Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

**Игра «Жизнь»**

Пояснительная записка к курсовой работе   
по дисциплине

«Практикум по объектно-ориентированному программированию»

Выполнил

студент гр. 429-3

\_\_\_\_\_\_\_\_\_\_\_\_\_ Бабец А.А.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Руководитель

доцент каф. АОИ, к.ф.-м.н.

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Зариковская Н.В.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

**ЗАДАНИЕ**

на курсовую работу по дисциплине   
«Практикум по объектно-ориентированному программированию»

студенту Бабец Алексею Алексеевичу

группа 429-3, факультет систем управления

Тема проекта: Моделирование игры «Жизнь».

Срок сдачи студентом законченного проекта:\_\_\_\_\_\_\_\_\_\_\_\_\_.

Исходные данные к проекту:

Содержание пояснительной записки: постановка задачи, анализ задачи, алгоритм и блок-схема, обоснованный выбор стека технологий, описание реализации основных процедур и функций, описание интерфейса программы, тестирование программы, заключение.

Дата выдачи задания:«12» февраля 2021г*.*

Руководитель:

Доцент каф. АОИ, к.ф.-м.н.

\_\_\_\_\_\_\_\_\_ Зариковская Н.В.

Задание принял к исполнению:

\_\_\_\_\_\_\_\_\_ Бабец А.А.

«12» февраля 2021 г

**Оглавление**

[1 Постановка задачи 4](#_Toc72486320)

[2 Анализ задачи 5](#_Toc72486321)

[3 Алгоритм и блок-схема 6](#_Toc72486322)

[4 Выбор стека технологий 7](#_Toc72486323)

[5 Описание реализации основных процедур и функций 8](#_Toc72486324)

[6 Описание интерфейса программы 10](#_Toc72486325)

[7 Тестирование программы 11](#_Toc72486326)

[8 Заключение 18](#_Toc72486327)

[Список использованных источников 19](#_Toc72486328)

# Постановка задачи

Реализовать игру «Жизнь». Разрабатываемое приложение представляет собой программную реализацию известной клеточной игры. Приложение должно обеспечивать возможность сохранения текущего состояния поля в файл, загрузки сохранений, редактирование поля, просмотр процесса жизни поля.

# Анализ задачи

Оставить только принцип

Рассмотрим более подробно процесс работы приложения:

1. при запуске приложения создаётся и отрисовывается поле «жизни» - матрица, длина и ширина которой задаются пользователем;
2. при начальной отрисовке поле представляет собой прямоугольник, состоящий из белых (мёртвых) клеток шириной 25 и длиной 50 клеток;
3. живые (зелёные) клетки расставляются пользователем с помощью компьютерной мыши;
4. при нажатии на кнопку «Старт» происходит запуск/остановка периодического пересчёта поля: для каждой клетки по заданной формуле просчитывается её состояние на следующем этапе жизни;
5. соседи крайних клеток, выходящие за отображаемую часть поля, считаются мёртвыми;
6. после каждого пересчёта состояния поля, происходит его переотрисовка, изменения в игровое поле можно вносить только при остановленном процессе жизни;
7. процесс периодического пересчёта и переотрисовки поля автоматически прекращается, когда случается ситуация, что новое поколение не отличается от предыдущего;
8. ещё пользователь может изменять задержку таймера в пределах 100-1000 мс;
9. также пользователь может сохранить текущее состояние поля в текстовый файл в виде размеров матрицы и самой матрицы;
10. сохранённые состояния можно загружать в программу;
11. для выхода из игры необходимо нажать кнопку выхода.

# Алгоритм и блок-схема

Основной функцией данной программы является функция, генерирующая следующее поколение клеток. Ниже описан её алгоритм.

Для каждой клетки происходит проверка всех восьми соседних клеток, при этом считается количество «живых» соседей. Клетки, находящиеся вне видимого поля считаются мёртвыми. После подсчёта «живых» соседей идёт генерация по следующим правилам:

* если у «мёртвой» клетки ровно три живых соседа, то она будет «живой» в следующем поколении;
* если у «живой» клетки меньше двух или больше трёх «живых» соседей, то она будет «мёртвой» в следующем поколении.

# Выбор стека технологий

Для написания программы был выбран язык C++ так как он кроссплатформенный, поддерживает разные технологии и стили программирования. Также он входит в десятку самых популярных языков программирования по рейтингам TIOBE и PYPL.

Также выбрана среда разработки QtCreator ввиду её кроссплатформенности, стабильности и скорости работы.

Добить полторы страницы

# Описание реализации основных процедур и функций

Для решения поставленной задачи опишем реализацию следующих процедур и функций:

- генерация следующего поколения public int nextGen(), класса life\_place;

- перерисовка поля по текущему состоянию поля void drawplace(bool new), класса mainwindow;

Генерация следующего поколения происходит с помощью двух вложенных циклов for следующим образом:

При первом и единственном полном проходе по всему полю для каждой клетки считается количество её «живых» соседей, далее, в зависимости от количества соседей и того, «жива» ли клетка на данный момент, принимается решение о её состоянии в следующем поколении. Тут действуют следующие правила:

* если у «мёртвой» клетки ровно три живых соседа, то она будет «живой» в следующем поколении;
* если у «живой» клетки меньше двух или больше трёх «живых» соседей, то она будет «мёртвой» в следующем поколении.

Так в процессе пересчёта нового поколения происходит подсчёт клеток, состояние которых изменилось, это количество является выводом функции. Как только будет сгенерированное новое поколение без изменений, обработчик события таймера остановит пересчёт и переотрисовку.

Переотрисовка поля начинается с копирования объекта класса QPixmap с объекта класса QExLabel, отображающего поле, далее идёт инициализация объекта класса QPainter, с помощью которого и будет производиться вся отрисовка. Затем происходит очистка полученного QPixmap методом eraseRect(int x, int y, int width, int height) класса QPainter. Далее с помощью двух циклов for и метода drawLine(int x1, int y1, int x2, int y2) класса QPainter рисуется сетка поля. Затем с помощью двух вложенных циклов for производится обход матрицы поля и отрисовка зелёных «живых» клеток на соответствующих координатах сетки с помощью метода drawRect(QRect rectangle) класса QPainter. В конце функции объекту класса QExLabel находящемуся в главном окне программы присваивается изменённый pixmap, который и отображается.

# Описание интерфейса программы

Окно, открывающееся при запуске программы представлено на рисунке 6.1.

