KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Minimax algoritması ile Connect4 oyunu

BİTİRME PROJESİ

Mehmet Santor

2019-2020 BAHAR DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Minimax algoritması ile Connect4 oyunu

BİTİRME PROJESİ

Mehmet Santor

2019-2020 BAHAR DÖNEMİ

IEEE Etik Kuralları
IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek, diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

Bitirme projesi çalışması bir projenin başlangıcından sonuna kadar ne tür zorluklarla karşılaşabileceğimi ve bu zorlukların üstesinden nasıl daha kolay bir şekilde gelebileceğimi öğretti.

Bir projenin gelişimi ağacın gelişimine benzer. Her şey küçük bir tohumla yani basit bir fikir ile başlar daha sonra ağacın dallarının gelişmesi gibi fikirde detaylanarak büyür. Bir ağacın gelişimi çok uzun yıllar sürdüğü gibi bir projenin de her zaman geliştirilecek detaylar, eklenebilecek özellikler ve giderilecek hatalar olacaktır. Bende bu proje boyunca elimden geldiğince fikirlerimi yapılabilirlik doğrultusunda hayata geçirmeye çalıştım.

Proje boyunca desteğini esirgemediği için sayın Cemal Köse hocama teşekkür ederim.

Mehmet Santor
Trabzon 2020

İÇİNDEKİLER

ÖZET

Günümüzde oyun dünyası çok büyük bir ivme ile gelişmekte.
Oyuncuların ise en çok dikkat ettiği özelliklerden biri olan yapay zekâ, nitelikli kodlandığı zaman oyunculara iyi bir oyun deneyimi yaşatıyor. Aksi durumda da oyuncuları çılgına döndüren bir oyun deneyimi sunuyor. Peki, yapay zekanın geliştirilmesi gelecekte oyunları nasıl etkileyecek?

Yapay zekâ ne diye soracak olursanız, oyunlar bağlamında yapay zekayı, herhangi bir oyunun dünyasında sizin dışınızda meydana gelen olayları gerçekleştiren bir mekanizma olarak tanımlayabiliriz. Bu olaylara örnek verecek olursak oyunlardaki yapay zekâ, GTA oyununda bir arabanın yolda gitmesi, Need for Speed'deki yarıştığınız arabaların sizi geçmesi, Call of Duty'deki düşman askerlerin size ateş etmesi gibi olayların gerçekleşmesini sağlıyor.

Gelecekte yapay zekanın oyunlarla nasıl daha fazla iç içe geçeceğini anlamak için iki sektörün de geçmişine bakmak gerekiyor. Oyun sektörünün ilk günlerinden beri geliştiriciler, yapay zekanın insan gibi davranması ve gerçek bir insana ihtiyaç duyulmadan, sıfırdan oyun dünyası yaratabilmesi için çaba gösteriyorlar.

Bu projede ise bir yapay zekânın nasıl geliştirebileceğimizi sorguladım.
Bunun sonucunda kendi yapay zekâ tabanlı oyunumu oluşturup en iyi hale getirmek istedim.

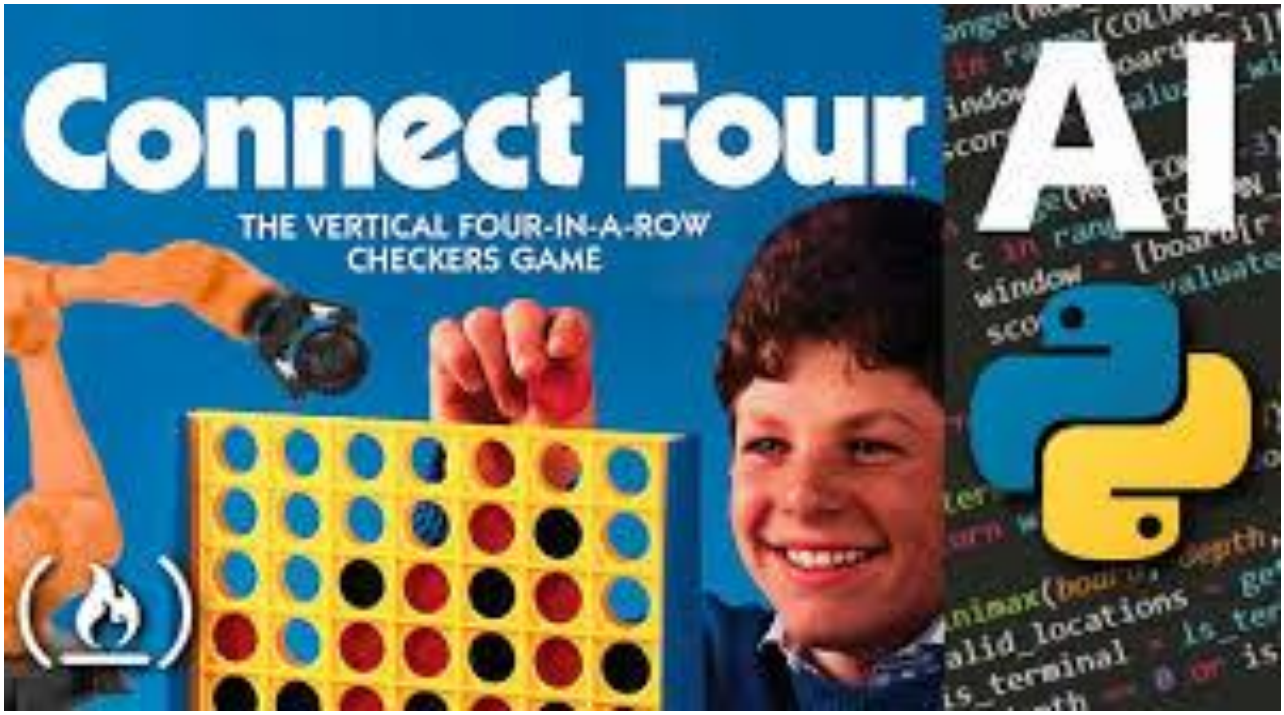
1. GENEL BİLGİLER

1.1.CONNECT 4

Dörtleme yani connect 4 oyunundaki amaç oyun içinde rakibinizi kandırarak dört taşınızı ardışık şekilde bir araya getirmektir. Bu sıralama yatay, dikey veya çapraz şekilde olabilir. İyi bir stratejik düşünme gerektiren dörtleme oyunu iki oyuncu arasında oynanır. Oyunu kazanabilmek için rakibin hamlelerini de düşünerek kendinize oyun planı belirlemeniz gerekir. Unutmayın ki; rakibiniz de kazanmak için aynı şeyleri düşünüyor. Kendi planınızı uygularken, rakibin hamlelerine de dikkat etmeniz gerekir. Rakibinizin oyununu bozacak şekilde stratejiler üretmelisiniz.
1974 yılında Milton Bradley firması tarafından oyun piyasasına sürülen connect 4, İngilizce isminden de anlaşılacağı gibi dört taşı birleştirmek üzerine kurgulanmıştır. Dörtleme oyunu farklı ebattaki oyun tahtalarında oynanabilmektedir. En çok kullanılan oyun tahtaları: 7x6, 8x7, 9x7 ve 10x7 olanlardır

OYUN NASIL OYNANIR?

En sık kullanılan 7 satır ve 8 sütundan oluşan 7x8 oyun tahtasıdır. Bu tahta, oyun disklerinin üst üste sıralanabileceği özel bir oyun tahtasıdır. Oyun iki farklı renklerden oluşan disklerle oynanmaktadır. Oyun içinde ilk dörtlemeyi yapan taraf oyunu kazanmış olur.
Her oyuncu oyuna başlamadan önce iki farklı renkten birini seçer ve oyuna başlar. Her oyuncu sahip olduğu diskleri sırasıyla oyun tahtası üzerindeki seçtiği sütundan içeri bırakır. Diskler sütunda inebilecekleri satıra kadar düşerler. Bu şekilde sırayla atılan diskler sayesinde sütun ve satırlar dolmaya başlar. Bu diskleri boşluklara atarken belirli bir stratejik plan içinde hareket etmeniz gerekir. Rastgele atılan diskler sıradan rakipler karşısında üstünlük sağlayabilir ama düşünmeden oynanan oyunun size keyif vereceğini de zannetmiyorum.
Dörtleme oyunu değişik bir oyun olup, oynaması oldukça zevk verir. Basit gibi gözüken bu oyun kişilerin farklı şekillerde düşünmesini sağlamaktadır. Oyun tahtası üzerinde atılan diskler özellikle çocuklar için eğlenceli gelir. Onlar eğlenirken fark etmeden algıları da üç boyutlu şekilde dörtleme oyunu sayesinde gelişecektir.



1.2.Python

Python programlama dili veri bilimi, makine öğrenimi, sistem otomasyonu, web ve API geliştirme ve daha fazlası için bir temel yapıdır.

1991'den beri Python programlama dili sadece gereksiz programlar için tamamlayıcı bir dil olarak değerlendiriliyordu. Hatta “Automate the Boring Stuff” (Türkçe ‘ye "Sıkıcı Şeyleri Otomatikleştiren" olarak çevirebileceğimiz popüler bir kitap) adında bir kitap dahi yayınlanmıştır.

Bununla birlikte son birkaç yılda Python modern yazılım geliştirme, altyapı yönetimi ve veri analizinde birinci sınıf bir programlama dili olarak ön plana çıkmıştır. Artık hackerler için bir arka kapı oluşturucusu değil, web uygulaması oluşturma ve sistem yönetiminde önemli rol alma, veri analizleri ve makine öğreniminde parlayan bir dil olarak ün kazanmıştır.

Python ’un Önemli Avantajları

Python programlama dili yeni başlayanlar veya Python ‘da uzmanlaşanlar için önemli avantajlara sahiptir.

Python’ u Öğrenmek ve Kullanmak

Diğer karmaşık program dillerini öğrenmek çok zaman alır ve kullanım alanları büyük olmasından dolayı kullanımını öğrenmek çok zordur. Ancak Python sözdizimi hem okunabilir hem de ileriye dönüktür. Öğrenim, kararlı programlama dili sayesinde basittir. Yeni başlayanlar için de ideal bir seçim olarak ön plana gelir. Sonuç olarak, Python kullanarak program geliştirmeye yeni bir adım atmış olan herkes hızlı ve basit şekilde ilerleme kaydedebilir. Diğer karmaşık dillere göre basitlik söz konusu olduğunda, en önde yer almaktadır.

Python’ un Uygulanabilirliği

İsminin az duyulmasına rağmen, yazılımcılar arasında hem popülerdir hem de yaygındır. Github projelerinin birçoğu Python tabanlıdır. Hatta Tiobe Index ve Github gibi programlama örneklerinin sıralandığı sistemlerde, Python tabanlı programları en üst sıralarda göstermektedir. Kullanım alanı en küçük işletim sistemlerinden en büyük işletim sistemlerine kadar birçok OS tarafından desteklenmektedir. İnternette yer alan büyük yazılım kütüphanelerin ve API-destekli servislerin mutlaka Python tabanlı bindings ya da wrappers’ları vardır. Bundan dolayı Python, bu servisler ile serbestçe ara yüz oluşturur veya bu kütüphaneleri doğrudan kullanır. Yazılım dilleri arasında en hızlısı olmasa da çoklu platformlarda kullanılması bir adım önde olmasını sağlamıştır.

Python Basit bir dil değildir

Oluşturulan scriptlerin ve otomasyon programlarının çoğu Python kod şemalarını kapsasa da Python ayrıca hem bağımsız uygulamalar hem de web hizmetleri olarak profesyonel kalitede yazılımlar oluşturmak için kullanılır.

Python ne için kullanılır

Ayrıca, Ansible ve Salt gibi araçlarda sistem gereksinimlerini ve yapılandırmasını sağlayan, bunlara ek olarak web tarayıcıları veya uygulama GUI leri ile etkileşimleri otomatikleştiren özellikleri de bulunmaktadır. Kısaca anlatmak gerekirse, script oluşturmak ve otomasyon Python için buz dağının sadece görünen kısmıdır.

