PROGRAMLAMA LABORATUVARI 2

1. PROJE

Murat Karakurt

Ahmet Burhan Bulut
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi

Özet

Bu doküman Programlama Laboratuvarı 2 dersi 1. Projesi için çözümümü açıklamaya yönelik oluşturulmuştur. Dökümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje sürecinde karşılaşılan problemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemi hazırlarken kullandığım kaynaklar bulunmaktadır.

1. Proje Tanımı

1.1. Proje Tanımı

Projede bizden istenen Bu proje ile nesneye yönelik programlama ve veri yapıları algoritmalarını kullanarak Şirinler oyunu tasarlamamız beklenmektedir. Seçilen oyuncunun Labirent içerisinde puanlarını bitirmeden önce Şirine'ye ulaşması gerekmektedir.

Oyuncu labirent içerisinde gezinirken her adım attığında düşman karakterler oyuncuya ulaşmak için gereken en kısa yolu hesaplar ve o yolu takip ederek oyuncuya ulaşmaya çalışır.

1.2. İsterler

Proje tanımında belirtilen kurallar ve isterler aşağıdaki gibidir:

- Oyun temasına uygun arayüz tasarlanması.
- Oyuncu 20 puan ile oyuna başlamalıdır.
- Oyuncu puanı 0 ya da 0'ın altına düştüğünde oyun oyuncunun başarısızlığı ile sonlanacaktır.
- Düşmanların başlangıç noktası ve oyun haritası .txt uzantılı dosyadan okunacaktır.
- Oyunda iki adet düşman karakter bulunacaktır.
- Düşman karakterlerin en kısa yolu bulması için Dijkstra algoritması kullanılmalıdır.
- En kısa yolu hesaplayan düşman karakterler oyun haritası üzerinde renkli olarak en kısa yolu gösterecektir.
- Oyuncunun her hamlesinde Dijkstra algoritması tekrar çalışarak yolu yeniden hesaplar.

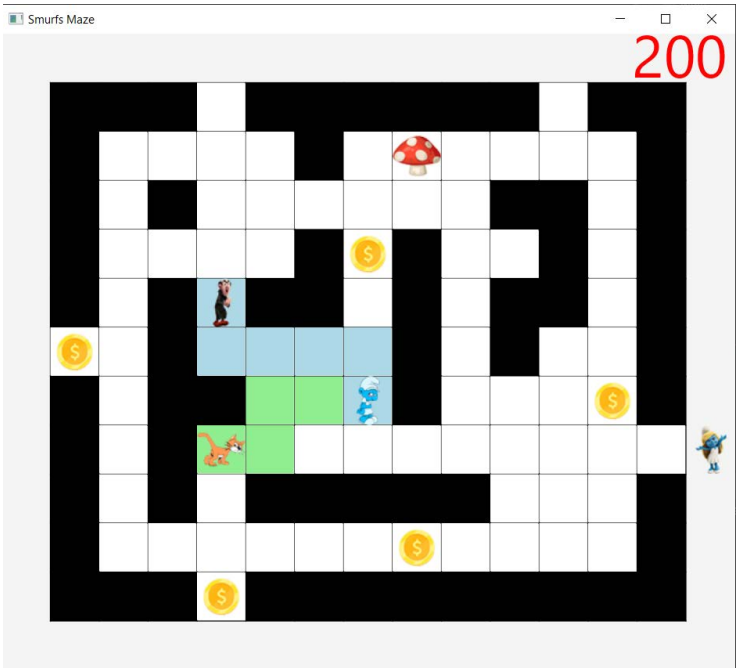
• Araştırmalar ve Yöntem

Projeye ilk olarak Dijktsra algoritmasının ne olduğunu, ne demek olduğunu araştırmakla başladık. Araştırmalardan yola çıkarak en kısa yol üretecek olan `dijktra` algoritmanın nasıl uygulanacağını öğrendik. Algoritmanın kod ile nasıl entegre çalışacağını araştırdık.

Projede istenilen altın ve mantarları ekranda belirli bir süre tutmamız gerektiği için Javafx için Timer araştırması yaptık.

.txt uzantılı dosyayı haritaya nasıl dönüştüreceğimizi ve Javafx ile nasıl haritayı çizdireceğimizi araştırdık. Harita çizildikten sonra haritada duvar olarak tanımlayabileceğimiz karakterlerin üzerinden geçemeyeceği bölgeleri belirledik.

Oyun temasına uygun oyun içi görseller ve karakter simgeleri ayarladık.



