**Özet: Bu çalışmada Minimum Enclosing Circle Probleminin algoritmasını oluşturulmuştur. Proje oluşturulması için Codeblocks IDE’si ve Allegro kütüphanesi kullanılmıştır. Projede; kullanıcı tarafından N nokta verildiğinde ,tam sayı koordinatlı iki boyutlu düzlemde tüm noktaları içeren ve minimum çevreleyen yarıçaplı çember çizdirilmektedir.**

**Anahtar kelimeler: Minimum Çevreleyen Çember Problemi, C Programlama Dili, Allegro kütüphanesi**

1. **GİRİŞ**

Proje için (bizden istenilen )C programlama Dili ve Codeblocks geliştirme ortamını kullandık.

* Codeblocks platformu; özgür açık kaynak kodlu bir [C++](https://tr.wikipedia.org/wiki/C%2B%2B) [tümleşik geliştirme ortamıdır](https://tr.wikipedia.org/wiki/T%C3%BCmle%C5%9Fik_geli%C5%9Ftirme_ortam%C4%B1). GNU/Linux, Mac, Windows platformlarında çalışır. Genişletilebilir eklenti desteğine sahiptir. Birden fazla derleyiciyi destekler. Başka programlama dillerini de desteklemesiyle birlikte kod yazımı esnasında ihtiyaç duyabileceğiniz birçok yardımcı fonksiyon sunar. Belirttiğimiz bize sağladığı kolaylıklar tercih etme sebeplerimizdir.
* Projede amacımız; Minimum çevreleyen çemberin merkezini ve yarıçapını bulmaktır. Minimum çevreleyen daire hesaplama problemi, Min-Max optimizasyon problemleri olarak bilinen problemler sınıfına aittir. Minimum çevreleyen çember, girilen tüm noktaların çemberin içinde veya sınırlarında yer aldığı bir çemberdir. Merkez noktasının hesaplanmasında noktaların merkeze olan uzaklıklarını sorgular. Minimum çevreleyen daire olması durumunda ise çemberin sınırında bulunan noktaları aramaktadır. Merkez noktasının koordinatlarını bulmak önemlidir. Merkez konumu belirlendiğinde yarıçap ve benzeri şeyler de bulunmuş olur.

Eliz KURTULUŞ

Kocaeli Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği(İÖ)

190202015*@kocaeli.edu.tr*

İrem ÇELİKKANAT

Kocaeli Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği(İÖ)

19020202124*@* kocaeli.edu.tr

**PROGRAMLAMA LABORATUVARI MİNİMUM ÇEVRELEYEN ÇEMBER (MEC)**

Projemizde, merkez noktalarını bulmak için iki ayrı koşul kullanılmaktadır. Girilen nokta sayısına bağlı olarak merkezi bulma yöntemimiz değişiklik göstermektedir. Girilen noktalar ise kullanıcının yazdığı txt dosyasından okutulur. Okunan değerler, birçok farklı yapıyı içinde tutabildiğinden struct yapısına atanır. Ardından uygun olunan koşul belirlenerek merkez noktaları ve yarıçap hesaplanır. Hesaplama işlemlerine yöntem bölümünde daha detaylıca değinilecektir. Kullandığımız arayüz ile de çember çizdirilir.

1. **YÖNTEM**

Bu kısımda projeyi oluştururken kullandığımız araçlar ve yöntemler hakkında ayrıntılı bilgi verilecektir.

1. Arayüz Tasarımı

Allegro, esas olarak video oyunlarına ve multimedya programlamaya yönelik bir çapraz platform kitaplığıdır. Pencereler oluşturma, kullanıcı girişini kabul etme, veri yükleme, resim çizme, ses çalma vb. Gibi yaygın, düşük seviyeli görevleri yerine getirir ve genel olarak temeldeki platformu soyutlar. Ancak, Allegro bir oyun motoru değildir : programı istediğimiz gibi tasarlayabilecek ve yapılandırabileceğimiz bir yapıdır. Allegro da genel olarak çizim, font vs. için kullanabileceğimiz fonksiyonlar barındırır. Bu projede kullanılan fonksiyonlar şunlardır:

