***KOCAELİ ÜNİVERSİTESİ***

***BİLGİSAYAR MÜHENDİSLİĞİ(iö)***

***ŞİRİNLER LABİRENT OYUNU PROJESİ***

Ali Atilla Aydemir Mehmet Alperen Onarır

***180202063*** 180202083

***atilla-734@hotmail.com*** [mehmetalperen95@gmail.com](mailto:mehmetalperen95@gmail.com)

***ÖZET***

Bu proje ile nesneye yönelik programlama ve veri yapıları algoritmalarını kullanarak Şirinler oyunu tasarlanması amaçlanmıştır. Seçilen oyuncunun Labirent içerisinde puanlarını bitirmeden önce Şirine’ye ulaşması gerekmektedir. Kullanıcı oyun başlamadan önce seçtiği oyunculardan birini klavye yardımı ile kontrol edecek ve Şirine’ye ulaştırmaya çalışacaktır. Düşman karakterler ise oyuncuyu Dijikstra Algoritması yardımı ile yakalamaya çalışacaktır.

***GİRİŞ***

***Java dili, Netbeans IDE ‘si ve JFRAME arayüz kütüphanesi yardımı ile yazılmış programımızda –tuşlarla oyuncu kontrolü için KeyEvent sınıfı ve bu sınıfın getirdiği VK\_RIGHT, VK\_LEFT, VK\_DOWN, VK\_UP özellikleri kullanılır. en kısa yol bilgisine ulasmak için dijisktra algoritmasından faydalanılmıstır.***

***YÖNTEM***

***Sınıf hiyerarşisi üzerine kurulan oyun mimarisi ana sınıf olan soyut karakter sınıfının temel ıd,lokasyon, vb bilgileri ile olusturulması ile baslar. düşman ve oyuncu sınıfları karakter sınıfından kalıtım alırlar ve getter setter marifetleri ile ve kendilerine has parametrelerle özelleştirilirler.***

***Seçim ekranı butonlar ile tembel veya gözlüklü olarak oyuncuya karakterini seçme sansı sunar. Oyuncu sınıfından türetilen tembel ve gözlüklü sınıfları özelleştirilmiştir. (örn adımSayisi) oyuncu karakteri sabit baslangıc noktasında baslamak üzere oyuncuObje.setlokX() gibi metotlarla oyuna hazırlanır.***

***Dışarıdan 0 ve 1 ler seklinde kodlanan labirent bilgisi ve string halde sunulan düşman ve düşmanın başlangıc durum bilgileri okunarak labirent çizimi sağlanır. Düşman okunan lokasyonlarda yerini almıs ve en kısa yol algoritması ile oyuncunun güncel lokasyonuna göre en kısa mesafeyi bulur, ekrana çizer ve sırası geldiğinde hamlesini yapar.***

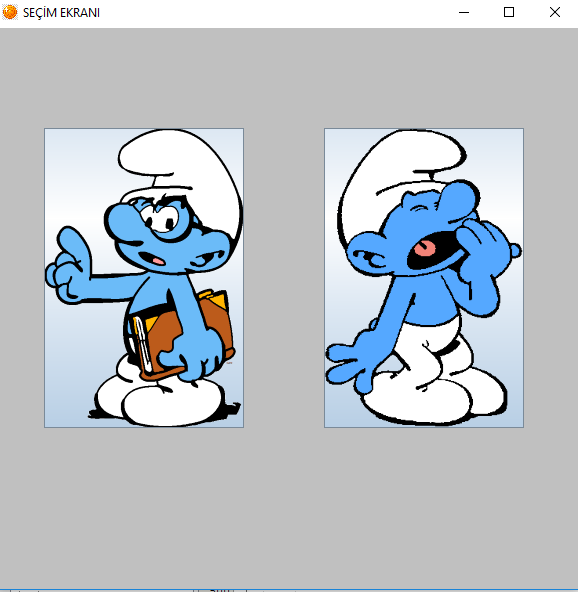
***Oyun view sınıfı içerisinde oynanmakta main metodumuzda dosya okuma islemleri yapılmaktadır.***

