YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CUBIC SPLINE
ve
YÜZEY GRAFİĞİ ELDE ETME

Mat.Müh. Müslüm ÖZIŞIK

F.B.E Matematik Mühendisliği Anabilim Dalında

hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Tahir ŞİŞMAN

Prof. Yavuz Aksoy

Doç. Dr. Mustafa Sivri

İSTANBUL, 1997

İÇİNDEKİLER

Bu Yüksek Lisans tezinin seçimi ve hazırlanışı esnasında, gerek lisans öğrenimim sırasında vermiş olduğu dersler dolayısıyla gerekse Yüksek Lisans öğrenimim sırasında bana her konuda yardımcı olan hocam Sayın Prof. Tahir Şişman'a teşekkür ederim.

ÖZET

Eğriler ve yüzeyler özellikle matematik ilmi içerisinde başlı başına önemli bir uğraş alanını oluşturmaktadır. Bu Yüksek Lisans Tezinin amacı, bilgisayar yardımı ile yüzey grafiği çizmenin ilk adımını oluşturan düzlemdeki bir eğriyi uzayda döndürmek sureti ile yüzey grafiği elde etmektir. Bu eğrinin elde edilmesi ise Cubic Spline interpolasyonu ile sağlanmaktadır.

İlk bölümde Cubic Spline interpolasyonu teorik olarak anlatılmıştır.

İkinci bölümde üç boyutlu uzaydaki bir görüntünün bilgisayar ekranına nasıl düşürüldüğü ve koordinat eksenlerinin dönüşümü anlatılmıştır. Üçüncü bölümde ise 'yüzey türetme' başlığı altında yüzeyin değişik konumları incelenmiştir. Dördüncü bölümde programın genel algoritması hakkında bilgi verilmiş, programın çıktısı ile birlikte algoritması ve ileriki aşamalarda ideal olarak çizdirilmek istenen bazı fonksiyonların yüzeylerine örnekler verilmiştir.

SUMMARY

Especially in mathematics science curves and surfaces has an important research subject. The aim of this study is, by rotating a curve which is the first stage of drawing a surface graphic, on a plane in space by using computers . For drawing this curve Cubic Spline interpolation is used.

In the first chapter, Cubic Spline Interpolation is explained theoritically. In the second chapter, how an image in 3D space can be obtain on a screen and rotation of the coordinate axes. In the third chapter, different stages of a surface is studied under the caption of surface generation. In the fourth chapter, details given about the general algorithm of the program and examples given about some functions which has been ideally desired to be drawn in the further stages.

Giriş :

Üç boyutlu uzayda. düzlemde belirlenen noktalardan geçen bir eğriden yüzey elde etmek için Cubic Spline interpolasyonu kullanılmaktadır.

İlk bölümde, Cubic Spline interpolasyonu teorik olarak açıklanmıştır. Daha sonra bilgisayar ekranında üç boyutlu görüntü elde etme konusu, sözkonusu matrisyel dönüşümler ayrıntılı olarak açıklanmak sureti ile anlatılmıştır. Öteleme yüzeyi ve döndürme yüzeyi konuları ise "Yüzey Türetme" başlığı adı altında verilmiştir.

Bütün bu anlatılanların uygulandığı bir bilgisayar programı yazılmış ve bu programın algoritması ayrı bir başlık altında incelenmiştir. En son olarak da programın akış diyagramı, programın printer çıktısı ve ideal olarak çizdirilmek istenen yüzeylere ait örnek çıkışlar verilmiştir.

1. Cubic Spline

Cubic Spline bilindiği gibi bir interpolasyon yöntemidir ve interpolasyon belirlenen her

(x_i,y_i) , (i= 1, 2, ..., n)

için her

h_i (i= 1,2,..., n-1)

aralıklarından üçüncü dereceden polinomlar geçirmek süretiyle yapılır.

$(x_1, x_2, \ldots, x_n)$ $x \in [x_{i-1}, x_i]$ $h_i = x_i - x_{i-1}$ (h_i : i. aralığın uzunlugu)

$x \in [x_1, x_2]$ $f(x) \approx P_1(x)$

$$P_1(x) = a_0^{(1)} + a_1^{(1)}x + a_2^{(1)}x^2 + a_3^{(1)}x^3$$

$$a_0^{(1)} + x_1 a_1^{(1)} + x_1^2 a_2^{(1)} + x_1^3 a_3^{(1)} = f_1$$

$$a_0^{(1)} + x_2 a_1^{(1)} + x_2^2 a_2^{(1)} + x_2^3 a_3^{(1)} = f_2$$

$$a_1^{(1)} + 2x_1 a_2^{(1)} + 3x_1^2 a_3^{(1)} = f_1'$$

$$a_1^{(1)} + 2x_2 a_2^{(1)} + 3x_2^2 a_3^{(1)} = f_2'$$

Buradaki $P_i(x)$ $[x_1, x_2]$ aralığından geçen üçüncü dereceden bir polinomdur. Dolayısı ile her h_i (i=1,...,n-1) için P_1, P_2..., P_{n-1} üçüncü dereceden polinomlar elde edilecektir. Buradaki $P_i(x)$ polinomu için

$$P_i(x) = \left\{ \begin{array}{r} \text{3.dereceden} \\ P, P', P'' \text{ her yerde sürekli} \end{array} \right\}$$

olduğunu söyleyebiliriz. Düğüm noktalarında ikinci türev üretileceğinden.

$$P_i''(x) = f''(x_i)\frac{(x - x_{i-1})}{h_i} + f''(x_{i-1})\frac{(x_i - x)}{h_i}$$

olarak $P''_i(x)$' in iki defa integralini almak sureti ile,

$$P_i(x) = f''(x_i)\frac{(x - x_{i-1})^3}{6h_i} + f''(x_{i-1})\frac{(x_i - x)^3}{6h_i} + A_i x + B_i \qquad (1.1)$$

elde edilir. $P_i(x)$ polinomundaki A ve B katsayıları ve ikinci türev değerlerini hesaplarsak (1.1) ile verilen $P_i(x)$ polinomunu elde etmiş oluruz. Polinomun x_{i-1} ve x_i noktalarındaki değerlerini yazalım.

$$A_i x_{i-1} + B_i + f''(x_{i-1})\frac{h_i^2}{6} = f(x_{i-1}) \qquad (1.2)$$

$$A_i x_i + B_i + f''(x_i)\frac{h_i^2}{6} = f(x_i) \qquad (1.3)$$

(1.2) ve (1.3) denklemlerinden

$$A_i = \frac{f(x_i) - f(x_{i-1})}{h_i} - \frac{h_i}{6}(f''(x_i) - f''(x_{i-1}))$$

$$B_i = f(x_i) - f''(x_i)\frac{h_i^2}{6} - \frac{f(x_i) - f(x_{i-1})}{h_i}x_i + \frac{h_i x_i}{6}(f''(x_i) - f''(x_{i-1}))$$

elde edilir.

i. aralık için $x \in [x_{i-1}, x_i]$ olarak

$$P_i'(x) = f''(x_i)\frac{(x - x_{i-1})^2}{2h_i} - f''(x_{i-1})\frac{(x_i - x)^2}{2h_i} + A_i$$

$$P_i'(x_i) = f''(x_i)\frac{h_i}{2} + A_i$$

(i+1). aralık için $x \in [x_i, x_{i+1}]$

$$P_{i+1}'(x) = f''(x_{i+1})\frac{(x - x_i)^2}{2h_{i+1}} - f''(x_i)\frac{(x_{i+1} - x)^2}{2h_{i+1}} + A_{i+1}$$

$$P'_{i+1}(x_i) = A_{i+1} - f''(x_i)\frac{h_{i+1}}{2}$$

$$P'_i(x_i) = P'_{i+1}(x_i) \tag{1.4}$$

Buradan görüleceği gibi $[x_i, x_{i+1}]$ aralığındaki $P_i(x)$ polinomunun birinci türevleri eşit alınıyor. $P_i(x)$ polinomu x=xi ve $x = x_{i+1}$ noktalarından geçiyor. Buna göre (1.4) eşitliğinden

$$A_{i+1} - f''(x_i)\frac{h_{i+1}}{2} = A_i + f''(x_i)\frac{h_i}{2} \qquad (i = 1, 2, ..., n-1) \tag{1.5}$$

bağıntısı elde edilir.

(1.5) nolu denklemde A_i ve A_{i+1} değerlerini yerine koyarsak

$$\frac{h_{i+1}}{6}f''(x_{i+1}) + \frac{(h_{i+1}-h_i)}{3}f''(x_i) + \frac{h_i}{6}f''(x_{i-1}) = \frac{f(x_{i+1})-f(x_i)}{h_{i+1}} - \frac{f(x_i)-f(x_{i-1})}{h_i}$$

eşitliği elde edilir. Bu ifade düzenlendiği zaman

$$\frac{h_i}{6}f''(x_{i-1}) + \frac{(h_{i+1}-h_i)}{3}f''(x_i) + \frac{h_{i+1}}{6}f''(x_{i+1}) = \alpha_i$$

şeklini alır. Sırasıyla, i=1, 2 ,..., n-2 değerleri için

$$i = 2 \qquad \frac{h_2}{6}f''(x_1) + \frac{(h_3-h_2)}{3}f''(x_2) + \frac{h_3}{6}f''(x_3) = \alpha_2$$

$$i = 3 \qquad \frac{h_3}{6}f''(x_2) + \frac{(h_4-h_3)}{3}f''(x_3) + \frac{h_4}{6}f''(x_4) = \alpha_3$$

$$i = 4 \qquad \frac{h_4}{6}f''(x_3) + \frac{(h_5-h_4)}{3}f''(x_4) + \frac{h_5}{6}f''(x_5) = \alpha_4$$

. . . . . .

$$i = n-2 \qquad \frac{h_{n-2}}{6} f''(x_{n-3}) + \frac{(h_{n-1}-h_{n-2})}{3} f''(x_{n-2}) + \frac{h_{n-1}}{6} f''(x_{n-1}) = \alpha_{n-2}$$

$$i = n-1 \qquad \frac{h_{n-1}}{6} f''(x_{n-2}) + \frac{(h_n-h_{n-1})}{3} f''(x_{n-1}) + \frac{h_n}{6} f''(x_n) = \alpha_{n-1}$$

denklemleri elde edilir. Yukarıda görüldüğü gibi bu bir lineer denklem sistemidir. Bu denklem sisteminde $f''(x_1)$ ve $f''(x_n)$ ikinci türev değerleri bilinmektedir. Bu durumda katsayılar matrisi şu şekilde oluşacaktır.

$$\begin{pmatrix} \frac{h_3-h_2}{3} & \frac{h_3}{6} & 0 & 0 & \dots & \dots & 0 \\ \frac{h_3}{6} & \frac{h_4-h_3}{3} & \frac{h_4}{6} & 0 & \dots & \dots & 0 \\ 0 & \frac{h_4}{6} & \frac{h_5-h_4}{3} & \frac{h_5}{6} & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & \frac{h_{n-1}}{6} & \frac{h_n-h_{n-1}}{3} \end{pmatrix}$$

$$\alpha_i = \frac{f(x_{i+1}) - f(x_i)}{h_{i+1}} - \frac{f(x_i) - f(x_{i-1})}{h_i}$$

$$i = 2 \qquad \alpha_2 = \frac{f(x_3) - f(x_2)}{h_3} - \frac{f(x_2) - f(x_1)}{h_2} - \frac{h_2}{6} f''(x_1)$$

$$i = 3, \dots, n-2 \qquad \alpha_i = \frac{f(x_{i+1}) - f(x_i)}{h_{i+1}} - \frac{f(x_i) - f(x_{i-1})}{h_i}$$

$$i = n-1 \qquad \alpha_{n-1} = \frac{f(x_n) - f(x_{n-1})}{h_n} - \frac{f(x_{n-1}) - f(x_{n-2})}{h_{n-1}} - \frac{h_n}{6} f''(x_n)$$

Bu denklemler ile lineer denklem sisteminin "sağ tarafı" hesaplanır. Böylece daha önce bahsettiğimiz lineer denklem sisteminin çözümünden ikinci türev değerleri bulunabilir. A ve B değerleri (1.1) denkleminde yerine yazılırsa $P_i(x)$ polinomu;

$$P_i(X) = \frac{f''(x_i) - f''(x_{i-1})}{6h_i} X^3 + \frac{x_i f''(x_{i-1}) - x_{i-1} f''(x_i)}{2h_i} X^2 +$$

$$\left(\frac{x_{i-1}^2 f''(x_i) - x_i^2 f''(x_{i-1})}{2h_i} + A\right) X + \frac{x_i^3 f''(x_{i-1}) - x_{i-1}^3 f''(x_i)}{6h_i} + B$$

şeklinde elde edilmiş olur.

Cubic Spline yönteminde eğrinin koordinatlarının dışında $f''(x_1)$ ve $f''(x_n)$ değerlerinin de verilmesi gerekir. Bunun nedeni Cubic Spline'ın daha başarılı yapılabilmesi içindir. Eğer $f''(x_1)$ ve $f''(x_n)$ değerleri sıfır olarak alınırsa bu durumda "Doğal Cubic Spline" yapılmış olur. $F''(x_1)$ ve $f''(x_n)$ değerleri hakkında hiçbir bilgimiz yoksa o zaman $[x_1, x_n]$ aralığının dışında x_0 ve x_{n+1} gibi iki dış nokta alınmak sureti ile Cubic Spline yapılmaya çalışılır. Yani eğrimizi x_0 ve x_{n+1} noktalarından geçiyormuş gibi düşünürüz. X_0 ve x_{n+1} noktaları yardımıyla $f''(x_1)$ ve $f''(x_n)$ değerleri bulunur. Bu durumda;

$$P_i'(x) = f''(x_i)\frac{(x - x_{i-1})^2}{2h_i} - f''(x_{i-1})\frac{(x_i - x)^2}{2h_i} + A_i$$

$$x = x_i \qquad P_i'(x_i) = [\frac{(x_i - x_{i-1})^2}{2h_i} - \frac{h_i}{6}]f''(x_i) + \frac{f(x_i) - f(x_{i-1})}{h_i} + \frac{h_i}{6}f''(x_{i-1})$$

$$i = 1 \qquad f''(x_1) = \frac{3}{h_1}[P_1'(x_1) - \frac{f(x_1) - f(x_0)}{h_1} - \frac{h_1}{6}f''(x_0)] \qquad (1.6)$$

$$i = n + 1 \qquad f''(x_n) = \frac{6}{h_{n+1}}[P_{n+1}'(x_{n+1}) - \frac{h_{n+1}}{3}f''(x_{n+1}) - \frac{f(x_{n+1}) - f(x_n)}{h_{n+1}}] \qquad (1.7)$$

(1.6) ve (1.7) denklemlerinde görüldüğü gibi $P'_1(x_1)$, $P'_{n+1}(x_{n+1})$, $f''(x_0)$, $f''(x_{n+1})$ değerleri bilinmiyor. Bu değerler ise eğrimizi hangi eğriye yaklaştırmak istiyorsak o eğrinin denkleminden hesaplanır. Örneğin eğrimizi $y=x^3$ parabolüne yakınlaştırabilirz. Dolayısıyla $P'_1(x_1)$, $P'_{n+1}(x_{n+1})$, $f''(x_0)$, $f''(x_{n+1})$ değerleri $y=x^3$ denkleminden hesaplanır. Dış nokta olarak da sadece x_0 veya x_{n-1} noktasını alabiliriz. Yani her iki dış noktanın ikisini de birden dahil etmek zorunda değiliz.

