CAD/CAM (BDT/BDÜ) TASARIMINDA KULLANILAN BAZI TEMEL İŞLEMLERİN PROGRAMLANMASI

Ahmet Naci ÇOKLAR

YÜKSEK LİSANS TEZİ

ELEKTRONİK VE BİLGİSAYAR SİSTEMLERİ
EĞİTİMİ ANABİLİM DALI

KONYA, 2004

T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CAD/CAM (BDT/BDÜ) TASARIMINDA KULLANILAN BAZI TEMEL İŞLEMLERİN PROGRAMLANMASI

Ahmet Naci ÇOKLAR

YÜKSEK LİSANS TEZİ

ELEKTRONİK VE BİLGİSAYAR SİSTEMLERİ EĞİTİMİ
ANABİLİM DALI

KONYA, 2004

T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CAD/CAM (BDT/BDÜ) TASARIMINDA KULLANILAN BAZI TEMEL İŞLEMLERİN PROGRAMLANMASI

Ahmet Naci ÇOKLAR
YÜKSEK LİSANS TEZİ

ELEKTRONİK VE BİLGİSAYAR SİSTEMLERİ EĞİTİMİ
ANABİLİM DALI

Bu tez .21./.01./2004 tarihinde aşağıdaki jüri tarafından

oybirliği/oyçokluğu ile kabul edilmiştir.

Danışman	**Üye**	**Üye**
Doç. Dr. Faruk ÜNSAÇAR	Prof. Dr. Novruz ALLAHVERDİ	Doç. Dr. Fatih BOTSALI

ÖZET

Yüksek Lisans Tezi

CAD/CAM (BDT/BDÜ) TASARIMINDA KULLANILAN BAZI TEMEL İŞLEMLERİN PROGRAMLANMASI

Ahmet Naci ÇOKLAR

Selçuk Üniversitesi Fen Bilimleri Enstitüsü

Elektronik Ve Bilgisayar Sistemleri Eğitimi Anabilim Dalı

Danışman : Doç. Dr. Faruk ÜNSAÇAR

2004, 117 sayfa

Jüri: Prof. Dr. Novruz ALLAHVERDİ

Doç. Dr. Fatih BOTSALI

Doç. Dr. Faruk ÜNSAÇAR

Bu çalışmada Bilgisayar Destekli Üretim(BDÜ)/Bilgisayar Destekli Tasarım (BDT) programlarındaki bazı temel işlemlerin programlanması ele alınmıştır. Taşıma, döndürme, eğri ve yüzey uydurma, yüzey normal vektörünün hesaplanması, kesici takım ofsetinin hesaplanması, kesici takım yolunun bulunması, G kodlarının üretilmesi, yüzey normal vektörünün tersyüz edilmesi, yüzey yoğunluğunun azaltılması, yüzeyin uzatılması işlemlerinin ne olduğu, matematiksel olarak ifadeleri ve akış diyagramları açıklanmıştır. Tüm bu işlemlerde, bilgisayar belleğinde tutulmakta olan ve yüzey verilerini içeren noktaların matematiksel olarak ifade edildiği dosyalar kullanılmaktadır. Bu yüzey veri dosyası yukarıda sıralanan işlemlerden hangisi isteniyorsa, ilgili işleme ait matematiksel ifade uygulanarak sonuç veri dosyası elde edilmektedir. Bu veri dosyaları ise sonuçta yüzeyi oluşturacak takım hareketlerini içeren G kod sisteminin elde edilmesini sağlamaktadır. Son olarak ise simülasyon programı ile takımın bu yüzeyde nasıl hareket ettiğinin görülebilmesi sağlanmıştır. Tüm bu işlemlerin programlanmasında Turbo C++ programlama dili kullanılmıştır.

Anahtar Kelimeler: Sayısal Denetim, Bilgisayar Destekli Tasarım, Bilgisayar Destekli Üretim, Yüzey Modelleme, Eğri ve Yüzey Uydurma, Simülasyon, G Kod Sistemi.

ABSTRACT

MSc Thesis

PROGRAMMING SOME BASIC OPERATIONS FOR DESIGN OF CAD/CAM

Ahmet Naci ÇOKLAR

Selçuk University Graduate School

Of Natural and Applied Sciences

Department of Electronik and Computer Systems Education.

Supervisor : Associate Prof. Dr. Faruk ÜNSAÇAR

2004, 117 pages

Jury: Prof. Dr. Novruz ALLAHVERDİ

Doç. Dr. Fatih BOTSALI

Doç. Dr. Faruk ÜNSAÇAR

In this study it is aimed to survey programming of some basic operations for Computer Aided Design (CAD) / Computer Aided Manufacturing (CAM) systems. Operations of rotation, translation, fitting curved line and surface, calculating of surface normal vector, calculating cutter offsets, calculating cutter location, producing G codes, inverting surface normal vector, reducing surface intensity, extending surface has been explained. Then, mathematical expressions and flow charts has been explained. Files which were saved in the computer memory for all operations has been used. The result files has been obtained by using mathematical expressions related with the above operations. This process obtains G code files of cutter movements. Finally with the simulation program it has been provided the observing of surface.

Key Words: Numerical Control, Computer Aided Design, Computer Aided Manufacturing, Suface Modelling, Fitting Curved Line and Surface, Simulation, G Code System.

TEŞEKKÜR

Çalışma esnasında değerli fikir ve önerilerini esirgemeyen ve bana her konuda yardımcı olan danışman hocam Doç. Dr. Faruk ÜNSAÇAR'a ve ayrıca matematiksel hesaplama işlemlerinde bana destek olan Doç. Dr. Cengiz ÇINAR'a teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

KISALTMALAR

TÜRKÇE KARŞILIĞI		İNGİLİZCE KARŞILIĞI	
Sayısal Denetim	SD	Numerical Control	NC
Bilgisayar Sayısal Denetim	BSD	Computer Numerical Control	CNC
Bilgisayar Destekli Tasarım	BDT	Computer Aided Design	CAD
Bilgisayar Destekli Üretim	BDÜ	Computer Aided Manufacturing	CAM
Yüzey Modelleme	YM	Surface Modelling	SM
Eğri Uydurma	EU	Fitting Curver Line	FCL
Yüzey Uydurma	YU	Fitting Surface	FS
Kesici Takım Ofseti	KTO	Cutter Ofset	CO
Yüzey Normal Vektörü	YNV	Surface Normal Vector	SNV
Doğrudan Sayısal Denetim	DSD	Direct Numerical Control	DNC

1. GİRİŞ

Bilgisayar teknolojilerindeki gelişmelere paralel olarak, bilgisayar sistemleri hayatımızın her alanında olduğu gibi üretim ve imalat sistemlerini de etkilemiş ve bu alanlarda çok daha etkin bir şekilde kullanılır hale gelmiştir. Bu doğrultuda önemi her geçen gün artmakta olan üretim sistemleri de sürekli gelişmekte ve çeşitlilik kazanmaktadır.

Bu gelişmeye yön veren ise üretilen malların çeşitliliğini ve niteliğini belirleyen tüketicilerdir. Tüketicilerin yeni ve değişik mamüller talep etmesi nedeni ile üretim hızı ve kolaylığı ön plana çıkmaktadır. Bu üretim hız ve esnekliği ise yeni üretim teknolojilerinin geliştirilmesi ile mümkün olmaktadır. Bu konuda bilgisayar teknolojileri özellikle tasarım alanında ön plana çıkmaktadır. Üretim esnasında bilgisayar kullanımı ise Bilgisayar Destekli Tasarım (BDT) konusu ile kendi içerisinde bir sektör haline gelmiştir. Günümüzde pek çok firma bu alana yatırım yapmakta ve ekonomik olarak çok yüksek kazançlar elde etmektedir.

Özellikle Bilgisayar Sayısal Denetimli (BSD) tezgahların seri üretim sağlaması nedeni ile üretim taleplerindeki önemli bir özellik olan hız ve üretim kalitesi sağlanmış olmaktadır. Bu tezgahların özelliği olan karmaşık G kod sistemi çoğu çizim programı aracılığı ile PC bilgisayar kullanılarak kolayca üretilebilmekte ve RS 232 portu aracılığı ile Bilgisayar Sayısal Denetimli tezgahın kontrol ünitesine aktarılabilmektedir. Çizim programlarının sağladığı bu avantaj üretim esnasında bilgisayar çizim programlarının tercih edilme nedenini artırmaktadır.

Ülkemizde kullanılan Bigisayar Destekli Tasarım (BDT) ve Bilgisayar Destekli Üretim (BDÜ) programları çok yüksek fiyatlı olması nedeni ile ülke ekonomisi için yüksek kayıplara neden olmakta ve yabancı dille yazılmış oldukları için çoğu işlevinin anlaşılamaması nedeniyle de tasarım ve üretimi sınırlamaktadır. Bu ise maliyet fiyatlarını artırırken, üretim hızını ve kalitesini düşürmektedir.

Çok yüksek fiyatlara mal olan bu tasarım ve imalat programları maalesef ülkemizde yeterli ilgiyi görememektedir. Genellikle BDT/BDÜ alanında çalışmalar bulunmaktadır. Ancak bu çalışmalar sadece G kod sistemine yönelik olup, grafik ortamı olarak sadece simülasyon ile sınırlı kalmaktadır. Çizim programlarının içerisinde taşıma, döndürme, ölçeklendirme, eğri ve yüzey uydurma, yüzey normal vektörünün hesaplanması, kesici takım ofsetinin hesaplanması, kesici takım yolunun

bulunması, G kodlarının üretilmesi, yüzey normal vektörünün tersyüz edilmesi, yüzey yoğunluğunun azaltılması, yüzeyin uzatılması işlemlerinin tamamının gerçekleştirilebilir olması gerekmektedir. Ancak günümüzde bu konuları kapsayan bir çalışma bulunmamaktadır. Bu konudaki eksikleri tamamlamak amacı ile böyle bir çalışma yapılmıştır.

Bu çalışma yüzey modelleme ile sınırlı bir çizim programını oluşturabilecek düzeyde kalmıştır. İşlemlerin ne olduğu, matematiksel olarak nasıl gerçekleştiği ve C programlama dilindeki kodları verilmiştir. Yapılan tüm işlemler matematiksel olarak gerçekleşmektedir. Bir çizim programı içerisinde bu işlemlerin nasıl gerçekleştiği anlatılmıştır. Bu prosedürler kullanılarak, bir çizim programı rahatlıkla gerçekleştirilebilir.

Çalışma içerisinde tek grafik ortam ise simülasyon programıdır. Bu simülasyon programı sayesinde bir yüzey için yukarıda adı geçen işlemlerden ilgili olana ait matematiksel ifadeleri içeren dosyalar G kod sistemine dönüştürüldükten sonra, bu yüzeyi elde etmek için gerekli olan takım hareketleri kontrol edilebilmektedir. Bağımsız olarak herhangi bir G kod sistemini satır satır inceleyebilmesi nedeni ile simülasyon programı eğitim amaçlı da kullanılabilmektedir.

Literatür olarak, taşıma ve döndürme gibi temel işlemlerin matematiksel ifadeleri için robotics (control, sensing, vision and intelligence), computer aided drafting and design ve teori ve problemlerle lineer cebir isimli kaynakların yanı sıra kaynaklar kısmında belirtilen internet adreslerinden faydalanılmıştır. Eğri ve yüzey uydurma, kesici takım yolunun bulunması, yüzeyin tersyüz edilmesi, yüzey yoğunluğunun azaltılması veya yüzeyin uzatılması işlemleri için ise numerically controlled machine tools isimli kitap ve bilgisayar destekli grafik modelleme ve yüzey modelleme esasları isimli yüksek lisans tez çalışmasından yararlanılmıştır.

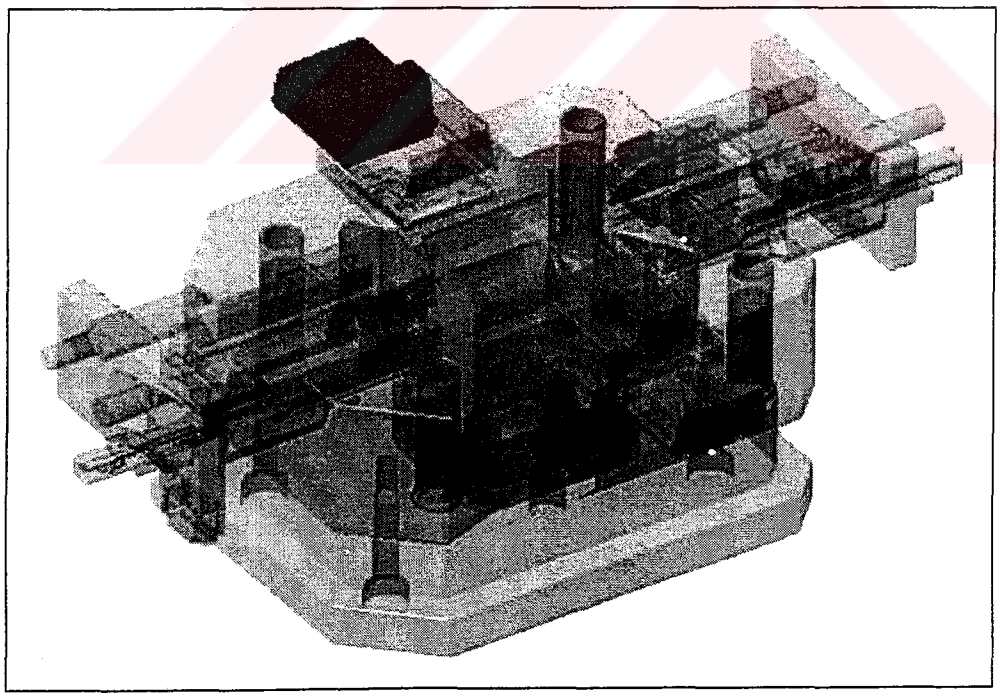
G kod sisteminin üretilmesi ile ilgili olarak çok sayıda kaynak bulunmaktadır. Bu konu içinde, freze tezgahları için CAD/CAM program geliştirme isimli yüksek lisans tezi ile CNC tezgahların programlanması ve numerically controlled machine tools isimli kitaplar referans alınmıştır. Simülasyon programının yazılması için computer aided drafting and design ve numerically controlled machine tools isimli kitaplar kaynak olarak kullanılmıştır.

2. GRAFİK MODELLEME

2.1. Grafik Modelleme Şekilleri

Üç boyutlu şekle Geometrik Model, bu şeklin oluşturulması tekniğine ise Geometrik Modelleme denir. Yaygın olarak kullanılan geometrik modelleme şekilleri iki gruba ayrılabilmektedir. Bunlar; Katı Modelleme (Solid Modeling) ve Yüzey Modelleme (Surface Modeling)'dir (Vickers 1990).

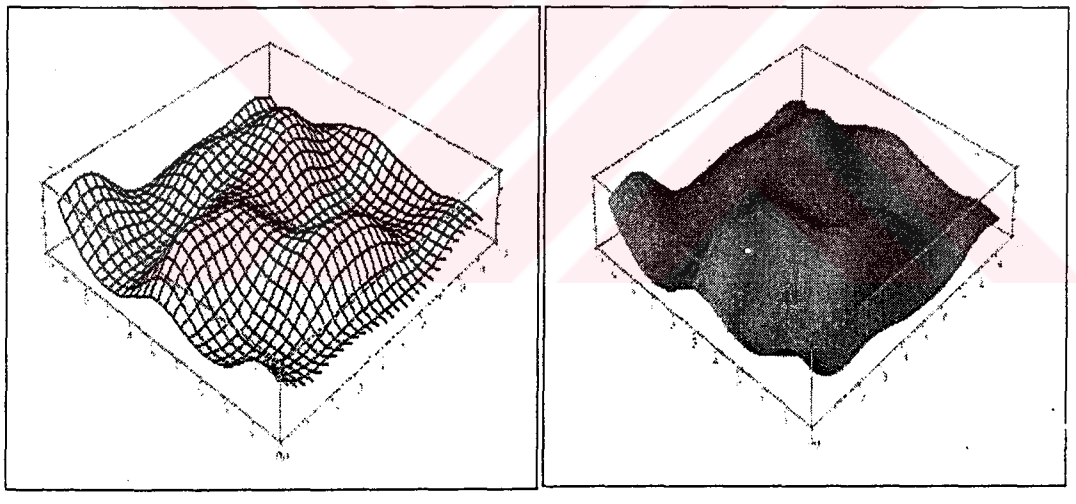
Katı modelleme prensip olarak dizaynı düşünülen katı olarak tanımlanan iki figürün kesiştirilmesi ile elde edilme tekniklerini içerir ve 3 boyutlu şekillerin tanımlanmasında yeterli olduğundan yaygın bir şekilde kullanılmaktadır. Katı modelleme tekniklerinde tasarımları oluşturmak için mühendislik çizim teknikleri yani geleneksel çizim yöntemleri yerine oluşturulacak şekil küp, silindir gibi hacim içeren şekiller yardımı ile oluşturulmaktadır. CAD çizim programlarının çoğu 3 boyutlu çizim işlemlerinde katı modelleme tekniğini kullanmaktadır. Aşağıdaki şekilde katı modelleme yöntemi kullanılarak oluşturulan 3 boyutlu bir tasarımın bir açıdan görünüşü görülmektedir (Şekil 2.1).



Şekil 2.1 Katı Modelleme Tekniği Kullanılarak Oluşturulmuş Bir Tasarım Örneği

CAD/CAM (BDT/BDÜ- Bilgisayar Destekli Tasarım/Bilgisayar Destekli Üretim) çizim programlarında yaygın olarak kullanılan bir diğer modelleme yöntemi olan yüzey modelleme ise, verilen noktalardan geçen eğri ve bu eğrilerden meydana gelen yüzey yaklaşım tekniklerini kapsar. Bu teknikte çizimi oluşturan temel noktalar belirtilmekte ve daha sonra bu noktalar kullanılarak şekli oluşturan eğriler, sonrasında da bu eğriler aracılığı ile nihai profili oluşturacak olan yüzeylerin hesaplanılması program tarafından gerçekleştirilmektedir. Daha çok kalıpçılık işlemleri veya haritacılık işlemlerinde kullanılmaktadır. Özellikle plastik ve dökme sanayiinde yapılacak imalat işlemleri için öncelikli olarak tasarımın içi boş bir şekilde kalıbının çıkarılması gerekmektedir. Bu teknik ise çok sayıda noktanın belirlenmesi ile parça şeklinin tanımlanmasını esas almaktadır (Mckissick 1987).

Aşağıdaki şekilde yüzey modelleme yöntemi kullanılarak oluşturulan 3 boyutlu bir tasarımın bir açıdan görünüşü görülmektedir (Şekil 2.2).



Şekil 2.2 Yüzey Modelleme Tekniği Kullanılarak Oluşturulmuş Bir Tasarım Örneği ve Bu Modelin Tel Kafes (Wire Frame) Gösterimi

3. YÜZEY MODELLEMEYLE KAVİSLİ (EĞİMLİ) YÜZEYLERİN MODELLENMESİ

3.1. Kavisli Yüzeylerin Modellenmesi ve Modelleme Yöntemleri

Geçmişten günümüze kadar kavisli yüzeylerin tanımlanması ve üretilmesi, daha çok katı modellemede kullanılan doğrusal yüzeylerin tanımlanması ve üretilmesinden çok daha fazla karmaşık ve problemli olmuştur. Ancak mühendislik imalatı içeren pek çok uygulama, imalata dair tasarım esnasında sıkça kavisli yüzeylerin kullanımını gerektirmektedir. Örnek olarak gemi, araba, uçak, telefon vb. gibi teknik donanımlar ve bunların parçalarının tasarlanması sırasında eğimli yüzeyler yoğun bir şekilde kullanılmaktadır. Benzer şekilde yapay kol ve bacak gibi imalatların tasarımında da eğimli yüzeylere dayalı tasarımlar ön plana çıkmaktadır. Bu tasarımların gerçekleştirilebilmesi için öncelikli olarak kalıplarının çıkartılması veya belirli bir oranda küçültülmüş modellerinin çizilmesi gerekmektedir.

Eğimli yüzeyler modelleme işlemi için tasarlanırken, birkaç yöntemle bu yüzeyler elde edilebilir. Oluşturulmuş modellerin yeniden düzenlenmesi, rasgele yüzey verilerinin kullanılması, şeklin lazerle taranması, matematiksel modellerle CAD programında çizilmesi gibi yöntemler temel teşkil etmektedir.

Çoğu metotta parçaların tüm yüzeyleri tanımlanmaz. Bunun yerine yüzey içerisindeki birkaç temel nokta tanımlanır. Bu birkaç nokta kullanılarak yüzeyler CAD programlarına hesaplatılır. Ancak, eğer veriler bazı ölçümler sonucunda elde edilmişse muhtemelen düzensiz ve hatalı veri girişleri de söz konusu olabilecektir. Veri girişinin düzgün olabilmesi için üretimde yüzeyi oluşturan noktalara ait düzenli bir dizinin matematiksel modelinin oluşturulması daha doğru olacaktır. Yani matematiksel modellere dayalı (noktalar esas alınarak) yapılacak bir tasarımda hata oranı daha az olacaktır (Vickers 1990).

Lazer tarama yönteminde, elde edilen veri kümesi oldukça tutarsız olabilmektedir. Bu yüzden eğimli yüzeylerin taratılması için çoğunlukla CAD programlarında yüzeye ait bazı noktalar verilmek suretiyle, yüzeyin oluşturulması işlemi programa hesaplattırılarak yapılmaktadır. Bunun için araba yüzeyi tasarımı gibi çizimlerin bazılarında yüzeylerde istenmeyen pürüzler görünmesi, çizimde temel noktaların verilip, eğimli yüzeyin bilgisayara oluşturulduğunun en önemli kanıtıdır.

3.2. Matematiksel Modellere Dayalı Yüzey Modelleme İşlemleri

Yüzey modellemede kullanılacak temel noktaların verilip, yüzeyin matematiksel modellerle hesaplatılması en çok kullanılan yöntemdir. Tüm tasarım işlemleri için tanımlama ve üretim esnasında uygulanacak temel işlemler aşağıdaki şekilde sıralanabilir:

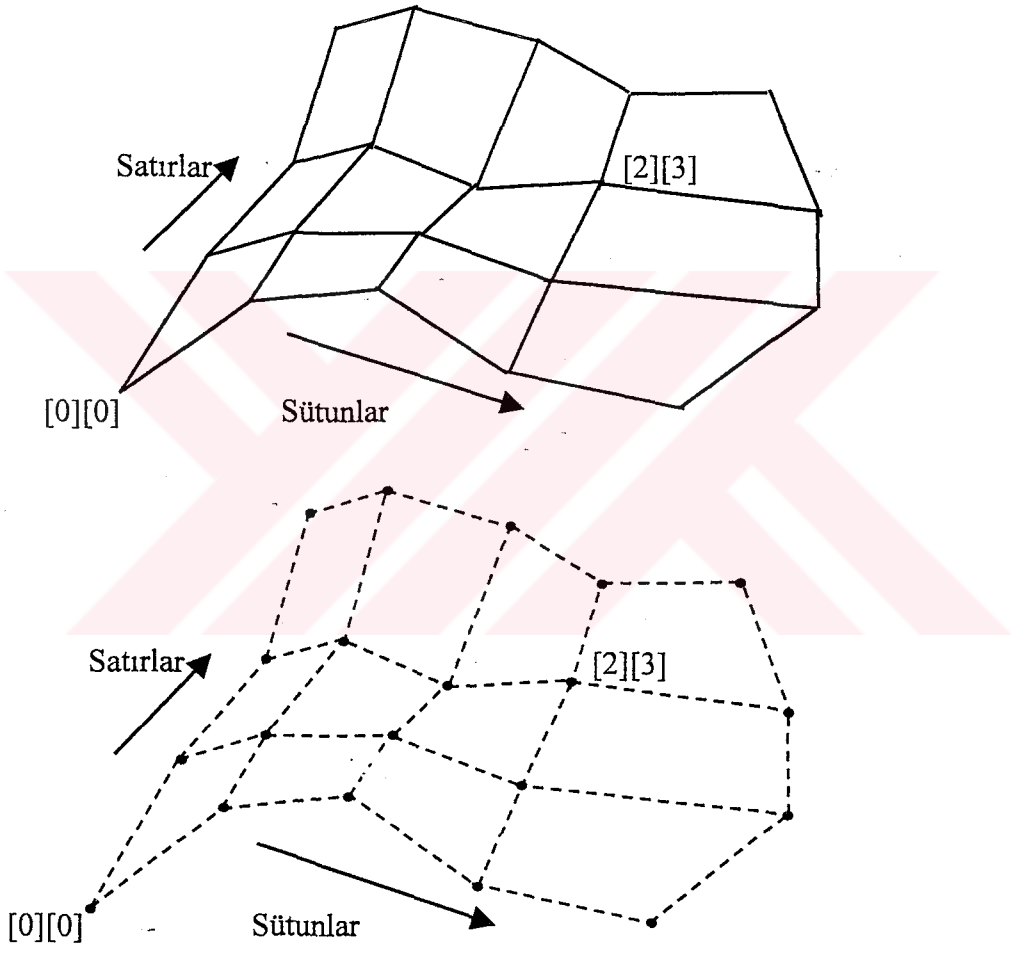
- Verinin Girilmesi: Tanımlanacak yüzeyin tamamına veya bir kısmına ait temel noktaların girilmesi.
- Yüzeyin Modellenmesi: Tanımlanacak yüzeye ait daha önce girilen verilerin üretim amaçlarına göre yüzeyinin oluşturulması.
- Yüzey Uydurma: Parça konumu ve takım yolu ile alakalı yüzeyin belirli bir düzene sokulması.
- Yüzey Normallerinin Hesaplanması: Tüm parça şekline ait yüzey normallerinin tanımlanması.
- Kesici Yolunun Hesaplanması: Parçaya ait tüm yüzeylerde kesici takımın alacağı noktaların hesaplanması.
- G kodlarının Üretilmesi: Tanımlanan parça şeklinin CNC tezgahına aktarılabilmesi için gerekli olan G kodlarının ürettirilmesi.

Eğimli yüzeylerin üretilmesi ile ilgili önemli bir gerçek te, yüzey tanımlama tipinin ve CNC tezgah türünün önemli olmaması gerektiğidir. Eğimli yüzeylerin işlenmesinde finish işlemi ile eğimli yüzeylerin pürüzsüz olması da istenilen bir özelliktir. Nokta sayısı ile düzenlendiği için eğimli yüzey tanımlanırken ne kadar çok nokta belirtilirse pürüzlülük te o kadar azalacaktır. Ancak buna bağlı olarak ta işlem hızı da o kadar yavaşlayacaktır. Yani işlem zamanı ile pürüzlülüğü gidermek için yapılması gereken son işlem olan, finish işlemi arasında ters orantılı bir ilişki vardır. Bir işlem için normalde uygun finish işlem payı ise 0.1 ile 2.0 mm arasında bir miktardır (Vickers 1990).

Bu tez kapsamında günlük hayatta kullanılan çizim programlarının arka zemininde bu işlemlerin nasıl gerçekleştirildiği ele alınacaktır. Rasgele oluşturulacak eğimli yüzeylerle alakalı olarak kesici takım yollarının tanımlanması incelenerek, hesaplamalar türetilecektir. C programlama dili ile bu işlemlerle ilgili görünüşler bir simülasyon programı ile izlenebilecektir.

3.3. Matematiksel Modellere Dayalı Yüzey Modelleme İşlemlerinde Verinin Girilmesi

Bir yüzeyi tanımlamak için çok çeşitli veri kaynaklarından yararlanılabilir. Veri giriş işlemindeki başlıca amaç, üretim işlemi için uygun bir şekilde verinin tutulabileceği bir dosyanın bulunması ve bu dosya içerisindeki bilgilerin matematiksel işlemlerde kullanılabilecek şekilde anlamlı bir sırada tutulmasıdır. Örnek olarak aşağıdaki şekilde görüldüğü gibi bir yüzey modelleme işleminde yüzey satır ve sütunlar yardımı ile tanımlanmıştır (Şekil 3.1).



Şekil 3.1 Satır ve Sütunları Kullanarak Bir Yüzeyin Tanımlanması

Bu tanımlama işleminin gerçekleştirilebilmesi için, çizgilerin tamamının kullanılmasına gerek yoktur. Bu işlem için her bir satır ve sütunun kesişim noktalarının bir matris oluşturacak şekilde, bir dosya içerisinde tanımlanması yeterlidir. Bu dosya içerisinde veri noktalarının dikdörtgensel matris şeklinde yüzey

tanımlanmış olacaktır. Bir yüzeye ait bir noktayı oluşturan üç adet koordinat bileşeni ortaya çıkacaktır (X, Y ve Z koordinatları). Örnek olarak yukarıdaki şekilde de görülebileceği gibi yüzeyi oluşturan temel bir nokta olan 2. satır üzerindeki 3. sütundaki bir koordinat X[2][3], Y[2][3] ve Z[2][3] şeklinde ifade edilebilir.

Yukarıdaki gibi bir şeklin bilgisayar ortamında tutulması ise bu noktaların matematiksel olarak ifadesinin aşağıdaki gibi bir dosyada tutulması şeklinde olmaktadır.

4	Dosyadaki toplam satır sayısı
4	Dosyadaki toplam sütun sayısı
0.000000 0.000000 6.945513	X, Y ve Z koordinatları
11.112825 0.000000 3.195067	
22.225651 0.000000 3.438101	
30.004660 0.000000 9.168039	
4	Satır içerisindeki sütun sayısı
0.000000 0.000000 6.945513	X, Y ve Z koordinatları
11.112825 7.396981 5.695435	
22.225651 6.250916 6.806497	
30.004660 0.000000 9.168039	
4	Satır içerisindeki sütun sayısı
0.000000 0.000000 11.078021	X, Y ve Z koordinatları
11.112825 7.396981 10.904495	
22.225651 6.667675 12.085170	
30.004660 0.000000 13.543784	
4	Satır içerisindeki sütun sayısı
0.000000 0.000000 11.668512	X, Y ve Z koordinatları
11.112825 7.396981 11.598998	
22.225651 6.756888 13.213834	
30.004660 0.000000 15.002296	

Tablo 1.1 Veri Dosyası Formatı

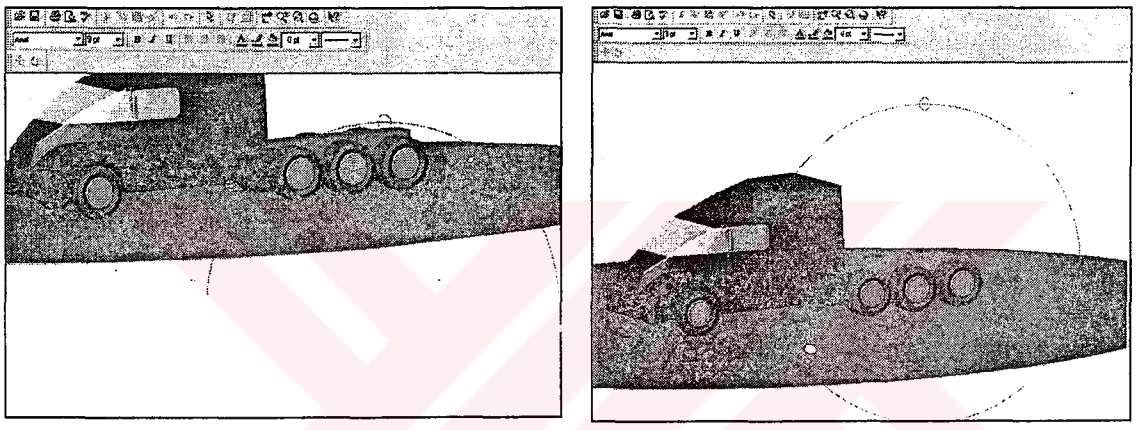
Tanımlanacak bir yüzey için veri dosya düzeni yukarıdaki tabloda verilmiştir. Bu tablodaki ilk satır, yüzey üzerindeki toplam satır sayısını göstermektedir. Benzer şekilde ikinci satırda her bir satır içerisindeki sütun sayısını göstermektedir. X, Y ve Z koordinat parçaları ise dosya içerisinde her bir satır üzerindeki her bir noktanın yerini temsil etmektedir. Örnek olarak [2][3] noktasının koordinatlarının neresi olduğu tespit edilebilir. Çizim programları da bu koordinatları kullanarak şekilleri ekrana yansıtmaktadır. Ancak bu veriler ham veri olup, öncelikle işlenilmesi gerekmektedir.

4. YÜZEYİN TAŞINMASI (ÖTELENMESİ)

4.1. Taşıma İşlemi

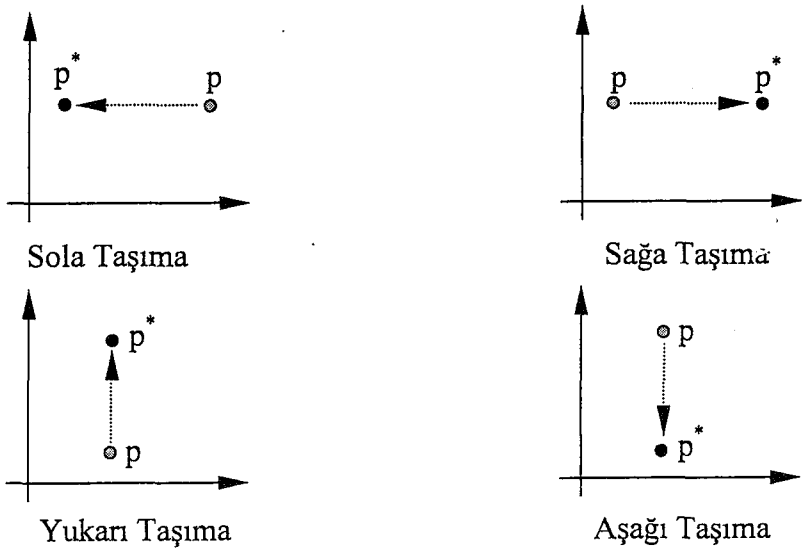
Bir şeklin bilgisayar koordinat sistemindeki yeri, farklı bir noktaya taşınabilmektedir. Bu işlem için bilgisayar belleği içerisinde bulunan veri dosyasındaki koordinatların konumu (vertex) belirtilen yer hesaplanılarak değiştirilmektedir. Bu hesaplama işlemi belirtilecek diğer bir nokta esas alınarak yapılabilmektedir (Demirci 1994).

Aşağıda verilen şekilde taşıma işlemi ile çizim alanında yeri değiştirilmiş (sağa ve aşağıya doğru taşınmış) bir şekil örneği görülmektedir (Şekil 4.1).



Şekil 4.1 Bir Şeklin Çizim Programı İçerisinde Taşınmasına Örnek

Taşıma işlemi sol, sağ, yukarı ve aşağı olmak üzere dört temel yönde olabilir.



Şekil 4.2 Bir Noktanın Sol, Sağ , Yukarı ve Aşağı Yönlerde Taşınması

4.2. Matematiksel İfade

Yukarıdaki şekillerin tamamında taşıma işlemi p noktasından p* noktasına yapılmıştır. Böyle bir taşıma işlemi için hesaplanılacak matris en temel ifade ile aşağıdaki gibi gösterilebilir (Fu 1987).

p noktası iki boyutlu uzayda p=[x,y] şeklinde ifade edilmek üzere;

p noktasına göre elde edilebilecek p^* noktası için transform matrisi

$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ şeklinde gösterilmektedir. Buna göre p^* noktası;

$p^* = p * T$ yani ➔ $p^* = [x, y] * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = [ax+cy, bx+dy]$ formülü ile hesaplanabilir. Bu transform matrisindeki a ve d değerleri taşıma işlemi için kullanılmasına karşın, b ve c değerleri kırpma işleminde kullanılmaktadır.

Sağa-Sola Taşıma Faktörü ← a b → Y Ekseni Üzerinde Kırpma Faktörü

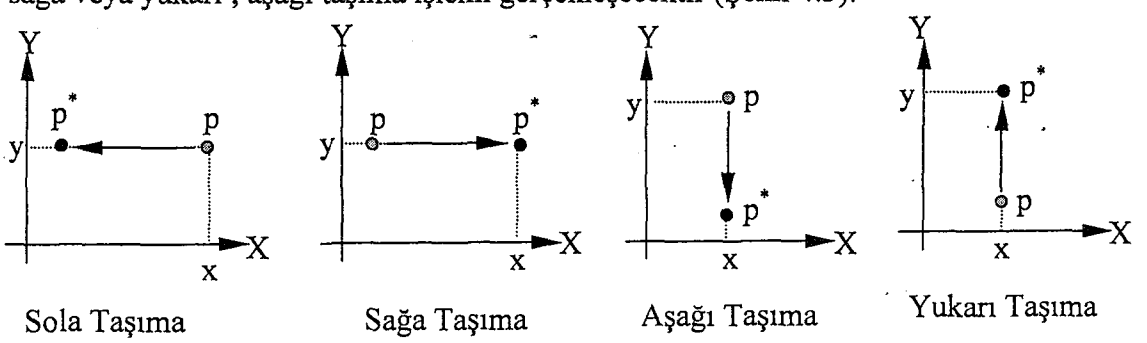
X Ekseni Üzerinde Kırpma Faktörü ← c d → Yukarı-Aşağı Taşıma Faktörü

Bu matris hesabında eğer sadece bir tane taşıma faktörü kullanılırsa dört temel işlem gerçekleştirilmiş olur. Bu temel işlemler için bu matris şu şekilde sadeleştirilebilir.

$$P^* = p \,.\, T = [x, y] \,.\, \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} = [ax + 0y,\ 0x + 1y] = [ax, y]$$

$$P^* = p \,.\, T = [x, y] \,.\, \begin{bmatrix} 1 & 0 \\ 0 & d \end{bmatrix} = [x + 0y,\ 0x + dy] = [x, dy]$$

Formüllerde de görüldüğü gibi verilecek a ve d faktörüne göre sadece sola, sağa veya yukarı , aşağı taşıma işlemi gerçekleşecektir (Şekil 4.3).



Şekil 4.3 Bir Noktanın Sola, Sağa, Aşağı ve Yukarı Taşınmasının Matematiksel İfadesi

Örnek olarak p=[100,100] noktasındaki bir piksele aşağıdaki gibi taşıma matrisleri uygulanırsa, p^* matrisinin yeri şu şekilde hesaplanabilir.

Soldaki bir noktaya taşımak için;

$$P^* = [100,100] \begin{bmatrix} 0.8 & 0 \\ 0 & 1 \end{bmatrix} = [100*0.8+0*100,\ 0*100+1*100] = [80,100]$$

Sağdaki bir noktaya taşımak için;

$$P^* = [100,100] \begin{bmatrix} 1.3 & 0 \\ 0 & 1 \end{bmatrix} = [100*1.3+0*100,\ 0*100+1*100] = [130,100]$$

Yukarı bir noktaya taşımak için;

$$P^* = [100,100] \begin{bmatrix} 1 & 0 \\ 0 & 1.2 \end{bmatrix} = [100*1+0*100,\ 0*100+1.2*100] = [100,120]$$

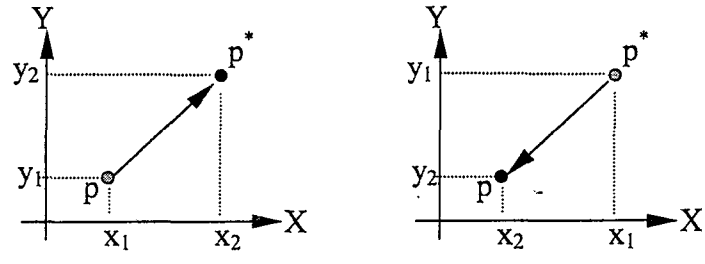
Aşağı bir noktaya taşımak için;

$$P^* = [100,100] \begin{bmatrix} 1 & 0 \\ 0 & 0.75 \end{bmatrix} = [100*1+0*100,\ 0*100+0.75*100] = [100,75]$$

Yukarıdaki transform matrisinde eğer a ve d değerlerinin her ikisine birden değer verilecek olursa sağ-sol ve yukarı aşağı her iki yönlü taşıma işlemi gerçekleştirilmiş olur.

$$P^* = p \,.\, T = [x_1, y_1] \,.\, \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = [ax_1 + 0y_1,\ 0x_1 + dy_1]\ [x_2,y_2] = [ax_1, dy_1]$$

Formülde de görüldüğü gibi verilecek a ve d faktörlerine göre yukarı-aşağı veya sol-sağ tüm yönlere (çapraz) taşıma işlemi gerçekleşecektir (Şekil 4.4).



Yukarı -Sağa Taşıma Aşağı-Sola Taşıma

Şekil 4.4 Bir Noktanın Çapraz Taşınmasının Matematiksel İfadesi

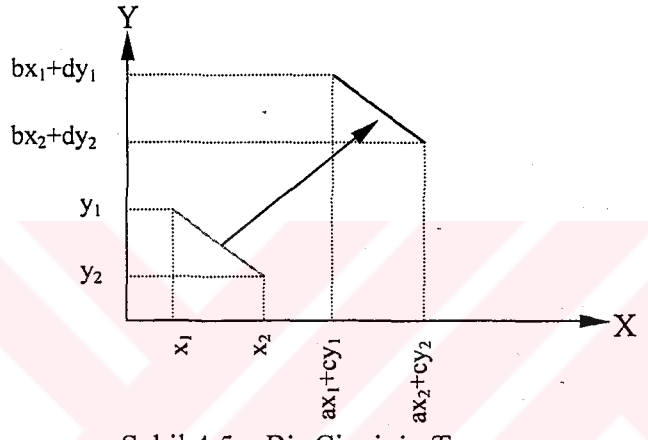
Düz bir çizginin taşınması için o çizginin $p_1=[x_1,y_1]$, başlangıç ve $p_2=[x_2,y_2]$ bitiş değerine göre taşıma matrisine göre p_1^* ve p_2^* noktalarının hesaplanması gerekmektedir.

$p_1=[x_1,y_1]$ ve $p_2=[x_2,y_2]$ ve Transform matrisi $T=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ olmak üz,ere

$$p_1^* = p_1.T = [x_1,y_1]\begin{bmatrix} a & b \\ c & d \end{bmatrix} = [ax_1+cy_1 \; , \; bx_1+dy_1]$$

$$p_2^* = p_2.T = [x_2,y_2]\begin{bmatrix} a & b \\ c & d \end{bmatrix} = [ax_2+cy_2 \; , \; bx_2+dy_2]$$ olarak hesaplanılır.

Böylece bir programlama dilinde çizilecek düz bir çizgi LINE(x_1,y_1,x_2,y_2); komutu yerine LINE(ax_1+cy_1 , bx_1+dy_1, ax_2+cy_2 , bx_2+dy_2); komutu kullanılarak istenilen yerine taşınabilir. Bu komutla çizgi aşağıdaki şekildeki gibi taşınmış olmaktadır.



Şekil 4.5 Bir Çizginin Taşınması

Eğimli bir çizginin taşınması (öteleme) için ise; $[x_1,y_1]$ ve $[x_2,y_2]$ noktaları verilen bir çizginin eğimi $m=\left[\frac{y2-y1}{x2-x1}\right]$'dir. Buna göre taşınacak olan şeklin eğimide şu şekilde hesaplanabilir.

$$m^* = \frac{y_2^* - y_1^*}{x_2^* - x_1^*} = \frac{(bx_2 - dy_2) - (bx_1 - dy_1)}{(ax_2 - cy_2) - (ax_1 - cy_1)}$$

$$= \frac{b(x_2 - x_1) + d(y_2 - y_1)}{a(x_2 - x_1) + c(y_2 - y_1)} = \frac{b + d\dfrac{(y_2 - y_1)}{(x_2 - x_1)}}{a + c\dfrac{(y_2 - y_1)}{(x_2 - x_1)}} = \frac{b + dm}{a + cm}$$

Bir şeklin taşınması için temel mantığı anlatmak amacı ile 2x2 matrisi anlatılmıştır ve bu genellikle bir noktanın iki boyutlu uzayda bulunduğu konuma göre taşınmasını sağlamaktadır. Ancak daha önce belirtildiği gibi bir şeklin

bilgisayar ortamında üç boyutlu görüntülenebilmesi için 3x3 boyutlu bir matrisin kullanılması gerekmektedir.

Bir p=[x,y] noktası 3x3 matrisi ile işlem yapılamamaktadır. Bu işlemin yapılabilmesi için [x', y', h] olarak temsil edilebilir. Bu ilave h değerinin nedeni 3x3 boyutlu b ir taşıma matrisi $p=[x,y]_{1*2}$ 1x2 boyutlu bir matris ile çarpılamamaktadır. Bu değerin çarpılabilmesi için matrisin sütun sayısının 3 olması gerekir. Yani 1x3 boyutunda olmalıdır. Formüldeki $[x', y', h]_{1*3}$ h değeri matrisin 3x3 boyutlu bir taşıma matrisi ile çarpılabilmesini sağlamıştır. Burada x ve y değerinin, 3x3 boyuta çıkarım işleminden etkilenmemesi için h değeri genellikle 1'e ayarlanır. Böylece çarpma işleminde etkisiz eleman olan 1 değeri sonucu etkilemeyecektir. Bu [x', y', h] matrisine ise homojen koordinat denilmektedir. Taşıma işlemi için gerekli olan taşıma matrisi ise;

$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l & m & 1 \end{bmatrix}$ olarak temsil edilmektedir. Burada l değeri X ekseni yönünde x noktasından olan mesafe oranı, m ise Y ekseni yönünde y noktasından olan mesafe oranını temsil etmektedir. Buna göre p^* noktası;

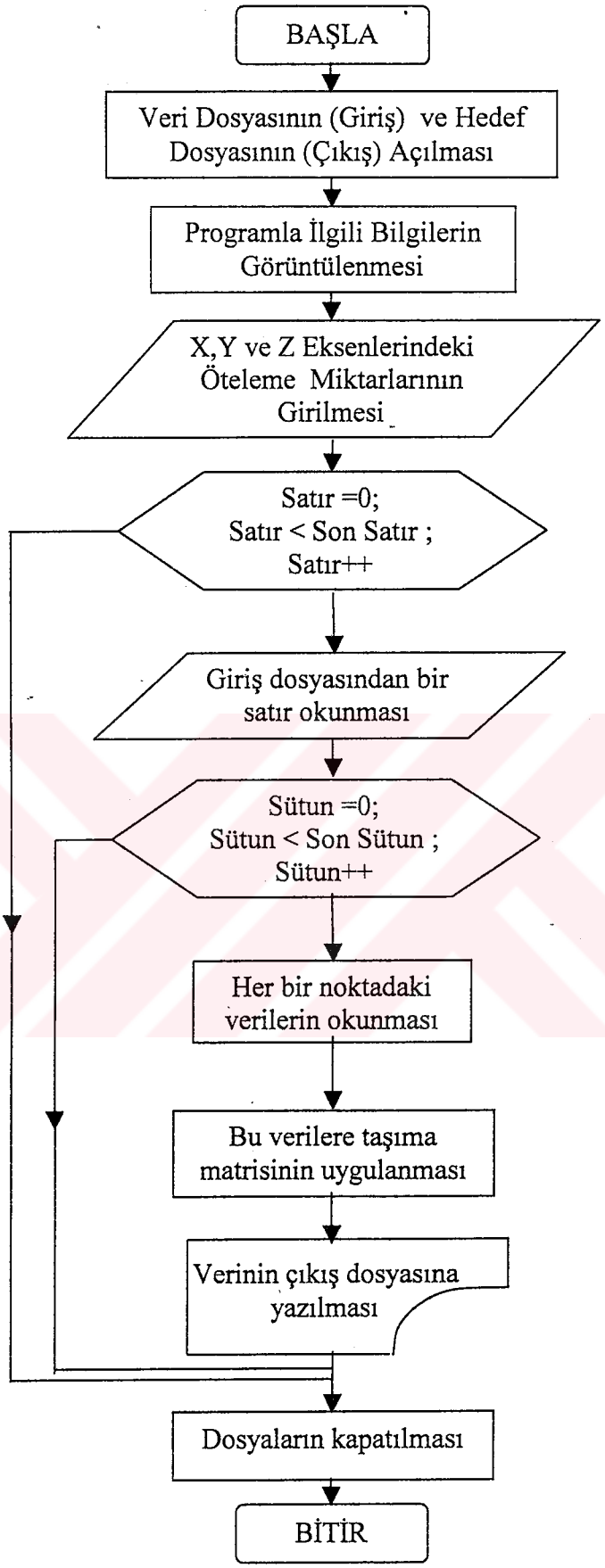
$$P^* = p \,.\, T = [x, y, 1] \,.\, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l & m & 1 \end{bmatrix} = [x+l,\ y+m,\ 1]$$

olarak bulunmuş olur ve taşıma işlemi için bir şekli oluşturan veri dosyası içerisindeki tüm noktalar bir döngü içerisinde yukarıdaki matrise uygulanarak şeklin başka bir noktaya taşınması sağlanmış olur (Fu 1987).

