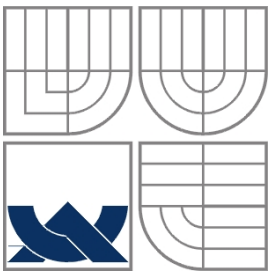
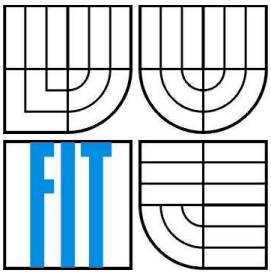


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMEDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ALGORITMY PRO ZJEDNODUŠOVÁNÍ MODELŮ

ALGORITHMS FOR POLYGON REDUCTION

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. Roman Schulz
AUTHOR

VEDOUCÍ PRÁCE — Ing. Ondřej Šiler
SUPERVISOR

BRNO 2008

Abstrakt

Cílem této práce je navrhnout a implementovat počítačový systém pro zjednodušení 3D modelů složených s trojúhelníkových sítí. Systém by měl umožňovat zjednodušit množství geometrických primitiv zadaneho objektu. Uživatel by měl mít možnost tento proces jednoduše ovlivnit výběrem metody, cílové velikosti souboru atp. Pro načítání a ukládání dat systém používá formát souboru 3DS. Čtenář by měl být schopen porozumět principům redukování modelů a měl by se seznámit s použitými algoritmy.

Klíčová slova

Počítačová grafika, úroveň detailu, redukce trojúhelníků, odstranění vrcholů, odstranění hran, shlukování vrcholů, odstranění trojúhelníků, metrika chybových kvadrik, hodnocení metrik, trojúhelníkové sítě, překlopení normály, měření chyby

Abstract

The purpose of this work is to design and implement a computer system, which should be used for simplification of a polygonal models in 3D space. The system should be allowed to reduce a given number of geometrical primitives from source object. User should have impact to the reducing process by selecting a method, size of target object etc. System uses 3DS file format for model loading and saving. Reader should understand principles of the polygon reduction and used algorithms.

Keywords

Computer graphics, level of detail, polygon reduction, vertex collapse, edge collapse, vertex clustering, triangle collapse, quadric error metrics, metric classification, triangle networks, mesh foldovers, measuring of error

Citace

Roman Schulz: Algoritmy pro zjednodušování modelů, diplomová práce, Brno, FIT VUT v Brně, 2008

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Ondřeje Šilera. Další informace mi poskytl pan Ing. Jan Pečiva. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Letovicích, dne 10. 5. 2008 ……

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu bakalářské práce Ing. Ondřeji Šilerovi a Ing. Janu Pečivovi, za jejich rady a připomínky k semestrálnímu projektu a diplomové práci.

Obsah

Seznam obrázků

Přílohy

1 Úvod

Počítačová grafika je obor informatiky, který využívá počítačovou techniku na vytváření umělých snímků (tzv. renderování). Tento obor je možné dělit na několik oblastí: 3D rendering v reálném čase (často využívaný v počítačových hrách), počítačová animace, video, střih speciálních efektů (často využívané ve filmu a televizi), oditování obrázků a modelování.

V grafických aplikacích, jako jsou například letecké simulátory nebo počítačové hry, je často potřeba zobrazit různé objekty. Ty většinou bývají vytvořeny v některém grafickém programu (3D Studio MAX, Blender, Cinema 4D, Rhinoceros, ..), nebo mohou být například pořízeny 3D Scannerem. Takto vytvořené objekty často bývají velmi detailní a obsahují velké množství nadbytečných informací, napřílkad když grafik vytváří model pomocí NURBS ploch, a výsledek automaticky exportuje na trojúhelníkovou síť. Při zobrazení takových objektů, nebo scén složených z desítek či stovek objektů, se naráží na problém spočívající v tom, že tyto objekty není možné zobrazit v reálném čase (při vysokém počtu zobrazených snímků za sekundu). Výkonnější počítač není řešením, protože je ve většině aplikací stejně potřeba zobrazit více objektů, než jeho výkon zvládne. Proto bylo vyvinuto několik algoritmů, které tento problém řeší tím, že takový objekt zobrazí v menším stupni detailu. Respektive vygenerují takový stupeň detailu objektu, který maximálně využije výkon počítače, aniž by jej zahltil.

Hlavním cílem této práce je vytvořit aplikaci která umožní snížit množství geometrických primitiv, ze kterých se objekt skládá. Dalším cílem je zpracování teoretické části, která by měla poskytnout čtenáři ucelený přehled o možnostech zobrazení objektů ve 3D grafice a popis principu nejčastěji používaných algoritmů. Naopak si tato práce neklade za cíl shromáždit vyčerpávající teoretickou studii, která by se zabývala podrobným popisem všech algoritmů a jejich variantami, nebo návrhem dalšího algoritmu. Samozřejmostí je popis důležitých částí zdrojového kódu, použitých tříd, metod, funkcí a struktur.

V první kapitole se čtenář seznámí s reprezentací objektů v počítačové grafice. Ve druhé kapitole jsou popsány nejpoužívanější metody používané pro zjednodušování objektů. Ve třetí kapitole jsou rozepsány metriky použitelné pro zjednodušení. Čtrvrtá kapitola se věnuje řešení problémů, které vznikají při redukci polygonálních modelů. Pátá kapitola je věnována popisu

implementace, včetně popisu jednotlivých částí aplikace. Šestá kapitola je věnována shrnutí dosažených výsledků, změření kvality a rychlosti jednotlivých metrik. Následuje závěr, vyhodnocení výsledků a popis možného využití aplikace. Za těmito kapitolami následují přílohy s naměřenými výsledky a seznam použité literatury.

1.1 Rozhraní pro zobrazení 3D scény

Zobrazení objektů ve 3D prostoru je výpočetně poměrně náročné, proto je vhodné využít některé již existující rozhraní. Rozšířená jsou v dnešní době rozhraní DirectX a OpenGL. Obě rozhraní se staly standartem, podporují i nejnovější vlastnosti grafických karet. Obě dvě jsou ve většině dnešních počítačích hardwarově akcelerovaná a tudíž velmi rychlá. Já jsem se pro zobrazení scény použil rozhraní Open Inventor, které pracuje jako vrstva nad OpenGL a zjednodušuje jeho ovládání. Hlavním důvodem k tomuto výběru bylo to, že rozhraní OpenGL dobře znám, narozdíl od konkurenčního DirectX, a že jej vyžadoval bývalý vedoucí projektu pan Ing. Jan Pečiva. Program byl vytvořen pro operační systém GNU/Linux, ale díky multi-platformním knihovnám jej lze přeložit a používat i v dalších systémech.

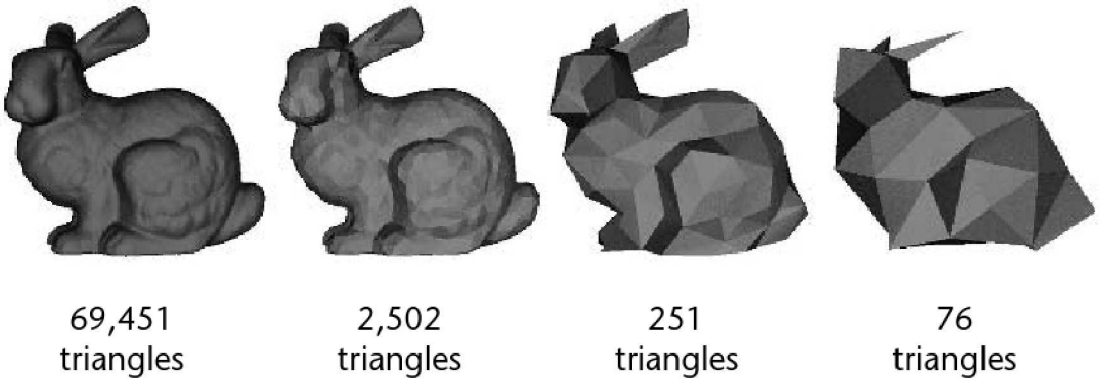
1.2 Reprezentace objektu jako trojúhelníkové sítě

Nejčastěji se v počítačové grafice na objekty pohlíží jako na seznam trojúhelníků, které udávají obálku objektu. Tyto trojúhelníky jsou navzájem spojeny v bodech a tvoří nepravidelnou trojúhelníkovou síť (irregular triangle network). Trojúhelníková síť je nejvíce patrná tehdy, pokud se zobrazí drátový model objektu (tj. když se zobrazí pouze hrany mezi trojúhelníky jako úsečky). Zjednodušování modelů spočívá ve zjednodušení této trojúhelníkové sítě tak, aby byl výsledný objekt co nejméně odlišný od původního.

1.3 Zobrazení různých úrovní detailu

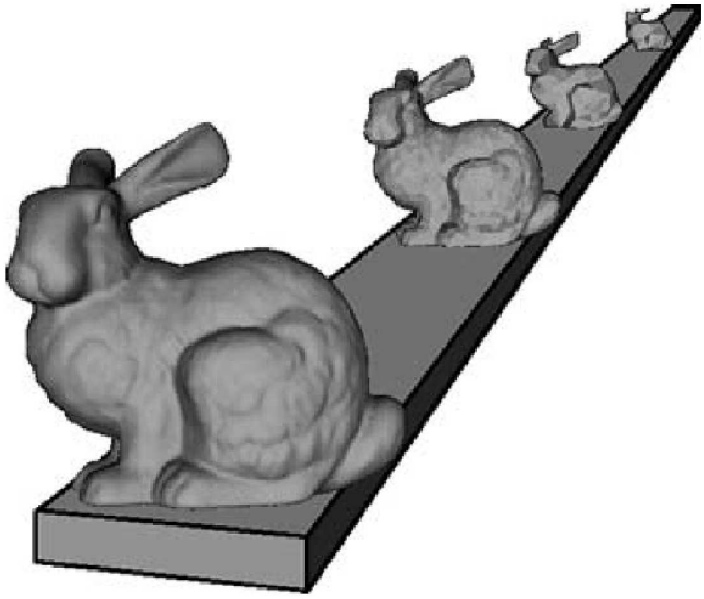
Často se stává, že objekty jsou rozmístěny v různé vzdálenosti od pozorovatele. Čím je objekt vzdálenější, tím je ve výsledném obrazu menší. Toho lze využít, vypočítat si menší stupeň detailu a objekt zobrazit mnohem rychleji v menším stupni detailu. Celá situace je znázorněna na obrázku 1, který je převzat z knihy [1], str. 5. Vlevo je zobrazen objekt v maximálním detailu a

směrem doprava se úroveň detailu snižuje k minimu. Tento objekt se jmenuje „Stanford Bunny“, byl pořízen 3D scannerem a obsahuje velké množství trojúhelníků. Proto se často používá k demonstraci a testování kvality algoritmů určených pro zjednodušování modelů.



Obr. 1: Jeden objekt v různém stupni detailu.

Obrázek 2 zobrazuje použití několika několika úrovní detailu v jedné scéně. Nejblíže je zobrazen objekt v maximálním stupni detailu, který se se vzdáleností od pozorovatele snižuje. Menší detail objektu vzadu je těžko postřehnutelný, i přesto že je složen z mnohem menšího počtu trojúhelníků. Výhodou že objekt s nejmenším stupněm detailu vpravo obsahuje 1000krát méně trojúhelníků než původní objekt, a bude přibližně 1000krát rychleji vykreslen na obrazovku.