1. al\_draw\_textf ( )
2. al\_draw\_line( )
3. al\_draw\_filled\_circle( )
4. al\_draw\_circle( )
5. **al\_draw\_textf() Fonksiyonu**

Gerekli font kütüphanelerini ekleyip fonksiyonları cağırdıktan sonra ekrana yazı yazdırmak amaçlı kullanılan bir fonksiyondur. Ekranda konumlandırılması istenen x ve y koordinatları parameter olarak girilir. Renk seçenekleri için RGB renk kodları belirlenir. Fonksiyon kullanımı şu şekildedir:

ALLEGRO\_FONT \*font=al\_load\_font(&quot;arial.ttf&quot;,12,NULL);

al\_init\_font\_addon();

al\_init\_ttf\_addon();

void **al\_draw\_text**(const ALLEGRO\_FONT \*font, **ALLEGRO\_COLOR** color, float x, float y, int flags, char const \*text)

1. **al\_draw\_line() Fonksiyonu**

X ve Y koordinatları belirlendikten sonra ekranda kalınlığını ve uzunluğunu belirleyebileceğiniz bir doğru parçasını çizmeye yarayan bir fonksiyondur. Fonksiyon kullanımı şu şekildedir:

void **al\_draw\_line**(float x1, float y1, float x2, float y2,**ALLEGRO\_COLOR** color, float thickness)

1. **al\_draw\_circle() Fonksiyonu**

Ekrana çember çizdirmek için kullanılan bir fonksiyondur. Girilen parametrelerle merkez x,y koordinat noktaları ve yarıçap uzunluğu belirlenir. Fonksiyon kullanımı şu şekildedir:

void **al\_draw\_circle**(float cx, float cy, float r, **ALLEGRO\_COLOR** color,float thickness)

1. **al\_draw\_filled\_circle() Fonksiyonu**

Ekrana içi dolu daire çizdirmek için kullanılan bir fonksiyondur. Girilen parametrelerle merkez x,y koordinat noktaları ve yarıçap uzunluğu belirlenir. Girilen RGB renk kodları sayesinde renk seçeneği isteğe göre belirlenebilir. Fonksiyon kullanımı şu şekildedir:

void **al\_draw\_filled\_circle**(float cx, float cy, float r, **ALLEGRO\_COLOR** color)

1. Kullanılan Yapılar

Kullanıcının girdiği koordinatları alabilmek için dosya işlemlerini kullandık. Dosya ile okuma ve ekleme modunda işlemler gerçekleştirdik. Okunan değerleri ise ‘dizi’ kullanarak ‘struct’ yapısına atadık. Karşılaştırma ve koşul durumlarında ‘if’ yapısından faydalandık . Merkez ve yarıçap hesaplamalarında ‘for döngü’lerinden yararlandık.

1. Proje Hesaplamaları

Dosya tanımlayarak dosyamızı read modunda açtık. Bize gönderilen dosyadan verilen noktaların koordinatlarını Struct nokta olarak adlandırdığımız içinde double x ve double y değişkenlerinitanımladığımız yapıya dizi yardımıyla atadık.Aynı zamanda bir struct dizisi oluşturduk ve noktaları bu dizide tuttuk. İlk başta genel çember çizme mantığından ilerledik. Ancak üçten fazla nokta girildiğinde çemberin içine dahil olmayan noktalar olduğunu fark ettik. Bu durumu düzeltmek için yeni bir algoritma oluşturduk. Böylece Nokta sayısına bağlı koşul ortaya çıktı. Eğer üç ve üçten az sayıda nokta varsa; iç içe for döngüleri içinde dizinin indislerini nokta\_uzaklik() fonksiyonuna yolladık.

