YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MİNİMAKS ALGORİTMASINA DAYALI BİR SATRANÇ OYUN YAZILIMI

YÜKSEK LİSANS TEZİ

Fedai ŞEKERCİOĞLU

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Haziran 2012

YALOVA ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MİNİMAKS ALGORİTMASINA DAYALI BİR SATRANÇ OYUN YAZILIMI

YÜKSEK LİSANS TEZİ
Fedai ŞEKERCİOĞLU
(105105004)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Yrd. Doç.Dr. Osman H. KOÇAL

Haziran 2012

YALOVA Üniversitesi Fen Bilimleri Enstitüsü'nün 105105004 numaralı Yüksek Lisans Öğrencisi **Fedai ŞEKERCİOĞLU** ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"MİNİMAKS ALGORİTMASINA DAYALI BİR SATRANÇ OYUN YAZILIMI"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Yrd.Doç. Dr. Osman H. KOÇAL** Yalova Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Ahmet AKBAŞ** Yalova Üniversitesi	.....
	Yrd.Doç. Dr. Osman H. KOÇAL Yalova Üniversitesi	.....
	Yrd.Doç. Dr. Kayhan İNCE Yalova Üniversitesi	.....

Teslim Tarihi : **18 Mayıs 2012**

Savunma Tarihi : **27 Eylül 2012**

Aileme,

ÖNSÖZ

Bu tez çalışması yapay zeka uygulamalarından biri olan oyun teorisi hakkında yapılan bir uygulamayı açıklamak amacıyla yazılmıştır. Oyun algoritmalarından biri olan minimaks algoritması, yapılan uygulamada kullanılmıştır. Yapılan uygulama kullanıcı ara birimi, satranç kurallarının uygulandığı metodlar, minimaks algoritması ve değerlendirme kısmından oluşmaktadır. Çalışmalarımda beni yönlendiren danışman hocam, Yrd.Doç.Dr. Osman Hilmi KOÇAL'a teşekkür ederim.

Haziran 2012 Fedai ŞEKERCİOĞLU

İÇİNDEKİLER

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MİNİMAKS ALGORİTMASINA DAYALI BİR SATRANÇ OYUN YAZILIMI

ÖZET

Bu tez çalışmasında oyun teorisi uygulamalarından biri olan minimaks algoritması anlatılmış ve bu algoritmanın bir uygulaması yapılmıştır. Minimaks algoritmasının ne olduğu ve çalışma prensibi açıklanmıştır. Algoritmanın açıklanması oyun ağaçları kullanılarak yapılmıştır.

Minimaks algoritmasının bir oyun programında nasıl kullanılacağının uygulaması anlatılmıştır. Bu algoritma uygulama olarak bir satranç programında kullanılmıştır. Program bilgisayara karşı satranç oynanırken, bilgisayarın yapacağı hamlenin belirlenmesini, minimaks algoritmasını kullanarak yapmaktadır. Programda kullanıcı beyaz taşlarla, bilgisayar ise siyah taşlarla oynamaktadır. Program satranç oyununun tüm kurallarını uygulamaktadır. Yazılan programda programlama dili olarak java kullanılmıştır.

Tezin giriş kısmında satranç kuralları kısaca tanıtılmıştır. Ayrıca arama algoritmaları, oyun teorisi gibi yapay zeka konularından genel olarak bahsedilmiştir.

Tezin ikinci bölümünde minimaks algoritmasının ne olduğu, çalışma prensibi, algoritma şeması anlatılmıştır. Algoritmanın nasıl değerlendirme yaptığı oyun ağacı kullanılarak gösterilmiştir. Oyunun değerlendirilmesinde iki adet değerlendirme kriteri kullanılmıştır. Taşların statik değerlendirilmesi, taşların türüne göre yapılmıştır. Oyunun dinamik değerlendirilmesi, taşların satranç tahtasında bulunduğu konuma göre yapılmıştır. Her olası hamle için taşların statik ve dinamik değerleri toplanarak minimaks algoritmasına gönderilmiştir. Minimaks algoritmasında yapılabilecek tüm olası hamleleri değerlendirdikten sonra bilgisayarın yapacağı en iyi hamle belirlenmektedir.

Tezin üçüncü bölümünde oyunun programlanması anlatılmıştır. Kullanıcı arayüzü java'nın grafik kütüphanesi kullanılarak gerçekleştirilmiştir. Taşların hareketinde java animasyonu kullanılmıştır. Bu sayede satranç taşlarının hareketi bilgisayar faresi yardımı ile sürükle bırak yöntemi kullanılarak sağlanmıştır. Yanlış hamle yapıldığında, şah çekildiğinde ve oyun bittiğinde uyarı mesajları ekrana gelmektedir. Her taş çeşidi için bir metot yazılmıştır. Bu metotlarda ilgili taş türüne ait satranç kurallarının denetimi yapılmaktadır. Örneğin piyon metodunda; piyonun bir kare ileri hareketi, iki hamle ileri hareketi, sağında yada solunda rakip taş var ise bunu alması, geçerken alma kuralının uygulanması ve piyonun son yataya ulaştığında vezir olması gibi o taş ile ilgili tüm satranç kuralları denetlenmektedir. Bir hamle yapıldığında o taşın yapabileceği tüm geçerli hamleler kontrol edilmektedir. Ayrıca ses metodu eklenmiş; bu metotta yapılan her geçerli hamle sonrası uyarı verilmesi sağlanmaktadır. Dördüncü bölün sonuç bölümüdür.

Bu tez çalışmasında önerilen satranç oyun algoritmasının başarısı, statik ve dinamik değerlendirmeler yapılan denemelerle test edilmiştir.

Yapılan değerlendirmeler bu tez çalışmasına özgün olarak önerilen dinamik puanlandırma durumunda önemli performans artışı sağlamıştır.

AN APPLICATION OF ARTIFICIAL INTELLIGENCE İN GAME THEORY

SUMMARY

In this thesis, minimax algorithm had been explained and an application had been applied using this algorithm. What is Minimax algorithm, discussed the operation principle. Explanation of the algorithm described by game trees. Minimax algorithm application is explained how to use this algorithm in a game program. This algorithm is used for chess program. Playing chess against the computer, the computer determines the move using minimax algorithm. Users play white pieces and computer plays the black pieces. This game to be written by java as the programming language.

The rules of chess had been introduced briefly in the introduction. Search algorithms had been told in the introduction, generally referred to as artificial intelligence, such as game theory.

What is the minimax algorithm? How is working operating prinsible? The algorithm of scheme had been told in second part of thesis. How the evaluation algorithm is shown through the game program. Evaluation criteria for the evaluation of the game is to use two. Static evaluation of the stones, the stones had been based on the type. Dynamic evaluation of the stones had been based to the location of the chessboard. Static and dynamic values of the stones collected for all possible moves and it had been sent to minimax algorithm. Minimaks algorithm, the computer will be be done the best move, after evaluating all posible moves is determined.

How is written chess game? It had been told in the third part of thesis. The user interface is realized using Java graphics library. Movement of the stones used in the java animation. Movement of the pieces had been provided using drag and drop. Warning messages had been displayed when the king is pulled and the game is finished and wrong moves is made. Written to a method for each type of stone. These mehods are the rule of chess for the type of stone. For example the method of pawn, pawn square forward movement, the forward movement of the two moves, the right or left of the competitor to take this of opposite player have the stone, the implementationof the rule of en passant and pawn reaches the rank of the last, it can change type of stone. When it moves the stone, that all possible moves is controlledwith current moving. In addition, volume method was added to the warning given in this method provided after each valid moves.

The fourth section had been section of result.

In this thesis, the success of the proposed algorithm to the game of chess, in both static and dynamic evaluations of trials tested.
Proposed in the original assessments made in this thesis has provided significant performance gains in case of dynamic grading.

1. GİRİŞ

Bu tez çalışması kullanıcı ara birimi olan ve yapay zekaya sahip bir satranç programı yapımını anlatılmaktadır. Programda satranç taşlarının hareketi sürükle bırak yöntemiyle yapılmaktadır. Bunun için oynanacak taş üzerine tıklanıp, gideceği kareye sürüklenerek hareketi sağlanmaktadır. Taşların satranç kurallarına gore hareket etmesi için, her bir taş için farklı metotlar yazılmıştır. Programın yapay zekasında minimaks algoritması kullanılmıştır.

1.1 Tezin Amacı

Minimaks algoritmasının bir oyun programında kullanılması ve yapay zeka değerlendirme kriterlerinin uygulamasının yapılmasıdır. Uygulama olarak bir satranç programı kullanılmıştır. Satranç programına karşı bir hamle yapıldığında, bilgisayarın karşı hamleyi yapması ve yapacağı bu hamlenin iyi bir hamle olması amaçlanmıştır.

1.2 Literatürde Benzer Makaleler

Literatürde makaleler incelenmiş, aşağıdaki makaleler referans alınmıştır.

Marc Boule 2002 yılındaki makalesinde ‘deep blue’ isimli satranç bilgisayarının çalışmasını anlatmaktadır. Bilgisayar yapılabilecek olası hamleleri hareket jeneratörünü kullanarak üretmektedir. Satranç tahtası 8x8’lik bir dizi olarak temsil edilmektedir. Saldırılan ve savunulan kareler belirlenmektedir. Bu işlemi bir döngü kullanarak gerçekleştirmektedir. Çözüm için oyun ağaıc kullanmıştır. Hareket üretimi için 4096 bit blok Ram kullanılmıştır. Uygulamada tahta dizisi oluşturulurken, bu makaledeki satranç tahta dizisi referans alınmıştır[1].

Hallain Nasreddine 2006 yılındaki makalesinde bir satranç oyununun her hamlede ortalama 35 olasılık olduğunu anlatmaktadır. Oyun ağacı ise 100 seviye derinliğe ulaşmaktadır. Hesaplamada alfa- beta budaması kullanılarak bazı dallar budanmaktadır. Bu makalede Shannon değerlendirme yöntemi kullanılmaktadır.

Değerlendirme fonksiyonunda her parçanın yapabildiği geçerli hamleler ve aynı hat üzerinde 2 piyon bulunmasına göre ağırlıkları değişmektedir. Üç hamle derinliğe bakmaktadır. Şah hariç diğer taşların 0 ile 100 arasında bir ağırlık değerleri bulunmaktadır. Taşların hareketliliği yüzde 10 olarak belirlenmiştir. Bu programın chessmaster 8000 satranç bilgisayarına karşı rekabet edebildiği gözlenmiştir. Uygulamada dinamik değerlendirme dizileri oluşturulmasında, bu makale referans alınmıştır[2].

S. H. Rubin 2008 yılındaki makalesinde satranç oyun geliştirme kullanıcı ara birimi anlatılmaktadır. Online oyunlarda istemci ve sunucu arasındaki iletişim için java kullanılmaktadır. Yönetim modülü, ghcm oyun modülü, veritabanı iletişim modülü ve haberleşme modüllerinden oluşmaktadır. Oyunlar veritabanına kaydedilir. Oyun kuralları ve grafik sınıfından türetilir. Mobil cihazlarlada kullanılabilmektedir[3].

Nasrul Humaimi Mahmood 2011 yılındaki makalesinde kaser öğrenme yöntemini açıklamaktadır. Oyun ağacı kullanılarak hesaplama yapılır. Oyun değerlendirilmesi;

1-Şah'ın merkeze yakınlığı

2-Oyuncunun alabileceği parça sayısı ve parçalar arasındaki değer farkı

3-Oyuncunun taşının oynayabileceği kareler sayısı

4-Rakip taşların oynayabileceği kareler sayısı

Yapılacak hamle oluşturulurken öncelik sırası;

1-Varsa yüksek değere sahip parça alınır.

2-Mümkünse tahtanın merkezine yakın parça alınır.

3-Şahın oynayabileceği kare sayısı azsa, değersiz taşlarını kullanarak şah'ı almak.

Farenin tıklanması ve fare imlecinin hareketi kullanılarak yapılır[4]. Uygulamadaki dinamik değerlendirme dizileri oluşturulurken, bu makaledeki değerlendirme kriterleri kullanılmıştır.

Satranç resimlerinin işaret işleme teknikleri kullanılarak, taşların çeşidinin belirlenmesi anlatılmaktadır[5].

Surakata satranç oyun programı tanıtılmaktadır. Oyunun tanıtımı ve algoritması anlatılmaktadır[6].

Kamerası olan ve satranç oynama yeteneğine sahip robot bilgisayar anlatılmaktadır. Oynanan hamle algılanıp bilgisayar kendi hamlesini yapmaktadır[7].

1.3 Oyun Teorisi

Oyun teorisi, İstatistik biliminin, sosyal bilimlerde, biyoloji, mühendislik, politik bilimler, yapay zekâ ve felsefede kullanılan bir dalıdır. Oyun kuramı, bireyin, başarısının diğerlerinin seçimlerine dayalı olduğu seçimler yapması olan bazı stratejik durumların matematiksel olarak davranış biçimlerini yakalamaya çalışır.

Her oyuncu yaptığı seçimlerde kazanma olasılığını artırmak için nash dengesine göre hareket etmelidir. Nash dengesi, her oyuncunun stratejisinin diğer oyuncuların stratejilerine en uygun cevap olan stratejiler profilidir.

Dinamik oyunlarda, bir oyunda daha sonra hareket eden oyuncu kendisinden önceki oyuncuların daha önce yaptıkları hamleleri bilmektedir. Bu yüzden daha önce hamle yapan oyuncular bu durumu hesaba katarak optimal stratejilerini düzenlemektedir. Bu durum, dinamik oyunlarda tahmin edilen davranışın her zaman açık ve anlaşılması kolay olmadığını ifade eden bir uyarıdır.

Statik Oyun Varsayımları:

i) Oyuncular eylem seçimlerini aynı anda ya da birbirlerinin haberi olmadan yaparlar.

ii) Tüm oyuncular akılcıdır.

iii) Tüm oyuncuların akılcılığı ortak bilgidir.

iv) Tüm oyuncular kusursuz fakat eksik bilgiye sahiptir.

1.3.1 Tutukluların İkilemi (Prisoners' Dilemma)

Bir soygun soruşturması sonucu Ali ve Veli isimli iki şüpheli yakalanmış ve ayrı odalarda ilk sorgulamalarının yapılmasını beklemektedirler. Güvenlik güçleri bu iki tutukluya bir anlaşma paketi önerir. Bu öneriye göre ikisi de suçu itiraf ederse beşer yıl, ikisi de reddederse ikişer yıl hapis cezası yiyeceklerdir. Eğer birisi itiraf, diğeri reddederse itirafçı serbest kalacak ve arkadaşı on yıl hapis cezası yiyecektir. Oyunun tanımı bu bilgilere göre yapılabilir.

1) I = {Ali, Veli}

2) Ai = {İtiraf, Red}, i = Ali, Veli

3) Bu oyunun her olası sonucu için getirileri bir getiri (kazanç) matrisi ile gösterilebilir:

Çizelge 1.1 : Oyun matrisi

		Veli
	İtiraf	**Red**
İtiraf	-5, -5	0, -10
Red	-10, 0	-2, -2

Ali

Dikkat edilecek nokta, yukarıdaki kazanç matrisindeki kazançların negatif olmasıdır. Çünkü bu oyunda kazançlar hapiste geçirilecek olan yıllardır. Her hücredeki ilk rakam satır oyuncusunun (Ali), ikincisi ise kolon oyuncusunun (Veli) getirileridir.

Bu stratejik çatışmada birbirleriyle iletişim kuramayan, akılcı tutukluların nasıl karar vereceklerini bilimsel bir yaklaşımla incelemek için, Nash dengesinden faydalanabiliriz.

Nash Dengesi : Nash dengesi kendine zorlayan (self enforcing) bir denge kavramıdır. Bu dengede, hiçbir oyuncu rakip oyuncunun eylemi sabit alındığında kendi seçimini değiştirmek istemez. Bir başka deyişle, hiçbir oyuncu, rakip oyuncunun stratejisi sabit alındığında, kendi eylemini değiştirerek kazancını arttıramaz.

Tutukluların ikilemi gibi 2x2 bir kazanç matrisi olan oyunlarda Nash dengesini (eğer varsa) bulmak çok kolaydır. Bunun için matrisin bütün hücrelerine tek tek bakmak yeterli olacaktır:

Veli'nin İtiraf eylemi sabit tutulursa, Ali'nin yapabileceği en iyi seçim İtiraf etmektir. Çünkü, itiraf ederse 5, etmezse 10 yıl yatacaktır. Veli'nin Red eylemi sabit tutulduğunda, Ali'nin en iyi seçimi yine İtiraf olacaktır. Çünkü Ali serbest kalmayı, 2 yıl hapse yeğleyecektir. Yani, Veli ne yaparsa yapsın itiraf etmek Ali için

dominant bir stratejidir. Veli için de aynı durum söz konusudur. Akılcı oyuncular ayrı odalarda, birbirlerinin nasıl davranacaklarını düşünürken ulaştıkları sonuç olan (itiraf, itiraf) gerçekten oyunun Nash dengesini verir, çünkü ne Ali ne de Veli rakibin itiraf stratejisi karşısında kendi itiraf stratejilerini değiştirmek istemezler. Oysa her ikisi de, beşer yıl yerine ikişer yıl hapis yatmayı tercih ederler. Bu tercihlerine rağmen, akılcı oldukları ve akılcılığın genel bilgi olduğu için işbirlikçi sonucu (Red, Red) elde edemezler. Oyunun ismindeki ikilem sözcüğü buradan kaynaklanmaktadır.

Bu oyun, oyuncuların dominant stratejilerine bakılarak da çözülebilir. Akılcı bir oyuncu domine edilen bir stratejiyi kesinlikle oynamayacaktır. Her iki oyuncunun da dominant stratejisi İtiraf etmektir. İtiraf stratejisi, Red seçimini domine eder. Akılcı Ali ile Veli Red stratejisini hiç düşünmeyeceklerdir bile. Dolayısıyla dominant stratejilerde denge de Nash dengesi ile aynı sonucu (itiraf, itiraf) verir. Bu şaşılacak bir sonuç değildir, zira her dominant strateji dengesi aynı zamanda Nash dengesidir.