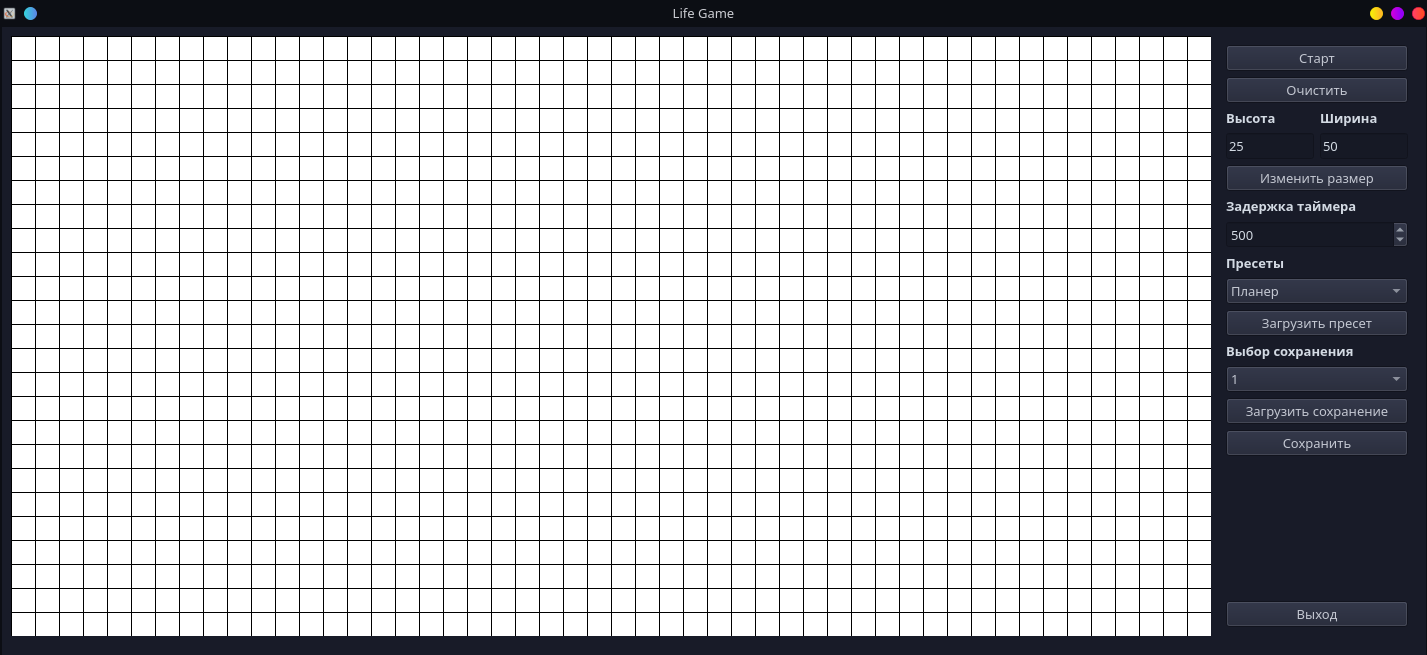


Рисунок 6.1 – интерфейс программы

Кнопка «Старт» (она же «Стоп») запускает или останавливает процесс «жизни», то есть просчёта следующего поколения и переотрисовки поля.

Кнопка «Очистить» очищает поле от живых клеток.

С помощью объектов QTextEdit, подписанных с помощью объектов QLabel, и кнопки «Изменить размер» пользователь может изменить размер поля.

С помощью объекта QComboBox можно выбрать один из имеющихся пресетов поля.

Кнопка «Загрузить пресет» загружает выбранный пресет и отрисовывает его.

Второй объект QComboBox позволяет выбрать сохранение для загрузки.

Кнопка «Загрузить сохранение» загружает выбранное сохранение.

Кнопка «Сохранить» сохраняет текущее состояние поля в файл.

Слева находится само «поле» жизни, на котором и отрисовываются все клетки. Так же на этом поле пользователь может сам расставлять живые клетки с помощью оптического манипулятора типа мышь.

# Тестирование программы

Тестирование проводилось с различными входными данными с целью выявления и предотвращения ошибок в работе программы. При представлении тестов были пропущены некоторые состояния поля для уменьшения количества картинок.

Первый тест: входные данные – рисунок 7.1. Дальнейшие состояния поля рисунки 7.2 – 7.6.

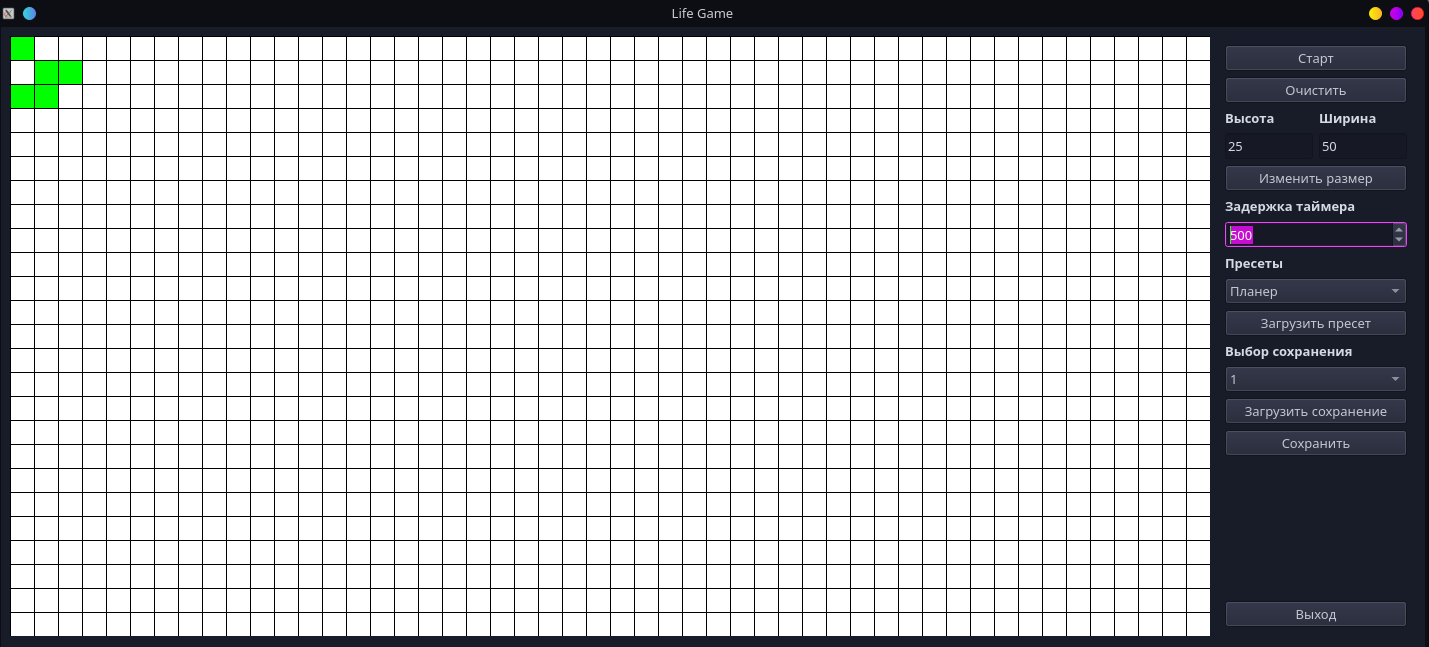


Рисунок 7.1 – входные данные.

