**Dezvoltarea aplicațiilor pe platforma .NET**

Pelle Remus-Nicolae

SDI, anul II

**Grafică 2D**

**Introducere**

Pentru a putea aplica elemente de grafică 2D în .Net, cel mai simplu proiect pe care îl putem crea este *Windows Forms App (.Net Framework).* După creare, vom vedea designer-ul formularului, unde putem adăuga diferite componente folosind toolbox-ul. Haideți să adăugăm un PictureBox și un Button

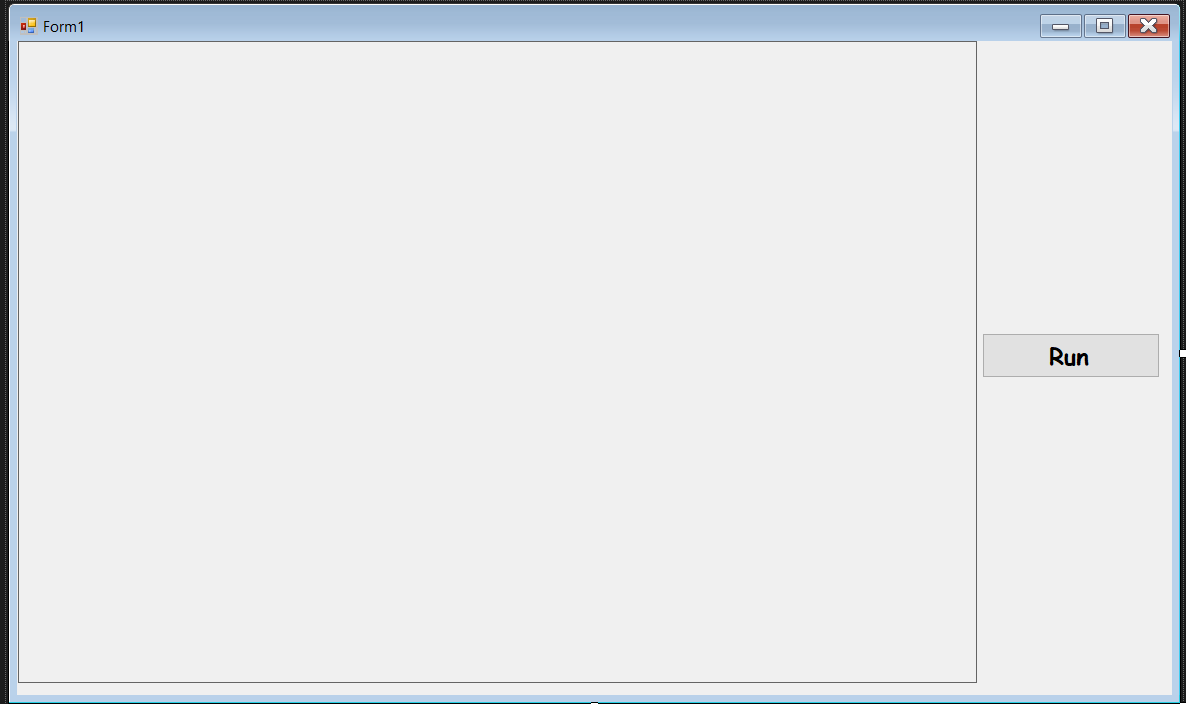


Figura 1: Formularul inițial

Pentru a scrie cod în cele mai comune evenimente ale fiecărei componente, putem da dublu click pe aceasta, și acest cod se va genera automat. Pentru formular în sine, se creează metoda *Form1\_Load*, iar pentru buton se creează metoda *button1\_Click*. Cele mai folosite obiecte pentru desenarea în pictureBox sunt un obiect *Graphics* și un obiect *Bitmap*. Obiectul bitmap va fi folosit ca și imaginea pictureBox-ului, iar obiectul graphics pentru a desena în acesta. Le vom inițializa când se încarcă formularul.

|  |
| --- |
| Graphics graphics;  Bitmap bitmap;  private void Form1\_Load(object sender, EventArgs e)  {  bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);  graphics = Graphics.FromImage(bitmap);  } |

Pentru a desena ceva în pictureBox, întâi trebuie să înțelegem care sunt regulile de desenare.

**Sistemul de coordonate**

În grafica 2D a calculatorului, originea (punctul de coordonate 0, 0) este colțul din stânga sus. Un punct de coordonate (x, y) se află la x pixeli în dreapta acestuia, respectiv la y pixeli în jos față de acesta. Deci se comportă ca și cadranul IV din sistemul de coordonate matematic.

