YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KATI CİSİMLERİN YAPICI KATI GEOMETRİSİNDE IŞIN İZLEME YÖNTEMİ KULLANILARAK MODELLENMESİ VE GÖRÜNTÜLENMESİ

Bilgisayar Müh. Mehmet ERSOYLU
F.B.E. Bilgisayar Bilimleri ve Mühendisliği Anabilim Dalında

hazırlanan

YUKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Tülay TİNLİ

İSTANBUL, 1997

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KATI CİSİMLERİN YAPICI KATI GEOMETRİSİNDE, IŞIN İZLEME YÖNTEMİ KULLANILARAK MODELLENMESİ VE GÖRÜNTÜLENMESİ

Bilgisayar Müh. Mehmet ERSOYLU

F.B.E. Bilgisayar Bilimleri ve Mühendisliği Anabilim Dalında
hazırlanan

YÜKSEK LİSANS TEZİ

Prof. Dr. Galip Cansever
12-11-1997

Tez Danışmanı : Yrd. Doç. Dr. Tülay TİNLİ

İSTANBUL, 1997

İÇİNDEKİLER

ŞEKİL LİSTESİ

ŞEKİL 1.1. IŞIN İZLEME AĞAÇ YAPISI 4

ŞEKİL 4.1. YANSIYAN IŞIN 13

ŞEKİL 4.2. KIRILAN IŞIN 16

ŞEKİL 5.1. KATI CİSMİN AĞAÇ YAPISINDA MODELİ 18

ŞEKİL 6.1. GORUNTU1.RT DOSYASININ GÖRÜNTÜSÜ 57

ŞEKİL 6.2. GORUNTU2.RT DOSYASININ GÖRÜNTÜSÜ 58

ŞEKİL 6.3. GORUNTU3.RT DOSYASININ GÖRÜNTÜSÜ 59

TABLO LİSTESİ

TABLO 5.1. - DOĞRULUK TABLOSU 21

TABLO 5.2. - KATI CİSMİN AĞAÇ YAPISININ AÇILIMI 25

TÜRKÇE ÖZET III

YABANCI ÖZET IV

BÖLÜM 1. IŞIN İZLEME YÖNTEMİ 1

BÖLÜM 2. IŞIN - CİSİM KESİŞİM YÖNTEMLERİ 5

2.1 IŞIN - KÜRE KESİŞİM YÖNTEMİ 5

2.2 IŞIN - DÜZLEM KESİŞİM YÖNTEMİ 8

2.3 IŞIN - POLİGON KESİŞİM YÖNTEMİ 8

BÖLÜM 3. YÜZEY FİZİĞİ 11

BÖLÜM 4. IŞIĞIN NAKİL MEKANİZMALARI 13

4.1 YOĞUN YANSIMA 13

4.2 HERYERDE AYNI YANSIMA 15

4.3 KIRILMA 15

BÖLÜM 5. YAPICI KATI GEOMETRİSİ 18

BÖLÜM 6. IŞIN İZLEME GENEL KOD YAPISI 28

SONUÇLAR 60

KAYNAKLAR 61

ÖZGEÇMİŞ

TÜRKÇE ÖZET

Bu çalışmanın amacı katı cisimlerin modellenmesi ve görüntülenmesidir. Karmaşık katı cisimler, ilkel cisimler dediğimiz (Küre, Silindir, Koni, Küp gibi) cisimlerin, bir ağaç yapısında, birtakım küme operatörleriyle (+, -, * gibi) birleştirilmesi sonucu modellenebilirler. Bu karmaşık cisimleri iki boyutlu ekranda görüntülemek için, hayali ışık ışınları gönderilir. Bu yönteme ışın izleme yöntemi denir. Bu yöntem, gerçeğe çok yakın üç boyutlu görüntüler elde etmemizi sağlar. Işın izleme yönteminde en fazla zaman kaybına neden olan durum doğru-cisim kesişim noktasını bulmaktır. Bu yüzden düzlem, ikinci dereceden yüzeyler gibi ilkel cisimler, kapsayıcı yüzeyler olarak kullanılabilirler. Doğrunun, kapsayıcı yüzeyi kesmediği durumlarda o yüzeyin içinde kalan cisimler doğrunun kesmediği cisimlerdir. Böylece gereksiz doğru-cisim kesim noktasını bulmaktan kaçınılmış olunur.

Yapıcı katı geometrisi kullanılarak kompleks cisimler bir ağaç yapısında modellenirler. Bu ağacın en alt seviyesini ilkel cisimler oluşturur. Bu yapıda, önce ışın izleme yöntemi ile gönderilen ışının ilkel cisimler ile olan kesim noktaları bulunur. Daha sonra, bulunan kesim noktaları, küme operatörlerine göre (+, -, *) birleştirilerek, ağacın bir üst seviyesindeki katı cismin, ışın ile olan kesim noktaları elde edilir. Tekrarlı bir yapıda bu işleme devam edilerek en üst seviyedeki istenen kompleks cismin ışın ile olan kesim noktası elde edilir. Bundan sonra, bulunan kesim noktasından yola çıkılarak, o noktadaki yüzeyin normali elde edilir. Yüzeyin normali ve ışık kaynağı arasındaki açının kosünüsü gerçek rengin oluşmasını sağlar.

YABANCI DİLDE ÖZET

The main objective of this project is the modelling and rendering of solid objects in computer environment. The solid objects are modeled as compositions of primitive solids (sphere, cone, cylinder, box), combined using boolean set operators union, intersection and difference. To visualize and analyze the composite solids modeled, virtual light rays are cast as probes. The procedure is called ray tracing or ray casting. This method provides us to obtain 3D photo realistic images of the solid objects.The most difficult and the most time-consuming part of this method is to find the line-surface intersection points. So surfaces such as planes, quadrics and even parametric surface patches may bound the primitive solids.

Using the constructive solid geometry method, solid objects are represented by a binary tree. The leaf nodes of the tree are primitives. The root node at the top represents the entire composition, but every node in the tree represents a complete solid. The ray casting algorithm composes of two important parts. First, intersect rays with primitives to find the enter-exit points, thereby providing ray-primitive classifications. Second, combine ray classifications according to the (+, -, *) operators. Raycast starts at the top of the solid composition tree, recursively descends to the bottom, classifies the ray with respect to the primitive solids, and then returns up the tree combining the classifications of the left and right subtrees. After finding the intersection point at the root, we find the surface normal at this intersection point. This surface normal helps us to find the color value of the intersection point. The cosine of the angle between the surface normal and light vector gives us the illumination at the intersection point.

BÖLÜM 1. IŞIN İZLEME YÖNTEMİ

Matematiksel denklemi belli olan üç boyutlu bir cismin, gerçeğe yakın görüntüsünü, bilgisayar ortamında elde etme yöntemlerinden biridir. Bu yöntemi daha iyi anlamak için öncelikle kameranın bir resmi nasıl çektiğini anlamak gerekir. Çünkü ışın izleme yöntemi kameranın yaptığı işin bir simülasyonudur.

En basit kamera modeli iğne delikli kamera modelidir. Bu modelde, bir fotoğraf filmi ışık geçirmez bir küpün arkasına yerleştirilir. Bir iğne ile bu küpün ön tarafı delinir. Daha sonra bu delinen kısım ışık geçirmez bir jelatin ile kapatılır. Resim çekilmek istendiğinde, kamera sabit tutulur ve bir an için ışık geçirmez jelatin çıkarılır. Böylece ışık delikten geçerek fotoğraf filmine çarpar. Filmi kısa bir süre için ışığa maruz bırakarak görüntünün film üzerinde oluşması sağlanır. İğne deliği burda önemli bir işlevde bulunmuştur. Eğer biz küpü ortadan kaldırıp, fotoğraf filmini ışığa maruz bıraksaydık, her yönden gelen ışık filmin tüm noktalarına çarpacaktı. Bu durumda film üzerinde beyaz bir görüntü elde edilecekti. İşte iğne deliği, sadece belli sayıda ışık ışınının film üzerine düşmesini sağlıyarak gerçek görüntünün oluşmasına sebeb olur. Bilgisayar ortamında bu modelde kullanılan kavramlar biraz değiştirilir. İğne deliği kavramının yerine gözün bulunduğu yer kavramı kullanılır. Fotoğraf filmi kavramının yerine görüntü düzlemi kavramı kullanılır. Bilgisayar ortamında bir cismin görüntüsü oluşturulurken aslında her bir pixel'e hangi rengin verileceği söz konusudur. İşte üç boyutlu bilgisayar grafiğinin çözmeye çalıştığı problemlerden en önemlisi budur. Burda sorulması gereken bir soru da şudur. Bir ışının rengi denince ne anlaşılıyor. Işık ışını, boşlukta giden bir ışık parçasının takip ettiği yol olarak düşünebiliriz. Bu ışık parçasına foton denir. Her fotonun bir enerjisi vardır. Gözümüze bir foton çarptığı zaman, bu enerji retinadaki alıcı hücrelere aktarılır. Bir fotondan bizim algıladığımız renk o fotonun enerjisi ile alakalıdır. Dolayısı ile farklı renklerdeki fotonlar farklı enerjilere sahiptir. Aslında bir fotonun enerjisi o fotonun dalgalanma enerjisidir. Fiziksel anlamda bir dalgalanma söz konusu olmasa dahi, dalgalanma olayı bir fotonun enerjisini matematiksel olarak ifade etmede oldukça kolaylık sağlar. Işığın dalga modelinde, farklı dalgalanma hızları farklı foton enerjilerine takabül eder. Bu yüzden her renk bir frekansa sahiptir.

Özetlersek, ışık kaynağından çıkan fotonlar görüntü uzayında değişik yollarda dolaşarak cisimlerin görüntülerinin oluşmasını sağlarlar. Genellikle, ışık, her yansımada veya kırılmada biraz daha enerji kaybeder. Sadece, görüntü uzayında dolaşıp en son gözün bulunduğu noktaya ulaşan fotonların görüntünün oluşmasına katkısı vardır. İşte bu örnekte anlatılan işleme ışın izleme yöntemi denir. Fakat bu yöntemde görüntünün oluşması bir hayli zaman alacaktır. Farzedelim, her bir ışık kaynağı, saniyede milyonlarca foton oluştursun. Bu fotonların çoğu, bizim görüş alanımızın dışında olan cisimlere vuracaktır. Sadece çok azı, görüş alanımız içindeki görüntünün oluşmasında katkıda bulunacaktır. Dolayısı ile bu yöntem zaman açısından oldukça maliyetli bir yöntemdir. Sonuç olarak fotonun başlangıç noktasını ışık kaynağı almak yerine gözün bulunduğu yeri fotonun başlangıç noktası olarak alırız. Böylece sadece görüş alanımıza girmiş olan fotonları kaale almış oluruz. Böyle bir fotonun takip ettiği yol, başlangıç noktası belli olan ancak bitiş noktası belli olmayan bir doğrudur, yani ışındır. Eğer bir foton gerçekten görüntünün oluşmasına katkıda bulunmuşsa, bu fotonun yolu, göz ile fotoğraf filmi üzerindeki noktayı birleştiren ışın üzerindedir. Bu foton, ışının yolu üzerinde bulunan göze en yakın cisimden geldi. Sonuç olarak ışının çarptığı ilk cisim, ışını yayan ilk cisimdir.

