**1. Algoritmul de aproximare al lui Pi folosind Monte Carlo**

Metodele Monte Carlo sunt o clasă largă de algoritmi de calcul care se bazează pe modelarea aleatorie repetată pentru a obține rezultate numerice. Unul dintre exemplele de bază pentru a începe cu algoritmul Monte Carlo este estimarea lui Pi.

Ideea este de a simula punctele aleatoare (x, y) dintr-un plan 2D cu domeniul un pătrat cu unitatea laturii 1. Imaginaţi-vă un cerc în interiorul aceluiași domeniu cu același diametru și înscris în pătrat. Apoi calculăm raportul dintre punctele din cerc și numărul total de puncte generate.

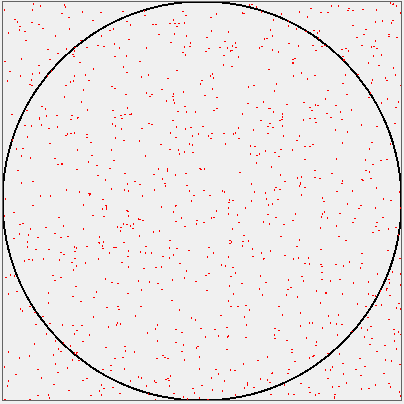


Fig. 1.1: Puncte aleatoare într-un pătrat, de unde se poate calcula raportul cerut.

Ştim că aria pătratului dat este 1, iar aria cercului dat este

, iar pentru un număr foarte mare de puncte (acoperind întreaga suprafaţă a pătratului)

Pentru acest program, nu este necesară afişarea acestor puncte. Generăm un punct aleator, cu x şi y valori între 0 şi 1 şi pur şi simplu verificăm

Dacă este adevărat, incrementăm numărul de puncte din cerc. Pentru un rezultat cât mai apropiat de valoarea lui pi, generăm un număr foarte mare de puncte. De aceea este „Aproximarea”, şi nu „Calcularea” lui pi.

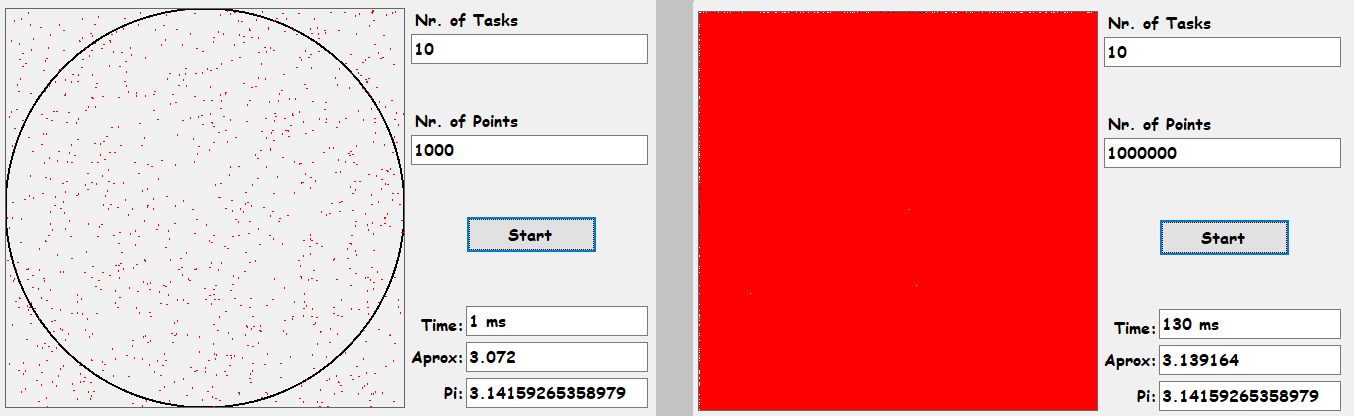


Fig. 1.2: Aproximări pentru numere de puncte diferite.

Pentru paralelizarea programului, pur şi simplu împărţim numărul de puncte la numărul de taskuri, iar fiecare task calculează numărul de puncte din cerc. La final, facem suma acestor numere pentru numărul total de puncte generate în cerc. Din moment ce vom avea mai multe puncte care se generează deodată datorită taskurilor, timpul diferă şi în funcţie de taskuri.

