Poligonal Basitleştirme Algoritmalarının Karşılaştırılması
(A Comparative Study on Polygonal Mesh Simplification Algorithms)

Murat Yirci[1,2], İlkay Ulusoy[2]

1. Görüntü İşleme Müdürlüğü
Mikroelektronik, Güdüm ve Elektro-Optik Grubu, Aselsan A. Ş.
myirci@mgeo.aselsan.com.tr

2. Elektrik ve Elektronik Mühendisliği Bölümü
Orta Doğu Teknik Üniversitesi
ilkay@metu.edu.tr

Özetçe

Poligonal yüzey modeller bilgisayar grafiği ve geometri işleme alanlarında üç boyutlu nesnelerin tanımlanmasında yaygın olarak kullanılmaktadır. Ancak gelişen teknoloji ile birlikte oluşturulan modeller gün geçtikçe daha da karmaşık hale gelmektedir. Poligonal modellerin karmaşıklık düzeylerinin artmasıyla birlikte gösterim kalitesi de artmaktadır ancak aynı zamanda modelin işlenmesi de bir o kadar zorlaşmaktadır. Bu nedenle poligonal modellerin birçoğunun oluşturulduktan sonra basitleştirilmesi gerekmektedir. Yaygın olarak kullanıldıkları için poligonal basitleştirme algoritmaları üç boyutlu poligonal modellerin işlenmesinde kullanılan hemen her kütüphaneye eklenmişlerdir. Bu çalışmada açık kaynak kodlu olarak geliştirilen ve geliştiriciler tarafından sıklıkla kullanılan CGAL, VTK ve OpenMesh kütüphanelerindeki poligonal basitleştirme algoritmaları Metro geometrik hata ölçüm aracı kullanılarak karşılaştırılmıştır. Bu şekilde kendi projelerinde poligonal basitleştirme algoritmalarını kullanmak isteyen geliştiricilere yol gösterilmesi amaçlanmıştır.

Abstract

Polygonal meshes are a common way of representing three dimensional surface models in many different areas of computer graphics and geometry processing. However, with the evolution of the technology, polygonal models are becoming more and more complex. As the complexity of the models increase, the visual approximation to the real world objects get better but there is a trade-off between the cost of processing these models and better visual approximation. In order to reduce this cost, the number of polygons in a model can be reduced by mesh simplification algorithms. These algorithms are widely used such that nearly all of the popular mesh editing libraries include at least one of them. In this work, polygonal simplification algorithms that are embedded in open source libraries: CGAL, VTK and OpenMesh are compared with the Metro geometric error measuring tool. By this way we try to supply a guidance for developers for publicly available mesh libraries in order to implement polygonal mesh simplification.

1. Giriş

Poligonal modeller değişik uygulama alanlarında değişik yöntemler kullanılarak oluşturulabilirler. Örneğin, bilgisayarla görmede lazer tarayıcılar kullanılırken, bilimsel gösterim (scientific visualization) alanında CT (Computer Tomography) ve MRI (Magnetic Resonance Image) ile elde edilen hacimsel verilerden eşyüzeyler çıkartılmaktadır. Bunlara ek olarak, uzaktan algılamada uydu fotoğraflarından üç boyutlu yeryüzü modelleri elde edilmekte, bilgisayar grafiği ve bilgisayar yardımıyla tasarlamada çeşitli araçlar kullanılarak üç boyutlu modeller oluşturulmaktadır. Günümüzde gelişen teknoloji ve uygulanan yeni yöntemler sayesinde birkaç milyon poligon içeren çok detaylı karmaşık modeller elde edilebilmektedir.

