1)

Bilgisayarlı Grafik : Oluşturulan veya toplanan verilerin bilgisayar teknolojileri vasıtasıyla görüntüle şekilde sunulmasını sağlayan bilim dalıdır.

Grafik Pipeline işlemi : 3 aşamalıdır.

1) Uygulama aşaması

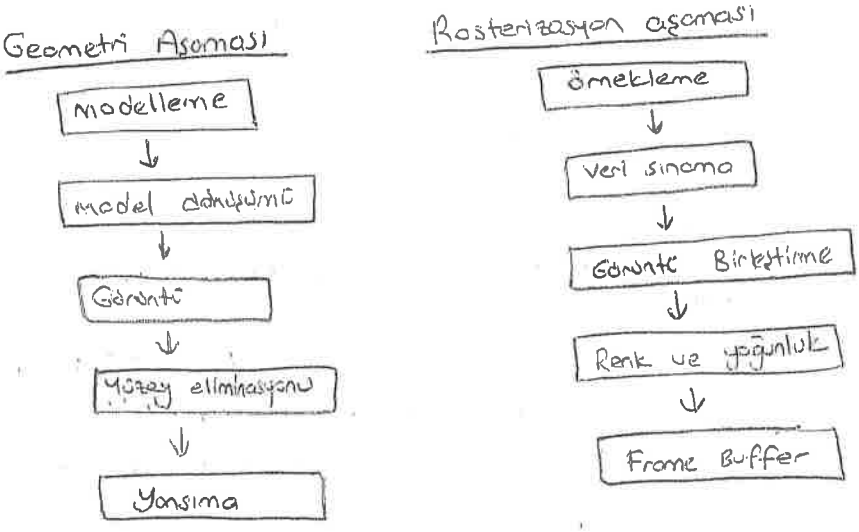
* Bu aşamada matematiksel veya veriye bağlı işlemler CPU üzerinde gerçekleştirilir.

2) Geometri Aşaması

* Ne tür modelleme yapılacağının ortaya konulduğu aşamadır.

3) Rasterizasyon Aşaması

* Ekran kartındaki renk değerlerinin ayarlanmasıdır.



3D model için oluşturulacak Aşamalar

1) Matematiksel model
2) Birleştirme
3) Veri Ekleme
4) Işık işlemleri
5) Pozisyon

(1.1)

Cudo = GPU için NVIDIA'nın sunduğu C programlama dili üzerinde eklenti olarak kullanma imkanı sağlayan bir mimari ve teknolojidir.

Voxel = Bir pikselin 3 boyutlu karşılığıdır.

CPU = Bilgisayardaki işlemleri yürüten ve sonuçları gerekli yerlere gönderen elemandır.

GPU = Grafik işleme ünitesidir. Grafik yaratımı için kullanılan aygıttır.

Stream = Verinin kaynağına denir.

* Eğriler *

* 3 şekilde gösterilir.

1) Kapalı form :

$f(x,y)=0$, $f(x,y,z)=0$

$y-x=0$ $\quad x^2+2x-2=0$

2) Açık form :

$y=f(x)$, $z=f(x,y)$

$y=x$ $\quad y=x^2+2x+1$

3) Parametrik Form

$x=r\sin\theta$ $\quad x=x(\theta)$

$y=r\cos\theta$ $\quad y=y(\theta)$

$x^2+y^2=r^2$

* Eğri modelleme yöntemleri *

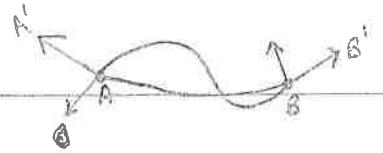
1- Hermit Eğrisi
2- Beaier Eğrisi
3- Cardinal Spline Eğrisi
4- Kochanek-Bertel Spline Eğrisi
5- B-Spline Eğrisi
6- Rasyonel Yapılar

* Eğri özellikler *

1) Matematiksel ifadesi olanlar (Dizilem takımı çöz)
2) Matematiksel ifadesi olmayanlar (Ayrık Datalar kullan)
   - 3D Scanner
   - 3D Digital
   - Ölçerek
3) İki yüzey kesişimi eğridir.
4) Başlangıç ve bitiş noktalarında bazı parametreler kullanılır.
5) Bir nokta ve o noktadaki eğrisellik değeri verilerek eğrilik değeri hesaplanabilir.
6) Eğrinin denklemi yerine zarfı verilerek eğri modellenebilir.
7) Parça parça modellenebilir.