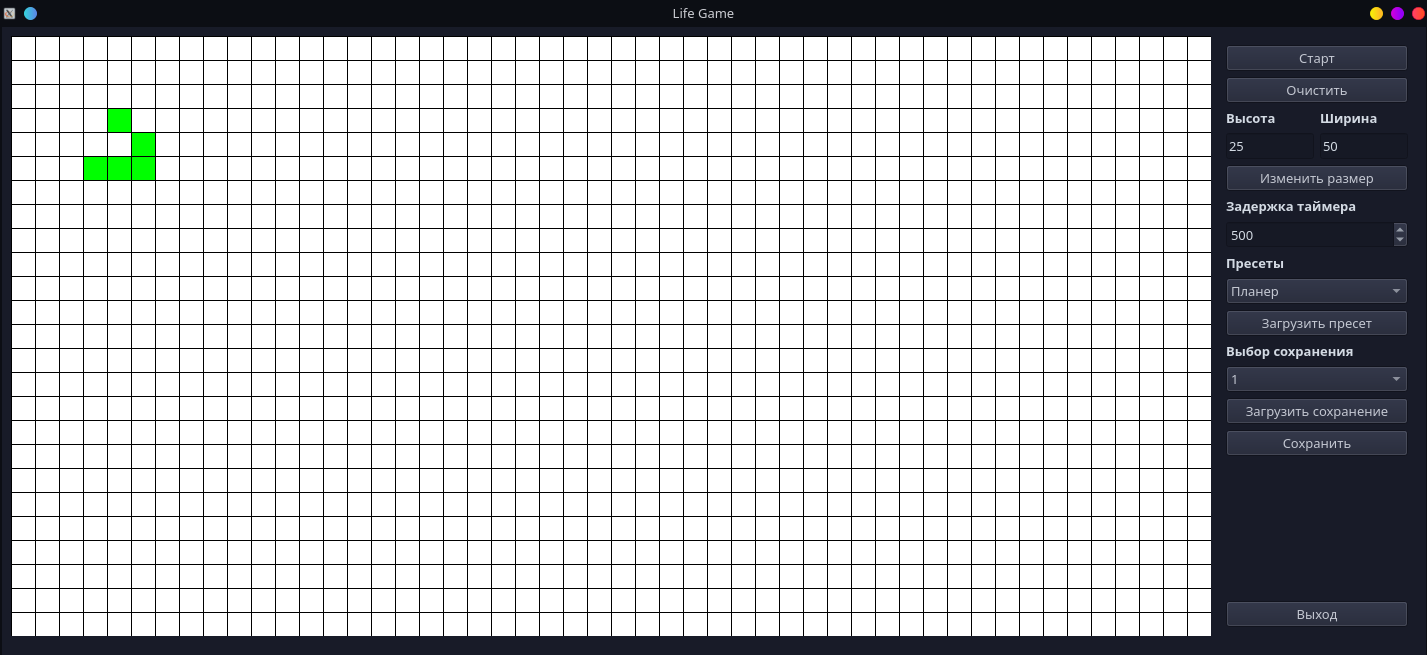


Рисунок 7.2

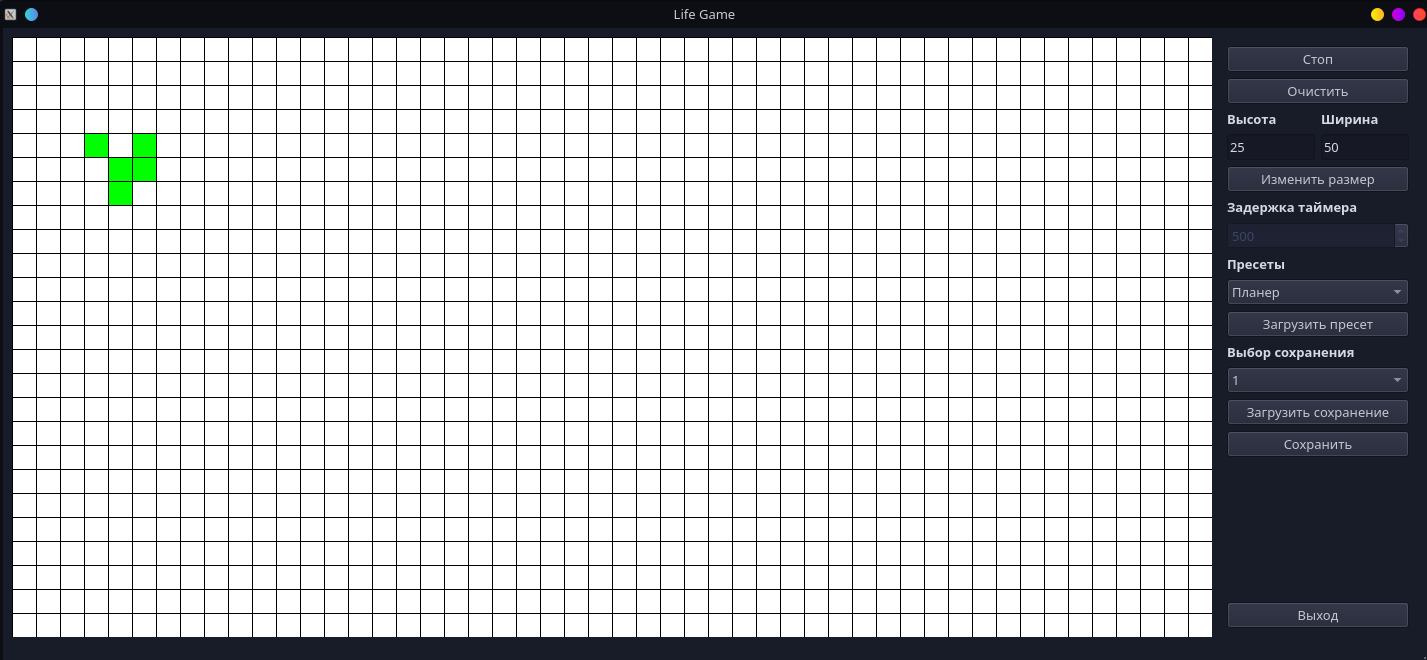


Рисунок 7.3

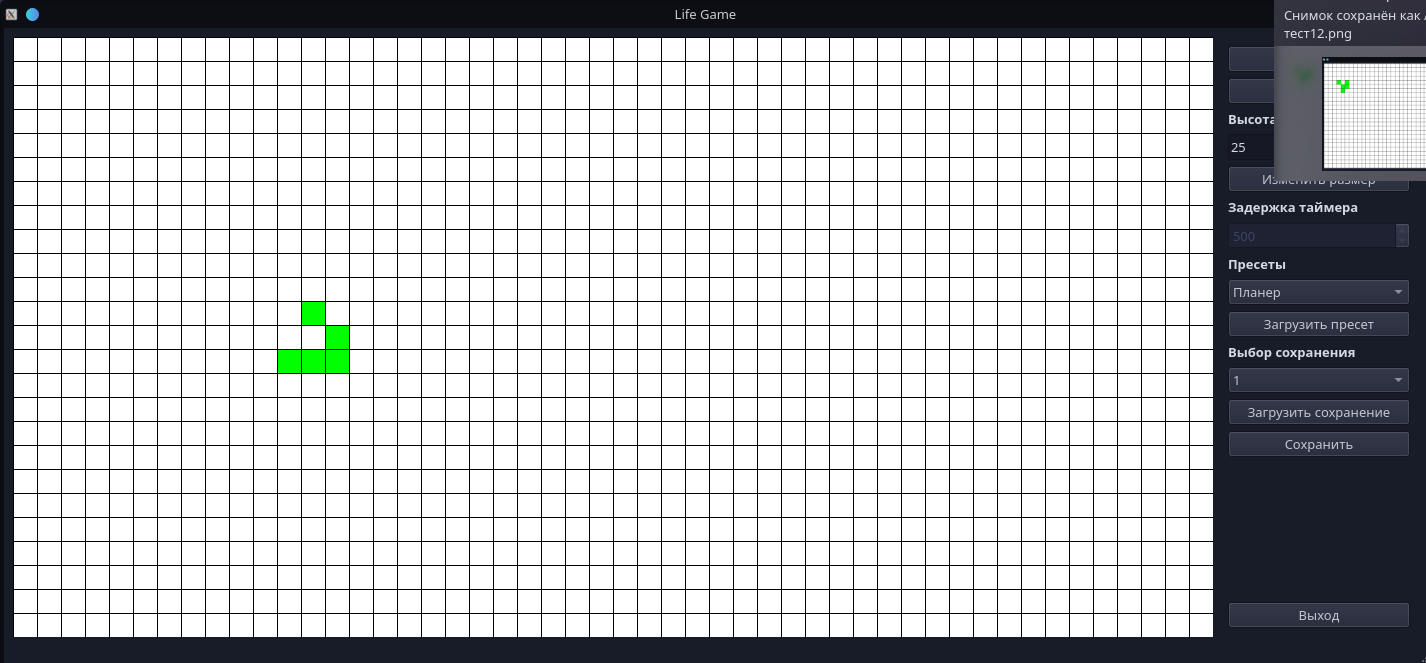


Рисунок 7.4

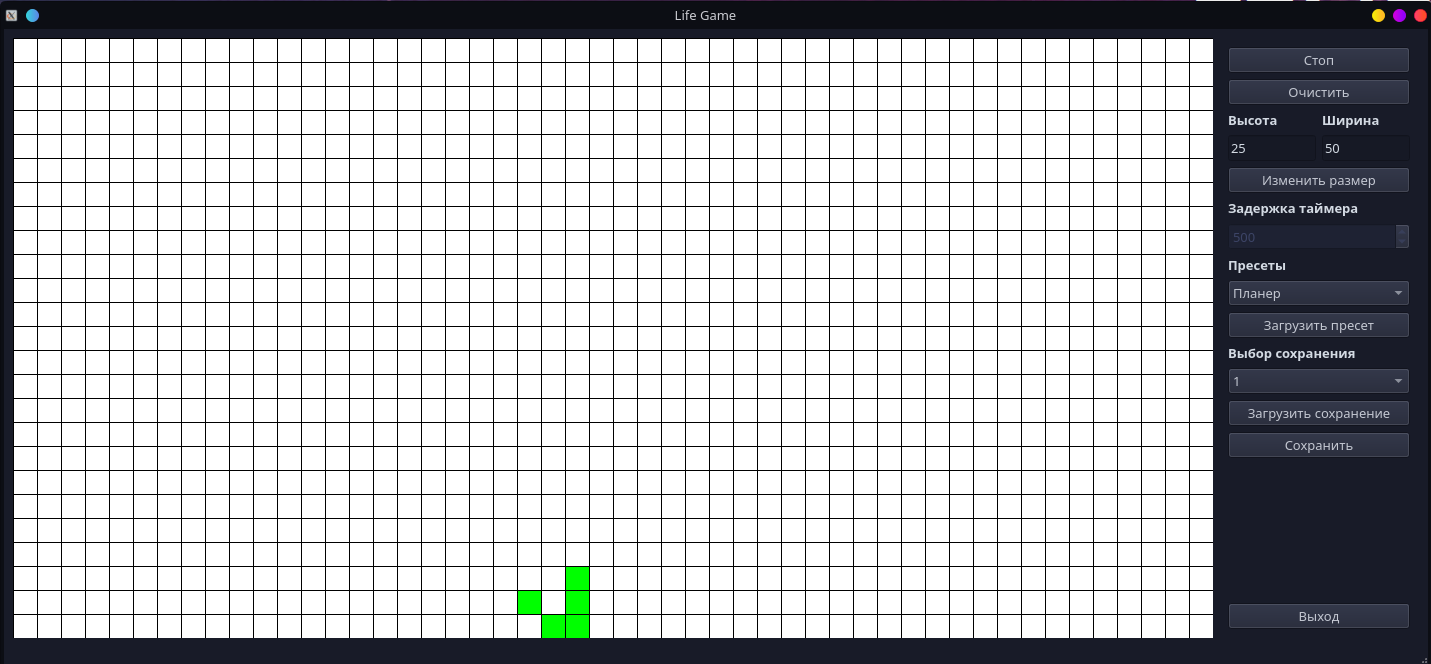


Рисунок 7.5

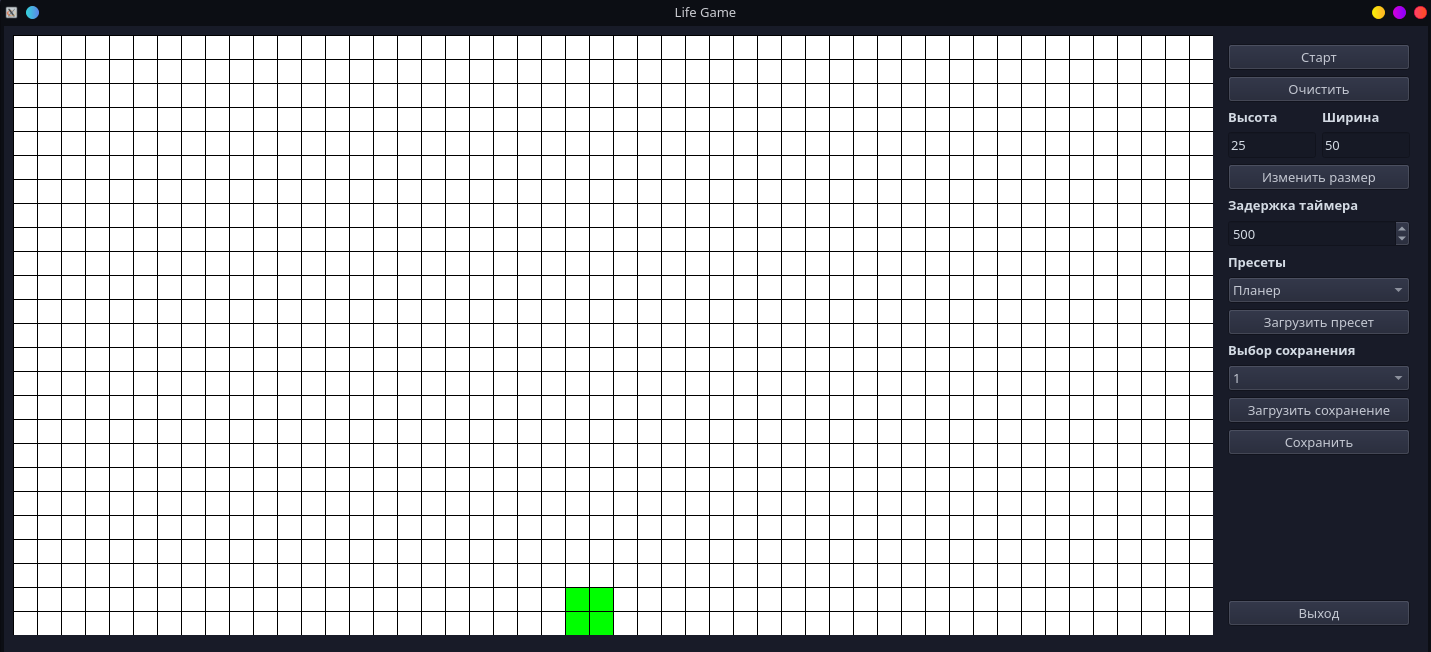


Рисунок 7.6 – конец первого теста.

В первом тесте был использован пересет с фигурой «планер». Данная фигура должна передвигаться вниз и вправо по диагонали, что мы и наблюдаем.

Тест второй: использован пресет, содержащий планерное ружьё Госпера. Рисунок 7.7 – входные данные. Дальнейшие состояния поля - рисунки 7.8 и 7.9.