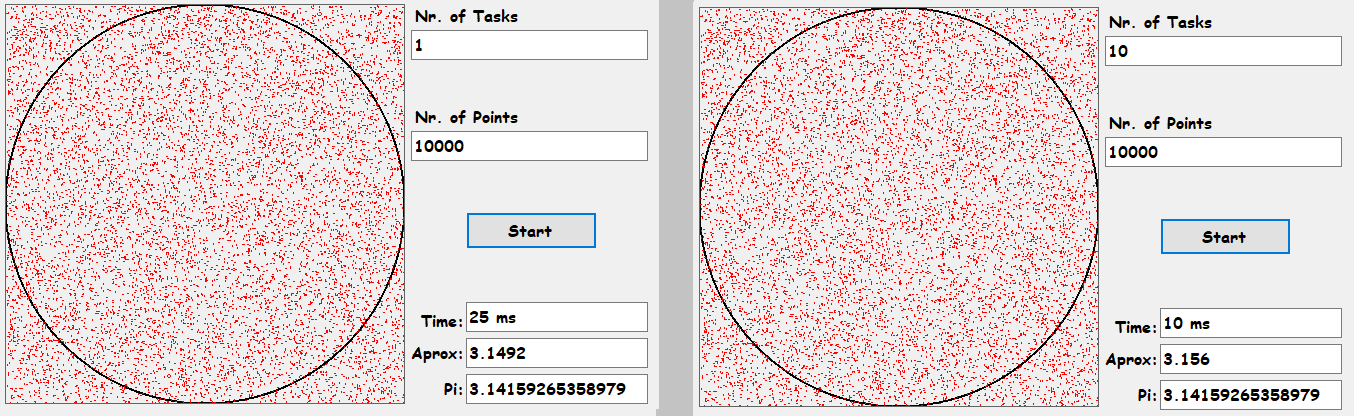


Fig. 1.3: Timpul diferă în funcţie de numărul de task-uri.

|  |
| --- |
| public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  Pi.Text = Math.PI.ToString();  }  Graphics grp;  Bitmap bmp;  List<PointF> pointList;  int scale;  private void Start\_Click(object sender, EventArgs e)  {  scale = pictureBox1.Width;  bmp = new Bitmap(scale, scale);  grp = Graphics.FromImage(bmp);  grp.DrawEllipse(new Pen(Color.Black, 2), -1, -1, scale - 1, scale - 1);  try  {  Stopwatch watch = Stopwatch.StartNew();  double result = CalculatePi(uint.Parse(nrOfThreads.Text), ulong.Parse(Points.Text));  watch.Stop();  Aprox.Text = result.ToString();  Time.Text = watch.ElapsedMilliseconds + " ms";  foreach (PointF p in pointList)  grp.DrawEllipse(Pens.Red, p.X, p.Y, 1, 1);  pictureBox1.Image = bmp;  }  catch  {  MessageBox.Show("Please insert numbers greater than 0!", "Error");  }  }  double CalculatePi(uint threads, ulong points)  {  ulong n = points / threads;  int pointsInCircle = 0;  Task<int>[] tasks = new Task<int>[threads];  pointList = new List<PointF>();  for (int i = 0; i < threads; i++)  {  tasks[i] = Task<int>.Factory.StartNew(() =>  {  Random random = new Random();  int insideCircleCount = 0;  for (uint j = 0; j < n; j++)  {  double x = random.NextDouble();  double y = random.NextDouble();  double result = x \* x + y \* y;  if (result <= 1)  insideCircleCount++;  pointList.Add(new PointF((float)x\*scale, (float)y\*scale));  }  return insideCircleCount;  });  }  for (int i = 0; i < threads; i++)  pointsInCircle += tasks[i].Result;  return 4.0 \* pointsInCircle / points;  }  } |

**2. Simularea unui mecanism matriceal cu transfer de căldură**

Automatele celulare sunt matrice ale căror elemente au un număr finit de stări, cum ar fi pornit și oprit. Pentru fiecare element, definim un set de elemente ca fiind vecinii acestuia. Se pornește de la o stare inițială, iar în următoarea generație, fiecare element este determinat de o regulă matematică în funcție de vecinii acestuia. Această regulă nu se modifică în timp și se aplică întregii matrice simultan.

Pentru transferul de căldură, vecinii „atomilor” sunt: atomii de deasupra, de sub, din stânga și din dreapta acestora. Excepțiile sunt elementele de pe marginea matricei, care au doar 3 sau 2 vecini.

