PROGRAMLAMA LABORATUVARI
PROJE 1

Anıl Engin Keretli

Kocaeli Üniversitesi

Kocaeli/TÜRKİYE

230201128

Ahmet Burak Karkaç

Kocaeli Üniversitesi

Kocaeli/TÜRKİYE

220201173

I. ÖZET

Bu projede, iki farklı ırkın (İnsan İmparatorluğu ve Ork Lejyonu) birimlerinin birbirleriyle savaştığı bir strateji oyunu simülasyonu geliştirilmiştir. Simülasyon, JSON dosyalarından alınan birim özelliklerine göre birimlerin saldırı, savunma, sağlık ve yorgunluk gibi faktörler çerçevesinde bir savaş mekanizmasını simüle etmektedir. Kahramanlar ve canavarlar, birimlere stratejik avantajlar sağlayarak savaşın sonucunu etkileyen özel birimlerdir. Savaşın simülasyon aşamalarında her iki tarafın saldırı ve savunma güçleri detaylı hesaplanarak sonuçlar elde edilmiştir.

1. Giriş

Bu çalışmanın amacı, oyun geliştirme ve savaş simülasyonları için kullanılabilecek bir savaş mekaniklerini içeren bir model oluşturmaktır. Projede, İnsan İmparatorluğu ve Ork Lejyonu adlı iki taraf arasında geçen savaşlar, birimlerin belirli özelliklerine göre hesaplanarak gerçekleştirilmiştir. Savaş, birliklerin saldırı ve savunma gücünü, kahramanların sağladığı bonusları ve yorgunluk faktörlerini içeren kapsamlı bir simülasyon modeli üzerinden yürütülmektedir. Proje, stratejik oyunlarda savaş dinamiklerini anlamak ve geliştirmek amacıyla önemli bir örnek teşkil etmektedir.

2 . Yöntem

2.1 Veri Okuma ve İşleme

JSON formatında verilen unit_types.json, heroes.json, creatures.json, ve research.json dosyalarındaki veriler, proje kapsamında C dili kullanılarak manuel olarak okunmuş ve ayrıştırılmıştır. Dosyalar proje dizinindeki “Files” klasöründe bulunmaktadır. Program, JSON dosyalarındaki verilerin sırasını dikkate almadan her bir veriyi doğru anahtarla eşleştirerek işleyecek şekilde geliştirilmiştir.

Eksik veri durumunda ise varsayılan değerler atanarak veri eksikliğinden kaynaklanabilecek hata durumlarının önüne geçilmiştir.

```c
/* Kahraman dosyasını okuma ve degerleri ayrıstırma fonksiyonları */
int kahramanları_dosyadan_oku(const char* filename, char* buffer, size_t size) {
    FILE *heroes = fopen(filename, "r");
    if (heroes == NULL) {
        printf("Acilamayan dosya: %s!\n", filename);
        return 1;
    }
    size_t heroes_json_length = fread(buffer, sizeof(char), size - 1, heroes);
    buffer[heroes_json_length] = '\0';
    fclose(heroes);
    return 0;
}
```

File açma

```c
void kahraman_bonus_degeri(char* start, char* end, struct hero* Hero) {
    char* bonus_str = strstr(start, "bonus_degeri");
    if (bonus_str && bonus_str < end) {
        char* colon = strchr(bonus_str, ':') + 1;
        while (*colon == ' ' || *colon == '"') {
            colon++;
        }
        char* end_pointer = colon;
        while (*end_pointer != ',' && *end_pointer != '}' && *end_pointer != '"') {
            end_pointer++;
        }
        int uzunluk = end_pointer - colon;
        char* deger = malloc(uzunluk + 1);
        strncpy(deger, colon, uzunluk);
        deger[uzunluk] = '\0';
        Hero->etki[0] = atoi(deger);
        free(deger);
    }
}
```

Yukarıdaki kodda, kahraman bonus değerini işleyen kahraman_bonus_degeri adlı C dilinde yazılmış fonksiyonun detayları açıklanmaktadır. Fonksiyon, belirli bir karakter aralığında arama yaparak kahramanın bonus değerini belirlemekte ve ilgili yapıya aktarmaktadır. Fonksiyon, bir JSON formatından veri çekme ya da özel formatlı bir metin belgesinden veri okuma durumları için kullanılabilir.