4.3. Akış Diyagramı

Program daha önce bahsedilen verinin tutulmasını sağlayan dosyadaki her bir veri noktasının (vertex) yukarıdaki bahsedilen formüle uygulanması sonucu elde edilecek verilerin yeni bir dosyaya yazılması ile bu işlem yerine getirilmektedir. Yani dosya içerisinde X, Y ve Z koordinat değerlerini gösteren veriler yeniden hesaplanarak ilgili yerlere yeni değerler yazılmaktadır.

EK-A'da programa ait veri kodları (TASIMA.C) verilmiştir.



Şekil 4.6 Yüzey Taşıma İşleminin Akış Diyagramı

5. YÜZEYİN BİR EKSENDE DÖNDÜRÜLMESİ (ROTATION)

5.1. Döndürme İşlemi

Üç boyutlu çizimden iki boyutlu çizimi ayıran en önemli özelliktir ve normalde iki boyutlu bir görünüşün döndürülmesi ancak bir fotoğrafın döndürülmesi gibidir (Lipschutz 1978).

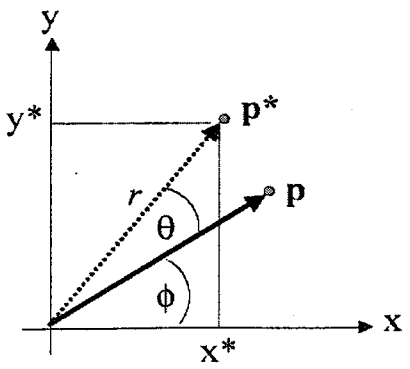
Dünya koordinat sisteminde bir parça insan eline alınıp, istenildiği şekilde döndürülerek incelenebilmektedir. İnsan gözünün bulunduğu noktaya göre oluşacak her farklı bakış açısı, o şeklin farklı algılanmasını sağlayacaktır. Bu bakış açısının yer değiştirilmesi döndürme işlemi olarak tanımlanmaktadır. Bu bakış açısı değiştirilmeden, bakılan nokta mesafesinin yakınlaştırılıp uzaklaştırılması ise ölçeklendirme yani scale işlemidir. Bu işlemlerin her ikisi de birlikte uygulanabilmektedir (Mckissick 1987).

Bilgisayar koordinat sisteminde de çizilen bir parça şekli, taşınabildiği gibi döndürülebilmeli veya ölçeklendirilebilmelidir. Bunun içinde taşıma işlemini sağlayan matrise benzer şekilde bir döndürme matrisi bulunmaktadır. Bu matris de 3x3 boyutlu olması halinde üç boyutlu bir görünüm elde edilecektir.

Aşağıdaki şekilde de görülebileceği gibi bir şeklin döndürülebilmesi belirli bir açıya göre yapılmaktadır. Bu ise veri dosyasında tutulan bir veri noktası değerinin döndürme matris değerine göre hesaplanılarak, bunun ekrana yansıtılmasını içermektedir.

5.2. Matematiksel İfade

P=[x, y] döndürülerek elde edilecek p^* noktası [x^*, y^*] olmak üzere(Şekil 5.1);



Şekil 5.1 Döndürme İşleminin Matematiksel İfadesi

$$x = r\cos\Phi, \quad y = \sin\Phi$$

$$x^* = r\cos(\Phi+\theta)$$

$$= r\cos\Phi\cos\theta - r\sin\Phi\sin\theta$$

$$= x\cos\theta - y\sin\theta$$

$$y^* = r\sin(\Phi+\theta)$$

$$= r\sin\Phi\cos\theta + r\cos\Phi\sin\theta$$

$$= y\cos\theta + x\sin\theta$$

Buradan p^* noktasını elde etmek için döndürme matrisi;

$T_{Rt} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ şeklinde olur.

Bu p^* noktası;

$$p^* = [x^*, y^*] = p \cdot T_{Rt} = [x, y]. \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

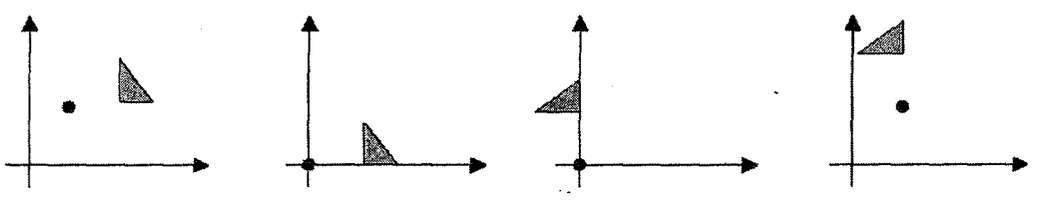
$$= [x\cos\theta - y\sin\theta, y\cos\theta + x\sin\theta]$$

Rasgele seçilecek bir noktaya göre döndürme işlemi için matris olarak $T_{Rt} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ dönüşüm matrisi kullanılmaktadır.

Rasgele seçilecek bir nokta üzerindeki bir dönüşüm için;

$$T = [T_{Tas}].[T_{Rt}].[T_{Tas}]^{-1}$$

Burada $[T_{Tas}]$ taşıma matrisini, $[T_{Rt}]$ ise döndürme matrisini temsil etmektedir.



Şekil 5.2 Bir Noktaya Göre Döndürme İşlemi

Burada daha açık bir şekilde ifade etmek gerekirse (m, n)'yi döndürme merkez noktası koordinatları olarak kabul edersek;

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

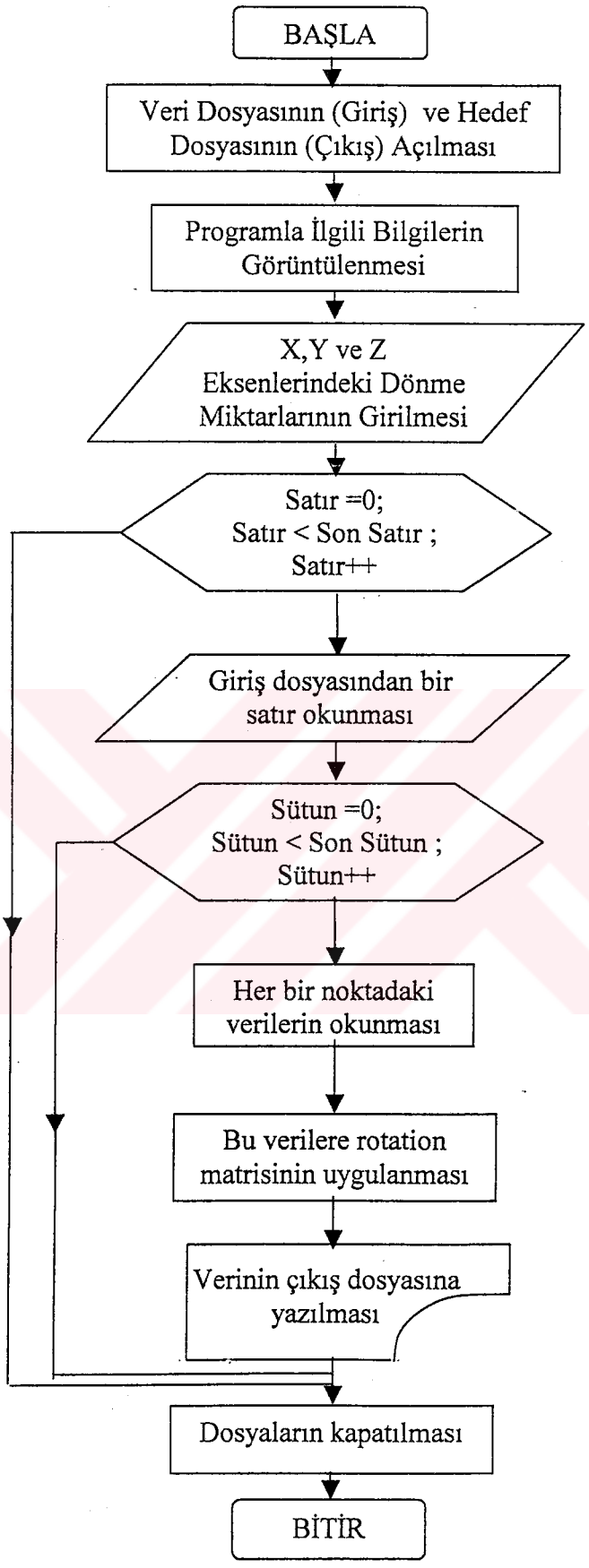
$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -m(\cos\theta - 1) + n\sin\theta & -n(\cos\theta - 1) - m\sin\theta & 1 \end{bmatrix}$$

şeklinde döndürme matrisi elde edilmiş olur (Fu 1987).

5.3. Akış Diyagramı

Program taşıma işlemine benzer şekilde verinin tutulmasını sağlayan dosyadaki her bir veri noktasının (vertex) yukarıdaki rotation yani döndürme formülüne uygulanması sonucu elde edilecek verilerin yeni bir dosyaya yazılması ile bu işlemi yerine getirmektedir. Yani dosya içerisinde X, Y ve Z koordinat değerlerini gösteren veriler yeniden hesaplanarak ilgili yerlere yeni değerler yazılmaktadır.

EK-B'de programa ait veri kodları (DONDUR.C) verilmiştir.



Şekil 5.3 Döndürme İşleminin Akış Diyagramı

6. EĞRİ VE YÜZEY UYDURMA

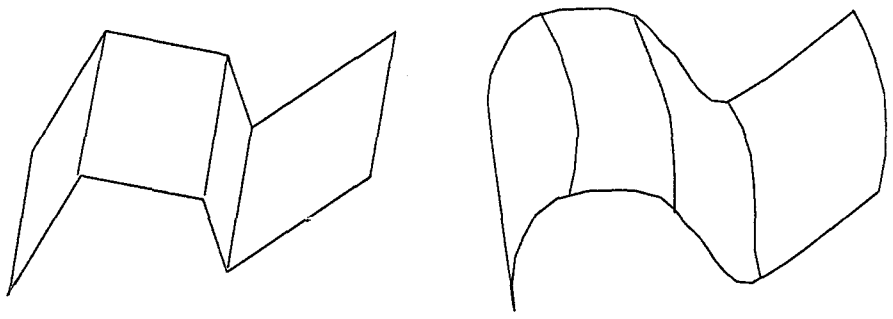
6.1. Eğri ve Yüzey Uydurma, B-Spline Çizgi ve Yüzeyler

Özellikle bir çizimin görünümünde ahengi sağlamak amacı ile kullanılan eğri ve yüzey uydurma işlemleri aynı zamanda şekle ait farklı parçaların entegre edilmesinde de ön plana çıkmaktadır. Bunun yanı sıra parça şeklindeki pürüzlülüğün ortadan kaldırılmasını da sağlamaktadır (Yıldız 1994).

Çizim programlarında bir parçaya ait şekil belirli noktalar kullanılarak öncelikle köşeli bir yüzey elde edilmekte, daha sonra bu noktalar eğri uydurma ve yüzey uydurma işlemi uygulanarak, yüzey oluşturma işleminde bu köşeler ortadan kaldırılmaktadır. Bu eğri ve yüzey oluşturma işlemlerinde en yaygın olarak kullanılan yöntem dörtlü B-Spline yaklaşımıdır. Bu işlem ile parçanın pürüzsüz ve uyumlu olarak bütünlüğünün elde edilmesi sağlanmaktadır. Ayrıca uygulanan B-Spline interpolasyon işleminin en önemli kullanım amaçlarından bir diğeri ise, bir parçanın sadece bir kısmının (lokal) çizimlerinin uydurulmasında değil, karmaşık yüzeylerin entegrasyonunda uyumu sağlayarak yüzey pürüzlülüğünü ortadan kaldırmasıdır. Günlük hayatta çoğu çizim programlarının eğri ve yüzey uydurma tekniğini kullandığı rahatça anlaşılabilir. Örnek olarak 3D (Üç Boyutlu) oyunlarda araba veya benzer şekillerin köşeli (pürüzlü) olduğu rahatlıkla fark edilebilir. Aynı şekilde çoğu çizim programında çizimlerin aşırı derecede büyütülmesi halinde düz çizgilerin dahi köşeli çizimlerden meydana geldiği görülebilir. Bu tür çizimlerin tamamı eğri veya yüzey uydurma işlemini kullanmaktadır (Vickers 1990). Aşağıda aynı çizginin normal ve B-Spline uydurulmuş hali görülmektedir (Şekil 6.1).

Şekil 6.1 Bir Çizgi ve Bu Çizginin Eğri Uydurma Yöntemiyle Elde Edilmiş B-Spline Hali

Yüzey uydurma işlemi de yukarıdaki örneğe benzetilebilir. Ancak temel fark şeklin bir satır değil, birden fazla satırın hem satır hemde sütunlarının uydurulması sonucu bir alanın yani yüzeyin elde edilmesidir. Aşağıdaki şekilde ise yüzey uydurma işlemine örnek görülmektedir (Şekil 6.2).



Şekil 6.2 Bir Çizgiden Elde Edilmiş Normal ve Uydurulmuş İki Yüzey Modeli

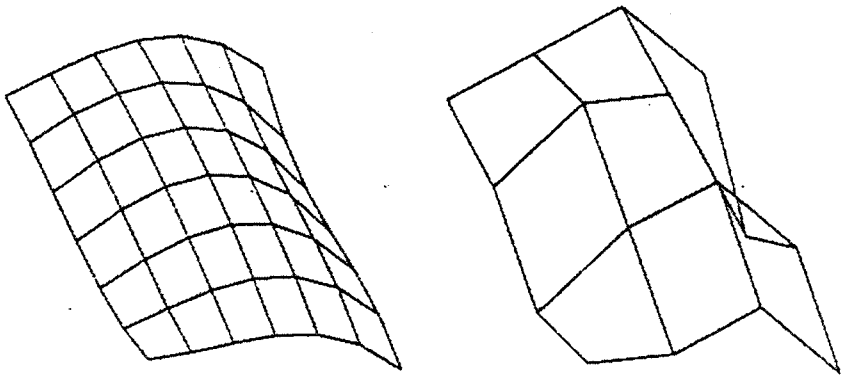
Yukarıdaki şekillerden de anlaşılabileceği gibi uydurulmuş şekillerde yüzey pürüzlülüğü ortadan kalkmaktadır. Bu işlem ise yine veri noktaları kullanılarak elde edilmektedir. Veri dosyasındaki veri noktalarına B-Spline matematiksel işlemleri uygulanarak elde edilen veri noktalarının yeni bir dosyaya yazılması ile nihai sonuç elde edilmektedir. Bu işlemler eğri uydurmayı ve yüzey uydurmayı sağlayan iki farklı program kullanılarak gerçekleştirilmektedir.

6.2. Çizim Programları İçerisinde Uydurulmuş Yüzeyler

Yüzeyler, ticari CAD çizim paketleri içerisinden de tanımlanabilmektedir. Bu işlem için daha önce verilen veri dosyası kullanılmaktadır. Bu veri dosyası kullanılarak tanımlama işleminde aşağıdaki yöntemler kullanılabilmektedir. Örnek olarak AutoCAD çizim programında yüzeylerin tanımlanması beş farklı yöntemle yapılmaktadır (Mckissick 1987).

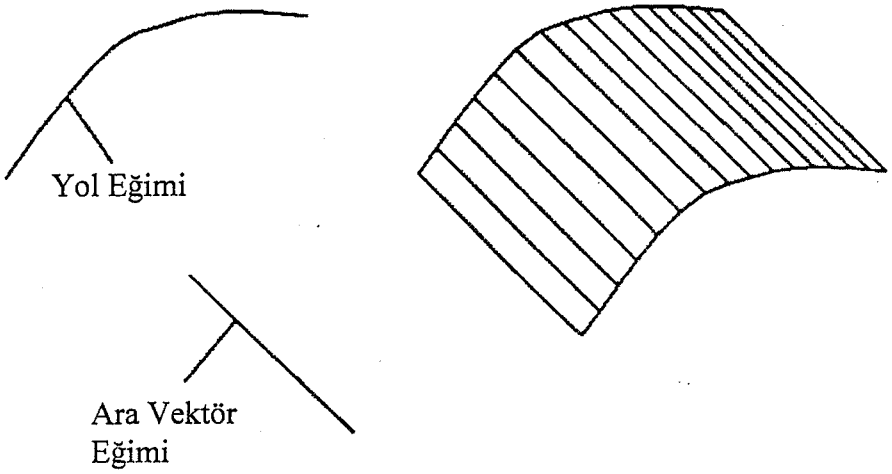
- Genel Çokgen Ağları (General Polygon Mesh)
- Orta Noktalar Bulunarak Çizilmiş Yüzeyler (Ruled Surface)
- Doğrudan Çizilmiş Yüzeyler (Tabulated Surface)
- Devir Yaptırılarak Elde Edilen Yüzeyler (Surface of Revolution)
- Kenarları Tanımlanarak Elde Edilen Yüzeyler (Edge-Defined Surface Path)
- Parametrik Karışımları Kullanan Autolisp ile Elde Edilen Yüzeyler (Parametric Blending Using Autolisp)

Tüm bu çizimler düz çizgi, yay, 3 boyutlu çokgen gibi geometrik elementler kullanılarak tanımlanabilir. Aşağıdaki şekillerin tamamında AutoCAD programında elde edilen yüzeylere ait örnekler verilmiştir (Şekil 6.3).

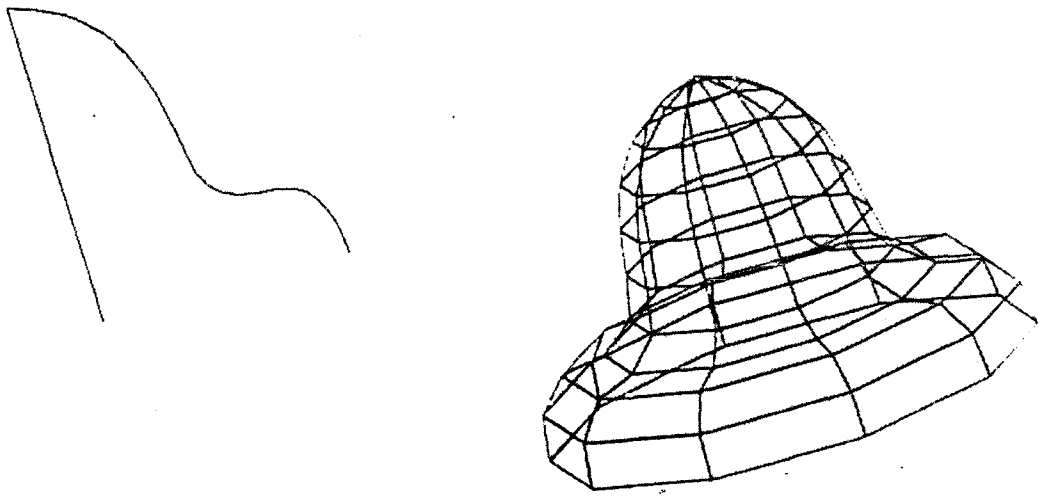


a) Genel Çokgen Ağları

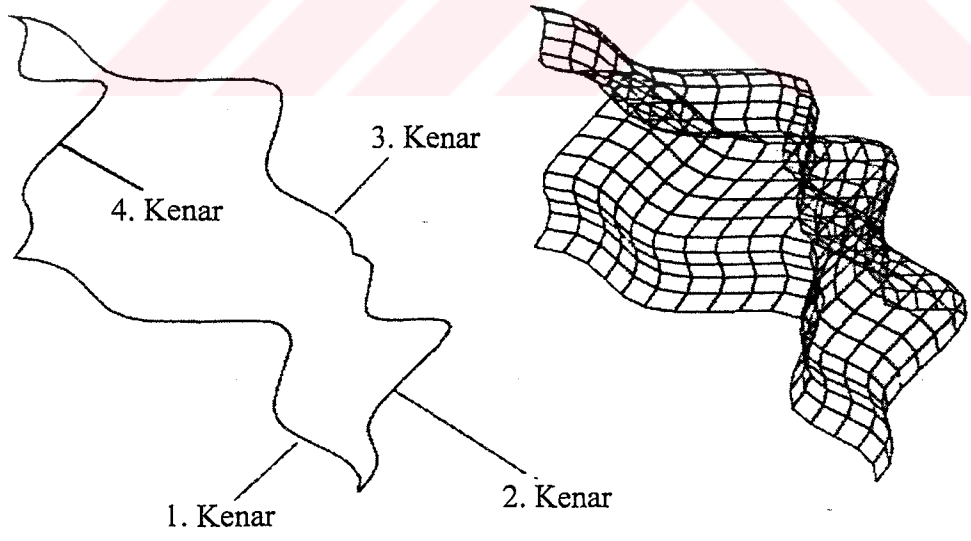
b) Orta Noktalar Bulunarak Çizilmiş Yüzeyler



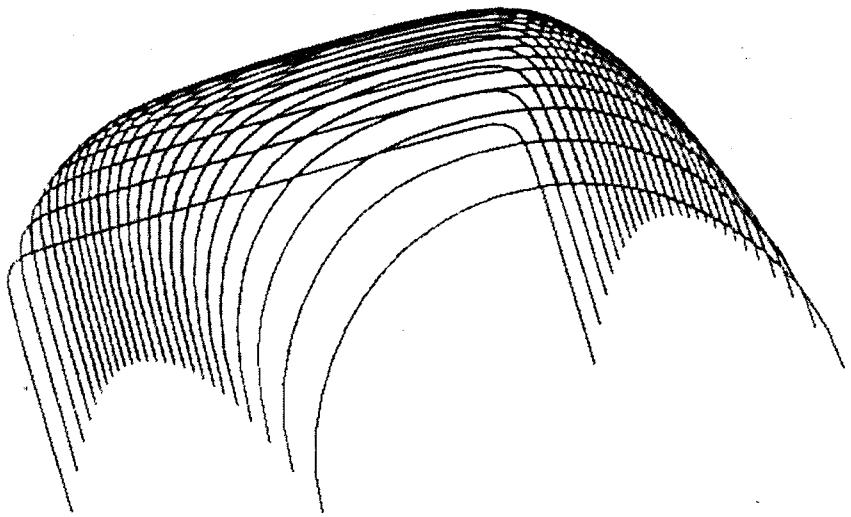
c) Doğrudan Çizilmiş Yüzeyler



ç) Devir Yaptırılarak Elde Edilen Yüzeyler



d) Kenarları Tanımlanarak Elde Edilen Yüzeyler

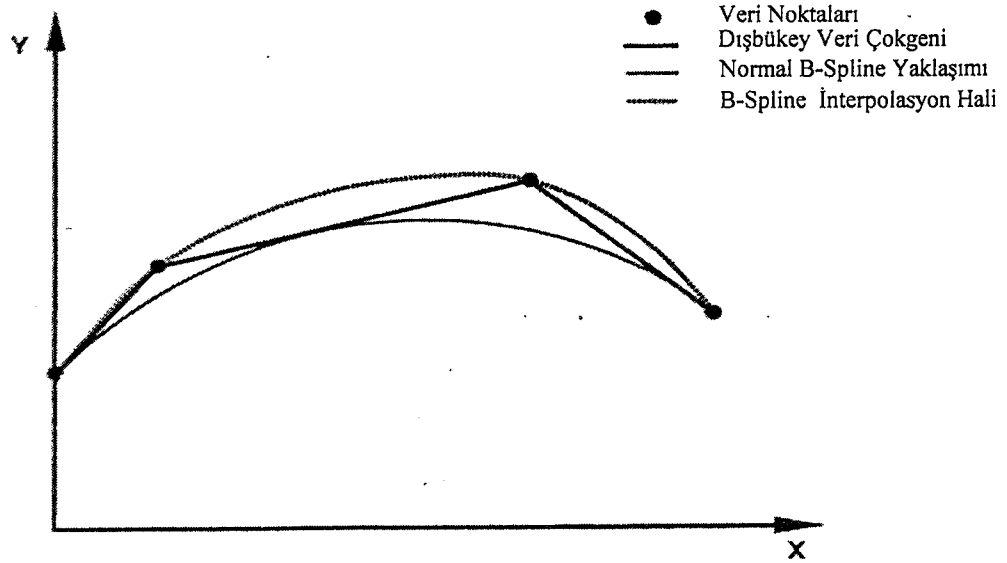


e) Parametrik karışımları kullanan Autolisp ile Elde Edilen Yüzeyler

Şekil 6.3 CAD Çizim Programında Elde Edilen Yüzeylere Ait Örnekler

6.3. Eğri ve Yüzey Uydurma İşlemleri

Bilgisayar Destekli Üretim (BDÜ / CAM) uygulamaları içerisinde, eğri ve yüzey modelleyebilmek için kullanılan yöntemlerden bir tanesi dörtlü B-Spline yaklaşımıdır. Bu B-Spline çizgiler bir yüzeyin tanımlanmasında kullanılmaktadır (Şekil 6.4). Şekildeki gibi belirli noktalar kullanılarak öncelikle köşeli bir yüzey elde edilmekte, daha sonra bu noktalar eğri uydurma işlemi yani B-Spline yaklaşımı uygulanarak yüzey oluşturma işleminde bu köşeler ortadan kaldırılmaktadır.



Şekil 6.4 Çokgenin Çevresini Kuşatmak Suretiyle Uygulanan B-Spline Yöntemi

6.4. Matematiksel İfade

Eğri ve yüzey uydurma programları içerisinde aşağıda görülen parametrik B-Spline yaklaşımlı eğriler için şu şekilde bir tanımlama yapılabilir (Vickers 1990).

$$R(t) = \sum_{Nokta=0}^{Ntoplamnoktaadedi} P_{nokta} . B_{nokta,4}(t) \qquad (1)$$

R(t) parametrik B-Spline yaklaşımında, P kontrol çokgenindeki köşe sayısını (yukarıdaki şekildeki veri noktalarının her birine ait bilgiyi) ve B(t) ise temel fonksiyon değerini göstermektedir. Parametrik B-Spline yaklaşımlı yüzeyler ise aşağıdaki şekilde tanımlanmaktadır;

$$R(u,v) = \sum_{Sat=0}^{NToplamSatırSayısı} \sum_{Nokta=0}^{Ntoplamnoktaadedi} Q_{sat,nokta} . B(u)_{nokta,4} B(v)_{sat,4} \qquad (2)$$

Yüzey uydurma işleminde kullanılan R(u,v) parametrik B-Spline yaklaşımında, Q değeri kontrol çokgenini ve B(u) ile B(v) ise temel fonksiyona ait normal değerlerini göstermektedir. Temel fonksiyon değerleri ise aşağıdaki gibi hesaplanmaktadır;

$$B_{nokta,dizi}(t) = \left\{ \frac{t - t_{nokta}}{t_{nokta+dizi-1} - t_{nokta}} \right\} B_{nokta,dizi-1}(t) + \left\{ \frac{t_{nokta+dizi} - t}{t_{nokta+dizi} - t_{nokta+1}} \right\} B_{nokta+1,dizi-1}(t) \qquad (3)$$

Bir sonraki aralık için başlangıç değerleri ise aşağıdaki eşitliğe göre hesaplanılmaktadır;

$$\begin{aligned} B_{nokta,1}(t) &= 1 \qquad \text{Eger } t_{nokta-1} \le t < t_{nokta+1} \\ B_{nokta,1}(t) &= 0 \qquad \text{Diger Durumlar} \end{aligned} \qquad (4)$$

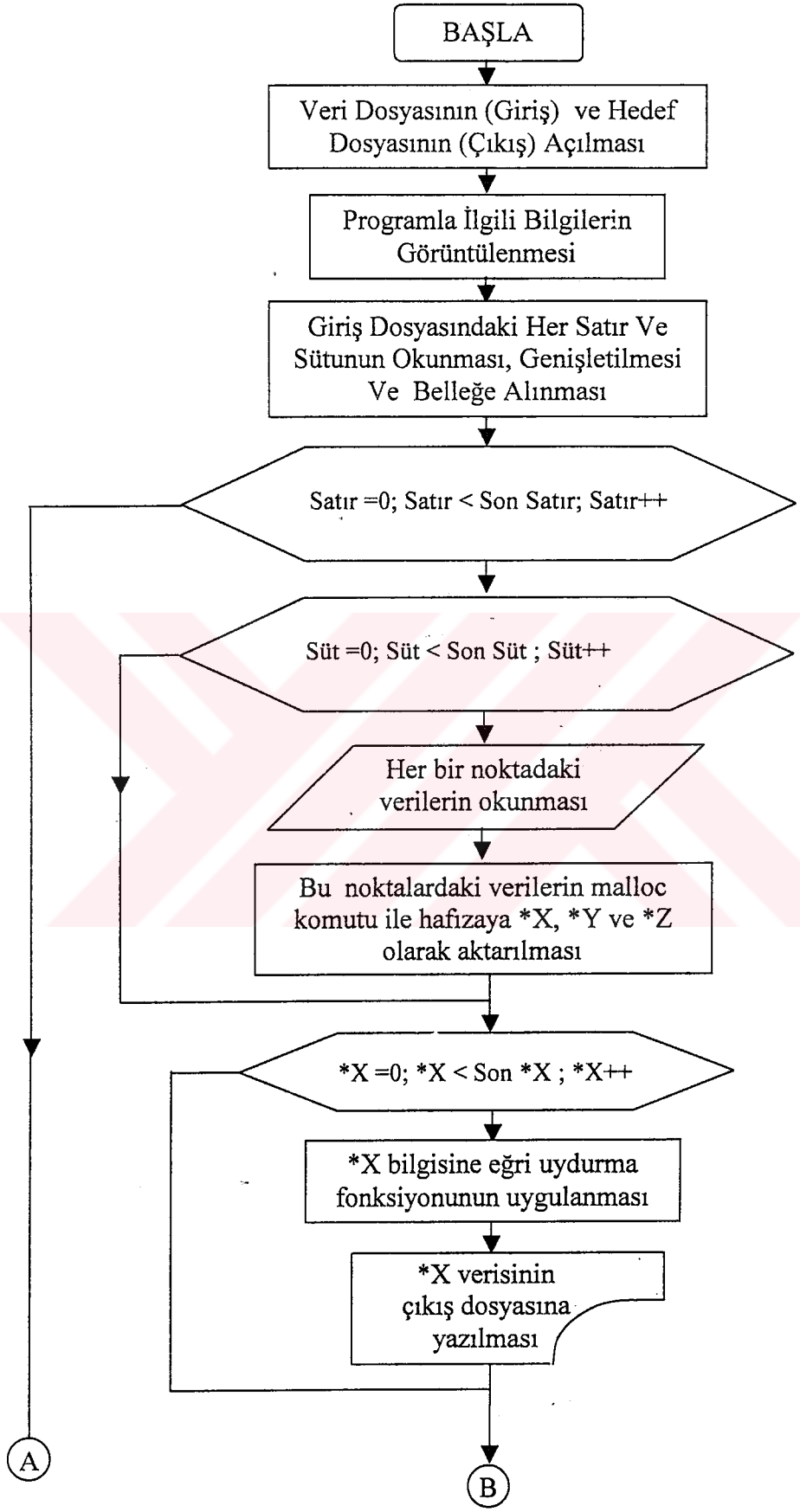
6.5. Akış Diyagramı

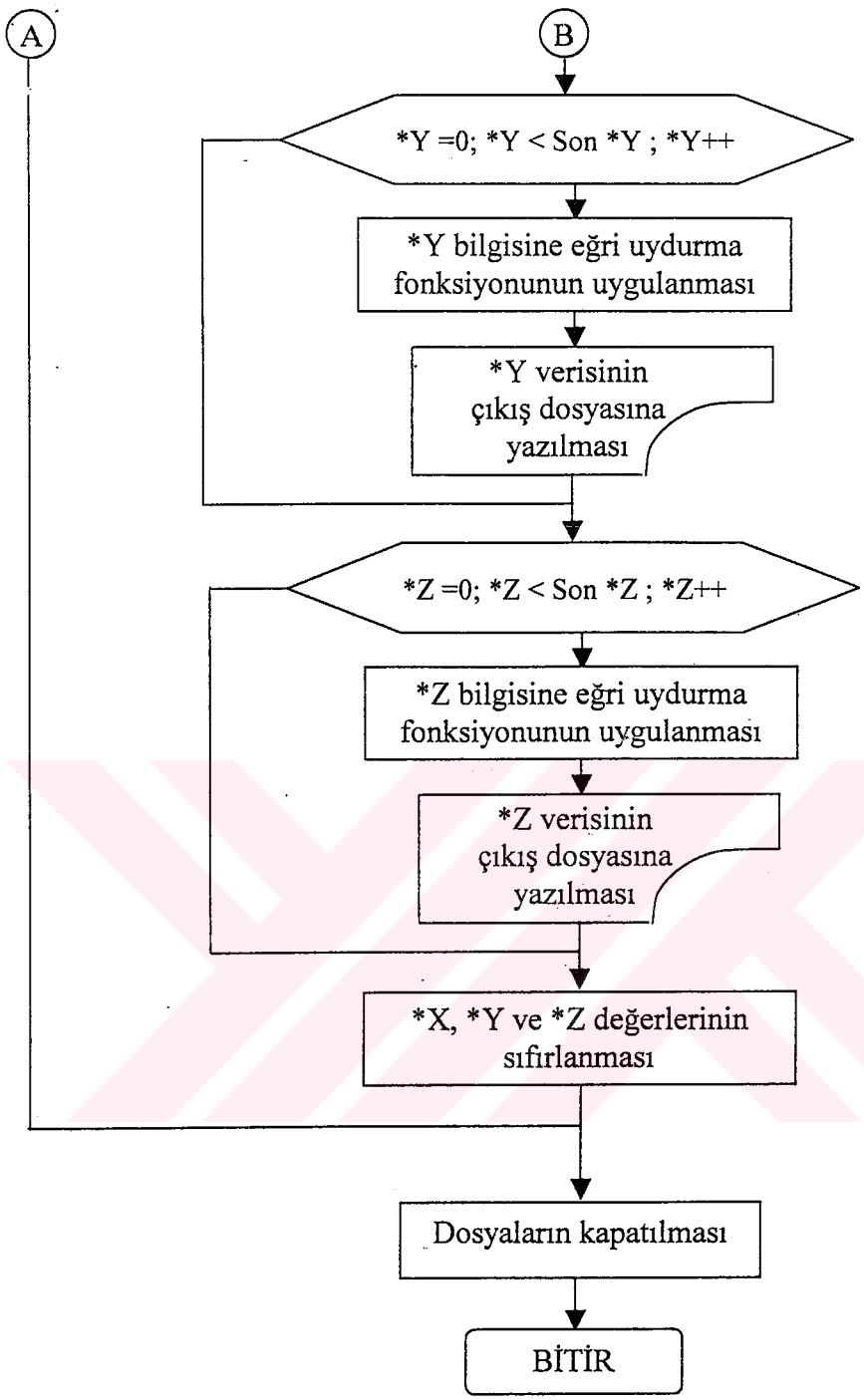
Eğri ve yüzey uydurma işlemleri iki farklı program tarafından ayrı ayrı gerçekleştirilmektedir. Bu programlarda verinin tutulmasını sağlayan dosyadaki her bir veri noktasının (vertex) yukarıdaki eğri uydurma işlemi için 1, yüzey uydurma için 2 numaralı formülün uygulanması sonucu elde edilen verilerin yeni bir dosyaya yazılması sonucu elde edilmektedir. Yani dosya içerisinde .X, Y ve Z koordinat değerlerini gösteren veriler yeniden hesaplanarak ilgili yerlere yeni değerler yazılmaktadır.

EK-C'de eğri uydurma programına (EGUYDUR.C), EK-Ç'de ise yüzey uydurma programına ait veri kodları (YUZUYDUR.C) verilmiştir. Eğri uydurma programı sadece satırları uydurmaktadır. Yani sütunlar arasındaki uyum olan interpolasyon işlemi dikkate alınmamaktadır. Eğer hem satır hemde sütunlar arasında interpolasyon işlemi isteniliyorsa bu durumda yüzey uydurma programının kullanılması gerekmektedir.

Aşağıda sırasıyla eğriuydurma ve yüzey uydurma programına ait akış diyagramları görülmektedir.

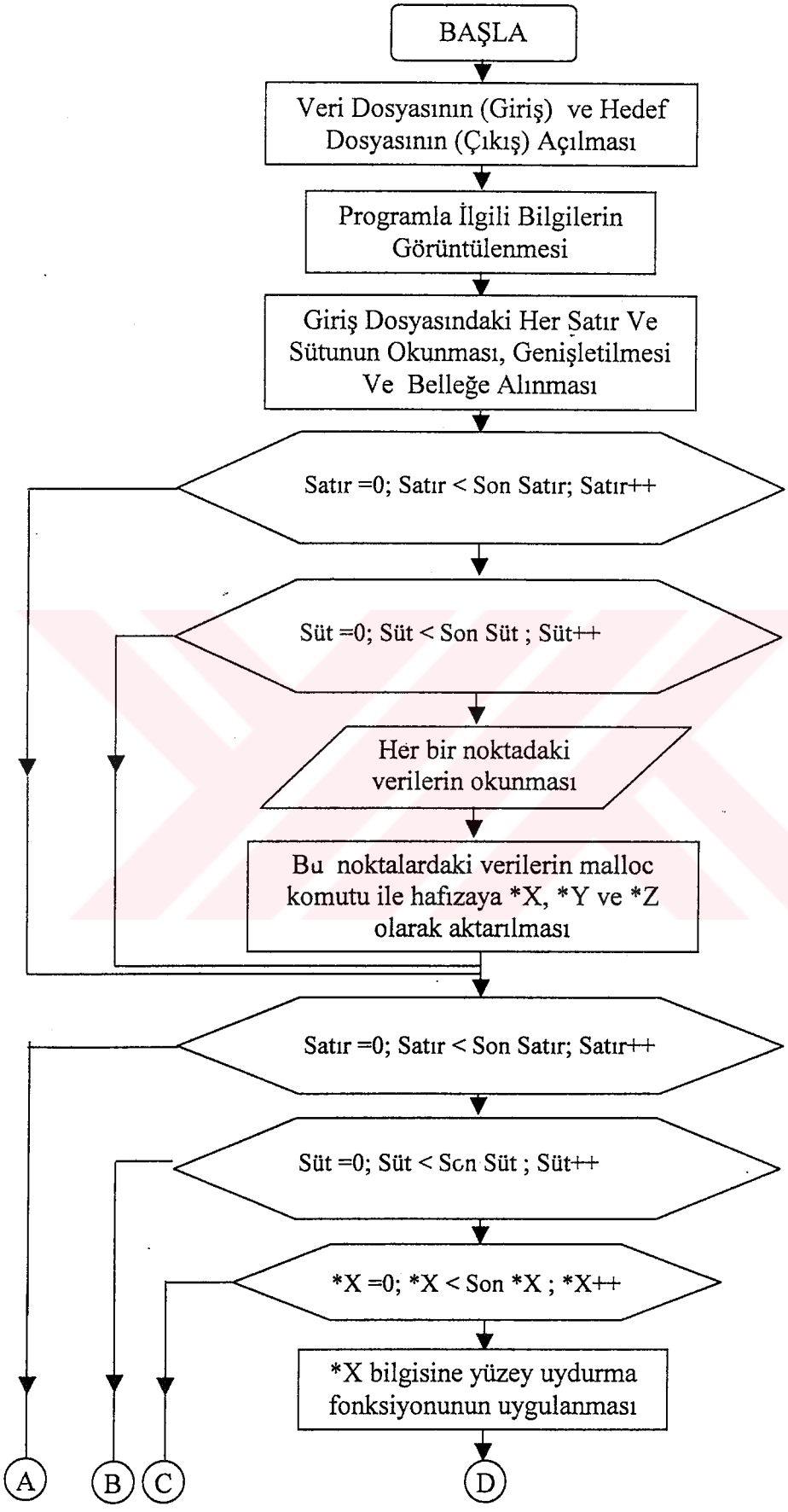
EK-C EGUYDUR.C isimli eğri uydurma programı akış diyagramı:

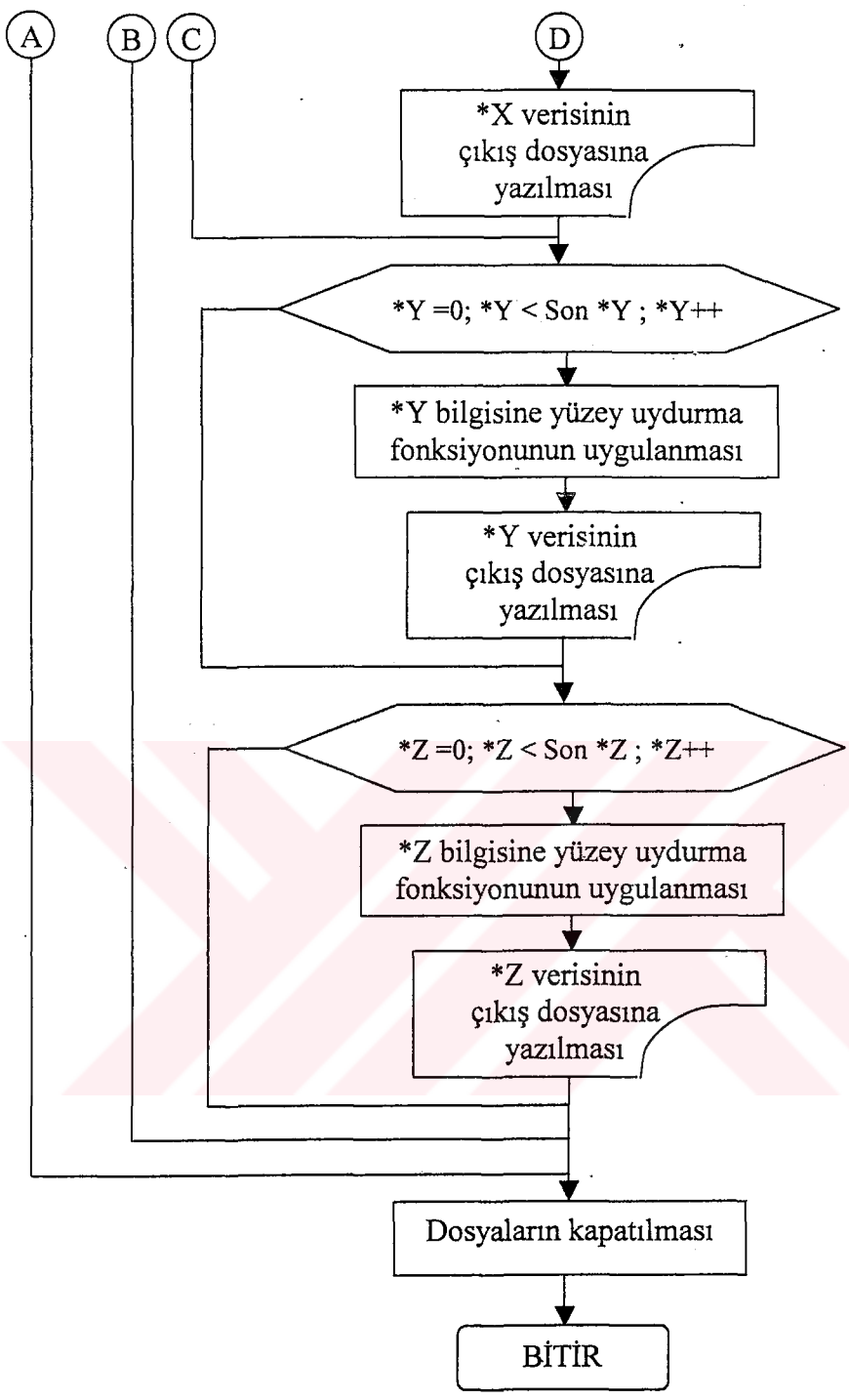




Şekil 6.5 Eğri Uydurma İşleminin Akış Diyagramı

EK-Ç YUZUYDUR.C isimli yüzey uydurma programı akış diyagramı:





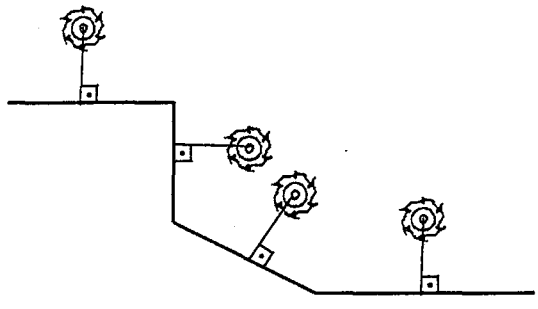
Şekil 6.6 Yüzey Uydurma İşleminin Akış Diyagramı

7. YÜZEY NORMAL VEKTÖRLERİNİN HESAPLANMASI

7.1. Yüzey Normal Vektörleri

Yukarıda bahsedilen döndürme, taşıma, eğri ve yüzey uydurma işlemleri, doğrudan bilgisayar ekranında bir parça şekline ait görünümlerin bellekte tutulmasını ve bellek içerisinde bu parça şeklini içeren veri dosyasına yapılacak müdahale ile ekran üzerinde bu parça şekillerinin düzenlenmesini içermektedir. Ancak SMARTCAM, MASTERCAM, APT, ANVIL gibi çizim programları sadece parça şeklinin ekran üzerinde üç boyutlu görünümü ve bu görünümler üzerinde taşıma, döndürme gibi işlemlerle sınırlı değildir. Bu tür çizim programlarında, parça şeklinin işlenebilmesi için bellek içerisindeki verilerden faydalanılarak, CNC tezgahları için kod üretme işlemi de gerçekleştirilmektedir. Bu işlem gerçekleşirken bellek içerisinde bulunan veri noktalarına göre, takımın izleyeceği yolun, takım yarıçapının hesaplanılması ve PC bilgisayar üzerinden CNC tezgahına aktarılabilmesi gerekmektedir (Ünsaçar 2003).

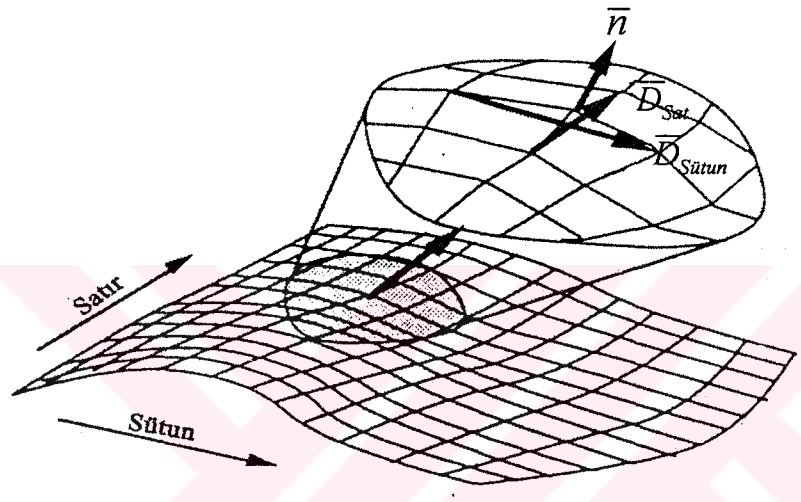
Bilgisayar üzerinde çizim yaparak elde edilmiş ve bellekte bir veri dosyası şeklinde bilgileri bulunan parça şeklinin CNC tezgahına aktarılabilmesi için öncelikle kesici yolunun yüzeye göre hesaplatılması gerekmektedir. CNC tezgahının bir özelliği ise kesicinin bulunduğu her noktada yüzeye dik olarak bulunması gerekmektedir. Yüzey normallerinin hesaplanması bu işlemi sağlamaktadır. Örnek olarak aşağıdaki şekilde kesicinin her pozisyonda yüzeye dik bir açı ile yaklaştığı görülmektedir (Şekil 7.1).



Şekil 7.1 Kesici Takımın Yüzeye Her Noktadan Dik Konumlanması

7.2. Matematiksel İfade

Yüzey normallerinin hesaplanmasında bulunulan noktada takım yerini göstermede kullanılacak vektör $\overline{n}$ olarak temsil edilmek üzere her bir nokta için bu $\overline{n}$ vektörü aşağıdaki şekilde görüldüğü gibi hesaplanabilir (Şekil 7.2). Burada köşe noktalarda da kesici takımın yüzeye dikliğini sağlamak amacı ile bir önceki ve bir sonraki satır ve sütunlardaki veri noktaları göz önüne alınmış ve böylece tüm noktalarda takımın yüzeye dik olarak hareketi sağlanmıştır. Bu işlem için aşağıdaki iki formül kullanılmaktadır.



