МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)

БГТУ.СМК-Ф-4.2-К5-01

Факультет И Информационные и управляющие системы
шифр наименование
Кафедра И5 Информационные системы и программная инженерия
шифр наименование
Дисциплина Программирование на ЯВУ

КУРСОВАЯ РАБОТА

НА ТЕМУ:

Игра «Tetris»

Выполнил студент группы И596

Орехов Руслан Вячеславович

Фамилия И.О.

Орлов О. В.

Фамилия И.О. Подпись

Оценка

«_____» 2021г.

САНКТ-ПЕТЕРБУРГ

2021 г.

СОДЕРЖАНИЕ

ОСНОВНАЯ ЧАСТЬ

1. Постановка задачи

Написать игру «Тетрис». В зависимости от варианта игры (классический или с усложнениями), результатов объектно-ориентированного анализа и проектирования для данного варианта и пользовательского интерфейса – сложность от 3 до 5.

Требования к работе:

- Использование C++.
- Использование SDL версии 2.x.
- Решение задачи методом ООП.

Основные требования к программе:

- Графический интерфейс.
- Минимум два окна.
- Использование абстракции.
- Использование ивентов.

2. Описание классов

2.1 Класс WINDOW

Это абстрактный класс, хранящий всю информацию об окне и содержащий основные методы работы с окном, такие как: очистить окно, нанести на окно поверхность, написать текст, обновить изображение и другие вспомогательные методы.

Поля класса:

1. SDL_Window *window – объект SDL2 для работы с окном;
2. SDL_Renderer *renderer – объект SDL2 для работы с окном;
3. SDL_Texture *texture – объект SDL2 для работы с окном, текстура;
4. SDL_Surface *temp_surface – объект SDL2, вспомогательная поверхность для отрисовки;
5. SDL_Rect info, dst – объекты для хранения размеров и координат;
6. TTF_Font *font – объект SDL2 для работы с текстом;
7. SDL_Color color – объект SDL2 для хранения цвета;
8. SDL_Surface *text_surface – объект SDL2, поверхность;
9. char name[30] – массив с названием окна;
10. POINT pos – структура с координатами окна;
11. POINT lastPosMouse – структура с координатами позиции мышки;
12. bool focus – переменная для хранения информации в фокусе ли окно или нет.

Методы класса:

1. WINDOW() – конструктор класса;
2. virtual ~WINDOW() – деструктор класса;
3. void setPosition(int, int) – метод установки позиции окна;
4. void setSize(int, int) – метод установки размеров окна;
5. void setName(char*) – метод установка названия окна;
6. void recreateWindow() – метод пересоздание окна;
7. void centerWindow() – метод для возврат окна на координаты, хранящиеся в pos;
8. void clearRender() – метод очистки render;
9. void clearTempSurface() – метод для очистки промежуточной поверхности;
10. void updateWindow() – метод для обновление изображения на окне;
11. void drawSurface(SDL_Surface*, int, int) – метод для копирования поверхности;
12. void drawText(char*, int, int, int, unsigned char, unsigned char, unsigned char) – метод для написания текста;
13. void setMousePos(int, int) – метод для запись координат курсора в структуру lastPosMouse;
14. POINT getMousePos() – метод для получение координат курсора;
15. void setFocus(bool) – метод для установки фокуса;
16. bool getFocus() – метод для получения значения фокуса;
17. virtual void pressLeftButton() {} – метод для обработки нажатия левого щелчка мышки.

2.2 Класс SCORE

Это класс для работы со списком рекордов и файлом, где он хранится.

Поля класса:

1. char *wayFile = "score.bin" – название файла с рекордами;
2. NODE mass[7] – массив хранящий рекорды, считанные из файла;
3. FILE *file – переменная для работы с файлом;
4. int i, j – вспомогательные переменные для работы с циклами.

