50091

SANAL GERÇEKLİK ORTAMININ PC'DE SİMULASYONU

Hasan Makara

Dumlupınar Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektronik Eğitimi Anabilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman: Yrd.Doç.Dr.Abdullah Çavuşoğlu

Şubat-1996

Hasan Makara'nın YÜKSEK LİSANS tezi olarak hazırladığı "SANAL GERÇEKLİK ORTAMININ PC'DE SİMULASYONU" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

14.03.1996

İMZA

Üye : Yrd.Doç.Dr.Abdullah Çavuşoğlu (Danışman)

İMZA

Üye : Yrd.Doç.Dr. Erkan İmal

İMZA

Üye : Yrd.Doç.Dr.Osman Gürdal

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..14.03.1996..... gün ve .....05..... sayılı kararıyla onaylanmıştır.

İMZA

Fen Bilimleri Enstitüsü Müdürü

Yrd. Doç. Dr. İlyas Ulusoğlu

ÖZET

Bu çalışmada, sanal gerçeklik ortamının bilgisayarda simulasyonu yapılmıştır. Bilgisayar grafik ekranında üç boyutlu modeller elde edilmiş ve bilgisayar ekranında kullanıcıya üç boyutlu ortamda bulunduğu hissi verilmiştir.

Noktalar kullanılarak doğrular, doğruların birleştirilmesiyle yüzeyler ve sonunda bunlar ile nesneler oluşturulmuştur. Bilgisayar ekranı iki boyutlu olduğundan, üç boyutlu nesnelerin, bir bakış uzaklığı ve açısına göre izdüşümü alınmıştır. Bunun için tek noktalı perspektif projeksiyon yöntemi kullanılmıştır.

Nesnelerin hareketinin sağlanması nesneleri oluşturan noktalara dönüşüm işlemlerinin uygulanmasıyla yapılmıştır. Nesneye olan uzaklık ve bakış açısı mouse yardımıyla değiştirilerek nesne etrafında hareket sağlanmıştır.

Anahtar Kelimeler : Üç Boyutlu Bilgisayar Grafikleri, Sanal Gerçeklik, Simulasyon, Nesne, Perspektif, Projeksiyon

SIMULATION OF VIRTUAL REALITY ON A PC

ABSTRACT

In this study, a virtual reality environment has been simulated on a computer. Three dimensional (3D) models of objects were also formed and on the computer screen the user were given the impression of being in a 3D world.

The lines were formed by vertex points, by joining lines surfaces were formed and finally the objects were formed by these surfaces. Since the computer screen is two-dimensional the 3D objects the were projected onto screen with a viewing angle and viewing distance. To achieve this aim single point perspective projection method was used.

To produce movements of objects, vertex points were transformed. The viewing distance and viewing angle ware changed by movie therefore movements around the objects were obscured.

Key words : Three dimensional computer graphics, Virtual reality, Simulation, Object, Perspective, Projection

TEŞEKKÜR

Bu tezin hazırlanmasında, bana danışmanlık yapan, Gazi Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Sistemleri A.B.D. başkanı Sayın Yrd.Doç.Dr. Abdullah Çavuşoğlu'na, gerekli kaynaklara ulaşmamda yardımcı olan ve temel bilgiler veren Bilkent Üniversitesi Elektronik Bölümü Öğretim Görevlisi Sayın Dr. Tahsin Kunç'a teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

ŞEKİLLER DİZİNİ x

1. GİRİŞ 1

1.1. Uygulamanın Tarihçesi 1

1.2. Araştırmanın Amacı 2

2. TEMEL İŞLEMLER 3

2.1. Noktaların Tanımlanması 3

2.1.1. Üç boyutlu koordinat sisteminde noktanın gösterimi 4

2.2. Doğru Parçalarının Tanımlanması 5

2.3. Yüzeylerin Tanımlanması 7

2.3.1. Üç boyutlu yüzeyler ve düzlemler 7

3. TEMEL DÖNÜŞÜM İŞLEMLERİ 9

3.1. Üç boyutlu dönüşümler 10

3.1.1. Üç boyutlu konum değiştirme 11

3.1.2. Üç boyutlu eksenel ölçeklendirme 12

3.1.3. Üç boyutlu eksenel meyillendirme 13

3.1.4. Üç boyutlu eksenel döndürme 17

3.2. Üç Boyutlu Dönüşümlerin Birleştirilmesi 19

3.2.1. Yansıtma 19

3.2.2. Keyfi bir eksene göre döndürme 20

4. DÜZLEM GEOMETRİ PROJEKSİYONLARI 20

4.1. Paralel Projeksiyonlar 21

4.1.1. Ortographic paralel projeksiyon 22

İÇİNDEKİLER (devam)

Sayfa

4.1.2. Axonometric paralel projeksiyon 22
4.1.3. Oblique paralel projeksiyon 24
4.2. Perspektif Projeksiyonlar 27

5. TURBO PASCAL İLE 3B NESNELERİN SİMULASYONU 29

5.1. Program Ana Bloğu 29
5.1.1. Program başlığı bölümü 29
5.1.2. Veri bölümü 29
5.1.3. Program bölümü 31
5.2. Grafik Sistemi 32
5.3. Görüntü Parametreleri 32
5.4. Köşe, Görüş ve Ekran Koordinat Boyutları 33
5.4.1. Görüş alanı değişimi 33
5.5. Yüzey 36
5.6. Görünürlük Testi 37
5.7. Görünür Kenar Testi 37
5.7.1. Yüzey-normal vektörü 38
5.7.2. Görüş-hattı vektörü 39
5.8. Minimax Test 41
5.9. Çizim 41
5.10. Nesnenin üstünlüğü 41
5.11. Kesişim Testi 42
5.11.1. Kenar bitiş noktalarının görünürlük durumu 42
5.12. İkinci Öncelikli Nesnenin Görünen Kenarlarının Çizimi 43

6. SONUÇLAR 44

7. ÖNERİLER 44

KAYNAKLAR DİZİNİ 46

İÇİNDEKİLER (devam)

EKLER

1. Turbo Pascal Programlama Dilinde Simulasyon Programı
2. θ=40°, ϕ=80° ve *vd*=600 için nesnelerin görünüşü
3. θ=45°, ϕ=50° ve *vd*=250 için nesnelerin görünüşü
4. θ=45°, ϕ=35° ve *vd*=500 için nesnelerin görünüşü
5. θ=30°, ϕ=10° ve *vd*=400 için nesnelerin görünüşü

ŞEKİLLER DİZİNİ

Şekil		Sayfa
2.1.	Dikdörtgen koordinat sistemi	4
2.2.	Küresel koordinat sistemi	5
2.3.	Doğru parçası	6
2.4.	Bir nesne ve koordinat tablosu	6
2.5.	Üç boyutlu koordinat sisteminde yüzeylerin gösterimi	7
2.6.	Üç boyutlu koordinat sisteminde düzlem	8
3.1.	Koordinat sistemleri, (a) sol el koordinat sistemi. (b) sağ el koordinat koordinat sistemi	10
3.2.	Bir küp için konum değiştirme	12
3.3.	Nesnenin ölçeklendirilmesi (a) üç eksen için farklı ölçeklendirme (b) üniform ölçeklendirme	13
3.4.	Meyillendirme işleminin şematik gösterimi	15
3.5.	Küp için meyillendirme işlemi	16
3.6.	Pozitif dönüş yönleri	17
3.7.	xy düzleminin dönüşü	18
3.8.	Orjinden geçen bir doğrunun eksenlerle yaptığı açılar	21
4.1.	Dimetrik projeksiyon koordinat sistemi	23
4.2.	İzometrik projeksiyon koordinat sistemi	24
4.3.	Oblique projeksiyonu	25
4.4.	Bir küpün oblique projeksiyonu ile elde edilmiş görüntüsü	26
4.5.	Perspektif projeksiyon	28
5.1.	Küpün görünen yüzeylerinin sayısı (a) üç yanı görünür (b) sadece iki yanı görünür	38
5.2.	Görünürlüğün tanımlanmasında normal ve görüş hattı vektörleri	39
5.3.	Kenar bitiş noktalarının görünürlük durumu	43

1. GİRİŞ

Simulasyon, benzetme ve taklit manasına gelen bir kelimedir. Burada üç boyutlu nesneleri bilgisayar ekranında taklid yada benzetme çalışması yapılmıştır. Sanal Gerçeklik (Virtual reality) bilgisayar grafik ekranı üzerindeki geometrik modellerin görüntülerinin insan beyninde üç boyutlu gerçek görüntü hissini vermesini yada sanmasını amaçlar.

İnsan hayatı beş duyu üzerine kuruludur. Teknoloji duyu organlarına ulaşarak, mümkün olduğunca etkilemek ve bu etki sonucunda en zevklisinden en tehlikelisine kadar tüm uğraşlarda avantajlar getirmekte kararlıdır (Cerit, 1996).

Virtual reality yada Türkçe'ye yerleşmeye aday tanımıyla sanal gerçeklik, gerçekte olmayan veya hissedilemeyecek kadar uzakta olan olayları, kullanıcıya "sandırarak" en doğru refleks yada tepkileri almayı amaçlıyor (Cerit, 1996).

Bilgisayar ekranından, kullanılan yazılımın üç boyutlu dünyasına girip, işlerimizi hissederek yaşayarak yapmak sanal gerçekliğin özüdür.

1.1 Uygulamanın Tarihçesi

Her yenilik gibi sanal gerçekliğinde zamana ihtiyacı vardır. Şu an emekleme aşamasında olup üretim çok azdır ve bu nedenle her ürün kendi başına çok pahalıdır (Cerit, 1996).

Bu konudaki kaynaklara baktığımızda bu alandaki çalışmaların 1980'li yıllardan itibaren başladığı söylenebilir. Bu yıllarda, üç boyutlu geometrik modellerin simulasyonu yapılmıştır. Nesnelerin kafes görüntüleri daha sonraları katı modelleme üzerine çalışmalar yapılmıştır. Bu yıllarda bilgisayar donanımındaki gelişmeler, grafik adaptörleri ve matematiksel işlemcilerin hız ve kapasitelerinin artması bu alandaki çalışmalarıda hızlandırmıştır.

1980'lere gelindiğinde güçlü iş istasyonları mühendislerin, bilim adamlarının, grafikerlerin ve mimarların standart yardımcı gereci haline gelmiştir. Bu gibi iş istasyonları, kullanıcıya sadece yüksek hesaplama gücünü değil aynı zamanda grafik komut kütüphaneleri, kullanımı kolay görsel arabirimler ve diğer cihazlarla bağlantılar vasıtasıyla, çeşitli özelliklerin kazanılması gibi güçlü grafik yardımcı gereçler setini de hazır olarak sunmuştur. CAD sistemi kullanıcıları şimdi, etkileşimli üç boyutlu katı modelleyicilerle çalışmaktadırlar. Animatörler ve TV-reklam ressamları mevcut donanım ve yazılımları kullanarak gerçeğe yakın resimleri oluşturabilmektedirler (Erdun, 1993).

Bugün her türden tasarımcılar, sanal gerçekliğin en büyük kullanıcı grubunu oluşturmaktadır. Bir bina yapılmadan önce bilgiler bilgisayara girilip binanın yapılmış şekli incelenebilmekte ve tek bir taş oynatmadan meydana çıkabilecek problemler çözülebilmektedir. Sanal gerçeklikle ilgili çalışmalar gelecekte çok artacaktır. Sanal gerçeklikle ilgili yazılımlar da kullanıcının rahatça programlayabileceği seviyelere ulaştırılmaya çalışılmaktadır. Şu an sanal gerçeklik için en büyük yatırımı Japonya yapmaktadır.

1.2 Araştırmanın Amacı

Bu araştırmanın amacı üç boyutlu geometrik modellerin yazılım yoluyla oluşturulması, kafes ve katı modellerinin incelenmesi ve temel dönüşüm denklemlerinin uygulanmasıdır.

İki nesne arasındaki uzaklık yakınlık durumunun gerçeğe yakın bir şekilde oluşturulması, görüş uzaklığı ve görüş açısının değiştirilerek nesnelerin görünürlük durumlarının incelenmesi ve üç boyutlu ortamın simulasyonunun yapılmasıdır.

Programda bir ev nesne olarak seçilmiş ve evin dışarıdan görünüşü etrafında hareket edilerek ekrana yansıtılmıştır.

Bu araştırmada, yapılan yazılım ile geometrik modelleme, modellerin kafes ve katı görüntülerinin elde edilmesi ve hareketi ile üç boyutlu ortamda görüldüğü hissini vermesi amaçlanmıştır.

2. TEMEL İŞLEMLER

Bilgisayarla bir nesne görüntüsünün çizilebilmesi ve bu görüntü üzerinde çeşitli matematiksel dönüşümlerin uygulanabilmesi için nesnenin tanımlanması gerekir. Bir nesnenin tanımlanması, seçilen uygun bir koordinat sistemine göre noktalar ve bunları birleştiren doğrularla yapılır. Nesne üzerinde herhangi bir dönüşümün gerçekleştirilmesi doğrudan bu noktalar kullanılarak yapılır. Örneğin, nesnenin büyüklüğünün değiştirilmesi, bir yerden başka bir yere taşınması, nesnenin döndürülmesi ve perspektifinin oluşturulması gibi dönüşümlerin yerine getirilmesi nesneyi tanımlayan noktaların herbirine bu dönüşüm işlemlerinin ayrı ayrı uygulanmasıyla yapılır. Bu dönüşümler, bilgisayarla grafiğin temelini oluşturur. Borland grafik paketi, programcıya sadece çizimlerde kullanılan temel ve yardımcı komutları verir. Nesne üzerinde dönüşümleri gerçekleştirmek için kullanılabilecek herhangi bir komutu içermez.

Dönüşümler, 2 ve 3 boyutlu nesneler için uygulanabilir matematiksel formüllerdir. Bilgisayarın ekranı iki boyutlu bir yüzey olduğundan programcı çizimlerinde iki boyutlu olarak çalışmak zorundadır. Üç boyutlu nesnelerin ekranda görüntülenmesi, nesnenin boyutunun bir azaltılmasıyla gerçekleşir. Boyutun azaltılması nesne görüntüsünün ekran düzlemi üzerine izdüşürülmesiyle yapılır (Erdun, 1993).

2.1 Noktaların Tanımlanması

Noktalar sıfır boyutlu nesneler olup uzayda bir yeri belirtirler. Bu noktalar kullanılarak doğruların ve doğrular kullanılarak da çeşitli nesnelerin çizimini yapmak mümkündür. Noktalar üç boyutlu bir koordinat sisteminde gösterilebilirler(Şekil 2.1).

Matematiksel olarak bir nokta matris formunda gösterilebilir. Noktaların matris şeklinde tanımlanması dönüşüm işlemlerin daha iyi anlaşılmasını ve yazımını sağlar. Üç boyutlu koordinat sisteminde nokta, matris olarak iki şekilde gösterilebilir (Erdun 1993).

$$P = [\, x \;\; y \;\; z \,]$$

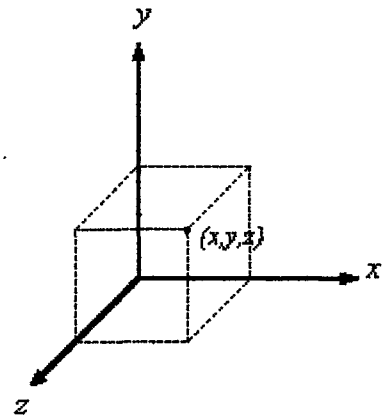
$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Burada tanımlanan P matrisleri, genelde vektör olarak isimlendirilirler.

2.1.1 Üç boyutlu koordinat sisteminde noktanın gösterimi

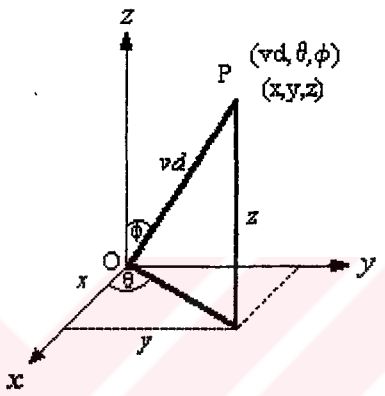
Üç boyutlu koordinat sisteminde noktaların gösterilmesi için iki ortak yol vardır: Dikdörtgen koordinat sistemi ve küresel koordinat sistemi.

Dikdörtgen koordinat sistem : Noktalar *yz*, *xz* ve *xy* yüzeylerindeki noktadan *x*, *y* ve *z* koordinatına doğrudan uzaklığı ile teşhis edilen üçlü sıra (x,y,z) değerleridir (Şekil 2.1).



