**Kocaeli Üniversitesi**

**Bilgisayar Mühendisliği Bölümü**

**Programlama Laboratuvarı I**

*Tunay Baştürk – Umut Kılıç*

[*190201032@kocaeli.edu.tr-190201028@kocaeli.edu.tr*](mailto:190201032@kocaeli.edu.tr-190201028@kocaeli.edu.tr)

**Projenin Özeti:**

*Programlama Laboratuvarı I projesi olarak bizden (minimum enclosing circle) minimum çevreleyen çember ve b-spline yapmamız istendi.*

*Biz proje için C programlama dili ve CodeBlocks geliştirme ortamını seçtik.*

*Projede biz OpenGL grafik uygulama geliştirme birimini kullandık. Projede bizden Alegro,Graphic.h ve OpenGL birinden yararlanmamız istendi. Biz OpenGL’de bulunan OpenGL ,Glut Utililty toolkitini kullanmaya karar verdik.*

*CodeBlocks’ta Glut kütüphanesi bulunmadığı için internetten araştırmamız sonucunda Glut Utililty toolkitini aktifleştirmek için internetten freeglut adlı dll’yi indirerek CodeBlocks’a dahil ettik. Böylece Glut Utililty toolkiti aktif oldu.*

*OpenGL’de hazır bulunan fonksiyonları kullanarak program arayüzünün rengini ve boyutunu belirledik. Daha sonra OpenGL’deki kütüphaneleri kullanarak koordinat sistemini çizdirdik. Aynı şekilde OpenGL’deki kütüphaneleri kullanarak minimum çevreleyen çemberi arayüze çizdirdik.*

*Daha sonra OpenGL’deki kütüphaneleri kullanarak ve B-Spline metodundaki matematiksel formüllerden yararlanarak B-Spline eğrisini arayüze çizdirdik.*

**1.GİRİŞ:**

Biz proje için C programlama dili ve CodeBlocks geliştirme ortamını kullandık.

C programlama dili; açık kodlu , taşınabilir , hızlı, statik olarak yazılmış bir dildir. Sonuç olarak C eski olmasına rağmen, sistem programlamasından fotoğraf düzenleme yazılımlarına kadar birçok uygulamaya kadar kullanılır. C programlamanın kullanıldığı bazı popüler uygulamalar :

İşletim Sistemi - Windows , Linux ,Android ,IOS

Veri tabanları – Oracle , MySQL

Diğer kullanım alanları – Ağ sürücüleri , Derleyiciler

CodeBlocks platformu; özgür açık kaynak kodlu bir C geliştirme ortamıdır. wxWidgets tabanlı özelleştirilebilir arabirimiyle , Linux , Windows , MacOS gibi platformlarda sorunsuzca çalışabilir. Bizim tercih etmemizin sebebi 1. Sınıfta C programlama dilini öğrenirken bu IDE’yi kullandığımızdan aşina olmamızdır ve ücretsiz olmasıdır.

**2.YÖNTEM:**

**A.Minimum Çevreleyen Çember:**

Dosyadan aldığımız koordinatları ilk olarak struct’a atadık. Struct’a atadığımız koordinatların uzaklıklarını tek tek karşılaştırıp en uzak 2 noktayı aldık . Böylece çemberin üstünde olacak 2 noktamızı bulmuş olduk. Bu iki noktayı başka bir struct’a atadık. Daha sonra çemberin üstünde 3. bir nokta var mı diye kontrol etmek için en uzak 2 noktamızın orta noktasıyla diğer koordinatlara olan uzaklığını en uzak 2 noktamızın uzaklığının yarısı ile karşılaştırdık . Eğer 2 noktamızın uzaklığının yarısından büyük ise 3. Noktamız en uzak 2 noktamızın orta noktasına en uzak olan koordinatımız olucaktır. Bu 3 üç noktanın oluşturduğu üçgenin çevrel çemberinin merkezi en küçük

çemberimizin merkezi olucaktır. Eğer 2 noktamızın uzaklığının yarısından küçük ise bu 2 noktamızın orta noktası en küçük çemberimizin merkezidir. Merkezimizin, çemberimizin üstünde olan herhangi bir noktaya uzaklığı yarıçapımızdır.

