ALTIN TOPLAMA OYUNU

Furkan Aydoğan

Bilgisayar Mühendisliği, Kocaeli Üniversitesi

180202085

Sefa Berke Kara

Bilgisayar Mühendisliği, Kocaeli Üniversitesi

180202086

Giriş

Altın Toplama Oyunu; mxn boyutlu bir dikdörtgen tahta üzerinde farklı özelliklere sahip olan oyuncuların altın toplama yarışına dayanır. Bu projede temel 2 amaç üzerine odaklanılmıştır. Bunlar farklı kısıtlara sahip arama algoritmalarının birbirlerine karşı etkinliklerini gözlemlemek ve dinamik özelliklere sahip bir program geliştirmektir.

Özet

Altın Toplama Oyunu iki ara yüzden oluşmaktadır. Bunlar oyunun oynandığı oyun ara yüzü ve de oyuna ait parametrelerin belirlendiği ayarlar sayfası ara yüzüdür. Ayarlar sayfasındaki parametreler varsayılan olarak hali hazırda atanmıştır. Kullanıcı dilediği takdirde parametreleri kendi istediği doğrultu da değiştirebilir. Ana oyun ara yüzünde “Altınları Ata” butonuna tıklayarak rastgele altınlar atanır. “Oyuna başla” butonuna tıklanarak da oyuna başlanmış olur. Anlık olarak A-B-C-D oyuncularının altın değerleri ara yüzden izlenebilmektedir. Oyun sayfasında hiçbir altın kalmayana kadar oyun devam eder. Altınlar bitiğinde de boyun otomatik olarak bitmiş olur.

Temel Bilgiler

Windows 10 işletim sistemine sahip bilgisayar ile geliştirme yapılmıştır. Program Java programlama dilinde geliştirilmiş olup, tümleşik geliştirme ortamı olarak “Netbeans” kullanılmıştır

Tasarım

Altın Toplama Oyunu programlanma aşamaları altta belirtilen başlıklar altında açıklanmıştır.

Algoritma

Altın Toplama oyunu ilk olarak “ Oyun-ozet.txt” adında bir dosya oluşturur. Dosyaya oyuncuların yazılması için A,B,C ve D oyuncuları adına 4 tane ArrayList oluşturur. Altin_Oyunu sınıfından nesne oluşturulur ve oyun başlar. İlk oyuncu olan A oyuncusu için a değişkeni true yapılır. Ardından clearmap() fonksiyonu çalıştırılarak oyun tahtası Map matrisine atanır. Bu işlemlerin ardından initialize() fonksiyonuna gidilerek oyun içi görsel ara yüz ve Map sınıfından oluşturulmuş “canvas” değişkeni ile oyun tahtası bu fonksiyon içinde oluşturulur. Her bir buton için ActionListener metotları override edilerek kullanıcının istediği değerler atanır. Altın Toplama Oyunu ilk başlatıldığında kullanıcın karşısına ara yüzde 3 tane buton sunulur. Bunlar “Oyuna Başla”, ”Altınları Ata” ve ”Ayarlar” butonlarıdır. Kullanıcın “Ayarlar ” butonuna tıklamasıyla birlikte yine aynı ad ile oluşturulmuş bir pencere sunulur. Bu ayarlar sayfasında kullanıcı oyun içi varsayılan değerlerin hepsini kendi isteği doğrultusunda değiştirebilir. “Altınları ata” butonuna tıklaması ile oyun tahtasında random olarak gizli altın ve gizli olmayan altınlar otomatik olarak bastırılır ve random olarak altın değerleri atanır. ”Oyuna Başla ” butonuna tıklanması ile de oyun başlar. “Oyuna başla “ butonuna basıldığında “devamMı” boolen değişkeni true yapar ve ardından Oyuna_basla() fonksiyonuna geçer.”devamMı” true olduğundan bu fonksiyon için Algoritma sınıfındaki Dijkstra() fonksiyonu çağrılır. A oyuncusu için kasasından her hedef için belirlenen maliyeti düşürülür. A oyuncusunun koordinatlarını dosyaya yazdırmak için A_ArrayList adında bir arraylist oluşturup A oyuncusunun x,y koordinat değerlerini bu arrayliste ataması yapılır. Bu işlemin ardından A oyuncusunun bulunduğu kordinatı priority adında oluşturulan ArrayList’ine atanır. A oyuncusunun o anki katman değerini +1 ekleyerek “katman” adındaki değişkene atanır.Ardından “Priority” de tutulan adres ve katman değerleri parametre olarak komsu_kesfet() fonksiyonuna atanır. A oyuncusunun bulunduğu konumunun üst, alt ,sağ ve soldaki komşularına bakılır. Eğer uygun ise komşuları keşfedildi ArrayListine eklenir ve .kesfet fonksiyonuna gönderilir. kesfet() fonksiyonu 4 tane parametre alır. Bunlar o anki konumunun komşusu kendi konumunun x ve y değerleri ve komşusunun katmanıdır. Komşu olan yere kendi A oyuncusun koordinatlarını ve katmanını ekler ve bu şekilde altın bulunan kadar döngüde dönmeye devam eder ve bağlı liste oluşturur. Altın bulunduğu zaman altının koordinatlar “a_X ve a_Y” değişkenlerinde tutulur. Hemen ardından dasonra da geri_takip() fonksiyonunu çalıştırılır. Katman sayısı length değişkenine eşitlenir. Ardından katman sayısı kadar next tipinde ileri adında bir dizi oluşturulur. Bu oluşturulan diziye bulunan altının x ve y değerleri eklenir. While döngüsünde hops değeri birden büyük olduğu sürece önceden bağlı liste şeklinde oluşturulmuş komşular ileri dizine atanır. Bu dizinin oluşturulma amacı A oyuncusunun hedefe yönelmesini sağlamaktır.Bu işlemlerin ardından devamMi değişkeni false yapılarak A oyuncusu için Dijikstra algoritması sonlandırılır. a_kontrol değişkeni ilk başta 0 olarak atandığı için if

