Poligonal Model Optimizasyonu Üzerinde İyileştirmeler*

Muhammet Mustafa Özdal, Uğur Güdükbay ve Bülent Özgüç

Bilkent Üniversitesi, Bilgisayar Mühendisliği Bölümü, 06533 Bilkent Ankara
e-mail: {ozdal, gudukbay, ozguc}@cs.bilkent.edu.tr

Özetçe

Bilgisayar grafiğinde kullanılan poligonal modellerin karmaşıklığını azaltmak için basitleştirme teknikleri kullanılmaktadır. Hugues Hoppe'nin Poligonal Model Optimizasyonu çok yüksek işlemci zamanı gerektiren, ancak çok kaliteli basitleştirme yapan bir yöntemdir [2]. Bu makalede bu yöntemi hızlandırmak için iyileştirmeler önerilmiş ve bu iyileştirmelerin etkileri, gerçekleştirilen program değişik modeller üzerinde denenerek gösterilmiştir.

1 Giriş

Bilgisayar grafiğinde mevcut teknoloji yüzbinlerce poligon içeren çok karmaşık modellerin üretilmesine ve kullanılmasına imkan tanımaktadır. Ancak donanım performansındaki artışlar, modellerin karmaşıklığı ile yarışamamaktadır. Bu nedenle modellerin verimli basitleştirme yöntemleri kullanılarak basitleştirilmesine ihtiyaç duyulmaktadır. Bu amaç için, modeller genellikle poligon kümeleri kullanılarak modellenmekte ve basitleştirme yöntemleri bu modellere uygulanmaktadır.

Basitleştirme işlemi değişik stratejiler kullanılarak yapılabilmektedir. Üçgenler içeren modeller için önerilen basitleştirme yöntemlerinin sınıflandırılması ve karşılaştırılması detaylı bir şekilde [1]'de verilmektedir. Yöntemlerin analizi, basitleştirme işleminde iki ana kriterin önemli olduğunu göstermektedir. Bunlar *işlemci zamanı* ve *basitleştirme kalitesi*'dir. Genellikle bu iki kriter birbiri ile ters orantılıdır. Eğer bir yöntem basitleştirmede iyi sonuçlar veriyorsa basitleştirme uzun zaman almakta, ya da bunun tam tersi olmaktadır.

Hugues Hoppe tarafından geliştirilen enerji optimizasyonuna dayalı model basitleştirme yöntemi çok fazla işlemci zamanı gerektiren ancak basitleştirme kalitesi açısından en iyi sonuçları veren yöntemlerden biridir [2]. Bu makalede, bu yöntemin işlemci zamanı ve mümkünse basitleştirme kalitesi üzerinde iyileştirmeler yapılması olanakları araştırılmış ve bu iyileştirmeler yöntemin kaynak kodları üzerinde gerçekleştirilerek performans karşılaştırması yapılmıştır.

Bu makalenin devamı şöyle organize edilmiştir. İlk olarak Hoppe'nin enerji optimizasyonuna dayalı model basitleştirme yöntemi anlatılmaktadır. Daha sonra yöntemi hızlandırmak için yapılan iyileştirmeler anlatılmaktadır. Daha sonra istatistiksel ve görsel sonuçlar verilmekte ve yapılan iyileştirmeler bu sonuçlara göre yorumlanmaktadır. Son olarak özet ve sonuçlar verilmektedir.

*Bu araştırma TÜBİTAK tarafından desteklenen EEEAG 198E018 no'lu proje kapsamında gerçekleştirilmiştir.

2 Enerji Optimizasyonuna Dayalı Yüzey Basitleştirme

Hugues Hoppe, enerji optimizasyonuna dayalı model basitleştirme yöntemini anlattığı makalesinde yüzey basitleştirme problemini şöyle tanımlamaktadır [2].

> "Üç boyutlu uzayda tanımlanmış olan bir noktalar kümesi ve üçgenlerden oluşan bir poligonal model (M_0) için, M_0 ile aynı topolojik yapıya sahip, nokta kümesine uyan ve daha az sayıda nokta içeren yeni bir poligonal model (M) üretmek"

Bir yüzey matematiksel olarak (K, V) çifti ile ifade edilmektedir. Burada K poligonal modeli tanımlamakta ve poligon köşelerinin, poligon kenarlarının ve poligonal yüzeylerin birbirleri ile olan ilişkilerini ifade etmektedir. V ise köşe kümesini tanımlamakta ve poligon köşelerinin üç boyutlu koordinatlarını ifade etmektedir.

