

Министерство науки и высшего образования
Пензенский государственный университет Кафедра
"Вычислительная техника"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу "Программирование"
на тему "Арканоид"

Выполнил:

Студент группы 24BBB2

Свинухов Н. Д.

Принял:

к.т.н., доцент Токарев А. Н.

уловн.
19.12.25
фс

Пенза 2025

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ



М.А. Митрохин

« ____ » ____ 20 __ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Виссариону Николаю Александровичу Группа 24ВВБЗ

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

*Управление действиями персонажем с помощью
клавиатуры
Система набора очков
Изменение цветовой палитры
Игра фрилансер*

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

1. Расчетная часть

2. Графическая часть

3. Экспериментальная часть

Срок выполнения проекта по разделам

Дата выдачи задания "27" декабря 2025

Дата защиты проекта “ ”

Руководитель _____

Задание получил " 26 " февраля 2025г.

Студент Свиридов Николай Дмитриевич *(подпись)*

Оглавление

Введение.....	4
1 Постановка задачи.....	4
2 Выбор решения.....	5
4 Отладка и тестирование.....	5
5 Описание программы.....	6
5.1 Разработка приложения Игра «Arkanoid».exe	7
6 Руководство пользователя	9
Заключение	11
Список используемых источников	12
Листинг программы	13

Введение

Игра “Арканойд” — это классика, известная многим поколениям игроков по всему миру. На сегодняшний день существует множество её компьютерных реализаций, каждая из которых имеет свои особенности и подходы к построению игрового процесса. Цель данной курсовой работы — разработка своей версии игры Arkanoïd на языке программирования Си. Тема остаётся актуальной, поскольку разработка простых, но увлекательных игр — важный шаг в освоении принципов гейм дизайна и алгоритмизации.

Постановка задачи

Игра “Арканойд” — это классика, известная многим поколениям игроков по всему миру. На сегодняшний день существует множество её компьютерных реализаций, каждая из которых имеет свои особенности и подходы к построению игрового процесса. Цель данной курсовой работы — разработка своей версии игры Arkanoïd на языке программирования Си. Тема остаётся актуальной, поскольку разработка простых, но увлекательных игр — важный шаг в освоении принципов гейм дизайна и алгоритмизации.

Использование ассемблерных вставок. В целях демонстрации взаимодействия языков высокого и низкого уровня в программе используется ассемблерная вставка в функции `saveMaxScore()`. С её помощью выполняется простая манипуляция с данными (инкремент значения, если оно равно нулю) перед сохранением в файл. Это позволяет оптимизировать выполнение критичных по времени операций и глубже понять работу процессора с данными.

Много модульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы.

Использование сложных типов данных — массивов, структур, файлов. Это необходимо для более простой и интуитивной обработки данных в коде программы.

Режим работы видеосистемы — текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации.

Устройство ввода-вывода — клавиатура. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы.

Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

Заставка необходима для того, чтобы пользователь, запустивший программу, смог получить достаточную информацию о ней.

Выбор решения

Разработанная программа состоит из 5 модулей:

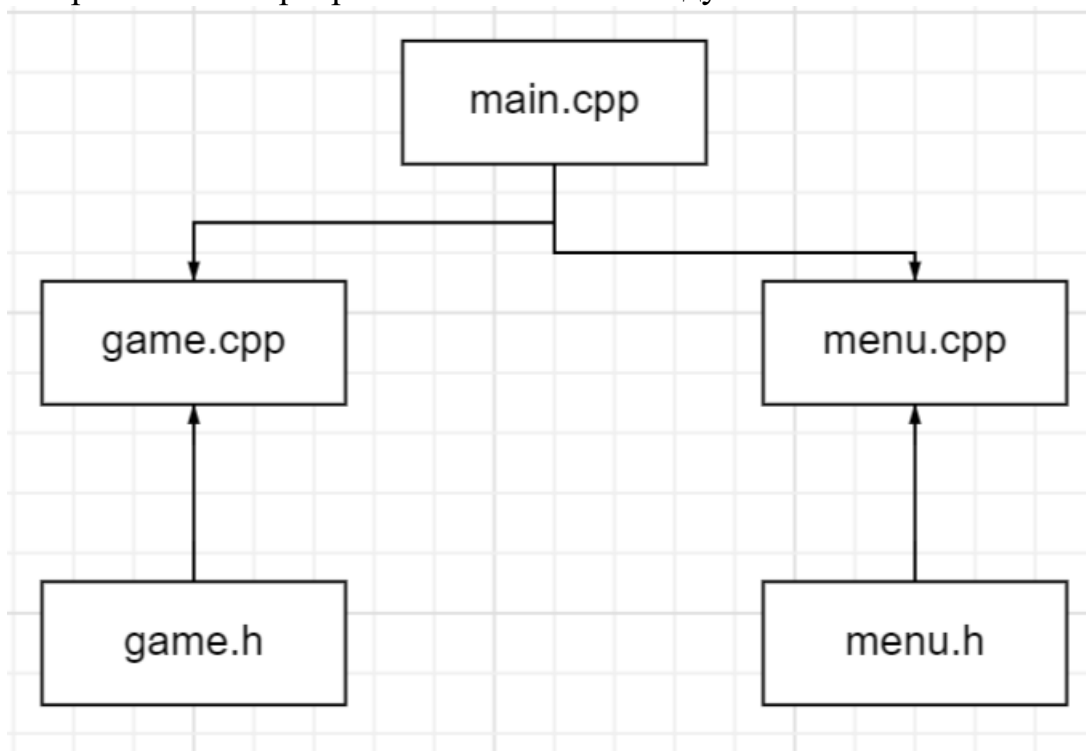


Рисунок 1 – Основные Модули

Основные модули и их функции

1. main.cpp:

- Основной модуль программы, содержащий точку входа main.
- Обрабатывает отображение главного меню и меню выбора сложности.
- Запускает основной игровой цикл, управляет отрисовкой и обновлением состояния игры.

