T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

PROGRAMLAMA LAB II - 1. PROJE RAPORU
LABİRENT OYUNU

Hazırlayan

UMUTCAN OĞURLU

230501031

PROJE ÖZETİ

Bu proje, nesne yönelimli programlama (OOP) ve veri yapıları konseptlerini uygulamak için bir oyun geliştirilmesini içerir. Oyunda, oyuncu bir labirentte belirlenen hedefe ulaşmaya çalışırken düşman karakterlerden kaçmak zorundadır. Proje, aşağıdaki ana bileşenlerden oluşmaktadır:

- İyi ve kötü karakterlerin sınıflarının oluşturulması
- En kısa yol algoritmasının uygulanması
- Grafik arayüzün tasarlanması
- Oyun mekaniğinin ve kurallarının belirlenmesi

1- Karakterler ve Sınıf Yapısı

Lokasyon Sınıfı

Her karakterin konum bilgisi (x, y koordinatları) ile tutulacaktır.

```python
class Lokasyon:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    def setX(self, yeniX):
        self.x = yeniX

    def setY(self, yeniY):
        self.y = yeniY
```

Karakter Sınıfı

Tüm karakterlerin ortak özelliklerini içeren temel sınıftır.

```python
class Karakter:
    def __init__(self, ad, tur, konum):
        self.ad = ad
        self.tur = tur
        self.konum = konum

    def getAd(self):
        return self.ad

    def setAd(self, ad):
        self.ad = ad

    def getTur(self):
        return self.tur

    def setTur(self, tur):
        self.tur = tur

    def getKonum(self):
        return self.konum

    def setKonum(self, konum):
        self.konum = konum
```

Kötü Karakterler

Kötü karakterler, temel Karakter sınıfından türetilerek özelleştirilecek.

DarthVader Sınıfı

```python
from karakter import Karakter

class DarthVader(Karakter):
    def __init__(self, ad, tur, konum):
        super().__init__(ad, tur, konum)

    def enKisaYol(self, harita, hedef):
        # Tüm duvarları kaldırarak en kısa yolu hesaplama algoritması
        duvarsiz_harita = [[0 if hucre == 1 else hucre for hucre in satir] for satir in harita]
        return super().enKisaYol(duvarsiz_harita, hedef)
```

KyloRen Sınıfı

```python
from karakter import Karakter

class KyloRen(Karakter):
    def __init__(self, ad, tur, konum):
        super().__init__(ad, tur, konum)

    def enKisaYol(self, harita, hedef):
        # Tek harekette iki birim birden gidecek şekilde en kısa yolu hesaplama algoritması
        satir_sayisi = len(harita)
        sutun_sayisi = len(harita[0])
        mesafe = [[float('inf')] * sutun_sayisi for _ in range(satir_sayisi)]
        mesafe[self.konum.getX()][self.konum.getY()] = 0
        pq = [(0, self.konum.getX(), self.konum.getY())]
        yonler = [(-2, 0), (2, 0), (0, -2), (0, 2)]

        while pq:
            mevcut_mesafe, x, y = heapq.heappop(pq)
            if (x, y) == (hedef.getX(), hedef.getY()):
                break
            for dx, dy in yonler:
                nx, ny = x + dx, y + dy
                if 0 <= nx < satir_sayisi and 0 <= ny < sutun_sayisi and harita[nx][ny] == 0:
                    yeni_mesafe = mevcut_mesafe + 1
                    if yeni_mesafe < mesafe[nx][ny]:
                        mesafe[nx][ny] = yeni_mesafe
                        heapq.heappush(pq, (yeni_mesafe, nx, ny))

        return mesafe[hedef.getX()][hedef.getY()]
```

Stormtrooper Sınıfı

```python
from karakter import Karakter

class Stormtrooper(Karakter):
    def __init__(self, ad, tur, konum):
        super().__init__(ad, tur, konum)

    def enKisaYol(self, harita, hedef):
        return super().enKisaYol(harita, hedef)
```

İyi Karakterler

İyi karakterler, temel Karakter sınıfından türetilerek özelleştirilecek.

MasterYoda Sınıfı

```python
from karakter import Karakter

class MasterYoda(Karakter):
    def __init__(self, ad, tur, konum, can):
        super().__init__(ad, tur, konum)
        self.can = can

    def getCan(self):
        return self.can

    def setCan(self, can):
        self.can = can
```

LukeSkywalker Sınıfı

```python
from karakter import Karakter

class LukeSkywalker(Karakter):
    def __init__(self, ad, tur, konum, can):
        super().__init__(ad, tur, konum)
        self.can = can

    def getCan(self):
        return self.can

    def setCan(self, can):
        self.can = can
```

En Kısa Yol Algoritması

Projede kullanılan en kısa yol algoritmaları Dijkstra, A* veya BFS algoritmalarıdır. Her bir kötü karakter, oyuncunun her hamlesine bağlı olarak dinamik en kısa yolu hesaplayarak hareket eder.