Şekil 2.1 - Dikdörtgen koordinat sistemi.

Küresel koordinat sistem : Bu sistemde her bir nokta (vd, θ, ϕ) değerlerinin üçlü dizisi ile gösterilir (Şekil 2.2). P noktasından orjine olan uzaklık vd'dir. Pozitif yöndeki z ekseni ve OP çizgisi arasındaki açı ϕ dir. Pozitif yöndeki x ekseni ile OP çizgisi arasındaki açı θ dır. Bu açı x ekseninden saat istikametinin ters yönündedir.

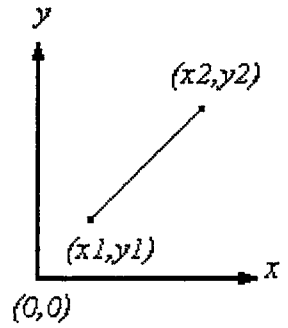


Şekil 2.2 Küresel koordinat sistemi

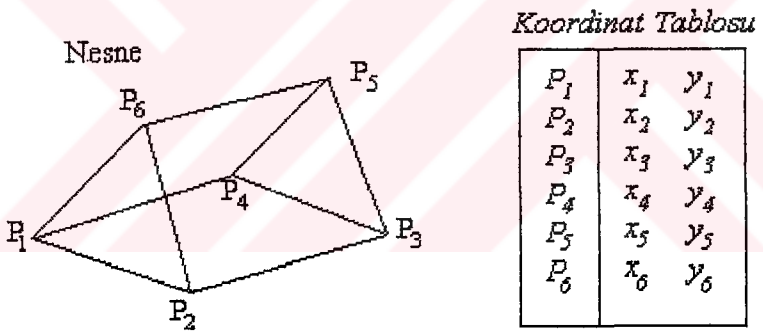
2.2 Doğru Parçalarının Tanımlanması

Bir doğrunun tanımlanabilmesi için en az iki noktaya ihtiyaç vardır. Bir doğru parçasının tanımlanabilmesi için ise doğru parçasının başlangıç ve bitiş noktalarının belirlenmesi gereklidir (Şekil 2.3).

Doğru parçasının diğer bir tanımlama şekli de vektör olarak gösterilebilmesidir. Bir lokal koordinat sisteminde, nesneyi oluşturan vektörleri belirleyen noktalar serisi, bir matris olarak bilgisayarda gösterilebilir. Bu matris bir nesneyi tanımlayan noktaların tümünü içerir. Bu durumda, bir nesnenin bir dönüşüme tabi tutulması, bu matris değerlerinin tümüne ilgili dönüşüm işlemlerinin uygulanması ile sağlanır (Erdun 1993).



Şekil 2.3 - Doğru Parçası



Şekil 2.4 - Bir nesne ve koordinat tablosu

Şekil 2.4'de, nesnenin koordinat değerlerini tutan matris *P* ile, bu matris içindeki bir değer ise *Pi* ile gösterilmektedir. Burada *i* noktanın sıra numarasını belirtir (i=1,2,3,4,5,6).

Bu durumda nesnenin *i.* noktası aşağıdaki gibi gösterilebilir.

$$P_i = [\ x_i \ \ y_i\ ]$$

2.3 Yüzeylerin Tanımlanması

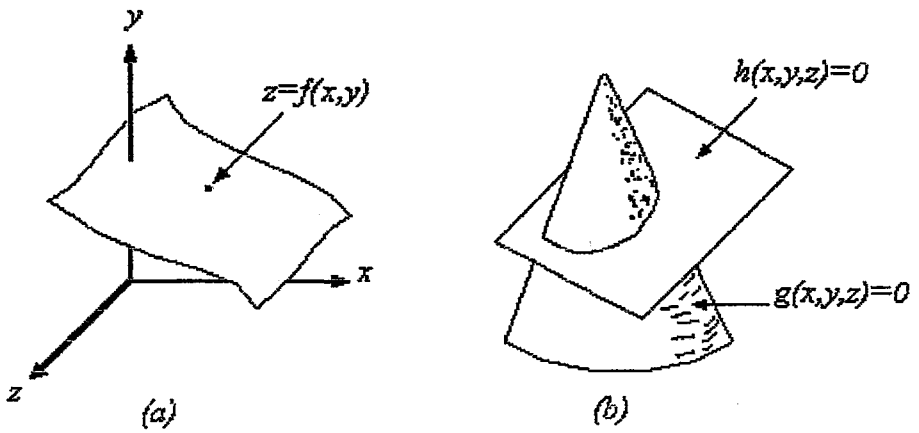
Nesnelerin geometrik modellerinin yüzey tanımlamaları yapılır. Bir nesnenin kaç yüzeyden oluştuğu ve bu yüzeylerin biçimi, yüzeyi oluşturan noktalar tarafından belirlenir. Bir küpün altı yüzeyi vardır ve her bir yüzey dört köşe ile tanımlanır. Nesneleri dışarıdan göründüğü şekliyle her bir yüzeyinin etrafında dizilen noktaları saat ibresinin ters yönünde tanımlamak önemlidir.

2.3.1 Üç boyutlu yüzeyler ve düzlemler

Üç boyutlu bir koordinat sisteminde bir yüzeyin gösterimi

$$z = f(x, y) \quad \text{veya} \quad g(x, y, z) = 0$$

şeklinde olmak üzere iki biçimdedir. Birincisinde x ve y koordinat değerleri kullanılarak z değerleri ile yüzey oluşturulur. İkincisinde bir değişkenin, örneğin z nin, her verilen sabit değeri için diğer değişken değerlerinin değiştirilmesiyle elde edilen eğri ailelerinden yüzeyler oluşturulur. Bu yüzeylere örnek olarak Şekil 2.5 ' deki şekiller verilebilir.



Şekil 2.5 - Üç boyutlu koordinat sisteminde yüzeylerin gösterimi

Bilgisayarla grafikte doğru denklemi

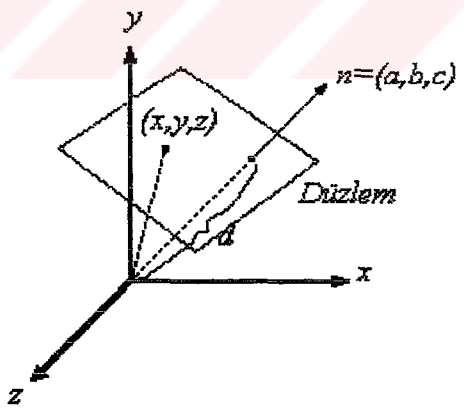
$$ax + by + cz + d = 0$$

şeklindedir.

Burada, eğer *d* sıfırdan farklı ise denklemin *d* ye bölünmesinden sonra düzlemde bir değişiklik olmaz. Bu durumda denklemde sadece üç nokta bir düzlemi belirlemeye yetecektir. Burada, *d* orjinden düzleme olan uzaklığı gösterir. Ayrıca, aynı *a*, *b*, ve *c* katsayılarına sahip denklemler ile elde edilen tüm düzlemler de birbirine paralel olacaktır. $d = 0$ ile bir düzlemi göz önüne alırsak,

$$ax + by + cz = 0$$

bu düzlem orjinden geçmelidir.



Şekil 2.6 Üç boyutlu koordinat sisteminde düzlem.

(x, y, z) noktası düzlem üzerinde keyfi bir noktadır ve daha önce verilen P matrisi $P = [\ x\ \ y\ \ z\ ]$ ile gösterilebilir. Orjinden (x, y, z)'ye olan doğru veya vektör, düzlem içinden geçer. Aynı zamanda (a, b, c) katsayı üçlüsünü üç boyutta bir nokta olarak ele alabilir ve aşağıdaki gibi yazabiliriz.

$$n = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

iki vektörün scaler çarpımını kullanarak, orjinden geçen düzlemin denklemini aşağıdaki gibi yazabiliriz.

$$P.n = [\, x \;\; y \;\; z \,] \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

n, düzlemdeki tüm vektörlere diktir. Bu yüzden üç boyutlu uzayda düzlemin istikametini kontrol eder. Sadece düzlemin orjinden olan uzaklığına etki ettiğinden a, b ve c daima

$$ax + by + cz + d = 0$$

düzlemine dik bir vektörü belirtir.

3. TEMEL DÖNÜŞÜM İŞLEMLERİ

Bir nesneye uygulanabilecek dört tane temel dönüşüm işlemi vardır. Bunlar :

1. Konum değiştirme (Translation)

2. Ölçeklendirme (Scaling)

3. Meyillendirme (Shearing)

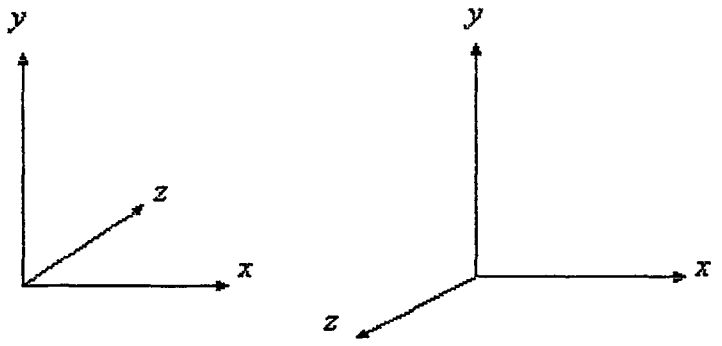
4. Döndürme (Rotating)

Bu dört temel işlem, ayrı olarak veya ard arda kullanılarak tüm dönüşümler gerçekleştirilir (Erdun, 1993).

3.1 Üç Boyutlu Dönüşümler

Üç boyutlu grafikler, bir çok yönden iki boyutlu grafiklerin uzantısıdır. Üç boyutlu grafiklerin bilgisayar ekranında görüntülenmesinde problemle karşılaşılır. Bu problem, bilgisayar ekranının iki boyutlu, nesnenin ise üç boyutlu olmasıdır. Üç boyutlu grafiklerin bilgisayar ekranında görüntülenmesi üç boyutlu grafiklerin iki boyutlu hale dönüştürülmesi ile yapılabilir. Bu da üç boyutlu nesne görüntüsünün projeksiyon yöntemi ile bilgisayar ekranına izdüşürülmesi ile olur.

Üç boyutlu nesnelerin tanımlanması noktalarla, bu noktaları birleştiren doğrularla ve bu doğruların oluşturduğu yüzeylerle yapılır. Üç boyutlu nesnenin tanımlanması bir koordinat sistemi kullanılarak yapılır. Bu koordinat sistemleri aşağıdaki şekillerde görüldüğü gibi sırasıyla sol-el ve sağ-el sistemleri olarak bilinir (Erdun, 1993).



a b

Şekil 3.1 - Koordinat sistemleri,(a) Sol el koordinat sistemi
(b) Sağ el koordinat sistemi.

Grafik uygulamalarında her iki sistem de kullanılır. Genellikle, nesnenin

tanımlanmasında sağ el, görüntülenmesinde ise sol el sistemi tercih edilir.

Üç boyutlu homojen koordinat sisteminde bir noktanın gösterimi için kullanılan P vektörü

$$P = [\, wx \;\; wy \;\; wz \;\; w \,]$$

şeklindedir. Burada w sıfırdan farklı bir değer olmalıdır. İkinci boyutta bir nokta, üç boyutlu homojen koordinatlarda kullanılacağı zaman w daima 1 olarak alınır. Fakat üç boyutlu nesnelerin ekrana görüntülenmesi durumunda w için farklı değerler kullanılabilir.

3.1.1 Üç boyutlu konum değiştirme

Üç boyutlu koordinat sisteminde bir nesnenin konum değiştirmesi, nesneyi belirten tüm noktaların x,y ve z koordinatlarına sırasıyla Δx, Δy ve Δz öteleme miktarlarının eklenmesiyle yapılır. Üç boyutlu konum değiştirme denklemleri

$$x' = x + \Delta x, \qquad y' = y + \Delta y, \qquad z' = z + \Delta z$$

şeklindedir.

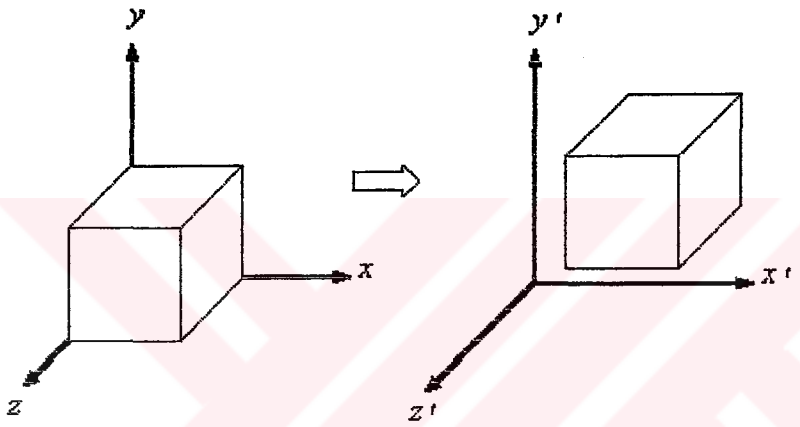
Bu konum değiştirme işleminin homojen koordinatlarında matris formunda gösterimi,

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix}$$

ve dönüşüm işleminin gösterimi,

$$[\,x'\ y'\ z'\ 1\,] = [\,x\ y\ z\ 1\,] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix}$$

şeklindedir. Konum değiştirme işleminin bir köşesi orjinde olan küp için uygulaması Şekil 3.2 'deki şekilde verilmiştir.



Şekil 3.2 - Bir küp için konum değiştirme.

3.1.2 Üç boyutlu eksenel ölçeklendirme

Üç boyutta bir nesnenin ölçeklendirilmesi, nesneyi belirten noktalara ait üç koordinat değerinin sırasıyla α, β, ve μ gibi birer ölçek katsayılarının çarpımıyla gerçekleştirilir. 3D ölçeklendirme dönüşüm denklemleri,

$$x' = \alpha x,$$

$$y' = \beta y,$$

$$z' = \mu z$$

Bu ölçeklendirme dönüşüm işleminin homojen koordinatlarında matris formunda gösterimi,

$$O = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ve dönüşüm işleminin gösterimi,

$$[\, x' \;\; y' \;\; z' \;\; 1 \,] = [\, x \;\; y \;\; z \;\; 1 \,] \cdot \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

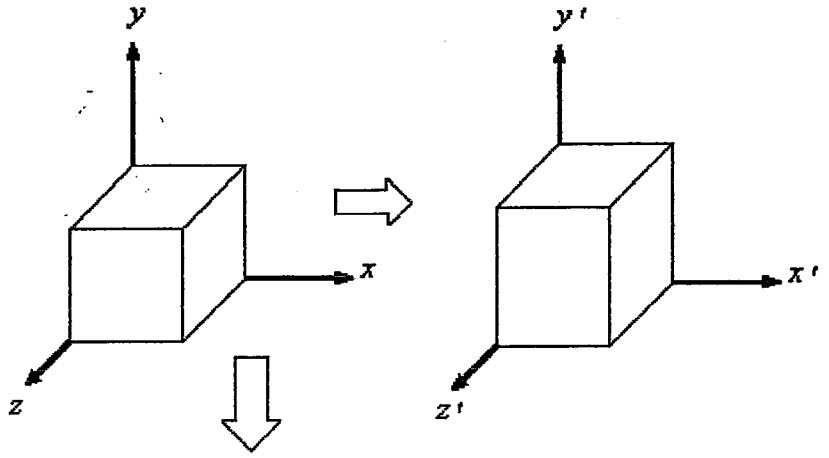
şeklindedir. Ölçeklendirme işleminde ölçeklendirme katsayılardan birinin 1'den küçük olması durumunda nesnenin boyu ilgili yönde küçülür. 1'den büyük olması durumunda nesnenin boyu ilgili yönde büyür. Nesne boyutunun her yönde eşit oranlarda büyütülmesi veya küçültülmesi, her üç ölçeklendirme katsayısının biribirine eşit olarak alınmasıyla sağlanır (üniform ölçeklendirme). Bu durumda ölçeklendirme dönüşüm matrisi aşağıdaki duruma gelir.

$$O = \begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

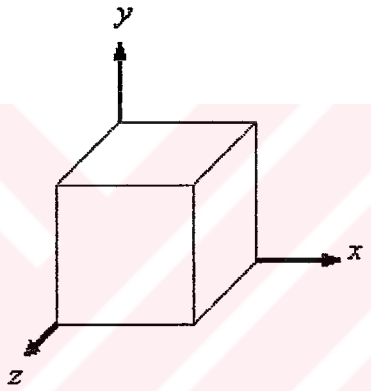
Ölçeklendirme işleminin bir köşesi orjinde olan küp için uygulaması şekil 3.3'de verilmiştir.

