ANİMASYON ORTAMLARI İÇİN MODEL GELİŞTİRME SİSTEMİ

YÜKSEK LİSANS TEZİ

(ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ)

Recep KILINÇ

AĞUSTOS 1996

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Danışman

Yrd.Doç.Dr. Abdullah ÇAVUŞOĞLU

Sınav Jürisi

Başkan : Prof.Dr. Halil Bilüklüoğlu

Üye : Doç.Dr. Abdullah Çavuşoğlu

Üye : Y.Doç.Dr. Osman Gürdal

Bu tez Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Esaslarına uygundur.

ANİMASYON ORTAMLARI İÇİN MODEL GELİŞTİRME SİSTEMİ

(YÜKSEK LİSANS TEZİ)

Recep KILINÇ

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 1996

ÖZ

Bu çalışmada, geliştirilen yazılım ile kullanıcı birbirine bağlı ızgara şeklinde bir yüzey yardımıyla kendi modelini yapar. Bu yöntemde kullanıcı, mouse kullanarak görüntünün gereken ölçülerinde tanımlanan yüzeyin şeklini değiştirir. Bu animasyon tekniği diğer metodlarla oluşturulması imkansız olan karışık veya biçimsiz modellerin oluşturulmasında oldukça kullanışlıdır. Bu tekniğin getirdiği diğer bir avantaj da nesne oluşturulurken her bir noktanın koordinat değerlerinin ayrı ayrı tanımlanmasına gerek olmamasıdır.

Anahtar kelimeler: Mouse, Animasyon tekniği, Koordinat değerleri, Nesne

TO DEVELOP A MODEL SYSTEM FOR ANIMATIONS ENVIRONMENTS

(M. Sc. Thesis)

Recep KILINÇ

GAZI UNIVERSITY

INSTITUE OF SCIENCE AND TECHNOLOGY

AUGUST 1996

ABSTRACT

In this study, the user starts building his model with an interconnected grid structure. By using mouse, the user changes the shape of this surface until the required scene element is formed. This animation technique is useful when the required model is either too complex or has an awkward shape that is imposible to generate using other methods. Another advantage of this technique is that the user does not need to describe the coordinate values of each individual vertex that makes up the object.

Key words : Mouse, Animation technique, Coordinate values, Object

TEŞEKKÜR

Çalışmalarım süresince her türlü yardımlarını esirgemeyen danışmanım olan Gazi Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Egitimi Anabilim Dalı BaşkanıYrd. Doç. Dr. Abdullah ÇAVUŞOĞLU'na, Araştırma Görevlisi Hayali ÇAYIR'a, anlayışlarından dolayı Aileme ve İş Arkadaşlarıma teşekkür ederim.

1996,Recep KILINÇ

İÇİNDEKİLER

Sayfa

Şekil **Sayfa**

1. GİRİŞ

Bilgisayar yazılım ve donanımında meydana gelen başdöndürücü gelişmeler oldukça hızlı donanım gerektiren grafik konusunda ileri adımlar atılabilmesine imkan tanımıştır. Bu sayede, bilgisayarlarda animasyon olayı gerçekçi üç boyutlu resimler oluşturabilmenin yanında bu resimleri hızlı donanımlar üzerinden bilgisayar ekranına aktarılması ile tamamen bilgisayar ürünü olan zahiri filmler oluşturulabilecek duruma ulaşmıştır.

Bilgisayarda animasyon yapılması, bu iş için geliştirilen arabirimler vasıtasıyla mümkün olabildiğinden bunların kullanışlı olması önemli bir faktör olarak ortaya çıkmaktadır.

Halihazırda geliştirilen animasyon sistemlerinde kullanıcıya önceden tanımlanmış nesneleri kullanabilme imkanı sağlanmaktadır. Ayrıca bu nesnelerin yine önceden belirlenen haraket kabiliyetleri çerçevesinde haraket etmeleri sağlanmaktadır.

Bu çalışmada geliştirilen sistem ile kullanıcının bilmesi gereken grafik bilgisi en az düzeyde farzedilerek tamamen karşılıklı etkileşim esasına dayalı olarak bir çalışma yapılmış ve üç boyutlu animasyonda kullanılabilecek sıradan veya sıradışı nesnelerin üretilebilmesi sağlanmıştır.

Sistem sayesinde kullanıcı, daha önceden tanımlanmış bir cisimler kütüphanesi kullanmak yerine kendine has bir metodla istediği cismi istediği şekilde oluşturabilmekte ve bu oluşumu adım adım bilgisayar ekranında takip ederek isteğine göre düzeltme yapabilmektedir. Geliştirilen cisimlerin önden, üstten, yandan, görüntüleri aynı anda ekrana aksettirilmiş ve çeşitli butonlar vasıtası ile de yapılacak işler mümkün mertebe otomatikleştirilmeye çalışılmıştır.

2. ÇALIŞMANIN TANITIMI

Bilgisayar teknelojisindeki gelişmeler duyu organlarını etkileyerek gerçekte olmayan veya algılanamayacak kadar zor olan olayları insana sandırarak en doğru tepkileri almayı amaçlıyor. Aynı zamanda normalde olması zor ve pahalı olayları bilgisayar ekranından kişilerin kullanımına sunması da zaman, işgücü ve maliyet gibi büyük tasarruflar sağlamaktadır.

Bilgisayar grafiklerinde ekrandaki geometrik modellerin görüntülerinin insan beyninde üç boyutlu gerçek görüntü hissini vermesi veya sanmasını amaçlayan bu konuya Sanal Gerçeklik (Virtual reality) denir. Üç boyutlu geometrik modellerin aslına uygun bilgisayar ekranına aktarılmasına da simülasyon adı verilir.

Burada üç boyutlu nesneleri bilgisayar ekranında görüntüleme çalışması yapılmıştır. Animasyonda kullanılan teknik Serbest-Biçim modelleme yöntemidir. Bu yöntemde kulanıcı kendi modelini birbirine bağlı ızgara şeklinde yapılmış düz bir yüzeyi kullanarak yapmaktadır. Kullanıcı mouse gibi aktif bir eleman kullanarak bu yüzeyin şeklini gerekli ölçüler kadar değiştirebilir.

Bu modelleme tekniği diğer modelleme tekniklerine göre imkansız olan karmaşık ve biçimsiz nesnelerin modellenmesinde oldukça kullanışlıdır. Bu

tekniğin diğer bir avantajıda nesneyi oluşturan her bir noktanın koordinat değerlerinin kullanıcı tarafından ayrı ayrı tanımlanmasına gerek yoktur.

Bilgisayar ekranında bir dağ oluşturduğumuzu düşünelim. Bu dağda tepeler ve değişen yükseklikler olacaktır. Bu dağın sınır değerlerini tanımlamak için poligon formunda her bir yükseltinin tam koordinat değerlerini bilmemiz ve bilgisayara girmemiz gerekir. Bu işlem sonunda başlangıçta istediğimiz şekilde dağın oluşup oluşmayacağını bilemeyiz. Fakat bir çok küçük poligondan oluşmuş bir yüzey kullanarak bunu oluştursak istediğimiz gibi ölçülerde değişiklik yaparak aslına uygun bir dağ şekli oluşturabiliriz.

2.1 Temel İşlemler ve Tanımlamalar

Bilgisayarla bir nesne görüntüsünün çizilebilmesi ve bu görüntü üzerinde çeşitli matematiksel dönüşümlerin uygulana bilmesi için nesnenin tanımlanması gerekir. Bir nesnenin tanımlanması, seçilen uygun bir koordinat sistemine göre noktalar ve bunları birleştiren doğrularla yapılır. Nesne üzerinde her hangi bir dönüşümün gerçekleştirilmesi doğrudan bu noktalar kullanılarak yapılır. Örneğin nesnenin büyüklüğünün değiştirilmesi, bir yeden başka bir yere taşınması, nesnenin döndürülmesi ve perspektifinin oluşturulması gibi dönüşümlerin yerine getirilmesi nesneyi tanımlayan

noktaların herbirine bu dönüşüm işlemlerinin ayrı ayrı uygulanması ile yapılır. Bu dönüşümler bilgisayarda grafik düzenlemelerin 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 iki üç boyutlu nesneler için uygulanabilir matematiksel formüllerdir. Bilgisayarın ekranı iki boyutlu bir yüzey olduğu için programcı çizimlerinde iki boyutlu olarak çalışmak zorundadır. Üç boyutlu nesnelerin ekranda görüntülenmesi, nesnenin boyutunun bir azaltılmasıyla gerçekleştirilir.Boyutun azaltılması ise nesne görüntüsünün ekran düzlemi üzerine izdüşürülmesiyle yapılır (Erdun, 1993).

Bilgisayar yardımıyla bir resmin tasarlanması için ekran üzerindeki noktalar kullanılır. Bazı dönüşüm işlemleri yardımıyla yüzeylerin oluşturulması için bir araya toplanmış noktalar kullanılır ve canlı resimler elde etmek içinde bu yüzeylerden faydalanılır.

2.1.1 Nokta tanımlaması

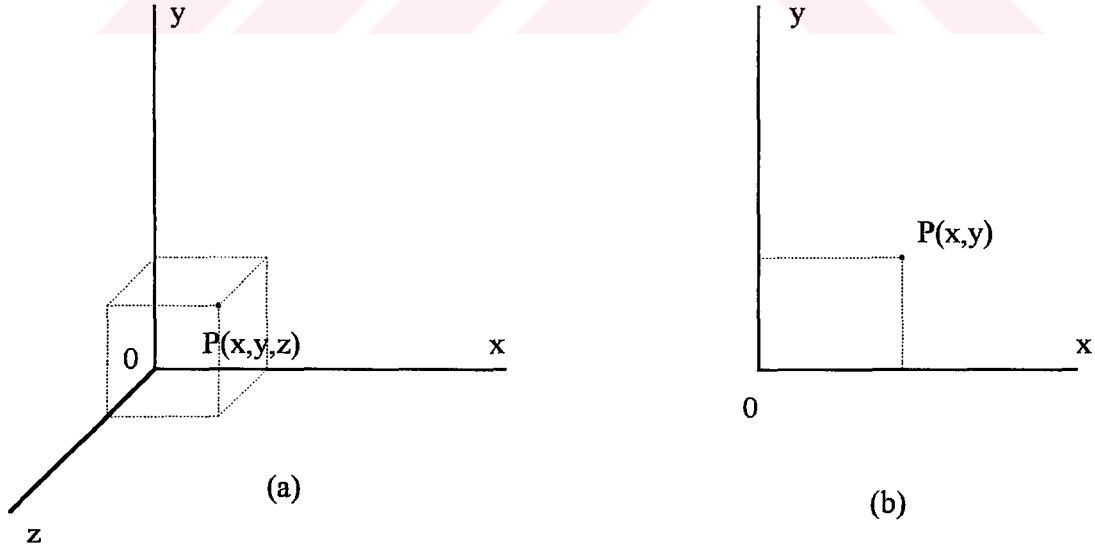
Uzayda bir yeri belirten sıfır boyutlu nesnelere nokta denir. Noktalar kullanılarak doğrular ve doğrular kullanılarak da çeşitli nesnelerin çizimi

yapılabilir. Noktalar iki ve üç boyutlu olarak koordinat sistemlerinde gösterilebilir (Şekil 2.1).

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

$$P = [x \quad y \quad z] \text{ veya } P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Burada gösterilen P matrisleri çoğunlukla vektör olarak isimlendirilirler.

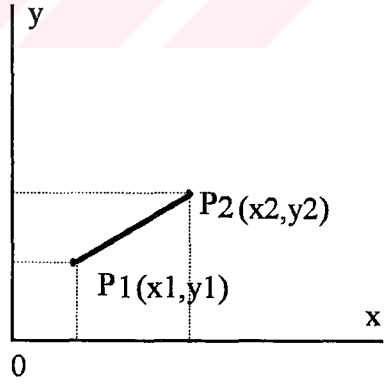


Şekil 2.1 Noktanın gösterilmesi. (a) üç boyutlu. (b) iki boyutlu

2.1.2 Doğru parçası tanımı

Bir doğru tanımlayabilmek için en az iki nokta kullanılmalıdır. Bir doğru parcası tanımlayabilmek için doğru parçasının başlangıç ve bitiş noktalarının belirlenmesi gereklidir (Şekil 2.2).

Bir doğru parçası bir vektör olarakta gösterilebilir. Koordinat sisteminde, nesneyi oluşturan vektörleri belirleyen noktalar serisi, bir matris olarak bilgisayarda ifade edilebilir. Bu matris bir nesneyi tanımlayan noktaların tümünü içerir. Bir nesneye dönüşüm işlemi uygulanması bu matris değerlerinin her birine ilgili dönüşüm işlemlerinin uygulanması ile sağlanır(Erdun, 1993).



Şekil 2.2 Bir doğru parcası

2.1.3 Yüzey tanımlaması

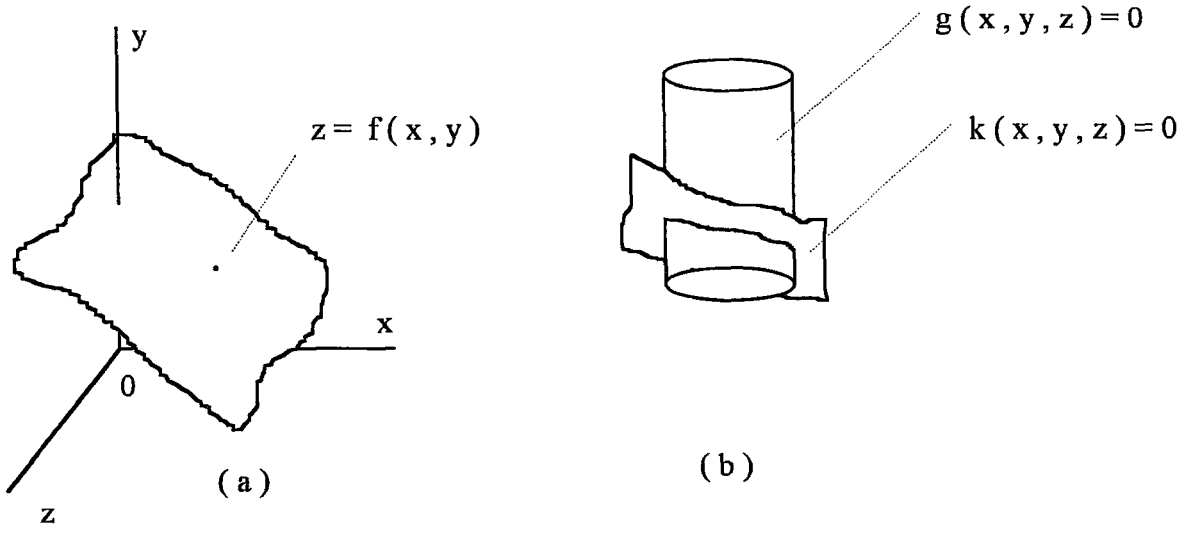
Bilgisayar ekranında nesnelerin geometrik modelleri oluşturulurken yüzey tanımlamaları yapılır. Bir nesnenin kaç yüzeyden oluştuğu ve bu yüzeylerin biçimi, yüzeyi oluşturan noktalar tarafından belirlenir. Bir küpün altı yüzeyi vardır ve her bir yüzey dört köşe ile tanımlanır. Nesneleri dışarıdan göründüğü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

2.1.3.1 Üç boyutlu yüzey

Üç boyutlu koordinat sisteminde bir yüzey iki şekilde gösterilebilir. Bunlar;

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

Birincisinde x ve y koordinat değerleri kullanılarak z değeri ile yüzey oluşturulur. Diğerinde ise bir değişkenin verilen sabit değerleri için diğer değişken değerlerinin değiştirilmesiyle elde edilen eğri gruplarından yüzeyler oluşturulur.



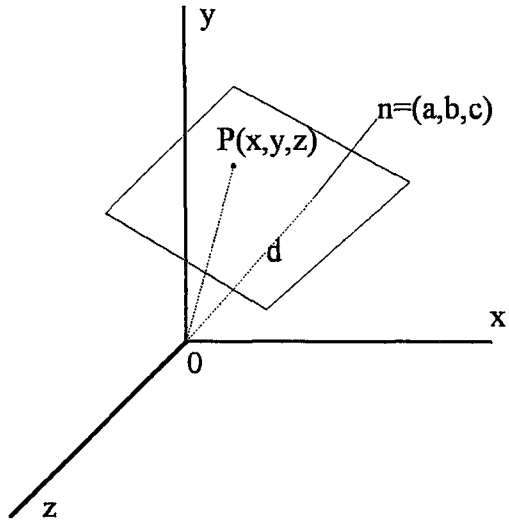
Şekil 2.3 Üç boyutlu koordinat sisteminde yüzey gösterimi.

Bilgisayar grafiklerinde bir düzlem denklemi aşağıdaki gibidir.

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

Burada, eğer d sıfırdan farklı ise denklemin d'ye bölünmesi ile düzlemde bir değişiklik olmaz. Bu nedenle denklemde sadece üç nokta bir düzlemi belirler. d orijinden düzleme olan uzaklık değerini gösterir ve aynı a,b,c katsayılarına sahip denklemlerden elde edilen bütün düzlemler birbirine paraleldir. d = 0 ise düzlem orijinden geçer.

$$ax + by + cz = 0$$



Şekil 2.4 Üç boyutlu koordinat sisteminde düzlem

P(x,y,z) noktası düzlem üzerinde olan keyfi bir noktadır ve **P** matrisi şeklinde gösterilebilir. Orijinden P(x,y,z) yönünde olan doğru veya vektör, düzlem içinden geçer. Aynı zamanda n(a,b,c) üç boyutlu bir nokta olarak ele alınabilir ve aşağıdaki gibi yazılabilir.

$$\mathrm{n} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad , \quad \mathrm{p} = \begin{bmatrix} x & y & z \end{bmatrix}$$

İki vektörün sıkaler çarpımını kullanarak, orijinden geçen düzlemin denklemi yazılabilir.

$$\mathrm{P*n} = \begin{bmatrix} x & y & z \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

n, düzlemdeki tüm vektörlere diktir. Bu nedenle üç boyutlu uzayda düzlemin doğrultusunu kontrol eder. Sadece düzlemin orijinden olan uzaklığına etki eder ve n(a,b,c) her zaman

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

düzlemine dik bir vektörü belirtir.