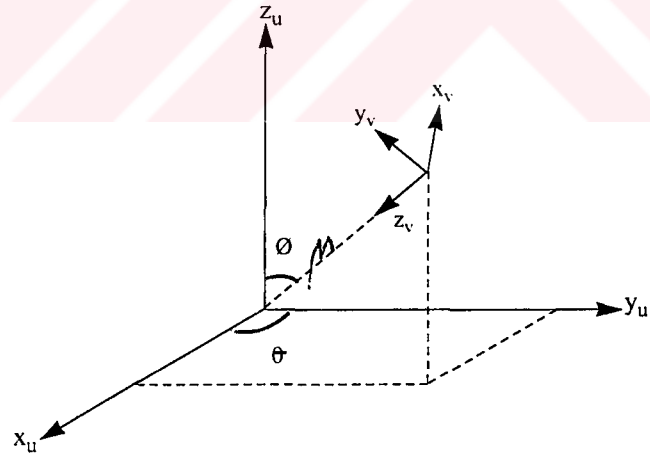
2. Ekranda Üç Boyutlu Görüntünün Oluşturulması :

Bilgisayar ekranında üç boyutlu görüntü, koordinat dönüşümü yapılarak ekrana izdüşüm ile gerçekleştirilir. Üç boyutlu uzayda iki koordinat takımı ele alınır. Bunlardan birincisi noktanın üzerinde bulunduğu ve hareketli olan Hareketli Koordinat Takımı (HKT = uzay koordinat takımı), ikincisi ise gözün üzerinde olduğu varsayılan ve sabit olan Görüntü Koordinat Takımıdır (GKT = view koordinat takımı). Noktanın görüntüsünün HKT'den GKT'ye dönüşümü yapıldıktan sonra ekrana izdüşüm yapılarak görüntü ekrana düşürülmüş olur.

2.1 Uzay Koordinat Takımından View Koordinat Takımına Dönüşüm

Yukarıda da belirtildiği gibi iki koordinat takımı ele alınıyor. Bunlar HKT ile GKT'dir. HKT'nin hareketi sağ el kuralına göre, GKT'nin hareketi ise sol el kuralına göre belirlenmektedir.

İlk olarak HKT'deki bir maddesel noktanın (cismin) GKT'deki bir göz noktasına göre dönüşümü yapılmaktadır. Bu dönüşümün amacı, HKT'deki cismin noktalarının GKT'deki sabit bir göz noktasına göre nasıl bir konumda olduğunu belirlemektir. Bu dönüşüm şu şekilde yapılmaktadır :



Şekil 2.1.1

Şekilden de görüldüğü gibi GKT'nin Z-ekseni HKT'nin örijininine çakışıktır.

$$[X_v\ Y_v\ Z_v\ 1] = [X_u\ Y_u\ Z_u\ 1]\ T_{view}$$

Yukarıdaki eşitlik yardımıyla dönüşüm sağlanır.

$[X_u\ Y_u\ Z_u\ 1]$: HKT'deki noktaların koordinatları.
$[X_v\ Y_v\ Z_v\ 1]$: GKT'deki noktaların koordinatları
T_{view} : HKT'den GKT'ye dönüşümü sağlayan (4X4) lük matris olmak üzere.

Buna göre Şekil-2.1.1'den

$$T_x = \mu Sin\phi Cos\theta \qquad T_y = \mu Sin\phi Sin\theta \qquad T_z = \mu Cos\phi$$

olarak bulunur. T_{view} matrisini oluşturmak için ilk önce HKT'yi GKT'nin konumuna öteleriz. Öteleme matrisi şu şekildedir :

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -T_x & -T_y & -T_z & 1 \end{pmatrix}$$

Yukarıdaki T_x, T_y, T_z değerlerini T_1 matrisininde yerine yazarsak T_1 matrisi;

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\mu Cos\theta Sin\phi & -\mu Sin\theta Sin\phi & -\mu Cos\phi & 1 \end{pmatrix}$$

şeklinde oluşacaktır. İkinci adım olarak HKT'yi saatin hareketi yönünde Z-ekseni etrafında $90 - \theta$ kadar döndürürsek;

$$T_2 = \begin{pmatrix} Sin\theta & Cos\theta & 0 & 0 \\ -Cos\theta & Sin\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

matrisini elde ederiz. Üçüncü adım olarak HKT'yi saatin hareketinin tersi yönde X-ekseni etrafında $180 - \phi$ kadar döndürürsek;

$$T_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -Cos\phi & -Sin\phi & 0 \\ 0 & Sin\phi & -Cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

matrisini elde ederiz. Son adımda HKT sol el kuralı ile koordinat takımına dönüştürülür. Bunu da aşağıdaki gibi ifade edebiliriz:

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

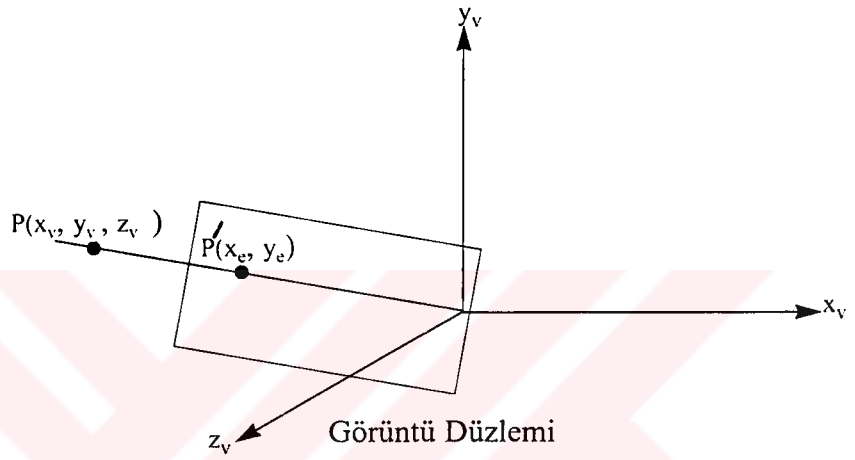
Bütün bu dönüşümler yardımıyla HKT (Hareketli Koordinat Takımı) ile GKT (Görüntü Koordinat Takımı) çakıştırılmış oldu. Bu dönüşümler ile elde ettiğimiz T_1, T_2, T_3 ve T_4 matrislerini sırası ile matris çarpımına tabi tutarsak ;

$$T_{view} = [T_1 T_2 T_3 T_4] \begin{pmatrix} -Sin\theta & -Cos\theta Cos\varphi & -Cos\theta Sin\varphi & 0 \\ Cos\theta & -Sin\theta Cos\phi & -Sin\theta Sin\varphi & 0 \\ 0 & Sin\phi & -Cos\phi & 0 \\ 0 & 0 & \mu & 1 \end{pmatrix}$$

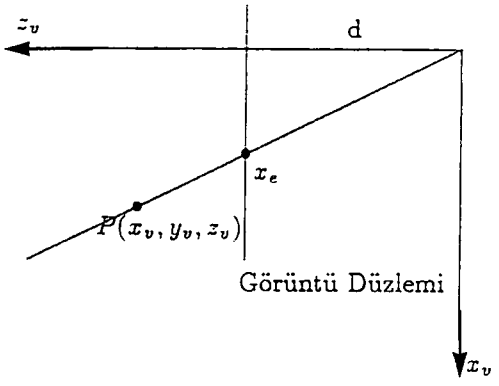
olarak T_{view} matrisini elde etmiş oluruz.

2. 2 Görüntü Düzlemine (Bilgisayar Ekranına) İzdüşüm :

HKT'deki bir (x_u, y_u, z_u) noktasının GKT'ye dönüşüm yapılarak elde edilen (x_v, y_v, z_v) noktasının görüntü düzlemine izdüşümü;

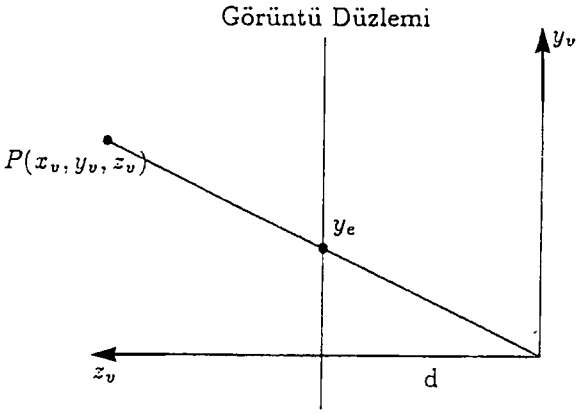


Şekil 2.2.1



(y ekseni boyunca bakılıyor)

Şekil 2.2.2



(x ekseni boyunca bakılıyor)

Şekil 2.2.3

Buradaki d parametresi görüntü düzleminin orijine olan uzaklığıdır.

Şekil-2.2.2 ve Şekil-2.2.3 deki üçgenlerin benzerliğinden,

$$\frac{x_e}{x_v} = \frac{d}{z_v} \quad , \quad \frac{y_e}{y_v} = \frac{d}{z_v}$$

bağıntıları elde edilir. Buradan;

$$x_e = \frac{x_v\, d}{z_v} \quad , \quad y_e = \frac{y_v\, d}{z_v}$$

olarak yazılabilir. Bunu (4X4'lük matris şeklinde yazmak istersek,

$$[x_e \;\; y_e \;\; d \;\; 1] = [\frac{x_v d}{z_v} \;\; \frac{y_v d}{z_v} \;\; d \;\; 1]$$

olur. İfadeyi aşağıdaki gibi düzenlersek,

$$[\frac{x}{w} \;\; \frac{y}{w} \;\; \frac{z}{w} \;\; 1] = [x_e \;\; y_e \;\; d \;\; 1] \qquad (w = \frac{z_v}{d})$$

$$[x \;\; y \;\; z \;\; w] = [x_v \;\; y_v \;\; z_v \;\; \frac{z_v}{d}] = PT_{pers},$$

şeklini alır. Böylece

$$T_{pers} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(4X4)'lük matris şeklinde yazılmış olur.

3. Yüzey Türetme

Cubic Spline yardımıyla çizilen eğriden daha önce de bahsedildiği gibi iki türlü yüzey oluşturuluyor. Bu yüzeyler :

1- öteleme yüzeyi
2- döndürme yüzeyi

olarak iki yüzeydir. Eğrimizi üç boyutlu uzayda xz-düzlemi içinde kabul ediyoruz.

3.1 Öteleme Yüzeyi :

Eğriyi belirleyen noktaların xz-düzlemi içerisinde olduğunu belirtmiştik. Buradan anlaşılacağı gibi belirlenen noktaların y koordinatları sabit ve birbirlerine eşit olacak şekilde alınmaktadır. İlk olarak Cubic Spline yardımıyla eğri oluşturulur. Öteleme işlemi y-ekseni boyunca yapılır. Öteleme işlemi, eğrinin bütün noktaları üzerinde yapılmaktadır. Bu nedenle eğriyi ötelerken eğri üzerindeki "bütün noktaların" sadece y koordinatları değişecektir. Eğrinin bütün noktalarının y koordinatlarına belirli bir sabit artım verilerek, eğri ötelenmiş olur. Bu işlem belirli bir sayıda tekrarlanarak öteleme yüzeyi oluşturulur. Eğriyi öteledikten sonra ikinci yüzey, tarama yüzeyi olup, yüzey üzerindeki doğrulardan ibarettir. Bu doğrular da oluşturulduktan sonra öteleme yüzeyi tamamıyle elde edilmiş olur.

Burada dikkat edilmesi gereken nokta; eğriyi her öteleyişte oluşturulan yeni eğri Cubic Spline vasıtasıyla yapılmamaktadır. Bunun nedeni ise ötelenmiş her düzlem eğrisinin, Cubic Spline vasıtasıyla elde edilen eğriden farkının sadece y koordinatına sahip olmasındandır. Yani bütün eğrilerin x, z koordinatları aynıdır.

3.2 Döndürme Yüzeyi :

Döndürme yüzeyi, eğriyi z-ekseni etrafında saat yönünün tersi yönde 360 derece döndürerek elde edilmektedir. Döndürme işlemi, öteleme yüzeyinde olduğu gibi "bütün noktalar üzerinde" yapılmaktadır. Bunun sebebi ise Cubic Spline'i belirleyen ikinci türev değerlerinde, döndürme işlemi esnasında nasıl bir değişikliğin olduğunun tespit edilememesinden kaynaklanmaktadır. Döndürme işlemi z-ekseni etrafında olduğu için noktaların x ve y koordinatları değişecek ancak z koordinatları değişmeyecektir. Döndürme işlemi şu şekilde yapılmaktadır.

İlk olarak Cubic Spline ile eğri xz-düzlemi üzerinde oluşturulur. Daha sona eğrinin her noktasının xy-düzleminde orijine olan uzaklıkları (r=yarıçap) ve x-ekseni ile yapmış olduğu açıları (θ) hesaplanır. Bu hesaplama şu formüllerle yapılır:

$$r = \sqrt{x^2 + y^2} \quad , \qquad \theta = \text{arctg}(y/x)$$

Bu hesaplamalardan sonra θ açılarına belirli bir sabit artım verilerek eğrinin bütün noktalarının dönmüş hali olan yeni noktaların x', y' koordinatları hesaplanır. Bu hesaplamalar da şu formüllerle yapılır:

$$x' = r\,\text{Cos}\theta \,, \quad y' = r\text{Sin}\theta \quad , z' = z$$

Eğriyi 360 derece döndürerek, döndürme yüzeyi tamamlanmış olur. Dönmüş yüzeyin "tarama yüzeyi" ise çemberlerdir. Bu çemberler xy-düzlemindedirler. Ancak çemberler farklı $z = c_i$ $(i = 1,...,n)$ koordinatlarına sahiptirler. Bu çemberlerin oluşturulmasıyla döndürme yüzeyi elde ediir.

4. Programın Algoritması

Program, Pascalda yazılmıştır. Program ilk olarak Cubic Spline vasıtasıyla 3.dereceden $P_1(x)$, $P_2(x)$,...,P_{n-1} polinomlarını hesaplamaktadır. Daha sonraki aşamada 3.derece polinomları vasıtasıyla eğrinin bütün noktaları hesaplanmaktadır.

Bilgisayar ekranında, eğriye ölçekleme yapılmıştır. Bilindiği gibi bilgisayar ekranındaki her bir görüntü pixeller yardımıyla gerçekleşir. Ölçekleme sayesinde eğri daha ayrıntılı bir şekilde çizdirilebilir. Böylece yüzey üzerinde detaylı analiz yapılabilir.

Eğrinin bütün noktalarının uzay koordinatları ve ekran koordinatları için bellekte yer ayrılmıştır. Bu ayrılan belleğin uzunluğu nokta sayısına ve ölçü birimine göre değişmektedir. (Dynamic bellek kullanımı). Eğrinin bütün noktaları için bellekte yer ayrılmasının nedeni "öteleme yüzeyi" ve "döndürme yüzeyi" konularında anlatılmıştır. Programın temel algoritmasını dinamik bellek kullanımı oluşturmaktadır. Çünkü ekran değerleri dinamik bellekte saklanmaktadır. Bu bellek bölümüne gerektiğinde eğrinin ekran koordinatlarına ulaşılarak eğri çizdirilir. Program eğriyi, eğrinin bütün uzay noktalarının ekran koordinatlarını hesapladıktan sonra çizmektedir.

