné zpětně odvodit tvar a topologii supermeshe. Klíčovým pojmem při generování supermeshe je tzv. „sdílená topologie“.

3.4.1. Sdílená topologie

Sdílená topologie označuje topologii, která je výsledkem sloučení dvou existujících (zdrojových) topologií. Nejlépe situaci demonstruje následující obrázek (pro jednoduchost ve 2D):



Obrázek 3-5: Postup výpočtu sdílené topologie

Na obrázku 3-5 a) je červeně znázorněn model M_1 a modře model M_2. Pro jednoduchost je modelem M_2 pouze čára (čárkovaně je naznačeno další pokračování trojúhelníkové sítě), aby bylo jasné postupu dosažení sdílené topologie. Pro dosažení sdílené topologie je třeba „vnutit" topologii modelu M_2 do topologie modelu M_1[7]. Na obrázku 3-5 b) jsou zeleně znázorněny průsečíky hran modelu M_1 a modelu M_2, které je nutné nalézt pro výpočet sdílené topologie. Na obrázku 3-5 c) jsou oba modely již sloučené – tedy supermesh M se sdílenou topologií (zeleně je označena hrana modelu M_1, která byla „vnucena" do M). Na obrázku je také vidět, že vznikly ještě další hrany (vyznačeny přerušovanou čarou). Tyto hrany je nutné vložit, aby výsledný supermesh byl triangularizován. Postup triangulace je detailněji popsán v oddílu 3.4.6. Triangulace supermeshe.

Jak je z obrázku vidět, supermesh M obsahuje jak vrcholy modelu M_1, tak vrcholy modelu M_2, ale ještě navíc vrcholy vzniklé z průsečíků hran modelů M_1 a M_2. Formálně lze tedy zapsat:

$$V = V_1 \cup V_2 \cup I \qquad (3.13)$$

, kde I je množina průsečíků a V_1, V_2 jsou množiny vrcholů modelů M_1, M_2. V následujícím textu se budeme zabývat otázkou, jak nalézt průsečíky a jak správně rozdělit hrany trojúhelníků.

3.4.2. Experimentální ověření významu sdílené topologie

Před popisem samotného výpočtu sdílené topologie bude ještě zmíněn jeden postup, který je jakýmsi experimentálním ověřením toho, že pro docílení sdílené topologie je třeba skutečně vypočítat průsečíky hran modelu M_1 s hranami modelu M_2. Jedná se o postup, který byl použit v prvopočátcích diplomové práce, kdy dosud nebyl zřejmý význam zobrazení objektů do společné parametrické domény a výpočet sdílené topologie. Samozřejmě to byl postup chybný, nicméně je vhodné diskutovat jeho výsledky.

Postup se opět omezoval pouze na star-shaped tělesa a základní idea byla následující. Ze star-pointu byl veden paprsek vrcholem modelu M_1 a na místo, kde tento paprsek protnul povrch modelu M_2, byl vložen nový vrchol, resp. byl rozdělen protnutý trojúhelník. Stejný postup byl aplikován i obráceně. Výsledná přerozdělená tělesa měla tedy stejný počet vrcholů. Bohužel až na základě výsledných animačních sekvencí byla zjištěna zásadní chyba této metody. Obrázek 3-6 ukazuje jeden z výstupů.

Obrázek 3-6: Animace morphingu v prvopočátcích diplomové práce

Z obrázku 3-6 je vidět, jak se výchozí objekt (krychle) mění na cílový objekt (koule). Během animace je však možné pozorovat, jak některé hrany zůstávají „zařízntuty"

[7] Samozřejmě je to možné dělat i obráceně, tj. vnucovat topologii M_1 do M_2.

v objektu, nejmarkantněji je toto vidět na posledním snímku animace. Předchozím postupem byla sice zjištěna korespondence vrcholů, vrcholy se i pohybují po správných trajektoriích, ale jejich pospojování hranami je jednoznačně chybné. Potíž je právě v tom, že těleso použité pro morphing nemá sdílenou topologii.

3.4.3. Výpočet sdílené topologie

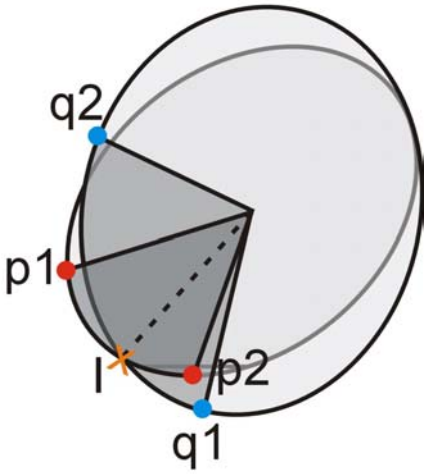
Z oddílu 3.4.1. je zřejmé, že pro sloučení dvou zobrazení je třeba nalézt průsečíky hran. Důležité je, že se nejedná o skutečné průsečíky, neboť hrany zdrojových těles jsou v obecném případě mimoběžné. Po zobrazení obou zdrojových těles do společné parametrické domény však již hrany mimoběžné nejsou a protínat se mohou. Při zobrazení byly dosud brány v úvahu pouze koncové body hran modelu, nicméně geometrické objekty dané hraniční reprezentací jsou invariantní vůči používaným druhům zobrazení, zobrazení vrcholů modelu je tedy postačující pro zobrazení celého modelu. Po zobrazení na jednotkovou kouli dosud lineární hrany přejdou v oblouky.

Jak již bylo uvedeno výše, průsečíky hran zdrojových modelů M_1 a M_2 jsou důležité pro dosažení sdílené topologie. Ovšem v obecném případě jsou hrany těles mimoběžné a ve skutečnosti se vůbec neprotínají. Hledání průsečíků se tedy řeší v zobrazení zdrojových těles do společné parametrické domény. Jako společná parametrická doména se často používá koule a zobrazení je pak projekcí na kouli. Po projekci na kouli se však hrany promítnou do oblouků[8], a je pak třeba řešit průsečíky oblouků. Naštěstí není třeba se uchylovat k řešení komplikovaných kvadratických rovnic, neboť oblouky jsou části kružnic, které mají střed ve středu parametrické domény (tj. jsou částmi hlavních kružnic), což situaci zjednodušuje (alespoň v tom smyslu, že není třeba řešit soustavu kvadratických rovnic, které mají obecně čtyři řešení). Výpočet průsečíků dvou oblouků je podrobněji popsán v následujícím oddílu. Dalším problémem je test polohy bodu v trojúhelníku, který je třeba již v předchozím kroku, kde se hledá mapování bodů modelu M_1 na povrch modelu M_2. Trojúhelník, podobně jako hrana, se v zobrazení stává trojúhelníkem sférickým. Tento test bude podrobněji rozebrán v oddílu Test polohy bodu ve sférickém trojúhelníku.

Průsečík dvou oblouků

Situaci demonstruje obrázek 3-7. Body p_1, p_2 jsou koncovými body oblouku p; body q_1, q_2 jsou koncovými body oblouku q. Bod I je pak průsečík oblouků p a q.

[8] Zobrazení hrany do parametrické domény lze také definovat jako nejkratší cestu na povrchu koule mezi dvěma koncovými body hrany.



Obrázek 3-7: Průsečík oblouků, které jsou částmi hlavních kružnic

Jak již bylo uvedeno výše, oblouky jsou částmi kružnic, které mají střed ve středu parametrické domény. Stačí tedy vypočítat průsečíky těchto kružnic a pak pouze určit, zda nalezené průsečíky jsou společným bodem obou oblouků. Koncové body oblouku spolu se středem domény tvoří rovinu. Z obrázku je vidět, že průsečík dvou kružnic leží na přímce, která je průnikem rovin definovaných oblouky. Směrový vektor této přímky určíme jednoduše ze vztahu:

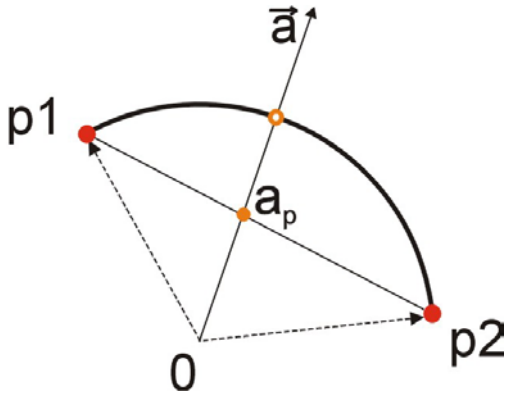
$$\vec{r} = \pm(\vec{p}_1 \times \vec{p}_2) \times (\vec{q}_1 \times \vec{q}_2) \qquad (3.14)$$

, kde $\vec{r}$ je směrový vektor přímky a vektory $\vec{p}_1, \vec{p}_2$, resp. $\vec{q}_1, \vec{q}_2$ jsou vektory dané star-pointem modelu (resp. nějakým vnitřním bodem jako je tomu v případě relaxace) a odpovídajícími koncovými body oblouků.
Kružnice se protínají právě ve dvou bodech nebo koincidují, z tohoto důvodu je třeba brát i opačný směrový vektor $-\vec{r}$. Dále je možné využít faktu, že parametrickou doménou je jednotková koule. Normalizujeme-li tedy směrový vektor $\vec{r}$, dostaneme přímo průsečík kružnic. Zbývá tedy určit, zda nalezený průsečík je společným bodem obou oblouků. Situace může totiž být i taková, že kružnice se protínají, nikoliv však oblouky. Matematicky lze podmínku formulovat následovně:

$$(\|\vec{r}\| \in p \wedge \|\vec{r}\| \in q) \vee (-\|\vec{r}\| \in p \wedge -\|\vec{r}\| \in q) \qquad (3.15)$$

, kde $\|\vec{r}\|$ je průsečík kružnic a *p, q* jsou oblouky.

Otázku, zda se nalezený průsečík nachází na oblouku, lze odpovědět následujícím testem. Existuje-li průsečík paprsku $\vec{a}$ a úsečky p_1p_2 a zároveň průsečík paprsku $\vec{a}$ úsečky q_1,q_2, pak je průsečík $\|\vec{r}\|$, resp. $-\|\vec{r}\|$ skutečným průsečíkem oblouků p_1p_2 a q_1q_2. Situaci lépe demonstruje obrázek 3-8, kde a_p je průsečík paprsku $\vec{a}$ a úsečky p_1p_2 (a také projekce průsečíku paprsku $\vec{a}$ a oblouku p_1p_2 na úsečku p_1p_2).



Obrázek 3-8: Průsečík oblouku s paprskem

Z obrázku je také vidět, že oblouky p_1p_2, resp. q_1,q_2 je možné v tomto případě aproximovat úsečkami p_1p_2 a q_1q_2 bez jakékoliv újmy na přesnosti (existuje-li průsečík paprsku a oblouku, pak také existuje průsečík paprsku a úsečky aproximující oblouk). Z předchozích úvah plynou následující dvě soustavy lineárních rovnic:

$$t_p\vec{r} = p_1 + s_p(p_2 - p_1)$$
$$t_q\vec{r} = q_1 + s_q(q_2 - q_1) \qquad (3.16)$$

, kde t_p, t_q jsou parametry paprsků; s_p, s_q je parametrické vyjádření průsečíku vzhledem k oblouku p, resp. q. Aby průsečík existoval, musí platit: $s_p, s_q \in <0;1>$ a $t_p, t_q > 0$.

Poznámka:

Soustavy rovnic jsou přeurčené, máme tři rovnice pro dvě neznámé. Je tedy možné jednu rovnici vyškrtnout, ovšem s tím, že je třeba dát pozor na to, aby zbylé dvě rovnice nebyly lineárně závislé[9].

Je nutné mít na zřeteli možné singulární případy, které mohou nastat, a které je třeba identifikovat a správně v dalším zpracování ošetřit. Ošetření singularit je popsáno v rozboru vlastní metody (kapitola 4). V tomto případě singulární případ nastává, pokud jsou vektorové součiny $(\vec{p}_1 \times \vec{p}_2)$ a $(\vec{q}_1 \times \vec{q}_2)$ rovnoběžné. Vektorové součiny budou rovnoběžné, právě když:

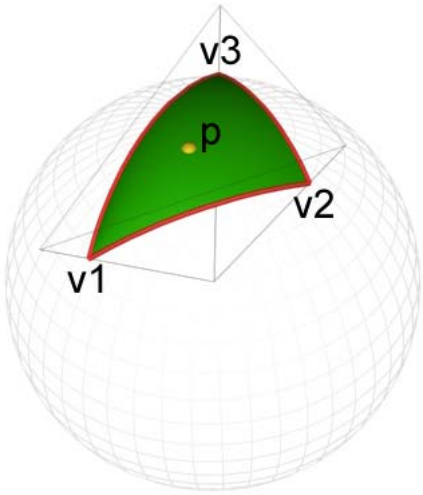
$$(\vec{p}_1 \times \vec{p}_2) \times (\vec{q}_1 \times \vec{q}_2) = \vec{0} \qquad (3.17)$$

Na základě předchozích úvah je nyní možné snadno odvodit test na polohu bodu ve sférickém trojúhelníku.

Test na polohu bodu ve sférickém trojúhelníku

Hrany trojúhelníku můžeme stejně jako v případě oblouků nahradit rovinami danými koncovými body hran a středem parametrické domény. Bod je pak testován, zda je stejně orientován vůči třem rovinám, resp. zda je bod vždy na stejné straně od roviny. Stačí tedy vyhodnocovat znaménko testu bod vůči rovině, což je standardní determinantový test výpočetní geometrie. Test je znázorněn na obrázku 3-9.

[9] Vhodnou metodou řešení takovýchto malých soustav je například Cramerovo pravidlo.



Obrázek 3-9: Test na polohu bodu ve sférickém trojúhelníku

Matematicky lze test vyjádřit následujícím zápisem:

$$((v_1 \times v_2).p \geq 0) \wedge ((v_2 \times v_3).p \geq 0) \wedge ((v_3 \times v_1).p \geq 0) \qquad (3.18)$$

, kde v_1, v_2 *a* v_3 jsou vrcholy sférického trojúhelníku a p je testovaný bod. Stejně jako v předchozím testu je třeba dbát na singulární případy – tj. testovaný bod na hraně sférického trojúhelníku.

3.4.4. Hledání průsečíků

S využitím předchozích testů je nyní možné sestrojit překrytí grafů. Jak bylo uvedeno výše, průsečíky jsou klíčovými elementy pro dosažení sdílené topologie. Prvním krokem je tedy výpočet množiny průsečíků, což bude později využito k dělení hran a následné tvorbě supermeshe.

Další otázkou je tedy způsob nalezení všech průsečíků. To je možné brutální silou testováním každé hrany s každou. Tento ne příliš efektivní algoritmus má složitost v nejhorším případě $O(E_1.E_2)$, kde E_1 je počet hran modelu M_1 a E_2 počet hran modelu M_2. Tímto spadá algoritmus do třídy algoritmů $O(N^2)$, z čehož plyne pro velká N velká časová náročnost. V následujícím oddílu je popsán algoritmus s menší očekávanou složitostí, který využívá informace o sousednosti jednotlivých trojúhelníků. Algoritmus však úzce souvisí s použitou datovou strukturou, nejdříve bude tedy vysvětlena datová struktura.

Datová struktura

Z předchozího popisu je zřejmé, že pro uložení modelů je vhodnější hranově orientovaná datová struktura. Je to zejména vhodné v případě, že je třeba ve smyčce projít všechny hrany, což by bylo v případě trojúhelníkově orientované datové struktury obtížnější. Článek [3] navrhuje datovou strukturu okřídlená hrana (winged edge), zatímco článek [1] navrhuje sofistikovanější datovou strukturu pracující s orientovanými polopřímkami (DCEL[10]).

