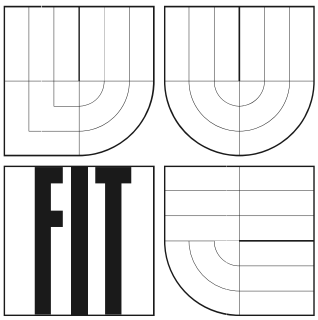
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Bakalářská práce

2007 Jan Pokorný

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pečivy.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Na tomto místě bych chtěl také poděkovat Ing. Janu Pečivovi za odbornou pomoc při řešení nejrůznějších problémů.

Abstrakt

Tato práce se věnuje algoritmům pro redukci polygonálních modelů. Nejdříve jsou popsány obecné principy zjednodušování modelů – jsou zde vysvětleny vlastnosti polygonálních povrchů a operace používané k redukci počtu trojúhelníků.
V dalších kapitolách jsou popsány tři metody redukce počtu trojúhelníků – Uniform Vertex Clustering, Floating Cell Vertex Clustering a Vertex Decimation. Tyto metody jsou testovány na několika modelech a porovnávány podle nejrůznějších hledisek – vizuální kvalita, časová náročnost a geometrická chyba aproximace.

Klíčová slova

Zjednodušování polygonálních modelů, LOD, decimace vertexů, shlukování vertexů, geometrická chyba aproximace

Abstract

This paper is dedicated to simlification algorithms of polygonal models. First chapter deals with common principles of model simplification – features of polygonal models are explained here along with operations used in simlification.
Next chapters introduce three simplification algorithms – Uniform Vertex Clustering, Floating Cell Vertex Clustering and Vertex Decimation.
These methods are tested on a number of models and compared in various aspects – visual quality, aproximation time and geometric error of aproximation.

Keywords

Polygonal model simplification, LOD, vertex clustering, vertex decimation, geometric error of aproximation

Obsah

1 Úvod

Ve světě počítačové grafiky se vždy muselo rozhodovat mezi kvalitou zobrazení a rychlostí zobrazení. Za účelem řešení tohoto problému byla v počítačové grafice zavedena samostatná disciplína jménem LOD (*Level of detail*) - úroveň detailu zobrazení.

Přestože se grafický hardware neustále vyvíjí dokáže pracovat s velkým množstvím polygonů, stále je mnoho aplikací, kde má tato disciplína své uplatnění.

Existují totiž zdroje, které produkují modely s velmi vysokým počtem polygonů a právě tyto jsou nejčastějším cílem algoritmů pro zjednodušování polygonálních modelů (*simlification*).

Zdroji takových polygonálních modelů jsou například :

- 3D skenery, systémy počítačového vidění a lékařská zobrazovací zařízení, která dokáží vytvářet modely reálných objektů
- CAD systémy produkující obvykle velmi detailní a složité modely.
- Metody pro rekonstrukci povrchu nebo extrakci izoploch, jejichž výstupem bývají často modely s velmi hustou polygonální sítí s pravidelným uspořádáním jejích vrcholů.

Modely vyprodukované těmito systémy mají obvykle vysoký počet polygonů, které jsou často rovnoměrně rozloženy (vzdálenosti mezi vertexy jsou velmi podobné) a to jak v oblastech plochých, tak i zakřivených.

Metody redukce pak často vedou na jiné, úspornější rozložení, které dává v důsledku model s menším počtem polygonů a vzhledem velmi podobným originálu.

Cílem této práce je implementovat různé metody pro zjednodušování polygonálních modelů a porovnat jejich výstupy.

1.1 Rozvržení technické zprávy

Tato technická zpráva je rozčleněna do sedmi kapitol. Po úvodní kapitole následuje část popisující obecné principy zjednodušování modelů – jsou zde vysvětleny vlastnosti polygonálních povrchů a operace používané k redukci počtu trojúhelníků. Kapitola tři popisuje metodu *Uniform Vertex Clustering*. Ve čtvrté kapitole je popsána metoda tuto metodu značně vylepšující – *Floating Cell Vertex Clustering*. Dále je uvedena metoda pracujícím na zcela jiném principu než předchozí metody – *Vertex Decimation*. V šesté kapitole je uvedeno srovnání všech tří metod a to jak vizuálně, tak numerickým srovnáním výstupu (geometrická chyba) a srovnáním časové náročnosti jednotlivých metod. Kapitola číslo sedm uzavírá celou práci zhodnocením dosažených výsledků a možnostmi vylepšení této práce do budoucna.

2 Obecný pohled na zjednodušování polygonálních modelů

V této kapitole jsou uvedeny základní informace o polygonálních modelech a principy jejich zjednodušování ve smyslu zmenšování počtu polygonů při zachování vzhledu objektu.
V dalším textu budeme předpokládat, že polygonální modely jsou na vstupu i výstupu zjednodušování tvořeny výlučně trojúhelníky. Modely mohou být samozřejmě tvořeny obecnými polygony (čtyřúhelníky, pětiúhelníky, atd.), ale pro většinu algoritmů určených k jejich zjednodušení se tyto polygony triangulují (převádějí na trojúhelníky). V této práci jsou tedy pojmy *polygon* a *trojúhelník* zaměnitelné.

2.1 Polygonální reprezentace modelů

Polygonální reprezentace je pravděpodobně nejčastěji používanou reprezentací objektů v počítačové grafice. Základním prvkem této reprezentace je trojúhelník, i když jsou podporovány i čtyřúhelníky a mnohoúhelníky. U mnohoúhelníků je však při zobrazování nutné určovat jejich konvexitu, což představuje určitou výpočetní zátěž. Tento problém u trojúhelníků nenastává.
Velkou výhodou polygonální reprezentace je, že zobrazování těchto modelů je v dnešní době prakticky vždy podporováno hardwarem počítače – grafickou kartou.

2.1.1 Uspořádání v polygonální reprezentaci

Model v polygonálním tvaru je obvykle určen seznamem vrcholů (vertexů) a seznamem trojúhelníků (případně i normál). Každý vertex je tvořen trojicí hodnot, které udávají jeho polohu v Eukleidovském prostoru R3. Každý trojúhelník je potom tvořen trojicí indexů v seznamu vertexů – každý index se odkazuje na jeden ze tří vertexů tvořící trojúhelník.
Fakt, že trojúhelníky jsou tvořeny třemi vrcholy je v geometrii obecně známý, ale v polygonální reprezentaci modelu bývá často důležité i uspořádání těchto vrcholů. Trojúhelník je plocha a jako takový má svůj normálový vektor. Tento vektor často z úsporných důvodů nebývá uložen spolu s modelem ale je dopočítán později. A právě uspořádání vertexů tvořících trojúhelník pak udává směr tohoto normálového vektoru. Nejčastější forma uspořádání vertexů je proti směru hodinových ručiček, ale uspořádání může být i opačné. Každopádně je nutné tuto uspořádání znát, aby bylo zajištěno správné zobrazení modelu.

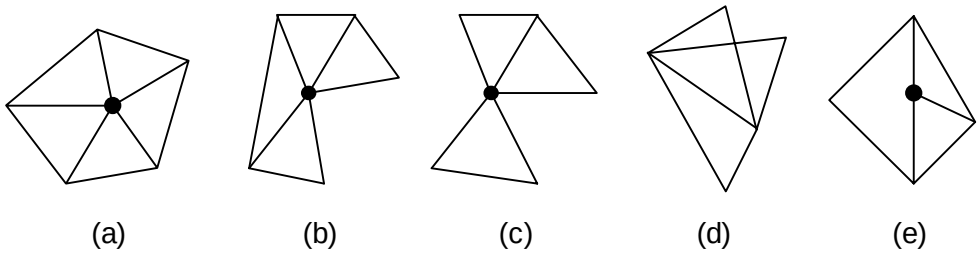
2.1.2 Topologie povrchu

V kontextu polygonální reprezentace modelů je topologie povrchu dána strukturou spojené polygonální sítě. Třída (genus) polygonální sítě udává počet děr, které tato síť obsahuje. Uzavřené objekty, jako koule nebo krychle mají genus rovný nule.

Lokální topologie sítě je pak určena konektivitou v určité oblasti sítě – okolí každého vertexu – tuto oblast budeme dále nazývat *trsem*. Trs trojúhelníků kolem určitého vertexu tvoří *všechny* trojúhelníky, které tento vertex obsahují.

Pokud je v oblasti sítě úplná konektivita, pak všechny trojúhelníky v trsu tvoří disk – tedy každý trojúhelník, který tento vertex obsahuje, sdílí obě hrany s *právě dvěma* jinými trojúhelníky trsu. V síti však mohou existovat i trsy, které tvoří neúplný disk – existují zde *právě dva* trojúhelníky, které mají jednu hranu nesílenou. Takové vertexy pak tvoří tzv. hraniční vertexy sítě *(boundary*). Objekty, které jsou tvořeny disky a hraničními vertexy pak nazýváme *manifold* (vyrobitelné). Naopak objekty, které tyto vlastnosti nemají, jsou nazývány *non-manifold* (nevyrobitelné).

Příklady manifold (a, b) a non-manifold (c, d, e) oblastí sítě jsou ukázány na *Obrázku 2.1*.