Obr. 2: Zobrazení objektu s různou úrovní detailu v jedné scéně.

V současné době existuje několik typů technik, které jsou zaměřeny na různé typy objektů. Jsou to například metody pro zjednodušování křivek, výškových polí atd. Největší pozornosti se však dostává technikám zjednodušujícím objekty reprezentované nepravidelnými polygonálními povrchy. Na jedné straně jsou algoritmy, které pracují velice rychle, ale produkují méně kvalitní výsledky, na druhé pak velmi dobře aproximující algoritmy, které jsou zase velmi pomalé. Proto je vždy třeba přistoupit na kompromis mezi rychlostí produkce a kvalitou aproximace.

Pokud by scéna obsahovala 100 objektů „Standfor bunny“, nebyla by zobrazitelná v reálném čase, protože obsahuje 7 mil. trojúhelníků. Pokud se však ve scéně zobrazí 5 nejbližších objektů v nejvyšším detailu, 20 objektů v trochu menším stupni detailu, 25 objektů ještě v menším a 50 v nejmenším stupni detailu, bude třeba zobrazit pouze 400 tis. trojúhelníků, což je 5% z původního množst+ví a takovou scénu již zobrazí většina dnešních počítačů v reálném čase.

2 Algoritmy pro zjednodušování objektů

Algoritmů pro zjednodušování objektů existuje celá řada, liší se mnoha parametry, např. efektivitou, množstvím zabrané paměti, škálovatelností atd. Nejčastěji používané metody jsou popsány v knize Level of Detail for 3D graphics [1] v kapitole 2.

Zjednodušení modelu je založeno ztrátě detailu, který je možné zanedbat. Vždy je ale nutné zvážit, zda je třeba rychlé zobrazení a větší ztráta detailu, nebo pomalejší zobrazení s větším detailu. Typicky v počítačových hrách je možné přistoupit na menší kvalitu protože se zobrazuje velké množství objektů, kdežto při zobrazení např. medicínských dat je důležitější velký detail a rychlost zobrazení nehraje zásadní roli.

Další možnost vytvoření zjednodušených modelů je ruční práce, kdy grafik či animátor vytváří současně několik verzí objektu. Toto řešení je však velmi drahé a náročné na čas, ale výsledky jsou lepší, protože jen grafik ví, které oblasti jsou v objektu důležité a potřebují větší detail a které nikoliv.

2.1 Odstranění hran

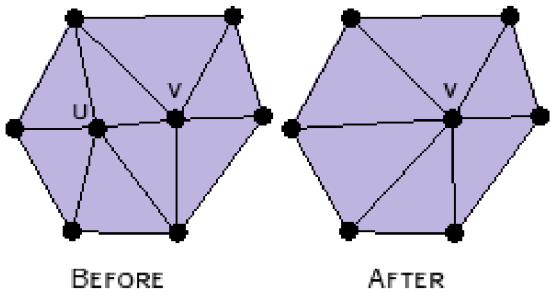
Tato metoda je označovaná jako edge collapse, je neupoužívanější ze všech, a hodí se pro obecné nepravidelné trojúhelníkové sítě. Algoritmus vybere podle některé metriky ze seznamu hran hranu *uv* s nejnižší vahou, z bodu *u* do bodu *v*, kde první bod má být přesunut do druhého. Poté jsou provedeny následující operace:

- Odstranění všech trojúhelníků, které obsahují oba body *u* i *v* (tj. hranu *uv*)
- Aktualizace všech ostatních trojúhelníků, které obsahují bod *u* tak, aby používaly bod *v* místo bodu *u*
- Odstranění bodu *u*

Každé provedení operace edge collapse odstraní tři hrany, dva trojúhelníky a jeden bod. Celý proces je opakován tak dlouho, dokud není dosaženo požadovaného množství trojúhelníků. Názorná ukázka odstranění hrany je předvedena na obrázku 3 (převzato z literatury [2]). Objekt s nízkým počtem polygonů (low polygonal model) je vytvořen tak, že se pokaždé vybere taková

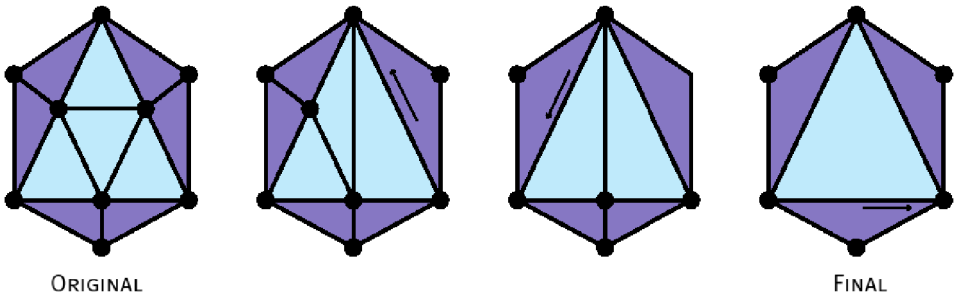
hrana, jejíž odstranění způsobí nejmenší vizuální změnu. Výběr hrany závisí na použité metrice, viz. kapitola 3.

Přestože je operace odstranění hrany jednoduchá, je třeba dát pozor na případ, kdy se normála některých okolních trojúhelníků přetočí na druhou stranu. O problematice řešení problémů je samostatná kapitola 4.



Obr. 3: Princip odstranění hran.

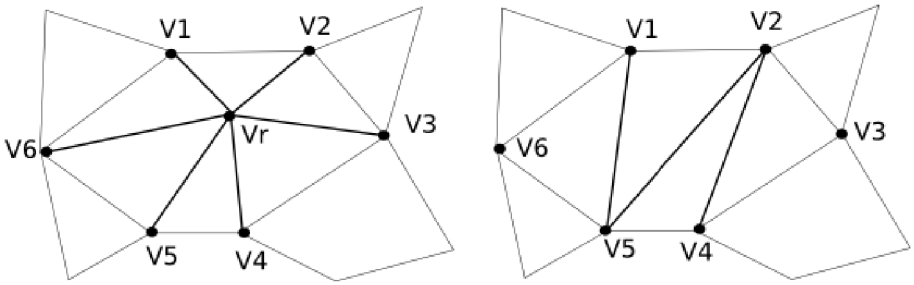
Tato metoda může být vylepšena tak, že se místo přesunu bodu *u* do bodu *v* vytvoří nový bod *w* mezi těmito body tak, aby došlo ok co nejmenší chybě. Tento výpočet je mnohem náročnější a je podrobně uveden v literatuře Hoppe [9]. Princip zjednodušení objektu pomocí několika odstranění hran je znázorněn na obrázku 4 (převzato z literatury [2]).



Obr. 4: Zjednodušení objektu pomocí odstranění několika hran

2.2 Odstranění vrcholů

Tuto metodu, označovanou jako Vertex decimation, uvedl v roce 1992 William Schroeder [6]. Spočívá ve výběru bodu s nejnižší vahou podle některé metriky ohodnocující váhu bodu. Poté dojde k ostranění vybraného bodu i všech přilehlých trojúhelníků. Nakonec dojde k vytvoření nových trojúhelníků (tzv. retriangulaci) v místě vzniklé díry. Princip je znázorněn na obrázku 5.



Obr. 5: Princip odstranění vrcholů.

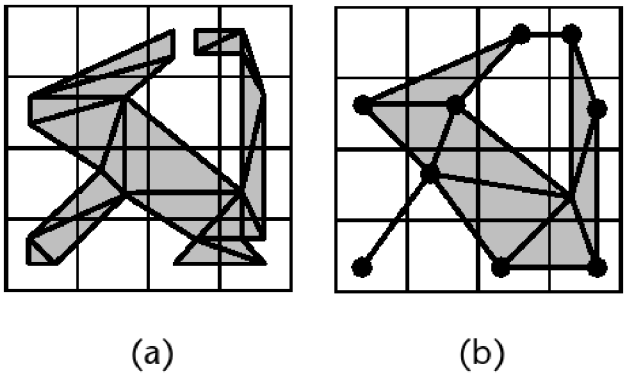
Vlevo je původní trojúhelníková síť, kde bod *Vr* byl vyhodnocen jako nejméně důležitý, proto došlo k ostranění bodu *Vr* a všech přilehlých trojúhelníků ($V_1V_2V_r$, V_2,V_3,V_r, atd.). Na tomto místě ale vznikla díra mezi body V_1-V_6, proto byly do sítě vloženy 4 nové trojúhelníky (obrázek vpravo).

Počet možných kombinací pro vyplnění vzniklé díry novými trojúhelníky je dán vztahem 1, kde *C(i)* je možnost kombinaci vyplnění díry která má i+2 stran. Kombinací vyplnění díry je mnoho a může být problém z nich rychle vybrat tu nejvhodnější.

$$C(i)=\frac{(2\mathrm{i})!}{(i+1)!\,i!} \tag{1}$$

2.3 Shlukování vrcholů

Tento algoritmus (označovaný jako Vertex clustering či Cell collapse) popsal J. Rossignac v [7]. Proces zjednodušení je založen na tom, že se okolo objektu vytvoří jemná mřížka. Všechny body v každé buňce mřížky jsou sjednoceny do jediného bodu a všechny hrany, které vedou z jednoho shluku do druhého, jsou sloučeny do jediné hrany. Tento proces je velice rychlý, ale podle literatury [1] dochází k viditelným deformacím objektu. Obrázek 6 znázorňuje princip shlukování vrcholů. Vlevo je originální objekt a v pravo výsledný, který vznikl shlukováním. Na obrázku je vidět, že výsledný bod nemusí být přímo ve středu buňky, ale může být například v místě, kde je buď nejdůležitější bod, nebo může být vypočten jako váhový průměr všech bodů v buňce.

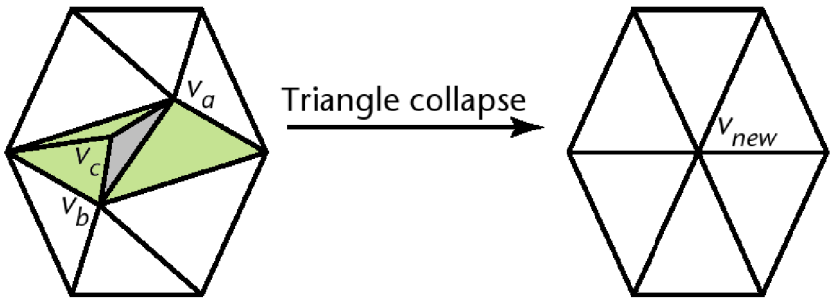


Obr. 6: Princip shlukování bodů.

Kvalita výstupu závisí na rozlišení mřížky, ale i přesto bývá velmi malá. Také není možné vytvořit výstup s předem stanoveným počtem trojúhelníků, protože jejich počet závisí pouze na velikosti mřížky. Případně může být výsledný bod dané buňky prezentován jako objem v prostoru (voxelová grafika). Tento altgoritmus byl původně navržen pro zjednodušení modelů vytvořených v CAD pro vizualizaci.

2.4 Odstranění trojúhelníků

Tato metoda (Triangle collapse) vybere nejméně důležitý trojúhelník $v_a v_b v_c$ a všechny jeho body spojí do jediného bodu. Při každé operaci dojde tak k ostranění 4 trojúhelníku, 6 hran a 3 bodů. Nový bod je buď jeden z v_a,v_b, v_c, nebo libovolný bod uvnitř původního trojúhelníku. Princip algoritmu odstranění trojúhelníku je znázorněn na obrázku 7 (převzato z [1]).



