**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](#_Toc155805001)

[1 Постановка задачи 3](#_Toc155805002)

[2 Описание иерархии классов 4](#_Toc155805003)

[2.1 Интерфейс IObject 5](#_Toc155805004)

[2.2 Класс FruitBase 5](#_Toc155805005)

[2.3 Класс Apple 5](#_Toc155805006)

[2.4 Класс Banana 6](#_Toc155805007)

[2.5 Класс Pear 6](#_Toc155805008)

[2.6 Класс CounterBase 6](#_Toc155805009)

[2.7 Класс CounterApple 7](#_Toc155805010)

[2.8 Класс CounterBanana 7](#_Toc155805011)

[2.9 Класс CounterPear 7](#_Toc155805012)

[2.10 Класс Form1 8](#_Toc155805013)

[2.11 Класс Screensaver 10](#_Toc155805014)

[2.12 Класс MainMenu 10](#_Toc155805015)

[2.13 Класс Description 11](#_Toc155805016)

[2.14 Класс InputName 11](#_Toc155805017)

[2.15 Класс RecordsView 11](#_Toc155805018)

[2.16 Класс ResultGame 13](#_Toc155805019)

[2.17 Демонстрация иерархий классов 13](#_Toc155805020)

[3 Использованные мультимедийные ресурсы и сторонние библиотеки 14](#_Toc155805021)

[4 Демонстрация работы 18](#_Toc155805022)

[ЗАКЛЮЧЕНИЕ 23](#_Toc155805023)

ВВЕДЕНИЕ

Установлено, что компьютерные игры оказывают значительное влияние на мировосприятие, мироощущение, миропонимание современного человека, становятся не только объектами культурного досуга, но и новой профессиональной деятельностью.

С одной стороны, компьютерные игры являются программами, с другой стороны - видеоиграми, с третьей - жанром искусства. Полностью проанализировать данное явление достаточно сложно в силу того, что каждый день появляются новые игры и игровые сообщества. Вектор развития игровой индустрии постоянно меняется в связи с возникновением новых игровых жанров и становится сегодня точкой бифуркации для определения роли и места компьютерных игр в истории человечества [1].

Целью курсовой работы является разработка интерактивного приложения с графическим пользовательским интерфейсом с использованием фреймворка .Net и технологии Windows Forms.

Для достижения поставленной цели необходимо решить следующие задачи:

* описать основные требования к разрабатываемому приложению;
* описать правила игры;
* разработать приложение;
* продемонстрировать работоспособность приложения;
* cделать выводы о проделанной работе.

1. Постановка задачи

В рамках курсовой работы необходимо разработать игру «Змейка».

На экране представлено поле, которое поделено на клетки. Пользователю нужно перемещаться по клеткам и собирать фрукты.

Правила игры: Управляя змейкой пользователь должен собирать фрукты (яблоко, банан, груша). После каждого съеденного фрукта длина змейки увеличивается на одну клетку поля карты. В случае, если змейка съедает более 25 фруктов её скорость передвижения по карте увеличивается в два раза. Также, в случае, если змейка съест более 40 фруктов её скорость возрастёт ещё на 20% от текущей скорости передвижения. При выходе за границы карты, а также при попадании змейки на клетку поля карты, где располагается её тело игра завершается. По итогу игроку нужно управляя змейкой съесть как можно больше фруктов. Количество съеденных змейкой фруктов (для каждого вида фрукта отдельный счетчик) указывается в левом верхнем углу экрана.

В рамках игры должно быть реализовано меню позволяющее игроку:

* + 1. Запустить игровой процесс;
    2. посмотреть управление и описание игры;
    3. сменить имя текущего игрока;
    4. просмотреть до 10 лучших результатов игры, которые записываются после каждой сыгранной игры, хранятся в бинарном файле и выводятся на экран по запросу пользователя (результаты проведённых игр можно как стереть с бинарного файла так и переписать 10 лучших результатов в текстовый файл).

Итак, в результате были сформированы требования, согласно которым будет разработано приложение.