**B*.* B-Spline:**

Dosyadan aldığımız noktaları 3’ten fazla olmak koşuluyla geçici bir 4 nokta alan structa atadık. Geçici structın ilk değerlerine[x,y] 0 atadık. Bir döngü içine alarak noktalarımızı sağdan sola olacak şekilde kaydırdık ve bu 4 nokta alan structımızın içine atadık. Daha sonra ise B-Spline eğrisinin 4 nokta için olan formülünü algoritmamıza aktardık . Formüldeki u (0<=u<=1) değişkenini t olarak aktardık . t’yi 1/1000 arttırarak kontrol nokta sayısını arttırdık . Böylece daha fazla nokta çizdirmiş olduk. Splinex ve spliney adında float tipindeki değişkenlerimizi bu bulduğumuz noktaların koordinatlarını formül yardımıyla aktararak eğrimizi çizdirmiş olduk.

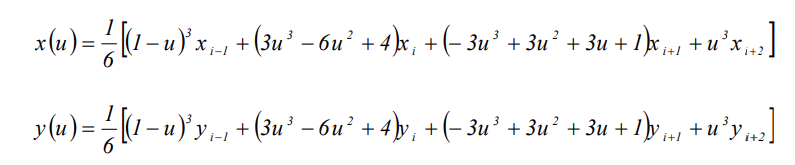
***Matematik Detayları:***



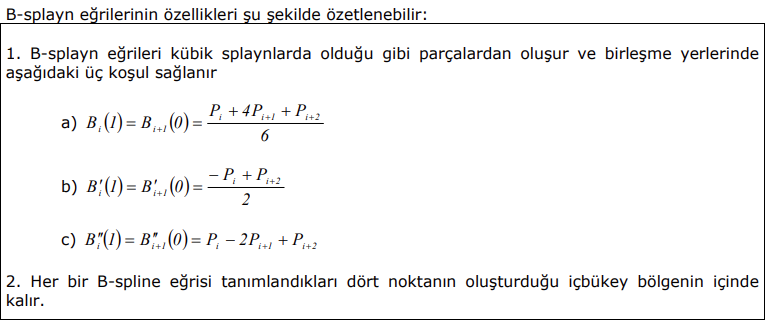
Buradaki  katsayıları dört noktanın etkisi için ağırlık faktörleri olup



şeklinde tanımlanmaktadır. B(i) yerine sırasıyla x ve y, P(i) yerine de x(i) ve y(i) konularak eğri üzerindeki noktaların koordinatları:

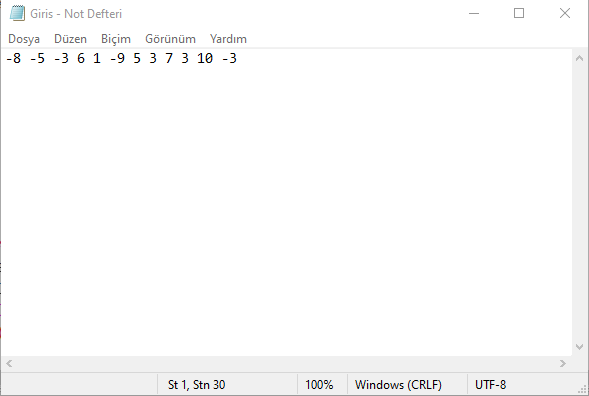


şeklinde elde edilecektir. Noktaların ağırlık faktörleri u parametresi ile değişmekte olup u=0 için sırasıyla 1/6, 2/3, 1/6, 0 ve u=1 için de 0,1/6, 2/3, 1/6 dır.