Obr. 7: Princip odstranění trojúhelníku.

Pokud je šedě vyznačený trojúhelník vybrán pro odstranění, dojde současně k odstranění přilehlých trojúhelníků vyznačených zeleně. Operace odstranění trojúhelníku je ekvivalentní dvěma operacím odstranění hrany, ale výpočet je rychlejší a navíc potřebuje k výpočtu méně paměti.

3 Chybové metriky

V předchozí kapitole byly popsány metody pro zjednodušení objektů. Předchozí metody však ke své práci potřebují algoritmy, které určí které hrany či trojúhelníky jsou nejméně důležité a lze je tedy odstranit. Temto algoritmům se říká chybové metriky, a jedná se vlastně o měření chyby, kterou způsobí zjednodušení či odstranění některé části objektu. To, jakým způsobem se výsledná chyba měří, zásadně ovlivňuje kvalitu zjednodušeného modelu, proto je dobré zvážit zda použít rychlou méně kvalitní metriku, nebo pomalejší a kvalitnější metriku.

3.1 Délka hrany

Jedná se o nejjednoduší algoritmus použitelný pro metodu edge collapse. Výpočet váhy jednotlivých hran závisí pouze na vzdálenosti bodů *u* a *v* se vypočítá podle vztahu 2:

$$cost(u,v)=\|u-v\|=\sqrt{(u_a-v_a)^2+(u_b-v_b)^2+(u_c-v_c)^2} \tag{2}$$

Metoda je však nejrychlejší a produkuje poměrně dobré výsledky. Jelikož se váha hrany používá pouze pro porovnání jednotlivých hran navzájem, je možné výpočet optimalizovat a z výpočtu velikosti vektoru *uv* vynechat operaci odmocňování.

Výsledek této metriky je takový, že se z objektu odstraňují hrany, které jsou nejkratší tak, až v něm zůstanou pouze delší hrany. Kratší hrany jsou v dálce méně viditelné než delší hrany, proto je možné je odstraňovat nejdříve a nedojde k viditelnému zkreslení při zobrazení objektu v dálce.

3.2 Metrika založená na tvaru objektu

Tento algoritmus popsal Stan Melax v Game Developer Magazine [2] v roce 1998, jedná se o metriku, která je použita ve hrách od společnosti Bioware. Byla použita v hrách Baldur's Gate, MDK 1&2, Neverwinter Nights a mnoha dalších. Výpočet váhy jednotlivých hran je dán vztahem 3:

$$cost(u,v) = \|u-v\| \times \max_{f \in Tu} \left\{ \min_{n \in Tuv} \left\{ (1 - f.normal \circ n.normal) \div 2 \right\} \right\} \quad (3)$$

kde *Tu* je množina trojúhelníků obsahujících bod *u*, a *Tuv* je množina trojúhelníků obsahujících body *u* a současně *v*. Metrika je tedy definovaná jako součin délky hrany a vztahu závisejícím na křivosti okolí hrany. Díky výpočtu skalárního součinu (úhlu dvou sousedních trojúhelníků) metoda zachovává ostré hrany – viz. ukázky u hodnocení metrik. Váha sjednocení bodu *u* do bodu *v* je zpravidla odlišná od sjednocení *v* do *u*.

Tato metrika podává v porovnání s ostatními metodami velmi kvalitní výsledky, výpočet je ale náročnější protože používá výběr z mixima z minim hodnoty závisející na skalárním součinu. Dále v textu bude tato metrika označena GDM.

3.3 Metrika Chybových kvadrik

Tuto metodu představili M. Garland a P. Heckbert, na konferenci Siggraph [5]. Metoda se označuje Quadric error metrics a často se zapisuje zkratkou QEM. Metoda je velmi rychlá a není paměťově náročná. Chybová kvadrika každé plochy je určena vztahem 4. Jedná se o symetrickou matici 4x4, která je definovaná jako součin vektoru normály plochy *p* s transponovaným vektorem p^T a obsahuje 10 různých reálných čísel. Kvadrika plochy *p* má následující tvar:

$$Q_p = p.p^T = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} . \begin{bmatrix} a & b & c & d \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & ab & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \quad (4)$$

Hodnoty *a, b, c, d* jsou koeficienty z obecné rovnice roviny plochy určené vztahem 5. Vektor [*a b c*] je normála roviny a *d* vyjadřuje posun vzhledem k počátku souřadného systému.

$$\begin{aligned} ax+by+cz+d &= 0 \\ a^2+b^2+c^2 &= 1 \end{aligned} \tag{5}$$

Kvadrika Q_v se určí součtem všech kvadrik Q_p v plochách p, které obsahují bod v.

$$Q_v = \sum_{p \in planes(v)} Q_p \tag{6}$$

Výsledná chyba v každém nově vypočítaném bodu se vypočítá podle vztahu 7, kde v je poloha nově vypočteného bodu:

$$E_v = \sum_{p \in planes(v)} (p.v)^2 = \sum_p (v^T . p)(p^T . v) = v^T [\sum_p (p.p^T)] v = v^T \sum_p Q_p v = v^T Q_v v \tag{7}$$

Podle výše uvedeného vztahu je příspěvek každé plošky do celkové chyby dán vynásobením kvadriky Q_v transponovaným vektorem v^T a na výsledek se provede skalární součin s vektorem v. Při sjednocení dvou sousedních bodlů se podle literatury [5] vypočte nová kvadrika jako součet kvadrik Q_v v obou bodech (maticový součet)

Algoritmus výpočtu je následující:

- Výpočet kvadrik pro všechny plochy
- Výpočet kvadrik pro všechny body
- Pro všechny hrany je pomocí nové kvadriky $Q_w = Q_u + Q_v$ vypočtena chyba E_v, která vznikne odstraněním hrany uv
- Nalezení hrany s nejnižší chybou E_v a její odstranění. $Q_v = Q_w$
- Opakování od bodu 3. dokud není odstraněn požadovaný počet hran

3.4 Další metriky

Metrik pro ohodnocení důležitosti vrcholů existuje mnohem více, jejich použití ale není příliš časté. Některé z nich byly vyvinuty pro konkrétní účely a nejsou příliš obecné. Níže uvedené metriky nejsou zahrnuty v implementaci.

3.4.1 Metrika průměrné normály

Tato metrika (Average normal metrics) určuje důležitost vrcholu podle odchylek normál trojúhelníků přilehlých k danému bodu od průměrné normály. Hodnota leží v intervalu <0,1>, kde hodnota 0 značí rovinu okolo zkoumaného vrcholu. Byla popsána v literatuře Junker [10].

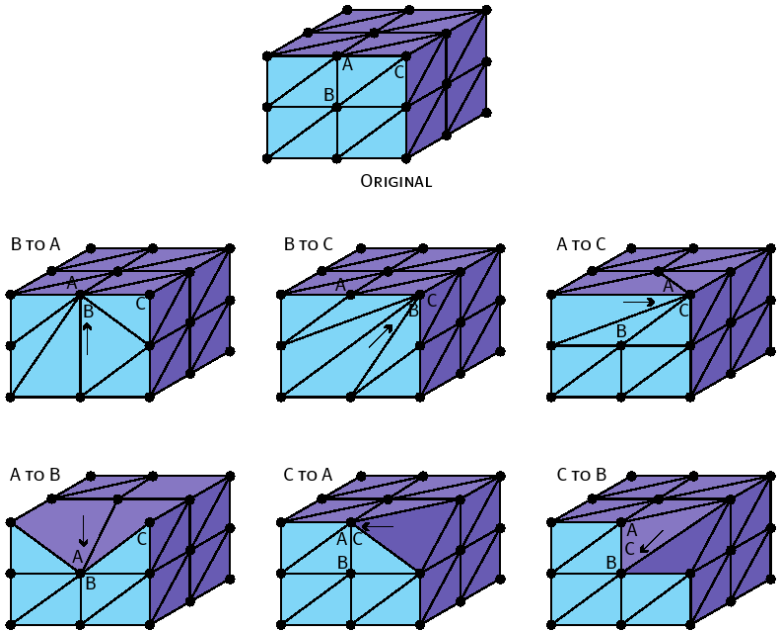
3.4.2 Metrika rozdílu výšek

Metrika rozdílu výšek, označovaná jako Height difference heuristic, sčítá hodnoty rozdílu výšky (u modelu terénu nadmořské výšky) mezi samotným vrcholem a jeho okolím. Tato metrika byla popsána v Junker [10]. Tato hodnota se bere jako míra důležitosti. Pokud je důležitost rovna nula, znamená to, že výška je určená zprůměrováním výšek jeho sousedů. Metrika byla vyvinuta za účelem zjednodušování terénů, čímž ztrácí na obecnosti, protože pracuje s pravidlenou trojúhelníkovou sítí.

3.5 Hodnocení metrik

Obrázek 8 ukazuje testovací příklad, jedná se o krychli složenou s 26 bodů a 32 trojúhelníků. V ideálním případě je požadováno, aby zjednodušený objekt byla krychla stejné velikosti jako původní, s tím rozdílem, že má obsahovat pouze 8 bodů a 12 trojúhelníků.

Na obrázku jsou znázorněny body A, B a C, a jsou zde znázorněny hrany vybrané metrikou pro algoritmus edge collapse. Odstranění hran BA, BC nebo AC nezpůsobí žádnou chybu (horní tři krychle). Výběrem hran AB, CA nebo CB však dojde k viditelné deformaci objektu (spodní tři krychle). To ale neznamená že se takové metody nepoužívají, naopak mají velkou výhodu ve své rychlosti, pokud je možné se dopustit nepřenosti.



Obr. 8: Objekt pro testování kvality metrik.

Například metrika, jejíž výpočet je založen na délce hran, vybírá i hrany uvedené na spodních 3 příkladech, pokud je hrana dostatečně krátká. Tato metoda je ale poměrně rychlá (viz. srovnání metod v kapitole 6), stejně jako metrika chybových kvadrik.

Metrika z GDM i QEM, na rozdíl od předchozí vybírá pouze z prvních 3 tu hranu, jejíž odstraněním dojde k nejmenší chybě, díky tomu, že bere v potaz i úhly se sousedními trojúhelníky, naopak je mnohem pomalejší. Bližší informace o výsledku testů jsou popsány v kapitole 6.1

3.5.1 Měření chyby

K dosažení kvalitního výstupu je vhodné měřit vzniklou chybu nového objektu a přestat se snižováním kvality, pokud chyba naroste nad určitou mez. Podle práce Garland [5] může být chyba měřena jako průměr druhé mocniny vzdálenosti redukovaného a původního objektu následovně podle vztahu 8:

$$
\begin{aligned}
E_i &= \frac{1}{|X_n|+|X_i|}\left(\sum_{v\in X_n} d^2(v, M_i)+\sum_{v\in X_i} d^2(v, M_n)\right) \\
d(v, M) &= min_{p\in M}\|v-p\|
\end{aligned}
\tag{8}
$$

kde X_n a X_i jsou body objektů M_n (původní model) a M_i (zjednodušený model), funkce *d(v,M)* udává minimální vzdálenost bodu *v* k ploše z objektu *M*.

