

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра комп'ютерної інженерії та кібербезпеки

**КУРСОВИЙ ПРОЕКТ**  
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Об'єктно-орієнтоване програмування»

на тему:

**«РОЗРОБКА ПРОГРАМИ ГРИ «ЖИТТЯ»»**

студента I курсу групи КІ-20-1  
спеціальності 123 «Комп'ютерна  
інженерія»

Магуріна Олексія Олександровича

(прізвище, ім'я та по-батькові)

Керівник старший викладач кафедри ПІЗ  
Власенко О. В.

Дата захисту: " \_\_ " \_\_\_\_\_ 20\_\_ р.

Національна шкала \_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп'ютерних технологій

Кафедра комп'ютерної інженерії та кібербезпеки

Освітній рівень: бакалавр

Спеціальність 123 «Комп'ютерна інженерія»

«ЗАТВЕРДЖУЮ»

зав. кафедри

А.А. Єфіменко

“ ” 20\_\_ р.

ЗАВДАННЯ  
НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ  
Магуріну Олексію Олександровичу

- Тема роботи: Розробка програми гри «Життя»,  
керівник роботи: старший викладач кафедри інженерії програмного забезпечення Власенко Олег Васильович.
- Строк подання студентом: “ 25 ” травня 2021р.
- Вихідні дані до роботи: Розробити програму гри "Життя" - симуляцію законів клітинного автомату.
- Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
  - Постановка завдання
  - Аналіз аналогічних розробок
  - Алгоритми роботи програми
  - Опис роботи програми
  - Програмне дослідження
- Перелік графічного матеріалу(з точним зазначенням обов'язкових креслень)
  - Презентація до КП

Посилання на репозиторій: <https://gitlab.com/2020-2024/20-1/mahurin-oleksii/game-of-life-windows-forms>
- Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2	Левківський В.Л., ст. викладач каф. КН		
1,2	Марчук Г.В., ст. викладач каф. КН		

--	--	--	--

7. Дата видачі завдання “ 17 ” лютого 2021 р.

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проекту	Строк виконання етапів проекту	Примітки
1	Постановка задачі	02 березня	
2	Пошук, огляд та аналіз аналогічних розробок	25 березня	
3	Формулювання технічного завдання	30 березня	
4	Опрацювання літературних джерел	15 квітня	
5	Проектування структури	25 квітня	
6	Написання програмного коду	8 травня	
7	Відлагодження	16 травня	
8	Написання пояснювальної записки	18 травня	
9	Захист	31 травня	

**Студент**

\_\_\_\_\_

(підпис)

Магурін О. О.

(прізвище та ініціали)

**Керівник проекту**

\_\_\_\_\_

(підпис)

Власенко О. В.

(прізвище та ініціали)

## РЕФЕРАТ

Завданням на курсового проекту (роботи) було створення гри «Життя».

Пояснювальна записка до курсового проекту (роботи) на тему «Розробка програми гри «Життя»» складається з вступу, трьох розділів, висновків, списку використаної літератури та додатків. Текстова частина викладена на 38 сторінках друкованого тексту. Пояснювальна записка має 10 сторінок додатків.

Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 23 рисунків. Загальний обсяг роботи – 48 сторінок. Ключові слова: WINDOWS FORMS, GAME OF LIFE, CONWAY'S GAME OF LIFE, ГРА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КЛІТИННИЙ АВТОМАТ

					ДУ «Житомирська політехніка».21.123.14.000 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка програми гри «Життя»	Літ.	Арк.	Аркушів
Розроб.		Магурін О.О.						
Перевір.		Левківський В.Л.					4	38
Керівник		Власенко О.В.				ФІКТ Гр. КІ-20-1[1]		
Н. контр.								
Зав. каф.		Єфіменко А.А.						

## ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	7
1.1 Аналіз задачі, засобів та методів її вирішення.....	7
1.2 Аналіз існуючого програмного забезпечення за тематикою курсового проекту.....	8
Висновки до першого розділу.....	14
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	15
2.1 Проектування загального алгоритму роботи програми.....	15
2.2 Розробка функціональних алгоритмів роботи програми.....	17
2.3 Розробка програмного забезпечення.....	20
Висновки до розділу.....	27
РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ.....	28
3.1 Опис роботи з програмним додатком.....	28
3.2 Тестування роботи програмного забезпечення.....	32
Висновки до розділу.....	35
ВИСНОВКИ.....	36
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	37
ДОДАТКИ.....	38

## ВСТУП

**Актуальність:** дана тема представляє теоретичні та практичні інтереси, тому що клітинні автомати є дискретною моделлю в математиці, яка проявляє та вивчається в теорії обчислюваності, фізиці, теоретичній біології та мікромеханіки.

**Ціль:** провести аналіз принципу роботи клітинних автоматів та розробити програму гри «Життя».

**Об'єкт:** клітинні автомати та інформаційні технології.

**Предмет:** розробка програми та реалізація законів клітинного автомату Джона Хортон Конвея.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

## РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

### 1.1 Аналіз задачі, засобів та методів вирішення

«Гра життя» — клітинний автомат, вигаданий англійським математиком Джоном Конвеєм 1970 року.

Місце дії гри — «всесвіт» — являє собою площину, поділену на клітинки. Кожна клітинка може перебувати в одному з двох станів: бути живою або бути мертвою. Клітинка має вісім сусідів. Розподіл живих клітинок на початку гри називається першим поколінням. Кожне наступне покоління утворюється на основі попереднього за наведеними нижче правилами:

- якщо в живій клітині два чи три живих сусіди — то вона лишається жити;
- якщо в живій клітині один чи немає живих сусідів — то вона помирає від «самотності»;
- якщо в живій клітині чотири та більше живих сусідів — вона помирає від «перенаселення»;
- якщо в мертвої клітині рівно три живих сусіди — то вона оживає.

Поставлено завдання розробити програмний додаток з використанням основних принципів ООП на цю тему.

Сама програма буде написана на мові C# з використанням середовища розробки Microsoft Visual Studio. Проект буде розроблятися на Windows Forms.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

## 1.2 Аналіз існуючого програмного забезпечення за тематикою курсового проекту

Перед початком роботи над своїм проектом, треба проаналізувати аналогічні програмні додатки, які існують на сьогодні, щоб визначити найголовніші функції майбутньої програми.

Найвідоміших версій гри «Життя» Джона Конвея всього декілька.

### John Conway's Game of Life by Edwin Martin

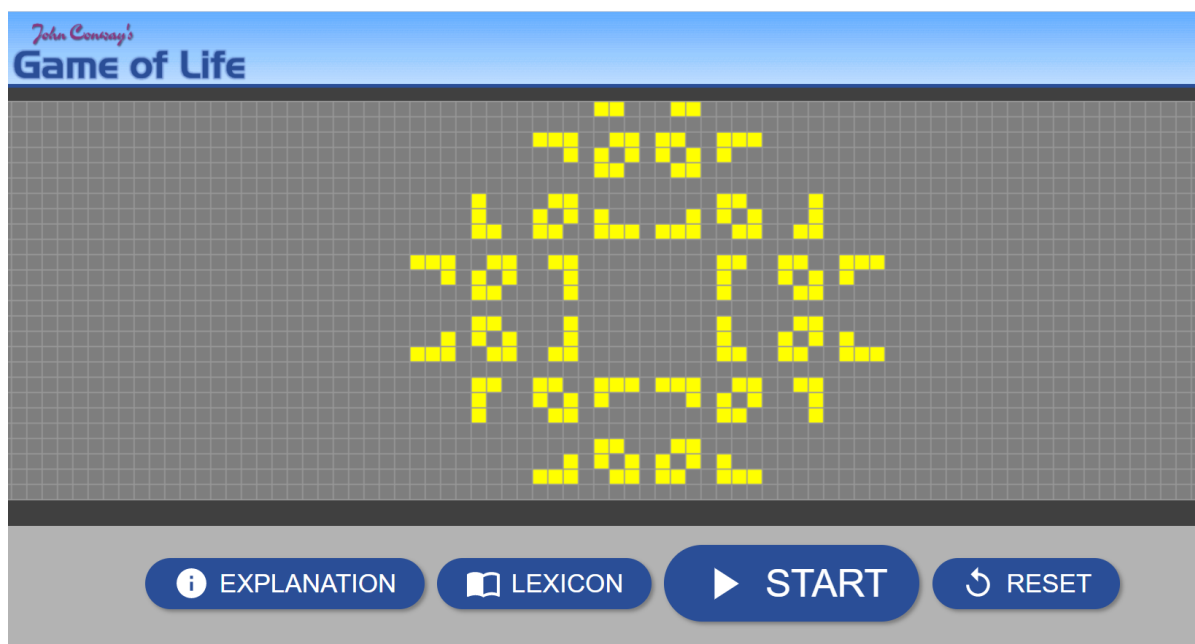


Рис. 1.1. Програма гри «Життя» Едвіна Мартіна, яка запускається в браузері

Перша гра запускається у браузері за адресою <https://playgameoflife.com/>. Програма має модуль з цікавими фігурами та можливістю виводити їх на поле, також присутні теоретичні відомості щодо роботи клітинного автомату, та кнопку старт-стоп та сбросу, які керують симуляцією.

Перевагою гри є те, що вона доступна в браузері, але недоліків більше – неможливо керувати швидкістю симуляції, змінити розмір поля та незручне малювання фігур власноруч.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



# Conways Game Of Life by Bytewire Software

Ця гра також є безкоштовною, і розміщена в магазині Microsoft Store.



Рис. 1.2. Сторінка в Microsoft Store гри Conways Game Of Life від компанії Bytewire Software.

На відміну від попередньої розглянутої в браузері гри, ця має гарний та приємний інтерфейс, та багато корисних налаштувань, наприклад: змінення розміру поля та правила гри (Classic Rule, Copy World, Explode, Labyrinth, Blinking). Присутнє також випадкове заповнення поля клітинами, кнопки старт-стоп та кнопка одиничної ітерації поля.

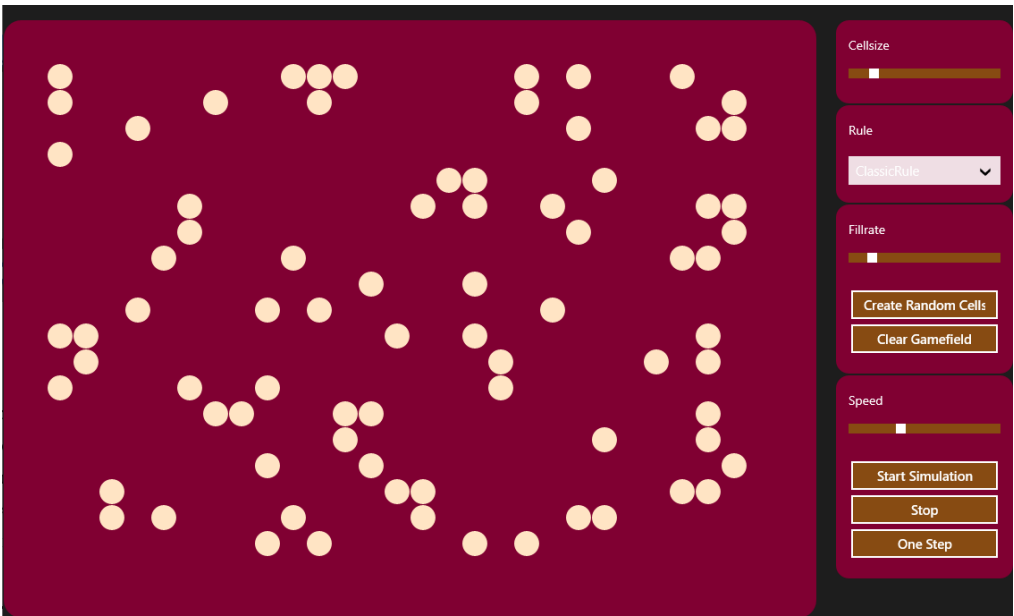


Рис. 1.3. Інтерфейс програми від компанії Bytewire Software

Хоч ця версія гри «Життя» набагато краща за попередню, в неї також є недоліки: недостатня гнучкість та незручність налаштування гри та присутність помилок: при звуженні вікна зникає поле з клітинами.

### Гра «Життя» Онлайн від Олексія Мічурина

Розглянемо третій, найбільш популярний в СНД країнах варіант гри «Життя».

Гра також є браузерною, як і перша.

