Programlama Laboratuvarı 1. Proje Ödevi

1. Esra Kurt
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
esrakurt221@gmail.com

2. Zeynep Tandoğan
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
zeyneptandoğan03@gmail.com

***Özetçe*—Bu rapor, Kocaeli Üniversitesi Programlama Laboratuvarı 1 dersinin 1. proje ödevi için hazırlanmıştır.**

Anahtar Kelimeler — C programlama,SDL2/SDL.h

I. ÖZET

Bu proje; C programlama dili ve grafik kütüphanelerinden SDL kütüphanesi kullanılarak denizlerde doğal kaynak arama ve çıkarma işlemlerini gerçekleştiren bir şirketin maksimum kar elde etmesini sağlayan bir program geliştirmek amacıyla yapılmıştır. Dokümanda projenin tanımı, kullanılan yöntem, kod bilgisi ve yararlanılan kaynaklara yer verilmiştir.

II. GİRİŞ

İki aşamadan oluşan bu projenin ilk adımında kullanıcıdan hangi satırdaki koordinat noktalarının çizdirileceği, birim sondaj maliyeti ve birim platform maliyetini girmesi, sondaj operasyonlarının yapılacağı bölgelerin belirlenmesi adına verilecek URL'den koordinat verilerinin alınması ve iki boyutlu kapalı şeklin veya şekillerin çizdirilmesi istenmektedir. Web sayfasından verileri okumak için bir HTTP isteği gerçekleştirilmelidir. Daha sonra çizilen kapalı alanın/alanların yüzey alanının hesaplanması ve rezerv miktarı belirlenmelidir. Kaynak rezerv değerinin hesaplanacak toplam alanın 10 katı olduğu verilmiştir.

2. Aşamanın adımlarında ise ilk olarak kapalı alanı belirli boyutlardaki düzgün karesel parçalara bölmek için bir algoritma geliştirilmelidir. Bu algoritma verilen kısıtlara uygun olmalıdır. İlk kısıt: "Rezerv bölge sınır çizgilerinin her içinden geçtiği ve içerisinde kalan tüm alanlarda sondaj faaliyeti yapılacaktır." Birim sondaj maliyeti kullanıcı tarafından 1 ile 10 arasında değer alabilecek bir değişken parametre olmalıdır. İkinci kısıt: Çıkarılan kaynakların depolanması maksadıyla maliyetleri her karesel alan için özdeş olan, bu karesel alanlarda yalnızca bir tane kurulabilen ve birim maliyeti yine kullanıcı tarafından belirlenen platformlar kurulmalı ve en optimal bölünme hesaplanmalıdır. Kısıt 3: Bölümlenen karesel alanların boyutu 1x1, 2x2, 4x4, 8x8, 16x16 lık alanlardan biri olmalıdır. Sınır komşusu olan her karesel alanın boyutu birbirinin ya bir küçüğü ya da bir büyüğü olmalıdır. Ve farklı boyutlardaki karesel alanlar farklı renklerle boyanarak gösterilmelidir. En son raddede ise toplam sondaj sayısı, toplam sondaj maliyeti, toplam platform sayısı, toplam platform maliyeti, toplam maliyet ve kar değerini kullanıcıya göstermek gerekir. Toplam maliyet; toplam sondaj maliyeti ve toplam platform maliyetinin toplamıyla, kar miktarı ise kaynak rezerv değerinden toplam maliyetin çıkarılmasıyla bulunacaktır. Bu hesaplamalar algoritmanın sonucuna dayalı olacaktır.