Python ile uygulama programlama

Python programlama dili ile konsol uygulamaları ve çoklu platformlara GUI uygulamaları oluşturabilirsiniz. Bunları bağım kurulum dosyaları olarak kullanabilirsiniz. Python ile oluşturulan bir Script kendi başına bir binary serisi oluşturamaz, ancak cx_Freeze and PyInstaller üçüncü taraf programlar sayesinde durum imkansızdan mümküne dönüşmektedir.

Veri Bilimi ve Makine Öğreniminde Python ‘un Yeri

Sofistike veri analizleri günümüzde IT için en önemli konular haline gelmiştir. Python ise bu durumlar için en elverişli programlama dili olmuştur. Python ara yüzündeki kütüphanelerin birçoğu makine öğrenimi ve veri bilimi üzerine elverişlidir. Bu alanlardaki kütüphanelerde ki yüksek kaliteli komutları, makine öğrenimi kütüphanelerinin ve diğer nümerik algoritma kütüphanelerinin sürekli gelişmesine çok yardımcı olmuştur.

Python ile Web Servisleri ve REST ful API’leri Kodlama

Python içerisinde bulunan yerel kütüphaneler üçüncü parti web yazılımları ile birleştiğinde, ortaya birkaç satır kod ile daha hızlı bir web site yönetimi elde edilebilir. REST API’lerin uygun kod blokları ile oluşturulması seri bir şekilde sitenin veri yürütme olayını hızlı yapmasına olanak sağlar. Özellikle Python ‘un son güncellemesi asenkron operasyonlarının daha güçlü bir yapıya dönüştürmüştür. Doğru kütüphaneler ile sitelerin saniyede binlerce kod istemine karşılık veri akışı sağlamasına yardımcı olmuştur.

Metaprogramlama ve Kod Derlemesi

Diğer diller ile karşılaştırıldığında Python ‘da yer alan bütün modüller ve kütüphaneler birer nesne olarak görev yapar. Bunun sayesinde Python, etkili bir kod derleyici olarak ön plana çıkmaktadır. Kendi kod özellikleri ile yazılan uygulamaları manipüle ederek, diğer dillerde yazılması çok zor olan ya da neredeyse imkânsız olan uygulamaları oldukça kolay yazılabilir hale getirir.

Python ‘un çoklu platform uygulamaları oluşturması da LLVM benzeri kod derleyici sistemlerinde etkili kodlar oluşturmak için uygun olmasını da sağlar.

Yapışkan Kod olarak Python

Sıklıkla Glue Code (Yapışkan Kod) olarak adlandırılan Python, bu takma ismini C dilinin kütüphanesindeki kodlar ile ortaklaşa bir yapı oluşturup, etkili program yapılmasına yardımcı olduğu için almıştır. Veri bilimden ve makine öğreniminde eşsiz bir seçenek olarak ön plana gelmesini sağlamıştır.

Python'un Eksiklikleri

Her ne kadar Python genel anlamda kaliteli uygulamalar yazmak için uygun olsa da bazı eksiklikleri ve yetemediği alanlarda vardır.

Yüksek düzeyde bir programlama dili olduğu için sistem düzeyinde programlama için uygun değildir. Ancak bu kategori içinde aygıt sürücüleri veya işletim sistemi çekirdeği yer almaz.

Ayrıca, platformlar arası bağımsız binary dosyalar için çağrı yapan durumlarda da ideal değildir. Windows, Mac OS ve Linux işletim sistemleri için bir uygulama geliştirebilirsiniz. Ancak oluşturulacak uygulama fazla görkemli uygulama olmaz

Genel olarak Python programlama dilin de hız ön planda olduğu için ağır programlarda fazla işlevsellik sağlamaz. Bundan dolayı en eski programlama dillerinden olan C / C ++ gibi diller kullanabilirsiniz.

Python işlerinizi nasıl kolay hale getirir?

Python program dili oldukça kısa ve okunabilir syntax' lardan oluşur. Python 'un en son ki 3.x güncellemesindeki standart "Merhaba Dünya" yazısı şu şekildedir.

```
print ("Hello world!")
```

Python 'da birçok yaygın program akışını açık bir şekilde ifade etmek için birçok nesne elemanları kullanabilirsiniz. Bir liste nesnesinin içindeki txt dosyasından satırları okuyan standart bir program düşün ve bunları dizi biçiminde sıralamasını istiyorsunuz. Bunun için kod aşağıdaki gibidir.

```
with open('apsisx.txt') as ap

sisx_file:

    file_lines = [x.strip('\n') for x in apsisx_file]
```

Programın kod dizimi açısından incelediğimizde "with/as" yapısı, bir kod bloğu bir nesneyi örneklendirmek ve daha sonra bu bloğun dışına atmak için etkili bir yol sağlayan bir içerik yöneticisidir. Bu durumda, nesnemiz open () işleviyle çağrılan apsisx_file olmuştur. Bu saye de dosyayı açmak, ondan tek tek satırları okumak ve daha sonra kapatmak için birkaç satırlık ekstra kod yazmanın gereği kalmamıştır.

[x.strip('\n') for x in apsisx_file] yapısı başka bir Python dil şemasıdır. Diğer öğeleri (buradaki, apsisx_file ve diğer kodlar) içeren yapı kodların yinelenmesini sağlar. Her yinelenen nesnenin (that is, each x) işlenmesini ve otomatik olarak bir listeye eklenmesine yardımcı olur.

Genel olarak Python ‘da başka bir dillerde de oluşturulan "for..." kalıbını döngü olarak yazabilirsiniz. Temel nokta, Python ‘un çoklu nesneler üzerinde yineleyen döngüler gibi şeyleri ekonomik olarak ifade etmenin bir yolu olduğu yönündedir. Ayrıca, döngüdeki her öğe için basit bir işlem gerçekleştirmeye veya açık bir örnekleme ya da elden çıkarma gerektiren şeylerle çalışmanıza yardımcı olur.

Bu gibi kod blokları, Python dili ile program geliştiren insanların farklılık ve okunabilirliği dengelemelerine izin verir.

Python ‘un diğer dil özellikleri, yaygın kullanım durumlarını tamamlayıcı niteliktedir. "Unicode dizeleri" gibi en modern nesne türleri doğrudan dil içerisine aktarılabilir. Listeler, sözlükler (hashmaps), tupllar (nesnenin değişmez yapılarını depolamak için kullanılır) ve setler (sadece belirli kod dizgileri için kullanılır) gibi veri yapıları mevcuttur.

Python 2 ve Python 3 Karşılaştırması

Python ‘ın mevcut olarak iki sürümü bulunmaktadır. Bunlar yeni kullanıcıların ufkunu açacak özelliğe sahiptir. Bunlardan Python 2.x, "legacy" olarak geçmektedir ve 2020'ye desteklenmeye ve güncellemeleri almaya devam edecektir. Ancak bu süre dolduktan sonra gayri resmi olarak devam edebilir. Python 3.x, şu an mevcut olarak işletim sistemlerine sunulan ve 2.x versiyonunda bulunmayan birçok özelliğe sahiptir. Bunlardan başlıcaları daha iyi eşzamanlılık kontrolleri ve daha verimli bir derleme sistemidir.

Üçüncü parti kütüphane desteklerinin Python 3'ü desteklemesi biraz yavaş olarak ilerlemektedir ve tam olarak entegre edilmesi için zaman alacaktır. Birçok Python kitaplığı yalnızca Python 2'yi desteklediğinden dolayı doğrudan geçişi zor hale getirmiştir. Ancak son birkaç yılda sadece Python 2'yi destekleyen kütüphanelerin sayısı azaldı. Bunun yerine çoğu kütüphane her iki sürümle uyumlu olarak çalışmaktadır. Bugün, Python 3'ü kullanmamanın birkaç nedeni var.

Python’ un kütüphaneleri

Python ‘un son zamanlarda bu kadar üstün bir başarıya sahip olması, kendine özgü zengin ve bunlara ek olarak üçüncü taraf yazılımların ekosistemine dayanıyor. Python hem güçlü bir standart kütüphaneden hem de üçüncü taraf geliştiricilerden kolayca elde edilen, edinilen bilgilerin de kolayca kullanılabilen kütüphanelerden destek alır. Bu programlama dili onlarca yıllık gelişmenin yanı sıra yapılan katkılarla daha da zenginleştirilmiştir.

Python ‘un standart kütüphanesi, matematik, dizi işleme, dosya ve dizin erişimi, ağ oluşturma, asenkron işlemler, iş parçacığı, çoklu işlem yönetimi gibi yaygın programlama görevlerine yönelik modüller sağlar. Ayrıca aynı zamanda modern uygulamaların ihtiyaç duyduğu ortak, üst düzey programlama görevlerini yöneten JSON ve XML gibi yapılandırılmış dosya formatlarını okuma ve yazma, sıkıştırılmış dosyaları kullanma, internet protokolleri ve veri formatlarıyla çalışma (web sayfaları, URL'ler, e-posta) gibi modülleri de içerir. C-uyumlu yabancı fonksiyon ara yüzünü ortaya çıkaran çoğu harici kod Python ‘un ctypes modülü ile erişilebilir.

Varsayılan Python sürümü tam olarak gelişmemiş bir sürümdür. Ancak Tkinter üzerinden çoklu platform GUI kütüphanesi ve SQL ite 3 veri tabanının gömülü bir kopyası ile kullanılabilir.

Python Paket Endeksinde (PyPI) bulunan binlerce üçüncü taraf kütüphanesi, Python ‘un popülaritesi ve çok yönlülüğün en büyük göstergesi olmaktadır.

Örneğin:

- BeautifulSoup kütüphanesi, HTML'in tam olarak analizini yaparak dışarıya bütün verileri ayıklayan herşeyin bir arada olduğu bir araç görevi görür.
- Flask ve Django gibi frameworkler hem basit hem de gelişmiş kullanım durumlarını kapsayan web servislerinin hızlı geliştirilmesine olanak tanır.
- Çoklu bulut hizmetleri, Python ‘un nesne modeli ile Apache Libcloud kullanılarak yönetilebilir.
- NumPy, Pandas ve Matplotlib, matematik ve istatistiksel işlemleri hızlandırır ve verilerin görselleştirilmesini kolaylaştırır.

Python’ u Alt Seviyeye Düşüren Özellikleri

C #, Java ve Go programlama dilleri gibi Python da toplanan bellek yönetimine sahiptir. Yani programcı, nesneleri izlemek ve bırakmak için kod uygulamak zorunda kalmamasını sağlar. Normalde, çöp toplama arka planda otomatik olarak gerçekleşir, ancak bu bir performans sorunu ortaya çıkarırsa, bunu manuel olarak tetikleyebilir veya tamamen devre dışı bırakabilirsiniz.

Python ‘un önemli bir yönü dinamizm bir yapıya sahip olmasıdır. Nesne tabanlı bir programlama dili olarak kullanıldığı için dildeki her şey, işlevler ve modüller de dahil olmak üzere, nesne olarak ele alınır. Bu sayede kullanılan nesneler üst düzey kod yazmayı çok daha kolay hale getirir. Geliştiriciler karmaşık nesne manipülasyonlarını yalnızca birkaç satır kod ile gerçekleştirebilir ve hatta bir uygulamanın parçalarını gerektiğinde değiştirilebilecek soyutlamalar düzenleyerek kullanabilirler.

Python ‘un "significant whitespace" denilen boşluklar kullanması, Python ‘un en iyi ve en kötü özelliklerinden biri olarak değerlendirilebilir. Aşağıdaki ikinci satırdaki girinti sadece okunabilirlik için değildir; Python’un kullanımında ki kod sıralamasının bir parçasıdır. Python derlemesi, kontrol akışını göstermek için uygun girintileri kullanmayan kod satırlarını reddeder.

```
with open(‘apsisx.txt’) as apsisx_file:

    file_lines = [x.strip(‘\n’) for x in apsisx_file]
```

Bu boşlukların kullanımı bazen zahmetli olduğu için kullanıma yeni başlayanlar için bir problem olarak nitelendirilir. Ancak katı girinti kuralları, genel olarak göründüğünden çok daha az zorlayıcı etmendir. Ayrıca görsel olarak daha okunabilir bir hale gelmesine yardımcı olur.