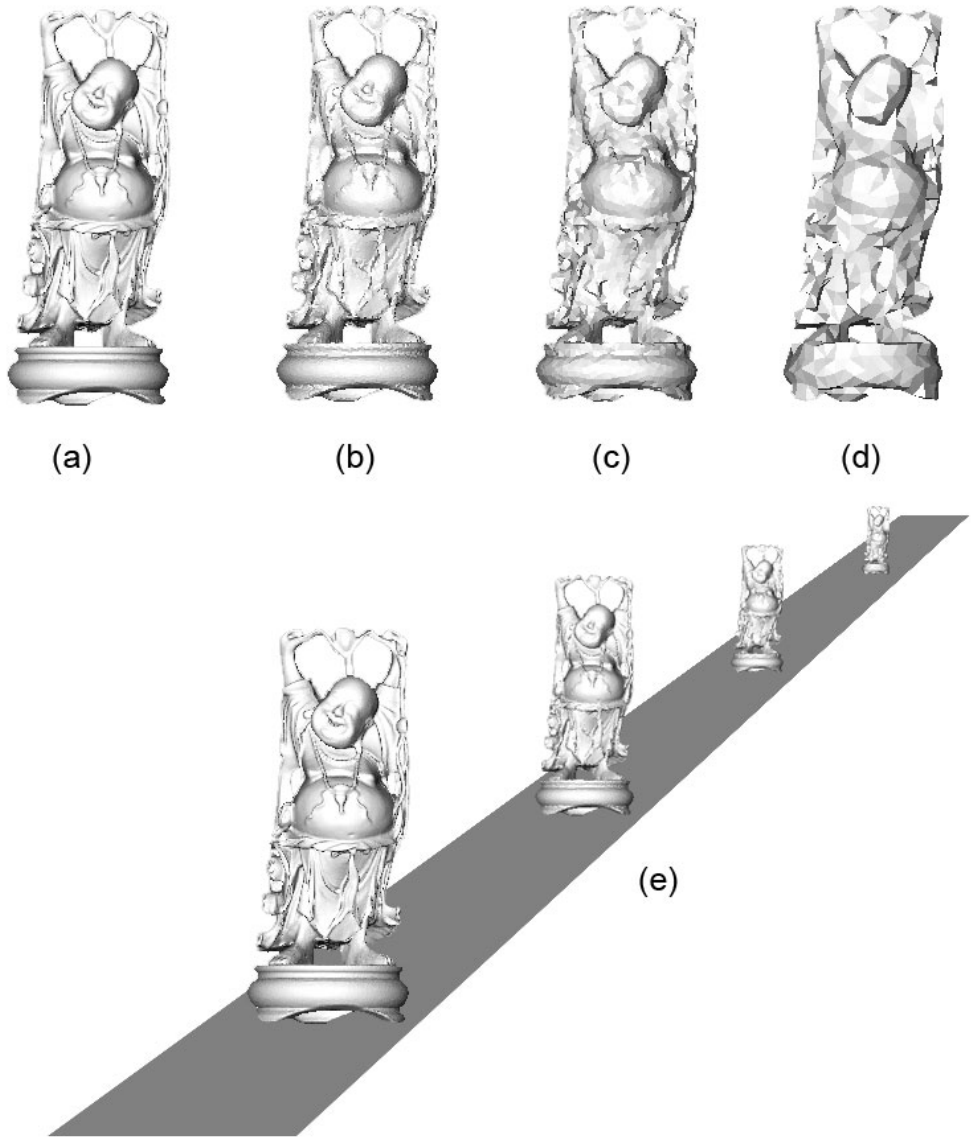


Obrázek 2.1: Manifold a non-manifold oblasti

a) disk, b) hraniční vertex, c) vertex je sdílen dvěma nespojenými skupinami trojúhelníků, d) hrana je sdílena třemi trojúhelníky, e) vertex dělí hranu

2.2 Zjednodušení

Polygonální modely jsou dány pevným počtem vertexů a trojúhelníků, což nemusí řadě aplikací vyhovovat. Při typech aplikací, jako jsou například počítačové hry je objekt v závislosti na vzdálenosti od pozorovatele reprezentován po renderingu určitým omezeným počtem bodů. Pokud je počet těchto bodů malý (objekt je ve velké vzdálenosti od pozorovatele), je možné použít objekt zredukovaný na mnohem menší počet polygonů bez ztráty vizuální kvality modelu (*Obrázek 2.2*).



Obrázek 2.2: Různé stupně zjednodušení v závislosti na jejich vzdálenosti od pozorovatele a) originál – 1025448 trojúhelníků b) 120545 trojúhelníků c) 65214 trojúhelníků d) 20145 trojúhelníků e) použití vhodného modelu v určité vzdálenosti

Podobně v nejrůznějších typech vizualizačních aplikací často modely s velmi vysokým počtem trojúhelníků zpomalují výkon takovéto aplikace – zde může být rovněž použito redukovaných modelů.

2.2.1 Typy LOD

Při použití zjednodušených modelů v nejrůznějších aplikacích můžeme použít různé způsoby jak a kdy v těchto aplikacích metody zjednodušení uplatnit – mluvíme pak o *diskrétním (discrete) LOD*, *kontinuálním (continuous) LOD* a *pohledově závislým (view-dependent) LOD*.

Pokud jde o *diskrétní LOD*, máme na mysli zjednodušení modelu, které je provedeno mimo danou aplikaci. Objekt je několikrát zjednodušen a výstupem je několik objektů s různou úrovní detailu. V aplikaci bude potom v závislosti na určitém kritériu (např. již zmíněná vzdálenost objektu od pozorovatele) vybrán objekt s vhodnou úrovní detailu.

Kontinuální LOD se pak od diskrétního liší v tom, že místo vytvoření několika objektů s různou úrovní detailu mimo běh aplikace, vytváří zjednodušené modely přímo v průběhu aplikace – v reálném čase. Poskytuje tak lepší granularitu než LOD diskrétní, neboť přechody mezi úrovněmi detailu jsou plynulé a navíc bývá obvykle méně náročný na rendering objektů, protože používá vždy jen nutný počet polygonů.

Rozšířením kontinuálního LOD je *pohledově-závislý LOD*, který je velmi často používán např. v leteckých simulátorech, kde je potřeba rozsáhlé povrchy, jako třeba terén zobrazovat *anisotropně* – různé oblasti povrchu nabývají různé úrovně detailu. Potom část terénu, která je blíže pozorovateli obsahuje vetší detail a vzdálenější část detail mnohem menší.

Pokud by byl pro tento účel použit diskrétní nebo kontinuální LOD, nikdy by nebyl zaručen optimální poměr mezi výkonem a kvalitou zobrazení – vždy by bylo jedno na úkor druhého.

V této práci jsou uvedeny výhradně metody zjednodušení pracující na principu *diskrétního LOD*.

2.3 Operátory lokálního zjednodušení

V této části jsou popsány operátory lokálního zjednodušení, které tvoří základ každého redukčního algoritmu. Jsou zde popsány pouze ty operátory, které jsou v práci použity – existuje však řada dalších.

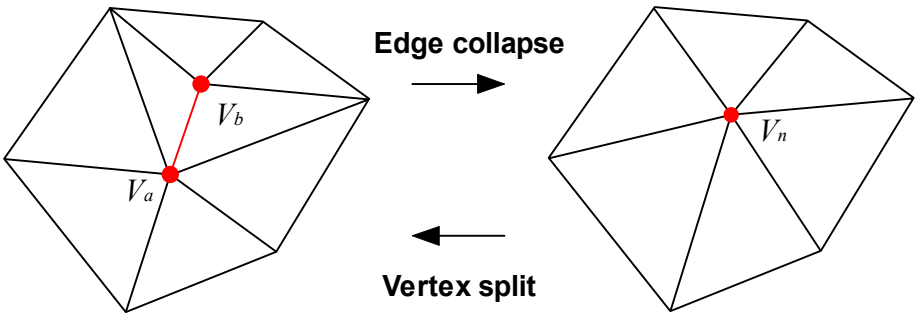
Každý z těchto operátorů určitým způsobem modifikuje malou část sítě – jejich opakované použití v různých částech sítě (určeno nejrůznějšími kritérii) pak v důsledku zjednoduší síť polygonů jako celku. U některých operátorů existují i tzv. inverzní operátory, které naopak přidávají do sítě více detailu.

2.3.1 Operátor kontrakce hrany (*edge collapse*)

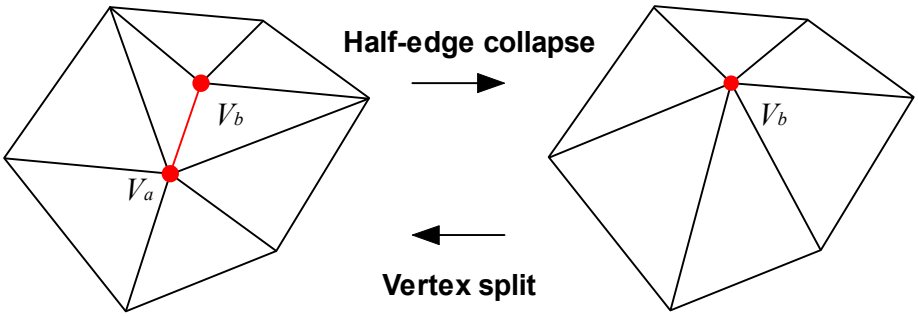
Operátor kontrakce hrany [Hoppe93] kontrahuje hranu (v_a, v_b) na jediný vertex v_n - viz. *Obrázek 2.3*. Tato operace způsobí odstranění této hrany a stejně tak trojúhelníků, které tuto hranu obsahují.

