Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Документация для приложения

«Змейка»

по дисциплине

«Введение в Информационные Технологии»

Подготовили: студенты группы БВТ1902

Ахмедов Х., Иракозе Д., Саввин Д.

Москва

2022

Оглавление

ОСНОВНАЯ ЧАСТЬ

О приложении.

В основе приложения лежит классическая игра «змейка», которая неустанно движется по игровому полю, собирая яблоки и увеличиваясь в размере, пока последний не станет проблемой. Количество съеденных яблок и будет являться конечным счетом игры. Игра немедленно закончится если змейка врежется в границы поля, либо же попытается съесть саму себя. Управление змейкой производится посредством клавиш со стрелками.

Возможности.

В игре предусмотрена возможность изменения сложности игры, то есть, скорости змейки. Для изменения этого параметра, необходимо из главного меню перейти во вкладку настроек, наверху которой расположен ползунок изменения скорости игры, чем выше его значение – тем выше скорость.

Также приложение хранит рекордный счет набранный пользователем, он отображается при каждом проигрыше. Также в меню настроек есть возможность обнулить рекорд.

В главном меню расположена кнопка «Помощь», при нажатии на которую пользователю будет выведено сообщение с кратким туториалом к игре.

Архитектура.

В ходе разработки архитектуры, были выделены следующие основные классы для реализации игры:

1) Point – Простейший класс точки в двумерном пространстве;
2) Apple – Класс описывающий яблоко, которое должна съесть змейка;
3) Field – Класс описывающий игровое поле;
4) Snake – Класс описывающий змейку;
5) Game – Класс объеденяющий, и обеспечивающий взаимодействие приведенных выше классов.

Из этих шаблонов была построена диаграмма классов, которую можно наблюдать на рисунке 1.

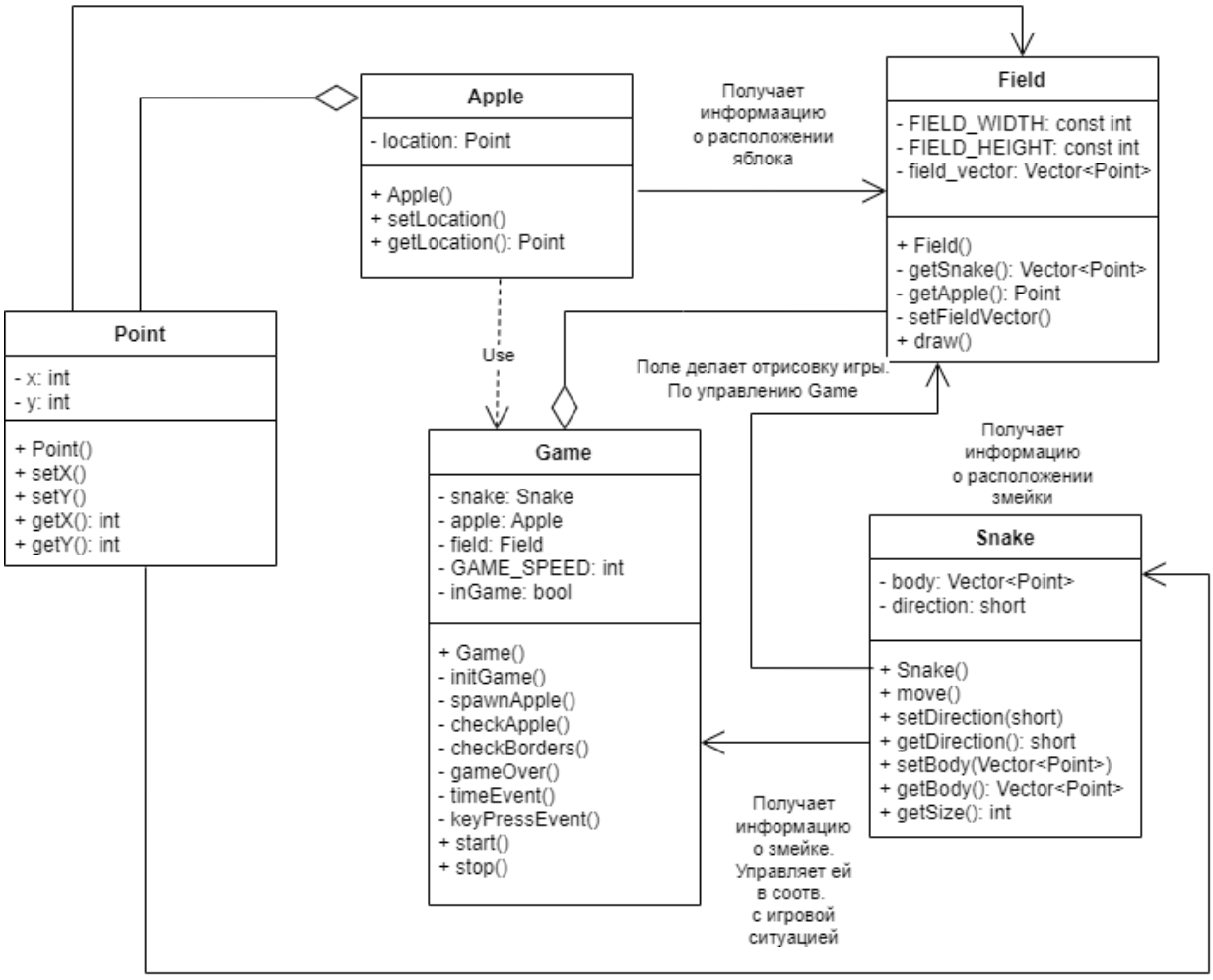

Рис. 1 – Диаграмма классов.

В конечной же реализации архитектура была упрощена, это связано с использованием фреймворка с его готовыми решениями. По итогу саму игру описывает один класс – game, он же в свою очередь используется в другом классе, описывающий окно приложения. Листинги кода вы можете изучить на странице 8.

Руководство пользователя.

После запуска программы “Snake”, перед вами должно открыться окно показанное на рисунке 2.

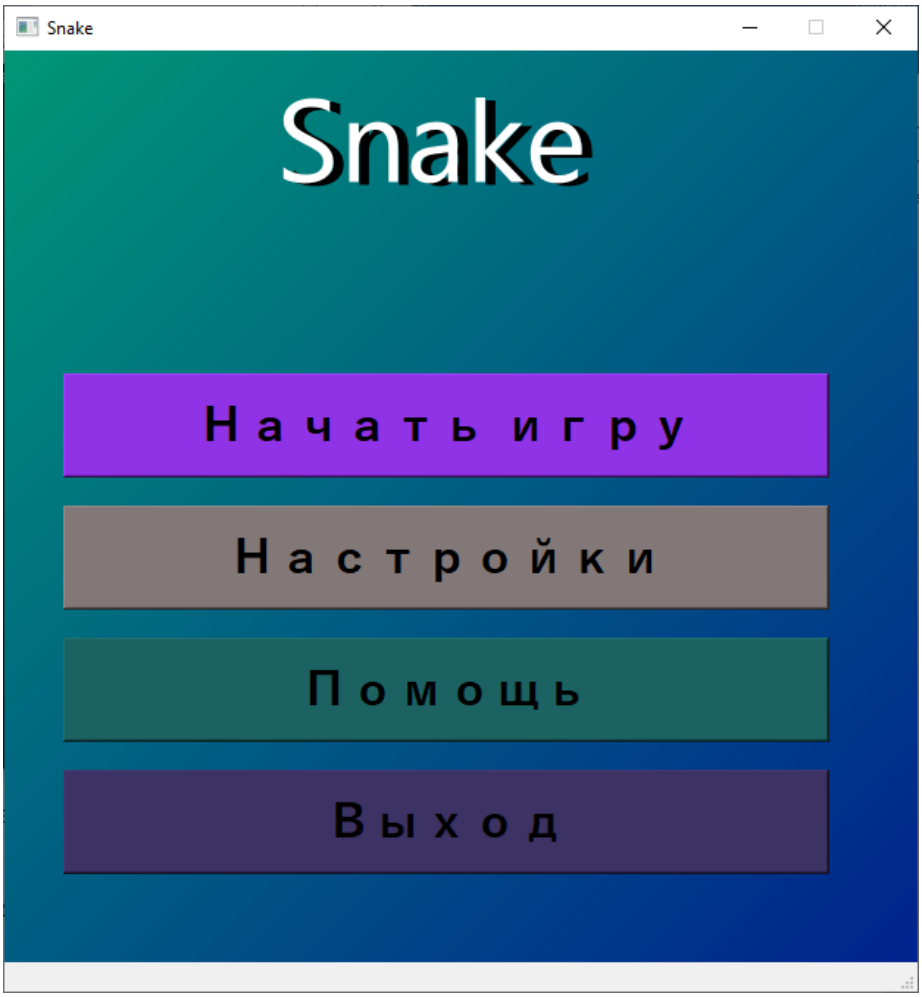

Рис. 2 – Главное окно приложения.