Özellikle modellerin girintili çıkıntılı bölgelerinde poligon sayısının fazla olması modelin görünüm kalitesini artırırken görece daha düz olan kısımlarda gereğinden fazla poligon kullanımı çeşitli zorluklar çıkarabilmektedir. Bu nedenle yukarıda bahsedilen yöntemler ile elde edilen modeller çoğu zaman ilgili uygulamalar açısından optimize edilmemiş ham modeller olarak düşünülebilir. Poligonal basitleştirme algoritmaları kullanılarak bu modellerdeki poligon sayısı modellerin görünüm kalitesi mümkün olduğunca korunmaya çalışılarak azaltılabilir.

Basitleştirilmiş bir poligonal model orijinal haline göre daha kolay işlenebilmektedir. Örneğin basit bir model boyanarak ekranda daha kolay görüntülenebilir, parametrize edilebilir. Bunlara ek olarak bilgisayar grafiğinde sıklıkla kullanılan ışın izleme ve çarpışma testi gibi işlemler daha hızlı gerçekleştirilebilir. Simülasyon ve bilgisayar oyunu gibi gerçek zamanlı ve inter-aktif uygulamalarda nesnelerin bakış noktasına olan uzaklığına göre aynı nesnenin değişik karmaşıklıktaki veya çözünürlükteki modelleri kullanılabilir (level of detail). Ayrıca, basitleştirilmiş bir model bilgisayar ağlarından aşamalı olarak gönderilebileceği gibi [1], hafızada ve sabit bellekte daha az yer kaplar. Bu yönüyle poligonal basitleştirme algoritmaları aynı zamanda model sıkıştırma amacıyla da kullanılabilir.

978-1-4244-4436-6/09/$25.00 ©2009 IEEE

Poligonal basitleştirme son yirmi yılda akademik olarak oldukça aktif olan bir konudur. Bu güne kadar birçok algoritma ve yöntem önerilmiştir [2]. Konunun çok çalışılmış olmasının nedeni üç boyutlu modellerin kullanıldığı hemen her uygulamada bu algoritmalara ihtiyaç duyulmasındandır. Yaygın olarak kullanılmalarına karşın bu algoritmaların kodlanması özellikle konu hakkında detaylı bilgisi olmayan geliştiriciler için zor olabilmektedir. Bu nedenle poligonal basitleştirme algoritmaları üç boyutlu modellerin işlenmesinde kullanılan hemen her kütüphaneye eklenmişlerdir. Böylece, geliştiriciler bu kütüphaneler aracılığıyla kendi uygulamalarında çeşitli basitleştirme algoritmalarını kullanabilirler.

Bu makalede özellikle akademik çevrelerce sıklıkla kullanılmakta olan açık kaynak kodlu CGAL [3], VTK [4] ve OpenMesh [5] kütüphanelerindeki poligonal basitleştirme algoritmaları çalışma zamanı ve elde edilen sonuçlar açısından karşılaştırılmıştır. Karşılaştırmada geometrik hata ölçüm aracı olarak Metro [6] kullanılmıştır. Bu çalışma ile uygulamalarında poligonal basitleştirmeye ihtiyaç duyan geliştiricilere kullanacakları kütüphaneyi seçmelerinde yol gösterilmesi amaçlanmıştır.

2. Poligonal Basitleştirme Algoritmaları'nın Sınıflandırılması

Poligonal basitleştirme algoritmaları birçok açıdan sınıflandırılabilir. Basitleştirme algoritmalarında genel olarak iki tip algoritma akışı vardır. Bunlardan ilkinde model basitleştirilirken iteratif bir yöntem izlenerek her bir adımda modelden belirli sayıda köşe ve poligon atılır. İstenilen basitleştirme oranına ulaşıldığında algoritmanın çalışması sonlandırılır. Diğer yöntem de ise modeller tek bir seferde iteratif bir sürece girilmeden basitleştirilir. İteratif algoritmalar daha yaygın olarak kullanılmakta ve kullanıcıya basitleştirmede daha çok kontrol imkânı tanımaktadır. Buna karşın, tek seferde çalışan algoritmalar genellikle kısa sürelerde çalışmaktadır.

