УДК 004.021+51.8

Д.С.Федоряка

МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ ГРИ «ЖИТТЯ» ТА ЇЇ МОДИФІКАЦІЙ НА ТРИКУТНИХ, КВАДРАТНИХ ТА ШЕСТИКУТНИХ ҐРАТКАХ

Розв'язана задача: розроблено алгоритми та програму аналізу еволюції колонії фішок на двовимірних гратках із правильних трикутників і шестикутників при різних початкових конфігураціях та різних умовах виживання фішок.

Ключові слова: комп'ютерне моделювання, фішки, колонії фішок, популяція фішок**,** еволюція, плоскі гратки, правильні трикутники й шестикутники, початкові конфігурації та умови виживання фішок.

Решена задача: разработаны алгоритмы и программа анализа эволюции колонии фишек на двумерных решётках из правильных треугольников и шестиугольников при разных начальных конфигурациях и разных условиях выживания фишек.

Ключевые слова: компьютерное моделирование, фишки, колонии фишек, популяция фишек, эволюция, плоские решётки, правильные треугольники и шестиугольники, начальные конфигурации и условия выживания фишек.

A problem is solved: algorithms and program are created for analysis of counters colonies evolution on two-dimension lattices filled with rectilinear triangles and hexagons under different initial configurations and survival conditions.

Keywords: computer simulation, counters, counters colonies, counters population, evolution, two-dimension lattices, rectilinear triangles and hexagons, counters initial configurations and survival conditions.

Вступ

Мета роботи – створити програму для моделювання клітинних автоматів та виконати з її допомогою деякі дослідження. Ідейною основою роботи є гра «Життя», розроблена американським математиком Джоном Конвеєм (*John Horton Conway*). Опис цієї гри вперше опубліковано 1970 року в жовтневому випуску журналу *Scientific American*, в рубриці «Математичні ігри» Мартіна Гарднера (*Martin Gardner*).

Вона є результатом серйозних досліджень в галузі штучного інтелекту та клітинних автоматів, розпочатих ще в 1940-х роках Джоном фон Нейманом (*John von Neumann*). Він намагався створити гіпотетичну машину, яка може відтворювати сама себе. Йому вдалося створити математичну модель такої машини з дуже складними правилами. Конвей намагався спростити ідеї, запропоновані Нейманом і створив правила гри «Життя».

Визначимо основні терміни, якими будемо користуватися.

Ігрове поле – прямокутник (принаймні, для квадратних комірок), розділений на квадратні комірки; розмір кожної комірки набагато менший від розмірів поля.

Фішки – аналоги живих організмів у комірках. Кожна фішка займає окрему комірку. Фішки можуть: 1) жити протягом певного числа ходів; 2) жити необмежено довго; 3) періодично змінювати свій стан між життям та смертю (інших станів у фішки немає).

Колонія фішок – група фішок, кожна з яких має хоча б одне спільні ребро або хоча б одну спільну вершину хоча б із одною іншою фішкою даної колонії.

Конфігурація колонії – взаємне розташування фішок у ній.

Хід – зміна або збереження стану фішки в даній комірці при переході до наступного стану колонії фішок. Цей стан визначається правилами гри, які задають умови виживання окремих фішок, та конфігураціями колоній.

Популяція фішок – сукупність усіх фішок на ігровому полі.

ЧАСТИНА 1. ГРА «ЖИТТЯ»

Правила гри.

Основна ідея гри «Життя» полягає в тому, щоб, почавши з якого-небудь простого розташування фішок ,розставлених по різних клітках дошки, прослідкувати за еволюцією від вихідної позиції під дією "генетичних законів" Конвея, які управляють народженням, загибеллю і виживанням фішок. Конвей ретельно підбирав свої правила і довго перевіряв їх "на практиці", добиваючись, щоб вони задовольняли таким умовам: 1) не повинно бути жодної початкової конфігурації, для якої існував би простий доказ можливості необмеженого зростання популяції; 2) повинні існувати такі початкові конфігурації, які явно володіють здатністю безмежно розвиватися; 3) не повинно бути великої кількості конфігурацій, що швидко зникають; 4) мають існувати прості початкові конфігурації, які протягом значного проміжку часу ростуть, зазнають різноманітні зміни і закінчують свою еволюцію одним з наступних трьох способів: а) повністю зникають (або через перенаселеність, тобто дуже великої густини фішок, або, навпаки, через розрідженість фішок, що утворюють конфігурацію); б) переходять в стійку конфігурацію і перестають змінюватися взагалі; 3) виходять на коливальний режим, при якому вони проходять нескінченний цикл перетворень з певним періодом.

Інакше кажучи, правила гри повинні бути такими, щоб поведінка популяції була достатньо цікавою, а головне, непередбачуваною.

Гра відбувається на полі з квадратних комірок, яке вважається нескінченним (на практиці поле робиться достатньо великим, щоб колонія при еволюції не доходила до краю). Кожну клітину поля оточують вісім сусідніх клітин: чотири мають з нею загальні сторони, а чотири інші - загальні вершини. Правила гри (генетичні закони) зводяться до наступного:

1. **Виживання.** Кожна фішка, у якої є два або три сусіди, виживає й переходить до наступного покоління;

2. **Загибель.** Кожна фішка, біля якої опиняється більше від трьох сусідів, гине через перенаселеність. Кожна фішка, навколо якої вільні всі сусідні клітки або є жива тільки одна фішка, гине від самотності;

3. **Народження.** Якщо число фішок, з якими межує яка-небудь порожня комірка, в точності дорівнює трьом (не більше і не менше), то в цій комірці відбувається народження нової фішки.

Формулювання задач роботи.

У моїй програмі є можливість моделювати описані вище популяції на плоских та тороподібних ґратках (в останньому випадку сусідами комірок, розташованих вздовж межі ігрового поля, є комірки, розташовані вздовж протилежної межі поля).

, щільно заповнених трикутниками, квадратами або шестикутниками. Створено інструменти для дослідження і є широкі можливості змін правил гри. Програма призначена для дослідження популяцій фішок, які еволюціонують за конкретних умов, а також дослідження особливостей цих популяцій в залежності від встановлених правил.

Проведено багато досліджень еволюції подібних популяцій, особливо - гри «Життя». Але про трикутні та гексагональні інтерпретації цієї гри відомо дуже мало. В мережі Internet мені не вдалося знайти програм, які їх моделюють.

Комп’ютерна реалізація

Генетичні правила необхідно застосувати для всіх фішок на кожному ході, причому при підрахунку сусідів беруться до уваги фішки, які існували перед даним ходом. Так утворюється нове покоління популяції.

Найпростіший алгоритм послідовно переглядає всі комірки ґратки та для кожної з них підраховує кількість сусідів, визначаючи долю кожної фішки (не зміниться, помре або народиться). Такий алгоритм використовує два двовимірних масиви — один для поточного покоління, другий — для наступного. Для збільшення швидкості можна на кожному ході складати список фішок для перегляду в наступному поколінні. Клітини, які не можуть змінитися, тобто такі, які не змінились самі і в яких не змінився жоден із сусідів, в списки не вносяться.

Найцікавіші конфігурації гри «Життя»

В класичній грі «Життя» було проведено багато досліджень та виявлено багато цікавих конфігурацій. Деякі з я відтворив та перевірив в своїй програмі, вони наведені нижче.

Спочатку розглянемо найпростіші стійкі конфігурації. В них не помирає і не народжується жодна фішка, тому їх називають «аматорами спокійного життя» (рис. 1а):

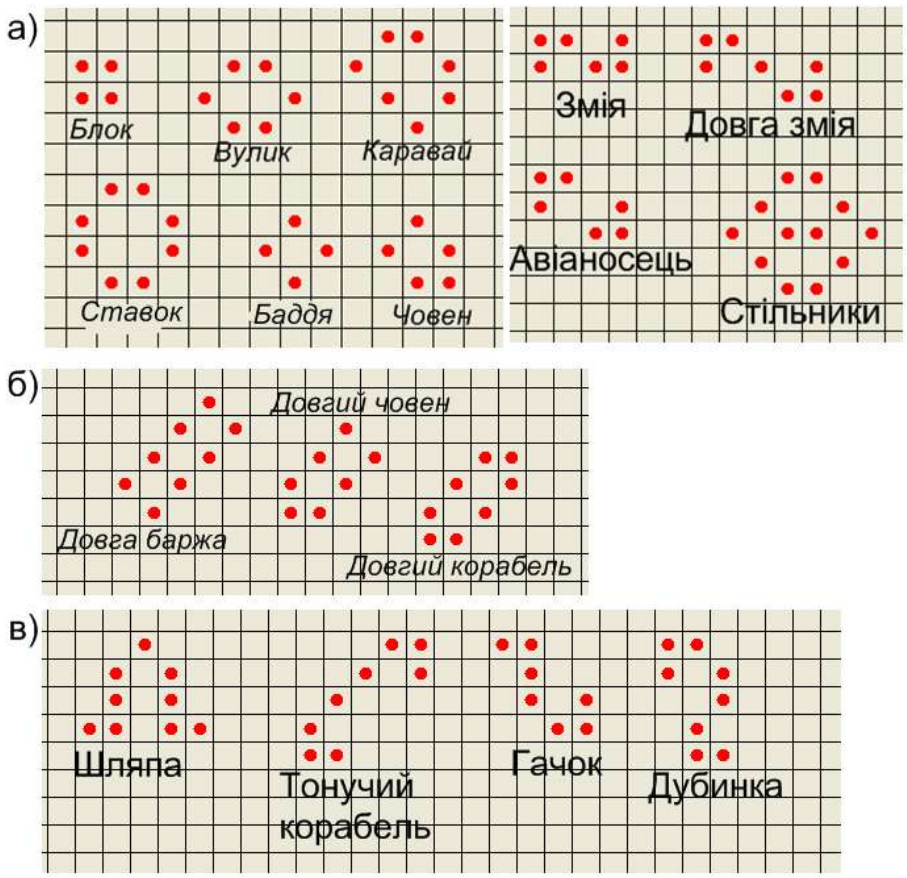

Рис. 1. Приклади стійких конфігурацій у грі «Життя»

Конфігурації типу «Корабель», «Човен» та «Баржа» можна розширювати безмежно (рис. 1б). Взагалі, стійкі конфігурації утворювати нескладно – стійким буде будь-який ланцюжок з фішок, кожна з яких має 2-3 сусіди, який не допускає народження нових фішок (рис. 1в).