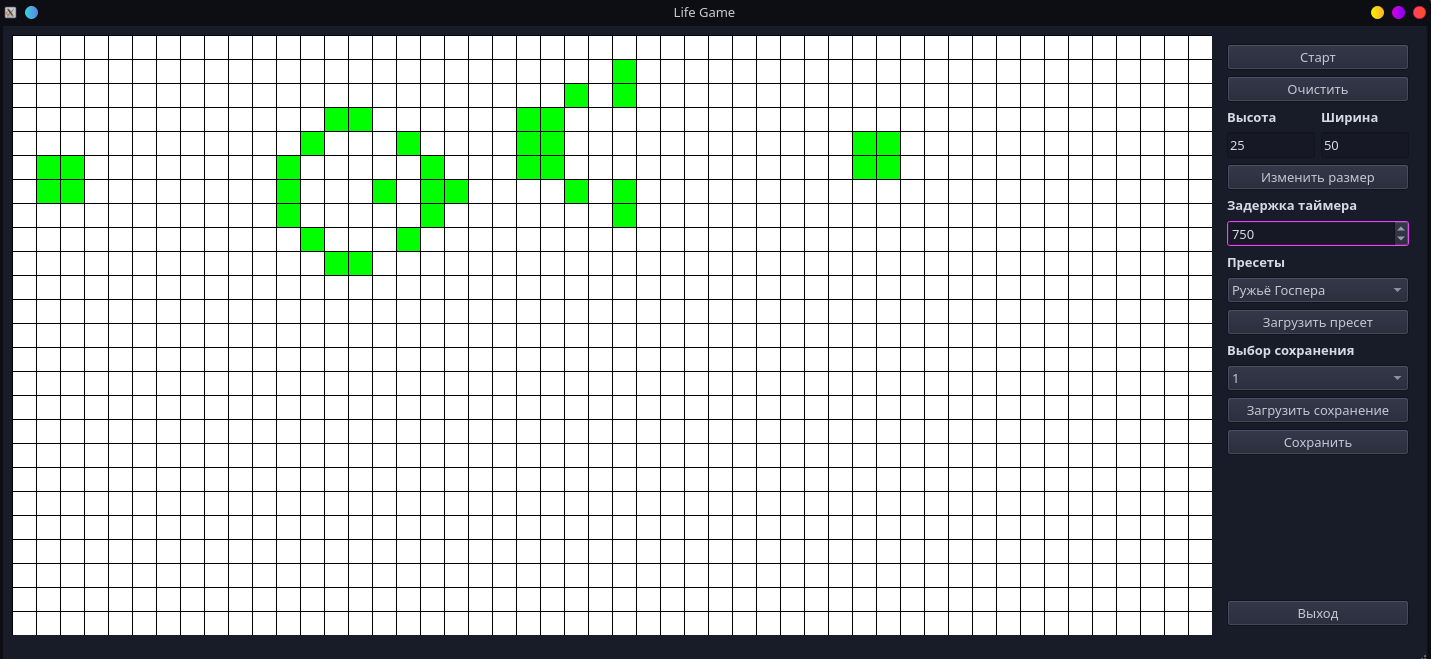


Рисунок 7.7 – входные данные.

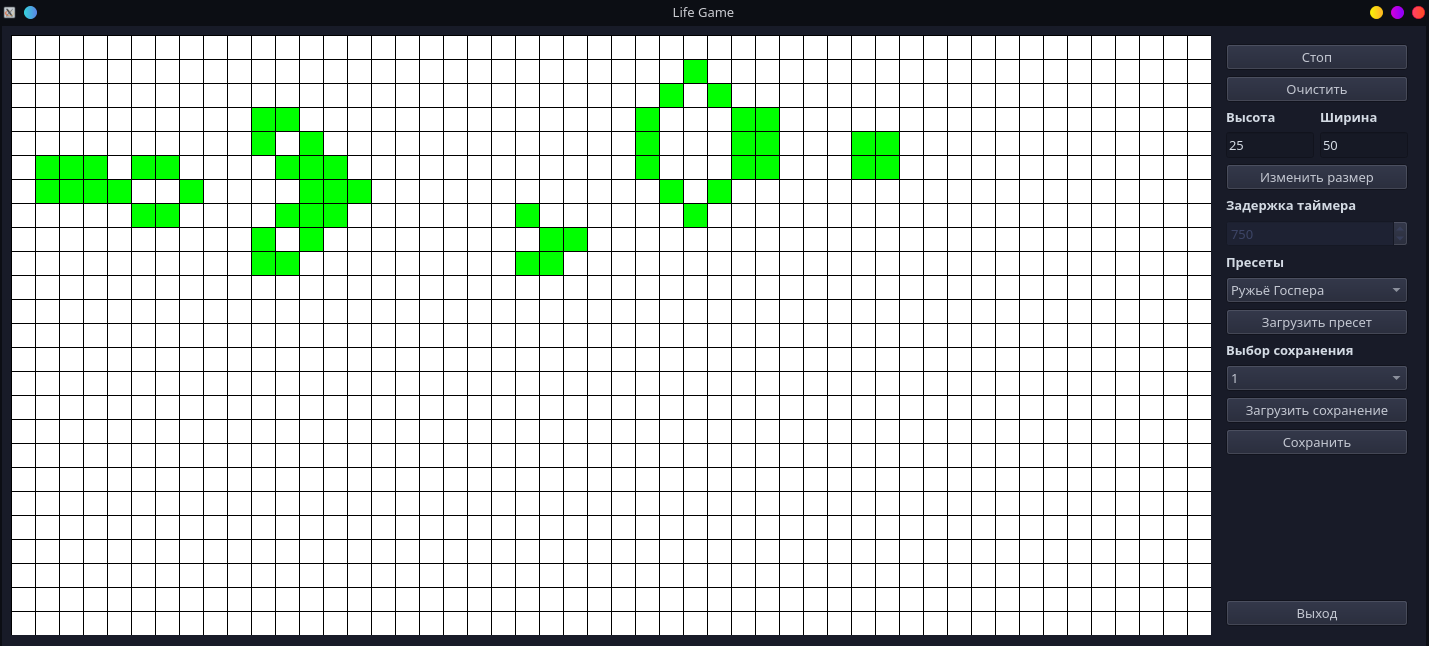


Рисунок 7.8

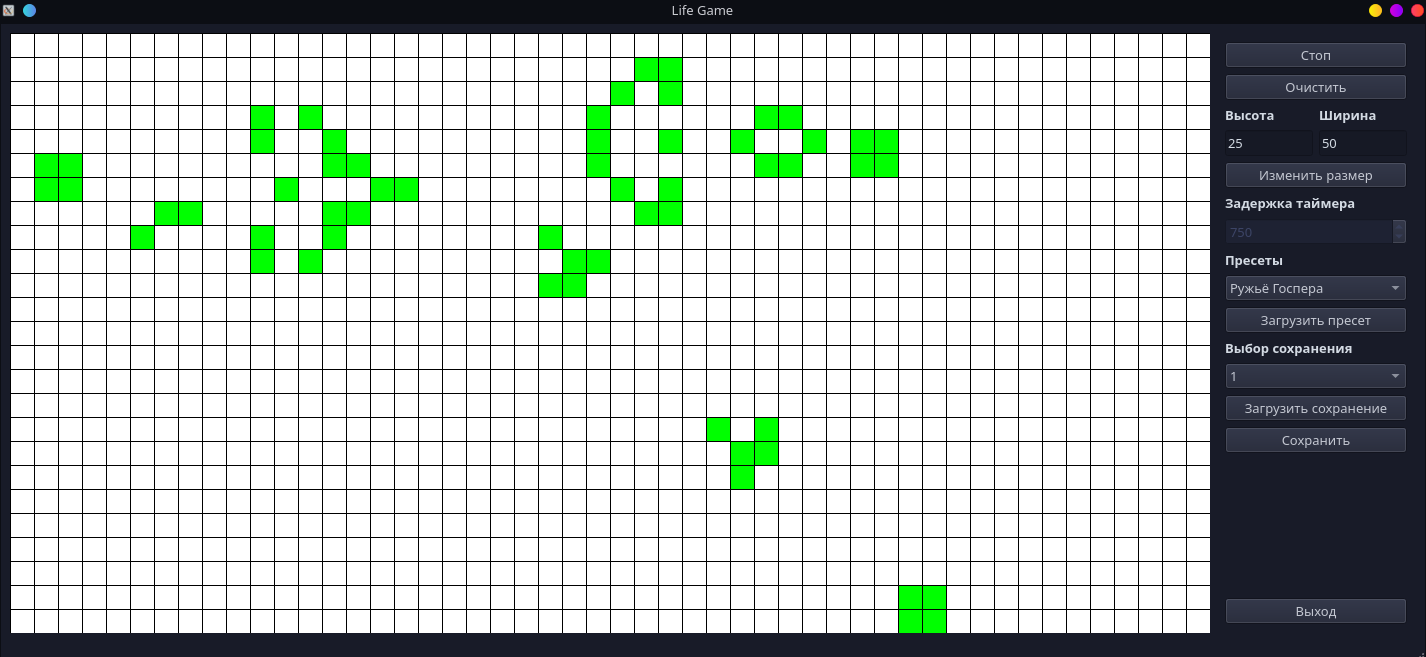


Рисунок 7.9 – конец второго теста.