Model basitleştirme işlemi, aşağıdaki enerji fonksiyonu minimize edilerek gerçekleştirilmektedir.

$$E(K, V) = E_{dist}(K, V) + E_{rep}(K) + E_{spring}(K, V). \quad (1)$$

Bu enerji fonksiyonunda yeralan E_{dist} terimi verilen nokta setindeki noktalardan modeldeki poligonal yüzeylere olan uzaklıkların karelerinin toplamını ifade etmektedir. Bu terimin optimizasyon işleminde kullanılmasının amacı modelin orjinal nokta kümesinden uzaklaşmasını engelleyerek modelin orjinalinin geometrisine sadık kalmasını sağlamaktır. Enerji fonksiyonunda yeralan ikinci terim olan E_{rep}, modeldeki nokta sayısının belli bir sabit ile çarpılması ile elde edilmektedir. Bu terimin amacı modeldeki nokta sayısını azaltarak daha basit bir model elde etmektir. Enerji fonksiyonunda yeralan üçüncü terim olan E_{spring} teriminin amacı optimizasyon işlemini düzenleyerek daha kısa sürede çözüme gidilmesini sağlamaktır.

Enerji fonksiyonundaki E_{dist} ve E_{rep} terimleri birbirleri ile yarışmaktadır. Diğer bir deyişle, köşe sayısının azaltılması, E_{rep} terimini azaltırken E_{dist} terimini arttırmakta, ya da bunun tam tersi olmaktadır. Kullanıcının belirlediği bir sabit, model basitleştirmenin derecesi ile elde edilen modelin kalitesi (orjinaline yakınlığı) arasında bir seçim yapmaya yarayan bir kontrol sağlamaktadır.

Yüzey optimizasyonu algoritması Şekil 1'de verilmektedir. Algoritma iki kısımdan oluşmaktadır. Bunlardan biri sabit bir model için enerji fonksiyonunu optimize eden alt optimizasyon, diğeri ise modeli değiştirerek fonksiyonu optimize eden bir ana optimizasyon işlemidir. Ana optimizasyon işlemi modeli değiştirmek için Şekil 2'de verilen transformasyonları kullanmaktadır.

Alt optimizasyon problemi `OptimizeVertexPositions(K, V)` altyordamı tarafından çözülmektedir. Bu altyordam poligonal modeli ve köşe pozisyonlarını girdi olarak alıp, enerji fonksiyonunu köşe pozisyonlarını değiştirerek optimize etmektedir. Burada E_{rep} sadece köşelerin sayısına bağlı olduğundan problem $E_{dist} + E_{spring}$ ifadesinin optimize edilmesine dönüşmektedir. Ancak, E_{dist} teriminin hesaplanması bu derece kolay değildir. Bu terim, verilen nokta setindeki noktalardan her biri için modeldeki en yakın poligonal yüzeye olan uzaklıkların karelerinin toplamını ifade etmektedir. Bu nedenle E_{dist} terimini hesaplamak için nokta setindeki her nokta için en yakın poligonu bulmak ve noktanın poligonal yüzeye uzaklığını bulmak gerekmektedir.

Alt optimizasyon işleminde iki altyordam kullanılmaktadır: `ProjectPoints(K, V)` ve `ImproveVertexPositions(K, B)`. `ProjectPoints` altyordamı nokta kümesindeki her noktaya modeldeki en yakın noktayı (poligonal yüzeyi) ve o noktanın üçgensel koordinatlarını bulmaktadır. `ImproveVertexPositions` altyordamı poligon köşelerinin pozisyonlarını üçgensel koordinatları kullanarak optimize etmektedir. `ImproveVertexPositions` altyordamı poligon köşe pozisyonlarını değiştirerek enerji fonksiyonunu optimize eder. Altyordam bu işi yaparken poligonal model ve üçgensel koordinat vektörleri değişmemektedir.

