МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический
университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 «Математика и компьютерные науки»

Отчет о выполнении лабораторной работы №1 по предмету «Теория алгоритмов»

Студент,
группы 5130201/20101 _____ Астафьев И. Е.

Преподаватель _____ Востров А. В.

«_____»_____ 2024г.

Санкт-Петербург, 2024

Содержание

Введение

В ходе выполнения лабораторной работы была создана программа на языке Python, которая включает в себя реализацию двумерного клеточного автомата с тороидальными граничными условиями и окрестностью фон Неймана. Программа представлена в виде приложения с графическим интерфейсом, поддерживающим различные способы задания начальных условий автомата.

1 Постановка задачи

В данной лабораторной работе необходимо [1]:

- создать клеточный автомат по правилу 617232;
- использовать тороидальные граничные условия;
- создать графическое окно с полями для ввода: количества строк в сетке, количества столбцов в сетке, количества итераций;
- создать графическое окно, отображающее созданный клеточный автомат и содержащее элементы управления им:
  - выбор способа задания начальных условий: вручную или случайно;
  - слайдер (ползунок) для перелистывания итераций состояния автомата;
- обработать некорректный пользовательский ввод.

2 Математическое описание

2.1 Клеточный автомат

Клеточные автоматы представляют собой математические модели, имитирующие поведение физических систем, где пространство и время дискретны, то есть разбиты на отдельные ячейки и временные шаги соответственно [1]. Клеточный автомат состоит из однородной решетки, обычно бесконечной по протяженности, с дискретной переменной в каждом узле - ячейке. В таких системах каждая ячейка может принимать одно из ограниченного набора возможных состояний.

Каждое состояние клеточного автомата полностью определяется значениями всех ячеек в данный момент времени. Эволюция системы происходит пошагово: на каждом новом временном шаге состояние каждой ячейки изменяется под влиянием состояния соседних ячеек на предыдущем шаге. Соседние клетки и сама клетка образуют окрестность. Все изменения происходят синхронно для всех ячеек согласно заранее установленным правилам, определяющим новые состояния на основе предыдущих состояний клеток в окрестности.

Одномерный автомат задается (бесконечной) строкой клеток, каждая из которых может принимать k различных значений. Чаще всего рассматриваются автоматы, в которых клетки принимают значения 1 или 0.

Для одномерного клеточного автомата окрестность представляет 3 клетки: центральная рассматриваемая клетка s_0 и две ее соседние клетки s_1, s_2.

$$\cdots|s_1|s_0|s_2|\cdots$$

Ниже показан пример набора локальных правил для элементарного одномерного клеточного автомата [1]. В верхней строке приведены все возможные ($2^3 = 8$) значения трех клеток в окрестности, а под каждой из них указано значение, которое получит центральная клетка на следующем временном шаге в соответствии с определенным локальным правилом.

$$\begin{array}{cccccccc} \underline{111} & \underline{110} & \underline{101} & \underline{100} & \underline{011} & \underline{010} & \underline{001} & \underline{000} \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}$$

Локальные правила для одномерного клеточного автомата с окрестностью из 3 клеток описываются восьмизначным двоичным числом, как в примере выше. (При описании клеточных автоматов мы используем это двоичное число взаимозаменяемо с его десятичным эквивалентом.) Поскольку любое восьмизначное двоичное число определяет клеточный автомат, возможны $2^{2^3} = 256$ различных правила одномерного клеточного автомата с окрестностями из трех клеток. Таким образом, если клетки автомата могут принимать лишь два значения, то для описания конкретного правила можно использовать бинаную функцию от трех переменных: $s_0^{t+1} = f(s_0^t, s_1^t, s_2^t)$, где каждому набору состояний окрестности сопоставлено новое состояние центральной клетки.

Аналогично можно рассмотреть двумерный клеточный автомат. Двумерный клеточный автомат представляется (бесконечным) двумерным массивом с клетками, которые могут принимать k различных значений.

Существует два основных типа окрестности в двумерном пространстве: окрестность Мура (Рис. 1) и окрестность фон Неймана (Рис. 2).

Окрестность Мура - окрестность квадртаной формы, которую можно использовать для определения набора ячеек, окружающих заданную ячейку x_0, y_0 (где x_0, y_0 - координаты клетки в двумерном пространстве), которые могут повлиять на эволюцию двумерного клеточного автомата на квадратной сетке [2]. Окрестность Мура с радиу-

сом r определяется как

$$N^M_{(x_0,y_0)} = \{(x,y) : |x - x_0| \leq r, |y - y_0| \leq r\}.$$

Окрестность фон Неймана - ромбовидная окрестность (или крестовидная), которую можно использовать для определения набора ячеек, окружающих заданную ячейку x_0, y_0 (где x_0, y_0 - координаты клетки в двумерном пространстве), которые могут повлиять на эволюцию двумерного клеточного автомата на квадратной сетке [3]. Окрестность фон Неймана с радиусом r определяется как

$$N^V_{(x_0,y_0)} = \{(x,y) : |x - x_0| + |y - y_0| \leq r\}.$$

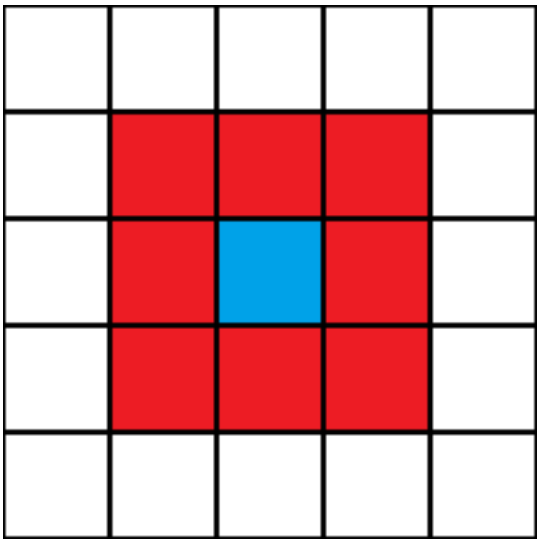

Рис. 1. Красные ячейки - это окрестности Мура для синей ячейки.

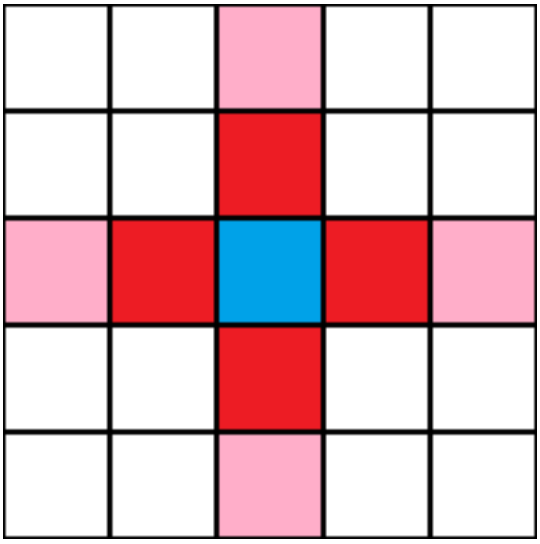

Рис. 2. Красные ячейки являются окрестностью фон Неймана для синей ячейки. «Перекрестная окрестность» диапазона-2 также включает розовые ячейки.

Однако для данного двумерного автомата будет рассматриваться окрестность фон Неймана с радиусом $r = 1$.

Таким образом, для задания правил двумерного клеточного автомата с клетками, принимающими два значения, можно использовать булеву функцию от 5 переменных

(клетки из окрестности фон Неймана), которая будет задавать новое состояние рассматриваемой клетки $s_0^{t+1} = f(s_0^t, s_1^t, s_2^t, s_3^t, s_4^t)$ [4]. Тогда существует $2^{2^5} = 2^{32}$ возможных функций перехода состояний.

$$\begin{array}{c} |s_1| \\ \hline |s_3|s_0|s_4| \\ \hline |s_2| \end{array}$$

В итоге по заданному вектору значений функции f, например заданному десятиричным числом, можно однозначно соотносить предыдущие состояния окрестности с новым состоянием данной клетки.

Вектор значений функции в работе считается по формуле:

$f =$ `номер варианта * 11 * год рождения * день * месяц`

$f = 2 \cdot 11 \cdot 2004 \cdot 14 \cdot 1 = 617232_{10} = 00000000000010010110101100010000_2$

Так как функция f от пяти переменных, то вектор функции дополнен незначащими нулями в начале, чтобы получить вектор из 32 значений (1).

Таблица 1. Таблица истинности по вектору значения функции f

x_0	x_1	x_2	x_3	x_4	f
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	0	1	0
0	1	1	1	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

Так как модель клеточного автомата реализуется на поле конечного размера, то требуется определить граничные условия, которые описывают ситуации, когда клетка находится на краю поля. Это включает в себя установление значений для соседних клеток за пределами границы сетки. В данной работе были выбраны тороидальные условия границ, при которых происходит следующее: когда клетка у верхнего края, ее верхний сосед находится в соответствующем столбце снизу (и наоборот для ситуации у нижней границы), а клетка у левой границы, ее левый сосед находится в соответствующей строке справа (и наоборот для ситуации у правой границы). Такие условия можно представить как склеивание левого и правого краев поля (прямоугольника) в форму цилиндра, а затем склеивание верхнего и нижнего краев полученного цилиндра в виде тора.

Ниже приведены общие формулы для вычисления координат клеток из окрестности фон Неймана для граничных случаев для клетки с координатами $(i, j), 0 \leq i < num\_cols, 0 \leq j < num\_rows$, где num_rows - количество строк, num_cols - количество столбцов поля.

$$
\begin{aligned}
s_0 &= (i, j) \\
s_1 &= ((i+1) \bmod \text{num\_cols}, j) \\
s_2 &= ((i-1) \bmod \text{num\_cols}, j) \\
s_3 &= (i, (j-1) \bmod \text{num\_rows}) \\
s_4 &= (i, (j+1) \bmod \text{num\_rows})
\end{aligned}
$$

2.2 Классификация

Стивен Вольфрам [5] предлагает разделять клеточные автоматы на четыре класса, основываясь на их эволюционном поведении. Эта классификация была первой попыткой систематизировать правила самих автоматов, а не только их поведенческие типы. Классы располагаются в порядке увеличения сложности следующим образом:

- Класс 1: Эволюция из начальных условий приводит к быстрому переходу в гомогенное и стабильное состояние. Любые негомогенные структуры быстро исчезают.
- Класс 2: Эволюция из начальных условий приводит к быстрому переходу в неизменное негомогенное состояние или к зацикленным последовательностям. Большинство начальных структур быстро исчезает, но некоторые сохраняются. Локальные изменения в начальных условиях оказывают влияние лишь на ближайшую область системы.
- Класс 3: В большинстве случаев эволюция начальных условий приводит к псевдослучайным, хаотическим последовательностям. Любые стабильные структуры, которые появляются, быстро разрушаются из-за окружающего их шума. Локальные изменения начальных условий вызывают неопределённое влияние на эволюцию системы.
- Класс 4: Эволюция порождает структуры, которые взаимодействуют сложным образом, формируя локальные, стойкие образования. В ходе эволюции могут возникать некоторые последовательности характерные для Класса 2. Локальные изменения начальных условий оказывают неопределённое влияние на развитие системы. Некоторые клеточные автоматы этого класса, такие как Правило 110 и игра "Жизнь обладают универсальностью в смысле Тьюринга.

3 Особенности реализации

3.1 Константы

Согласно заданию, правило для клеточного автомата задается десятиричным числом:

$f =$ `номер варианта * 11 * год рождения * день * месяц`

В программе эта информация задается отдельными константами:

```
VARIANT = 2
YEAR_OF_BIRTH = 2004
MONTH_OF_BIRTH = 1
DAY_OF_BIRTH = 14
RULE_VALUE = VARIANT * 11 * YEAR_OF_BIRTH * MONTH_OF_BIRTH * DAY_OF_BIRTH
NUM_OF_VARIABLES = 5
NUM_OF_DIGITS = 2 ** NUM_OF_VARIABLES
ALL_RULES = 2 ** NUM_OF_DIGITS
```

- `RULE_VALUE` задает правило - функцию перехода клеточного автомата.
- `NUM_OF_DIGITS` задает длину вектора значений функции перехода состояния.

3.2 Класс Cell

Класс Cell представляет реализацию клетки автомата. Таким образом Cell являет оберткой типа bool.

Поля класса:

- `state` - состояние клетки, которое может принимать два значения: True или False.

Конструктор

Вход: `state` - состояние для создаваемой клетки.

Выход: клетка автомата с заданным состоянием.

В конструкторе задается начальное состояние клетки. По умолчанию значение False.

```
class Cell:
    def __init__(self, state: bool = False):
        self.state = state
```

Метод new_state()

Вход: `state` - новое состояние для клетки.

Выход: клетка с обновленным состоянием.

Метод присваивает текущей клетке новое состояние `state`.

```
def new_state(self, state: bool):
    self.state = state
```

3.3 Класс AutomataGrid

Класс AutomataGrid описывает поле (сетку) клеточного автомата. Класс содержит двумерный массив `ndarray` библиотеки Numpy, в элементах которого находятся клетки, описываемые классом Cell. Данный класс имеет возможность вычислять следующее состояние конкретной клетки и всей сетки целиком.