Серед цих конфігурацій особливо виділяється «гачок». По-перше, це найменша стійка конфігурація, що не має жодної симетрії. По-друге, він здатний знищувати конфігурації (в тому числі рухомі), що знаходяться справа знизу від нього, за що його ще називають «поглиначем».

Зараз відомі всі стійкі конфігурації, що мають у складі до 13 клітин (див. наступну таблицю 1).

Таблиця 1. Приклади стійких конфігурацій фішок у грі «Життя».

Кількість клітин	Кількість конфігурацій	Приклади
1-3	Жодної	
4	2	«Блок», «Баддя»
5	1	«Човен»
6	5	«Вулик», «Баржа», «Корабель», «Змія», «Авіаносець»

7	4	«Каравай», «Гачок», «Довгий човен», «Довга змія»
8	9	«Ставок», «Довга баржа», «Дубинка», «Тонучий корабель»
9	10	«Шляпа»
10	25	
11	46	
12	121	«Стільники»
13	149	

Комбінуючи різні конфігурації фішок, можна отримати цікаві композиції:

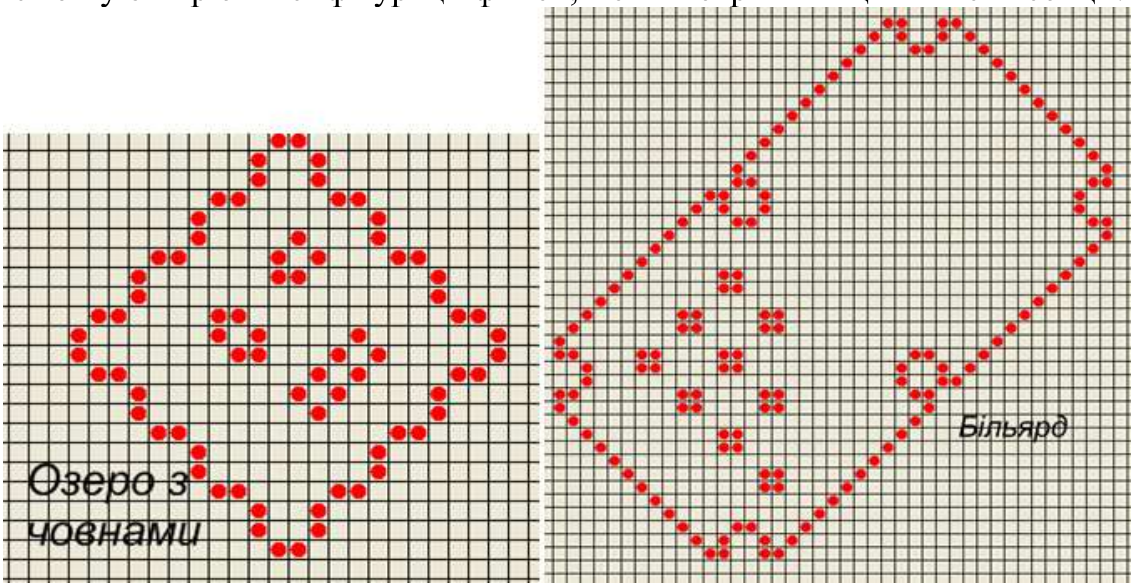

Рис. 2. Комбінації стійких конфігурацій у грі «Життя».

Далі, розглянемо періодичні конфігурації, тобто такі, що відтворюють себе з періодом в кілька поколінь. Найпростіші з них, так звані «фліп-флопи», мають період, рівний двом:

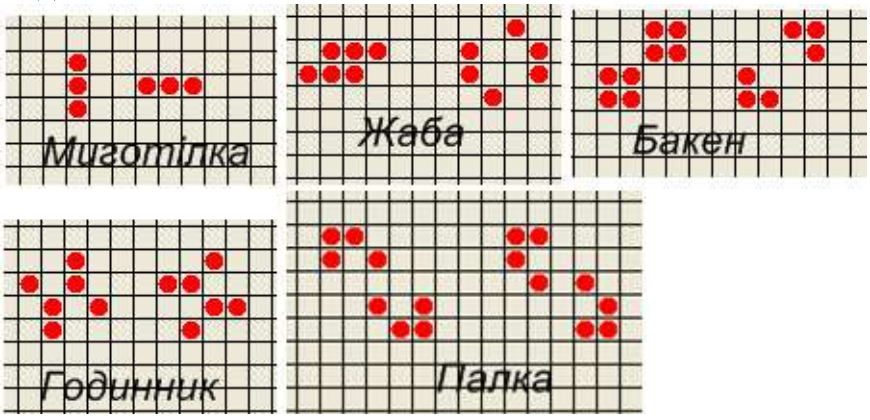

Рис. 3. Конфігурації, які періодично повторюються

Всі вони, крім «бакену», зберігають кількість живих клітин. Найпростіша з них – «миготілка», часто утворюється під час еволюції випадкових колоній. Особливою є конфігурація «палка», бо її можна нескінченно розтягувати і кожний з її двох станів є дзеркальним відображенням іншого:

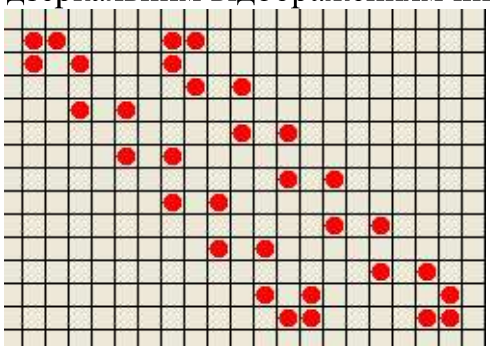

Рис. 4. Періодична конфігурація, здатна до розтягування

Розглянемо періодичні конфігурації з більшим періодом:

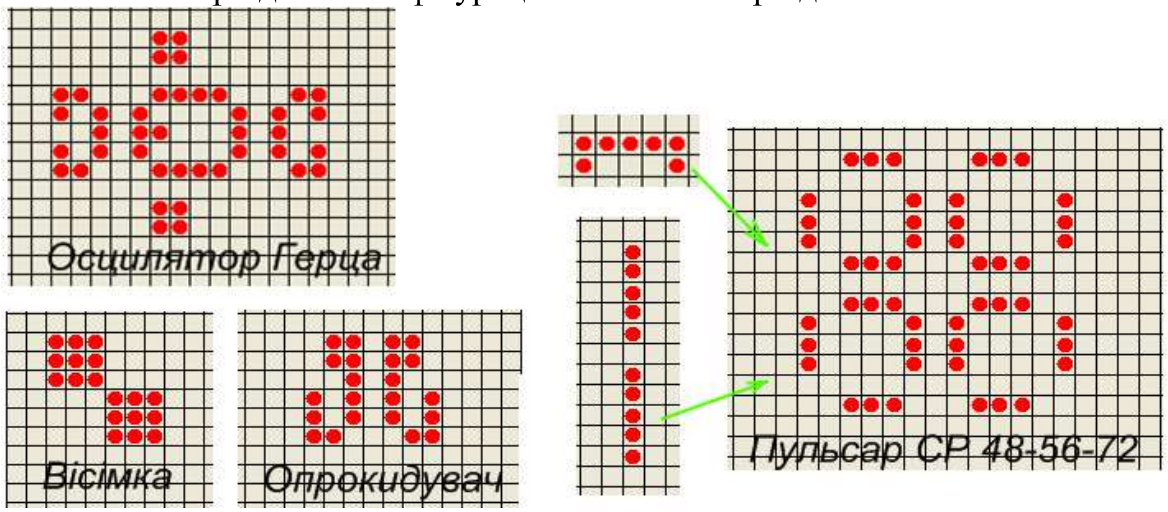

Рис. 5. Конфігурації, що повторюються з порівняно великим періодом

«Опрокидувач» має 22 клітини та кожні сім ходів перевертається. Його період складає 14. В «осциляторі Герца» змінюється лише внутрішня область: кожні чотири ходи фішка всередині прямокутника переходить до краю, а розмірпопуляції коливається між 41 і 44. «Вісімка» змінює популяцію від 12 до 26 і коливається з періодом вісім. «Пульсар CP 48-56-72» утворюється при еволюції ряду з 11 клітин або π-гептаміно і має період три.