Это главное меню игры. Здесь мы можем увидеть логотип приложения, а также четыре кнопки:

1. «Начать игру»;
2. «Настройки»;
3. «Помощь»;
4. «Выход».

«Начать игру» – При нажатии на эту кнопку начнется игра, змейка тут же побежит из левого угла поля в правый, так что рекомендуем вам быть готовым, особенно если уровень сложности достаточно высок;

«Настройки» – При нажатии на эту кнопку вы перейдете во вкладку настроек (рис. 3). Здесь вы можете изменить сложность игры при помощи ползунка с подписью «Скорость змейки:», а также сбросить статистику, установить натройки по умолчанию, либо же вернуться обратно в главное меню, нажав на соответствующие кнопки.

Рис. 3 – Меню настроек.

«Помощь» – При нажатии на эту кнопку, вам будет показано сообщение с кратким описанием игрового процесса и вашей задачи в нем (рис. 4). Если вы решите что вам все понятно, нажмите на соответствующую кнопку “Yes”, приятной игры! В противном же случае, мы предложим вам сыграть с низкой сложностью.

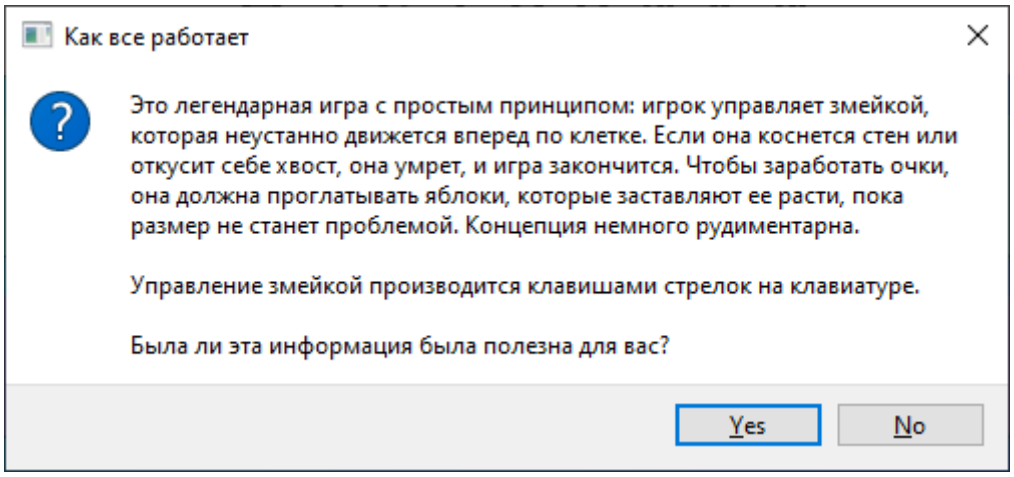

Рис. 4 – Сообщение пользователю.

«Выход» – При нажатии на эту кнопку приложение немедленно закроется.

РЕАЛИЗАЦИЯ

Средства разработки.

В ходе разработки приложения был использован фреймворк Qt с его стандартной библиотекой для языка C++, а также готовыми решениями такими как виджеты, переопределением функций обрабатывающих события нажатия клавиш, счетчика времени, а также отрисовки и перерисовки интерфейса.

Код программы.

Программы состоит из 7 основных файлов:

1. game.h;
2. game.cpp;
3. mainwindow.h;
4. mainwindow.cpp;
5. mainwindow.ui;
6. main.cpp;
7. CMakeLists.txt.

Листинги вышеперечисленных файлов приведены ниже, в том же порядке.

Листинг 1 – Код файла game.h.

```
#ifndef GAME_H
#define GAME_H

#include <iostream>
#include <QWidget>
#include <QVector>
#include <QPoint>
#include <QKeyEvent>
#include <QPainter>
#include <QTime>
#include <QImage>
#include <cstdlib>
#include <climits>
#include <algorithm>

class Game : public QWidget
{

public:

    Game();

    short getDirection() { return m_dir; }
    void setDirection(short dir) { m_dir = dir; }
    int getDelay() { return DELAY; }
    void setDelay(int del) { DELAY = del; }
    int getScore() { return score; }

    bool isInGame() { return m_in_game; }

    void initGame();
```

```
    static const int DOT_WIDTH = 30;  //  Ширина и высота точки (части
змейки).
    static const int DOT_HEIGHT = 30;
    static const int FIELD_WIDTH = 20;  //  Ширина и высота игрового поля.
    static const int FIELD_HEIGHT = 18;
    int DELAY = 148;  //  Задержка обновления игрового процесса. default: 148
(4 * 37)

protected:
    void timerEvent(QTimerEvent*) override;
    void paintEvent(QPaintEvent*) override;

private:

    void doDrawing();
    void localApple();
    void move();
    void checkField();
    void checkApple();



    short m_dir;  //  Поле направления. (0 - Up, 1 - Right, 2 - Down, 3 -
Left);

    int m_timer_id;

    QPoint m_apple;

    bool m_in_game;
    QVector<QPoint> m_dots;  //  Вектор содержащий информацию о положении
змейки.

    int score;

    QImage apple_img;
    QImage head_up_img;
    QImage head_down_img;
    QImage head_left_img;
    QImage head_right_img;
    QImage body_img;
    QImage body_vertical_img;
    QImage body_12_to_3_img;
    QImage body_3_to_6_img;
    QImage body_6_to_9_img;
    QImage body_9_to_12_img;
    QImage tail_up_img;
    QImage tail_down_img;
    QImage tail_left_img;
    QImage tail_right_img;

};

#endif // GAME_H
```

