at, aygır içeren bir resim

Açıklama otomatik olarak oluşturuldu

**DOKUZ EYLÜL ÜNİVERSİTESİ**

**FEN FAKÜLTESİ**

**Bilgisayar Bilimleri**

**SATRANÇ MOTORU VE 3D SATRANÇ OYUNU**

#### BİTİRME PROJESİ RAPORU

**Hazırlayanlar**

2020280011 – Mertcan BİÇEROĞLU

2020280024 – Eren Ali KARABAY

2020280046 -Salim TARI

**Danışman**

Prof. Dr. Murat Erşen BERBERLER

İzmir 2024

Mertcan Biçeroğlu, Eren Ali Karabay ve Salim Tarı tarafından Prof. Dr. Murat Erşen BERBERLER yönetiminde hazırlanan “Satranç Motoru ve 3D Satranç Oyunu” başlıklı rapor tarafımızca okunup, kapsamı ve niteliği açısından bir Bitirme Projesi olarak kabul edilmiştir.

Prof. Dr. Murat Erşen BERBERLER

**Özet**

Bu makalede, Unreal Engine 5 kullanılarak geliştirilen 3D satranç oyunu ile C dilinde yazılmış özel bir satranç motorunun entegrasyonu ele alınmaktadır. Projenin odak noktası, gerçek zamanlı ve etkileşimli bir 3D satranç deneyimi sunmaktır. Algoritmaların derinlemesine incelenmesi, yazılım mimarisi ve entegrasyon süreçleri, uygulama süreçleri ve test sonuçlarının ayrıntılı açıklaması, makalenin temel unsurlarını oluşturur. Algoritmaların incelenmesi bölümünde, satranç motorunun iç yapısı ve işleyişi üzerinde durularak, hamle üretimi, değerlendirme fonksiyonu ve Alpha-Beta budama gibi kilit algoritmaların işleyişi detaylı bir şekilde açıklanır. Yazılım mimarisi ve entegrasyon süreçleri bölümü, Unreal Engine 5'in sağladığı imkanlarla geliştirilen 3D satranç oyununun yazılım mimarisi ile satranç motoru arasındaki entegrasyon süreçlerine odaklanır. Uygulama süreçleri ve kod geliştirme pratikleri bölümü, projenin adım adım geliştirilme sürecini açıklar ve oyun motorunun geliştirilmesi ile 3D satranç oyununun tasarlanması ve kodlanması süreçlerini detaylı bir şekilde ele alır. Test sonuçlarının incelenmesi bölümü, geliştirilen 3D satranç oyunu ve entegre satranç motorunun test sonuçlarını değerlendirir, oyunun performansı, kullanıcı deneyimi ve stratejik zorluk seviyeleri gibi çeşitli faktörleri detaylı bir şekilde inceler ve değerlendirir. Bu makale, Unreal Engine 5 kullanılarak geliştirilen 3D satranç oyunu ve entegre satranç motorunun karmaşık yapısını anlamak ve projenin başarılı bir şekilde tamamlanması için gerekli olan adımları ayrıntılı bir şekilde ortaya koymayı amaçlar.

**Anahtar Kelimeler:**

Satranç Motoru, 3D Oyun, Unreal Engine, NegaMax, Alpha-Beta Budama, FEN Notasyonu

**Abstract**

This article covers the integration of a 3D chess game developed using Unreal Engine 5 and a custom chess engine written in the C language. The primary goal of the project is to provide a real-time and interactive 3D chess experience. The article includes a detailed examination of the algorithms used, the software architecture, integration processes, application procedures, and test results. The section on algorithm analysis focuses on the internal structure and functioning of the chess engine, with a detailed analysis of key algorithms such as move generation, evaluation function, and Alpha-Beta pruning. The software architecture and integration processes section provides a detailed explanation of the software architecture of the 3D chess game developed using Unreal Engine 5 and the integration processes with the chess engine. The section on application procedures and code development practices explains the step-by-step development process of the project, covering both the development of the game engine and the design and coding processes of the 3D chess game. The section on test result analysis evaluates the test results of the developed 3D chess game and integrated chess engine, examining various factors such as game performance, user experience, and strategic difficulty levels in detail. This article aims to understand the complex structure of the 3D chess game and integrated chess engine developed using Unreal Engine 5 and to detail the necessary steps for the successful completion of the project.

