**Kocaeli Üniversitesi**

**Bilgisayar Mühendisliği Bölümü**

**Programlama Laboratuvarı I**

**Minimum Çevreleyen Çember – B Spline**

*Samet Yavuz*

*190201066@kocaeli.edu.tr*

**Özet**

Programlama Laboratuvarı I dersi 1.projesi için bizden istenen kullanıcı tarafından tamsayı koordinatlı 2 boyutlu düzlemde N tane nokta verildiğinde, tüm noktaları kapsayan *Minimum Çevreleyen Çember*i ve *Spline* eğrisi kullanarak bu N tane noktanın en yakınından geçen bir eğri çizdiren bir algoritma yazmak ve sunmaktır

Projede bizden C programlama dilini ve 3 farklı kütüphaneden birini kullanmamız istendi.

Bu algoritma için geliştirme ortamı olarak CodeBlocks programını kütüphane ve arayüz tasarımı için de *Graphics.h* kütüphanesini kullandım.

Proje dokümanlarında belirtilen ;

1. 2 boyutlu düzlem için gereken N adet noktanın koordinatlarının bir dosya üzerinden okunması (.txt veya .csv) ve dosyanızda her bir noktanın X ve Y koordinatlarının bulunması,
2. Bu (X,Y) koordinatları temsil eden her bir noktanın bir arayüz aracılığı ile (koordinat düzleminde) ekranda gösterilmesi,
3. Bu noktaların tümünü içine alan en küçük daireyi ekrana çizdirilmesi,
4. Tüm noktaları içeren en küçük dairenin hesapladığınız yarıçapını v e merkezini çıktı olarak ve ekranda dairenin üzerinde gösterilmesi,
5. Verilen tüm noktaların (N tane) en yakınından geçen bir eğrinin çizdirilmesi,
6. Raporunuzda kullanmış olduğunuz yöntemin (B Spline yönteminin) matematik detaylarına yer verilmesi,
7. En son olarak da yazdığınız algoritmanın z aman karmaşıklığı hesabı yapılması Yaptığınız tüm hesaplamaları proje raporunuzda gösterilmesi ve açıklaması ile birlikte ayrıntılarıyla anlatılması,

ister olarak belirtilmiş ve tüm gösterimlerimizin de X ve Y koordinat düzlemini kullanmamız ve noktaları ona göre göstermemiz beklenmiştir.

**1.0 GİRİŞ**

Projem için CodeBlocks programını kullandım.

CodeBlocks kullanmamın sebebi ücretsiz olmasının yanında kullanıcı ara yüzünde sağladığı kolaylıklardır. Ayrıca geçen sene Bilgisayar Laboratuvarı derslerinde kullanmayı öğrenmiş olmam da var.

**2.0 YÖNTEM**

**2.1 Bézier Eğrileri – B Spline**

İlk olarak 1959 yılında Renault bünyesinde çalışan Pierre Bézier tarafından ortaya atılmıştır.

Bézier eğrisi matematiksel olarak kontrol noktaları ve onları inşa edecek temel fonksiyon ile tanımlıdır ve temelde Bernstein Polinomuna dayanır. Seçilen ilk kontrol noktası başlangıç, son kontrol noktası ise bitiş noktası olurken aradaki noktalar da eğriye şeklini verecek olan noktalardır. Genellikle eğri ara kontrol noktalarından geçmez.

Matematiksel olarak tanımlamak gerekirse i’ninci kontrol noktası, ise Bernstein Polinomuna karşılık gelir.

Bernstein Polinomu ise Bernstein tarafından, Weierstrass'ın yaklaşım teoreminin basit bir ispatını vermek için inşa edilmiştir.

u [0,1] olmak üzere;

fonksiyonuna Bernstein Polinomu denir

Bu iki formülü birleştirdiğimizde ise N tane kontrol noktasının yakınından geçen bir Bézier eğrisi elde ederiz.

