**Capitolul I**

**Introducere**

Încă din vechiurile timpuri jocurile au fost privite ca o modalitate de divertisment dar și ca o forma de dezvoltarea a abilităților si aptitudinilor atât fizice cât și de gândire. În timp ce evoluția omului a fost una ascendentă, așa au evoluat si jocurile devenind tot mai importante în viața oamenilor.

Inițial, sintagma „joc video” reprezenta transpunerea pe un ecran semnalele video produse de un emițător de tip mecanic cu scopul de divertisment sau pentru educație. Însă în prezent, acesta a căpătat un alt sens, fiind o întreaga industrie cu un număr enorm de platforme și aplicații capabile de a genera divertisment sau de a provoca gândirea. În figura 1 putem observa câștigurile din industria de jocuri video din China (una dintre cele mai mari industrii din lume) care este într-o continuă creștere.

Unul dintre pionierii jocurile video este „Spacewar” un joc ce a fost conceput în anii 1960 dar abia 10 ani mai târziu, în 1970, a fost posibilă comercializarea acestuia deoarece prețul componentelor hardware s-a micșorat iar jocul s-a bucurat de o popularitate neașteptată.

Șahul se crede că a avut originea în India în jurul secolului al VI-lea . Jocul s-a răspândit în vest prin comerț, dar șahul a obținut statutul social și permanența mai puternic decât multe alte jocuri. Șahul a devenit un joc de îndemânare și tactici, forțându-i adesea pe jucători să gândească două sau trei mișcări înaintea adversarului lor doar pentru a ține pasul. Acest joc a devenit, de asemenea, acceptat de mulți drept un punct de reper pentru inteligență; oamenii care au devenit maeștri au fost considerați deștepți. Jocul înfățișează pioni, cai, regi, regine, nebuni și ture. Câteva portretizează poziții efective în armata istorică europeană. Fiecare piesă are un model unic de mișcare. De exemplu, calul este restricționat să se deplaseze într-o formă L în lungime de două pătrate și un pătrat în lateral, tura se poate deplasa doar în linie dreaptă pe verticală sau pe orizontală, iar nebunii se pot deplasa în diagonală pe tablă.

„Computerele electronice pot fi setate să joace un joc destul de puternic de șah, apărând întrebarea dacă pot gândi”[[1]](#footnote-1). Încă de la sfârșitul secolului al 18-lea, începutul secolului al 19-lea, un inventator maghiar numit: Wolfgang von Kempelen, a uluit Europa cu un device denumit: „The Maelzel Chess Automaton” (Figura 1) . Acesta presupunea jucarea unei partide de șah automată dar s-a dovedit mai târziu că a existat un omuleț înăuntru care juca.

Figura 1:Mașinăria de jucat șah al secolului al 18-lea a fost de fapt condus de om înăuntru

O încercare mai onestă de a realiza o mașinărie automată de șah a fost în 1914 de către inventatorul spaniol Leonardo Torres y Quevedo. Acest a construit un dispozitiv care putea să ducă o partidă la final. Jocul era compus dintr-un rege si o tură contra la un rege. Mașinăria juca de partea regelui și a turei și forța șahul mat în cât mai puține mișcări, regele advers fiind controlat de către o persoană umană. Dispozitivul a fost prezentat la Paris în același an și este considerat primul joc pentru computer.[[2]](#footnote-2)

După ce au fost considerate doar o curiozitate, programele de șah pe calculator au crescut în abilitate până la punctul în care să poată concura în mod serios cu maeștri umani.

Kasparov clasat pe primul loc în lume, a jucat un meci de șase jocuri împotriva calculatorului IBM de șah Deep Blue, în 1996. Deep Blue a șocat lumea prin câștigarea primului joc, dar Kasparov a câștigat convingător meciul prin câștigarea a 3 jocuri și 2 remize. Meciul de revanșă în 1997 a fost câștigat de către mașină, care a fost ulterior retrasă de către IBM. În octombrie 2002, Vladimir Kramnik a jucat într-un meci de opt jocuri contra programului de calculator Deep Fritz. În 2003, Garry Kasparov a jucat atât un meci de șase jocuri cu programul de calculator Deep Junior, în februarie, cât și un meci de patru jocuri împotriva lui X3D Fritz, în luna noiembrie.