1. Описание иерархии классов

В процессе разработки приложения были созданы следующие классы:

* Form1 – класс, описывающий основную логику игры;
* IObject – интерфейс, содержащий общие поля для абстрактных классов FruitBase и CounterBase;
* FruitBase – абстрактный класс, наследующий интерфейс IObject и описывающий логику расположения фруктов на клетках карты;
* Apple – дочерний класс базового класса FruitBase, для управления поведением объекта яблока;
* Banana – дочерний класс базового класса FruitBase, для управления поведением объекта банана;
* Pear – дочерний класс базового класса FruitBase, для управления поведением объекта груши;
* CounterBase – абстрактный класс, наследующий интерфейс IObject и описывающий счётчики для фруктов;
* CounterApple – дочерний класс базового класса CounterBase, описывающий счётчик для яблок;
* CounterBanana – дочерний класс базового класса CounterBase, описывающий счётчик для бананов;
* CounterPear – дочерний класс базового класса CounterBase, описывающий счётчик для груш;
* Screensaver – класс, описывающий заставку игры;
* MainMenu – класс, описывающий главное меню игры;
* Description – класс, описывающий страницу описания цели игры и её управления;
* InputName – класс, описывающий страницу установки или смены имени игрока;
* RecordsView – класс, описывающий страницу для просмотра результатов игры и их управлением;
* ResultGame – класс, описывающий страницу для просмотра результатов сыгранной игры;

Далее будет приведено описание членов каждого из этих классов и интерфейса.

2.1 Интерфейс IObject

Данный интерфейс требует реализации трёх свойств:

* PictureBox pictureBox – определяет изображение с которым объект будет отображаться в окне;
* int pX – определяет координату позиции по x;
* int pY – определяет координату позиции по y.

2.2 Класс FruitBase

Данный класс реализует интерфейс IObject.

Данный класс содержит следующие приватное статическое поле: Random random – данное свойство используется для рандомизации позиции объекта фрукта на карте.

Также в данном классе есть защищённый конструктор:

FruitBase(int width, int height, int sof) – конструктор содержащий вызов метода генерации рандомной позиции фрукта на карте (generateFruit()). В конструктор передаются 3 переменные типа int (width, height, sof) размеры окна по ширине и высоту и размер клетки карты соответсвенно. Эти переменные нужны для корректного расчета рандомной позиции на карте для объекта.

В данном классе объявлен защищённый абстрактный метод InitializePictureBox() для задания изображения и размеров объекта.

Также, в данном классе реализован защищённый метод void generateFruit(), который отвечает за логику рандомного размещения объекта фрукта на карте.

2.3 Класс Apple

Данный класс является дочерним классом базового класса FruitBase.

В этом классе есть конструктор:

public Apple(int width, int height, int sof) : base(width, height, sof) конструктор, отвечающий за вызов переопределённого метода инициализации объекта фрукта яблока.

Также в этом классе есть переопределённый защищённый метод:

void InitializePictureBox() – метод для инициализации объекта фрукта яблока.

**2.4 Класс Banana**

Данный класс является дочерним классом базового класса FruitBase.

В этом классе есть конструктор:

public Banana(int width, int height, int sof) : base(width, height, sof) конструктор, отвечающий за вызов переопределённого метода инициализации объекта фрукта банана.

Также в этом классе есть переопределённый защищённый метод:

void InitializePictureBox() – метод для инициализации объекта фрукта банана.

**2.5 Класс Pear**

Данный класс является дочерним классом базового класса FruitBase.

В этом классе есть конструктор:

public Pear(int width, int height, int sof) : base(width, height, sof) конструктор, отвечающий за вызов переопределённого метода инициализации объекта фрукта груши.

Также в этом классе есть переопределённый защищённый метод:

void InitializePictureBox() – метод для инициализации объекта груши.

2.6 Класс CounterBase

Данный класс реализует интерфейс IObject.

Данный класс содержит следующие публичные поля:

* Label count – поле для отображения счёта по фрукту на экране;
* int cnt – поле для хранения счёта по фрукту.

Также, данный класс содержит конструктор:

public CounterBase(int width, int height, int sof) – конструктор с переданными параметрами типа int (width, height, sof), хранящими значения размеров ширины и высоты экрана, и значение хранящее размер клетки карты.

Данный класс содержит объявление абстрактного защищённого метода:

InitializeCounter() – метод для инициализации счетчика фрукта.

2.7 Класс CounterApple

Данный класс является классом наследником базового класса CounterBase.

В этом классе есть конструктор:

public CounterApple(int width, int height, int sof) : base(width, height, sof) конструктор нужен для передачи параметров, необходимых для расчета позиции размещения счетчика в окне. В конструкторе вызывается функция InitializeCounter().

Также, в данном классе переопределяется защищённый метод:

InitializeCounter() – метод для инициализации счетчика фрукта яблока.

2.8 Класс CounterBanana

Данный класс является классом наследником базового класса CounterBase.

В этом классе есть конструктор:

public CounterBanana(int width, int height, int sof) : base(width, height, sof) конструктор нужен для передачи параметров, необходимых для расчета позиции размещения счетчика в окне. В конструкторе вызывается функция InitializeCounter().

Также, в данном классе переопределяется защищённый метод:

InitializeCounter() – метод для инициализации счетчика фрукта банана.

2.9 Класс CounterPear

Данный класс является классом наследником базового класса CounterBase.

