Star Wars Labirent Oyunu

Yaşar Can Çilingir Gülfem Kavak

170202017 170202054

Kocaeli Üniversitesi Bilgisayar Mühendisliği

[yasarcanccd@gmail.com](mailto:yasarcanccd@gmail.com), [gulfemkavak@gmail.com](mailto:gulfemkavak@gmail.com)

***Özetçe—***Star wars labirent oyunu, kullanıcının seçtiği Luke Skywalker veya Master Yoda iyi karakterlerinden birisini hareket ettirerek labirentin çıkışına, kötü karakterlerden birisine yakalanmadan gitmeyi amaçlar. Master Yoda’nın 6 kere, Luke Skywalker’ın 3 kere yakalanma hakkı vardır. Kötü karakterlerden Darth Vader labirent duvarlarından geçebilir, Kylo Ren her hamlede 2 adım birden ilerleyebilir, Stormtrooper’ın ise ek bir özelliği yoktur. Kötü karakterler, iyi karakterin her hamlesinde iyi karaktere ulaşabileceği en kısa yolu hesaplar ve ona göre hareket eder.

1. **GİRİŞ**

Star Wars labirent oyunu, karakter sınıfından hiyerarşik bir düzenle oluşturulan karakterlerden oluşur. Master Yoda ve Luke Skywalker iyi karakterlerdir, Darth Vader, Stormtrooper, Kylo Ren kötü karakterlerdir. Master Yoda’nın 6 kez yakalanma hakkı vardır, Luke Skywalker’in ise 3 kez yakalanma hakkı vardır.

Oyun başladıktan sonra iyi karakter her adım attığında kötü karakterler oluşturulan en kısa yol algoritmasına göre iyi karaktere ulaşabileceği en kısa yolu hesaplayıp ekranda gösterir ve ona göre hamlesini yapar. En kısa yol bulma algoritması oluşturulurken BFS (breadth first search) algoritmasından esinlenilmiştir.

1. **Temel Bilgiler**

Projede JAVA programlama dili kullanılmıştır.

Proje geliştirme ortamı olarak tümleşik geliştirme ortamı Netbeans IDE kullanılmıştır.

Grafik ile oluşturulan ızgaranın sol üst köşesi 0,0 konumu olarak kabul edilmiştir. X ekseni sağa doğru gittikçe, Y ekseni aşağı doğru gittikçe artmaktadır.

Kötü karakterlerin ve haritanın bilgisi text dosyasından okunmaktadır. Haritada 1 olan yerlerde karakterler hareket edebilmektedir, 0 olan yerlerde hareket edememektedir.

1. **TASARIM**

Proje aşağıdaki alt başlıklar altında geliştirilmiştir.

* 1. **Hiyerarşik Yapı**

Star Wars Labirent projesinde tüm karakterler Karakter sınıfından türetilmiştir. Karakterlerin her birinin ayrı ayrı lokasyonlarını tutan lokasyon sınıfı da vardır.

Node sınıfı ise en kısa yol bulma algoritması için x, y değerlerini tutan sınıftır.

* 1. **Yöntemler**

Projede kullanılan methodlar aşağıda açıklanmıştır.

BFS() : En kısa yolu bulma algoritmasıdır, iyi karakterin ve kötü karakterin başlangıç konumları gönderilir, labirent üzerinden aralarındaki en kısa mesafeyi hesaplar.

BFSvader() : Darth Vader için özel en kısa yol bulma algoritmasıdır çünkü Darth Vader duvarlardan geçebildiği için diğer karakterlerden en kısa yolu labirent üzerinde farklıdır.

AdimAt() : Hesaplanan en kısa yola göre kötü karakterlerin adım atması işlemidir. Kylo Ren hesaplanan adımın 2. Sine giderken diğerleri ilk adıma gider.

KeyPressed(): Kullanıcının klayvede bastığı ok tuşuna göre iyi karakterin hareket etmesini sağlar ve kötü karakterlerin en kısa yola göre adım atmasını tetikler.