Bu eğrinin formülünü C dilinde kodladım. Buradaki u değerinin fonksiyondaki karşılıklarına konulan noktalar da eğrinin çizgi oluşturmasını sağlıyor. Dolayısıyla u değerlerinin artış miktarı açılan pencerenin çözünürlük oranına göre değişiyor.

**2.2 Minimum Çevreleyen Çember**

**(Minimum Enclosing Circle (MEC))**

İlk önce tüm noktaların orta noktasını bulmak için x ve y koordinatlarını topladım ve nokta sayısına böldüm.

Daha sonra elde ettiğim noktaya en uzak noktayı buldum ve orta noktayı bu noktaya yakınlaştırmak için değeri 1 olan bir sayıyla çarpıp, çarptığım sayıyı 0.001 azaltarak 0’a yakınlaştırdım. Bu küçülme sırasında orta noktaya en uzak noktayı sürekli hesaplattım. Ta ki orta noktaya başka bir nokta, daha uzak olana kadar.

Böylece orta nokta yeni uzak noktaya doğru kaymaya başladı ve böyle devam etti.

Algoritma bittiğinde ise minimum iki nokta orta noktaya en uzak kaldı. Bu orta nokta Minimum Çevreleyen Çemberin merkeziydi. En uzak noktaya olan uzaklığı da çemberin yarıçapını vermiş oldu ve “circle()” fonksiyonuyla daireyi çizdirdim.

**3.0 Yalancı Kod**.

1- Başla.

2- Grafik penceresini aç.

3- Koordinat sistemini çizdiren fonksiyonu çağır.

1- X koordinat sayılarını yaz.

2- Y koordinat sayılarını yaz.

3- Yatay çizgileri çiz.

4- Dikey çizgileri çiz.

5- X0 ve Y0 doğrularını kırmızı yap.

4- Noktaları dosyadan alıp iki tane diziye at.

5- Noktaları sırala.

6- Noktaları koordinat sistemine yerleştir.

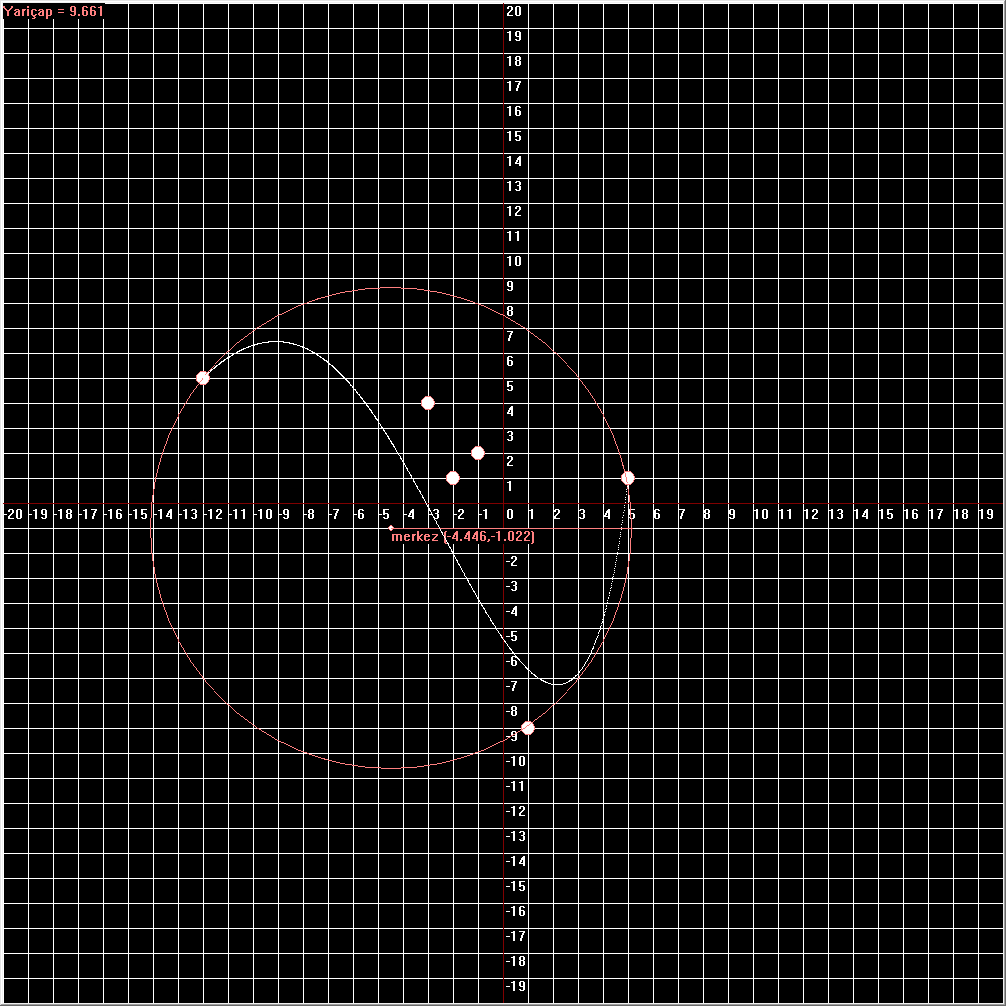
7- Minimum Çevreleyen Çemberi bul.