Bir ışının rengini oluşturan, o ışının yolu boyunca oluşan bütün fotonların renklerinin karışımıdır. Eğer kırmızı bir ışın ile yeşil bir ışın kendilerini aynı yol üzerinde bulurlarsa, ortaya sarı renkli bir ışın çıkar. Mantıksal olarak ışık ışınları dört ana sınıfa ayrılabilir. Bunlar; göz ışını, aydınlanma ışını, yansıyan ışın, kırılan ışın. Göz ışını, ekrandaki bir noktadan göze doğrudan ışığı taşıyan ışındır. Aydınlanma ışını, ışık kaynağından aldığı ışığı cisme taşıyan ışındır. Yansıyan ışın, cisim tarafından yansıtılan ışığı taşıyan ışındır. Kırılma ışını, cisim tarafından geçirilen ışığı taşıyan ışındır.

Bir cismin üzerindeki herhangibir noktadaki ışığın kaynağını aslında iki temel ışın oluşturmaktadır. Birisi cismin o noktasındaki aydınlanma, diğeri de o noktada, cismin yüzey karakteristiğine göre, meydana gelen yansıma ve/veya kırılma olayıdır. Farzedelim ki biz bir cismin üzerindeki bir noktadayız. Acaba bulunduğum bu noktaya ışık kaynaklarından herhangibir ışık geliyor mu? Bu sorunun cevabını bulmak için her bir ışık

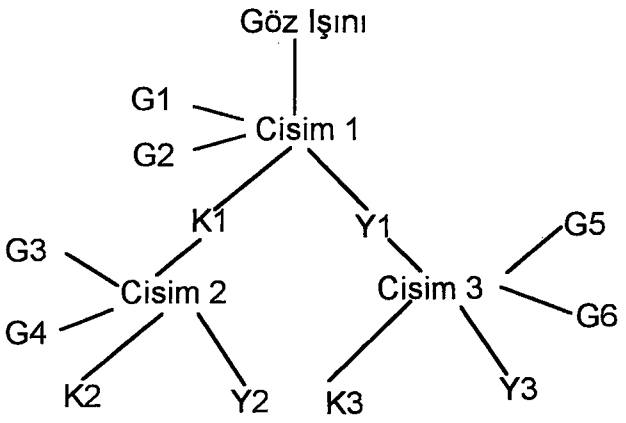
kaynağına bakarım. Eğer ışık kaynağını görebiliyorsam o zaman bulunduğum nokta ile ışık kaynağı arasında açık boş bir yol vardır. İşte bazı fotonlar bu açık yol üzerinden bulunduğum noktaya ulaşırlar. Eğer ışık kaynağını görmemi engelleyen herhangi başka bir cisim bu yol üzerinde varsa, o zaman bu ışık kaynağına göre bulunduğum nokta gölgededir. Sonuç olarak bir noktada ışığın oluşmasına katkıda bulunan ışınlardan ilki bu bahsedilen aydınlanma ışınlarıdır.

Düz parlak bir yüzeye baktığımızda, bu yüzeyde diğer cisimlerin yansımalarını görürüz. Bu yansımaları görmemizin sebebi, bu yüzeye diğer cisimlerden gelen ışınların yüzeyden yansıyarak görüş alanımızın içine girmiş olmasıdır.

Bir noktadaki yansıyan ışığı bulmak için önce o noktadaki yansıyan ışın bulunur. Yansıyan ışının rengini bulmak için bu ışının hangi cisimden kaynaklandığını bulmak gerekir. İşte bu cismi yansıyan ışın boyunca terkeden ışığın rengi yansıyan ışının rengidir.

O halde bir noktada oluşan ışığın kaynağı üç şekilde tecelli etmektedir. Birincisi, ışık kaynaklarından doğrudan gelen fotonların oluşturduğu aydınlanma. İkincisi, başka cisimlerden yansıma yolu ile gelen fotonların oluşturduğu aydınlanma. Üçüncüsü, başka cisimlerden kırılma yolu ile gelen fotonların oluşturduğu aydınlanma. Bir noktadaki yansıyan ve kırılan ışığın rengini bulurken, o noktaya bu ışınların hangi cisimlerden geldiğini bulmak gerekir. Peki bu önceki cisme bu yansıyan veya kırılan ışınlar nasıl gelmiştir sorusunun cevabını aynı analiz yöntemiyle bulabiliriz. Bu gözlem bize tekrarlı bir yapının varlığını gösterir.

Işın izleme yöntemi gözün bulunduğu yerden bir ışın göndererek başlar. Bir sonraki adımda bu ışının hangi cisme çarptığı bulunur.



ŞEKİL 1.1

Şekil 1.1 yukarıda anlatılan olayı şematik bir yapıda göstermektedir. Bu yapıya ışın ağaç yapısı denir. Bir ışının takibine aşağıdaki iki durumdan birisi ortaya çıktığında son verilir. Ya o ışın görüş alanımızın dışına çıkmıştır veya bir noktada rengin oluşmasına katkıda bulunması yok denecek kadar azalmıştır.

Işın ağacından aşağı doğru inildikçe, her yeni ışının; ki bunlar yansıyan ve kırılan ışınlardır, son görüntüye katkısı daha da azalır. Pratikte, bu işlemi durduracak bir eşik değeri verilir. Ağacın her seviyesinde, aşağıya doğru indikçe, ışının görüntüye katkı derecesi bu eşik değerinden az olursa bu işleme son verilir.

BÖLÜM 2. IŞIN-CİSİM KESİŞİM YÖNTEMLERİ

Işın izleme yönteminin ana yapısını ışın-cisim kesişim yöntemleri oluşturmaktadır. Işın-Cisim kesişim noktasını bulmak, ışın izleme yönteminin en çok zaman alan bölümüdür.

Bu projede ele alacağımız cisimler ikinci dereceden yüzeyleri kapsamaktadır. Bu cisimlere ilkel cisimler denir. Bunlar küre, koni , silindir, küp gibi cisimlerdir.

A) IŞIN-KÜRE KESİŞİM YÖNTEMİ

Küre, ışın izleme yönteminde en çok kullanılan basit ikinci dereceden cisimlerden biridir. Bir ışın ile bir kürenin kesim noktasının bulunması, diğer ikinci dereceden yüzeylere göre çok basit olmasından dolayı, küreler ışın izleme yöntemini hızlandırmak amacı ile kullanılan, çevreleyen hacimler olarak kullanılmaktadır.

Bir ışının parametrik denklemi aşağıdaki gibidir.

Işının başlangıç noktası : $R_0 = (X_0 Y_0 Z_0)$ (2.1)

Işının yön vektörü : $R_d = (X_d Y_d Z_d)$ (2.2)

ve $X_d^2 + Y_d^2 + Z_d^2 = 1$ (2.3)

olacak şekilde yön vektörü normalize edilir.

Sonuç olarak ışının parametrik denklemi aşağıdaki gibidir.

$$R(t) = R_0 + R_d * t, \qquad t > 0 \tag{2.4}$$

Bu doğru üzerinde (t<0) olan noktalar ışının başlangıç noktasının gerisinde kalacaktır. Denklem (2.4) bir ışının parametrik denklemidir. Böylece ışın üzerindeki

bütün noktalar (t) parametresini değiştirerek elde edilebilir. (t=0) anında başlangıç noktasında olan bir foton (t=1) anında bitiş noktasında olacaktır.

Küre denklemini de aşağıdaki şekilde tanımlayabiliriz.

Kürenin Merkezi : $S_c = (X_c Y_c Z_c)$ (2.5)

Kürenin Yarıçapı : S_r (2.6)

Sonuç olarak kürenin denklemi aşağıdaki gibidir.

$$(X-X_c)^2+(Y-Y_c)^2+(Z-Z_c)^2-S_r^2=0 \quad (2.7)$$

Yukardaki küre denklemi kapalı denklem olarak bilinir. Bu çeşit denklemlerle yüzeydeki noktalar doğrudan oluşturulamaz. Bu denklemle sadece verilen bir noktanın denklemi sağlayıp sağlamadığını kontrol edebiliriz. Eğer verilen nokta kapalı denklemi sağlıyorsa, bu nokta yüzeyin üzerindedir.

Bir ışının bir küre ile olan kesim noktasını bulmak için, ışın denklemi küre denkleminde yerine konulur ve (t) değeri bulunur.

$$(X_0+X_d*t-X_c)^2+(Y_0+Y_d*t-Y_c)^2+(Z_0+Z_d*t-Z_c)^2=S_r^2 \quad (2.8)$$

Denklemde sabit olan terimlere A,B ve C dersek denklem şu şekli alır.

$$\begin{aligned} &A*t^2+B*t+C=0 \\ &A=X_d^2+Y_d^2+Z_d^2=1 \\ &B=2*(X_d*(X_0-X_c)+Y_d*(Y_0-Y_c)+Z_d*(Z_0-Z_c)) \\ &C=(X_0-X_c)^2+(Y_0-Y_c)^2+(Z_0-Z_c)^2-S_r^2 \end{aligned} \quad (2.9)$$

A katsayısı ışının yön vektörünün boyudur. Bu değer normalize edildiği için değeri birdir (A=1). Bu ikinci derece denklemi (t) için çözecek olursak,

$$t_0 = \frac{-B - \sqrt{B^2 - 4 * C}}{2}$$
$$t_1 = \frac{-B + \sqrt{B^2 - 4 * C}}{2} \qquad (2.10)$$

Karekök içindeki değer negatif ise, ışın küreyi kesmez. En küçük pozitif reel kök, ışının cismi kestiği en yakın noktayı verecektir. (t) değeri bulunduktan sonra, yani, ışının cismi hangi zamanda kestiği bulunduktan sonra, gerçek kesim noktasını bulmak için (t) değeri ışın denkleminde yerine konur.

Kesim noktasındaki birim normal vektörü de aşağıdaki şekilde bulunur. Aslında, kesim noktasındaki gradient değeri bize o noktadaki normal vektörü verir.

$$r_n = (x_n y_n z_n)$$
$$x_n = \frac{(x_i - X_c)}{S_r}$$
$$y_n = \frac{(y_i - Y_c)}{S_r} \qquad (2.11)$$
$$z_n = \frac{(z_i - Z_c)}{S_r}$$

B) IŞIN-DÜZLEM KESİŞİM YÖNTEMİ

Düzlemin kapalı denklemi aşağıdaki gibidir.

$$A*x+B*y+C*z+D=0$$
$$A^2+B^2+C^2=1 \qquad (2.12)$$

D, koordinat sisteminin orijin noktasından düzleme olan uzaklıktır. Işının düzlemle olan kesim noktasını bulmak için ışın denklemi düzlem denkleminde yerine konur ve (t) değeri bulunur.

$$t=\frac{-(AX_0+BY_0+CZ_0+D)}{AX_d+BY_d+CZ_d} \qquad (2.13)$$

C) IŞIN-POLİGON KESİŞİM YÖNTEMİ

Işının düzlem ile olan kesim noktası bulunduktan sonra, bir sonraki adımda kesim noktasının poligonun içinde olup olmadığına bakılır. Işının poligon ile olan kesim noktası bulunurken aşağıdaki algoritma kullanılır.