Program Cubic Spline'in başarılı bir şekilde çalışması için kontrol yapmaktadır. Program, ikinci türevlerini hesaplarken katsayılar matrisinin determinantına göre gerçekleşir. Determinant değeri 10-E12 değerinden küçük ise Cubic Spline gerçekleştirilmez. Bunun yerine belirlenen noktalardan geçen bir poligon (kırık çizgiler) çizilir.

Programda, ikinci türev değerlerinin oluşturduğu denklem sistemi Gauss eliminasyon yöntemi ile çözülmektedir. Gauss eliminasyon yöntemi ana program içerisinde olmayıp, ayrı bir unit program olarak tasarlanmıştır.

Sonuç

Amacımız düzlemdeki eğriyi y-ekseni boyunca öteleme yaparak öteleme yüzeyi, z-ekseni etrafında döndürerek döndürme yüzeyi ve aynı eksenle yapmış olduğu açıyı değiştirerek eksen yüzeyi oluşturmaktır.

Düzlemdeki eğri Cubic Spline yöntemiyle amaçlandığı gibi yapılmıştır. Bu eğri yardımıyla öteleme yüzeyi ve döndürme yüzeyi oluşturulmuştur. Ancak eksen yüzeyi tamamlanamamıştır. Bunun sebebi ise eğriyi z-ekseni etrafında ve eksen açısını değiştirerek döndürürken, ikinci türev değerlerinin nasıl bir değişikliğe uğradığı hakkında gerekli incelemeyi yapabilecek zaman olmadığından dolayıdır. Bu işlemin sadece teorik kısmı incelenmiştir.

Sadece ön fikir olması açısından, eksen açısı değiştirildiğinde nokta küresel koordinatlarda hareket ettirilmiş olur. Küresel koordinatlarda bir noktanın x,y,z koordinatları:

$$x = r\cos\theta\sin\phi,\quad y = r\sin\theta\sin\phi,\quad z = r\cos\phi$$

şeklindedir.

Burada r = noktanın orijine olan uzaklığı,
θ = noktanın x-ekseni ile yapmış olduğu açı,
Ø = noktanın z-ekseni ile yapmış olduğu açıdır.

Bu veriler yardımı ile daha değişik yüzeyleri çizdirme imkanı bulunabilecektir.

Bundan sonraki aşamada gerçekleştirmek istediğim asıl amaç :
Verilen herhangi bir fonksiyonun (trigonometrik, ters trigonometrik, üstel, polinom, logaritmik veya bunların herhangi bir formu) tanım aralığı kısıtlaması olmadan bilgisayarın CPU'sunu kullanmak sureti ile Dynamic Belleğe bağımlı kalmadan aynı anda istenildiğinde fonksiyonu;

- Uzayda
- Düzlemde

çizdirebilecek bir program tasarlamaktır. Söz konusu çalışmaya yeni başlamış bulunmakta ve şu anda matematiksel modeli üzerinde çalışmaktayım.