Однією з найцікавіших конфігурацій в грі «життя» є «глайдер» (glider). Вона складається з п’яти фішок і через кожні чотири ходи повторює себе, зміщуючись на одну комірку вниз і праворуч:

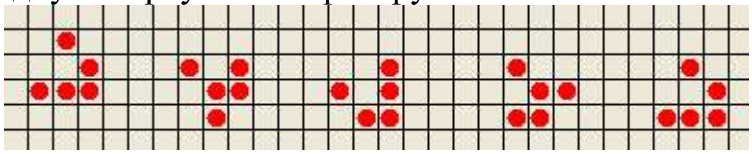

Рис. 6. «Глайдер» - конфігурація, що еволюціонує та повторюється у процесі руху

Таку ж властивість до самовідтворення з пересуванням мають «космічні кораблі»:

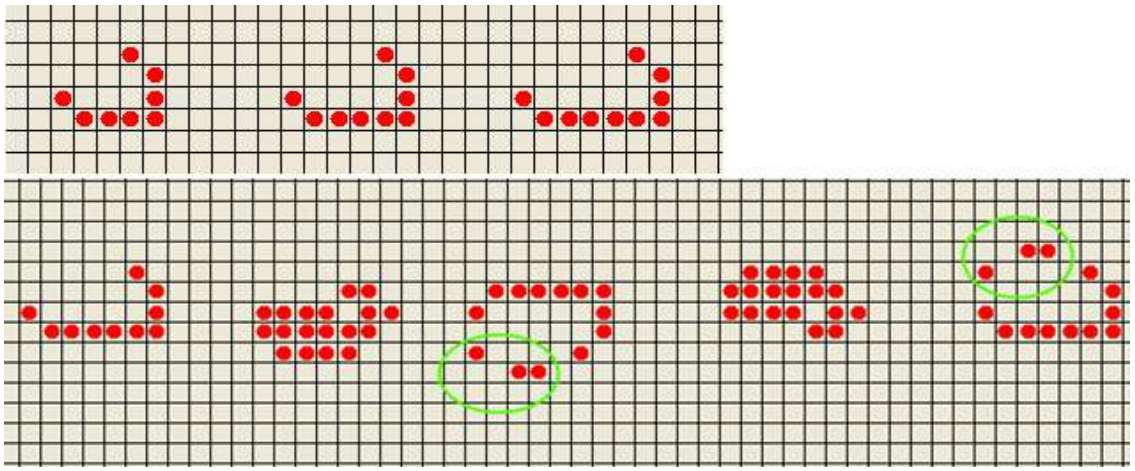

Рис. 7. Конфігурації, що еволюціонують та повторюються у процесі руху – «космічні кораблі»

Такі кораблі можна видовжувати до нескінченності, але якщо в горизонтальній лінії («корпусі») буде більше шести фішок - «іскри» руйнуватимуть корабель. Щоб цього не сталося, поряд із кораблем розміщують ескорт із менших кораблів.

Дослідниками було знайдено багато інших «космічних кораблів», але вони, як правило, дуже великі й мають великий період. Ось приклади найменших із них:

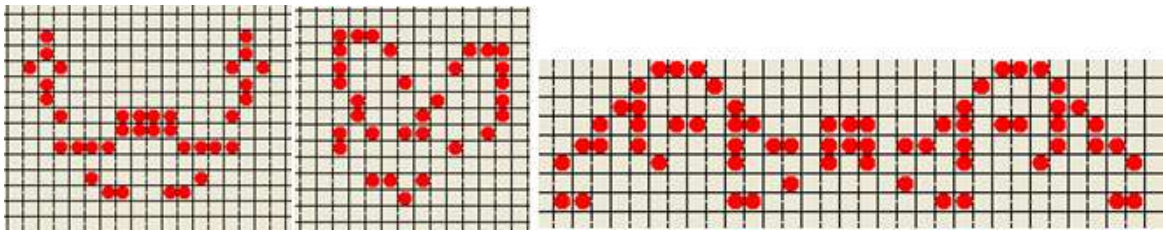

Рис. 8. Приклади конфігурацій типу «космічні кораблі»

Далі розглянемо «гармати» (guns). Ці конфігурації періодично повторюються з виникненням рухомих фігур. Першу з них відкрив математик Госпер (*Bill Gosper),* довівши існування конфігурацій з безмежним ростом популяції, за що отримав премію в $50 від самого Конвея, який заперечував можливість їх існування. За 30 ходів ця конфігурація повертається до початкового вигляду, при цьому утворюється «глайдер»:

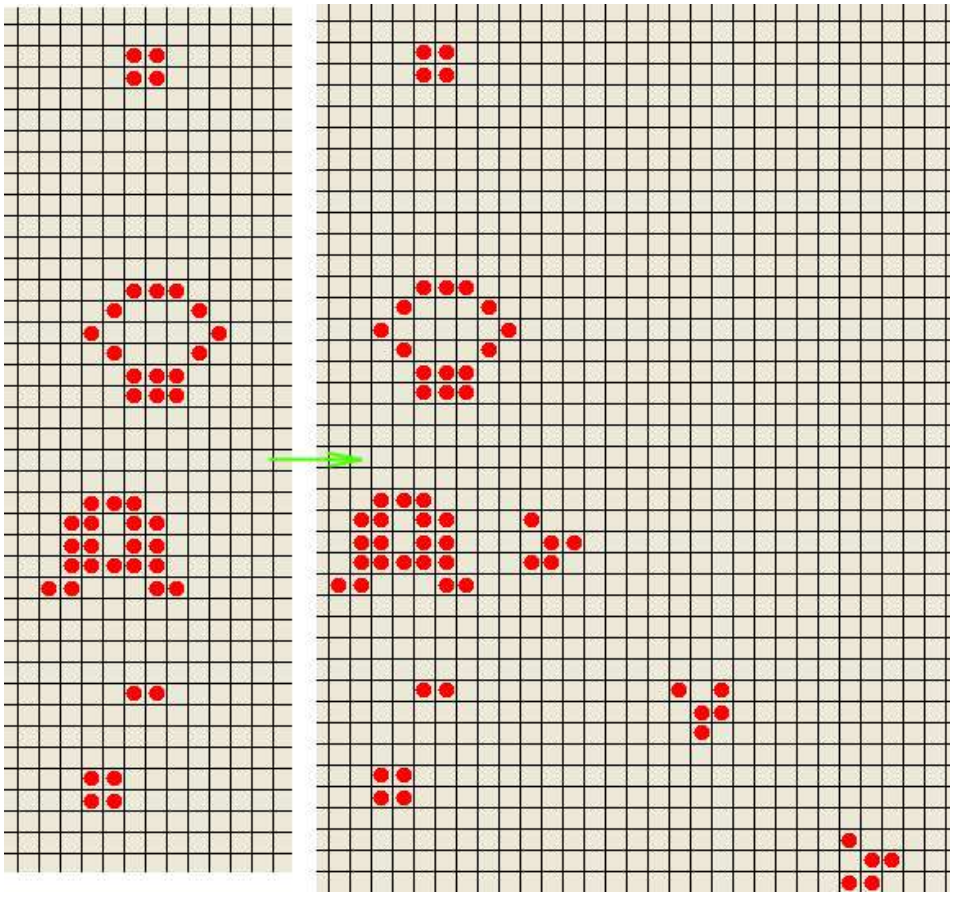

Рис. 9. Конфігурація «Гармата»

Крім того, в грі «Життя» існує багато інших типів конфігурацій.

Відбивачі (reflectors) – конфігурації, при зіткненні з якими рухомі конфігурації змінюють напрям

Паровози – рухомі фігури, що залишають за собою слід зі стійких та періодичних фігур

Паразити – фігури, що дублюються при зіткненні з іншими фігурами.

Фітилі – паровози, що рухаються по умовно нескінченним ланцюжкам клітин.

Сади Едему – конфігурації, що не можуть мати попередників.

Еволюція колоній у грі «Життя»

В класичній грі «Життя» колонії в залежності від початкової конфігурації можуть розвиватися по-різному: 1) усі фішки можуть вмерти (більшість конфігурацій тяжіє до повільного зменшення популяції, але вимирають зазвичай дуже густонаселені конфігурації; 2) після деякого кроку колонія перестає змінюватися (перетворюється на «аматора спокійного життя»); 3) конфігурації починають повторюватися з певним періодом; 4) популяція (кількість фішок) безмежно росте (це можливо за наявності «гармат» або «паровозів»).

Найчастіше еволюція великих колоній завершується розпадом на невеликі стійкі («блоки», «вулики», «човни») та періодичні («миготілки») конфігурації.

Часто утворюються «глайдери», які можуть зруйнуватися або «втекти» за межі поля.

Існують невеликі конфігурації, що мають складну й довгу еволюцію. Їх називають «довгожителями». Найменша така конфігурація – це r-пентаміно (рис. 10). Її еволюція закінчується лише через 1103 ходи з утворенням шести «глайдерів», чотирьох «миготілок» та дванадцяти стійких конфігурацій.

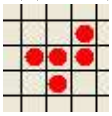

Рис. 10. Конфігурація «r-пентаміно»

За допомогою моєї програми було досліджено залежність концентрації фішок у популяції, що утворюється на кінцевій стадії еволюції, від початкової концентрації фішок у класичній грі «Життя» (тут і далі під концентрацією розуміється відношення кількості фішок до загальної кількості комірок на ігровому полі). Дослідження проводилися на полі 20Х20. Для звичайної моделі був отриманий графік:

Рис. 11. Графік залежності кінцевої концентрації від початкової в класичній грі "Життя"

Для моделі поверхні тора був отриманий графік:

Рис. 12. Графік залежності кінцевої концентрації від початкової в класичній грі "Життя" (модель тора)

З цих графіків бачимо, що при початкових концентраціях не більше від 5% і більше від 80% популяція вимирає, а в при концентрації від 10% до 70% популяція переходить у стійкі та періодичні конфігурації з густиною 4-6%. В цих межах кінцева концентрація майже не залежить від початкової.