3. GEOMETRİK DÖNÜŞÜM İŞLEMLERİ

Burada temel iki boyutlu dönüşümlerin anlatımından yola çıkarak üç boyutlu dönüşüm ve nesnelerin modellenmesi anlatılacaktır.

3.1 Temel Dönüşüm İşlemleri

Bilgisayar grafiklerinde dönüşüm işlemi olarak adlandırılan üç temel işlem vardır. Bunlar:

1. Konum değiştirme (Translation)
2. Ölçeklendirme (Scaling)
3. Döndürme (Rotating)

Bu üç temel işlem, ya ayrı ayrı veya arka arkaya kullanılarak tüm dönüşümler gerçekleştirilir.

Bunun için matematiksel hesaplamalara ihtiyaç duyulur. Bilgisayar grafiklerinde nesne uzayı terimi bizim anladığımız gerçek dünyada nesnelerin koordinatlarını tanımlamak için kullanılır. Aynı nesnenin bilgisayar ekranında tanımlaması içinde ekran uzayı terimi kullanılır. Nesneler dönüşüm işlemlerinin bir arada kullanılmasıyla nesne uzayından iki boyutlu ekran uzayına tanımlanırlar.Yine geometrik hesaplamalar kullanılarak üç boyutlu resimlerin animasyonu iki boyutlu ekran üzerinde gerçekleştirilir.

Bir çok bilgisayar grafik uygulamalarında dönüşüm işlemleri yaygın olarak kullanılır. Robotların simülasyonu, eklemli figürlerin animasyonu gibi . homojen koordinatlar matrisi olarak isimlendirilen matematiksel gösterim matris çarpımları gibi bütün dönüşüm işlemlerinin yapılmasına olanak tanır. Bu metodla üç boyutlu döndürme işlemini anlatmak için 3x3 matrise 4. kolon ve satır eklenir. Aynı matrisle konum değiştirme de yapılabilir. Bütün dönüşüm işlemleri tek bir matrisle yapılabilir.Bir noktanın koordinatlarının gösterilmesinde P(x,y,z,w) veya iki boyutluda P(x,y,w) kullanılır. Bazı nedenlerle w "1" olarak alınır ve matris çarpımının bir sonucu olarak dönüşüm işlemleri üzerinde etkili olmaz.

3.2 İki Boyutlu Dönüşüm İşlemeri

3.2.1 Konum değiştirme

İki boyutlu konum değiştirme P(x y) noktasının koordinat değerlerine Tx,Ty konum değiştirme değerleri eklenerek gerçekleştirilir. Tx x ekseni yönünde,Ty y ekseni yönünde P noktasının haraketini belirtir (Şekil 3.1). Bu işlem :

$$Px' = Px + Tx$$

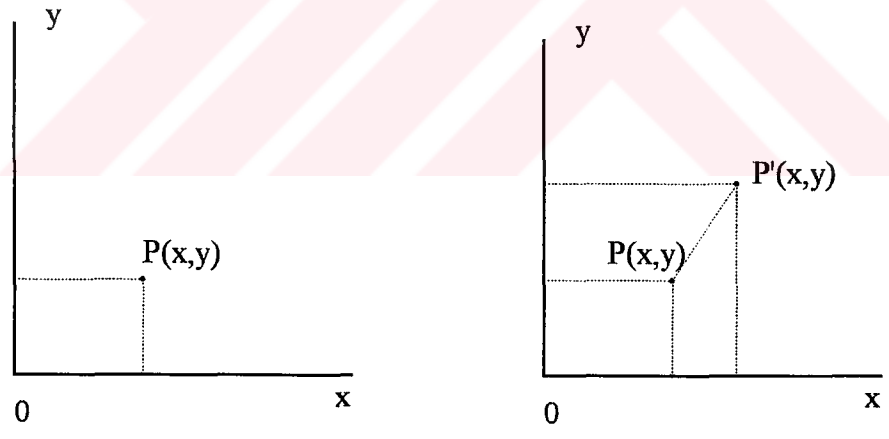
$$Py' = Py + Ty$$

Aynı işlem homejen koordinatlar matrisi şeklinde aşağıdaki gibi gösterilir.

$$P'(x,y,1) = P(x,y,1) * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

T konum değiştirme matrisini gösterirse;

$$P' = P * T$$



Şekil 3.1 İki boyutlu konum değiştirme

3.2.2 İki boyutlu ölçeklendirme

Bir resmin x, y koordinat ekseninde boyutlarının ölçeklendirilmesi için resmi oluşturan bütün noktalar aynı ölçeklendirme katsayıları Sx, Sy ile sırayla çarpılmalıdır. Şekil 3.2'de görüldüğü gibi bütün noktalar (P1..P4) ölçeklendirme katsayısı 2 ile (Sx ve Sy için) çarpılmıştır. Eğer örnek olarak P1 noktası ele alınırsa ölçeklendirme işlemi aşağıdaki şekilde yapılmış olur.

$$P_{1x}' \; P_{1x}' = P_{1x} * Sx$$

$$P_{1y}' = P_{1y} * Sy$$

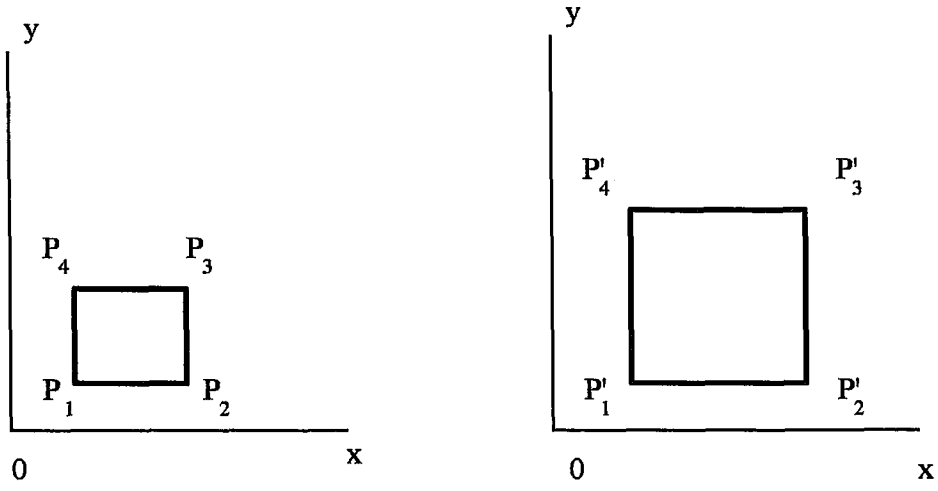
matris formunda yazarsak;

$$P_1'(x,y,1) = P_1(x,y,1) * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

kısa formda yazmak istersek;

$$P' = P * S$$

S ölçeklendirme matrisini göstermektedir.



Şekil 3.2 İki boyutlu ölçeklendirme

3.2.3 İki boyutlu döndürme

Orijindeki bir noktanın iki boyutlu döndürme işlemleri dönme açılarının sinüs ve cosinüs değerlerinin kullanılmasıyla yapılır. Eğer bir P(x,y) noktası orjinden saat yönünün tersi yönde θ acısı ile döndürülürse aşağıdaki eşitlikler yazılabilir. Bu durum şekil 3.3'de gösterilmiştir.

$$Px' = Px * Cos\theta - Py * Sin\theta$$

$$Py' = Px * Sin\theta - Py * Cos\theta$$

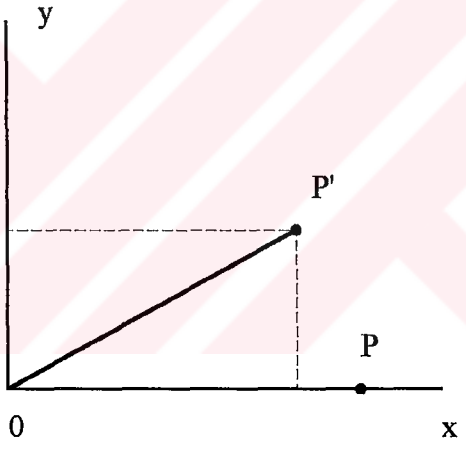
veya matris şeklinde yazarsak

$$P'(x,y,1) = P(x,y,1) * \begin{bmatrix} Cos\theta & Sin\theta & 0 \\ -Sin\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

R döndürme matrisi ise

$$P' = P * R$$

şeklinde bir ifade yazılabilir.

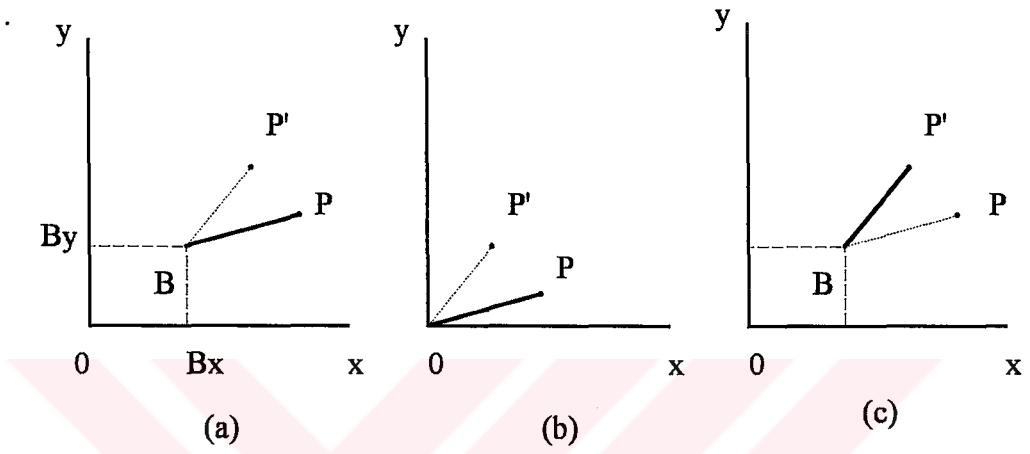


Şekil 3.3 iki boyutlu döndürme

3.2.4 Herhangi bir noktanın iki boyutlu döndürülmesi

Orijinde olmayan bir noktanın döndürülmesi işlemini yapmak robot

teknolojisinde ve robot ilişkili bilgisayar grafiklerinin uygulamalarında sıkça kullanılır. Bu döndürme işlemi şekil 3.4'de gösterildiği gibi üç adımda yapılabilir.



Şekil 3.4 İki boyutlu herhangi bir noktanın döndürülmesi.

Şekildeki P(x,y) noktasını B(x,y) noktası üzerinde α açısı kadar çevirmek istersek ilk iş olarak B noktasını orijine taşırız. Bu nedenle P(x,y) noktası -Bx ve -By değerleri kadar taşınmış olur. Bu durum matris formunda aşağıdaki gibi gösterilebilir.

$$T1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -Bx & -By & 0 \end{bmatrix}$$

Daha sonra döndürme işlemi bu taşınmış vektör çevirme matrisi R ile çarpılarak elde edilir.

$$R = \begin{bmatrix} Cos\alpha & Sin\alpha & 0 \\ -Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Sonuç olarak döndürülmüş nokta geri eski yerine taşınır. Bu $T1^{-1}$ ile tersine taşınma işlemi gerçkleştirilir.

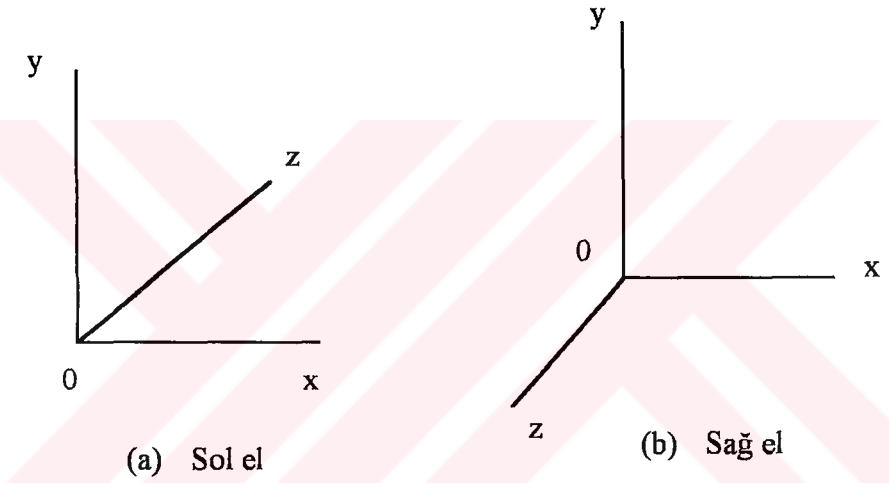
$$T1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Bx & By & 1 \end{bmatrix}$$

Yukarıda verilen üç dönüşüm matrisi şekil 3.4'deki dönüşüm işlemlerini yapmak içindir. Burada bu dönüşümlerle ilgili önemli bir faktör bu dönüşüm işlemlerinin yukarıda anlatıldığı şekliyle ve sırasıyla olması gerekliliğidir. Aksi halde dönüşüm işlemi gerçekleşmeyecektir.

3.3 Üç Boyutlu Geometri

Geçerli iki boyutlu tanıma bir üçüncü boyutun eklenmesiyle üç boyutlu koordinat sistemi elde edilir. Bu yeni boyut derinliği verir veya Z ekseni olarak isimlendirilir. Üç boyutlu grafikler de bir çok yönden iki boyutlu grafiklerin uzantısıdır. Üç boyutlu grafiklerin bilgisayar ekranında görüntülenmesi iki

boyutlu hale dönüştürülerek yapılır. Bu da üç boyutlu nesne görüntüsünün projeksiyon yöntemi ile bilgisayar ekranına izdüşürülmesi ile olur. Üç boyutlu bilgisayar grafiklerinde sol-el ve sağ-el koordinat sistemi olmak üzere iki çeşit koordinat sistemi kullanılır. Genellikle, nesnenin tanımlanmasında sağ-el, görüntülenmesinde ise pozitif z nokta değerlerinin içerde kaldığı sol-el koordinat sistemi tercih edilir(şekil 3.5).



Şekil 3.5 Koordinat sistemleri.

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

$$P = [wx \quad wy \quad wz \quad w]$$

şeklindedir. ikinci boyutta bir nokta, üç boyutlu homojen koordinatlarda kullanılacağı zaman w daima 1 olarak alınır. üç boyutlu nesnelerin ekranda görüntülenmesi durumunda w için farklı değerler kullanılabilir. Üç boyutlu homojen dönüşüm matrisinin iki boyutlu homojen dönüşüm matrisinden tek farkı sadece üçüncü boyut hesaplamalar için bir satır ve bir sütün eklenmiş olmasıdır.

3.3.1 Üç boyutlu konum değiştirme işlemi

Bir nesnenin konum değiştirmesi için nesneyi oluşturan bütün noktaların x,y,z koordinat değerlerine sırayla Tx, Ty, Tz öteleme miktarlarının eklenmesiyle yapılır. Üç boyutlu konum değiştirme denklemleri

$$Px' = \mathrm{Px} + \mathrm{Tx}$$

$$Py' = \mathrm{Py} + \mathrm{Ty}$$

$$Pz' = \mathrm{Pz} + \mathrm{Tz}$$

şeklindedir.

Konum değiştirme işleminin homojen koordinat matrisi şeklinde gösterimi aşağıdaki gibidir.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$$

Bir P(x,y,z) noktası için konum değiştirme işleminin gösterimi ise,

$$P'(x,y,z,1) = P(x,y,z,1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$$

şeklindedir. Tz'nin önceki değerine göre konum değiştirme işlemi z ekseni üzerindedir.

3.3.2 Üç boyutlu ölçeklendirme işlemi

Diğer işlemlerde olduğu gibi ölçeklendirme işlemi iki boyutludaki gibidir. Tek fark yeni ölçeklendirme katsayısı Sz dir. Bu durum matris formunda aşağıdaki gibi gösterilir.

$$S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Bir P(x,y,z) noktası için ölçeklendirme işleminin gösterimi ise,

$$P'(x,y,z,1) = P(x,y,z,1) * \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

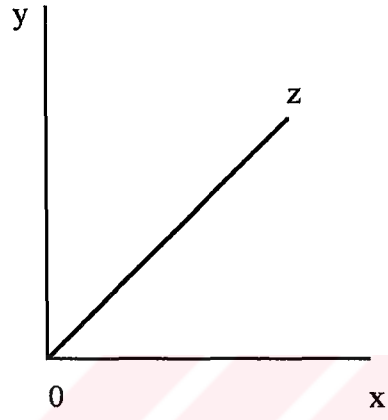
şeklindedir. Ölçeklendirme işleminde katsayılardan birinin 1'den küçük olması durumunda nesnenin boyu ilgili yönde küçülür. 1'den büyük olması durumunda nesnenin boyu ilgili yönde büyür. Nesnenin boyutunun her yönde eşit oranlarda büyütülmesi veya küçültülmesi, her üç ölçeklendirme katsayısının birbirine eşit olarak alınmasıyla sağlanır.

3.3.3 Üç boyutlu döndürme işlemi

Üç boyutluda da iki boyutlu döndürme işlemi geçerli olur. Bu nedenle eğer bir nesne üç boyutlu sistemde döndürülmek istenirse dönüş ekseni mutlaka belirtilmelidir. Ayrıca pozitif dönüş yönünün ne olacağı önemlidir. Bir eksen etrafında pozitif dönüş, döndürme ekseninin pozitif tarafından orijine doğru bakıldığında saat ibresinin dönüş yönünün tersi olarak belirlenir.

Üç eksen etrafında birbirinden 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. Bu nedenle bir z düzleminde döndürme işlemi iki boyutludur. Saat yönünün tersi yöndeki x, y, z eksenleri üzerindeki dönüşler aşağıda gösterilmiştir.



Şekil 3.6 Saat yönünün tersi yönde üç boyutlu döndürme

x ekseni etrafındaki dönüşte nesne noktalarının x değerleri sabit tutulur.

$$Rx = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\theta & Sin\theta & 0 \\ 0 & -Sin\theta & Cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y ekseni etrafındaki dönüşte nesne noktalarının y değerleri sabit tutulur.

$$Ry = \begin{bmatrix} Cos\theta & 0 & -Sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ Sin\theta & 0 & Cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

z ekseni etrafındaki dönüşte nesne noktalarının z değerleri sabit tutulur.

$$Rz = \begin{bmatrix} Cos\theta & Sin\theta & 0 & 0 \\ -Sin\theta & Cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

θx,θy,θz açılarına bağlı olarak bir nesnenin her üç eksene göre döndürülmesini sağlayan dönüşüm matrisi, her eksen için ayrı olarak yazılan döndürme matrislerinin çarpımı ile elde edilir.