KAYNAKLAR

1. Fröberg Eric - Carl
Introduction to Numerical Analysis

2. A. Teukolsky Saul
H.Press William
P.Flannery Brain
T.Vetterling William
Numerical Recipes in C

3. Alan Watt
Fundementals of Three-Dimensional Computer Graphics

```
Program YUZEY_GRAFIGI_ELDE_ETME;
{*****************************************************************************}
{                                Müslüm Özışık                                }
{                          Y.T.Ü Fen Bilimleri Enstitüsü                      }
{                              Master Programı                                }
{                                 9402006                                     }
{*****************************************************************************}

Uses Crt,Graph,TEZ_ALT_PROG;

Type
   NoktaDizi       = Array [0..25] of extended;
   UzayNoktaPtr    = ^UzayNokta;
   Uzay_Nokta      = record Xx,Yy,Zz,t : extended;
                     end;
   ViewPtr         = ^View;
   View            = record Xs,Ys : extended;
                     end;
   Pol             = Array [0..3] of extended;
   UzayDizi        = Array [1..25] of Uzay_Nokta;

Const
   ikinciTurev : Dizi =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   Yogunluk_1   : byte = 0;
   Yogunluk_2   : byte = 0;
   olcu = 20;
   d    = 100;

   Eksen_Takimi : Array [1..9] of Uzay_Nokta = ( (Xx:50; Yy:0; Zz:0; t:1),
                                                 (Xx:0; Yy:50; Zz:0; t:1),
                                                 (Xx:0; Yy:0; Zz:50; t:1),
                                                 (Xx:-30; Yy:0; Zz:0; t:1),
                                                 (Xx:0; Yy:-30; Zz:0; t:1),
                                                 (Xx:0; Yy:0; Zz:-30; t:1),
                                                 (Xx:50; Yy:5; Zz:0; t:1),
                                                 (Xx:5; Yy:50; Zz:0; t:1),
                                                 (Xx:0; Yy:5; Zz:50; t:1) );

   GozNoktasi : Uzay_Nokta = (Xx:2; Yy:2; Zz:2; t:1 );
```

```
Var
   h,A,b                  : NoktaDizi;
   Cx,Cy,Cz               : NoktaDizi;
   Dz_Uzay                : UzayDizi;
   TurevMatris            : Matris;
   TView                  : Array [1..4,1..4] of extended;

    r,fi,teta                          : extended;
   UcunPol                             : Array [2..25] of Pol;
   Yed_Ek_Takimi                       : Array [1..9] of Uzay_Nokta;
   NoktaSayisi,i,j,k                   : integer;
   XMerkez,YMerkez                     : integer;
   Gd,Gm                               : integer;
   P1,HeapAdresi1                      : UzayNoktaPtr;
   P2,HeapAdresi2                      : ViewPtr;
   UzayPixel1,UzayPixel2               : Uzay_Nokta;
   EkranNokta1,EkranNokta2             : View;
   Oteleme_Adimi                       : extended;
   Ote_Say                             : integer;
   m1,Yuzey_Kodu                       : integer;
   Prizma                              : Array [1..8] of Uzay_Nokta;
   PrizmaEkran                         : Array [1..8] of View;
   Ch                                  : Char;

   Y_ilkTurevy1,Y_y0Turev,y1Turev      : extended;
   Pointx1,Pointx2,Pointy1,Pointy2     : extended;
   R1,alfa,HeapAdresi3,HeapAdresi4     : ^extended;
   Segment1,offset1,Segment2,offset2,
   Segment3,offset3,Segment4,offset4,
   size1,size2,size3,size4             : integer;
   adres1,adres2,adres3,adres4         : integer;
{
*****************************************************************************************
}
Procedure TViewMatrisi;
begin
{ *****************************************************************************************
                    TView Matrisinin Tanımlanması
*****************************************************************************************
}
```

```
      TView[1,1]:=-Sin(teta*pi/180);
      TView[1,2]:= Cos(teta*pi/180)*Cos(fi*pi/180);
      TView[1,3]:= Cos(teta*pi/180)*Sin(fi*pi/180);
      TView[1,4]:= 0;
      TView[2,1]:= Cos(teta*pi/180);
      TView[2,2]:= Sin(teta*pi/180)*Cos(fi*pi/180);
      TView[2,3]:= Sin(teta*pi/180)*Sin(fi*pi/180);
      TView[2,4]:= 0;
      TView[3,1]:= 0;
      TView[3,2]:= Sin(fi*pi/180);
      TView[3,3]:=-Cos(fi*pi/180);
      TView[3,4]:= 0;
      TView[4,1]:= 0;
      TView[4,2]:= 0;
      TView[4,3]:= r;
      TView[4,4]:= 1;

end;

{

*******************************************************************************************

}
Procedure ViewPlaneSpace (var UzayNokta1 : Uzay_Nokta );
Var
  UzayNokta2 : Uzay_Nokta;
  begin

{
*******************************************************************************************
 Genel Koordinat Sisteminden View Koordinat Sistemine Gecis
*******************************************************************************************
                                                                                          }
  UzayNokta1.Zz :=-UzayNokta1.Zz;

  UzayNokta2.Xx := UzayNokta1.Xx*Tview[1,1]+UzayNokta1.Yy*Tview[2,1]+
           UzayNokta1.Zz*Tview[3,1]+UzayNokta1.t*Tview[4,1];

  UzayNokta2.Yy := UzayNokta1.Xx*Tview[1,2]+UzayNokta1.Yy*Tview[2,2]+
           UzayNokta1.Zz*Tview[3,2]+UzayNokta1.t*Tview[4,2];
```

```
  UzayNokta2.Xx := UzayNokta1.Xx*Tview[1,3]+UzayNokta1.Yy*Tview[2,3]+
            UzayNokta1.Zz*Tview[3,3]+UzayNokta1.t*Tview[4,3];

  UzayNokta2.t := 1;

            if ( UzayNokta2.Zz)  then  UzayNokta2.Zz:=0;
               UzayNokta1 :=UzayNokta2;

  end;
{
*******************************************************************************************
}

Procedure ViewPlaneScreen (UzayNokta1      : Uzay_Nokta;
                  Var Ekran_Nokta : View);
  begin

{
*******************************************************************************************
View Koordinat sistemindeki Noktalarin Ekrandaki Konumlari
*******************************************************************************************
}
  if ( UzayNokta1.Zz) then
     begin
        Ekran_Nokta.Xs := UzayNokta1.Xx/(UzayNokta1.Zz/d);
        Ekran_Nokta.Ys := UzayNokta1.Yy/(UzayNokta1.Zz/d);
    end
      else
         begin
            Ekran_Nokta.Xs := UzayNokta1.Xx/d;
            Ekran_Nokta.Ys := UzayNokta1.Yy/d;
         end;
    end;

{
*******************************************************************************************
}

Procedure ViewPlanePtr ( UzayNokta1 : Uzay_Nokta;
                                Var p     : ViewPtr;
                              Var adres  : integer;
                    size,Segment,offset : integer );
```

```
Var
  UzayNokta2  : Uzay_Nokta;
  Ekran_Nokta : View;

  begin

{
*****************************************************************************************
 View Koordinat Sistemindeki Noktalarin Ekrandaki Konumlari
*****************************************************************************************
}

   UzayNokta1.Zz :=-UzayNokta1.Zz;
   UzayNokta2.Xx := UzayNokta1.Xx*Tview[1,1]+UzayNokta1.Yy*Tview[2,1]+
            UzayNokta1.Zz*Tview[3,1]+UzayNokta1.t*Tview[4,1];

   UzayNokta2.Yy := UzayNokta1.Xx*Tview[1,2]+UzayNokta1.Yy*Tview[2,2]+
            UzayNokta1.Zz*Tview[3,2]+UzayNokta1.t*Tview[4,2];

   UzayNokta2.Xx := UzayNokta1.Xx*Tview[1,3]+UzayNokta1.Yy*Tview[2,3]+
            UzayNokta1.Zz*Tview[3,3]+UzayNokta1.t*Tview[4,3];
   UzayNokta2.t    := 1;

 if ( UzayNokta2.Zz)  then UzayNokta2.Zz:=0;

{
*****************************************************************************************
        View Koordinat sistemindeki Noktalarin Ekrandaki Konumlari
*****************************************************************************************
}

  if ( UzayNokta1.Zz) then
    begin
       Ekran_Nokta.Xs := UzayNokta2.Xx/(UzayNokta2.Zz/d);
       Ekran_Nokta.Ys := UzayNokta2.Yy/(UzayNokta2.Zz/d);
    end
      else
         begin
            Ekran_Nokta.Xs := UzayNokta2.Xx/d;
            Ekran_Nokta.Ys := UzayNokta2.Yy/d;
         end;
    p^       := Ekran_Nokta;
```

```
    adres  := adres+size;
    P      := Ptr( Segment, offset+adres );
end;

{     .
*****************************************************************************************
}

Procedure EksenTakimi;
Var
  u : integer;
  begin
     Move ( Eksen_Takimi,Yed_Ek_Takimi,Size of (Eksen_Takimi) );
     XMerkez := 50;
     YMerkez := 50;
     SetLineStyle(SolidLn,0,NormWidth);
     for u:=1 to 3 do
       begin
          ViewPlaneSpace(Eksen_Takimi[u]);
          ViewPlaneScreen(Eksen_Takimi[u],EkranNokta1);
          Line (XMerkez,YMerkez,XMerkez+Round(EkranNokta1.Xs),
                              YMerkez+Round(EkranNokta1.Ys) );
       end;
     SetLineStyle( DashedLn, 0, NormWidth);
     for u:=4 to 6 do
       begin
          ViewPlaneSpace(Eksen_Takimi[u]);
          ViewPlaneScreen (Eksen_Takimi[u],EkranNokta1);



 Line (XMerkez,YMerkez,XMerkez+Round(EkranNokta1.Xs),
                          YMerkez+Round(EkranNokta1.Ys) );
       end;
     SetLineStyle(SolidLn, 0, NormWidth);

     ViewPlaneSpace( Eksen_Takimi[7] );
     ViewPlaneScreen (Eksen_Takimi[7], EkranNokta1 );

OutTextXY(XMerkez+Round(EkranNokta1.Xs),YMerkez+Round(EkranNokta1.Ys),'X');
          ViewPlaneSpace(Eksen_Takimi[8] );
          ViewPlaneScreen (Eksen_Takimi[8], EkranNokta1 );
```

```
OutTextXY(XMerkez+Round(EkranNokta1.Xs),YMerkez+Round(EkranNokta1.Ys), 'Y');
            ViewPlaneSpace( Eksen_Takimi[9] );
            ViewPlaneScreen ( Eksen_Takimi[9], EkranNokta1 );

OutTextXY(XMerkez+Round(EkranNokta1.Xs),YMerkez+Round(EkranNokta1.Ys), 'Z');
            XMerkez := 319;
            YMerkez := 239;
      Move ( Yed_Ek_Takimi,Eksen_Takimi,Size of (Yed_Ek_Takimi) );
  end;

{
*******************************************************************************************
}

Procedure Cember ( rr , t : extended );
Var
   I : integer;
   begin
      for I:=Round(t)+1 to 360 do
         begin
            UzayPixel2.Xx := rr*Cos(I*pi/180);
            UzayPixel2.Yy := rr*Sin(I*pi/180);
            UzayPixel2.Zz := P1^.Zz;

           ViewPlaneSpace ( UzayPixel2 );
            ViewPlaneScreen ( UzayPixel2,EkranNokta2 );
            Line( XMerkez+Round(olcu*EkranNokta1.Xs),
                     YMerkez+Round((olcu/4)*EkranNokta1.Ys),
                XMerkez+Round(olcu*EkranNokta2.Xs),
                     YMerkez+Round((olcu/4)*EkranNokta2.Ys));

  Move ( UzayPixel2,UzayPixel1,Size of (UzayPixel2) );
            Move ( EkranNokta2,EkranNokta1,Size of (EkranNokta2) );
         end;
   end;

{
*******************************************************************************************
}
```

```
Procedure Poligon ( DD : UzayDizi );
Var
   h : integer;
   begin
      UzayPixel1 := Dz_Uzay[1];
      ViewPlaneSpace ( UzayPixel1 );
      ViewPlaneScreen ( UzayPixel1,EkranNokta1 ):
      for h := 2 to NoktaSayisi do
         begin
            UzayPixel2 := Dz_Uzay[h];
            ViewPlaneSpace ( UzayPixel2 );
            ViewPlaneScreen ( UzayPixel2,EkranNokta2 );
            Line( XMerkez+Round(olcu*EkranNokta1.Xs),
                     YMerkez+Round((olcu/4)*EkranNokta1.Ys),
                  XMerkez+Round(olcu*EkranNokta2.Xs),
                     YMerkez+Round((olcu/4)*EkranNokta2.Ys));

            Move ( UzayPixel2,UzayPixel1,Size of (UzayPixel2) );
            Move ( EkranNokta2,EkranNokta1,Size of (EkranNokta2) );
         end;
   end;

{
*****************************************************************************************
}

Procedure PrizmaCiz;
Var
   kk   : byte;
   fark : extended;

   begin
      Prizma[1].Xx := Cx[1];
      Prizma[1].Yy := Cy[1];
      Prizma[1].Zz := Cz[1];
      Prizma[1].t  := 1;

      Prizma[4].Xx := Cx[NoktaSayisi];
      Prizma[4].Yy := Cy[NoktaSayisi];
      Prizma[4].Zz := Cz[NoktaSayisi];
      Prizma[4].t  := 1;
```

```
if ( Prizma[1].Zz > Prizma[4].Zz ) then
                                    Prizma[1].Zz := Prizma[4].Zz
   else
      Prizma[3].Zz := Prizma[1].Zz;

Move ( Prizma[1],Prizma[2],Size  of (Prizma[1]) );
Prizma[2].Yy := Cy[1]+otestep;

Move ( Prizma[4],Prizma[3],Size  of  (Prizma[4]) );
Prizma[3].Yy := Cy[NoktaSayisi]+otestep;

for kk := 1 to 4 do
   begin
      UzayPixel1 := Prizma[kk];
      ViewPlaneSpace ( UzayPixel1 );
      ViewPlaneScreen ( UzayPixel1,PrizmaEkran[kk] );
   end;

for kk := 1 to 3 do
   Line ( XMerkez+Round(olcu*PrizmaEkran[kk].Xs),
             YMerkez+Round((olcu/4)*PrizmaEkran[kk].Ys),
        XMerkez+Round(olcu*PrizmaEkran[kk+1].Xs),
             YMerkez+Round((olcu/4)*PrizmaEkran[kk+1].Ys));

   Line ( XMerkez+Round(olcu*PrizmaEkran[4].Xs),
             YMerkez+Round((olcu/4)*PrizmaEkran[4].Ys),
        XMerkez+Round(olcu*PrizmaEkran[1].Xs),
             YMerkez+Round((olcu/4)*PrizmaEkran[1].Ys));

        Prizma[1].Zz := Cz[1];
        Prizma[2].Zz := Cz[1];
        Prizma[4].Zz := Cz[NoktaSayisi];
        Prizma[3].Zz := Cz[NoktaSayisi];

   Move ( Prizma[1],Prizma[5],Size  of  (Prizma[1]) );
   Prizma[5].Zz := Prizma[1].Zz-olcu*1;

   Move ( Prizma[2],Prizma[6],Size  of  (Prizma[2]) );
   Prizma[6].Zz := Prizma[2].Zz-olcu*1;
```

```
        Move ( Prizma[3],Prizma[7],Size  of  (Prizma[3]) );
        Prizma[7].Zz := Prizma[3].Zz-olcu*1;

        Move ( Prizma[4],Prizma[8],Size  of  (Prizma[8]) );
        Prizma[8].Zz := Prizma[4].Zz-olcu*1;

      for kk := 1 to 8 do
        begin
          UzayPixel1 := Prizma[kk];
          ViewPlaneSpace ( UzayPixel1 );
          ViewPlaneScreen ( UzayPixel1,PrizmaEkran[kk] );
        end;

      for kk := 1 to 4 do
        Line ( XMerkez+Round(olcu*PrizmaEkran[kk].Xs),
                 YMerkez+Round((olcu/4)*PrizmaEkran[kk].Ys),
               XMerkez+Round(olcu*PrizmaEkran[kk+4].Xs),
                 YMerkez+Round((olcu/4)*PrizmaEkran[kk+4].Ys));

      for kk := 5 to 7 do
        Line ( XMerkez+Round(olcu*PrizmaEkran[kk].Xs),
                 YMerkez+Round((olcu/4)*PrizmaEkran[kk].Ys),
               XMerkez+Round(olcu*PrizmaEkran[kk+1].Xs),
                 YMerkez+Round((olcu/4)*PrizmaEkran[kk+1].Ys));
        Line ( XMerkez+Round(olcu*PrizmaEkran[8].Xs),
                 YMerkez+Round((olcu/4)*PrizmaEkran[8].Ys),
               XMerkez+Round(olcu*PrizmaEkran[5].Xs),
                 YMerkez+Round((olcu/4)*PrizmaEkran[5].Ys));
  end;

{
*******************************************************************************************
}

Procedure aktarma;
Var
  I : integer;
  begin
    for I := 1 to NoktaSayisi do
      begin
        Dz_Uzay[1].Xx := Cx[1];
        Dz_Uzay[1].Yy := Cy[1];
```

```
            Dz_Uzay[1].Zz := Cz[1];
            Dz_Uzay[1].t  := 1;
          end;
    end;

{
*****************************************************************************************
}

Procedure CubicSpline ( XX,YY          : NoktaDizi;
                        ilkTurevy1,y0turev  : extended;
                        n                   : integer );
    begin
       { h[i] degerlerinin hesabi }
       for i := 0 to NoktaSayisi-1 do
          h[i+1] := abs(XX[i+1]-XX[i]);
       if ( ilkTurevy1  <> 0  then
          ikinciTurev[1] := (3/h[1])*(ilkTurevy1-(XX[1]-XX[0])/h[1]
                            -(h[1]/6)*y0Turev)
         else
           ikinciTurev[1] := 0;
           ikinciTurev[n] := 0;
           y1Turev        := ikinciTurev[1];

{
*****************************************************************************************
        ikinci turev matrisinin olusturulmasi
*****************************************************************************************
}

      TurevMatris[1,1]    := (h[3]-h[2])/3;
      TurevMatris[1,2]    := h[3]/6;
      TurevMatris[1,n-1]  := ((YY[3]-YY[2])/h[3]
                          -((YY[2]-YY[1])/h[2])-h[2]/6*ikinciTurev[1]);
      j := 1;
      for i := 3 to n-2 do
         begin
            TurevMatris[i-1,j]     := h[i]/6;
            TurevMatris[i-1,j+1 ]  := (h[i+1]-h[i])/3;
            TurevMatris[i-1,j+2]   := h[i+1]/6;
            TurevMatris[i-1,n-1]   := (YY[i+1]-YY[i])/h[i+1]-(YY[i]-YY[i-1])/h[i];
```

```
            j := j+1;
        end;
    TurevMatris[n-2,n-3] := (h[n-1])/6;
    TurevMatris[n-2,n-2] := (h[n]-h[n-1])/3;
    TurevMatris[n-2,n-1] := ((YY[n]-YY[n-1])/h[n] - ((YY[n-1]-YY[n-2])/h[n-1])
                                                    -h[n]/6*ikinciTurev[n]);

    Gauss( TurevMatris, ikinciTurev,n-2,n-1,False );

{
*****************************************************************************************
          IkinciTurev Dizisinin Duzenlenmesi
*****************************************************************************************
}
    for i:=1 to n-2 do
        ikinciTurev[n-i] := ikinciTurev[n-i-1];
        ikinciTurev[1]   := y1Turev;

{
*****************************************************************************************
              A[i] ve B[i] Degerlerinin Hesaplanması
*****************************************************************************************
}
    for i := 2 to n do
        begin
            A[i] := ((YY[i]-YY[i-1])/h[i])-
                    h[i]/6*(ikinciTurev[i]-ikinciTurev[i-1]);
            B[i] := YY[i]-ikinciTurev[i]*h[i]*h[i]/6-A[i]*XX[i] {;}
        end;

{
*****************************************************************************************
          Polinomların Katsayılarının Hesaplanması
*****************************************************************************************
}

    for i := 2 to n do
        begin
            UcunPol[i,3] := (ikinciTurev[i]-ikinciTurev[i-1])/(6*h[i]);

            UcunPol[i,2] := (XX[i]*ikinciTurev[i-1]-XX[i-1]*
                            ikinciTurev[i])/(2*h[i]);
```

```
            UcunPol[i,1] := (XX[i-1]*XX[i-1]*ikinciTurev[i]-XX[i]*
                        XX[i]*ikinciTurev[i-1])/(2*h[i])+A[i];

            UcunPol[i,0] := (XX[i]*XX[i]*XX[i]*ikinciTurev[i-1] -
                        XX[i-1]*XX[i-1]*XX[i-1]*
                        ikinciTurev[i])/(6*h[i])+B[i];
        end;
    end;

    {
    ********************************************************************************
    }

Procedure PolHesabi ( XX,YY,ZZ          : NoktaDizi;
                      n                 : integer;
                      P,HeapAdresi      : UzayNoktaPtr;
                      adres             : integer;
                      size,Segment,offset : integer;
                      EP,EHeapAdresi    : ViewPtr;
                      Eadres            : integer;
                      Esize,ESegment,Eoffset : integer
                            );

{********************************************************************************
                        Polinomlarin Hesabı
********************************************************************************
}
    begin
        P       := HeapAdresi;
        adres   := 0;
        EP      := EHeapAdresi;
        Eadres  := 0;

        Pointx1 := XX[1];
        Pointy1 := YY[1];
        Pointx2 := 0;

        UzayPixel1.Xx := XX[1];
        UzayPixel1.Zz := Pointy1;
        UzayPixel1.Yy := ZZ[1];
        UzayPixel1.t  := 1;
```

```
      p^ := UzayPixel1;
      ViewPlanePtr ( UzayPixel1,EP,Eadres,Esize,ESegment,Eoffset );

      for.i:= 2 to n do
      begin
         repeat
            if XX[i-1]then
              Pointx2 := Pointx1+(1/olcu)
              else
                 Pointx2 := Pointx1-(1/olcu);

              Pointy2 := UcunPol[i,3]*Pointx2*Pointx2+
                      UcunPol[i,2]*Pointx2*Pointx2+
                      UcunPol[i,1]*Pointx2+UcunPol[i,0];

              UzayPixel2.Xx := PointX2;
              UzayPixel2.Zz := Pointy2;
              UzayPixel2.Yy := ZZ[i];
              UzayPixel2.t     := 1;

              adres := adres+size;
              P     := Ptr( Segment,offset+adres );
              P^    := UzayPixel2;
              ViewPlanePtr ( UzayPixel2,EP,Eadres,Esize,ESegment,Eoffset );

              Pointx1 := Pointx2;
              Pointy1 := Pointy2;

         until ( Pointx2 >= XX[i]-1/olcu );
      end;
   end;

   {
   ********************************************************************************************
   }

Procedure Polcizim ( P,HeapAdresi      : ViewPtr;
                     adres,size        : integer;
                     Segment,offset,m  : integer
                   );
```

```
  begin
     P              := Ptr(Segment,offset);
     adres          := 0;
     EkranNokta1    := P^;

     repeat
        adres            := adres+size;
        P                := Ptr(Segment,offset+adres);
        EkranNokta2      := P^;

  Line ( XMerkez+Round(olcu*EkranNokta1.Xs),
           YMerkez+Round((olcu/4)*EkranNokta1.Ys),

       XMerkez+Round(olcu*EkranNokta2.Xs),
           YMerkez+Round((olcu/4)*EkranNokta2.Ys));

       Move ( EkranNokta2,EkranNokta1,size of (EkranNokta2) );

     until (adres>=m*size-20 )

  end;

  {
 ********************************************************************************
  }

Procedure  BellekAyirma;
  begin
     m1                    := Round ( (Cx[Noktasayisi]-Cx[1])*olcu );
     size1                 := size of ( P1^ );
     adres1                := 0;
     Mark ( HeapAdresi1 );
     P1                    := HeapAdresi1;
     GetMem        ( P1, m1*size1+40 );
     Segment1              := Seg ( P1^ );
     offset1               := ofs ( P1^ );

     size2                 := Sizeof ( P2^ );
     adres2                := 0;
     Mark ( HeapAdresi2 );
     P2                    := HeapAdresi2;
     GetMem        ( P2, m1*size2+40 );
```

```
    Segment2                := Seg ( P2^ );
    offset2                 := ofs ( P2^ );

    size3                   := Sizeof ( R1^ );
    adres3                  := 0;
    Mark ( HeapAdresi3 );

    R1                      := HeapAdresi3;
    GetMem      ( R1, m1*size3 );
    Segment3                := Seg ( R1^ );
    offset3                 := ofs ( R1^ );

    size4                   := sizeof ( alfa^ );
    adres4                  := 0;
    Mark ( HeapAdresi4 );
    alfa                    := HeapAdresi4;
    GetMem      ( alfa, m1*size4 );
    Segment4                := Seg ( alfa^ );
    offset4                 := ofs ( alfa^ );

end;

{
*******************************************************************************
}

Procedure EgriOteleme ( P,HeapAdresi            : UzayNoktaPtr;
                        adres,size,segment,offset : integer;
                        EP,EHeapAdresi          : ViewPtr;
                        Eadres,Esize,Esegment,Eoffset,m : integer
                      );
Var
  u : byte;
  begin
    Oteleme_Adimi  := 0;
    EP             := EHeapAdresi;
    EAdres         := 0;
    PolCizim ( EP,EHeapAdresi,EAdres,Esize,Esegment,Eoffset,m );

    for i := 1 to Round((Ote_Say-1))  do
```

```
        begin
          P                := HeapAdresi;
          adres            := 0;
          EP               := EHeapAdresi;
          Eadres           :=0;
          Oteleme_Adimi    := Oteleme_Adimi + Yogunluk_1/olcu;

          repeat
            UzayPixel1.Xx := P^.Xx;
            UzayPixel1.Yy := P^.Yy+Oteleme_Adimi;
            UzayPixel1.Zz := P^.Zz;

            ViewPlanePtr ( UzayPixel1,EP,Eadres,Esize,Esegment,Eoffset );
            adres := adres + size;
            P := Ptr ( Segment,offset+adres );

          until ( Eadres>= m*Esize );

         PolCizim ( EP,EHeapAdresi,Eadres,Esize,Esegment,Eoffset,m );

        end;
    end;

    {
    ********************************************************************************
    }
```