***Puan durumu güncel olarak hesaplanmakta, ve ekrana yansıtılmaktadır. Oyun şirineye ulasıldıgını veya puanın 0 a düştüğünü yakalarsa sona ermektedir.***

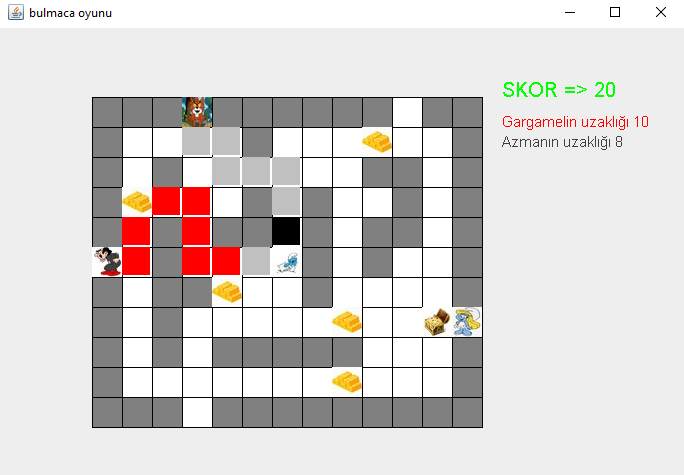
***Karşılaşılan promlem; KeyEvent.VK\_LEFT ile gerçekleştirilen tuş algılama fonksiyonu tusun basılması ve cekilmesi olarak iki kere calısmaktadır. Bir sayac yardımı ile tus basılınca artan sayac içerisinde sadece cekildiğinde sayacın%2==0 kosulu saglandıgı için calısan komut blogu aktif edilmistir.***

***DENEYSEL SONUÇLAR-***

***Seçim Ekranı:***



***Labirent bilgisinin, 2 düsman karakterin ve baslangıc kordinatlarının okunması sonrası baslangıc durumu***



***Oyun içinde sirineye ulasıldıgında kazanma durumu***



***YALANCI KOD***

***START***

***int flag=0; // tuslardaki çift algılama sorunu için***

***int hamleSirasi=0; // tek rakamlarda pc, çift rakamlarda oyuncu oynuyor, static int adimTemp=0; // gozluklunun 2 adım atmasını saglamak için***