2.1. Karşılaşılan Problemler

Haritayı koordinat sistemi gibi düşünerek koordine bir şekilde çalışmaları için matematiksel hesaplamalar yaparken kaymalar olduğu için simgeler ve karakterler tam olarak istenilen şekilde konumlandırılamıyordu.

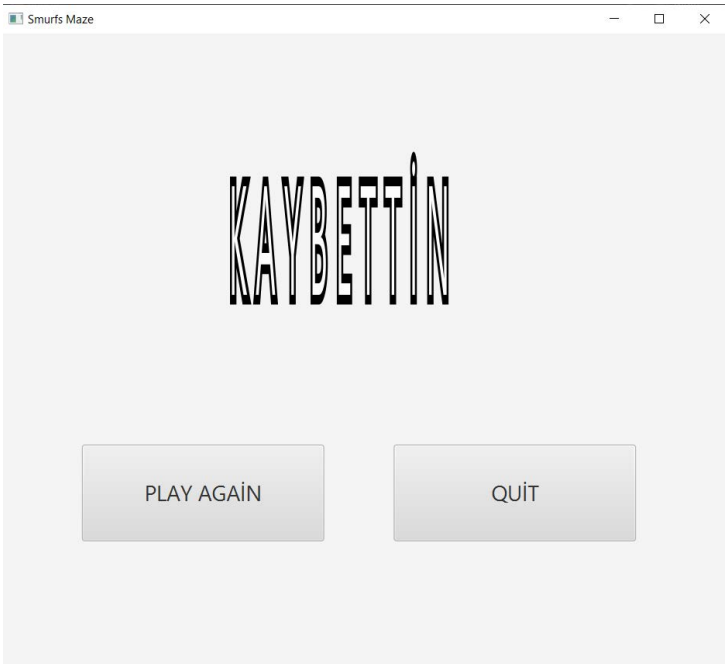
Mantar ve altınların sürekli yeniden oluşmalarını sağlayamadık. Bu problemi çözmek için Javafx de zamanlayıcı kullanmayı düşündük araştırmalar sonucu bunun mantıklı olacağına karar verip timer kullanarak projede istenilen şekilde sorunu çözdük.

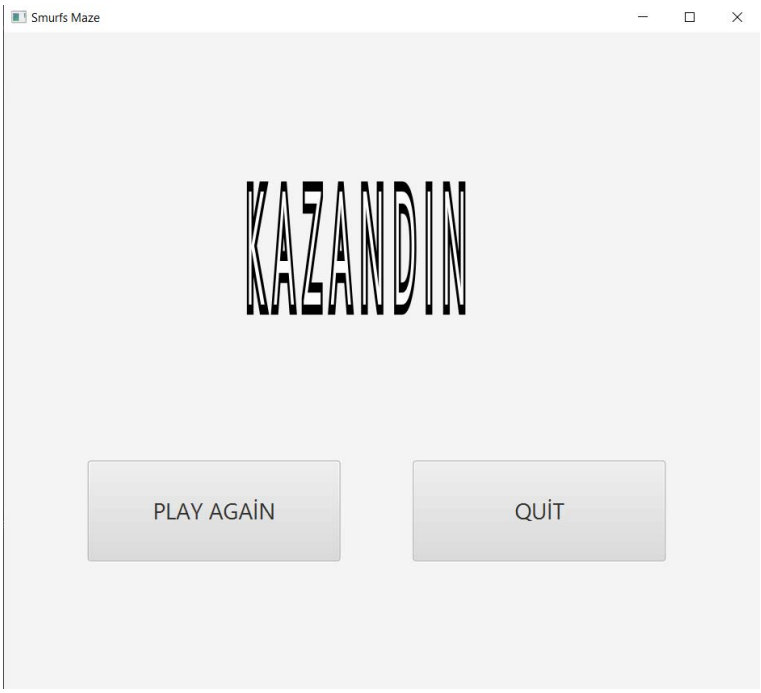
Djikstra algoritması ile oyun haritamız eşleşmiyordu. Bu problemin çözümü için ekrandaki tüm harita karelerine Javafx rectangle atadık böylece dijkstra algoritması ile harita eşleşmiş oldu.

Düşman karakterlerinden Gargamel Azman karakterinin üzerinden atlayabilmesi gerekirken Azman karakterinin Gargamel'in üzerinden atlayamaması gerekiyordu bunu ilk olarak çözemedik iki karakter de birbiri üzerinden geçebiliyordu daha sonra bu problemin üzerinden yoğunlaşarak gerekli koşulları yazarak Azman karakterinin Gargamel'in üzerinden atlamasını engelledik.