koşuluna girmeyerek else koşuluna girer. Bu koşul altında renk() fonksiyonu A oyuncusu için çalışıtırlır. A oyuncusu için oyuncusunun kasasından adım maliyeti kadar altın düşürülür. Ardından bulunduğu konum renklendirilerek for döngüsü içine girilir. Bu for döngüsü içinde A_altin_uzaklık sayısına kadar döndürülür. Eğer oyuncu hedefine giderken gizli altın üzerinden geçerse tüm oyuncular için görünür kılınır. Eğer oyuncu oyun tahtasında altın üzerine geldiyse burada oyuncunun hedeflediği altın üzerinde bulunduğu altına eşit mi koşulu kontrol edilir. Eğer üzerinde bulunduğu altın hedeflediği altına eşit değilse o altını kendi kasasına eklemez. Şayet hedeflediği altın ise if koşulunun içine girerek kendi kasasına ekleyerek bulunduğu konumu renklendirip belirtir. Eğer gizli altın veya altın koşulu sağlanmaz ise else koşuluna girerek oyun tahtası lacivert renginde renklendirilerek adım atması görsel olarak gösterilmiş olur. Burada oyunun başında belirtilmiş olan adım sayısı kadar belirtilip eğer hedefine ulaşmadıysa orada kendini durdurur. En son kaldığı adımda a_en_son_kalinan_adim değişkeninde tutulur.Bu işlemler yapıldıktan sonra renk() fonksiyonu beyazYap() fonksiyonunu çağırır. BeyazYap() fonksiyonunda a koşulunun sağlandığı if çalıştırılır.Adım animasyonu vermek amacıyla renklendirilen yerler belli koşullar çerçevesinde tekrardan oyun tahtası rengine(beyaz) çevrilerek update() fonksiyonu çağrılır. Keşfedilen komşuları “kesfedildi” ArrayList’ine atanır.Bakılan komşuyu priority Arraylistinden kaldırılır. Kalan komşuları da kontrol edilmek üzere priority ArrayList’ine atanır.Hemen ardından Update() ve delay() fonksiyonları tekrardan çağrılır. Dijikstra() fonksiyonu bitmesi ile de değer() fonksiyonu çağrılır. B oyuncusunun değeri true olarak atayıp oyundaMi() fonksiyonu çağrılır. Bu fonksiyon altında kontrol() fonksiyonu çağrılarak oyun tahtasında altın olup olmadığı kontrol edilir. B oyuncusunun gerekli koşulları if ve else koşulları altında kontrol edilir. Eğer oyuncunun hamle maliyeti ve hedef maliyeti B oyuncusunun kasasındaki paraya yetiyorsa b_oyun değişkeni true olarak kalmaya devam ettirilir. oyundaMi() fonksiyonundan eğer oyuncunun devam edeceği anlaşılırsa B oyuncusu tekrardan çağrıldığı yere yani değer)() fonksiyonuna geri gelir. Eğer B oyuncusu daha önce hedef belirlememiş ise hesapla() fonksiyonuna giderek en karlı altını hesaplar. B oyuncusu oyun tahtasındaki tüm altınları bir matriste tutulur ve bu altınların kendisine olan uzaklığını bir path adındaki bir dizide tutar En son olarak da Maliyet fonksiyonu çağrılır. Hesapla() fonksiyonunda tutulan bu altın koordinatları maliyet matrisinin içine yazılarak altının değerleri alınır. Yine hesapla() fonksiyonunda tutulan altına olan uzaklıklar belli işlemler ile hesaplama yapılarak kar_hesabı dizisine atanır. Bu işlemin ardından en karlı altın hesap edilerek next sınıfında oluşturulmuş yol dizisine oyuncunun adım atması için gerekli yollar tutulur ve oyuncunun kasasından hedef maliyeti kadar düşürülür. Ardından renk() fonksiyonunu çağırıp renk() fonksiyonunda gidilen yolu renklendirir. Her hamlenin maliyeti kasadan düşürülür. Eğer gizili altın üstünden geçerse normal altına çevrilir. Eğer altın üzerinden geçerse kendi hedeflediği altın olup olmadığını kontrol eder. Şayet kendi hedeflediği altın ise altının maliyetini B oyuncusunun kasasına ekler. Bu işlemler ardından beyazYap() fonksiyonu oyun tahtasında adım atmak için renklendirilen bölgeleri tekrardan oyun tahtası rengine döndürür ve kaldığı yeri yeni konumu olarak atar. Eğer B oyuncusu hedefini belirlediyse else koşuluna girerek adım atmadan önce altının başka oyuncu tarafından alınıp alınmadığını kontrol eder. Eğer alınmamış ise renk() fonksiyonunu çağırarak adım atmaya devam eder, alınmış ise hesapla() fonksiyonu tekrardan yine çağrılarak en karlı altın belirlenir. Bu işlemlerin ardından sıra C oyuncusuna geçer. oyundaMi() fonksiyonun da C oyuncusunun oyuna devam edebilmesi için gerekli koşullar kontrol edilir.Eğer kasadaki altın miktarı oyuna devam etmeye yetiyorsa bu sefer de oyun tahtasında altın olup olmadığını kontrol() fonksiyonu ile kontrol eder.C oyuncusunun hedefi yoksa veya hedefine vardıysa C_hesapla() fonksiyonunu çalıştırır.Oyun tahtasındaki oyuncular ve altınları C oyuncusu için oluşturulan sanal bir map matrise kopyalanır.Gizli altın adet sayısını sayıp koordinatlarını ve gizli altınlara olan uzunluğunu Manh_uzakligi adındaki dizide tutar.En yakından uzağa bütün gizli altınları ve koordinatları sıralanıp enn yakın 2 gizli altın görünür yapılır.Hemen ardından yeni oyun tahtasını C_Sanal_Map adındaki matrise kopyalanır. Güncellenen altın ve gizli altın sayılarını tekrardan sayarak C oyuncusu oyun tahtasındaki tüm altınları tarayarak kendisine olan uzaklıklarını dizi de ve kordinatlarını da bir ArrayList de tutar.Tutulan koordinatlar C oyuncusunun en karlı altını bulabilmesi için Maliyet_C() fonksiyonuna gönderilir.C oyuncusunun konumu ve oyun tahtasındaki altınların değerlerini ve konumunu alarak kar hesabı yapıp kar_hesabı dizisine atanır. Ardından en karlı olan yolu hesapla next tipindeki yol_C dizisine atanır. Ardından renk() fonksiyonunu çağırıp renk() fonksiyonunda gidilen yolu renklendirir. Her hamlenin maliyeti kasadan düşürülür. Eğer gizili altın üstünden geçerse normal altına çevrilir. Eğer altın üzerinden geçerse kendi hedeflediği altın olup olmadığını kontrol eder. Şayet kendi hedeflediği altın ise altının maliyetini C oyuncusunun kasasına ekler. Bu işlemler ardından beyazYap() fonksiyonu oyun tahtasında adım atmak için renklendirilen bölgeleri tekrardan oyun tahtası rengine döndürür ve kaldığı yeri yeni konumu olarak atar. C oyuncusu hedefini daha önceden belirlediyse ve sıra tekrardan C oyuncusuna geldiyse else koşuluna girerek daha önceden hedeflediği altının orada olup olmadığını kontrol eder. Eğer altın var ise varsayılan adım kadar adım atmaya devam eder. Eğer altın yok ise de tekrardan C_hesapla() fonksiyonunu çağırarak 2 tane gizli altın açılır ve en karlı altını tespit edip adım atmaya devam eder. Bu işlemlerin ardından sıra D oyuncusuna geçer. D oyuncusu için oyundaMi() fonksiyonu çağrılır. Hamle maliyeti ve değer maliyeti kasadaki altın sayısına yetiyorsa D_oyun true olarak kalmaya devam eder. Ardından kontrol() fonksiyonu çağrılarak D oyuncusunun koordinatları ile diğer oyuncuların hedeflediği altınlar arsındaki uzaklığı hesaplayıp ve uzunluk dizisini atanır. Oyun tahtasındaki altın sayısını say. Eğer altın sayısı 1 ise ilk if’ e gir ve oyuncuların o altına olan uzaklıkları hesaplanır. Diğer oyunculardan daha önce D oyuncusunun kalan son altına ulaşıp ulaşamayacağı