[10] DCEL – Doubly Connected Edge List.

Nehledě na konkrétní datovou strukturu, je třeba mít k dispozici následující informace, resp. je třeba získávat je v konstantním čase.

- Každý průsečík obsahuje odkazy na dvě hrany, které daný průsečík tvoří a parametrické vyjádření průsečíku vzhledem k protínajícím se hranám[11].
- Každá hrana má seznam odkazů na průsečíky, které daná hrana má.
- Každý vrchol obsahuje seznam odkazů na hrany, které z daného vrcholu vycházejí. Tato informace není nutná pro samotnou fázi hledání průsečíků, ale je třeba ve fázi triangulace supermeshe[12].

Algoritmus hledání průsečíků

Algoritmus vychází z článku [3], využívá znalosti sousednosti jednotlivých trojúhelníků. Vylepšení původního $O(N^2)$ algoritmu spočívá v tom, že dopředu je možné určit množinu hran z modelu M_2, které by mohly být eventuelně protnuty hranou z modelu M_1. Algoritmus lze schématicky zapsat následovně:

1. $v_1 \leftarrow$ první vrchol modelu M_1
2. $f \leftarrow$ trojúhelník modelu M_2, na který se mapuje vrchol v_1
3. Všechny hrany vycházející z vrcholu v_1 přidat do pracovního seznamu WorkList (WL) a nastavit jim příznak used
4. Dokud není seznam WL prázdný:
    a) $e_a \leftarrow$ další hrana ze seznamu WL
    b) $v_1, v_2 \leftarrow$ koncové vrcholy hrany e_a
    c) Přidat všechny hrany tvořící trojúhelník f do seznamu CandidateList (CL)
    d) Dokud není CL prázdný:
        i. $e_b \leftarrow$ další hrana z CL
        ii. Jestliže existuje průsečík e_a a e_b pak:
            1. Přidat průsečík i do seznamu průsečíků
            2. Nastavit odkazy z e_a a e_b na i
            3. $f \leftarrow$ trojúhelník sousedící s f přes hranu e_b
            4. Přidat zbývající dvě hrany f do seznamu CL
    e) Přidat všechny nepoužité hrany vycházející z v_2 do seznamu WL a nastavit jim příznak used

Složitost algoritmu

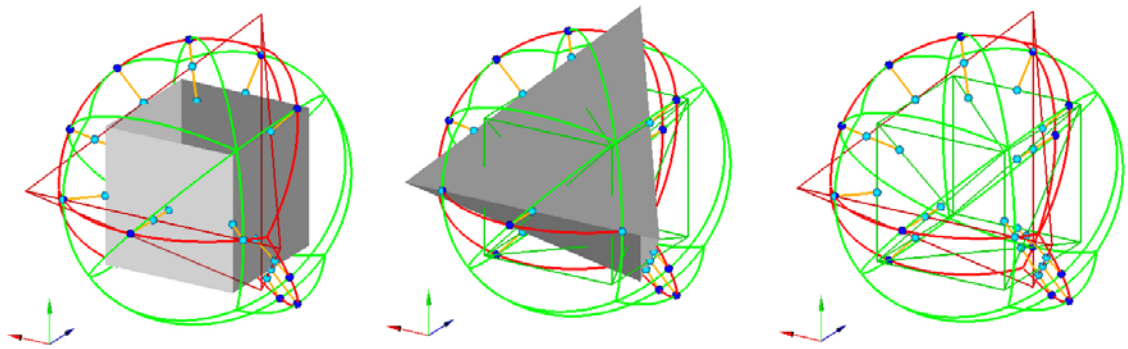
V článku [3] je uvedena složitost tohoto algoritmu $O(E_{M1} + I_{tot})$, kde E_{M1} je počet hran modelu M_1 a I je počet nalezených průsečíků. Pro většinu modelů se očekává $I_{tot} << E_{M1}.E_{M2}$, kde E_{M1}, E_{M2} jsou počty hran modelu M_1, resp. M_2.

3.4.5. Generování supermeshe

Až doposud se pracovalo s parametrizacemi modelů. Hledání korespondence (mapování) a průsečíku probíhalo ve společné parametrické doméně. Nyní lze využít informace získané v předchozích krocích ke konstrukci supermeshe, tj. modelu, který je kombinací obou zdrojových modelů. Ačkoli byly počítány průsečíky oblouků, resp. jejich parametrické vyjádření, je možné tyto průsečíky zobrazit zpět na hrany. Jak již

[11] Viz parametry s_p a s_q v soustavě rovnic 3.16.
[12] Viz oddíl 3.4.6.

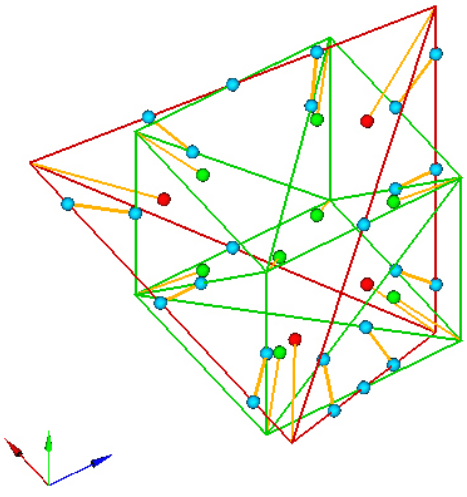
bylo uvedeno, hrany zdrojových těles jsou v obecném případě mimoběžné, protínají se až po zobrazení do společné parametrické domény. Nicméně parametrické vyjádření průsečíků zůstává stále stejné. Zobrazením zpět se tedy rozumí prosté převzetí parametrických vyjádření průsečíků z obrazového prostoru a jejich výpočet pro vzorové modely. Následující obrázek demonstruje situaci.



Obrázek 3-10: Zobrazení modelů do parametrické domény a zpětné promítnutí průsečíků na zdrojové modely

Z obrázku je vidět, že z jednoho průsečíku v obrazovém prostoru vznikají obecně dva body v prostoru vzorovém[13], tj. jeden na hraně modelu M_1 a druhý na hraně modelu M_2. Důležité také je, že tyto body jsou korespondující a při animaci se budou pohybovat z extrémní polohy na hraně modelu M_1 do druhé extrémní polohy na hraně modelu M_2.

Jsou-li tedy průsečíky zobrazené zpět na vzorové hrany, máme spolu s mapováním vrcholů M_1 na povrch M_2 a obráceně kompletní informaci o korespondenci vrcholů. Tuto skutečnost schematicky znázorňuje obrázek 3-11. Červené body znázorňují mapování vrcholů modelu M_1 na povrch M_2, zelené body znázorňují mapování vrcholů M_2 na povrch modelu M_1 a modře jsou znázorněny průsečíky. Žlutými úsečkami jsou spojeny korespondující vrcholy.



Obrázek 3-11: Kompletní korespondence vrcholů

Pokud bude při interpolaci vrcholů použita prostá lineární interpolace, budou žluté úsečky rovněž znázorňovat trajektorii korespondujících vrcholů.

[13] Pouze jeden bod by vznikl, pokud by se hrany protínaly i v prostoru obrazovém.

Cílem dalšího postupu je sloučit oba zdrojové objekty do jednoho. Místem, kde se modely sloučí, jsou právě nalezené průsečíky. Nejdříve je však potřeba provést sjednocení datových struktur obou zdrojových objektů. Je tedy třeba provést sloučení množin vrcholů ($V = V_1 \cup V_2$) a sloučení množin hran ($E = E_1 \cup E_2$). Nemá smysl sjednocovat množiny trojúhelníků, neboť výslednou trojúhelníkovou síť bude teprve třeba vytvořit[14].

Další postup slučování je následující. Nejdříve se projdou všechny hrany modelu M_1 a v každém průsečíku je hrana rozdělena. Toto dělení samozřejmě způsobí vznik nových vrcholů, které se vkládají do datové struktury supermeshe. Při vložení nového vrcholu je vhodné uložit, který průsečík daný vrchol vytvořil. V dalším kroku se projdou všechny hrany modelu M_2 a opět se dělí hrany, na rozdíl od předchozího kroku se však již nevkládají nové vrcholy, ale používají vrcholy již vložené – využívá se uložené informace o tom, který průsečík vytvořil který vrchol. Tímto jsou oba modely „svařeny“ a tvoří jeden model – supermesh. Postup spojení je formálně zachycen v následujícím algoritmu.

1. sjednocení množin vrcholů a hran zdrojových modelů
2. pro všechny hrany e z modelu M_1:
    a) pro všechny průsečíky i hrany e:
        i. rozdělit hranu v průsečíku i $\Rightarrow$ nový vrchol v_{new}
        ii. uložit do datové struktury průsečíku i odkaz na vrchol v_{new}
3. pro všechny hrany e z modelu M_2:
    a) pro všechny průsečíky i hrany e:
        i. rozdělit hranu v průsečíku, pro rozdělení použít vrchol, na který je odkaz v datové struktuře i

Důležitým krokem ve výše popsaném algoritmu je procedura dělení hrany, proto bude podrobněji popsána v následujícím textu.

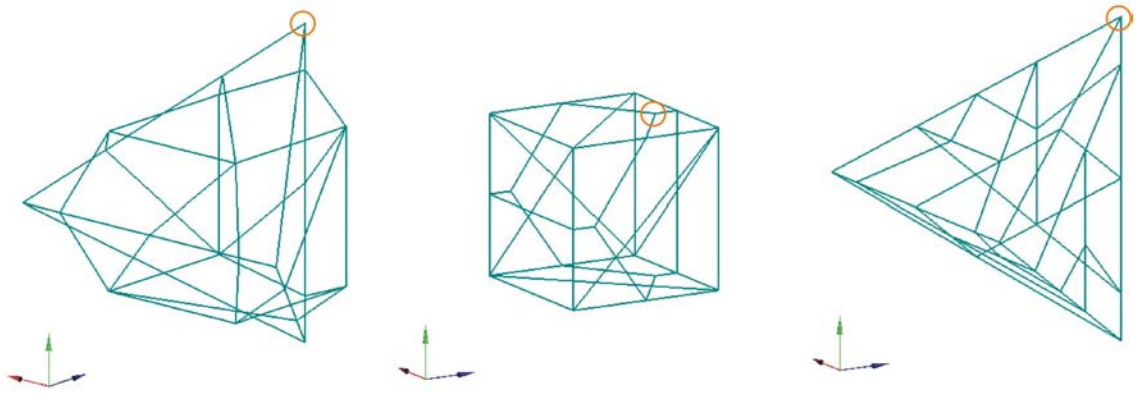
Dělení hrany

V datové struktuře pro uložení hrany je odkaz na lineární seznam průsečíků, které daná hrana má. V proceduře dělení hrany je tedy třeba projít sekvenčně tento seznam a hranu rozdělit. Je vhodné, aby byl tento seznam seřazený, pak je totiž možné spolu s procházením seznamu ihned generovat výsledné rozdělené úseky. Po zpracování celé hrany je možné rozdělenou hranu z datové struktury vyloučit, případně jí nastavit příznak neaktivní.

Úprava geometrie supermeshe

Nyní tedy máme k dispozici supermesh, který je kombinací obou zdrojových těles. Bylo docíleno sdílené topologie, ale po zobrazení vypadá supermesh zatím následovně (obrázek 3-12 a).

[14] Viz oddíl 3.4.6.



a) b) c)

Obrázek 3-12: Supermesh a) Dosud neztransformovaný supermesh b) Supermesh ztransformovaný do tvaru modelu M_1 c) Supermesh ztransformovaný do tvaru modelu M_2

Z obrázku 3-12 a) je vidět, že některé vrcholy ještě nemají správnou polohu. Příkladem nesprávně umístněného vrcholu vzhledem k modelu M_1 je oranžově označený vrchol na obrázku 3-12 a). Tvar supermeshe je nyní něco mezi tvarem modelu M_1 a tvarem modelu M_2. Nyní je možné si zvolit, na jaký tvar supermesh transformovat, a tedy zvolit tvar M(0). Aby byla zachována logická posloupnost, je vhodné supermesh transformovat do tvaru M_1. Konkrétně se jedná o transformaci těch vrcholů, které byly původně vrcholy modelu M_2 a zůstaly ve svých původních polohách. Zbývá tedy posunout tyto vrcholy na povrch modelu M_1, k čemuž lze využít informace o mapování vrcholů získané v kroku hledání korespondence (transformace supermeshe do tvaru M_1 a M_2 se nachází na obrázku 3-12 b) resp. 3-12 c); aby bylo možné pozorovat, co se děje s konkrétním vrcholem, je na obrázku jeden vrchol označen).

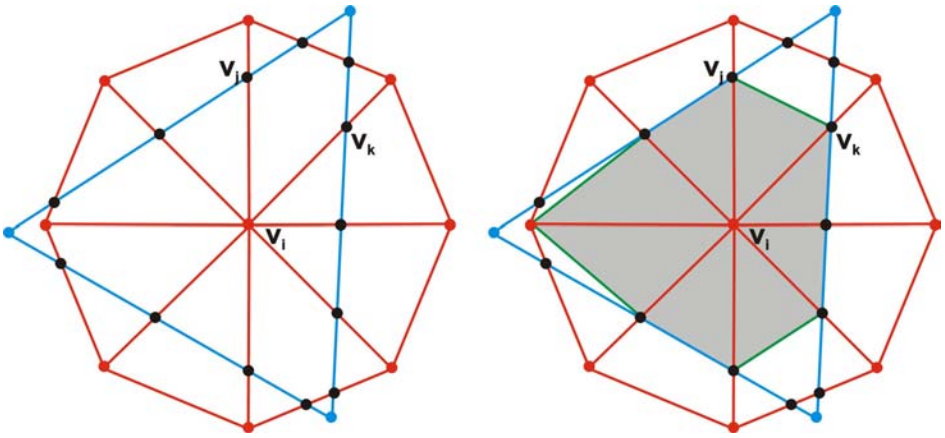
Poznámka:

Pokud by bylo třeba z nějakých důvodů, aby tvar M(0) byl totožný s tvarem M_2, bylo by třeba zase naopak transformovat vrcholy, které původně náležely modelu M_1, ale navíc ještě polohy průsečíků.

3.4.6. Triangularizace supermeshe

Nyní máme supermesh popsaný množinou hran. Pro účely stínování a další manipulace je vhodné mít supermesh vyjádřen jako trojúhelníkovou síť. Z předchozího postupu vznikly polygony, které mají v obecném případě až šest hran. Postup metody triangularizace je následující.

Pro každý vrchol v_i supermeshe je udržován seznam hran, které z tohoto vrcholu vycházejí. Označme tento seznam N_i. Nová trojúhelníková ploška f = (i, j, k), kde i, j, k jsou indexy vrcholů, bude vytvořena ze dvou sousedních hran seznamu N_i. Pokud neexistuje hrana mezi vrcholy v_j a v_k, je navíc vložena ještě nová hrana. Nová hrana musí být opět přidána do seznamu N_j, resp. N_k, tj. do seznamu hran vycházejících z vrcholu v_j, resp. v_k. Situace je podrobněji znázorněna na obrázku (pro jednoduchost ve 2D).



Obrázek 3-13: Trinagulace okolí bodu v_i

Pro správné procházení seznamů hran vycházejících z daného vrcholu je třeba mít tyto hrany seřazené podle úhlu, resp. je třeba stanovit jednoznačné pořadí. Hrany ovšem nejsou v jedné rovině, řazení je tedy třeba provádět v projekci do roviny.

Algoritmus trinagularizace supermeshe je zachycen v následujícím pseudokódu.

