# ŞİRİNLER

Metehan MART, M. Gürkan CİNGÖZ

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

metehanmart@gmail.com, gurkancingoz@yandex.com

# ÖZET

*Bu projede bizden şirinler temalı bir labirent oyunun yapılması istenmiştir. Oyuncunun kontrol edeceği karakteri seçebilmesi ,oyunda çeşitli objelerin(altın ve mantar) olması, düşman karakterlerinin oyuncunun kontrol ettiği karaktere en kısa yoldan gitmesi için Dijikstra’nın algoritmasının kullanılması ve bu yolun harita boyanarak gösterilmesi istenilmiştir.*

# **GİRİŞ**

Oyunun çalışması için önce metin belgesindeki harita bilgisini aldık ve haritayı yazdırmak için Graphics kütüphanesini kullandık. Oyuncunun kontrol ettiği karakterin hareket etmesi için belirli tuşlara basılıp basılmadığına bakmamız gerekiyordu. Bunu Al sınıfında yaptık. Belirlediğimiz tuşlardan herhangi birine basılması durumunda kullanıcının seçtiği karakterin pozisyon bilgisini değiştirdik ve bu değişikli ekranda göstermek için Board sınıfındaki paint methodunu kullandık. Oyuncun karakter seçimini oyuncu hareketini tamamladıktan sonra Main sınıfında yaptık. Düşman karakterlerinin haraketi ve en kısa yol bilgisi için Dijikstra’nın algoritmasını kullandık.

## **YÖNTEM**

Main sınfımızdaki main methodumuz main constructorını başlatıyor. Main constructorının içinde framemizi oluşturuyoruz.Board sınıfının objesi olan “board”ı başlatıyoruz. Kullanıcının karakter seçimi için jbuttonlar oluşturup onlara actionlistener ekliyoruz. Actionlistener sayesinde seçim değişkenine farklı değerler atıyabiliyoruz. Oluşturduğumuz jbuttonları framemize ekliyoruz. Sonra daha önce başlattığımız boardı ekliyoruz. Board başlatıldığında constructorı çalışıyor.Belirli değişkenler başlatılıyor ve paint methodunun her 25 ms’de bir çağrılıması için timer kullanılıyor. Burada Map sınıfınının objesi m başlatılıyor. Mapin constructorında çeşitli Imageler oluşturuluyor ve dosyanın okunması için readFile methodu bulunuyor. readFile methodunda whileın içinde satır satır txtden verileri alıp veri stringine atıyor. Veriden düşman bilgileri alınıp düşman arraylistinde tutuluyor. Veri int bir değer barındırıyorsa o satır Map arraylistine atılıyor. Board sınıfının paint methoduna dönecek olursak ilk olarak giriş kapılarını, şirinenin yerini, duvarları ve boş alanları yazdırıyor. Düşman karakterlerin başlama yerlerini giriş kapılarına göre set ediyor. Giriş kapılarını algılaması için switch case kullandık. Öncelikle en köşe çerçeveleri tespit edip oradaki 0 ve 1 lere bakıyor. Zaten maksimum 5 adet giriş olacağından(a, b, c, d ve şirine)ilk olarak ilk tespit edilen 1 e a sonrakine b sonrakine c sonrakine şirine sonrakine ise d yerleştiriliyor. Bu da switch case de koyduktan sonra başta 0 olarak ayarlanan değişken arttırılarak sıra sıra sağlanıyor.

yolAzman Azman karakterinin oyuncunun karakterine olan en kısa yol bilgisini tutar. Bu bilgiyi eklemek için grapShow kullanılır yani Dijikstra’nın algoritaması kullanılır. yolGargamel de yol azmanla aynı şekilde yol bilgisini alır yol bilgisi null olması durumunda program çökmesin diye if komutları hem yolAzman hem yolGargamel için kullanılmaktadır.

Zaman tutulması için kullandığımız yöntem zaten 25 ms’de bir yenilendiğinden bir değişken oluşturup bu değişkenin her paint yenilendiğinde artması sağlanarak zaman ölçümü sağlanmıştır. Örneğin: Bir altının 5 saniye sonra yok olması mı gerekli? Bu durumda bize paint metodunda bulunan time+=1 den 200 kez dönmesi gerekli (200\*25=50000MS=5 Saniye) bu şekilde bunları zaman gerektiren her şeye uyarlandı.

