Kocaeli Üniversitesi

Bilgisayar Mühendisliği

Programlama Lab-I

Strateji Oyunu Savaş Mekanikleri

Yunus Emre AYIKER – 230202082

Enes ÜLKÜ - 230202083

I. ÖZET

Bu proje c dilinde yazılmıştır. Oyunun amacı iki farklı ırkın (İnsan İmparatorluğu ve Ork Lejyonu) birimleri, kahramanları ve canavarları ile birbirleriyle savaştığı belli bir senaryoya göre strateji oyunu simülasyonu yapmaktadır. Json dosyalarından veriler okunarak saldırı ve savunma hesaplanır. 20x20 ızgara üzerinde insan İmparatorluğu ve Ork Lejyonunun birimleri, kahramanları ve canavarları görüntüde karşılıklı yerleştirilir. Görüntüde savaşın başı ve sonu gösterilmiştir. Hesaplamaların sonucunda kazanan birimlerin sağlığı bar ile gösterilir ve birim sayıları, birim resimlerinin üzerinde bulunur. Kimin kazandığı ilan edilerek simülasyon biter. Her adım detaylı olarak dosyaya kaydedilerek savaşın seyrini görebiliriz.

II. GİRİŞ

Birimlerin saldırı, savunma, sağlık, kritik şansı; araştırmaların seviyelerine göre hangi birime ne türde ve ne kadar artış yaptığı; kahramanların hangi birime hangi türde ve ne kadar bonus verdiği, canavarların da etkilediği tür ve değerleri bütün bilgiler json dosyası içerisinde Files klasöründe yüklüdür. Senaryo için istenilen link indirilerek veriler çekilir ve senaryo.json'a kaydedilir. Senaryo ve orduların bilgileri, json dosyalarındaki manuel okuma işlemi ile ayrıştırılıp ilgili değişkenlere atama yapılır ve savaş mekaniğinde kullanılır.

Bir ordunun birimlerinde saldırı, savunma, sağlık ve kritik şans bilgileri bulunur. Saldırı ve savunma, o birimin ne kadar güçte saldırı ya da savunma yapacağının sayısal değeridir ve diğer etkenlere göre artıp azalabilir. Sağlık durumu, her bir birimin ne kadar hasar aldığı ve savaşa ne kadar devam edebileceğini belirler. Sağlık puanı düştükçe, birimler güç kaybeder ve yok olabilir. Kritik şans yüzdesi, kaç adımda bir saldırı gücünün 1.5 katına artacağının değeridir. Birimlerin bu değerleri kahraman, canavar ve araştırmaların etkileri ile artıp azalabilir.

Kahramanlar ve canavarlar, dost birimlere bonus sağlayabilir veya düşman birimlerine dezavantajlar uygulayabilir. Bu etkiler, net hasarı artırabilir veya savunma gücünü düşürebilir. Savaşın seyrini değiştirebilir. Kahraman, canavar ve araştırma etkileri her adımda vardır. Canavar ve kahramanın bonusu belli birime, belli türde (saldırı savunma kritik şans) ve belirli değerde bonus uygular.

Örneğin bir birimin savunma gücü %20 artar. Araştırmalar; saldırı, savunma veya kritik vuruş

oranlarını artırabilir. Araştırma seviyesi ilerledikçe, birimlerin performansı artar.

Birimler üzerinde yorgunluk da vardır. Savaş uzadıkça, birimlerin yorgunluk nedeniyle performansları azalır. Her 5 turda bir yorgunluk, saldırı ve savunma gücünde %10 azalmaya neden olur.

Bu hesaplamalara göre birimlerin saldırı, savunma ve kritik şans oranları güncellenir. Ordunun toplam saldırı/savunma gücü, her bir birim türünün saldırı/savunma gücü ile birim sayısı çarpılarak hesaplanır. Her birim türü için ayrı ayrı toplam saldırı/savunma gücü hesaplanır ve bu değerler toplanarak toplam saldırı/savunma gücü bulunur.

