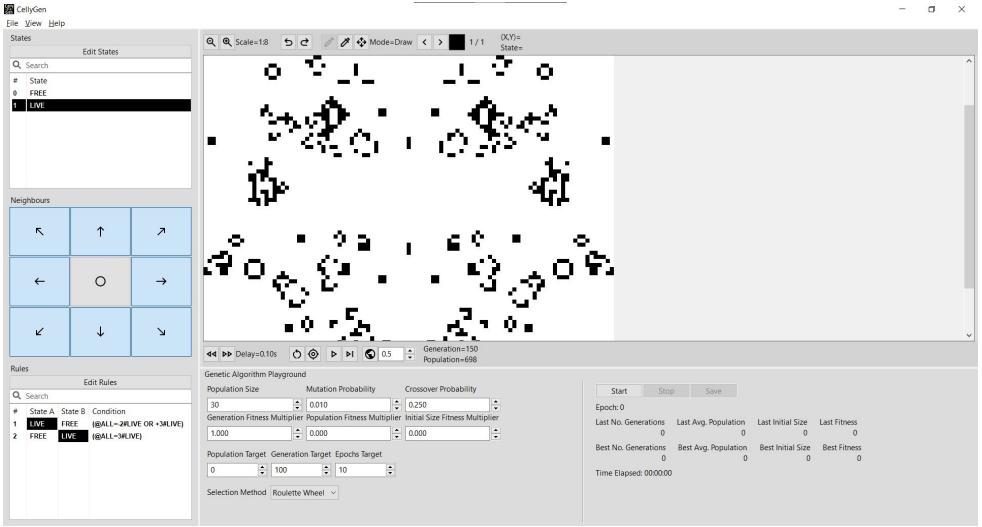
CellyGen - aplicație pentru explorarea automatelor celulare folosind algoritmi genetici

Absolvent: **Bita Mihai-Alexandru**
Coordonator Științific: Asist. dr. **Croitoru Eugen**

Cuprins

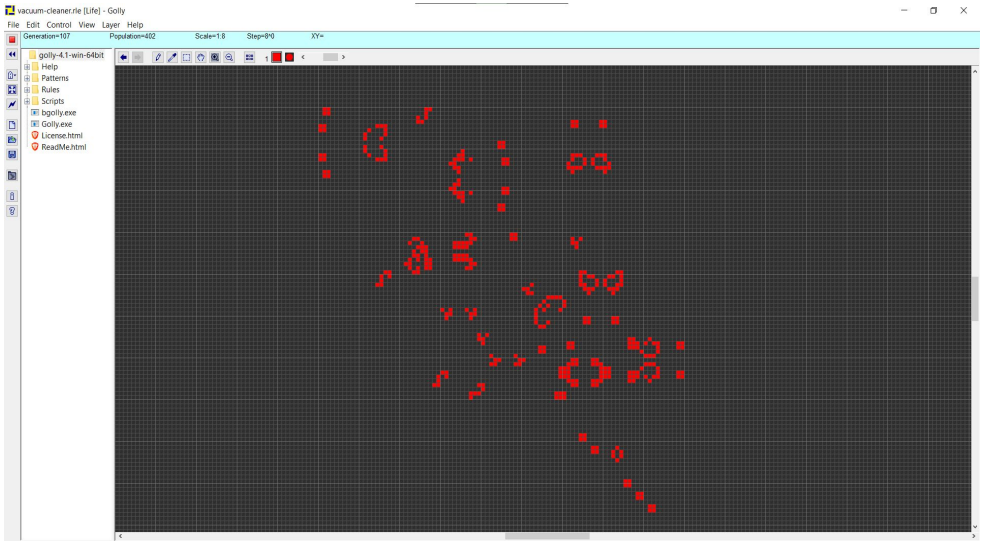
Introducere: Obiective principale

- simularea automatelor celulare
- explorarea folosind algoritmi genetici
- ușor de utilizat



