Otonom Hazine Avcısı

Berat Ölmez, Sait Ömer San
210201074, 220201001
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
Kocaeli/Türkiye
beratolmez123@hotmail.com
sainsaitömer@gmail.com

***Özetçe*—**Bu proje, geçmiş dönemde öğrenilen nesneye yönelik programlama ve veri yapıları bilgilerinin uygulanması ve problem çözme becerisinin geliştirilmesi amacı ile yapılan, oyun ve çeşitli yazılım geliştirme kavramlarına odaklanılmış, otonom karakter hareketine dayanan bir oyun ve yazılım projesidir.

Anahtar Kelimeler —Veri Yapıları, Nesne Yönelimli Programlama (OOP), Abstraction (Soyutlama), Encapsulation (Kapsülleme), (Inheritance) Miras, Polymorphism (Polimorfizm), DFS (Depth First Search), Interface

I.GİRİŞ

Bu proje, nesneye yönelik programlama ve veri yapıları konularındaki bilgileri pekiştirmeyi, uygulamalarını sağlamayı ve problem çözme becerilerini geliştirmeyi hedeflemektedir. Oyunun amacı, otonom hareket eden bir karakterin, rastgele oluşturulan bir harita üzerinde çeşitli engelleri aşarak en kısa sürede tüm hazine sandıklarını toplamasını sağlayacak bir algoritmanın tasarlanması ve bu algoritmanın grafik birimleri ile gösterilmesidir. Bu proje kapsamında, programlama dili olarak Java kullanılmıştır. Karakterin engellere takılmadan en kısa yoldan tüm hazineleri toplamasını sağlamak için geliştirilecek yazılım, mantıksal düşünme yetenekleri ve algoritmik yaklaşımları geliştirmeyi amaçlamaktadır. Bu rapor, projenin tasarım sürecini, kullanılan yöntemleri, karşılaşılan zorlukları ve elde edilen sonuçları detaylı bir şekilde ele alacaktır.

II.YÖNTEM

A. Nesnelerin Tanımlanması

Engeller, hareketli engeller ve karakteri tanımlamanın ve oluşturmanın temelini oluşturmak adına nesne yönelimli programlamanın temel kavramlarından olan abstraction (soyutlama) kavramını kullanarak soyut 'Element' sınıfını oluşturduk. Ekranda belirecek her nesne bu element sınıfının alt sınıfıdır, bu sınıf görüntüsü olan nesnenin atasıdır. Element sınıfı x ve y koordinat bilgilerine, haritada görüntülemek için image özelliğine sahip. Her alt sınıfın kendine ait özel görselleri bulunur. Rastgele koordinat oluşturma ve oluştuğu koordinatta haritada işaretleme yapma, oluşturulan koordinatı kontrol etme, getter, setter metotlarına sahiptir. Yapıcı (Constructor) metodu oluşturulacak alt nesnenin çağırılırken kendi koordinatlarını oluşturması ve harita matrisinde kendine bir yer edinmesini sağlar. Her nesnenin koordinatı olması gerektiği için nesne yönelimli programlamanın az kod çok iş prensibini burada uyguluyoruz.

1. Character Sınıfı

Karakterimiz hareket etme, sandık bulma, gittiği yolu renklendirme gibi metotlara sahiptir. Kalıtıldığı 1x1 birimlik yer kaplar. 7x7 lik görüş alanına sahiptir ve görüş alanına giren sisleri hareketi sırasında haritadan silecektir.

2. Obstacle Sınıfı ve Alt Sınıfları

Engeller yaz engelleri (Summer Obstacle), kış engelleri (Winter Obstacle) ve hareketli engeller (Moving Obstacle) olmak üzere 3 adet soyut alt sınıf ve yaz ve kış bölgelerine bağlı kalmayan 10x1 boyutunda duvar (Wall) ve 1x1 boyutunda kaya (Rock) olmak üzere 2 adet alt sınıf kalıtılır. Her engel haritada oluştuğu koordinatın çevresinde kapladığı birimkare alan kadar harita matrisinde 1 işaretler.

Matris işaretlemeleri nesnelerin oluşurken üst üste çakışmasını önlemek ve karakterin hareketi sırasında engelleri tanımasını sağlamak amacıyla kullanılır.

