Otonom Hazine Avcısı

Meryem PAŞAOĞLU
Bilgisayar Mühendisliği
220201123@kocaeli.edu.tr

***Abstract*—Bu proje, otonom hareket eden bir karakterin, içerisinde çeşitli hazineler ve engeller bulunan bir harita üzerindeki hazineleri topladığı bir oyun tasarımı içerir . Oyunda amaç, karakterin tüm hazineleri en kısa sürede toplamasını sağlayacak algoritmanın tasarlanmasıdır.**

I. GİRİŞ

Proje C sharp dili ile Visual Studio kullanılarak hazırlanmıştır. Toplamda 2 Form oluşturulmuştur. Projede, altınları engellere rağmen en az adım sayısı ile toplayan bir karakter bulunmaktadır. Uygulama arayüzden çalıştırılmaktadır.

A. isterler

Projede istenilen isterler şu şekildedir:

- Rastgele oluşturulmuş bir harita üzerinde, belirlenen başlangıç noktasından hareket ederek, engellerle karşılaşmadan ızgaradaki tüm hazine sandıklarını toplamak için karakterin en kısa sürede ve en kısa yol üzerinden hareket etmesi gerekmektedir. Oyun boyunca karakterin tüm ızgarayı değil, sadece gerekli yolları gezmesi önemlidir.
- Program ilk kez çalıştırıldığında, kullanıcıdan haritanın boyutuyla ilgili bilgi alınmalıdır. Çalışmalar küçük boyutlu haritalarda gerçekleştirilebilir; ancak, sunum sırasında 1000x1000 boyutlu haritalar oluşturulması istenebilir. Kullanıcıdan alınan harita boyut bilgisine göre, her uygulama başladığında harita yeniden oluşturulmalıdır.
- Geliştirilen algoritma, her adımda rastgele bir harita üretmek ve bu haritanın doğruluğunu doğrulamak için etkili yöntemler içermelidir. Kaba kuvvet uygulamalarından kaçınılarak, mümkün olduğunca verimli çözümler geliştirilmelidir. Bu doğrultuda, harita üretimi için kullanılan algoritma, rastgele dağıtılmış engellerle ve hazine sandıklarıyla birlikte haritayı oluşturmalıdır. Doğrulama işlemi ise, oluşturulan haritanın geçerliliğini kontrol ederek gerçekleştirilmelidir. Bu sayede her adımda rastgele bir harita üretilip doğrulandığından emin olunabilir.
- Harita oluşturulduğunda, sol taraf kış temasını yansıtırken sağ taraf ise yaz temasında olmalıdır. Bu doğrultuda, bir sonraki adımda anlatılacak sabit nesnelerin görselleri, belirtilen kış ve yaz temalarına uygun olarak değiştirilmelidir. Bu yaklaşım, haritanın görsel bütünlüğünü ve atmosferini güçlendirecektir.
- Tüm hazine sandıklarının ulaşılabilir (engellerin ortasında kalmadığından) olduğundan emin olunması gerekir.
