KOCAELİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Programlama Laboratuvarı II- I. Proje

Şirinler Oyunu

Gökçe Yılmaz yilmaz.gokce.tr@gmail.com

Dilara Çataltepe dilaracataltepee@gmail.com

Özet

Bu projedeki amaç nesneye yönelik programlama ve veri yapıları algoritmalarını kullanarak bir şirinler oyunu ortaya çıkarmaktır. Oyun başlangıcında seçilen oyuncunun labirent içerisinde puanlarını bitirmeden önce Şirine'ye ulaşması gerekmektedir. Oyun içerisinde iki farklı oyuncu ve iki farklı düşman karakter vardır. Kullanıcı oyun başlamadan önce seçtiği oyunculardan birini klavye yardımı ile kontrol ederek ve Şirine'ye ulaştırmaya çalışacaktır. Yine oyun başlamadan önce seçmiş olduğu düşman karakterlerden biri veya birden fazlası da onu durdurmaya çalışacaktır.

I. Giriş

Projede Java programlama dili ve Netbeans geliştirme ortamı kullanıldı. Arayüz tasarımı için Java Swing paketinden yararlanılmıştır. Oyun Ekranı, Oyun, Altın, Azman, Dijkstra, Düşman, Gargamel, Gözlüklü, Karakter, Lokasyon, Lokasyon, Mantar, Obje, Puan, Tembel sınıfları oluşturuldu ve proje bu sınıflar üzerinden yürütüldü. Oyun ilerleyişi Oyun sınıfı üzerinden takip edildi.

II. Yöntem

İlk olarak *OyunEkrani* adlı bir main class açtık. Daha sonra *JPanel*'den kalıtım alan, *KeyListener* ve *ActionListener*'ı implemente eden bir *Oyun* classı açtık. Oyun sınıfında *labirent* adli bir matris tanımladık. Kullanıcının hangi karakterle oynamak istediğini seçmesi için bir *JOptionPane* oluşturduk (Resim 1).

Daha sonra dosyadan haritayı ve gargamelle azmanın konumlarını okumak için dosyadan okuma işlemlerini yaptık (Resim 2).Birinci satırdan gargamelin konumunu aldık ve *String yergargamel* değişkenine atadık. İkinci satırdan azmanın konumunu aldık ve *String yerazman* değişkenine atadık. Bu iki satırın altındaki harita için iç içe for döngüsüyle değerleri alıp labirent matrisine gönderdik.

```
int selectedOption = JOptionPane.showConfirmDialog(null,
  "Oynamak istediğiniz oyuncuyu seçiniz.Gözlüklü Şirin için yes, "
          + "tembel şirin için no'ya basınız",
  "Oyuncu Seçimi",
 JOptionPane.YES_NO_OPTION);
 if (selectedOption == JOptionPane.YES_OPTION) {
    secilenoyuncu="gozluklu";
 }
 else if (selectedOption == JOptionPane.NO_OPTION) {
    secilenoyuncu="tembel";
 }
 System.out.println("Seçilen oyuncu:"+secilenoyuncu);
```

Resim 1 : Oyuncu seçimi

```
File dosya = new File("C:\\Users\\Gökçe Yılmaz\\Documents"
        + "\\NetBeansProjects\\SirinlerOyun\\harita.txt");
Scanner scn = new Scanner(dosya);
while (scn.hasNextLine()){
   yergargamel = scn.nextLine();
   yerazman = scn.nextLine();
for(int i=0 ; i<11 ; i++){

    for(int j=0 ; j<13 ; j++) {

        labirent[i][j]=scn.nextInt();

    }
}
}
```