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

4. ÜÇ BOYUTLU GÖRÜNTÜ OLUŞTURMA

Üç boyutlu nesne bilgisayar ekranında görüntülenene kadar bir çok değişik aşamalardan geçmek zorundadır. Burada bu aşamaları kısaca gözden geçirecegiz.

3B Görüntü İçin Dönüşümler (3D Viewing Transformations) : 3B'lu bilgisayar terimlerinden "dünya koordinatları uzayı", içinde yaşadığımız dünyayı tanımlar. Eğer biz bu uzayda bulunan bir nesneyi bilgisayar ekranında görüntülemek istersek aşağıdakileri tanımlamamız gerekir.

- Nesneye bakarken bulunduğumuz noktayı
- Bakış eğimi açısını
- Nesnede kesin olarak baktığımız noktayı (interest point)

Birde özel bir bakış noktasından görüntülenebilir kısmın belirlenmesi gerekir. Bütün tanımlamalar yapıldıktan sonra görüntülenecek resmin bilgileri doğrultusunda nesnenin bilgisayar ekranına aktarılma işlemine geçilir. Bu bilgiler ışığında 3B'lu dönüşümün ilk adımı tam bir dönüşüm için nesneler nesne uzayından ekran uzayına aktarılırlar.

Kesit Alma İşlemi (Clipping) : Gözün doğal özelliklerinden dolayı bakış alanının boyutları sınırlıdır. Çünkü arkamızda ne var göremeyiz. Bu

sınırlama bilgisayar ekranında kesme işlemiyle gerçekleştirilir. Nesnelerin üzerinde bakış alanı sınırları dışında kalan hatlara kesme işlemi uygulanır.

Görünüşün Ölçeklendirilmesi (Perspektive Scaling) : Nesnelerin doğal görünüşünü elde etmek, başka bir deyişle cismin ekrana uzaktaki görüntüsü küçük, yakındaki görüntüsü geniş görünüşlü olması için görünüşün ölçeklendirilmesi gerekir. Aksi halde nesnenin görüntüsü aslından farklı olur.

Görünmeyen Yüzeyin Kaldırılması (Hidden Surface Removal) : Nesnenin görünmeyen parcalarının işlemlerinin tamamlanması sırasında bunların görüntülenmesine gerek yoktur. Bu işleme "görünmeyen yüzeyin kaldırılması işlemi" denilir.

Resim İşleme (Rendering) : Bir diğer işlemde, benzer işleyişleri gerektiren "resim işleme" dir. Görüntülenemez hatlar veya yüzey kaldırılması yerine görüntüyü oluşturan her bir pixel görüntülebilir yüzey üzerinde olup olmadığı ayrı ayrı test edilir. Birde daha kaliteli resim için gölgeler ve diğer ışık ilişkili özellikler aynı tip işlemle sağlanır.

Görüntü Sistemi (Display System) : Sonuçta grafiklerin bilgileri bu değişik aşamalarda işlenip, çizimi ekranda tutan display kontrolörü ve frame buffer içeren display sistemine gönderilebilir olmalıdır.

4.1 Sanal Kamera (Virtual Camera)

Burada kullanılan kavramlar (Çavuşoğlu, A., 1993)'dan özetlenerek aktarılmıştır. Animasyon programcılarının başlıca problemlerinden birisi bilgisayar ekranında üzerinde güzel görüntü efeklerinin yapılabildiği bir görüntü penceresi yapacak piksel dizininin nasıl tanımlanması gerektiğine karar vermektir. Kaliteli resimlerin hazırlanması problemini, kullanıcı, gerçek kameranın simülasyonu ve özel efeklerin yardımıyla çözebilmektedir. Dijital kamera, sanal kamera, sentetik kamera terimleri aynı şeyi ifade ederler ve gerçek hayatta film çekimleri için kullanılan kameranın simülasyonunu veren aynı düşüncenin sonuçlarıdır. Sanal kamera terimi üç boyutlu bilgisayar grafiklerinde sıkça kullanılır ve üç boyutlu dünya koordinat sistemi içindeki bir nesnenin bilgisayar ekranı üzerinde iki boyutlu izdüşümünü alan dönüşüm işlemlerinin tümünü içeren bir program parçasının sonucudur (Magnenat and Thalmann, 1986).

Bir sanal kamera üç boyutlu dünya koordinatlarında tanımlanmış göz noktası ve bakılan nokta olarak isimlendirilmiş iki nokta ile tanımlanabilir. Sanal kameranın tanımı için kullanılan bazı yaygın bilgisayar terimleri aşğıda açıklanmıştır.

• *Dünya Koordinat Sistemi (World Coordinate System , WCS) :* Bazan nesne uzayı olarak kullanılsada nesnelerin tanımlandığı temel koordinat sistemidir.

• *Bakış Düzlemi (View Plane , VP) :*Üç boyutlu resmin izdüşümünün yapıldığı düzlem. Genellikle bilgisayar ekranı olarak alınır.

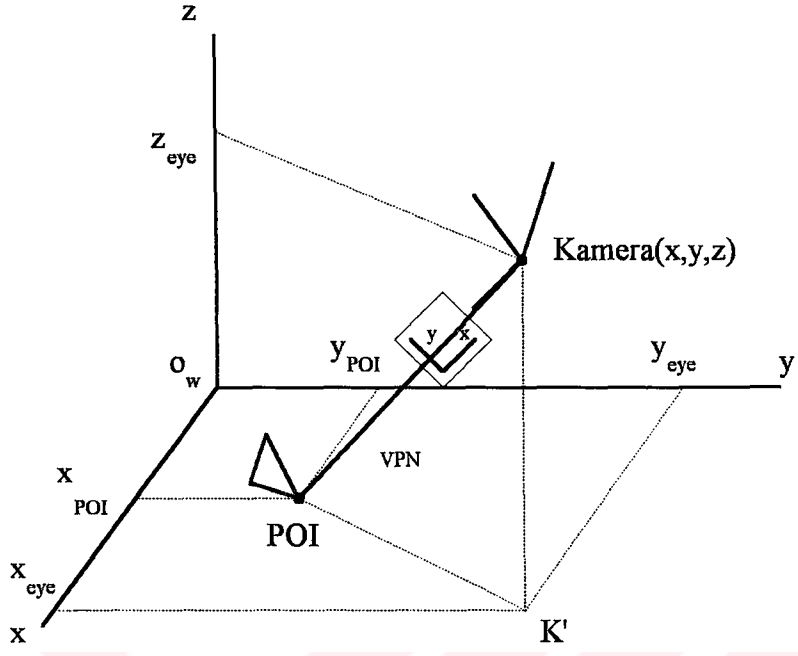
• *Bakış Düzlemi Koordinatları (View Plane Coordinates, VPC) :* Bilgisayar ekranı koordinat sistemidir. VP ile ilişkilidir.

• *İlgilenilen Nokta (Point of Interest , POI) :* Bakılan görüntü üzerindeki ilgilenilen merkez nokta. Bazan bakış pozisyonu olarakta tanımlanabilir. Eğer bu nokta değişirse ekrandaki görüntü direk olarak etkilenir.

• *Bakış Düzleminin Normali (View Plane Normal , VPN):*Bakış düzleminden ilgilenilen noktaya yönlendirilmiş bir vektördür ve bakış düzlemine diktir.

• *Bakış Düzleminin Mesafesi (View Plane Distance , VPD) :*Bakış VPN boyunca POI ve VP arasında ki uzaklığa bakış düzleminin mesafesi denilir.

• *Bakış Noktası (Viewing Point , VPT) :* Göz noktası olarakta bilinir. WCS ile ilişkili olarak kamera pozisyonu veya bakış pozisyonunun üç boyutlu koordinat değerleri ile tanımlandığı noktadır.



Şekil 4.1 Üç boyutlu bir görüntüde bakış.

Şekil 4.1'da üç boyutlu sanal kamera dünya koordinatlarında (WCS) yerleştirilmiştir. Bulunulan noktadan xy, yz ve xz yüzeylerine dik olan mesafeler kamera pozisyonunu verir. Z ekseni yukarı bakış yönünü gösterir ve POIχ , POIγ , POIz değerleri dünya koordinat sistemi (WCS) ile ilişkili verilmiştir. Bakış noktası normali (VPN) kamera objektifi ile ilgilenilen noktayı (POI)'i birbirine bağlar ve bilgisayar ekranına diktir. Kamera yönü POI'e doğrudur. Bilgisayar ekranı üzerinde sol alt köşe orijin olarak ve Xs noktaları sağa doğru Ys noktaları yukarı doğru alınmıştır. Bu dönüşümler kullanıldığında görüntü koordinatları bilgisayar ekranı koordinat sistemine (VPC) dönüştürülmüş olur. Bu işlemler birkaç adımda gerçekleştirilir. İlk

olarak orijin ilgi noktasına kaydırılır daha sonra P(Xc,Yc,Zc) değerleri alınarak bakış noktasına çevrilir. Xc ekseni üzerinde Yc ekseni Zwcs'yi kesene kadar koordinat sistemi çevirilir ve eksen sistemi de Xc ekseni etrafında Zc ekseni ilgilenilen noktayı gösterene kadar döndürülür. Sonuçta koordinat sistemini sol-el koordinat sistemine dönüştürmek için Xc ekseninin yönü ters çevirilir.Bu beş dönüşüm işlemi aşağıda görüldüğü gibi dört matris işlemi olarak ifade edilmiştir. Bunlar;

$$T1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -Xc & -Yc & -Zc & 1 \end{bmatrix}$$

$$T2 = \begin{bmatrix} Y'/L & -X'/L & 0 & 0 \\ -X'/L & Y'/L & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -Z'/D & -L/D & 0 \\ 0 & L/D & -Z'/D & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T4 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dönüşüm matrisleri içinde $P(Xc,Yc,Zc)$ dünya koordinat sistemiyle ilişkilendirilmiş üç boyutlu kamera koordinatları tanımına uygun gelir. POI ile kamera koordinatlarını birleştiren D doğrusu aşağıdaki gibi ifade edilebilir;

$$D = \sqrt{(Xc - X_{POI})^2 + (Yc - Y_{POI})^2 + (Zc - Z_{POI})^2}$$

D'nin x-y düzlemindeki yansıması olan L değeri ise ;

$$L = \sqrt{(Xc - X_{POI})^2 + (Yc - Y_{POI})^2}$$

şeklinde yazılır ve $P(X',Y',Z')$ 'nin değerleri de aşağıdaki gibi hesaplanır.

$$X' = (Xc - X_{POI})$$

$$Y' = (Yc - Y_{POI})$$

$$Z' = (Zc - Z_{POI})$$

Böylece dünya koordinatlarında verilen bir $P(X_w,Y_w,Z_w)$ noktası, aşağıdaki formül ile kamera koordinatlarına dönüştürülebilir.

$$P(X_c,Y_c,Z_c) = P(X_w,Y_w,Z_w) * T$$

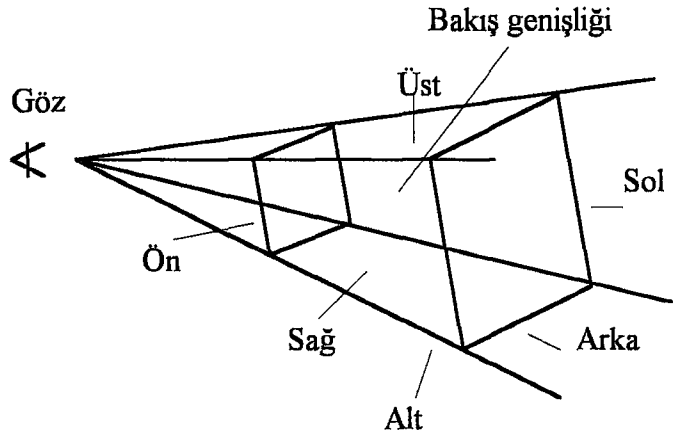
T yukarıda verilen dönüşüm işlemlerinin tümünü bir seri şeklinde gösteren dönüşüm matrisleridir.

$$P(X_c,Y_c,Z_c) = P(X_w,Y_w,Z_w) * (T_1,T_2,T_3,T_4)$$

Verilmiş bir noktanın yukarıdaki gibi kamera koordinatlarının bulunmasından sonra ekran koordinat değerleri de hesaplanabilir. Bizim görüntümüzü oluşturan noktaların bütün koordinat değerleri yukarıda tanımlandığı şekliyle hesaplanır. Kullanıcı kamera hareketlerini tanımladığı gibi bakış noktasını ve ilgilenilen noktayı da değiştirmekte serbestir. Bu durum değişken bakış şartlarının ortaya çıkmasını sağlar. Her bir çerçevenin hesaplanması sırasında program gerekli dönüşüm parametrelerini kontrol eder ve eğer onlar değişirse bir sonraki çerçeve yeni tanımlanan parametrelere göre hesaplanır.

4.2 3B'lu Kesit Alma İşlemi (Clipping)

Bir animasyon sistemi içerisinde üç boyutlu kesit alma işlemi, görüntüleme işlemlerinin temel parçasıdır. Kesit alma işlemleri ile görüntü oluşturmanın genliğini sınırlarken bazı istenmeyen etkiler olabilir. Örneğin negetif noktaların yansıması gibi veya kesit alma sınırları dışında kalan noktaların neden olduğu grafik ekran sınır değerleri ihlalleri resimleri bozabilir.



Şekil 4.2 Üç boyutlu sınırlı görüntü genliği.

Şekil 4.2'de görüldüğü gibi görüntü oluşturma genliği sağ, sol, üst ve alt kesme yüzeylerinden oluşur. Görüntü oluşturma algoritması görüntünün görüntülenebilir bölgelerini sınırlayan bu yüzeylerle nesnelerin mevcut hatlarını keser. Uzak noktalar için böyle bir sınırlama olmamasından dolayı bu görüntü oluşturma genliğiyle sonsuz bir görüş elde edilebilir. Görüntüyü daha gerçekçi yapmak için ön ve arka kesme yüzeyleri adı ile iki yeni kesme yüzeyi daha mevcut dört yüzeye eklenmiştir. Ön kesme yüzeyinin amacı görüntü yüzeyinin veya göz noktasının arkasında bulunan noktaların görüntülenmesini sağlamaktır. Arka kesme yüzeyi ise uzak görüntü genliğinde gerçeğe aykırı olarak görüntülenmiş olan noktaları kesmek için kullanılmıştır. Bu iki yüzey ile kesilmiş görüntü genliğine "sınırlı görüntü genliği" adı verilir.

3B'lu grafikte kesme işlemi genellikle 3B'lu göz koordinat sistemi içinde yapılır. Göz koordinatları içindeki kesme, kesme işleminden sonra görünür olmayan noktaların dönüşüm işlemlerini gerektirebilir. Fakat 3B'lu nesne uzayındaki kesme işlemlerinin zorlukları yanında bu durum önemsenmeyebilir (Hill, 1988).

4.3 Ekran Üzerinde Görüntü Oluşturma

Değişik görsel efekler elde etmek için değişik projeksiyon teknikleri kullanılabilir. Üç boyutlu bilgisayar grafiklerinde 3B'lu bir nesnenin projeksiyonu "planar geometric projections " olarak isimlendirilen bir dizi dönüşümle 2B'lu bilgisayar ekranı üzerinde noktaların izdüşümleri alınarak yapılır yani nesnenin z koordinatı atılır.

Projeksiyon teknikleri kullanım açısından paralel ve perspektif projeksiyon olarak iki gruba ayrılabilir. Projeksiyonda önemli olan, projeksiyon çizgilerinin projeksiyon düzlemi ile yaptığı acı, 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. Paralel projeksiyon da kendi aralarında gruplara ayrılmaktadır. Paralel projeksiyon

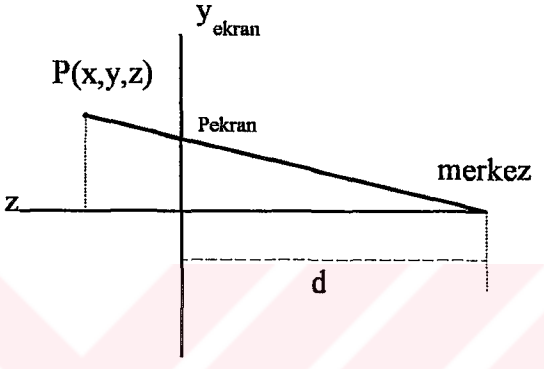
mimarlar, desinatörler, mühendisler ve diğer çizim işiyle uğraşanlar tarafından tercih edilir (Michenar, 1980).

4.3.1 Perspektif projeksiyon

Gerçek dünyada nesneler üç boyut içinde algılanır, fakat bilgisayar ekranında sadece iki boyut vardır (x,y). Bu nedenle üç boyutlu dünya koordinatlarındaki bir nesnenin iki boyutlu ekran veya görüntü koordinatlarına dönüştürmenin bir yolunu bulmaya gerek vardır. Nesnelerin uzaktaki görünüşlerinin gerçekçi olarak görüntülenmesi için küçük ve dar, ekrana yakın olan görünüşleri büyük ve geniş olmalıdır. Bu perspektif projeksiyonla yapılır. Perspektif projeksiyon bize daha gerçekçi bir görünüş sunmakla birlikte bu işlem uygulandığı zaman nesnelerin tam boyutları kaybolmaktadır. Bu nedenle yaygın olarak tanıtım, reklam, animasyon veya diğer benzer grafik uygulamalarında kullanılır.

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.

Eğer dünya koordinatlarında tanımlanmış bir $P(X_W, Y_W, Z_W)$ noktasının perspektif projeksiyonunu bulmak istersek, bir D mesafesinden noktanın bulunduğu yere bakılmış olsun bu projeksiyonun ekran üzerindeki oluşumu aşağıdaki şekilde gösterilmiştir.