Adım adım hesaplama ile savaş mekaniğinde ilk adımda İnsan ordusu saldırır, Ork ordusu savunur. Diğer adımda Ork ordusu saldırır ve İnsan ordusu savunur. Bir ordunun birimleri tamamen yok olana kadar adımlar böyle devam eder. Net hasar; saldıran tarafın saldırı gücünden, savunan tarafın savunma gücü çıkartılarak bulunur. Herhangi bir adımda ordu, saldırıda üstünlük sağlayamaz ise (yani net hasar negatif olursa) savunan taraf üzerinde sağlık ve birim sayısı azaltması yapılamaz. Eğer saldırı gerçekleşirse net hasar, savunan tarafın toplam savunma gücü üzerinden orantılı olarak her birlik türüne dağıtılır. Her bir birim türü, ordunun toplam savunmasına yaptığı katkıya göre hasar alır. Hasar dağılımı = Net hasar x (Birim türü savunma gücü / Toplam savunma gücü) şeklinde hesaplanır. Yani herhangi bir birim toplam savunmaya ne kadar çok katkı sağlarsa o kadar çok hasar alır. Birim kayıplarının hesaplanması; her birlik türünün aldığı hasarın o birim türünün sağlık puanına bölünerek kaç birimin öldüğü hesaplanır. Birim Kaybı = Hasar / Birim Türü Sağlık Puanı şeklinde hesaplanır. Aslında sağlık puanının önemi buradadır. Sağlık fazla ise daha az birim kaybı oluşur. Eğer birim kaybı sonucunda bir birlik türü tamamen yok olursa o birlik türüne yapılan hasar, diğer birimlere yeniden orantılı olarak dağıtılır. Saldırı gerçekleşirse savunan tarafın sağlığında da güncelleme olur. Yeni sağlık değeri; eski sağlıktan, net hasarın birim sayısına bölünmesinden çıkarılarak bulunur. (Sağlık Puanı = Eski Sağlık Puanı - Net Hasar / Birim Sayısı) şeklinde hesaplanır. Sağlık, birim sayısından önce sıfırlanabilir. O zaman da birimlerin öldüğünü kabul ederiz. Bunların sonucunda saldırı gerçekleşmiş olur ve diğer adıma geçilir. Tüm birimlerini kaybeden taraf, savaşı kaybeder.

Görüntü; savaşın başında kahraman, canavar ve birimlerin sayılarına göre ordular karşılıklı olacak şekilde ızgaraya yerleştirilir. Orduların ızgara içerisinde görselleri mevcuttur. Birimlerin ne kadar sayıda olduğu yazar ve sağlığının yüzdesi barda gösterilir. Kazanan tarafın kalan birliklerinin sağlık ve birim sayıları gösterilerek simülasyon biter.

III. YÖNTEM

Öncelikle İnsan ve Ork orduları için struct oluşturduk. Bunların içinde birimlerin, kahramanların, canavarların ve araştırmaların alt structı bulunur. Birimler structında, birimlerin özelliklerini tutan kahraman, canavar ve araştırma için ayrı ayrı alt structlar oluşturuldu. Böylece zincerleme bir sistem kuruldu. Senaryo okunmadan önce bütün değişkenler memset fonksiyonu ile sıfırlandı. Her yapılan işlem için fonksiyonlara bölündü ve projenin okunabilirliği arttırıldı.

Senaryo indirme işlemi için cURL kütüphanesini kullanıldı. Dinamik şekilde program her çalıştırıldığında istenilen 1-10 arasındaki numaralanmış senaryo indirilir. İndirilecek dosya dizinini kontrol amacıyla getcwd fonksiyonu kullanılarak main.c nin olduğu dizin alındı. Alınan dizine Files klasörü eklenerek json dosyalarının konumuna erişildi. Access fonksiyonu kullanılarak da bu konumdaki json dosyalarının varlığı ve isimleri kontrol edildi.

Kontrolden geçen senaryo dosyası, fgets fonksiyonu ile while içerisinde satır satır manuel okunmuştur. Okunan satır içerisinde strstr fonksiyonu ile istenilen birim, kahraman, canavar ve araştırmalar ilgili ordunun struct içindeki

değişkenlerine atanmıştır ve savaş için taraflar hazır hale getirilmiştir.

Savaştan önce tarafların durumu; birim türleri ve sayıları, varsa kahraman ve canavarlar ekranda SDL ve TTF kütüphaneleri başlatılarak grafik arayüz ve yazı tiplerinin kullanılması ile göterilmiştir. Kullanılan yazı tipi Roboto-BlackItalic.ttf dosyasından yüklenmiştir. Tam ekran modunda bir pencere (SDL_Window) ve bir renderer (SDL_Renderer) oluşturulmuştur. Ayrıca, ekranın çözünürlüğü otomatik olarak alınarak dinamik pencere boyutlandırması yapılmıştır. Savaş alanındaki birimler ve arka plan için BMP formatındaki görüntüler SDL'de yüklenip, renderer üzerinden ekrana çizilmiştir. Bu görüntüler, insan imparatorluğu ve ork lejyonuna ait askerler, kahramanlar ve canavarlar için belirli dizinlerde saklanmıştır.