(1)

1) Hermit Eğrisi



* 3. dereceden bir eğridir.
* Kübik eğrileridir.
* Boyuttan bağımsızdır.
* Hem 2D hem 3D modellenebilir.

$$p(u) = au^3 + bu^2 + cu + d \qquad 0 \le u \le 1$$

$u=0$ için $\boxed{p(0) = d}$

$u=1$ için $\boxed{p(1) = a+b+c+d}$

$$p'(u) = 3au^2 + 2bu + c$$

$u=0$ için $\boxed{p'(0) = c}$

$u=1$ için $\boxed{p'(1) = 3a+2b+c}$

* a, b, c, d → bilinmeyen

$$\begin{cases} p(0) = d \\ p(1) = a+b+c+d \\ p'(0) = c \\ p'(1) = 3a+2b+c \end{cases}$$

$$\boxed{p(1) = a+b+p'(0)+p(0)}$$

$$\boxed{p'(1) = 3a+2b+p'(0)}$$

$$a+b = p(1) - p'(0) - p(0)$$

$$3a+2b = p'(1) - p'(0)$$

Genel formu :

$$\boxed{p(u) = (2u^3 - 3u^2 + 1).(p(0)) + (-2u^3+3u^2).(p(1)) + (u^3 - 2u^2 + u).p'(0) + (u^3 - u^2).p'(1)}$$

2) Bezier Eğrisi

* Yaklaşım eğrisidir.
* Başlangıç ve bitiş noktasından eğri geçer.
* Türevlenebilen eğridir.
* Bölgesel kontrol yoktur.
* Hermit eğrisine göre daha yumuşaktır.

$$P(u) = \sum_{i=0}^{n} P_i \, B_{i,n}(u) \qquad 0 \le u \le 1$$

$$B_{i,n}(u) = C\binom{n}{i} \cdot u^i \cdot (1-u)^{n-i}$$

$$C\binom{n}{i} = \frac{n!}{(n-i)! \, i!}$$

* Kontrol noktası 3 ise $n=2$ olur.

$$P(u) = P_0 * (1-u)^2 + P_1 * 2 * u(1-u) + P_2 * u^2$$

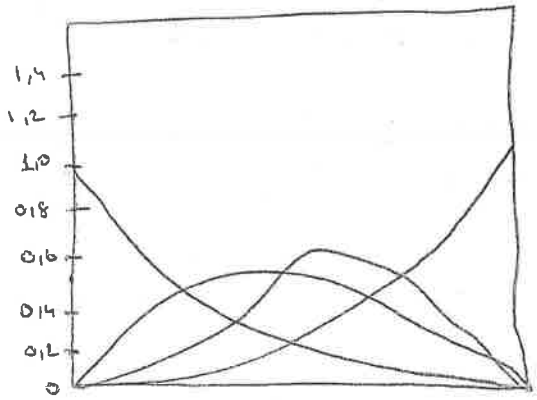
* 4 kontrol noktası var ise; $n=3$

$$P(u) = P_0 * (1-u)^3 + P_1 * 3 * u * (1-u)^2 + P_2 * 3 * u^2 * (1-u) + P_3 * u^3$$

Genel formu:

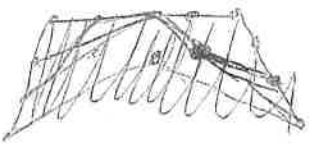
$$P(u) = P_0 * (1-u)^n + P_1 * C(n,1) * u(1-u)^{n-1} + P_2 * C(n,2) * u^2 (1-u)^{n-2} + \dots$$

Singleton = Bezier fonksiyonu hangi noktadan kesilirse kesilsin fonksiyon değerleri toplamı 1'dir.