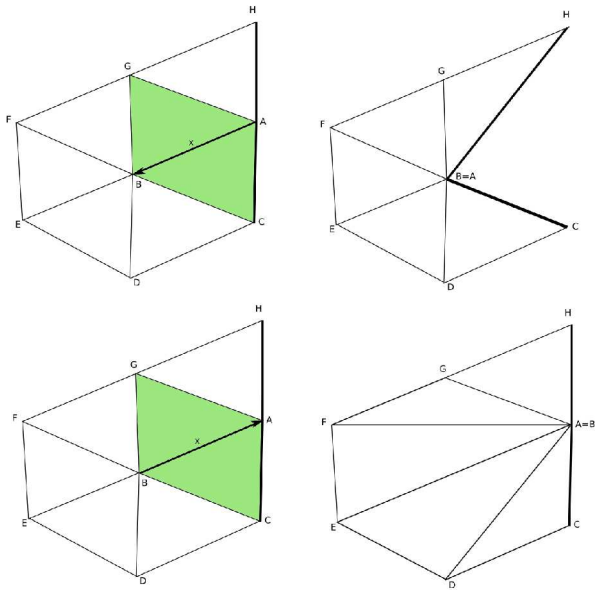
Existuje však program Metro, který vypočítá rozdíl mezi dvěma povrchy (mezi objektem a jeho zjednodušenou verzí). Tento program implementuje metody pro měření vzdálenosti bodu a plochou, výsledek vrací numerický (maximální a průměrná chyba, rozdíl těles, rozdíl povrchů, atd.), a vizuální, kdy části objektu barevně obarví podle velikosti chyby. Může měřit vzdálenost bodů z originálního objektu, či zvolit vlastní množinu bodů metodou Monte Carlo. Program Metro pro Linux i Windows včetně zdrojových kódů je k dispozici ke stažení zdarma v [8].

4 Řešení problémů

Při zjednodušování většiny objektů je možné narazit na několik druhů problémů, obvzláště u rovinných útvarů s okrajem (typicky výškové mapy, terén).

4.1 Zachování okrajů objektu

Algoritmus je třeba doplnit tak, aby zachovával okraje objektu. Na obrázku níže jsou dvě situace. Tučně zvýrazněná přímka mezi body H, A a C je hranice objektu. V prvním případě má nejmenší chybou spojení bodu A do bodu B (obrázek nahoře vlevo). V tomto případě ale dojde ke změně hranice objektu (obrázek nahoře vpravo) a je vhodné takové operaci zamezit. Celá situace je znázorněna na obrázku 9.



Obr. 9: Zachování okrajů.

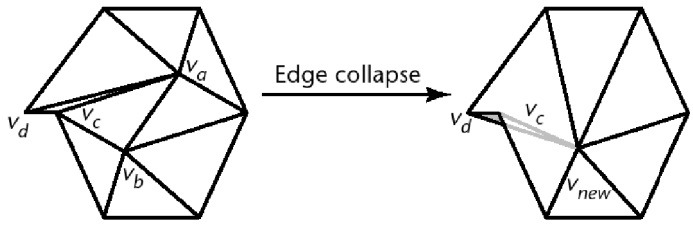
Ve druhém případě má nejmenší chybu spojení bodu B do bodu A. V tomto případě nedojde ke změně hranice objektu a operace může být bez obav provedena. Tato situace je znázorněna na spodních dvou obrázcích. Z výsledku plyne, že je třeba zamezit odstranění hrany z bodu A do

bodu B, kde A náleží hranici objektu. Metoda by tuto hranu měla ignorovat a místo ní vybrat další na řadě nejméně důležitou. Opačným směrem je možné hranu odstarnit.

Zachování okrajů u terénu je velice důležité, u uzavřených objektů se tento problém většinou nevyskytuje. Pokud ano, jde o hranici více materiálů. V takovém případě je dobré zamezit zjednodušení hrany, která začíná na hranici materiálu. Dále je dobré zamezit zjednodušení hrany, která je na hranici objektu (např. hrana HA z obrázku 9) buď tak, že se zjednodušení takové hrany úplně zakáže, nebo že se přidá podmínka, aby úhel HAC byl menší než požadovaná hodnota.

4.2 Překlopení normály

Překlopení normály (mesh foldovers, mesh inversion) je nepředvídatelný efekt některých operací edge collapse. Na obrázku je znázorněn trojúhelník $v_a v_c v_d$. Pokud je hrana $v_a v_b$ sjednocena do nového bodu v_{new}, stane se, že se normála trojúhelníku $v_a v_c v_d$ otočila o 180° (tj. překlopila se dozadu). Viz obrázek 10 (převzato z literatury [1]):



Obr. 10: Překlopení normály.

Na obrázku vlevo má sledovaný trojúhelník normálu přilehlou k pozorovateli, v pravo je normála směrem od pozorovatele. Tato situace může být ošetřena detekováním změny normálového vektoru všech sousedů před a po odstranění hrany. Tato změna by neměla být větší než 90° (lépe je zvolit maximální odchylku nové normály například okolo 40°) a je vhodné větší změně zamezit výběrem jiné hrany k odstranění. Překlopením normály vznikají nepříjemné artefakty.

5 Popis implementace

Aplikace je psána jako multiplatformní, lze ji spustit v prostředí GNU/Linux (testováno v distribucích Debian Sarge/Etch/Lenny, Ubuntu 6.10/7.10, Red Hat Enterprise), FreeBSD (testováno ve verzi 6.3) i v MS Windows (testováno ve verzi XP) na architektuře 32bitů i 64bitů. Aplikace se ovládá z příkazové řádky, díky tomu umožňuje jednoduše spustit redukci mnoha objektů pomocí skriptu. Na obrázku 11 je znázorněno blokové schéma aplikace. Program je rozdělen na několik samostatně fungujících částí:

- Aplikace – Zajišťuje ovládání veškerých procesů, provádí zpracování parametrů příkazové řádky pro zjednodušování objektů (parametry zpracovává pomocí getopt), stará se o měření času výpočtu, volání správných handlerů pro daný 3d grafický formát a hlavně spouštění všech výpočtů.
- 3DS Handler – Obsahuje rutiny pro načítání a ukládání zjednodušených 3DS souborů. Aplikace podporuje přibližně 100 druhů bloků dat formátu 3ds, některé jsou podporovány tak, že je aplikace umí přečíst a uložit aniž by jim rozuměla a parsovala je (Zachovává nastavení scény, materiály atd.), jiné jsou podporovány plně (Načtení geometrie, orientace atd.). Důvod k tomuto řešení je ten, že ne všechny bloky dat 3ds souboru jsou veřejně specifikovány.
- IV Handler – Rutiny pro načítání Open Inventor scény. Používá knihovnu Coin (v GNU/Linuxu se jedná o balíčky libcoin40, libcoin40-dev, libsoqt), která je implementací Open Inventoru. Pokud v systému není Open Inventor nainstalován, je možné program zkompilovat i bez podpory tohoto formátu.
- OFF handler – Obsahuje kód pro ukládání do formátu OFF, který byl napsán pouze pro výměnu dat s programem pro měření odchylky (Program Metro nenačítal žádný podporovaný formát, proto jsem tento jednoduchý geometrický formát implementoval)
- Display – Zobrazení objektu v samostatném okně. Pro použití je nutné mít nainstalovanou knihovnu Coin. Pro zkompilování projektu s podporou zobrazení je nutné odkomentovat makro USE_INVENTOR v main.h. Defaultně je zakomentované protože jsem chtěl, aby program běžel bez problémů s knihovnami na co nejvíce platformách.
- Edge Collapse – Techniky pro zjednodušování modelů. Program používá pro zjednodušení techniku odstranění hran s metrikami délky hran (edge size), chybovými

kvadrikami (QEM) a metrikou uvažující tvar objektu (GDM). Tyto metriky jsou popsány v kapitole 3.

Pro načítání .iv objektů používá Open Inventor, načítání .3ds a .off objektů je řešeno vlastním kódem. Pro zobrazení je použita knihovna Open Inventor. Zjednodušené objekty je možné uložit zpět do .3ds souboru.

5.1 Popis ovládání

Aplikace se překládá pomocí programu make a gcc. Výsledkem je jediný soubor reducer (popř. reducer.exe ve Windows). Ten lze spustit s mnoha různými parametry

./reducer <parametry>

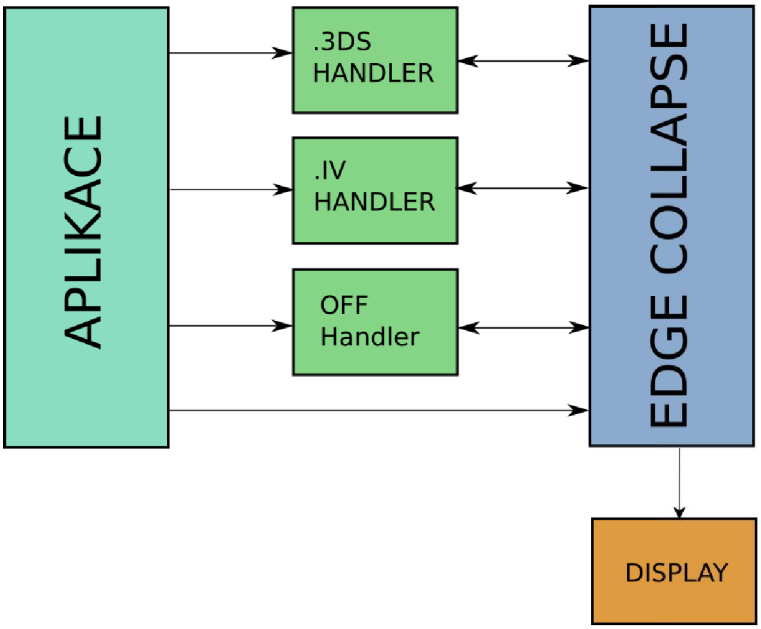
Parametry můžou být:

- -i <model>: Cesta k modelu, který je třeba zjednodušit. Jedná se o umístění ve file systému, musí obsahovat koncovku 3ds nebo iv. Jedná se o povinný parametr.
- -o <model>: Cesta, na níž se uloží redukovaný model. Jedná se o umístění ve file systému, musí obsahovat koncovku 3ds, iv nebo off.
- -q <kvalita>: Kvalita výsledného modelu (0-100, 100=bez změn, 0=maximální možná redukce)
- -a <algoritmus>: Nastavení algoritmu, který se má použít pro zjednodušení modelu. K dispozici jsou možnosti EDGESIZE, EDGEANGLES a QUADRICS
- -b: Přepínač, který nastaví že není potřeba zachovávat hranici objektu. Některé nedostatky použitých algoritmů jsou řešeny tak, že je zakázáno redukovat i trojúhelníky na hranici objektu (viz. kapitola 4.1). Tato volba takové zjednodušení zakáže.
- -j: Přepínač, který nastaví, že není potřeba spojovat 2 různé body na stejných souřadnicích do jednoho. Aplikace standardně slučuje vertexy. U některých otexturovaných objektů se díky tomu může vyskytnout špatné mapování textur u zjednodušeného modelu. Tato volba spojování vertexů zakáž. Nespojené body ale nikdy nevytvoří trojúhelníkovou síť, kterou je možné redukovat.