Poligonal basitleştirmede dikkate alınan hususlardan bir diğeri de algoritmalar tarafından girdi olarak kabul edilen modellerin topolojik ve yapısal özellikleridir. Örneğin, bazı algoritmalar sadece üçgenlerden oluşan ve manifold topolojiye sahip modelleri kabul ederken diğerleri böyle bir ayrım yapmamaktadır. Bir diğer sınıflandırma da algoritmaların basitleştirmede kullandığı operatörlere bakılarak yapılabilir. Basitleştirmede kullanılan operatörlere örnek olarak köşelerin yok edilmesi (vertex decimation), kenarların yok edilmesi (edge contraction), kümelenmiş köşelerin birleştirilmesi (vertex clustering) ve bu yöntemlerin çeşitli türevleri verilebilir. Günümüzde kabul gören birçok algoritma kenarların yok edilmesi operatörü ve bu operatörün türevlerini kullanmaktadır.

Çoğu basitleştirme algoritması işletimleri sırasında hata hesaplamaları yapar. Yapılan bu hesaplamalar basitleştirme sürecinde kararların verilmesinde kullanılır. Bazı algoritmalar hata hesaplamalarını yerel değişkenlere göre yaparken bazıları da global değişkenleri dikkate alır. Global hata hesaplaması yapan algoritmalar genellikle daha iyi sonuçlar verirler ancak buna karşın çalışma süreleri de uzundur. Hata hesaplamasını yerel olarak yapan ancak hesaplanan hata miktarlarını biriktirerek ilerleyen kimi algoritmalar da mevcuttur. Hata biriktirme sayesinde bu algoritmalar bir nevi hafızaya kavuşarak modellerin belirli bölgelerinde hata oranının daha fazla artmasına engel olurlar. Bu algoritmalar çalışma zamanı ve elde edilen sonuçlar dikkate alındığında genellikle iyi sonuçlar vermektedir. Son olarak ta poligonal basitleştirme algoritmaları topoloji koruyan ve korumayan olmak üzere iki gruba ayrılabilir. Topoloji koruyan algoritmalar manifold topolojiye sahip modelleri basitleştirirken modelin manifold topolojisine sadık kalır ve varsa modellerin deliklerini kapatmazlar. Topoloji korumayan algoritmalar ise modelin topolojik özelliklerini koruyacakları yönünde bir garanti vermezler. Modellerin topolojilerinin korunması bilimsel gösterim gibi alanlarda önemli iken bilgisayar grafiğinde önemli olmayabilir.

3. Poligonal Model İşleme Kütüphaneleri ve Basitleştirme Algoritmaları

Üç boyutlu modellerin tanımlanmasında iki temel bilgi kullanılır. Bunlardan ilki geometrik bilgi, ikincisi ise topolojik bilgidir. Geometrik bilgi modeldeki poligon köşelerinin üç boyutlu koordinatlarını içerirken topolojik bilgi bu köşelerin bir birlerine nasıl bağlandığını belirler. Poligonal modeller işlenilmek istenildiğinde, modellerin geometrik ve topolojik bilgilerinin programlara diğer bir ifade ile veri yapılarına aktarılması gerekir. Hafıza ve zaman açısından etkili, kullanımı kolay ve geniş kapsamlı veri yapılarının geliştirilmesi çok zor ve karmaşık bir işlemdir. Bu nedenle poligonal modellerin işlenebilmesi için önce bu veri yapılarını içeren kütüphaneler oluşturulmuş daha sonra zamanla bu kütüphanelere poligonal basitleştirme, parametrize etme ve alt bölümleme gibi çeşitli algoritmalar eklenmiştir.

Bu makalede CGAL, VTK ve OpenMesh kütüphaneleri poligonal basitleştirme algoritmaları açısından ele alınmıştır. Bunlar açık kaynak kodlu, ticari amaçlar dışında kullanımı ücretsiz olan ve C++ programlama dili kullanılarak geliştirilmiş kütüphanelerdir.