Yüzey1

```
Procedure Yyzey1;
Var
  k : integer;
  begin
     PolHesabi ( Cx,Cy,Cz,Noktasayisi,P1,HeapAdresi1,adres1,size1,
           segment1,offset1,P2,HeapAdresi2,adres2,size2,
           segment2,offset2 );

     EgriOteleme ( P1,HeapAdresi1,adres1,size1,Segment1,offset1,
           P2,HeapAdresi2,adres2,size2,Segment2,offset2,m1 );

     for k := 1 to Round(m1/Yogunluk_2) do
       begin
          UzayPixel1.Xx := P1^.Xx;
          UzayPixel1.Yy := P1^.Yy;
```

```
        UzayPixel1.Zz := P1^.Zz;
        UzayPixel1.t  := P1^.t;
        ViewPlaneSpace ( UzayPixel1 );
        ViewPlaneScreen ( UzayPixel1,EkranNokta1 );

        UzayPixel2.Xx := P1^.Xx;
        UzayPixel2.Yy := P1^.Yy+Oteleme_Adimi;
        UzayPixel2.Zz := P1^.Zz;
        UzayPixel2.t  := 1;
        ViewPlaneSpace ( UzayPixel2 );
        ViewPlaneScreen ( UzayPixel2,EkranNokta2 );

    Line ( XMerkez+Round(olcu*EkranNokta1.Xs),
         YMerkez+Round((olcu/4)*EkranNokta1.Ys),

    XMerkez+Round(olcu*EkranNokta2.Xs),
         YMerkez+Round((olcu/4)*EkranNokta2.Ys));

        adres1 := adres1+Yogunluk_2*size1;
        P1 := Ptr( Segment1,offset1+adres1 );

    end;

P1              := Ptr ( Segment1, offset1+m1*size1-40 );
UzayPixel1.Xx   := P1^.Xx;
UzayPixel1.Yy   := P1^.Yy;
UzayPixel1.Zz   := P1^.Zz;
UzayPixel1.t    := P1^.t;
ViewPlaneSpace ( UzayPixel1 );
ViewPlaneScreen ( UzayPixel1,EkranNokta1 );

UzayPixel2.Xx   := P1^.Xx;
UzayPixel2.Yy   := P1^.Yy+Oteleme_Adimi;
UzayPixel2.Zz   := P1^.Zz;
UzayPixel2.t    := 1;
ViewPlaneSpace ( UzayPixel2 );
ViewPlaneScreen ( UzayPixel2,EkranNokta2 );

Line ( XMerkez+Round(olcu*EkranNokta1.Xs),
       YMerkez+Round((olcu/4)*EkranNokta1.Ys),
```

```
            XMerkez+Round(olcu*EkranNokta2.Xs),
              YMerkez+Round((olcu/4)*EkranNokta2.Ys));

      PrizmaCiz;
  end;

  {
  ******************************************************************************************
  }
Procedure Yyzey2;
Var
  k                    : integer;
  aci1,aci2,aciartim   : extended;
  yaricap,alfa1        : extended;
  begin
     PolHesabi ( Cx,Cy,Cz,Noktasayisi,P1,HeapAdresi1,adres1,size1,
           segment1,offset1,P2,HeapAdresi2,adres2,size2,
           segment2,offset2 );

     PolCizim  ( P2,HeapAdresi2,adres2,size2,segment2,offset2,m1 );

     repeat
        P1    := Ptr(Segment1,offset1+adres1);
        R1    := Ptr(Segment3,offset3+adres3);
        alfa  := Ptr(Segment4,offset4+adres4);

        UzayPixel1 := P1^;

        R1^   := Sqrt( Sqr(UzayPixel1.Xx) + Sqr(UzayPixel1.Yy) );
        alfa^ := ArcTan ( UzayPixel1.Yy / UzayPixel1.Xx)*180/pi;

        adres1 := adres1 + size1;
        adres3 := adres3 + size3;
        adres4 := adres4 + size4;

     until ( adres1 >= m1*size1 );

     aci1       := 10;
     aciartim   := 10;
```

```
for k := 0 to Trunc(360/aciartim) do
  begin
    P1          := HeapAdresi1;
    R1          := HeapAdresi3;
    alfa        := HeapAdresi4;
    adres1      := 0;

    adres3 := 0;
    adres4 := 0;

    repeat
      P1                := Ptr ( Segment1, offset1+adres1 );
      R1                := Ptr ( Segment3, offset3+adres3 );
      alfa              := Ptr ( Segment4, offset4+adres4 );
      aci2              := alfa^ + aci1;
      UzayPixel1.Xx     := R1^*Cos(aci2*pi/180);
      UzayPixel1.Yy     := R1^*Sin(aci2*pi/180);
      UzayPixel1.Zz     := P1^.Zz;

      ViewPlanePtr ( UzayPixel1,P2,adres2,size2,Segment2,offset2 );

      adres1 := adres1 + size1;
      adres3 := adres3 + size3;
      adres4 := adres4 + size4;

    until ( adres1>=m1*size1 );
    PolCizim ( P2,HeapAdresi2,adres2,size2,segment2,offset2,m1 );

    P2    := HeapAdresi2;
    adres2 := 0;
    aci1  := aci1 + aciartim;
  end;

P1    := HeapAdresi1;
adres1 := 0;
for k:=1 to  Round(m1/Yogunluk_2) do
  begin
    P1          := Ptr (Segment1,offset1+adres1);
    UzayPixel1  := P1^;
    yaricap     := Sqrt( Sqr(UzayPixel1.Xx) + Sqr(UzayPixel1.Yy ) );
    alfa1       := ArcTan ( UzayPixel1.Yy / UzayPixel1.Xx)*180/pi;
    ViewPlaneSpace ( UzayPixel1 );
```

```
        ViewPlaneScreen ( UzayPixel1,EkranNokta1 );
        Cember ( yaricap,alfa1 );

        adres1 := adres1 + Yogunluk_2*size1;
      end;

    P1              := Ptr (Segment1, offset1+m1*size1-40);
    UzayPixel1      := P1^;
    yaricap         := Sqrt( Sqr(UzayPixel1.Xx) + Sqr(UzayPixel1.Yy ) );
    alfa1           := ArcTan ( UzayPixel1.Yy / UzayPixel1.Xx)*180/pi;

    ViewPlaneSpace ( UzayPixel1 );
    ViewPlaneScreen ( UzayPixel1,EkranNokta1 );
    Cember ( yaricap,alfa1 );

    adres1 := 0;
    adres3 := 0;
    adres4 := 0;

  end;

  {

  ********************************************************************************************
  }

Procedure Yyzey3;
Var
  k,l : integer;
  begin
    otestep := 0;
    otestep := otestep +  Yogunluk_1/olcu;

    aktarma;
    for k := 1 to Ote_Say do
      begin
        Poligon (Dz_Uzay);
        for l := 1 to NoktaSayisi do
          Dz_Uzay[1].Yy := Dz_Uzay[1].Yy+otestep;
      end;

    aktarma;
```

```
    for k := 1 to NoktaSayisi do
      begin
        UzayPixel1.Xx := Dz_Uzay[k].Xx;
        UzayPixel1.Yy := Dz_Uzay[k].Yy;

        UzayPixel1.Zz := Dz_Uzay[k].Zz;
        UzayPixel1.t  := Dz_Uzay[k].t;
        ViewPlaneSpace ( UzayPixel1 );
        ViewPlaneScreen ( UzayPixel1,EkranNokta1 );

        UzayPixel2.Xx := Dz_Uzay[k].Xx;
        UzayPixel2.Yy := Dz_Uzay[k].Yy + (Ote_Say-1)*otestep;
        UzayPixel2.Zz := Dz_Uzay[k].Zz;
        UzayPixel2.t    := Dz_Uzay[k].t;
        ViewPlaneSpace ( UzayPixel2 );
        ViewPlaneScreen ( UzayPixel2,EkranNokta2 );

        Line ( XMerkez+Round(olcu*EkranNokta1.Xs),
         YMerkez+Round((olcu/4)*EkranNokta1.Ys),

         XMerkez+Round(olcu*EkranNokta2.Xs),
         YMerkez+Round((olcu/4)*EkranNokta2.Ys));

      end;
  end;

  {
  ********************************************************************************
  }

Procedure Yyzey4;
Var
  k,l,aciStep   : integer;
  yaricap,alfa1 : extended;
  begin
    aciStep := 0;
    aktarma;
    Poligon (Dz_Uzay);

    for k := 1 to Round(360/Yogunluk_1) do
```

```
      begin
       for I := 1 to NoktaSayisi do
        begin
         yaricap := Sqrt ( Sqr(Dz_Uzay[1].Xx) + Sqr(Dz_Uzay[1].Yy) );
         alfa1   := ArcTan (Dz_Uzay[1].Yy / Dz_Uzay[1].Xx )*180/pi;

         Dz_Uzay[1].Xx := yaricap*Cos((alfa1+acistep)*pi/180);
         Dz_Uzay[1].Yy := yaricap*Sin((alfa1+acistep)*pi/180);
        end;
       Poligon (Dz_Uzay);
       aktarma;
       acistep := acistep + Yogunluk_1;
      end;
    aktarma;
    for k := 1 to NoktaSayisi do
     begin
      UzayPixel1 := Dz_Uzay[k];
      yaricap    := Sqrt( Sqr(UzayPixel1.Xx) + Sqr(UzayPixel1.Yy) );
      alfa1      := ArcTan ( UzayPixel1.Yy / UzayPixel1.Xx)*180/pi;
      ViewPlaneSpace ( UzayPixel1 );
      ViewPlaneScreen ( UzayPixel1,EkranNokta1 );

      for I := Round(alfa1)+1 to 360 do
       begin
        UzayPixel2.Xx := yaricap*Cos(I*pi/180);
        UzayPixel2.Yy := yaricap*Sin(I*pi/180);
        UzayPixel2.Zz := Dz_Uzay[k].Zz;

        ViewPlaneSpace ( UzayPixel2 );
        ViewPlaneScreen ( UzayPixel2,EkranNokta2 );

        Line ( XMerkez+Round(olcu*EkranNokta1.Xs),
             YMerkez+Round((olcu/4)*EkranNokta1.Ys),

          XMerkez+Round(olcu*EkranNokta2.Xs),
          YMerkez+Round((olcu/4)*EkranNokta2.Ys));

          Move ( UzayPixel2,UzayPixel1,Size of (UzayPixel2) );
          Move ( EkranNokta2,EkranNokta1,Size of (EkranNokta2) );
       end;
     end;
end;
```

```
  {
  ******************************************************************************
  }

Procedure Giris1;
Var
  sa,su : byte;
  i     : byte;
  begin
  sa := 2;
  su := 1;
  Clrscr;
  Write ('Nokta Sayisini Giriniz.....:');
  Read( NoktaSayisi );
  for i := 1 to Noktasayisi do
     begin
        GotoXY ( su,sa+i ); Write  ( 'x[',i,']=' ); Read ( Cx[i] );
        GotoXY ( su+30,sa+i ); Write  ( 'z[',i,']=' ); Read ( Cz[i] );
     end;
  WriteLn;
  Write ('x[1] noktasindaki birinci turev.....');
      ReadLn (Y_ilkTurevy1);
  Write ('x[0] noktasindaki ikinci  turev.....');
      ReadLn (Y_y0Turev);
  end;

  {
  ******************************************************************************
  }

Procedure Giris2;
   begin
     Clrscr;
     WriteLn ('Olcu = 20 pixel ( 1 br. 20 pixel alinmistir )');
     Write ('Yogunluk1.....','('',Yogunluk_1,')');ReadLn( Yogunluk_1 );
     Write ('Yogunluk2.....','('',Yogunluk_2,')');ReadLn( Yogunluk_2 );
     WriteLn ('GozNoktasi (',GozNoktasi.Xx:8:4,GozNoktasi.Yy:8:4,
                                GozNoktasi.Zz:8:4,') :');

     ReadLn ( GozNoktasi.Xx,GozNoktasi.Yy,GozNoktasi.Zz );
     Write ('Yyzey kodu ( 0 = oteleme yuzeyi, 1 = dondurme yuzeyi ) ');
     ReadLn ( YyzeyKod );
```

```
    end;

    {
    ******************************************************************************************
    }

Procedure Kontrol1;
    begin
     Case Yyzeykod of
         0 : Yyzey1;
         1 : Yyzey2;
     end;
    end;

    {
    ******************************************************************************************
    }

Procedure Kontrol2;
    begin
     Case Yyzeykod of
         0 : Yyzey3;
         1 : Yyzey4;
     end;
    end;

    {
    ******************************************************************************************
    }

     begin
       Ote_Say := 15;
       m1 := 0;
       for i :=1 to 25 do
          move(ikinciTurev,TurevMatris[i,1],Size of (ikinciTurev));
       for i := 0 to 25 do
          begin
            h[i] := 0;
            A[i] := 0;
            B[i] := 0;
          end;
       Giris1;
```

```
repeat
  for k := 0 to NoktaSayisi do

    Cy[k] := 0;
    CubicSpline (Cx,Cz,Y_ilkTurevy1,Y_y0Turev,NoktaSayisi);
    GotoXY (40,24); Write('Determinant .....:',Determ:12:6);
    ReadLn;

    BellekAyirma;

    repeat
      ClrScr;
      Giris2;
GozNoktasi.Xx := olcu*GozNoktasi.Xx;
GozNoktasi.Yy := olcu*GozNoktasi.Yy;
GozNoktasi.Zz := olcu*GozNoktasi.Zz;

r := Sqrt( Sqr(GozNoktasi.Xx) + Sqr(GozNoktasi.Yy) +
                    Sqr(GozNoktasi.Zz) );
teta := ArcTan ( GozNoktasi.Yy / GozNoktasi.Xx )*180/pi;

fi := Arctan ( Sqrt( Sqr(GozNoktasi.Xx) + Sqr(GozNoktasi.Yy) )
                    / GozNoktasi.Zz)*180/pi;

      TViewMatrisi;

      Gd := Detect;
      InitGraph ( Gd, Gm, 'c:\bp\bgi' );
      if Graphresult then Halt(1);
      EksenTakimi;

      if (determ > 10e-12 ) then
        Kontrol1
        else
          Kontrol2;
      ch := Readkey;
      ClearDevice;
      TextMode ( LastMode );

    until ( Ch = #27 ) or ( Ch = #32 );

if ch = #32 then
```

```
        begin
          P1 := HeapAdresi1;
              FreeMem( P1,m1*size1 );
          P2 := HeapAdresi2;
              FreeMem( P2,m1*size2 );
          R1 := HeapAdresi3;
              FreeMem( R1,m1*size3 );
          alfa := HeapAdresi4;
              FreeMem( alfa,m1*size4 );
          Giris1;
        end;

    until ( ch = #27 );

  CloseGraph;
end.
{
*********************************************************************************
}
```

```
Unit TEZ_ALT_PROG;

interface

{$n+}
uses Crt;

type
   matris  = Array [1..25,1..25] of extended;
   dizi    = Array [1..25] of extended;
Var
   Determ : extended;
{
  ********************************************************************************
}
Procedure gauss ( var denk_matris    : matris;
             var cozum          : dizi;
                m,n             : integer;
                made            : Boolean);
{
  ********************************************************************************
}
Procedure determinant ( det_matris    : matris;
                 var detern    : extended;
                        n        : integer);

{
  ********************************************************************************
}
implementation
{
  ********************************************************************************
}
Procedure matris_input ( var input_mat     : matris;
                                    m,n          : integer);
var
  a,b   : integer;
  begin
     ClrScr;
     WriteLn('Katsayilar matrisini giriniz...');
     for a := 1 to m do
        for b := 1 to n-1 do
           begin
             Write('[',a,' ',b,']......');
```