- -e <epsilon>: Nastaví maximální vzdálenost dvou bodů, které se mají spojit do jednoho. Některé objekty jsou uloženy tak, že pokud se jeden bod vyskytne ve 2 polygonech, je v souboru uložen 2x. Jelikož algoritmy pro zjednodušování modelů pracují s trojúhelníkovou sítí kde je každý bod uložen pouze jednou, je nutné stejné body pospojovat. Tato vzdálenost závisí na měřítku objektu.
- -h: Zobrazí nápovědu programu

Součástí odevzdané práce jsou následující soubory:

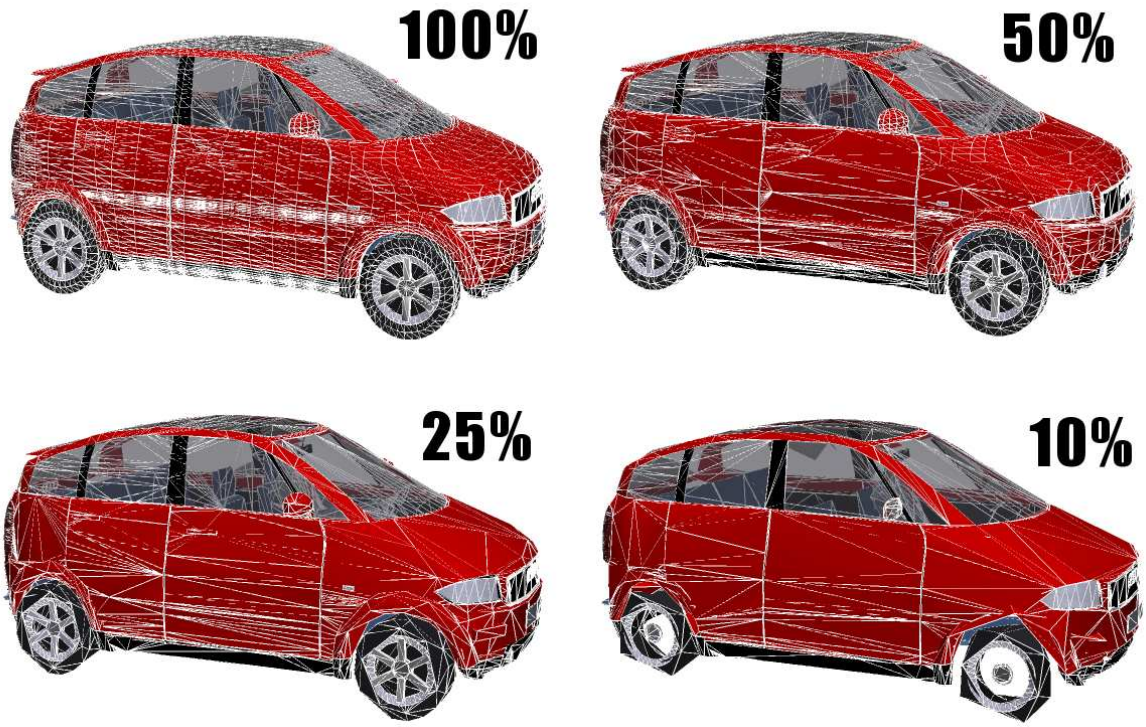
- adresář src – Obsahuje zdrojové kódy programu, včetně Makefile pro Linux a Windows
- adresář doc – Obsahuje použitou dokumentaci, dokumentaci vygenerovanou ze zdrojových kódů, technickou zprávu včetně PDF a ODT formátu a obrázky použité ve zprávě
- adresář data – Obsahuje objekty, na nichž probíhalo testování. Jedná se o sadu objektů, které jsou podrobněji popsány v tabulce v kapitole 6
- reducer.exe – Spustitelný soubor pro Windows
- metro.exe – Nástroj pro měření chyby zjednodušeného objektu
- readme.txt



Obr. 11: Blokové schéma aplikace.

6 Dosažené výsledky

Na obrázku 12 je uvedeno několik aproximací modelu Audi A2. Zároveň je vidět srovnání výsledných modelů po zjednodušení kvality na 50%, 25%, 10%. Původní objekt měl 60.000 polygonů a zjednodušené mají 30.000, 15.000 a 6.000 polygonů. Zobrazen je jak povrch modelu, tak i jeho trojúhelníková síť. Při vyšším stupni redukce (objekt se 6.000 trojúhelníky) sice dochází ke ztrátám detailů, například v oblasti kol, ale i přesto je výsledek dobře použitelný. Na modelu lze pozorovat větší hustotu trojúhelníků v oblasti křivých ploch, než v oblasti rovin, což je dobře. V tomto případě je při použití algoritmu GDM patrné zachování tvaru i po 90% redukci.



Obr. 12: Zjednodušení objektu se 60 000 trojúhelníky

Pro ohodnocení odchylky redukovaného modelu byl použit program Metro. Pracuje na principu měření Hausdorffovy vzdálenosti mezi dvěma plochami (tvořenými trojúhelníkovou sítí). Výpočet je proveden tak, že je na ploše prvního modelu vygenerováno několik stovek až tisíc bodů a ke každému z těchto bodů je hledána nejkratší vzdálenost k ploše ve druhém modelu. Nejkratší vzdálenost se vypočte i obráceně a potom Hausdorffova vzdálenost udává maximum z vypočítaných hodnot. Bližší popis výpočtu je uveden v literatuře [1] na stranách 50-53 v kapitole Geometric Error. Rozmístění bodů na ploše je možné nastavit, buď je lze rozložit pravidelně (půlením trojúhelníků) nebo náhodně (Monte Carlo).

Tento program je k dispozici pro Windows i Linux, nevýhodou je že nepodporuje načítání 3ds souborů. Tento handicap jsem vyřešil tak, že jsem napsal export do jednoduchého formátu, který Metro podporuje.

Všechny testy jsem prováděl na sadě modelů s následujícími parametry:

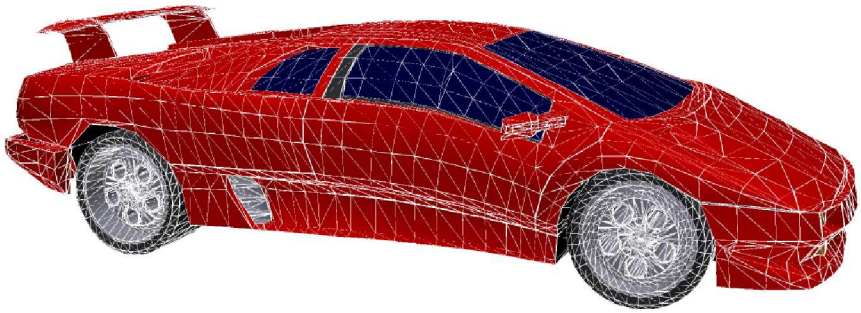
Soubor	Velikost [kB]	# trojúhelníků	# bodů
lamborgini.3ds	346,1	16515	9465
spitfire.3ds	269,3	11356	5751
jeep.3ds	67,1	2032	1190
dolphins.3ds	34,6	1692	855
box3.3ds	1,9	48	26

Pro každý model jsem spustil 60 testů, celkem bylo provedeno 300 testů. Pro každý model jsem vypočítal odchylku pro algoritmus edge size (Chyba závislá na délce hrany), edge angles (Algoritmus z Game Developers Magazine) a quadrics (Metrika chybových kvadrik). Zároveň byl měřen čas délky výpočtu. Toto všechno bylo měřeno pro kvalitu od 5% do 95% s krokem po 5 procentních bodech.

6.1 Porovnání kvality algoritmům

Obecně jsem zjistil, že se každý algoritmus lépe hodí na jiný model. Nejhorší výsledky poskytoval algoritmus edge size. V příloze 8.1 jsou vyneseny grafy závislosti Hausdorffovy chyby na míře redukce. Vodorovná osa udává míru redukce, 5% je nejmenší redukce a 95% je největší. Při měření byl stanoven krok 5 procentních bodů. Svislá osa udává naměřenou chybu, hodnotou je Hausdorffova vzdálenost, která je závislá na měřítku objektu. To znamená že objekt,

jehož souřadný systém je v centimetrech má nižší hodnoty Hausdorffovy vzdálenosti než objekt, jehož souřadný systém je v metrech. Při testu neporovnávám jednotlivé objekty mezi sebou, ale kvalitu různých algoritmů vždy na jednom objektu. Proto je možné s takto naměřenou chybou pracovat.

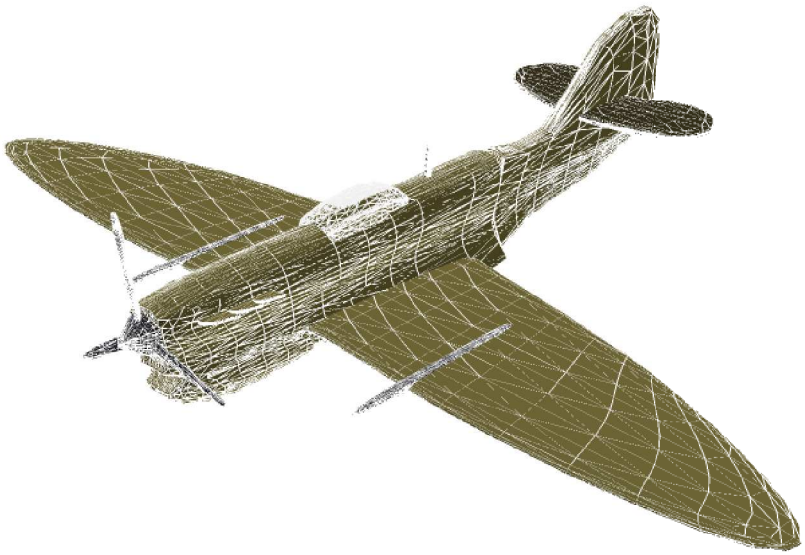


Obr. 13: Testovací objekt lamborgini.3ds

První testovaný objekt byl jeep.3ds. Tento objekt by se dal přiřadit do skupiny objektů s nízkým počtem polygonů, u kterých je další redukce poměrně složitá. U tohoto objektu si podával největší chybu algoritmus redukující pouze pomocí délky hrany. U něj je při 10% redukci chyba stejně velká, jako při 45% redukci u algoritmu edge angles, nebo při 55% redukci s použitím metriky chybových kvadrik. To je rozdíl velmi propastný a naznačuje malou použitelnost algoritmu v praxi. Ostatní 2 algoritmy podávají podobné výsledky, edge angles je podstatně lepší při redukci 0-50%, chybové kvadriky jsou naopak lepší při redukci 50-90%.

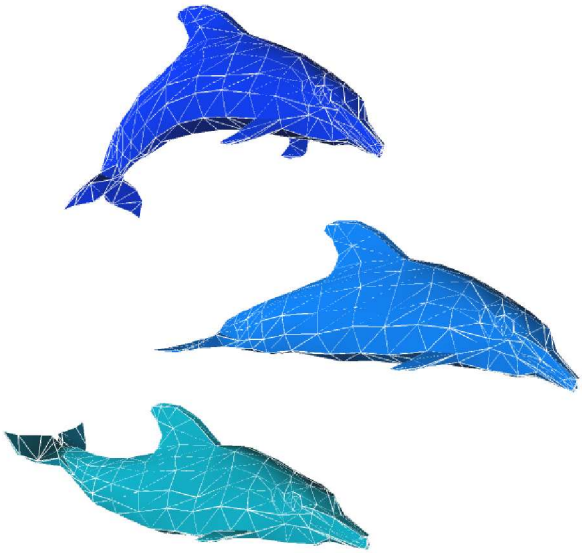
Obr. 14: Testovací objekt jeep.3ds

Druhý testovaný objekt je lamborgini.3ds. Tento objekt se složen z 16 000 polygonů a obsahuje velké množství polygonů, které lze bez způsobení větší chyby odstranit. Při 40% redukci způsobí všechny algoritmy přibližně stejnou chybu. Při dalším zvyšování stupně redukce má nejmenší chybu algoritmus chybových kvadrik, který následovaný metodou edge angles a opět největší chybu způsobil algoritmus edge angles.