1- Noktaların X ve Y değerlerinin ortalamasını bul.(Geçici merkez)

2- Ortalamadan elde edilen noktaya en uzak noktayı bul.

3- En uzak ikinci noktayı bulana kadar geçici merkezi noktaya yakınlaştır.

4- Varsa üç, yoksa iki noktaya de eşit uzaklıktaki noktayı merkez belirle

 5- En uzak noktaya olan mesafeyi yarıçap belirle.

6- Çemberi çiz,

7-Yarıçap doğrusunu ve merkez noktasını yazdır.

8- B Spline eğrisini bul.

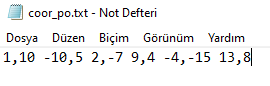
1- Nokta sayısına göre X ve Y için Bézier denklemi bul.

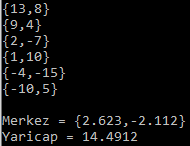
2- 0’dan 1’e kadar 0,001 arttırarak denkleme yaz.

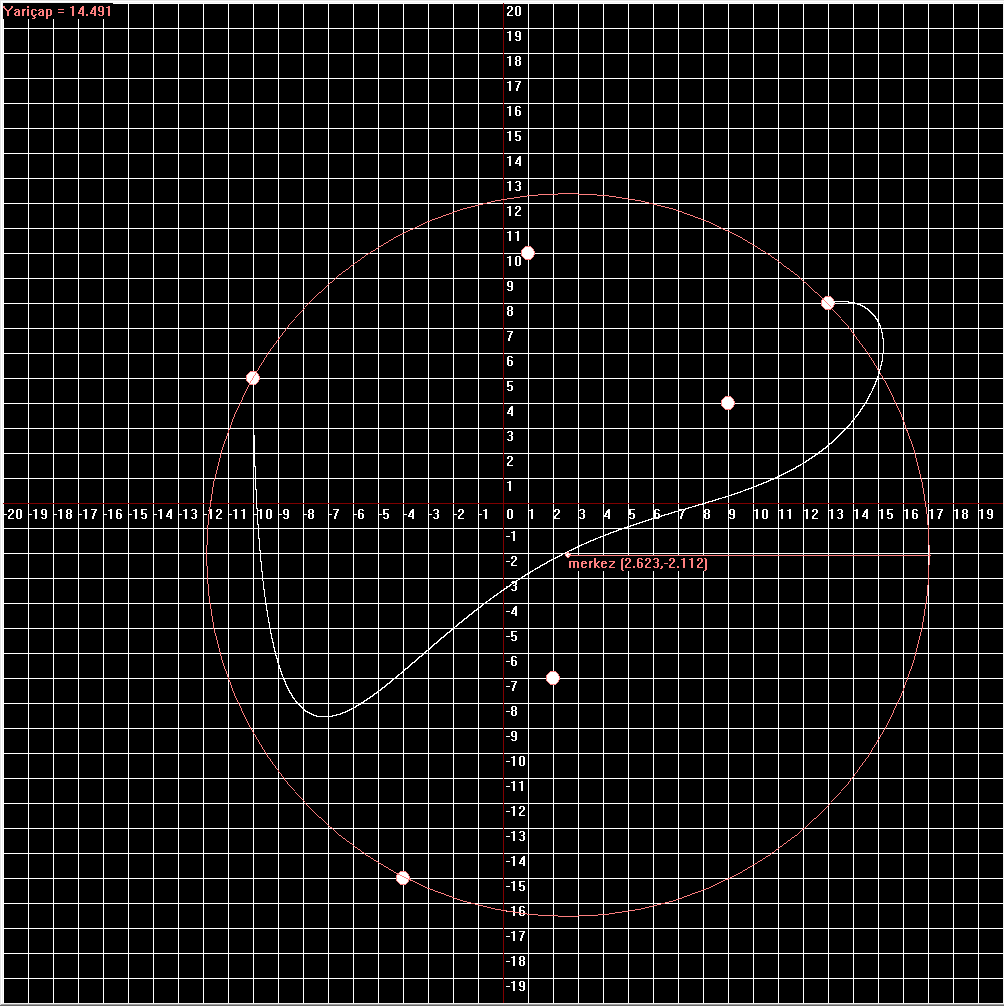
3- Çıkan her X ve Y koordinatlarının gösterdiği pikselleri boya.

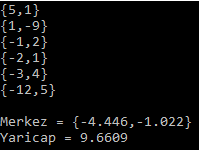
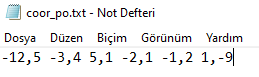
9-Bitir.