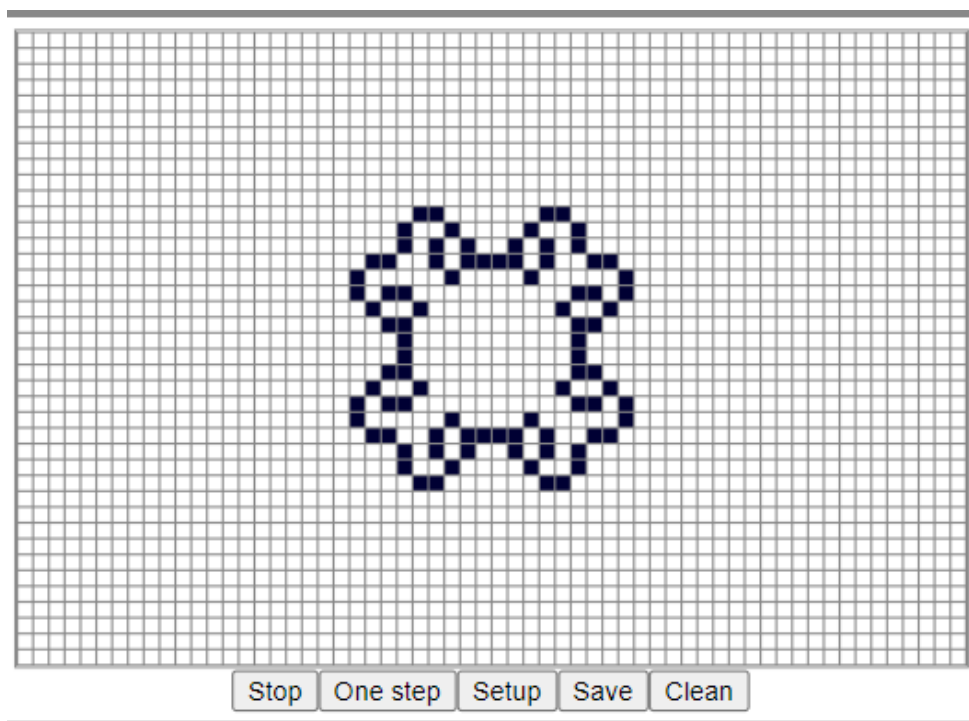


Рис 1.4. Графічний інтерфейс гри «Життя» Олексія Мічурина

Дана гра знаходиться на сайті <http://www.michurin.net/online-tools/life-game.html> і розроблена Олексієм Мічуриним.

На перший погляд, гра має поганий інтерфейс, але функціонал дуже цікавий та гнучкий, користувач має можливість налаштувати розмір поля за своїм бажанням, та затримку між ітерацією, очищення поля та One step ітерацію. Але найбільшою перевагою цієї версії гри «Життя» є величезна база заготовлених фігур. Користувач може легко додати одну із 120 заготовлених фігур на поле і подивитися, які чудові візерунки та цикли створюють такі прості правила.

		Магурін О. О.			ДУ «Житомирська політехніка». 21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Timeless
Glider
Classic spaceships
Lightweight spaceship
Middleweight spaceship
Heavyweight spaceship
Spaceships
Brain
Dart
Flotilla
Hivenudg
Schick Engine
Turtle
Weekend
Spider
Pushalong
Orion
Swan
Guns/Eaters/Generators
Gosper gun
Gosper gun + Eater
Glider Generator
Oscillators (period 2)
Blinker
Toad
Clock
Beacon
Bipole
Tripole
Quadpole
Pentapole
Ring A
Phoenix
Leaf
Dragonfly
Z
Test Tube Baby
Arrow
Wolf
Cavity
Skewed quad
Quad
Snake pit
Spark coil
Great on-off
Light bulb
Negentropy
Revolver
Scrubber
Muttering Moat
Fore and Back
Glasses
Oscillators (long period)
Relay
Twin Bees Shuttle
Prepulsar
Cuphook
Pulsar Quadrant
Jam
Mold
Caterer
Mazing
1-2-3-4
Boss
Cloverleaf
Confused Eaters
Eater Block Frob
Lightweight Emulator
Middleweight Emulator
Heavyweight Emulator
Hustler II
Jack
Fountain
Babbling Brook
Gray Counter
Monogram
Penny Lane
Pinwheel
Clock II
Snake Pit III
Cross
Maltese Cross
Diamond Ring
Eater-Bound Pond
French Kiss
Hustler
Keys
Pressure Cooker
Star
Trice Tongs
Tubber
Two Eaters
Long period oscillators (6, 7, 8)
A for All
Extremely Impressive
Pipsquitter
Rats
Sombreros
Unix
Airforce
Burloaferimeter
Hebdarole
Blocker
Cauldron
Coe's p8
Figure 8
Hertz Oscillator
Kok's Galaxy
Pyrotechnecium
Quilt Square
R2D2
Roteightor
Smiley
Very long period oscillators
Achim's p144
Champagne Glass (period 22)
Toadsucker (period 60)
Gourmet (period 32)
Pi Portraitor (period 32)
Popover (period 32)
Eureka (period 30)
Queen Bee Shuttle (period 30)
Queen Bee Shuttle (period 30) (B)
Achim's p16
Sailboat
Pentadecathlon (15)
Al Jolson (15)
Tumbler (14)
Baker's Dozen (12)
Crown (12)
Dinner Table (12)
Snacker (9)

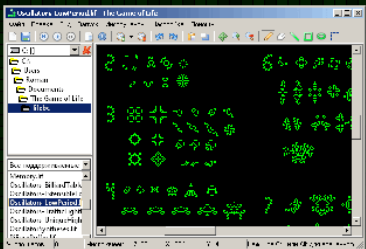
Рис. 1.5. База цікавих фігур гри «Життя»

## The Game of Life by Roman Parpalak

І остання програма, яка буде розглянута, це «The Game of Life», розроблена Романом Парпалаком, яка створена ще в 2005 році та знаходиться на сайті <https://life.written.ru/> . На цей час, це найкраща версія гри «Життя», яку є у вільному доступі.

# The Game of Life

The Game of Life — программа для моделирования игры «Жизнь», работает на любом компьютере под управлением Windows XP, Vista, 7.

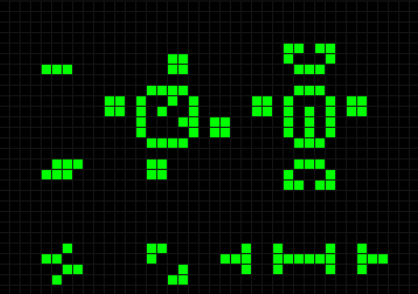


**Скачать**  
The Game of Life 3.6 beta (1 Mб)

Игра «Жизнь» — это увлекательная математическая игра, не похожая на остальные забавы. Происходящие в «Жизни» события лучше всего наблюдать на экране компьютера.

**Запустите «Жизнь» прямо сейчас!**

[Пустое поле](#)
[Осцилляторы](#)
[Устойчивые](#)  
[Космические корабли](#)
[Планерное ружье](#)



54

Рис. 1.6. Веб-сторінка про гру «Життя» та історія її розробки

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Ця програма включає в себе майже всі переваги попередніх та має багато нових функцій, які роблять її відмінним інструментом, щоб досліджувати роботу клітинних автоматів.

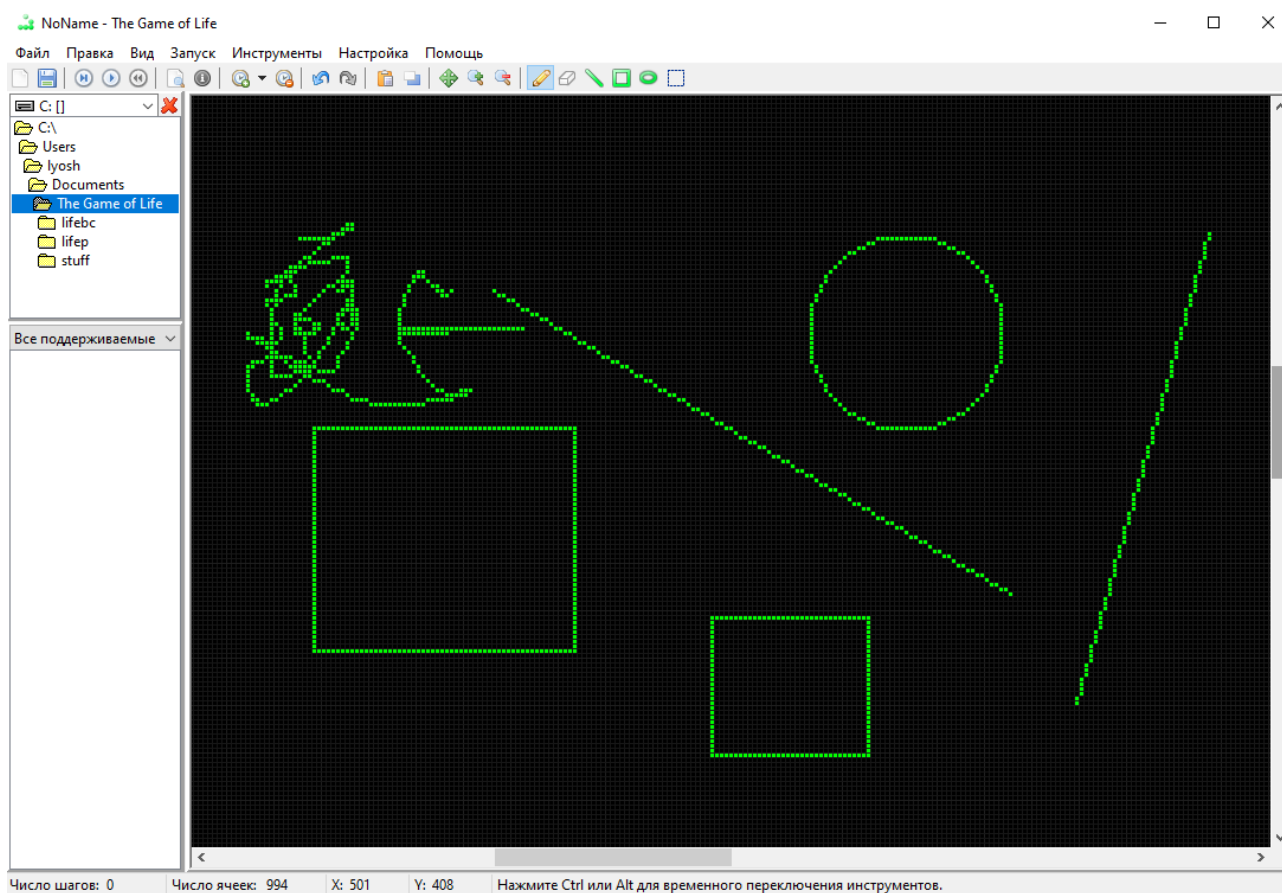


Рис. 1.7. Інтерфейс програми «The Game Of Life» Р. Парпалака

Дана програма має вдосконалений інтерфейс. Можливість налаштування параметрів гри та зручне управління. Також присутня робота з директоріями: користувач може створити свій файл з фігурами, зберігати та відкривати інші файли.

Не можна не зазначити зручність управління грою. Є можливість масштабування та переміщення по полю, що значно спрощує користування програмою на відміну від попередніх розглянутих версій гри.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

I, на мою думку, найголовніша функція, якої не було в попередніх розглянутих версій гри, це малювання фігур. Користувачу доступне малювання ліній, квадратів, прямокутників, кіл та еліпсів. Завдяки цим функціям, можна створювати безліч цікавих візерунків при запуску програми.

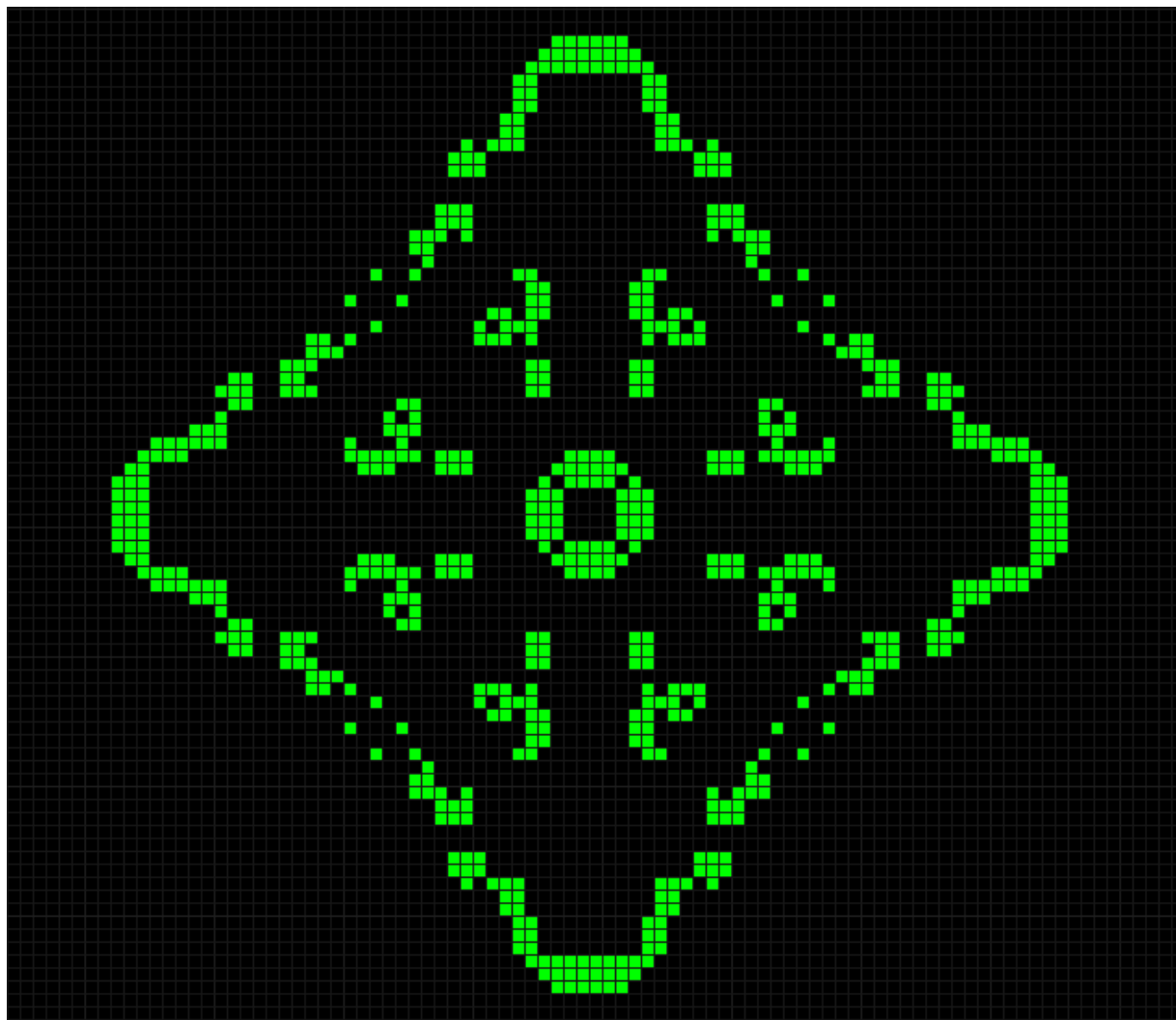


Рис. 1.8. Візерунок, який виник після декількох ітерацій з квадрату

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

## Висновки до першого розділу

У цьому розділі курсового проекту досліджено галузь з симуляції клітинних автоматів та отримано теоретичні відомості щодо реалізації гри «Життя».