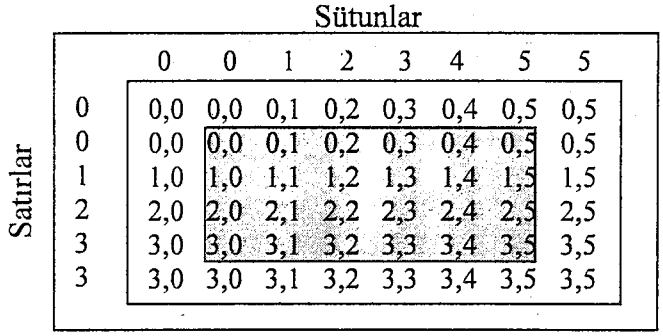
Şekil 7.2 Bir Yüzeye Ait Yüzey Normallerinin Gösterimi

$$\overline{n} = \frac{\overline{D}_{sat} * \overline{D}_{sütun}}{\left|\overline{D}_{sat} * \overline{D}_{sütun}\right|} \tag{5}$$

ve

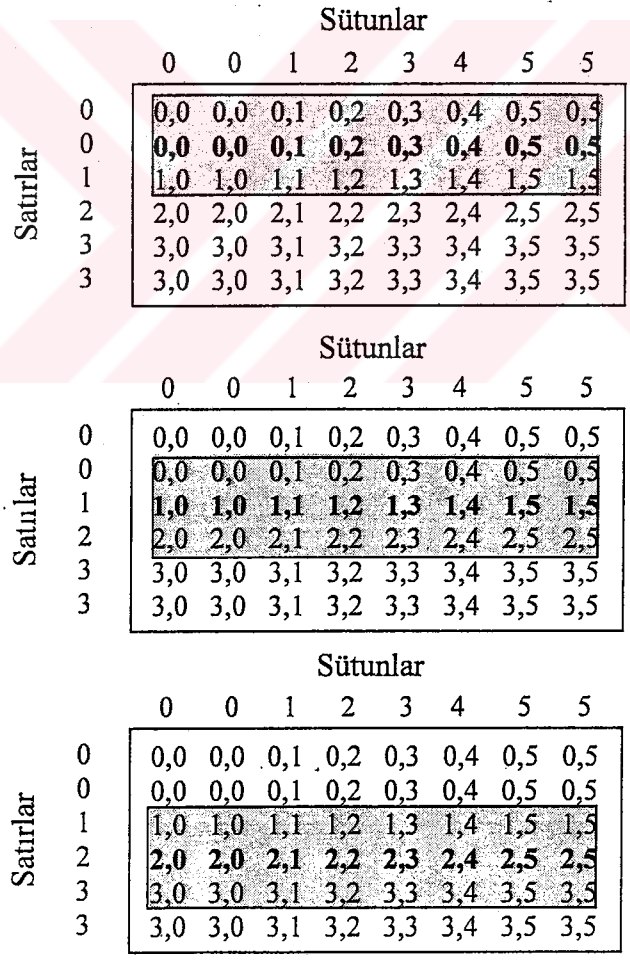
$$\overline{D}_{sat} = P_{[sat+1,Süt]} - P_{[sat-1,Süt]}$$
$$\overline{D}_{süt} = P_{[sat,Süt+1]} - P_{[sat,Süt-1]} \tag{6}$$

Bu iki formül bellek içerisindeki her bir satır ve sütuna ait ilk ve son satır kopyalanarak genişletilmiş matrise uygulanmaktadır. Bellekte dört satır ve altı sütunlu bir matrisin genişletilmesine aşağıdaki gibi bır örnek verilebilir (Şekil 7.3). Bu örnekteki gölgeli tonlu olan kısım matrisin bellekte tutulan kısmıdır.



Şekil 7.3 Bellek İçerisinde Normalleri Hesaplatılacak Bir Yüzeye Ait Verilerin Matris Gösterimi

Matris şeklinde yukarıdaki şekilde dönüştürülerek bellek içerisinde elde edilmiş, genişletilmiş matrise üçer satır olacak şekilde yukarıda verilen formül 5 ve 6 uygulanarak normal vektörlerinin hesaplanması tamamlanmış olunur. Bu işlem ise aşağıdaki sırada olmaktadır (Şekil 7.4).



Şekil 7.4 0, 1 ve 2. Satırlara Ait Normal Vektörlerinin Hesaplanması İçin Verilerin Formüllere Yüklenme Sırası

Burada öncelikli olarak yüzey dosyası içerisinden belleğe yukarıdaki düzende veriler alınmaktadır. Sonrasında ise tüm noktalar için her bir satırdan yararlanılarak yani satır satır taramak suretiyle yüzey normal vektörleri hesaplattırılmaktadır. Burada bellek içerisine üç adet satır alınmasına karşın ortadaki satır formül 5 ve 6 uygulanarak hesaplatılmaktadır. Önemli olan diğer bir nokta ise satır olarak taramak yüzey normallerini hesaplamak için yeterli olmaktadır. Bir başka ifade ile sütunlar üzerinde işlem yapmaya gerek kalmamaktadır (Vickers 1990).

7.3. Akış Diyagramı

Program, verinin tutulmasını sağlayan dosyadaki her bir satır ve sütunun genişletilmesini ve genişletilmiş matris şeklindeki verilerin satır satır normal vektörlerin bulunması için kullanılan formüllere uygulanarak elde edilen verilerin yeni bir dosyaya yazılması sonucu bu işlemi yerine getirmektedir.

EK-D'de programa ait veri kodları (YUZNORM.C) verilmiştir.

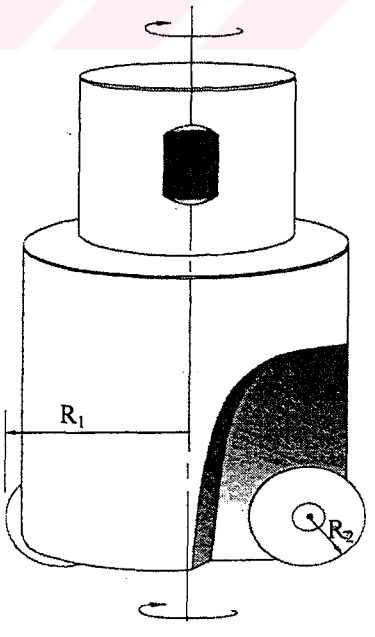
8. KESİCİ TAKIM OFSETLERİNİN HESAPLANMASI

8.1. Kesici Takım Ofsetleri

Yüzey normallerinin elde edilmesinden sonra elde edilen bu veri dosyası kullanılarak G kodları olarak adlandırılan kod sistemi üretilebilir. Ancak bu işlem hatalı olacaktır. G kodlarının üretilmesinden önce kullanılacak takım yarıçap değerlerinin yani ofset değerlerinin yüzeyden telafisi gerekmektedir. Çünkü takımın yapacağı hareketlerin hesaplatılmasında takımın şekli ve yarıçap değerinin göz önünde bulundurulması gerekmektedir. Bu değerlere göre takım ofset değerleri de yüzey normal vektörlerine ilave edilerek takımın izleyeceği yol tam olarak yansıtılmaktadır.

8.2. Kesici Takım Türleri

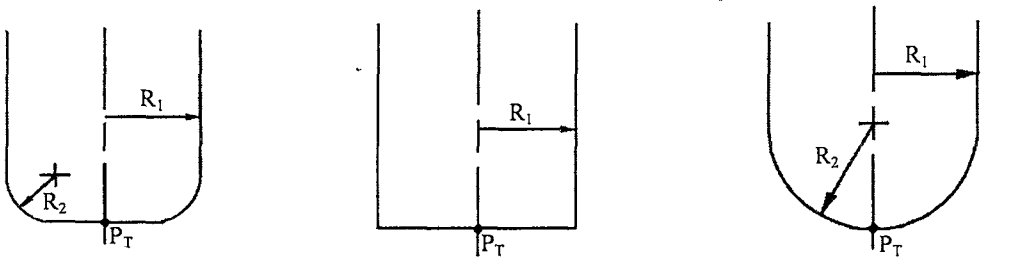
Aşağıda genel şekilli bir kesici takım görülmektedir (Şekil 8.1). Şekilde de görüldüğü gibi iki adet yarıçap değeri (R_1 ve R_2) bulunmaktadır. Bu iki değerin birbirinden farklı olması durumunda genel şekilli bir kesici takım, $R_1=R_2$ olması durumunda yuvarlak uçlu bir kesici takım ve $R_2=0$ olması durumunda da köşeli kesici takım elde edilmektedir.



Şekil 8.1 Genel Şekilli Bir Kesici Takım

8.3. Matematiksel İfade

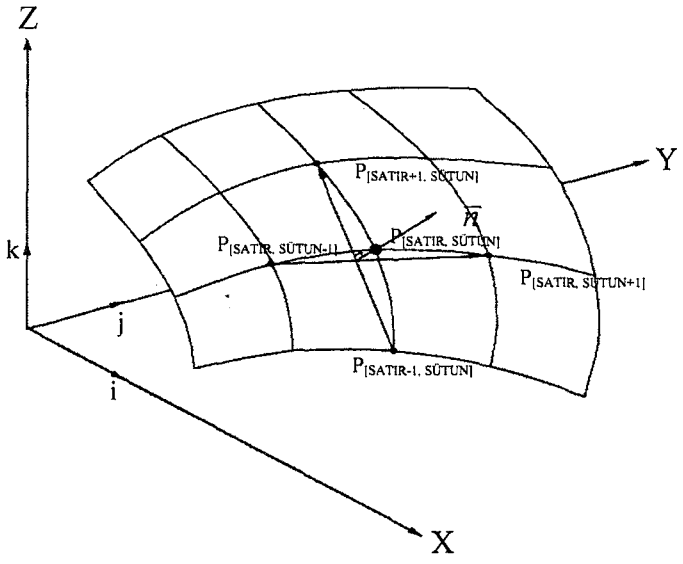
Takımın yüzeye olan mesafesini hesaplayabilmek için takım üzerindeki bir noktanın referans alınması gerekmektedir. Bu nokta için takımın merkez noktası kullanılmaktadır (P_T) (Şekil 8.2). Tüm alt programlar için kesici takım üzerindeki bu nokta, referans noktası olarak kullanılmaktadır.



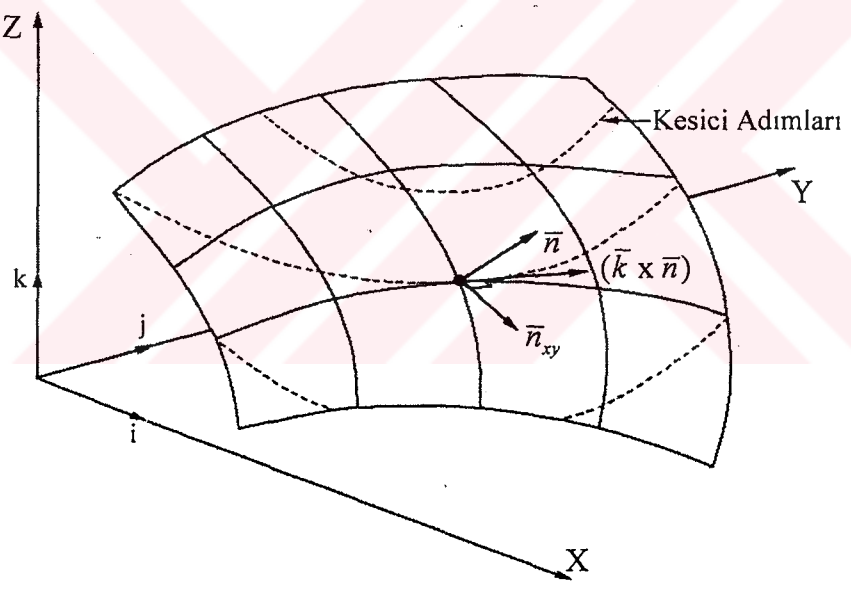
a)Genel Şekilli Kesici Takım b)Sivri Uçlu Kesici Takım $R_2=0$ c)Yuvarlak Uçlu Kesici Takım $R_1=R_2$

Şekil 8.2 Kesici Takım Türleri ve Yarıçap Değerlerinin İfadesi

Kesici takım yolunun tam olarak hesaplanması için bilinmesi gereken bilgilerden bir tanesi bir önceki konu olan veri noktası içerisindeki her bir nokta için yüzey normal vektörünün ($\bar{n}$) hesaplanmasıdır. Bu normal vektör aşağıdaki şekilde görüldüğü gibi ifade edilebilir (Şekil 8.3). Ancak bilinmesi gereken diğer bir bilgi olan kesici takım ofsetinin hesaplanması içinde yine veri dosyası içerisindeki her bir nokta için diğer bir vektörün bulunması gerekmektedir. Kesici takımın işlem esnasında sadece X-Y düzlemlerinde işlem yaptığı düşünülürse, hesaplamalarda kullanılacak normal vektör $\bar{n}_{xy}$ şeklinde ifade edilmektedir. Hesaplama için gerekli olan bir diğer vektör ise $\bar{n}_{xy}$ vektörüne dik olarak hesaplanarak elde edilen ($\bar{k}$ x $\bar{n}$) vektörüdür. Bu vektör ise yatay yönde yüzeye dik olarak hareket eden kesici takımın yüzeyde ilerleyeceği adımlarının her birini temsil etmektedir. [($\bar{k}$ x $\bar{n}$)x$\bar{k}$] ise bu yüzey adımlarının bulunduğu noktadaki normal vektörünü ifade etmektedir. Aşağıdaki şekillerde bu vektörlerin yüzey veri dosyası üzerindeki gösterimi görülmektedir (Şekil 8.3 ve 8.4).



Şekil 8.3 Yüzeyin Bir Noktası Üzerinde Normal Vektörün Gösterimi



Şekil 8.4 Bir Veri Noktası Üzerinde X-Y Düzlemine Göre Normal Vektörün Gösterimi

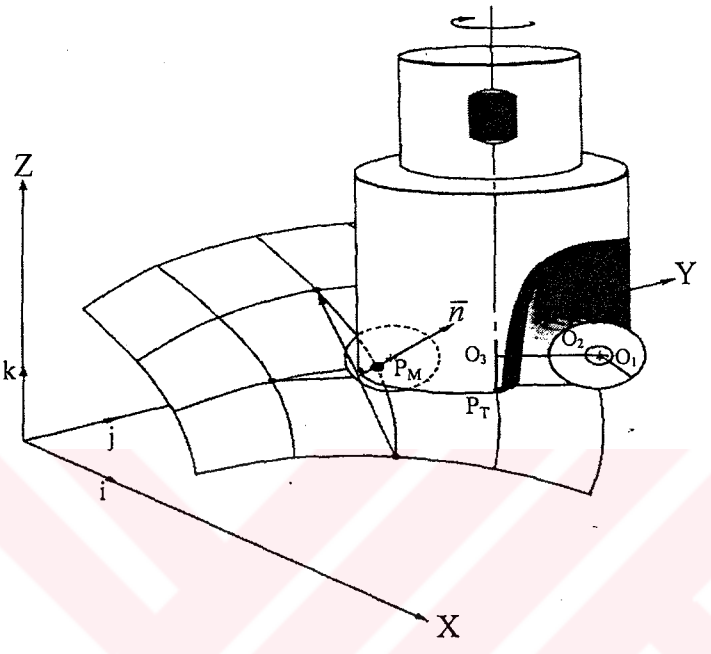
Bu sonuçla yüzey normal parça vektörü aşağıdaki gibi hesaplanabilir.

$$\overline{n}_{xy} = \frac{\left(\overline{k} \times \overline{n}\right) \times \overline{k}}{\left|\left(\overline{k} \times \overline{n}\right) \times \overline{k}\right|} \tag{7}$$

Buradan X-Y düzlemi içerisinde $\overline{n}$ yüzey normal vektörünün hesaplanılması sayısal olarak gerçekleştirilebilir. $\overline{n}$ 'de X ve Y düzlemleri göz önünde bulundurulurken, Z

düzlemi 0'a ayarlanmaktadır. Bu durumda normal vektör $\overline{n}_{xy}$ aşağıdaki gibi hesaplanır;

$$\overline{n}_{xy} = \left[\frac{n_x}{|n_{xy}|}, \frac{n_y}{|n_{xy}|}, 0 \right] \tag{8}$$



Şekil 8.5 Yüzey Veri Noktasında Genel Şekilli Bir Kesici Takımın Konumlandırılması

Genel şekilli bir kesici takım için P_T olarak kabul edilen takım referans noktasının kesin koordinatı, yukarıdaki şekilde görüldüğü gibi yüzey nokta vektörleri ve takım ofset vektörünün (veya kesici takımın yüzeyle temas noktası olan P_M ve referans noktası olan P_T) eklenilmesi ile elde edilebilir (Şekil 8.5).

Takım referans noktasının (P_T) yeri aşağıdaki gibi hesaplanabilir.

$$[P_T] = [P_M] + [O_1] + [O_2] + [O_3] \tag{9}$$

Takım yarıçap değerlerine bağlı olarak ofset vektörleri ise;

$$\begin{aligned} \overline{O}_1 &= \overline{n} R_2 \\ \overline{O}_2 &= \overline{n}_{xy} (R_1 - R_2) \\ \overline{O}_3 &= \overline{k} R_2 \end{aligned} \tag{10}$$

Yuvarlak uçlu bir kesici takım için ise ofset değeri

$$O_{yuvarlak} = \bar{n}R_2 - \bar{k}R_2 \quad (11)$$

Köşeli uçlu bir kesici takım için ise ofset değeri

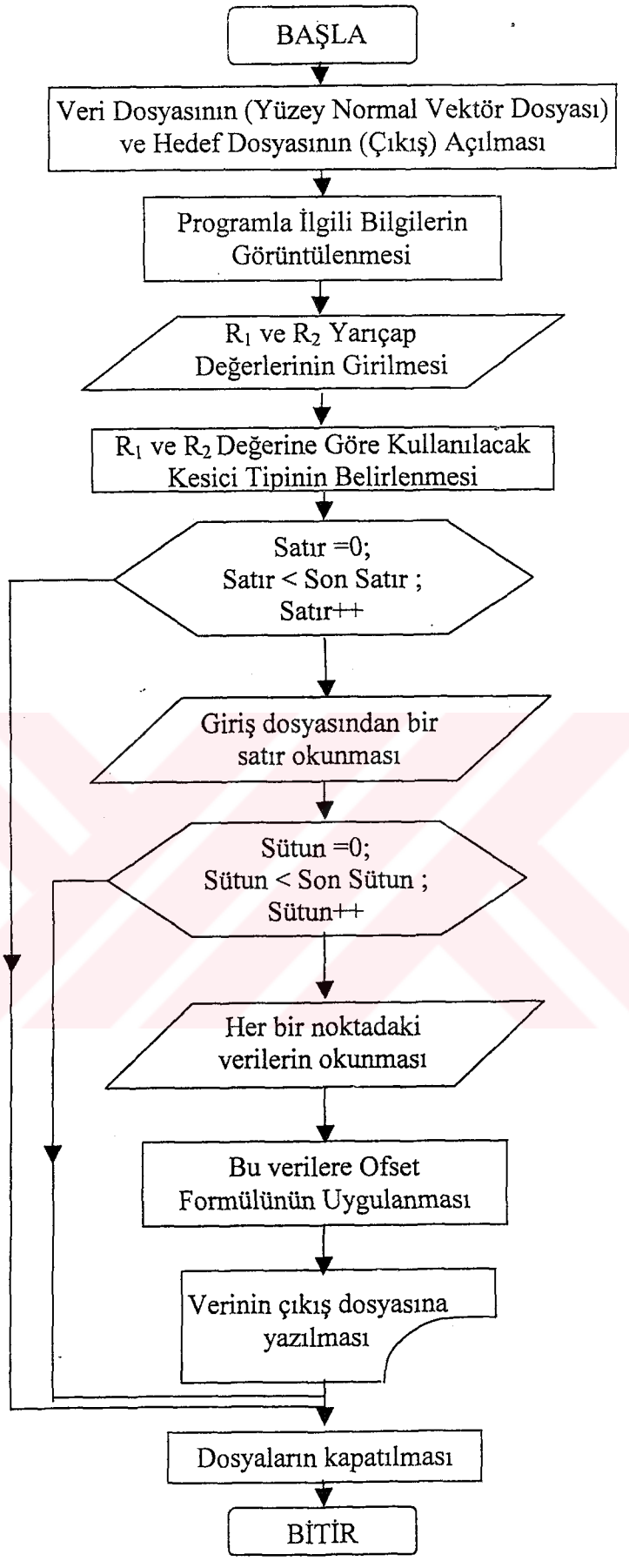
$$O_{köşeli} = \bar{n}_{xy}R_1 \quad (12)$$

olarak hesaplanmış olur (Vickers 1990).

8.4. Akış Diyagramı

(OFSETHES.C) isimli program matris şeklindeki daha önce hesaplatılmış olunan yüzey normal vektörlerini içeren dosyadaki veri noktalarına, program içerisinde belirtilen bilgilere göre hesaplatmaktadır. Program içerisinden kullanılacak kesici takım türü seçilebilmektedir. Eğer köşeli uçlu bir kesici takım kullanılacaksa otomatik olarak R_2 yarıçapı 0 ve yuvarlak uçlu bir kesici takım kullanılacaksa $R_1=R_2$ olacak şekilde ofset değerleri hesaplatılmaktadır. Programda öncelikli olarak normal vektörleri içeren giriş dosyası ve sonuçların yazılacağı bir dosya açılmakta, ofset değerleri de formül 8 ve 10 kullanılarak hesaplatılmaktadır. Son olarak ise bu hesaplatılan ofset değerlerine ait veriler çıkış dosyasına yazılmaktadır.

OFSETHES.C isimli program C programlama dilinde yazılmış olup, programa ait kodlar EK-E'de verilmiştir.



Şekil 8.6 Kesici Takım Ofsetlerinin Hesaplanmasına Ait Akış Diyagram

9. KESİCİ TAKIM YOLUNUN BULUNMASI

9.1. Kesici Takım Yolu

Kesici takım yolu, takımın hareketi için yüzey normal vektörleri ile kesici takım ofset değerlerinin eklenilmesi anlamına gelmektedir. Böylece hesaplatılan bu iki değer sonucunda kesici takımın yüzey üzerinde hareketi esnasında tüm detaylar göz önüne alınarak hatasız bir işleme gerçekleştirilmiş olacaktır.

9.2. Matematiksel İfade

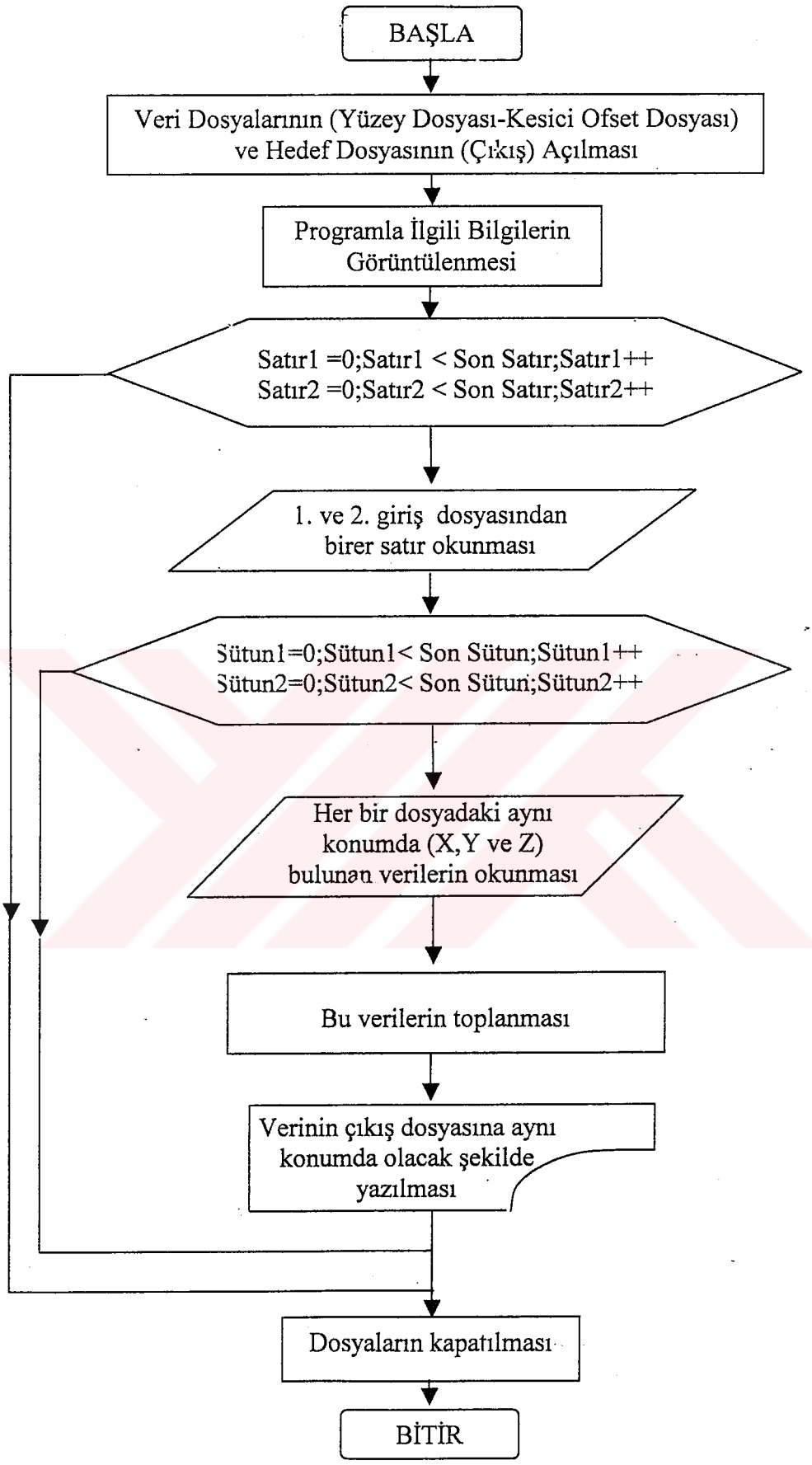
Kesici takım yolunun hesaplanması için daha önce bahsedilen iki vektör olan yüzey normal vektörü ($\vec{n}$) ofset vektör değerinin (O_{normal}, $O_{yuvarlak}$ veya $O_{köşe}$) ilave edilmesi ile elde edilmiş verilerin yüzey veri dosyasına ilave edilmesi gerekmektedir. Sonuç vektörü S ile temsil edilirse formül 13'teki gibi hesaplanabilir;

$$S=P_{[Satır,Sütun]} + (\vec{n} + O) \tag{13}$$

9.3. Akış Diyagramı

OFSETEKLE.C isimli program kesici takım yolunun hesaplanması için kullanılmaktadır. Kesici takım yolu hesaplandıktan sonra bir sonraki konuda ele alınacak ve CNC tezgahları için anlamlı olan G kod dosyası bu dosya dikkate alınarak hesaplatılmaktadır. Böylece takımın telafisi yüzey dosyasına yansıtılarak nihai şekli elde edilecek takım hareketleri üretilmektedir. Programda veri olarak iki adet giriş ve bir adette çıkış dosyası bulunmaktadır. Yüzey veri dosyası ile takım hareketlerini içeren sonuç dosyası olan kesici takım ofset dosyası giriş dosyasıdır. G kodunda kullanılacak bir dosya ise çıkış dosyası olarak kullanılmaktadır. Programda bu iki giriş dosyası açıldıktan sonra dosya uzunluklarının aynı olup olmadığı da kontrol edilmektedir. Bir döngü ile iki dosyaya ait aynı sıradaki veri noktaları (X,Y ve Z düzlem değerleri) toplanılarak çıkış dosyasının aynı konumuna yazılmaktadır. Programın akış diyagramı aşağıda görülmektedir.

OFSETEKLE.C isimli program C programlama dilinde yazılmış olup, programa ait kodlar EK-F'de verilmiştir.



Şekil 9.1 Kesici Takım Yolunun Bulunmasına Ait Akış Diyagram

10. G KODLARININ ÜRETİLMESİ

10.1. G Kodları

CNC olarak adlandırılan Bilgisayar Destekli Nümerik Kontrollü tezgahların (CNC-Computer Numerical Control) temelinde bir dizi sayı ile belirtilen komutlar sayesinde, tezgah hareketleri ve takımların değiştirilmesi gibi birçok işlevin kontrolünü yapabilmemizi sağlayan sistemler yatmaktadır. Bu anlamda CNC tezgahı bilgi girilebilen, bu bilgileri işleyen ve işlenilen bilgileri kullanarak verilen komutlara göre hesaplama yaparak takım hareketini sağlayan tezgahlardır. Bilgi girişi harfler ve şekillerle olmaktadır. Bilgisayarın anlayabileceği şekilde bilgilerle beslenmesi olarak ifade edilebilecek bilgi giriş işlemine programlama denilmektedir. Bir başka ifade ile bilgisayarın bildiği ve anlayabildiği bir dilde, bilgilerin uygun bir sırada listeler halinde girilmesidir. Özel olarak CNC'lerde işlem yapabilmek için ortaya çıkarılmış standart kodlarla bilgi girişi yapılmaktadır. CNC'ler ancak bu dilden anlamaktadır ve bu kodlara genel olarak **G kodları** denilmektedir.

Takım hareketlerinin kontrol edilmesi bir dizi basit komut kodlarının, yardımcı fonksiyonların ve tezgah döndürme hareketlerini ayarlayan komutların mantıklı bir sıra ile kullanılması sonucu elde edilmektedir ve **G00** şeklinde kullanılmaktadır. Her bir komutun farklı bir kullanım şekli ve farklı bir görevi bulunmaktadır. Bu komutlar hazırlık komutları ve yardımcı komutlar olarak iki grupta toplanabilir. Hazırlık komutları; talaş kaldırma işlemi için gereken fonksiyonların tezgah kontrol birimine girilmesinde kullanılır. Örneğin; kontrol ünitesine, program içerisinde boyutsal olarak verilen kızak hareketlerinin, metrik veya ingiliz sisteminde mi olduğu veya iş mili dönüş yönünün saat yönü veya saat yönünün tersine mi olduğu bildirilmek zorundadır. Bu tür ayarlamaların tamamında hazırlık komutları kullanılmaktadır ve G kodları ile gösterilmektedir. İkinci grup olan yardımcı komutlar ise hazırlık komutlarını desteklemek amacı ile zaman zaman destekleyici görevleri yerine getiren komutlar olarak kullanılmaktadır. Örneğin; talaş kaldırma esnasında soğutma işlemini gerçekleştirmek amacı ile soğutma sıvısı kullanılabilmektedir.

Bir parça üretim programı, bilgi satırları içeren blokların düzenlenmesinden oluşmaktadır. Her bir blok özel makine işlemlerini yerine getirmek için gerekli bir satır komutu içermektedir (Boz 1998). Aşağıda bir grup komutu içeren örnek bir program bloğu görülmektedir:

N200 G01 X120 Y100 Z80 F150;

Tüm programlar birden fazla blok yada satırdan meydana gelmekte ve en son **M30** komutu içeren satır ile de bitirilmektedir. İşte burada N ile temsil edilen 200 numara tüm satırlar içerisinde bu satırın hangi sırada geldiğini göstermektedir. Genellikle program içerisinde her bir blok programcı tarafından 10'ar 10'ar artırılır. Örneğin;

"

"

N120 <komut veya komutlar>

N130 <komut veya komutlar>

N140 <komut veya komutlar>

"

"

N200 G01 X120 Y100 Z80 F150;

Örnek olarak verilen 200 numaralı bu satırda sadece G01 komutu yani tek komut kullanılmıştır. Bu komutun anlamı o an takımın bulunduğu konuma (yani bir önceki X,Y ve Z değerlerinden) X=120 mm, Y=100 mm ve Z=80 mm ekleyerek, 150 mm/dakika hızla parçayı kesmek suretiyle takımı istenilen koordinata taşır.

Örnek: G Kodları İle Gerçekleştirilmiş Bir Programının Başlangıcından Bitişine Kadar Sıralaması

Komut	***Anlamı***
Ø006;	Program numarası
N10 G91 G28 Z0;	Z ekseninde referans noktasına git
N20 G91 G28 X0 Y0;	X ve Y eksenlerinde referans noktasına git
N30 G92 X250 Y125 Z75;	Parça koordinatlarının girilmesi

N40 T18;	18 numaralı takımı kullanmak için otomatik takım değiştiricisini çağırır.
N50 T19 M06;	18 numaralı takım ile parça işlenirken, 19 numaralı takımı hazır halde bekletir.
N60 G90 G21 G43 G00 Z100 H18;	Metrik koordinat sisteminin kullanılmasını (G21), mutlak koordinat sisteminin kullanılmasını (G90) ve takım uzunluğunun sağa telafi edilerek(G43) Z100 koordinatına kesme işlemi yapmadan takımın hızlı yanaşmasını (G00) sağlar.
N70 G00 X0 Y0;	X=0 ve Y=0 noktalarına takım hızlı bir şekilde yanaşır.
N80 S2000 M03;	Aynanın dönüş hızının ayarlanması (S2000) ve aynanın saat yönünde döndürülmesinin sağlanması (M03).
N90 N100 “ “ N190 N200	Program satırları
N210 G91 G28 Z0;	Z ekseninde referans noktasına git
N220 G91 G28 X0 Y0;	X ve Y eksenlerinde referans noktasına git
N230 M05;	Aynanın durdurulması
N240 M30;	Program sonu
N250 %;	Şeride yazımın sonlandırılması

Bu programda, programın başlamasıyla daha önceden program içerisinde ayarlanmış başlangıç X,Y ve Z değerlerine takımın gönderilme işlemini sağlayan G91 ve G28 komutları iyi bir örnek olarak görülmektedir. Bu başlangıç noktaları tezgah üreticileri tarafından sınırlandırılmışlardır ve normalde X_{max}, Y_{max} ve Z_{max} olarak temsil edilmektedirler. Güvenlik amaçlı olarak Z ekseni genellikle ilk başlangıç noktasına döndürülmektedir. G92 X250 Y125 Z75; komutu işlenen parçanın başlangıç pozisyonları ile bulunulan pozisyon arasında ilişki kurmaktadır. T18 adresinde tutulan takım, M06 komutu ile sisteme eklenilmiş ve saklanıldığı yerde kullanıma hazır halde bekletilmektedir. Kesin koordinatlar milimetre ölçü

biriminde kullanılmaktadır. Takım hızlı bir şekilde Z=100 mm noktasından X=0 mm ve Y=0 mm (veya başka bir deyişle yeni bir başlangıç noktasına) taşınmaktadır. Ayna hızı 2000 mm/dak. olarak S2000 komutu ile ayarlanılmakta ve M03 komutu ile de döndürülmeye başlanılmaktadır (Ünsaçar 2003).

Özetle G Kod sistemini CNC tezgahının programlanabilmesini sağlayan komutlar dizisi olarak adlandırabiliriz.

10.2. G Kod Dosyasının Üretilmesi

Eğimli yüzeylerde kesici takım hareketlerinin makine koduna dönüştürülmesinde yüzey üzerinde takıma bir dizi doğrusal veya dairesel hareket kazandırılmaktadır. İlgili komutlar kullanılarak takımın hareketi gerçekleştirilmektedir. Bunun haricinde kullanılan çok sayıda komut bulunmaktadır. Finiş (finish) işlemi olarak adlandırılan G70 komutu ile yüzey üzerindeki bırakılmış pay daha hassas bir şekilde alınmakta ve yüzey pürüzsüzlüğü sağlanmaktadır. Program içerisinde programlara mutlaka isim verilmelidir. S parametresi ile milin dönüş hızı ve F parametresi ile de takımın kesme işlemi sırasında dakikada ne kadar keseceği belirtilmelidir. Buna benzer çok sayıda komut kullanılarak G kod dosyası üretilmektedir. Ancak bu dosya üretilirken bu işlemlerin belli bir sırada olması gerekmektedir. Örnek olarak programa verilecek isim ilk başta olmak zorundadır veya programa başlamadan önce sıfır noktasının tanımlanması gerekir.

Burada önemli olan nokta, kesici takım yolu dosyasındaki geometrik bilgilere göre G kodlarının üretilmesi gerekmektedir. Bunun haricinde yukarıda bahsedilen önemli noktalar aşağıdaki tanımlama kurallarına göre ayarlanmaktadır.

- Program isminin (numarasının) girilmesi (bu bir CNC tezgah programının başlığı olmaktadır), sıfır noktasının tanımlanması gibi başlangıç ayarları sırası ile yapılmaktadır,
- $X_{Başlangıç}$, $Y_{Başlangıç}$ ve $Z_{Başlangıç}$ şeklinde kesici takım başlangıç noktalarının tanımlanması,
- Feedrate, yani keserek ilerleme (G01 komutu) yapılırken takımın dakikadaki ilerleme hızının girilmesi,
- Spindle speed yani milin dakikadaki dönüş hızının girilmesi,

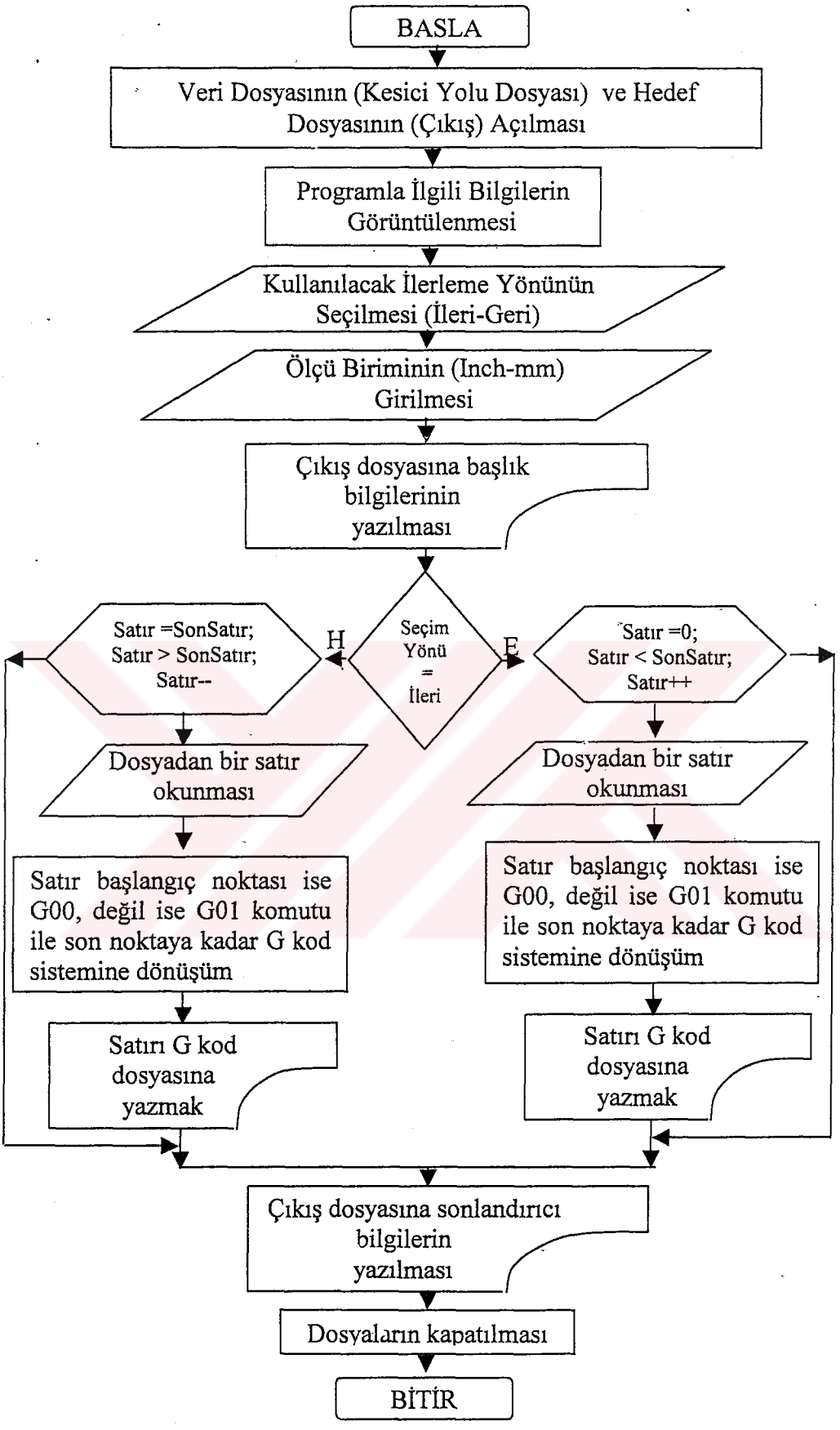
- Kesim işlemleri – takım işleme kodları ve şekli, normalde kesici takım yolu dosyasında matris şeklinde bulunan veri noktaları kullanılarak üretilmektedir. Takım hareketi ileri veya geri yönlendirilebilmektedir. İleri seçilmesi durumunda veri dosyasındaki ilk noktadan son noktaya kadar tek tek, satır satır kesici takım hareketleri hesaplanmaktadır. Bir satır tamamlandıktan sonra önceki satırdan kalan bilgiler temizlenerek bir sonraki satırın tanımlanmasına geçilir. İstenilirse program içerisinde geri seçeneği seçilebilir. Bu durumda da ters yönlü yani son satırdan ilk satıra kadar takıma hareket kazandırılmış olur.
- Ölçü sistemi – yüzey tanımlamasında kullanılacak ölçü birimi milimetre veya inch olarak seçilebilmektedir.

10.3. Akış Diyagramı

GKODYAP.C isimli program üç boyutlu yüzeylerin işlenmesi için G kodunu üretmektedir. Kesici takım yolu dosyasından okuduğu verilere göre G kod dosyasına bilgileri üretmektedir.

Programda ilk olarak giriş ve çıkış dosyaları açılmaktadır. Programa başlık verildikten sonra, kullanıcıdan gerekli olan sıfır noktası, $X_{Başlangıç}$, $Y_{Başlangıç}$ ve $Z_{Başlangıç}$ değerleri ile ilerleme hızı, mil dönüş hızı ve ölçü sistemi bilgileri istenmektedir. Sonrasında istenilen ilerleme şeklini seçmek için (ileri – geri) kullanıcıya bir seçenek sunulmaktadır. Hesaplamalara göre G kodları üretilmekte ve son olarakta G kod dosyası mil dönüşünün durdurulması gibi işlemlerle bitirilmektedir. Bu programın hazırlanmasında FANUC firması standartları göz önüne alınarak G kodları ürettirilmiştir. Programın akış diyagramı kabaca aşağıdaki şekilde görülmektedir.

GKODYAP.C isimli program C programlama dilinde yazılmış olup, programa ait kodlar EK-G'de verilmiştir.



Şekil 10.1 G Kodlarının Üretilmesine Ait Akış Diyagram

Çıkış dosyasına başlık ve sonlandırıcı bilgiler aşağıdaki gibi standart olarak yazılmaktadır. Değişen tek şey ölçü birimidir. mm seçilmişse G21, inch seçilmişse G20 komutu kullanılmaktadır.

Başlık Bilgileri

N1 O23;
N2 G92 $X_{Baş}$, $Y_{Baş}$, $Z_{Baş}$;
N3 G21; (veya N3 G20;)
N4 G90;
N5 S(devir hızı);
N6 T(takım numarası);

N...
N... } G kod satırları
N...

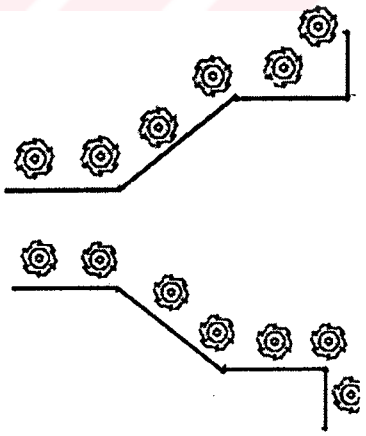
Bitiş Bilgileri

(100 satırdan oluştuğu varsayılırsa)
N98 G00 $X_{Baş}$, $Y_{Baş}$, $Z_{Baş}$; (Takımı başlangıç noktasına taşır)
N99 M05; (Mil Dönüşünü durdurur)
N100 M30; (Programı bitirir)
%

11. YÜZEY NORMAL VEKTÖRÜNÜN TERSYÜZ EDİLMESİ (INVERT)

11.1. Tersyüz İşlemi

Yüzey normal vektörlerinin yönü formül 5'te verilen eşitlikten yararlanılarak satır ve sütunların kesişimi şeklinde düzenlenmektedir ve bu takımın ilerleyeceği hareket yönünü de ortaya koymaktadır. Bir yüzeye ait şekil içbükey(concav) veya dışbükey(convex) olabilmektedir. Ancak yüzey normallerini hesaplayan program içerisinde sonradan yapılacak işlemler göz önüne alınarak malzemenin hangi yönünün kullanılacak kısım, hangi yönünün ise artık kalan kısım olacağı bilinememektedir. Yani program içerisinde kullanıcının şekli tersine çevirerek şeklin ne durumda olacağını veya ne yönde ilerleyeceğini yüzey normalleri hesaplayan programla kestirmesi mümkün değildir. Çizim programları içerisinde bu eksikliği gidermek amacı ile kullanıcının şekli tersine çevirmesine izin verilmektedir ve böylece farklı bakış açıları ile şekil bir bütün olarak incelenebilmektedir. Tersyüz işlemi bu amaçla kullanılmaktadır. Bir su bardağının üstünden bakmak ile ters çevirdikten sonra bakmak tersyüz işlemine örnek verilebilir. Aşağıdaki şekilde kullanıcının bir malzemeyi tersyüz etmesi ile ilgili örnek görülmektedir (Şekil 11.1). Burada hem yüzeyin hem de yüzey normal dosyasının giriş dosyası kullanılması ile takımın izleyeceği yolların tersyüz edilmesi görülmektedir.



Şekil 11.1 Bir Yüzey ve Tersyüz Edilmiş Hali İle Yüzey Normal Dosyasının Tersyüz Edilmesi ile Takım Hareketlerinin Değişimi

11.2. Matematiksel İfade

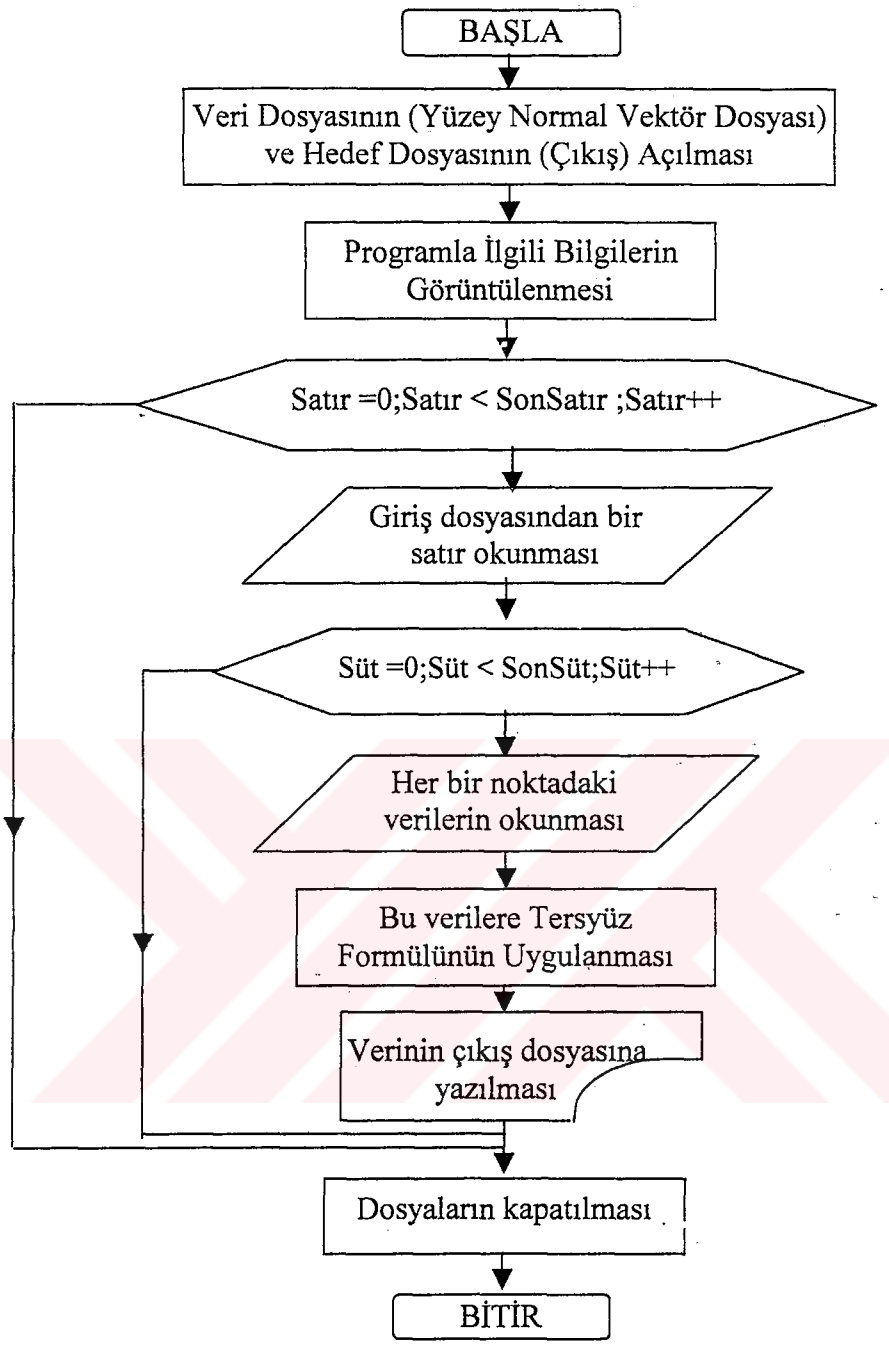
Tersyüz etme işlemi çok basit bir işlemdir. Bunun için veri dosyası içerisinde bulunan tüm noktalar invert edilmektedir. Yani -1 ile çarpılmaktadır. Formül olarak gösterilecek olursa;

$$P_{TERSYÜZ[Satır,sütun]} = -1 * P_{[Satır,sütun]} \quad (14)$$

11.3. Akış Şeması

TERSYUZ.C isimli program üç boyutlu yüzeylerin tersyüz(invert) edilmesi için kullanılmaktadır. Burada yüzey veri dosyası giriş dosyası olarak kullanılırsa yüzey dosyasının, yüzey normal vektör dosyası giriş dosyası olarak kullanılırsa takım hareketlerinin tersyüz edilmesi sağlanmış olmaktadır. Programda ilk olarak giriş ve çıkış dosyaları açılmaktadır. İçi içe açılacak iki döngü içerisinde giriş dosyasından satır ve sütunlar okutulmakta ve sonrasında da tüm bu noktalar (-1) ile çarpılarak çıkış dosyasına yazdırılmaktadır.