3.1.3 Üç boyutlu eksenel meyillendirme

Bir nesne, üç pozitif üç negatif eksen olmak üzere altı bağımsız yönde meyillendirilebilir. *x*-ekseni boyunca yapılan meyillendirme, sırasıyla *y* ve *z* eksenlerinin *x* ekseniyle 90 dereceden farklı bir açı yapacak şekilde meyillendiril-



(a)



(b)

Şekil 3.3 - Nesnenin ölçeklendirilmesi.
(a) Üç eksen için farklı ölçeklendirme. (b) Üniform ölçeklendirme.

mesi şeklindedir. Bu işlem aşağıdaki denklemlerle yapılır.

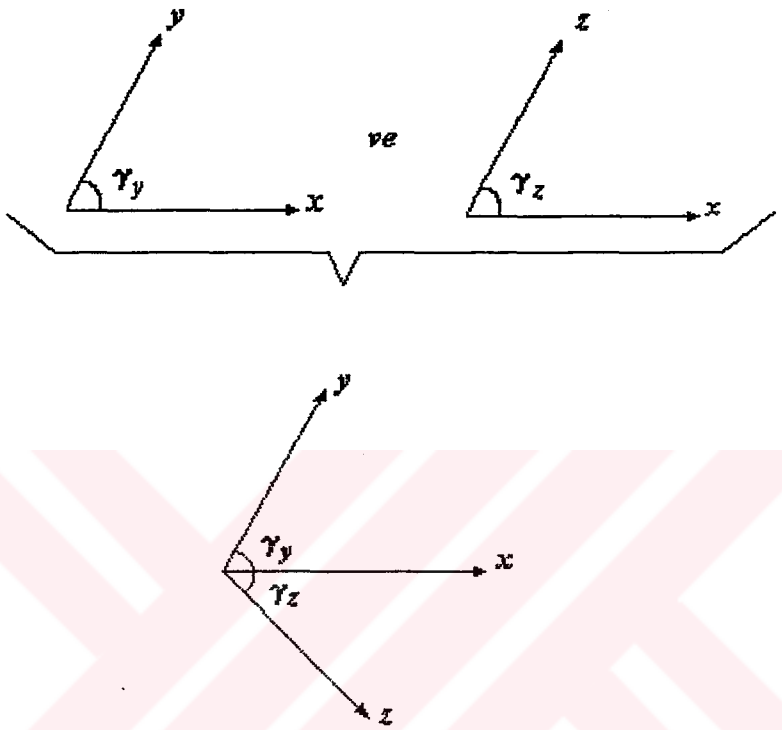
$$x' = x + y.\text{cotan}\gamma_y + z.\text{cotan}\gamma_z$$

$$y' = y$$

$$z' = z$$

Burada meyillendirme x ekseni yönünde olduğundan nesnenin sadece x koordinatları değiştirilir, y ve z koordinat değerleri değiştirilmeden bırakılır.

Meyillendirme işleminin şematik gösterimi aşağıda verilmiştir.



Şekil 3.4 - Meyillendirme işleminin şematik gösterimi.

x ekseni yönündeki meyillendirme işleminin homojen koordinatlarda matris formunda gösterimi

$$Mx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \cot an\gamma_y & 1 & 0 & 0 \\ \cot an\gamma_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ve dönüşüm işleminin gösterimi

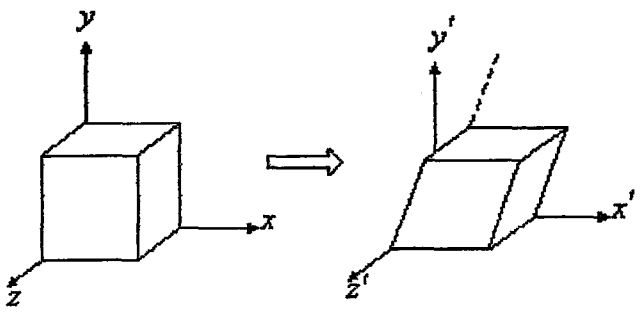
$$[\,x'\ y'\ z'\ 1\,] = [\,x\ y\ z\ 1\,] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ \cot an\gamma_y & 1 & 0 & 0 \\ \cot an\gamma_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklindedir. Buna benzer olarak y ve z yönündeki My ve Mz dönüşüm matrisleri de bulunabilir.

$$My = \begin{bmatrix} 1 & \cot an\gamma_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \cot an\gamma_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Mz = \begin{bmatrix} 1 & 0 & \cot an\gamma_x & 0 \\ 0 & 1 & \cot an\gamma_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

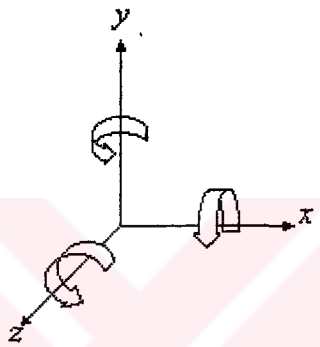
Meyillendirme işleminin bir köşesi orjinde olan küp için uygulaması aşağıdaki şekilde verilmiştir.



Şekil 3.5 - Küp için meyillendirme işlemi.

3.1.4 Üç boyutlu eksenel döndürme

Üç boyutlu koordinat sisteminde pozitif dönme yönünün ne olacağı önemlidir. Bir eksen etrafında pozitif dönüş, döndürme eksenin pozitif tarafından orjine doğru bakıldığında saat ibresinin dönüş yönünün tersi olarak belirlenir. Kartezyen koordinat sisteminde pozitif dönüş yönleri şekil 3.6'da gösterilmiştir.



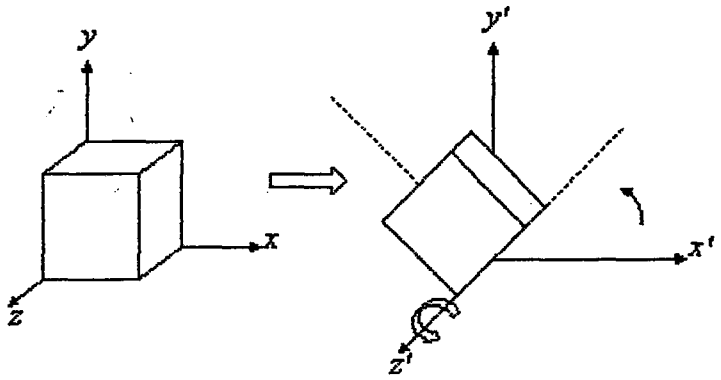
Şekil 3.6 - Pozitif dönüş yönleri.

Üçüncü boyutta, üç eksen etrafında biribirinden bağımsız dönüşler yapılabilir. Bir eksen etrafında döndürme işlemi yapılırken nesnenin koordinatlarından, döndürmenin yapıldığı eksene ait koordinat değeri sabit tutularak diğer koordinat değerleri dönüşüm işlemine tabi tutulur. Örneğin z ekseni etrafında bir dönüş (xy düzleminin dönüşü) şekil 3.7'de verilmiştir. Şekilden de görüleceği gibi z ekseni etrafında bir noktayı döndürdüğümüz zaman, z değeri sabit kalmaktadır.

Bu gözlem bize, üç boyutlu problemi sabit bir z düzleminde iki boyuta indirgememize izin verir. z ekseni etrafında dönüş için dönüşüm denklemleri:

$$x' = x.\cos\theta_z - y.\sin\theta_z$$

$$y' = x.\sin\theta_z + y.\cos\theta_z \qquad \text{ve} \qquad z' = z \qquad \text{dir.}$$



Şekil 3.7 - xy düzleminin dönüşü.

Bu döndürme işleminin homojen koordinatlarda matris formunda gösterimi,

$$R_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklindedir. Burada z indisi döndürmenin yapıldığı ekseni göstermektedir. Bu dönüşüm, basitce z ekseni etrafında xy düzleminin θ_Z kadar döndürülmesidir. z ekseni etrafındaki dönüşe benzer olarak x ve y ekseni için dönüşüm denklemleri çıkartılabilir. x ekseni etrafında dönüşte nesne noktalarının x değerleri sabit tutulur. y ekseni için de y değerleri sabit tutulur. Bu eksenler için homojen koordinatlarda dönüşüm matrisleri aşağıdaki gibi verilebilir.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

θ_x, θ_y ve θ_z açılarına bağlı olarak bir nesnenin her üç eksene göre döndürülmesini sağlayan dönüşüm matrisi, her eksen için ayrı olarak yazılan döndürme dönüşüm matrislerinin çarpımı ile elde edilir.

$$R = [Rx].[Ry].[Rz]$$

3.2 Üç Boyutlu Dönüşümlerin Birleştirilmesi

3.2.1 Yansıtma

Yansıtma işlemi 3 Boyutta bir düzleme göre yapılır. Objenin boyutlarının değiştirilmeden yansıtma dönüşümünün yerine getirilmesi için, dönüşümün matrisinin determinantı -1 olmalıdır.

x -y düzlemine göre yansıtma işleminde, sadece nesnenin z koordinat değerleri değiştirilecektir. Bu gerçekte z değerinin işaretinin değiştirilmesidir. Bu yüzden, x-y düzlemine göre bir yansıtma için kullanılan dönüşüm matrisi homojen koordinatlarda aşağıdaki gibi verilir.

$$T_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y-z düzlemine göre bir yansıtma için kullanılan dönüşüm matrisi

$$T_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

x-z düzlemine göre bir yansıtma için kullanılan dönüşüm matrisi

$$T_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklindedir.

Keyfi bir düzleme göre simetrik görüntünün oluşturulmasında izlenmesi gereken yol; simetri için kullanılan düzlemin eksen düzlemlerinden biri ile çakıştırılması, yukarıda verilen dönüşüm matrislerinden uygun olanı kullanarak simetrik görüntünün oluşturulması ve son olarak düzlemin tekrar eski durumuna döndürülmesi şeklinde yapılır.

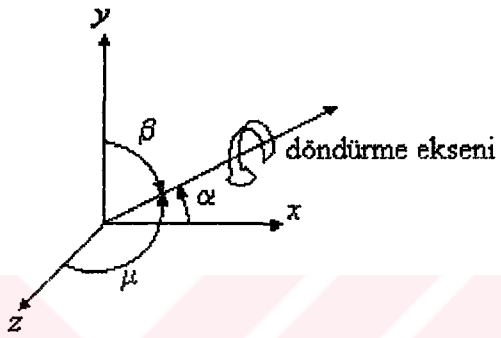
3.2.2 Keyfi bir eksene göre döndürme

Üç boyutlu koordinat sisteminde bir nesnenin döndürülmesi, koordinat sisteminin orjininden geçen bir eksen etrafında yapılır. Bunun için kullanılan dönüşümlerin sırası: Bir üç boyutlu öteleme, Bir orjin etrafında döndürme ve son olarak nesneyi ilk pozisyona geri öteleme şeklindedir (Şekil 3.8).

4. DÜZLEM GEOMETRİ PROJEKSİYONLARI

Üç boyutlu bir nesnenin oluşturulma ve dönüşüm işlemleriden sonra görüntüleme işlemi yapılacaktır. Bilgisayarda grafikle görüntüleme ortamı olarak

kullanılan ekran iki boyutlu bir düzlemdir. Bu aşamada karşımıza çıkan problem üç boyutlu nesnenin iki boyutlu ortamda nasıl görüntülenebileceğidir. Tek çözüm iki boyutlu nesneleri iki boyuta indirgemektir. Bu boyut azaltma işlemi üç boyutlu bir görüntünün iki boyutlu bir düzlem üzerine iz düşürülmesi yani nesnenin *z* koordinatının atılmasıdır.



Şekil 3.8 - Orjinden geçen bir doğrunun eksenlerle yaptığı açılar.

Üç boyutlu nesnenin iki boyutlu düzlemde iz düşümünü elde etmek için kullanılan çeşitli teknikler mevcuttur. Bunlardan kullanımı en yaygın olanları: Paralel projeksiyon ve perspektif projeksiyondur. Projeksiyonda önemli olanlar, projeksiyon çizgilerinin projeksiyon düzlemi ile yaptığı açı (projeksiyon yönü- nesneye hangi yönden bakıldığı) ve projeksiyon merkezinin nesneye olan uzaklığıdır. Paralel projeksiyonda, projeksiyon merkezinin uzaklığı sonsuz olarak kabul edilir. Bu yüzden projeksiyon çizgileri birbirine paraleldir. Perspektif projeksiyonda ise projeksiyon çizgileri paralel değildir. Bu projeksiyonlar kendi içinde çeşitli sınıflara ayrılırlar.

4.1. Paralel Projeksiyonlar

Paralel projeksiyon, projeksiyon çizgileri birbirine paralel olacak şekilde nesne üzerine gelerek projeksiyon düzlemi üzerine nesne görüntüsünün iz düşürülmesidir.

4.1.1. Ortographic paralel projeksiyon

Ortographic paralel projeksiyonda birbirine paralel olan projeksiyon çizgileri (projeksiyon yönü), projeksiyon düzlemine diktir. Burada projeksiyon yönü kartezyen koordinat sisteminin ana eksenlerinden biri olarak alınır. Eğer projeksiyon yönü *z* ekseni yönünde alınırsa, nesnenin iz düşümü görüntüsü bir *zy* düzleminde yer alacaktır.

Bu projeksiyonla elde edilen görüntü, nesne hakkında çok fazla bilgi vermez. Tam bilgi edinmek için nesnenin altı yöndeki projeksiyonlarına bakmak gerekmektedir.

Ortographic projeksiyonlar başta makina parçaları ve binalar olmak üzere bunların ön cephe, üst ve kenar görüntülerinin elde edilmesi için mühendislik çizimlerinde sıkca kullanılırlar.

Bu projeksiyonda z koordinatları sıfırlanmaktadır. Yani diğer bir deyişle tüm x,y noktaları z=0 projeksiyon düzlemi üzerinde kalmaktadır. Üç boyutlu bir nesnenin ortogonal projeksiyonunu oluşturmak için dönüşüm matrisi nesnenin tüm noktalarına uygulanır. Bu dönüşüm işleminin yapılmasından sonra projeksiyon görüntüsünün ekranda çizimi için sadece x ve y kordinatları kullanılır.

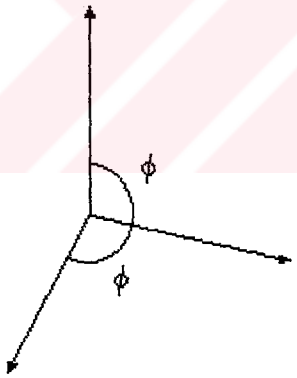
4.1.2 Axonometric paralel projeksiyon

Axonometric paralel projeksiyonda birbirine paralel olan projeksiyon çizgileri projeksiyon düzlemine dik olmayan açı yaparak gelirler. Bu yüzden, bu projeksiyon yardımıyla nesnenin bir anda birden çok farklı yüzleri görüntülenebilir. Elde edilen projeksiyon görüntüsünde nesneyi oluşturan doğruların paralelliği korunmaktadır, fakat açılar değişmektedir.

Axonometric projeksiyonlar, determinatı 0 olan bir dönüşüm matrisi ile üretilir. Geometri ve mühendislik çizimlerin tanımlanmasında (elektronik ve makina müh.) uygulama alanı bulan axonometric projeksiyonların çeşitli tipleri vardır. Bunlardan önemli olanları şunlardır:

Trimetrik Projeksiyon : Dönüşüm matrisi, eksenlerden birinin 90° 'lik dönüşünü belirtir. Projeksiyon işlemi sonucunda eksenler dik kalır. Burada elde edilen projeksiyon görüntüleri Ortogonal projeksiyonlarla aynıdır. Mühendislik çizimlerinde yardımcı görüntüler oluşturmak için kullanılır.

Dimetrik Projeksiyon : Bu projeksiyonda nesnenin şekli, üç ana eksenden ikisinin yönünde eşit oranda kısaltılır. Bu durumda koordinat eksenleri şekil 4.1'deki gibi gösterilebilir.

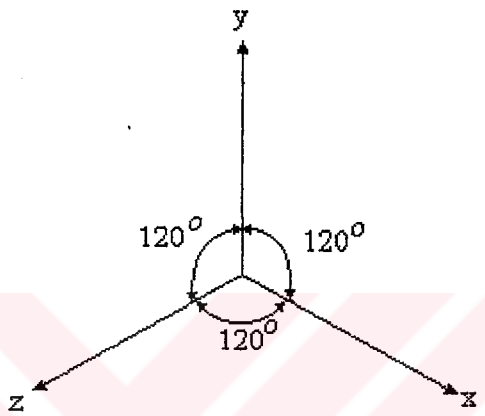


Şekil 4.1 - Dimetrik projeksiyon koordinat ekseni.