kontrol edilir.Eğer D oyuncusunun kalan adım sayısı 4 den daha küçük ise direkt hedeflediği altına gidilir. Eğer altın sayısı 1'den farklı ise else kısmına gidilir. Ardından A, B, ve C oyuncularının hedeflediği altınlar arasındaki uzaklıkları alınır.H er oyuncu için o oyuncunun hedefine ondan önce gidip gidemeyeceğine bakılır. Eğer gidebiliyorsa değişkene 1 değeri atanır, gidilemiyorsa kendinden önceki oyuncuların hedeflediği altınlar dışındaki altınların hepsini D_Map'e atayarak D_Map de altın var mı yok mu diye kontrol edilir. Eğer yok ise D oyuncusunu oyundan atar(oyun tahtasında durdur). Değer() fonksiyonuna geri gelinir. Ardından d değişkeninin true olduğu değer çalıştırılır. Eğer D oyuncusu daha önceden hedef belirlemediyse hesapla_D() fonksiyonuna gidilir. Bu fonksiyonda node tipinde D_sanal_Map adında bir matris oluşturulur. Hemen ardından kontrol(9 fonksiyonunda hangi gerekli şart sağlandıysa önceden 1 değeri atanan değişkenler bu fonksiyonla kontrol edilir. Eğer ilk if 'e girerse A oyuncusundan önce A'nın hedefine ulaşılabilir demektedir. Burada D_sanal_Map ' e gerekli renklendirmeler yapılarak A'nın hedefinin yolu D oyuncusu için tutulur. Eğer else if ' e girilirse bu B oyuncusundan önce hedefine gidiyor demektir burada D_sanal_Map de gerekli renklendirmeler yapılarak B 'nin hedefinin yolu D oyuncusu için tutulur. Diğer else if koşuluna girerse de bu işlemler C oyuncusu için yapılır. Bu koşullardan hiçbirine girmez ise de else koşuluna girerek bu oyuncuların hedeflerine onlardan önce ulaşamayacağı için bu oyuncuların hedeflediği altınların dışındaki altınlar D_sanal_Map matrisine atanır. Atama işleminden sonra oyun tahtasındaki altınlar sayılır For döngüsünde altın sayısı kadar döndürülerek D oyuncusunun altınlara olan yolları,uzaklıkları değişkenler de tutulur. Bu işlemin ardından da Maliyet_D() fonksiyonuna gidilir. Hesapla_D() fonksiyonunda tutulan altınkordinatları maliyet matrisinde yerine yazılarak tüm altınlar için altınların değerleri alınır. Bu alınma işlemine paralel olarak da altınların yolları alınarak kar hesabı yapılır .Yapılan kar hesabı kar_hesabı dizisine atanır. Burada gidebileceği en karlı yol bulunur. Bulunan yol adım atması için bir dizide tutulur. Bu işlemlerin ardından renk() fonksiyonu çağrılır. D oyuncusunun yol dizisi adım atması için for döngüsünde döndürülür. Varsayılan adım kadar ilerlediğinde eğer altına ulaşmadıysa bulunduğu bölgede kalır ve bir daha sıra geldiğinde kaldığı yerden devam eder. Ardından her hamlenin maliyetini kasadan düş.Eğer gizili altın üstünden geçerse normal altına çevir. Eğer altın üzerinden geçerse kendi hedeflediği altın olup olmadığını kontrol et. Eğer kendi hedeflediği altın ise altının maliyetini D oyuncusunun kasasına ekler. Bu işlemler tamamlandıktan sonra beyazYap() fonksiyonu çağırarak renklendirilen kısımlar tekrardan beyaz hale getirilir. Eğer hedefine ulaştıysa diğer oyuncuya sırasını vermeden hedef belirletilir. Bu işlemler tamamlandıktan sonra beyazYap() fonksiyonu çağırarak renklendirilen kısımlar tekrardan beyaz hale getirilir. Eğer hedefine ulaştıysa diğer oyuncuya sırasını vermeden hedef belirlenir. D oyuncusu için if koşulu sağlanmaz ise else koşuluna girer. Eğer D oyuncusunun hedefi başka bir oyuncu tarafından alındıysa yeni bir hedef belirlenir. Hedefi belirli ise altını başka oyuncu tarafından alınmadı ise renk() fonksiyonu çağrılarak adım atmaya devam edilir.Bu işlemleri oyuncuların kasasındaki altınlar bitene kadar ya da oyun tahtasındaki altınlar bitene kadar devam eder. Son olarak da oyuna ait özet tablo ekrana çıktı olarak basılır. "oyun-ozet.txt" dosyasında ise oyuncuların oyun başladığı andan itibaren attığı adımlar bastırılır.