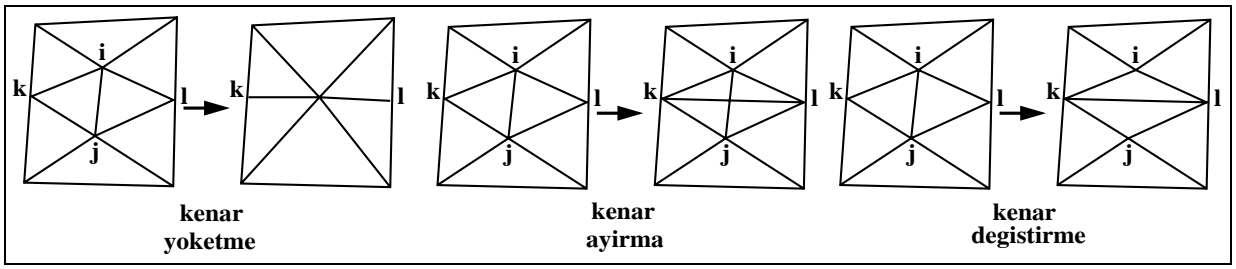
```
OptimizeMesh(K0, V0)
{
    K = K0
    V = OptimizeVertexPositions(K0, V0)

   // the outer minimization problem
   repeat
   {
      (K', V') = GenerateLegalMove(K, V)
      V' = OptimizeVertexPositions(K', V')
      if E(K', V') < E(K, V) then
         (K, V) = K', V')
      endif
   }
   until convergence
   return (K, V)
}
```

```
// the inner minimization problem
OptimizeVertexPositions(K, V)
{
   repeat
   {
      B = ProjectPoints(K, V)
      V = ImproveVertexPositions(K, B)
   }
   until convergence
   return V
}

GenerateLegalMove(K, V)
{
   Select a legal move K -> K'
   Locally modify V to obtain V' (for K')
   return (K', V')
}
```

Şekil 1: Enerji optimizasyonuna dayalı yüzey basitleştirme algoritması. (Algoritma, orjinal makaledeki terimlere sadık kalmak amacı ile İngilizce olarak verilmiştir.)



Şekil 2: Legal transformasyonlar

3 Yöntemde Yapılan İyileştirmeler

Programın denediği transformasyonların büyük bir kısmı başarısız transformasyonlardır. Başarısız transformasyonların sayısını azaltmak program üzerinde yapılabilecek en önemli iyileştirmelerden birisi olacaktır. Bu amaçla aşağıda anlatılan üç değişik yöntem kullanılmıştır.

3.1 Transformasyonların sadece kenar yoketme ile sınırlanması

Buna yönelik en önemli adım kenar transformasyonlarını sadece kenar yoketme ile sınırlamaktır. Hugues Hoppe'nin de belirttiği gibi [3] diğer iki transformasyon (kenar ayırma ve kenar değiştirme) bir yüzeyi nokta kümesinden elde etmekte faydalı işlemler olmakla birlikte, basitleştirme için çok gerekli işlemler değillerdir. Bu nedenle uygulanacak transformasyonlar sadece kenar yoketme ile sınırlanabilir. (Bu iki transformasyon kullanıldığında elde edilecek basitleştirilmiş modelin kalitesinde çok az da olsa bir iyileştirme olabilmektedir. Ancak bu işlemci zamanında kazanılacak iyileştirme kadar önemli olmamaktadır.)

3.2 Sonraki iterasyonlar için aday kenar kümesinin küçültülmesi

İşlemci zamanında yapılabilecek ikinci bir iyileştirme de optimizasyon işleminin sonraki safhalarında denenen işlemlere göre oransal olarak oldukça artan başarısız işlemlerin sayısını azaltmaktır. Bu amaçla, başlangıçtaki kenar sayısı kadar eleman içeren ve iki alanı olan bir dizi veri yapısı kullanılabilir. Dizi veri yapısının indisi kenarı belirlemekte ve bu indisteki birinci alan bu kenar yokedildiğinde enerjide (ΔE_{dist}) meydana gelecek değişikliği ve diğer alan ise bu kenar yokedildiğinde E_{spring}/κ'da meydana gelecek değişikliği tutmaktadır. Bu dizi başlangışta boştur. İlk iterasyonda (yay sabitinin ilk değeri için), başarısız transformasyonlar için kenarlar aday kenar kümesinden atıldıkça bu dizi doldurulacaktır. Başarısız transformasyonların çoğu enerji değerinin artmasından kaynaklanmaktadır. Bu nedenle aday kenar kümesinden atılacak olan bir kenarın yokedilmesi ile enerjide meydana gelebilecek değişiklik zaten program tarafından hesaplandığından bu değerlerin dizide saklanması programa fazla bir ek yük getirmemektedir.