Inverzní operátorem je pak operátor rozdělení vertexu (*vertex split*).

Existují dvě verze tohoto operátoru – úplná kontrakce hrany (*full-edge collapse*) a poloviční kontrakce hrany (*half-edge collapse*). Poloviční kontrakce hrany na rozdíl od obecnější úplné kontrahuje hranu na již existující vertex (*Obrázek 2.4*). Při použití tohoto operátoru tak vertexy zjednodušené sítě tvoří podmnožinu vertexů sítě původní – viz. metoda *Vertex decimation* v *Kapitole 5*.



Obrázek 2.3: Operátor kontrakce hrany

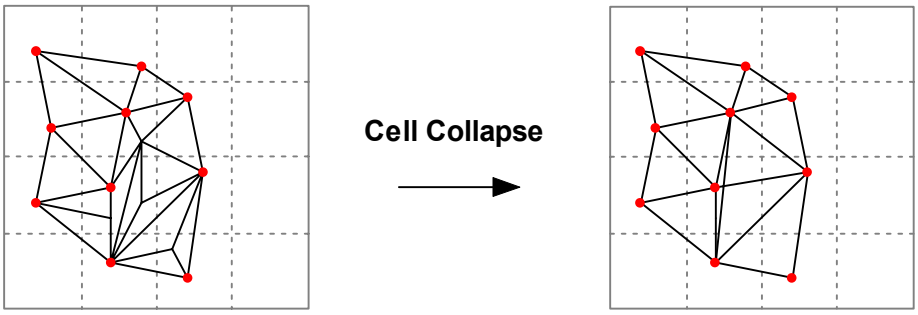


Obrázek 2.4: Operátor poloviční kontrakce hrany

2.3.2 Operátor shlukování v buňce (cell collapse)

Operátor shlukování v buňce redukuje síť trojúhelníků shlukováním několika vertexů na jediný vertex v určitém objemu – buňce (*cell*). Trojúhelníky, které jsou tímto operátorem redukovány (degenerovány) na čáru nebo bod jsou následně ze sítě odstraněny.

Poprvé byl tento operátor použit u metody *vertex clustering* [Ross93], kde byla každá buňka pevně umístěna v trojrozměrné mřížce (*grid*) – tato operace je ilustrována na *Obrázku 2.5*. V metodě *floating cell vertex clustering* [Low97] je zase každá buňka svázána s jednotlivými vertexy – viz. Kapitola 4. Vertex, na který jsou ostatní redukovány, je tvořen buď jedním z původním vertexů buňky, nebo jde o vertex nějakým způsobem vypočítaný z okolních (uměle vytvořený). V metodách uvedených v této práci je však vždy použit jeden z původních vertexů sítě.



Obrázek 2.5: Operátor shlukování v buňce

Jak je vidět na *Obrázku 2.5*, operace shlukování nezachovává původní topologii sítě – trojúhelníky se překrývají (*mesh folding*).

2.4 Hodnocení kvality aproximace polygonálních modelů

Při procesu zjednodušování modelů je velmi důležitým atributem kvalita aproximace (tedy na kolik se redukovaný objekt blíží objektu původnímu).

Konečným článkem v posuzování kvality aproximace je člověk – ten může provést vizuální srovnání obou modelů. Toto srovnání je ale značně subjektivní při posuzování kvality, proto jsou zavedeny exaktní metody k určování kvality aproximace. V této práci je k posouzení

kvality použita geometrická chyba, která posuzuje geometrickou podobnost objektů před a po zjednodušení.

2.4.1 Geometrická chyba

Polygonální modely v 3D grafice jsou reprezentovány souřadnicemi vertexů v 3D prostoru. Aproximace těchto modelů tedy snižuje počet vertexů což vede ke změně povrchu modelu.
Měření geometrické chyby a dosažení její minimální velikosti pak vede na kvalitnější aproximaci a přesnější vzhled výsledného modelu při stejném počtu vertexů po zjednodušení.
Vzdálenost mezi dvěma body $p_1 = (x_1, y_1, z_1)$ a $p_2 = (x_2, y_2, z_2)$ je podle Eukleidovské geometrie

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{2.1}$$

Vzdálenost dvou povrchů je poněkud složitější – je definována jako vzdálenost všech bodů, které se na povrchu nacházejí.
Díky polygonální povaze povrchů v 3D grafice je možno tento proces podstatně zjednodušit na hledání vzdáleností mezi významnými body povrchu – vertexy.

2.4.2 Hausdorffova vzdálenost

Hausdorffova vzdálenost je pojem známý z topologie, používaná pro zpracování obrazu, modelování povrchů, atd. Je definována na množině bodů a jelikož povrch lze popsat jako spojitou množinu bodů, lze ji použít i pro povrchy.
Pokud máme dvě množiny bodů A a B, pak Hausdorffova vzdálenost je maximální hodnota minimální vzdálenosti mezi body z těchto dvou množin.
Tedy pro každý bod z množiny A je třeba nalézt nejbližší bod z množiny B a obráceně (oboustranná Hausdorffova vzdálenost). Poté se vypočte vzdálenost mezi všemi těmito nejbližšími body a získá se maximální hodnota této vzdálenosti. To je možné zapsat následujícím způsobem

$$H(A, B) = \max(h(A, B), h(B, A)), \tag{2.2}$$

kde

$$h(A, B) = \max_{a \in A} \min_{b \in B} \| a - b \| \tag{2.3}$$

Využití Hausdorffovy vzdálenosti pro určení geometrické chyby aproximace je výpočetně velice náročné – byť by šlo pouze o jednostrannou Hausdorffovu vzdálenost. Její časová náročnost je pak *O(2*v*v)* kde *v* je počet vertexů.
Při skutečně velkých modelech (milion a více polygonů) to může v porovnání s časem nutným k samotné aproximaci, která je téměř lineární, představovat skutečně neúměrně velkou hodnotu.
Pro naše účely „rychlého" algoritmu musíme tedy najít jinou metodu pro porovnání aproximace modelů.

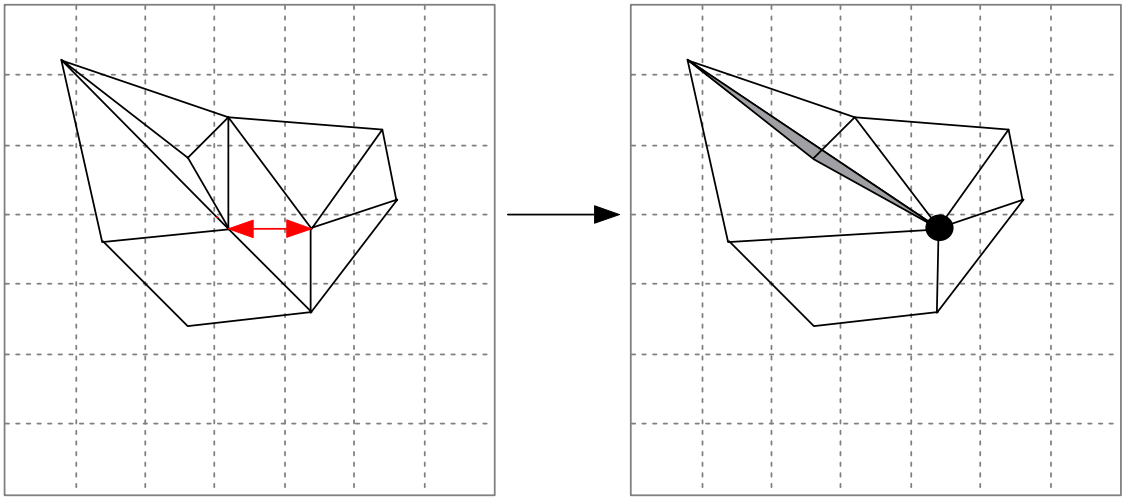
2.4.3 Porovnání povrchu

Další možností, jak porovnat kvalitu aproximace je porovnání povrchu obou modelů.
Vypočteme jednoduše sumu povrch všech polygonů před a po aproximaci a tyto dvě hodnoty vydělíme – získáme tak poměr velikosti povrchu před a po zjednodušení :

$$A_r = \frac{A_2}{A_1}, \tag{2.4}$$

kde A_1 je celková plocha sítě polygonů původního modelu, A_2 je celková plocha polygonů zjednodušeného modelu a A_r je poměr těchto ploch.
Tento poměr nabývá hodnot v intervalu *<0..1>*, kde hodnota *0* znamená maximální zjednodušení (na nulový povrch) a *1* minimální zjednodušení (stejné modely).

S ohledem na použité metody zjednodušení ovšem nastává podstatný problém, který zde uvedené porovnávání povrchů činí pro tyto metody nepoužitelné.
Jak je uvedeno v *části 2.3.2*, u metod, které používají jako zjednodušovací operátor shlukování vertexů v buňce mřížky může dojít k jevu zvanému "mesh folding" (překrývání polygonů).
Tento jev je ilustrován na *Obrázku 2.6*.



Obrázek 2.6: Mesh folding

V důsledku tohoto jevu pak může snadno dojít k tomu, že povrch zjednodušeného modelu bude větší než povrch původní.

2.4.4 Míra posunutí vertexů

Jako metoda pro určení chyby aproximace byla nakonec zvolena metoda, která jednoduše porovnává míru posunutí jednotlivých vertexů – tedy o jakou vzdálenost se jednotlivé vertexy zjednodušeného modelu posunou oproti vertexům původního modelu. Míra posunutí jednoho vertexu nám dává lokální chybu aproximace a součet míry posunutí všech vertexů pak celkovou chybu aproximace.