MatrisOlustur(): Karakterlerin matrisin dahil olduğu harita üzerinde sadece 1’ler üzerinde hareket edebilmesini kontrol edebilmek için çift boyutlu dizi oluşturma işlemidir.

paint(): Haritanın, tüm karakterlerin, en ksa yolun ve can sayacının, projedeki grafiğe dair her şeyin çizim işleminin yapıldığı fonksiyondur.

İyiKarakterOlustur(): Kullanıcının seçtiği iyi karakterin hangisinin olduğunun kontrolü ve ona göre başlangıç konumunun karaktere atanması işleminin yapıldığ fonksiyondur.

KotuKarakterOlustur(): Textten okunan kötü karakterlerin hangilerinn olduğunu belirleyen ve ona göre kötü karakterlerin konumlarını atanması işleminin yapıldığı yerdir.

1. **KARŞILAŞILAN SORUN VE ÇÖZÜMLERİ**

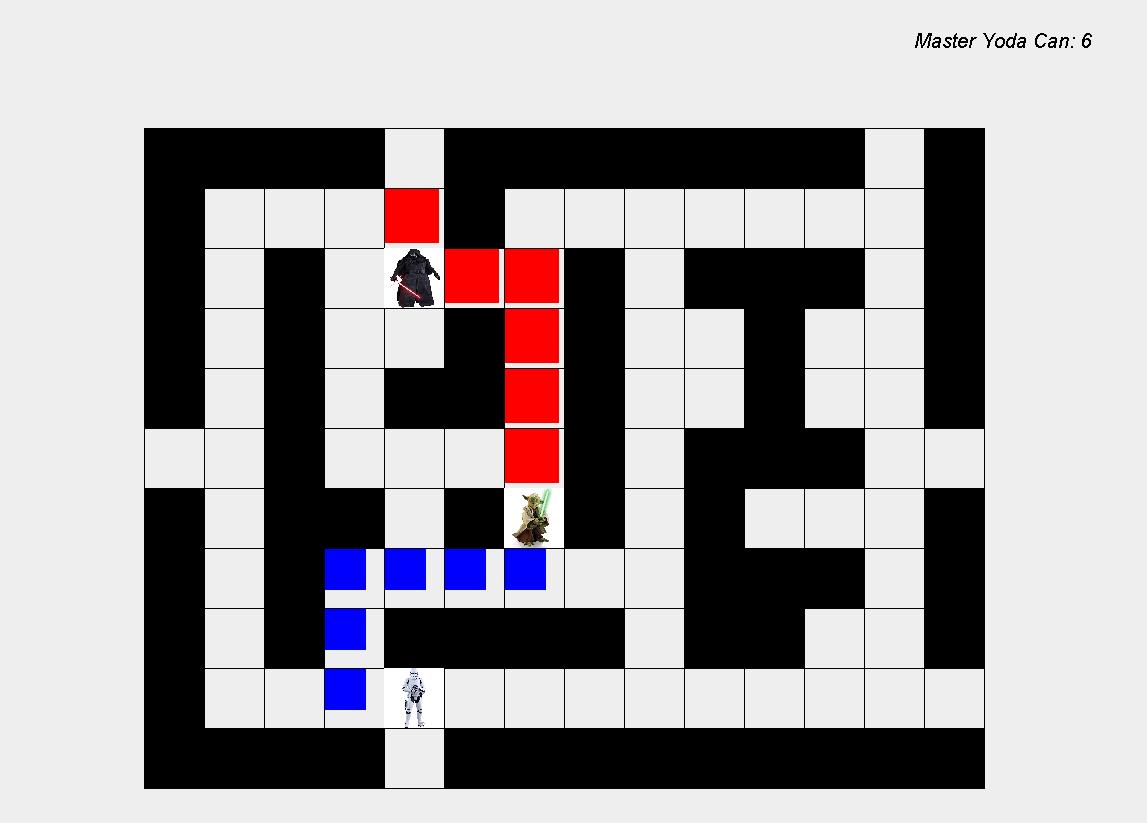
**Labirent içinde en kısa yol bulma algoritması**

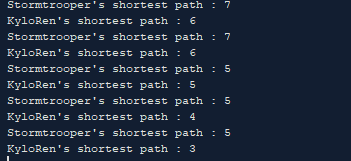
Labirent içinde en kısa mesafeyi ve yolu bulma algoritması için çeşitli hazır algoritmalar incelenmiştir. Breadth First Seartch algoritmasından esinlenerek algoritma oluşturulmuştur.