2.1 Yaz Engelleri

Yaz engelleri haritanın sadece sağ yarısında oluşmak zorunda olan ve koordinat oluşturma metodu bu koşulu sağlamak amacıyla özelleştirilmiş engel sınıfıdır. Bu soyut sınıftan 3x3 boyutunda ağaç (Tree) ve 15x15 boyutunda yaz dağı (Summer Mountain) ismi verilen alt sınıflar kalıtılır.

2.2 Kış Engelleri

Kış engelleri haritanın sadece sol yarısında oluşmak zorunda olan ve koordinat oluşturma metodu bu koşulu sağlamak amacıyla özelleştirilmiş engel sınıfıdır. Bu soyut sınıftan 3x3 boyutunda çam ağacı (Pine Tree) ve 15x15 boyutunda kış dağı (Winter Mountain) ismi verilen alt sınıflar kalıtılır.

2.3 Hareketli Engeller

Hareketli engeller haritanın herhangi bir bölgesinde oluşabilir. Bu soyut sınıftan 2x2 boyutunda 3 birim sağ ve sol yönlü hareket becerisine sahip arı (Bee) ve 2x2 boyutunda 5 birim yukarı ve aşağı yönlü hareket becerisine sahip kuş (Bird) sınıfı kalıtılır.

3. Chest Sınıfı

Sandık sınıfı harita üzerinde karakter tarafından toplanması gereken önemliden önemsize sırasıyla altın (Gold Chest), gümüş (Silver Chest), zümrüt (Emerald Chest) ve bronz (Bronze Chest) olmak üzere 1x1 boyutlarında 4 adet alt sınıfı kalıtan soyut sınıftır. Oluştuğu koordinatta harita matrisi üzerinde 2 işaretlemesini yapar. Karakter sandıkları haritada 2 ile işaretlenen koordinatlar ile tanıyacaktır.

4. Sis Sınıfı

Haritanın oluşturulması aşamasında karakterin görüş alanı dışında tüm haritada 1x1 boyutlarında oluşmak ile görevlidir. Sis ekleme ve sis güncelleme metotlarına sahiptir.

5. En Kısa Yol Sınıfı

En kısa yolun bulunması için kodda en kısa yol sınıfı tanımlanmıştır. Bu sınıf verilen koordinatlar arası karakterin engellerden geçmeyerek sandıklara ulaşabileceği en kısa yolun hesaplanması amacıyla hazırlanmıştır. Bu sınıfta BFS(Breadth First Search) algoritması kullanan findshortestpath fonksiyonun kullanması için bağlı liste oluşturulmuştur. findshortestpath fonksiyonu Karakterin gidebileceği 4 yönde hareket esnasında engel olup olmadığını ve daha önce ziyaret edilip edilmediğini hesaplamaktadır. Bu sınıf kuyruk yapısını kullanarak bir noktadan başka bir noktaya hareket edilmesi için engellerden geçilmeyecek şekilde en kısa yolun koordinatlarını BFS algoritmasıyla bulmaktadır ve bu koordinatları liste olarak döndürmektedir. Sandık bulma algoritmasında görünür bölgedeki sandıkların toplanması için bu sınıftaki findshortestpath fonksiyonu kullanılmıştır.

B. Arayüz ve Harita Oluşturma

Projemizin arayüzü JavaFX yazılım aracı kullanılarak geliştirilmiştir ve kullanıcıdan gelen verileri işleyerek oyunu oluşturmakla görevlidir.

İlk panel kullanıcıdan harita boyutu ve birim boyutu bilgilerini ister. Kullanıcı isteğine göre dağ engellerinin boyutunu da buton yardımı ile açılan yeni panel aracılığı ile değiştirebilir. Girilen verilere göre her nesne sırası ile oluşturulur ve çağrılır. Eğer oluşturulacak nesneler harita üzerinde oluşacak yer bulamazsa program başa sarar ve işlem başarılı olana kadar devam eder. İşlem başarılı olursa ilk panel kapatılır ve yeni panel açılır. Bu panelde nesneler bütünü ile gösterilir. Panelin üst bölümünde bulunan başlat butonu ile kullanıcı karakterin hareketini başlatabilir. Karakter her sandık bulduğunda panelin sağ bölümünde sandığın konum bilgilerini toplandı olarak görebilir.