İlk olarak bonus_degeri anahtar kelimesini strstr fonksiyonu ile start ve end aralığında arar. Eğer anahtar kelime bulunursa, anahtar kelimenin geçtiği yerden sonrasına bakmak için colon işaretçisi konumlandırılır:

": " sembolünden sonra gelen boşluk veya tırnak gibi gereksiz karakterleri atlayarak değerin başlangıcına gelir. Değer kısmının sonunda "}," boşluk veya tırnak işaretine gelene kadar end_pointer işaretçisi ilerletilir:

Libcurl Kütüphanesi Kullanımı

Bu projede, json dosyalarının içindeki verileri almak için libcurl kütüphanesi kullanılmıştır.

Libcurl, veri çekme ve aktarma işlemleri için geniş kapsamlı destek sunan bir C kütüphanesidir ve HTTP, HTTPS gibi birçok protokol ile çalışabilmektedir. Bu projede, libcurl ile harici bir JSON veri kaynağına bağlanarak sernaryo verilerini almakta ve bu veriler işlenmektedir.

Libcurl Kullanımı Adımları

Oturum Başlatma: curl_easy_init() fonksiyonu ile bir curl oturumu başlatılır.

URL Ayarı: curl_easy_setopt() fonksiyonu kullanılarak veri çekilecek URL belirlenir ve diğer gerekli parametreler ayarlanır.

Veri Alımı: curl_easy_perform() fonksiyonu çağrılarak belirlenen URL'den veri alınır ve bu veri, projenin kahraman_bonus_degeri gibi fonksiyonlarında işlenir.

Oturum Sonlandırma: İşlem tamamlandıktan sonra, curl_easy_cleanup() fonksiyonu ile oturum kapatılır ve kullanılan bellek serbest bırakılır.

```
if(curl_handle) {
    // Ýstenilen URL
    curl_easy_setopt(curl_handle, CURLOPT_URL, senaryo);

    // Veriyi callback ile almak için ayar
    curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, WriteCallback);

    // Veriyi kaydedeceðimiz buffer
    curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, scenario_string);

    // Ýsteði gönder
    res = curl_easy_perform(curl_handle);

    // Ýstek baþarýlý oldu mu?
    if(res != CURLE_OK) {
        fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
    } else {
        //printf("%s", scenario_string);  // Gelen veriyi ekrana yazdýr
    }
    // Temizlik
    curl_easy_cleanup(curl_handle);
}
```

Senaryoyu belirtilen sabit adresten çeker:
#define BASE_URL "https://
yapbenzet.org.tr/"

Savaş Mekanikleri

Savaş, birimlerin belirli özelliklerine göre saldırı ve savunma gücünün hesaplanmasıyla yürütülmektedir. Her birim, birim başına saldırı gücüne ve savunma gücüne sahip olup, birim sayısı ile bu değerler çarpılarak toplam saldırı ve savunma gücü elde edilmektedir. Kritik vuruş mekanizması ise birimlerin savaş performansını artıran bir faktör olarak %10 ve %20 gibi oranlarla modellenmiştir. Her savaş turunda, birimler karşılıklı olarak saldırmakta ve aldıkları hasara göre sağlık puanları düşmektedir.
Saldırı ve savunma hesaplama değerleri proje kapsamında verilmiş olup, istenilen senaryo dosyasından veriler okunarak, aşağıda prototipleri verilen fonksiyonlarda hesaplama yapılmıştır:

```
int saldiri_gucu_hesapla(
    struct birim Unit, int birim_index, struct creature Creatures[], struct hero Heros[],
    struct research Arastirma[][3], int hero_sayisi, int creature_sayisi, int tur);

int savunma_gucu_hesapla(
    struct birim Unit, int birim_index, struct creature Creatures[], struct hero Heros[],
    struct research Arastirma[][3], int hero_sayisi, int creature_sayisi, int tur);

void saldiri_etkisi_hesapla(int dusman_saldiri_gucu, int toplam_savunma_gucu, struct birim* Unit);
```