C veya Java gibi diller ile kıyaslandığında bir başka potansiyel sıkıntı, Python ‘un değişken yazmayı nasıl ele aldığıdır. Varsayılan olarak Python, hızlı kodlama için harika, ancak büyük kod tabanlarında sorun yaratabilecek dinamik veya "duck" denilen ara kod yazımını kullanır. Bununla birlikte, Python son zamanlarda isteğe bağlı derleme-zamanı ipucu desteği ekledi. Bu sayede statik yazmadan yararlanabilecek projeler kullanabilir hale getirilebilir.

Python çok mu yavaş? Olmak zorunda değil

Python hakkında söylenilen olumsuz söylentiler arasında yavaş olması yer almaktadır. Genel açıdan değerlendirmek gerekirse, evet Python yavaştır. Python aracılığı ile oluşturulan programlar genellikle C / C ++ veya Java'daki karşılık gelen programlardan çok daha yavaş çalışır.

Neden bu kadar yavaş? Derleme programlarının satır satır kodları analiz etmesinden kaynaklı olarak bu durum yavaş senkronize gerektirir. Python ‘da yer alan nesneler kullanılan kodlar aracılığı ile belirlendiği için, derlendiğinde bile, dil için hızını optimize etmeyi zorlaştırması gerçeğidir. Bununla birlikte, Python ‘un hızı, göründüğü kadar önemli bir konu olmayabilir ve onu hafifletmenin yolları vardır.

Python’ un hızını artıracak oldukça fazla yol vardır

Genel anlamda yavaş olarak değerlendirilen Python her zaman yavaş olmaz. Pek çok Python programı yavaştır, çünkü Python ‘da yer alan veya üçüncü taraf standart kütüphanelerdeki görevleri düzgün bir şekilde içe aktaramazlar. NumPy ve Pandas gibi kütüphaneleri kullanarak matematik ve istatistik işlemleri muazzam bir şekilde artırılabilir ve PyPy çalışma zamanı, birçok Python uygulaması için fark edilir miktarda hız sağlar.

Genel olarak yazılımcılar arasında yer alan bir söz vardır. Programın %90'ı işlevselliğidir, geri kalan %10'u ise programın kodlarıdır. %10'luk dilimi optimize ettiğinizde programın hızlanması oldukça başarılı olacaktır. Python ile, Cython veya Numba gibi projeleri kullanarak bu yüzde 10'unu C'ye hatta assembly diline dönüştürebilirsiniz. Ancak ortaya çıkan sonuç, C'de oluşturulan programa göre biraz daha yavaş bir kullanım sergiler, ama C'nin hafıza mikro yönetimi özelliği göz ardı edildiğinde C ile yarışabilecek potansiyele sahiptir.

Ya da başka bir şekilde söylemek gerekirse: Birçok görev için, geliştirme hızı yürütme hızını yener.

Aynı programın hem Python ile hem de başka bir dilde çalışma süreleri değişiklik gösterir. Örneğin Python ‘da program 6 saniyede aktif hale gelir, başka bir dilde bu süre 1 saniyeye kadar düşmektedir. Ancak bir geliştiricinin Python ara yüzün bir programı oluşturması on dakikasını alırken, aynı programı başka platformda oluşturması saatlerini alabilir. Python programının yürütülmesinde kaybedilen zaman miktarı, geliştirme sürecinde kaydedilen zamandan daha az olduğu için daha idealdir.

Açıkçası bir ticaret uygulaması gibi yüksek verimli, düşük eş zamanlılık gerektiren talepleri olan yazılımlar yazarken bu daha az doğrudur. Ancak birçok sıradan uygulamalar için, sistem yönetiminden makine öğrenmeye kadar uzanan alanlarda, Python yeterince hızlı olduğunu kanıtlayacaktır.

Ayrıca, Python ‘un sağladığı esneklik ve ilerleme hızı, diğer dillerde elde etkisi daha zor ve zaman alıcı olan program geliştirmelerde güzel bir adım olabilir.

Geliştirme hızı ve programlayıcı rahatlığı ön planda olacaksa ya da makinenin işlem sırası arka planda olacaksa, Python her iki durumda da yani geliştiricinin rahat olarak çalışmasına yardım ederek hızlı bir program oluşturmak için en iyi araç olabilir.

1.3.Pygame kütüphanesi

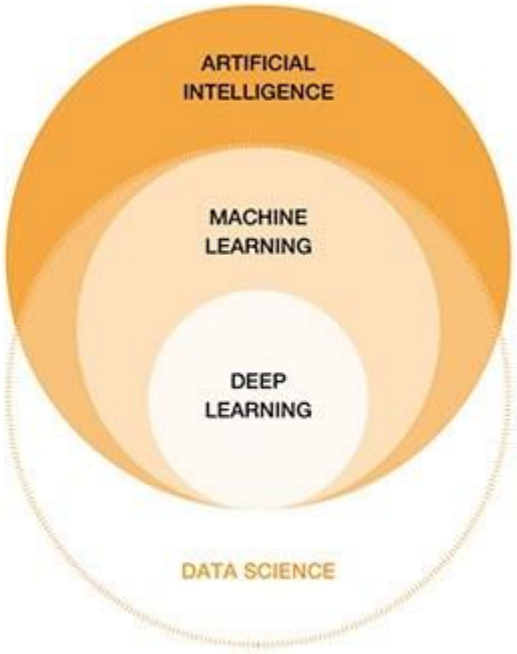
Oyun gibi muti medya uygulamaları yapmak için açık kaynaklı ve ücretsiz python programlama dili kütüphanesidir. Her işletim sisteminde; Windows, MacOS ve Linux çalışmaktadır. Pygame, Pete Shinners tarafından Python diline uyumlu etkileşimli oyun hazırlamak için SDL kütüphane üzerine kurulmuş olan bir kütüphanedir.
Pygame, kullanıcılarına birçok medya türünün desteklendiği bir arayüz sunar. Bu ara yüzde; .jpg, .gif, .png, .bitmap, .mp3, .wav, ve .midi medya türlerini desteklemektedir. Ara yüzde fare ve klavye için bulunan birden girdi fonksiyonuyla kullanım kolaylığı sağlamakla beraber etkileşimi kolaylaştırır. Barındırdığı birçok özellik ile geliştiricilerin oyun üzerinde ki hakimiyetini arttırır.
Pygame sadece oyun da değil, müzik, sanat, video, multimedya projeleri gibi birçok projeyi bünyesinde ağırlıyor. İçerisinde barındırdığı birçok modüllerle birlikte her alanda kullanım kolaylığı sağlamaktadır. Bununla Windows üzerinde hazırlanan içerikler diğer işletim sistemlerinde de sorunsuz olarak çalıştırılmaktadır. Pygame ile geliştirdiğiniz ürünleri lisanslayıp yayınlama imkânı da sunmaktadır. Hazırladığınız oyunu lisanslayarak maddi kazanca dönüştürme fırsatı da yakalayabilirsiniz.
Pygame Nedir: Pygame SDL, Simple DirectMedia Layer denen bir kütüphaneden güç almaktadır. SDL, C ve Assembly dili kullanılarak hazırlanan, çeşitli optimizasyon aşamalarından geçmiş ve hız konusunda başarılı bir platformdur. Pygame de bu noktalardan SDL'den güç alarak hazırlanmıştır. Daha kullanışlı bir platform olmayı amaçlayarak kullanıcılarına kaliteli bir arayüz sunmaktadır.

Pygame ile oyun hazırlama:

Pygame oyun hazırlamak gibi komplike işlerde başarılı olduğu gibi sıradan bir video kurgusu yapmaya kadar geniş bir alanda kaliteli hizmet vermektedir. Python destekli olmasıyla kullanıcıların daha fazla ilgisini çekmektedir. Python dili kolay anlaşılır olmasıyla tercih edilmektedir. Python kullandığı C koduyla yavaş kalmaktadır. Pygame de bu noktada SDL yardımıyla bu sorunu çözmeyi amaçlamaktadır.
Aynı zaman da Pygame kullanıcılarını oyun hazırlama teşvik etmek için çeşitli çalışmalar yapmaktadır. PyWeek diye adlandırılan 'bu hafta en iyi oyunu kim yapacak' etkinlikleriyle geliştiricileri teşvik etmeyi amaçlamaktadır. Siteyi ziyaret ederek yayınlanan oyunları oylayabilirsiniz. Yayınlanan oyunların hepsinin açık kaynaklı olmasıyla oyunları inceleme fırsatı da yakalayabilirsiniz.

1.4.Yapay Zekâ

En basit ifadeyle Yapay zekâ (AI), görevleri yerine getirmek için insan zekasını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistemler veya makineler anlamına gelir. Yapay Zekâ pek çok biçimde kendini gösterir. Örneğin:
Sohbet robotları, müşterilerin sorunlarını daha hızlı bir şekilde anlamak ve daha verimli cevaplar vermek için yapay zekadan yararlanır
Akıllı asistanlar, zamanlamayı iyileştirmek için büyük kullanıcı tanımlı veri kümelerinden kritik bilgileri çekmek için yapay zekadan yararlanır
Öneri motorları, kullanıcıların izleme alışkanlıklarına göre TV programları için otomatik öneriler sunabilir
Yapay Zekâ, herhangi bir özel biçim veya işlevden ziyade süper güçlendirilmiş düşünce ve veri analizi yeteneği ve süreci ile ilgilidir. Yapay Zekâ dendiğinde zihinlerde dünyayı ele geçiren çok fonksiyonel, insan benzeri robotlar canlansa da yapay Zekâ insanların yerine geçmek üzere tasarlanmamıştır. İnsan yeteneklerini ve katkılarını önemli ölçüde geliştirmek üzere tasarlanmıştır. Bu nedenle oldukça değerli bir ticari varlıktır.



Yapay Zekânın geleceği

Yapay zekâ hayatımızın bir parçası haline gelmektedir. Gelecekte ne olacağı belli olmamakla birlikte yapay zekâ ile ilgili filmler, kitaplarda ve ünlü teknoloji şirketlerin sahipleri yapay zekâ geleceği hakkında söylenen sözleri sizin için derledik. Birçok kişi tarafından yapay zekâ geleceği önemli noktalara geleceği bilinmekte ve ilerleyen zamanda bizi ne bekleyeceği bilinmiyor…

Konuyla ilgili söylenen sözler

- “Yapay zekâ (AI) insanlık tarihini en büyük olayı olacak ya insanlığın basına gelen en iyi şey ya da en kötü şey olacak.” (Stephen Hawking)

- “Yapay zekâ (AI) muhtemelen büyük olasılıkla dünyanın sonuna götürecek, ancak bu arada büyük şirketler olacak.” (Sam Altman, OpenAI Başkanı)

- “Makine öğrenimi (Machine Learning), insan anlayışını aşan yazılım çözümleri geliştirmemize olanak tanıyor ve yapay zekanın her endüstriyi nasıl canlandırabileceğini gösteriyor.” (Steve Jurvetson, SpaceX ve Tesla Yönetim Kurulu Üyesi)

- “Yapay zekâ ve bilgisayar bilimi vaadi, uçağın icadı demiryolu endüstrisini olumsuz yönde etkilemekle birlikte, bazı işler üzerindeki etkisinin genel olarak daha ağır bastığını gösteriyor.” (Paul Allen, Microsoft Kurucu Ortağı)

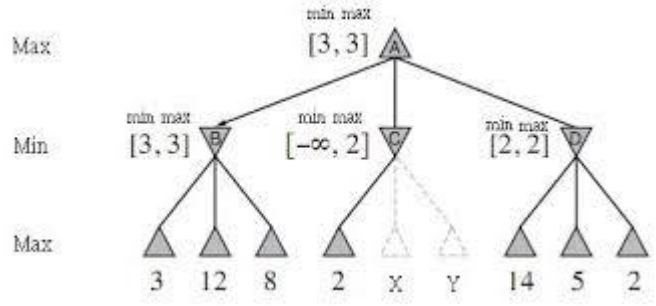
- “Hepimizin yapması gereken yapay zekayı (AI) insanlığın yararına değil, insanlığın yararına olacak şekilde kullandığımızdan emin olmaktır.” (Tim Cook, Apple CEO’su)

- “Uzun vadede, yapay zekâ ve otomasyon insanlara amaç hissi veren şeylerin çoğunu devralacak.” (Matt Bellamy, Muse Baş Şarkıcısı)

- “Toplumun yapay zekâ ile nasıl başa çıktığını görmek ilginç olacak, ama kesinlikle harika olacak.” (Colin Angle, iRobot CEO’su ve Kurucusu)

- “Yapay zekanın (AI) dönüşmeyeceğini düşünmek zor. Sağlık, eğitim, ulaşım, perakende, iletişim ve tarım da buna dahil olmuştur. Yapay zekanın tüm bu sektörlerde büyük bir fark yaratması için şaşırtıcı derecede açık yollar var.” (Andrew Ng, Yapay Zekâ Bilgisayar Bilimcisi ve Küresel Lider)

1.5. Minimax Algoritması



Yapay Zekanın ortaya çıkışından bu yana, oyun oynamak AI'nın en ilginç uygulamalarından biri olmuştur.
İlk satranç programları, 1950'de bilgisayarların programlanabildiği anda Claude Shannon ve Alan Turing tarafından yazılmıştır.