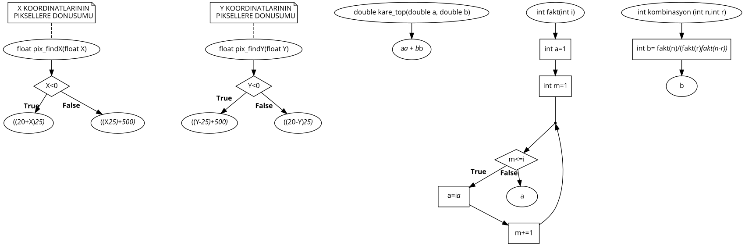
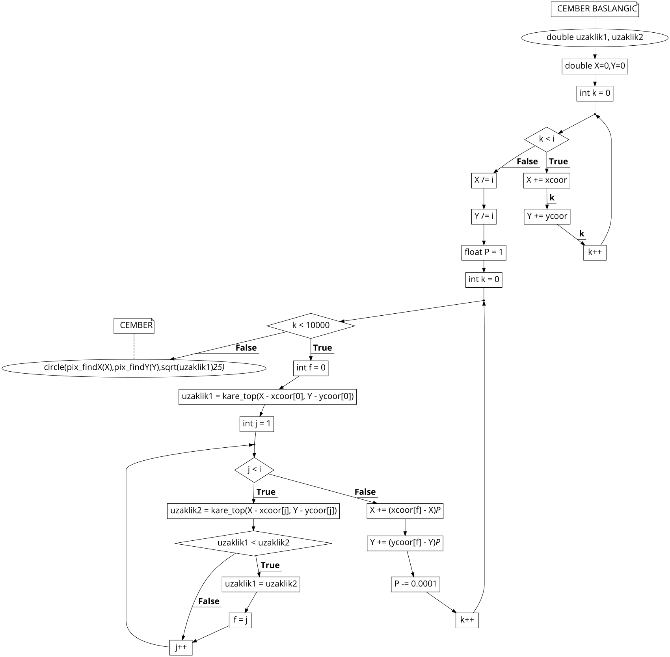
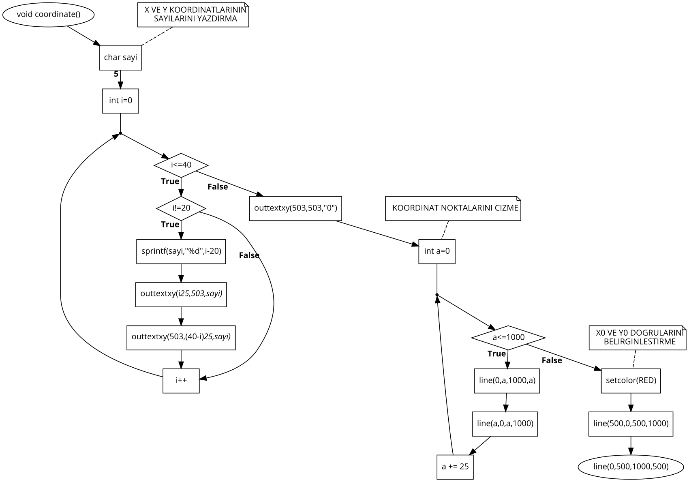
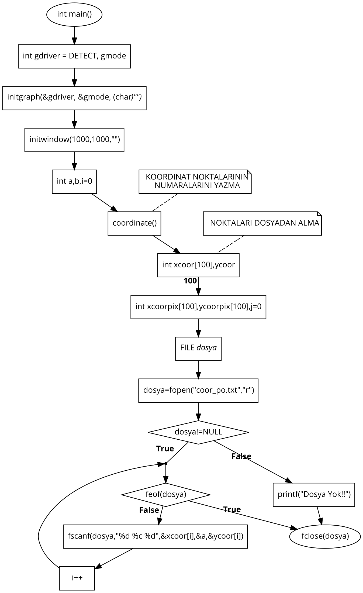
**4.0 Deneysel Sonuçlar**

