SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3609

SMANJENJE BROJA POLIGONA OBJEKATA

Tomislav Maljić

Zagreb, lipanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 12. ožujka 2014.

ZAVRŠNI ZADATAK br. 3609

Pristupnik: **Tomislav Maljić (0036467675)**
Studij: Računarstvo
Modul: Računalno inženjerstvo

Zadatak: **Smanjenje broja poligona objekata**

Opis zadatka:

Proučiti metode koje su koriste u redukciji broja poligona za objekte koji su zadani mrežom trokuta. Proučiti utjecaj ovakve simplifikacije na kvalitetu objekta. Razraditi nekoliko algoritama za smanjenje broja poligona te ih usporediti. Ostvariti programsku implementaciju koja omogućuje prikaz i usporedbu različitih metoda redukcije broja poligona. Na različitim primjerima prikazati ostvarene rezultate. Načiniti konačnu ocjenu ostvarenih rezultata.
Izraditi odgovarajući programski proizvod. Koristiti programski jezik C i grafičko programsko sučelje OpenGL. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:
Prof.dr.sc. Željka Mihajlović

Djelovođa:
Prof.dr.sc. Danko Basch

Predsjednik odbora za završni rad modula:
Prof.dr.sc. Mario Žagar

SADRŽAJ

1. UVOD

Mreža trokuta je uobičajeni način prikaza trodimenzionalnih objekata u različitim granama računalne grafike i obrade geometrije. S evolucijom moderne tehnologije kompleksnost takvih objekata postaje sve veća stoga i vizualna aproksimacija u odnosu na objekt iz stvarnog svijeta postaje vjernija. Na primjer, CAD (dizajn potpomognut računalom) alati danas proizvode veoma kompleksne modele s milijunima poligona no tu se pojavljuje problem jer takvo povećanje kompleksnosti još uvijek nadmašuje poboljšanja u performansama sklopovske opreme. Kao rezultat svega toga kompleksni objekti moraju se podvrgnuti simplifikaciji tj. smanjenju broja poligona uporabom različitih algoritama. Do danas su razvijene efikasne metode za simplifikaciju mreže koje pružaju najbolji omjer između razine detalja i zahtjeva aplikacije i koje su sposobne napraviti vjernu aproksimaciju u odnosu na originalnu mrežu uz manje elementa same mreže.

Kroz ovaj rad, prvo će biti prikazane definicije vezane uz elemente od kojih je sastavljena mreža trokuta zatim će se dati pregled i objašnjenja nekih algoritama za simplifikaciju. Na kraju će se analizirati i usporediti različiti algoritmi za smanjivanje broja poligona objekata na određenim primjerima.

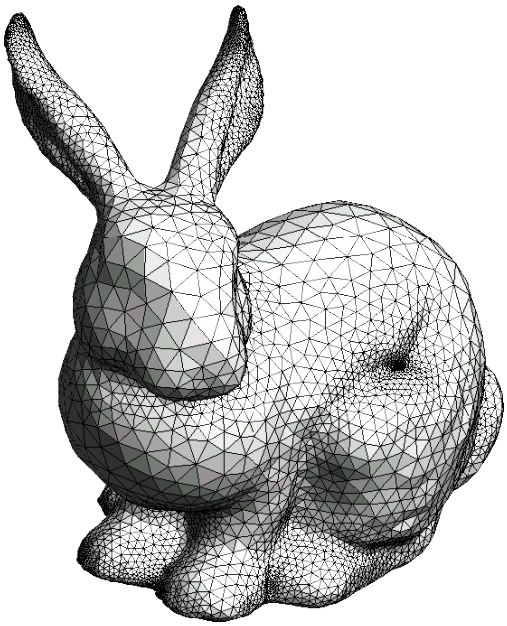
2. Mreže poligona

Prije ulaženja u detalje algoritama za simplifikaciju mreže objekta potrebno je definirati generalna svojstva mreže.

2.1. Definicija mreže trokuta (poligona)

Mreža poligona se u računalnoj grafici definira kao linearna površinska aproksimacija objekta iz stvarnog svijeta. Sastoji se od vrhova (točka), bridova koji spajaju te vrhove i poligona. Mreža trokuta je podskup mreže poligona u kojoj su svi poligoni trokuti (Slika 1.).

Matematički gledano, mreža trokuta M sastoji se od para (V, K), gdje je $V=\{v_1, ..., v_m\}$, $v_i \in R^3$ skup točaka u R^3, a K je varijabla koja predstavlja povezanost mreže. Ako se neki element iz K sastoji samo od $\{i\} \in K$ tada je riječ o točki, kada se element iz K sastoji od $\{i, j\} \in K$ tada je riječ o bridu i kada se sastoji od $\{i, j, k\} \in K$ tada je riječ o poligonu (trokutu) [1].



Slika 1. *Primjer objekta koji se sastoji od mreže trokuta*

2.2 . Formati i struktura podataka mreže trokuta

2.2.1 Formati datoteka

Sve mreže imaju dva zajednička svojstva, geometriju koja definira poziciju vektora i točke u trodimenzionalnom prostoru i topologiju koja definira povezanost točaka. Kad se mreža sprema u datoteku, zapravo se spremaju njezina geometrija i topologija. Također neke mreže mogu imati spremljene i dodatne informacije kao normalne točaka, normale ravnina, boje točaka itd.

Postoji ogroman broj formata za spremanje mreža. Razlog tomu je što većina programa odnosno biblioteka generiraju svoj format spremanja pa je teško opisati ih jedinstveno. Najpoznatiji formati su Wavefront Obj format (obj.), OFF (Object File Format), VRML (Virtual Reality Modeling Language) i PLY (Polygon File Format). Primjer obj. formata:

```
# Lista točaka sa(x, y, z) koordinatama
  v 0.123 0.234 0.345
  v ...
  ...

# Koordinate teksture sa (u ,v [,w]) koordinatama
  vt 0.500 1 [0]
  vt ...
  ...

# Normale u (x,y,z) obliku
  vn 0.707 0.000 0.707
  vn ...
  ...

# Definicije poligona, mogu se zadati kao same točke v1 v2
v3, kao točka/koordinata-teksture v1/vt1 v2/vt2 …, kao
točka/normala i kao točka/koordinata-teksture/normala
v1/vt1/vn1 …
  f 1 2 3
  f 3/1 4/2 5/3
  f 6/4/1 3/5/3 7/6/5
  f ...
```

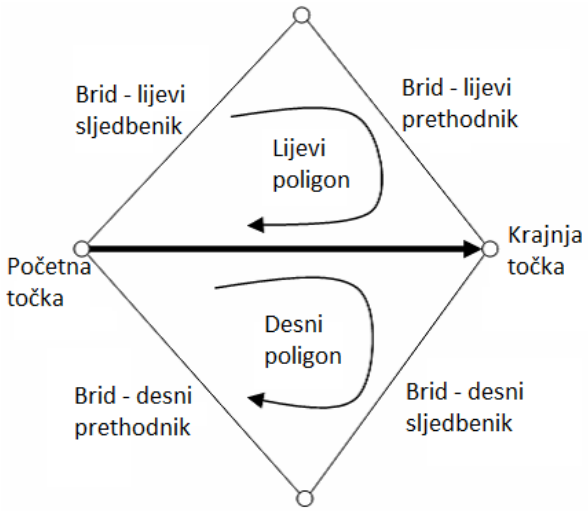
2.2.2. Struktura podataka

Odgovarajuća struktura podataka za mreže se odabire uzimajući u obzir zahtjeve koji nam postavljaju algoritmi ugrađeni u različita programska rješenja.

