

# ÜÇGENLEME ALGORITMASI KONG

Mustafa Erkin Dal  
Bilgisayar Mühendisliği 3. Sınıf

21360859047

21/03/2024

# İÇİNDEKİLER

## 1 – Üçgenleme Algoritaması Nedir?

1.1 – Neden Üçgen?

## 2 – Kong Algoritması

2.1 – Sözde Kod

2.2 – Ear Tanımı ve İç Bükeylik

2.3 – Algoritma Aşamaları

2.4 – Şekil Oryantasyonu Önemi

## 3 – Algoritma Metodları

3.1 – Poligon Oryantasyonu Tespiti

3.2 – İç Bükey Noktaların Bulunması

3.3 – Ear Kontrolü

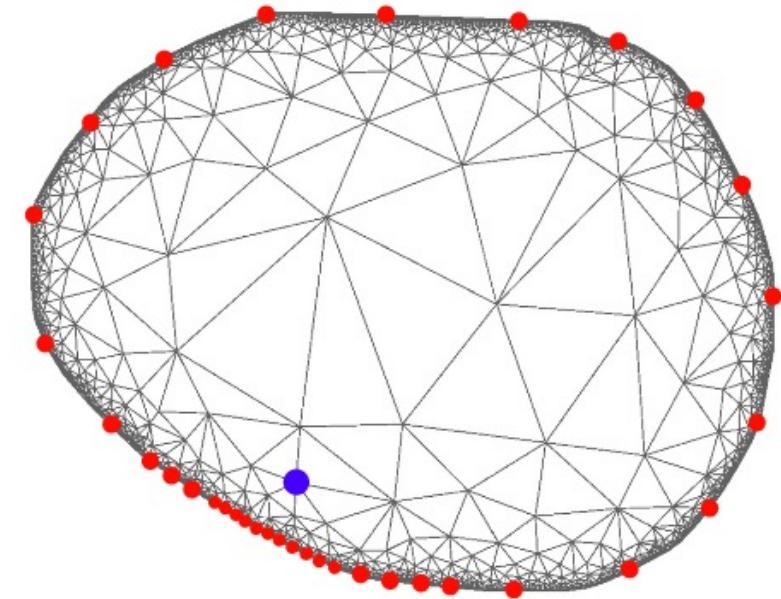
4 – Uygulama Resimleri

5 – Kaynaklar

6 - Sorularınız

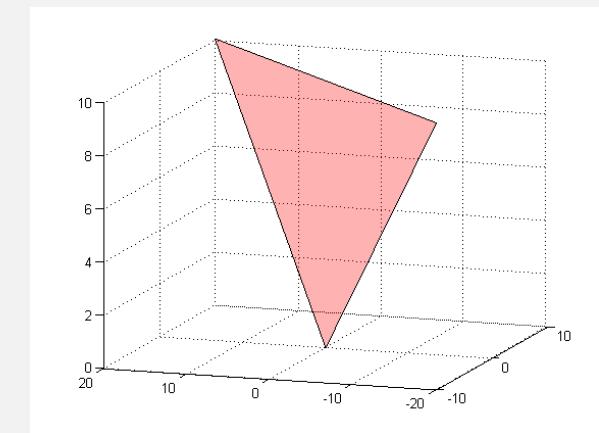
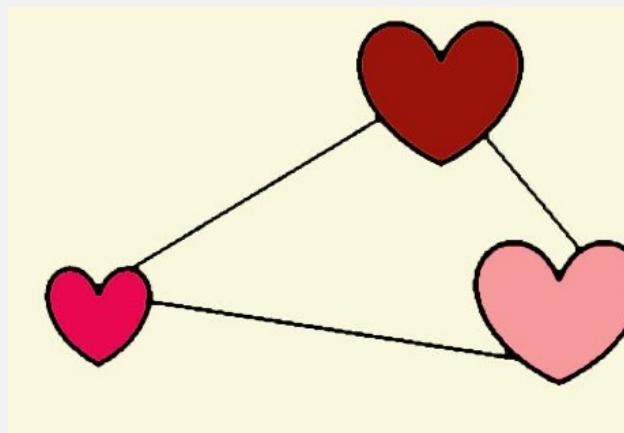
# I- ÜÇGENLEME ALGORITMASI NEDİR?