Поля класса:

- `rows` - количество строк в поле;
- `cols` - количество столбцов в поле;
- `grid` - двумерный массив поля, содержащий в себе клетки автомата.

Конструктор

Вход: `rows` - количество строк, `cols` - количество столбцов.
Выход: клеточный автомат с заданными размерами поля.

Конструктор создает объект класса AutomataGrid с двумерным `ndarray` массивом размера `rows` на `cols` и заполняет его клетками Cell с начальным состоянием False.

```
def __init__(self, rows: int, cols: int):
        self.rows = rows
        self.cols = cols
        self.grid = np.empty((rows, cols), dtype=Cell)
        for r in range(rows):
                for c in range(cols):
                        self.grid[r, c] = Cell(False)
```

fill_randomly()

Вход: клеточный автомат.
Выход: заполненное случайным образом поле клеточного автомата.

Метод проходит по всем элементам массива `grid` здавая случайные значения клеткам автомата.

```
def fill_randomly(self):
        for r in range(self.rows):
                for c in range(self.cols):
                        random_state = np.random.choice([True, False])
                        self.grid[r, c].new_state(random_state)
```

__rule()

Вход: `cells_states` - кортеж состояний клеток из окрестности фон Неймана.
Выход: новое состояние текущей центральной клетки.

Статический метод вычисляет новое состояние клетки согласно заданному вектором значений правилу. Вектор значений булевой функции вычисляется переводом из

десятичного представления константы NUM_OF_DIGITS в двоичное и добавлением незначащих нулей. Затем кортеж значений переводится в десятичное число, которое обозначает номер строки в таблице истинности. Затем по индексу из всектора берется новое значение состояния.

```
@staticmethod
def __rule(cells_states: Neumann) -> bool:
        initial_rule_value = const.RULE_VALUE
        num_of_state = bools_to_int(cells_states)
        bin_rule_value = format(initial_rule_value, f'0{const.NUM_OF_DIGITS}b')
        string_result = bin_rule_value[num_of_state]
        return str_to_bool(string_result)
```

__cell_next_state()

Вход: `row` - номер строки клетки, `col` - номер столбца клетки.

Выход: клетка с новым состоянием.

Метод вычисляет новое состояние для клетки с заданными координатами. Координаты соседних клеток определяются согласно тороидальным граничным условиям. Затем для кортежа клеток окрестности вызывается метод `__rule()`.

```
def __cell_next_state(self, row: int, col: int) -> Cell:
        num_rows = self.rows
        num_cols = self.cols

        # Get the current cell and its neighbors using modular arithmetic for wrapping
        s_0 = self.grid[row, col]
        s_1 = self.grid[(row + 1) % num_rows, col]  # Down
        s_2 = self.grid[(row - 1) % num_rows, col]  # Up
        s_3 = self.grid[row, (col - 1) % num_cols]  # Left
        s_4 = self.grid[row, (col + 1) % num_cols]  # Right

        # Collect the states of the Neumann neighborhood
        neumann_neighbourhood = (s_0.state, s_1.state, s_2.state, s_3.state, s_4.state)
        return Cell(self.__rule(neumann_neighbourhood))
```

next_iteration()

Вход: клеточный автомат в текущем состоянии.

Выход: клеточный автомат в следующем итерации.

Метод определяет состояния всех клеток автомата в следующий момент времени с помощью метода `__cell_next_state()` и задает состояние автомата в следующей итерации в целом.

```
def next_iteration(self):
        grid_copy = np.empty((self.rows, self.cols), dtype=Cell)
        for row in range(self.rows):
                for col in range(self.cols):
```

```
                    grid_copy[row, col] = self.__cell_next_state(row, col)
        self.grid = grid_copy
```

3.4 Интерфейс

Интерфейс реализован с помощью фреймворка Qt. Используется набор расширений PyQt6, в частности модули QtCore и QtWidgets.

На основе QWidget из QtWidgets создано окно ChooseWindow с полями выбора размера сетки автомата и количества итераций. В классе данного окна есть обработка пользовательского некорректного ввода: неверные числовые значения и нечисловые значения. Нажатие на кнопку «Создать поле» создает новое окно с полем клеточного автомата.

Окно поля клеточного автомата UIGrid так же создано на основе QWidget. Оно представляет собой: поле заданного размера состоящее из кнопок, которые имеют белый цвет для состояния False и красный для состояния True; кнопки управления режимом работы (ручное или случайное задание начального состояния); слайдер для перелистывания итераций; кнопку для просчитывания итераций; а также кнопку очистки. Кнопки отображающие состояние клеток автомата активны в ручном режиме и отключены (не «кликабельны») в режиме случайной генерации.

При нажатии на кнопку «Просчитать итерации» для заданного состояния клеточного автомата вычисляются состояния в `n` следующих итерациях с помощью метода `next_iteration()`. При перемещении слайдера будут отображаться соответствующии этапы эволюции изначального автомата.

4 Результаты работы

При запуске приложения открывается окно с выбором размера поля и количества итераций (Рис. 3).

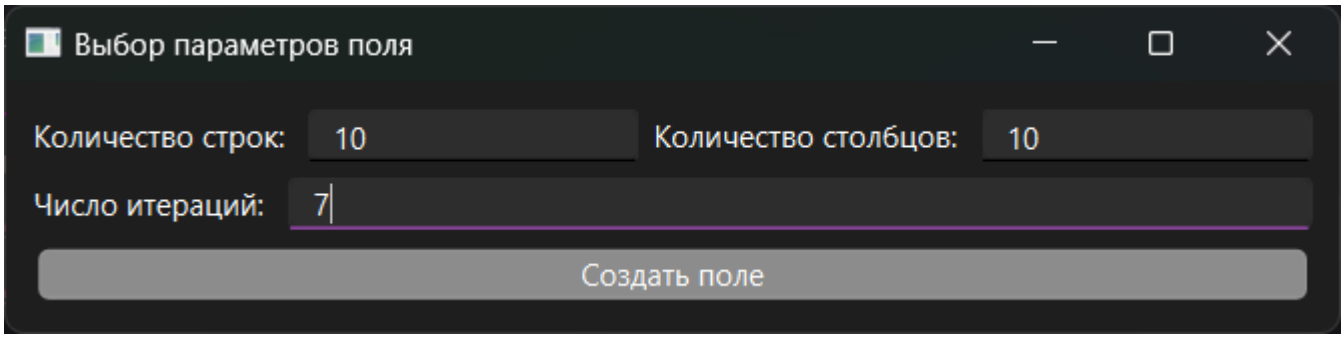

Рис. 3. Начальное окно

При вводе некорректных данных, пользователю выводится соответствующее окно с предупреждением (Рис. 4, 5).

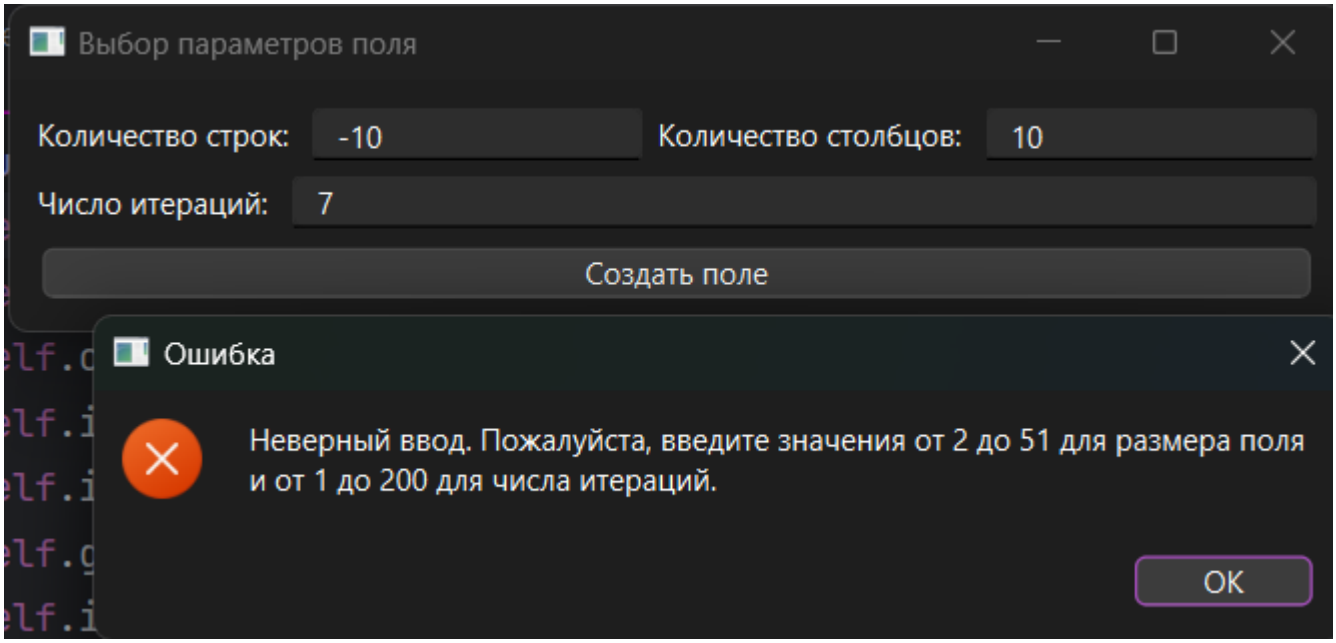

Рис. 4. Неверные числовые значения

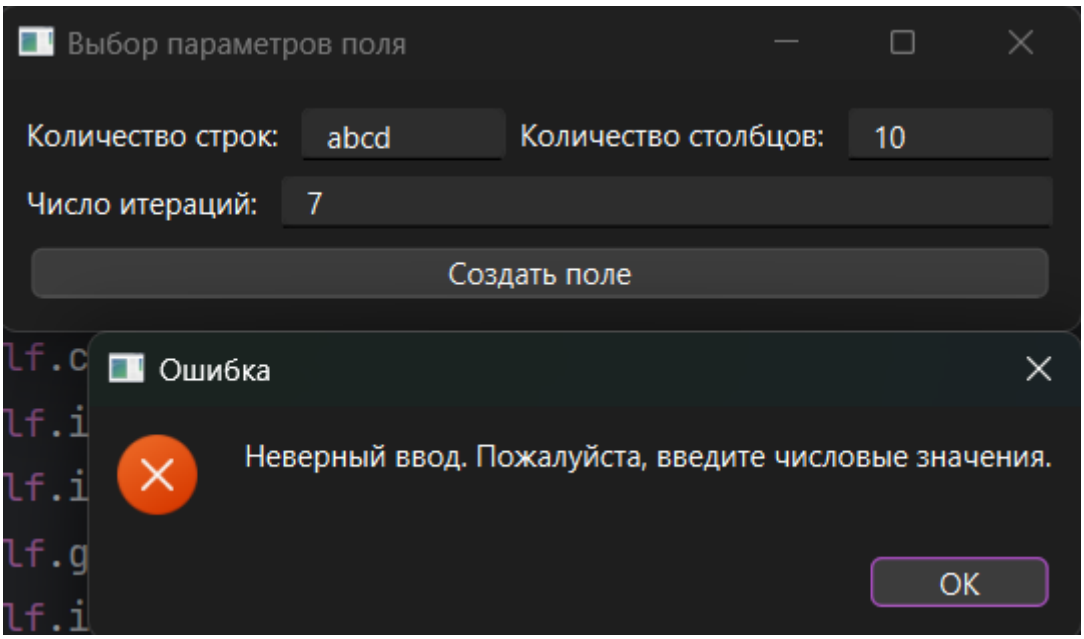

Рис. 5. Случай ввода строки вместо числового значения

После нажатия на кнопку «Создать поле» открывается окно с полем клеточного автомата (Рис. 6). По умолчанию включен ручной режим задания начальных условий.