2. game.h:

- Заголовочный файл, содержащий объявления констант, глобальных переменных и функций, используемых в игре.
- Позволяет разделить код на несколько файлов и управлять зависимостями между ними.

3. game.cpp:

- Реализует основные функции игры, такие как отрисовка элементов, обновление позиции мяча, проверка коллизий и обработка ввода.
- Содержит глобальные переменные, необходимые для функционирования игры, такие как положение платформы, позиция мяча и скорость игры.

Функции:

- initializeGame(): Инициализирует игру.
- startGameLoop(): Главный цикл игры. Отрисовывает игровое поле, обрабатывает ввод, обновляет состояние мяча и проверяет таймер.
- drawBall(): Отрисовывает мяч.
- clearScreen(): Очищает экран, устанавливая курсор в начало консоли (без миганий).
- drawBoard(): Отрисовывает игровое поле.
- updateBall(): Обновляет позицию мяча и проверяет коллизии.
- handleInput(): Обрабатывает ввод пользователя.
- saveMaxScore(): Сохраняет рекорд игрока в бинарный и текстовый файлы. Перед сохранением выполняется ассемблерная вставка, которая обрабатывает значение счёта: если счёт равен нулю, он инкрементируется.
- loadMaxScore(): Загружает максимальный счёт из файла.
- showGameOver(bool win): Показывает сообщение о проигрыше или победе, отображает счёт, предлагает перезапуск (R) или выход (Esc).

4. **menu.h:** ➤ setColor(WORD color): Устанавливает цвет текста в консоли (используется для визуального оформления).

- initializeBallPosition(): Инициализирует позицию мяча.
- Заголовочный файл, содержащий объявления функций для отображения меню.
- clearScreen(): Очищает экран.
- drawPaddle(): Отрисовывает платформу.
- drawBall(): Отрисовывает мяч.
- drawBrick(int x, int y): Отрисовывает кирпич на заданной позиции.
- drawScore(): Отображает текущий и максимальный счёт.
- drawBoard(): Отрисовывает игровое поле.
- updateBallPosition(): Обновляет позицию мяча и проверяет коллизии.
- checkPlatformCollision(): Проверяет коллизии мяча с платформами.
- handleInput(): Обрабатывает ввод пользователя.
- saveMaxScore(): Сохраняет максимальный счёт в файл.
- loadMaxScore(): Загружает максимальный счёт из файла.
- initializeAdditionalPlatforms(): Инициализирует дополнительные платформы для сложных уровней.
- initializeAdditionalPlatforms2(): Инициализирует дополнительные платформы для среднего уровня сложности.
- initializeBallPosition(): Инициализирует позицию мяча.

5. `menu.cpp`:

- Реализует функции для отображения главного меню и меню выбора сложности.
- Функции:
- `showMenu()`: Отображает главное меню.
- `showDifficultyMenu()`: Отображает меню выбора сложности.

Описание разработки программы

Для реализации игры был выбран язык Си, поскольку он обеспечивает необходимую гибкость и позволяет глубже понять принципы системного программирования. В качестве среды разработки использовалась Microsoft Visual Studio 2022. Интерфейс реализован в виде текстового меню — это один из самых простых и при этом эффективных способов взаимодействия с пользователем.

Управление осуществляется с помощью цифровых клавиш, что позволяет интуитивно перемещаться по пунктам меню даже тем, кто впервые запускает игру. Такая структура делает работу с программой удобной и понятной с первых секунд.

При запуске пользователю демонстрируется приветственная заставка. После нажатия клавиши Enter отображается экран выбора уровня сложности. Переключение между уровнями реализовано через нажатие клавиш 1, 2 или 3.

Описание программы

Функция `main.c` является основным модулем программы. При запуске программы происходит вывод заставки-приветствия, после чего пользователю необходимо выбрать пункт меню для начала игры. Выход из меню осуществляется после выбора пользователем нужного пункта.

Программа представляет собой простую текстовую игру в стиле арканоида. Основные элементы игры включают управление платформой, отрисовку игрового поля, обработку коллизий мяча и платформ, а также меню для выбора уровня сложности.