III. YÖNTEM

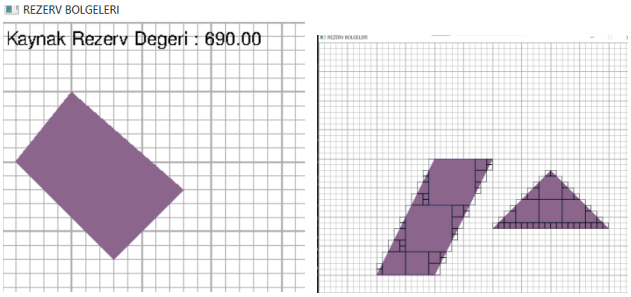
İlk olarak hangi grafik kütüphanesini ve hangi geliştirme ortamını kullanacağımıza karar verdik. SDL kütüphanesinde hem grafik çizebilmek hem de diğer grafik kütüphanelerinin URL'den bilgi almak için ek olarak Curl ve benzeri kütüphanelere ihtiyaç duymasının tam aksine kendi içinde bulundurduğu "SDL_net.h" kütüphanesiyle verileri webden alabilme özelliğine sahip olduğunu öğrendik ve tercihimizi SDL2/SDL.h kütüphanesinden İDE tercihimizi de Visual Studio Code'dan yana kullandık. Diğer geliştirme ortamlarının bilgisayar işlemcilerinin 32 veya 64 bit olma durumuna göre birtakım sorunlar yarattığına tanık olduk. Bu durum bizi Visual Studio Code'a yöneltti.

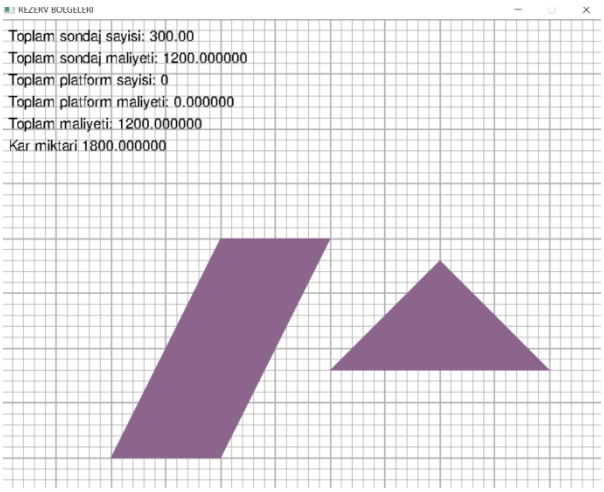
Daha önce grafik kütüphanelerini kullanmamış olduğumuzdan kaynaklar taradık ve araştırmalar yaptık. Kullanacağımız kütüphanenin işlevlerini iyice kavramak adına kütüphaneye dair pdfler okuduk, yardımcı videolar izledik ve örnek kodlar inceledik. Bu aşamadan sonra kafamızda yavaşça oturan kodlar üzerine çalışmalara başladık. Ardından koordinatların ve çizilebilecek çokgenlerin dizilerinin tutulabildiği iki yapı kurduk. Sonrasında ise gerekli fonksiyonları ve kapalı alan/alanların boyanması işlevi için gerekli olan çarpışma işlemlerini ve SDL kütüphanesinin birden fazla kullanacağımız pencere oluşturma, renderlama ve font işlevlerini tanımladık.

SDL kütüphanesinde kullanıcıdan girdileri direkt alabildiğimiz scanf() fonksiyonu bloklu olduğundan dolayı bu sorunu nasıl aşabileceğimizi araştırdık. SDL'nin klavye ve fare olaylarını, basışları algılayan, bunları kontrol etmeyi döngülerle sağlayan "event" işlevini öğrendik ve kullandık. Bu kısmı if-else ve switch-case gibi akış kontrol yapıları ile destekledik. Sonrasında olayları event ile kontrol eden döngünün içine arka planı hazırlayan, gerekli isterleri sağlayan, metinleri tutan, konumlarını ve renklerini belirleyen ve kullanıcıdan girdileri alan kısmı yazdık. Daha sonra tanımladığımız olay başlatma ve bitirme fonksiyonlarını, arka plan çizdirme fonksiyonunu(ki bahse konu bu grid-ızgara yapısını oluşturmak için tıpkı matris yapısını destekleyen iki boyutlu dizi, döngünün içinde iç içe döngüleri kullandık.) metin yazdırma, webden veri alma, çokgenleri çizdirme, alan hesaplama(burada birim kare saydırma yönteminden ziyade

matematiksel hesaplama yapmayı tercih ettik.) boyamayı yaptıracak çarpışma fonksiyonlarını çağırdık.