İşbirliği ile rekabet arasında bir gerilim bulunan her stratejik karşılaşmanın özünde bu tip bir ikilem yatar. Bu yüzden bu tip oyunlar genel olarak tutukluların ikilemi oyun kategorisine girerler.

1.4 Arama Algoritmaları

Bilgisayar bilimlerinde, çeşitli veri yapılarının (data structures) üzerinde bir bilginin aranması sırasına kullanılan algoritmaların genel ismidir. Örneğin bir dosyada bir kelimenin aranması, bir ağaç yapısında (tree) bir düğümün (node) aranması veya bir dizi (array) üzerinde bir verinin aranması gibi durumlar bu algoritmaların çalışma alanlarına girer.

Arama algoritmalarını kullanmamızdaki amaç, algoritmanın isminden de anlaşılacağı üzere- aradığımız veriye en kısa yoldan ulaşmamızı sağlayacak bir yöntem bulmaktır.

i) Derinlik Öncelikli Arama (Depth-First Search)

ii) Genişlik Öncelikli Arama (Breadth-First Search)

1.4.1 Oyunlar ve arama problemleri

• Ne yapacağı belli olmayan "Unpredictable" durumlarda rakibin karşılık verebileceği her hareketi incelemek gerekir.

• Zaman limiti belirlenen zaman içinde amacı sağlayan çözümü bulmak pek mümkün değil, yakınsama (approximate) veya varsayımlar, olasılıklar kullanmak gerekir.

1.5 Oyunun Temel Kuralları

1.5.1 Satranç oyununun doğası ve amaçları

Satranç oyunu, kare seklindeki, satranç tahtası üzerinde, iki rakip arasında taşların sırayla hareket ettirilmesi ile oynanır. Oyunu beyaz taşlarla oynayan oyuncu başlatır.

Her iki tarafın da amacı, rakip şahı, geçerli bir hamleyle önlenemeyecek biçimde tehdit altında tutmaktır. Bunu başaran taraf rakibini mat yapmış demektir ve oyunu kazanır. Oyuncunun; şahını tehdit altında bırakmasına, tehdit altına sokacak hamle yapmasına ve rakibinin şahını almasına izin verilmez. Şahı mat olan taraf oyunu kaybeder.

Hiç bir oyuncunun mat yapma olanağının kalmadığı bir konumda, oyun beraberedir.

Satranç tahtası, çizgili 64 (8x8) eşit kareden oluşur ve kareleri sırayla açık (beyaz) ve koyu (siyah) renktedir.

Satranç tahtası, iki rakip arasına beyaz renkli köşe karesi sağ tarafta olacak şekilde yerleştirilir.

Oyunun başlangıcında bir taraf 16 adet açık renkli (beyaz), diğer taraf 16 adet koyu renkli (siyah) taşa sahiptir.

Bu taşlar aşağıda gösterilmiştir:

Çizelge 1.2 : Satranç taşları ve simgeleri

 Siyah piyon 8 tane	 Beyaz piyon 8 tane
 Siyah at 2 tane	 Beyaz at 2 tane
 Siyah fil 2 tane	 Beyaz fil 2 tane
 Siyah kale 2 tane	 Beyaz kale 2 tane
 Siyah vezir 1 tane	 Beyaz vezir 1 tane
 Siyah şah 1 tane	 Beyaz şah 1 tane

Başlangıç konumu tahta üzerinde aşağıdaki gibi gösterilir.

Şekil 1.1 : Başlangıç konumu

Sekiz dik karenin oluşturduğu sütunlara dikeyler, sekiz ufki karenin oluşturduğu satırlara yataylar, tahtanın bir kenarından bitişik diğer bir kenarına uzanan**,** aynı renkli karelerden oluşan düzgün hatta ise çapraz denir.

Hiçbir taş aynı renkte başka bir taşın bulunduğu bir kareye gidemez. Eğer bir taş rakip bir taşın olduğu kareye giderse, aynı hamlenin bir parçası olarak rakip taşı almış olur ve alınan taş tahta dışına çıkarılır.

Bir taş, kendi şahını tehdit altına sokacağı ya da tehdit altında bırakacağı için bulunduğu kareden ayrılamıyor olsa bile, gidebileceği bir kareyi/kareleri tehdit altında tuttuğu kabul edilir.

Fil, bulunduğu karenin çaprazları üzerindeki herhangi bir kareye gidebilir. Filin hareketi gösterilmiştir (Şekil A.1).

Kale, bulunduğu karenin dikey ve yatayları üzerindeki herhangi bir kareye gidebilir. Kalenin hareketi gösterilmiştir(Şekil A.2).

Vezir, bulunduğu karenin dikey, yatay ve çaprazları üzerindeki herhangi bir kareye gidebilir. Vezirin hareketi gösterilmiştir(Şekil A.3).

Fil, kale veya vezirle bu hamleler yapılırken, hamle yolu üzerinde bulunan bir

taşın üzerinden atlanamaz.

At, aynı yatay, dikey ya da çapraz üzerinde olmamak koşulu ile bulunduğu kareye en yakın karelerden birine gidebilir. Atın hareketi gösterilmiştir(Şekil A.4).

Piyon bulunduğu dikeyde hemen önündeki boş kareye doğru ilerleyebilir, ya da

ilk hareketinde piyon, bir kare ilerleyebilir veya istenirse, bulunduğu dikeyde önündeki her iki karenin de boş olması koşulu ile iki kare ilerleyebilir, ya da piyon, bitişik dikeyde, bir on çaprazı üzerinde bulunan rakip taşı alarak bu kareye gidebilir.

Piyonun hareketi gösterilmiştir(Şekil A.5). Bir piyon, başlangıç konumundan en uzak yataya ulaştığında, aynı hamlenin bir parçası olarak, aynı karede, kendisi ile aynı renkte, yeni bir taş ile değiştirilmelidir. Piyonun bu değişimine terfi denir ve yerine konan taş kurallara uygun olarak derhal devreye girer. Bir piyon; tek bir hamlede başlangıç konumundan iki kare ilerlemiş bulunan rakip piyonun geçtiği kareyi tehdit altında tutuyor ise, bu rakip piyonu, sanki o hamlede tek bir kare ilerlemiş gibi, alabilir. Bu alış bicimi, sadece sözü edilen piyon sürüsünü izleyen hamle için geçerlidir, ve geçerken alma (en passant) olarak adlandırılır. Piyonun geçerken alma hareketi gösterilmiştir(Şekil A.6).

Şahı hareket ettirmenin iki farklı yolu vardır, bir ya da daha fazla rakip taş tarafından tehdit edilmeyen herhangi bir bitişik kareye giderek. Şahın hareketi gösterilmiştir ya da rok(Şekil A.7). Bir şah hamlesi olarak kabul edilen ve aynı renkteki şah ile kalenin oyuncunun ilk yatayı üzerinde gerçekleştirdikleri bir hamledir; ve şöyle yapılır: şah, başlangıç karesinden, yine başlangıç karesinde bulunan kaleye doğru iki kare ilerler, ardından kale, şahın en son geçmiş olduğu kareye konur. Beyaz ve siyah şahın rok hareketi gösterilmiştir(Şekil A.8). Şahın, bir ya da daha fazla rakip taşın tehdidi altında olması, kendisine şah çekildiği anlamına gelir, ve bu; şah çeken rakip taşların, kendi şahlarını tehdit altında bırakacakları (ya da sokacakları) için bulundukları karelerden ayrılamadıkları hallerde dahi, geçerlidir. Hiç bir taşla kendi şahını tehdit altında bırakacak ya da tehdit altına sokacak hamle yapılamaz.

1.5.2 Oyunun Sonuçlanması

Oyun, rakibini, mat yapan oyuncu tarafından kazanılır. Mat konumunu yaratan hamlenin geçerli bir hamle olması durumunda, oyun derhal sona erer.

Oyun, rakibi terk ettiğini bildiren oyuncu tarafından kazanılır. Oyun derhal sona erer.

Şahı tehdit altında olmayan hamledeki bir oyuncunun, yapabileceği geçerli bir hamle yoksa, oyun berabere olur. Bu durumda oyun pat olarak bitmiş olur. Pat konumunu yaratan hamlenin geçerli bir hamle olması durumunda, oyun derhal sona erer.

Her iki tarafın da, geçerli herhangi bir hamleler dizisiyle rakip şahı mat edemeyecekleri bir konum oluştuğunda, oyun berabere sonuçlanır. İki taraf oyun sırasında aralarında anlaşırlarsa oyun berabere biter. Oyun derhal sona erer.

Herhangi özdeş bir konumun, satranç tahtası üzerinde, en az uç kez oluşması ya da oluşmak üzere olması halinde, oyun berabere olarak sonuçlandırılabilir.

Ardışık son 50 hamle, her iki oyuncu tarafından da, piyon sürülmeden ve tas alınmadan geçirilmiş ise oyun berabere olarak sonuçlandırılabilir.

2. MİNİMAKS ALGORİTMASI

Minimaks, kaybetme olasılığını minimuma indirirken kazanma potansiyelini maksimuma çıkarmaya yarayan karar verme algoritmasıdır. Uygulama alanı oyun ağaçları olduğu için oyun teorisi için belirtilen kısıtların hepsi minimaks için de geçerlidir.
Minimaks bilgisayar mühendisliğinde, yapay zeka konusunda kullanılan bir karar ağacı türüdür. Minimaks ağaçları bilgisayar bilimlerine işletme bilimindeki oyun teorisinden (game theory) girmiştir.

Temel olarak sıfır toplamlı bir oyunda (zero sum game), yani birisinin kaybının başka birisinin kazancı olduğu (veya tam tersi) oyunlarda karar vermek için kullanışlıdırlar. Örneğin çoğu masa oyunu (satranç, othello, tictactoe gibi) veya çoğu finansal oyunlar (borsa gibi) veya çoğu kumar oyunları sıfır toplamlı oyunlar arasında sayılabilir (yani birisinin kaybı başka birisinin kazancıdır ve sonuçta toplam sıfır olur).

Yukarıda bahsedilen bu oyunlarda doğru karar verilmesini sağlayan minimaks ağacı basitçe kaybı asgariye indirmeye (mimize etmeye) ve dolayısıyla kazancı azamiye çıkarmaya (maximize etmeye) çalışır.

Ağaç basitçe her düğümde (node) farklı alternatiflerin değerlerini hesaplar. Son düğümden yapraklardan yukarıya doğru değerleri seçerek gelir ve en sonunda bütün ağaçtaki en doğru seçenek seçilmiş olur.

Altüst adı da verilen bu yöntem zeka (mantık) oyunlarında sıklıkla kullanılan bir yapay zeka tekniğidir. Bu yöntemin kullanılabilmesi için aşağıdaki şartların sağlanması gerekmektedir:

İki kişilik olmalıdır

1) Sırayla oynanmalıdır. (Bir 1. oyuncu bir 2. oyuncu şeklinde)

2) Oyun durumu ilgili bilgiyle birlikte tahta olarak temsil edilebilmelidir.

3) İki oyuncuda sonraki olası hamlelerin hepsi hakkında bilgiye sahip olmalıdır. Buna mükemmel bilgi (“perfect knowledge”) denilmektedir. Örneğin Poker bu duruma uymaz. Rakibimizin elindeki kartları bilemeyiz.

4) Oyun hamleleri rasgele olmamalıdır. (Zar oyunları dışarıda kalmaktadır)

5) Oyunlar sonlu olmalıdır. Bir şekilde bitiş durumu olmalıdır.

Minimaks yöntemi yukarıdaki şartları sağlayan oyunlar için bir arama yöntemidir. Bu arama daha önce bahsettiğimiz oyun ağacı üzerinde en uygun hamlenin bulunmasına yönelik bir aramadır.

2.1 İşlem basamakları

-Ağacın yaprakları puanlandırılır.

-Hamle sırası oyuncudaysa birbirine bağlı düğümlerin maksimumu, rakipteyse minimumu alınır ve bağlı oldukları bir üst seviyedeki düğüme aktarılır.

-Bu seviyede önceki seçimin tam tersi yapılır. Bu işlemler köke ulaşıncaya kadar devam eder.

-Köke ulaşan değer ile değeri yollayan yaprak arasındaki yol izlenmesi gereken en uygun yoldur.

2.2 Minimaks bileşenleri

• **MAX** = Bizim olası hamlelerimizden kendi açımızdan en iyi sonucu verenin seçilmesi

• **MIN** = Rakibin olası hamlelerinden kendi açısından en iyi sonucu veren, bizim açımızdan en kötü sonucu verenin seçilmesi

• **UTILITY** = Oyunun sonucunu gösteren durum (Ör: kazanırsak +1, berabere 0, kaybedersek -1)

2.3 Minimaks algoritması

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Şekil 2.1: Minimaks algoritması genel

Minimaks algoritmasında kök'ten başlanarak max – min şeklinde seviyelendirme yapılır. Kök max olacak şekilde seviyelendirmeye başlanır. Bir altındaki seviye min olarak adlandırılır. Altındaki seviye tekrar max olur. Bu şekilde devam edilerek seviyelendirme yapraklara uşaşılıncaya kadar devam eder.

Min fonksiyonu aynı seviyede bulunan ve aynı ata'ya sahip olan düğümlerdeki değerlerden minimum olanını bulur. Bu değer ata'sının düğüm değeri olarak yazılır.

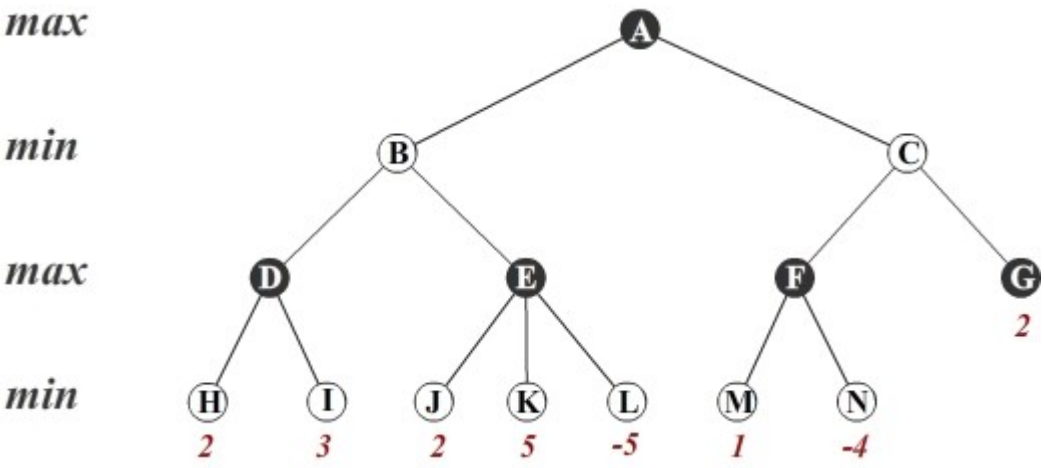
Max fonksiyonu aynı seviyede bulunan ve aynı ataya sahip olan düğümlerdeki değerlerden maksimum olanını bulur. Bu değer ata'sının düğüm değeri olarak yazılır.

2.4 Çalışma Prensibi

Minimax'ta her iki oyuncunun da her zaman en iyi hamleyi seçeceği varsayılır. Sıfır-toplamlı kuralına göre; bir hamle, yapan oyuncu için ne kadar iyiyse rakibi için aynı oranda kötüdür. Oyun analizinde bir oyuncu seçilir, kazanç ve kayıplar her zaman bu oyuncu üzerinden hesaplanır.

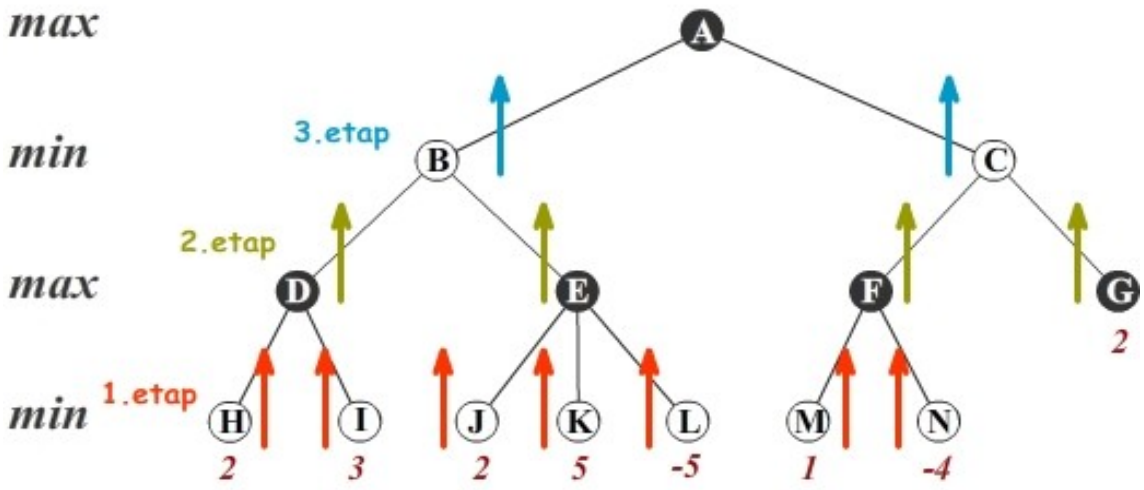
Minimaks algoritması, iki oyuncunun da tüm olası hamlelerini içeren ağaç üzerinde çalışır. Ağacın en alt basamağına kadar inilerek yapraklar puanlandırılır. Hamle sırası oyuncudaysa birbirine bağlı düğümlerin maksimumu, rakipteyse minimumu alınır ve bağlı oldukları bir üst seviyedeki düğüme aktarılır. Bu seviyede önceki seçimin tam tersi yapılır. Bu işlemler köke ulaşıncaya kadar devam eder. Köke ulaşan değer ile değeri yollayan yaprak arasındaki yol izlenmesi gereken en uygun yoldur.

