

Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту  
по курсу «Программирование»  
на тему «Разработка программы сложной структуры  
методом нисходящего программирования»

Выполнил  
студент группы 22ВОА1  
Пальков Э.Р.

Принял  
Митрохин М. А.

кор  
29.12.23  


Пенза 2023

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
Факультет Вычислительной техники  
Кафедра «Вычислительная техника»

«УТВЕРЖДАЮ»  
Зав. кафедрой ВТ  
Митрохин М. А.

«\_\_\_» \_\_\_\_\_ 2023г

**ЗАДАНИЕ**  
на курсовое проектирование по курсу

Программирование

Студенту Палькову Эрнесту Рустамовичу Группа 22ВОА1  
Тема проекта: Разработка программы сложной структуры методом  
нисходящего программирования

**Исходные данные (технические требования) на проектирование**  
Наименование программы: Приложение «Змейка».

**Задание.** Программа должна выполнять следующие функции:

- Вывод на экран поля для игры.
- Появление фруктов в случайном месте.
- Остановка и запуск программы.
- Возможность пользователю управлять змейкой.
- Сохранение и вывод результатов игры пользователя.

**Обязательные требования к программе:**

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, 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 — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2022.
9. Язык программирования — C++.

**Объем работы по курсу**

**1. Расчетная часть**

Разработка игры.

**2. Графическая часть**

Структура модулей программы, схемы алгоритмов программы.

**3. Экспериментальная часть**

Отладка программы.

**Срок выполнения проекта по разделам**

1. В соответствии с графиком выполнения курсового проекта.

2. \_\_\_\_\_

3. \_\_\_\_\_

Дата выдачи задания « 11 » сентября

Дата защиты проекта « \_\_\_\_ » \_\_\_\_\_ 2

Руководитель Мирохин Максим Александрович

Задание получил « 11 » сентября

Студент Полков Артём Юстианович

# Оглавление

|                                                     |    |
|-----------------------------------------------------|----|
| Введение .....                                      | 4  |
| 1 Постановка задачи .....                           | 5  |
| 2 Выбор решения .....                               | 6  |
| 2.1 Определение необходимых модулей программы ..... | 8  |
| 2.2 Определение алгоритма программы .....           | 9  |
| 3 Описание разработки программы .....               | 11 |
| 4 Отладка и тестирование .....                      | 12 |
| 5 Описание программы .....                          | 13 |
| 6 Руководство пользователя .....                    | 17 |
| Заключение .....                                    | 19 |
| Список использованных источников .....              | 20 |
| Приложение А .....                                  | 21 |
| Исходный код программы .....                        | 21 |
| Приложение Б .....                                  | 39 |
| Снимки экрана .....                                 | 41 |

## Введение

Визуализация данных — это графическое представление информации и аналитики: графики, диаграммы, карты, дашборды. Цифры в таблицах и документах не показывают наглядно взаимосвязи между процессами, периоды роста или спада, зависимости показателей.

Визуальный формат представляет информацию и вырисовывает цельную картину происходящего. Когда за обилием цифр легко утратить смысл, стоит обратиться к визуализации больших данных. Этот инструмент приводит огромное количество информации в порядок, помогает быстрее понимать её, подчёркивает тренды. В бизнесе визуализация помогает на многих этапах — от кадровых решений до предоставления скидки конкретному покупателю. [1]

Задачей данной курсовой работы стала разработка и визуализация игры «Змейка». Разработанная программа, предназначенная для игры «Змейка», может использоваться людьми, не имеющими квалификации программиста. Программа разработана с использованием современных концепций программирования - объектно-ориентированного программирования и языка C++.

Пояснительная записка содержит техническое задание, описание программы, программу и методику испытаний, описание применения, текст программы.

# 1 Постановка задачи

Необходимо разработать программу «Змейка». Проанализировав существенные программные решения, были определены следующие функциональные требования к разрабатываемой программе. Многомодульность программы. Необходимо поделить программу на логические модули. Это повысит расширяемость программы. Использование сложных типов данных – классов и объектов. Эти составные типы данных являются основными концепциями объектно-ориентированного программирования (ООП).

Программа должна иметь простой, но в то же время понятный и наглядный графический интерфейс, который не должен перегружать ресурсы компьютера. Наличие графического интерфейса позволяет осуществлять взаимодействие человека с компьютером в форме диалога с использованием окон, и элементов управления. Устройство ввода-вывода – клавиатура, микрофон и мышь. Управление в основном происходит с помощью клавиатуры. Пользовательский интерфейс должен быть построен на основе меню или панели инструментов. При разработке программы необходимо использовать среду Qt Qcreator. Язык программирования - C++. Язык программирования C++ имеет возможность применения классов при использовании обычного стиля программирования и поддерживает визуальное программирование с немедленным отображением графических объектов.

Программа должна выполнять следующие функции:

- Вывод на экран поля для игры.
- Появление фруктов в случайном месте.
- Остановка и запуск программы.
- Возможность пользователю управлять змейкой.
- Сохранение и вывод результатов игры пользователя командами;

Программа должна работать без прерываний по программным ошибкам. В случае ошибки должна выдать соответствующее сообщение.