20x20 boyutlarında bir ızgara ekranın ortasında çizilmiştir. Izgara, her iki tarafın birimlerinin, kahramanlarının ve canavarlarının belirli konumlarda yerleştirilmesini sağlar. GRID_SIZE ve TILE_SIZE gibi sabitler, ekranın çözünürlüğüne bağlı olarak dinamik şekilde ayarlanmıştır. Izgara üzerinde birimler, her karoda en fazla 100 tane olacak şekilde gösterilmiştir ve birim sayıları karo üzerine yazılmıştır. Her karodaki birimin sağlık yüzdesi sağlık barı ile gösterilmiştir. Olay döngüsü içinde SDL_PollEvent() fonksiyonu ile kullanıcı etkileşimleri yakalanmıştır. Kullanıcının sağ ok tuşuna basmasıyla simülasyon ilerlerken, ESC tuşuna basıldığında oyun döngüsü sonlandırılmıştır. Birimler ekrana yansıtıldıktan sonra savaş süreci savas() fonksiyonu çağrılarak simülasyon ilerletilmiştir. Böylelikle savaş öncesi tarafların durumları grafik ekranında gösterilmiştir. Savaş sonunda sağ kalan birimler yine aynı yöntem ile grafik ekranında gösterilmiştir.

Senaryo okunmadan önce tüm struct lardaki verilerin memset ile sıfırlanması sayesinde bir birimin savaşta olup olmadığı birim sayısı ile kontrol edilebildi. Sağlıktan ya da birim kaybından dolayı birim ölürse, o birimin sayısı sıfırlandı. Mekanik hesaplamasında koşullar; birim sayısının pozitif olma durumuna göre savaşa dahil edilip hesaplamaları yapılmıştır.

Savaş mekaniğinde hesaplamalar yapılırken dosyaya kaydetme işlemleri yapıldı. Mekanikte tek olan adımlarda İnsan saldırısı, çift adımlarda Ork saldırısı olacak şekilde koşul ile ayrıldı. Saldırı gerçekleşmeden önce kahraman, canavar, araştırma etkileri her adımda olacak şekilde hesaplandı. Eğer adım 5 ve katları ise yorgunluk uygulandı. Örnek olarak İnsan saldırı adımındayız.

Örnek olarak savaşın başında ilgili özelliklere kahraman, canavar ve araştırma etkileri; birimin özelliklerinde güncelleme yaptı. Birimin saldırı gücü 30 iken 50 oldu gibi. Sonra İnsan saldırı adımında olduğumuzu düşünelim. Toplam saldırıyı bulamak için birimin sayısı ile güncel saldırı gücünü çarptık. Eğer kritik şansı varsa 1.5 katını aldık ve bunu savaşta olan her birim için yaparak üst üste topladık. Ork leginin de toplam savunmasını bulurken birim sayısı ile son savunma gücünü çarparak her birim için bunu yaparak üst üste topladık. İnsan saldırıdan Ork savunmasını çıkartarak net hasarı bulduk. Net hasar negatif ise saldırı gerçekleşmemiş olacak orkdan eksilme hesapları olmaycak. Saldırı gerçekleşirse Ork leginin birimleri orantılı hasar dağılımı alırlar. Orantılı Hasar Dağılımı, savaş sırasında net hasarın, savunan ordunun her birim türüne savunma katkısına göre dağıtılmasını sağlar. Bu yöntem, bir ordunun daha güçlü savunma birimlerinin daha fazla hasar almasına neden olur. Net hasar hesaplandıktan sonra, bu hasar her birlik türüne, savunma gücüne göre orantılı olarak dağıtılır. Sonra birimin aldığı hasara göre birim kayıpları gerçekleştirilir. Birim Kaybı = Hasar / Birim Türü Sağlık Puanı formülü ile hesaplanır. Herhangi birim, birim kaybından tamamen yok olurlarsa aldığı hasarı diğer birimlere orantılı şekilde üstüne ekleyerek dağıtır.

Dikkat ettiğimiz bug oluşturabilecek olaylar şunlardır: Kritik şans kahraman araştırma gibi etkenlerle artabilir. %100 den fazla olmamalı çünkü %200 adım olsa yarım adımda bir olur

demek ve bu da olamaz. Kritik yüzdesinden adım sayısını bulurken payda 0 oluyor matematiksel hata yapmış oluruz. Çözüm olarak kritik şans değeri 100 üzerine çıkarsa maksimum 100 yaptık. Başka bir bug olabilecek durum da ölen birim olursa if koşulunda 0 ve altı ise hasar dağılımı yapar. Ancak savaşta olmayan birimin de birim sayısı 0 ve her adımda bu koşula girer. Bunu engellemek için kontrol değişkeni ile bir kere if içine girer ve öyle hasar dağılımı gerçekleşir.

IV. DENEYSEL SONUÇLAR

Program çalıştırıldığında kullanıcının karşısına 1 ile 10 arasında bir senaryo seçmesi isteyen bir ekran çıkar. Seçilen senaryoya göre kullanıcıya, tarafların savaştan önceki durumu; birim türleri, birim sayıları, varsa kahraman ve canavar gösterilir. Klavyeden sağ ok tuşuna basıldığında savaş yapılır ve savaş sonu kaybeden tarafın birimleri tamamen yok olur. ESC tuşuna basılırsa direk olarak simülasyondan çıkılır.

