KOCAELİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA LABORATUVARI II

ŞİRİNLER

ÖMER HAMSİ-ALPER TALHA KÜÇÜK

[180202111@kocaeli.edu.tr](mailto:180202111@kocaeli.edu.tr) / [180202034@kocaeli.edu.tr](mailto:180202034@kocaeli.edu.tr)

***Özet***

Bu projede bizden dijkstra algoritmasını kullanıp oyuncuyla düşman arasında en kısa yolu çizerek nesneye yönelik programlama temelleri ile bir oyun yapmamız istenmiştir.

# Giriş

Bu projede dijkstra ile nesneye yönelik programlamayı birlikte kullanmaya çalıştık. Swing ile arayüz geliştirmesi gerçekleştirdik.Yön tuşlarıyla oyuncu hareket ettirme ve timer task sınıfıyla belirli bir süre objeler ortaya çıkarma işlemini gerçekleştirdik. Haritamızın verilerini bir .txt dosyasından okuyarak projeye entegre etme işlemlerini gerçekleştirdik. Bu adımlarla ilgili detaylı bilgileri “Yöntem” kısmından okuyabilirsiniz.

# Yöntem

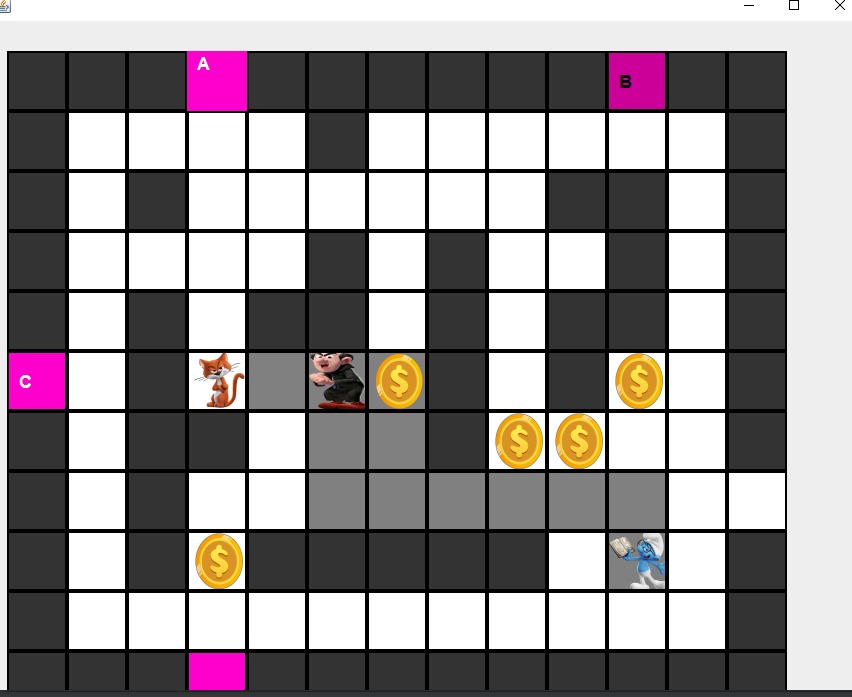
Öncelikle, haritadan okuma işlemlerini gerçekleştirip düşman karakterleri, kapı bilgilerini ve harita bilgilerini farklı array listelerde tuttuk. Daha sonra Swing kütüphanesini kullanarak arayüz hazırlama işlemini gerçekleştirdik.Daha sonra kullanıcıya click metodunu kullanarak kullanıcıya gözlüklü ve tembel şirin seçeneklerini verdik. Daha sonra karakter sınıfından tıklanan ikona göre tembel ya da gözlüklü sınıfıymış gibi davranan bir obje oluşturduk. Bu şekilde polymorphism kullanmış olduk. Oyuncunun koordinatlarına göre mavi bölgede oyuncunun oluşması sağlanmıştır. Dosyadan okunan düşman karakterlerinin isim bilgilerine göre düşman sınıfından düşman objeleri oluşturduk. Haritadan alınan kapı bilgilerine göre konum bilgileri ekleyip gerekli kapılarda ikonların oluşması sağlanmıştır. Daha sonra dijkstra algoritması kullanılarak her bir kutu, bir düğüm olarak atanıp bir array liste tutulmuştur. Haritada 1 olan alanların komşuluklarını belirlemek için çift boyutlu matris içerisinde yukarı-aşağı-sağ-sol alanları 1 olan alanlar her düğüm için komşulukları olarak belirlenmiştir. Her biri için mesafeler 1 olarak atanmıştır.Dijkstra short path compute shortest path, oyuncunun bulunduğu düğümü başlangıç düğümü olarak gönderip gönderilen düğümü uzaklığı 0 olarak atanıp diğer her düğümün uzaklığı sonsuz olarak başlangıçta belirlenmiştir. Daha sonra başlangıç düğümünün komşularıyla olan mesafeleri kontrol edilip komşu düğümün ilk atandığı ilk mesafeden küçük ise komşu düğüme yeni mesafesi olarak atanmıştır. Bu bilgiler bir priority queue’de tutularak bu listedeki en küçük mesafeye sahip düğüm seçilerek o düğümün komşulukları arasındaki mesafeler tekrar incelenip bu şekilde priority queue boşalana kadar bu işlem devam eder. Her kuyruğa gelen bir önceki kuyruk bilgisi düğüm içindeki predecessor içinde tutulup daha sonra hesaplanan en kısa mesafeyi bulmak için dijkstra sınıfı içerisindeki get shortest path to