TERSYUZ.C isimli program C programlama dilinde yazılmış olup, programa ait kodlar EK-Ğ'de verilmiştir.

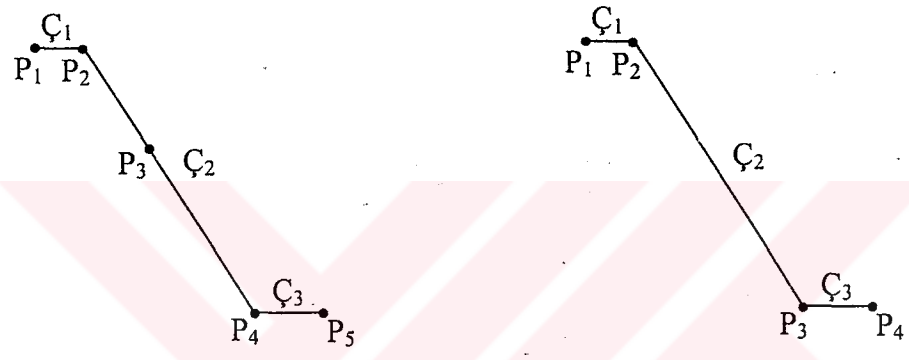


Şekil 11.2 Yüzey Normal Vektörünün Tersyüz Edilmesine Ait Akış Diyagram

12. YÜZEY YOĞUNLUĞUNUN AZALTILMASI

12.1. Yüzey Yoğunluğu

Yüzeyi oluşturan nokta sayısının azaltılması olarak ifade edilebilir. Bu işlem sayesinde yüzey veri dosyası daha basit şekilde ifade edilebilmektedir. Aşağıdaki şekilde görüldüğü gibi ilk şekildeki $Ç_2$ isimli çizgiyi oluşturabilmek için kullanılan P_3 noktası gereksiz olmaktadır (Şekil 12.1). Yani böyle bir nokta ikinci şekilde görüldüğü gibi kaldırılabilmektedir. Yüzey yoğunluğu azaltılarak bu gereksiz noktanın yüzey veri dosyasından atılması sağlanabilir.



Şekil 12.1 Yüzey Yoğunluğu Azaltılmadan Önce ve Sonra Bir Çizginin Noktalarla Gösterimi

Makina işlemi olarak bakıldığı zaman ise yüzey işleme süresince bırakılan finiş işlemine gelinceye kadar takım tek adımda yüzeyi işlemek yerine, belirlenecek adım kadar tekrarlanacak paso alma işleminden sonra finiş payına ulaşabilmektedir. Finiş işlemine gelinceye kadar ham bir parçanın işlenebilmesi için takımda belirlenecek her bir paso miktarının kesin koordinatlarda olması gerekmemektedir. Tüm koordinatların kullanılması yerine hassas olmayan kesim için yüzeye ait önemli noktaların kullanılması yeterli olacaktır. Bunun en önemli avantajı ise işlem zamanını azaltacaktır. CNC tezgahlarının seri üretim yaptığı göz önüne alındığı zaman bu çok önemli bir avantajdır (Vickers 1990).

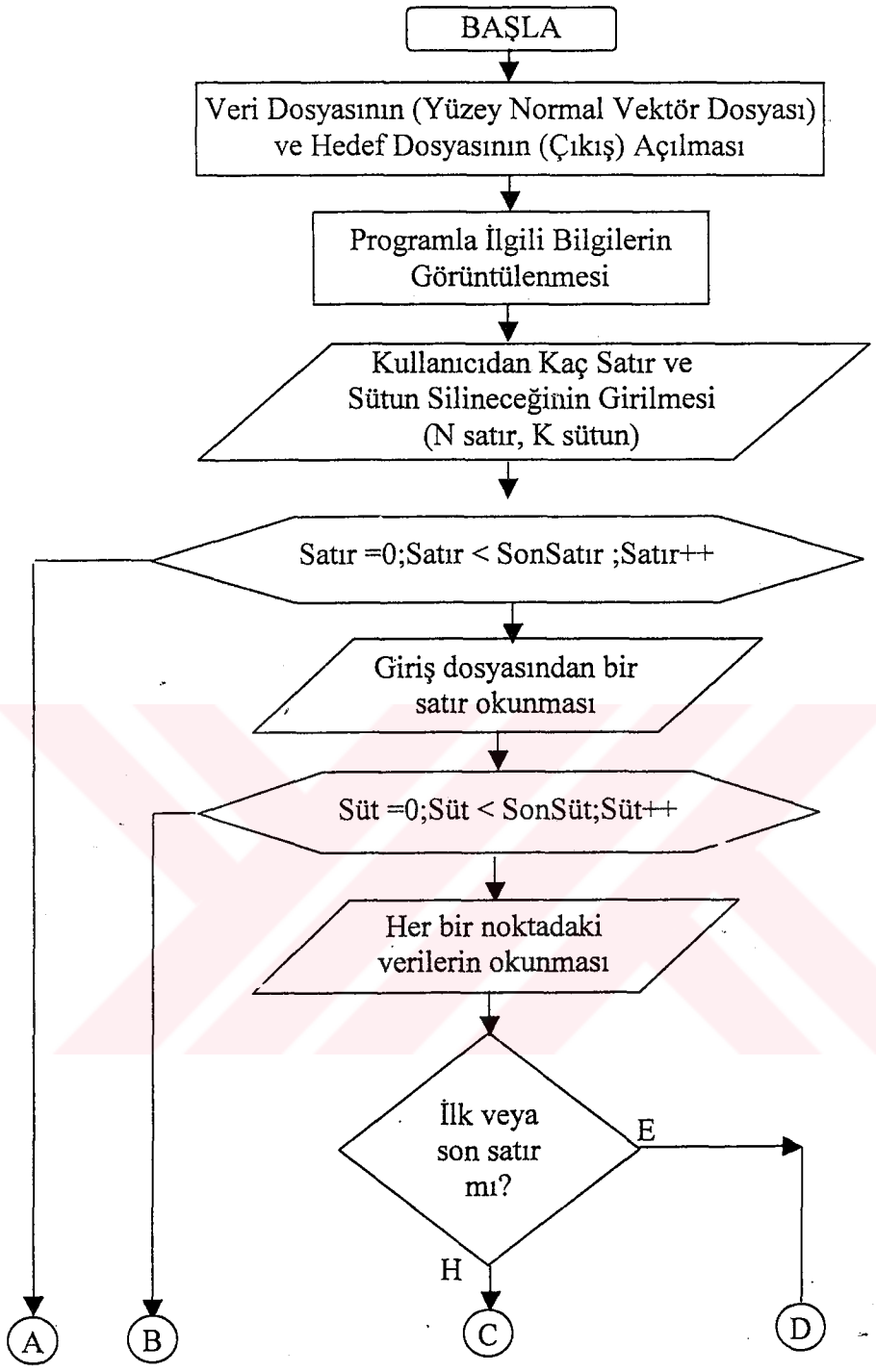
12.2. Matematiksel İfade

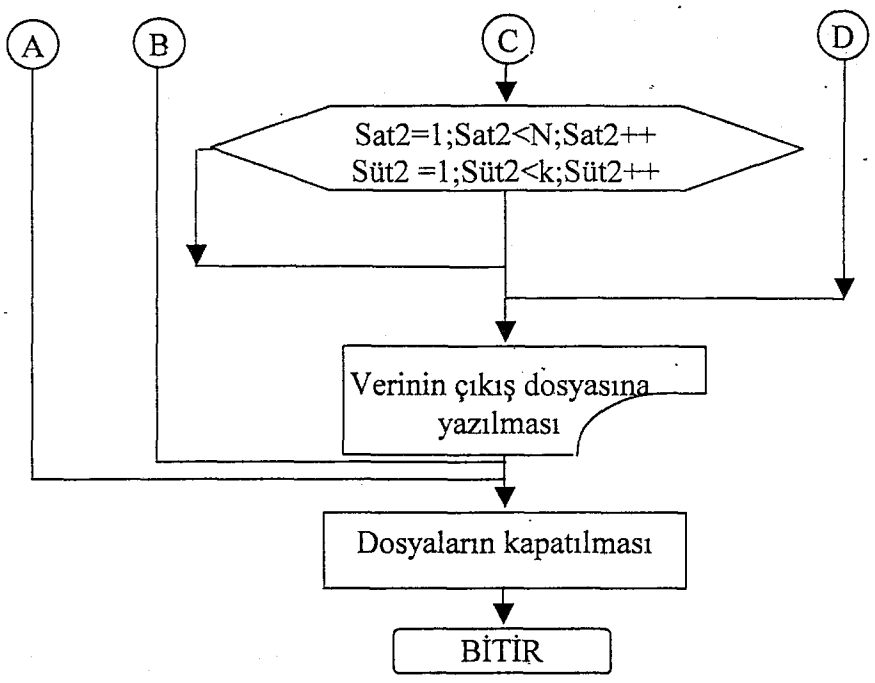
Bu işlem için kullanıcıdan istenilecek kaç satır ve sütun silineceği bilgisine göre açılacak iki döngü ile satır ve sütun bilgilerine müdahale edilmektedir. İlk satır ve sütun bilgileri başlangıç olduğu için bu bilgilere dokunulmamaktadır. Ancak ikinci satır veya sütundan itibaren dosyadan veri noktaları silinmeye başlamaktadır. Silinemeyen bir diğer satır veya sütun bilgisi ise son satır veya sütundur.

12.3. Akış Şeması

AZALT.C isimli program yüzey yoğunluğunun azaltılması için kullanılmaktadır. Burada yüzey veri dosyası giriş dosyası olarak kullanılmaktadır. Programda ilk olarak giriş ve çıkış dosyaları açılmaktadır. İçi içe açılacak iki döngü içerisinde giriş dosyasından satır ve sütunlar okutulmakta ve sonrasında da belirtilen satır ve sütun kadar ikinci satır ve sütundan başlanarak veri dosyası satırlarının çıkış dosyasına yazılması engellenmektedir.

AZALT.C isimli program C programlama dilinde yazılmış olup, programa ait kodlar EK-H'de verilmiştir.


BAŞLA
Veri Dosyasının (Yüzey Normal Vektör Dosyası) ve Hedef Dosyasının (Çıkış) Açılması
Programla İlgili Bilgilerin Görüntülenmesi
Kullanıcıdan Kaç Satır ve Sütun Silineceğinin Girilmesi (N satır, K sütun)
Satır =0;Satır < SonSatır ;Satır++
Giriş dosyasından bir satır okunması
Süt =0;Süt < SonSüt;Süt++
Her bir noktadaki verilerin okunması
İlk veya son satır mı?
E
H
A
B
C
D

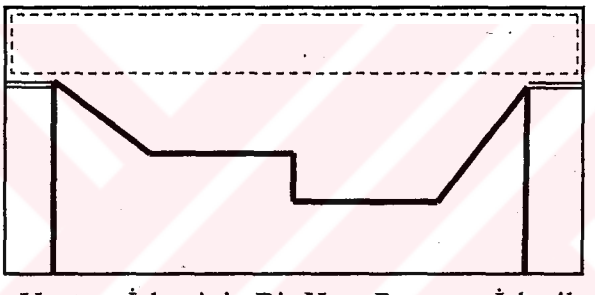


Şekil 12.2 Yüzey Yoğunluğunun Azaltlmasına Ait Akış Diyagram

13. YÜZEYİN UZATILMASI

13.1. Bir Yüzeyin Uzatılması

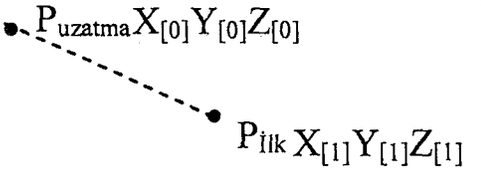
Düzensiz ve büyük çaplı talaş kaldırma işlemleri için, talaş kaldırılacak parçaların üzerinden büyük çapta, dikdörtgensel şekilde bir talaş kaldırma işlemi yapılmaktadır. Ham parçanın işlenilen şekle en yakın hale getirilmesi amacı ile kullanılmaktadır. Bu işlem için yüzeyin uç noktalarına ilave edilerek dikdörtgensel şekilde talaş kaldırma işlemi kolayca yapılabilmektedir. Aşağıdaki şekilde bu uzatma işlemine bir örnek görülmektedir (Şekil 13.1). Bu şekilde kesikli çizgi ile görünen kısım ham parça üzerinde gereksiz olan kısımdır ve bu kısım uzatma işlemi ile yüzey genişletildikten sonra bir baştan diğer başa kadar alınarak, takımın sonraki hareketlerinin kolayca yapılabilmesi sağlanmıştır. Koyu kısım istenilen nihai hedefi ve çift çizgi ile gösterilen kısım ise uzatılan mesafeyi göstermektedir.



Şekil 13.1 Uzatma İşleminin Bir Ham Parçanın İşlenilmesindeki Rolü

13.2. Matematiksel İfade

Bu işlem için kullanıcıdan iki adet bilgi istenmektedir. Bu bilgiler ilk ve son noktalardaki X ekseni yönünde uzatılacak nokta mesafesidir. Bu iki bilgiye göre Y ve Z eksen değerleri hesaplattırılmaktadır. Bu hesaplatma işleminde belirtilen $X_{başlangıç}$ ve $X_{bitiş}$ değerleri direkt koordinatlar olmayıp, uzatılmadan önceki mesafeye ne kadar uzaklıkta bu değerin olacağını göstermektedir. Ayrıca belirtilen bu noktaya göre Y ve Z değerleri hesaplatılırken, uzatılacak nokta düz çizgi şeklide yani doğrusal olarak uzatılmaktadır. Aşağıdaki şekilde ilk satırdan sola doğru genişleme sağlanmıştır ve bu nokta P_{uzatma} olarak adlandırılmıştır (Şekil 13.2). Bu noktanın aynısı bir de son noktaya eklenilmektedir. Böylece iki yönlü olarak uzatma işlemi sağlanabilmektedir.



Şekil 13.2 Bir Parçanın İki Yönlü Uzatılması

Burada verilmesi gereken değer $X_{[1]}$ ile $X_{[0]}$ arasındaki mesafedir. Bu değer R olarak temsil edilirse;

$$X_{[0]} = X_{[1]} + R$$

Aradaki uzaklık doğrusal olarak kabul edildiği için diğer Y ve Z koordinatları aşağıdaki gibi hesaplatılmaktadır. Ancak bunun için öncelikle bu doğrusal çizgiye göre $Y_{[0]}$ ve $Y_{[1]}$ ile $Z_{[0]}$ ve $Z_{[1]}$ noktaları arasındaki eğimin hesaplanması gerekir. Bu eğimler MY ve MZ olarak kabul edilirse;

$$MY = \frac{Y[1] - Y[0]}{X[1] - X[0]} \qquad MZ = \frac{Z[1] - Z[0]}{X[1] - X[0]}$$

Sonra Y ve Z eksenleri ile kesişimin sağlanması için aşağıdaki hesaplamanın yapılması gerekir;

$$BY = Y_{[0]} - (X_{[0]} - MY) \qquad BZ = Z_{[0]} - (X_{[0]} - MZ)$$

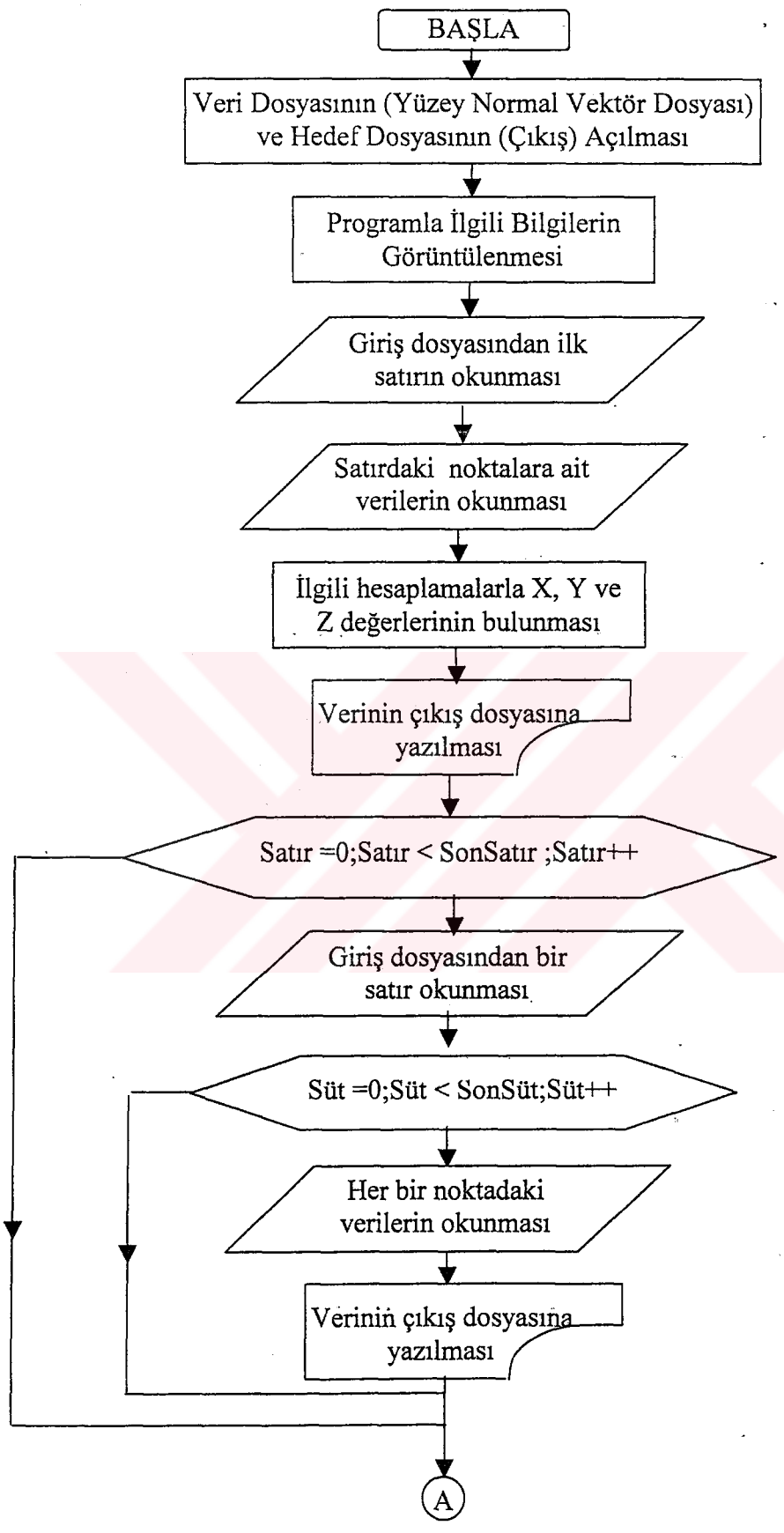
Uzatılmış noktadaki Y ve Z koordinat değerleri şu şekilde hesaplanabilir;

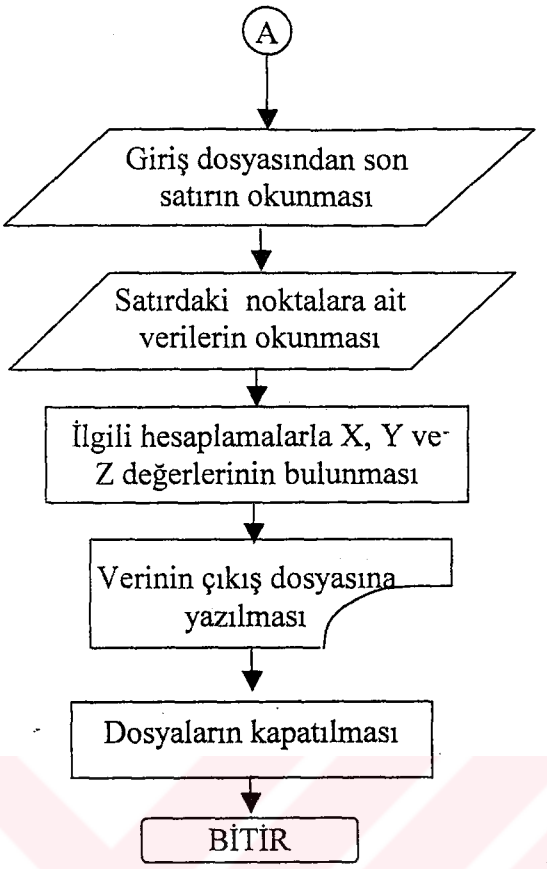
$$Y = BY + R * MY \qquad Z = BZ + R * MZ$$

13.3. Akış Şeması

EKLE.C isimli program yüzeyi biri başlangıç ve diğeri bitiş noktası olmak üzere iki noktadan genişletmektedir. Burada yüzey veri dosyası giriş dosyası olarak kullanılmaktadır. Programda ilk olarak giriş ve çıkış dosyaları açılmaktadır. Kullanıcıdan istenilecek veri hem başlangıç hem de bitiş noktasında X yönündeki mesafe olacaktır. Bu veriye göre başlangıç ve bitiş noktaları için ayrı ayrı Y ve Z eksen değerleri hesaplattırılarak çıkış dosyasına yazılmaktadır. Sonuçta kaç sütun olursa olsun çıkış dosyası giriş dosyasından iki sütun fazla olarak sonuçlandırılmış olacaktır. İlk ve son sütun haricinde aradaki sütunlar ve satırlar giriş dosyasının aynısıdır. Yani sadece ilk ve son satıra işlem yapılmakta aradaki satırlara dokunulmadan aynen çıkış dosyasına yazdırılmaktadır.

EKLE.C isimli program C programlama dilinde yazılmış olup, programa ait kodlar EK-I'da verilmiştir.


BAŞLA
Veri Dosyasının (Yüzey Normal Vektör Dosyası) ve Hedef Dosyasının (Çıkış) Açılması
Programla İlgili Bilgilerin Görüntülenmesi
Giriş dosyasından ilk satırın okunması
Satırdaki noktalara ait verilerin okunması
İlgili hesaplamalarla X, Y ve Z değerlerinin bulunması
Verinin çıkış dosyasına yazılması
Satır =0;Satır < SonSatır ;Satır++
Giriş dosyasından bir satır okunması
Süt =0;Süt < SonSüt;Süt++
Her bir noktadaki verilerin okunması
Verinin çıkış dosyasına yazılması
A



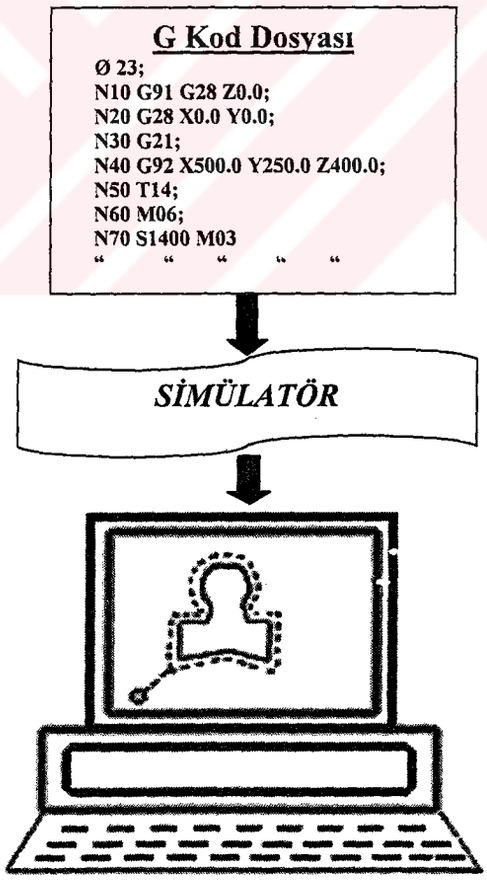
Şekil 13.3 Yüzeyin Uzatılmasına Ait Akış Diyagram

14. SİMULASYON PROGRAMI

14.1. Simülasyon Programı Hakkında

Bu program CNC parçalarının işlenilmesini test etmek amacı ile kullanılmaktadır. Normalde bir parça şeklinin CNC üzerinde doğrudan test edilmesi hem pahalıya mal olabilmekte, hem de tehlikeli sonuçlar doğurabilmektedir. Ancak bu işlem bilgisayar ortamında rahatlıkla denetlenilebilmektedir. CNC tezgahının pahalı olması ve verilebilecek zarar maliyetinin çok fazla olabileceği göz önüne alınırsa, CNC üzerinde bir parçayı işlemek yerine bilgisayar ortamında işlemek çok daha mantıklı olmaktadır.

Simülasyon programı, G kodlarını kullanarak parça şeklini oluşturan ***takım hareketlerinin*** bilgisayar ekranından izlenebilmesini sağlamaktadır. Böylece maliyet olmaksızın istenildiği kadar tekrarlanarak parça şekli görüntülenebilmektedir. Aşağıda bir parçanın simülasyonu ile ilgili simülasyon örneği görülmektedir (Şekil 14.1).



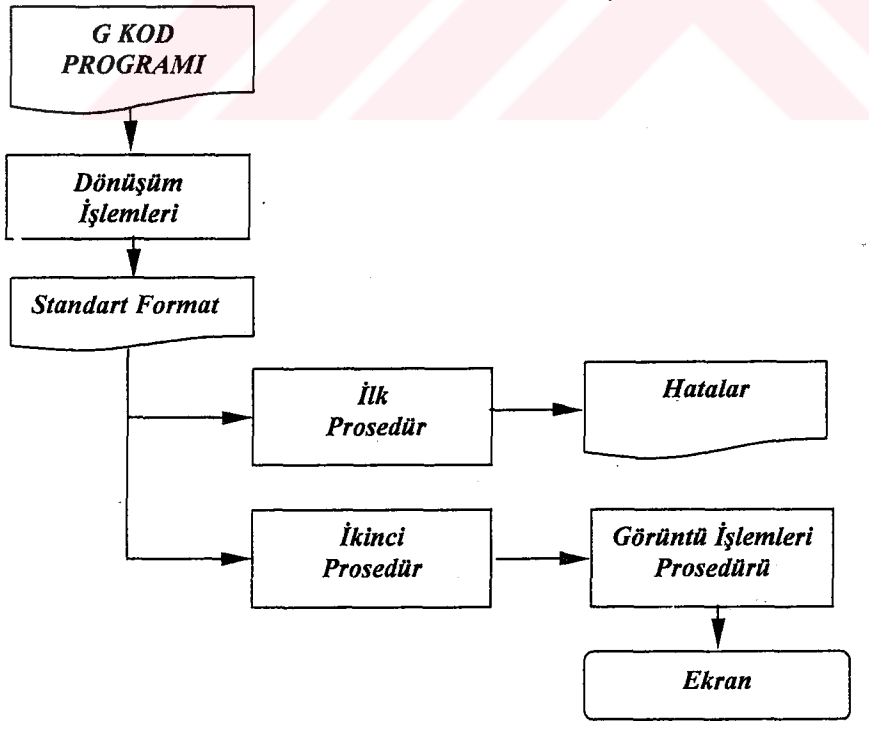
Şekil 14.1 Bir Simülatör Programının Temel İşleyiş Yapısı

Ancak simülasyon programı sadece G kod veri dosyasındaki bilgilere göre işlem yapmaktadır. Yani program içerisinden döndürme, taşıma gibi işlemleri doğrudan gerçekleştirmek mümkün olmamaktadır. Bunun yerine daha önce adı geçen programlar kullanılarak (TASI.C, DONDUR.C vb.) bu işlemler gerçekleştirildikten sonra, G kodlarına dönüşümü sağlayan GKODYAP.C isimli programla yeniden elde edilecek G kodları bu simülasyon programı içerisinden çağrılarak istenilen işlemlerin yapmış olduğu değişiklikler gözlemlenebilmektedir. Yani simülasyon programı sadece verilen G kodlarının temsil ettiği takım hareketlerini izleyebilmemizi sağlamaktadır.

14.2. Simülasyon Programı Akış Diyagramı

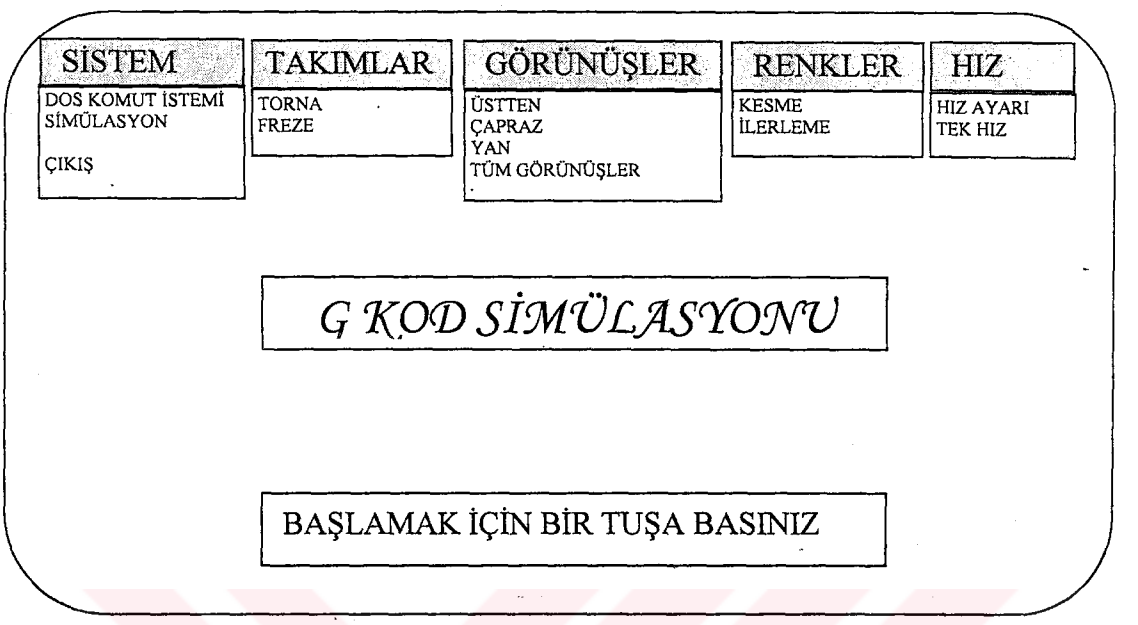
Bir CNC simülasyon programının amacı, tüm G kod satırlarının doğruluğunu kontrol edecek şekilde bir hata kontrol taraması yapmaktır. Bu amaçla Turbo C programlama dilinde grafik ve mouse destekli bir simülasyon programı hazırlanmıştır. SIMULASYON.C isimli program EK-İ'de verilmiştir.

Aşağıda simülasyon test programına ait akış diyagramı görülmektedir (Şekil 14.2). Bu akış diyagram üç ana bölümden oluşmuştur. Dönüşüm işlemleri, G kod işlemleri, görüntüleme işlemleri.



Şekil 14.2 Simülasyon Test Programının Akış Diyagramı

Program menüler tarafından kontrol edilmektedir. Beş temel menü bulunmaktadır. Bu menüler aşağıdaki şekilde görüldüğü gibi alt menülere açılmaktadır (Şekil 14.3).



Şekil 14.3 Simülasyon Programının Ana (Başlangıç) Menüsü

Program, kullanıcının dosya adını girmek suretiyle istenilen G kod dosyasına müdahale edebilmesini ve hataları kontrol ederek G kod dosyasının düzeltilebilmesini sağlamaktadır. Aynı zamanda takım ve takım yolları monitör üzerinden takip edilebilmektedir. Monitör üzerinden takip etme işleminde de belli bir takım ayarlamalar yapılabilmektedir. Çoklu görünüş(üstten, çapraz ve yan), keserken veya hareket esnasında farklı renk seçimi, simülasyonun hız ayarı gibi bir takım seçenekler menülerden ayarlanabilmektedir. Yukarıda bu işlemleri yapabilmemizi sağlayan menülerin bağlı bulunduğu ana menü görülmektedir (Şekil 14.3). Ayrıca takım hareket ederken hareketi yönlendiren G kod satırına ait bilgiler ekranın bir parçasında görünmekte ve o anki takım hareketi kontrol edilebilmektedir. Eğer satırda hata varsa hata mesajları ekran üzerinde gösterilerek veya bir hata dosyasına yazdırılarak kullanıcının hatayı düzeltmesi sağlanmaktadır.

Şekil 14.2'de verilen simülasyon programının akış diyagramı farklı kısımlardan oluşmaktadır. Bu bölümlerin her biri aşağıdaki işlevleri yerine getirmektedir.

14.2.1. Dönüşüm İşlemleri

Dönüşüm işlemleri ile ilgili prosedür içerisinde ilk olarak G kod dosyası açılmaktadır. Sonra G kod dosyasındaki satırların işlenebilecek şekilde tekli satırlara dönüşümü sağlanmaktadır. Bu satırlar anlamlı kodlar yani gereksiz bilgilerden arındırılmış olarak ifade edilebilir. Örnek olarak program başlığı her G kod dosyasında bulunmak zorundadır ve tüm CNC tezgahları bu program numarası içeren başlıkla işleme başlamaktadır. Ancak simülasyon programı içerisinde program başlığı takım hareketlerinin gösterimi sırasında bir anlam kazanmamaktadır. Bu açılan G kod dosyasında yazılmayacağı anlamına gelmemektedir. G kod dosyasında mutlaka program numarası bulunmak zorundadır. Olmazsa program hata vererek, hata GKOD.HAT dosyasına yazılmaktadır. Ancak takım hareketi için bu başlığa gerek duyulmadığı için dönüşüm işlemleri içeren prosedürde bu bilgi iptal edilmektedir. Ayrıca bitişik olarak yazılan G, M, S ve T komutları birer boşlukla ayrılmaktadır. Yani T14 şeklindeki gibi bir bilgi Tø14 şeklinde arada bir boşluk olacak şekilde ayrılmaktadır. Aşağıdaki tablo 2.1'de standart ve dönüşüm prosedüründen elde edilmiş G kod dosyasına örnek görülmektedir.

Standart G Kod Dosyası	Dönüşmüş G Kod Dosyası
Ø23;	
N10 G91 G28 Z0.0;	
N20 G28 X0.0 Y0.0;	
N30 G21;	G 21
N40 G92 X500.0 Y250.0 Z400.0;	G 92 X 500.0 Y 250.0 Z 400.0
N50 T14;	T 14
N60 M06;	M 06
N70 S1400 M03;	S 1400
	M 03
N80 G90 G43 G00 Z100.0 H14;	G 00 Z 100.0
N90 G00 X-170.0 Y-70.0;	G 00 X -170.0 Y -70.0
N100 G01 Z0.0 F500.0;	G 01 Z 0.0
	F500.0
N110 G01 X170.0 F1000.0;	G 01 X 170.0
	F 1000.0
N120 Y-20.0;	G 01 Y -20.0
N130 X-170.0;	G 01 X -170.0
N140 Y80.0;	G 01 Y 80.0
N150 X170.0;	G 01 X 170.0
N160 Y80.0;	G 01 Y 80.0

N170 X-170.0;	G-01 X -170.0
N180 G91 G28 Z0.0;	
N190 G28 X0.0 Y0.0;	
N1170 G40;	
N1180 G49;	
N1190M30;	M30
%	

Tablo 2.1 Standart ve Dönüştürülmüş G Kod Dosya Örneği

14.2.2. Hata İşlemleri

Simülasyon programında ikinci aşama olarak dönüştürülmüş G kod dosyası üzerinde yapılan hata denetimi yer almaktadır. Dönüştürülmüş G kod dosyası satır satır okunmakta ve hata kontrolü de satır satır yapılmaktadır. Okunan satırlar içerisindeki bilgiler string yani karaktersel bilgi olarak bulunmaktadır. Aynı zamanda dönüşüm işlemine uğradığı için tüm G komutları ile sayısal değerler arasında bir boşluk kullanılmaktadır. Bu boşluklardan yararlanılarak kullanılan G komutunun kurallara uygun olup olmadığı kontrol edilmektedir. Bu kontrollere örnek olarak G komutunun alabileceği maksimum ve minimum değer aralığında bir değere sahip olup olmadığı, fonksiyonel olarak kullanımın doğru olup olmadığı kontrol edilmektedir. Örnek olarak bir 'G' karakterinden sonra 00, 01, 02, 03, 17, 18, 19, 20, 21, 50, 90 ve 91 komutları kullanılabilir komutlardır ve bu karakterden sonra bu değerler kontrol edilmektedir. Programın eksikliği olarak tüm G komutları kontrol edilememektedir. 'M' karakterinden sonra ise 01, 03, 04, 05, 30 sayılarının gelip gelmediği kontrol edilmektedir. Aynı zamanda komutların kullanımının yanı sıra bir G kod dosyasında kullanılması mutlaka zorunlu olan mil dönüş hızının belirtilmesi, mil dönüşünün başlatılması ve durdurulması, takım seçimi, ilerleme hızının seçimi gibi kontrollerde yapılmaktadır. Bu hatalar GKOD.HAT dosyasına yazılmaktadır. Standart olarak aşağıdaki şartlar öncelikli olarak taranmaktadır;

- Mil hızı, takım ve ilerleme hızının seçimi,
- Mil dönüşünün başlatılması ve durdurulması,
- G92 ve G00 komutlarının kullanım yeri,
- Kesim işleminin başlangıç yeri,
- Program sonunun yeri.

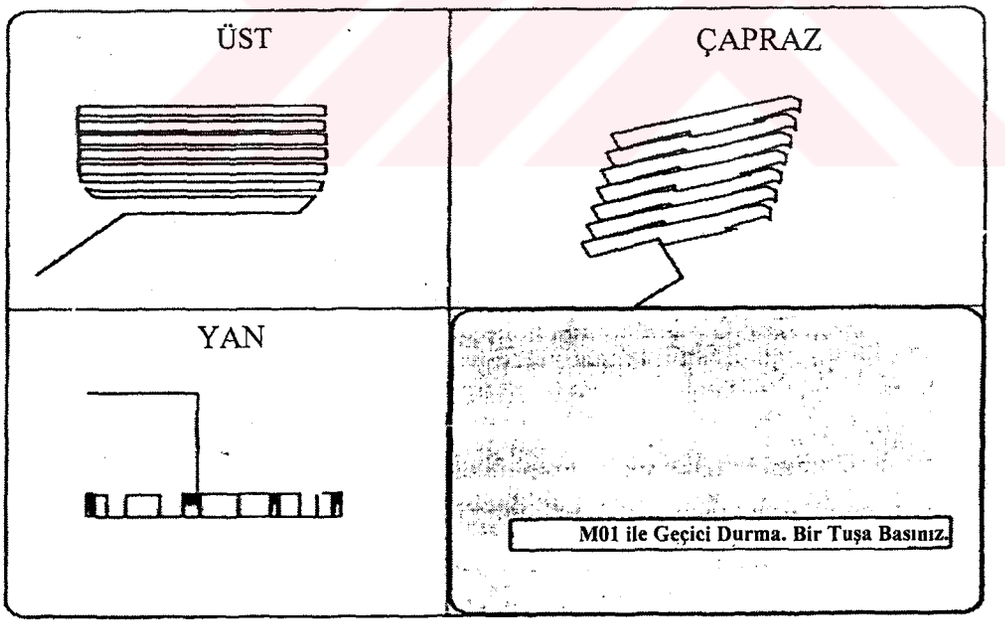
Bu tarama işlemleri yapıldıktan sonra bir hata ile karşılaşılmamışsa ikinci hata tarama işlemi olan, G kodları ile ilgili hata tarama işlemlerine geçilmektedir. Burada yine hata bulunursa hatalar yine GKOD.HAT dosyasına yazılmaktadır. Örnek olarak "Kesme işlemine başlamadan önce mil dönüşünü başlatmadınız" şeklinde kullanıcının hatalarını tespit edecek şekilde hatalar kaydedilmektedir. Aşağıdaki Tablo 2.2'de tipik hata mesajlarının listesi görülmektedir.

1) Sınır dışı G kod sayı değeri
2) Sınır dışı X koordinat değeri
3) Sınır dışı Y koordinat değeri
4) Sınır dışı Z koordinat değeri
5) Sınır dışı S kod sayı değeri
6) Sınır dışı I kod sayı değeri
7) Sınır dışı J kod sayı değeri
8) Sınır dışı K kod sayı değeri
9) Sınır dışı R yarıçap değeri
10) Sınır dışı F ilerleme hızı değeri
11) Sınır dışı T kod değeri
12) Sınır dışı M kod değeri
13) Mil dönüş hızı ayarlanmadı
14) Takım seçilmedi
15) İlerleme hızı girilmedi
16) Mil dönüşü başlatılmadı
17) Mil dönüş hızı ayarlanmadan mil dönüşü başlatıldı
18) Takım seçilmeden mil dönüşü başlatıldı
19) G92/50 seçiminde mutlak sıfır noktası bulunamadı
20) Mil dönüşü başlatılmadan önce G00'dan farklı bir G kodu ile karşılaşıldı
21) Mil dönüşü başlatılmadan önce program sonu ile karşılaşıldı
22) G00'dan başka G kodu olmadan işlem yapılamaz
23) Mil dönüşü durdurulmadı
24) Program sonu ile karşılaşılamadı
25) Teybi yeniden sarma işareti olan % ile karşılaşılamadı
26) Program torna tezgahı için ayarlanmış, Y eksen değeri 0 olmalı
27) G92/50'den önce G 0/1/2/3 kodları ile karşılaşıldı

Tablo 2.2 G Kod Hata Mesajları Örnekleri

14.2.3. Görüntü İşlemleri

Eğer G kod dosyası bir önceki kısım olan hata işlemleri isimli prosedürden hatasız bir şekilde geçerse görüntü işlemlerini içeren prosedür kullanılarak takım yolunun ekran üzerinde görüntülenmesi sağlanabilir. Ekran dört eşit ölçüde pencereye bölünmüştür. Bunlardan üçü ekranda takım yolunun üst, yan ve çapraz görünümleri için kullanılmaktadır. Dördüncü pencere ise o anki görüntülenmekte olan takım yoluna ait takım numarası, mil dönüş hızı, ilerleme hızı, X, Y ve Z koordinat değerlerini göstermektedir. Aşağıdaki şekilde simülasyon programının görünümü ile ilgili ekran örneği görülmektedir (Şekil 14.4). "Görünüşler" menüsünden takıma ait istenilen görünüş şekli tam ekran olacak şekilde seçilebilir. Üstten ve yandan görünümler tam ekran olacak şekilde ölçeklendirilmektedir. Takım yolu ise iki farklı renkte gösterilmektedir. Birincisi takımın keserek ilerlediği(G01) durumlar, ikincisi ise takım kesme olmaksızın hareket ettiği(G00) durumlar. Böylece takımın nerde boşta ilerlediği, nerede kesim işlemi yaptığı anlaşılabilmektedir. Bu hareket esnasında kullanılacak renkler ise "Renkler" menüsü kullanılarak yapılabilmektedir.



Şekil 14.4 Takım Yolunun Gösterim Ekranı

Aynı zamanda kullanıcı takım yolunun simülasyonunun gösterim hızını da seçebilmektedir. "Hız" menüsü içerisinde iki adet seçenek bulunmaktadır. Eğer tek hız seçilirse kullanıcı ESC veya SPACE BAR tuşlarına bastıkça bir sonraki G kod satırına geçilmektedir. Bunun en büyük avantajı kullanıcı takım hareketlerini adım adım gözlemleyebilmektedir. İkinci seçenekte ise kullanıcı 0=hızlı, 999=yavaş olacak şekilde arada bir sayı değeri girerek hız ayarını yapmaktadır. Böylece simülasyon belirtilen hızda olacaktır. Kullanıcının herhangi bir tuşa basmasına gerek kalmayacaktır.

Ana program üzerinde yer alan diğer bir menü ise "Görünüşler" menüsüdür. Standart olarak ilk açıldığında tüm görünüşler aktif olacak şekilde program karşımıza gelmektedir. Bu görünüşte, ekranda yukarıdaki şekilde görüldüğü gibi dört adet pencereden üç tanesinde aynı şekil, farklı konumlardan görüntülenmektedir (Şekil 14.4). Bunlar üst, çapraz ve yan görünüşlerdir. Bu menü ile takım hareketlerine ait istenilen görünüş şekli tam ekran olacak şekilde ekrana yansıtılmaktadır. Yani bir ölçekleme işlemi ile takım hareketlerinin boyutu büyütülmektedir. Bu üstten, yandan veya çaprazdan olabilir. İstenildiği takdirde tüm görünüşler seçeneği seçilerek programın tekrar üç görünüş şeklini gösterecek şekilde ayarlanması sağlanabilir.

Bu işlem için grafik ortamında bulunulduğu için noktadan noktaya çizgi çizimi yapılmaktadır. LINE komutu ile çizim işlemi gerçekleştirilmektedir. Kullanılan G00 komutu kesme olmaksızın, G01 komutu ise kesme olarak yapılan çizimi göstermektedir. Bu seçeneğe göre çizgilerin renkleri ayarlanmaktadır. G02 ve G03 komutlarında ise ARC komutu ile dairesel çizim işlemi gerçekleştirilmektedir. Böylece takıma ait hareketlerin simülasyonu tamamlanmaktadır. Scale denilen ölçeklendirme işlemi ise görünüşler arası geçişte kullanılmaktadır. Yani tam ekran olarak üst, çapraz ve yan görünüş seçildiği anda ekran önce silinmekte, seçime göre çizim komutları belirli oranda büyültülen verilere göre yeniden uygulanarak tam ekran olması sağlanmaktadır. Tüm görünüş seçildiği anda büyütme oranı iptal edilerek çizimlerin eski boyutuna gelmesi sağlanmaktadır.

14.3. Ölçeklendirme (Scaling) İşlemi

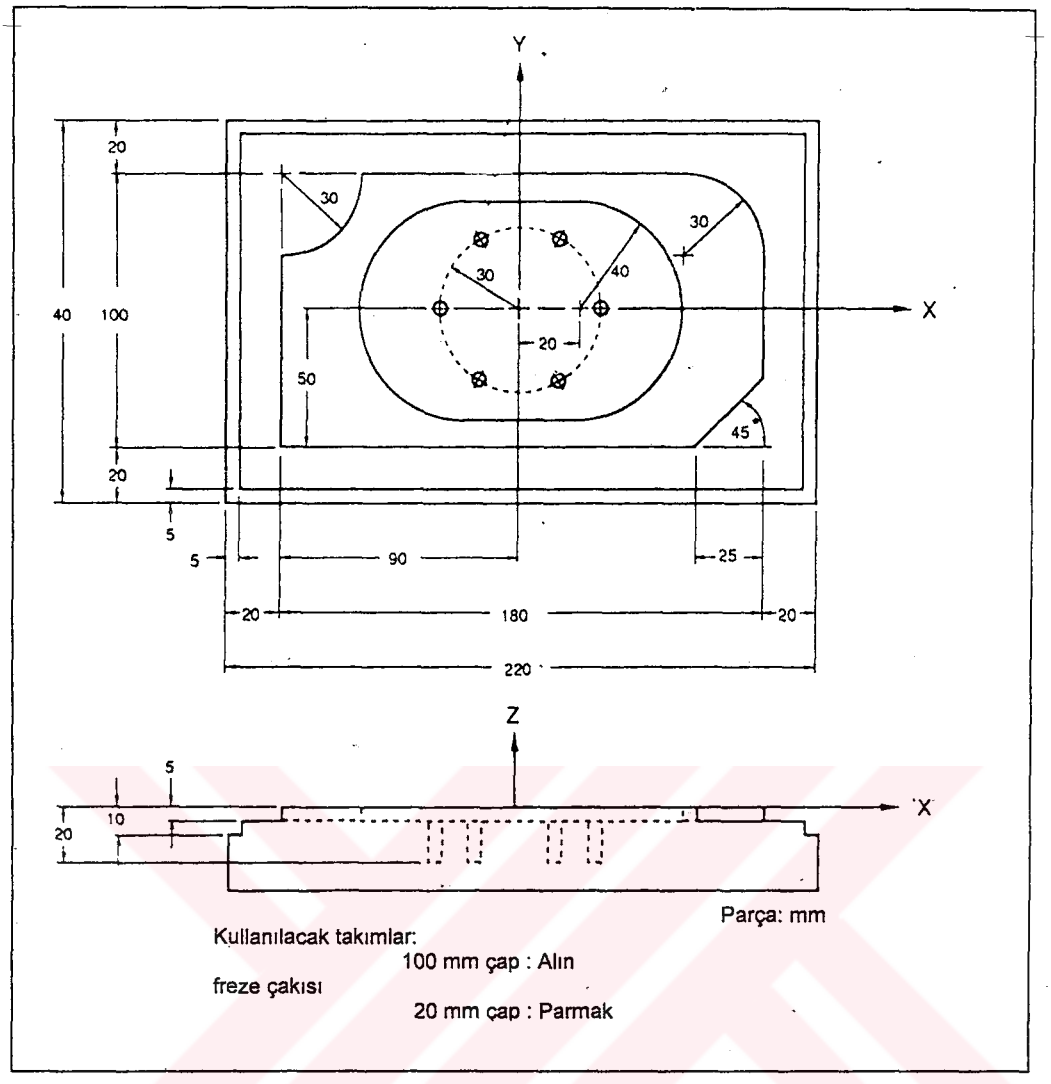
Simülasyon programları içerisinde ölçeklendirme işlemi gerçekleştirilmektedir. Bu ölçeklendirme işlemi program içerisinde görünüşler menüsünde bir seçenek seçildiği zaman aktif hale gelmektedir. Tüm görünüşler içerisindeki yan, üst ve çapraz görünüşlerden herhangi bir tanesi seçildiği anda tam ekran olarak büyütülerek karşımıza gelmektedir. Tersi durum olduğu zaman yani tüm görünüşlere geçildiği anda ise çizimler küçültülerek ekrana aktarılmaktadır. Bu işlem için yine bir matris kullanılmaktadır. Aşağıda üç boyutlu şekiller için taşıma, döndürme ve ölçeklendirme işlemlerini özetleyerek gösterebileceğimiz bir matris görülmektedir. Buradaki S parametresi ölçeklendirme işlemini gerçekleştirmektedir. Döndürme, taşıma ve ölçeklendirme matrisi [R] olarak kabul edilirse bu matris aşağıdaki gibi gösterilebilir;

$$[R] = \begin{bmatrix} a & b & c & T_x \\ e & f & g & T_y \\ h & i & j & T_z \\ O & O & O & S \end{bmatrix}$$

Buradaki a, b, ... i, j arasındaki değişkenler özel döndürme değerlerini, T_x, T_y, T_z özel taşıma değerlerini ve S değeri ise şekil büyüklüğünün ekrandaki görünümünün ölçeklendirme miktarını göstermektedir. Bu matrise 4 x 4 transform matrisi denilmektedir.