Bonuslar ve Yorgunluk Mekanizması

Kahramanlar ve canavarlar, belirli birim türlerine saldırı, savunma, kritik vuruş veya moral bonusu sağlayarak savaşın gidişatını önemli ölçüde etkilemektedir. Ayrıca her 5 turda bir devreye giren yorgunluk faktörü, birimlerin saldırı ve savunma gücünü %10 oranında azaltmaktadır. Bu etkenler, savaşın uzun sürdüğü durumlarda birimlerin performansını kademeli olarak düşürmektedir.

```
int yorgunluk_sayisi=tur/5;

for (int i=0;i<yorgunluk_sayisi;i++)
{
    saldiri_gucu=(float)saldiri_gucu*(float)(0.9);
}
```

Yukarıdaki kod parçası, kaç tam "yorgunluk" durumu (yani, kaç 5 tur geçmiş) olduğunu hesaplar. Örneğin, tur 10 ise, yorgunluk_sayisi 2 olur. Bu, 10. turda 2 yorgunluk durumu olduğu anlamına gelir.
for (int i = 0; i < yorgunluk_sayisi; i++):

Bu döngü, yorgunluk_sayisi kadar tekrarlanır. Her bir döngü, bir "yorgunluk" durumu temsil eder.
saldiri_gucu = (float)saldiri_gucu * (float)(0.9);:

Bu satır, mevcut saldiri_gucu değerini 0.9 ile çarparak azaltır. Yani, her yorgunluk durumu için saldırı gücü %10 azalır.

```
if (toplam_kritik_sans!=0 && tur%(100/(int)toplam_kritik_sans)==0)
{
    saldiri_gucu=((float)saldiri_gucu*(float)1.5);
}
```

toplam_kritik_sans: Bu, belirli bir birim ya da yaratığın kritik şansını temsil eder.

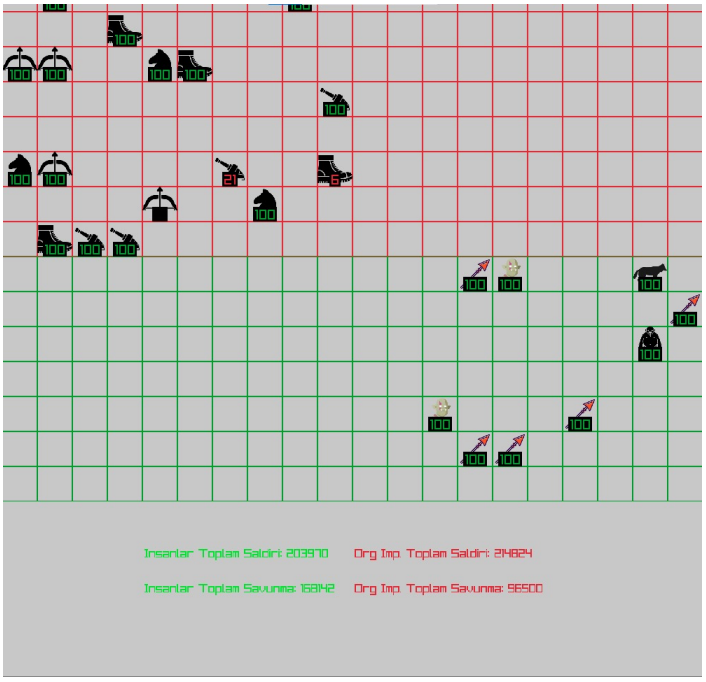
100 / (int)toplam_kritik_sans: Bu bölüm, toplam kritik şansın yüzde birimini hesaplar.

Örneğin, eğer toplam_kritik_sans 20 ise, bu ifade 100 / 20 ile 5 eder. Yani her 5 turda bir, kritik bir etkinin meydana gelme ihtimali var demektir.

tur % (...) == 0: Bu kısım, tur değişkeninin yukarıdaki hesaplama sonucuyla tam bölünüp bölünmediğini kontrol eder. Eğer tur bu değere tam bölünüyorsa, yani kalan 0 ise, kritik bir etki meydana gelebilir.

SAVAŞ SİMÜLASYONU GÖRSELLEŞTİRME

Savaş sona erdiğinde, her iki ordunun kalan birlikleri tekrar ızgaralar üzerinde gösterilir. Savaşın sonunda, bir tarafın tüm birlikleri ölmüştür. Eğer bir taraf tamamen yok olmadıysa, savaş devam eder.