## 2 Выбор решения

### 2.1 Определение необходимых модулей программы

Программа Snake состоит из 4 модулей: mainWindow, gameWindow, Game, helpField,. При запуске программы запускается модуль main, необходимый для связи и управления остальными модулями, затем запускается модуль mainWindow – главный графический модуль в котором располагается пункт меню и из которого осуществляется переход в остальные модули программы.

Описание модулей приведено в таблице 2.

Таблица 2 – Описание модуля

| Имя модуля | Форма      | Примечание                                 |
|------------|------------|--------------------------------------------|
| main       | mainWindow | Главный модуль программы                   |
| mainWindow | mainWindow | Модуль программы для запуска главного окна |
| gameWindow | gameWindow | Модуль программы отображения игры          |
| Game       | gameWindow | Модуль программы отвечающий за логику игры |
| helpField  | gameWindow | Вспомогательный модуль программы           |

Схема взаимодействия модулей программы представлена на рисунке 1.

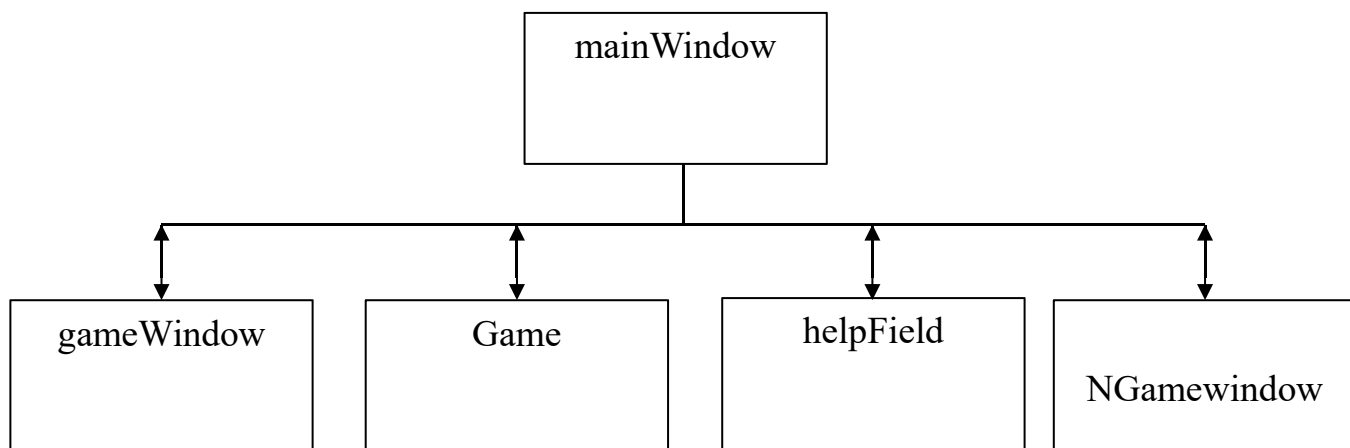


Рисунок 1 — Схема взаимодействия программ

## 2.2 Определение алгоритма программы

Для представления принципа работы программы ниже приведена ее функциональная схема (рис.2)



**Рисунок 2. Функциональная схема программы.**

В интерфейсе mainWindow содержатся кнопки перехода из одной формы в другие — GameWindow(игра).

### 3 Описание разработки программы

Язык программирования C++ является одним из самых популярных языков в мире и активно используется для разработки различных типов приложений.

C++ является кросс-платформенным языком, что означает, что приложение, разработанное на C++, может быть запущено на различных операционных системах, таких как Windows, macOS и Linux, без необходимости перекомпилирования. Это делает C++ идеальным выбором для разработки приложений с интерфейсом, которые должны работать на разных платформах.

C++ имеет обширную экосистему библиотек, позволяющую реализовывать различные функциональности в приложении с интерфейсом.

В качестве среды программирования был выбран программный продукт Qt Creator.

Интерфейс информационно-вычислительной системы будет представлен по большей части в виде кнопок и виджетов.

## 4 Отладка и тестирование

В качестве среды разработки была выбрана программа Qt Creator. Программа предоставляет все средства, необходимые при разработке и отладке вырабатываемых модулей и программ. Для отладки использовались такие инструменты, как точка останова, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с освобождением выделенной памяти и корректным отображением и т.д.

## 5 Описание программы

Проект программы включает в себя главную программу и следующие подпрограммы: Структура программы приведена на рисунке 2.

Описание процедур и функций приведено в таблице 3.

Таблица 3 – Описание процедур и функций

| Имя модуля | наименование процедуры или функции | Примечание                                          |
|------------|------------------------------------|-----------------------------------------------------|
| game       | SetGameStatus();                   | функция предназначена для отслеживания статуса игры |
|            | CreateFood();                      | функция предназначена для создания яблок            |
|            | StartNewGame();                    | функция предназначена для начала новой игры         |
|            | GameOver()                         | Функция предназначена для завершения игры           |
| mainWindow | openWindow1();                     | Функция предназначенная для открытия окна игры      |

Обработчик события SetGameStatus вызывается при запуске игрового окна и предназначен для отслеживания статуса игры.

Алгоритм процедуры SetGameStatus представлен на рисунке 3.

