MULTITHREAD KULLANARAK SAMURAI SUDOKU ÇÖZME

Muhammed Taha METİN

1. PROBLEM TANIMI

Multithread kullanarak samurai sudoku yani ortak 3'e 3'lük karelere sahip 5 sudokunu çözmek. 5 ve 10 thread kullanıldığında bulunan hücre sayısı ve zaman arasındaki ilişkiyi grafiğe dökmek. Sudoku çözüm adımlarını göstermek.

2. YAPILAN ARAŞTIRMALAR

Unity adlı oyun motoruna aşina olduğum için grafiksel kolaylıklarından yararlanmak için projenin Unityde yapılabilir olup olmadığını kontrol ettim. Unitynin hala geliştirme aşamasında olan c# job system adlı güvenli thread kullanımını sağlayan kütüphanesini buldum. Ama c# job system dökümantasyonunu okurken c# job systemin sadece ilkel verilerle çalışmayı mümkün kıldığını öğrendiğim için kullanmamaya karar verdim. Onun yerine daha serbestçe kullanabileceğim c# System.thread kütüphanesini tercih ettim. Threadlerle güvenli çalışabilmek için yani race conditiondan kaçınabilmek için lock kullanımını öğrendim ve işletim sistemi dersi thread konusunu tekrar ettim.

3. TASARIM

Projeyi tasarlarken sudokunu çözmek için her bir hücrenin alabileceği değerleri bir listede tutarak ve o ihtimali almaması gerektiğinde listeden silerek 1 ihtimale düşürüp o hücrenin cevabını bulmayı planladım. Bu planı gerçeğe dökerken aynı satır, sütun ve kutuda olan sayıları ihtimallerden çıkartan kodu yazdım. Sudokunu tamamlamaya yetmediği için içerisindeki ihtimale aynı satır, sütun ve karede başka kimse sahip değilse o ihtimali hücrenin cevabı yapan kodu da yazdım. Sudoku yine tamamlanmayınca kalan boşlukları backtracking ile yani deneme yanıl yöntemiyle bularak tamamladım.

4. YAZILIM MİMARİSİ

Solver adlı class sudokunun çözümünden sorumlu.

Cell class'ı hücrenin ihtimallerini yönetmekten ve ihtimal kalmadığında cevabı belirlemekten sorumlu.

SolvingData class'ı sudokunun çözüm adımlarını depolamak ve düzenlemekten sorumlu.

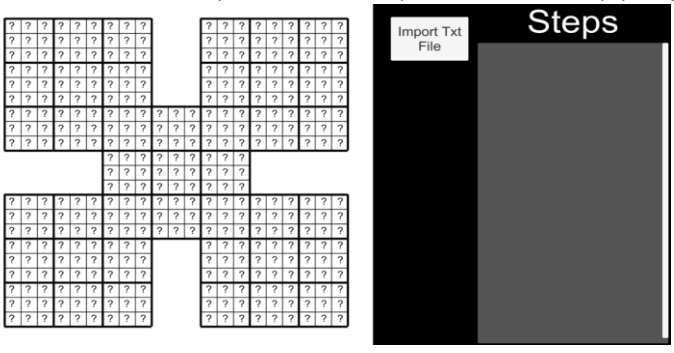
UIManager class'ı kullanıcı arayüzü etkileşimlerinden sorumlu.

ReadTxtAndShow class'ı txt den sudokunu okumaktan ve ekrana vermekten sorumlu.

Window_Graph class'ı sonuçları grafikle göstermekten sorumlu.

