

**YAPAY ZEKA PROJESİ**

**YAPAY ZEKA ALGORİTMALARIYLA PONG OYUNU**

**POUYA YAVARI**

**13011907**

**PROJENİN AMACI:**

Projenin amacı tek paddle’dan oluşan bir Pong oyununu bir yapay zeka algoritmasına oynatmak ve belli bir puana varmaktır. Top her paddle’a değdiği zaman puan 1 artacaktır ve top paddle’i geçtiği zaman puan sıfırlanacak ve oyun en baştan başlayacaktır.

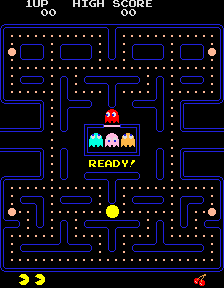
**PROJENİN DETAYLARI VE SEÇİLEN YÖNTEMLER:**

Bu projede oyunu oynamak için genetik algoritma kullanılıyor ve algoritma oyundan herhangi bir bigi çekmeden ve sadece bilgisayar ekranına bakarak ve ekran görüntülerinden gerekli bilgileri çıkararak bulunduğu durumu tahmin edip onagöre davranış sergiliyor.

Proje iki dilde yazılmıştır, oyunu yazmak için C++ ve SFML kütüphanesi kullanılmıştır ve genetik algoritmayı geliştirmek için Python dili kullanılmıştır. Python da kullanılan kütüphaneler, pyscreenshot, keyboard, time, numpy ve CV2’dir.

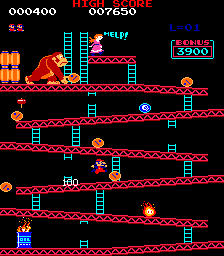
**NEDEN EKRANI OKUYARAK?**

Klasik bilgisayar oyunlarının çoğu (PONG, Space Invaders, Pac Man, Donky Kong, Tetris ve...) karmaşık grafikleri olmayan ve siyah arkapilan da oynanan oyunlar. Bu proje tasarlanırken önemli amaçlarından biri bu oyunların tamamını oynayabilecek tek bir algoritma geliştirmek. Kodun şimdiki hali sadece tek bir oyunu oynamak üzerine odaklanmış halde ama kodda değişiklikler yaparak ve bazı yöntemler geliştirilerek, bu algoritma ekrana bakıp yukarda söylenilen oyunlardan herhangi birini oğrenmeyi başarabilir ve oynamayı başarabilir (projenin bu yönde geliştirilmesine devam edilecektir).

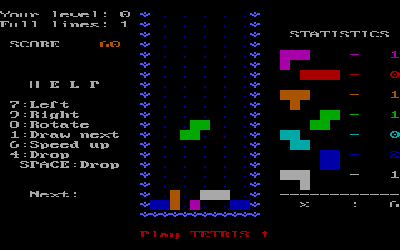




1.a: Space Invaders 1.b: Pac Man



1.c: Donkey Kong



1.d: Tetris

**KULLANILAN PONG CLONE’U HAKKIND:**

Kullanılan pong oyunu 550x600 pixel büyüklüğünde bir siyah ekranda oynanan, tek oyunculu bir pong oyunudur. Bu oyun tek paddle yardımı ile ve tek topla oynanıyor. Bu oyunun ekranında sol üst tarafta oyuncunun skor ve can bilgileri yazılmaktadır. Bu bigilerin tamamı DS-DIDI.ttf fontuyla yazılmaktadır. Top 10x10 pixellik bir top ve paddle 50x5 lik dir. Topun çıkabildiği üst limit oyun ekranının yüksekliğinin 100 pixel altıdır ve bu topun skor ve can bilgilerinin içine girip okunmalarını zorlaştırmaması için yapılmıştır. Ayrıca paddle in ekranın alt limitiyle olan mesafesi 15 pixeldir. Oyun her yeniden başlatıldığında topun X koordinatı için random bir sayı saçiliyor ve (X , 100) koordinatı, topun başlangıç koordinatı olarak belirleniyor. Top her ekranın dışına çıktığı zaman, yani paddle topu yakalayamadığında, top bu başlangıç koordinatına geri koyuluyor. Top her başlangıç koordinatına geri koyulduğu zaman yeniden harekete başlaması için ‘space’ tuşuna basılmalıdır. Ayrıca paddle’i sağa hareket ettirmek için klavyenin sağ tuşuna ve sola hareket ettirmek için klavyenin sol tuşuna basılmalıdır. Top her ekranın üst limitine isabet ettiğinde geri dönecektir ve oyuncunun skorlarına bir poan eklenecektir ve top her ekranın en alt kısmından ekrandan çıktığı zaman oyuncunun canı bir eksilecektir, eğer oyuncunun canı 0’a düşerse oyun yeniden başlayacaktır, bu oyuncunun skorunun sıfırlanması ve canının başlangıçta olduğu değere getirilmesi demektir. Ayrıca top her ekrandan çıktığı zaman paddle ekranın tam ortasına geri getirtilecektir. Bu oyunda başlangıç can sayısı 2 ve başlangıç skor sayısı ise tahmin edileceği gibi 0’dır.