Putem desena linii între diferite puncte folosind metoda *DrawLine*. Dacă nu specificăm coordonatele unui punct, acesta va fi inițializat în origine. Cum am specificat obiectului graphics să deseneze pe obiectul bitmap, nu vom vedea niciun rezultat. Așa că la final, trebuie să-i spunem la pictureBox1 că imaginea sa este bitmap.

Vom desena aceste linii la apăsarea butonului.

|  |
| --- |
| Random random = new Random();  private void button1\_Click(object sender, EventArgs e)  {  graphics.DrawLine(Pens.Black, new Point(), new Point(100, 100));  graphics.DrawLine(Pens.Black, new Point(50, 60), new Point(150, 80));  graphics.DrawLine(Pens.Black,  new Point(random.Next(pictureBox1.Width), random.Next(pictureBox1.Height)),  new Point(random.Next(pictureBox1.Width), random.Next(pictureBox1.Height)));  pictureBox1.Image = bitmap;  } |



Figura 2: Desenarea unor linii

Putem schimba culoarea și grosimea creionului folosit pentru desenare folosind un obiect din clasa *Pen*. Dar acum, observăm că dacă apăsăm de mai multe ori pe buton, vom avea mai multe linii aleatorii.

|  |
| --- |
| private void button1\_Click(object sender, EventArgs e)  {  graphics.DrawLine(new Pen(Color.Black, 2), new Point(), new Point(100, 100));  graphics.DrawLine(new Pen(Color.Blue, 3), new Point(50, 60), new Point(150, 80));  graphics.DrawLine(  new Pen(Color.FromArgb(random.Next(255), random.Next(255), random.Next(255)), random.Next(10)),  new Point(random.Next(pictureBox1.Width), random.Next(pictureBox1.Height)),  new Point(random.Next(pictureBox1.Width), random.Next(pictureBox1.Height)));  pictureBox1.Image = bitmap;  } |

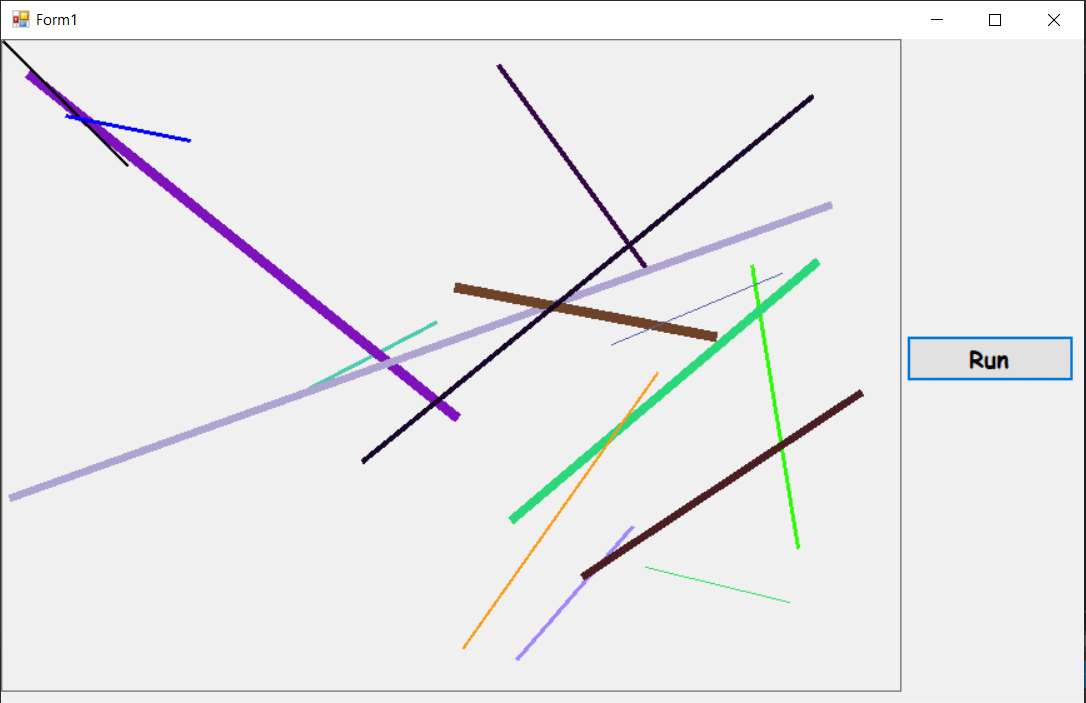


Figura 3: Linii de culori și grosimi aleatorii, dar avem mai multe decât ne așteptam

