

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Конструирование программного обеспечения (КПО)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

Игра «Тетрис»

БГУИР КП 1-40 01 01 003 ПЗ

Студент: гр. 151003 Барановский Р.А.

Руководитель: асс. Шостак Е.В.

Минск 2022

Учреждение образования
«Белорусский государственный университет информатики и
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ПОИТ

(подпись)
Лапицкая Н.В. 2022г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Барановскому Роману Алексеевичу

1. Тема работы Игра «Тетрис»

2. Срок сдачи законченной работы 19.12.2022г.

3. Исходные данные к работе Среда программирования Visual Studio. Наличие графической реализации интерфейса игры. Возможность управлять фигурами игры при помощи клавиш. Наличие в игре статистики по упавшим фигурам, очкам, времени. Наличие игрового меню с настройками, содержащими возможность включения/отключения различных опций игры, таких как фон игрового поля, отображение сетки поля, отображение тени фигуры, а также редактирование размеров игрового поля. Наличие звукового сопровождения в игре и меню. Возможность отключения звукового сопровождения. Поддержка двух языков(русский и английский).

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение

1 Анализ литературных источников

2 Постановка задачи

3 Разработка программного средства

4 Тестирование и проверка работоспособности программного средства

5 Руководство по установке и использованию программного средства

Заключение

Список использованной литературы

Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

Схема алгоритма в формате А1

6. Консультант по курсовой работе Шостак Е.В.

7. Дата выдачи задания 16.09.2022г.

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

Раздел 1. Введение к 15.09.2022г. – 10 % готовности работы;

Раздел 2 к 15.10.2022г. – 30% готовности работы

Раздел 3 к 15.11.2022г. – 60% готовности работы

Раздел 4, 5. Заключение к 15.12.2022 – 90 % готовности работы;

Оформление пояснительной записки и графического материала к 17.12.2022 – 100 % готовности работы.

Защита курсового проекта с 17.12.2022г. по 20.12.2022г.

РУКОВОДИТЕЛЬ _____ Шостак Е.В.
(подпись)

Задание принял к исполнению _____ Барановский Р.А. 16.09.2022г.
(дата и подпись студента)

СОДЕРЖАНИЕ

| | | |
|-------|---|----|
| 1 | Анализ литературных источников | 7 |
| 1.1 | Анализ существующих аналогов..... | 7 |
| 1.1.1 | «Тетрис» на ресурсе www.min2win.ru | 7 |
| 1.1.2 | «Тетрис» от NADO games | 8 |
| 1.1.3 | «Тетрис» от N3TWORK Inc. | 9 |
| 1.2 | Анализ методов и способов разработки | 10 |
| 1.2.1 | Используемые библиотеки и технологии..... | 10 |
| 1.2.2 | Используемые структуры данных..... | 10 |
| 2 | Постановка задачи | 12 |
| 2.1 | Назначение разработки..... | 12 |
| 2.2 | Перечень функциональных требований | 13 |
| 2.3 | Структура программы..... | 14 |
| 2.4 | Входные и выходные параметры | 15 |
| 2.5 | Состав и параметры технических и программных средств | 16 |
| 3 | Разработка программного средства..... | 17 |
| 3.1 | Описание алгоритмов решения задачи | 17 |
| 3.2 | Структура типов | 19 |
| 3.3 | Схема алгоритмов решения задач по ГОСТ 19.701-90 | 23 |
| 3.3.1 | Схема алгоритма <code>Figure::move(const int step, const Direction& direction)</code> | 23 |
| 3.3.2 | Схема алгоритма <code>Figure::rotateFigure(const bool isRotate)</code> | 25 |
| 3.3.3 | Схема алгоритма <code>Figure::fall()</code> | 28 |
| 3.3.4 | Схема алгоритма <code>Field::destroyLines()</code> | 31 |
| 3.3.5 | Схема алгоритма <code>Field::isFullfilled()</code> | 35 |
| 4 | тестирование и проверка работоспособности программного средства | 37 |
| 4.1 | Запуск меню и игры | 37 |
| 4.1.1 | Тест 1 | 37 |
| 4.1.2 | Тест 2..... | 37 |
| 4.1.3 | Тест 3..... | 38 |
| 4.2 | Применение настроек | 39 |
| 4.2.1 | Тест 4 | 39 |
| 4.2.2 | Тест 5 | 40 |
| 4.2.3 | Тест 6 | 40 |
| 4.2.4 | Тест 7 | 41 |
| 4.2.5 | Тест 8 | 42 |
| 4.2.6 | Тест 9 | 42 |
| 4.3 | Проверка работоспособности игры..... | 43 |
| 4.3.1 | Тест 10 | 43 |
| 4.3.2 | Тест 11 | 43 |
| 4.3.3 | Тест 12 | 44 |
| 4.3.4 | Тест 13 | 44 |
| 4.3.5 | Тест 14 | 45 |

| | |
|---|----|
| 4.3.6 Тест 15 | 45 |
| 4.3.7 Тест 16 | 46 |
| 4.3.8 Тест 17 | 46 |
| 5 Руководство по установке и использованию программного средства .. | 48 |
| 5.1 Установка | 48 |
| 5.2 Использование | 51 |
| 5.2.1 Работа с главным меню | 51 |
| 5.2.2 Работа с меню настроек..... | 52 |
| 5.2.3 Игра 53 | |
| Приложение А | 56 |
| Приложение Б | 64 |
| Приложение В..... | 69 |

ВВЕДЕНИЕ

Данный курсовой проект посвящен созданию игры «Тетрис». Данная игра была придумана и разработана советским программистом и геймдизайнером Алексеем Пожитновым. «Тетрис» представляет собой головоломку, построенную на использовании геометрических фигур «тетрамино» - разновидности полимино, состоящих из четырех квадратов. Идею «Тетриса» Пожитнову подсказала игра в пентамино. Первоначальная версия игры была написана Пожитновым на языке Паскаль для компьютера «Электроника 60». Первая коммерческая версия игры была выпущена американской компанией Spectrum HoloByte в 1987 году. В последующие годы «Тетрис» во множестве различных версий был портирован на великое множество устройств, включая всевозможные компьютеры и игровые консоли, а также такие устройства, как графические калькуляторы, мобильные телефоны и многие другие. Данная игра является невероятно популярной и по сегодняшний день. По количеству проданных коммерческих версий «Тетрис» превосходит любую другую компьютерную игру в истории – лишь для одной GameBoy было продано 35 миллионов копий. В 2007 году «Тетрис» вошел в число десяти важнейших компьютерных игр, принятых на сохранение в Библиотеку Конгресса. На сегодняшний день есть великое множество различных версий и реализаций тетриса на самых разных платформах, начиная от стандартных компьютеров, заканчивая осциллографом. В данной работе была поставлена цель сделать игру с приятной графикой и интерфейсом, сделав игру максимально настраиваемой.

1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

1.1 Анализ существующих аналогов

Современные разновидности «Тетриса» отличаются огромным разнообразием. Они кардинально отличаются как графически, так и реализацией. Однако стандартные правила тетриса остаются везде практически без изменений.

1.1.1 «Тетрис» на ресурсе www.min2win.ru

На данном сайте представлена самая простая версия данной игры без каких-либо необычных особенностей. Представлено стандартное управление с возможностью ставить игру на паузу.

Достоинства игры:

- простота;
- хороший подсчет очков.

В свою очередь к недостаткам можно отнести:

- отсутствие интересных особенностей;
- медленная работа игры.

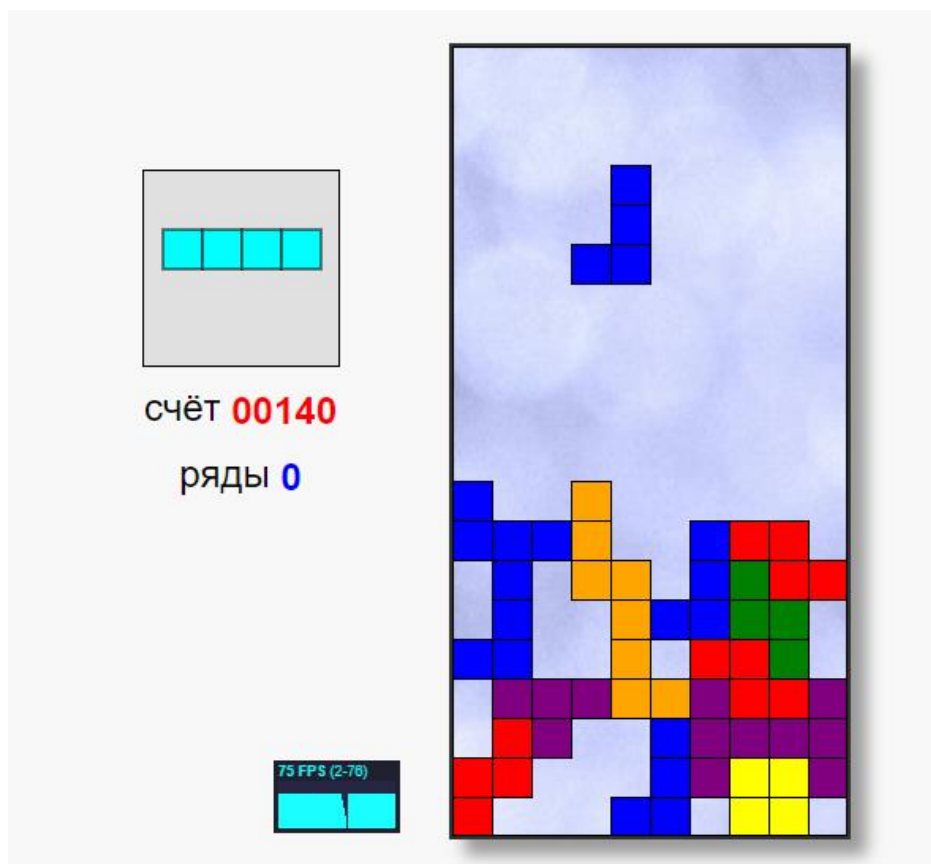


Рисунок 1.1 – «Тетрис на сайте www.min2win.ru»

1.1.2 «Тетрис» от NADO games

Данная версия «Тетриса» выпущена компанией NADO games и представляет собой приложение под платформы Android и IOS. В игре ведется подробная статистика и проводятся соревнования между игроками.

Достоинства игры:

- наличие подробной статистики;
- соревнования между игроками;
- красочный интерфейс;
- отличное музыкальное сопровождение.

Недостатков выявлено не было.

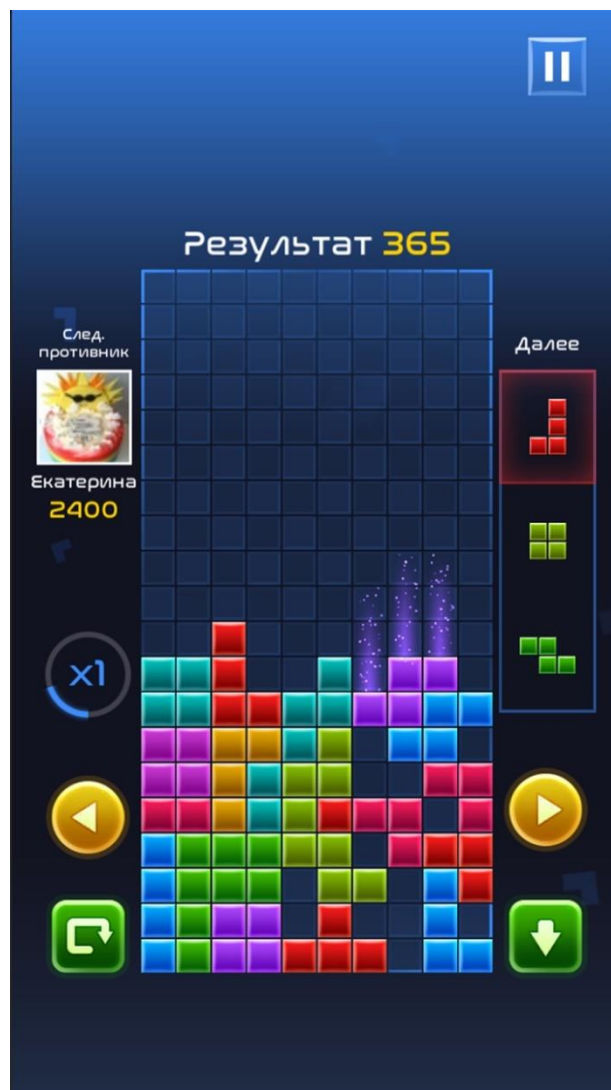


Рисунок 1.2 – «Тетрис от NADO games»

1.1.3 «Тетрис» от N3TWORK Inc.

Данная версия «Тетриса» выпущена компанией NADO games и представляет собой приложение под платформы Android и IOS. В игре присутствуют различные эффекты и бонусы. Имеется возможность выбирать различные стили игрового поля, в наличии огромное количество разнообразных красивых анимаций.

Достоинства игры:

- большое количество анимаций;
- красивое оформление уровней;
- возможность выбора стиля поля.

Недостатков выявлено не было.

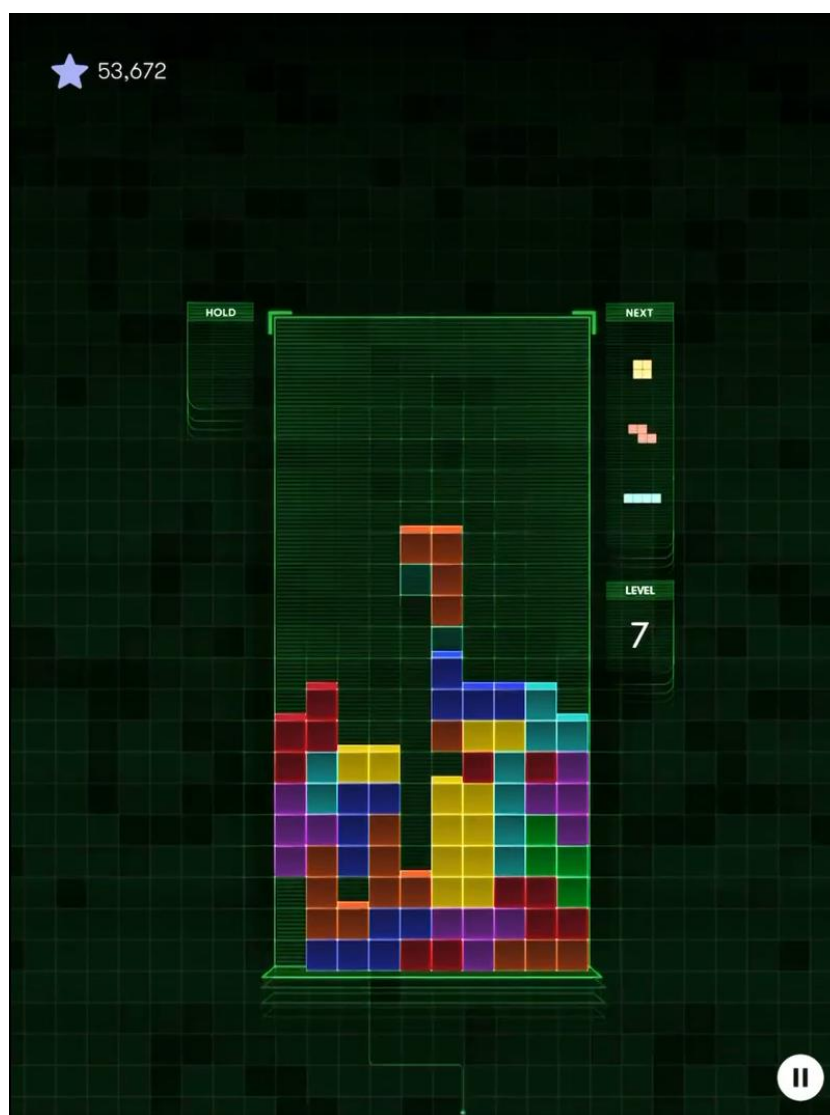


Рисунок 1.3 – «Тетрис от N3TWORK Inc.»