Najjednostavnije rješenje je oblik „triangle soup“ struktura podataka koja se sastoji od koordinata točaka i točaka koje čine pojedini poligon. Ovakva struktura nije memorijski efikasna jer se točke koju dijele više ravnina repliciraju.

Poboljšani pristup je struktura u kojoj se u obzir uzimaju točke koju dijele više poligona. Za takve točke se koriste reference tj. pokazivači na te točke odnosno indeks te točke u polju. Jedna i druga struktura dijele isti problem, a to je da povezanost mreže nije eksplicitno zadana te se za nju mora utrošiti dodatno vrijeme za izračun.

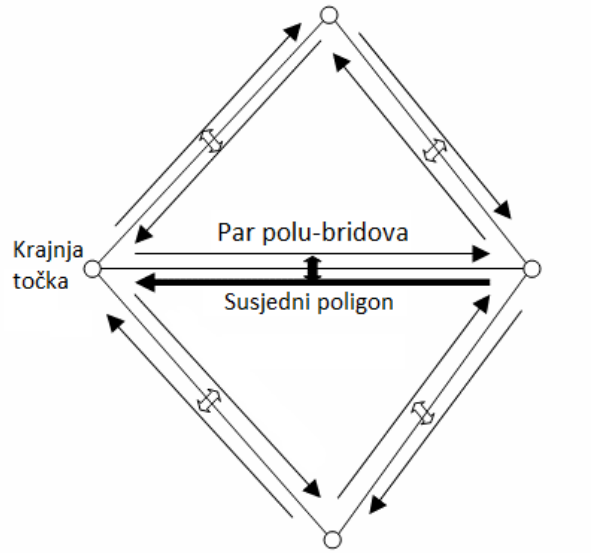
Taj problem rješava „winged-edge“ struktura u kojoj je brid najvažniji element. Bridovi su povezani sa osam referenci: dvije točke (početak brida i kraj brida), dva poligona koji taj brid omeđuju (lijeva i desna) te četiri brida (lijevi prethodnik i sljedbenik i desni prethodnik i sljedbenik) koji se koriste za obilazak lijevog i desnog poligona (Slika 2.). Za ravnine i točke dovoljno je spremiti referencu na jedan od njihovih susjednih bridova.



Slika 2. *„Winged-edge“ struktura*

No i za ovakvu strukturu postoji problem, smjer bridova nije uvijek konzistentan tj. brid se obilazi u drugačijim smjerovima kad se obilaze lijeva odnosno desna ravnina.

„Half-edge“ struktura je dizajnirana da riješi problem s orijentacijom brida. Kod nje svaki brid se dijeli na dva polu-brida koji su suprotnog smjera s namjerom da su svi polu-bridovi orijentirani u smjeru kazaljke na satu oko svakog poligona (Slika 3.). Za svaki polu-brid potrebna je referenca na par polu-bridova, sljedeći polu-brid i krajnju točku.

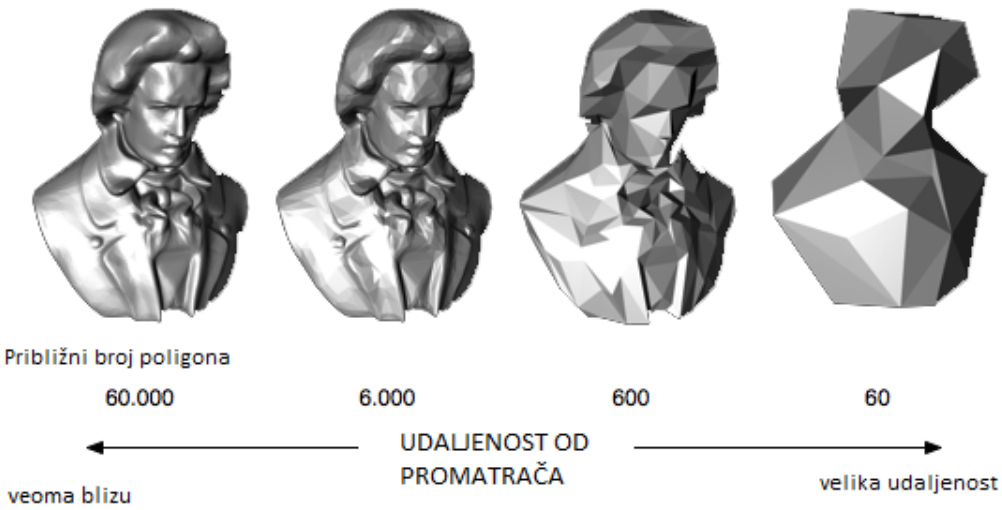


Slika 3. *Half-edge struktura*

3. Algoritmi za simplifikaciju mreže poligona

Kao što je već bilo navedeno moderne tehnologije u stanju su proizvesti kompleksne mreže s milijunima poligona. Takva kompleksnost ponekad uzrokuje poteškoće pri iscrtavanju i obradi mreže. Algoritmi za simplifikaciju pokušavaju eliminirati redundantne elemente mreže kako bi se objekt mogao što brže iscrtati i obraditi, ali bez značajnijih promjena samog oblika objekta. Nadalje uklanjanjem redundancije smanjuje se i veličina objekta koji zauzima prostor u memoriji ili disku.

Za interaktivne aplikacije kao što su video igre, CAD itd. performanse u realnom vremenu su bitne. Objekti koji se koriste u takvim aplikacijama mogu se simplificirati koristeći tehniku *razina detalja (eng. Level of details)* čija je osnovna ideja da udaljeni objekti ne zahtijevaju rezoluciju kao bliži. Kada je objekt bliže promatraču on se iscrta s kompleksnijom verzijom mreže odnosno objekt koji je udaljen od promatrača se iscrta s jednostavnijom verzijom te iste mreže (Slika 4.).