Algoritmayı aşağıdaki ağaç üzerinde inceleyelim:



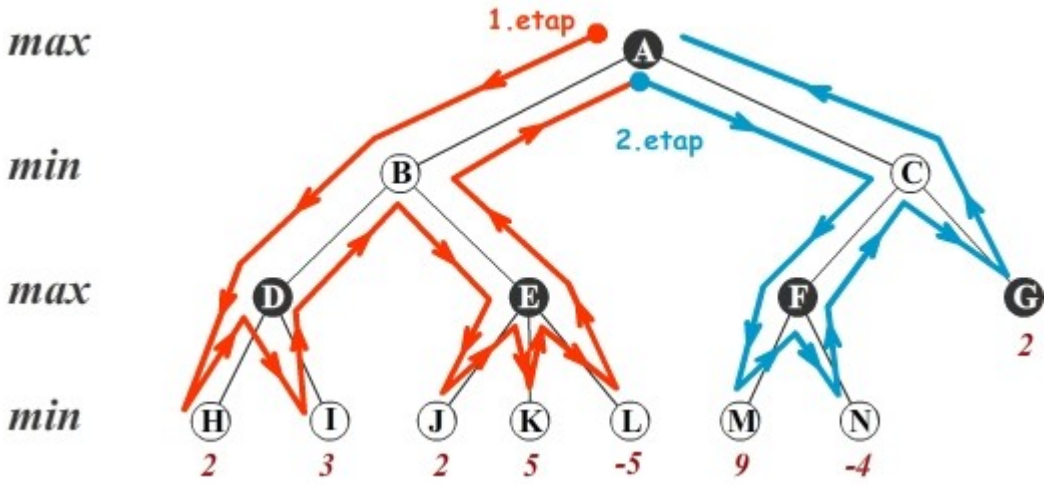
Şekil 2.2 :Oyun Ağacı Örneği

Az önceki anlatım üzerinden algoritmanın şu şekilde çalıştığını düşünebilirsiniz:



Şekil 2.3: Sığ Öncelikli Arama

Bu arama şekline “Sığ Öncelikli Arama” (Breadth First Search) denir. Arama yatayda ilerlemektedir. Sonuç olarak aynı işi yapsalar da bizim algoritmamız “Derin Öncelikli Arama” (Deep First Search) yapmaktadır ve dikeyde hareket etmektedir:



Şekil 2.4:Derin Öncelikli Arama

Bu bilgi doğrultusunda aramamızı başlatalım:

A: max(B, C)
  B: min(D, E)
    D: max(H, I)
      H: 2
    D: max(2, I)
      I: 3
    D: max(2, 3) = 3
  B: min(3, E)
    E: max(J, K, L)
      J: 2
    E: max(2, K, L)
      K: 5
    E: max(2, 5, L)
      L: -5
    E: max(2, K, -5) = 5
  B: min(3, 5) = 3
A: max(3, C)

C: min(F, G)

F: max(M, N)

M: 1

F: max(1, N)

N: -4

F: max(1, -4) = 1

C: min(1, G)

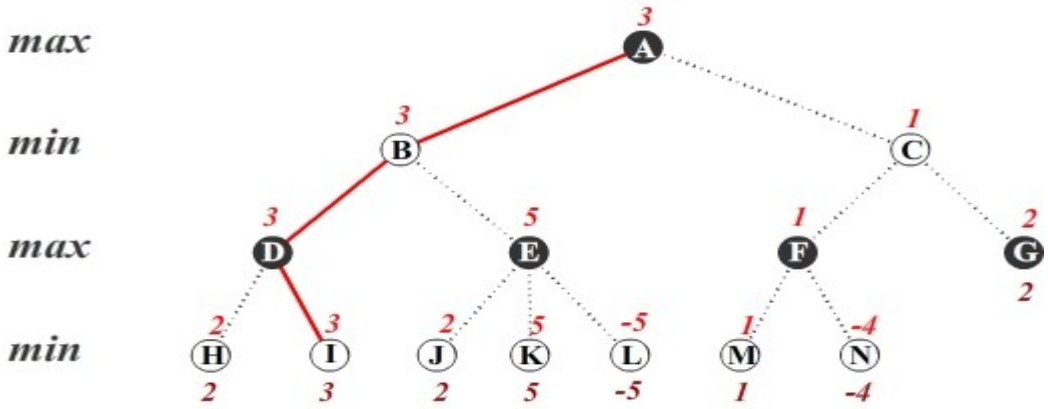
G: 2

C: min(1, 2) = 1

A: max(3, 1) = 3

Minimaks Algoritması Analizi

Bu durumda izlenecek en iyi yol A->B->D->I olacaktır:



Şekil 2.5: Minimaks Algoritması Sonucu Seçilen Yol

Minimaks algoritması iki fonksiyondan oluşur. Bir fonksiyon maksimum değerli düğümü seçerken diğeri minimum değerliyi seçer. Fonksiyonlar birbiri içine geçmiştir. Düğümünü seçen fonksiyon değeri öteki fonksiyona yollar. Bu işlemler ağaç bitene kadar devam eder.

Minimaks algoritması ağaç üzerindeki bütün düğümleri kontrol etmektedir. Bu da algoritmanın performansının düşmesine neden olmaktadır.

Yapay zekanın karar vermesi minimaks algoritmasına göre yapılıyor. Bu algoritma oyunlar için kullanılan bir yapay zeka algoritmasıdır. Algoritmanın mantığı şöyledir hamleyi yapacak olan taraf için kendi avantajını maksimum yapacak olan hamle seçilirken, rakibin ise avantajını minimum yapan hamle seçilir. Bu sayede yapay

zeka yapılabilecek en iyi hamleyi bulmuş olur. Yapay zeka kısmı programda geçerli hamlelerin hesaplandığı hamle metodunun içinde yapılmaktadır. Böylece her geçerli hamle için minimaks algoritması çalıştırılmış olur. Minimaks algoritması aşağıdaki şekilde gösterilmiştir.

```
if (derinlik % 2 == 0)
    {   //İnsan
        if (deger > minimaks [derinlik] )
            minimaks [derinlik] = deger;

    } else {
        //Bilgisayar
        if (deger <= minimaks [derinlik] )
        {
            minimaks [derinlik] = deger;
            if (derinlik == 1)
                hareket = basla * 100 + son;
        }
}
```

Şekil 2.6 Minimaks algoritması

"if" koşulu kullanılarak hamlenin siyahta mı, yoksa beyazda mı olduğu bulunmaktadır. Programda minimaks'ın başlangıç değeri atanmıştır. Minimaks değeri ile değerlendirme metodundan dönen değer karşılaştırılır. Eğer value değeri küçükse bu değer minimaks değerine atanır. Böylece rakibin avantajını minimum yapacak olan değer minimaks değişkeninde depolanmış olur. Hamle bilgisayara geçtiğinde ise minimaks değeri ile kendi taşlarını hesaplanması sonucu bulunan, yani değerlendirme metodundan dönen değer karşılaştırılır. Eğer metoddan geri dönen değer minimaks değerinden büyük ise bu değer minimaks değişkenine atanır. Böylece kendi avantajını maksimum yapacak olan değer minimax değişkeninde depolanmış olur.

2.5 Yapay Zeka Değerlendirme

Yapay zekanın değerlendirip karar vermesi için değerlendirme isimli bir metod yazılmıştır. Bu metodda 2 niteliğe bakıp öyle değerlendirme yapmaktadır.

2.5.1 Değerlendirme fonksiyonları

• Oyunun sonucunun çok derinlerde olduğu durumlarda durumların değerlendirilmesi için kullanılan fonksiyonlardır.

• Satranç için bu fonksiyon genellikle önceden belirlenen özniteliklerin doğrusal toplamı olarak düşünülür.

$$Eval_{(s)} = w_1 f_{1(s)} + w_2 f_{2(s)} + \ldots + w_n f_{n(s)} \quad \textbf{(3.1)}$$

• Örnek, $w_1 = 9$ ve

• $f_{1(s)}$ = (beyaz vezir sayısı) – (siyah vezir sayısı), vs.

• Ağırlıklı toplam, bileşenlerin birbirinden bağımsız

olduğunu varsayar.

Satranç için değerlendirme fonksiyonu aşağıdaki sorulara cevap bulmaya çalışır.

• Asıl iş: f'leri ve w'leri belirlemek:

– Etrafı boş olan piyonlar kötüdür.

– Şahın korumaları var mı?

– Hakeret kabiliyetin nasıl?

– Tahtanın ortasının kontrolü sende mi?

– Oyun süresince w lerin değerleri değişir mi?

Yapay zeka kullanılan makalelerde eşitlik (3.1) de görülen değerlendirme fonksiyonuna benzer değerlendirme yöntemleri kullanmaktadırlar[8-13]. Yapılan uygulamada iki değerlendirme kriteri kullanılmıştır. Makalelerde daha çok kriter kullanılmıştır.

2.5.2 Taşların tahtada bulunduğu yere göre değerlendirme

Satranç oyununda tahtanın merkezde bulunan taşlar, tahtanın kenarında bulunan taşlardan daha fazla hareket yeteneğine sahiptir. Bu yüzden aynı türde bir taş dahi olsa merkezde bulunan bir taş, kenarda bulunan bir taşa göre daha değerlidir. Bu nedenle taşların tahtada bulundukları yere göre dinamik değerleri değişmektedir. Genel olarak merkezde bulunan taşların dinamik değerleri artarken kenarlarda azalmaktadır. Bu durum taşların türüne göre bazı farklılıklar gösterebilmektedir. Örneğin kale kenarlarda ve merkezde yaklaşık aynı hareket yeteneğine sahip olduğundan, kalenin merkezde olmasıyla çok fazla dinamik değeri değişmez Dinamik değerlendirmede her taş için ayrı bir değerlendirme dizisi oluşturulmuştur. Bu diziler taşların hareket özellikleri ve tahtanın merkezi dikkete alınarak tarafımızdan oluşturulmuştur. Bazı taşlar için yönleri farklı olduğundan siyah ve beyaz için ayrı ayrı diziler oluşturulmuştur.

Çizelge 2.1 : Beyaz kale için değerlendirme dizisi

0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
0.15	0.15	0.15	0.2	0.2	0.15	0.15	0.15
0.15	0.15	0.15	0.2	0.2	0.15	0.15	0.15
0.15	0.15	0.15	0.2	0.2	0.15	0.15	0.15
0.15	0.15	0.15	0.2	0.2	0.15	0.15	0.15
0.15	0.15	0.15	0.2	0.2	0.15	0.15	0.15
0.1	0.15	0.15	0.2	0.2	0.15	0.15	0.1

Çizelge 2.2 : Beyaz piyon için değerlendirme dizisi

0	0	0	0	0	0	0	0
0.05	0.1	0.15	0.2	0.2	0.1	0.1	0.05
0.04	0.08	0.12	0.16	0.16	0.08	0.08	0.04
0.03	0.06	0.09	0.12	0.12	0.06	0.06	0.03
0.02	0.04	0.06	0.08	0.08	0.04	0.04	0.02
0.01	0.02	0.03	-0.1	-0.1	0.02	0.02	0.01
0	0	0	-0.04	-0.04	0	0	0
0	0	0	0	0	0	0	0

2.5.3 Taşların statik değerlendirmesi

Taşların statik aşağıdaki puanlandırma sistemiyle yapılmaktadır. Kullanılan değerler satranç camiası tarafından kabul görmüş değerlerdir.

Çizelge 2.3 : Taşların statik değerlerinin puanlandırılması

Piyon= 1 puan	Piyon= 1 puan
At=3 Puan	At=3 Puan
Fil =3 puan	Fil =3 puan
Kale=4.5 puan	Kale=4.5 puan
vezir=9 puan	vezir=9 puan
şah puanı yok	şah puanı yok

2.6 Oyun ağacında yaprakların oluşturulması

Bir hamle yapıldıktan sonra hamle metod'u çağrılmaktadır. Bu metod rakibin yapabileceği tüm geçerli hamleleri bir diziye kaydeder. Hamle dizisi oluşturulurken, her hamle için değerlendirme metod'u çağırılır. Her bir taş için taşların statik ve dinamik değerleri toplanır. Taşların statik değerleri verilmiştir (Çizelge 2.3). Taşların dinamik değerleri için dinamik değerlendirme dizileri yazılmıştır. Örneğin beyaz kale için dinamik değerlendirme dizisi verilmiştir (Çizelge 2.1). Her bir taş farklı özelliğe sahip olduğundan herbir taşın dinamik değerlendirme dizisi birbirinden farklıdır. Bir döngü kullanılarak tahta dizisi baştan sona taranmaktadır. Taşın rengi ve çeşidine bakılmaktadır.

Örneğin oyunun başlangıcında beyazın e4 hamlesini yaptıktan sonraki konumda siyah'ın yapabileceği ilk hamle için değerlendirme işlemi görülmektedir

(Çizelge 2.4). Siyahın a7 karesinde durmakta olan siyah piyonunun a6 karesine oynaması sonucunda elde edilebilecek değerlendirme işlemi görülmektedir. Siyahın

a7 karesinin boş ve piyonun a6 karesinde olduğu düşünülerek değerlendirmeye başlanır. Döngü sıfır nolu indeksten başlar. Bu karade siyah kalenin olduğu tahta dizisinin değerine bakılarak bulunur. Siyah kalenin statik değeri bulunur

(Çizelge 2.3).

Dinamik değeri ise siyah kale değerlendirme dizisinin aynı numaralı indeks değeri alınarak, taşın statik değerine eklenir. Daha sonra indeks değeri bir artırılarak işleme devam edilir. Bu indekste siyah at bulunmaktadır. At'ın statik değeri çizelgeden bakılarak 3 olarak eklenir. Dinamik değeri ise atın dinamik değerlendirme dizisindeki aynı numaralı indekse bakılır. Bu değerin -0.3 olduğu bulunur. Bu iki değer eklenerek işleme devam edilir. Döngü a7 karesine geldiğinde, taşı a6'ya oynamış gibi düşündüğümüzden bu karede dinamik ve statik değer sıfırdır. Döngü a6 karesine geldiğinde ise siyah piyonun statik değerinin 1 ve piyonun dinamik değernin ise siyah piyon dinamik değerlendirme dizisindeki a6 da bulunan indeksdeki değerinin 0.01 olduğu bulunur. Bu değerler birbirine eklenir. Yapay zeka siyahlar için hamle üreteceğinden siyah taşların statik ve dinamik değerleri pozitif olarak alınır. Beyaz taşlar için ise statik ve dinamik değerlerinin negatif değer olduğu düşünülür. Döngüye devam ederek beyaz taşlar içinde taşların statik ve dinamik değerlerinin toplamını alarak işleme devam edilir. Döngü tamamlandığında a6 nolu yaprak için değerlendirme değeri oluşturulur. Oluşan bu değer minimaks değeriyle karşılaştırılır. Eğer minimaks değerinden daha büyük bir değer ise minimaks değeri bu değere eşitlenir. Ayrıca hareket değişkenine bu piyonun başlangıç indeksi ve bitiş indeksi kaydedilir.

Eğer bu değer minimaks değerinden daha büyük değil ise döngünün bir sonraki elemanına geçilir.

Çizelge 2.4 : İlk yaprak için değerlendirme işlemi

+4.5 +0.1	+3 +(-0.3)	+3 +(-0.2)	+9 +0.1	 +0.0	+3 +(-0.2)	+3 +(-0.3)	+4.5 +0.1
	+1 +0.0	+1 +0.0	+1 +(-0.04)	+1 +(-0.04)	+1 +0.0	+1 +0.0	+1 +0.0
+1 +0.01							
				-1 -0.08			
-1 -0.0	-1 -0.0	-1 -0.0	-1 -(-0.04)		-1 -0.0	-1 -0.0	-1 -0.0
-4.5 -0.1	-3 -(-0.3)	-3 -(-0.2)	-9 -0.1	 -0.0	-3 -(-0.2)	-3 -(-0.3)	-4.5 -0.1

Siyah'ın yapabileceği hamlelerden biri olan piyonun b7 karesinden b6 karesine hareketinin değerlendirilmesi görülmektedir (Çizelge 2.5). Yukarıdaki yaprak için yapılan değerlendirme işlem basamakları aynen burada da uygulanmaktadır. Tahta dizisinda kullanılan döngü yine baştan sona işletilir. Döngü başladığında yine taşların statik ve dinamik değerleri bulunarak bunların toplamları alınır. Döngünün ilk indeksinde siyah kale bulunmaktadır. Bu kalenin statik değeri bulunur (Çizelge 2.3). Dinamik değeri ise siyah kale dinamik dizisinin aynı numaralı indeksindeki değeridir. Bu iki değer toplanarak bir sonraki indeks değerine geçilir. Döngü devam edip a7 karesindeki indeks değerine geldiğinde bu kez siyah piyon'un orada olduğu düşünülerek statik değeri olan 1 sayısı ve dinamik değeri siyah piyon dinamik dizisindeki karşılığı olan sıfır değeri toplanır. Daha sonra a7 karesindeki indeks değerine geçilir. Bu kez de b7 piyonun'un b6 oynadığını düşünerek, bu karedeki statik ve dinamik değer sıfır alınır. Döngü devam edip sıra b6 karesindeki indeks değerine geldiğinde piyon'un statik değeri yerine 1 dinamik değerine de siyah piyon dinamik değerlendirme dizisindeki aynı indeksteki değeri karşılığı olan 0.02 değeri konulur. Aynı işlem beyaz taşlar içinde devam edilir. Yine beyaz taşların statik ve dinamik değerlerinin negatifleri toplamı olacak şekilde işleme devam edilmektedir.

Sonuçta bulunan değer bu hamle için yaprak düğüm değeridir. Bu değer minimaks algoritmasından geçirilir. Eğer minimaks değerinden daha büyük bir değer ise minimaks değeri bu değere eşitlenir. Hareket değişkenine bu piyonun başlangıç indeksi ve bitiş indeksi kaydedilmektedir.