Було зроблено аналіз різних версій гри, знайдених у відкритих джерелах.

Аналіз допоміг освідомити в якому напрямленні розвивати програму та які функції потрібно реалізувати в проекті.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

## РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1 Проектування загального алгоритму роботи програми

Після аналізу аналогічних програм, було отримано напрямок розробки програми. Гра буде графічною програмою.

Після запуску гри користувач вводить бажаний розмір поля, на якому будуть відображатися клітини, після створення поля, відкривається нове вікно, на якому знаходитиметься поле з клітинами та панель управління. Користувач може почати малювати олівцем, чи обрати фігури, серед яких є лінія, прямокутник та еліпс.

Після малювання, гравець розпочинає оновлення поля за допомогою кнопки «старт». Поле буде неперервно оновлюватися. Оновлення буде нескінченно тривати, доки користувач не натисне кнопку «стоп».

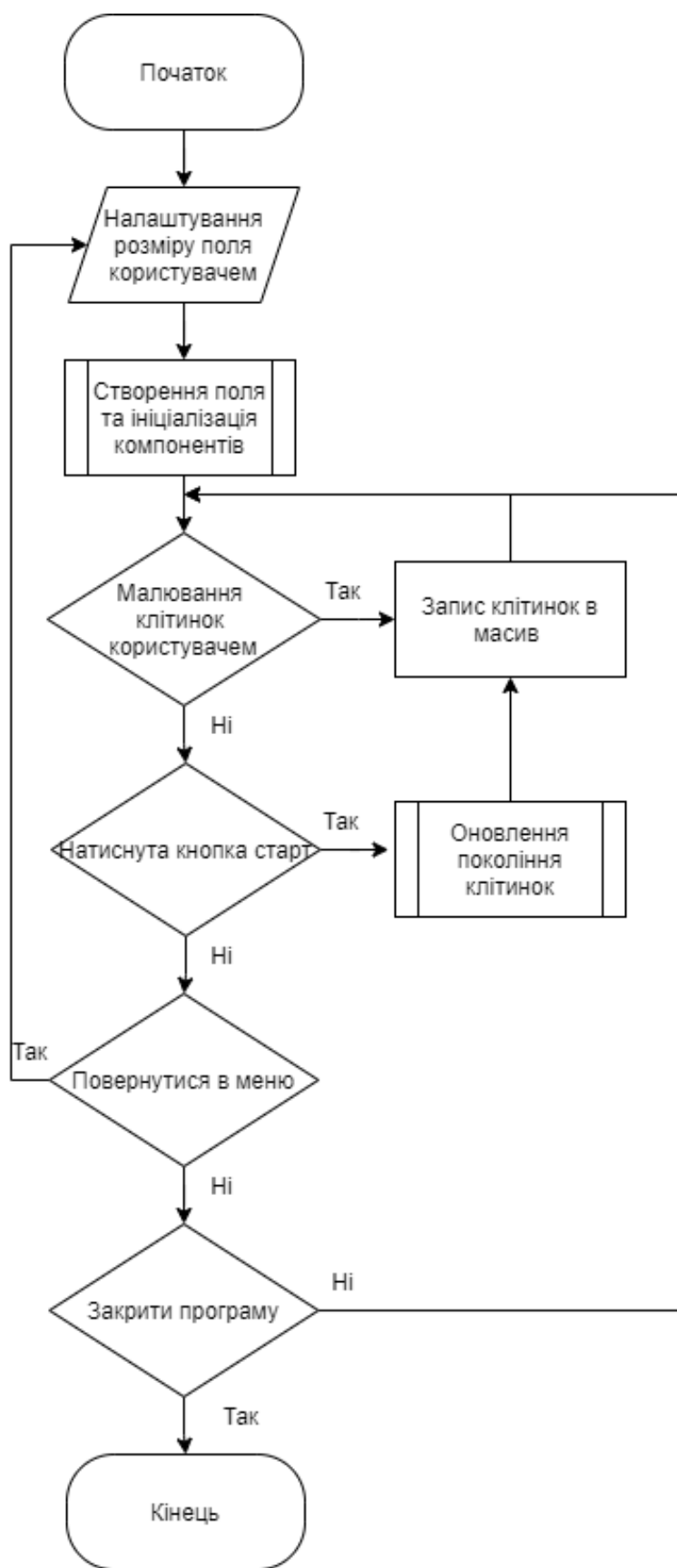
Завдяки зручному слайдеру, є можливість керувати затримкою між оновленням поля, щоб сповільнити, або пришвидшити симуляцію. Також, враховується розмір поля – при великому розмірі поля, встановлюється мінімальна затримка, менше якої користувач не зможе встановити. Це допомагає запобігати перевантаження пристрою та дозволяє грі стабільно працювати.

Щоб краще зрозуміти та роздивитися процеси та правила, які існують у грі «Життя» доступна кнопка «один крок», яка оновлює поле один раз.

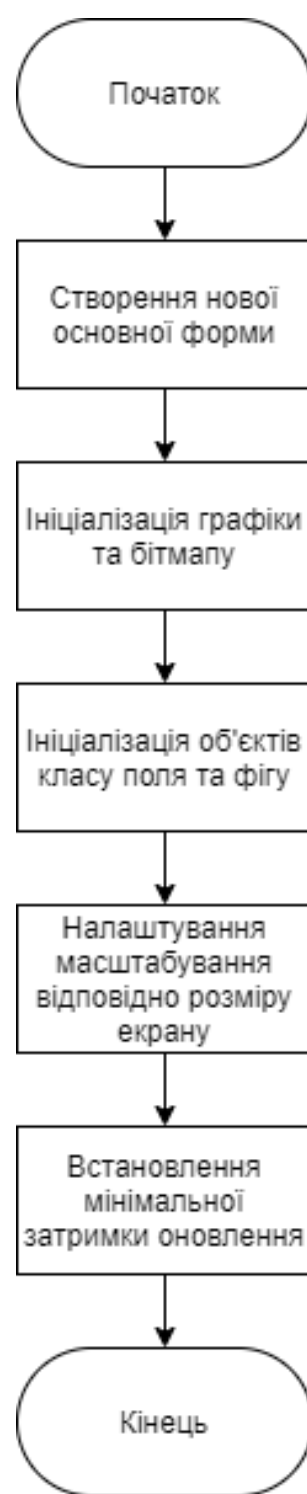
Якщо гравець забажає очистити поле від клітинок, він скористається кнопкою «очистити».

Якщо необхідно змінити розмір поля, можна повернутися до головного меню та почати гру заново.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		



а)



б)

Рис. 2.1.а) Блок-схема загального алгоритму програми

б) алгоритм створення поля та ініціалізації компонентів

		Магурін О. О.		
		Власенко О.В.		
Змн.	Арк.	№ докум.	Підпис	Дата



## 2.2 Розробка функціональних алгоритмів роботи програми

Після складання загального алгоритму роботи програми, є можливість проаналізувати його детальніше та розбити на більш малі та прості підпрограми. В результаті будуть отримані алгоритми, які більш детально описують роботу деякої частини програми, яке дозволить краще уявити про принцип роботи та потрібний функціонал та методи вирішення поставлених у технічному завданні задачах.

Поставлена технічна задача орієнтована на створення додатку з простим інтерфейсом, тому пріоритет в розробці надається зручності використання та технічним можливостям програми.

### Алгоритм створення поля та ініціалізації компонентів

Після того, як користувач ввів бажаний розмір поля та натиснув кнопку «створити нову гру», ініціалізується об'єкт класу, який має в собі інформацію про ширину та висоту поля та двовимірний масив, в який будуть записуватися стан клітинок, колір тла та координатної сітки.

При старті також ініціалізує графіку, тобто об'єкт Bitmap та Graphics, які дозволять робити зображення клітин на об'єкті ImageBox у Windows Forms.

Через те, що розмір поля для малювання, визначений користувачем може перевищувати чи бути меншим за розмір елемента на формі, під який він був виділений, при старті та зміні розміру вікна програма сама буде масштабувати тло з клітинами на весь екран, збільшуючи зображення в X разів.

$$[\text{коефіцієнт масштабування}] = [\text{ширина форми}] * 0.75 / [\text{ширина тла}];$$

Ширина форми домножається на 0.75, бо 75% ширини буде займати елемент ImageBox, на якому буде знаходитися тло, а інші 25% ширини – це панель управління. Одночасно й ініціалізується об'єкт класу фігур, для подальшого використання при малюванні. В залежності від встановленого розміру поля, встановлюється також мінімальна затримка в оновленні поля, щоб не перенавантажувати пристрій. Блок-схема наведена в рис. 2.1 б).

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

## Алгоритм оновлення покоління клітинок

Головний алгоритмом в цій програмі буде оновлення клітинок на полі відповідно правила B3/S23.

	A	B	C	D	E	F	G	H	I
1									
2									
3									
4									
5									
6									
7									
8									

Рис. 2.2. Приклад оновлення клітин

За цим правилом клітина народжується при рівно 3 сусідах, та виживає при рівно 2 або 3 сусідах, в інших випадках – помирає. Стан клітин зберігається в двовимірному масиві з розміром, вказаним користувачем.

Щоб оновити стан клітинок, треба перебрати увесь масив, та перевірити для кожної клітинки сусідні вісім клітин за правилом B3/S23. Перевірені клітини записуються у тимчасовий масив, коли перевіриться остання клітина в основному масиві, то в основний масив запише в себе данні з тимчасового, а тимчасовий масив видалиться.

Наприклад, на рис 2.2. досліджувана клітина – F3 (блакитний колір), то ж будуть перевірятися сусідні клітини E2, F2, G2, E3, G3, E4, F4, G4. Для клітини B2 – 3 сусіди, тож зараз вона записується у тимчасовий масив, і з'явиться у наступному поколінні.

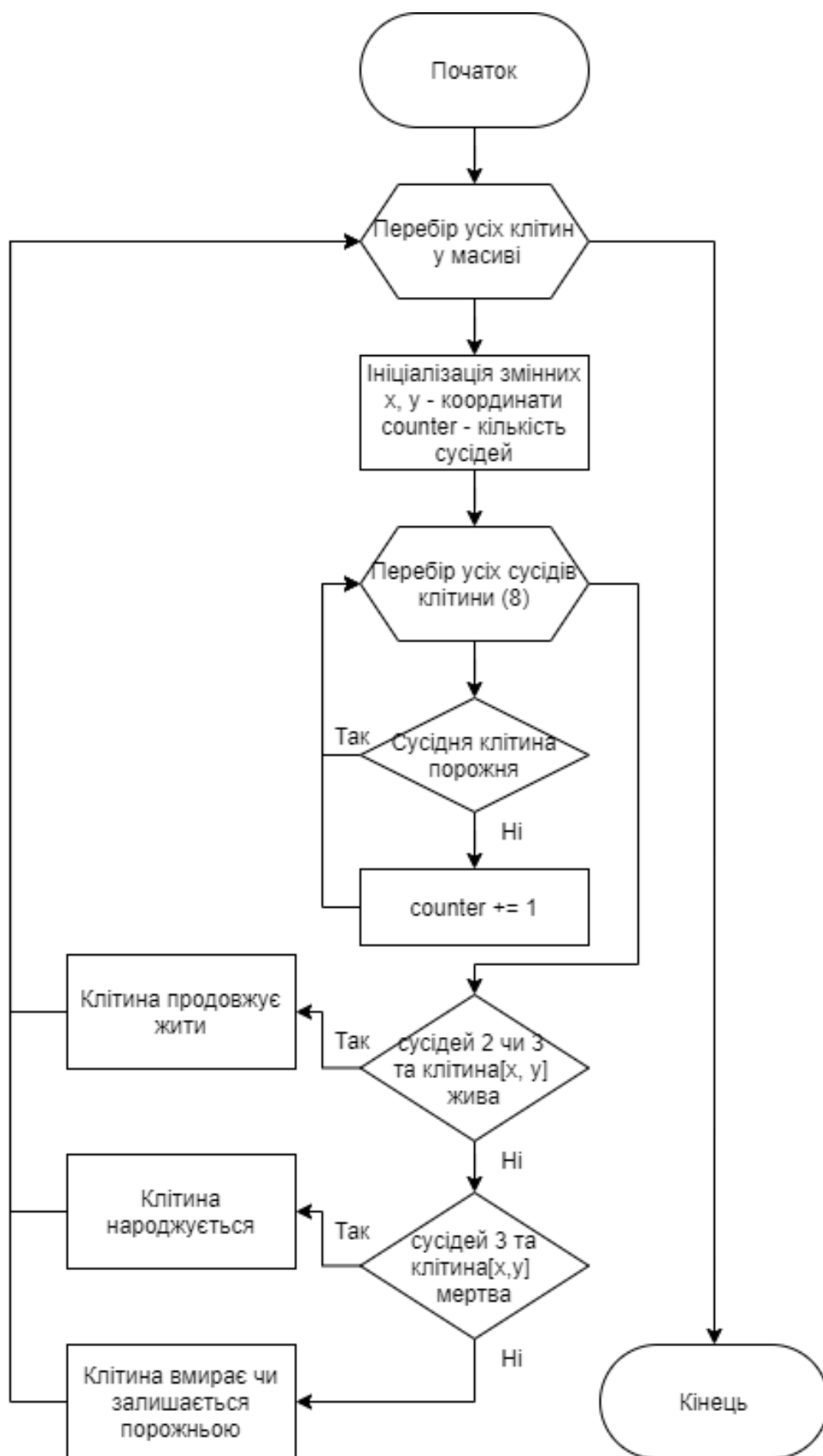


Рис. 2.3.Блок схема алгоритму оновлення клітинок

## 2.3 Розробка програмного забезпечення

Після завершення етапу проектування, можна перейти до розробки програмного забезпечення. Етап проектування спростив розробку додатку та допоміг отримати структуру проекту. Також стало зрозуміло основні класи та методи майбутньої програми. Розробка програмного забезпечення поділяється на два етапи: розробка інтерфейсу та функціональної частини.

