

PROGRAMLAMA LABORATUVARI 2 PROJE 1

1st Gülsüm Nur Maslak
Kocaeli üniversitesi

Kocaeli/Türkiye
220201033

2nd Selim Eren Kaya
Kocaeli üniversitesi

Kocaeli/Türkiye
230201127

I. ÖZET

Bu projede random haritalar oluşturup, karakteri otonom bir şekilde yürütmeye çalışıp haritadaki hazine sandıklarını en kısa yolu kullanarak toplayan bir kod yazmamız isteniyor. Bu raporda proje tanıtımı, araştırmalar ve yöntemi, tasarım, deneysel sonuçlar, sonuç ve kaynakça kısımları bulunmaktadır. Bunlara ek olarak kod çıktıları, UML diyagramı ve yalancı kod bulunmaktadır.

II. GİRİŞ

Proje karakter tarafından hazine sandıklarının en kısa yoldan toplanabilmesini algoritma kullanarak yapmayı amaçlamaktadır. Random oluşturulan haritalar da random şekilde yerleştirilen engeller oluşturarak arayüz de tasarlanması istenmiştir. Arayüzün yaz ve kış olmak üzere iki farklı teması vardır.

III. YÖNTEM

Proje başlarken ilk başta hangi dil ile yazabiliriz diye düşündük ve bizim için en uygun olarak gördüğümüz java dilini seçtik. Arayüz için swing kullandık. Kısa yol algoritması içinde A* algoritmasını kodumuza entegre ettik.

A. *public class Character*

Bu class karakter id, ad, konum ilemlenen konum değişkenlerini tutar. Gui değişkenlerini, animasyon değişkenlerini, sandık ararken kullanılacak metodlar ve değişkenleri de tutar. Get set değişkenleri de buradadır

B. *public void gorulenNesneleriKaydet ()*

Bu metodun amacı, karakter 7x7'lik alanı tarar. Eğer alanda altın, gümüş, zümrüt, bakır sandıklar varsa onları bulur. Mesela gümüş sandık bulursa ekrana "GÜMÜŞ SANDIK BULUNDU ! konumu : [][] " şeklinde hangi koordinat noktasında bulduysa yazdırılır.

C. *public void enKisaYol()*

Hedef konuma ulaşmak için shortestPath nesnesini kullanarak ulaşılması gereken lokasyonları belirler. Bu sayede karakterin konuma en kısa yoldan gitmesini sağlar.

D. *public void goToLocation()*

Karakterin belirli bir konuma gitmesini sağlar. Karakter hareket edince de karakterin konumu, ziyaret edilen konumlar, koordinatlar, görülen nesneler ve diğer bilgiler güncellenir.

E. *public void gezilebilecekKareleriGuncelle()*

Karakterin gezebileceği kareleri günceller. Karakterin engelleri aşarak hedef konuma ulaşmasını sağlar.

F. *public void baslangicDegerleriOlustur()*

Oyun ile ilgili başlangıç değerlerini tutar. Ekran üzerindeki başlangıç konumu, haritanın başlangıç konumunu karakterin hızı ve diğer değerlerin belirlenmesini sağlar.

G. *public void update()*

Oyunun karakterin hareketlerini ve durumlarını güncellemek için kullanılır. Oyun giriş ekranındayken karakterin kıpırdama işlemleri kontrol edilir. Eğer oyun durumu "giris" ise, klavye girişlerine göre giriş ekranının konumunu günceller. Harita oluşturma aşamasında karakterin yapabileceklerine bakılır. Eğer oyun durumu "olusturma" ise, klavye girişlerine göre karakterin dünya üzerindeki konumunu günceller. Oyun aşamasındaysa karakterin neler yapabileceğine bakılır. Eğer oyun durumu "oyun" ise, karakterin etrafındaki engelleri bulma ve kaydetme, gezilmiş, ve gezilebilecek kareleri güncelleme, sandıklara doğru hareket etme ve toplama, tüm sandıkların toplandığı kontrol edilir. Karakterin rastgele gezinmesine de bakılır. Bitis, aşamasında karakterin yapabilecekleri kontrol edilir. Eğer oyun durumu "bitis" ise, klavye girişlerine göre karakterin konumunu günceller. Ayrıca her aşamada karakterin animasyonu ve hızı güncellenir.