- Harita üzerinde, yol, engeller ve hazine sandıklarını içeren farklı tiplerde kareler bulunmalıdır. Engeller iki çeşit olup, sabit ve hareketli olarak tanımlanmaktadır. Sabit engeller, yaz ve kış temalarına göre ayrıştırılmalıdır ve ağaç, dağ, kaya ve duvar gibi hareket etmeyen nesneleri içermelidir. Hareketli engeller ise kuş ve arı gibi nesnelerdir. Kuşlar 5 karelik bir alanda yukarı-aşağı, arılar ise 3 karelik bir alanda sağ-sol şeklinde hareket etmelidir. Her bir hareketli nesne, 2x2 birimlik bir alanı kaplamalıdır. Bu şekilde çeşitli engel tipleri ve hareketli nesneler haritadaki dinamizmi artıracaktır.
- Her harita oluşturulduğunda, en az 20 adet sabit nesne (her sabit nesneden en az 2 adet oluşturulması zorunludur) ve 3 adet hareketli nesnenin üretilmesi ve harita üzerinde uygun konumlara yerleştirilmesi beklenmektedir. Harita üzerinde hareketli engellerin izlediği yol kırmızı renkle gösterilmelidir. Bu yaklaşım, haritanın çeşitliliğini artırarak oyun deneyimini zenginleştirecektir.
- Karakter bir sandığı topladığında, ilgili sandık harita üzerinde kaybolmalıdır. Sandıkların toplanma sırasına göre (altın sandık, gümüş sandık, zümrüt sandık ve bakır sandık) ekranın sağ üst köşesinde sandığın türüne göre bilgi verilmelidir. Örneğin, toplanan sandık altın sandık ise ”Altın sandık toplandı! (8,16-8,18) konumunda bulundu” gibi bir mesaj görüntülenmelidir. Bu bilgi, oyuncuya topladığı sandık türü ve konumu hakkında detaylı bir geribildirim sağlayacaktır.
- Her harita oluşturulduğunda her sandık türünden en az 5 adet üretilmiş ve harita üzerinde uygun yerlere yerleştirilmiş olması beklenmektedir.
- Karakterin başlangıç noktası, harita üzerinde uygun (engel ya da sandık içermeyen) karelerden rastgele olarak belirlenmelidir. Bitiş noktası ise, geliştirilen algoritmaya göre toplanan en son hazine sandığının konumu olmalıdır. Bu yaklaşım, oyunun dinamizmini artırarak karakterin başlangıç ve bitiş noktalarını her oyun için farklılaştıracaktır.
- Karakterin görme yeteneği 3x3 bir alanda sınırlı olup, hareketi otomatik olarak gerçekleşmelidir. Program başlatıldığında, karakter, tüm sandıkları toplamak için hesaplanan en kısa yol üzerinde ilerlemelidir. Karakter başlangıçta tüm haritayı bilmemelidir ve öğrenmediği kareler sisli olarak gösterilmelidir. Program sonlandığında, karakterin öğrenmediği kareler sisli olarak gösterilmeli ve öğrenilen kareler açılarak ilgili nesneye göre (engel, sandık, vb.) belirtilmelidir. Öğrenilen karelerin içeriği, öğrenilme sırasına göre yukarıdan