* **nokta\_uzaklik() Fonksiyonu**

Noktaların yerini belirten sıralı ikililerin apsislerinin büyük olanından küçük olanı çıkartılıp, ordinatlarının da büyük olanından küçük olanı çıkartılır. Apsislerin farkının karesi ile ordinatların farkının karesinin toplamı bize iki nokta arasındaki uzunluğun karesini verir. Bu toplamın karekökü ise iki nokta arasındaki uzaklığı verir. Bu yöntem sonucunda noktalar arası uzaklık bulunmuş olur. Uzaklık bulma formülü kısaca şu şekilde gösterilebilir:

IABI = sqrt(pow(x1-x2,2)+pow(y1-y2,2))

Uzaklıklar hesaplandıktan sonra sıralanması gerekir. Amaç en uzak iki noktayı bulabilmektir. Sıralama algoritması kullanmak gerekiyordu. Biz prototip aşamasında basit uygulanabilir olmasından dolayı Selection Sort algoritmasını kullanmaya karar verdik Algoritmanın çalışma ilkesi basittir.

* **Selection Sort Algoritması**

Başlarken dizinin ilk öğesini en küçük öğe kabul eder. Tabii, bu kabul geçicidir. Sonra kalan terimler arasında daha küçükler varsa, onların en küçüğü olan terimle takas eder. O terimi en sola koyar; bu terim sıralama sonunda ilk terim olacaktır. Diziden seçilen bu en küçük terimi çıkarınca, geri kalan alt dizine aynı yöntemi uygular. Alt diziden seçtiği en küçük öğeyi, ilk seçtiğinin sağına koyar. Dizinin sol ucunda iki terimli alt dizi küçükten büyüğe doğru sıralı bir alt dizi oluşturur. Bu işleme kalan altdizinler bitene kadar devam edilir. Her adım başlarken, sol yanda sıralı bir alt dizi, sağ yanda sırasız bir alt dizi vardır. Her adımda sıralı dizi bir terim artar, sırasız dizi bir terim azalır. Sağ yandaki sırasız alt dizi bitince, sıralama işlemi biter.

Bu algoritma bizim noktalar arasındaki uzaklığı bulduktan sonra uzaklıkları karşılaştırma aşamasında kullanmak için fazlasıyla yeterliydi. Daha sonraki aşamalarda da değiştirme gereği duymadık. Bulunan en uzak iki noktanın aralarındaki mesafenin ikiye bölünmesiyle yarıçapı hesaplamış olduk. Bulduğumuz iki noktanın x ve y değişkenlerini, ayrı ayrı tanımladığımız Struct yapılarına atadık. Merkez\_bul() fonksiyonuna göndererek merkezin koordinatlarını belirledik. Bu fonksiyonda merkez bulma formülü olan x0= (x1+x2)/2 ve y0= (y1+y2)/2 kullanıldı. Sonuç, merkez için tanımladığımız m Structına gönderilir.

Verilen nokta sayısı üçten büyükse merkezi, orijin kabul ettik. Burada ise dizinin indisleri yeni\_uzaklik() fonksiyonuna göndererek karşılaştırdık.

Yeni\_uzaklik()=sqrt((n1\*n1)+(n2\*n2))

Merkez, noktaların uzaklıklarına göre yerinden milim milim kaydırılarak hesaplatılır. Döngüde en son hesaplatılan uzunluğu sqrt() fonksiyonuna göndererek yarıçapı bulduk. Böylece çemberi çizdirmek için gerekli değerlere sahip olmuş olduk.

1. MATEMATİKSEL İŞLEMLER

B-Spline fonksiyonu ile noktaların yakınından veya üzerinden geçen eğri çizdirildi.Catmull-Rom Splinedan yararlanıldı. Catmull-Rom yinelemeli bir algoritma kullanılarak değerlendirilebilir. Dört kontrol noktası tarafından tanımlanan enterpolasyonlu bir eğri (kontrol noktalarından geçen bir eğri) türüdür. Kontrol noktaları P0,P1,P2,P3’tür.Eğri sadece P0 ve P1 noktalarından oluşur.