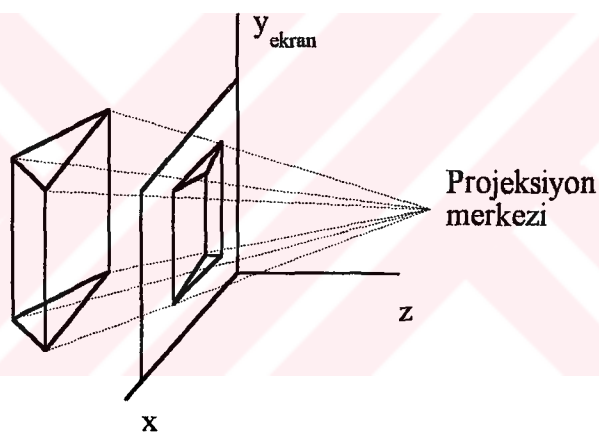
Şekil 4.3 Noktanın perspektif projeksiyonu

Daha önce gösterildiği gibi bir üç boyutlu noktanın kamera koordinatları önceki bölümde tanımlanan matrisler kullanılarak bulunabilir. Noktanın kamera koordinat değerleri hasaplandıktan sonra noktanın Px_S, Py_S iki boyutlu ekran koordinatları bulunabilir. Uzaktaki nesneleri küçük görünüşlü yakın nesneleri geniş görünüşlü yapmak ve istenen görüntüyü elde etmek için X_{eye} ve Y_{eye} değerleri Z_{eye} ile bölünür. Böylece;

$$\text{Xscn} = \left(\frac{D}{S}\right) * \left(\frac{X_{eye}}{Z_{eye}}\right)$$

$$Yscn = \left(\frac{D}{S}\right) * \left(\frac{Y_{eye}}{Z_{eye}}\right)$$

Xscn ve Yscn noktanın iki boyutlu ekran değerlerini gösterir. D perspektif mesafesi, S yarım ekran ölçüsüdür. Bu iki değerin birbirine oranı ölçeklendirme faktörü olarak yukarıdaki eşitliklerde kullanılmıştır. Bu oran perspektif ölçülerini değiştirmeden resmin ölçülerini değiştirir (Foley and VanDam, 1982).



Şekil 4.4 Perspektif projeksiyon.

4.4 Görünmeyen Yüzeyin Kaldırılması

Poligon yüzeylerle ifade edilen katı modellerin görüntülenmesindeki problemlerden biri belirlenmiş kamera pozisyonundan görüntülenemeyecek gizli yüzeylerin kaldırılma gerekliliğidir. Görünmeyen yüzeylerin kaldırılması ile katı cisimlerin görüntülenmesi için bir çok algoritma geliştirilmiştir.

İki farklı yaklaşım kullanılarak nesnelerin görünmeyen parçaları kaldırılabilir. Birinci grup algoritmalar "görüntü uzayı" algoritmalarıdır ve ekran uzayında çalışırlar. Kısaca görüntüyü oluşturan çerçevenin her pikseline onun yoğunluğu göz önünde bulundurularak karar verilir. Diger bir deyişle bu pikselde birden fazla yüzey çalışması varsa hangi yüzeyin görüntüleneceğini bulmak için kullanılır. Z buffer algoritması bu yaklaşıma örnek olarak verilebilir. "Nesne uzayı" algoritmaları nesne uzayında bulunan nesnelerin görüntülenebilirliğine karar verir. Bu algoritmalar başlangıçta nesne uzayında kullanılır fakat son görüntü, görüntü uzayında yer alır. Katı cisimlerin görünmeyen yüzeylerini kaldırarak görüntülenmesini sağlayan bir çok algoritma kullanılmaktadır. Animasyonda bir görüntü üç değişik formda görüntülenebilir.

- *Kafes-görüntü formu (Wire-frame form),* arka yüzeyler hariçinde kalan bütün yüzeyler görüntülenebilir.
- *Görünmeyen yüzeylerin kaldırıldığı form(Hidden surfaces removed form),* görünmeyen çizgiler kaldırılır.
- *Pürüzsüz şekillendirme formu(Smoothly rendered form),* gölgelendirme ve diğer yüzey bilgilerinin bulunabildiği formdur.

Wire-frame resimleri seri hareketlerin hızlı incelenmesi için kullanışlıdır. Nesnenin görünmeyen parcalarını bulmayı içeren fazladan hesaplama olmadığı için görüntüleme çok hızlı olabilir. Görüntüleme süresi özel grafik yetenekleri bulunmayan sıradan çalışma merkezlerinin hızına yakındır.

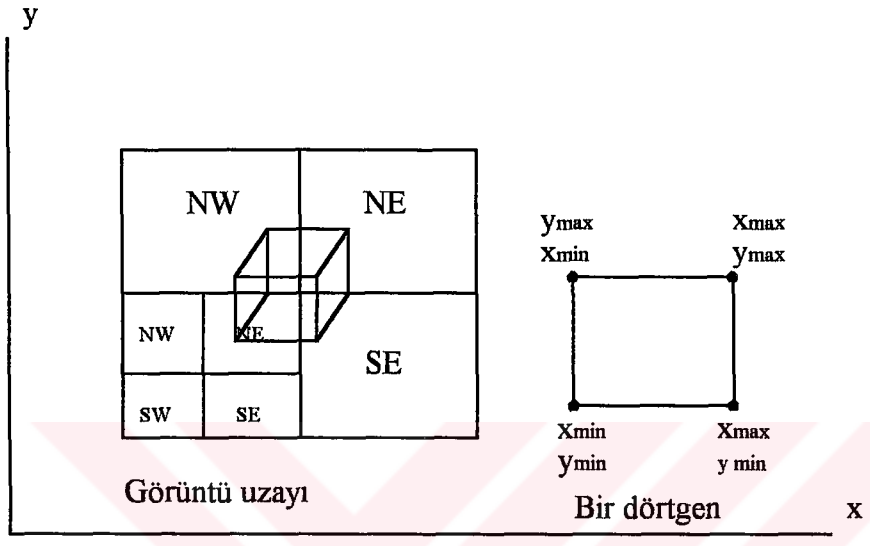
Görünmeyen yüzeylerin kaldırıldığı görünüşler, görüntüye daha güzel bir bakış sağlayabilir. Animatör animasyonu yapılmış nesnenin durumunu (ekran dışına taşıp taşmadığını) test edebilir.

En son görüntüleme formu rendered- formdur. Animatörün yansımalar, gölgeler ve yazılarıyla nesnelerin görüntülenebildiği son formdur. Sonuç görüntüleri bu formda üretilir. Animatörün net bir görüntü için en azından nesnenin görünmeyen yüzeylerinin kaldırılmasına ihtiyacı vardır.

4.4.1 Quadtree temeline göre görünmeyen yüzeylerin kaldırılması

Bu işlem her köşenin yüzey bilgilerinin test edilmesine dayalıdır. Bilgilerin test edilmesi işlemi için görüntü uzayının küçük dörtgenlere bölünmesi gerekir. Nesnenin her dörtgende görünebilirliğine karar vermek için bu küçük dörtgenler birbirinden ayrı olarak test edilir. Görüntü uzayının

defalarca küçük parçalara bölünmesi bu işlemi kolaylaştırır. Böyle bir



Şekil 4.5 Yardımcı dörtgenlerle bölünmüş bir görüntü.

algoritmayı açıklamak için görüntü yardımcı dörtgenlere bölünür ve yardımcıdörtgenler de içlerinde nesnenin bölümleri bulunana kadar tekrar kendi içlerinde yardımcı dörtgenlere bölünür. Bu küçük parçalara bölme işlemi piksel seviyesine ulaşana kadar devam eder. Bu noktada bir piksel olarak ifade edilen yüzey bilgisi bakış noktasına en yakın yüzeyi tanımlamak için test edilir. Bundan sonra seçilen yüzeyin rengi görüntülenir. Şekil 4.5'de göründüğü gibi görüntü dört ana dörtgene bölünmüştür. Her dörtgen kuzey-batı (North-west, NW), kuzey-doğu (North-east, NE), güney-batı (South-west, SW), güney-doğu (South-east,SE) isimleriyle etiketlenmiştir. Daha sonra

nesnenin X,Y koordinatlarındaki maksimum ve minumum uç değerleri (her köşenin Xmin, Xmax, Ymin, Ymax değerleri) bu köşelerde nesnenin uçlarının bulunup bulunmadığını tespit etmek için karşılaştırılır. Karşılaştırma sonucu olumlu ise küçük dörtgenlere bölme işlemine devam edilir. Şekilde SW'nin bölündüğü gibi. Bu işlem dörtgenin içi boş olana kadar yada iki köşenin kesiştiği noktayi içinde bulundurmayana kadar devam eder. Bu küçük parçalara bölme işlemini canlandırmak için, bölme işlemi her tekrarlandığında durumu ifade etmek için dörtgenler çizilir. Aşağıdaki eşitlikler bir dörtgeni içerebilecek yüzeylerin derinlik mesafelerini hesaplamak için kullanılabilir.

$$Ax + By + Cz + D = 0$$

$$-Cz = Ax + By + D$$

$$z = -(Ax + By + D) \div C$$

5. TURBO PASCAL'LA PROGRAMLAMA

5.1 Pascal Program Formatı

Bir pascal programı üç ana bloktan oluşur.

- Program başlığı bloğu
- Tanımlama bloğu
- Ana program bölümü

5.1.1 Program başlığı bloğu

Programın ilk satırı derleyici deyimlerini içerir. Bunlar derleme esnasında derleyiciye özel birtakım direktifler vermeye yararlar ve genelde {--} şeklinde ifade edilirler. Bunlar programın daha kolay çalıştırılmasını, hata bulma ve çıkarma işlemlerinin kolayca yapılmasını, programın belleğe sığmadığı zamanda sabit diski bellek olarak kullanmak gibi işlevleri yerine getirirler. Bu deyimlerin yazılıp yazılmaması kullanıcının arzusuna bağlıdır.

Programın ikinci satırı programın ismidir. Bu da kullanıcının arzusuna bağlı olarak yazılır veya yazılmaz. Bu durumun programın işleyişine bir etkisi olmaz. Program ismi programın yapısı hakkında bir fikir bir önbilgi versin diye kullanılabilir.

5.1.2 Tanımlama başlığı bloğu

Bu bölümde ana pogramın işleyişi sırasında kullanılacak bütün isimlerin tanımlanması gerekir. Bu tanımlama için aşağıda verilen bloklar kullanılır;

- Etiket tanımlama bloğu (Label)
- Sabit tanımlama bloğu (Const)
- Tip tanımlama bloğu (Type)
- Değişken tanımlama bloğu (Var)
- İşlem tanımlama bloğu (Procedure)
- Fonksiyon tanımlama bloğu (Function)

Bunların program içinde bulunma sıraları önemli değildir.

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

Sabit tanımlama bloğu (Const): Program içerisinde çokça tekrar edilen bir takım sabitlerin değerleri yerine onları temsil eden sabit isimlerin kullanılması okunabilirliği arttırdığı gibi programın yazımını da kolaylaştırır.

Tip tanımlama bloğu (Type): Pascal' da programcının standart veri tiplerine bağlı kalma zorunluluğu yoktur. Programcı standart veri tiplerinin yanı sıra kendi özel tiplerini oluşturma ve bunları program içinde kullanabilme imkanına sahiptir. program içerisinde kullanılacak her diziye bir tanımlama adı verilmesi gerekir. Diziye verilecek tanımlama adı değişken tanımlama kurallarına uygun olmalıdır.

Degişken tanımlama bloğu (Var): Program yapılırken kullanılacak bütün değişkenlerin tanımlandığı bloktur. Tanımlanmış bir değişken hafızada önceden ayrılmış kullanıma hazır yer demektir. Tanımlama sırasında değişkenin tipi belirtilmelidir. Bunlar tamsayı (integer), gerçel sayı (real), mantık (bolean), karakter (char), ve dizi (string) formlarında olabilir.

İşlem tanımlama bloğu (Procedure): Ana programdan gelen değerleri alıp işleyen ve sonuçlar üretebilen altprogramlardır. Altprogram, ana program tarafından kullanılan ve bağımsız işlem yapabilen program parcasıdır. 'Procedure' deyimi bir alt programın başlığını bildirir.

Fonksiyon tanımlama bloğu (Function): Fonksiyonlar birden fazla değeri alıp işlem yaparak bir tek değerin üretilmesi amacıyla tasarlanmış altprogramlardır. Bir fonksiyon ismi procedure gibi kendi başına bir komut

olarak kullanılamaz. Ancak bir ifade içerisinde veya komut cümlesi içnde işlemci olarak yer alabilir.

5.1.3 Ana program bölümü

Bu bölümde blok 'BEGIN' komutuyla başlayıp 'END.' komutuyla sona erer. Ana program bloğunun kullanılması zorunludur. Bütün altprogramlar burada çağrılarak kullanılabilir.

5.2 Pascal Grafik Sistemi

Turbo Pascal'da grafik çalışmak için öncelikle metin ekranının grafik ekranına dönüştürülmesi gerekir. Metin ekranının boyutları 25 satıra 80 sütun olarak belirlenmiştir. Grafik ekranındaysa bu değerler MaxX, MaxY kadardır ve çalışırken grafik moduna göre değişmektedir. Ekranı grafik için düzenleyen dosya 'GRAPH.TPU' dosyasıdır. Grafik amacı taşıyan programların başında bu dosya yüklenir.

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

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

Bu işlem sırasında meydana gelebilecek hatalar 'GraphResult' komutu ile kontrol edilir. Bu komut hata olduğu yada olmadığı zaman bir tam sayı üretir. Bu sayının değeri sıfır olması durumunda hata yok, negatif olması durumunda ise bir grafik hatası oluşmuştur demektir.

Grafik ekranından metin ekranına geçilmesi veya grafik sisteminin kapatılması için 'CloseGraph' komutu kullanılır.

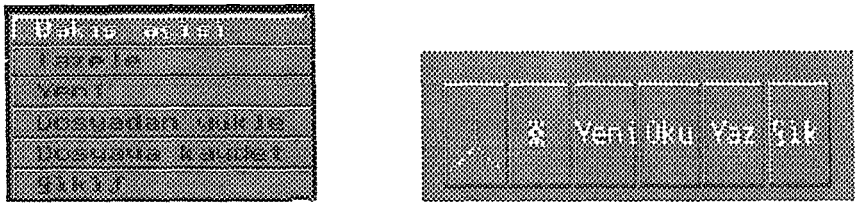
6. ANİMASYON PENCERESİNİN İNCELENMESİ

Ana pencere kullanıcının sürekli olarak etkileşimde bulunduğu bölümdür. Bu ana pencerede Şekil 6.2'de gösterildiği gibi kısayol düğmeleri, menülerin bulunduğu altpencereler, animasyonun gerçekleştirildiği dört ayrı pencere(ön görünüş, yan görünüş, üst görünüş, üç boyutlu perspektif görünüş), aktif pencere göstergesi, dosya adı, mouse pozisyon göstergesi, perspektif bakış açıları bulunmaktadır.

Mouse sağ tuşuna tıklayarak açılan menüler ile programın çalışması yönlendirilebilmektedir. Ayrıca kısayol düğmeleri ilede aynı yönlendirme yapılabilir. Bununla birlikte bazı alt işlemler için klavyeden basılacak tuşlar ilede yönlendirme yapılır.

6.1 Menü Seçme Altpenceresi Ve Kısayol Düğmeleri

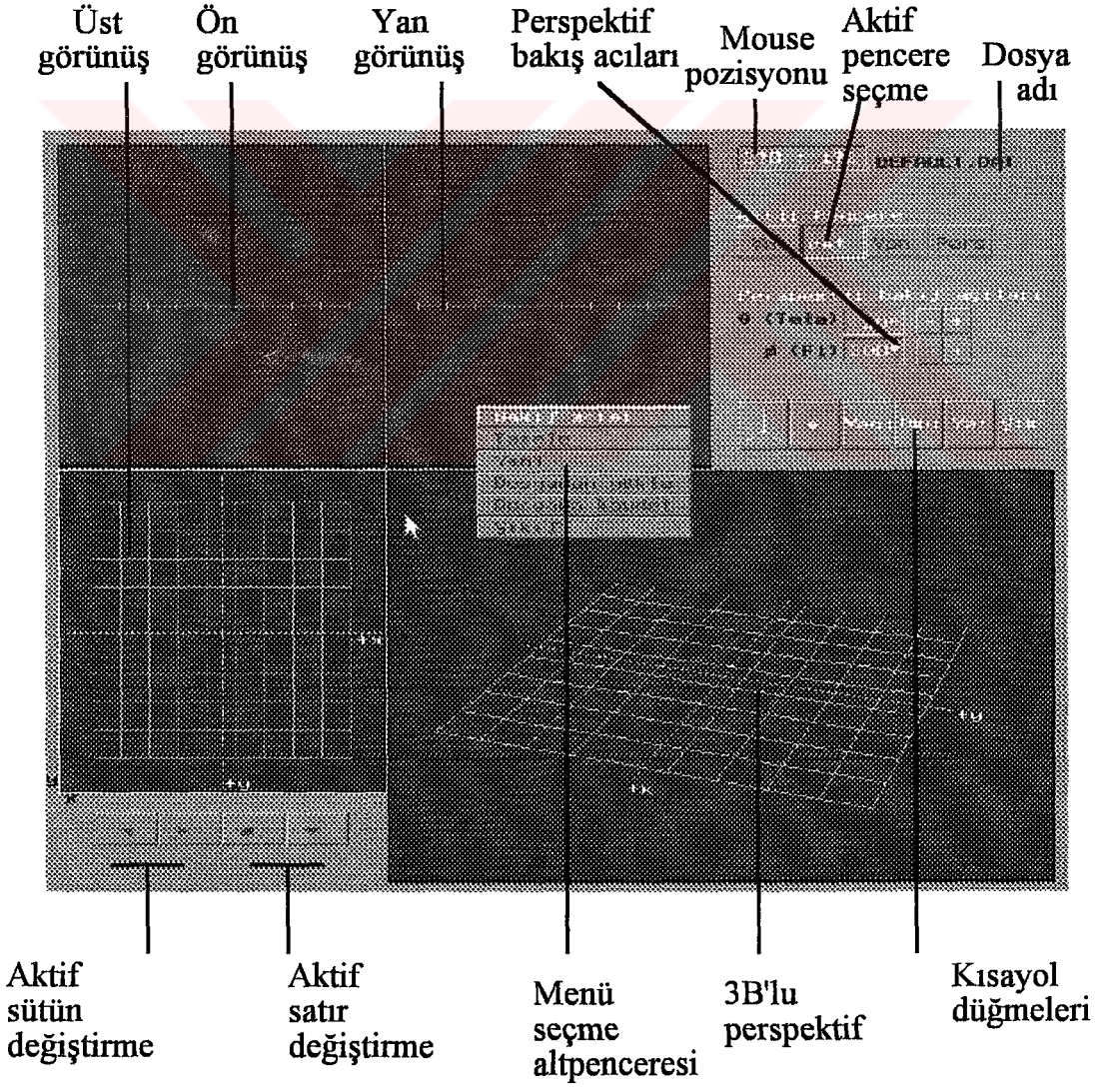
Kullanıcı ana pencerede iken mouse sağ tuşunu tıklayarak menü seçme altpenceresine geçebilir. Burada mouse sol tuşunu tıklayarak herhangi bir menüyü aktif hale getirebilir.



