Minimum Enclosing Circle ve B-Spline Project

Computer Engineering, Kocaeli University
Berkay Yasin Çiftçi
190202056@kocaeli.edu.tr

Computer Engineering, Kocaeli University
Ferhat Açıkalın
180202065@kocaeli.edu.tr

ÖZET

Minimum çevreleyen çember ve B-Spline projesinde "dosya_islemleri.h" adlı kütüphaneden yararlanarak "noktalar.txt"dosyasını okur. Bu dosyanın içindeki x ve y koordinat bilgilerini program tarafında erişim sağlanır. Dosyalardaki her bir x ve y koordinat bilgileri "nokta" tipindeki "noktalar" adlı struct dizisine kaydedilir. Dosyadaki noktalar "cizim_algoritmalari.h" kütüphanesinden matematiksel işlemlerden geçer ve "cizim_fonksiyonları.h" kütüphanesinden gerekli fonksiyonlar çağrılarak girilen noktaları minimum çevreleyen çember çizdirilir. "egri_cizdirme_algoritmalari.h" kütüphanesinden matematiksel işlemlerden geçer ve "eğri_cizdirme_fonksiyonlari.h" **kütüphanesindeki** fonksiyonlar çağrılarak aynı koordinat ekseninde noktalardan geçen veya yakınından geçen eğriyi çizer.

GİRİŞ

Proje ilk görsel kütüphane içeren projemiz olduğu için bazı sorunlarla karşılaşıp bunları özümsedik. Çizim işlemleri için OpenGl ve Glut kütüphanesini kullandık.

YAPILAN ARASTIRMALAR

İlk olarak grafik kütüphanesi nedir, nasıl eklenir gibi soruları internet kaynakları vasıtasıyla araştırıp özümsedik.

Devamında grafik kütüphanesindeki bizim için gerekli olabilecek fonksiyonları işlevlerini internetten araştırıp özümsedik.

Temel olarak grafik kütüphanesi hakkında bilgiler edindikten sonra projenin kalan kısmını algoritma ve matematiksel olay için gerekli araştırmalar yapmamız gerekti.

Verilen noktaların arasından minimum çevreleyen çember ve eğriyi çizdirmek için gerekli matematiksel ifadeleri edinmek için araştırma yaptık. Yapılan araştırmalar kaynakça kısmında belirtilecektir.

Yapılan araştırmalar ve algoritma hakkında daha detaylı bilgiler temel bilgiler altında sizlerle paylaşılacaktır.

TEMEL BİLGİLER

Eğri Çizdirme Yöntemi:

Eğri çizdirmek için bezier curve adı verilen yöntemi kullandık. Bu yöntemde 2, 3 ve 4 adet nokta barındıran eğriler çizilmesinde kullanılır. Aşağıdaki denklem dikkat edilirse binom üçgenindeki bulunan katsayılar kullanılmıştır. Binom üçgeninin üçüncü satırı kullanılmıştır. Formülü ise aşağıdaki gibidir:

B(t)=(1-t)^3*P0+3(1-t)^2*t*P1+3(1-t)*t^2*P2+t^3P3 0<=t<=1

Bu formülü her nokta için tek tek yapmak istersek aşağıdaki gibi formül elde ediyoruz.

Aşağıdaki denklemi her bir nokta için toplarsak yukarıdaki denklem elde edilir.

```
x=x+(comb(n,i)*[(1-t)^(n-i)]*(t^i)*Px);
 y=y+(comb(n,i)*[(1-t)^(n-i)]*(t^i)*Py);
```

buradaki n, nokta sayısının bir eksiğidir. 0 < i <nokta sayısı, t değeri ise ne kadar küçük olursa çizilen eğrimiz o kadar az hata payımız olur.

Başlangıç olarak x =0 y =0 alıyoruz ve eğrinin başlangıç ve bitiş noktaları belirleyip arasında eğri oluşacaktır.