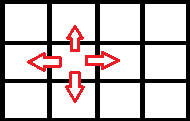


Fig. 2.1: Vecinii unui element

Stările elementelor vor fi temperaturile moleculelor, cu valori float de la 0 la 100 de grade celsius. Fiecărei valori îi corespunde o culoare pentru a putea realiza aplicaţia vizuală.



Fig. 2.2: Stările elementelor reprezentate de temperaturi între 0 şi 100°C

Regula matematică după care se decid generaţiile următoare este dedusă din definiţia căldurii la nivel microscopic: mişcare. Atomii dintr-un mediu se mişcă (sau vibrează) mai rapid cu cât mediul este mai fierbinte. Transferul de căldură se poate realiza în mai multe moduri, printre care se numără şi cunductivitatea, la obiectele solide. Considerând două cuburi de cupru egale în masă şi dimensiune, cu temperaturile constante, unul de 25°C şi celălalt de 75°C, ce se întâmplă atunci când aducem cele două cuburi în contact? (considerăm ca nu există alte surse de a transmite sau prelua căldura). În zona de contact, cum atomii de 75°C vibrează mai rapid decât cei de 25°C, la un moment dat un atom se va ciocni de altul mai încet, transferându-i energia sa cinetică. Şi astfel, în cubul de 25°C va exista un atom de 75°C şi invers. Acest lucru se va întâmpla pe întreaga suprafaţă, rezultând într-un strat de atomi micşti de 25°C si de 75°C în ambele cuburi. Aceasta se va realiza pentru întregile mase de atomi ale cuburilor şi, în timp, temperatura ambelor cuburi va fi 50°C, media aritmetică a celor două temperaturi.

Acestea fiind spuse, regula matematică este media aritmetică a vecinilor pentru fiecare atom din matrice. Din moment ce un atom are 4 vecini, va primi căldură de la fiecare dintre aceştia, şi le va da căldură la fiecare în mod egal. Dacă scriem formula ca şi o sumă de 4 fracţii, observăm că temperatura din fiecare element se împarte la 4. Cum fiecare element apare doar în formulele vecinilor săi, suma aceasta ne dă valoarea iniţială, deci nu există pierderi sau căştiguri de căldura, deci se respectă şi legea conservării energiei.

Având toate acestea clarificate, tot ce ne mai trebuie este starea iniţială. Am decis să se poată alege dintre mai multe stări iniţiale, prima ilustrând exemplul cu cele două cuburi de dimensiuni egale, dar cu temperaturi iniţiale diferite. Putem observa cum, pe parcursul rulării aplicaţiei, zona de contact ajunge foarte repede la valoarea de 50°C, restul elementelor urmând imediat după, până când întreaga matrice are doar valori de 50°C. Acest experiment se poate realiza cu diferite valori iniţiale pentru cele două obiecte, dar de fiecare dată valoarea finală va fi media aritmetică a celor două valori iniţiale.