Obr. 15: Testovací objekt spitfire.3ds

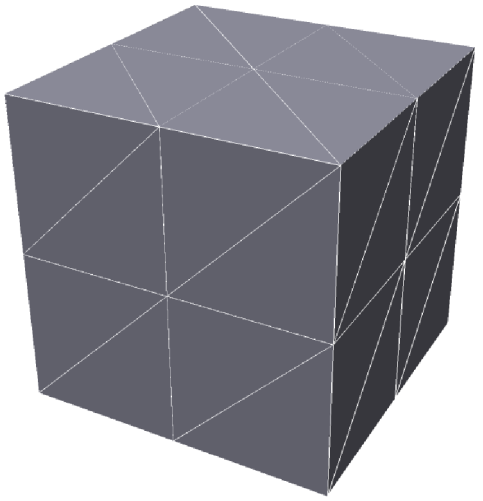
Dalším testovaným objektem byl model letadla spitfire.3ds. Tento objekt je složen z 11 000 polygonů. Z nezjištěných důvodů nejhůře dopadl algoritmus chybových kvadrik. Už při 30% redukci způsobil chybu větší chybu, než oba dva zbývající algoritmy při redukování 95% polygonů. O jeden řád menší chybu měl algoritmus edge size, a od dva řády menší chybu způsobil algoritmus edge angles. Algoritmus edge angles u tohoto modelu vyšel při všech stupních redukci nejlépe.



Obr. 16: Testovací objekt dolphins.3ds

V pořadí čtvrtým testovacím objektem byl dolphins.3ds. Tento lze zařadit do skupiny objektů s nízkým počtem polygonů. Na rozdíl od jeep.3ds je charakteristický tím, že neobsahuje žádné ostré hrany, tudíž se dá předpokládat že půjde zredukovat všemi algoritmy s malou chybou. Tento předpoklad se potvrdil, již při letmém pohledu byl redukovaný objekt téměř nerozpoznatelný od originálu i při redukci 60%. Rozptyl naměřené chyby pro různé algoritmy je u tohoto objektu menší než u ostatních objektů. Nejlépe dopadl algoritmus edge angles, následovaný algoritmy edge size a chybovými kvadrikami.

Posledním testovaným objektem je box3.3ds. Tento objekt byl popsán v kapitole 3.5, jako speciální testovací případ, ve kterém se projeví nedostatky některých algoritmů. Tady dopadl nejhůře algoritmus založený na délce hran. U něj byla změřena jistá chyba už při 5% redukci a s dalším zjednodušování rostla. Oba ostatní algoritmy nezpůsobily žádnou chybu i při 60% redukci.

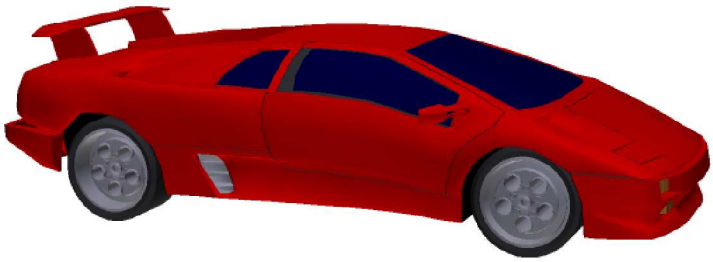


Obr. 17: Testovací objekt box.3ds

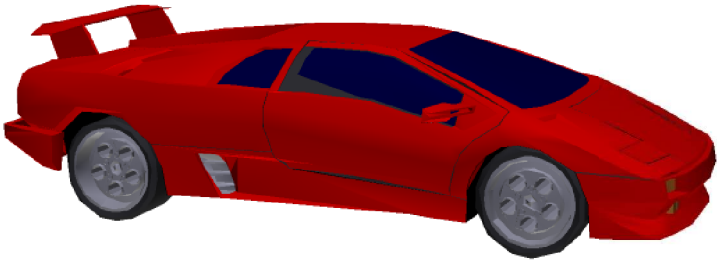
Vizuální měření kvality je poměrně subjektivní záležitost, proto jsem výsledky konzultoval s dalšími lidmi, aniž by věděli o kterou metriku ze tří testovaných se jedná. Ti se shodli na tom, že nejlepší výsledky podává algoritmus edge angles. Tento algoritmus a metrika chybových kvadrik také správně redukuje objekt z kapitoly 3.5, nejdříve v něm redukuje hrany, jejichž odstraněním nevznikne žádná chyba, a až potom vše ostatní. U algoritmu založeném na délce hran se vždy najde nějaká špatně redukovaná hrana, která způsobí zbytečnou chybu.

Obecně tedy nelze určit který algoritmus je nejlepší, protože záleží na redukovaném modelu, i na stupni kvality. Často se stává že při malé redukci je lepší jeden algoritmus a při zvýšení míry redukce naopak jiný algoritmus. Navíc u jiného modelu může být situace opačná. Proto je dobré s redukovaným modelem experimentovat, a z několika výsledků výbrat ten nejlepší. Toho by se dalo dosáhnout tak, že se zvolí požadovaný stupeň redukce a provede se redukce se všemi algoritmy. Pomocí programu Metro by se změřila odchylka všech tří a vybral by se ten s nejmenší chybou.

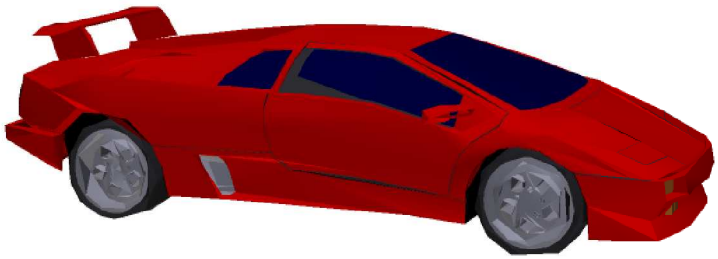
Na obrázcích 18 – 20 je uveden příklad zjednodušení objektu lamborgini.3ds. První obrázek znázorňuje původní objekt, na druhém je zjednodušení 50% a na třetím 70%. Zjednodušení tohoto objektu proběhlo metodou chybových kvadrik.



Obr. 18: Objekt lamborgini.3ds – originál (16000 polygonů)



Obr. 19: Objekt lamborgini.3ds – redukce 50% (8000 polygonů)



Obr. 20: Objekt lamborgini.3ds – redukce 70% (4500 polygonů)

6.2 Porovnání rychlosti algoritmů

Měřením rychlosti algoritmů bylo zjištěno, že metrika chybových kvadrik a zjednodušení podle délky hran trvají přibližně stejnou dobu. Odchylky rychlosti těchto dvou algoritmů se pohybují u všech testovaných případů do 5%. Srovnámím algoritmu GDM a zbylých dvou algoritmů bylo zjištěno, že je přibližně 5x pomalejší. To prakticky vylučuje jeho použití v aplikacích, kdy je třeba mít rychle vypočítaný objekt s nižším množstvím detailu.

Pokud záleží na rychlosti výpočtu, vychází z mého testování nejlépe algoritmus chybových kvadrik. Pokud na rychlosti výpočtu nezáleží, volil bych buď opět chybové kvadriky nebo GDM. Naopak algoritmus redukující podle délky hrany bych nedoporučil, protože je stejně rychlý jako chybové kvadriky, ale podává mnohem horší výsledky.

7 Závěr

3D objekty určené pro většinu aplikací mají typicky velký počet nadbytečných trojúhelníků, které nejsou pro celkové vnímání objektu důležité a zbytečně zatěžují proces vykreslování. Proto je vhodné eliminovat nadbytečné trojúhelníky, čímž se vykreslování většinou znatelně zrychlí. Tyto nadbytečné informace vznikly buď při vytváření objektu v grafickém editoru, nebo při pořizování z reálného světa (3D scanner). Například můžou být způsobeny postupem, kdy grafik vytváří objekt pomocí NURBS ploch a pak jej s nevhodným nastavením převede na trojúhelníkovou sít.

Velká výhoda v zjednodušení modelů je i ve škálovatelnosti. Je možné vytvořit stejný objekt s různým počtem trojúhelníků, a libovolná aplikace zobrazující daný objekt se může rozhodnout zda jej má zobrazit v plném detailu, či vybrat některou z nižších úrovní detailu. Takto se může rozhodnout podle výkonu počítače (na počítači se slabým CPU a integrovanou grafickou kartou v nízkém detailu a na jiném počítači s rychlým CPU a moderní grafickou kartou ve velmi vysokém detailu), požadované překreslovací frekvenci, či podle vzdálenosti modelu od pozorovatele. Čím dále objekt je, tím může mít nižší detail.

Některé systémy (např. mobilní telefony s podporou Java aplikací) v současné době nemají výkon potřebný na zobrazení velkých detailních objektů. Řešením je buď vytvoření vhodných modelů ručně, nebo automatické převedení existujících modelů pomocí programu, jenž je hlavním tématem této práce.

7.1 Možné využití aplikace a další vývoj projektu

Zobrazení objektů je možné například využít v průmyslu, při přezentaci strojů nebo výrobků zákazníkům. Zobrazení objektů může být také využito i v počítačových hrách. Další možností je využití v reklamě, například při prodeji nemovitostí nebo jejich interiéru. Zájemce si může stáhnout data a po objektu se projít, aniž by musel odejít od počítače. Toto může udělat například přes webový prohlížeč a wrml či x3d plugin. Zde je zjednodušení modelu vhodné i z důvodu přenosu dat přes internet.

Program je primárně určen pro ovládání z příkazové řádky tak, aby bylo snadné celý proces zautomatizovat. Implementoval jsem načítání a ukládání z formátu 3ds, protože se jedná o standard podporovaný mnoha aplikacemi. Tím se zvýší možnost dalšího využití programu.

7.2 Srovnání s konkurenčními programy

Před psaním této práce jsem vyzkoušel několik komerčních programů a byl jsem poměrně zklamaný jejich možnostmi. Některé načetly objekt, ale byly schopny uložit zpět pouze geometrii objektu bez materiálů. Jiné byly schopny načíst pouze vrml formát, či některý jiný, v praxi málo používaný. Nejlepší možnosti měl program Rational Reducer, který byl schopen načíst a uložit formát 3ds, při redukování ale docházelo k artefaktům jako překlopení normály či nezachování okraje objektu. Těmto nepříjemnostem jsem se v aplikaci vyhnul. Navíc cena těchto programů je poměrně vysoká, pohybuje se v rozmezí 50$ za Action 3D reducer, který podporuje formát 3ds a používá metodu progresivní mesh, přes VizUp, který podporuje pouze VRML a OBJ soubory ale neudává použitou metodu za 100$ až po Rational Reducer, který podporuje 3ds a VRML formáty za 6600$. Jiné alternativy jsem nenašel.