Eğer bu değer minimaks değerinden daha büyük değil ise döngünün bir sonraki elemanına geçilmektedir.

Çizelge 2.5 : İkinci yaprak için değerlendirme işlemi

+4.5 +0.1	+3 +(-0.3)	+3 +(-0.2)	+9 +0.1	+0.0	+3 +(-0.2)	+3 +(-0.3)	+4.5 +0.1
+1 +0.0		+1 +0.0	+1 +(-0.04)	+1 +(-0.04)	+1 +0.0	+1 +0.0	+1 +0.0
	+1 +0.02						
				-1 -0.08			
-1 -0.0	-1 -0.0	-1 -0.0	-1 -(-0.04)		-1 -0.0	-1 -0.0	-1 -0.0
-4.5 -0.1	-3 -(-0.3)	-3 -(-0.2)	-9 -0.1	-0.0	-3 -(-0.2)	-3 -(-0.3)	-4.5 -0.1

Hamle dizisindeki tüm hamleler için yukarıdaki yapılan değerlendirme işlem basamakları uygulanmaktadır. Değerlendirme sonucu minimaks değerinden büyük

ise minimaks değeri ilgili sonuca eşitlenmektedir. Bu o hamlanin iyi bir hamle olduğunu gösterir. Aynı zamanda o hamlelerin indeks değerleri de hareket değişkeninde tutulduğundan döngü bittiğinde bu değişken en iyi hamlenin indeks değerini tutuyor olacaktır. Yapay zekanın yapacağı hamle de bu indeks değerinde tutulan hamle olacaktır.

Eduardu Vazquez 2011 yılındaki makalesinde satranç taşlarının değerleri aşağıdaki gibi puanlendırmıştır[14]:

Piyon: 100

At: 300

Fil: 330

Kale : 500

Vezir: 900

Şah: puanı yok

Taşların hareketliliğinin 0 ile 300 arasında değişmektedir. Değerler Yukarıda verilen makaledeki değerlere yakındır (Çizelge 2.3).

Tahta dizisi bir döngü ile taranmaktadır. Taşların tahtadaki yeri ve türü tespit edilmektedir. Taşların türüne göre statik değerlendirme deger değişkenine atanmaktadır. Taş piyon ise 1 at ise değeri 3 olmaktadır. Taşın bulunduğu indeksi kullanarak o taşa ait değerlendirme dizisi kullanılarak bulunan sonuç deger değişkenine eklenmektedir. Döngü bittiğinde o pozisyonun değeri bulunmuş olmaktadır. Bu değerlendirme sonucu minimax algoritmasında kullanılmaktadır.

2.7 Yapay zekanın yapacağı hamlenin seçilmesi

Bulunulan konumda , yani derinlik 1 iken; yapay zekanın yapabileceği hamleler içinde hangi hamlenin değerlendirme sonucu en büyük olan hamleyi bilgisayar oynamaktadır. Bilgisayarın yapacağı hamlenin indeks'i move değişkeninde depolanır. Bulunan bu değer hamle yap metoduna parametre olarak gönderilerek yapay zekanın hareket yapması sağlanmaktadır.

3. TASARIM VE UYGULAMA

3.1 Neden Java Kullanıldı

Nesne yönelimli bir programlama dili kullanmanın getireceği bazı avantajları bulunmaktadır. Bunlardan bazıları: Nesne yönelimli olmayan bir programlama diline göre daha az satır kod parçası yazarak aynı işin yapılması sağlanmaktadır. Daha az satır kod olması aynı zamanda programda oluşabilecek hata ihtimalini aza indirmektedir.

Uygulamada nesne tabanlı programlama dili olan java kullanılmıştır.

3.2 Oyunda Kullanılan Kısımlar

Kullanıcı ara birimi, taşların mouse ile hareketi, taşların kurallara göre hareket etmesi ve yapay zeka kısımlarından oluşmaktadır.

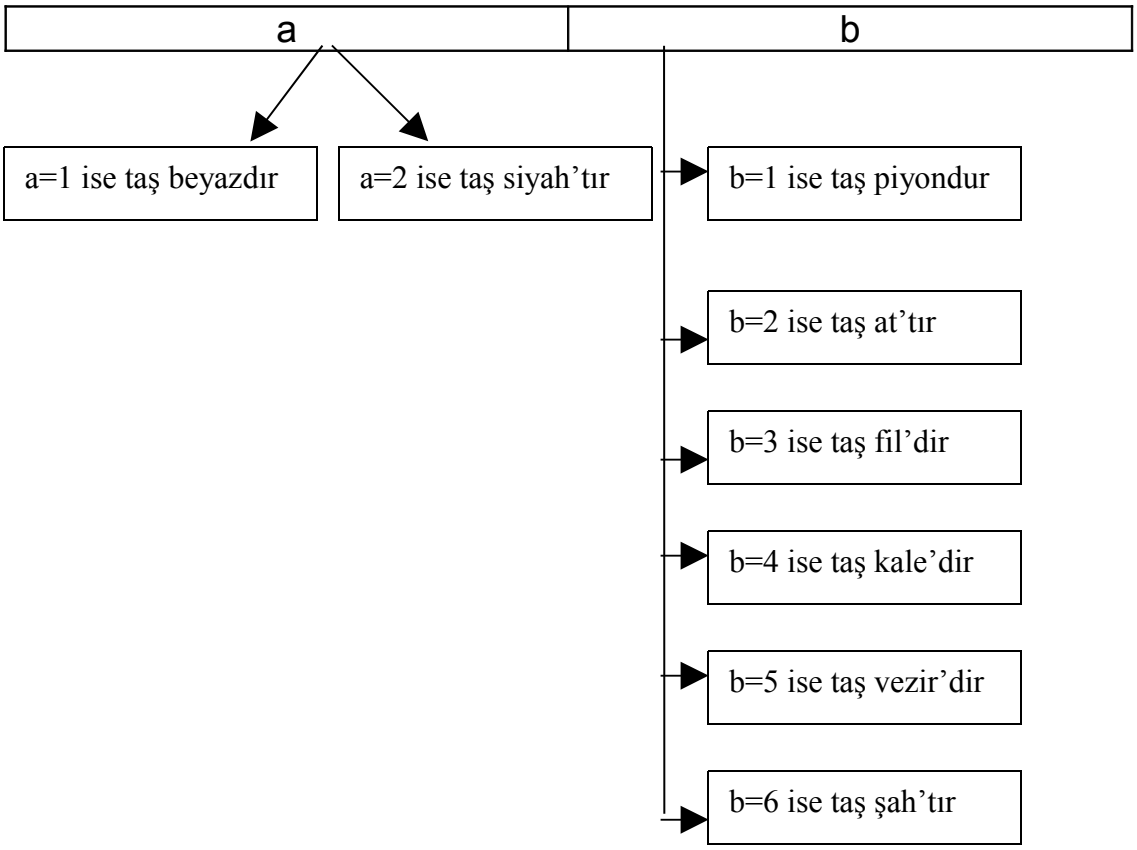
3.3 Görünüm

Yapılacak olan uygulama java programlama dili kullanılmıştır. Tahta isminde bir sınıf oluşturulmuştur ve bu sınıf JFrame sınıfından türetilmiştir. Böylece JFrame sınıfının metod'larını kullanabilir hale gelmiştir. Main metodu içinde bir nesnesi oluşturularak sınıfın yapıcı metod'unun çalıştırılması sağlanmıştır. Programın başlığı satranç şeklinde yazdırılmıştır. Frame'in görünür hale gelmesi sağlanmıştır. Programın penceresinde bulunan kapatma butonuna basıldığında kapanması sağlanmıştır. Satranç tahtasının büyüklüğü 600*600 piksel olarak ayarlanmıştır. Satranç tahtasının ekranın merkezine yerleştirilmesi sağlanmıştır. Satranç tahtasının büyüklüğünün kullanıcı tarafından değiştirilmesi önlenmiştir. Böylece tahtanın sabit bir büyüklükte ekranda kalarak, tahtanın görünümünün büyültülerek veya küçültülerek bozulması önlenmiştir. Bazı makaleler yalnızca görünüm(gui) kısmına sahiptir[13,15]. Bu makalede 2-d ve 3-d görünüm özelliklerine sahip kullanıcı ara birimlerine sahiptir. Yapılan uygulama 2-d özelliğe sahiptir.

3.4 Satranç Tahtasının Çizdirilmesi

Constructor içinde tahtanın çizdirilmesini sağlayacak başlat isimli metdod'u çağrılmıştır. Bu method içerisinde dizi uzunluğu 64 olan bir integer tipinde dizi

oluşturulmuştur ve bu dizinin değer atamaları yapılmıştır. Dizi içerisinde kullanılan integer değerler taşların rengini ve taşın tipini göstermektedir.



Şekil 3.1 : Dizideki kullanılan rakamların karşılıkları

Şekilde ab iki basamaklı bir sayıdır. A bu sayının onlar basamağını göstermektedir. b bu sayının birler basamağını göstermektedir.

Kullanılan sayının birler basamağı taşın tipini göstermektedir.

1 değeri taşın piyon olduğunu göstermektedir.

2 değeri taşın at olduğunu göstermektedir.

3 değeri taşın fil olduğunu göstermektedir.

4 değeri taşın kale olduğunu göstermektedir.

5 değeri taşın vezir olduğunu göstermektedir.

6 değeri taşın şah olduğunu göstermektedir.

Sayının onlar basamağı ise taşın rengini göstermektedir.

1 değeri taşın beyaz renkli olduğunu göstermektedir.

2 değeri taşın siyah renkli olduğunu göstermektedir.

Paint metod'u içerisinde bir döngü oluşturulmuştur. Satranç tahtasında 64 adet kare bulunduğundan, döngü sayısını 64 olarak ayarlanmıştır. Bu döngü içerisinde kare isimli metod çağrılmıştır. Bu metot integer bir parametre almaktadır. Bu parametre karenin indeks değerdir.

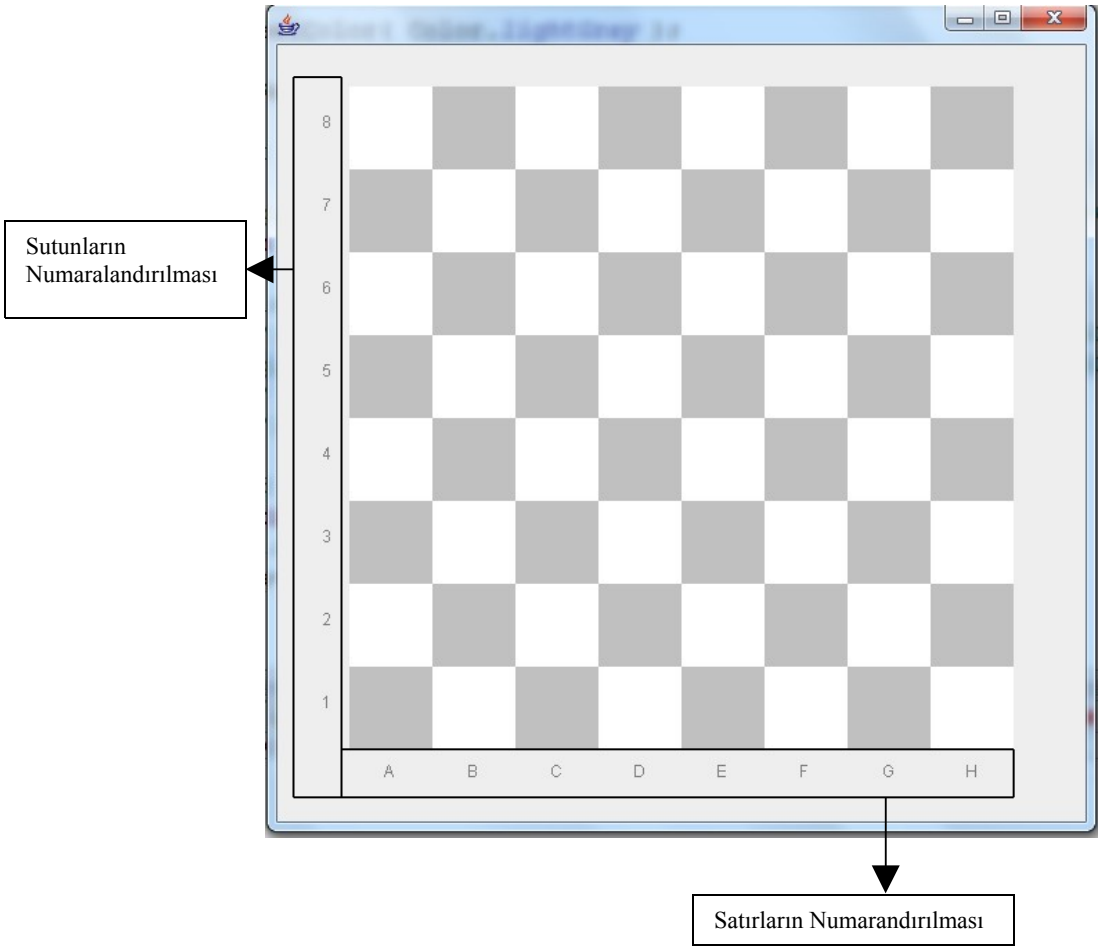
Bu parametre değerinin mod 8' i alındığında çizdirilecek olan karenin yatay indeks değerini vermektedir. Bu değer 0 ile 7 arasındadır.

Bu parametre değerini 8' e böldüğümüzde çizdirilecek olan karenin dikey indeks değerini vermektedir. Bu değer 0 ile 7 arasındadır.

Daha sonra yatay ve dikey indeks değerleri toplanıldığında ve toplam değerinin mod 2' ini alındığında çizdirilecek olan karenin rengi belirlenmiş olmaktadır. Eğer mod değeri 0 değerine eşitse o karenin rengi beyaz yapılmaktadır. Eğer mod değeri 1 değerine eşitse o karenin rengi açık gri yapılmaktadır.

Daha sonra grachics nesnesini kullanarak belirttiğimiz koordinatlarda ve belirlenen genişlik ve yükseklikte kare çizdirilmektedir. Programda karelerin genişlik ve yükseklik değerleri 60 pixel olarak ayarlanmıştır.

Paint methodunun içerisinde karelerin yatay ve dikeydeki yerlerini gösteren rakam ve harfleri grafik(graphics) nesnesini string metodu ile çizdirilmektedir. Yatay ve dikeyde 8 'er adet kare bulunduğundan döngü sayısını 8 olarak ayarlanmıştır. Döngü sayısının artış değerini dikeyde isimlendirme yapılırken kullanılmıştır.



Şekil 3.2 : Yapılan harflendirme ve numaralandırmalar görülmektedir.

Yataydaki isimlendirme isimlendirme içinse A,B,…H diye tanımlanan harf dizisinin elemanlarını kullanılmıştır.

3.5 Satranç Taşlarının Çizdirilmesi

Birden fazla taş çeşidi olduğundan bir resim(image) dizi nesnesi oluşturulmuştur. 6 adet beyaz taş ve 6 adette siyah taş çeşidi bulunmaktadır. Araç(Toolkit) sınıfını kullanarak image dizisinin elemanlarını tanımlamıştır. Bu tanımlamaları yaparken parametre olarak resmin dizin yolunu kullanılmıştır. Kare metod'unun içinde grafik (graphics) sınıfının resim çizdirme komutu kullanılarak karelerin içine taşlar çizdirilmektedir.

3.6 Satranç Taşlarının Başlangıç Konumlarının Çizdirilmesi

Taşların başlangıç konumlarının belirlenmesi kare metod'unun içinde image çizdirilirken yapılmaktadır. İmage çizdirme metod'unun aldığı ilk parametre image nesnesidir. Koordinatları ve de genişlik ve yükseklik değerleri kullanılmaktadır. Genişlik ve yükseklik değerleri karenin boyutlarıyla aynı olacak şekilde 60 piksel olarak ayarlanmıştır.

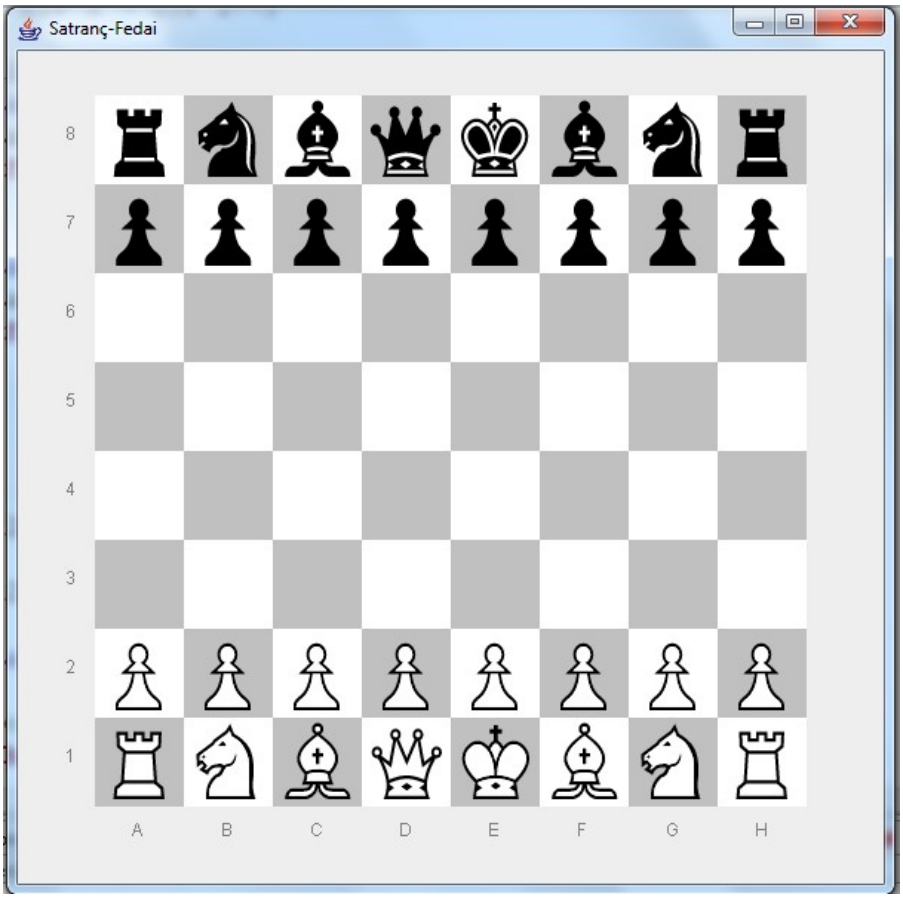
Taşların renginin belirlenmesi, taşların tipinin belirlenmesi ve bulunulan karede taş olup olmadığının belirlenmesi için integer tipinde ve dizi uzunluğu 64 olan tahta isimli bir dizi oluşturulmuştur. Karede taş olmadığını 0 sayısı belirtmektedir.