Tato metoda se jeví jako objektivní, neboť lokální zjednodušovací operátory použité v této práci pracují na základě posunutí vertexů o co nejkratší vzdálenost.

Tato metoda je časově velice úsporná a velmi dobře zapadá do stávajících metod aproximace.

Jediným problémem zůstávají absolutní čísla, které tato metoda produkuje. Při zvětšení nebo zmenšení modelu tak dostáváme zcela jiné hodnoty posunutí.

Aby byl tento nežádoucí jev eliminován, jsou velikosti posunutí vyděleny diagonálou bounding boxu modelu – tedy největší možnou velikostí posunutí. Tato hodnota je zároveň maximální možnou lokální chybou. Takto je vypočítána relativní chyba vzhledem k velikosti modelu.

3 Zjednodušování modelů metodou *Uniform Vertex Clustering*

Původní metoda vertex clustering [Ros92] zůstává dodnes velice významným algoritmem aproximace polygonálních modelů. Stal se základem mnoha dalších algoritmů, které ho po všech stránkách vylepšují a posouvají dále – jako příklad je možno uvést algoritmus z tohoto přímo vycházející – algoritmus *Floating Cell Vertex Clustering* [Low97], jenž je rovněž součástí této práce.
Algoritmus je velice rychlý, robustní a s poměrně jednoduchou implementací.

3.1.1 Výhody a nevýhody metody

Výhody metody :

- Vysoká rychlost – použitelné pro velké modely
- Možno mít na vstupu manifold i non-manifold objekty – nezávislé na topologii vstupu

Nevýhody metody :

- Nezachovává topologii povrchu (někde nevadí)
- Nekvalitní výstup
- Není jednoduchým způsobem možné zadat požadovaný počet trojúhelníků na výstupu (závisí na rozlišení mřížky)
- Konečný vzhled je značně závislý na natočení modelu – jiné natočení dává jiný výstup.

3.2 Popis metody

Metoda vertex clustering začíná ohodnocením všech vertexů v síti přiřazením váhy (důležitosti) jednotlivým vertexům. Největší důležitost je přiřazena vertexům, které jsou součástí polygonů s větší plochou a které zároveň leží v oblasti s největším zakřivením – u těch je pravděpodobnější, že budou ležet na siluetě modelu.
Poté je vytvořena 3D mřížka shodná s bounding boxem modelu, která je rozdělena na jednotlivé buňky (*cell*). Všechny buňky mřížky mají stejnou velikost.
Vertexy jsou v těchto buňkách shlukovány na jeden nejdůležitější vertex v dané buňce a trojúhelníky takto redukované (degenerované) na čáru nebo bod jsou odstraněny.

Míra rozdělení této mřížky na jednotlivé buňky (její rozlišení) v důsledku určuje míru zjednodušení modelu, protože mřížka s hrubší rozlišením (menším počtem buněk) bude shlukovat více vertexů než mřížka s jemnějším rozlišením (větším počtem buněk).

3.2.1 Ohodnocení důležitosti vertexů

Kritéria pro ohodnocení vertexů již byly uvedeny výše, zde budou rozebrány podrobněji. Důležitost vertexů je dána dvěma faktory : velikostí nejdelší hrany vycházející z vertexu a nejmenšího úhlu který svírají každé dvě hrany vycházející z vertexu.

Konečná míra důležitosti vertexu je dána váženým součtem těchto dvou faktorů.

Co se týče určení prvního faktoru, je nutné aby délka nebyla závislá na velikosti modelu. Je tedy možné tuto hodnotu vydělit buď diagonálou bounding boxu modelu nebo nejdelší hranou. V aktuálním algoritmu byla zvolena druhá možnost – faktor délky pak bude nabývat hodnot v intervalu *<0,1>*.

Druhý faktor souvisí s velikostí úhlu mezi hranami – čím je tento úhel menší, tím je větší pravděpodobnost, že vertex leží na siluetě modelu nebo alespoň v oblasti z velkým zakřivením. V původním algoritmu je pro tento faktor určena hodnota $\frac{1}{\varphi}$, kde φ je nejmenší úhel mezi hranami. V dalších rozšířeních byly použity i jiné hodnoty – např. u metody Floating Cell Vertex Clustering je použita hodnota $\cos(\frac{\varphi}{2})$, která lépe vystihuje možnost, že vertex leží na siluetě. Tato hodnota byla použita i zde a to u obou metod shlukování vertexů.

3.2.2 Shlukování vertexů v mřížce (clustering)

Po té, co jsou všechny vertexy ohodnoceny, je nutné je umístit do buněk 3D mřížky.

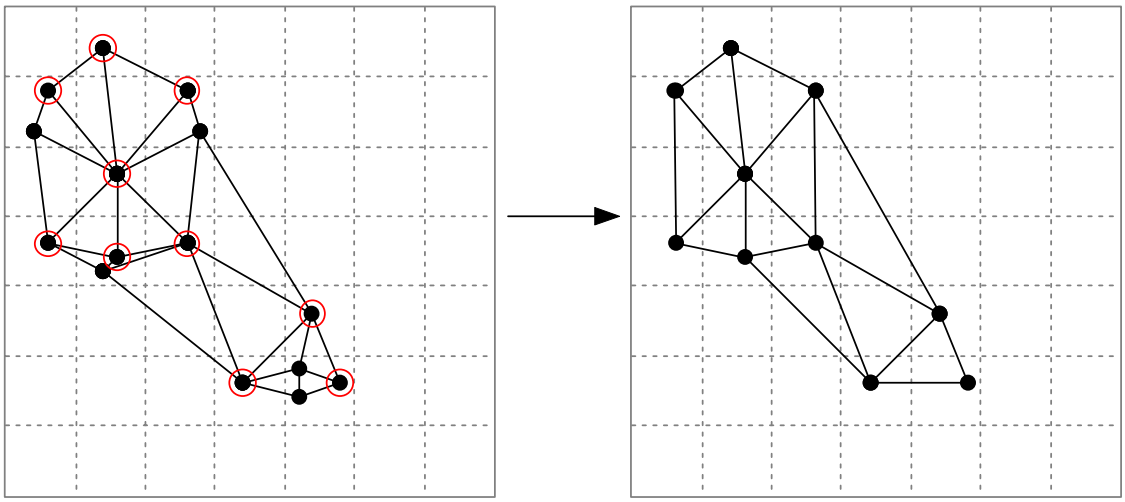
Toto umístění je provedeno pomocí mapovací funkce

$$g_i = (\text{int})\frac{x - o_x}{sph} + (\text{int})\frac{y - o_y}{sph} p_x + (\text{int})\frac{z - o_z}{sph} p_x p_y , \qquad (3.1)$$

Kde g_i je index buňky v mřížce, (*x,,y,z*) souřadnice vertexu v prostoru, (o_x, o_y, o_z) jsou souřadnice počátku 3D mřížky v prostoru, p_x je počet buněk v ose x a p_y počet buněk v ose y.

Vertexy se tedy jeden po druhém umísťují do buněk mřížky pomocí uvedené mapovací funkce a pokud je již v buňce umístěn vertex s větší důležitostí, nahradí se zkoumaný vertex tímto vertexem. Pokud má však zkoumaný vertex výšší důležitost, nahradí se vertex umístěný v buňce tímto a tento nový vertex se uloží se do buňky.

Tento postup je ukázán na následujícím obrázku – vyznačené vertexy jsou v dané buňce nejdůležitější.



Obrázek 3.1: Shlukování vertexů v buňkách mřížky

3.2.3 Odstranění redukovaných trojúhelníků

Předchozím procesem shlukování vzniknou trojúhelníky, které jsou redukovány na čáry nebo body. Všechny tyto trojúhelníky jsou ze sítě odstraněny a indexy bodů zbývajících trojúhelníků jsou přepsány. Rovněž vertexy, které již netvoří vrchol žádného trojúhelníku, jsou odstraněny. V konečné fázi algoritmu je provedena úprava indexů vertexů u zbývajících trojúhelníků.
Podobný postup je aplikován na všechny metody – nebude tedy dále uváděn.

3.3 Implementace metody

3.3.1 Celkový postup algoritmu

1. Ohodnoť všechny vertexy podle jejich důležitosti
2. Shlukuj vertexy v buňkách mřížky ne nejdůležitější vertex v každé buňce
3. Odstraň redukované polygony
4. Odstraň nadbytečné vertexy
5. Uprav indexy vertexů v polygonech

S uvedenými akcemi souvisí i odstranění indexů materiálu a textur, které se provádějí společně s odstraňováním polygonů nebo vertexů.

3.3.2 Časová náročnost

V následující tabulce je uvedena časová náročnost jednotlivých částí algoritmu.

Ohodnocení vertexů	*O(v)*
Shlukování vertexů	*O(v)*
Odstranění redukovaných trojúhelníků	*O(n)*
Odstranění vertexů	*O(v)*
Úprava indexů	*O(n')*

Tabulka 3.1: Časová složitost jednotlivých částí algoritmu

‚kde *v* je počet vertexů, *n* počet trojúhelníků původního modelu a *n'* je počet trojúhelníků po zjednodušení modelu.