В этом классе есть конструктор:

public CounterPear(int width, int height, int sof) : base(width, height, sof) конструктор нужен для передачи параметров, необходимых для расчета позиции размещения счетчика в окне. В конструкторе вызывается функция InitializeCounter().

Также, в данном классе переопределяется защищённый метод:

InitializeCounter() – метод для инициализации счетчика фрукта груши.

2.10 Класс Form1

Данный класс содержит следующие приватные поля:

* static int WIDTH – статичное поле для хранения параметра ширины экрана (начальное значение 600);
* static int HEIGHT – статическое поле для хранения параметра высоты экрана (начальное значение 600);
* private static int sizeOneFields – статическое поле для хранения размера клетки карты;
* List<PictureBox> snake\_array – поле динамический массив для хранения сегментов змейки;
* int dirX – поле для хранения значения определяющего направление движения змейки, а также расположения её головы;
* int dirY – поле для хранения значения определяющего направление движения змейки, а также расположения её головы;
* Image currImage – поле для хранения текущего изображения головы змейки;
* Timer timer – поле для таймера, регулирует скорость передвижения змейки;
* int score – поле для хранения суммы съеденных фруктов, переменная необходимая для регулировки длины змейки (начальное значение 0);
* bool isPause – булевое поле, которое отвечает за регулирование постановки игры на паузу;
* Label textPause – поле для хранения надписи, которая появляется при постановке игры на паузу;
* MainMenu menu – поле хранящее экземпляр класса MainMenu, поле необходимо для регулирования окон игрового процесса и окна меню;
* string username – поле для хранения имени игрока;
* const string filename – константное поле для хранения пути файла куда будут сохраняться результаты игр (значение “game\_result.dat”);
* Apple apple – поле для объекта класса Apple;
* Banana banana – поле для объекта класса Banana;
* Pear pear – поле для объекта класса Pear;
* CounterApple cntApple – поле для объекта класса CounterApple;
* CounterBanana cntBanana – поле для объекта класса CounterBanana;
* CounterPear cntPear – поле для хранения объекта класса CounterPear.

В данном классе присутствует конструктор:

Form1(MainMenu menu, string name) – конструктор инициализирует игровой процесс на форме.

Также, в данном классе присутствуют определения приватных методов:

* void InitializeTimer() – метод, инициализирующий работу таймера;
* void textPause\_Paint(object sender, PaintEventArgs e) – метод для отрисовки границ окна паузы;
* void InitializeTextMenu() – метод для инициализации текста сообщения, появляющегося в окне паузы;
* bool checkBorder() – метод проверки выхода змейки за пределы карты;
* bool eatSelf() – метод для проверки попадания на клетку карты, в которой в данный момент находится тело змейки;
* void eatFruits() – метод для поедания фруктов, в котором происходит создание нового сегмента змейки после поедания ею фрукта, а также в этом методе происходит подсчет съеденных фруктов;
* void moveSnake() – метод для реализации логики движения змейки;
* void update(Object myObject, EventArgs eventArgs) – метод обновления кадра на окне;
* void SaveResult() – метод для сохранения результата игры в бинарный файл;
* Form1\_Paint(object sender, PaintEventArgs e) – метод для рисования клеток на карте;
* void geterationMap() – метод для отрисовки карты;
* void Move(object sender, KeyEventArgs e) – метод для отслеживания событий нажатия на клавиши, чтобы управлять змейкой;
* void InitCounters() – метод для инициализации счетчиков на окне;
* void InitializeFruits() – метод для инициализации объектов фруктов на карте.

2.11 Класс Screensaver

В данном классе реализован конструктор:

Screensaver() – конструктор для инициализации окна заставки игры.

А также в классе присутствует определение приватного асинхронного метода:

void ShowInputNameFormAsync() – метод для показа заставки и перехода к окну ввода имени.

2.12 Класс MainMenu

Данный класс содержит одно приватное поле:

string username – поле для хранения имени игрока;

В данном классе реализован конструктор:

MainMenu(string username) – конструктор для инициализации окна меню и сохранения имени игрока.

Также, в данном классе присутствует определение следующих приватных методов:

* void StartGameButton\_Click(object sender, EventArgs e) – метод для срабатывания события нажатия на кнопку, начинающую игровой процесс;
* void buttonDesc\_Click(object sender, EventArgs e) – метод для срабатывания события нажатия на кнопку открытия окна описания игры;
* void ChangeName\_Click(object sender, EventArgs e) – метод для срабатывания события нажатия на кнопку открытия окна смены имени игрока;
* void RecordsButton\_Click(object sender, EventArgs e) – метод для срабатывания события нажатия на кнопку просмотра окна рекордов;
* void buttonQuit\_Click(object sender, EventArgs e) – метод для срабатывания события нажатия на кнопку выхода из игры.