1.2 Анализ методов и способов разработки

1.2.1 Используемые библиотеки и технологии

Предполагается, что разрабатываемая игра будет обладать двумерной графикой с возможностью передвижений фигур, фигуры должны уметь падать, поворачиваться, ускоряться, двигаться горизонтально. В то же время необходимо обеспечить взаимодействие между фигурами и полем. Для этих целей будет использована сторонняя библиотека «SFML».

«SFML» (англ. Simple and Fast Multimedia Library) – свободная кроссплатформенная мультимедийная библиотека. Написана на C++, но доступна также для C, C#, .Net, D, Java, Python, Ruby, OCaml, Go и Rust. Представляет собой объектно-ориентированный аналог SDL.

«SFML» содержит ряд модулей для простого программирования игр и мультимедиа приложений. Предполагается использование следующих модулей:

- Graphics – делает простым отображение графических примитивов и изображений;
- Audio – предоставляет интерфейс для управления звуком.

В то же время ставится задача создания квадратной двумерной графики с высоким разрешением. В то же время будет использован тайловый, или плиточный, метод создания поля и изображения различных объектов. Тайлы – небольшие изображения одинаковых размеров, служащие фрагментами большой картины. Количество тайлов на один «мир» может достигать нескольких сотен. Матрица клеток при этом хранит только номера тайлов, за счет чего достигается экономия памяти при построении огромных двумерных пространств.

1.2.2 Используемые структуры данных

В данном проекте будут использоваться две основные структуры данных:

- массив (array);
- вектор (vector).

В данной работе структура массив (array) предназначена в основном для хранения информации о составе фигур (тетрамино) и хранения их координат в отображенном на поле формате. Структура массива представлена на рисунке 1.4.



Рисунок 1.4 – Массив

В свою очередь структура вектор предназначена в основном для хранения информации о поле. Так как в данной реализации игры «Тетрис» предусмотрена возможность изменения размера игрового поля, появилась необходимость использования структуры данных, которая сможет динамически изменять свой размер во время выполнения программы. Также класс, реализующий данную структуру данных, обладает набором удобных в использовании методов, упрощающих логику программы.

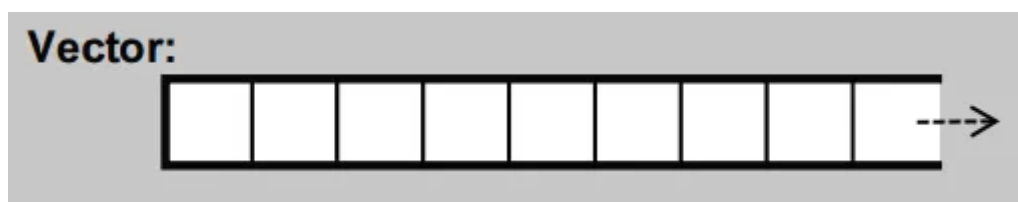


Рисунок 1.5 – Вектор

2 ПОСТАНОВКА ЗАДАЧИ

2.1 Назначение разработки

Назначением проектирования является разработка игры «Тетрис». На основании произведенного обзора существующих аналогов, выявленных преимуществ и недостатков данных игр, сделан вывод, что для решения поставленной цели необходимо выполнить следующие задачи:

- проектирование архитектуры игры;
- проектирование графического сопровождения;
- разработка алгоритмов передвижения объектов;
- разработка алгоритмов взаимодействия объектов между собой;
- разработка алгоритма загрузки тайлов тетрамино из внешнего файла;
- разработка алгоритмов отрисовки передвижения объектов;
- создание меню для настроек игры
- тестирование приложения.

2.2 Перечень функциональных требований

Целью разработки игры «Тетрис» является объединение основных достоинств рассмотренных существующих аналогов, а также компенсация недостатков этих игр. В результате разработки необходимо предоставить реализацию следующих функций:

- загрузка тайлов тетрамино из внешнего файла;
- управление тетрамино в декартовой системе координат;
- взаимодействие между падающим тетрамино и полем;
- отображение количества очков;
- отображение количества времени текущей игры;
- отображение статистики упавших фигур.
- отображение всего текста на выбранном пользователем языке.

2.3 Структура программы

При разработке приложения было использовано 9 модулей:

- main.cpp – главный модуль, содержащий отображаемое окно игры;
- Field.cpp (Field.h) – модуль, обеспечивающий логику работы и отрисовку игрового поля с уже упавшими тетрамино;
- Figure.cpp (Figure.h) – модуль, обеспечивающий логику работы и отрисовку падающего тетрамино, которым управляет игрок;
- Language.cpp (Language.h) – модуль, обеспечивающий поддержку двух языков в игре(русский и английский);
- Menus.cpp (Menus.h)– модуль, обеспечивающий поддержку всех меню приложения;
- Music_Sounds.cpp (Music_Sounds.h) – модуль, обеспечивающий звуковое сопровождение для меню и игры;
- Random.cpp (Random.h) – модуль, обеспечивающий рандом в игре;
- Statistics.cpp (Statistics.h) – модуль, обеспечивающий статистику в игре(время, очки, количество упавших тетрамино);
- TextureSpriteWork.cpp (TextureSpriteWork.h) – модуль, обеспечивающий более удобную работу с спрайтами и текстурами;

2.4 Входные и выходные параметры

Входными данными для приложения являются следующие данные:

- нажатие клавиши управления тетрамино;
- настройки игрового поля, заданные пользователем;
- настройки тетрамино, заданные пользователем;

Выходными данными для приложения будут выступать следующие данные:

- изменение координат тетрамино;
- изменение статистики по ходу игры;
- удаление тетрамино при уничтожении ряда;
- отрисовка измененного поля и тетрамино на экране игрока.

2.5 Состав и параметры технических и программных средств

Игра «Тетрис» должна функционировать на персональных компьютерах со следующими характеристиками:

- Операционная система Windows 10;
- RAM: 2 GB;
- Пространство на диске: 1 GB;
- Процессор: минимальное требование - Pentium 2 266 МГц;
- Браузеры: Internet Explorer 9 и выше, Firefox;
- Монитор;
- Мышь;
- Клавиатура.

В данном разделе указаны минимальные технические требования для запуска игры. Для эксплуатации в реальных условиях могут потребоваться более мощные технические средства. Разработанная игра должна корректно функционировать на более мощном оборудовании.

3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Описание алгоритмов решения задачи

Таблица 1 – Описание алгоритмов решения задачи

| № п.п. | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагаемый тип реализации |
|--------|--|---|--|--|
| 1 | main | Служит отправной точкой выполнения программы. Создает окно игры. | Формальные параметры отсутствуют | Функция. Возвращаемый параметр – код завершения вызывавшему процессу |
| 2 | mainMenu(RenderWindow& window, Field& field, Figure& figure) | Служит для отображения главного меню. Параметры поля и фигуры передаются для изменения | window – окно приложения; field – поле игры; figure - тетрамино | Процедура |
| 3 | settings(RenderWindow& window, Field& field, Figure& figure) | Служит для отображения меню настроек. Параметры поля и фигуры передаются для изменения | window – окно приложения; field – поле игры; figure - тетрамино | Процедура |
| 4 | pauseMenu(RenderWindow& window, Field& field, Figure& figure) | Служит для отображения меню паузы. Параметры поля и фигуры передаются для изменения | window – окно приложения; field – поле игры; figure - тетрамино | Функция. Возвращаемый параметр – false, если остаться в игре, true, если выйти в главное меню |
| 5 | gameOverMenu(RenderWindow& window, Field& field, Figure& figure) | Служит для отображения меню конца игры. Параметры поля и фигуры передаются для изменения | window – окно приложения; field – поле игры; figure - тетрамино | Функция. Возвращаемый параметр – false, если играть заново, true, если выйти в главное меню |
| 6 | Figure::move(const int step, const | Служит для перемещения | step – размер смещения | Процедура |

Продолжение таблицы 1

| | | | | |
|----|---|---|--|--|
| | Direction& direction) | фигуры, первый параметр задает размер смещения, второй направление смещения | direction – направление смещения | |
| 7 | Figure::rotateFigure(const bool isRotate) | Служит для поворота тетрамино на поле, параметр – true, если необходимо выполнить поворот, false в противном случае | isRotate – true, если тетрамино необходимо повернуть, false в противном случае | Процедура |
| 8 | Figure::fall() | Используется для падения фигуры через определенные промежутки времени | Формальные параметры отсутствуют | Процедура |
| 9 | Field::destroyLines() | Используется для уничтожения заполненных линий в игре | Формальные параметры отсутствуют | Функция. Возвращаемый параметр – количество очков за уничтоженные ряды |
| 10 | Field::isFullfilled() | Проверяет, заполнено ли поле(что означает конец игры) | Формальные параметры отсутствуют | Функция. Возвращаемый параметр – true, если поле заполнено, false в противном случае |

3.2 Структура типов

Таблица 2 – структура типов программы

| Элементы данных | Рекомендуемый тип | Назначение |
|-----------------|---|--|
| Field | <pre> class Field { private: int m_height; int m_width; std::vector <std::vector<int>> m_field; Texture m_tilesImage; Sprite m_gridImage; Sprite m_backgroundImage; bool m_gridOn; bool m_backgroundOn; public: Field(const int height = 0, const int width = 0); void setSize(const int height, const int width); int getHeight(); int getWidth(); std::vector<int>& operator[](const int index); int destroyLines(); bool isFullfilled(); void drawField(RenderWindow& window); void setTilesImage(const Texture& tilesImage); void setGridImage(const Texture& gridImage); </pre> | <p>Класс игрового поля.</p> <p>Поля данных:</p> <p>m_height – высота поля; m_width – ширина поля; m_field – массив, представляющий поле; m_tilesImage – текстура, содержащая все возможные цвета тайлов, из которых складывается тетрамино; m_gridImage – спрайт решетки, отображающейся на поле по желанию пользователя; m_backgroundImage – спрайт картинки заднего фона, отображающейся по желанию пользователя; m_gridOn – переменная типа bool, отражающая, необходимо ли отображать решетку на поле; m_backgroundOn – переменная типа bool, отражающая, необходимо ли отображать картинку заднего фона на поле;</p> <p>Методы класса:</p> <p>Field(const int height, const int width) – конструктор класса, принимающий размеры поля; setSize(const int height, const int width) – меняет размеры поля; getHeight() – возвращает высоту поля; getWidth() – возвращает ширину поля;</p> |

Продолжение таблицы 2

| | | |
|--------|---|---|
| | <pre> void setBackgroundImage (const Texture& gridImage); void gridSwitch(); bool isGridOn(); void backgroundSwitch(); bool isBackgroundOn(); void reset(); }; </pre> | <p>operator[](const int index) – перегрузка оператора индексации для упрощенного доступа к ячейкам поля;</p> <p>destroyLines() – уничтожение заполненных линий поля и подсчет очков за это;</p> <p>isFullfilled() – проверить, заполнено ли поле (закончилась ли игра);</p> <p>drawField(RenderWindow& window) – нарисовать поле на экране;</p> <p>setTilesImage(const Texture& tilesImage) – задать изображение тайлам тетрамино;</p> <p>setGridImage(const Texture& gridImage) – задать изображение решетки поля;</p> <p>setBackgroundImage (const Texture& gridImage) – задать изображение фона поля;</p> <p>gridSwitch() – включить/отключить решетку;</p> <p>isGridOn() – проверить, включена ли решетка;</p> <p>backgroundSwitch() – включить/отключить фон поля;</p> <p>isBackgroundOn() – проверить, включен ли фон поля;</p> <p>reset() – очистить все поле;</p> |
| Figure | <pre> Figure { struct Point { int x, y; }; int m_figures[7][4]; </pre> | <p>Класс падающего тетрамино. Поля данных:</p> <p>Point – структура для хранения координат точки;</p> <p>m_figures – массив, содержащий все возможные тетрамино;</p> |

Продолжение таблицы 2

| | | |
|--|--|--|
| | <pre> Point m_tilesCoordinates[4]; Point m_shadowCoordinates[4]; int m_figureNum; int m_tileType; Texture m_tilesImage; Field& m_field; bool m_shadowOn; public: enum class Direction { VERTICAL, HORIZONTAL }; private: void makeCoordinates(); void move(const int step, const Direction& direction); void savePreviousCoordinates(Point* coordinates); void calculateShadowCoordinates(); public: Figure(Field& field, int figureNum = 0, int tileType = 0); void setImage(const Texture& tilesImage); void setTileType(const int tileType); bool isInField(); void drawFigure(RenderWindow& window); void moveFigure(const int dx); </pre> | <p> m_tilesCoordinates[4] – массив с координатами тайлов текущего тетрамино; m_shadowCoordinates[4] – массив с координатами тени текущего тетрамино; m_figureNum – номер тетрамино; m_tileType – тип тайла, использовавшегося для составления тетрамино; m_tilesImages - текстура, содержащая все возможные цвета тайлов, из которых складывается тетрамино; m_field – ссылка на поле для взаимодействия падающего тетрамино с ним; m_shadowOn – поле, отражающее, включена ли пользователем опция отображения тени тетрамино; Direction – перечислимый тип для хранения направления движения тетрамино; </p> <p> Методы класса: makeCoordinates() – метод, переводящий координаты тетрамино из массива в координаты на поле; move(const int step, const Direction& direction) – двигать тетрамино в направлении direction на step единиц; savePreviousCoordinates(Point * coordinates) – сохранить координаты тетрамино; calculateShadowCoordinates() – посчитать координаты тени тетрамино; </p> |
|--|--|--|

Продолжение таблицы 2

| | | |
|--|---|---|
| | <pre> void rotateFigure(const bool isRotate); void fall(); void shadowSwitch(); bool isShadowOn(); int getFigureType(); bool isJustAppeared(); void reset(); }; </pre> | <p>Figure(Field& field, int figureNum = 0, int tileType = 0) – конструктор класса, принимающий ссылку поля, в котором находится тетрамино, а также тип тетрамино и тип тайлов, из которого оно состоит;</p> <p>setImage(const Texture& tilesImage) – задать изображение всех тайлов, из которых может состоять тетрамино;</p> <p>setTileType(const int tileType) – задать тип тайла тетрамино;</p> <p>isInField() – проверить, не вышел ли тетрамино на границы поля;</p> <p>drawFigure(RenderWindow& window) – нарисовать тетрамино на экране;</p> <p>moveFigure(const int dx) – двигать тетрамино вдоль оси ox на dx единиц;</p> <p>rotateFigure(const bool isRotate) – повернуть тетрамино;</p> <p>fall() – метод, заставляющий тетрамино упасть;</p> <p>shadowSwitch() – включить/отключить тень тетрамино;</p> <p>isShadowOn() – проверить, включена ли тень тетрамино;</p> <p>getFigureType() – возвращает тип тетрамино;</p> <p>isJustAppeared() – возвращает только ли что тетрамино появилось на поле;</p> <p>reset() – создает новый тетрамино в вершине поля</p> |
|--|---|---|

3.3 Схема алгоритмов решения задач по ГОСТ 19.701-90

3.3.1 Схема алгоритма `Figure::move(const int step, const Direction& direction)`

Схема алгоритма передвижения тетрамино.