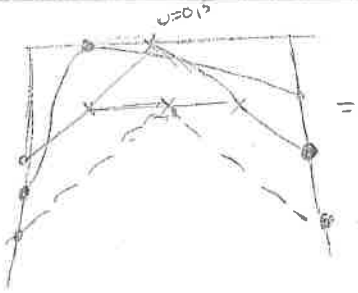


③ Casteljau Algoritması

* Eğriyi alt parçalara bölerek çizim tekniğini kullanmayı amaçlar.



(Resime tekrar Bak :)



$(1-u)A + uB$

$= 0,5A + 0,5(B)$

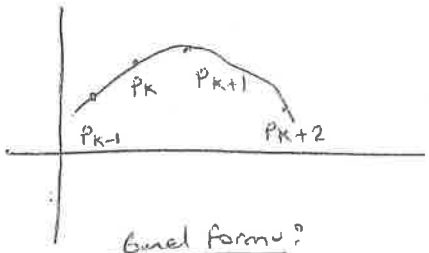
$= \frac{1}{2}A + \frac{1}{2}B$

u=0
u=0,25
u=0,5
u=0,75

* Bezier ile aynı eğriyi üretir.
* Bezier'deki faktoriyel yok. Matematiksel yükü azdır.
* Recursif çalışır. Hızlıdır.

3) Cardinal Spline eğriler

* Parça parça modellemeye sahip eğrilerdir.
* 4 kontrol noktasına göre çalışır.
* 3. dereceden kübik eğrilerdir.
* Eğri üzerindeki kontrol iki şekilde sağlanır
  - ↳ kontrol nokta pozisyonları değiştirilerek
  - ↳ Gerilme katsayısı değiştirilerek



Genel formu:

$$p(u) = P_{K-1}(-su^3 + 2su^2 - su) + P_K[(2-s)u^3 + (s-3)u^2 + 1]$$
$$+ P_{K+1}[(s-2)u^3 - (3-2s)u^2 + su] + P_{K+2}(su^3 - su^2)$$

$0 \le u \le 1$

$s = (1-t)/2$

t = Gerilme katsayısı

4) Kochonek-Bartel Spline Eğriler

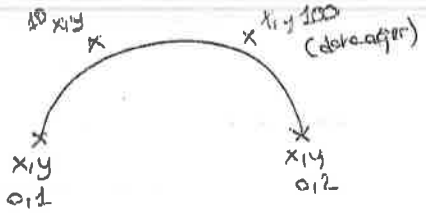
* Cardinal Spline eğrisine ilave 2 parametre ile elde edilen parametredir.
  - ↳ Süreklilik
  - ↳ Bias
* Arabulma ve yaklaşım yüzeyleri şeklinde modellenebilir.
  (interpolation)

5) B-Spline Eğrisi

* Parça parça modellemeye sahip eğrilerdir.
* Bölgesel kontrol vardır.
* Süper eğri olarak adlandırılır.
* uniform veya ~~non uniform~~ nonuniform (NURBS)
  ↳ B-spline
* Eşit aralıklı veya eşit aralıklı olmayan
* Esnek değil veya esnek
* Hem yaklaşım hem de interpolation eğrisi ile modellenebilir.

6) Rasyonel Yapılar

* Kontrol noktalarına göre çalışır.



* Eğri, yüzey ve katı modellemede kullanılan kontrol noktalarının darbeleme ne kadar etkili ettireceği değerdir. O no katsayı ağırlığıdır.

* Süreklilik şartı *

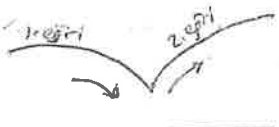
* Parça parça modellenmiş eğrilerin analizinde kullanılan ifadelerdir.