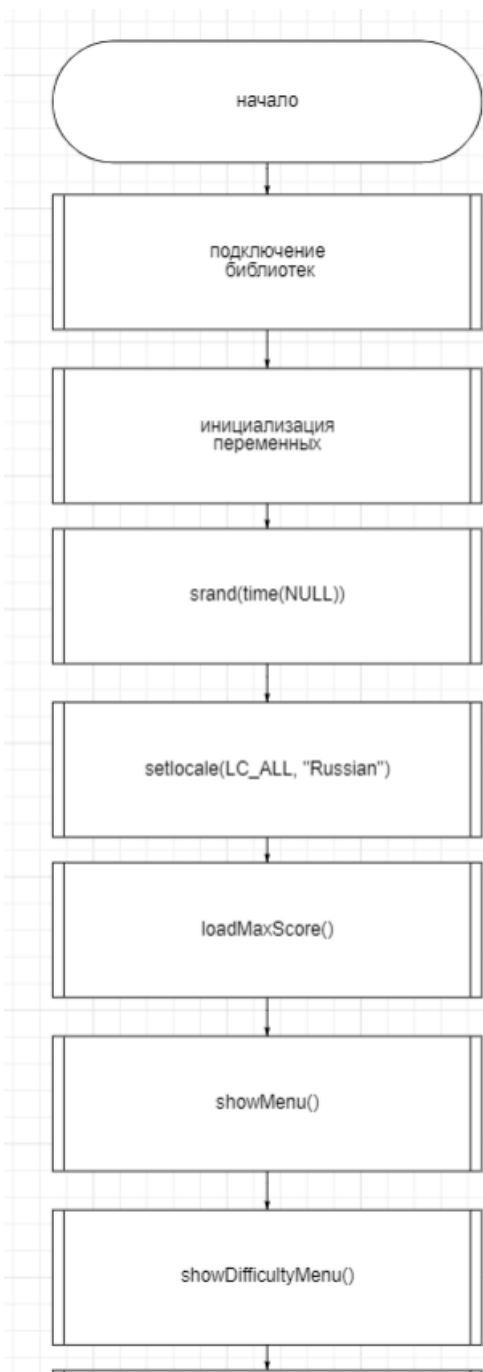


Рисунок 2

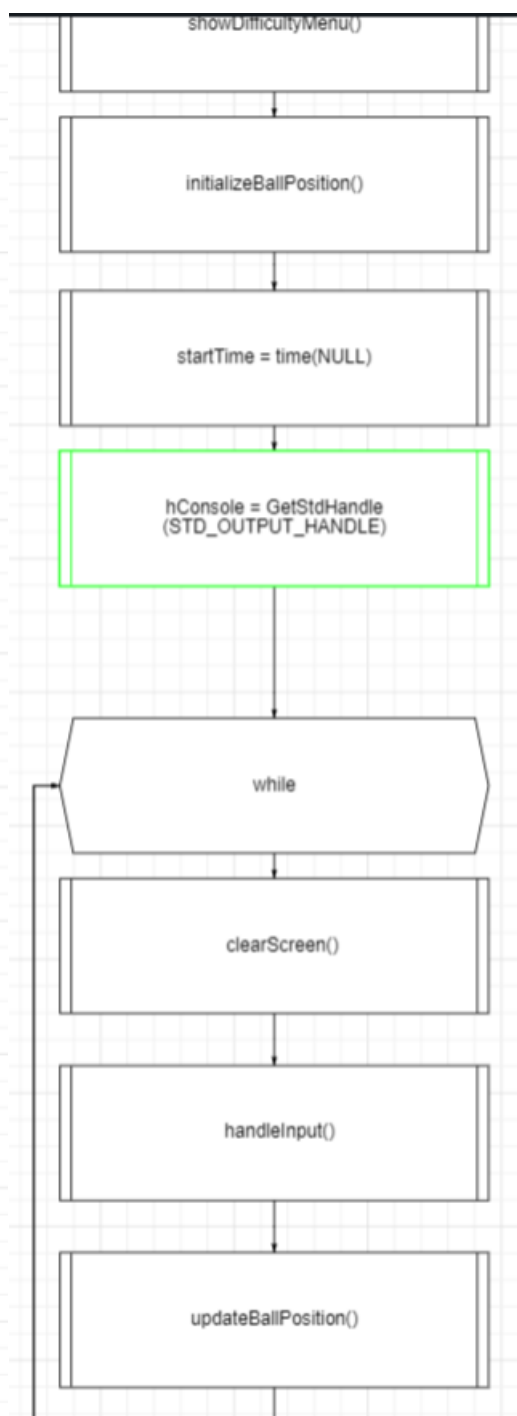


Рисунок 3

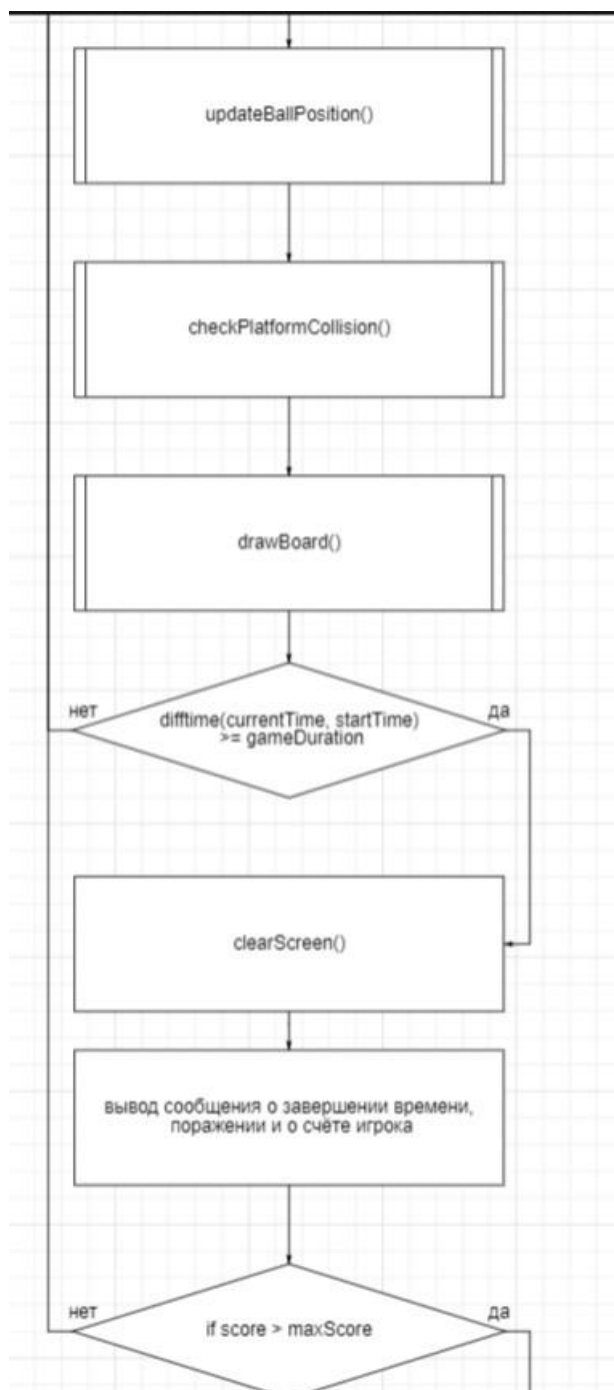


Рисунок 4



Рисунок 5

Рисунки 2-5 – Main.cpp

Game.h



Рисунок 6 – Game.h

Menu.h



Рисунок 7 – Menu.h

Menu.c



Рисунок 8 – ShowMenu()

