Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Казанский (Приволжский) федеральный университет»

*Институт вычислительной математики и информационных технологий*

**ОТЧЕТ**

**по производственной (проектно-технологической) практике**

Обучающийся Фаррахова Л.Ф., гр. 09-222 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от КФУ

ассистент кафедры прикладной математики и искусственного интеллекта Царьков М.В.

Оценка за практику \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата сдачи отчета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc167403687)

[ПОСТАНОВКА ЗАДАЧИ 4](#_Toc167403688)

[ХОД РАБОТЫ 6](#_Toc167403689)

[ЗАКЛЮЧЕНИЕ 17](#_Toc167403690)

[СПИСОК ЛИТЕРАТУРЫ 18](#_Toc167403691)

# ВВЕДЕНИЕ

Проектно-технологическая практика проходила в Казанском (Приволжском) федеральном университете, в институте вычислительной математики и информационных технологий, на кафедре прикладной математики и искусственного интеллекта в период с 15.02.2024 по 23.05.2024.

В ходе проектно-технологической практики были поставлены следующие цели:

1. Получение опыта в разработке приложений на языке C++;
2. Освоение библиотеки raylib для разработки графических приложений и игр;
3. Изучение лучших практик разработки программного обеспечения, включая написание чистого, понятного и хорошо документированного кода, применение принципов объектно-ориентированного программирования;
4. Изучение принципов работы систем контроля версий и наработка практических навыков в их использовании для работы над проектами;
5. Развитие умений в поиске и изучении дополнительной информации, материалов и инструментов, необходимых для разработки приложений на C++.

# ПОСТАНОВКА ЗАДАЧИ

Игра «Game of life» - клеточный автомат, придуманный английским математиком Джоном Конвеем в 1970 году. Это игра без игроков, в которой человек создаёт начальное состояние, а потом лишь наблюдает за её развитием.

Распределение «живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по правилам.

В пустой («мёртвой») клетке, с которой соседствуют три живые» клетки, зарождается жизнь. Если у «живой» клетки есть две или три «живые» соседки, то эта клетка продолжает жить. В противном случае (если «живых» соседей меньше двух или больше трёх) клетка умирает («от одиночества» или «от перенаселённости»).

Игрок не принимает активного участия в игре. Он лишь расставляет или генерирует начальную конфигурацию «живых» клеток, которые затем изменяются согласно правилам.

Возможности приложения:

1. Графический интерфейс:

* Отображение игрового поля с клетками;
* Визуализация состояния клеток («живые» или «мертвые»).

1. Режим случайного размещения клеток:

* Случайное размещение живых клеток на игровом поле при старте;
* Добавление новых клеток кликами мыши;
* Генерация случайного количества клеток на поле с помощью клавиш.

1. Режим ручного размещения клеток:

* Ручное размещение клеток пользователем кликами мыши;
* Управление запуском или остановкой эволюции клеток с помощью клавиш.

1. Управление игрой:

* Запуск и остановка игры;
* Очистка игрового поля;
* Регулировка скорости;
* Скрытие/раскрытие меню;
* Переключения между режимами игры.

Для реализации этих возможностей были выполнены следующие шаги:

1. Изучение концепций игры, анализ требований и особенностей реализации клеточных автоматов;
2. Проектирование основных классов и структур данных, определение взаимодействия между компонентами приложения;
3. Создание классов для представления клеток и игрового поля, реализация эволюции клеток на основе правил игры;
4. Установка и настройка библиотеки raylib, разработка функций для отрисовки клеток и игрового поля, создание элементов управления;
5. Проведение ручного и автоматического тестирования для выявления и исправление ошибок.

## ХОД РАБОТЫ

Реализация задачи

Данные задачи были реализованы на языке C++ с использованием библиотеки raylib. До начала разработки была проведена аналитическая работа с поиском и изучением необходимой информации, уделив особое внимание правилам игры «Game of life» Джона Конвея, особенностям реализации графического интерфейса и методам обработки событий.

Первым этапом работы стало создание графического интерфейса. Настроено окно приложения и разработаны функции для отображения игрового поля. Визуализация состояния клеток выполнена с использованием библиотеки raylib: «живые» клетки отображаются одним цветом, а «мертвые» – другим, что позволяет легко различать их на игровом поле.