Işının poligonu içeren düzlemi kesim noktasından herhangibir yönde bir ışın daha gönderilir. Ve bu ışının poligonu oluşturan doğru parçalarını kestiği sayı bulunur. Eğer ışın, doğru parçalarını tek sayıda kesiyorsa, nokta poligonun içindedir. Eğer çift sayıda noktada kesiyorsa nokta poligonun dışındadır.

Poligonun N tane noktadan oluştuğunu varsayalım. Bu noktalar tarafından tanımlanan düzlemin denklemi,

$$A*x+B*y+C*z+D=0$$
$$A^2+B^2+C^2=1 \qquad (2.14)$$

Farzedelim bize kesim noktası olarak $R_i = (X_i Y_i Z_i)$ noktası verilsin. Biz bu noktanın verilen poligonun içinde olup olmadığını kontrol edeceğiz.

İlk adım olarak poligonun iki boyutlu düzleme izdüşümü alınır. Bu düzlemde bütün (X,Y,Z) noktaları (U,V) noktalarına dönüştürülür. Düzlemin normal vektörünü oluşturan (X,Y,Z) koordinatının en büyüğü atılarak bu dönüşüm sağlanır. Poligon iki boyutlu düzleme indirgendikten sonra, kesim noktası orijinde olacak şekilde poligonun noktaları yer değiştirilir. Bu yerleri değiştirilen yeni poligon noktalarına (U',V') diyelim. Şimdi kesim noktasını kontrol etmek istediğimiz noktadan U' ekseni boyunca bir ışın gönderelim. Poligonun her köşegeninin ışın ile olan kesim durumu incelenir. Eğer ışın köşegeni kesiyorsa daha önceden sıfırladığımız bir sayaç bir arttırılır. Işının poligonun bütün köşegenleri ile olan kesim durumu incelendikten sonra sayacın değeri tek ise, incelenen kesim noktası poligonun içindedir. Aksi takdirde dışındadır.

Sonuç olarak yukarıda anlatılan yöntemin algoritması aşağıda verilmiştir.

1) FOR n=0 TO (NV-1)

1.1) $(U_n,V_n) = PROJECT(X_n,Y_n,Z_n)$

1.2) $(U'_n,V'_n) = TRANSLATE(U_n,V_n)$

/* Kesim noktası orijinde olacak şekilde poligonun köşe noktaları yer değiştirilir. */

3) NC = 0 /* Kesim noktası sayısını tutan sayaç değeridir. */

4) SH = 1

5) $IF(V'_0 < 0)$

4.6) SH = -1

7) FOR a=0 TO (NV-1)

5.8) b = (a+1) MOD NV

5.9) NSH = 1

5.10) $IF(V_b^{'} < 0)$

5.3.11) NSH = -1

5.12) IF (SH <> NSH)

5.4.13) $IF(U_a^{'} > 0 AND U_b^{'} > 0)$

5.4.1.14) NC = NC + 1

5.4.15) $IF(U_a^{'} > 0 OR U_b^{'} > 0)$

5.4.2.16) $IF(U_a^{'} - V_a^{'} * \frac{(U_b^{'} - U_a^{'})}{(V_b^{'} - V_a^{'})} > 0$

5.4.2.1.17) NC = NC + 1

5.18) SH = NSH

19) IF (NC MOD 2 <> 0)

6.20) Verilen kesim noktası Poligonun içindedir.

21) ELSE

7.22) Verilen kesim noktası Poligonun dışındadır.

BÖLÜM 3. YÜZEY FİZİĞİ

Bir cismin görüntüsünü tarif ederken, genellikle o cismin renginden bahsederiz. Işığın yapısını şu ana kadar kimse tam olarak anlamış değildir. Işığın pek çok özelliğini tarif eden iki tane model mevcuttur. Fakat bunların hiçbiri tam olarak doğru değildir. Bu modellerden birisi ışığın dalga modeli, diğeri de ışığın parçacık modelidir. Işığın dalga modeli ışığın hareketini bir su dalgasına benzetir. Işığın parçacık modeli de ışığın pek çok küçük parçacıktan oluştuğunu varsayar.

Işın izleme yöntemi daha çok ışığın parçacık modelini baz alır. Işık bazı durumlarda dalga bazı durumlarda ise parçacık gibi davranır. Burada ışığın parçacık modeli üzerinde durulacaktır.

Işığın en küçük parçacığına foton denir. Fotunu, uzayda hareket eden bir bilardo topuna benzetebiliriz. Ancak fotonun bu hareketi doğrusal değildir. Fotunun dalgalı, bir sinüs eğrisine benzeyen hareketi mevcuttur. Dolayısı ile her fotonun bir frekansı vardır.

Görecelik teorisine göre, ışığın hızı her ortamda sabittir. Bu gözlemi aşağıdaki şu denklem ile özetleyebiliriz.

$$\lambda = \frac{c}{f} \tag{3.1}$$

Burda λ dalga boyunu, C ışık hızını, F de frekansı vermektedir. Bir fotonun enerjisi o fotonun frekansına bağlıdır.

$$E = h * f \tag{3.2}$$

Burda E fotonun enerjisini, h ise planck sabitini göstermektedir. Bir fotonun enerjisi ve frekansı ile o fotonun rengi arasında doğrudan bir ilişki söz konusudur. İşte fotonların farklı frekanslara sahip olması farklı renkleri algılamamızı sağlar.

Gerçeğe yakın görüntüler elde etmek için öncelikle ışığın cisimlerin yüzeylerinde nasıl davrandıklarını anlamamız gereklidir. Bir foton bir yüzeye çarptığı zaman fotonun yönünde ve renginde değişiklikler olur. Bu değişme, yüzeyin karakteristik özelliklerine bağlıdır. Beyaz ışık saçan bir ampüle baktığımızda beyaz ışığı görürüz. Aslında gökkuşağına bakarsak beyaz ışık diye bir şey yoktur. Pek çok farklı frekansta foton kendilerini aynı yol üzerinde bulduklarında ortaya beyaz ışık çıkar.

Dalgasal bir hareketi olan fotonun bir cismin yüzeyine çarptığını farzedelim. Bu cismin atomları çeşitli frekanslarda hareket etmektedir. Bir foton bir atoma çarptığında, enerjisinin tamamını veya bir kısmını atoma transfer etme şansına sahiptir. Eğer foton, çarptığı atomu bir üst enerji seviyesine çıkartacak kadar bir enerjiye sahipse, foton emilir ve atom bir üst enerji seviyesine geçer. Eğer bu transfer için fotonun yeterli enerjisi yoksa, atom fotonun enerjisinin bir kısmını o an için emer ve hemen bu enerjiyi ısı şeklinde yayarak kaybeder. Mavi renkli bir kitabı düşünelim. Mavi renkli olmayan fotonlar emilir ve ısıya dönüştürülür. Mavi fotonlar ise cisim tarafından emilir ve tekrar yayınlanır. Aslında bu olayla mavi fotonlar cismin yüzeyinden yansımıştır.

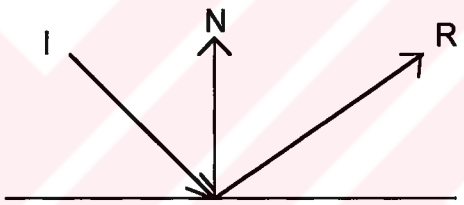
Cisimlerin yüzeylerinde ışığın nasıl davrandığını anlamak için, o yüzeyin geometrik özelliklerinden biri olan yüzeyin normalini bilmek gerekir. Yüzeyin normali bir noktada yüzeye dik olan vektördür. Mesela, bir düzlemin normal vektörü her yerde aynıdır. Veya bir kürenin bir noktadaki normal vektörü, o noktayı merkeze birleştiren vektördür.

BÖLÜM 4. IŞIĞIN NAKİL MEKANİZMALARI

Bir ışığın yüzeyle olan etkileşimi dört şekilde olur.

A) YOĞUN YANSIMA

Farzedelim ki bir basketbol sahasındayız. Topu takım arkadaşımıza atmak istediğimizde arkadaşımla benim aramdaki mesafenin orta noktasına gelecek şekilde topu atarız. İşte bir ışığın bir yüzeyden yansıma olayı da aynı bunun gibidir. Mesela, aynalarda gördüğümüz görüntüler, gelen ışınların yansımasından başka bir şey değildir. Yine, metalik yüzeylerin belli bölümlerinde ışığın parlaması da aynı yansımadan kaynaklanmaktadır.



ŞEKİL 4.1

Şekil 4.1, gelen ışığın (I), bir yüzeyden yansımasını göstermektedir. Burda gelen ışınla yüzeyin normali arasındaki açıya geliş açısı denir. Bunu, Φ_i ile gösterelim. Yüzeyin normali ile yansıyan ışın arasındaki açıya yansıma açısı denir. Bunu da Φ_r ile gösterelim. Bize verilen N ve I değerlerine göre R vektörünü şu şekilde bulabiliriz.

İki fizik kuralı R değerini bulmamızda yardımcı olacaktır. Birincisi, gelen ışın, yüzeyin normali ve yansıyan ışın aynı düzlem üzerindedir. Dolayısı ile yansıyan ışın, gelen ışın ile yüzeyin normal vektörünün lineer bir kombinasyonudur. İkincisi ise, geliş açısı yansıma

açısına eşittir. Bu iki fizik kuralını matematiksel olarak aşağıdaki şekilde ifade edebiliriz.

$$\begin{aligned} R &= \alpha I + \beta N \\ \Phi_I &= \Phi_R \end{aligned} \tag{4.1}$$

Şekil 4.1'e bakacak olursak, aşağıdaki sonuçları çıkartırız.

$$\begin{aligned} \cos(\Phi_I) &= -I \bullet N \\ \cos(\Phi_R) &= N \bullet R \end{aligned} \tag{4.2}$$

Denklem 4.1'in ikinci kısmını aşağıdaki şekilde tekrar yazabiliriz.

$$\begin{aligned} \cos(\Phi_I) &= \cos(\Phi_R) \\ -I \bullet N &= N \bullet R \\ -I \bullet N &= N \bullet (\alpha I + \beta N) \\ -I \bullet N &= \alpha (N \bullet I) + \beta (N \bullet N) \\ -I \bullet N &= \alpha (N \bullet I) + \beta \end{aligned} \tag{4.3}$$

Rastgele alfa değerini bir (1) seçersek o zaman,

$$\beta = -2(N \bullet I) \tag{4.4}$$

Dolayası ile yansıyan ışığın yönünü veren denklem aşağıdadır.

$$R = I - 2(N \bullet I)N \tag{4.5}$$

Burda I gelen ışığı, N yüzeyin normalini, R de yansıyan ışığı göstermektedir.

B) HERYERDE AYNI YANSIMA

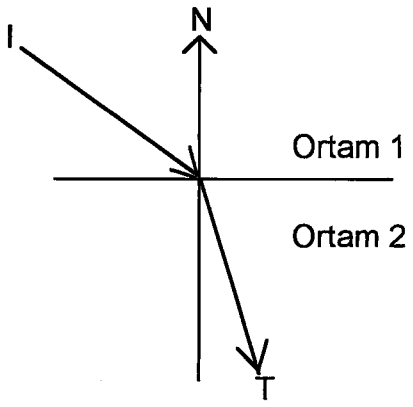
Yoğun yansıma olayı parlak yüzeyler veya yansıtıcı özelliğe sahip yüzeyler için oluşan bir olaydı. Pürüzlü yüzeylerde ışığın davranışı ise daha farklıdır. Yansıyan ışığın pürüzlü yüzeylerde yukardaki gibi basit geometrik bir yapısı yoktur.