Çizelge 3.1 : Başlangıç konumunda dizinin değerleri

24	22	23	25	26	23	22	24
21	21	21	21	21	21	21	21
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
11	11	11	11	11	11	11	11
14	12	13	15	16	13	12	14

Bulunulan karenin indeks değerini tahta dizisinin indeks değeri olarak kullanılarak tahta dizisinin o indeksteki değerini elde edilmiştir. İmage dizisindeki beyaz taşları 1 ile 6 arasındaki indeks değerlerinde ve siyah taşlarıda 11 ile 16 arasındaki indeks değerlerinde bulunmaktadır. Tahta dizisinde beyaz taşlar 11 ile 16 arasındaki değerlere sahip ve de siyah taşlar 21 ile 26 arasındaki değerlere sahiptir. İmage dizisi ile tahta dizisi arasında bulunan 10 farkı çıkarılmış ve ve böylece image dizisindeki doğru indeks değerine ulaşılmıştır.

Taşın olmadığını belirtmek için 0 sayısını kullanılmıştır. Resim dizisinin 0 nolu indeks değerinde bir resim olmadığından boş bir resim çizdirilmektedir.



Şekil 3.3 : Başlangıç konumunun kullanıcı ara birimi görünümü

Başlat metod'u kullanılarak tahta dizisinin değerleri Çizelge 3.1 deki gibi atanmaktadır.

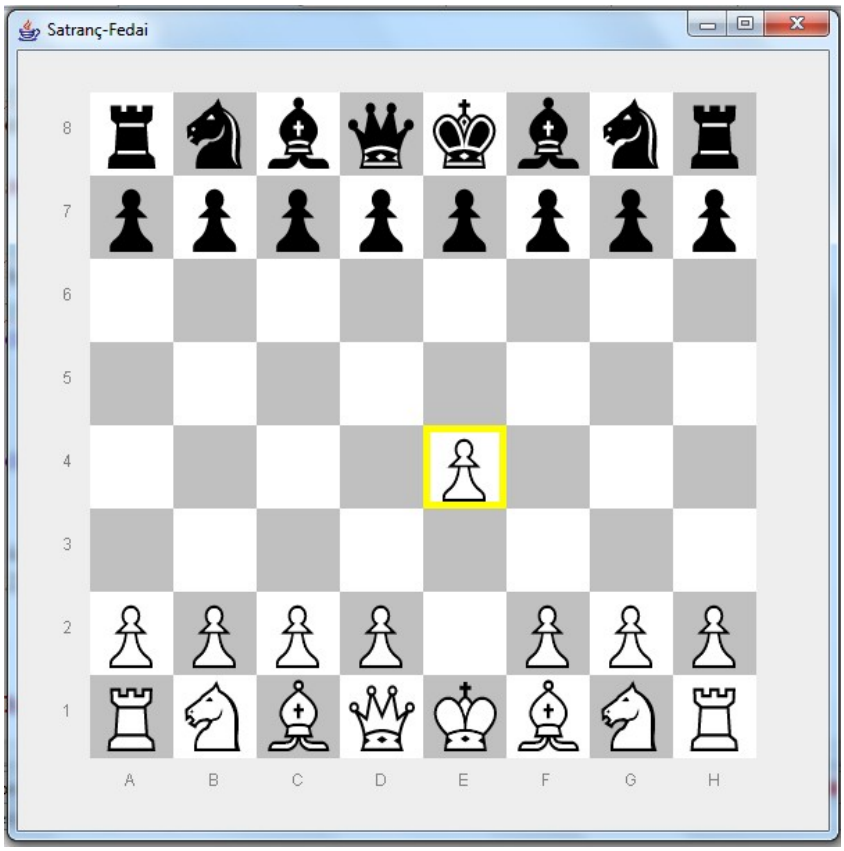
3.7 Taşların Hareketinin Programa Tanıtılması

Beyaz oyuncu başlangıç konumunda e4 piyon hamlesini yapmış olsun.

Çizelge 3.2 : Beyazın e4 hamlesi sonrası tahta dizisinin değerleri

24	22	23	25	26	23	22	24
21	21	21	21	21	21	21	21
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	11	0	0	0
0	0	0	0	0	0	0	0
11	11	11	11	0	11	11	11
14	12	13	15	16	13	12	14

Beyazın e4 hamlesi sonrasında tahta dizisinde e2 karesinde artık taş olmadığından bu kareye sıfır değeri yazılmıştır. Piyon e4 karesine hareket ettiğinden bu kareye denk gelen indeks değerine beyaz piyonu temsil eden 11 sayısı yazılmıştır. Programda taşların hareketi bu şekilde sağlanmaktadır.



Şekil 3.4 : Beyaz e4 hamlesi sonrası kullanıcı ara birimi görünümü

Beyazın e2 karesinde taşı olmadığından bu karede bulunan taşı silinmiştir. Bu taş hareket ettirilmiş olduğu e4 karesine çizilmiştir. Bu sayede taşın kullanıcı ara birimindeki hareketi sağlanmıştır.

3.8 Satranç Taşlarının Bilgisayar Faresi Yardımıyla Hareket Ettirilmesi

Sınıfı oluştururken bilgisayar faresi dinleyicisi (mouselistener) ve bilgisayar faresi hareket dinleyicisinden (mousemotionlistener) türetilmiştir. Boş metod'lardan bilgisayar faresi tuşu basıldığında (mousepressed) ve bilgisayar faresi tuşu bırakıldığında mousereleased metodlarını kullanılmıştır. Bu metodlar içinde mouse'un bulunduğu noktanın piksel değerini alınmıştır. Bu değeri karenin genişlik ve yükseklik değeri olan 60 değerine bölündüğünde o karenin 0 ile 7 arasında yatayda bulunan indeks değerini bulunmaktadır. Aynı şekilde mouse'nin bulunduğu noktanın dikey piksel değerini alarak bu değeri 60 değerine bölünerek o karenin 0 ile 7 arasındaki dikeyde bulunan indeks değerini bulunmaktadır. Bulunulan karenin indeks değerini hesaplarken dikey indeks değerini 8 ile çarptıktan sonra yatay indeks değeriyle toplanmaktadır.

Mouse'nin bulunduğu nokta satranç tahtasının dışında ise bunu bulunulan noktanın piksel değerini satranç tahtasının başlangıç ve bitiş değerleriyle karşılaştırılmaktadır. Eğer Mouse tahtanın dışındaysa indeks değerini -1 sayısına eşitlenmektedir.

3.8.1 Bilgisayar faresinin tuşu basıldığında çalışan metot (mousepressed)

Bilgisayar faresi tıklandığında bu metod çalışmaktadır. Bilgisayar faresinin bulunduğu kareye sarı renkli bir çerçeve çizdirilmektedir. Çerçeve çizdirilirken önce sarı renkte büyük bir kare çizdirilmektedir, sonra bilgisayar faresinin bulunduğu karenin rengini daha küçük bir kare olarak çizdirilmektedir.

Daha önce çizilmiş olan çerçeveyi kaldırmak için kare metod'una o karenin indeks değerini göndererek, o karede bulunan çerçeve kaldırılmaktadır.

3.8.2 Bilgisayar faresinin tuşu sürüklendiğinde çalışan metot (mousedrag)

Bu metod kullanılarak taşların hareketi sağlanmaktadır. Bu genel olarak java animasyon yöntemidir. Bilgisayar faresi'nin her hareket ettiği pikselde satranç tahtası tamamıyla yeniden çizdirilerek taşların hareket etmesi sağlanmaktadır. Sürükleme yapılırken mouse'un bulunduğu yatay ve dikey eksendeki konumu alınmaktadır. Bu konuma sürüklenen taş çizdirilmektedir.

3.8.3 Bilgisayar faresinin tuşu bırakıldığında çalışan metot (mousereleased)

Bilgisayar faresi bırakıldığında çalışan metot'tur . Bilgisayar faresi bırakıldığında bulunulan kareye sarı renkli bir çerçeve çizdirilmektedir. Bulunulan kareye önce sarı renkte büyük bir kare çizdirilmektedir, sonra mouse'nin bulunduğu karenin rengini daha küçük bir kare olarak çizdirilmektedir.

Bilgisayar faresi tıklandığında karenin indeks değerini kullanarak tahta dizisinin o indeksteki değerini, tahta dizisinin bilgisayar faresi bırakıldığındaki karenin indeks değerine eşitllenmektedir. Daha sonra grafik (graphics) nesnesini kullanarak o kareye resim (image) çizdirilmektedir.

3.9 Taşın önceki yerinin silinmesi

Tahta dizisinin mouse basıldığındaki indeks değerine 0 sayısı atanmaktadır. Böylece artık o kare boş bir kare olarak değerlendirilmektedir. Kare metod'una indeks değeri gönderildiğinde boş bir kare çizdirilmektedir.

3.10 Uyarı mesajlarının verilmesi

Koordinatları satranç tahtasının merkezine gelecek şekilde , genişliği 100 piksel ve yüksekliği 50 piksel olan sarı renkte bir kare çizdirilmiştir. Bu karenin içine verilecek olan uyarı mesajını string olarak yazdırılmaktadır. Yazılan yazının rengini kırmızı olarak ayarlanmıştır. Daha önceki karenin çerçevesini kaldırmak için o karenin indeks değerini kare metod'una gönderilmiştir. Uyarının belli bir süre ekranda kalıp daha sonra kendiliğinden gitmesi şeklinde ayarlanmıştır. Uyarı süresi olarak 3 saniye ayarlanmıştır. Bu sürenin sonunda repaint metod'unu kullanarak uyarı mesajı kaldırılarak tahtanın tekrar eski haline dönmesini sağlanmıştır.

3.11 Ses efekti

Ses için ses metod'u yazılmıştır. Bu metod aslında java dosya okuma işlemi yapmaktadır. Bu metod'ta ses dosyası okunmaktadır. Uygulamada waw uzantılı ses dosyası kullanılmıştır. Bu metod hamle yap metod'unun içinde çağrılmasından dolayı, geçerli bir hamle yapıldığında bu metod çağrılmakta ve ses verilmesi sağlanmaktadır. Uygulamada bir saniye süreli bir ses dosyası kullnılmıştır.

3.12 Satranç Taşlarının Kurallara Göre Hareket Etmesi

Taşın bulunduğu karenin indeks'ini ve taş'ın hareket ettirilmek istendiği karenin indeks değerlerini kullanarak tüm taşların yapabilecekleri bütün hamleleri bir dizinin içine kaydedilmektedir. Bunun için hareket isimli bir methot yazılmıştır. Bu methot tüm olasılıklara bakarak yapılabilecek hamleleri belirlemektedir.

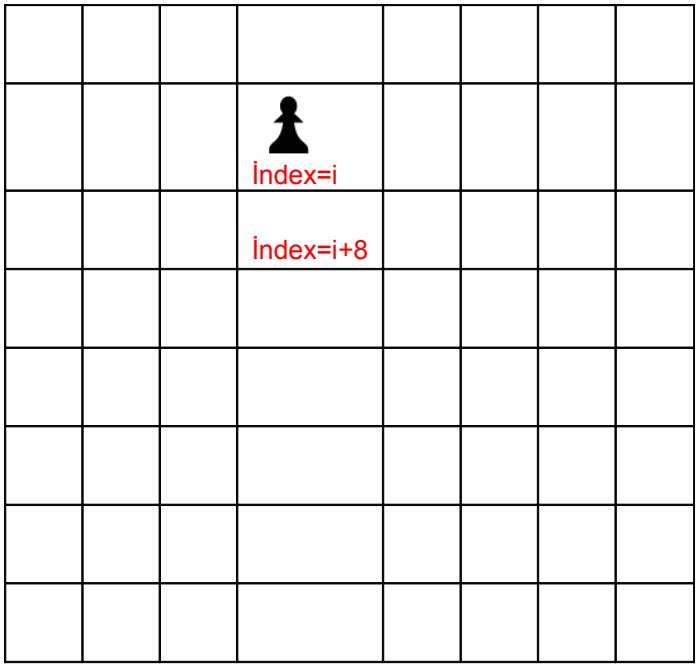
Bu işlem satranç tahtasında bulunan tüm kareler bir döngü ile taranarak yapılmaktadır. Oynayacak tarafın taşlarının rengi ile aynı renkteki taşları bir koşul kullanılmıştır. Bu sayede taşların kendi renginde olan bir taş'ı alması önlenmektedir. Daha sonra o karede bulunan taşın cinsini belirlenmektedir. Bir switch case yapısı kullanarak her taş cinsi için değişik case durumları oluşturulmuştur.

3.12.1 Piyon

Eğer taş piyon ise 4 değişik hareket olasılığı bulunmaktadır.

3.12.1.1 Piyonun bir kare önündeki kare boş ise

Beyaz piyon için bulunduğu karenin ve önündeki karenin indeksleri görülmektedir (Şekil C.1).



Şekil 3.5 : Piyonun bulunduğu karenin ve önündeki karenin indeksleri

3.12.1.2 Piyonun iki kare ileri hareket etmesi

Piyonun iki kare ileri hareket etmesi için bir kare önündeki ve iki kare ilerisindeki karenin boş olması gerekmektedir. Siyah piyon için bulunduğu karenin ve önündeki karenin ve iki kare ilerideki karenin indexleri görülmektedir (Şekil C.2). Beyaz piyon için bulunduğu karenin ve önündeki karenin ve iki kare ilerideki karenin indeksleri görülmektedir (Şekil C.3).

3.12.1.3 Piyonun sol çaprazındaki kareye hareket etmesi

Piyonun sol çaprazındaki kareye hareket etmesi için piyonun sol çaprazında bulunan karede rakip bir taşın bulunduğunu kontrol edilmektedir. Siyah piyon için bulunduğu karenin ve sol çaprazındaki indeksleri görülmektedir (ŞekilC.4). Beyaz piyon için bulunduğu karenin ve sol çaprazındaki indeksleri görülmektedir (ŞekilC.5).

3.12.1.4 Piyonun sağ çaprazındaki kareye hareket etmesi

Piyonun sağ çaprazındaki kareye hareket etmesi için piyonun sağ çaprazında bulunan karede rakip bir taşın bulunduğunu kontrol edilmektedir. Siyah piyon için bulunduğu karenin ve sağ çaprazındaki indeksleri görülmektedir (ŞekilC.6). Beyaz piyon için bulunduğu karenin ve sağ çaprazındaki indeksleri görülmektedir (ŞekilC.7).

Eğer piyon bu hamlelerden herhangi birini yaparsa bu hamleyi hamlelistesi dizinine kaydedilmektedir. Hamle listesi dizisi oynayacak olan tarafın yapılabilecek tüm hamlelerin kaydedildiği dizidir. Daha sonra bu dizi kişinin yaptığı hamlenin doğru hamle olup olmadığının kontrolünde kullanılmaktadır.

3.12.2 At

At'ın L şeklinde bir hareketi vardır. Toplamda maksimum 8 kareye gidebilir. At'ın bulunduğu ve gidebileceği karelerin indexleri şekilde gösterilmiştir.

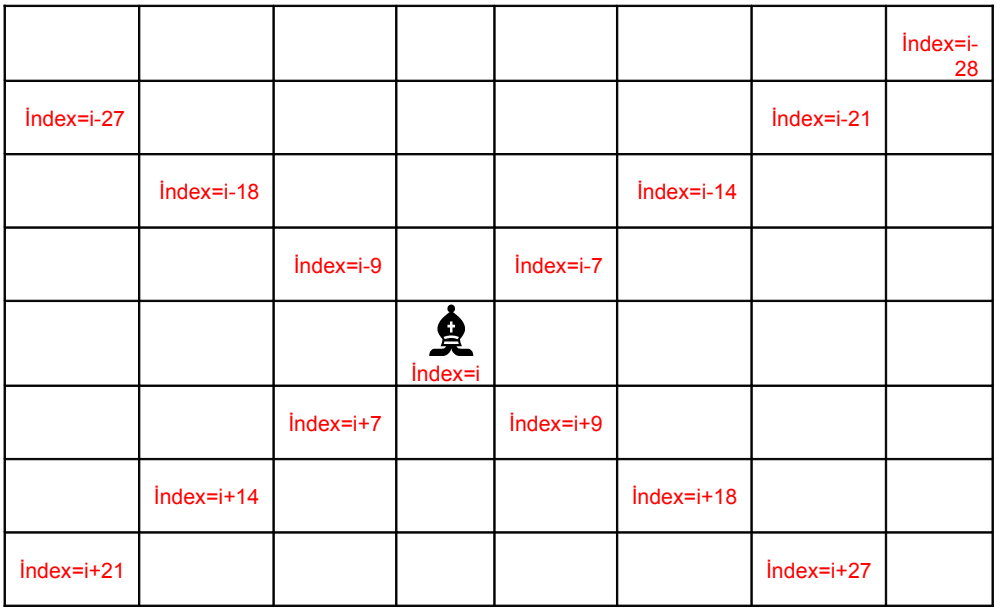
		İndex=i-17		İndex=i-15			
	İndex=i-10				İndex=i-6		
			 İndex=i				
	İndex=i+6				İndex=i+10		
		İndex=i+15		İndex=i+17			

Şekil 3.6 : At için bulunduğu karenin ve gidebileceği karelerin indeksleri

Atın gidebileceği kareler tahtanın içinde ise bu hamleler hamle listesi dizisine kaydedilmektedir.