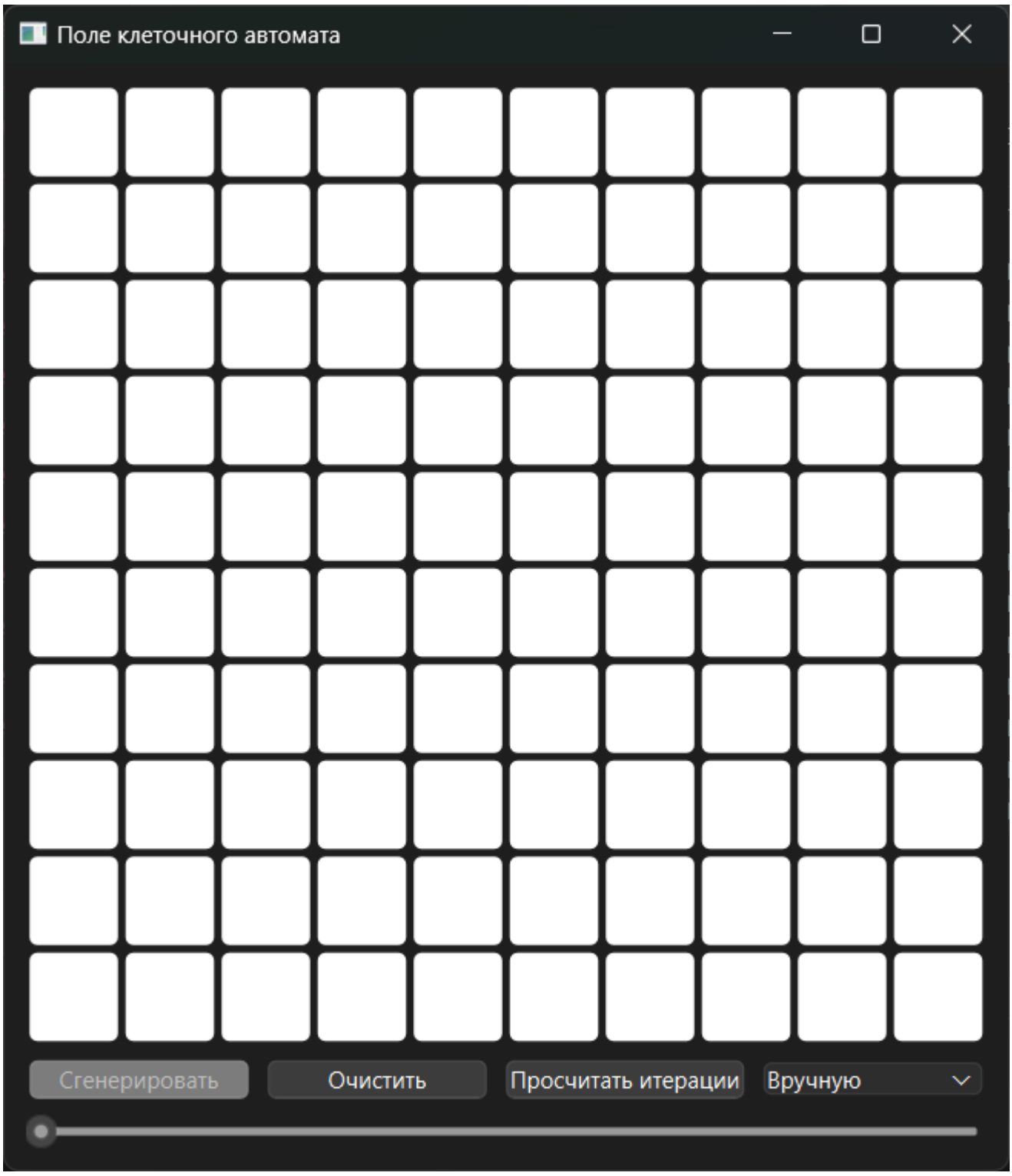

Рис. 6. Окно поля автомата

Затем, задав начальное состояние автомата (Рис. 7), можно нажать на кнопку «Просчитать итерации», после чего активируется слайдер перелистывания итераций. При передвижении «ползунка» слайдера вправо в окне будет отображаться соответствующий этап эволюции автомата (Рис. 8).

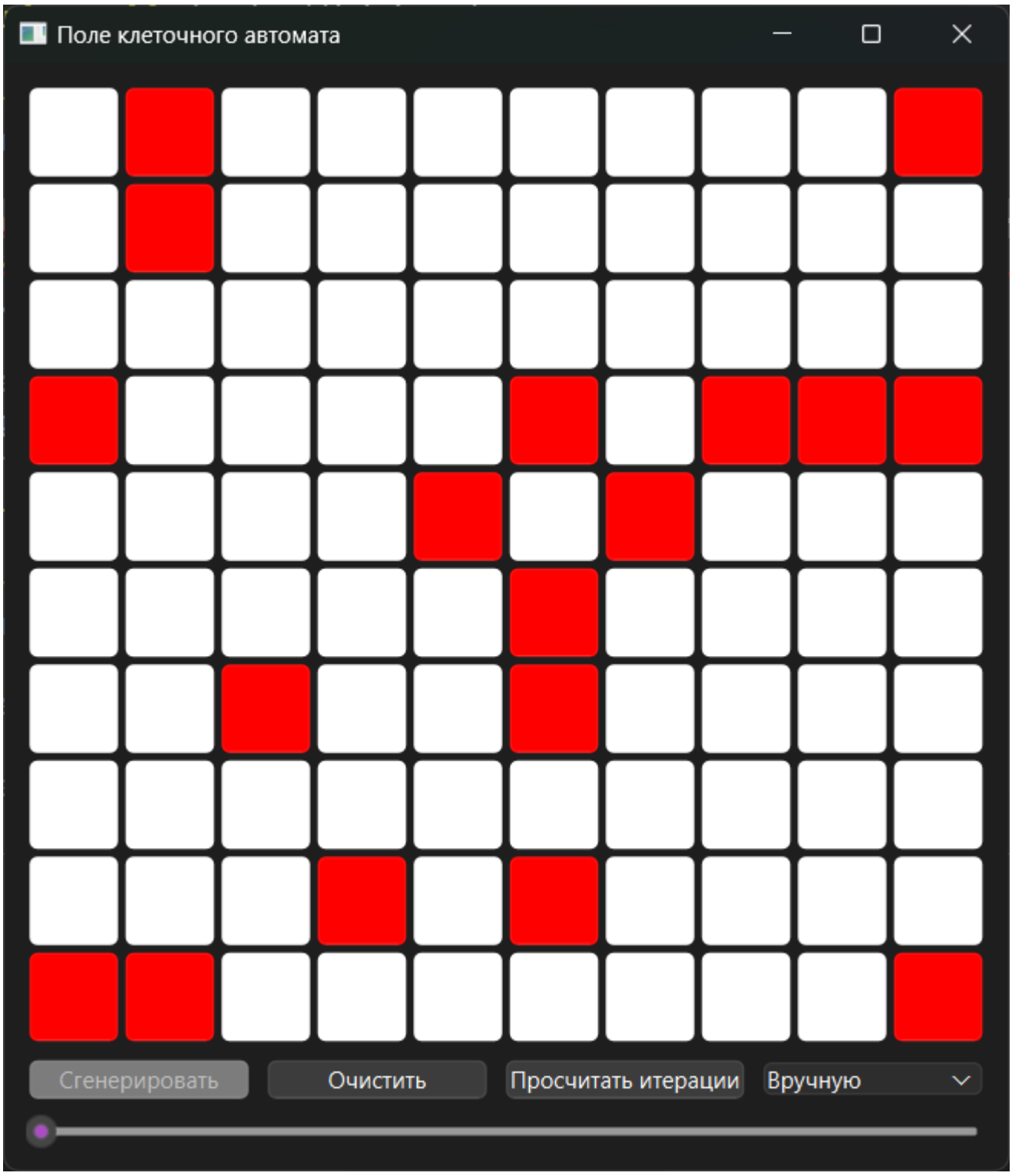

Рис. 7. Клеточный автомат, заданный вручную

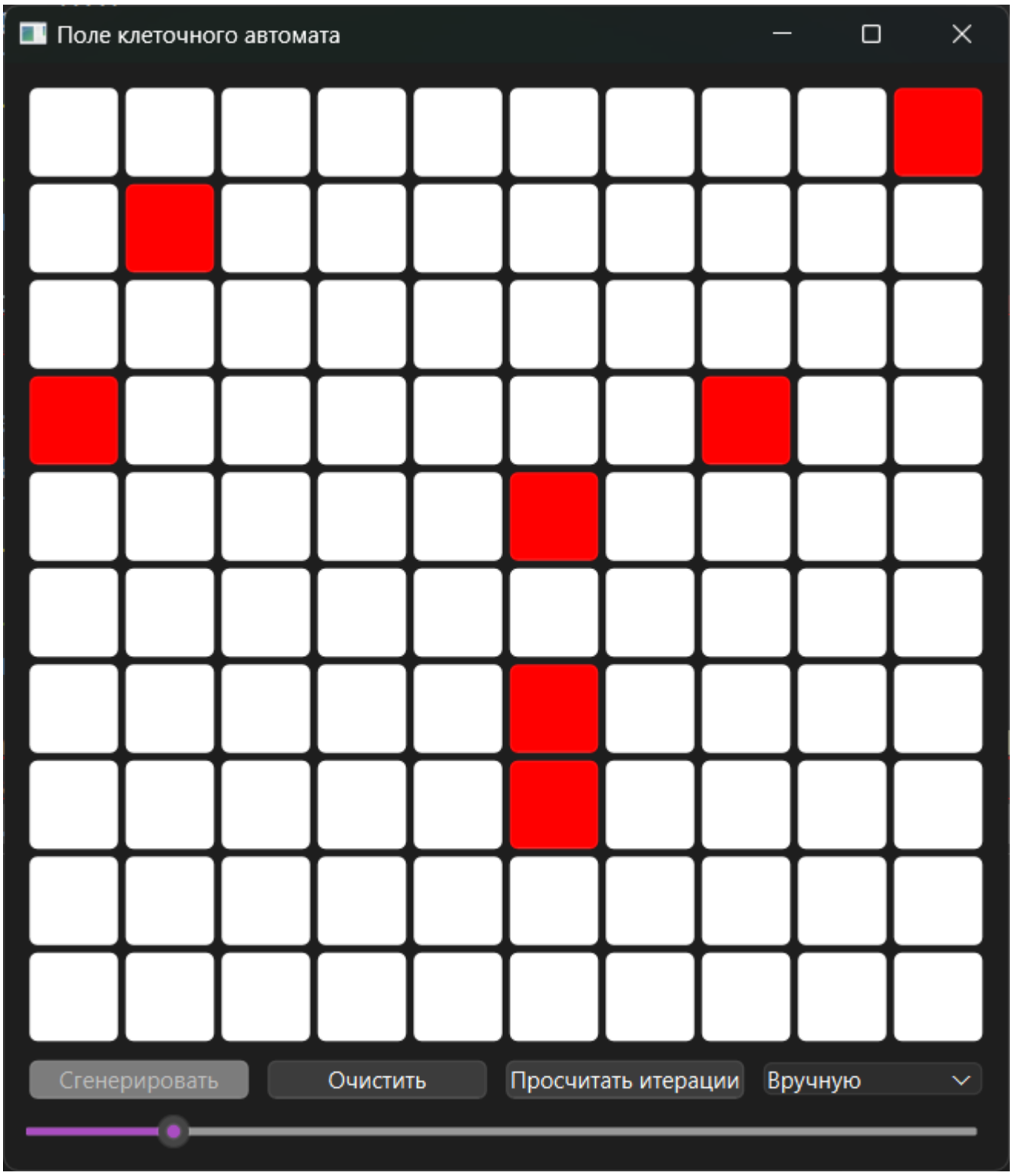

Рис. 8. 2 итерация автомата, заданного вручную

Также, если переключить режим на «Случайно», можно сгенерировать случайное начальное состояние автомата (Рис. 9).

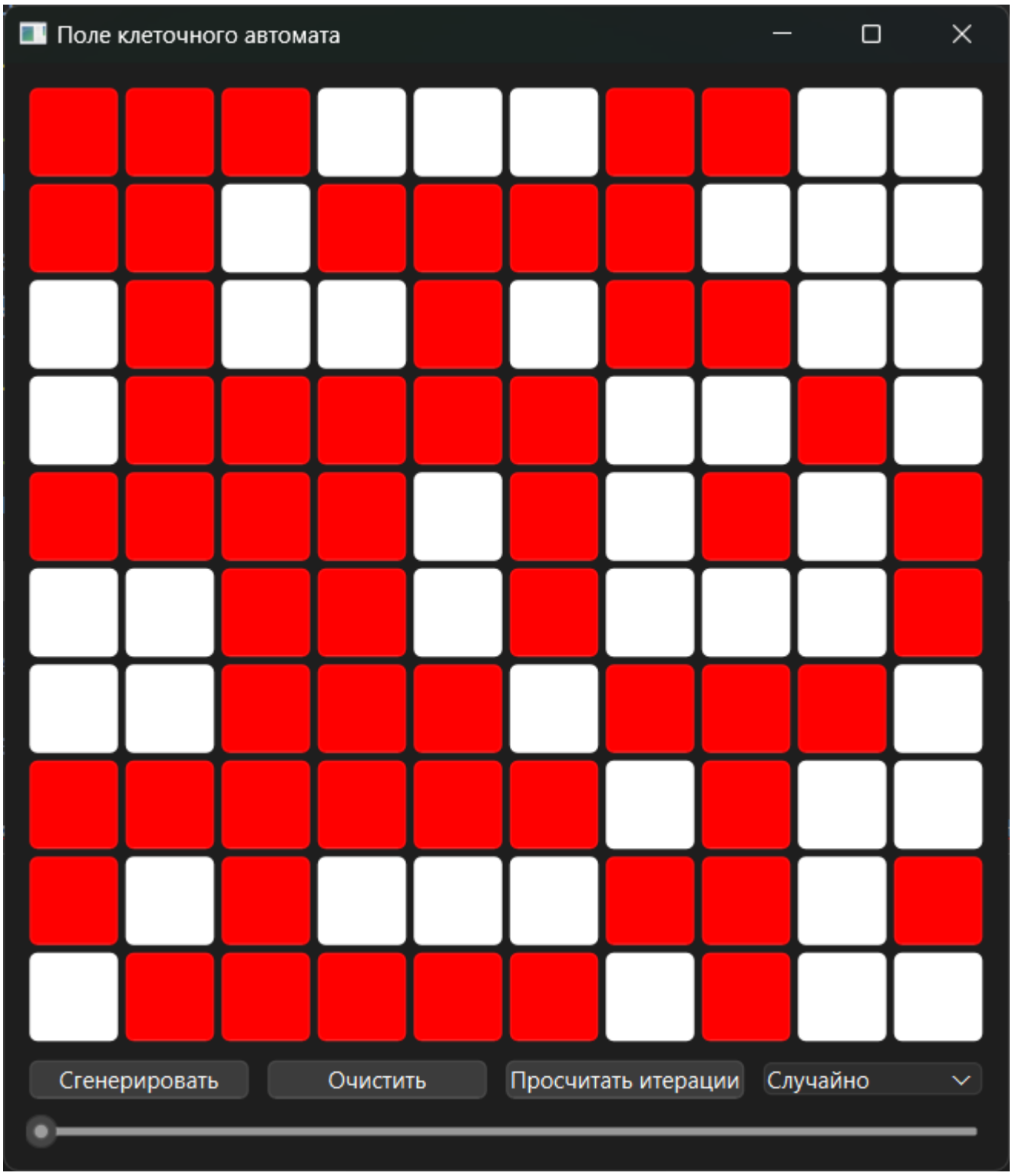

Рис. 9. Клеточный автомат, заданный случайно

5 Анализ клеточного автомата

Для анализа автомата было задано поле размером 15 x 15 и 10 итераций. Эволюция автомата с пустого поля всегда оставляет поле пустым начиная с первой итерации (Рис. 10).