Детальний опис основних результатів, одержаних за правилами Дж. Конвея виправданий тим, що одержані результати доведеться з ними порівнювати.

ЧАСТИНА 2. ОПИС СТВОРЕНОЇ ПРОГРАМИ

Головною частиною курсової роботи є програма (Windows - додаток) “**LifeGame.exe**”. Вона створена в середовищі **Visual Studio 2008** на мові програмування **Visual Basic**. Вона складається з трьох форм:

Form1 – головне вікно для моделювання та налаштувань

Form2 – для побудови графіку концентрацій

AboutBox1 – для виведення інформації про програму

Загальний вигляд програми

Головне вікно програми має наступний вигляд:

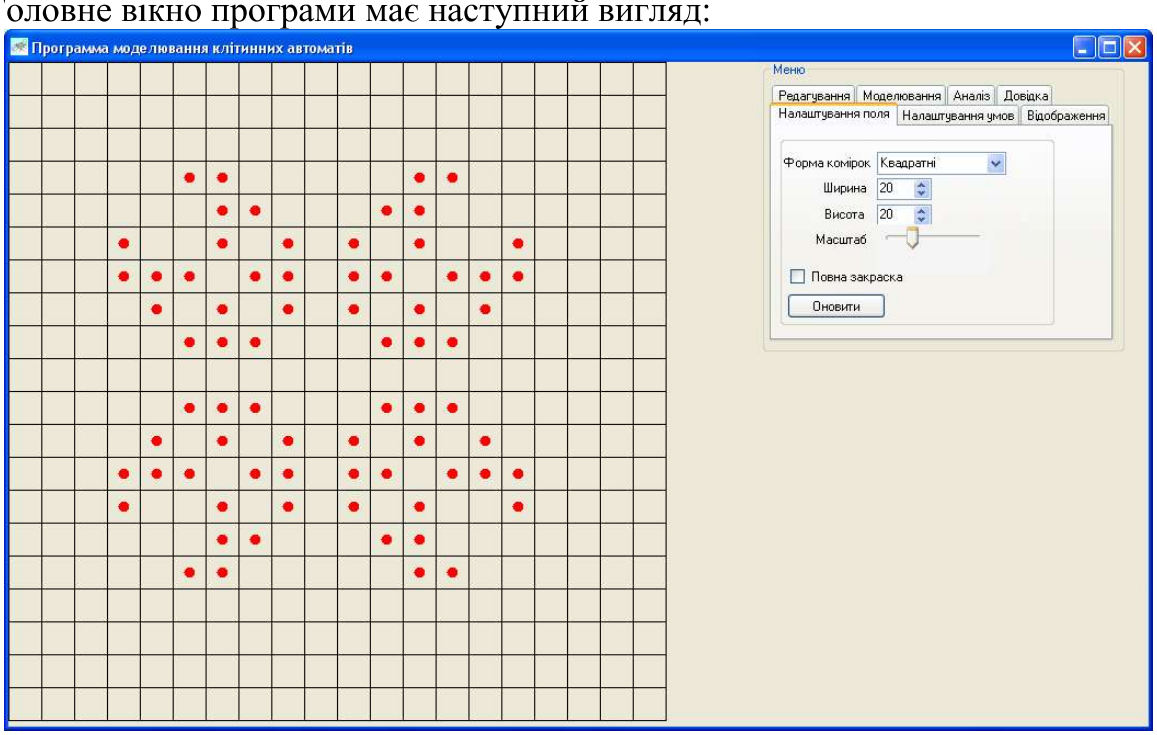

Рис. 13. Головне вікно програми

Воно складається власне з поля, яке цілком промальовується програмно, та з головного меню, яке розміщене в компоненті **TabControl**. Меню складається з наступних пунктів:

Налаштування поля. Зміна типу, розмірів та масштабу ґратки

Налаштування умов. Зміна генетичних законів

Відображення. Зміна кольорів

Редагування. Доступ до можливостей редагування

Моделювання. Керування моделюванням

Аналіз. Доступ до можливостей аналізу

Довідка. Перегляд інформації про програму

Меню можна робити невидимим за допомогою клавіші **F8**. В інтерфейсі використовується українська мова.

Загальна модель

Хоча були проведені деякі дослідження щодо нестандартних ґраток гри «Життя», чітко визначених правил для них не існує. Тому нижче наведено

модель, що використовується в моїй програмі: 1) моделювання відбувається на площині, щільно заповненій комірками з правильних многокутників (трикутників, квадратів чи шестикутників); 2) кожна комірка може містити фішку; 3) кожна фішка має сусідів – це фішки, що мають з нею спільні сторону або вершину; 4) всі зміни відбуваються відповідно до генетичних законів, аналогічних законам класичної гри «Життя».

Для опису життєвого простору фішок використано двовимірний масив **a(200,200)** типу Boolean. Він є статичним, але при моделюванні використовується лише його частина (m,n). **m** та **n** – це розміри поля, які задаються користувачем в інтервалі від 1 до 199. Таким чином, найбільше число клітин може бути 39601. Значення **True** означає фішку в даній комірці, значення **False** – порожню комірку. Кожній комірці відповідає певний елемент масиву.

У комп'ютерній програмі неможливо змоделювати нескінченне поле, але його можна зробити достатньо великим, щоб межі не заважали моделюванню. До того ж, у програмі існує режим моделювання на торі.

Ключовим в програмі є процес переходу від опису комірок у масиві до їх розташування на гратці або торі. На квадратній ґратці все дуже просто – елемент **a(i,j)** відповідає **i**-й клітині **j**-го зверху рядку.

Для шестикутних ґраток використовується наступна модель:

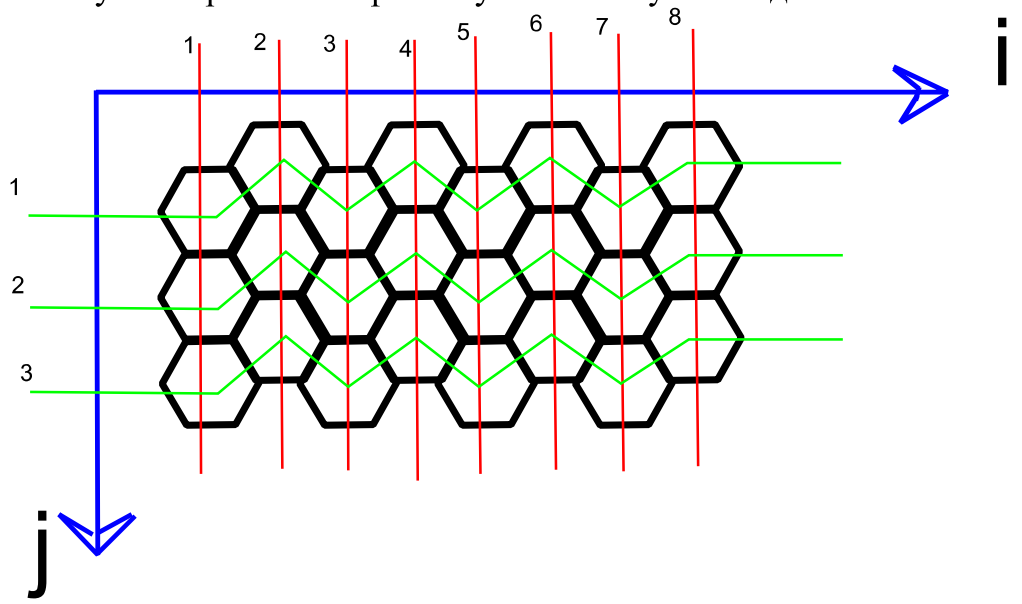

Рис. 14. Модель представлення колонії в масиві для шестикутної ґратки

Для трикутних ґраток використовується наступна модель:

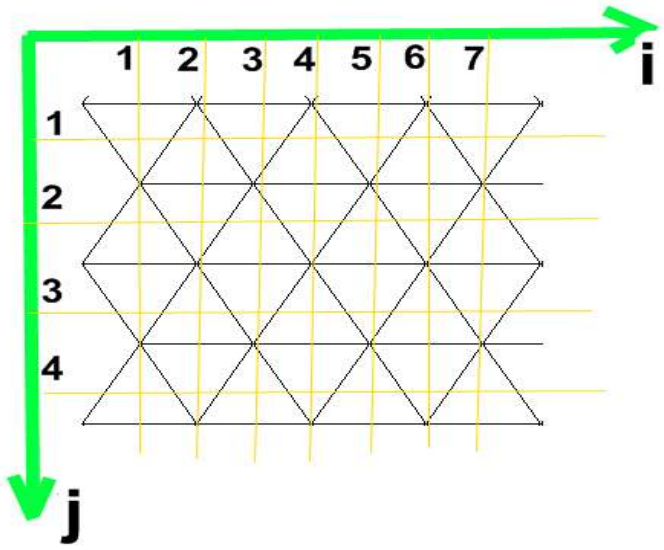

Рис. 15. Модель представлення колонії в масиві для шестикутної ґратки

Вважається, що фігури, що мають спільну сторону чи вершину, є сусідами. Таким чином, кожен шестикутник має по 6 сусідів, а трикутник – по 12:

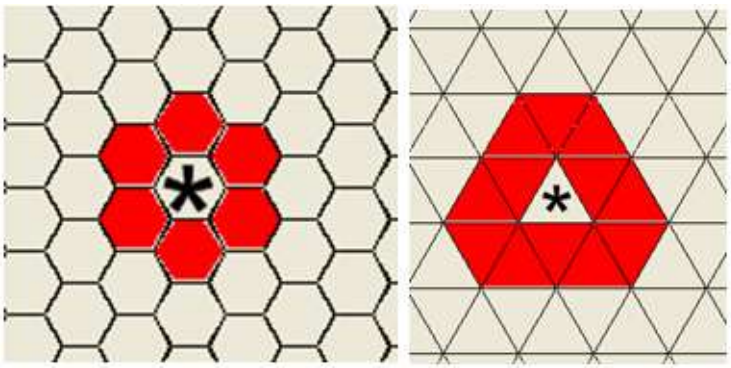

Рис. 16. Кількості найближчих сусідів у трикутної та шестикутної комірок

Крім визначення сусідів, моделювання еволюції популяції на трикутниках чи шестикутниках цілком аналогічне моделюванню на квадратах.