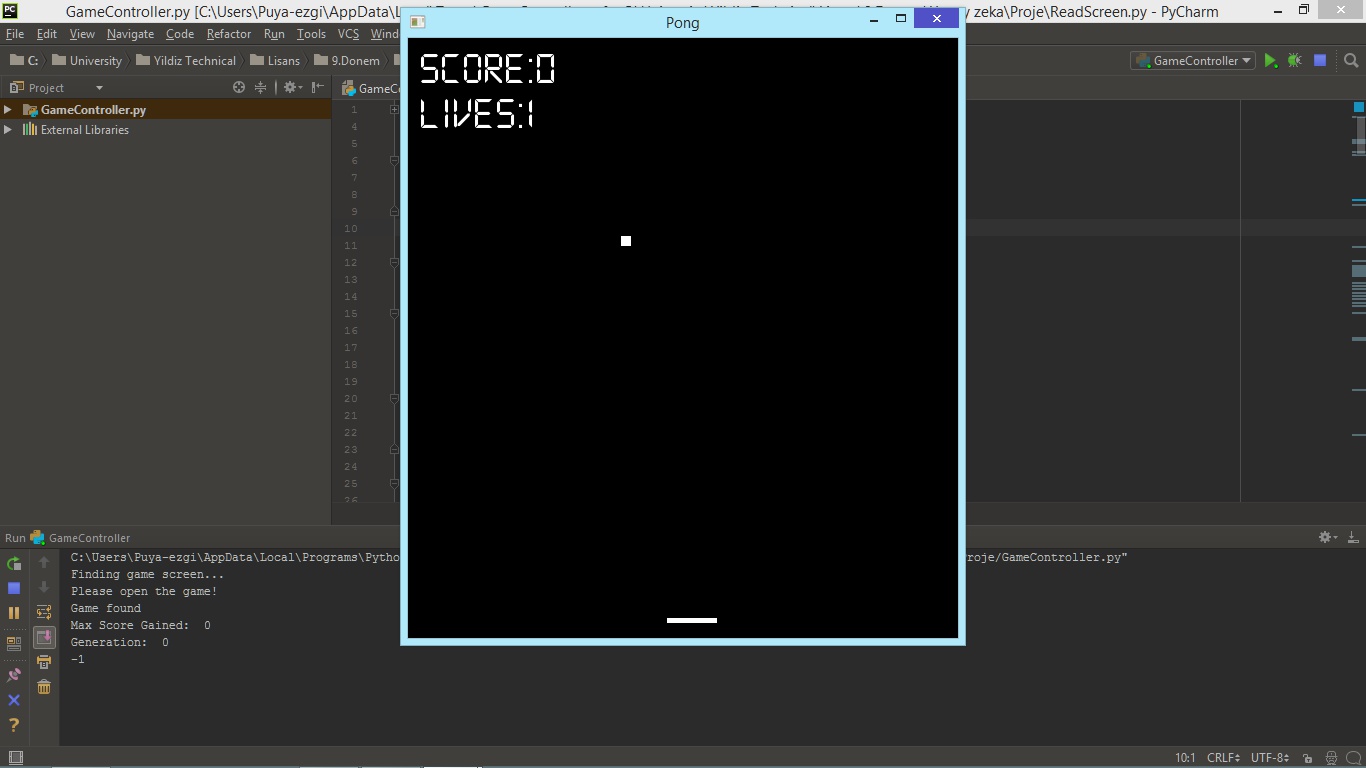
2: Oyun ekranının bir örnek görüntüsü

**EKRAN OKUMADA KULLANILAN YÖNTEM:**

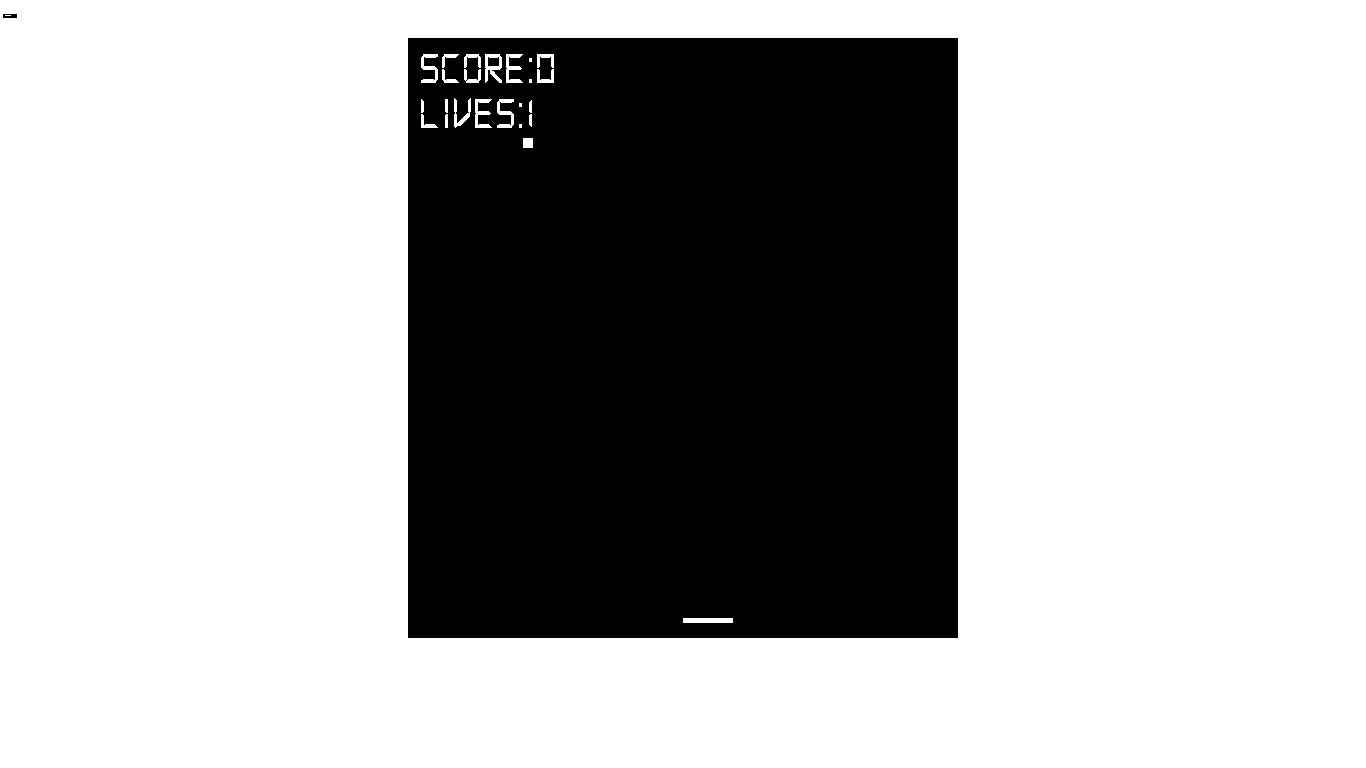
Programın oyunu oynayabilmesi için ilk başta oyun ekranını görmesi gerekmektedir, yani oyun ekranı açık ve tamamen görülebilir halde olmalıdır.