4 Zjednodušování modelů metodou *Floating Cell Vertex Clustering*

Metoda je rozšířením metody předchozí. Dává lepší aproximaci a výsledný povrch je více konzistentní. Co se týče rychlosti této metody, je téměř shodná s metodou předchozí – časová náročnost je zde rozšířena především o řazení vertexů podle důležitosti.

4.1 Popis metody

Tato metoda, publikovaná v roce 1997 [Low97], dává podstatně lepší výsledky než původní clustering. Ačkoliv je na původní metodě založena, její postup je poněkud jiný.
Ohodnocení vertexů probíhá shodně, poté jsou však vertexy podle jejich důležitosti setříděny od vertexu s *největší* důležitostí.
Následně jsou tyto setříděné vertexy postupně umísťovány do mřížky, ale jejich shlukování probíhá jinak.
Místo toho, aby byly buňky rovnoměrně rozmístěny v mřížce, jsou tvořeny kolem středů umísťovaných vertexů, čímž je možné dosáhout rovnoměrnějšího rozložení výsledné sítě.

4.1.1 Výhody a nevýhody metody

Výhody metody :

- Vysoká rychlost – použitelné pro velké modely
- Možno mít na vstupu manifold i non-manifold objekty – nezávislé na topologii vstupu

Nevýhody metody :

- Nezachovává topologii povrchu (někde nevadí)
- Jiné, pomalejší metody produkují vizuálně kvalitnější výstup
- Není jednoduchým způsobem možné zadat požadovaný počet trojúhelníků na výstupu (závisí na rozlišení mřižky)
- Konečný vzhled je značně závislý na natočení modelu – jiné natočení dává jiný výstup.

4.1.2 Třídění vertexů

Jelikož je nutné v této metodě narozdíl od předchozí vertexy setřídit podle důležitosti, bylo také potřeba použít dostatečně rychlý algoritmus, neboť u velkých modelů by tato část mohla svou časovou náročností přesahovat únosné hodnoty.
Pro třídění vertexů je použit jeden z nejrychlejších algoritmů – algoritmus *quick sort*.
Časová náročnost tohoto algoritmu je logaritmická *O(v*log v)* (nejhorší případ) – kde *v* je počet tříděných vertexů.
Protože však důležitost jednotlivých vertexů nabývá značně rozdílných hodnot, je skutečná časová náročnost jejich setřídění podstatně menší a blíží se linearní.

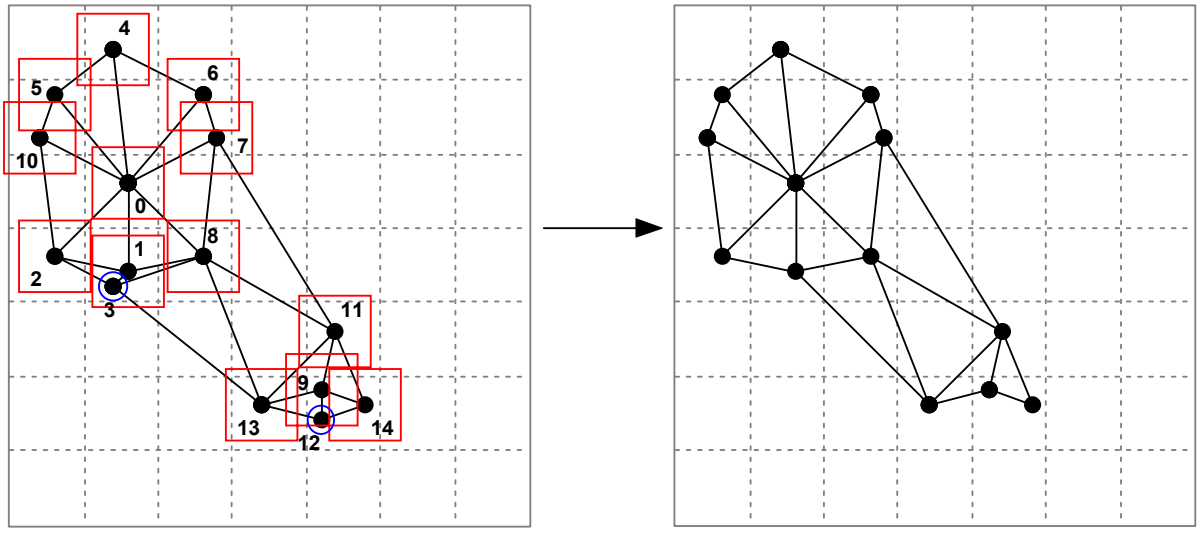
4.1.3 Shlukování vertexů v mřížce

Jak již bylo uvedeno, shlukování vertexů se zde liší od klasického clusteringu. I zde sice existuje rovnoměrná mřížka, ale narozdíl od klasického clusteringu tato mřížka neobsahuje hodnoty nejdůležitějšího vertexu v každé buňce, ale pouze seznam vertexů, které se v buňkách nacházejí. Tyto vertexy pak určují středy tzv. plovoucích buněk (*floating cell*). Délka hrany těchto buněk je totožná s délkou hrany buněk v rovnoměrné mřížce. Mapovací funkce je totožná s funkcí uvedenou v *části 3.2.1*.

Postup algoritmu shlukování :

1. Vezmi další nejdůležitější vertex seznamu
2. Zjisti jeho polohu v mřížce pomocí mapovací funkce
3. Zjisti zda vertex neleží uvnitř nějaké plovoucí buňky
4. Pokud ano, shlukuje se tento vertex na středový vertex této plovoucí buňky
5. Pokud ne, přidej vertex na seznam vertexů v buňce mřížky
6. Pokračuj na 1.) dokud nejsi na konci seznamu vertexů

Dále může nastat situace, že vertex leží uvnitř více než jedné buňky. V takovém případě je možné vybrat buď podle nejmenší vzdálenosti, nebo podle největší důležitosti vektorů v daných buňkách.
S odkazem na původní text [Low97] je použito kritérium vzdálenosti, neboť při použití druhého zmíněného kritéria dochází k nežádoucímu jevu - rozšiřování trojúhelníků.



Obrázek 4.1: Shlukování vertexů v buňkách vytvořených kolem vertexů

Na *Obrázku 4.1*. je ilustrován postup algoritmu. Je zde 15 vertexů označených čísly od 0 po 14 – což je od největší důležitosti po nejmenší. Jak je z obrázku patrné, vertexy 3 a 12 budou shlukovány, neboť se nacházejí v plovoucích buňkách daných vertexy o vetší důležitosti.
Ostatní vertexy jsou zapsány do mřížky a už shlukovány nebudou – budou tvořit nové buňky.
S přidáváním dalších buněk stoupá u zkoumaných vertexů pravděpodobnost, že je nějaká buňka "pohltí" a budou shlukovány.
Při porovnání s *Obrázkem 3.1* je evidentní, že tato metoda dosahuje poněkud jiného výstupu.
Otázkou však zůstává jak zjistit, zda zkoumaný vertex leží v nějaké plovoucí buňce. Samozřejmě je možné použít naivní algoritmus, který porovnává každé dva vertexy, ale jelikož časová náročnost takového postupu je dosti velká, je toto řešení krajně nevhodné.
Nejlepším řešením je využít toho, že velikost buňky umístěné kolem vertexu je totožná s velikostí buňky v mřížce. Potom stačí porovnat zkoumaný vertex s vertexy které leží v sousedních osmi buňkách (3D), protože buňky kolem vertexů ve vzdálenějších buňkách z principu nemohou tento vertex obsahovat.

4.1.4 Víceúrovňový clustering

V souvislosti s metodami shlukování se nabízí myšlenka možnosti využít již zjednodušený model (ale jen touto metodou) mřížkou o určitém rozlišení a aplikovat na něj mřížku o menším rozlišení. Toto lze aplikovat několikrát za sebou vždy s menším rozlišením mřížky a výsledný model s určitým počtem polygonů se bude lišit od modelu se stejným počtem polygonů

zjednodušeným najednou. Je však jasné, že pokud na již zjednodušený model aplikujeme mřížku se stejným nebo vyšším rozlišením, žádné další zjednodušení to již nepřinese. Porovnání výstupů je často velmi zajímavé.

4.2 Implementace metody

4.2.1 Průběh algoritmu

1. Ohodoť všechny vertexy podle jejich důležitosti
2. Seřaď vertexy do seznamu podle jejich důležitosti *od největší po nejmenší*
3. Vezmi další *nejvíce* důležitý vertex v seznamu a umísti ho do mřížky – vytvoř plovoucí buňku nebo shlukuj
4. Pokud je dosažen konec seznamu, jdi na 5.), jinak jdi na 3.)
5. Odstraň redukované polygony
6. Odstraň nadbytečné vertexy
7. Uprav indexy vertexů v polygonech

4.2.2 Časová náročnost algoritmu

V následující tabulce je uvedena časová náročnost jednotlivých částí algoritmu.

Ohodnocení vertexů	*O(v)*
Seřazení vertexů	*O(v*log v)*
Shlukování vertexů	*O(v)*
Odstranění redukovaných trojúhelníků	*O(n)*
Odstranění vertexů	*O(v)*
Úprava indexů	*O(n')*

Tabulka 4.1: Časová složitost jednotlivých částí algoritmu

‚kde *v* je počet vertexů, *n* počet trojúhelníků původního modelu a *n'* je počet trojúhelníků po zjednodušení modelu.