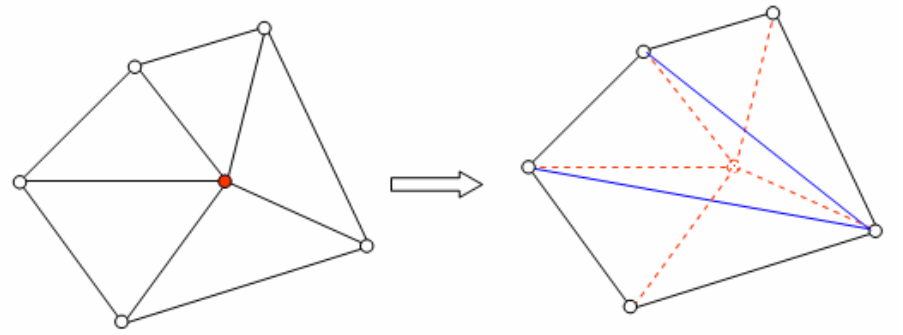
Slika 4. *Primjer tehnike „razina detalja“*

3.1. Iterativni algoritmi za simplifikaciju

Iterativni algoritmi za simplifikaciju uklanjaju jedan element mreže u jednoj iteraciji. Prvo se računa cijena uklanjanja pojedinog elementa i u konačnici element mreže s najmanjom pogreškom se briše. Zbog brisanja elementa nastaje rupa u tom području te se provodi re-triangulacija u kojoj opet dobivamo mrežu trokuta. Postupak se ponovno provodi samo za susjedne elemente mreže na koje je utjecalo uklanjanje prethodnog elementa.

3.1.1. Uklanjanje točke (vertex decimation)

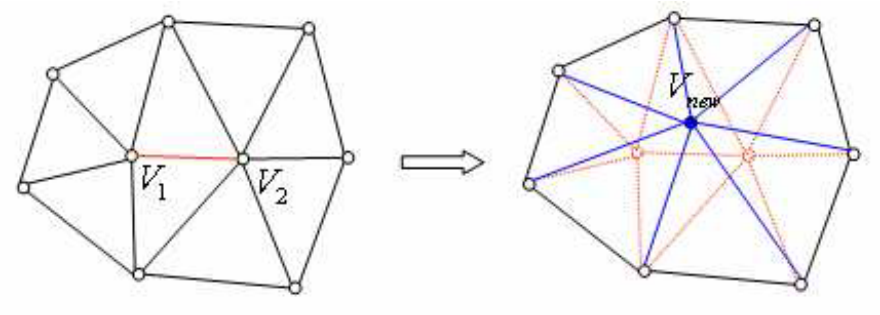
Uklanjanje točke je metoda kojom se uklanja po jedna točka valencije *n* (broj susjednih točaka) zbog koje ostaje n-strana rupa. Potom se provodi triangulacija nakon koje nastaje *n-2* trokuta. Nakon jedne iteracije uklanjanjem jedne točke dolazi i do uklanjanja dvaju poligona i triju bridova (Slika 5.) [1].



Slika 5. *Primjer algoritma uklanjanje točke*

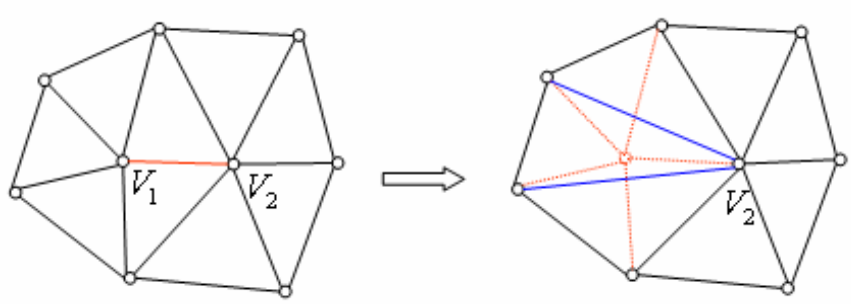
3.1.2. Uklanjanje bridova (edge collapse)

Potpuno uklanjanje brida je metoda kojom se uzimaju dvije susjedne točke i uklanja se brid između njih. Točka koja nastaje uklanjanjem brida pomiče se na novu poziciju (Slika 6.) [1].



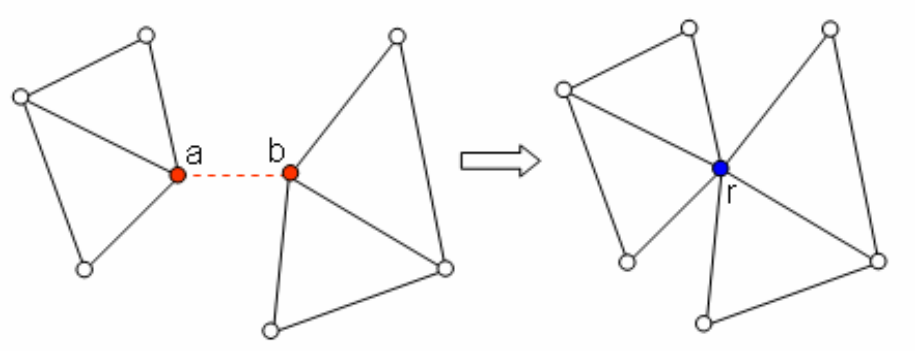
Slika 6. *Primjer uklanjanja brida*

Postoji i druga varijanta algoritma, „half-edge collapse“ u kojem se ne može birati pozicija nove točke već je ona predodređena jednom od krajnjih točaka između kojih se brisao brid (Slika 7.).



Slika 7. *Primjer algoritma „half-edge“ collapse*

Još jedna varijanta algoritma je i spajanje parova točki u kojem se dvije točke spajaju u jednu iako nisu povezane bridom. Ovakva operacija smanjuje broj točaka no broj bridova i poligona ostaje isti (Slika 8.).



Slika 8. *Primjer spajanja dviju točaka*

3.2. Računanje pogreške

Metode za mjerenje pogreške koriste se za odabir elementa mreže na koji će se primijeniti simplifikacija. Mjerenja određuju promjene na lokalnoj razini (susjedne točke i bridovi) ili globalnoj razini (cijela mreža) koje nastaju kao posljedica uklanjanja elementa.

3.2.1. Računanje pogreške na lokalnoj razini

Jedna od metoda za računanje pogreške na lokalnoj razini je računanje kvadrične pogreške koja se danas učestalo koristi zbog jednostavnosti izračuna.

Ako imamo ravninu t_k, njena jednadžba bi izgledala:

$$a_k x + b_k y + c_k z + d_k = 0 \qquad (1)$$

gdje su $n_k = (a_k, b_k, c_k)$ mjere normale ravnine t_k, a d_k je konstanta.