Işığın yüzey tarafından emilmesi ve tekrar yayılması olayını hatırlarsak, yoğun yansıyan ışık bu olaya daha az maruz kalır. Fakat difüzyon yolu ile yansıyan ışığın, yüzeyle olan irtibatı daha yoğundur. Bir foton, yüzeydeki atom tarafından emildiğinde, ya ısıya dönüştürülür veyahut da tekrar yayınlanır. Eğer foton tekrar yayınlanırsa, fotonun hangi yönde gideceği meçhuldür. Aslında bir foton tek bir yönde gitse dahi, birden fazla foton, zaman içersinde muhtemel yönlerde gitme eğilimindedir. Netice itibari ile, difüzyon yolu ile yüzeyden yansıyan ışık, aynı yoğunlukta bütün yönlerde yansır. Şu halde yüzeyin ne kadarlık kısmının ışık kaynağına gözüktüğü bizi ilgilendiren kısımdır. Bunu da gelen ışın ile yüzeyin normali arasındaki açıdan bulabiliriz. Yüzeye düşen ışık miktarı bu açının kosünüsü ile orantılıdır.

C) KIRILMA

Işığı geçiren cisimlerde, cismin arkasından gelen ışık, cismin içinden geçer ve öteki tarafdan cismi terkeder. Bu tür ışınlara kırılma ışınları denir.

Cismin her iki tarafındaki ortam farklı olabilir. Işık farklı ortamlardan geçerken farklı kırılımlara uğrar. Bu yüzden içi su dolu bir kovaya bir cetvel batırdığımızda, cetvelin havada kalan kısmı ile sudaki kısmı arasında bir eğiklik görürüz. Her ortamın bir kırılma katsayısı vardır. Buda ışığın boşluktaki hızına göre bu ortamdaki hızını belirler.



ŞEKİL 4.2

Geliş açısı Φ_1, kırılma açısı Φ_2 olsun. Gelen ışın, yüzeyin normali ve kırılan ışın aynı düzlem üzerindedir. SNELL kuralına gore,

$$\frac{\sin(\Phi_1)}{\sin(\Phi_2)} = \eta_{21} = \frac{\eta_1}{\eta_2} \tag{4.6}$$

η_1, birinci ortamın boşluğa göre kırılma katsayısı, η_2, ikinci ortamın boşluğa göre kırılma katsayısı, η_{21}, ikinci ortamın birinci ortama göre kırılma katsayısını göstermektedir.

Burdan da anlaşılacağı gibi, kırılma katsayısı, gelen ışığın dalga boyuna bağlıdır. Bir prizma üzerine düşen ışığın renklere ayrılması, farklı dalga boylarının farklı miktarlarda kırılmasından kaynaklanmaktadır.

Işık, çok yoğun bir ortamdan az yoğun bir ortama dar bir açı ile geçmek istediğinde bir iç yansıma söz konusu olur. Bu durumda ışık kırılacağına yansımaya uğrar ve yüzeyin içine doğru yansır. Bu olay, fiber optiklerin çalışma mekanizmalarını oluşturmaktadır. Işık bir alanın dışına çıkmaya çalıştığında, bu olay gerçekleştirilerek o alandan dışarı çıkması engellenir. Böylece, ışık bir uçtan diğer bir uca taşınmış olur.

Işık, kritik bir açıdan sonra, çok yoğun bir ortamdan az yoğun bir ortama geçmeye çalıştığında, bu olay vuku bulur. Bu kritik açıdan daha

küçük açılarda, ışık hem kırılır hem de yansır. Fakat kritik açıdan büyük eşit açılarda sadece yansıma olur. Bu olaya iç yansıma denir.

Öncelikle, kırılan ışığın yönünü veren denklemi elde etmeye çalışalım.

İki fizik kuralı ve SNELL kuralından yola çıkarak sonuçta aşağıdaki formülü elde ediriz.

$$C_I = \cos(\Phi_1) = (N \bullet -I)$$
$$T = \eta_{IT} I + (\eta_{IT} C_I - \sqrt{1 + \eta_{IT}^{\ 2}(C_I^{\ 2} - 1)})N \qquad (4.7)$$

Eğer karekök içindeki ifade negatif ise, bu durumda iç yansıma söz konusudur ve kırılma olmaz.

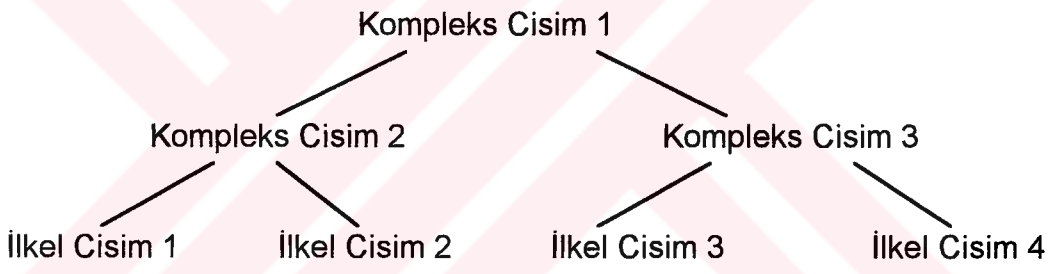
BÖLÜM 5. YAPICI KATI GEOMETRİSİ

Karmaşık katı cisimler, küre, silindir, koni, dikdörtgenler prizması gibi ikinci dereceden ilkel cisimlerin bir ağaç yapısında, bileşim, kesişim ve fark küme operatörleri ile birleştirilmesi neticesinde elde edilebilirler. Yapıcı katı geometrisi kullanılarak, kompleks katı cisimler ilkel cisimlerin birleştirilmesi ile modellenebilir.

Kompleks katı bir cisim, bir ağaç yapısında kökteki şekli temsil etmektedir. Bu ağacın yapraktaki düğümleri yukarda bahsedilen ilkel cisimleri oluşturmaktadır.

Yapıcı katı geometrisinde yine ışın izleme yönteminden faydalanılacaktır. Ancak burada, ışının ilk kestiği nokta değil, ışının bir cisme girdiği ve çıktığı noktaların hepsi bir dizi yapısında tutulur.

Aşağıda muhtemel bir katı cismin ağaç yapısı örnek teşkil etmesi bakımından gösterilmiştir.



ŞEKİL 5.1

Böyle bir yapıda N tane ilkel cisim varsa, o zaman (N-1) tane katı cisim mevcuttur. Katı cisim ve ilkel cisim için asgari bilgi yapısı aşağıda gösterilmiştir.

KATI CİSİM :

İSMİ

YAPILAN İŞLEM

L_SOLID_PTR

R_SOLID_PTR

İLKEL CİSİM :

İSMİ

TİPİ (Silindir, Küre, Koni, Düzlem gibi)

YÜZEY_PTR (Tip alanı ile belirtilen ilkel cismin yüzey özelliklerine işaret eder)

İsim alanı kullanıcının oluşturmak istediği katı cismi belirtirken her bir ara cisme verdiği bir isimdir.

Bize bir ışın ve katı cismin ağaç yapısı verildiğinde, yapılacak iş, ışını ağaç yapısındaki bütün cisimlere göre sınıflandırmaktır. Bu sınıflandırma, ışının hangi kısmının cismin içinde olduğunun ve hangi kısmının cismin dışında olduğunun tesbiti şeklindedir. Ağacın en tepesinden başlayarak, tekrarlı bir yapıda en alt seviyeye kadar inerek, burda ilkel cisimlerden elde edilecek olan ışın-cisim kesişim kümeleri, iki ilkel cisim arasında istenilecek olan küme operasyonuna (Bileşim, Kesişim, Fark) göre birleştirilerek ağacın bir üst seviyesine döndürülür. Bu işlemi tekrarlı bir yapıda ağacın diğer seviyelerindeki cisimler için de uygularsak, en üst seviyedeki kompleks cisim ile ışının kesim kümesini elde etmiş oluruz.

Aşağıda böyle tekrarlı bir yapının akış diyagramı gösterilmektedir. Bu yapıda RAYCAST modülü, verilen bir ışının, ağacın kökünde oluşacak katı kompleks cisme göre ışın-cisim kesişim kümesini oluşturur.

RAYCAST : PROCEDURE (SOLID_PTR) : RETURNS (Classification)

```
IF (SOLID_PTR.CLASS = "KATI CİSİM") THEN
BEGIN
        L_Classification = RAYCAST ( SOLID_PTR.L_SOLID_PTR )
        R_Classification = RAYCAST ( SOLID_PTR.R_SOLID_PTR )
        RETURN ( COMBINE ( L_Classification, R_Classification ) )
END

ELSE
        RETURN ( Işının ilkel cisim ile olan kesişim kümesi )
```

RAYCAST modülüne parametre olarak ışın denklemi ve kompleks katı cismi temsil eden ağaç yapısına işaret eden bir yapı verilir. Modülden çıkış olarak ışının kompleks katı cisim ile olan kesişim kümesi döndürülür.

Sol ve sağdaki ilkel cisimlerin ışın ile olan kesişim kümeleri istenen operasyona (Bileşim, Kesişim, Fark) göre aşağıdaki akış diyagramları ışığında birleştirilir. Bunun için soldaki ilkel cismin ışın ile olan kesişim kümesini L_LIST, sağdaki ilkel cismin ışın ile olan kesişim kümesini R_LIST ile gösterelim. Burda ayrıca kesişim kümesindeki noktaların cismin içinde mi dışında mı olup olmadığı bilgisi IO_LIST isimli bir dizide tutulur. L_LIST ve R_LIST dizileri istenen operatöre göre birleştirilir. Yeni oluşan kesim noktalarının cismin içinde olması için IO_LIST dizisinde ilgili kesim noktası aralığının 1 olması gereklidir. Bu şekilde tekrarlı bir yapıda ağaç taranarak, en üst seviyede kompleks cisme ait kesişim kümesi bulunur. Gerçek kesim noktası, en üst seviyede IO_LIST dizisinin ilk 1 değerine tekabül eden indise ait T_LIST dizisindeki kesim noktası aralığıdır. Bu aralığın en küçük değeri bize göze en yakın kesim noktasını vermektedir. T_LIST dizisi, L_LIST ve R_LIST dizilerinin istenen operatöre göre birleştirilmesi ile oluşturulur.

İki ilkel cismin belirtilen operasyona göre kesişim kümeleri birleştirilirken boolean cebirden faydalanılır. Bunun için aşağıdaki tablodan faydalanılır.

Operator	Sol	Sağ	Katı Cisim
+	IN	IN	IN
+	IN	OUT	IN
+	OUT	IN	IN
+	OUT	OUT	OUT
*	IN	IN	IN
*	IN	OUT	OUT
*	OUT	IN	OUT
*	OUT	OUT	OUT
-	IN	IN	OUT
-	IN	OUT	IN
-	OUT	IN	OUT
-	OUT	OUT	OUT

TABLO 5.1

Aşağıda RAYCAST fonksiyonunda geçen COMBINE fonksiyonuna ait akış diyagramı mevcuttur.