İzometrik Projeksiyon : Bu projeksiyonda nesnenin şekli, üç ana eksen yönünde eşit oranda kısaltılır. Projeksiyonun yönü üç ana koordianat eksenleri ile eşit açı yapar. Bu durumda koordinat eksenlerinin görünümü şekil 4.2'deki gibidir.

Bu üç projeksiyondan her biri, birbirine dik olan koordinat eksenlerinin

döndürülmesiyle nesnenin farklı yönden görüntülenmesine dayanır. Döndürme işleminin sırasıyla hem x ekseni hem de y ekseni olmak üzere iki eksen etrafında yapılması yeterlidir.



Şekil 4.2 - İzometrik projeksiyon koordinat ekseni.

4.1.3 Oblique paralel projeksiyon

Oblique paralel projeksiyon, ortographic projeksiyonun ön-cephe, üst ve kenar projeksiyon özelliklerinin, axonometrik projeksiyon özellikleri ile birleştirilmiş bir halidir. Bu projeksiyonda, projeksiyon düzlemi koordinat sistemini ana eksenlerinden birine diktir. Bunun sonucu olarak, projeksiyon düzlemine paralel olan nesnenin projeksiyon görüntüsünden açılar ve uzaklıklar gerçek değerleri ile doğrudan ölçülebilir. Bu sebepten oblique projeksiyonlar, mimarlar tarafından sıkca kullanılırlar. Oblique projeksiyonlarda, projeksiyon yönü projeksiyon düzlemine dik değildir.

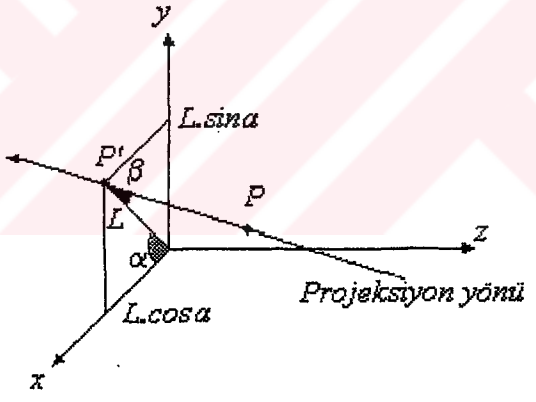
(xp, yp, zp) noktasından orjine olan doğru projeksiyon yönü olarak kabul edilirse z=0 düzlemi üzerinde projeksiyon görüntüsünü oluşturmak için

$$x' = x + z \,.(-x_p / z_p)$$

$$y' = y + z \,.(-y_p / z_p)$$

denklemleri kullanılır. Burada x, y ve z projeksiyon işlemine tabi tutulan nesne koordinatını, x', y', z' ise projeksiyon sonucunda elde edilmiş koordinatı belirtirler.

Oblique projeksiyon denklemi diğer taraftan yaygın olarak projeksiyon düzlemindeki uzunluklar ve açılar cinsinde de gösterilir. Şekil 4.3'de projeksiyon yönünün projeksiyon düzlemi ile β açısı yapması durumunda P noktasının z=0 düzlemine oblique projeksiyonu sonucunda elde edilen P' noktası gösterilmektedir.



Şekil 4.3 - Oblique projeksiyonu.

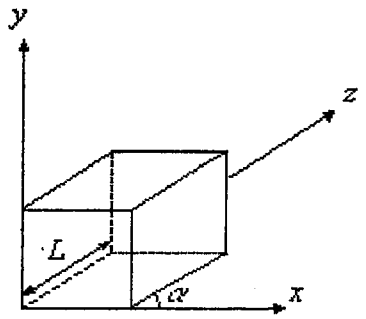
Bu durumda oblique denklemi aşağıdaki gibi verilir.

$$x' = x + z \,.L.\cos\alpha$$

$$y' = y + z.L.\sin\alpha$$

Bir küpün 'oblique' projeksiyon ile elde edilmiş görüntüsü L uzaklığı ve α

açısına bağlı olarak aşağıdaki şekilde gösterilmiştir. Burada L ve α değerleri projeksiyon işleminden önce belirlenmelidir.



Şekil 4.4 - Bir küpün oblique projeksiyonu ile elde edilmiş görüntüsü

Oblique projeksiyonların önemli olan tipleri:

Cavalier Oblique projeksiyon,

Cabinet Oblique projeksiyon.

Cavalier Oblique Projeksiyon : Cavalier projeksiyonda, projeksiyon yönü projeksiyon düzlemi ile $\beta = 45°$ açı yapar. Bunun sonucunda projeksiyon düzlemine dik olan doğrular projeksiyon düzlemine gerçek uzunlukları ile izdüşürülür. Bu yüzden projeksiyon düzleminden gerçek değerleri ile uzunluk ölçümleri yapılabilir.

Cavalier projeksiyonu için verilmiş olan oblique dönüşüm denkleminde

$$L = 1, \qquad \beta=45°$$

değerleri kullanılır.

Cabinet Projeksiyon : Cabinet projeksiyonda, projeksiyon yönü projeksiyon düzlemi ile β=ArcCot(1/2) açı yapar, öyle ki nesneyi oluşturan doğrulardan projeksiyon düzlemine dik olanlar projeksiyon düzlemine gerçek uzunluklarının yarısı şeklinde iz düşürülür. Görsel olarak, cabinet projeksiyonlar cavalier projeksiyonlardan daha gerçekçidir.

Cabanet projeksiyon için yukarıda verilen Oblique dönüşüm denkleminde

$L=1/2$ β=ArcTan(2) veya yaklaşık β= 63.4°

değerleri kullanılır.

Eğer L= 0 ve α = 0° olarak alınırsa sonuçta Ortographic projeksiyon elde edilir.

4.2. Perspektif Projeksiyonlar

Perspektif projeksiyonlarda, projeksiyon doğruları birbirine paralel değildir. Bunlar projeksiyon merkezi olarak adlandırılan bir noktadan çıkarılır. Projeksiyon merkezi tek ise bu projeksiyona tek noktalı, iki tane ise iki noktalı, üç tane ise üç noktalı perspektif projeksiyon denir. Burada sadece tek noktalı projeksiyon tipi ele alınacaktır.

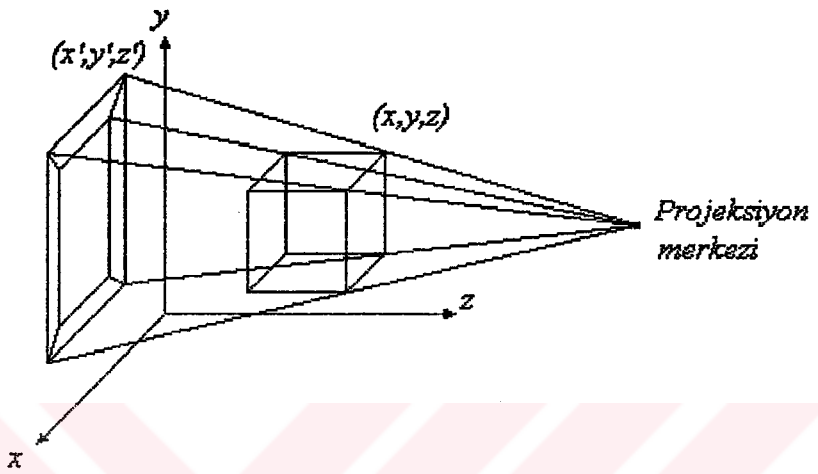
Tek noktalı projeksiyonda, projeksiyon düzleminde nesnenin iz düşümünü oluşturmak için bir projeksiyon merkezi kulanılır.

Projeksiyon merkezi (x_c, y_c, z_c) noktasında bulunuyorsa ve nesnenin projeksiyon edilecek noktası (x, y, z) ise dönüşüm denklemleri olarak aşağıdaki denklemler kullanılır.

$$x' = (x_c z - z_c x) \ / (z - z_c)$$

$$y' = (y_c z - z_c y) \ / (z - z_c)$$

Bu denklemleri kullanarak nesnenin tüm noktaları için işlem yapılarak nesnenin z=0 düzlemi üzerine perspektif görüntüsü elde edilir (Şekil 4.5).



Şekil 4.5 - Perspektif Projeksiyon.

Projeksiyon merkezinden ya da bakış noktasından çıkan görüş hatları nesneyi oluşturan köşelerden geçerek z=0 düzlemi üzerinde izdüşümü alınır. Tek noktalı perspektif projeksiyon yöntemi ile küpün yakın kenarları daha büyük uzak kenarları daha küçük görülür. Perspektif projeksiyon yönteminde görüş uzaklığına bağlı olarak nesnenin derinlik mesafesinin değiştiği görülür. Böylece nesnenin üç boyutlu görüntüsünün benzeri bir perde üzerine aktarılmış olur. Üç boyutlu nesnenin görüntüsü iki boyutlu görüntüye çevrilir.

Üç boyutlu bir nesnenin hareketlerinin bilgisayar ekranında bir pencere içine alınması bir problemdir. Gerçek hayattaki kameranın taklidi ile bir gerçek olmayan kamera kullanılır. Bu terim "gerçek olmayan kamera" Üç boyutlu alanların bilgisayar ekranına, iki boyutlu izdüşümünü sağlayan bütün gerekli dönüşümlerin toplandığı bir program parçasıdır ve üç boyutlu bilgisayar grafiklerinde çok sık kullanılır.

5. TURBO PASCAL İLE ÜÇ BOYUTLU NESNELERİN SİMULASYONU

5.1 Program Ana Bloğu

Program ana bloğu her bir özel test veya hesaplamaları takip eden kontrol yönetim ve adımlar dizisidir. Bir Turbo Pascal Programı ana blok adını verdiğimiz üç bölümden oluşur.

- Program başlığı bölümü

- Veri bölümü

- Program bölümü

5.1.1 Program başlığı bölümü

Program başlığı bölümünde, Turbo Pascal programının ilk satırı derleyici deyimleri ve program ismini içerir. Derleyici deyimlerinin yazılıp yazılmaması kullanıcının isteğine bağlıdır. Turbo Pascal derleyicisi, programın daha kolay çalıştırılmasını ve daha kolay hata ayıklama işlemi gerçekleştirilmesini sağlamak amacıyla {$E +}, {$N -} gibi çeşitli derleyici deyimlerini içerir. Programın ikinci satırı program ismidir. Bu satırın yazılıp yazılmamasıda kullanıcının isteğine bağlıdır. Program isminin kullanılıp kullanılmamasının program işleyişine bir etkisi yoktur. Program ismi programın genel yapısı hakkında bilgi vermek için kullanılabilir ("Program Simulation" vb.).

5.1.2 Veri bölümü

Veri bölümünde, programın işleyişi sırasında kullanılacak bütün isimlerin tanım-

lamaları yapılır. Farklı amaçlar için kullanılan bu isimler, farklı tanımlama blokları içinde gösterilir. Bu tanımlama blokları,

- Etiket tanımlama bloğu (Label),

- Sabit tanımlama bloğu (Const),

- Tip tanımlama bloğu (Type),

- Değişken tanımlama bloğu (Var)

şeklinde adlandırılır.

Etiket tanımlama bloğu : Etiketler, GOTO deyimiyle birlikte program içindeki akış yönünü değiştirmek için kullanılır. GOTO deyimiyle gönderilen adresin aynı program bloğu içinde bulunması gereklidir.

Sabit tanımlama bloğu : Programda kullanılacak sabit değerlerin tanımlanan bir isim altında kullanılmasını sağlayan bloktur. Program içinde kullanılan çok sayıda sabit değer programın anlaşılabilirliğini azaltmaktadır. Sabit değer değiştirilmek istendiğinde de problem olmaktadır. CONST deyiminde bir sabit ismi ile tanımlanan değerler bu problemi ortadan kaldırmaktadır.

Tip tanımlama bloğu : Program içinde kullanılacak her diziye bir tanımlama adı verilmesi gerekir. Diziye verilecek tanımlama adı değişken tanımlama kurallarına uygun olmalıdır. Kullanılacak tip Turbo Pascal programlama diline ait bir tip olabileceği gibi, kullanıcının tanımlayacağı özel bir veri tipide olabilir.

Değişken tanımlama bloğu : Programda kullanılacak bütün değişkenlerin önceden tanımlandığı bloktur. Bir değişken hafızanın isimlendirilmiş ve ayrılmış

bir ya da bir kaç gözüdür. Bu tanımlama sırasında değişkenin tipi kesinlikle verilmelidir. Programda kullanılacak değişkenlerin tipi tamsayı (integer), gerçel (real) sayı, mantıksal (boolean), karakter (char) ya da karakter dizi (string) formlarında tanımlanabilir.

5.1.3 Program bölümü

Bir Turbo Pascal programının üçüncü ana parçası program bölümüdür. Bu bölümde program içinde bulunan tüm işletilebilir deyimler yer alır. Bu bölüm,

- fonksiyon tanımlama bloğu (Function),
- işlem tanımlama bloğu (Procedure),
- ana program bloğu (Begin / End.)

adı verilen üç bloktan oluşur.

Fonsiyon tanılama bloğu : Fonsiyon tanımlama blokları, birden fazla değeri alıp bu değerleri işleyen ve sadece tek bir sonuç döndüren işlevsel operatörlerdir. Bu sebeble fonksiyonun dönüş tipi belirtilir. Bir fonksiyon ismi program içinde tek başına kullanılamaz. Fonksiyon isimleri bir işlem yada komut içinde kullanılabilir.

İşlem tanımlama bloğu : İşlem tanımlama blokları, ana programdan gelen bir yada daha fazla değeri alıp bu değerleri işleyen ve birden fazla sonuç döndürebilen alt programlardır. Alt program, ana program tarafından kullanılan ve kendi başına bağımsız işlem yapabilen program parçasıdır. Procedure kelimesi bir alt programın başladığını bildirir. Begin - End; deyimleri arasında yazılan deyimler alt programın deyimleridir.

Ana program bloğu : Bu blok BEGIN kelimesiyle başlayıp END. deyimiyle sona eren Turbo Pascal programlama dili deyimleri topluluğudur. Yukarıda açıklanan bloklardan ana program bloğunun kullanılması zorunlu olup, diğer blokların kullanılması zorunlu değildir.

5.2. Grafik Sistemi

InitGraph komutu ile bilgisayarın grafik sisteminin adaptör ve modu belirlenir. Bu komut ile GraphDriver ve GraphMode değişkenleri grafik sisteminde bulunan adaptör ve moduna ait bir tamsayı değer alırlar. Bu bilgileri içeren uygun grafik arabirim dosyasını hafızaya yükleyerek grafik sistemini başlatır.

Borland Grafik Paketi tarafından desteklenen tüm grafik adaptör ve modlarında programın çalışması için grafik adaptör ve modunun program tarafından otomatik olarak belirlenmesi gerekir. Bu işlem DetectGraph komutu tarafından gerçekleştirilir.

Bu işlemler sırasında meydana gelebilecek hatalar GraphResult komutu ile kontrol edilir. GraphResult komutu hata olduğu yada olmadığı zaman bir tamsayı üretir. Bu değerin sıfır olması durumunda hata yok, negatif olması durumunda ise bir hata meydana geldi demektir.

Grafik ekranla ilgili çalışmalar bittiğinde grafik ekrandan metin ekranına geçilmesi veya grafik sisteminin kapatılması gerekir. Bu işlem için CloseGraph komutu kullanılır.

5.3 Görüntü Prametreleri

Nesneler, vd görüş uzaklığı, θ ve ϕ bakış açısından gözlemlenir. Dikdörtgen koordinat sisteminde görüş noktasının konumu vd uzaklığı, θ ve ϕ

açısının değerleri fare yardımı ile değiştirilerek üç boyutlu nesnenin ekranda hareketi sağlanır.

Ekran projeksiyon konumu vd=500 'den başlatılmıştır. Bakış açısının yönü ve konumu vd,θ,ϕ ile kontrol edilmiştir.

5.4. Köşe Görüş ve Ekran Koordinat Boyutları

I nesnesinin köşe nokta sayısı NV[I] olarak tanımlanır. Bir küp için sekiz köşe noktası, bir pramit için ise dört köşe noktası kullanılır. Görüş alanımızın sınırları K1<=z_e<=K2 arasındaki z_e değerleri ile belirlenir. K1 ve K2 değerleri istenilen şekilde seçilebilir.

Tüm ekranı içine alması için, önce herbir köşenin en büyük ve en küçük z_e değerini bularak z_e koordinatı hesaplanır. Böylece K1=7 ve K2=10 değerleri ile tüm ekranın kullanımı sağlanabilir.