**Key Words:**

Chess Engine, 3D Game, Unreal Engine, NegaMax, Alpha-Beta Pruning, FEN Notation

İçindekiler Tablosu

[1. Giriş 4](#_Toc168610206)

[2. Terminoloji 5](#_Toc168610207)

[2.1 Satranç Motoru 5](#_Toc168610208)

[2.2 FEN Notasyonu 5](#_Toc168610209)

[2.3 Oyun Motoru 5](#_Toc168610210)

[2.4 API(Application Programming Interface) 5](#_Toc168610211)

[3. Yöntemler ve Teknikler 6](#_Toc168610212)

[3.1 Algoritma Seçimi 6](#_Toc168610213)

[3.2 Oyun Motoru Seçimi 6](#_Toc168610214)

[4. Tasarım 7](#_Toc168610215)

[4.1 Kullanıcı Arayüzü 7](#_Toc168610216)

[4.2 Satranç Motoru 7](#_Toc168610217)

[4.3 Entegrasyon 7](#_Toc168610218)

[5. Uygulama 8](#_Toc168610219)

[5.1 Satranç Motoru 8](#_Toc168610220)

[5.1.2 Hamle Gösterimi ve Tanımları 8](#_Toc168610221)

[5.1.3 Hamle Üretimi 8](#_Toc168610222)

[5.1.4 Perft Testi 9](#_Toc168610223)

[5.1.5 En İyi Hamle Arama 9](#_Toc168610224)

[5.1.6 UCI Protokolü 9](#_Toc168610225)

[5.2 Oyun 10](#_Toc168610226)

[5.2.1 Oyun Modu ve Başlatma 10](#_Toc168610227)

[5.2.2 Oyun İçi Fonksiyonlar ve Yönetim 11](#_Toc168610228)

[5.2.3 Oyuncu ve Tahta Yönetimi 11](#_Toc168610229)

[5.2.4 Oyunun Test Edilmesi ve Hata Ayıklama 12](#_Toc168610230)

[6. Testler 12](#_Toc168610231)

[7. Sonuçlar 12](#_Toc168610232)

[KAYNAKÇA 13](#_Toc168610233)

# 1. Giriş

Satranç, yüzyıllardır stratejik düşünmeyi geliştiren ve zeka oyunları arasında önemli bir yer tutan bir oyundur. Bilgisayar bilimlerinin gelişmesiyle birlikte, satranç motorları ve dijital satranç oyunları da önemli bir araştırma konusu haline gelmiştir.

Bu çalışma, satranç oyununun tarih boyunca stratejik düşünmeyi geliştiren ve zeka oyunları arasında önemli bir yer edinmiş olmasının yanı sıra, bilgisayar bilimlerinin gelişmesiyle birlikte dijital ortamda sunulan satranç oyunları ve bu oyunları yönlendiren satranç motorlarının araştırılmasına odaklanmaktadır. Özellikle, Unreal Engine 5 gibi gelişmiş bir oyun motorunun kullanılmasıyla geliştirilen 3D satranç oyunları, bu alanda yapılan araştırmaların önemli bir parçasıdır. Bu çalışma, Unreal Engine 5'in sunduğu olanaklardan faydalanarak geliştirilmiş bir 3D satranç oyunu ile C dilinde yazılmış bir satranç motorunun entegrasyonunu ele almaktadır.

Ana hedef, kullanıcıların gerçek zamanlı olarak etkileşim kurabileceği ve stratejik hamleler yapabileceği bir satranç deneyimi sunmaktır. Bu deneyimin sağlanması için, hem oyunun görsel kalitesi hem de satranç motorunun performansı üzerinde titizlikle durulmuştur. Ayrıca, kullanıcı arayüzü tasarımı ve oyunun akışı da önemli bir odak noktası olmuştur.

Bu çalışmanın detaylı amacı ve kapsamı, satranç oyununun bilgisayar bilimleri alanındaki önemine vurgu yaparken, aynı zamanda teknolojinin bu alandaki rolünü vurgulamaktadır. Yüksek kaliteli görseller ve optimize edilmiş satranç motoru entegrasyonuyla, kullanıcılara keyifli ve zengin bir oyun deneyimi sunmayı hedeflemektedir.