***static int dusmanDurumu; // 0 azman, 1 gargamel , 2 azman + gargamel***

***static boolean OyunDurumu=true; // oyun bitince false donecek***

***static boolean mazeGosterilsinMi=false;***

***static int [][] maze= new int[11][13];***

***static Lokasyon[][] lokasyon = new Lokasyon[11][13];***

***create oyuncuObje, tembelObje, gözlukluObje, azmanObje, altınObje, gargamelObje,hazineObje,oyuncuGoruntu***

***main()***

***Baslangic() JFrame frame=new JFrame("SEÇİM EKRANI");***

***JButton gozluklu=new JButton();***

***JButton tembel=new JButton();***

***actionPerformed(ActionEvent e)***

***tembel ( secim=0 ; mazeGosterilsinMi=true; )***

***gozluklu ( secim=1 ; mazeGosterilsinMi=true; )***

***frame.setVisible(false)***

***if secim=0 or 1 oyuncuObje = gozlukluObje or tembelObje;***

***oyuncuGoruntu = tembel.img or gozluklu.ımg***

***oyuncuObje.setBaslangicNoktasi(5,6)***

***start and stop hazineTimer and altınTimer for visibility***

***open file and read maze and enemy enformation***

***azmanStartX,azmanStartY***

***View view = new View();***

***Viev()***

***kare k = new kare();***

***yol yolBulucu = new yol(maze);***

***int uzaklikG=0;***

***int uzaklikA=0;***

***for(int row =0; row<maze.length;row++) {***

***for(int col=0;col<maze[0].length;col++) {***

***switch(maze[row][col]){***

***case 1: color = Color.WHITE; break;***

***case 0: color = Color.gray; break;***

***g.fillRect(30\*col,30\*row, 30, 30);***

***if(altinVarlik){***

***g.drawImage(altinGoruntu,altin.getX()\*30,altin1.getY()\*30,this);***

***if(hazineVarlik)***

***{g.drawImage(hazineGoruntu,hazine1.getX()\*30,hazine1.getY()\*30,this); }***

***if( hamleSirasi%2==0){***

***switch(dusmanDurumu){***

***case 0 only azman***

***yolBulucu.rotaHesaplaHazırlık(azmanObje.getLokY(),azmanObje.getLokX());***

***uzaklikA = yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getAgirlik();***

***k = yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getOnceki();***

***for(int aa=0; aa<yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getAgirlik() ; aa++){***

***g.fillRect(30\*k.getY(),30\*k.getX(), 28, 28); g.setColor(Color.LIGHT\_GRAY);***

***if(k.getOnceki()==null) {break;} else {k = k.getOnceki();}}break;}***

***case 1 only gargamel***

***case 2 azman + gargamel***

***//oyuncu düsmanın üstüne gelirse***

***if(oyuncuObje.getLokX()==azmanObje.getLokX() && oyuncuObje.getLokY()==azmanObje.getLokY()){***

***azmanObje.setLokX(azmStartX);azmanObje.setLokY(azmStartY);***

***puanObje.PuanGuncelle(oyuncuObje, azmanObje);***

***hamleSirasi++;}***

***// sirineye ulasılırsa***

***if(oyuncuObje.getLokX()==12 && oyuncuObje.getLokY()==7){***

***g.setColor(Color.CYAN);***

***g.setFont(new Font("TimesRoman", Font.PLAIN, 20));***

***g.drawString(" Kahramanım!!", 400, 210);***

***g.setColor(Color.GREEN);***

***g.setFont(new Font("TimesRoman", Font.PLAIN, 25));***

***g.drawString(" KAZANDIN!!", 150, -20);***

***OyunDurumu=false;***

***}***

***Enemy move***

***if(hamleSirasi%2==1){***

***switch(dusmanDurumu){***

***case 0:{***

***yolBulucu.rotaHesaplaHazırlık(azmanObje.getLokY(),azmanObje.getLokX());***

***uzaklikA=yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getAgirlik()-1;***

***k = yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getOnceki();***

***for(int aa=azmanObje.getAdimSayisi()-1; aa<yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getOnceki().getAgirlik() ; aa++){***

***g.fillRect(30\*k.getY(),30\*k.getX(), 28, 28); g.setColor(Color.LIGHT\_GRAY);***

***if(k.getOnceki()==null) {break;} else {k = k.getOnceki();}}***

***k = yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()];***

***for(int z=azmanObje.getAdimSayisi(); z<yolBulucu.kareMatrisi[oyuncuObje.getLokY()][oyuncuObje.getLokX()].getAgirlik(); z++){k=k.getOnceki();}***