3.12.3 Fil

Fil çaprazda ve aynı renklerde hareket edebilen bir taştır. Fil sadece çevresindeki karelere değil bir hat boyunca hareket ettiğinden dolayı bir döngü kullanarak bu hat boyunca yapabileceği tüm hamleler hamle listesi dizisine kayıt edilmiştir.



Şekil 3.7 : Filin bulunduğu karenin ve çaprazlarda gidebileceği karelerin indeksleri

3.12.4 Kale

Kale düz bir hat boyunca yatay veya dikey hareket edebilen bir taştır. Kale de fil gibi sadece çevresindeki karelere değil bir hat boyunca hareket ettiğinden dolayı bir döngü kullanarak o hat boyunca hareket etmesini sağlanmaktadır. Kalenin yapabileceği tüm hamleler hamle listesi dizisine kayıt edilmektedir.

			İndex=i-32				
			İndex=i-24				
			İndex=i-16				
			İndex=i-8				
İndex=i-3	İndex=i-2	İndex=i-1	 İndex=i	İndex=i+1	İndex=i+2	İndex=i+3	İndex=i+4
			İndex=i+8				
			İndex=i+16				
			İndex=i+24				

Şekil 3.8 : Kale bulunduğu karenin ve çaprazlarda gidebileceği karelerin indeksleri

3.12.5 Vezir

Vezir hem fil gibi bir çapraz boyunca aynı renkte hareket edebilen bir taştır ve de aynı zamanda kale gibi bir hat boyunca yatay veya dikey hareket edebilen bir taş olduğundan bu taşın hareketi fil ile kalenin bir bileşimidir. Fil ve Kale için gerekli case durumlarını yazıldığından dolayı vezir için ayrıca bir kod yazmaya gerek kalmamıştır. Aşağıdaki şekilde vezirin bulunduğu ve gidebileceği kareler gösterilmiştir. Vezirin yapabileceği tüm hamleler hamle listesi dizisine kayıt edilmektedir.

			İndex=i-32				İndex=i-28

İndex=i-27			İndex=i-24			İndex=i-21	
	İndex=i-18		İndex=i-16		İndex=i-14		
		İndex=i-9	İndex=i-8	İndex=i-7			
İndex=i-3	İndex=i-2	İndex=i-1	 İndex=i	İndex=i+1	İndex=i+2	İndex=i+3	İndex=i+4
		İndex=i+7	İndex=i+8	İndex=i+9			
	İndex=i+14		İndex=i+16		İndex=i+18		
İndex=i+21			İndex=i+24			İndex=i+27	

Şekil 3.9 : Vezirin bulunduğu karenin ve gidebileceği karelerin indeksleri

3.12.6 Şah

Sadece çevresindeki karelere bir kare ilerleyebilen bir taştır. Şah'ın bulunduğu ve gidebileceği kareler şekilde gösterilmiştir. Şah'ın yapabileceği Tüm hamleler hamle listesi dizisine kayıt edilmektedir.

		İndex=i-9	İndex=i-8	İndex=i-7			
		İndex=i-1	 İndex=i	İndex=i+1			
		İndex=i+7	İndex=i+8	İndex=i+9			

Şekil 3.10 : Şahın bulunduğu karenin ve gidebileceği karelerin indeksleri

3.12.7 Yapılan Hamlenin Doğru Hamle Olup Olmadığının Kontrolü

Mouse listener ile alınan hamlenin başlangıç ve bitiş indeks'leri alınmaktadır. Bu indeks değerleri kullanılarak yapılan hamlenin geçerli bir hamle olup olmadığının

kontrolü gecerli() isimli method'ta yapılmaktadır. Hamlenin indeks'leri kullanılarak bir hesaplama yapılmaktadır. Yapılan hesaplama sonucunu geçerli() method'una gönderilmektedir. Gelen değer bir döngü yardımıyla hamle listesi dizisinde aranmaktadır. Bu dizide bir eşleşme varsa yapılan hamle geçerli bir hamledir.

3.12.8 Yanlış Hamle Yapıldığında Uyarının Verilmesi

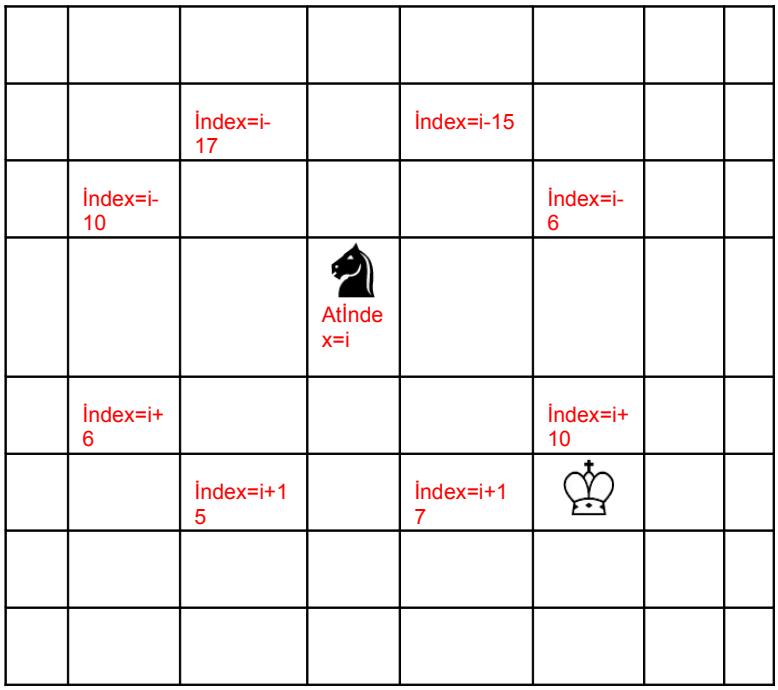
Yapılan hamle geçerli bir hamle değilse, uyari isimli metod çağırılır. Bu method ekrana yanlış hamle uyarısının yazıldığı metoddur. Bu method parametre olarak string bir değer almaktadır. String değer olarak "Yanlış Hamle" string değeri gönderilmektedir. Bu uyarıyı belirli bir süre ekranda gösterilmektedir. Daha sonra uyarı ekrandan silinmektedir.

Şekil 3.11 : Yanlış hamle yapıldığında ekrana gelen uyarı

3.12.9 Şah Çekilip Çekilmediğinin Kontrolü

Şah çekilip çekilmediğinin kontrolü için şah isimli bir method yazılmıştır. Bu method eğer şah çekilmişse true, şah çekilmemişse false değerini döndürmektedir. Bu methodun içinde ilk önce şahın bulunduğu indeksi board dizisi içinden aratarak bulmaktadır. İndeks değeri olarak şahın bulunduğu indeks değeri verildiğinden dolayı, eğer rakip taşların herhangi birisi şah'ı alabilecek durumdaysa şah çekilmiş olmaktadır. Bu metod şah çekildiğinde true değeri, şah çekilmediği durumda false

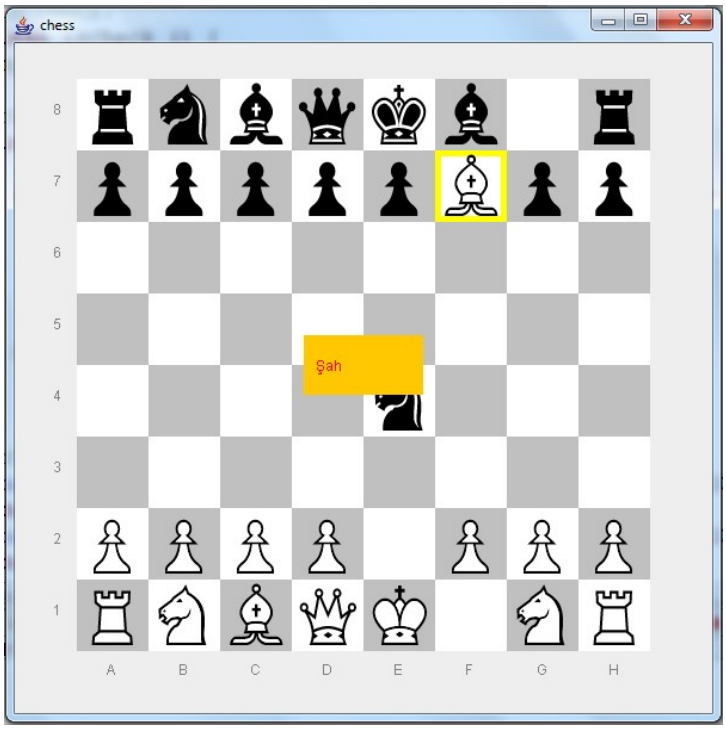
değeri döndürmektedir At'ın gidebileceği indeks değerleri şah'ın bulunduğu indeks değerine eşitse, at şah çekilmiş olur (Şekil 3.12).



Şekil 3.12 : At için bulunduğu karenin ve gidebileceği karelerin indeksleri

3.12.10 Şah Çekildiğinde Uyarının Verilmesi

Şah isimli metod çağırılarak, şah çekilip çekilmediğinin kontrolü yapılmaktadır. Bu metod true değeri döndürmüşse uyarı metodu çağırılmaktadır. Metod parametre değeri olarak şah string'ini almaktadır. Böylece ekrana şah çekilmiş olduğunu gösteren uyarı mesajı ekrana gelmektedir.



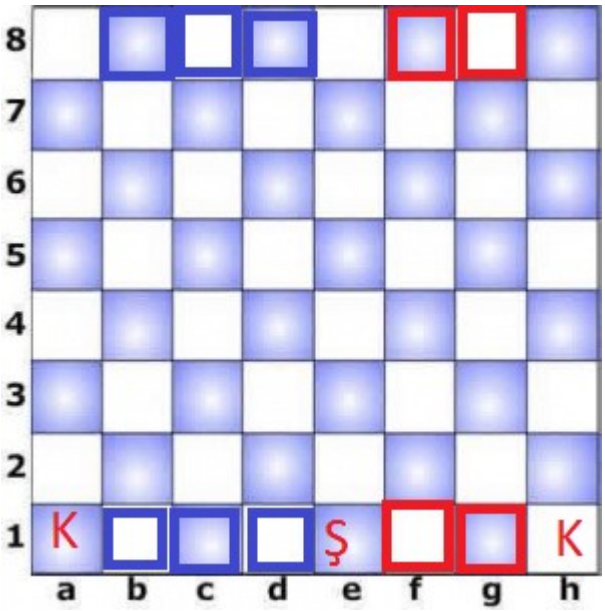
Şekil 3.13 : Şah çekildiğinde ekrana gelen uyarı mesajı

3.12.11 Rok Hareketinin Yapılması

Büyük rok hamlesi ve küçük rok hamlesi iki ayrı koşul ile yapılmaktadır.

3.12.11.1 Küçük rok hamlesi

Şahın bulunduğu karenin solundaki ve 2 kare solundaki karenin boş olup olmadığının ve ayrıca sol köşede kale'nin bulunup bulunmadığının kontrolü yapılmaktadır. Daha sonra şah bir kare sola alınmaktadır ve şah çekilip çekilmediği sahmı isimli metod ile kontrol edilir. Bu metod false döndürmüşse şah bir solundaki kareye konularak bu kare'ninde şah kontrolü yapılmaktadır. Daha sonra sol köşede bulunan kale şah'ın sağ tarafına konularak rok işlemi tamamlanmaktadır.



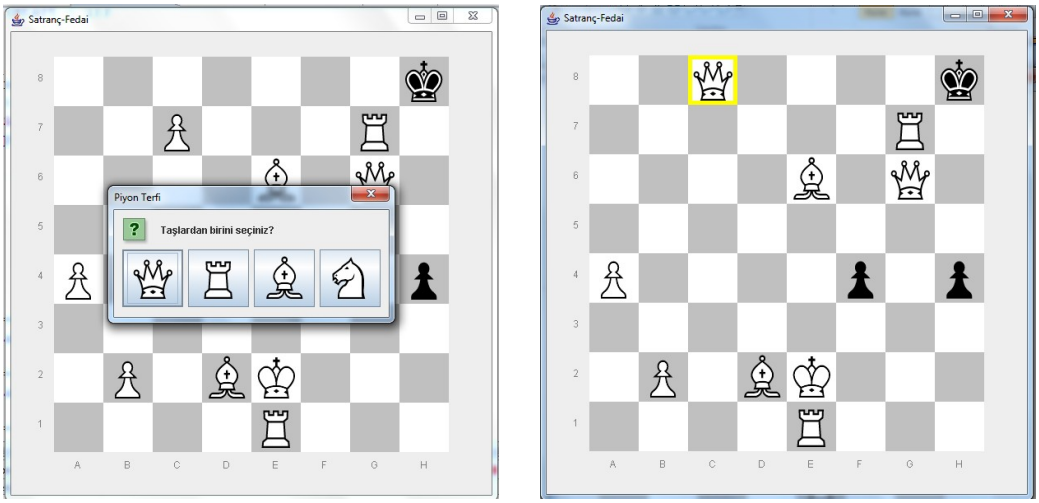
Şekil 3.14 : Küçük rok'ta kırmızı, büyük rok'ta mavi kareler kontrol edilir

3.12.11.2 Büyük rok hamlesi

Şah'ın bulunduğu karenin sağ tarafındaki karenin, 2 kare sağındaki karenin ve 3 kare sağındaki karenin boş olup olmadığının kontrolü yapılır. Sağ köşede kalenin bulunup bulunmadığının kontrolü yapılmaktadır. Şartlar sağlanmışsa şah bir kare sağa alınarak ve şah çekilip çekilmediğinin kontrolü yapılmaktadır. Şah çekilmemişse şah 2 kare sağındaki kareye konulur ve bu karenin şah kontrolü yapıldıktan sonra sağ köşedeki kale şah'ın sol tarafına konularak rok işlemi tamamlanmaktadır.

3.12.12 Piyon dönüşümü

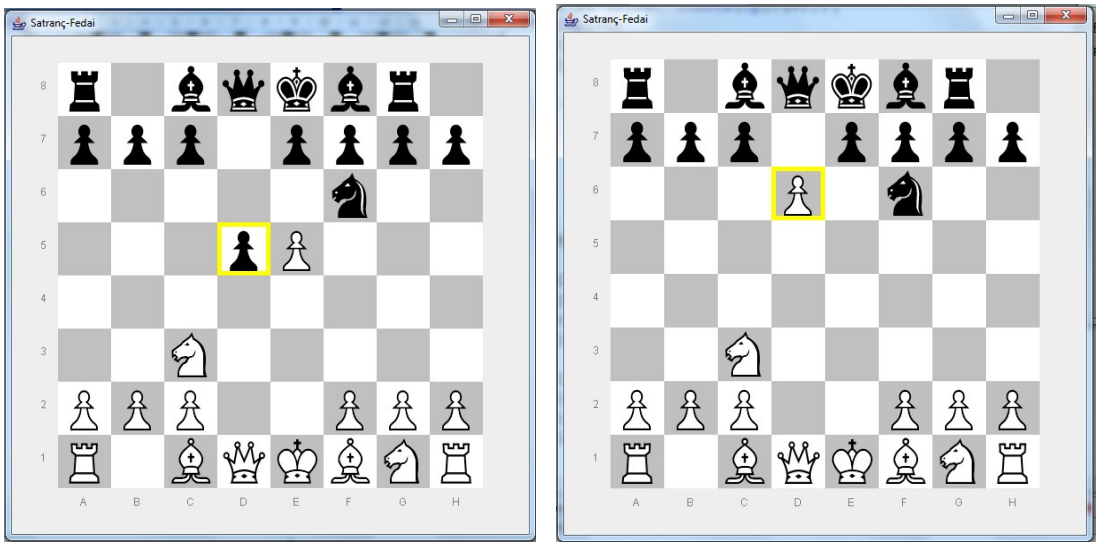
Beyaz piyonlar için 8.yataya , siyah piyonlar için ise 1.yataya piyonun ulaşması durumunda piyon vezir, kale, At yada filden biri seçilerek o taşa dönüştürülür. Bu işlem için hareket ettirilen karenin 8. veya 1. yatayda olup olmadığının kontrolü yapılmaktadır. Eğer şart sağlanıyorsa ekrana gelen dialog penceresinden seçilmek istenilen taşın bulunduğu icon'a tıklanılarak piyonun seçilen icondaki taş'a dönüşümü sağlanmaktadır.



Şekil 3.15 : Beyaz piyon 8. yataya ulaştığında ekrana gelen dialog penceresi soldaki şekilde görülmektedir. Dialog penceresinden vezir seçildiğinde pozisyonun konumu sağda görülmektedir.

3.12.13 Geçerken Alma Kuralı

Geçerken alma kuralı piyon metodunun içinde uygulanmaktadır. Oyunda herhangi bir taraf oyunda piyonu 2 kare hareket ettirdiğinde, hareket edilen konumun indeks'i bir değişkende tutulur ve bu değişkenin değeri eğer piyon 2 kare hareket ettirilmemişse sıfır değerine eşitlenmektedir. Bir piyon hareketi yapıldığında bu piyonun sağ'ındaki veya solundaki karenin indeks değeri, geçer piyonun indeksinin tutulduğu değişkenin değerine eşitse, alınmak istenilen karedeki piyonun bir kare gerisine piyon konularak, rakip piyon tahtadan alınmaktadır.



Şekil 3.16 : Siyah d5 oynadığındaki durum solda görülmektedir. Sağda beyaz e5 piyonu ile siyahın d5 karesindeki piyonunu alarak d6 oynar.