IV. DENEYSEL SONUÇLAR





V. SONUÇ

Sonuç olarak ekran çıktısı; alınan koordinatlarla kapalı bir şekil çizen yani rezerv bölgelerini belirleyen maksimum kar hesaplanması için optimal bölümlenmiş alan/alanlar olan, kaynak rezerv miktarıyla birlikte maliyetlerini ve karı hesaplayan bir program geliştirdik. Bölümlemede şart koşulan sınır komşularının ardışık boyut olma ve farklı renk kullanma kısıtlarını maalesef gerçekleştiremedik. Bu projeyle birlikte grafik kütüphanelerini kullanmayı ve bunu özellikle görsel ve grafik uygulamaları için çok nadir tercih edilen C programlama dilinde kullanmış olmak gibi spesifik ama bi' o kadar da iyi bir tecrübe edindik.

VI. KOD BİLGİSİ

struct Polygon
SDL_Point points[ ],int count;
struct AllPolygons
Polygon polygons[ ],int count;
struct Square
int size,int x.int y;
struct Square
int size; int x; int y;

SDL_Window *p_window;
SDL_Renderer *p_renderer;
TTF_Font *p_font1;

main
-initialize fonksiyonuna git:
-tanımlamaları yap.
-for döngüsü tanımı:
-done adlı değişken tanımla ,
-kullanıcı çıkış yapana kadar çalışmaya devam et
-Olay İzleme Döngüsü:
-Olay Türüne Göre İşlem:
-pencere kapatılmak isteniyosa SDL_QUIT e git done değişkenini true yap
-kullanıcı metin girdisi yaptıysa SDL_TEXTINPUT a git girdiyi input_text dizisine ata.
-kullanıcı ENTER tuşuna bastıysa:
-diziyi float türüne dönüştür ve input_val e ata.
-eğer selected_line 0'a eşit ya da küçükse girilen değeri selected_line değerine ata.
-değilse ve sondaj maliyeti 10'dan büyük veya 1'den küçükse girilen değeri bu değere ata.
-değilse ve platform maliyeti 0 dan küçükse girilen değeri bu değere ata.

-draw_background fonksiyonuna git arka plan çiz:

-seçilen satır 0'a eşit veya küçükse:
-konum ve renk belirle
-SDL_StartTextInput() başlat
-print_text fonksiyonuna git
(fonksiyona metin konum ve renk bilgilerini yolla)

-sondaj maliyeti istenen aralıkta değilse:
-konum ve renk belirle
-SDL_StartTextInput() başlat
-print_text fonksiyonuna git istek metnini ekrana yazdır.

-platform maliyeti istenen aralıkta değilse:
-konum ve renk belirle
-SDL_StartTextInput() başlat
-print_text fonksiyonuna git istek metnini ekrana yazdır.

- Değilse ve ekran değişkeni 0'a eşitse:
-get_informataion_from_web fonksiyonuna git
(seçilen satır değerini ve all_polygons struct yapısını yolla)
-return true ise ekran(screen) değişkenini 1 yap kalvye girişini kapat.
-return false ise seçilen satırı(selected_line) 0 yap.
-değişken tanımla
- for döngüsü her bir polygon için:
-rengi belirle

-draw_filled_polygon fonksiyonuna git
(render,polygon, renk değeri yollanır)

-polygon_area fonksiyonuna git
(ilgili polygonu yolla)

-rengi belirle
-İç içe for döngüsüyle tüm noktaları gez
-dörtgen tanımla
-koordinatları belirle

-detect_collision_poly_rect_without_corners fonksiyonuna git
-return true ise çizgileri çiz

-eğer ekran değeri 1 ise
-kaynak rezervi hesaplayıp karakter dizisine ata
-konum renk belirle
-print_text fonksiyonuna git ve değeri ekrana yazdır.

-eğer ekran değeri 2 ise
-calculate_collision_map fonksiyonuna git
(all_polygons struct yapısını ve collision_map dizisini alır

-for döngüsüyle tüm polygonları tara
-int square_count = 0;
-Square squares[MAX_SQUARE_IN_A_POLYGON];
-cover_shape fonksiyonuna git
-for döngüsüyle kare sayısı kadar:
-her boyutta kare için farklı renkler ayarla
-x,y koordinatlarının ve dif değişkeninin(boyut) değerlerini hesapla
-SDL_Rect yapısıyla bir rect tanımla
-rect(kare) yapısını ekrana çizdir

-eğer ekran değeri 3 ise
-tanımlamaları yap
-print_text fonksiyonuyla değerleri ekrana yazdır