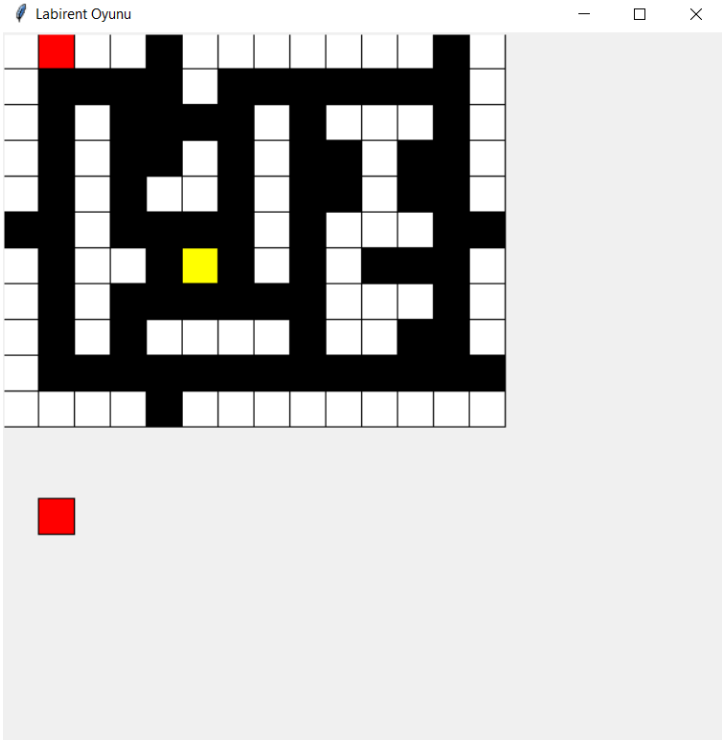
```
def enKisaYol(self, harita, hedef):
    # Dijkstra algoritması
    satir_sayisi = len(harita)
    sutun_sayisi = len(harita[0])
    mesafe = [[float('inf')] * sutun_sayisi for _ in range(satir_sayisi)]
    mesafe[self.konum.getX()][self.konum.getY()] = 0
    pq = [(0, self.konum.getX(), self.konum.getY())]
    yonler = [(-1, 0), (1, 0), (0, -1), (0, 1)]

    while pq:
        mevcut_mesafe, x, y = heapq.heappop(pq)
        if (x, y) == (hedef.getX(), hedef.getY()):
            break
        for dx, dy in yonler:
            nx, ny = x + dx, y + dy
            if 0 <= nx < satir_sayisi and 0 <= ny < sutun_sayisi and harita[nx][ny] == 0:
                yeni_mesafe = mevcut_mesafe + 1
                if yeni_mesafe < mesafe[nx][ny]:
                    mesafe[nx][ny] = yeni_mesafe
                    heapq.heappush(pq, (yeni_mesafe, nx, ny))

    return mesafe[hedef.getX()][hedef.getY()]
```

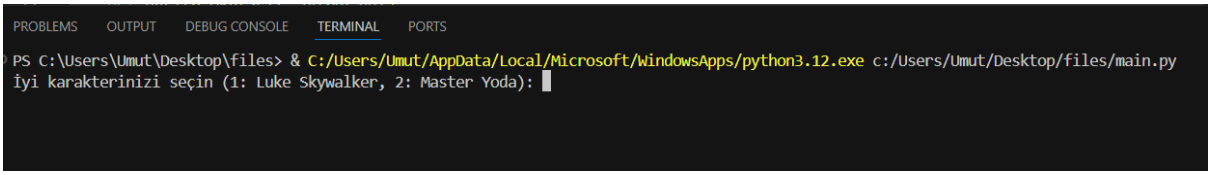
2- Arayüz Tasarımı

Oyunun arayüzü, grafiksel olarak kullanıcıya sunulacak ve çeşitli grafik kütüphaneleri kullanılabilir. Harita, karakterlerin konumları ve hareketleri ekranda gösterilecektir.



3- Oyun Mekaniği

Oyun, kullanıcının iyi karakterini seçmesiyle başlar ve klavye ok tuşlarıyla yönlendirilir. Kötü karakterler, dinamik en kısa yol algoritmalarını kullanarak oyuncuyu kovalar. Oyuncunun canı tükenmeden hedefe ulaşması gerekmektedir.



5- Çıkarılan Sonuçlar

Bu proje, nesne yönelimli programlama ve veri yapıları konseptlerini uygulayarak bir oyun geliştirilmesini içermektedir. Proje kapsamında tüm sınıflar doğru bir şekilde belirlenmiş, en kısa yol algoritmaları uygulanmış, grafiksel arayüz tasarlanmış ve oyun mekaniği belirlenmiştir.