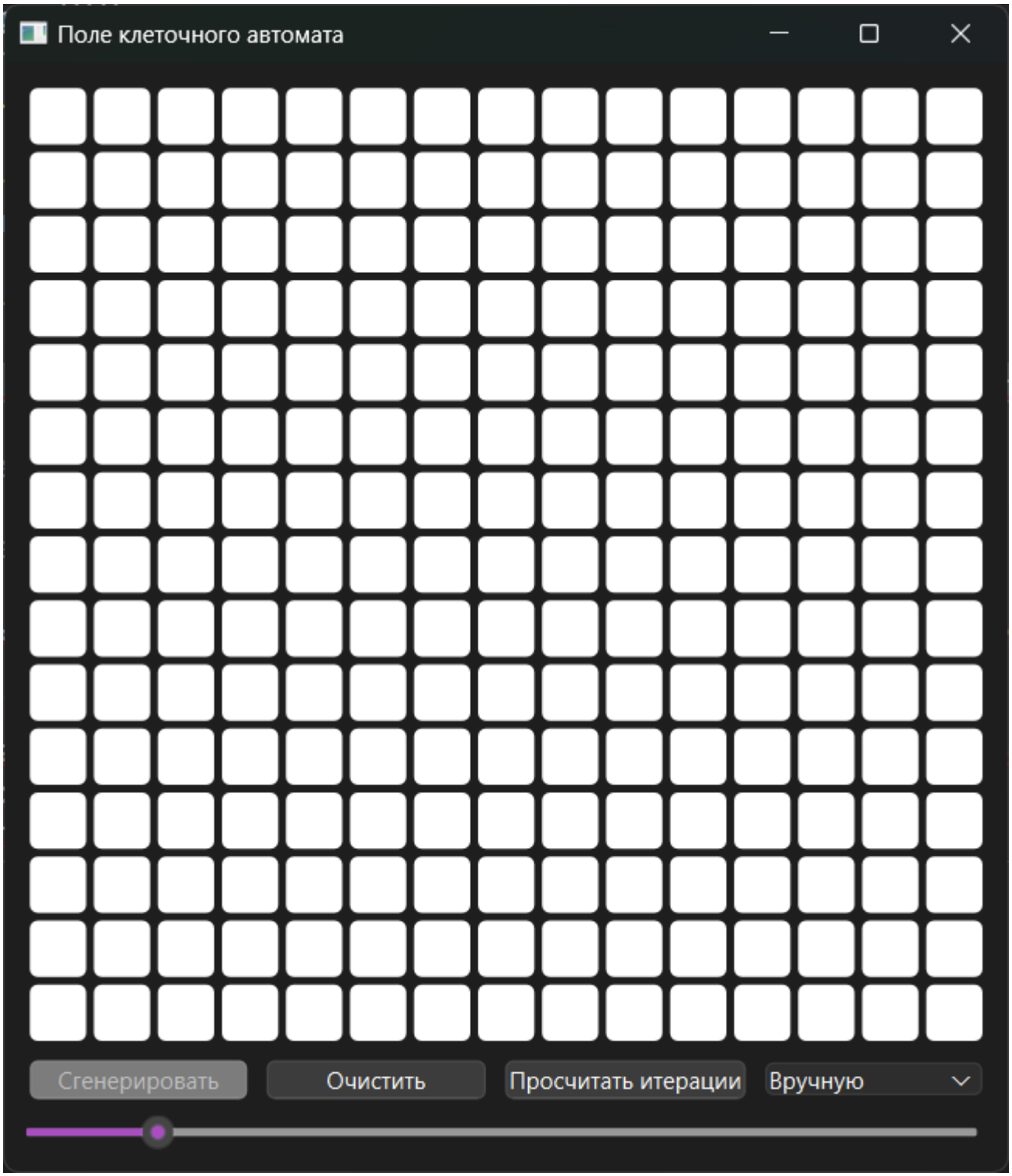

Рис. 10. Эволюция автомата с пустым полем

При задании поля с «активными» клетками по периметру, клетки постепенно будут исчезать и на 9 итерации поле станет пустым (Рис. 11 - Рис. 16).