Decada dintre anii 2006-2017 este epoca de aur a motoarelor de șah pentru calculatoare. Acestea au devenit atât de bune încât nici un jucător de șah uman nu le putea face față chiar dacă unele partide se jucau cu un ușor handicap pentru motoarele de șah. Până la sfârșitul anului 2017, nu s-a întâmplat nimic revoluționar. Puterea lor s-a schimbat constant, dar încet. Totul părea familiar și nimeni nu se aștepta la schimbări radicale, la descoperiri majore. Pe 5 Decembrie 2017 este lansat un nou motor de șah de către cei de la DeepMind ( compania Google de inteligență artificială) care propunea o noua abordare. În loc de căutare alfa-beta și funcție de aproximare liniară pentru evaluarea poziției utilizate de motoarele tradiționale, AlphaZero folosește o funcție de aproximare neliniară bazată pe o rețea neuronală profundă și simularea Monte Carlo. În consecință, în loc să forțeze brute prin analiză, AlphaZero este capabil să „auto-învețe” șahul. Totodată Stockfishul rămâne un punct stabil în materia motoarelor de șah deoarece acesta este open source și se poate analiza modalitatea de gândire pentru a te perfecționa ca jucător de șah, înțelegerea anumitor principii solide ale jocului și un mediu prielnic pentru a modifica euristica acestuia în condițiile adaptării la nevoile personale.

Bucurându-se șahul de o popularitate crescută, au început să apară diferite variații ale acestuia cu scopul de a crea noi posibilități de joc, bazându-se în principiu pe aceeași tablă de joc și același efectiv de piese. Aceste reguli schimbate pot diferi de la modalitate de câștig până la poziția de start a pieselor. Dintre sutele de variații existente putem enumera cele mai populare:

* 3-Check: Presupune aceleași reguli ca jocul clasic doar cu condiția suplimentară că ești desemnat învins dacă pe perioada partidei acumulezi un număr mai mare de 3 șahuri.
* King of the Hill: este o variantă interesantă în care obiectivul este să-ți treci regele în centrul plăcii sau „vârful dealului”. Jocurile se pot încheia în moduri tradiționale de șah-mat, impas și time-out. Jocul se poate încheia și atunci când un jucător își mută regele în oricare dintre cele patru pătrate centrale: e4, d4, e5 și d5 sunt pătrate „de sus”, câștigând jocul la fața locului pentru alb sau negru, indiferent de natura poziției.
* Șah 960: Este o variantă inventată de regretatul campion mondial Bobby Fischer. Regulile jocului sunt aceleași ca șahul standard, dar în efortul de a reduce impactul teoriei deschiderii, piesele au fost amestecate la întâmplare pe randul din spate al fiecărui jucător. Jocurile se termină în modurile tradiționale de checkmate, impas și time-out. Nebunii trebuie să fie încă pe culori opuse în poziția inițială de Șah960. Regele trebuie să fie între ture pentru a menține capacitatea de a realiza rocada în ambele sensuri. Asta înseamnă că un rege poate fi plasat doar între pătratele b1 și g1 sau b8 și g8 pentru negru. Asta înseamnă:
  + Indiferent de locația regelui și a turei, se aplică reguli standard de rocadă. Trebuie să fie prima mișcare atât pentru rege, cât și pentru tură, regele s-ar putea să nu călătorească în sau prin șah și nu poate exista piese pe vreun pătrat prin care se traversează o piesă.
  + Regele și tura pun capăt procesului de rocadă în cazul în care ar urma într-o partidă de șah standard. Exemplu: Chiar dacă un rege alb este pe b1 și o tură este pe e1, rocada regelui ar putea implica glisarea regelui alb în g1 (pătratul său final de destinație), ceea ce va muta automat tura albă de la e1 la f1, întrucât Albul va avea întotdeauna regele pe g1 și tura pe f1 în poziția finală de rocadă.

Un motor(engine) de șah este pur și simplu un program software care joacă și analizează șahul. Cuvântul „motor” se referă pur și simplu la un fel de program cu putere mare, care face multă căutare și procesare - similar cu un motor de căutare.

Oamenii au fost întotdeauna fascinați de mașinării - inclusiv mașini de șah. În 1796, a fost creată o mașină falsă de joc de șah, numită „Automaton”, dar avea un jucător puternic de șah uman care se ascundea în jocul mișcărilor. În 1912, a fost creată o mașină care ar putea efectiv să dea șah cu o tura și un rege contra la un singur rege. Dar abia în 1951, Alan Turing a scris un program pe calculator care a putut juca șah. Pentru următorii 50 de ani, programatorii au lucrat la îmbunătățirea motoarelor de șah, iar îmbunătățirile hardware au permis un joc mai puternic. Până în 2005, motoarele de șah deveniseră cu siguranță mai puternice decât cei mai buni jucători umani. În anii de atunci, s-au îmbunătățit semnificativ, iar acum există sute de programe de calculator mai puternice decât cei mai talentați oameni la șah.