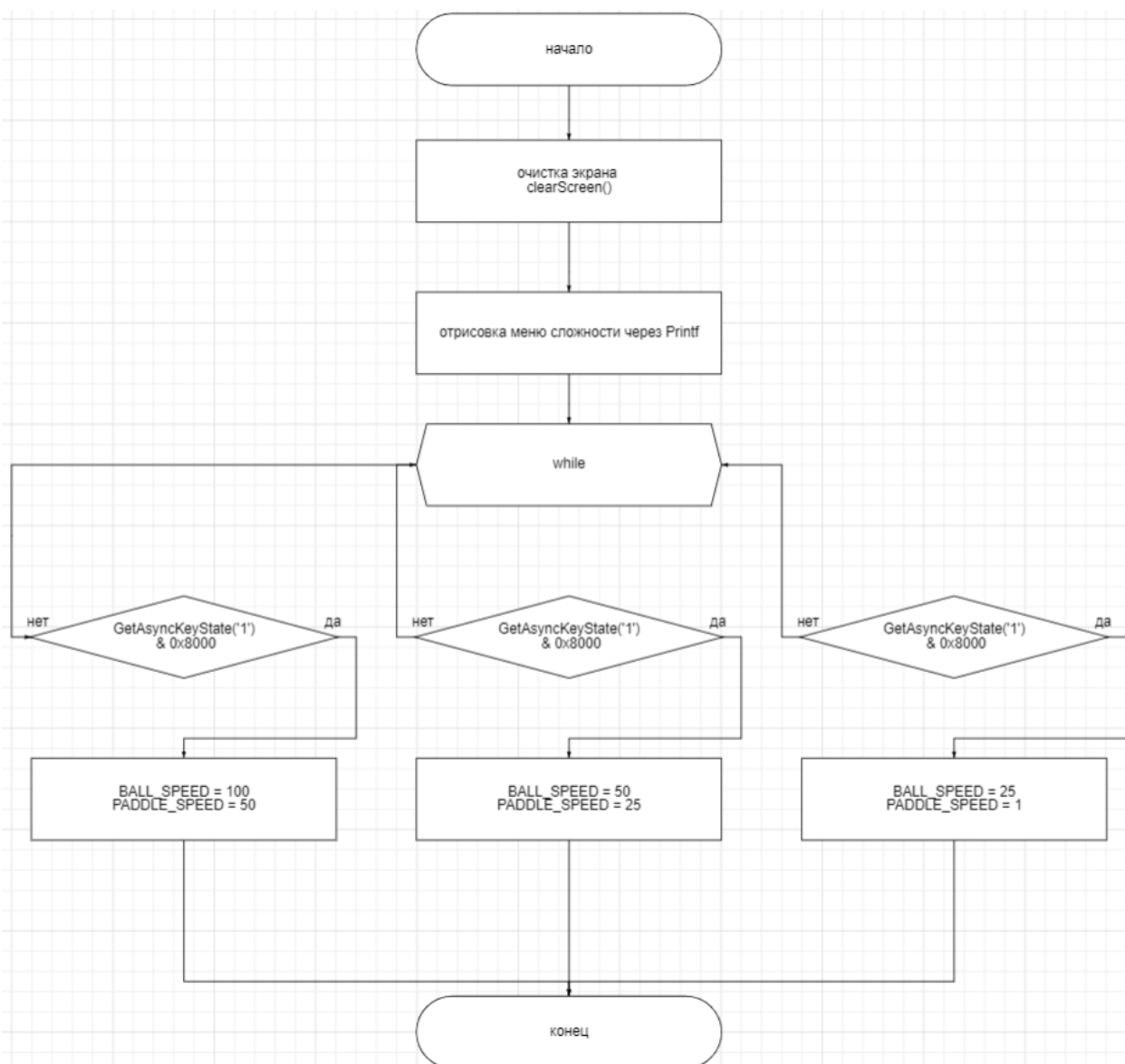


Рисунок 9 – ShowDifficultyMenu()

6. Руководство пользователя

Программа `arkanoid.cpp` предназначена для создания видеоигры на языке C. Программа имеет консольный интерфейс и несколько уровней сложности, отличающиеся друг от друга количеством новых слоёв платформ, а также увеличенной скоростью игры. При этом, каждый новый ваш рекорд будет записываться в файл.

После запуска будет выведена приветственная заставка.