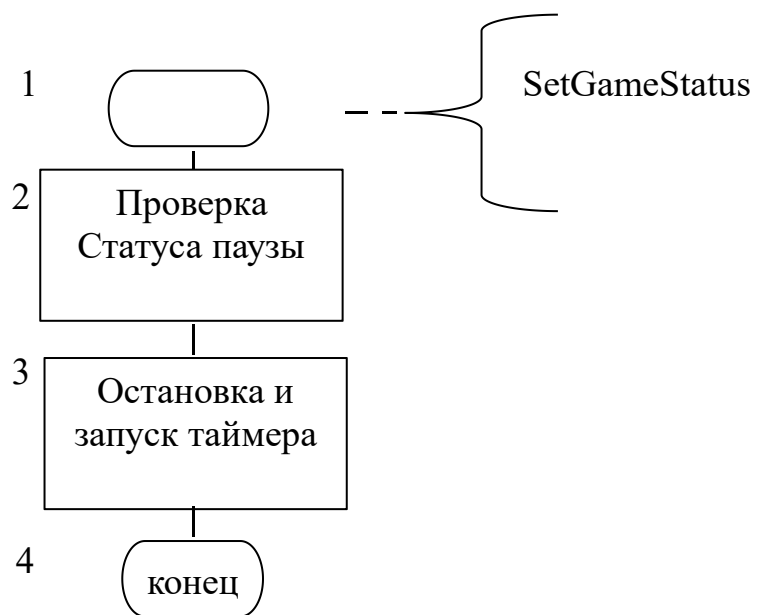


Рисунок 3 — Схема функции SetGameStatus

Алгоритм функции начинается с проверки статуса паузы(блок 2). Затем осуществляется остановка и запуск таймера(блок 3).

Функция CreateFood предназначена для создания яблок в рандомном месте .

Алгоритм функции CreateFood представлен на рисунке 4.

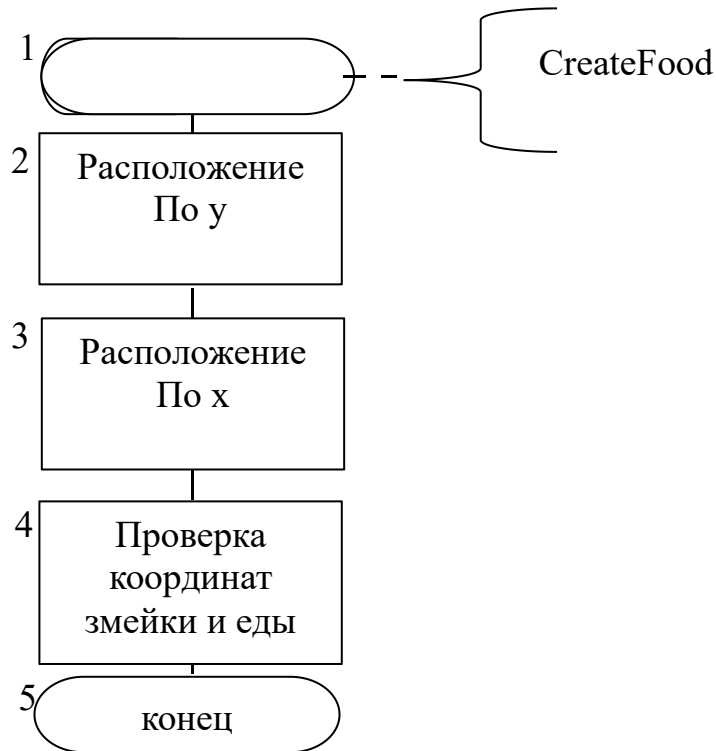


Рисунок 4 — Схема процедуры CreateFood

Алгоритм функции начинается с расположения относительно оси у яблока(блок 2). Затем по оси х(блок 3).После осуществляется проверка на сопадение координат змейки и еды(блок 4).

Функция StartNewGame предназначен для старта новой игры .

Алгоритм функции StartNewGame представлен на рисунке 5.

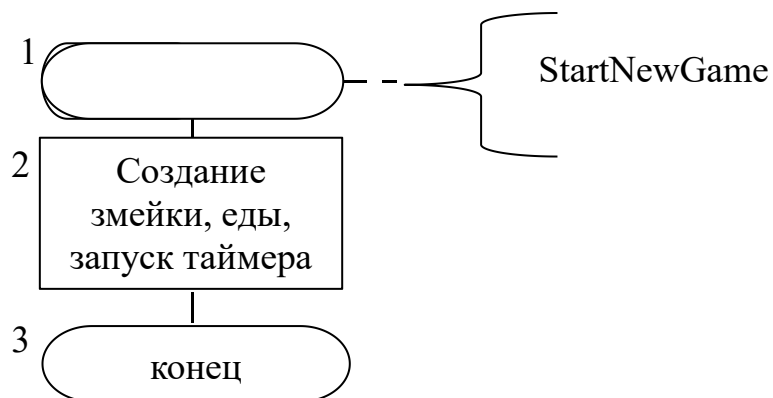


Рисунок 5 — Схема функции StartNewGame

Алгоритм функции состоит из создания змейки, еды, запуск таймера(блок 2).