H. *public void draw*

Oyun karakterinin çizimini gerçekleştirmek için kullanılır. İlk metod karakterin koordinatlarına ve oyun panelini tile boyutuna göre çizim yaparken, ikinci metod istenilen genişlik ve yükseklik değerlerine göre çizim yapılmasını sağlar.

I. *public void ilerlenenKareleriCizdir()*

Karakterin geçtiği konumları belirli bir renkle görselleştirmeye çalışır

J. *Character()*

Constructor metodudur. Karakter id, adını ve konumu tutar.

K. *public class Location*

Lokasyon sınıfı, bir konumun koordinatlarını, bölgesini, engel durumunu ve konumda bulunan nesneyi temsil eder.

L. public class KeyHandler

Bu class KeyHandler adında bir sınıfı temsil etmektedir ve KeyListener arabirimini uygular. Bu sınıf, klavye olaylarına bakarak kullanıcının tuşlara basma ve tuşları bırakma işlemlerini takip eder. Sınıf içinde bulunan alanlar arasında upPressed, downPressed, leftPressed, rightPressed yer almaktadır. Bu alanlar, yön tuşlarının basılıp basılmadığını tutan boolean değerlerdir. Sınıf içinde bulunan metotlar arasında KeyHandler(GuiPanel gp) constructor metodu, keyTyped(KeyEvent e), keyPressed(KeyEvent e) ve keyReleased(KeyEvent e) metotları da vardır. Constructor metodu, bir GuiPanel nesnesi olarak sınıfın bir örneğini oluştururken, diğer metotlar klavye olaylarına göre belirli işlemleri gerçekleştirmektedir. keyPressed metodu, klavyeden bir tuşa basıldığında tetiklenir ve basılan tuşa göre belirli işlemler yapar.

M. public class Gui extends JFrame()

JFrame sınıfını extends ederek pencere oluşumunu sağlar. Bu classta gui için get ve set metotları yer alır.

N. public GUI()

Gui sınıfının constructor metodu, çeşitli parametreler olarak bir GUI oluşturmayı amaçlar. Bu parametreler arasında lokasyonlar, kare sayıları, karakter nesnesi, engel listesi, dinamik engel listesi ve sandık listesi bulunur. Constructor içinde bu parametrelerle ilgili sınıfın özelliklerine atanır. Oluşturulan JFrame'in kapatılma şekli ve boyutu ayarlanır, başlığı belirlenir. GuiPanel adında bir panel oluşturulur ve constructor parametreleriyle birlikte eklenir. Son olarak, JFrame uygun boyuta getirilir, merkezlenir ve görünür hale getirilir. Paneldeki oyun başlatma işlemi için startGameThread() metodu çağrılır. Bu sayede belirtilen parametrelerle bir GUI oluşturulur ve oyun başlatılmaya hazır hale getirilir.

O. class GuiPanel extends JPanel implements Runnable()

Oyunun grafiksel arayüzünü kontrol etmek için kullanılır. Oyunun durumunu belirleyen değişkenler, ekranlar arasında geçiş yapmayı sağlayan String değişkenler ve mouse için değişkenler içerir. Ayrıca, font nesnesi, oyun nesneleri (karakter, engel listesi, sandık listesi, dinamik engel listesi), ekran ayarları için çeşitli int değişkenler ve dünya değişkenleri (yatay ve dikey kare sayısı, genişlik ve yükseklik) tanımlanmıştır.

P. public void startGameThread()

Ana oyun döngüsünü başlatmak için kullanılır.

Q. public void run()

FPS değerine göre oyun nesnelerinin güncellenmesi ve ekrana çizilmesi işlemlerini yapar. Delta değişkeni zaman farkını takip eder ve belirli bir değeri aştığında oyun nesneleri güncellenir ve ekrana çizilir.

R. public void update()

Oyunun akışı için gerekli güncellemeleri yapar.

S. public void paintComponent(Graphics g)

Oyunun farklı durumlarına göre ekranın nasıl çizileceğini belirler

T. public void girisEkranıCiz(Graphics2D g2)

Metin fontu belirlenir ve ardından arka plan ve ön plan resimleri ekran üzerine çizilir. Karakterin belirli bir konuma çizilmesi sağlanır. MouseListener ile mouse ayarları kontrol edilir. Eğer kullanıcı butona tıklarsa ve belirli bir alanı doğru şekilde tıklarsa, oyun durumu "olusturma" olarak değiştirilir ve harita oluşturma işlemi başlatılır. Bu sayede oyunun ilerlemesi sağlanır.

