Muhammed Halil Akkaynak
190202075
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
halilakkaynak71@gmail.com

MİNİMUM ÇEVRELEYEN ÇEMBER

B-SPLINE

Erce Uslu
190202083
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
erceonline@gmail.com

Özet - Bu proje, verilen koordinatlar ile çizilebilecek en küçük çemberi çizip ve bu noktalar veya yakınından B-Spline eğrisi çizme üzerine bir çalışmadır.

Anahtar Kelimeler - B-Spline, Koordinat, Minimum Çevreleyen Çember.

I. GİRİŞ

Bu proje temelde projeyi yapan kişilerin koordinat düzlemi ve grafik kütüphaneleriyle oluşturalabilecek algoritmalarını geliştirmeyi amaçlamıştır.

Bu projeye başladığımız anda koordinat sistemini ekrana nasıl bastırabileceğimizi düşündük ve bu işlemi “graphics.h” kütüphanesi ile yapabileceğemize karar verdik. Ardından biraz araştırma sonucu koordinat sistemini ekrana bastırmayı başardık. Akabinde verilen koordinatlarımızı .txt dosyasından projemize tarattık ve koordinat sistemimize bastırdık. Verilen noktaları kapsayan, çizilebilecek en küçük çemberi bulmamıza yarayan matematiksel formül üzerinde çalıştık, ve çemberimizi başarılı bir şekilde koordinat sistemimize entegre ettik.B-Spline çizerken bir problemle karşılaştık.(Karşılaşılan Problemler’ de yer verilmiştir.) B-Spline’ımızı yay ve eğim kullanarak çizdik ve koordinat sistemimize bastırdık.

II. KARŞILAŞILAN PROBLEMLER

İlk karşılaştığımız problem çemberin merkezini bulmaya yarayan matematiksel formülü oluşturmaktı.Bu problemi biraz zaman alsa da araştırmalarımız sonucunda çözdük.

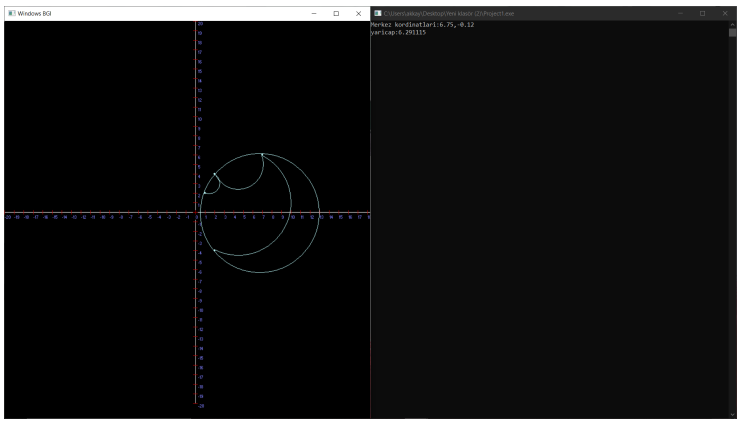
İkinci karşılaştığımız problem ise B-Spline çiziminde oldu.B-Spline çizimini yaparken son iki nokta arası yay çizimini yapamıyorduk.Bu problemi de bir süre çalışmadan sonra çözdük.

III. GENEL YAPI

KULLANICI KISMI

Proje başlatıldığında ekrana merkezin koordinatları ve çemberin yarıçapının uzunluğu ekrana bastırılır ve yeni bir pencerede graphics.h kütüphanesi yardımıyla ekrana koordinat sistemimiz, minimum kapsayan çemberimiz ve B-Spline’ımız ekrana çizilir.

Ekran çıktısının görseli yan sayfada verilmiştir.