Bu dizi, daha sonraki iterasyonlar için (değişik yay sabiti değerleri) aday kenar kümesini belirlemekte kullanılacaktır. Daha sonraki iterasyonlarda, her kenar için şu işlem tekrarlanacaktır: Dizinin bu kenar için saklanan elemanının ikinci alanı yay sabiti ile çarpılarak birinci alanı ile toplanacaktır. Bu değer sıfırdan büyükse bu kenar aday kenar kümesine eklenecektir. Bu şekilde başarısız işlemlerin büyük bir kısmı ileriki iterasyonlarda denenmemiş olacaktır. Bu dizinin her başarılı kenar yoketme işlemi sonrasında güncellenmesi amacı ile transformasyon uygulanmış her kenar için, bu kenarın artık varolmadığını gösteren bir işaret konulmaktadır.

Bu iyileştirme ile işlemci zamanında büyük bir kazanç elde edilmektedir. Bunun nedeni başarısız işlemlerinin büyük bir kısmının denenmesinin bu yöntemle önlenmesidir. Ancak, bu iyileştirmenin işlemlerin büyük bir kısmının gerçekleştirildiği ilk iterasyon için bir etkisi olmamaktadır.

3.3 Daha iyi sonuç verecek kenar yoketme işlemlerinin seçilmesi

Başarısız işlemlerin sayısını azaltmaya yönelik diğer bir iyileştirme de uygulanacak kenar yoketme işleminin seçimi ile ilgilidir. Orjinal algoritma, uygulanacak transformasyonu gelişigüzel olarak seçmektedir. Bu kenarın yokedilmesi legal ise ve toplam enerjiyi azaltıyorsa bu kenar yoketme işlemi hemen uygulanmaktadır. Bunun yerine dik iniş ("steepest descent") gibi stratejiler yokedilecek kenarın seçiminde kullanılabilir [4]. Bu stratejide bir kenarı yoketmenin bedelini gösteren bir fonksiyon tanımlanmakta ve her kenar yoketme işleminin bedeli bu fonksiyona göre hesaplanmaktadır. Buna göre kenarlar bir kuyruk veri yapısına yerleştirilmekte ve bu veri yapısı kullanılarak daima en az bedeli olan kenara yoketme işlemi uygulanmaktadır.

Bu yöntem çok işlemci zamanı gerektirdiği için dik iniş tekniğini andıran, ancak her işlemde "en iyi" kenar yerine, "daha iyi" kenarlar seçilmesini sağlayan şöyle bir teknik kullanılmıştır: Her iterasyonda yay sabitini değiştirmek yerine birkaç iterasyon için yay sabitini aynı tutarak yokedilen bir kenarın komşularını aday kenar kümesine eklememek.

Orjinal algoritmada bir kenar yokedildiğinde komşu kenarlar aday kenar kümesine eklenmekte ve her iterasyon bu kümedeki kenarların hepsi denenene kadar devam etmektedir. Bir kenar yoketme işleminin başarısız olduğunu varsayalım. Bunun komşuluğundaki başka bir kenar yokedildikten sonra bu kenarın enerji fonksiyonunu azaltma olasılığı denenmemiş olan diğer kenarların enerji fonksiyonunu azaltma olasılığından daha düşüktür. Bu nedenle önceden başarısız olan kenarın yerine diğer kenarların denenmesi daha mantıklıdır. Bunun için yokedilen kenarların komşularını aday kenar kümesine eklemeyerek diğer kenarların ilk önce denenmesi sağlanmıştır.

Bu yöntem basitleştirme kalitesini bir miktar arttırmasının yanında, ilk yay sabiti iterasyonundaki başarısız kenar transformasyonu sayısını azaltmaktadır.