Haritanın arka paneli her biri boyut olarak küçültülmüş resimlerden oluşur. Kış bölgesi temasına uygun olarak farklı resimle donatılırken yaz bölgesi temasına uygun olarak farklı resimle donatılır.

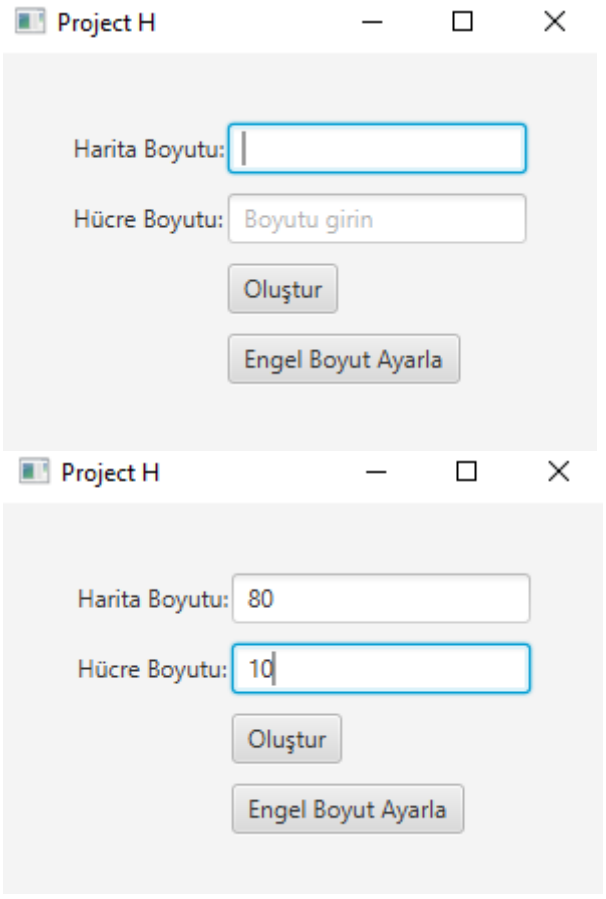
C. Karakter Hareketi

Kodda karakter ve hareketli engellerin hareketi için JavaFX kütüphanesi içinde bulunan Timeline ve