Satranç, tic-tac-toe ve Go gibi oyunlar ilginç çünkü iki ordu arasındaki rekabeti tamamen soyutlıyorlar.

Yapay Zekanın araştırmaları için oyunu cazip bir alan haline getiren bu soyutlamadır.

Minimax algoritması nedir?
Minimax, diğer oyuncunun da en iyi şekilde oynadığını varsayarak bir oyuncu için en uygun hareketi seçmek için kullanılan özyinelemeli bir algoritmadır.
Tic Tac Toe, go, satranç, Isola, dama ve diğer birçok iki oyunculu oyunlar gibi oyunlarda kullanılır.
Bu oyunlara mükemmel bilgi oyunları denir çünkü belirli bir oyunun tüm olası hareketlerini görmek mümkündür.
Scrabble gibi mükemmel olmayan iki oyunculu oyunlar olabilir, çünkü rakibin hareketi tahmin edilemez.
Bir oyun oynadığımızda nasıl düşündüğümüze benzer: "Bu hamleyi yaparsam, o zaman rakibim sadece bu hamleleri yapabilir" vb.
Minimax buna denir çünkü diğer oyuncu maksimum zarara sahip olan stratejiyi seçtiğinde kaybın en aza indirilmesine yardımcı olur.

Terminoloji
Oyun Ağacı: Oyunun durumundan bir sonraki aşamaya geçmenizi sağlayan tüm olası hareketlerden oluşan bir ağaç şeklinde bir yapıdır.

Bir oyun, aşağıdaki bileşenlerle bir arama problemi olarak tanımlanabilir:
İlk durum: Tahtanın pozisyonunu ve hareketinin kim olduğunu gösterir.

Halef işlevi: Bir oyuncunun yapabileceği yasal hamlelerin ne olduğunu tanımlar.

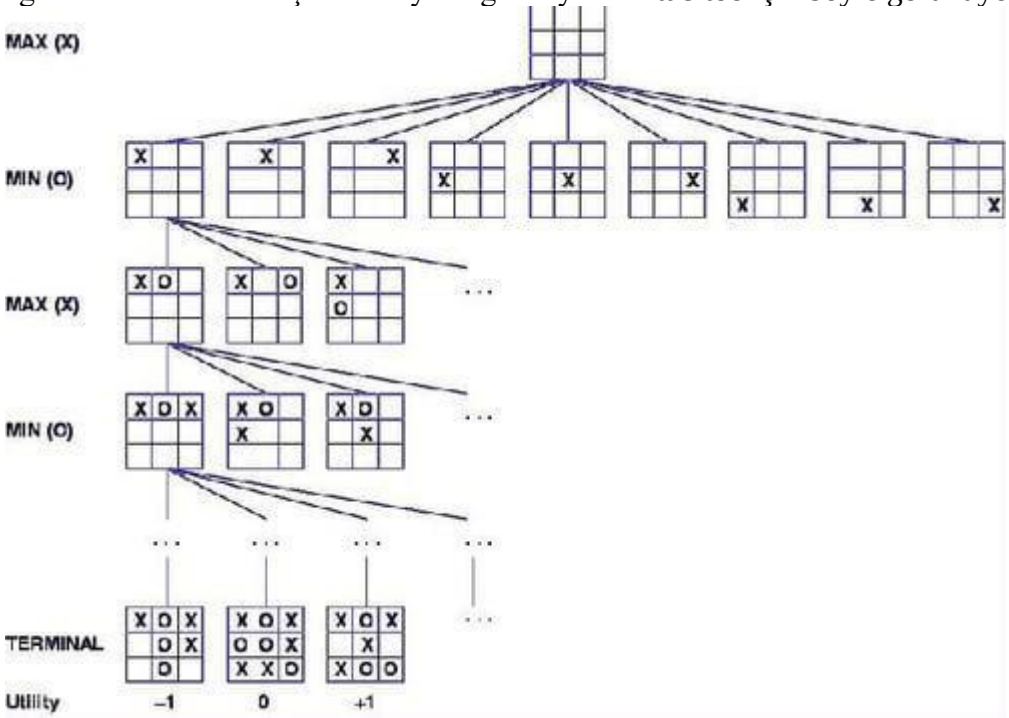
Terminal durumu: Oyun bittiğinde tahta pozisyonudur.

Fayda fonksiyonu: Bir oyunun sonucuna sayısal bir değer atayan bir fonksiyondur. Örneğin, satrançta veya tic-tac-toe'da, sonuç ya bir kazanç, bir kayıp ya da beraberliktir ve bunlar sırasıyla +1, -1 ya da 0 değerleri ile gösterilebilir. Çok daha geniş bir sonuç yelpazesine sahip oyunlar var; örneğin, tavladaki yardımcı programlar +192 ile -192 arasında

değişmektedir. Bir yardımcı fonksiyon aynı zamanda bir geri ödeme fonksiyonu olarak da adlandırılabilir.

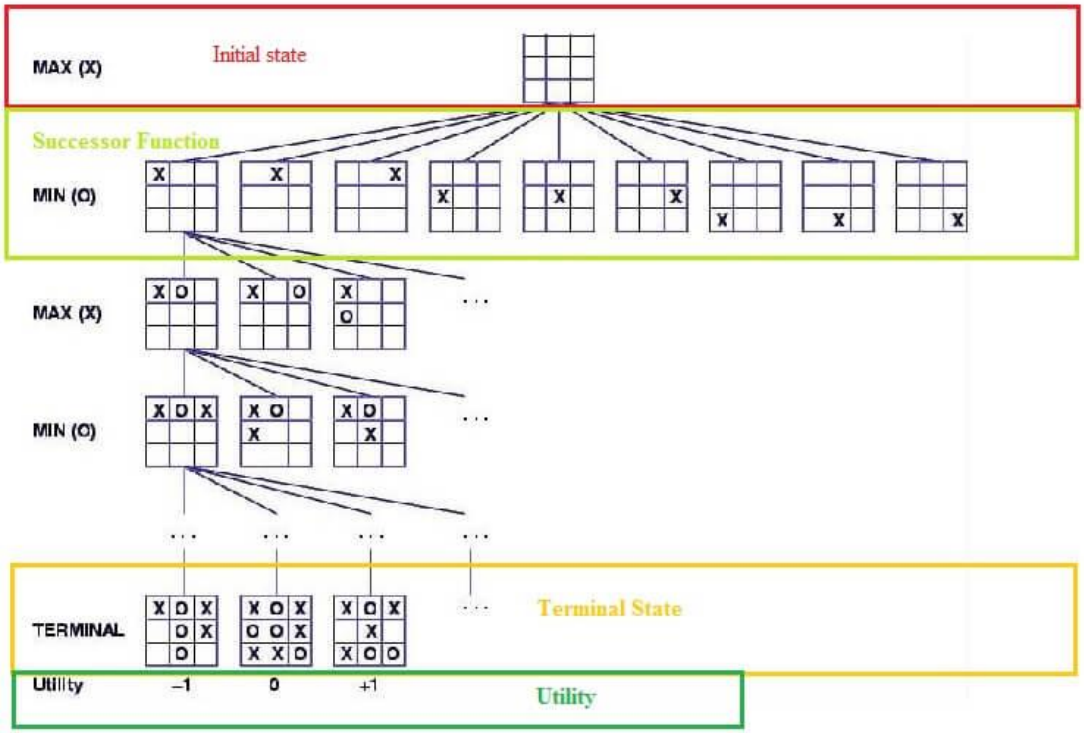
Algoritma nasıl çalışır?
Bir oyuna MIN ve MAX denilen iki oyuncu var. MAX oyuncusu mümkün olan en yüksek puanı almaya çalışır ve MIN mümkün olan en düşük puanı almaya çalışır, yani, MIN ve MAX birbirleriyle zıt davranmaya çalışırlar.

Minimax algoritmasının genel süreci aşağıdaki gibidir:
Adım 1: Öncelikle, oyunun mevcut durumundan başlayarak terminal durumlarına kadar oyun ağacının tamamını oluşturun. Oyun ağacı oyun tic-tac-toe için böyle görünüyor.



Tic Tac Toe

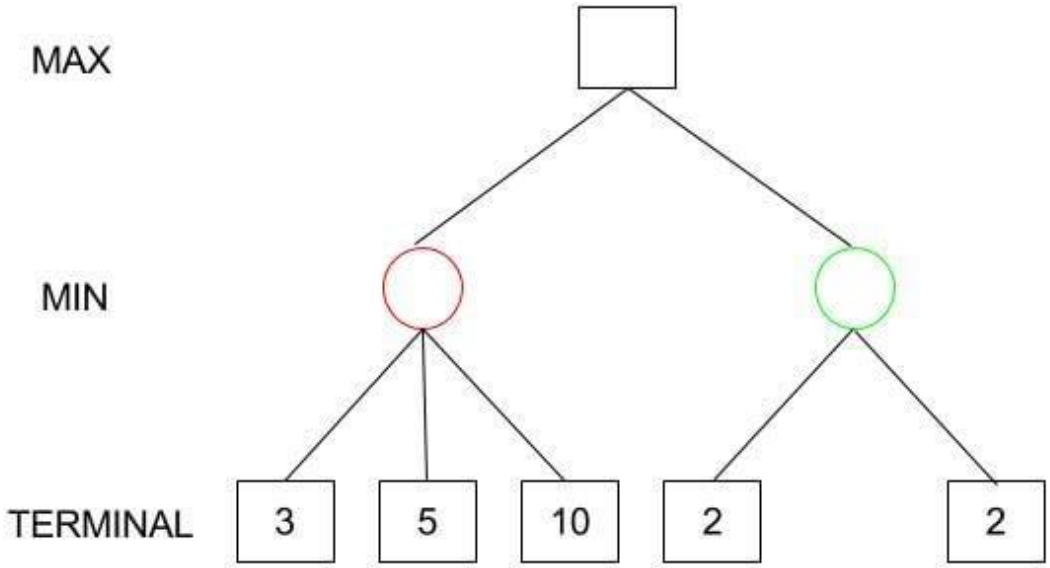
İlk durum, tahtanın boş olduğunu tanımlayan ilk katmandır ve MAX'ın oynama sırasıdır.
Halefi işlevi, olası tüm halefi hareketlerini listeler. Ağaçtaki tüm katmanlar için tanımlanmıştır.
Terminal Durumu ağacın son halini gösteren son katmandır, yani MAX oyuncunun kazanması, kaybetmesi veya rakibe bağlanması gibi.
Terminal durumlar için bu durumda yardımcı programlar, daha önce tartışıldığı gibi 1, 0 ve -1'dir ve diğer düğümlerin yardımcı programlarını da belirlemek için kullanılabilirler.



tic-tac-toe - minimax

Adım 2: Tüm terminal durumları için yardımcı değer elde etmek için yardımcı program işlevini uygulayın.