Ek-Resim1.PNG

KOD KISMI

İlk önce main fonksiyonu içerisindeki tüm algoritmaları çalıştaracak olan fonksiyonları görüyoruz ve .txt uzantılı Koordinatlar dosyasından verilen koordinatları alıyoruz.

```
int main(int argc, char** argv) {
    initwindow(PENCERE_BOYUTU,PENCERE_BOYUTU,"");
    int kordinatSayisi=0;
    FILE *file;
    file=fopen("Kordinatlar.txt","rt");
    char saydirma[100];//Kordinat sayisini bulmak
    int i=0;
    if(file!=NULL)
    {
        while(fgets(saydirma,100,file) != NULL)
        {
            kordinatSayisi++;
        }
        fclose(file);
        file=fopen("Kordinatlar.txt","r");
        char sekiller[kordinatSayisi][4];//Parantez ve virgulleri ayirmak icin
        int kordinatlar[kordinatSayisi][2];
        while(!feof(file))
        {
            fscanf(file,"%c%d%c%d%c%c",&sekiller[i][0],&kordinatlar[i][0],&sekiller[i][1],&kordinatlar[i][1],&sekiller[i][2],&sekiller[i][3]);
            i++;
        }
    kordinantSistemi();//Koordinat sistemini ekrana cizen fonksiyondur
    noktalarikordinatSistemineKoyma(kordinatlar,kordinatSayisi);//Koordinat sisteminine noktalari cizen fonksiyondur
    cemberiOlusturma(kordinatlar,kordinatSayisi);//Koordinat sisteminine cemberi cizen fonksiyondur
    kordinatlariSiralama(kordinatlar,kordinatSayisi);//Koordinatlari ilk noktanin yakinlik derecesine gore siralayan fonksiyondur
    }
    else
    {
        printf("Dosya bulunamadi");
    }
    getch();
    closegraph();
    fclose(file);
    return 0;
}
```

Ek-Resim2.PNG

Eğer iki tane koordinat verildiyse diğer durumlarda merkezi 3 nokta ile bulan bir algoritma yazdığımız için bu koşulu kullanıyoruz. Bu koşulda verilen iki koordinatın orta noktasını merkez kabul edip, bu noktaların da merkeze olan uzaklığını yarıçap kabul edip çemberimizi çizdiriyoruz.

```
double merkezX,merkezY,yaricap;
if(kordinatSayisi==2)
{
    merkezX=(kordinatlar[0][0]+kordinatlar[1][0])/2;
    merkezY=(kordinatlar[0][1]+kordinatlar[1][1])/2;
    yaricap=sqrt(pow(kordinatlar[1][0]-merkezX,2)+pow(kordinatlar[1][1]-merkezY,2));
    circle((merkezX*BUYULTME_OLCUTU)+PENCERE_BOYUTU/2,PENCERE_BOYUTU/2-(merkezY*BUYULTME_OLCUTU),yaricap*BUYULTME_OLCUTU);
}
```

Ek-Resim3.PNG

Diğer sayfadaki kod parçasında iki nokta arası uzaklık formülünden birbirine en uzak olan iki noktayı bulduk.

```
else
{
double temp=0.0; //En uzak 2 noktanin koordinatlarini bulmak için gecici deger
double enUzakIkiNokta;//En uzak 2 nokta mesafesidir
for (int i=0;i<kordinatSayisi;i++)
{
    for (int j=0;j<kordinatSayisi;j++)
    {
        if(j!=i){
            enUzakIkiNokta = sqrt((pow((kordinatlar[i][0]-kordinatlar[j][0]),2))+(pow(kordinatlar[i][1]-kordinatlar[j][1],2)));//2 noktanin arasindaki mesafenin matematiksel formuludur
            if (enUzakIkiNokta>temp)
            {
                temp=enUzakIkiNokta;
            }
        }
    }
}
int ilkNokta=0,IkinciNokta=0,ucuncuNokta=0;//Birbirine en uzak 3 noktayi secmek için tanimlanan degiskenler
double control;
//Cemberin uzerindeki ilk ve ikinci nokta:
for (ilkNokta=0;ilkNokta<kordinatSayisi;ilkNokta++)
{
    for(IkinciNokta=0;IkinciNokta<kordinatSayisi;IkinciNokta++)
    {
        control = sqrt((pow((kordinatlar[ilkNokta][0]-kordinatlar[IkinciNokta][0]),2))+(pow(kordinatlar[ilkNokta][1]-kordinatlar[IkinciNokta][1],2)));
        if(control==temp)
        {
            break;
        }
    }
    if(control==temp)
    {
        break;
    }
}
```