Resim 2: Dosya okuma

Seçilen gargamel ve azman konumlarına göre gargamel ve azman sınıfında set metotlarıyla gargamel ve azmanın konumlarını belirledik. Ölçeklendirmemize göre konum belirtirken matris değerlerini *40* ile çarptık (Resim 3).

```
if(yergargamel.equals("Karakter:Gargamel,Kapi:A")==true){
gargamel.setkonum(3*40, 0*40);}
else if(yergargamel.equals("Karakter:Gargamel,Kapi:B")==true){
gargamel.setkonum(10*40, 0*40);}
else if(yergargamel.equals("Karakter:Gargamel,Kapi:C")==true){
gargamel.setkonum(0*40, 5*40);}
else if(yergargamel.equals("Karakter:Gargamel,Kapi:D")==true){
gargamel.setkonum(3*40, 10*40);}

if(yerazman.equals("Karakter:Azman,Kapi:A")==true){
azman.setkonum(3*40, 0*40);}
else if(yerazman.equals("Karakter:Azman,Kapi:B")==true){
azman.setkonum(10*40, 0*40);}
else if(yerazman.equals("Karakter:Azman,Kapi:C")==true){
azman.setkonum(0*40, 5*40);}
else if(yerazman.equals("Karakter:Azman,Kapi:D")==true){
azman.setkonum(3*40, 10*40);}
```

Resim 3

Çizdirme işlemleri için *override* ettiğimiz *paint* metodunu kullandık. Labirenti çizdirirken labirent matrisinin satır ve sütunlarını iç içe for döngüsüne aldık. Boyama işlemi için switch case yapısı kullandık. Matris sıfıra eşitse griye, bire eşitse beyaza boyadık (Resim 4). Aynı yerde *drawImage* metoduyla okları, karakter resimlerini, *drawString* metoduyla giriş kapılarının harflerini çizdirdik.

```
@Override
public void paint(Graphics g) {
    super.paint(g);
    g.translate(100, 200);

    // labirenti çizdir
    for (int satir = 0; satir < labirent.length; satir++) {
        for (int sutun = 0; sutun < labirent[0].length; sutun++) {
            Color color;
            switch (labirent[satir][sutun]) {
                case 0 : color = Color.GRAY;
                break;
                case 1 : color = Color.WHITE;
                break;
                default : color = Color.BLUE;
            }
            g.setColor(color);
            g.fillRect(40 * sutun, 40 * satir, 40, 40);
            g.setColor(Color.BLACK);
            g.drawRect(40 * sutun, 40 * satir, 40, 40);


        }
    }
```

Resim 4: Labirenti çizdirme

Altın ve mantarların oyun içinde rastgele pozisyonlarda çıkması için *Lokasyon* sınıfını kullanarak *konum* isimli bir *arraylist* oluşturduk. Altın ve mantarların çıkabileceği olası yerleri, yani matrisin *1* değerini verdiği bölgeleri bu *arrayliste* ekledik. Daha sonra bir *int counter* değişkeni oluşturduk ve proje çalıştığı sürece bu değişkeni arttırdık. Counter değeri bine ulaştığında *arraylisti Collections.shuffle* metoduyla karıştırdık ve *counter* değişkenini sıfırladık. Bu sayede her beş saniyede bir altın ve mantarların konumu değişmiş oldu fakat sürekli ekranda altın ve mantarlar bulunuyor konumda oldu(Resim 5).

```
counter++;

if(counter>1000){
Collections.shuffle(konum);
counter=0;
}
```

Resim 5

Tembel şirin ve gözlüklü şirinin klavyeden hareketini sağlamak için *override* ettiğimiz *processKeyEvent* metodunu kullandık. Seçilen oyuncu hangisiyse ok tuşlarına basıldığında onun konumu değişirdik (Resim 6).

```
@Override
protected void processKeyEvent(KeyEvent ke) {
    if (ke.getID() != KeyEvent.KEY_PRESSED) {
        return;
    }

    if (ke.getKeyCode() == KeyEvent.VK_RIGHT) {
        if(secilenoyuncu=="gozluklu"){
        if(labirent[y][x+2]!=0 && labirent[y][x+1]!=0){
        x += 2;
          gozluklu.setkonum(y*40, x*40);
        }

       }
        else if(secilenoyuncu=="tembel"){
        if(labirent[y][x+1]!=0){
        x += 1;
          tembelsirin.setkonum(y*40, x*40);
        }
        }
         System.out.println(y+" " + x + " "+ labirent[y][x]);
```