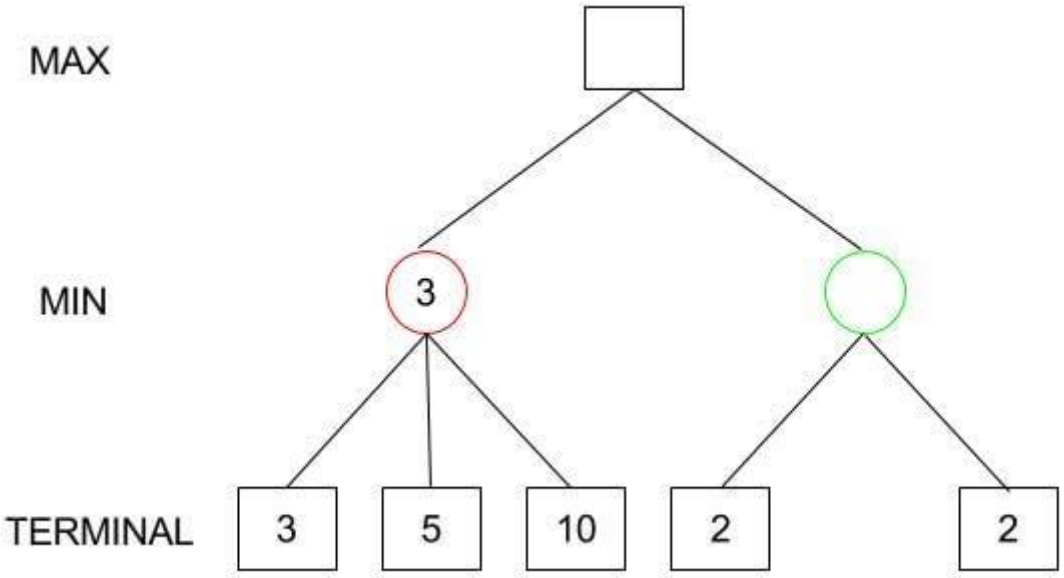
Adım 3: Terminal düğümlerin yardımcı programlarının yardımıyla daha yüksek düğümlerin yardımcı programlarını belirleyin. Örneğin, aşağıdaki şemada, meydanlarda yazılan terminal durumları için yardımcı programlara sahibiz.



Minimax

Terminalin üstündeki katmanın sol düğümü (kırmızı) için yardımcı programı hesaplayalım. MIN oyuncunun hamlesi olduğu için tüm hizmetlerin minimumunu

seçeceğiz. Bu durumda, kesinlikle 3 olduğunu bildiğimiz MIN {3, 5, 10} değerini değerlendirmek zorundayız. Bu nedenle, kırmızı düğümün faydası 3'tür.



Minimax

Adım 4: Ağacın köküne kadar her seferinde bir katman göz önüne alınarak yaprakların yardımıyla fayda değerlerini hesaplayın.
Adım 5: Sonunda, tüm yedeklenen değerler ağacın köküne, yani en üst noktaya ulaşır. Bu noktada, MAX en yüksek değeri seçmek zorundadır.
Örneğimizde, sadece 3 katmana sahibiz, bu yüzden hemen kökündeyiz, ancak gerçek oyunlarda, çok daha fazla katman ve düğüm olacaktır. Bu yüzden 3 olan MAX {3,2} 'yi değerlendirmek zorundayız.
Bu nedenle, MAX için en iyi açılış hareketi sol düğümdür (veya kırmızı olan). Bu hamleye minimax kararı denir ve rakibin de en aza indirmek için en iyi şekilde oynadığı varsayımı sonrasında aracı en üst seviyeye çıkarır.

Özetle:

Minimax Kararı = MAX {MIN {3,5,10}, MIN {2,2}}
= MAX {3,2}
= 3

```
function minimax(node, depth, maximizingPlayer)
        if depth = 0 or node is a terminal node
            return the utility of the node

        if maximizingPlayer
            bestValue: = ??
        for each child of node
            v: = minimax(child, depth ? 1, FALSE)
            bestValue: = max(bestValue, v)
        return bestValue

        else (* minimizing player *)
```

```
bestValue: = +?
for each child of node
      v: = minimax(child, depth ? 1, TRUE)
      bestValue: = min(bestValue, v)
return bestValue
```

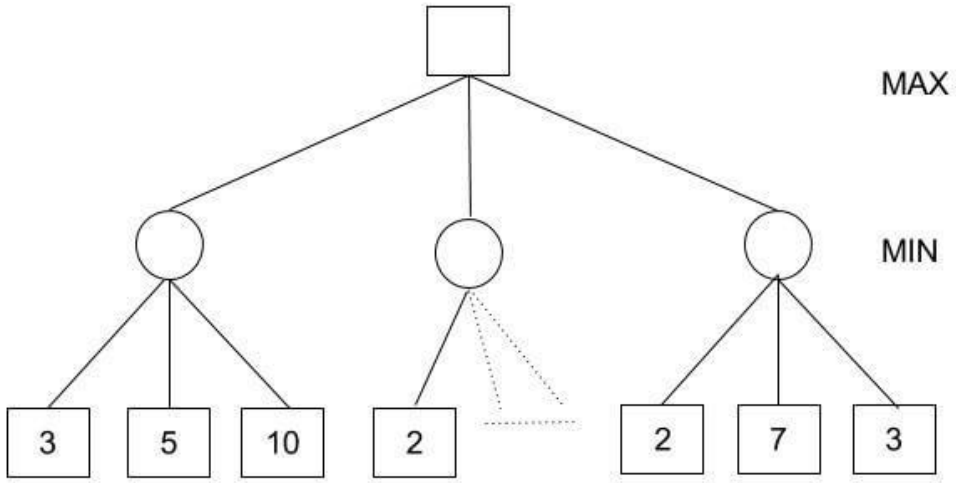
Optimizasyon

Oyun ağaçları, genel olarak, inşa etmek çok zaman alır ve sadece kısa sürede üretilebilecek basit oyunlar içindir.
Eğer varsa b yasal hamleler, yani b Her noktadaki düğümler ve ağacın maksimum derinliği mminimax algoritmasının zaman karmaşıklığı düzendedir bm (O (b)m)).
Bu durumu ortadan kaldırmak için, algoritmaya eklenebilecek birkaç optimizasyon vardır.
Neyse ki, oyun ağacının her düğümüne bakmadan gerçek minimax kararını bulmak mümkün. Dolayısıyla, düğümleri analiz etmeden ağaçtan kaldırıyoruz ve bu işleme budama denir.

Alfa-beta budama

Alfa-beta budamasını standart bir minimax algoritmasına uygularsak, standart olanla aynı hareketi döndürür, ancak nihai kararı etkilemeyen tüm düğümleri kaldırır.

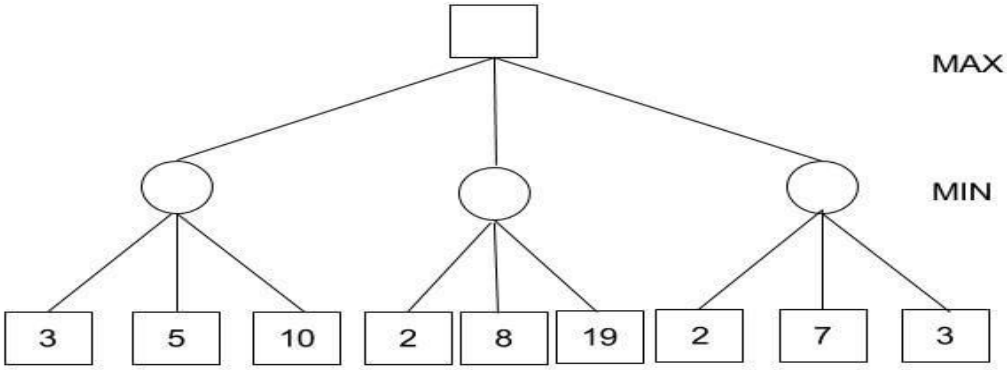
Diyelim ki şu oyun ağacına sahibiz:



Minimax

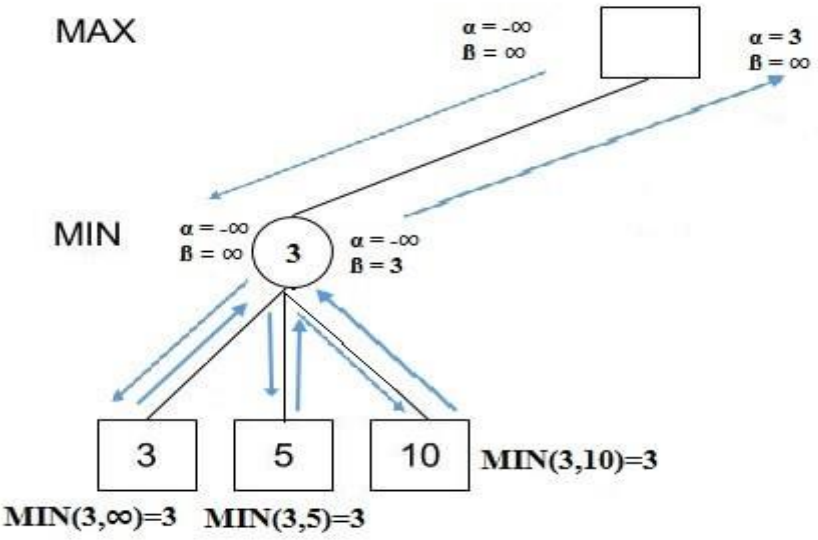
Bu durumda,
Minimax Kararı = MAX {MIN {3,5,10}, MIN {2, a, b}, MIN {2,7,3}}
= MAX {3, c, 2}
= 3
Şaşırdın!
Maksimum değeri eksik bir değerle nasıl hesaplayabiliriz? İşte hile. MIN {2, a, b} kesinlikle 2'den küçük veya ona eşit olacaktır, yani c <= 2 ve bu nedenle MAX {3, c, 2}, 3 olmalıdır.

Şimdi soru şu ki c'yi gerçekten hesaplamamız gerekiyor mu? Tabii ki değil.
Bu düğümlere bakmadan sonuca varabilirdik. Ve burası alfa-beta budamasının resmin içine girdiği yer.
Birkaç tanım:
Alfa: MAX oyuncu için şimdiye kadarki en iyi seçimdir. Burada mümkün olan en yüksek değeri elde etmek istiyoruz.
Beta: MIN için şu ana kadarki en iyi seçim ve mümkün olan en düşük değer olması gerekiyor.
Not: Her düğümün alfa ve beta değerlerini izlemesi gerekir. Alpha yalnızca MAX'ın sırası geldiğinde ve benzer şekilde beta yalnızca MIN'in şansı olduğunda güncellenebilir.
Alfa-beta budama nasıl çalışır?
Mümkün olan en kötü durum olarak alfa = -finity ve beta = sonsuzluğu başlat. Bir düğümü budama koşulu, alfa beta değerinden büyük veya ona eşit olduğunda meydana gelir.



Minimax

Alfa ve beta'nın başlangıç değerlerini kök dizinine atama ile başlayın ve alfa betadan daha küçük olduğu için bunu budamıyoruz.
Bu alfa ve beta değerlerini soldaki alt düğüme taşıyın. Ve şimdi terminal durumunun fayda değerinden, alpha ve be'nin değerlerini güncelleyeceğiz, bu yüzden beta değerini güncellememiz gerekmiyor. Yine, budama yapmayız çünkü durum aynı kalır. Benzer şekilde, üçüncü çocuk düğümü de. Daha sonra kök dizinine geri dönüyoruz, alpha = 3 olarak ayarlıyoruz, çünkü bu, alfa değerinin sahip olabileceği minimum değerdir.