Ek-Resim4.PNG

İki nokta uzaklık formülünden ikinci noktaya en uzak üçüncü noktayı bulduk.

```
//Cemberin uzerindeki ucuncu nokta:
for(ucuncuNokta=0;ucuncuNokta<kordinatSayisi;ucuncuNokta++)
{
    if(ucuncuNokta!=IkinciNokta && ucuncuNokta!=ilkNokta)
    {
        enUzakIkiNokta = sqrt((pow((kordinatlar[IkinciNokta][0]-kordinatlar[ucuncuNokta][0]),2))+(pow(kordinatlar[IkinciNokta][1]-kordinatlar[ucuncuNokta][1],2)));
        if (enUzakIkiNokta>temp)
            {
                temp=enUzakIkiNokta;
            }
    }
}
for (ucuncuNokta=0;ucuncuNokta<kordinatSayisi;ucuncuNokta++)
{
    control = sqrt((pow((kordinatlar[IkinciNokta][0]-kordinatlar[ucuncuNokta][0]),2))+(pow(kordinatlar[IkinciNokta][1]-kordinatlar[ucuncuNokta][1],2)));
    if(control==temp)
    {
        break;
    }
}
```

Ek-Resim5.PNG

Çemberin merkezini bulmak için çevrel çember merkezinin kartezyen koordinatları formülünden yararlanarak bulduk ve yarıçapı da oluşturduğumuz yarıçap fonksiyonu ile bulduk. Çemberi çizdirdik, merkezini ve yarıçapını ekrana bastırdık.

```
,
double dHesaplama1=kordinatlar[ilkNokta][0]*(kordinatlar[IkinciNokta][1]-kordinatlar[ucuncuNokta][1]);
double dHesaplama2=kordinatlar[IkinciNokta][0]*(kordinatlar[ucuncuNokta][1]-kordinatlar[ilkNokta][1]);
double dHesaplama3=kordinatlar[ucuncuNokta][0]*(kordinatlar[ilkNokta][1]-kordinatlar[IkinciNokta][1]);
double d=2*(dHesaplama1+dHesaplama2+dHesaplama3);//Merkez koordinatlarini bulmak için gerekli hesaplama -> ustekt 3 islemde adim adim yaptik.
double merkezXHesaplama1=(pow(kordinatlar[ilkNokta][0],2)+pow(kordinatlar[ilkNokta][0],2))*(kordinatlar[IkinciNokta][1]-kordinatlar[ucuncuNokta][1]);
double merkezXHesaplama2=(pow(kordinatlar[IkinciNokta][1],2)+pow(kordinatlar[IkinciNokta][0],2))*(kordinatlar[ucuncuNokta][1]-kordinatlar[ilkNokta][1]);
double merkezXHesaplama3=(pow(kordinatlar[ucuncuNokta][1],2)+pow(kordinatlar[ucuncuNokta][0],2))*(kordinatlar[ilkNokta][1]-kordinatlar[IkinciNokta][1]);
merkezX=(merkezXHesaplama1+merkezXHesaplama2+merkezXHesaplama3)/d;//Merkezin x koordinatini bulmak için gerekli hesaplama -> ustekt 3 islemde adim adim yaptik.
double merkezYHesaplama1=(pow(kordinatlar[ilkNokta][1],2)+pow(kordinatlar[ilkNokta][0],2))*(kordinatlar[ucuncuNokta][0]-kordinatlar[IkinciNokta][0]);
double merkezYHesaplama2=(pow(kordinatlar[IkinciNokta][1],2)+pow(kordinatlar[IkinciNokta][0],2))*(kordinatlar[ilkNokta][0]-kordinatlar[ucuncuNokta][0]);
double merkezYHesaplama3=(pow(kordinatlar[ucuncuNokta][1],2)+pow(kordinatlar[ucuncuNokta][0],2))*(kordinatlar[IkinciNokta][0]-kordinatlar[ilkNokta][0]);
merkezY=(merkezYHesaplama1+merkezYHesaplama2+merkezYHesaplama3)/d;//Merkezin y koordinatini bulmak için gerekli hesaplama -> ustekt 3 islemde adim adim yaptik.
yaricap=yariCap(kordinatlar,kordinatSayisi,merkezX,merkezY);
circle(merkezX*BUYULTME_OLCUTU)+PENCERE_BOYUTU/2,PENCERE_BOYUTU/2-(merkezY*BUYULTME_OLCUTU),yaricap*BUYULTME_OLCUTU);
}
printf("Merkez koordinatlari:%.2f,%.2f\n",merkezX,merkezY);
printf("yaricap:%f",yaricap);
}
```