Глобальна змінна **type,** приймаючи значення 3,4 або 6 вказує на тип ґратки. Завдяки її використанню в операторі **Select Case … End Select** можна використовувати одні й ті ж процедури = незалежно від форми клітин - для вводу, виводу та моделювання.

Розглянемо докладно, як це виглядає в програмі.

Відображення

Для виведення на екран у програмі використовуються наступні методи класу **Graphics: DrawLine, DrawPolygon, DrawEllips, FillPolygon.** Виведення сітки та колонії на екран описані у процедурі **OutPut,** яка викликається в кількох місцях програми. Послідовність дій буде такою.

1. Визначаємо тип ґратки (type=3,4,6)
2. Для квадратів промальовуємо ґратку лініями

3 В подвійному циклі продивляємося всі активні клітини масиву:

```
For i = 1 To m
For j = 1 To n
   Вивід комірки a(i,j)
            Next
        Next
```

4. Визначаємо координати центру клітини **x** та **y** з урахуванням масштабу **d**:
Для трикутників:

```
If (i + j) Mod 2 = 0 Then
  x = d * i / 2 + dx
  y = d * ((j - 2 / 3) * (0.75 ^ 0.5))
Else
  x = d * i / 2 + dx
  y = d * ((j - 1 / 3) * (0.75 ^ 0.5))
End If
```

Для квадратів:

```
x = d * (i - 0.5)
y = d * (j - 0.5)
```

Для шестикутників:

```
x = 1.5 * d * i
y = Math.Sqrt(3) * d * j
```

5. Створюємо масив із точок – вершин многокутника:
6. Переходимо до віконних координат з урахуванням прокрутки
7. Промальовуємо многокутник
8. Якщо a(i,j)=True, малюємо посередині кружечок червоного кольору, інакше – кольору фону (щоб не використовувати щоразу метод Clear, що викликає ефект миготіння).

Процес виведення був прискорений наступним чином: Ґратка промальовується не на кожному ході, а лише при зміні її типу, масштабу або положення прокрутки. На кожному ході промальовуються лише ті комірки, що на цьому ході змінили свій стан.

У програмі є можливість прокрутки для відображення великих полів. Вона реалізується за допомогою компонентів **VScrollBar** і **HScrollBar,** які автоматично активізуються, коли розмір поля перевищує розмір вікна і дозволяють змінювати віконні координати верхньої лівої точки ігрового поля.

У програмі можна ввести конфігурацію за допомогою миші. Цей процес описано в процедурі, що опрацьовує подію клацання по формі. Вона працює таким чином: 1) отримує екранні координати курсора (**Cursor.Position)**; 2) переводить їх у віконні координати з урахуванням положення форми; 3) для квадратів – за формулами знаходить індекси комірки, на якій знаходиться курсор і, якщо така існує, змінює її стан; 4) для

інших фігур – проходимо циклом по всіх комірках, і якщо відстань між курсором та центром комірки менше радіусу вписаного кола, змінюємо її стан.

Додаткові можливості редагування

Програма надає наступні можливості редагування: 1) заповнення масиву **a** значенням **False; 2)** виконання операції **Not** над всіма клітинами; 3) генерування випадкової колонії, що має задану густину **plt** за наступним алгоритмом:

```
For i = 1 To m
 For j = 1 To n
     a(i, j) = (Rnd() * 100 < plt)
```

За допомогою методу **WriteAllText** можна зберегти параметри гри та колонію в текстовий файл.

За допомогою функції **OpenTextFieldParser** можна відтворити раніше збережену колонію.

Можна завантажити конфігурацію, текстовий опис якої було скопійовано до буферу обміну.

Програма має власний алгоритм запису конфігурацій до файлів, але може розпізнавати і створювати файли конфігурацій за стандартом **Run Length Encoding**, використаному в багатьох моделюючих програмах.

Моделювання

Програма має можливість автоматичного (за допомогою таймера зі змінною частотою, який запускається і зупиняється командними кнопками або клавішею **F9**) та покрокового (натисканням на кнопку) моделювання. В обох випадках на кожному кроці викликається процедура **Generation,** яка працює наступним чином:

В циклі переглядаються всі комірки ґратки;

Для кожної комірки **a(i,j)** визначаються координати сусідів:

Таблиця 2. Координати найближчих сусідів для різних типів ґраток.

Трикутна ґратка	Квадратна ґратка	Шестикутна ґратка
a(i,j-1)	a(i-1,j-1)	a(i,j-1)
a(i,j+1)	a(i-1,j+1)	a(i,j+1)
a(i+1,j-1)	a(i+1,j-1)	a(i-1,j)
a(i+1,j)	a(i+1,j+1)	a(i+1,j)
a(i+1,j+1)	a(i,j-1)	a(i-1,j+1-2*(i mod 2))
a(i-1,j-1)	a(i,j+1)	a(i+1, j+1-2*(i mod 2))

a(i-1,j) a(i-1,j+1) a(i-2,j) a(i+2,j) a(i-2, j-1+2*((i+j) mod 2)) a(i+2, j-1+2*((i+j) mod 2))	a(i-1,j) a(i+1,j)	

Якщо індекси якихось сусідів виходять за межі ігрового поля - такі сусіди не враховуються. Якщо моделювання проходить в режимі тору (в цьому режимі вважається, що колонія розвивається на поверхні тору, тобто крайній верхній ряд є сусіднім з нижнім, а крайній лівий – з крайнім правим; це дуже зручно при дослідженні рухомих конфігурацій) і сусід має індекси **(x,y)**, вони опрацьовуються за наступним алгоритмом:

```
If x < 0 Then x += m Else If x > m Then x -= m
If y < 0 Then y += n Else If y > n Then y -= n
```

Далі обчислюється кількість фішок-сусідів;

Кількість таких сусідів порівнюється зі змінними, що задають правила, визначається наступний стан кожної комірки, який заноситься до допоміжного масиву **b.** У коді це виглядає так:

```
b(i, j) = a(i, j)
If a(i, j) Then
If (sum < aliveMin)Or(sum > aliveMax) Then b(i, j) =
False
  If Not b(i, j) Then Population -= 1
Else
  If (sum<=appearMax)And(sum>=appearMin)Then b(i, j) =
True
  If b(i, j) Then Population += 1
End If
```

Після циклу масив **b** переписується до масиву **a;**

Викликається процедура виводу **OutPut.**

Крім того, для прискорення роботи введено логічні масиви **edit1** (він вказує, які комірки треба перевіряти) та **edit2,** всі елементи якого на початку ходу дорівнюють **False,** а під час роботи всі елементи, які змінились, та сусіди таких елементів змінюють значення на **True.** Після циклу масив **edit2** переписується до масиву **edit1.** Це дозволяє не перевіряти комірки, які не зміняться і значно прискорити роботу програми.

Налаштування

В меню є широкі можливості змінювати параметри програми. Серед них: 1) ширина та висота поля – змінюються змінні **m** та **n;** 2) **м**асштаб – змінюється змінна **d** за допомогою **TrackBar; 3)** режим відображення – кружечком або зафарбовуванням; 4) умови виживання та народження (генетичні закони): 5) **aliveMin –** мінімальне число сусідніх фішок для виживання; 6) **aliveMax –** максимальне число сусідніх фішок для виживання; 7) **appearMin –** мінімальне число сусідніх фішок для народження; 8) **appearMax** - максимальне число сусідів для народження; 9) моделювання в режимі площини чи тору; 10) моделювання в режимі врахування далеких фішок (лише для шестикутників); 11) частота моделювання – змінюється інтервал таймеру **Timer1; 12)** колір фону, ґратки та клітин – за допомогою **ColorDialog**

Інструменти аналізу

В меню «Аналіз» моєї програми є наступні функції: 1) **підрахунок популяції** (кількості фішок); 2) **підрахунок кількості ходів**, що відбулися з певного моменту; 3) **підрахунок періоду.** Ця функція дозволяє знайти період, з яким конфігурація поля повторюється, для чого поле зберігається в окремий масив і на кожному ході порівнюється з ним, доки не буде знайдено період