### Розробка інтерфейсної частини

При запуску нас зустрічає меню, в якому є назва та заставка (анімована) гри, також можна буде вибрати налаштування розміру поля: будуть заготовлені варіанти (мале, середнє, велике і величезне) поле, або користувач може ввести розмір поля на власний розсуд.

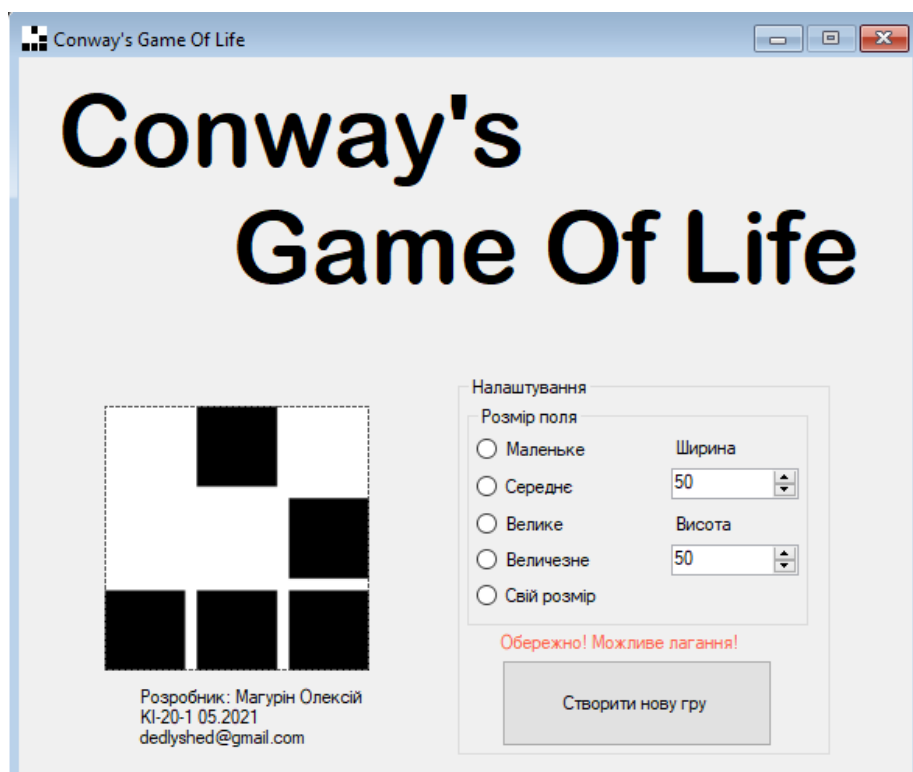


Рис. 2.4.Вигляд стартової форми у конструкторі

Після вибору розміру поля, користувач натискає кнопку «створити нову гру». Після цього меню зникає та відкривається нове вікно, в якому буде сама гра.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс головного вікна складатиметься з:

- Графічного поля, на якому проходитиме гра «Життя»:
  - Поле з координатною сіткою;
  - Полосок прокрутки поля;
- Меню управління оновленням поля:
  - Кнопки «старт-стоп», які керуватимуть процесом гри;
  - Слайдер, який встановлюватиме затримку оновлення поля;
- Меню малювання:
  - Кнопка «олівець» - при затисканні лівої кнопки миши ставить клітинки;
  - Кнопка «лінія» - малює суцільну лінію;
  - Кнопка «прямокутник» - малює прямокутник;
  - Кнопка «еліпс» - малює еліпс;
  - Кнопка «стерти», яка очищає все поле від клітинок;
- Інформаційна панель:
  - Інформація про положення курсора на полі по координатам;
  - Інформація про масштаб поля;

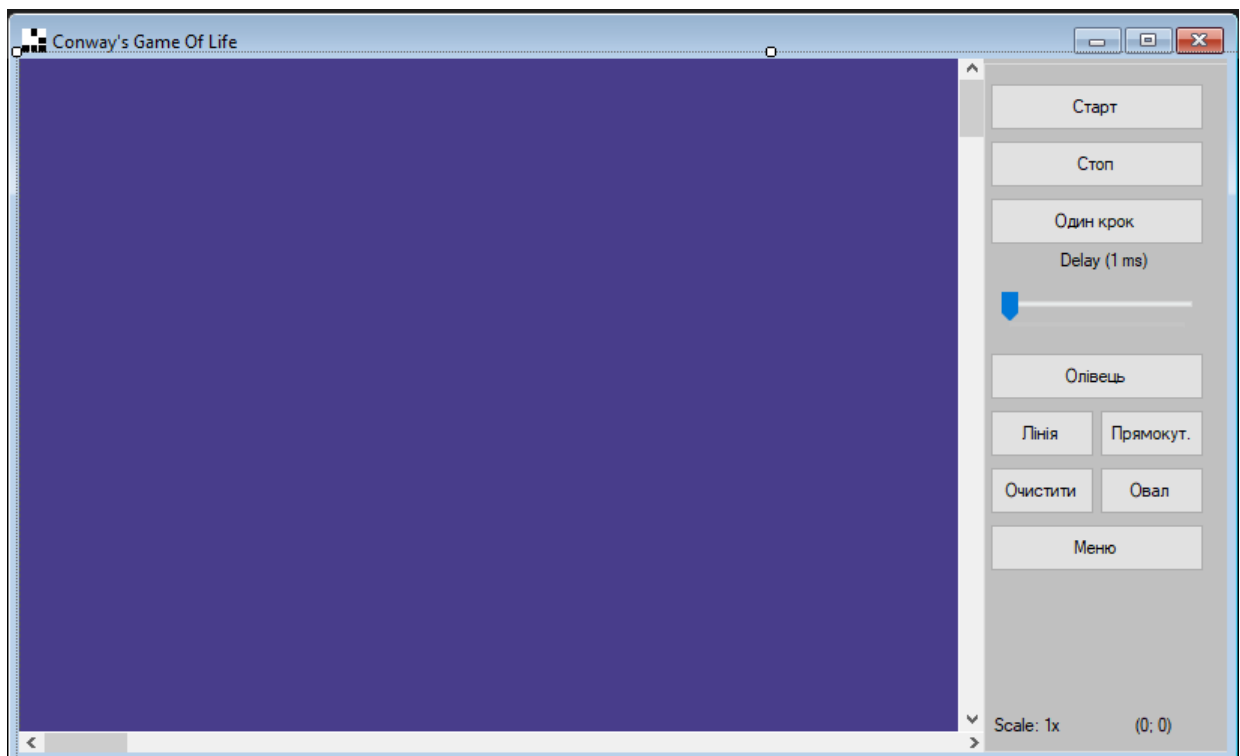


Рис. 2.5.Вигляд головної форми у конструкторі

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

## Розробка функціональної частини

Програма гра «Життя» складається з 6 класів.

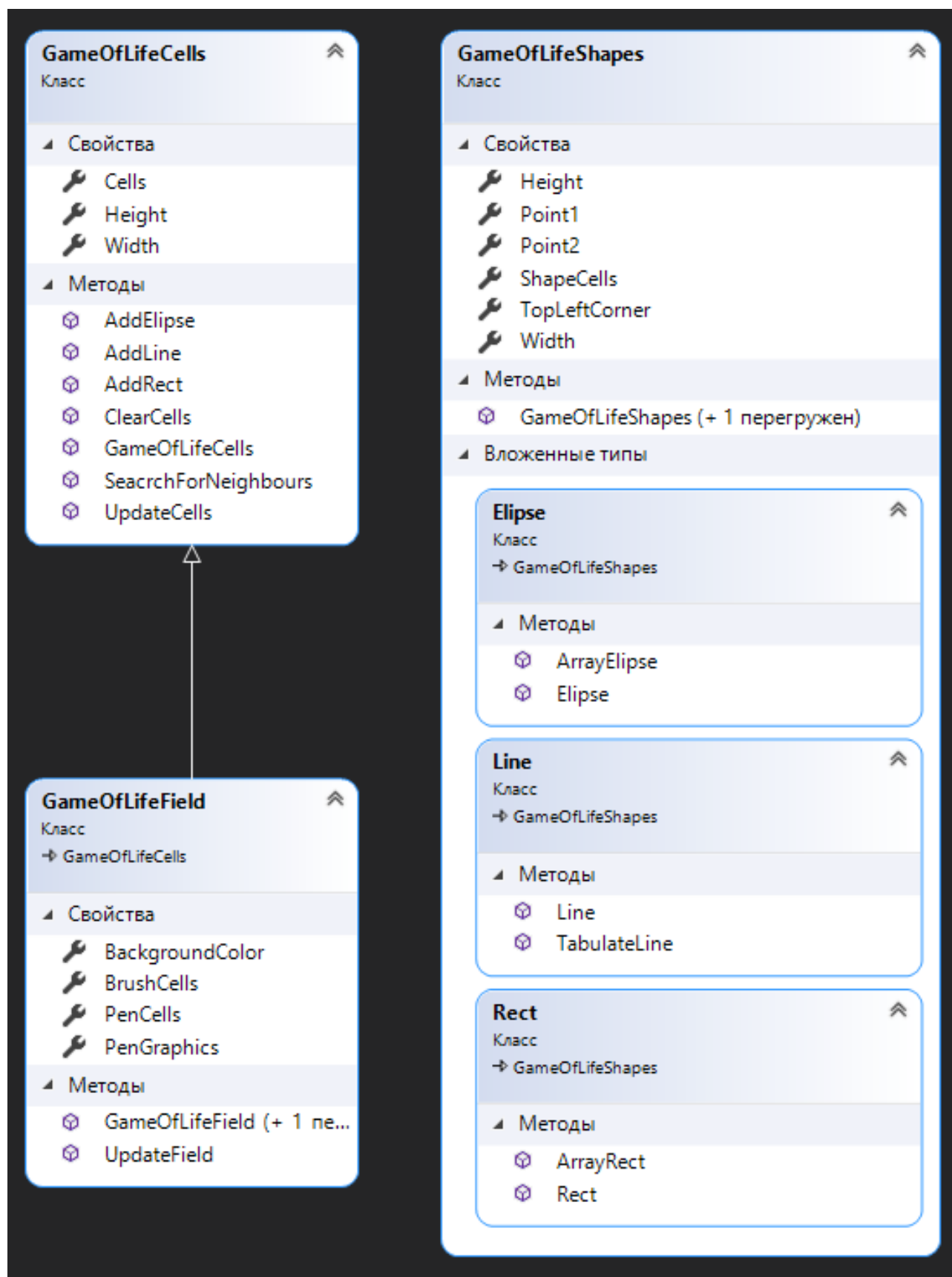


Рис. 2.6. Діаграма класів

<b>Класс GameOfLifeCells</b> - відповідає за основу поля	
<b>Властивості</b>	
Cells	Двовимірний bool масив, який зберігає стан клітин з поля
Height	Висота поля (масиву)
Width	Ширина поля (масиву)
<b>Методи</b>	
GameOfLifeCells	Ініціалізує об'єкт класу, створює двовимірний масив Cells, використовуючи аргументи (ширину та висоту)
AddLine	Приймає на вхід одномірний масив класу Point (System.Drawing) табульованої функції $y = k/x$ та додає точки в масив Cells. Використовуються методи класу Line, щоб отримати масив таб. функції
AddEllipse	Приймає на вхід двовимірний масив класу bool з точками еліпсу та додає точки в масив Cells. Використовуються методи класу Ellipse, щоб отримати масив з еліпсом.
AddRect	Приймає на вхід двовимірний масив класу bool з точками прямокутника та додає точки в масив Cells. Використовуються методи класу Rect, щоб отримати масив з еліпсом.
ClearCells	Очищає масив клітин (всі клітини - false)
UpdateCells	Оновлює поле клітин за правилом b3/s23
SearchForNeighbours	Відповідальний за пошук сусідніх живих клітин, використовується методом UpdateCells для кожної клітини масиву.