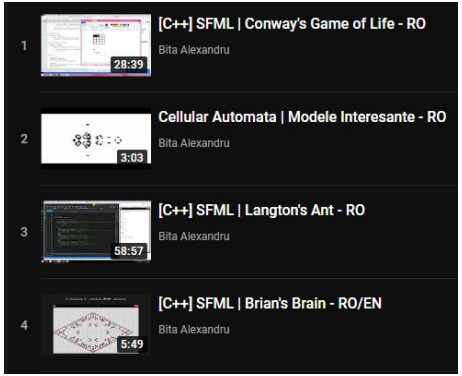
Introducere: Aplicații similare

- Golly



Introducere: Motivație

- facilitarea lucrului cu automate celulare
- interesul pentru automate celulare și programarea genetică
- descoperirea unor configurații si structuri interesante

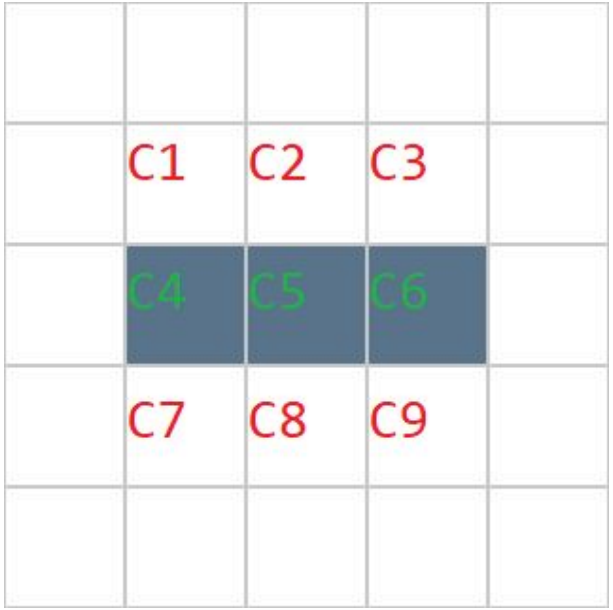


Fundamentare teoretică: Automatele celulare

- Univers
- Celule
- Stări
- Reguli
- Vecinătate

Fundamentare teoretică: Automatele celulare

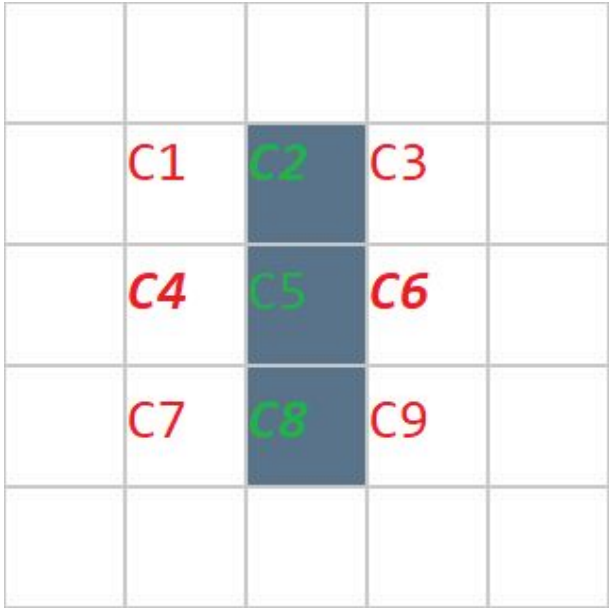
- Univers = matrice M×N
- Celule = elementele matricei
- Stări = {POPULAT, NEPOPULAT}
- Reguli
  a. o celulă POPULATĂ devine NEPOPULATĂ dacă vecinătatea sa are mai puțin de 2 celule POPULATE sau mai mult de 3 celule POPULATE
  b. o celulă NEPOPULATĂ devine POPULATĂ dacă vecinătatea sa are exact 3 celule POPULATE
- Vecinătate = vecinătatea Moore



Exemplu: „Oscilator” în *Conway's Game of Life*

Fundamentare teoretică: Automatele celulare

- Univers = matrice M×N
- Celule = elementele matricei
- Stări = {POPULAT, NEPOPULAT}
- Reguli
  a. o celulă POPULATĂ devine NEPOPULATĂ dacă vecinătatea sa are mai puțin de 2 celule POPULATE sau mai mult de 3 celule POPULATE
  b. o celulă NEPOPULATĂ devine POPULATĂ dacă vecinătatea sa are exact 3 celule POPULATE
- Vecinătate = vecinătatea Moore