Функция GameOver предназначена для инициализации смерти змейки.

Алгоритм функции GameOver представлен на рисунке 6.

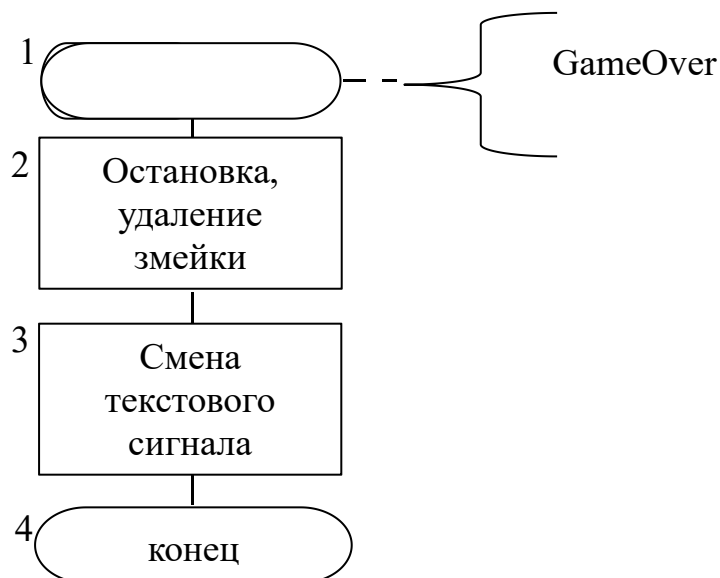


Рисунок 6 — Схема функции GameOver

Алгоритм функции начинается с Остановка, удаление змейки (блок 2). Затем осуществляется смена текстового сигнала(блок 3).

Функция openWindow1 предназначена для открытия игрового поля.

Алгоритм процедуры openWindow1 представлен на рисунке 7.



Рисунок 7 — Схема функции openWindow1

В блоке 2 выполняется создание и показ нового окна. Затем выполняется скрывание главного окна(блок 3).

## 6 Руководство пользователя

При запуске программы откроется графический интерфейс, представленный на рисунке 8.

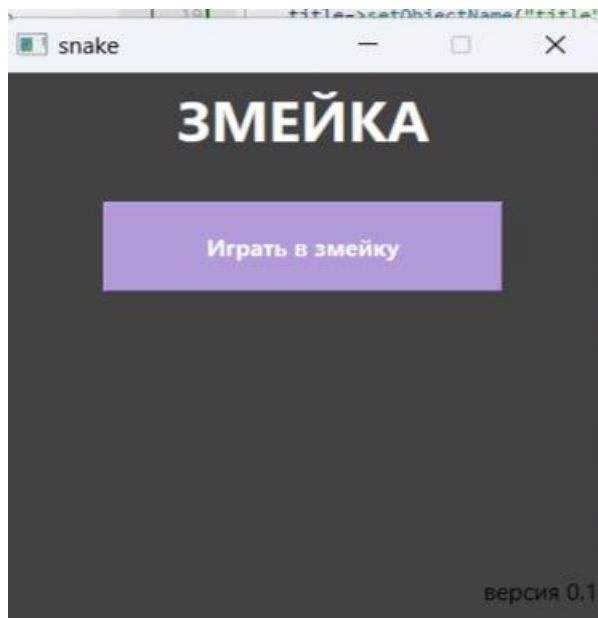


Рисунок 8 — главный экран

На главном экране расположена кнопка - «Играть в змейку», при нажатии на нее запускается игра.

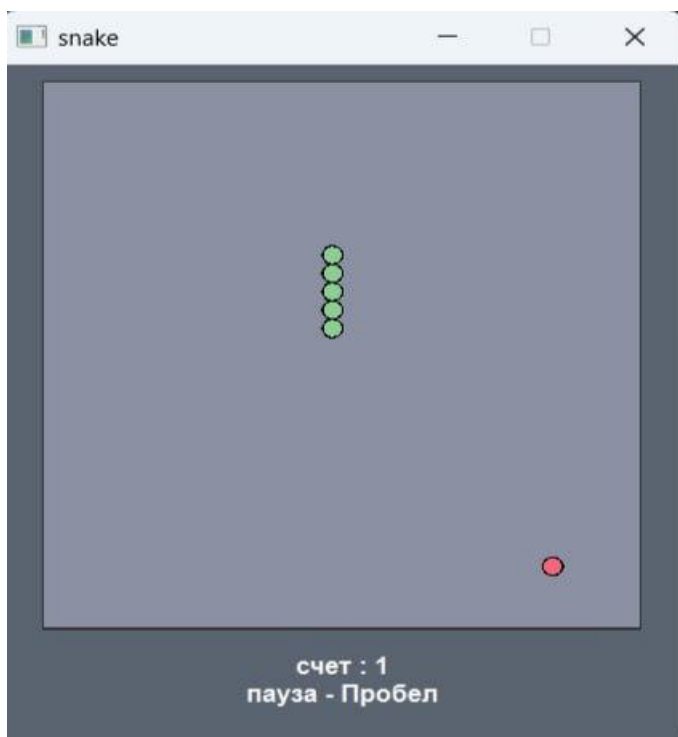


Рисунок 9 — Окно игры

В данном окне игры управление осуществляется стрелочками по нажатию кнопки пробел игра входит в режим паузы и наоборот — рисунок 9.