2.13 Класс Description

В данном классе содержится одно приватное поле:

MainMenu mainMenu – поле для хранения экземпляра класса MainMenu, необходимое для возврата на окно главного меню

В данном классе реализован конструктор:

Description(MainMenu menu) – конструктор для инициализации окна описания игры.

Также, в данном классе реализован приватный метод:

void backToMenu\_desc\_Click(object sender, EventArgs e) – метод для срабатывания события возврата в главное меню.

2.14 Класс InputName

В данном классе содержится одно приватное константное поле:

int MaxNameLength – поле для хранения максимальной длины имени игрока (значение 10).

В данном классе реализован конструктор:

InputName() – конструктор для инициализации окна ввода имени.

Также, в данном классе присутствует реализация приватного метода:

void Submit\_Click(object sender, EventArgs e) – метод для срабатывания события сохранения имени игрока.

2.15 Класс RecordsView

Данный класс содержит приватную структуру:

struct Records – структура, хранящая в себе публичные поля:

* + string name – поле для хранения имени игрока;
  + int cntApple – поле для хранения количества съеденных яблок;
  + int cntBanana – поле для хранения количества съеденных бананов;
  + int cntPear – поле для хранения количества съеденных груш;
  + int all – поле для хранения общего количесва съеденных фруктов.

В данном классе имеются следующие приватные поля:

* string filename – поле для хранения пути к бинарному файлу, в котором хранится запись сыгранных игр (начальное значение “game\_result.dat”);
* string txtfile – поле для хранения пути к текстовому файлу, в который будет происходить запись рекордов по запросу пользователя (начальное значение “ records.txt ”);
* List<Records> records – поле динамический массив для хранения записей результатов игр и работы с ними.
* MainMenu menu – поле для хранения экземпляра MainMenu.

В данном классе имеется реализация конструктора:

* RecordsView(MainMenu menu) – конструктор для инициализации окна рекордов и для вызова основных методов отвечающих за отображение рекордов на окне.

Также в данном классе реализованы следующие приватные методы:

* ReadBinFileRecords() – метод для чтения бинарного файла и записи результатов в динамический массив;
* void SortByBetterResult() – метод для сортировки рекордов от лучше результата к худшему;
* void WriteRecordsInForm() – метод отображения 10 лучших рекордов;
* void backToMenu\_records\_Click(object sender, EventArgs e) – метод для срабатывания события при нажатии на кнопку возвращения на главное меню;
* void DeleteRecords\_Click(object sender, EventArgs e) – метод для срабатывания события при нажатии на кнопку удаления рекордов;
* void ConvertTotxt\_Click(object sender, EventArgs e) – метод для срабатывания события при нажатии на кнопку отправки результатов в текстовый файл.

2.16 Класс ResultGame

В данном классе содержится одно приватное поле:

MainMenu mainMenu – поле для хранения экземпляра класса MainMenu;

В данном классе реализован конструктор:

ResultGame(int cntApple, int cntBanana, int cntPear, MainMenu menu, string name) – конструктор для инициализации окна результатов игры и записи в подготовленные Label поля информации по прошедшей игре.

Также, в данном классе присутствует следующий приватный метод:

void backToMenu\_result\_Click(object sender, EventArgs e) – метод для срабатывания события при нажатии на кнопку возвращения в главное меню.

2.17 Демонстрация иерархий классов

Разработанные иерархии наглядно продемонстрирована на диаграммах классов, приведённых на рисунках 1, 2, 3.

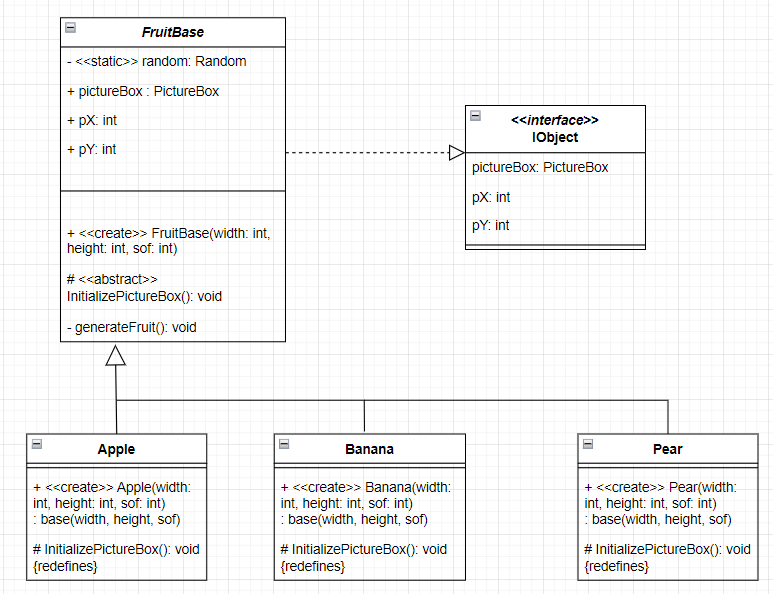
**

Рисунок 1 – Диаграмма классов иерархии «Фрукты»