3.1. CGAL - Computational Geometry Algorithms Library

CGAL [5] kütüphanesi çeşitli ülkelerden dokuz üniversite ve bilimsel kuruluştan oluşan bir konsorsiyum tarafından 1996 yılından itibaren geliştirilmektedir. CGAL'de hesaplamalı geometri, bilgisayarla görme, bilimsel gösterim ve bilgisayar grafiği ile ilgili birçok algoritma mevcuttur. Bu çalışmada kütüphanenin Ağustos 2007'de yayınlanan 3.2.1 sürümü kullanılmıştır. CGAL'de poligonal basitleştirme için iki farklı strateji izlenebilir.

3.1.1. Lindstrom-Turk (LT) Stratejisi

Lindstrom ve Turk [7] tarafından önerilen bu algoritmada girdi olarak kabul edilen model iteratif olarak basitleştirilir. Basitleştirmenin her bir adımında bir optimizasyon problemi çözülerek modelden atılacak köşe ve poligonlar modelin yüzey alanını ve hacmini en az değiştirecek şekilde seçilir.

3.1.2. Kenar Uzunluğu – Orta Nokta (KuOn) Stratejisi

Bu algoritma oldukça basit olup basitleştirme adımlarında yok edilmesi için seçilen kenarlar kısa olandan uzun olana göre sırayla seçilir. Seçilen kenarın yerine konulacak köşenin yeri ise yok edilen kenarın orta noktası olarak belirlenir.

3.2. VTK - Visualization Toolkit

VTK [4] kütüphanesi asıl olarak Kitware (www.kitware.org) şirketi tarafından geliştirilmekle birlikte dünyanın birçok ülkesinden gönüllüler de kütüphanenin geliştirilmesine katkıda bulunmaktadır. Kütüphane C++'ın yanında Java, Tcl/Tk ve

Phyton ile de kullanılabilmektedir. VTK bünyesinde modelleme, görüntü işleme, bilimsel gösterim ve hesaplamalı geometri ile ilgili çok çeşitli algoritmalar barındırmaktadır. Bu çalışmada kütüphanenin Ocak 2008'de yayınlanan 5.0.4 versiyonu kullanılmıştır. VTK'da üç farklı poligonal basitleştirme algoritması bulunmaktadır.

3.2.1. vtkDecimatePro (vtkDP)

Bu algoritma Schroeder [8] tarafından önerilen algoritmanın biraz değiştirilmiş şeklidir. vtkDecimatePro'da modelin topolojisi korunmayabilir ve basitleştirme operatörü olarak köşe yok etme yerine kenar yok etme kullanılır. Ayrıca hata biriktirme opsiyonu da eklenmiştir.

3.2.2. vtkQuadricDecimation (vtkQD)

vtkQuadricDecimation algoritması Garland ve Heckbert [9] ile Hoppe'un [10] çalışmalarına dayanmaktadır. Garland ve Heckbert yeni bir hata hesaplama ve biriktirme yöntemi bulmuş, (Quadric Error), Hoppe ise köşe dikme ve renkleri gibi model özelliklerini de bu hesaplamaya dâhil etmiştir. vtkQuadricDecimation'da opsiyonel olarak her iki yöntem de kullanılabilmektedir.

3.2.3. vtkQuadricClustering (vtkQC)

vtkQuadricClustering tek seferde çalışan bir algoritma olarak diğerlerinden farklılık gösterir ve Lindstrom'un [11] önerdiği basitleştirme algoritmasına dayanır.