Program oyun ekranını bulunana kadar aşağıdaki işlemleri adım adım yapacaktır:

1. İlk adımda ekranın görüntüsü alınacaktır
2. İkinci adımda bu ekran görüntüsü grayscale’e çevirilecektir
3. Üçüncü adımda bu grayscale ekran görüntüsüne bir binary threshholding uygulanacaktır ve gri değeleri 20 nin üstünde olan tüm pixeller tamamen beyazlaştırılacaktır
4. Dördüncü adımda ekranın en köşesinden başlayarak gri değeri 10’dan küçük olan pixeller aranacaktır böyle bir pixel bulunduğu zaman, o pixelin koordinatlarına (X, Y) diyelim, eğer (X, Y)’den, (X , Y + 599) ve (X + 549, Y)’ye kadar ve aynı zamanda (X , Y + 599)’dan, (X + 549 , Y + 599)’a ve (X + 549, Y)’dan (X + 549 , Y + 599)’a kadar (kısacası oyun ekranının tüm çevresi) tüm pixellerin değerleri 10’dan küçükse, (X, Y) oyun ekranının başlangıç koordinatları olarak kabul edilecektir. Aksi takdirde bir sonraki pixellere bakılacaktır.



3: İlk adımın sonucunda elde edilen resim



4: Üçüncü adımın sonunda elde edilen sonuç

Oyun ekranının koordinatları bulunduktan sonra oyun ekranının konumu değişmediği veya önüne bişey gelmediği sürece yukardaki işlemler bir daha yapılmayacaktır.

Bir sonraki adımda oyun ekranının koordinatları yardımı ile ve oyun ekranının boyutlarını kullanarak oyun ekranı Crop ediliyor. Bunun sonucunda resim 2 gibi bir resim elde ediliyor. Bu işlem 5 kere belli aralıklarla arka arkaya yapılarak oyundaki değişiklikler gözetleniyor ve bunun sonucunda topun hareketleri, başarı yüzdesi(uygunluk), skor ve can sayısı ve değişimleri incelenebiliyor. Bu 5 ekran görüntüsü belli aralıklarla ve periyodik olarak sürekli çekiliyorler. Bunun bir örneği altta gösterildiği gibidir.

5.a: 5.b:



5.c: 5.d:



5.e:



Oyun ekranından skor ve canı okumak için ilk başta bu ekrandan skor ve can sayılarının yazıldığı kısım crop ediliyor. Bu işlem önceden belirlenmiş skor ve can sayılarının yazıldıkları bolgenin koordinatları yardımı ile yapılıyor (projenin başında bu işlemi CNN yardımı ile yapmak planlanıyordu ama sınırlı işletim gücü ve problemin gerçek zamanlı bir problem olması nedeni ile önceden belirlenmiş koordinatların kullanılmasına karar verildi). Bu işlemin sonucunda elimizde aşağadakine benzeyen resimler olacaktır.



6.a: Sıfır sayısı 6.b: bir sayısı

Bu resimler kullanılarak tek bir pikselin değeri yardımı ile can değerini oğrenebiliriz. Bu pikselin gri seviyesi 200’un üstünde ise yani beyaz bir piksel ise elimizde 2 can var aksi takdir de elimizde 1 can vardır. Bu piksel aşağıda gösterilmiştir.



7

Yükardaki paragrafta anlarılan işleme benzer bir işlem skor değerini hesaplamak için kullanılıyor ama bu sefer kontrol edilen pixel sayısı 1 değil 7’dir ve bunlara göre skor sayısının kaç olduğu belirlenir.

Bu sayılar sürekli kontrol edilerak ve topun hızı ve bulunması gereken tahmini bölge hesaplanarak oyunun durumu anlaşılabilir. Örnek olarak oyunun ilk başında top hareket etmeye başladıktan sonra ilk 5 ekran görüntüsünde (yani yaklaşık olarak ilk 3 saniyede) can sayısında bir değişim yoksa bunun 2 nedeni olabilir ilk ve en muhtemel neden topun paddle’e isabet etmiş olması ki bu durumda bir sonraki 5’li de skor artımı beklenmektedir, ikinci ihtimal olarak ‘space’ tuşunun oyun tarafından algılanmamış olamsı ki bu durumda bir sonraki 5’li ekranda skor ve can değerlerinin sabit kaldıkları gözlemlenecektir.