Sözde Kod

1- "Oyun-Ozet.txt" adından bir dosya oluştur.
2- PathFinding() class'ından bir nesne oluştur. Constructor içine gir.
3- A değişkenine true değerini ata, ClearMap() fonksiyonunu çalıştır
4- ClearMap() fonksiyonunda oyun tahtasını oluştur ve Node tipindeki Map adındaki bir matrise ata.
5- nitialize() fonksiyonunu çalıştır.
6- Map sınıfından canvas adında bir değişken oluşturular oyun tahtasını çizdir.
7- Oyun penceresini ve kullanıcı ara yüzünü oluştur.
8- startSearc() fonksiyonunu çalıştır.
9- A oyuncusu için Dijkstra algoritmasını çalıştır.
10- A oyuncusu için kasasından her hedef için belirlenen maliyeti düşür.
11- A oyuncusunun koordinatlarını dosyaya yazdırmak için A_ArrayList adında bir arraylist oluştur ve A oyuncusunun x,y koordinat değerlerini bu arrayliste ata.
12- A oyuncusunun bulunduğu kordinatı priority arraylistine ata
13- A oyuncusunun o anki katman değerini +1 ekleyerek katman değişkenine ata.
14- Priority de tutulan adres ve katman değerleri parametre olarak komsu_kesfet() fonksiyonuna atanır.
15- A oyuncusunun bulunduğu konumunun üst,alt ,sağ ve soldaki komşularına bak
16- Eğer uygun ise komşuları keşfedildi ArrayListine ekle.
17- Kesfet fonksiyonuna git
18- Keşfet fonksiyonunda parametre olarak uygun komşunun adresini,kendi koordinatlarını ve katmanını parametre olarak gönder.
19- Komşu olan yere kendi A oyuncusun koordinatlarını ve katmanını ekle.
20- Bu şekilde altın bulunan kadar döngüde dönmeye devam et ve bağlı liste oluştur.
21- Altın bulunduktan sonra da geri_takip() fonksiyonunu çalıştır.
22- Altın yolunun koordinatlarını next tipinde oluşturulan ileri dizisine ata.
23- Renk() fonksiyonunu çağırarak haritanın renklendirilmesini sağla
24- beyazYap() fonksiyonu çağırarak renklendirilen kısımlar tekrardan beyaz hale getirilir. Eğer hedefine ulaştıysan diğer oyuncuya sıranı vermeden hedef belirle.
25- Keşfedilen komşuları "kesfedildi" ArrayList'ine ata.
26- Bakılan komşuyu priority Arraylistinden kaldır.