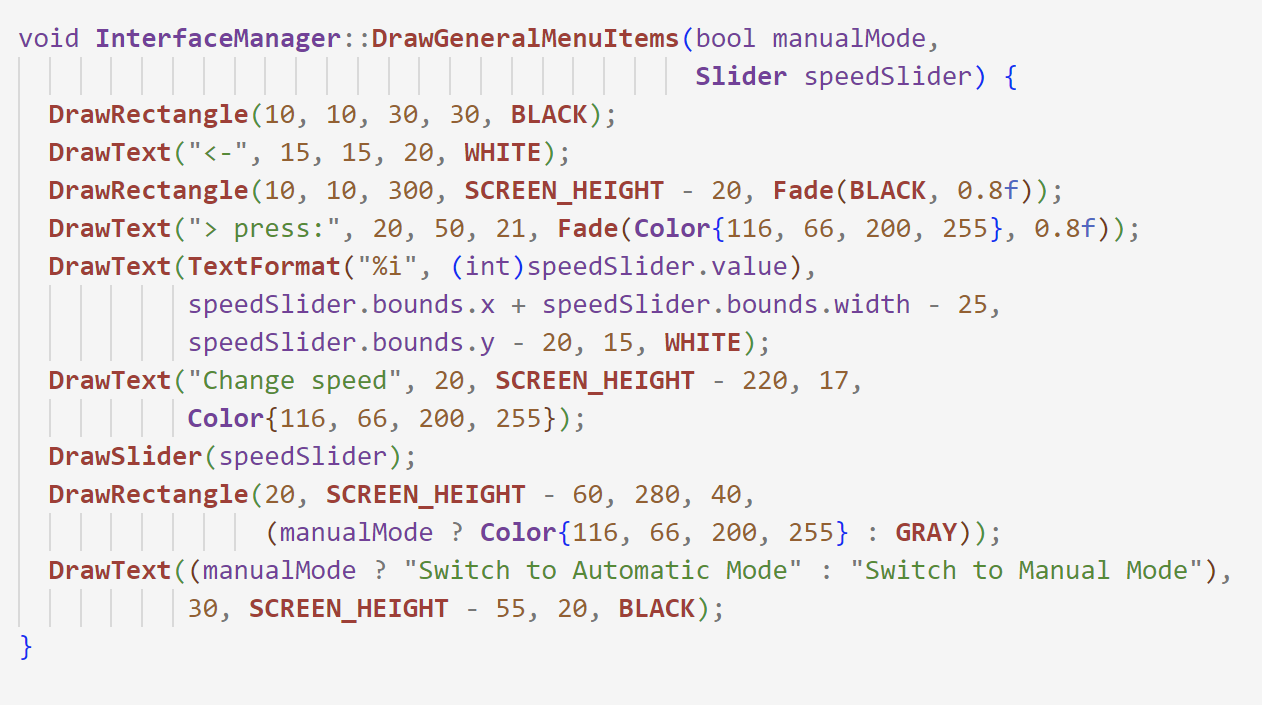


Рисунок 1-часть кода, реализующего создание интерфейса приложения

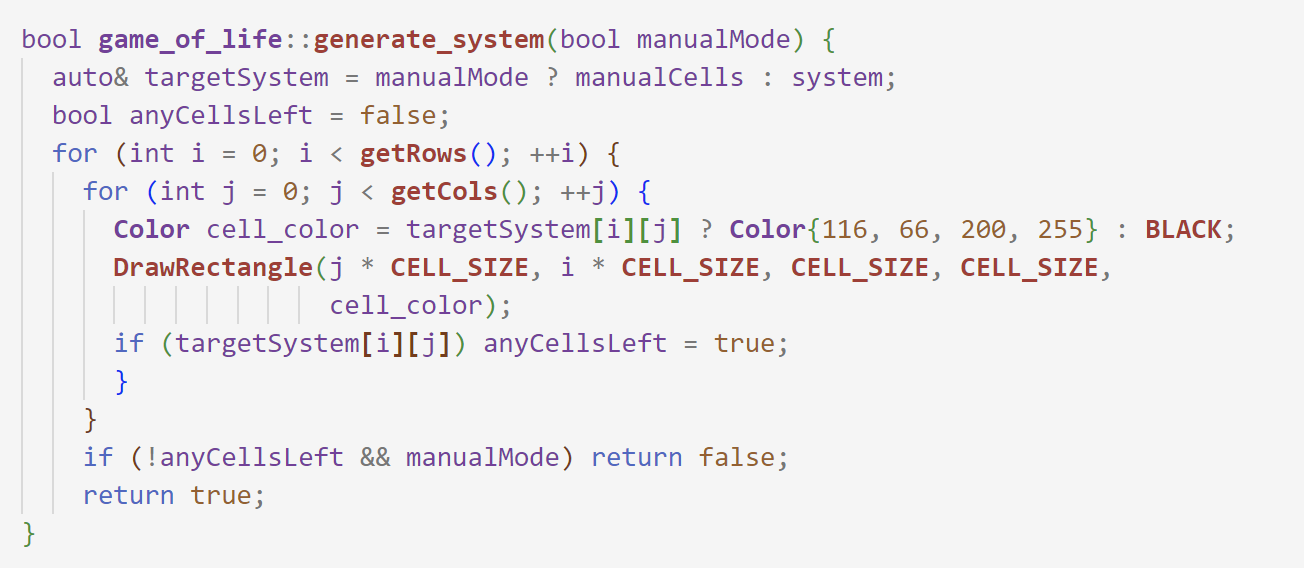


Рисунок 2-метод, реализующий отображение клеток на экране

Далее была добавлена функциональность для случайного размещения клеток. Разработана функция, которая автоматически заполняет игровое поле живыми клетками в случайных позициях при запуске игры. Реализованы обработчики событий для добавления новых клеток на игровое поле кликами мыши. Дополнительно создана функциональность для генерации случайного количества живых клеток на поле при нажатии определенных клавиш, что добавляет вариативности в игровой процесс.