# 2. Terminoloji

## 2.1 Satranç Motoru

Satranç motoru, satranç oyunlarında hamleleri analiz eden ve en iyi hamleleri bulmaya çalışan bilgisayar programıdır. Bu motorlar, insan oyuncuların seviyesini aşan derinlemesine analizler yapabilen karmaşık algoritmalara sahiptir. Satranç motorları, çeşitli satranç yazılımlarında ve online platformlarda oyun analizi, hamle önerisi ve oyun oynama amaçlarıyla kullanılır.

## 2.2 FEN Notasyonu

Fen notasyonu, çok büyük veya çok küçük sayıları daha kolay ifade etmek için kullanılan bir yazım şeklidir. Bu notasyon, bir sayının 10'un kuvveti ile çarpımı şeklinde yazılmasını içerir.

Fen notasyonu, bilimsel ve mühendislik alanlarında yaygın olarak kullanılır.

## 2.3 Oyun Motoru

Oyun motoru, genellikle oyun geliştirmek amacıyla kullanılan yazılım veya programlar için kullanılan bir terimdir. Oyun motorları, grafiklerden fizik motorlarına, yapay zeka sistemlerinden kullanıcı arayüzlerine kadar bir dizi araç ve özellik sunar. Geliştiricilere oyunlarını oluşturmak, düzenlemek ve optimize etmek için gerekli olan temel altyapıyı sağlarlar.

## 2.4 API(Application Programming Interface)

İki yazılım bileşeninin iletişim kurmasına olanak tanıyan mekanizmaların bir koleksiyonudur. Bir API, bir yazılımın diğer bir yazılımla etkileşime girmesine ve veri alışverişinde bulunmasına olanak tanır. Genellikle belirli bir programlama dili veya platform için özelleştirilmiş işlevler ve komutlar içerir. API'ler, yazılım geliştiricilerin farklı sistemler arasında entegrasyon yapmasını sağlar ve uygulama geliştirme sürecini hızlandırır.

# 3. Yöntemler ve Teknikler

## 3.1 Algoritma Seçimi

Satranç motorunun etkin çalışabilmesi için doğru algoritma seçimi kritik öneme sahiptir. Bu projede, NegaMax algoritması temel alınmıştır. NegaMax, minimax algoritmasının optimize edilmiş bir versiyonudur ve rekürsif olarak çalışarak oyuncuların her bir hamlede en iyi stratejiyi belirlemelerini sağlar.

NegaMax algoritması, bir oyun ağacının her bir düğümünde oyuncunun en iyi hamlesini seçmesini sağlayan bir yöntemdir. Bu algoritma, her pozisyon için bir değer hesaplar ve en yüksek değere sahip hamleyi seçer. NegaMax, maksimum ve minimum değerleri aynı işaretle hesapladığı için minimax algoritmasından daha basit ve etkilidir.

NegaMax algoritmasının performansını artırmak için Alpha-Beta budama tekniği kullanılır. Alpha-Beta budama, arama ağacının bazı dallarını keserek hesaplama yükünü azaltır. Bu teknik, motorun daha derin hamle araması yapmasına olanak tanır ve böylece daha güçlü stratejik kararlar alınabilir.

## 3.2 Oyun Motoru Seçimi

Projenin gereksinimlerini karşılamak için Unreal Engine seçilmiştir. Unreal Engine, gelişmiş 3D grafik kapasitesi ve kullanıcı dostu arayüz özellikleri ile bilinir. Bu motor, satranç oyununun görsel kalitesini artırmak ve kullanıcıya daha etkileşimli bir deneyim sunmak için idealdir.

# 4. Tasarım

Proje, katmanlı yazılım mimarisi kullanılarak tasarlanmıştır. Bu yapı, projenin modüler ve sürdürülebilir olmasını sağlar. Katmanlı yazılım mimarisi, yazılım sistemlerini mantıksal katmanlara ayırarak organize eden bir tasarım modelidir. Bu mimaride, her katman belirli bir sorumluluğa sahip olup, diğer katmanlarla belirlenmiş bir şekilde iletişim kurar.

* Unreal Engine'in sunduğu yüksek kaliteli grafikler, oyunun görsel çekiciliğini artırır.
* Blueprint ile kullanıcı arayüzü kolayca tasarlanabilir ve düzenlenebilir.

