Kaynak Arama ve Çıkarma Operasyonları Projesi

Bilgisayar Mühendisliği
Kocaeli Üniversitesi

Yaren Aybala KOŞAR
220201023
yarenaybala2004@hotmail.com

Çağla Gök
220201060
caglagok369@gmail.com

I. ÖZET

Denizlerde doğal kaynak arama ve çıkarma operasyonları gerçekleştiren bir şirket maksimum kar elde edebilmek amacıyla arama bölgesini en doğru sayı ve optimum boyutlara bölünmüş alanlara ayırmayı hedeflemektedir. Görevimiz bir URL ile verilecek olan koordinat verilerini kullanarak uygun görselleştirme yapmak ve belirtilmiş koşulları sağlayan kar-maliyet durumunun çıktı olarak sunulmasıdır.

Projeye başlarken öncelikle bize verilen ve bizden istenilen verileri analiz ettik. Projeyi yazacağımız programlama dili olan C için uygun IDE ve kütüphane araştırması yaptık. Daha sonra aşamaları kendi içinde parçalayarak adım adım neler yapabileceğimiz üzerinde konuştuk. Kodlama kısmına geçtiğimizde ise ilerlemek sandığımızdan da zor oldu. Yeni öğrendiğimiz kütüphaneleri kullanmak, çeşitli matematiksel hesaplamalar yapmak ve kompleks algoritmalar yazmak çok vaktimizi aldı. Ama zorlu ve uzun süren uğraşlar sonucu projede epey yol kat ettik. Bu raporda da yaptığımız projeyi her detayıyla paylaşıyor olacağız.

II. GİRİŞ

C dilinde yazacağımız projenin grafik işlemlerini yerine getirmesi için graphics.h kütüphanesinin kullanılabileceğini düşündük ancak çok eski bir kütüphane olduğundan 32bitte çalışan bir IDE'ye ihtiyaç duyduk. Bunun için DevC++ idesinin farklı bit ortamlarında çalışabilmesinden faydalandık. Verilen URLdeki verileri alabilmek için Libcurl kütüphanesinin 32 bit versiyonunu kullandık.

Programın ana fonksiyonları şunlardır:

- URL'den veri çekmek.
- Kullanıcıya kodun devamında kullanılacak koordinatları seçtirmek.
- Koordinatlarda verilen şekilleri doğru bir şekilde BGI penceresinde göstermek.
- Şekillerin alanlarının hesaplanması ve rezerv bölge sınırlarının belirlenmesi.
- Şekillerin karesel bölgelere ayrılması ve ikinci pencerede gösterilmesi.
- Maliyet hesaplarının yapılması.

III. YÖNTEM VE İLERLEYİŞ

Şekilleri oluşturan koordinat noktalarını URL den çekmek için libcurl kütüphanesinden faydalandık. Bu kütüphaneyi kullanabilmek için kütüphanenin fonksiyonlarını öğrendik.

İlk olarak çekilen koordinatları kullanabilmek için bir fonksiyon tanımladık ve bu fonksiyon sayesinde URL'nin içindeki bilgileri bir diziye kopyalayıp ekrana yazdırdık.

```
1B(5,5)(13,12)(8,17)(1,10)(5,5)F
2B(20,20)(30,20)(20,40)(10,40)(20,20)(40,22)(50,32)(30,32)(40,22)F
```

(Verilen koordinatların ekran görüntüsü)

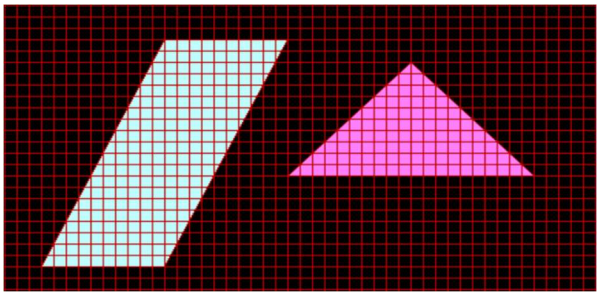
Bu diziyi, kullanıcıdan seçmesi istenilen satırı seçtirtmek için kullandık. Satırı seçtikten sonra seçilen satırın içinde bulunan 2B ve F karakterlerini silmemiz gerekmekteydi. Biz de bu karakterleri sildirebilmek için seçilen satırı yeni bir diziye (secilen_satir) atadık.