Метод UpdateCells:

```
public void UpdateCells()
{
    //тимчасовий масив
    bool[,] tmp = new bool[Width, Height];
    //продивляється всі клітини крім граничних
    //записує нове покоління клітин у тимчасовий масив
    for (int y = 1; y < Height - 1; y++)
    {
        for (int x = 1; x < Width - 1; x++)
        {
            int neighbours = SeacrchForNeighbours(x, y);
            if (neighbours == 2 && Cells[x, y] == true) tmp[x, y]
= true;
            else if (neighbours == 3) tmp[x, y] = true;
            if (neighbours < 2 || neighbours > 3) tmp[x, y] =
false;
        }
    }

    //з тимчасового масиву в основний
    for (int y = 1; y < Height - 1; y++)
    {
        for (int x = 1; x < Width -1; x++)
        {
            Cells[x, y] = tmp[x, y];
        }
    }
}
```

Метод SearchForNeighbours:

```
public int SeacrchForNeighbours(int x, int y)
{
    int neighbours = 0;
    //перевіряє 8 сусідніх клітин
    for (int i = -1; i<2; i++)
    {
        for (int j = -1; j<2; j++)
        {
            if (i == 0 && j == 0) continue;
            if (Cells[x+i, y+j] == true) neighbours++;
        }
    }
    return neighbours;
}
```



**Клас GameOfLifeField** - наслідує поля та методи класу GameOfLifeCells і існує для того, щоб взаємодіяти з графікою та бітмапом із Windows Forms.

#### Властивості

BackgroundColor	Колір тла поля
BrushCells	Пензлик для кольору клітин
PenGraphics	Ручка, що відповідає за колір ліній координатної сітки

#### Методи

GameOfLifeField	Ініціалізує об'єкт класу з кольорами поля
UpdateField	Відповідає за оновлення графіки на формі відповідно стану клітин на полі

**Клас GameOfLifeShapes** – клас, що є основою для інших фігур, від якого вони наслідують поля та методи.

#### Властивості

Height, Width	Ширина та висота фігури
P1, P2	Точки на полі, початку та кінця фігури
ShapeCells	Двовимірний bool масив зі станом клітин фігур (Rect, Elipse)
TopLeftCorner	Координати верхнього лівого кута фігури відносно основного поля

#### Методи

GameOfLifeShapes	Ініціалізує об'єкт класу
------------------	--------------------------

**Клас Line** - наслідує GameOfLifeShapes та потрібен для створення об'єкту класу лінії та табулювання її по координатам для подальшого додавання на поле методом AddLine класу GameOfLifeCells.

#### Методи

Line	Створює об'єкт класу лінії
TabulateLine	Табулює точки лінії і повертає одномірний Point масив

Метод TabulateLine:

```

public Point[] TabulateLine ()
{
    if ( Point1.X > Point2.X)
    {
        Point tmp = Point1;
        Point1 = Point2;
        Point2 = tmp;
    }
    Point difference = new Point(Point2.X - Point1.X, Point2.Y
- Point1.Y);
    Point[] linePoints = new Point[difference.X];
    for (int x = Point1.X, i = 0; x < Point2.X; x++, i++)
    {
        linePoints[i].X = x;
        linePoints[i].Y = (int)Math.Round((double)(Point1.Y +
difference.Y * ((float)(x - Point1.X) / difference.X)));
    }
    return linePoints;
}

```

**Клас Rect** - наслідує GameOfLifeShapes та потрібен для створення об'єкту класу прямокутника отримання масиву клітинок з прямокутником для подальшого додавання на поле методом AddRect класу GameOfLifeCells.

#### Методи

Rect	Створює об'єкт класу Rect
ArrayRect	Повертає масив з станом клітин у вигляді прямокутника

Метод ArrayRect:

```

public bool[,] ArrayRect ()
{
    ShapeCells = new bool[Width, Height];
    for (int i = 0; i < Width; i++)
    {
        ShapeCells[i, 0] = true;
        ShapeCells[i, Height-1] = true;
    }
    for (int j = 0; j < Height; j++)
    {
        ShapeCells[0, j] = true;
        ShapeCells[Width-1, j] = true;
    }
    return ShapeCells;
}

```

Клас `Ellipse` працює так само, як `Rect`, використовуються тригонометричні функції.

Метод `ArrayEllipse`:

```
public bool[,] ArrayEllipse ()
{
    ShapeCells = new bool[Width+1, Height+1];
    int x, y;
    for (float i = 0; i < 6.28; i+=0.01f)
    {
        x = (int)Math.Round(Width/2.0 - (Width / 2.0) *
Math.Cos(i));
        y = (int)Math.Round(Height/2.0 - (Height / 2.0) *
Math.Sin(i));
        ShapeCells[x, y] = true;
    }
    return ShapeCells;
}
```

Повний лістинг програми у (Додатку А).

## Висновки до другого розділу

У цьому розділі було спроектовано загальний алгоритм роботи програми, розглянуто основні принципи і основи методи на яких повинна працювати програма.

Розроблені основні алгоритми роботи програми, детальніше опрацьовано подробиці щодо роботи деяких методів та отримано орієнтоване представлення системи класів.

Розроблено інтерфейсну частину проекту. Завдяки зручним інструментам, наданим у Visual Studio, було оформлено форми у конструкторі Windows Forms. Завдяки цим можливостям, також легше писати та рефакторити код, що значно допомагає зберегти час при створенні функціональної частини проекту. Було створено основну систему класів та методів, які відповідають за роботу гри «Життя».

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

# РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

## 3.1 Опис роботи з програмним додатком

Після створення готової програми гри «Життя», необхідно розробити керівництво користувача до неї, щоб рівень входження до програми був нижчим та користувачі швидко та легко розібралися з інтерфейсом та головним функціоналом.

Для запуску програми треба завантажити ехе-файл з репозиторію за адресою: <https://gitlab.com/2020-2024/20-1/mahurin-oleksii/game-of-life-windows-forms/-/blob/master/GameOfLifeWF/bin/Debug/GameOfLifeWF.exe>

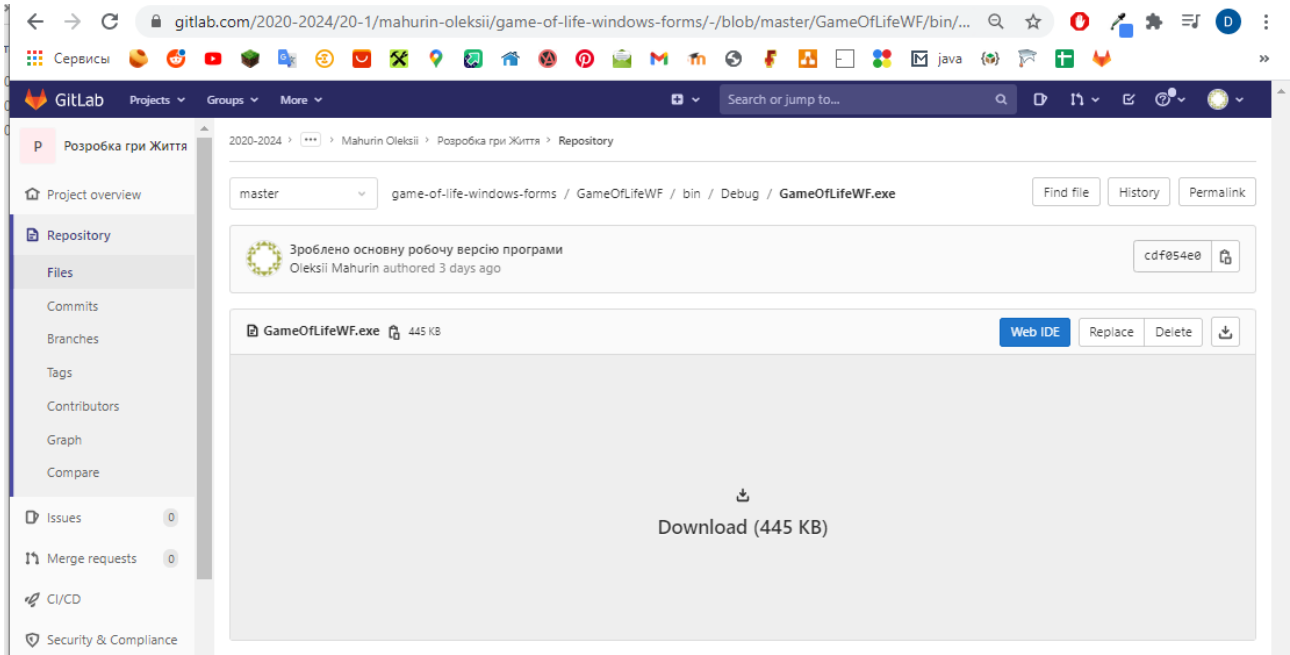


Рис. 3.1. Репозиторій з виконавчим файлом гри

Після завантаження гри «Життя» з репозиторію та запуску, відкриється перша форма – головне меню з назвою гри та анімованою заставкою, та початковими налаштуваннями. Користувач має змогу вибрати один із заготовлених розмірів поля, оптимізованих під співвідношення розміру вікна, що рекомендовано. Але є можливість обрати самостійно ширину та висоту поля для клітинок, від 50 до 1000. Отже, максимальне допустимий розмір поля: 1000x1000, що значить мільйон клітинок.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Хоча це можливо але не рекомендовано для використання, бо можуть бути проблеми з зависанням програми та перевантаження пристрою. Для попередження користувача про це, спливає підказка при виборі «Великого» або більше поля. Рекомендовані розміри поля – «Маленьке» та «Середнє».

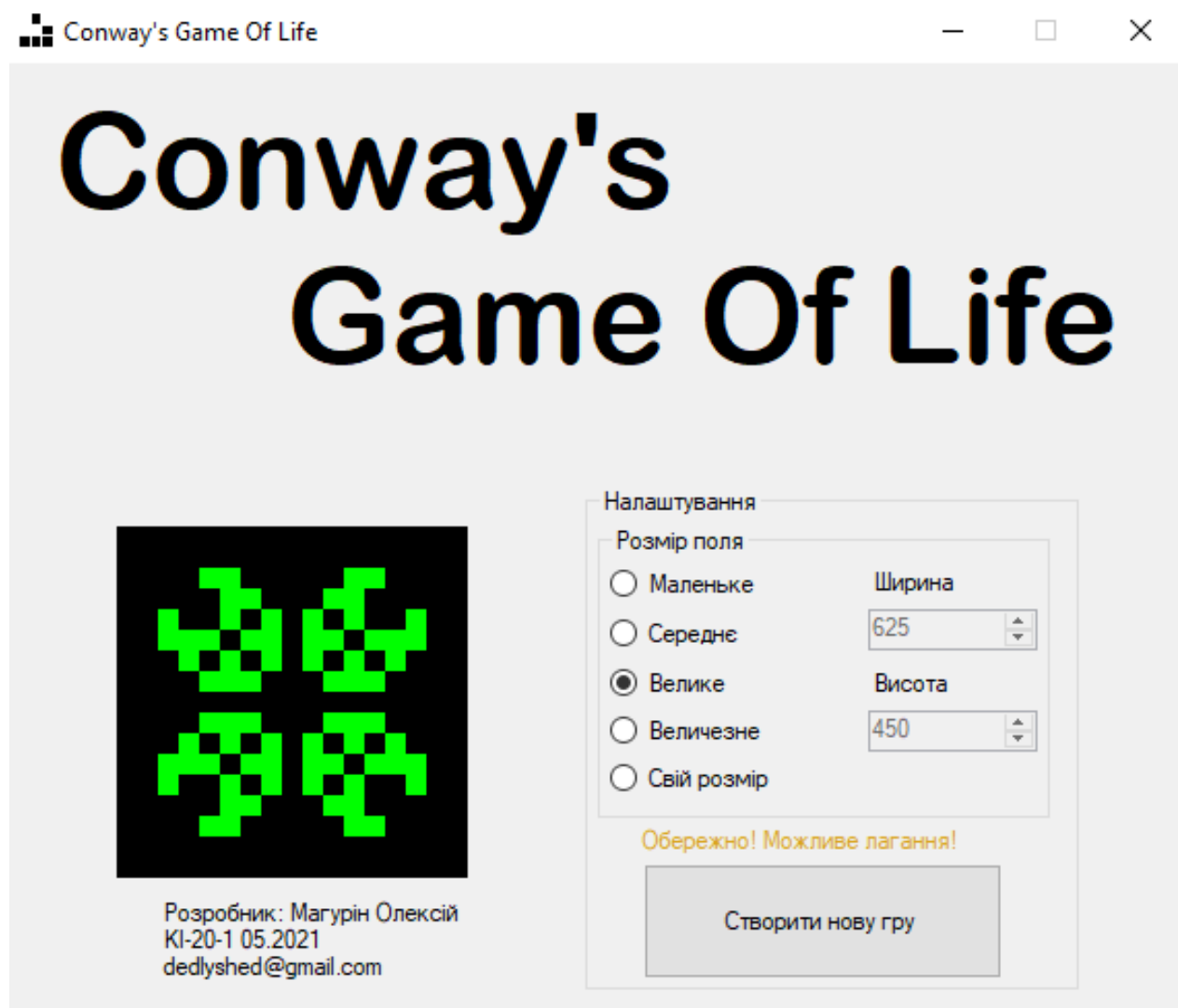


Рис. 3.2. Головне меню гри

Після вибору розміру поля, гравець натискає кнопку «Створити нову гру», після чого відкривається нове вікно, де і буде відбуватися симуляція.

Більшу частину вікна займає поле для малювання з координатною сіткою, який працює як тло, на ньому і буде відбуватися малювання. Праворуч буде панель управління, яка має всі потрібні інструменти для проведення експериментів у грі «Життя».