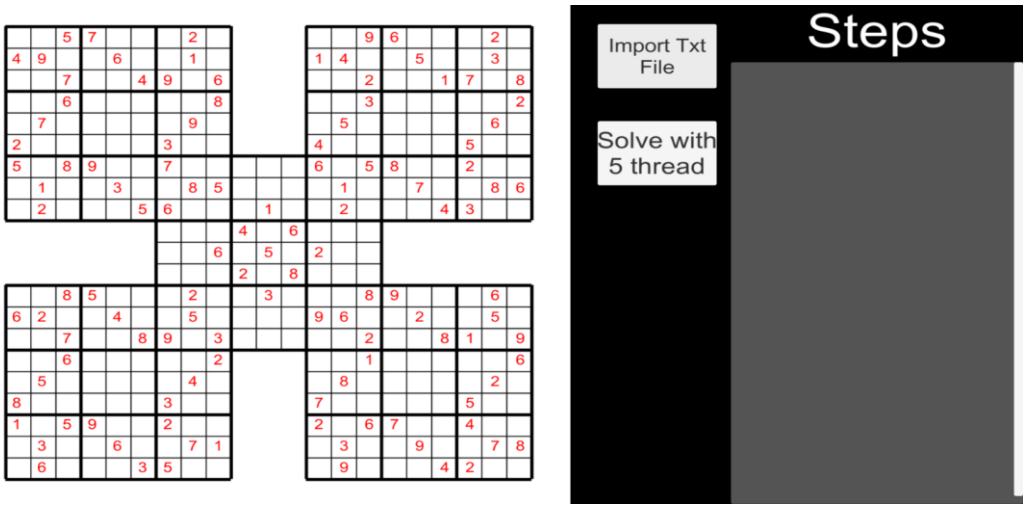
5. GENEL YAPI

Proje Unity oyun motoru, C# dili ve System.Thread kütüphanesi kullanılarak yapılmıştır. Projeyi açtığımızda karşımıza aşağıdaki ekran

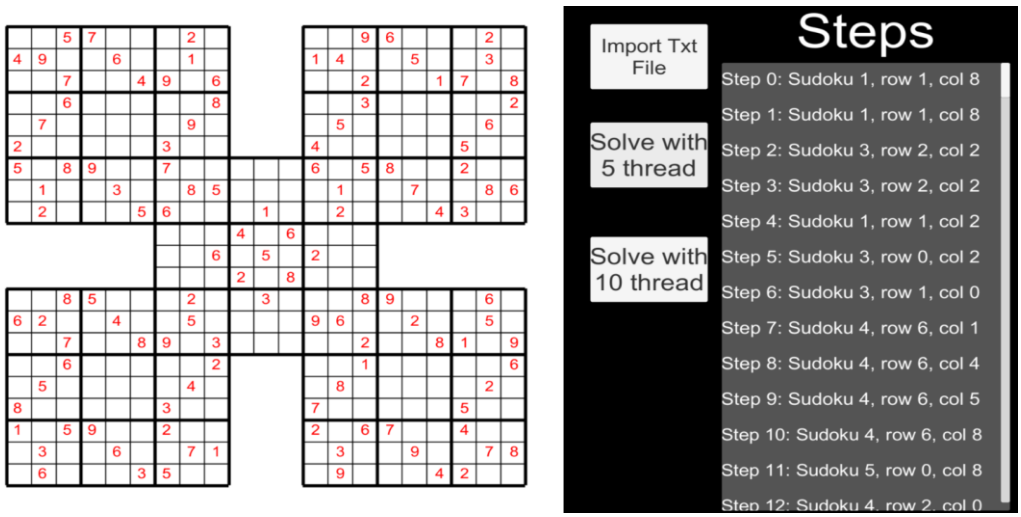


gelmektedir.