Minimax

Şimdi, alfa = 3 ve beta = kökündeki sonsuzluk. Yani biz budamıyoruz. Bunu merkez düğüme taşıyan ve MIN {2, sonsuz} değerini hesaplayarak alfa = 3 ve beta = 2 elde ederiz.
İkinci ve üçüncü alt düğümleri budama çünkü alfa şimdi betadan büyüktür.
Kökteki alfa 3 olarak kalır çünkü 2'den büyüktür. Bunu en sağ alt düğüme taşıyan MIN (sonsuz, 2} = 2) değerini değerlendirin. Beta'yı 2'ye güncelleyin ve alfa 3 kalır.
İkinci ve üçüncü alt düğümleri budama çünkü alfa şimdi betadan büyüktür.
Dolayısıyla sırasıyla sol, orta ve sağ MIN düğümlerinde 3, 2, 2 alıyoruz. Ve MAX {3,2,2} 'yi hesaplarsak, 3 elde ederiz. Bu nedenle, dört yaprağı izlemeden minimax kararını doğru bir şekilde bulabiliriz.

```
evaluate (node, alpha, beta)
    if node is a leaf
      return the utility value of node
    if node is a minimizing node
      for each child of node
         beta = min (beta, evaluate (child, alpha, beta))
         if beta <= alpha
            return beta
         return beta
    if node is a maximizing node
      for each child of node
         alpha = max (alpha, evaluate (child, alpha, beta))
         if beta <= alpha
            return alpha
         return alpha
```

Sonuç

Oyunlar çok çekici ve oyun oynama programları yazmak belki de daha heyecan verici. Grand Prix yarışlarının otomobil endüstrisi için ne olduğu, oyun oynamak AI'dir.
Bir yarış arabasının engebeli bir yolda mükemmel şekilde çalışmasını beklemeyeceğimiz gibi, oyun oynama algoritmalarının her durum için mükemmel olmasını beklememeliyiz. Minimax algoritması da öyle. AI olması gereken her türlü bilgisayar oyunu için en iyi çözüm olmayabilir.

Ancak iyi bir uygulama verildiğinde, zorlu bir rakip oluşturabilir.

2. PROJE TASARIMI

2.1.GEREKSİNİM ANALİZİ

Kullanılacak dil ve platform

Python’ un geniş kütüphane havuzunun da bana çok yardımcı olacağını bildiğim için Projemi python ile VSCode (Visual Studio Code) üzerinde geliştirmeye karar verdim.

Arayüz

Projede tek kişi olduğumdan ve daha çok projenin back-end kısmına yoğunlaşmayı istediğim için arayüz konusuna çok vakit harcamak istemedim, bu sebepten basit ve anlaşılması kolay bir arayüz yapmaya karar verdim.
Ama ileride projeyi geliştirdikçe son haline iyi bir arayüz de yapmayı planlıyorum

Görsellik ve müzik

Proje bir oyun projesi olduğu için son kullanıcı yanlısı olması gerektiğini düşündüğümden Python’ un pygame kütüphanesi yardımı ile kolay anlaşılabilir bir arayüz ve oyun yapmayı planlıyorum.

Bunun yanı sıra oyunumda kullanmak üzere görseller, müzik ve efektleri, telif hakkı talep etmeyen sitelerden temin edebiliri.

Kendimin de görsel olarak bazı eklemeler yapabilirim.

Yapay zekâ

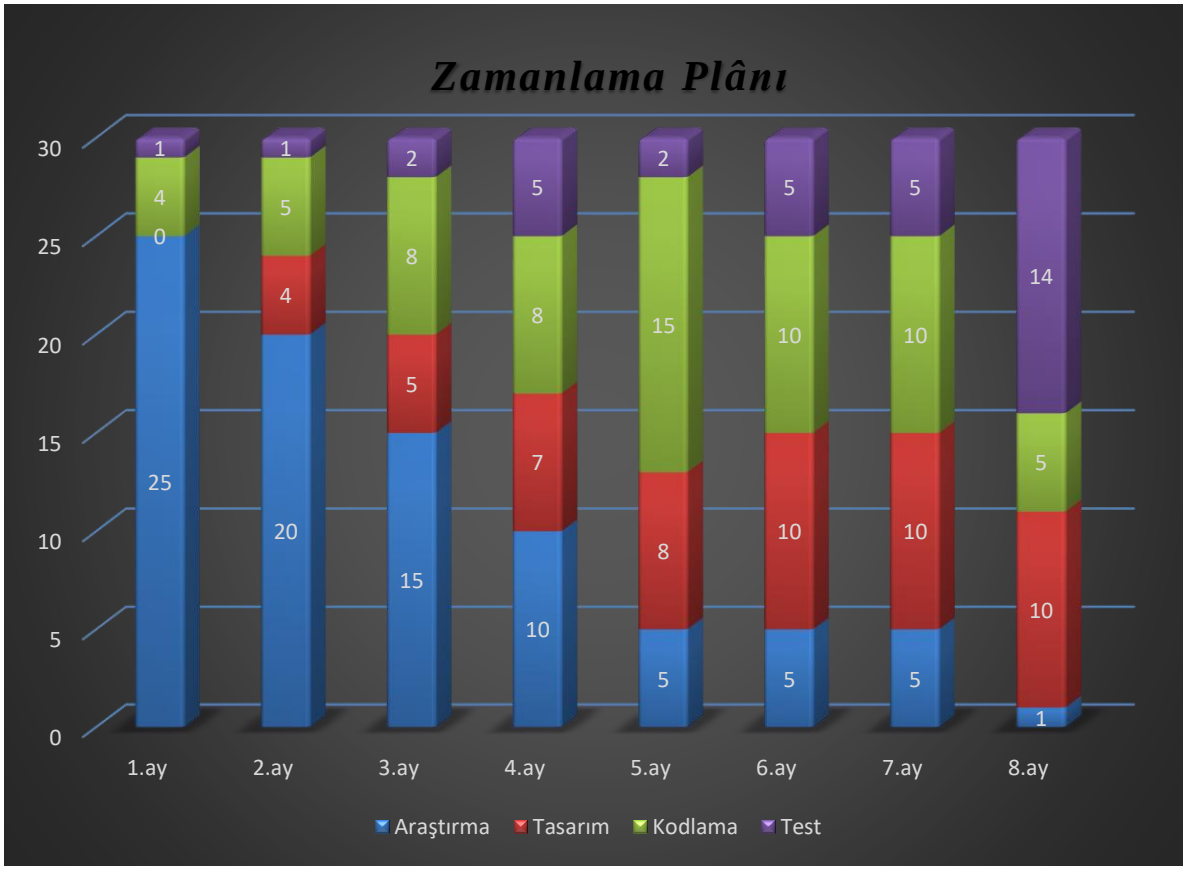
Oyun her ne kadar PvP (Person vs Person) oynanabilir olsa da projedeki asıl amaç yapay zekâya karşı bir oyun deneyimi sunmaktır.
Bu sebepten minimax algoritması kullanarak bir bot geliştirmeyi düşünüyorum.
Algoritmanın derinlik değişkenini kullanarak farklı zorluk seviyeleri de oluşturabilirim.
Ayrıca kendi yaptığım bir algoritmayı da farklı bir bot olarak ekleyebilirim.

Riskler

- Tek kişi olduğum için projenin yetişmemesi.
- Kullanıcının programı anlamaması.
- Yapay zekanın basit olması.

Çözüm Önerileri

- Projenin zamanında tamamlanması için sıkı çalışma
- Kullanıcı dostu bir program için kolay anlaşılır arayüz.
- Yapay zekânın iyi bir seviyede olabilmesi için kendi geliştirdiğim algoritmanın yanı sıra zaten kullanılan bir yapay zekâ algoritması eklemek.



2.2.MİMARİ TASARIM

1. 7X7'lik matristen oluşan bir Oyun tahtası fonksiyonu oluşturup oyunun oynanacağı tahta nesnesini oluştur.

2. Tahtayı manipüle etmek için boş_yerler() , taşı_bırak() ,tahta_doldumu() vs. kontrol fonksiyonları oluştur.

3. Oyunu görselleştirip testleri kolaylaştırmak için terminalde sembollerle göstericek tahtayı_yaz() fonksiyonu oluştur.

4. 2 oyuncudan biri kazanana kadar ya da oyun tahtasında olası hamle kalmayana kadar oyunun oynanacağı döngünün iskeletini oluştur.

5. Her iki oyuncunun manipülasyon fonksiyonları yardımıyla sırayla hamle yapabileceği ve her hamleden sonra tahtanın yazdırılacağı sıralı koşullar örgüsünü döngünün içine yerleştir.

6. Kolay Aİ botu basit bir random fonksiyonu ile 2. oyuncu yerine işle.

7. Normal Ai botu için tatayı inceleyip yapabileceği tüm hamleleri karşılaştırıp en iyi hamleyi yapacak bir algoritma yaz.

8. Zor(minmax) Aİ botu için kendi yaptığın algoritma yerine min max algoritmasını inceleyerek koda entegre et.

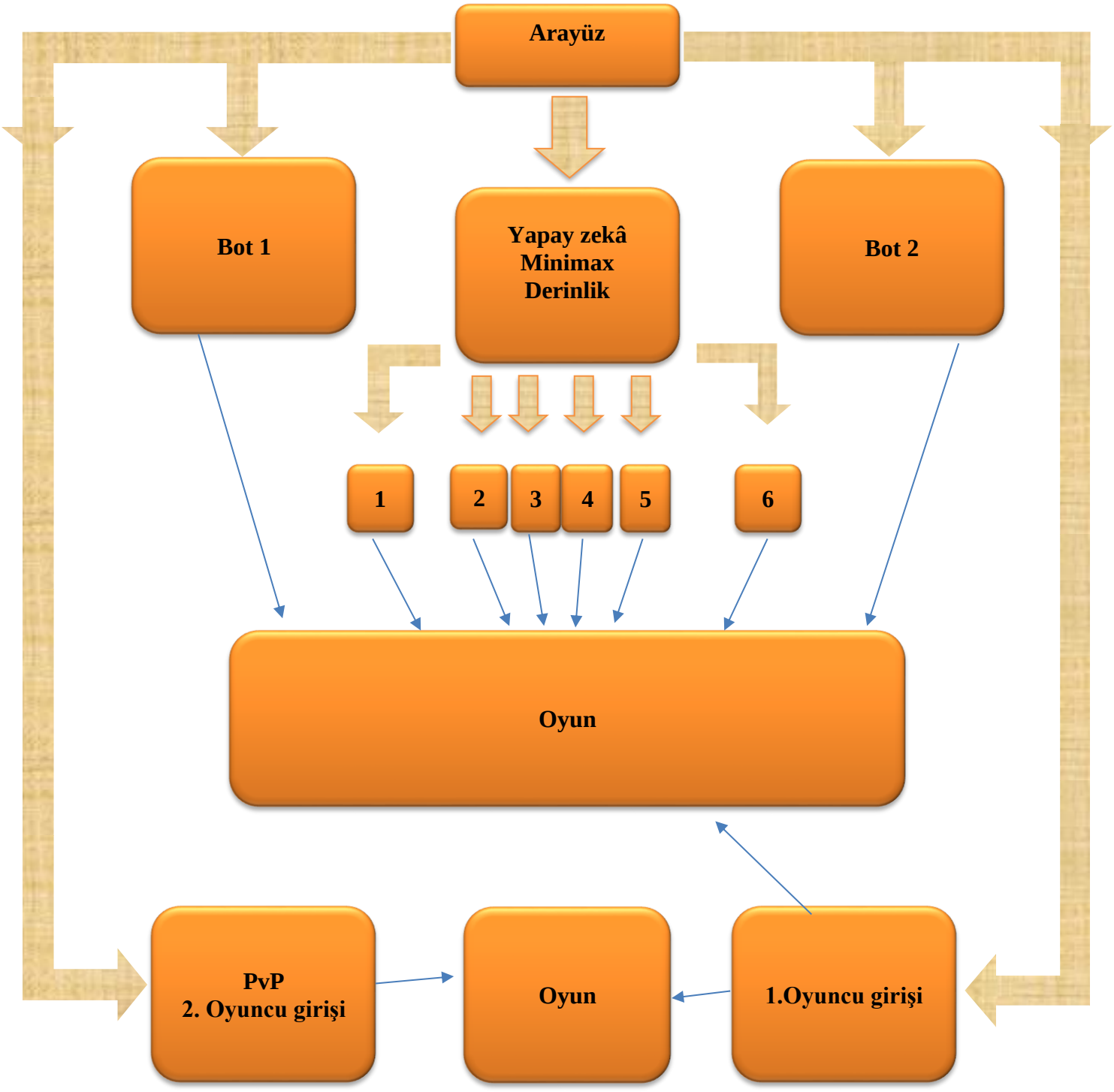
9. Oyunu ve Aİ botları test et.

10. Oyun ve botlar sorunsuz çalışıyorsa pygame kütüphanesi ile tahtayı yaz fonksiyonu yerine tahtayı çiz fonksiyunu oluşturarak oyunun görsel arayüzü olmasını sağla.