U. public int getXOrtalanmisBaslik(String text, Graphics2D g2)

Yazının Ekran için ortalananmış x kordinatını döndürür.

V. public void olusturmaEkranıCiz(Graphics2D g2)

Harita oluşturma ekranının görsellerini çizmek için kullanılır. Arka plan rengi ve metin fontu ayarlandıktan sonra, başlıklar belirlenir ve ekrana yerleştirilir. Harita elemanları ve buton resmi çizilir. Fare tıklamalarını değiştiren bir MouseListener eklenir.

W. public void oyunEkranıCiz(Graphics2D g2)

Oyun ekranının görsellerini çizer. Arkaplan belirlenir, kare sınırları, karakter, engeller ve sandıklar çizilir.

X. public void oyunBitisCiz(Graphics2D g2)

Oyun bitiş ekranını günceller.

Y. public void kareSinirlariCiz(Graphics2D g2)

Karelerin sınırlarını çizdirir

Z. public void zoomInOut()

Oyunun görüntüsünü yakınlaştırmak veya uzaklaştırmak için kullanılır. Karelerin boyutunu değiştirerek oyunun genişlik ve yükseklik ölçülerini ayarlar. Karakterin dünya koordinatları da yeni boyuta göre güncellenir ve karakterin hızı da bu değişime uyum sağlar.

. public void playMusic()

Oyun müziğini ayarlar.

. public GuiPanel()

GuiPanel sınıfının constructor metodudur. Farklı parametreler olarak oyunun başlangıç ayarlamalarını yapar. Lokasyon bilgilerini saklar, font ayarlarını belirler, nesneleri atar ve panelin boyutunu ayarlar.

. public class shortestPath

A* (A star) algoritmasını kullanarak bir harita üzerinde en kısa yolu bulmayı amaçlar. shortestPath sınıfı, harita boyutunu ve karakterin başlangıç ve hedef konumlarını saklar. haritaOlustur fonksiyonu, haritayı oluşturur ve engelleri işaretler. startSearch fonksiyonu, en kısa yol arama işlemi başlatır. AStarAlgoritması iç içe sınıfı, A* algoritmasını gerçekleştirirken Node iç içe sınıfı, haritadaki düğümleri temsil eder.

. public class Main

Harita üzerinde belirli bir türdeki sabit engellerin (ağaçlar, kayalar, dağlar gibi) rastgele bir şekilde yerleştirilmesini sağlar. Öncelikle, kullanıcıdan alınan harita boyutlarına ve belirlenen minimum engel sayılarına göre harita oluşturulur. Bu harita, kış ve yaz bölgelerine ayrılır ve her bölge için belirli türde engeller yerleştirilir. Sabit engel türlerinin haritaya yerleştirilmesi için belirli kurallar ve koşullar sağlanır. Her bir engel türü için belirlenen minimum sayılar doğrultusunda rastgele yerleştirme işlemi gerçekleştirilir. Bu işlem sırasında, engellerin belirli bir bölgeye (kış veya yaz) ait olması da dikkate alınır. Engellerin yerleştirilmesi sırasında, harita üzerindeki uygun konumlar belirlenir ve engellerin birbiriyle çakışmaması sağlanır. Ayrıca, her engel türünün haritaya uygun boyutlarda yerleştirilmesi ve haritadaki diğer nesnelerle çakışmaması için kontroller yapılır. Haritaya duvar, kus, ve arı gibi dinamik engeller eklenmektedir. Duvar engelleri belirli bir minimum sayıda rastgele konumlara yerleştirilir. Her duvarın haritada kapladığı alan kontrol edilerek uygun bir konum bulunur ve engelin yerleştirilmesi sağlanır. Kuş ve arı engelleri de benzer şekilde belirli bir minimum sayıda rastgele konumlara yerleştirilir. Haritaya altın, gümüş, zümrüt ve bakır gibi farklı değerlere sahip sandıkların rastgele yerleştirilmesi işlemi gerçekleştirilmektedir. Her tür sandık için belirli bir minimum sayıda sandık rastgele konumlara yerleştirilir. Her sandık için uygun bir konum bulunana kadar denemeler yapılır. Sandıkların yerleştirileceği bölgede engel olmaması kontrol edilir ve uygun konum bulunduğu sandık oluşturulur ve haritaya eklenir. Karakter nesnesi de benzer şekilde rastgele bir konuma yerleştirilir. Son olarak, harita ve karakter bilgileriyle bir arayüz oluşturulur, böylece oyun görüntülenebilir hale gelir.

