Oyun çözülecek labirentin nasıl üretileceğini soran 2 seçenekle başlıyor:

1. Labirenti dosyadan oku
2. Rastgele labirent oluşturma

Kullanıcı 1. seçeneği (Labirenti dosyadan okuma) seçtiğinde:

- Kullanıcıdan programın çözeceği labirentin dosya yolu *string* veri tipinde *filePath* değişkeni olarak alınır.
- Alınan dosya yolundan labirenti iki boyutlu integer array'e dönüştürmek için *getMazeFromFile()* metodu *filePath* değişkeni parametre olarak verilerek çalıştırılır. Bu metod iki boyutlu array olan bizim matrisimizi geri döndürür. Bu matris bizim labirentimiz olacaktır. Bu labirent matrisini **maze** değişkeni olarak bellekte saklarız.

Kullanıcı 2. seçeneği (Labirenti rastgele oluşturma) seçtiğinde:

- Kullanıcıdan programın oluşturup çözeceği labirentin dosya adı *string* veri tipinde *fileName* değişkeni olarak alınır.
- *MazeManager* sınıfından *createMaze()* metodu ile iç içe for döngüleri ile rastgele bir 30*30 integer matris oluşturulur ve maze değişkeni olarak belleğe kaydedilir.
- Oluşturulan maze ve fileName değişkenleri *MazeManager* sınıfındaki *writeToTextFile()* metoduna parametre olarak verilerek çalıştırılır.
- Ardından *PathManager* sınıfından *createRandomPath*() metodu ile bir yol oluşturulur. Bu metod, yolun her bir noktasının koordinatlarının x ve y değerlerini barındırabilen Coordinat sınıfını içeren *List<Coordinat>* veri tipinde bir liste döndürür.
- Gelen yol listesi ve labirent matris *PathManager* sınıfındaki *placePath()* metoduna parametre olarak verilir ve matrisimize bir yol oluşturulur. Sonuç olarak elimizde yolu çizili bir labirent matrisimiz vardır. Bu labirent matrisini **maze** değişkeni olarak bellekte saklarız.

Yukarıdaki işlemlerin devamı ortak şekilde **maze** değişkeni ile devam edecektir

---

- Elde ettiğimiz labirenti çözdürmek için *MazeSolver* sınıfından bir nesne oluşturulur ve bu nesneye **maze** değişkeni parametre olarak verilir. Arsından *solveMaze()* metodu çalıştırılır. Bu metod geriye yol kordinatlarını gönderir. Yol kordinatları *List<Coordinat>* türünde **path** değişkeni olarak belleğe kaydedilir.
- Şimdi labirente bomba yerleştireceğiz. *MiningManager* sınıfındaki *getRandomMines()* metodunu labirent matrisini ve yol koordinatları girerek çalıştıracağız. Bu method bize bomba koordinatlarını *List<Coordinat>* türünde geri döndürecek.
- Artık elimizde *List<Coordinat>* türünde 3 adet bombanın koordinatlarını içeren bir **mines** değişkeni, labirenti 1 ve 0 dan oluşan iki boyutlu bir array olarak barındıran **maze** değişkeni ve labirentin yol koordinatlarını barındıran **path** değişkeni bulunur.
- Sonra *startGame()* metodu çalıştırılarak oyun başlar. Bu noktadan sonra ekrana labirent labirent *printMaze()* metodu ile yazdırılır. Kullanıcı X tuşuna basarsa labirentin yolu, B tuşuna basarsa bombaların yeri, L tuşuna basarsa labirentin normal hali ekrana yazdırılır.

Yukarıda verilen en önemli sınıflar ve metotların çalışma mantığı aşağıdadır.