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

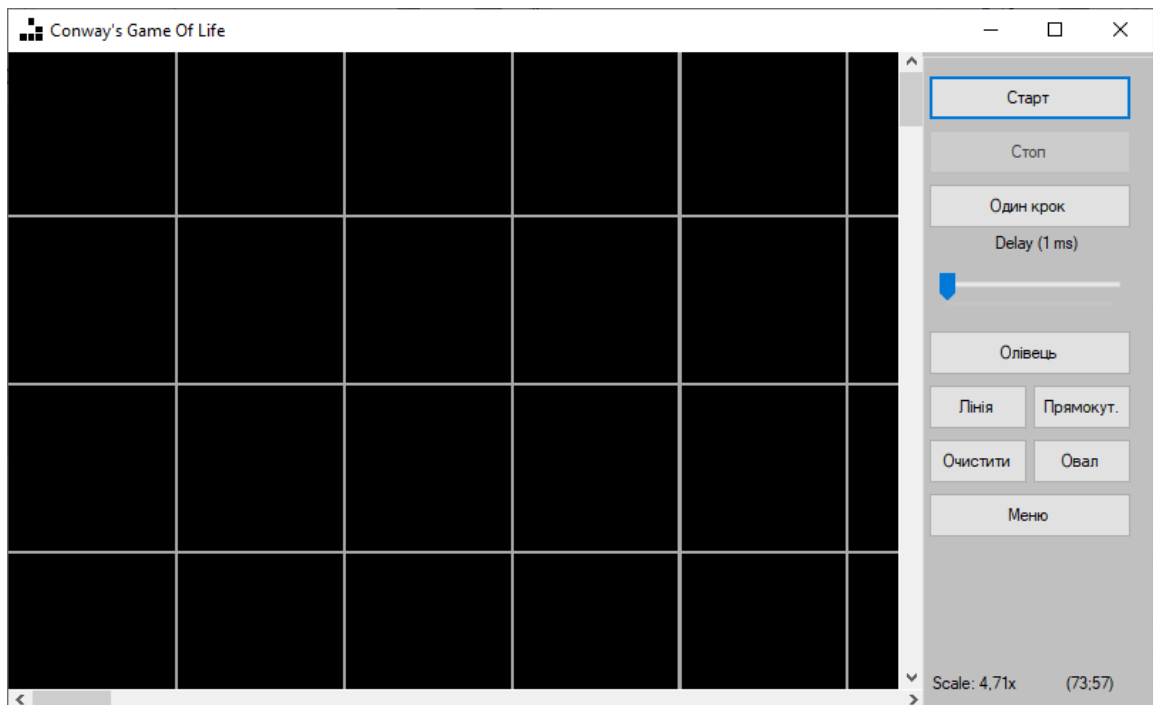


Рис. 3.3. Головне вікно гри «Життя»

У правому нижньому куті можна побачити поточне нахождение курсору на полі (на скріншоті курсор не відображається) і масштабування поля з клітинами відносно розмірам тла.

Щоб почати малювати, користувач має вибрати «Олівець» з панелі управління та почати водити з затиснутою лівою кнопкою миші по полю.

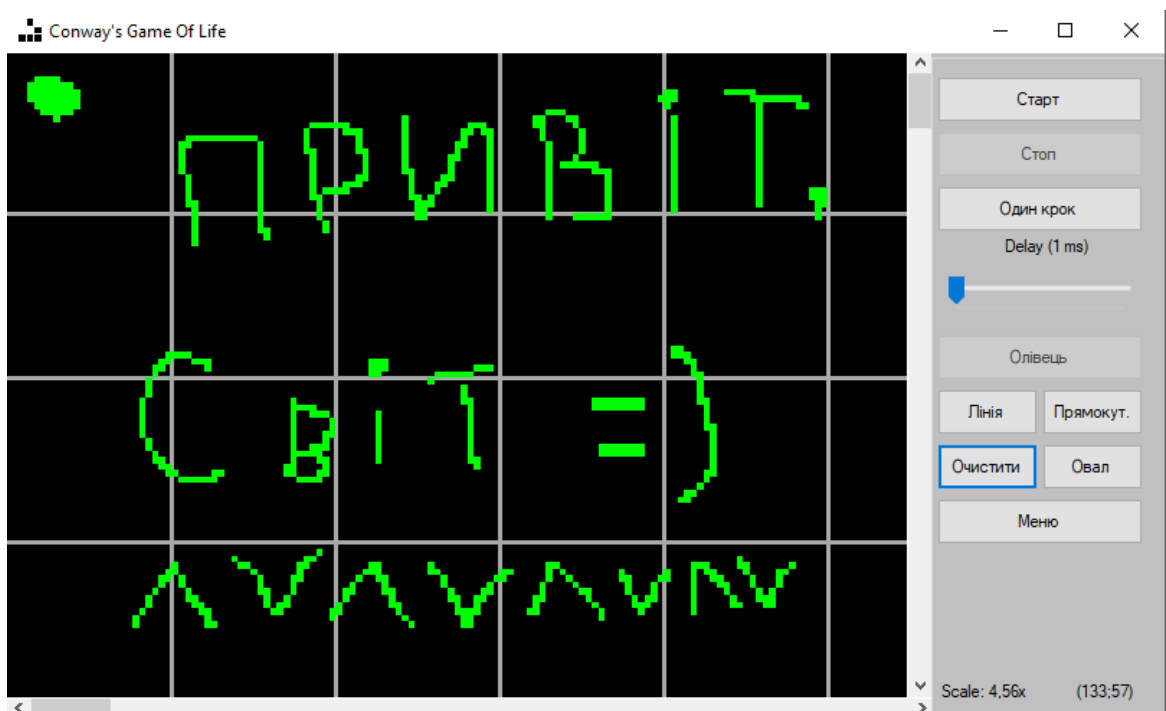


Рис. 3.4. Приклад використання олівця

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк. 30
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Щоб намалювати фігуру/лінію, треба вибрати її з панелі управління, затиснути ліву кнопку миші у верхній лівій вершині фігури, та відпустити у правій нижній. Якщо користувач захоче очистити поле, він може використати кнопку «Очистити».

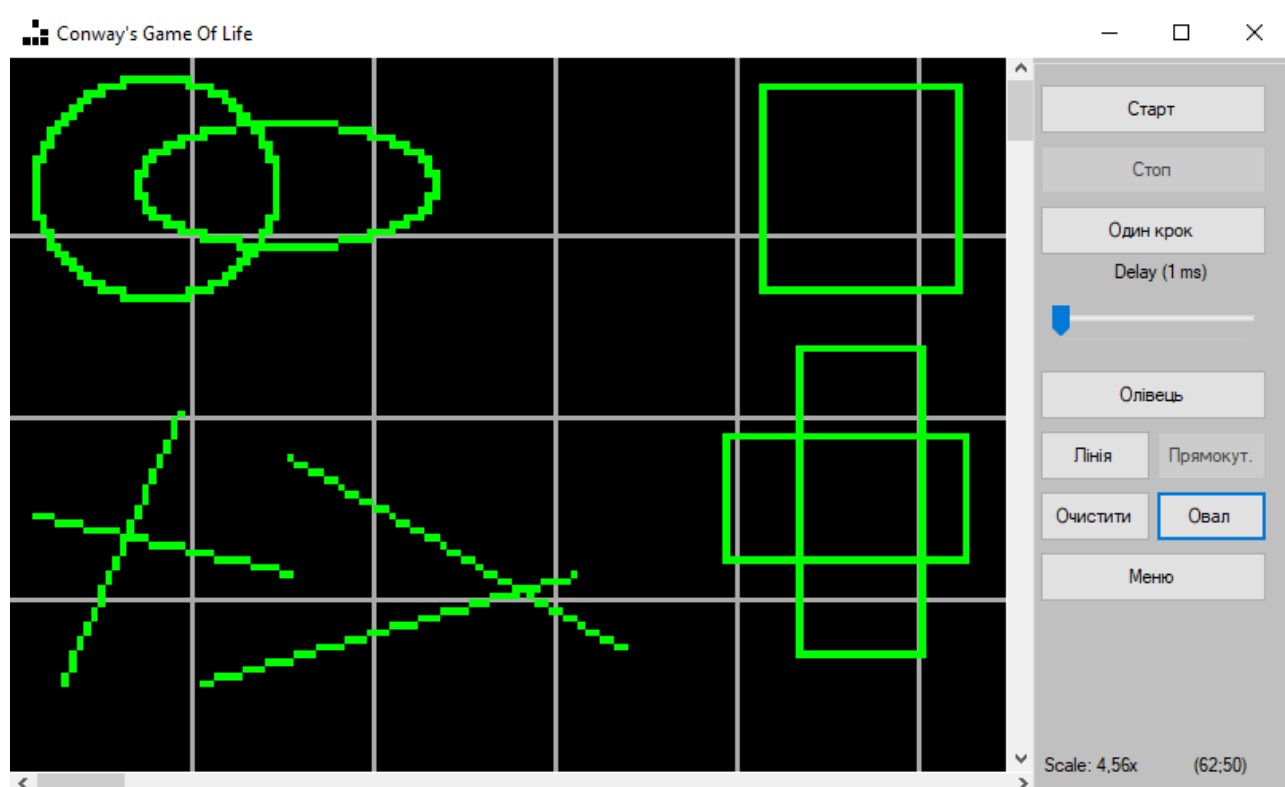


Рис. 3.5. Приклад малювання фігур та ліній

Після того, як користувач намалював бажаний малюнок чи фігури для симуляції, можна починати оновлення поля. Щоб краще роздивитися процеси симуляції, можна встановити більшу затримку, використовуючи слайдер.

Після натискання кнопки «Старт», гра безперервно починає оновлювати поле клітинок з затримкою, виставленою на слайдері. Слід зауважити, що малювання олівцем чи створення фігур на полі також можливе під час оновлення. Щоб зупинити оновлення, можна натиснути кнопку «Стоп». Також, доступна кнопка «Один крок», яка оновлює поле лише один раз. Це допомагає краще зрозуміти процеси та правила гри «Життя».

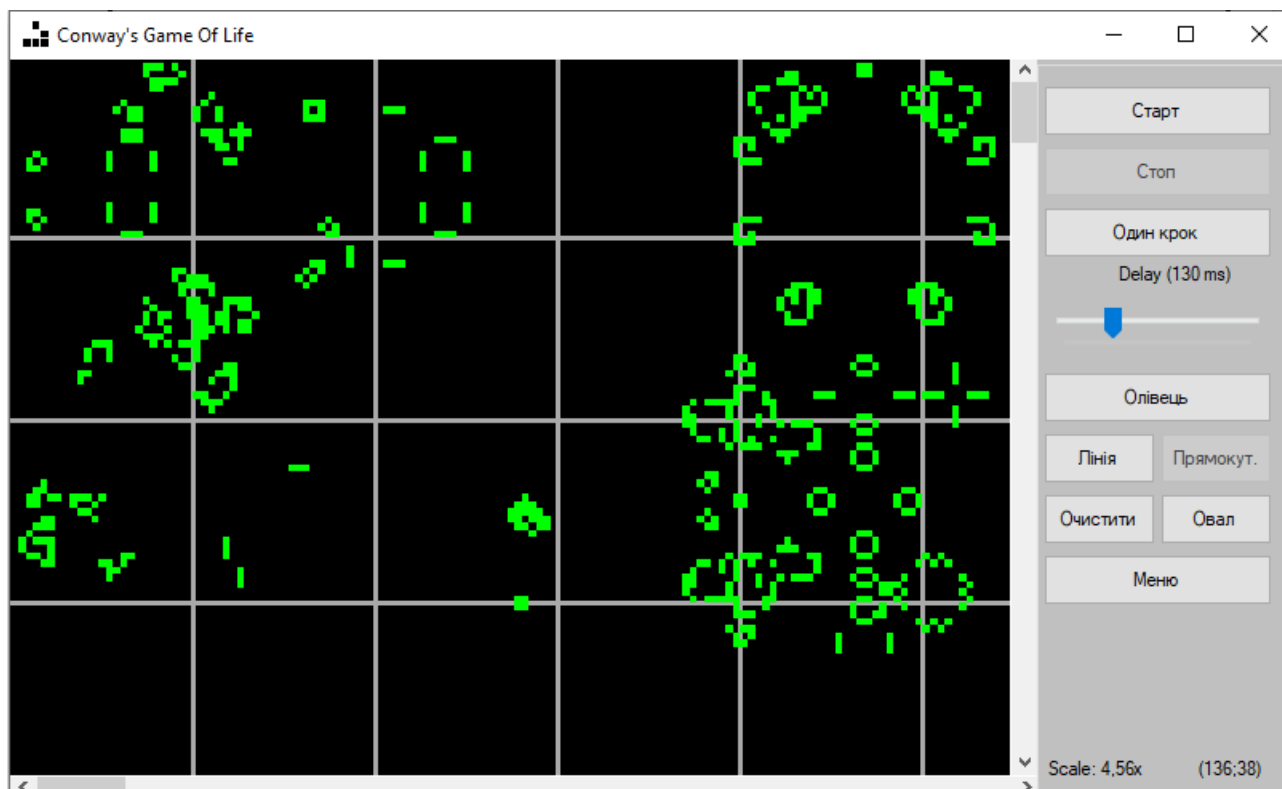


Рис. 3.6. Поле із рис.3.5. після деякої кількості ітерацій оновлення поля

На випадок, якщо гравець забажає змінити розмір поля, він може повернутися до меню та вибрати знову. Використовуючи вищепоказані дії, користувач може експериментувати з полем та клітинами, шукаючи нові цікаві фігури.