Soluția este simplă: înainte să desenăm liniile, folosim comanda *Clear*, care „șterge” conținutul deja desenat de graphics

|  |
| --- |
| graphics.Clear(Color.White); |

Dacă am dori să lucrăm cu un grafic matematic, astfel având nevoie de sistemele de coordonate în centrul ecranului, iar y-ul să mearga in sus, și nu în jos? Cea mai simplă sooluție ar fi să nu luăm punctele direct din clasa Point, ci să apelăm o functie creată de noi care mapează punctul dorit la noul punct, considerând centrul pictureBox-ului ca fiind originea. Putem și crea o funcție ce deseneaza axele de coordonate în sine.

|  |
| --- |
| // ...  Point center = new Point();  private void Form1\_Load(object sender, EventArgs e)  {  // ...  center = new Point(pictureBox1.Width / 2, pictureBox1.Height / 2);  }  private void button1\_Click(object sender, EventArgs e)  {  graphics.Clear(Color.White);  DrawXoYAxes();  graphics.DrawLine(new Pen(Color.Black, 2), GetMappedPoint(), GetMappedPoint(100, 100));  // ...  }  private void DrawXoYAxes()  {  graphics.DrawLine(new Pen(Color.Black, 2), GetMappedPoint(-center.X, 0), GetMappedPoint(center.X, 0));  graphics.DrawLine(new Pen(Color.Black, 2), GetMappedPoint(0, -center.Y), GetMappedPoint(0, center.Y));  }  private Point GetMappedPoint(int x, int y) => new Point(center.X + x, center.Y - y);  private Point GetMappedPoint() => GetMappedPoint(0, 0); |

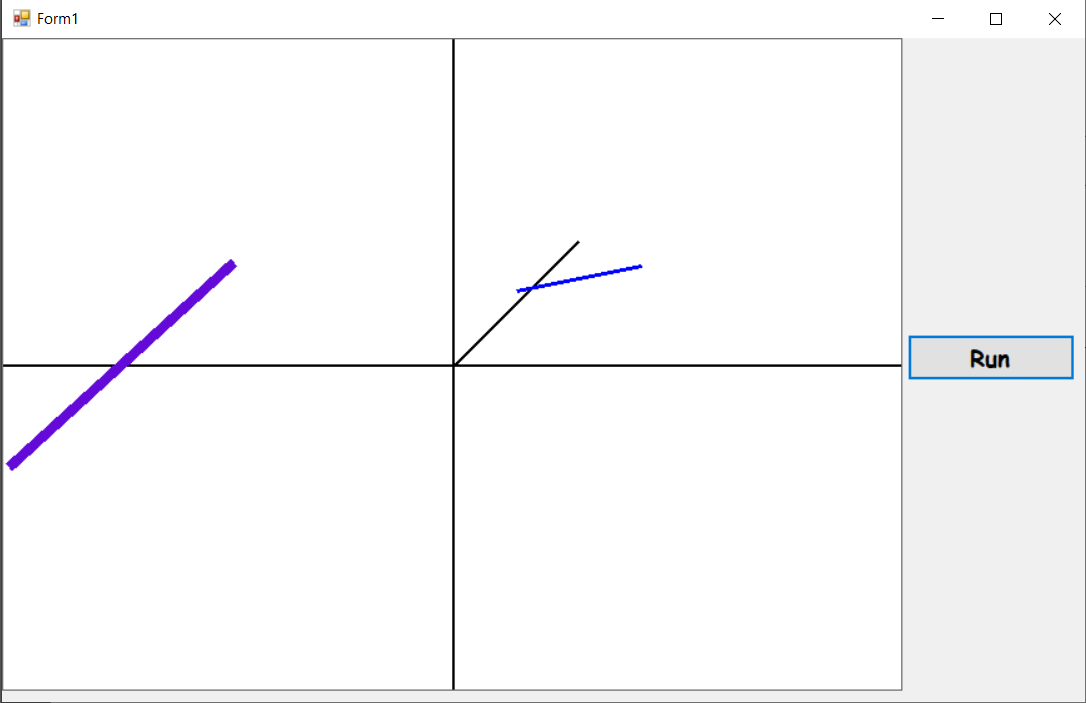


Figura 4: Sistem de coordonate propriu

Parcă este puțin greu să ne dăm seama că aceste linii reprezintă axele de coordonate. Pentru un simț familiar, am putea „desena” scrisul X, O si Y în locurile potrivite. De asemenea, putem desena aceste linii ca săgeți.