Catmull-Rom formülasyonu türlerine kıyasla birçok istenen matematiksel özelliğe sahiptir. İlk olarak, bir eğri parçası içinde döngü veya kendi kendine kesişme oluşturmayacaktır. İkincisi, bir eğri segmenti içinde doruk noktası asla oluşmaz. Üçüncüsü, kontrol noktalarını daha sıkı takip eder. P0,P,P2,P3 için kullanılan ağırlık faktörleri aşağıdaki gibidir;

Y1=t3+22-t

Y2=3x3-5x2+2

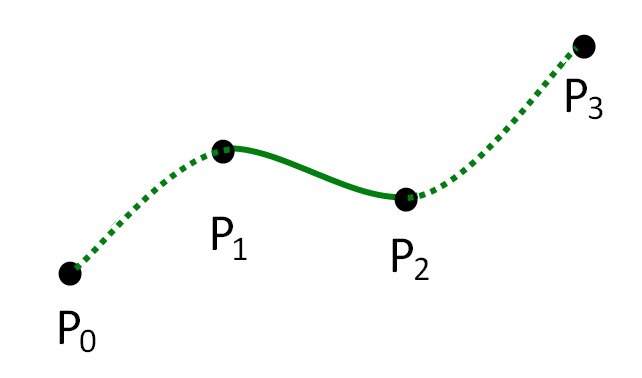
Y3=-3x3+4x2+x

Y1=x3-x2

**Formül:**

P0(t3+22-t)+P1(3x3-5x2+2)+P2(-3x3+4x2+x

)+P3(x3-x2)



Şekil 1 Spline Örneği

**Spline Kaba Kodu**

►Başla

► i yi 0.000 dan başlayarak girilen sayı kadar 0.001 arttır

►spline\_hesaplama() fonksiyonuna artış miktarı ve dizideki noktaları ve nokta sayısını yolla

►Po P1 P2 P3 ağırlık faktörlerini hesapla

►u değerini 1’in üstüne çıkmaması için int olarak kendinden çıkar ve hep 1 in altında kalmasını sağla

►af1,af2,af3 ve af4 de ağırlık faktörlerinin matematiksel fonksiyonlarını hesapla

►Dizideki her bir noktayı kendi ağırlık faktörü ile carp

►Bunu her x ve y noktası için yap ve struct Spline.x ve Spline.y’ye ata

►Fonksiyonda spline structını döndür

►Gelen fonksiyon değerini spline\_nokta\_ciz()fonksiyonuna yolla

►Gelen değerleri koordinat sistemine uyarla

► al\_draw\_filled\_circle() fonksiyonu ile spline noktalarını çizdir

►Bitir

1. PERFORMANS

Programda kullanılan en yüksek dereceli fonsiyon üzerinden hesaplanma yapılmıştır.B-Spline ağırlık faktörlerinden biri baz alınmıştır.

Q1=-t3+2t2-t+0 için k=0

t>0 için (bu aynı zamanda t>k)

