**ALTIN TOPLAMA OYUNU PROJESİ**

Derya ÖZGÜ, Mehmet KIYAK

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[180201115@kocaeli.edu.tr](mailto:180201115@kocaeli.edu.tr) , [180201135@kocaeli.edu.tr](mailto:180201135@kocaeli.edu.tr)

Özet

*Proje dört oyuncu (A,B,C,D) tarafından oynanmaktadır. Oyunda kullanıcılar kendilerine en yakın kutu içerisindeki altınları hedefleyerek hamle yapmaktadır. Oyunculara göre hedef ve hamle maliyetleri değiştirmektedir.*

*Oyuncularda bulunan altın miktarı 0 (sıfır) olduğunda veya alanda ki kutularda altın kalmadığında bitmektedir. Oyunun sonunda kullanıcıların hamle ve altın skorları gösterilemektedir.*

# Giriş

Program ilk çalışma esnasında kullanıcıdan oyun tahtası için tahta boyutu, karelerin yüzde (%) kaçında altın olacak ve bunların yüzde (%) kaçı gizli olacak, kullanıcıların ilk oyuna başlama sırasında ne kadar altını olacak ve hamle başına adım sayısı istenmektedir.

Ardından oyuncular (A,B,C,D) için ayrı ayrı hamle ve hedef maliyetini istemektedir. Ardından “Oyunu Başlat” butonuna tıklayarak oyunu başlatmaktadır.

# Temel Bilgiler

Program C# programlama dilinde geliştirilmiş olup, tümleşik geliştirme ortamı olarak “Microsoft Visual Studio Preview” kullanılmıştır. Akış diyagramı için [draw.io](http://draw.io) online diyagram yapma programı kullanılmıştır

# Tasarım

Altın Toplama Oyunu programının programlanma aşamaları altta belirtilen başlıklar altında açıklanmıştır.

# Algoritma

Program ilk çalışma esnasında kullanıcıdan oyun tahtası için tahta boyutu, karelerin yüzde (%) kaçında altın olacak ve bunların yüzde (%) kaçı gizli olacak, kullanıcıların ilk oyuna başlama sırasında ne kadar altını olacak ve hamle başına adım sayısı ve kullanıcıların hamle ve hedef maliyetleri istenmektedir. Ardından “Oyunu Başlat” butonuna tıklayarak oyunu başlatmaktadır.

Oyun başladıktan sonra “Oyna” butonuna her tıklanmasında önce A kullanıcısı ardından B, C ve D kullanıcıları sırayla hamle yapmaktadır. Kullanıcılar hamle yapmadan önce kendisine en yakın altın karesini hedefleyerek ileri, geri, aşağı ve yukarı olacak şekilde hareket etmektedir. Hamle ve hedeflemelerde kullanıcı adına göre belirlenen maliyetler kullanıcıdan tahsil edilmektedir.

Oyun bu şekilde kullanıcılarda altın tükenene kadar ve hatta tahta da altın kalmayana kadar devam etmektedir. Oyun sonunda oyunculara göre hamle ve altın gibi maliyetlerin skorları görünmektedir.

1. **Main.cs :** Programın başlamasını sağlayan sınıftır. İçerisinde “**GameSettingsForm**” formunu çalıştırarak oyunu başlamaktadır.
2. **GameSettingsForm.cs:** Bu formekranında kullanıcıdan oyun tahtası için tahta boyutu, karelerin yüzde (%) kaçında altın olacak ve bunların yüzde (%) kaçı gizli olacak, kullanıcıların ilk oyuna başlama sırasında ne kadar altını olacak ve hamle başına adım sayısı ve kullanıcılar için ise kullanıcıların hamle ve hedef maliyetleri istenmektedir.
3. **GameStartForm.cs:** Kullanıcı tarafından istenilen oyun ayarlarından sonra “**GameStartForm**” formu çalışarak öncelikle tahtanı ve tahta üzerindeki kareler oluşturuluyor ve üzerine belirtilen miktarlarda altınlar yerleştiriliyor oyun sahnesini hazırlandıktan sonra oyun oynanmaktadır. kullanıcıdan ad soyad, eposta, doğum tarihi ve şifre istenilmektedir. Oyun sırasında altınlar bitene kadar bu form ekranı oyun sahnesi olarak açık kalıyor . Oyun sonunda skorlar gösterilip oyun sona eriyor.