|  |
| --- |
| private void DrawXoYAxes()  {  Pen pen = new Pen(Color.Black, 2);  Brush brush = new SolidBrush(Color.Black);  Font font = new Font("Arial", 12, FontStyle.Bold);  graphics.DrawLine(pen, GetMappedPoint(-center.X, 0), GetMappedPoint(center.X, 0));  graphics.DrawLine(pen, GetMappedPoint(0, -center.Y), GetMappedPoint(0, center.Y));  graphics.DrawLine(pen, GetMappedPoint(center.X-20,10), GetMappedPoint(center.X,0));  graphics.DrawLine(pen, GetMappedPoint(center.X-20,-10),GetMappedPoint(center.X,0));  graphics.DrawLine(pen, GetMappedPoint(10,center.Y-20), GetMappedPoint(0,center.Y));  graphics.DrawLine(pen, GetMappedPoint(-10,center.Y-20),GetMappedPoint(0,center.Y));  graphics.DrawString("X", font, brush, GetMappedPoint(center.X - 20, -10));  graphics.DrawString("O", font, brush, center);  graphics.DrawString("Y", font, brush, GetMappedPoint(10, center.Y));  } |

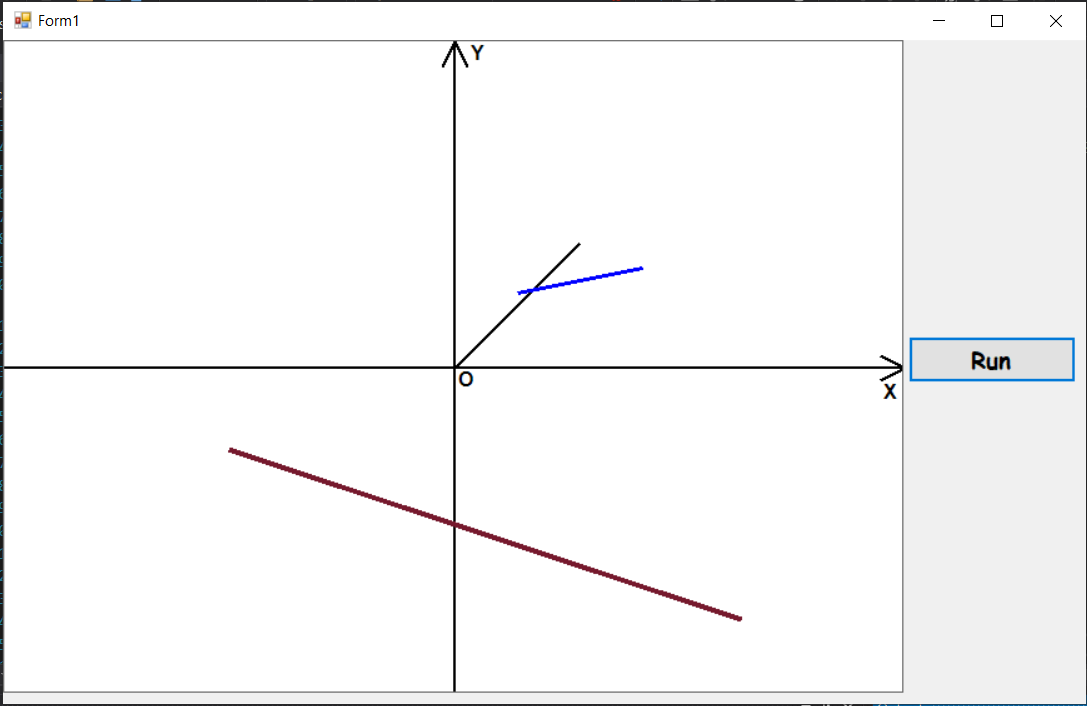


Figura 5: Axele de coordonate desenate într-un mod mai familiar, ușor de recunoscut

**Alte posibile forme de desenat folosind Graphics**

Pe lângă linii și text, sunt multe alte forme ce se pot desena:Elipsă, Dreptunghi, Poligon, etc. Acestea se pot și umple dacă așa dorim.