### 3.2 Тестування роботи програмного забезпечення

Перед тим як завершити розробку програми, слід виявити можливі помилки, які можуть використанні програми користувачем, проаналізувати їх природу та виправити їх.

На першому етапі слід виявити найбільш критичні програми, які призводять до крашу програми.

Перша помилка, визначена на етапі розробки – це виліт програми при некоректному зазначенні розміру поля. При вводі нульової чи від’ємної ширини чи висоти поля некоректно визначався масив, що призводило до вилітів програми.

Рішення – створення елемента numericUpDown для значення розміру поля та обмеження мінімального та максимального розміру поля – від 50 до 1000.



Друга помилка виникала при звертанні вікна. Через те, що масштабування поля на формі було відносно розміру вікна, при звертанні програми воно дорівнювало нулю, що призводило до некоректного завершення програми.

Рішення – зробити висновок, якщо розмір вікна дорівнював нулю, то масштаб не змінювати.

Третя важлива помилка стосується метода оновлення клітинок. При початку оновлення, на граничних клітинах, наприклад  $P1(0, 0)$ , при пошуку сусідніх клітин, програма завершувалася некоректно через те, що виходила за межі масиву, так як не існує, наприклад, клітини за індексом  $P2(-1, -1)$ .

Рішення – не оновлювати клітини, які граничать з будь-якою із сторін поля, на якість програми це не впливає.

На цьому, всі виявлені помилки щодо некоректного завершення програми завершилися. Але існувало ще декілька критичних проблем, через які неправильно працює гра.

Наприклад, четверта помилка – неправильне оновлення поля з клітинами. Хоча поле і оновлювалось, воно працювало не за законом гри «Життя». Проблема була в тому, що нове покоління клітин записувалося в основний масив, який в цей час оброблявся і оновлював клітини, тому при перерахунку сусідів, враховувалися клітини із майбутнього покоління, що не повинно було робити.

Рішення – створити тимчасовий масив, в який записувалися значення нового покоління, та після оновлення попереднього покоління, записувати в основний масив клітини з тимчасового. Проблема вирішена.

Помилка п'ята – при малюванні лінії, у яких кут з горизонтальною сіткою координат більше 45 градусів, виникав ефект переривчатої лінії.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

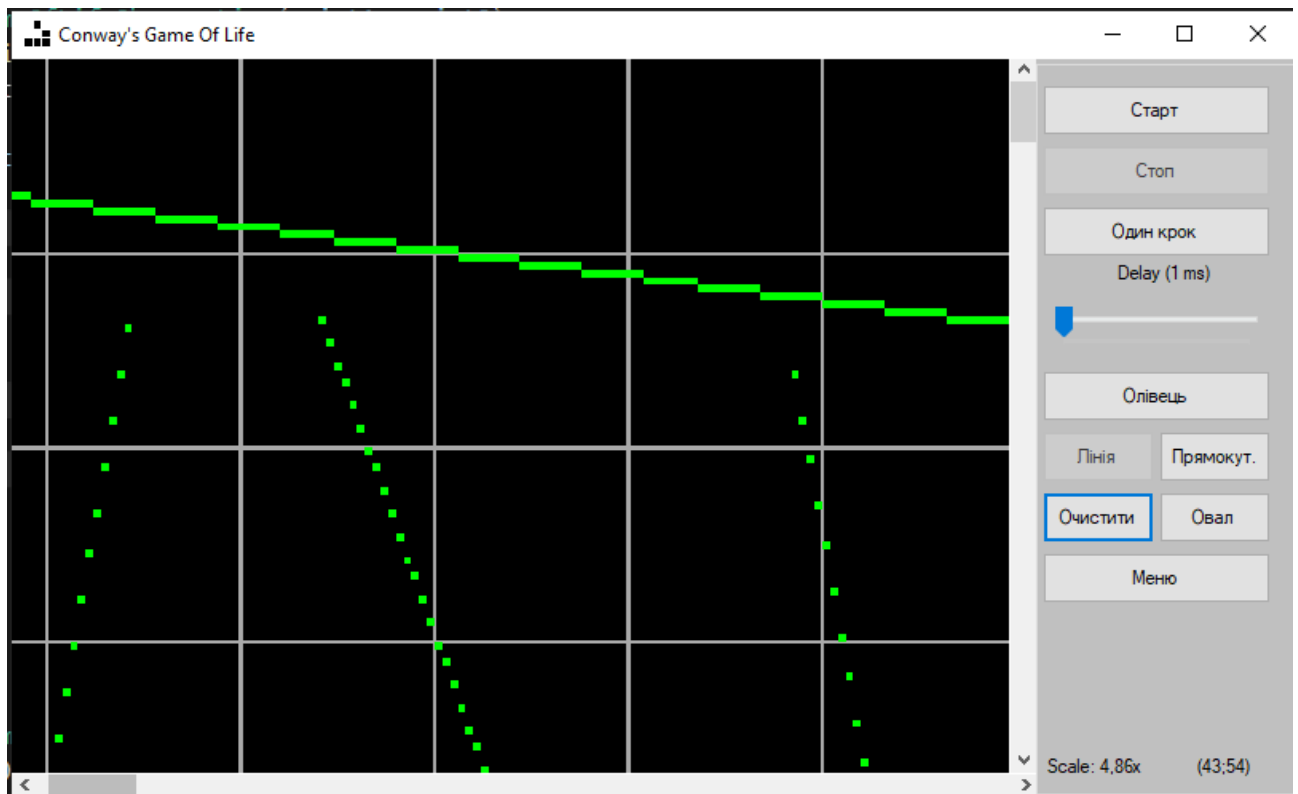


Рис. 3.7. Приклад проблеми з преривчатими лініями

Проблемою було те, що на вхід до малювання на полі лінія приходила табулювання відносно  $x$  з кроком в одиницю. Тому при куті менше 45 градусів лінія відмальовувалася нормально, а при більшому ставала переривчастою.

Рішення – а) проводити табулювання функції  $k/x$  ще відносно  $y$ , б) домальовувати кінці ліній завдяки двом циклам, вверх та униз. Найпростіше було реалізувати другий варіант.

```

ссылка: 1
public void AddLine(Point point1, Point point2)
{
    GameOfLifeShapes.Line line = new GameOfLifeShapes.Line(point1, point2);
    Point[] linePoints = line.TabulateLine();
    for (int i = 0; i < linePoints.Length; i++)
    {
        Cells[linePoints[i].X, linePoints[i].Y] = true;
        if (i + 1 == linePoints.Length) continue;
        for (int j = linePoints[i].Y; j < linePoints[i + 1].Y; j++)
        {
            Cells[linePoints[i].X, j] = true;
        }
        if (i - 1 < 0) continue;
        for (int k = linePoints[i].Y; k > linePoints[i + 1].Y; k--)
        {
            Cells[linePoints[i].X, k] = true;
        }
    }
}

```

Рис. 3.8. Вирішення проблеми з преривчатими лініями

### Висновки до третього розділу

В цьому розділі було зроблено опис роботи з програмним додатком та його тестування, що повинно допомогти кінцевому користувачу зрозуміти, як користуватися даною програмою та про основні можливості додатка.

Також було пророблено роботу щодо усунення помилок та опис вирішених при розробці помилок, які могли вивести з ладу програму. Основні виявлені помилки вирішені та не будуть заважати в подальшій роботі.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

## ВИСНОВКИ

В результаті виконання цього проекту було отримано знання щодо природи клітинних автоматів – дискретної моделі в математиці, на прикладі вигаданої Джоном Хортоном Конвесем гри «Життя» - двомірна клітинно-автоматна модель з двома станами клітин. Незважаючи на простоту роботи правил цієї гри, з неї випливає безліч цікавих ситуацій, від повного хаосу, до стабільних фігур, або до нескінченних періодичних фігур, які змінюються. Відомо, що в природі існують приклади клітинних автоматів, такі як візерунки на мушлях деяких видів молюсків, або деякі хімічні реакції.

Орієнтуючись на існуючі доступні версії гри «Життя», було прийнято рішення розробити свій варіант, з пріоритетом на зручне малювання та створення фігур, масштабуванням та зручною та швидкою симуляцію та оновлення поля.

Розроблено основні методи роботи програми у вигляді блок-схем, щоб краще розуміти існуючу задачу. Це значно спростило створення методів та класів при створенні програмного коду.

У процесі реалізації даної гри з простими правилами виникли деякі складні задачі, наприклад – автоматичне масштабування вікна та поля з клітинами. Через те, що малювання фігур треба було робити з записом їх у масив клітин, треба було розробити свої методи малювання та табулювання фігур, використовуючи дві точки, як початкові данні.

Після розробки програми, вирішення помилок та створення опису роботи з програмним додатком, отримано готовий функціонуючу програму, з кращою реалізацією, ніж деякі існуючі аналоги.

Перш за все, основна мета цього курсового проекту – закріпити знання по використанню об'єктно-орієнтованого програмування на мові C# на практиці, використовуючи графічні можливості Windows Forms та середу розробки Microsoft Visual Studio. І, на мою думку, цей курсовий проект виконав свою мету.

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [illegible]

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Власенко О.В.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

# ДОДАТКИ

		Магурін О. О.			ДУ «Житомирська політехніка».21.123.14.000 - ПЗ	Арк.
		Зласенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

## Лістинг програми

## GameOfLifeCells.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace GameOfLifeWF
{
    class GameOfLifeCells
    {
        public int Width { get; set; }
        public int Height { get; set; }
        public bool[,] Cells { get; set; }
        public GameOfLifeCells (int width, int height) { Width = width; Height = height; Cells =
new bool[Width, Height]; }

        public int SeacrchForNeighbours(int x, int y)
        {
            int neighbours = 0;
            for (int i = -1; i<2; i++)
            {
                for (int j = -1; j<2; j++)
                {
                    if (i == 0 && j == 0) continue;
                    if (Cells[x+i, y+j] == true) neighbours++;
                }
            }
            return neighbours;
        }

        public void ClearCells()
        {
            Cells = new bool[Width, Height];
        }

        public void UpdateCells()
        {
            bool[,] tmp = new bool[Width, Height];
            for (int y = 1; y < Height - 1; y++)
            {
                for (int x = 1; x < Width - 1; x++)
                {
                    int neighbours = SeacrchForNeighbours(x, y);
                    if (neighbours == 2 && Cells[x, y] == true) tmp[x, y] = true;
                    else if (neighbours == 3) tmp[x, y] = true;
                    if (neighbours < 2 || neighbours > 3) tmp[x, y] = false;
                }
            }
            for (int y = 1; y < Height - 1; y++)
            {
                for (int x = 1; x < Width -1; x++)
                {
                    Cells[x, y] = tmp[x, y];
                }
            }
        }

        public void AddLine(Point point1, Point point2)
        {
            GameOfLifeShapes.Line line = new GameOfLifeShapes.Line(point1, point2);
        }
    }
}
```

```

Point[] linePoints = line.TabulateLine();
for (int i = 0; i < linePoints.Length; i++)
{
    Cells[linePoints[i].X, linePoints[i].Y] = true;
    if (i + 1 == linePoints.Length) continue;
    for (int j = linePoints[i].Y; j < linePoints[i + 1].Y; j++)
    {
        Cells[linePoints[i].X, j] = true;
    }
    if (i - 1 < 0) continue;
    for (int k = linePoints[i].Y; k > linePoints[i + 1].Y; k--)
    {
        Cells[linePoints[i].X, k] = true;
    }
}

public void AddRect(Point point1, Point point2)
{
    GameOfLifeShapes.Rect rect = new GameOfLifeShapes.Rect(point1, point2);
    bool[,] arrayRect = rect.ArrayRect();

    for (int i = 0; i < rect.Width; i++)
    {
        for (int j = 0; j < rect.Height; j++)
        {
            if (arrayRect[i, j]) Cells[rect.TopLeftCorner.X + i, rect.TopLeftCorner.Y +
j] = true;
        }
    }
}

public void AddEllipse(Point point1, Point point2)
{
    GameOfLifeShapes.Ellipse ellipse = new GameOfLifeShapes.Ellipse(point1, point2);
    bool[,] arrayEllipse = ellipse.ArrayEllipse();

    for (int i = 0; i < ellipse.Width+1; i++)
    {
        for (int j = 0; j < ellipse.Height+1; j++)
        {
            if (arrayEllipse[i, j]) Cells[ellipse.TopLeftCorner.X + i,
ellipse.TopLeftCorner.Y + j] = true;
        }
    }
}
}
}

```

## GameOfLifeField.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace GameOfLifeWF
{
    class GameOfLifeField : GameOfLifeCells
    {
        public Pen PenCells { get; set; }
        public Brush BrushCells { get; set; }
        public Pen PenGraphics { get; set; }
        public Color BackgroundColor { get; set; }
        public GameOfLifeField(int width, int height) : base(width, height) { BrushCells =
Brushes.Lime; PenGraphics = new Pen(Color.DarkGray, 0.5f); }
    }
}

```



```

        public GameOfLifeField(int width, int height, Brush brushCells, Pen penGraphics) :
base(width, height) { BrushCells = brushCells; PenGraphics = penGraphics; }
        public void UpdateField(Graphics g)
        {
            g.Clear(Color.Black);
            for (int i = 25; i < Width; i += 25){
                g.DrawLine(PenGraphics, i, 0, i, Height);
            }
            for (int j = 25; j < Height; j+= 25)
            {
                g.DrawLine(PenGraphics, 0, j, Width, j);
            }
            for (int x = 1; x < Width - 1; x++)
            {
                for (int y = 1; y < Height - 1; y++)
                {
                    if (Cells[x, y]) g.FillRectangle(BrushCells, x, y, 1, 1);
                }
            }
        }
    }
}