Minimum Çevreleyen Çember:

Minimum çevreleyen çemberi çizdirirken kullanılan yaklaşım şu şekildedir:
Çember iki noktayı kesiyorsa, bu iki nokta arasındaki doğru çemberin çapıdır dolayısıyla bu doğrunun ortası da çemberin merkezini verir.
Çember üç veya daha fazla noktayı kesiyorsa, üç noktadan çember eşitliğini kullanılarak çemberin merkezi bulunur. Üç noktadan çember eşitliği şu şekildedir:
A,B,C üç nokta olsun AB ve BC iki doğru olsun, AB doğrusunun ortasından çizilen ve AB doğrusuna dik olan doğru ile BC doğrusunun ortasından çizilen ve BC doğrusuna dik olan doğrunun kesiştiği nokta çemberin merkezidir.

$$y'_{AB} = -\frac{1}{m_{AB}}\left(x - \frac{x_A + x_B}{2}\right) + \frac{y_A + y_B}{2}$$

$$y'_{BC} = -\frac{1}{m_{BC}}\left(x - \frac{x_B + x_C}{2}\right) + \frac{y_B + y_C}{2}$$

Bu eşitliklerden,

$$x = \frac{m_{AB}.m_{BC}(y_A - y_B) + m_{BC}(x_A + x_B) - m_{AB}(x_B + x_C)}{2(m_{BC} - m_{AB})}$$

elde edilir. Çemberin y noktasını hesaplamak için üstteki eşitliklerden herhangi birine bulunan x noktası yazılarak hesaplanabilir.
Çemberin yarıçapı merkez ile çemberin üzerindeki bir noktanın arasındaki uzaklığa eşittir.

Algoritma:

Program ilk olarak “dosya_islemleri.h” kütüphanesinde tanımlı olan “dosya_oku” adlı fonksiyonu çalıştırarak “noktalar.txt” dosyasında satır yazılmış olan noktaları “global_tanimlamalar.h” içinde tanımlı olan “nokta” tipindeki “noktalar” adlı struct dizisine kaydeder. “noktalar” dizisi tüm fonksiyonlar içinden global olarak erişilebilirdir. Ayrıca dosyadan kaç nokta okunduğunu “nokta_sayisi” adındaki global bir değişkene atar.
Daha sonra program, “cizim_algoritmalari.h” içinden çağrılan “cember_t” (“global_tanimlamalar.h” içinde tanımlı) tipindeki “en_kucuk_kapsayan_cember” adlı fonksiyona “noktalar” dizisini parametre olarak gönderir. Bu fonksiyon, ilk olarak “cember_t” tipinde ”enkkc” adında değişken tanımlar, “enkkc” çemberin x,y koordinatlarını ve yarıçap bilgisini tutar. Yarıçap başlangıçta sonsuz olarak tanımlanır.
Sonrasında noktaları iç içe iki for döngüsüyle ikili olarak “iki_noktadan_cember” adlı fonksiyona gönderir. “iki_noktadan_cember” fonksiyonu parametre olarak aldığı noktaların, matematiksel detayları daha önce anlatıldığı gibi, orta noktasını “merkez” değişkenine atar. “merkez” ile parametre olarak alınan noktalardan ilki “uzaklik” fonksiyonuna gönderilir. “uzaklik” fonksiyonunun döndürdüğü değer yarıçapa eşittir.
“iki_noktadan_cember” fonksiyonu merkezi ve yarıçapı bulunan çemberi döndürür. Döndürülen değer “çember_t” tipindeki “temp” değişkenine atanır. “temp” değişkeninde tutulan çemberin yarıçapı “enkkc” değişkeninde tutulan çemberin yarıçapından küçük ise ve geçerli bir çember ise “enkkc=temp” şeklinde atanır. Çemberin geçerli bir çember olup olmadığı “geçerli_cember” fonksiyonunun tüm noktaların çember içerisinde veya üzerinde olup olmadığını tek tek kontrol etmesi ile bulunur.
Döngü bittikten sonra, en_kucuk_kapsayan_cember fonksiyonu iç içe üç for döngüsü içerisinde, noktaları üçlü olarak “uc_noktadan_cember” fonksiyonuna gönderir, matematiksel detayları daha önce anlatılan üç noktadan çember eşitliğinden çemberin merkezi bulunur. Parametre olarak alınan üç noktadan ilki ve “merkez”, “uzaklik” fonksiyonuna gönderilir, “uzaklik” fonksiyonu yarıçap değerini döndürür. “uc_noktadan_cember” yarıçapı ve merkezi bilinen çemberi döndürür. Buradan gelen değer “çember_t” tipindeki “temp” değişkeninde tutulur. “temp” değişkeninde tutulan çemberin yarıçapı