## **Заключение**

При выполнении данной курсовой работы были изучены навыки разработки многомодульных программ. В результате выполнения данного проекта был разработан программный продукт "Змейка". При помощи этой программы можно выполнять определенные операции-управления компьютером используя стандартные функции .

- Программа не занимает много места, не требовательна к установленному программному обеспечению.

В целом, поставленная в начале курсового проекта цель была достигнута. В программе выполняются все необходимые функциональные требования.

## Список использованных источников

1. Электронный ресурс. practicum.yandex.ru Официальный сайт. — URL: <https://practicum.yandex.ru/blog/vizualizaciya-dannyh> (дата обращения 12.11.2023).
2. Электронный ресурс. Qt.io Официальный сайт. — URL: <https://www.qt.io/> (дата обращения: 13.11. 2023).
3. Электронный ресурс. Github.com Официальный сайт. — URL: <https://github.com/> (дата обращения: 13.11.2023).

# Приложение А

## Исходный код программы

### Приложение А.1

#### Файл «main.cpp»

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

## Приложение А.2

### Файл «mainwindow.cpp»

```
#include "mainwindow.h"
#include "windowgame.h"
// #include "windowNet.h"
#include <QVBoxLayout>
#include <QPalette>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
    setFixedSize(300, 300);

    button1 = new QPushButton("Играть в змейку", this);
    button1->setGeometry(50, 70, 200, 50);
    // button2 = new QPushButton("Тренировать нейронную сеть", this);
    // button2->setGeometry(50, 150, 200, 50);

    QLabel *title = new QLabel("ЗМЕЙКА", this);
    title->setObjectName("title");
    title->setGeometry(50, 10, 200, 30);
    title->setAlignment(Qt::AlignCenter);

    QLabel *versionLabel = new QLabel("версия 0.1", this);
    versionLabel->setGeometry(240, 270, 57, 20);
    versionLabel->setAlignment(Qt::AlignRight | Qt::AlignBottom);

    this->setStyleSheet("QWidget { background-color: #424242; }"
        "QPushButton { background-color: #B19CD9; color: white; font-weight:"
        "bold; }"
        "QLabel#title { color: white; font-size: 30px; font-weight: bold; }"
        "QLabel#versionLabel { color: white; font-size: 12px; }");

    connect(button1, &QPushButton::clicked, this, &MainWindow::openWindow1);
    // connect(button2, &QPushButton::clicked, this, &MainWindow::openWindow2);
```

```

}

MainWindow::~MainWindow() {}

void MainWindow::openWindow1()
{
    windowGame *window1 = new windowGame();
    window1->show();
    this->hide();
}

// void MainWindow::openWindow2()
// {
//     // windowNet *window2 = new windowNet();
//     // //window2->show();
//     // this->hide();
// }

```

## Приложение А.3

### Файл «mainwindow.h»

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QWidget>
#include <QPushButton>
#include <QGridLayout>
#include <QMainWindow>
#include <QLabel>
//#include "windowgame.h"

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void openWindow1();
    // void openWindow2();

private:
    QPushButton *button1;
    QPushButton *button2;
    QLabel *title;
    QLabel *versionLabel;
    QVBoxLayout *layout;

};
#endif // MAINWINDOW_H
```

## Приложение A.4

### Файл «gamewindow.cpp»

```
#include "windowgame.h"

windowGame::windowGame(QWidget *parent) : QMainWindow(parent)
{
    setFixedSize(340, 370);

    m_gameField = new GameField();
    m_helpField = new HelpField();

    connect(m_gameField, &GameField::ChangeTextSignal, m_helpField,
&HelpField::ChangeTextSlot);

    m_centralWidget = new QWidget();
    m_gridLayout = new QGridLayout();
    m_gridLayout->addWidget(m_gameField, 0, 0);
    m_gridLayout->addWidget(m_helpField, 1, 0);
    m_centralWidget->setLayout(m_gridLayout);
    setCentralWidget(m_centralWidget);
    setStyleSheet("background: rgb(90, 100, 112);");
}
```

## Приложение A.5

### Файл «gamewindow.h»

```
#ifndef WINDOWGAME_H
#define WINDOWGAME_H

#include <QWidget>
#include <QPushButton>
#include <QGridLayout>
#include <QMainWindow>
#include "game.h"
#include "helpfield.h"
#include <QLabel>
#include <QObject>
#include <QKeyEvent>

class windowGame : public QMainWindow
{
    Q_OBJECT
public:
    windowGame(QWidget *parent = nullptr);

private:
    GameField *m_gameField;
    HelpField *m_helpField;
    QWidget *m_centralWidget;
    QGridLayout *m_gridLayout;

    // protected:
    //     void keyPressEvent(QKeyEvent *event) override;
};

#endif // WINDOWGAME_H
```

## Приложение A.6

### Файл «game.h»