Tablo 1: Orjinal program kullanılarak elde edilen deneysel sonuçlar

Model Adı	Orjinal Model				Basitleştirilmiş Model				İşlemci Zamanı (saniye)
	Köşe sayısı	Üçgen sayısı	E_{dist}	E_{toplam}	Köşe sayısı	Üçgen sayısı	E_{dist}	E_{toplam}	
hypersheet	2032	3832	0,0	0,2999	288	531	0,0119	0,0513	224
teapot	7704	15382	0,0	0,8124	245	487	0,0100	0,0355	470
mannequin	7834	15543	0,0	0,0636	1532	3018	0,0007	0,0023	433
oilpmp	22741	45478	0,0	2,3526	505	1006	0,0257	0,0762	1107
budha	543652	1087716	0,0	54,4111	2560	5532	0,1026	0,3586	25125

Tablo 2: İyileştirilmiş program kullanılarak elde edilen deneysel sonuçlar

Model Adı	Orjinal Model				Basitleştirilmiş Model				İşlemci Zamanı (saniye)
	Köşe sayısı	Üçgen sayısı	E_{dist}	E_{toplam}	Köşe sayısı	Üçgen sayısı	E_{dist}	E_{toplam}	
hypersheet	2032	3832	0,0	0,2999	279	513	0,0134	0,0519	103
teapot	7704	15382	0,0	0,8124	227	452	0,0121	0,0356	235
mannequin	7834	15543	0,0	0,0636	1518	2991	0,0008	0,0024	253
oilpmp	22741	45478	0,0	2,3526	508	1012	0,0281	0,0789	530
budha	543652	1087716	0,0	54,4111	2810	6032	0,1186	0,3996	13509

4 Deneysel Sonuçlar

Yukarıda anlatılan iyileştirmeler gerçekleştirilmiş ve sonuçlar incelenmiştir. Birinci iyileştirme (kenar ayırma ve kenar değiştirme işlemlerinin yokedilmesi) basitleştirme kalitesini bir miktar azaltmış, ancak çalışma hızını arttırmıştır. İkinci ve üçüncü iyileştirmeler basitleştirme kalitesini az da olsa arttırmış, aynı zamanda kullanılan işlemci zamanını azaltmıştır.

Orjinal ve iyileştirilmiş programlar değişik modeller için aynı parametre değerleri kullanılarak Sun Ultra Enterprise 4000 model bir bilgisayarda (8 adet 256 MHz işlemcili ve 1GByte sabit bellek kapasiteli) çalıştırılmış ve elde edilen sonuçlar Tablo 1 ve 2'de verilmiştir. Burada E_{dist} orjinal model üzerinde tanımlanan nokta kümesinin modele uzaklığını gösteren uzaklık enerji terimini ve E_{toplam} toplam enerji değerini ifade etmektedir. Bu istatistiklere göre basitleştirilmiş modellerin kalitesi (uzaklık enerji değerinin (E_{dist}) küçüklüğü bunun bir göstergesidir) orjinal program ile elde edilen basitleştirilmiş modellerle hemen hemen aynı olmasına karşın iyileştirilmiş program işlemci zamanları açısından orjinal programa göre yaklaşık iki kat daha hızlıdır. Orjinal ve iyileştirilmiş programlar ile elde edilmiş basitleştirme örnekleri Şekil 3'de verilmiştir[1].

Kaynakça

[1] P. Cignoni, C. Montani, and R. Scopigno, "A Comparison of Mesh Simplification Algorithms", *Computers & Graphics*, 22(1), s. 37-54, 1998.

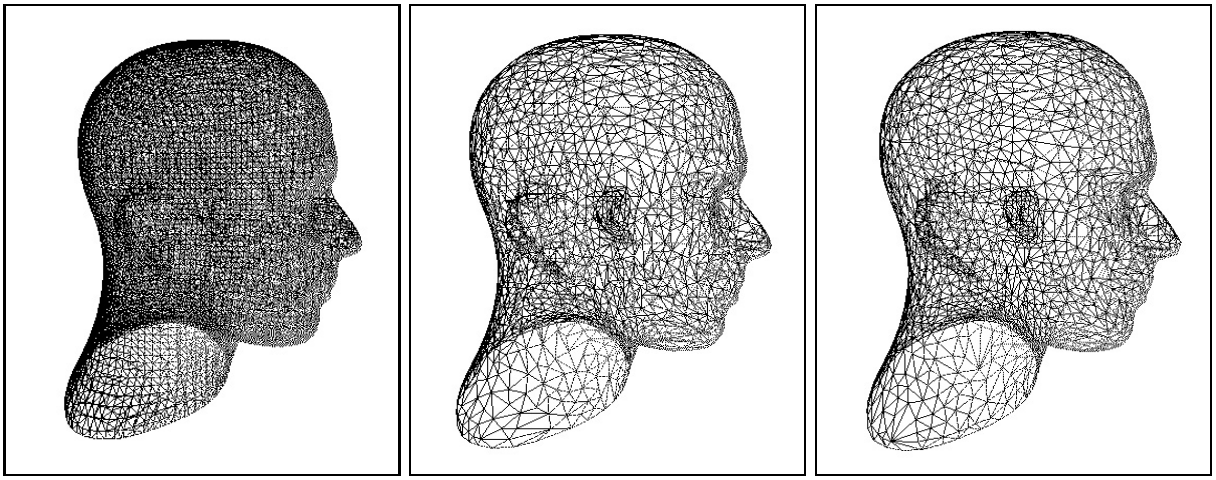
[2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Mesh Optimization", *Proc. of SIGGRAPH'93*, s. 19-26, 1993.