IV. DENEYSEL SONUÇLAR

A. ALGORİTMANIN YALANCI KODU

```
import java.util.ArrayList; import java.util.Hashtable;
public class Main
    public static void main(String[] args) Hashtable<String,
Location> locationHashtable = new Hashtable<>(); Character
karakter = new Character(new Location(0, 0)); Location hedef
= new Location(5, 5); int haritaBoyut = 10; ShortestPath
shortestPath = new ShortestPath(locationHashtable, karakter,
hedef, haritaBoyut);
    static class Character Location konum;
    Character(Location konum) this.konum = konum;
    public Location getKonum() return konum;
    static class Location int xCoord; int yCoord; boolean
engelDurum;
    Location(int xCoord, int yCoord) this.xCoord = xCoord;
this.yCoord = yCoord;
    public int getXCoord() return xCoord;
    public int getYCoord() return yCoord;
    public boolean getEngelDurum() return engelDurum;
    static class ShortestPath int haritaBoyut; int karakterXCord;
int karakterYCord; int hedefX; int hedefY; boolean
```

```
solving = true; Node[][] map; AStarAlgorithm Alg = new
AStarAlgorithm(); ArrayList<Node> gidilecekYol = new Ar-
rayList<>(); Location hedef; Hashtable<String, Location> lo-
cationHashtable; ArrayList<Location> gidilecekLokasyonlar =
new ArrayList<>();
```

```
    public ArrayList<Location> getGidilecekLokasyonlar()
for (Node konum : gidilecekYol) gidilecekLokasyon-
lar.add(locationHashtable.get(konum.getXCord() + "," +
konum.getYCord())); return gidilecekLokasyonlar;
```

```
    public void haritaOlustur(Hashtable<String, Location> loca-
tionHashtable) map = new Node[haritaBoyut][haritaBoyut];
for (int xCord = 0; xCord < haritaBoyut; xCord++) for
(int yCord = 0; yCord < haritaBoyut; yCord++) if (lo-
cationHashtable.get(xCord + "," + yCord).getEngelDurum())
map[xCord][yCord] = new Node(2, xCord, yCord); else
map[xCord][yCord] = new Node(3, xCord, yCord);
```

```
    map[karakterXCord][karakterYCord] = new
Node(0, karakterXCord, karakterYCord);
map[karakterXCord][karakterYCord].setHops(0);
map[hedefX][hedefY] = new Node(1, hedefX, hedefY);
```

```
    public ArrayList<Node> startSearch() if (solving) return
Alg.AStar(); return null;
```

```
    public ShortestPath(Hashtable<String, Location>
locationHashtable, Character karakter, Location hedef,
int haritaBoyut) hedefX = hedef.getXCoord();
hedefY = hedef.getYCoord(); karakterXCord =
karakter.getKonum().getXCoord(); karakterYCord =
karakter.getKonum().getYCoord(); this.haritaBoyut =
haritaBoyut; this.hedef = hedef; this.locationHashtable
= locationHashtable; haritaOlustur(locationHashtable);
startSearch();
```

```
    static class AStarAlgorithm public ArrayList<Node> AS-
tar() ArrayList<Node> oncelikliDugumSiralama = new Ar-
rayList<>(); oncelikliDugumSiralama.add(map[0][0]); return
oncelikliDugumSiralama; // Yalancı kod, gerçek bir A* algo-
ritması değil
```

```
    static class Node private int konumDurum; private int hops;
private int xCord; private int yCord; private int lastX; private
int lastY;
```

```
    public int getXCord() return xCord;
    public int getYCord() return yCord;
    public int getLastX() return lastX;
    public int getLastY() return lastY;
    public int getKonumDurum() return konumDurum;
    public int getHops() return hops;
    public void setKonumDurum(int konumDurum)
this.konumDurum = konumDurum;
    public void setLastNode(int x, int y) lastX = x; lastY = y;
    public void setHops(int hops) this.hops = hops;
    public Node(int konumDurum, int x, int y)
this.konumDurum = konumDurum; this.xCord = x; this.yCord
= y; hops = -1;
```