Рисунок 3-метод, реализующий случайное размещение клеток на экране

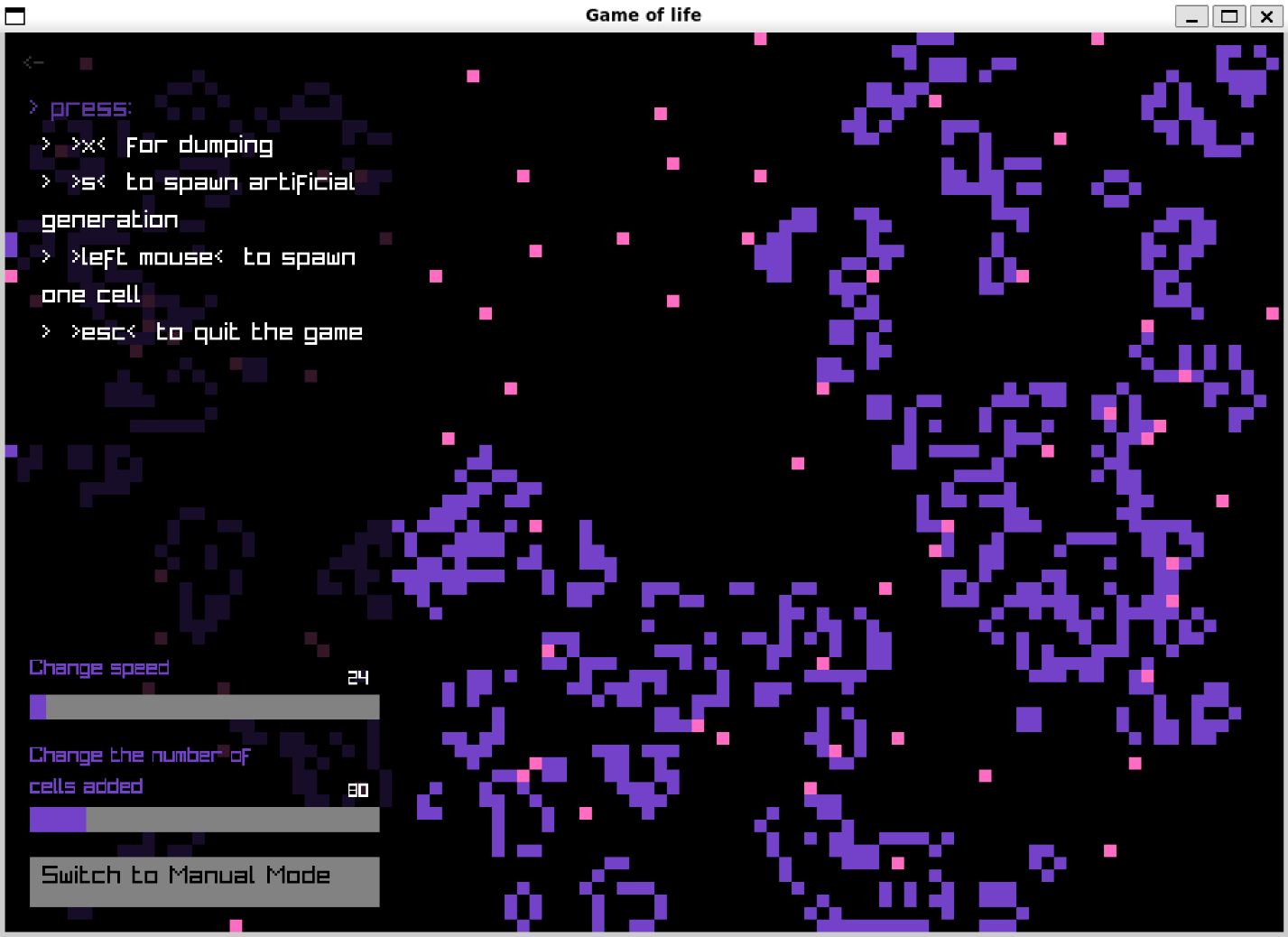


Рисунок 4-реализация случайного размещения клеток на экране в приложении

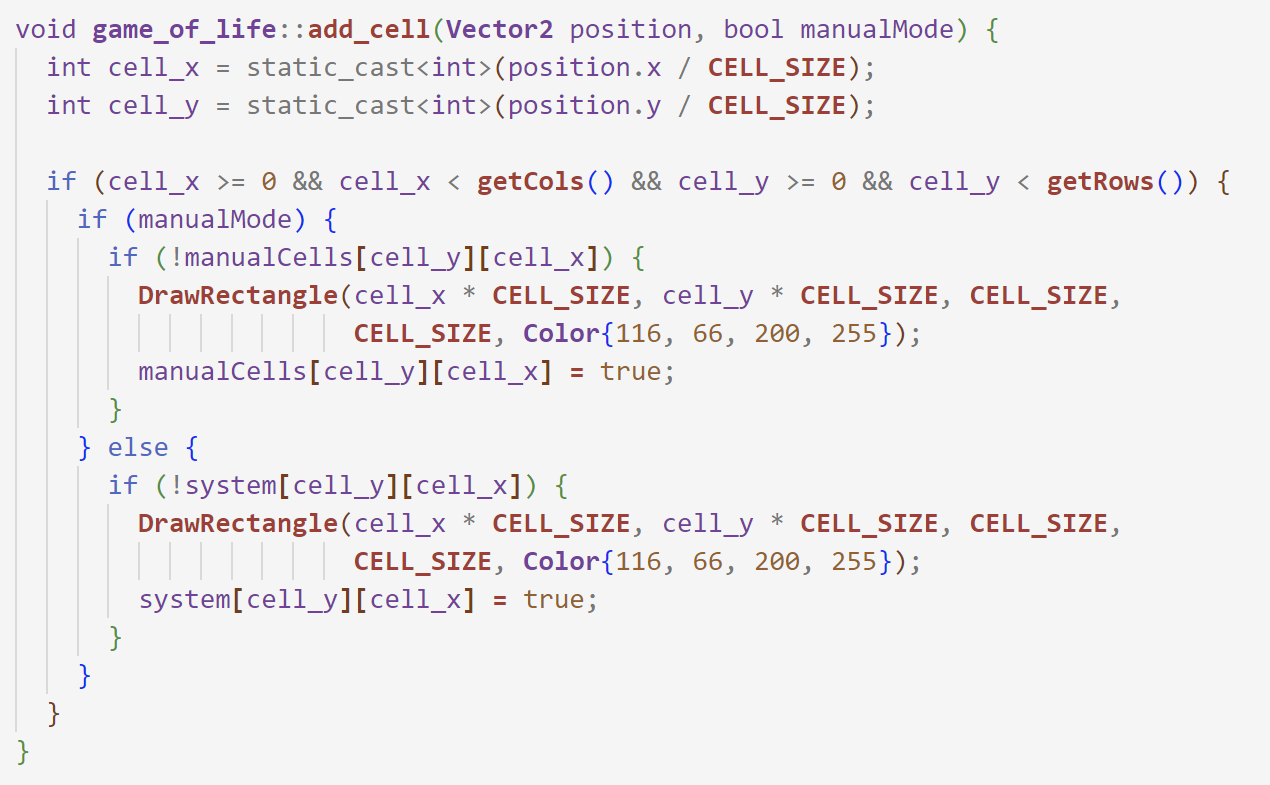


Рисунок 5-метод, обрабатывающий событие нажатия пользователем на экран для добавления новой клетки

Следующим шагом стала реализация режима ручного размещения клеток. Пользователю предоставлена возможность самостоятельно размещать клетки на игровом поле путем кликов мыши. Созданы соответствующие обработчики событий для управления запуском и остановкой процесса эволюции клеток с помощью клавиш, что позволяет пользователям полностью контролировать размещение клеток и управление их эволюцией.



Рисунок 6-часть обработчиков событий приложения

Для управления игрой в целом разработано несколько ключевых функций. Созданы функции для запуска и остановки игры, позволяющие контролировать процесс эволюции клеток. Реализована возможность очистки игрового поля от всех клеток, что дает пользователям возможность начинать игру с чистого листа в любой момент. Добавлена функция для регулировки скорости эволюции клеток, предоставляющая пользователям возможность изменять темп игры в соответствии с их предпочтениями.

Кроме того, реализована функциональность для скрытия и отображения дополнительного меню управления. Это меню позволяет пользователям переключаться между различными режимами игры и управлять другими аспектами приложения. Обеспечена возможность переключения между режимами случайного и ручного размещения клеток, что предоставляет пользователям большую гибкость в выборе стиля игры.

Важным аспектом разработки стала реализация самого алгоритма игры в объектно-ориентированном стиле. Были созданы классы для представления клеток и игрового поля. Основной класс отвечал за управление игровым процессом, включающим создание начальной конфигурации, обновление состояния клеток в соответствии с правилами игры и отображение текущего состояния на экране. Объектно-ориентированный подход позволил организовать код более структурированно и модульно, обеспечив легкость в поддержке и расширении функциональности.