Oamenii s-au îmbunătățit la șah de-a lungul timpului, în timp ce învață din înțelepciunea colectivă a jucătorilor din trecut. Calculatoarele au accelerat acest progres, deoarece motoarele de șah au adăugat noi cunoștințe și înțelegere jocului. Cei mai buni jucători folosesc astăzi motoarele de șah pentru a analiza pozițiile și a genera idei. Din păcate, acest lucru a introdus și înșelăciunea în șah, unde orice jucător care folosește doar un telefon mobil și un motor de șah poate juca mai bine decât orice maestru.

Motoarele de șah sunt complexe. Cu toate acestea, în termeni simpli, fac două lucruri importante:

1. *Evaluarea*. Motoarele de șah privesc pozițiile individuale și evaluează care poziție este mai bună. Aproape toate motoarele de șah afișează un număr de evaluare, sau „eval”, pe baza aceluiași punctaj pe care îl folosesc majoritatea jucătorilor de șah (un pion în valoare de un punct, o piesă minoră trei etc.). Fiecare motor de șah face acest lucru în mod diferit, dar majoritatea motoarelor privesc lucrurile precum materialul din fiecare parte, toate amenințările de pe tabla de joc, siguranța regelui și structura pionului. Scorul cumulat al celei mai bune evaluări în viitor este însumat la un număr. Motoarele tradiționale evaluează în mod similar cu oamenii, deoarece au fost proiectate de oameni. Motoarele cu rețea neuronală se evaluează diferit. Poziția din Figura 3 este obținută cu un scor cumulativ de +3 de către motorul de calcul Stockfish, chiar dacă materialul este egal, deoarece dezvoltarea pieselor lui White este mult mai bună. Aceasta înseamnă că poziția albă este cu aproximativ trei pioni mai bună.
2. *Căutarea*. Ca și jucătorii de șah buni, motoarele încearcă să privească profund în poziție. Cu cât pot vedea mai departe, cu atât mișcarea poate fi mai bună acum, deoarece pot evalua pozițiile care vor rezulta după cele mai bune mișcări posibile în viitor. Fiecare mișcare individuală de șah este numită „nivel” (un nivel) și adâncimea este explicată în câte nivele de adâncime. La 20 de nivele (10 mișcări albe și 10 mișcări negre), majoritatea motoarelor evaluează deja mult mai adânc și mai puternic decât oamenii. În funcție de timpul permis și de complexitatea poziției, motoarele pot arăta mai mult de 50 de nivele adâncime. Din poziția actuală, un motor începe să privească toate mișcările și răspunsurile posibile. Și apoi toate răspunsurile posibile la asta. Și apoi toate răspunsurile posibile la asta! Imaginează-ți că există 32 de mișcări posibile în orice poziție. După patru mișcări, există deja mai mult de un milion de poziții de evaluat. După doar alte patru mișcări, aceasta ar fi mai mult de un trilion de poziții. Asta devine extrem de nepractic. Deci, în schimb, motoarele încearcă să folosească „tăierea” inteligentă pentru a privi profund doar liniile cele mai promițătoare și să le ignore pe cele evident. Motorul menține o „variație principală” (PV) în mișcare a celor mai promițătoare mișcări în fiecare poziție. Motoarele tradiționale de șah folosesc funcții complexe de evaluare și algoritmi de căutare inteligentă pentru a găsi cea mai bună mișcare posibilă. Puterea lor este, de asemenea, legată de puterea procesării procesorului pe care o are telefonul, computerul sau serverul. Cu cât procesoarele sunt mai puternice și mai abundente, cu atât motorul devine mai puternic.