⬇

Labirenti çözdüren metodlar

solveMaze()
Programın en karmaşık kısmı burası. Labirenti bilgisayarı çözebilmesi için başlangıç ve bitiş noktalarına ihtiyacımız var. Labirent başlangıç ve bitiş noktalarını parametre alarak çözmeye çalışacaktır. Fakat labirentin başlangıç noktası ve bitiş noktası da belli değil. Bunun için labirentin başlangıç noktaları için sol köşeden başlayarak bütün 0 koordinatları bir listeye ekledim. Bitiş koordinatları için 30. satır ve 30 sütunu ekledim. Sonra bu noktalar arası çözüm var mı diye **move()** metodu ile tek tek kontrol ettim.

move()
Başlangıç noktasının koordinatı noktaların hareket edilebilir olup olmadığını (duvar olup olmadığını) **isValidPoint()** ile kontrol ettim. **isValidPoint()** o koordinat üzerinde duvar olup olmadığını kontrol eder. Duvar varsa false, duvar yoksa true değeri döndürür. true değeri dönerse noktamız oradan geçebilir demektir. Sonrasında noktanın üstünü, sağını ve solunu kontrol edilir. Nokta hareket etmeye uygunsa bu nokta yol listemize eklenir ve yine **move()** metodu çağrılır. Sonuç olarak noktamız bitiş noktamıza eşit olduğunda elde kalan yol koordinatlarımız yolumuzu ihtiva eder. Şayet move() false değeri döndürürse çözüm bu iki nokta için başarısız olarak sona erer.

Diğer yardımcı metodlar

getMazeFromFile()
Bu metod içine aldığı dosya yolu parametresi ile o dosya yolundaki txt dosyasını okur. Txt dosyasından gelen string veriler 30 ayrı satıra string veri tipinde bölünür. Her bir satır için for döngüsü açılır ve 0 ile 1 değerleri stringten integere çevrilerek 30*30 matrisimize atılır. Köşeli parantezler ilk satırda ve son satırda farklı olduğu için ilk ve son satıra özel if koşulları bu metotta yer alır.

createMaze()
30*30 int türünde bir matris tanımlanır ve iç içe for döngüleri ile 0 veya 1 değeri her bir haneye yazılır. Ortaya çıkan matris geri döndürülür.

writeToTextFile()
Bu metot içine aldığı matris parametresini (yani labirent) txt dosyasına yazmak için StreamWriter yapısını kullanır. İç içe for döngüleriyle her bir 0 ile 1 değerini string'e çevirip dosyaya yazar.

createRandomPath()
Bu metodun asıl amacı rastgele oluşturmuş olduğumuz matrise yol çizmek. Rastgele oluşturduğumuz için yol oladabilir olmayadabilir. Bir yol ihtiva edebilmek için bu metotta matrise yol çizeriz. 0,0 koordinatından başlarız ve bunu bir listeye eklenir. While döngüsü başlatılır. %50 ihtimalle ya x koordinatı ya da y koordinatını bir artırıp yeni oluşan koordinatı listeye eklenir. Döngünün herbir adımında listenin son elemanına bakarak yeni oluşacak noktayı oluşturulur. Döngü listedeki koordinatın son elemanının x veya y koordinatı 29'a eşit olduğu anda döngü kırılır. Geriye kalan liste bizim yolumuz olur ve bunu geri döndürürüz.

placePath()
Bu metot rol koordinatlarını içeren bir listeyi ve labirent matrisini parametre olarak içine alır. İç İçe for döngüsüyle yol koordinatları labirent matrisinin aynı noktaları 0’ eşitlenir. Böylece çözümü kesin olan bir labirent matris elde ederiz.

getRandomMines()
Bu metod aldığı iki boyutlu labirent matrisini v yol listesini parametre olarak alır. Ardından bombalar bir liste olarak tanımlanır. while döngüsü bomba sayısı 3 olmama koşuluyla başlatılır. Döngünün içinde 0,29 arası rastgele c ve y koordinatlarını sahip bomba üretilir. Bu bombanın koordinatı labirent matrisi üzerinde kontrol edilir. Yol listemizle bir eşleşme olursa o bombalar eklenmez. Eğer o noktada yol listemizin koordinatlarında bir nokta yoksa o bomba koordinatı listeye eklenir ve liste geri döndürülür.

printMaze()
Kullanıcın bastığı tuşlara bağlı olarak ekrana labirenti farklı şekillerde yazar.