# Kullanılan Sınıflar ve Fonksiyonlar

* + 1. **GameSettingsForm.cs :** İçerisindeki değişkenler kullanıcıdan alınan değerlere göre doldurulmaktadır.  
         
       Bunları tanımlayacak olursak;  
         
       **tahtaSatir :** Tahta üzerinde kaç adet satır olacağı bilgisini tutmaktadır.  
        **tahtaSutun :** Tahta üzerinde kaç adet sütun olacağı bilgisini tutmaktadır.  
        **kareYuzde :** Tahta üzerindeki karelerin yüzde (%) kaçında altın olacağını tutmaktadır.  
         
       **gizliKareYuzde :** Belirtilen karelerin yüzde (%) kaçında saklı altın bulunacağı değerini tutmaktadır.  
        **kullaniciAltinSayisi :** Kullanıcının ilk başlangıçta kaç adet altını olacağı bilgisini tutmaktadır.  
        **adimSayisi :** Kullanıcının her hamlede ne kadar adım atacağı bilgisini tutmaktadır.  
         
       **aHamle :** A oyuncusunun hamle maliyetini tutmaktadır. **aHedef :** A oyuncusunun hedef maliyetini tutmaktadır. **bHamle :** B oyuncusunun hamle maliyetini tutmaktadır. **bHedef :** B oyuncusunun hedef maliyetini tutmaktadır.  
         
       **cHamle :** C oyuncusunun hamle maliyetini tutmaktadır. **cHedef :** C oyuncusunun hedef maliyetini tutmaktadır. **dHamle :** D oyuncusunun hedef maliyetini tutmaktadır. **dHedef :** D oyuncusunun hedef maliyetini tutmaktadır.  
         
       Alınan değerlerden sonra “**GameStartForm”** formunun constructer fonksiyonuna değerler gönderilmektedir.
    2. **GameStartForm.cs :** Oyun bu form içerisinde gerçekleşiyor. Kullanıcılar, yollar ve tahta sınıfları içeriye dahil ediliyor.  
       Bu sınıf içerisindeki fonksiyonlar ve tanımları;