Ek-Resim6.PNG

Aşağıdaki fonksiyonda koordinat sistemini, koordinat sisteminin üzerindeki çizgileri ve sayıları ekrana çizdirdik.

```
void kordinantSistemi(){
    setcolor(15);
    line(0,PENCERE_BOYUTU/2,PENCERE_BOYUTU,PENCERE_BOYUTU/2); //x koordinat duzlemi
    line(PENCERE_BOYUTU/2,0,PENCERE_BOYUTU/2,PENCERE_BOYUTU); //y koordinat duzlemi
    int xkordinatSayilari=-20;
    int ykordinatSayilari=20;
    char sNum[41];
    for (int i=0;i<41;i++)
    {
        setcolor(4);
        line(i*BUYULTME_OLCUTU,(PENCERE_BOYUTU/2)+5,i*BUYULTME_OLCUTU,(PENCERE_BOYUTU/2)-5); //x koordinat duzlemindeki çizgiler
        line((PENCERE_BOYUTU/2)-5,i*BUYULTME_OLCUTU,(PENCERE_BOYUTU/2)+5,i*BUYULTME_OLCUTU); //y koordinat duzlemindeki çizgiler
        settextstyle(2,0,4);
        setcolor(9);
        itoa(xkordinatSayilari,sNum,10);//sayilari integerden char tipine cevirmek için
        outtextxy(i*BUYULTME_OLCUTU,(PENCERE_BOYUTU/2)+5,sNum); //x kooridnat duzlemindeki numaralar
        if(xkordinatSayilari!=0)
        {
            itoa(ykordinatSayilari,sNum,10);
            outtextxy((PENCERE_BOYUTU/2)+5,i*BUYULTME_OLCUTU,sNum); //y kooridnat duzlemindeki numaralar
        }
        ykordinatSayilari--;
        xkordinatSayilari++;
    }
}
```

Ek-Resim7.PNG

Noktaları graphics.h fillelipse fonksiyonu yardımıyla ekrana bastırdık.

```
void noktalariKordinatSistemineKoyma(int kordinatlar[][2],int kordinatSayisi){
    setcolor(11);
    setfillstyle(1,11);
    for(int i=0;i<kordinatSayisi;i++)
    {
        fillellipse((kordinatlar[i][0]*BUYULTME_OLCUTU)+PENCERE_BOYUTU/2,PENCERE_BOYUTU/2-(kordinatlar[i][1]*BUYULTME_OLCUTU),2, 2);//Noktalar daha belirgin olmasi amaciyla elips seklinde cizdik
    }
}
```

Ek-Resim8.PNG

Merkeze en uzak nokta ile merkez arası uzaklığı yarıçap olduğundan merkeze en uzak noktayı bulduk ve yarıçapı hesapladık. Bu fonksiyonu çemberi oluşturma fonksiyonu içerisinde çağırdık.

```
double yariCap(int kordinantlar[][2],int kordinatSayisi,double merkezX,double merkezY){
    // yaricap=merkeze en uzak nokta
    double yariCapTemp[kordinatSayisi];//Merkeze en uzak noktayi bulmak için gerekli degisken
    double yariCap;
    for (int i=0;i<kordinatSayisi;i++)
    {
        yariCapTemp[i]=sqrt((pow((kordinantlar[i][0]-merkezX),2))+(pow(kordinantlar[i][1]-merkezY,2)));
    }
    for(int i=0;i<kordinatSayisi-1;i++)
    {
        for(int j=1;j<kordinatSayisi;j++)
        {
            if (yariCapTemp[i]>yariCapTemp[j])
            {
                yariCap=yariCapTemp[i];
                yariCapTemp[i]=yariCapTemp[j];
                yariCapTemp[j]=yariCap;
            }
        }
    }
    return yariCap;
}
```