|  |
| --- |
| private void button1\_Click(object sender, EventArgs e)  {  // ...  graphics.FillRectangle(new SolidBrush(Color.ForestGreen), 0,0, center.X, center.Y);  Point[] points = new Point[] { GetRandomMappedPoint(), GetRandomMappedPoint(), GetRandomMappedPoint(), GetRandomMappedPoint(), GetRandomMappedPoint() };  graphics.FillPolygon(  new SolidBrush(Color.FromArgb(random.Next(255), random.Next(255), random.Next(255))),  points);  graphics.DrawEllipse(Pens.Red, center.X-(center.Y/2),center.Y/2,center.Y,center.Y);  pictureBox1.Image = bitmap;  }  // ...  private Point GetRandomMappedPoint() =>  GetMappedPoint(random.Next(-center.X, center.X), random.Next(-center.Y, center.Y)); |

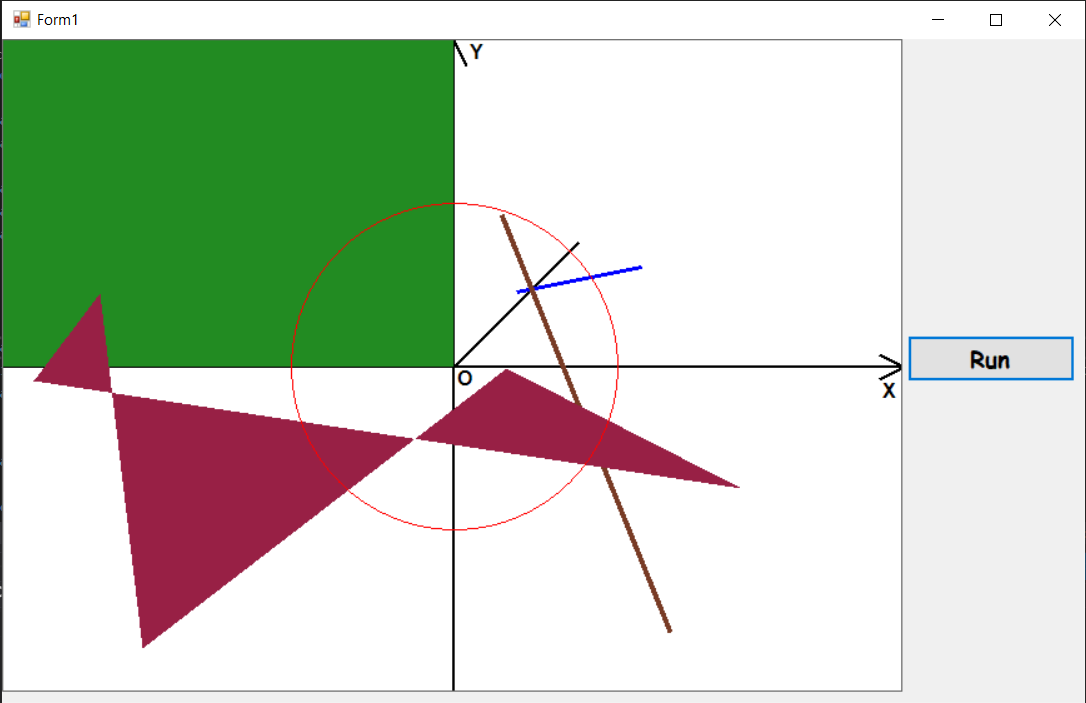


Figura 6: Diferite forme desenate folosind Graphics

**Aplicare practică a graficii 2D: Jocuri 2D**

Dupa teorie, este bine să știm și cum sau la ce să punem în practică. Crearea unui joc 2D mai simplu este un mod bun de a învăța mai bine grafica 2D și alte lucruri ce se pot folosi împreună cu aceasta din interiorul aplicațiilor WFA.

Sunt o multitudine de jocuri ce pot avea la bază o matrice de elemente:

* Un joc platformer poate avea hărțile nivelelor ca și o matrice de diferite tipuri de platforme: spațiu gol, platformă solidă, platformă care rănește jucătorul, platformă cu recompense înăuntru, platforme pentru câștigat, si pot fi diferite variații pentru fiecare din acestea, cu aspect diferit. Exemplu: jocul Super Mario;
* Un Role-Playing Game (RPG) va avea similar, matricea hărților, dar tile-urile sunt puțin diferite, cum harta este privită de sus, nu din lateral. Exemplu: jocul Arcuz;
* Jocuri specifice care au în mod natural o matrice la bază, cum ar fi Tetris, 2048 sau Minesweeper.