Görselleştirme için raylib kütühanesi kullanılmıştır.

Raylib Kütüphanesi Kullanımı

Bu projede simülasyonun görselleştirilmesi için raylib kütüphanesi kullanılmıştır. Raylib, C dili için geliştirilmiş, basit ve hızlı bir grafik kütüphanesidir. Özellikle 2D ve 3D oyun geliştirme için uygundur ve kolay öğrenilebilir yapısı sayesinde projeye esneklik katmıştır.

Raylib Kullanımı Adımları
Pencere ve Ekran Ayarları: InitWindow() ve SetTargetFPS() gibi fonksiyonlarla simülasyon için gereken pencere ayarları yapılmıştır.

Grafik ve Görsel Elemanlar: Kahramanlar, canavarlar ve diğer birimlerin görselleştirilmesi için raylib'in DrawText(), DrawRectangle() gibi fonksiyonları kullanılmıştır.

Etkileşim ve Kontrol: Kullanıcıdan gelen etkileşimler ve girişler IsKeyPressed() gibi raylib fonksiyonları ile kontrol edilmiştir.

Raylib, simülasyonun görsel yönünü basit ama etkili bir şekilde oluşturmayı sağlamış ve projeye kullanıcı dostu bir arayüz eklemiştir.

Aşağıdaki fonksiyon prototipi ve kod parçalarıyla görsel sağlanmıştır.

```
void randomYerlestir(struct birim Unit, int birim_index, struct Cell ekran[][26]);
void draw_cell(struct Cell ekran, int i, int j);

Texture2D birimGorsel[8];
Texture2D orkKahramanGorsel[4];
Texture2D insanYaratikGorsel[5];
Texture2D orkYaratikGorsel[6];
```

```
InitWindow(ekranEni,ekranBoy,"---Savas Similasyonu---");
SetTargetFPS(25);

birimGorsel[0]=LoadTexture("Icons\\piyadeler.png");
birimGorsel[1]=LoadTexture("Icons\\okcular.png");
birimGorsel[2]=LoadTexture("Icons\\suvariler.png");
birimGorsel[3]=LoadTexture("Icons\\kusatma_makineleri.png");
birimGorsel[4]=LoadTexture("Icons\\ork_dovusculeri.png");
birimGorsel[5]=LoadTexture("Icons\\mizrakcilar.png");
birimGorsel[6]=LoadTexture("Icons\\varg_binicileri.png");
birimGorsel[7]=LoadTexture("Icons\\troller.png");
```

Deneysel Sonuçlar

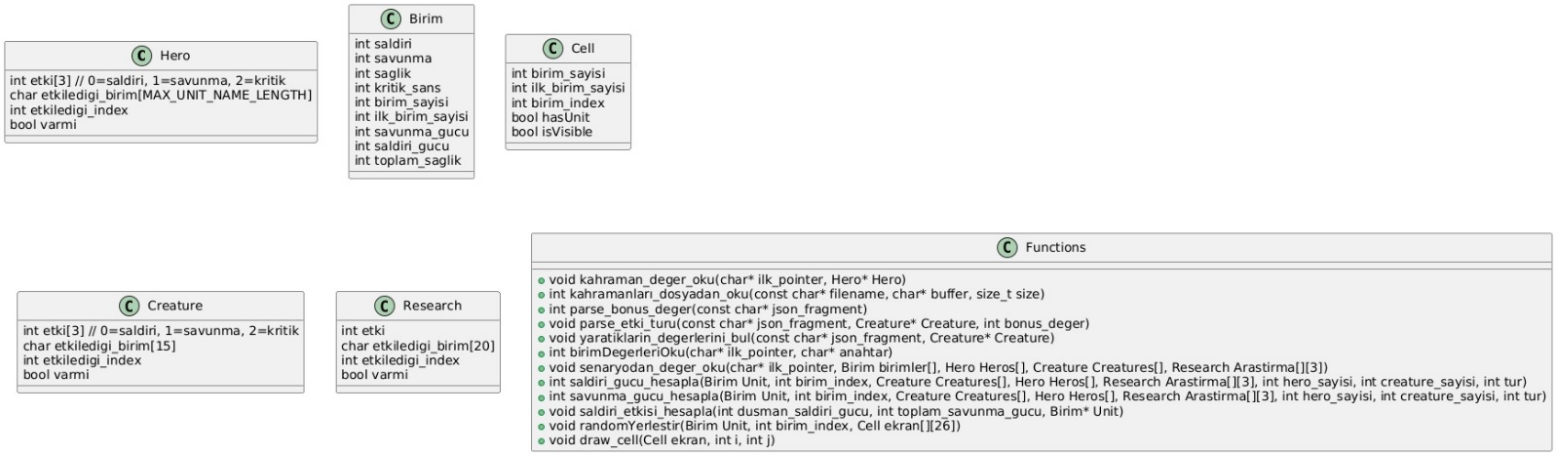
Simülasyonda her iki tarafın güçleri arasında yapılan karşılaştırmalar sonucunda,Verilen senaryoya göre savunma ve saldırı bonusları değiştiğinden İnsan İmparatorluğu'nun savunma bonusları sayesinde savaşın başlangıç aşamalarında avantaj sağladığı gözlemlenmiştir. Ork Lejyonu'nun saldırı gücünü artıran araştırma seviyeleri ise savaşın ilerleyen turlarında üstünlük kazanmasını sağlamıştır.