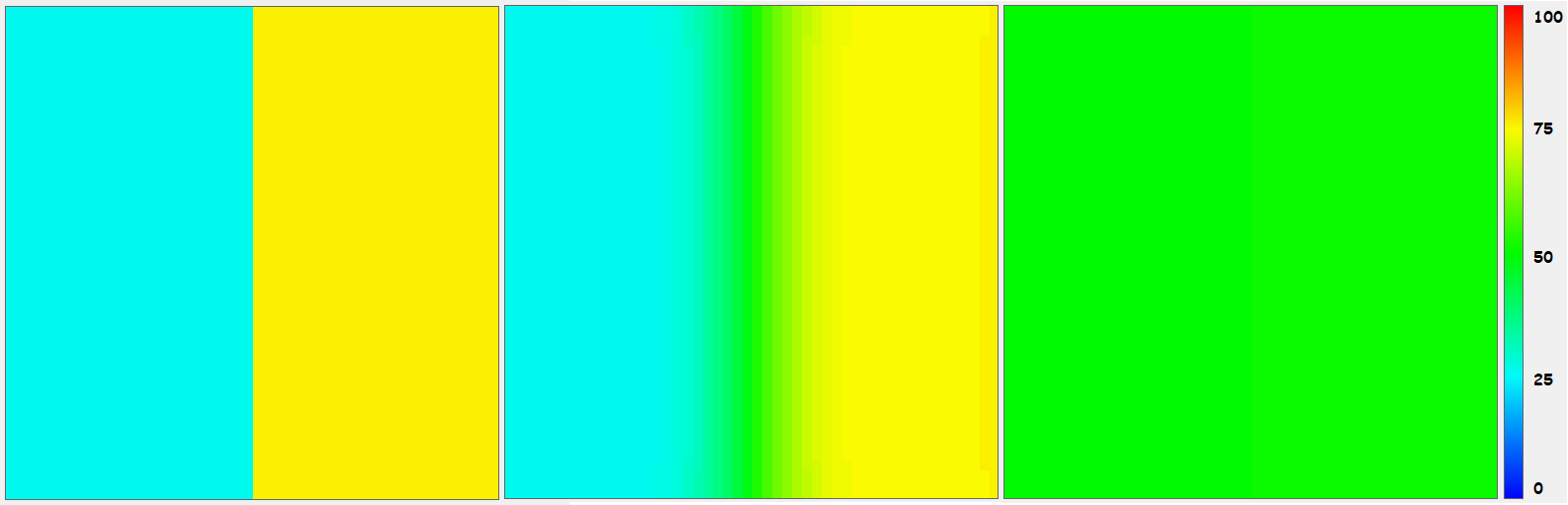


Fig. 2.3: Starea iniţială 1; Un rezultat intermediar; Starea finală 1.

Pentru următoarea stare iniţială, am considerat un obiect la temperatura 0°C, dar cu o regiune în centrul acestui obiect la 100°C. La scurt timp, se observă cum temperatura este de o valoare intermediară de aproximativ 50°C în regiunea care era iniţial fierbinte, iar pe măsură ce se îndreaptă spre extremităţile matricei, valoarea scade treptat. Într-un final, întreaga matrice va avea aceeaşi temperatură datorită transferului de căldură, temperatură de o valoare puţin mai mare decât cea iniţială: între 0°C şi 25°C. Această valoare dă o culoare plăcută de albastru întregii matrice.

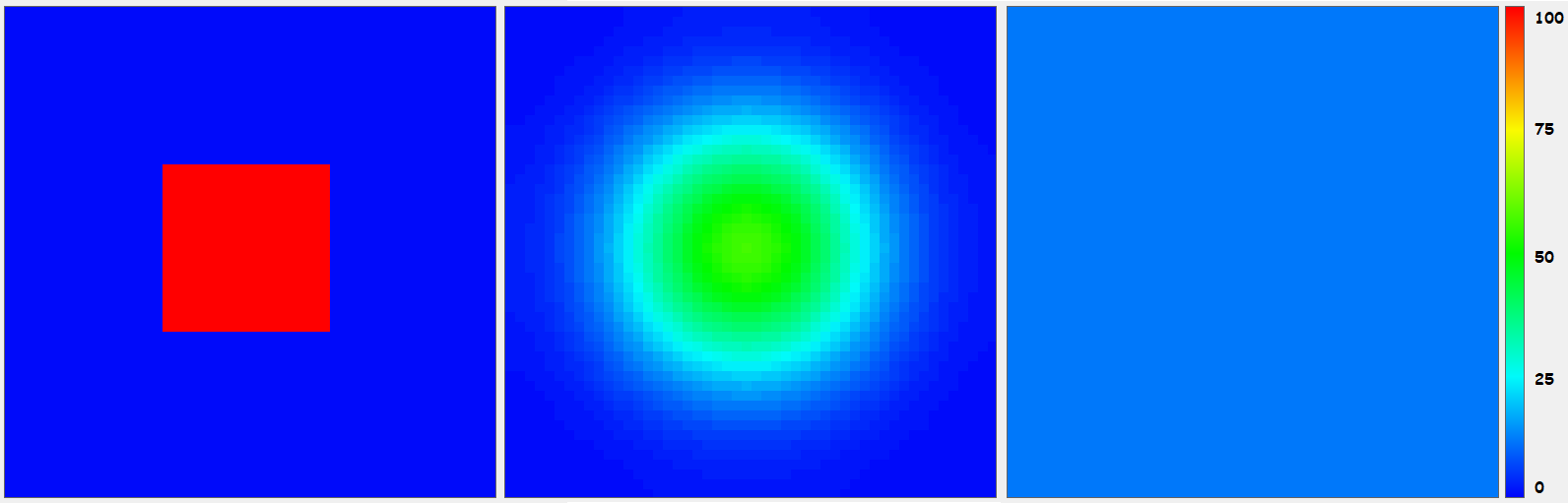


Fig. 2.4: Starea iniţială 2; Un rezultat intermediar; Starea finală 2.