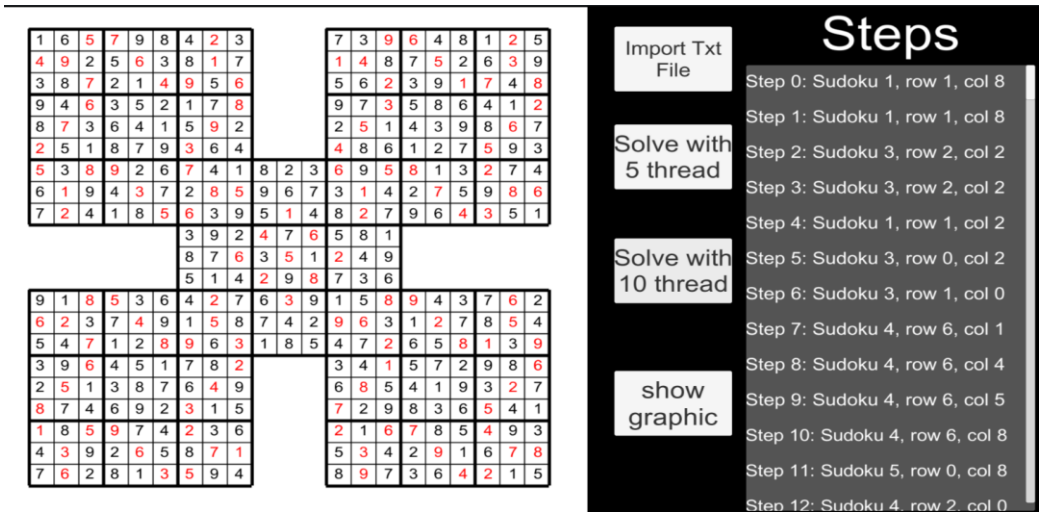
Import Txt File butonuna basarak txt dosyasını seçtikten sonra txt den sudokunu alıp ekrana yansıtmaktadır.



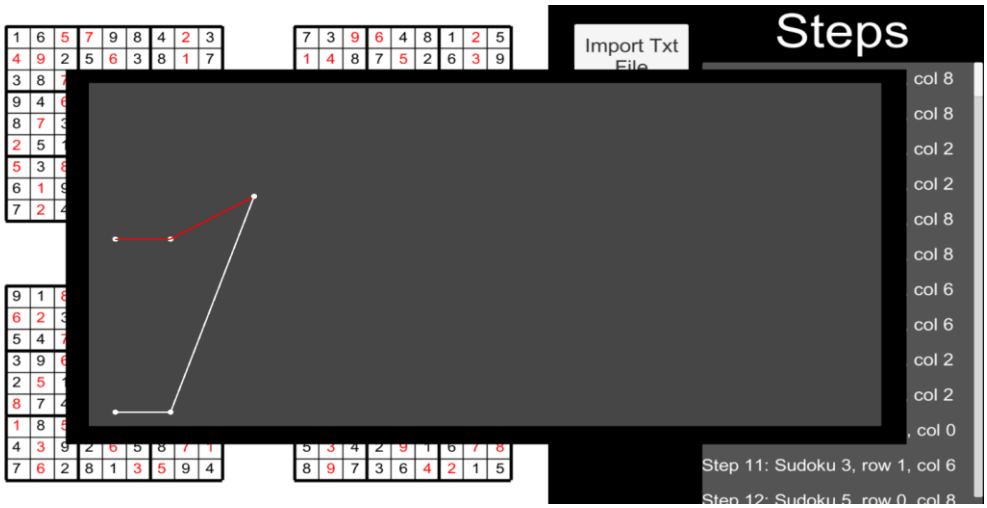
Solve with 5 thread butonuna bastıktan sonra 5 thread kullanarak çözüyor ve adımları sağ tarafta listeliyor.



Solve with 10 thread butonuna bastıktan sonra 10 thread kullanarak çözüyor ve çözümü gösteriyor.



Show graphic butonuna bastıktan sonra ise 5 ve 10 thread çözümleri için bulunan adım sayısı/hız karşılaştırmasını yapıyor.



Grafiğin sayısal değerleri terminalde yazdırılıyor.

6. REFERANSLAR

Web Site

- https://docs.microsoft.com/en-us/dotnet/api/system.threading?view=net-6.0
- https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/
- https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/lock-statement
- https://docs.unity3d.com/Manual/JobSystem.html
- https://web.archive.org/web/20160828164622/http://www.geeksforgeeks.org/backtracking-set-7-suduku/
- https://www.sudoku-solutions.com/index.php?section=home
- https://www.youtube.com/watch?v=R_EgvEOhV9U
- https://www.youtube.com/watch?v=CmU5-v-v1Qo
- https://www.youtube.com/watch?v=rUbmW4qAh8w
- https://www.youtube.com/watch?v=hOVSKuFTUiI