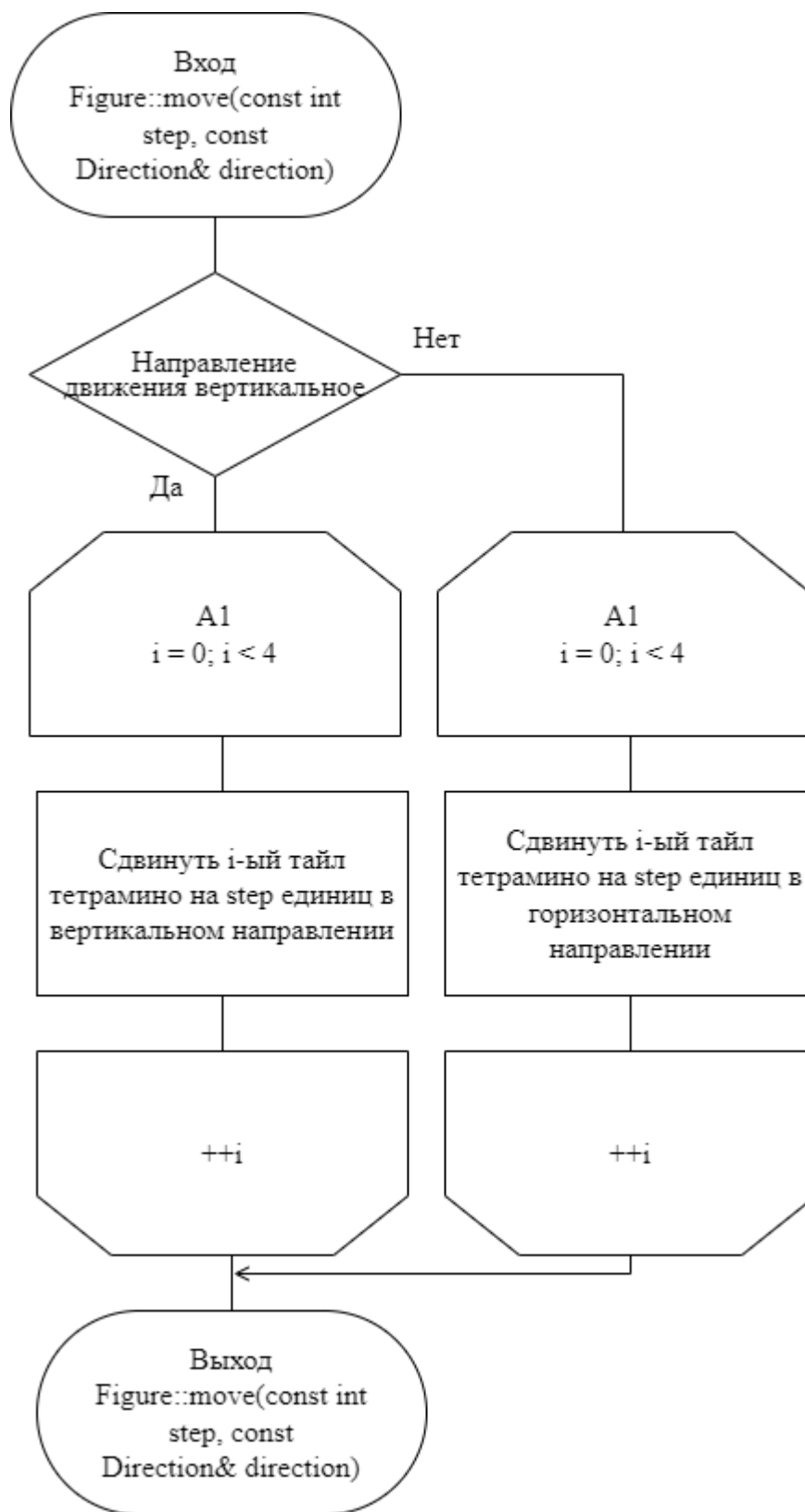


Рисунок 3.1 – Схема алгоритма Figure::move(const int step, const Direction& direction)

3.3.2 Схема алгоритма Figure::rotateFigure(const bool isRotate)

Схема алгоритма поворота тетрамино.

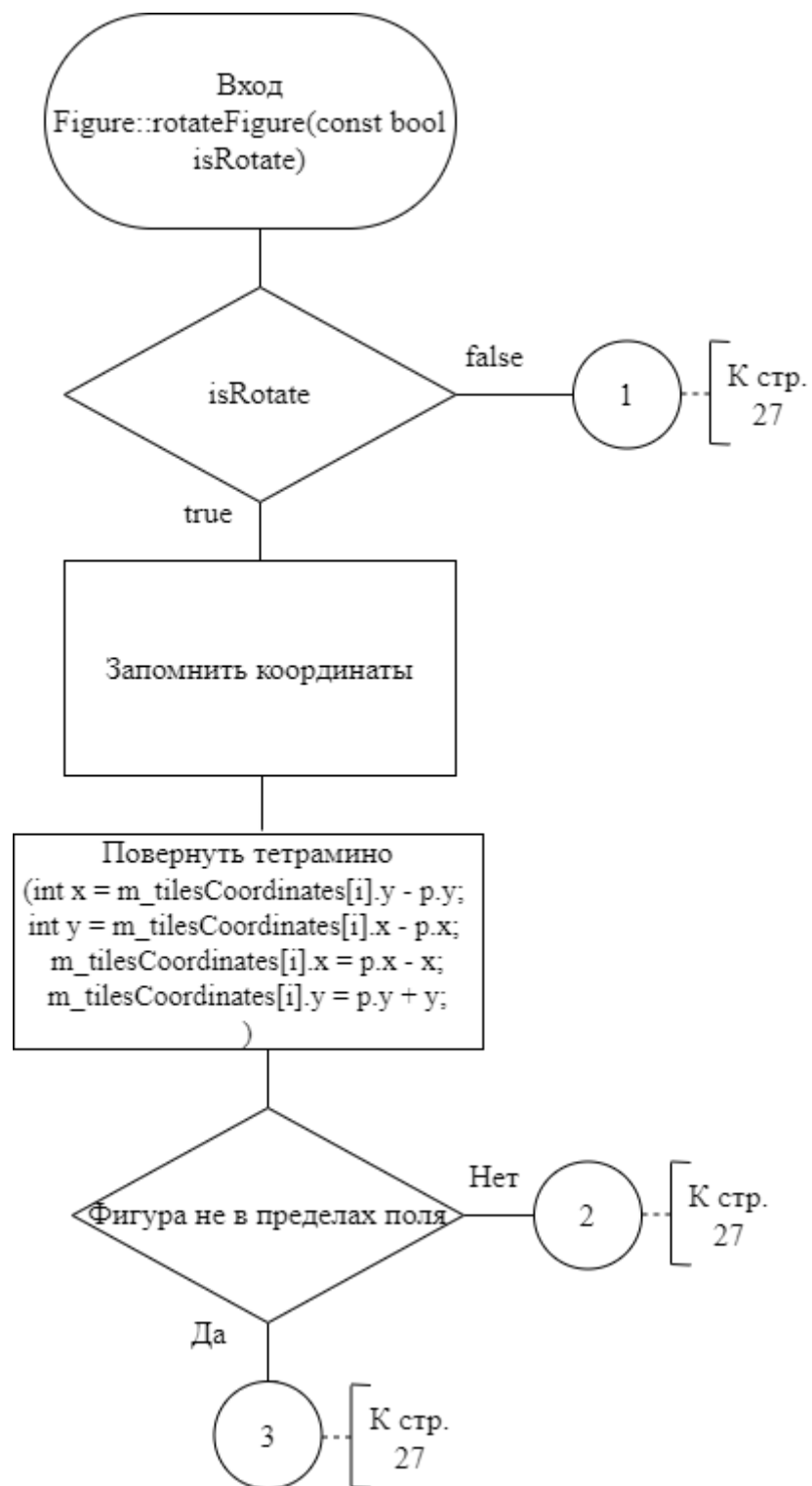


Рисунок 3.2 – Схема алгоритма Figure::rotateFigure(const bool isRotate)
(часть 1)

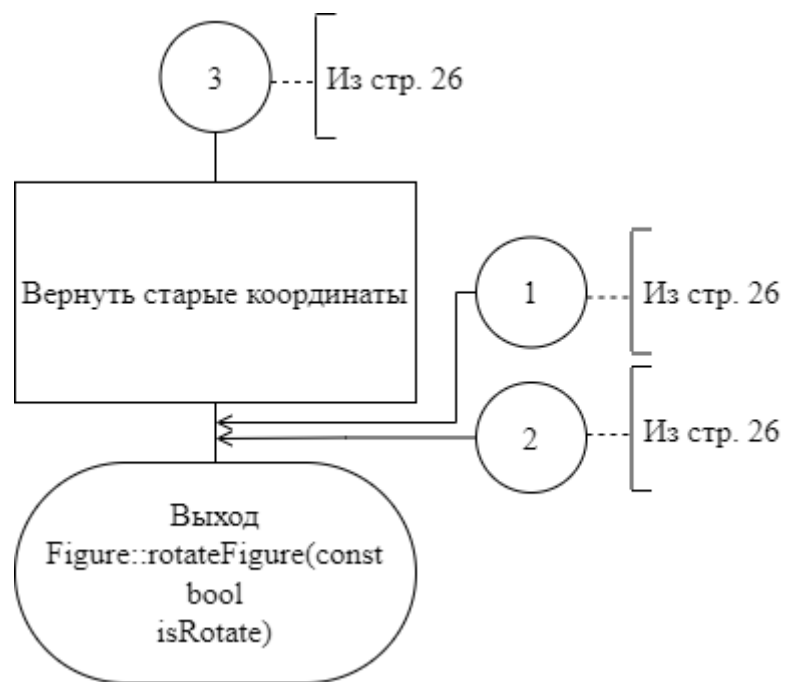


Рисунок 3.2 – Схема алгоритма `Figure::rotateFigure(const bool isRotate)` (часть 2)

3.3.3 Схема алгоритма Figure::fall()

Схема алгоритма падения фигуры. Выход за пределы поля при падении фигуры обозначает ее приземление. Соответственно в этом случае необходимо создать новую фигуру вверху игрового поля.

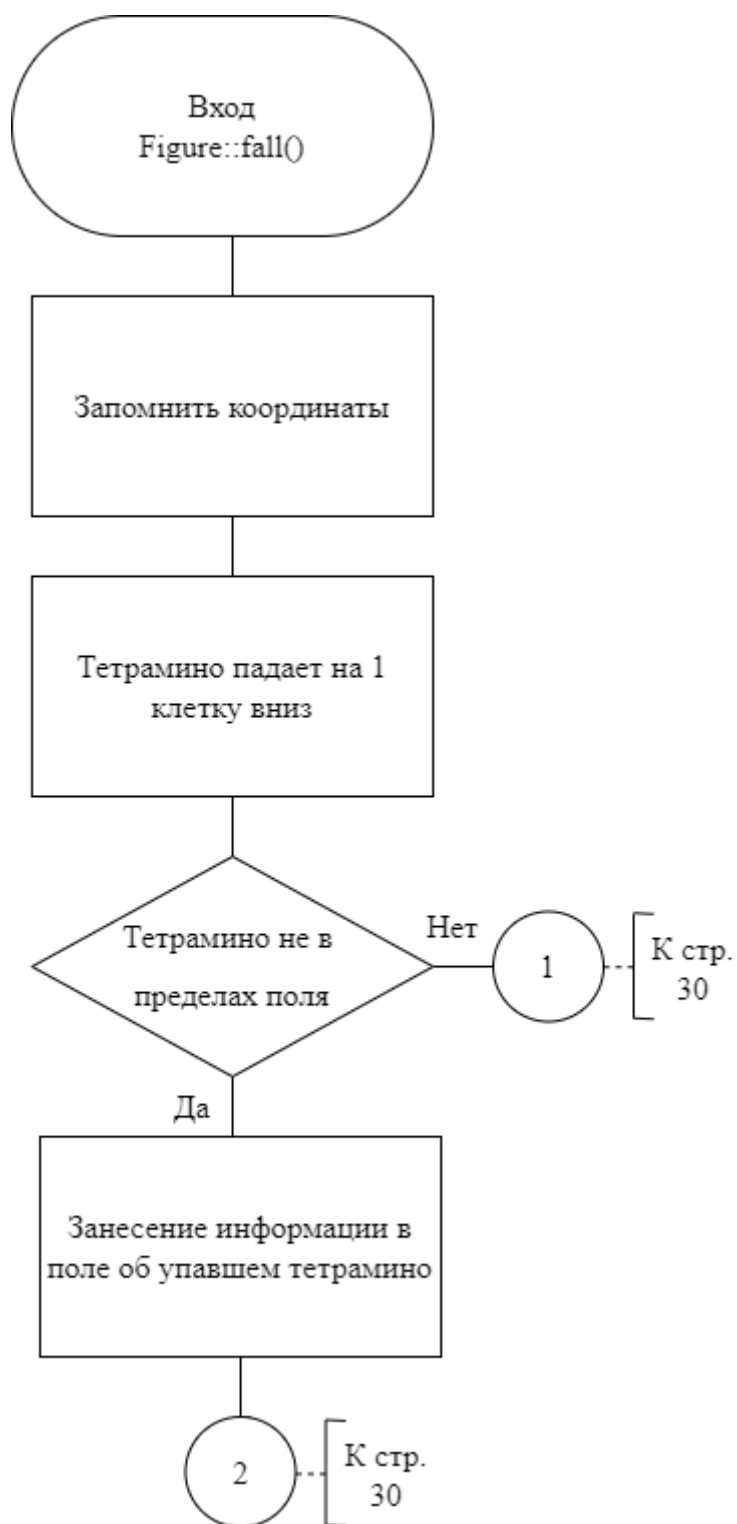


Рисунок 3.3 – Схема алгоритма `Figure::fall()` (часть 1)

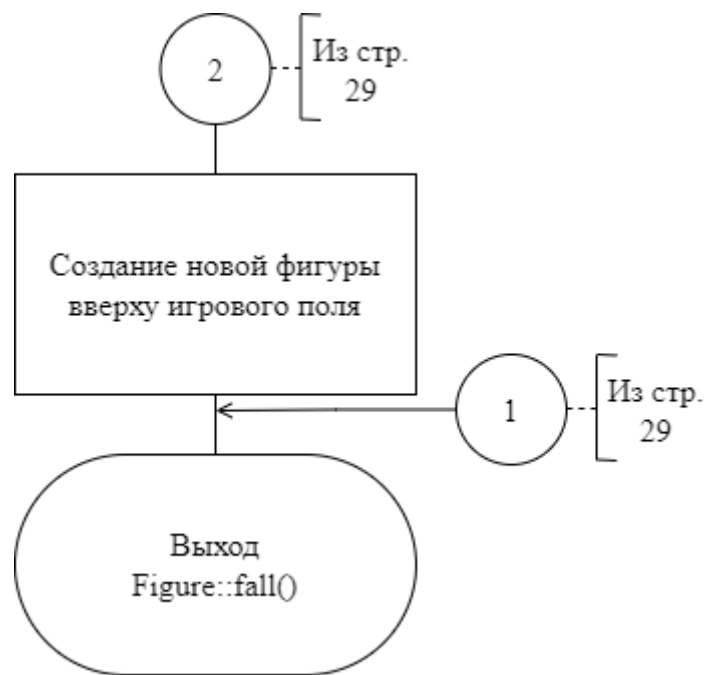


Рисунок 3.3 – Схема алгоритма `Figure::fall()` (часть 2)

3.3.4 Схема алгоритма Field::destroyLines()

Схема алгоритма уничтожения заполненного ряда. При встрече заполненного ряда он уничтожается, а все ряды сверху смещаются вниз. Нижеприведенная блоксхема нуждается в более подробных пояснениях. Изначально переменная k равняется значению переменной i на первой итерации внешнего цикла, а во внутреннем цикле, отвечающем за прохождение по ряду игрового поля, происходит следующее присваивание: $m_field[k][j] = m_field[i][j]$, которое означает следующее: перенести элементы ряда i в ряд k . Обратим внимание на условный блок, если в ряду есть пустые ячейки, то выполняется $k--$, что означает, что ровно до тех пор, пока нет необходимости уничтожать ряд, переменная k будет уменьшаться и ряд будет заменяться самим собой. В противном случае переменная k не уменьшается и ряд заменяется рядом, лежащим на 1 ячейку выше его.