27- Kalan komşuları da kontorl edilmek üzere priority ArrayList'ine atanır.
28- Update() ve delay() fonksiyonlarını çağır
29- Dijikstra() fonksiyonu bittikten sonra değer() fonksiyonunu çağır.
30- B oyuncusunun değerini true yap.oyundaMi() fonksiyonunu çağır.
31- oyundaMi() fonksiyonun da B oyuncusunun oyuna devam edebilmesi için gerekli koşullar kontrol edilir.
32- Eğer kasadaki altın miktarı oyuna devam etmeye yetiyorsa bu sefer de oyun tahtasında altın olup olmadığını kontrol() fonksiyonu ile kontrol et.
33- B oyuncusunun hedefi yoksa veya hedefine vardıysa hesapla() fonksiyonunu çalıştır
34- B oyuncusu oyun tahtasındaki tüm altınları bir matriste tut ve bu altınların kendisine olan uzaklığını bir path adındaki bir dizide tut En son olarak da Maliyet fonksiyonunu çağır.
35- Ardından renk() fonksiyonunu çağır.
36- Renk() fonksiyonunda gidilen yolu renklendir.Her hamlenin maliyetini kasadan düş..Eğer gizili altın üstünden geçerse normal altına çevir.Eğer altın üzerinden geçerse kendi hedeflediği altın olup olmadığını kontrol et. Eğer kendi hedeflediği altın ise,altının maliyetini B oyuncusunun kasasına ekle.
37- beyazYap() fonksiyonunu çalıştır ve oyun tahtasındaki daha önceden renkelndirilmiş renkleri beyaz yap.Yeni konumunu başlangıç konumu olarak ata .Ardından oyuncu_sira +1 arttırılarak C oyuncusuna geç.
38- C oyuncusu altın aldıktan sonra eğer kalan altın sayısı bire eşit ise sıra D oyuncusuna geçtiğinde A oyuncusunun hedefsiz kalmaması için yeni hedef belirle.
39- Bu işlemler tamamlandıktan sonra beyazYap() fonksiyonu çağırarak renklendirilen kısımlar tekrardan beyaz hale getirilir. Eğer hedefine ulaştıysa diğer oyuncuya sırasını vermeden hedef belirlet.
40- B oyuncusu için if koşulu sağlanmaz ise else koşuluna gir.Eğer B oyuncusunun hedefi başka bir oyuncu tarafından alındıysa yeni bir hedef belirle.
41- Değer() fonksiyonunu tekrardan çağırarak C oyuncusuna geç.
42- C oyuncusunun değerini true yap.oyundaMi() fonksiyonunu çağır.
43- oyundaMi() fonksiyonun da C oyuncusunun oyuna devam edebilmesi için gerekli koşullar kontrol edilir.
44- Eğer kasadaki altın miktarı oyuna devam etmeye yetiyorsa bu sefer de oyun tahtasında altın olup olmadığını kontrol() fonksiyonu ile kontrol et.
45- C oyuncusunun hedefi yoksa veya hedefine vardıysa C_hesapla() fonksiyonunu çalıştır.
46- Oyun tahtasındaki oyuncular ve altınları C oyuncusu için oluşturulan sanal bir map matrise kopyalar.
47- Gizli altın adet saysını say koordinatlarını ve gizli altınlara olan uzunluğunu Manh_uzakligi adındaki dizide tut.
48- En yakından uzağa bütün gizli altınları ve koordinatları sırala.
49- En yakın 2 gizli altını görünür yap.
50- Yeni oyun tahtasını C_Sanal_Map adındaki matrise kopyala.
51- Güncellenen altın ve gizli altın sayılarını tekrardan say.
52- C oyuncusu oyun tahtasındaki tüm altınları tarayarak kendisine olan uzaklıklarını dizi de ve kordinatlarını da bir ArrayList de tutar.
53- Tutulan koordinatlar C oyuncusunun en karlı altını bulabilmesi için Maliyet_C() fonksiyonuna gönderilir.
54- C oyuncusunun konumu ve oyun tahtasındaki altınların değerlerini ve konumunualarak kar hesabı yapıp kar_hesabı dizisine ata.
55- En karlı olan yolu hesapla next tipindeki yol_C dizisine ata.
56- Ardından renk() fonksiyonunu çağır.
57- Renk() fonksiyonunda gidilen yolu renklendir.Her hamlenin maliyetini kasadan düş.. Eğer gizili altın üstünden geçerse normal altına çevir. Eğer altın üzerinden geçerse kendi hedeflediği altın olup olmadığını kontrol et.
58- C oyuncusu altın aldıktan sonra eğer kalan altın sayısı bire eşit ise sıra D oyuncusuna geçtiğinde A ve B oyuncularının hedefsiz kalmaması için yeni hedef belirle.
59- Eğer kendi hedeflediği altın ise altının maliyetini C oyuncusunun kasasına ekle.
60- Bu işlemler tamamlandıktan sonra beyazYap() fonksiyonu çağırarak renklendirilen kısımlar tekrardan beyaz hale getirilir. Eğer hedefine ulaştıysa diğer oyuncuya sırasını vermeden hedef belirlet.
61- C oyuncusu için if koşulu sağlanmaz ise else koşuluna gir. Eğer C oyuncusunun hedefi başka bir oyuncu tarafından alındıysa yeni bir hedef belirle.
62- Hedefi belirli ise altını başka oyuncu tarafından alınmadı ise renk() fonksiyonu çağrılarak adım atmaya devam et.
63- Değer() fonksiyonunu tekrardan çağırarak D oyuncusuna geç.
64- İlk önce kontrol() fonksiyonunu çalıştır.
65- D oyuncusunun koordinatları ile diğer oyuncuların hedeflediği altınlar arsındaki uzaklığı hesapla ve uzunluk dizisini ata.
66- Oyun tahtasındaki altın sayısını say. Eğer altın sayısı 1 ise ilk if' e gir ve oyuncuların o altına olan uzaklıklarını hesapla.
67- Diğer oyunculardan daha önce D oyuncusunun kalan son altına ulaşıp ulaşamayacağını kontrol et.
68- Eğer D oyuncusunun kalan adım sayısı 4 den daha küçük ise direkt hedeflediği altına git.
69- Eğer altın sayısı 1'den farklı ise else kısmına gir.
70- A, B, ve C oyuncularının hedeflediği altınlar arasındaki uzaklığını al.