1. pro každý vrchol v_i
   a) stanovit pořadí hran v seznamu N_i
   b) $e_j \leftarrow$ první hrana seznamu N_i, $v_j \leftarrow$ koncový vrchol hrany e_j
   c) pro každou hranu seznamu N_i
      i. $e_k \leftarrow$ další hrana seznamu N_i, $v_k \leftarrow$ koncový vrchol hrany e_k
      ii. jestliže neexistuje hrana mezi body v_j, v_k
         1. vložit novou hranu e_{new} tvořenou body v_j, v_k
         2. přidat hranu e_{new} do seznamů N_j, N_k
         3. znovu uspořádat seznamy N_j, N_k
      iii. jinak $e_{new} \leftarrow$ hrana mezi vrcholy v_j, v_k
      iv. vytvořit novou trojúhelníkovou plošku tvořenou hranami e_j, e_k, e_{new} a vrcholy v_i, v_j, v_k

Složitost algoritmu závisí na způsobu zjištění, zda mezi danými dvěma vrcholy existuje hrana. Pokud jsme toto schopni určit v konstantním čase, je složitost algoritmu O(N), kde N je počet vrcholů supermeshe. Počet vrcholů supermeshe je:

$$|V| = |V_1| + |V_2| + |I| \qquad (3.19)$$

, kde $|V_1|$ počet vrcholů modelu M_1, $|V_2|$ počet vrcholů modelu M_2 a $|I|$ počet průsečíků.

3.5. Interpolace

Je-li supermesh popsaný trojúhelníkovou sítí, je možné přikročit k vlastní animaci, tj. interpolaci korespondujících vrcholů. Většina metod toto řeší pouze lineární interpolací poloh korespondujících vrcholů, tj.:

$$V(t) = V(0) + t(V(1) - V(0)), t \in <0;1> \qquad (3.20)$$

, kde *V* je množina vrcholů supermeshe, *V(0)* je poloha vrcholů supermeshe M taková, že M je ve tvaru M_1 a podobně *V(1)* je poloha vrcholů supermeshe M taková, že M je ve tvaru M_2.

Kromě interpolace vrcholů je možné interpolovat i jiné parametry. Například barvu, průhlednost, texturové souřadnice atd. Například pro interpolaci barvy je nutné již ve fázi výpočtu mapování vrcholů modelu M_1 na povrch modelu M_2 kromě výpočtu polohy mapovaného vrcholu dále vypočítávat barvu jako lineární kombinaci barvy ve vrcholech trojúhelníka, na který se daný vrchol mapuje.

4. Rozbor použité metody

Použitá metoda byla inspirována články [3], [1] a [4]. Z teoretického rozboru je vidět, že výpočet morphingu se skládá z několika dílčích kroků. Zde prezentovaný postup tedy nesleduje jedinou metodu, ale různé kroky čerpá z různých zdrojů, neboť použité články se většinou soustředí pouze na konkrétní krok a ostatní jsou zmíněny pouze okrajově. Dalším důvodem byla snaha o nalezení nejvhodnějších postupů, které by zároveň nebyly náročné na implementaci.

V následujícím textu budou postupně rozebrány jednotlivé kroky, které byly nastíněny v teoretickém rozboru. Naše metoda se liší od postupů popsaných v článcích zejména v tom, že se snaží zachytit a ošetřit možné singulární případy. V [1] je tento problém, který mimochodem nastává poměrně často, řešen perturbací bodů. V [4] jsou zdroje singularit filtrovány již ve fázi zobrazení, a to tak, že vrcholy způsobující singularity jsou zafixovány spolu s hraničními vrcholy. V [3] nejsou singulární případy brány v úvahu.

4.1. Hledání korespondence

4.1.1. Zobrazení

Metoda, která je předmětem práce, se omezuje pouze na star-shaped tělesa. Omezení vyplývá z použité metody zobrazení – sférické projekce. Bylo učiněno několik pokusů s relaxační metodou[15], nicméně největším zdrojem potíží bylo hroucení zobrazení. Alexa zamezuje hroucení heuristikou, která je ovšem poměrně implementačně náročná a z časových důvodů se ji nepodařilo důkladně prozkoumat.

Dalším zdrojem problémů bylo hledání star-pointu, jakožto středu promítání. Star-point leží v jádře star-shaped mnohostěnu. Toto jádro lze nalézt protnutím všech poloprostorů definovaných jednotlivými ploškami modelu. Toto zahrnuje výpočet geometrie a topologie, řeší se obvykle technikou D&C[16]. Druhou alternativou je lineární programování. Tento způsob sice nenalezne celé jádro, ale pouze jeden bod, což ovšem pro naše účely stačí. Obě techniky jsou značně implementačně náročné a rovněž nebyly z časových důvodů zatím realizovány. Předpokládá se jejich řešení v budoucnu v rámci doktorandského studia.

V práci je star-point aproximován těžištěm objektu. Existují však tělesa, jejichž těžiště není star-pointem nebo dokonce vůbec neleží uvnitř tělesa. V těchto případech je bohužel nutné polohu star-pointu upravit manuálně. Kontrolu, zda je daný bod star-pointem, je možné provést v lineárním čase testováním, zda všechny normály plošek modelu směřují od star-pointu, resp. jestli star-point má stejnou polohu vůči všem rovinám, které tvoři plošky modelu.

[15] Viz oddíl 3.3.2.
[16] D&C – Divide and Conquer.

4.1.2. Výpočet mapování

Pro hledání mapování byl použit test na polohu bodu ve sférickém trojúhelníku popsaný v oddílu 3.4.3. Pro každý vrchol v_i modelu M_1 jsou procházeny všechny trojúhelníky f_j modelu M_2. Jakmile je nalezen trojúhelník, který obsahuje daný bod, je vyjádřena poloha bodu vzhledem k trojúhelníku v barycentrických souřadnicích. Mějme tedy trojúhelník T daný vrcholy v_1, v_2, v_3 a bod p, který se mapuje na T. Je třeba si však uvědomit, že pracujeme v obrazovém prostoru, bod p tedy leží na jednotkové kouli a neleží v rovině T. Vyřešením následující soustavy získáme souřadnice t, u, w, které vyjadřují vrchol p jako lineární kombinaci vrcholů trojúhelníka T.

$$\begin{aligned} tv_{1x} + uv_{2x} + wv_{3x} &= p_x \\ tv_{1y} + uv_{2y} + wv_{3y} &= p_y \\ tv_{1z} + uv_{2z} + wv_{3z} &= p_z \end{aligned} \qquad (4.1)$$

Protože však bod p neleží v rovině trojúhelníku T, je třeba vypočítat souřadnice t', u', w' bodu p', který je průmětem bodu p z povrchu koule do roviny trojúhelníka T. Souřadnice t', u', w' vypočítáme takto:

$$\begin{aligned} t' &= \frac{t}{t+u+v} \\ u' &= \frac{u}{t+u+v} \\ w' &= \frac{v}{t+u+v} \end{aligned} \qquad (4.2)$$

Souřadnice t', u', w' jsou barycentrické souřadnice bodu p′ vzhledem k vrcholům trojúhelníku T.

4.1.3. Hledání průsečíků

Pro hledání průsečíků byl použit algoritmus „procházky“ (walking) popsaný v oddílu 3.4.4. Při samotném testování, zda se dva oblouky protínají, je třeba brát v úvahu singulární případy:

- oblouky se dotýkají,
- oblouky se úplně nebo částečně překrývají.

První singularita je odhalitelná poměrně snadno, nicméně je třeba si uvědomit důsledky. Oblouky se dotýkají, pokud jeden z parametrů[17] je roven jedné nebo nule.

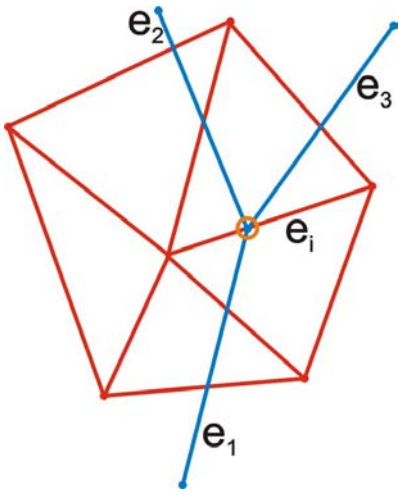
Poznámka:

Zde je třeba připomenout, že rovnost dvou reálných čísel není vhodné testovat v programovém zápisu přímo, ale pomocí nějakého ε okolí, tj. testy typu x = 0.0 nahradit testy $(x<\varepsilon) \wedge (x>-\varepsilon)$. ε je třeba experimentálně určit, pro konkrétní aplikaci a pro konkrétní reprezentaci reálných čísel.

Důsledky prvního druhu singularity jsou následující. Dojde-li k dotyku dvou oblouků, tj. alespoň jeden oblouk se protíná s druhým v jednom ze svých koncových bodů,

[17] Viz parametrické vyjádření průsečíků dvou oblouků (vztah 3.16).

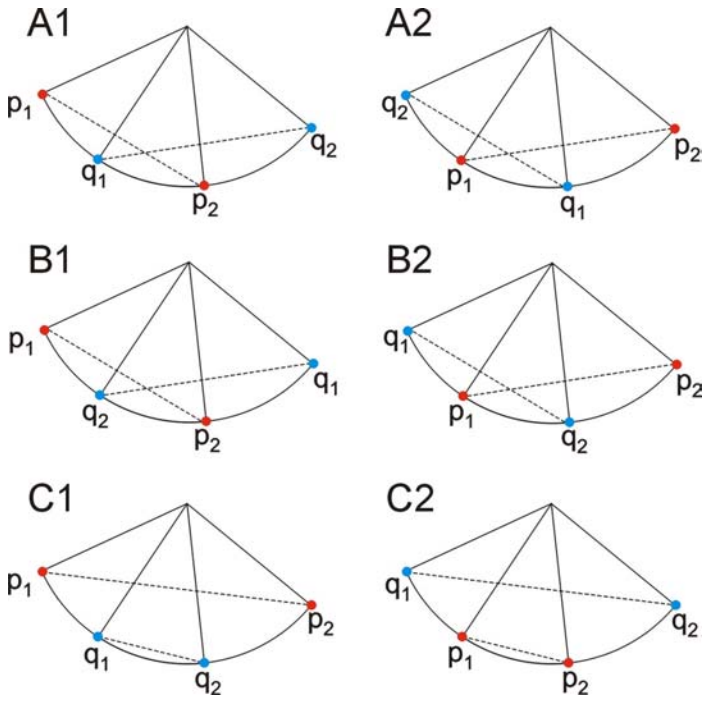
existuje alespoň jeden další oblouk, který se dotýká ve stejném místě. Pokud by tomu nebylo tak, existovala by hrana, z jejíhož jednoho vrcholu nevychází žádná jiná hrana, což je pro zde uvažovanou třídu těles nepřípustné (jednalo by se o non-2-manifold). Jinak řečeno, z koncového bodu, ve kterém dochází k protínání, musí nutně vycházet alespoň jedna další hrana. Situace je pro názornost ukázána na obrázku 4-1 (pro jednoduchost ve 2D).



Obrázek 4-1: Dotyk hran. Dotýká-li se hrana e_1 hrany e_i, pak existuje alespoň jedna hrana (zde e_2, e_3), která se dotýká ve stejném bodě

Důsledkem toho je, že při výpočtu průsečíků budou objeveny nejméně dva průsečíky na stejném místě, což by později ve fázi dělení hran působilo vznik hran nulové délky. V případě znázorněném na obrázku bude mít hrana e_i tři průsečíky na stejném místě (tj. průsečíky s hranami e_1, e_2, e_3). Tento případ není dost dobře možné filtrovat již ve fázi hledání průsečíků. Nicméně je to možné zařídit poté, a to tak, že se prochází pole průsečíků a průsečíky se stejnou polohou se ztotožňují.

Druhý druh singularity se opět dá odhalit poměrně snadno, ale obtížnější je jeho další zpracování. Oblouky se částečně nebo úplně překrývají, pokud roviny, v nichž tyto oblouky leží, koincidují. Pokud se tedy oblouky překrývají, mají společnou určitou část. Pro další zpracování nás pouze zajímá parametrické vyjádření extrémů překrývajícího se úseku oblouků. Jednotlivé případy, které mohou nastat jsou znázorněny na obrázku 4-2.



Obrázek 4-2: Singularity v případě, že se oblouky překrývají

Samozřejmě existují ještě další případy, kdy má jeden oblouk nulovou délku, oblouky se kompletně překrývají (jsou totožné), nebo se dotýkají, ale to jsou extrémní případy výše uvedených. Výše zobrazené případy je třeba odlišovat (což znamená testování vzájemných poloh bodů) a dodržovat správné pořadí těchto průsečíků.

V okamžiku, kdy jsou nalezeny všechny průsečíky, je možné přejít ke kroku konstrukce supermeshe.

4.2. Konstrukce supermeshe

Prvním krokem konstrukce supermeshe je sjednocení datových modelů M_1 a M_2. Bylo by sice možné upravovat jeden ze zdrojových modelů, tímto by se však ztratila informace o původní reprezentaci, je tedy vhodnější vygenerovat nový model M. Pro sjednocení je třeba nakopírovat datové struktury modelu M_1 a M_2 do datové struktury modelu M. Je třeba si uvědomit, že struktury jsou provázány ukazateli (ukazatel na seznam průsečíků, ukazatel na seznam hran atd.), je tedy třeba vytvořit hluboké kopie, aby nebyly nějaké odkazy neplatné.

Dalším krokem je dělení hran. Předtím je však vhodné seřadit průsečíky v nichž se budou hrany dělit. Pokud však použijeme pro hledání průsečíků algoritmus „procházky“ z oddílu 3.4.4., budou již průsečíky hran modelu M_1 ve správném pořadí, což vychází ze způsobu „pochodu“ pro trojúhelnících. Zbývá tedy pouze seřadit průsečíky hran modelu M_2. Pro řazení byl použit algoritmus bubble-sort z důvodů jednoduchosti.

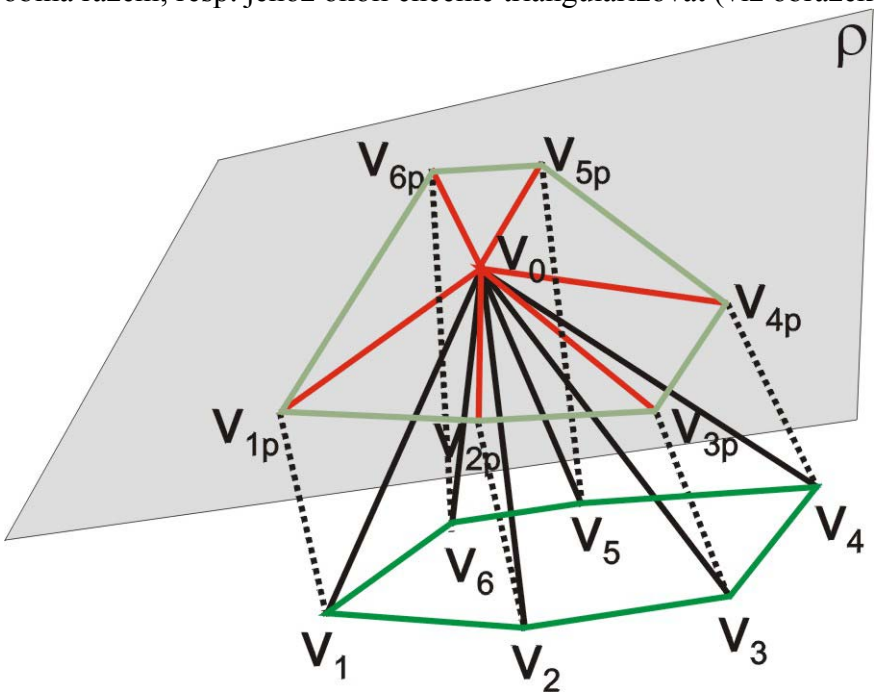
Ve fázi dělení hrany je třeba počítat s jistými singulárními případy. Je-li parametrické vyjádření průsečíku 0.0 nebo 1.0, tj. je-li průsečík ve vrcholu hrany, není třeba vkládat nový vrchol. Tyto případy mohou nastat, pokud se oblouky v obrazovém prostoru dotýkají. Rovněž je třeba ošetřit případ, kdy je více průsečíků koincidentních.