3.3. OpenMesh

OpenMesh [5] kütüphanesi Aachen Üniversitesi tarafından geliştirilmektedir. Bu kütüphane diğer iki kütüphaneye göre daha az sayıda algoritma içermektedir. Bu çalışmada kütüphanenin Şubat 2007'de yayınlanan 1.1.0 sürümü kullanılmıştır. OpenMesh kütüphanesinde beş faklı modül: Quadric, Roundness, Normal Flipping, Independent Sets ve Progressive Mesh, birbiri ardına eklenilerek değişik basitleştirme algoritmaları elde edilebilir. Algoritma oluşturulurken Quadrics ve Roundness modüllerinden en az bir tanesi muhakkak seçilmelidir. Diğer modüller aktif modüller olmayıp basitleştirmenin sınırlandırılmasında ve basitleştirme işlemi sırasında bilgi toplanılmasında kullanılırlar. Independent Sets modülü basitleştirmeyi çok fazla sınırlandırdığından çoğu zaman istenilen basitleştirme oranına ulaşılamaz. Progressive Mesh ise aşamalı modellerin oluşturulması için bilgi toplar. Son olarak Normal Flipping modülü basitleştirme sırasında poligonların yüzey dikmeleri arasındaki açı değişimlerini kontrol eder ve bu açıların iki basitleştirme adımı arasındaki değişimini sınırlar.

3.3.1. Quadrics Modülü (Qmod)

Garland ve Heckbert'in [9] önerdiği "Quadric Error" yöntemi ile birlikte kenar yok etme operatörünün değiştirilmiş bir şekli olan yarım kenar yok etme operatörünü kullanır.

3.3.2. Roundness Modülü(Rmod)

Bu modül modeldeki üçgenlerden eş-üçgenlere yakın olanları korurken diğerlerinin modelden çıkarılmasına öncelik verir.

3.4. Algoritmaların Sınıflandırılması

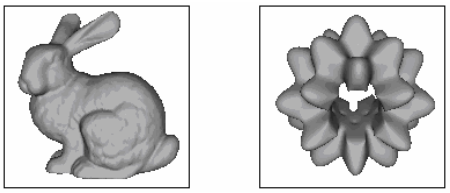
CGAL, VTK ve OpenMesh'teki basitleştirme algoritmalarının özellikleri bölüm 2'de bahsedilen sınıflandırma kümeleri açısından Tablo 1'de verilmiştir.

Tablo 1: Basitleştirme algoritmalarının sınıflandırılması

Algor.	Model Türü	Topoloji Koruma	Basitleştirme Operatörü	Hata Hesap.
LT	Üçgensel, Manifold	Korur	Kenar yok etme	Global
KuOn	Üçgensel, Manifold	Korur	Kenar yok etme	Yerel
vtkDP	Üçgensel	Seçime bağlı	Kenar yok etme	Yerel
vtkQD	Üçgensel	Koruma-yabilir	Kenar yok etme	Yerel
vtkQC	Poligonal	Koruma-yabilir	Köşelerin birleştirilmesi	Yerel
Qmod	Poligonal	Korur	Yarım kenar yok etme	Yerel
Rmod	Poligonal	Korur	Yarım kenar yok etme	Yerel

4. Basitleştirme Algoritmalarının Karşılaştırılması

Bölüm 3'te bahsedilen CGAL ve VTK kütüphanelerindeki tüm algoritmalar karşılaştırmaya dâhil edilmiştir. OpenMesh kütüphanesinde ise Quadrics modülü tek başına kullanılırken, Roundness modülü Normal Flipping modülü ile birlikte kullanılmıştır (RNFmod). vtkDP'nin hata biriktirme özelliği açılırken (vtkDPhb), vtkQD'nin de köşe dikmelerini hata hesaplamasına katan (vtkQDkd) opsiyonu ayrıca seçilerek toplamda sekiz algoritma karşılaştırılmıştır.



(a) (b)

Şekil 1: (a) Stanford Bunny modeli 34834 köşe ve 69451 üçgenden oluşmaktadır (www-graphics.stanford.edu). (b) Bumpy Torus modeli 16815 köşe ve 33630 üçgenden oluşmaktadır (http://shapes.aim-at-shape.net).