11. Görselleştirilmiş oyun sorunsuz çalışıyorsa Oyun modlarını birleştirip ses, zroluk düzeyi seçme, skor kaydı gibi özellikler ekleyip kullanıcı dostu tek bir program haline getir.

12. Programın son halinide test ettikten sonra executable bir dosya haline getir.

2.3.UML NESNE MODELİ



2.4.YAPILAN ÇALIŞMALAR

```
# Gerekli kütüphaneler

import numpy as np
import pygame
import sys
import math
import os
import random
from pygame.locals import *
```

```

#Muzik ve img dosyalarını dahil etmek için local dosya pathlari

filepath = os.path.abspath(__file__)
filedir = os.path.dirname(filepath)

music_path = os.path.join(filedir, "music/arka_plan_music.mp3")

efekt_path = os.path.join(filedir, "music/Efekt/Efekt")

img_path = os.path.join(filedir, "img/arka_plan_resim.png")

#Oyun tahtasinin satir ve sutun değişken atamalari

SATIR_SAYISI=6
SUTUN_SAYISI=7
```

```
#Projede kullanilan RGB renk atamalari

SIYAH=(0,0,0)
MAVI=(0,0,255)
KIRMIZI=(255,0,0)
BORDO=(128,0,0)
SARI=(255,255,0)
```

```
#Oyunda turu belirlemede kullanilacak degiskenler

OYUNCU1=0
OYUNCU2=1
AI=1

BOS=0
OYUNCU1_TASI=1
```

```
OYUNCU2_TASI=2
AI_TASI=2
```

```
#Oyun tahtasinin ve taslarin boyutlarini belirleyen atamalar

PENCERE_BOYU=4
KARE_BOYUT = 100
YARICAP = int(KARE_BOYUT/2-5)
genislik = SUTUN_SAYISI*KARE_BOYUT
yukseklik = (SATIR_SAYISI+1)*KARE_BOYUT
boyut = (genislik,yukseklik)
```

```
#pygame modulunu kullanmak için yapilan ayarlamalar

pygame.init()
ekran = pygame.display.set_mode(boyut)
pygame.display.set_caption("CONNECT 4")
pygame.display.update()
```

```
#Oyundaki yazilarin fontu

myfont = pygame.font.SysFont("monospace", 60, bold=True, italic=True)
```

```
#Oyun alaninin olusturulmasi

def oyun  alanı():
    tahta=np.zeros((SATIR_SAYISI,SUTUN_SAYISI))
    return tahta
```

```
#Oyun alanindaki bos yerlerin kontrolu

def yer_bosmu(tahta,sutun):
    return tahta[5][sutun]==0
```

```
#Tasin en alttaki bos satira aktarimi

def siradaki_bos_satir(tahta,sutun):
    for i in range(SATIR_SAYISI):
        if tahta[i][sutun]==0:
            return i
```

```
#Oyunu konsolda gorsel olarak yazdirma
```

```
def tahta_yaz(tahta):
    print(np.flip(tahta,0))
```

```
#Oyun alani taranarak kazanma koşullarinin olusup olusmadiginin testi

def kazandinmi(tahta, tas):
    #dikey kazanma koşullari
    for c in range(SUTUN_SAYISI-3):
        for r in range(SATIR_SAYISI):
            if tahta[r][c] == tas and tahta[r][c+1] == tas and tahta[r][c+2] == tas and tahta[r][c+3] == tas:
                return True

     #yatay kazanma koşullari
    for c in range(SUTUN_SAYISI):
        for r in range(SATIR_SAYISI-3):
            if tahta[r][c] == tas and tahta[r+1][c] == tas and tahta[r+2][c] == tas and tahta[r+3][c] == tas:
                return True

    #capraz kazanma koşullari
    for c in range(SUTUN_SAYISI-3):
        for r in range(SATIR_SAYISI-3):
            if tahta[r][c] == tas and tahta[r+1][c+1] == tas and tahta[r+2][c+2] == tas and tahta[r+3][c+3] == ta
s:
                return True

    #capraz kazanma koşullari
    for c in range(SUTUN_SAYISI-3):
        for r in range(3, SATIR_SAYISI):
            if tahta[r][c] == tas and tahta[r-1][c+1] == tas and tahta[r-2][c+2] == tas and tahta[r-3][c+3] == tas:
                return True
```

```
#Oyun alanının gorsel olarak açilir pencereye aktarilmasi

def tahta_ciz(tahta):
    for c in range(SUTUN_SAYISI):
        for r in range(SATIR_SAYISI):
            pygame.draw.rect(ekran, MAVI, (c*KARE_BOYUT, r*KARE_BOYUT+KARE_BOYUT, KARE
_BOYUT, KARE_BOYUT))
            pygame.draw.circle(ekran, SIYAH, (int(c*KARE_BOYUT+KARE_BOYUT/2), int(r*KARE_BO
YUT+KARE_BOYUT+KARE_BOYUT/2)), YARICAP)

    for c in range(SUTUN_SAYISI):
        for r in range(SATIR_SAYISI):
            if tahta[r][c] == OYUNCU1_TASI:
```

```
            pygame.draw.circle(ekran, KIRMIZI, (int(c*KARE_BOYUT+KARE_BOYUT/2), yukseklik-
int(r*KARE_BOYUT+KARE_BOYUT/2)), YARICAP)
        elif tahta[r][c] == AI_TASI:
            pygame.draw.circle(ekran, SARI, (int(c*KARE_BOYUT+KARE_BOYUT/2), yukseklik-
int(r*KARE_BOYUT+KARE_BOYUT/2)), YARICAP)
    pygame.display.update()
```

```
#Oyuncunun sectigi sutuna tasın yerlestirilmesi

def tasi_birak(tahta,satir,sutun,tas):
    tahta[satir][sutun]=tas
    tahta_ciz(tahta)
```

```
#Yapay zekanin olasi hamleleri hesaplamasi icin geceici tas birakimi

def gecici_tasi_birak(tahta,satir,sutun,tas):
    tahta[satir][sutun]=tas
```

```
#Yapay zekanin yapabilecegi hamleler arasinda en iyisini secebilmesi icin puanlama sistemi

def puanlama(pencere,tas):
    k_tas = OYUNCU1_TASI
    if tas == OYUNCU1_TASI:
        k_tas = AI_TASI
    skor=0
    if pencere.count(tas) == 4:
        skor += 100
    elif pencere.count(tas) == 3 and pencere.count(BOS) == 1:
        skor += 5
    elif pencere.count(tas) == 2 and pencere.count(BOS) == 2:
        skor += 2
    if pencere.count(k_tas) == 3 and pencere.count(BOS) == 1:
        skor -= 4

    return skor
```

```
#Yapay zekanin tahtayi ve o anki durumu gozlemleyerek ilerisi icin yapabilecegi en iyi hamleyi hesaplama
si

def gozlem(tahta,tas):
    skor=0

    #Merkez sutun gözlemi
    merkez_dizin = [int(i) for i in list(tahta[:, SUTUN_SAYISI//2])]
```

```
    merkez_sayac = merkez_dizin.count(tas)
    skor += merkez_sayac * 3

    #Yatay gözlem
    for r in range(SATIR_SAYISI):
        satir_dizisi=[int(i)for i in list(tahta[r,:])]
        for c in range(SUTUN_SAYISI-3):
            pencere = satir_dizisi[c:c+PENCERE_BOYU]
            skor+=puanlama(pencere,tas)

    #Dikey gözlem
    for c in range(SUTUN_SAYISI):
        sutun_dizisi=[int(i)for i in list(tahta[:,c])]
        for c in range(SATIR_SAYISI-3):
            pencere = sutun_dizisi[r:r+PENCERE_BOYU]
            skor+=puanlama(pencere,tas)

    #Çapraz gözlem
    for r in range(SATIR_SAYISI-3):
        for c in range(SUTUN_SAYISI-3):
            pencere = [tahta[r+i][c+i] for i in range(PENCERE_BOYU)]
            skor+=puanlama(pencere,tas)

    for r in range(SATIR_SAYISI-3):
        for c in range(SUTUN_SAYISI-3):
            pencere = [tahta[r+3-i][c-i] for i in range(PENCERE_BOYU)]
            if pencere.count(tas) == 4:
                skor +=100
            elif pencere.count(tas) == 3 and pencere.count(BOS)==1:
                skor+=10

    return skor
```

```
# Yapay zekanin oynaya bilecegi bos yerlerin bulunmasi

def bos_bul(tahta):
    bos_yerler = []
    for col in range (SUTUN_SAYISI):
        if yer_bosmu(tahta,col):
            bos_yerler.append(col)

    return bos_yerler
```

```
#Yapay zekanin en iyi hamlesini bulmasi icin bos yerleri karsilastirmasi

def en_iyi_hamle(tahta,tas):
    bos_yerler=bos_bul(tahta)
```

```
    en_iyi_skor =-10000
    en_iyi_sutun =random.choice(bos_yerler)
    for col in bos_yerler:
        row = siradaki_bos_satir(tahta,col)
        gecici_tahta = tahta.copy()
        tasi_birak(gecici_tahta,row,col,tas)
        skor = gozlem(gecici_tahta,tas)
        if skor > en_iyi_skor:
            en_iyi_skor = skor
            en_iyi_sutun = col

    return en_iyi_sutun
```

```
#Oyunun bitip bitmediginin kontrolu

def oyun_sonu_mu(tahta):
    return kazandinmi(tahta,OYUNCU1_TASI) or kazandinmi(tahta,AI_TASI) or len(bos_bul(tahta)) == 0
```

```
#Yapay zekanin tahtayi gozlemleyerek yapacagi hamleye karar vermesi icin min-max algoritmasi

def minimax(tahta, derinlik, alpha, beta, maximazingplayer):
    bos_yerler = bos_bul(tahta)
    terminal_dugum = oyun_sonu_mu(tahta)

    #Yapılacak hamle kalmadıysa
    if derinlik == 0 or terminal_dugum:
        if terminal_dugum:
            if kazandinmi(tahta,AI_TASI):
                return (None,100000000000000000000)
            elif kazandinmi(tahta,OYUNCU1_TASI):

                return (None,-100000000000000000)
            else: #Yapılacak hamle kalmadı
                return (None,0)
        else: #Derinlik sıfır
            return (None,gozlem(tahta,AI_TASI))

    #Rakibin o anki durumu ve maximum yapabilecegi hamlelerin karsılastırılması
    if maximazingplayer:
        deger = -math.inf
        sutun = random.choice(bos_yerler)
        for col in bos_yerler:
            row = siradaki_bos_satir(tahta,col)
            gecici = tahta.copy()
            gecici_tasi_birak(gecici,row,col,AI_TASI)
            yeni_skor = minimax(gecici,derinlik-1, alpha, beta, False)[1]
```

```
            if yeni_skor > deger:
                deger = yeni_skor
                sutun = col

            alpha = max(alpha, deger)
            if alpha >=beta:
                break

        return sutun,deger

    else:   #Minimazing
        deger = math.inf
        sutun = random.choice(bos_yerler)
        for col in bos_yerler:
            row = siradaki_bos_satir(tahta,col)
            gecici = tahta.copy()
            gecici_tasi_birak(gecici,row,col,OYUNCU1_TASI)
            yeni_skor = minimax(gecici,derinlik-1, alpha, beta, True)[1]
            if yeni_skor < deger:
                deger = yeni_skor
                sutun = col

            beta = min(beta, deger)
            if alpha >=beta:
                break

        return sutun,deger
```

```
#Arayüze yazı yazdırma fonksiyonu

def metin_yaz(msj,font,renk,ekran,x,y):
    obj = font.render(msj,1,renk)
    ekran.blit(obj,(x,y))
```

```
#Menu arayuzu

click = False

def menu():

    # Ana muzik yuklemesi
    pygame.mixer.music.load(music_path)
    pygame.mixer.music.play(-1,0.0)

    while True:

        #Efekt ve arka plan resim yuklemesi
```

```
    efekt = efekt_path
    rand = random.randint(1,10)
    efekt += str(rand)+".wav"
    arka_plan_resim = pygame.image.load(img_path).convert()
    ekran.blit(arka_plan_resim, (0, 0))

    #Pygame ile oyuncu hareketlerinin alinmasi
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == MOUSEBUTTONDOWN:
            if event.button == 1:
                click = True


    metin_yaz("CONNECT4", myfont, MAVI, ekran, 150, 10)

    #Mouse un anlik pozisyonu
    mouse_x , mouse_y = pygame.mouse.get_pos()

    #Butonlarin olusturulmasi
    buton1 = pygame.Rect(50,100,200,50)
    buton2 = pygame.Rect(50,200,200,50)
    buton3 = pygame.Rect(50,300,200,50)
    buton4 = pygame.Rect(10,500,50,50)
    buton5 = pygame.Rect(100,500,50,50)
    buton6 = pygame.Rect(200,500,50,50)
    buton7 = pygame.Rect(10,600,50,50)
    buton8 = pygame.Rect(100,600,50,50)
    buton9 = pygame.Rect(200,600,50,50)


    #Buton aksiyonlari ile oyun baslatilmasi
    if buton1.collidepoint(mouse_x, mouse_y):
        if click:
            pygame.mixer.music.load(efekt)
            pygame.mixer.music.play(0,0.0)
            oyun(0)
            pygame.mixer.music.load(music_path)
            pygame.mixer.music.play(-1,0.0)

    if buton2.collidepoint(mouse_x, mouse_y):
        if click:
            pygame.mixer.music.load(efekt)
            pygame.mixer.music.play(0,0.0)
            oyun(1)
            pygame.mixer.music.load(music_path)
            pygame.mixer.music.play(-1,0.0)
```

```
if buton3.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(2)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)

if buton4.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(3,1)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)

if buton5.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(3,2)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)

if buton6.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(3,3)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)

if buton7.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(3,4)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)

if buton8.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(3,5)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)
```

```
if buton9.collidepoint(mouse_x, mouse_y):
    if click:
        pygame.mixer.music.load(efekt)
        pygame.mixer.music.play(0,0.0)
        oyun(3,6)
        pygame.mixer.music.load(music_path)
        pygame.mixer.music.play(-1,0.0)



#Hareketli efektleri
if 50 < mouse_x < 50+200 and 100 < mouse_y < 100+50:
    pygame.draw.rect(ekran,BORDO,buton1)
else:
    pygame.draw.rect(ekran,KIRMIZI,buton1)

metin_yaz("PVP",myfont,MAVI,ekran,55,90)

if 50 < mouse_x < 50+200 and 200 < mouse_y < 200+50:
    pygame.draw.rect(ekran,BORDO,buton2)
else:
    pygame.draw.rect(ekran,KIRMIZI,buton2)

metin_yaz("BOT 1",myfont,MAVI,ekran,55,190)

if 50 < mouse_x < 50+200 and 300 < mouse_y < 300+50:
    pygame.draw.rect(ekran,BORDO,buton3)
else:
    pygame.draw.rect(ekran,KIRMIZI,buton3)

metin_yaz("BOT 2",myfont,MAVI,ekran,55,290)


metin_yaz("Yapay  ", myfont, SARI, ekran, 30, 350)
metin_yaz("Zeka ", myfont, SARI, ekran, 30, 420)


if 10 < mouse_x < 10+50 and 500 < mouse_y < 500+50:
    pygame.draw.rect(ekran,BORDO,buton4)
else:
    pygame.draw.rect(ekran,KIRMIZI,buton4)

metin_yaz("1",myfont,MAVI,ekran,5,490)

if 100 < mouse_x < 100+50 and 500 < mouse_y < 500+50:
    pygame.draw.rect(ekran,BORDO,buton5)
else:
    pygame.draw.rect(ekran,KIRMIZI,buton5)
```

```
    metin_yaz("2",myfont,MAVI,ekran,105,490)

    if 200 < mouse_x < 200 + 50 and 500 < mouse_y < 500+50:
        pygame.draw.rect(ekran,BORDO,buton6)
    else:
        pygame.draw.rect(ekran,KIRMIZI,buton6)

    metin_yaz("3",myfont,MAVI,ekran,205,490)

    if 10 < mouse_x < 10+50 and 600 < mouse_y < 600+50:
        pygame.draw.rect(ekran,BORDO,buton7)
    else:
        pygame.draw.rect(ekran,KIRMIZI,buton7)

    metin_yaz("4",myfont,MAVI,ekran,5,590)

    if 100 < mouse_x < 100+50 and 600 < mouse_y < 600+50:
        pygame.draw.rect(ekran,BORDO,buton8)
    else:
        pygame.draw.rect(ekran,KIRMIZI,buton8)

    metin_yaz("5",myfont,MAVI,ekran,105,590)

    if 200 < mouse_x < 200+50 and 600 < mouse_y < 600+50:
        pygame.draw.rect(ekran,BORDO,buton9)
    else:
        pygame.draw.rect(ekran,KIRMIZI,buton9)

    metin_yaz("6",myfont,MAVI,ekran,205,590)

    click = False

    #Ekran update i
    pygame.display.update()
```

```
#Oyun arayuzu

def oyun(seviye,derinlik=1):
    #Oyun tahtasi olusturma
    tahta = oyun  alanı()
    tahta_ciz(tahta)
    #tahta_yaz(tahta)

    #Ilk atamalar
    oyun_sonu = False
    tur=random.randint(OYUNCU1,AI)
```

```
#Oyun dongusu
while not oyun_sonu:

    #Oyun efektlerinin yuklenmesi
    efekt1 = efekt_path
    rand = random.randint(1,10)
    efekt1 += str(rand)+".wav"

    # Oyuncu 1 Girisi
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEMOTION:
            pygame.draw.rect(ekran, SIYAH, (0,0, genislik, KARE_BOYUT))
            posx = event.pos[0]
            if tur == OYUNCU1 and not oyun_sonu:
                pygame.draw.circle(ekran, KIRMIZI, (posx, int(KARE_BOYUT/2)), YARICAP)
            if seviye == 0 and not tur == OYUNCU1 and not oyun_sonu:
                pygame.draw.circle(ekran, SARI, (posx, int(KARE_BOYUT/2)), YARICAP)


        if event.type == pygame.MOUSEBUTTONDOWN:
            pygame.draw.rect(ekran, SIYAH, (0,0, genislik, KARE_BOYUT))
            if tur == OYUNCU1 and not oyun_sonu:

                posx = event.pos[0]
                col = int(math.floor(posx/KARE_BOYUT))

                if yer_bosmu(tahta, col):
                    row = siradaki_bos_satir(tahta, col)
                    tasi_birak(tahta, row, col, OYUNCU1_TASI)

                    pygame.mixer.music.load(efekt1)
                    pygame.mixer.music.play(0,0.0)

                    if kazandinmi(tahta, OYUNCU1_TASI):
                        metin_yaz("Oyuncu1 Kazandı!!   ",myfont,KIRMIZI,ekran,40,10)
                        oyun_sonu = True
                tur += 1
                tur %= 2


            # Oyuncu 2 Girisi
            elif seviye == 0 and not tur == OYUNCU1 and not oyun_sonu:

                posx = event.pos[0]
```

```
            col = int(math.floor(posx/KARE_BOYUT))

            if yer_bosmu(tahta, col):
                row = siradaki_bos_satir(tahta, col)
                tasi_birak(tahta, row, col, OYUNCU2_TASI)


                pygame.mixer.music.load(efekt1)
                pygame.mixer.music.play(0,0.0)

                if kazandinmi(tahta, OYUNCU2_TASI):
                    metin_yaz("Oyuncu2 Kazandı!!   ",myfont,SARI,ekran,40,10)
                    oyun_sonu = True
            tur += 1
            tur %= 2

    # Bot Girisleri
    if not seviye ==0:

        if tur == AI and not oyun_sonu:

            #Bot1 girisi
            if seviye == 1:
                pygame.time.wait(2000)
                col = random.randint(0,SUTUN_SAYISI-1)

            #Bot2 girisi
            elif seviye == 2:
                pygame.time.wait(2000)
                col = en_iyi_hamle(tahta,AI_TASI)

            #Yapay zeka girisi
            elif seviye == 3:
                col,minimax_skor = minimax(tahta, derinlik, -math.inf , math.inf , True)

            #Bot hamle belirlemesi
            if yer_bosmu(tahta, col):
                row = siradaki_bos_satir(tahta, col)
                tasi_birak(tahta, row, col, AI_TASI)

                efekt1 = efekt_path
                rand = random.randint(1,10)
                efekt1 += str(rand)+".wav"
                pygame.mixer.music.load(efekt1)
                pygame.mixer.music.play(0,0.0)

                if kazandinmi(tahta, AI_TASI):
                    metin_yaz("AI Kazandı!!   ",myfont,SARI,ekran,40,10)
```

```
                    oyun_sonu = True
                tur += 1
                tur %= 2

        #Oyun sonu kontrolu
        if bos_bul(tahta) == [] and not oyun_sonu:
            msj = myfont.render("BERABERE!!", 1, MAVI)
            ekran.blit(msj, (40,10))
            oyun_sonu = True


        #tahta_yaz(tahta)
        tahta_ciz(tahta)


        if oyun_sonu:
            pygame.time.wait(3000)
```

```
#Menu cagirimi
menu()
```

