//BaseGrid.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameOfLife.Models

{

public abstract class BaseGrid

{

public int Generation { get; set; }

public abstract int Rows { get; }

public abstract int Cols { get; }

public abstract void Step();

public abstract void Clear();

public abstract int CountAlive();

}

}

//GameGrid.cs

using System;

using System.Collections.Generic;

namespace GameOfLife.Models

{

public class GameGrid : BaseGrid

{

public List<List<CellModel>> Cells { get; set; }

public override int Rows => Cells.Count;

public override int Cols => Cells[0].Count;

public GameGrid(int rows, int cols)

{

Cells = new List<List<CellModel>>();

for (int i = 0; i < rows; i++)

{

var row = new List<CellModel>();

for (int j = 0; j < cols; j++)

{

row.Add(new CellModel());

}

Cells.Add(row);

}

}

public override void Step()

{

var newCells = new List<List<CellModel>>();

for (int i = 0; i < Rows; i++)

{

var newRow = new List<CellModel>();

for (int j = 0; j < Cols; j++)

{

int aliveNeighbors = CountAliveNeighbors(i, j);

bool isAlive = Cells[i][j].IsAlive;

var newCell = new CellModel();

if (isAlive)

{

if (aliveNeighbors < 2 || aliveNeighbors > 3)

{

newCell.IsAlive = false;

newCell.Age = 0;

newCell.JustBorn = false;

}

else

{

newCell.IsAlive = true;

newCell.Age = Cells[i][j].Age + 1;

newCell.JustBorn = false;

}

}

else

{

if (aliveNeighbors == 3)

{

newCell.IsAlive = true;

newCell.Age = 1;

newCell.JustBorn = true;

}

else

{

newCell.IsAlive = false;

newCell.Age = 0;

newCell.JustBorn = false;

}

}

newRow.Add(newCell);

}

newCells.Add(newRow);

}

Cells = newCells;

Generation++;

}

public override void Clear()

{

for (int i = 0; i < Rows; i++)

{

for (int j = 0; j < Cols; j++)

{

Cells[i][j] = new CellModel();

}

}

Generation = 0;

}

public override int CountAlive()

{

int count = 0;

foreach (var row in Cells)

foreach (var cell in row)

if (cell.IsAlive) count++;

return count;

}

private int CountAliveNeighbors(int row, int col)

{

int count = 0;

for (int i = row - 1; i <= row + 1; i++)

{

for (int j = col - 1; j <= col + 1; j++)

{

if (i == row && j == col)

continue;

if (i >= 0 && i < Rows && j >= 0 && j < Cols)

if (Cells[i][j].IsAlive)

count++;

}

}

return count;

}

}

}

//CellModel.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameOfLife.Models

{

public class CellModel

{

public bool IsAlive { get; set; }

public int Age { get; set; }

public bool JustBorn { get; set; }

}

}

//Cell.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameOfLife.Models

{

public class Cell

{

public bool IsAlive { get; set; }

public bool JustBorn { get; set; }

public int Age { get; set; }

public Cell()

{

IsAlive = false;

JustBorn = false;

Age = 0;

}

public void Die()

{

IsAlive = false;

JustBorn = false;

Age = 0;

}

public void Revive()

{

IsAlive = true;

JustBorn = true;

Age = 1;

}

public void IncrementAge()

{

if (IsAlive) Age++;

}

}

}

//GameController.cs

using GameOfLife.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Timers;

using System.Windows;

namespace GameOfLife.Controllers

{

public class GameController

{

private Timer \_timer;

private BaseGrid \_grid;

private Queue<string> lastHashes = new Queue<string>();

private HashSet<string> seenHashes = new HashSet<string>();

public BaseGrid Grid => \_grid;

public event Action GridUpdated;

public GameController(int rows, int cols)

{

\_grid = new GameGrid(rows, cols);

\_timer = new Timer(200);

\_timer.Elapsed += (s, e) => StepOnce();