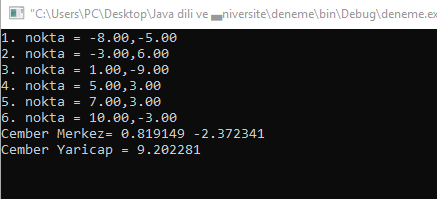


**3.DENEYSEL SONUÇLAR:**

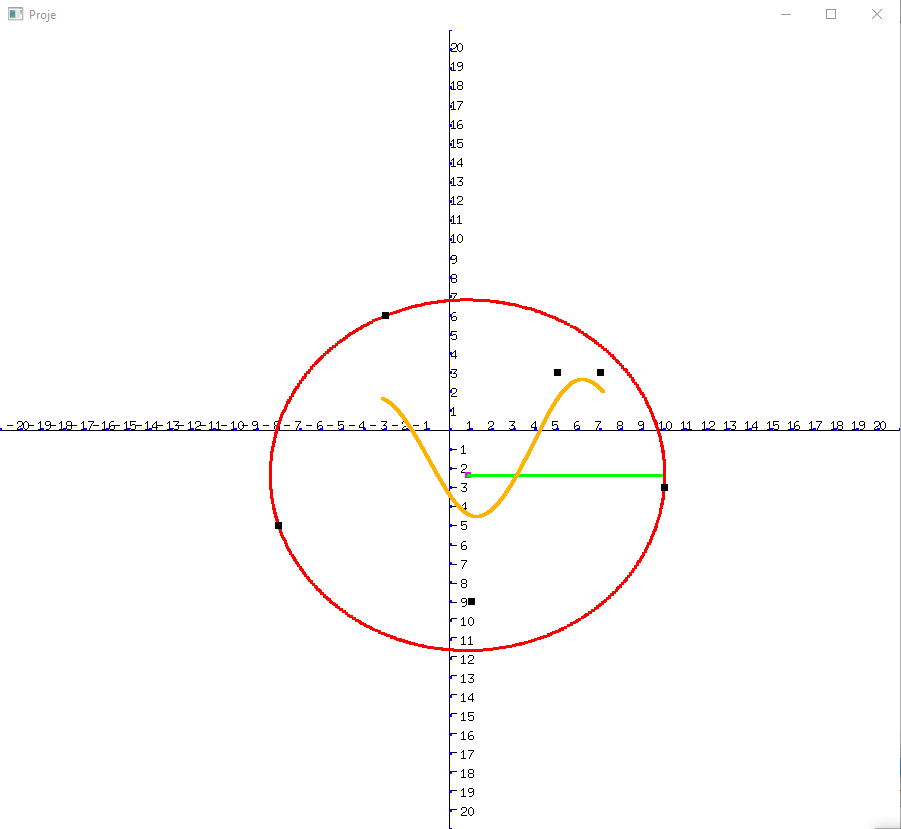
***TXT’den Okutma:***



**Konsol Çıktısı:**



**Arayüz Çıktısı:**



**Zaman Karmaşıklığı:**

***En küçük Çember***

**1-)**

Kullanıcı m adet nokta girdiğinde bu döngü m kez döner. Her dönüşte 1 okuma,3 atama işlemi yapıldığından, toplam 4 x m işlem yapar. Döngüden de 2m+2 işlem yapacağından toplam T(m)=6m+2 işlem yapmış olur zaman karmaşıklığı ise O(m)= m olur

**2-)**

Kullanıcı m adet nokta girer döngü m kez döner . Her dönüşte 1 işlem yapacağından m kez işlem yapmış olur. Döngüde de 2m+2 işlem yaptığından toplam T(m)=3m+2 işlem yapmış olur. zaman karmaşıklığı ise O(m)= m olur.

**3-)**

En dıştaki döngü m defa döner .2m+2 iş yapılır. İçteki döngü (m2-m)/2 defa döner ve m2-m+2 defa iş yapar içerdeki döngülerin başında 5 x ((m2-m)/2) defa atama yapar.

İf durumuna da her durumda gireceğini düşünürsek (m2-m)/2 iş yapar. İf durumunun içindeki atamaların işleminin toplamı da 5 x ((m2-m)/2) olur.

İçerideki döngünün bütün işlemlerini toplarsak 11 x ((m2m)/2) kadar işlem olur. Döngülerin atam ve karşılaştırma işlemleri ile toplarsak

T(m)=11 x ((m2-m)/2)+m2+m+4 olur bu ifadenin de zaman karmaşıklığı O(m)= m2 olur.

**4-)**

Döngüde 2m+2 işlem yapar. Döngünün içerisinde 6m iş yapar toplam T(m)=8m+2 işlem yapılır . Zaman karmaşıklığı ise O(m)=m.

***B SPLİNE***

**1-)**

Dışarıdaki döngü m-3 defa döndüğünden her dönüşte 8 atama yapacağından 8(m-3)atama yapar. Döngüde 2x(m-3)+2 adet atama yapar. İçerideki döngüye gelirsek m’ye bağlı olmadığından her durumda 1000 defa döner. İçeride 2 adet atama olduğundan toplam 2000 atama olur ve bu döngü dışarıdaki döngüden etkilenerek m-3 defa döneceğinden toplam 2000x(m-3) atama işlemi olur. içerideki döngü toplam 2002 adet karşılaştırma ve atama işlemi yapar. Toplam T(m)=2010m-4026 kadar işlem yapar. bunun da zaman karmaşıklığı O(m)=m olur.