Je-li supermesh vytořen, je třeba ho dále transformovat do tvaru modelu M_1. Konkrétně je třeba transformovat vrcholy, které náležely původně modelu M_2. Každý vrchol modelu M_2 je vyjádřený pomocí barycentrických souřadnic vzhledem k nějakému trojúhelníku modelu M_1. Stačí tedy převést barycentrické souřadnice vrcholů modelu M_2 zpět na kartézské, čímž se body dostanou do roviny daného trojúhelníku modelu M_1. Posledním krokem je triangularizace supermeshe. Jedná se v podstatě o variantu metody navržené v [4] (a popsané v teoretickém popisu), s tím rozdílem, že v [4] je tato metoda použitá pro 2D případ. Naše metoda musela být přizpůsobena pro 3D případ.

Poznámka:

Připomeňme, že v případě harmonických map ([4]) bylo použito zobrazení $h: M \rightarrow H$, kde $M \in R^3$ a $H \in R^2$. Triangulace probíhá v zobrazení, tj. v rovině, kde není třeba promítat hrany do roviny, jako je tomu v případě naší metody (viz dále).

Pro triangularizaci je třeba mít hrany seřazené podle úhlu, resp. je třeba mít hrany v nějakém pořadí, tak aby při spojování jejich koncových bodů nedocházelo ke křížení. Stanovení takového pořadí ve 3D však není dost dobře možné (polygony supermeshe jsou v obecném případě neplanární). Je tedy třeba promítnout tyto hrany do 2D a tam provést klasické úhlové řazení. Hrany nejdříve promítneme do roviny vrcholu, okolo kterého probíhá řazení, resp. jehož okolí chceme triangularizovat (viz obrázek 4-3).



Obrázek 4-3: Projekce hran vycházejících z vrcholu v_0 do roviny ρ

Na obrázku 4-3 je v_0 vrchol, okolo kterého probíhá řazení, ρ je rovina určená bodem v_0 a vektorem v_0, star-point. Body $v_{1p}, v_{2p}, \ldots, v_{6p}$ jsou kolmé projekce bodů na tuto rovinu. Rovina ρ je stále v obecné poloze, pro řazení bodů by bylo vhodné ji buď rotovat do roviny xy nebo promítnout do směru největšího spádu (tj. podle maximální složky normály určit, zda promítnout do roviny xy, xz nebo zy). Byly vyzkoušeny oba způsoby; i když je druhý způsob pouhou aproximací, pro naše účely je dostačující. Navíc první způsob je komplikovanější, neboť zahrnuje výpočet úhlů. V obrázku jsou dále vyznačeny zeleně hrany, které tvoří triangulaci okolí bodu v_0.

Při triangularizaci supermeshe postupem z oddílu 3.4.6. jsou vkládány nové hrany. Aby nedocházelo k duplicitám, je třeba při každém vložení hrany zjistit, zda hrana již neexistuje. Hranu lze hledat sekvenčním prohledávání pole hran, ale tímto by o řád vzrostla algoritmická složitost triangularizace. Je tedy vhodné vytvořit pomocnou datovou strukturu, která by na dotaz na existenci hrany odpovídala v konstantním čase. V práci byla použita následující datová struktura. Pro každý vrchol v_i je udržován lineární seznam vrcholů v_j, které s vrcholem v_i sousedí (tj. jsou spojeny hranou). Při dotazu, zda existuje hrana mezi vrcholy v_a, v_b stačí pak projít seznam asociovaný s vrcholem v_a a hledat v něm výskyt vrcholu v_b. Velikost seznamu asociovaného s vrcholem odpovídá stupni vrcholu, přičemž průměrný stupeň vrcholu v jednoduchých mnohostěnech je 6. Lze tedy říci, že na dotaz na existenci hrany je možné odpovědět v konstantním čase.

Postup uvedený v oddílu 3.4.6. může produkovat duplicitní trojúhelníky. Tomu lze snadno zabránit přidáním čítače ke každé hraně. Čítač se inkrementuje, pokud se hrana stane hranou nějakého trojúhelníku. Jedna hrana je u neohraničených jednoduchých mnohostěnů sdílena právě dvěma trojúhelníky, pokud tedy čítač bude větší jak 2, nebude hrana použita, tj. nebude vložen ani nový trojúhelník.

Poté, co jsou všechny vrcholy modelu zpracovány postupem z oddílu 3.4.6., je nutné ještě zkontrolovat orientaci všech trojúhelníků (tj. po směru hodinových ručiček nebo proti směru hodinových ručiček). Pro stínování a další manipulace se supermeshem je vhodné, aby byla orientace v celém modelu konzistentní.

Posledním krokem je výpočet normál triangulovaného supermeshe. To zahrnuje výpočet normál plošek a výpočet normál ve vrcholech. Normály plošek jsou vypočítány klasickým vektorovým součinem, normály ve vrcholech pak Gouraoudovou metodou ([9]).

4.3. Interpolace

Byla použita jednoduchá lineární interpolace mezi extrémní polohami vrcholů. Jedna extrémní poloha je vrchol na povrchu modelu M_1 a druhá vrchol na povrchu modelu M_2. Ve výsledném supermeshi jsou tři druhy vrcholů. Vrcholy z původního modelu M_1, vrcholy z původního modelu M_2 a vrcholy vzniklé z průsečíků.
Předpokládejme, že t se postupně mění od 0.0 do 1.0. Pak pro skupinu původních vrcholů modelu M_1 platí:

$$v_i(t) = v_{iM1} + t(\phi(b_{i1}) - v_{iM1}) \qquad (4.3)$$

, kde v_{iM1} je poloha vrcholu v_i z modelu M_1, $\phi(b_{i1})$ je funkce převádějící barycentrické souřadnice b_{i1} na kartézské a argument funkce ϕ vyjadřuje barycentrické souřadnice vrcholu v_i vzhledem k trojúhelníku modelu M_2. Podobně lze zapsat pro skupinu původních vrcholů modelu M_2:

$$v_j(t) = \phi(b_{j2}) + t(v_{jiM2} - \phi(b_{j2})) \qquad (4.4)$$

, kde v_{jM2} je poloha vrcholu v_j z modelu M_2, $\phi(b_{j2})$ je funkce převádějící barycentrické souřadnice b_{j2} na kartézské a argument funkce ϕ vyjadřuje barycentrické souřadnice vrcholu v_j vzhledem k trojúhelníku modelu M_1. Pro vrcholy vzniklé z průsečíků platí:

$$v_k(t) = i_{kM1} + t(i_{kM2} - i_{kM1}) \qquad (4.5)$$

, kde i_{kM1} je poloha průsečíku na modelu M_1 (tj. na hraně modelu M_1 – viz parametr s ve vztahu 3.16) a i_{kM2} je poloha průsečíku na modelu M_2 (tj. na hraně modelu M_2 – viz parametr t v rovnici 3.16).

4.4. Shrnutí

Kroky popsané v teoretickém popisu a rozboru metody budou nyní stručně shrnuty do následujícího algoritmu. Dílčí algoritmy, které byly uvedeny již výše, nebudou opakovány, bude na ně pouze odkázáno.

1. Načtení zdrojových modelů M_1 a M_2
2. Zobrazení modelů M_1, M_2 do společné parametrické domény D
3. Výpočet mapování vrcholů V_1 na F_2 (plochy modelu M_2) a V_2 na F_1
4. Výpočet průsečíků (viz algoritmus v oddílu 3.4.4.)
5. Výpočet supermeshe
    a) Sjednocení $V = V_1 \cup V_2$, $E = E_1 \cup E_2$
    b) Řazení seznamů průsečíků asociovaných s hranami E_1, E_2
    c) Shlukování koincidentních průsečíků
    d) Dělení hran
    e) Transformace supermeshe M do tvaru modelu M_1
    f) Triangulace supermeshe (viz algoritmus v oddílu 3.4.6.)

5. Implementace

Pro implementaci bylo použito vývojové prostředí Borland Delphi firmy Inprise pod operačním systémem Windows 2000. Toto prostředí bylo zvoleno pro jednodušší způsob práce s vizuálními komponentami[18] než je tomu u jiných vývojových prostředí. Je tak možné se lépe soustředit na řešení problémů místo na zápolení s různými komplikovanými vizuálními prvky uživatelského rozhraní. Pro zobrazování grafických výstupů byla použita knihovna OpenGL spolu se svými nadstavbami Glu a Glut.

V dalším textu budou popsány implementační detaily metody rozebrané v kapitole 4. Detaily budou popsány pokud možno slovně, bez přímé vazby na jakýkoliv programovací jazyk.

5.1. Vstup

Vstupem programu jsou geometrické objekty dané hraniční reprezentací. Program načítá vstupní data ve standardním formátu TRI[19] a ve formátu ASC. Formát ASC je textový exportní formát aplikace 3ds max. Formát ASC byl použit, protože často bylo třeba experimentovat s různými objekty, které lze snadno v 3ds max vymodelovat. Problémem formátu TRI je, že v tomto formátu není dostatek testovacích množin a objekty v něm uložené nelze snadno modifikovat v nějaké aplikaci. Ve formátu ASC, podobně jako TRI, obsahuje popis trojúhelníkové sítě ve formě tabulky vrcholů a tabulky trojúhelníků, přičemž jeden trojúhelník je určen trojicí indexů (i, j, k) odkazujících do tabulky vrcholů.

5.2. Výstup

Výstupem programu je animace přechodu výchozího tělesa v cílové. Jednotlivé fáze je možné ukládat do obrazového formátu BMP. Sekvence obrázků je pak možné „sloučit“ do jednoho .AVI souboru pomocí externího programu pjBmp2Avi, který mimo jiné umožňuje vybrat i videokodek pro uložení videa.

5.3. Datové struktury

Jak již bylo uvedeno výše, pro tvorbu supermeshe je vhodná hranově orientovaná datová struktura. V této implementaci byla použita varianta struktury okřídlená hrana. Datová struktura se skládá z pole hran (na obrázku 5-1 označeno jako edgeArray), pole vrcholů (na obrázku 5-1 označeno jako vertArray) a pole trojúhelníků.

Struktura pro uložení hrany obsahuje položky:

- indexy do pole vrcholů udávající, kterými vrcholy je daná hrana tvořena,
- indexy do pole trojúhelníků udávající, které trojúhelníky hranu sdílí,

[18] Knihovna VCL – Visual Component Library.
[19] Formát TRI je textový formát systému MVE. Viz [URL2].

- lineární seznam průsečíků podél hrany.

Struktura pro uložení vrcholu obsahuje položky:

- kartézské souřadnice polohy vrcholu,
- souřadnice ve společné parametrické doméně (v našem případě souřadnice projekce bodu na jednotkovou kouli se středem promítání ve star-pointu),
- lineární seznam indexů hran vycházejících z daného vrcholu,
- normály ve vrcholu pro tvar M(0) a pro tvar M(1).

Struktura pro uložení trojúhelníku obsahuje položky:

- trojice indexů odkazující do pole vrcholů na vrcholy trojúhelníku,
- trojice indexů odkazující do pole hran na hrany trojúhelníku,
- normály trojúhelníku pro tvar M(0) a pro tvar M(1).

Poznámka:

Normály jsou ukládány vždy pro tvar M(0) a pro tvar M(1) proto, aby při animaci morphingu nebylo nutné normály stále dopočítávat, ale aby je bylo možné interpolovat.

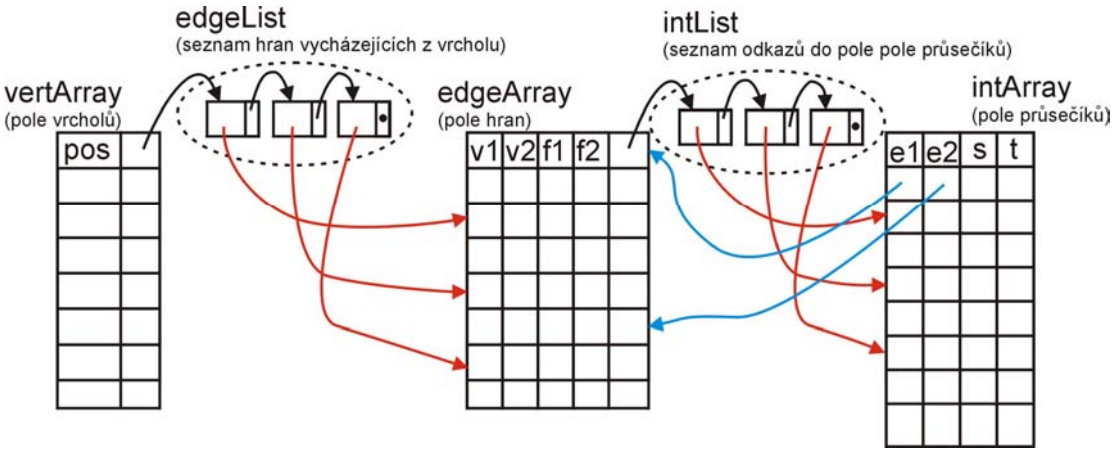
Výše uvedené tabulky slouží pro reprezentaci objektů. Během výpočtu morphingu je však ještě třeba udržovat informace o průsečících a o mapování vrcholů jednoho modelu na povrch druhého a obráceně. Obě tyto informace jsou opět ukládány v polích. Struktura pro uložení průsečíku obsahuje položky:

- indexy hran odkazující do pole hran a udávající, které dvě hrany se protínají,
- parametrické vyjádření průsečíku vzhledem k oběma hranám,
- index vrcholu, který z průsečíků vznikl ve fázi dělení hran.

Struktura pro uložení informací o mapování vrcholů obsahuje položky:

- index vrcholu,
- index trojúhelníku, na který se vrchol mapuje,
- barycentrické souřadnice vrcholu vzhledem k trojúhelníku,
- druh singularity (vrchol se mapuje na vrchol trojúhelníku nebo na hranu),
- místo singularity (která hrana, resp. který vrchol).

Následující obrázek znázorňuje schématicky provázání jednotlivých datových struktur.



Obrázek 5-1: Grafické znázornění datové struktury

Jak z předchozího textu vyplývá, jako hlavní datové struktury se používají pole (vyjma lineárních seznamů edgeList a intList). Pole je vždy počátečně naalokováno na určitou velikost, pokud však počet prvků pole překročí tuto velikost, je třeba pole zvětšit. Proces zvětšení zahrnuje vytvoření nového (většího) pole, nakopírování starého pole do nového a uvolnění starého pole. V jednom okamžiku je v paměti jak staré pole, tak pole nové, a může pak docházet k problémům s nedostatkem paměti. Je tedy třeba vhodně volit počáteční velikost pole a velikost relokační jednotky.

5.3.1. Naplnění datové struktury

Jak bylo uvedeno výše, zdrojová tělesa jsou dodávána ve formě trojúhelníkové sítě. Pro výpočet supermeshe je třeba naplnit datovou strukturu okřídlená hrana. Naplnění struktury zahrnuje vytvoření množiny hran a výpočet sousednosti, tj. které dva trojúhelníky sdílí danou hranu.