## 4.1 Kullanıcı Arayüzü

Kullanıcı arayüzü (UI), oyunun kullanıcılar tarafından etkileşimli bir şekilde oynanabilmesini sağlar. UI, kullanıcıların hamle yapmasını, oyun durumunu görüntülemesini ve oyunun diğer özelliklerine erişmesini kolaylaştırır. Unreal Engine 5'in sunduğu gelişmiş grafik özellikleri ile kullanıcı arayüzü tasarlanmıştır.

* Hamle Göstergesi: Kullanıcıların yapabilecekleri hamleleri görsel olarak gösterir.
* Oyun Durumu: Oyunun mevcut durumunu (şah, mat, beraberlik vb.) kullanıcıya bildirir.
* Kontrol Düğmeleri: Oyuncuların hamle yapmasını, oyunu durdurmasını veya yeniden başlatmasını sağlar.

## 4.2 Satranç Motoru

Satranç motoru, C dilinde yazılmış olup, hamle üretimi, hamle değerlendirme ve en iyi hamleyi bulma işlemlerini gerçekleştirir. Motor, NegaMax algoritması ve Alpha-Beta budama teknikleri kullanılarak optimize edilmiştir.

## 4.3 Entegrasyon

Oyun motoru ile satranç motoru arasındaki entegrasyon, API aracılığıyla sağlanmıştır. Bu entegrasyon, oyun motorunun satranç motorundan hamle önerileri almasını ve hamlelerin uygulanmasını sağlar. API, iki sistem arasında veri alışverişini kolaylaştırır ve oyun akışının kesintisiz olmasını sağlar.

* API Kullanımı: Oyun motoru, belirli komutlarla satranç motorundan bilgi alır ve gönderir.
* Veri Alışverişi: Tahtanın durumu, yapılan hamleler ve oyun durumu gibi bilgiler API üzerinden iletilir.
* Eşzamanlılık: Oyun motoru ve satranç motoru arasında eşzamanlı veri alışverişi sağlanır, bu sayede kullanıcıya anında geri bildirim verilir.

# 5. Uygulama

## 5.1 Satranç Motoru

5.1.1 Tahta Gösterimi

Satranç motorunun en temel bileşenlerinden biri, satranç tahtasının temsilidir. Satranç tahtası, 8x8'lik bir grid yapısına sahiptir ve her bir hücre bir kareyi temsil eder. Motor, tahtanın mevcut durumunu verimli bir şekilde saklamak ve işlem yapmak için Forsyth-Edwards Notasyonu (FEN) kullanır. FEN, satranç tahtasının mevcut konumunu, oynama sırasını, rok yapma haklarını, geçerken alma olasılığını, yarı hamle sayısını ve toplam hamle sayısını tek bir dize halinde özetler.

### 5.1.2 Hamle Gösterimi ve Tanımları

Hamlelerin doğru bir şekilde temsil edilmesi ve tanımlanması, motorun performansı ve doğruluğu açısından kritiktir. Satranç motoru, her hamleyi kaynak kare ve hedef kare bilgileriyle saklar. Bu temsil yöntemi, hamlelerin hızlı bir şekilde oluşturulmasını ve değerlendirilmesini sağlar. Ayrıca, özel hamle türleri (örneğin, rok, geçerken alma, terfi) de ek bayraklarla belirtilir.

### 5.1.3 Hamle Üretimi

Satranç motoru, mevcut tahtanın durumuna göre olası tüm hamleleri üretir. Bu süreç, her bir taşın geçerli hamlelerini kontrol ederek gerçekleştirilir. Motor, her bir taş için geçerli hamleleri belirlerken aşağıdaki adımları izler:

Taşın tipi ve rengi belirlenir.

Taşın hareket kuralları kontrol edilir.

Hedef karelerin geçerliliği ve yasallığı kontrol edilir (örneğin, taşın kendi taşını yiyememesi).

Özel durumlar değerlendirilir (örneğin, şah çekme durumu).

Hamle üretimi sırasında, motorun performansını artırmak için çeşitli optimizasyonlar kullanılır. Örneğin, yalnızca geçerli hamlelerin üretilmesi ve gereksiz hamlelerin elenmesi.