Графік концентрацій. Ця функція корисна для дослідження загальних тенденцій еволюції колонії за певних правил. Вона будує графік залежності кінцевої густини колонії (відношення кількості живих клітин до загальної кількості комірок) від початкової і дозволяє наочно бачити, при яких початкових концентраціях колонія вимирає, а при яких – досягає певної стійкої конфігурації. Працює вона наступним чином: 1) у циклі для **i1** проходяться концентрації від 0% до 100% з кроком 1%; 2) певну кількість разів генеруємо випадкову колонію для кожної концентрації; 3) моделюємо без графічного виводу еволюцію цієї колонії, доки 20 разів поспіль популяція не зміниться, або доки кількість ходів не перевищить 1000; 4) обчислюємо для кожного кінцевого розташування фішок густину та обчислюємо середнє значення для кожної початкової густини **i1** й заносимо його до масиву **a** форми **Form2**; 5) за значеннями елементів цього масиву будується графік на формі **Form2; 6)** обчислюємо значення дисперсії для отриманої вибірки:

$$\boldsymbol{D} = \frac{\sqrt{\frac{\sum_{i=0}^{100}(avg - a(i))^2}{100}}}{avg}$$, де $\boldsymbol{avg} = \frac{\sum_{i=0}^{100} a(i)}{101}$

Після закінчення розрахунків (максимально вони можуть займати до 30 хвилин) графік має наступний вигляд:

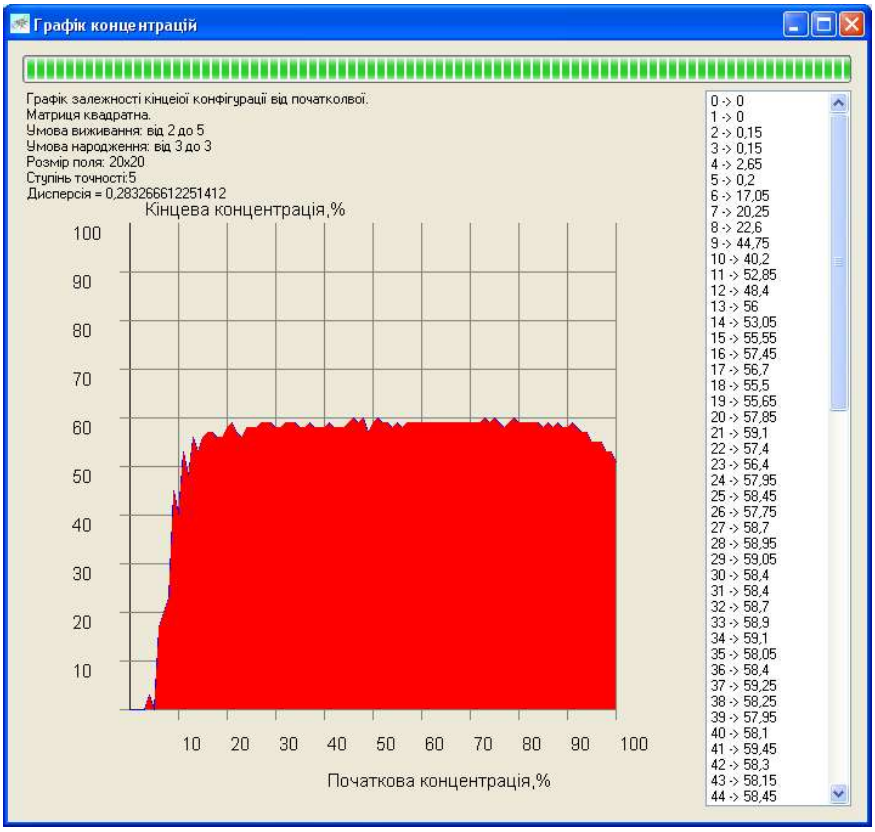

Рис. 17. Вікно, що відображає залежність кінцевої концентрації клітин від початкової

Вікно графіку також містить таблицю результатів, за якими побудовано графік. Її можна зробити невидимою, натиснувши кнопку **F8.**

ЧАСТИНА 3. ВИКОНАНІ ДОСЛІДЖЕННЯ

ДОСЛІДЖЕННЯ ГРИ «ЖИТТЯ» НА ТРИКУТНІЙ ҐРАТЦІ ЗА КЛАСИЧНИХ ПРАВИЛ

Для початку розглянемо прості стійкі конфігурації («любителі спокійного життя»). Взагалі, їх дуже мало, тому що кожен трикутник має по 12 сусідів і велика ймовірність смерті клітини в ньому від перенаселення. Найпростіші мають від 4 клітин у своєму складі:

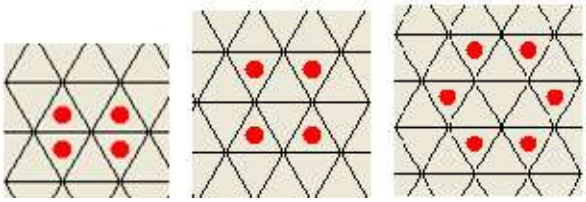

Рис. 18. Найпростіші стабільні конфігурації для правильних трикутників

До першої з них переходять майже всі конфігурації, які не самознищуються. Незважаючи на нестабільність конфігурацій, існують циклічні конфігурації. Одна з них має період 8 і змінює свою популяцію від 4 до 8. До того ж, через пів періоду вона симетрично відображається:

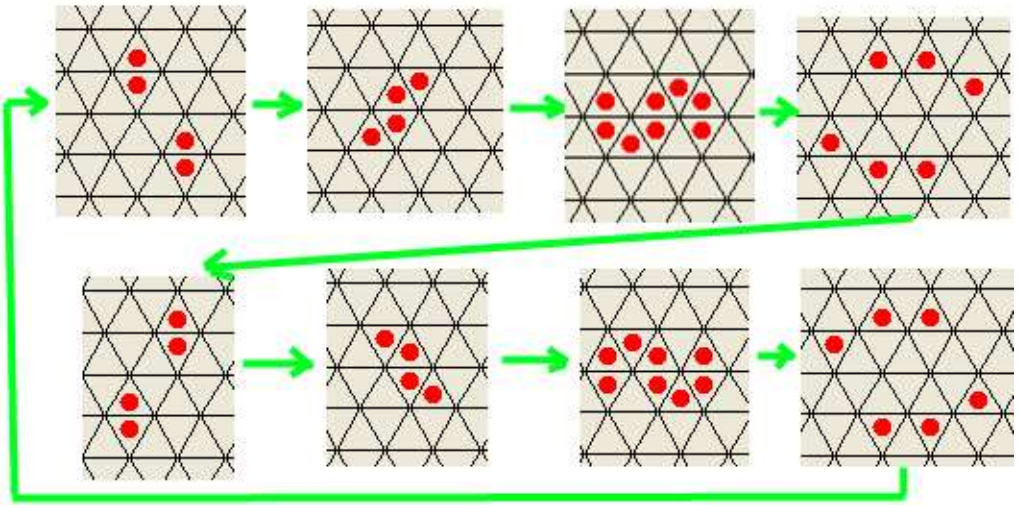

Рис. 19. Приклад циклічної конфігурації на трикутній ґратці

При дослідженні випадкових конфігурацій було виявлено, що еволюція (при початкових концентраціях 5..75%) проходить дуже довго, але закінчується повним вимиранням, або переходом до кількох стійких конфігурацій. Часто утворюються рухомі конфігурації та конфігурації, що швидко ростуть. Але й вони самознищуються або переходять в найпростіші стійкі конфігурації. Випадкові конфігурації з концентрацією до 5% та вище 75% вимирають практично миттєво.

При дослідженні кінцевих концентрацій випадкових конфігурацій різної концентрації на полі з 400 трикутників було отримано наступний графік:

Рис. 20. Графік залежності кінцевої конфігурації від початкової для трикутної ґратки

Перехід до стійкої конфігурації починається при початковій концентрації фішок 5% та закінчується при 72% **(кінцева – 0.1% в обох випадках).** Максимальна концентрація стійкої конфігурації (1,3%) досягається при початковій – 16% (тобто в середньому 5,2 стійких клітин з 64 початкових). Це ще раз підтверджує тяжіння досліджуваної популяції до самознищення.

Пошук складних конфігурацій (глайдерів, гармат, паровозів) не дав плідних результатів, бо кожне покоління змінює дуже багато клітин, і можливі лише дуже прості стабільні структури.

Можна зробити висновок, що гра «Життя» на трикутній ґратці в цілому нагадує класичну (на квадратній ґратці), але є породжує менше стійких конфігурацій.

ДОСЛІДЖЕННЯ ГРИ «ЖИТТЯ» НА ШЕСТИКУТНІЙ ҐРАТЦІ ЗА КЛАСИЧНИХ ПРАВИЛ

При дослідженні випадкових конфігурацій були знайдені статичні конфігурації («любителі спокійного життя»). Найпростіші з них містять від трьох клітин і є легко передбачуваними. Стійких конфігурацій з трьох і чотирьох клітин існує лише по одній:

Рис. 21. Найпростіші стійкі конфігурації для ґратки з правильних шестикутників

Вони частіше всього утворюються при еволюції випадкових колоній. Немає стійких конфігурацій із п’яти комірок, а з шести їх є дві:

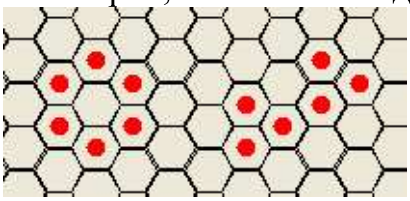

Рис. 22. Стійкі конфігурації з 5 та 6 клітин

Ось іще кілька стійких конфігурацій, що містять від семи до дев’яти клітин:

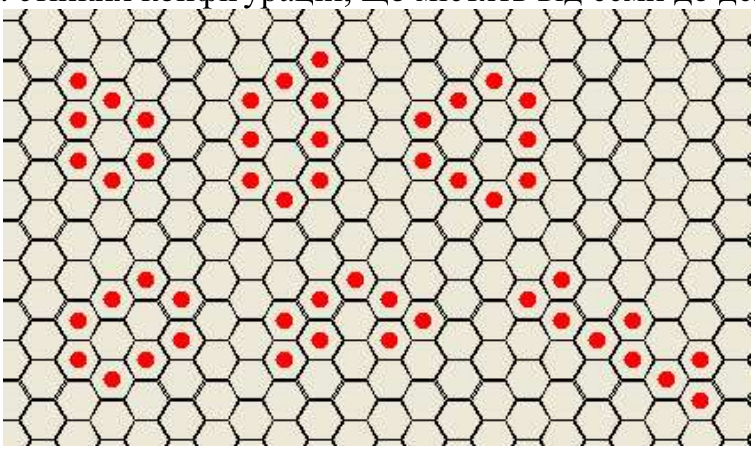

Рис. 23. Стійкі конфігурації з 7 – 9 клітин

Хоча число стійких конфігурацій мале, майже всі невеликі конфігурації переходять до стійких.