**4-KABA KOD**

float MerkezCemberX(struct Cember\_ust Koordinatlar[])

{

1- başla

2- x1 tanımla ve x1’ e (Koordinatlar[1].x+Koordinatlar[0].x)/2.0 degerini ata

3- y1 tanımla ve y1 e(Koordinatlar[1].x+Koordinatlar[0].x)/2.0 degerini ata

4- dy1 tanımla ve dy1 e Koordinatlar[1].x-Koordinatlar[0].x degerini ata

5- dx1 tanımla ve dx1 e -(Koordinatlar[1].y-Koordinatlar[0].y) degerini ata

6-bitir

1- başla

2- x2 tanımla ve değşkene (Koordinatlar[2].x+Koordinatlar[1].x)/2.0 degerini ata

3- y2 tanımla ve y2ye (Koordinatlar[2].y+Koordinatlar[1].y)/2.0 degerini ata

4- dy2 tanımla ve dy2ye Koordinatlar[2].x-Koordinatlar[1].x degerini ata

5- dx2 tanımla ce dx2ye = -(Koordinatlar[2].y-Koordinatlar[1].y) degerini ata

6-bitir

1- başla

2- cx tanımla ve cx degişkenine (y1\*dx1\*dx2+x2\*dx1\*dy2-x1\*dy1\*dx2-y2\*dx1\*dx2)/(dx1\*dy2-dy1\*dx2)degerini ata

3- cy tanımla ve xy degişkenine (cx-x1)\*dy1/dx1+y1 degerini ata

4- bitir

1- başla

2- return cx (cx deĞerinin daha sonra kullanmak üzere döndürürüz.)

3- bitir

}

float MerkezCemberY(struct Cember\_ust Koordinatlar[])

{

1- başla

2- x1 tanımla ve x1 e (Koordinatlar[1].x+Koordinatlar[0].x)/2.0 değerini ata

3- y1 tanımla ve y1 e(Koordinatlar[1].x+Koordinatlar[0].x)/2.0 değerini ata

4- dy1 tanımla ve dy1 e Koordinatlar[1].x-Koordinatlar[0].x değerini ata

5- dx1 tanımla ve dx1 e -(Koordinatlar[1].y-Koordinatlar[0].y) değerini ata

6-bitir

1- başla

2- x2 tanımla ve değişkene (Koordinatlar[2].x+Koordinatlar[1].x)/2.0 değerini ata

3- y2 tanımla ve y2ye (Koordinatlar[2].y+Koordinatlar[1].y)/2.0 değerini ata

4- dy2 tanımla ve dy2ye Koordinatlar[2].x-Koordinatlar[1].x değerini ata

5- dx2 tanımla ce dx2ye = -(Koordinatlar[2].y-Koordinatlar[1].y) değerini ata

6-bitir

1- başla

2- cx tanımla ve cx değişkenine (y1\*dx1\*dx2+x2\*dx1\*dy2-x1\*dy1\*dx2-y2\*dx1\*dx2)/(dx1\*dy2-dy1\*dx2)degerini ata

3- cy tanımla ve xy degişkenine (cx-x1)\*dy1/dx1+y1 değerini ata

4- bitir

1- başla

2- return cy (cy değerinin daha sonra kullanmak üzere döndürürüz.)

3- bitir

}

void display()

1- başla

2- i tanımla

3- j yi tanımla ve 9 değerini ata

4- x,y,c tanımla

5- struct (tutucu geçici[4]) tanımla

6- n,k,kontrol tanımla ve kontrol değişkenine 0 ata

7- x1,x2,y1,y2,yaricap tanımla

8- Enuzak2nokta[4] tanımla

9- uzaklik,temp,Mx,My değişkenlerini tanımla ve temp değişkenine 0 ata

10-FILE\*dosya=fopen("C:\\Users\\PC\\Desktop\\Giris2.txt","r") tanımla ve giris text’ini okut

11- q tanımla ve 0 ata

12- okunan\_x tanımla

13- okunan\_y tanımla

14- okunan\_x tanımla

15- okunan\_y tanımla

16- t tanımla;

17- struct Girilen\_noktalar Giris[1000];

18- bitiş

1- başla

2- for(int j=0;!feof(dosya);j++){

3- dosyadan iki noktayı oku.Bunları okunan x ve okunan y ye ata.