Листинг 2 – Код файла game.cpp.

```
#include "game.h"

#include <QDebug>

//  Конструктор.
Game::Game()
{
    this->setWindowFlags(Qt::Widget | Qt::FramelessWindowHint);

    resize(DOT_WIDTH * FIELD_WIDTH, DOT_HEIGHT * FIELD_HEIGHT);

    //  Загрузка изображений.
    apple_img.load("./textures/apple.png");
    apple_img = apple_img.scaled(30, 30, Qt::KeepAspectRatio);

    head_up_img.load("./textures/head_up.png");
    head_up_img = head_up_img.scaled(30, 30, Qt::KeepAspectRatio);
    head_down_img.load("./textures/head_down.png");
    head_down_img = head_down_img.scaled(30, 30, Qt::KeepAspectRatio);
    head_left_img.load("./textures/head_left.png");
    head_left_img = head_left_img.scaled(30, 30, Qt::KeepAspectRatio);
    head_right_img.load("./textures/head_right.png");
    head_right_img = head_right_img.scaled(30, 30, Qt::KeepAspectRatio);

    body_img.load("./textures/body.png");
    body_img = body_img.scaled(30, 30, Qt::KeepAspectRatio);
    body_vertical_img.load("./textures/body_vertical.png");
    body_vertical_img = body_vertical_img.scaled(30, 30,
Qt::KeepAspectRatio);
    body_3_to_6_img.load("./textures/body_3_to_6.png");
    body_3_to_6_img = body_3_to_6_img.scaled(30, 30, Qt::KeepAspectRatio);
    body_6_to_9_img.load("./textures/body_6_to_9.png");
    body_6_to_9_img = body_6_to_9_img.scaled(30, 30, Qt::KeepAspectRatio);
    body_9_to_12_img.load("./textures/body_9_to_12.png");
    body_9_to_12_img = body_9_to_12_img.scaled(30, 30, Qt::KeepAspectRatio);
    body_12_to_3_img.load("./textures/body_12_to_3.png");
    body_12_to_3_img = body_12_to_3_img.scaled(30, 30, Qt::KeepAspectRatio);

    tail_up_img.load("./textures/tail_up.png");
    tail_up_img = tail_up_img.scaled(30, 30, Qt::KeepAspectRatio);
    tail_down_img.load("./textures/tail_down.png");
    tail_down_img = tail_down_img.scaled(30, 30, Qt::KeepAspectRatio);
    tail_left_img.load("./textures/tail_left.png");
    tail_left_img = tail_left_img.scaled(30, 30, Qt::KeepAspectRatio);
    tail_right_img.load("./textures/tail_right.png");
    tail_right_img = tail_right_img.scaled(30, 30, Qt::KeepAspectRatio);

}


//  Таймер (движение игрового процесса).
void Game::timerEvent(QTimerEvent *)
{
    if (m_in_game)
    {
        checkApple();
        move();
        checkField();
    }

    this->repaint();  //  Перерисовать картинку.
}
```

```
//  Отрисовка.
void Game::paintEvent(QPaintEvent *e)
{
    Q_UNUSED(e);

    doDrawing();
}

//  Главная логика игры, отрисовка.
void Game::doDrawing()
{
    QPainter qp(this);

    if (m_in_game)  //  Пока не проиграли.
    {
        //  Отрисовка яблока.
        qp.drawImage(QPoint(m_apple.x() * DOT_WIDTH, m_apple.y() *
DOT_HEIGHT), apple_img);

        //  Отрисовка змейки.
        for (int i = 0; i < m_dots.size(); i++)
        {
            if (i == 0)  //  Отрисовка головы змейки.
            {
                switch (m_dir) {
                case 0:  //  Движется вверх.
                    qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y() * DOT_HEIGHT), head_up_img);
                    break;
                case 1:  //  Движется вправо.
                    qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y() * DOT_HEIGHT), head_right_img);
                    break;
                case 2:  //  Движется вниз.
                    qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y() * DOT_HEIGHT), head_down_img);
                    break;
                case 3:  //  Движется влево.
                    qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y() * DOT_HEIGHT), head_left_img);
                    break;
                }
            }

            else if (i == (m_dots.size() - 1))  //  Отрисовка кончика хвоста
змейки.
            {
                //  Хвост тянет вверх.
                if (m_dots[m_dots.size() - 2].y() < m_dots[m_dots.size() -
1].y())
                {
                    qp.drawImage(QPoint(m_dots[m_dots.size() - 1].x() *
DOT_WIDTH, m_dots[m_dots.size() - 1].y()
                                 * DOT_HEIGHT), tail_down_img);
                }
                //  Хвост тянет вниз.
                else if (m_dots[m_dots.size() - 2].y() > m_dots[m_dots.size()
- 1].y())
                {
                    qp.drawImage(QPoint(m_dots[m_dots.size() - 1].x() *
DOT_WIDTH, m_dots[m_dots.size() - 1].y()
                                 * DOT_HEIGHT), tail_up_img);
```

```
                }
                //  Хвост тянет влево.
                else if (m_dots[m_dots.size() - 2].x() < m_dots[m_dots.size()
- 1].x())
                {
                    qp.drawImage(QPoint(m_dots[m_dots.size() - 1].x() *
DOT_WIDTH, m_dots[m_dots.size() - 1].y()
                            * DOT_HEIGHT), tail_right_img);
                }
                //  Хвост тянет вправо.
                else if (m_dots[m_dots.size() - 2].x() > m_dots[m_dots.size()
- 1].x())
                {
                    qp.drawImage(QPoint(m_dots[m_dots.size() - 1].x() *
DOT_WIDTH, m_dots[m_dots.size() - 1].y()
                            * DOT_HEIGHT), tail_left_img);
                }
            }

            else //  Отрисовка тела змейки (все кроме головы и кончика
хвоста).
            {
                //  Отрисовка прямых частей тела.
                if ((m_dots[i].y() == m_dots[i+1].y())&&(m_dots[i].y() ==
m_dots[i-1].y()))  //  Горизонтальное положение.
                {
                    qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y()
                            * DOT_HEIGHT), body_img);
                }
                else if ((m_dots[i].x() == m_dots[i+1].x())&&(m_dots[i].x()
== m_dots[i-1].x()))  //  Вертикальное положение.
                {
                    qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y()
                            * DOT_HEIGHT), body_vertical_img);
                }

                else
                {
                    //  Отрисовка кривых частей тела.
                    //  12 to 3 pm.
                    if ((m_dots[i].y() == std::min(m_dots[i-1].y(),
m_dots[i+1].y()))&&(m_dots[i].x() == std::max(m_dots[i-1].x(),
m_dots[i+1].x())))
                    {
                        qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y()
                                * DOT_HEIGHT), body_12_to_3_img);
                    }
                    //  3 to 6 pm.
                    else if ((m_dots[i].y() == std::max(m_dots[i-1].y(),
m_dots[i+1].y()))&&(m_dots[i].x() == std::max(m_dots[i-1].x(),
m_dots[i+1].x())))
                    {
                        qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y()
                                * DOT_HEIGHT), body_3_to_6_img);
                    }
                    //  6 to 9 pm.
                    else if ((m_dots[i].y() == std::max(m_dots[i-1].y(),
m_dots[i+1].y()))&&(m_dots[i].x() == std::min(m_dots[i-1].x(),
m_dots[i+1].x())))
                    {
```

```
                        qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y()
                                * DOT_HEIGHT), body_6_to_9_img);
                    }
                    //  9 to 12 pm.
                    else if ((m_dots[i].y() == std::min(m_dots[i-1].y(),
m_dots[i+1].y()))&&(m_dots[i].x() == std::min(m_dots[i-1].x(),
m_dots[i+1].x())))
                    {
                        qp.drawImage(QPoint(m_dots[i].x() * DOT_WIDTH,
m_dots[i].y()
                                * DOT_HEIGHT), body_9_to_12_img);
                    }
                }
            }
        }
    }
    else  //  Проигрышь.
    {
        m_in_game = false;
    }
}

//  Спавн яблока.
void Game::localApple()
{
    QTime time = QTime::currentTime();
    qsrand((uint) time.msec());

    m_apple.rx() = qrand() % FIELD_WIDTH;
    m_apple.ry() = qrand() % FIELD_HEIGHT;
}

//  Движение змейки.
void Game::move()
{

    //  Установка начальной позиции змейки.
    for (int i = m_dots.size() - 1; i > 0; i--)
    {
        m_dots[i] = m_dots[i-1];
    }

    //  В зависимости от направления меняем координаты.
    switch (m_dir)
    {
    case 0:     {m_dots[0].ry() -= 1; break;}  //up
    case 1:     {m_dots[0].rx() += 1; break;}  //right
    case 2:     {m_dots[0].ry() += 1; break;}  //down
    case 3:     {m_dots[0].rx() -= 1; break;}  //left
    }
}

//  Обработчик проигрышей.
void Game::checkField()
{
    //  Не съела ли змейка саму себя.
    if (m_dots.size() > 4)
    {
        for (int i = 1; i < m_dots.size(); i++)
        {
            if (m_dots[0] == m_dots[i])
                m_in_game = false;
        }
```

```
    }

    //  Не врезалась ли змейка в границы.
    if (m_dots[0].x() >= FIELD_WIDTH)  { m_in_game = false; }
    if (m_dots[0].x() < 0)             { m_in_game = false; }
    if (m_dots[0].y() >= FIELD_HEIGHT) { m_in_game = false; }
    if (m_dots[0].y() < 0)             { m_in_game = false; }

    if (!m_in_game)
    {
        killTimer(m_timer_id);  //  Остановка таймера, в случае проигрыша.
    }
}

//  Проверка, съела ли змейка яблоко.
void Game::checkApple()
{
    if (m_apple == m_dots[0])
    {
        m_dots.push_back(QPoint(0, 0));
        score++;
        localApple();
    }
}

//  Начальная инициализация игры.
void Game::initGame()
{
    m_in_game = true;
    m_dir = 1;
    score = 0;

    m_dots.resize(3);

    for (int i = 0; i < m_dots.size(); i++)
    {
        m_dots[i].rx() = m_dots.size() - i - 1;
        m_dots[i].ry() = 0;
    }

    localApple();

    m_timer_id = startTimer(DELAY);
}
```

Листинг 3 – Код файла mainwindow.h.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QVBoxLayout>
#include "game.h"
#include <QKeyEvent>
#include <QTime>
#include <QDebug>
#include <QThread>
#include <QLabel>
#include <QFile>
#include <QTextStream>


QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    void gameOver();

    void hideAll();

    void showMainMenu();
protected:
    void keyPressEvent(QKeyEvent*) override;
    void timerEvent(QTimerEvent*) override;

private slots:
    void on_button_start_game_clicked();

    void on_button_go_main_menu_clicked();

    void on_button_restart_clicked();

    void on_button_exit_clicked();

    void on_button_options_clicked();

    void on_button_go_main_opt_clicked();

    void on_button_set_default_clicked();

    void on_button_reset_stat_clicked();

    void on_button_help_clicked();

private:
    Ui::MainWindow *ui;
    Game *game;
    QLabel *game_bar;
    QVBoxLayout *layout;
```