**Motivația**

Încă de la vârste fragede am cochetat cu jocul de șah și am încercat pe tot posibilul să devin cel mai bun la acest joc. Am participat la numeroase competiții școlare în care majoritatea turneelor au fost contra prietenilor mei. De fiecare dată încercam să copiem tactica unui jucător celebru pentru a putea fi imprevizibili din punct de vedere al strategiei de joc al partidei. Punând problema dincolo de sfera clasică, mi-am imaginat cum ar arăta un joc de șah fără ca adversarul să vină de acasă cu o tactică învățată ca pe o poezie și încercând sa creez un mediu în care să fii stimulat să gândești pe loc strategia pe loc și să o pui in aplicare. Pornind de la simplul joc de șah și fiind curios de strategiile pe care le poți aborda pentru câștigarea unei partide, mi-a venit ideea de a crea propriul joc de șah în care eu decid care sunt regulile de joc pentru a putea observa noi strategii de joc, gândite pe loc, pentru un diferit tip de date ale jocului. Am vrut sa surprind cum se poate adapta jucătorul la o iterație nou de șah, fără a mai putea folosi de vechile reguli de șah. De la această simplă idee până la final, a fost un drum lung, drum în care s-au alăturat și alte idei, precum schimbarea dimensiunii tablei de joc și definirea piesei care termină partida, de la excluderea anumitor piese din joc, pana la așezarea în mod liber a pieselor pe tabla de șah.

**Capitolul II**

**Tehnologii utilizate**

Limbajul C#

C# este un limbaj de programare orientat obiect dezvoltat de Microsoft. Se bazează pe C++ și Java dar are multe extensii suplimentare pentru a efectua o abordare de programare orientată pe componente. Acesta, programarea orientată pe obiecte, reprezintă de fapt sublinierea separării preocupărilor în ceea ce privește funcționalitatea largă disponibilă în cadrul unui anumit sistem software. În opinie mea punctele forte ale acestui limbaj care îl diferențiază de restul sunt:

* Modern și ușor de înțeles
* Sursă deschisă, eficient și rapid

Platforma .Net

Framework-ul .Net este un cadru software dezvoltat de Microsoft care rulează în principal pe Microsoft Windows. Include o bibliotecă mare de clase numită Framework Class Library (FCL) și oferă interoperabilitatea limbajelor (fiecare limbaj poate folosi cod scris în alte limbaje) în mai multe limbaje de programare. Programele scrise pentru .Net Framework se execută într-un mediu software (spre deosebire de un mediu hardware) denumit Common Language Runtime (CLR). CLR este o mașină virtuală de aplicație care oferă servicii precum securitatea, gestionarea memoriei și gestionarea excepțiilor. Ca atare codul computerului scris folosit .Net Framework se numește „cod gestionat”. FCL și CLR împreună constituie .NET Framework.

Serializarea și deserealizarea

Serializarea reprezintă transformarea unui obiect într-o secvență de octeți, din care să poată fi refăcut ulterior obiectul original. Acest din urmă procedeu poarta denumirea de deserealizare. Cele două operații au fost folosite cu scopul de a salva, la finalul configurării unei partide, starea jocului pentru a putea fi împărtășită și în altă sesiune de șah, pentru a putea distribui unui prieten. Serializarea și deserealizarea au o utilitate largă prin prisma faptului că:

* Mecanism simplu de salvare și restaurare a datelor
* Persistența obiectelor
* Compensarea diferențelor între sistemele de operare
* Transmiterea datelor în rețea

Acestea, în funcție de formatul ales, pot fi de mai multe tipuri :

* Binară : Serializarea binară este procesul în care convertim obiectele .NET într-un flux de byte. În serializarea binară, toți membrii publici, privați, chiar și cei care sunt doar de tipul read only, sunt serializați și convertiți în bytes;
* SOAP: SOAP este un protocol bazat pe XML, conceput special pentru transportul apelurilor de procedură folosind XML. Deoarece un mesaj SOAP este construit folosind XML, serializarea XML poate fi utilizat pentru serializarea claselor și generarea mesajelor SOAP codificate;
* XML: Serializarea XML este procesul de serializare a unui obiect .Net sub forma XML sau de la un obiect XML la .Net.

În aceasta abordare am folosit serializarea binară deoarece oferă o performanță îmbunătățită, este mai rapidă și chiar mai puternică în sensul că oferă suport pentru obiecte complexe, nu citește numai proprietăți ci chiar si referințe circulare. Clasa BinaryFormatter este cea care se ocupa cu serializarea binară în .Net. Această clasa are un constructor numit: BinaryFormatter() care se apelează pentru fiecare obiect al clasei și are rol de a inițializa o nouă instanță a clasei BinaryFormatter cu valorile default. Cu ajutorul metodei Serialize (Stream, Object) se serializează obiectul sau graficul obiectelor în fluxul dat, iar cu metoda Deserialize (Stream) se deserializează fluxul specificat într-un grafic de obiecte.