14.4. Simülasyon Programının Çalışması İle İlgili Bir Örnek

Simülasyon programının çalışması G kod sistemine göre olmaktadır. Bu G kod sistemine göre kesici takım hareketleri sabit bir hızla veya adım adım izlemek amacı ile bir tuşa basılınca bir G kod satırını işletecek şekilde izlenebilmektedir. Böylece hangi satırda ne işlem yapıldığı anlaşılabilmektedir. Bu bilgiler dördüncü ekran olan bilgi ekranından takip edilebilmektedir. Simülasyon programının çalışmasını açıklamak amacı ile daha önce CNC tezgahında işlenmiş olan şekil 14.5'de görülen parçaya ait takım hareketlerinin, simülasyon programı içerisinde izlediği yollar ekran çıktısı şeklinde verilmiştir.



Şekil 14.5 Simülasyon Programı İçerisinde Tanımlanacak Parça Şekli

Bu parça şeklini oluşturabilmek için bir CNC tezgahında veya simülasyon programı içerisinde takım hareketlerini izleyebilmemiz için gerekli olan G kod satırları ise tablo 2.3'de görülmektedir.

Başlama Durumuna Getirmek Ø 23; N10 G91 G28 Z0.0; N20 G28 X0.0 Y0.0; N30 G21; N40 G92 X500.0 Y250.0 z400.0; N50 T14; N60 M06; N70 S1400 M03;	N460 G01 Z-5.0 F1000.0; N470 G42 G01 X-90.0 Y-50.0 D25 F1000.0; N480 X65.0; N490 X90.0 Y-25.0; N500 Y20.0; N510 G03 X60.0 Y50.0 R30.0; N520 G01 X-60.0; N530 G02 X-90.0 Y20.0 R30.0; N540 G01 Y-50.0; N550 G01 G40 X-110.0 Y-90.0; N560 G01 Z10.0; N570 G91 G28 Z0.0; N580 G20 X0.0 Y0.0;

Yüzey Frezeleme

N80 G90 G43 G00 Z100.0 H14;
N90 G00 X-170.0 Y-70.0;
N100 G01 Z0.0 F500.0;
N110 G01 X170.0 F1000.0;
N120 Y-20.0;
N130 X-170.0;
N140 Y80.0;
N150 X170.0;
N160 Y80.0;
N170 X-170.0;
N180 G92 G28 Z0.0;
N190 G28 X0.0 Y0.0;
N200 M05;

Dış Kenarların Kesilmesi

N210 T15;
N220 M06;
N230 G49;
N240 S1400 M03;
N250 G90 G43 G00 Z100.0 H15;
N260 G00 X-130.0 Y-70.0;
N270 G01 Z-10.0 F500.0;
N280 G42 G01 X-110.0 Y-70.0 D25 F1000.0;
N290 G01 X110.0 F1000.0;
N300 Y70.0;
N310 X-110.0;
N320 Y-70.0;
N330 G01 G40 Y-100.0;
N340 Z10.0;

İç Profilin Oyulması

N350 G00 X0.0 Y0.0;
N360 G01 Z-5.0 F500.0;
N370 G42 G01 Y-40.0 D25 F1000.0;
N380 X20.0;
N390 G03 X20.0 Y40.0 R40.0;
N400 G01 X-20.0;
N410 G03 X-20.0 Y-40.0 R40.0;
N420 G01 X10.0;
N430 G01 G40 X20.0 Y20.0;
N440 G01 Z10.0;

Kenarların Şekle Uygun Kesilmesi

N450 G00 X-140.0 Y-70.0;

Delik Açma

N590 T16;
N600 M06;
N610 S1400 M03;
N620 G49;
N630 G90 G43 G00 Z100.0 H16;
N640 G01 X20.0 Y0.0 F1000.0;
N650 G01 Z-20.0 F500.0;
N660 G04 X1.0;
N670 G01 Z0.0;
N680 G00 Y17.32 X10.0;
N690 G01 Z-20.0 F500.0;
N700 G04 X1.0;
N710 G01 Z0.0;
N720 G01 X-10.0;
N730 G02 Z-20.0 F500.0;
N740 G04 X1.0;
N750 G01 Z0.0;
N760 G01 X-20.0 Y0.0;
N770 G01 Z-20.0 F500.0;
N780 G04 X1.0;
N790 G01 Z0.0;
N800 X-10.0 Y-17.32;
N810 G01 Z-20.0 F500.0;
N820 G04 X1.0;
N830 G01 Z0.0;
N840 G01 X10.0;
N850 G01 Z-20.0 F500.0;
N860 G04 X1.0;
N870 G01 Z0.0;

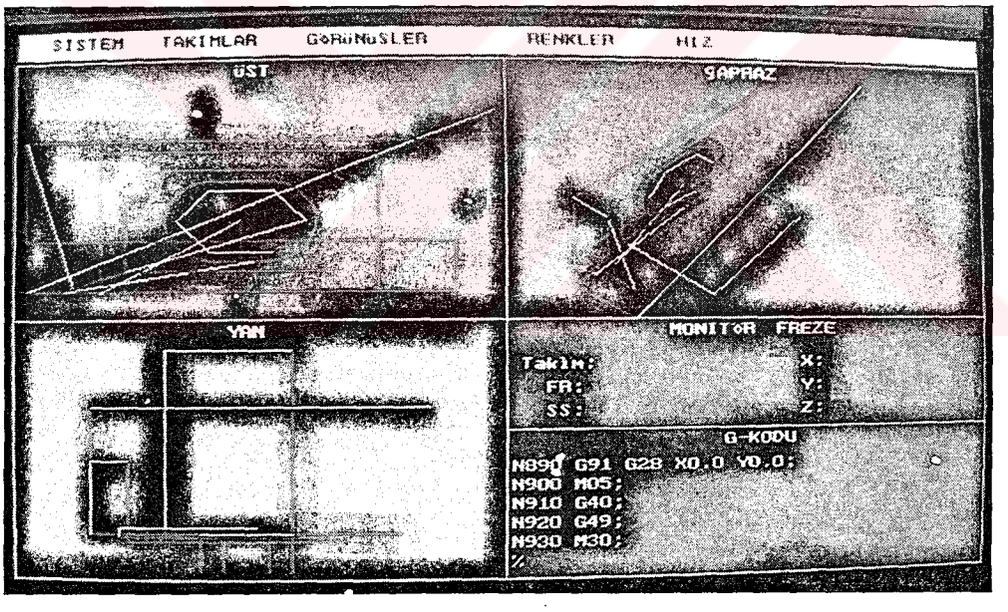
Programın Bitirilmesi

N880 G91 G28 Z0.0;
N890 G91 G28 X0.0 Y0.0;
N900 M05;
N910 G40;
N920 G49;
N930 M30;
%

Tablo 2.3 Şekil 14.5'deki Parça Şekline Ait G Kod Satırları

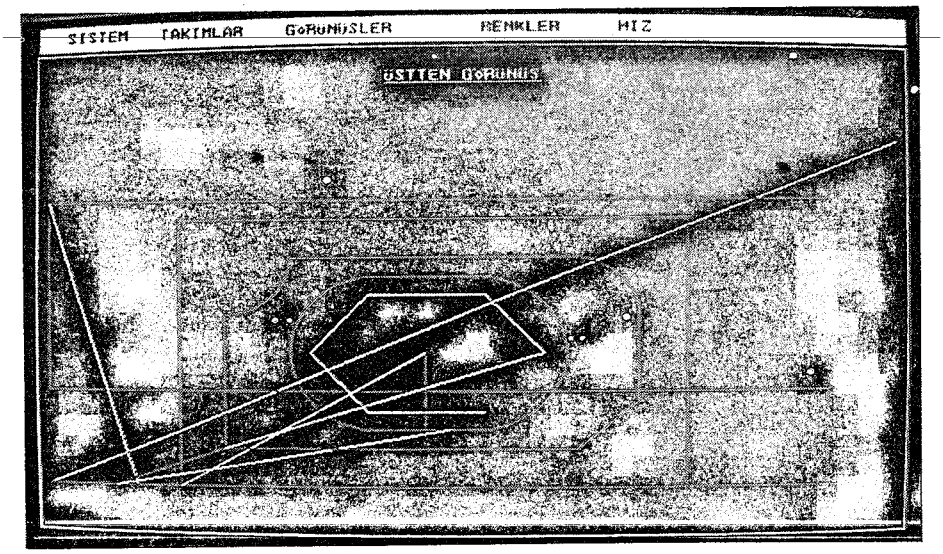
Bu G kod satırları bir text dosya şeklinde yazılarak, bilgisayar hafızasına bir isim verilerek kaydedilir. Kaydedilen bu dosya simülasyon programı içerisinden çağrılacaktır. Bu çağırma işlemi sistem menüsü içerisindeki simülasyon seçeneği ile yapılmaktadır. Dikkat edilmesi gereken bir nokta bu bir freze tezgah işlemi olduğu

ve program menülerinden olan takımlar menüsündeki freze seçeneği standart seçili olduğu için bu seçeneği değiştirmeye gerek kalmamaktadır. Bir torna tezgah işlemi ile karşılaşıldığında ise, takımlar menüsündeki torna seçeneği seçilmelidir. Sistem menüsündeki simülasyon seçeneği seçildiği zaman aktif dizin olan yani simülasyon.exe programının bulunduğu dizin içerisinde bulunmak zorunda olan text dosyasının isminin girilmesin isteyen bir komut ekrana gelir. Örnek olarak yukarıdaki kod satırları text programlarından herhangi bir tanesi ile yazıldıktan sonra KAYNAK.TXT olarak kaydedilebilir. Bu durumda simülasyon seçeneği seçildiği zaman aktif dizinde bulunan bu dosya olan, kaynak.txt ismini komut olarak girip enter tuşuna basılır. Bu durumda aktif olan tüm görünüşler seçeneğinde adım adım üst, yan ve çapraz olarak üç şekilde kesici takım hareketleri karşımıza gelmektedir. Bilgi ekranı olan dördüncü ekranda ise hangi satırın, takımı hangi şekilde yönlendirdiği izlenebilir. Aşağıdaki şekilde tüm görünüş ekranında, tablo 2.3'deki G kod satırlarının uygulanmış hali görülmektedir (Şekil 14.6).

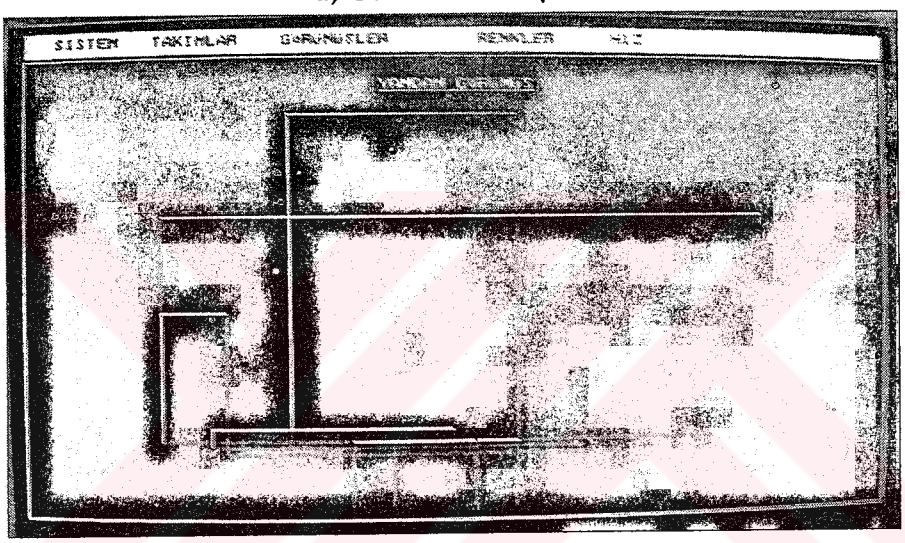


Şekil 14.6 G Kodlarının Tüm Görünüşler Seçeneğindeki Görünümü

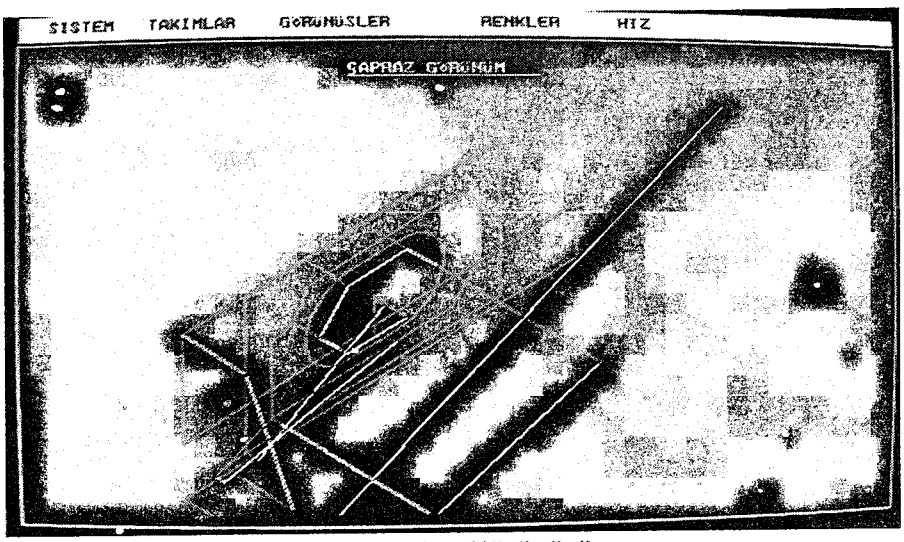
İstenilirse görünüşler seçeneğinde bulunan üst, çapraz ve yan seçenekleri ile tüm görünüşlerde küçük olarak görünen takım hareketleri tam ekran olarak görülebilir. Burada ölçeklendirme yani scaling işlemi kullanılmaktadır. Aşağıdaki şekilde aynı G kod satırlarının görünüşler menüsünden seçilerek elde edilmiş üst, yan ve çapraz olarak görünüşleri görülmektedir (Şekil 14.7). İstenilirse yine bu menüdeki tüm görünüşler seçeneği seçilebilir.



a) Üstten Görünüşü



b) Yandan Görünüşü



c) Çaprazdan Görünüşü

Şekil 14.7 Şekil 14.6'daki Parça Şekline Ait Diğer Görünüşler

15. SONUÇ VE TARTIŞMA

Çalışmada günümüz açısından bilgisayarın en önemli kullanım yeri olan imalat ve tasarım alanları içerisinde bilgisayar sistemlerinin nasıl çalıştığı ele alınmıştır. Bu kapsamda, BDT/BDÜ alanlarının alt kolu olan yüzey modelleme içerisinde bilgisayar çizim programlarında yapılabilen, bir parça şeklinin hem tasarlanarak taşınması, döndürülmesi, belirli bir oranda küçültülmesi gibi temel işlemler gerçekleştirilebilmekte, hem de bu tasarıma ait makina işlemleri için gerekli olan G kod sistemi, verilen yüzey bilgilerine göre ürettirilmektedir. Bu amaçla geliştirilen ve C programlama dilinde yazılan programlar, bilgisayar belleğinde bulunan veri dosyası üzerinde işlem yapmaktadır. Her işlem kendi içerisinde çalışmakta ve bellek içerisinde bulunan veri dosyasını girdi olarak kullanmakta, kendi işlemine ait formüllere uygulandıktan sonra çıktı olarak yeniden bir veri dosyası belleğe kaydedilmektedir. Tüm programlara ait işlemin ne olduğu, matematiksel ifade ve akış diyagram konu anlatımları içerisinde verilmiştir. Özellikle BDT/BDÜ paket programları içerisinde yer alan yüzey normal vektör hesabı ve kesici takım yarıçapının dikkate alınarak kesici takım yolunun hatasız olarak tasarımı gerçekleştirilebilmektedir. Ayrıca yine bir program aracılığı ile ürettirilen G kod dosyasının, bir simülasyon programı kullanılarak takımın yaptığı hareketlerin kullanıcı tarafından izlenebilmesi sağlanmaktadır. İstediği takdirde kullanıcı bu G kod dosyasına müdahale ederek takım hareketlerinin değişimini izeleyebilmektedir. Bu ise simülasyon programını bağımsız hale getirebilmektedir. Yani kullanıcı bağımsız G kod dosyalarına ait takım hareketlerinide simülasyon programını kullanarak izleyebilmektedir.

Çizim programları kullanılmaksızın, deneme yanılma yöntemiyle imalat tasarım ve dizaynı şüphesiz hem tehlikeli hemde BSD tezgahların pahalı olduğu göz önüne alınırsa yüksek maliyetlere varan hasarlara neden olabilmektedir. Eğitim açısından değerlendirildiğinde ise maliyetinin yüksek olması nedeniyle eğitim ve öğretim kurumlarına BSD'li torna tezgahlarının verilmesi mümkün olamamaktadır. Ancak bunun yerine maliyeti daha düşük olan bilgisayar ile simüle programlara bu işlem yaptırılabilmektedir. Bu nedenle bilgisayar destekli çizim programları tasarım esnasında sıklıkla kullanılmaktadır. Ancak çoğu çizim programının yabancı dille

yazılmış olması, kullanıcının çizim programına tam hakim olmasını engellemektedir. Ayrıca ülke ekonomisi açısından bu çizim programları için yurt dışına büyük miktarda para kaybı söz konusu olmaktadır. Kendimize ait çizim programlarının geliştirilmesi hem ülke ekonomisi açısından kazancın, hemde dil olarak Türkçe kullanıldığı için kullanıcının programa hakim olması nedeniyle verimliliğin artmasını sağlayacaktır.

Gelecek çalışmalar için öneriler:

Çalışmada simülasyon programı haricinde ele alınan programlar sadece matematiksel olarak işlem yapmaktadır. Yani sayısal olarak ifade edilmektedir. Tüm işlemler simülasyon programı içerisinde birleştirilebilir. Yani teorik haricinde profesyonel anlamda bir çizim programı elde edilebilir. Ayrıca BDT/BDÜ çizim programlarında sıkça kullanılan özellik olan Doğrudan Sayısal Denetim (DSD-DNC Direct Numerically Control) alanına ağırlık verilebilir. DSD aracılığı ile çizim programında yapılan tasarıma ait G kodların RS 232 portu aracılığı ile aktarılması sağlanabilir.

Bu çalışmadan yola çıkılarak entegre bir çizim programının tasarlanmasının, karmaşık ve zor olarak düşünülen çizim programlarının geliştirilmesi yolunda önemli bir adım olacağını söyleyebiliriz.

16. KAYNAKLAR

[1] Demirci, F. , Bülbül H. İ. ve Arıcı N., "Bilgi İşleme Giriş", Gazi Kitabevi, Ankara, 1994.

[2] Boz, O., "Freze Tezgahları İçin CAD/CAM Program Geliştirme", Yüksek Lisans Tezi, Ankara, 1998.

[3] Fu, K. S. , Gonzalez, R.C. ve Lee C. S. G., "Robotics (Control, Sensing, Vision and Intelligence)", McGraw-Hill Book Company, Newyork ,1987.

[4] Lipschutz, Seymonur ve DOST S., "Teori ve Problemlerle Lineer Cebir", Güven Kitabevi Yayınları, Ankara, 1978.

[5] Mckissick, M. L., "Computer Aided Drafting and Design", Prentice Hall Inc. , New Jersey, 1987.

[6] Vickers, G. W. , Ly, M. ve Oetter, R.G. , "Numerically Controlled Machine Tools", Ellis Horwood Limited , London , 1990.

[7] Yıldız, Alper. ,"Bilgisayar Destekli Grafik Modelleme ve Yüzey Modelleme Esasları".Yüksek Lisans Tezi, İstanbul, 1994.

[8] Ünsaçar, F., Çoklar A. N., "CNC Tezgahların Programlanması", Nobel Kitabevi, Ankara, 2003.

[9] http://graphics.cs.ucdavis.edu/CAGDNotes/

[10] http://hlakin.cmpe.boun.edu.tr/593/gorusimge

[11] http://tide.it.bond.edu.au/inft340/INFT340site/340Tutorials.htm

[12] http://www2.acae.cuhk.edu.hk/~ace7410/main.htm

17. EKLER

EK A

TAŞIMA (TRANSLATION) İŞLEMİ YAPAN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*          TASIMA.C    */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <library.h>

#define    X_COMPONENT    0
#define    Y_COMPONENT    1
#define    Z_COMPONENT    2

void information
    (
    char **FileNames
    );

void translate_surface_file
    (
     char  *SurfaceFile,
     char  *OffsetsFileFile,
     double TransX,
     double TransY,
     double TransZ
     );

main
  (
   int NumberOfParameters,
   char **ParameterStrings
  )
{
   double TransX,
          TransY,
          TransZ;

check_parameters_two(NumberOfParameters,
ParameterStrings);

information ( ParameterStrings );

TransX = get_one_double_value("X EKSENİNDE
D™ND$RME M~KTARI?:");
TransY = get_one_double_value("Y EKSENİNDE
D™ND$RME M~KTARI?:");
TransZ = get_one_double_value("Z EKSENİNDE
D™ND$RME M~KTARI?:");

translate_surface_file( ParameterStrings[1],
                        ParameterStrings[2],
                        TransX, TransY, TransZ);
printf ( "\n\nPROGRAM BAŞSRIYLA
TAMAMLANDI\n" );
return(0);
}
void information
 (
  char **FileNames
 )
{
clrscr();
printf("     *** TAŞIMA  PROGRAMI ***\n");
printf("     YÜZEY DOSYASININ OKUNMASI\n\n");
printf("     KULLANICIDAN BİLEŞENLERİN VE
DEĞERLERİN ALINMASI\n" );
printf("     HER İKİSİNE BİRDEN EKLENİLMESİ\n\n");
printf("     SONUÇLARIN ÇIKIŞ DOSYASINA
YAZILMASI\n\n");
printf("  YÜZEY DOSYASININ ADI= %s\n", FileNames[1]);

printf(" DÖNDÜRÜLMÜŞ DOSYANIN ADI= %s\n",
FileNames[2]);
printf("\n");
}
void translate_surface_file
  (
   char  *SurfaceFile,
   char  *TrnsltedFile,
   double TransX,
   double TransY,
   double TransZ
  )

{

FILE   *fSurfaceFile;
double  SurfaceX,
        SurfaceY,
        SurfaceZ;

int  NLinesSurface,
   NVerticesSurface ;

FILE  *fTrnsltedFile;
double  TrnsltedX,
        TrnsltedY,
        TrnsltedZ;
register int  Line,
              Vertex;

fSurfaceFile = open_input_file(SurfaceFile);
fTrnsltedFile= open_output_file(TrnsltedFile);


fscanf(fSurfaceFile,"%d",&NLinesSurface);
fprintf (fTrnsltedFile, "%d\n", NLinesSurface);

gotoxy(10,17);
printf("%d SATIR İŞLENİLMEKTEDİR\n", NLinesSurface);
gotoxy(10,18);
printf("İŞLENİLEN SATIR  # ");

for( Line=0 ; Line<NLinesSurface ; Line++ )
{
gotoxy(27,18);
printf("%4d", Line+1);

fscanf(fSurfaceFile,"%d",&NVerticesSurface);
fprintf(fTrnsltedFile,"%d\n",NVerticesSurface);

for( Vertex=0 ; Vertex<NVerticesSurface ; Vertex++)
{
fscanf(fSurfaceFile, "%lf %lf %lf", &SurfaceX, &SurfaceY,
&SurfaceZ);
```

```
TrnsltedX = SurfaceX + TransX;
TrnsltedY = SurfaceY + TransY;
TrnsltedZ = SurfaceZ + TransZ;

fprintf(fTrnsltedFile, " %9.6f %9.6f
%9.6f\n",TrnsltedX,TrnsltedY,TrnsltedZ);
   }
  }
}

FILE *open_input_file
            (
            char *Name
            )

{
FILE   *fin;

/* Okuma Hakkı içinde Dosyanın Açılması */
            fin = fopen( Name, "r" );

/* Eğer Programı Açma İŸlemi BaŸarısız ise */
            if( fin == NULL )
            {
                        printf(" DOSYA
AÇILAMIYOR : %s\n", Name);
                        exit (1);
            }
/* Dosya İŸaretçisinin Geri G"nderilmesi */

 return ( fin );
}

FILE *open_output_file
            (
            char *Name
            )
{
FILE   *fout;

/* Dosyanın Yazma Modunda Açılması */

            fout = fopen( Name, "w" ) ;

/* Eğer  Dosya Açma iŸlemi BaŸarızsa */

            if(fout == NULL)
            {
            printf("DOSYA AÇILAMIYOR :
%s\n", Name);
            exit (1);
            }

/* Dosya İŸaretçisinin Geri G"nderilmesi */

 return ( fout );
}

void beep
            (
            void
            )

{
/* Saniyenin Onda biri kadar süreyle hoparl"re
beep sesinin g"nderilmesi*/
sound(1000);
delay(100);
nosound();
}
double get_one_double_value
            (
            char *Question
            )

{
            double Answer;
            int  NfieldsRead;

/* Sorunun Ekranda G"sterilmesi */
printf("%s", Question);

/*Geçerli bir Değer Girilinceye Kadar Veri Girilmesi */
do
{
NfieldsRead=scanf("%lf",&Answer);
/* Girilen Değerin Geçerli olup olmadığının Kontrol Edilmesi*/
if (NfieldsRead !=1)
  {
  beep();
  }
}while (NfieldsRead !=1);

/* İmlecin Yeni Bir Satıra Getirilmesi */
 printf("\n");
 /* Okunan Değerin Geri D"ndürülmesi */
 return (Answer);
}

void check_parameters_two
            (
            int NumberOfParameters,
            char **FileNames
            )

{
 if (NumberOfParameters!=3)
   {
    /* Ekranın Temizlenmesi */
            clrscr();
            printf("            %s PROGRAMINI ÇALIŞTIRMA
HATASI\n",FileNames[0]);
            printf("    GİRİŞ VE ÇIKIŞ DOSYA ADLARINI
DOĞRU OLARAK GİRMEK ZORUNDASINIZ\n");
            printf("    KULLANIM  >>> %s <GİRİŞ
DOSYASI> <ÇIKIŞ DOSYASI>\n", FileNames[0]);
            exit(1);
   }
}
```

EK B

DÖNDÜRME (ROTATION) İŞLEMİ YAPAN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*          DONDUR.C          */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>

#include <library.h>


void information
  (
    char **FileNames
  );

void rotate_surface_file
  (
          char  *InputFile,
          char  *OutputFile,
          double Alpha,
          double Beta,
          double Gamma
  );

main
  (
          int NumberOfParameters,
          char **ParameterStrings
  )
{
          double Alpha,
                 Beta,
                 Gamma;


check_parameters_two( NumberOfParameters,
ParameterStrings );

information( ParameterStrings );

gotoxy(10,10);
Alpha = get_one_double_value( "X EKSENINDE
DÖNÜS AÇISINI GIRINIZ..: " );
Alpha *= (M_PI/180.0);

gotoxy(10,11);
Beta = get_one_double_value( "Y EKSENINDE DÖNÜS
AÇISINI GIRINIZ..: " );
Beta *= (M_PI/180.0);

gotoxy(10,12);
Gamma = get_one_double_value( "Z EKSENINDE
DÖNÜS AÇISINI GIRINIZ..: " );
Gamma *= (M_PI/180.0);

rotate_surface_file(ParameterStrings[1],
ParameterStrings[2], Alpha, Beta, Gamma);
gotoxy(25,22);
printf("PROGRAM BASARIYLA TAMAMLANDI\n");

return(0);
}

void information
   (
          char  **FileNames )
{
clrscr();
/* PROGRAM HAKKINDA BİLGİ VERİLMESİ      */
printf("                **** DÖNDÜRME PROGRAMI
*** \n");
printf("            X VE Y EKSEN DEGERLERININ
AYARLANMASI \n\n");
printf(" GIRIS DOSYA ADI = %s\n", FileNames[1]);
printf(" ÇIKIS DOSYA ADI = %s\n", FileNames[2]);

}
void rotate_surface_file
  (

          char *InputFile,
          char *OutputFile,
          double    Alpha,
          double    Beta,
          double    Gamma
   )
{
   int   Vertex,
             Line;

FILE   *fIn,
          *fOut;

double X,Y,Z;

double RotX,RotY,RotZ;

int NLines,NVertices;

/* GIRIS VE ÇIKIS DOSYALARININ AÇILMASI */

fIn = open_input_file(InputFile);
fOut = open_output_file(OutputFile);

FILE       */
fscanf(fIn, "%d",  &NLines);
fprintf(fOut, "%d\n", NLines);

gotoxy(1,15);
printf("%d SATIR BULUNMAKTADIR\n",NLines);
gotoxy(6,18);
printf("SATIR ISLENIYOR ");

/* LOOP THROUGH EVERY LINE          */
for (Line=0 ; Line<NLines ; Line++)
{

gotoxy(1,18);
printf("%4d.", Line+1);

fscanf(fIn, "%d", &NVertices);
fprintf(fOut,"%d\n", NVertices);
```

```
for (Vertex=0 ; Vertex<NVertices ; Vertex++ )
{
   fscanf(fIn, "%lf %lf %lf", &X, &Y, &Z);
            RotX =
                        (X * ( sin(Gamma)*cos(Beta) ) +
                         Y * ( sin
(Gamma)*sin(Beta)*sin(Alpha) +
                         cos(Gamma)*cos(Alpha) ) +
                         Z * (
sin(Gamma)*sin(Beta)*cos(Alpha) -
                         cos(Gamma)*sin(Alpha)));

            RotY =
                        (X * ( cos (Gamma)*cos(Beta) ) +
                         Y * (
cos(Gamma)*sin(Beta)*sin(Alpha) -
                         sin(Gamma)*cos(Alpha) ) +
                         Z * (
cos(Gamma)*sin(Beta)*cos(Alpha) -
                         sin(Gamma)*sin(Alpha))) ;

            RotZ =
                        (-X *  sin (Beta)  +
                          Y * ( cos(Beta)*sin(Alpha) ) +
                          Z * ( cos(Beta)*cos(Alpha) ) );
fprintf(fOut, "%10.6lf %10.6lf %10.6lf\n", RotX, RotY,
RotZ);
            }
   }
}

FILE *open_input_file
            (
            char *Name
            )

{
FILE   *fin;

/* Okuma Hakkı içinde Dosyanın Açılması */
            fin = fopen( Name, "r" );

/* Eğer Programı Açma İŸlemi BaŸarısız ise */
            if( fin == NULL )
            {
                        printf(" DOSYA AÇILAMIYOR :
%s\n", Name);
                        exit (1);
            }
/* Dosya İŸaretçisinin Geri G"nderilmesi */

 return ( fin );
}


FILE *open_output_file
            (
            char *Name
            )

{
FILE   *fout;

/* Dosyanın Yazma Modunda Açılması */

            fout = fopen( Name, "w" ) ;

/* Eğer  Dosya Açma iŸlemi BaŸarızsa */

            if(fout == NULL)
            {
                        printf("DOSYA AÇILAMIYOR : %s\n",
Name);
            exit (1);
            }

/* Dosya İŸaretçisinin Geri G"nderilmesi */

 return ( fout );
}

void beep
            (
            void
            )

{
/* Saniyenin Onda biri kadar süreyle hoparl"re beep
sesinin g"nderilmesi*/
sound(1000);
delay(100);
nosound();
}
double get_one_double_value
            (
            char *Question
            )
{
            double Answer;
            int  NfieldsRead;
/* Sorunun Ekranda G"sterilmesi */
printf("%s", Question);

/*Geçerli bir Değer Girilinceye Kadar Veri Girilmesi */
do
{
NfieldsRead=scanf("%lf",&Answer);
/* Girilen Değerin Geçerli olup olmadığının Kontrol
Edilmesi*/
if (NfieldsRead !=1)
  {
  beep();
  }
}while (NfieldsRead !=1);

/* Imlecin Yeni Bir Satıra Getirilmesi */
 printf("\n");
 /* Okunan Değerin Geri D"ndürülmesi */
 return (Answer);
}

void check_parameters_two
            (
            int NumberOfParameters,
            char **FileNames
            )
{
 if (NumberOfParameters!=3)
   {
   /* Ekranın Temizlenmesi */
            clrscr();
            printf("          %s PROGRAMINI
ÇALISTIRMA HATASI\n",FileNames[0]);
            printf("     GIRIS VE ÇIKIS DOSYA
ADLARINI DOGRU OLARAK GIRMEK
ZORUNDASINIZ\n");
            printf("    KULLANIM  >>> %s <GIRIS
DOSYASI> <ÇIKIS DOSYASI>\n", FileNames[0]);
            exit(1);
   }
}
```

EK C

EĞRİ UYDURMA PROGRAMINA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/* EGRIUYD.C */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include  "library.h"
#include  "fitline2.h"
#include  <math.h>
#include  <ctype.h>
#include  <dos.h>
#include  <alloc.h>
#define MAX_ORDER  4

void information
(char **FileNames);

void fit_3d_curves
            (
            char *InputFileName,
            char *OutputFileName
            );

main
(
int   NParameters,char **ParameterStrings
            )
{

check_parameters_two( NParameters, ParameterStrings);

information ( ParameterStrings );
fit_3d_curves(ParameterStrings[1],ParameterStrings[2]);
gotoxy ( 25, 22 );
printf ( "PROGRAM BASARIYLA TAMAMLANDI\n"
);
return(0);
}


void information
(           char **FileNames     )
{
clrscr( );
printf( " ***   ÇIZGI UYDURMA PROGRAMI  ***\n");
printf( "    B-SPLINE ÇIZGI (VEYA ÇIZGILERININ )
BILGILERININ AYARLANMASI \n\n");
printf( "    ÇIZGI VERILERINI IÇEREN DOSYA ADI =
%s\n", FileNames[1] );
printf( "    YERLESTIRILMIS  ÇIZGILERI IÇEREN
DOSYA ADI (VERTEX AYARI) = %s\n", FileNames[2]
);
}

void fit_3d_curves
            (
char *InputFileName,char *OutputFileName
            )

{

FILE *FinSprs;
float *XSprs,     *YSprs,     *ZSprs;
int NVrtcsSprs; int NLinesSprs;
FILE *FoutFit;
float   *XFit,*YFit,*ZFit;
int     NVrtcsFit;
double   DeltaT;
int     Line, Vrtx;
FinSprs = open_input_file( InputFileName );
FoutFit = open_output_file( OutputFileName );
fscanf( FinSprs, "%d", &NLinesSprs );
fprintf( FoutFit, "%d\n", NLinesSprs );
gotoxy( 25,7 );
printf("YÜZEY %3d SATIRA SAHIPTIR", NLinesSprs
);
gotoxy ( 32, 16 );
printf( "SATIR ISLENILDI" );
Line = 0;
gotoxy(25,16);
printf( "%4d.", Line+1);
fscanf( FinSprs,"%d", &NVrtcsSprs);

XSprs = (float*) malloc( (unsigned)( NVrtcsSprs *
sizeof(float) ) );
YSprs = (float*) malloc( (unsigned)( NVrtcsSprs *
sizeof(float) ) ) ;
ZSprs = (float*) malloc( (unsigned) ( NVrtcsSprs * sizeof
(float) ) );
if( (XSprs==NULL) || (YSprs==NULL) ||
(ZSprs==NULL) )
{
printf( "HAFIZAYA ALINAMADI\n" );
exit(1);
}
gotoxy(25,9);
 deltaT=1;
NVrtcsFit  =  (int) ( ( (NVrtcsSprs - 1) / DeltaT ) + 1.5 );
XFit = (float*) malloc( (unsigned)( NVrtcsFit *
sizeof(float)));
YFit = (float*) malloc( (unsigned)( NVrtcsFit *
sizeof(float)));
ZFit = (float*) malloc( (unsigned)( NVrtcsFit *
sizeof(float)));
if( (XFit==NULL) || (YFit==NULL) ||  (ZFit==NULL) )
{
printf( "HAFIZAYA ALINAMADI \n" );exit ( 1 );
}
for( Vrtx=0 ; Vrtx<NVrtcsSprs ; Vrtx++)
{
fscanf( FinSprs, "%f %f %f", &(XSprs[Vrtx]),
&(YSprs[Vrtx]), &(ZSprs[Vrtx]));
}
bspline_3d_curve( XSprs, YSprs, ZSprs,
  XFit,  YFit,  ZFit,      NVrtcsSprs, DeltaT );


write_one_line( XFit, YFit, ZFit, NVrtcsFit, FoutFit );
for( Line=1; Line<NLinesSprs ; Line++)
{
gotoxy ( 25, 16 );printf( "%4d.", Line+1 );
fscanf ( FinSprs, "%d", &NVrtcsSprs );

for( Vrtx=0 ; Vrtx<NVrtcsSprs ; Vrtx++ )
            {
fscanf( FinSprs, "%f %f %f",  &(XSprs[Vrtx]),
&(YSprs[Vrtx]), &(ZSprs[Vrtx]) );
            }

bspline_3d_curve( XSprs, YSprs, ZSprs,
XFit,  YFit,  ZFit,  NVrtcsSprs, DeltaT );
write_one_line( XFit, YFit, ZFit, NVrtcsFit, FoutFit );
}}
```

```
int bspline_3d_curve
            (
float XSprs[],float YSprs[], float ZSprs[],
float SplnX[], float SplnY[], float SplnZ[],
int  NVrtcs,
double DeltaT
            )
{
register int Vrtx,   NKnots;
float        *Knotvector;
int          NSplinedVrtcs;
float        *X,*Y,*Z;
X = (float*) malloc( (unsigned) ( (NVrtcs+2) *
sizeof(float) ) );
Y = (float*) malloc( (unsigned) ( (NVrtcs+2) *
sizeof(float) ) );
Z = (float*) malloc( (unsigned) ( (NVrtcs+2) *
sizeof(float) ) );
if((X==NULL) || (Y==NULL) || (Z==NULL))
{
printf( "bspline_3d_curve FONKSIYONUNDA HATA:
YETERSIZ BELLEK\n");exit(1);
}
Knotvector = (float*) malloc((unsigned) ( NVrtcs +
MAX_ORDER*2) *sizeof(double) );

if( Knotvector == NULL )
{
printf( "bspline_3d_curve FONKSIYONUNDA HATA:
YETERSIZ BELLEK\n");exit(1);
}

for (Vrtx=0;Vrtx<NVrtcs;Vrtx++)
{
X[Vrtx]=XSprs[Vrtx];Y[Vrtx]=YSprs[Vrtx];
Z[Vrtx]=ZSprs[Vrtx];
}

NVrtcs=add_vertices_in_end_intervals( X, Y, Z, NVrtcs);
NKnots=knot_vector(Knotvector, MAX_ORDER,
NVrtcs);
NSplinedVrtcs = spline_3d( X, Y, Z, SplnX, SplnY,
SplnZ, Knotvector, MAX_ORDER, NVrtcs, NKnots,
DeltaT);
free ( X );free ( Y );free ( Z ) ;free ( Knotvector );
return( NSplinedVrtcs );
}

int add_vertices_in_end_intervals
(float X[], float Y[],  float Z[],   int NVrtcs )
{
register int Vrtx; double  XFirst, YFirst, ZFirst, XLast,
YLast, ZLast;
double     One3rd = 1.0/3.0;
double     Two3rd = 2.0/3.0;
XFirst = X[0] + One3rd * ( X[1] - X[0]);
YFirst = Y[0] + One3rd * ( Y[1] - Y[0]);
ZFirst = Z[0] + One3rd * ( Z[1]  - Z[0]);
XLast = X[NVrtcs-2] + Two3rd *( X[NVrtcs-1] -
X[NVrtcs-2]);
YLast = Y[NVrtcs-2] + Two3rd *( Y[NVrtcs-1] -
Y[NVrtcs-2]);
ZLast = Z[NVrtcs-2] + Two3rd *( Z[NVrtcs-1] -
Z[NVrtcs-2]);

X[NVrtcs+1] = X[NVrtcs-1];Y[NVrtcs+1] =Y[NVrtcs-1];
Z[NVrtcs+1] = Z[NVrtcs-1];


for( Vrtx=NVrtcs-1 ; Vrtx>1 ; Vrtx--)
{
 X[Vrtx] = X[Vrtx-1];
 Y[Vrtx] = Y[Vrtx-1];
 Z[Vrtx] = Z[Vrtx-1];}
X[1] = XFirst;Y[1] = YFirst;Z[1] = ZFirst;
X[NVrtcs] = XLast;Y[NVrtcs] = YLast;
Z[NVrtcs] = ZLast;
return( NVrtcs + 2 );
}

int knot_vector
            (
float KnotVector[],   int  Ordr, int  NVrtcs
            )


{
register int Knot;int Vrtx;int NKnots;
double LastPoint;
NKnots = NVrtcs + (Ordr-1) * 2;

for( Knot=0 ; Knot<(Ordr-1) ; Knot++)
{
   KnotVector[Knot] = 0.0;
}

for( Vrtx=0 ; Vrtx<(NVrtcs-2) ; Vrtx++, Knot++ )
{
  KnotVector[Knot] = (double) Vrtx;
}

LastPoint = (double) Vrtx-1 ;

for ( ; Knot<NKnots ; Knot++ )
{
  KnotVector[Knot] = LastPoint;
}

return ( NKnots );

}


int spline_3d
(float *X,float *Y,float *Z,float SplnX[],float SplnY[],
float SplnZ[],float KnotVector[],int  MaxOrdr,
            int  NVrtcs,int  NKnots,double DeltaT
            )
{
register int Knot,    SplnVrtx;
            int  Vrtx;int  Ordr;
            float **Basis;
            double TStart,     TEnd,     T;
            int  LastKnot;int  MaxVrtcsSpln;
MaxVrtcsSpln = (int)(((NVrtcs-3) / DeltaT) + 1.5 );
Basis = allocate_two_d_float_array( MAX_ORDER,
NKnots );

if (Basis==NULL)
{
printf(" bspline_3d_curve FONKSIYONUNDA HATA :
YETERSIZ BELLEK\n" );
exit (1);
}
SplnVrtx=0; SplnX[SplnVrtx] = 0.0;
SplnY[SplnVrtx] = 0.0; SplnZ[SplnVrtx] = 0.0;
LastKnot = (NVrtcs - 2) + (MAX_ORDER - 2) ;
for( Knot=3 ; Knot<=LastKnot ; Knot++ )
{
  if( Knot < LastKnot)
   {
     basis_order_1( Basis[0], NKnots, Knot );
   }
   TStart = KnotVector[Knot];
   if ( Knot == NVrtcs+MaxOrdr-2 )
            {
```

```
            TEnd = KnotVector[Knot+1] + 0.000001;
            }
else
            {
TEnd = KnotVector[Knot+1] - DeltaT + 0.000001;
            }
 T = TStart;
do
{

for( Ordr=2 ; Ordr<=MAX_ORDER ; Ordr++ )
{
 basis_order_n(Basis,KnotVector, T, Ordr, NKnots);
}
for (  Vrtx=0 ; Vrtx < NVrtcs ; Vrtx++ )
{
 SplnX[SplnVrtx] += ( X[Vrtx] * Basis [3] [Vrtx]);
 SplnY[SplnVrtx] += ( Y[Vrtx] * Basis [3] [Vrtx]);
 SplnZ[SplnVrtx] += ( Z[Vrtx] * Basis [3] [Vrtx]);
}
 SplnVrtx++;
    if( SplnVrtx<MaxVrtcsSpln )
     {
 SplnX[SplnVrtx] = 0.0; SplnY[SplnVrtx] = 0.0;
 SplnZ[SplnVrtx] = 0.0;
     }
  T = T + DeltaT;
  }while( T < TEnd );

}

free_two_d_float_array( MAX_ORDER, Basis );

 return ( SplnVrtx );
}

void basis_order_1
(float  *Basis1, int NKnots,  int  KnotNumber)
{
register int Knot;
for (Knot=0 ; Knot<NKnots ; Knot++)
{
 if ((Knot==KnotNumber))
 {
   Basis1[Knot] = 1.0;
 }
else
  {
  Basis1[Knot] = 0.0;
 } }}

void basis_order_n
            (
            float **Basis,
            float  KnotVector[],
            double T,
int   Ordr, int   NKnots
            )
{  double  C, CDiv,   D,          DDiv;
   register int BOrdr;
   int Knot;
   BOrdr=Ordr-1;
   for (Knot=0;Knot<=NKnots-BOrdr-1 ; Knot++)
   {
    if (Basis[BOrdr-1][Knot]== 0.0)
    {
     C=0.0;
    }
    else
    {
     CDiv=KnotVector[Knot+Ordr-1]-KnotVector[Knot];
     if (CDiv==0.0)
     {
            C=1.0;
     }
     else
     {
C=Basis[BOrdr-1][Knot] * (T-KnotVector[Knot])/CDiv;
     }   }
if(Basis[BOrdr-1][Knot+1]==0.0)
  {
   D=0.0;
  }
  else
  {
   DDiv=KnotVector[Knot+Ordr]-KnotVector[Knot+1];
     if (DDiv==0.0)
            D=1.0;
     else
D=Basis[BOrdr-1][Knot+1] * (T-notVector[Knot+Ordr]-
T)/DDiv;
  }
  Basis[BOrdr][Knot]=C+D;
 }
}

FILE *open_input_file
( char *Name)
{
FILE   *fin;
fin = fopen( Name, "r" );
            if( fin == NULL )
            {
printf(" DOSYA AÇILAMIYOR : %s\n", Name);
                        exit (1);
            } return ( fin );
}

FILE *open_output_file
(char *Name)
{
FILE   *fout;
            fout = fopen( Name, "w" ) ;
            if(fout == NULL)
            {
printf("DOSYA AÇILAMIYOR : %s\n", Name);
            exit (1);
            }
 return ( fout );
}

void beep  (void)
{
sound(1000); delay(100); nosound();
}

double get_one_double_value
            (char *Question)
{
double Answer; int  NFieldsRead;
printf("%s", Question);

do
{
NFieldsRead=scanf("%lf",&Answer);
Edilmesi*/
if (NFieldsRead !=1)
  {
  beep();
  }
}while (NFieldsRead !=1);
printf("\n");
return (Answer);
}
```

```
void check_parameters_two
(
int NumberOfParameters,char **FileNames
)
{
 if (NumberOfParameters!=3)
   {
            clrscr();
printf("            %s PROGRAMINI ÇALISTIRMA
HATASI\n",FileNames[0]);
printf("     GIRIS VE ÇIKIS DOSYA ADLARINI
DOGRU OLARAK GIRMEK ZORUNDASINIZ\n");
printf("    KULLANIM  >>> %s <GIRIS DOSYASI>
<ÇIKIS DOSYASI>\n", FileNames[0]);
exit(1);
   }
}

void write_one_line
            (
float *X,   float *Y,   float *Z,
int  NVertices,FILE  *fout
            )
{
 register int Vertex;
fprintf( fout, "%d\n", NVertices );
for ( Vertex=0 ; Vertex<NVertices ; Vertex++)
              {
fprintf( fout, "%10.6f %10.6f %10.6f\n",  X[Vertex],
Y[Vertex], Z[Vertex] );
              }
}

float **allocate_two_d_float_array
            (
            int NRows,int NCols
            )
{
float **A;
int n;