aşağıya doğru ekranın sağ üst köşesinde yazdırılmalıdır. Bu yöntem, karakterin keşfettiği bölgeleri ve karşılaştığı nesneleri oyuncuya sağlıklı bir şekilde iletecektir.

- Tasarlanan algoritma sonucunda elde edilen en kısa yol, harita üzerinde yeşil renkle gösterilmeli ve bu yolun kaç kare üzerinden geçtiği bilgisi ekranda yazdırılmalıdır. Yeşil renk, karakterin izlediği en kısa yolu vurgulayarak oyuncuya net bir yol gösterimi sağlayacaktır. Ayrıca, kaç kare üzerinden geçildiği bilgisi, karakterin yolculuğunun zorluğunu anlamak için önemli bir ölçüdür. Bu bilgi, oyuncunun karakterin hareketlerini daha iyi anlamasına yardımcı olacaktır.
- istenen sınıflar :
- Karakter Sınıfı:
  Bu sınıfta karakterlerin temel özelliklerini ve fonksiyonlarını tanımlıyoruz. Her karakterin bir ID ve Ad bilgisi bulunmaktadır. Ayrıca karakterin ilerlediği koordinatları Lokasyon değişkeni ile takip ediyoruz. Karakterler sadece sağ, sol, yukarı ya da aşağı yönde hareket edebilirler. Bu sınıfta şu metodlar bulunmaktadır:
  Constructor: Karakterin ID, Ad ve başlangıç lokasyonu ile oluşturulmasını sağlar. Get ve Set Metotları: ID, Ad ve lokasyon bilgilerine erişim ve değişiklik yapmamızı sağlar. En Kısa Yol Metodu: Karakterin hedefe en kısa yoldan ulaşmasını sağlayan algoritmayı içerir.
- Lokasyon Sınıfı:
  Bu sınıf, ızgara üzerindeki konumları temsil eder. Her Lokasyon nesnesi, x ve y koordinatlarını içerir. Sınıf şu metodları içerir:
  Constructor: Belirli bir x ve y koordinatı ile bir Lokasyon nesnesi oluşturur. Get ve Set Metotları: x ve y koordinatlarına erişim ve değişiklik yapmamızı sağlar.
- Engel Sınıfı:
  Bu sınıf, oyunda bulunan engellerin genel özelliklerini tanımlar. Sabit ve hareketli olmak üzere iki alt sınıfa sahiptir. Engel sınıfının alt sınıfları şunlardır:
  HareketsizEngel: Ağaçlar, kayalar, duvarlar ve dağlar gibi sabit engelleri temsil eder. DinamikEngel: Kuşlar ve arılar gibi hareketli engelleri temsil eder. Her engelin özellikleri ve davranışları sınıfın alt sınıflarında ayrıntılı olarak tanımlanmıştır.
- Uygulama Sınıfı:
  Bu sınıf, oyunun işleyişini kontrol eder ve oyuncunun karakterin hedefe kaç adımda ulaştığı, hangi nesneleri elde ettiği gibi bilgileri saklar ve ekranda gösterir.

II. YÖNTEM

Bu bölümde oyunun kontrol mekanizmaları ve oyun akışı detaylı bir şekilde açıklanmaktadır.

Oyun Kontrolleri ve Oyun Akışı Açıklaması Bu bölümde oyunun kontrol mekanizmaları ve oyun akışı detaylı bir şekilde açıklanmaktadır.

Oyun Alanı ve Engeller Oyun alanı, bir ızgara şeklinde oluşturulmuştur. Bu ızgara, karakterin hareket edebileceği bölgeyi sınırlar. Oyun alanında çeşitli engeller bulunur. Engeller, karakterin yolunu keserek ilerlemesini zorlaştırır veya engelleri aşmak için belirli stratejiler gerektirir.

Hareketsiz Engeller: Bu engeller, karakterin üzerinden geçemediği sabit engellerdir. Örneğin, ağaçlar ve kayalar bu kategoriye girer.

Hareketli Engeller: Bu engeller, oyun sırasında belirli bir desende hareket eden engellerdir. Örneğin, arılar ve kuşlar bu kategoriye girer.

Skorlama ve Kazanma Durumu Oyuncu, oyun alanındaki belirli öğeleri toplayarak puan kazanır. Örneğin, altınları toplamak oyuncuya puan kazandırır. Oyuncunun amacı, oyun alanındaki tüm hedefleri tamamlayarak maksimum puanı elde etmektir. Tüm hedefler tamamlandığında, oyuncu oyunu kazanır.

Oyun Sonu ve Skor Gösterimi Oyun, belirli bir koşul gerçekleştiğinde veya oyun alanındaki tüm hedefler tamamlandığında sona erer. Oyun sona erdiğinde, oyuncunun elde ettiği puan ve oyun sonu mesajı ekranda gösterilir.

III. SONUÇLAR

Projede bulunan Methodlar: Metod Adı: İzgaraCiz

Amaç: Formun arka planına ızgara deseni ekler.

Parametreler:

satir: Oluşturulacak ızgaranın satır sayısı. sutun: Oluşturulacak ızgaranın sütun sayısı. Açıklama: Bu metod, belirtilen satır ve sütun sayısına göre bir ızgara deseni oluşturur ve formun arka planına bu deseni ekler.

Metod Adı: GridOlustur

Amaç: Belirtilen boyutlarda bir ızgara deseni oluşturur.

