Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

PROGRAMLAMA LAB. I- I. Proje

Minumum Çevreleyen Çember

(Minimum Enclosing Circle(MEC))-B Spline

Oğulcan Kırtay

[190202005@kocaeli.edu.tr](mailto:190202005@kocaeli.edu.tr)

Projenin Özeti:

Programlama laboratuvarı I Projesi olarak bizden “Minimum Çevreleyen Çember (Minimum Enclosing Circle(MEC))-B Spline” adındaki bir uygulama geliştirmemiz beklenmektedir.

Proje dokümanında bizden beklenen tamsayı koordinatlı 2 boyutlu bir düzlemde N nokta verildiğinde, tüm noktaları içeren minimum çevreleyen yarıçaplı daireyi çizdirmemiz, verilen N noktanın en yakınından geçen eğriyi çizdirmemiz ve çizdirmiş olduğum dairenin ise yarıçapını ve merkezini hesaplamam istenmektedir.

Projede ilk olarak bir file dosyası oluşturdum verileri içerisine girdim sonra bu verileri aldım bu değerlerle araştırmış olduğum matematiksel algoritma ile çember merkezini ve yarıçapını hesaplattım daha sonra C programlama dilinde bulunan “Graphics.h” adlı Arayüz tasarım kütüphanesinden yararlandım. Bu kütüphane sayesinde bizden istenilen Verilen tüm noktaların (N tane) tüm noktaları içeren minimum çevreleyen yarıçaplı daireyi ve Verilen tüm noktaların (N tane) en yakınından geçen bir eğrinin çizdirdim.

1.GİRİŞ

Proje için C programlama dili ve CodeBlocks geliştirme ortamını kullandım.

C programlama dili; c orta seviye nesneye yönelik programlama dilidir

CodeBlocks platformu; CodeBlocks özgür açık kaynak kodlu bir  C ve [C++](https://tr.wikipedia.org/wiki/C%2B%2B) [geliştirme ortamıdır](https://tr.wikipedia.org/wiki/T%C3%BCmle%C5%9Fik_geli%C5%9Ftirme_ortam%C4%B1). [GNU/Linux](https://tr.wikipedia.org/wiki/GNU/Linux), [Microsoft](https://tr.wikipedia.org/wiki/Microsoft) [Windows](https://tr.wikipedia.org/wiki/Microsoft_Windows), [MacOS](https://tr.wikipedia.org/wiki/MacOS" \o "MacOS) platformlarında sorunsuzca kullanılabilmektedir. Gelişmiş plugin desteğiyle kod yazımı esnasında ihtiyaç duyabileceğiniz birçok yardımcı fonksiyon sunar ve kod üzerinde tam bir hakimiyet kurmanıza olanak tanır.

2. TEMEL BİLGİLER

Projemde; dosya\_oku , dist, hesapla, cizimyap main olmak üzere 5 adet fonksiyondan oluşmaktadır.

MAİN():

Main fonksiyonu ana fonksiyon olup yan fonksiyonları çalıştıran ana fonksiyondur.

int main()

{

int a;

float x[1000];

float y[1000];

float xk,yk,r;

dosya\_oku(x,y,&a);

hesapla(x,y,&a,&xk,&yk,&r);

cizimyap(&xk,&yk,&r,x,y,&a);

}

Buradaki tanımlamış olduğum a integerını txt tipi dosyadan alacak olduğum koordinat sayısını temsil etmektedir. Float tipi “x” ve “y” arrayleri ise (x,y) koordinatlarını temsil etmektedir. Float tipi “xk” ve “yk” sayılar merkez koordinatları “r” ise yarıçap için tanımlıdır.

dosya\_oku():

Bu fonksiyon göndermiş olduğum x y diziler ve a değerine sırasıyla x koordinatı y koordinatı ve kaç adet nokta girildiği bilgisini txt dosyasından alıp bu değerlerin içerisine almamı sağlıyor.