A = (float **) malloc( (unsigned) ( NRows * sizeof(float
*) ) );

if( !A )
            {
printf( "alloc_two_d_float_array IÇERISINE
YERLESTIRME HATASI\n" );
            return ( NULL );
            }

            for ( n=0 ; n<NRows ; n++ )
            {
A[n] = (float *) malloc( (unsigned) ( NCols * sizeof(float)
) );
if( !A[n] )
            {
printf( "alloc_two_d_float_array IÇERISINE
YERLESTIRME HATASI\n");
return( NULL ) ;
            }
   }
  return(A);
}

void free_two_d_float_array
            (
int NRows, float **A
            )
{
            register int n;
            for ( n=0 ; n<NRows ; n++)
            {
            free(A[n]);
            }
            free ( A ) ;
}
```

EK Ç

YÜZEY UYDURMA PROGRAMINA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*      YUZUYDUR.C      */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <process.h>
#include <graphics.h>
#include <alloc.h>
#include <ctype.h>
#include "library.h"
#include "fitsurf2.h"
#define  MAXORDER   4

double OneThird = 1.0 / 3.0;
double OneSixths = 1.0 / 6.0;
double TwoThird  = 2.0 / 3.0;
double TwoThirdSquared = 4.0 / ( 3.0 * 3.0 );
double OneNinths = 1.0 / 9.0;
double OneSixthsSquared = 1.0 /(6.0 * 6.0);
void information (   char **FileNames);

main (   int   NParameters,
   char **ParameterStrings)
{
    float   **XSrf, **YSrf,   **ZSrf;
    int  NVrtcsSrf;     int  NLinesSrf;
    float **FitX,   **FitY,   **FitZ;
    int NLinesFit;    int NVrtcsFit;
    double  DTLine,       DTVrtx;
check_parameters_two( NParameters, ParameterStrings );
information( ParameterStrings );
read_file_allocate( &XSrf, &YSrf, &ZSrf,   &NLinesSrf,
&NVrtcsSrf, ParameterStrings[1]);
gotoxy(5,12);
printf("YÜZEY %3d SATIR VE %3d TEPE NOKTAYA
SAHIPTIR ", NLinesSrf, NVrtcsSrf );
gotoxy(5,14);
DTLine=1;
gotoxy(5;15);
DTVrtx = 1;
 NLinesFit = (int)(((NLinesSrf - 1) / DTLine ) + 1.5 );
NVrtcsFit = (int)(((NVrtcsSrf - 1) / DTVrtx ) + 1.5 );
FitX = allocate_two_d_float_array( NLinesFit, NVrtcsFit
);
FitY = allocate_two_d_float_array( NLinesFit, NVrtcsFit
);
FitZ = allocate_two_d_float_array( NLinesFit, NVrtcsFit
);

if((FitX==NULL) || (FitY==NULL) ||  (FitZ==NULL) )
{
  printf("YETERSIZ HAFIZA\n");

  exit(1);
}
bspline_surface( XSrf, YSrf, ZSrf, FitX, FitY, FitZ,
 NLinesSrf, NVrtcsSrf, DTLine, DTVrtx, NLinesFit,
NVrtcsFit );
write_file( FitX, FitY, FitZ, NLinesFit, NVrtcsFit,
ParameterStrings[2]);
gotoxy(25,22);
printf( "PROGRAM BASARIYLA TAMAMLANDI\n" );
return(0);
}

void information (  char **FileNames)
{
clrscr();
printf( "***  YÜZEY UYDURMA PROGRAMI  ***\n");
printf( "  BIR 4 YÜZEYLI YÜZEY UYDURMA \n\n");
printf( "    GIRIS DOSYA ADI = %s\n", FileNames[1] );
printf( "    UYDURULMUS YÜZEY DOSYA ADI =
%s\n", FileNames[2] );
}

void write_file(
float **X,  float **Y,  float **Z,
int  NLines, int    NVertices, char   FileName[]
            )
{
            register int Line;
            FILE *fout;
            fout = open_output_file( FileName );
            gotoxy( 5, 20 );
printf ( "%s DOSYASINA YAZILIYOR ", FileName );
            fprintf( fout, "%d\n", NLines );
            for ( Line=0 ; Line<NLines ; Line++ )
            {
  write_one_line( X[Line], Y[Line], Z[Line], NVertices,
fout );
            }
            fclose(fout);
            }
void write_one_line   (
float *X,   float *Y,   float *Z,int   NVertices,FILE  *fout
            )
{
 register int Vertex;
            fprintf( fout, "%d\n", NVertices );
            for ( Vertex=0 ; Vertex<NVertices ; Vertex++)
              {
   fprintf(fout, "%10.6f %10.6f %10.6f\n",   X[Vertex],
Y[Vertex], Z[Vertex] );
              }
}

FILE *open_output_file (char *Name)
{
FILE   *fout;
fout = fopen( Name, "w" ) ;
            if(fout == NULL)
            {
printf("DOSYA AÇILAMIYOR : %s\n", Name);
            exit (1);
            }
return ( fout );
}

void spline_surface
(   float **X,    float **Y,    float **Z,    float **SplnX,
 float **SplnY,  float **SplnZ,  float KnotVectorLine[],
 float KnotVectorVrtx[],   int   NLines,   int   NVrtcs,
 int   NKnotsLine, int NKnotsVrtx,   double DeltaTLine,
    double DeltaTVrtx, int    NSplnVrtcs,

 int    SplnLines
)
```

```
{
register int Vrtx, Line;  float **BasisLine, **BasisVrtx;
float **Basis4Line,  **Basis4Vrtx;
 int KnotV, KnotL;  int SplnV, SplnL;  int  Ordr;
 double T,  TStart,  TEnd;
 int  LastKnotLine,  LastKnotVrtx;
BasisLine=allocate_two_d_float_array( MAXORDER,
NKnotsLine );
BasisVrtx=allocate_two_d_float_array( MAXORDER,
NKnotsVrtx );
if( BasisLine==NULL || BasisVrtx==NULL)
{
printf(" BSPLINE_YÜZEY İÇERISINDE HATA:
YETERSIZ BELLEK\n");exit(1);
}

Basis4Line=allocate_two_d_float_array( NSplnLines,
NLines );
Basis4Vrtx=allocate_two_d_float_array( NSplnVrtcs,
NVrtcs );
if(Basis4Line==NULL || Basis4Vrtx==NULL )
{
printf(" BSPLINE_YÜZEY İÇERISINDE HATA:
YETERSIZ BELLEK\n");
exit(1);
}
LastKnotVrtx = NVrtcs - 2 + MAXORDER - 2;
SplnV = 0;
for( KnotV=3 ; KnotV<=LastKnotVrtx ; KnotV++ )
{
 if( KnotV < LastKnotVrtx )
   {
     basis_order_1( BasisVrtx, NKnotsVrtx, KnotV );
   }
TStart = KnotVectorVrtx[KnotV];
if( KnotV==NVrtcs+MAXORDER-2)
{
 TEnd = KnotVectorVrtx[KnotV+1] + 0.001;
}
else
{
TEnd = KnotVectorVrtx[KnotV+1] - DeltaTVrtx + 0.001;
}
T = TStart;
do
{
for( Ordr=2 ; Ordr<=MAXORDER;Ordr++ )
{
basis_order_n( BasisVrtx, KnotVectorVrtx, T,
Ordr,NKnotsVrtx);
}
for (Vrtx=0; Vrtx<NVrtcs;Vrtx++ )
{
Basis4Vrtx[SplnV][Vrtx] = BasisVrtx[MAXORDER-
1][Vrtx];
}
T = T + DeltaTVrtx; SplnV++;
}while(T<=TEnd);
}
LastKnotLine = NLines-2 + MAXORDER - 2;
SplnL = 0;
for( KnotL=3 ; KnotL<=LastKnotLine;KnotL++ )
{
 if (KnotL<LastKnotLine)
   {
     basis_order_1(BasisLine, NKnotsLine, KnotL);
   }
TStart = KnotVectorLine[KnotL];
if( KnotL==NLines+MAXORDER-2)
{
TEnd = KnotVectorLine[KnotL+1] + 0.001;
}
else
{
TEnd = KnotVectorLine[KnotL+1] - DeltaTLine + 0.001;
}
T = TStart;
do
{
for( Ordr=2;Ordr<=MAXORDER ; Ordr++ )
{
  basis_order_n( BasisLine, KnotVectorLine, T,
Ordr,NKnotsLine );
}
for (Line=0;Line<NLines;Line++ )
{
Basis4Line[SplnL][Line] = BasisLine[MAXORDER-
1][Line];
}
T=T + DeltaTLine; SplnL++;
}while( T<=TEnd );
}

gotoxy(5,17);
printf("EGRI YÜZEY - %d SATIR VE %d TEPE
NOKTASI HESAPLANIYOR",NSplnLines, NSplnVrtcs
);
gotoxy(8,18); printf ( "SATIR ISLENILDI " );
for ( SplnL=0 ; SplnL<NSplnLines ; SplnL++)
{
 gotoxy(2,18);
 printf("%4d. ",SplnL+1);


for( SplnV=0 ; SplnV<NSplnVrtcs ; SplnV++)
{
SplnX[SplnL][SplnV] = 0.0; SplnY[SplnL][SplnV] = 0.0;
SplnZ[SplnL][SplnV] = 0.0;
for (Line=0;Line<NLines ; Line++ )
{
 for( Vrtx=0 ; Vrtx<NVrtcs ; Vrtx++ )
 {
  SplnX[SplnL][SplnV]  += X[Line][Vrtx] *
Basis4Vrtx[SplnV][Vrtx] * Basis4Line[SplnL][Line];
  SplnY[SplnL][SplnV]  += Y[Line][Vrtx] *
Basis4Vrtx[SplnV][Vrtx] * Basis4Line[SplnL][Line];
  SplnZ[SplnL][SplnV]  += Z[Line][Vrtx] *
Basis4Vrtx[SplnV][Vrtx] * Basis4Line[SplnL][Line];
  } } } }
free_two_d_float_array( MAXORDER, BasisLine );
free_two_d_float_array( MAXORDER, BasisVrtx );
free_two_d_float_array( NSplnLines, Basis4Line );
free_two_d_float_array ( NSplnVrtcs, Basis4Vrtx );
}

void basis_order_1
( float **Basis,  int  NKnots,  int  KnotNumber)
{
  register int Knot;
for( Knot=0 ; Knot<NKnots ; Knot++ )
{
if( Knot == KnotNumber )
{
  Basis[0][Knot] = 1.0;
}
else
{
  Basis[0][Knot] = 0.0;
}}}
float **allocate_two_d_float_array
            (int NRows,int NCols)
{
float **A;int n;
A=(float**) malloc((unsigned)(NRows*sizeof(float*)));
            if(!A)
            {
```

```
printf( "alloc_two_d_float_array IÇERISINE
YERLESTIRME HATASI \n" );
            return ( NULL );
            }
for (n=0;n<NRows;n++)
            {
A[n] = (float *) malloc( (unsigned) ( NCols * sizeof(float)
) );
if( !A[n] ) {
printf( "alloc_two_d_float_array IÇERISINE
YERLESTIRME HATASI \n");
            return( NULL ) ;
            } }
   return(A);
}

void bspline_surface
(
 float **XOrg,   float **YOrg,   float **ZOrg,
  float **SplnX,   float **SplnY,   float **SplnZ,
 int   NLines,   int   NVrtcs,   double DeltaTLine,
  double DeltaTVrtx,   int   NSplnLines, int NSplnVrtcs
)
{
  register int Vrtx;   int    Line;
  int    NKnotsLine, NKnotsVrtx;
  float **XCopy, **YCopy,  **ZCopy;
  float  *KnotVectorLine,  *KnotVectorVrtx;
  int    NCopyLines, NCopyVrtcs;
NCopyLines = NLines + 2;NCopyVrtcs = NVrtcs + 2;
NKnotsLine = NCopyLines + (MAXORDER-1) * 2;
NKnotsVrtx = NCopyVrtcs + (MAXORDER-1) * 2;
XCopy = allocate_two_d_float_array( NCopyLines,
NCopyVrtcs );
YCopy = allocate_two_d_float_array( NCopyLines,
NCopyVrtcs );
ZCopy = allocate_two_d_float_array( NCopyLines,
NCopyVrtcs );

if( XCopy==NULL || YCopy==NULL || ZCopy==NULL
) {
   printf( " bspline yüzey fonksiyonunda hata -->
YETERSIZ HAFIZA \n" );  exit(0);
}
KnotVectorLine = (float*) malloc( (unsigned)
(NKnotsLine * sizeof(float)));
KnotVectorVrtx = (float*) malloc( (unsigned)
(NKnotsVrtx * sizeof(float)));
if( KnotVectorLine==0 || KnotVectorVrtx==0 )
{
 printf( " bspline yüzey fonksiyonunda hata -->
YETERSIZ HAFIZA \n" ); exit(0);
}

for (Line=0;Line<NLines;Line++)
{
 for(Vrtx=0;Vrtx<NVrtcs;Vrtx++)
  {
    XCopy[Line][Vrtx] = XOrg[Line][Vrtx];
    YCopy[Line][Vrtx] = YOrg[Line][Vrtx];
    ZCopy[Line][Vrtx] = ZOrg[Line][Vrtx];
  }
}
NCopyLines = add_lines_in_endintervals( XCopy,
YCopy, ZCopy, NLines,NVrtcs );
NCopyVrtcs = add_vertices_in_endintervals( XCopy,
YCopy, ZCopy,NCopyLines, NVrtcs);
knot_vector( KnotVectorLine, NCopyLines,
NKnotsLine);
knot_vector( KnotVectorVrtx, NCopyVrtcs,
NKnotsVrtx);
spline_surface( XCopy, YCopy, ZCopy, SplnX, SplnY,
SplnZ, KnotVectorLine, KnotVectorVrtx, NCopyLines,
NCopyVrtcs, NKnotsLine, NKnotsVrtx, DeltaTLine,
DeltaTVrtx, NSplnVrtcs, NSplnLines );
free_two_d_float_array( NCopyLines, XCopy );
free_two_d_float_array( NCopyLines, YCopy );
free_two_d_float_array( NCopyLines, ZCopy );
free( KnotVectorVrtx );
free( KnotVectorLine );
}
double get_one_double_value
(char *Question)
{
double Answer;int  NfieldsRead;
printf("%s", Question);
do
{
NfieldsRead=scanf("%lf",&Answer);
if (NfieldsRead !=1)
  {  beep();  }
}while (NfieldsRead !=1);
printf("\n");return (Answer);
}
void read_file_allocate
(float ***X,float ***Y,float ***Z,int *NLines,
int *NVertices,char * FileName )
{
            register int Line;
            FILE *fin;
fin=open_input_file(FileName);
gotoxy(5,10);
printf("VERILER %s DOSYASINDAN OKUNUYOR ",
FileName);
fscanf( fin, "%d", NLines);
*X = (float **) malloc( (unsigned) ( *NLines *
sizeof(float *)));
*Y = (float **) malloc( (unsigned) ( *NLines * sizeof
(float *)));
*Z = (float **) malloc( (unsigned) ( *NLines * sizeof
(float *)));
if ( !*X || !*Y  ||  !*Z )
{
printf ( "n_read_ship_file_alloc Içerisine Yerleştirme
Hatası 1\n");exit(1);
}
for( Line=0 ; Line<*NLines ; Line++)
            {
*NVertices = read_one_line_allocate ( & ((*X)[Line]) ,
   & ((*Y)[Line]) ,   & ((*Z)[Line]) ,fin ) ;
            }
fclose(fin);
}

void check_parameters_two
(int NumberOfParameters,char **FileNames)
{
 if (NumberOfParameters!=3)
  {
clrscr();
printf("  %s PROGRAMINI ÇALISTIRMA
HATASI\n",FileNames[0]);
printf("     GIRIS VE ÇIKIS DOSYA ADLARINI
DOGRU OLARAK GIRMEK ZORUNDASINIZ\n");
printf("   KULLANIM  >>> %s <GIRIS DOSYASI>
<ÇIKIS DOSYASI>\n", FileNames[0]);
exit(1);
   }
}

void basis_order_n
(
float **Basis,   float  KnotVector[],    double T,
  int   Ordr,    int   NKnots
)
{
```

```
double  C, CDiv, D, DDiv;
register int BOrdr;
int      Knot;
BOrdr = Ordr - 1;
for( Knot=0 ; Knot <= NKnots-BOrdr-1 ; Knot++)
{
if( Basis[BOrdr-1][Knot] == 0.0 )
{   C = 0.0; }
else
{
  CDiv = KnotVector[Knot+Ordr-1] - KnotVector[Knot];
  if( CDiv == 0.0)
  {       C = 1.0;    }
  else
  {C=Basis[BOrdr-1][Knot] * ( T - KnotVector[Knot] ) /
CDiv;    }
}
if( Basis[BOrdr-1][Knot+1] == 0.0)
{   D = 0.0; }
else
{
  DDiv = KnotVector[Knot+Ordr] - KnotVector[Knot+1];
  if( DDiv == 0.0)
  {    D = 1.0;  }
  else
  {
D = Basis[BOrdr-1][Knot+1] * ( KnotVector[Knot+Ordr]
- T )/ DDiv;}
}
Basis[BOrdr][Knot] = C + D;
}

}

FILE *open_input_file(char *Name)
{
FILE   *fin;
            fin = fopen( Name, "r" );
            if( fin == NULL )
            {
printf(" DOSYA AÇILAMIYOR : %s\n", Name);
                        exit (1);
            }
return ( fin );
}

void beep( void)
{
sound(1000);delay(100);nosound();
}

void free_two_d_float_array
(int NRows,float **A )
{
            register int n;
            for ( n=0 ; n<NRows ; n++)
            {
            free(A[n]);
            }
            free ( A ) ;
}

void knot_vector (
    float KnotVector[],    int  NVrtcs,    int  NKnots
)
{

    int Vrtx;    register int Knot;    double  LastVrtx;

for( Knot=0;Knot<MAXORDER-1;Knot++)
{
KnotVector[Knot]=0.0;
}

for( Vrtx=0;Vrtx<(NVrtcs-2) ; Vrtx++, Knot++)
{   KnotVector[Knot] = (float)(Vrtx); }
LastVrtx = (double) Vrtx-1;
for ( ; Knot<NKnots;Knot++ )
{   KnotVector[Knot] = LastVrtx; }
}

int add_vertices_in_endintervals (
 float  **X,  float  **Y,  float  **Z,  int NLines,
 int NVrtcs)
{
  register int Vrtx, Line;
NVrtcs = NVrtcs + 2;
Vrtx = NVrtcs - 1;
for (Line=0 ; Line<NLines ; Line++ )
{
 X[Line][Vrtx] = X[Line][Vrtx-2];
 Y[Line][Vrtx] = Y[Line][Vrtx-2];
 Z[Line][Vrtx] = Z[Line][Vrtx-2];
}
Vrtx = NVrtcs - 2;
for (Line=0;Line<NLines;Line++)
{
 X[Line][Vrtx]=X[Line][Vrtx-
2]+TwoThird*(X[Line][Vrtx+1] - X[Line][Vrtx-2]);
  Y[Line][Vrtx]=Y[Line][Vrtx-
2]+TwoThird*(Y[Line][Vrtx+1] - Y[Line][Vrtx-2]);
  Z[Line][Vrtx]=Z[Line][Vrtx-
2]+TwoThird*(Z[Line][Vrtx+1] - Z[Line][Vrtx-2]);
}
for( Vrtx=NVrtcs-3;Vrtx>0;Vrtx--)
{
 for (Line=0;Line<NLines;Line++)
 {
   X[Line][Vrtx]=X[Line][Vrtx-1];
   Y[Line][Vrtx]=Y[Line][Vrtx-1];
   Z[Line][Vrtx]=Z[Line][Vrtx-1];
 }
}
Vrtx = 1;
for (Line=0;Line<NLines;Line++ )
{
  X[Line][Vrtx] = X[Line][Vrtx-1]+
OneThird*(X[Line][Vrtx+1]-X[Line][Vrtx-1]);
  Y[Line][Vrtx] = Y[Line][Vrtx-1]+
OneThird*(Y[Line][Vrtx+1]-Y[Line][Vrtx-1]);
  Z[Line][Vrtx] = Z[Line][Vrtx-1]+
OneThird*(Z[Line][Vrtx+1]-Z[Line][Vrtx-1]);
}
return(NVrtcs);
}

int add_lines_in_endintervals
(  float **X,   float **Y,   float **Z,   int NLines,
int NVrtcs )
{
register int Vrtx, Line;NLines = NLines + 2;
Line = NLines - 1;
for( Vrtx=0 ; Vrtx<NVrtcs ; Vrtx++ )
{
 X[Line][Vrtx] = X[Line-2][Vrtx];
 Y[Line][Vrtx] = Y[Line-2][Vrtx];
 Z[Line][Vrtx] = Z[Line-2][Vrtx];
}
Line = NLines - 2;
for( Vrtx=0 ; Vrtx<NVrtcs ; Vrtx++ )
{
  X[Line][Vrtx] = X[Line-
2][Vrtx]+TwoThird*(X[Line+1][Vrtx]-X[Line-2][Vrtx]);
  Y[Line][Vrtx] = Y[Line-
2][Vrtx]+TwoThird*(Y[Line+1][Vrtx]-Y[Line-2][Vrtx]);
```

```
Z[Line][Vrtx] = Z[Line-
2][Vrtx]+TwoThird*(Z[Line+1][Vrtx]-Z[Line-2][Vrtx]);
}
for ( Line=NLines-3;Line>0;Line--)
{  for( Vrtx=0;Vrtx<NVrtcs;Vrtx++ )
   {
            X[Line][Vrtx] = X[Line-1][Vrtx];
            Y[Line][Vrtx] = Y[Line-1][Vrtx];
            Z[Line][Vrtx] = Z[Line-1][Vrtx];
   }
}
Line=1;
for( Vrtx=0 ; Vrtx<NVrtcs ; Vrtx++ )
{
   X[Line][Vrtx] = X[Line-1][Vrtx] + OneThird
*(X[Line+1][Vrtx]-X[Line-1][Vrtx]);
   Y[Line][Vrtx] = Y[Line-1][Vrtx] + OneThird
*(Y[Line+1][Vrtx]-Y[Line-1][Vrtx]);
   Z[Line][Vrtx] = Z[Line-1][Vrtx] + OneThird
*(Z[Line+1][Vrtx]-Z[Line-1][Vrtx]);
}return(NLines);
}

int read_one_line_allocate
(float **X,float **Y, float **Z,  FILE *fin  )
{
register int n; int NVertices;
fscanf(fin,"%d\n", &NVertices);
if (NVertices==0)
            {
              allocate_memory_for_one_line(X,Y,Z,1);
            }else
            {
allocate_memory_for_one_line( X, Y, Z, NVertices);
            }
            for (n=0 ;n<NVertices ; n++ )
            {
 fscanf(fin, " %f   %f   %f ", &((*X)[n]), &((*Y)[n]),
&((*Z)[n]));
            } return(NVertices);
}
```

```
void allocate_memory_for_one_line
            (
float       **X,float  **Y,float  **Z,
int         NVertices
            )
{
*X =(float*) malloc((unsigned)(NVertices *
sizeof(float)));
*Y =(float*) malloc((unsigned)(NVertices *
sizeof(float)));
*Z =(float*) malloc((unsigned)(NVertices *
sizeof(float)));

if ((*X==NULL) || (*Y==NULL) ||  (*Z==NULL))
  {
  printf( " allocate_memory_for_one_line
FONKSIYONU IÇERISINDE HATA\n" );
  printf ( "%3d NOKTAYI HAFIZAYA
YERLESTIRMEYI DENIYOR\n", NVertices);
  printf( "HAFIZA TASMASI\n" );
  exit ( 1 ) ;
  }
}
```

EK D

YÜZEY NORMALLERİNİ HESAPLAMA İŞLEMİ YAPAN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*    YUZNORM.C      */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <library.h>

void information
(
char **FileNames
);
void make_normal_vectors_on_surface
(
char *InputFile,
char *OutputFile
);
void make_normal_vector_on_one_line
(
float X[][1000], float Y[][1000], float Z[][1000],
float NX[], float NY[], float NZ[],
int  NVertices
);
void read_one_line
(
float *X, float *Y, float *Z,
int *NVertices,
FILE  *fin
);

void write_one_line
(
float *X,  float *Y, float *Z,
int NVertices,
FILE *fout
);
void copy_line
(
float  XFrom[], float  YFrom[], float  ZFrom[],
float  XTo[],  float  YTo[], float  ZTo[],
int  NVertices
);
main
(
  int NumberOfParameters,
  char **ParameterStrings
)
{
clrscr();
check_parameters_two(NumberOfParameters,
ParameterStrings);
information( ParameterStrings );
make_normal_vectors_on_surface(
ParameterStrings[1],  ParameterStrings[2]);
printf( "\n\n\n\n\n\n\nPROGRAM BAŞARIYLA
TAMAMLANDI\n" );
return(0);
}
void information
(   char **FileNames )
{
clrscr();
printf("       *** YÜZEYNORMALLERİ PROGRAMI ***\n");
printf("      BİR GRİD ÜZERİNDEKİ YÜZEY NORMAL
VEKTÖRÜNÜN HESAPLANMASI \n\n");
printf("  GİRİŞ DOSYA ADI = %s\n", FileNames[1]);
printf("  ÇIKIŞ DOSYA ADI = %s\n", FileNames[2]);
}

void make_normal_vectors_on_surface
(char *InputFile, char *OutputFile )
{
register int Line; int NLines,    NVertices;
FILE  *fSurfaceFile,
     *fNormalVectorFile;
static float X[3][1000], Y[3][1000],    Z[3][1000];
static float NX[1000],  NY[1000], NZ[1000];
fSurfaceFile = open_input_file(InputFile);
fNormalVectorFile = open_output_file(OutputFile);
fscanf(fSurfaceFile,"%d", &NLines );
fprintf(fNormalVectorFile, "%d\n", NLines);
gotoxy(10,10);
printf("%d SATIR İŞLENİYOR\n", NLines);
gotoxy(10,11);
printf( "İŞLENİLEN SATIR #");
gotoxy(27,11);
printf("%4d",1);
read_one_line(X[1],Y[1],Z[1],&NVertices,fSurfaceFile);
read_one_line(X[2],Y[2],Z[2],&NVertices,fSurfaceFile);
copy_line(X[1],Y[1],Z[1],X[0],Y[0],Z[0],NVertices);
make_normal_vector_on_one_line(X,Y,Z,NX,NY,NZ,NVertices
);
write_one_line( NX, NY, NZ, NVertices, fNormalVectorFile);
for( Line=1 ; Line<NLines-1 ; Line++ )
{
gotoxy(27,11);
printf("%4d",Line+1);
copy_line( X[1],Y[1],Z[1],X[0],Y[0],Z[0],NVertices);
copy_line( X[2],Y[2],Z[2],X[1],Y[1],Z[1],NVertices);
read_one_line(X[2],Y[2],Z[2], &NVertices, fSurfaceFile );
make_normal_vector_on_one_line( X, Y, Z, NX, NY, NZ,
NVertices );

write_one_line( NX, NY, NZ, NVertices, fNormalVectorFile );
}
gotoxy(27,11);
printf("%4d",Line+1);
copy_line(X[1],Y[1],Z[1],X[0],Y[0],Z[0], NVertices );
copy_line(X[2],Y[2],Z[2],X[1],Y[1],Z[1], NVertices );

make_normal_vector_on_one_line(X,Y, Z, NX, NY, NZ,
NVertices );
write_one_line( NX, NY, NZ, NVertices, fNormalVectorFile );
}

void make_normal_vector_on_one_line
(
  float X[][1000], float Y[][1000],  float Z[][1000],   float NX[],
  float NY[],  float NZ[],
  int NVertices
)
{
register int Vrtx;
double DVX,      DVY,      DVZ;
```

```
double DLX,  DLY,   DLZ;
double  LengthNormalVector;
for( Vrtx=1 ; Vrtx<(NVertices+1) ; Vrtx++ )
{
DVX=X[1][Vrtx+1]-X[1][Vrtx-1];
DVY=Y[1][Vrtx+1]-Y[1][Vrtx-1];
DVZ=Z[1][Vrtx+1]-Z[1][Vrtx-1];
DLX=X[2][Vrtx]-X[0][Vrtx];
DLY=Y[2][Vrtx]-Y[0][Vrtx];
DLZ=Z[2][Vrtx]-Z[0][Vrtx];

NX[Vrtx-1]=DVY*DLZ - DLY*DVZ;
NY[Vrtx-1]=(-1) * ( DVX * DLZ - DLX * DVZ);
NZ[Vrtx-1]=DVX*DLY-DLX*DVY;

LengthNormalVector=sqrt(NX[Vrtx-1] * NX[Vrtx-
1]+ NY[Vrtx-1] * NY[Vrtx-1]+NZ[Vrtx-1] *
NZ[Vrtx-1]);

if( LengthNormalVector > 0.0 )
{
 NX[Vrtx-1] /= LengthNormalVector;
 NY[Vrtx-1] /= LengthNormalVector;
 NZ[Vrtx-1] /= LengthNormalVector;
}
else
{
 gotoxy(20,15);
 printf("%d. VERTEX ÜZERİNDEKİ NORMAL
VEKTÖR UZUNLUĞU- 0.0'DIR\n",Vrtx);
 NX[Vrtx-1]= NY[Vrtx-1] = NZ[Vrtx-1] = 1.0;
  }
 }
}
void read_one_line
(
    float *X,     float *Y,     float *Z,
    int  *NVertices,
    FILE  *fin
)
{
  register int Vertex;
fscanf( fin, "%d", NVertices);
for( Vertex=1 ; Vertex< (*NVertices+1) ;
Vertex++)
{
 fscanf(fin,"%f %f %f", &(X[Vertex]),
&(Y[Vertex]), &(Z[Vertex]) );
}

X[0]=X[1]; Y[0]=Y[1]; Z[0]=Z[1];

X[*NVertices+1]=X[*NVertices];
 Y[*NVertices+1]=Y[*NVertices];
 Z[*NVertices+1]=Z[*NVertices];
}

void copy_line
(
  float XFrom[],  float YFrom[],  float ZFrom[],
  float XTo[],  float YTo[],  float ZTo[],
  int NVertices
)
{
register int Vertex;

for( Vertex=0 ;Vertex<(NVertices+2) ; Vertex++ )
  {
  XTo[Vertex] = XFrom[Vertex];
  YTo[Vertex] = YFrom[Vertex];
  ZTo[Vertex] = ZFrom[Vertex];
  }
}
FILE *open_input_file
(           char *Name  )
{
FILE  *fin;
            fin = fopen( Name, "r" );
            if( fin == NULL )
            {
            printf(" DOSYA AÇILAMIYOR : %s\n", Name);
            exit (1);
}
return ( fin );
}
FILE *open_output_file
            (
            char *Name
            )
{
FILE  *fout;
            fout = fopen( Name, "w" ) ;
            if(fout == NULL)
            {
            printf("DOSYA AÇILAMIYOR : %s\n", Name);
            exit (1);
            }

return ( fout );
}

void check_parameters_two
            (
            int NumberOfParameters,
            char **FileNames
            )
{
 if (NumberOfParameters!=3)
   {
   /* Ekranın Temizlenmesi */
            clrscr();
            printf("          %s PROGRAMINI ÇALIŞTIRMA
HATASI\n",FileNames[0]);
            printf("     GİRİŞ VE ÇIKIŞ DOSYA ADLARINI
DOĞRU OLARAK GİRMEK ZORUNDASINIZ\n");
            printf("    KULLANIM >>> %s <GİRİŞ
DOSYASI> <ÇIKIŞ DOSYASI>\n", FileNames[0]);
            exit(1);
   }
}

void write_one_line
            (
            float *X,
            float *Y,
            float *Z,
            int  NVertices,
            FILE  *fout
            )
{
 register int Vertex;

            fprintf( fout, "%d\n", NVertices );
            for ( Vertex=0 ; Vertex<NVertices ; Vertex++)
              {
  fprintf( fout, "%10.6f %10.6f %10.6f\n",
                        X[Vertex], Y[Vertex], Z[Vertex] );
              }
}
```

EK E

KESİCİ OFSETLERİNİ HESAPLAYAN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

OFSETHES.C

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <library.h>

void information
(
char **FileNames
);

void calculate_offsets_general_mill
 (
double CutterEdgeRadius,
double  CutterOuterRadius,
char *InputFile,
char  *OutputFile
 );


main
(
   int NumberOfParameters,
   char **ParameterStrings
)

{
    double  CutterEdgeRadius,
                CutterOuterRadius;

check_parameters_two( NumberOfParameters,
ParameterStrings );
information( ParameterStrings );

CutterOuterRadius = get_one_double_value("R1
DEGERİNİ GİRİNİZ..:");

CutterEdgeRadius  = get_one_double_value("R2
DEGERİNİ GİRİNİZ..:");

if (fabs(CutterOuterRadius-CutterEdgeRadius) <
0.000001 )
{
printf("YUVARLAK UÇLU KESİCİ
AYARLANIYOR\n" );
}
else
if( CutterEdgeRadius == 0.0 )
{
printf("SİVRİ UÇLU KESİCİ AYARLANIYOR\n"
);
}
else
{
printf("GENEL ŞEKİLLİ BİR KESİCİ
AYARLANIYOR\n" );
}
calculate_offsets_general_mill( CutterEdgeRadius,
CutterOuterRadius, ParameterStrings[1],
ParameterStrings[2]);
printf( "\n\nPROGRAM BAŞARIYLA TAMAMLANDI\n");
return(0);
}

void information
(
char **FileNames
)

{
clrscr();

printf("          *** OFFSETHESAPLA PROGRAMI ***\n");
printf("       GENEL BİR ŞEKİLDEKİ KESİCİ İÇİN
OFFSETLERİN AYARLANMASI\n\n");
printf("  GİRİŞ DOSYA ADI = %s\n", FileNames[1]);
printf("  ÇIKIŞ DOSYA ADI = %s\n", FileNames[2]);
printf("\n");
}

void   calculate_offsets_general_mill
 (
     double CutterEdgeRadius,  double CutterOuterRadius,
      char  *InputFile,   char  *OutputFile
 )
 {
     FILE  *fNormalVectorFile;
     double NormalVectorX,
                NormalVectorY,
                NormalVectorZ;

     int   NLines,NVertices;
     double NormalXYPlaneX,
                NormalXYPlaneY,
                NormalXYPlaneZ;

    FILE  *fOffsetFile;
    double OffsetX,      OffsetY,    OffsetZ;
    double o1X,    o1Y,    o1Z;
    double o2X,    o2Y,    o2Z;
    double o3X,    o3Y,    o3Z;

    double  LengthNormalVectorXYPlane;
    register int  Line,Vertex;

fNormalVectorFile = open_input_file(InputFile);
fOffsetFile = open_output_file( OutputFile );

fscanf(fNormalVectorFile,"%d",&NLines);
fprintf(fOffsetFile,"%d\n",NLines);

gotoxy( 10, 12 );
printf("%d. SATIR İŞLENİYOR\n",NLines);
gotoxy(10,13);
printf ( "SATIR İŞLENİLDİ # " );
for(Line=0;Line<NLines;Line++)
{
gotoxy(27,13);
printf("%4d",Line+1);

fscanf(fNormalVectorFile, "%d",&NVertices);
fprintf(fOffsetFile,"%d\n", NVertices);

for(Vertex=0;Vertex<NVertices;Vertex++)
{
```

```
fscanf(fNormalVectorFile, "%lf %lf %lf",
&NormalVectorX, &NormalVectorY,
&NormalVectorZ);

LengthNormalVectorXYPlane =
sqrt((NormalVectorX * NormalVectorX ) +
( NormalVectorY * NormalVectorY));

NormalXYPlaneX = NormalVectorX /
LengthNormalVectorXYPlane;
NormalXYPlaneY = NormalVectorY /
LengthNormalVectorXYPlane;
NormalXYPlaneZ = 0.0;

o1X = NormalXYPlaneX * (CutterOuterRadius -
CutterEdgeRadius);
o1Y = NormalXYPlaneY * (CutterOuterRadius -
CutterEdgeRadius);
o1Z = NormalXYPlaneZ * (CutterOuterRadius -
CutterEdgeRadius);
o2X = NormalVectorX * CutterEdgeRadius;
o2Y = NormalVectorY * CutterEdgeRadius;
o2Z = NormalVectorZ * CutterEdgeRadius;

o3X = 0;
o3Y = 0;
o3Z = CutterEdgeRadius;

OffsetX = o1X + o2X - o3X;
OffsetY = o1Y + o2Y - o3Y;
OffsetZ = o1Z + o2Z - o3Z;
fprintf(fOffsetFile, "%10.6f %10.6f
%10.6f\n",OffsetX,OffsetY, OffsetZ);
   }
  }
 }


 FILE *open_input_file
(char *Name)
{
FILE   *fin;

/* Okuma Hakkı içinde Dosyanın Açılması */
            fin = fopen( Name, "r" );

/* Eğer Programı Açma İşlemi Başarısız ise */
            if( fin == NULL )
            {
                        printf(" DOSYA
AÇILAMIYOR : %s\n", Name);
                        exit (1);
            }
/* Dosya İşaretçisinin Geri Gönderilmesi */

 return ( fin );
}


FILE *open_output_file
            (
            char *Name
            )
{
FILE   *fout;

            fout = fopen( Name, "w" ) ;
            if(fout == NULL)
            {
            printf("DOSYA AÇILAMIYOR : %s\n", Name);
            exit (1);
            }

/* Dosya İşaretçisinin Geri Gönderilmesi */

 return ( fout );
}
double get_one_double_value
(           char *Question        )
{
            double Answer;
            int  NfieldsRead;

/* Sorunun Ekranda Gösterilmesi */
printf("%s", Question);

/*Geçerli bir Değer Girilinceye Kadar Veri Girilmesi */
do
{
NfieldsRead=scanf("%lf",&Answer);
/* Girilen Değerin Geçerli olup olmadığının Kontrol Edilmesi*/
if (NfieldsRead !=1)
  {
  beep();
  }
}while (NfieldsRead !=1);

/* İmlecin Yeni Bir Satıra Getirilmesi */
 printf("\n");
 /* Okunan Değerin Geri Döndürülmesi */
 return (Answer);
}

void beep
            (
            void
            )

{
/* Saniyenin Onda biri kadar süreyle hoparlöre beep sesinin
gönderilmesi*/
sound(1000);
delay(100);
nosound();
}

void check_parameters_two
            (
            int NumberOfParameters,
            char **FileNames
            )

{
 if (NumberOfParameters!=3)
   {
   /* Ekranın Temizlenmesi */
            clrscr();
            printf("PROGRAM İÇERİSİNDE HATA
%s\n",FileNames[0]);
            printf("GİRİŞ VE ÇIKIŞ DOSYA ADLARINI
DOĞRU OLARAK GİRMEK ZORUNDASINIZ\n");
            printf("KULLANIM >>> %s GİRİŞ DOSYASI
ÇIKIŞ DOSYASI\n", FileNames[0]);
            exit(1);
   }
}
```

EK F

KESİCİ YOLUNU BULAN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*          OFSETEKLE.C     */

#include <stdio.h>
#include <math.h>
#include <stdiib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <library.h>

void information
 (   char **FileNames   );

void add_offsets_to_surface
 (     char  *SurfaceFile,
    char  *OffsetsFileFile,
    char  *CutterLocationFile
 );

void check_for_equal
(
   int Number1,  int Number2
);
void main
(
  int NumberOfParameters,
  char **ParameterStrings
)
{
check_parameters_three( NumberOfParameters,
ParameterStrings );
information( ParameterStrings );
add_offsets_to_surface( ParameterStrings[1],
ParameterStrings[2], ParameterStrings[3] );
printf( "\n\nPROGRAM BAŞARIYLA
TAMAMLANDI\n" );
}

void information
 (    char **FileNames   )
{
clrscr();
printf( "       ***OFSETEKLE PROGRAMI ***\n");
printf( "     YÜZEY DOSYASININ OKUNMASI\n\n");
printf( "     OFFSET DOSYASININ OKUNMASI\n\n");
printf( " HERİKİ DOSYANIN BİRLİKTE
EKLENMESİ\n\n") ;
printf( " KESİCİ YERİ DOSYASINA SONUÇLARIN
YAZILMASI \n\n");
printf(  "    YÜZEY DOSYA ADI   =  %s\n",
FileNames[1]);
printf(  "    OFFSET DOSYA ADI  =  %s\n",
FileNames[2]);
printf(  "    KESİCİ YERİ DOSYA ADI =  %s\n",
FileNames[3]);
}
void add_offsets_to_surface
(  char  *SurfaceFile, char  *OffsetsFile,
           char  *CutterLocationFile
)
{
  FILE   *fSurfaceFile;
  double SurfaceX,  SurfaceY,   SurfaceZ;
  int    NLinesSurface, NVerticesSurface;
FILE   *fOffsetsFile;
double  OffsetsX, OffsetsY, OffsetsZ;
int    NLinesOffsets, NVerticesOffsets;
FILE  *fCutterLocationFile;
double CutterLocationX, CutterLocationY,
CutterLocationZ;

register int    Line,Vertex;
fSurfaceFile = open_input_file( SurfaceFile );
fOffsetsFile = open_input_file( OffsetsFile );
fCutterLocationFile = open_output_file(
CutterLocationFile );
fscanf(  fSurfaceFile, "%d",   &NLinesSurface );
fscanf(  fOffsetsFile, "%d",   &NLinesOffsets );

check_for_equal( NLinesSurface, NLinesOffsets );
fprintf( fCutterLocationFile, "%d\n",  NLinesSurface);

gotoxy(10,10);
printf("%d. SATIR İŞLENİYOR\n", NLinesSurface );
gotoxy(10,11);
printf( "SATIR İŞLENİLDİ # " );
for (Line=0;Line<NLinesSurface;Line++)
{
gotoxy(27,11);
printf("%4d", Line+1);
fscanf(  fSurfaceFile, "%d",   &NVerticesSurface );
fscanf(  fOffsetsFile, "%d",   &NVerticesOffsets );
check_for_equal( NVerticesSurface, NVerticesOffsets );

fprintf( fCutterLocationFile, "%d\n",  NVerticesSurface );

for( Vertex=0 ; Vertex<NVerticesSurface ; Vertex++)
{
fscanf( fSurfaceFile, "%lf %lf %lf",&SurfaceX,
&SurfaceY, &SurfaceZ );
fscanf( fOffsetsFile, "%lf %lf %lf",&OffsetsX,
&OffsetsY, &OffsetsZ );
CutterLocationX = SurfaceX + OffsetsX;
CutterLocationY = SurfaceY + OffsetsY;
CutterLocationZ = SurfaceZ + OffsetsZ;

fprintf( fCutterLocationFile, "%10.6f %10.6f
%10.6f\n",CutterLocationX, CutterLocationY,
CutterLocationZ );
}
}
}

void check_for_equal
(
  int Number1,   int Number2
)
{
if( Number1 != Number2 )
{
gotoxy( 20, 20 );
printf( "HATA!: YÜZEY VE OFFSET DOSYA
BÜYÜKLÜKLERİ AYNI DEĞİL\n" );
printf( "PROGRAM SONLANDIRILDI\n" );
exit(1);
}
}
```

```
FILE *open_input_file
(               char *Name              )
{
FILE   *fin;fin = fopen( Name, "r" );
                if( fin == NULL )
                {
printf(" DOSYA AÇILAMIYOR : %s\n", Name);
                exit (1);
                }
return ( fin );
}
FILE *open_output_file
(char *Name)
{
FILE   *fout;
                fout = fopen( Name, "w" ) ;
                if(fout == NULL)
                {
printf("DOSYA AÇILAMIYOR : %s\n", Name);
                exit (1);
                }
return ( fout );
}
```