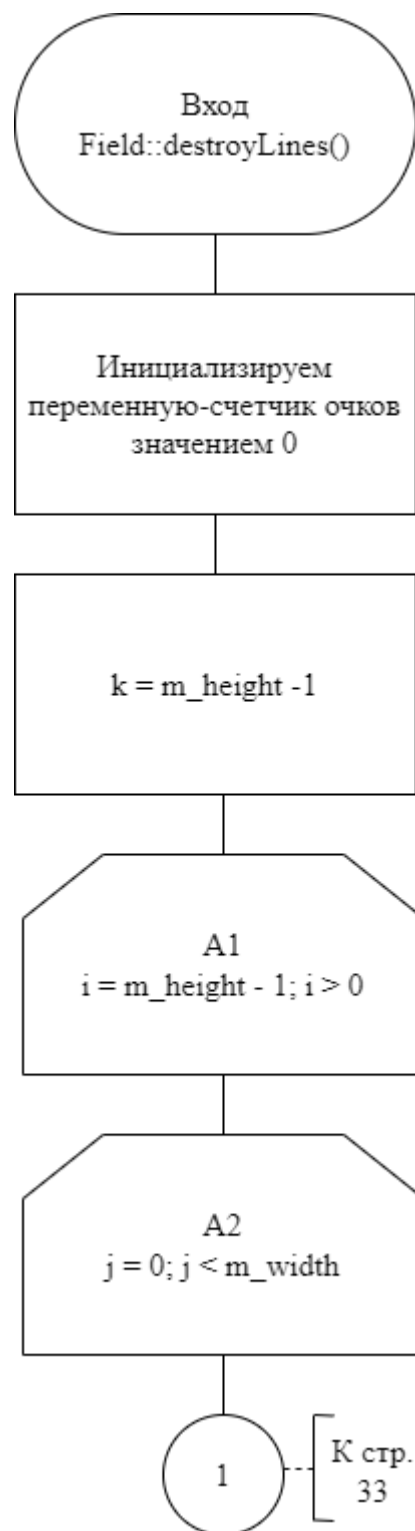


Рисунок 3.4 – Схема алгоритма `Field::destroyLines()` (часть 1)

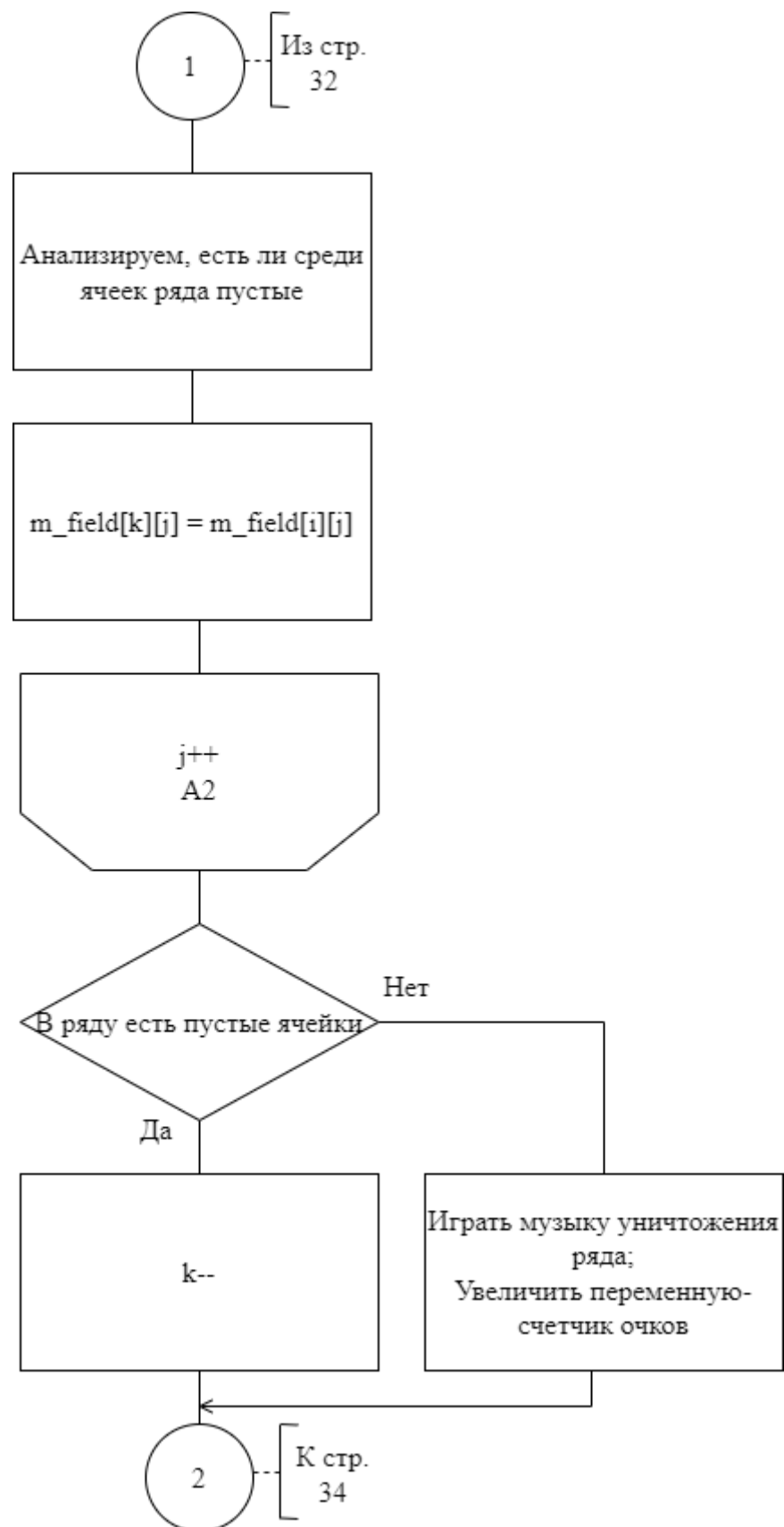


Рисунок 3.4 – Схема алгоритма Field::destroyLines() (часть 2)



Рисунок 3.4 – Схема алгоритма `Field::destroyLines()` (часть 3)

3.3.5 Схема алгоритма Field::isFullfilled()

Схема алгоритма проверки заполнения поля. Хотя бы одна непустая ячейка в 0 строке поля означает его заполнение и поражение в игре.

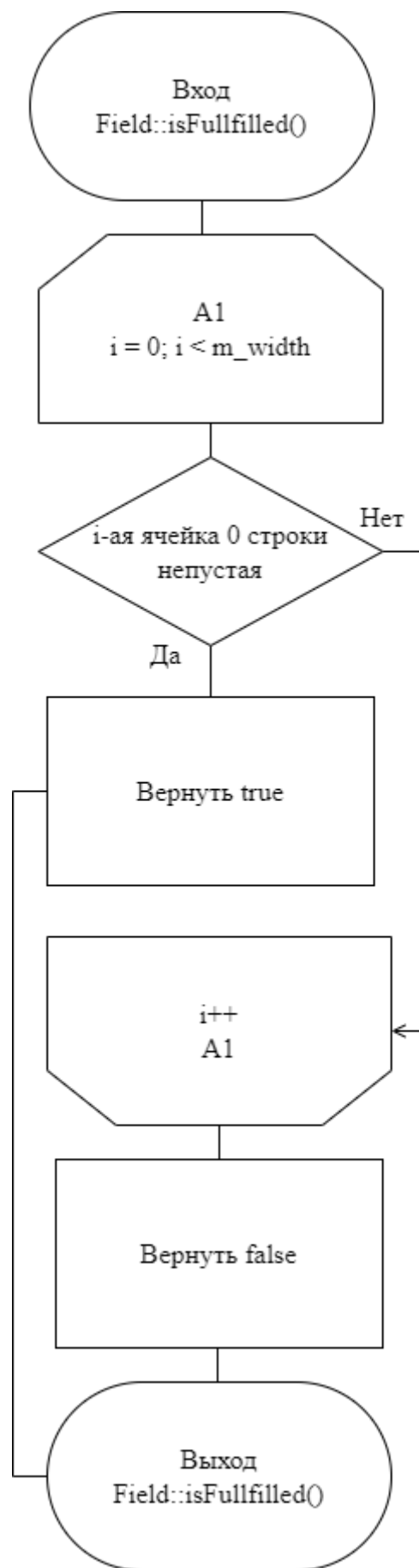



Рисунок 3.5 – Схема алгоритма `Field::isFullfilled()` (часть 1)

4 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

4.1 Запуск меню и игры

4.1.1 Тест 1

Таблица 3 – Тест 1


| | |
|------------------------|---|
| Тестовая ситуация: | Проверка корректности отображения главного меню при запуске программы |
| Исходный набор данных: | Запуск программы |
| Ожидаемый результат: | Корректное открытие главного меню |
| Полученный результат: |  |

4.1.2 Тест 2

Таблица 4 – Тест 2

| | |
|------------------------|--|
| Тестовая ситуация: | Проверка корректности открытия меню настроек |
| Исходный набор данных: | Нажатие на кнопку «Настройки» |

Продолжение таблицы 4 – Тест 2

| | |
|-----------------------|---|
| Ожидаемый результат: | Корректное открытие меню настроек |
| Полученный результат: |  |

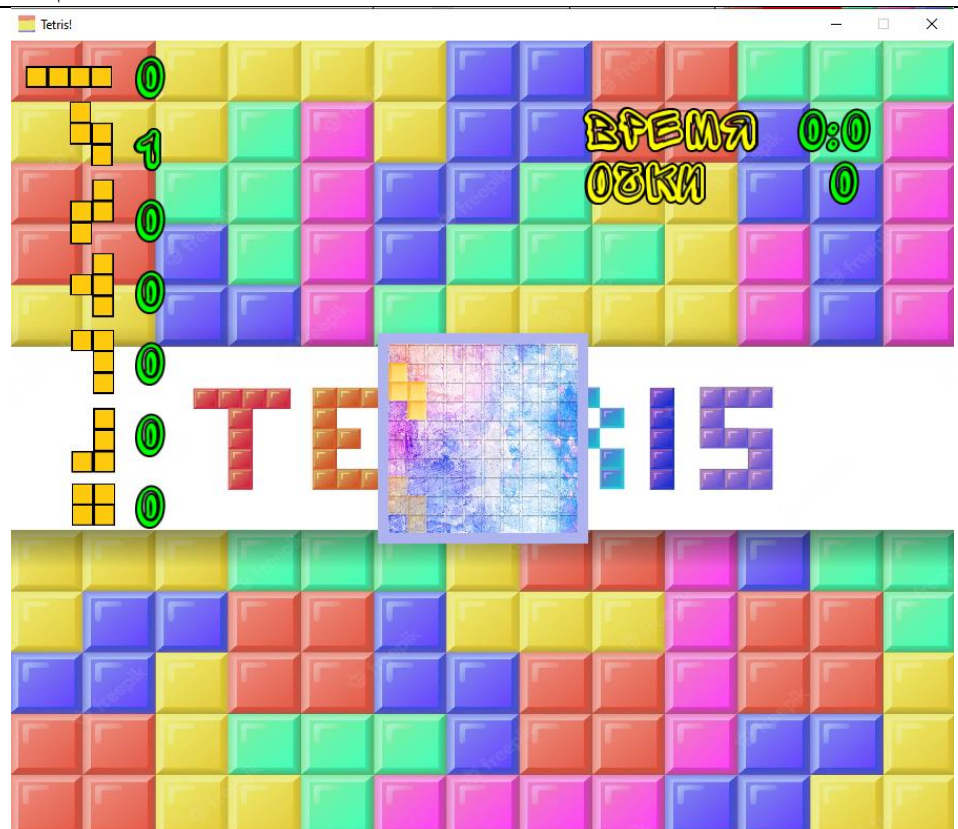
4.1.3 Тест 3

Таблица 5 – Тест 3

| | |
|------------------------|-------------------------------------|
| Тестовая ситуация: | Проверка корректности открытия игры |
| Исходный набор данных: | Нажатие на кнопку «Играть» |
| Ожидаемый результат: | Корректное открытие игры |

Продолжение таблицы 5 – Тест 3

Полученный
результат:



4.2 Применение настроек

4.2.1 Тест 4

Таблица 6 – Тест 4

| | |
|------------------------|--|
| Тестовая ситуация: | Проверка работоспособности опции «Изменить язык» |
| Исходный набор данных: | Нажатие на кнопку «Изменить язык» |
| Ожидаемый результат: | Смена языка на английский |
| Полученный результат: | |

4.2.2 Тест 5

Таблица 7 – Тест 5

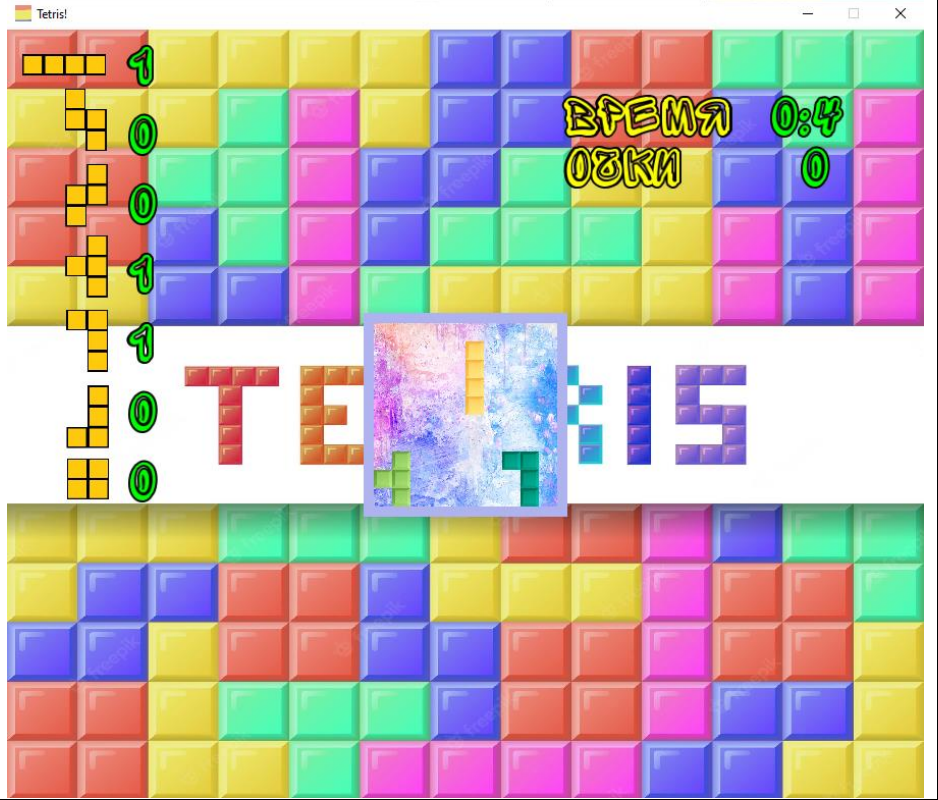
| | |
|------------------------|--|
| Тестовая ситуация: | Проверка работоспособности опции «Решетка» |
| Исходный набор данных: | Нажатие на кнопку «Решетка» |
| Ожидаемый результат: | Отключение решетки на игровом поле |
| Полученный результат: |  <p>The screenshot shows the Tetris game interface. The game board is filled with colorful blocks. The score is 0, and the time is 0:8. The word 'TETRIS' is displayed in the center. The 'Grid' option is disabled, as indicated by the text 'ВРЕМЯ 0:8' and 'ОЗКИ 0'.</p> |