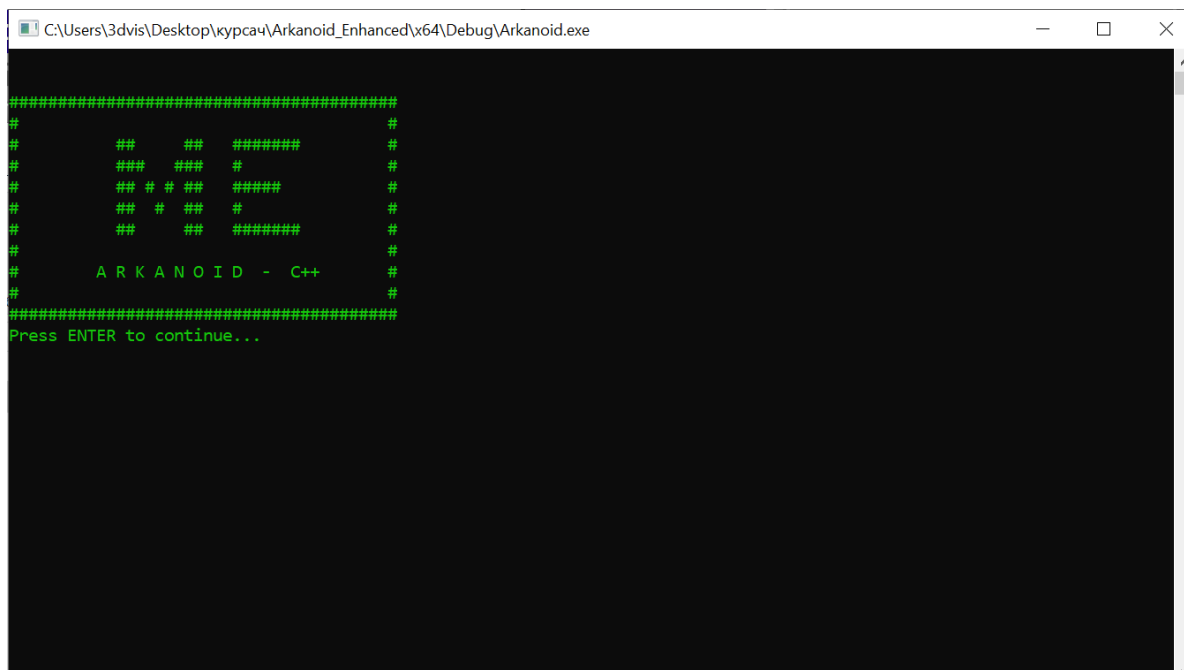
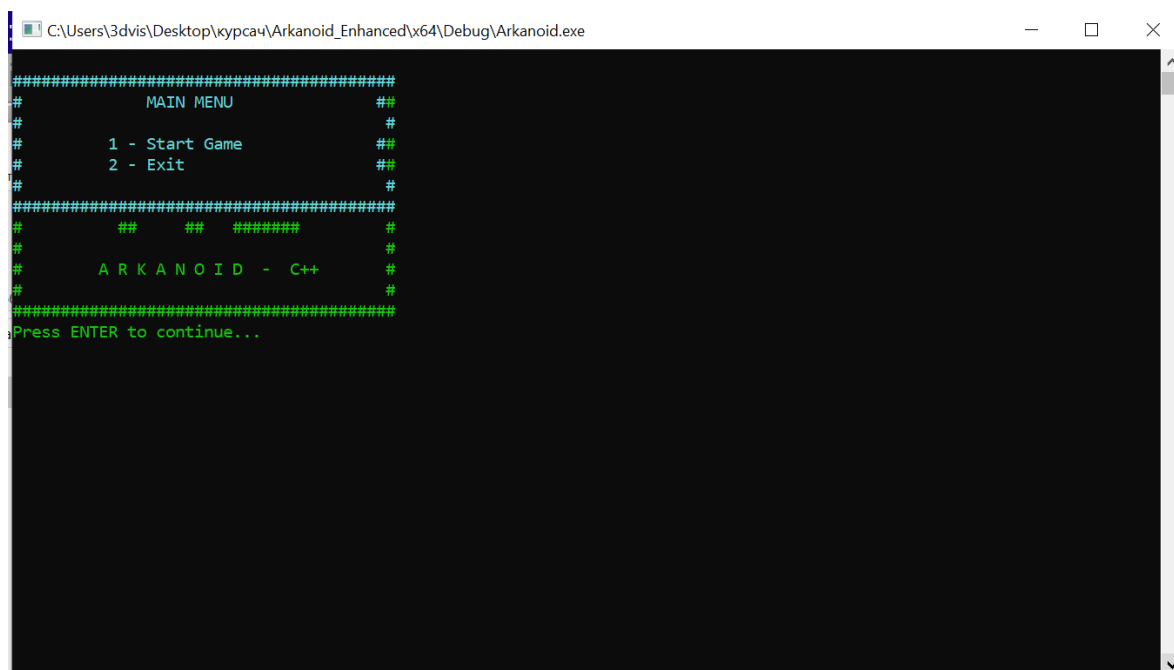


Рисунок 10 – Главное Меню игры

Выбор уровня сложности



Рисунок 11 – Меню выбора сложности игры

После того как игрок нажал на Enter и выбрал уровень сложность, запускается игра.