(1) se može zapisati i u obliku: $n^T v + d = 0$ (2)

Pomoću (1) ili (2) može se izračunati funkcija kvadratne udaljenosti $D^2(v)$ koja daje kvadratnu udaljenost između pripadajuće ravnine i točke $v = (x, y, z)$.

$$D^2(v) = (n^T v + d)^2 \quad (3)$$

Iz nje se potom može izračunati greška točke v kao:

$$E = \sum_i D_i^2(v) = \sum_i (n^T v + d)^2 \quad (4)$$

Izraz (3) može se poprimiti i oblik:

$$\begin{aligned} D^2(v) &= (n^T v + d)^2 \\ &= (v^T n + d)(n^T v + d) \\ &= v^T n n^T v + 2 d n^T v + d^2 \\ &= v^T (n n^T) v + 2 (dn)^T v + d^2 \end{aligned}$$

Vrijedi i izraz:

$$E = \sum_i D_i^2(v) = \sum_i Q_i(v)$$

Kako bi se olakšao izračun pogreške, kvadrična pogreška Q može se promatrati kao matrica:

$$Q = \begin{bmatrix} nn^T & dn \\ dn^T & d^2 \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Ova metoda za mjerenje pogreške može se koristiti za sve varijacije metode uklanjanja brida, a parovi točaka (V_1, V_2) birat će se na način da nakon uklanjanja brida između njih nastane najmanja pogreška. Kvadrična pogreška Q_{nova} za simplificiranu točku (V_{nova}) računa se jednostavno zbrajajući Q_1 i Q_2:

$$Q_{nova} = Q_1 + Q_2$$

Konačno, izraz za izračun greške za novu točku je:

$$E_{nova} = \tilde{v}^T{}_{nova} Q_{nova} \tilde{v}_{nova}$$

S ovakvim zapisom za računanje kvadrične pogreške, izbjegnut je izračun kvadratnih udaljenosti te je ono zamijenjeno sa jednostavnim zbrajanjem dviju matrica 4x4 što uvelike doprinosi brzini algoritma [3].

3.2.2. Računanje pogreške na globalnoj razini

Hausdorffova udaljenost je jedna od metoda za računanje pogreške na globalnoj razini. Koristi se za kontrolu aproksimacije pogreške između originalnog i simplificiranog modela. Pri svakom koraku algoritma briše se točka iz mreže koja sa svojim brisanjem najmanje doprinosi ukupnoj Hausdorffovoj udaljenosti bez prekoračenja predefinirane Hausdorffove udaljenosti.

Hausdorffova udaljenost između dva skupa točaka iz A i iz B može se definirati kao:

$$H(A,B) = \max\ (h(A,B), h(B,A))$$

gdje je $h(A,B)$ jednostrana Hausdorffova udaljenost:

$$h(A,B) = \max \min \big||a-b|\big|, \qquad a \in A, b \in B$$

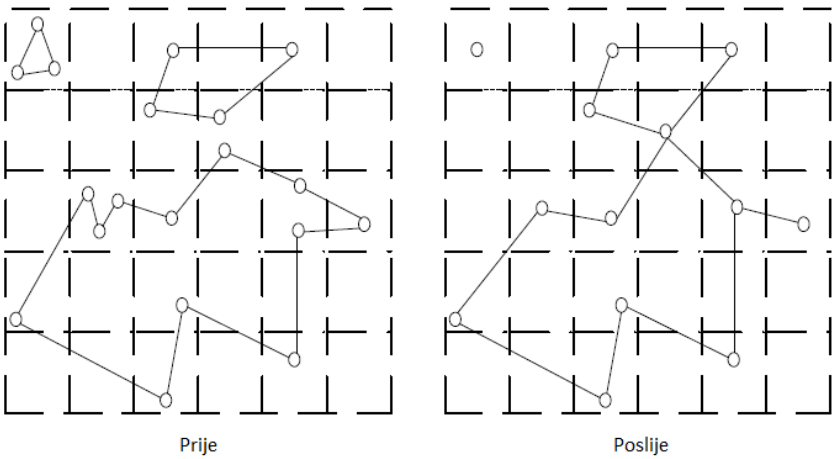
Ovaj izraz kaže da se pri izračunu jednostrane Hausdorffove udaljenosti $h(A,B)$, za svaku točku a iz A pronalazi najbliža točka b iz B i zatim se uzima maksimum od razlika tih dviju točaka kao jednostrana Hausdorffova udaljenost. Također jednostrana Hausdorffova udaljenost nije simetrična, stoga $h(A,B)$ ne mora biti jednaka $h(B,A)$ [2].

3.3. Jedno-prolazni algoritmi

Jedno-prolazni algoritmi obrađuju mrežu u cjelini bez iterativnih postupaka. Oni su po pitanju obrađivanja puno brži od iterativnih, ali iterativni algoritmi daju bolje rezultate.

Među jedno-prolaznim algoritmima izdvajaju se algoritmi grupiranja točaka (eng. *vertex clustering*). Kod njih se točke mreže prostorno grupiraju u grupe (eng. *cluster*)

s obzirom na geometrijsku blizinu. Potom se odabire nova točka koja će predstavljate sve ostale točke u toj grupi (Slika 9.). Ovi algoritmi obično rade veoma brzo, ali pate od velike slabosti zbog grupiranja. Topologija originalne mreže podložna je promjenama i to posebice ako se izabere manji broj grupa kako bi se ubrzala simplifikacija. U tom slučaju dolazi do većih gubitaka detalja objekta [4].



Slika 9. *Primjer grupiranja točaka*

4. Algoritmi za simplifikaciju ugrađeni u biblioteke

Implementacija platforme koja će na vremenski i memorijski efikasan i jednostavan način dozvoliti manipulaciju podataka mreže nije lagan zadatak. U tu svrhu su se razvile i biblioteke koje su otvorenog koda. Najpopularnije su: Visualization Toolkit (VTK), Computational Geometry Algorithms Library (CGAL) i OpenMesh. Navedene biblioteke ne implementiraju samo strukturu podataka mreže nego i nude niz operacija i algoritama kao što su algoritmi za redukciju broja poligona.

4.1. Simplifikacija uz CGAL biblioteku

Algoritmi iz ove biblioteke mogu se primijeniti jedino na mreži trokuta koristeći metodu uklanjanja brida. Istaknuti algoritam je Lindstrom-Turk strategija [6].