C^0 şartı = Birinci eğrinin bitiş noktası ikinci eğrinin başlangıç noktasına temas etmeli.

C^1 şartı = İki eğri birleşeceği zaman 1. türev yönlerinin aynı olması

C^2 şartı = Birleşim bölgesindeki 2. türev yönlerin aynı olması

* C^0 şartı



* C^1 şartı

Geometrik süreklilik şartı

$G^0 = C^0$ 

G^1 = Birleşim bölgesinde hesaplanacak 1. türevi değerlerinin belli bir k sabiti ile birbirine eşit olması.

$G^1 = k \cdot C^1$

G^2 = Birleşim bölgelerinde 2. türevlerin belli bir k sabitiyle eşit olması.

$G^2 = K \cdot C^2$

* YÜZEY MODELLEME *

* Literatürde 1'den fazla yüzey modelleme yöntemi mevcuttur.

Genel olarak 2 türlüdür.

- ↳ matematiksel modelli yüzeyler
- ↳ matematiksel modeli olmayan yüzeyler.

free form : Belli bir sınır eğrisine bağlı kalmadan modelleme işlemi gerçekleştirmek.

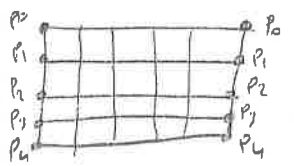
twist : DNA sarmalı gibi bir eksenli döndürme işlemidir.

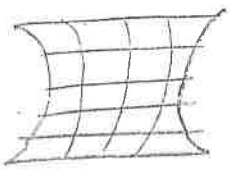
* Yüzey modelleme yöntemleri *

1) Kurallı yüzey modelleme
2) Döndürmeli yüzey modelleme
3) Bilinear yüzey modelleme

1) Kurallı yüzey modelleme

* Yerleştirirken bir kural veya düzene göre yerleştirilmiyorsa.

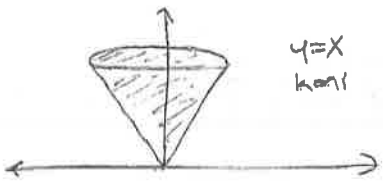




$P(t) = (1-t) \cdot p_0 + t \cdot p_1$ Eğriden bağımsız $0 \leq t \leq 1$

* Eğer parçalar arasındaki uzaklık eşit ise homojendir.
* iki eğride açık formda olmalı (2D için)
* Herikli eğrinin de kontrol noktası sayısı aynı olmalı

2) Döndürmeli Yüzey modellemesi

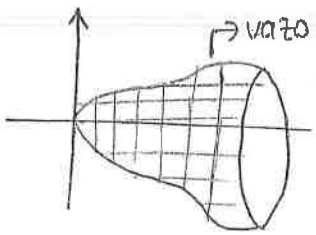


1) Dairesel yapılar → üst parçası çember
2) Elipsoidal yapılar → üst parçası elips

$p(t) = [\ x(t) \quad y(t)\ ]$ //parametrik

$\theta(t,\theta) = [\ x(t) \quad y(t).r_1.\cos\theta \quad y(t).r_2.\sin\theta\ ]$

X ekseline göre :



$r_1 = r_2$ → çemberde

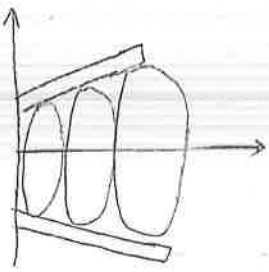
$r_1 = r_2$ = Dairesel

$r_1 \neq r_2$ = Elipsoidal

$0 \leq \theta \leq 360$ (Tamamı)