Seçilen tüm algoritmalar yüzey eğimi ve topolojik açıdan farklı özelliklere sahip iki adet poligonal model (Şekil 1) üzerinde toplamda dört farklı basitleştirme oranında: %30, %50, %70, %90 çalıştırılmıştır (kullanılan bilgisayar: 2.4 Ghz işlemci hızı, 3GB bellek ve Windows Xp). Basitleştirme sonucunda elde edilen modeller Metro [6] aracı kullanılarak karşılaştırılmıştır. Metro aracı ile aynı nesnenin basit ve karmaşık olan iki modeli arasındaki ortalama ve en büyük uzaklık (hata) ölçülerek basit modelin orijinal modele ne kadar benzediği geometrik olarak ölçülebilmektedir. Ayrıca algoritmalar çalışma süreleri açısından da incelenmiştir.

4.1. Sonuç ve Vargılar

Elde edilen sonuçlar Tablo 2'de verilmiştir. Özetlemek gerekirse Stanford Bunny modelinde vtkQD ve LT algoritmaları ön plana çıkmıştır. Bu iki algoritmadan vtkQD en büyük hata ölçümünde tüm basitleştirme yüzdelerinde birinci, LT algoritması ise ikinci olmuştur. Ortalama hata ölçümünde ise LT algoritması vtkQD'den birinciliği devralmıştır. Qmod algoritması ise en büyük hata ölçümünde

Tablo 2: Ortalama ve en büyük hata değerleri (değerler normalize edilmiş ve 10^6 ile çarpılmıştır) ile çalışma süreleri (saniye)

Algoritmalar		Stanford Bunny Model				Bumpy Torus Model			
		%30	% 50	%70	90%	%30	% 50	%70	90%
LT	En Bü.	480	635	1099	2454	1133	1581	1980	8026
	Ort.	16	32	60	148	32	70	134	376
	Süre	561	804	1110	1330	214	325	444	543
KuOn	En Bü.	2486	3085	4700	10186	5639	6126	7349	16487
	Ort.	64	128	248	791	140	294	634	2218
	Süre	165	241	320	365	72	105	144	178
Qmod	En Bü.	3365	4540	5339	7597	1246	1365	2544	6002
	Ort.	16	44	92	284	46	119	267	872
	Süre	84	108	140	160	33	41	51	59
RNFmod	En Bü.	4695	6142	6857	17966	1929	3160	6245	18407
	Ort.	40	88	224	1023	129	278	651	1955
	Süre	137	168	222	299	41	52	73	93
vtkDPhb	En Bü.	1530	1970	2406	7241	2676	2698	3724	12345
	Ort.	48	84	152	723	131	255	556	1487
	Süre	1.46	1.85	2.04	2.46	0.66	0.89	0.95	1.23
vtkQD	En Bü.	248	376	651	2094	601	794	2546	3675
	Ort.	16	40	76	196	35	82	164	481
	Süre	1.96	2.4	2.54	2.63	0.76	0.84	0.86	0.98
vtkQDkd	En Bü.	1463	4060	6334	12448	518	734	1516	3677
	Ort.	76	212	519	1550	36	82	165	481
	Süre	5.3	6.1	6.27	6.46	2.29	2.33	2.4	3.01
vtkQC	En Bü.	4344	7285	7792	12568	1959	2731	5347	16965
	Ort.	48	80	140	416	87	156	296	948
	Süre	1.32	0.84	0.54	0.28	0.51	0.39	0.26	0.14

gerilerde kalmasına karşın ortalama hata ölçümünde en iyi performansı gösteren üçüncü algoritma olmuştur. Diğer algoritmalar daha sonraki sıralarda yer alırken belirli bir düzen gözlenememiştir. Ancak RNFmod ve vtkQC algoritmaları en büyük hata ölçümü açısından genellikle sonlarda yer alırken ortalama hata açısından orta sıralarda yer almışlardır. Ortalama hata ölçümünde vtkQDkd ve KuOn algoritmalarının en fazla hata üreten algoritmalar oldukları saptanmıştır.