CGAL	
Algoritam	Lindstrom – Turk strategija
Tip mreže	Mreža trokuta
Očuvanje topologije	Da
Metoda simplifikacije	Uklanjanje brida

Tablica 1. Algoritam za simplifikaciju CGAL biblioteke

4.2. Simplifikacija uz OpenMesh biblioteku

Biblioteka OpenMesh Quadric koristi iterativni algoritam za simplifikaciju. Nadalje algoritam koristi metodu „half-edge collapse“, a točka koja će se obrisati određuje na temelju mjera. Kvadrična mjera izračunava prioritet uklanjanja bridova na temelju računanja kvadrične pogreške [7].

OpenMesh	
Algoritam	Quadric Module
Tip mreže	Mreža poligona
Očuvanje topologije	da
Metoda simplifikacije	Half-edge collpase

Tablica 2. Algoritam za simplifikaciju OpenMesh biblioteke

4.3. Simplifikacija uz VTK biblioteku

vtkDecimatePro je iterativni algoritam koji koristi metodu uklanjanja bridova (eng. edge collapse). Kao ulaz prihvaća samo mrežu trokuta no može se koristiti i na drugim poligonskim mrežama ako se prije provede pretvaranje poligona u trokute. Također nije zagarantirano da će se očuvati topologija kao kod izvornog objekta. Prvi je korak klasifikacija ulaznih točaka mreže, svakoj točki se dodjeljuje jedna od pet mogućnosti: jednostavna, složena, granična, unutrašnja. Nakon što se točke klasificiraju ubacuju se u red po prioritetu koji se određuje na temelju prosječne udaljenosti točke od ravnine (jednostavne i rubne točke). Potom se svaka točka u prioritetnom redu briše iterativno sve dok red ne postane prazan.

vtkQuadricDecimation je također iterativni algoritam koji koristi računanje kvadrične pogreške i metodu uklanjana brida. S izvođenjem prestaje kada je postignut željeni nivo redukcije odnosno ograničenja u topologiji sprječavaju daljnju redukciju [5].

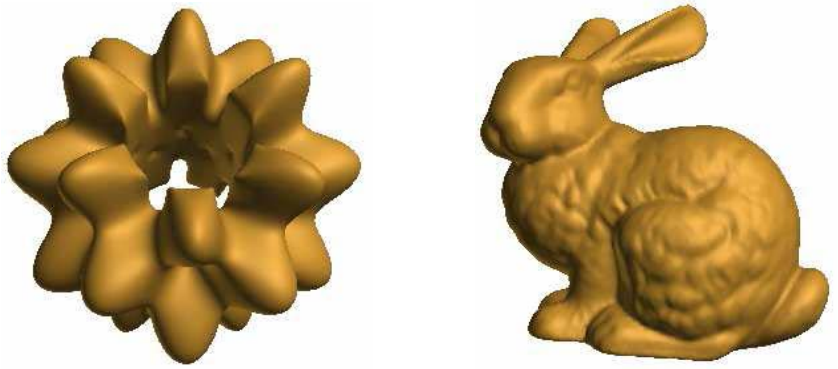
VTK		
Algoritam	vtkDecimatePro	vtkQuadricDecimation
Tip mreže	Mreža trokuta	Mreža trokuta
Očuvanje topologije	Ovisi o izboru	Ne nužno
Metoda simplifikacije	Uklanjanje brida (edge collapse)	Uklanjanje brida (edge collapse)

Tablica 3. Algoritmi za simplifikaciju VTK biblioteke

5. Usporedba algoritama za simplifikaciju mreže trokuta

Za usporedbu će se koristiti tri biblioteke koje su navedene u prethodnom poglavlju. Svi algoritmi iz tih biblioteka su iterativni algoritmi no razlikuju se u načinu odlučivanja koja će se točka izbrisati iz mreže. Također za sve algoritme se može navesti postotak željene redukcije no hoće li se uspješno doći do tog postotka ovisi o algoritmu koji se koristi. Ako se topologija mreže neće sačuvati onda će se željena redukcija postići uspješno svaki put.

Za analizu će se koristiti 3D modeli torusa i stanfordskog zeca (Slika 10.) koji su sačinjeni od mreže trokuta. Model torusa se sastoji od 16815 točaka, 33630 trokuta i 50445 bridova. Model zeca je složeniji i već dugi niz godina se koristi za testiranje algoritama koji se koriste u obradi mreže. Sastoji se od 34834 točke, 69451 trokuta i 104288 bridova.



Slika 10. 3D modeli torusa i zeca

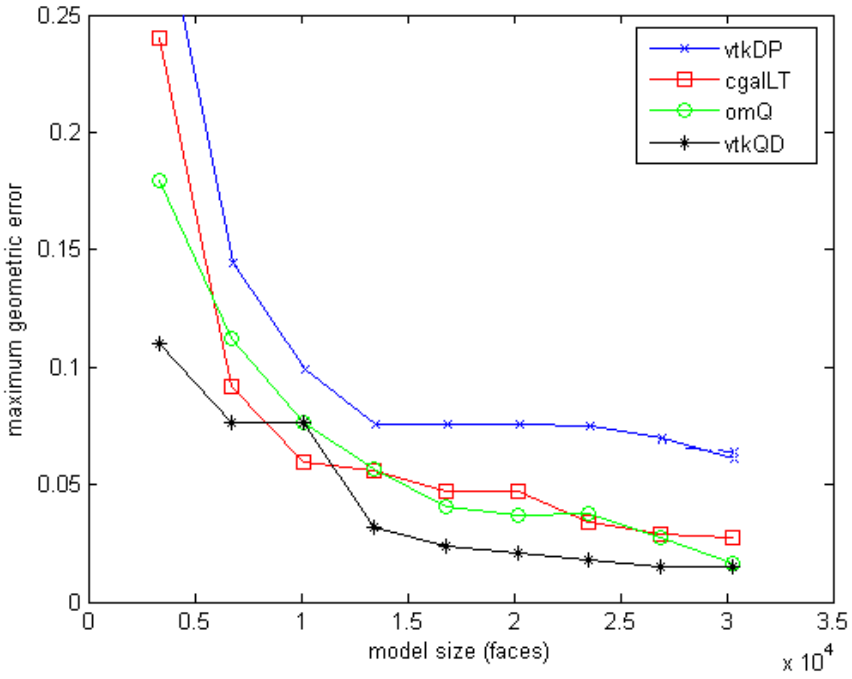
Svaki algoritam se izvršava za devet različitih vrijednosti redukcije (10%, 20%, …, 90%) i kako bi se mogla odrediti različitost između dvije mreže prikupljeni rezultati će se usporediti s alatom „Metro“. To je besplatan program otvorenog koda koji računa devijaciju između originalnog i simplificiranog modela uzimajući u obzir maksimalnu (Hausdorffova udaljenost) i srednju vrijednost pogrešaka. Metro započinje uzrokovanjem originalne mreže, zatim računa udaljenost između tih uzoraka i simplificirane mreže. Postupak se nakon toga provodi obrnuto, uzrokuje se simplificira mreža i računa se udaljenost u odnosu na originalnu mrežu. Na temelju tih izračuna Metro vraća maksimalnu i srednju vrijednost pogreške između tih dviju mreža. Također Metro je sposoban rezultate prikazati i grafički,

iscrtavanjem originalne mreže koja će imati obojanu svaku točku bojom koja odgovara pojedinoj vrijednosti pogreške.