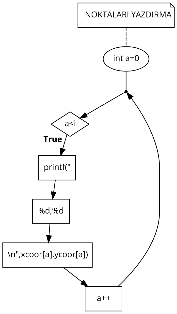
**4.1**

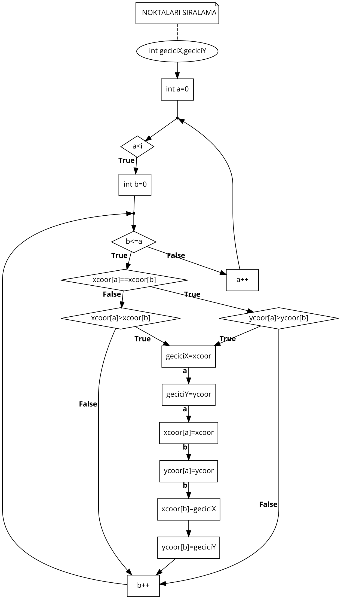


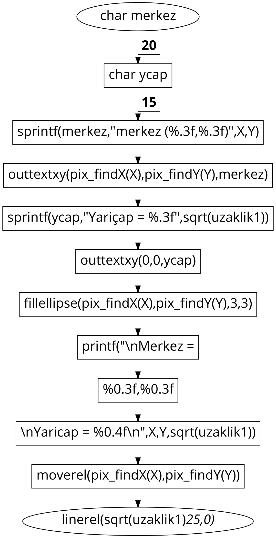


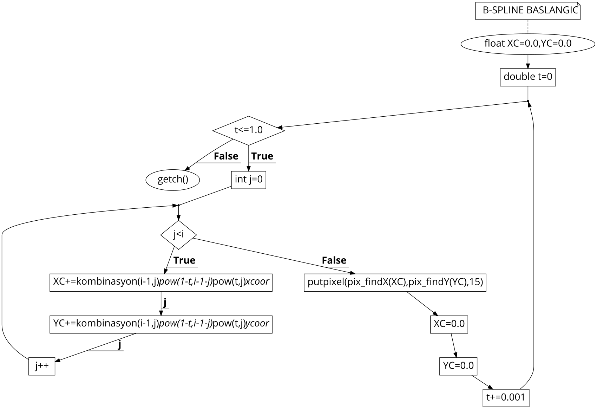
**4.2**

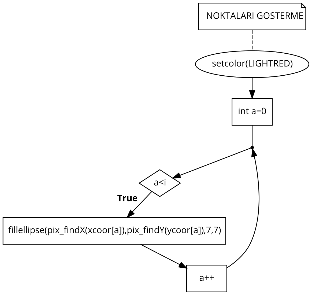
**5.0 Akış Diyagramı**

****

****

****

****

****

**6.0 Big O - Algoritma Karmaşıklık Hesabı**

**6.1 B Spline**

float XC=0.0,YC=0.0; (2)

for(double t=0; t<=1.0; t+=0.001) (1000)

{

for(int j=0; j<i; j++) (N)

{

XC+=kombinasyon(i-1,j)\*pow(1-t,i-1-j)\*pow(t,j)\*xcoor[j]; (1)

YC+=kombinasyon(i-1,j)\*pow(1-t,i-1-j)\*pow(t,j)\*ycoor[j]; (1)

}

putpixel(pix\_findX(XC),pix\_findY(YC),15); (1000)

XC=0.0; (1000)

YC=0.0; (1000)

}

O(2+1000\*2\*N+1000+1000+1000)=O(2000\*N+3002)=O(N)

**6.2 Minimum Çevreleyen Çember**

double uzaklik1, uzaklik2; (2)

double X=0,Y=0; (2)

for (int k = 0; k < i; k++) (N)

{

X += xcoor[k]; (1)

Y += ycoor[k]; (1)

}

X /= i; (1)

Y /= i; (1)

float P = 1; (1)

for (int k = 0; k < 10000; k++) (10000)

{

int f = 0; (1)

uzaklik1 = kare\_top(X - xcoor[0], Y - ycoor[0]); (1)