Враховуючи, що для виживання клітини достатньо двох сусідів, деякі конфігурації можна «витягувати» до нескінченності, подібно до «кораблів» та «барж» на квадратній ґратці (рис. 20а).

До речі, такі конструкції мають особливість: якщо всередині них розміщувати фішки (не сусідні одна з одною), вони зникають. Це чимось подібне до конструкції «агар» на чотирикутній ґратці.

Взагалі, «Життя» на шестикутній ґратці за класичними правилами тяжіє до стабільних структур. Так, будь-який ланцюг (тобто комбінація, де в кожної

фішки по два сусіди) з фішок буде стабільним, якщо лише він не буде народжувати нові фішки (тобто, в місцях вигину біля одної порожньої клітини має бути не три, а чотири фішки, див рис. 20б).

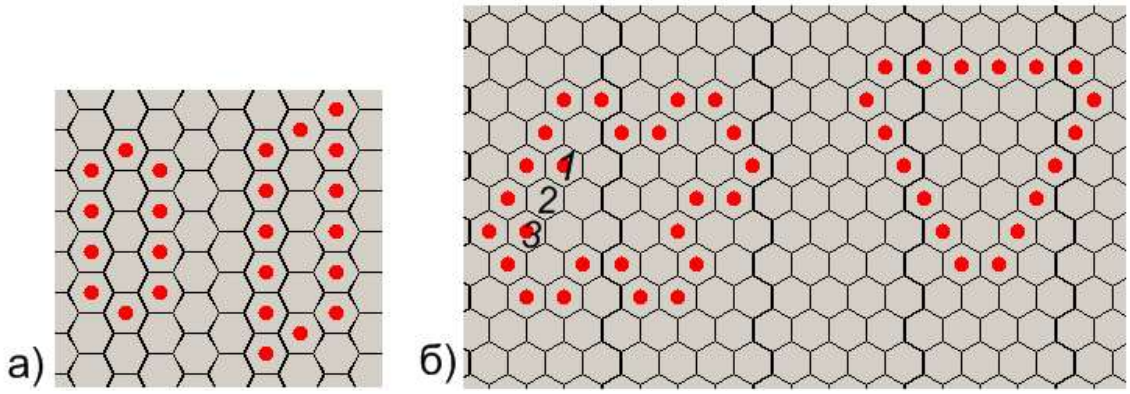

Рис. 24. Приклади витягування стійких конфігурацій

Для стабільності в деяких місцях (клітини 1 і 3) перший ланцюг був «розширений». Інакше, якщо клітина 1 була б порожньою, у клітині 2 народилася б нова клітина, померла б клітина 3 і стабільність порушилася б. Другий ланцюг не змінюється, бо в місцях повороту внутрішня порожня клітина має 4 сусіди, - там не з'являється клітина.

Можуть бути стабільними й незамкнені ланцюги:

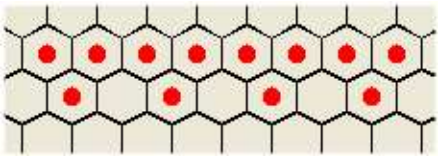

Рис. 25. Приклад стійкого незамкненого ланцюгу

Цей ряд складений з найпростіших стабільних конфігурацій і може бути продовжений нескінченно. Тут комірки, що знаходяться на межі ігрового поля та **комірки другого ряду мають двох сусідів**, інші клітини - по 3 сусіди і поява нових клітин неможлива.

Взагалі, стабільним можна зробити будь-який незамкнений ланцюг, побудований за принципом неможливості народження нових фішок. Для цього з обох боків необхідно приєднати спеціальні стабілізуючі структури:

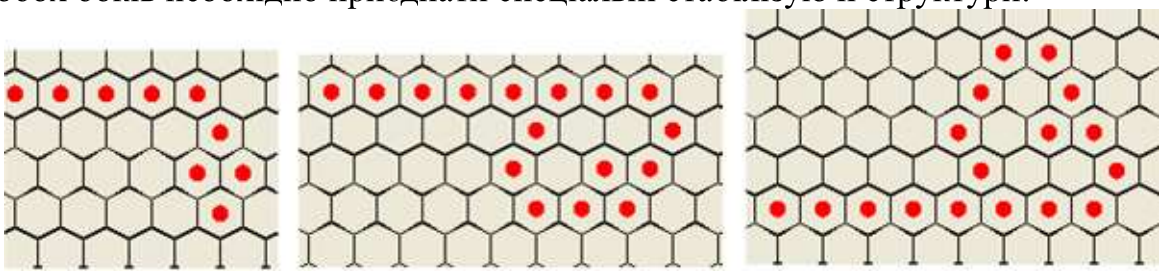

Рис. 26. Приклади стабілізуючих структур

Вони самі є стабільними і біля місця приєднання ланцюгу фішки не помирають і не народжуються. За допомогою цих структур можна отримати такі конфігурації:

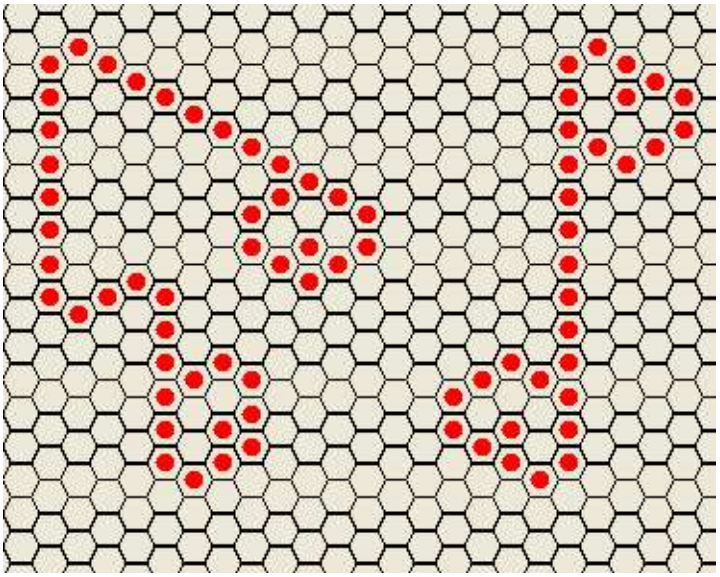

Рис. 27. Приклади стабілізованих конфігурацій

Якщо ж такий ланцюжок розірвати або «заглушити» лише з однієї сторони, то він буде зникати, втрачаючи за хід по одній клітині, а потім «заглушка» за кілька ходів знову перетвориться на стійку конфігурацію.

Взагалі, стійких конфігурацій за класичних умов нескінченно багато. Можна навести ще кілька прикладів, отриманих з еволюції випадкових конфігурацій:

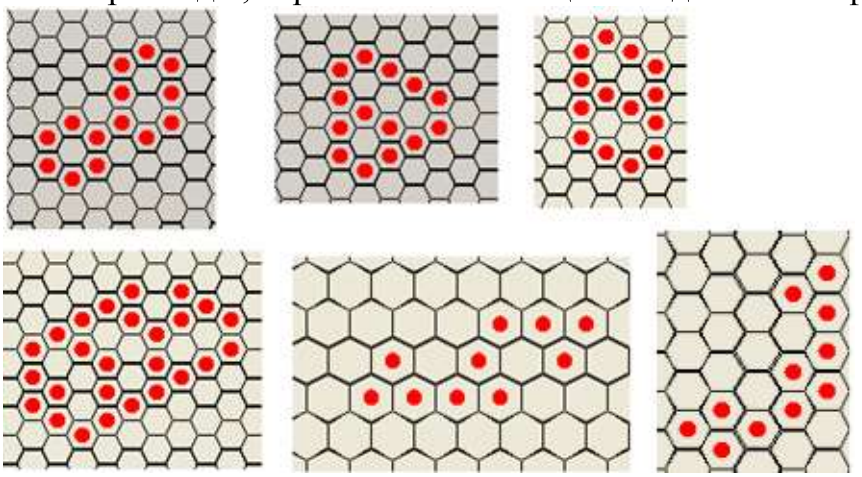

Рис. 28. Приклади стійких конфігурацій, одержаних з випадкових початкових конфігурацій

Тепер розглянемо періодичні конфігурації. Найпростіша з них складається з чотирьох фішок і має період, рівний двом:

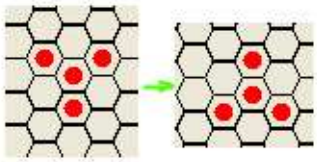

Рис. 29. Найпростіша періодична конфігурація на шестикутній ґратці

Вона працює так само, як «миготілка» на чотирикутній ґратці. Центральна фішка живе постійно, а три бокові кожного разу вмирають, бо в них лише

один сусід. Та одночасно народжуються три фішки в інших місцях. При цьому чисельність популяції не змінюється.