COMBINE (L_LIST, R_LIST, L_IO_LIST, R_IO_LIST, L_THR_SRF, R_THR_SRF, POINTER->OPRTR)

/*** L_LIST ve R_LIST ile verilen kesişim kümelerini POINTER->OPRTR operatörüne göre birleştiren ve yeni kesişim kümesini T_LIST'e atan ve T_LIST'teki noktaların cismin içinde mi dışında mı olup olmadığını gösteren IO_LIST'i oluşturan bir fonksiyondur. ***/

1) I = 0
2) WHILE (L_LIST(I)) <> -1
   2.1) I = I + 1

```
3)  N = I
4)  I = 0
5)  WHILE ( R_LIST(I) ) <> -1
        5.6) I = I +1
7)  M = I
8)  I = 0
9)  J = 0
10) K = -1
11)  WHILE (I<N AND J<M)
        10.12) K = K + 1
        10.13)  IF ( L_LIST(I) < R_LIST(J) )
                10.2.14) T_LIST(K) = L_LIST(I)
                10.2.15)  THR_SRF(K) L_THR_SRF(I)
                10.2.16)  I = I + 1
        10.17) ELSE
                10.3.18) T_LIST(K) = R_LIST(J)
                10.3.19)  THR_SRF(K) = R_THR_SRF(J)
                10.3.20)  J = J + 1
21) IF ( I = N )
        11.22) FOR L = J TO (M-1)
                11.1.23) K = K + 1
                11.1.24)  T_LIST(K) = R_LIST(L)
                11.1.25)  THR_SRF(K) = R_THR_SRF(L)
        11.26) FOR L = I TO (N-1)
                11.2.27) K = K + 1
                11.2.28)  T_LIST(K) = L_LIST(L)
                11.2.29)  THR_SRF(K) = L_THR_SRF(L)
30) J = 1
31) B_LEFT(0..(N+M)) = 0
```

```
14) IF (POINTER->OPRTR = "-")
        14.15) B_RIGHT(0..(N+M)) = 1
16) ELSE
        15.17) B_RIGHT(0..(N+M)) = 0
18) I = 0
19) WHILE (I<N)
        17.20) WHILE( (L_IO_LIST(I) = 0 AND (I<N) )
                17.1.21) I = I + 1
        17.22) IF (I=N) BREAK
        17.23) X = L_LIST(I-1)
        17.24) Y = L_LIST(I)
        17.25) J = 0
        17.26) WHILE ( T_LIST(J) <> X )
                17.6.27) J = J + 1
        17.28) J = J + 1
        17.29) BAS = J
        17.30) WHILE ( T_LIST(J) <> Y )
                17.9.31) J = J + 1
        17.32) WHILE (BAS<=J)
                17.10.33) B_LEFT(BAS) = 1
                17.10.34) BAS = BAS + 1
        17.35) I = I + 1
36) I = 0
37) WHILE (I<M)
        19.1) WHILE( (R_IO_LIST(I) = 0 AND (I<M) )
                19.1.1) I = I + 1
        19.2) IF (I=M) BREAK
        19.3) X = R_LIST(I-1)
```

```
19.4) Y = R_LIST(I)
19.5) J = 0
19.6) WHILE ( T_LIST(J) <> X )
            19.6.7) J = J + 1
19.8 ) J = J + 1
19.9 BAS = J
19.10 WHILE ( T_LIST(J) <> Y )
        19.9.11) J = J + 1
19.12) WHILE (BAS<=J)
        19.10.1) IF POINTER->OPRTR = "-"
                19.10.1.1) B_LEFT(BAS) = 0
        19.10.2) ELSE
                19.10.2.1) B_LEFT(BAS) = 1
        19.10.3) BAS = BAS + 1
    19.11) I = I + 1
20) FOR I = 0 TO (N+M)
    20.21) IF ( POINTER->OPRTR = "-" OR PONITER->OPRTR = "*" )
        20.1.22) IO_LIST(I) = B_LEFT(I) * B_RIGHT(I)
    20.23) ELSE
        20.2.1) IO_LIST(I) = 0
        20.2.2) IF ( B_LEFT(I)=1 OR B_RIGHT(I)=1 )
                20.2.2.1) IO_LIST(I) = 1
```

Buraya kadar olan kısımda CSG için gerekli olan temel yapı taşları üzerinde duruldu. Genel olarak, kullanıcı, oluşturacağı modeli, bir dosya yapısında aşağıdaki şekilde girer. Bu yapıdan faydalanılarak bir ağaç yapısı oluşturulur.

/*** P : İlkel cisim olduğunu gösterir. ***/

/*** C : Kompleks cisim olduğunu gösterir. ***/

İsim	Tip	İsim	Tip	Operatör	Tip
P1	P	P2	P	+	C1
P3	P	P4	P	*	C2
P5	P	P7	P	*	C3
P7	P	P8	P	-	C4
C1	C	C2	C	-	C5
C3	C	C4	C	+	C6
C5	C	C6	C	*	C7

TABLO 5.2

Yukarıdaki yapıdan faydalanılarak bir ağaç yapısı oluşturulur. Bu ağaç yapısında her bir düğüm bir cisme karşılık gelmektedir. En üst seviyedeki kompleks cismi oluşturmak için, tekrarlı bir yapıda ışın izleme yöntemi ağacın solundaki ve sağındaki cisimler için uygulanır. Ancak burda ışının her bir cismi kestiği tüm noktalar bir dizi yapısında tutulur. İstenen operatöre göre (+, -, *), bu iki cisimden elde edilen kesim noktalarına işaret eden parametreler birleştirilir. Elde edilen yeni parametre listesi, yeni oluşan cismin gönderilen ışın ile olan kesim noktalarını verir. Tekrarlı yapıda bu işleme devam edilerek en üst seviyedeki kompleks cismin kesim noktalarını veren dizi elde edilir. Zaten sıralı olan bu kesim noktalarından en küçüğü göze en yakın olan cismi bize verecektir. Bu nokta da dizinin en küçük indisine tekabül etmektedir.

Aşağıda Tablo 5.2'de verilen yapıdan yola çıkarak ağaç yapısının nasıl oluşturulacağına ait akış diyagramı verilmiştir.

1) I = -1

2) L = -1

3) WHILE NOT EOF

3.4) READ (NAME1, CLASS1, NAME2, CLASS2, OPRTR, NAME3)

3.5) IF (CLASS1 = "P" AND CLASS2 = "P")

3.2.6) NEW (POINTERLEFT)

3.2.7) POINTERLEFT->NAME = NAME1

3.2.8) POINTERRIGHT->CLASS = CLASS1

3.2.9) NEW(POINTERRIGHT)

3.2.10) POINTERRIGHT->NAME = NAME2

3.2.11) POINTERRIGHT->CLASS = CLASS2

3.2.12) NEW(POINTER)

3.2.13) POINTER->LEFT = POINTERLEFT

3.2.14) POINTER->RIGHT = POINTERRIGHT

3.2.15) POINTER->CLASS = "C"

3.2.16) I = I + 1

3.2.17) POINTERARRAY (I) = POINTER

3.2.18) POINTERNAME (I) = NAME3

3.19) ELSE IF (CLASS1 = "P")

3.3.20) NEW (POINTERLEFT)

3.3.21) POINTERLEFT->NAME = NAME1

3.3.22) POINTERLEFT->CLASS = CLASS1

3.3.23) L = 0

3.3.24) WHILE (NAME2<>POINTERNAME(L))

3.3.5.25) L = L + 1

3.3.26) POINTERRIGHT = POINTERARRAY(L)

3.27) ELSE IF (CLASS2 = "P")

3.4.28) NEW(POINTERRIGHT)

3.4.29) POINTERRIGHT->NAME = NAME2

3.4.30) POINTERRIGHT->CLASS = CLASS2

3.4.31) L = 0

3.4.32) WHILE (NAME1<>POINTERNAME(L))

3.4.5.33) L = L + 1

3.4.34) POINTERLEFT = POINTERARRAY(L)

3.35) ELSE

3.5.36) L = 0

3.5.37) WHILE (NAME1, POINTERNAME(L))

3.5.2.38) L = L + 1

3.5.39) POINTERLEFT = POINTERARRAY(L)

3.5.40) L = 0

3.5.41) WHILE (NAME2, POINTERNAME(L))

3.5.5.42) L = L + 1

3.5.43) POINTERRIGHT = POINTERARRAY(L)

3.44) NEW(POINTER)

3.45) POINTER.LEFT = POINTERLEFT

3.46) POINTER.RIGHT = POINTERRIGHT

3.47) I = I + 1

3.48) POINTERARRAY(I) = POINTER

3.49) POINTERNAME(I) = NAME3

BÖLÜM 6. IŞIN İZLEME GENEL KOD YAPISI

Aşağıda ışın izleme yönteminin akış diyagramı verilmiştir. Bu akışa yukardaki CSG modüllerini de eklersek bahsedilen kompleks cisimleri modelleme ve görüntüleme işlemini tamamlamış oluruz.

ANA BÖLÜM

/**** Ana Bölümün Başlangıcı ****/

1) Cisimlerin bulunduğu ortamı ve ortamdaki cisimleri tanımlayan ekran dosyası okunur ve bilgiler ilgili cisimlerin bilgi yapılarına yerleştirilir.

2) Ekran modu, ekran dosyasındaki bilgiler ışığında ayarlanır.

3) FOR Y=0 TO MaxYÇözünürlük

FOR X=0 TO MaxXÇözünürlük

3.1) Gözün bulunduğu yerden şu andaki pixel lokasyonu olan (X,Y,0)'a bir ışın gönder ve bu ışının yön vektörünü INTIALDIR değişkenine ata.

3.2) TRACERAY (ObsPos, InitialDir, MaxWgt, 1, Col)

/* ObsPos: Gözün bulunduğu yer.
/* MaxWgt: Değeri birdir. Ancak, her yansıyan veya kırılan ışın gönderildiğinde yansıma veya kırılma kaybı kadar azalır.
/* Col: Dönen renk değeri.

3.3) PLOT(X,Y,Col)

/**** Ana Bölümün Bitişi ****/

TRACERAY (Start, Dir, TotWgt, Depth, Col)

/**** TraceRay Modülünün Başlangıcı ****/

1) ObjHit = SHOOTRAY (Start, Dir, &TR_Shp, &TR_Obj, &Dist)

/* Gönderilen ışının uzayda tanımlı herhangibir cismi kesip kesmediğini ve eğer kesmişse hangi cisim olduğunu ve kesim noktasını parametrik olarak bulur. Hangi cismin en yakın cisim olduğu bilgisi TR_Shp ve TR_Obj değişkenlerine döndürülür. Kesim noktasının parametrik değeri DIST değişkenine döndürülür.

2) IF (ObjHit = TRUE)

2.1) GETINTRPT (Start, Dir, Dist, IntrPt)

/* IntrPt = Start + Dir*Dist

2.2) GETSRFNRM (TR_Shp, TR_Obj, IntrPt, SrfNrm)

/* Işının göze en yakın cismi kesim noktasındaki yüzeyin normal vektörünü SRFNRM değerine döndürür.