-ekranı güncelle
-40ms bekle ve sil
-eğer ekran 1 se
-ekranı 2 yap 5 sn bekle ve sil

-eğer ekran 2 ise ekranı 3 yap

-deinitialize fonksiyonuna git
return 0;

initialize()
-SDL kütüphanesini başlat
-pencere ve renderer oluştur
-font dosyasını aç.

deinitialize()
-font ayarlarını kapat
-pencereyi kapat
-font dosyasından çık
-SDL_Net ve SDL ayarlarını serbest bırak

draw_background()
-renk ayarla pencereyi o renkle temizle.
-çizgi rengini belirle.
-PARCEL_UNIT_SIZE kadar aralıklarla yatay ve dikey çizgiler çiz.
-koordinatı 4 ün katı olan her çizgide ölçeği 2 yap

print_text(metin ,konum ,renk)
-metin yüzeyini ve metni oluştur.
-yüzeyin ölçülerini belirle
-dokuyu belirtilen konum üzerine çiz
-gereksiz dokuyu ve yüzeyi bellekten temizle

get_informataion_from_web(satır, p_polygons)
- HTTP yanıtının alınacağı bir karakter dizisi oluştur.
-HTTP isteğini (GET isteği) tanımla.
-sunucu adresini ve port numarasını saklayan IPaddress yapısını tanımla.
-sunucu IP adresini çöz
-TCP soketi oluştur sunucuyla iletişim kur
- sunucuya HTTP isteği gönder ve gönderilen bayt sayısını sent_length değişkenine kaydet.
-recv_length 0 dan büyük olduğu sürece döngüyle gelen veriyi al ve recv_data dizisine ekle.
-soketi kapat bağlantıyı sonlandır.
-yanıtın başlangıç adresini bul *p_start_address'e ata.
-adres kontrolü yap eğer NULL ise return false;.
-while döngüsüyle belirtilen satıra kadar yanıtı atla
-satırın sonunu bulup sonlandırma işareti koy.
-poligon ve köşe sayısı uygun olduğu sürece döngüde kal:
-x ve y değerlerini bul ve ata
-ilk nokta ve son nokta aynıysa diğer çokgene geç
-return true;

draw_filled_polygon(render, polygon, renk)
-tanımlamaları yap
-noktaları gez en üst y değerini bul
-sol ve sağ noktayı tanımla
-başlangıç ve bitiş noktasını ayarla
-sağ ve sol eğimleri hesapla
(Eğim, x değişimini yükseklik değişimine bölerek hesaplanır.)
-çizimi rengini ayarla
-ana döngüye git işlenmemiş köşeler varsa devam et
-noktaları birleştirerek çizgilri çiz
-Başlangıç ve bitiş noktalarını, eğimlere göre güncelle.
-sağ ve sol köşeye ulaşıldığında yeni üst köşe belirle eğimi güncelle
-iç içe döngülerle her seviyede en üste gelene kadar çizim yap
-return true;

polygon_area(polygon)
-area ve j değişkenini tanımla. j değişkeni son kenarın indeksi

-for la her kenarı tara
-formülle alanı hesapla, area değişkenine ata her döngü sonrası area değerlerini topla.
-area/2 değerinin mutlak değerini döndür.

calculate_collision_map(p_pollygons, p_collision_map[])
-for döngüsüyle tüm polygonları gez
-find_min_max_points fonksiyonuna git
(polygon ve int değişkenler yollandı)
-iç içe forla min x ve y den max x ve y koordinatlarına kadar gez
-dörtgen tanımla
-noktalarının koordinatlarını belirle
-detect_collision_poly_rect_without_corners fonksiyonuna git
(polygon ve dörtgen yollanır)
-eğer return true ise çakışma haritsına 1 olarak kaydet.

find_min_max_points(p_polygon, minx, miny, maxx, maxy)
-atamaları ilk elemana göre yap
-for döngüsüyle tüm noktaları gez
-if le büyüklük küçüklük karşılaştır
-min ve max değerlere ata

detect_collision_poly_rect_without_corners(p_polygon1, p_polygon2)
-her bir parceli köşelerden 1 pixelküçült
-detect_collision_poly_poly ifadesine git ve bu değeri döndür.