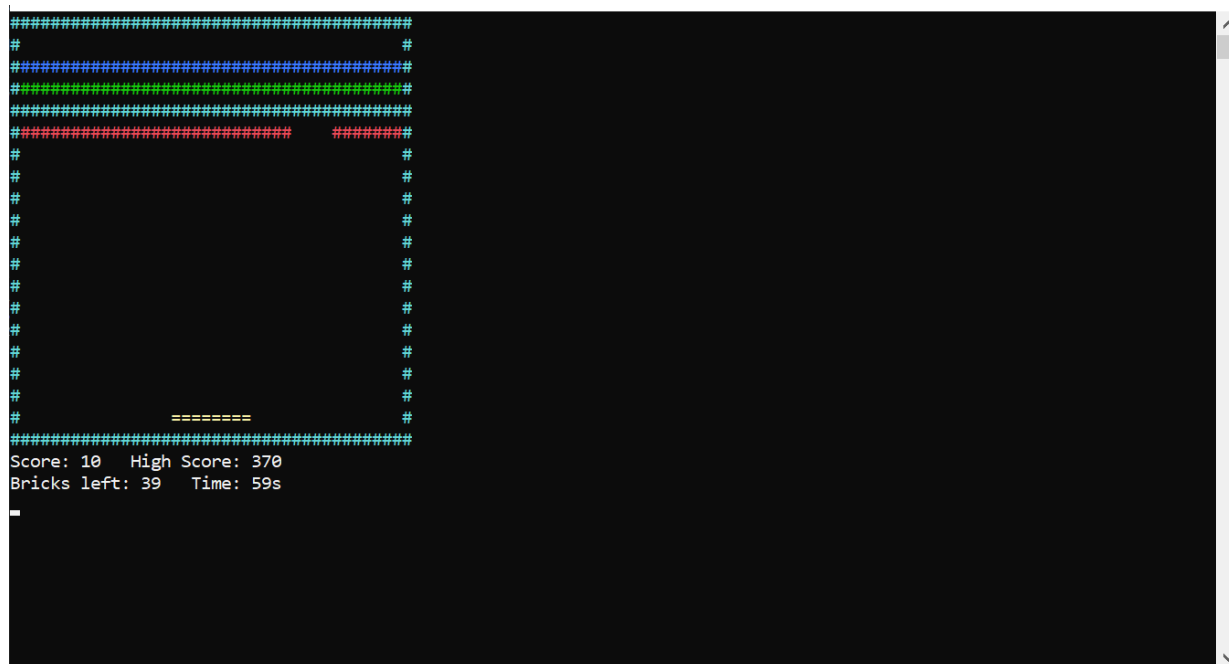


Рисунок 12 – игровой процесс

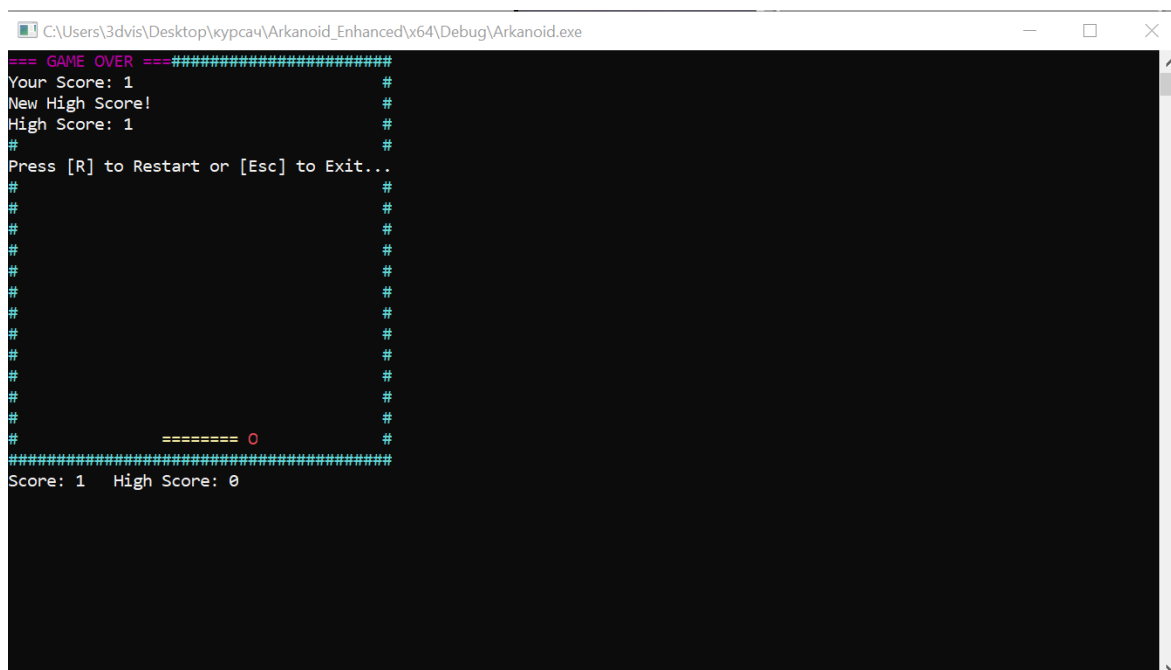


Рисунок 13 – окно завершения игры

По правилам оригинальной игры Arkanoid игрок контролирует небольшую платформу-ракетку, которую можно передвигать горизонтально от одной стенки до другой с помощью клавиш A и D, подставляя её под шарик и предотвращая его падение вниз. Удар шарика по кирпичу приводит к его разрушению. После того как количество кирпичиков, достаточное для того, чтобы мяч попал по противоположному краю поля, разрушается и мяч долетает до противоположного края поля, игра завершается. При этом существует таймер, и если игрок по его истечению, не дойдет до дальнего края поля, то проиграет

Заключение

При выполнении курсовой работы были получены навыки разработки многомодульных программ. Были освоены способы обработки событий при вводе с клавиатуры и мыши. Усвоены базовые навыки программирования на языке C/ C++ и Ассемблер. Изучены основные возможности среды программирования Visual Studio 2022. Получены навыки тестирования и отладки программ.

В рамках выполнения курсовой работы была написана программа, необходимая для разработки игры жанра арканойд. Программа предоставляет небольшой, но достаточный для использования список возможностей. Не требует большого количества системных ресурсов и обеспечивает быструю обработку строки выражений.

В дальнейшем программу можно улучшить, путём добавления графического интерфейса и игрового процесса

Список используемых источников

1. Язык C++ и программирование на нём: учебное пособие / В.И. Рейзлин ; Томский политехнический университет. – 3-е изд.,

перераб. – Томск : Изд-во Томского политехнического университета, 2021. – 208 с.

2. Ассемблер в примерах и задачах: учебное пособие / О.В. Гаркуша, Н.Ю. Добровольская; Министерство науки и высшего образования Российской Федерации, Кубанский государственный университет. – Краснодар: Кубанский гос.

ун-т, 2022. – 134 с. – 500 экз.

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.