Interfața grafică

Windows Forms (cunoscut și sub denumirea de WinForms) este o bibliotecă a clasei de interfețe grafice folosite de către utilizator care este incorporată în .Net Framework. Scopul său principal este de a oferi o interfață mai ușoară pentru dezvoltarea aplicațiilor pentru laptop, tabletă sau calculatoare. WinForms pot fi folosite pentru crearea de aplicații fizice nu și pentru aplicații web. Aceasta este o tehnologie de programare bazată pe evenimente. GUI-urile sunt compuse din controale care au proprietăți, metode și evenimente. Aceasta librărie am folosit-o la interfața grafică a jocului de șah deoarece oferă un mediu de lucru ușor iar componentele sale comunică excelent între ele. Din această interfață am folosit lucruri predefinite precum enumuri pentru culorile atât de text cât și de background dar pentru a înfrumuseța mai bine aplicația am folosit culori custom redate cu ajutorul spectrului de culori rgb. Pentru a putea descrie fiecare pagină în parte am folosit Form-urile, ele fiind în principiu ferestre independente alei GUI-ului. În momentul când creăm o fereastră grafică nouă de tipul Forms, sunt generate automat 3 fișiere care conțin: primul conține clasa parțiala generală unde vom descrie metodele si evenimentele, cea de a doua conține descrierea grafică a controalelor existente în fereastra și a treia descrie resursele externe existente pentru fereastra în cauză.

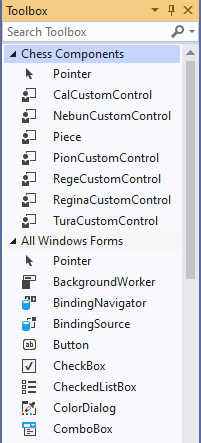
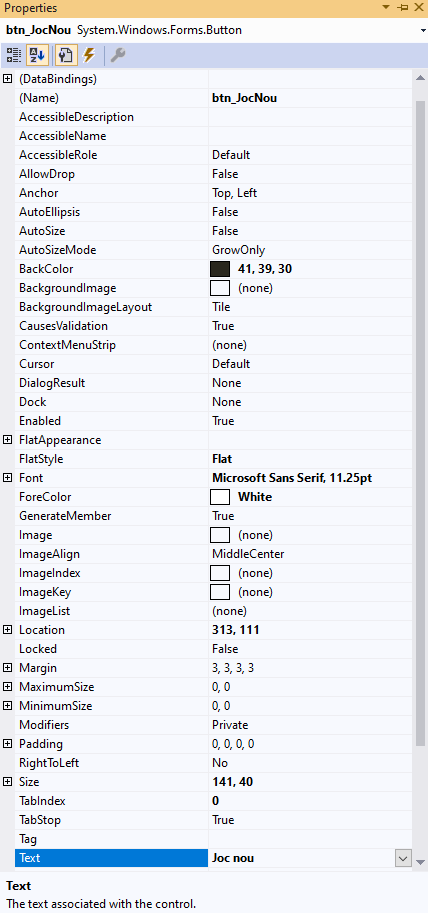
 Fereastra Toolbox presupune afișarea controalelor care pot fi adăugate la aplicația actuală. Când autorul dorește să creeze un control custom, dacă aplicația necesită această funcționalitate pe care nu o regăsim în controalele standard, putem folosi user control-urile. Acestea sunt cele mai simple tipuri de control. Ele se moștenesc de la clasa System.Windows.Forms.UserControl și urmează modelul de compoziție. De obicei, comenzile utilizatorului combină mai multe controale într-o logica unitate (cum ar fi un grup de casete de text pentru introducerea informațiilor cu privire la adresă).

Figura 2: Toolbox

În figura 2 putem observa în partea superioara controalele custom (Chess Components), iar în partea inferioară controalele predefinite din Windows Forms. Acestea au funcționalitatea de drag and drop în locația dorită ceea ce le face ușor de utilizat în aplicațiile desktop. În momentul când plasăm sub aceasta formă un control, atunci se creează automat o serie de instrucțiuni predefinite și se deschide în dreapta o fereastră proprietăți cu valorile default pentru a putea să le modelăm mai ușor. (Figura 3).