detect_collision_poly_poly(p_polygon1, p_polygon2)
-for döngüsüyle tüm köşeleri gez
-detect_collision_poly_line fonksiyonuna git
-eğer doğruysa return true;
-detect_collision_poly_point fonksiyonuna git
-eğer doğruysa return true;
-return false;

detect_collision_poly_line(p_polygon, nokta1, nokta2)
-for döngüsüyle tüm köşeleri gez
- hit=detect_collision_line_line fonksiyonuna git
-hit=return true ise
return true;
return false;

detect_collision_line_line(nokta1, nokta2, nokta3, nokta4)
-if koşulunda matematiksel hesaplamalarla kesişim var mı diye kontrol edilir.
-değer belirlenen aralıktaysa kesişim var return true;
-değilse return false;

detect_collision_poly_point(p_polygon, nokta)
-collision ddeğişkenini false olarak tanımla
-for döngüsünde tüm köşeleri gez
-SDL_Points yapısıyla sıralı iki köşeyi tut
- nokta iki kenarın y ve x koordinatları arasındaysa collision değerini tersine çevir
-return collision(çarpışma);

cover_shape(collision_map_for_one_poly[], p_squares, p_squares_count)
-tanımlamalarını yap
-for döngüsüyle farklı boyutlardaki kareleri sırayla ele al
-iç içe döngüyle çarpışma haritasını tara
-valid=1;
-iç içe döngüyle boyut kadar alan tara
-eğer çarpışma yoksa parcel değeri 1 değilse
-valid=0
-break(çık)
-eğer valid=0 sa
-break(çık)
- eğer valid 1 se
-karenin boyutunu kareler dizisine kaydet
-karenin sol üst köşesinin x koordinatlarını ata
-karenin sol üst köşesinin y koordinatlarını ata
-o boyuttaki kare sayısı ve toplam kare sayısını bir arttır
-for döngüsü
-kaplanan alanın değerini 0 yap
-bir sonraki kare kontrolü için boyut kadar geri gel

KAYNAKÇA

[1] https://www.tutorialspoint.com/http/index.htm
[2] https://www.scribd.com/document/363779277/SDL-Kitap
[3] https://github.com/catsocks/sdl-grid
[4] https://stackoverflow.com/questions/21560384/how-to-specify-width-or-point-size-in-sdl-2-0-draw-points-lines-or-rect
[5] https://www.geeksforgeeks.org/area-of-a-polygon-with-given-n-ordered-vertices/
[6] https://discourse.libsdl.org/t/sdl-net-http-request/25507
[7] https://stackoverflow.com/questions/40986826/how-to-draw-polygons-in-sdl
[8] https://github.com/jeffThompson/CollisionDetection/blob/master/CodeExamples/PolyPoint/PolyPoint.pde
[9] https://stackoverflow.com/questions/30818645/minimum-exact-cover-of-grid-with-squares-extra-cuts
[10] https://lazyfoo.net/tutorials/SDL/27_collision_detection/index.php
[11] https://www.jeffreythompson.org/collision-detection/poly-poly.php
[12] https://whatheco.de/2011/02/10/camelcase-vs-underscores-scientific-showdown/
[13] https://www.abecem.net/web/renk.html
[14] https://wiki.libsdl.org/SDL2/FrontPage
[15] https://www.geeksforgeeks.org/visual-studio-vs-visual-studio-code/
[16] https://youtube.com/@JacobSorber?si=FpQLSAZdv-mEPgdx
[17] https://youtube.com/@MikeShah?si=L6SlfIZJHz7UCmf3
[18] https://youtu.be/vRLIAdKMQK8?feature=shared
[19] https://youtu.be/DgpcRIK2uug?feature=shared
[20] https://sudo.ubuntu-tr.net/sdl-ile-oyun-programciligi
[21] https://wiki.libsdl.org/SDL2/Tutorials-TextInput
[22] https://en.wikipedia.org/wiki/Branch_and_price
[23] https://www.bilgigunlugum.net/prog/cprog/2c_degisken
[24] https://web.itu.edu.tr/kabak/dersler/EM302/pdf/TSP.pdf
[25] https://cimpress.com/cimpress-tech-challenge-winner-dimitar-blagoev/