4. Свободная энциклопедия Wikipedia

Листинг программы

Main.cpp

```
#include <iostream>
#include <locale.h>
#include <windows.h>
#include "game.h"
#include "menu.h"

int main() {
    setlocale(LC_ALL, "English");
    loadMaxScore();    // Load high score
    showSplash();      // Welcome splash
    showMainMenu();    // Main menu
    showDifficultyMenu(); // Difficulty selection

    initializeGame();  // Setup game
    startGameLoop();   // Start game loop

    return 0;

}
```

Game.h

```
#ifndef GAME_H
#define GAME_H

#include <windows.h>

#define WIDTH 40
#define HEIGHT 20
#define PADDLE_LENGTH 8
#define BRICK_ROWS 4
#define BRICK_COLS 10
#define BRICK_WIDTH 4

extern int paddleX, ballX, ballY, ballDirX, ballDirY;
extern int score, maxScore;
extern int gameDuration;
extern int bricks[BRICK_ROWS][BRICK_COLS];
extern time_t startTime; // <-- ДОБАВИТЬ ЭТУ СТРОКУ
extern HANDLE hConsole;

void initializeGame();
void drawBoard();
void clearScreen();
void updateBall();
void handleInput();
void saveMaxScore();
void loadMaxScore();
void showGameOver(bool win);
void startGameLoop();
void setColor(WORD color);
void initializeBricks();
void checkBrickCollision();
int countBricks();

#endif
```

Game.Cpp

```
#include "game.h"
```

```

#include <iostream>
#include <ctime>
#include <conio.h>
#include <fstream>

using namespace std;

int paddleX = WIDTH / 2 - PADDLE_LENGTH / 2;
int ballX = WIDTH / 2;
int ballY = HEIGHT - 5;
int ballDirX = 1, ballDirY = -1;
int score = 0, maxScore = 0;
int gameDuration = 30;
int bricks[BRICK_ROWS][BRICK_COLS];
time_t startTime; // <-- ОБЪЯВЛЕНИЕ ПЕРЕМЕННОЙ
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

void setColor(WORD color) {
    SetConsoleTextAttribute(hConsole, color);
}

void clearScreen() {
    COORD coord = { 0, 0 };
    SetConsoleCursorPosition(hConsole, coord);
}

void initializeBricks() {
    for (int i = 0; i < BRICK_ROWS; ++i) {
        for (int j = 0; j < BRICK_COLS; ++j) {
            bricks[i][j] = 1; // 1 = кирпич существует, 0 = разрушен
        }
    }
}

int countBricks() {
    int count = 0;
    for (int i = 0; i < BRICK_ROWS; ++i) {
        for (int j = 0; j < BRICK_COLS; ++j) {
            count += bricks[i][j];
        }
    }
    return count;
}

void drawBoard() {
    setColor(11); // Цвет границ

    for (int y = 0; y < HEIGHT; ++y) {
        for (int x = 0; x < WIDTH; ++x) {
            // Границы поля
            if (y == 0 || y == HEIGHT - 1 || x == 0 || x == WIDTH - 1) {
                cout << "#";
            }
            // Область кирпичей (ряды 2-5)
            else if (y >= 2 && y < 2 + BRICK_ROWS) {
                int brickRow = y - 2;
                int brickCol = x / BRICK_WIDTH;

                if (brickRow >= 0 && brickRow < BRICK_ROWS &&
                    brickCol >= 0 && brickCol < BRICK_COLS &&
                    bricks[brickRow][brickCol] == 1 &&
                    x >= brickCol * BRICK_WIDTH &&
                    x < (brickCol + 1) * BRICK_WIDTH) {

                    // Разные цвета для разных рядов кирпичей

```

```

        setColor(brickRow + 9); // 9-12 разные цвета
        cout << "#";
        setColor(11);
        continue;
    }
    else {
        cout << " ";
    }
}
// Платформа
else if (y == HEIGHT - 2 && x >= paddleX && x < paddleX + PADDLE_LENGTH) {
    setColor(14); // Желтый
    cout << "=";
    setColor(11);
}
// Мяч
else if (x == ballX && y == ballY) {
    setColor(12); // Красный
    cout << "O";
    setColor(11);
}
// Пустое пространство
else {
    cout << " ";
}
}
cout << "\n";
}

// Отображение счета
setColor(15); // Белый

// Расчет оставшегося времени
time_t currentTime = time(0);
int remainingTime = gameDuration - (int)difftime(currentTime, startTime);
if (remainingTime < 0) remainingTime = 0;

cout << "Score: " << score << " High Score: " << maxScore << "\n";
cout << "Bricks left: " << countBricks() << " Time: " << remainingTime << "s\n";
}

void checkBrickCollision() {
    // Проверяем, находится ли мяч в области кирпичей (ряды 2-5)
    if (ballY >= 2 && ballY < 2 + BRICK_ROWS) {
        int brickRow = ballY - 2;
        int brickCol = ballX / BRICK_WIDTH;

        if (brickRow >= 0 && brickRow < BRICK_ROWS &&
            brickCol >= 0 && brickCol < BRICK_COLS &&
            bricks[brickRow][brickCol] == 1) {

            // Разрушаем кирпич
            bricks[brickRow][brickCol] = 0;
            score += 10;

            // Меняем направление мяча в зависимости от того, с какой стороны ударили
            // Простая логика: если мяч двигался вниз, отскакиваем вверх и наоборот
            ballDirY *= -1;

            // Можно добавить звук или эффекты
            return;
        }
    }
}
}