Resim 6: Sağ ok tuşu için konum değiştirme

Oyuncuların mantar ve altınlara geldiğinde puanlarının artması için *if* bloklarını kullandık. Eğer altın ve oyuncunun konumlarının x ve y değerleri birbirine eşitse *abstract* Oyuncu sınıfında oluşturduğumuz ve Puan sınıfında override ettiğimiz *abstract* bir metod olan *PuaniGoster* metoduna (Resim 8) altına değmişse *5* değerini, mantara değmişse *50* değerini gönderdik ve puanı arttırmış olduk (Resim 7). Her değme işleminden sonra altın ve mantarların konumlarının değişmesi için *arraylisti* tekrar karıştırdık. Aynı işlemi gargamele değdiğinde metoda *-15* değerini, azmana değdiğinde -5 değerini göndererek yaptık. *PuaniGoster* metodunda elde ettiğimiz puan değerini *drawString* metoduyla oyun ekranında güncel olarak gösterdik.

```
if((gozluklu.getkonum_x()==mantar.getMantarkonum_y() &&
        gozluklu.getkonum_y()==mantar.getMantarkonum_x())
        || (tembelsirin.getkonum_x()==mantar.getMantarkonum_y()
        && tembelsirin.getkonum_y()==mantar.getMantarkonum_x()) ){

puan.PuaniGoster(50);
Collections.shuffle(konum);


}
```

Resim 7

```
@Override
public int PuaniGoster(int x) {
    puan+=x;
    return puan;
}
```

Resim 8

Oyun ilerleyişinde eğer oyuncu şirinenin önündeki kareye ulaşmışsa bir başarı sekmesi oluşturduk. Oyuncu şirineye ulaşamadan puan değeri sıfıra eşitlenmiş veya sıfırın altına düşmüşse bir kaybetme sekmesi oluşturduk (Resim 9).

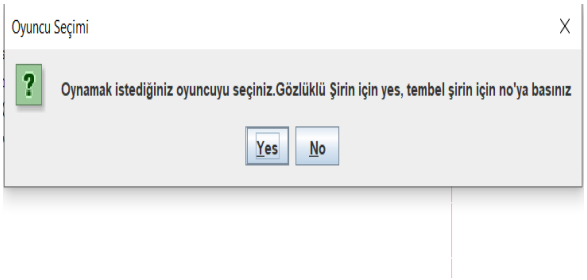
```
if((gozluklu.getkonum_x()==7*40 && gozluklu.getkonum_y()==12*40)
|| (tembelsirin.getkonum_x()==7*40 && tembelsirin.getkonum_y()==12*40) ){
JLabel message=new JLabel("Tebrikler Şirineyi Kurtardın!!!");
message.setFont(new Font("Arial",Font.BOLD,35));
JOptionPane.showMessageDialog(null, message);
}

if(puan.PuaniGoster(0)<=0){
JLabel message=new JLabel("Başaramadın :( ");
message.setFont(new Font("Arial",Font.BOLD,35));
JOptionPane.showMessageDialog(null, message);
}
```

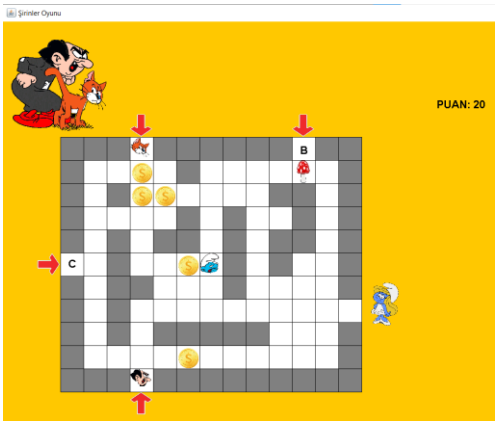
Resim 9: Sonuç yazısı

Oyun ilerleyişinde gargamel ve azmanın hareketini gösteremedik çünkü dijkstra algoritmasını araştırmamıza ve yazmaya çalışmamıza rağmen projemize entegre edemedik.