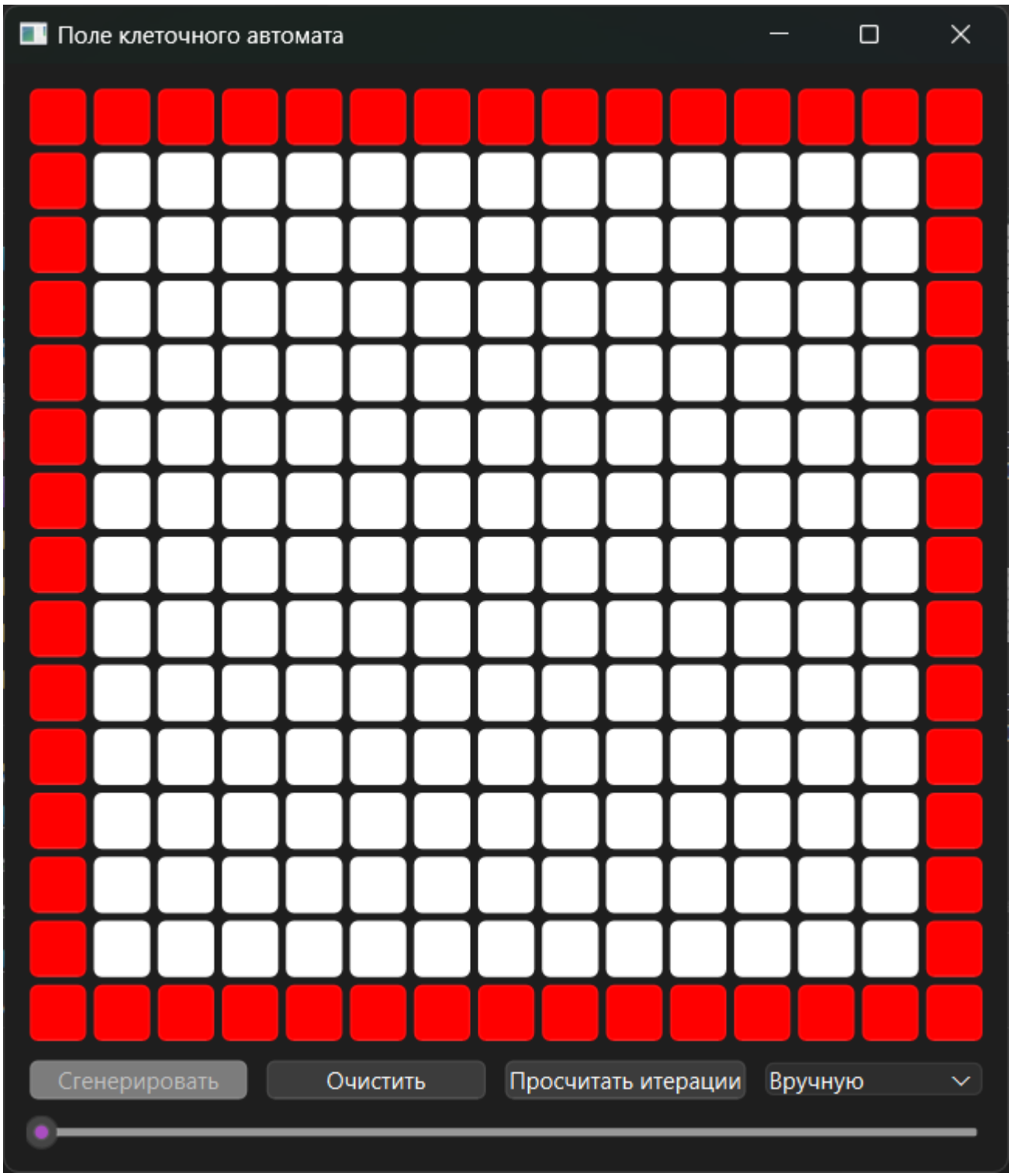

Рис. 11. Эволюция автомата с клетками по периметру. Начальное состояние

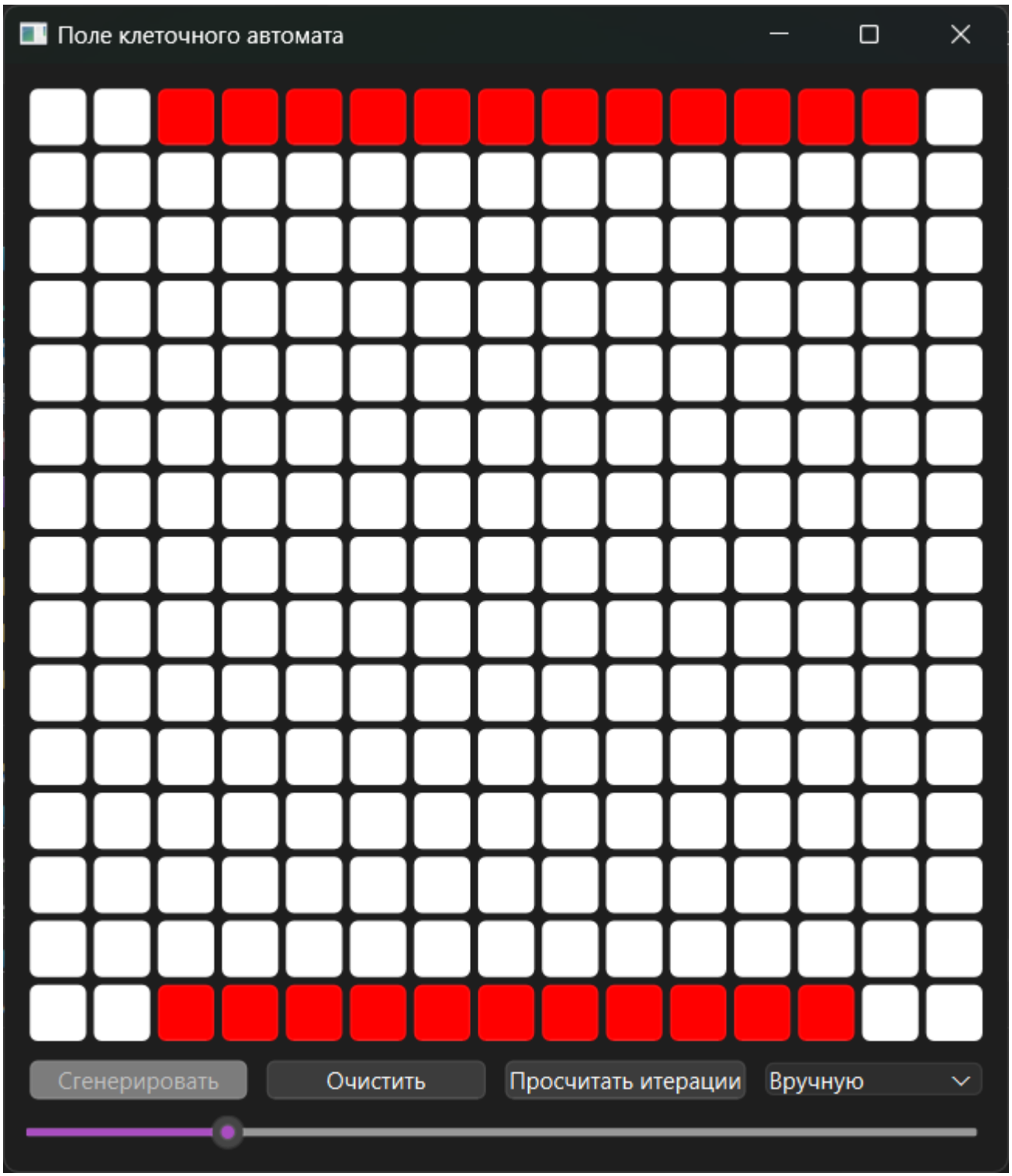

Рис. 12. Эволюция автомата с клетками по периметру. 2 итерация

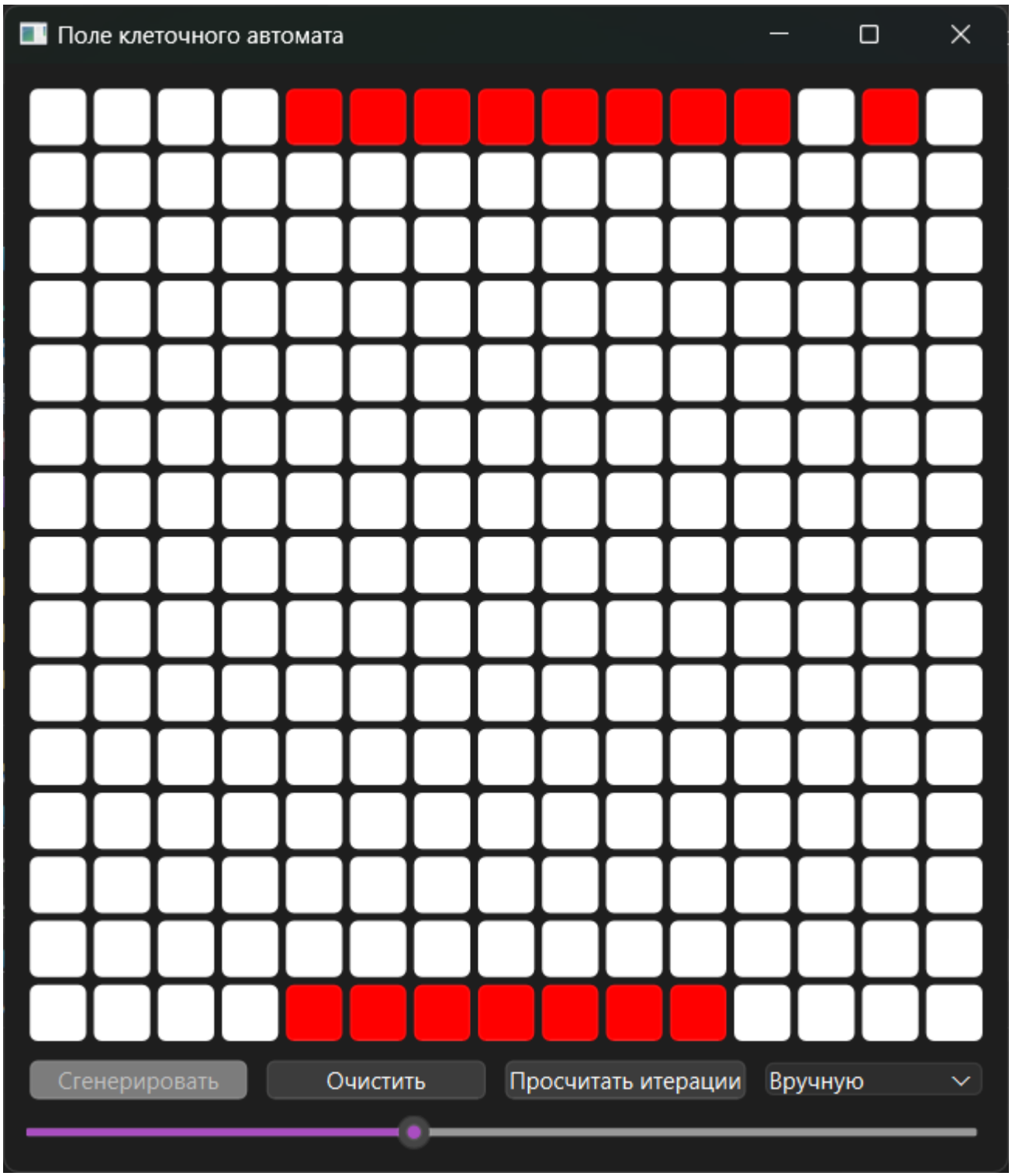

Рис. 13. Эволюция автомата с клетками по периметру. 4 итерация

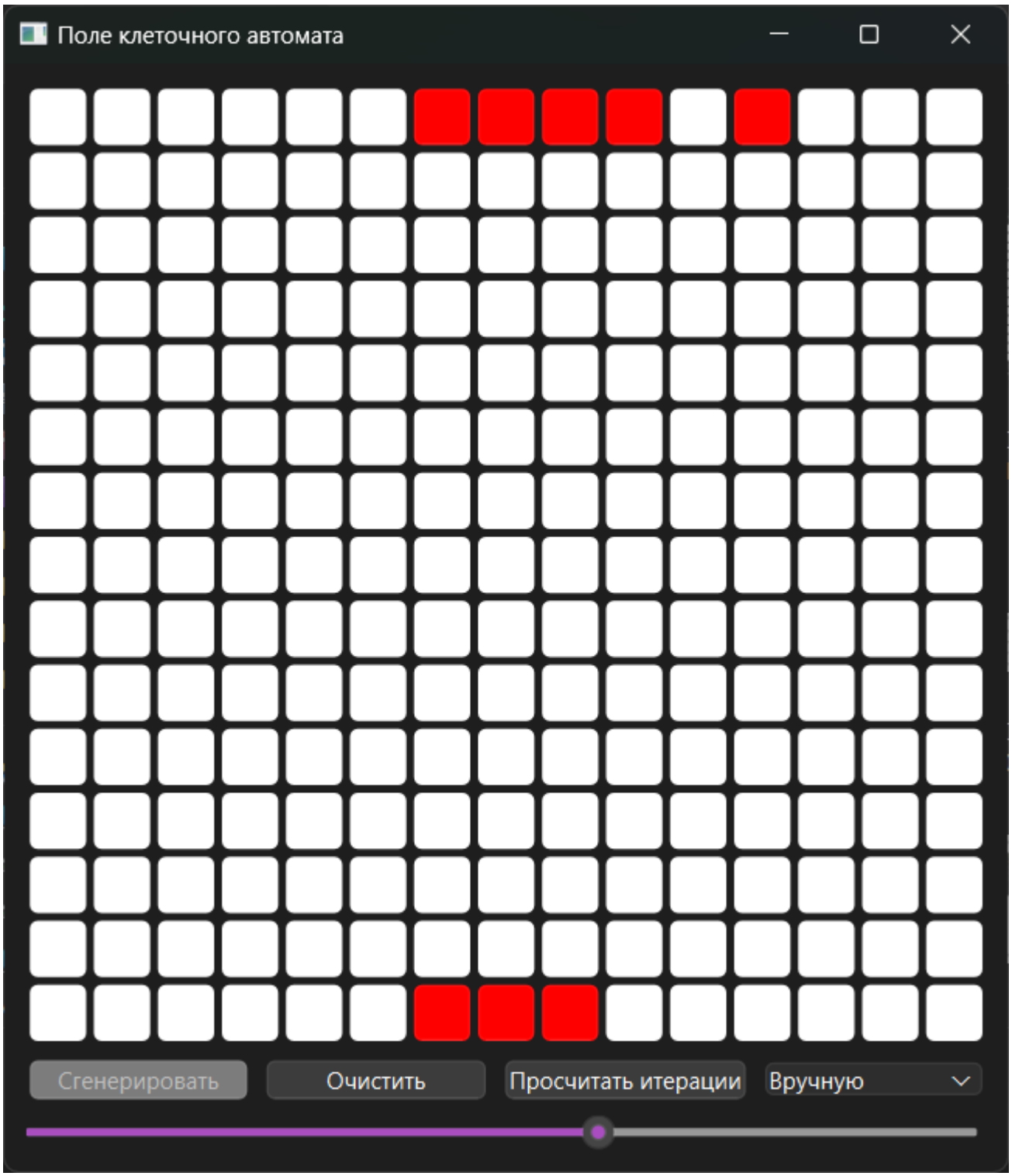

Рис. 14. Эволюция автомата с клетками по периметру. 6 итерация