Ardından koordinat noktalarını farklı işlemlerde kullanmak üzere x_values ve y_values adı altında iki diziye daha parçaladık ve bu dizilerin eleman sayılarını index değişkeni ile tuttuk.

Parçalanmış x_values ve y_values dizilerini koordinatlar içerisindeki şekil ve bu şekillerin köşe sayılarını buldurmak için bir döngüde kullandık. Çokgen sayısını tutan bir değişken ve çokgenlerin köşe sayısını tutan yeni bir dizi tanımladık.

Şekilleri çizdirebilmek için ayrık olan x_values ve y_values dizimizi sonuç_dizi dizisinde birleştirdik.

Graphics.h kütüphanesiyle birlikte ilk açılan pencerede seçtiğimiz satırdaki koordinatlardaki şekilleri sorunsuz bir şekilde çizdirebildik. Ancak verilen koordinatların pencere üzerinde küçük görünmesinden ve anlaşılmasının güç olmasından dolayı ölçeğimizi 10 kat büyütmemiz gerekti bu nedenle olcek_sonuc_dizi olarak bir dizi oluşturduk.



(Burada görülen çokgenler 10 kat büyütülmüş boyuttadır.)

Şekillerin birden çok olma durumlarında çizimlerin yanlış olduğunu gözlemledik bu nedenle şekil sayılarını hesaplatıp koordinatın içerdiği şekil sayısına göre koşullandırdık.

Daha sonra graphics.h fonksiyonlarını kullanarak çizme ve dolgu işlemlerini yaptık. Koordinat düzleminde gösterebilmek adına birim kareler çizdirdik. İlk penceredeki birim kareleri çizdirmek için "cizilenKareler" fonksiyonunu yazdık.

Şekillerin alan hesabı için çokgen alanı hesaplayan matematiksel bir yöntem olan "Shoelace Formülü"nü tercih ettik.

Shoelace Formülü:

Eğer bir çokgenin köşe noktalarını $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$ olarak temsil ediyorsa, bu çokgenin alanı aşağıdaki formülle hesaplanabilir:

$$\left|\frac{1}{2}\sum_{i=1}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) + \frac{1}{2}(x_n y_1 - x_1 y_n)\right|$$

Burada, n çokgenin kenar sayısını temsil eder ve köşe noktaları sırayla $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$ şeklindedir.

İşte verilen fonksiyonun bu formül ile ilişkilendirilmesi:

int x1 = sonuc_dizi[i]; ve int y1 = sonuc_dizi[i + 1]; ile ilk köşenin x1 ve y1 koordinatları alınır.

int x2 = sonuc_dizi[(i + 2) % num_sonuc_dizi]; ve int y2 = sonuc_dizi[(i + 3) % num_sonuc_dizi]; ile ikinci köşenin *x2* ve *y2* koordinatları alınır.

Bu değerleri kullanarak, $x_i y_{i+1} - x_{i+1}$'yi hesaplanır ve tüm kenarlar için bu değerlerin toplamı alınarak sonuç bulunur. Formüldeki i+1 ve i+2 değerlerinde modül işlemi, dizinin döngüsel olmasını sağlamak için kullanılır.

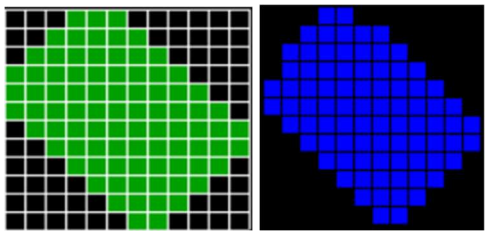
Daha sonra, bu toplam değer mutlak değer alınır ve 2'ye bölünerek çokgenin alanı elde edilir.

Projemizin 2. aşamasında ise bizden istenilen şekillerin birim karelere bölünmüş alanları üzerinden yapılacak olan hesaplamalar için birim kare saydırabileceğimiz bir algoritma geliştirmemiz gerekti.