- Bir yüzeyin üçgenlerle bölünmesini sağlayan algoritmadır.
- Görüntü işleme uygulamalarında ve geometri problemlerinde kullanılır.



## I.I – NEDEN ÜÇGEN?

- Basit Geometri
- Noktalarında Aynı Düzlem Garantisi
- İç Bükeylik Durumu Olmaması
- Güvenilir ve Mühendis Dostu



## 2 – KONG ALGORITMASI

KONG

ERKIN

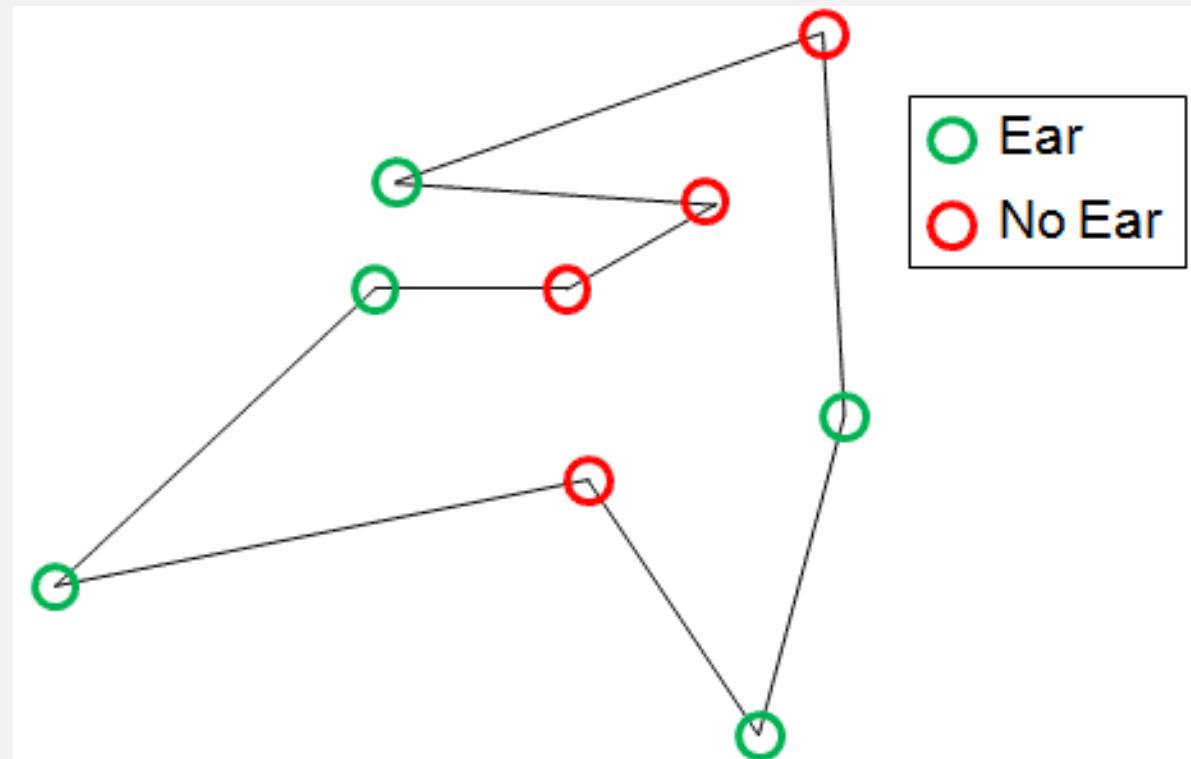
## 2. I – SÖZDE KOD

```
// Precondition: R contains all non-convex points of polygon
BOOL isEar(Vertice P-1, Vertice P, Vertice P1)
begin
    If R is empty
        return False
    else