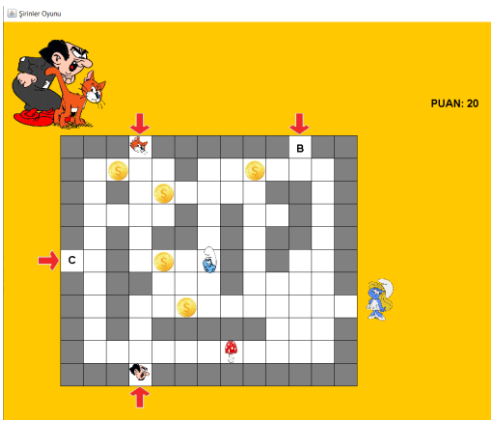
IV. Deneysel sonuçlar



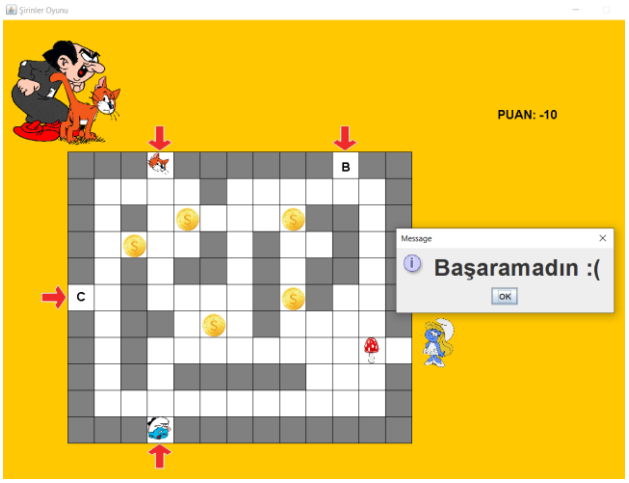
Resim 10: Oyuncu seçim ekranı



Resim 11: Tembel şirin seçildiğinde başlangıç ekranı



Resim 12: Gözlüklü şirin seçildiğinde başlangıç ekranı



Resim 13: Puan sıfırın altına düştüğünde sonuç ekranı

Resim 14: Çıkış karesine gelindiğinde sonuç ekranı

V. Sonuç

Bu projede bizden beklenen veri yapıları algoritmalarını kullanmak ve nesneye yönelik programlama becerilerimizi geliştirmekti. Nesneye yönelik programlama kısmını yapabilsek de algoritma geliştirme kısmında istediğimiz başarıyı elde edemedik.

VI. Yalancı kod

1. Başla
2. Oyuncu seç
3. Dosyayı oku
4. Labirenti çizdir
5. Ok tuşlarına basılırsa konumu değiştir
6. Altına değerse puanı beş arttır
7. Mantara değerse puanı elli arttır
8. Gargamele değerse puanı on beş azalt
9. Azmana değerse puanı beş azalt
10. Puan sıfırın altına düşerse kaybetme sekmesi aç