Методы класса:

1. SCORE() – конструктор класса;
2. ~SCORE() – деструктор класса;
3. int lengNumber(int) – метод, возвращающий длину числа;
4. void viewMass() – вспомогательный метод для отладки;
5. void sortMass() – метод для сортировки массива;
6. void clear() – метод для очистки файла;

7. void load() – метод для загрузки файла;
8. void save() – метод для сохранения файла;
9. void add(int, int) – метод для добавление рекорда;
10. NODE *getMass() – метод получения указателя на массив.

2.3 Класс LOGICS_TETRIS

Это класс, содержащий всю логику игры тетрис. В нем хранится массив с полем, шаблоны фигур, координаты текущей фигуры, информация о следующей фигуре и другие вспомогательные переменные. Так же этот класс содержит указатель на класс SCORE. Это нужно для того, чтобы иметь возможность добавлять новый рекорд.

Поля класса:

1. int M = 26, N = 10 – переменные, хранящие размеры поля;
2. int **field - указатель на массив с полем;
3. POINT a[4], b[4], nextFigure[4] – переменные, хранящие координаты фигур;
4. char color, colorNext, type, typeNext, rot – переменные, хранящие цвет, тип и ориентацию фигур;
5. int figures[7][4] = {{1, 3, 5, 7}, {2, 4, 5, 7}, {3, 5, 4, 6}, {3, 5, 4, 7}, {2, 3, 5, 7}, {3, 5, 7, 6},{2, 3, 4, 5}} – массив с координатами стандартных фигур;
6. int score – переменная для хранения текущего количества очков;
7. int line – переменная для хранения количества очищенных линий;
8. SCORE *_score – указатель на объект класса SCORE;
9. int i, j - вспомогательные переменные;
10. bool status – переменная для определения состояния игры (проиграл\не проиграл).

Методы класса:

1. LOGICS_TETRIS() – конструктор класса;
2. ~LOGICS_TETRIS() – деструктор класса;
3. bool check() – метод для проверки возможности расположение фигуры;
4. void newFigure() – метод для генерации новой фигуры;
5. void setFigure() – метод для установка фигуры на поле;
6. void moveDownFigure() – метод для сдвига фигуры вниз;
7. void moveRightFigure() – метод для сдвига фигуры вправо;
8. void moveLeftFigure() – метод для сдвига фигуры влево;
9. void rotateFigure() – метод для поворота фигуры;
10. int **getField() – метод для получения указателя на поле;
11. POINT getSize() – метод для получения размера поля;

12. POINT *getFigure() – метод для получения указателя на массив с координатами фигуры;
13. char getColor() – метод для получения цвета фигуры;
14. POINT *getNextFigure() – метод для получение указателя на массив с координатами следующей фигуры;
15. char getNextColor() – метод для получения цвета следующей фигуры;
16. char getNextType() – метод для получения типа следующей фигуры;
17. int getScore() – метод для получения значения рекорда;
18. int getLine() – метод для получения количества удаленных линий;
19. bool getStatus() – метод для получения статуса игры (проиграл/не проиграл);
20. void clearGame() – метод для обнуление игры;
21. void setPointerScore(SCORE*) – метод для связки объекта с рекордами и текущего класса.

2.4 Класс WINDOW_GAME

Это класс, который наследуется от класса WINDOW. Этот класс отвечает за окно 1. В нем есть все необходимое для отрисовки игры и пунктов “Help” и “About author”. Этот класс содержит указатель на класс TETRIS_LOGICS. Это нужно для отрисовки игры.

Поля класса:

1. LOGICS_TETRIS *game – указатель на “логику” игры;
2. SDL_Surface *tile[8] – поверхности с текстурами кубиков;
3. SDL_Surface *background – поверхность с текстурой фона;
4. SDL_Surface *frame – поверхность с текстурой рамки для обвода кнопки;
5. char wayTiles[30] = ".\\images\\tiles*.bmp" – путь к картинке с кубиками;
6. char *wayBackground = ".\\images\\gameBackground.bmp" – путь к картинке с фоном;
7. char *wayFrame = ".\\images\\frame.bmp" – путь к картинке с рамкой;
8. int i, j, M, N – вспомогательные переменные;
9. char number[10] – вспомогательный массив для перевода числа в строку;
10. char type – переменная для определения, что будет выведено на экран.