Pentru următorul experiment, am ales o stare iniţială şi am ales o sursă cu temperatură fixată: întreaga matrice are temperatura 0°C, iar în mijloc avem o sursă de căldură de 100°C pentru care regula matematică nu se aplică. Astfel, simulăm o lumânare într-o încăpere rece, dar desigur, cum încăperea este în 2D şi nu în 3D, căldura se va transmite mult mai repede. După câteva iteraţii, se ajunge la un rezultat similar cu cel intermediar de la exemplul anterior, doar că în centru încă avem punctul roşu de 100°C. Căldura se transmite foarte încet, deoarece avem o singură sursă de căldură pentru o suprafaţă vastă, deci chiar şi în imediata vecinătate a lumânării nu se resimte temperatura sa, de 100°C. Însă acest lucru se va schimba după ce întreaga matrice are valori de cel puţin 25°C, apoi 50°C. Cum atomii din apropierea lumânării nu mai trebuie să îşi transmită toată energia atomilor vecini, aceştia încep în sfârşit să se încălzească. În jurul lumânării, temperaturile se apropie de 100°C, iar restul matricei de 75°C. După aceasta, transferul în restul matricei se face mult mai rapid şi, în final, se ajunge la 100°C în întreaga matrice.

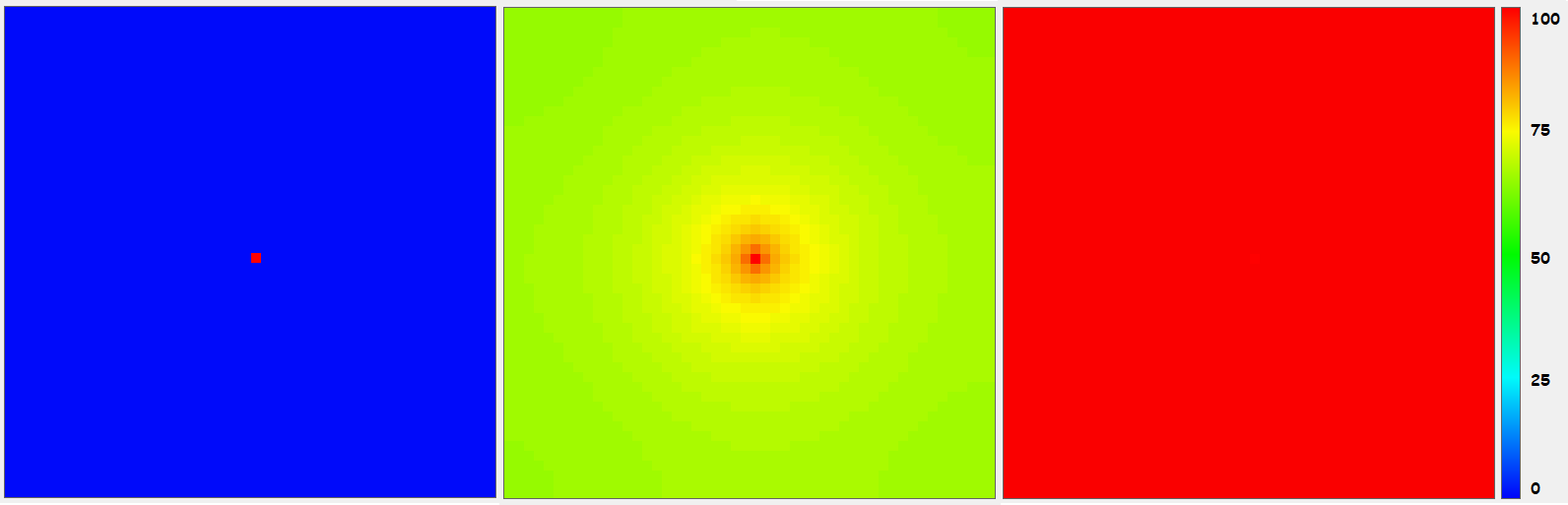


Fig. 2.5: Starea iniţială 3; Un rezultat intermediar; Starea finală 3.