“enkkc” değişkeninde tutulan çemberin yarıçapından küçük ise ve geçerli bir çember ise “enkkc=temp” şeklinde atanır. “en_kucuk_kapsayan_cember” fonksiyonu son olarak “enkkc” değişkenini döndürür. Döndürülen değer daha sonra kullanılmak üzere “cizilen_cember” değişkeninde tutulur. **Bu işlemin zaman karmaşıklığı analizi yapılırken en kötü senaryo durumuna bakılır. İlk döngüde iç içe iki for döngüsü var, bu döngünün de içerisinde çağrılan “gecerli_cember” fonksiyonu da bir döngüye sahiptir. İkinci döngümüz iç içe üç for döngüsünden oluşuyor ve yine her defasında “gecerli_cember” fonksiyonu çağrılıyor. Bu döngülerin tamamı lineer artandır. Buradan, ilk döngü için T(n)=n*(n-1)*n , ikinci döngü için T(n)=n*(n-1)*(n-2)*n şeklinde zaman karmaşıklığı ifade edilebilir. İki ifade birleştirilirse T(n)=(n^4)-2(n^3)+(n^2) karmaşıklığı bulunur. Big O notasyonuna göre en büyük artış gösteren terim alınır, bu durumda zaman karmaşıklığı O(n^4) olarak bulunur.** Sonrasında, program “eğri_cizdirme_algoritmalari.h” içerisinde tanımlı “egri_hesaplama” fonksiyonunu “noktalar” parametresi ile çalıştırır. “egri_hesaplama” fonksiyonu içerisinde for döngüsü “t=0.0” dan başlayıp değerini 0.001 arttırarak “t<1.0” olana kadar çalışır. “t” değişkeninin arttırılma oranı eğrinin hassasiyetini gösterir. Döngünün içerisinde “x =0” ve “y=0” olarak tanımlanır. Bu döngünün içerisinde yeni bir for döngüsünde her nokta için matematiksel detayları verilen denklem hesaplanır, “x” ve “y” üzerine eklenir. İçerideki döngü bitince eğrinin bir noktası bulunmuş olur. Bulunan nokta “global_tanimlamalar.h” içerisinde tanimli nokta tipindeki egri_noktalari dizisine eklenir. Dışarıdaki döngü “t<1.0” koşulunu sağlamayıncaya kadar çalışır. Bu durumda çizilecek eğri için elimizde 1000 tane nokta olur. **Bu işlemin zaman karmaşıklığı analizi yapılırken en kötü senaryo durumuna bakılır. “egri_hesaplama” fonksiyonu içerisinde iç içe iki for döngüsü var. Dışarıdaki döngü “t=0.0” değişkeni “t<1.00” olduğu sürece 0.001 arttırılarak çalıştığı için toplam 1000 defa çalışır. Bu döngünün içerisindeki döngü “i=0” dan başlayıp “i<nokta_sayisi” olduğu sürece çalışır. Eğrinin noktalarını bulurken kullandığımız “i” parametresini alan “comb” fonksiyonun içinde de bir döngü tanımlıdır. Bu döngü aldığı “i” ye eşit olan değişken kadar çalışır. Toplamda 0+….+(nokta_sayisi-1) kadar çalışır. “comb” fonksiyonu x ve y noktaları için iki defa çağrılır. Tüm bu veriler kullanılarak T(n)=1000*n*(n^2+n)=1000n^3+1000n^2 karmaşıklığı elde edilir. Big O notasyonuna göre en büyük artış gösteren terim alınır, bu durumda zaman karmaşıklığı O(n^3) olarak bulunur.**

Bir sonraki adımda program elde edilen verileri kullanıcıya gösterme aşamasına geçer. Bu nedenle ilk olarak, “GL/freeglut.h” kütüphanesinden çağrılan fonksiyonlar ile gerekli ayarlamalar yapılır. “glutDisplayFunc(display)” şeklinde opengl içeriği oluşturmak için kullanacağımız fonksiyonun adı “display” olarak bildirilir. “glutMainLoop” fonksiyonu ile grafik arayüz döngüsü başlatılır. “display” fonksiyonu içerisinde ilk olarak “koordinat.h” içerisinden “koordinat_duzlemi_ciz” fonksiyonu çağrılır. Bu fonksiyon ilk olarak eksenleri çizer, sonra -20 den 20 ye kadar x ve y eksenlerine koordinat değerlerini yazar. Sonra, “cizim_fonksiyonlari.h” içerisinden çağrılan “noktalari_ciz” fonksiyonu, “noktalar” parametresiyle çalışır ve ekranda dosyadan okunan noktaları gösterir. Yine “cizim_fonksiyonlari.h” içerisinden çağrılan “enkk_ciz” fonksiyonu, “cizilen_cember” parametresiyle çalışır ve ekrana minimum çevreleyen çemberi çizdirir. “enkk_ciz” fonksiyonu içerisinde for döngüsü “i=0” dan “i<360” koşulu bozulana kadar çalışır. Döngünün içerisinde çemberin parametrik

denklemine göre her açı için x ve y noktaları bulunur (“i” değişkeni açıyı temsil eder.) ve ekrana çizdirilir. Çizilen çemberin bilgileri ekranın köşesine yazılır. En son “egri_cizdirme_fonksiyonlari.h” içinden çağrılan “egri_ciz” fonksiyonu “egri_noktalari” dizisinde tutulan noktalardan oluşan eğriyi ekrana çizer.