```
    const char* STYLE_1 = "background-color: qlineargradient(spread:pad,
x1:0, y1:0, x2:1, y2:1, stop:0 rgba(0, 29, 23, 255), stop:1 rgba(0, 8, 34,
255))";
    const char* GAME_OVER_SCORE_TEXT = "Набрано очков: \n\nРекорд: ";
    int timer;

    int checkRecordFile();
};
#endif // MAINWINDOW_H
```

Листинг 4 – Код файла mainwindow.cpp.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("Snake");

    game = new Game;
    game->setFixedSize(game->DOT_WIDTH * game->FIELD_WIDTH, game->DOT_HEIGHT
* game->FIELD_HEIGHT);

    game_bar = new QLabel(this);
    game_bar->setGeometry(0, 0, 600, 60);
    game_bar->setFixedSize(600, 60);
    game_bar->setStyleSheet("background-color: 'black'");
    game_bar->setStyleSheet("color: 'white'");
    game_bar->setFont(QFont("Myanmar Text", 38));
    game_bar->setAlignment(Qt::AlignLeft);

    //  Объединение виджета игры с таблицей счета над ней.
    layout = new QVBoxLayout(this);
    layout->addWidget(game_bar);
    layout->addWidget(game);
    ui->centralwidget->setLayout(layout);

    //  Выбор 10 уровней сложности на слайдере.
    ui->slider_difficulty->setMinimum(0);
    ui->slider_difficulty->setMaximum(10);
    ui->slider_difficulty->setTickPosition(QSlider::TicksBelow);
    ui->slider_difficulty->setTickInterval(1);

    this->setFixedSize(621, 641);

    hideAll();
    showMainMenu();

}

MainWindow::~MainWindow()
{
    delete ui;
    delete game;
    delete game_bar;
    delete layout;
```

```
    delete STYLE_1;
    delete GAME_OVER_SCORE_TEXT;
}

void MainWindow::keyPressEvent(QKeyEvent *event)
{
    int key = event->key();
    if (key == Qt::Key_Left && game->getDirection() != 1)  { game-
>setDirection(3); /*left*/  }
    if (key == Qt::Key_Right && game->getDirection() != 3) { game-
>setDirection(1); /*right*/ }
    if (key == Qt::Key_Up && game->getDirection() != 2)    { game-
>setDirection(0); /*up   */ }
    if (key == Qt::Key_Down && game->getDirection() != 0)  { game-
>setDirection(2); /*down */ }
}

void MainWindow::timerEvent(QTimerEvent *)
{
    if (!game->isInGame())
    {
        killTimer(timer);
        this->gameOver();
    }
    game_bar->setText(std::to_string(game->getScore()).c_str());
}



int MainWindow::checkRecordFile()
{
    //  Открытие файла с рекордом.
    QFile record_file("./record.txt");  //  ! НЕОБХОДИМО ДОБАВИТЬ ФАЙЛ
record.txt В ДИРЕКТОРИЮ СБОРКИ !.

    if (!record_file.open(QFile::ReadOnly | QIODevice::Text)) { return -1; }
    QString record_str = record_file.readLine();
    int record;

    //  Не пустой ли файл.
    if (record_str.size() < 1)
    {
        //  Записываем 0, если пустой.
        if (record_file.isOpen()) { record_file.close(); }

        if (!record_file.open(QFile::WriteOnly | QIODevice::Text)) { return -
2; }

        QTextStream out(&record_file);

        out << "0";

        record = 0;
    }
    else
    {
        try
        {
            record = std::stoi(record_str.toStdString().c_str());
        }  catch (std::exception e) { return -3; }
    }

    //  Если рекорд побит.
    if (game->getScore() > record)
```

```
    {
        if (record_file.isOpen()) { record_file.close(); }

        if (!record_file.open(QFile::WriteOnly | QIODevice::Text)) { return -
4; }

        QTextStream out(&record_file);

        out << std::to_string(game->getScore()).c_str();
        record = game->getScore();
    }

    if (record_file.isOpen()) { record_file.close(); }

    return record;
}

void MainWindow::gameOver()
{
    hideAll();

    int record = checkRecordFile();  //  Получение и проверка рекорда.

    ui->label_game_over_score->setText(GAME_OVER_SCORE_TEXT);

    //  Вывод набранных очков.
    //  Позиция для счета в label - 15.
    ui->label_game_over_score->setText(ui->label_game_over_score-
>text().insert(15, std::to_string(game->getScore()).c_str()));

    ui->label_game_over_score->setText(ui->label_game_over_score->text() +
std::to_string(record).c_str());
    ui->centralwidget->setStyleSheet(STYLE_1);
    ui->label_game_over->show();
    ui->label_game_over_score->show();
    ui->button_restart->show();
    ui->button_go_main_menu->show();
}

void MainWindow::hideAll()
{
    ui->centralwidget->setStyleSheet("background-color:
qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0 rgba(0, 151, 119,
255), stop:1 rgba(0, 34, 142, 255))");
    //  Main menu.
    ui->title_label->hide();
    ui->title_label_2->hide();
    ui->button_start_game->hide();
    ui->button_options->hide();
    ui->button_help->hide();
    ui->button_exit->hide();

    //  Game.
    game->hide();
    ui->label_game_over->hide();
    ui->label_game_over_score->hide();
    ui->button_restart->hide();
    ui->button_go_main_menu->hide();
    game_bar->hide();

    //  Options.
    ui->button_go_main_opt->hide();
    ui->button_set_default->hide();
    ui->button_reset_stat->hide();
```

```
    ui->slider_difficulty->hide();
    ui->label_difficulty->hide();
}

void MainWindow::showMainMenu()
{
    //  Main menu.
    ui->title_label->show();
    ui->title_label_2->show();
    ui->button_start_game->show();
    ui->button_options->show();
    ui->button_help->show();
    ui->button_exit->show();
}


void MainWindow::on_button_start_game_clicked()
{
    ui->centralwidget->setFocusPolicy(Qt::StrongFocus);  //  Установка
фокусировки на клавиатуру.
    hideAll();

    ui->centralwidget->setStyleSheet(STYLE_1);
    game->show();
    game_bar->show();

    game_bar->setText("0");
    game_bar->show();

    game->initGame();
    timer = startTimer(game->getDelay());
}


void MainWindow::on_button_go_main_menu_clicked()
{
    hideAll();
    showMainMenu();
}


void MainWindow::on_button_restart_clicked()
{
    hideAll();
    on_button_start_game_clicked();
}


void MainWindow::on_button_exit_clicked()
{
    exit(0);
}

//  Переход в раздел "Настройки"
void MainWindow::on_button_options_clicked()
{
    hideAll();

    //  Title.
    ui->title_label->show();
    ui->title_label_2->show();

    ui->button_go_main_opt->show();
    ui->button_set_default->show();
```

```
    ui->button_reset_stat->show();
    ui->slider_difficulty->show();
    ui->label_difficulty->show();
}


void MainWindow::on_button_go_main_opt_clicked()
{
    game->setDelay(600 / (ui->slider_difficulty->value() + 1));
    on_button_go_main_menu_clicked();
}


void MainWindow::on_button_set_default_clicked()
{
    ui->slider_difficulty->setValue(3);
}

void MainWindow::on_button_reset_stat_clicked()
{

}

void MainWindow::on_button_help_clicked()
{
    QMessageBox::StandardButton reply;

    reply = QMessageBox::question(this, "Как все работает", "Это легендарная
игра с простым принципом: игрок управляет змейкой, "
                                                                "которая
неустанно движется вперед по клетке. Если она коснется стен или откусит себе
хвост, "
                                                                "она умрет, и
игра закончится. Чтобы заработать очки, она должна проглатывать яблоки,
которые "
                                                                "заставляют ее
расти, пока размер не станет проблемой. Концепция немного рудиментарна."
                                                                "\n\nУправление
змейкой производится клавишами стрелок на клавиатуре."
                                                                "\n\nБыла ли эта
информация была полезна для вас?" , QMessageBox::Yes | QMessageBox::No);

    if (reply == QMessageBox::Yes) {
       QMessageBox::information(this, "Тест", "Тогда проверим что вы
поняли");
        hideAll();
        on_button_start_game_clicked();}

    else {QMessageBox::information(this, "Совет", "Мы установим вам один из
самых низких уровней игры, и предложим вам протестировать ее самостоятельно.
"
                                                  "Уверенны, что вы будете
поражены ее простотой.");
           ui->slider_difficulty->setValue(1);
           game->setDelay(600 / (ui->slider_difficulty->value() + 1));
           on_button_start_game_clicked();
    }


}
```

Листинг 5 – Код файла mainwindow.ui.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>600</width>
    <height>641</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="sizePolicy">
    <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="layoutDirection">
    <enum>Qt::LeftToRight</enum>
   </property>
   <property name="styleSheet">
    <string notr="true">QWidget {background-color:
qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0 rgba(0, 151, 119,
255), stop:1 rgba(0, 34, 142, 255))}</string>
   </property>
   <widget class="QLabel" name="title_label">
    <property name="geometry">
     <rect>
      <x>90</x>
      <y>32</y>
      <width>401</width>
      <height>91</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Myanmar Text</family>
      <pointsize>60</pointsize>
     </font>
    </property>
    <property name="autoFillBackground">
     <bool>false</bool>
    </property>
    <property name="styleSheet">
     <string notr="true">QLabel {background-color: rgba(0, 0, 0, 0)}
QLabel {color: 'white'}</string>
    </property>
    <property name="text">
     <string>Snake</string>
    </property>
    <property name="alignment">
```