***azmanObje.setLokX(k.getY());azmanObje.setLokY(k.getX());***

***hamleSirasi++;break;}***

***if same localion oyuncu and altın or hazine***

***puanıGuncelle(oyuncuObje,altın or hazine)***

***// puan 0 olursa bitirir***

***if(puanObje.PuaniGoster(oyuncuObje)<=0){ OyunDurumu=false;***

***g.setColor(Color.red);***

***g.setFont(new Font("TimesRoman", Font.PLAIN, 25));***

***g.drawString(" KAYBETTİN!!", 150, -20);***

***}***

***g.drawImage sirine,enemy,oyuncuGoruntu.getLokX()\*30***

***protected void processKeyEvent(KeyEvent e)***

***{***

***if(e.getKeyCode()==KeyEvent.VK\_LEFT && OyunDurumu==true){***

***flag++;if(flag%2==0) {System.out.println("solll");***

***adimTemp++ ;oyuncuObje.setLokX(oyuncuObje.getLokX()-1);***

***if(oyuncuObje.getLokX() <0){oyuncuObje.setLokX(0);} // sol kapından sola kaçmaması için***

***if( lokasyon[oyuncuObje.getLokY()][oyuncuObje.getLokX()].lokasyonKontrol()==1){repaint(); }***

***else { oyuncuObje.setLokX(oyuncuObje.getLokX()+1);repaint();}}***

***if(oyuncuObje.getAdimSayisi()-adimTemp==0){ hamleSirasi++; adimTemp=0;repaint();}***

***}***

***KARMAŞIKLIK ANALİZİ :***

***kareMatrisi();***

***for (int i = 0; i < 11; i++) {***

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

***kareMatrisi[i][j] = new kare();***

***kareMatrisi[i][j].setX(i);kareMatrisi[i][j].setY(j);***

***} } }***

***11\*13\*(1+1+1)***

***costmatrishesapla();***

***for (int i = 0; i < 11; i++) {***

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

***for (int k = 0; k < 11; k++) {***

***for (int l = 0; l < 13; l++) {***

***costMatris[i][j][k][l][0] = 999;***

***if(j==l && k==i+1 && haritaMatrisi[k][l] == 1){costMatris[i][j][k][l][0]=1;} // alt***

***if(j==l && i==k+1 && haritaMatrisi[k][l] == 1){costMatris[i][j][k][l][0]=1;}***

***if(i==k && l==j+1 && haritaMatrisi[k][l] == 1){costMatris[i][j][k][l][0]=1;}***

***if(i==k && j==l+1 && haritaMatrisi[k][l] == 1){costMatris[i][j][k][l][0]=1;} //SOL***

***if(i==k && j==l && haritaMatrisi[k][l] == 1 ){costMatris[i][j][k][l][0]=0;} // kendi ise***

***} }} }}***

***11\*13\*11\*13\*5\*(1+1 +1 +1 +1)***

***rotaHesaplaHazırlık()***

***kareMatrisiTemizle();***

***rotaHesapla(startx, starty)***

***kareMatrisiTemizle();***

***for (int i = 0; i < 11; i++) {***

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

***if(haritaMatrisi[i][j]==0){kareMatrisi[i][j].setAgirlik(-1); kareMatrisi[i][j].setKullanildiMi(true);}***

***else{kareMatrisi[i][j].setAgirlik(999);kareMatrisi[i][j].setKullanildiMi(false);kareMatrisi[i][j].setOnceki(null); }***