1. **GameStartForm\_Load :** Bu fonksiyon içerisinde karelere gelecek altınların sayısı belirtilmektedir. Ardından sırasıyla tahtayı ve karelerin içine altınları yerleştiren fonksiyonlar çağrılmaktadır.
2. **Oyun\_Tahtasini\_Olustur :** Kullanıcıdan alınan satır ve sütun değerlerinde butonlardan oluşan kutular oyun tahtasını oluşturmaktadır.
3. **KareAltin\_Ekle :** Oyun tahtasında oluşan karelere kullanıcıdan alınan yüzdelik dilime göre altınlar kutulara rastgele olarak ve 5 ila 20 değerlerini alarak yerleştirilmektedir.
4. **GizliKareAltin\_Ekle :** Karelere açık olarak altınlar eklendikten sonra kullanıcıdan alınan yüzdelik dilimine göre boş değerlere sahip kutulara gizli olarak altın atamaları yapılmaktadır.
5. **Kutu\_Degerini\_Goster :** Kutulara eklenen altınların kontrolü yapılıp gizli olmayan kutuların altın değerlerini göstermektedir.
6. **A\_Hedef\_Belirle :** A oyuncusu için en kısa mesafedeki hedefin belirlenmesi sağlanmaktadır. Eğer A oyuncusunun altını bitmiş hedef için maliyet ödeyemeceğinden A oyuncusunu oyun dışı ediliyor.
7. **B\_Hedef\_Belirle :** B oyuncusu için en kısa mesafedeki hedefin belirlenmesi sağlanmaktadır. Eğer B oyuncusunun altını bitmiş hedef için maliyet ödeyemeceğinden B oyuncusunu oyun dışı ediliyor.
8. **C\_Hedef\_Belirle :** C oyuncusu için en kısa mesafedeki hedefin belirlenmesi sağlanmaktadır. Eğer C oyuncusunun altını bitmiş hedef için maliyet ödeyemeceğinden C oyuncusunu oyun dışı ediliyor.
9. **D\_Hedef\_Belirle :** D oyuncusu için en kısa mesafedeki hedefin belirlenmesi sağlanmaktadır. Eğer D oyuncusunun altını bitmiş hedef için maliyet ödeyemeceğinden D oyuncusunu oyun dışı ediliyor.
10. **Cebini\_KontrolEt(Player Oyuncu) :** İçerisine gelen oyucunun altının yapacağa işleme göre yani hamle ve hedefleme için yeterli olup olmadığına bakılmaktadır.
11. **HamleYap(Player Oyuncu, Way hedef, int AdimAdeti, int k) :** Bu fonksiyonda kullanıcının hangi hedefe odaklandığı ne kadar adım gideceği ile ilgili bilgiler gelmektedir. Ardından kullanıcının hamle yapacak altını var mı diye kontrol ediliyor eğer altını yok ise hamle yapamamaktadır. Eğer yeteri kadar altını var ise hamle yapabilmektedir.
12. **Hedefi\_Kontrol\_Et(Way hedef) :** İçerisine gelen hedefin daha önce açılıp açılmadığını kontrol etmektedir.
13. **Hamleyi\_Kontrol\_Et(Player Oyuncu, Way hedef) :** Oyuncunun hedefe yapacağı hamleyi kontrol etmektedir.
14. **BtnOyna\_Click(object sender, EventArgs e) :** Form üzerinde görünen butonun tıklanmasıyla çalışan bu fonksiyon öncelikle sıranın hangi kullanıcıda olduğunu kontrol edip daha sonra oyuncunun hamlesini ve hedefini kontrol etmektedir. Yapılan kontroller sonrasında herhangi bir hata yok ise kullanıcının hedefine göre hamle yapılmasını sağlamaktadır.
15. **RenkVer(Player Oyuncu) :** Üzerinden geçilen kutunun rengini bu fonksiyon içerisinde gri tonlarında boyanmaktadır.
16. **Uc\_Gizli\_Kutuyu\_Ac :** Butona tıklandıktan ve hamlenin gerçekleşmesinden sonra gizli olan kutulardan 3 tanesini açmaya yarayan fonksiyondur.
17. **Siraki\_Kim(string oynayan) :** Hamle yapıldıktan sonra sıranın hangi kullanıcıda olduğunu bulup döndüren fonksiyondur.
    * 1. **Range.cs :** Bu sınıf içerisinde uzaklık, kazanç, x koordinatı ve y koordinatı değişkenleri yer almaktadır. Bize hedefin ne kadar uzaklıkta ve ne kadar kazanca sahip olduğunu ayrıca x ve y koordinatlarını vermektedir.
      2. **Player.cs :** Oyuncu sınıfında oyuncuya gerekli bazı bilgiler tutulmaktadır. Bunlar; oyuncu adı, harcanan altın, toplanan altın, kalan altın, adım sayısı, hedefe varılıp varılmadığı gibi değişkenleri ve fonksiyonları tutmaktadır.
      3. **Way.cs :** Yol sınıfı sadece içerisinde gidilecek hedefin hangi satırda ve sütunda olduğunu tutan iki değişkeni barındırmaktadır.
      4. **Board.cs :** Tahta sınıfı içerisinde oyun üzerinde oluşturulan kutuların hangi satır ve sütunda olduğunu hangi x y koordinatlarında olduğunu gizli mi değil mi olduğuna bakan ve belirtilen kutunun adını ve değerini tututmaktadır.