Parametreler:

satir: Oluşturulacak ızgaranın satır sayısı. sutun: Oluşturulacak ızgaranın sütun sayısı. genislik: Oluşturulacak ızgaranın genişliği. yukseklik: Oluşturulacak ızgaranın yüksekliği. Açıklama: Bu metod, belirtilen boyutlarda bir ızgara deseni oluşturur ve bu ızgarayı bir Bitmap olarak döndürür. İzgara deseni, her bir hücre arasında belirli bir dolgu ve kenarlık ile çizilmiş karelerden oluşur.

Metod Adı: Oyunn

Amaç: Oyunun ana işlevlerini gerçekleştiren metod.

Açıklama: Bu metod, oyunun ana oynanış mekaniklerini yönetir. Oyun içinde karakterin hareketini, altınları toplamayı, engellerden kaçınmayı ve oyunun sonunu kontrol eder. Skor Güncelleme:

Skor etiketini, mevcut puanı göstermek için günceller. Altın Konumlarını Belirleme:

Oyundaki görünür altınların merkez koordinatlarını bir listeye ekler. Hedef Belirleme:

Oyuncunun en yakın altına gitmesi gereken konumu belirler. Hareket Yönünün Belirlenmesi:

Oyuncunun hareket yönünü belirler: sol, sağ, yukarı veya aşağı. Karakter Hareketi:

Oyuncunun hareket yönüne göre karakteri hareket ettirir ve karakterin görüntüsünü günceller. Engellerden Kaçınma:

Karakterin engellerle çarpışmasını kontrol eder. Eğer bir çarpışma varsa, karakterin hareketini durdurur ve yeni hareket yönünü belirler. Altınları Toplama:

Karakterin altınlarla çarpışmasını kontrol eder. Eğer bir çarpışma varsa, puanı artırır ve altını görünmez yapar. Kalan Altın Sayısını Kontrol Etme:

Oyundaki kalan altın sayısını kontrol eder. Eğer hiç altın kalmamışsa, oyunu sonlandırır ve ”Kazandınız!” mesajını gösterir.

Metod Adı: AgacHareketsizEngelEkle

Amaç: Belirtilen boyuttaki oyun alanına hareketsiz ağaç engelleri eklemek için kullanılır. Engeller rastgele konumlara yerleştirilir.

Parametreler:

satiirr: Oyun alanının satır sayısı. sutuunn: Oyun alanının sütun sayısı. Gerçekleştirilen İşlemler:

Rasgele sayı üretmek için bir Random nesnesi oluşturulur. Kare hücresi boyutu, pencere boyutuna ve satır/sütun sayısına bağlı olarak belirlenir. Hücreler arasındaki dolgu miktarı hesaplanır. Daha önce işgal edilen konumları takip etmek için bir HashSet oluşturulur. Alanın maksimum X ve Y koordinatları belirlenir. Belirlenen aralıkta rastgele bir sayıda hareketsiz ağaç engeli oluşturulur. Oluşturulan engeller rastgele konumlara yerleştirilir. Engeller PictureBox formunun kontrollerine eklenir. Notlar:

Oluşturulan engeller, resimlerinin boyutuna göre uygun bir şekilde boyutlandırılır ve form üzerinde yerleştirilir. Engeller, çakışma kontrolü yapılarak birbirlerinin üzerine gelmemesi sağlanır. Engeller, ”hareketliEngel” etiketiyle işaretlenir ve form üzerinde görünür hale getirilir.

Metod Adı: KayaHareketsizEngelEkle

Amaç: Belirtilen boyuttaki oyun alanına hareketsiz kaya engelleri eklemek için kullanılır. Engeller rastgele konumlara yerleştirilir.