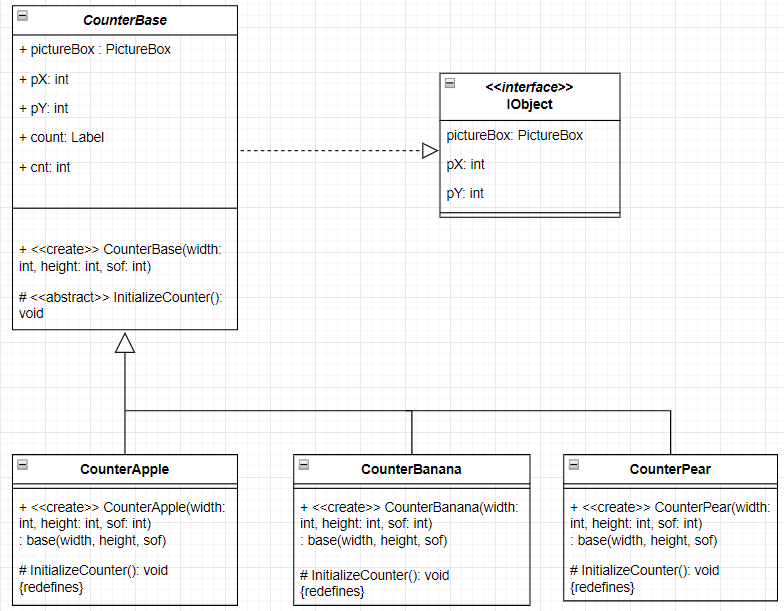


Рисунок 2 – Диаграмма классов иерархии «Счетчики»

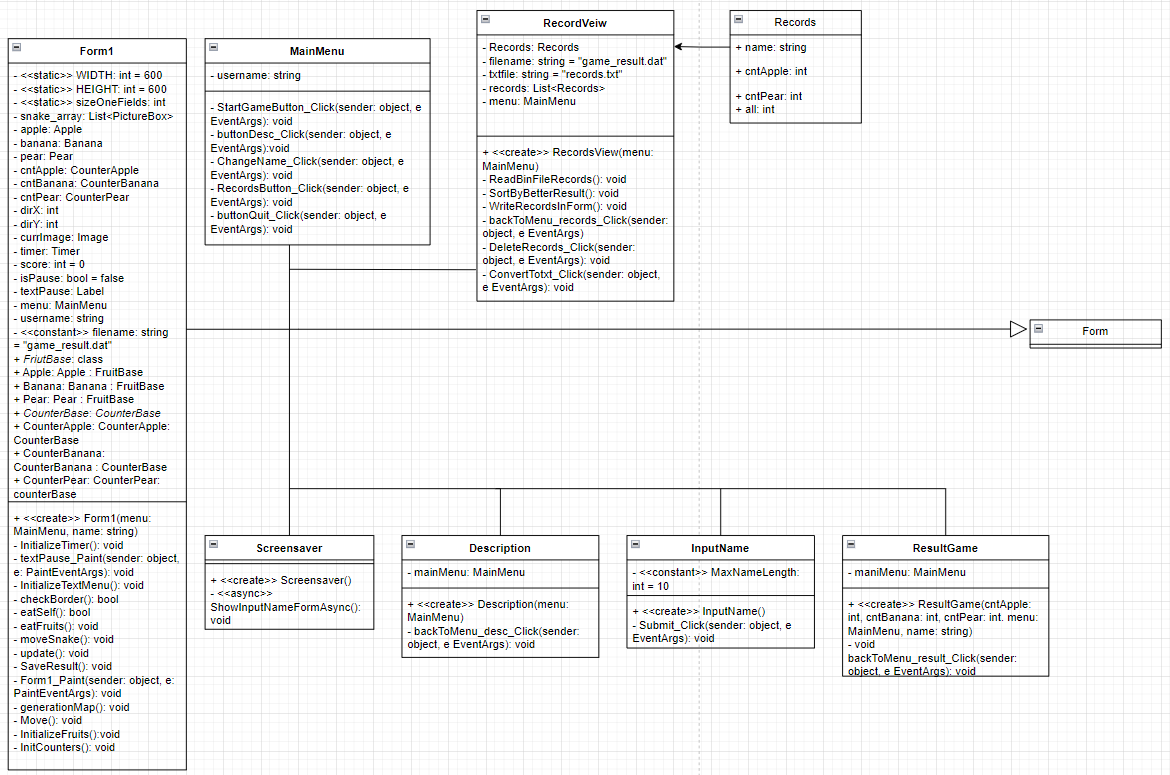


Рисунок 3 – Диаграмма классов иерархии форм

Разработанная иерархия классов позволит реализовать поставленную задачу в соответствии со сформулированными требованиями.