-t3+2t2-t+0≤-t3+2t3-t3+-t3

+t3 ≤ t3

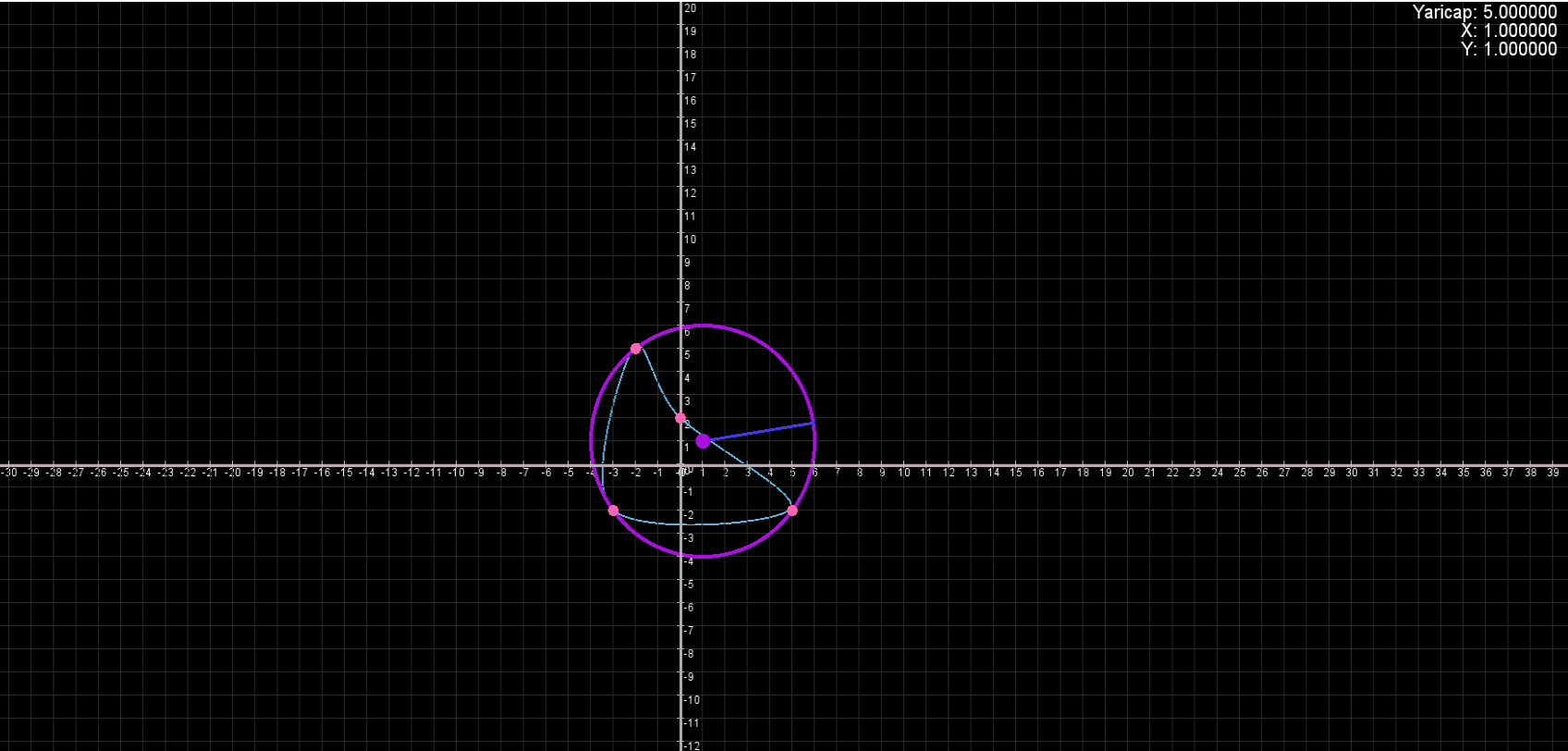
Öyleyse C=1 ve k=0

F(x) ≤ 1t2+0, x>k durumu sağlanır.

Sonuç olarak O(t3) olur.

1. **DENEYSEL SONUÇLAR**

Bu projede sıklıkla Geometri ve Matematik uygulamaları kullanıldı ve bir çok sonuç elde edildi.Elde edilen sonuçların doğruluğunu test etmek amacıyla veriler görselleştirerek, kontrol edilip bu modellerin çıktılarıyla kıyaslanarak kendi test yöntemlerimizi oluşturduk. Test edilen sonuçlar ekran çıktılarıyla kesinleşti.



Şekil 2 Ekran Çıktısı

1. **SONUÇ**

Bu projede arayüz tasarım kütüphaneleriyle tanıştık.Ekip çalışması ile proje yönetimi yeteneğimizi geliştirdik. Projede, algoritma oluşturmaya ve kod yazmaya dair tecrübeye sahip olduk. Teşekkür ederiz.

1. **KAYNAKÇA**

(1)<https://tr.wikipedia.org/wiki/Code::Blocks> (2)<http://cs.rkmvu.ac.in/~shreesh/media/mec.pdf> (3)<https://en.wikipedia.org/wiki/Smallest-circle_problem> (4)[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/%20)

ÇEMBER AKIŞ DİYAGRAMI