(a) (b)

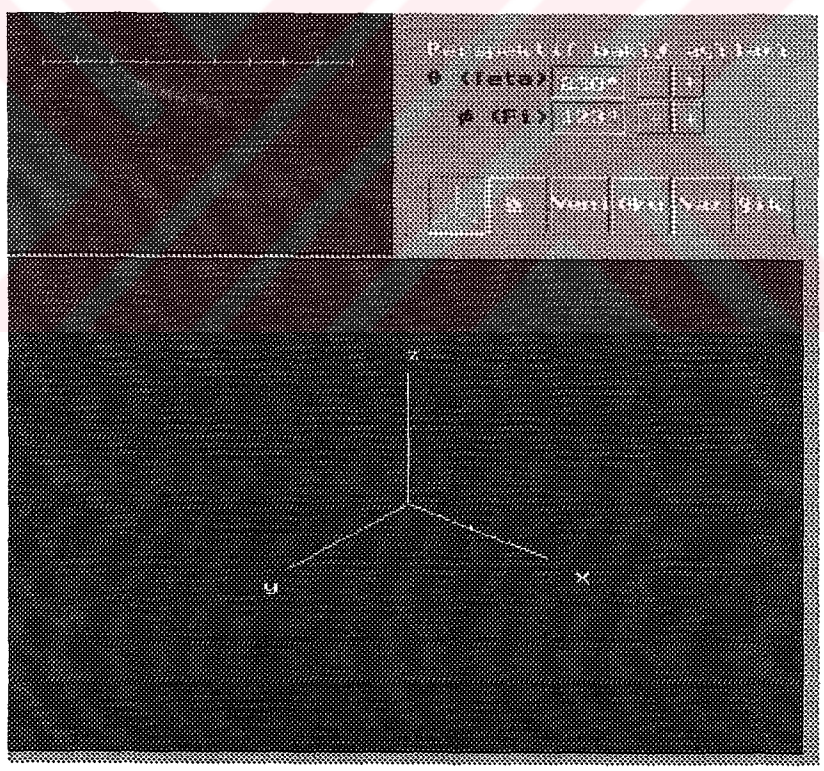
Şekil 6.1 (a) Menü seçme altpenceresi, (b) kısayol düğmeleri.

Menüde bulunan secenekler kısaca açıklanacak olunursa;



Şekil 6.2 Ana pencere

• ***Bakış açısı:*** Mouse okunu *bakış açısı* düğmesinin üzerine getirip mouse sol tuşunu tıklayarak bakış açısı menüsü seçilebilir. Şekil 6.3'de görüldüğü gibi kısayol düğmeside mouse sol tuşu ile tıklandığı andaki durumu alır. Aynı zamanda üç boyutlu perspektif görüntü penceresinde 3B'lu koordinat sistemi oluşur. Bundan sonra mouse haraket ettirilerek perspektif bakış açıları hızlıca değiştirilebilir. Perspektif bakış açıları kısayol düğmeleri kullanılarakta daha hassas bakış açıları değişikliği yapılabilir. Değiştirme işlemi tamamlandıktan sonra mouse sol tuşuna tıklanarak ana pencereye dönülür.



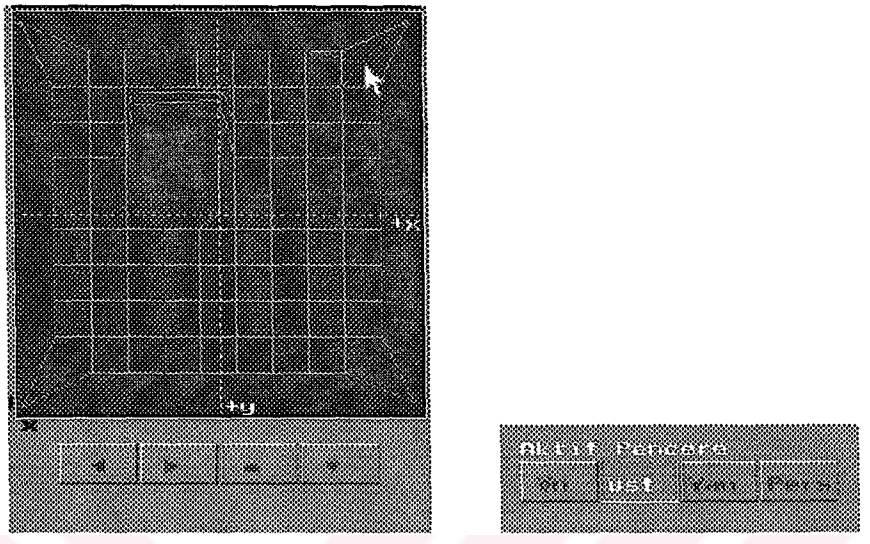
Şekil 6.3 Perspektif bakış açısı değiştirme

- ***Tazele:*** Mouse okunu *tazele* düğmesinin üzerine getirip mouse sol tuşunu tıklayarak tazele menü düğmesi seçilebilir. Mouse sol tuşuna tıklandıktan sonra kısayol düğmeside mouse sol tuşu ile tıklandığı andaki durumu alır ve bilgiler değişmeden ana pencere yeniden tazelenir.
- ***Yeni:*** Mouse yardımıyla *yeni* menüsü seçilirse ana pencerede eğer bilgiler varsa kaybolarak kullanıma hazır yeni bir ana pencere oluşturulur. Aynı zamanda *yeni* kısayol tuşuda tıklandığı andaki durumu alır.
- ***Dosyadan yükle:*** Bu menü ilede daha önceden dosya adı verilerek kayıt edilen animasyonu yapılmış nesne bilgileri aynı dosya adı seçilerek tekrar çağırılıp kullanılabilir. *Oku* kısayol düğmesine mouse sol tuşu ile tıklanırsa aynı işlemi gercekleştirmek mümkün olur.
- ***Dosyaya kaydet:*** Animasyonu yapılan nasneye ait bilgiler bu menü ile dosya adı verilerek hafızada saklanabilir. Bunun için menü seçildikten sonra verilecek dosya adını yazmak için bir pencerecik açılır. Buraya klavyeden dosya adı yazılıp enter tuşuna basılacak olursa .DAT uzantılı olarak bilgiler hafızada saklanır. Aynı işlem *yaz* kısayol düğmesi ilede yapılabilir.
- ***Çıkış:*** Bu düğmeye tıklandıktan sonra animasyon ana penceresi kapatılarak programdan çıkılır. *Çık* ksayol düğmesi ilede prgramdan çıkılabilir.

6.2 Üst, Ön, Yan, Ve Perspektif Görünüş Pencereleri

Bu pencereler nesnelerin animasyonunun yapıldığı pencerelerdir.

6.2.1 Üst görünüş penceresi



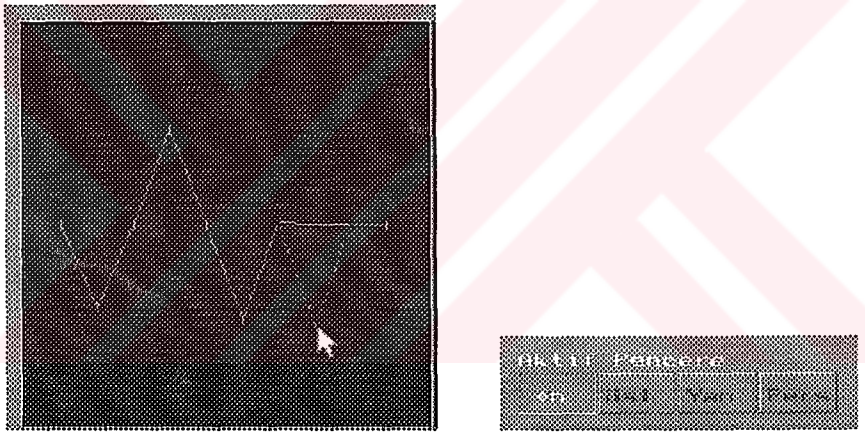
(a) (b)

Şekil 6.4 (a) Üst görünüş penceresi, (b) aktif pencere seçme düğmeleri

Kullanıcı öncelikle aktif hale gelmesini istediği üst pencere alanına veya pencere seçme butonlarından *üst* düğmesine mouse sol tuşu ile tıklayarak üst görüntü penceresini aktif hale getirir. Şekil 6.4 (a)'de görüldüğü gibi üst görünüş penceresi 100 ayrı noktadan oluşan üst görünüş hazırlama ızgarası ile aktif satır, sütün seçme düğmelerinden oluşmaktadır. Nesnenin üst görüntüsü burada oluşturulur. Kullanıcı mouse okunu istediği bir noktanın üstüne gelmesini sağlayarak mouse sol tuşu ile bu noktayı tutar ve istediği yere sürükleyerek götürüp bırakır. Yapılan değişiklik sürükleme işlemi bırakıldıktan sonra diğer görünüş pencerelerinde de kendini anında gösterir. Mouse sol tuşu

ile tutulup sürüklenen noktayı oluşturan çizgiler, kesik çizgiler halinde görünür. Sürükleme işlemi sırasında noktanın eski çizgileri ve yeri sarı renkli olarak görünür ve işlem bittikten sonra noktanın eski çizgileri ve yeri silinir. Seçilen noktanın satır ve sütünu aktif hale gelir ve kırmızı renkli olur. Ayrıca kısayol düğmeleri üzerine mouse sol tuşu ile tıklanarakta seçilmek istenen noktanın satır ve sütünu aktif hale getirilebilir.

6.2.2 Ön görünüş penceresi



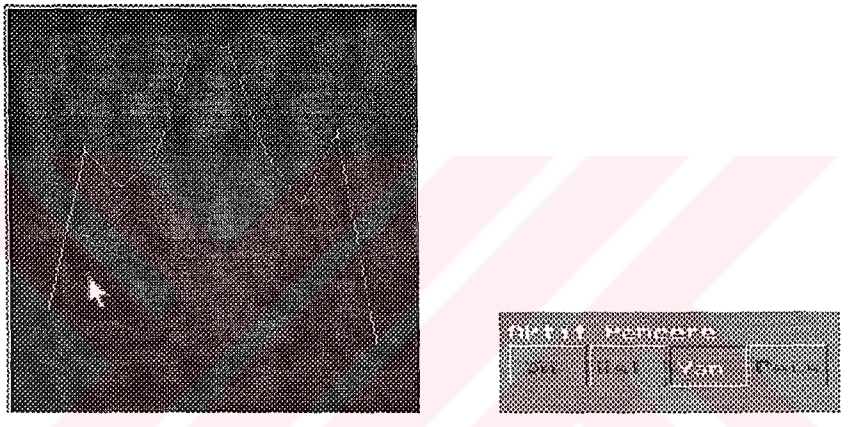
(a) (b)

Şekil 6.5 (a) Ön görünüş penceresi, (b) Aktif ön düğmesi

Kullanıcı öncelikle aktif hale gelmesini istediği ön pencere alanına veya pencere seçme butonlarından *ön* düğmesine mouse sol tuşu ile tıklayarak ön görüntü penceresini aktif hale getirir. Şekil 6.5 (a)'de görüldüğü gibi ön görünüş penceresi aktif olan satırda bulunan 10 ayrı noktadan oluşur. Aynı

zamanda aktif sutun üzerinde bulunan noktada kırmızı renkli olarak görülebilir. Kullanıcı kırmızı olan aktif noktayı mouse sol tuşu ile tıklayarak tutup aşağı veya yukarı istediği bir yere sürükleyip bırakabilir. İşlem bitirildiğinde değişiklik diğer pencerelerde de kendisini gösterir.

6.2.3 Yan görünüş penceresi



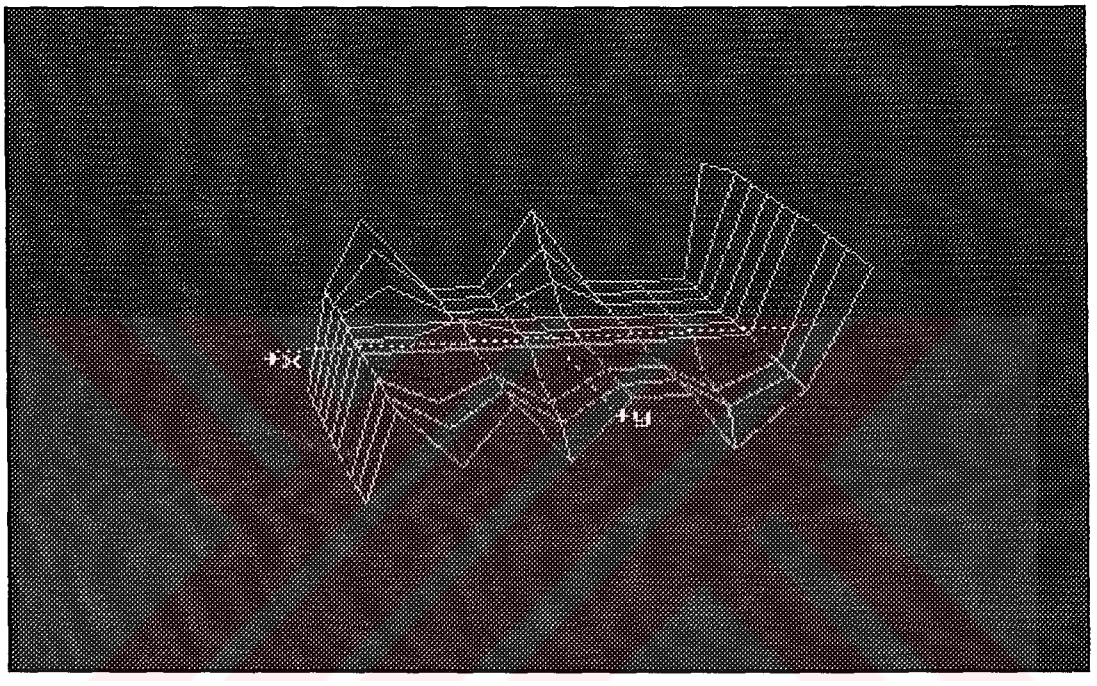
(a) (b)

Şekil 6.6 (a) Yan görünüş penceresi, (b) Aktif yan düğmesi

Kullanıcı öncelikle aktif hale gelmesini istediği yan pencere alanına veya pencere seçme butonlarından *yan* düğmesine mouse sol tuşu ile tıklayarak yan görüntü penceresini aktif hale getirir. Şekil 6.6 (a)'de görüldüğü gibi yan görünüş penceresi aktif olan sütünda bulunan 10 ayrı noktadan oluşur. Aynı zamanda aktif satır üzerinde bulunan noktada kırmızı renkli olarak görülebilir. Kullanıcı kırmızı olan aktif noktayı mouse sol tuşu ile

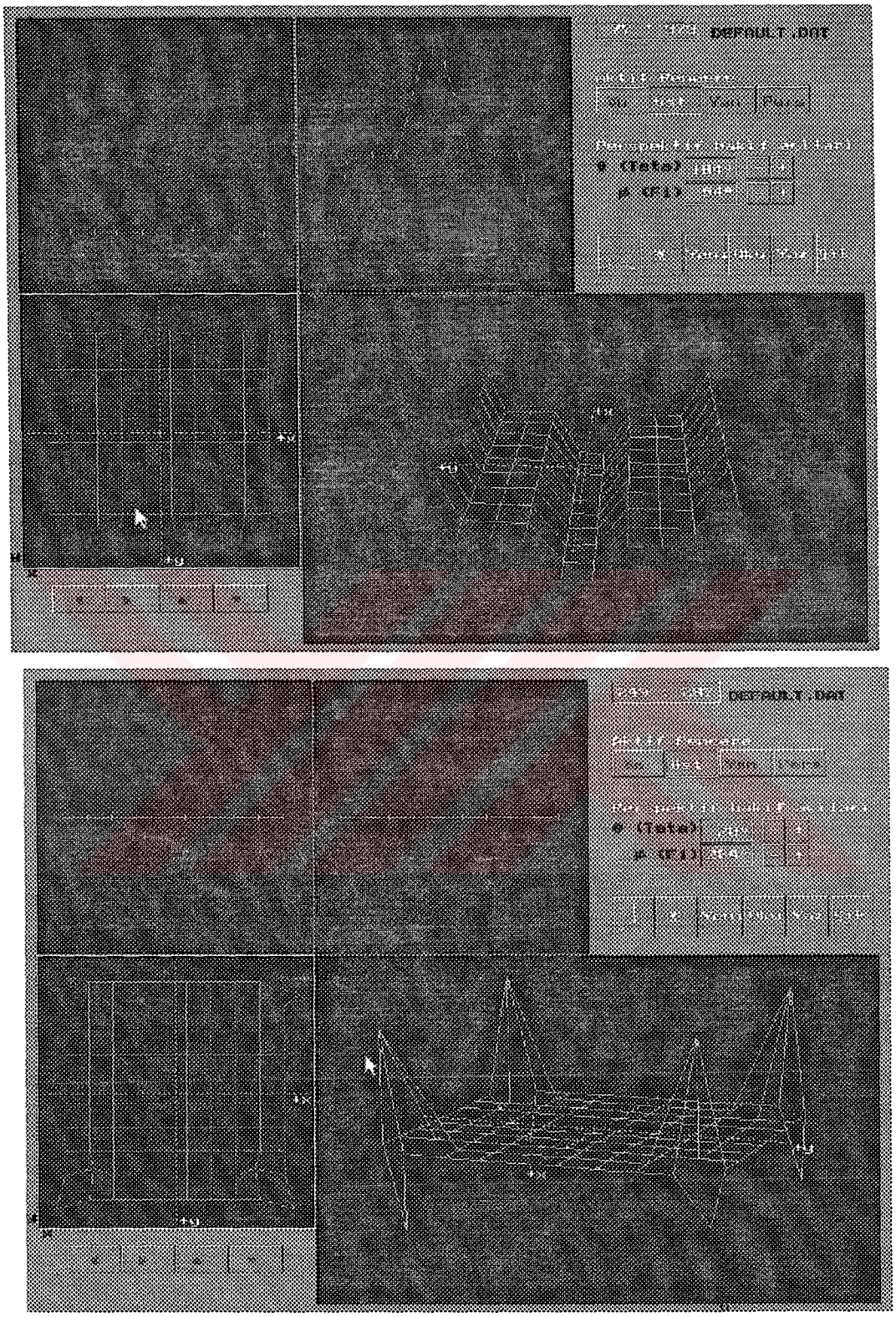
tıklayarak tutup aşağı veya yukarı istediği bir yere sürükleyip bırakabilir. İşlem bitirildiğinde değişiklik diğer pencerelerde de kendisini gösterir.