1. Использованные мультимедийные ресурсы и сторонние библиотеки

В курсовой работе применялся шрифт MS UI Gothic:



Рисунок 4 – Шрифт

Также в курсовой работе были использованы цвета:

* Цвет шрифта: ControlText;
* цвет шрифта: ButtonHighlight;
* цвет фона кнопок: rgb(192, 255, 192).

Изображения головы змейки для движений в разные стороны, приведенные на рисунках 5 – 8.



Рисунок 5 – Голова змейки «вверх»



Рисунок 6 – Голова змейки «влево»



Рисунок 7 – Голова змейки «вправо»



Рисунок 8 – Голова змейки «вниз»

Изображение фона для заставки, приведенное на рисунке 9.



Рисунок 9 – Фоновое изображение для заставки

Изображение для фона меню и подпунктов меню, приведенное на рисунке 10.



Рисунок 10 – Фон для меню и его подпунктов

Изображение фона игрового процесса, приведенное на рисунке 11.

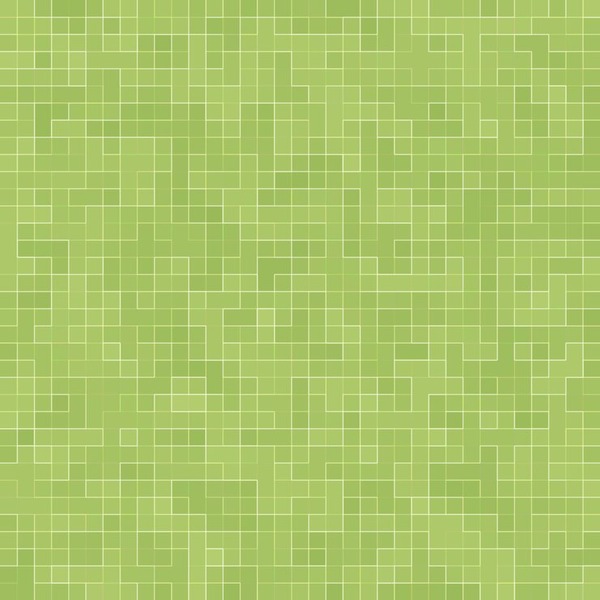


Рисунок 11 – Фон для игрового процесса

Также, в курсовой работе использовались следующие пространства имен:

* System.Collections.Generic – cодержит интерфейсы и классы, определяющие универсальные коллекции, которые позволяют пользователям создавать строго типизированные коллекции, обеспечивающие повышенную производительность и безопасность типов по сравнению с не универсальными строго типизированными коллекциями [2];
* System.Drawing – Предоставляет доступ к основным графическим функциям GDI+ [3];
* System.IO – cодержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок [4];
* System.Windows.Forms – cодержит классы для создания приложений Windows, которые позволяют наиболее эффективно использовать расширенные возможности пользовательского интерфейса, доступные в операционной системе Microsoft Windows [5];
* System.LINQ – Предоставляет классы и интерфейсы, поддерживающие запросы с использованием LINQ [6].

Использование приведённых ресурсов и библиотек позволит упростить процесс разработки приложения.

1. Демонстрация работы

При запуске приложения пользователю демонстрируется заставка на рисунке 12:



Рисунок 12 – Заставка игры

После этого пользователь должен ввести имя с помощью клавиатуры, изображение представлено на рисунке 13. Это имя будет сохранено вместе с набранными в игре баллами в списке рекордов.



Рисунок 13 – Ввод имени игрока

Затем пользователь попадает в главное меню, изображение меню представлено на рисунке 14. В меню пользователю предоставляется возможность выбора одного из пяти пунктов: «Играть», «Описание», «Смена имени», «Рекорды», «Выйти».