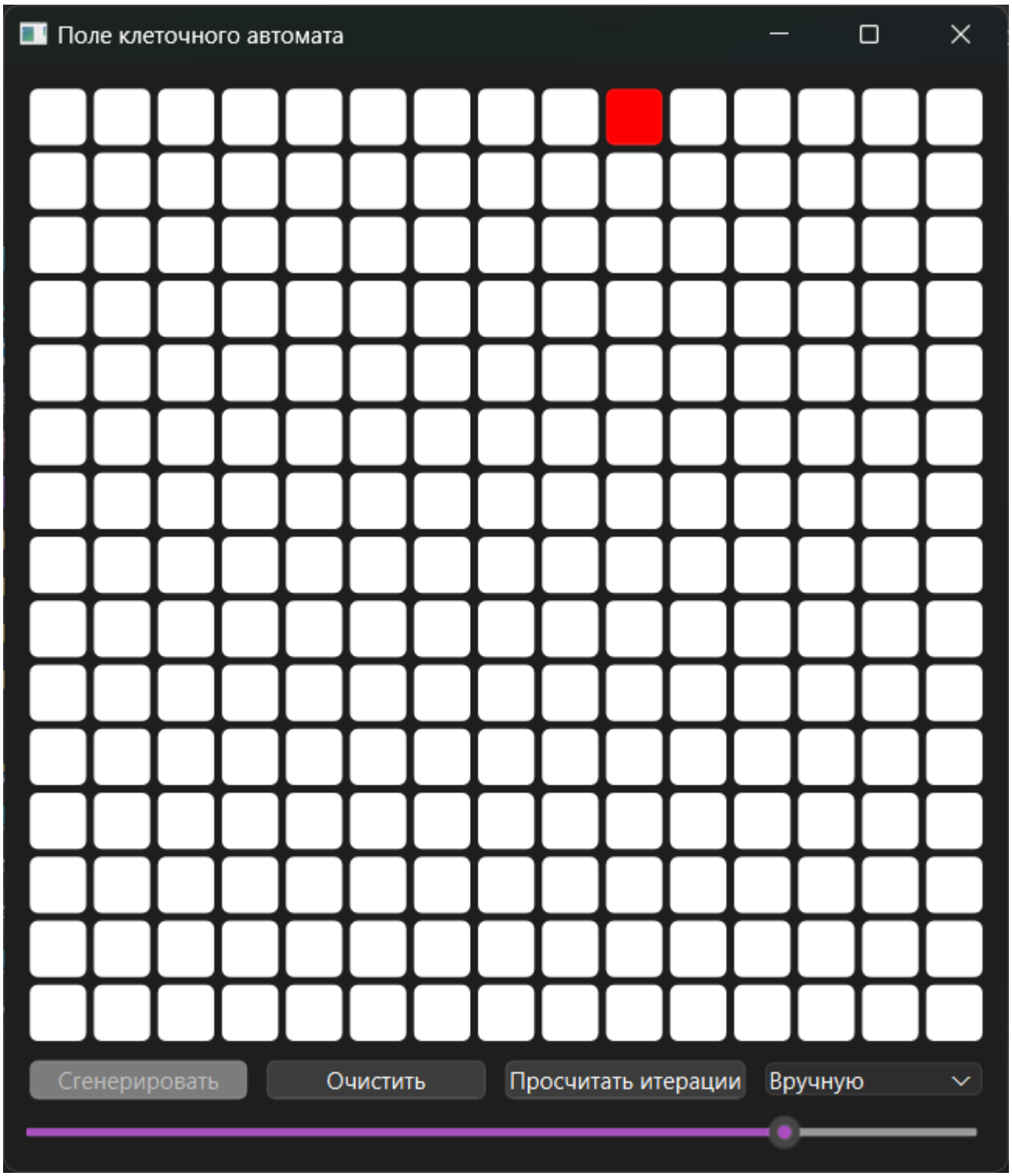

Рис. 15. Эволюция автомата с клетками по периметру. 8 итерация

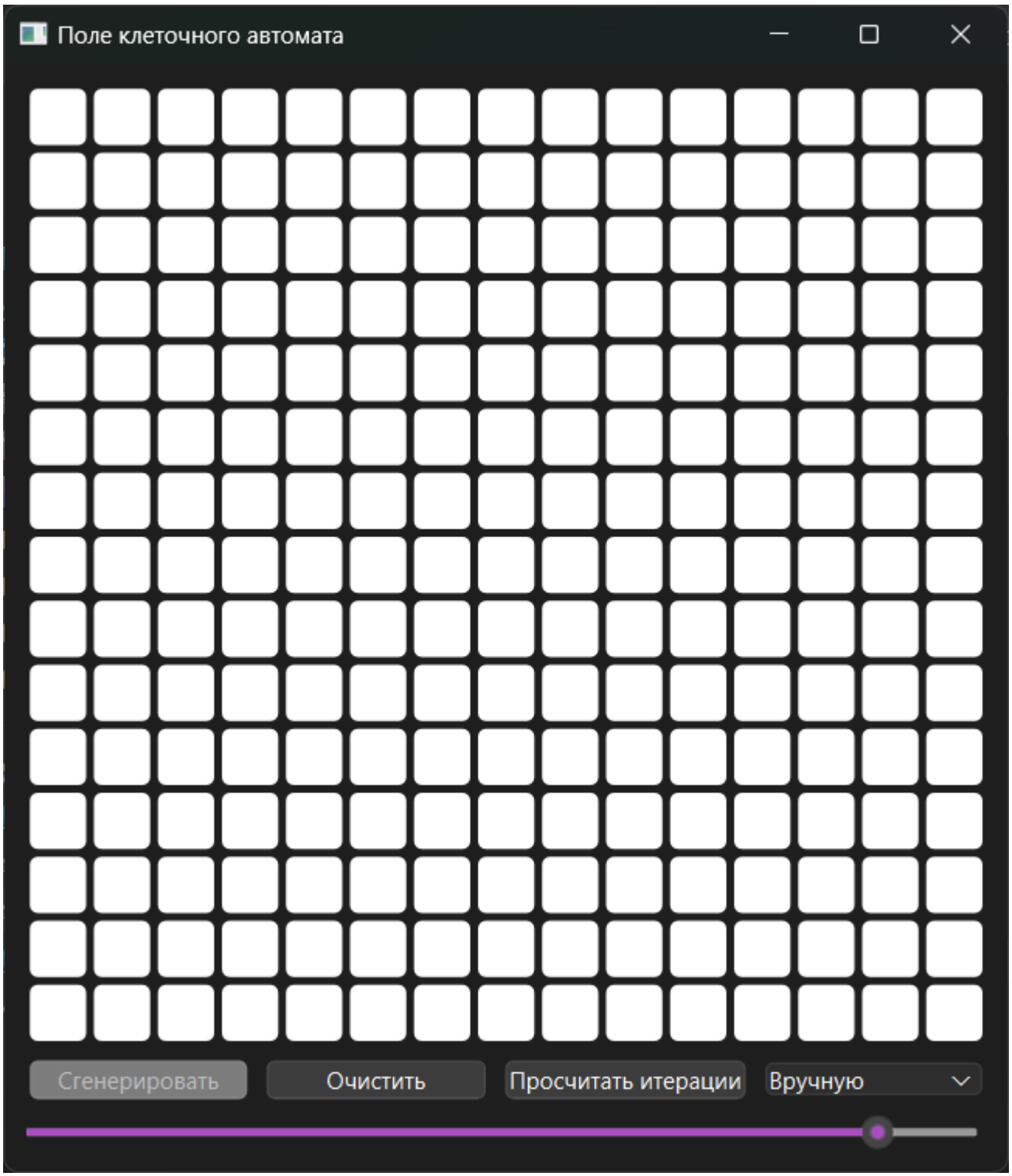

Рис. 16. Эволюция автомата с клетками по периметру. 9 итерация

Далее автомат был запущен на случайных начальных условиях (Рис. 17 - Рис. 20). Начиная с 4 итерации автомат сохраняет статичное состояние на всех последующих иетрациях.