6.2.4 3B'lu perspektif görünüş penceresi



Şekil 6.7 3B'lu perspektif görünüş penceresi

Bu pencerede animasyonu yapılan nesnenin 3B'lu perspektif görünüşü çizilir. Kullanıcı bu pencereye doğrudan müdahalede bulunamaz. Fakat diğer pencerelerde yapacağı değişikliklerle buradaki görüntüyü değiştirebilir.



Şekil 6.8 Değişik uygulama örnekleri

7. SONUÇ VE TARTIŞMA

Bu çalışma ile ortaya çıkan sistem kullanıcı tarafından kolaylıkla öğrenilerek kullanılabilir. Tasarlanan 3B'lu bir animasyon ekranı ile kullanıcı kendi nesnelerini oluşturabildiği gibi ayrıca nesneyi oluşturan noktaların koordinatlarını da bilmesine gerek yoktur. Bu nedenle kullanıcılar iyi bilgisayar bilmeselerde sıkılmadan bu programı kullanabilirler. Sadece mouse ve bazı tuşları kullanarak değişik uygulamalar yapabilirler.

Bu çalışmada tasarlanarak kullanıcıya sunulan 3B'lu ızgara ekrandaki nokta sayısı daha da artırılarak animasyonlarda kullanılabilecek nesneler oluşturulabilecektir. Bu nesneler, animasyonu yapan kişinin becerisine bağlı olarak en basit bir şekilcikten karmaşık bir yer görüntüsüne kadar bir spekturum içerisinde yer alır. Ayrıca sisteme değişik resim işleme (rendering) mekanizmaları eklenerek oluşturulan nesnelerin katı görüntülerinin daha gerçekçi biçimde izlenmeleri sağlanabilir.

KAYNAKLAR

ALTAN, N., 1993, Turbo Pascal 7.0 ,Alfa, 262-319

ÇAVUŞOĞLU, A. , 1993, *Providing Language Facilities For 3D Articulated Figure Animation*, Ph.D.Thesis, University of Sussex, Brighton, 40-65

ÇAVUŞOĞLU, A., THOMAS, A.L. ,1994, '*Animating Multible Figures Using a Language Interperter*' In Melecon'94 Mediterranean Elecrotecnical Conferance April 12-14, 1994,Turkey

ERDUN, H., 1994, *Turbo ve Borland C & Pascal ile Grafik* , 321-354

FOLEY, J.D. , VanDam, A., 1982, *Fundementals of Interactive Computer Graphics*, Addison Wesley

GLASSNER, A.S., 1989, *3D Computer Graphics*, Herbert

HILL, F.S., 1988, *Computer Graphics*, McMillan

JAMSA, K., NAMEROFF, S.,1990, Turbo Pascal Programers Library,McGraw Hill

POKORNY, C., GERALD, C.F., 1988, *Computer Graphics*, Franclin,Beedle & Associates, 116-200

SAMET, H., TAMMINEN, M.,1985, *ACM Computer Graphics* ,121-130

SUTHERLAND, I.E., SPROULL, R.F., SCHUMACER, R.A., 1974, *ACM Computing Surveys*, 1-55