Рисунок 14 – Меню игры

Если пользователь выберет первый пункт меню, то есть «Играть», то начнется игровой процесс. Игровой процесс представлен на рисунке 15:

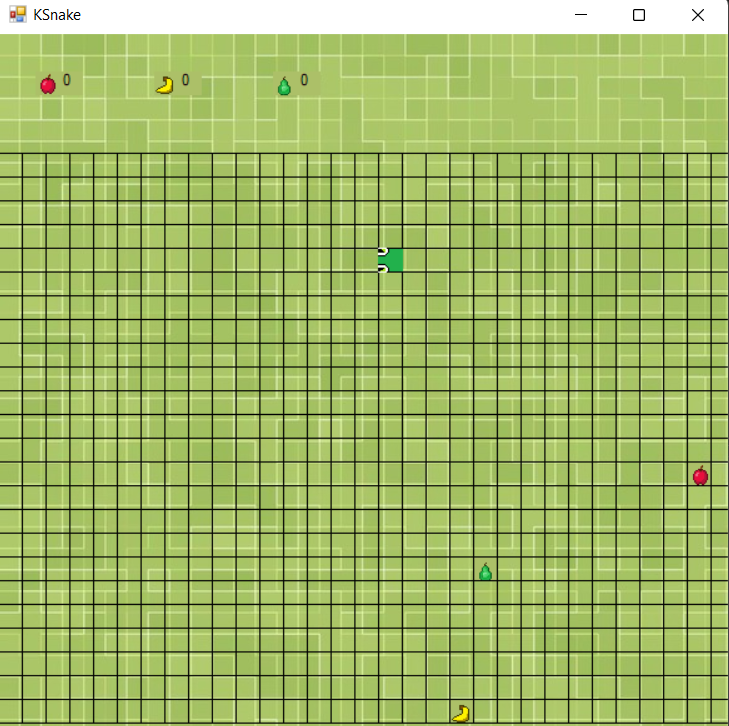


Рисунок 15 – Игровой процесс

Нажав на клавишу ESC можно поставить игру на паузу, для возобновления игры нужно заново нажать на клавишу ESC. Пауза представлена на рисунке 16.

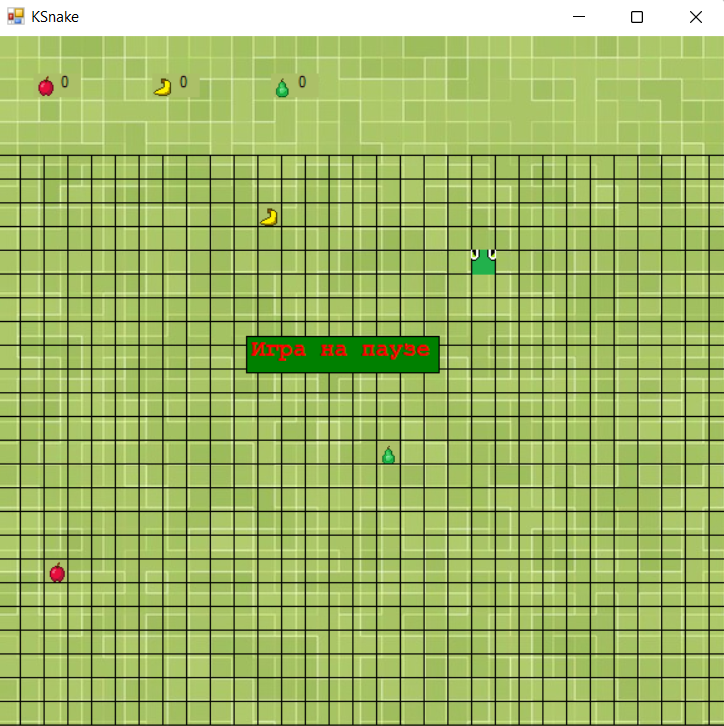


Рисунок 16 – Пауза

По окончании игрового процесса появляется окно демонстрирующее результат за прошедшую игру, окно представлено на рисунке 17. При нажатии на кнопу «В меню» производится возвращение к главному меню игры.