4.2.3 Тест 6

Таблица 8 – Тест 6

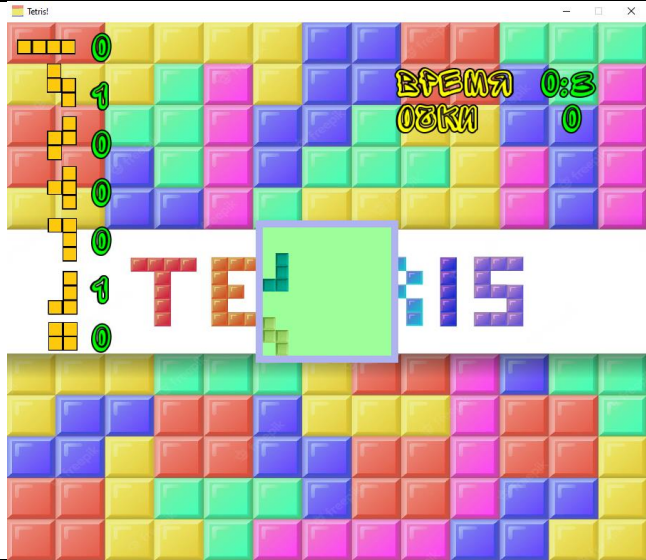
| | |
|------------------------|--|
| Тестовая ситуация: | Проверка работоспособности опции «Тень фигуры» |
| Исходный набор данных: | Нажатие на кнопку «Тень фигуры» |
| Ожидаемый результат: | Отключение тени фигуры на игровом поле |

Продолжение таблицы 8 – Тест 6

| | |
|-----------------------|--|
| Полученный результат: |  |
|-----------------------|--|

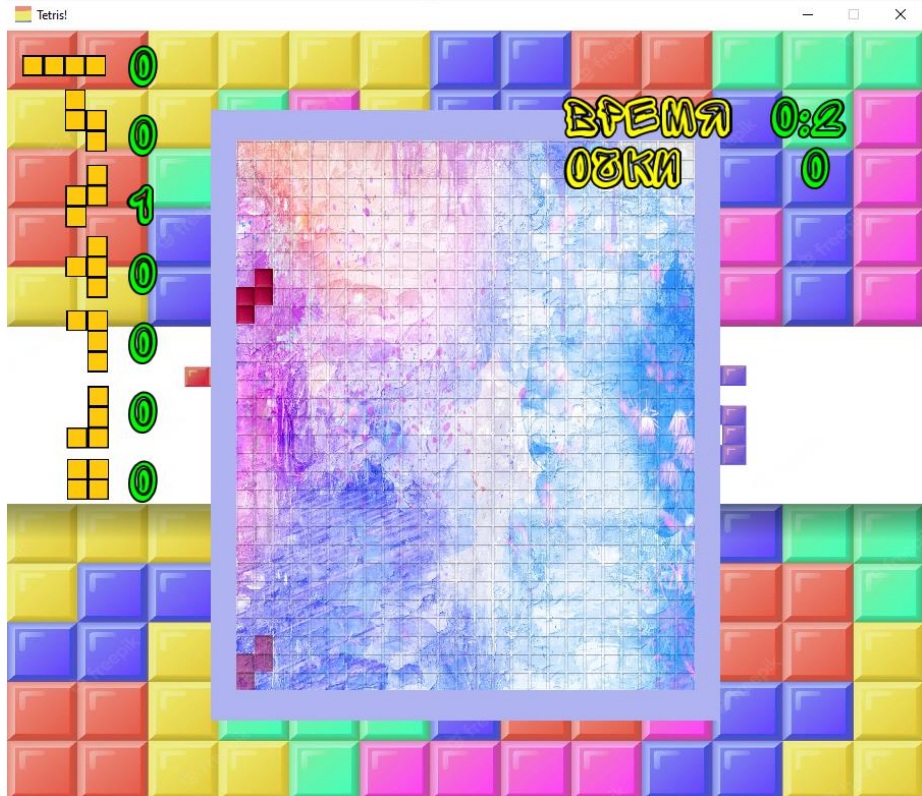
4.2.4 Тест 7

Таблица 9 – Тест 7

| | |
|------------------------|--|
| Тестовая ситуация: | Проверка работоспособности опции «Показать фон поля» |
| Исходный набор данных: | Нажатие на кнопку «Показать фон поля» |
| Ожидаемый результат: | Отключение фона игрового поля |
| Полученный результат: |  |

4.2.5 Тест 8

Таблица 10 – Тест 8

| | |
|------------------------|---|
| Тестовая ситуация: | Изменение размеров игрового поля |
| Исходный набор данных: | Нажатия на кнопки изменения размеров игрового поля |
| Ожидаемый результат: | Изменение размера игрового поля |
| Полученный результат: |  |

4.2.6 Тест 9

Таблица 11 – Тест 9

| | |
|------------------------|------------------------------------|
| Тестовая ситуация: | Отключение звука приложения |
| Исходный набор данных: | Нажатие на кнопку «Отключить звук» |
| Ожидаемый результат: | Отключение звука приложения |
| Полученный результат: | Отключение звука приложения |

4.3 Проверка работоспособности игры

4.3.1 Тест 10

Таблица 12 – Тест 10

| | |
|------------------------|--|
| Тестовая ситуация: | Управление тетрамино нажатиями клавиш «Вверх», «Вниз», «Влево», «Вправо» |
| Исходный набор данных: | Нажатия клавиш «Вверх», «Вниз», «Влево», «Вправо» |
| Ожидаемый результат: | Поведение тетрамино согласно нажатым клавишам |
| Полученный результат: | Корректное движение тетрамино согласно нажатым клавишам |

4.3.2 Тест 11

Таблица 13 – Тест 11

| | |
|------------------------|--|
| Тестовая ситуация: | Падение тетрамино в самый низ игрового поля |
| Исходный набор данных: | Падение тетрамино в самый низ игрового поля |
| Ожидаемый результат: | Появление нового тетрамино сверху игрового поля |
| Полученный результат: |  |

4.3.3 Тест 12

Таблица 14 – Тест 12

| | |
|------------------------|---|
| Тестовая ситуация: | Падение тетрамино на другое тетрамино |
| Исходный набор данных: | Падение тетрамино на другое тетрамино |
| Ожидаемый результат: | Появление нового тетрамино сверху игрового поля |
| Полученный результат: |  |

4.3.4 Тест 13

Таблица 15 – Тест 13

| | |
|------------------------|---|
| Тестовая ситуация: | Попытка вывода тетрамино за пределы игрового поля путем движения его в горизонтальном направлении |
| Исходный набор данных: | Движение тетрамино вправо |
| Ожидаемый результат: | Коллизия с игровым полем, невозможность вывести тетрамино за пределы игрового поля |
| Полученный результат: | Коллизия с игровым полем, невозможность вывести тетрамино за пределы игрового поля |

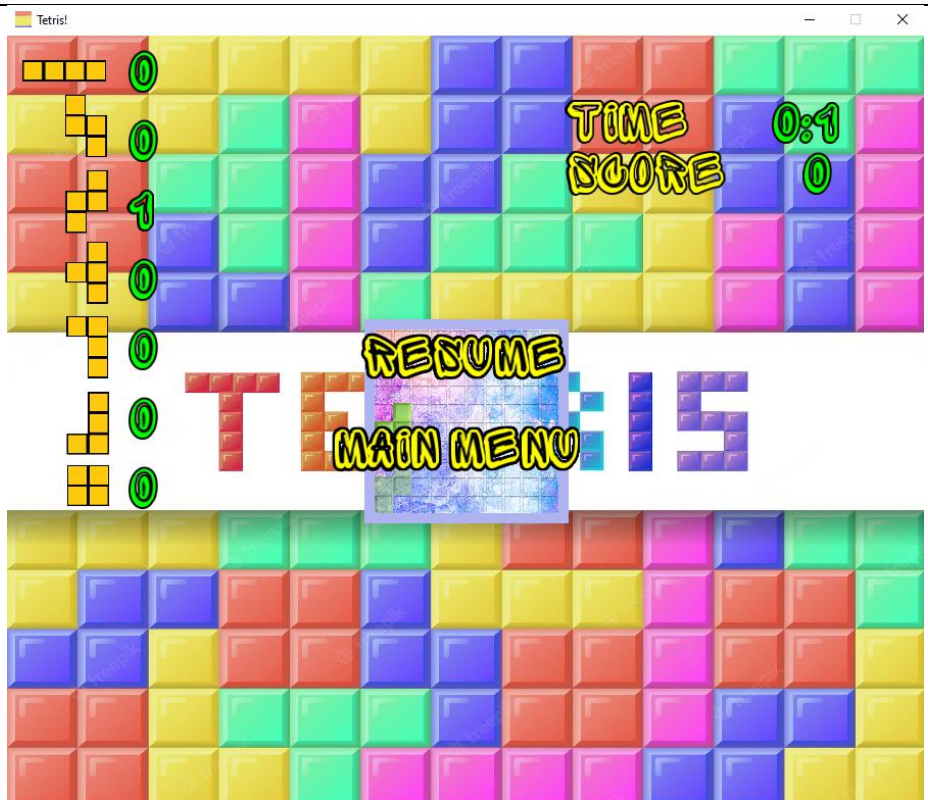
4.3.5 Тест 14

Таблица 16 – Тест 14

| | |
|------------------------|---|
| Тестовая ситуация: | Попытка вывода тетрамино за пределы игрового поля путем поворота его вблизи границы игрового поля |
| Исходный набор данных: | Поворот тетрамино |
| Ожидаемый результат: | Коллизия с игровым полем, невозможность вывести тетрамино за пределы игрового поля |
| Полученный результат: | Коллизия с игровым полем, невозможность вывести тетрамино за пределы игрового поля |

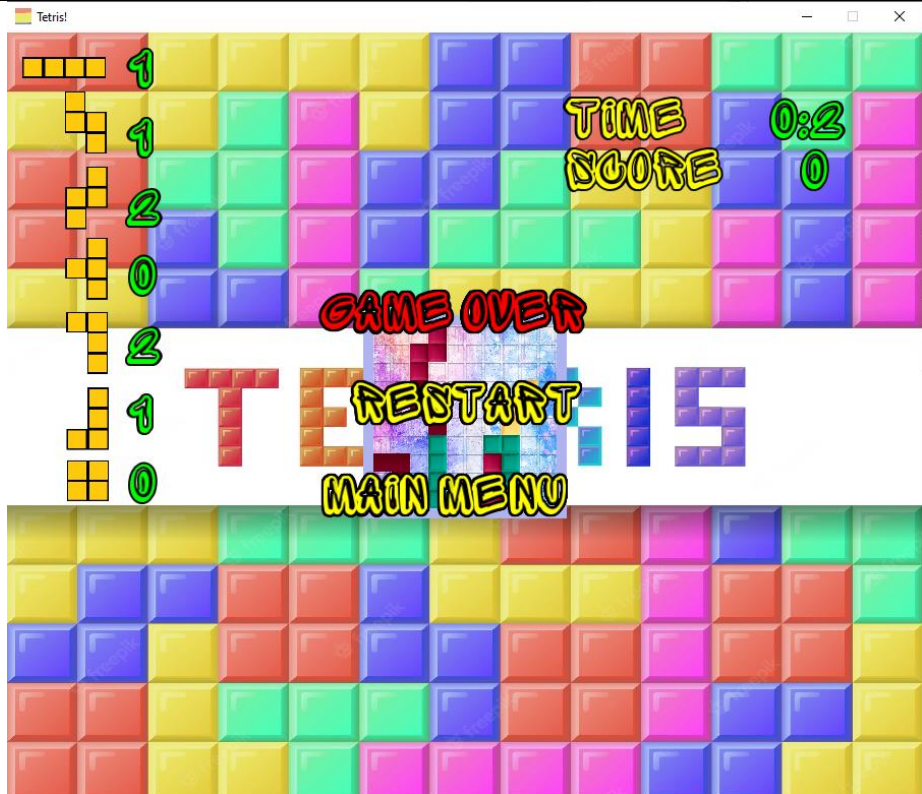
4.3.6 Тест 15

Таблица 17 – Тест 15

| | |
|------------------------|---|
| Тестовая ситуация: | Постановка игры на паузу |
| Исходный набор данных: | Нажатие клавиши «Escape» во время игры |
| Ожидаемый результат: | Постановка игры на паузу |
| Полученный результат: |  <p>The screenshot shows the Tetris! game interface. The game board is filled with various colored blocks (yellow, blue, red, green, pink). A pause menu is overlaid on the screen, featuring the text 'TETRIS!' in large letters. Below the title, there are two buttons: 'RESUME' and 'MAIN MENU'. The score is displayed as '0' and the time as '0:1'. The game is paused, and the background is dimmed.</p> |

4.3.7 Тест 16

Таблица 18 – Тест 16

| | |
|------------------------|---|
| Тестовая ситуация: | Завершение игры |
| Исходный набор данных: | Поражение в игре |
| Ожидаемый результат: | Появление меню завершения игры |
| Полученный результат: |  |

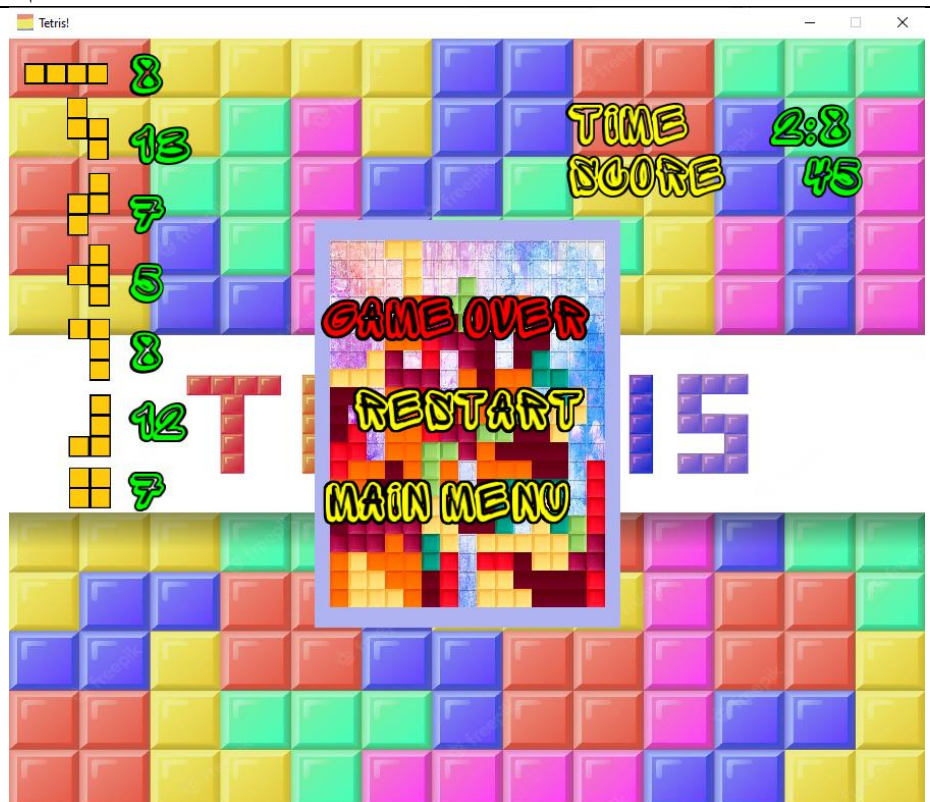
4.3.8 Тест 17

Таблица 19 – Тест 17

| | |
|------------------------|---|
| Тестовая ситуация: | Корректное ведение статистики в игре(время, очки, упавшие фигуры) |
| Исходный набор данных: | Проведение игровой сессии длительностью приблизительно 2 минуты |
| Ожидаемый результат: | Корректное ведение статистики в игре(время, очки, упавшие фигуры) |