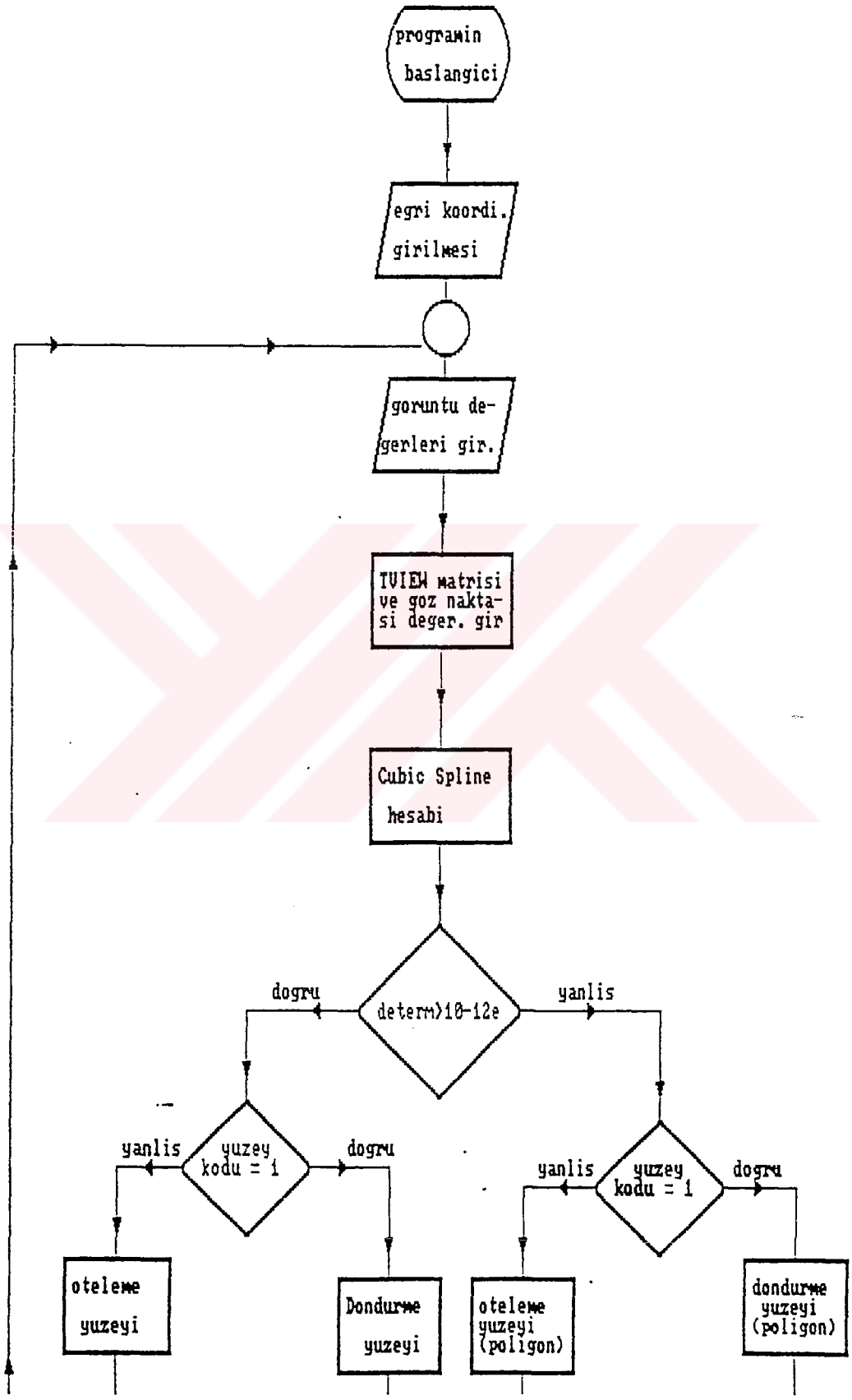
```
            Read(input_mat[a,b]);
          end;
      WriteLn;
      WriteLn('Denklemlerin sag taraflarini giriniz');
      for a := 1 to m do
        begin
          Write('[',a,' ',n,'].....');
          Read(input_mat[a,n]);


        end;
      ClrScr;
  end;

{
  ********************************************************************************
}
Procedure satir_degistirme ( var degis_matris     : matris;
                    var degis_dizi                : dizi;
                      s,p,r                       : integer;
                      make                        : Boolean);
var
  c                 : integer;
  ara_degisken      : extended;

  begin
    if make then
      begin
       for c := 1 to s do
          begin
            ara_degisken      := degis_matris[p,c];
            degis_matris[p,c] := degis_matris[r,c];
            degis_matris[r,c]  := ara_degisken;
          end;
      end
      else
        begin
          ara_degisken  := degis_dizi[p];
          degis_dizi[p] := degis_dizi[r];
          degis_dizi[r] := ara_degisken;
        end;
  end;
```

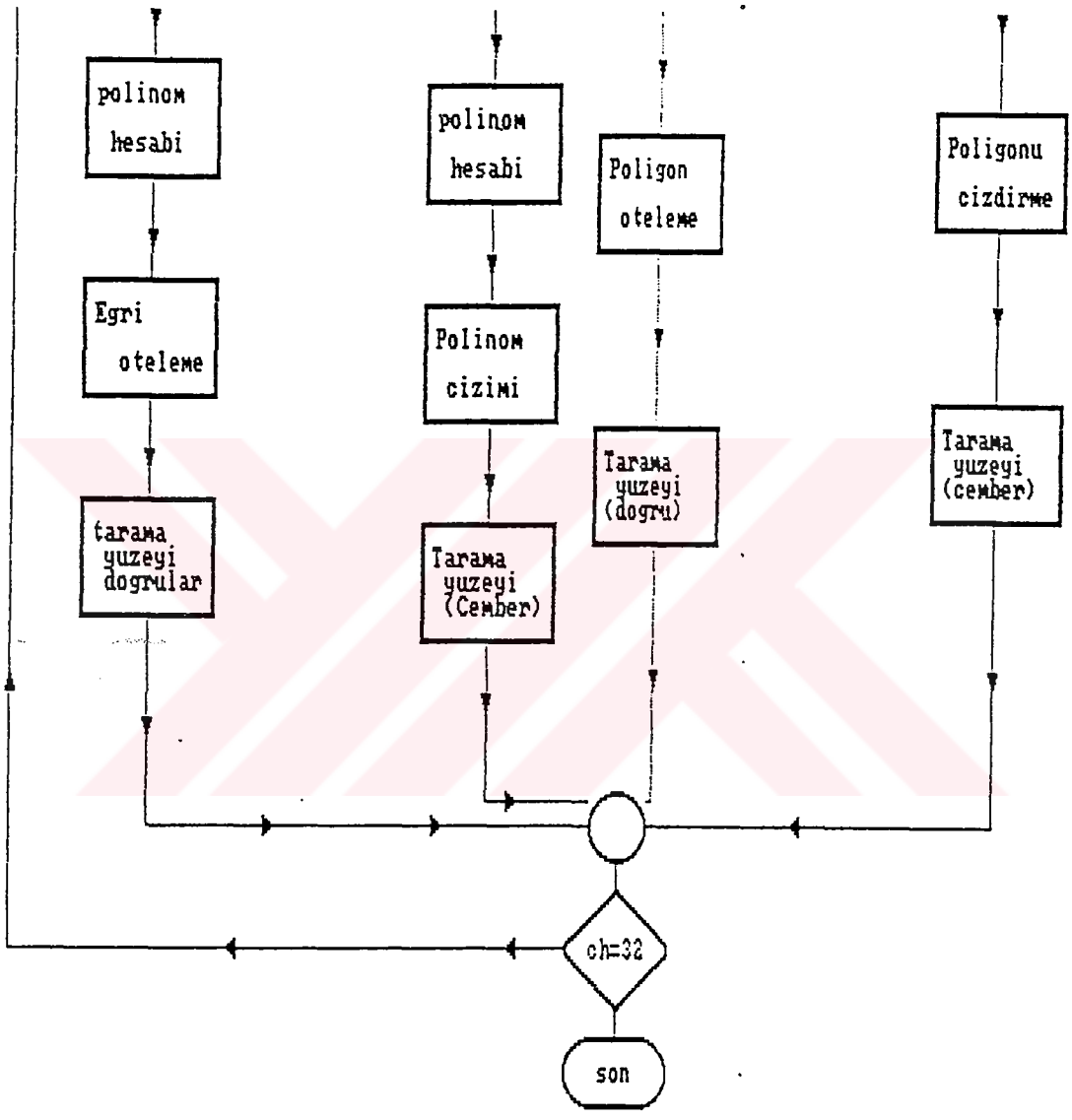
```
Procedure kosegenlestirme ( var s_matris    : matris;
                    var sag_taraf           : dizi;
                      m,n                   : integer);
var
  i,j,k,l                  : integer;
  katsayi,sifir_degeri     : extended;
  done                     : Boolean;

  begin
    for k := 1 to m-1 do
       begin
         l := k;
         if s_matris[k,k] = 0 then
           begin
            repeat
              sifir_degeri := s_matris[1+k,k];
              l := l+1;
            until (sifir_degeri or (l=m);
            if sifir_degerithen
              begin
                satir_degistirme(s_matris,sag_taraf,m,k,l,true);
                satir_degistirme(s_matris,sag_taraf,n,k,l,false);
              end;
            if l = m then exit;
           end;
         for i := k+1 to m do
           begin
              katsayi := -(s_matris[i,k]/s_matris[k,k]);
              for j := 1 to n do
                s_matris[i,j] := katsayi*s_matris[k,j]+s_matris[i,j] ;
              for j := 1 to n do
                sag_taraf[j] := s_matris[j,n];
           end;
       end;
  end;
  {
   ********************************************************************************
  }
Procedure determinant(    det_matris    : matris;
                    var determ          : extended;
                      n                 : integer);
```

```
var
  q : integer;

  begin
     detern := 1;
     for q := 1 to n do
        detern := detern*det_matris[q,q];
  end;
  {
   ***************************************************************************************
  }
Procedure gauss ( var denk_matris    : matris;
            var cozum                : dizi;
               m,n                   : integer;
               made                  : Boolean);
var
  sag_mat          : dizi;
  i,j              : integer;
  deger            : extended;

  begin
     deger              := 0;
     for i              :=1 to m do
       sag_mat[i]       := denk_matris[i,n];
     kosegenlestirme(denk_matris,sag_mat,m,n);
     determinant(denk_matris,determ,m);
     if determ = 0 then
       begin
          WriteLn('Katsayilar Matrisinin Determinanti Sifirdir...');
          Halt;
       end;
     if denk_matris[m,n-1] then
                              cozum[m] := sag_mat[m]/denk_matris[m,n-1]
                              else
                             cozum[m] := 0;
     for i := m-1 downto 1 do
        begin
          for j := i+1 to n-1 do
             deger      := deger + denk_matris[i,j]*cozum[j];
             deger      := sag_mat[i] - deger;
             cozum[i ]  := deger/denk_matris[i,i];
             deger      := 0;
```

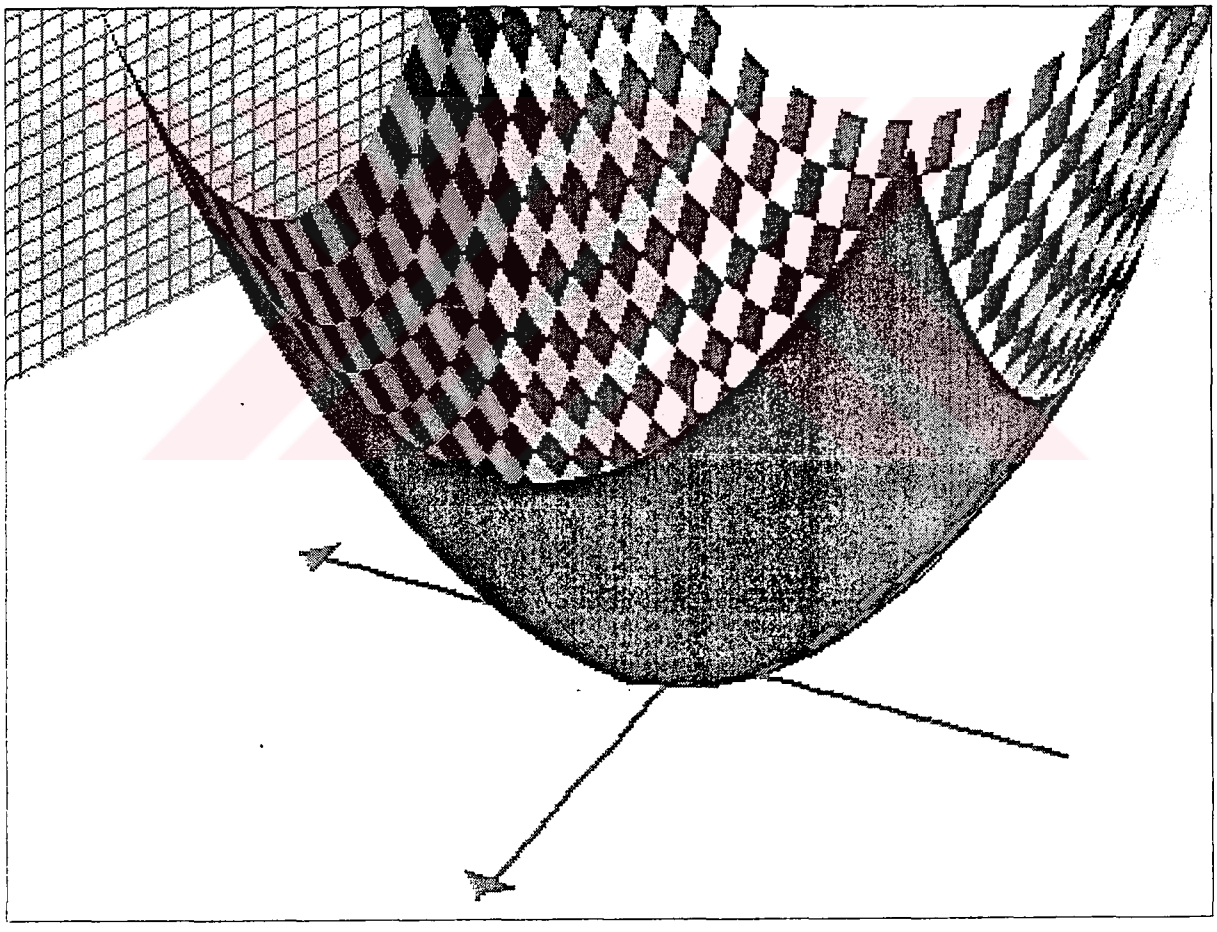
```
        end;
        if made then
                begin
                   WriteLn('Denklemin Cozumu');
                   for i := 1 to.m do
                      WriteLn('x',i,'.....',cozum[i]:16:8);
                end;

end;
{
 ********************************************************************************
}
end.
```