```
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
    <widget class="QLabel" name="title_label_2">
     <property name="geometry">
      <rect>
       <x>92</x>
       <y>34</y>
       <width>411</width>
       <height>91</height>
      </rect>
     </property>
     <property name="font">
      <font>
       <family>Myanmar Text</family>
       <pointsize>60</pointsize>
      </font>
     </property>
     <property name="autoFillBackground">
      <bool>false</bool>
     </property>
     <property name="styleSheet">
      <string notr="true">QLabel {background-color: rgba(0, 0, 0, 0)}
QLabel {color: rgb(0, 0, 0)}</string>
     </property>
     <property name="text">
      <string>Snake</string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
    <widget class="QPushButton" name="button_start_game">
     <property name="geometry">
      <rect>
       <x>40</x>
       <y>220</y>
       <width>521</width>
       <height>71</height>
      </rect>
     </property>
     <property name="palette">
      <palette>
       <active>
        <colorrole role="Button">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>144</red>
           <green>50</green>
           <blue>230</blue>
          </color>
         </brush>
        </colorrole>
        <colorrole role="Base">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>144</red>
           <green>50</green>
           <blue>230</blue>
          </color>
         </brush>
        </colorrole>
        <colorrole role="Window">
         <brush brushstyle="SolidPattern">
```

```
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
  </colorrole>
 </active>
 <inactive>
  <colorrole role="Button">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Base">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Window">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
  </colorrole>
 </inactive>
 <disabled>
  <colorrole role="Button">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Base">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Window">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>144</red>
     <green>50</green>
     <blue>230</blue>
    </color>
   </brush>
```

```
       </colorrole>
      </disabled>
     </palette>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>25</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgba(144,50,230,
255)}</string>
    </property>
    <property name="text">
     <string>Начать игру</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_options">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>310</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="palette">
     <palette>
      <active>
       <colorrole role="Button">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>130</red>
          <green>120</green>
          <blue>120</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>130</red>
          <green>120</green>
          <blue>120</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>130</red>
          <green>120</green>
          <blue>120</blue>
         </color>
        </brush>
       </colorrole>
      </active>
      <inactive>
```

```
    <colorrole role="Button">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>130</red>
       <green>120</green>
       <blue>120</blue>
      </color>
     </brush>
    </colorrole>
    <colorrole role="Base">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>130</red>
       <green>120</green>
       <blue>120</blue>
      </color>
     </brush>
    </colorrole>
    <colorrole role="Window">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>130</red>
       <green>120</green>
       <blue>120</blue>
      </color>
     </brush>
    </colorrole>
   </inactive>
   <disabled>
    <colorrole role="Button">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>130</red>
       <green>120</green>
       <blue>120</blue>
      </color>
     </brush>
    </colorrole>
    <colorrole role="Base">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>130</red>
       <green>120</green>
       <blue>120</blue>
      </color>
     </brush>
    </colorrole>
    <colorrole role="Window">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>130</red>
       <green>120</green>
       <blue>120</blue>
      </color>
     </brush>
    </colorrole>
   </disabled>
  </palette>
 </property>
 <property name="font">
  <font>
   <family>Yu Gothic</family>
   <pointsize>25</pointsize>
   <weight>75</weight>
```

```
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgba(130, 120, 120,
255)}</string>
    </property>
    <property name="text">
     <string>Настройки</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_help">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>400</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="palette">
     <palette>
      <active>
       <colorrole role="Button">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>29</red>
          <green>98</green>
          <blue>98</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>29</red>
          <green>98</green>
          <blue>98</blue>
         </color>
        </brush>
       </colorrole>
       <colorrole role="Window">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>29</red>
          <green>98</green>
          <blue>98</blue>
         </color>
        </brush>
       </colorrole>
      </active>
      <inactive>
       <colorrole role="Button">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>29</red>
          <green>98</green>
          <blue>98</blue>
         </color>
        </brush>
       </colorrole>
```

```
        <colorrole role="Base">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>29</red>
           <green>98</green>
           <blue>98</blue>
          </color>
         </brush>
        </colorrole>
        <colorrole role="Window">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>29</red>
           <green>98</green>
           <blue>98</blue>
          </color>
         </brush>
        </colorrole>
       </inactive>
       <disabled>
        <colorrole role="Button">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>29</red>
           <green>98</green>
           <blue>98</blue>
          </color>
         </brush>
        </colorrole>
        <colorrole role="Base">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>29</red>
           <green>98</green>
           <blue>98</blue>
          </color>
         </brush>
        </colorrole>
        <colorrole role="Window">
         <brush brushstyle="SolidPattern">
          <color alpha="255">
           <red>29</red>
           <green>98</green>
           <blue>98</blue>
          </color>
         </brush>
        </colorrole>
       </disabled>
      </palette>
     </property>
     <property name="font">
      <font>
       <family>Yu Gothic</family>
       <pointsize>25</pointsize>
       <weight>75</weight>
       <bold>true</bold>
      </font>
     </property>
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="styleSheet">
      <string notr="true">QPushButton {background-color: rgba(29,98 ,98,
255)}</string>
```

```
    </property>
    <property name="text">
     <string>Помощь</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_exit">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>490</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>25</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgba(60,50,100,
255)}</string>
    </property>
    <property name="text">
     <string>Выход</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_game_over">
    <property name="enabled">
     <bool>true</bool>
    </property>
    <property name="geometry">
     <rect>
      <x>90</x>
      <y>70</y>
      <width>441</width>
      <height>101</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>DejaVu Sans</family>
      <pointsize>30</pointsize>
     </font>
    </property>
    <property name="styleSheet">
     <string notr="true">QLabel {background-color: rgba(0, 0, 0, 0)}
QLabel {color: 'white'}</string>
    </property>
    <property name="text">
     <string>Вы проиграли:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
   <widget class="QPushButton" name="button_restart">
    <property name="geometry">
```

```
     <rect>
      <x>40</x>
      <y>400</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>25</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgba(0, 0, 0, 0)}
QPushButton {color: 'white'}</string>
    </property>
    <property name="text">
     <string>Начать заново</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_go_main_menu">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>490</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>25</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgba(100, 50, 230,
0)}
QPushButton {color: 'white'}</string>
    </property>
    <property name="text">
     <string>Главное меню</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_game_over_score">
    <property name="geometry">
     <rect>
      <x>110</x>
      <y>210</y>
      <width>421</width>
      <height>121</height>
     </rect>
    </property>
```

```
    <property name="font">
     <font>
      <pointsize>24</pointsize>
     </font>
    </property>
    <property name="styleSheet">
     <string notr="true">QLabel {background-color: rgba(0, 0, 0, 0)}
QLabel {color: 'white'}</string>
    </property>
    <property name="text">
     <string>Набрано очков:

Рекорд: </string>
    </property>
   </widget>
   <widget class="QSlider" name="slider_difficulty">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>180</y>
      <width>521</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">QSlider {background-color: rgba(0, 0, 0, 0)}
QSlider {color: 'white'}</string>
    </property>
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
   </widget>
   <widget class="QLabel" name="label_difficulty">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>140</y>
      <width>361</width>
      <height>41</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Myanmar Text</family>
      <pointsize>24</pointsize>
     </font>
    </property>
    <property name="styleSheet">
     <string notr="true">QLabel {background-color: rgba(0, 0, 0, 0)}
QLabel {color: 'white'}</string>
    </property>
    <property name="text">
     <string>Скорость змейки:</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_set_default">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>400</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
```

```
    <property name="sizePolicy">
     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>12</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgb(222, 222,
222)}</string>
    </property>
    <property name="text">
     <string>Установить настройки по умолчанию</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_go_main_opt">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>490</y>
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="sizePolicy">
     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>12</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgb(222, 222,
222)}</string>
    </property>
    <property name="text">
     <string>Вернуться в главное меню</string>
    </property>
   </widget>
   <widget class="QPushButton" name="button_reset_stat">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>310</y>
```

```
      <width>521</width>
      <height>71</height>
     </rect>
    </property>
    <property name="sizePolicy">
     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="font">
     <font>
      <family>Yu Gothic</family>
      <pointsize>12</pointsize>
      <weight>75</weight>
      <bold>true</bold>
     </font>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton {background-color: rgb(222, 222,
222)}</string>
    </property>
    <property name="text">
     <string>Сбросить статистику</string>
    </property>
   </widget>
   <zorder>button_options</zorder>
   <zorder>button_reset_stat</zorder>
   <zorder>button_start_game</zorder>
   <zorder>label_game_over_score</zorder>
   <zorder>button_help</zorder>
   <zorder>button_go_main_menu</zorder>
   <zorder>button_exit</zorder>
   <zorder>button_restart</zorder>
   <zorder>button_set_default</zorder>
   <zorder>button_go_main_opt</zorder>
   <zorder>label_game_over</zorder>
   <zorder>title_label_2</zorder>
   <zorder>title_label</zorder>
   <zorder>slider_difficulty</zorder>
   <zorder>label_difficulty</zorder>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>600</width>
     <height>21</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```