7.3 Přínos aplikace

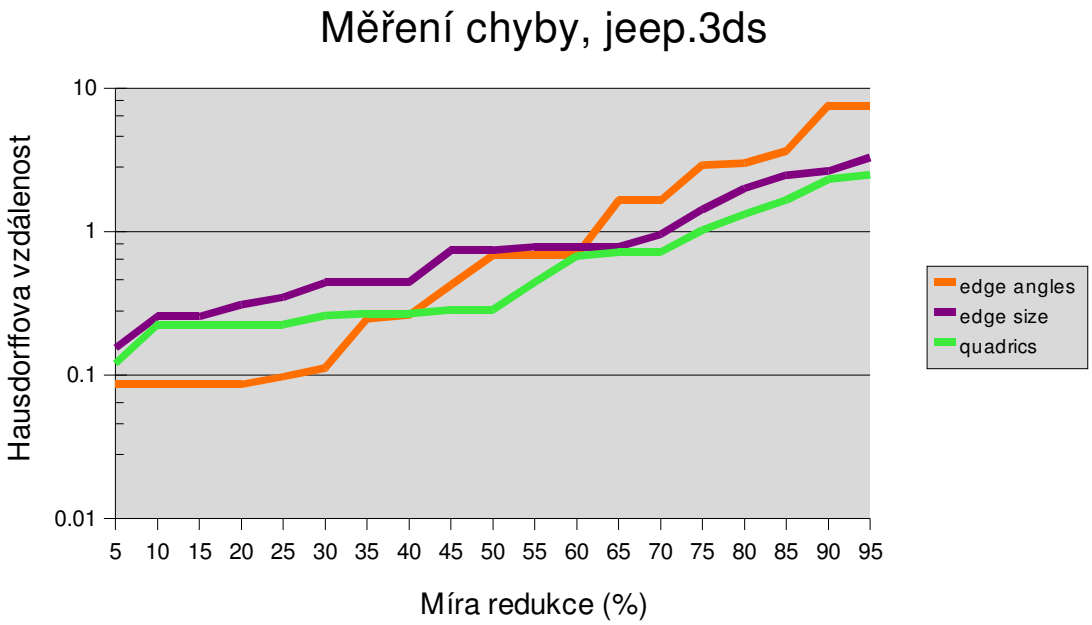
Myslím si, že program který jsem vytvořil je pro demonstraci zjednodušení modelů ve 3D grafice vyhovující a díky možnému načítání a ukládání do formádu 3DS jej lze použít i ve spolupráci se spoustou dalších aplikací. Snažil jsem se o co nejvyšší možnost budoucí rozšiřitelnosti, efektivní provádění operací a co nejjednoduší ovládání. Podařilo se mi implementovat edge collapse se třemi různými metrikami.

Díky rozvržení aplikace je možné snadno napsat části pro import a export do dalších grafických formátů. Aplikace by do budoucna mohla být rozšířena o GUI pro jednoduší ovládání, protože i přes výhody ovládání programu přes příkazovový řádek dávají někteří lidé přednost grafickému ovládání. Nakonec by bylo možné uvažovat o implementaci projektu jako samostatné sdílené knihovny pro použití v dalších aplikacích.

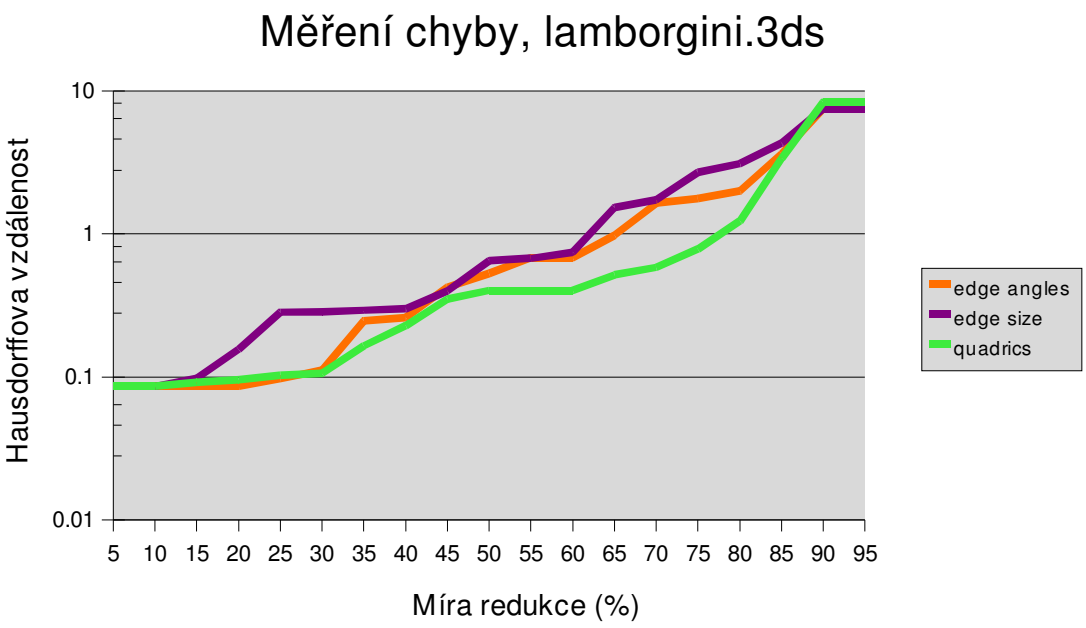
8 Přílohy

8.1 Naměřené chyby

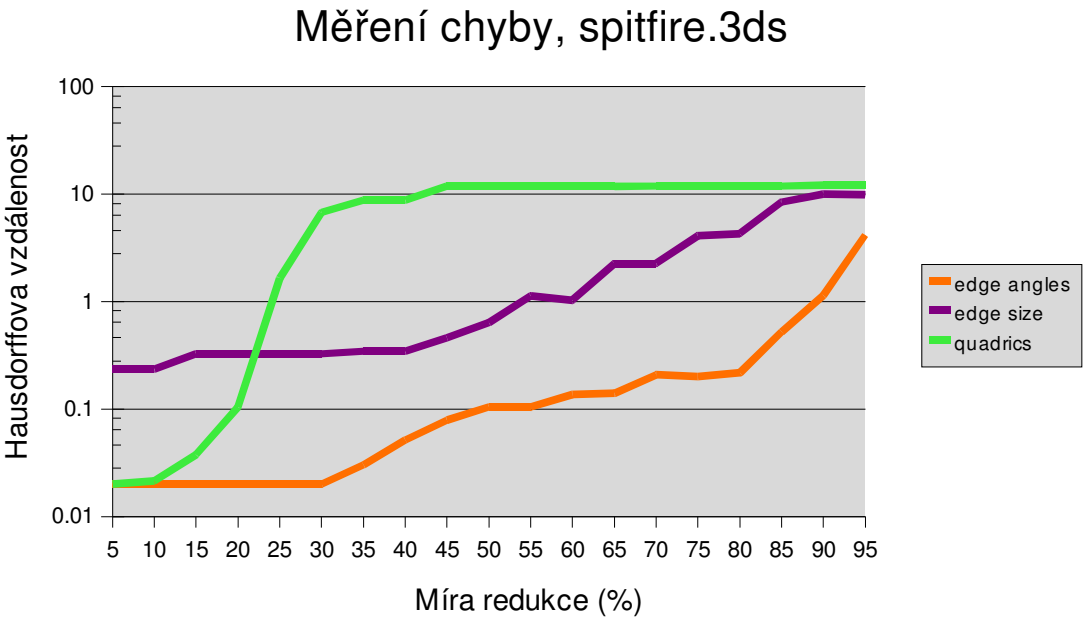
Níže jsou uvedeny výsledky měření chyby. Na svislé ose je vynesena naměřená Hausdorffova vzdálenost originálního objektu a zjednodušeného objektu. Jelikož tato vzdálenost závisí na měřítku každého modelu, nelze srovnávat naměřené chyby mezi různými modely navzájem. Grafy slouží k porovnání kvality několika různých algoritmů pouze na měřeném modelu. Na vodorovné ose je vynesena míra redukce. Například hodnota 95% udává velké zjednodušení, protože zjednodušený objekt obsahuje pouze 5% polygonů z původního počtu.



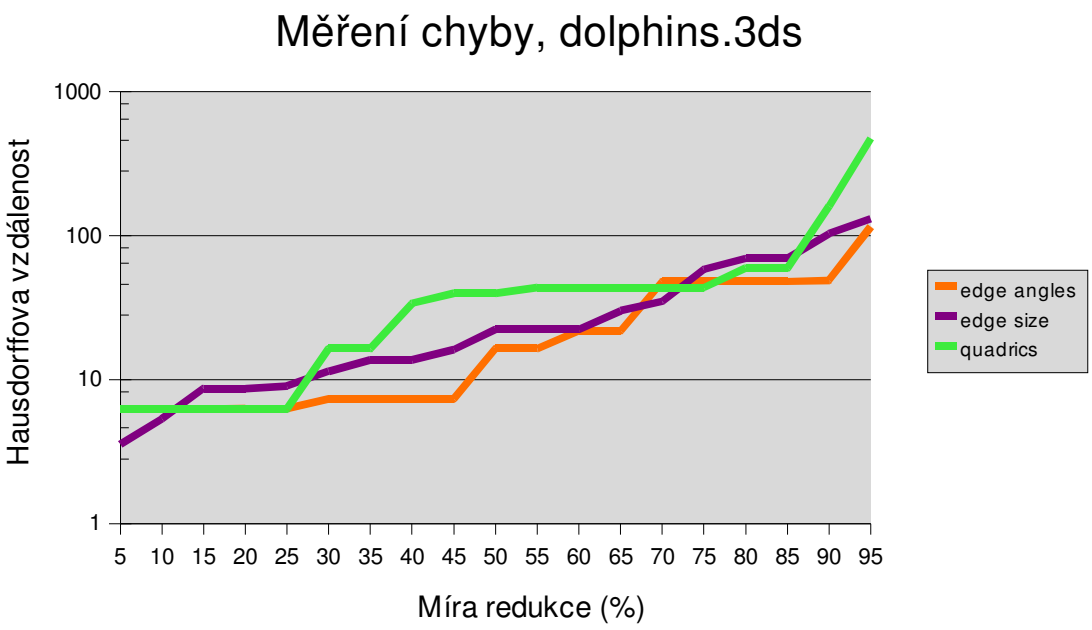
Příloha 1: Chyba u souboru jeep.3ds



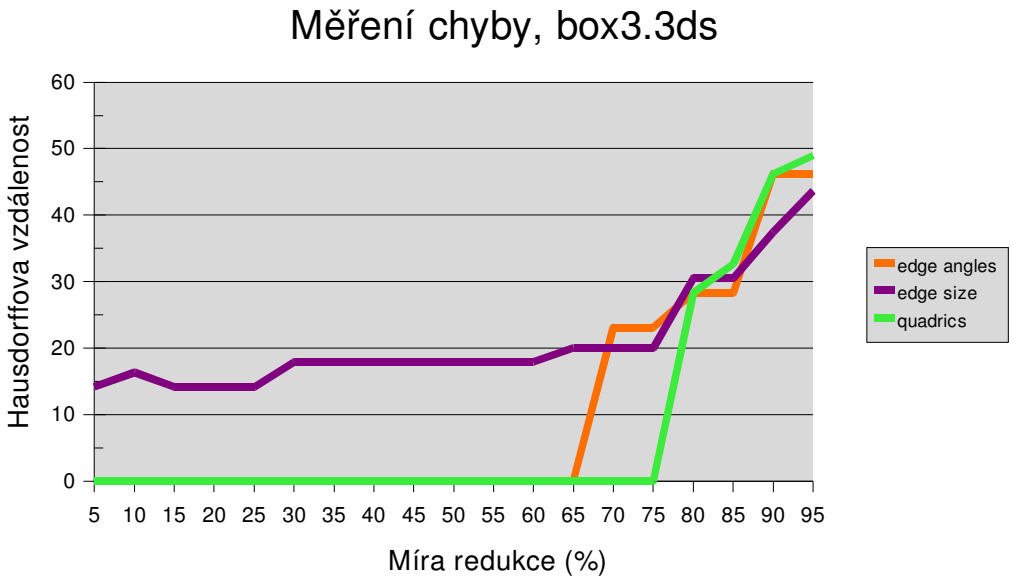
Příloha 2: Chyba u souboru lamborgini.3ds