Методы класса:

1. WINDOW_GAME() – конструктор класса;
2. virtual ~WINDOW_GAME() – деструктор класса;
3. void drawBackground() – метод для отрисовки фона;
4. void drawMenu() – метод для отрисовки меню;
5. void drawGame() – метод для отрисовки игры;
6. void drawEnd() – метод для отрисовки уведомления о конце игры;

7. void drawHelp() – метод для отрисовки информации об игре;
8. void drawAbout() – метод для отрисовки информации об авторе;
9. void start(char*, int, int, int, int, LOGICS_TETRIS*) – метод для "запуска" окна;
10. void drawAll() – метод для обновления изображения в окне;
11. void pressLeftButton() – метод для обработки нажатия левой кнопки мышки;
12. void setType(char) – метод для установки, что будет выведено на экран.

2.5 Класс WINDOW_SCORE

Это класс, который наследуется от класса WINDOW. Этот класс отвечает за окно 2. Он отрисовывает список рекордов. Этот класс содержит указатель на класс SCORE. Это нужно для того, чтобы иметь возможность выводить список рекордов.

Поля класса:

1. SCORE *score – указатель на объект для добавления рекордов;
2. SDL_Surface *background[2] – массив поверхностей с текстурой фонов;
3. SDL_Surface *frame – поверхность с рамкой для обвода кнопки;
4. char wayBackground[30] = ".\\images\\scoreBackground*.bmp" – путь к картинке с фоном;
5. char *wayFrame = ".\\images\\frame.bmp" – путь к картинке с рамкой;
6. int i, j – вспомогательные переменные;
7. bool flag – переменная для определения того, что будет выведено на экран;
8. char number[10] – вспомогательный массив для перевода числа в строку.

Методы класса:

1. WINDOW_SCORE() – конструктор класса;
2. virtual ~WINDOW_SCORE() – деструктор класса;
3. void start(char*, int, int, int, int, SCORE*) – метод для "запуска" окна;
4. void drawAll() – метод для обновления изображения в окне;
5. void setFlag(bool) – метод для установки, что будет выведено на экран;
6. void pressLeftButton() – метод для обработки нажатия левой кнопки мышки;
7. void drawScore() – метод для отрисовки рекордов;
8. void drawClear() – метод для отрисовки окна с вопросом об очистке.

3 Иерархия классов

На рисунке 1 изображена иерархия классов.