Ek-Resim9.PNG

Koordinatlar[0][0] noktasını referans noktası olarak seçip diğer noktaların referans noktasına olan uzaklıklarına göre küçükten büyüğe koordinat sıralaması yaptık ve bu fonksiyonda bsplineCizimi fonksiyonunu çağırıyoruz.

```
void kordinatlariSiralama(int kordinatlar[][2],int kordinatSayisi){
    //Koordinatlari siralamak için gerekli olan referans koordinatlariz kullanicidan alinan ilk noktadir
    double uzaklikDizi[kordinatSayisi-1];
    for(int i=1;i<kordinatSayisi;i++)
    {
        uzaklikDizi[i-1]=sqrt((pow(kordinatlar[0][0]-kordinatlar[i][0],2))+pow(kordinatlar[0][1]-kordinatlar[i][1],2));//Noktalarin referans noktasina olan uzakliklarini diziye atadik
    }
    for(int i=0;i<kordinatSayisi-2;i++)
    {
        for(int j=1;j<kordinatSayisi-1;j++)
        {
            if(uzaklikDizi[i]>uzaklikDizi[j])
            {
                double temp=uzaklikDizi[i];
                uzaklikDizi[i]=uzaklikDizi[j];
                uzaklikDizi[j]=temp;
            }
        }
    }
    //Asagidaki 7 satirda koordinatlarin referans noktasina olan uzakligina gore kucukten buyuge siralanmistir
    for(int i=0;i<kordinatSayisi-1;i++)
    {
        double uzaklik=sqrt((pow(kordinatlar[0][0]-kordinatlar[i][0],2))+pow(kordinatlar[0][1]-kordinatlar[i][1],2));
        if(uzaklikDizi[i] == uzaklik )
        {
            kordinatlar[i+1][0]=kordinatlar[i+1][0];
            kordinatlar[i+1][1]=kordinatlar[i+1][1];
        }
    }
    bsplineCizimi(kordinatlar,kordinatSayisi);//Koordinatlari referans noktasina olan uzakligina gore kucukten buyuge seklinde siralandiktan sonra bspliani cizdirdik
}
```

Ek-Resim10.PNG

B-Spline'ı karşılaştığımız problemden dolayı arc ile çizmeye çalıştık. Arc kullanırken yay başlangıç ve bitiş açısına ihtiyacımız olduğundan dolayı eğimi bulduk. Eğimin arctan'ı sayesinde yayın başlangıç radyanını bulduk. Bunu da 57.3 ile çarparak yayın başlangıç açısını bulduk. Başlangıç açısına 180 derece ekleyerek de bitiş açısını bulduk.

```
void bsplineCizimi(int kullanilanKordinatlar[][2],int kordinatSayisi){
    for(int i=0;i<kordinatSayisi-1;i++)
    {
        //Matematiksel egim formulu ile egim hesaplandi
        double egimX=kullanilanKordinatlar[i+1][0]-kullanilanKordinatlar[i][0];
        double egimY=kullanilanKordinatlar[i+1][1]-kullanilanKordinatlar[i][1];
        double egim=egimY/egimX;
        //2 nokta arasi uzaklik formulunden yayin merkezini buldu
        double yayMerkeziXHesapla=(kullanilanKordinatlar[i+1][0]+kullanilanKordinatlar[i][0]);
        double yayMerkeziYHesapla=(kullanilanKordinatlar[i+1][1]+kullanilanKordinatlar[i][1]);
        double yayMerkeziX=yayMerkeziXHesapla/2;
        double yayMerkeziY=yayMerkeziYHesapla/2;
        double yayYaricap=sqrt(pow(kullanilanKordinatlar[i][0]-yayMerkeziX,2)+pow(kullanilanKordinatlar[i][1]-yayMerkeziY,2));
        double yayBaslangicAcisi=atan(egim)*57.3;
        double yayBitisAcisi=180+yayBaslangicAcisi;
        yayMerkeziX=(yayMerkeziX*BUYULTME_OLCUTU)+PENCERE_BOYUTU/2;
        yayMerkeziY=PENCERE_BOYUTU/2-(yayMerkeziY*BUYULTME_OLCUTU);
        arc(yayMerkeziX,yayMerkeziY,yayBitisAcisi,yayBaslangicAcisi,yayYaricap*BUYULTME_OLCUTU);
    }
}
```

Ek-Resim10.PNG

IV. BIG-O ANALİZİ

Projemizde oluşturduğumuz algoritmaların algoritma karmaşıklığını hesapladığımızda iç içe iki döngüden dolayı zaman karmaşıklığı $O(N^2)$ dır.

V. KULLANILAN TEKNOLOJİLER

Projemizde kullandığımız yazılım dili C dir.

Projemizde kullandığımız IDE Dev-C++ dir.

Projemizde kullandığımız grafik kütüphanesi graphics.h dır.

VI. AKIŞ ŞEMASI

(Akış şeması son sayfada yer almaktadır.)