Výpočet množiny hran je následující. Projde se pole trojúhelníků a postupně jsou přidávány hrany tvořící trojúhelník. Každá hrana je sdílena právě dvěma trojúhelníky, je tedy třeba dávat pozor, aby nebyla jedna hrana přidána dvakrát. Před každým vložením nové hrany je tedy třeba zjistit, zda vkládaná hrana není již v poli hran obsažena. To by bylo možné sekvenčním způsobem, ale z výpočtu množiny hran by se pak stával $O(N^2)$ problém. Použitím vhodné vyhledávací datové struktury je však možné očekávanou složitost snížit. Jako pomocná datová struktura byla použita varianta invertovaného seznamu, tj. pro každý vrchol v_i je udržován seznam vrcholů v_j, do kterých vede z v_i hrana. Pro zjištění, zda existuje hrana daná vrcholy v_k, v_l stačí pouze projít seznam asociovaný vrcholu v_k a hledat v něm výskyt vrcholu v_l. Tímto by měla být očekávaná složitost nižší, neboť se předpokládá, že seznam vrcholů nebude mít velikost O(N).

Problém, jak určit, které dva trojúhelníky sdílejí danou hranu, lze řešit současně s prováděním předchozí fáze. Lze využít toho, že daná hrana je sdílena právě dvěma trojúhelníky. Pokud je možné vložit hranu do pole hran (tj. hrana v poli dosud neexistuje), nastaví se zároveň atribut obsahující první sousední trojúhelník na právě zpracovávaný trojúhelník. Pokud hrana již existuje, nevloží se sice do seznamu hran, ale nastaví se její atribut obsahující index druhého sousedního trojúhelníku na právě zpracovávaný trojúhelník.

Dále je třeba doplnit struktury pro uložení trojúhelníků o informaci o hranách, kterými jsou jednotlivé trojúhelníky tvořeny. To je možné provést v lineárním čase smyčkou přes všechny hrany s využitím informace o sousednosti, získané v předchozím kroku.

5.3.2. Výběr star-pointu

Jak bylo uvedeno výše, star-point je aproximován těžištěm. Konkrétně je to provedeno tak, že star-point je pevně umístněn do počátku souřadného systému (tj. do bodu (0, 0, 0)) a těleso je posunuto tak, aby jeho těžiště leželo ve středu souřadného sytému. Nicméně uživatelské rozhraní dovoluje měnit polohu tělesa vůči star-pointu, což je možné využít v případě, že těžiště není star-pointem, nebo pokud je třeba docílit nějakých dalších transformací během animace přechodu. Například je možné obě tělesa vzájemně posunout, přičemž je třeba dbát, aby počátek souřadné soustavy byl stále star-pointem. Během animace se pak nebude měnit pouze tvar výchozího tělesa v cílové, ale navíc se bude těleso postupně posouvat z polohy výchozího tělesa do polohy cílového tělesa.

5.3.3. Shlukování průsečíků

Jak bylo uvedeno v oddílu 4.1.3., z dotyků oblouků vznikají koincidentní průsečíky. S koincidentními průsečíky se poměrně obtížně zachází ve fázi dělení hran, bylo by třeba množství podmínek řešících jednotlivé případy, čímž by byl kód poměrně netransparentní a obtížně by se ladil. Jako vhodnější řešení, než „odchycení" a speciální ošetření nejrůznějších případů, které z koincidujících průsečíků plynou, se jeví postprocessing pole průsečíků. Postprocessing spočívá v tom, že se v poli průsečíků hledají průsečíky, které mají stejnou polohu bez ohledu na to, jakým způsobem vznikly. Průsečíky na stejných místech se sloučí a označí se příslušným příznakem.

Nalezení totožných průsečíků brutální silou opět bohužel znamená $O(N^2)$ problém. Je tedy vhodné opět použít nějakou vhodnou pomocnou datovou strukturu, která by snížila alespoň očekávanou složitost. Vhodnou strukturou se ukázala pravidelná trojrozměrná mřížka (3d grid). Pole průsečíků je vloženo do jednotlivých buněk mřížky v závislosti na své poloze. Vyhledávání (a porovnávání na stejnou polohu) pak probíhá pouze v určité buňce. Aby bylo vyhledávání efektivní, je třeba vhodně zvolit rozlišení mřížky (tj. počet buněk v jednotlivých osách). Teorie říká, že je vhodné volit rozlišení M následujícím způsobem:

$$M = \sqrt[3]{\frac{N}{K}} \qquad (5.1)$$

, kde N je rozměr dat (v našem případě počet průsečíků), K je počet elementů v jedné buňce (tj. počet průsečíků v jedné buňce). Je třeba si však uvědomit, že toto platí pro rovnoměrně rozložená data, což v případě průsečíků dvou sférických projekcí platí[20].

Zbývá rozmyslet jak fyzicky realizovat shlukování totožných průsečíků. V práci je toto řešeno následujícím způsobem. Totožné průsečíky jsou spojeny ukazateli do cyklického seznamu. Jakmile je ve fázi dělení hran některý z těchto totožných průsečíků zpracován, rozšíří se pomocí seznamu příznak, že tento průsečík byl již zpracován a všechny ostatní nemají být brány v úvahu (resp. z průsečíku nebude vznikat nový vrchol).

5.3.4. Uživatelské rozhraní

Bylo vytvořeno jednoduché uživatelské rozhraní (popis ovládání se nachází v příloze B). Uživatelské rozhraní tvoří dvě okna. Okno rendereru a okno pro nastavování nejrůznějších voleb. Zobrazování v okně rendereru je řešeno pomocí knihovny OpenGL. V rendereru je možné provádět standardní pohledové transformace a volit způsob stínování.

5.4. Integrace do MVE

Aplikace byla integrována v podobě DLL knihovny do systému MVE. MVE je modulární vizualizační prostředí vyvinuté na Západočeské univerzitě v Plzni. Modul nepotřebuje žádné speciální vstupy, neboť obsahuje vlastní rozhraní pro načítání dat. Podobně i výstup je vyřešen uvnitř modulu, není tedy třeba výstup dále napojovat na renderer.

[20] Samozřejmě lze nalézt modely, pro něž toto platit nebude, zde však jde o průměrný případ.

6. Rozbor výsledků

V následující kapitole budou popsány výsledky konkrétních experimentů. Nejdříve bude zrekapitulována celková algoritmická složitost. Dále bude změřena časová náročnost programu, a to jak celkový čas potřebný pro výpočet, tak čas dílčích kroků výpočtu. Důležitým hlediskem je také počet vrcholů, hran a plošek výsledného supermeshe. V posledním oddílu budou rozebrány případy, jak metoda reaguje na tělesa, která nejsou star-shaped. Příklady několika animací je možné si prohlédnout v příloze C a na přiloženém CD.

6.1. Analýza algoritmické a paměťové složitosti

Zde bude stručně popsána celková složitost algoritmu. V některých krocích byla složitost diskutována již v teoretickém rozboru nebo v rozboru metody. Nebudeme uvažovat kroky nutné pro tvorbu datové struktury, neboť nejsou signifikantní. Analýza složitosti odpovídá krokům zapsaným ve shrnutí kapitoly 4 (oddíl 4.4.). Pro zápis složitosti pomocí O notace je vhodné zavést (resp. zopakovat) několik označení.

- $|F_1|$ – počet plošek modelu M_1.
- $|F_2|$ – počet plošek modelu M_2.
- $|V_1|$ – počet vrcholů modelu M_1.
- $|V_2|$ – počet vrcholů modelu M_2.
- $|E_1|$ – počet hran modelu M_1.
- $|E_2|$ – počet hran modelu M_2.
- $|I|$ – počet průsečíků hran modelu M_1 s hranami modelu M_2.

Krok zobrazení modelů na jednotkovou kouli je jednoznačně lineární záležitost. Stačí projít všechny vrcholy modelu a provést s nimi příslušnou operaci, tj. $O(|V_1|+|V_2|)$. Paměťová složitost tohoto kroku je rovněž lineární.

Výpočet mapování má složitost $O(|V_1|.|F_2| + |V_2|.|F_1|)$, tj. spadá do třídy algoritmů $O(N^2)$. Je totiž třeba projít všechny vrcholy množiny V_1 a hledat, na kterou plošku z množiny F_2 se mapují a obráceně. Pro uložení informací o mapování je třeba prostor $O(|V_1|+|V_2|)$, tj. lineární.

Složitost výpočtu průsečíků je $O(|E_1|+|I|)$, přičemž se očekává, že počet průsečíků $|I|$ je podstatně menší než $|E_1|.|E_2|$. Paměťová náročnost uložení průsečíků je $O(|I|)$.

Sjednocení datových struktur je opět lineární záležitost, neboť se jedná v podstatě o pouhé kopírování. Z paměťového hlediska to znamená, že je vytvořena další struktura o velikosti $O(|V_1|+|E_1|+|F_1|+|V_1|+|E_1|+|F_1|)$.

Pro řazení průsečíků hran byl použit algoritmus bubble-sort, jehož asymptotická složitost je $O(N^2)$. Nicméně je třeba si uvědomit několik „polehčujících okolností“. Díky algoritmu „procházka“ pro hledání průsečíků jsou průsečíky podél hran z množiny

E_1 již seřazené, nicméně je třeba je zkontrolovat, neboť důsledkem singulárních případů je bohužel i nesprávné řazení průsečíků podél hrany. Algoritmus bubble-sort je navíc naimplementován tak, že nedojde-li k výměně prvků v jedné smyčce, algoritmus končí. Kontrolu, zda jsou průsečíky seřazené, je možné provést v lineárním čase $O(T_i)$, kde T_i je počet průsečíku podél hrany e_i. Předpokládá se, že hrana e_i z modelu M_1 nemá $|E_2|$ průsečíků. Platí:

$$\sum_{i \in E_1} T_i = |I| \qquad (6.1)$$

, kde T_i je počet průsečíků podél hrany e_i a e_i je i-tá hrana modelu M_1. Očekávaná složitost je tedy O(|I|), tj. lineární.

Při hledání a shlukování koincidentních průsečíků je použita datová struktura trojrozměrná mřížka. Označíme-li počet elementů v buňce K, pak lze jeden element najít se složitostí O(K). Při shlukování průsečíků je třeba projít |I| průsečíků, v každém kroku je provedeno O(K) dotazů do buňky. Opět je třeba připomenout, že toto platí pro rovnoměrné rozložení průsečíků v mřížce, což je v průměru splněno. Celková složitost shlukování je tedy O(|I|.K). Z hlediska paměťového je třeba pomocná datová struktura, jejíž paměťová náročnost je O(M.M + |I|), kde M je rozlišení mřížky.

Krok dělení hran má složitost $O(|E_1|+|E_2|)$, neboť se jedná o výpočetní smyčku přes všechny hrany supermeshe. Paměťová náročnost tohoto kroku závisí na počtu hran, které vzniknou dělením. Počet hran je $2|I|+|E_1|+|E_2|$ (viz dále), paměťová náročnost je tedy $O(2|I|+|E_1|+|E_2|)$.

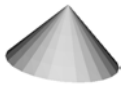
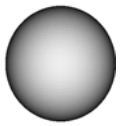
Podobně transformace supermeshe do požadovaného tvaru je rovněž lineární záležitost.

Posledním krokem je triangularizace supermeshe. Triangularizace probíhá ve smyčce přes všechny vrcholy supermeshe, kterých je $|V_1|+|V_2|+|I|$. Z každého vrcholu v_i supermeshe vychází L_i hran. Zpracování každého vrcholu v_i obnáší L_i dotazů, zda existuje hrana mezi koncovými vrcholy hran vycházející z vrcholu v_i. Pomocí vyhledávací datové struktury popsané v 4.2. je možné tento dotaz zodpovědět v průměru v konstantním čase. Složitost triangulace vrcholu v_i je tedy $O(L_i)$. Celková složitost je tedy $O((|V_1|+|V_2|+|I|).L_i)$, přičemž se opět očekává, že L_i je podstatně menší než celkový počet hran supermeshe. Z paměťového hlediska datová struktura v tomto případě naroste o $|E_{tr}|$ hran, tj. o počet hran, které jsou vloženy během triangularizace.

6.2. Měření časů a analýza složitosti supermeshe

Všechny časové testy byly provedeny na počítači s procesorem Pentium II 300 MHz s 128 MB RAM. Na tomto stroji probíhala i implementace. Ačkoli je takový stroj v dnešní době poměrně slabý, jeho parametry nutily optimalizovat některé kroky (zejména se jednalo o kroky se složitostí $O(N^2)$).

Pro měření celkového času potřebného pro výpočet morphingu byly vybrány 4 dvojice zdrojových těles. Tělesa budou nazývána podle názvů souborů, v kterých jsou tělesa uložena. V tabulce 6-1 jsou zapsány údaje charakterizující složitost testovacích těles (tj. počet vrcholů, hran a plošek).

číslo	název tělesa	počet vrcholů	počet hran	počet trojúhelníků	náhled
1	face4.asc	3946	11832	7888	
2	geosphere.asc	66	192	128	
3	capsule.asc	182	540	360	
4	dodeca.asc	12	30	20	
5	star1.asc	182	540	360	
6	cone.asc	122	360	240	
7	geosphere-high res.asc	2562	7680	5120	
8	pokemon.asc	2090	6264	4176	

Tabulka 6-1: Modely použité pro měření času

Kromě měření času byl zároveň sledován počet vrcholů, hran a trojúhelníků výsledného supermeshe. Pro každou dvojici modelů byl proveden výpočet morphingu z modelu M_1 do modelu M_2 a obráceně. Obrácený směr byl testován z toho důvodu, že animace morphingu je vizuálně naprosto stejná (pouze opačné pořadí jednotlivých fází) nehledě na to, zda provádíme morphing $M_1 \rightarrow M_2$ či $M_2 \rightarrow M_1$. Topologie supermeshe však stejná není, ačkoli reprezentuje shodný tvar. Tabulka 6-2 udává kromě doby potřebné na výpočet ještě charakteristiku složitosti výsledného supermeshe.

	$1 \rightarrow 2$	$2 \rightarrow 1$	$3 \rightarrow 4$	$4 \rightarrow 3$	$5 \rightarrow 6$	$6 \rightarrow 5$	$7 \rightarrow 8$	$8 \rightarrow 7$
počet vrcholů	7290	7290	402	402	1372	1372	18018	18229
počet hran	22070	22078	1213	1200	4196	4113	56393	57367
počet plošek	14772	14789	814	800	1831	2743	38507	39045
počet průsečíků	3278	3278	456	456	1454	1454	13382	13593
počet nových hran	3490	3498	218	208	1153	1051	15714	16266
doba výpočtu [ms]	3838	3337	270	242	691	661	26863	27534

Tabulka 6-2: Naměřené hodnoty

V následujícím textu bude provedena diskuse údajů v tabulce. První řádek tabulky udává počty vrcholů supermeshe. Jak již bylo uvedeno výše supermesh má maximálně $|V_1|+|V_2|+|I|$ vrcholů. Je-li například průsečík ve vrcholu, pak se v tomto místě hrana nedělí a nevzniká tak nový vrchol, v obecném případě je tedy vrcholů supermeshe díky singulárním případům méně.

Prvním možná zarážejícím údajem je, že při morphingu 7→8 a 8→7 se liší počty vrcholů supermeshe. Tento rozdíl je způsoben tím, že objekt 8 nesplňuje přesně požadavky star-shaped objektů, resp. má chybně umístněný star-point. Zajímavý je také počet hran výsledného supermeshe, který bezpochyby závisí na počtu hran modelů M_1 a M_2, resp. na počtu jejich průsečíků. Jak již bylo uvedeno očekává se, že počet průsečíků $|I|$ je podstatně menší než $|E_1|.|E_2|$. Tento předpoklad je splněn pro modely zachycené v tabulce stejně jako pro ostatní modely z testovací množiny.