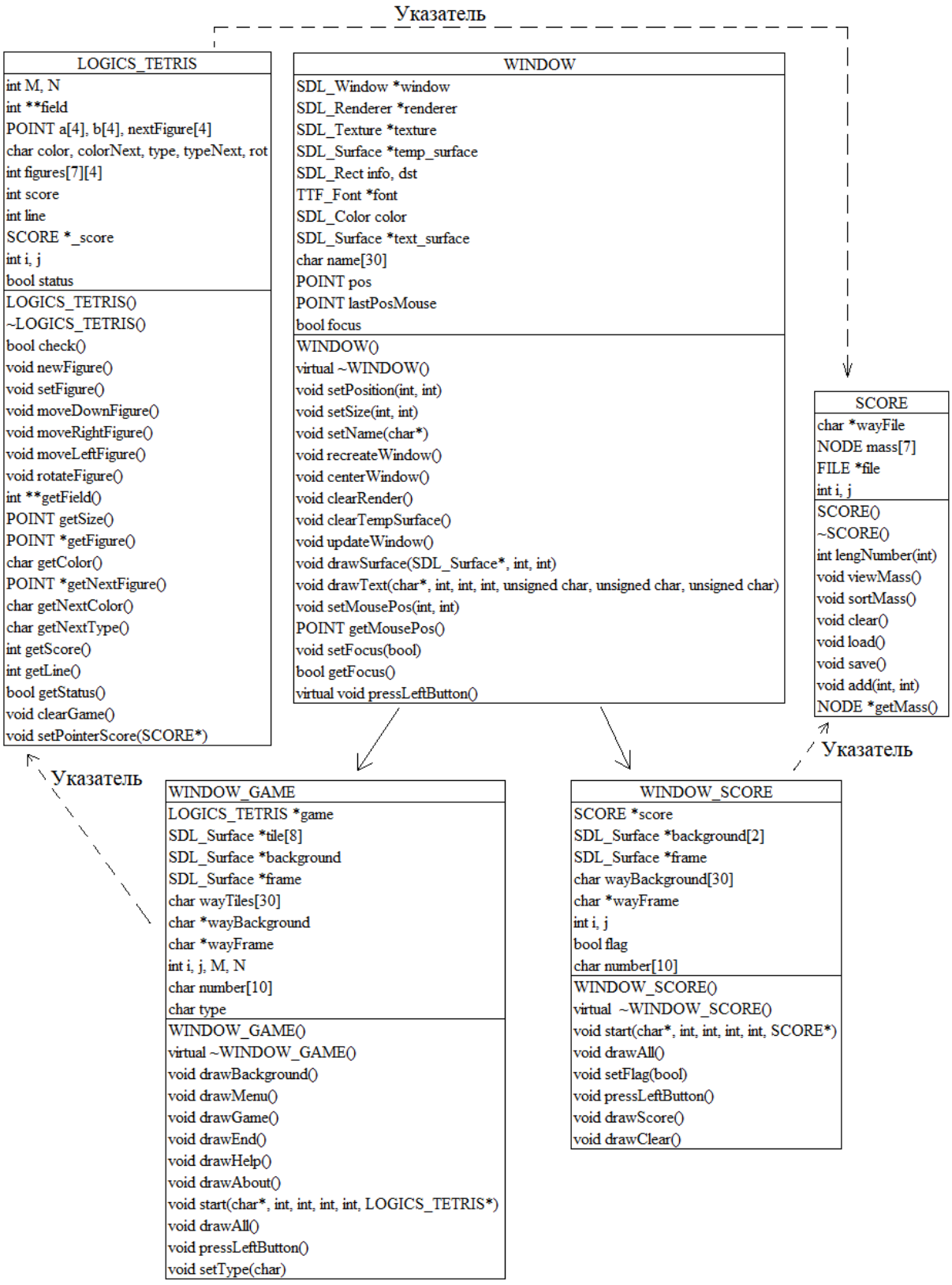

Рисунок 1 – Иерархия классов.

РЕЗУЛЬТАТ РАБОТЫ

На рисунке 2 представлены 2 окна приложения. Первое окно – окно с главным меню и игрой, а на втором окне – список рекордов.