for (int j = 1; j < i; j++) (N)

{

uzaklik2 = kare\_top(X - xcoor[j], Y - ycoor[j]); (1)

if (uzaklik1 < uzaklik2)(1)

{

uzaklik1 = uzaklik2;(1)

f = j;(1)

}

}

X += (xcoor[f] - X)\*P; (1)

Y += (ycoor[f] - Y)\*P; (1)

P -= 0.0001; (1)

}

circle(pix\_findX(X),pix\_findY(Y),sqrt(uzaklik1)\*25); (1)

O(1+1+2\*N+1+1+1+10000\*(1+1+N\*(1\*(1+1)))+1+1+1)=O(20008+20000i)=O(i)

**7.0 Sonuç**

Bu proje sayesinde C programlama dilini daha etkin bir şekilde kullanmayı, graphics.h kütüphanesinin kullanmayı ve algoritmada bellek kullanımının önemini öğrenmiş olmakla birlikte bir proje hazırlamanın zorluklarını tecrübe etmiş oldum.

**8.0 Kaynakça**

***(PDF) Solution methodologies for the smallest enclosing circle problem*. (n.d.). ResearchGate.** [**https://www.researchgate.net/publication/37595346\_Solution\_Methodologies\_for\_the\_Smallest\_Enclosing\_Circle\_Problem**](https://www.researchgate.net/publication/37595346_Solution_Methodologies_for_the_Smallest_Enclosing_Circle_Problem)