Hangi karakter kullanıldığını ve oyunun bitmesi için “secim” adında bir değişken kullandık. Bu değişken sayesinde eğer tembelle ilgili bir şey varsa secim=2’de eğer gözlüklü ile ilgili bir şey varsa secim=1’de gerçekleşiyor. Başlangıçta Mainden ilk seçimin 1 mi 2 mi olduğunu butonlar yardımıyla belirleyerek Board class’ına başlıyoruz. Daha sonra eğer oyuncu şirineye dokunursa bu değişkeni 3 yaparak kazanmam ekranı getiriyoruz. Ve aynı şekilde eğer oyuncunun puanı 0 veya altına düşerse de secim değişkenini 4 yaparak kaybetme ekranı getirerek programı sonlandırıyoruz.

Oyuncu klavyeden herhangi bir tuşa basarsa Al sınıfının içindeki keyPressed methodu çalışır. Bu methodun içinde düşman karakterlerinin nereye hareket edeceğine karar veren bir mekanizma bulunur. En kısa yol için Dijikstra’nı algoritması kullanılırken grapShow’un bize döndürdüğü değer kısa yolun tamamıdır ancak burada bizim sadece ikinci düğüme ihtiyacımız vardır. Bu ikinci düğümü bulmak için forun içinde ikinci düğümü bulan bir algortima kurduk.Dijikstran’ın çalışabilmesi için oyuncunun seçtiği karakterin kordinatı gereklidir. Bunun için seçim değişkenin durumuna göre hareket etmemiz gerekti. Seçim 1 eşitken Gözlüklü şirin 2’yken tembel şirinin koordinatları alındı.

Puanlamak için ise tutulan tembel şirin ile gözlüklü şirinin koordinatlarının altın, mantar, Gargamel veya Azman ile eşit olup olmadığına bakılmıştır. Puanı başta yukarı yazıp daha sonra değiştiği zaman setText metodunu kullanarak azaltılmış ya da arttırılmış hali olarak güncelledik.

Program sırasında gözlüklünün altının üzerinden atlayıp objeyi almama problemiyle karşılaştık. Çünkü direkt olarak gözlüklünün konumu değişip objenin konumuyla eşitlenmemesinden objeyi alamıyordu. Bu sorunu her tuş basışından sonra gözlüklünün gittiği konumun bir arkasını yani atladığı yeri tutarak çözdük. Örneği “W” tuşu basıldığında 2 yukarı giderken değişkende de 2 yukarının bir aşağısı yani önceki konumunun 1 yukarısı atladığı yer tutuldu. Eğer objenin konumu ve bu konum aynıysa da aynı işlemlerin gerçekleşmesi için if’in içerisinde veya kullanıldı.

## **DIJKSTRA KARMAŞIKLIK ANALİZİ**

Öncelikle GraphShow’un constructırı çalışıyor:

Burada ilk olarak iç içe 2 for döngüsü içerisinde new NodeWeighted ile Node Weighted’ın constructırı çalışıyor bu constructıra baktığımız zaman karmaşıklığını döngü olmadığından 1 olarak görüyoruz bu durumda yalnızca iç içe 2 adet döngüden karmaşılığımız n2 ‘dir.

2. İç içe forda GraphWeighted’ın addEdge metodu çalışmaktadır. Bu metotta da addEdgeHelper adlı metod çalışmakta ve bu metodun da bir adet for döngüsü bulunmaktadır. O forun içeriğinin ise 1 karmaşıklığı vardır. Bu durumda 2. iç içe forun karmaşıklığı n2\*n den n3 olmuş oldu.

3.,4. Ve 5. İç içe forlar 2. İç içe forla aynıdır bu nedenle toplam 4\*n3+n2 gibi bir karmaşıklık elde ettik. Bu da Big O(n3) ‘e eşittir.