        if x is convex
        then
            if InsideAreaOfTriangle(P-1, P, P1) contains no point of R
                return true
            else
                return false
        else
            return false
end

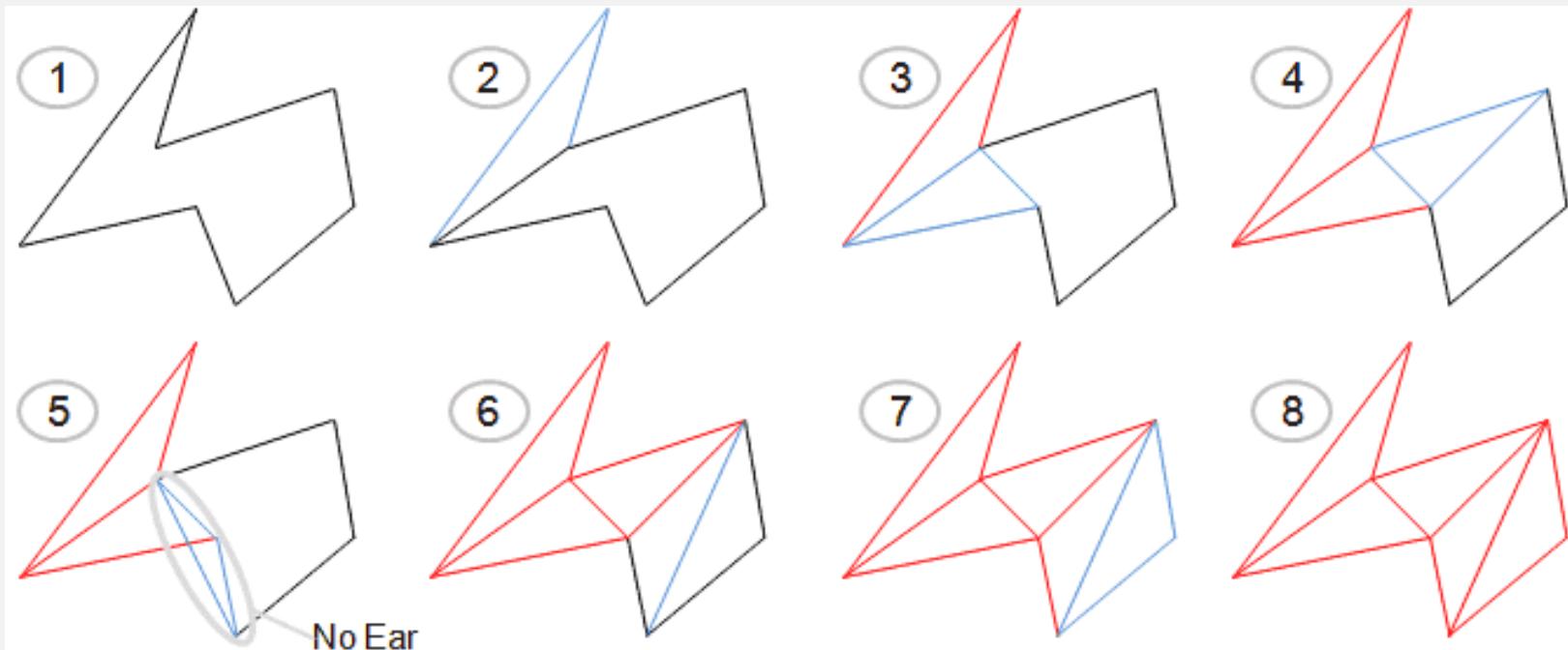
// Preconditions:
// Q contains all points/vertices of the polygon
void DoKong()
begin
    while |Q| > 3
        if isEar(P-1, P, P1) then
            addTriangle(P-1, P, P1)          // e.g. to a list
            remove P from Q
            P = P-1;
        else
            P = P1;
end
```