***1.3.5 algorithms for Bézier curves*. (n.d.). MIT - Massachusetts Institute of Technology.** [**https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node13.html#:~:text=Continuity%20algorithm%3A%20Bézier%20curves%20can,Bézier%20curves%20end%20to%20end**](https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node13.html#:~:text=Continuity%20algorithm%3A%20Bézier%20curves%20can,Bézier%20curves%20end%20to%20end)

***Bernstein polynomial*. (2003, October 13). Wikipedia, the free encyclopedia. Retrieved November 5, 2020, from** [**https://en.wikipedia.org/wiki/Bernstein\_polynomial**](https://en.wikipedia.org/wiki/Bernstein_polynomial)

***Bézier eğrisi*. (2013, December 26). Vikipedi: Özgür Ansiklopedi. Retrieved November 5, 2020, from** [**https://tr.wikipedia.org/wiki/Bézier\_eğrisi**](https://tr.wikipedia.org/wiki/Bézier_eğrisi)

***C Dilinde Graphics.h Kütüphanesi 1. Bölüm*. (2020, November 1). Fatih Alparslan.** [**https://fatihalparslann.wordpress.com/2017/02/02/c-dilinde-graphics-h-kutuphanesi-1-bolum/**](https://fatihalparslann.wordpress.com/2017/02/02/c-dilinde-graphics-h-kutuphanesi-1-bolum/)

***C/C++ Dilinde Graphics.h Kütüphanesi 2. Bölüm*. (2020, October 26). Fatih Alparslan.** [**https://fatihalparslann.wordpress.com/2020/10/24/c-c-dilinde-graphics-h-kutuphanesi-2-bolum/**](https://fatihalparslann.wordpress.com/2020/10/24/c-c-dilinde-graphics-h-kutuphanesi-2-bolum/)

***C'de grafik fonksiyonları*. (2015, 20). KODEGON - Kod Dünyası!.** [**https://kodegon.blogspot.com/2015/01/cde-grafik-fonksiyonlar.html**](https://kodegon.blogspot.com/2015/01/cde-grafik-fonksiyonlar.html)

**Guidev. (2018, March 22). YouTube.** [**https://www.youtube.com/watch?v=pnYccz1Ha34**](https://www.youtube.com/watch?v=pnYccz1Ha34)

**Mathematics of Computer Graphics and Virtual Environments. (2015, February 9). YouTube.** [**https://www.youtube.com/watch?v=2HvH9cmHbG4**](https://www.youtube.com/watch?v=2HvH9cmHbG4)

***Minimum enclosing circle | Set 1*. (2020, March 23). GeeksforGeeks.** [**https://www.geeksforgeeks.org/minimum-enclosing-circle-set-1/**](https://www.geeksforgeeks.org/minimum-enclosing-circle-set-1/)

**CodeWar. (2020, October 3). YouTube.** [**https://www.youtube.com/watch?v=VEkAj-xVTKQ**](https://www.youtube.com/watch?v=VEkAj-xVTKQ)

***Graphics (graphics.h) - C programming*. (2018, June 24). Developer Insider.** [**https://developerinsider.co/graphics-graphics-h-c-programming/**](https://developerinsider.co/graphics-graphics-h-c-programming/)

***Graphics.h Kütüphanesinin Fonksiyonları*. (n.d.). erdi yurdakul.** [**https://www.erdiyurdakul.com/graphics-h-kutuphanesinin-fonksiyonlari/**](https://www.erdiyurdakul.com/graphics-h-kutuphanesinin-fonksiyonlari/)

**Tdm-gcc. (n.d.). *Tdm-gcc*. Site not found · GitHub Pages.** [**https://jmeubank.github.io/tdm-gcc/**](https://jmeubank.github.io/tdm-gcc/)

**VCR Games. (2016, April 15). YouTube.** [**https://www.youtube.com/playlist?list=PL5UFsTza4wWSNhe0xuO6ELw7ORU-UHNDO**](https://www.youtube.com/playlist?list=PL5UFsTza4wWSNhe0xuO6ELw7ORU-UHNDO)