# KOCAELİ ÜNİVERSİTESİ

# BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

# BLM210 PROGRAMLAMA LAB. II, 2018-2019

# PROJE 1

180202116 Mert BİLGİÇ [bilgic.mert44@gmail.com](mailto:bilgic.mert44@gmail.com)

ÖZET

Projemiz öncelikle kullanıcıdan herhangi bir veri almakadır.Tüm verileri txt dosyası içerisinden alınmaktadır.İlerlemeye devam edersek harita.txt içinden gerekli veri akışı başlamaktadır.İlk başta oyunda olan kötü karakteri dosyada okunup devamında ise gerekli dizilerin içine aktarım yapılmaktadır.Karakter okuma işlemleri bittiğinde labirent oluşturmamız için verilen harita gerekli diziye aktarılmaktadır.AnaEkran adlı classımız bir JFrame’den miras almaktadır.Bu kısımda grafik oluşturmak için gerekli fonksiyonlar kullanılmıştır.Projenin dokümanımda verilen tüm gerekli objeler oluşturulmuştur.Karakter sınıfından objeler oluşturulmuş objelerimizle karakterlerimiz kontrol altına alınmıştır.Tüm karakterler Karakter sınıfında türediği için karaktere dönüşüm yapılabilir.Bunu kullanmak için bir sınıf üzerinde objeler return yapılmaktadır.GrafikKullanımı adılı sınıf JPanel’den türetilmiştir.Bu sınıfta tüm grafik işlemlerimiz yapılmaktadır.Projenin amaçlarından biri olan veri yapıları dersinin pekiştirilmesi için kullanılan en kısa yol bulma algoritması rekürsif fonksiyon’ların kullanılması ile 0 ve 1 ile karşılaşılma durumlarına göre şekillenmektedir. Fonksiyonun içinde veriler tutulup grafik kısmında kullanılmaktadır.Karakter sınıfı içerisinde kullandığımız fonksiyon DarkVeader sınıfı için geçersiz kalınıyor.Çünkü DarkVeader sınıfı duvarların içinden geçebilmektedir.Eğer biz tüm haritayı 1 şeklinde fonksiyona yollarsak sonsuz sürekli 1 ile karşılaşacağı için örneğin ilk koşulla karşılaşıp sürekli öne gider daha sonra sol,sağ ve alt işlemleri yapar bu de bize en kısa yolu bulma konusunda problem çıkartır.Bu yüzden DarkVaedar içinde aynı ismi bir fonksiyon override edilmiştir.Bu fonksiyon daha basit bir mantıkla çalışmaktadır.

***Anahtar Kelimeler:****Rekürsif Fonksiyon,JPanel Kalıtım,Obje*

GİRİŞ

Bu kısımda projenin konusu anlatılmıştır.

Projede öncelikli amacımız aldığımız nesneye dayalı programla ve veri yapıları derlerindeki bilgilerimizi pekiştirmektir.Dosyadan okuduğumuz verileri gerekli işlemlere parçalayarak gerekli dizilere atadıktan sonra aldığımız verileri gerekli yerlerde kullanmamız gerekmektedir.Bu bilgilerle gerekli objeler oluşturulup karakterin oyuna sürülmesini sağlamak haritanın çizdirilmesi karakterlerin başlangıç noktasının ayarlanması karakterler isimlerine göre resimlerin belirlenmesi gibi işlemlerde kullanılmıştır.Sınıflar arasında kalıtım yapılarak nesneye yönelimli programlama hakkında bilgi seviyemiz kontrol ettikten sonra bu yapıları uygun bir şekilde kullandık.Projenin en önemli noktalarından biri dosyada okuduğumuz verileri dağıttıktan sonra başlamaktadır.Kötü karakterlerin iyi karakterleri takip etmesi ve takip ederken en kısa yolu kullanmalarını sağlamaktır.En kısa yolu bulmak için veri yapıları dersinde gördüğümüz rekürsif fonksiyonları kullandık.Bu yapı sayesinde kullandığımız algoritma kendi içinde bulduğu kısa yolu kaydederek daha kısasını bulana kadar saklamaktadır.Buradan aldığımız koordinatları her tur başı güncelleyerek kötü karakterin iyi karakteri takip etmesi ve aynı zamanda en kısa yolun boyanmasını sağladık.Devamında iyi karakterin yakalanınca canının azalmasını sağladık.Her turda iyi karakterin yakalanıp yakalanmadığını veya bitiş noktasına ulaşıp ulaşmadığını kontrol ederek can sistemini ve oyun bitişini ayarladık.

YÖNTEM

Bu kısımda izlediğimiz yolu ve çalışmalarımızı anlatmaktayız.

Projede izlediğimiz yolu özetlemek gerekirse öncelikli amacımız projeyi nesneye dayalı programlama yapısına uygun şekle getirerek en verimli şekilde kod yazmaktır.