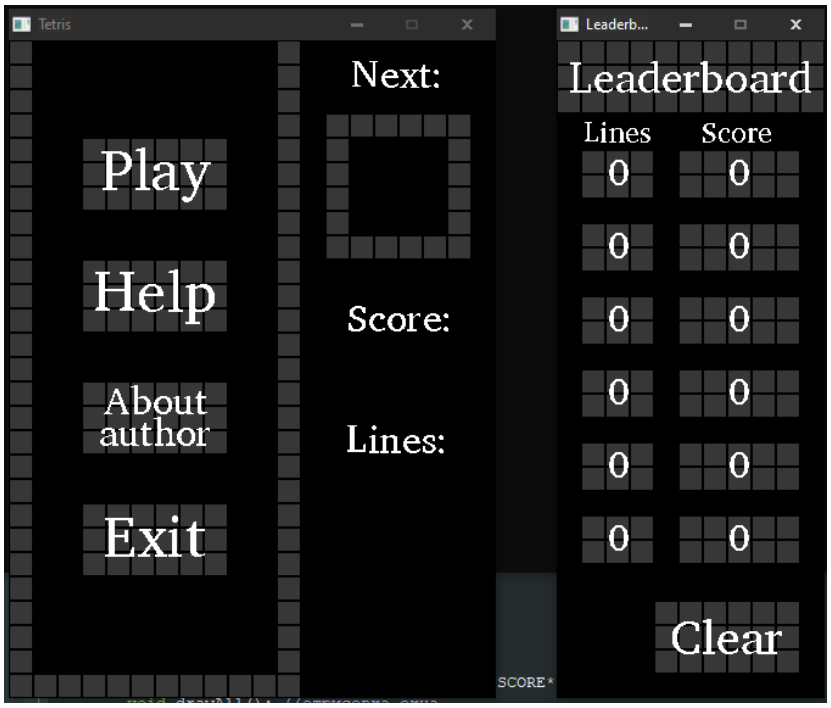

Рисунок 2 – Два окна.

Если нажать кнопку “Help” на первом окне, то появиться подсказка как играть в игру. Рисунок 3 это демонстрирует.

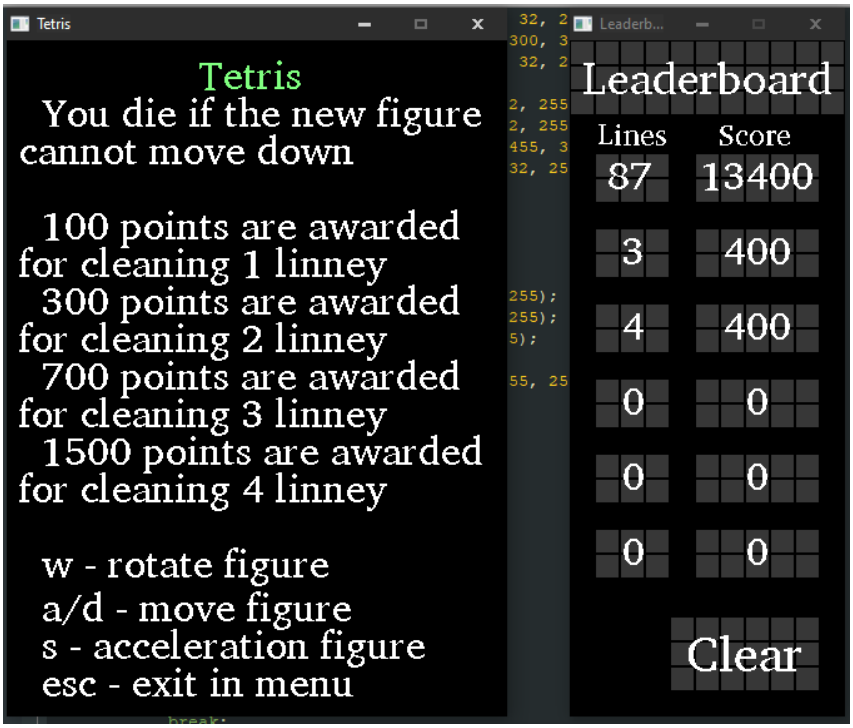

Рисунок 3 – Меню “Help”.

Если нажать кнопку “About author” на первом окне, то появиться информация об авторе работы. Рисунок 4 это демонстрирует.