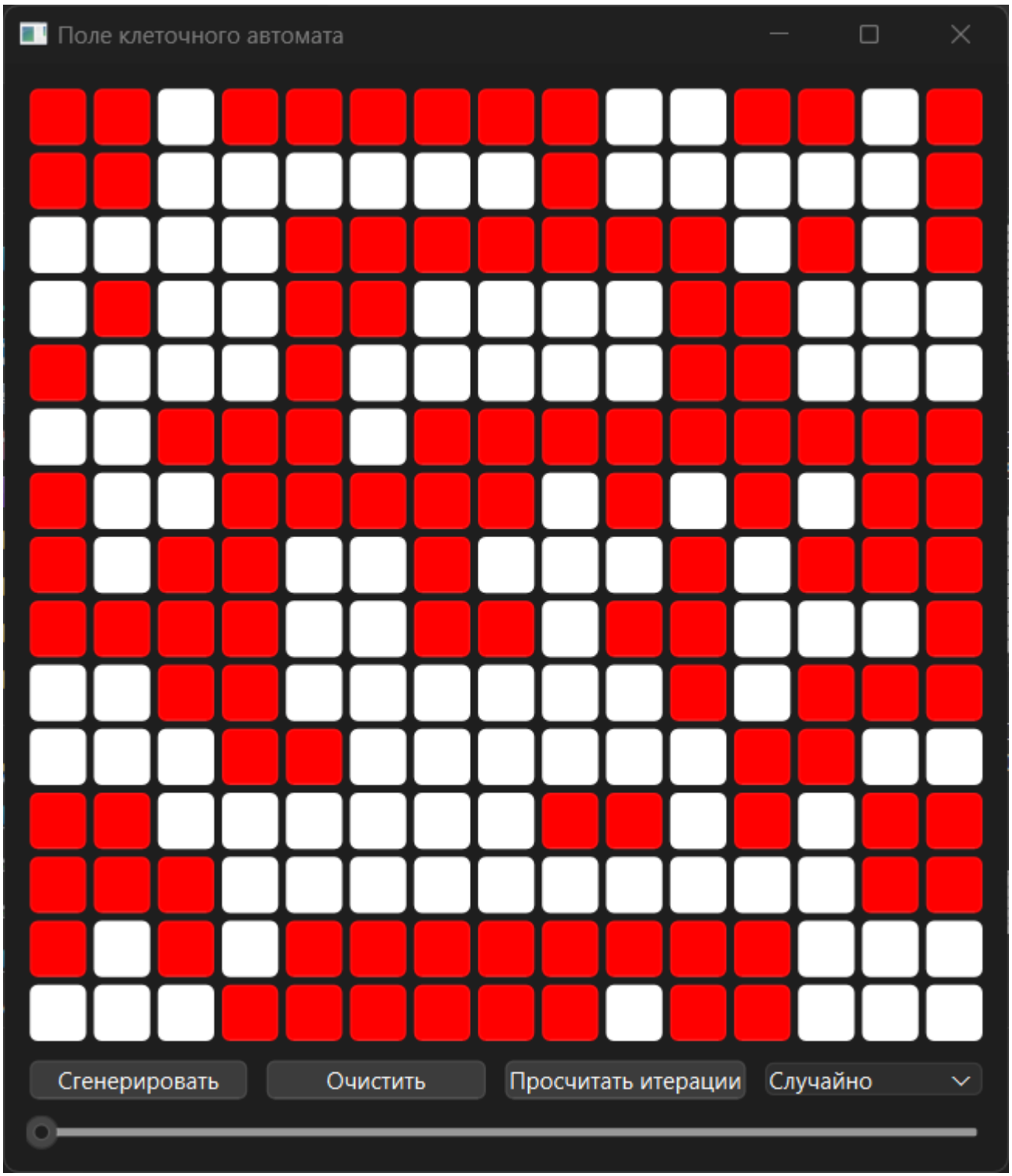

Рис. 17. Эволюция автомата со случайными начальными условиями

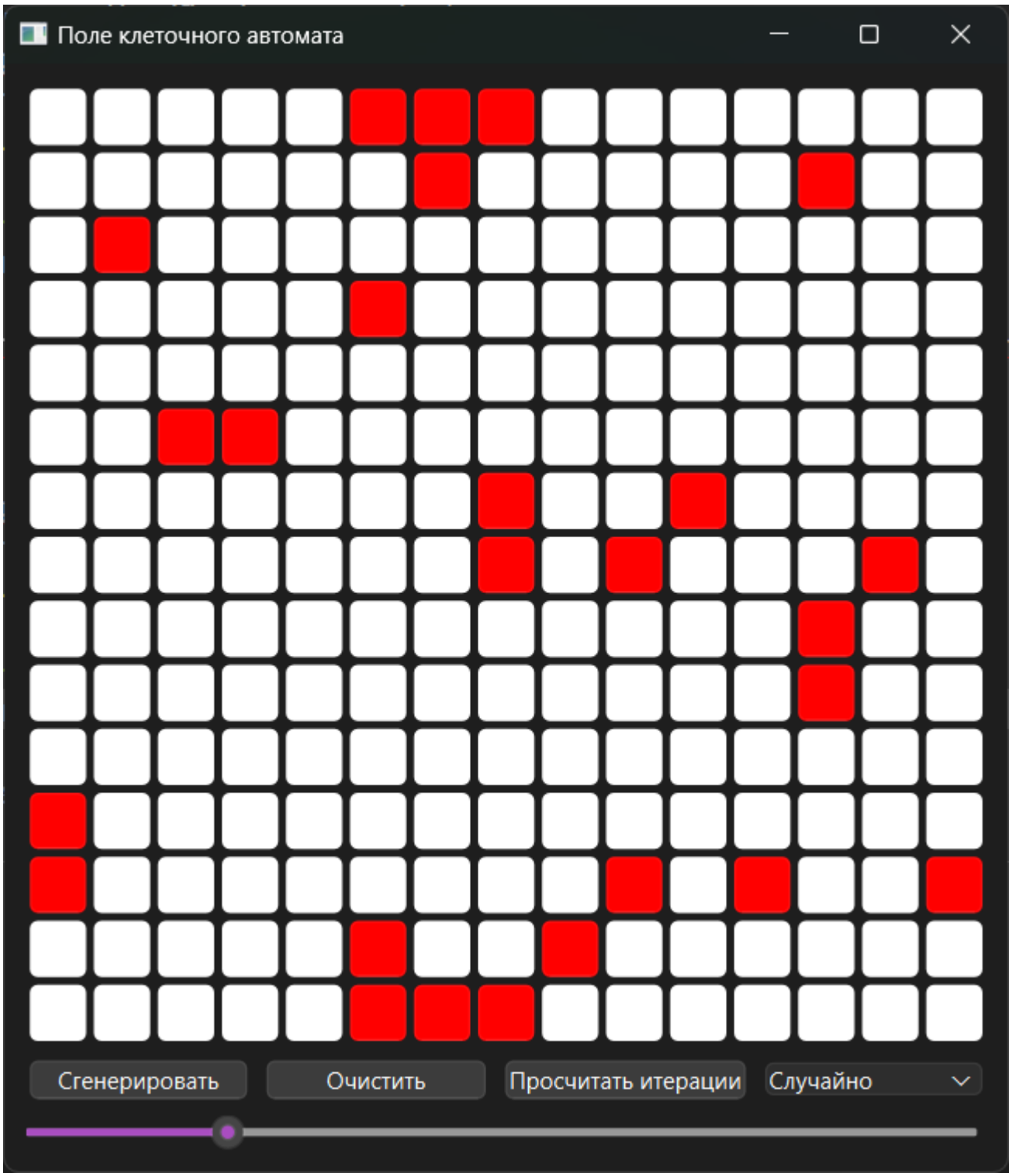

Рис. 18. Эволюция автомата со случайными начальными условиями. 2 итерация

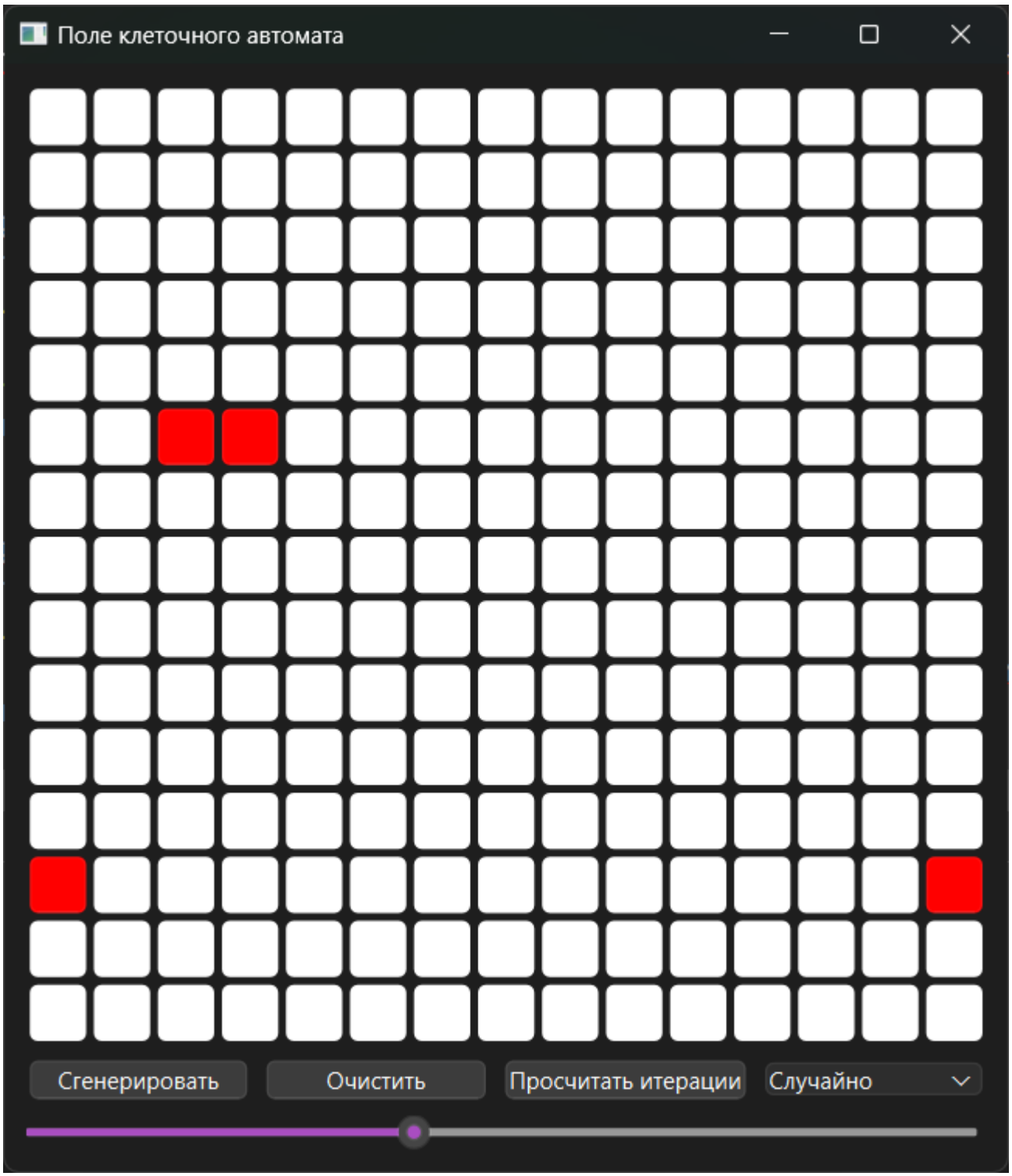

Рис. 19. Эволюция автомата со случайными начальными условиями. 4 итерация

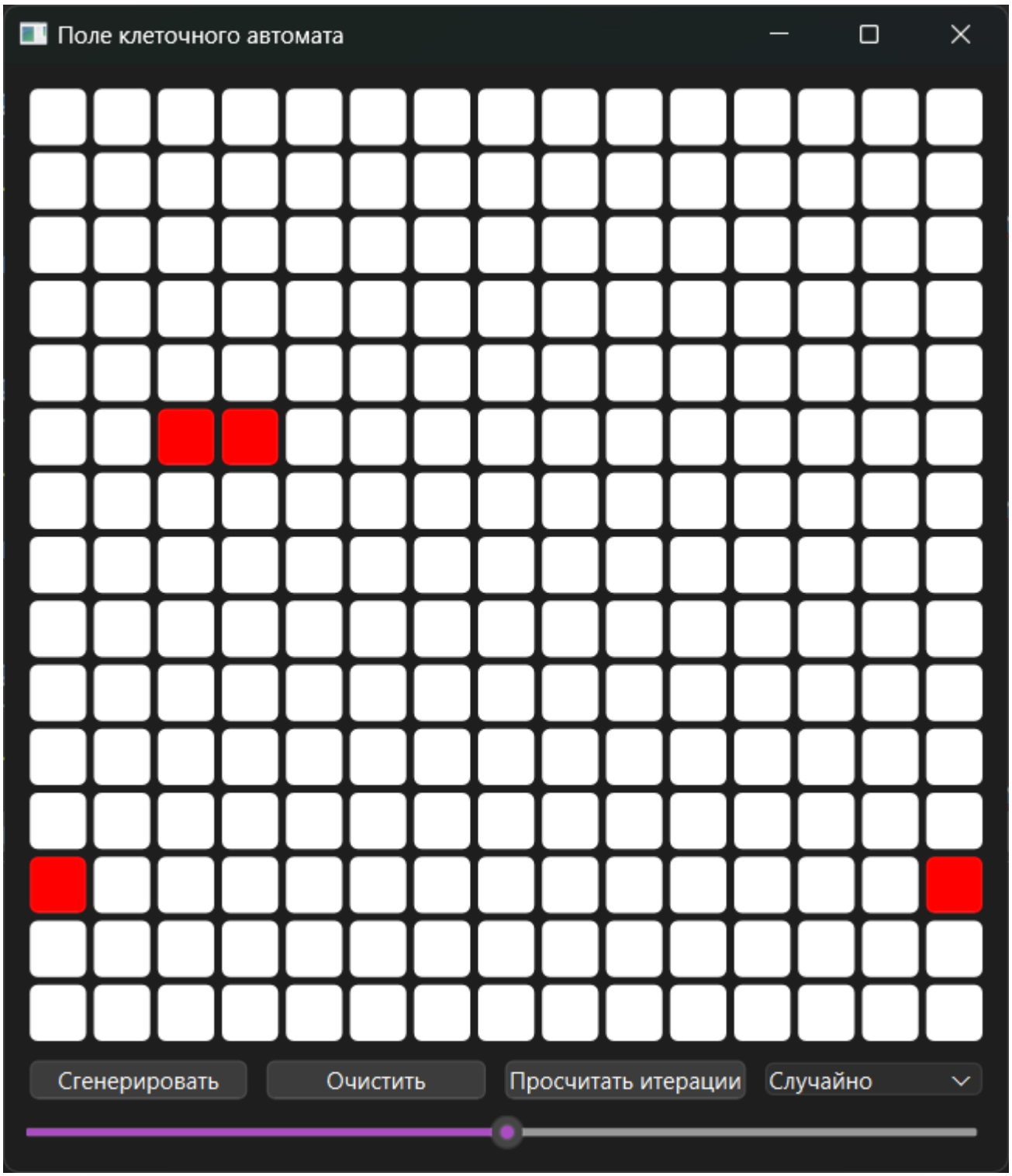

Рис. 20. Эволюция автомата со случайными начальными условиями. 5 итерация

Для заполненного вручную поля автомат снова приходит в статичное состояние, в данном случае к пустому полю (Рис. 21 - Рис. 24).