Продолжение таблицы 19 – Тест 17

Полученный
результат:



5 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

5.1 Установка

Для установки программного обеспечения необходимо два файла:

- setup.exe
- Setup.msi



| | | | |
|---|------------------|--------------------|----------|
|  setup.exe | 17.12.2022 15:43 | Приложение | 584 КБ |
|  Setup.msi | 17.12.2022 15:43 | Пакет установщи... | 9 575 КБ |

Рисунок 5.1 – Установка (этап 1)

Далее необходимо запустить файл setup.exe, вследствие чего откроется следующее окно:

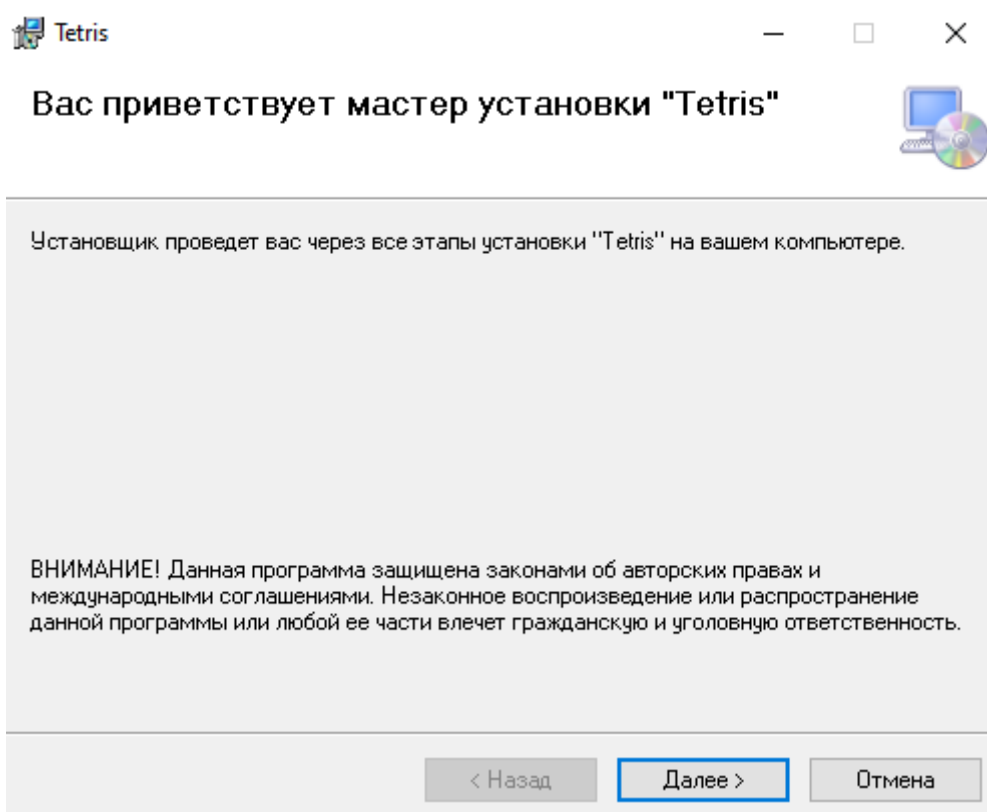


Рисунок 5.2 – Установка (этап 2)

Далее следует этап выбора пути установки приложения и выбора пользователей, для которых оно будет установлено:

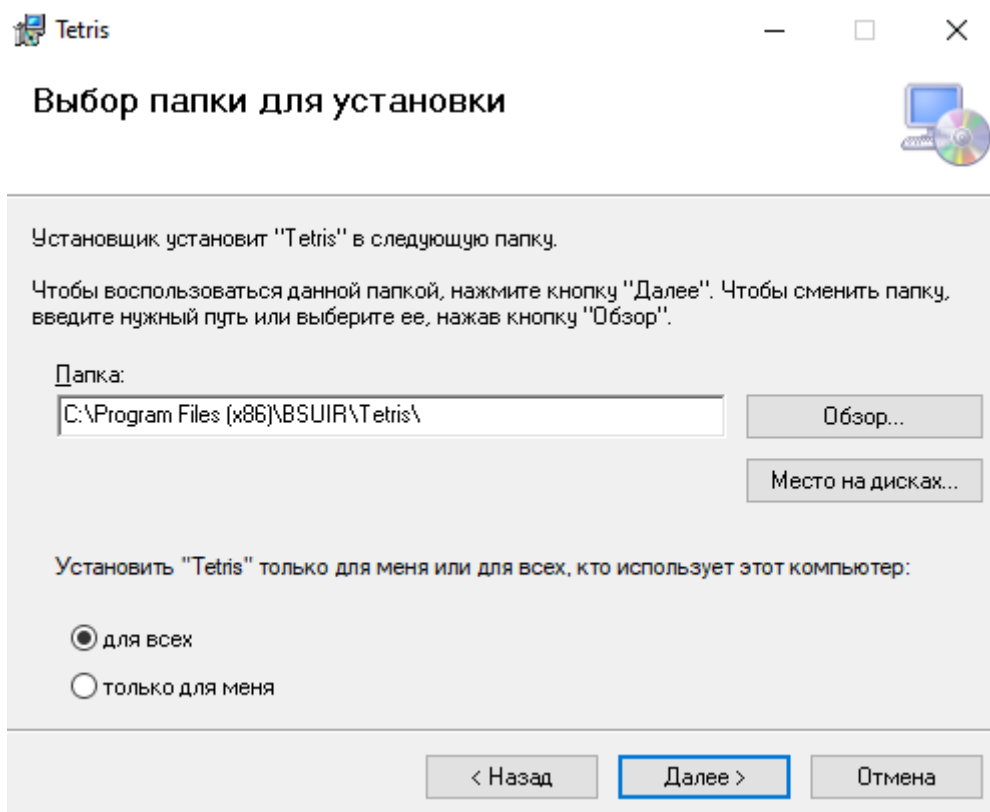


Рисунок 5.3 – Установка (этап 3)

На следующем шаге пользователю будет предложено подтвердить установку:

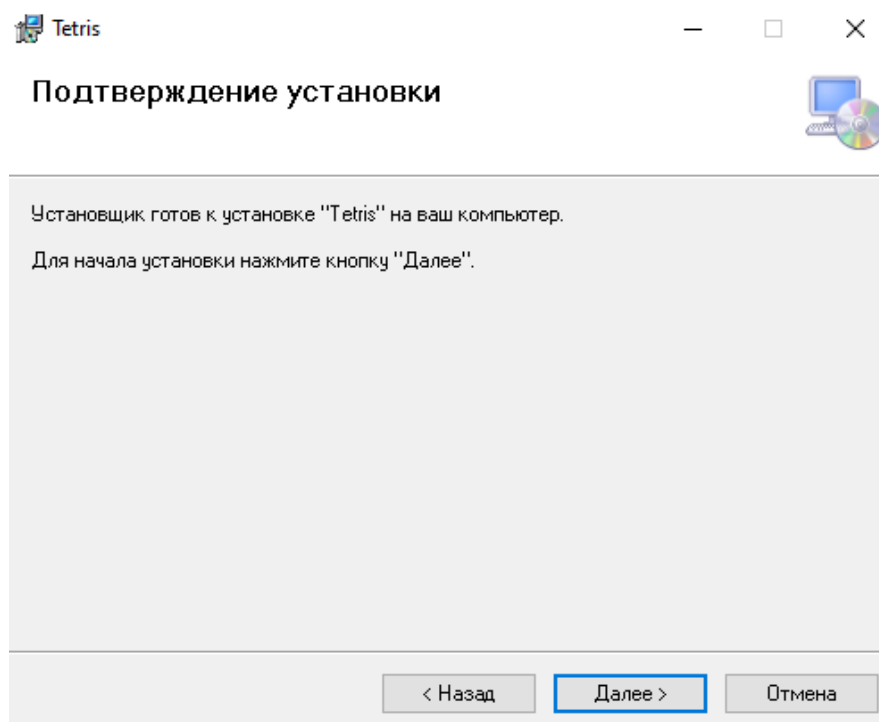


Рисунок 5.4 – Установка (этап 4)

Впоследствии пользователю будет выведена информация о том, что приложение успешно установлено на его компьютер:

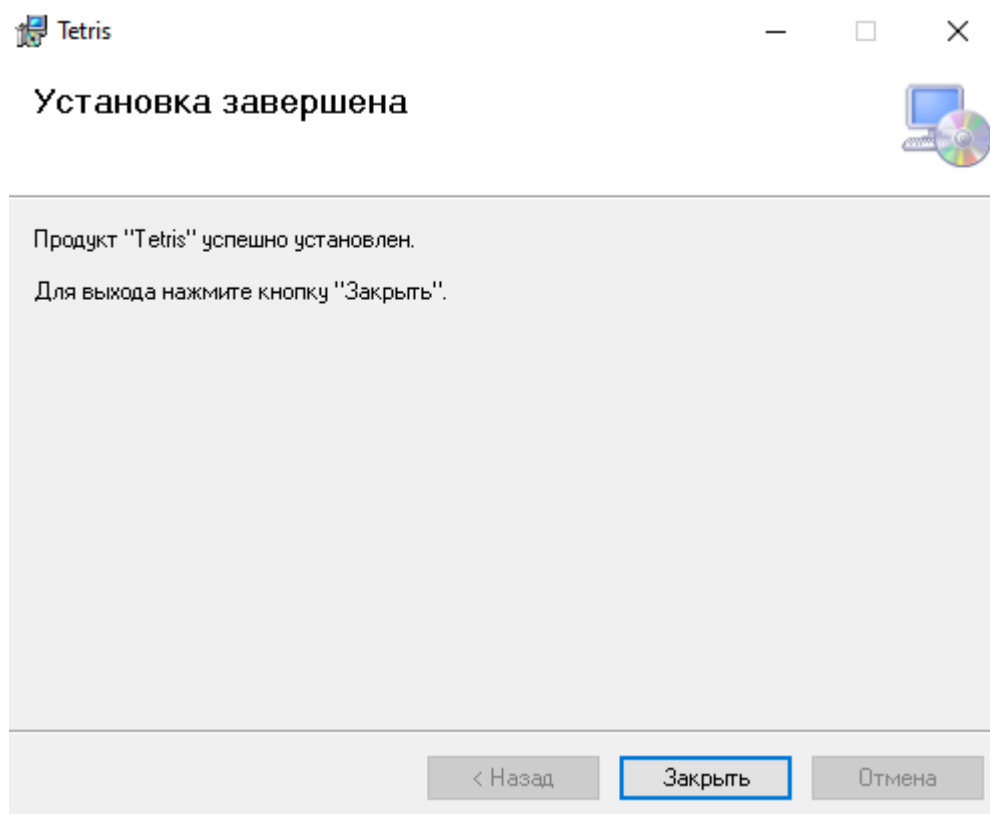


Рисунок 5.5 – Установка (этап 5)

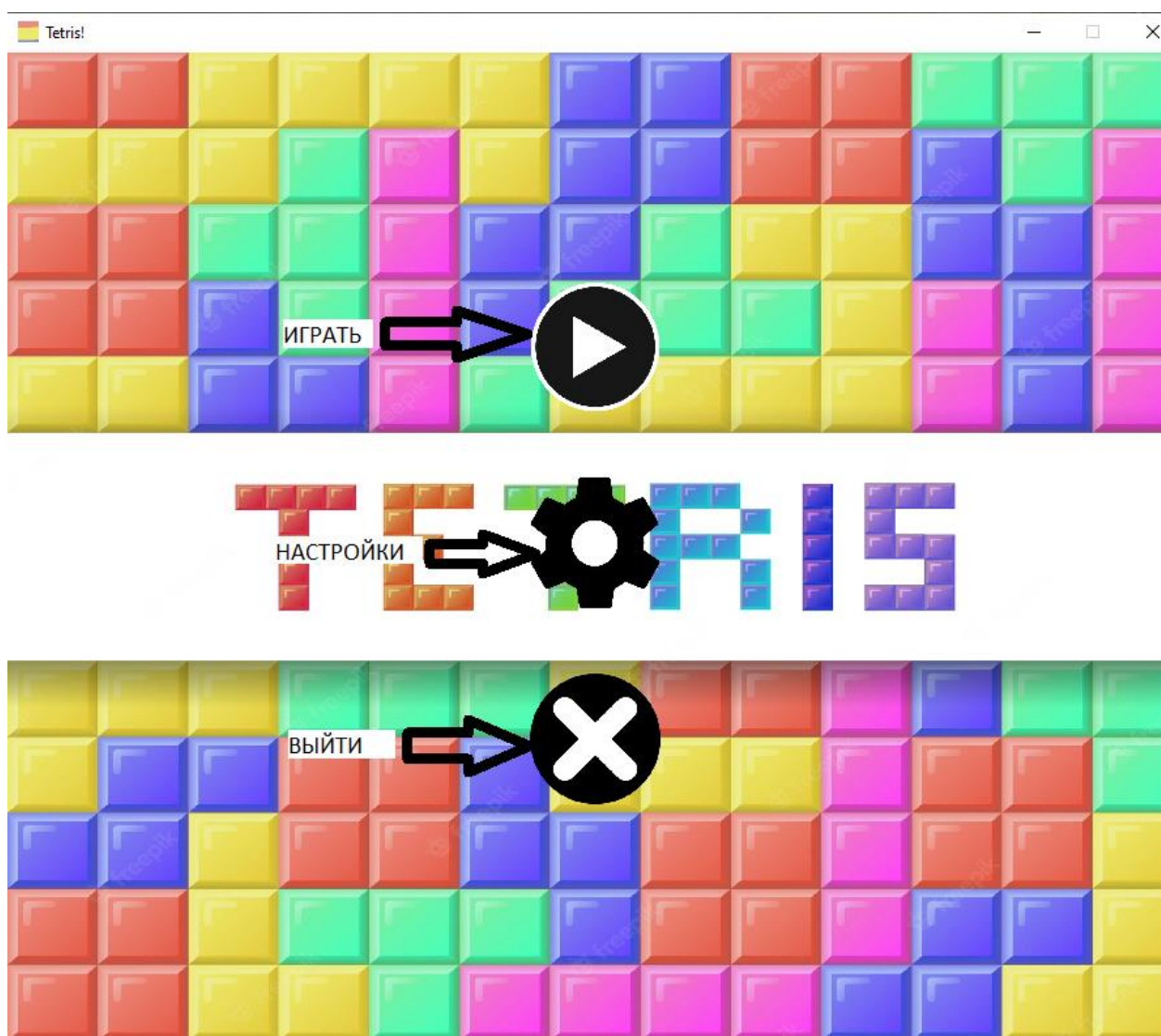
5.2 Использование

5.2.1 Работа с главным меню

В главном меню предусмотрено 3 опции:

- Играть
- Настройки
- Выйти

При нажатии на кнопку «Играть» начнется основная игра. При нажатии на кнопку «Настройки» будет произведен вход в меню настроек. При нажатии кнопки «Выйти» будет произведен выход из приложения. На следующем шаге пользователю будет предложено подтвердить установку:

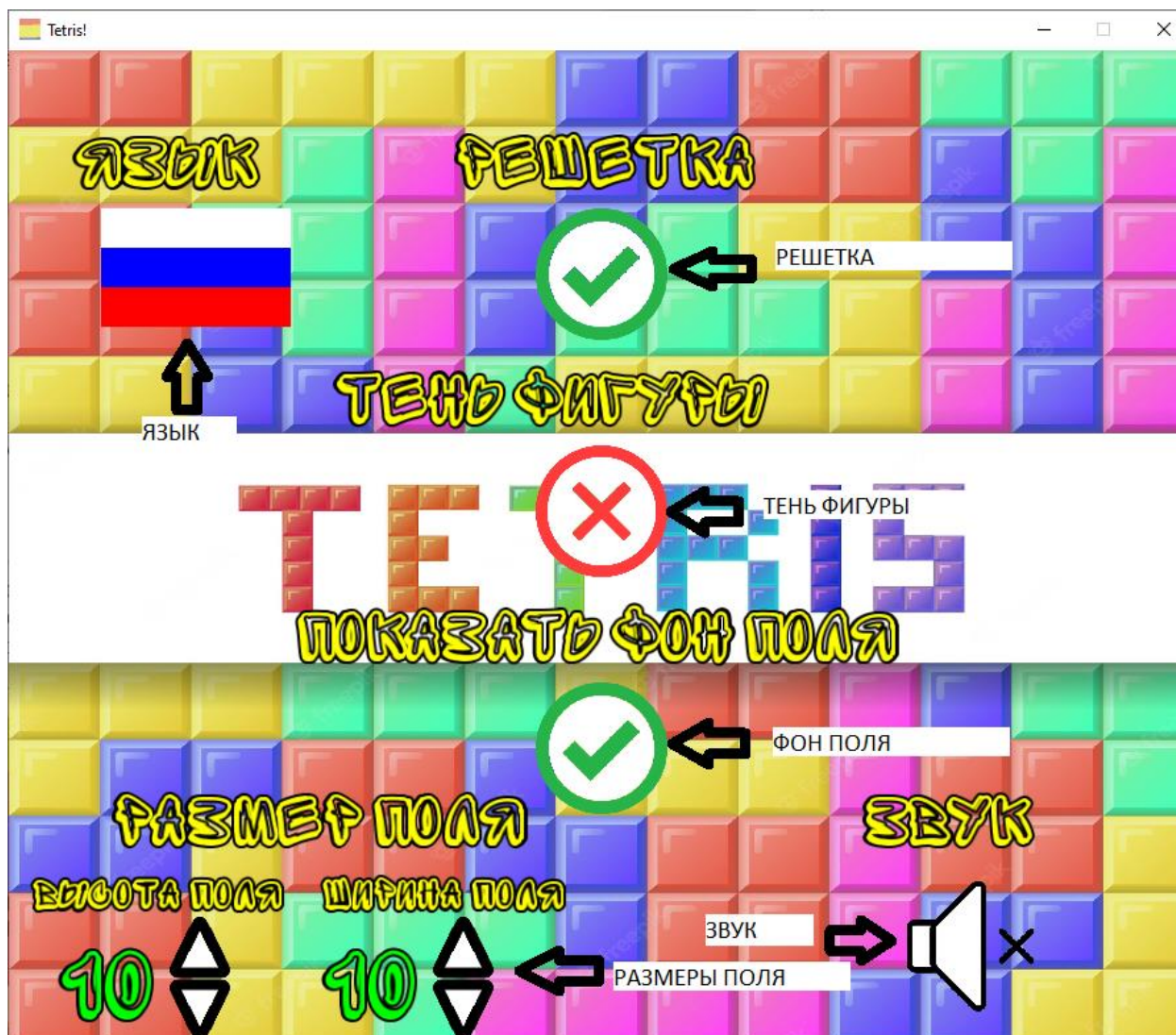


5.2.2 Работа с меню настроек

В меню настроек предусмотрено 6 опций:

- Язык
- Решетка
- Тень
- Фон
- Размеры поля
- Звук

Опция язык позволяет переключаться между русским и английским языком. Решетка позволяет включать и отключить решетку поля. Тень позволяет включать и отключать тень фигуры. Фон позволяет включать и отключить фон игрового поля. Опция размеров поля позволяет редактировать размеры поля. Опция звука позволяет включать и выключать звук:

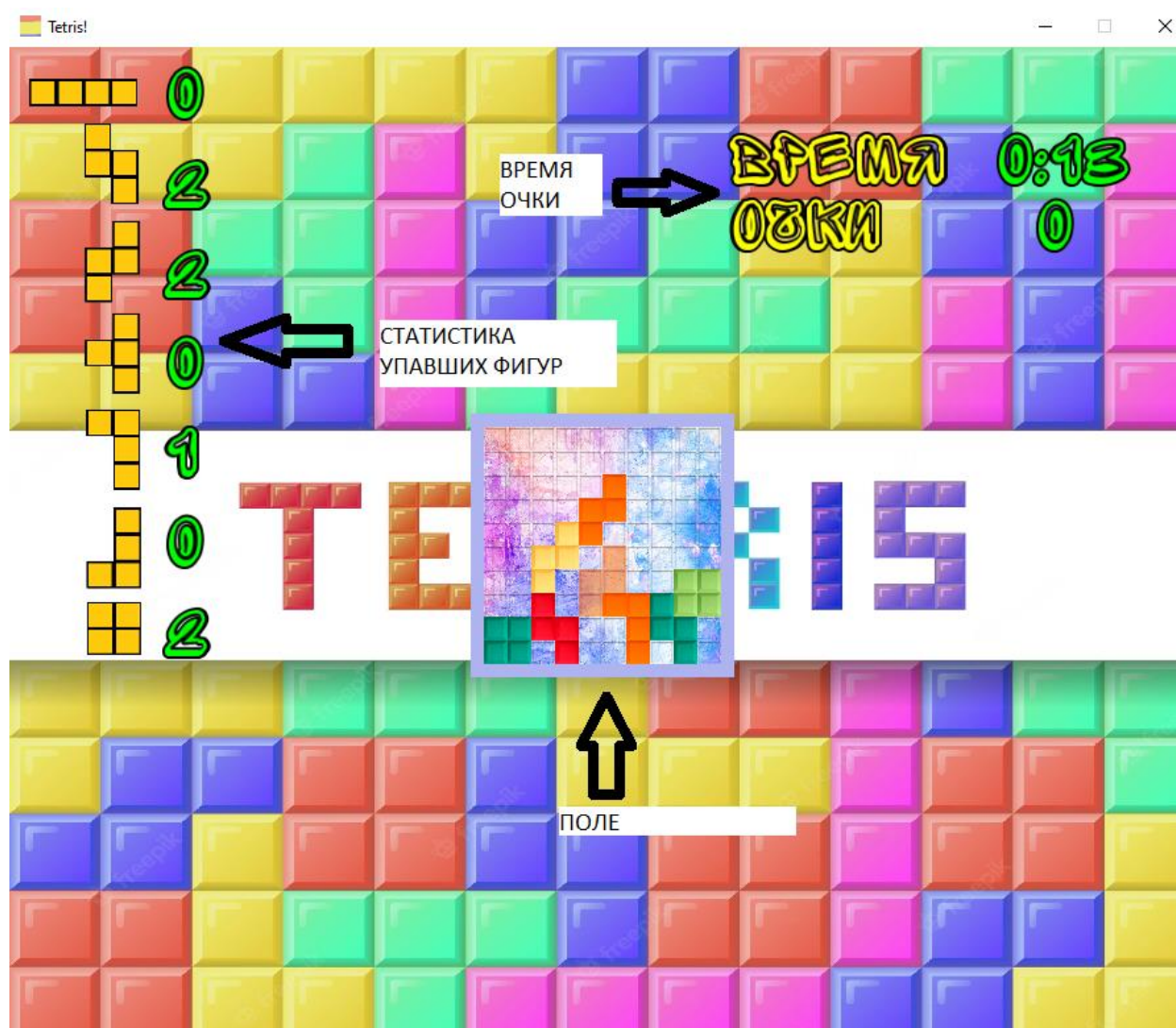


5.2.3 Игра

Для управления тетрамино предназначено 4 клавиши:

- «Вверх»
- «Вниз»
- «Влево»
- «Вправо»

Нажатие клавиши «Вверх» приведет к повороту тетрамино по часовой стрелке. Нажатие клавиши «Вниз» приведет к ускорению падения тетрамино. Нажатие клавиши «Влево» приведет к перемещению тетрамино влево. Нажатие клавиши «Вправо» приведет к перемещению тетрамино вправо. Для выхода из игры в меню паузы необходимо нажать клавишу «Escape». Во время игры в правом верхнем углу ведется статистика игрового времени и очков. В левом верхнем углу ведется статистика упавших фигур.



ЗАКЛЮЧЕНИЕ

В конечном счете, разработанная игра «Тетрис» представляет собой программное средство, позволяющее насладиться любимой игрой с приятной графикой и музыкальным сопровождением. Приложение позволяет вести статистику во время игры, что делает игровой процесс более интересным, предоставляет множество настроек для игры, что вносит разнообразие в нее. Программное средство поддерживает 2 языка, что позволяет охватить более широкую аудиторию.

Данное приложение является простым для использования и интуитивно понятным, так как все функции реализованы с использованием максимально понятного графического интерфейса. Само управление в игре также является интуитивно понятным и удобным для использования. Для успешного выполнения всех поставленных целей потребовалось ознакомиться со средой разработки Visual Studio. Для создания графического интерфейса требовалось изучить различные аналоги игры в данном жанре, а также графическую библиотеку SFML.

Все выявленные в процессе тестирования приложения неполадки устранены. Возможна дальнейшая доработка при выявлении неполадок.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- [1] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2004. – 1 ч.
- [2] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2005. – 2 ч.
- [3] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2007. – 3 ч.
- [4] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2013. – 4 ч.
- [5] Серебряная Л.В. Структуры и алгоритмы обработки данных: учеб.-метод. пособие / Л.В. Серебряная, И.М. Марина. – Минск: БГУИР, 2013. – 5 с.
- [6] Вирт Н. Алгоритмы и структуры данных / Н. Вирт. – Москва: Мир 1989. – 90 с.
- [7] Глухова Л.А. Основы алгоритмизации и программирования: учебное пособие / Л.А. Глухова. – Минск: БГУИР, 2006. – 1 ч.
- [8] Библиотека SFML[Электронный ресурс]. – Режим доступа: <https://www.sfm1-dev.org/> – Дата обращения: 10.12.2022.
- [9] Ravesli[Электронный ресурс]. – Режим доступа: <https://ravesli.com/> – Дата обращения: 25.11.2022.
- [10] Документация Microsoft к Visual Studio [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/> – Дата обращения: 02.12.2022.

ПРИЛОЖЕНИЕ А
(обязательное)
Исходный код программы
Модуль main

```
#include <SFML\Graphics.hpp>
#include <cmath>
#include <ctime>
#include <vector>
#include <windows.h>
#include "TextureSpriteWork.h"
#include "Random.h"
#include "Figure.h";
#include "Menus.h"
#include "Statistics.h"
#include "Music_Sounds.h"

using namespace sf;

int main()
{
    Music_Sounds::initialize();
    srand(time(0));
    rand();

    Field field{ 10, 10 };
    //Create object - main window
    RenderWindow window(VideoMode(18*50, 18 * 42),
        "Tetris!", sf::Style::Close | sf::Style::Titlebar);

    //Set icon for app
    Image icon;
    if (!icon.loadFromFile("../\\images\\background.jpg"))
    {
        return 1;
    }
    window.setIcon(32, 32, icon.getPixelsPtr());

    Texture tilesImage;
    tilesImage.loadFromFile("../\\images\\tiles.jpg",
        IntRect(0, 0, 1320, 192));

    //Texture and sprite to create grid
    Texture squareT;
    squareT.loadFromFile("../\\images\\square.png");
    Texture gameBackground;
    gameBackground.loadFromFile("../\\images\\gameBackground
.jpg");
```



```

field.setBackgroundImage(gameBackground);

Sprite tile{ getTile(tilesImage, genRandNum(0,6)) };

Figure figure{field, genRandNum(0,6)};
figure.setImage(tilesImage);
field.setTilesImage(tilesImage);
field.setGridImage(squareT);
figure.setTileType(genRandNum(0, 6));

Texture backgroundImage;
backgroundImage.loadFromFile("../\\images\\background.jpg");
Sprite background { backgroundImage };
setSpriteSize(background, 900.0f, 756.0f);
background.setPosition(0.0f, 0.0f);

Font font;
font.loadFromFile(TextCharacteristics_default::txtFont)
;
Text txtElapsedTime;
txtElapsedTime.setFont(font);
txtElapsedTime.setPosition(550, 50);
txtElapsedTime.setCharacterSize(TextCharacteristics_default::txtCharacterSize);
txtElapsedTime.setOutlineThickness(TextCharacteristics_default::txtOutlineThickness);
txtElapsedTime.setOutlineColor(TextCharacteristics_default::txtOutlineColor);
txtElapsedTime.setFillColor(TextCharacteristics_default::txtFillColor);
txtElapsedTime.setStyle(TextCharacteristics_default::txtStyle);

Texture allFiguresImage;
allFiguresImage.loadFromFile("../\\images\\figures.png");
;
Sprite allFigures{ allFiguresImage };
setSpriteSize(allFigures, 100.0f, 449.0f);
allFigures.setPosition(10.0f, 20.0f);

/*WORK FROM HERE!*/
Text txtFigures;
txtFigures.setFont(font);
txtFigures.setPosition(120, 0);
txtFigures.setCharacterSize(TextCharacteristics_default::txtCharacterSize);

```

```

        txtFigures.setOutlineThickness(TextCharacteristics_default::txtOutlineThickness);
        txtFigures.setOutlineColor(TextCharacteristics_default::txtOutlineColor);
        txtFigures.setFillColors(Color::Green);
        txtFigures.setStyle(TextCharacteristics_default::txtStyle);
        txtFigures.setString("0");

        Text txtElapsedTimeValue;
        txtElapsedTimeValue.setFont(font);
        txtElapsedTimeValue.setPosition(750, 50);
        txtElapsedTimeValue.setCharacterSize(TextCharacteristics_default::txtCharacterSize);
        txtElapsedTimeValue.setOutlineThickness(TextCharacteristics_default::txtOutlineThickness);
        txtElapsedTimeValue.setOutlineColor(TextCharacteristics_default::txtOutlineColor);
        txtElapsedTimeValue.setFillColors(Color::Green);
        txtElapsedTimeValue.setStyle(TextCharacteristics_default::txtStyle);
        txtElapsedTimeValue.setString("0:0");

        Text txtScore;
        txtScore.setFont(font);
        txtScore.setPosition(550, 100);
        txtScore.setCharacterSize(TextCharacteristics_default::txtCharacterSize);
        txtScore.setOutlineThickness(TextCharacteristics_default::txtOutlineThickness);
        txtScore.setOutlineColor(TextCharacteristics_default::txtOutlineColor);
        txtScore.setFillColors(TextCharacteristics_default::txtFillColor);
        txtScore.setStyle(TextCharacteristics_default::txtStyle);

        Text txtScoreValue;
        txtScoreValue.setFont(font);
        txtScoreValue.setPosition(780, 100);
        txtScoreValue.setCharacterSize(TextCharacteristics_default::txtCharacterSize);
        txtScoreValue.setOutlineThickness(TextCharacteristics_default::txtOutlineThickness);
        txtScoreValue.setOutlineColor(TextCharacteristics_default::txtOutlineColor);
        txtScoreValue.setFillColors(Color::Green);

```

```

    txtScoreValue.setStyle(TextCharacteristics_default::txt
Style);
    txtScoreValue.setString("0");

    //Main menu may call settings menu, which must have
opportunity to change options of game
    mainMenu(window, field, figure);
    txtElapsedTime.setString(Language::getPhrase(12));
    txtScore.setString(Language::getPhrase(13));

    //Figure hasn't moved yet
    int dx{ 0 };

    //Figure hasn't rotated yet
    bool rotate{ false };

    //Game is not over
    bool isGameOver{ false };

    //Set elapsed time
    float timer = 0;

    //Elapsed time
    float elapsedTime = 0;
    Statistics statistics;

    // Clock (timer)
    Clock clock;

    //Variable which assists to correctly increment
statistics about figures
    bool isIncrementedOnce{ false };

    //Play game music
    Music_Sounds::s_musicGame.play();

    float want_fps = 80;
    Clock loop_timer;

    //While window is open
    while (window.isOpen())
    {
        //Set delay in fallings
        float delay = 0.3;

        float time = clock.getElapsedTime().asSeconds();
        clock.restart();
        timer += time;
    }