1.Başla:

2.krdnt.txt’den oku:

3.if(dosya!=Null) : while(feof(dosya))

Dosyadan koordinatları oku –x,y;

4.else

Dosya bulunamadı:

hesapla();

Bu fonksiyonda önce tüm x ve y noktalarının orta noktasını hesapladım sonra iki tane döngü üzerinde karşılaştırma yaparak teker teker yarıçapları karşılaştırarak noktaları içerisinde bulunduran en küçük yarıçaplı çemberi hesapladım ve değerleri ana main fonksiyonuna döndürdüm.

Başla:

İ= 0 dan \*a ya kadar dön

Xk ya x[i] değerlerini ekle

yk ya y[i] değerlerini ekle

xk/\*a;

yk/\*a;

i=0 dan … kadar dön:

k1=xk x[0] arası ve yk y[0] arası mesafe nin karesine eşit

j=0 dan j<a ya kadar dön

k2 ye diğer indexlerin mesafesini ata

k1>k2 k1 i k2 ile değiştir mesafeyi 0.1 oranında küçült:

bitir:

Cizimyap();

Bu fonksiyonla öncelikle graphics.h kütüphanesini kullanarak pencere oluşturdum ve bunu deneme yoluyla 20 + yönde 20 – yönde olmak üzere 40 ar birimlik koordinat düzlemi oluşturdum. Aldığım merkez değerine ve yarıçapa göre minimum çevreleyen çemberi merkez koordinatını ve diğer almış olduğum koordinat noktalarını belirten dizi elemanlarıyla noktaların yerini belirledim ve çizdim son olarak Bezier curve algoritmasını araştırarak noktaların yakınından geçen eğriyi çizdirdim.

Bezier curve:

for(t=0; t<=1; t+=0.0001)

{

px=0;

py=0;

for(i=0; i<=n; i++)

{

px=px+(comb(n,i)\*pow(1-t,n-i)\*pow(t,i)\*x[i]);

py=py+(comb(n,i)\*pow(1-t,n-i)\*pow(t,i)\*y[i]);

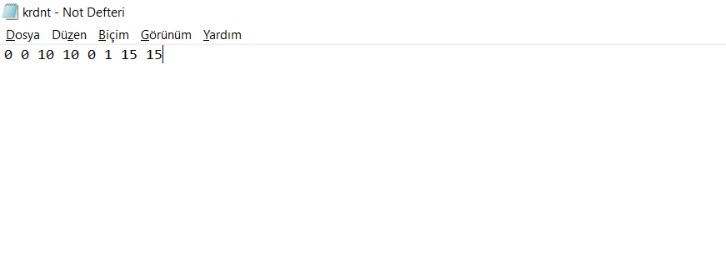
}

putpixel(px,py,2);}

Kodun bu kısmında minimal artışlar yaparak noktalar halinde x ve y koordinatlarının yakınından geçen eğriler çizdirme işlemini yaptım araştırmış olduğum kadarıyla girmiş olduğum koordinatı kaç elemanı varsa eleman sayısının kombinasyonu indeks numarasına öre alınıp t değerinin bire yakınlığının n-i. dereceden üssü ve t değerinin i. Üssünü alıp çarparak ulaşılmaktadır.

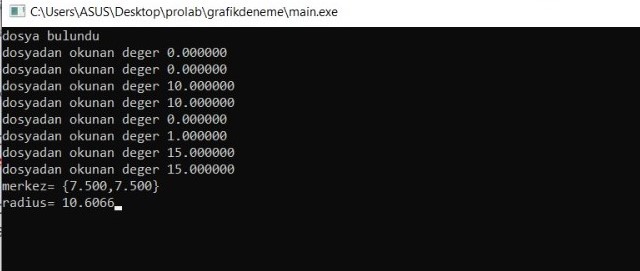
3.DENEYSEL SONUÇLAR:

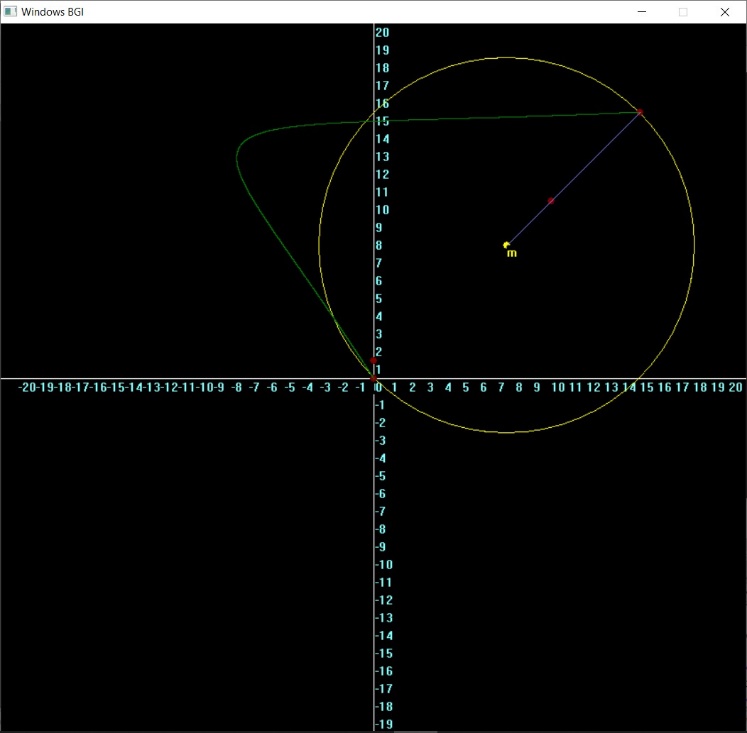
1. koordinat ekleme:



2. Koordinat Değerleri Alma Ve Hesaplama:

(1. Okunan x koordinatı 2. Okunan y koordinatı)





4. KARŞILAŞILAN SORUNLAR VE ÇÖZÜMLER:

Projede başta değişkenleri dosyadan alırken değerleri fonksiyonlara işaretçi olarak göndermek yerine direk göndermiştim bu yüzden dolayı gelen değerler okuduğum değerlere farklı geldiğini gördüm ve değerleri işaretçi olarak göndererek bu sorunu düzelttim. Grafik çizimi yaparken grafik kütüphaneleri hakkında en ufak derecede bir bilgim yoktu bu yüzden graphics.h kurumu yaparken idenin other links options değerini yanlış girdiğim için ide de hatalar aldım daha sonra araştırarak düzelttim. Grafik için line komutunu kullanırken düzlem oluşturmada sorun yaşadım sonra araştırarak onunda birinci değeri için x ikinci değeri için y değerini sol üste göre aldığını öğrendim ve konumunu ona göre ayarladım.

5. SONUÇ:

Bu proje sayesinde grafik kütüphanelerinin nasıl kurulduğunu C programlama dilinde grafik kütüphanelerini kullanmayı öğrendim. Bilmediğim konular hakkında araştırma yapıp bunları kendi başıma öğrenebileceğimi öğrendim.

AKIŞ DİYAGRAMI:

Alınan değişkenlerden merkez koordinatlarını ve yarıçapı hesapla ve ata

K1 i diğer küçük olan indis mesafesiyle değiştir

Eğer k1<diğer indislerle olan mesafeden

i. koordinatlar ile xk ve yk değerleri arası mesafeyi hesapla

Ve k1 e 0.indekteki mesafeyi ata

İ+=1;

İ=0;i<3000;