EKLER

```
{$M 16384,0,655360}
{N-}
uses crt,graph,mousetpu,cht012;
const max_x=10;max_y=10;tl=2;

desen:Fillpatterntype=(255,255,255,255,255,255,2
55,255);

 nokta_say=106;bag_say=182;
type
 point_type=record
   x,y,z:integer;
 end;
 but_type=record
   x,y:integer;
   z:boolean;
   st:string[5];
 end;

 win_point_type=record
   x,y,x1,y1:integer;
 end;
 point_array=array[1..max_x,1..max_y] of
point_type;
 gorunus=(ust,on,yan,per,alt);
{**********}
 koor=record
   x,y,z:real;
 end;
 piksel_tipi=record
   x,y:integer;
 end;
 bag_tipi=record
  bir,iki:byte;
 end;

 nokta_dizi=array[1..nokta_say] of koor;
 piksel_dizi=array[1..nokta_say] of piksel_tipi;
 bag_dizi=array [1..bag_say] of bag_tipi;

var
 win:array[ust..per] of win_point_type;
 yangor:array[1..max_y] of point_type;
 ongor:array[1..max_x] of point_type;
 point,ustgor:point_array;
 katsayi:integer;
 tmp_p,tmp_p1:point_type;
 i,k,gd,gm:integer;
 row,col,rw,cl:integer;
 mousefind:boolean;
 ch:char;
 test,son:boolean;
 ac_win,bak:gorunus;
 kx,ky:integer;

 l,cb:integer;
 nokta:nokta_dizi;
 piksel:piksel_dizi;

{ bag:bag_dizi;}
 d,teta,fi,s,msf:real;
 a_fi,a_teta:integer;
 tus:char;
 dosyaadi:string[11];

procedure ml;
begin
 sound(900);
 delay(10);
 nosound;
end;

procedure hm;
begin
 hiddenmousecursor;
end;
procedure sm;
begin
 showmousecursor;
end;

procedure writemousepos;
var s,s1:string;
begin
 if (tmp_p.x=getmousex) and
(tmp_p.y=getmousey) then exit;
 setfillpattern(desen,7);

bar(win[yan].x1+22,win[yan].y+2,win[yan].x1+98,
win[yan].y+13);
 str(getmousex:3,s);
 s1:=s;
 str(getmousey:3,s);
 s1:=s1+' : '+s;
 setcolor(white);
 outtextxy(win[yan].x1+24,win[yan].y+4,s1);
 tmp_p.x:=getmousex;
 tmp_p.y:=getmousey;
end;
procedure writefilename;
begin
 setfillpattern(desen,7);
 bar(520,15,620,30);
 setcolor(red);
 outtextxy(522,17,dosyaadi);
end;
procedure writeangle;
var s:string;
begin

 setcolor(white);
 outtextxy(435,100,'Perspektif bak□Ÿ a‡□lar□');
 setcolor(red);
 outtextxy(435,115,'é (Teta) ');
```

```
 outtextxy(435,135,' ĺ (Fi) ');
 box3d(500,112,538,128,true);
 box3d(500,131,538,147,true);
 setfillpattern(desen,7);
 bar(502,118,533,126);
 bar(502,133,533,145);
 str(a_teta:3,s);
 setcolor(white);
 outtextxy(505,118,s+'ø');
 str(a_fi:3,s);
 outtextxy(505,135,s+'ø');
end;



function deg_rad(aci:real):real;
begin  deg_rad:=(aci*3.141592654)/180; end;

function xe(x,y:real):real;
begin  xe:=-x*sin(teta)+y*cos(teta); end;

function ye(x,y,z:real):real;
begin ye:=-x*cos(teta)*cos(fi)-
y*sin(teta)*cos(fi)+z*sin(fi); end;

function ze(x,y,z:real):real;
begin ze:=-x*cos(teta)*cos(fi)-y*sin(teta)*cos(fi)-
z*sin(fi)+d; end;

function xs(xe,ze:real):integer;
begin
xs:=round(1.2*(((msf/s)*(xe/ze))*200+(win[per].x
+(win[per].x1-win[per].x)div 3))); end;

function ys(ye,ze:real):integer;
begin
 ys:=round(-
((msf/s)*(ye/ze))*200+(win[per].y+(win[per].y1-
win[per].y)div 2));end;

procedure x_dondur(aci:real);
begin
 aci:=(aci*(22/7))/180;
 for i:=1 to nokta_say do
 begin
   nokta[i].y:=nokta[i].y*cos(aci)-
nokta[i].z*sin(aci);

nokta[i].z:=nokta[i].y*sin(aci)+nokta[i].z*cos(aci);
 end;
end;
procedure y_dondur(aci:real);
begin
 aci:=(aci*(22/7))/180;
 for i:=1 to nokta_say do
 begin
  nokta[i].x:=nokta[i].x*cos(aci)-
nokta[i].z*sin(aci);

nokta[i].z:=nokta[i].x*sin(aci)+nokta[i].z*cos(aci);
 end;
end;
procedure z_dondur(aci:real);
begin
 aci:=(aci*(22/7))/180;
 for i:=1 to nokta_say do
 begin
  nokta[i].x:=nokta[i].x*cos(aci)-
nokta[i].y*sin(aci);

nokta[i].y:=nokta[i].x*sin(aci)+nokta[i].y*cos(aci);
 end;
end;
procedure otele(tx,ty,tz:real);
begin
 for i:=1 to nokta_say do
 begin
  nokta[i].x:=nokta[i].x+tx;
  nokta[i].y:=nokta[i].y+ty;
  nokta[i].z:=nokta[i].z+tz;
 end;
end;
procedure boyutlandir(sx,sy,sz:real);
begin
 for i:=1 to nokta_say do
 begin
  nokta[i].x:=nokta[i].x*sx;
  nokta[i].y:=nokta[i].y*sy;
  nokta[i].z:=nokta[i].z*sz;
 end;
end;
procedure noktalari_hesapla(n_say:byte);
var a,b,c:real;
begin
 for i:=1 to n_say do
 begin
  a:=xe(nokta[i].x,nokta[i].y);
  b:=ye(nokta[i].x,nokta[i].y,nokta[i].z);
  c:=ze(nokta[i].x,nokta[i].y,nokta[i].z);
  nokta[i].x:=a;nokta[i].y:=b;nokta[i].z:=c;
 end;
 for i:=1 to n_say do
 begin
  piksel[i].x:=xs(nokta[i].x,nokta[i].z);
  piksel[i].y:=ys(nokta[i].y,nokta[i].z);
 end;
end;

procedure p2_3;
var f:text;
begin
 gd:=0;
```

```
 for i:=1 to max_x do
  for k:=1 to max_y do
  begin
    gd:=gd+1;
    nokta[gd].x:=(((point[i,k].x)div katsayi)-
16.5)/1;
    nokta[gd].y:=(((point[i,k].y) div katsayi)-
16.5)/1;
    nokta[gd].z:=(point[i,k].z)/10;
  end;
  nokta[101].x:=-15; nokta[101].y:=0;
nokta[101].z:=0;
  nokta[102].x:=15; nokta[102].y:=0;
nokta[102].z:=0;
  nokta[103].x:=0; nokta[103].y:=-15;
nokta[103].z:=0;
  nokta[104].x:=0; nokta[104].y:=15;
nokta[104].z:=0;
  nokta[105].x:=15.5; nokta[105].y:=0;
nokta[105].z:=0;
  nokta[106].x:=0; nokta[106].y:=15.5;
nokta[106].z:=0;
end;

procedure bag_ciz;
begin
 gd:=0;
 setcolor(14);
 for k:=1 to max_x do
 begin
  for i:=1 to max_y-1 do
  begin
    gd:=gd+1;
    if (k=col) then setcolor(red) else setcolor(14);

line(piksel[gd].x,piksel[gd].y,piksel[gd+1].x,piksel
[gd+1].y);
  end;
  gd:=gd+1;
 end;
 gd:=0;
 for i:=1 to max_x-1 do
  for k:=1 to max_y do
  begin
    gd:=gd+1;
    if (k=row) then setcolor(red) else setcolor(14);

line(piksel[gd].x,piksel[gd].y,piksel[gd+max_y].x,
piksel[gd+max_y].y);
  end;

 setcolor(white);
 SetLineStyle(1,0,1);

line(piksel[101].x,piksel[101].y,piksel[102].x,pikse
l[102].y);

line(piksel[103].x,piksel[103].y,piksel[104].x,pikse
l[104].y);
 outtextxy(piksel[105].x,piksel[105].y,'+x');
 outtextxy(piksel[106].x,piksel[106].y,'+y');
 SetLineStyle(0,0,1);
end;

procedure ucd;
begin
 setfillpattern(desen,cyan);

bar(win[per].x,win[per].y,win[per].x1,win[per].y1);
 fi:=deg_rad(a_fi);teta:=deg_rad(a_teta);
 p2_3;
 noktalari_hesapla(nokta_say);
 bag_ciz;
end;
procedure data;
begin
   nokta[10].x:=0; nokta[10].y:=0; nokta[10].z:=0;
   nokta[1].x:=1; nokta[1].y:=0; nokta[1].z:=0;
   nokta[2].x:=0; nokta[2].y:=1; nokta[2].z:=0;
   nokta[3].x:=0; nokta[3].y:=0; nokta[3].z:=1;
   nokta[4].x:=-1; nokta[4].y:=0; nokta[4].z:=0;
   nokta[5].x:=0; nokta[5].y:=-1; nokta[5].z:=0;
   nokta[6].x:=0; nokta[6].y:=0; nokta[6].z:=-1;
   nokta[7].x:=1.2; nokta[7].y:=0; nokta[7].z:=0;
   nokta[8].x:=0; nokta[8].y:=1.2; nokta[8].z:=0;
   nokta[9].x:=0; nokta[9].y:=0; nokta[9].z:=1.2;
end;

procedure bag_ciz2(r:boolean);{r false ise siler}
begin
 if not r then setcolor(cyan);
 if r then begin
   if ((a_fi>180) and (a_fi<360)) or ((a_teta>90)
and (a_teta<270))
   then setcolor(red) else setcolor(white);
 end;

line(piksel[10].x,piksel[10].y,piksel[1].x,piksel[1].
y);
 outtextxy(piksel[7].x,piksel[7].y,'x');
 if r then begin
   if ((a_fi>180) and (a_fi<360)) or ((a_teta>180)
and (a_teta<360))
   then setcolor(red) else setcolor(white);
 end;

line(piksel[10].x,piksel[10].y,piksel[2].x,piksel[2].
y);
 outtextxy(piksel[8].x,piksel[8].y,'y');
 if r then begin
   if ((a_fi>90) and (a_fi<270)) then setcolor(red)
else setcolor(white);
 end;
```

```
line(piksel[10].x,piksel[10].y,piksel[3].x,piksel[3].
y);
  outtextxy(piksel[9].x,piksel[9].y,'z');
end;
procedure change_angle;
var i,j,k,l:integer;
begin
  setfillpattern(desen,cyan);

bar(win[per].x,win[per].y,win[per].x1,win[per].y1);
  d:=50;  msf:=200;   s:=10;
  fi:=deg_rad(a_fi);teta:=deg_rad(a_teta);
  data;
  noktalari_hesapla(10);
  bag_ciz2(true);
  writeangle;
  delay(100);
  k:=getmousex;l:=getmousey;
    repeat
      i:=getmousex;j:=getmousey;
      if i<>k then
        a_teta:=(a_teta+(i-k)*3)mod 360;
      if j<>l then
        a_fi:=(a_fi+(j-l)*3) mod 360;
      if a_teta<0 then a_teta:=360+a_teta;
      if a_fi<0 then a_fi:=360+a_fi;
      if (i<>k) or (j<>l) then
      begin
        writeangle;
        data;
        bag_ciz2(false);
        fi:=deg_rad(a_fi);teta:=deg_rad(a_teta);
        noktalari_hesapla(10);
        bag_ciz2(true);
        delay(50);
      end;
      k:=getmousex;l:=getmousey;
      if (k<10) or (k>630) then begin
setmousepos(320,l);k:=320;end;
      if (l<10) or (l>470) then begin
setmousepos(k,240);l:=240;end;

    until (getleftbutton) or (getrightbutton);
  d:=50;  msf:=15;   s:=10;

bar(win[per].x,win[per].y,win[per].x1,win[per].y1);
  ucd;
end;




procedure yaz;
var s,s1:string;
begin
  str(msf:5:1,s);
  s1:='msf:'+s;
  str(d:5:1,s);
  s1:=s1+' d:'+s;
  str(a_teta:3,s);
  s1:=s1+' t:'+s;
  str(a_fi:3,s);
  s1:=s1+' f:'+s;
  bar(win[yan].x1-
50,win[yan].y+110,win[yan].x1+198,win[yan].y+1
20);
  outtextxy(win[yan].x1-50,win[yan].y+110,s1);
end;



procedure real_to_scr_ust;
begin
  kx:=win[ust].x;ky:=win[ust].y;
  for i:=1 to max_x do
    for k:=1 to max_y do
    begin
      ustgor[i,k].x:=kx+point[i,k].x;
      ustgor[i,k].y:=ky+point[i,k].y;
    end;
end;

procedure scr_to_real_ust;
begin
  kx:=win[ust].x;ky:=win[ust].y;
  for i:=1 to max_x do
    for k:=1 to max_y do
    begin
      point[i,k].x:=ustgor[i,k].x-kx;
      point[i,k].y:=ustgor[i,k].y-ky;
    end;
end;

procedure real_to_scr_on;
begin
  kx:=win[on].x;
  ky:=win[on].y+((win[on].y1-win[on].y) div 2);
  for i:=1 to max_x do
  begin
    ongor[i].x:=point[i,row].x+kx;
    ongor[i].y:=point[i,row].z+ky;
  end;
end;

procedure scr_to_real_on;
begin
  kx:=win[on].x;
  ky:=win[on].y+((win[on].y1-win[on].y) div 2);
  for i:=1 to max_x do
  begin
```

```
{   point[i,row].x:=ongor[i].x-kx;}
   point[i,row].z:=ongor[i].y-ky;
  end;
end;

procedure real_to_scr_yan;
begin
  kx:=win[yan].x;
  ky:=win[yan].y+((win[yan].y1-win[yan].y) div 2);
  for i:=1 to max_y do
  begin
    yangor[i].x:=point[col,i].y+kx;
    yangor[i].y:=point[col,i].z+ky;
  end;
end;

procedure scr_to_real_yan;
begin
  kx:=win[yan].x;
  ky:=win[yan].y+((win[yan].y1-win[yan].y) div 2);
  for i:=1 to max_y do
  begin
{   point[col,i].y:=yangor[i].x-kx;}
   point[col,i].z:=yangor[i].y-ky;
  end;
end;

function find_point_ust(p:point_type;var
a,b:integer):boolean;
begin
  a:=0;b:=0;
  find_point_ust:=false;
  case ac_win of
    ust:begin kx:=win[ust].x;ky:=win[ust].y; end;
    on:begin end;
  end;
  for i:=1 to max_x do
    for k:=1 to max_y do
      if (ustgor[i,k].x-tl<=p.x) and (ustgor[i,k].y-
tl<=p.y)
        and (ustgor[i,k].x+tl>=p.x) and
(ustgor[i,k].y+tl>=p.y)then
      begin
        a:=i;
        b:=k;
        find_point_ust:=true;
        exit;
      end;
end;

function find_point_on(p:point_type;var
a:integer):boolean;
begin
  find_point_on:=false;
  for i:=1 to max_x do
  begin
    if (ongor[i].x-tl<=p.x)and (ongor[i].x+tl>=p.x)
and (ongor[i].y-tl<=p.y)
      and (ongor[i].y+tl>=p.y) then
    begin
      a:=i;
      find_point_on:=true;
      exit;
    end;
  end;
end;
function find_point_yan(p:point_type;var
a:integer):boolean;
begin
  find_point_yan:=false;
  for i:=1 to max_y do
  begin
    if (yangor[i].x-tl<=p.x)and
(yangor[i].x+tl>=p.x) and (yangor[i].y-tl<=p.y)
      and (yangor[i].y+tl>=p.y) then
    begin
      a:=i;
      find_point_yan:=true;
      exit;
    end;
  end;
end;

function find_win(p:point_type):gorunus;
var s:gorunus;
begin
  find_win:=ac_win;
  for s:=ust to per do
    if (p.x>=win[s].x) and (p.y>=win[s].y) and
      (p.x<=win[s].x1) and (p.y<=win[s].y1) then
    begin
      find_win:=s;
    end;
end;

procedure readfromfile(a:string);
var f:text;
begin
  if a='' then a:='DEFAULT.DAT';
  dosyaadi:=a;
  assign(f,a);
  reset(f);
  for i:=1 to max_x do
    for k:=1 to max_y do
    begin
      readln(f,point[i,k].x,point[i,k].y);
      point[i,k].z:=0;
    end;
    close(f);
end;
procedure writetofile(a:string);
var f:text;
begin
```

```
  assign(f,a);
  rewrite(f);
  gd:=0;
  for i:=1 to max_x do
   for k:=1 to max_y do
   begin
     writeln(f,' ',round(point[i,k].x/katsayi),' ',
                 round(point[i,k].y/katsayi),' ',
                 round(point[i,k].z/katsayi));
   end;
  close(f);
end;

procedure  katsayideg;
begin
  for i:=1 to max_x do
    for k:=1 to max_y do
    begin
      point[i,k].x:=point[i,k].x*katsayi;
      point[i,k].y:=point[i,k].y*katsayi;
      point[i,k].z:=point[i,k].z*katsayi;
    end;
end;

procedure load_z;
begin
  for i:=1 to max_x do
  begin
    ongor[i].x:=point[row,i].z;
    ongor[i].y:=point[row,i].z;
  end;
  for i:=1 to max_y do
  begin
    yangor[i].x:=point[i,col].z;
    yangor[i].y:=point[i,col].z;
  end;
end;
procedure drawline;forward;
procedure menubutton;
begin
 {****pencere d□§meleri*****}
  setcolor(white);
  outtextxy(win[yan].x1+20,win[yan].y+40,'Aktif
Pencere');

box3d(win[yan].x1+20,win[yan].y+50,win[yan].x1
+58,win[yan].y+70,ac_win=on);
  outtextxy(win[yan].x1+28,win[yan].y+58,'™n');

box3d(win[yan].x1+60,win[yan].y+50,win[yan].x1
+98,win[yan].y+70,ac_win=ust);
  outtextxy(win[yan].x1+64,win[yan].y+58,'šst');

box3d(win[yan].x1+100,win[yan].y+50,win[yan].x
1+138,win[yan].y+70,ac_win=yan);
  outtextxy(win[yan].x1+104,win[yan].y+58,'Yan');

box3d(win[yan].x1+140,win[yan].y+50,win[yan].x
1+178,win[yan].y+70,ac_win=per);
  outtextxy(win[yan].x1+144,win[yan].y+58,'Pers');

 {****‡izgi hareket d□§meleri***}

box3d(win[ust].x+20,win[ust].y1+15,win[ust].x+58
,win[ust].y1+35,false);
  outtextxy(win[ust].x+28,win[ust].y1+23,'
box3d(win[ust].x+60,win[ust].y1+15,win[ust].x+98
,win[ust].y1+35,false);
  outtextxy(win[ust].x+64,win[ust].y1+23,'
box3d(win[ust].x+100,win[ust].y1+15,win[ust].x+1
38,win[ust].y1+35,false);
  outtextxy(win[ust].x+104,win[ust].y1+23,' -');

box3d(win[ust].x+140,win[ust].y1+15,win[ust].x+1
78,win[ust].y1+35,false);
  outtextxy(win[ust].x+144,win[ust].y1+23,' ');
  {******bak□Ÿ d□§mesi*****}
  box3d(435,170,465,200,false);
  setcolor(white);

line(450,175,450,188);line(450,188,438,195);line(
450,188,462,195);
  {******tazele d□§mesi*****}
  box3d(467,170,497,200,false);
  setcolor(white);
  outtextxy(477,181,#15);
  {******yeni d□§mesi*****}
  box3d(500,170,530,200,false);
  setcolor(white);
  outtextxy(501,181,'Yeni');
  {******dosya a‡ d□§mesi*****}
  box3d(532,170,562,200,false);
  setcolor(white);
  outtextxy(536,181,'Oku');
  {******dosyaya kaydet d□§mesi*****}
  box3d(564,170,594,200,false);
  setcolor(white);
  outtextxy(568,181,'Yaz');
  {******‡□k□Ÿ d□§mesi*****}
  box3d(597,170,627,200,false);
  setcolor(white);
  outtextxy(599,181,'□□k');
  {******bak□Ÿ a‡□s□ +/- d□§meleri*****}
  box3d(546,112,562,128,false);
  box3d(564,112,580,128,false);
  box3d(546,131,562,147,false);
  box3d(564,131,580,147,false);
  setcolor(white);
  outtextxy(551,117,'-');
  outtextxy(569,117,'+');
  outtextxy(551,136,'-');
```

```
 outtextxy(569,136,'+');

end;
procedure refresh;forward;
procedure prc_bakac;
 begin
  hiddenmousecursor;
  box3d(435,170,465,200,true);
  setcolor(white);

line(450,175,450,188);line(450,188,438,195);line(
450,188,462,195);
  change_angle;
  box3d(435,170,465,200,false);
  showmousecursor;
 end;
procedure  prc_yeni;
 begin
  hm;
  ml;
  box3d(500,170,530,200,true);
  delay(25);
  if not yes_no(150,170,'Bilgiler yenilensin mi?')
then
  begin
   readfromfile('');;
   katsayideg;
   real_to_scr_ust;
   real_to_scr_on;
   refresh;
  end;
  box3d(500,170,530,200,false);
  refresh;
  sm;
 end;

procedure prc_dosyaac;
var a:string;
begin
 hm;
 a:=select_file(100,100,'*.dat',red,lightgray);
 readfromfile(a);
 katsayideg;
 refresh;
 sm;
end;
procedure prc_dosyakaydet;
var a:string;
   p:pointer;
   s:Word;
   x,y:integer;
   t:boolean;
begin
 x:=400;
 y:=220;
 hm;
 t:=false;
 s:=Imagesize(x,y,630,300);
 GetMem(p,s);
 GetImage(x,y,630,300,p^);
 box(x+5,y+5,620,295,lightgray);
 wrtxy(x+10,y+10,'Dosya ad□n□ girin ',red);
 wrtxy(x+100,y+40,'.DAT',red);
 repeat
  a:=inputgr(x+30,y+40,8,blue,darkgray);
  for i:=9 downto 1 do if a[i]=' ' then delete(a,i,1);
  a:=a+'.DAT';
  if (a<>'') then
  begin
   if fileexist(a) then
   begin
    t:=yes_no(130,300,a+' zaten var. šzerine
yazmak istiyor musunuz ?');
    if not t then
    begin
     dosyaadi:=a;
     writetofile(a);
     t:=true;
     writefilename;
    end else t:=false;
   end
   else
   begin
    dosyaadi:=a;
    writetofile(a);
    t:=true;
    writefilename;
   end;
  end else t:=true;
 until t;
 putimage(x,y,p^,NormalPut);
 sm;
end;


procedure menubuttonselect;
var ac:gorunus;
begin
 {*****pencere hareket d□§meleri***}
 ac:=ac_win;
 if
mouse_in(win[yan].x1+20,win[yan].y+50,win[yan]
.x1+58,win[yan].y+70) then
  ac_win:=on
 else
 if
mouse_in(win[yan].x1+60,win[yan].y+50,win[yan]
.x1+98,win[yan].y+70) then
  ac_win:=ust
 else
 if
mouse_in(win[yan].x1+100,win[yan].y+50,win[yan
].x1+138,win[yan].y+70) then
  ac_win:=yan
```

```
 else
 if
mouse_in(win[yan].x1+140,win[yan].y+50,win[yan
].x1+178,win[yan].y+70) then
   ac_win:=per;
 if ac<>ac_win then
 begin
   ml;
   hiddenmousecursor;
   setcolor(7);
   rectangle(win[ac].x-2,win[ac].y-
2,win[ac].x1+2,win[ac].y1+2);
   menubutton;
   setcolor(white);
   rectangle(win[ac_win].x-2,win[ac_win].y-
2,win[ac_win].x1+2,win[ac_win].y1+2);
   showmousecursor;
   exit;
 end;
 {*********************************}
 {*****‡izgi hareket d□§meleri******}
 setfillpattern(desen,cyan);
 if
mouse_in(win[ust].x+20,win[ust].y1+15,win[ust].x
+58,win[ust].y1+35) then
 begin
   hiddenmousecursor;

box3d(win[ust].x+20,win[ust].y1+15,win[ust].x+58
,win[ust].y1+35,true);
   delay(20);

box3d(win[ust].x+20,win[ust].y1+15,win[ust].x+58
,win[ust].y1+35,false);
   showmousecursor;
   if col>1 then
   begin
     col:=col-1;

bar(win[yan].x,win[yan].y,win[yan].x1,win[yan].y1
);
     drawline;
   end;
   exit;
 end;
 if
mouse_in(win[ust].x+60,win[ust].y1+15,win[ust].x
+98,win[ust].y1+35) then
 begin
   hiddenmousecursor;

box3d(win[ust].x+60,win[ust].y1+15,win[ust].x+98
,win[ust].y1+35,true);

   delay(20);

box3d(win[ust].x+60,win[ust].y1+15,win[ust].x+98
,win[ust].y1+35,false);
   showmousecursor;
   if col<max_x then
   begin
     col:=col+1;

bar(win[yan].x,win[yan].y,win[yan].x1,win[yan].y1
);
     drawline;
   end;
   exit;
 end;
 if
mouse_in(win[ust].x+100,win[ust].y1+15,win[ust].
x+138,win[ust].y1+35) then
 begin
   hiddenmousecursor;

box3d(win[ust].x+100,win[ust].y1+15,win[ust].x+1
38,win[ust].y1+35,true);
   delay(20);

box3d(win[ust].x+100,win[ust].y1+15,win[ust].x+1
38,win[ust].y1+35,false);
   showmousecursor;
   if row>1 then
   begin
     row:=row-1;

bar(win[on].x,win[on].y,win[on].x1,win[on].y1);
     drawline;
   end;
   exit;
 end;
 if
mouse_in(win[ust].x+140,win[ust].y1+15,win[ust].
x+178,win[ust].y1+35) then
 begin
   hiddenmousecursor;

box3d(win[ust].x+140,win[ust].y1+15,win[ust].x+1
78,win[ust].y1+35,true);

   delay(20);

box3d(win[ust].x+140,win[ust].y1+15,win[ust].x+1
78,win[ust].y1+35,false);
   showmousecursor;
   if row<max_y then
   begin
     row:=row+1;

bar(win[on].x,win[on].y,win[on].x1,win[on].y1);
     drawline;
   end;
```

```
  exit;
end;

{******bak□Ÿ d□§mesi*****}
if mouse_in(435,170,465,200) then prc_bakac;
{******bak□Ÿ a‡□s□ +/- d□§meleri*****}

if mouse_in(546,112,562,128) then
begin
  hm;
  box3d(546,112,562,128,true);
  delay(25);
  a_teta:=(a_teta-1) mod 360;
  if a_teta<0 then a_teta:=360+a_teta;
  writeangle;
  box3d(546,112,562,128,false);
  ucd;
  sm;
  exit;
end;
if mouse_in(564,112,580,128) then
begin
  hm;
  box3d(564,112,580,128,true);
  delay(25);
  a_teta:=(a_teta+1) mod 360;
  writeangle;
  box3d(564,112,580,128,false);
  ucd;
  sm;
  exit;
end;
if mouse_in(546,131,562,147) then
begin
  hm;
  box3d(546,131,562,147,true);
  delay(25);
  a_fi:=(a_fi-1) mod 360;
  if a_fi<0 then a_fi:=360+a_fi;
  writeangle;
  box3d(546,131,562,147,false);
  ucd;
  sm;
  exit;
end;
if mouse_in(564,131,580,147) then
begin
  hm;
  box3d(564,131,580,147,true);
  delay(25);
  a_fi:=(a_fi+1) mod 360;
  writeangle;
  box3d(564,131,580,147,false);
  ucd;
  sm;
  exit;
end;
{*********************************}
{******‡□k□Ÿ d□§mesi*****}
if mouse_in(597,170,627,200) then
begin
  ml;
  hm;
  box3d(597,170,627,200,true);
  delay(25);
  son:=yes_no(150,170,'□□kmak istediginizden
emin misiniz ?');
  box3d(597,170,627,200,false);
  sm;
  exit;
end;
{******yeni d□§mesi*****}
if mouse_in(500,170,530,200) then prc_yeni;
{******tazele d□§mesi*****}
if mouse_in(467,170,497,200) then
begin
  hm;
  box3d(467,170,497,200,true);
  delay(25);
  cleardevice;

box3d(win[yan].x1+20,win[yan].y,win[yan].x1+10
0,win[yan].y+15,true);
  writeangle;
  refresh;
  box3d(467,170,497,200,false);
  sm;
end;
{******ac d□§mesi*****}
if mouse_in(532,170,562,200) then
begin
  hm;
  box3d(532,170,562,200,true);
  delay(25);
  prc_dosyaac;
  box3d(532,170,562,200,false);
  sm;
end;
{******kaydet d□§mesi*****}
if mouse_in(564,170,594,200) then
begin
  hm;
  box3d(564,170,594,200,true);
  delay(25);
  prc_dosyakaydet;
  box3d(564,170,594,200,false);
  sm;
end;

end;
procedure cerceve;
begin
  menubutton;
  setcolor(1);
```

```
 setfillpattern(desen,cyan);
 bar(win[ust].x,win[ust].y,win[ust].x1,win[ust].y1);
 bar(win[on].x,win[on].y,win[on].x1,win[on].y1);

bar(win[yan].x,win[yan].y,win[yan].x1,win[yan].y1
);

bar(win[per].x,win[per].y,win[per].x1,win[per].y1);
 rectangle(win[ust].x-1,win[ust].y-
1,win[ust].x1+1,win[ust].y1+1);
 rectangle(win[on].x-1,win[on].y-
1,win[on].x1+1,win[on].y1+1);
 rectangle(win[yan].x-1,win[yan].y-
1,win[yan].x1+1,win[yan].y1+1);
 rectangle(win[per].x-1,win[per].y-
1,win[per].x1+1,win[per].y1+1);
 setcolor(white);
 rectangle(win[ac_win].x-2,win[ac_win].y-
2,win[ac_win].x1+2,win[ac_win].y1+2);
 setcolor(1);
 outtextxy(win[ust].x+2,win[ust].y1+2,'x');
 outtextxy(win[ust].x-10,win[ust].y1-10,'y')
end;
procedure xyeksen;
begin
 setcolor(white);
 SetLineStyle(1,0,1);
 line(110,216,110,414);
 line(11,316,209,316);
 outtextxy(112,408,'+y');
 outtextxy(195,317,'+x');
 SetLineStyle(0,0,1);
end;
procedure drawline;
begin
 xyeksen;
 kx:=win[ust].x;ky:=win[ust].y;
 setcolor(11);
 real_to_scr_ust;
 for k:=1 to max_y do
  for i:=1 to max_x-1 do
  begin
   if row=k then setcolor(12) else setcolor(11);

line(ustgor[i,k].x,ustgor[i,k].y,ustgor[i+1,k].x,ustgo
r[i+1,k].y);
  end;

 for i:=1 to max_x do
  for k:=1 to max_y-1 do
  begin
   if col=i then setcolor(12) else setcolor(11);

line(ustgor[i,k].x,ustgor[i,k].y,ustgor[i,k+1].x,ustgo
r[i,k+1].y);
  end;

 real_to_scr_on;
 for i:=1 to max_x-1 do
 begin
  setcolor(11);
  line(ongor[i].x,
ongor[i].y,ongor[i+1].x,ongor[i+1].y);
  if i=col then setcolor(12) else setcolor(14);
  line(ongor[i].x,ongor[i].y-
2,ongor[i].x,ongor[i].y+2);
 end;
  if (i+1)=col then setcolor(12) else setcolor(14);
 line(ongor[i+1].x,ongor[i+1].y-
2,ongor[i+1].x,ongor[i+1].y+2);

 real_to_scr_yan;
 for i:=1 to max_y-1 do
 begin
  setcolor(11);

line(yangor[i].x,yangor[i].y,yangor[i+1].x,yangor[i
+1].y);
  if i=row then setcolor(12) else setcolor(14);
  line(yangor[i].x,yangor[i].y-
2,yangor[i].x,yangor[i].y+2);
 end;
 if (i+1)=row then setcolor(12) else setcolor(14);
 line(yangor[i+1].x,yangor[i+1].y-
2,yangor[i+1].x,yangor[i+1].y+2);
 ucd;
end;
procedure refresh;
begin
 SetLineStyle(0,0,1);
 setbkcolor(7);
 setcolor(11);
 cerceve;
 drawline;
 writefilename;
end;

procedure prc_ust;
begin
 real_to_scr_ust;
     if find_point_ust(tmp_p,cl,rw) then
     begin

setmouserange(win[ac_win].x+1,win[ac_win].y+1,
          win[ac_win].x1-1,win[ac_win].y1-1);

      SetLineStyle(1,0,1);
      row:=rw;col:=cl;
      tmp_p1:=ustgor[col,row];

setmousepos(ustgor[col,row].x,ustgor[col,row].y);
      test:=true;
      repeat
       writemousepos;
```

```
        if test then
        begin
        hiddenmousecursor;
        xyeksen;
        SetLineStyle(0,0,1);
        setcolor(yellow);
        if row>1 then
          line(ustgor[col,row-1].x,ustgor[col,row-
1].y,
               ustgor[col,row].x,ustgor[col,row].y);
        if col>1 then
          line(ustgor[col-1,row].x,ustgor[col-
1,row].y,
               ustgor[col,row].x,ustgor[col,row].y);
        if row<max_y then

linc(ustgor[col,row+1].x,ustgor[col,row+1].y,
               ustgor[col,row].x,ustgor[col,row].y);
        if col<max_x then

linc(ustgor[col+1,row].x,ustgor[col+1,row].y,
               ustgor[col,row].x,ustgor[col,row].y);
        tmp_p:=tmp_p1;

        setcolor(cyan);
        if row>1 then
          line(ustgor[col,row-1].x,ustgor[col,row-
1].y,
               tmp_p1.x,tmp_p1.y);
        if col>1 then
          linc(ustgor[col-1,row].x,ustgor[col-
1,row].y,
               tmp_p1.x,tmp_p1.y);
        if row<max_y then

linc(ustgor[col,row+1].x,ustgor[col,row+1].y,
               tmp_p1.x,tmp_p1.y);
        if col<max_x then

linc(ustgor[col+1,row].x,ustgor[col+1,row].y,
               tmp_p1.x,tmp_p1.y);
        tmp_p1.x:=getmousex;
        tmp_p1.y:=getmousey;
        setcolor(lightcyan);
        SetLineStyle(1,0,1);
        if row>1 then
          line(ustgor[col,row-1].x,ustgor[col,row-
1].y,
               tmp_p1.x,tmp_p1.y);
        if col>1 then
          line(ustgor[col-1,row].x,ustgor[col-
1,row].y,
               tmp_p1.x,tmp_p1.y);
        if row<max_y then

linc(ustgor[col,row+1].x,ustgor[col,row+1].y,
               tmp_p1.x,tmp_p1.y);
        if col<max_x then

line(ustgor[col+1,row].x,ustgor[col+1,row].y,
               tmp_p1.x,tmp_p1.y);
        end;
        if not test then
        begin
          tmp_p.x:=getmousex;
          tmp_p.y:=getmousey;
        end;
        if (tmp_p.x<>tmp_p1.x) or
(tmp_p.y<>tmp_p1.y) then
          test:=true else test:=false;
        showmousecursor;
        until not getleftbutton;
        ustgor[col,row].x:=tmp_p1.x;
        ustgor[col,row].y:=tmp_p1.y;
        scr_to_real_ust;
        hiddenMouseCursor;
        refresh;
        ShowMouseCursor;
       end;
  setmouserange(0,0,640,480);
end;
procedure prc_on;
begin
  real_to_scr_on;

  if find_point_on(tmp_p,cl) then
  begin
    setmouserange(ongor[cl].x,win[ac_win].y+1,
                  ongor[cl].x,win[ac_win].y1-1);
    SetLineStyle(1,0,1);
    col:=cl;
    tmp_p1:=ongor[col];
    setmousepos(ongor[col].x,ongor[col].y);
    test:=true;
    repeat
      writemousepos;
      if test then
      begin
        hiddenmousecursor;
        setcolor(yellow);
        SetLineStyle(0,0,1);
        if col<max_x then
line(ongor[col].x,ongor[col].y,

ongor[col+1].x,ongor[col+1].y);
        if col>1 then line(ongor[col].x,ongor[col].y,
                           ongor[col-1].x,ongor[col-1].y);

        tmp_p:=tmp_p1;
        setcolor(cyan);
        if col<max_x then
line(tmp_p1.x,tmp_p1.y,ongor[col+1].x,ongor[col+
1].y);
```

```
      if col>1 then
linc(tmp_p1.x,tmp_p1.y,ongor[col-1].x,ongor[col-
1].y);
      tmp_p1.x:=getmousex;
      tmp_p1.y:=getmousey;
      setcolor(lightcyan);
      SetLineStyle(1,0,1);
      if col<max_x then
linc(tmp_p1.x,tmp_p1.y,ongor[col+1].x,ongor[col+
1].y);
      if col>1 then
linc(tmp_p1.x,tmp_p1.y,ongor[col-1].x,ongor[col-
1].y);
     end;
     if not test then
     begin
      tmp_p.x:=getmousex;
      tmp_p.y:=getmousey;
     end;
     if (tmp_p.x<>tmp_p1.x) or
(tmp_p.y<>tmp_p1.y) then
        test:=true else test:=false;
      showmousecursor;
      until not getleftbutton;
      ongor[col].x:=tmp_p1.x;
      ongor[col].y:=tmp_p1.y;
      scr_to_real_on;
        hiddenmousecursor;
        refresh;
        ShowMouseCursor;
  end;
  setmouserange(0,0,640,480);
end;
procedure prc_yan;
begin
  real_to_scr_yan;

  if find_point_yan(tmp_p,rw) then
  begin
    setmouserange(yangor[rw].x,win[ac_win].y+1,
              yangor[rw].x,win[ac_win].y1-1);
    SetLineStyle(1,0,1);
    row:=rw;
    tmp_p1:=yangor[row];
    setmousepos(yangor[row].x,yangor[row].y);
    test:=true;
    repeat
      writemousepos;
      if test then
      begin
        hiddenmousecursor;
        setcolor(yellow);
        SetLineStyle(0,0,1);
        if row<max_x then
linc(yangor[row].x,yangor[row].y,

yangor[row+1].x,yangor[row+1].y);
      if row>1 then
line(yangor[row].x,yangor[row].y,
                    yangor[row-1].x,yangor[row-
1].y);
      tmp_p:=tmp_p1;
      setcolor(cyan);
      if row<max_x then
line(tmp_p1.x,tmp_p1.y,yangor[row+1].x,yangor[r
ow+1].y);
      if row>1 then
line(tmp_p1.x,tmp_p1.y,yangor[row-
1].x,yangor[row-1].y);
      tmp_p1.x:=getmousex;
      tmp_p1.y:=getmousey;
      setcolor(lightcyan);
      SetLineStyle(1,0,1);
      if row<max_x then
line(tmp_p1.x,tmp_p1.y,yangor[row+1].x,yangor[r
ow+1].y);
      if row>1 then
line(tmp_p1.x,tmp_p1.y,yangor[row-
1].x,yangor[row-1].y);
     end;
     if not test then
     begin
      tmp_p.x:=getmousex;
      tmp_p.y:=getmousey;
     end;
     if (tmp_p.x<>tmp_p1.x) or
(tmp_p.y<>tmp_p1.y) then
        test:=true else test:=false;
      showmousecursor;
      until not getleftbutton;
      yangor[row].x:=tmp_p1.x;
      yangor[row].y:=tmp_p1.y;
      scr_to_real_yan;
        hiddenMouseCursor;
        refresh;
        ShowMouseCursor;

  end;
  setmouserange(0,0,640,480);
end;

begin

  InitGraph(gd, gm,");
  if GraphResult <> grOk then
    Halt(1);
  readfromfile(");
  win[ust].x:=10;  win[ust].y:=215;
win[ust].x1:=210; win[ust].y1:=415;
  win[on]. x:=10;  win[on]. y:=10;   win[on].
x1:=210; win[on].y1:=210;
  win[yan].x:=215; win[yan].y:=10;
win[yan].x1:=415;  win[yan].y1:=210;
```

```
 win[per].x:=215; win[per].y:=215;
win[per].x1:=630;  win[per].y1:=472;
 bak:=on;
 d:=20;
 msf:=11;
 s:=10;
 d:=50;  msf:=20;   s:=10;
 a_fi:=300;a_teta:=20;
 katsayi:=6;
 row:=5;
 col:=1;
 katsayideg;
 ac_win:=ust;
 setbkcolor(7);
 clcardevice;
 real_to_scr_ust;
 real_to_scr_on;
 refresh;writeangle;
{
box(win[yan].x1+20,win[yan].y,win[yan].x1+100,
win[yan].y+15,3);moupos}

box3d(win[yan].x1+20,win[yan].y,win[yan].x1+10
0,win[yan].y+15,true);
 mousefind:=mouseinit;
 kx:=win[ust].x;ky:=win[ust].y;
 load_z;
 setbkcolor(7);
 son:=true;
 if mousefind then
 begin
  showmousecursor;
  repeat
   tmp_p.x:=getmousex;
   tmp_p.y:=getmousey;
   writemousepos;
   if getleftbutton then
   begin
    tmp_p.x:=getmousex;
    tmp_p.y:=getmousey;
    menubuttonselect;
    if find_win(tmp_p)<>ac_win then
    begin
     hm;
     setcolor(7);
     rectangle(win[ac_win].x-2,win[ac_win].y-
2,win[ac_win].x1+2,win[ac_win].y1+2);
     ac_win:=find_win(tmp_p);
     setcolor(white);
     rectangle(win[ac_win].x-2,win[ac_win].y-
2,win[ac_win].x1+2,win[ac_win].y1+2);
     menubutton;
     sm;
    end;

    if ac_win=ust then
    begin
     prc_ust;
    end
    else if ac_win=on then
    begin
     prc_on;
    end
    else if ac_win=yan then
    begin
     prc_yan;
    end
   end;

   if getrightbutton and not test then
   begin
{    hiddenmousecursor;}
    mouse1:=true;
    if getmousex>400 then
setmousepos(400,getmousey);
    i:=menu(getmousex,getmousey,' Bak□Ÿ
a‡□s□ _ Tazele _ Yeni _ Dosyadan y□kle _
Dosyaya kaydet _ □□k□Ÿ ',blue,white);
    showmousecursor;
    if i=2 then begin refresh; end;
    if i=6 then begin
hm;son:=yes_no(150,170,'□□kmak istediginizden
emin misiniz ?');sm;end;
    if i=1 then prc_bakac;
    if i=3 then prc_yeni;
    if i=4 then prc_dosyaac;
    if i=5 then prc_dosyakaydet;
   end;
   ch:=#32;
   if keypressed then ch:=readkey;
   if ch=#27 then begin
hm;son:=yes_no(150,170,'□□kmak istediginizden
emin misiniz ?');sm;end;
  until not son;
 end;
 closegraph;
 scr_to_real_ust;scr_to_real_on;

end.
```

Bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

.....

Tez Yöneticisi

Yrd.Doç.Dr. Abdullah ÇAVUŞOĞLU

Bu çalışma, jürimiz tarafından Elektronik ve Bilgisayar Eğitimi Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

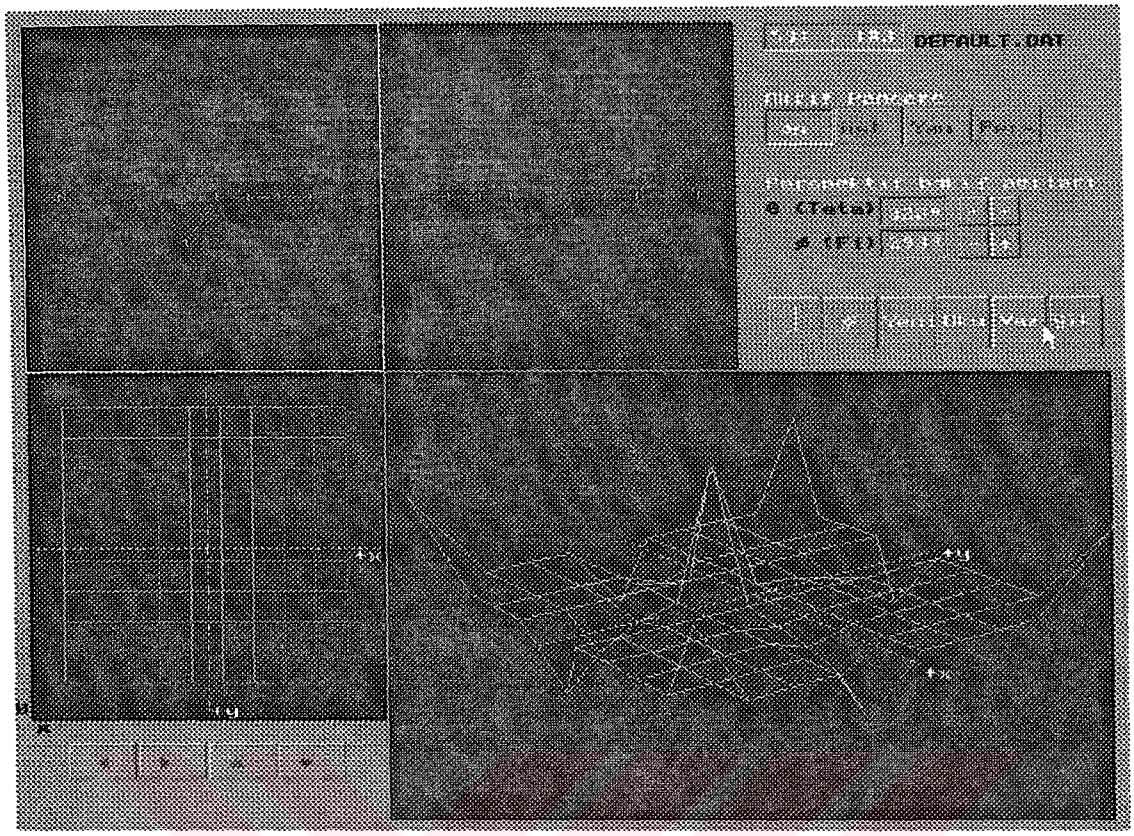
Sınav Jürisi

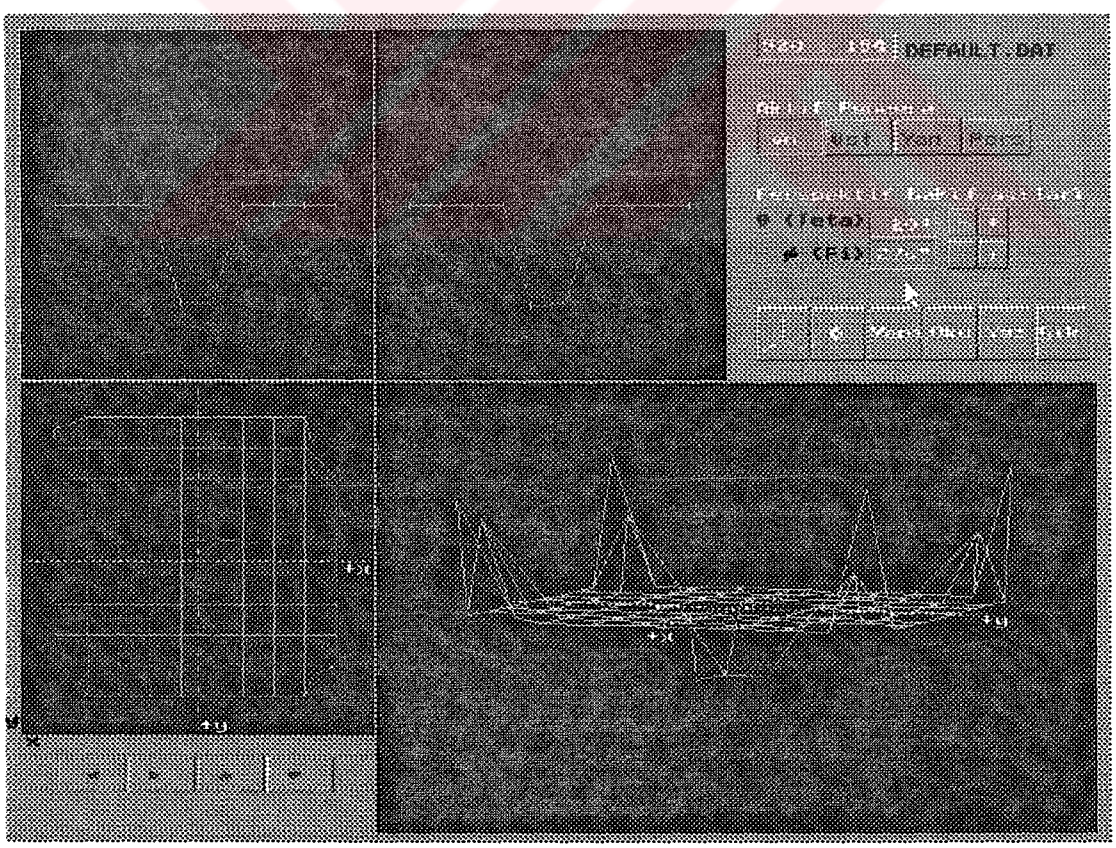
Başkan :

Üye :

Üye :

Bu tez Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Esaslarına uygundur.


DEFAULT.DAT
+x
+y
+z


DEFAULT.DAT
+x
+y
+z