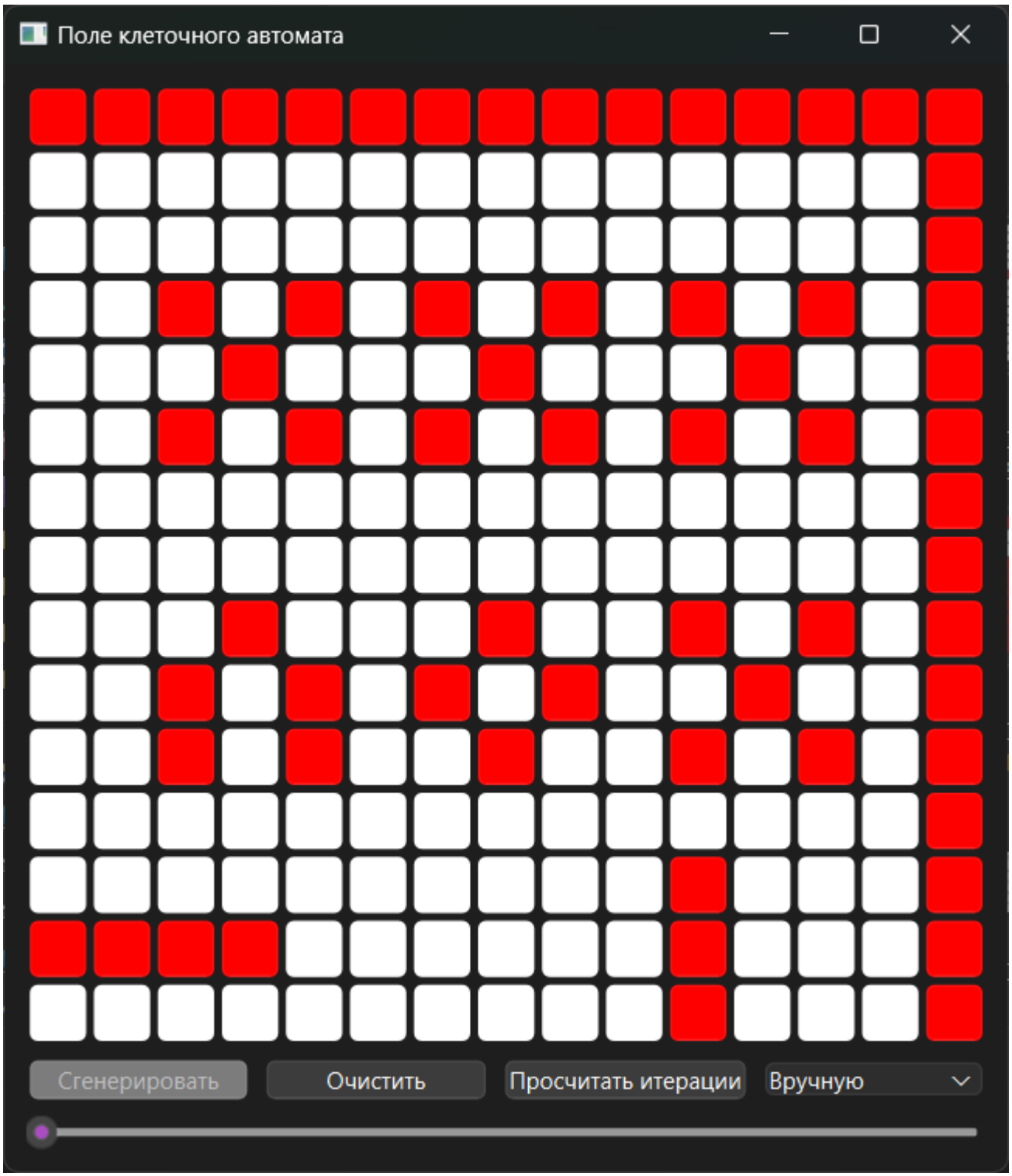

Рис. 21. Эволюция автомата с начальными условиями заданными вручную

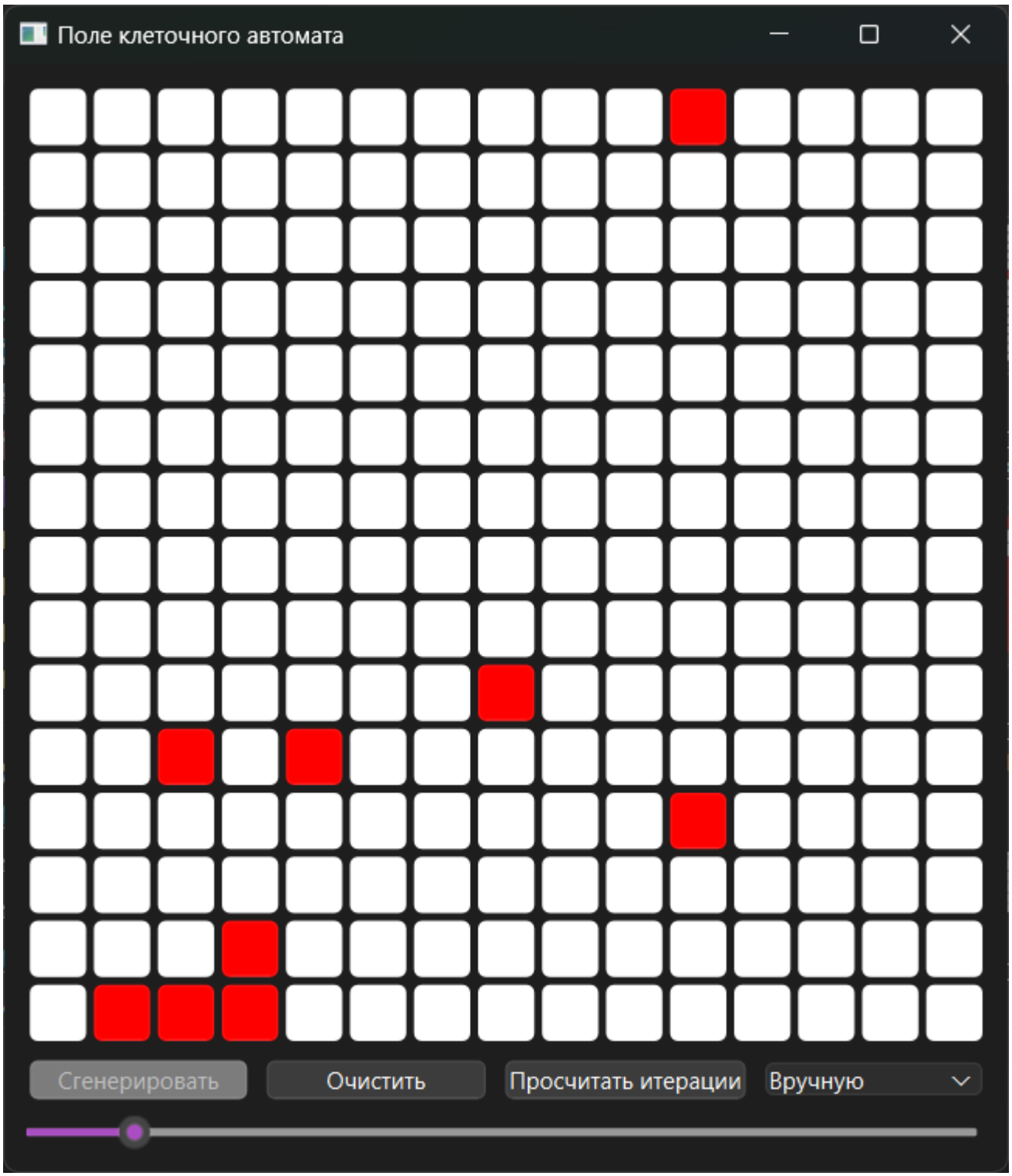

Рис. 22. Эволюция автомата с начальными условиями заданными вручную. 1 итерация

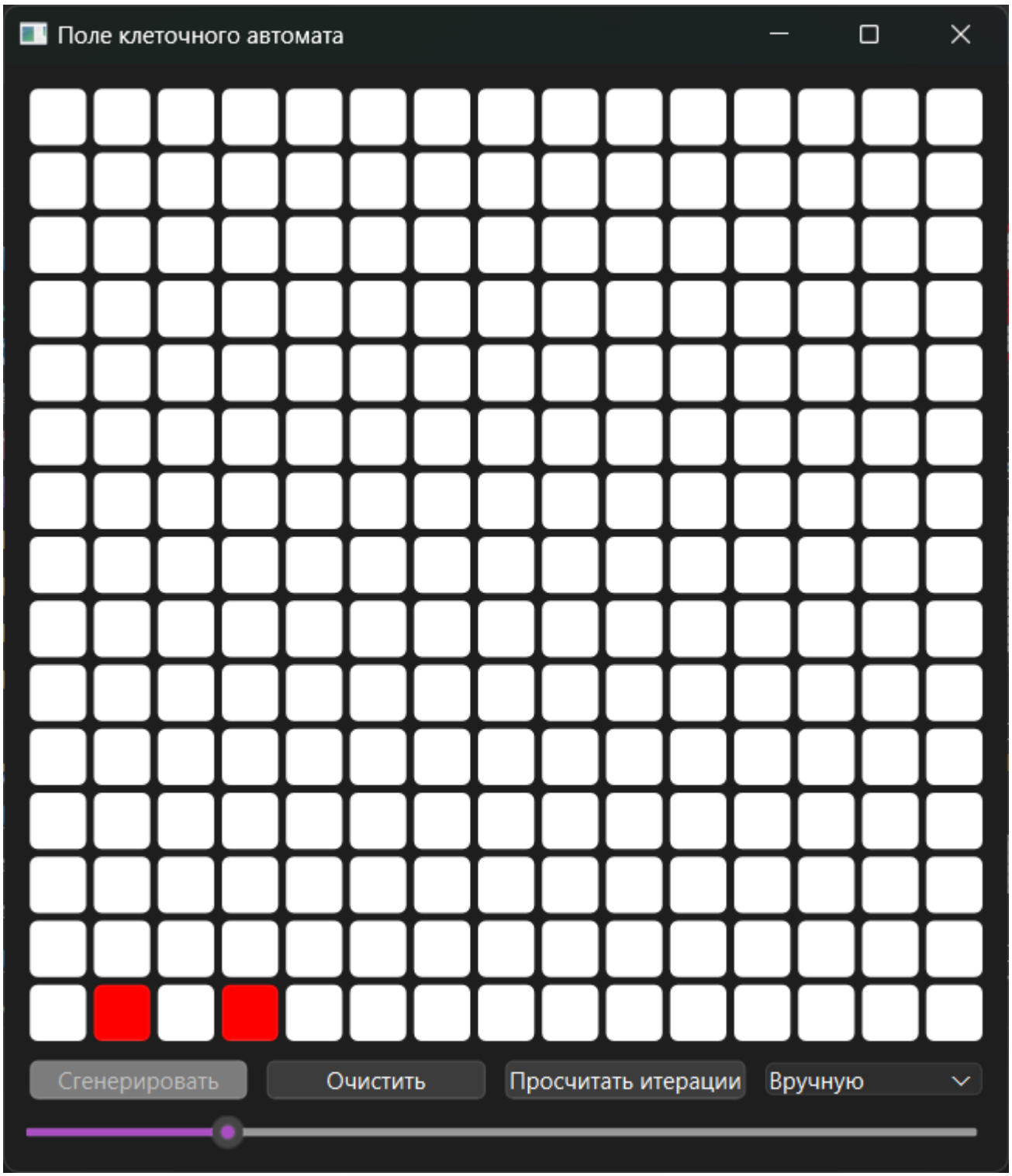

Рис. 23. Эволюция автомата с начальными условиями заданными вручную. 2 итерация

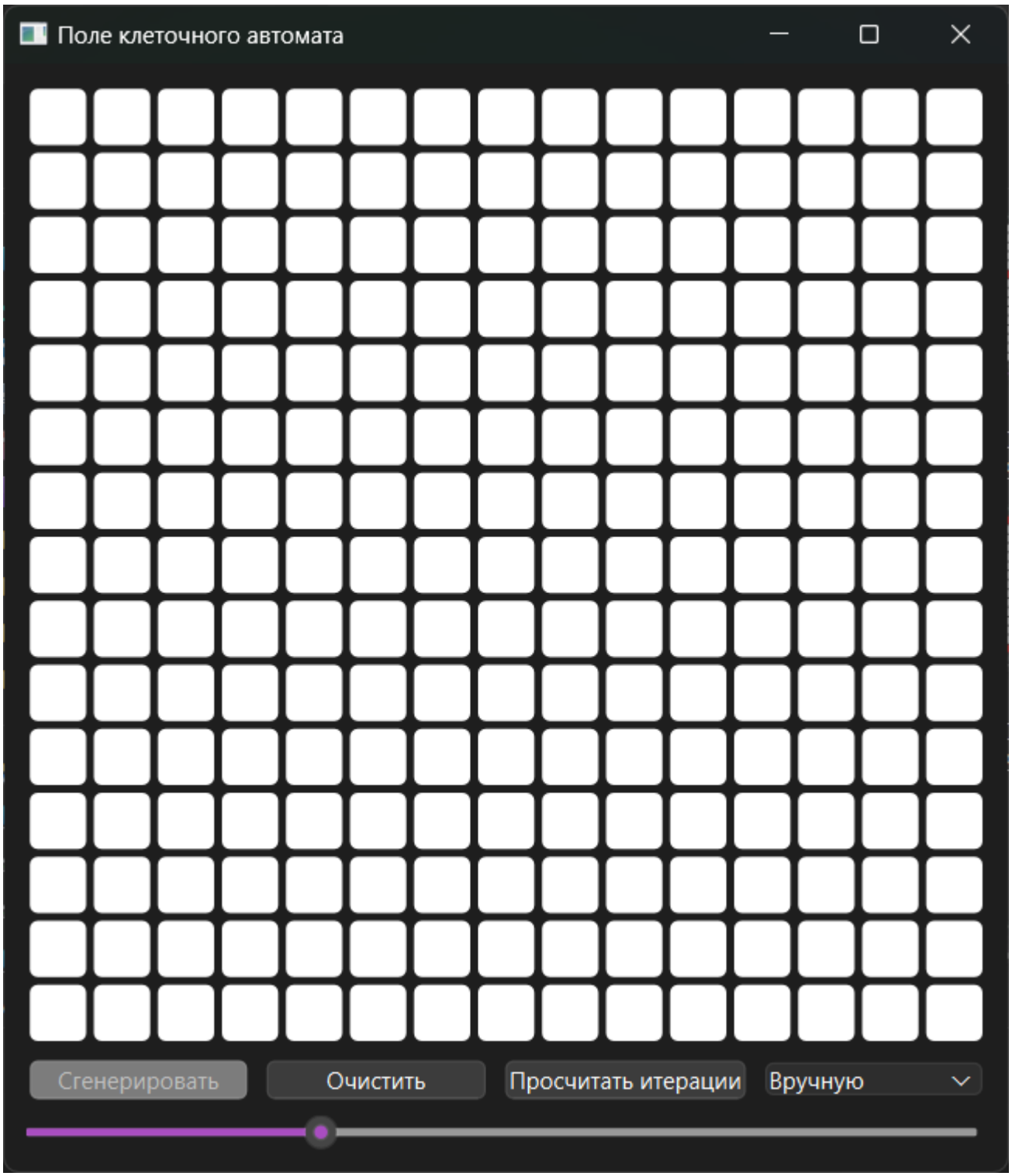

Рис. 24. Эволюция автомата с начальными условиями заданными вручную. 3 итерация

На Рис. 25 изображен график аппроксимации 20 графиков со значениями живых клеток на каждой итерации. Значения были получены при запуске эволюции в 50 итераций для 20 случайно сгенерированных клеточных автоматов размера 15 x 15.

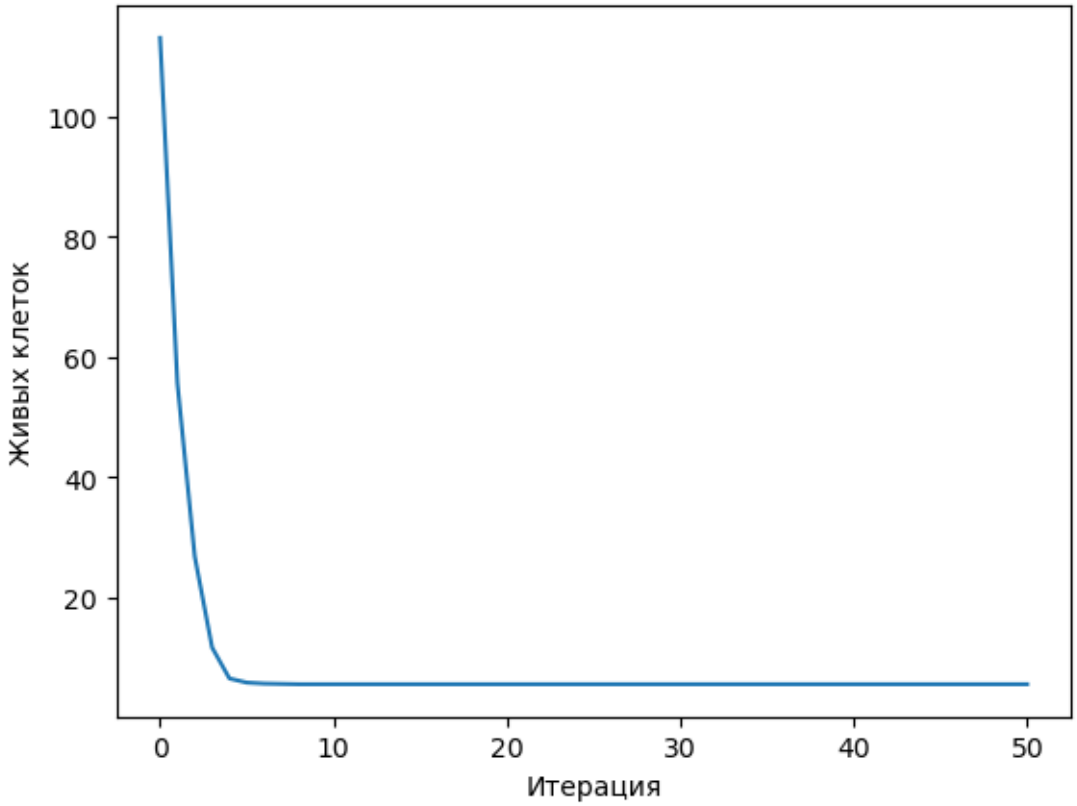

Рис. 25. График аппроксимации 20 графиков эволюции клеточного автомата

Из проведенного вручную анализа и построенного графика можно заключить, что клеточный автомат всегда будет приходить к гомогенному состоянию: либо к пустому полю, либо к полю со статичными фигурами из постоянного количества живых клеток. Таким образом, автомат можно отнести к 1 классу по классификации Вольфрама [5].

Заключение

Во время выполнения лабораторной работы был разработан двумерный клеточный автомат с графическим отображением, с окрестностью фон Неймана, и работающий по правилу 617232. Используются тороидальные граничные условия. Пользователь имеет возможность задать размер поля и количество итераций, а в отдельном окне просматривать состояния автомата по разным итерациям. Также предусмотрен ввод различных начальных условий — как вручную, так и случайным образом, в зависимости от выбора пользователя. Был проведен анализ клеточного автомата, в результате которого установлено, что он относится к первому классу по классификации Стивена Вольфрама.

Достоинства:

- Разделение программы на интерфейсную и вычислительную части.
- Кросплатформенность приложения за счет использования расширения фреймворка Qt для Python.
- Графическое представление клеточного автомата.

Недостатки:

- Для размера поля больше чем 30 x 30 происходит снижение скорости работы приложения в связи с использованием языка Python и кнопок QPushButton в качестве клеток поля.
- Сложный код интерфейса на PyQt, так как не был использован компоновщик форм графического интерфейса Qt Designer.
- Отсутствие буферизации, то есть хранение всех состояний автомата в оперативной памяти даже для большого размера поля и большого количества итераций.

Масштабирование программы:

- Добавить другие единичные и нулевые граничные условия.
- Реализовать алгоритмы сжатия данных или буферизацию, чтобы уменьшать потребление памяти при больших размерах поля и числе итераций.

Программа была разработана в среде разработки PyCharm Community Edition 2023.1 и Python версии 3.11.

Список литературы

[1] Stephen Wolfram, Statistical mechanics of cellular automata // Rev. Mod. Phys. – Princeton : American Physical Society, 1983. – С. 601-644.

[2] Weisstein, Eric W. Moore Neighborhood // MathWorld–A Wolfram Web Resource. - URL: https://mathworld.wolfram.com/MooreNeighborhood.html (дата обращения 01.11.2024)

[3] Weisstein, Eric W. von Neumann Neighborhood // MathWorld–A Wolfram Web Resource. - URL: https://mathworld.wolfram.com/vonNeumannNeighborhood.html (дата обращения 01.11.2024)

[4] Stephen Wolfram, Norman H. Packard, Two-dimensional cellular automata // Journal of Statistical Physics : Springer, 1985. – С. 901-946.

[5] Stephen Wolfram, A New Kind of Science : Wolfram Media, 2002. – 1197 с. – ISBN 1-57955-008-8.