71- Her oyuncu için o oyuncunun hedefine ondan önce gidip gidemeyeceğine bak.
72- Eğer gidebiliyorsa değişkene 1 değerini ata.
73- Eğer gidemiyorsa kendinden önceki oyuncuların hedeflediği altınlar dışındaki altınların hepsini D_Map'e ata.
74- D_Map de altın var mı yok mu diye kontrol et.
75- Eğer yok ise D oyuncusunu oyundan at(oyun tahtasında durdur).
76- Eğer oyun tahtasında oyun var ise oyunculardan herhangi birinin altınına ulaşırsa o oyuncuya ait if koşuluna gir.
77- Oyuncunun hedefine gidebiliyorsa hedefine yönel.
78- Eğer gidemiyorsa oyun tahtasındaki diğer altınların yollarını tut.
79- Kar hesabı yapmak için Maliyet_D() fonksiyonuna git.
80- D oyuncusunun konumu ve oyun tahtasındaki altınların değerlerini ve konumunualarak kar hesabı yapıp kar_hesabı dizisine ata.
81- En karlı olan yolu hesapla next tipindeki yol_D dizisine ata.
82- Ardından renk() fonksiyonunu çağır.
83- Renk() fonksiyonunda gidilen yolu renklendir.Her hamlenin maliyetini kasadan düş. Eğer gizili altın üstünden geçerse normal altına çevir. Eğer altın üzerinden geçerse kendi hedeflediği altın olup olmadığını kontrol et.
84- Eğer kendi hedeflediği altın ise altının maliyetini D oyuncusunun kasasına ekle.
85- Bu işlemler tamamlandıktan sonra beyazYap() fonksiyonu çağırarak renklendirilen kısımlar tekrardan beyaz hale getirilir. Eğer hedefine ulaştıysa diğer oyuncuya sırasını vermeden hedef belirlet.
86- D oyuncusu için if koşulu sağlanmaz ise else koşuluna gir. Eğer D oyuncusunun hedefi başka bir oyuncu tarafından alındıysa yeni bir hedef belirle.
87- Hedefi belirli ise altını başka oyuncu tarafından alınmadı ise renk() fonksiyonu çağrılarak adım atmaya devam et.
88- Bu işlemleri oyuncuların kasasındaki altınlar bitene kadar ya da oyun tahtasındaki altınlar bitene kadar devam et.

Karşılaştığımız Sorunlar

Nasıl bir ara yüz kullanacağımız konusunda ve de ara yüzü tasarlarken sorunlar yaşadık.

Kullanacağımız arama algoritmasını hemen kararlaştırmamıza rağmen bu algoritmayı Altın Toplama oyununa nasıl uyarlayabileceğimiz konusunda sorunlar yaşadık.

D oyuncusunun kendinden önceki oyuncuların hedeflerine göre hareket etmesi için yazdığımız algoritma da çok fazla seçenek üzerine düşünüp bu algoritmayı yazmak konusunda sorunlar yaşadık.

Projenin Bize Kattığı Yararlar

Dijikstra arama algoritmasını bir oyuna nasıl uyarlanabileceğini öğrendik.

Oyun içi animasyon yapımını öğrendik.

Dinamik özelliklere sahip bir proje yaparak uygulama başlamadan önce değiştirilen değerleri uygulamaya implement etmeyi öğrendik.

Kullanılan fonksiyonlar

public void Dijkstra()

Altına olan en kısa yolun hesaplandığı fonksiyon

public void explore(Node current, int lastx, int lasty, int hops)

Altına olan en kısa yol için arama yaparken her hücrenin etrafındaki komşuları keşfetmeye yarayan fonksiyon.

public void backtrack(int lx, int ly, int hops)

Atılan adım sayısı kadar yolun geriye doğru sönen renkler ile görselleştirmesini sağlayan fonksiyon

public void silcyan()

A oyuncusunun kendisine en yakın hedefi ararken görselleştirme olarak Cyan rengi seçildi. Bu rengin arama işlemi bittikten sonra ortadan kaldırılması için kullanılan fonksiyon.

public void altin_sayisi()

Oyun esnasında kaç altın olduğunu bulan fonksiyon.

public void renk()

A,B,C ve D oyuncularının oyun ekranında renklerinin gösterilmesini sağlayan fonksiyon.

public void beyazYap()