Figura 3: Proprietățile unui buton

În figura 3 avem ilustrat proprietățile unui buton. După cum spuneam, în momentul creării unui control se generează automat instrucțiuni default în clasa parțială „NumeleClasei.Designer”. În momentul în care modificăm orice proprietate din figura 3 atunci în clasa parțiala Designer se generează automat instrucțiunea aferentă pentru ca în momentul în care se va executa programul, acesta să știe exact informațiile despre controler, fereastra proprietăți fiind de fapt o interfață de lucru mai facilă cu controlerul. Cu ajutorul acestei ferestre putem seta si evenimentele corespunzătoare controler-ului selectat.

|  |
| --- |
| //  // btn\_JocNou  //  private System.Windows.Forms.Button btn\_JocNou;  this.Controls.Add(this.btn\_JocNou);  this.btn\_JocNou.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(41)))), ((int)(((byte)(39)))), ((int)(((byte)(30)))));  this.btn\_JocNou.FlatAppearance.BorderSize = 0;  this.btn\_JocNou.FlatStyle = System.Windows.Forms.FlatStyle.Flat;  this.btn\_JocNou.Font = new System.Drawing.Font("Microsoft Sans Serif", 11.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));  this.btn\_JocNou.ForeColor = System.Drawing.Color.White;  this.btn\_JocNou.Location = new System.Drawing.Point(313, 111);  this.btn\_JocNou.Name = "btn\_JocNou";  this.btn\_JocNou.Size = new System.Drawing.Size(141, 40);  this.btn\_JocNou.TabIndex = 0;  this.btn\_JocNou.Text = "Joc nou";  this.btn\_JocNou.UseVisualStyleBackColor = false;  this.btn\_JocNou.Click += new System.EventHandler(this.btn\_JocNou\_Click); |

În codul de mai sus avem exemplificat exact cum sunt generat instrucțiunile aferente modificărilor din proprietăți. Prima dată se declară obiectul în cauză după care este adăugat la fereastra curentă, urmând ca să fie descrise particularitățile acestuia. Această implementare reprezintă varianta statică a creării unui controler dar exista o și o modalitate dinamică în care putem crea un controler la faza de runtime descriind în clasa principală comportamentul controler-ului.

**Capitolul III**

**Prezentarea aplicației**

În acest capitol voi prezenta modul cum a fost creat jocul, problemele întâmpinate și rezolvarea lor, structura aplicației și diferitele particularități ale implementării.