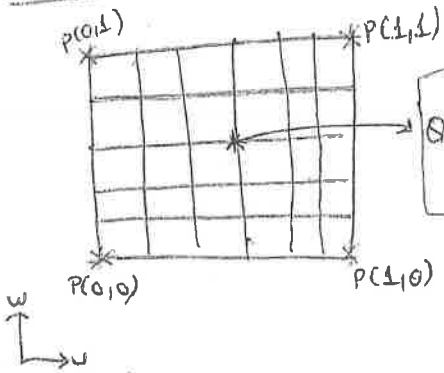
$0 \leq \theta \leq 180$ (kesit) (Yarısı)

Hacim için :

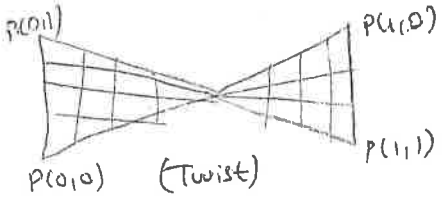


Buradaki her parçanın adı ; gold, ızgara, mesh. ... denir.

3) Bi Linear yüzey



$$Q(u,w) = p(0,0)(1-w)(1-u) + p(0,1)(1-u).w + p(1,0).u(1-w) + p(1,1).u.w$$




```
for(u=0.0  u<=1.0  u+=0.1){
for(w=0.0  w<=1.0  w+=0.2){
```

* Parça sayısı büyüdükçe hassasiyet artar.
* nokta sayısı azaldıkça işlem hızı artar.

(5) Çizgi ve Çember Çizme algoritması

Amaç:

* ~~Ara değerler ve ondalıklı değerleri~~ tek bir nokta olarak saymak.
* Gereksiz hesaplamalardan kurtulup minimum nokta ile çalışmak.

Çizgi çizme algoritması (The Bresenham Line Algorithm)

1) x_0 ve y_0 başlangıç noktası olsun.

2) (x_0, y_0) noktasını çiz.

3) P_0 karar parametresini hesapla.

$$P_0 = 2\Delta y - \Delta x \qquad \Delta y = y_1 - y_0 \qquad \Delta x = x_1 - x_0$$

4) Karar parametresi P_k;

$P_k < 0$

(x_{k+1}, y_k) → aydınlat

$P_{k+1} = P_k + 2\Delta y$

$P_k > 0$

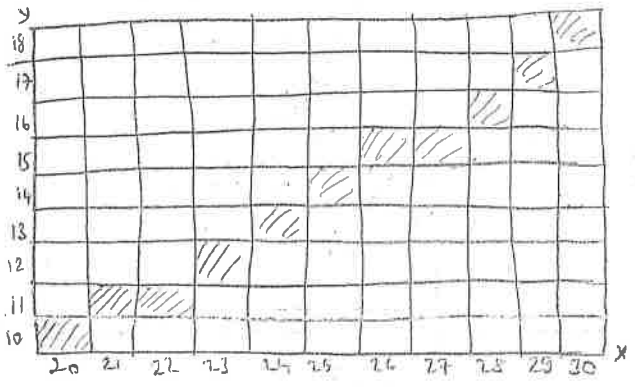
(x_{k+1}, y_{k+1}) → aydınlat

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$

5) 4. adımı $\Delta x - 1$ defa çalıştır.

Örnek:

$(20, 10)$ ve $(30, 18)$ noktasına kadar çizgi çizmek isteniyor.



$\Delta x = 30 - 20 = 10$

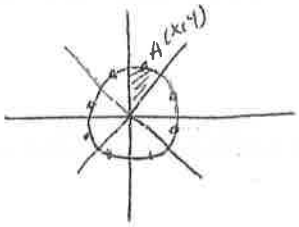
$\Delta y = (18 - 10) = 8$

$2\Delta y - 2\Delta x = -4$

$2\Delta y = 16 \quad 2\Delta x = 20$

$P_0 = 2 \times 8 - 10 = 6$
$P_1 = 6 + (-4) = 2$
$P_2 = 2 + (-4) = -2$
$P_3 = -2 + 16 = 14$
$P_4 = 14 + (-4) = 10$
$P_5 = 10 + (-4) = 6$
$P_6 = 6 + (-4) = 2$
$P_7 = 2 + (-4) = -2$
$P_8 = -2 + 16 = 14$
$P_9 = 14 + (-4) = 10$
$P_{10} = 10 + (-4) = 6$

* Çember çizme Algoritması (Basit) *



* Bu A noktasını bulursak tam çevreyi buluruz.