## 2.2 – EAR TANIMI VE İÇ BUKEYLIK

- Ear : Kendisinden önceki ve sonraki noktalar arasından bir kenar çizildiğinde tamamen şeklin içinde kalan ve başka herhangi bir kenara temas etmeyen nokta
- İç bükey hiç bir nokta Ear olamaz!
- Ear bulunduğuunda kesiliyor.

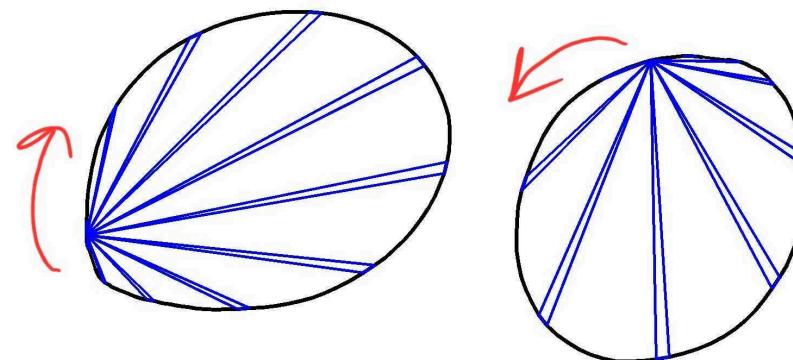


## 2.3 – ALGORITMA AŞAMALARI



## 2.4 – ŞEKİL ORYANTASYONU ÖNEMI

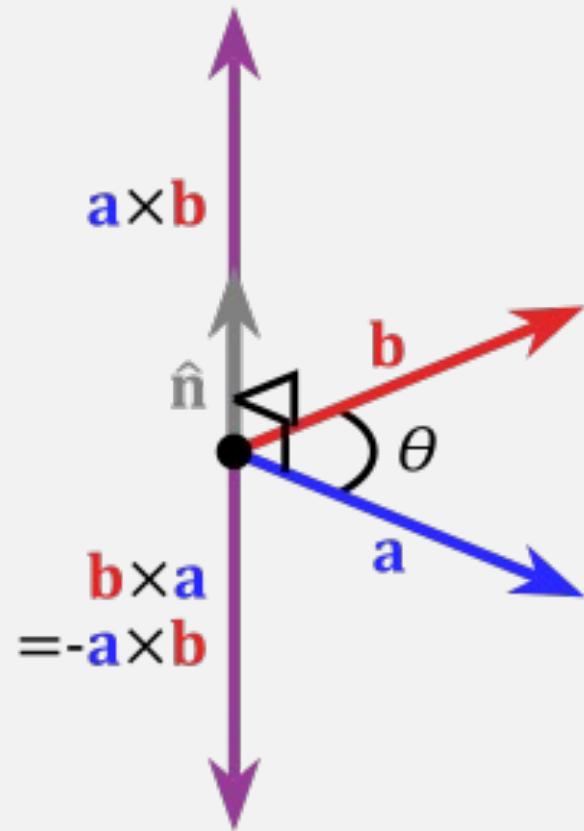
- Şekli hangi yöne doğru çizdiğimiz önemli.
- Ear bulmaya çalışırken noktanın iç bükeyliğine dikkat ediyoruz. Bunun için çizim yönü bilinmeli.



## 3– ALGORITMA METODLARI

### 3. I – POLİGON ORYANTASYONU TESPİTİ

- 1- Başlangıç noktası seç (en küçük x olan)
- 2- Bir Önceki ve sonraki noktaları al
- 3- Başlangıç noktasından bu iki noktaya olan vektörlerin vektörel çarpımının işaretine bak
- Vektörel çarpımda yön belirleyicidir.



## 3.2 – İÇ BUKEY NOKTALARIN BULUNMASI

- Poligon oryantasyon yönü ve sıradı gelecek olan noktaların yön vektörü kullanılır.
- Bu iki vektörün vektörel çarpımlarının toplamının oryantasyon vektörüne göre olan durumuna göre karar verilir.

Handwritten mathematical notes and diagrams include:

- $F = \frac{q_1 q_2}{4\pi \epsilon_0 \epsilon r^2}$
- $\Phi = \int \delta \cos \alpha ds$
- $\hat{F} = \frac{U_m}{L}$
- $W_n = k/(\Delta n)^2$
- $C_r = \frac{r^2}{2} R$
- $I = \int_{t_0}^{t_1} I_0 \sin^2 \alpha dt$
- $M = I \bar{E}$
- $I = \frac{U}{R}$
- $\langle D \rangle = \frac{n_2 - n_1}{n_1 - n_2}$
- $\bar{a} = \bar{a}_n + \bar{a}_{n+1}$
- $\Delta S = S_n - S_{n+1}$
- $A = \oint L ds$
- $A = \oint L ds e^{-pt}$
- $A = p(V_L - V_i)$
- $A = \frac{\pi}{4} RT \ln \frac{V_f}{V_i}$
- $Q = \Delta U + A$
- $\sigma = \frac{dQ}{dT}$
- $C = C_p M$
- $S_2 - S_1 = \frac{1}{T} \int dQ$
- $\vec{E} = \sum_{i=1}^N E_i$
- $\Psi(x)$
- $\lambda = R z^2 \left( \frac{1}{m_2} - \frac{1}{n_2} \right)$
- $P = mg$
- $C = \frac{C_0 \cdot \mathcal{E}}{S}$
- $L = \mu \mu_0 \cdot V$
- $T_0 = 2\pi \sqrt{\frac{m}{k}}$
- $\gamma = \ln \frac{f(t)}{f(t+\tau)}$
- $\Psi_n = \sqrt{\frac{2}{l}} \sin \frac{n\pi x}{l}$
- $V_k = \frac{A}{h}$
- $v = \frac{1}{T}$
- $\omega = 2\pi v$
- $T = \frac{2\pi}{\omega}$
- $\chi = \beta T$
- $E = mc^2$
- $\infty = f \cos(\omega t + \alpha)$
- $\omega = 2\pi v$
- $\Phi = \beta S \cos \alpha$
- $\sigma = 5,67 \cdot 10^{-8} \frac{B^2}{K^4}$
- $W = |\Psi|^2$
- $R = \alpha \sigma T^4$
- $\alpha = A_0 e^{-\beta t} \cos(\omega t + \alpha)$
- $\lambda_m = \frac{b}{T}$
- $b = 2,9 \cdot 10^{-3} m \cdot K$
- $\varphi = \arctan \frac{A_1 \sin \alpha_1 + A_2 \sin \alpha_2}{A_1 \cos \alpha_1 + A_2 \cos \alpha_2}$
- $\Delta = \pm m \lambda_m, m = 0, 1, 2, \dots$
- $\nu = \frac{c}{\lambda}$
- $\omega = 2\pi v$
- $T = \frac{2\pi}{4\pi^2 \cdot \beta^2}$
- $k = \frac{2\pi}{\lambda}$
- $\lambda = vT$
- $f_0 = \frac{f_0}{2\beta + \omega_s - \beta^2}$
- $W = \frac{1}{2} m \theta^2 \omega^2$
- $\xi = f \cos(\omega t - kx)$
- $k = 138 \cdot 10^{-11}$
- $\xi = \tilde{p}_x + \tilde{p}_y + \tilde{p}_z$
- $f_0 = \frac{f_0}{2\beta + \omega_s - \beta^2}$
- $\Delta \varphi = \frac{2\pi}{\lambda} \Delta x$
- $P = nkT$
- $M = F \circ$
- $\eta = \frac{1}{3} P \langle v \rangle \lambda$
- $U = \frac{1}{2} \frac{m}{\mu} \theta^2 T$
- $\frac{P}{T} \cdot \frac{m}{\mu} \cdot k = \frac{3}{4} M$
- $\nu = \frac{h}{\lambda} = \frac{c}{v}$
- $\sigma = e n (u_n + u_p)$
- $\mathcal{E}_1 = \frac{3}{2} \hbar \omega (n=1)$
- $\mathcal{E}_2 = \frac{5}{2} \hbar \omega (n=2)$
- $\mathcal{E}_3 = \frac{7}{2} \hbar \omega (n=3)$
- $\mathcal{E}_4 = \frac{9}{2} \hbar \omega (n=4)$
- $\mathcal{E}_5 = \frac{11}{2} \hbar \omega (n=5)$
- $D = \frac{1}{3} \langle v \rangle \lambda$
- $\Delta = L_2 - L_1$
- $\mathcal{E} = \frac{q}{4\pi \epsilon_0 \epsilon r^2}$
- $\mathcal{E}_1 = \frac{3}{2} \hbar \omega (n=1)$
- $\chi = \eta \frac{1}{2} \frac{R}{\mu}$
- $\vec{P}_n = \frac{3\pi}{8} \frac{r}{ne}$
- $P = P_0 e^{-\frac{h}{\lambda}}$
- $\psi = N \varphi$
- $\Delta u = \frac{\Delta v}{v_0}$
- $f(v) = 4\pi \left[ \frac{2\pi k T}{m_0} \right]^{1/2} v^2 e^{-\frac{mv^2}{2kT}}$
- $\Delta u = \frac{\Delta v}{v_0}$
- $\lambda_K = \frac{\hbar}{A}$
- $\vec{E} = \frac{\vec{F}}{q_0}$
- $w = mg h$
- $F_{n\mu} = \mu N$
- $\zeta_s = -L \frac{dl}{dx}$
- $\langle v \rangle = \sqrt{\frac{8kT}{\pi m_0}} = \sqrt{\frac{8kT}{\mu N}}$
- $A = F \Delta s \cos \alpha$