[3] Hoppe, H., "Progressive Meshes", *Proc. of SIGGRAPH'96*, s. 99-108, Ağustos 1996.

[4] P. Lindstrom and G. Turk, "Fast and Memory Efficient Polygonal Simplification", *Proc. of IEEE Visualization'98*, s. 279-286, Ekim 1998.

[1] Happy Budha modeli Stanford University, Computer Graphics Laboratuvarı'ndan sağlanmıştır. Diğer modeller ve orjinal basitleştirme yazılımı Hugues Hoppe'den temin edilmiştir

Mannequin Modeli

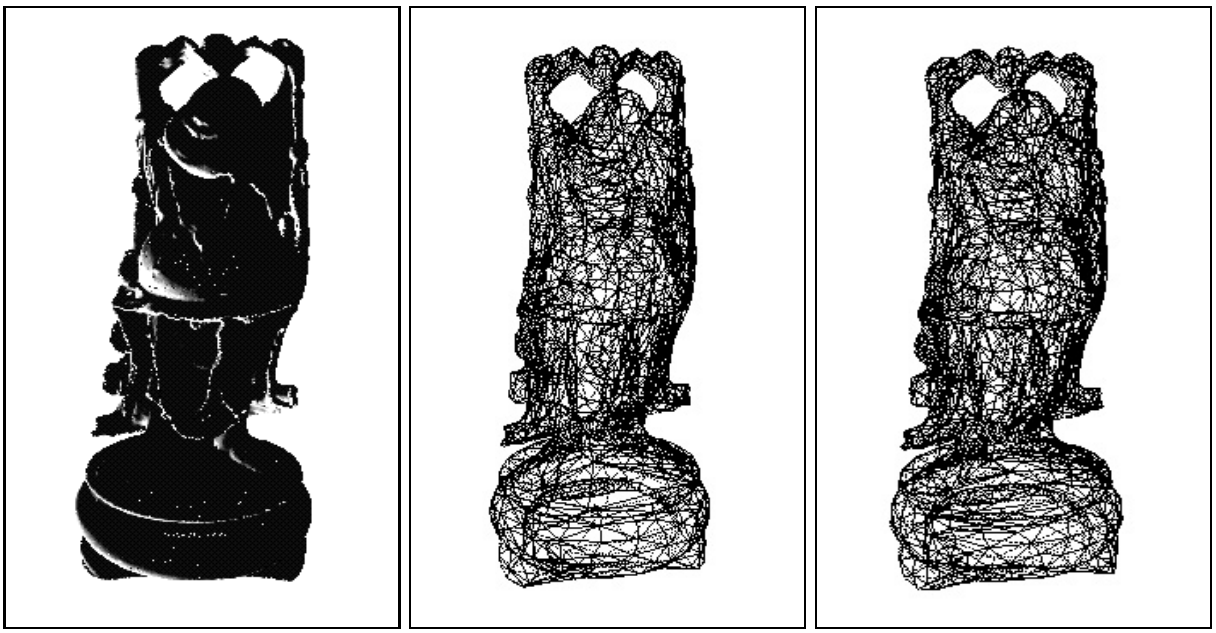


(a) Orjinal Model

(b) Basitleştirilmiş Model (Orjinal Program)

(c) Basitleştirilmiş Model (İyileştirilmiş Program)

Happy Budha Modeli



(a) Orjinal Model

(b) Basitleştirilmiş Model (Orjinal Program)

(c) Basitleştirilmiş Model (İyileştirilmiş Program)

Şekil 3: Basitleştirme Örnekleri