5.4.1 Görüş alanı değişimi

Bakış açısının ve yönünün değişiminde dünya koordinat sisteminden görüş koordinat sistemine dönüşüm ihtiyacı duyulur. Bir noktaya bakış yönünün değişmesi koordinat sisteminin değişmesi ile mümkün olur. Çünkü bir noktanın fiziksel konumu sabittir. Fakat bizim görüş noktamızdan yada projeksiyon merkezinden görüş koordinat sistemi referans alınır. Bir noktanın taşınmasında dört dönüşüme ihtiyaç duyulur ki bu ekranda hissedilebilsin.

Küresel koordinatlarda (vd,θ,ϕ) ile teşhis edilen nokta, görüş noktası yada projeksiyon merkezidir. Küresel koordinatlarda,

$$x = vd.\sin\phi.\cos\theta$$

$y = vd.\sin\phi.\sin\theta$

$z = vd.\cos\phi$

değerleri (vd,θ,ϕ) ile belirlenir (Bkz. Şekil 2.2).

Aşağıda gösterildiği şekilde değişimler yapıldığında,

1.Adım : Projeksiyon merkezini değiştirerek görüş noktasında yeni bir koordinat sistemi kurmak için gerekli dönüşüm matrisi

$$T_1=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -vd\sin\phi\cos\theta & -vd\sin\phi\sin\theta & -vd\cos\theta & 1 \end{bmatrix}$$

şeklindedir.

2.Adım : z ekseni etrafında saat ibresi yönünde xy düzlemini θ açısı kadar döndürmek için,

$$T_2=\begin{bmatrix} \sin\theta & \cos\theta & 0 & 0 \\ -\cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

dönüşüm matrisi kullanılır.

3.Adım : eksen sistemini x ekseni etrafında saat ibresinin ters yönünde ϕ açısı kadar döndürmek için gerekli olan değişim matrisi,

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & -\cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklindedir.

4.Adım : Nesnenin boyutlarının değiştirilmeden yansıtma dönüşümünün yerine getirilmesi için x koordinat değerleri değiştirilir. Aslında x değerinin işareti değiştirilerek yz düzlemine göre bir yansıtma için kullanılan, dönüşüm matrisi işlemi

$$T_4 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

şeklindedir.

Bu dört basit değişimin birleştirilmesi ile meydana gelen T dönüşüm matrisi, şu dur. ($T = T_1.T_2.T_3.T_4$)

$$T = \begin{bmatrix} -\sin\theta & -\cos\theta\cos\phi & -\cos\theta\sin\phi & 0 \\ \cos\theta & -\sin\theta\cos\phi & -\sin\theta\sin\phi & 0 \\ 0 & \sin\phi & -\cos\phi & 0 \\ 0 & 0 & D & 1 \end{bmatrix}$$

Böylece (x,y,z) noktalarının dünya koordinatlarını bildiğimizde, görüş koordinatları (x_e,y_e,z_e) değerlerini,

$$(x_e,y_e,z_e,1) = (x,y,z,1).T$$

veya

$$x_e = -x.\sin\theta + y.\cos\theta$$

$$y_e = -x.\cos\theta.\cos\phi - y.\sin\theta.\cos\phi + z.\sin\phi$$

$$z_e = -x.\cos\theta.\sin\phi - y.\sin\theta.\sin\phi - z.\cos\phi + D$$

eşitlikleri ile bulunabilir.

Görüş koordinatlarını bulduğumuzda, aşağıdaki eşitliklerden ekran koordinatları basit bir işlem ile hesaplanır.

$$x_s = (vd / s) . (x_e / z_e)$$

$$y_s = (vd / s) . (y_e / z_e)$$

5.5 Yüzey

Nesnelerin yüzey tanımlamalarının yapıldığı bloktur. Mesela bir küpün altı yüzeyi, bir pramitin ise dört yüzeyi bulunur. Bu programda kullanılan iki nesneden binanın altı ve çatının 5 yüzeyi vardır. Yüzey boyutları SF[I, J, K] isimli değişkende I nesne sayısını , J yüzey sayısını, K ise her bir J yüzeyinin bağlantı köşelerinin sayısını belirtir. Binanın her bir yüzeyi dört köşe ile tanımlanır. Çatının üst yüzeyleri üç köşe ile alt kısmı ise dört köşe ile tanımlanır. Tanımlanan I nesnesinin J yüzeyinin tanımlanan köşelerinin sayısı NPS[I, J] dir. I nesnesinin J yüzeyinin köşeleri SF[I, J, NPS[I, J]] içinden tanımlanır. Nesnenin dışarıdan göründüğü şekliyle her bir yüzeyinin etrafında dizilen noktaları saat ibresinin ters yönünde listelemek çok önemlidir. NS[i] isimli değişkenle yüzey sayısı belirtilir. Her yüzey için NPS[I, J] isimli değişkene tanımlanması gereken nokta sayısı verilir. Buna bağlı olarak SF[I, J, K] isimli

değişkende, nesnenin numaralanmış noktaları her bir yüzey için saat istikametinin ters yönünde sıralanır.

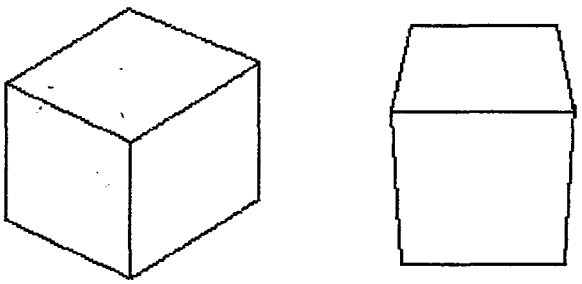
5.6 Görünürlük Testi

Şimdi, nesnelerin her bir yüzeyi görünürlük testi uygulanmaya hazırdır. Her bir yüzey için normal yüzeyin başlama noktasından görüş noktasına kadar *l* uzunluğunda bir görüş vektörü seçilir. Sonra *n.l* çarpımı hesaplanır ve bu çarpımın işareti belirlenir. Her I nesnesinin J yüzeyinin görünürlük durumunu saklanır. Eğer görünürse vsf[i, j] = 1'dir, değilse vsf[i, j] = 0'dır.

5.7 Görünür Kenar Testi

Kenar boyutları E[i, j, k] olarak tanımlanır. İlk boyut elemanı I nesne olarak tanımlanır. Her I için boyut elemanı J nesnede kenar sayısı olarak tanımlanır. Her J için, K elemanı dört bilgi içerecektir. Bunlar kenarın son noktaları (K= 1, 2), bir sınır olan kenar için görülen yüzeylerin sayısı (K= 3) ve kenara ait olan yüzey numarasıdır (K= 4). Bir küpün 8 köşesi, 12 kenarı ve 6 yüzeyi vardır. Her yüz çokgen sınırlı bir düz yüzeydir. Bir bakış noktasından küpe bakıldığında sadece iki veya üç yüzeyi görülebilir (Şekil 5.1). Diğer yüzeyler görüş alanından uzaktır.

Gizli çizgileri ve küp yüzeylerini çizmemek için gizli çizgileri görünür çizgilerden ayırt eden bir görünürlük testi gerekir. Görünürlük testinde her yüzey, yüzey normal vektörü ve görüş vektörü ile birleştirilen iki yönelme vektörü görülür. *l* görüş vektörü, görüş noktasından geçen çizgiyi, *n* yüzeye dik ve dışarıya doğru bir vektör, yüzey normal vektörünü tanımlar. *n* ve *l* arasındaki açı hesaplanır. Bu açı 0°-90° arasında ise yüzey görünürdür ve gösterilir. Eğer bu açı 90° büyükse yüzey bakış açısından uzak yüzeydir. Bu yüzey gözükmez ve gösterilmez.



(a) (b)

Şekil 5.1 - Bir küpün görünen yüzeylerinin sayısı : (a) üç yanı görünür ve (b) sadece iki yanı görünür.

5.7.1. Yüzey-normal vektörü

Yüzeyle birleştirilen normal vektörü hesaplamak için küpün her bir yüzeyinin saatin ters yönündeki kenarlarının sayısı gerekmektedir. Verilen bir yüzey için başlama noktası keyfidir fakat yüzeyin yüzüne, küpün dışarıdan bakıldığı gibi kalan köşelerinin saatin ters yönünde numaralandırılması önemlidir. Şekil 5.2'deki gibi kenarlar numaralandırıldıktan sonraki adım 1 numaralı köşeden 2 numaralı köşeye yönelen u vektörünü ve 1 numaralı köşeden 3 numaralı köşeye yönelen v vektörünü teşhis etmektir. u x v yüzeye dik ve dışa doğrudur. (Eğer noktalar saat yönünde tanımlanır ise n = u x v vektörü içe doğrudur.) Lineer cebirde tanımlanan iki vektörün çarpımı,

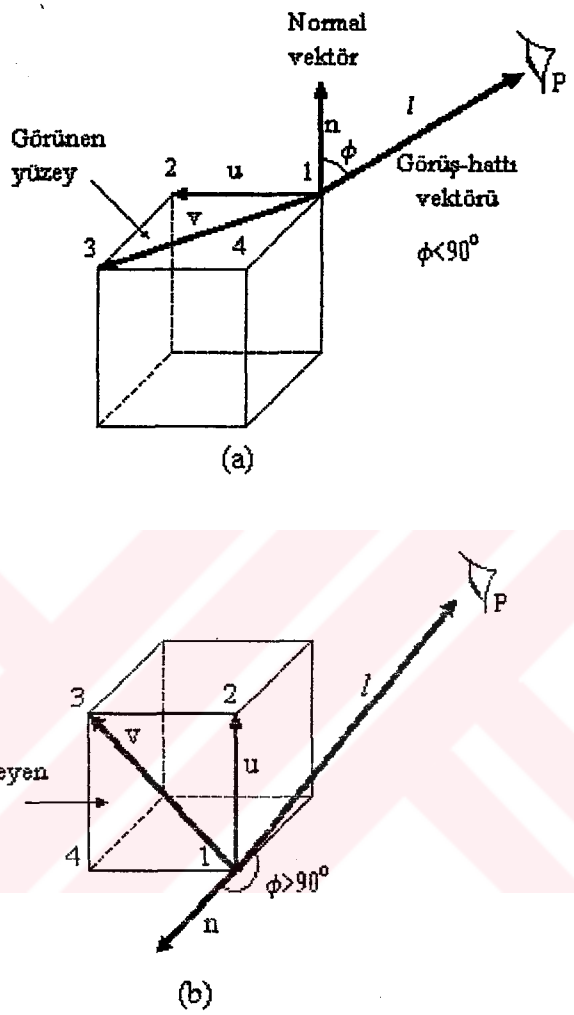
$$n = u \times v$$

$$n = (bf - ce, cd - af, ae - bd)$$

buradan

$$u = (a, b, c) \quad ve \quad v = (d, e, f)$$

küpün ön yüzeyinin köşeleri tanımlanır.



Şekil 5.2 - Görünürlüğün tanımlanmasında normal ve görüş hattı vektörleri.

5.7.2. Görüş-hattı vektörü

Nesnenin yüzeyinde herhangi bir nokta için P(vd, θ, ϕ) görüş noktasına uzatılan bir çizgi ile görüş hattı vektörü hesaplanır. Kolaylık için yüzey normal vektörünün başlama noktası seçilir. Küresel koordinatdaki görüş noktası tanımlanır. Sonra,

$$l = (vd.\sin\phi.\cos\theta,\ vd.\sin\phi.\sin\theta,\ vd.\cos\phi) - (x_1,y_1,z_1)$$

Görüş hattı vektörü hesaplanır. Burada (x_1,y_1,z_1) birinci köşenin koordinatlarıdır.

Görünürlük testi : n ve l vektörlerinin çarpımı,

$$n.l = (n_1,n_2,n_3) \,.\, (l_1,l_2,l_3) = n_1.l_1 + n_2.l_2 + n_3.l_3$$

denklemi ile tanımlanır. Bu denklem lineer cebirde şöyle bilinir.

$$n.l = |n| \,.\, |l| \,.\, \cos\phi$$

$$\phi = \text{arc}\cos(\ (n.l)\ /\ (|n| \,.\, |l|),$$

burada

$|n| = n$ 'in uzunluğu

$|l| = l$ 'nin uzunluğu

$\phi = n$ ve l arasındaki açı

Bir görünen yüzey için ϕ, 0° ile 90° arasında ve $n.l > 0$ olacaktır. Bir görünmez yüzey için ise ϕ 90° ile 180° arasında ve $n.l < 0$ olacaktır. Her yüzey için görüş çizgisi vektörü l vektörünün koordinatlarını hesaplamak için LX, LY ve LZ değişkenleri bulunur(Bkz. Şekil 5.2). Normal procedure alt programında bulunan N normal vektörleri ile görüş hattı vektörleri,

```
T= n[i, j, 1]*Lx + n[i, j, 2]*Ly + n[i, j, 3]*Lz
```

denklemi ile çarpılır. T<0 ise görünen yüzey değişkeni vsf[i, j]=0 olur. T>0 ise vsf[i, j]=1 değerini alır.

5.8 Minimax Test

Her bir nesne için x koordinatının ve y koordinatının maksimum ve minimum değerleri bulunur. Bu değerlerin kontrolu ile iki nesnenin üst üste gelip gelmediği kontrol edilir. Üst üste gelme yoksa overlap = 0, varsa overlap = 1 değerini alır.

5.9 Çizim

Eğer minimax testinde iki nesne arasında üst üste gelme imkanı olmazsa basitce iki nesne çizilir. Üst üste gelme halinde arkada kalan nesnenin, uygun bitiş noktalarının tanımlanmasıyla görünen kenarları çizilir. Bu işlem Plot Procedure alt programında, ekranın tx ve ty konum değerlerine bağlı olarak hesaplanan, çizginin başlama ve bitiş koordinatlarına göre,

```
Line(aa,bb,cc,dd)
```

komutu ile nesneler ekrana çizilir.

5.10 Nesnenin Üstünlüğü

İki nesne arasında üst üste gelme durumu olduğunda nesnenin üstünlüğü işlemi yapılır. Nesnelerin uzak olanları yakın olanlara tercih edilir. Bu yüzden nesnelerin ilgili pozisyonları sınıflandırılır. Burada esas olan bakana hangi nesnenin daha yakın göründüğünü tanımlamaktır. Yakın olan nesne tercihli nesne olarak tanımlanır. Önce nesnelerin biribirini kesip kesmediği kontrol edilir.

Kesişim Testi Uygulanır. Bir kesişim varsa bakana hangi nesnenin daha yakın olduğunu belirtmek için Derinlik Testi uygulanır. Öncelikli nesnenin görülebilen kenarları çizilir. İkinci öncelikli nesnenin kenarları baskın geleceğinden öncelikli nesne tarafından gizlenmesi mümkün olmayacaktır. İki nesne arasında kesişim yoksa Kapsama Testi uygulanır. Kapsama Testi bir nesnenin diğer bir nesneyi tamamen örttüğünü gösterir. En küçük z değeri ile nesnelerin öncelik sırası belirlenir. İkinci öncelikli nesne öncelikli nesnenin her şeyini kapsarsa öncelikli nesne tamamen saklı olur ve çizilmez. Eğer durum ters çevrilirse ikinci nesnenin dış kenarları önce çizilir ve iç kenarlar sonradan kontrol edilir. Sadece görünen iç kenarlar çizilir.

5.11 Kesişim Testi

Bir nesnenin görülebilen her kenarı diğer görüntüdeki kenarlarla kesişim elde etmek için kıyaslanır. Önce hangi çift kenarların kıyaslanacağı tanımlanır. Kenarlar için yapılan minimax testi o iki kenarın kesişip kesişmemesi işleminin tanımlanmasını hızlandırır. Kesişim varsa kesişim noktasının hesabı yapılır ve NCUT = 1 değerini alır. Sonra, NCUT = 1 olduğundan Derinlik Testi yapılır. Derinlik değeri z_S küçük olan nesne bakana daha yakındır. Bu yüzden bu nesnenin önceliği vardır.

5.11.1 Kenar bitiş noktalarının görünürlük durumu

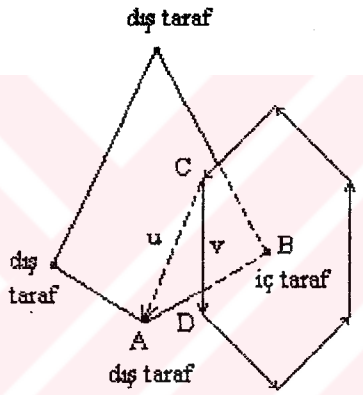
Kenar bitiş noktalarının görünürlük durumu üç şekilde olur.