Листинг 6 – Код файла main.cpp.

```
#include "mainwindow.h"

#include <QApplication>
#include <QLocale>
#include <QTranslator>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "Project_Snake_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    MainWindow w;
    w.show();
    return a.exec();
}
```

Листинг 7 – Код файла CMakeLists.txt.

```
cmake_minimum_required(VERSION 3.5)

project(Project_Snake_Cmake VERSION 0.1 LANGUAGES CXX)

set(CMAKE_INCLUDE_CURRENT_DIR ON)

set(CMAKE_AUTOUIC ON)
set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)

set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

find_package(QT NAMES Qt6 Qt5 COMPONENTS Widgets REQUIRED)
find_package(Qt${QT_VERSION_MAJOR} COMPONENTS Widgets REQUIRED)

set(PROJECT_SOURCES
        main.cpp
        mainwindow.cpp
        mainwindow.h
        mainwindow.ui
        game.cpp
        game.h

)

if(${QT_VERSION_MAJOR} GREATER_EQUAL 6)
    qt_add_executable(Project_Snake_Cmake
        MANUAL_FINALIZATION
        ${PROJECT_SOURCES}
    )
# Define target properties for Android with Qt 6 as:
```

```
#    set_property(TARGET Project_Snake_Cmake APPEND PROPERTY
QT_ANDROID_PACKAGE_SOURCE_DIR
#                 ${CMAKE_CURRENT_SOURCE_DIR}/android)
# For more information, see https://doc.qt.io/qt-6/qt-add-
executable.html#target-creation
else()
    if(ANDROID)
        add_library(Project_Snake_Cmake SHARED
            ${PROJECT_SOURCES}
        )
# Define properties for Android with Qt 5 after find_package() calls as:
#    set(ANDROID_PACKAGE_SOURCE_DIR "${CMAKE_CURRENT_SOURCE_DIR}/android")
    else()
        add_executable(Project_Snake_Cmake
            ${PROJECT_SOURCES}
        )
    endif()
endif()

target_link_libraries(Project_Snake_Cmake PRIVATE
Qt${QT_VERSION_MAJOR}::Widgets)

set_target_properties(Project_Snake_Cmake PROPERTIES
    MACOSX_BUNDLE_GUI_IDENTIFIER my.example.com
    MACOSX_BUNDLE_BUNDLE_VERSION ${PROJECT_VERSION}
    MACOSX_BUNDLE_SHORT_VERSION_STRING
${PROJECT_VERSION_MAJOR}.${PROJECT_VERSION_MINOR}
)

if(QT_VERSION_MAJOR EQUAL 6)
    qt_finalize_executable(Project_Snake_Cmake)
endif()
```

Демонстрация работы.

В данном разделе будет продемонстрирована работа приложения. На рисунке 5 изображено главное меню, на рисунке 6 – меню настроек. На рисунках 7, 8 и 9 представлены скриншоты игрового процесса. На рисунке 10 представлен скриншот после проигрыша.

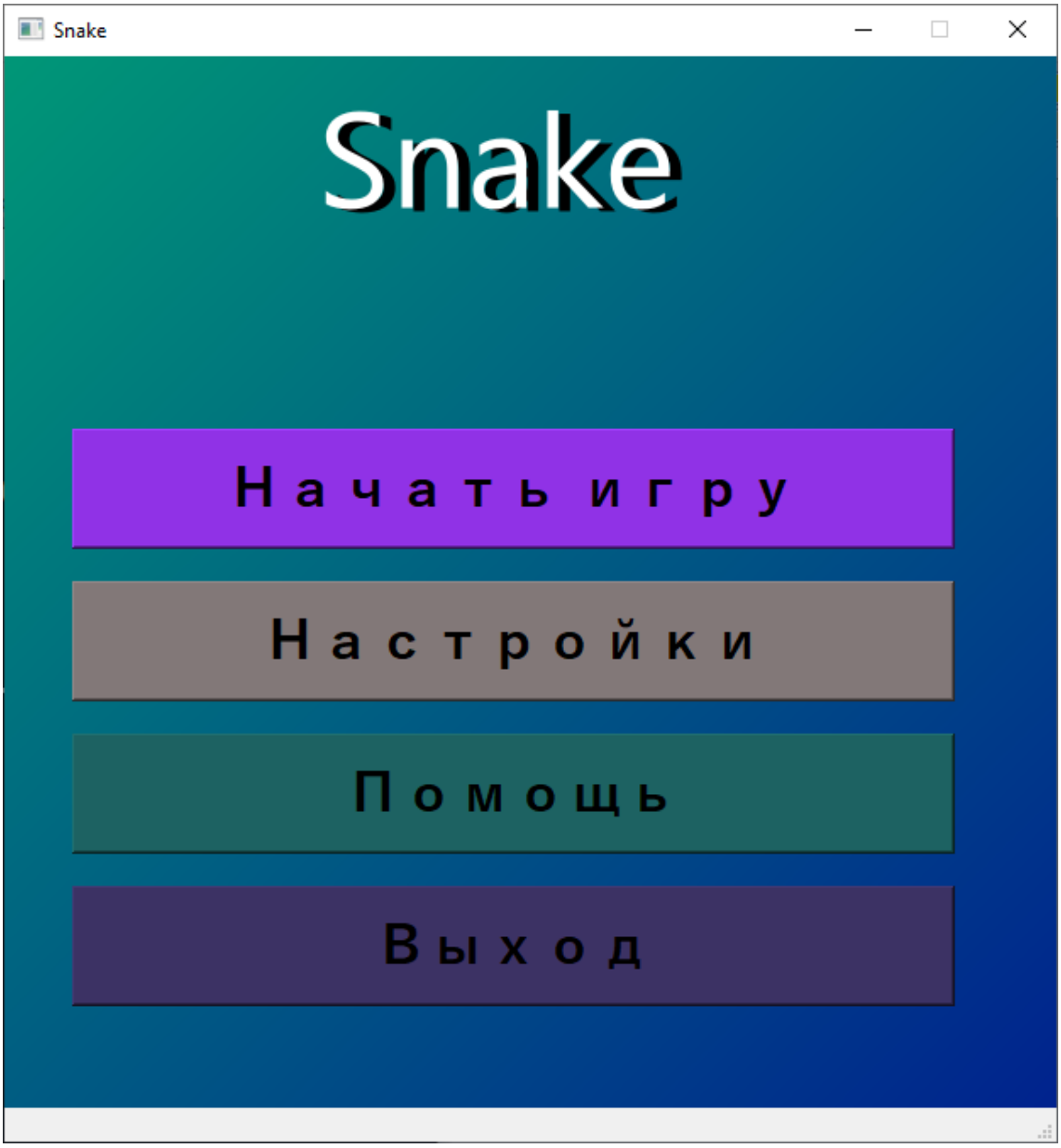

Рис. 5 – Главное меню.