Příloha 3: Chyba u souboru spitfire.3ds



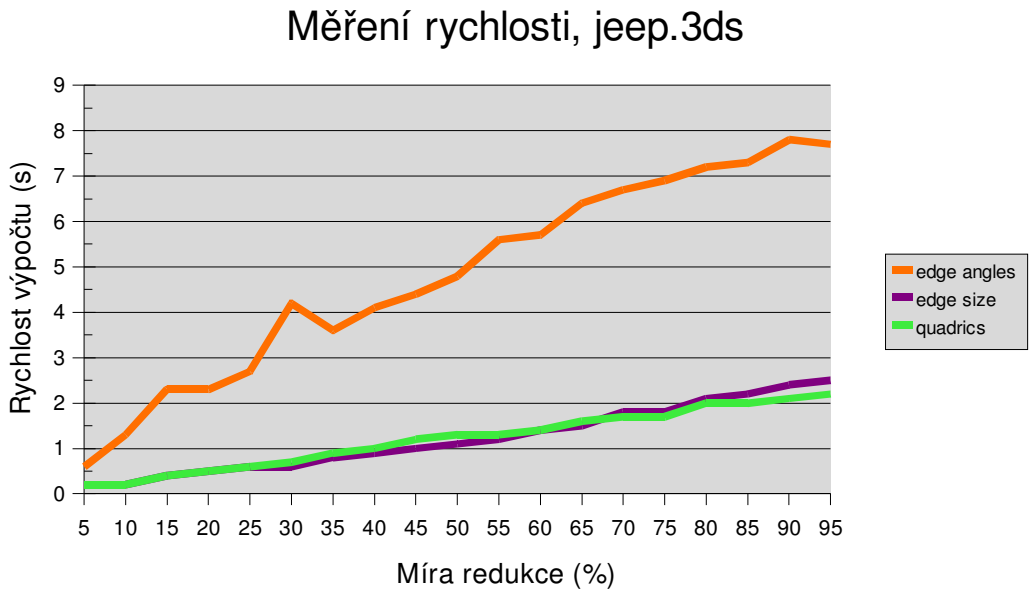
Příloha 4: Chyba u souboru dolphins.3ds



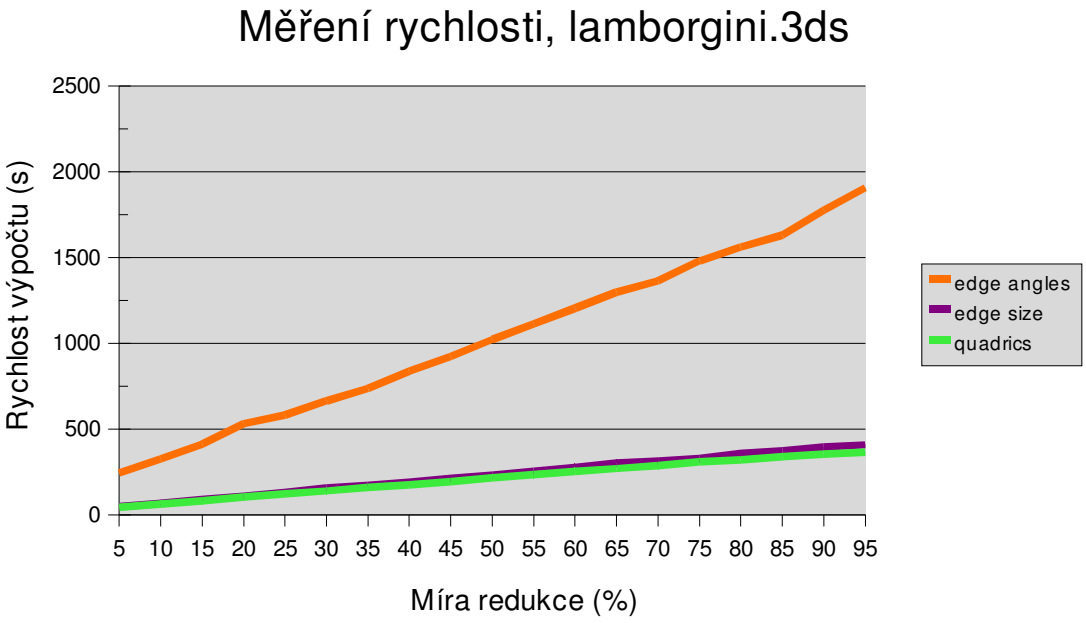
Příloha 5: Chyba u souboru box3.3ds

8.2 Naměřené rychlosti

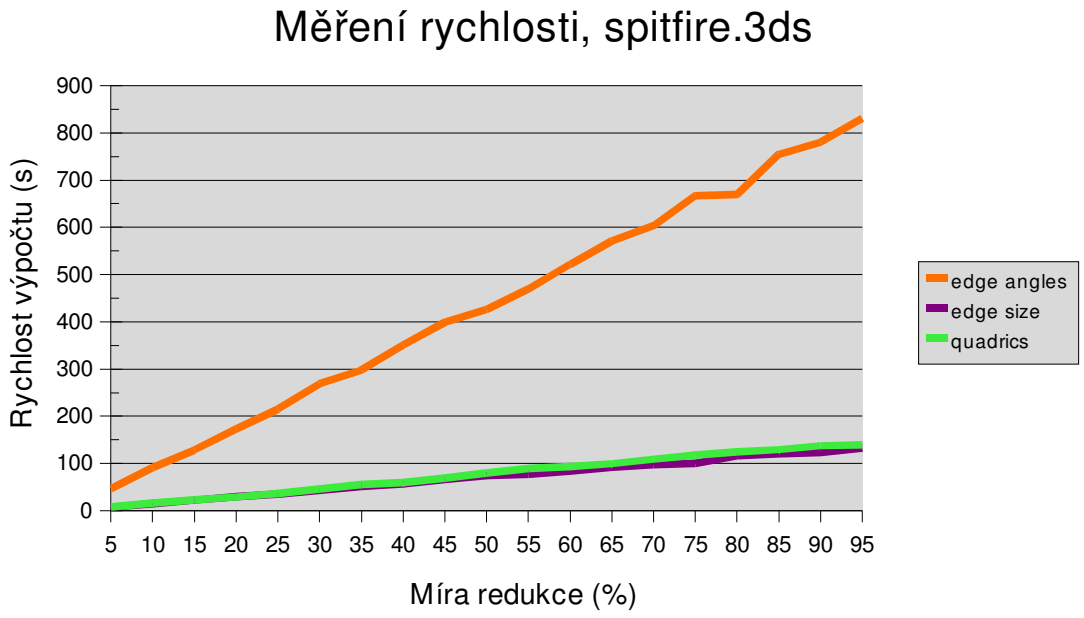
Níže jsou uvedeny výsledky měření rychlosti zjednodušování testovaných objektů. Na svislé ose je vynesen čas potřebný na zjednodušení objektu. Do této hodnoty se nepočítá načtení nebo uložení objektu do souboru, ale pouze jeho zjednodušení a případné předpočítání hodnot potřebných ke zjednodušování. Vodorovná osa udává míru redukce stejně, jako grafy v příloze 8.1.



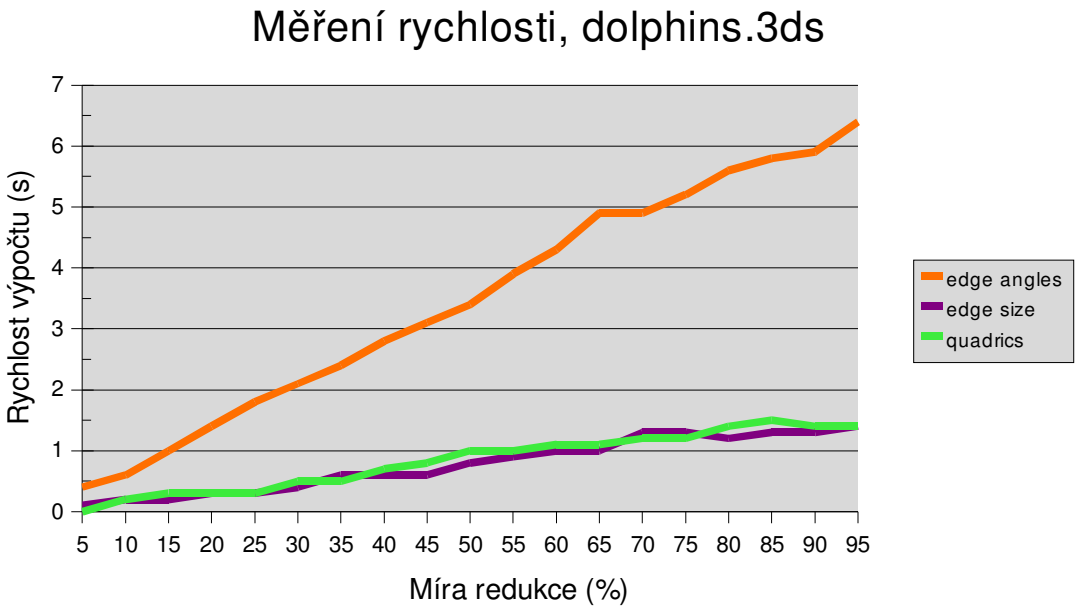
Příloha 6: Měření rychlosti jeep.3ds



Příloha 7: Měření rychlosti lamborgini.3ds



Příloha 8: Měření rychlosti spitfire.3ds



Příloha 9: Měření rychlosti dolphins.3ds

9 Reference

[1] David Luebke, Martin Reddy, Jonathan Cohen. Level of Detail for 3D Graphics. Morgan Kaufmann Publishers, 2003, 390 stran, ISBN 1-55860-838-9

[2] Stan Melax. A Simple, Fast and Effective Polygon Reduction Algorithm. Game Developer Magazine, November 1998, strany 44-49

[3] J. Žára, B. Beneš, P. Felkel. Moderní počítačová grafika. Computer Press, Brno, 2004, ISBN 80-251-0454-0.

[4] P. Lindstrom, G. Turk. Fast and memory efficient polygonal simplification. IEEE Visualization 98 Conference Proceedings, 1998.

[5] M. Garland, P. Heckbert. Surface simplification using quadric error metrics. SIGGRAPH 97 Conference Proceedings, 1997

[6] William Schroeder, Jonathan Zarge, William Lorensen. Decimation of triangle meshes. SIGGRAPH 92 Conference Proceedings, 1992

[7] Jarek Rossignac, Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. 1993.

[8] Metro. Program pro výpočet odlišnosti objektů, je k dispozici ke stažení na adrese http://vcg.sourceforge.net/tiki-index.php?page=Metro

[9] Hugues Hoppe, Tony DeRose. Mesh Optimization. Proceedings of SIGGRAPH 93. 1993

[10] Bernd Junger, Jack Snoeink. Selecting independent vertices for terrain simplification. WSCG '98, Plzeň, February 1998.