Rezultati će se prikazati u obliku grafova, gdje će se na x-osi nalaziti broj poligona (trokuta), a na y-osi maksimalna ili srednja vrijednost pogreške.

5.1. Analiza algoritama na modelu torusa

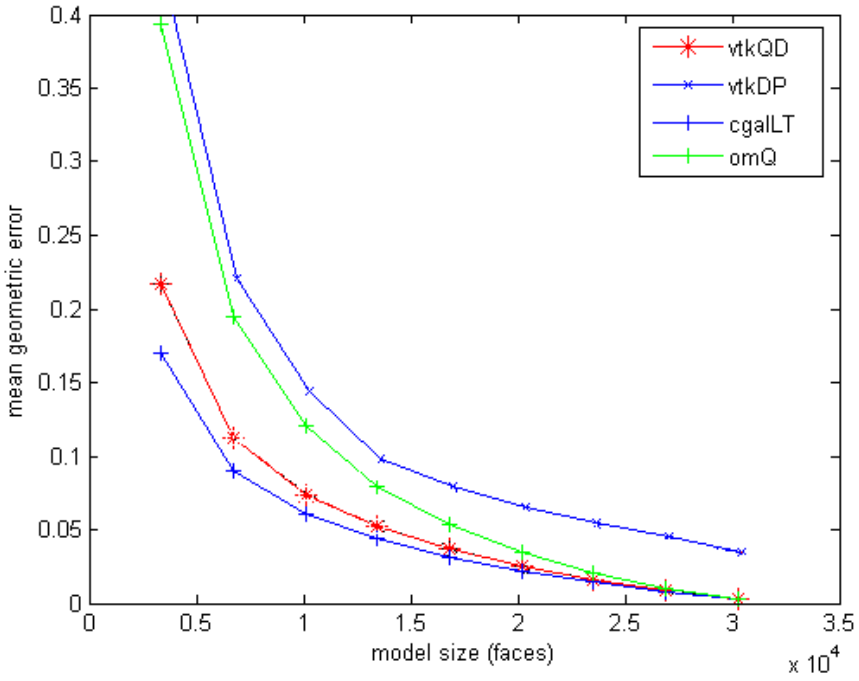
Na temelju analize grafa (Slika 11.) može se zaključiti da za maksimalnu vrijednost pogreške najbolji rezultat u većini slučajeva daje algoritam vtkQuadricDecimation. Drugo mjesto je već teže odabrati jer algoritmi Lindstrom-Turk i OpenMesh Quadric imaju slučajeve u kojima su bolji jedan od drugog. Zadnje mjesto pripada algoritmu vtkDecimationPro.



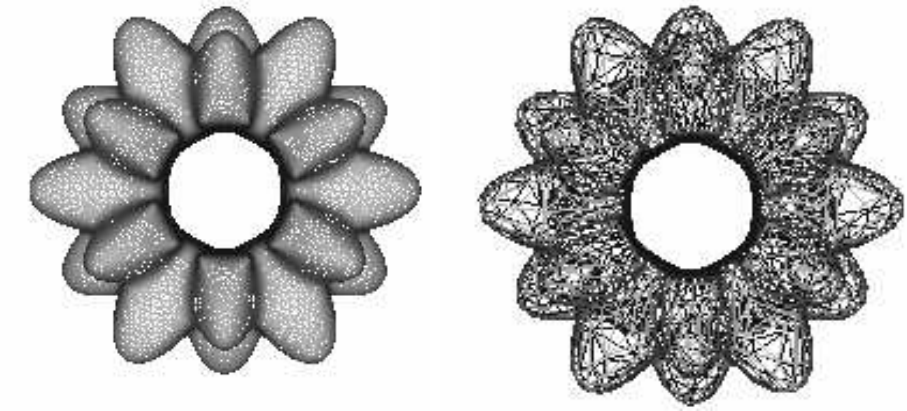
Slika 11. Graf usporedbe algoritama s obzirom na maksimalnu vrijednost pogreške

Što se tiče grafa kada se gleda srednja vrijednost pogreške, algoritam Lindstrom-Turk iz biblioteke CGAL postiže najbolje rezultate. Drugo mjesto pripada algortimu

vtkQuadricDecimation, treće algoritmu OpenMesh Quadric, a zadnje opet algoritmu vtkDecimationPro (Slika 12.).



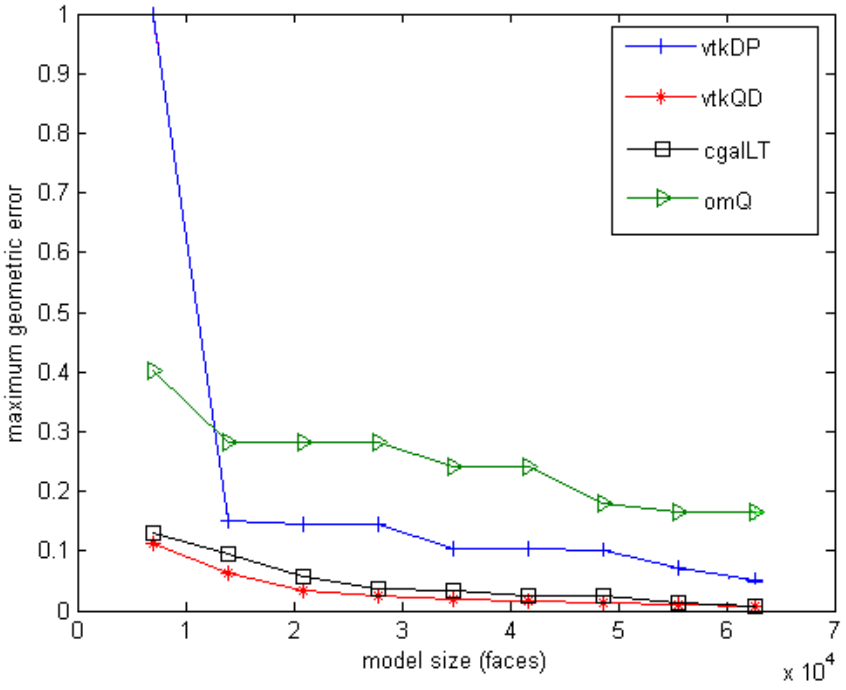
Slika 12. Graf usporedbe algoritama s obzirom na srednju vrijednost pogreške