Parametreler:

satiirr: Oyun alanının satır sayısı. sutuunn: Oyun alanının sütun sayısı. Gerçekleştirilen İşlemler:

Rasgele sayı üretmek için bir Random nesnesi oluşturulur. Kare hücresi boyutu, pencere boyutuna ve satır/sütun sayısına bağlı olarak belirlenir. Hücreler arasındaki dolgu miktarı hesaplanır. Daha önce işgal edilen konumları takip etmek için bir HashSet oluşturulur. Alanın maksimum X ve Y koordinatları belirlenir. Belirlenen aralıkta rastgele bir sayıda kaya oluşturulur. Oluşturulan kayalar rastgele konumlara yerleştirilir. Kayalar PictureBox formunun kontrollerine eklenir. Notlar:

Oluşturulan kayalar, resimlerinin boyutuna göre uygun bir şekilde boyutlandırılır ve form üzerinde yerleştirilir. Kayalar, çakışma kontrolü yapılarak birbirlerinin üzerine gelmemesi sağlanır. Kayalar, ”hareketliEngel” etiketiyle işaretlenir ve form üzerinde görünür hale getirilir.

Metod Adı: YatayHareketliAriEkle

Amaç: Belirtilen aralıkta rastgele sayıda yatay hareketli arılar eklemek için kullanılır. Her arı için bir PictureBox nesnesi oluşturulur, rastgele konumlara yerleştirilir ve yatay hareket ettirilir.

Gerçekleştirilen İşlemler:

Rastgele sayı üretmek için bir Random nesnesi oluşturulur. Belirtilen aralıkta rastgele bir sayıda hareketli arı oluşturulur. Oluşturulan arılar için döngü başlatılır. Her arı için bir PictureBox nesnesi oluşturulur ve özellikleri atanır. PictureBox'in konumu belirlenir, rastgele bir noktaya yerleştirilir. PictureBox'in etiketi ”hareketliEngel” olarak atanır. PictureBox formunun kontrollerine eklenir. Her arı için bir zamanlayıcı oluşturulur ve arının yatay hareketi bu zamanlayıcı ile kontrol edilir. Notlar:

Oluşturulan arılar, resimlerinin boyutuna göre uygun bir şekilde boyutlandırılır ve form üzerinde rastgele konumlara yerleştirilir. Her arı için ayrı bir zamanlayıcı oluşturulur ve her zamanlayıcı arının yatay hareketini kontrol eder. Arılar, her zamanlayıcı tetiklendiğinde YatayHareketEt metodunu çağırarak yatay olarak hareket ederler.

Metod Adı: DikeyHareketliKusEkle

Amaç: Belirtilen aralıkta rastgele sayıda dikey hareketli kuşlar eklemek için kullanılır. Her kuş için bir PictureBox nesnesi oluşturulur, rastgele konumlara yerleştirilir ve dikey hareket ettirilir.

Gerçekleştirilen İşlemler:

Rastgele sayı üretmek için bir Random nesnesi oluşturulur. Belirtilen aralıkta rastgele bir sayıda hareketli kuş oluşturulur. Oluşturulan kuşlar için döngü başlatılır. Her kuş için bir PictureBox nesnesi oluşturulur ve özellikleri atanır. PictureBox'in konumu belirlenir, rastgele bir noktaya yerleştirilir. PictureBox'in etiketi ”hareketliEngel” olarak atanır. PictureBox formunun kontrollerine eklenir. Her kuş için bir zamanlayıcı oluşturulur ve kuşun dikey hareketi bu zamanlayıcı ile kontrol edilir. Notlar:

Oluşturulan kuşlar, resimlerinin boyutuna göre uygun bir şekilde boyutlandırılır ve form üzerinde rastgele konumlara yerleştirilir. Her kuş için ayrı bir zamanlayıcı oluşturulur ve her zamanlayıcı kuşun dikey hareketini kontrol eder. Kuşlar, her zamanlayıcı tetiklendiğinde DikeyHareketEt metodunu çağırarak dikey olarak hareket ederler.