## 3.3 – EAR KONTROLÜ

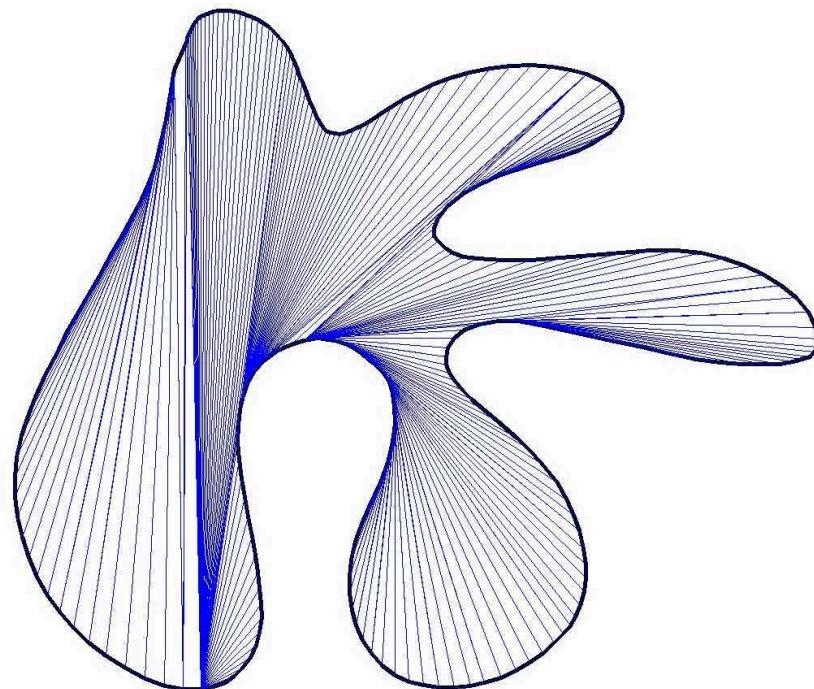
- İçbükey noktalar Ear olamaz
- Dışbükey noktaların komşuları ile oluşturduğu üçgen başka bir noktayı içeriyor mu diye bakılır.
- İçermiyorsa Ear'dır.

```
public static boolean isInside(Point x, Point y, Point z, Point p)
{
    Point v1 = new Point(x: y.x - x.x, y: y.y - x.y);
    Point v2 = new Point(x: z.x - x.x, y: z.y - x.y);

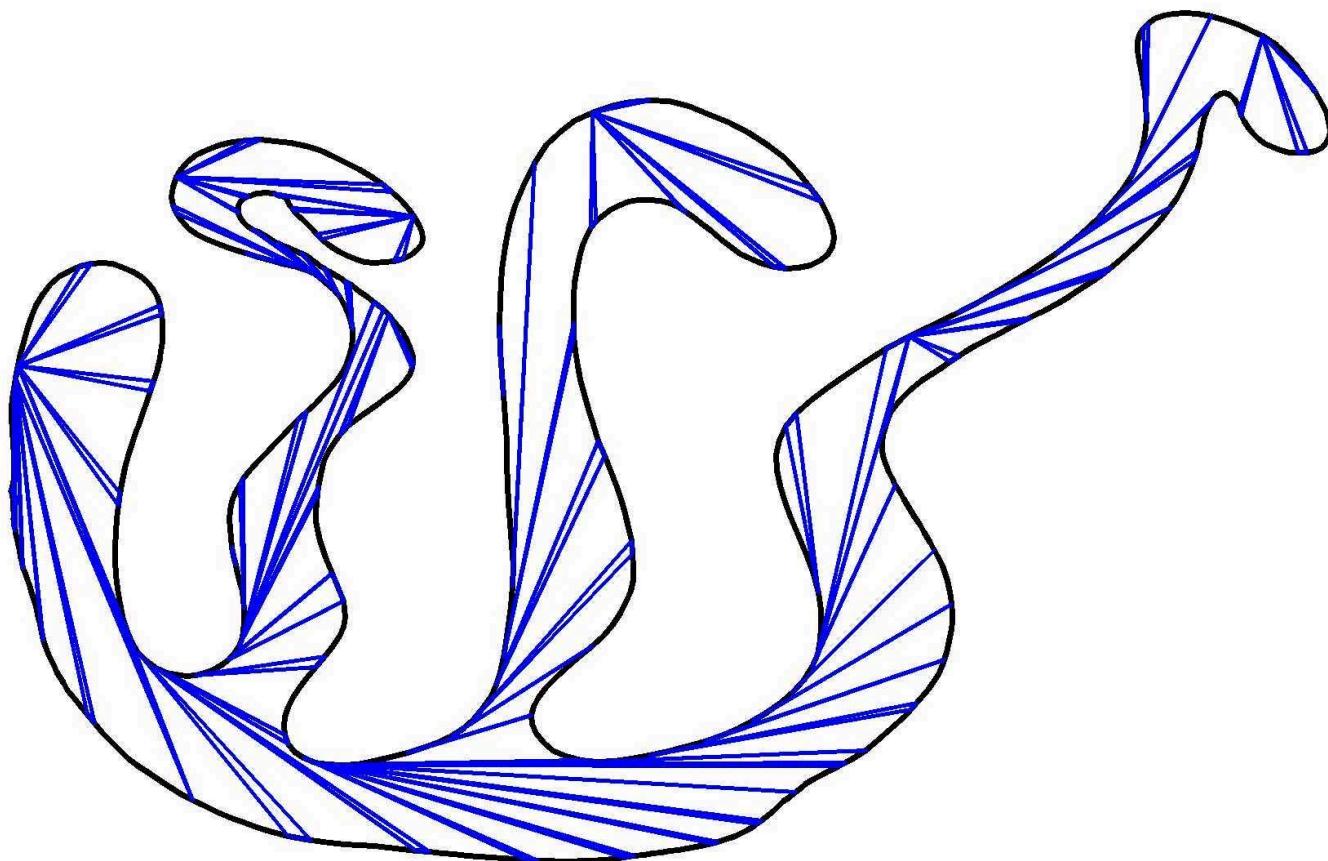
    double det = v1.x * v2.y - v2.x * v1.y;
    Point tmp = new Point(x: p.x - x.x, y: p.y - x.y);
    double lambda = (tmp.x * v2.y - v2.x * tmp.y) / det;
    double mue = (v1.x * tmp.y - tmp.x * v1.y) / det;

    // >0 implies that they are in the same direction
    // (lambda+mue) < 1 to check if the point lies on the vertex
    // if so one CP will be 0 and the other will be = det.
    return (lambda > 0 && mue > 0 && (lambda+mue) < 1);
}
```