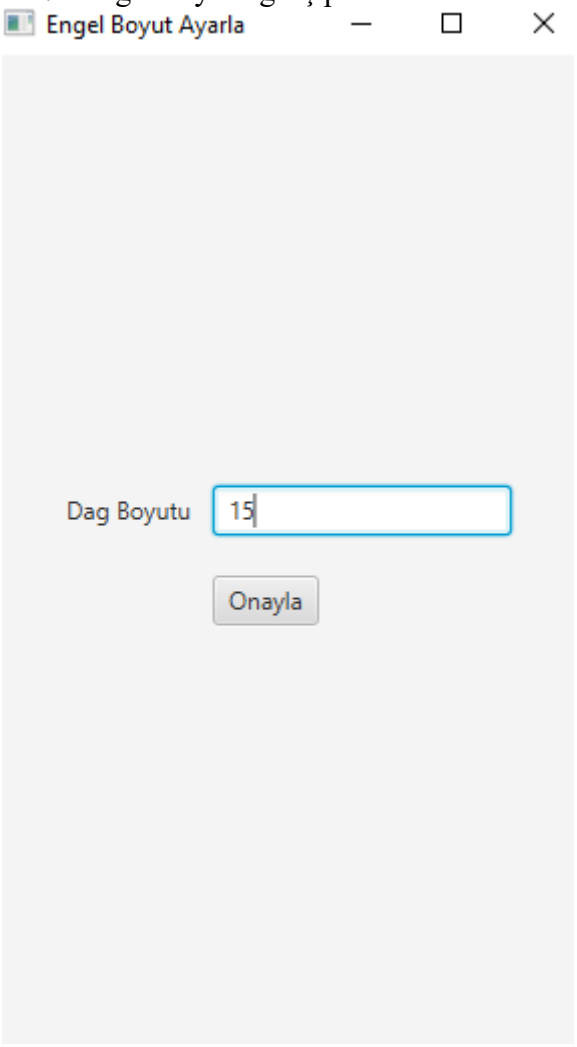
Key-Frame kullanılmıştır. Key-Frame, animasyonun belirli bir süre boyunca hangi değeri alacağını tanımlar. Bir zaman dilimini ve o zaman dilimindeki değerleri belirtir. Timeline, KeyFrame'leri içeren ve bunların hangi zaman diliminde uygulanacağını belirten bir yapıdır. Yani, bir animasyonun zaman çizelgesini belirleyen ve KeyFrame'lerin ne zaman uygulanacağını kontrol eden yapıdır. Birden fazla KeyFrame, belirli bir zaman aralığında bir Timeline içinde birleştirilir ve bu zaman aralığında animasyon gerçekleşir. Karakterin ve hareketli engellerin hareketinin aynı anda gerçekleşmesi farklı Timeline'lar üzerinden gerçekleştirilmiştir bu sayede arılar kuşlar ve karakter aynı anda hareket edebilmiştir. Kodda karakterin hareketi 2 farklı yoldan da yaptırılmıştır. Bunlardan birincisi sandiklara_git_7x7 fonksiyonudur. Bu fonksiyon tüm sandıkları bulancaya kadar bir while döngüsü içindeki işlemleri yapmaktadır. Bu işlemlerden ilki karakterin toplam görüş alanında sandık yoksa karakterin hareketini sis olan bir koordinata götürür. İkincisi karakterin toplam görüş alanında sandık varsa en kısa yolu bulma sınıfında yer alan FindShortestPath isimli fonksiyonla sandığa en kısa yol üzerinden gidilir. Bu iki işlem haritadaki sandıklar toplanıncaya kadar tekrarlanır. Kodda kullanılan ikinci hareket yöntemiyse sandiklara_git fonksiyonu tarafından yapılır. Bu yöntemde yapılan yol bulma işlemi görüş açısı kullanılmadan direkt olarak en kısa yolu bularak sandıkların toplanmasıdır. Bu işlemi gerçekleştirmek için sandıklar karakterin doğduğu noktaya ve daha sonra sandığı bulduğu noktalara bağlı olarak sıralama algoritması kullanılarak sıralanmıştır. Bunun sebebi karakterin başlangıçta en uzak sandığa gitmesi gibi durumların önlenmesidir. Karakter sandıkları bulduktan sonra toplam adım sayısı yazdırılmıştır.