Metot Adı: DikeyHareketEt

Amaç: Verilen PictureBox nesnesinin dikey hareketini gerçekleştirir. Hareket yönüne göre PictureBox'in Tag özelliği değiştirilir ve PictureBox'in dikey konumu güncellenir.

Parametreler:

hareketliEngel: Dikey hareket ettirilecek PictureBox nesnesi. Gerçekleştirilen İşlemler:

PictureBox'in mevcut konumu alınır. Hareket mesafesi belirlenir. Hareket yönüne göre PictureBox'in Tag özelliği değiştirilir. Dikey konum güncellenir ve belirli bir aralıkta sınırlanır. PictureBox'in yeni konumu atanır.

Metot Adı: YatayHareketEt

Amaç: Verilen PictureBox nesnesinin yatay hareketini gerçekleştirir. Hareket yönüne göre PictureBox'in Tag özelliği değiştirilir ve PictureBox'in yatay konumu güncellenir.

Parametreler:

hareketliEngel: Yatay hareket ettirilecek PictureBox nesnesi. Gerçekleştirilen İşlemler:

PictureBox'in mevcut konumu alınır. Hareket mesafesi belirlenir. Hareket yönüne göre PictureBox'in Tag özelliği değiştirilir. Yatay konum güncellenir ve belirli bir aralıkta sınırlanır. PictureBox'in yeni konumu atanır.

Metot Adı: AltinlariEkle

Amaç: Belirtilen boyuttaki bir alana rastgele altınlar ekler. Altınların konumları, daha önce işgal edilmemiş rastgele noktalara yerleştirilir.

Parametreler:

satiirr: Oluşturulan alanın satır sayısı. sutuunn: Oluşturulan alanın sütun sayısı. Gerçekleştirilen İşlemler:

Rasgele sayı üretmek için bir Random nesnesi oluşturulur. Kare hücresi boyutu, pencere boyutuna ve satır/sütun sayısına bağlı olarak belirlenir. Hücreler arasındaki dolgu miktarı hesaplanır. Daha önce işgal edilen konumları takip etmek için bir HashSet oluşturulur. Alanın maksimum X ve Y koordinatları belirlenir. Belirlenen aralıkta rastgele bir sayıda altın oluşturulur. Her bir altın için bir PictureBox nesnesi oluşturulur ve özellikleri atanır. Altının konumu belirlenir ve işgal edilen konumlar listesine eklenir. PictureBox formun kontrollerine eklenir.

Metot Adı: OyunuSifirlaa

Amaç: Oyunu sıfırlar ve başlangıç durumuna getirir. Skor sıfırlanır, karakterin başlangıç konumu ve hızı ayarlanır, tüm oyunelementleri görünür hale getirilir ve zamanlayıcı başlatılır.

Gerçekleştirilen İşlemler:

Skor metni sıfırlanır ve puann değişkeni sıfırlanır. Karakter hızı sıfırlanır. Karakterin başlangıç konumu ayarlanır (sol üst köşe). Formdaki tüm PictureBox kontrolü olan nesneler görünür hale getirilir. Zamanlayıcı başlatılır. Metot Adı: oyunSonu

Amaç: Oyun sonu durumunu işler. Zamanlayıcı durdurulur ve skor metni ile oyun sonu mesajı gösterilir.

Parametreler:

mesaj: Oyun sonu mesajı. Gerçekleştirilen İşlemler:

Zamanlayıcı durdurulur. Skor metni ve oyun sonu mesajı gösterilir.

Diğer Form da ise Kullanıcıdan alınacak verilere göre oluşturulacak bir ızgara ekranı bulunmaktadır.

REFERENCES

[1] https://tr.pngtree.com/
[2] https://github.com/grgttdln/IT111LPixelMazeGame?tab=readme-ov-file
[3] https://www.youtube.com/watch?v=jaN4jhR8eMg
[4] https://www.youtube.com/watch?v=GC-nBgi9r0U