```

## GameOfLifeShapes.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace GameOfLifeWF
{
    class GameOfLifeShapes
    {
        public Point Point1 { get; set; }
        public Point Point2 { get; set; }
        public Point TopLeftCorner { get; set; }
        public int Width { get; set; }
        public int Height { get; set; }
        public bool[,] ShapeCells { get; set; }
        public GameOfLifeShapes() { }
        public GameOfLifeShapes (Point point1, Point point2)
        {
            Point1 = point1;
            Point2 = point2;
            Width = Math.Abs(Point1.X - Point2.X)+1;
            Height = Math.Abs(Point1.Y - Point2.Y)+1;

            int x; int y;
            if (point1.X < point2.X) x = point1.X;
            else x = point2.X;
            if (point1.Y < point2.Y) y = point1.Y;
            else y = point2.Y;
            TopLeftCorner = new Point(x, y);
        }

        public class Line : GameOfLifeShapes
        {
            public Line (Point point1, Point point2) : base (point1, point2) {}
            public Point[] TabulateLine ()
            {
                if ( Point1.X > Point2.X)
                {
                    Point tmp = Point1;
                    Point1 = Point2;
                    Point2 = tmp;
                }
                Point difference = new Point(Point2.X - Point1.X, Point2.Y - Point1.Y);
            }
        }
    }
}

```

```

        Point[] linePoints = new Point[difference.X];
        for (int x = Point1.X, i = 0; x < Point2.X; x++, i++)
        {
            linePoints[i].X = x;
            linePoints[i].Y = (int)Math.Round((double)(Point1.Y + difference.Y *
((float)(x - Point1.X) / difference.X)));
        }
        return linePoints;
    }
}
public class Rect : GameOfLifeShapes
{
    public Rect(Point point1, Point point2) : base(point1, point2) { }
    public bool[,] ArrayRect ()
    {
        ShapeCells = new bool[Width, Height];
        for (int i = 0; i < Width; i++)
        {
            ShapeCells[i, 0] = true;
            ShapeCells[i, Height-1] = true;
        }
        for (int j = 0; j < Height; j++)
        {
            ShapeCells[0, j] = true;
            ShapeCells[Width-1, j] = true;
        }
        return ShapeCells;
    }
}
public class Elipse : GameOfLifeShapes
{
    public Elipse (Point point1, Point point2) : base(point1, point2) { }
    public bool[,] ArrayElipse ()
    {
        ShapeCells = new bool[Width+1, Height+1];
        int x, y;
        for (float i = 0; i < 6.28; i+=0.01f)
        {
            x = (int)Math.Round(Width/2.0 - (Width / 2.0) * Math.Cos(i));
            y = (int)Math.Round(Height/2.0 - (Height / 2.0) * Math.Sin(i));
            ShapeCells[x, y] = true;
        }
        return ShapeCells;
    }
}
}
}

```

## MenuForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GameOfLifeWF
{
    public partial class MenuForm : Form
    {
        Graphics gLogo;
        Bitmap canvasLogo;
        GameOfLifeField logoAnimated;
        byte minDelay = 1;
    }
}

```

```

public MenuForm()
{
    InitializeComponent();
    CreateLogo();
    AnimationLogo();
}

private void buttonStart_Click(object sender, EventArgs e)
{
    if (numericUpDownWidth.Value == 0 || numericUpDownHeight.Value == 0)
    {
        MessageBox.Show(
            "Field size not specified. Choose field size in menu.",
            "Field size not specified",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information
        );
        return;
    }
    int width = (int)numericUpDownWidth.Value;
    int height = (int)numericUpDownHeight.Value;
    Game game = new Game(width, height, minDelay);
    game.Show();
    this.Hide();
}

private void CreateLogo()
{
    logoAnimated = new GameOfLifeField(17, 17);
    byte[,] tmp =
    {
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0 },
        {0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0 },
        {0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0 },
        {0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 },
        {0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0 },
        {0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0 },
        {0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    };
    for (int i = 0; i < 17; i++)
    {
        for (int j = 0; j < 17; j++)
        {
            if (tmp[i, j] == 1) logoAnimated.Cells[i, j] = true;
        }
    }
    canvasLogo = new Bitmap(logoAnimated.Width*10, logoAnimated.Height*10);
    gLogo = Graphics.FromImage(canvasLogo);
    gLogo.ScaleTransform(10f, 10f);
    logoAnimated.UpdateField(gLogo);
    pictureBoxLogo.Image = canvasLogo;
}

private async void AnimationLogo()
{
    for (int i = 0; i < 1000; i++)
    {
        logoAnimated.UpdateCells();
        logoAnimated.UpdateField(gLogo);
    }
}

```

```

        pictureBoxLogo.Image = canvasLogo;
        await Task.Delay(400);
    }
}

private void radioButtonSmall_CheckedChanged(object sender, EventArgs e)
{
    numericUpDownWidth.Value = 175;
    numericUpDownHeight.Value = 100;
    minDelay = 1;
}

private void radioButtonMedium_CheckedChanged(object sender, EventArgs e)
{
    numericUpDownWidth.Value = 350;
    numericUpDownHeight.Value = 200;
    minDelay = 10;
}

private void radioButtonBig_CheckedChanged(object sender, EventArgs e)
{
    numericUpDownWidth.Value = 625;
    numericUpDownHeight.Value = 450;
    minDelay = 20;
}

private void radioButtonLarge_CheckedChanged(object sender, EventArgs e)
{
    numericUpDownWidth.Value = 875;
    numericUpDownHeight.Value = 630;
    minDelay = 50;
}

private void radioButtonCustom_CheckedChanged(object sender, EventArgs e)
{
    numericUpDownWidth.Enabled = !numericUpDownWidth.Enabled;
    numericUpDownHeight.Enabled = !numericUpDownHeight.Enabled;
}

private void numericUpDownWidth_ValueChanged(object sender, EventArgs e) { LagWarning();
}

private void numericUpDownHeight_ValueChanged(object sender, EventArgs e) { LagWarning();
}

private void LagWarning()
{
    int cellsAmount = (int)(numericUpDownHeight.Value * numericUpDownWidth.Value);
    if (cellsAmount > 500000) labelLagWarning.ForeColor = Color.Red;
    else labelLagWarning.ForeColor = Color.Goldenrod;
    labelLagWarning.Visible = true;
    if (cellsAmount < 250000) labelLagWarning.Visible = false;
}
}
}

```

## Game.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GameOfLifeWF
{

```

```

public partial class Game : Form
{
    //ініціалізація поля та графіки
    GameOfLifeField field;
    GameOfLifeShapes shape;
    Point canvasPoint = new Point();
    Graphics g;
    Bitmap canvas;
    //коеф. масштабування
    float scale = 1;
    //потрібно для асинхронної роботи гри
    bool stopgame = false;
    //режими малювання
    bool shapemode = false;
    bool linemode = false;
    bool rectmode = false;
    bool elipsemode = false;
    bool isdrawing = false;

    public Game(int width, int height, byte minDelay)
    {
        InitializeComponent();
        canvasBox.MouseWheel += CanvasBox_MouseWheel;
        field = new GameOfLifeField(width, height);
        GameOfLifeShapes shape = new GameOfLifeShapes();
        canvas = new Bitmap(field.Width, field.Height);
        g = Graphics.FromImage(canvas);
        Pen pen = new Pen(Color.Red, 10);
        canvasBox.Image = canvas;
        //0.76219 = 76.219% - це відсоток ширини поля з клітинками від всієї ширини вікна

        //початкове масштабування, в залежності від розміру поля
        scale = (float)Math.Round((this.Width * 0.75 / (field.Width)), 2);
        ScaleField();
        //встановлення мінімальної затримки між оновленням поля
        trackBarDelay.Minimum = minDelay;
        LabelDelay.Text = $"Delay ({trackBarDelay.Value} ms)";
    }

    //асинхронний метод нескінченно оновлює поле, доки користувач не натисне кнопку "стоп"
    private async void buttonStart_Click(object sender, EventArgs e)
    {
        buttonStart.Enabled = false;
        buttonStop.Enabled = true;
        buttonOneStep.Enabled = false;
        stopgame = false;
        while(true)
        {
            if (stopgame == true) break;
            field.UpdateCells();
            field.UpdateField(g);
            canvasBox.Image = canvas;
            await Task.Delay(trackBarDelay.Value);
        }
    }
    //зупиняє оновлення поля
    private void buttonStop_Click(object sender, EventArgs e)
    {
        buttonStart.Enabled = true;
        buttonStop.Enabled = false;
        buttonOneStep.Enabled = true;
        stopgame = true;
    }

    private void buttonOneStep_Click(object sender, EventArgs e)
    {
        field.UpdateCells();
        field.UpdateField(g);
        canvasBox.Image = canvas;
    }
}

```

форми

```

private void Form1_Resize(object sender, EventArgs e)
{
    canvasBox.Width = 10000;
    canvasBox.Height = 10000;
    scale = (float)(this.Width * (1 - 120 / this.Width) / (field.Width));
    ScaleField();
}

private void CanvasBox_MouseWheel(object sender, MouseEventArgs e)
{
    if (e.Delta > 0) { scale += 0.15f; }
    else if (e.Delta < 0) { scale -= 0.15f; }
    if (scale <= 0) scale = 0.05f;
    else if (scale*field.Width >= 5500) scale -= 0.15f;
    ScaleField();
}

private void ScaleField()
{
    scale = (float)Math.Round(scale, 2);
    if (scale == 0) scale = 1;
    canvas = new Bitmap((int)(field.Width * scale), (int)(field.Height * scale));
    g = Graphics.FromImage(canvas);
    g.ScaleTransform(scale, scale);
    field.UpdateField(g);
    canvasBox.Image = canvas;
    labelScale.Text = "Scale: " + scale + "x";
}

private void canvasBox_MouseMove(object sender, MouseEventArgs e)
{
    canvasPoint.X = (int)(e.X / scale);
    canvasPoint.Y = (int)(e.Y / scale);
    labelCoords.Text = "(" + canvasPoint.X + ";" + canvasPoint.Y + ")";
}

private void vScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
    canvasBox.Top = -100 * e.NewValue;
}

private void hScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
    canvasBox.Left = -100 * e.NewValue;
}

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void trackBarDelay_Scroll(object sender, EventArgs e)
{
    LabelDelay.Text = $"Delay ({trackBarDelay.Value} ms)";
}

private void canvasBox_MouseDown(object sender, MouseEventArgs e)
{
    if (shapemode)
    {
        shape = new GameOfLifeShapes();
        if (e.Button.ToString() == "Left")
        {
            shape.Point1 = canvasPoint;
        }
    }
    if (!shapemode)
    {
        isdrawing = true;
    }
}

```

```

        if (e.Button.ToString() == "Left") DrawCells(true);
        else if (e.Button.ToString() == "Right") DrawCells(false);
    }

}

private void canvasBox_MouseUp(object sender, MouseEventArgs e)
{
    if (shapemode)
    {
        if (e.Button.ToString() == "Left")
        {
            shape.Point2 = canvasPoint;
            drawShape();
        }
    }
    isdrawing = false;
}

private async void DrawCells(bool draw)
{
    while (isdrawing)
    {
        if (canvasPoint.X < field.Width && canvasPoint.Y < field.Height)
        {
            field.Cells[canvasPoint.X, canvasPoint.Y] = draw;
            field.UpdateField(g);
            canvasBox.Image = canvas;
        }
        await Task.Delay(1);
    }
}

private void buttonClear_Click(object sender, EventArgs e)
{
    field.ClearCells();
    field.UpdateField(g);
    canvasBox.Image = canvas;
}

private void buttonLine_Click(object sender, EventArgs e)
{
    shapemode = true;
    buttonRect.Enabled = true;
    buttonLine.Enabled = false;
    buttonEllipse.Enabled = true;
    buttonPencil.Enabled = true;
    linemode = true;
    rectmode = false;
    elipsemode = false;
}

private void buttonRect_Click(object sender, EventArgs e)
{
    shapemode = true;
    buttonRect.Enabled = false;
    buttonLine.Enabled = true;
    buttonEllipse.Enabled = true;
    buttonPencil.Enabled = true;
    linemode = false;
    rectmode = true;
    elipsemode = false;
}

private void buttonPencil_Click(object sender, EventArgs e)
{
    shapemode = false;
    buttonRect.Enabled = true;
    buttonLine.Enabled = true;
    buttonEllipse.Enabled = true;
}

```

```

        buttonPencil.Enabled = false;
        linemode = false;
        rectmode = false;
        elipsemode = false;
    }

    private void buttonEllipse_Click(object sender, EventArgs e)
    {
        shapemode = true;
        buttonRect.Enabled = true;
        buttonLine.Enabled = true;
        buttonEllipse.Enabled = false;
        buttonPencil.Enabled = true;
        linemode = false;
        rectmode = false;
        elipsemode = true;
    }

    private void drawShape()
    {
        if (linemode) field.AddLine(shape.Point1, shape.Point2);
        if (rectmode) field.AddRect(shape.Point1, shape.Point2);
        if (elipsemode) field.AddElipse(shape.Point1, shape.Point2);
        field.UpdateField(g);
        canvasBox.Image = canvas;
    }

    private void buttonMenu_Click(object sender, EventArgs e)
    {
        MenuForm menuForm = new MenuForm();
        menuForm.Show();
        this.Dispose();
    }
}

```

## Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GameOfLifeWF
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MenuForm());
        }
    }
}

```