Programin Akis Diyagrami

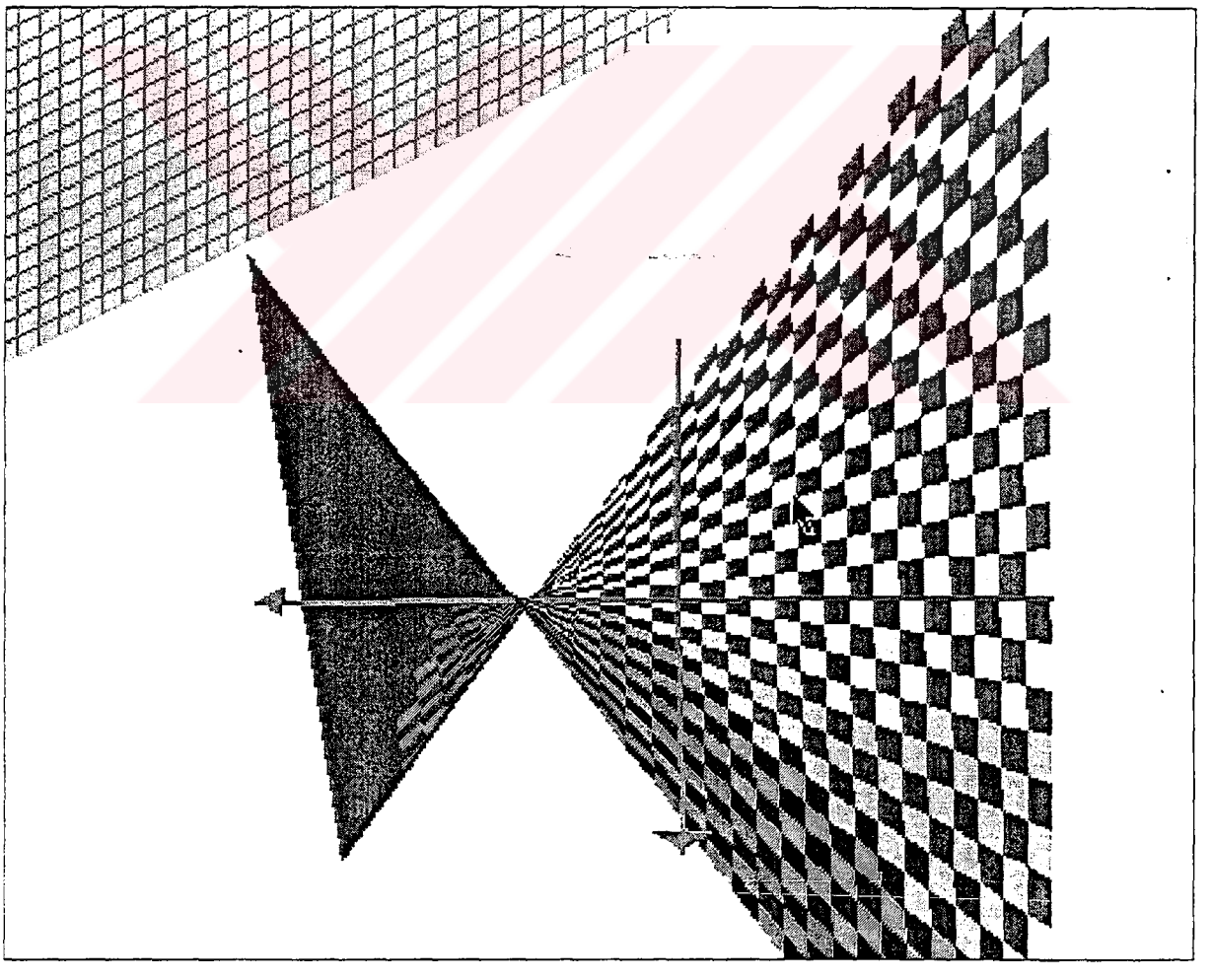



polinom hesabi
Egri oteleme
tarama yuzeyi dogrular
polinom hesabi
Polinom cizimi
Tarama yuzeyi (Cember)
Poligon oteleme
Tarama yuzeyi (dogru)
Poligonu cizdirme
Tarama yuzeyi (cember)
ch=32
son