Bumpy Torus modelinde Stanford Bunny modelinde başarılı olamayan vtkQDkd algoritmasının her iki hata ölçümünde de ön sıralara çıktığı gözlenmiştir. Bu algoritma köşe dikmelerini de dikkate aldığı için Bumpy Torus gibi girinti ve çıkıntısı fazla olan bir modelde başarıyı yakalamıştır. Ortalama olarak en az hata üreten algoritma ise yine LT algoritması olmuştur. KuOn ve RNFmod algoritmalarının her iki hata ölçümünde de son sıralarda yer alması bu algoritmaların yüzey alanı keskin eğimli bölgeler içeren modellerde başarılı olamadığını göstermiştir.

Algoritmaların çalışma sürelerine bakıldığında her bir iterasyonda optimizasyon problemi çözen ve global hata hesaplaması yapan LT algoritması çalışma süresi açısından en çok zaman tüketen algoritma olmuştur. Buna karşın KuOn algoritmasının oldukça basit olmasına rağmen uzun çalışma süreleri gerektirmesi CGAL kütüphanesinde kullanılan veri yapılarının yeterli verimlilikte olmamasından kaynaklanabilir. OpenMesh algoritmaları orta sıralarda yer alırken VTK kütüphanesindeki tüm algoritmaların diğerlerine göre çok hızlı olduğu görülmüştür.

Sonuç olarak çalışma zamanı ve yapılan hata ölçümlerine göre vtkQD algoritmasının her iki modelde de iyi sonuçlar verdiği görülmüştür. Ancak geliştiricilerin çalışma süresi açısından bir kısıtları yok ise ortalama hata ölçümlerinde en iyi performansı gösteren LT algoritması tercih edilebilir. Bunlara ek olarak eğimli ve girintili çıkıntılı yüzeylere sahip modellerde vtkQDkd algoritması da denenebilir. OpenMesh kütüphanesi kullanılacak ise Quadrics modülü her zaman ilk olarak tercih edilmelidir.

5. Kaynakça

[1] H. Hoppe, “Progressive meshes”, *in Proc. SIGGRAPH 96, 1996, pp. 99-108.*

[2] D. P. Luebke, "A developers survey of polygonal simplification algorithms", *IEEE Computer Graphics and Applications, vol. 21, pp. 24-35, 2001.*

[3] Computational Geometry Algorithms Library, Retrieved Dec. 2008 from *www.cgal.org*

[4] Visualization Toolkit Library, Retrieved Dec. 2008 from *www.vtk.org*

[5] OpenMesh, Retrieved Dec. 2008 from *www.openmesh.org*

[6] P. Cignoni, C. Rochini, and R. Scopigno, "Metro: measuring error on simplified surfaces", *Computer Graphics Forum, vol. 17, pp. 167-174, 1998.*

[7] P. Lindstrom, and G. Turk, "Fast and memory efficient polygonal simplification", *in Proc. IEEE Visualization 98, pp. 279-286.*

[8] W. Schroeder, J. Zarge, and W. Lorenson, "Decimation or triangle meshes", *in Proc. SIGGRAPH 92, 1992, pp. 65-70.*

[9] M. Garland, and P. Heckbert, "Surface simplification using quadric error metrics", *in Proc. SIGGRAPH 97, 1997, pp. 209-216.*

[10] H. Hoppe, “New quadric metric for simplifying meshes with apperance attributes”, *in Proc. IEEE Visualization 99, 1999, pp. 59-66.*

[11] P. Lindstrom, "Out-of-core simplification of large polygonal models", *in Proc. SIGGRAPH 00, 2000, pp. 259-262.*