sınıfından gerekli düğüm bilgileri alındı. Daha sonra bu düğüm bilgilerinin kutuları griye boyanarak çizilmiştir. Oyuncunun her hareketinde yeni konum bilgisi, en kısa yolu hesaplamak için compute shortest path içerisine başlangıç düğümü olarak tekrar atanır. Oyuncu hareket ettikten sonra hareket etmeden önceki en kısa yoldaki ilk kutuya hareket eder ve daha sonra o kutu bilgisinden oyuncunun yeni kutusuna olan en kısa yolu tekrar hesaplar, daha önceki gri çizilmiş alanlar silinip yeni kutular gri olarak çizilir. Eğer oyuncu, herhangi bir düşman karakterle üst üste gelir ya da gözlüklü şirinde üzerinden geçer ise başlangıçta düşman karakterin başlangıç konumunu tutan bir liste yardımıyla başlangıç konumuna geri döner ve puanı Azman ya da Gargamel ile bağlantılı olarak azalır. Başlangıçta düşman karakter objelerini tuttuğumuz dizi sayesinde oyuncunun her hareketinde bir for döngüsü ile düşman listesi içerisindeki ne kadar karakter var ise her biri için bu işlemler gerçekleştirilir. Bir obje sınıfı oluşturup bu sınıfın alt sınıfları oluşturarak altın ve mantar sınıfları oluşturulmuş, bu sınıfların adet ve değer özellikleri bulunmaktadır. Başlangıçta altın adedi 5 ve değeri 5, mantarın adedi 1 ve değeri 50 olarak belirlenmiştir. Daha sonra haritada 1 olan alanlarda timer task sınıfı kullanılarak belirli saniye aralıklarında ve belirli sürede ekranda kalan ikonlar oluşturulmuştur. Eğer başlangıçta oluşan altın ya da mantar, kullanıcı üzerinde oluşursa direkt olarak gerekli puan, abstract olarak tanımlanan karakter sınıfındaki abstract olan puan göster metodu karakter sınıfının alt sınıfı olan puan sınıfında tanımlanmıştır. Puan sınıfının skoru, başlangıçta oyuncunun skor özelliği olarak atanmıştır. Eğer oyuncu gözlüklü şirin ise, altın ya da mantarın üzerine gelmesi ya da üzerinden geçtiğinde puan sınıfının skor özelliği altın ya da mantar olmasına göre güncellenmiştir. Tembel şirin ise puan alması için üzerine gelmesi yeterlidir.Oyuncunun her hareketinde puanı ve başlangıç noktasında olup olmadığı kontrol edilerek eğer puanı 0 ya da 0’ın altında ise ya da hedef noktasına ulaşmış ise gerekli bilgi pop-up mesajıyla verilir.

# Sonuçlar

Bu projede veri yapıları algoritmaları nesneye yönelik programlama temelinde kullanma deneyimi yaşadık. Ayrıca dijkstra algoritmasını nasıl koda dökeceğimizi ve projeye nasıl entegre edebileceğimizle ilgili deneyimler edindik. Timer task sınıfıyla belirli sürede belirli aralıklarla işlemler yapmayı öğrendik. İkon hareket ettirmeyle ilgili eventlar üzerinde çalışmalar yaptık.Swing kütüphanesi üzerinde yeni özellikler araştırıp kullanmaya çalıştık.

# Çıktılar

# 6cdaf2da-6a0b-4871-880a-35715aee7024.jfif

****

# Pseudo Code:

1-Dosyadan okuma işlemlerini gerçekleştir. Düşman ve kapı bilgilerini liste içerisinde tut.

2-Deneme sınıfından bir obje oluştur, deneme sınıfından Run et.

