PROGRAMLAMA LABORATUVARI - II

1. PROJE

Ahmet Furkan EREN – Recep TEKTAŞ

Bilgisayar Mühendisliği Bölümü(İÖ)

Kocaeli Üniversitesi

Özet

Bu doküman Programlama Laboratuvarı 1 dersini 1. Projesi için çözümümüzü açıklamaya yönelik oluşturulmuştur. Dokümanda projemizin tanımı, çözüme yönelik yaptığımız araştırmalar, kullandığımız yöntemler, proje sürecinde karşılaştığımız problemler, proje hazırlanırken kullandığımız geliştirme ortamı, ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Dokümanın sonunda projeyi hazırlarken kullandığımız kaynaklar bulunmaktadır.

1. Proje Tanımı

1.1 Proje Tanımı

Projede bizden istenen binary olarak verilen labirent üzerinde, djikstra algoritması ile en kısa yol bulan bir program oluşturmamızdır.

Labirent üzerinde kullanıcı hedefe doğru ilerlerken, bir veya iki farklı düşman karakterinden de kaçmak zorundadır.

Bu düşman karakterleri djikstra algoritması ile kullanıcı karaktere ulaşmaya çalışacaktır. Düşman karakter kullanıcının karakterine her ulaştığında kullanıcın puanı azalacak ve kullanıcının puanı bittiğinde ise elenecektir.

Ayrıca oyun sırasında random olarak oluşması gereken altın ve mantar itemleri olacak ve oyuncu karakter bunlardan topladıkça puanı artacaktır. Altınlar ve mantarlar random olarak belli bir süre için ekrana gelecektir ve belirli bir süre sonra kaybolacaktır.

1.2 İsterler

- Labirent için bir arayüz tasarlanmalı. Garfik kütüphanesi kısıtlaması yok.
- Oyuncu 20 puan ile başlamalı ve 0 olduğunda başarısız olmuş olmalı.
- En kısa yol algoritması olarak sadece Dijikstra kullanılacaktır.
- 2 farklı oyuncu karakteri bulunmalı. Gözlüklü şirin ve Tembel Şirin. Gözlüklü Şirinin puanı farklı şekilde azalıyor olmalı.
- İki farklı düşman karakteri bulunmalı. Gargamel ve Azman. Tek hamlede Gargamel iki birim ilerleyebiliyorken, Azman bir birim ilerleyebilmeli.

2. Araştırmalar ve Yöntem

Projemize başlamadan önce Java veya C++ dillerinden birini seçmemiz gerekiyordu. Java dilinin bu proje için daha iyi olacağına karar verdik. Ardından görüntü kütüphanesi için JavaFx veya Swing kullanmak arasında kaldık. Daha önce Swing ile az miktarda da olsa tecrübemiz vardı. Buna karşın JavaFx daha gelişmiş olsa da bu proje için Swing kullanmak hem yeterli hem de bizim için daha hızlı bir seçenekti. Bu yüzden Swing kütüphanesini seçmiş olduk.

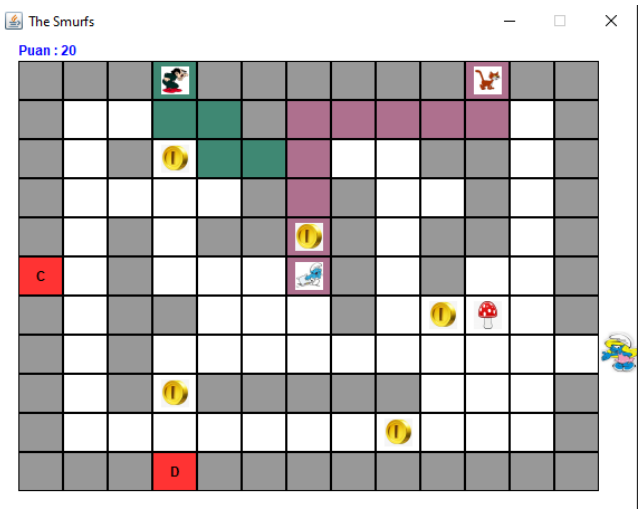
Öncelikle Swing kütüphanesini ve Djikstra algoritmasını araştırmakla başladık. Swing kütüphanesini Youtube'dan ve çeşitli sitelerden örnekler ile kısa sürede kavradık. Zaten baştan beri basit ve kolay olduğu için seçmiştik. Düşündüğümüz gibi oldu.