1. Tamamen görünür.

2. Kısmen görünür.

3. Hiç görünmez.

Yüzey boyutlarının bilgileri saat istikametinin ters yönünde verildiğinden nesnenin dış kısımlarıda saat istikametinin ters yönünde olur. Arkadaki nesnenin yarım görünen bitiş noktası öndeki nesnenin her kenarı ile kıyaslanır. Örnek olarak Şekil 5.3 verilmiştir. Öndeki nesnenin her CD noktasındaki kenar için iki vektör hesaplanır. Vektör *v* C'den D'ye uzanır, vektör *u* C'den A'ya uzanır. Nokta test edilir. A noktası öndeki nesnenin sağ yarım düzleminde ise EndPoint(J) = 1 değerini alır ve test işlemi diğer nokta için devam eder. Eğer *J* bitiş noktası sol yarım düzlemde öndeki nesnenin kenarlarının dışında ise EndPoint(J) = -1 değerini alır.



Şekil 5.3 - Kenar bitiş noktalarının görünürlük durumu.

5.12 İkinci Öncelikli Nesnenin Görünen Kenarlarının Çizimi

İkinci öncelikli nesnenin, her bir kenar noktasının görünürlük durumu bir kez tanımlandığından, görünen kenarları çizime hazırdır. Nesne tam olarak çizilir.

6. SONUÇLAR

1) Üç boyutlu dünya koordinat sisteminde noktaların tanımlanması ile üç boyutlu geometrik şekiller oluşturulur.

2) Nesnenin yüzeylerini oluşturan noktaların, her bir yüzey için saat istikametinin ters yönünde tanımlanması önemlidir.

3) Ekran iki boyutlu olduğundan üç boyutlu nesnelerin ekran düzlemi üzerine izdüşümü yapılır.

4) Nesnenin hareketi, nesneye olan uzaklık vd ve bakış açısı θ ve ϕ değerleri değiştirilerek yapılır. Nesneyi oluşturan noktaların yeni konumu dönüşüm formülleriyle hesaplanır. Nesnenin eski konumunda taban renginde çizim yaptırılarak, nesnenin yeni konumunun temiz bir görütüsü elde edilir ve nesnenin sanal ortamda hareketi sağlanır.

5) Nesnelerin katı görüntüsünün elde edilmesi için Görünürlük Testi, birbirleriyle çakışma durumunda ise Kesişim Testi, Derinlik Testi ve Kapsama Testi uygulanarak nesneleri oluşturan kenarların görünürlük durumu hesaplanır. Kenarların görünen kısımları çizilir.

7. ÖNERİLER

1) Sanal gerçeklik (virtual reality) alanında yapılan çalışmalar henüz yeni olduğundan yapılan üretim az ve pahalıdır. Özellikle ABD ve Japonya bu alanda yapılan çalışmalara öncülük etmekte ve en büyük yatırımı Japonya yapmaktadır. Türkiye'de henüz yazılım alanında çalışmalar yenidir. Bu alanda yapılacak yatırım ve çalışmalar ülkemiz için önemli bir iş sektörü olabilir.

2) Bu alanda yapılacak çalışmalarda grup çalışmalarına önem verilmeli programcılar, grafikerler, mimarlar, mühendisler ve matematikçilerin oluşturduğu bir grup ve iyi bir bilgisayar donanımı ile çok daha kapsamlı ve ileriye dönük çalışmalar yapılmalıdır.

3) Türkiye'de bir grafik külübü kurulmalı, bu alandaki yayınlar arttırılmalı ve yapılacak akademik çalışmalara önem verilmelidir.

4) Bu konuda araştırma yapacak olanlar, öncelikle üç boyutlu geometrik modeller, koordinat sistemleri, geometri matematiği üzerinde bilgi sahibi olmalıdır. Bilgisayar grafik ünitesi ve grafik komutlarının işlemleri üzerinde yeterli bilgisi olmalıdır.

KAYNAKLAR DİZİNİ

Newman, M.N. and Sproul, R.F., 1985, Principles of Interactive Computer Graphics, 293-366 p.

Foley, J.D. and Van Dam, A., 1984, Fundamentals of Interactive Computer Graphics, 267-318, 505-535 p.

Erdun, H., 1993, Turbo ve Borland C & Pascal ile Grafik, 321-354 s.

Rankin, J.R., 1992, Computer Graphics Software Construction Using the Pascal Language, 255-351 p.

Vince, J., 1992, 3D Computer animation, 50-53 p.

Donalt, H., 1994, Computer Graphics, 164-169 p

Bradberry, J., 1993, Computer Graphics Environments, 78-82 p.

Journal, 1993, I.E.E.E. Computer Graphics in Application, Bikent University Library, 56 p.

Ezzel, B., 1991, Graphics Programming in Turbo Pascal, 73-119 p.

Foley, J.D., 1990, Computer Graphics : principles and practice, 431-445 p.

Schulz, G.B. and Schulz, Ch., 1990, 3D Graphics in Pascal, 169 p.

SANAL GERÇEKLİK ORTAMININ
PC'DE SİMULASYONU

EKLER

Ek-1. Turbo Pascal Programlama Dilinde Simulasyon Programı
Ek-2. θ=40°, ϕ=80° ve *vd*=600 için nesnelerin görünüşü
Ek-3. θ=45°, ϕ=50° ve *vd*=250 için nesnelerin görünüşü
Ek-4. θ=45°, ϕ=35° ve *vd*=500 için nesnelerin görünüşü
Ek-5. θ=30°, ϕ=10° ve *vd*=400 için nesnelerin görünüşü