Sözde Kod:

1) “dosya_oku” fonksiyonunu çağırıp noktalar.txt dosyasını oku.
2) Okunan noktaları “struct nokta” tipindeki “noktalar” dizisine ekle.
3) Okunan nokta sayısını “nokta_sayisi” değişkenine ata.
4) “struct cember_t” tipindeki “en_kucuk_kapsayan_cember” fonksiyonuna “noktalar” parametresini gönder.
5) Minimum çevreleyen çemberi tutan “struct çember_t” tipinde “enkkc” değişkenini oluştur.
6) “enkkc” de tutulan çemberin yarıçapını sonsuz olarak ata.
7) For döngüleri ile noktaları ikişer ikişer “iki_noktadan_cember” fonksiyonuna gönder.
8) “iki_noktadan_cember” fonksiyonunda, iki noktayı kesen çemberin merkezini bul.
9) “iki_noktadan_cember” fonksiyonunda uzaklık formülü ile yarıçapı bul.
10) “iki_noktadan_cember” fonksiyonunda “struct cember_t” tipinde “temp” değişkenine yarıçap ve merkezi ata.
11) “iki_noktadan_cember” fonksiyonunda “temp” çemberinin yarıçapı “enkkc” çemberinden küçük ise “enkkc=temp” olarak ata.
12) Döngüyü bitir.
13) İç içe for döngüleri ile noktaları üçerli olarak “uc_noktadan_cember” fonksiyonuna gönder.
14) “uc_noktadan_cember” fonksiyonunda çemberin merkezini bul.
15) “uc_noktadan_cember” fonksiyonunda çemberin yarıçapını bul.
16) “uc_noktadan_cember” fonksiyonunda “struct cember_t” tipinde “temp” değişkenine yarıçap ve merkezi ata.
17) “uc_noktadan_cember” fonksiyonunda “temp” çemberinin yarıçapı “enkkc” çemberinden küçük ise “enkkc=temp” olarak ata.
18) “enkkc” değişkenini return et.
19) “egri_hesaplama” fonksiyonuna noktalar parametresini gönder.
20) Fonksiyonda for döngüsünü “t=0.0” dan başlat .
21) “x=0” ve “y=0” olarak tanımla.
22) For döngüsü ile “noktalar” dizisini saydır.
23) Her nokta “t” değişkeniyle birlikte denklemde işleme sokularak “x” ve “y” üzerine eklenir.
24) Tüm noktalar bitince, “x” ve “y” değişkenlerini “struct nokta” tipindeki “egri_noktalari” dizisine ekle.
25) “t<1.0” ifadesi sağlanmayınca döngüyü bitir.
26) Elde edilen verileri kullanıcıya gösterme aşamasına geç.
27) Pencere oluşturmak için gerekli Glut fonksiyonlarını çağır.
28) “glutDisplayFunc” fonksiyonuna “display” adlı fonksiyonun ismini gönder.
29) “glutMainLoop” grafik arayüzü başlat.
30) “display” fonksiyonu içerisinde “koordinat_duzlemi_ciz” fonksiyonu ile koordinatları çiz.
31) “display” fonksiyonu içerisinde “noktalari_ciz” fonksiyonunu ”noktalar” parametresi ile çağır.
32) Ekranda noktaları göster.
33) “display” fonksiyonu içerisinde “enkk_ciz” fonksiyonunu “cizilen_cember” parametresi ile çağır.
34) For döngüsünü “i=1” den “i<360” a kadar döndür.

35) Her “i” değeri için çemberin parametrik denkleminde çemberin noktalarını hesapla.
36) Hesaplanan noktaları birleştirerek minimum çevreleyen çemberi çiz.
37) “display” fonksiyonu içerisinde “egri_ciz” fonksiyonunu “egri_noktalari” parametresi ile çağır.
38) Eğriyi ekrana çizdir.