```
void check_parameters_three
                (
                int NumberOfParameters,
                char **FileNames
                )
{
 if (NumberOfParameters!=4)
   {
                clrscr();
                printf("PROGRAM İÇERİSİNDE HATA
%s\n",FileNames[0]);
                printf("İKİ ADET GİRİŞ DOSYA ADINI
GİRMEK ZORUNDASINIZ\n");
                printf("VE BİR TANEDE ÇIKIŞ DOSYA ADI
GİRMEK ZORUNDASINIZ\n");
                printf("KULLANIM >>> %s GİRİŞ
DOSYASI1 GİRİŞ DOSYASI2 ÇIKIŞ DOSYASI\n",
FileNames[0]);
                exit(1);
    }
}
```

EK G

G KODLARINI ÜRETEN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/* GKODYAP.C */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include "library.h"

void information
   (
            char **FileNames
   );
int get_machining_procedure
   (
   void
   );
void make_gcode_for_surface
(
   char *CutterLocationFile,
   char *GCodeFile
);

int make_gcode_forward
 (
          FILE     *fCutterLocationFile,
          FILE     *fGCodeFile,
          float  ZClear,
          float    FeedRate,
          int        CodeLine
 );

int make_gcode_backward
  (
          FILE     *fCutterLocationFile,
          FILE     *fGCodeFile,
          float  ZClear, float FeedRate,
          int   CodeLine
  );
int make_gcode_alternating
   (
          FILE  *fCutterLocationFile,
          FILE  *fGCodeFile,
          float  ZClear, float  FeedRate,int
CodeLine
  );

int header
   (
   FILE     *fGCodeFile,
   float     XStart, float     YStart,   float ZStart,
   int SpindleSpeed,  int   MetricImperial
   );

int make_gcode_forward_one_line
   (
          FILE    *fGCodeFile,
          float  *X, float  *Y, float  *Z,
          int  NVrtcs,  float ZClear, int
CodeLine,
          float FeedRate, int  StartCondition
   );

int make_gcode_backward_one_line
   (
          FILE    *fGCodeFile,
          float  *X, float  *Y, float  *Z,
          int  NVrtcs,  float ZClear,
  int  CodeLine, float FeedRate, int  StartCondition
   );
int gcode_begin_line_g00
   (
          FILE     *fGCodeFile,
          float     X, float     Y, float     Z,
          float     ZClear, float     FeedRate,
          int       CodeLine
   );

int ender
   (
       FILE     *fGCodeFile,
       int     CodeLine,
       float  XStart, float  YStart, float  ZStart, float  ZClear
   );

void read_one_line
   (
       float   *X, float   *Y, float   *Z,
       int       *NVertices,
       FILE     *fin
   );
#define FORWARD 0
#define BACKWARD 1
#define ALTERNATING 2
#define   METRIC 0
#define   IMPERIAL 1
#define MAX_VERTICES 2000
#define G00_START   0
#define G01_START   1

void main
(
    int  NumberOfParameters,
    char **ParameterStrings
)
{
check_parameters_two( NumberOfParameters, ParameterStrings
);
information( ParameterStrings );
make_gcode_for_surface( ParameterStrings[1],
ParameterStrings[2]);
printf( "\n\nPROGRAM BASARIYLA TAMAMLANDI \n" );
}

void information
(
          char **FileNames
)
{
clrscr();
printf( "      *** G KOD ÜRETME PROGRAMI ***\n");
printf( "    TAKIM YOLU IÇEREN DOSYADAN G KOD
ÜRETEN PROGRAM \n\n");
printf( " TAKIM YOLUNU IÇEREN DOSYA ADI= %s\n",
FileNames[1] );
printf( " G-KOD IÇEREN DOSYA        = %s\n",
FileNames[2] );
}
```

```
int get_machining_procedure
(void) .
{
int  Procedure,
    Continue;
char Answer;
gotoxy(1,14);
printf("=====================================\n");
printf ("  BIR PARÇANIN ISLENMESINDE 2
TEMEL YÖNTEM VARDIR \n" );
printf("=====================================\n");
printf("   1. VERTİCELERIN BELIRLI BIR
DÜZENDE TANIMLANMASI (I)LERI   \n");
printf("   2. TERS YÖNDE TANIMLAMA (G)ERI
\n\n");
do
{
gotoxy(10,21);
printf("(I) (G) SEÇINIZ....:? ");
Answer = toupper( getch());
putch(Answer);
switch(Answer)
{
case 'I':
            Procedure = FORWARD;
            gotoxy(10,13);
            printf( "ISLEME YÖNTEMI : ILERI");
            Continue = FALSE;
            break;
case 'G':
            Procedure = BACKWARD;
            gotoxy ( 10, 13 );
            printf( "ISLEME YÖNTEMI : GERI" );
            Continue = FALSE;
            break;
default:
beep();
Continue = TRUE;
break;
}
}while( Continue == TRUE);
gotoxy(1,14);
printf("                                        \n");
printf("                                        \n");
printf("                                        \n");
printf("                                        \n");
printf("                                        \n");
printf("                                        \n");
printf("                                        \n");
return(Procedure);
}

int get_measurement_system
(
void
)
{
int  System,  Continue;
char Answer; gotoxy(1,16);
printf("=====================================\n" );
printf("  KULLANILACAK ÖLÇÜ BIRIMI INÇ
VEYA MILIMETREMI OLACAK? \n" );
printf("=====================================\n");
printf ("         MILIMETRE   (M):\n");
printf ("         INÇ         (I):\n");
do
{
gotoxy(10,20);
printf("TERCIHINIZ..:(M)(I)? ");
Answer = toupper( getch() );
putch ( Answer ) ;
switch( Answer )
case 'M':
            System = METRIC;
            gotoxy(10,14);
            printf( "ÖLÇÜ SISTEMI : MILIMETRE" );
            Continue = FALSE;
            break;
case 'I':
            System = IMPERIAL;
            gotoxy(10,14);
            printf( "ÖLÇÜ SISTEMI : INÇ" );
            Continue = FALSE;
break;
default:
beep ();
Continue = TRUE;
break;
}
}while( Continue == TRUE );

gotoxy(1,15);
printf("                                                \n" );
printf("                                                \n" );
printf("                                                \n" );
printf("                                                \n" );
printf("                                                \n" );
printf("                                                \n" );
return(System);
}
void make_gcode_for_surface
(
            char *CutterLocationFile,
            char *GCodeFile
)
{
            int  Procedure;
            FILE  *fCutterLocationFile,
                *fGCodeFile;
            float ZClear,
                XStart,
                YStart,
                ZStart,
                FeedRate;
            int  SpindleSpeed;
            int  System;
            int  CodeLine;
fCutterLocationFile = open_input_file( CutterLocationFile );
fGCodeFile = open_output_file ( GCodeFile );

gotoxy(10,7);
ZClear = (float) get_one_double_value( " SIFIR NOKTASINA
OLAN MESAFE : " );
gotoxy(10,8);
XStart = (float) get_one_double_value( " X BASLANGIÇ YER
DEGERINI GIRINIZ :");
gotoxy(10,9);
YStart = (float) get_one_double_value( " Y BASLANGIÇ YER
DEGERINI GIRINIZ :");
gotoxy(10,10);
ZStart = (float) get_one_double_value( " Z BASLANGIÇ YER
DEGERINI GIRINIZ :");
gotoxy(10,11);
FeedRate = (float) get_one_double_value( " ILERLEME HIZINI
GIRINIZ(FEEDRATE) :      ");
gotoxy(10,12);
SpindleSpeed = (int) get_one_double_value( "MIL DÖNÜS
HIZINI GIRINIZ(SPINDLE SPEED): ");
Procedure = get_machining_procedure( );

System = get_measurement_system();

CodeLine = header( fGCodeFile, XStart, YStart, ZStart,
                    SpindleSpeed, System );
```

```
switch ( Procedure )
{
case FORWARD:
      CodeLine = make_gcode_forward(
fCutterLocationFile, fGCodeFile,   ZClear,
FeedRate, CodeLine);
                break;
case BACKWARD:
   CodeLine = make_gcode_backward(
fCutterLocationFile, fGCodeFile,   ZClear,
FeedRate, CodeLine );
                break;

}

ender( fGCodeFile, CodeLine, XStart, YStart,
ZStart, ZClear );
}
int make_gcode_forward
(
            FILE  *fCutterLocationFile,
            FILE   *fGCodeFile,
            float  ZClear, float  FeedRate,
            int    CodeLine
)
{
int NLines, Line, NVertices;
float *X, *Y, *Z;
X = (float *) malloc( (unsigned) ( sizeof (float) *
MAX_VERTICES));
Y = (float *) malloc( (unsigned) ( sizeof (float) *
MAX_VERTICES));
Z = (float *) malloc( (unsigned) ( sizeof (float) *
MAX_VERTICES));
if((X==NULL)||(Y==NULL)||(Z==NULL))
{
printf( "HATA : YETERSIZ BELLEK       ");
exit(1);
}
fscanf( fCutterLocationFile, "%d", &NLines );
gotoxy(10,22);
printf("ISLENECEK %d SATIR VAR");
gotoxy(10,23);
printf("SATIRLAR ISLENILIYOR" );
for ( Line=0 ; Line<NLines ; Line++ )
{
gotoxy(25,23);
printf( "%4d", Line+1 );
read_one_line( X, Y, Z, &NVertices,
fCutterLocationFile );
CodeLine = make_gcode_forward_one_line(
fGCodeFile, X, Y, Z,   NVertices, ZClear,
CodeLine, FeedRate, G00_START );
}
return ( CodeLine );
}

int make_gcode_backward
(
     FILE  *fCutterLocationFile,
     FILE. *fGCodeFile,
     float  ZClear,
     float  FeedRate,
     int    CodeLine
)
{
int NLines, Line, NVertices;
float *X, *Y, *Z;
X = (float *)  malloc( (unsigned) ( sizeof(float) *
MAX_VERTICES ) );
Y = (float *)  malloc( (unsigned) ( sizeof(float) *
MAX_VERTICES ) );
Z = (float *)  malloc( (unsigned) ( sizeof(float) *
MAX_VERTICES ) );
if((X==NULL)||(Y==NULL)||(Z==NULL))
{
printf("HATA : YETERSIZ BELLEK" );
exit(1);
}
fscanf( fCutterLocationFile, "%d", &NLines );
gotoxy(10,20);
printf("ISLENECEK %d SATIR VAR");
gotoxy(10,21 ) ;
printf("SATIRLAR ISLENILIYOR" );
/* LOOP THROUGH ALL LINES  */
for ( Line=0 ; Line<NLines ; Line++ )
{
gotoxy(25,21);
printf( "%4d", Line+1 );
read_one_line( X, Y, Z, &NVertices, fCutterLocationFile );
CodeLine = make_gcode_backward_one_line( fGCodeFile, X,
Y, Z,  NVertices, ZClear, CodeLine, FeedRate, G00_START );
}
return ( CodeLine );
}

int make_gcode_alternating
(
    FILE  *fCutterLocationFile,
    FILE  *fGCodeFile,
    float  ZClear,
    float  FeedRate,
    int    CodeLine
)
{
          int NLines, Line, NVertices;
          float *X, *Y, *Z;

X = (float *) malloc( (unsigned) ( sizeof (float) *
MAX_VERTICES ) );
Y = (float *) malloc( (unsigned) ( sizeof (float) *
MAX_VERTICES ) );
Z = (float *) malloc( (unsigned) ( sizeof (float) *
MAX_VERTICES ) );
if( (X==NULL) || (Y==NULL) || (Z==NULL) )
{
printf( "HATA : YETERSIZ BELLEK" );
exit(1);
}
fscanf( fCutterLocationFile, "%d", &NLines );
gotoxy(10,20);
printf( "ISLENECEK %d SATIR VAR" );
gotoxy(10,21);
printf( "SATIRLAR ISLENIYOR" );
Line=0;
gotoxy(25,21);
printf( "%4d", Line+1 );
read_one_line( X, Y, Z, &NVertices, fCutterLocationFile );

CodeLine = make_gcode_forward_one_line( fGCodeFile, X, Y,
Z,NVertices,ZClear, CodeLine, FeedRate, G00_START );
Line++;
while( Line < NLines )
{
gotoxy(25,21);
printf( "%4d", Line+1);
read_one_line( X, Y, Z, &NVertices, fCutterLocationFile );

CodeLine = make_gcode_backward_one_line( fGCodeFile, X,
Y, Z,NVertices, ZClear, CodeLine, FeedRate, G01_START );
Line++;
if( Line < NLines )
{
gotoxy(25,21);
printf( "%4d", Line+1);
```

```
read_one_line( X, Y, Z, &NVertices,
fCutterLocationFile );
make_gcode_forward_one_line( fGCodeFile, X, Y,
Z, NVertices, ZClear, CodeLine, FeedRate,
G01_START );
Line++;
}
}
return ( CodeLine );
}

int header
(
        FILE  *fGCodeFile,
        float XStart,float YStart,float ZStart,
        int   SpindleSpeed, int   MetricImperial
)
{
fprintf(fGCodeFile, "N1 O23;\n" );
fprintf(fGCodeFile, "N2 G92 X%5.3f Y%5.3f
Z%5.3f;\n", XStart, YStart, ZStart );

if( MetricImperial == METRIC )
{
 fprintf(fGCodeFile, "N3 G21;\n" );
}
else
{
 fprintf(fGCodeFile, "N3 G20;\n" );
}

fprintf(fGCodeFile, "N4 G90;\n" );
fprintf(fGCodeFile, "N5 T06;\n" );
fprintf(fGCodeFile, "N6 S%4d M03;\n",
SpindleSpeed );
return(7);
}

int ender
(
FILE  *fGCodeFile,
int  CodeLine, float XStart,
float YStart, float ZStart, float ZClear
)
{
fprintf(fGCodeFile, "N%d G00 Z%6.4f;\n",
CodeLine++, ZClear );
fprintf(fGCodeFile, "N%d G00 X%6.4f
Y%6.4f;\n", CodeLine++, XStart, YStart );
fprintf(fGCodeFile, "N%d G00
Z%6.4f;\n",CodeLine++, ZStart );
fprintf(fGCodeFile, "N%d M05;\n",CodeLine++ );
fprintf(fGCodeFile, "N%d M30;\n",CodeLine );
fprintf(fGCodeFile, "%\n");
}

int gcode_begin_line_g00
(
        FILE    *fGCodeFile,
        float   X,float    Y,float    Z,
        float   ZClear,float    FeedRate,
        int       CodeLine
)
{
fprintf(fGCodeFile, "N%d G00
Z%6.4f;\n",CodeLine++, ZClear );
fprintf (fGCodeFile, "N%d G00 X%6.4f
Y%6.4f;\n", CodeLine++, X, Y );
fprintf (fGCodeFile,"N%d G01 Z%6.4f F%3.1f;\n",
CodeLine++, Z, FeedRate);
return ( CodeLine );
}

int make_gcode_forward_one_line
(
        FILE *fGCodeFile,
        float *X,  float *Y,  float *Z,
        int  NVrtcs, float  ZClear,
        int  CodeLine, float FeedRate,
        int   StartCondition
)
{
register int Vrtx;
if( CodeLine > 9000 )
{
CodeLine = 1;
}
Vrtx = 0;
if( StartCondition == G00_START )
{
CodeLine = gcode_begin_line_g00( fGCodeFile, X[Vrtx],
Y[Vrtx], Z[Vrtx],  ZClear, FeedRate, CodeLine );
}
else
{
fprintf( fGCodeFile, "N%d G01 X%6.4f Y%6.4f Z%6.4f;\n",
    CodeLine++, X[Vrtx], Y[Vrtx],Z[Vrtx] );
}
Vrtx++;
for( ; Vrtx<NVrtcs ; Vrtx++ )
{
fprintf(fGCodeFile, "N%d G01 X%6.4f Y%6.4f Z%6.4f;\n",
            CodeLine++, X[Vrtx], Y[Vrtx], Z[Vrtx] );
}
return ( CodeLine );
}
int make_gcode_backward_one_line
(
FILE  *fGCodeFile,
float *X, float *Y, float *Z, int  NVrtcs, float  ZClear,
        int  CodeLine, float FeedRate,  int  StartCondition
)
{
register int Vrtx;
UP 9999      */
if( CodeLine > 9000 )
{
CodeLine=1;
}
Vrtx = NVrtcs-1;
if( StartCondition == G00_START )
{
CodeLine = gcode_begin_line_g00( fGCodeFile, X[Vrtx],
Y[Vrtx], Z[Vrtx],  ZClear,  FeedRate, CodeLine );
}
else
{
fprintf( fGCodeFile, "N%d G01 X%6.4f Y%6.4f Z%6.4f;\n",
    CodeLine++, X[Vrtx], Y[Vrtx],Z[Vrtx] );
}
Vrtx--;
for ( ; Vrtx>=0 ; Vrtx-- )
{
fprintf(fGCodeFile, "N%d G01 X%6.4f Y%6.4f Z%6.4f;\n",
            CodeLine++, X[Vrtx],Y[Vrtx],Z[Vrtx] );
}
return ( CodeLine );
}

void read_one_line
(
float   *X, float   *Y, float   *Z, int     *NVertices,
FILE    *fin
)
{
register int   Vertex;
```

```
fscanf( fin, "%d", NVertices );
for( Vertex=0 ; Vertex<*NVertices ; Vertex++)
{
fscanf( fin, "%f %f %f",
&(X[Vertex]),&(Y[Vertex]),&(Z[Vertex]) );
}
}

FILE *open_input_file
        (
        char *Name
        )
{
FILE   *fin;
        fin = fopen( Name, "r" );
        if( fin == NULL )
        {
        printf(" DOSYA AÇILAMIYOR :
%s\n", Name);
                exit (1);
        }
return ( fin );
}

FILE *open_output_file
        (
        char *Name
        )
{
FILE   *fout;
        fout = fopen( Name, "w" ) ;
        if(fout == NULL)
        {
        printf("DOSYA AÇILAMIYOR :
%s\n", Name);
        exit (1);
        }

return ( fout );
}

void beep
        (
        void
        )
{
sound(1000);
delay(100);
nosound();
}
```

```
double get_one_double_value
        (
        char *Question
        )

{
        double Answer;
        int  NfieldsRead;
printf("%s", Question);
do
{
NfieldsRead=scanf("%lf",&Answer);
if (NfieldsRead !=1)
  {
  beep();
  }
}while (NfieldsRead !=1);
printf("\n");
return (Answer);
}

void check_parameters_two
        (
        int NumberOfParameters,
        char **FileNames
        )
{
 if (NumberOfParameters!=3)
   {
        clrscr();
        printf("PROGRAM IÇERISINDE HATA
%s\n",FileNames[0]);
        printf("GIRIS VE ÇIKIS DOSYA ADLARINI
DOGRU OLARAK GIRMEK ZORUNDASINIZ\n");
        printf("KULLANIM  >>> %s GIRIS DOSYASI
ÇIKIS DOSYASI\n", FileNames[0]);
        exit(1);
   }
}
```

EK Ğ

YÜZEY NORMAL VEKTÖRÜNÜ TERSYÜZ EDEN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*                TERSYUZ.C                */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
 #include <library.h>
void information
   (char **FileNames    );
void invert_normal_vector
   (
          char  *InputFile,
          char  *OutputFile
   );
void main (
          int NumberOfParameters,
          char **ParameterStrings
   )
{
check_parameters_two( NumberOfParameters,
ParameterStrings );
information( ParameterStrings );
invert_normal_vector( ParameterStrings[1],
ParameterStrings [2] );
printf( "\n\nPROGRAM BAŞARIYLA
TAMAMLANDI\n" );
}

void information
   (
       char **FileNames
   )
{
clrscr( );
printf ( "     *** T E R S Y Ü Z  P R O G R A M I
***\n");
printf( "      YÜZEY NORMAL VEKTÖRÜNÜN
TERSYÜZ EDİLMESİ\n\n");
printf ( " GİRİŞ DOSYA ADI = %s\n",
FileNames[1] );
printf ( " ÇIKIŞ DOSYA ADI = %s\n",
FileNames[2] );
}
void invert_normal_vector
  (
    char  *InputFile,  char  *OutputFile
  )
{
  register int   Line,   Vertex;
   FILE   *fIn, *fOut;
   double NX, NY, NZ;
   int    NLines,NVertices;
fIn = open_input_file(InputFile);
fOut= open_output_file(OutputFile);
fscanf(fIn,"%d", &NLines );
fprintf(fOut,"%d\n", NLines);
gotoxy(10,10);
printf("%d. SATIR İŞLENİYOR\n", NLines);
gotoxy(10,11);
printf( "İŞLENİLEN SATIR # " );
for( Line=0 ; Line<NLines ; Line++ )
{
gotoxy(27,11); printf("%4d", Line+1);
fscanf( fIn, "%d",  &NVertices );
fprintf( fOut,"%d\n",  NVertices );
for ( Vertex=0 ; Vertex<NVertices ; Vertex++)
{
fscanf( fIn, "%lf %lf %lf", &NX, &NY, &NZ );
fprintf( fOut, "%10.6lf %10.6lf %10.6lf\n", -NX, -NY, -NZ );
}} }

FILE *open_input_file
(char *Name          )
{
FILE   *fin;
          fin = fopen( Name, "r" );
          if( fin == NULL )
          {
          printf(" DOSYA AÇILAMIYOR : %s\n", Name);
                    exit (1);
          }
return ( fin );
}

FILE *open_output_file
(         char *Name          )
{
FILE   *fout;

          fout = fopen( Name, "w" ) ;
          if(fout == NULL)
          {
          printf("DOSYA AÇILAMIYOR : %s\n", Name);
          exit (1);
          }
return ( fout );
}

void beep  ( void )
{
sound(1000);
delay(100);
nosound();
}
void check_parameters_two
          (
          int NumberOfParameters,
          char **FileNames
          )
{
 if (NumberOfParameters!=3)
  {
          clrscr();
          printf("PROGRAM İÇERİSİNDE HATA
%s\n",FileNames[0]);
          printf("GİRİŞ VE ÇIKIŞ DOSYA ADLARINI
DOĞRU OLARAK GİRMEK ZORUNDASINIZ\n");
          printf("KULLANIM >>> %s GİRİŞ DOSYASI
ÇIKIŞ DOSYASI\n", FileNames[0]);
          exit(1);
   }
}
```

EK H

YÜZEY YOĞUNLUĞUNU AZALTAN PROGRAMA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*        AZALT . C      */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <library.h>

#define SKIP TRUE
#define NO_SKIP FALSE

void information
 (
    char **FileNames
 );

void skip (
 char  *InputFile,  char  *OutputFile,
 int   SkipNLines,  int   SkipNVrtcs
   );

int how_many_new_values
   (
          int NOldValues, int NSkip
   );

int skip_this_one    (
          int LastNoSkip,  int Now,
          int NSkip,  int NValues     );

void lines_and_vertices_to_skip
    ( int *SkipNLines,  int *SkipNVrtcs
    );

void main
(
          int NumberOfParameters,
          char **ParameterStrings
)
{
   int SkipNLines,     SkipNVrtcs;

check_parameters_two( NumberOfParameters,
ParameterStrings);
information( ParameterStrings );
lines_and_vertices_to_skip( &SkipNLines,
&SkipNVrtcs );
skip( ParameterStrings[1], ParameterStrings[2],
SkipNLines, SkipNVrtcs );
printf( "\n\nPROGRAM BAŞARIYLA
TAMAMLANDI\n" );
}

void information
(
   char **FileNames
)
{
clrscr();
printf( "       *** A Z A L  T M A  P R O G R A M
I ***\n");
printf ( " İSTENİLEN SATIR VE VERTİCENİN
ATLANILMASI \n\n");
printf( "  GİRİŞ DOSYA ADI = %s\n", FileNames[1] );
printf ( " ÇIKIŞ DOSYA ADI = %s\n", FileNames[2] );
}

void lines_and_vertices_to_skip
(
int *SkipNLines, int *SkipNVrtcs
)
{
char Answer = 'E';
do
{
gotoxy(10,10);
printf ( "                                                  " );
gotoxy(10,10);
printf( "KAÇ SATIR İPTAL EDİLECEK? " );
scanf( "%d", SkipNLines );
gotoxy(10,10);
printf( "KAÇ SÜTUN İPTAL EDİLECEK? " );
scanf( "%d", SkipNVrtcs );
gotoxy ( 10, 10 );
printf ( "%3d SATIR VE  %d VERTICE İPTALEDİLECEK",
*SkipNLines,*SkipNVrtcs );
gotoxy( 10, 11 );
printf ( "BİLGİ DOĞRU MU(E/H)?" );
Answer = toupper( getch() );
gotoxy( 10, 11 );
printf ( "                                                  " );

}
while( Answer != 'E' );
}
void   skip
(
char     *InputFile,  char     *OutputFile,
int      SkipNLines, int       SkipNVrtcs
)
{
register int   Line, Vrtx;
FILE  *fIn, *fOut;

double X, Y, Z;
int    NOldLines, NOldVrtcs;
int    NNewLines, NNewVrtcs;
int    LastLineWritten = -9999,
          LastVrtxWritten = -9999;
fIn = open_input_file(  InputFile );
fOut = open_output_file( OutputFile );
fscanf(fIn,"%d",&NOldLines);
NNewLines = how_many_new_values( NOldLines, SkipNLines
);
fprintf( fOut, "%d\n",  NNewLines );
gotoxy(10,13);
printf ("%d SATIR İŞLENİLİYORn", NOldLines);
gotoxy ( 10, 14 );
printf ( "İŞLENİLEN SATIR #");
for( Line=0 ; Line<NOldLines ; Line++ )
{
gotoxy(27,14);
printf("%4d", Line+1);
```

```
fscanf( fIn, "%d",  &NOldVrtcs );
if( skip_this_one( LastLineWritten, Line,
SkipNLines, NOldLines) == SKIP )
{
for( Vrtx=0 ; Vrtx<NOldVrtcs ; Vrtx++)
{
fscanf( fIn, "%lf %lf %lf", &X, &Y, &Z );
}
}
else
{
LastLineWritten = Line;

NNewVrtcs = how_many_new_values (
NOldVrtcs, SkipNVrtcs );
fprintf( fOut,"%d\n",NNewVrtcs);
for( Vrtx=0 ; Vrtx<NOldVrtcs ; Vrtx++ )
{
fscanf( fIn, "%lf %lf %lf", &X, &Y, &Z );
if( skip_this_one( LastVrtxWritten, Vrtx,
SkipNVrtcs, NOldVrtcs )==NO_SKIP )
{
LastVrtxWritten = Vrtx;
fprintf( fOut, "%10.6lf %10.6lf %10.6lf\n", X, Y, Z
);
}
}
LastVrtxWritten = -9999;
}
}
}

int how_many_new_values
(
        int NOldValues,
        int NSkip
)

{
register int n;
int NNewValues;
int LastNoSkip = -9999;
NNewValues = 0;

for( n=0 ; n<NOldValues ; n++ )
{
if( skip_this_one( LastNoSkip, n, NSkip,
NOldValues ) != SKIP )
{
NNewValues++;
LastNoSkip = n;
}
}
return( NNewValues );
}

int skip_this_one (
  int LastNoSkip,   int Now,   int NSkip,   int
NValues
)
{
if ( Now > (LastNoSkip+NSkip))
{
return ( NO_SKIP ) ;
}

else
if( Now == (NValues-1) )
{
return( NO_SKIP );
}
else
{
return ( SKIP );
}
}

FILE *open_input_file
        (
        char *Name
        )
{
FILE   *fin;
        fin = fopen( Name, "r" );
        if( fin == NULL )
        {
        printf(" DOSYA AÇILAMIYOR : %s\n", Name);
                exit (1);
        }
 return ( fin );
}
FILE *open_output_file
        (
        char *Name
        )
{
FILE   *fout;
        fout = fopen( Name, "w" ) ;
        if(fout == NULL)
        {
        printf("DOSYA AÇILAMIYOR : %s\n", Name);
        exit (1);
        }
return ( fout );
}


void check_parameters_two
        (
        int NumberOfParameters,
        char **FileNames
        )

{
  if (NumberOfParameters!=3)
    {
clrscr();
printf("PROGRAM İÇERİSİNDE HATA %s\n",FileNames[0]);
printf("GİRİŞ VE ÇIKIŞ DOSYA ADLARINI DOĞRU
OLARAK GİRMEK ZORUNDASINIZ\n");
printf("KULLANIM  >>> %s GİRİŞ DOSYASI ÇIKIŞ
DOSYASI\n", FileNames[0]);
exit(1);
    }
}
```

EK I

YÜZEYİN UZATILMASINI SAĞLAYAN PROGRAMA AİT VERİ KODLARI .

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*          EKLE.C       */

#include <stdio.h>
#include <conio.h>
#include "library.h"

void information
  (
          char **FileNames
  );

void add_ends_to_surface
  (
char      *SurfaceFile, char      *ExtendedFile,
double    DistanceStart, double    DistanceEnd
  );
void  make_new_vertex
  (
float     XStart, float     XEnd, float     YStart,
float     YEnd, float     ZStart, float     ZEnd,
float    *ExtX,  float    *ExtY,  float    *ExtZ,
float     NewXLoc
  );

void main
(
    int  NParameters,    char **ParameterStrings
)
{
double DistanceStart, DistanceEnd;
check_parameters_two( NParameters,
ParameterStrings );
information( ParameterStrings );

DistanceStart = get_one_double_value(
"BAŞLANGIÇ SATIRINA NE KADAR
EKLENECEK :?");
DistanceEnd  = get_one_double_value( "BİTİŞ
SATIRINA NE KADAR EKLENECEK.:?" );
add_ends_to_surface( ParameterStrings[1],
ParameterStrings[2], DistanceStart, DistanceEnd );
gotoxy( 10, 22 );
printf( "PROGRAM BAŞARIYLA
TAMAMLANDI\n" );
}

void information
(
    char **FileNames
)
{
clrscr();
printf ( "      *** YÜZEY UZATMA PROGRAMI
***\n");
printf( " SON SATIRA VERTICE EKLEYEREK
BİR YÜZEYİN GENİŞLETİLMESİn\n");
printf( "  UZATILMAMIŞ YÜZEY DOSYA ADI =
%s\n", FileNames[1] );
printf( "    UZATILMIŞ DOSYA ADI = %s\n",
FileNames[2] );
}

void   add_ends_to_surface
(
char  *SurfaceFile,   char *ExtendedFile,
double  DistanceStart, double  DistanceEnd
)
{
FILE   *fSurfaceFile;
     double  X[2], Y[2], Z[2];
int    NLines, NVrtcs;
FILE   *fExtendedFile;
float  ExtX,ExtY,ExtZ;
register int    Line,Vertex;
fSurfaceFile = open_input_file( SurfaceFile );
fExtendedFile= open_output_file( ExtendedFile );
fscanf(fSurfaceFile, "%d",   &NLines);
fprintf( fExtendedFile,"%d\n",NLines);
gotoxy(10,17);

printf("%d SATIR İŞLENECEK\n", NLines );
gotoxy(10,18);
printf( "SATIR İŞLENİLDİ # " );
for( Line=0 ; Line<NLines ; Line++ )
{
gotoxy(27,18);
printf("%4d", Line+1);

fscanf (  fSurfaceFile, "%d",   &NVrtcs );
fprintf( fExtendedFile, "%d\n",  NVrtcs+2 );
fscanf ( fSurfaceFile, "%lf %lf %lf", &(X[0]), &(Y[0]),
&(Z[0]));
fscanf ( fSurfaceFile, "%lf %lf %lf", &(X[1]), &(Y[1]),
&(Z[1]));

make_new_vertex( X[0], X[1], Y[0], Y[1], Z[0], Z[1],
                 &ExtX, &ExtY, &ExtZ,
                 X[0] + DistanceStart);

fprintf( fExtendedFile, " %9.6f %9.6f %9.6f\n", ExtX, ExtY,
ExtZ);
fprintf( fExtendedFile, " %9.6f %9.6f %9.6f\n", X[0], Y[0],
Z[0]);

fprintf( fExtendedFile, " %9.6f %9.6f %9.6f\n", X[1], Y[1],
Z[1]);
for( Vertex=2 ; Vertex< (NVrtcs-1) ; Vertex++)
{
fscanf(fSurfaceFile,"%lf %lf %lf", &(X[0]), &(Y[0]), &(Z[0]));
fprintf( fExtendedFile," %9.6f %9.6f %9.6f\n",X[0],Y[0],Z[0]);
}

fscanf(fSurfaceFile,"%lf  %lf  %lf", &(X[1]), &(Y[1]),&(Z[1]));
fprintf(fExtendedFile,"%9.6f %9.6f %9.6f\n", X[1],Y[1],Z[1]);

make_new_vertex(X[0],X[1],Y[0],Y[1], Z[0],Z[1],
          &ExtX,&ExtY,&ExtZ,X[1]+DistanceEnd );
fprintf(fExtendedFile,"%9.6f %9.6f %9.6f\n",ExtX,ExtY,ExtZ);
}
}
void  make_new_vertex
   (
float     XStart, float     XEnd, float     YStart, float YEnd,
float     ZStart, float     ZEnd, float    *ExtX, float    *ExtY,
float    *ExtZ, float     NewXLoc    )
{
```

```
double MY,MZ; double BY, BZ;

MY = (YEnd-YStart) / (XEnd-XStart);
MZ = (ZEnd-ZStart) / (XEnd-XStart);

BY = YStart - (XStart * MY);
BZ = ZStart - (XStart * MZ);

*ExtX = NewXLoc;
*ExtY = BY + *ExtX * MY;
*ExtZ = BZ + *ExtX * MZ;
}
FILE *open_input_file
        (
        char *Name
        )
{
FILE   *fin;
        fin = fopen( Name, "r" );
        if( fin == NULL )
        {
        printf(" DOSYA AÇILAMIYOR : %s\n",
Name);
        exit (1);
        }
return ( fin );
}

FILE *open_output_file
        (
        char *Name
        )
{
FILE   *fout;
        fout = fopen( Name, "w" ) ;
        if(fout == NULL)
        {
        printf("DOSYA AÇILAMIYOR : %s\n",
Name);
        exit (1);
        }
return ( fout );
}
```

```
double get_one_double_value
        (
        char *Question
        )
{
        double Answer;
        int  NfieldsRead;

printf("%s", Question);
do
{
NfieldsRead=scanf("%lf",&Answer);
if (NfieldsRead !=1)
  {
  beep();
  }
}while (NfieldsRead !=1);
printf("\n");
 return (Answer);
}

void check_parameters_two
        (
        int NumberOfParameters,
        char **FileNames
        )
{
 if (NumberOfParameters!=3)
   {
        clrscr();
printf("PROGRAM İÇERİSİNDE HATA %s\n",FileNames[0]);
printf("GİRİŞ VE ÇIKIŞ DOSYA ADLARINI DOĞRU
OLARAK GİRMEK ZORUNDASINIZ\n");
printf("KULLANIM  >>> %s GİRİŞ DOSYASI ÇIKIŞ
DOSYASI\n", FileNames[0]);
exit(1);
   }
}

void beep  (void)

{
sound(1000);
delay(100);
nosound();
}
```

EK İ

SİMÜLASYON PROGRAMINA AİT VERİ KODLARI

(C PROGRAMLAMA DİLİNDE HAZIRLANMIŞTIR)

```
/*          PROGRAM  CAM.C
*/
/* Bu program Borland grafik kütüphanesi
(GRAPHICS.LIB)    */
/* kullanılarak yazılmıştır. Uygun fontlar veekran sürücü
*/
/* grafik ortamını destekliyor olması gerekmektedir.
/* Aşağıdaki dosyalar grafik kütüphanesi içerisinde
program */ /*çalıştırıldığı zaman  mutlaka
yönlnedirilmelidir. */
/* TRIP.CHR    - Karakter yazı dosyası */
/* SANS.CHR    - Karakter yazı dosyası */
/* EGAVGA.BGI  - grafik sürücü dosyası */
/* Not: Bu program VGA veya EGA ekran kartı olan bir
makinede çalışacaktır */

/*  Kütüphanelerin Açılması      */
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <io.h>
#include <fcntl.h>
#include <mem.h>
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include <math.h>
#include <graphics.h>
#include "cam.h"
#define  BLK_SIZE 513
void  flagreset(int *mouseStatusPtr, int
*numberOfButtonsPtr)
{
        union REGS regs;
        regs.x.ax=0;
        int86(MOUSE, &regs,&regs);
        *mouseStatusPtr = regs.x.ax;
        *numberOfButtonsPtr=regs.x.bx;
}
void showcursor(void)
{
        union REGS regs;
        regs.x.ax=1;
        int86(MOUSE, &regs, &regs);
}
void hidecursor(void)
{
        union REGS regs;
        regs.x.ax=2;
        int86 (MOUSE, &regs, &regs);
}
void getposbut (int *buttonStatus, int *horizontal, int
*vertical)
{
        union REGS regs;
        regs.x.ax = 3;
        int86 (MOUSE, &regs, &regs);
        *buttonStatus = regs.x.bx;
        *horizontal  =  regs.x.cx;
        *vertical  =  regs.x.dx;
}
void setcursorpos(int horizontal, int vertical)
{
        union REGS regs;
        regs.x.ax = 4;
        regs.x.cx = horizontal;
        regs.x.dx = vertical;
        int86(MOUSE, &regs, &regs);
}
void  sethorizontallimits(int minPos, int maxPos)
{
        union REGS regs;
        regs.x.ax = 7;
regs.x.cx = minPos;regs.x.dx = maxPos;
        int86 (MOUSE, &regs, &regs);
}
void setverticallimits(int minPos, int maxPos)
{
union REGS regs; regs.x.ax = 8;
regs.x.cx = minPos;   regs.x.dx = maxPos;
 int86(MOUSE, &regs, &regs);
}

int     foreground,background;
int     mouse_flag,loop_flag;
int     scale_flag; int     machine;
int     num_pnts; int     move_color;
int     cut_color; int     xasp,yasp;
int     speed;
float  xyratio;
char   single_step; char   dgcode[6][40];
char   blank[40];

struct pnt
{
int tool; float x,y,z,xz,xiso,yiso; float xwc,ywc,zwc;
};
struct pnt point[max_pnts];
FILE *instream; FILE   *outstream; FILE   *errorfile;
char    gcode[6]; char    last_gcode[6];
char    X[15],Y[15],Z[15]; char I[15],J[15] ,K[15]
,R[15];
char   spindle[15], feedrate[15], toolnum[15],
        gcodefile[12],gcodest[40];
float  xmax,ymax,zmax,xmin,ymin,zmin;
float  xprev,yprev,zprev;
int    error_flag, spin_flag,  tool_flag, feed_flag, strt_flag,
        g_flag,stop_flag, pend_flag,  g92_flag,
gz_flag,
        g17_flag,  g18_flag,  g19_flag,  g20_flag,
        g90_flag,  initial_flag, counter,
err_array[2][10];
void main()
{
        char *s[15][15]; int  h,v,i,a; int  status,
numofbut;
        i=0;
        s[i][0] = "SISTEM";
        s[i][1] = "DOS KOMUT ISTEMI ";
        s[i][2] = "SIMÜLASYON      ";
        s[i][3] = "                ";
        s[i][4] = "PROGRAMDAN ÇIKIS";
        s[i][5] = "";
        i = 1;
        s[i][0] = "TAKIMLAR ";
        s[i][1] = "TORNA    ";
        s[i][2] = "FREZE    ";
        s[i][3] = "";
        i = 2;
        s[i][0] = "GÖRÜNÜSLER    ";
```

```
            s[i][1] = "ÜST          ";
            s[i][2] = "ÇAPRAZ       ";
            s[i][3] = "YAN          ";
            s[i][4] = "TÜM GÖRÜNÜSLER ";
            s[i][5] = "";
            i = 3;
            s[i][0] = "RENKLER  ";
            s[i][1] = "KESME    ";
            s[i][2] = "ILERLEME ";
            s[i][3] = "";
            i = 4;
            s[i][0] = "HIZ ";
            s[i][1] = "HIZ AYARI ";
            s[i][2] = "TEK HIZ   ";
            s[i][3] = "";

i = 5;
s[i][0] = "";
mouse_flag = FALSE;loop_flag  = TRUE;
scale_flag = FALSE;machine = MILL;num_pnts  = 0;
move_color = WHITE;cut_color  = RED;speed=
0;single_step = 'H';
blank[0]  = 32; blank[1]  = 0;
flagreset(&status,&numofbut);
if (status == -1)
{
mouse_flag = TRUE; sethorizontallimits(2, 317);
setverticallimits(131, 341);
}
for (i=0 ; i<=5 ; i++)   strcpy(dgcode[i]," ");
init_graphics();
getaspectratio(&xasp,&yasp);
xyratio = 1.29;
background = BLUE;foreground = WHITE;
set_menu(s);title_screen();set_screen();
 h=0; v=0;
 do
 {
   a = get_key();
   if (a == RETURN)
            a = ESC;
   switch(a)
   {
          case ESC:
             use_menu(s,&h,&v,&a);
             if (a == 1)
                    {
                     switch(h)
                     {
                              case 0:
                              system_sub(v);
                              if (v == 1)
                              set_menu(s);
                                       break;
                              case 1:
                              machine_sub(v);
                                       break;
                              case 2:
                              views_sub(v);
                                       break;
                              case 3:
                              colors_sub(v);
                                       break;
                              case 4:
                              speed_sub(v);
                                       break;
                      }
                    }
             break;
          }
   }
  while (loop_flag == TRUE);
closegraph();
}
void system_sub(int choice)
{
          switch(choice)
          {
           case 1:
  if (scale_flag == FALSE)   doscommand();
            break;
           case 2:
             if (scale_flag == FALSE)
             {
                      fileread(); firstpass();
             }
             break;
           case 3:
             break;
           case 4:
             loop_flag = FALSE;
             break;
          }
    return;
}
void machine_sub(int choice)
{
          switch(choice)
          {
                    case 1:
                    if (scale_flag == FALSE)
                    {
          machine = LATHE;   setcolor (BLUE);
          setfillstyle(1,BLUE);
          bar3d(495,189,550,199,0,1);
          setcolor(WHITE); outtextxy(495,189,
"TORNA");
                    }
                    break;
                    case 2:
                     if (scale_flag==FALSE)
                     {
          machine = MILL; setcolor(BLUE);
          setfillstyle(1,BLUE);
          bar3d(495,189,550,199,0,1);
          setcolor(WHITE); outtextxy(495,189,
"FREZE");
                     }
                    break;
          }
return;
}

void views_sub (int choice)
{
  setviewport(1,21,638,348,1);
  clearviewport ();
  switch(choice)
  {
          case 1:
          top_view();set_view(FULL);
          break;
          case 2:
          iso_view();set_view(FULL);
          break;
          case 3:
          side_view(); set_view(FULL);
          break;
          case 4:
          set_screen();
if (scale_flag == TRUE)
          {
          set_view(TOP);
          outtextxy(110,75,"LÜTFEN BEKLEYIN...");
          scale_path(FALSE);  setcolor(BLACK);
```

```
            bar3d(110,75,270,85,0,1); scale_flag =
FALSE;
            }
            d_path();  d_gcode(blank);
            break;
            }
   return;
}


void colors_sub(int choice)
    {
setfillstyle(1,BLACK);
            switch(choice)
            {
                        case 1:
                           if (scale_flag == TRUE)
                {
input_int(&cut_color,"KESIM IÇIN BIR RENK GIRIN:
","99",400,30,4);setcolor(BLACK);
bar3d(390,25,638,47,0,1);
                }
                else
                {
set_view(GCODE);   setfillstyle(1,BLUE);
setcolor (BLUE); bar3d(0,0,317,73,0,1);
input_int(&cut_color,"KESIM IÇIN BIR RENK GIRIN:
", "99", 50, 30, 4);setfillstyle (1,BLUE); setcolor(BLUE);
bar3d(0,0,317,73,0,1); d_gcode(blank);
    }
                           break;
            case 2:
                if (scale_flag == TRUE)
                        {
input_int(&move_color, HAREKET IÇIN BIR RENK
GIRIN: ","99",390,30,4);   setcolor(BLACK);
bar3d(380,25,638,47,0,1);
                        }
                        else
                        {
            set_view(GCODE); setfillstyle(1,BLUE);
            setcolor(BLUE); bar3d(0,0,317,73,0,1);
            input_int(&move_color,"HAREKET IÇIN BIR
RENK GIRINIZ: ","99",40,30,4);setfillstyle(1,BLUE);
            setcolor(BLUE);bar3d(0,0,317,73,0,1);
            d_gcode(blank);
                        }
                        break;
                        }
setcolor(WHITE); set_view(FULL);
return;
            }

void speed_sub(int choice)
{
            if (scale_flag == FALSE)
            {
set_view(GCODE);   setfillstyle(1,BLUE);
setcolor (BLUE); bar3d(0,0,317,73,0,1);
            switch(choice)
            {
            case 1:
input_int(&speed,"HIZ GIRINIZ (HIZLI(0)-
YAVAS(999)): ","999",14,30,4);
                        break;
            case 2:
                        do
                        {
input_str(&single_step,"SABIT HIZ (E/H)? ","A", 65,
30,4);
                        if (single_step >= 'a' &&
single_step <= 'z')
                                                single_step =
single_step - 'a' + 'A';
                        }

            while ( (single_step != 'E') && (single_step !=
'H') );
                        break;
            }
setfillstyle(1,BLUE); setcolor (BLUE);
bar3d(0,0,317,73,0,1);
d_gcode(blank) ;setcolor(WHITE);set_view(FULL);
            }
 return;
}
void init_graphics(void)
{
            int  gdriver = EGA,   gmode  = EGAHI;
            if (registerbgidriver( EGAVGA_driver)<0)
            {
printf( "UYGUN EKRAN KARTI BULUNAMADI\n" );
            exit(1);
            }
            initgraph(&gdriver,&gmode,"");
            if (registerbgifont(triplex_font) < 0 )
            {
printf ( "GEÇERLI YAZI FONTU BULUNAMADI" );
            exit(1);
            }
if (registerbgifont(sansserif_font) < 0 )
{
printf ( "GEÇERLI YAZI FONTU BULUNAMADI" );
exit(1);
}
return;
}
void title_screen()
{
struct textsettingstype oldset; bar3d(0,20,639,349,0,1);
gettextsettings(&oldset); setcolor(YELLOW);
settextjustify(LEFT_TEXT, TOP_TEXT);
settextstyle(TRIPLEX_FONT, HORIZ_DIR, 6);
outtextxy(65,80,"G-KOD SIMÜLASYONU");
settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 3) ;
settextjustify(oldset.horiz, oldset.vert);
settextstyle(oldset.font, oldset.direction, oldset.charsize);
setcolor(WHITE);bar3d(210,308,470,324,0,1);
outtextxy(213,313, " BASLAMAK IÇIN TUSA
BASINIZ ");
get_key();
    return;
}
void set_screen()
{
setviewport(1,21,638,348,1);clearviewport();
set_view(FULL);
setcolor(WHITE);setfillstyle(0,foreground);
bar3d(0,20,639,349,0,1);line(320,21,320,349);
line(1,185,639,185);setfillstyle(1,BLUE);
bar3d(320,185,639,349,0,1); setfillstyle(1,BLACK);
outtextxy(425,189, "MONITÖR");
            if (machine == 0)
            outtextxy(495,189, "TORNA");
            else
outtextxy(495,189, "FREZE"); outtextxy(330,210,
"Takım:");
outtextxy(346,225, "FR:   ");outtextxy(346,240, "SS:   ");
outtextxy(511,210, "X:    "); outtextxy(511,225, "Y:   ");
outtextxy(511,240, "Z:   ");line(321,255,639,255);
outtextxy (460,259, "G-KODU"); outtextxy (145, 24,
"ÜST");
outtextxy (450, 24, "ÇAPRAZ");outtextxy (140, 189,
"YAN");
return;
```

```
}
void iso_view()
{
int  i; outtextxy(240,15,"ÇAPRAZ GÖRÜNÜM");
outtextxy(240,16,"_____________") ;
setviewport(10,60,630,340,1);
if (scale_flag == FALSE)
        {
        outtextxy(240,150,"LÜTFEN
BEKLEYINIZ...");
        scale_path(TRUE);   setcolor(BLACK);
        bar3d(240,150,380,160,0,1);scale_flag =
TRUE;
        }
        for (i=1 ; i<num_pnts ; i++)
        {
        if (point[i].tool == move)
                setcolor(move_color);
        else
                setcolor(cut_color);
d_line(point[i-1].xiso,point[i].xiso,
            point[i-1].yiso,point[i].yiso);
    }
 return;
}
void top_view()
{
        int i;
outtextxy(265,15,"ÜSTTEN GÖRÜNÜS");
        outtextxy(265,16,"_____________");
        setviewport(10,60,630,340,1);
        if (scale_flag == FALSE)
        {
outtextxy(240,150,"LÜTFEN BEKLEYINIZ...");
scale_path(TRUE);   setcolor(BLACK);
bar3d(240,150,380,160,0,1);     scale_flag = TRUE;
        }
        for (i=1 ; i<num_pnts ; i++)
        {
        if (point[i].tool == move)
                setcolor(move_color);
        else
                setcolor(cut_color);
d_line (point[i-1].x,point[i].x,point[i-1].y,point[i].y);
        }
return;
}
void side_view()
{
        int i;
outtextxy(263,15,"YANDAN GÖRÜNÜS");
        outtextxy(263,16,"_____________");
        setviewport(10,60,630,340,1);
        if (scale_flag == FALSE)
        {
outtextxy(240,150,"LÜTFEN BEKLEYINIZ...");
scale_path(TRUE);   setcolor(BLACK);
bar3d(240,150,380,160,0,1);scale_flag = TRUE;
        }
for (i=1 ; i<num_pnts ; i++)
        {
        if (point[i].tool == move)
            setcolor(move_color);
        else
            setcolor(cut_color);
d_line(point[i-1].xz,point[i].xz,point[i-1].z,point[i].z);
        }
 return;
}

void set_view(int view)
{
   switch(view)
        {
        case 0:
                setviewport(0,0,639,349,1);
                break;
        case 1:
                setviewport(8,40,313,180,1);
                break;
        case 2:
                setviewport(321,34,638.184,1);
                break;
        case 3:
                setviewport(8,204,313,344,1);
                break;
        case 4:
                setviewport(321,198,638,254,1);
                break;
        case 5:
                setviewport(321,275,638,348,1);
                break;
        }
return;
}

void d_gcode(char newcode[40])
{
        int i;
set_view(GCODE);   setfillstyle(1,BLUE);
        if (newcode[0] == 32)
            for (i=0 ; i<=60 ; i+=12)
            {
   setcolor(BLUE);     bar3d(3,i,317,i+9,0,1);
   setcolor(WHITE);     outtextxy(3,i,dgcode[(i/12)]);
            }
        else
        if (strcmp(dgcode[5],newcode))
        {
        set_view(GCODE);   setfillstyle(1,BLUE);
                for (i=0 ; i<=4 ; i++)

        strcpy(dgcode[i],dgcode[i+1]);
srcpy(dgcode[5],newcode) ;
        for (i=0 ; i<=60 ; i+=12)
        {
   setcolor(BLUE);   bar3d(3,i,317,i+9,0,1);
setcolor(WHITE);    outtextxy(3,i,dgcode[(i/12)]);
        }
        }
        set_view(FULL);return;
}

void d_line(int x0,int x1,int y0,int y1)
{
moveto(x0,y0);lineto(x1,y1);
return;
}
void d_path()
{
        int i;
        for (i=1 ; i<num_pnts ; i++)
        {
          if (point[i].tool == move)
                setcolor(move_color);
          else
                setcolor(cut_color);
        set_view(TOP);
        d_line(point[i-1].x,point[i].x,point[i-
1].y,point[i].y);
        set_view(SIDE);
        d_line(point[i-1].xz,point[i].xz,point[i-
1].z,point[i].z);
        set_view(ISO);
d_line(point[i-1].xiso,point[i].xiso,point[i-
1].yiso,point[i].yiso);
```