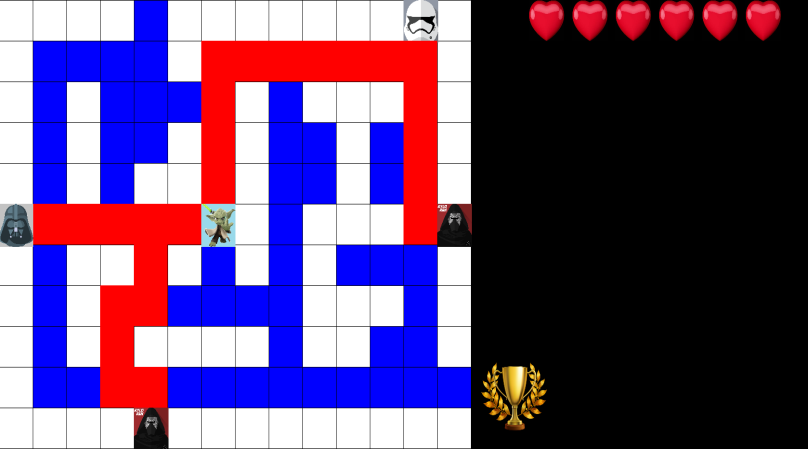
İzlediğimiz yolu sırasıyla şu şekilde aktarabiliriz:

Projeyi inceledikten sonra genel yapıyı oluşturduk.Genel yapıyı oluşturduktan sonra karakterleri bir sınıf içinde üreterek Karakter sınıfına yolladık.Burada üst sınıfın alt sınıftan obje oluşturmasından yararlandık.Bu kısmı böyle yapmamızın sebebi nesneye dayalı programlamayı daha iyi anlayıp alternatif yolları denemekti.Bu yapıyı kullanırken karakterleri daha rahat kullandığımızı ve gereksiz obje oluşumlarını engellemiş olduk.Elimizden geldiğince konuyu kavrayabildiğimizce işleri fonksiyonlara yaptırmaya çalıştık.Kullandığımız en kısa yolu bulma algoritması DarkVeader için uygun değildi burada bunu override ederek yeni bir yöntem daha öğrenmiş olduk her ne kadar objeler karakter sınıfına aktarılsada sıra DarkVeader’ a geldiğinde kendi içindeki override edilmiş fonksiyonu çağırdı.Projenin dokümanında verilen hiyerarşik yapıyı kurarak ve devamında isterleri tamamladık.

SONUÇ

Sonuç olarak projede nesneye dayalı programlamanın temel işlemleri pekiştirildi aynı zamanda veri yapılarında öğrendiğimiz bilgileri bilgisayar ortamında test edip tecrübe edinmiş olduk.Grafik kütüphanesi ile biraz konsoldan uzaklaşarak verilerimizi görsel olarak göstermenin ne kadar etkileyici olabileceğini gördük.

DENEYSEL SONUÇLAR





Deneysel sonuçta görüldüğü gibi iyi karakter olan en kısa yol karelerin kırmızı ile boyanması ile belirtilmiştir.Görselde olmayan durumlarda iyi karakterin hareketi sırasında karakterlerin hareket özelliklerine göre hareket ettiği görülmüştür.Takip ve yakalama işlemi sorunsuz şekilde çalışmaktadır.

KABA KOD

İşlemlere ilk başladığımız sınıfımız AnaEkran sınıfı bu sınıfta txt dosyasından okuduğumuz verileri parçalıyoruz.Öncelikle karakter ismi ve giriş bilgilerini düzene uygun tutarak ilerliyoruz.Eğer dosyayı okurken iyi karakterlerden birinin ismine denk gelirsek ayırma için if(karakter[i].equals("MasterYoda")||karakter[i].equals("LukeSkywalker")) koşulunu kullanıyoruz burada bir değişkene iyi karakterin ismi atanıyor.İlerleyen kısımlara haritadaki duvar bilgisini txt dosyasından çekiyoruz.Aynı zamanda AnaEkranda ekran.setResizable(true);//Ekranın genişleyebilir olmasını engelliyoruz ekran.setFocusable(false);//JFrame odaklanmasını engelliyoruz. ekran.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); ekran.setSize(1920,1000); gibi fonksiyonlarla JPanel ve JFrame arasında olacak olayları ayarlaşmış oluyoruz.Ve en çok işin olduğu GrafikKullanımı sınıfına işi aktarıyoruz. public void oyunBittiMi()---public void kazandiMi()---public void OyunuSıfırla() gibi fonksiyonlarla oyunun yönetimini sağlıyoruz. public void paint(Graphics g) grafik kısmı belirttiğimiz fonksiyon yardımı ile sağlanıyor.Burada haritadan çektiğimiz verileri grafik ekranında oluşacak şekilde hazır özelliklerle sağlıyoruz. g.fillRect(70\*j,70\*i,70,70); g.setColor(Color.blue); gibi özellikler ile gerekli renk işlemlerini sağlayıp txt den çektiğimiz haritanın oluşmasını sağlıyoruz. Constructor içinde karakterlerin simgelerini yüklerken aynı zamanda txt den okunan iyi karakterin oluşturulmasını sağlıyoruz.Karakter oluşumu KarakterUret adılı sınıfımız içinde gerçekleştirilmektedir.Buradaki fonksiyonlara göre işlemler yapılmaktadır.Bu kısımda iki fonksiyonumuz bulunmaktadır. Karakter karakter\_uret(String karakter\_adı) Karakter karakter\_uret2(String karakter\_adı,int indis) ilk fonksiyonumuz iyi karakterin objelerini return yaparken ikinci fonksiyonumuzda kötü karakterlerin return işlemini yapıyoruz burada ayırmamızın sebebi kötü karakterleri bir static değişken içinde tutup belli indislerin içinde barındırmamız.GrafikKullanımı sınıfımız ile devam edicek olursak txt den okuduğumuz verilere göre karakter oluşumu oyun boyu devam etmektedir.Bu oluşumu karakter isimlerine göre for(c=0;c<z;c++){