***}}}***

***11\*13\*(1+1+1)***

***RotaHesapla()***

***kareMatrisi[startx][starty].setKullanildiMi(true);***

***for (int i = 0; i < 11; i++) { // costmatriste gidebileceği her nokta bulunuyor ve güncelleniyor***

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

***if (kareMatrisi[i][j].isKullanildiMi()==false***

***&& kareMatrisi[startx][starty].getAgirlik()+ costMatris[startx][starty][i][j][0] < kareMatrisi[i][j].getAgirlik()***

***){***

***kareMatrisi[i][j].setAgirlik(kareMatrisi[startx][starty].getAgirlik() + costMatris[startx][starty][i][j][0]) ;***

***kareMatrisi[i][j].setOnceki(kareMatrisi[startx][starty]);***

***}}}kareMatrisMinBul(); }***

***11\*13\*(1+1+1+1+1+1)***

***KareMatrisMinBul***

***int flag=0;***

***int temp =999;***

***int tempx = 0,tempy = 0;***

***for (int i = 0; i < 11; i++) {***

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

***if(kareMatrisi[i][j].isKullanildiMi()==false && haritaMatrisi[i][j]==1) flag++;{}***

***if(kareMatrisi[i][j].isKullanildiMi()==false && kareMatrisi[i][j].getAgirlik() < temp ){***

***tempx=i;tempy=j;***

***temp=kareMatrisi[i][j].getAgirlik();***

***}}}***

***if(flag==0){return;}***

***rotaHesapla(tempx,tempy);***

***}***

***11\*13\*(1+1+1+1+1+1+1+1)***

***Dijikstra Algoritması big O notasyonu algoritmanın rekrüsif oldugu noktanın n degerinin max noktaya ulaşması sebebi ile bu noktada belirlenir.***

***RotaHesapla – kareMatrisMinBul metotları arasında belirlenen big O değeri logaritmik seyretmektedir.***

***Metot düğüm değeri kadar tekrar edecektir .burada 11\*13 düğüm değerini temsil etmektedir M = düğüm sayısı kabul edilecek olursa***

***Her tekrar için costMatriste mevcut yol sayısı kadar hesaplama yapma gereği ortaya cıkmaktadır. Fakat algoritma çalıştıkça kullanılmıs değerler isaretlenmekte ve her adımda adım sayısı azalmaktadır. Buda logaritmik bir azalmaya/artmaya sebep olmaktadır.***

***Dolayısıyla m düğüm ve n kenar için dijikstra***

***big O M log N olarak hesaplanmıstır.***

***Zaman karmaşıklığı hesabı =***

***+11\*13\*(1+1+1) +***

***+ 11\*13\*11\*13\*5\*(1+1 +1 +1 +1)***

***+11\*13\*(1+1+1)***

***+ 11\*13\*(1+1+1+1+1+1) \* 11\*13\*(1+1+1+1+1+1+1+1)***

***429 +*** 511225 + 429 + 981552 = 1493635

***SONUÇ :***

***tembel ya da gözlüklü şirini seçebildiğimiz bir ekranla oyuna başlarız. seçimimizi yaptıktan sonra oyun ekranı(görsel harita) karşımıza çıkar. bu ekran dosya bilgilerine göre dinamik olarak okutulur.***

***1 ve 0 durumuna göre duvarların ve geçilebilecek patikaların yazdırılması sağlanır. bunun yanında okunan string değerlere göre azman ve gargamelin ya da azman veya gargamelin oyuna***