Oyuncular yer değiştirdiğinde bir önceki konumun tekrardan beyaz yapılabilmesi için kullandığımız fonksiyon.

public void altin_degerAta()

Oyun ekranına random olarak altın atamasını yapan fonksiyon.

public void GizliAltin()

Oyun ekranına random olarak gizli altın atamasını yapan fonksiyon.

public void altinata()

Gizli ve gizli olmayan altınlara değer atamasının yapıldığı fonksiyon.

public void arama_baslat()

Oyunun başlamasını sağlar

public void durdur()

Oyunun durdurulmasını sağlar

public void harita_temizle()

Oyun haritasındaki bütün oyuncu, altın ve gizli altınları temizleyen fonksiyon.

private void initialize()

Bütün ara yüz ve içindeki komponentlerin bulunduğu fonksiyon.

public void delay()

Oyunu belli bir süre yavaşlamasını sağlar

public void delay1(int sayi)

Oyunu verilen parametresi kadar belli bir süre yavaşlamasını sağlar

public void Guncelle()

Oyuncudan alınan Altın sayısı ve Gizli altın sayısına göre anlık olarak altın atamasını günceller aynı zaman da kullanıcıdan alınan satır ve sutun boyutuna göre tekrardan oyun haritasını güncelleyerek anlık bir şekilde görselleştirmesini yapar.

public void Guncelle1()

Oyun başlamadan önce girilen toplam puanı oyuncuların kasasına ekler.

public void kontrol()

Oyun tahtasında altın olup olmadığını kontrol eder eğer altın varsa ve sıra hangi oyuncudaysa o oyuncuya true değeri atayarak o ouncunun oyun içinde devam etmesini sağlayan fonksiyondur.

public void ozet_Tablo()

Oyun sonlandığında ekrana oyuncuların oyunla ilgili özet verilerini basan bir özet tablo oluşturan fonksiyon.

public void oyundaMi()

Oyuncuların oyun esnasında hedef belirlemeden ve adım atmadan önce kasadaki toplam altın sayısının yetip yetmeyeceğini kontrol eden fonksiyon.

public void maliyet()

B oyuncusu için altınlar arasında en karlı olanı seçip o yolu “next” tipinde oluşturulmuş bir diziye atar.

public void Maliyet_C()

C oyuncusu için altınlar arasında en karlı olanı seçip o yolu “next” tipinde oluşturulmuş bir diziye atar.

public void Maliyet_D()

D oyuncusu için altınlar arasında en karlı olanı seçip o yolu “next” tipinde oluşturulmuş bir diziye atar.

public void degersifirla()

B,C ve D oyuncuları için oluşturulan sanal matrislerin tekrardan Dijikstra Algoritmasının çalışması için uğranmamış olarak(-1) atanmasını sağlar.

public void C_hesapla()

İlk olarak C oyuncusu oyun tahtasındaki gizli altınları bulur ve kendisine en yakın 2 tane gizli altını açar. İkinci olarak da oyun tahtasındaki altınların yollarını ve koordinatlarını alır ve kendisinin altınlara olan uzaklığını bir dizide tutar.Ardından da Maliyet_C() fonksiyonuda çağırılır.

public void hesapla_D()

D oyuncusu kendisi haricindeki diğer oyuncuların altın hedeflerine bakarak onlardan önce ulaşıp ulaşamayacağını kontrol eder. Eğer ulaşabiliyorsa o altına doğru yönelir. Şayet ulaşamıyorsa kendisine en karlı hedefi belirler.En son olarak da Maliyet_D() fonksiyonu çağrılır.

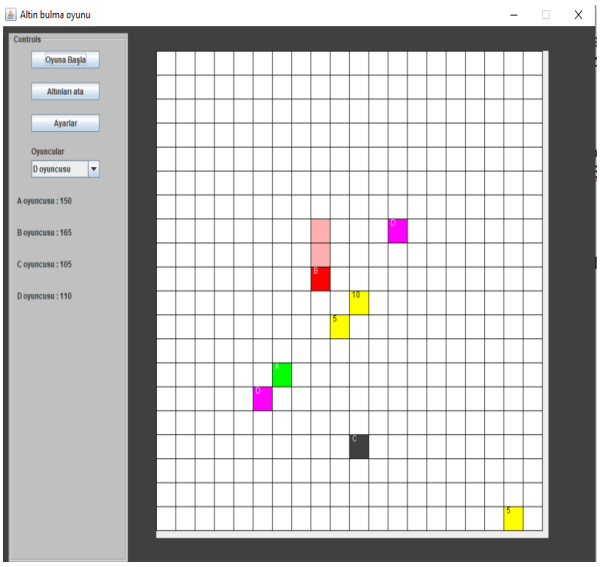
public void hesapla()

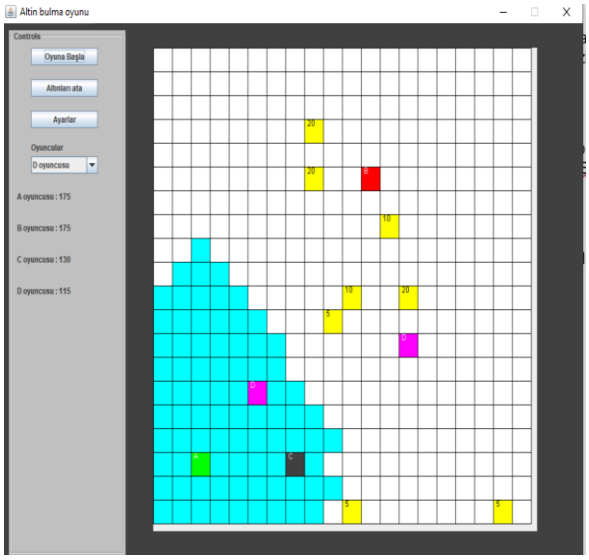
B oyuncusu oyun tahtasındaki tüm altınları bir matriste tutar ve bu altınların kendisine olan uzaklığını bir path adındaki bir dizide tutar. En son olarak da Maliyet fonksiyonunu çağırır.