Рисунок 7-точка входа в программу

В результате проделанной работы было создано приложение, которое позволяет пользователю наблюдать за развитием клеточного автомата «Game of life»». Приложение поддерживает два режима размещения клеток: случайный и ручной. Пользователь может управлять запуском и остановкой игры, очищать игровое поле, регулировать скорость и переключаться между режимами игры.

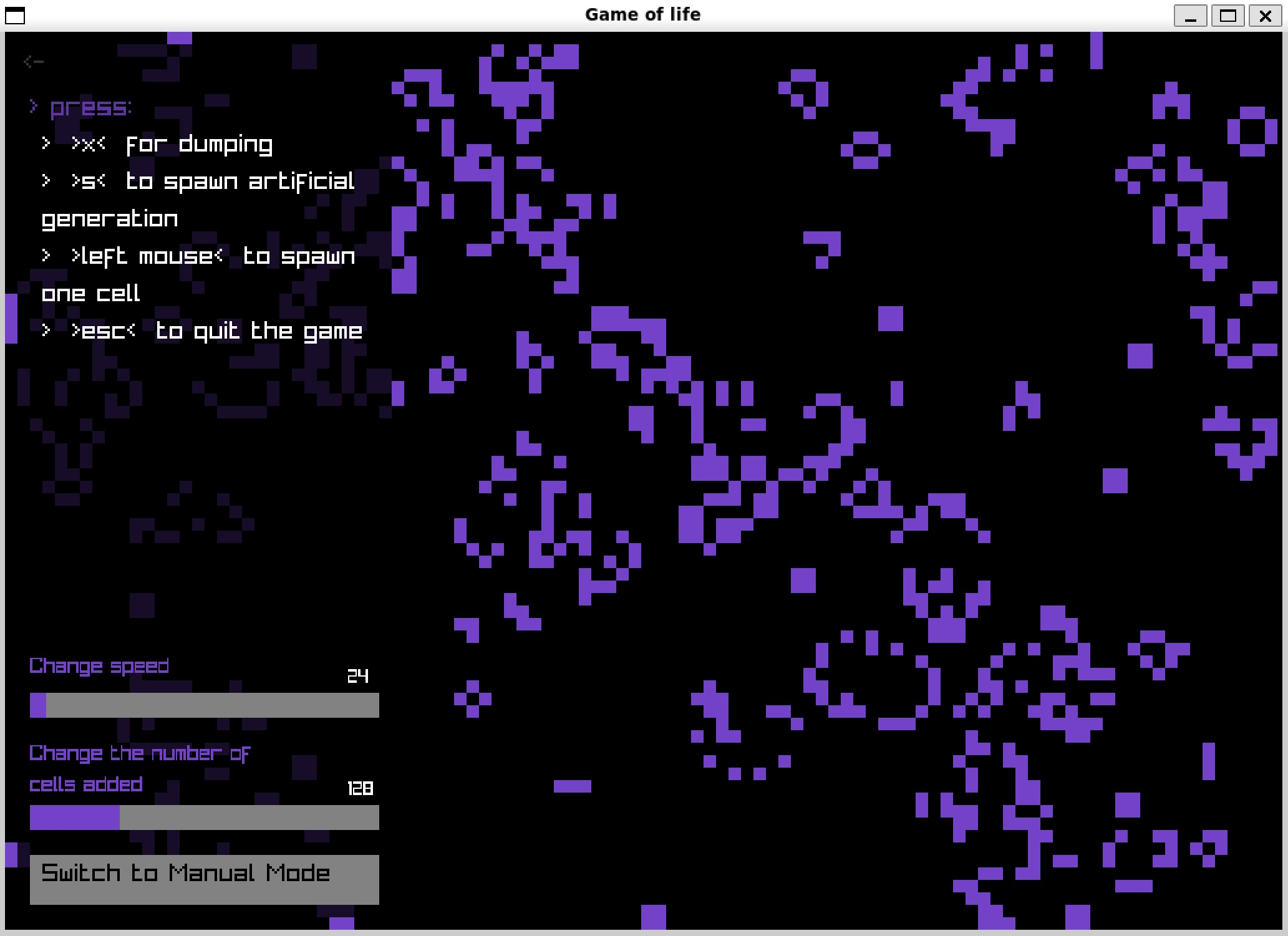


Рисунок 8-вид программы, режим случайного размещения клеток



Рисунок 9-вид программы, режим ручного размещения клеток

Описание алгоритма

Разумеется, реализация алгоритма Джона Конвея была одной из ключевых задач проекта. Для начала необходимо было разработать алгоритмическую модель, которая бы максимально точно отражала правила игры. Основные принципы алгоритма состояли в том, чтобы каждая клетка в игровом поле могла находиться в одном из двух состояний: «живая» или «мертвая», а обновление состояния каждой клетки зависело от числа ее живых соседей.

Важной частью алгоритма игры является метод для подсчета живых соседей с заданными координатами (x, y) на игровом поле (рис.10).