11. Çıkış karesine gelinirse kazanma sekmesi aç
12. Bitir

VII. Kaynakça

https://www.happycoders.eu/algorithms/shortest-path-algorithm-java/

https://www.youtube.com/watch?v=az4nbeppSdA

https://www.youtube.com/watch?v=0XPoKzGLIa0

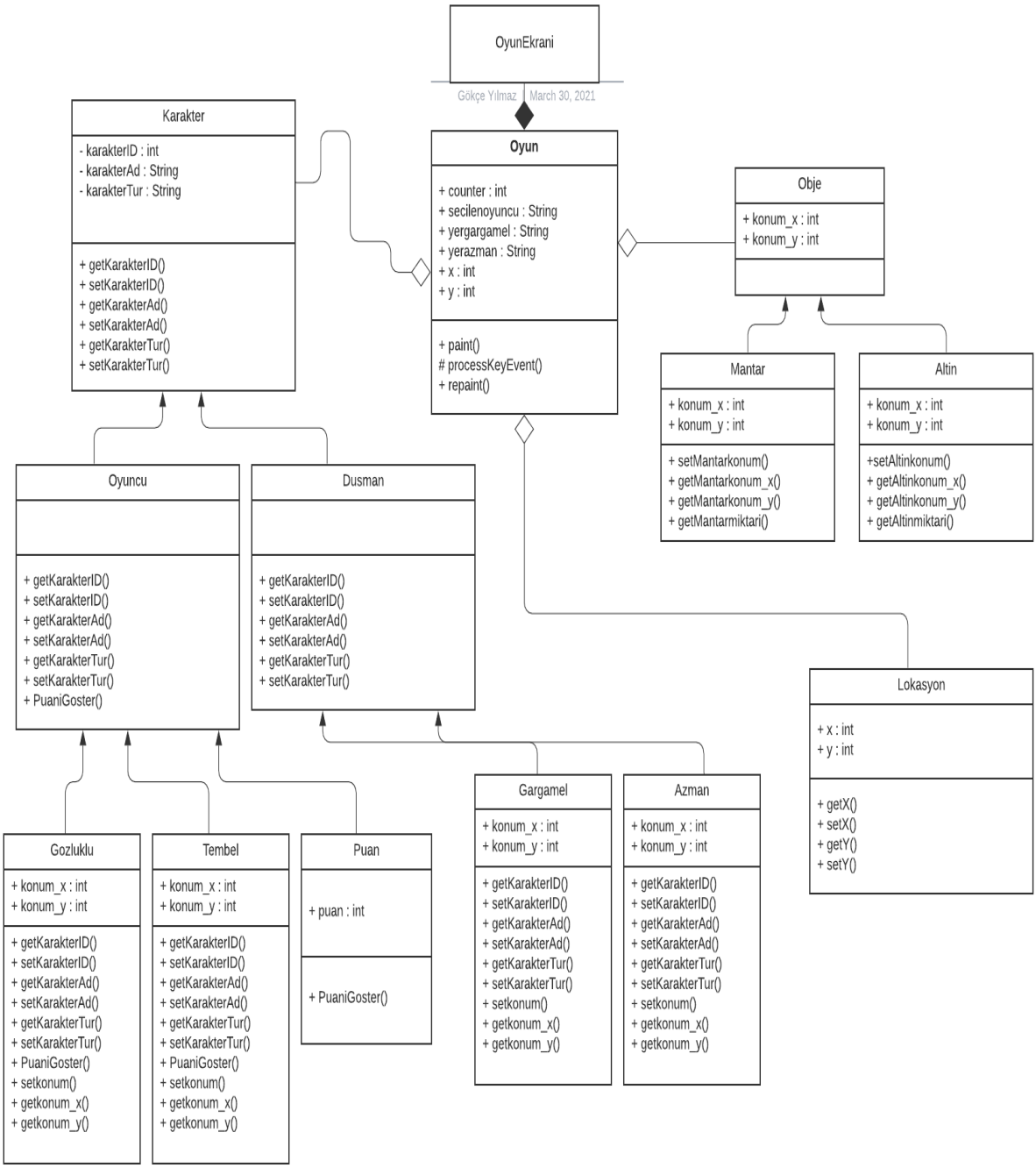
https://medium.com/t%C3%BCrkiye/graf-teorisi-4-en-k%C4%B1sa-yol-problemi-322a648c864e

https://stackoverflow.com/questions/21815839/simple-java-2d-array-maze-sample

https://www.codota.com/code/java/methods/java.awt.Component/getFont

https://www.baeldung.com/java-dijkstra

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


OyunEkrani
Gökçe Yılmaz
March 30, 2021
Karakter
- karakterID : int
- karakterAd : String
- karakterTur : String
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
Oyun
+ counter : int
+ secilenoyuncu : String
+ yergargamel : String
+ yerazman : String
+ x : int
+ y : int
+ paint()
# processKeyEvent()
+ repaint()
Obje
+ konum_x : int
+ konum_y : int
Mantar
+ konum_x : int
+ konum_y : int
+ setMantarkonum()
+ getMantarkonum_x()
+ getMantarkonum_y()
+ getMantarmiktari()
Altin
+ konum_x : int
+ konum_y : int
+setAltinkonum()
+ getAltinkonum_x()
+ getAltinkonum_y()
+ getAltinmiktari()
Oyuncu
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
+ PuaniGoster()
Dusman
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
Lokasyon
+ x : int
+ y : int
+ getX()
+ setX()
+ getY()
+ setY()
Gozluklu
+ konum_x : int
+ konum_y : int
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
+ PuaniGoster()
+ setkonum()
+ getkonum_x()
+ getkonum_y()
Tembel
+ konum_x : int
+ konum_y : int
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
+ PuaniGoster()
+ setkonum()
+ getkonum_x()
+ getkonum_y()
Puan
+ puan : int
+ PuaniGoster()
Gargamel
+ konum_x : int
+ konum_y : int
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
+ setkonum()
+ getkonum_x()
+ getkonum_y()
Azman
+ konum_x : int
+ konum_y : int
+ getKarakterID()
+ setKarakterID()
+ getKarakterAd()
+ setKarakterAd()
+ getKarakterTur()
+ setKarakterTur()
+ setkonum()
+ getkonum_x()
+ getkonum_y()