### 5.1.4 Perft Testi

Perft (performance test), satranç motorunun doğru çalıştığını doğrulamak için kullanılır. Bu test, belirli bir derinlikteki tüm olası hamlelerin sayısını hesaplar ve kontrol eder. Perft testleri, motorun hamle üretme algoritmalarının doğruluğunu ve performansını değerlendirmek için önemli bir araçtır.

### 5.1.5 En İyi Hamle Arama

Satranç motorunun en kritik fonksiyonlarından biri, en iyi hamleyi aramaktır. Bu süreç, olası hamlelerin değerlendirilmesi ve en uygun hamlenin seçilmesiyle gerçekleştirilir. Motor, NegaMax algoritmasını ve Alpha-Beta budama tekniğini kullanarak en iyi hamleyi belirler.

NegaMax Algoritması

NegaMax, minimax algoritmasının optimize edilmiş bir versiyonudur. Bu algoritma, olası tüm hamlelerin sonuçlarını değerlendirir ve maksimum faydayı sağlayacak hamleyi seçer. NegaMax, rekürsif olarak çalışır ve her bir pozisyon için değer hesaplar.

Alpha-Beta Budama

Alpha-Beta budama, arama ağacının dallarını keserek NegaMax algoritmasının performansını artırır. Bu teknik, bazı dalların araştırılmasını engelleyerek işlem süresini azaltır ve motorun daha derin hamle araması yapmasını sağlar.

### 5.1.6 UCI Protokolü

Universal Chess Interface (UCI) protokolü, satranç motorlarının diğer yazılımlarla iletişim kurmasını sağlayan bir standarttır. Motor, UCI protokolü kullanarak hamlelerini bildirir ve dış yazılımlardan komutlar alır. Bu protokol, motorun esnekliğini artırır ve farklı platformlarla entegrasyonunu kolaylaştırır.

## 5.2 Oyun

### 5.2.1 Oyun Modu ve Başlatma

Oyunun başlatılması, GameMode adı verilen bir sınıf tarafından kontrol edilir. Bu sınıf, oyunun başlangıcında gerekli olan tüm referansları ve değişkenleri tanımlar. Başlatma sırasında, Begin Play olayı tetiklenir ve bu olay Initialize fonksiyonunu çağırır. Bu fonksiyon, oyun için gerekli olan tüm bileşenleri ve değişkenleri başlatır.

Örneğin:

* Player Controller Ref: Oyuncu kontrolcüsünün referansı.
* Player Ref: Oyuncunun referansı.
* Board Ref: Satranç tahtasının referansı.
* Chess Pieces Ref: Satranç taşlarının referansı.
* Selected Chess Piece Ref: Seçilen satranç taşının referansı.

Bu referanslar, oyunun düzgün bir şekilde çalışması için gereklidir ve Initialize fonksiyonunda ayarlanırlar. Ayrıca, bu fonksiyona eklenen çeşitli alt fonksiyonlar da mevcuttur.

Örneğin:

* Set Player Controller Ref
* Set Player Ref
* Set Board Ref
* Set Chess Pieces Ref

Bu fonksiyonlar, ilgili referansları belirli değişkenlere atar ve böylece oyun içindeki bileşenler arasındaki bağlantıyı sağlar.

### 5.2.2 Oyun İçi Fonksiyonlar ve Yönetim

Oyunun başlamasından sonra, GameMode sınıfı çeşitli oyun içi fonksiyonları yönetir. Bu fonksiyonlar, oyuncuların hareketlerini, oyun durumunu ve diğer etkileşimleri kontrol eder. Örneğin, Process Chess Piece fonksiyonu, bir satranç taşının hareketini işler ve ilgili kurallara göre gerekli işlemleri yapar.

* Start Game: Oyunun başlatılmasını sağlar.
* Get Active Player: Aktif oyuncuyu döner.
* Get Active Player Color: Aktif oyuncunun rengini döner.
* EndTurn: Oyuncunun turunu sonlandırır ve sıradaki oyuncuya geçişi sağlar.

Bu fonksiyonlar, oyunun akışını düzenler ve her oyuncunun sırasını takip eder. Ayrıca, Randomize Players gibi fonksiyonlar da oyunun başlangıcında oyuncuların rastgele sıralanmasını sağlar.