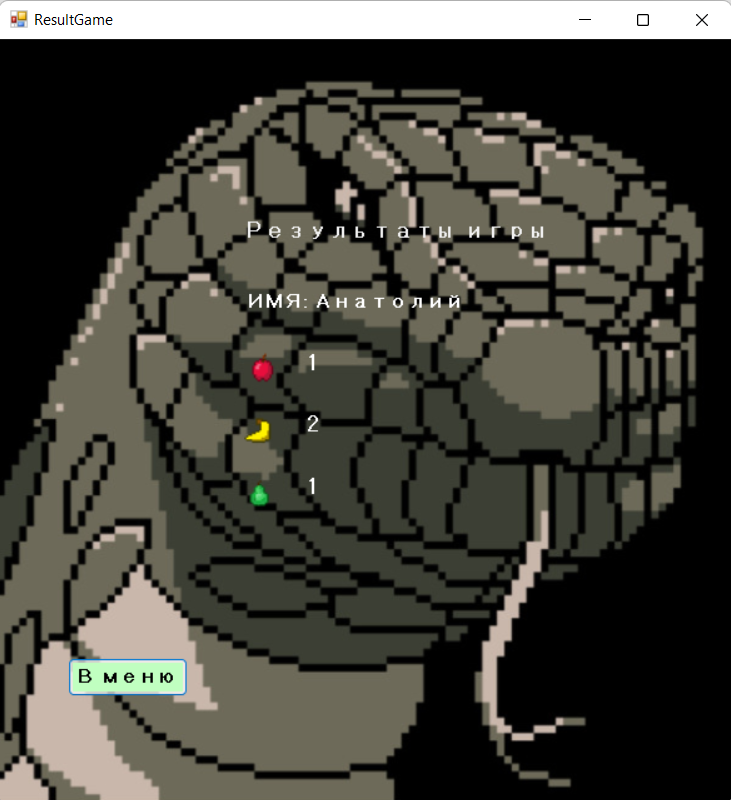


Рисунок 17 – Результаты игры

Если выбрать пункт меню «Описание», то отобразится текст, касающийся управления и описания игры. Описание игры представлено на рисунке 18. При нажатии на кнопу «В меню» производится возвращение к главному меню игры.

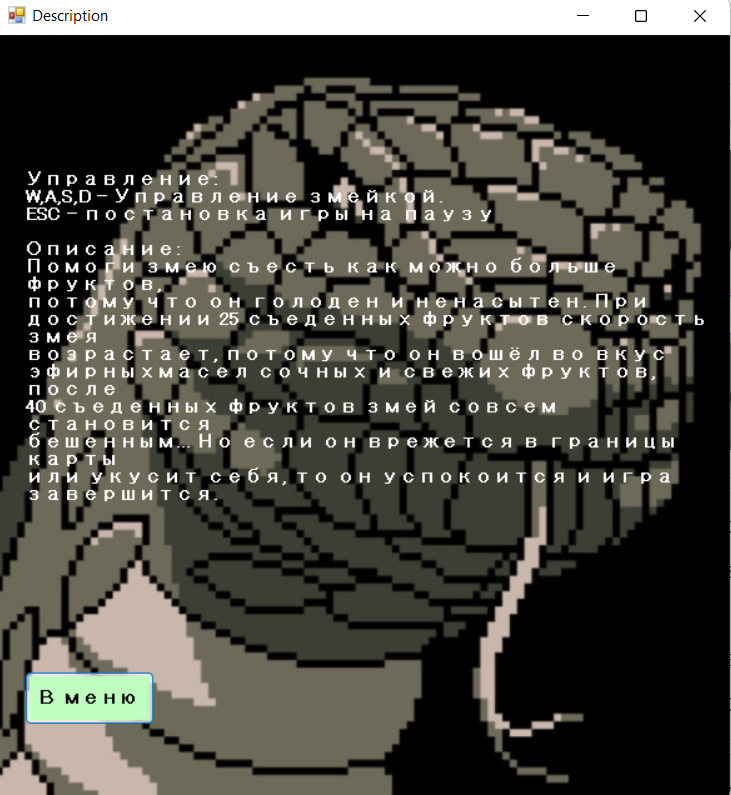


Рисунок 18 – описание игры

Если пользователь нажмет на пункт меню «Рекорды», то отобразится таблица лучших результатов игры, таблица представлена на рисунке 19. Кнопка «В меню» возвращает пользователя в главное меню игры, кнопка «Удалить результаты» удаляет все сохранённые записи по результатам игр, кнопка «Рекорды->.txt» информация по рекордам будет направлена в текстовый файл.



Рисунок 19 – Демонстрация рекордов

Зайдя повторно в пункт «Рекорды» после удаления рекордов, никаких рекордов не отображается, что представлено на рисунке 20.



Рисунок 20 – Список рекордов пуст

При нажатии пункта меню «Смена имени» вновь появляется окно (Рисунок 13), в котором будет предложено сменить имя игрока.

При нажатии на «Выйти» приложение будет закрыто.

Демонстрация возможностей приложения подтверждает его соответствие выдвинутым требованиям и поставленной задаче.

ЗАКЛЮЧЕНИЕ

В рамках выполнения курсовой работы по дисциплине "Информационные технологии и программирование" была успешно разработана игра "Змейка". Одним из важных достижений работы является создание дружественного интерфейса, который обеспечивает комфортное взаимодействие пользователя с игрой. Кроме того, были разработаны обоснованные иерархии классов, которые соответствуют принципам объектно-ориентированного программирования.

Разработка игры "Змейка" стала ценным опытом, который предоставил хорошую практику работы с языком программирования C# и библиотекой для создания графического интерфейса пользователя Windows Forms.

Все задачи выполнены, поставленная цель данной курсовой работы достигнута.