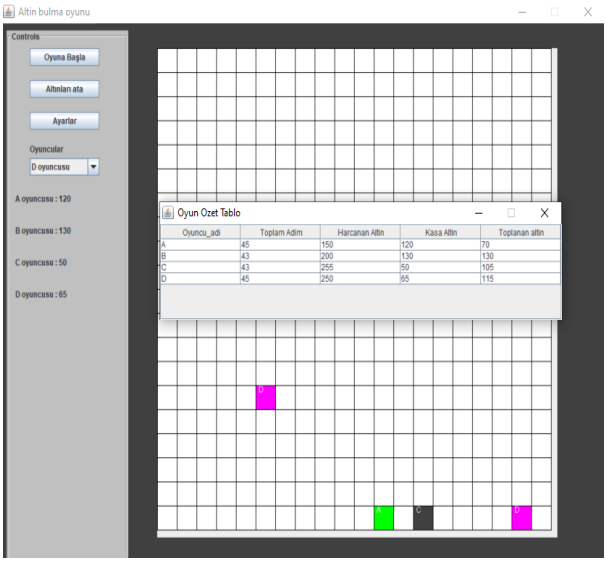
Public void altin_ata()

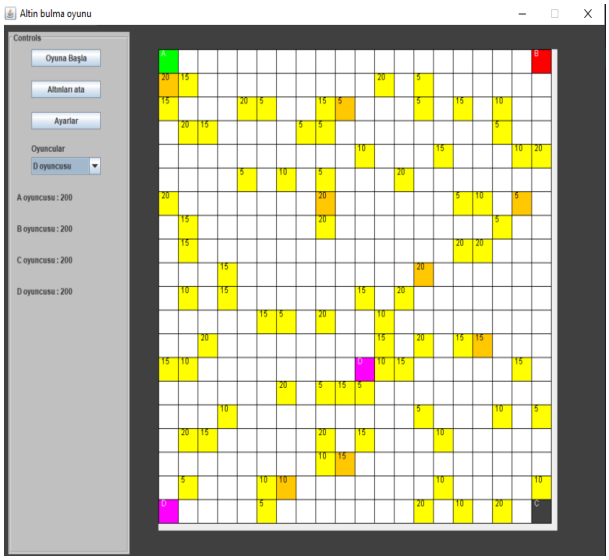
Maliyet dizisine altın değerlerini atar

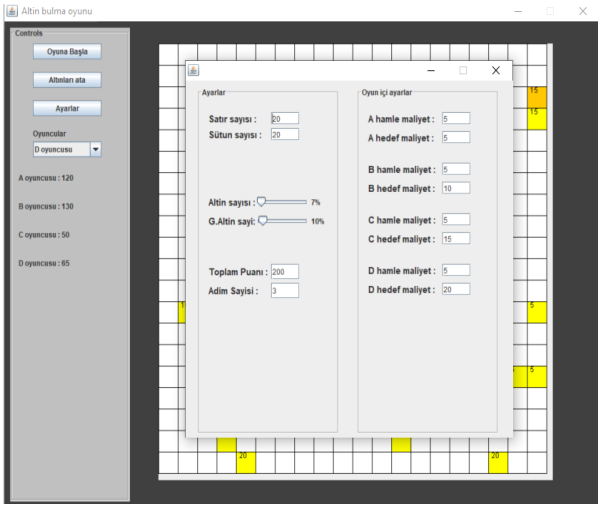
Ekran Görüntüleri

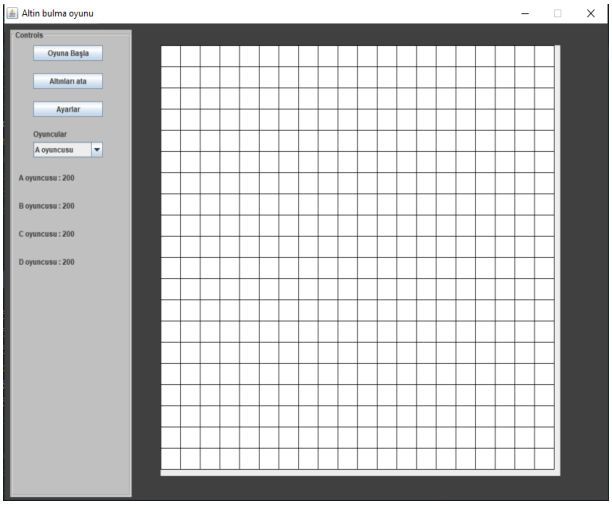












Kaynakça

Path Finding

https://github.com/greerviau/Path-Finding/blob/master/source/PathFinding.java

Java Graphics Interface III - paintComponent

https://www.bogotobogo.com/Java/tutorials/javagraphics3.php

MANHATTON Path

http://www.haruneskar.com/2008/09/manhattan.html