Fakat Dijkstra algoritmasını daha önce görmüş olmamıza rağmen bunu binary bir labirent üzerinde uygulamak ve koda dökmek kolay olmadı. Dijkstar algoritmasının genelde düğümler(node) üzerinde kullanıldığını gördük. Benzer bir stratejiyi bu projedeki labirent üzerinde uygulamanın mantıklı olmadığına karar verdik. Binary labirentler üzerinde yapılan en kısa yol bulma algoritmalarını inceledik ve buradan yola çıkarak Dijikstar algortiması ve bizim labirentimizde doğru şekilde çalışan yapıyı oluşturmayı başardık.

Ayrıca karakterlerin konumlarını tutmak ve her seferinde dinamik olarak en kısa yolu hesaplatmak zorlayıcı bir kısımdı. Bunun için çeşitli örnekler ve projeler inceledik. Yolları tutumak için ArrayList kullanmanın daha iyi olacağına karar verdik.

Altın ve mantarları zamana bağlı çıkacak şekilde ve belli bir süre sonra kaldırılacak şekilde tasarladık. Duvarlarda değilde sadece yol üzerinde çıkmalarını sağladık. Karakter bu itemlerden topladığında puanı artıyor ve daha uzun süre oyuna devam edebiliyor.

Daha sonra text üzerinden binary labirenti okuyup doğru şekilde görselleştirmek için çalışmalara devam ettik. Bunun için de bir sürü kaynaktan örnekler incelememiz gerekti. Karakter adlarını okumak ve ardından gelen labirent blokları (0/1'ler) alınırken hata oluşmaması için extra özen göstermemiz gerekti. Çünkü karakter satırı bir veya iki satır olabilmekte.



Yukarıdaki kısımlar sırasında Swing ile basit tarzda garfiklerimizi oluşturmuştuk. Labirenti Swing penceresine ekledik. Ardından altın ve mantar gibi arada çıkması gereken itemleri ayarladık. Bunların konumları random olarak belirleniyor ve belli bir süre içinde kayboluyor.

Bu süreçler sırasında her aşamada kodların doğruluğu ve uyumluluğunu kontrol ettik. Tüm sorunlarımızı kolayca online kaynaklardan temin edip projemizi başarı ile sonuçlandırdık.

2.1 Karşılaşılan Problemler

Dijikstra algoritmasının genelde düğümler(node) üzerinde kullanıldığını gördüğümüzde bunu binary bir labirente nasıl uyarlayacağımız konusunda endişelendik. Çünkü her birimi bir node olarak kabul etsek bir sürü bağlantıyı yapıyor olmamız gerekecekti. Araştırmalarımız derinleştikçe bu kısmın nasıl olabileceği hakkında görüş ve fikir sahibi olmuş olduk.

Karakterlerin konumunu tutmak ve karakteri takip kısmından nasıl bir yapı kullanacağımız konusunda farklı seçenekler arasından en iyisini seçmeye çalıştık. ArrayList , Queue veya Stack yapılarından birini seçmeliydik ve ArrayList kullanmaya karar verdik. Çünkü hem çok basit hem de işimizi kolayca göreceğini düşündük.

2.2 Kazanımlar

Öncelikle Java dili ile yapmış olduğumuz bu proje hem java bilgimizi hem de Object Oriented Programming bilgilerimizi pekiştirmede çok faydalı oldu.

Inheritance , constructor ve get-set metotları ile sınıf yapılarını aktif şekilde kullandığımız bir proje oldu.

Veri yapıları konusunda ise ArrayList'leri kullanmış olmamıza ragmen kısaca diğer methodlara da kısaca bakmış olduk ve bu konunda da pekiştirici bir çalışma idi.

Ayrıca görsel kütüphanesi olarak Swing kullanmış olmamıza ragmen JavaFx kütüphanesine de bakma fırsatımız oldu ve bu sayede JavaFx hakkında genel yapısı için bir öngörümüz olmuş oldu.