4- giriş structının x ine okunan x i y sine okunan y’yi ata

5- q yı bir artır

6- döngü biter

7- bitir

1-başla

2- m i tanımla m’ye q yu ata

3- for(q=0;q<m;q++)

4- printf girilen noktaları yaz

5- döngü biter

6- dosyayı kapat

7- bitir

1- başla

2- struct Cember\_ust Koordinatlar[m] tanımla;

3- float Enuzak2\_noktanin\_merkezinin\_diger\_noktalara\_uzakligi[m] tanımla;

4- eğer m 1 den büyük ise bu işleme gir

5- for(i 0 dan m e kadar don){

6- for(int k=i+1; k<m; k++){

7- uzaklık =giriş structının x’lerinin farkının karesi ile y’lilerin farkının karelerin toplamının karekökünü al

8- x1 e Giris[i].x ata

9- x2 e Giris[k].x ata

10- y1 e Giris[i].y ata

11- y2 e Giris[k].y ata

12- eğer uzaklık tempten büyük ise gir

temp e uzaklik’ği ata

Enuzak2nokta[0] ya x1 ata

Enuzak2nokta[1]ya y1 ata

Enuzak2nokta[2] ya x2 ata

Enuzak2nokta[3] ya y2 ata

13- ifden çıkar

14- içteki döngü sona erer

15- sondaki döngü sona erer

16- bitir

1- başla

2- Koordinatlar[0].x=Enuzak2nokta[0] ata

3- Koordinatlar[0].y=Enuzak2nokta[1] ata

4- Koordinatlar[1].x=Enuzak2nokta[2] ata

5- Koordinatlar[1].y=Enuzak2nokta[3] ata

6- Mx=(Enuzak2nokta[0]+Enuzak2nokta[2])/2.0 ata

7- My=(Enuzak2nokta[1]+Enuzak2nokta[3])/2.0 ata

8- float Merkezin\_uzak\_2\_noktaya\_uzakligi=sqrt(pow((Mx-Enuzak2nokta[0]),2)+pow((My-Enuzak2nokta[1]),2)) tanımla ve merkezin x i ile enuzak[0] noktanın farkını ve yine aynı sekile y’lerin farkının karesini alıp karekökünü alıp bu tanımladığımız değere atadık

9- float yeninokta[2] ata

10- for(int i=0; i<m; i++)

11- Enuzak2\_noktanin\_merkezinin\_diger\_noktalara\_uzakligi[i]=sqrt(pow((Mx-Giris[i].x),2)+pow((My-Giris[i].y),2)) bu değeri değişkenimize tanımladık

12- eğer Enuzak2\_noktanin\_merkezinin\_diger\_noktalara\_uzakligi[i] Merkezin\_uzak\_2\_noktaya\_uzakligi`ndan büyük ise gir

Merkezin\_uzak\_2\_noktaya\_uzakligi=Enuzak2\_noktanin\_merkezinin\_diger\_noktalara\_uzakligi[i] ata

yeninokta[0]=Giris[i].x ata

yeninokta[1]=Giris[i].y ata

kontrol değişkeninin birer kez artır

13- if durumu biter

14- döngü biter

15- bitir

1- başla

2-eger m birden büyük ise gir

3-eger kontrol sıfıra eşit değil ise gir

4- Koordinatlar[2].x=yeninokta[0] ata

5- Koordinatlar[2].y=yeninokta[1] ata

6- Mx=MerkezCemberX(Koordinatlar) ata

7- My=MerkezCemberY(Koordinatlar) ata

8- yaricap=sqrt(pow((Mx-Koordinatlar[0].x),2)+pow((My-Koordinatlar[0].y),2)) ata

9- konsola çemberin merkezinin noktalarını yazdır

10- konsola çemberin yarıçapı yazdır

11-bitir

1- başla

2- glColor3f ile renk tanımlaması yapılır

3- glBegin(3.0) ile noktaların boyutu belirlenir

4- for(c = 0.5; c < 360; c+=0.5)

5- x = yaricap \* sin(((double)c)) x degişkenine sağdaki ifadeyi çarpıp atarız.

6- y = yaricap \* cos(((double)c)) y degişkenine sağdaki ifadeyi çarpıp atarız.

7- glvertex2f(x+Mx,y+My) noktaların yazılacağı yeri verip çizdiririz

8- döngüden çıkar

9- glEnd() ile çizimi bitiririz.