2.3) GETLOCLCOL (TR_Shp, TR_Obj, Dir,

IntrPt, SrfNrm,
Dist, LoclCol)
/* Işının cismi kesim noktasındaki renk
değerini o noktadaki yüzeyin
normalinden de faydalanarak
LOCLCOL'a döndürür.

2.4) IF ((Depth = MaxDepth) OR
(TotWgt < MinWgt))

2.4.1) Col = LoclCol * LoclWgt

2.5) ELSE IF (Cisim Yansıtıcı Bir
Özelliğe Sahipse)

2.5.1) CALCDIROFREFRAY(Dir, SrfNrm,
ReflDir)

/* Yansıyan ışığın yön
vektörünü REFLDIR'e
döndürür.

2.5.2) Wgt = TotWgt * ReflWgt

/* Yüzeyin kaçta kaçının ışığı
yansıttığı bilgisinden
yararlanarak tekrarlı bir
yapıda ışının yüzeye olan
toplam etkisini bir sonraki
adımda azaltır.

2.5.3) TRACERAY (IntrPt, ReflDir,

```
                                        Wgt, (Depth+1), ReflCol )

                        /* Tekrarlı bir yapıda, başlangıç
                           noktası kesim noktası
                           olan, yön vektörü yansıyan
                           ışığın yön vektöru olan bir
                           ışın gönderilir. Böylece
                           yansıyan ışığın cismin
                           renginin oluşmasındaki
                           etkisi bulunur.

                2.5.4) COMB ( LoclCol, LoclWgt,
                                RefCol, ReflWgt, Col )

                        /* Col = LoclCol*LoclWgt +
                                 ReflCol*ReflWgt

        2.6) ELSE

                2.6.1) ReflCol = 0

                2.6.2) COMB ( LoclCol, LoclWgt,
                                RefCol, ReflWgt, Col )

                        /* Col = LoclCol*LoclWgt +
                                 ReflCol*ReflWgt
3) ELSE
        Col = 0

/**** TraceRay Modülünün Bitişi  ****/



SHOOTRAY ( Start, Dir, *Shp, *Obj, *Dist )
```

```
/**** ShootRay Modülünün Başlangıcı ****/

/* Genel olarak, bir ışının, uzayda göze en yakın olan
   cisim ile olan kesim noktasını döndürür. Göze en
   yakın cismi ve bu cisim ile olan kesim noktasını
   bulmak için, ışının  uzayda tanımlı bütün cisimlerle
   olan kesim noktası varsa bulunur. Göze en yakın
   nokta, kesim noktası pozitif en küçük olan
   noktadır.

1) *Shp = -1 , *Obj = -1, *Dist = -1,
   ObjHit = FALSE

2)  FOR ShapeNum = 0 TO MaxShapeType

        FOR ObjectNum = 1 TO ObjCnt[ShapeNum]

            2.1) IF ShapeNum = 0

                    NewDist = INTERSECT ( Start,
                                          Dir, Ground, 0 )
                        /* Yer'in şekil numarası sıfır
                           tanımlanmıştır. Dolayısı ile,
                           ışının yer düzlemi ile olan
                           kesim noktasının parametrik
                           değerini NEWDIST
                           değişkenine atar.

            2.2) ELSE

                    NewDist = INTERSECT ( Start,
                                          Dir, ShapeNum,
                                          ObjectNum )
            2.3) IF (NewDist > Epsilon)
```

2.3.1) IF (*Dist = -1)

2.3.1.1) ObjHit = TRUE

2.3.1.2) *Dist = NewDist

2.3.1.3) *Shp = ShapeNum

2.3.1.4) *Obj = ObjectNum

2.3.2) ELSE IF (NewDist<*Dist)

2.3.2.1) *Dist = NewDist

2.3.2.2) *Shp = ShapeNum

2.3.2.3) *Obj = ObjectNum

3) RETURN (ObjHit)

/**** ShootRay Modülünün Bitimi ****/

INTERSECT (Pt, Dir, Shp, Obj)

/**** Intersect Modülünün Başlangıcı ****/

1) CASE (Shp) = Ground

IF (Dir[2]=0)
Intersection = -1

ELSE IF ((-Pt[2]/Dir[2]) >Epsilon)
Intersection = -Pt[2]/Dir[2]

```
        ELSE
                Intersection = -1

2)  CASE (Shp) = Sphere

        I_Temp = Pt - Sphr[Obj].Loc

        I_b = VecDot ( Dir, I_Temp) * -2

        I_c= VecDot (I_Temp, I_Temp) -
                Sphr[Obj].Rad*Sphr[Obj].Rad

        I_disc = I_b*I_b - 4*I_c

        IF (I_disc <=0 )
                Intersection = -1

        ELSE

                I_sroot = Sqrt (I_disc)

                I_t1 = (-I_b - I_sroot)*0.5

                I_t2 = (-I_b + I_sroot)*0.5

                Intersection = MinPositive (I_t1, I_t2)

3)  CASE (Shp) = Cylinder

        Intersection =  QUADRATICINTERSECT
                                        (&Cyl[Obj], Pt, Dir)

4) CASE (Shp) = Cone
        Intersection =  QUADRATICINTERSECT
                                        (&Con[Obj], Pt, Dir)
```

5) CASE (Shp) = OtherShapes

/* Diğer cisimler için de kesim noktaları
bulunarak bu kısma eklenebilir.

6) RETURN (Intersection)

/**** Intersect Modülünün Bitimi ****/

QUADRATICINTERSECT (*List, Pt, Dir)

/*** QuadraticIntersect Modülünün Başlangıcı ***/

1) NewLoc = List.Loc - Pt

2) I_c = List.c4 + List.c1*NewLoc[0]*NewLoc[0]
+ List.c2*NewLoc[1]*NewLoc[1]
+ List.c3*NewLoc[2]*NewLoc[2]

3) I_b = 2*(List.c1*NewLoc[0]*Dir[0] +
List.c2*NewLoc[1]*Dir[1] +
List.c3*NewLoc[2]*Dir[2]

4) a = List.c1*Dir[0]*Dir[0] +
List.c2*Dir[1]*Dir[1] +
List.c3*Dir[2]*Dir[2]

5) disc = I_b*I_b - 4*a*I_c

6) IF (disc < 0)
Intersection = -1

7) ELSE

```
        I_sroot = Sqrt(disc)

        I_t2 = (-I_b + I_sroot) / (a+a)

        GETINTRPT(NewLoc, Dir, I_t2, loc2)

        IF ( loc2 cismin sınırları dışında ise )
                I_t2 = -1

        I_t1 = (-I_b - I_sroot) / (a+a)

        GETINTRPT(NewLoc, Dir, I_t1, loc1)

        IF ( loc1 cismin sınırları dışında ise )
                I_t1 = -1

        Intersection = MinPositive ( I_t1, I_t2 )

8) RETURN (Intersection)

/*** QuadraticIntersect Modülünün Bitimi  ***/

GETSRFNRM ( Shp, Obj, IntrPt, SrfNrm )

/**** GetSrfNrm Modülünün Başlangıcı  ****/

1)  CASE (Shp) = Ground

        Vec (0, 0, 1, SrfNrm)

2)  CASE (Shp) = Sphere

        SrfNrm = Sphr[Obj].Loc - IntrPt
```

VecNormalize (SrfNrm)

3) CASE (Shp) = Cylinder

QUADRATICSRFNRM (IntrPt, SrfNrm,
&Cyl[Obj])

4) CASE (Shp) = Cone

QUADRATICSRFNRM (IntrPt, SrfNrm,
&Con[Obj])

5) CASE (Shp) = OtherShapes

/* Diğer cisimler için de normal vektorleri
bulunarak bu kısma eklenebilir.

/**** GetSrfNrm Modülünün Bitimi ****/

QUADRATICSRFNRM (IntrPt, SrfNrm, *List)

/*** QuadraticSrfNrm Modülünün Başlangıcı ***/

1) NewPos = List.Loc - IntrPt

2) SrfNrm[0] = List.c1 * NewPos[0]

SrfNrm[1] = List.c2 * NewPos[1]

SrfNrm[2] = List.c3 * NewPos[2]

3) VecNormalize (SrfNrm)

/*** QuadraticSrfNrm Modülünün Bitimi ***/

```
GETLOCLCOL ( Shp, Obj, Dir, IntrPt, SrfNrm,
                                Dist, LoclCol )

/**** GetLoclCol Modülünün Başlangıcı  ****/

1)  IF (Shp = Lamp )

        1.1) IntensFactor = (1- DistEffect) +
                    DistEffect*( -VecDot(SrfNrm,Dir) /
                                Sqrt(Dist) )

        1.2) LoclCol = IntensFactor *
                        Lmp[Obj].Intens

2) LC_Mtl = Cismin Ait Olduğu Meteryalin
            Numarası.

3)  AMBIENT ()

        /* Difüzyon yolu ile yayılan ışığın etkisini
           veren faktör.

4)  FOR LC_Src = 1 TO ObjCnt[Lamp]

        4.1 ) LC_LmpDir = IntrPt - Lmp[LC_Src].Loc

        4.2) VecNormalize (LC_LmpDir)

        4.3) SHADOWFEELER (IntrPt)

            /* Kesim noktasından ışık kaynağı
               doğrultusunda bir ışın gönderir.
               Eğer bu ışın, ışık kaynağını kesiyorsa
               o zaman o nokta bu ışık kaynağına
               göre aydınlıktır, aksi takdirde, ışın
```

```
                başka bir cismi kesiyorsa o nokta bu
                ışık kaynağına göre karanlıktır. Işın,
                ışık kaynağını kesiyorsa, LC_ObjHit
                değişkeni TRUE yapılır.

        4.4) HitItself = LC_ObjHit &
                        (LC_ShadShp = Lamp ) &
                        (LC_ShadObj = LC_Src)

        4.5) IF (( !(LC_ObjHit)) || (HitItself)

                4.5.1) UNSHADOWEDSURFACE (
                                        Dir, SrfNrm)

                        /* Kesim noktasında cismin
                           rengini PHONG aydınlanma
                           modeline göre bulur.

                4.5.2) LoclCol = LC_Total1 +
                                LC_Total2

/**** GetLoclCol Modülünün Bitimi  ****/

AMBIENT ()

/**** Ambient Modülünün Başlangıcı  ****/

1) LC_Total1 = AmbIntens *
                Matl[LC_Mtl].AmbRfl
/**** Ambient Modülünün Bitimi  ****/

SHADOWFEELER ( IntrPt)

/**** ShadowFeeler Modülünün Başlangıcı  ****/
```

```
1) LC_ObjHit = SHOOTRAY (IntrPt, LC_LmpDir,
                         &LC_ShadShp, &LC_ShadObj,
                         &LC_ShadDist )

/**** ShadowFeeler Modülünün Bitimi  ****/


UNSHADOWEDSURFACE (Dir, SrfNrm)

/** UnShadowedSurface Modülünün Başlangıcı  **/

1) SS_Lamp = VecDot (SrfNrm, LC_LmpDir)

2)  IF (SS_Lamp<=0)

        2.1) SS_Diff = 0

        2.2) SS_Spec = 0

3)  ELSE

        3.1) SS_Diff = SS_Lamb *
                       Matl[LC_Mtl].DifRfl *
                       Lmp[LC_Src].Intens

        3.2) SPECULAR (Dir, SrfNrm)
             /* Parlak metalik yüzeylerde ışığın belli
                bir noktada yoğunlaşması neticesinde
                o bölgenin parlak gözükmesini
                modelleyen kısım.
4) LC_Total2 = SS_Diff + SS_Spec

/** UnShadowedSurface Modülünün Bitimi  **/
```

```
SPECULAR (Dir, SrfNrm)

/**** Specular Modülünün Başlangıcı  ****/

1) Temp = (Dir - LC_LmpDir) * 0.5

2) VecNormalize (Temp)

3) index = VecDot (SrfNrm, Temp)

4)  IF (index > 0.6)

        4.1) Intensity = Mstl[LC_Mtl].Gloss *
                         Log(index)

        4.2) Intensity = Exp (Intensity)

        4.3) SS_Spec = Intensity *
                       Matl[LC_Mtl].SpcRfl *
                       Lmp[LC_Src].Intens

5) ELSE
        SS_Spec = 0

/**** Specular Modülünün Bitimi  ****/

CALCDIROFREFLRAY ( Dir, SrfNrm, ReflRay)

/*** CalcDirOfReflRay Modülünün Başlangıcı  ***/
/* Yansıyan ışığın yön vektörünü hesaplar.

1) Tmp = -2 * VecDot (Dir, SrfNrm)

2) ReflRay = Tmp * SrfNrm + Dir
/*** CalcDirOfReflRay Modülünün Bitimi  ***/
```