Z počtu hran vyplývá i počet trojúhelníků. Je zřejmé, že počet plošek supermeshe bude výrazně větší než počet plošek obou zdrojových těles dohromady. Jedná se o poměrně důležitý aspekt, s kterým je třeba počítat zejména při práci se složitými objekty. Určení závislosti počtu plošek výsledného supermeshe na základě informací o zdrojových objektech je obtížné, ale můžeme učinit odhad, resp. určit, na kterých parametrech počet plošek závisí.

Pro výsledný supermesh musí jistě platit Eulerova formule pro jednoduchý mnohostěn:

$$V - E + F = 2 \qquad (6.2)$$

, kde V je počet vrcholů, E počet hran a F počet plošek. Vyjádříme-li ze vztahu počet plošek F, dostaneme:

$$F = 2 - V + E \qquad (6.3)$$

Celkový počet vrcholů V supermeshe je:

$$|V| = |V_1| + |V_2| + |I| \qquad (6.4)$$

, kde $|V|$ je celkový počet vrcholů supermeshe. Je třeba si uvědomit, že toto platí přesně pouze za předpokladu, že neexistují průsečíky ve vrcholech. Existuje-li průsečík ve vrcholu, pak se hrana samozřejmě nedělí a je použit samotný vrchol. Zbývá tedy určit počet hran supermeshe. Ten lze také odhadnout na základě počtu průsečíků (tj. $|I|$). Označme T_i^{M1} počet průsečíků hrany e_i modelu M_1. Po rozdělení hrany e_i vznikne $T_i^{M1}+1$ nových hran. Celkový počet nových hran E_{new} můžeme tedy spočítat jako:

$$E_{new} = \sum_{i=0}^{|E_1|} (T_i^{M1} + 1) \qquad (6.5)$$

Pokud uvážíme, že $\sum_{i=0}^{|E_1|} T_i^{M1} = |I|$, pak lze předchozí vztah rozepsat jako:

$$\sum_{i=0}^{|E_1|}(T_i^{M1}+1) = \sum_{i=0}^{|E_1|} T_i^{M1} + \sum_{i=0}^{|E_1|} 1 = |I| + |E_1| \qquad (6.6)$$

Totéž platí pro hrany modelu M_2, tj. označíme-li T_i^{M2} počet průsečíků hrany e_i modelu M_2, pak celkový počet hran vzniklých z dělení hran modelu M_2 je:

$$\sum_{i=0}^{|E_2|}(T_i^{M2}+1) = \sum_{i=0}^{|E_2|} T_i^{M2} + \sum_{i=0}^{|E_2|} 1 = |I| + |E_2| \qquad (6.7)$$

Celkový počet hran E_{tot} vzniklých dělením v průsečících pak bude:

$$E_{tot} = 2|I| + |E_1| + |E_2| \qquad (6.8)$$

Toto však není celkový počet hran supermeshe, neboť jsou ještě přidány další hrany ve fázi triangulace. Označíme-li počet hran vzniklých při triangularizaci $|E_{tr}|$, pak lze dosadit do vztahu 6.3 pro výpočet celkového počtu trojúhelníků supermeshe.

$$F = 2 - (|V_1| + |V_2| + |I|) + 2|I| + |E_1| + |E_2| + |E_{tr}| = 2 - |V_1| - |V_2| + |I| + |E_1| + |E_2| + |E_{tr}| \qquad (6.9)$$

Opět lze využít Eulerova vztahu $F_1 = 2 - V_1 + E_1$, resp. $F_2 = 2 - V_2 + E_2$ a dosadit do vztahu 6.3.

$$F = |F_1| + |F_2| - 2 + |I| + |E_{tr}| \qquad (6.10)$$

Z předchozího vztahu vyplývá, že počet plošek supermeshe závisí na počtu plošek obou zdrojových těles, na počtu průsečíků a na počtu hran, které vzniknou během triangularizace. Počet průsečíků je složité odhadnout, neboť závisí na konkrétní topologii a geometrii zdrojových objektů. Je třeba zdůraznit, že se jedná o horní odhad, neboť je založen na předpokladu 6.4. Skutečný počet plošek supermeshe bývá obvykle menší (díky singulárním případům), rozhodně by však neměl překročit mez danou vztahem 6.10.

Z tabulky 6-2 je dále možné vidět, že některé výsledné supermeshe nesplňují přesně Eulerovu formuli pro jednoduchý mnohostěn. Pro testovací množinu je odchylka počtu plošek supermeshe od počtu plošek daných Eulerovou formulí pro hrany a vrcholy (tj. F = 2-V+E) řádově v jednotkách. Tato odchylka může být způsobena:

- chybným umístněním středu projekce tak, že neleží v jádře mnohostěnu (tj. není star-pointem)[21],
- numerickou chybou, která většinou plyne z koincidencí průsečíků.

[21] Diskuse výsledků v případě, že těleso nesplňuje přesně požadavky na star-shaped tělesa bude uvedena dále.

Dalším údajem v tabulce je počet průsečíků, který by měl být pro oba směry morphingu stejný. V posledních dvou případech (7 → 8 a 8 → 7) tomu však tak není. Toto opět plyne z chybného umístnění středu projekce. Chybné umístnění středu projekce znamená nejednoznačnost v parametrické doméně (viz oddíl 6.4.). Algoritmus hledání průsečíků „procházka“ pak nemusí objevit všechny průsečíky jako je tomu v případě 7 → 8 a 8 → 7. Pokud však použijeme na výpočet průsečíků algoritmus brutální síly, pak v obou případech dojdeme ke stejnému počtu průsečíků.

Poslední údaj v tabulce udává počet hran, které bylo třeba vložit při triangularizaci supermeshe.

Při výpočtu morphingu nemá smysl měřit celkový čas v závislosti na počtu vrcholů, hran, případně plošek zdrojových těles, neboť záleží na jejich vzájemné poloze, resp. na počtu průsečíků. Nicméně z tabulky 6-2 lze usoudit, že celkový čas není kvadratický vzhledem k počtu vrcholů, hran ani plošek.

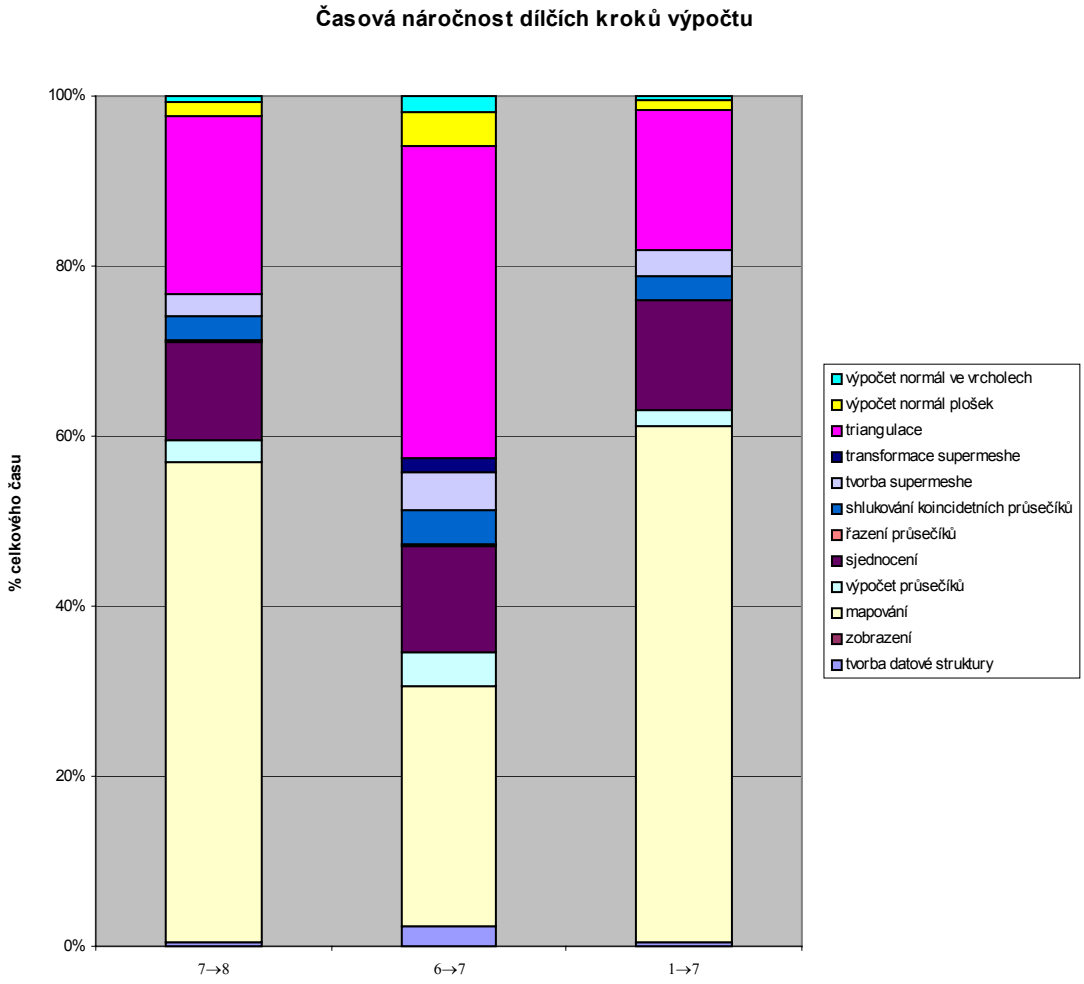
6.3. Měření časů dílčích kroků výpočtu morphingu

Z analýzy složitosti provedené v předchozím oddílu lze usuzovat, jaký čas spotřebují konkrétní kroky výpočtu. V následujících tabulkách bude uvedeno měření pro tři případy. Kroky jsou v tabulce stručně pojmenovány a odpovídají krokům, které byly zkoumány v analýze složitosti. Každé měření probíhalo třikrát, přičemž údaje v tabulce jsou průměrem jednotlivých měření.

	7→ 8		6 → 7		1 → 7	
	čas [ms]	**% celkového času**	**čas [ms]**	**% celkového času**	**čas [ms]**	**% celkového času**
tvorba datové struktury	150	0,56	82	2,27	230	0,40
zobrazení	5	0,02	3	0,08	8	0,01
mapování	15121	56,42	1024	28,29	35040	60,83
výpočet průsečíků	663	2,47	145	4,01	1041	1,81
sjednocení	3129	11,67	447	12,35	7497	13,02
řazení průsečíků	18	0,07	10	0,28	20	0,03
shlukování koincidetních průsečíků	757	2,82	143	3,95	1569	2,72
tvorba supermeshe	723	2,70	165	4,56	1802	3,13
transformace supermeshe	5	0,02	62	1,71	8	0,01
triangulace	5616	20,95	1327	36,66	9463	16,43
výpočet normál plošek	410	1,53	142	3,92	624	1,08
výpočet normál ve vrcholech	206	0,77	70	1,93	297	0,52
CELKOVÝ ČAS	**26803**	**100,00**	**3620**	**100,00**	**57599**	**100,00**

Tabulka 6-3: Dílčí časy jednotlivých fází výpočtu

Pro lepší představu byly údaje z tabulky s absolutními časy zobrazeny do grafů.



Obrázek 6-1: Graf časové náročnosti dílčích kroků výpočtu

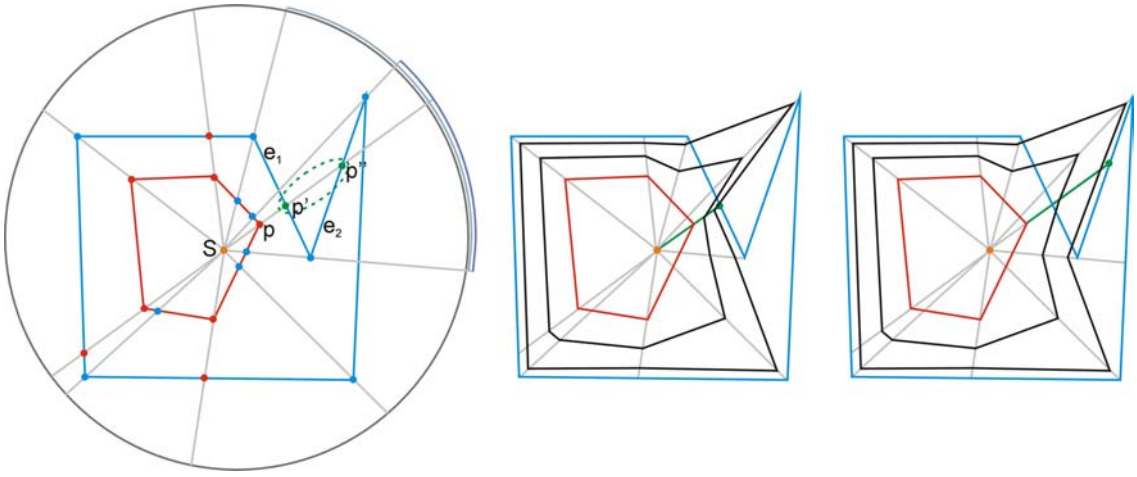
Zejména z grafu je názorně vidět, které kroky zabírají nejvíce času. Podle očekávání je to výpočet mapování vrcholů výchozího objektu na povrch cílového objektu. Podle analýzy je toto krok se složitostí $O(N^2)$. Dále je vidět, že na první pohled kritický krok řazení průsečíků algoritmem bubble-sort spotřebovává minimální podíl celkového času výpočtu.

Poměrně velkou část celkového času spotřebovává sjednocení datových struktur, ačkoli se jedná o prosté kopírování datových struktur. Řešením je použít lepší metody pro kopírování, například přesun celých bloků paměti.

6.4. Non star-shaped tělesa

Nyní se budeme zabývat případy, kdy těleso nebude přesně star-shaped, tj. existují vrcholy modelu tělesa, které nejsou viditelné ze star-pointu. Takových vrcholů by ovšem mělo být ve srovnání s celkovým počtem vrcholů modelu podstatně méně. Konkrétním příkladem tělesa, které nesplňuje přesně požadavky star-shaped těles, je těleso face4.asc. Toto těleso znázorňuje hlavu panáčka, kde v oblasti úst jsou vrcholy, které nejsou vidět z implicitního star-pointu (tj. z těžiště tělesa). Nicméně na výsledném

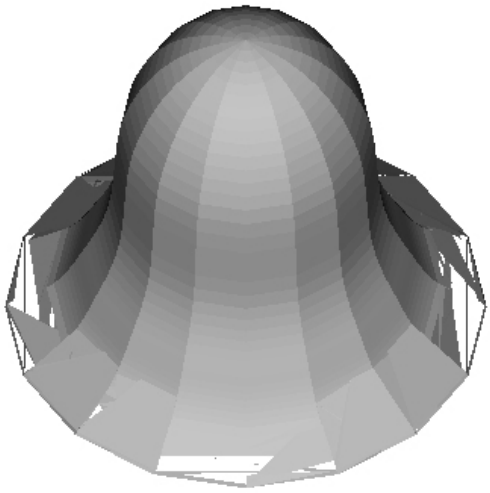
supermeshi tato chyba není takřka k poznání. Rozeberme důkladně, co přesně nastane. Pro rozbor dobře poslouží následující obrázek. Obrázek demonstruje situaci pro názornost ve 2D, situace ve 3D je obdobná. Ve 2D platí stejné podmínky jako ve 3D, objekty jsou promítány přes star-point na jednotkovou kružnici. Místo sférických trojúhelníků máme ve 2D hrany, které přejdou do oblouků (oblouky jsou částmi jednotkové kružnice).



a) b) c)

Obrázek 6-2: Znázornění situace, kdy těleso nesplňuje přesně podmínky star-shaped těles

Na obrázku 6-2 a) je zobrazení modelů do společné parametrické domény. Model M_2 (znázorněn modře) však nesplňuje podmínky star-shaped vůči bodu S. Problémy působí hrany e_1 a e_2, které se na společné parametrické doméně překrývají. V případě 2D dochází tedy k překrývání oblouků na parametrické doméně, v 3D případě pak by toto znamenalo překrývání sférických trojúhelníků na parametrické doméně. V tomto případě již nelze mluvit o zobrazení, neboť je porušena podmínka, že každému bodu x (vzor) je přiřazen právě jeden bod y (obraz). Důsledkem toho je, že ve fázi hledání mapování vrcholů modelu M_1 na povrch modelu M_2 není mapování jednoznačné. V obrázku 6-2 a) může být bod p namapován na hranu e_1 nebo na hranu e_2. Bude-li namapován na hranu e_1, pak výsledná trajektorie vrcholu při animaci bude po úsečce pp' (předpokládáme-li lineární interpolaci). Náznak takové animace je na obrázku 6-2 b), kde je zeleně vyznačena trajektorie vrcholu p. Naopak, namapuje-li se vrchol p na hranu e_2, pak jeho trajektorie bude dána úsečkou pp''. Tento případ ukazuje obrázek 6-2 c). V obou případech je vidět, že výsledný tvar rozhodně nebude stejný jako tvar M_2, oblast nesplňující podmínky star-shaped bude „překlenuta“. Ve 3D případě navíc toto způsobuje problémy s triangularizací a výsledkem toho je, že některé trojúhelníky z částí nesplňujících podmínky star-shped vypadnou. Příklad „vypadnutí“ trojúhelníků lze pozorovat na obrázku 6-3.