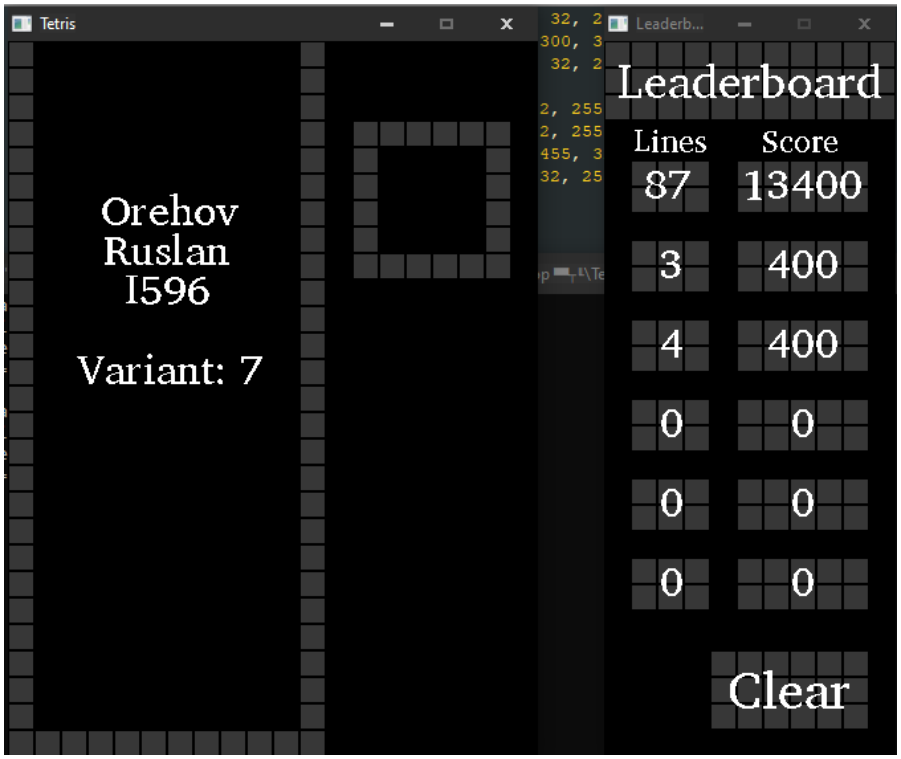

Рисунок 4 – Меню “About author”.

Если нажать кнопку “Exit” на первом окне, то приложение завершит свою работу. Так же произойдет, если нажать на крестик на любом из окон. Если нажать на кнопку “Play” на первом окне, то запуститься игра. Рисунок 5 демонстрирует это.

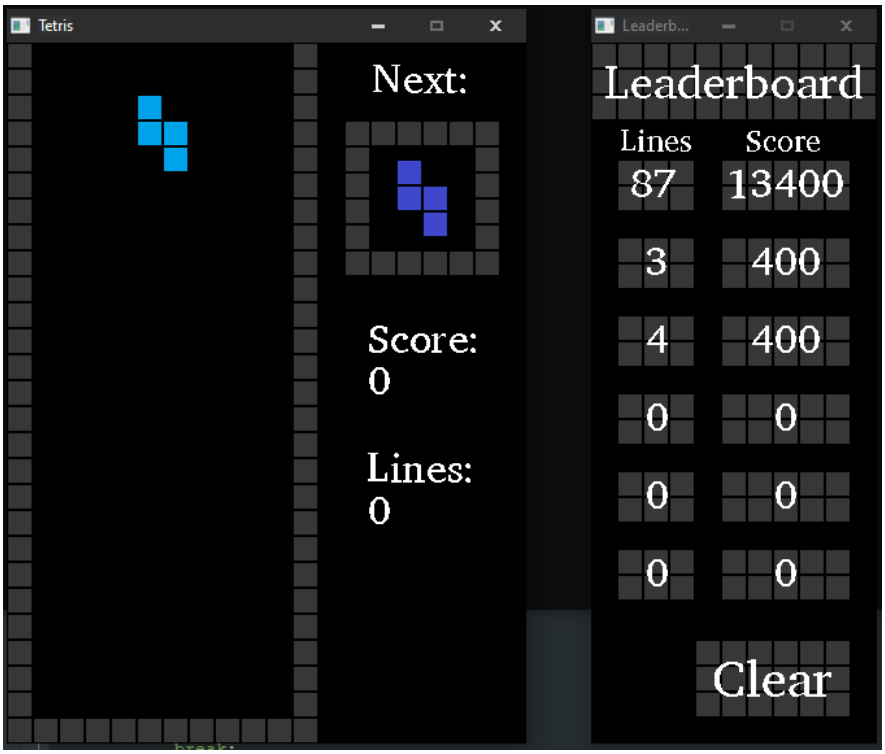

Рисунок 5 – Начало игры.