3.YALANCI KOD

- .txt uzantılı dosyadan veriler okunarak bir matrise atandı karakterlerin giriş kapıları ve oyun içindeki duvarlar belirlendi.
- Belirlenilen karakter yerleri ve duvarlara göre harita oluşturuldu.
- Oluşturulan harita üzerindeki kareler dosyadan okunan matristeki verilere göre boyandı.
- Karakter yönlendirme kontrolleri yapıldı.
- Altınlar ve mantar belirli bir sürede oluşup toplanılmaması durumunda yok edilecek şekilde ayarlandı.
- Oyuncu ve düşman karakterler duvarların içinden ve üzerinden geçilmeyecek şekilde ayarlandı.
- Oyuncunun karakter seçimine göre (Tembel ve Gözlüklü) karakterlerin yetenekleri belirlendi.
- Düşman karakterler Gargamel her oyuncu hamlesinden sonra 2 adım Azman karakteri 1 adım atacak şekilde ayarlandı.
- Azman karakterinin Gargamel karakterinin üzerinden atlaması engellendi.
- Dijktra algoritması ile düşman karakterler oyuncuya en kısa yolu hesaplaması sağlanıldı.
- Düşman karakterler en kısa yolu bulduktan sonra en kısa yolu renkli şekilde gösterilmesi sağlanıldı. Her oyuncu hamlesinde dijkstra algoritması tekrar çalışarak yeniden en kısa yolu hesaplaması için düzenlemeler yapıldı.
- Oyuncu puanı 0 veya 0'ın altına düştüğünde oyunu bitirdi.
- Oyuncu mantar topladığı zaman 50 puan altın topladığı zaman 10 puan kazanacağı şekilde ayarlandı.
- Oyuncu Şirine karakterine ulaştığında Kazandın ekranına ve oyunun bittiğini gösteren ekrana geçiş sağlanıldı.

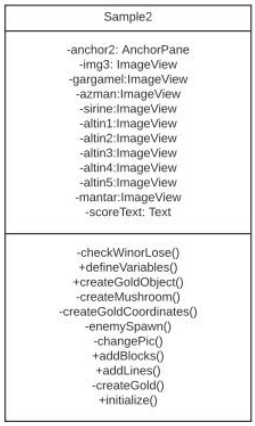


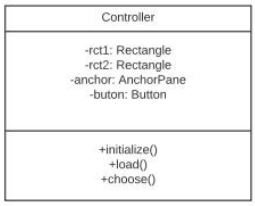


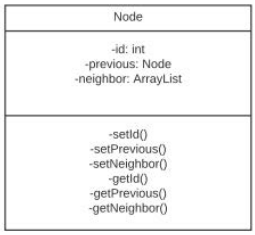
4. Geliştirme Ortamı

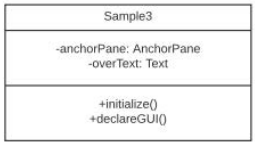
Projeyi Windows işletim sistemi üzerinde, Intellij IDEA üzerinde geliştirdik. Projenin gelişimini ve versiyonlarını takip edebilmek için de Git versiyon kontrol sistemini kullandık.