```
#ifndef GAME_H
#define GAME_H

#include <QWidget>
#include <QPainter>
#include <QList>
#include <QTimer>
#include <QKeyEvent>
#include <QRandomGenerator>

class SnakeItem;
class Snake;
class GameField;
class SnakeItem
{
    friend Snake;
    friend GameField;

public:
    SnakeItem(int x, int y);

private:
    int m_x;
    int m_y;
};

class Snake
{
    friend GameField;
```

```

public:
    Snake();

private:
    QList<SnakeItem*> m_snakeBody;
    int m_snakeBeginSize;
    enum SnakeDirection
    {
        up,
        left,
        down,
        right
    };
    SnakeDirection m_snakeDirection;
};

class GameField : public QWidget
{
    Q_OBJECT

public:
    GameField();

protected:
    void paintEvent(QPaintEvent *e) override;
    void keyPressEvent(QKeyEvent *e) override;

private:
    Snake *m_snake;

```

```

    SnakeItem *m_food;

    int m_snakeItemSize;

    QTimer *m_moveSnakeTimer;

    int m_fieldSize;

    int m_score;

    bool m_isPause;

    bool m_isMoveBlocked;

    bool m_isGameOver;

    void SetGameStatus();

    void GameOver();

    void StartNewGame();

    void CreateFood();


private slots:

    void MoveSnakeSlot();


signals:

    void ChangeTextSignal(QString text);

};


#endif // GAME_H

```

## Приложение А.7

### Файл «game.cpp»

```
#include "game.h"

GameField::GameField()
{
    setFixedSize(300, 300); //игровое поле
    setFocusPolicy(Qt::StrongFocus);
    m_snakeItemSize = 10;
    m_fieldSize = width()/m_snakeItemSize;
    StartNewGame();
}

void GameField::paintEvent(QPaintEvent *e)
{
    Q_UNUSED(e)

    QBrush gameFieldBrush(QColor(139, 144, 163), Qt::SolidPattern);
    QBrush snakeBrush(QColor(140, 206, 147), Qt::SolidPattern);
    QBrush foodBrush(QColor(247, 103, 123), Qt::SolidPattern);
    QPainter painter;
    painter.begin(this);
    if(m_isGameOver)
    {
        painter.setPen(QColor(250, 250, 250));
        painter.setFont(QFont("Arial", 15, 700));
        painter.drawText(QRect(0, 0, width(), height()), Qt::AlignCenter, "Вы
ПОГИБЛИ\nсчет : " + QString::number(m_score));
        return;
    }

    //отрисовка поля
    painter.setBrush(gameFieldBrush);
```

```

painter.setPen(QColor(50, 50, 50));

painter.drawRect(0, 0, width()-1, height()-1);

//отрисовка змейки

painter.setBrush(snakeBrush);

painter.setPen(QColor(0, 0, 0));

for(int i = 0; i < m_snake->m_snakeBody.size(); i++)
{
    painter.drawEllipse(m_snake->m_snakeBody[i]->m_x * m_snakeItemSize, m_snake->
m_snakeBody[i]->m_y * m_snakeItemSize, m_snakeItemSize, m_snakeItemSize);
}

//отрисовка еды

painter.setBrush(foodBrush);

painter.drawEllipse(m_food->m_x * m_snakeItemSize, m_food->m_y * m_snakeItemSize,
m_snakeItemSize, m_snakeItemSize);

painter.end();

m_isMoveBlocked = false;
}

void GameField::keyPressEvent(QKeyEvent *e)
{
    if(e->key () == Qt:: Key_Space)
    {
        if(m_isGameOver)
        {
            StartNewGame();

            return;
        }

        m_isPause = !m_isPause;

        SetGameStatus();
    }

    if(m_isMoveBlocked)
    {

```

```

        return;
    }

    if(e->key () == Qt:: Key_Left && m_snake->m_snakeDirection != Snake::
SnakeDirection::right)
    {
        m_snake->m_snakeDirection = Snake:: SnakeDirection::left;
    }

    if(e->key () == Qt:: Key_Right && m_snake->m_snakeDirection != Snake::
SnakeDirection::left)
    {
        m_snake->m_snakeDirection = Snake:: SnakeDirection::right;
    }

    if(e->key () == Qt:: Key_Down && m_snake->m_snakeDirection != Snake::
SnakeDirection::up)
    {
        m_snake->m_snakeDirection = Snake:: SnakeDirection::down;
    }

    if(e->key () == Qt:: Key_Up && m_snake->m_snakeDirection != Snake::
SnakeDirection::down)
    {
        m_snake->m_snakeDirection = Snake:: SnakeDirection::up;
    }

    m_isMoveBlocked = true;
}

void GameField::SetGameStatus()
{
    QString text;

    if(m_isPause)
    {
        m_moveSnakeTimer->stop();

        text = "счет : " + QString::number(m_score) + "\nпродолжить - ПРОБЕЛ";
    }
}

```

```

    }

    else

    {

        m_moveSnakeTimer->start(100);

        text = "счет : " + QString::number(m_score) + "\nпауза - ПРОБЕЛ";

    }

    emit ChangeTextSignal(text);
}

void GameField::GameOver()
{
    m_isGameOver = true;

    QString text = "Для возрождения нажмите ПРОБЕЛ";

    emit ChangeTextSignal(text);

    m_moveSnakeTimer->stop();

    delete m_snake;

    delete m_moveSnakeTimer;

    delete m_food;