```

```

void updateBall() {
    ballX += ballDirX;
    ballY += ballDirY;

    // Столкновение с боковыми стенами
    if (ballX <= 1 || ballX >= WIDTH - 2) {
        ballDirX *= -1;
    }

    // Столкновение с потолком
    if (ballY <= 1) {
        ballDirY *= -1;
    }

    // Столкновение с платформой
    if (ballY == HEIGHT - 2 && ballX >= paddleX && ballX < paddleX + PADDLE_LENGTH) {
        ballDirY = -1; // Всегда отскакиваем вверх
        // Можно добавить изменение угла в зависимости от места попадания
        int hitPos = ballX - paddleX;
        if (hitPos < 2) ballDirX = -1;
        else if (hitPos > PADDLE_LENGTH - 3) ballDirX = 1;
        else if (hitPos < PADDLE_LENGTH / 2) ballDirX = -1;
        else ballDirX = 1;
    }

    // Проверка столкновений с кирпичами
    checkBrickCollision();

    // Проверка на проигрыш (мяч упал)
    if (ballY >= HEIGHT - 1) {
        showGameOver(false);
    }

    // Проверка на победу (все кирпичи разрушены)
    if (countBricks() == 0) {
        showGameOver(true);
    }
}

void handleInput() {
    if (GetAsyncKeyState('A') & 0x8000 && paddleX > 2) paddleX--;
    if (GetAsyncKeyState('D') & 0x8000 && paddleX + PADDLE_LENGTH < WIDTH - 2) paddleX++;
}

void initializeGame() {
    ballX = WIDTH / 2;
    ballY = HEIGHT - 5;
    paddleX = WIDTH / 2 - PADDLE_LENGTH / 2;
    score = 0;
    ballDirX = 1;
    ballDirY = -1;
    initializeBricks();
    startTime = time(0);
}

void saveMaxScore() {
    int scoreToSave = maxScore;

    __asm {
        mov eax, scoreToSave
        mov ebx, eax
        mov ecx, ebx
        add ecx, 1234
        sub ecx, 1234
        xor eax, eax
    }
}

```

```

    mov eax, scoreToSave
    test eax, eax
    jnz not_zero_label
    inc eax
    not_zero_label :
    mov scoreToSave, eax
}

ofstream file("score.bin", ios::binary);
if (file.is_open()) {
    file.write(reinterpret_cast<const char*>(&scoreToSave), sizeof(int));
    file.close();
}

ofstream txtFile("score.txt");
if (txtFile.is_open()) {
    txtFile << maxScore;
    txtFile.close();
}
}

void loadMaxScore() {
    FILE* binFile = nullptr;
    errno_t err = fopen_s(&binFile, "score.bin", "rb");
    if (err == 0 && binFile != nullptr) {
        size_t result = fread(&maxScore, sizeof(int), 1, binFile);
        fclose(binFile);
        if (result == 1) {
            return;
        }
    }
}

ifstream file("score.txt");
if (file.is_open()) {
    file >> maxScore;
    file.close();
}
else {
    maxScore = 0;
}
}

void showGameOver(bool win) {
    clearScreen();

    if (win) {
        setColor(10); // Зеленый
        cout << "=== YOU WIN! ===\n";
        cout << "All bricks destroyed!\n";
    }
    else {
        setColor(12); // Красный
        cout << "=== GAME OVER ===\n";
    }
}

setColor(15); // Белый
cout << "Your Score: " << score << "\n";

if (score > maxScore) {
    maxScore = score;
    saveMaxScore();
    setColor(14); // Желтый
    cout << "New High Score!\n";
}
}

```

```

setColor(15);
cout << "High Score: " << maxScore << "\n\n";
cout << "Press [R] to Restart or [Esc] to Exit...\n";

while (true) {
    if (GetAsyncKeyState('R') & 0x8000) {
        initializeGame();
        startGameLoop();
        break;
    }
    if (GetAsyncKeyState(VK_ESCAPE) & 0x8000) {
        exit(0);
    }
}

void startGameLoop() {
    startTime = time(0);
    while (true) {
        clearScreen();
        drawBoard();
        handleInput();
        updateBall();

        // Проверка времени (если время вышло)
        time_t currentTime = time(0);
        if (difftime(currentTime, startTime) >= gameDuration) {
            showGameOver(false);
            break;
        }

        // Проверка победы (все кирпичи разрушены)
        if (countBricks() == 0) {
            showGameOver(true);
            break;
        }

        Sleep(30);
    }
}

```

Menu.cpp

```

#include "menu.h"
#include "game.h"
#include <iostream>
#include <windows.h>

void showSplash() {
    setColor(10); // Green
    std::cout << R"(

#####
#           #
#  ##  ##  #####  #
#   ##  ##  #      #
#   ##  ##  #####  #
#   ##  ##  #      #
#   ##  ##  #####  #
#           #
#  A R K A N O I D  -  C++  #
#           #
#####
Press ENTER to continue...
)";
    while (!(GetAsyncKeyState(VK_RETURN) & 0x8000)) {}
}

```

```

    Sleep(300);
}

void showMainMenu() {
    clearScreen();
    setColor(11); // Cyan
    std::cout << R"(
#####
#      MAIN MENU      #
#                    #
#    1 - Start Game    #
#    2 - Exit          #
#                    #
#####
)";
    while (true) {
        if (GetAsyncKeyState('1') & 0x8000) break;
        if (GetAsyncKeyState('2') & 0x8000) exit(0);
    }
    Sleep(300);
}

void showDifficultyMenu() {
    clearScreen();
    setColor(14); // Yellow
    std::cout << R"(
#####
#    SELECT DIFFICULTY    #
#                    #
#    1 - Easy (60s)      #
#    2 - Medium (40s)    #
#    3 - Hard (25s)      #
#                    #
#####
)";
    while (true) {
        if (GetAsyncKeyState('1') & 0x8000) { gameDuration = 60; break; }
        if (GetAsyncKeyState('2') & 0x8000) { gameDuration = 40; break; }
        if (GetAsyncKeyState('3') & 0x8000) { gameDuration = 25; break; }
    }
    Sleep(300);
}

```

Menu.h

```
#ifndef MENU_H
#define MENU_H

void showSplash();
void showMainMenu();
void showDifficultyMenu();

#endif
```