V. KATKILAR

Projeye dışardan kütüphane eklemeyi öğrenildi. cURL kütüphanesi ile link indirip dosyaya keydetmeyi öğrenildi. Görüntü kütüphanesindeki fonksiyonları araştırarak ızgara yapmayı, resim eklemeyi ve istenilen konuma resim yerleştirmeyi, görüntü ekranına yazı yazmayı öğrenildi.

VI. KAYNAKÇA

http://fredericgoset.ovh/sdl/en/installation.html

https://forums.codeblocks.org/index.php?topic=20203.0

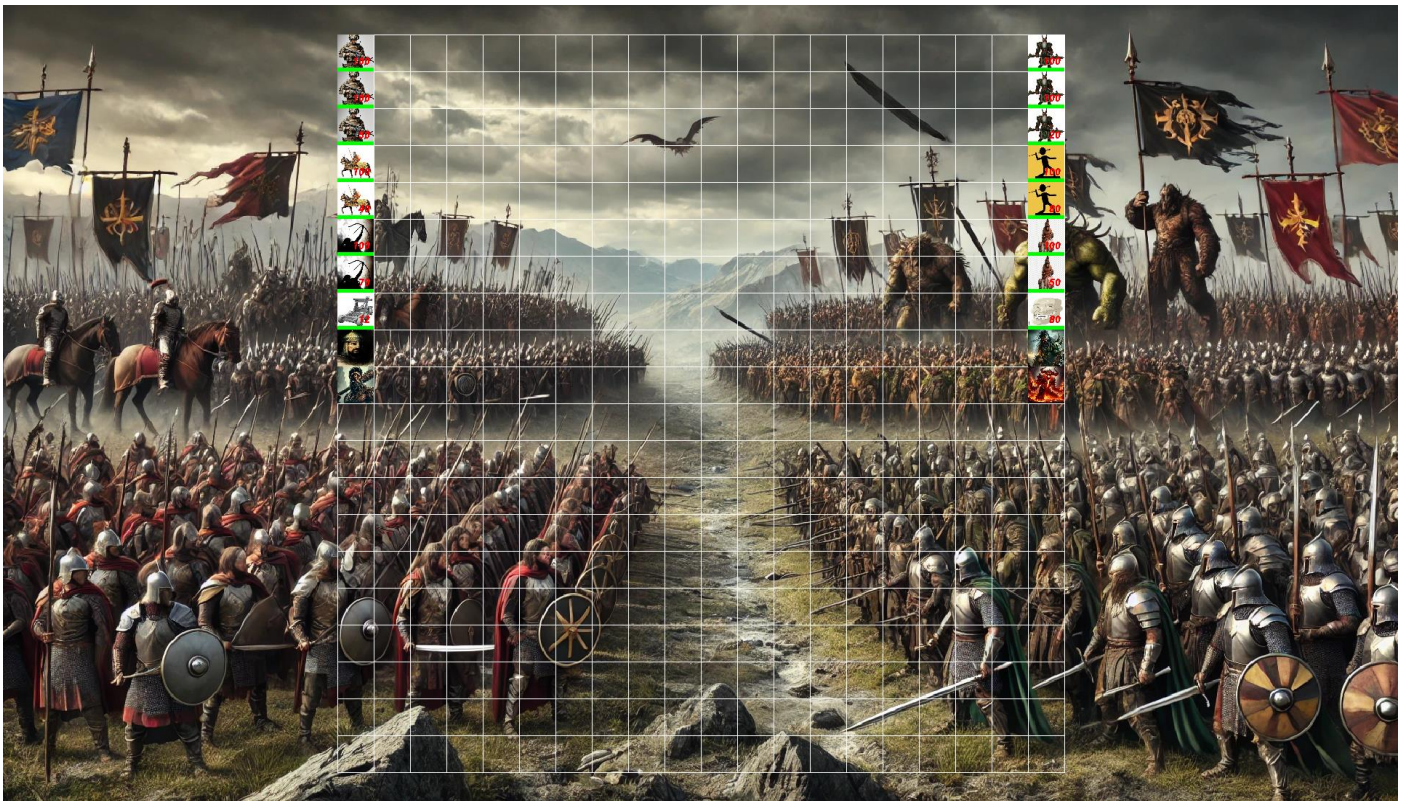
https://stackoverflow.com/questions/5822081/sdl-ttf-not-working-in-codeblocks

https://medium-com.translate.goog/@scarletskynned/file-i-o-unistd-h-80dd92b9f782

https://stackoverflow.com/questions/22886500/how-to-render-text-in-sdl2

VII. GÖRSELLER

Savaş başlangıç örneği.



Savaş sonu örneği.