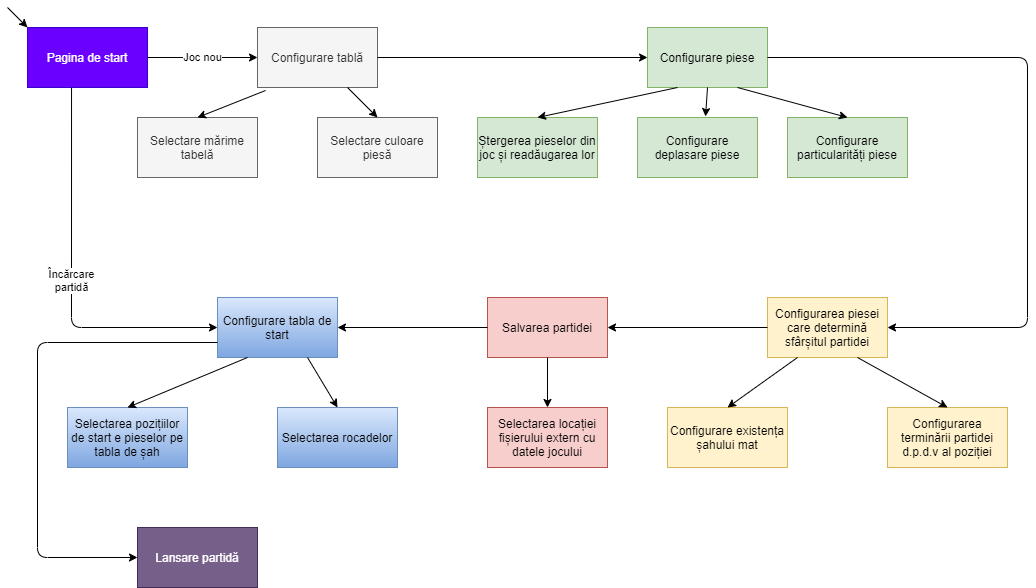


Diagrama 1: Arhitectura aplicației

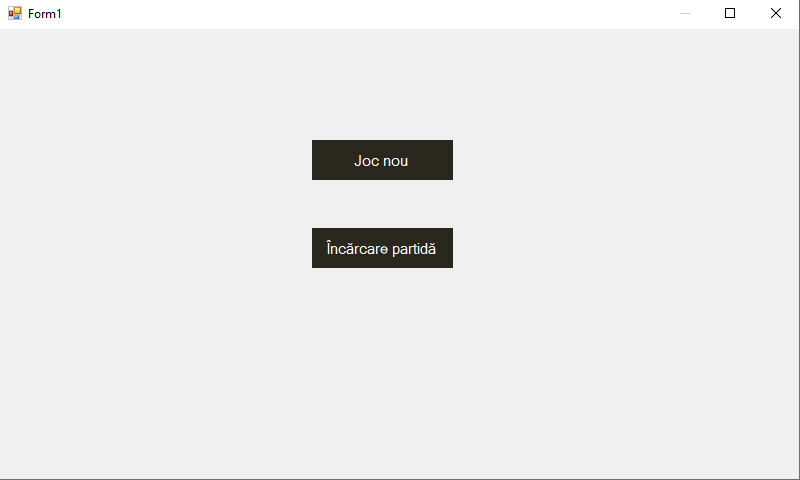
În diagrama 1 putem observa arhitectura aplicației. Aceasta debutează cu pagina de start. Primul buton „Joc nou” începe o nouă partidă de joc în care vom putea selecta de la bun început regulile de joc după bunul plac iar al doilea buton încarcă setările partidei dintr-un fișier extern și lansează o partidă nouă.

Figura 1:Pagina de start

Butoanele din Figura 1 au fost create cu ajutorul ToolBox-ului. Butonul „Încărcare partidă” deserializează datele dintr-un fișier extern pentru a putea încărca în memorie detaliile partidei. Clasa sau structura pe care o dorim să o serializăm trebuie să fie marcată cu atributul de serializare [Serializable] după cum urmează:

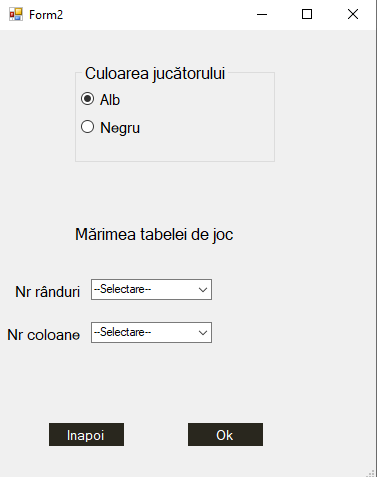
|  |
| --- |
| [Serializable]  public struct DateleNouluiJoc  {...} |

Operațiunea de serializare într-un fișier extern presupune serializarea fiecărui obiect în parte, de exemplu dacă dorim să serializăm un array, luăm fiecare element în parte și este serializat individual. Același procedeu se va aplica și pentru deserializare.

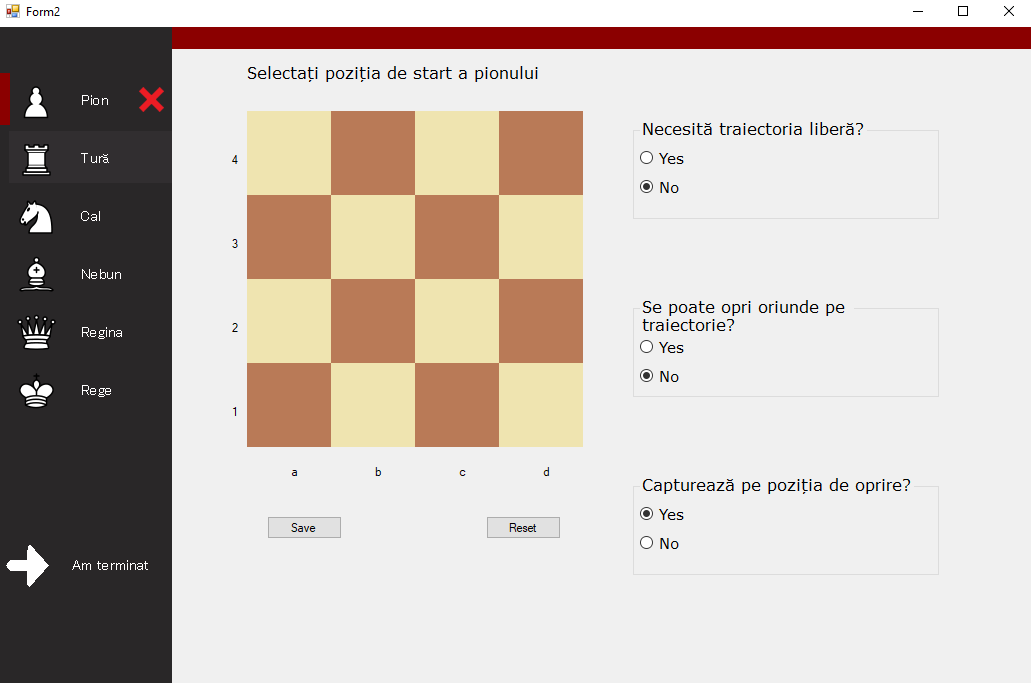
|  |
| --- |
| SaveFileDialog sfd = new SaveFileDialog();  sfd.Filter = "Text files (\*.txt)|\*.txt";  sfd.FileName = "GameRules";  if (sfd.ShowDialog() == System.Windows.Forms.DialogResult.OK)  {  try  {  string path = sfd.FileName;  IFormatter formatter = new BinaryFormatter();  Stream stream = new FileStream(path, FileMode.Create, FileAccess.Write);  formatter.Serialize(stream, date1);  for (int i = 0; i < date2.Length; i++)  formatter.Serialize(stream, date2[i]);  stream.Close();  }  catch(Exception ex)  {  Console.WriteLine(ex.Message);  } |