5 Zjednodušování modelů metodou *Vertex Decimation*

Metoda pocházející z roku 1992 používaná (s úpravami) dodnes. Už původní algoritmus [Schroeder92] zaručoval poměrně rychlé a topologii zachovávající zjednodušení modelů, které se hodí zejména pro vědecké a lékařské vizualizace.

5.1 Popis metody

Původní algoritmus používal jako lokální zjednodušovací operátor operátor odstranění vertexu s následnou triangulací vzniklé díry. V této práci je však použita upravená verze tohoto algoritmu [Schroeder97], kde je místo tohoto operátoru použit *operátor kontrakce hrany*, přesněji *operátor poloviční kontrakce hrany* – viz. *část 2.3.1*. Není tak potřeba aplikovat obecnou triangulaci, ale kvůli zachování topologie sítě je nutné použít nějaký algoritmus, který bude testovat konzistenci povrchu po operaci kontrakce hrany.
Metoda dělí vertexy na různé typy a podle toho k nim přistupuje
Na rozdíl od předchozích metod je tato metoda víceprůchodová – k dosažení

5.1.1 Výhody a nevýhody metody

Výhody:

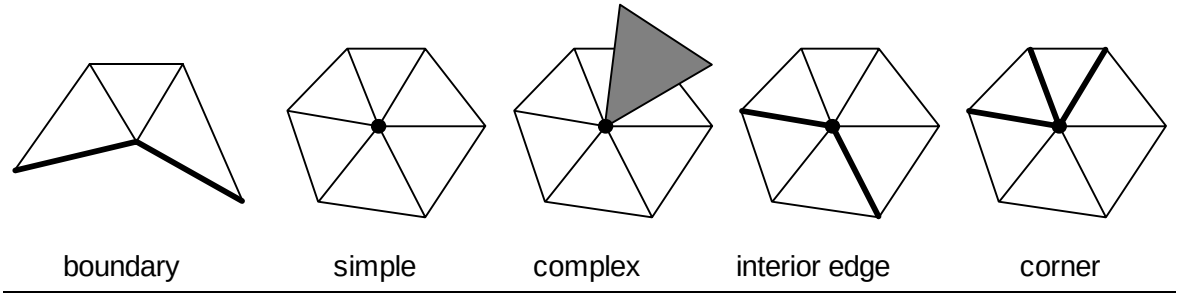
- Poměrně snadná implementace
- Relativně vysoká rychlost
- Zachování topologie modelu
- Možnost zredukovat model na přesně daný počet trojúhelníků

Nevýhody:

- Díky zachování topologie menší možnosti zjednodušení
- Nevhodné pro drastické zjednodušení modelu

5.1.2 Typy vertexů

V metodě *vertex decimation* se rozeznávají různé typy vertexů – jednoduchý, složitý, hraniční, hranový a rohový (viz. *Obrázek 5.1*).



Obrázek 5.1: Typy vertexů sítě [Schroeder92]

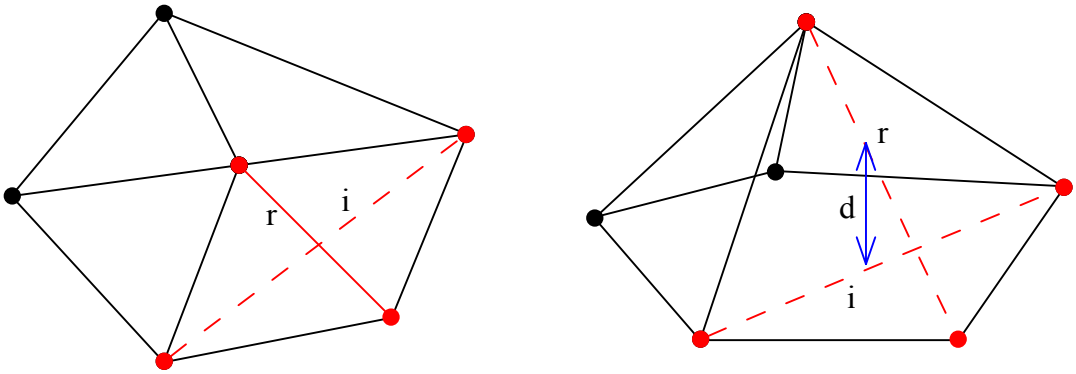
Každý z těchto typů má svou specifickou funkci při ohodnocování vertexu v síti.
Jednoduchý vertex (*simple*) je ohraničen trsem trojúhelníků, kde každý z nich má společné hrany s jiným – trs tvoří disk. Tyto vertexy jsou zpravidla v modelu nejčastější a jsou tedy hlavním cílem decimace. Dalším typem je vertex hraniční (*boundary*), který je ohraničen neuzavřeným trsem trojúhelníků – existují tedy dva trojúhelníky, které obsahují jednu hranu jenž není sdílena žádným dalším.
Je-li nějaká hrana společná více než dvěma trojúhelníkům, nebo se ve vrcholu dotýká trojúhelník, který má s ostatními trojúhelníky okolo tohoto vrcholu společný právě jen tento bod, je vrchol označen jako složitý (*complex*).
Dále je možné u jednoduchých vertexů posuzovat důležitost hran, které z něj vycházejí. Tato důležitost se posuzuje prostřednictvím úhlu mezi normálami sousedních trojúhelníků v trsu – pokud je tento úhel větší než určený mezní úhel, je tato hrana označena jako *významná*.
Vertex, z něhož vycházejí dvě takové hrany se nazývá hranový (*interior edge*). Pokud z vertexu vychází tři a více takových hran, nazývá se rohový (*corner*).

5.1.3 Hodnocení důležitosti vertexů

V algoritmu jsou pro určení míry důležitosti jednotlivých vertexů v souvislosti s typem vertexu použita následující kritéria :

- Jednoduchý vertex (*simple*) :

  V původním algoritmu [Schroeder 92] je pro jednoduché vertexy kritériem vzdálenost vertexu od roviny.
  Tato rovina se nalezne jako tzv. průměrná rovina, tj. rovina proložená mezi koncovými body hran vycházejících ze zkoumaného vrcholu. Průměrnou ji nazýváme z toho důvodu, že tyto body obecně v rovině neleží – je dána vypočítaným – tzv. průměrným normálovým vektorem.
  Toto kritérium zde nebude dále rozváděno, protože ve stávajícím algoritmu je použito kritérium jiné, které je dáno vzdáleností mezi hranami trsu [Franz 00].
  Pro každou vnitřní hranu trsu, která je hranou reálnou (*r*) se zjistí její vzdálenost (*d*) od hrany imaginární (*i*), která je tvořena body, jenž jsou koncovými body hran, které vycházejí z bodu, který je koncovým bodem hrany reálné (viz. *Obrázek 5.2*).
  Součet těchto vzdáleností nám dává konečnou důležitost vertexu. Vertex, který leží v rovině tak má důležitost rovnu nule.



Obrázek 5.2: Ohodnocení důležitosti vertexu metodou edge switching [Franz00]

- Hraniční vertex (*boundary*) :

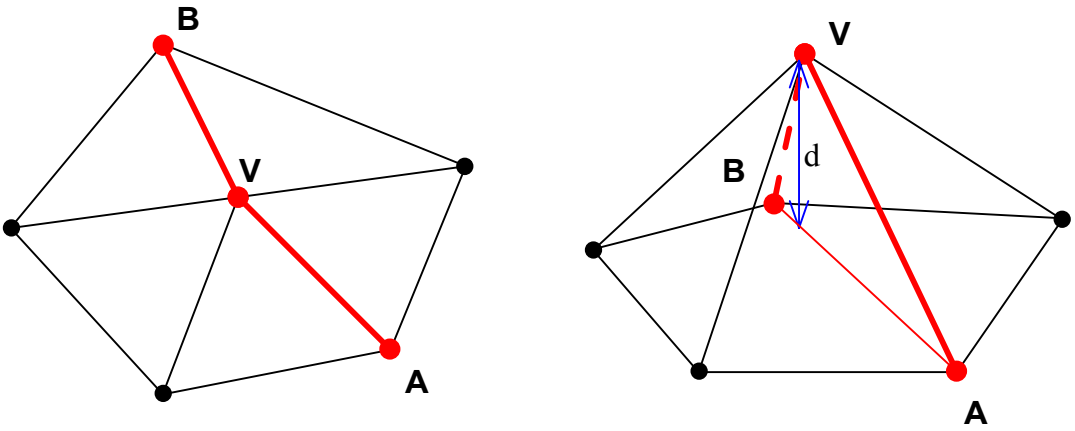
  Ačkoliv v původním algoritmu je hraniční vertex decimován, v této verzi je ponechán – jeho důležitost se tedy nezjišťuje

- Složitý vertex (*non-manifold*) :

  Složitý vertex se ze sítě neodstraňuje, jeho důležitost tedy není podstatná.

- Hranový vertex (*interior edge*) :

  Pro určení míry důležitosti hranového vertexu je použito jednodušší kritérium, než u vertexu jednoduchého. Toto kritérium spočívá ve výpočtu vzdálenosti vertexu od hrany, která je určena koncovými body vnitřní hrany [Schroeder92] - na *Obrázku 5.3* označené *A* a *B*.