Bir görüntüdeki cisimlere ait parametreler bir dosyada aşağıdaki şekilde tanımlanır. Aşağıda GORUNTU1.RT dosyasının içeriği gözükmektedir. Bu dosyada kullanıcı, görüntü ortamının özelliklerini ve görüntüdeki cisimlerin yerlerine ait bilgileri aşağıdaki şekilde tanımlar. Kullanıcı, görüntüdeki cisimlerin birbirleriyle olan etkileşimini CSG başlığı altında belirtir. RAYCAST.EXE programı çalıştırılıp parametre olarak GORUNTU1.RT dosyasının adı verilir. Şeklin gri tonlu görüntüsü ekranda oluşur. Oluşan şeklin renk değerleri GORUNTU1.CPR dosyasına atılır. Daha sonra SHOW.EXE programı çalıştırılıp paremetre olarak GORUNTU1.CPR dosyasının ismi verilerek şeklin renkli görüntüsü elde edilir.

GORUNTU1.RT dosyasının içeriği :

RESOLUTION

OUTFILE = GORUNTU1.CPR
XRES = 320
YRES = 200
XPITCH = 1.000
YPITCH = 1.000
XRANGE = 0 1
YRANGE = 0 1

ENVIRONMENT

LOCLWGT = 0.800 0.800 0.800
REFLWGT = 0.200 0.200 0.200
MINWGT = 0.030 0.030 0.030
MAXWGT = 1.000 1.000 1.000
RDEPTH = 1

OBSERVER

FLENGTH = 3.500

OBSPOS = 450.000 -850.000 300.000

ROTATE = -25.000

TILT = 8.000

AMBIENTLIGHT

INTENS = 0.200 0.200 0.200

MATERIAL

TYPE = A1

TEXTURE = CHECKER

AMBRFL = 0.100 0.100 0.100

DIFRFL = 0.700 0.700 0.700

SPCRFL = 0.200 0.200 0.200

GLOSS = 4.000

CHECK

TILE1 = 0.000 1.000 0.400

TILE2 = 0.500 0.200 0.800

TILE = 0.007

MATERIAL

TYPE = A2

TEXTURE = DUZ

AMBRFL = 0.000 0.000 0.000

DIFRFL = 0.800 0.100 0.100

SPCRFL = 0.100 0.100 0.100

GLOSS = 1.000

MATERIAL

TYPE = A3

TEXTURE = DUZ

AMBRFL = 0.100 0.100 0.100

DIFRFL = 0.200 0.200 0.500

SPCRFL = 0.200 0.200 0.200

GLOSS = 10.000

GROUND

MATL = A1

NAME = P10

LAMP

LOC = -1000.000 -1000.000 1000.000

RADIUS = 500.000

INTENS = 1.000 1.000 1.000

SPHERE

LOC = 80.000 0.000 90.000

RADIUS = 90.000

MATL = A1

NAME = P1

ID = 1

SPHERE

LOC = 130.000 -60.000 120.000

RADIUS = 23.000

MATL = A3

NAME = P3

ID = 3

SPHERE

LOC = 95.000 -100.000 90.000

RADIUS = 25.000

MATL = A3

NAME = P4

ID = 4

SPHERE

LOC = 150.000 100.000 100.000

RADIUS = 65.000

MATL = A2

NAME = P5

ID = 5

CONE

LOC = 80.000 0.000 250.000

DIRECT = 0.000 0.000 1.000

RADIUS = 60.000

HEIGHT = 90.000

MATL = A3

NAME = P7

ID = 12

SPHERE

LOC = -60.000 100.000 100.000

RADIUS = 65.000

MATL = A2

NAME = P8

ID = 13

BOX

LOC = -50.000 -70.000 60.000

V1 = 150.000 0.000 0.000

V2 = 0.000 300.000 0.000

V3 = 0.000 0.000 150.000

MATL = A3

NAME = P6

ID = 6

BOX

LOC = -100.000 -70.000 60.000

V1 = 250.000 0.000 0.000

V2 = 0.000 300.000 0.000

V3 = 0.000 0.000 250.000

MATL = A3

NAME = P9

ID = 14

CSG

P9 P P6 P - C7

C7 C P8 P - C8

C8 C P5 P - C9

C9 C P10 P + C10

END

GORUNTU2.CPR dosyasının içeriği :

RESOLUTION

OUTFILE = GORUNTU2.CPR

XRES = 320

YRES = 200

XPITCH = 1.000

YPITCH = 1.000

XRANGE = 0 1

YRANGE = 0 1

ENVIRONMENT

LOCLWGT = 0.800 0.800 0.800

REFLWGT = 0.200 0.200 0.200

MINWGT = 0.030 0.030 0.030

MAXWGT = 1.000 1.000 1.000

RDEPTH = 1

OBSERVER

FLENGTH = 3.500

OBSPOS = 450.000 -850.000 300.000

ROTATE = -25.000

TILT = 8.000

AMBIENTLIGHT

INTENS = 0.600 0.600 0.600

MATERIAL

TYPE = A1

TEXTURE = CHECKER

AMBRFL = 0.100 0.100 0.100

DIFRFL = 0.700 0.700 0.700

SPCRFL = 0.200 0.200 0.200

GLOSS = 4.000

CHECK

TILE1 = 1.000 1.000 0.400

TILE2 = 0.500 0.800 0.800

TILE = 0.005

MATERIAL

TYPE = A2

TEXTURE = DUZ

AMBRFL = 0.200 0.200 0.200

DIFRFL = 1.000 0.100 0.100

SPCRFL = 0.100 0.100 0.100

GLOSS = 25.000

MATERIAL

TYPE = A3

TEXTURE = DUZ

AMBRFL = 0.200 0.200 0.200

DIFRFL = 0.175 0.675 0.475

SPCRFL = 0.100 0.100 0.100

GLOSS = 250.000

MATERIAL

TYPE = A4

TEXTURE = DUZ

AMBRFL = 0.200 0.200 0.200

DIFRFL = 0.933 0.200 0.483

SPCRFL = 0.100 0.100 0.100

GLOSS = 125.000

GROUND

MATL = A1

NAME = P10

LAMP

LOC = -1000.000 -800.000 1000.000

RADIUS = 500.000

INTENS = 1.000 1.000 1.000

SPHERE

LOC = 80.000 40.000 90.000

RADIUS = 90.000

MATL = A3

NAME = P1

ID = 1

SPHERE

LOC = 65.000 -45.000 125.000

RADIUS = 23.000

MATL = A3

NAME = P2

ID = 2

SPHERE

LOC = 130.000 -20.000 120.000

RADIUS = 23.000

MATL = A3

NAME = P3

ID = 3

SPHERE

LOC = 95.000 -60.000 90.000

RADIUS = 25.000

MATL = A3

NAME = P4

ID = 4

SPHERE

LOC = 120.000 -60.000 90.000

RADIUS = 25.000

MATL = A3

NAME = P5

ID = 5

BOX

LOC = 60.000 -60.000 30.000

V1 = 70.000 0.000 0.000

V2 = 0.000 300.000 0.000

V3 = 0.000 0.000 20.000

MATL = A2

NAME = P6

ID = 6

CONE

LOC = 80.000 40.000 250.000

DIRECT = 0.000 0.000 1.000

RADIUS = 60.000

HEIGHT = 90.000

MATL = A4

NAME = P7

ID = 12

CSG

P2 P P3 P + C1

P1 P C1 C - C2

P4 P P5 P * C3

C2 C C3 C + C4

C4 C P6 P - C5

C5 C P7 P + C6

C6 C P10 P + C7

END

GORUNTU3.CPR dosyasının içeriği :