Obrázek 6-3: Příklad „vypadnutí“ některých trojúhelníků, které je způsobeno chybným umístněním středu projekce

7. Závěr

Morphing patří mezi pokročilejší animační techniky. Způsob řešení morphingu v profesionálních animačních nástrojích však neodpovídá aktuálním řešením, která fungují podstatně lépe. Přitom jeden ze stěžejních článků zabývající se řešením morphingu ([3]) byl prezentován na Siggraphu již v roce 1992. Jednou z možných příčin je, že ačkoli bylo navrženo a realizováno množství technik, žádná z nich dosud nefunguje zcela univerzálně a často se omezuje pouze na jistou třídu těles.

V teoretickém úvodu práce bylo popsán základní postup při morphingu 3D geometrických objektů daných hraniční reprezentací. Jedná se o poměrně úzkou oblast, neboť existuje řada technik pracujících s jinou reprezentací těles (objemovou, implicitní). Hraniční reprezentace je však stále pro modelování a animaci poměrně hojně využívaná zejména z důvodů efektivního uložení, zobrazování a editace.

Práce se omezuje pouze na star-shaped tělesa, a to z důvodů použité metody zobrazení. Nicméně pro star-shaped objekty produkuje realistické přechody a funguje zcela automaticky. V některých případech je pouze třeba upravit pozici star-pointu, neboť star-point je aproximován těžištěm. Metoda funguje poměrně rychle, nicméně rychlost zatím není stěžejním hlediskem hodnocení metod morphingu, hlavním hlediskem je věrohodnost, resp. estetičnost animace. Rovněž se nepředpokládá, že by byly morphovány rozsáhlé datové množiny. Morphing je zejména technika využívaná animátory, kteří obvykle nepracují s extrémně složitými modely. Modely sice mohou pocházet z různých 3D scannerů, ale nejen pro účely morphingu jsou složité modely často zjednodušovány a případné detaily jsou simulovány renderovacími metodami.

V implementaci bylo vytvořeno jednoduché uživatelské rozhraní, které umožňuje načítat zdrojová tělesa, zobrazovat a ukládat jednotlivé fáze animace. Testovací tělesa byla připravena pomocí nástroje 3ds max, nicméně program by měl pracovat s libovolnými jinými tělesy splňujícími podmínky star-shaped těles.

Ve srovnání se současnými metodami je zde prezentovaná metoda kdesi na počátku, nicméně studium existujících metod přineslo poměrně jasnou představu o dalším postupu.

Další práce by tedy měla jistě směřovat k rozšíření třídy zdrojových těles alespoň na genus 0 tělesa. K tomu by měly posloužit metody zobrazení do společné parametrické domény popsané v teoretickém rozboru. Dalším krokem je aplikace morphingu nejenom na tvary objektů, ale i na vlastnosti povrchu, jako například barva, průhlednost, texturové souřadnice atd. Pro praktické využití výstupů (tj. animací) by bylo rovněž vhodné integrovat tento program jako modul (plug-in) do nějakého existujícího animačního a modelovacího nástroje. Případně jako samostatný modul do nějaké vizualizační knihovny typu VTK[22].

[22] VTK – Visualization Toolkit. Sada knihoven pro vizualizaci a zpracování grafických dat. Viz [URL3].

7.1. Osobní zhodnocení

Osobně mě toto téma velmi zajímalo. Ačkoli v době zadávání tématu jsem měl takřka nulové informace o této problematice, lákaly mě zejména případné grafické výstupy, které jsem zatím sledoval pouze ve filmovém nebo herním průmyslu. Další motivací byla zkušenost s touto technikou v animačním a modelovacím nástroji 3ds max r3, kde jsou její možnosti poměrně omezené. Podařilo se vytvořit jednoduchý systém, který zatím dobře zvládá pouze star shaped objekty, nicméně myslím, že je dobrou výchozí cestou pro další experimenty a studium.

Obtíže zpočátku kladly anglické články, které se mi jevily dosti komplikované. Množství kroků bylo pochopeno až po naprogramování a zobrazení dílčích výsledků. Nicméně nyní jsou všechny kroky poměrně jasné a je možné se soustředit na nějaký významnější vlastní přínos do problematiky morphingu.

Literatura

Seznam použité literatury

[1] Alexa, M.: Merging Polyhedral Shapes with Scattered Features, The Visual Computer, Vol.16, No. 1, str. 26-37, 2000.
[2] Alexa, M.: Mesh Morphing (State of The Art Report), Eurographics, 2001.
[3] Kent, J. R., Carlson, W. E., Parent R. E.: Shape Transformation for Polyhedral Objects, Computer Graphics (Proceedings of SIGGRAPH 92), Vol. 26, No. 2, str. 47-54, 1992.
[4] Kanai, T., Suzuki, H., Kimura, F.: 3D Geometric Metamorphosis based on Harmonic Map, Pacific Graphics ’97, 1997.
[5] Lazarus, F., Verroust, A.: Three-dimensional metamorphosis: a survey, The Visual Computer, Vol. 14, str. 373-389, 1998.
[6] Dharmarante, A., Harada, K.: Vertex Correspondence for Polygon Morphing, EAST-WEST-VISION International Workshop & Project Festival, str. 155-160, 2002.
[7] Sedeberg, W. T., Greenwood, E.: A Physically Based Approach to 2-D Shape Blending, SIGGRAPH '92, Vol. 26, str. 25-34, 1992.
[8] Shapira, M., Rappoport, A.: Shape Blending Using the Star-Skeleton Representation, IEEE Computer Graphics & Applications, Vol. 15, No. 2, str. 44-50, 1995.
[9] Váša, L.: Normály ve vrcholech trojúhelníkových sítí (semestrální práce z předmětu KIV/APG), 2002.
[10] Erikson, C.: Morphing Three Dimensional Polyhedral Objects, Honors Papers, 1994.
[11] Lazarus, F., Verroust, A.: Featured-based shape transformation for polyhedral objects, Technical report, No. 2264, Institut national de recherche en informatique et automatique, 1994.

Seznam publikací

[12] Parus, J., Kolingerová, I.: Mesh Morphing, Proceedings of the 7th Central European Seminar on Computer Graphics, 2003 (poster).
[13] Parus, J., Kolingerová, I.: Mesh Morphing, Student EEICT, 2003 (zasláno k publikaci).

Odkazy

[URL1] www.algorithmic-solutions.info/leda_guide/graph_algorithms/spring_embedding.html
[URL2] herakles.zcu.cz/research/mveruntime/mve/developer/tri10.php
[URL3] www.kitware.com

Příloha A – programová dokumentace

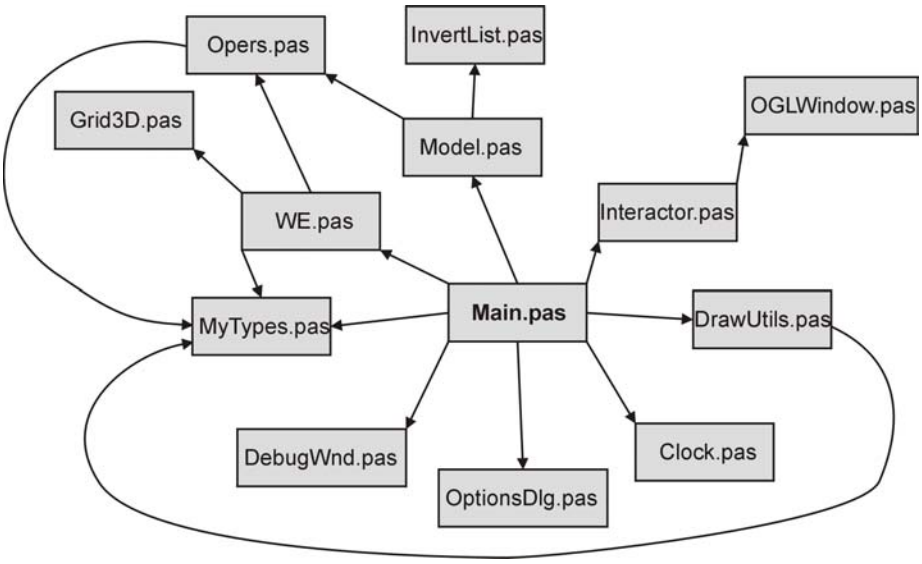
V této příloze bude stručně popsána struktura programu, tj. jednotlivé moduly a jejich provázání.

Program se skládá z následujících modulů:

Název modulu	Popis
OGLWindow.pas	Okno zobrazující výstup OpenGL. V třídě TOGLWindow jsou základní metody pro inicializaci OpenGL a nastavení renderovacího kontextu.
Interactor.pas	Implementace třídy odděděné od TOGWindow umožňující provádění pohledových transformací.
Main.pas	Hlavní jednotka programu. Hlavní okno aplikace je odděděno od třídy TInteractor, přičemž je dospáno tělo abstraktní metody Draw(), která zajišťuje vykreslování scény do okna.
Model.pas	Implementace třídy TModel pro práci s geometrickými objekty. Třída obsahuje metody pro načítání, tvorbu display listů, výpočet normál a transformaci modelů.
InvertList.pas	Pomocná třída pro tvorbu invertovaného seznamu.
Opers.pas	Obsahuje pomocné funkce pro operace s vektory a maticemi, funkce na výpočet barycentrických souřadnic a průsečíků oblouků a rutiny pro alokaci, realokaci a dealokaci dynamické paměti.
WE.pas	Jednotka pro práci s datovou strukturou okřídlená hrana a pro výpočet jednotlivých dílčích kroků morphingu.
MyTypes.pas	Definice základních datových typů a konstant.
Clock.pas	Implementace třídy TClock pro měření časových intervalů.
OptionsDlg.pas	Dialogové okno sloužící pro nastavení nejrůznějších parametrů výpočtu. Obsahuje pouze reakce na události pocházející od ovládacích prvků uživatelského rozhraní.
DrawUtils.pas	Pomocné funkce na vykreslování reprezentace bodů, oblouků, úseček atd.
Grid3D.pas	Třída pro práci s pomocnou vyhledávací datovou strukturou trojrozměrná mřížka. Obsahuje metody pro alokaci, naplnění a vyhledávání v mřížce.
DebugWnd.pas	Zobrazování ladících výpisů v separátním okně.

Tabulka A-1: Seznam modulů programu

Struktura programu je graficky znázorněna na obrázku A-1. Na obrázku nejsou zachyceny systémové jednotky Delphi.



Obrázek A-1: Struktura programu

Příloha B – uživatelská dokumentace

Instalace programu

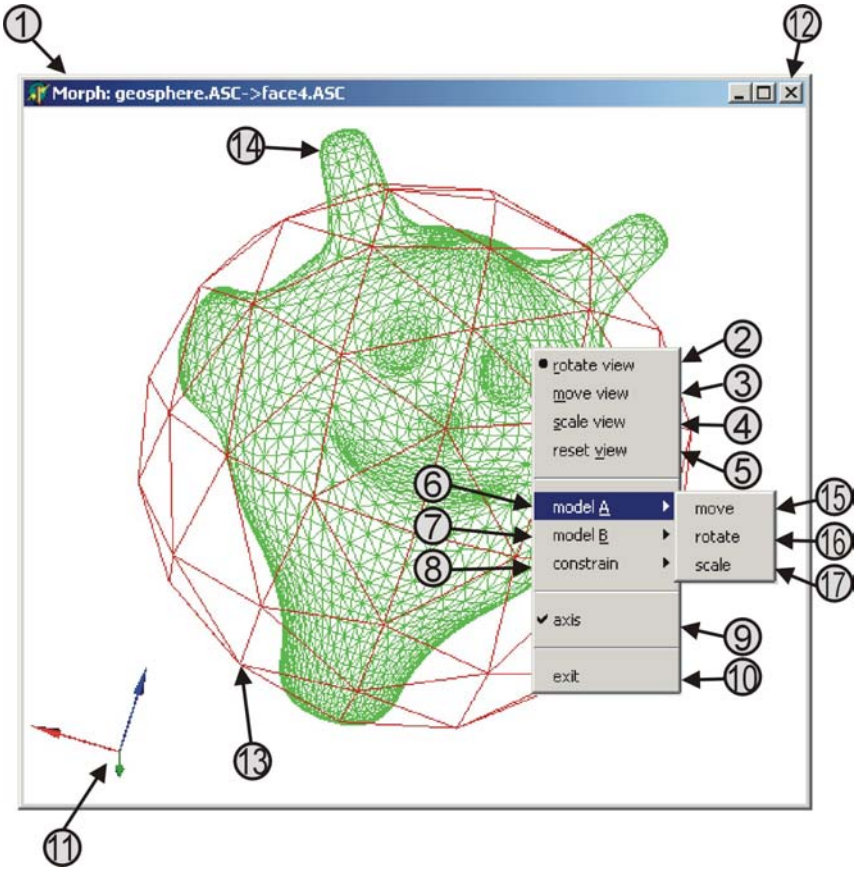
Program nevyžaduje žádnou speciální instalaci. Program může být spouštěn z CD, případně je možné program nakopírovat na pevný disk, přičemž je třeba zachovat adresářovou strukturu.

Ovládání programu

Po spuštění programu se objeví dvě okna. Okno renderu a okno sloužící pro nastavení různých parametrů výpočtu. Nejdříve bude popsáno okno rendereru.

Okno rendereru

Okno rendereru zobrazuje grafické výstupy výpočtu. Obsahuje kontextové menu (přístupné přes pravé tlačítko myši), pomocí kterého lze nastavovat pohledové transformace a transformace objektů. Na následujícím obrázku je okno rendereru společně s kontextovým menu, čísly jsou označeny jednotlivé prvky, které budou dále popsány.