Dijkstra algoritmasının binary labirent üzerinde implement edilmesi ve en kısa yolu dinamik olarak hesaplaması konusu her ne kadar zorlayıcı olmuş olsa bile sonuçta bu algoritmayı doğru şekilde oluşturmuş olmak ve başarılı bir şekilde çalışmasını sağlamak Java dili üzerinde olan özgüvebimizi arttırmış oldu.

3. Geliştirme Ortamı

Projemizi Windows işletim sistemi üzerinde Java dili ve Eclipse IDE üzerinde geliştirdik.

Görsel tasarım için ise Java Swing kütüphanesini kullandık.

Sürüm kontrolü için de Git versiyon kontrol sistemi kullandık.

4. Kod Bilgisi

4.1 Akış Şeması

Akış şeması ektedir. [Ek-1]

Uml diyagramı ektedir. [Ek-2]

4.2 Algoritma

Program çalıştığında kullanıcıdan bir oyuncu seçmesi beklenir. Kullanıcının seçtiği oyuncuya göre oyuncu nesnesi oluşturulur. Haritanın oluşturulması için harita.txt dosyası okunur. Dosyada bulunan binary bilgiler doğrultusunda 0 ise duvar 1 ise yol olacak şekide harita olşturulur. Yine dosya içerisinde bulunan düşman bilgisi ve haritaya giriş yapacağı kapı bilgisi okunur. Kullanının seçmiş olduğu oyuncu da haritada (5,6) noktasına yerleştirilir.Bu bilgiler doğrultusunda harita oyuna başlamak için hazır olur ve kullanıcının önüne grafiksel olarak sunulur.

Harita hazır olduğunda düşman karakterler (5,6) noktasında bulunan oyuncu karaktere en kısa yoldan ulaşmak için Dijkstra algoritmasını içeren *enKısaYol* metodu çağırılır ve metotdan dönen sonuç doğrultusunda kareler renkli hale getirilir. Bu işlemden sonra kullanıcıdan ok tuşlarına basarak ilerlemesi beklenmekterdir. Kullanıcı her ok tuşuna baştığında seçmiş olduğu oyuncu karakterin kaç birim ilerleyebileceği bilgisine göre oyuncunun ilerlemesi sağlanır ve hemen ardından düşman karakterler tekrardan *enKısaYol* metodunu çağırır ve dönen sonuca göre düşman karakterlerin ilerlemesi sağlanır.

Kullanıcı oyunu oynarken haritada rastgele yerlerde belirli süreler dahilinde altınlar ve mantarlar oluşur. Oyuncu altının üzerine geldiğinde 5 puan, mantarın üzerine geldiğinde ise 50 puan kazanır.

Kullanıcı oyuncusunu her hareket ettiğinde düşman karakterlerde ilerler ve oyuncuya yaklaşır. Eğer oyun içerisinde düşman karakter oyuncu karaktere dokunursa oyuncunun puanı azalır. Eğer dokunan düşman gargamel ise oyuncu 15 puan, azman ise 5 puan kaybeder.

Kullanıcı oyunu oynarken (7,12) noktasına ulaşmayı başarırsa oyunu kazanır ama (7,12) noktasına ulaşamadan puanı sıfır veya sıfırın altına düşerse oyunu kaybeder.

4.3 İstatistik

Projemiz toplamda 20 dosya içermekle beraber 1109 satır kod içermektedir. Buna ek olarak kod düzenini sağlamak ve okunabilirliği arttırmak için 270 boşluk satırı kullanılmıştır.

5.Sonuç

Proje üzerinde yapmış olduğumuz yaklaşık üç haftalık çalışmanın sonunda, proje başarı ile sonuçlanmıştır.

Tüm isterler ve gereksinimler başarı ile karşılanmıştır. Böylelikle projemiz son bulmuştur.