**EKRANA GÖRE FİTNESS HESAPLAMA:**

Oyun boyunca can değeri kontrol edilerek topun oyundan çıkmış olup olmadığı anlaşılabilir. Eğer topun ekrandan çıktığı fark edilirse yani can değerinde bir değişim olursa, top ekrandan çıkmadan önceki oyun ekranının son görüntüsüne bakılır ve, top ve paddle arasındaki City Block uzaklık hesaplanır. Bu uzaklık değeri mümkün olan en yuksek mesafeye yani 996 sayısına bölünür ve bu bölmenin sonucu 1 sayısından çıkartılır ve en son olarak elde edilen sayı 100 ile çarpılır. Böylelikle top her ekranın dışına çıktığı zaman paddle’in top’u kaçırdığı mesafeye göre bir fitness hesaplanır.

**GENETİK ALGORİTMANIN UYGULANMA TEMELLERİ:**

Bu sistemde genetik algoritma her skordan bir sonraki skora geçmenin yöntemini aramaktadır ve her skordan bir sonraki skora geçme yöntemi bulunduğu zaman bu yöntem o skorun çözümü olarak sistemde kayit ediliyor ve bir sonraki skorun çözümünü bulmak için tamamen yeni bir populasyon oluşturuluyor. Ayrıca herhangi bir nedenden dolayı (bazen fps salınımları yüzünden paddle hareketlerinin etkilenmesi, cekilen ekran görüntülerinin zamanlamalarında hata olması veya tuşların oyun tarafından algılanmaması) algoritmanın o belli bir skor için ürettiği çözümün yanlış olduğu fark edilirse, o skor için üretilen çözüm silinecek ve tamamen yeni bir populasyonla birinci jenerasyondan başlayarak çözüm yeniden üretilecektir.

**POPULASYONLAR VE BİREYLER:**

Bu sistemde kullanılan genetik algoritmadaki her populasyon, 5 bireyden oluşmaktadır. Bu bireylerin herbiri 15 uzunluğunda -1 ve 1 lerden oluşan bir listedir. Sistem her gördüğü -1 için sol tuşuna ve her gördüğü 1 için sağ tuşuna basacaktır ayrıca her bireyden sonra sistem bir kez ‘space’ tuşuna basıp ekran kaydına başlıyor.

Eğer ekran kaydının sonunda (5 resimin sonuncusunda) can değeri başta olduğundan farkıysa, top ekrandan çıkmış demektir ve o bireyin fitness’i hesaplanır ve genetik algoritma işlemine devam eder.

Ama eğer ekran kaydının sonunda can değeri değişmemiş ise, topun, paddle e değdiği vay sayılır ve bu adımdaki bireyi oluşturan -1 ve 1’lerin sayısal toplamı bir çözüm önerisi olarak kabul edilir. Örnek olarak eğer bu toplam 3’e eşitse bu o skora gelindiğinde 3 kere sağ tuşuna basılarak top yakalanabilir anlamına geliyor.

Bir sonraki adımda bu çözüm önerisinin geçerli bir sonuç olup olmadığı test edilir ve eğer öyleyse, bu çözüm, çözümler kümesine eklenir.

Genetik algoritmanın ürettiği her birey çalıştırılmadan önce eğer çözümler kümesinde herhangi bir çözüm var ise önce o çözümler çalıştırılır. Mesela eğer çözüm kümesinde 5 tane sayı varsa önce onlar çalıştırılır ve skor 5’e yükseltilir ve sonra genetik algoritmanın 6 skorunun çözümünü bulmak için populasyonları çalıştırılır.

**YENİ JENERASYONLARIN ÜRETİLMESİ:**

Oncelikle sıralama seçimi yöntemi ile bir sonraki jenerasyona aktarılacak 2 birey seçilir. Sonrasında bu iki birey kullanılarak ve cross over yontemi yardımıyla 3.birey sonrasında bu üçunden ikisini seçerek 4.bire ve sonrasında bu 4 bireyin ikisini kullanarak 5.birey üretilecektir. Cross over işlemlerinin hem bir noktadan ve hem iki noktadan olma ihtimali var ve bu iki ihtimal birbirine eşittir.