Geliştirdiğimiz algoritmanın biraz karmaşık olmasından kaynaklı olarak istediğimiz birim kare hesabını ve istenilen birim karelerin boyanmasını bazı istisna şekillerde küçük sapmalarla doğru yaptıramadık ama genel anlamda yazdığımız fonksiyon sonucu doğruya yakın bulmaktadır.

Bu algoritmayı oluştururken ray casting ve collision detection algoritmalarını kullanmanın bizim problemimizi çözebileceğini düşündük ve çeşitli teşebbüslerde bulunduk fakat fonksiyonlar bize hem istediğimiz sonuçları vermedi hem de fazlasıyla zaman kaybı yaşamamıza sebep oldu.

En sonunda maksimum ve minimum koordinat değerlerini buldurup getpixel yardımıyla hangi karelerinin içinin mavi olduğunun buldurulmasıyla kareler çizdirmeye çalıştık. Bu koddaki sıkıntı işaretlememiz gereken her kareyi kabul etmemesi. Altta ray casting kullanırken aldığımız çıktı (1) ve son tercihimiz olan getpixel çıktısı (2) yer almakta.



(1) (2)

Birim kare hesabı için ise yine graphics.h fonksiyonu olan getpixel kullanarak kareleri saydırdık.

Hesaplanan birim kareler sayesinde optimum alan-kare sayısı durumu buldurmayı basit bir algoritmayla başardık.

Bu algoritma sonucun doğruluğunu etkileyen birkaç hataya sahip. Bu hataların en büyüğü şekilleri ayırmadan tek bir düzenli çokgenmişçesine hesaplama yapıyor olması. Bu hatayı düzeltmek için çalışmalar yaptık ancak daha optimize bir sistemi bu dilde ve bu grafiklerde kullanmayı başaramadık. Yazdığımız bu algoritma sayesinde tam istenilen şekilde olmasa da genel anlamda alanın içine yerleştirilen ölçekli kareler, optimum değeri bulmamızı sağlıyor.

Sonrasında kullanıcıdan istememiz gereken girdileri aldırdık. Bizden istenen toplam platform sayısı, toplam sondaj maliyeti, toplam maliyet ve kar durumunu verilen yöntemleri kullanarak buldurduk.

Aşağıda çıktımız görülmektedir:

```
Toplam Platform Sayisi: 2
toplam sondaj maliyeti: 1600
toplam platform maliyeti: 12
toplam maliyet: 1612
kar durumu: 1388
```

IV. DENEYSEL SONUÇLAR

Karelere bölme işlemi sırasında fazlasıyla zorlanmamızın ardından öncelikle konu hakkında daha önce yazılmış bir verinin varlığını araştırmaya karar verdik. Araştırmalarımızın sonucunda "tiling problems" denilen bir problemin varlığından haberdar olduk. Bu problemin çözüm grafiklerinin bizim problemimize benzerliği bizim bazı açılardan bu problem hakkında yazılmış kodlardan faydalanmaya itti ancak kendi kodumuzda bu konuda bir sonuç alamadık.

Aynı konuda araştırma yaparken dancing links ve exact cover denilen diğer konu başlıklarıyla karşılaştık bu konuları da yardımcı olabilmesi adına enine boyuna inceledik ve kodlarımıza entegre etmeye çalıştık fakat bir sonuç alamadık.

En uygun platform değerinin hesaplanması için yaptığımız araştırmalar sonucunda greedy algoritmasıyla karşılaştık bunu kendi kodumuza uyguladık ve doğru sonuç aldık.