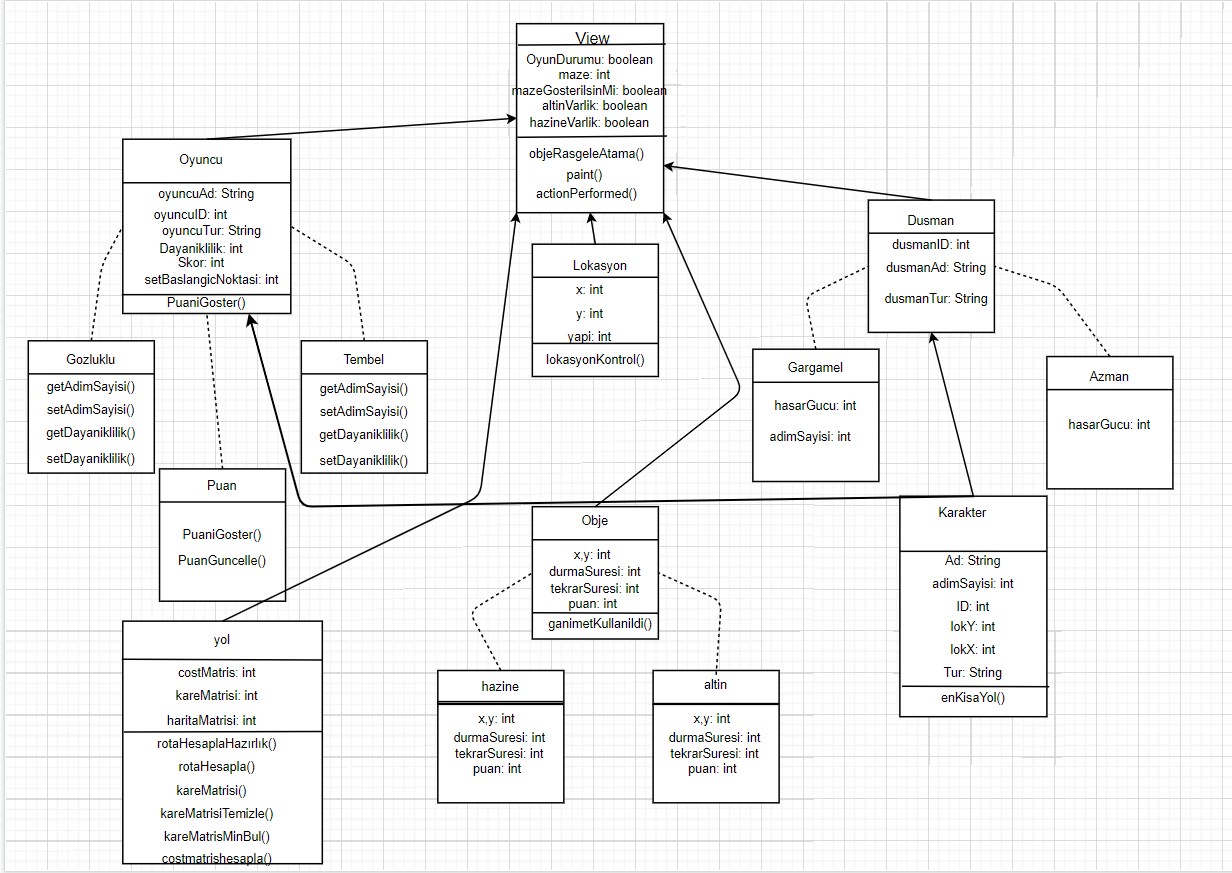
***başlamasına karar verilir. azman tek birim, gargamel ise 2 birim hareket eder. zamana bağlı olarak altınlar ve hazine ekranda belirip kaybolur. puanlar dinamiktir, hazinenin kaç puan***

***ifade ettiği kod üzerinde değişebilecek şekilde tasarlanmıştır. her puan karakterimizin yaşaması için ek puan sağlayacaktır. dijkstra algoritmasıyla bizi en kısa sürede yakalamaya çalışan***

***düşman karakter/karakterler bize ulaşmadan şirineye ulaşılması durumunad oyun kazanılır. aksi takdirde oyun kaybedilir. programda abstract sınıflar, get ve setot metotları, encapsülation gibi***

***nesneye yönelik konseptlerin hepsi kullanılmıştır. 13 adet sınıf view'e bağlıdır ve bu sınıf üzerinden programın işleyişi kontrol edilir.***

***Uml sınıf diagramı***



**KAYNAKÇA**

1. <https://thrkardak.medium.com/javascript-ne-ola-ki-bu-zaman-karma%C5%9F%C4%B1kl%C4%B1%C4%9F%C4%B1-ve-b%C3%BCy%C3%BCk-o-notasyonu-dedikleri-%C5%9Feyler-c84648b93072>
2. https://thrkardak.medium.com/javascript-ne-ola-ki-bu-zaman-karma%C5%9F%C4%B1kl%C4%B1%C4%9F%C4%B1-ve-b%C3%BCy%C3%BCk-o-notasyonu-dedikleri-%C5%9Feyler-c84648b93072
3. https://en.wikipedia.org/wiki/Dijkstra%27s\_algorithm
4. https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html
5. https://www.w3schools.com/java/java\_abstract.asp
6. https://www.journaldev.com/875/java-read-file-to-string