Pentru a nu mai da de aceeaşi problemă ca şi în exemplul anterior, în care sursa de temperatură fixată era prea mică relativ la restul suprafeţei, într-un ultim exemplu am decis să dau pereţilor matricei temperaturi constante şi să observăm rezultatele intermediare. Peretele din stânga are 50°C, cel din dreapta 0°C, cei de sus şi de jos 100°C, iar restul matricei va avea 0°C. Cum avem mai mulţi pereţi de temperaturi ridicate, se observă imediat transferul de căldură în toată matricea. Acesta se realizează treptat, si astfel se realizează un curcubeu de culori la toate rezultatele intermediare. În rezultatul final, se pot observa diferitele efecte date de incălzirea, respectiv răcirea matricei în diferite zone. Desigur, se ajunge şi la un rezultat final, deoarece valorile pereţilor sunt constante şi se va ajunge la un moment dat la valori constante în matrice.

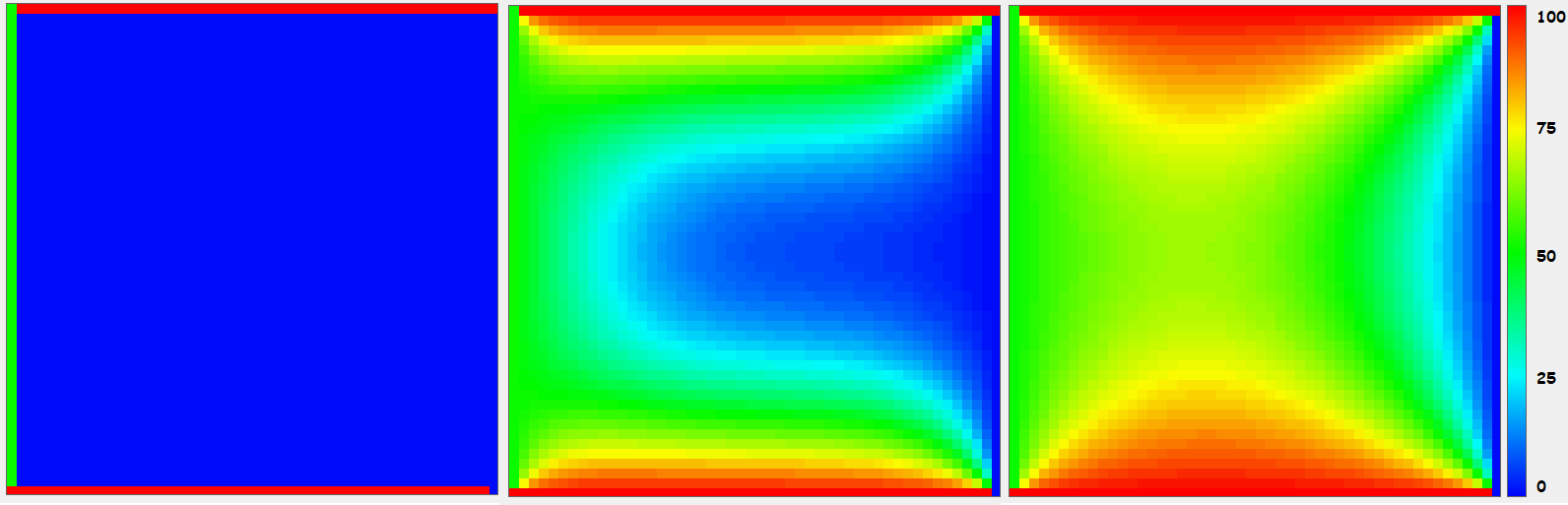


Fig. 2.6: Starea iniţială 4; Un rezultat intermediar; Starea finală 4.