Orta nokta çember Algoritması (midpoint circle algorith)

1) $(x_0, y_0) = (0, r)$

2) $p_0 = 1 - r$

3)

$p_k < 0$	$p_k \geq 0$
(x_{k+1}, y_k) → aydınlat	(x_{k+1}, y_{k+1}) aydınlat
$p_{k+1} = p_k + 2x_{k+1} + 1$	$p_{k+1} = p_k + 2x_{(k+1)} + 1 - 2y_{k+1}$

4) 8 yollu simetriyi kullanarak diğer noktaları çiz.

5) $x >= y$ olana kadar 3. ve 4. adımı tekrarla.

code:

```
x=0;
y=r;
d=1-r;
w pixel (x,y);
while (y≥x) → xp=x;
                yp=y;
  x++;
  if (d<0)
    d=d+2*x+3
    y=y;
  else
    y=y-1;
  p = p + (y-1/2)^2 - (yp-1/2)^2 - 4(xp+1)
```

örnek:

* Merkezi (0,0) olan yarıçapı 10 olan orta nokta çember Algoritması

$(0,0) = (0,10)$

$P_0 = 1 - r = 1 - 10 = -9$

k	P_k	x_k	y_k
0	$P_0 = -9$	1	10
1	$P_1 = -6$	2	10
2	$P_2 = -1$	3	10
3	$P_3 = 6$	4	9
4	$P_4 = -3$	5	9
5	$P_5 = 8$	6	8
6	$P_6 = 3$	7	7
		—	—

$x >= y$

$P_1 = -9 + 2*1 + 1 = -6$

$P_2 = -6 + 2*2 + 1 = -1$

$P_3 = -1 + 2*3 + 1 = 6$

$P_4 = 6 + 1*4 + 1 - 2*9 = -3$

$P_5 = -3 + 2*5 + 1 = 8$

$P_6 = 8 + 2*6 + 1 - 2*8 = 5$

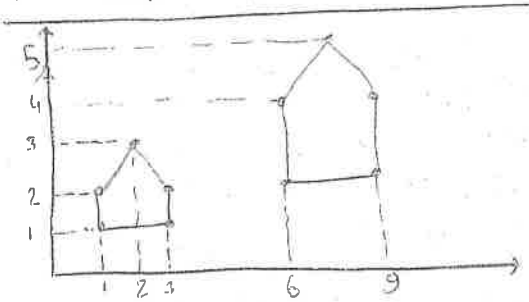
6 Geometrik Dönüşümler

* 3 temel dönüşüm işlemi vardır: (ötelleme, ölçeklendirme ve büyütüp küçültme)

~~1 - Nesne üzerinde yapılan işlemler~~

2 - Kamera üzerinde yapılan işlemler.

İki boyutta ölçeklendirme işlemi



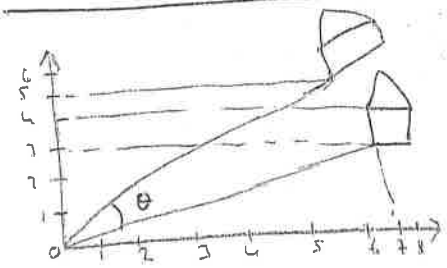
$v = \begin{bmatrix} x \\ y \end{bmatrix}$ → orjinal vertex

$v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ → new vertex

* Yaklaştırmak isterken ötelemiş olduk. Bu istenmeyen durumdur.

Bunu yok etmek için orjine geliyoruz.

İki boyutta döndürme işlemi



$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

θ → saat yönünün tersinde olmak zorunda

θ küçülürse → süreklı olarak görürüz.