3.12.14 Oyunun Bitişi

A-Beyazın yada siyahın kazanması durumu

B-Oyunun berabere bitmesi

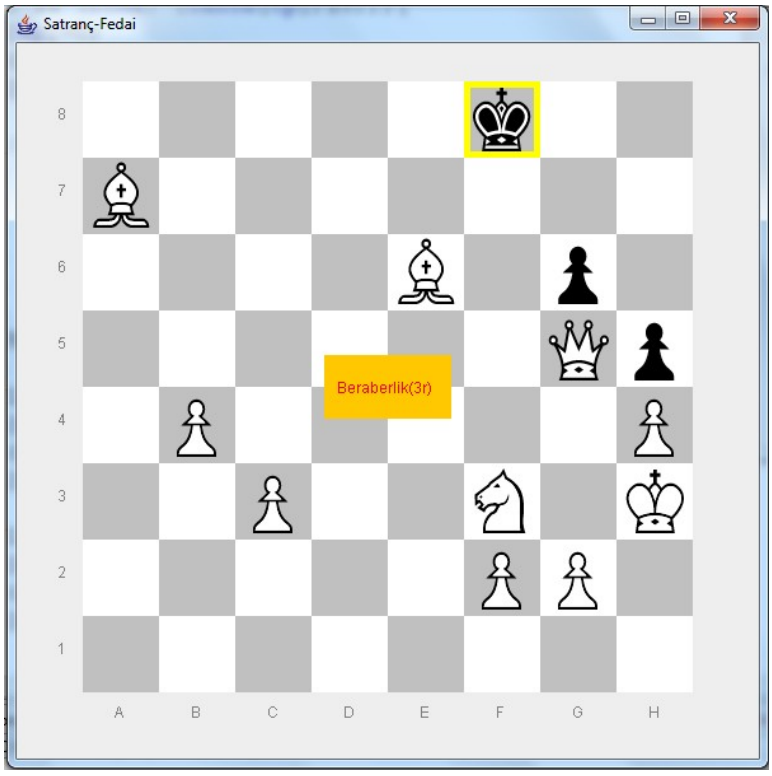
a-Şah çekilmediği durumda rakibin oynayacak hamlesinin olmaması

b- Elli Hamle Kuralı

c-Üç kez pozisyon tekrarı ile oyunun bitmesi

3.12.14.1 Üç kez pozisyon tekrarı ile oyunun bitmesi

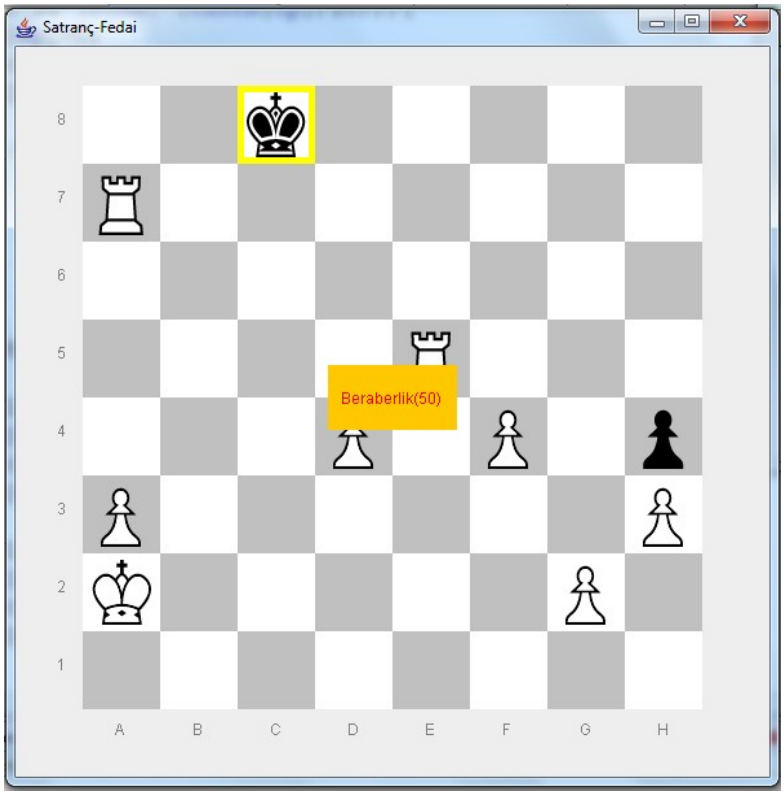
Aynı pozisyon üç kez tekrarlandığında oyun berabere bitmektedir. Bu tekrarların peş peşe olması gerekli değildir. Bu kural hamle yap metodunun içinde uygulanmaktadır. Her hamle yapıldığında, yapılan hamle piyon hamlesi değilse yada taş değişimi olmamışsa pozisyonun konumu bir vector içinde tutulmaktadır. Vector içinde tutulan pozisyon değerlerinin eşitliği bir döngü yardımı ile kontrol edilmektedir. Eğer bu eşitliklerin sayısı 3 değerine ulaşmışsa oyun beraberlik ile bitmekte ve ekrana "Beraberlik(3r)" şeklinde bir uyarı gelmektedir. Bu uyarının anlamı oyunun 3 kez pozisyon tekrarı ile bittiğini göstermektedir.



Şekil 3.17 : Üç hamle tekrarı ile oyunun bitmesi görülmektedir.

3.12.14.2 Elli Hamle Kuralı

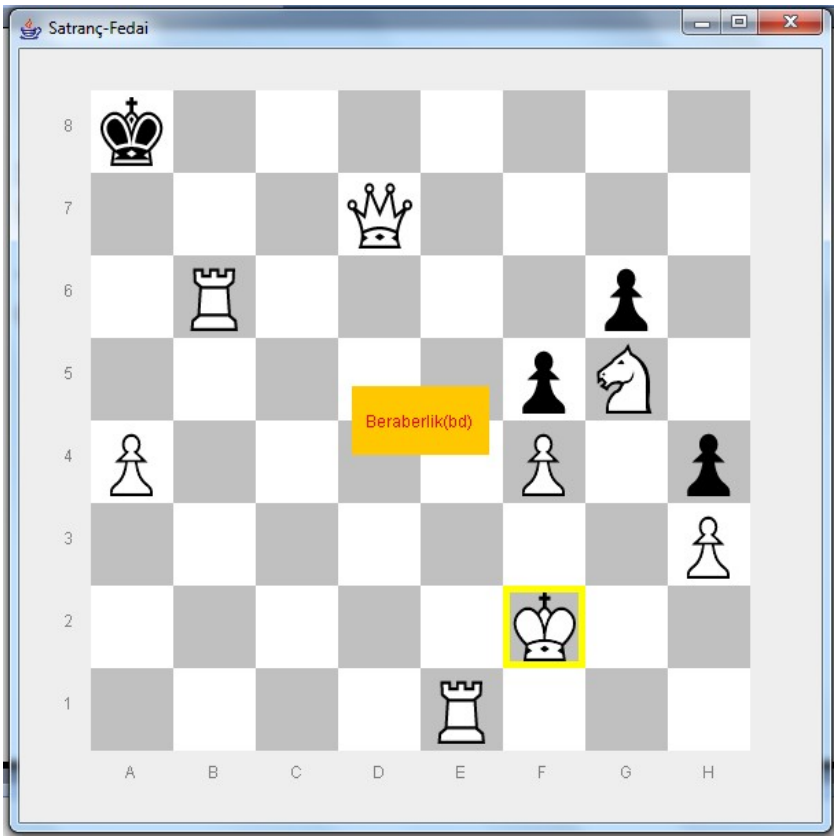
Bir piyon hareketi veya bir taş değişimi olmadığı zaman taraflardan birisi 50 hamle oynadığında oyun beraberlik ile bitmektedir. Bu kontrol hamle yap metodunun içinde yapılmaktadır. Önce yapılan hamlenin piyon hamlesi olup olmadığını ve yapılan hamlede bir taş değişiminin olup olmadığı kontrol edilmektedir. Bir sayaç değişkeni kullanarak eğer piyon hamlesi yapılmışsa ya da taş değişimi olmuş ise bu değişken 1 değerine eşitlenmektedir. Şartlar sağlanmamışsa sayaç değişkeninin değeri bir artırılmaktadır. Sayaç değişkeninin değerinin 50 değerinden küçük olup olmadığı kontrol edilmektedir. Eğer bu değer 50 değerine eşit yada büyük ise oyun beraberlik ile bitmektedir. Ekrana "Beraberlik(50)" yazan bir uyarı gelmektedir. Bu uyarı oyunun 50 hamle kuralına göre bittiğini göstermektedir.



Şekil 3.18 : Elli hamle kuralına göre oyunun bitmesi

3.12.14.3 Diğer berabere durumları

Rakip şah çekmediği durumda oynanacak geçerli hamle olmadığı durumda oyun berabere bitmektedir. İngilizcede 'basic draw denilen durumdur. Her hamle sonrası rakibin yapabileceği tüm geçerli hamleler hamle metod'u çağırılarak hamle dizisi oluşturulmaktadır. Bu hamle dizisinin eleman sayısı sıfır ise, rakibin şah çekip çekmediğine bakılmaktadır. Rakip şah çekmemiş ise oyun berabere bitmektedir.



Şekil 3.19 : Şah çekilmediği durumda siyahın oynayacak geçerli hamlesi yok

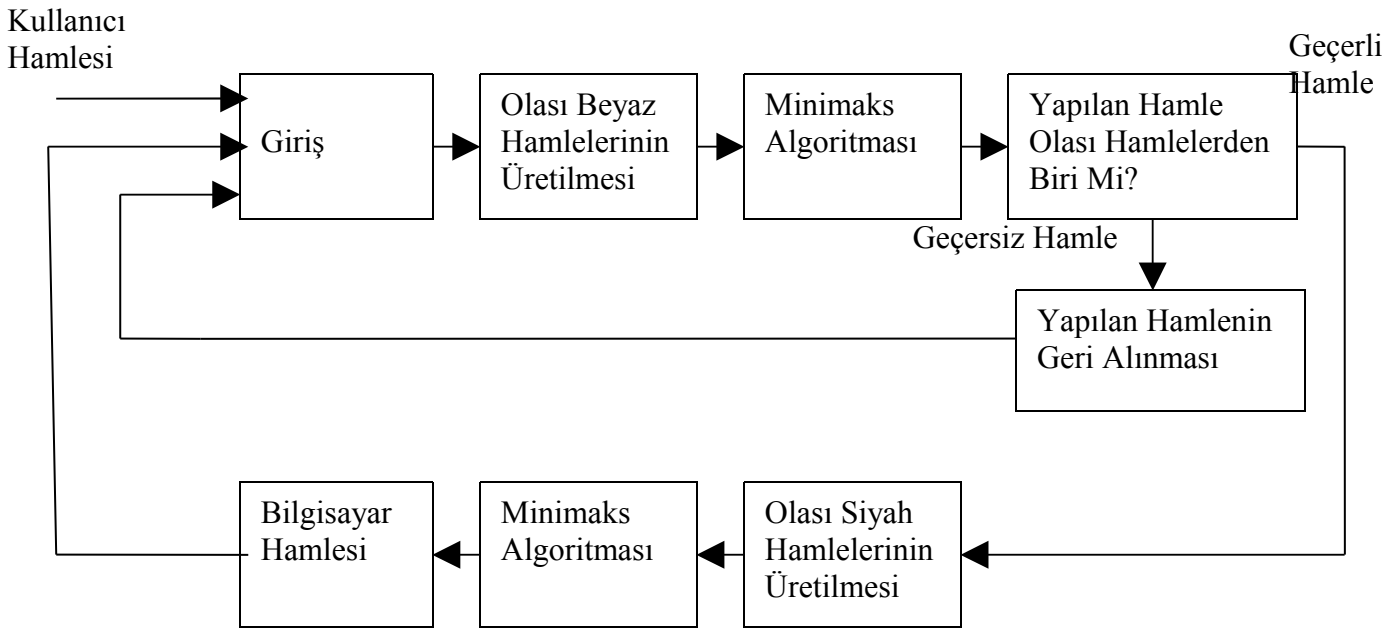
3.12.15 Beyazın yada siyahın kazanması durumu

Şah çekildiği durumda rakibin yapacak geçerli bir hamlesi kalmamış ise oyunu şah çeken taraf kazanmış olmaktadır. Şah çeken taraf hamlesini yaptıktan sonra, hamle metod'u çağırılarak, rakibin yapabileceği tüm hamleler hamle diziisne kaydedilmektedir. Bu hamle dizisini eleman sayısı sıfır ise, şah çeken taraf oyunu kazanmış olmaktadır. Sonrasında ekrana oyunun kazanıldığını gösteren uyarı mesajı gelir. Oyunu beyaz kazanmış ise beyaz kazandı, siyah kazanmış ise siyah kazandı uyarı mesajı görüntülenmektedir.

Şekil 3.20 : Beyaz oyunu kazandığında gelen uyarı mesajı

Kazanıldı uyarı mesajından sonra oyun bitmiş olduğundan, yeni bir oyunun başlatılmasının istenilip istenilmediğini soran dialog penceresi ekrama gelmektedir. Evet butonuna tıklanıldığında yeni bir oyun başlatılmakta ve oyun başlangıç durumuna alınmaktadır. Evet butununa tıklanıldığında başlat isimli metod çağırılarak tahta dizisi oyun başlangıç değerlerine ayarlanarak başlangıç konumuna ulaşılmaktadır. Eğer hayır butonuna tıklanmış ise exit metodu kullanılarak java uygulaması sonlandırılmaktadır.

Şekil 3.21 : Yeniden oynamak istermisiniz dialog penceresi



Şekil 3.22 : Akış Diyagramı

Kullanıcının beyaz taşlardan birini, bilgisayar faresini kullanarak, oynamak istediği kareye sürüklemesi ile kullanıcı girişi yapılmış olur. Sürüklenen taşın bırakılması ile birlikte hamle girişi tamamlanmış olur. Beyazın ilgili poziyonda yapabileceği tüm olası hamleler üretilir. Hamlelerin üretilmesinde herbir taş çeşidi için gerekli metod çağırılarak yapılr. Olası beyaz hamleler oluşturulurken herbir hamle minimaks algoritmasından geçirilir. Yapılan kullanıcı hamlesi bu olası hamlelerden biri değil ise geçersiz amle girişi yapılmış olur. Bu durumda yapılan kullanıcı hamlesi geri alnarak, tekrar kullanıcı girişi yapılması beklenir.

Kullanıcının yaptığıhamle olası beyaz hamlelerinden biri ise geçerli hamle girişi yapılmış olur. Siyahın yapabileceği tüm olası hamleler üretilir. Hamlelerin üretilmesinde herbir taş çeşidi için gerekli metod çağırılarak yapılır. Olası hamleler oluşturulurken herbir hamle minimaks algoritmasından geçirilir. Bu sayede oyun ağacındaki yapraklar puanlandırılmış olur. Minimaks algoritmasında yapılan değerlendirme sonucunda bulunan hamleyi bilgisayar oynar. Sonrasında kullanıcının bir sonraki hamleyi yapması beklenir.

4. SONUÇ VE ÖNERİLER

Tez uygulamasında yapay zeka algoritmalarından minimaks algoritması kullanılmıştır. Algoritmanın değerlendirmesinde; statik ve dinamik değerlendirme kullanılmıştır. Bu değerlendirmeler sonucunda programın rasgele hareketler yapmadığı, kendisine verilen değerlendirme kriterlerine göre değerlendirme yaparak en iyi hamleyi yaptığı görülmüştür. Makalelerde kullanılan değerlendirme kısımlarıyla benzerlik göstermektedir.

Çizelge 4.1 : Statik ve dinamik değerlendirme sonuçları

	Windows 7 Satranç (Chess Titans) Düzey1		Satranç (Chessmaster 10) Lisa 576	
	Statik Değerlendirme	Statik ve Dinamik Değerlendirme	Statik Değerlendirme	Statik ve Dinamik Değerlendirme
Siyah Kazanç Sayısı	3	5	1	2
Beraberlik Sayısı	3	3	5	6
Beyaz Kazanç Sayısı	4	2	4	2

Yapılan uygulama iki farklı satranç programı ile oynatılmış (windows 7 chess titans ve chessmaster 10), statik ve dinamik değerlendirmenin kazanç üzerindeki etkisi test edilmiştir. Statik değerlendirmede, oyunun dinamik değerlendirme kısmı iptal edilmiştir. Minimaks algoritması sadece statik değerlendirme yapmaktadır. Bu durumdaki test sonuçları statik değerlendirme kolonunda gösterilmiştir. Dinamik değerlendirmenin etkisi, statik ve dinamik değerlendirme birlikte kullanılarak test edilmiştir.

Yapılan program siyah ile oynamaktadır. Rakip ise beyaz ile oynamaktadır. Siyah kazanç sayısı yaptığımız programın kazanma sayısını, beyaz kazanç sayısı ise rakibin kazanma sayısını göstermektedir. Beraberlik sayısı ise berabere kalınan oyun sayısını göstermektedir.

Yapılan test sonuçlarına göre statik ve dinamik değerlendirme birlikte kullanıldığında, programın performans artışı olduğu görülmektedir.

Dinamik değerlendirmenin daha doğru hamlelerin seçilmesinde ve kazanç üzerinde olumlu etkisinin olduğu görülmüştür.

Çizelge 4.2 : Katman sayısının kazanç üzerindeki etkisi

	Windows 7 Satranç (Chess Titans) Düzey1			Satranç (Chessmaster 10) Lisa 576		
	Katman Sayısı 3	Katman Sayısı 5	Katman Sayısı 10	Katman Sayısı 3	Katman Sayısı 5	Katman Sayısı 10
Siyah Kazanç Sayısı	3	5	7	3	4	6
Beraberlik Sayısı	5	2	1	4	3	2
Beyaz Kazanç Sayısı	2	3	2	3	3	2

Yapılan uygulama iki farklı satranç programı ile oynatılmış (windows 7 chess titans ve chessmaster 10) , minimaks algoritmasında kullanılan katman sayısı değişiminin kazanç üzerindeki etkisi test edilmiştir. Katman sayısı 3, 5 ve 10 olarak ayarlanarak test edilmiştir. Yapılan program siyah ile oynamaktadır. Rakip ise beyaz ile oynamaktadır. Katman sayısının değişimine bağlı olarak siyahın kazanma sayısı, beyazın kazanma sayısı ve beraberlik sayıları görülmektedir.