}

public void Start() => \_timer.Start();

public void Stop() => \_timer.Stop();

public void SetTimerInterval(int interval)

{

\_timer.Interval = interval;

}

public void StepOnce()

{

if (\_grid.CountAlive() == 0)

{

Stop();

MessageBox.Show("Усі клітини мертві. Гра завершена.", "Кінець",

MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

\_grid.Step();

GridUpdated?.Invoke();

string currentHash = GetGridHash();

if (lastHashes.Any() && lastHashes.Last() == currentHash)

{

Stop();

MessageBox.Show("Стабільне поєднання клітин. Гра завершена.", "Стабільність",

MessageBoxButton.OK, MessageBoxImage.Information);

lastHashes.Clear();

seenHashes.Clear();

return;

}

lastHashes.Enqueue(currentHash);

if (lastHashes.Count > 5)

lastHashes.Dequeue();

if (seenHashes.Contains(currentHash))

{

Stop();

MessageBox.Show("Виявлено цикл. Гра завершена.", "Цикл",

MessageBoxButton.OK, MessageBoxImage.Information);

lastHashes.Clear();

seenHashes.Clear();

return;

}

seenHashes.Add(currentHash);

}

private string GetGridHash()

{

var gameGrid = \_grid as GameGrid;

if (gameGrid == null) return "";

var sb = new StringBuilder();

for (int i = 0; i < gameGrid.Rows; i++)

for (int j = 0; j < gameGrid.Cols; j++)

sb.Append(gameGrid.Cells[i][j].IsAlive ? $"1{i}, {j};" : "");

return sb.ToString();

}

public void Clear()

{

\_grid.Clear();

lastHashes.Clear();

seenHashes.Clear();

GridUpdated?.Invoke();

}

public void RandomFill()

{

var rand = new Random();

for (int i = 0; i < \_grid.Rows; i++)

{

for (int j = 0; j < \_grid.Cols; j++)

{

var cell = ((\_grid as GameGrid)?.Cells[i][j]);

if (cell != null)

{

cell.IsAlive = rand.Next(2) == 0;

cell.Age = cell.IsAlive ? 1 : 0;

cell.JustBorn = cell.IsAlive;

}

}

}

lastHashes.Clear();

seenHashes.Clear();

GridUpdated?.Invoke();

}

public void SetGrid(BaseGrid newGrid)

{

\_grid = newGrid;

lastHashes.Clear();

seenHashes.Clear();

GridUpdated?.Invoke();

}

}

}

//GridModel.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameOfLife.Models

{

public class GridModel

{

public CellModel[,] Cells { get; set; }

public int Generation { get; set; }

public int Rows => Cells.GetLength(0);

public int Cols => Cells.GetLength(1);

public int CountAlive()

{

int count = 0;

foreach (var cell in Cells)

if (cell.IsAlive) count++;

return count;

}

}

}

//MainWindow.xaml.cs

using GameOfLife.Controllers;

using GameOfLife.Models;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Security.Cryptography;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Timers;

namespace GameOfLife

{

public partial class MainWindow : Window

{

private GameController controller;

private Rectangle[,] cellRects;

private int rows = 30;

private int cols = 30;

private bool isMouseDown = false;

private bool isErasing = false;

private List<string> previousHashes = new();

private int stabilityCounter = 0;

private const int StabilityThreshold = 3;

private int lastAliveCount = -1;

private bool gameWasStopped = false;

public MainWindow()

{

InitializeComponent();

InitializeGame();

SizeChanged += (s, e) => DrawGrid();

if (controller != null)

{

controller.SetTimerInterval((int)SpeedSlider.Value);

}

}

private void InitializeGame()

{

controller = new GameController(rows, cols);

controller.GridUpdated += DrawGrid;

cellRects = new Rectangle[rows, cols];

GameCanvas.MouseLeftButtonDown += Canvas\_MouseLeftButtonDown;