3-Gözlüklüye tıklanırsa gözlüklü sınıfından bir obje oluştur, tembele tıklanırsa tembel sınıfından bir obje oluştur ve Jpanel74’ü kapat, Jpanel5’i aç. Process metodunu çalıştır.

4-Label tanımlama metodunu çalıştır.

5-Düğümlerin komşuluklarını belirle ve beyaz alanları bir listede tut.

6-Timer task sınıfından bir obje oluştur. 5 adet random sayı oluştur.

7-Beyaz alan dizisi içindeki random sayılarla oluşturulan alanlarda ikon oluştur.

8-Başlangıçta oluşan ikon, oyuncunun konumuyla aynı yerdeyse oyuncunun puanını artır.

9-Bu görevi 5 saniye boyunca uyut ve sonra altınları yok et.

10-Yeni bir timer task oluştur ve 1 adet random kutuda ikon oluşmasını sağlar.

11-Bu görevi 20 saniyede bir çalışacak şekilde başlat ve 7 saniye ekranda kalsın.

12-Oyuncu konumunu başlangıç düğümü olarak belirle ve compute short path fonksiyonuna gönder.

13-Path’i tutacak bir array list oluştur ve dijkstra sınıfından geriye bir path döndür.

14-Alınan path’i tek tek gez ve griye boya.

15-Kullanıcı sağa-sola-yukarı ya da aşağı hareket ederse harita matrisi üzerinde alan 1 ise kullanıcı konumunu güncelle.

16-Dijkstra fonksiyonunu çalıştır.

17-Düşman objelerini tutan liste üzerinde for döngüsüyle dön.

18-Düşmanın adı Gargamel ise bütün düğümleri sonsuz mesafe olarak ata.

19-Düşman konumu, oyuncu konumu aynı ise oyuncunun puanını azalt.

20-Düşman, hareket edeceği konumu 1 ise düşman konumunu güncelle.

21-Düşman konumunu compute shortest path’e başlangıç konumu olarak gönder.

22-Get shortest path’e oyuncunun konumunu hedef düğüm olarak gönder.

23-Path içerisindeki bilgilere göre kutuları griye boya.

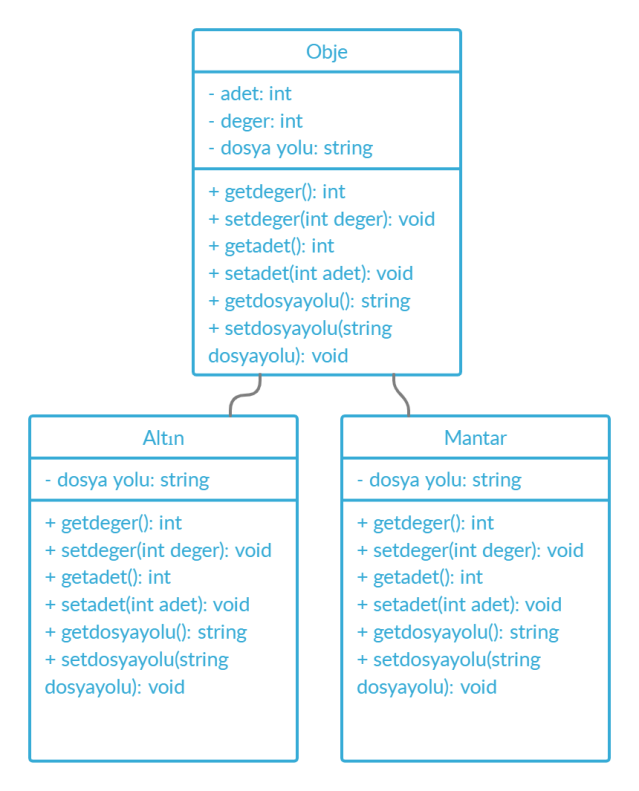
24-Oyuncunun yeni konumu altın ya da mantar üzerinde ya da üzerinden geçmişse oyuncu puanını artır.

25-Oyuncu puanı 0 ya da 0’dan küçükse pop-up mesajıyla bilgilendir.

26-Oyuncu hedef noktasına ulaşmışsa pop-up mesajı ile bilgilendir.

# UML DİYAGRAMI:

# Untitled Document (2).jpg

****

# KAYNAKÇA:

# <https://www.baeldung.com/java-dijkstra>

# <https://www.geeksforgeeks.org/java-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/>

# <https://www.softwaretestinghelp.com/dijkstras-algorithm-in-java/>

# <https://java2blog.com/dijkstra-java/>

# <https://serdarkuzucu.com/javada-timer-ve-timertask-siniflarinin-kullanimi/>