Slika 13. Usporedba mreža modela, originalni model (lijeva strana) i simplificirani model (redukcija od 90%) (desna strana) uz algoritam vtkQuadricDecimation

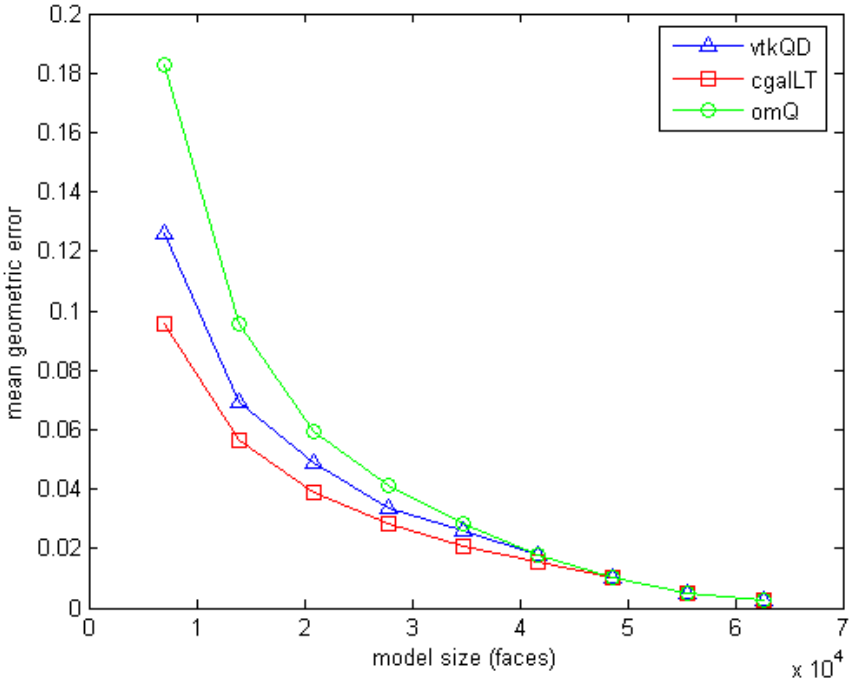
5.2. Analiza algoritama na modelu zeca

Na temelju analize grafa (Slika 14.) može se zaključiti da za maksimalnu vrijednost pogreške najbolji rezultat daje vtkQuadricDecimation algoritam. Također, vrlo blizu prethodno navedenom algoritmu je i Lindstrom-Turk algoritam. Treće mjesto zauzima algoritam vtkDecimationPro s već zamjetnijom razlikom. Algoritam Quadric Module iz biblioteke OpenMesh nije ostvario dobar rezultat iako koristi jednu od najboljih metoda za izračun pogreške i stoga dolazi na zadnje mjesto.



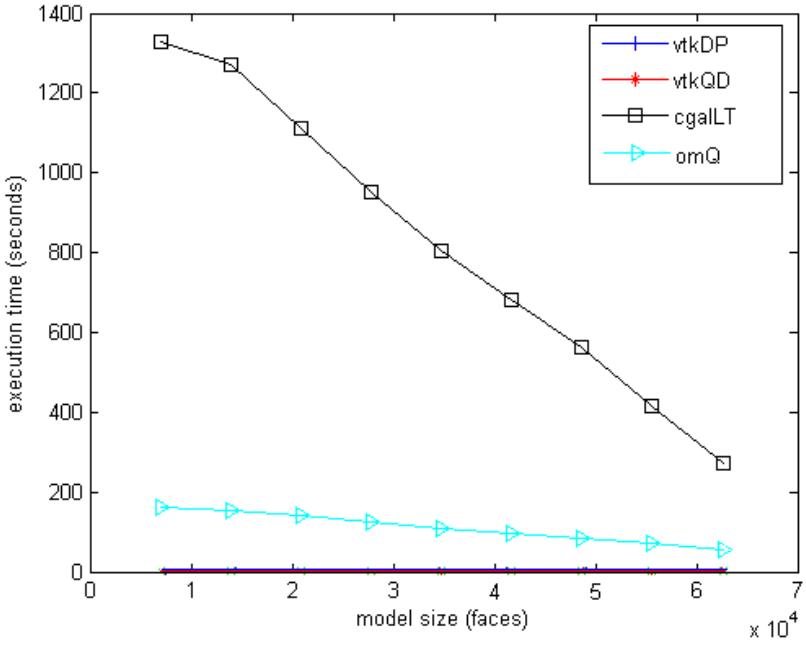
Slika 14. Graf usporedbe algoritama s obzirom na maksimalnu vrijednost pogreške

Kada se uzme u obzir srednja vrijednost pogreške (Slika 15.) dolazi do promjene u rangu. Algoritam Lindstrom-Turk iz biblioteke CGAL preuzima prvo mjesto od algoritma vtkQuadricDecimation. Nadalje algoritam OpenMesh Quadric preuzima treće mjesto od algoritma vtkDecimationPro.

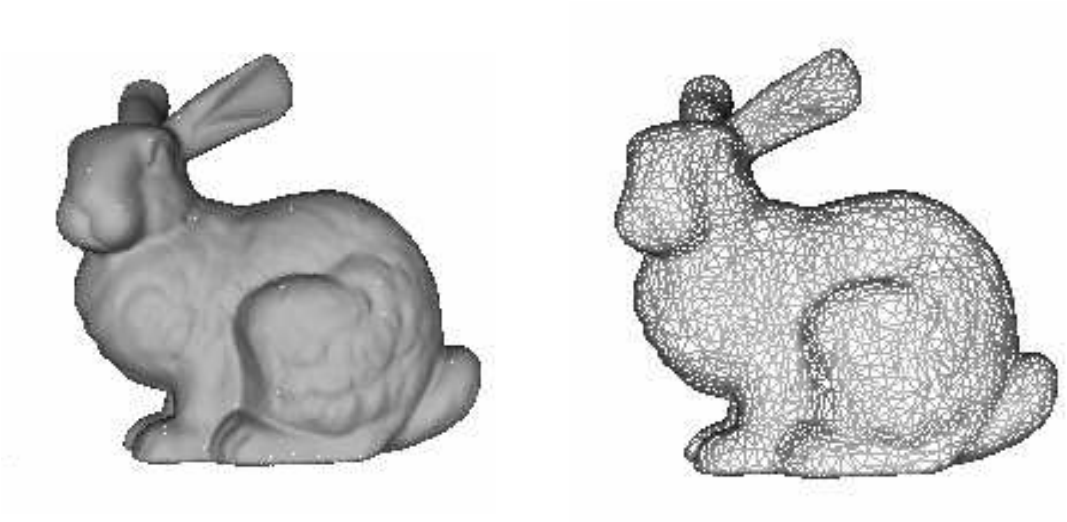


Slika 15. Graf usporedbe algoritama s obzirom na srednju vrijednost pogreške

Još jedan važan kriterij je uzet u obzir, a to je vrijeme izvođenja algoritama. Algoritmi iz biblioteke VTK imaju zavidno vrijeme izvođenja (do tri sekunde) u odnosu na ostale algoritme posebice Lindstrom-Turk algoritam (Slika 16.).