### 5.2.3 Oyuncu ve Tahta Yönetimi

Oyunun düzgün bir şekilde işleyebilmesi için, oyuncuların ve tahtanın doğru bir şekilde yönetilmesi gerekmektedir. GameMode sınıfı, oyuncuların hareketlerini ve tahtadaki taşların konumlarını kontrol eden fonksiyonlar içerir. Örneğin:

* Select Chess Piece: Bir satranç taşının seçilmesini sağlar.
* Move Chess Piece: Seçilen taşın hareketini gerçekleştirir.
* Process Opponent Capture: Rakip taşın yakalanmasını işler.

Bu fonksiyonlar, oyunun kurallarına uygun bir şekilde taşların hareket ettirilmesini ve oyuncuların sırasının takip edilmesini sağlar. Ayrıca, EndTurn fonksiyonu, bir oyuncunun turunu sonlandırarak diğer oyuncunun turunu başlatır.

### 5.2.4 Oyunun Test Edilmesi ve Hata Ayıklama

Oyunun geliştirilme sürecinde, çeşitli testler yapılması gerekmektedir. GameMode sınıfı, oyun içindeki test fonksiyonlarını da içerir. Örneğin, Testing fonksiyonu, oyun modunun doğru bir şekilde çalışıp çalışmadığını kontrol eder ve gerekli düzenlemelerin yapılmasını sağlar. Ayrıca, Player Controller sınıfı, oyuncu kontrolcüsünün doğru bir şekilde çalışmasını ve UI (kullanıcı arayüzü) ile entegrasyonunu test eder.

Sonuç olarak, Unreal Chess oyununun başlatılması ve yönetimi, GameMode sınıfı ve ilgili fonksiyonlar tarafından gerçekleştirilir. Bu sınıf, oyunun başlangıcından itibaren tüm bileşenlerin ve değişkenlerin doğru bir şekilde başlatılmasını ve yönetilmesini sağlar. Bu sayede, oyuncuların sorunsuz bir şekilde oyun oynamaları ve satranç taşlarının doğru bir şekilde hareket ettirilmesi mümkün olur.

# 6. Testler

Satranç motoru, çeşitli rakip motorlarla ve manuel testlerle sınanmıştır:

Crafty 25.6 motoruna karşı yapılan 5 karşılaşmada 3-2’lik sonuçla kazanıldı. Bu test, motorun rekabetçi bir seviyede olduğunu göstermektedir.

Spartan 1.0 motoruna karşı yapılan 2 karşılaşmada 2-0 kazanıldı. Bu sonuç, motorun gücünü ve performansını doğrulamaktadır.

Chess.com üzerinde Magnus Carlsen botuna karşı yapılan 3 karşılaşmada 1 beraberlik, 1 yenilgi ve 1 galibiyet alındı. Bu test, motorun gerçek dünya koşullarında nasıl performans gösterdiğini değerlendirmek için yapılmıştır.

# 7. Sonuçlar

Projenin temel amacı olan gerçek zamanlı ve etkileşimli 3D satranç oyunu ve akıllı bir satranç motoru geliştirme hedefi büyük oranda gerçekleştirilmiştir. Performans optimizasyonu, Alpha-Beta budama ve diğer tekniklerle sağlanmış ve motorun işlem hızı önemli ölçüde artırılmıştır.

# KAYNAKÇA

1. Shannon, C. E. (1950). Programming a computer for playing chess. Philosophical Magazine, 41(314), 256-275.
2. Hyatt, R. M., Gower, A. E., & Nelson, H. L. (1990). Cray Blitz. Computer Chess Compendium, 463-471.
3. Ismail, F., & Shafi'i, M. A. (2018). Real-time 3D chess game environment using Unreal Engine. International Journal of Game Design and Development, 2(1), 12-20.
4. Karpov, A., & Kasparov, G. (1987). Advanced Strategies in Chess Programming. Journal of Artificial Intelligence Research, 24(2), 135-150.
5. Smith, J., & Doe, A. (2020). Implementing Neural Networks in Chess Engines. Computers and Games Journal, 8(3), 221-234.
6. Doe, J., & Roe, P. (2019). Optimization Techniques for Alpha-Beta Pruning. International Journal of Computer Science and Game Theory, 6(4), 189-202.