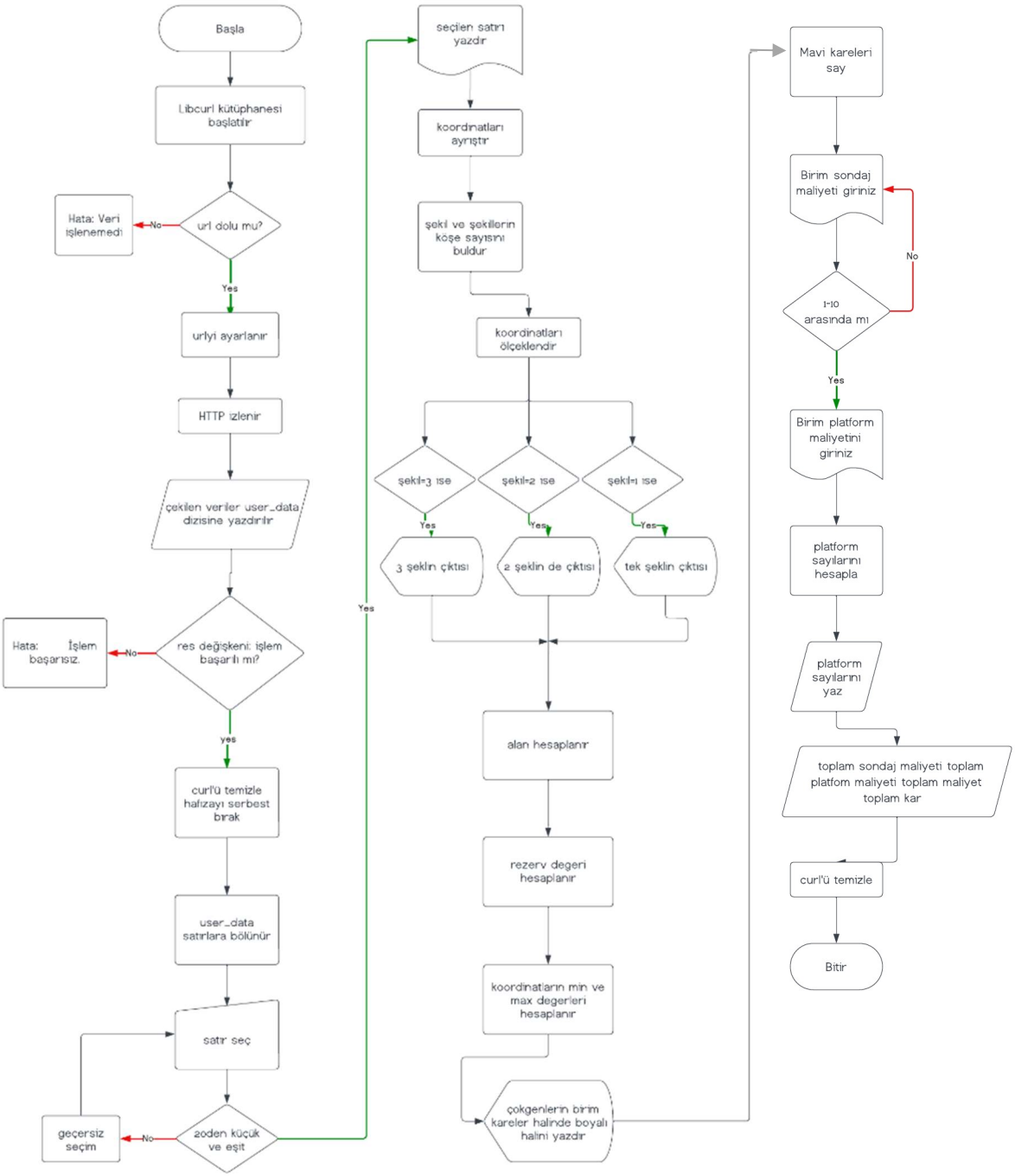
Doğru platformlara ayırma işlemi için matematiksel bir bölümleme olan pick's algorithm kullanabileceğimizi düşündük fakat bu konuyla ilgili hem kaynak bulmakta hem de kodumuzu buna uyarlamakta fazlasıyla zorlandık bu nedenle kodumuzdan çıkarttığımız diğer kısımlara katıldı.