Exemplu: „Oscilator” în *Conway’s Game of Life*

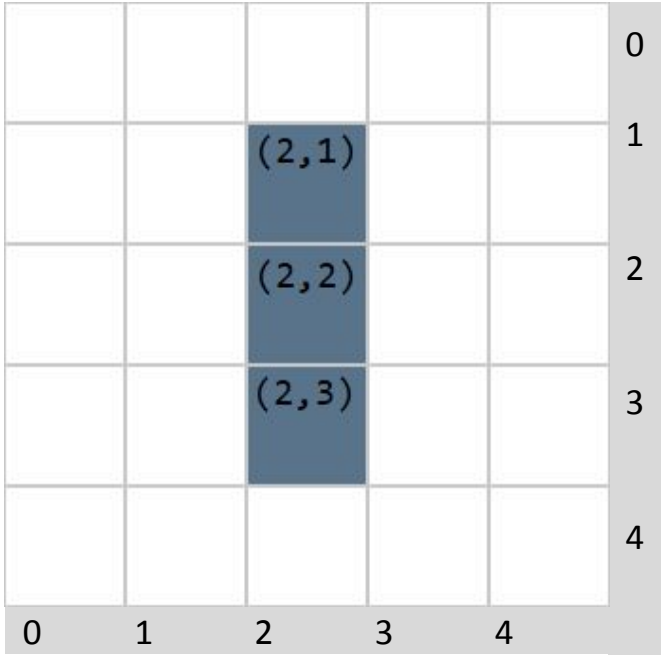
Fundamentare teoretică: Automatele celulare

- Implementare

```
stări: [
    stare: { “nume”: “culoare” }
]
ex: { {“NEPOPULAT”: “ALB”}, {“POPULAT”: “GRI”} }

celule: [
    celulă: { (coloană,linie):  stare }
]
ex: { {(2,1): {POPULAT: GRI}}, {(2,2): {POPULAT: GRI}},
{(2,3): {POPULAT: GRI}} }

grupări: [
   grupare: { stare: [(coloană,linie)] }
]
ex: { {POPULAT: {(2,1), (2,2), (2,3)}} }
```



Fundamentare teoretică: Automatele celulare

```
vecini: [
   vecin: "direcție"
]
ex: { "NW", "N", "NE", "W", "E", "SW", "S", "SE"}

reguli: [
    regulă: {
        stareActuală: stare,
        tranziție: {
            stareUrmătoare: stare,
            vecinătateEvaluată: vecini,
            condiții: [
                condiție: {
                    stareEvaluată: stare,
                    (numărDeCelule, tipDeComparație)
                }
            ]
        }
    }
]
```

```
1 POPULAT / NEPOPULAT :
2       (@ALL = -2#POPULAT or +3#POPULAT);
3
4 NEPOPULAT / POPULAT :
5       (@ALL = 3#POPULAT);
```

Fundamentare teoretică: Algoritmii genetici

- Soluții candidat: indivizi (cromozomi)
- Operatori genetici: selecție, crossover, mutație
- Funcția de evaluare: fitness-ul unui cromozom

Fundamentare teoretică: Algoritmii genetici

- Selecție: *Wheel of Fortune*, *Rank*, *Tournament*, *Steady State*, *Elitism*, *Random*
- Crossover: 2 puncte de tăiere
- Mutație
- Fitness: în funcție de prioritățile setate de utilizator

```
cromozom: {
    configurațieInițială: [int],
    celule: celule,
    grupări: grupări,

    nrGenerații: int,
    mediaPopulației: int,
    mărimeaInițială: int,

    fitness: double
}
ex:
{“ALB” = 0, “ROȘU” = 1, “VERDE” = 2}

c1 = [ [0, 1, 1],   → [0, 1, 1, 1, 2, 0, 2, 0, 1]
       [1, 2, 0],
       [2, 0, 1] ]
```

Demonstrații

- Definirea unui automat celular
- Rularea unui automat celular
- Configurarea algoritmului genetic
- Rularea algoritmului genetic

Prezentare video

Tehnologii utilizate

- C++
- *wxWidgets*

Concluzii

- Obiective atinse
  - simplu de utilizat
  - instrument de explorare bazat pe algoritmi genetici
- Optimizări
  - structura de reguli / *caching*
  - OpenGL
- Funcționalități:
  - explorare de reguli
  - vecinătăți cu distanța Manhattan $r > 1$