Для каждого соседа проверяется, находится ли он внутри границ игрового поля. Для этого вычисляются координаты соседа (another\_x, another\_y) путем прибавления i и j к текущим координатам (x, y).

Если сосед находится внутри границ поля, то проверяется его состояние: живая клетка или нет. Если сосед живой, то значение neighbors\_counter увеличивается на 1. После перебора всех соседей метод возвращает общее количество живых соседей для заданной клетки.

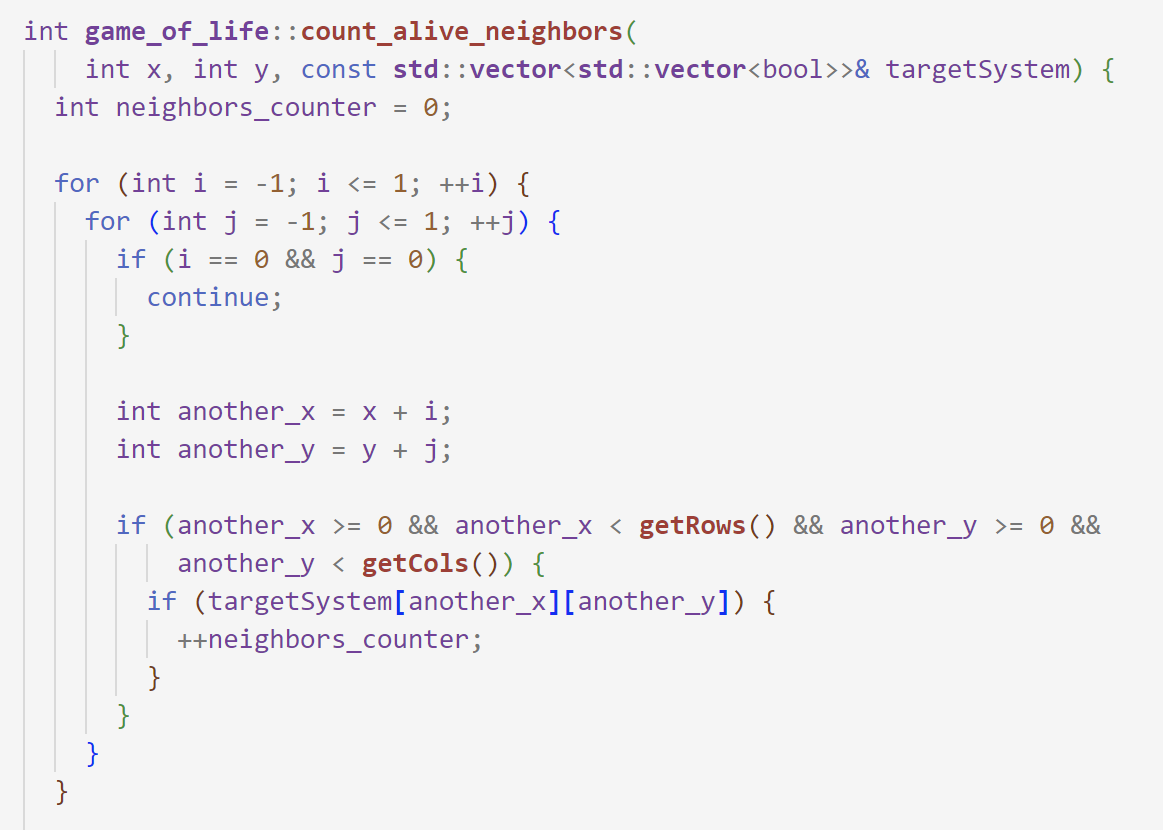


Рисунок 10-метод для подсчета живых соседей

Также был написан метод для генерации случайной начальной конфигурации игрового поля в режиме автоматической генерации (рис.11). При вызове этого метода происходит создание указанного количества новых «живых» клеток на случайных позициях на игровом поле.

Для начала проверяется параметр manualMode, который определяет, находится ли игра в ручном режиме. Если параметр установлен в true, что означает ручной режим, метод завершает свою работу, поскольку в этом режиме генерация происходит вручную пользователем.

В случае, если игра находится в режиме автоматической генерации (параметр manualMode равен false), создается генератор случайных чисел. Это необходимо для получения случайных координат клеток на игровом поле.

Затем происходит цикл, в котором создается указанное количество новых клеток. Для каждой новой клетки генерируются случайные координаты random\_x и random\_y с использованием равномерного распределения в заданных пределах. Эти координаты указывают на позицию новой клетки на игровом поле.