III.DENEYSEL YÖNTEM

1. Kullanıcıdan harita boyutu ve birim boyutu verileri girişi

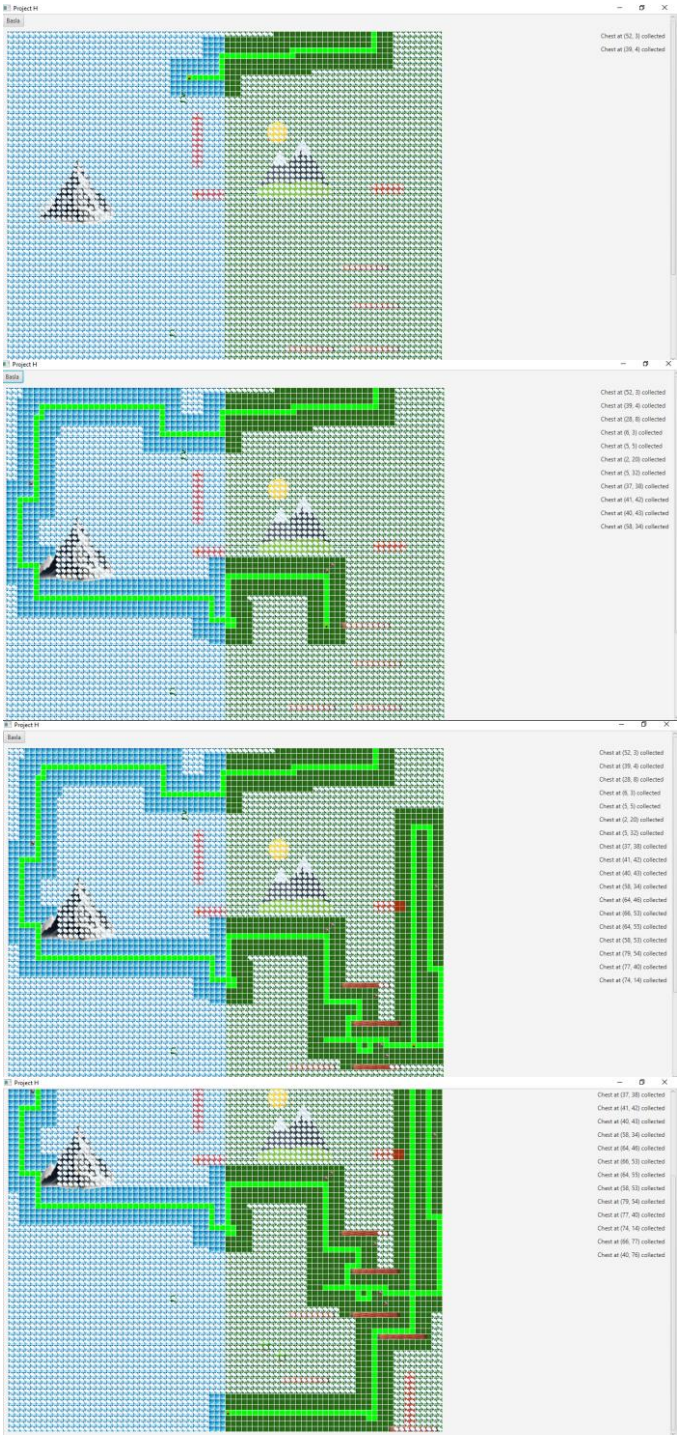


2. Engel boyutu giriş paneli

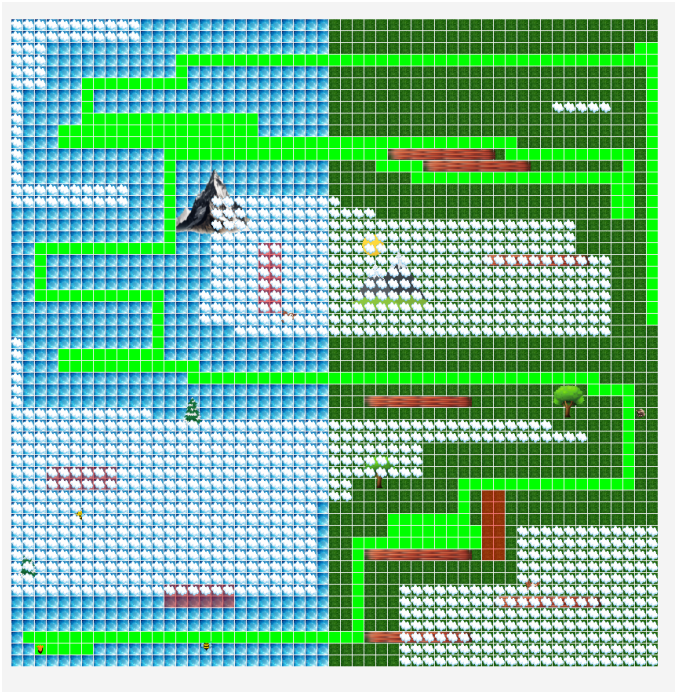


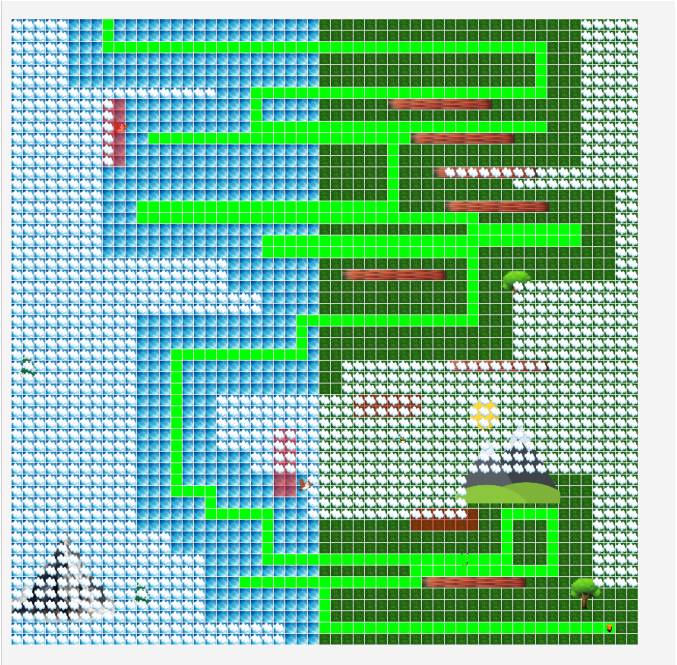
3. Haritanın oluşturulması

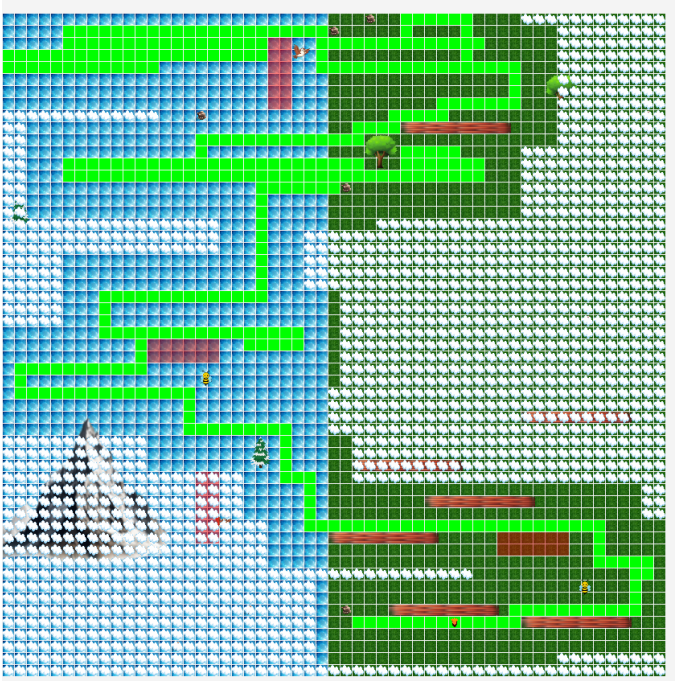
4. Karakter hareketi

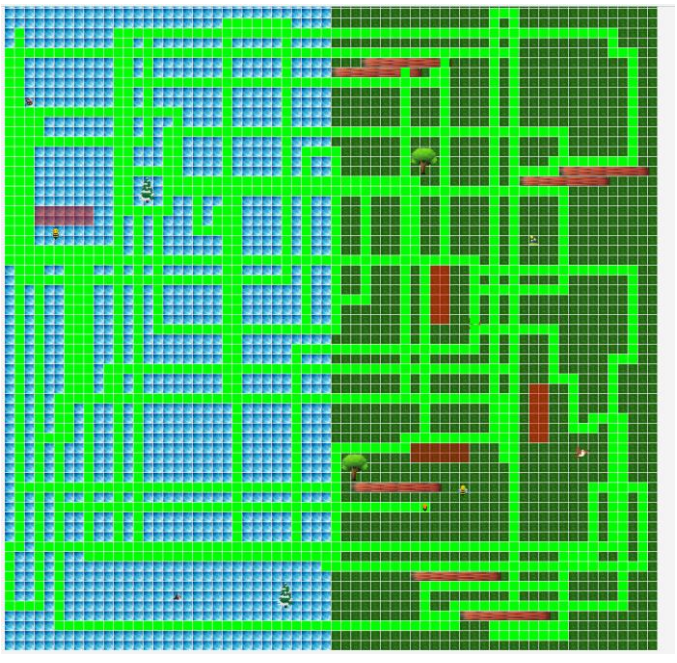


5. Karakterin 7x7 görüş alanına sahip olduğu hareket haritası









IV. SONUÇ

Yapılan deneyler girilen farklı harita boyutlarında programın başarılı bir şekilde haritayı ve nesneleri oluşturduğunu ispatlamıştır. Karakterimiz gerçekleştirdiği hareketi boyunca engelleri tanımış, yok sayıp üzerlerinden geçmemiş, sandıkların hepsini doğru sıralama ile bulmuştur. Bu proje oyun geliştirme ve nesne yönelimli programlama metotlarını gerçekleştirme açısından başarıya ulaşmıştır.

V.UML DİYAGRAMI

KAYNAKLAR

- https://www.youtube.com/watch?v=KiCBXu4P-2Y
- https://docs.oracle.com/javase/8/javafx/api/toc.htm
- https://www.javatpoint.com/javafx-tutorial