θ büyürse → atlamalı olarak görünür.

* 3D Dönüşümler *

Ölçeklendirme:

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

Döndürme:

x eksen:

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

y eksen:

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

z eksen

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

2 Boyut Devam

Öteleme

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$x' = x + tx$

$y' = y + ty$

Ölçeklendirme

* Sx, Sy x-y yönündeki ölçek fonk. olsun.

$x' = x \cdot x \cdot Sx$

$y' = y \times Sy$ } orjindeki noktaya göre

Döndürme

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

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

not/ Sx, Sy, Sz eşit ise homojen

* Homojen koordinatlar *

1) Ekranda gözükmeyen çok küçük değerleri gözükebilir hale getirebiliriz

$P[x\ y] \quad P_h[x\ y\ w]$

$P[3,2\ \ 6,8] \longrightarrow [32\ \ 68]$

$\frac{3,2}{0,1} \quad \frac{6,8}{0,1}$

$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$

2) Çok büyük değerleri ekranda göstermek için kullanılır. Bu durumda matrisin boyutu 1 artar. $2\times2 \rightarrow 3\times3$, $3\times3 \rightarrow 4\times4$ } haline getirin

3) Öteleme matrisini kesinlikle homojen koordinat olarak ifade edebiliyoruz.

$T = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$ ← Translation (öteleme)

$T_v = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+dx \\ y+dy \\ 1 \end{bmatrix} = X'$

Scaling (ölçeklendirme) $= \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$

rotation (döndürme) $= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Örnek

$x=15$

$y=17$

Scaling

$\begin{bmatrix} 15 & 0 & 0 \\ 0 & 17 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Ters dönüşümlerde :

Scaling :

$\begin{bmatrix} 1/sx & 0 & 0 \\ 0 & 1/sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotation :

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

Translation :

$\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$

(7)

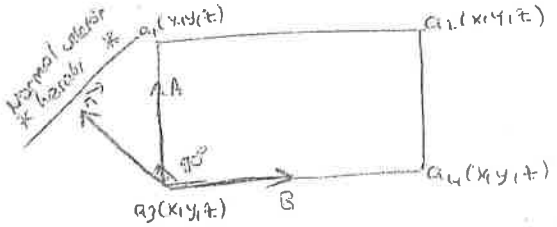
* Aydınlatma (Gölgeleme) Teknikleri *

* 2 yada 3 boyutlu uzayda nesne üzerinde diğer ışık kaynaklarına göre aydınlatma işlemlerinin yapılmasıdır.

$\theta = 0^\circ$ ise → En parlak bölge

$\theta \geq 90^\circ$ ise → karanlık

Normal vektör = yüzeye dik olan vektördür.



$a \times b \neq b \times a$

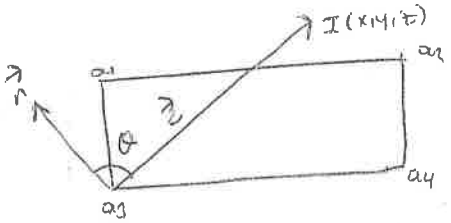
$$\vec{A} = (a_{1x} - a_{3x})\vec{i} + (a_{1y} - a_{3y})\vec{j} + (a_{1z} - a_{3z})\vec{k}$$

$$= A_x\vec{i} + A_y\vec{j} + A_z\vec{k}$$

$$\vec{B} = (a_{4x} - a_{3x})\vec{i} + (a_{4y} - a_{3y})\vec{j} + (a_{4z} - a_{3z})\vec{k}$$

$$= B_x\vec{i} + B_y\vec{j} + B_z\vec{k}$$

$$A \times B = \begin{bmatrix} i & j & k \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{bmatrix} = (A_y \cdot B_z - A_z B_y)\vec{i} - (A_x B_z - A_z B_x)\vec{j} + (A_x B_y - A_y B_x)\vec{k}$$



$$\vec{C} = (a_{3x} - I_x)\vec{i} + (a_{3y} - I_y)\vec{j} + (a_{3z} - I_z)\vec{k}$$

$x_{max} \Rightarrow \vec{C}$ ile $\vec{n}$ arasındaki açıyı bulabiliriz.