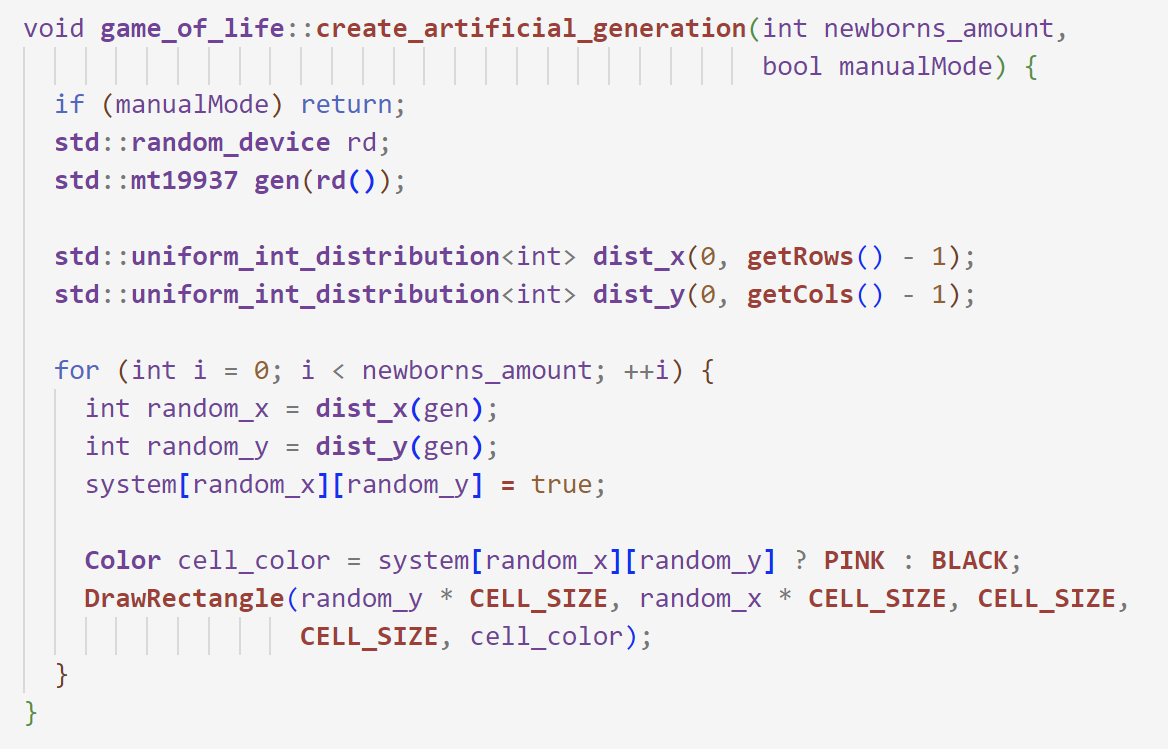


Рисунок 11-метод генерации случайной начальной конфигурации игрового поля

Для реализации бесконечного взаимодействия клеток (за исключением случая, когда пользователь ставит игру на паузу) все время, пока приложение запущено, выполняется метод для обновления состояния игровой системы (рис.13).

Сначала определяется, какую систему следует обновлять: основную (system) или ручную (manualCells), в зависимости от выбранного режима (manualMode).

Затем создается временная система new\_system, которая будет хранить обновленное состояние клеток. Вектор newCells очищается, чтобы быть готовым для новых клеток в следующем поколении.

Далее происходит итерация по каждой клетке в игровой системе. Для каждой клетки вычисляется количество живых соседей с помощью метода count\_alive\_neighbors.

На основе количества живых соседей и текущего состояния клетки определяется ее новое состояние. Если клетка живая и имеет 2 или 3 живых соседа, она остается живой в новом поколении. Если клетка мертвая и имеет ровно 3 живых соседа, она становится живой в новом поколении.

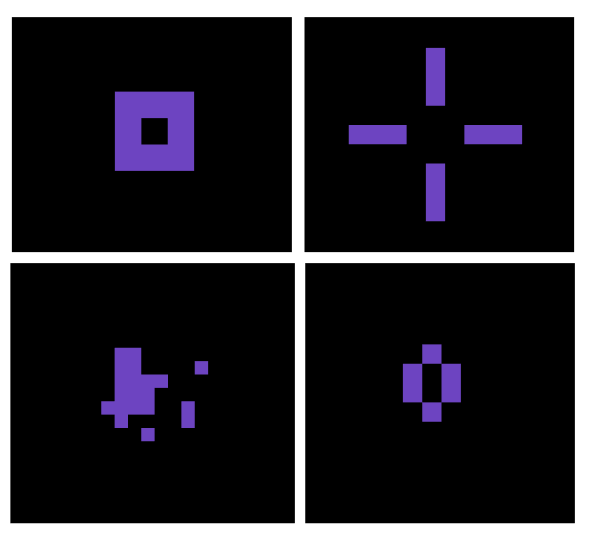


Рисунок 12-реализация взаимодействия клеток между собой (слева – начальное состояние, справа – промежуточный этап взаимодействия)

После обновления состояний всех клеток новая система присваивается целевой системе (targetSystem), тем самым завершая процесс обновления.

