T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

LORDS OF THE POLYWARPHISM

ÖĞRENCİ ADI:
İbrahim Bener Karaca
https://github.com/BenerKaraca
Enes Küçük
https://github.com/EnessKucukk
ÖĞRENCİ NUMARASI:
220501019
220501017

DERS SORUMLUSU:
DR. ÖĞR. ÜYESİ Ercan Ölçer

22.03.2024

1 GİRİŞ

1.1 Projenin amacı

- Projede amaç, polymorphism kavramını kullanarak bir oyun geliştirmekti.
- Projede gerçekleştirilmesi beklenenlerin maddeler halinde yazılması

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

- Kullanıcı bir kare seçer ve ardından savaşçısını seçer
- Savaşçılar menzil alanlarındalarsa eğer birbirlerine hasar verir.
- Savaşçılarının canları biten oyuncu kaybeder.

3 TASARIM

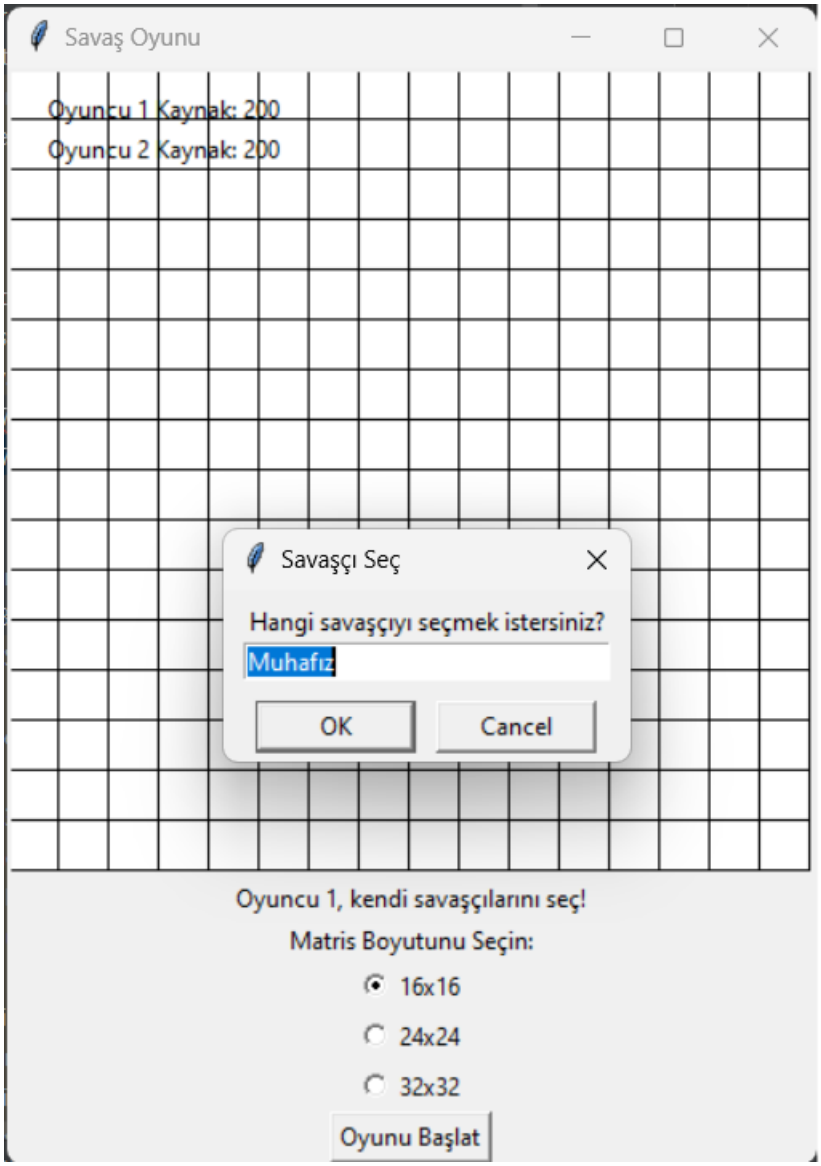
3.1 Kullanılacak teknolojiler

- Projeyi Python dilinde yazdık.
- Projede Tkinter ve Random kütüphanelerini kullandık.