}

void GameField::StartNewGame()
{
    m_isPause = false;

    m_isMoveBlocked = false;

    m_isGameOver = false;

    m_snake = new Snake();

    m_food = new SnakeItem(m_fieldSize/2,m_fieldSize/2);

    m_moveSnakeTimer = new QTimer();

    connect( m_moveSnakeTimer, &QTimer::timeout, this, &GameField::MoveSnakeSlot);

    m_moveSnakeTimer->start(100);

    m_score = 0;

    QString text = "счет : " + QString::number(m_score) + "\nпауза - ПРОБЕЛ";

```

```

        emit ChangeTextSignal(text);
    }
    void GameField::CreateFood()
    {
        m_food->m_x = QRandomGenerator::global()->bounded(0, m_fieldSize - 1);
        m_food->m_y = QRandomGenerator::global()->bounded(0, m_fieldSize - 1);
        // проверка совпадения координат змейки и еды
        for(int i = 0; i < m_snake->m_snakeBody.size(); i++)
        {
            if(m_food->m_x == m_snake->m_snakeBody[i]->m_x && m_food->m_y == m_snake-
>m_snakeBody[i]->m_y)
            {
                return CreateFood();
            }
        }
    }
    void GameField::MoveSnakeSlot()
    {
        SnakeItem *newSnakeItem;
        if(m_snake->m_snakeDirection == Snake::SnakeDirection::right)
        {
            newSnakeItem = new SnakeItem(m_snake->m_snakeBody[0]->m_x + 1, m_snake-
>m_snakeBody[0]->m_y);
        }
        else if(m_snake->m_snakeDirection == Snake::SnakeDirection::left)
        {
            newSnakeItem = new SnakeItem(m_snake->m_snakeBody[0]->m_x - 1, m_snake-
>m_snakeBody[0]->m_y);
        }
        else if(m_snake->m_snakeDirection == Snake::SnakeDirection::up)
        {

```

```

        newSnakeItem = new SnakeItem(m_snake->m_snakeBody[0]->m_x, m_snake-
>m_snakeBody[0]->m_y - 1);
    }
    else
    {
        newSnakeItem = new SnakeItem(m_snake->m_snakeBody[0]->m_x, m_snake-
>m_snakeBody[0]->m_y + 1);
    }
    // ограничение поля
    if(newSnakeItem->m_x >= m_fieldSize)
    {
        newSnakeItem->m_x = 0;
    }
    else if(newSnakeItem->m_x < 0)
    {
        newSnakeItem->m_x = m_fieldSize - 1;
    }
    else if(newSnakeItem->m_y < 0)
    {
        newSnakeItem->m_y = m_fieldSize - 1;
    }
    else if(newSnakeItem->m_y >= m_fieldSize)
    {
        newSnakeItem->m_y = 0;
    }
    // СМЕРТЬ
    for(int i = 1; i< m_snake->m_snakeBody.size(); i++)
    {
        if(m_snake->m_snakeBody[0]->m_x == m_snake->m_snakeBody[i]->m_x && m_snake-
>m_snakeBody[0]->m_y == m_snake->m_snakeBody[i]->m_y)
        {

```

```

        GameOver();
    }
}

if(newSnakeItem->m_x == m_food->m_x && newSnakeItem->m_y == m_food->m_y)
{
    m_score++;

    CreateFood();

    QString text = "счет : " + QString::number(m_score) + "\n\nпауза - Пробел";
    emit ChangeTextSignal(text);
}

else
{
    m_snake->m_snakeBody.removeLast();
}

m_snake->m_snakeBody.insert(0, newSnakeItem);

repaint();
}

SnakeItem::SnakeItem(int x, int y)
{
    m_x = x;
    m_y = y;
}

Snake::Snake()
{
    m_snakeBeginSize = 4;
    for(int i = 0; i< m_snakeBeginSize; i++)
    {
        m_snakeBody.insert(0, new SnakeItem(i, 0));
    }

    m_snakeDirection = SnakeDirection::down;
}

```

## Приложение А.8

### Файл «helpfield.cpp»

```
#include "helpfield.h"

HelpField::HelpField()
{
    setFixedSize(300, 50); //размер;
    m_text = "счет : 0\nпауза - ПРОБЕЛ";
}

void HelpField::paintEvent(QPaintEvent *e)
{
    Q_UNUSED(e)
    QPainter painter;
    painter.begin(this);
    painter.setPen(QColor(250, 250, 250));
    painter.setFont(QFont("Arial", 10, 700));
    painter.drawText(QRect(0, 0, width(), height()), Qt::AlignCenter, m_text);
    painter.end();
}

void HelpField::ChangeTextSlot(QString text)
{
    m_text = text;
    repaint();
}
```

## Приложение А.9

### Файл «helpfield.h»

```
#ifndef HELPFIELD_H
#define HELPFIELD_H

#include <QWidget>
#include <QPainter>

class HelpField : public QWidget
{
public:
    HelpField();

protected:
    void paintEvent(QPaintEvent *e) override;

private:
    QString m_text;

public slots:
    void ChangeTextSlot(QString text);

};