# Sonuçlar

Programımız proje kapsamında istenilen tüm isterleri yerine getirememektedir. Program da sadece iki eksik mevcut bunları çözümlerine ulaştırma konusunda yeterli zamana sahip olamadık. Hata olarak oyun çok hızlı bir şekilde bitiyor ve eksik olarak oyun sonunda oyunculara ait skorlar gösteremedik.

# Kaynakça

1. <https://www.udemy.com/course/sifirdan-ileri-seviye-csharp-programlama/> kaynağından c# diline ait gerekli terimler ve form işlemleri hakkında bilgiler edinildi.
2. <https://docs.microsoft.com/tr-tr/visualstudio/debugger/how-to-debug-an-executable-not-part-of-a-visual-studio-solution?view=vs-2019> kaynağından karşılaştığımız bazı hatalara karşı çözümler elde edildi.

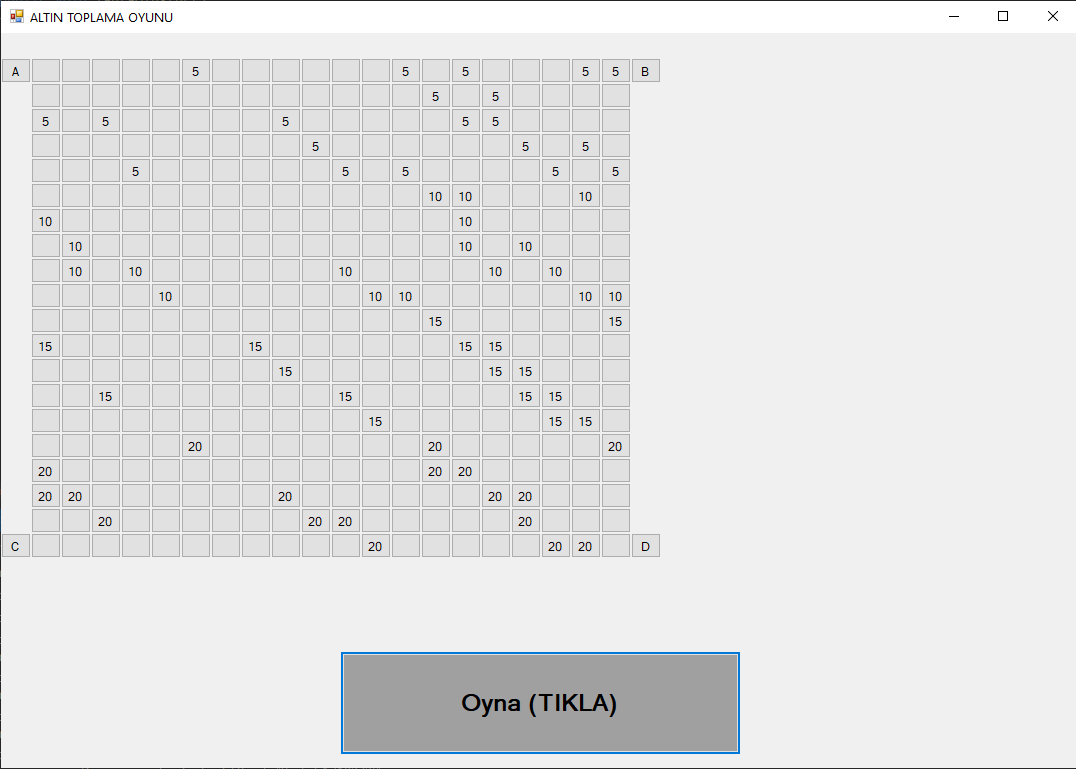
# 6. Denklemler

Program içerisinde matematiksel olarak çok büyük hesaplama bulunmadığı üzere küçük bazı denklemler kullanılmıştır.

# 7. Ekran Görüntüleri



**Resim 1. Oyuna ait dinamik ayarlar ekranı**



**Resim 2. Oyun Alanı**