Sonuç

Bu proje, strateji oyunlarındaki savaş dinamiklerinin modellenmesi konusunda önemli bir örnek sunmaktadır. Birim özelliklerinin, savaş bonuslarının ve yorgunluk etkilerinin savaş sürecine katkısı incelenmiş ve belirli stratejik hamlelerin savaşın gidişatını nasıl etkilediği gözlemlenmiştir. Proje sonucunda, gerçek zamanlı strateji oyunlarının daha etkin ve dinamik bir şekilde modellenebilmesi için yorgunluk gibi faktörlerin önemine vurgu yapılmıştır.

Kaynakça

https://www.raylib.com/cheatsheet/cheatsheet.html
https://curl.se/libcurl/
https://wiki.codeblocks.org/index.php/Main_Page


Hero
int etki[3] // 0=saldiri, 1=savunma, 2=kritik
char etkiledigi_birim[MAX_UNIT_NAME_LENGTH]
int etkiledigi_index
bool varmi
Birim
int saldiri
int savunma
int saglik
int kritik_sans
int birim_sayisi
int ilk_birim_sayisi
int savunma_gucu
int saldiri_gucu
int toplam_saglik
Cell
int birim_sayisi
int ilk_birim_sayisi
int birim_index
bool hasUnit
bool isVisible
Creature
int etki[3] // 0=saldiri, 1=savunma, 2=kritik
char etkiledigi_birim[15]
int etkiledigi_index
bool varmi
Research
int etki
char etkiledigi_birim[20]
int etkiledigi_index
bool varmi
Functions
+ void kahraman_deger_oku(char* ilk_pointer, Hero* Hero)
+ int kahramanları_dosyadan_oku(const char* filename, char* buffer, size_t size)
+ int parse_bonus_deger(const char* json_fragment)
+ void parse_etki_turu(const char* json_fragment, Creature* Creature, int bonus_deger)
+ void yaratiklarin_degerlerini_bul(const char* json_fragment, Creature* Creature)
+ int birimDegerleriOku(char* ilk_pointer, char* anahtar)
+ void senaryodan_deger_oku(char* ilk_pointer, Birim birimler[], Hero Heros[], Creature Creatures[], Research Arastirma[][3])
+ int saldiri_gucu_hesapla(Birim Unit, int birim_index, Creature Creatures[], Hero Heros[], Research Arastirma[][3], int hero_sayisi, int creature_sayisi, int tur)
+ int savunma_gucu_hesapla(Birim Unit, int birim_index, Creature Creatures[], Hero Heros[], Research Arastirma[][3], int hero_sayisi, int creature_sayisi, int tur)
+ void saldiri_etkisi_hesapla(int dusman_saldiri_gucu, int toplam_savunma_gucu, Birim* Unit)
+ void randomYerlestir(Birim Unit, int birim_index, Cell ekran[][26])
+ void draw_cell(Cell ekran, int i, int j)