Фигура двигается вниз. Пользователь с помощью кнопок ‘a’ и ‘b’ может двигать фигуру влево и вправо. С помощью кнопки ‘w’ можно вращать фигуру, а с помощью кнопки ‘s’ можно ускорить движение фигуры вниз. Так же на экране отображается какая будет следующая фигура, количество очищенных линий и количество очков.

Если нажать кнопку “Clear” на втором окне, то высветится предупреждающее окно, спрашивающее, точно ли вы хотите удалить список рекордов, и две кнопки – “yes” и “no”. Рисунок 6 демонстрирует это. На рисунке 7 изображен пустой список рекордов.

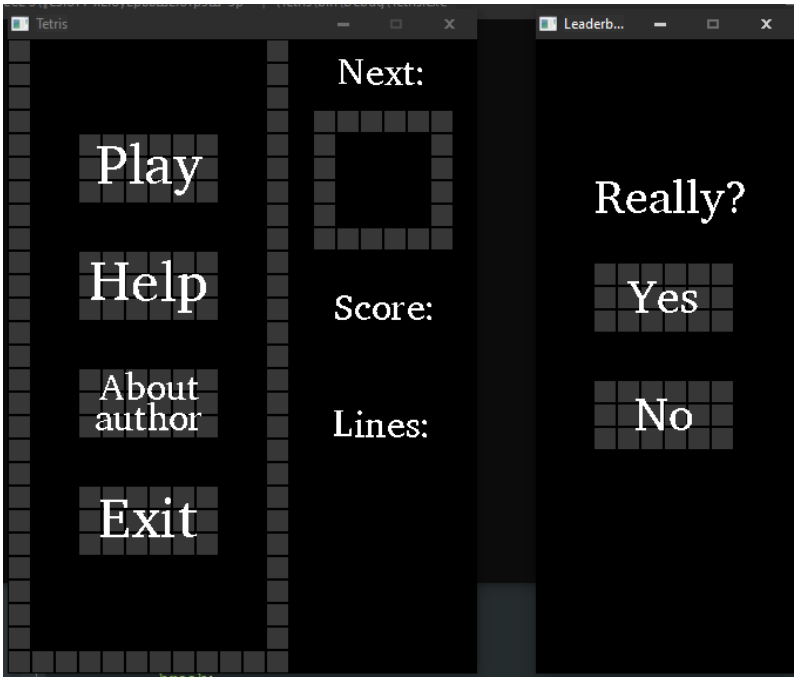

Рисунок 6 – удаление списка рекордов.