//Karakterler dizi şekilinde kullanılarak gereksiz obje oluşumunun önüne geçildi.

kötüK[c]=uret.karakter\_uret2(Karakter[c],c);

verilen örnekteki gibi yaptık. return new DarthVader(AnaEkran.kapiY[indis],AnaEkran.kapiX[indis],"DarthVader","kötü"); şeklinde oyun bitene kadar return işlemleri yapılıyor. AnaEkran.kapiY[indis],AnaEkran.kapiX[indis] şurada kullandığımız static değişkenler sayesinde oyun boyu her obje oluşturulduğunda ilerlemiş konumlarıdan devam edilmesi sağladı.Karakter yaklanandığında public void OyunuSıfırla(){

for(int p=0 ;p<z;p++){

if(AnaEkran.kapi[p].equals("A")){

AnaEkran.kapiX[p]=0;

AnaEkran.kapiY[p]=5;

}

Örnekte verilen fonksiyon ile her karakterin başlangıç noktasına dönmesi sağlanmıştır.Bu şekilde oyunun devamlılığı bir fonksiyon ile sağlanmış bulunmaktadır. . public void oyunBittiMi()---public void kazandiMi() fonksiyonları ile koordinat kontrolü yapılarak durumlar ele alınmıştır. public void keyPressed(KeyEvent e) {

int tus =e.getKeyCode();

fonksiyonu klavyeden gelen tuşlar alınmış ve koşula uygun ve matriste sıfıra denk gelmiyorsa hareket etmesi sağlanmış.repaint() fonksiyonu ile güncelleme işlemiş yapılmıştır.Asıl işin en önemli kısmına gelirsek en kısa yol algoritmasını rekürsif bir yapı kullanarak haletlik.

public int findShortestPath(int mat[][], int[][] ziyaret,int i, int j, int x, int y, int min\_adım, int adım)

adlı fonksiyonumuz verileri alıp matrisin içinde ziyaret ettği kısımları ayırarak en kısa yolu rekürsifbir yapı ile bulmaya çalışıyor.

if (alanKontrol(i, j - 1) && güvenliMi(mat, ziyaret, i, j - 1)) { gibi 4 koşul ile harita üzerinde gezerek her yeni bulduğu min değerde veri tuttuğu kısmı güncelleyerek ilerliyor.

List<Integer>path2= new ArrayList<Integer>();

ArrayList’in içinde verilere tutularak daha sonra sa adlı değişkene aktarılıyor.Bu değişken ve Arraylist her tur başı sıfırlanarak gereksiz veri birikimin önüne geçiliyor.

AnaEkran.sa=""; AnaEkran.path.clear();

DarthVeader karakteri için bu fonksiyon kulanım dışı kalıyor tüm harita bilgilerini 1 yaptığımızda doğru olmayan sonuçlar elde etmeye başlıyoruz.Onun için daha basit bir algoritmayla çözüme gittik.İyi karakterin koordinatlarıyla farkını alarak karakter için kısa yolu bulan bir algoritma oluşturturduk.

f(KY>=0){

if(KY!=y1&&KY<y1){

KY++;

AnaEkran.path.add(KY);

AnaEkran.path.add(KX);

}

Genel hatlarıyla algoritmamız bu şekildedir.

BELLEK VE ZAMAN KARMAŞIKLIĞI

DarthVader’da kullandığımız algoritma ve diğer iki kötü karakter kullandığımız algoritma aynı değil bunun sebebi DarthVader’da duvar kavramının olmamasıdır.

**DarthVader karakterinin analizi:**

O((y2-y1)+(x2-x1))

iyi ve kötü karakterlerin koordinatlarının farkından oluşan adım kadar dönüyor.

**Diğer karakterlerin analizi:**

**Zaman Karmaşıklığı:**

O(M.N)

Kullandığım kodun analizi hakkında araştırmalara sonucunda net bir sonuca ulaşamadım.Eğer bu algoritmayı graflar uygularsak üstteki sonuca ulaşırız

Kaynakça

https://www.techiedelight.com/find-shortest-path-in-maze/

https://www.geeksforgeeks.org/shortest-path-in-a-binary-maze/

https://www.mobilhanem.com/graph-teorisi-bfs-algoritmasi/

https://www.techiedelight.com/lee-algorithm-shortest-path-in-a-maze/

<http://bilgisayarkavramlari.sadievrenseker.com/2011/11/04/labirentte-yol-bulma-kodu/>

https://www.youtube.com/watch?v=dyrvXiMumXc