Şubat - 1996

```
Program Three_Dimension_Simulation;
uses
        graph,crt,drivers;
        {grafik, ekran ve sürücüler üniteleri tanımlanır.
        Sürücüler ünitesi ile mouse kullanımı sağlanmış olur.}
label 20,21;
const
        fon     : FillPatternType = ($FF, $FF, $FF, $FF, $FF, $FF, $FF);
var
gd,gm           : integer;
  v,ec          : array [1..2,1..8,1..3] of real;
  sc            : array [1..2,1..8,1..3] of real;
  sf            : array [1..2,1..6,1..5] of byte;
  nps           : array [1..2,1..6] of byte;
  vsf           : array [1..2,1..6] of shortint;
  ns            : array [1..2] of byte;
  n             : array [1..2,1..6,1..3] of real;
  e             : array [1..2,1..12,1..4] of byte;
  nve           : array [1..2] of shortint;
  nv            : array [1..2] of byte;
  endpoint      : array [1..12] of shortint;
  ix,iy         : array [1..2] of integer;
  xmx,xmn,ymx,ymn,zmx,zmn  : array[1..2] of real;
  i,j,k,l,ii,jj,kk,ll,ij,q : byte;
  x,y,z,xe,ye,ze,xs,ys,zs,d,vd,theta,phi,mxx,myy,scf,s,sn1,sn2,cn1,cn2,w : real ;
  tx,ty,vx,vy,k1,k2,x1,x2,y1,y2,test       : integer;
  overlap,ncut,aa,bb,cc,dd : integer;
  ax,ay,bx,by,cx,cy,dx,dy,laa,lbb,lcc,ldd  : real;
  a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3,xi,yi,v1,v2,u1,u2 : real;
  vi : shortint;
  tcut,kkk : byte;
```

```
  xm,ym    : integer;
  z0,z1,z2,z3,r1,r2,r3,r4,m1,m2,m3 :real;
  s1,s2,s3,s4 : shortint;
  dt1,dt2,xx,yy,zz,t1,t2 : real;
  aaa,bbb,ccc,ddd,eee,fff : real;
  CommandXMin, CommandXMax,
  CommandYMin, CommandYMax : Integer;
  start : Boolean;

procedure v_endpoint;
{Görülen son nokta}
label 14,15;
begin
  for j:=1 to nve[ii] do
  begin
    aa:=e[ii,j,1];
    for k:=1 to nve[ll] do
    begin
      if e[ll,k,3] <> 1 then goto 14;
      cc:=e[ll,k,1]; dd:=e[ll,k,2];
      ax:=(sc[ii,aa,1]); ay:=(sc[ii,aa,2]);
      cx:=(sc[ll,cc,1]); cy:=(sc[ll,cc,2]);
      dx:=(sc[ll,dd,1]); dy:=(sc[ll,dd,2]);
      v1:=dx-cx; v2:=dy-cy;
      u1:=ax-cx; u2:=ay-cy;
      { Burada arkada kalan nesnenin görülen son noktaları belirlenir. }
      test:=round(u2*v1-u1*v2);
      { Kenar noktalarının öndeki nesnenin içinde mi dışında mı kaldığı
        test edilir }
      if test<=0 then begin endpoint[j]:= 1; goto 15; end;
14:   end;
```

```
    endpoint[j]:=-1;
15:end;
end;

procedure Ciz;
{ İki nesne arasında bir çakışma yoksa her iki nesne burada çizilir. }
var
kj,f1,f2  : byte;
begin
  for kj:=1 to nve[ll] do
  begin
    if e[ll,kj,3]<>0 then
    begin
        f1:=e[ll,kj,1];  f2:=e[ll,kj,2];
        aa:=(sc[ll,f1,1]); bb:=(sc[ll,f1,2]);
        cc:=(sc[ll,f2,1]); dd:=(sc[ll,f2,2]);
        aa:=round(scf*aa+tx); bb:=round(ty-bb);
        cc:=round(scf*cc+tx); dd:=round(ty-dd);
        line(aa,bb,cc,dd);
    end;
  end;
end;

function sgn(r:real):shortint;
begin
  if r>1 then sgn:= 1;
  if r=0 then sgn:= 0;
  if r<1 then sgn:=-1;
end;

procedure ilk_nesne_belirle;
```

```
{ eğer nesneler arasında bir çakışma varsa birinci öncelikli
  ve ikinci öncelikli nesnelerin belirlenmesi işlemi yapılır.
  Bu işlem bakış konumuna göre yakında ve uzakta olan nesnenin
  belirlenmesi için yapılır.}
label
  3,4,5,6,7,26,27;
begin
  i:=1; l:=2;
  for j:=1 to nve[i] do
  begin
    aa:=e[i,j,1];  bb:=e[i,j,2];
    for k:=1 to nve[l] do
    begin
      if e[l,k,3]=1 then
      begin
        cc:=e[l,k,1]; dd:=e[l,k,2];
        { ---------- Kesişim Testi ---------- }
        ax:=sc[i,aa,1]; ay:=sc[i,aa,2];
        bx:=sc[i,bb,1]; by:=sc[i,bb,2];
        cx:=sc[l,cc,1]; cy:=sc[l,cc,2];
        dx:=sc[l,dd,1]; dy:=sc[l,dd,2];
        r1:=ax-bx; r2:=ay-by; r3:=cx-dx; r4:=cy-dy;
        s1:=sgn(r1); s2:=sgn(r2); s3:=sgn(r3); s4:=sgn(r4);
        a1:=ax; b1:=bx; c1:=cx; d1:=dx; a2:=ay; b2:=by; c2:=cy; d2:=dy;
        if s1>0 then begin a1:=bx; b1:=ax; end;
        if s2>0 then begin a2:=by; b2:=ay; end;
        if s3>0 then begin c1:=dx; d1:=cx; end;
        if s4>0 then begin c2:=dy; d2:=cy; end;
        { üst üste gelen kenarlar için minimax testi }
        if (b1<c1) or (d1<a1) or (b2<c2) or (d2<a2) then begin
                                                    ncut:=0;
```

```
                                                    goto 3;
                                                end;
    m1:=(r2/r1); m2:=(r4/r3); m3:=(m2-m1);
    if (m3=0) or (m3<=0.001) then begin
                                        ncut:=0;
                                        goto 3;
                                    end;
    { Kesişim noktası hesabı }
    z0:=ay-cy; z1:=m1*ax-m2*cx; z2:=(ay-m1*ax)*m2; z3:=(cy-2*cx)*m1;
    xi:=(z0-z1)/m3; yi:=(z2-z3)/m3;
    if (xi<a1) or (xi>b1) or (xi<c1) or (xi>d1) or (yi<a2) or
      (yi>b2) or (yi<c2) or (yi>d2) then ncut:=0 else ncut:=1;
    { ---------------------------------------- }
    if ncut=0 then goto 3;
    { ------------ Derinlik Testi ------------- }
    ii:=i; ij:=j;
    a1:=ax; a2:=ay; a3:=sc[i,aa,3]; q:=e[i,j,4];
    b1:=bx; b2:=by; b3:=sc[i,bb,3];

    for jj:=1 to nps[ii,q] do;
    begin
        if sf[ii,q,jj] <> e[ii,ij,2] then goto 6;
        kk:=sf[ii,q,jj+1]; goto 7;
6:  end;
7:  c1:=sc[ii,kk,1]; c2:=sc[ii,kk,2]; c3:=sc[ii,kk,3];
    aaa:=b1-a1; bbb:=b2-a2; ccc:=b3-a3;
    ddd:=c1-a1; eee:=c2-a2; fff:=c3-a3;
    xx:=bbb*fff-ccc*eee; yy:=ccc*ddd-aaa*fff; zz:=aaa*eee-bbb*ddd;
    t1:=xx*a1+yy*a2+zz*a3;
    t2:=xx*xi+yy*yi;
    if (t1=0) and (t2=0) then dt1:=0.000001 else dt1:=(t1-t2)/zz;
```

```
        { ikinci nesnenin derinlik testi}
        ii:=l; ij:=k;
        a1:=cx; a2:=cy; a3:=sc[l,cc,3]; q:=e[l,k,4];
        b1:=dx; b2:=dy; b3:=sc[l,dd,3];

        for jj:=1 to nps[ii,q] do;
        begin
            if sf[ii,q,jj] <> e[ii,ij,2] then goto 26;
            kk:=sf[ii,q,jj+1]; goto 27;
26:     end;
27:     c1:=sc[ii,kk,1]; c2:=sc[ii,kk,2]; c3:=sc[ii,kk,3];
        aaa:=b1-a1; bbb:=b2-a2; ccc:=b3-a3;
        ddd:=c1-a1; eee:=c2-a2; fff:=c3-a3;
        xx:=bbb*fff-ccc*eee; yy:=ccc*ddd-aaa*fff; zz:=aaa*eee-bbb*ddd;
        t1:=xx*a1+yy*a2+zz*a3;
        t2:=xx*xi+yy*yi;
        if (t1=0) and (t2=0) then goto 5;
        dt2:=(t1-t2)/zz;
        { ---------------------------------------- }
        if dt1>dt2 then goto 4 else goto 5;
4:      kkk:=0; ii:=1; ll:=2; Ciz; exit;
5:      kkk:=0; ii:=2; ll:=1; Ciz; exit;
        end;
3:   end;
  end;
  { ------------ Kapsama Testi ------------ }
  for i:=1 to 2 do
  begin
    zmx[i]:=0; zmn[i]:=1;
    for j:=1 to nv[i] do
    begin
```

```
      if sc[i,j,3] > zmx[i] then zmx[i]:=sc[i,j,3];
      if sc[i,j,3] < zmn[i] then zmn[i]:=sc[i,j,3];
    end;
  end;
  { ------------------------------------ }
  { Nesnelerin öncelik sırasına göre çizilmesi }
  if ( zmn[1] < zmx[2] ) and ( ymx[2] > ymx[1] ) then begin
                                                    ll:=2;
                                                    Ciz;
                                                    start:=true;
                                                    exit;
                                                  end;
  if ( zmx[2] < zmn[1] ) and ( ymx[1] > ymx[2] ) then begin
                                                    ll:=1;
                                                    Ciz;
                                                    start:=true;
                                                    exit;
                                                  end;
if (ymx[2] > ymx[1]) then begin
                            kkk:=1; ii:=1; ll:=2;
                            Ciz;
                            exit;
                          end;
  if (ymx[1] > ymx[2]) then begin
                            kkk:=1; ii:=2; ll:=1;
                            Ciz;
                            exit;
                          end;
end;

procedure minimax;
```

```
begin
  for i:=1 to 2 do
  begin
    xmx[i]:=-999999999; xmn[i]:=999999999;
    ymx[i]:=xmx[i];     ymn[i]:=xmn[i];
    for j:=1 to nve[i] do
    begin
      if e[i,j,3]=1
       then
       begin
         k:=e[i,j,1];
         aa:=round(sc[i,k,1]); bb:=round(sc[i,k,2]);
         if aa>xmx[i] then xmx[i]:=aa;
         if aa>xmn[i] then xmn[i]:=aa;
         if bb>ymx[i] then ymx[i]:=bb;
         if bb>ymn[i] then ymn[i]:=bb;
       end;{if}
    end;{next j}
  end;{next i}
  if (xmx[1]<xmn[2]) or (xmx[2]<xmn[1]) or
    (ymx[1]<ymn[2]) or (ymx[2]<ymn[1])
     then overlap:=0 else overlap:=1;
end;

procedure Gorunen_Kenar;
label 1,2;
var
  e1,e2 : byte;
  m  : shortint;
begin
  for i:=1 to 2 do { i : nesne sayısı için döngü işlemi }
```

```
begin
  m:=1;
  for j:=1 to ns[i] do { ns : nesnenin yüzey sayısı için döngü işlemi }
  begin
    { Eğer yüzey görünmüyorsa işlemleri yapma }
     if vsf[i,j]=0 then goto 1;
    e1:=sf[i,j,1]; { yüzeyin tanımlanan ilk noktasının numarası }
    for k:=2 to nps[i,j] do  { yüzeyi oluşturan noktalar için döngü }
    begin
      e2:=sf[i,j,k]; {yüzeyin tanımlanan ikinci
                      ve sonraki noktalarının numaraları}
      for l:=1 to m do
      begin
        if (e[i,l,1]=e2) and (e[i,l,2]=e1)
        then
          begin
            e[i,l,3]:=2; goto 2;
          end;
      end;
      {Görünen yüzeylerin kenar bilgileri
       1,2 : kenarın uç noktaları,
        3  : görünen kenarın kaç yüzeye ait olduğu,
        4  : kenarın ait olduğu yüzey numarası }
      e[i,m,1]:=e1; e[i,m,2]:=e2; e[i,m,3]:=1; e[i,m,4]:=j;
      m:=m+1; { nesnenin görünen kenar sayısı }
2:    e1:=e2; { son tanımlanan yüzey noktasının değerinin bir önce
                tanımlanan yüzey noktasının yerine aktarılması}
    end;
    nve[i]:=m-1; {bir nesne için toplam görünen kenar sayısı}
1:  end;
end;
```

```
end;

procedure Gorunurluk;
var
  xp,yp,zp,lx,ly,lz,t : real;
begin
  { görüş noktasının belirlenmesi }
  xp:= d * sn2 * cn1;  yp:= d * sn2 * sn1;  zp:= d * cn2;
  for i:=1 to 2 do
  { her i nesnesinin j yüzeyi için l görüş vektörünün hesabı }
  begin
    for j:=1 to ns[i] do
    begin
      lx:= xp - v[i,sf[i,j,1],1];
      ly:= yp - v[i,sf[i,j,1],2];
      lz:= zp - v[i,sf[i,j,1],3];
      { 0° <n.l <90° ise t>0°,  90° <n.l<180° ise t<=0 olur}
      t:= n[i,j,1] * lx + n[i,j,2] * ly + n[i,j,3] * lz;
      {Eğer t>0 ise yüzey görünen yüzeydir ve vsf[i,j]=1 olur.}
      if t<=0 then vsf[i,j]:=0 else vsf[i,j]:=1;
    end;
  end;
end;

procedure normal;
var
  u,w   : array[1..3] of real;
begin
  { yüzey normal vektörlerinin hesabı }
  for i:=1 to 2 do
  begin
```

```
    for j:=1 to ns[i] do
    begin
      for k:=1 to 3 do
      begin
        u[k]:=v[i,sf[i,j,2],k] - v[i,sf[i,j,1],k];
        w[k]:=v[i,sf[i,j,3],k] - v[i,sf[i,j,1],k];
      end;
      n[i,j,1]:= u[2] * w[3] - w[2] * u[3];
      n[i,j,2]:= u[3] * w[1] - w[3] * u[1];
      n[i,j,3]:= u[1] * w[2] - w[1] * u[2];
    end;
  end;
end;

procedure Yuzey;
{ Nesnelerin yüzey sayısı, her bir yüzeyini oluşturan nokta sayısı
  ve numralandırılmış noktaların tanımlandığı blok }
begin
  ns[1]:= 5;  ns[2]:= 6;  { ns : nesnelerin yüzey sayısı }

  { nps : her bir yüzeyi oluşturan nokta sayısı }
  nps[1,1]:= 4;  nps[1,2]:= 4;  nps[1,3]:= 4;  nps[1,4]:= 4;  nps[1,5]:= 5;
  nps[2,1]:= 5;  nps[2,2]:= 5;  nps[2,3]:= 5;
  nps[2,4]:= 5;  nps[2,5]:= 5;  nps[2,6]:= 5;

  { sf : Numaralandırılmış noktalar ile yüzey tanımlama }
  sf[1,1,1]:= 1; sf[1,1,2]:= 2; sf[1,1,3]:= 3; sf[1,1,4]:= 1;
  sf[1,2,1]:= 2; sf[1,2,2]:= 5; sf[1,2,3]:= 3; sf[1,2,4]:= 2;
  sf[1,3,1]:= 5; sf[1,3,2]:= 4; sf[1,3,3]:= 3; sf[1,3,4]:= 5;
  sf[1,4,1]:= 4; sf[1,4,2]:= 1; sf[1,4,3]:= 3; sf[1,4,4]:= 4;
  sf[1,5,1]:= 1; sf[1,5,2]:= 4; sf[1,5,3]:= 5; sf[1,5,4]:= 2; sf[1,5,5]:= 1;
```

```
  sf[2,1,1]:= 1; sf[2,1,2]:= 2; sf[2,1,3]:= 3; sf[2,1,4]:= 4; sf[2,1,5]:= 1;
  sf[2,2,1]:= 6; sf[2,2,2]:= 7; sf[2,2,3]:= 2; sf[2,2,4]:= 1; sf[2,2,5]:= 6;
  sf[2,3,1]:= 5; sf[2,3,2]:= 8; sf[2,3,3]:= 7; sf[2,3,4]:= 6; sf[2,3,5]:= 5;
  sf[2,4,1]:= 4; sf[2,4,2]:= 3; sf[2,4,3]:= 8; sf[2,4,4]:= 5; sf[2,4,5]:= 4;
  sf[2,5,1]:= 4; sf[2,5,2]:= 5; sf[2,5,3]:= 6; sf[2,5,4]:= 1; sf[2,5,5]:= 4;
  sf[2,6,1]:= 7; sf[2,6,2]:= 8; sf[2,6,3]:= 3; sf[2,6,4]:= 2; sf[2,6,5]:= 7;

end;

procedure Kose;
begin
  nv[1]:=5;nv[2]:=8;
  k1:=7;k2:=10;w:=k2/(k2-k1);
  { Nesneleri oluşturan noktaların numaralandırılması }
   v[1,1,1]:= 1.3; v[1,1,2]:= 2.3; v[1,1,3]:= 2.6;
   v[1,2,1]:=-1.3; v[1,2,2]:= 2.3; v[1,2,3]:= 2.6;
   v[1,3,1]:= 0.1; v[1,3,2]:= 0.1; v[1,3,3]:= 3.5;
   v[1,4,1]:= 1.3; v[1,4,2]:=-2.3; v[1,4,3]:= 2.6;
   v[1,5,1]:=-1.3; v[1,5,2]:=-2.3; v[1,5,3]:= 2.6;

   v[2,1,1]:= 1;   v[2,1,2]:= 2;   v[2,1,3]:= 0.5;
   v[2,2,1]:= 1;   v[2,2,2]:= 2;   v[2,2,3]:= 2.5;
   v[2,3,1]:= 1;   v[2,3,2]:=-2;   v[2,3,3]:= 2.5;
   v[2,4,1]:= 1;   v[2,4,2]:=-2;   v[2,4,3]:= 0.5;
   v[2,5,1]:=-1;   v[2,5,2]:=-2;   v[2,5,3]:= 0.5;
   v[2,6,1]:=-1;   v[2,6,2]:= 2;   v[2,6,3]:= 0.5;
   v[2,7,1]:=-1;   v[2,7,2]:= 2;   v[2,7,3]:= 2.5;
   v[2,8,1]:=-1;   v[2,8,2]:=-2;   v[2,8,3]:= 2.5;

  for i:=1 to 2 do
```

```
    begin
     for j:=1 to nv[i] do
       begin
         x:=v[i,j,1];
         y:=v[i,j,2];
         z:=v[i,j,3];
         { göz koordinat değerlerinin hesabı }
         xe:= -x * sn1 + y * cn1;
         ye:= -x * cn1 * cn2 - y * sn1 * cn2 + z * sn2;
         ze:= -x * sn2 * cn1 - y * sn2 * sn1 - z * cn2 + d;
         ec[i,j,1]:=xe;
         ec[i,j,2]:=ye;
         ec[i,j,3]:=ze;
         { ekran koordinat değerlerinin hesabı }
         xs:= (vd / s) * (xe / ze);
         ys:= (vd / s) * (ye / ze);
         zs:= w * (1 - k1 / ze);      { zs : nesneyi oluşturan köşelerin }
         sc[i,j,1]:=xs;               {      derinlik deşeri }
         sc[i,j,2]:=ys;
         sc[i,j,3]:=zs;
       end;
     end;
end;

procedure gorunen_kenar_parca_ciz;
label 17,18;
begin
{ Nesneleri çakışması, üst üste gelmesi ile kesişim, derinlik ve kapsama
  testleri sonrası görünen kenar parçalarının çizimi }
  if (tcut= 0) and (vi= 1) then begin line(ax,ay,bx,by); exit; end;
  if (tcut= 0) and (vi=-1) then exit;
```

```
  if (tcut= 1) and (vi= 1) then begin line(ax,ay,ix[1],iy[1]); exit; end;
  if (tcut= 1) and (vi=-1) then begin line(ix[1],iy[1],bx,by); exit; end;
  if (ax<bx) and (ix[1]<ix[2]) then goto 17;
  if (ax>bx) and (ix[2]<ix[1]) then goto 17 else goto 18;
17:line(ax,ay,ix[1],iy[1]); line(ix[2],iy[2],bx,by); exit;
18:line(ax,ay,ix[2],iy[2]); line(ix[1],iy[1],bx,by); exit;
end;

procedure ParcaCiz;
{Üst üste gelen kenarlar için minimax testi ve kesişen noktalarının
 hesabı yapılır. }
label
  8,9,10,11,29;
begin
  i:=ii; l:=ll;
  for j:=1 to nve[ii] do
  begin
    vi:=endpoint[j]; ncut:=0; tcut:=0;
    aa:=e[ii,j,1]; bb:=e[ii,j,2];
    for k:=1 to nve[ll] do
    begin
      if kkk=1 then goto 8;
      if e[ll,k,3] <> 1 then goto 11 else goto 9;
8:    if e[ll,k,3] <> 2 then goto 11 else goto 9;
9:    cc:=e[ll,k,1]; dd:=e[ll,k,2];

      ax:=sc[i,aa,1];       ay:=sc[i,aa,2];
      bx:=sc[i,bb,1];       by:=sc[i,bb,2];
      cx:=sc[l,cc,1];       cy:=sc[l,cc,2];
      dx:=sc[l,dd,1];       dy:=sc[l,dd,2];
       r1:=ax-bx+0.2; r2:=ay-by+0.2; r3:=cx-dx+0.2; r4:=cy-dy+0.2;
```

```
        s1:=sgn(r1); s2:=sgn(r2); s3:=sgn(r3); s4:=sgn(r4);
        a1:=ax; b1:=bx; c1:=cx; d1:=dx; a2:=ay; b2:=by; c2:=cy; d2:=dy;
        if s1>0 then begin a1:=bx; b1:=ax; end;
        if s2>0 then begin a2:=by; b2:=ay; end;
        if s3>0 then begin c1:=dx; d1:=cx; end;
        if s4>0 then begin c2:=dy; d2:=cy; end;
        { üst üste gelen kenarlar için minimax testi }
        if (b1<c1) or (d1<a1) or (b2<c2) or (d2<a2) then begin ncut:=0; goto 29;
end;
        m1:=(r2/r1); m2:=(r4/r3); m3:=(m2-m1);
        if (m3=0) or (m3<=0.001) then begin ncut:=0; goto 29; end;
        { kesişen noktaların hesabı }
        z0:=ay-cy; z1:=m1*ax-m2*cx; z2:=(ay-m1*ax)*m2; z3:=(cy-m2*cx)*m1;
        xi:=(z0-z1)/m3; yi:=(z2-z3)/m3;
        if (xi<a1) or (xi>b1) or (xi<c1) or (xi>d1) or
        (yi<a2) or (yi>b2) or (yi<c2) or (yi>d2) then ncut:=0 else ncut:=1;
29:     if ncut=0 then goto 11 else goto 10;
10:     tcut:=tcut+ncut;
        ix[tcut]:=round(scf*xi+tx); iy[tcut]:=round(ty-yi);
11:   end;
    ax:=round(scf*ax+tx); ay:=round(ty-ay);
    bx:=round(scf*bx+tx); by:=round(ty-by);
    gorunen_kenar_parca_ciz;
  end;
end;

procedure parametreler;
var
  pi :real;
begin
  d:=10;tx:=240;ty:=300;
```

```
  pi:=3.141593;scf:=1.2;s:=1;
  theta:=theta*pi/180;phi:=phi*pi/180;
  sn1:=sin(theta);sn2:=sin(phi);cn1:=cos(theta);cn2:=cos(phi);
end;

procedure grafik;
begin
  { grafik adaptör ve modunun belirlenmesi }
  DetectGraph(Gd,Gm);
  if GraphResult <> grOk then begin
                              Writeln('Graphics error...');Halt(1);
                            end;
  InitGraph(Gd, Gm, 'c:\bp\bgi');
  Initevents; { Mouse tanımlama }
end;

procedure DrawButton(XC : Integer; AStr : string; Pressed : Boolean);
{ Butonların çizildiği altprogram}
{ Butonun seçme ve bırakma durumunda tekrar çizilmesi }
begin
  CommandYMin:= CommandYMin + 30;
  SetLineStyle(SolidLn,0,NormWidth);
  If Pressed then
    Setcolor(black)
  else
    Setcolor(white);
  Line(round(CommandXMin)+5,round(CommandYMin)+7+XC*45,
       round(CommandXMax),round(CommandYMin)+7+XC*45);
  Line(round(CommandXMin)+5,round(CommandYMin)+7+XC*45,
       round(CommandXMin)+5,round(CommandYMin)+30+XC*45);
  If Pressed then
```

```
    Setcolor(white)
  else
    Setcolor(black);
  Line(round(CommandXMin)+5,round(CommandYMin)+30+XC*45,
       round(CommandXMax),round(CommandYMin)+30+XC*45);
  Line(round(CommandXMax),round(CommandYMin)+7+XC*45,
       round(CommandXMax),round(CommandYMin)+30+XC*45);
 Outtextxy(round(CommandXMin)+8,round(CommandYMin)+17+XC*45,Astr);
  CommandYMin:= CommandYMin - 30;
  Settextstyle(Triplexfont,Horizdir,3);
  Setcolor(black);
  Outtextxy(round(CommandXMin)+20,round(CommandYMin)+5,'3 D');
  Outtextxy(round(CommandXMin)+2,round(CommandYMin)+28,'Graphic');
  Setcolor(red);
  Outtextxy(round(CommandXMin)+18,round(CommandYMin)+3,'3 D');
  Outtextxy(round(CommandXMin)+0,round(CommandYMin)+26,'Graphic');
  Settextstyle(Defaultfont,Horizdir,1);
end;

procedure GorusNoktasi;
var
 C : Char;
 xm,ym :integer;
begin
{Mouse hareketlerinin ekran konumu konrol edilerek gerekli işlemler yapılır}
{theta, phi ve vd parametrelerinin değerleri değiştirilir}
    Xm:= 8 * MOUSEWHERE.X; Ym:= 8 * MOUSEWHERE.Y;
    SetViewPort(20, 20, getmaxx-120, getmaxy-20, clipoff);
    SetViewPort(getmaxx-107, 20, getmaxx-20, getmaxy-20, clipon);
    if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
                                   and (ym>95) and (ym<120)
```

```
then begin
        C:='1';
        Drawbutton(1,' right',True);
        mxx:=mxx+2;
        if mxx>358 then mxx:=2;
      end;
if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
                                and (ym>145) and (ym<170)
then begin
        C:='2';
        Drawbutton(2,' left',True);
        mxx:=mxx-2;
        if mxx<2 then mxx:=360;
      end;
if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
                                and (ym>185) and (ym<210)
then begin
        C:='3';
        Drawbutton(3,' upwards',True);
        myy:=myy-2;
        if myy<2 then myy:=2;
      end;
if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
                                and (ym>235) and (ym<260)
then begin
        C:='4';
        Drawbutton(4,'downward',True);
        myy:=myy+2;
        if myy>88 then myy:=88;
      end;
if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
```