Розглянемо ще одну конфігурацію з періодом, рівним двом:

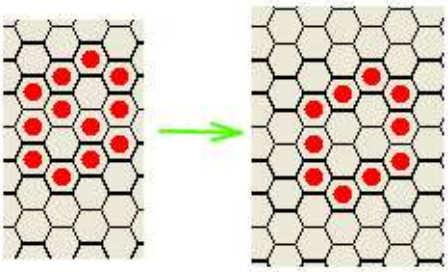

Рис. 30. Періодична конфігурація

Тут змінюються стани лише двох центральних комірок. Фішки в них народжуються, бо в них є по три сусідніх фішки. Але вони є сусідами одне одному, тому в наступному поколінні помирають від перенаселення. Отже, чисельність популяції коливається між 10 і 12 фішками. За принципом дії ця конфігурація нагадує «Бакен».

На шестикутній ґратці є конфігурації і з більшим періодом, але вони складніші:

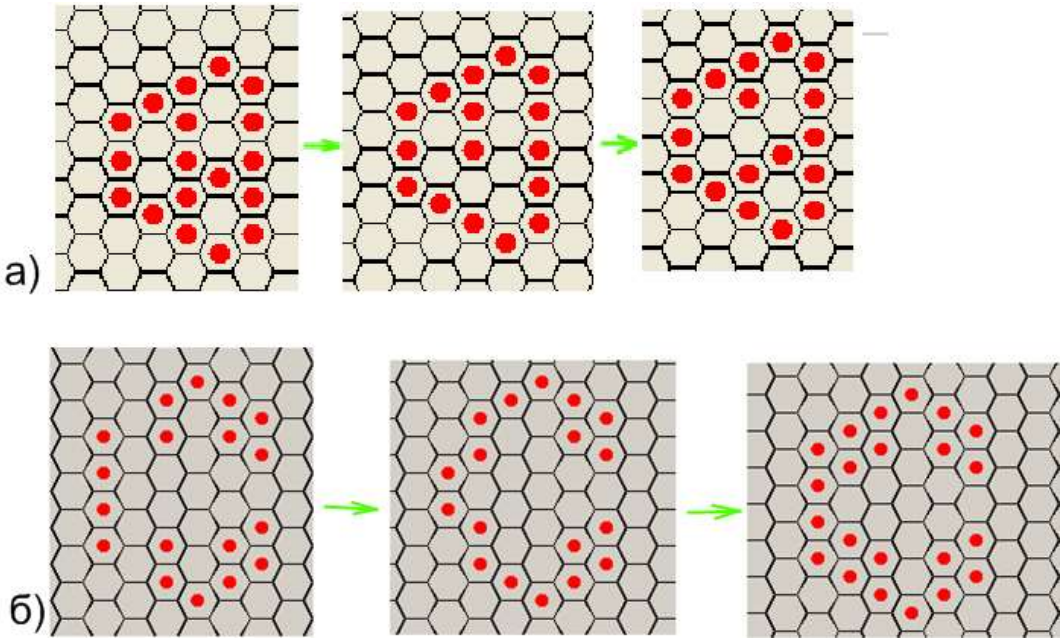

Рис. 31. Складні періодичні конфігурації

Обидві наведені конфігурації мають період, рівний трьом. В першій конфігурації змінюються стани лише трьох центральних комірок,. При цьому число фішок у популяції коливається між 16 та 18. У другій конфігурації змінюється лише ліва частина, число фішок у популяції протягом двох ходів складає 16, на третьому – 20 і має симетрію.

Таким чином, гра «Життя» на шестикутниках багато в чому подібна до класичної на квадратах.

Спроби знайти більш складні конфігурації, такі як «гармати» або «космічні кораблі» поки невдалі.

Також за допомогою моєї програми було досліджено загальні тенденції еволюції колонії в залежності від її початкової концентрації. Для звичайної моделі поля 20X20 було отримано графік:

Рис. 32. Графік залежності кінцевої конфігурації від початкової для шестикутної ґратки

Для моделі тору 20X20 було отримано графік:

Рис. 33. Графік залежності кінцевої конфігурації від початкової для шестикутної ґратки (модель тору).

Як можна бачити з графіків, перехід до стійких конфігурацій спостерігається при початковій концентрації фішок - від 3% до 95%. Кінцева концентрація фішок більша від 5% досягається при початковій концентрації фішок від 23% до 72%. Їх максимальна кінцева концентрація складає 8% і досягається при початковій 64%. Видно, що при малих початкових концентраціях кінцева концентрація зростає зі збільшенням початкової, а при великих – спадає.

ДОСЛІДЖЕННЯ ГРИ «ЖИТТЯ» НА ШЕСТИКУТНІЙ ҐРАТЦІ В РЕЖИМІ ВРАХУВАННЯ ДАЛЬНІХ СУСІДІВ

Досліджена адаптація гри «Життя» для шестикутників, запропонована в 1999 році Д. Белліджером (*David G. Ballinger*). Його модель передбачає врахування при моделюванні не лише шести фішок - безпосередніх сусідів, а й шести «дальніх сусідів», що разом утворюють зірку Давида:

Рис. 34. Модель гри «Життя» на шестикутній ґратці за Д. Белліджером

При підрахунку кількості усіх сусідніх фішок їх необхідно враховувати з коефіцієнтом 0.3, тоді як безпосередніх сусідів – з коефіцієнтом 1. Тоді максимальна сума складає 6*1+6*0.3=7.8, що наближається до восьми сусідів на квадратній ґратці.

До того ж, щоб найточніше відтворити еволюцію на квадратній ґратці (відсутність швидко помираючих та швидко зростаючих конфігурацій, складна еволюція простих конфігурацій), необхідно скоригувати генетичні закони: 1) фішка виживає, якщо кількість сусідніх фішок з урахуванням коефіцієнтів лежить у діапазоні [2,0;3,3], інакше вона вмирає; 2) у порожній комірці народжується фішка, якщо кількість фішок-сусідів знаходиться в діапазоні [2,3;2,9].

У моїй програмі є можливість моделювання при таких умовах - в тому числі у режимі тору.

Досліджено особливості розвитку колоній у цьому режимі. Еволюція будь-яких конфігурацій проходить дуже довго (час пропорційний розміру поля), але врешті-решт всі фішки зникають. При цьому їх початкова концентрація не є суттєвою. Простих стійких конфігурацій існує дуже мало. Одна з них має такий вигляд:

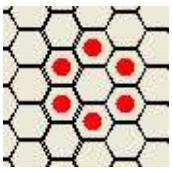

Рис. 35. Приклад простої стійкої конфігурації

Але в цьому режимі існує рухома конфігурація, дуже схожа на «глайдер». Через кожні чотири ходи вона відтворює себе і повертається, а за вісім ходів повертається до початкового вигляду зі зсувом на чотири клітини. Ця конфігурація часто утворюється при еволюції випадкових колоній:

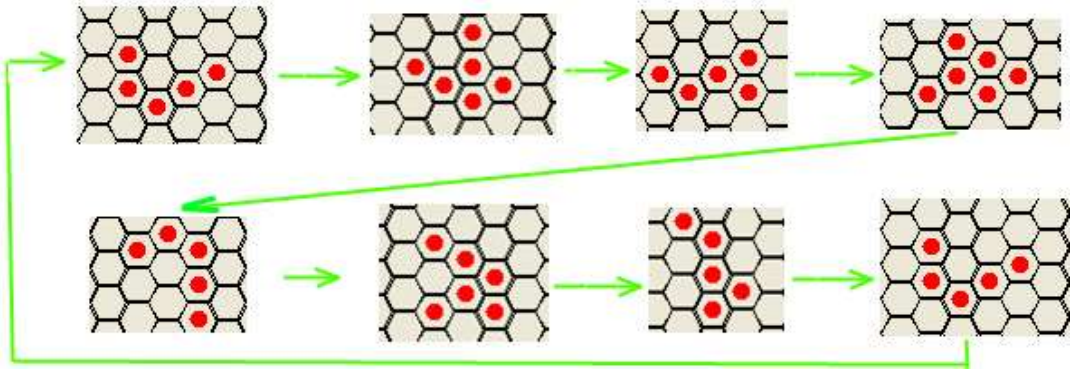

Рис. 36. Еволюція аналогу «глайдера» на шестикутній ґратці

Висновки

1. Створено програму для моделювання та дослідження еволюції колоній клітинних автоматів на двовимірних гратках із правильних трикутників, чотирикутників та шестикутників. Спостережено еволюцію відповідних колоній при класичних (за Конвеєм) та відмінних від класичних умовах

виживання. Комп'ютерне дослідження проведено для різних початкових конфігурацій та для різних початкових заповнень граток.

2. Для різних видів ґраток та умов виживання визначено стійкі та періодичні конфігурації та досліджено загальні тенденції еволюції колоній. Всі одержані результати порівнювалися з класичними.

Подяка

Автор роботи висловлює подяку науковому керівнику Ентіну Йосифу Абрамовичу за постановку задачі та корисні рекомендації до роботи.

Література

1. Мартин Гарднер. Математические досуги. – М.: ОНИКС, 1995 – 496 с.
2. http://ru.wikipedia.org/wiki/Жизнь_(игра)
3. http://www.well.com/~dgb/hexlife.html