Ca și aplicație practică, vom construi jocul Minesweeper de la 0 pentru a observa întreg procesul creării acestuia de la început. Creăm un nou proiect WFA, adăugăm un pictureBox și un buton, și dăm dublu-click pe formular si pe buton.

După cum spuneam, vom avea o matrice de elemente. Aceste elemente vor fi fiecare pătrățel din joc în acest caz, deci vom crea clasa Tile pentru acestea. În clasă, trebuie să ținem minte valoarea de pe tile, poziția acestui tile pe linie și pe coloană, și să avem o referință către butonul ce urmează să fie afișat pe ecran.

|  |
| --- |
| public class Tile  {  public int Line { get; set; }  public int Column { get; set; }  public int Value { get; set; }  public Button Button { get; set; }  public Tile(int line, int column)  {  Line = line;  Column = column;  Button = new Button();  }  } |

Dar dacă încercăm acum să creăm un Tile, butonul nu apare pe ecran. Asta pentru că nu are niciun părinte încă. Vrem ca pictureBox1 să fie părintele acestor butoane, așa că vom crea o clasă publică și statică numită Engine unde vom pune toate proprietățile necesare pentru funcționalitate. Vom avea și un PictureBox Display, pe care îl inițializăm în metoda Initialize. Apelăm acea metodă la Form1\_Load.

Tot în Engine vom adăuga proprietățile Lines, Columns și Size, pentru a determina câte elemente vom avea în matrice și ce dimensiune vor avea acestea. În clasa Tile, dăm mai multe detalii despre Buton.

|  |
| --- |
| public static class Engine  {  public static int Lines { get; set; }  public static int Columns { get; set; }  public static int Size { get; set; }  public static PictureBox Display { get; set; }  public static void Initialize(PictureBox pictureBox)  {  Display = pictureBox;  Lines = 8;  Columns = 10;  Size = Display.Height / Lines;  }  }  public partial class Form1 : Form  {  // ...  private void Form1\_Load(object sender, EventArgs e)  {  Engine.Initialize(pictureBox1);  }  // ...  }  public class Tile  {  // ...  public Tile(int line, int column)  {  // ...  Button = new Button();  Button.Parent = Engine.Display;  Button.Size = new Size(Engine.Size, Engine.Size);  }  } |

Acum vrem să creăm matricea și să o vedem pe ecran. Le vom adăuga în clasa Engine.

|  |
| --- |
| public static Tile[,] Matrix { get; set; }  public static void InitializeMatrix()  {  Matrix = new Tile[Lines, Columns];  for (int i = 0; i < Lines; i++)  for (int j = 0; j < Columns; j++)  {  Matrix[i, j] = new Tile(i, j);  }  } |



Figura 7: Prima încercare de a afișa butoanele

Observăm că butoanele nu sunt afișate corect. Trebuie să le calculăm și poziția la inițializare. Ne vom folosi de valoarea liniei și a coloanei.

Mai departe, adăugăm un număr de mine pe teren: schimbăm valoarea unor poziții i și j aleatoare la 9. Trebuie să fim atenți să nu se repete poziția, rezultând în mai puține mine.

Următorul pas va fi să calculăm numărul de mine din jurul fiecărui Tile. Pentru asta, parcurgem matricea de 3x3 din jurul fiecărui tile și creștem valoarea de fiecare dată când întâlnim valoarea 9. Aici trebuie să fim atenți să nu ieșim înafara matricei. După calcule, dăm culoarea corespunzătoare textului de pe buton în funcție de valoare și pentru vizualizare, adăugăm textul la buton.