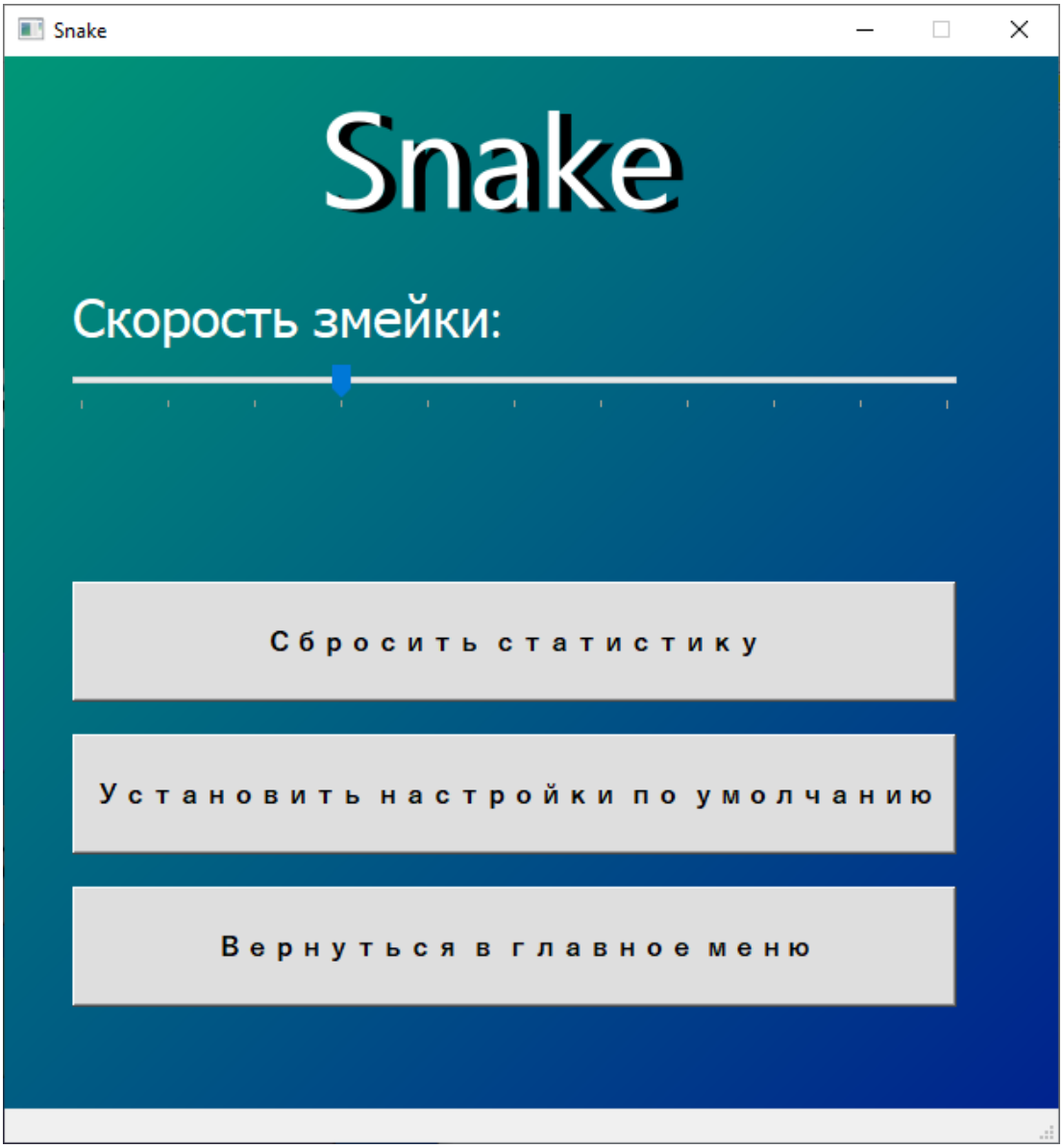

Рис. 6 – Меню настроек.

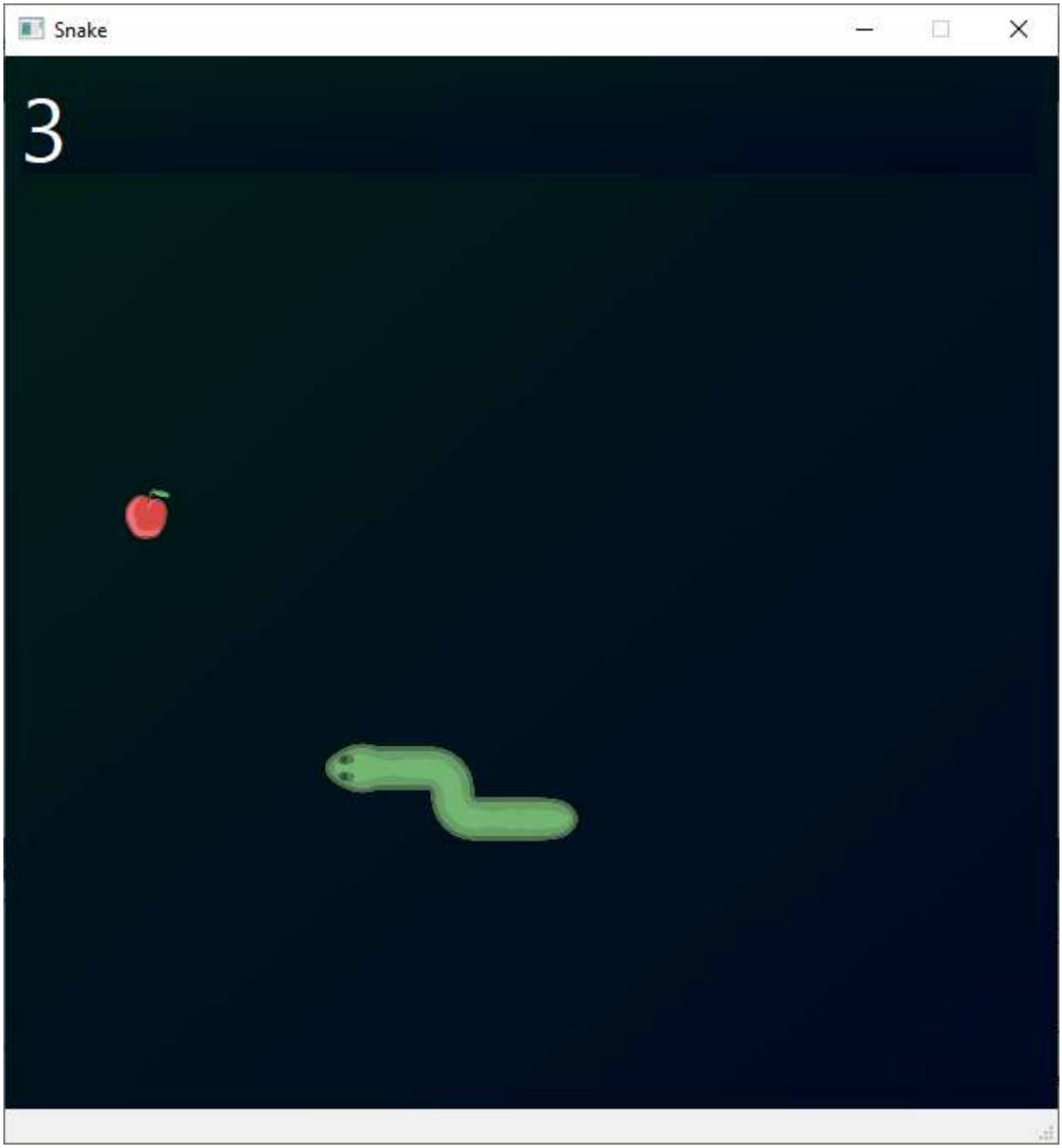

Рис. 7 – Игровой процесс.

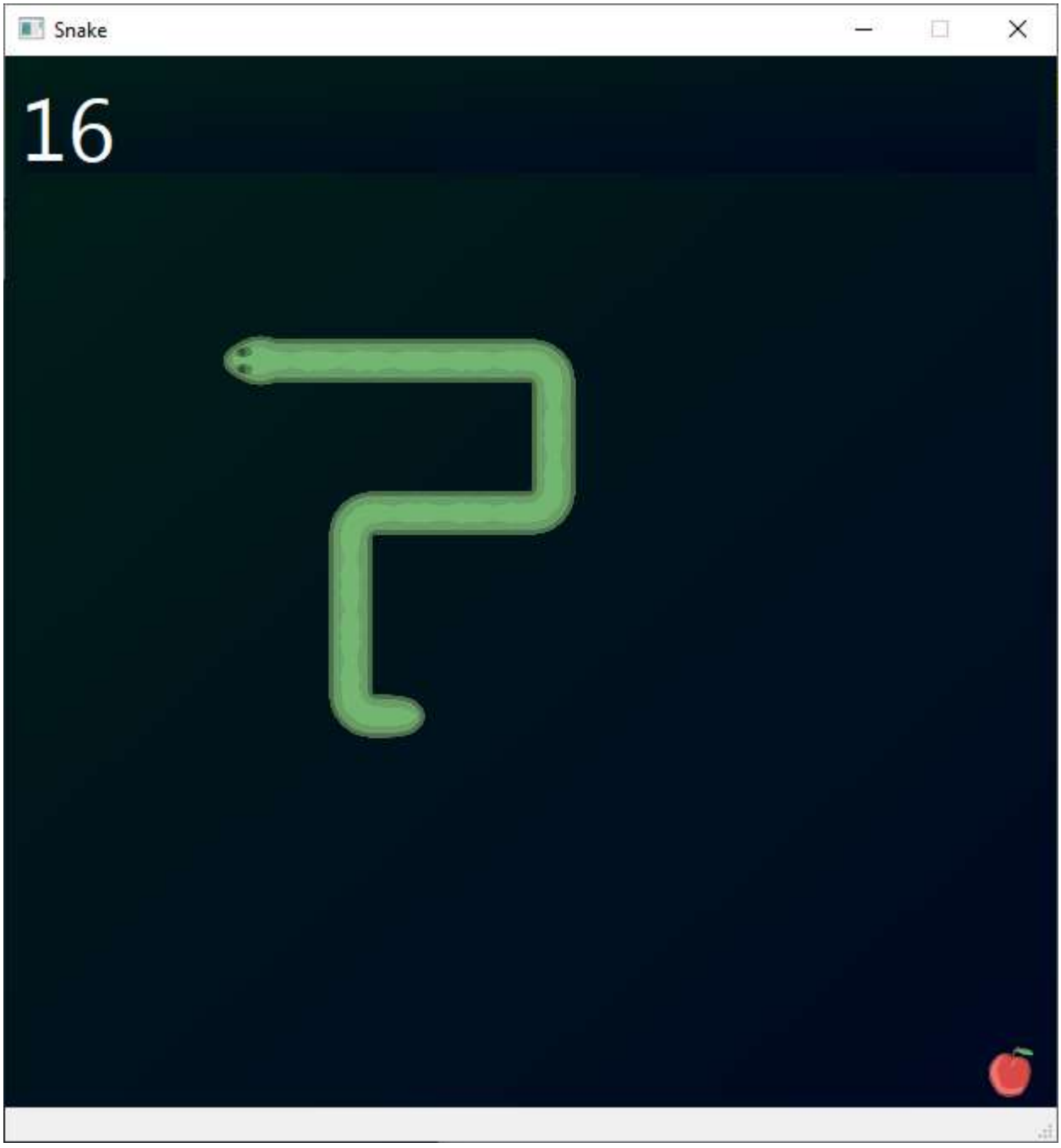

Рис. 8 – Игровой процесс.