```

```

elapsedTime += time;

//Determine type of event
Event event;

while (window.pollEvent(event))
{
    //Close the app
    if (event.type == Event::Closed)
        window.close();

    //Key press process
    if (event.type == Event::KeyPressed)
    {
        if (event.key.code == Keyboard::Up)
            rotate = true; //Rotate figure
        else
            if (event.key.code ==
Keyboard::Left)
                dx = -1; //Move figure left
            else
                if (event.key.code ==
Keyboard::Right)
                    dx = 1; //Move figure right
                else
                    if (event.key.code ==
Keyboard::Escape)
                        {

                            Music_Sounds::s_musicGame.pause();
                                if (pauseMenu(window,
field, figure))
                                    {

                                        Music_Sounds::s_musicGame.stop();

                                        txtElapsedTimeValue.setString("0:0");

                                        txtScoreValue.setString("0");

                                        statistics.reset();

                                                                mainMenu(window,
field, figure);

                                                                clock.restart();
                                                                elapsedTime = 0;

                                        txtElapsedTimeValue.setString(Language::getPhrase(12));

```

```

txtScore.setString(Language::getPhrase(13));
    }

Music_Sounds::s_musicGame.play();
    }
}

//Press down - accelerate tetramino
if (Keyboard::isKeyPressed(Keyboard::Down)) delay
= 0.05;

//Set main window background color
window.clear(Color(156, 255, 153, 255));

window.draw(background);

//Move or rotate on pressed keys
figure.moveFigure(dx);
figure.rotateFigure(rotate);

//Draw field(figures, which already fell) with
grid
field.drawField(window);

//draw figure
figure.drawFigure(window);

//Fall figure
if (timer > delay)
{
    isIncrementedOnce = false;
    figure.fall();
    timer = 0;
}

//Update time
if (elapsedTime > 1)
{
    elapsedTime = 0;
    statistics.setTime({
statistics.getTimeMinutes(), statistics.getTimeSeconds() +
1 });
    if (statistics.getTimeSeconds() > 59)

```

```

        statistics.setTime({
statistics.getTimeMinutes() + 1, 0 });

        txtElapsedTimeValue.setString(std::to_string(statistics
.getTimeMinutes()) + ":" +
std::to_string(statistics.getTimeSeconds()));
    }

        //Destroy fullfilled lines and update score
        statistics.setScore(statistics.getScore() +
field.destroyLines());

        txtScoreValue.setString(std::to_string(statistics.getSc
ore()));

        //Update statistics with figures types
        if (figure.isJustAppeared() && !isIncrementedOnce)
        {

statistics.incfigureTypesFell(figure.getFigureType());
        isIncrementedOnce = true;
        }

        window.draw(txtElapsedTime);
        window.draw(txtElapsedTimeValue);
        window.draw(txtScore);
        window.draw(txtScoreValue);

        window.draw(allFigures);
        for (int i = 0; i < 7; ++i)
        {
            txtFigures.setPosition(120, 68*i);

            //This block was made to solve undefined
behaviour of last figure fell value in statistics, solution
is !!!AWFUL!!!
            if (field.isFullfilled())
                if (i == figure.getFigureType())
                    statistics.setfigureTypesFell(i,
statistics.getfigureTypesFell(i) - 1);

            txtFigures.setString(std::to_string(statistics.getfigur
eTypesFell(i)));
            window.draw(txtFigures); //HERE WRITE CODE TO
DRAW NUMBER OF FIGURES FELL
        }
        txtFigures.setPosition(120, 0);

```

```

        //Stop game if there is no more space for
tetramino
        if (field.isFullfilled())
        {
            Music_Sounds::s_soundGameOver.play();
            Music_Sounds::s_musicGame.stop();
            Sleep(1000);

            if (gameOverMenu(window, field, figure))
                mainMenu(window, field, figure);
            else
            {
                field.reset();
                figure.reset();
            }
            isIncrementedOnce = false;
            Music_Sounds::s_musicGame.play();
            txtElapsedTimeValue.setString("0:0");
            txtScoreValue.setString("0");
            statistics.reset();
            clock.restart();
            elapsedTime = 0;

            txtElapsedTime.setString(Language::getPhrase(12));
            txtScore.setString(Language::getPhrase(13));
        }

        dx = 0;
        rotate = false;

        //Draw main window
        window.display();

        sf::Int32 frame_duration =
loop_timer.getElapsedTime().asMilliseconds();
        sf::Int32 time_to_sleep = int(1000.f / want_fps) -
frame_duration;
        if (time_to_sleep > 0) {
            sf::sleep(sf::milliseconds(time_to_sleep));
        }
        loop_timer.restart();
    }

    return 0;
}

```

ПРИЛОЖЕНИЕ Б
(обязательное)
Исходный код программы
Модуль Figure

```
#include "Figure.h"

void Figure::makeCoordinates()
{
    for (int i = 0; i < 4; i++)
    {
        for (int i = 0; i < 4; i++)
        {
            m_tilesCoordinates[i].x =
m_figures[m_figureNum][i] % 2;
            m_tilesCoordinates[i].y =
m_figures[m_figureNum][i] / 2;
        }
    }

    void Figure::move(const int step, const Direction&
direction)
    {
        if (direction == Direction::VERTICAL)
            for (int i = 0; i < 4; ++i)
                m_tilesCoordinates[i].y += step;
        else
            for (int i = 0; i < 4; ++i)
                m_tilesCoordinates[i].x += step;
    }

    void Figure::savePreviousCoordinates(Point*
coordinates)
    {
        for (int i = 0; i < 4; ++i)
            coordinates[i] = m_tilesCoordinates[i];
    }

    void Figure::calculateShadowCoordinates()
    {
        Point tilesCoordinatesCopy[4];
        savePreviousCoordinates(tilesCoordinatesCopy);
        while (this->isInField())
        {
            this->move(1, Direction::VERTICAL);
            if (!this->isInField())
            {
```



```

        for (int i = 0; i < 4; ++i)
        {
            m_shadowCoordinates[i].x =
m_tilesCoordinates[i].x;
            m_shadowCoordinates[i].y =
m_tilesCoordinates[i].y - 1;
            m_tilesCoordinates[i] =
tilesCoordinatesCopy[i];
        }
        break;
    }
}

Figure::Figure(Field& field, int figureNum, int
tileType) : m_field{ field }, m_tilesCoordinates { {0, 0},
{ 0,0 }, { 0,0 }, { 0,0 } },
    m_figureNum{ figureNum }, m_shadowOn{ true }
{
    makeCoordinates();
}

void Figure::setImage(const Texture& tilesImage)
{
    m_tilesImage = tilesImage;
}

void Figure::setTileType(const int tileType)
{
    m_tileType = tileType;
}

bool Figure::isInField()
{
    for (int i = 0; i < 4; i++)
        if (m_tilesCoordinates[i].x < 0 ||
m_tilesCoordinates[i].x >= m_field.getWidth() ||
m_tilesCoordinates[i].y < 0 || m_tilesCoordinates[i].y >=
m_field.getHeight()) return 0;
        else
            if
(m_field[m_tilesCoordinates[i].y][m_tilesCoordinates[i].x]
!= -1)

                return false;

    return true;
}

```

```

void Figure::drawFigure(RenderWindow& window)
{
    Sprite tile{ getTile(m_tilesImage, m_tileType) };
    Sprite m_shadowImage{ tile };
    m_shadowImage.setColor(Color(255, 255, 255, 128));
    if (m_shadowOn)
        this->calculateShadowCoordinates();
    float displacement_y{ (window.getSize().y -
m_field.getHeight() * 18.0f) / 2};
    float displacement_x{ (window.getSize().x -
m_field.getWidth() * 18.0f) / 2};
    for (int i = 0; i < 4; ++i)
    {

        tile.setPosition(static_cast<float>(m_tilesCoordinates[
i].x * 18) + displacement_x,
static_cast<float>(m_tilesCoordinates[i].y * 18) +
displacement_y);
        if (m_shadowOn)
        {

            m_shadowImage.setPosition(static_cast<float>(m_shadowCo
ordinates[i].x * 18) + displacement_x,
static_cast<float>(m_shadowCoordinates[i].y * 18) +
displacement_y);
            window.draw(m_shadowImage);
        }
        //Draw one tile
        window.draw(tile);
    }
}

void Figure::moveFigure(const int dx)
{
    Point previousTilesCoordinates[4];
    savePreviousCoordinates(previousTilesCoordinates);
    move(dx, Direction::HORIZONTAL);
    if (!isInField()) {
        for (int i = 0; i < 4; i++)
            m_tilesCoordinates[i] =
previousTilesCoordinates[i];
    }
}

void Figure::rotateFigure(const bool isRotate)
{
    // Rotate figure if needed
    if (isRotate)

```

```

    {
        Point p = m_tilesCoordinates[1]; // Center of
rotation
        Point previousTilesCoordinates[4];
        savePreviousCoordinates(previousTilesCoordinates);

        for (int i = 0; i < 4; i++)
        {
            int x = m_tilesCoordinates[i].y - p.y; // y -
y0
            int y = m_tilesCoordinates[i].x - p.x; // x -
x0

            m_tilesCoordinates[i].x = p.x - x;
            m_tilesCoordinates[i].y = p.y + y;
        }

        if (!isInField()) {
            for (int i = 0; i < 4; i++)
                m_tilesCoordinates[i] =
previousTilesCoordinates[i];
        }
    }

void Figure::fall()
{
    Point previousTilesCoordinates[4];
    savePreviousCoordinates(previousTilesCoordinates);
    move(1, Direction::VERTICAL);

    if (!this->isInField()) {

        //Save information about fallen figure in field
array to draw frame in future
        for (int i = 0; i < 4; i++)
m_field[previousTilesCoordinates[i].y][previousTilesCoordin
ates[i].x] = m_tileType;

        m_tileType = 1 + genRandNum(0, 6);
        //this->setTileType(m_tileType);
        m_figureNum = genRandNum(0, 6);
        int displacement = genRandNum(0,
m_field.getWidth() - 2);
        for (int i = 0; i < 4; i++)
        {
            m_tilesCoordinates[i].x =
m_figures[m_figureNum][i] % 2 + displacement;

```

```

        m_tilesCoordinates[i].y =
m_figures[m_figureNum][i] / 2;
    }

}

void Figure::shadowSwitch()
{
    m_shadowOn ^= 1;
}

bool Figure::isShadowOn()
{
    return m_shadowOn;
}

int Figure::getFigureType()
{
    return m_figureNum;
}

bool Figure::isJustAppeared()
{
    return (m_tilesCoordinates[0].y == 0 ||
m_tilesCoordinates[1].y == 0 || m_tilesCoordinates[2].y ==
0 || m_tilesCoordinates[3].y == 0);
}

void Figure::reset()
{
    m_tileType = 1 + genRandNum(0, 6);
    m_figureNum = genRandNum(0, 6);
    for (int i = 0; i < 4; i++)
    {
        m_tilesCoordinates[i].x =
m_figures[m_figureNum][i] % 2;
        m_tilesCoordinates[i].y =
m_figures[m_figureNum][i] / 2;
    }
}

```

ПРИЛОЖЕНИЕ В
(обязательное)
Исходный код программы
Модуль Field

```
#include "Field.h"

Field::Field(const int height, const int width) :
m_height{ height }, m_width{ width }, m_gridOn{ true },
m_backgroundOn{ true }
{
    m_field.resize(height);
    for (int i = 0; i < m_field.size(); ++i)
        m_field[i].resize(width);
    for (int i = 0; i < m_height; ++i)
        for (int j = 0; j < m_width; ++j)
            m_field[i][j] = -1;
}

void Field::setSize(const int height, const int width)
{
    m_height = height;
    m_width = width;
    m_field.resize(height);
    for (int i = 0; i < m_field.size(); ++i)
    {
        m_field[i].resize(width);
        std::fill(m_field[i].begin(), m_field[i].end(), -
1);
    }
    setSpriteSize(m_backgroundImage, 18.0f * m_width, 18.0f
* m_height);
}

int Field::getHeight()
{
    return m_height;
}

int Field::getWidth()
{
    return m_width;
}

std::vector<int>& Field::operator[](const int index)
{
    return m_field[index];
}
```

```

int Field::destroyLines()
{
    int score{ 0 };
    int k = m_height - 1;
    for (int i = m_height - 1; i > 0; i--)
    {
        int count = 0;
        for (int j = 0; j < m_width; j++)
        {
            if (m_field[i][j] != -1) count++;
            m_field[k][j] = m_field[i][j];
        }
        if (count < m_width) k--;
        else { Music_Sounds::s_soundStageClear.play();
score += m_width; }
    }
    return score;
}

bool Field::isFullfilled()
{
    for (int i = 0; i < m_width; ++i)
        if (m_field[0][i] != -1)
            return true;
    return false;
}

void Field::drawField(RenderWindow& window)
{
    RectangleShape rectangle{ Vector2f(m_width * 20.0f,
m_height * 20.0f)};
    rectangle.setFillColor(Color(175, 180, 240, 255));
    rectangle.setPosition(Vector2f((window.getSize().x -
m_width * 20.0f) / 2, (window.getSize().y - m_height *
20.0f) / 2));
    window.draw(rectangle);
    rectangle.setSize(Vector2f(m_width * 18.0f, m_height *
18.0f));
    rectangle.setFillColor(Color(156, 255, 153, 255));
    rectangle.setPosition(Vector2f((window.getSize().x -
m_width * 18.0f) / 2, (window.getSize().y - m_height *
18.0f) / 2));
    window.draw(rectangle);

    float displacement_y{ (window.getSize().y - m_height *
18.0f) / 2 };

```

```

        float displacement_x{ (window.getSize().x - m_width *
18.0f) / 2 };
        if (m_backgroundOn)
        {
            m_backgroundImage.setPosition(displacement_x,
displacement_y);
            window.draw(m_backgroundImage);
        }
        for (int i = 0; i < m_height; i++)
            for (int j = 0; j < m_width; j++)
            {
                if (m_gridOn)
                {
                    m_gridImage.setPosition(j * 18.0f +
displacement_x, i * 18.0f + displacement_y);
                    window.draw(m_gridImage);
                }
                if (m_field[i][j] == -1) continue;
                Sprite tile = getTile(m_tilesImage,
m_field[i][j]);
                tile.setPosition(j * 18 + displacement_x, i *
18 + displacement_y);
                window.draw(tile);
            }
    }

    void Field::setTilesImage(const Texture& tilesImage)
    {
        m_tilesImage = tilesImage;
    }

    void Field::setGridImage(const Texture& gridImage)
    {
        m_gridImage.setTexture(gridImage);
        setSpriteSize(m_gridImage, 18.0f, 18.0f);
    }

    void Field::setBackgroundImage(const Texture&
backgroundImage)
    {
        m_backgroundImage.setTexture(backgroundImage);
        setSpriteSize(m_backgroundImage, 18.0f * m_width, 18.0f
* m_height);
        m_backgroundImage.setPosition(0, 0);
    }

    void Field::gridSwitch()
    {

```

```

m_gridOn ^= 1;
}

bool Field::isGridOn()
{
return m_gridOn;
}

void Field::backgroundSwitch()
{
m_backgroundOn ^= 1;
}

bool Field::isBackgroundOn()
{
return m_backgroundOn;
}

void Field::reset()
{
for (int i = 0; i < m_height; ++i)
    for (int j = 0; j < m_width; ++j)
        m_field[i][j] = -1;
}

```