Facem și o funcție de resetare a valorilor, iar la apăsarea butonului New Game, vom reseta matricea, adăuga mine și calcula valorile acesteia.

|  |
| --- |
| public static void StartNewGame()  {  ResetMatrix();  AddMines();  CalculateMatrixValues();  }  private static void ResetMatrix()  {  for (int i = 0; i < Lines; i++)  for (int j = 0; j < Columns; j++)  {  Matrix[i, j].Value = 0;  Matrix[i, j].Button.Text = "";  Matrix[i, j].Button.ForeColor = Color.Black;  }  }  private static void AddMines()  {  int nrOfMines = 20;  for (int k = 0; k < nrOfMines; k++)  {  int i, j;  do  {  i = Random.Next(Lines);  j = Random.Next(Columns);  } while (Matrix[i, j].Value == 9);  Matrix[i, j].Value = 9;  }  }  private static void CalculateMatrixValues()  {  for (int i = 0; i < Lines; i++)  for (int j = 0; j < Columns; j++)  {  if (Matrix[i, j].Value == 9)  {  Matrix[i, j].Button.Text = "X";  continue;  }  for (int k = Math.Max(i - 1, 0); k <= Math.Min(i + 1, Lines-1); k++)  for (int l = Math.Max(j - 1, 0); l <= Math.Min(j + 1, Columns-1); l++)  {  if (Matrix[k, l].Value == 9)  Matrix[i, j].Value++;  }  if (Matrix[i, j].Value == 0)  continue;  Matrix[i, j].Button.Text = Matrix[i, j].Value.ToString();  switch (Matrix[i, j].Value)  {  case 1: Matrix[i, j].Button.ForeColor=Color.FromArgb(20, 100, 200);break;  case 2: Matrix[i, j].Button.ForeColor = Color.Green; break;  case 3: Matrix[i, j].Button.ForeColor = Color.Red; break;  case 4: Matrix[i, j].Button.ForeColor = Color.DarkBlue; break;  case 5: Matrix[i, j].Button.ForeColor = Color.DarkRed; break;  case 6: Matrix[i, j].Button.ForeColor = Color.DarkTurquoise; break;  case 7: Matrix[i, j].Button.ForeColor = Color.Black; break;  default: break;  }  }  } |



Figura 8: Afișarea butoanelor corect, completate cu mine și restul valorilor

Dar jocul trebuie să ascundă valorile acestea până când dai click pe un tile. Vom adăuga o variabilă WasVisited în clasa Tile, iar abia atunci când acel tile a fost visitat să îl afișăm. Dacă se dă click pe un tile cu mină, am pierdut jocul, iar dacă nu, traversăm matricea și dezvăluim valorile din spațiile libere.

|  |
| --- |
| public void Visit()  {  WasVisited = true;  Button.BackColor = Color.White;  if (Value != 0)  Button.Text = Value.ToString();  }  private void Button\_MouseClick(object sender, MouseEventArgs e)  {  if (WasVisited)  return;  if (Value == 9)  Engine.GameOver();  else  Engine.Traverse(Line, Column);  }  public static void Traverse(int line, int column)  {  if (line >= 0 && line < Lines && column >= 0 && column < Columns && !Matrix[line, column].WasVisited)  {  Matrix[line, column].Visit();  if (Matrix[line, column].Value == 0)  {  Traverse(line - 1, column);  Traverse(line, column + 1);  Traverse(line + 1, column);  Traverse(line, column - 1);  }  }  }  public static void GameOver()  {  for (int i = 0; i < Lines; i++)  for (int j = 0; j < Columns; j++)  {  Matrix[i, j].Button.Enabled = false;  if (Matrix[i, j].Value == 9)  Matrix[i, j].Button.Text = "X";  }  MessageBox.Show("You Lose!");  } |