Koordinat sayısı kadar dön

İ=0’dan koordinat sayısına kadar dön

i. koordinatları topla ve ortalamasını alıp xk ve yk değişkenlerine ata

Programın çalıştığı klasördeki tüm dosyalar bitene kadar oku

0’dan 840’a kadar yatay çizgi çiz

0’dan 800’e kadar dikey çizgi çiz

Merkez koordinatları ile koordinatların m. İndeksi arası çizgi çiz

Verilen x ve y koordinatlarını n bulunduğu noktalara yarı çapı 1,2,3 v e 4 olan dört çember çiz

İ’nin değerini koordinatlara yaz

-İ’nin değerini koordinatlara yaz

İ++;

İ=0 dan i<20; dön

İ’nin değerini koordinatlara yaz

Eb=i. İndeksteki merkez ile koordinatlar arsı mesafe M+=1;

Eb<merkez ile diğer koordinatalar arası mesafe

Koordinat sayısı kadar dön

Eb=sıfırıncı indisle merkezin mesafesi int m=0;

Koordinat sayısı kadar dön

İ=0 dan i<20; dön

İ++;

İ=0 dan i<20; dön

İ++;

-İ’nin değerini koordinatlara yaz

İ=0 dan i<20; dön

İ++;

t=0;t<=1;

px=0,py=0;

İ=0;i<kordinat sayısı

px=px+(comb(n,i)\*pow(1-t,n-i)\*pow(t,i)\*x[i]);

py=py+(comb(n,i)\*pow(1-t,n-i)\*pow(t,i)\*y[i]);

px ve py koordinatlarına nokta koy

ALGORİTMANIN KARMAŞIKLIK ANALİZİ

for (int i = 0; i < 30000; i++)

{

int f = 0;

k1=dist(\*xk-x[0],\*yk-y[0]);

for (int j=1; j<\*a; j++)

{

k2=dist(\*xk-x[j],\*yk-y[j]);

if (k1<k2)

{

k1 =k2;

f = j;

}

}

\*xk+=(x[f]-\*xk)\*p;

\*yk+=(y[f]-\*yk)\*p;

p\*= 0.999;

}

Kodun bu parçasında Big O Notasyonuna göre sabit sayıları kabul etmeyiz döngünün sonsuza gittiğindeki karmaşıklığına baktığımız için döngünün karmaşıklığı O(a)

for(t=0; t<=1; t+=0.0001)

{

px=0;

py=0;

for(i=0; i<=n; i++)

{

px=px+(comb(n,i)\*pow(1-t,n-i)\*pow(t,i)\*x[i]);

py=py+(comb(n,i)\*pow(1-t,n-i)\*pow(t,i)\*y[i]);

}

putpixel(px,py,2);

}

Kodun bu parçasında ise dış döngü sabit olup 1000 kez

Dönecek içerde ise kullanıcını girdiği koordinat sayısına göre değişeceği için n kez dönecek her döngüde belirli işlemler yapılıyor ama bu işlemlerin sayısı girilen değişkenlerin büyümesine göre çok küçük kalacağı için dikkate alınmaz ve buradaki kodunda o(n) olarak kabul edilir.

Genel olarak baktığımızda kodun Big O Notasyonuna göre karmaşıklığı O(n)’dir.

6. KAYNAKÇA:

<https://fatihalparslann.wordpress.com/2017/02/02/c-dilinde-graphics-h-kutuphanesi-1-bolum/?wref=tp>

<http://yunus.hacettepe.edu.tr/~b0145801/c/cgrafik.htm>

<http://kadirselen.blogspot.com/2014/01/grafik-kullanm.html>

<https://www.geeksforgeeks.org/cubic-bezier-curve-implementation-in-c/>

<https://programmerbay.com/c-program-to-draw-bezier-curve-using-4-control-points/>

<https://www.codeproject.com/Articles/25237/Bezier-Curves-Made-Simple>

<https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/FOURTH_EDITION/PROGRAMS/bezier.c>

<https://www.sciencedirect.com/science/article/pii/S0925772114000558>

<https://www.wikiwand.com/en/Smallest-circle_problem>