Gelecek jenerasyonu oluşturmaktaki son adım mutasyon’dur. Mutsyon aynı zamanda bu sistemdeki en önemli işlemlerden biridir. Bunun nedeni, sistemin her birey için çektiği ekran görüntüsünün zamanının tam olarak diger bireylerinki ile aynı olmaması ve bu durumun bizim elimizde ortalama bir fitness olmasına ve fitness değerlerinin tamamen güvenilir olmamasıdır. Bu fitness değerleri bizim topa yaklaşmamıza sebep olacaklardır ama sadece bu fitness yardımı ile topun tam yerini bulmak çok zordur. Bu durumun çözümü için sistemde yüksek bir mutasyon oranına sahip olmalıdır. Mutasyon oranı %60 olarak belirlenmiştir.

**ÇÖZÜM ÖNERİLERİ VE ÇÖZÜMLER:**

Sistem her çalıştırıldığı zaman böş bir çözüm kümesi üretir. Yukarda da anlatıldığı gibi topun her paddle’e çarptığı tespit edildiği zaman, çalıştırılan en son bireyin sayısal toplamı bir çözüm önerisi olarak kabul ediliyor. Bu çözüm önerisi bir çözüm olarak kabul edilip çözümler kümesine eklenmeden önce test edilir. Bu test, elimizdeki çözüm önerisini 3 kere çalıştırıp her seferinde skor değerini kontrol ederek eğer bu 3 çalıştırmanın 2 tanesinde skor da artış gözüküyorsa öneri testi geçmiştir ve cözüm olarak kabul edilmiş demektir. Aksi takdirde öneri ve o öneriyi üreten populasyon silinecektir ve genetik algoritma yeniden skor artışı için bir çözüm üretmeye çalışacaktır.

Sistem bu işlemi 10 poana ulaşana kadar devam edecektir.

**SENKRONİZASYON PROBLEMİ:**

Bu sistemin tasarımında karşılaşılan en büyük sorunlardan bir tanesi senkronizasyon problemi ve bu problem bilgisayarın performansı ve işleyişi gibi kontrol edilemez parametrelere bağlı olduğu için, ve diğer yandan bu problemin sistemin sonucunu tamamen etkileyebileceği için sistemin çoğu tasarımı bu problemin etkisini minimuma indirmek amacıyla tasarlanmış.

Bu problemi çözmek için ilk başta oyunun fps sayısı 30 a sabitlenmiştir böylelikle topun hareketleri daha kolya ön görülebilir. Fps limitine rağmen bilgisayarın performansı nedeniyle fps de salınımların yaşanma ihtimali mevcut ve bu durum yanlış çözümlerin üretilmesine neden olabilir ki çözümleri test etmenin amacı bu sorunu gidermek.

Bir sonraki adımda bilgisayara işlem zamanı vermek ve ekran görüntülerini zamanında alabilmek için, oyuna başlat tuşu eklendi ve bu tuşa basılmadığı sürece topun hareket etmesi engellendi.

Top harekete geçtikten sonra herşey gerçek zamanlı olarak çalışmalıdır. Bu durumu sağlamak için sisteme bekleme süreleri verilmiştir. İlk başta eğer herhangi bir çözüm mevcut ise çözumler tek tek çalıştırılacaktır, her iki çözüm arasında 3.25 saniyelik bir bekleme süresi vardır (bu süre topun ekranın yukarı sınırına kadar gidip geri gelmesi içindir). En son olarak genetik algoritmanın ürettiği sonuç uygulanır, sistem 0.4 saniye bekler ve ekran görüntülerini çekmeye başlar. Bunun sonucunda topun dışarı çıkmış olup olmadığı görülebilir.

Senkronizasyon amacıyla yapılan bir başka şey ise, top her paddle’e çarptığında ve yeni bir çözüm önerisi üretiliğinde sistemin test aşamasına geçmeden önce, topun bir kez oyundan çıkmasını beklemesidir. Böylelikle sistem test aşamasında herşeyi sıfırdan ve senkronize bir şekilde başlatabilir.