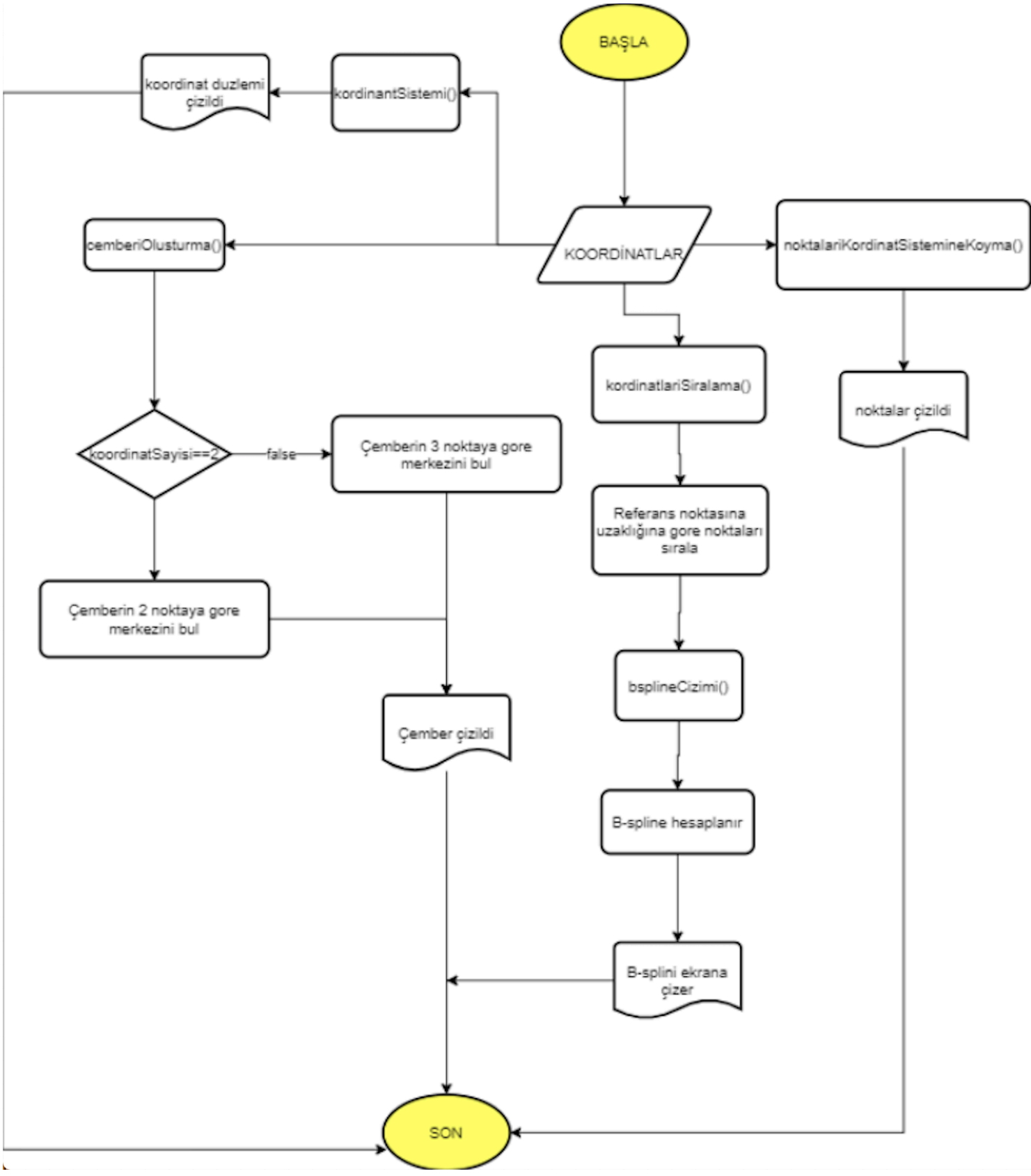
VII. KAYNAKÇA

https://tr.wikipedia.org/wiki/Çevrel_çember

https://web.ogu.edu.tr/Storage/egulbandilar/Uploads/AlgoritmaAnalizi.pdf

https://web.ogu.edu.tr/Storage/egulbandilar/Uploads/AlgoritmaAnalizi.pdf

http://kodegon.blogspot.com/2015/01/cde-grafik-fonksiyonlar.html


BAŞLA
koordinat duzlemi çizildi
kordinantSistemi()
cemberiOlusturma()
KOORDİNATLAR
noktalariKordinatSistemineKoyma()
kordinatlariSiralama()
noktalar çizildi
koordinatSayisi==2
false
Çemberin 3 noktaya gore merkezini bul
Referans noktasına uzaklığına gore noktaları sırala
Çemberin 2 noktaya gore merkezini bul
bsplineCizimi()
Çember çizildi
B-spline hesaplanır
B-splini ekrana çizer
SON