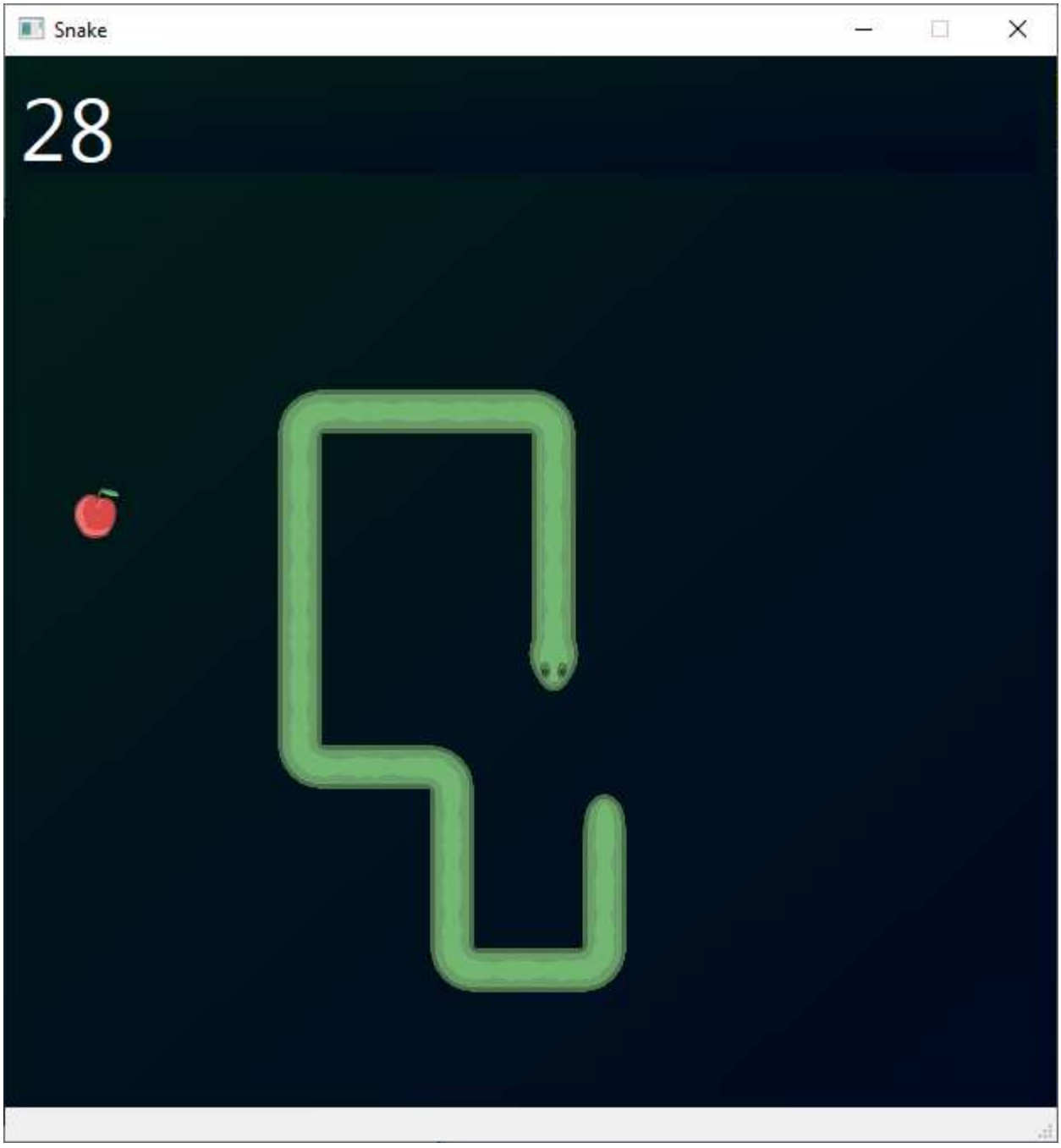

Рис. 9 – Игровой процесс.

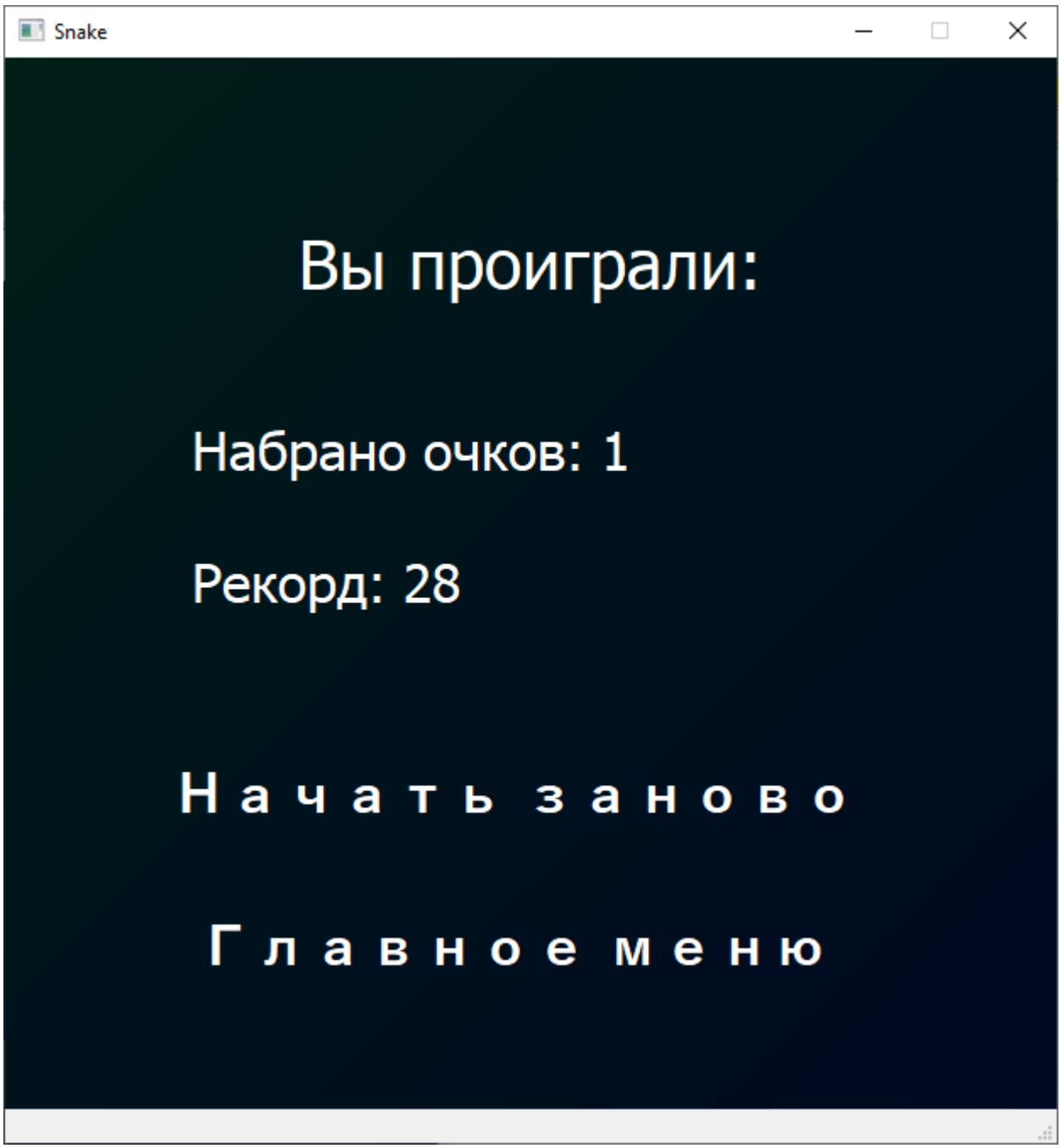

Рис. 10 – Меню проигрыша.