10- bitir.

**B SPLİNE**

1- başla

2- gecici[0].x=0 değişkenine 0 ata

3- gecici[0].y=0 değişkenine 0 ata

4- gecici[1].x=Giris[0].x ata

5- gecici[1].y=Giris[0].y ata

6- gecici[2].x=Giris[1].x ata

7- gecici[2].y=Giris[1].y ata

8- gecici[3].x=Giris[2].x ata

9- gecici[3].y=Giris[2].y ata

10- float splinex tanımla

11- float spliney tanımla

12- glColor3f ile renk tanımlaması yapılır

13- glBegin(3.0) ile noktaların boyutu belirlenir

14- int a=0 ile tanımlar ve 0 atarız.

15- for(i=3;i<m;i++)

16- gecici[0].x=gecici[1].x ata

17- gecici[0].y=gecici[1].y ata

18- gecici[1].x=gecici[2].x ata

19- gecici[1].y=gecici[2].y ata

20- gecici[2].x=gecici[3].x ata

21- gecici[2].y=gecici[3].y ata

22- gecici[3].x=Giris[i].x ata

23- gecici[3].y=Giris[i].y ata

24- for(t=0;t<1;t=t+0.001){

25- splinex=( (pow((1-t),3)\*gecici[0].x)+ ((3\*pow(t,3)-(6\*(pow(t,2)))+4)\*gecici[1].x) - (((3\*pow(t,3))-(3\*pow(t,2))-(3\*t)-1)\*gecici[2].x) + (pow(t,3)\*gecici[3].x))/6.0 ata

26- spliney=( (pow((1-t),3)\*gecici[0].y)+ ((3\*pow(t,3)-(6\*(pow(t,2)))+4)\*gecici[1].y) - (((3\*pow(t,3))-(3\*pow(t,2))-(3\*t)-1)\*gecici[2].y) + (pow(t,3)\*gecici[3].y))/6.0 ata

27- glBegin ile noktaların çizimine başlar

28- glvertex2f(splinex,spliney) noktaları çizdir

29- glEnd() fonksiyonu ile çizdirmeyi bitir

30- a’yı birer birer artır

31- döngüyü bitir

32- bitir

1-başla

2- glColor3f(0,0,0) noktaların rengini belirleriz

3- glPointSize(7) noktanın boyutunu belirtiriz

4- glBegin(GL\_POINTS) noktaların çizimine başlar

5- for(int i=0;i<m;i++){

6- glVertex2f(Giris[i].x,Giris[i].y) noktaları çizdir

7- döngü bitir

8- glend() cizdirmeyi bitirir

9- bitir

en sonda glFlush() ile arayüze aktarırız.

**SONUÇ:**

Bu proje sonucunda çember algoritmasını , B-Spline algoritmasını anladık . Takım halinde çalışmayı , azimli olmayı , araştırma yapmayı , sabırlı olmayı öğrendik . Çember ve B-Splin’ı arayüze aktarırken Glut Kütüphanesindeki çeşitli fonksiyonlardan yararlanarak çizdirmeyi öğrendik.

**KAYNAKÇA:**

[1] https://www.youtube.com/watch?v=3aJ8OR1C6pk&t=778s&ab\_channel=ThePentamollisProject

[2] http://www.belgeler.org/howto/opengl-giris\_islevler.html

[3] https://en.wikipedia.org/wiki/RGBA\_color\_model

[4] https://tr.wikipedia.org/wiki/Code::Blocks

[5] https://www.cdersleri.com/c-programlama-dilinin-ozellikleri

[6] https://web.itu.edu.tr/yukselen/HM504/02Ek-%20Bezier%20e%f0rileri.pdf

[7] https://www.geeksforgeeks.org/minimum-enclosing-circle-set-1/

[8] https://en.wikipedia.org/wiki/Smallest-circle\_problem

[9] https://www.cs.mcgill.ca/~cs507/projects/1998/jacob/problem.html

[10] https://www.elektrikport.com/teknik-kutuphane/yalanci-kod-nedir/14925#ad-image-0

[11] http://bilgisayarkavramlari.com/2009/08/10/splines-seritler/?highlight=b%20spline

[12] http://people.cs.vt.edu/yongcao/teaching/cs4204/fall2008/materials/documents/ParametricOpenGL\_2pp.pdf

[13] https://www.ft.unicamp.br/~magic/redbook/chapter11.html