|  |
| --- |
| public static class Engine  {  public static Color[] colors = new Color[101];  public static Form1 form;  public static PictureBox[,] pbMatrix;  public static float[,] matrix, temp;  public static int n, length, state = 0;  public static void Init(Form1 f)  {  int r = 0, g = 0, b = 250;  for (int i = 0; i < 25; i++)  colors[i] = Color.FromArgb(r, g += 10, b);  for (int i = 25; i < 50; i++)  colors[i] = Color.FromArgb(r, g, b -= 10);  for (int i = 50; i < 75; i++)  colors[i] = Color.FromArgb(r += 10, g, b);  for (int i = 75; i < 100; i++)  colors[i] = Color.FromArgb(r, g -= 10, b);  colors[100] = Color.FromArgb(255, 0, 0);  form = f;  n = 50;  length = f.pictureBox1.Width / n;  pbMatrix = new PictureBox[n, n];  matrix = new float[n, n];  temp = new float[n, n];  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  {  pbMatrix[i, j] = new PictureBox();  pbMatrix[i, j].Parent = f.pictureBox1;  pbMatrix[i, j].Size = new Size(length, length);  pbMatrix[i, j].Location = new Point(i \* length, j \* length);  }  InitialValues();  }  public static void InitialValues() {...}  public static void EternalValues(){...}  public static void Step()  {  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  {  float s = 0, nr = 0;  if (i - 1 >= 0)  {  s += matrix[i - 1, j];  nr++;  }  if (j - 1 >= 0)  {  s += matrix[i, j - 1];  nr++;  }  if (j + 1 < n)  {  s += matrix[i, j + 1];  nr++;  }  if (i + 1 < n)  {  s += matrix[i + 1, j];  nr++;  }  temp[i, j] = s / nr;  }  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  {  matrix[i, j] = temp[i, j];  temp[i, j] = 0;  }  EternalValues();  }  public static void Afisare()  {  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  pbMatrix[i, j].BackColor = colors[(int)matrix[i, j]];  }  } |

**3. Prognoza meteo**

Pornind de la aplicaţia precedentă, se poate construi o aplicaţie care are ca şi stare iniţială o hartă văzută din satelit cu temperaturile din prezent. Programul ar avea ca scop prezicerea temperaturilor peste câteva ore, o zi sau chiar şi mai multe zile.