#endif // HELPFIELD_H
```

## **Приложение Б**

### **Снимки экрана**

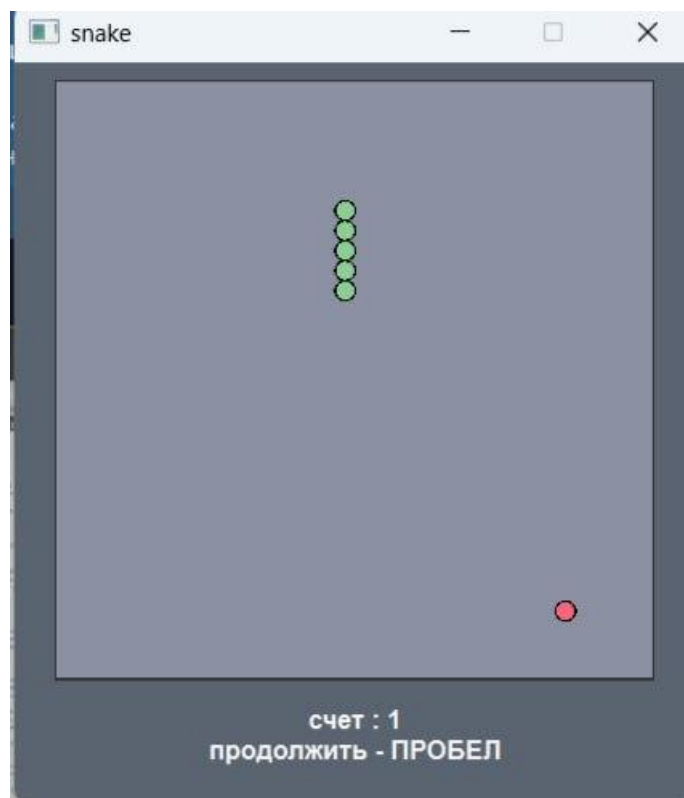


Рисунок 10 — окно паузы



Рисунок 11 — окно Смерти

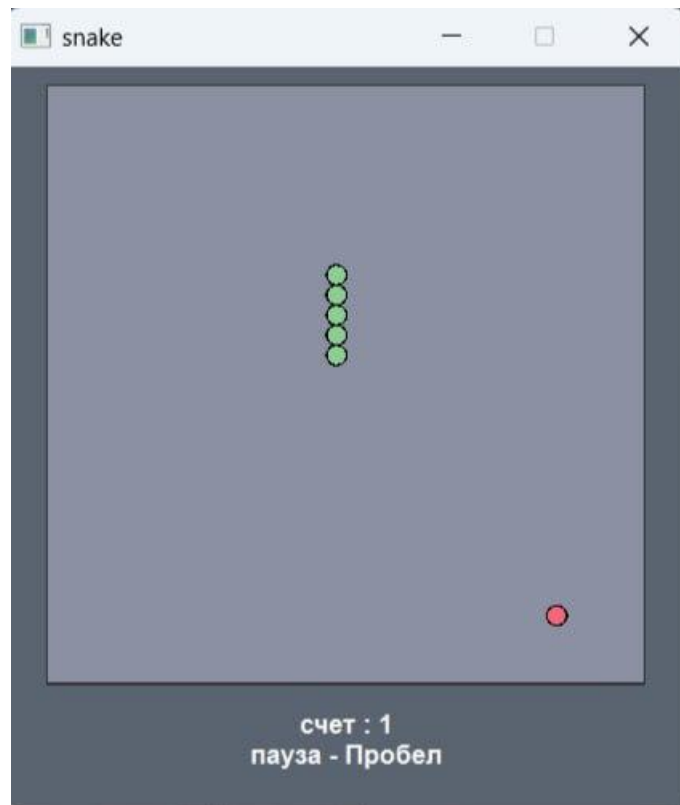


Рисунок 12 — окно «Игры»

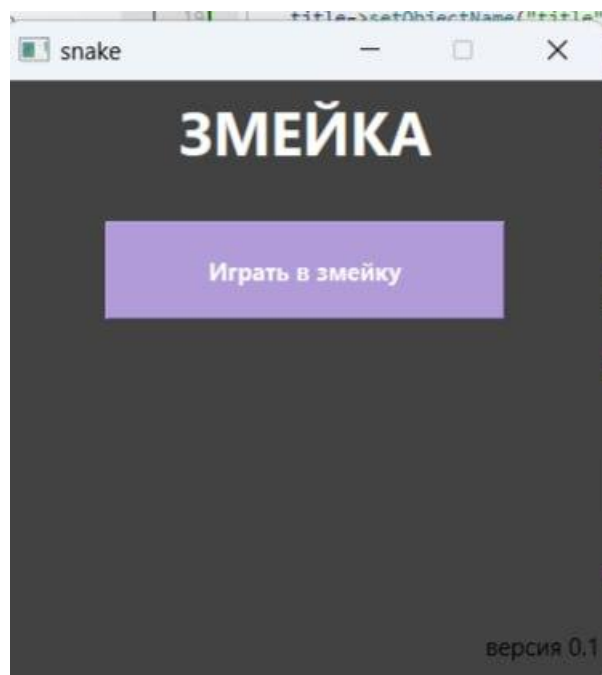


Рисунок 13 — Главное окно