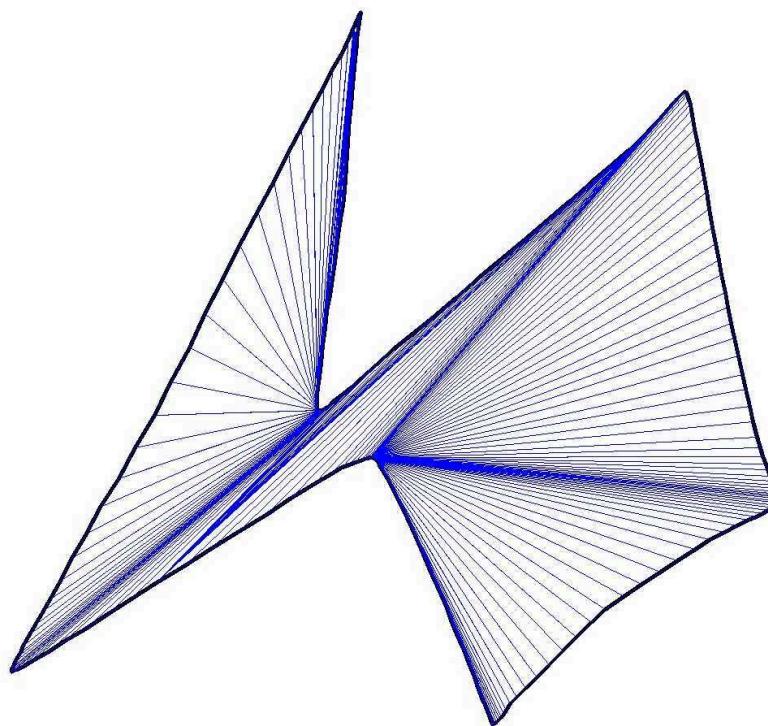
## 4 – UYGULAMA RESIMLERİ



## 4 – UYGULAMA RESIMLERİ

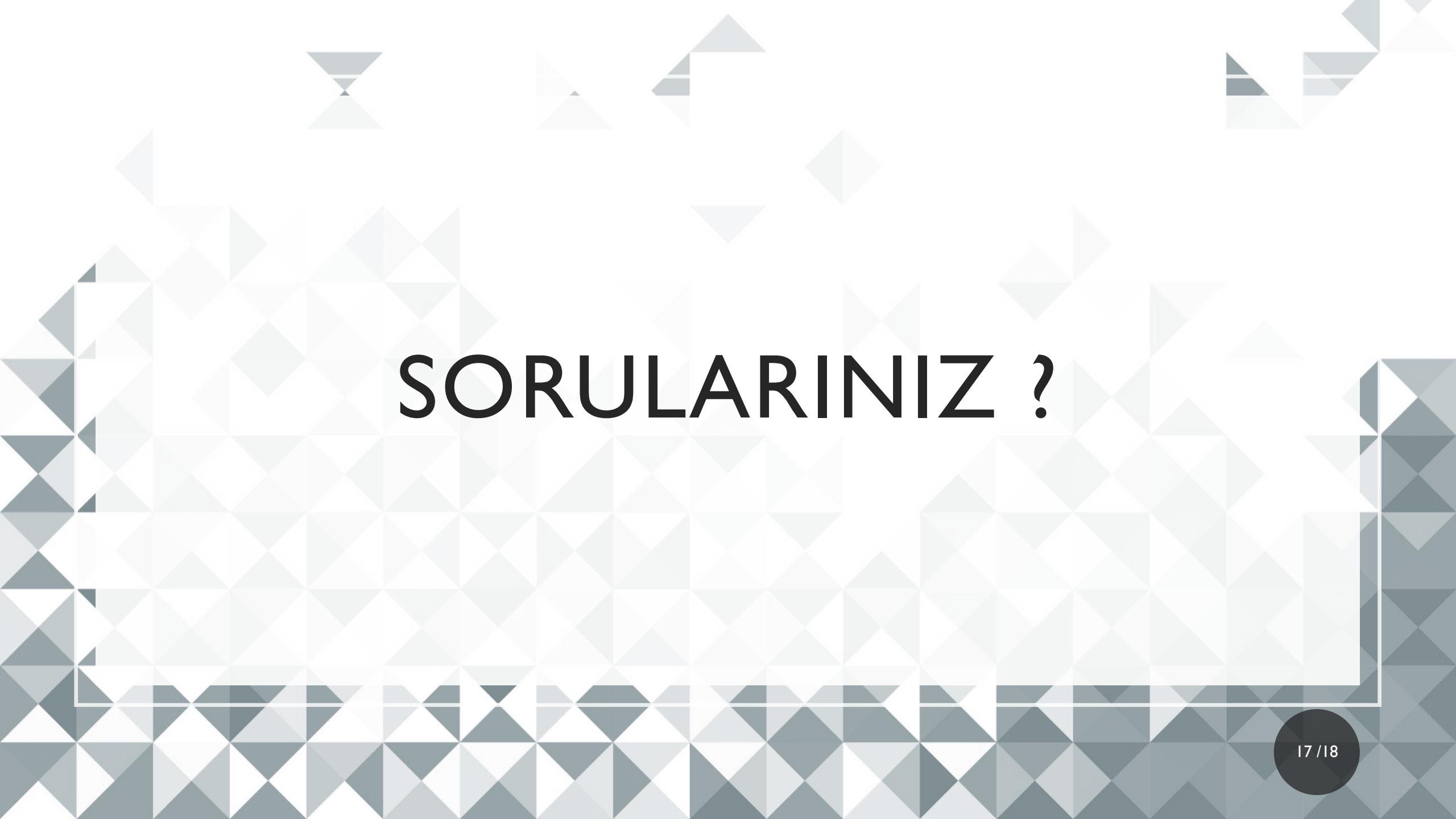


## 4 – UYGULAMA RESIMLERİ



## 5 - KAYNAKÇA

- <http://www.sunshine2k.de/coding/java/Polygon/Kong/Kong.html>



# **SORULARINIZ ?**



**TEŞEKKÜRLER**