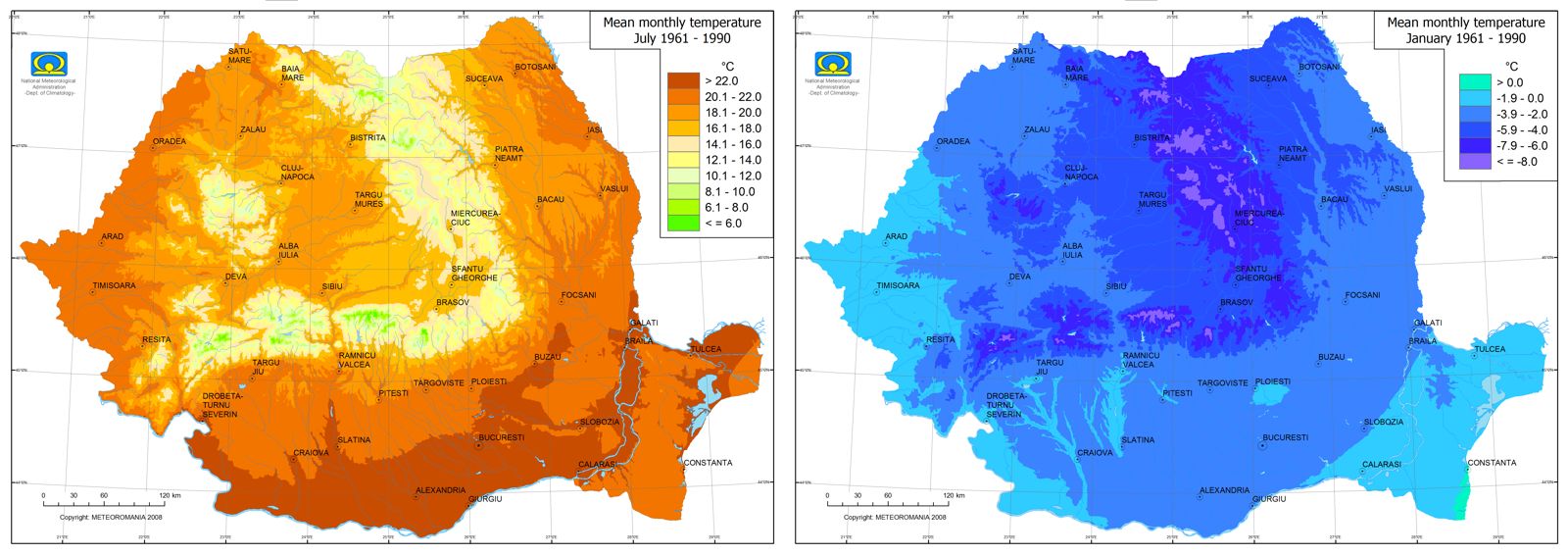


Fig. 3.1: Hărţi ale Romăniei reprezentate cu diferite valori de temperatură

Regula matematică după care se calculează următoarele generaţii va trebui modificată. Aceasta nu va mai depinde doar de temperaturile din jur, ci şi de o multitudine de alţi factori: de la umiditate, precipitaţii şi vânt, până la altitudine, vegetaţie şi tipul de sol. Pe deasupra, sursele fixate de temperatură din programul precedent aici devin surse externe de temperatură, dar care pot fi mobile. Vânturile reci vor scădea temperatura în funcţie de viteza acestora, iar poziţia lor se schimbă repede. Soarele va creşte temperatura doar dacă nu sunt foarte mulţi nori sau, desigur, dacă nu este noapte.

Sunt multe lucruri de care trebuie ţinut cont, dar din moment ce oricum trebuie să determinăm vânturile, umiditatea şi precipitaţiile, mişcările norilor etc., se pot face mai multe astfel de modele: hărţi pentru umiditate şi precipitaţii, pentru mişcările vânturilor, pentru mişcările norilor etc.

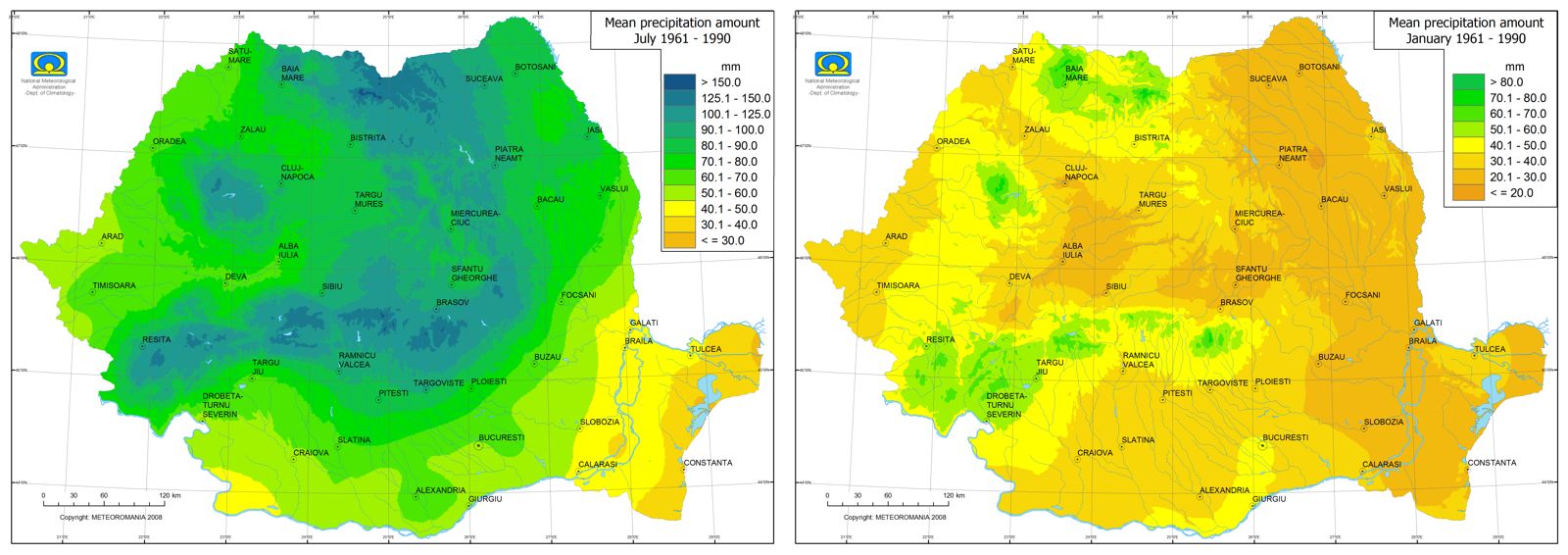


Fig. 3.2: Hărţi ale Romăniei reprezentate cu diferite valori de precipitaţii