GameCanvas.MouseLeftButtonUp += (s, e) => isMouseDown = false;

GameCanvas.MouseRightButtonDown += (s, e) => { isMouseDown = true; isErasing = true; };

GameCanvas.MouseRightButtonUp += (s, e) => { isMouseDown = false; isErasing = false; };

GameCanvas.MouseMove += Canvas\_MouseMove;

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

var rect = new Rectangle

{

Stroke = Brushes.Gray,

Fill = Brushes.White

};

GameCanvas.Children.Add(rect);

cellRects[i, j] = rect;

}

}

DrawGrid();

}

private void Canvas\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

isMouseDown = true;

isErasing = false;

ModifyCellAtMouse(e);

}

private void Canvas\_MouseMove(object sender, MouseEventArgs e)

{

if (isMouseDown)

{

ModifyCellAtMouse(e);

}

}

private void ModifyCellAtMouse(MouseEventArgs e)

{

Point position = e.GetPosition(GameCanvas);

double cellSize = Math.Min(GameCanvas.ActualWidth / cols,

GameCanvas.ActualHeight / rows);

int j = (int)(position.X / cellSize);

int i = (int)(position.Y / cellSize);

var gameGrid = controller.Grid as GameGrid;

if (gameGrid == null) return;

if (i >= 0 && i < rows && j >= 0 && j < cols)

{

var cell = gameGrid.Cells[i][j];

if (isErasing && cell.IsAlive)

{

cell.IsAlive = false;

cell.Age = 0;

cell.JustBorn = false;

DrawGrid();

}

else if (!isErasing && !cell.IsAlive)

{

cell.IsAlive = true;

cell.Age = 1;

cell.JustBorn = true;

DrawGrid();

}

}

}

private void DrawGrid()

{

try

{

Dispatcher.Invoke(() =>

{

double cellSize = Math.Min(GameCanvas.ActualWidth / cols,

GameCanvas.ActualHeight / rows);

var gameGrid = controller.Grid as GameGrid;

if (gameGrid == null) return;

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

var cell = gameGrid.Cells[i][j];

var rect = cellRects[i, j];

rect.Width = rect.Height = cellSize;

Canvas.SetLeft(rect, j \* cellSize);

Canvas.SetTop(rect, i \* cellSize);

rect.Fill = cell.IsAlive

? (cell.JustBorn ? Brushes.LightGreen : Brushes.Black)

: Brushes.White;

}

}

GenerationText.Text = $"Покоління: {controller.Grid.Generation}";

AliveCountText.Text = $"Живих клітин: {controller.Grid.CountAlive()}";

var hash = GetGridHash(gameGrid);

int currentAlive = controller.Grid.CountAlive();

if (currentAlive == lastAliveCount && previousHashes.Count > 0 &&

previousHashes[^1] == hash)

{

stabilityCounter++;

}

else

{

stabilityCounter = 0;

}

lastAliveCount = currentAlive;

previousHashes.Add(hash);

if (previousHashes.Count > StabilityThreshold)

previousHashes.RemoveAt(0);

});

}

catch (TaskCanceledException)

{

}

catch (Exception ex)

{

MessageBox.Show($"Помилка під час малювання: {ex.Message}");

}

}

private string GetGridHash(GameGrid grid)

{

var bits = new StringBuilder();

foreach (var row in grid.Cells)

foreach (var cell in row)

bits.Append(cell.IsAlive ? '1' : '0');

using var sha = SHA256.Create();

var hashBytes = sha.ComputeHash(Encoding.UTF8.GetBytes(bits.ToString()));

return Convert.ToBase64String(hashBytes);

}

private void StartButton\_Click(object sender, RoutedEventArgs e)

{

if (controller.Grid.CountAlive() == 0)

{

MessageBox.Show("Поле порожнє. Додайте хоча б одну живу клітину, щоб почати гру.", "Увага", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

if (!gameWasStopped)