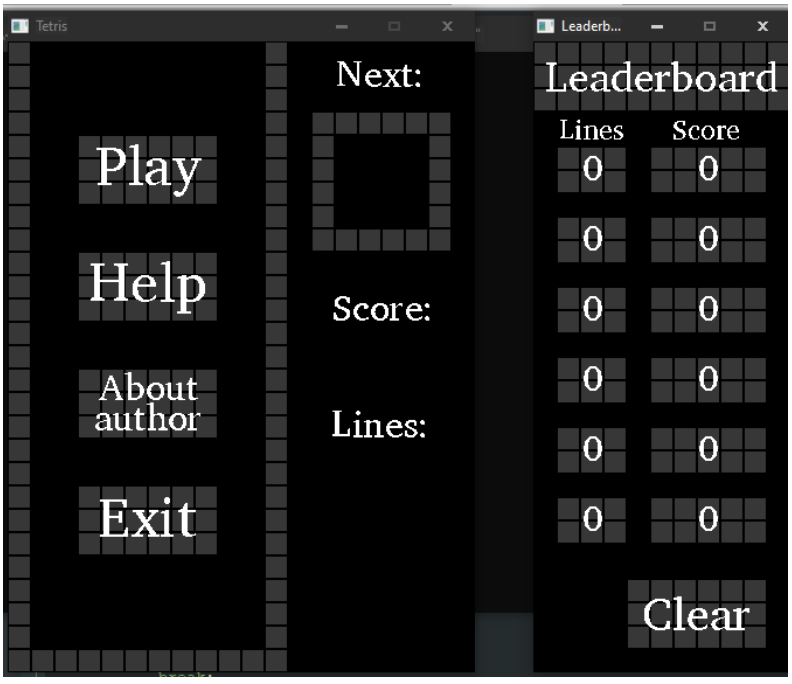

Рисунок 7 – очищенный список рекордов.

После проигрыша в игре количество очков и очищенных линий автоматически идет в список рекордов. Если во время игры нажать кнопку ‘esc’, то пользователь выйдет в главное меню (игра при этом не сохраняется).

ЗАКЛЮЧЕНИЕ

С помощью SDL2 удалось реализовать графический интерфейс игры тетрис, а с помощью навыков объектно-ориентированного программирования – гибкость кода и простую возможность дальнейшего расширения функционала.

Благодаря этой работе была создана игра тетрис на C/C++ с использованием классов, значит задача, поставленная перед этой работой - решена.

Также, в ходе создания игры были получены ценные навыки проектирование графических приложений, который будут несомненно полезны в дальнейшей профессиональной деятельности.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация библиотеки SDL [Электронный ресурс]. URL: https://wiki.libsdl.org/ (дата обращения 28.01.2021).

2. Документация к SDL_ttf [Электронный ресурс]. URL: https://www.libsdl.org/projects/SDL_ttf/docs/ (дата обращения: 28.01.2021).

3. Правила тетриса [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Тетрис (дата обращения 22.01.2021).

4. О.В. Арипова, О.А. Палехова, А.Н. Гущин. Программирование на языке высокого уровня. Лабораторный практикум. Санкт-Петербург, 2014, -94с.

5. О.А. Палехова, Основы программирования на языке С. Практикум. Санкт-Петербург, 2016, -95с.

ПРИЛОЖЕНИЕ А

Приложение содержит в себе исходные файлы проекта.

Имя файла: «Приложение.zip».

Для разработки программы используется среда программирования Code::Blocks версии 17.12.

Для написания программы подключены основная библиотека SDL 2 и дополнительная библиотека для работы с текстом SDL2_ttf.

Проект включает в себя файлы с текстом программы:

1. «main.cpp» – главный файл проекта. Объединяет все классы и SDL между собой.

2. Файлы «WINDOW.h» и «WINDOW.cpp» содержат класс «WINDOW».

3. Файлы «SCORE.h» и «SCORE.cpp» содержат класс «SCORE».

4. Файлы «LOGICS_TETRIS.h» и «LOGICS_TETRIS.cpp» содержат класс «LOGICS_TETRIS».

5. Файлы «WINDOW_GAME.h» и «WINDOW_GAME.cpp» содержат класс «WINDOW_GAME».

6. Файлы «WINDOW_SCORE.h» и «WINDOW_SCORE.cpp» содержат класс «WINDOW_SCORE».

7. Файл «POINT.h» содержит структуру POINT – вспомогательная структура с двумя переменными типа int, для хранения координат какого-то объекта.

7. Файл «gameBackground.bmp» - фон для окна 1.

8. Файл «scoreBackground1.bmp» - фон 1 для окна 2.

9. Файл «scoreBackground2.bmp» - фон 2 для окна 2.

10. Файл «tiles*», где * - число от 0 до 7 – текстура кубиков для отрисовки фигур.

11. Файл «score.bin» содержит информацию о рекордах.

12. Вспомогательные файлы для работы SDL и итогового приложения:

    12.1 «libfreetype-6.dll».

    12.2 «SDL2.dll».

    12.4 «SDL2_ttf.dll».

    12.5 «zlib.dll».

    12.6 « CharisSILR.ttf».