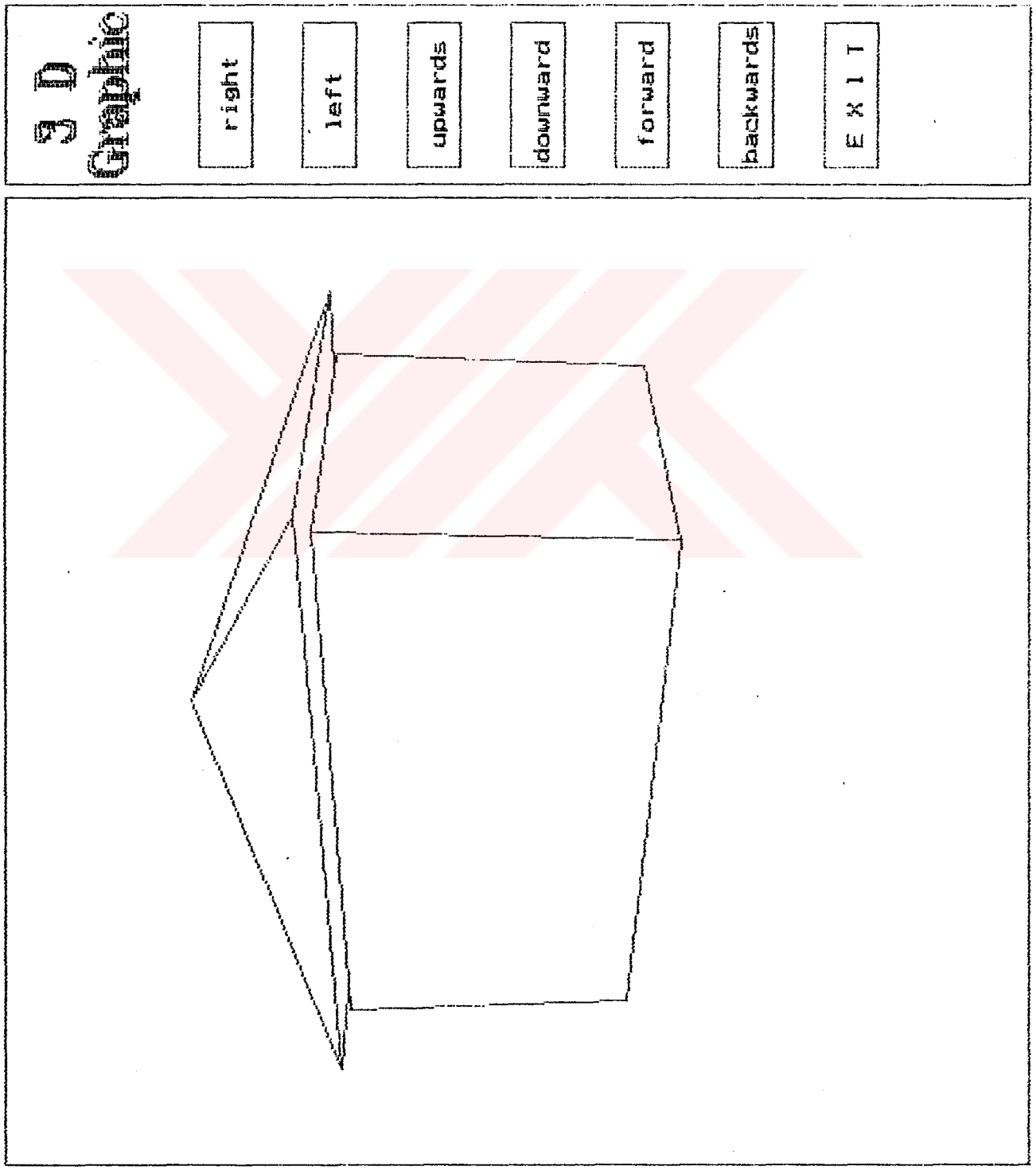
```
                                    and (ym>275) and (ym<300)
then begin
        C:='5';
        Drawbutton(5,' forward',True);
        vd:=vd+5;
        if vd>1500 then vd:=1500;
      end;
if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
                                    and (ym>325) and (ym<350)
then begin
        C:='6';
        Drawbutton(6,'backwards',True);
        vd:=vd-5;
        if vd<15 then vd:=15;
      end;
if (mousebuttons=mbleftbutton) and (xm>530) and (xm<620)
                                    and (ym>375) and (ym<390)
then begin
        C:='7';
        CloseGraph;
        halt(1);
      end;
Case C of
   '1' : DrawButton(1,'  right ',False);
   '2' : DrawButton(2,'  left  ',False);
   '3' : DrawButton(3,' upwards ',False);
   '4' : DrawButton(4,'downward',False);
   '5' : DrawButton(5,' forward ',False);
   '6' : DrawButton(6,'backwards',False);
end;
SetViewPort(getmaxx-107, 20, getmaxx-20, getmaxy-20, clipoff);
```

```
    SetViewPort(20, 20, getmaxx-120, getmaxy-20, clipon);
    theta:=mxx; phi:=myy;
end;

procedure ekran;
begin
{Ekranda görüntü ve menü kutuları ile butonlar çizilir}
  setfillpattern(fon,8);
  Bar(19, 19, GetMaxX - 119, GetMaxY - 19);
  Bar(529, 19, GetMaxX - 19, GetMaxY - 19);
  Rectangle(18, 18, GetMaxX - 118, GetMaxY - 18);
  Rectangle(528, 18, GetMaxX - 18, GetMaxY - 18);
  SetViewPort(20, 20, getmaxx-120, getmaxy-20, clipoff);
  SetViewPort(getmaxx-107, 20, getmaxx-20, getmaxy-20, clipon);
  DrawButton(1,' right ',False);
  DrawButton(2,' left ',False);
  DrawButton(3,' upwards ',False);
  DrawButton(4,'downward',False);
  DrawButton(5,' forward ',False);
  DrawButton(6,'backwards',False);
  DrawButton(7,' E X I T',False);
  SetViewPort(getmaxx-107, 20, getmaxx-20, getmaxy-20, clipoff);
  SetViewPort(20, 20, getmaxx-120, getmaxy-20, clipon);
end;

begin      { Program Ana Bloğu }
  grafik;
  mxx:=57;myy:=68;vd:=500;
  CommandXMin := 0; CommandXMax := 80;
  CommandYMin := 0; CommandYMax := 450;
  ekran;
```

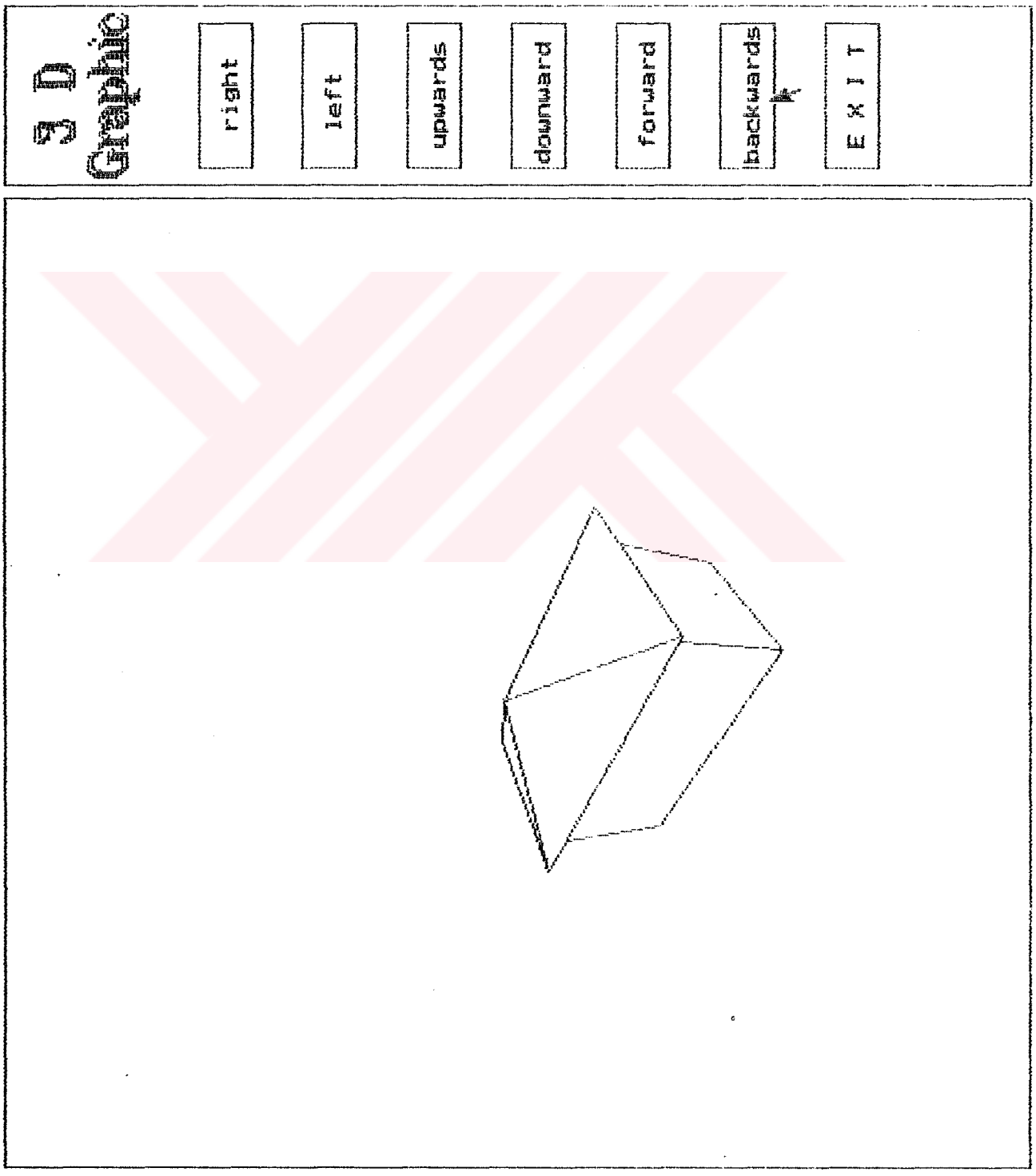
```
Repeat
    GorusNoktasi;
    parametreler;
    Kose;
    Yuzey;
    normal;
    Gorunurluk;
    Gorunen_Kenar;
    setcolor(15);
    if overlap=0
    then
    {Nesnelerin üst üste gelmesi durumu yoksa iki nesne çizilir }
    begin
     for ll:=1 to 2 do
     begin
       Ciz;
     end;
     repeat
     {Mouse sol tuşuna basılana kadar program burada bekler
      Nesneler çizilmiş olarak ekranda durur.}
     until mousebuttons=mbleftbutton;
     {Mouse ile nesnelerin yeni konumu belirlenmeden önce
     nesne taban renginde çizdirilerek ekrandan silinmesi sağlanır.}
     setfillpattern(fon,8);
     SetViewPort(20, 20, getmaxx-120, getmaxy-20, true);
     setcolor(8);
     for ll:=1 to 2 do
     begin
       Ciz;
     end;
     goto 20;
```

```
        end;
        {-------------------------------------------------------------}
        {Nesnelerin üst üste gelmesi durumunda program buradan itibaren
         çalışarak kesişim, derinlik ve kapsama testleri yapılır }
        minimax;
        start=false;
        ilk_nesne_belirle;  {Önceliği olan, bakış noktasına göre
                             uzakta olan nesne belirlenir. }
        if start=true then goto 21;
        v_endpoint;  { Görünen uç nokta }
        ParcaCiz;  { Çizgi parçasının çizimi }
21:     repeat
          {Mouse sol tuşuna basılana kadar program burada bekler
           Nesneler çizilmiş olarak ekranda durur.  }
        until mousebuttons=mbleftbutton;
        {Mouse ile nesnelerin yeni konumu belirlenmeden önce
          nesne taban renginde çizdirilerek ekrandan silinmesi sağlanır }
        setfillpattern(fon,8);
        Bar(19, 19, GetMaxX - 119, GetMaxY - 19);
        SetViewPort(20, 20, getmaxx-120, getmaxy-20, true);
        setcolor(8);
        minimax;
        start:=false;
        ilk_nesne_belirle;
        if start=true then goto 20;
        v_endpoint;
        ParcaCiz;  { Nesnenin taban renginde çizdirilmesi }
20:Until 2=3;
end.
```

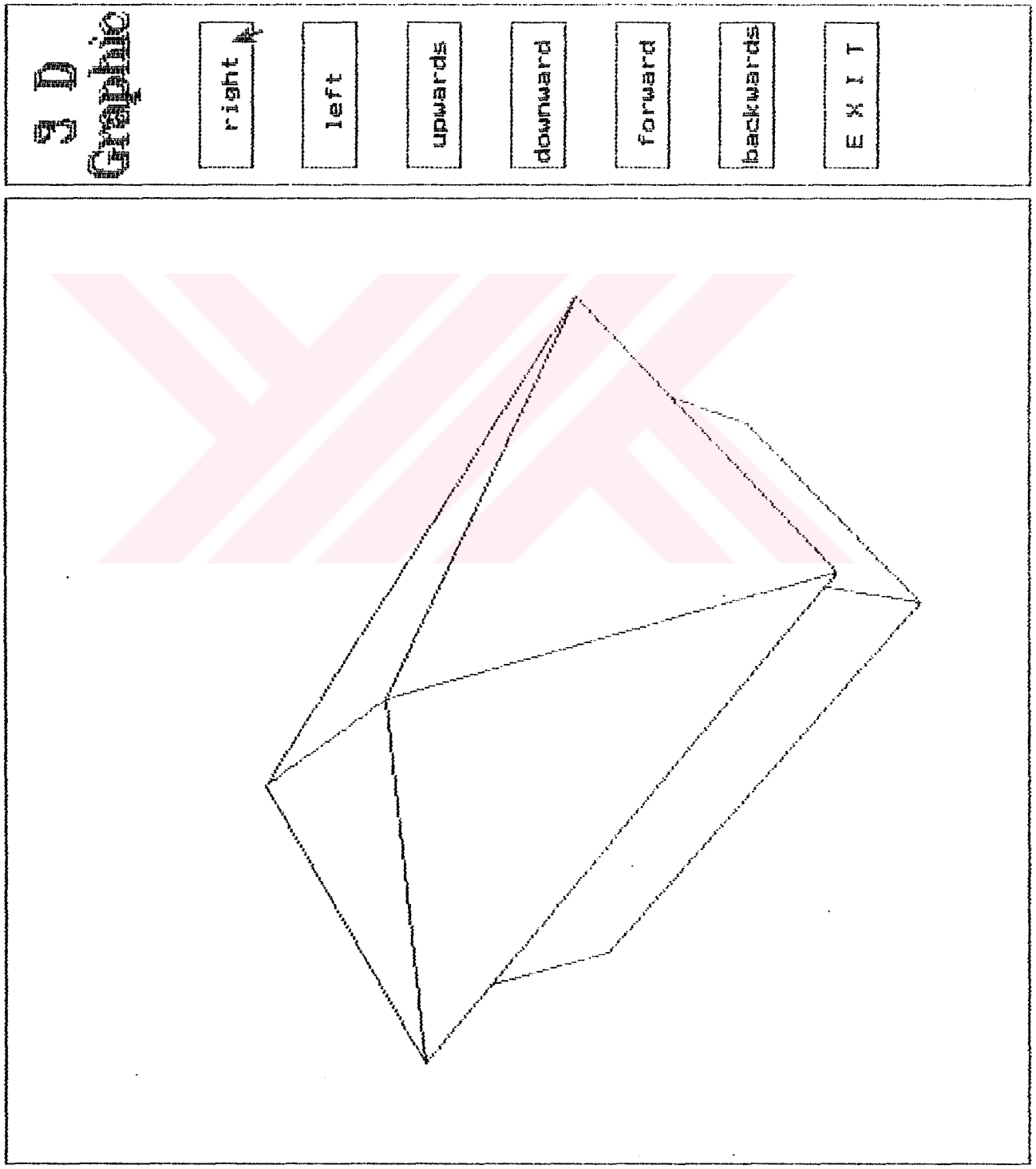
Ek-2. θ=40°, ϕ=80° ve *vd*=600 için nesnelerin görünüşü



Ek-3. θ=45°, ϕ=50° ve *vd*=250 için nesnelerin görünüşü



Ek-4. θ=45°, ϕ=35° ve *vd*=500 için nesnelerin görünüşü



Ek-5. θ=30°, ϕ=10° ve *vd*=400 için nesnelerin görünüşü