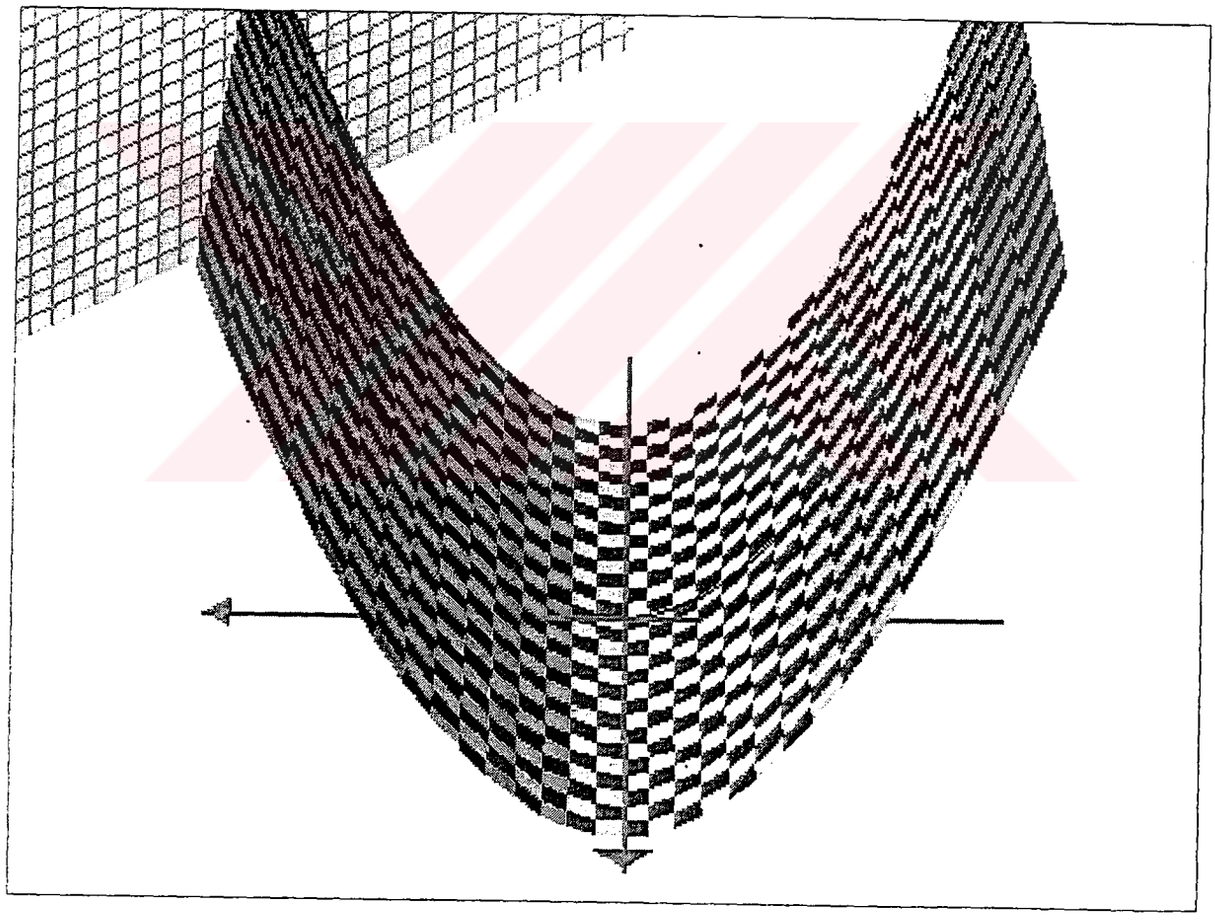
$$z = x^2 + y^2$$



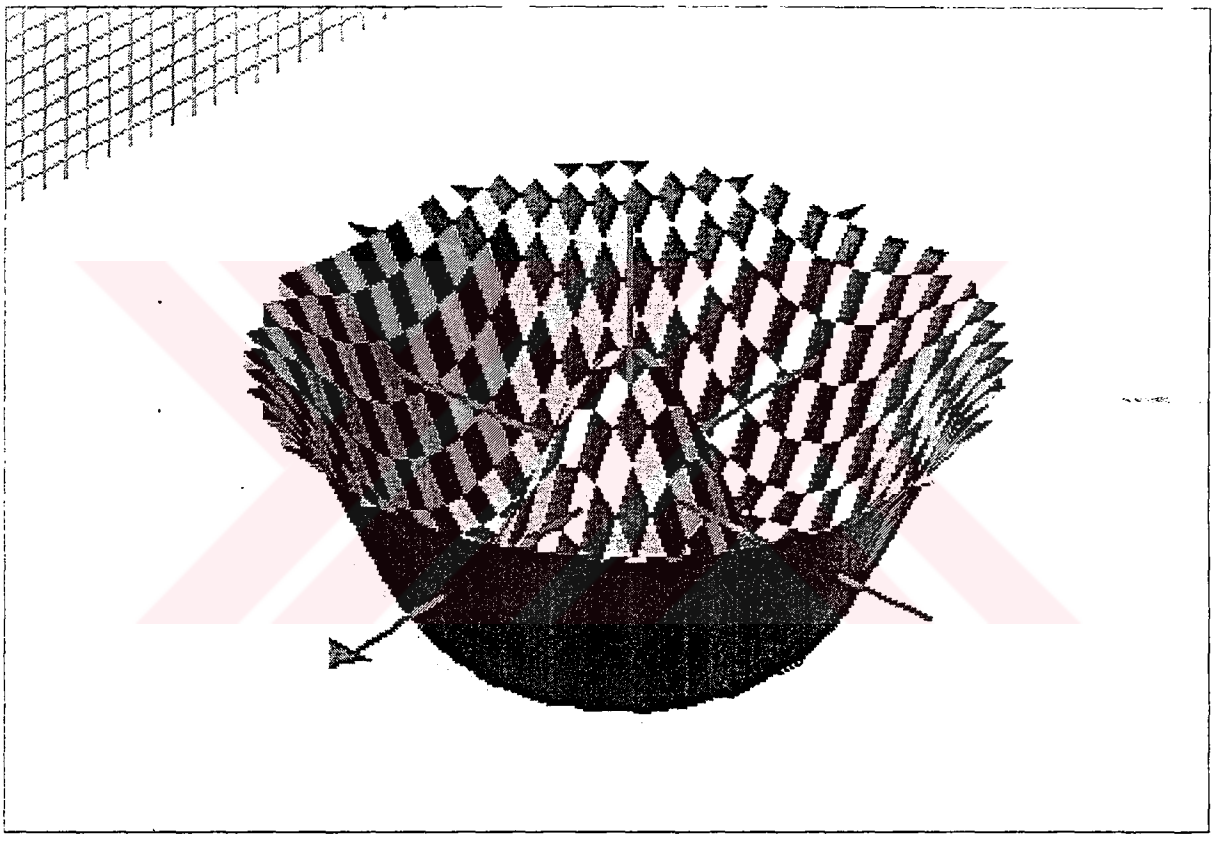
$$z = xy$$



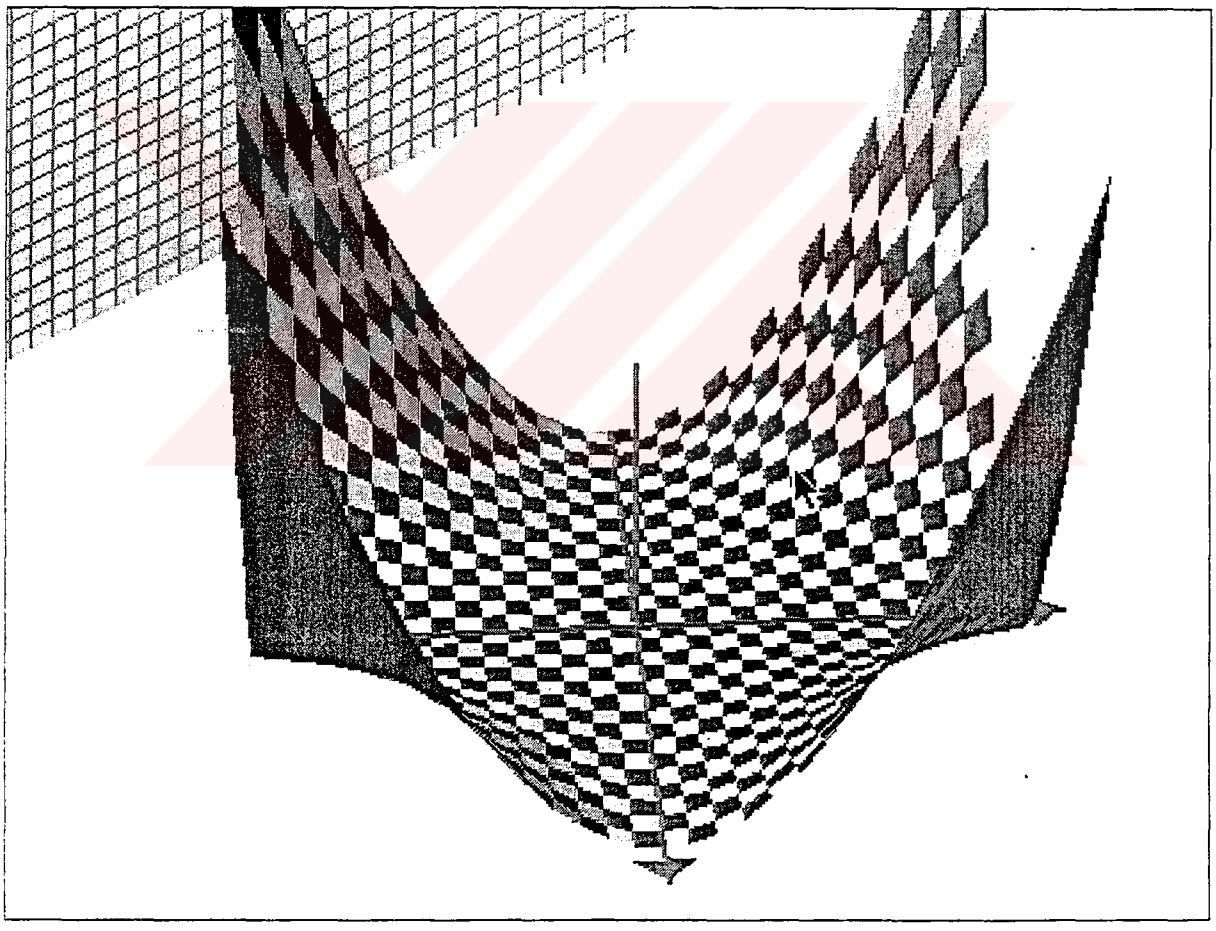
$$z = x^2$$



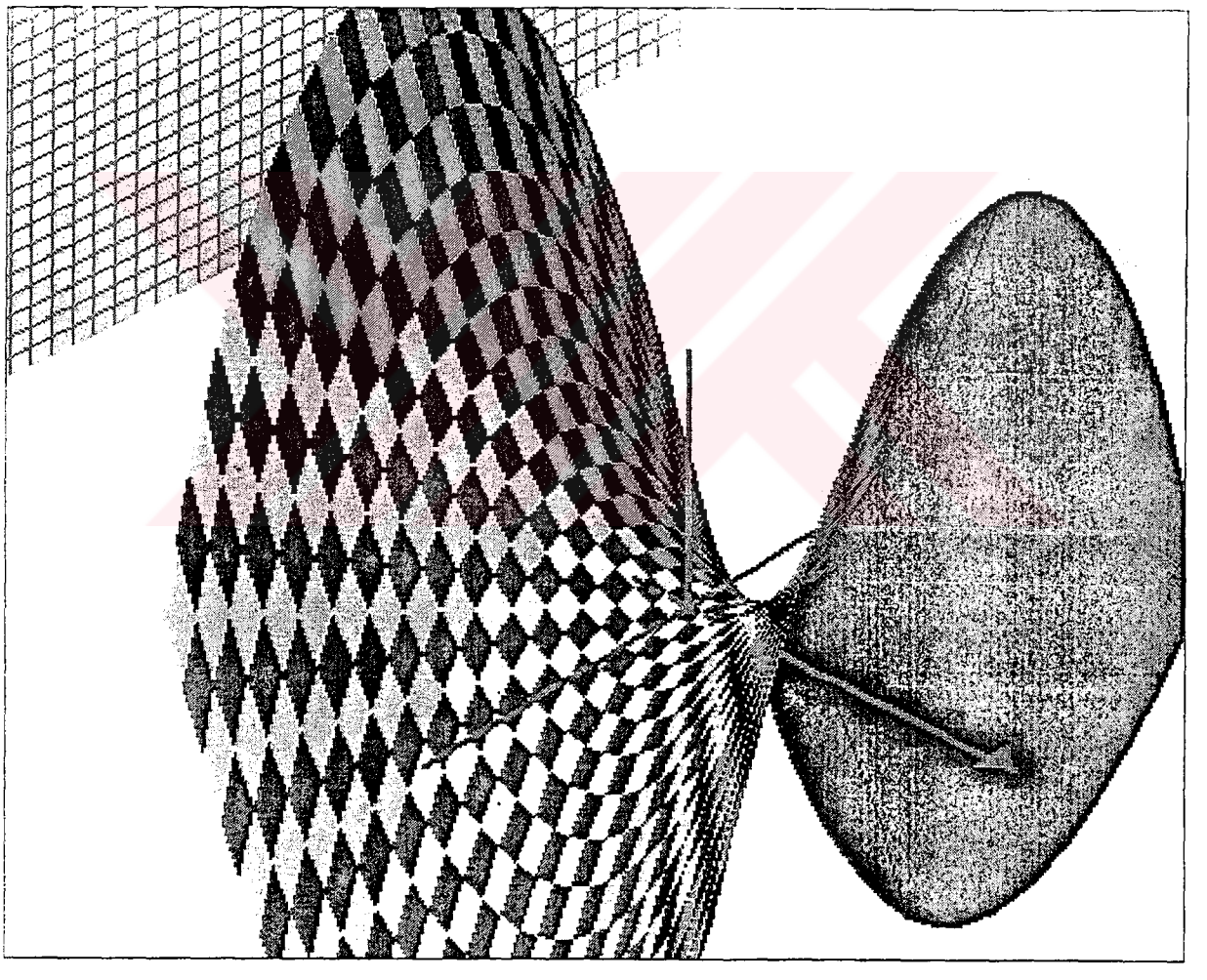
$$Z = \cos \pi r$$



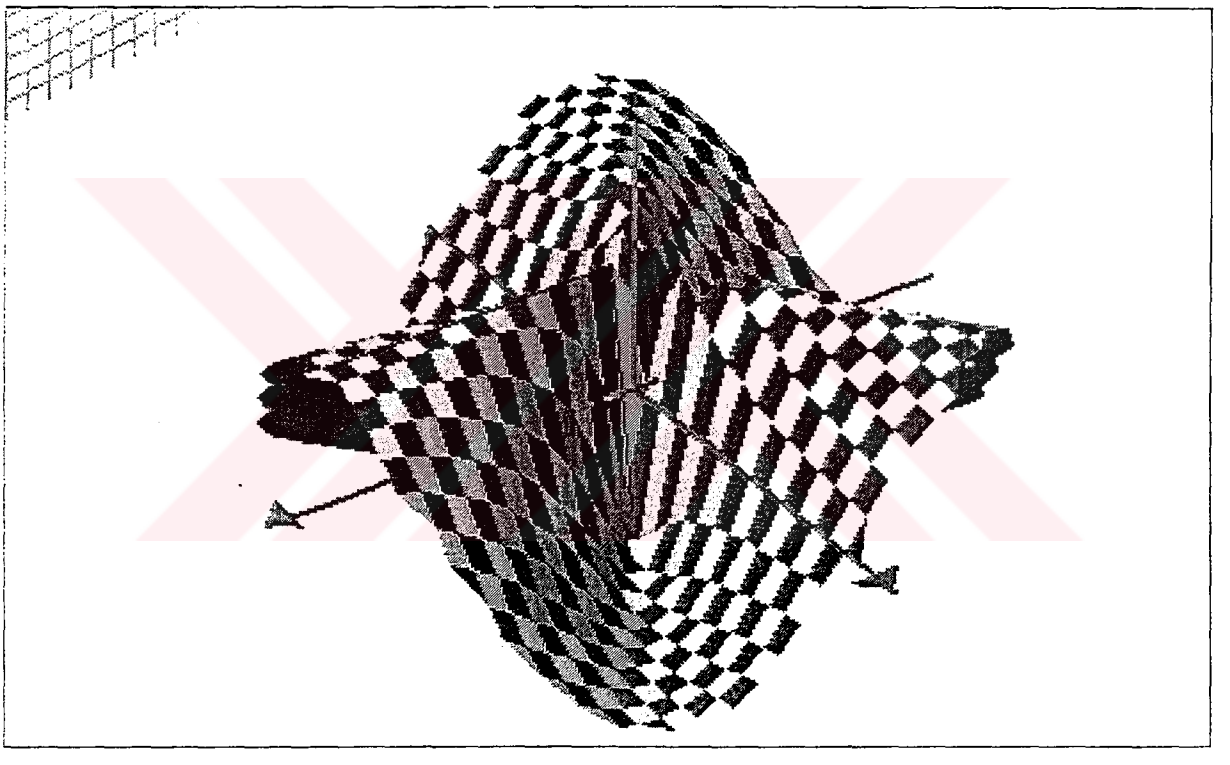
$$z = x^2y^2$$



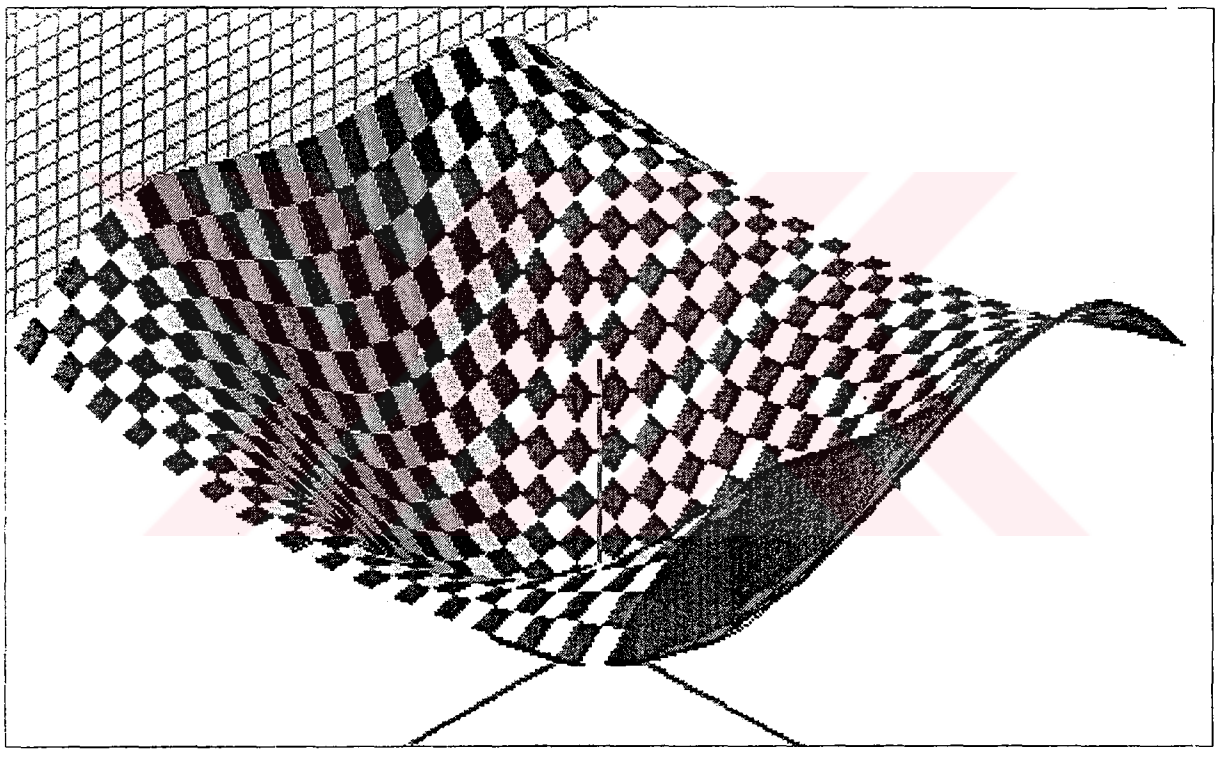
$$z = x^2 - y^2$$



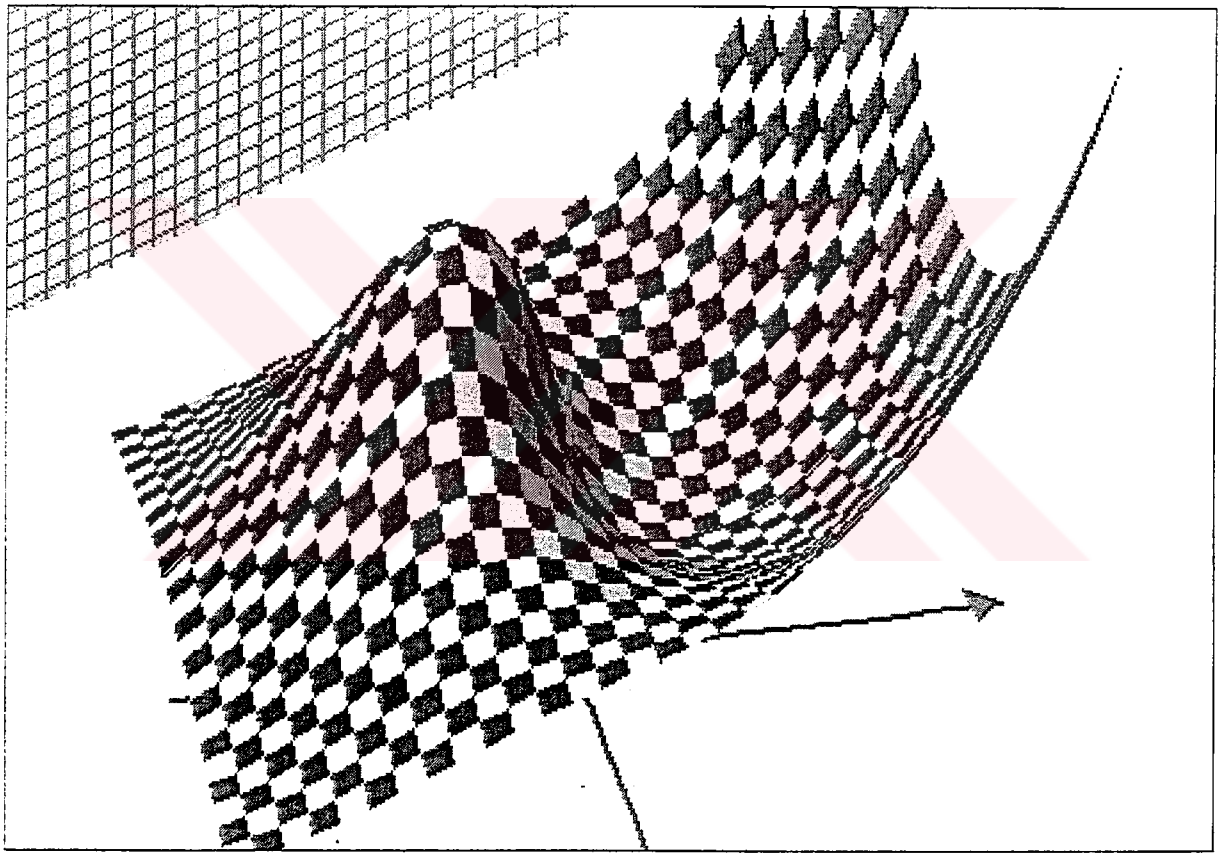
$$z = \cos 3\theta$$



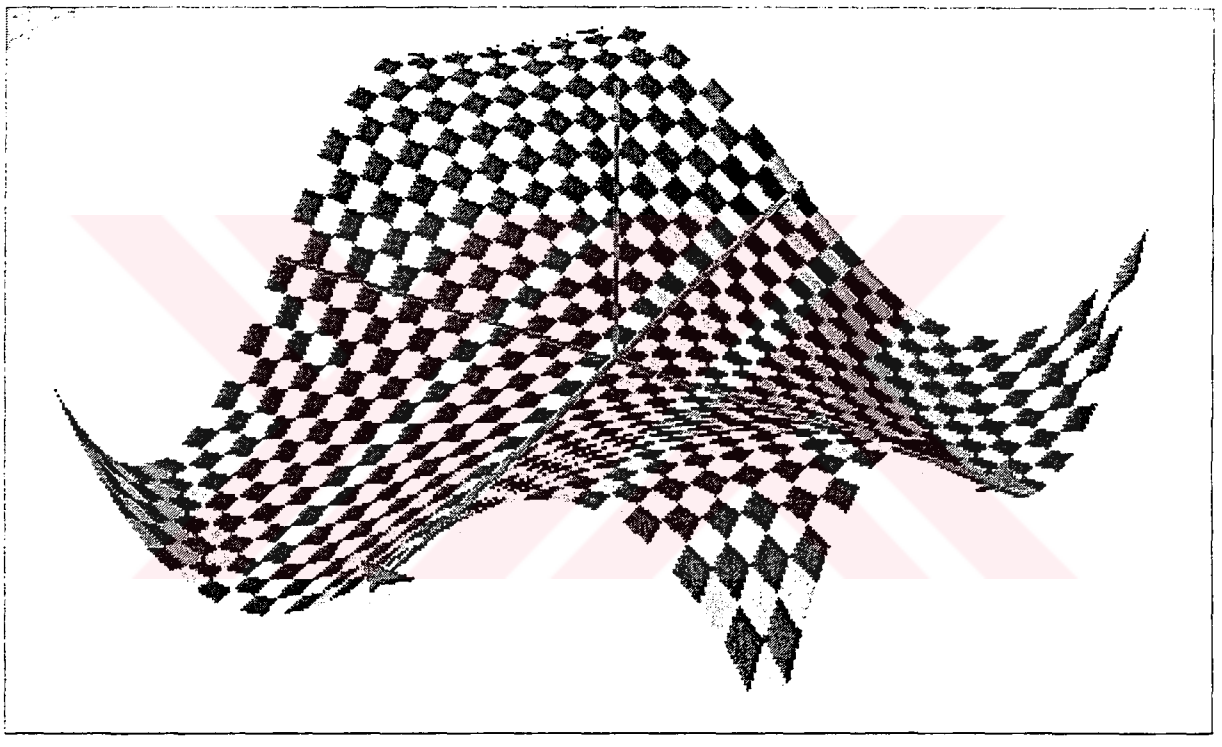
$$z = (x^2 + y^2)^{\cos x}$$



$$z = (x^2 + y^2)^{\sin x}$$



$$z = \sin((xy)\cos xy)$$



ÖZGEÇMİŞ

Adı soyadı	:	Müslüm ÖZIŞIK
Doğum tarihi	:	05.07.1969
Doğum yeri	:	Ayancık / SİNOP
1975 - 1980	:	Sakarya İlkokulu'ndan mezun.
1980 - 1983	:	Yenisahra Ortaokulu'ndan mezun.
1983 - 1988	:	Haydarpaşa Endüstri Meslek Lisesi'nden Haydarpaşa Teknik Lisesi Elektronik Bölümüne geçiş ve bu bölümden mezuniyet.
1988 - 1992	:	Yıldız Üniversitesi Mühendislik Fakültesi Matematik Mühendisliği Bölümü'ne giriş ve bu bölümden mezuniyet.
1992 - 1994	:	İ.T.Ü F.B.E Sistem Analizi'ne giriş ve Y.T.Ü Kimya - MetalurjiFakültesi Araştırma Görevliliği kadrosuna atamamın yapılabilmesi için bu bölümden zorunlu olarak ayrılma.
1994 - 1995	:	Y.T.Ü F.B.E Matematik Mühendisliği Bölümü'nde yeniden Yüksek Lisans'a başlangıç ve bir yıl hazırlık sınıfı eğitimi.
1995 - 20.1.1997	:	Y.T.Ü F.B.E Matematik Mühendisliği Bölümü'nde derse başlama ve bu bölümden mezuniyet.