3.Kaynakça

- https://bilginc.com/tr/blog/158/python-nedir-python-hakkinda-hersey
- https://docs.python.org/3/tutorial/
- https://www.w3schools.com/python/default.asp
- https://pythondunyasi.com/pygame-nedir/
- https://www.w3schools.com/graphics/game_score.asp
- https://pythondunyasi.com/yapay-zeka-gelecegi-hakkinda-soylenen-sozleri/
- https://blog.niximera.com/minimax-algoritmasi/#:~:text=Minimax%20algoritmas%C4%B1%20nedir%3F,oyunculu%20oyunlar%20gibi%20oyunlarda%20kullan%C4%B1l%C4%B1r.
- https://www.thesprucecrafts.com/how-to-win-at-connect-four-basic-strategy-tips-412539
- https://github.com/sahlamba/connect4/blob/master/resources/Pseudo-Code%20--%20Connect%204.txt
- https://www.youtube.com/watch?v=fInYh90YMJU&list=RDCMUCPme28sMOcWS50CgtTWUZIw&start_radio=1&t=4
- https://en.wikipedia.org/wiki/Minimax
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
- https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning
- https://www.youtube.com/watch?v=Ntu8nNBL28o
- https://www.youtube.com/watch?v=trKjYdBASyQhttps://www.youtube.com/watch?v=l-hh51ncgDI

Müzikler ve Efektler

- https://www.youtube.com/watch?v=y7kvGqiJC4g
- https://www.playonloop.com/

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Var olan bir algoritmayı kendi projemde kullandım.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Projeme zaten var olan minimax algoritmasını entegre ettim buna alternatif olarak kendimde buna benzer bir algoritma geliştirdim.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Python ve algoritma geliştirme bilgilerimi kullandım.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

Projemde gerekli gördüğüm mühendislik standardı yok.

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

a) Ekonomi

Projemi kullanıcılara ulaştıracak olan platformlarda yer almasını sağlamak ve bilinirliğini arttırmak için reklamların yapılması ekonomik bir maliyet getirecektir.

b) Çevre sorunları:

Projem sadece yazılımdan oluştuğu için herhangi bir çevre sorunu bulunmamaktadır

c) Sürdürülebilirlik:

Projemi öncelikle kullanıcıların beğenisini kazanmalı ve bir kullanıcı kitlesi oluşturmalıdır. Eğer bunlar gerçekleşirse projem her zaman geliştirmelere açık olacağından sürdürülebilir olacaktır.

d) Üretile bilirlik:

Proje belirli bir düzeye ulaştıktan sonra grafiksel ve ek uygulamalarla elden geçirilmesi ile birçok platform üzerinden kullanıcılara sunulabilir

e) Etik:

Projem kullanıcıların gerekli ahlaki ve etik kurallara uyması durumunda sadece kişisel verinin korunumu zorunluluğunu getirmektedir.

f) Sağlık:

Bilgisayar başında fazla süre geçirileceği için bazı sağlık problemlerine sebep olabilir. Ancak bu tamamen kullanıcıların sorumluluğundadır.

g) Güvenlik:

Projede korsan ürünler sebebi ile güvenlik açığı olabilir.

h) Sosyal ve politik sorunlar:

Projemde sosyal ve politik sorunlar bulunmamaktadır.