V. SONUÇ

Projenin isterlerinden:
Programda iki ayrı çizim yapılmalıdır.
İlk çizimde kullanıcıdan istenen satırdaki noktaların oluşturduğu rezerv bölgesi çizdirilerek kullanıcıya görsel olarak gösterilmelidir.
Çizdirilen alanların rezerv değer miktarı hesaplanarak kullanıcıya gösterilmelidir.
İkinci çizimde karesel alanlara ayrılmış rezerv alanı çizdirilerek kullanıcıya görsel olarak gösterilmelidir.
İkinci çizim sonucunda kullanıcıya toplam platform sayısı toplam sondaj sayısı, toplam platform maliyeti, toplam sondaj maliyeti, toplam maliyet (=toplam sondaj maliyeti + toplam platform maliyeti) ve kâr miktarı (=rezerv değeri-toplam maliyet) kullanıcıya gösterilmelidir.
Maddeleri başarıyla tamamlanmıştır.

KAYNAKLAR

[1] https://coderspace.io/blog/greedy-algoritmasi/
[2] https://www.geeksforgeeks.org/boundary-fill-algorithm/
[3] https://www.geeksforgeeks.org/flood-fill-algorithm-using-c-graphics/
[4] https://www.geeksforgeeks.org/number-of-ways-to-go-from-one-point-to-another-in-a-grid/
[5] https://www.geeksforgeeks.org/program-to-print-a-hollow-triangle-inside-a-triangle/
[6] https://www.geeksforgeeks.org/sierpinski-triangle-using-graphics/
[7] https://www.geeksforgeeks.org/area-of-a-polygon-with-given-n-ordered-vertices/
[8] https://sogrekci.com/ders-notu/python-ve-bilimsel-hesaplama/42-grafik-cizimi/
[9] https://www.geeksforgeeks.org/tiling-problem-using-divide-and-conquer-algorithm/
[10] https://mathigon.org/task/paths-on-a-grid
[11] https://math.mit.edu/~rstan/papers/tilings.pdf
[12] https://www.mdpi.com/1999-4893/15/2/65
[13] https://www.programmingsimplified.com/c/graphics.h
[14] https://www.chegg.com/homework-help/questions-and-answers/tromino-puzzle-tromino-accurately-right-tromino-l-shaped-tile-formed-three-1-times-1-squar-q119312634
[15] https://en.wikipedia.org/wiki/Dancing_Links
[16] https://en.wikipedia.org/wiki/Exact_cover
[17] https://mathigon.org/task/picks-theorem
[18] https://stackoverflow.com/questions/31628687/install-curl-in-dev-c
[19] https://curl.se/
[20] https://sourceforge.net/projects/orwelldevcpp/
[21] https://www.youtube.com/watch?v=Lhbu6KpTdCc
[22] https://www.youtube.com/playlist?list=PLD5D5Hj95BCFid63gy2VtVBXJZQXuw12R
[23] https://github.com/SagarGaniga/Graphics-Library/blob/master/winbgim.h
[24] https://home.cs.colorado.edu/~main/bgi/doc/
[25] Dahlke, Karl. "Shoelace Formula". Retrieved 9 June 2008.


Başla
Libcurl kütüphanesi başlatılır
url dolu mu?
No
Hata: Veri işlenemedi
Yes
urlyi ayarlanır
HTTP izlenir
çekilen veriler user_data dizisine yazdırılır
res değişkeni: işlem başarılı mı?
No
Hata: İşlem başarısız.
yes
curl'ü temizle hafızayı serbest bırak
user_data satırlara bölünür
satır seç
geçersiz seçim
No
2'den küçük ve eşit
Yes
seçilen satırı yazdır
koordinatları ayrıştır
şekil ve şekillerin köşe sayısını buldur
koordinatları ölçeklendir
şekil=3 ise
şekil=2 ise
şekil=1 ise
Yes
Yes
Yes
3 şeklin çıktısı
2 şeklin de çıktısı
tek şeklin çıktısı
alan hesaplanır
rezerv değeri hesaplanır
koordinatların min ve max değerleri hesaplanır
çokgenlerin birim kareler halinde boyalı halini yazdır
Mavi kareleri say
Birim sondaj maliyeti giriniz
1-10 arasında mı
No
Yes
Birim platform maliyetini giriniz
platform sayılarını hesapla
platform sayılarını yaz
toplam sondaj maliyeti toplam platform maliyeti toplam maliyet toplam kar
curl'ü temizle
Bitir