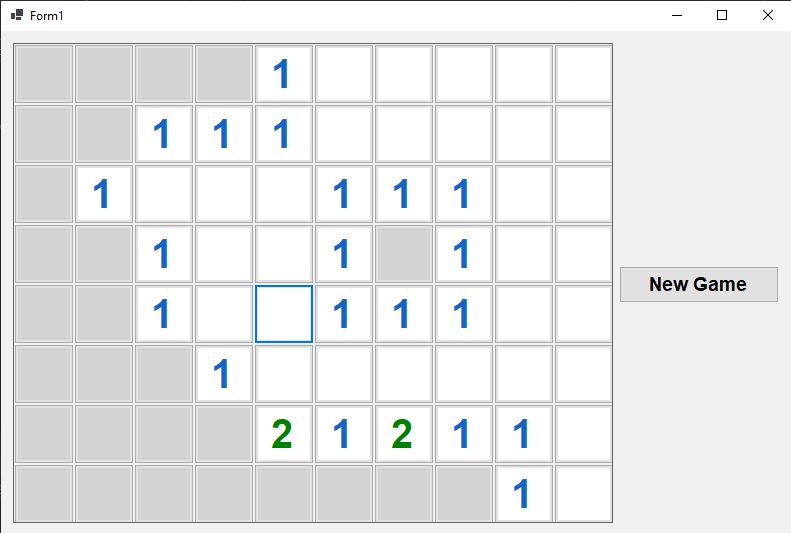


Figura 9: Ascunderea valorilor tile-urilor până când dăm click pe unul

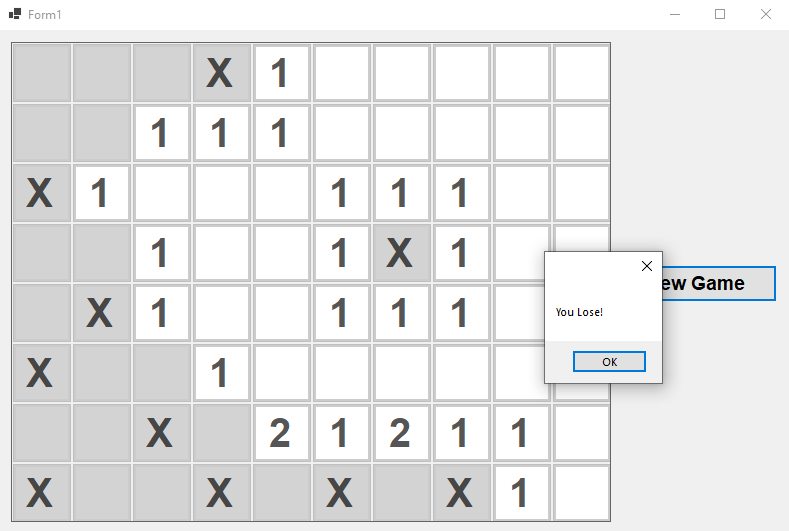


Figura 10: Jocul pierdut

Acum că putem peirde jocul, mai trebuie să îl putem și câștiga. Când mișcăm rotița de la mouse, vom pune un steguleț pe butonul respectiv (sau îl vom lua de pe el daca deja există). Vom reprezenta asta prin fundalul roșu al tile-ului. Când toate tile-urile cu mine sunt cu steag, și niciun alt tile nu e cu steag, atunci am câștigat jocul.

|  |
| --- |
| private void Button\_MouseWheel(object sender, MouseEventArgs e)  {  if (WasVisited)  return;  if (IsFlagged)  Button.BackColor = Color.LightGray;  else  Button.BackColor = Color.Red;  IsFlagged = !IsFlagged;  Engine.CheckIfYouWon();  }  public static void CheckIfYouWon()  {  bool ok = true;  for (int i = 0; i < Lines; i++)  for (int j = 0; j < Columns; j++)  {  if (Matrix[i, j].Value == 9 && !Matrix[i, j].IsFlagged)  ok = false;  if (Matrix[i, j].Value != 9 && Matrix[i, j].IsFlagged)  ok = false;  }  if (ok)  {  MessageBox.Show("You Win!");  Display.Enabled = false;  }  } |

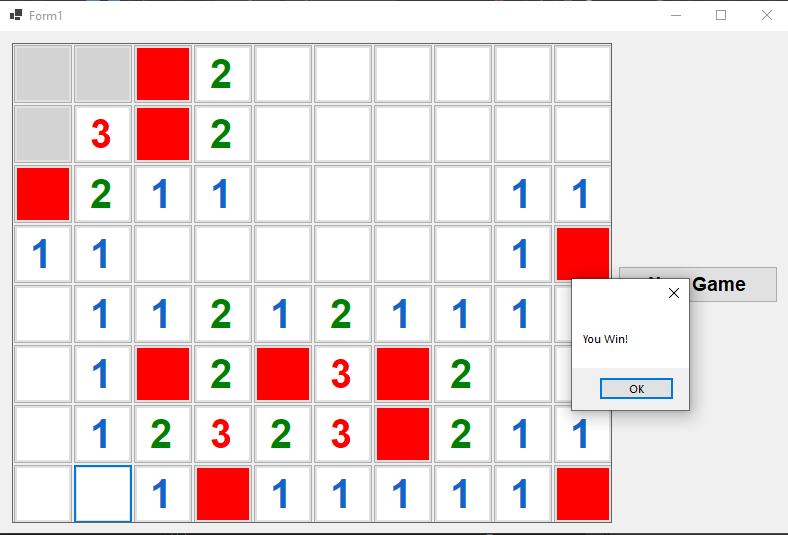


Figura 11: Jocul castigat