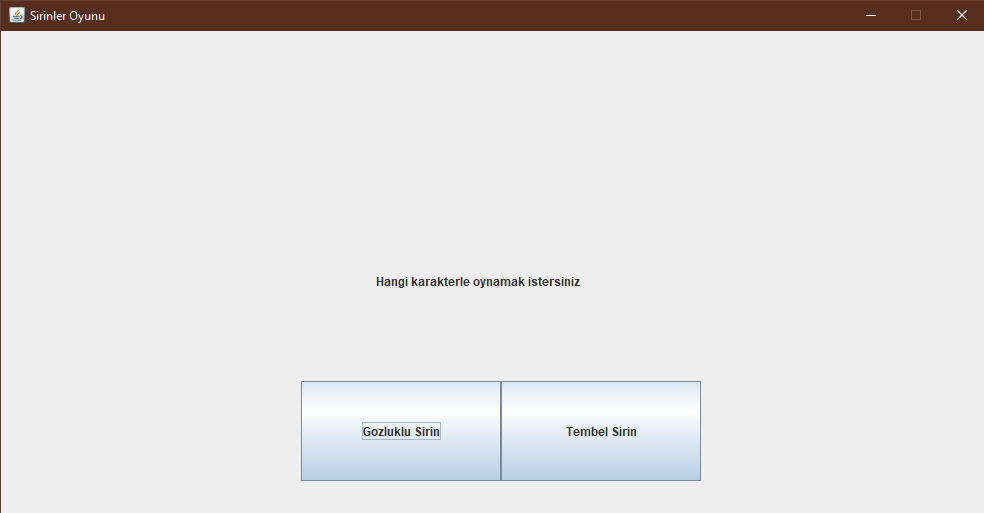
Daha sonra GraphShow class’ının Show metodu çalışır: Bu metodda yalnızca GraphWeighted class’ını DijkstraShortestPath metodu içine bazı parametrelerle return edilir.

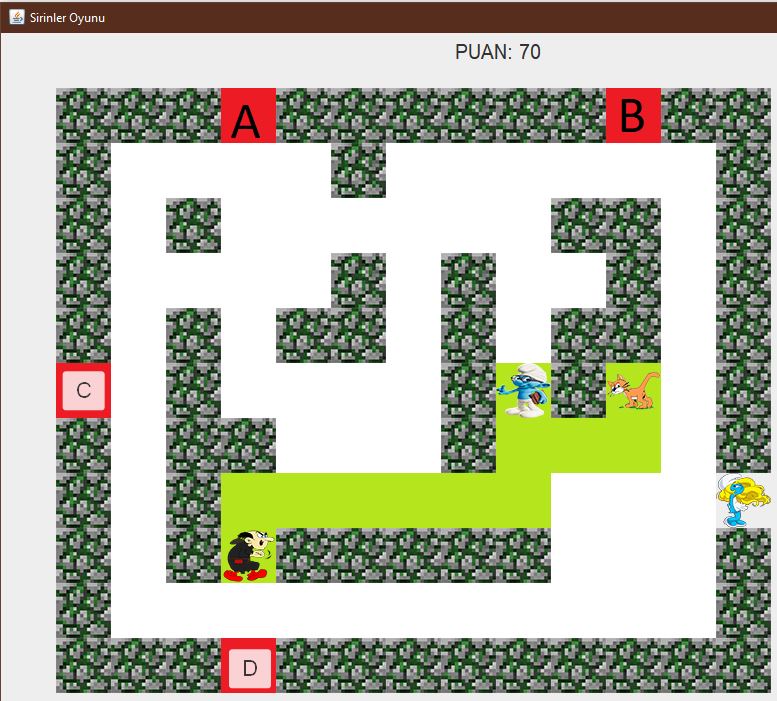
Bu durumda DijkstraShortestPath metodunu incelediğimizde: Daha sonra bazı 1 karmaşıklığında olaylardan sonra 1 adet For döngüsü ve bunun içerisinde ise karmaşıklığı 1 şeyler bulunuyor. Bu durumda bu For döngüsünün karmaşıklığı n diyebiliriz. İkinci bir For ile karşılaşıyoruz ve bu da 1. For ile aynı şekilde n karmaşıklığında. İkinci For’dan sonra bir adet koşulu “true” olan while döngüsü başlamakta.

Bu while döngüsünü incelediğimizde: while ‘ın içerisinde karmaşıklığı n olan(içerisinde bir adet for döngüsü bulunduran)metoda eşitlenen bir NodeWeighted nesnesi başlatılıyor. Daha sonra bu atamadan sonra bir adet daha while karmaşıklığı n olan bir while döngüsü başlıyor. While dan sonra bir adet karmaşıklığı n olan bir For bulunmakta. Bu durumda bütün bu döngüleri ele aldığmızda hepsi tek tek çarpı baştaki while= 3\*n2. Yani bu kısımda toplamda DijkstraShortestPath metodunun karmaşıklığı: 3\*n2+2\*n’den Big O(n2) dir.

Sonuç olarak: toplam Dijkstra karmaşıklığı **Big O(n3)** tür.