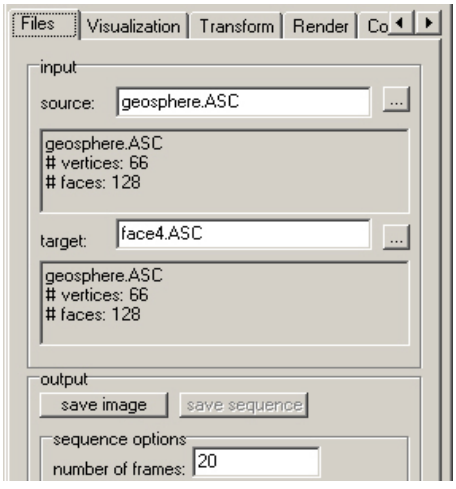
Obrázek B-1: Okno rendereru

Číslo	Popis
1	Titulek okna. Obsahuje jméno souboru výchozího a cílového objektu.
2	Rotace pohledu. Rotace probíhá standardním způsobem myší.
3	Posun pohledu.
4	Přiblížení nebo oddálení pohledu.
5	Návrat k původnímu pohledu
6	Menu pro transformaci výchozího modelu. Umožňuje měnit polohu výchozího objektu vzhledem počátku souřadné soustavy, kde se nachází star-point.
7	Menu pro transformaci cílového modelu. Umožňuje měnit polohu cílové objektu vzhledem počátku souřadné soustavy, kde se nachází star-point (stejné podmenu jako u 6).
8	Nastavení omezení transformace modelů v souřadných osách. Je výhodné používat při drobných úpravách polohy tělesa. Lze se omezit pouze na jednu osu (x, y, z) nebo na rovinu (xy, xz, yz).
9	Zapínání a vypínání znázornění orientace souřadného systému (11).
10	Ukončení aplikace.
11	Vizualizace orientace souřadného systému. Červeně je znázorněna osa x, zeleně osa y a modře osa z.
12	Uzavřením okna se ukončí aplikace.
13	Zobrazení výchozího objektu.
14	Zobrazení cílového objektu.
15	Posun výchozího modelu.
16	Rotace výchozího modelu.
17	Změna měřítka výchozího modelu.

Okno pro nastavení parametrů výpočtu

V tomto okně je možné nastavit načítat zdrojové soubory, nastavovat parametry zobrazování a parametry výpočtu. Okno obsahuje seznam karet, které je možné aktivovat volbou příslušné záložky.

Záložka Files

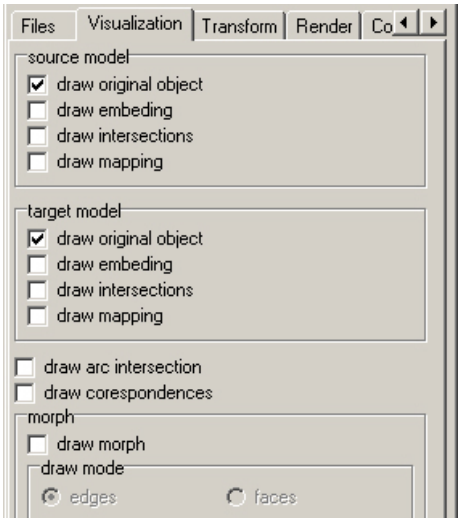


Tato záložka slouží pro načítání zdrojových objektů. Po spuštění programu jsou implicitně načteny dva zdrojové objekty. Načtení jiných objektů je možné pomocí tlačítka ... Je možné načítat soubory ve formátu .TRI a .ASC. TRI[23] je textový formát používaný systémem MVE, ASC je textový exportní formát 3ds max. Po načtení objektu jsou dále zobrazeny informace o počtu vrcholů a plošek. Dále je možné tlačítkem **save image** uložit aktuální obrázek zobrazovaný v rendereru ve formátu BMP. Pokud je vypočítána nějaká animace, pak je možné tlačítkem **save sequence** uložit posloupnost obrázků

[23] Viz [URL2].

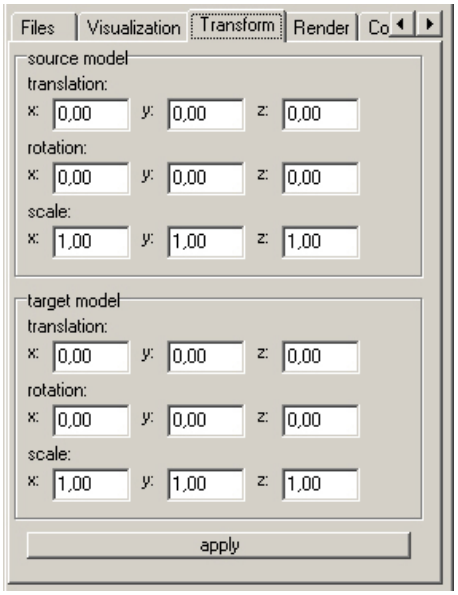
reprezentujících animaci, přičemž lze upravit počet snímků animace. Posloupnost obrázků lze pak sloučit externím programem pjBmp2Avi, který je rovněž dodán na přiloženém CD.

Záložka Vizualization



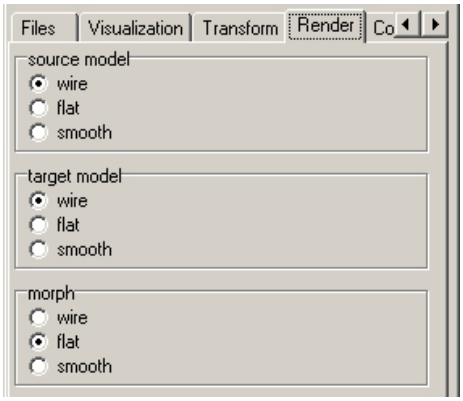
Na této záložce je možné nastavovat různé volby týkající se zobrazování v okně rendereru. Pro oba zdrojové modely je možné zobrazit jejich zobrazení na kouli (**draw embedding**), projekce průsečíků z param. domény zpět hrany (**draw intersections**) a mapování vrcholů na povrch druhého objektu (**draw mapping**). Dále je možné ukázat průsečíky v zobrazení (**draw arc intersections**) a schématicky znázornit korespondenci vrcholů (**draw corespondence**). Pokud je vypočítána animace morphingu, pak je možné zaškrtnout **draw morph** a vybrat, zda se budou vykreslovat pouze hrany (**edges**) nebo plošky (**faces**). V případě, že jsou vybrány plošky, objekt je stínován.

Záložka Transform



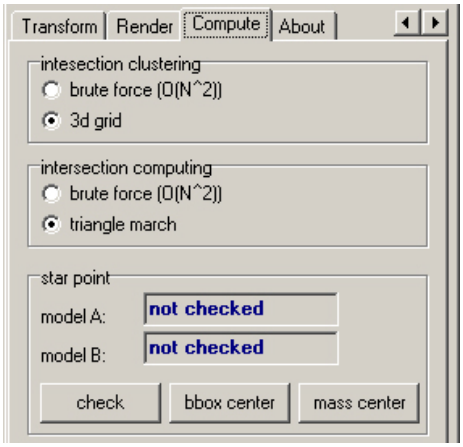
Zde je možné transformovat zdrojové objekty. Umožňuje nastavovat polohu (**translation**), rotaci okolo os souřadného systému (**rotation**) a změnu měřítka (**scale**). Po nastavení příslušných políček je třeba provést změny stisknutím tlačítka **apply**. Transformace je možné rovněž provádět myší po výběru příslušné položky z kontextového menu. Pokud se transofrmace provádějí myší je možné na záložce **Transform** sledovat číselné vyjádření transformací.

Záložka Render



Tato záložka dovoluje volit způsob vykreslování zdrojových objektů a morphovaného objektu. Je možné nastavit drátěné vykreslování (**wire**), konstantní stínování (**flat**) a Gouraudovo stínování (**smooth**).

Záložka Compute



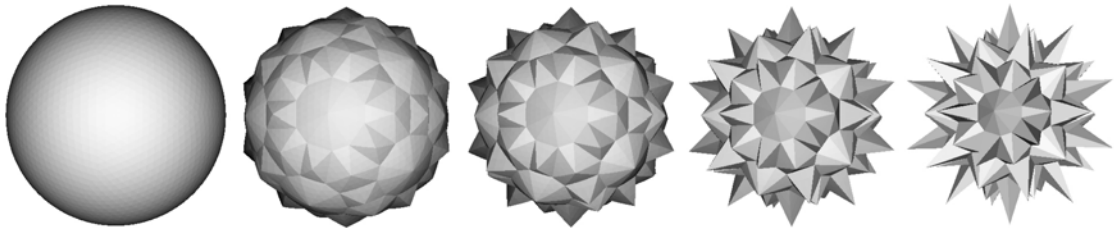
Záložka **Compute** slouží pro nastavení některých voleb výpočtu. Je možné vybrat algoritmus pro shlukování průsečíků (**intersection clustering**). K dispozici jsou dvě varianty – brutální síla (**brute force**) a trojrozměrná mřížka (**3d grid**). Podobně je možné vybrat jakým způsobem počítat průsečíky (**intersection computing**). Je třeba připomenout, že volby brutální síly jsou značně časově náročné.
Dále je možné kontrolovat umístnění star-pointu (tlačítko **check**) a nastavovat star-point do středu ohraničujícího kvádru (**bbox center**) a do těžiště (**mass center**).

Pod seznamem karet se nachází tlačítko **compute**, které spustí výpočet morphingu. Průběžné informace o výpočtu jsou zobrazovány v stavovém řádku. Po proběhnutí výpočtu je možné pomocí scroll-baru prohlížet jednotlivé fáze animace. Animaci je rovněž možné ovládat tlačítky  se standardním významem.

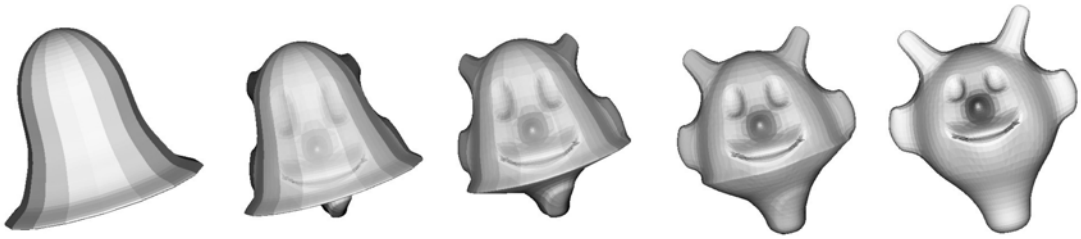
Příloha C – ukázky výstupů

V této příloze budou uvedeny ukázky výstupu programového vybavení. Obrázky jsou stínovány konstantním stínováním, aby bylo možné sledovat topologii jednotlivých fází. Jednotlivé příklady budou popsány jmény zdrojových objektů, které je možné nalézt na přiloženém CD. Složitost některých objektů je možné nalézt v tabulce 6-1.

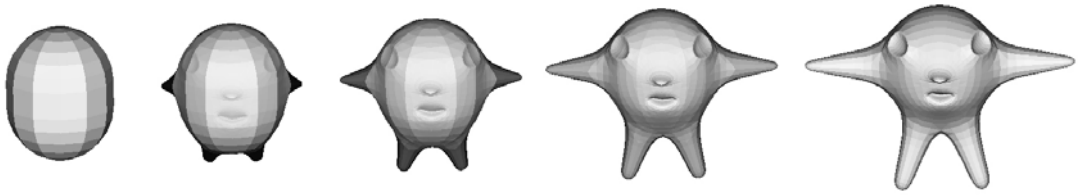
- **geosphere-high res.asc → star1.asc**



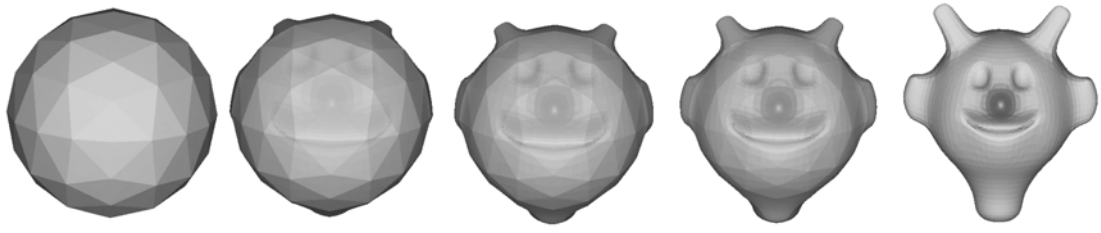
- **bell.asc → face4.asc**



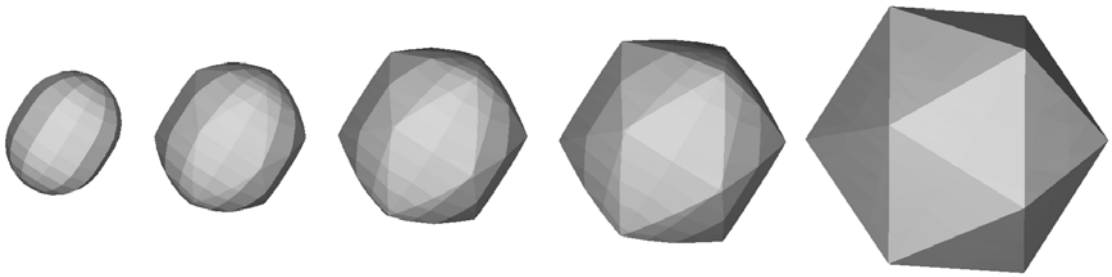
- **capsule.asc → pokemon.asc**



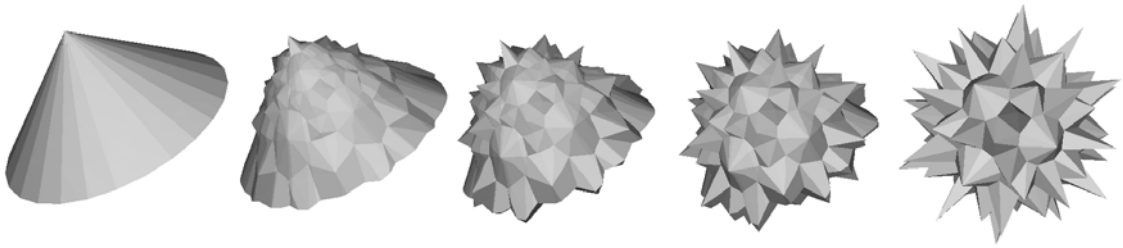
- **geosphere.asc → face4.asc**



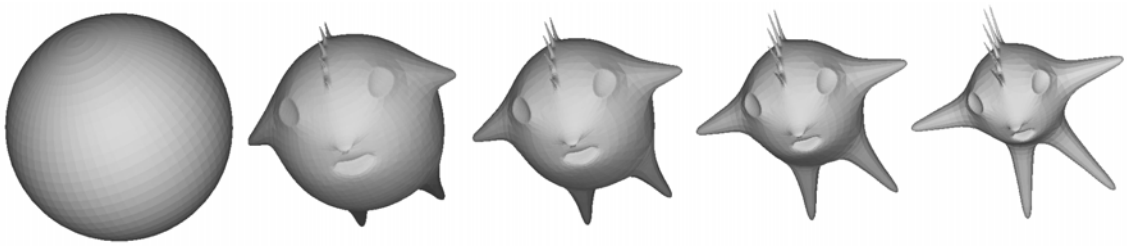
- **capsule.asc → dodeca.asc**



- **cone.asc → star1.asc**



- **geosphere-high res.asc → pokemon2.asc**



Evidenční list

Souhlasím s tím, aby má diplomová práce byla půjčována k prezenčnímu studiu v Univerzitní knihovně ZČU v Plzni.

Datum: Podpis:

Uživatel stvrzuje svým čitelným podpisem, že tuto diplomovou práci použil pouze ke studijním účelům a prohlašuje, že ji uvede mezi použitými prameny.

Jméno	**Fakulta / katedra**	**Datum**	**Podpis**