Kaynakça

I. Dijikstra Algortiması :

https://medium.com/t%C3%BCrkiye/graf-teorisi-4-en-k%C4%B1sa-yol-problemi-322a648c864e

II. Dijikstra Algortiması En Kısa Yol :

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

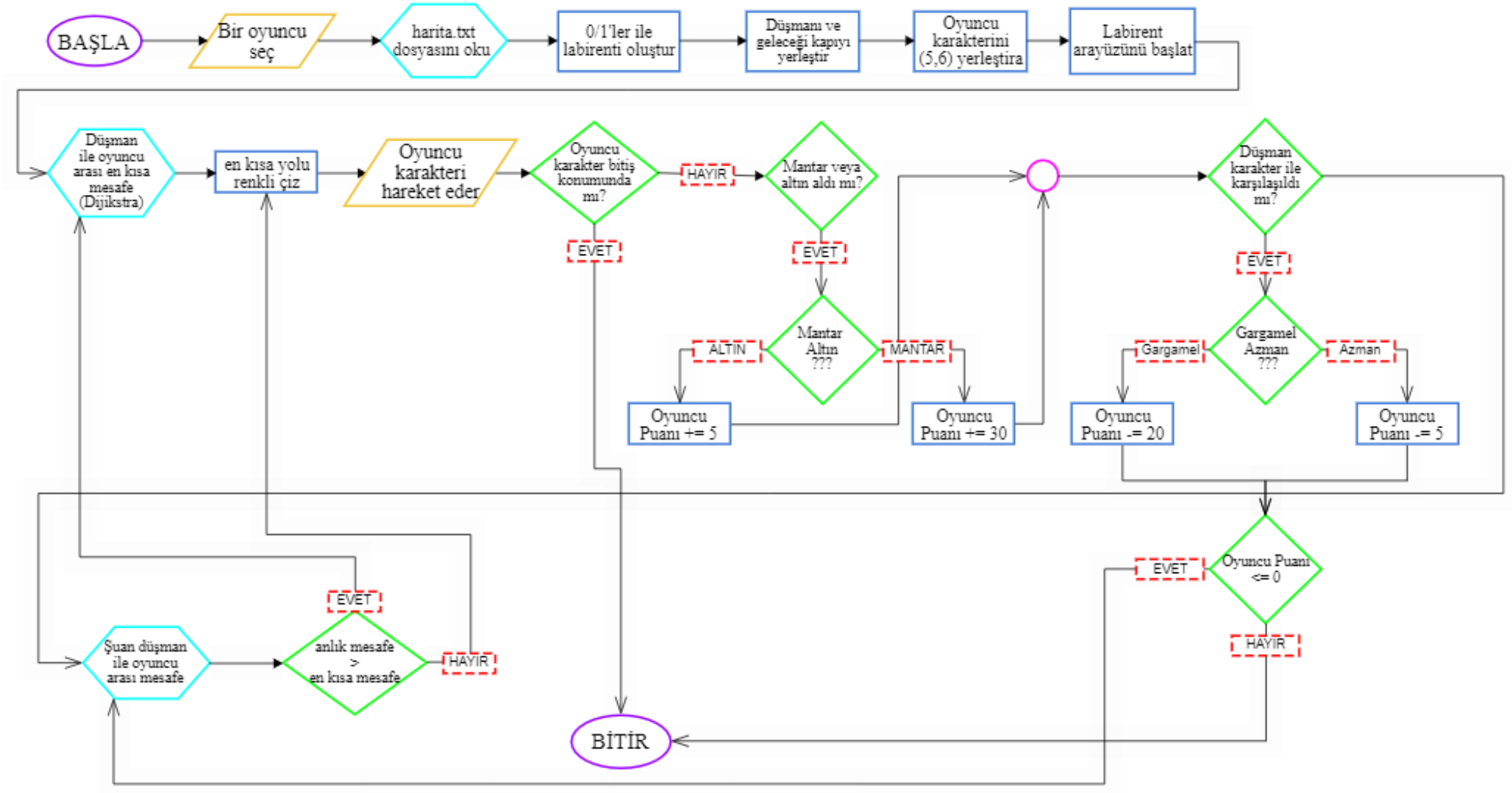
III. Java ArrayList Hakkında :

https://www.javatpoint.com/java-arraylist

IV. Java Swing Kütüphanesi Hakkında :

https://www.youtube.com/results?search_query=java+swing+k%C3%BCt%C3%BCphanesi

AKIŞ ŞEMASI - Ek[1]



UML Diyagram [Ek-2]