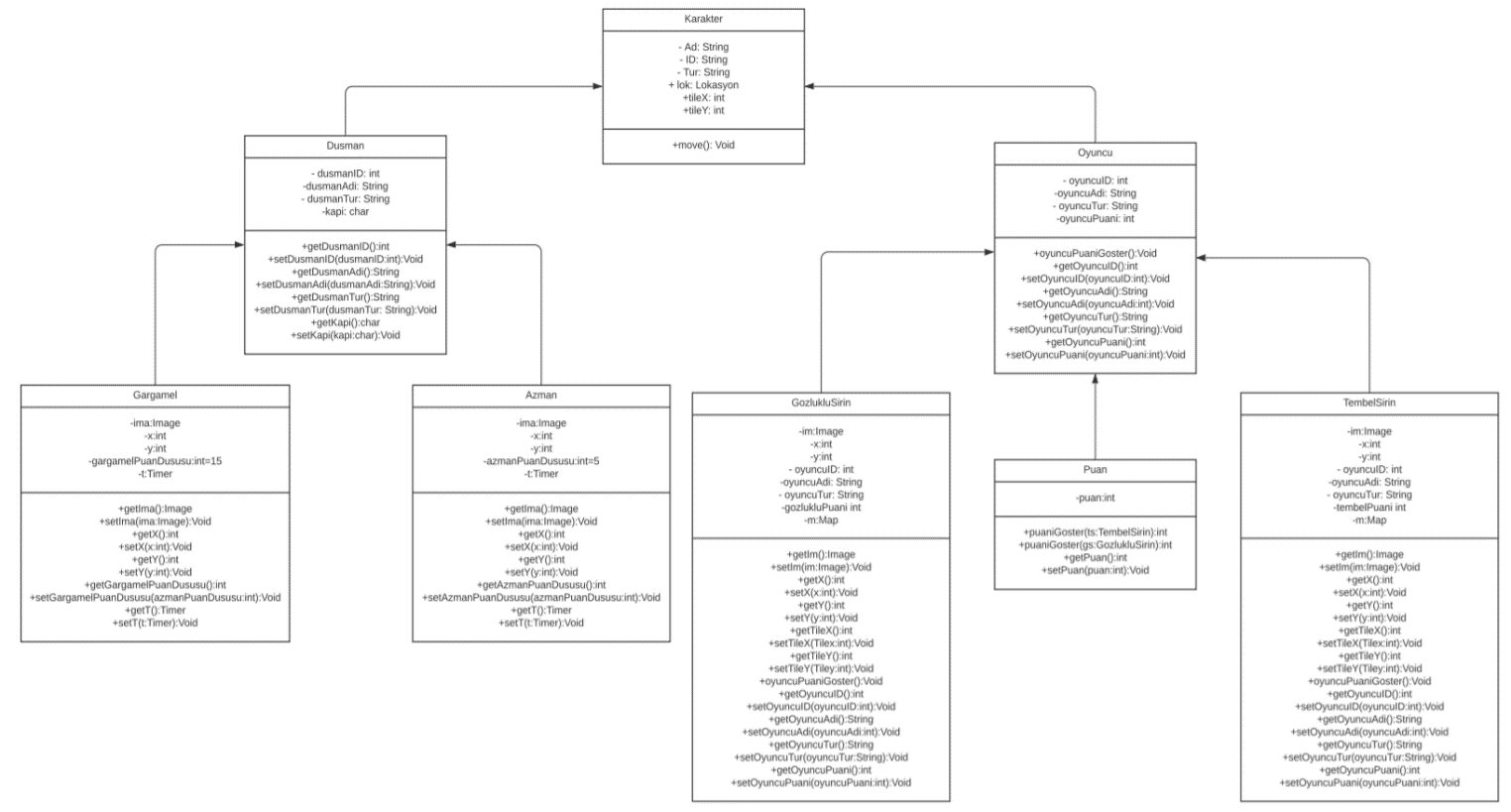
# **DENEYSEL SONUÇLAR**







# **UML SINIF DIAGRAMI**



Tamamı: <https://drive.google.com/file/d/1VcfajDa39NKo8IqhSkSfRMCQKcqzJHKa/view>

# **FLOWCHART**

<https://drive.google.com/file/d/1V_-8JOons5AOpqX01Z1--IaX-I6Vh1gD/view>

## **KAYNAKÇA**

<https://www.youtube.com/playlist?list=PLA38BD857DC0EE786>

<https://stackabuse.com/graphs-in-java-dijkstras-algorithm/>

<https://www.w3schools.com/java/java_files_read.asp>

<https://www.geeksforgeeks.org/how-to-remove-all-white-spaces-from-a-string-in-java/>

<https://stackoverflow.com/questions/23623159/how-to-have-string-image-disappear-after-a-set-time-in-java>