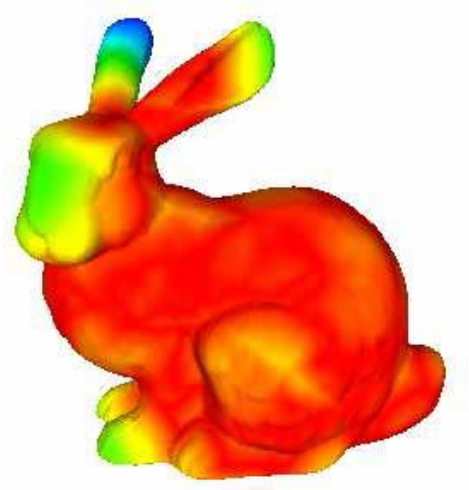
Slika 16. Graf usporedbe algoritama s obzirom na vrijeme izvođenja



Slika 17. Usporedba mreža modela, originalni model (lijeva strana) i simplificirani model (redukcija od 90%) (desna strana) uz algoritam Lindstrom-Turk

Iz ovih analiza može se izvesti zaključak da ako želimo srednju devijaciju između originalne mreže i simplificirane svesti na minimum onda je najbolje koristiti algoritam Lindstrom-Turk iz CGAL biblioteke, a ako želimo maksimalnu vrijednost pogreške svesti na minimum onda se predlaže algoritam vtkQuadricDecimation za korištenje. Nadalje algoritam Lindstrom-Turk je zasigurno jedno od najboljih rješenja no potrebno mu je najviše vremena za izvođenje to se posebno odnosi na ogromne mreže. Tako bi npr. za mrežu sa oko milijun poligona trebalo od dva do četiri sata da se simplificira pomoću Lindstrom-Turk algoritma.

Kao što je već prije bilo navedeno, program Metro je sposoban razliku između originalnog i simplificiranog modela prikazati i grafički. Pritom će sve točke na mreži originalnog modela obojati bojom koja odgovara nekoj izračunatoj vrijednosti pogreške za te točke. Područja koja su obojana crvenom bojom mogu se smatrati kao područja u kojima prevladava relativno mala pogreška, a područja u kojima boja prelazi iz crvene u žutu, zelenu i plavu mogu se smatrati područjima u kojima pogreška raste i to redom kako su navedene boje Slika (18.).



Slika 18. Prikaz grafičke usporedbe na originalnom modelu zeca

Zaključak

U ovom radu proučeno je važno područje u obradi mreža: smanjivanje broja poligona objekata. To je istraživački vrlo aktivno područje zbog nastojanja da se sve složeniji objekti koji u današnje vrijeme nastaju mogu simplificirati i koristiti što brže i optimalnije u realnom vremenu. Tako su razvijeni mnogi algoritmi koji omogućavaju efikasnu simplifikaciju i implementirani su u razne biblioteke namijenjene obradi mreža. Uspoređeni su neki od algoritama iz takvih besplatnih biblioteka i na temelju rezultata usporedbe zaključeno je da se ne može točno specificirati najbolji algoritam jer isti algoritmi mogu dati drugačije rezultate na različitim tipovima mreža. Među svim tim algoritmima postoje dva koji se izdvajaju od ostalih, a to su algoritam Lindstrom-Turk iz biblioteke CGAL i algoritam vtkQuadricDecimate iz biblioteke VTK. Jedan i drugi daju zadovoljavajuće rezultate kad je riječ o simplifikaciji, a algoritam vtkQuadricDecimate uz to ima i odličnu brzinu izvođenja. U segmentu izvođenja Lindstrom-Turk algoritam pati od značajne sporosti u odnosu na ostale. Svi algoritmi koji su obrađeni kroz ovaj rad su otvorenog koda te se mogu modificirati kako bi se dobili još bolji rezultati tj. bolji algoritmi.

Literatura

[1] Hoppe Hughes. Mesh Optimization, In Proceedings of ACM SIGGRAPH 93, pg. 19-26, 1993.

[2] Luebke D., Reddy M., Cohen Jonathan D., Varshney A., Watson B. and Huebner R. Level of Detail for 3D Graphics, Morgan Kaufmann, 2003.

[3] Ronfard R. and Rossignac J. Full-Range Approximations of Triangulated Polyhedra, In Proceedings of Eurographics, Vol. 15, C-67, 1996.

[4] J. Rossignac and P. Borrel, “Multi-resolution 3D approximations for rendering complex scenes”, Geometric Modeling in Computer Graphics, Springer Verlag, Berlin, 1993, pp. 455-465.

[5] Visualization Tookit Library (VTK). Retrieved August 2008 from http://www.vtk.org/

[6] Computational Geomerty Algorithms Library (CGAL). Retrieved August 2008 from thttp://www.cgal.org/

[7] OpenMesh Library. Retrieved August 2008 from http://www.openmesh.org/

Sažetak

Naslov: Smanjenje broja poligona objekata

Ovaj rad opisuje algoritme za smanjenje broja poligona objekata. U prvom poglavlju su obrađeni osnovni pojmovi vezani uz mrežu poligona. Zatim je dan pregled i opis algoritama i metoda koji se koriste za simplifikaciju i izračun pogreške koja nastaje simplifikacijom. U četvrtom poglavlju prikazani su algoritmi koji su implementirani u besplatne biblioteke. Konačno, algoritmi su uspoređeni i analizirani na primjerima te je donesen zaključak ta temelju rezultata.

Ključne riječi: poligon, mreža, trokut, točka, algoritam, simplifikacija

Abstract

Title: Polygonal object simplification

This thesis describes simplification algorithms that are used on polygonal objects. In the first chapter, basic terms on polygonal mesh were defined. Then an overview on algorithms and methods which are used for simplification and error calculation was given. In the fourth chapter algorithms that are implemented in free libraries were showed. Finally, algorithms were compared and analyzed on different examples and conclusion was reached based on the results.

Keywords: polygon, mesh, triangle, vertex, algorithm, simplification