{

controller.Grid.Generation = 0;

}

else

{

gameWasStopped = false;

}

controller.Start();

}

private void StopButton\_Click(object sender, RoutedEventArgs e)

{

controller.Stop();

gameWasStopped = true;

}

private void StepButton\_Click(object sender, RoutedEventArgs e)

{

controller.StepOnce();

}

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

controller.Clear();

gameWasStopped = false;

}

private void RandomButton\_Click(object sender, RoutedEventArgs e)

{

controller.RandomFill();

gameWasStopped = false;

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

var dialog = new SaveFileDialog { Filter = "JSON Files (\*.json)|\*.json" };

if (dialog.ShowDialog() == true)

{

var json = JsonSerializer.Serialize(controller.Grid as GameGrid);

File.WriteAllText(dialog.FileName, json);

}

}

private void LoadButton\_Click(object sender, RoutedEventArgs e)

{

var dialog = new OpenFileDialog { Filter = "JSON Files (\*.json)|\*.json" };

if (dialog.ShowDialog() == true)

{

var json = File.ReadAllText(dialog.FileName);

var grid = JsonSerializer.Deserialize<GameGrid>(json);

controller.SetGrid(grid);

}

}

private void SpeedSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

if (controller != null)

{

double sliderValue = e.NewValue;

double newInterval = (SpeedSlider.Minimum + SpeedSlider.Maximum) - sliderValue;

controller.SetTimerInterval((int)newInterval);

}

}

}

}

Дизайн

//MainWindow.xaml

<Window x:Class="GameOfLife.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Гра Життя" Height="700" Width="1000"

WindowStartupLocation="CenterScreen"

ResizeMode="NoResize">

<Grid Background="Gray">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="250"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!-- Панель кнопок зліва -->

<StackPanel Grid.Column="0" Margin="10" VerticalAlignment="Top">

<TextBlock Text="Керування" FontWeight="Bold" FontSize="18" Margin="0,0,0,15"/>

<Button Content="▶ Старт" Click="StartButton\_Click" Margin="0,5" Height="35"/>

<Button Content="⏸ Стоп" Click="StopButton\_Click" Margin="0,5" Height="35"/>

<Button Content="⏭ Крок вперед" Click="StepButton\_Click" Margin="0,5" Height="35"/>

<Button Content="🧹 Очистити" Click="ClearButton\_Click" Margin="0,5" Height="35"/>

<Button Content="🎲 Випадково" Click="RandomButton\_Click" Margin="0,5" Height="35"/>

<Button Content="💾 Зберегти" Click="SaveButton\_Click" Margin="0,5" Height="35"/>

<Button Content="📂 Завантажити" Click="LoadButton\_Click" Margin="0,5" Height="35"/>

<TextBlock Text="Швидкість" FontWeight="Bold" FontSize="16" Margin="0,20,0,5"/>

<Slider x:Name="SpeedSlider" Minimum="50" Maximum="500" Value="200" TickPlacement="BottomRight" TickFrequency="50" IsSnapToTickEnabled="True" ValueChanged="SpeedSlider\_ValueChanged"/>

<TextBlock x:Name="CurrentSpeedText" FontSize="12" Margin="0,5,0,0" Text="{Binding ElementName=SpeedSlider, Path=Value, StringFormat='{}Поточна швидкість: {0} мс'}"/>

<!-- Статистика -->

<StackPanel Margin="0,20,0,0">

<TextBlock Text="Статистика" FontWeight="Bold" FontSize="16" Margin="0,0,0,10"/>

<TextBlock x:Name="GenerationText" FontSize="14" Margin="0,2"/>

<TextBlock x:Name="AliveCountText" FontSize="14" Margin="0,2"/>

</StackPanel>

</StackPanel>

<!-- Поле гри -->

<Canvas x:Name="GameCanvas" Grid.Column="1" Background="Gray" Margin="10"/>

</Grid>

</Window>