* Açı hesabı *

Scaler çarpım ile;

$$\vec{A} \cdot \vec{B} = |\vec{A}| \cdot |\vec{B}| \cdot \cos\theta$$

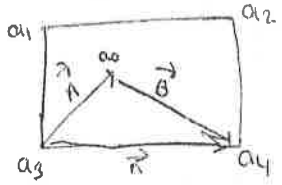
$$A_x B_x + A_y B_y + A_z B_z = \sqrt{A_x^2 + A_y^2 + A_z^2} \cdot \sqrt{B_x^2 + B_y^2 + B_z^2} \cdot \cos\theta$$

$$\theta = A \cdot \cos\left(\frac{A_x B_x + A_y B_y + A_z B_z}{\sqrt{A_x^2 + A_y^2 + A_z^2} \cdot \sqrt{B_x^2 + B_y^2 + B_z^2}} \right)$$

* Yöntemler *

1) Sabit Gölgeleme (Constant Shading)

* Herbir yüzey, nesne parçasının orta noktasındaki yüzey normaline göre θ açısı hesaplanır.



$$a_0 = \frac{\Sigma a}{4}$$

$$\vec{N} = \vec{A} \times \vec{B}$$

$$\vec{A} = (a_{0x} - a_{3x})\vec{i} + (a_{0y} - a_{3y})\vec{j} + (a_{0z} - a_{3z})\vec{k}$$

$$\vec{B} = (a_{4x} - a_{0x})\vec{i} + (a_{4y} - a_{0y})\vec{j} + (a_{4z} - a_{0z})\vec{k}$$

Not:
Sabit gölgeleme algoritması yüzey parçasının orta noktasındaki yüzey normali ile ışık kaynağı vektörü arasındaki açının hesaplanmasıdır.

θ açısına göre renk seçimi:

R G B (Red Green Blue)

8 bit $= 2^8 = 256$ [0, 255]

Red → [0, 255] Açı = 0 - 90°

$k = 255 / 90 \cong 2{,}84$

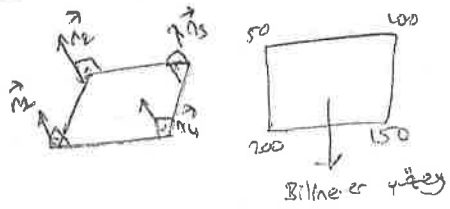
$\theta = 0$ ise $\Rightarrow 255 - \underbrace{\text{Açı} \times 2{,}84}_{0} = 255$

$\theta = 30$ ise $\Rightarrow 255 - 30 \times 2{,}84 = 170$

$\theta = 90$ ise $\Rightarrow 255 - 90 \times 2{,}84 = 0$

* Görüntü kalitesinin iyi olmamasıdır. Eğrisel modellerde parça parça birleştirilmiş gibi görünür.
* Sabit gölgeleme hızlıdır.

2) Ground Shading



* Kullanılan yüzey parçacığının herbir köşesindeki renk değeri hesaplanır. Sonra bu renk değeri yüzey üzerine bilineer yüzey modelleme yöntemiyle dağıtılır.
* Sabit gölgelemeye göre daha net görüntü verir.
* Hızlı ton geçişlerinin olduğu yerlerde sıkıntılıdır.

3) Phong gölgeleme modeli

* Bu model nesne üzerindeki düşen piksellere bütün ışığı hesaplar. Dolayısıyla diğer 2 algoritmadaki problemlerle karşılaşmaz.
* İşlem yükü fazladır. * Kütüphaneleri açık kaynak kodludur.

OPENGL, WEBGL } kütüphane olarak Phong kullanır.