Так как планерное ружьё Госпера может работать бесконечно, тест был остановлен, когда первый планер «разбился».

Тест третий: использована фигура, обнаруженная при тестировании во время написания программы. Рисунок 7.10 – входные данные. Дальнейшие состояния поля - рисунки 7.11 – 7.16.

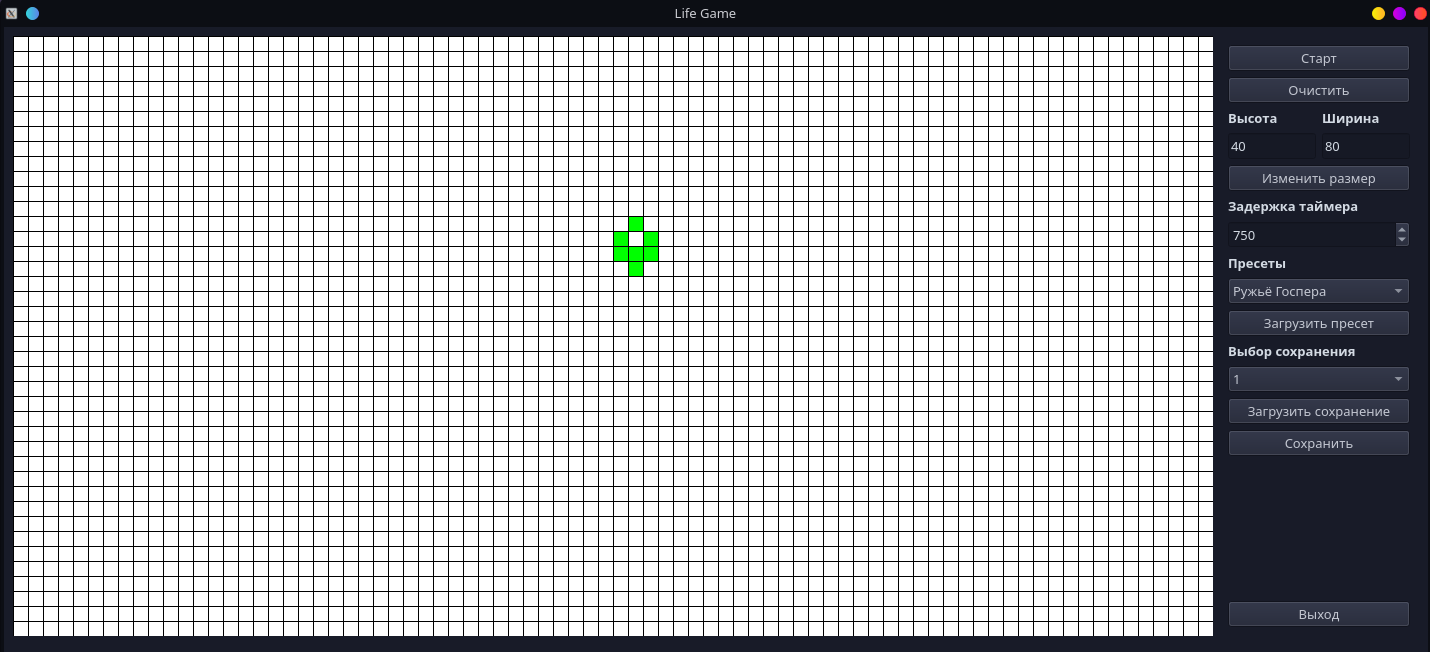


Рисунок 7.10 – входные данные.