```
        }
  set_view(FULL);
  return;
}
void monitor(float x,float y,float z)
{
        char buf[20];
set_view(MONITOR);
setfillstyle(1,BLUE);setcolor(BLUE);
bar3d(57,12,187,22,0,1); setcolor(WHITE);
outtextxy(57,12,toolnum);setcolor (BLUE);
bar3d(57,27,187,37,0,1);setcolor(WHITE);
outtextxy(57,27,feedrate);setcolor (BLUE);
bar3d(57,42,187,52,0,1); setcolor(WHITE);
outtextxy (57,42,spindle);setcolor(BLUE);
bar3d(214,12,315,22,0,1); setcolor(WHITE);
outtextxy(214,12,gcvt(x,4,buf)); setcolor(BLUE);
bar3d(214,27,315,37,0,1);setcolor(WHITE);
outtextxy(214,27,gcvt(y,4,buf)); setcolor(BLUE);
bar3d(214,42,315,52,0,1);setcolor(WHITE);
outtextxy(214,42,gcvt(z,4,buf)); setcolor (BLUE);
setcolor(WHITE); return;
}

void scale_path (float scale)
{
        int i;
        float  xztemp;
        for (i=0 ; i<num_pnts ; i++)
        {
   point[i].x = point[i].xwc;   point[i].y = point[i].ywc;
          point[i].z = point[i].zwc;
        }

        if (scale == TRUE)
                for (i=0 ; i<num_pnts ; i++)
                {
                  xztemp = point[i].x;
                  if (machine==MILL)
    getxyz(&point[i].x,&point[i].y,0,0,620,280,TOP);
  getxyz(&xztemp,&point[i].z,0,0,620,280,SIDE);
  point[i].xz = xztemp;
  point[i].xiso=((0.7071*point[i].x) -
(0.7071*(point[i].z)));
point[i].yiso=((-0.4082*point[i].x)+(0.8156*point[i].y)-
(0.4082*(point[i].z)));
 point[i].xiso = point[i].xiso*0.7 + 225;
point[i].yiso = point[i].yiso*0.7 + 190;
        }
        else
for (i=0 ; i<num_pnts ; i++)
        {
          xztemp = point[i].x;
                  if (machine==MILL)
        getxyz(&point[i].x,&point[i].y,0,0,305,140,TO
P);
        getxyz(&xztemp,&point[i].z,0,0,305,140,SIDE
);
                                point[i].xz = xztemp;
point[i].xiso=((0.7071*point[i].x)-(0.7071*(point[i].z)));
point[i].yiso=((-0.4082*point[i].x)+(0.8156*point[i].y)-
(0.4082*(point[i].z)));
point[i].xiso = point[i].xiso*0.7+100;
point[i].yiso = point[i].yiso*0.7+90;
    }
return;
}

void line_pts(int xn,int yn,int zn)
{
        int i;
i = num_pnts;
point[i].x = xn; point[i].y = yn;point[i].z = zn;
point[i].xiso = ((0.7071*point[i].x) -
(0.7071*(point[i].z)));
point[i].yiso=( (-0.4082*point[i].x) + (0.8156*point[i].y) -
                                (0.4082*(point[i].z)));
point[i].xiso = point[i].xiso*0.7 + 100;
point[i].yiso = point[i].yiso*0.7 + 90;
if (point[i].tool == move)
setcolor(move_color);
      else
setcolor(cut_color);
set_view(TOP);
d_line(point[i-1].x,point[i].x,point[i-1].y,point[i].y);
set_view(SIDE);
d_line(point[i-1].xz,point[i].xz,point[i-1].z,point[i].z);
set_view(ISO);
d_line(point[i-1].xiso, point[i].xiso,point[i-
1].yiso,point[i].yiso);
num_pnts = num_pnts+1;set_view(FULL);
return;
}

void initial_pnt(int xi, int yi, int zi)
{
int i; i=num_pnts; point[i].x=xi;
point[i].y=yi;point[i].z=zi;
point[i].xiso=((0.7071*point[i].x) - (0.7071*(point[i].z)));
point[i].yiso=((-0.4082*point[i].x) + (0.8156*point[i] .y) -
                                (0.4082*(point[i].z)));
point[i].xiso = point[i].xiso*0.7 + 100;
point[i].yiso = point[i].yiso*0.7 + 90;
num_pnts = num_pnts + 1;
return;
}

void normalize(float *x,float *y,float *z,float sxmax,float
symax,float szmax)
{
float  xztemp = *x;
point[num_pnts].xwc=*x;point[num_pnts].ywc=*y;
point[num_pnts].zwc=*z;
if (machine==MILL)
        getxyz(x,y,0,0,sxmax,symax,TOP);
        getxyz(&xztemp,z,0,0,sxmax,szmax,SIDE);
        point[num_pnts].xz = xztemp;
        return;
}

void getxyz(float *x,float *y,float sxmin,float symin,float
sxmax,float symax, int zflag)
{
float  obratio,scratio,xfactor,yfactor;
float  scalefact;float  xcorrect = 0;float  ycorrect = 0;
float  ymaxmin,xmaxmin;
if (zflag == TOP)
        ymaxmin = ymax - ymin;
else
ymaxmin = zmax - zmin;
xmaxmin = xmax - xmin;
obratio    = fabs(xmaxmin / ymaxmin);
scratio    = fabs(((sxmax - sxmin) / (symax - symin)) /
1.29);
xfactor    = fabs((sxmax - sxmin) / xmaxmin);
yfactor    = fabs(((symax - symin)*1.29) / ymaxmin);
if (obratio >= scratio)
{
scalefact = xfactor;
ycorrect = ((1.29 * (symax-symin)) - (scalefact *
ymaxmin))/2;
}
else
{
scalefact = yfactor;
xcorrect  = ((sxmax-sxmin) - (scalefact * xmaxmin))/2;
```

```
}
*x = ((*x - xmin)*scalefact) + xcorrect;
if (zflag == TOP)
    *y = symax - ((((*y - ymin)*scalefact) +
ycorrect)/1.29);
else
    *y = symax - ((((*y - zmin)*scalefact) +
ycorrect)/1.29);
 return;
}

void doscommand()
{
char  dos[63];
setviewport(10,75,639,105,1);
strcpy(dos,"");
input_str(dos, "DOS Komutu:  ",
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
10,10,5);
dos[62]=0;set_view(FULL);system("CLS");
clearviewport();
closegraph();printf("%s\n",dos);system(dos);
printf ("Devam Etmek İçin Bir Tuşa Basınız\n");
get_key();system("CLS");init_graphics();set_screen();
if (num_pnts > 0)
{
  d_path();  d_gcode(blank);
}
return;
}
void fileread()
{
set_view(GCODE);   setfillstyle(1,BLUE); setcolor
(BLUE);bar3d(0,0,317,73,0,1); input_str(gcodefile,"G-
KOD DOSYASINI GIRINIZ:
","XXXXXXXXXXXX",11,30,4);
setfillstyle(1,BLUE);setcolor(BLUE);bar3d(0,0,317,73,0,
1);
   return;
}

void firstpass()
{
        int i; int  cnt1,cnt2;
char dumm[2];  dumm[0] = 32;  dumm[1] = 0;
        set_screen();num_pnts =0;
        for (i=0 ; i<=5 ; i++)
                strcpy(dgcode[i]," ");
        errorfile = fopen("gkod.hat","w");
        printf("\n");printf ("\n");
        if (parse(gcodefile) == 0)
                return;
xprev=0; yprev=0; zprev=0;
xmax=-3.0E38;ymax=-3.0E38;  zmax=-3.0E38;
xmin= 3.0E38;ymin= 3.0E38;   zmin= 3.0E38;
error_flag=FALSE;  spin_flag=FALSE;
tool_flag=FALSE;feed_flag=FALSE; strt_flag=FALSE;
stop_flag=FALSE;  pend_flag=FALSE;  g_flag=FALSE;
g92_flag=FALSE; gz_flag=FALSE;g17_flag=TRUE;
g18_flag=FALSE;  g19_flag=FALSE;
if (machine==LATHE)
        {
g17_flag=FALSE;g18_flag=TRUE;
        }
counter=0;
        for (cnt1=0;cnt1<2;cnt1++)
        for (cnt2=0;cnt2<10;cnt2++)
          err_array[cnt1][cnt2]=FALSE;
g20_flag=FALSE;    g90_flag=TRUE;
first();printf("\n");int_error();
setcolor(BLACK);bar3d(110,75,270,85,0,1);
fclose(errorfile) ;
if (error_flag==FALSE)
{
g20_flag=FALSE;g90_flag=TRUE;  g17_flag=TRUE;
g18_flag=FALSE;  g19_flag=FALSE;
                if (machine==LATHE)
                {
        g17_flag=FALSE;g18_flag=TRUE;
                }
                secondpass();
        }
        else
        {
set_view(GCODE);   setfillstyle(1,BLUE);
setcolor(BLUE); bar3d(0,0,317,73,0,1);
input_str(dumm," G-KOD Dosya İçerik Hatası
","X",48,18,2);
input_str(dumm,"Hata Dosyasına
Bakınız.:GKOD.HAT","X",28,45,2);
get_key();
setfillstyle(1,BLUE);
        setcolor(BLUE);bar3d(0,0,317,73,0,1);
set_view(FULL);
        }
        return;
}

void secondpass()
{
xprev=0;   yprev=0;    zprev=0;
g20_flag=FALSE; g90_flag=TRUE;initial_flag=TRUE;
strcpy(spindle,"");  strcpy (feedrate,"");
strcpy(toolnum,"");   strcpy(gcodest,"");
second();
return;
}

void first()
{
char block[513];FILE *gcode;
if((gcode=fopen("gcode.dat","r"))==NULL)
        {
        printf("GCODE.DAT Dosyası Açılamadı\n");
                exit (1);
        }
strcpy(block," ");
while(fgets(block,BLK_SIZE,gcode)!=NULL)
        {
block[strlen(block)]='\0';process(block);strcpy(block," ");
        }
fclose(gcode);
return;
}

void process(char blk[])
{
char  let_tok[5],gc[200],gctrun[40];
it fin_flag,   indx,   x_flag,   y_flag,  z_flag,   i_flag,
   i_flag,  k_flag,   rad_flag,  tnum, mnum, gnum,
snum;
float x,y,z, feed,   rad, i,j,k;
char  *TokPnt;
char CheckString[512] ;
x=0.0;  y=0.0;  z=0.0;  feed=0.0;  rad=0.0;  i=0.0;  j=0.0;
k=0.0;
tnum=0; mnum=0; gnum=0; snum=0;
fin_flag=FALSE;   x_flag=FALSE;   y_flag=FALSE;
z_flag=FALSE;i_flag=FALSE;    j_flag=FALSE;
k_flag=FALSE;   rad_flag=FALSE;strcpy(gc,"");
strcpy(gctrun,"");strcpy(CheckString,blk );
if ((TokPnt = strtok(blk," ")) == NULL)
{
printf( "HATA : G KOD DOSYASI İÇERISINDE BOS
BLOK\n");
```

```
            exit(1);
          }
strcpy(let_tok,TokPnt);counter++;
while (fin_flag==FALSE)
          {
            switch (let_tok[0])
            {
    case 'G':
    gnum=atoi(strtok(NULL," "));
    if ((gnum==50) && (machine==LATHE))
gnum=92;
    if ((gnum<GMINR) || (gnum>GMAXR))
  { error_flag=TRUE; error_num(2,gnum,x);  }
 if (((gnum==1) || (gnum==2) || (gnum==3)) &&
g_flag==FALSE))
{
        g_flag=TRUE;
          err_array[0][4]=TRUE;err_array[1][4]=counter
;
                    }

  if (((gnum==0)||(gnum==1)||(gnum==2)||(gnum==3))
&& (gz_flag==FALSE))  {
                    gz_flag=TRUE;
err_array[0][8]=TRUE; err_array[1][8]=counter;
                    }
    if ((gnum==92) && (g92_flag==FALSE))
                    {
                    g92_flag=TRUE;
err_array[0][7]=TRUE;err_array[1][7]=counter;
                    }
    if (gnum==20) g20_flag=TRUE;
    if (gnum==21)g20_flag=FALSE;
    if (gnum==90)g90_flag=TRUE;
    if (gnum==91)g90_flag=FALSE;
if (machine==MILL) {
          if (gnum==17) {
g17_flag=TRUE; g18_flag=FALSE; g19_flag=FALSE;
                    }
if (gnum==18)
{
g17_flag=FALSE;g18_flag=TRUE;g19_flag=FALSE;
          }
if (gnum==19){
g17_flag=FALSE;g18_flag=FALSE;g19_flag=TRUE;
          }
    }
 break;
 case 'X':
if ((TokPnt= strtok(NULL," "))==NULL)
{
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );
exit(1);
}
x=atof(TokPnt);
if ((x<XMINR)||(x>XMAXR))
{
error_flag=TRUE;error_num(3,gnum,x);
break;
}
if (machine==LATHE)
   x=x/2;x_flag=TRUE;
   break;
case 'Y':
if ((TokPnt = strtok(NULL," ")) == NULL)
{
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );
exit(1);
}
y=atof(TokPnt);
if ((y<YMINR)||(y>YMAXR))
{error_flag=TRUE;error_num(4,gnum,y);
break;
}
y_flag=TRUE;
break;
case 'Z':
if ((TokPnt = strtok(NULL," ")) == NULL )
{printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );
exit(1);
}
z=atof(TokPnt);
if ((z<ZMINR)||(z>ZMAXR))
{
error_flag=TRUE;    error_num(5,gnum,z);
break;
}
z_flag=TRUE;
break;
case 'S':
if ((TokPnt = strtok(NULL," ")) == NULL)
{
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );exit(1);
}
snum=atoi(TokPnt);
if ((snum<SMINR)||(snum>SMAXR))
          {
error_flag=TRUE;    error_num(6,snum,x);
          }
if (spin_flag==FALSE){
spin_flag=TRUE;     err_array[0][0]=TRUE;
err_array[1][0]=counter;
 }
break;
case 'I':
if ((TokPnt =strtok(NULL," ")) == NULL )
{
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );
exit(1);
}
i=atof(TokPnt);
if ((i<IMINR)||(i>IMAXR))
{
error_flag=TRUE;    error_num(7,gnum,i);
break;
}
i_flag=TRUE;
break;
case 'J':
if ((TokPnt = strtok(NULL," ")) == NULL)
{
rintf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );exit(1);
}
j=atof(TokPnt);
if ((j<JMINR)||(j>JMAXR)) {
          error_flag=TRUE;    error_num(8,gnum,j);
break;
}
j_flag=TRUE;
break;
case 'K':
if ((TokPnt = strtok(NULL," ")) == NULL){
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
```

```
printf("%s", CheckString );exit(1);
}
k=atof(TokPnt);
if ((k<KMINR)||(k>KMAXR))
{
error_flag=TRUE;     error_num(9,gnum,k);
break;
}
k_flag=TRUE; break;

case 'R':
if ((TokPnt = strtok(NULL," ")) == NULL)
{printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );exit(1);
}
rad=atof(TokPnt);
if ((rad<RMINR)||(rad>RMAXR)) {
          error_flag=TRUE; error_num(10,gnum,rad);
break;
}
rad_flag=TRUE;
break;
case 'F':
if ((TokPnt = strtok(NULL," ")) == NULL)
{printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );exit(1);
}
feed=atof(TokPnt);
if ((feed<FMINR)||(feed>FMAXR))
          {error_flag=TRUE;
          error_num(11,gnum,feed);
          }
if (feed_flag==FALSE) {
feed_flag=TRUE;err_array[0][2]=TRUE;
err_array[1][2]=counter;
          }
break;
case 'T':
if ((TokPnt = strtok(NULL," ")) == NULL)
{printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );
exit(1); }
tnum=atoi(TokPnt);
if ((tnum<TMINR)||(tnum>TMAXR))
          {
error_flag=TRUE; error_num(12,tnum,x);
          }
if (tool_flag==FALSE)
{tool_flag=TRUE;
rr_array[0][1]=TRUE; err_array[1][1]=counter;
}
break;
case 'M':
          if ((TokPnt = strtok(NULL," ")) == NULL)
          {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );
exit(1);
}
mnum=atoi(TokPnt);
if ((mnum<MMINR)||(mnum>MMAXR))
          {
error_flag=TRUE;     error_num(13,mnum,x);
          }
if (((mnum==3)||(mnum==4))&&(strt_flag==FALSE))
          {
          strt_flag=TRUE;err_array[0][3]=TRUE;
err_array[1][3]=counter;}
if ((mnum==5)&&(stop_flag==FALSE))
          {
stop_flag=TRUE;     err_array[0][5]=TRUE;
err_array[1][5]=counter;    }
if ((mnum==30)&&(pend_flag==FALSE))
{          pend_flag=TRUE;err_array[0][6]=TRUE;
err_array[1][6]=counter;}
break;
case '*': if ((TokPnt = strtok(NULL,"\n")) == NULL)
{printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
printf("%s", CheckString );       exit(1); }
strcpy(gc,TokPnt);
for (indx=0; indx<=38; indx++)
{ gctrun[indx]=gc[indx]; }
gctrun[39]='\0';
break;
default:
break;
}
          if ((TokPnt = strtok(NULL," ") ) == NULL)
          { fin_flag=TRUE; }
          else
          { strcpy(let_tok,TokPnt); }
}
if ((machine==LATHE)&&(y!=0.0))
{error_flag=TRUE;error_num(27,gnum,y); }
if ((gnum==0)||(gnum==1)||(gnum==2) || (gnum==3) ||
(gnum==92))
{          if (g20_flag==TRUE)
          { x*=25.4; y*=25.4; z*=25.4; }
          if (x_flag==TRUE)
          {if (g90_flag==FALSE)x+=xprev;}
          else x=xprev;
          if (y_flag==TRUE)
          {if (g90_flag==FALSE)
          y+=yprev; }
          else      y=yprev;
          if (z_flag==TRUE)
          {if (g90_flag==FALSE)
                    z+=zprev; }
          else   z=zprev;
          if (machine==LATHE) x=x/2;
          if ((gnum!=2)&&(gnum!=3))
          {
if (x>xmax) xmax=x; if (x<xmin) xmin=x;
if (y>ymax) ymax=y; if (y<ymin) ymin=y;
if (z>zmax) zmax=z; if (z<zmin) zmin=z;
          }else
          {
if (((i_flag==TRUE) || (j_flag==TRUE)) ||
(k_flag==TRUE))
circ(xprev,yprev,zprev,x,y,z,0.0,x+i,y+j,z+k,gnum,0);
if (rad_flag==TRUE)
circ(xprev,yprev,zprev,x,y,z,rad,0.0,0.0,0.0,gnum,0);
          }
xprev=x; yprev=y; zprev=z;
          }
          return;}
void int_error()
{
          int dum1; float dum2; dum1=0; dum2=0.0;
          if (err_array[0][0]==FALSE)
          {error_flag=TRUE;error_num(14,dum1,dum2)
;
          }
          if (err_array[0][1]==FALSE)
          {error_flag=TRUE;error_num(15,dum1,dum2)
;
          }
          if (err_array[0][2]==FALSE)
          {error_flag=TRUE;error_num(16,dum1,dum2)
;
          }
```

```
            if (err_array[0][3]==FALSE)
            {error_flag=TRUE;error_num(17,dum1,dum2)
;
            } else
    {if (err_array[1][0]>err_array[1][3])
{error_flag=TRUE;   error_num(18,dum1,dum2);}
if (err_array[1][1]>err_array[1][3])
{error_flag=TRUE;   error_num(19,dum1,dum2);      }
if (err_array[1][3]>err_array[1][4])
{error_flag=TRUE;   error_num(21,dum1,dum2);      }
if (err_array[1][3]>err_array[1][6])
{error_flag=TRUE;error_num(22,dum1,dum2);}
    }
if (err_array[0][4]==FALSE)
{error_flag=TRUE;error_num(23,dum1,dum2); }
if (err_array[0][5]==FALSE)
{error_flag=TRUE;   error_num(24,dum1,dum2); }
if (err_array[0][6]==FALSE)
{error_flag=TRUE;   error_num(25,dum1,dum2); }
            else
{if ((err_array[1][6]!=(counter-
1))&&(err_array[1][6]!=(counter-
2))){error_flag=TRUE;error_num(26,dum1,dum2);     }
}
if (err_array[0][7]==FALSE)
{error_flag=TRUE;   error_num(20,dum1,dum2); }
else
if (err_array[1][8]<=err_array[1][7])
{error_flag=TRUE; error_num(28,dum1,dum2);}
            return;
}
void error_num(int err_num, int par1, float par2)
{
            switch(err_num)
            {
            case 2:
fprintf(errorfile,"Hatalı G Kodu(Sınır Dışı) %d\n",par1);
                        break;
            case 3:
fprintf(errorfile,"Hatalı X Koordinatı(Sınır Dışı)
%f\n",par2);
                        break;
            case 4:
fprintf(errorfile,"Hatalı Y Koordinatı(Sınır Dışı)
%f\n",par2);
                        break;
            case 5:
printf(errorfile,"Hatalı Z Koordinatı(Sınır Dışı)
%f\n",par2);
                        break;
            case 6:
printf(errorfile, "Hatalı S Kodu(Sınır Dışı) %d\n",par1);
                        break;
            case 7:
fprintf(errorfile,"Hatalı I Koordinatı(Sınır Dışı)
%f\n",par2);
                        break;
            case 8:
fprintf(errorfile,"Hatalı J Koordinatı(Sınır Dışı)
%f\n",par2);
                        break;
            case 9:
fprintf(errorfile,"Hatalı K Koordinatı(Sınır Dışı)
%f\n",par2);
                        break;
            case 10:
fprintf(errorfile,"Hatalı Yarıçap Değeri(Sınır Dışı)
%f\n",par2);
                        break;
            case 11:
fprintf(errorfile,"Hatalı Hatve Miktarı(Sınır Dışı)
%f\n",par2);
                        break;
            case 12:
fprintf(errorfile,"Hatalı T Kodu(Sınır Dışı) %d\n",par1);
                        break;
            case 13:
fprintf(errorfile,"Hatalı M Kodu(Sınır Dışı) %d\n",par1);
                        break;
            case 14:
fprintf(errorfile,"Mil Devir Hızı Ayarlanmadı\n");
                        break;
            case 15:
fprintf(errorfile,"Takım Seçilmedi\n");
                        break;
            case 16:
fprintf(errorfile,"Ilerleme Miktarı(Hatve) seçilmedi\n");
                        break;
            case 17:
fprintf(errorfile,"Mil Dönmeye Başlatılmadı\n");
                        break;
            case 18:
fprintf(errorfile,"Mil Dönmeye başlamadan önce mil devir
hızı ayarlanmadı\n");
                        break;
            case 19:
fprintf(errorfile,"Mil Dönmeye başlamadan önce Takım
seçilmedi\n");
                        break;
            case 20:
fprintf(errorfile,"G92/50 Mutlak Sıfır Noktası
Bulunamadı\n");
                        break;
            case 21:
fprintf(errorfile,"Mil dönmeye başlamadan önce G00 dan
başka bir G kodu ile karşılaşıldı\n");
                        break;


case 22:
fprintf(errorfile,"Mil dönmeye başlamadan önce program
sonu ile karşılaşıldı\n");
            break;
case 23:
fprintf(errorfile,"G00'dan başka bir G kodu
bulunamadı\n");
                        break;
case 24:
fprintf(errorfile,"Mil Dönüşü durdurulmadı\n");
                        break;
case 25:
fprintf(errorfile,"Program Sonu Bulunamadı\n");
                        break;
case 26:
fprintf (errorfile,"Teyip Geri Sayımı ,%, Bulunamadı\n");
                        break;
case 27:
fprintf(errorfile,"Program Torna için ayarlanmış, Y değeri
%f, ancak 0 olmalı\n",par2);
                        break;
 case 28:
fprintf(errorfile,"G92/50den önce 0/1/2/3 G kodu ile
karşılaşıldı\n");
                        break;
            }return;
}
void second()
{
            char block[BLK_SIZE];
            FILE *gcode;
if ((gcode=fopen("gcode.dat","r"))==NULL)
            {
             printf("Dosya Açılamadı\n");exit(1);
            }
strcpy(block,"");
while(fgets(block,BLK_SIZE,gcode)!=NULL)
```

```
{
block[strlen(block)]=NULL;process2(block);strcpy(block,
"");
}
 fclose(gcode);
return;
}
void process2(char blk[BLK_SIZE])
{
            char let_tok[5],  gc[200],   dumm[2];
            int  fin_flag,   indx, gen_flag, x_flag,
y_flag, z_flag,
                i_flag, j_flag, k_flag, rad_flag, mnum,
gnum;      float x,y,z,  rad, i,j,k,  tempx,tempy,tempz;
            char *TokPnt;
x=0.0; y=0.0; z=0.0; rad=0.0; i=0.0; j=0.0;
k=0.0;mnum=0;    gnum=0;      tempx=0.0;  tempy=0.0;
tempz=0.0;  dumm[0]=32;  dumm[1]=0;
fin_flag=FALSE;  x_flag=FALSE;   y_flag=FALSE;
z_flag=FALSE;i_flag=FALSE;   j_flag=FALSE;
k_flag=FALSE;   rad_flag=FALSE;gen_flag=FALSE;
            strcpy(gc,"");
            if ((TokPnt=strtok(blk," ")) == NULL)
            {
            printf( "HATA: BOS G KOD SATIRI\n" );
            exit(1);
            }
strcpy(let_tok,TokPnt);
while (fin_flag==FALSE)
{
switch(let_tok[0])
{
            case 'G':
            if ((TokPnt=strtok(NULL," "))== NULL)
            {
   printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
               exit(1);
            }
            gnum=atoi(TokPnt);
            if ((gnum==50)&&(machine==LATHE))
               gnum=92;
if ((gnum==0)||(gnum==1)
||(gnum==2)||(gnum==3)||(gnum==92)) gen_flag=TRUE;
if (gnum==20) g20_flag=TRUE;if (gnum==21)
g20_flag=FALSE;if (gnum==90) g90_flag=TRUE;
if (gnum==91) g90_flag=FALSE;
            if (machine==MILL)
            {
            if (gnum==17)
            {
g17_flag=TRUE;g18_flag=FALSE;g19_flag=FALSE;
                        }
                        if (gnum==18)
                        {
g17_flag=FALSE;g18_flag=TRUE;g19_flag=FALSE;
                        }
                        if (gnum==19)
                        {
g17_flag=FALSE;g18_flag=FALSE;g19_flag=TRUE;
                        }
            }
            break;
            case 'X':
if ((TokPnt=strtok(NULL," "))= = NULL)
            {
  printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
               exit(1);
                        }
x=atof(TokPnt);
if (machine==LATHE) x=x/2;
                        x_flag=TRUE;
            break;
            case 'Y':
                        if ((TokPnt=strtok(NULL," ")) ==
NULL)
                        {
  printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                           exit(1);
                        }
y=atof(TokPnt );       y_flag=TRUE;
            break;
case 'Z':
                        if ((TokPnt=strtok(NULL," ")) ==
NULL)
                        {
  printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                           exit(1);
                        }
            z=atof(TokPnt);        z_flag=TRUE;
            break;
            case 'S':
                        if ((TokPnt=strtok(NULL," ")) ==
NULL)
                        {
 printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                           exit(1);
                        }
strcpy(spindle,TokPnt);
            break;
            case 'I':
                        if ((TokPnt=strtok(NULL," ")) ==
NULL)
                        {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                        exit(1);
                        }

i=atof(TokPnt);        i_flag=TRUE;
            break;
            case 'J':
            if ((TokPnt=strtok(NULL," ")) == NULL)
                        {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                        exit(1);
                        }

j=atof(TokPnt);j_flag=TRUE;
            break;
            case 'K':
                        if ((TokPnt=strtok(NULL," ")) ==
NULL)
                        {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                        exit(1);
            }
                        k=atof(TokPnt);k_flag=TRUE;
            break;
            case 'R':
                        if ((TokPnt=strtok(NULL," ")) ==
NULL)
                        {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                        exit(1);
                        }
rad=atof(TokPnt);rad_flag=TRUE;
            break;
            case 'F':
```

```
                    if((TokPnt=strtok(NULL," ")) ==
NULL)
                    {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                    exit(1);
                    }
                    strcpy(feedrate,TokPnt);
          break;
          case 'T':
                    if ((TokPnt=strtok(NULL," ")) ==
NULL)
                    {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                    exit(1);
                    }
                    strcpy(toolnum,TokPnt);
          break;
          case 'M':
                    if ((TokPnt=strtok(NULL," ")) ==
NULL)
                    {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                    exit(1);
                    }
                    mnum=atoi(TokPnt);
          break;
          case '*':
                    if ((TokPnt=strtok(NULL,"\n")) ==
NULL )
                    {
printf("HATA : KOD SATIRININ
SONLANDIRILMASI\n");
                    exit(1);
                    }
                    strcpy(gc, TokPnt);
                    for (indx=0; indx<=38; indx++)
                    {
                              gcodest[indx]=gc[indx];
                    }
                    gcodest[39]='\0';
          break;
          }
          if ((TokPnt=strtok(NULL," ")) == NULL)
          {         fin_flag=TRUE;      }
          else      {         strcpy(let_tok,TokPnt);
          }
}
if (((gnum==0)||(gnum==1) || (gnum==2) || (gnum==3) ||
(gnum==92)) && (gen_flag==TRUE))
          {
          if (g20_flag==TRUE)
          {
                    x*=25.4;  y*=25.4;  z*=25.4;
                    }
if (x_flag==TRUE)
                    {
                      if (g90_flag==FALSE)
                              x+=xprev;
                    }
                    else      x=xprev;
                    if (y_flag==TRUE)
                    {
            if (g90_flag==FALSE)     y+=yprev;
                    }
                    else  y=yprev;
                    if (z_flag==TRUE)
                    {
                       if (g90_flag==FALSE)
                              z+=zprev;
                    }
                              else z=zprev;
                              tempx=x;  tempy=y;  tempz=z;
                    if ((gnum==92) && (initial_flag==TRUE))
                              {
                                        d_gcode(gcodest);
                    normalize(&tempx,&tempy,&tempz,305,140,1
40);
                    initial_pnt((int)tempx,(int)tempy,(int)tempz);
                    initial_flag=FALSE;
                              }
                    if ((gnum==2) || (gnum==3))
                              {
monitor(x,y,z);           d_gcode(gcodest);
if (((i_flag==TRUE) || (j_flag==TRUE)) ||
(k_flag==TRUE))
  circ (xprev,yprev,zprev,x,y,z,0.0,x+i,y+j,z+k,gnum,1);
                    if (rad_flag==TRUE)
                    circ(xprev,yprev,zprev,x,y,z,rad,0.0,0.0,0.0,gn
um,1);
                              }
                    xprev=x;  yprev=y;  zprev=z;
                              if ((gnum==0) || (gnum==1))
                              {
                              if (gnum==0)
                              point[num_pnts].tool=move;
                    else
                    point[num_pnts].tool=cut;
                    monitor(x,y,z); d_gcode(gcodest);
normalize(&x,&y,&z,305,140,140);
line_pts((int)x,(int)y,(int)z);
                    }
}
else
{
                    x=xprev;  y=yprev;  z=zprev;
monitor(x,y,z);d_gcode(gcodest) ;
}
                    if (mnum==1)
                    {
set_view(GCODE);   setfillstyle(1,BLUE);
                    setcolor(BLUE);
bar3d(0,0,317,73,0,1);
input_str(dumm,"M01 ile İsteğe Bağlı Durma, Devam için
Tuşa Basınız","X",15,30,3);
get_key(); setfillstyle(1,BLUE); setcolor(BLUE);
bar3d(0,0,317,73,0,1);d_gcode(blank);
                    }
                    if (single_step=='E')
                              if (get_key()==ESC)
                                        single_step='H';
                              else;
                    else delay (speed);
                    hot_key();
                    return;
}
int getlinechar(ln,j)
char ln[MAXLINELENGTH];
int *j;
{
                    int i; i = *j; (*j)++;
                    return(ln[i]);
}
void getnextparm(ln,j,s,pc)
                    char ln[MAXLINELENGTH];
                    int *j; char *s; int *pc;
{
                    int i;        s[0]= ' ';    i=1;
                    *pc=getlinechar(ln,j);
                    while (!isalpha(*pc) && *pc != ';' )
                    {
                              if (!isspace(*pc))
                              {
                                        s[i]=*pc;   i++;
```

```
                }
        *pc = getlinechar(ln,j);
        }

        s[i]=' '; s[i+1]= '\0';
}
void getnext(ln,j,s,pc)
        char ln[MAXLINELENGTH];
        int *j; char *s; int *pc;
{
while(*pc!='G' && *pc!='F' && *pc!='T' && *pc!='S'
&& *pc!='M' && *pc!=';')
        {
                getnextparm(ln,j,s,pc);
        }
}
void echoline(line, writeorig)
char line[MAXLINELENGTH];
int *writeorig;
{
if (gcode[0]=='\0' && (X[0]!='\0' || Y[0]!='\0' ||
Z[0]!='\0'))
                strcpy( gcode, last_gcode );
fputs(gcode, outstream);fputs(X, outstream );fputs(Y,
outstream );
fputs(Z, outstream );fputs(I, outstream );fputs(J, outstream
);
fputs(K, outstream ); fputs(R, outstream );
gcode[0]='\0'; X[0]='\0'; Y[0]='\0'; Z[0]='\0';
I[0]='\0';J[0]='\0';
K[0]='\0'; R[0]='\0';
        if (*writeorig)
        {
fprintf( outstream, "\t\t * "); fputs( line, outstream );
*writeorig = FALSE;
        }
}
int isgcode(char code[6])
{
int num; num=atoi(&code[2]);
if (num==0 || num==1 || num==2 || num==3 || num==17 ||
num==18 || num==19 || num==20 || num==21 || num==50
|| num==90 || num==91 || num== 92)
        return ( TRUE );
        else
return( FALSE );
}
int parse(char file[12])
{
char temp[15];char line[MAXLINELENGTH];char
dumm[2];int i; int c;int first_ln; int SaveLine =
TRUE;dumm[0] = 32;
dumm[1] = 0;
        if ((instream=fopen( file,"r"))==NULL)
        {
set_view(GCODE);   setfillstyle(1,BLUE);
        setcolor(BLUE);
bar3d(0,0,317,73,0,1);
input_str(dumm,"DOSYA
BULUNAMADI","X",91,30,3);
get_key(); setfillstyle(1,BLUE); setcolor(BLUE);
bar3d(0,0,317,73,0,1);d_gcode(blank);
        return 0;
        }
set_view(TOP);outtextxy(110,75,"LÜTFEN
BEKLEYIN...");
outstream=fopen("gcode.dat","w");first_ln=TRUE;
gcode[0]='\0';X[0]='\0';Y[0]='\0';Z[0]='\0'; I[0]='\0';
J[0]='\0'; K[0]='\0'; R[0]='\0';
        fgets(line, MAXLINELENGTH, instream );
        i=0;
        while (!(feof(instream)))
        {
        c=getlinechar( line, &i );
        do
        {
        switch (c)
        {
                case 'O':
                case 'o':
                case 'N':
                case 'n':
                {
        temp[0]=c;
getnextparm(line, &i, &temp[1], &c );
                        break;
                }
                case 'X':
                case 'x':
                {
                        X[0]=c;
getnextparm(line, &i, &X[1], &c);
        break;
                }

                case 'Y':
                case 'y':
                {
                        Y[0]=c;
getnextparm(line, &i, &Y[1], &c);
                        break;
                }
                case 'Z':
                case 'z':
                {
                        Z[0]=c;
getnextparm(line, &i, &Z[1], &c);
                        break;
                }
                case 'I':
                case 'i':
                {
                        I[0]=c;
        getnextparm(line, &i, &I[1], &c);
                        break;
                }
                case 'J':
                case 'j':
                {
                        J[0]=c;
        getnextparm(line, &i, &J[1], &c);

                        break;
                }
                case 'K':
                case 'k':
                {
                        K[0]=c;
        getnextparm(line, &i, &K[1], &c);
                        break;
                }
                case 'R':
                case 'r':
                {
                        R[0]=c;
        getnextparm(line, &i, &R[1], &c);
                        break;
                }

                case 'G':
                case 'g':
                {
                        gcode[0]=c;
        getnextparm(line, &i, &gcode[1],
&c);
                if ( isgcode(gcode))
```

```
                        {
                    strcpy(last_gcode, gcode);
                        }
                        else
                        {
            gcode[0]='\0';getnext(line, &i, &temp[1], &c);
                        }
                        break;
                        }
                        case 'F':
                        case 'f':
                        case 'T':
                        case 't':
                        case 'S':
                        case 's':
                        case 'M':
                        case 'm':
                        temp[0]=c;
            if (gcode[0]!='\0' || X[0]!='\0' || Y[0]!='\0' ||
Z[0]!='\0')
                    echoline( line, &first_ln );
                    getnextparm(line, &i, &temp[1],
&c);
                    fputs(temp,outstream);
                    if (!first_ln)
                    fputc('\n', outstream);
                    else
                            echoline( line,
&first_ln);
            break;
            default:
if (c!='G' && c!='g' && c!='F' && c!='f' && c!='S' &&
c!='s' &&   c!='T' && c!='t' && c!='M' && c!='m' &&
c!='N' && c!='n' &&   c!='O' && c!='o' && c!='X' &&
c!='x' && c!='Y' && c!='y' &&   c!='Z' && c!='z' &&
c!='I' && c!='i' && c!='J' && c!='j' &&   c!='K' &&
c!='k' && c!='R' && c!='r' && c!=' ' && c!='%' &&
c!='H' && c!='D')
                    {
                if( is_empty_string(line))    {
fprintf(errorfile,"BOS VE ATLANILMIS SATIR\n");
                            SaveLine=FALSE;
                        }
            else
                    {
fprintf(errorfile,"HATA G-Kod Içerisinde Geçersiz Özel
Harf\n");
fprintf( errorfile,"HATALI SATIR >> %s", line);
fprintf( errorfile,"GEÇERSIZ HARF >> %c\n",c);
            }
set_view(TOP);outtextxy(10,95,"HATA DOSYASI :
GCODE.ERR");
                    }
                    if (c!='\n')
                    {
            getnextparm(line, &i, &temp[1], &c);
                    }
            break;
            }
    } while(c!=';' && c!='+' && c!='\n');
            if( SaveLine == TRUE )
            {
                    echoline(line, &first_ln);
            }
            SaveLine = TRUE;
            fgets(line, MAXLINELENGTH, instream);
            i=0;first_ln = TRUE;
}

fclose(instream); fclose(outstream);
 return 1;
}
void doxyz(float x, float y, float z, int draw)
{
if (draw)
            {
   point[num_pnts].tool=cut;
normalize(&x,&y,&z,305,140,140);
   line_pts((int)x, (int)y, (int)z);
            }
            else
            {
   if (x < xmin) xmin = x;    if (x > xmax) xmax = x;
   if (y < ymin) ymin = y;    if (y > ymax) ymax = y;
   if (z < zmin) zmin = z;    if (z > zmax) zmax = z;
            }
}
void circ(float Xs,float Ys,float Zs,float Xe,float Ye,float
Ze,
float R, float Xc,float Yc,float Zc,int dir,int draw)
{
double Xm,Ym,Zm;   double theta,phi,beta;
double startangle,endangle;double X,Y,Z;
double arcinterval = PI/36.0;
if (Xs != Xe && Ys != Ye && Zs != Ze)
                    {
 fprintf(errorfile, "HATA : Düzlemsel Yay
Tanımlanmadı");
                    }
            if (Xc==0.0 && Yc==0.0 && Zc== 0.0)
            {
Xm = (Xe - Xs)/2;Ym = (Ye - Ys)/2;
Zm = (Ze - Zs)/2;
if (sqrt( Xm*Xm + Ym*Ym + Zm*Zm )/fabs(R) > 1.0)
            {
fprintf(errorfile,"HATA : Geçersiz Yarıçap Değeri %d
\n",R);
                    theta=0.0;
            }
else
theta=acos(sqrt(Xm*Xm + Ym*Ym + Zm*Zm)/fabs(R));
            if (g17_flag) beta=atan2(Ym,Xm);
            if (g18_flag) beta=atan2(Xm,Zm);
            if (g19_flag) beta=atan2(Zm,Ym);
if ((dir == CW && R > 0.0 ) || (dir == CCW && R < 0.0
))
                                    phi = beta -
theta;
if ((dir == CW && R < 0.0 ) || (dir == CCW && R > 0.0
))
                                    phi = beta +
theta;
                    if (g17_flag)
                    {
Xc=Xs+fabs(R)*cos(phi);Yc=Ys+fabs(R)*sin(phi);Zc=Ze
;
                    }
                    if (g18_flag)
                    {
Zc=Zs+fabs(R)*cos(phi);Xc=Xs+fabs(R)*sin(phi);Yc=Ye
;
                    }
                    if (g19_flag)
                    {
Yc=Ys+fabs(R)*cos(phi);Zc=Zs+fabs(R)*sin(phi);Xc=Xe
;
                    }
            }
            else
if (R == 0.0) R=sqrt((Xe-Xc)*(Xe-Xc) + (Ye-Yc)*(Ye-
Yc) + (Ze-Zc)*(Ze-Zc));
            if (g17_flag)
            {
startangle=atan2(Ys-Yc,Xs-Xc);
endangle=atan2(Ye-Yc,Xe-Xc);
            }
```

```
            if (g18_flag)
            {
startangle=atan2(Xs-Xc,Zs-Zc);
endangle=atan2(Xe-Xc,Ze-Zc);
            }
            if (g19_flag)
            {
startangle=atan2(Zs-Zc,Ys-Yc);
endangle=atan2(Ze-Zc,Ye-Yc);
            }
            if (dir==CW) arcinterval *= -1.0;
if ((startangle < endangle) && (dir == CW)) startangle
+=2*PI;
            if ((startangle > endangle) && (dir == CCW))
startangle -=2*PI;

for ( theta = startangle; fabs(theta-endangle) >
fabs(arcinterval); theta += arcinterval)
            {
                        if (g17_flag)
                        {
X=Xc+fabs(R)*cos(theta);Y=Yc+fabs(R)*sin(theta);Z=Z
e;
                        }
                        if (g18_flag)
                        {
Z=Zc+fabs(R)*cos(theta);X=Xc+fabs(R)*sin(theta);Y=Y
e;
                        }
                        if (g19_flag)
                        {
Y=Yc+fabs(R)*cos(theta);Z=Zc+fabs(R)*sin(theta);X=X
e;
                        }
            doxyz(X,Y,Z,draw);
            }
doxyz(Xe,Ye,Ze,draw);
}
int get_key()
{
int x,y,button,a,res,key;int xmin,xmax,ymin,ymax,cx,cy;
if (mouse_flag == TRUE)
            {
cx=160;   cy=237;   res=45;    xmin=cx-1.5*res;
max=cx+1.5*res;     ymin=cy-res;ymax=cy+res;
a=0; setcursorpos(cx,cy);
            do
                        {
getposbut(&button,&x,&y);key=bioskey(1);


if (key==0)
            {
            if (y>ymax)
            a=DOWN;
            if (x>xmax) a=RIGHT;
            if (y<ymin)  a=UP;
            if (x<xmin) a=LEFT;
            if (button==1) a=RETURN;
            if (button == 2)a=ESC;
            }
else
            {
            a=getch();
            if (a == 0) a=getch()+1000;
            }
}while (a==0);
do
            getposbut(&button,&x,&y);
while (button != 0);
            setcursorpos(cx,cy);
            }
            else
            {
a=getch();
if (a==0)   a=getch()+1000;
            }
            return a;
}
int  input_str(char *s, char *m, char *p, int h, int v, int
option)
{
char *hold;int x,max,xx,yy,a;int loop,change,null_flag;
int l,t,r,b; int pic,msg;
a=0;
hold=malloc(200);max=strlen(p);
pic=max*8;msg=strlen(m)*8;    l=h-5;t=v-5;
r=h+msg+pic+5;b=v+13;setfillstyle(0,foreground);
setcolor(foreground); bar(l,t,r,b);
            if (option>0) outtextxy(h,v,m);
            if (option>1)
            {
            null_flag=0;
for(x=0; x<max; x++)
                        {
            if (*(s+x) == NULL)  null_flag=1;
            if (null_flag==1)*(hold+(x*2))=' ';
                        else
*(hold+(x*2))=*(s+x);*(hold+(x*2)+1)=NULL;
            xx=msg+h+(x*8); yy=v;
            outtextxy(xx,yy,hold+(x*2));
                        }
            }
            if (option>2) rectangle(l,t,r,b);
            if (option>3)
            {
setfillstyle(1,foreground);setcolor(background);
            for(x=0; x<max; x++)
            {
            xx=msg+h+(x*8);yy=v;
            bar(xx-1,yy-2,xx+8,yy+10);
            outtextxy(xx,yy,hold+(x*2));
            }
            loop=TRUE;
            x=0;
            do
            {
xx=msg+h+(x*8);yy=v+2;outtextxy(xx,yy,"_");
a=get_key();bar(xx,yy+7,xx+8,yy+8);
switch(a)
                        {
            case RIGHT:
                        x++;
                        if (x>max-1) x=max-1;
                        break;
            case LEFT:
                        x--;
                        if (x<0)     x=0;
                        break;
            case RETURN:
            for(x=0; x<max; x++) *(s+x)=*(hold+(x*2));
                                    loop=0;
                        break;
                        case ESC:
                        loop=0;
                        break;
            case BACKSPACE:
            xx=msg+h+(x*8);yy=v;bar(xx,yy,xx+7,yy+8);
*(hold + (x*2))=SPACE;outtextxy(xx,yy,hold+(x*2));
                        x--;
                        if (x<0) x=0;
                        break;
            default:
                        change=0;
            if (*(p+x)=='X'){
if (a>31 && a<256)  change=1;
```

```
                                        }
if (*(p+x)=='A')        {
if (a>64 && a<123 || a==32)
                        change=1;
                        }
if (*(p+x)=='9'){
            if (a>47 && a<58  || a==32)
            change=1;
                        }
if (*(p+x)=='S'){
            if (a>47 && a<58  || a==32 || a==45)
            change=1;
            }
            if (*(p+x)=='#'){
            if (a>47 && a<58  || a==32 || a==46)
                        change=1;
                        }
            if (change==1) {
            xx=msg+h+(x*8);yy=v;bar(xx,yy,xx+7,yy+8);
            *(hold+(x*2))=a;outtextxy(xx,yy,hold+(x*2));
            x++;
            if (x>max-1)x=max-1;
                        }
                }
            }
            while(loop==TRUE);
}
            if (option>4)
                        {
            setfillstyle(0,foreground);setcolor(foreground);
            bar(l,t,r,b);
            }
            free(hold); setfillstyle(0,foreground);
            setcolor(foreground);
            return a;
}
int  input_int(int *i, char *m, char *p, int h, int v, int
option)
{
char *s;int max,a;max=strlen(p);s=malloc(100)
;itoa(*i,s,10);
a=input_str(s,m,p,h,v,option);*(s+max)=' ';
            *(s+max+1)=NULL;
*i=atoi(s); free(s);
return a;
}
int  input_real(double *r,char *m,char *p,int h,int v,int
option)
{
char *s;int max,a;      max=strlen(p);s=malloc(50);
gcvt(*r,max,s); a=input_str(s,m,p,h,v,option); *(s+max)='
'; *(s+max+1)=NULL; *r=atof(s); free(s);
return a;
}
int horz_tab(char *s[15][15],int h)
{
 int c,i; c=3;
            for (i=0; i<h; i++)  c=c+strlen(s[i][0])+3;
  return c;
}
void show_string(char *s[15][15],int h, int v)
{
int x,y,c;   c=horz_tab(s,h);       x=c*8;
            if (v==0)               y=5;
            else        y=v*18+6;
outtextxy(x,y,s[h][v]);
return;
}

void  set_menu(char *s[15][15])
{
int i,hmax;
for (i=0; strlen(s[i][0]) !=0; i++);
hmax=i-1 ;
 setfillstyle(1,foreground);setcolor(background);
bar3d(0,0,640,18,0,1);
for (i=0; i<=hmax; i++) show_string(s,i,0);
setfillstyle(0,foreground); setcolor(foreground);
return;
}
void set_bar(char  *s[15][15],int  h,int v)
{
 int c,w,l,t,r,b; void *buffer; unsigned size;
c=horz_tab(s,h); w=strlen(s[h][v]); l=(c-1)*8+1;
r=(c+w+1)*8-1;
            if (v==0)   t=3;
            else  t=v*18+4;
b=t+10;size=imagesize(l,t,r,b);   buffer=malloc(size);
 getimage(l,t,r,b,buffer); putimage(l,t,buffer,4);
free(buffer);
return;
}
void use_menu(char *s[15][15],int *ph,int *pv,int *status)
{
void *buffer; unsigned size; int r,t,l,b;  int c,i,a,loop;
int h,v,hmax,vmax,htmp; h=*ph; v=*pv; loop=1;
for (i=0; strlen(s[i][0]) !=0; i++);
            hmax=i-1;
if (h<0)  h=0; if (h>hmax) h=hmax;
setfillstyle(1,foreground); setcolor(BLACK);
do
            {
            for (i=1; strlen(s[h][i]) !=0; i++);
            vmax=i-1;
if (v<0) v=0;if (v>vmax) v=vmax; if (vmax>0)
            {
c=horz_tab(s,h); l=(c-1)*8; t=18;
r=(c+strlen(s[h][1])+1)*8; b=vmax*18+18;
size=imagesize(l,t,r,b);  buffer=malloc(size);
getimage(l,t,r,b,buffer);  bar3d(l,t,r,b,0,1);
            for (i=1; i<=vmax; i++) show_string(s,h,i);
            }
htmp= h;  set_bar(s,htmp,0);
            do
            {
            if (v>0)    set_bar(s,h,v);
a=get_key();
            if (v>0)    set_bar(s,h,v);
            switch(a)
            {
            case DOWN: v++;
            if (v>vmax)
            {
             if (vmax==0)  v=0;
               else       v=vmax;
            }
                        break;
            case UP:   v--;
            if (v<1)
            {
                        if (vmax==0)        v=0;
                                    else    v=1;
                                }
                                break;
            case RIGHT:
                        h++;
            if (h>hmax)h=hmax;
                        v=0;
            break;

            case LEFT:h--;
            if (h<0)    h=0;v=0;
                        break;
                        case RETURN:
            *status=1; loop=0;
                                    break;
```

```
            case ESC: *status=0; loop=0;
                      break;
                      }
            }
            while (h==htmp && loop==1);
set_bar(s,htmp,0);
            if (vmax>0)
            {
            putimage(l,t,buffer,0);free(buffer);
            }
 }
  while (loop==1);
  setfillstyle(0,foreground);  setcolor(foreground);
  *ph=h;   *pv=v;
  return;
}
void hot_key()
{
int a;a=0;
if (bioskey(1)!=0)
            {  a=getch();
             if(a==SPACE)
             {
set_view(TOP); clearviewport(); set_view(SIDE);
clearviewport(); set_view(ISO);  clearviewport();
             }
 if ((a==ESC) && (single_step=='E'))single_step='H';
 if ((a==ESC) && (single_step=='H'))single_step='E';
        }
return;
}
int is_empty_string(char *s)
{
int  n=0;   for( ; ; )
        {
            if((s[n]=='\n') || (s[n]==EOF))
                {
                        return(TRUE);
                }
                else
                if ((s[n]==' ') || (s[n]=='\t'))
                {
                  ;
                }
                else
                {
                        return(FALSE);
                }
                n++;
                  } }
```