Obrázek 5.3: Ohodnocení důležitosti vertexu metodou edge switching

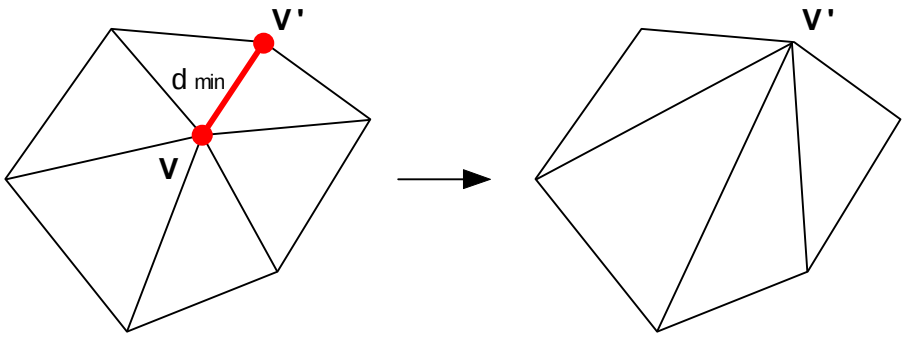
- Rohový vertex (*corner*) :

  Rohový vertex není odstraňován – jeho důležitost není zjišťována.

5.1.4 Kritérium kontrakce hrany

Před samotnou decimací vertexu je nutné rozhodnout, jaká z vnitřních hran trsu bude kontrahována. Pro kritérium této kontrakce existuje více možností, v případě tohoto algoritmu bylo jako kritérim použito zjištění nejkratší vnitřní hrany – tato bude nakonec kontrahována – jak je vidět na *Obrázku 5.4*.

Při kontrakci nejkratší hrany dojde také k nejmenší lokální chybě aproximace, protože dojde k nejmenšímu posunutí vertexu.

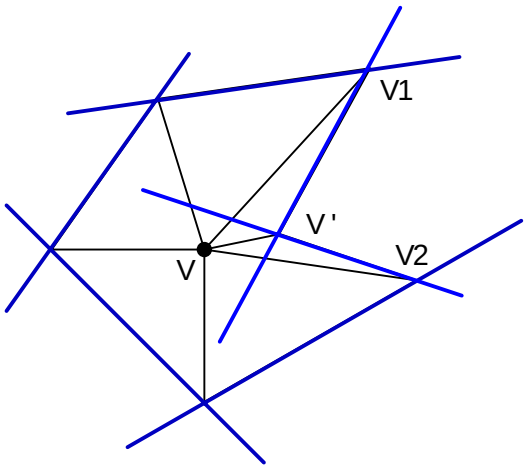


Obrázek 5.4: Kontrakce hrany s použitím kritéria nejmenší délky

5.1.5 Test konzistence povrchu

Aby byla splněna podmínka zachování konzistence povrchu, je použit test konzistence, který ještě před samotnou kontrakcí hrany zjišťuje, zda tato hrana může být skutečně kontrahována.

Tento test je založen na zjištění, zda leží oba vertexy kontrahované hrany ve stejné polorovině, jež je dána vnější hranou trsu a normálou, která je vypočtena jako nejkratší vzdálenost mezi vnější hranou a středovým vertexem trsu [Ciar96].



Obrázek 5.5: Test konzistence povrchu

Na O*brázku 5.5* je vidět, které hrany mohou být kontrahovány a které ne. Hrana dána vektorem (*V1-V)* nemůže být kontrahována, neboť vertex *V1* leží v opačné polorovině než vertex *V*. To samé platí pro hranu (*V2-V*). Ostatní hrany mohou být kontrahovány a v souvislosti s výše uvedeným kritériem kotrakce bude vybrána hrana (*V'-V*).

Pro urychlení algoritmu ale ve skutečnosti nedojde k testování konzistence výše uvedených hran, protože na vstupu jsou už hrany seřazeny podle délky a tak je testována jen hrana (*V'-V*), která testem konzistence projde a tato hrana je kontrahována.

5.2 Implementace metody

5.2.1 Průběh algoritmu

1. Procházej trojúhelníky a tvoř trsy
2. Procházej vertexy, urči jejich typ a důležitost
3. Seřaď vertexy podle důležitosti od nejmenší po největší algoritmem quick-sort
4. Vezmi další nejméně důležitý vertex v seznamu a kontrahuj hranu v trsu tímto vertexem určeným
5. Pokud je dosažen konec seznamu, jdi na 6.), jinak jdi na 4.)
6. Odstraň redukované polygony
7. Odstraň nadbytečné vertexy
8. Uprav indexy vertexů v polygonech
9. Pokud je dosaženo požadovaného počtu polygonů, skonči, pokud ne, jdi na 1.

5.2.2 Časová náročnost

V následující tabulce je uvedena časová náročnost jednotlivých částí algoritmu.

Tvorba trsů	*O(n)*
Ohodnocení vertexů	*O(v)*
Setřídění vertexů	*O(v*log v)*
Decimace vertexů	*O(v)*
Odstranění redukovaných trojúhelníků	*O(n)*
Odstranění vertexů	*O(v)*
Úprava indexů	*O(n')*

Tabulka 5.1: Časová složitost jednotlivých částí algoritmu

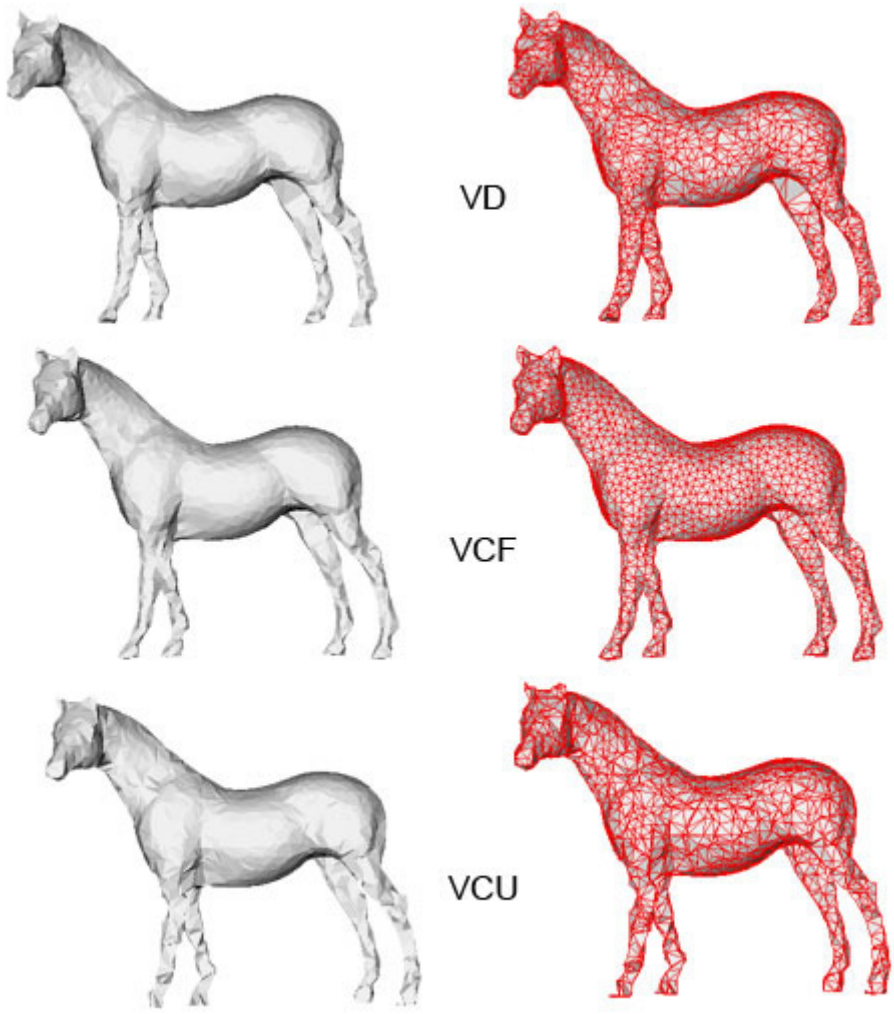
‚kde *v* je počet vertexů a *n* počet polygonů původního modelu a n' počet polygonů redukovaného modelu.

Při imlementaci této metody je potřeba si uvědomit, že je obvykle nutné udělat více průchodů, což v důsledku zvyšuje celkovou časovou náročnost algoritmu. Avšak počet vertexů s každým průchodem řádově klesá, takže celková časová náročnost je podstatně nižší než pouhé vynásobení časové náročnosti jednoho průchodu počtem průchodů.

6 Porovnání výsledků jednotlivých metod

Pro porovnávání výstupů jednotlivých metod byly použity dva modely – menší model *horse (96966 trojúhelníků)* a větší model *dragon (871414 trojúhelníků)* – viz. *Příloha B*. Výstupy byly porovnány z hlediska vizuálního, časového a dosažené chyby. U vizuálního porovnání je model zredukován na 5% původního počtu trojúhelníků.