RESOLUTION

OUTFILE = GORUNTU3.CPR

XRES = 320

YRES = 200

XPITCH = 1.000

YPITCH = 1.000

XRANGE = 0 1

YRANGE = 0 1

ENVIRONMENT

LOCLWGT = 0.800 0.800 0.800

REFLWGT = 0.200 0.200 0.200

MINWGT = 0.030 0.030 0.030

MAXWGT = 1.000 1.000 1.000

RDEPTH = 1

OBSERVER

FLENGTH = 3.500

OBSPOS = 450.000 -850.000 300.000

ROTATE = -25.000

TILT = 8.000

AMBIENTLIGHT

INTENS = 0.200 0.200 0.200

MATERIAL

TYPE = A1

TEXTURE = CHECKER

AMBRFL = 0.100 0.100 0.100

DIFRFL = 0.700 0.700 0.700

SPCRFL = 0.200 0.200 0.200

GLOSS = 4.000

CHECK

TILE1 = 1.000 0.000 0.000

TILE2 = 0.000 1.000 1.000

TILE = 0.007

MATERIAL

TYPE = A3

TEXTURE = DUZ

AMBRFL = 0.200 0.200 0.200

DIFRFL = 0.200 0.800 0.200

SPCRFL = 0.100 0.100 0.100

GLOSS = 5.000

MATERIAL

TYPE = A2

TEXTURE = DUZ

AMBRFL = 0.200 0.200 0.200

DIFRFL = 0.100 0.100 0.800

SPCRFL = 0.100 0.100 0.100

GLOSS = 200.000

GROUND

MATL = A1

NAME = P10

LAMP

LOC = -1000.000 -800.000 1000.000

RADIUS = 600.000

INTENS = 1.000 1.000 1.000

BOX

LOC = 40.000 -70.000 180.000

V1 = 50.000 0.000 0.000

V2 = 0.000 300.000 0.000

V3 = 0.000 0.000 40.000

MATL = A3

NAME = P1

ID = 1

SPHERE

LOC = 40.000 -45.000 125.000

RADIUS = 50.000

MATL = A3

NAME = P2

ID = 7

BOX

LOC = 20.000 -70.000 60.000

V1 = 200.000 0.000 0.000

V2 = 0.000 300.000 0.000

V3 = 0.000 0.000 250.000

MATL = A3

NAME = P3

ID = 8

BOX

LOC = 100.000 -70.000 100.000

V1 = 350.000 0.000 0.000

V2 = 0.000 300.000 0.000

V3 = 0.000 0.000 50.000

MATL = A3

NAME = P4

ID = 15

BOX

LOC = 150.000 -70.000 200.000

V1 = 150.000 0.000 0.000

V2 = 0.000 100.000 0.000

V3 = 0.000 0.000 200.000

MATL = A3

NAME = P5

ID = 22

SPHERE

LOC = 185.000 20.000 230.000

RADIUS = 20.000

MATL = A3

NAME = P6

ID = 29

CSG

P3 P P2 P - C1

C1 C P1 P - C2

C2 C P4 P - C3

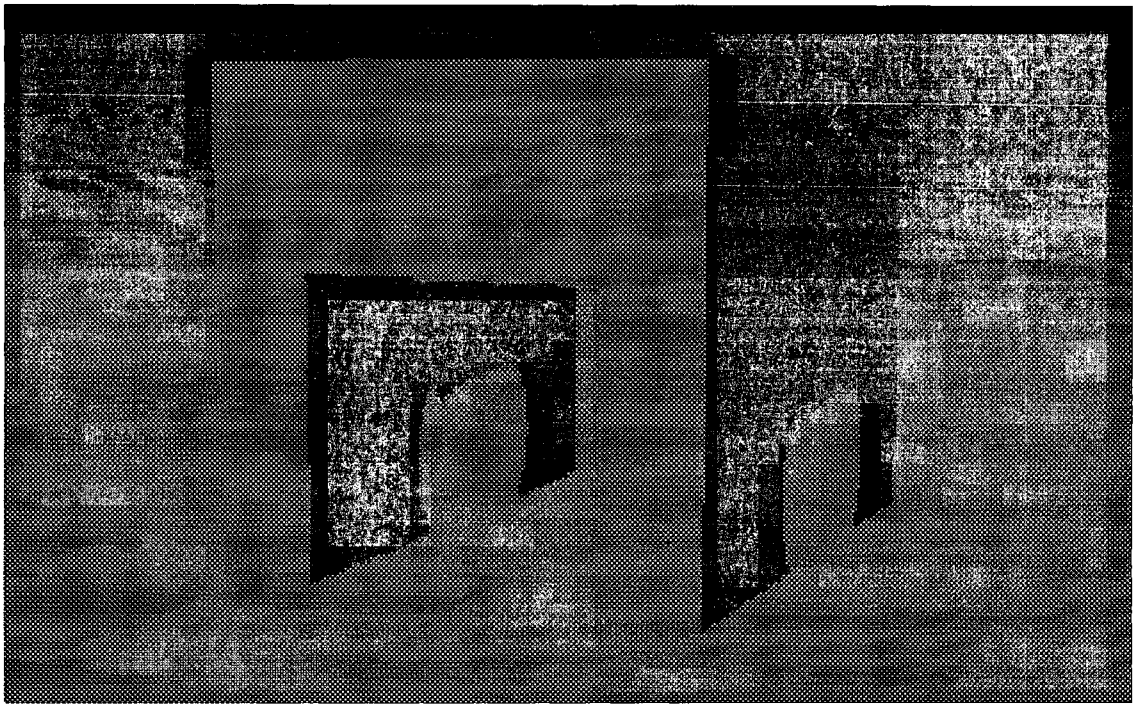
C3 C P5 P - C4

C4 C P6 P - C5

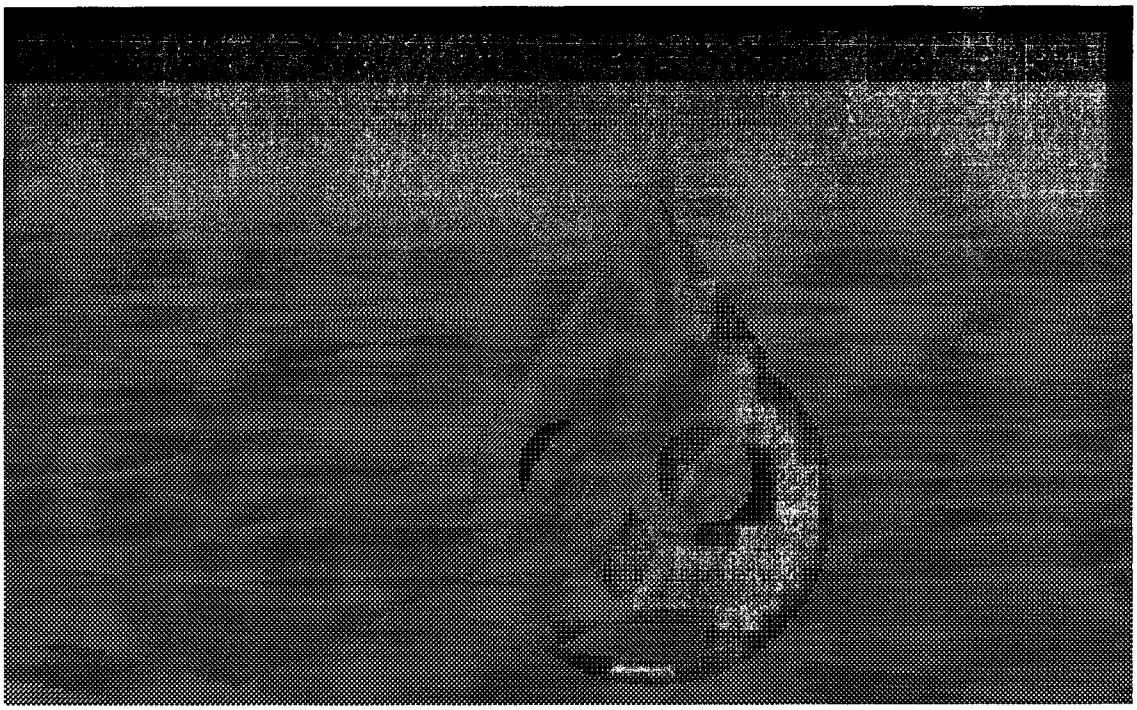
C5 C P10 P + C6

END

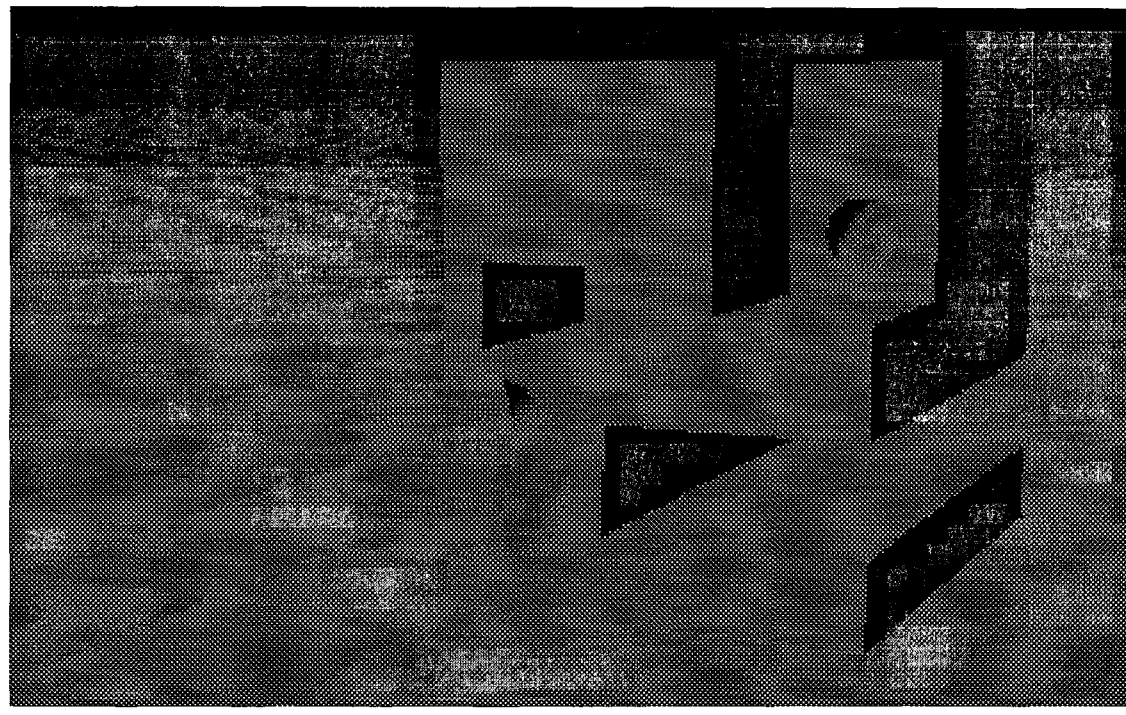
Aşağıda GORUNTU1.RT, GORUNTU2.RT ve GORUNTU3.RT dosyalarındaki bilgiler ışığında oluşan şekillerin görüntüleri gözükmektedir.



ŞEKİL 6.1



ŞEKİL 6.2



ŞEKİL 6.3

SONUÇLAR

Bir CAD/CAM sisteminin katı cisimleri modelleme yapısına sahip olması gereklidir. Yukarıda bahsedilen ışın izleme yönteminden faydalanmak bu ihtiyacı kısmen karşılamaktadır. Bahsedilen yöntemde, tekrarlı bir yapıda cismi oluşturan ağaç yapısının düğümlerindeki her bir cismin ışın ile olan kesim noktaları bulunur. Bu şekilde en üst seviyedeki kompleks cismin kesim noktaları tekrarlı yapıdan faydalanılarak elde edilir.

Performansı arttırıcı bir takım yöntemler kullanılabilir. Bunların başında, kapsayıcı cisimler (Küre, Küp gibi) kullanmaktır. Mevcut cisimleri kapsayacak şekilde bu cisimlerden faydalanılır. Kapsayıcı cisimlerin kesim noktalarını bulmak kolay olduğu için hız açısından büyük bir avantaj sağlanmış olunur. Gönderilen ışının önce bu cisimlerle kesişip kesişmediği bulunur. Bu cisimlerden herhangibirini kesmeyen bir ışın, onun içindeki cisimleri de kesmeyecektir. Performansı arttırıcı başka bir yöntem de, mevcut kodların Assembly dilinde yazılmasıdır. Ayrıca, kayan noktalı işlemleri tamsayılı işlemlere belli noktalarda dönüştürerek de performans artışı sağlanabilir.

KAYNAKLAR

Adams Alan, Rogers David, 1989.
"Mathematical Elements For Computer Graphics"

Finney Ross, Thomas George, 1990.
"Calculus And Analytic Geometry"

Harrington Steven, 1990.
"Computer Graphics A Programming Approach"

Kalley Gordon, Plastock Roy, 1989.
"Schaum's Outline Series Theory And Problems Computer Graphics"

Siggraph 88.
"An Introduction To Ray Tracing. Course Notes"

ÖZGEÇMİŞ

Doğum Tarihi : 2 Ocak 1973

Doğum Yeri : İstanbul

Eğitim : 1978 - 1983 Atatürk İlkokulu

1983 - 1986 Girne Anafartalar Lisesi

1986 - 1989 Malatya Lisesi

1989 - 1994 Yıldız Teknik Ünv. Bilgisayar Bil. Ve Müh.