1. **KAZANIMLAR**
2. Inheritance yapısının daha iyi kullanılması
3. Java grafik kütüphanesinin klayve input-outputları ile uyumlu olarak çalışması
4. Kısa yol algoritmasını etkin bir şekilde oluşturabilmek ve kullanabilmek.
5. **PSEUDO KOD**
6. Başla
7. Oluşturulacak iyi karakter çeşidini kullanıcıdan al.
8. Text dosyasını oku
9. Text dosyasındaki kötü karakterleri belirle
10. Text dosyasındaki kötü karakterlerin oluşturulacağı kapıları belirle
11. Text dosyasındaki 1 ve 0 ları iki boyutlu diziye aktar.
12. Aktarılan iki boyutlu diziye göre labirenti oluştur.
13. Oluşturulan labirentte 0 olan yerleri siyah boya.
14. İyi karakter için kullanıcıdan ok işaretleri ile input al.
15. Alınan değeri geçici temp koordinatlarına ata.
16. Temp Koordinatlarındaki matrisin değeri 1 mi 0 mı kontrol et.
17. 1 ise iyi karakterin lokasyon koordinatlarını temp geçici koordinatlarına eşitle.
18. 0 ise eşitleme, aynı konumda kal.
19. Kötü karakterlerin konumlarını ve iyi karakterin konumunu en kısa yol bulma algoritmasına gönder.
20. En kısa yol bulma algoritmasında kötü karakter konumunu adım atabileceği her yere götür.
21. Her adımında bir önceki geldiği yolu kopyala.
22. Kötü karakter konumu ile iyi karakter konumu eşit mi değil mi kontrol et.
23. Diziden çıktı mı kontrol et.
24. 1 mi 0 mı kontrol et.
25. Daha önce ziyaret edildi mi kontrol et.
26. Adım atılabildiyse visited matrisinde ilgili konumu true olarak güncelle.
27. İyi karakter konumuna eşit ise kopyaladığı kısa yol listesini paint fonksiyonu ile çizdir.
28. Kötü karakterin konumunu kısa yol listesinin ilk konumuna eşitle.
29. Eğer Kylo Ren ise 2. Konuma eşitle.
30. Eğer Darth Vader ise kısa yol 0’lardan da geçebilir.
31. Eğer kötü karakter konumu ile iyi karakter konumu aynı mı kontrol et
32. Eğer aynı ise iyi karakter canını azalt.
33. Tüm karakterlerin konumlarını en baştaki haline getir.
34. Eğer Master Yoda ise 6 kere yanabilir.
35. Eğer Luke Skywalker ise 3 kere yanabilir.
36. Eğer kullanıcı ödüle ulaşırsa oyun biter.
37. **SPACE VE TİME ANALİZİ**

O(n), O(V^2)

1. **BAZI EKRAN GÖRÜNTÜLERİ**

****

****

****

1. **KAYNAKÇA**
2. JAVA, Swing kütüphanesi kullanımı

<https://www.youtube.com/watch?v=hR-kzJwQhIc&list=PLHfYetw_BGF-Gm_MsqKApw5nHPuHsytr3&index=35>

1. Labirent içinde örnek kısa yol algoritması

<https://www.techiedelight.com/lee-algorithm-shortest-path-in-a-maze/>

1. Labirent içinde örnek kısa yol algoritması

https://www.lavivienpost.com/shortest-path-between-cells-in-matrix-code/