3.2 Kullanıcı arayüzü tasarımı

- Arayüzü tasarlarken Tkinter modüllerinden faydalandık.

- “



4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

1. **Savaşçı Sınıfı (`Warrior`)**:
   - `Savaşçı` sınıfı, savaşçıların temel özelliklerini ve saldırı işlevini tanımlar.
   - `__init__` yöntemi, savaşçının adını, maliyetini, sağlığını ve saldırı menzilini tanımlar.
   - `saldır` yöntemi, bir düşmana saldırmak için kaynak kullanımını ve hasarı hesaplar.
2. **Alt Savaşçı Sınıfları**:
   - `Muhafız`, `Okçu`, `Topçu`, `Atlı` ve `Sağlıkçı` sınıfları, `Savaşçı` sınıfından türetilmiştir.
   - Her bir alt sınıf, ilgili savaşçı türünün özelliklerini ve maliyetini ayarlar.
3. **Dünya Sınıfı (`World`)**:
   - `Dünya` sınıfı, oyun dünyasını temsil eder.

- `__init__` yöntemi, dünya boyutunu ve diğer oyun özelliklerini tanımlar.
- `dünyayı_yazdır` yöntemi, dünyayı konsola yazdırır.
- `savaşçı_yerleştir` yöntemi, belirli bir pozisyona savaşçı yerleştirir.
- `savaşçı_al` yöntemi, belirli bir pozisyondaki savaşçıyı alır.
- `savaşçı_seç` yöntemi, seçilen savaşçıyı belirler.

4. **Oyuncu Sınıfı (`Player`)**:
   - `Oyuncu` sınıfı, oyuncuları temsil eder.
   - `__init__` yöntemi, oyuncunun adını tanımlar.
   - `savaşçı_ekle` yöntemi, oyuncunun savaşçı listesine bir savaşçı ekler.
5. **Oyun Uygulaması Sınıfı (`GameApp`)**:
   - `OyunUygulaması` sınıfı, tkinter tabanlı bir oyun uygulamasını oluşturur.
   - `__init__` yöntemi, tkinter arayüz öğelerini ve başlangıç ayarlarını tanımlar.
   - `dünyayı_çiz` yöntemi, oyun dünyasını çizer.
   - `oyunu_başlat` yöntemi, oyunu başlatır.
   - `tıklamayı_işle` yöntemi, fare tıklamasını işler.
   - `savaşçı_yerleştir` yöntemi, seçilen savaşçıyı belirli bir pozisyona yerleştirir.
   - `savaşçı_seç` yöntemi, oyuncunun savaşçı seçmesini sağlar.
   - `savaş_çatışmasını_işle` yöntemi, savaşçıların birbirlerine olan saldırılarını işler.
   - `saldır` yöntemi, bir savaşçının saldırısını gerçekleştirir.
6. **Ana Fonksiyon (`main`)**:
   - `main` fonksiyonu, tkinter uygulamasını başlatır.

4.2 Görev dağılımı

- Başta sınıfları tanımladık ve ardından sınıflara fonksiyonları ekledik.
- Kodu tamamladık ardından tüm adımları tek tek iki kişi yazdık.

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

- Oyun gelistirme konusunda bir deneyimimiz olmadığından oldukça zorlandık fakat farklı kaynaklar aracılığıyla bu sorunu çözdük. Tkinter kütüphanesi için de aynı durum geçerli.

4.4 Proje isterlerine göre eksik yönler

- Projede 4 kişiye kadar çok oyunculu istenmişti fakat biz 2 kişilik yapabildik.
- Yapay zeka ekleyemedik.
- Görselleştirmede ufak komplikasyonlar bulunmakta.
- Kaynak yönetimi hatası

5 TEST VE DOĞRULAMA

5.1 Yazılımın test süreci

- Program için herhangi bir test uygulaması yapılamadı. Program tek sayfada yapıldı ve doğal olarak aynı sayfada test edildi.

5.2 Yazılımın doğrulanması

- Test sonucunda kod istenilenden farklı çalışmaktadır.
- Savaşçılar doğru, bulundukları koordinatlar da doğru fakat kaynak eksilmelerinde bazı hatalar bulunmaktadır.