6.1 Vizuální porovnání



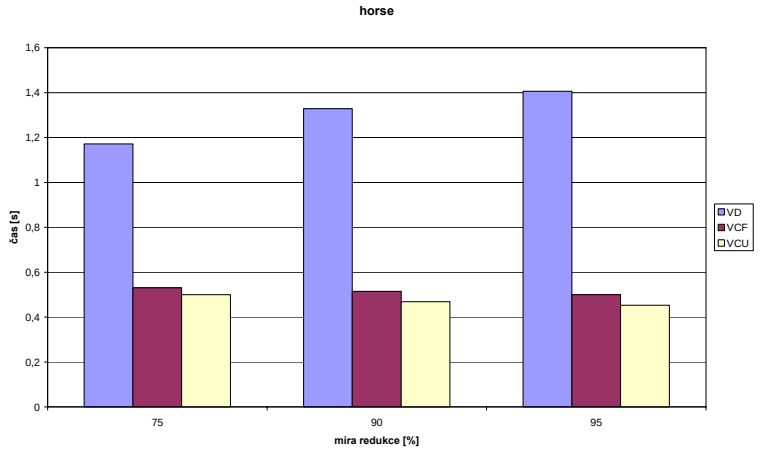
Obrázek 6.1: Vizuální porovnání jednotlivých metod – model horse (5% trojúhelníků)

Pozn.: Na *Obrázku 6.1* jsou z úsporných důvodů použity zkratky metod : VD – Vertex Decimation, VCF – Floating Cell Vertex Clustering, VCU – Uniform Vertex Clustering.

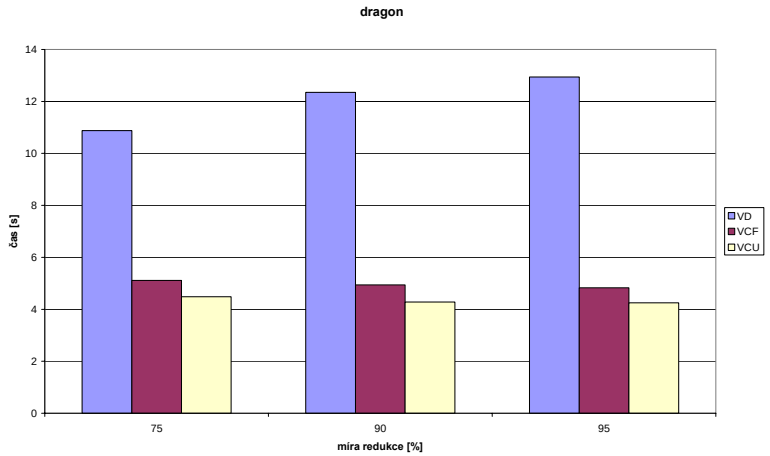
Na *Obrázku 6.1* je vidět, že nejhorší aproximace je provedena metodou Uniform Vertex Clustering – povrch je nekonzistentní a v určitých částech sítě (nohy koně) je patrná poměrně velká vizuální chyba.
Metoda Floating Cell Vertex Clustering poskytuje poměrně konzistentní a rovnoměrně rozloženou síť trojúhelníků. U metody Vertex Decimation je vidět, že v detailech jako jsou nohy koně je použita hustší síť polygonů než ve velkých oblastech (trup koně). Je zde nerovnoměrné rozložení polygonů.

6.2 Porovnání časové náročnosti

Na následujících obrázcích je formou grafů porovnána časová náročnost jednotlivých metod. V legendě grafů jsou uvedeny zkratky metod (VD – Vertex Decimation, VCF – Floating Cell Vertex Clustering, VCU – Uniform Vertex Clustering).



Obrázek 6.3: Porovnání časové náročnosti jednotlivých metod – model horse

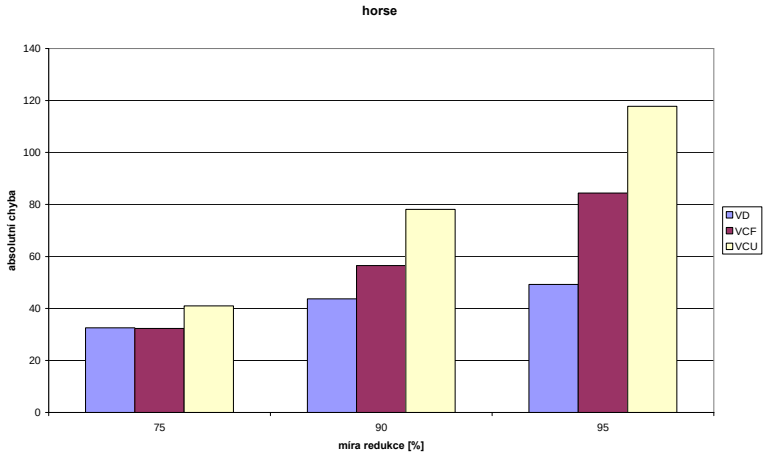


Obrázek 6.4: Porovnání časové náročnosti jednotlivých metod – model dragon

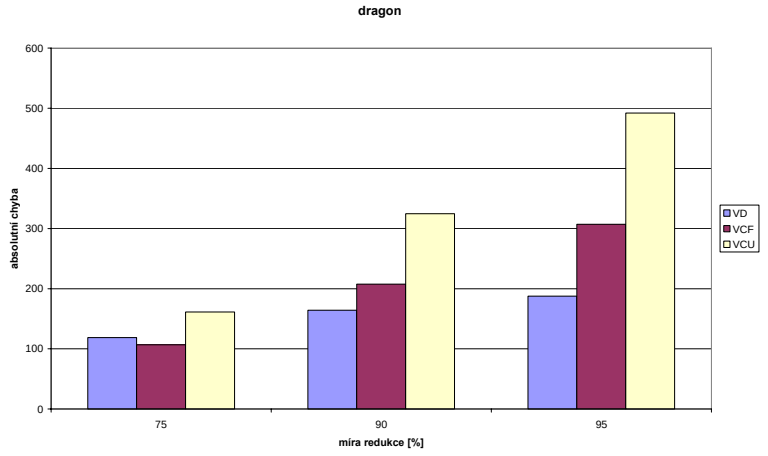
Při porovnání časové náročnosti je patrné, že algoritmus metody *Vertex Decimation* trvá nejdelší dobu – je také víceprůchodový. Drobný časový rozdíl mezi metodami *Uniform Vertex Clustering* a *Floating Cell Vertex Clustering* je způsoben především nutností řazení vertexů podle jejich důležitosti u druhé jmenované.

6.3 Porovnání geometrické chyby

Na následujících obrázcích je formou grafů porovnána časová náročnost jednotlivých metod. V legendě grafů jsou uvedeny zkratky metod (VD – Vertex Decimation, VCF – Floating Cell Vertex Clustering, VCU – Uniform Vertex Clustering).



Obrázek 6.5: Porovnání geometrické chyby jednotlivých metod – model horse



Obrázek 6.6: Porovnání geometrické chyby jednotlivých metod – model dragon

Jak je vidět z grafů, největší geometrická chyba je dosažena u metody *Uniform Vertex Clustering*.

7 Závěr

Při implementaci a testování uvedených metod bylo dosaženo dobrých výpočetních časů, ačkoliv kvalita aproximace je u pomalejších a přesnějších redukčních algoritmů jistě větší.
V této práci však šlo hlavně o porovnání výsledků několika metod a ne tolik o výslednou kvalitu aproximace.
Budoucí rozšíření této práce by zřejmě obsahovalo kvalitnější kritérium pro ohodnocení vertexů (např. Quadric Error Metrics [Garland 97]).
Metody v této práci se v textech označují jako rychlé – případně by tedy mohla být přidána další metoda(y), která by dosahovala kvalitnější aproximace povrchů než stávající metody, byť za cenu zvýšené výpočetní náročnosti.

Reference

[Ciar96]

Ciarlet, P and Lamour, F. *Does contraction preserve triangular meshes?* Numerical Algorithms, 1996

[Franz00]

Franz M., *Metody redukce trojúhelníkových sítí,* Diplomová práce, ZČU Plzeň, 2000

[Garland97]

Garland, M and P Heckbert. Surface Simplification Using Quadric Error Metrics. Proceedings of SIGGRAPH 97. pp. 209–216. 1997.

[Hoppe93]

H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. *Mesh optimization*. SIGGRAPH ’93 Proceedings, 1993.

[Low97]

Low, K and T Tan.Model Simplification Using Vertex-Clustering. Proceedings of 1997 Symposium on Interactive 3D Graphics.April 27–30, pp. 75–81, 188. 1997.

[Ros92]

Rossignac, J and P Borrel.Multi-Resolution 3D Approximations for Rendering Komplex Scenes. Technical Report RC 17687-77951. IBM Research Division, T J Watson Research Center, Yorktown Heights, NY. 1992.

[Schroeder92]

Schroeder, W J, J A Zarge, and W E Lorensen. Decimation of Triangle Meshes.Proceedings of SIGGRAPH 92. pp. 65–70. 1992.

[Schroeder97]

Schroeder, W. A Topology-Modifying Progressive Decimation Algorithm. Proceedings of IEEE Visualization ’97. pp. 205–212. 1997.

8 Přílohy

8.1 (A) Poznámky k implementaci

Programová část této práce byla realizována v prostředí MS Visual C++ 6.0 s použitím knihovny Coin (Open Inventor) verze 2.4.4.

URL : http://www.coin3d.org/

Vstupní i výstupní modely metod používají formát Open Inventoru (*.iv).

8.2 (B) Použité modely

Všechny modely použité k testování metod zjednodušování a srovnávání jejich výstupů jsou uvedeny v následující tabulce a jejich vzhled je vidět na obrázku 1.1.

Název	*Vertexů*	*Trojúhelníků*
bunny	35947	69451
horse	48485	96966
hand	327323	654666
dragon	437645	871414
happy	543652	1087716

Tabulka 8.1: Použité modely a jejich atributy

Obrázek 8.1: Vzhled použitých modelů

Všechny tyto modely pocházejí z internetové stránky Georgia Institute of Technology (GIT).

URL : http://www-static.cc.gatech.edu/projects/large_models/