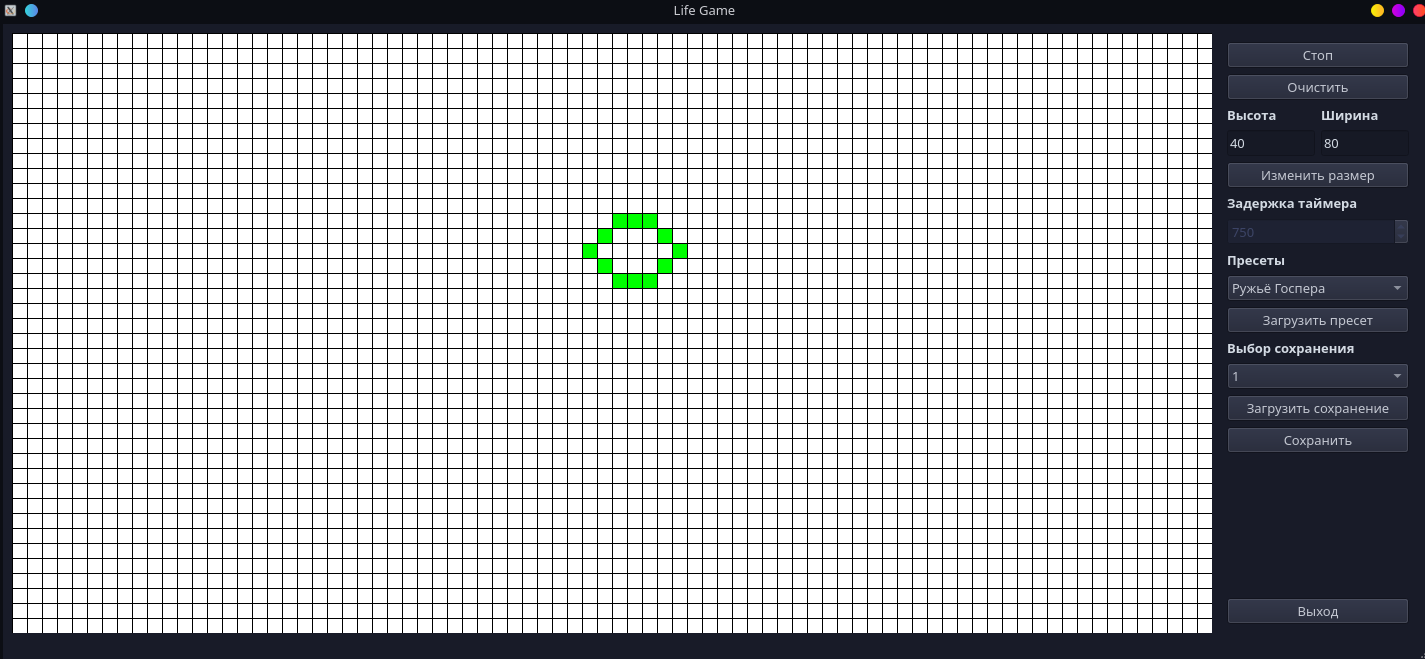


Рисунок 7.11

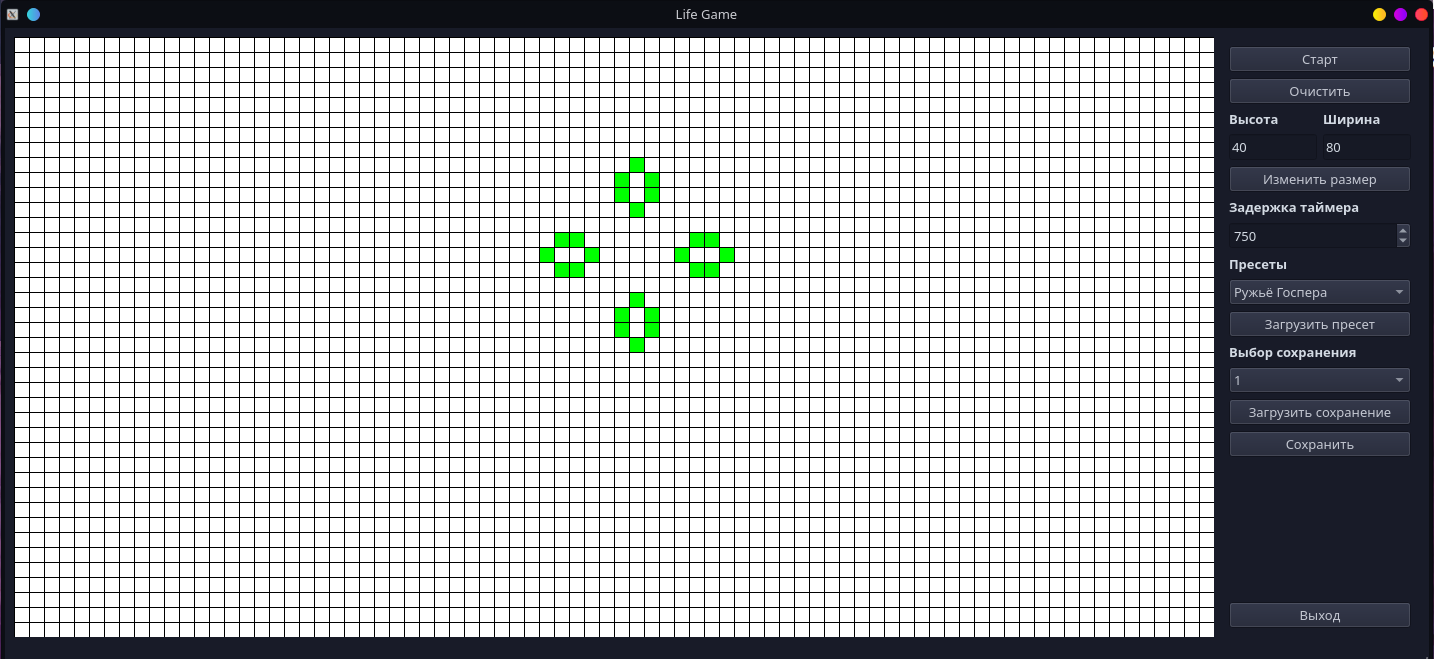


Рисунок 7.12

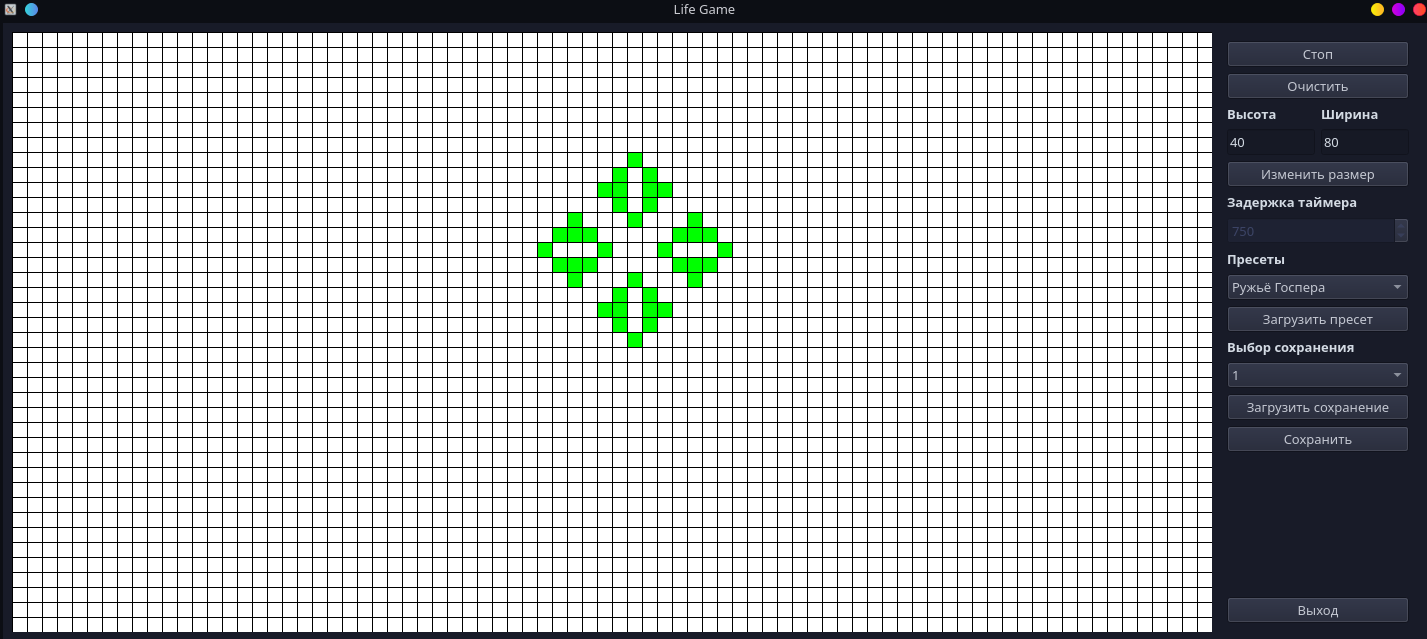


Рисунок 7.13

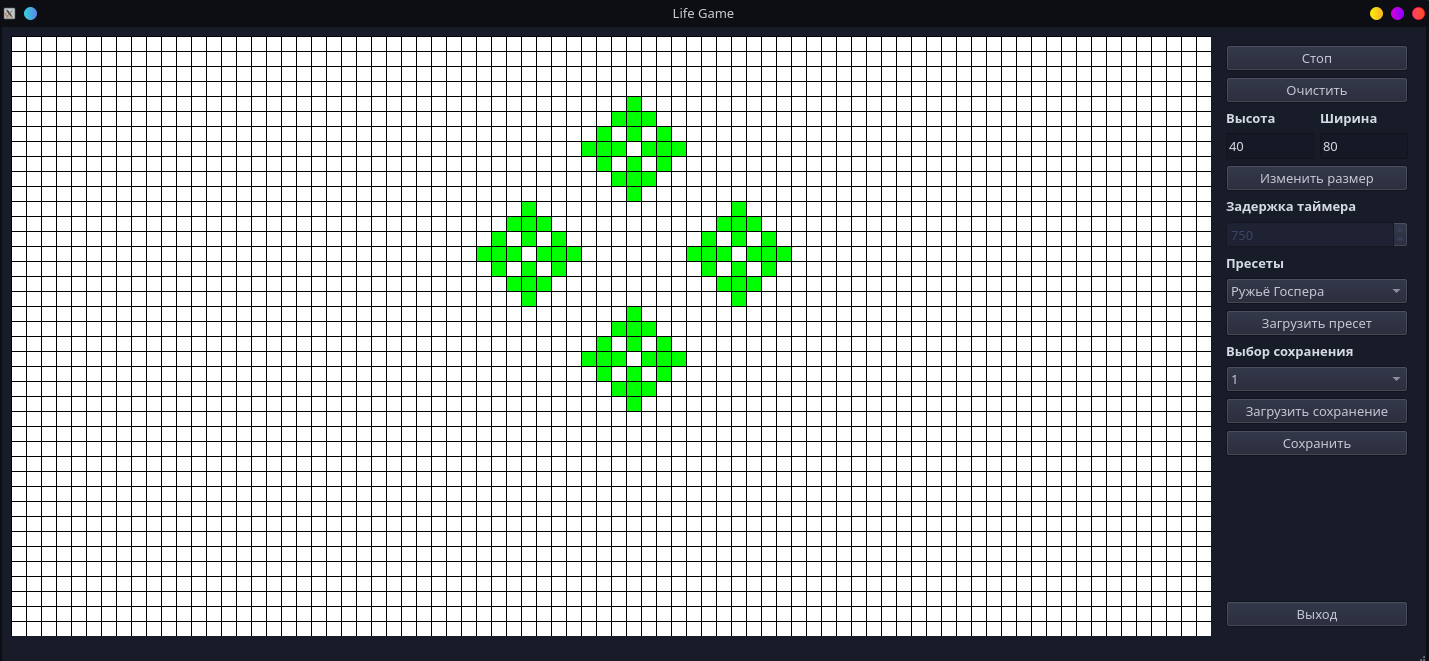


Рисунок 7.14

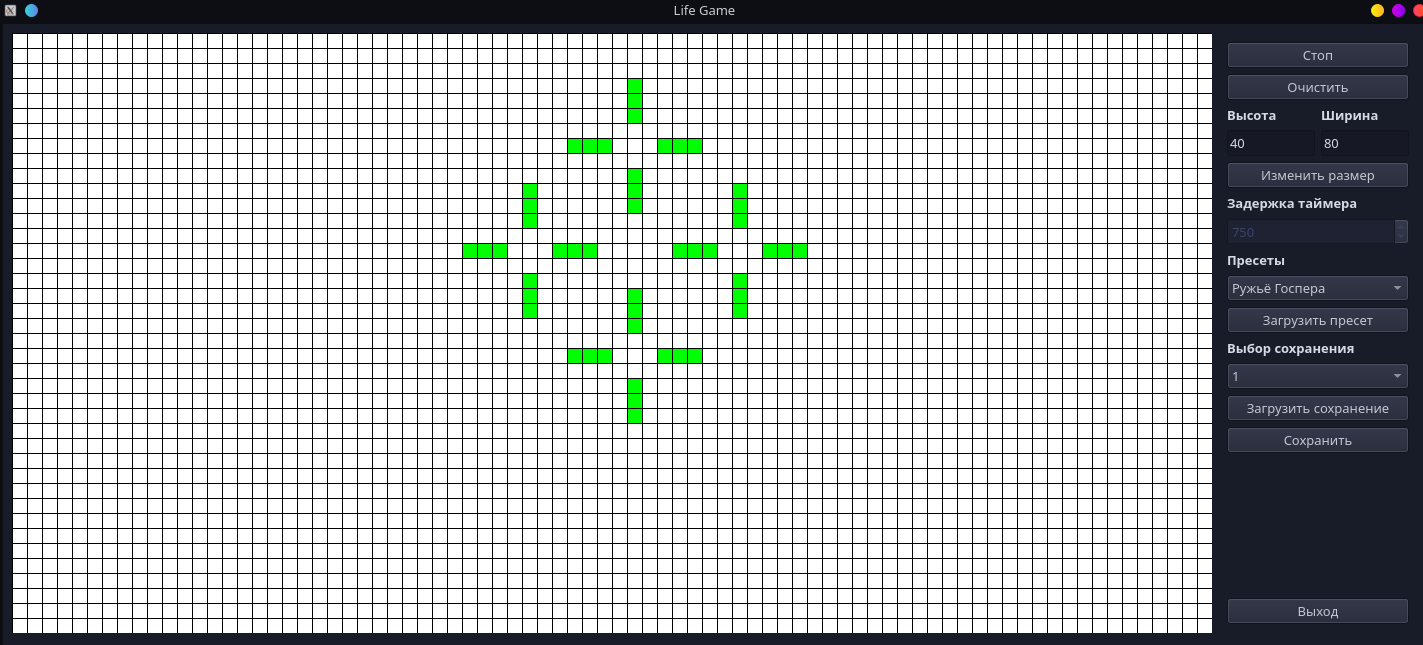


Рисунок 7.15

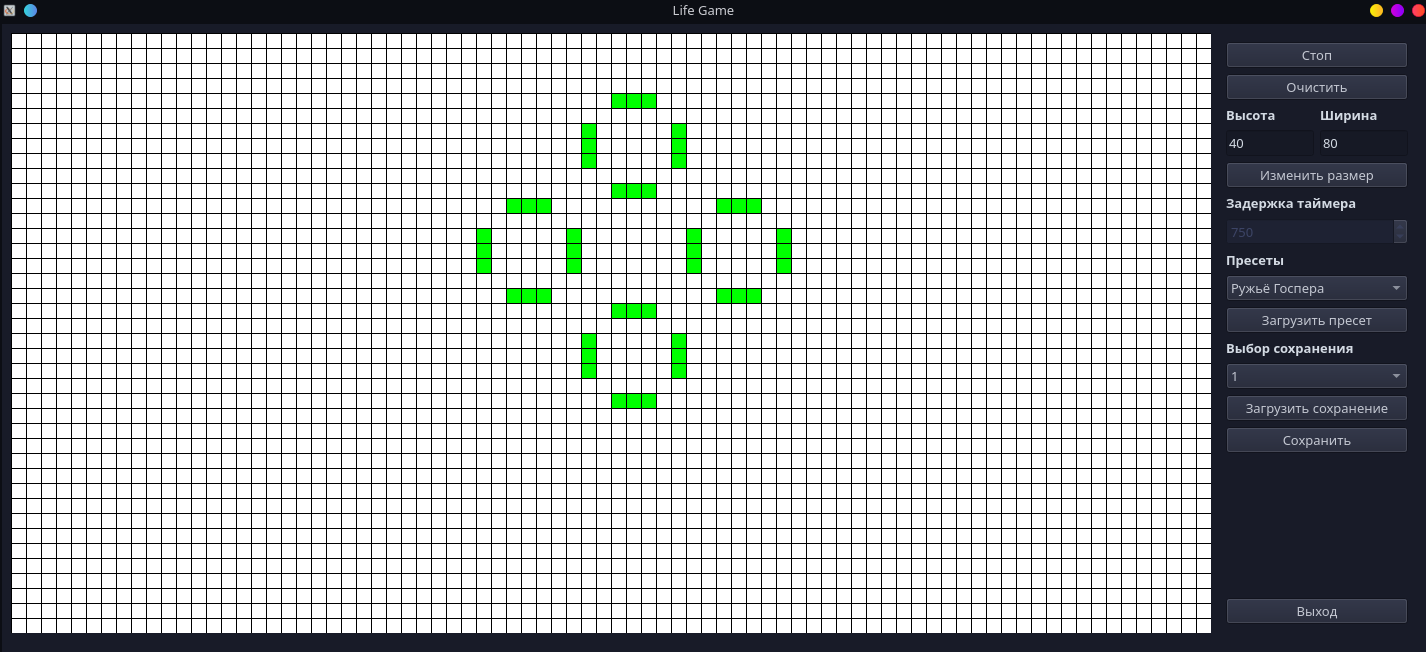


Рисунок 7.16 – конец третьего теста.

Судя по тестам, программа работает правильно, все поставленные задачи выполнены.

# Заключение

В ходе выполнения курсового проекта по дисциплине «Практикум по объектно-ориентированному программированию», была реализована программа «Жизнь», которая отвечает заявленным требованиям и выполняет поставленную перед ней задачу. Были изучены: возможности языка программирования C++, [межплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) QtCreator, а также требования к оформлению курсовой работы.

# Список использованных источников

1. QT Documentation [Электронный ресурс] – Режим доступа: <https://doc.qt.io> (дата обращения 13.04.2021)
2. Архив номеров научно-популярного физико-математического журнал «Квант» [Электронный ресурс] – Режим доступа: <http://kvant.mccme.ru/index.htm> (дата обращения 10.03.2021)
3. Добить до 10ти