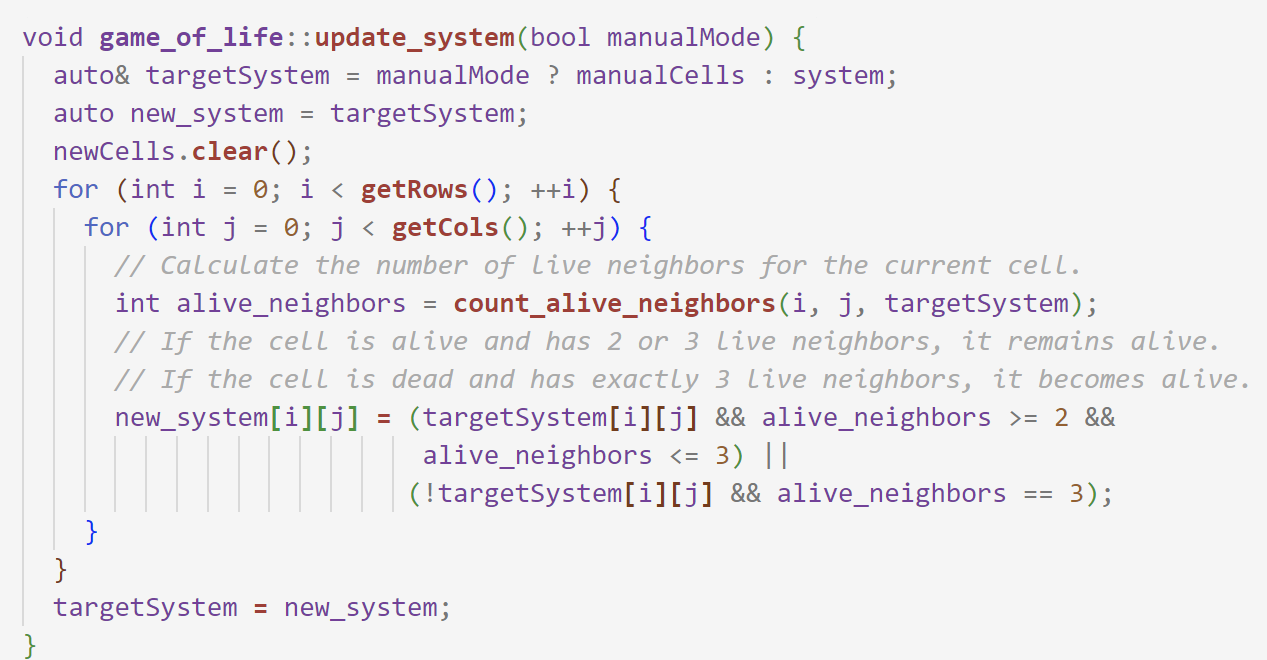


Рисунок 13-метод для обновления состояния игровой системы

Тестирование программного кода

После реализации графического интерфейса и алгоритма фокус работы сместился на написание Google тестов для проверки правильности функционирования программного кода.

Процесс тестирования начался с создания набора тестов. Эти тесты включали в себя проверку правильности обновления состояния системы и подсчета количества живых соседей. Были созданы различные сценарии для проверки граничных условий и разнообразных начальных конфигураций клеток.

Далее было проведено тестирование графического интерфейса, включая проверку корректного отображения игрового поля и состояния клеток. Эти тесты также включали проверку взаимодействия пользователя с интерфейсом, таких как добавление новых клеток и изменение режима работы. Проверялась корректность обновления состояния системы в соответствии с действиями пользователя в графическом интерфейсе.

Наконец, было проведено тестирование производительности для оценки скорости работы алгоритма в зависимости от размера игрового поля и количества живых клеток.



Рисунок 14-часть тестов для тестирования программного кода

# ЗАКЛЮЧЕНИЕ

В период реализации поставленных задач были усвоены новые, а также закреплены уже пройденные ранее теоретические и практические знания в разработке программ на языке C++, изучены и применены принципы тестирования кода с использованием Google Test.

Для разработки приложений изучалась дополнительная литература с целью поиска необходимой информации. Весь код выгружался на платформу GitHub, что дало больше понимания о работе платформы.

В результате прохождения ознакомительной практики мной были освоены навыки применения теоретических знания на практике, работы различными инструментами и технологиями для создания приложения.

В течение практики я узнала об основных этапах создания приложений: от предварительной идеи до запуска завершенного продукта.

# СПИСОК ЛИТЕРАТУРЫ

## Белоусов, Л. «Игра «Жизнь» Джона Конвея.» Программист-практик, выпуск 2, 2019 год

## Левин, М. Б. "Математические модели развития живых систем.» Журнал «Наука и жизнь», выпуск 5, 1982 год

## Официальная документация Raylib. URL: https://www.raylib.com\

## «Игры на C++ с использованием библиотеки Raylib», Tproger. URL: https://tproger.ru/translations/games-in-c-with-raylib/