Katman sayısının artmasına bağlı olarak programın performansının arttığı görülmüştür.

Çizelge 4.3 : Oyun seviyelerinin kazanç üzerindeki etkisi

	Windows 7 Satranç (Chess Titans)			Satranç (Chessmaster 10)	
	Düzey1	Düzey2	Düzey4	Andre 317	Lisa 576
Siyah Kazanç Sayısı	5	4	2	4	2
Beraberlik Sayısı	3	3	4	4	6
Beyaz Kazanç Sayısı	2	3	4	2	2

Yapılan uygulama iki farklı satranç programı ile oynatılmış (windows 7 chess titans ve chessmaster 10) , oyun seviyesi artışının kazanç üzerindeki etkisi test edilmiştir. Windows 7 (chess titans) düzey1, düzey 2 ve düzey 4 seviyelerine karşı oynanmıştır.

Satranç (chessmaster 10) programının Andre 317 ve Lisa 576 seçenekleri ile oynanmıştır.

Yapılan program siyah ile oynamaktadır. Rakip ise beyaz ile oynamaktadır. Oyun seviyesinin artışına bağlı olarak programın performasının düştüğü görülmektedir.

Yapılan makale araştırmasında bazı makalelerin, satranç oyununun çeşitli kısımlarıyla ilgili olduğu görülmüştür. Bazıları sadece kullanıcı ara birimine

sahiptir[12]. Bazıları sadece yapay zeka kısmına sahiptir[5]. Yapılan bu tez uygulamasında program kullanıcı ara birimi ve yapay zeka gibi tüm satranç özelliklerine sahiptir.

Bazı makalelerde farklı yapay zeka teknikleri kullanılmıştır. Oyun sonu için yapay sinir ağları kullanılmıştır[16]. Yapay sinir ağları sadece oyunsonunda kullanılmıştır. Bazı makalelerde ise elektronik çözüm kullanılmıştır[17]. Böyle bir çözüm donanım gerektirmektedir.

KAYNAKLAR

[1] **Marc Boule and Zeljko Zilic**., 2002. An FPGA based move generator for game of chess, McGill university, Canada

[2] **Hallam Nasreddine, and Graham Kendall** 2006: Using an evolutionary algorithm for the tuning of a chess evaluation function based on a dynamic boundry strategy, The university of Nottingham Malaysia

[3] **S.H. Rubin and G. Lee** 2008 Learning conceptual chess for testing evolutionary programming versus a reasining-based soft expert system: the Kaser, San Diego

[4] **Nasrul Humaimi Mahmood1 and Rubita Sudirman**, 2011 Low cost electronic chess set for chess tournament, University Teknologi, Malaysia

[5] **Tim Mc Grew**, 1997. Collaborative intelligence, Western Michigan University

[6] **Wen- Yuan. Fogel, chiu-Yu Yen**, The Chinese –chess image identification techniques on spatial domain, National Chin-Yi university of echnology.

[7] **Ligun Zhang, Lili ding**, 2011. A stroge structure and capture juggng algorithm of reliazing the computer game program of surakarta chess.

[8] **Emui Satic, Melita Ahie-Djokic**, 2008 Simple computer vision system for chess playing robot manipulator as a project- based learning example, University of sarajevo.

[9] **Hans Berliner**, 1989. Hitech Chess: from master to seniormaster with no hardware change, Carneige Mellon University, Pittsburgh.

[10] **Vladon Wekovic**, 2003. The Realization of the advanced Hash system in geniss axon xp chess application.

[11] **Chrilly Donninger, Alex Kure**, 2004. The First distributed FPGA accelerated chess program, University of Paderborn, Paderborn.

[12] **David B. Fagel, Timothy**, 2006. The Blondie 25 chess program computes against Fritz 8.0 and a human chess master.

[13] **Guofeng Tang, Ying Qu**, 2009. Human-computer interactive gaming system. A Chinese chess robot , Norteastern University.

[14] **Eduardo V´azquez-Fern´andez and Carlos A..,** 2011: An evalutionary algorithm for tuning a chess evaluation function, *Mexiko*

[15] **Tang Lai Yu**, 2009. Chess gaming and graphics using open source tools, California State University, San Bernardino.

[16] **Jie Si, Rilun Tang**, 1999. Treined Neural Networks play chess endgames, University of Nevada, Reno.

[17] **Feng- Hsiung Hsu**, 1987. A two million moves cmos single chip chess move generator, Carneige Mellon University, Pittsburgh.

[18] **Wei-Po Lee. and Li-Jen Liu**, 2006. A component-based framework to rapidly prototype online chess games for home entertainment, October 8-11 Taiwan.

[19] **Henry S. Baird, Ken Thompson**, 1990. Reading chess , Princeton University, Princeton.

[20] **David L. Herrick, Paul K. Lee**, 1996. Chess a new reliable heigh speed hf radio, October 8-11 Taiwan.

[21] **Vasıf V. Nabiyev**, 2011. Satranç metinlerinin incelenmesi çeşitli notasyonlar arasında uyumun sağlanması, Karadeniz Teknik Üniversitesi.

[22] **M. Piskoree, N. Antilov Fantulin**, 2011. Computer vision system for the chess game reconstruction, University of Zagreb, Zagreb.

[23] **David R. herrick, Paul K. Lee**, 1996. New Reliable High Speed HF Radio Nashua.

[24] **David B. Fogel, Timothy J. Hays**, 2006. the blondie25 chess program computes against fritz 8.0 and a humanchess master, USA.

[25] **Zui Retchluman Königsberg**, 2008 A combinational game mathematical checkmate control procedure for a class of chess endgames.

[26] **Eduardo Varquez Fernandex, Carlos A. Coello**, 20012, An evaluationary algorithm coupled with the hooke-jeeves algorithm for tuning a chess evaluation function.

[27] **Tong Lai- Yu,** 2009, Chess cocuning and graphics using open source tools, Calfornia state university.

[28] **Yuyuyan Du**, 2006. Research of china chess game hardware system, Northeastern university.

[29] **M. Piskorec, N. Antulow- Fatulin**, 2011. Computer vision system for the chess game reconstruction, University of zagreb .

[30] **Zedin Chen, Miao Liu**, 2008, Regtechnologies analisis of three sides Chinese chess computer game, Guanzghou university.

Url-1 < http://www.ba.metu.edu.tr/~adil/BA-web/oyunteorisi.htm >26.04.2012

Url-2<http://www.masonlar.org/masonlar_forum/index.php?topic=4388.0> 06.04.2012

Url-3 < http://kodveus.blogspot.com/2006/07/minimax-ve-xox-oyunu-yabanclar-tic-tac.html>11.03.2012

Url-4 <http://www.canaktan.org/ekonomi/oyn-teorisi/anasayfa-oynn.htm> 24.03.2012

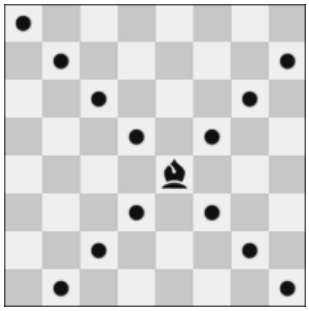
Url-5 < http://tr.wikipedia.org/wiki/Oyun_kuram%C4%B1>12.03.2012

Url6<http://www.tsf.org.tr/images/stories/Hakemler/kurallar/kurallar_turkce_19.05.09.pdf>26.04.2012

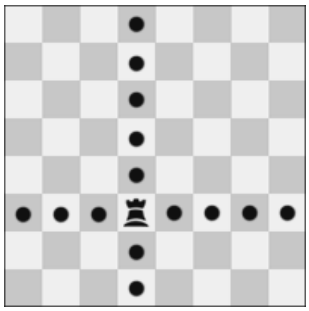
Url-7 < http://enm.blogcu.com/oyun-teorisi-game-theory/9578064>16.04.2012

EKLER

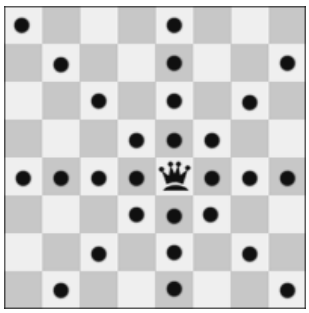
EK A.1



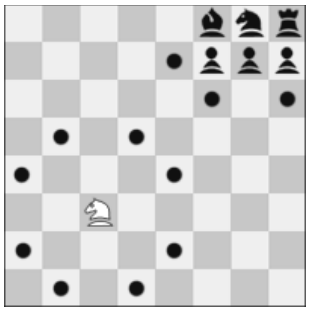
Şekil A.1: Fil'in çaprazlardaki hareketi



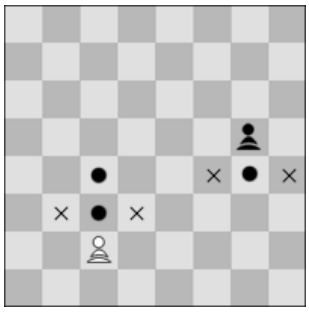
Şekil A.2: Kale'nin dikey ve yatay karelerdeki hareketi



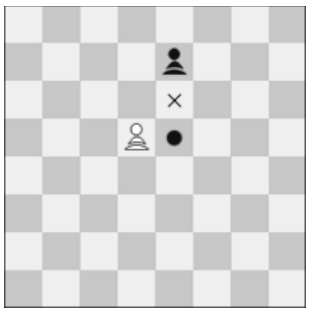
Şekil A.3: Vezir'in hareketi



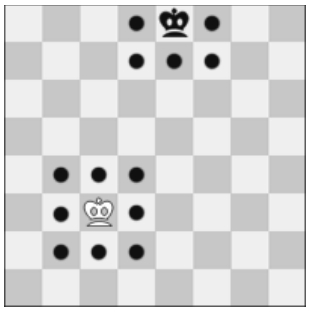
Şekil A.4: At'ın hareketi



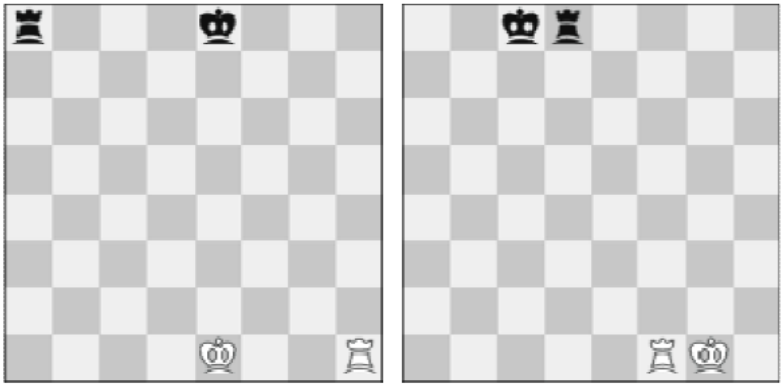
Şekil A.5: Piyon'un sağ ve sol çapraz karelere hareketi



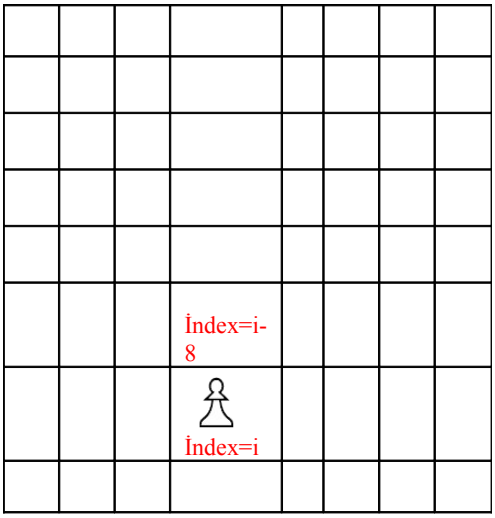
Şekil A.6: Piyon'un geçerken alma hareketi



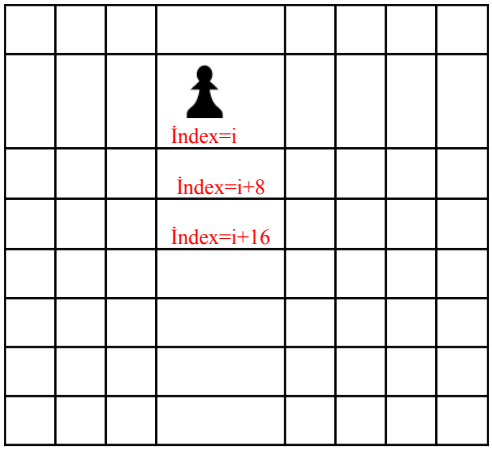
Şekil A.7: Şah'ın hareketi



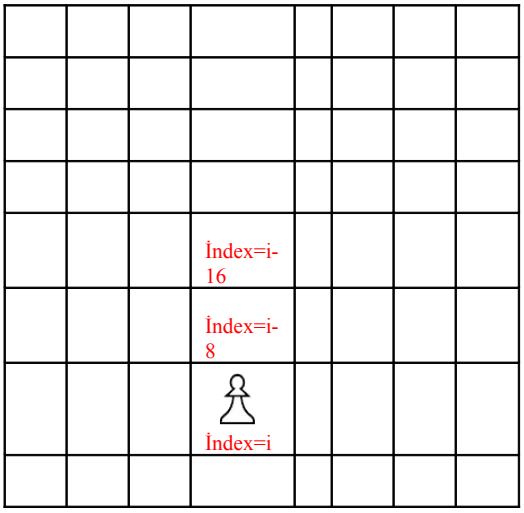
Şekil A.8: Beyaz ve siyah şah'ın rok hareketleri



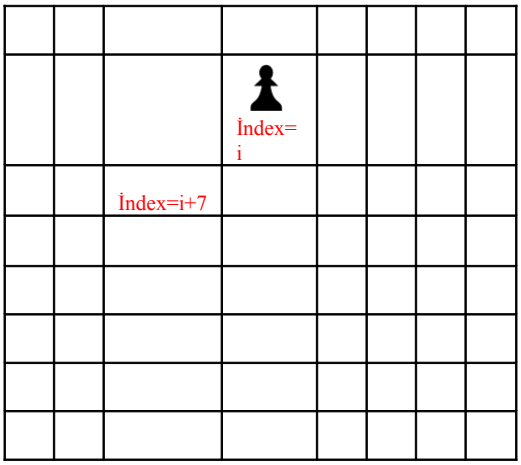
Şekil C.1: Beyaz piyon için bulunduğu karenin ve önündeki karenin indeksleri



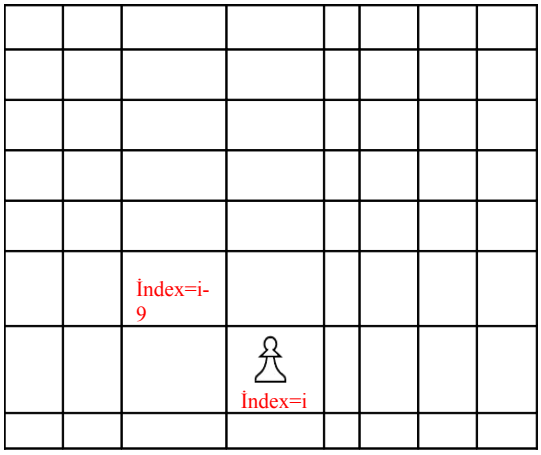
Şekil C.2: Siyah piyon için bulunduğu karenin ve önündeki karelerin indeksleri



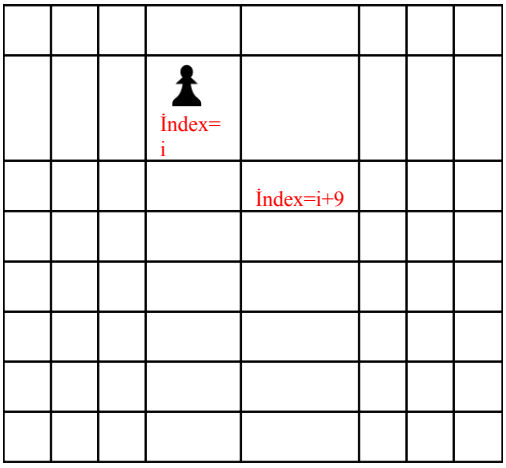
Şekil C.3: Siyah piyon için bulunduğu karenin ve önündeki karelerin indeksleri



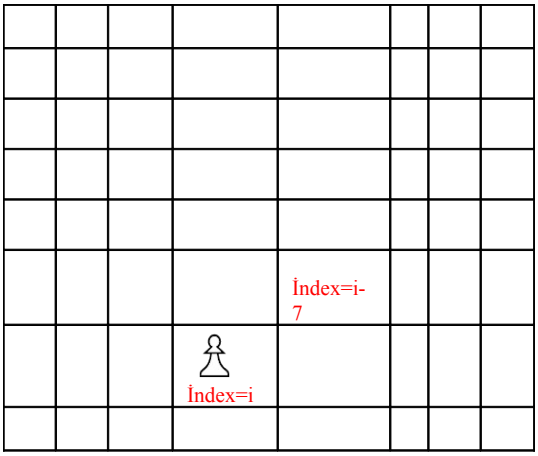
Şekil C.4: Siyah piyon'un karenin ve sol çaprazındaki karenin indeksleri



Şekil C.5: Beyaz piyon'un karenin ve sol çaprazındaki karenin indeksleri



Şekil C.6: Siyah piyon'un karenin ve sağ çaprazındaki karenin indeksleri



Şekil C.7: Beyaz piyon'un karenin ve sağ çaprazındaki karenin indeksleri

Çizelge A.1 : Ekler bölümünde çizelge örneği.

Kolon A	Kolon B	Kolon C	Kolon D
Satır A	Satır A	Satır A	Satır A
Satır B	Satır B	Satır B	Satır B
Satır C	Satır C	Satır C	Satır C

ÖZGEÇMİŞ

Ad Soyad:		**Fedai ŞEKERCİOĞLU**
Doğum Yeri ve Tarihi	**:**	**SİVRİHİSAR 1-3-1976**
Lisans Üniversite	**:**	**ULUSLARARASI KIBRIS ÜNİVERSİTESİ**