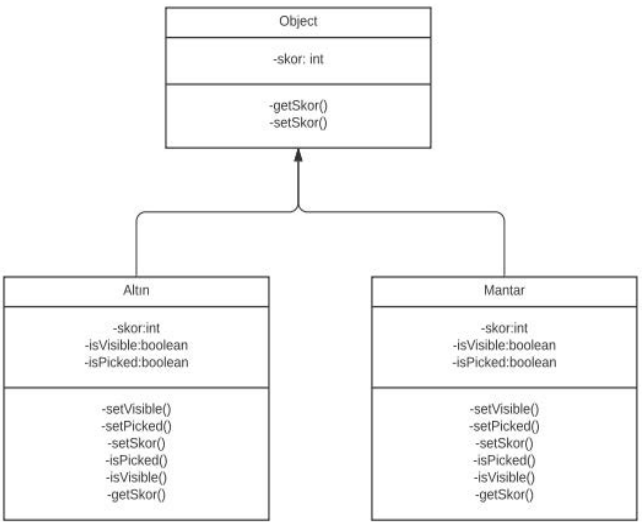
5. UML Diyagram

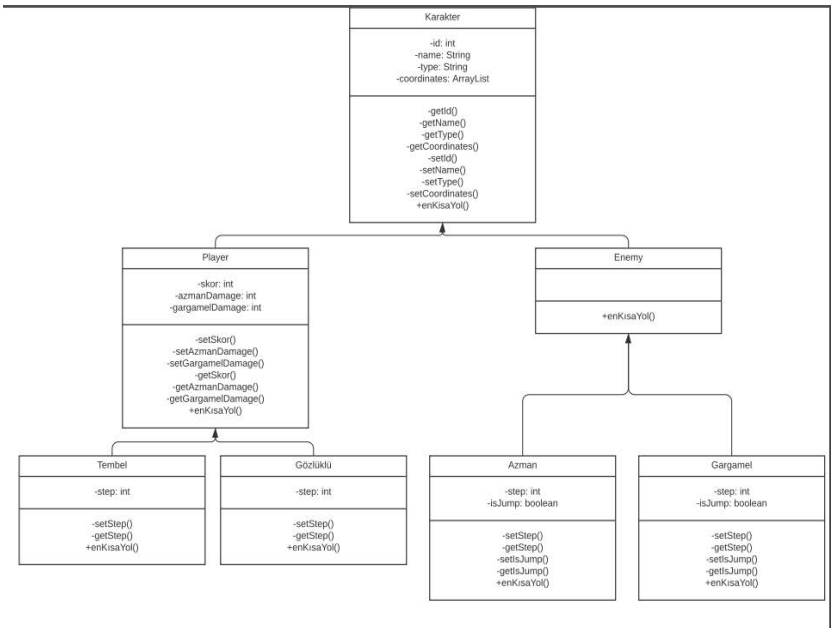












UML diyagramı tek sayfa olarak görmek için tıklayınız.

6. Karmaşıklık Analizi

```java
public static void dijkstra(int enemyId,int playerId){
    ArrayList<Node> clone = new ArrayList<>(nodes);
    ArrayList<Integer> distance = new ArrayList<>();
    ArrayList<Boolean> visited =  new ArrayList<>();

    for (int i = 0; i < clone.size(); i++) {
        distance.add(Integer.MAX_VALUE);
        visited.add(false);
    }
    distance.set(enemyId,0);

    for (int i = 0; i < clone.size(); i++) {
        int min = Integer.MAX_VALUE;
        int minIndex = 0;

        for (int j = 0; j < distance.size(); j++) {
            if (distance.get(j) < min && visited.get(j) != true){
                minIndex=j;
                min = distance.get(j);
            }
        }

        if (min == Integer.MAX_VALUE ) continue;
        visited.set(minIndex,true);
        for (Integer id : clone.get(minIndex).getNeighbours()){
            int alt = distance.get(minIndex) + 1;
            if (alt<distance.get(id)){
                distance.set(id,alt);
                nodes.get(id).setPrevious(nodes.get(minIndex));
            }
        }
    }
}
```

ZAMAN KARMAŞIKLIĞI: O(N^2)
İÇ İÇE 2 TANE FOR OLDUĞU İÇİN

SPACE COMPLEXİTY: O(V)
V -> EKRANDAKİ KARE SAYISI

```java
public void createMaze() {
    File file = new File( pathname: "C:\\Users\\mrtkr\\Desktop\\a\\src\\sample\\harita.txt");
    try {
        int i = 0, z = 0;
        Scanner fileScanner = new Scanner(file);
        while (fileScanner.hasNextLine()) {
            String data = fileScanner.nextLine();
            String[] datas = data.split( regex: " ");
            if (i >= 2) {
                for (int j = 0; j < datas.length; j++) {
                    maze[z][j] = Integer.parseInt(datas[j]);
                }
                z++;
            }
            i++;
        }
    } catch (FileNotFoundException e) {
        System.out.println("Bir Hata Olustu...");
        e.printStackTrace();
    }
    players[0] = gozluklu;
    players[1] = tembel;

}
```

O(n^2)

7. Kaynakça

- https://www.javatpoint.com/javafx-tutorial
- https://www.educba.com/javafx-game/
- https://www.youtube.com/watch?v=HKOJXlEJy6U
- https://stackoverflow.com/questions/35512648/adding-a-timer-to-my-program-javafx
- https://stackoverflow.com/questions/18619394/loading-new-fxml-in-the-same-scene
- https://github.com/tutsplus/Introduction-to-JavaFX-for-Game-Development/blob/master/Example4K.java