Karşılaştığımız Sorunlar:

Grafik kütüphanesini kullanarak koordinat ekseni, çember ve eğri çizdirme konusunda bir takım sorunlar yaşadık.

Noktaları dosyadan okuduktan sonra nasıl tutacağımızı konusunda başlangıçta biraz sorun yaşadık.

Minimum çevreleyen çember ‘ in nasıl yapacağımız konusunda fikir sahibi olamadık. Biraz araştırma sonucunda fikir sahibi olduk.

Eğri çizdirme konusunda birkaç farklı yöntem olduğunu araştırmalar sonucunda öğrendik. Bunlar dan ikisi B-Spline ve Bezier Curve yöntemi hakkında araştırma yapmamıza rağmen bir sonuç elde edemedik. Sonra Bezier Curve yöntemini kullanmaya karar verdik.

Projenin Bize Kattığı Yararlar:

OpenGl Glut kütüphanesini kullanmayı öğrendik.
Bezier Curve yöntemini ve minimum çevreleyen çember çizdirmek için gerekli matematiksel işlemleri öğrendik.

Kullanılan Fonksiyonlar:

en_kucuk_kapsayan_cember():
Minimum çevreleyen çemberi çizdirir.

uzaklik():
İki nokta arasındaki uzaklığı döndürür.

gecerli_cember():
Çemberin geçerli olup olmadığını bulur.

iki_noktadan_cember():
Verilen iki noktadan çember oluşturur.

uc_noktadan_cember():
Verilen üç noktadan çember oluşturur.

noktalari_ciz():
Ekrana noktaları çizer.

enkkc_ciz():
Minimum çevreleyen çemberi çizer.

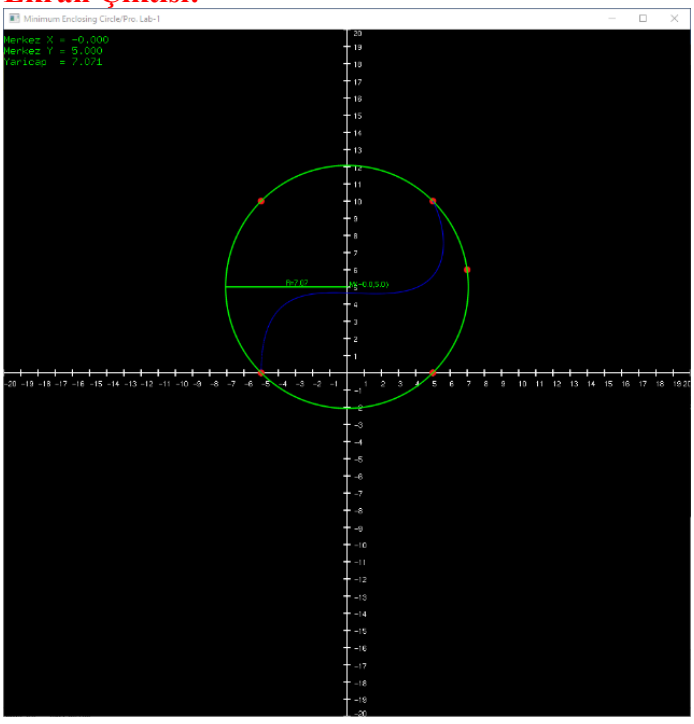
egri_hesaplama():
Eğrinin noktalarını bulur.

egri_ciz():
Ekrana eğriyi çizer.

dosya_oku():
noktalar.txt dosyasını okur.

koordinat_duzlemi_ciz():
Ekrana koordinat düzlemini çizer.

Ekran Çıktısı:



KAYNAKÇA:

https://www.youtube.com/watch?v=mkPMv8rtaII&feature=emb_logo

https://tr.wikipedia.org/wiki/B%C3%A9zier_e%C4%9Frisi
https://web.archive.org/web/20150414092649/http://bilgisayarkavramlari.sadievrenseker.com/2009/10/31/bezier-egrileri-bezier-curves/
https://tr.mort-sure.com/blog/difference-between-bezier-curve-and-b-spline-curve/
https://zehraduman.wordpress.com/2015/08/31/bezier-curve-algoritmasi/
https://www.glprogramming.com/red/chapter12.html
https://www.tutorialspoint.com/computer_graphics/computer_graphics_curves.htm
http://www.swiftless.com/opengltuts.html
http://paulbourke.net/geometry/circlesphere/
http://freeglut.sourceforge.net/docs/api.php