În urma deserializări din fișierul extern, în caz de reușită, se va deschide o fereastră de configurare a tabelei de joc, în scopul de a avea posibilitatea ca jucătorul să așeze după bunul plac piesele pe poziția de start. Prin această etapă jucătorul poate stabili și numărul de piese de același fel existent la poziția de start, acest număr nefiind limitat

În momentul în care selectăm primul buton se va deschide o fereastră adiacentă după cum putem vedea în figura 2. Aici putem selecta culoarea piesei dar și dimensiunea tablei de joc atât de lungime cât și pe lățime (rândurile și coloanele). În caz că nu selectăm nici o valoare dintre cele listate sau o valoare care nu face parte dintre ele, se va selecta automat valoarea cea mai mică, adică valoarea 4.

Figura 2: Pagina adiacentă

La apăsarea butonului ok se va deschide o fereastra nouă după putem observa în figura 3. În partea din stânga putem observa piese de joc disponibile, putem exclude anumite piese din joc dacă nu dorim acest lucru sau putem o data ce am exclus o piesa, putem să o reincludem. Pentru fiecare piesa am creat separat un user control pentru a putea defini în parte proprietățile pe care le are fiecare piesă. Pentru a reda tabla de șah am creat o structură de tip panel care are ca și background imaginea unei table de șah de 10x10 casuțe. În funcție de dimensiunea panel-ului aceasta se va decupa automat pentru dimensiunea cerută. Ca această proprietate să dea rezultate am desenat imaginea cu tabla de șah de 840pixeli pe 840 pixeli iar fiecare căsuță cu dimensiunea de 84x84. Pentru a putea controla piesele și căsuțele de pe acest panel am desenat pătrate invizibile deasupra acestuia astfel încât în momentul în care facem click pe „poză” să se activeze acele pătrate. Dimensiunea laturii pătratelor este de 84 pixeli.

Figura 3:Pagina principală de selectare a proprietăților

În partea dreapta a user control-ului avem o serie de întrebări care vor ajuta la descrierea tuturor detaliilor cu privire la traseul pe care îl poate urma piesa.

Bibliografie

* Titlu: Istoria jocului de șah;

Link: <https://www.chess.com/ro/blog/nic01ae/istoria-jocului-de-sah>; Data: 10/4/2021

* Titlu: Diferite tipuri de șah cu reguli schimbate;

Link: <https://www.chess.com/article/view/chess-variants>; Data: 12/4/2021

* Titlu: Șah engine;

Link: <https://www.chess.com/article/view/computer-chess-engines>; Data:13/4/2021

* Titlu: Istoria C sharp;

Link: <https://www.javatpoint.com/csharp-history>; Data:16/4/2021

* Titlu: User control;

Link: <https://www.akadia.com/services/dotnet_user_controls.html>; Data:20/4/2021

* Titlu: Serializarea;

Link: <http://net-informations.com/faq/net/serialization.htm> Data:14/5/2021

* Titlu: Serializarea binară;

Link: <https://docs.microsoft.com/en-us/dotnet/api/system.runtime.serialization.formatters.binary.binaryformatter?view=net-5.0>; Data:14/5/2021

1. Shannon, C. (1950). A Chess-Playing Machine. *Scientific American,* *182*(2), 48-51. Retrieved June 15, 2021, from http://www.jstor.org/stable/24967381 [↑](#footnote-ref-1)
2. Montfort, Nick (2005), Twisty Little Passages: An Approach to Interactive Fiction, MIT Press, ISBN 978-0-262-63318-5 [↑](#footnote-ref-2)