**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ**

**ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ**

**ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ**

**«РЕАЛЬНАЯ ШКОЛА»**



**Отдел\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Программа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КУРСОВАЯ РАБОТА**

**(Тема курсовой работы)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись) (Дата) (Фамилия, инициалы)

**Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись) (Дата) (Фамилия, инициалы)

Воронеж 2022

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ КУРСОВОЙ РАБОТЫ**

Учащемуся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, инициалы) (Группа)

1 Тема курсовой работы\_\_\_\_

утверждена приказом

по МБУДО ЦДО «Реальная школа» № \_\_\_\_\_\_\_\_\_\_\_от

2 Срок представления к защите

3 Особенности задания

4 Содержание пояснительной записки (перечень вопросов, подлежащих разработке)

4.1

4.2 4.3

4.4

4.5

5 Дата выдачи задания на выполнение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (Инициалы, фамилия)

Замечания руководителя

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc101183355)

[**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ** 6](#_Toc101183355)

[Язык программирования C#, и история его создания 6](#_Toc101183355)

[Платформа Windows Forms 9](#_Toc101183355)

[**ПРАКТИЧЕСКАЯ ЧАСТЬ** 12](#_Toc101183355)

[Моделирование интерфейса 12](#_Toc101183355)

[Листинг программы 13](#_Toc101183355)

1. [Form2 13](#_Toc101183355)
2. [Form1 14](#_Toc101183355)
3. [Form3 25](#_Toc101183355)

[Инструкция пользователя 26](#_Toc101183355)

[Тестовый пример 27](#_Toc101183355)

[Актуальность выбранной темы 30](#_Toc101183355)

[**ЗАКЛЮЧЕНИЕ** 31](#_Toc101183355)

[**СПИСОК ЛИТЕРАТУРЫ** 32](#_Toc101183355)

**ВВЕДЕНИЕ**

Игра традиционно считается детским занятиям, она готовят ребёнка к взрослой жизни и без них невозможно нормальное развитие личности. Это школа жизни, общения, взаимодействия. В жизни взрослых людей игра также играет важную роль. Она заполняет досуг, восполняет потребность в отдыхе и разрядке, дает эмоции и новые впечатления, позволяет выйти за пределы обыденной жизни и окунуться в иной, игровой мир, где царят свои правила и свой порядок. Будучи однажды сыгранной, она остается в памяти как некое творение или ценность, передается далее, как традиция, в тоже время игра всегда уникальна, невозможно в точности повторить одну и ту же игру дважды. Игра позволяет нам моделировать события, пробовать проявлять себя в безопасной, комфортной обстановке.  
Чтобы игра осуществилась, нам всегда нужен кто-то или что-то, с кем (чем) ведется взаимодействие, кто (что) будет отвечать встречным ходом на ход игрока. В игре всегда есть элемент неожиданности, риск, напряжение, возможность нескольких исходов. Именно это придает игре особое очарование и позволяет личности развиваться творчески. Можно сказать, что игра – это один из основных видов человеческой деятельности и её элементы в том или ином виде присутствуют на каждом этапе развития личности.

Приложением является компьютерная игра, написанная на языке C#. В этой игре целью является уберечь муравьев, которыми управляет пользователь, от лягушек как можно дольше. Если же пользователь поставил максимальное время удержания муравьев живыми, рекорд автоматически сохраняется в этой игре.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**Язык программирования C#, и история его создания**

Тест выполнен на языке программирования C#, так как он имеет ряд достоинств перед некоторыми другими. Кроме того, C# относится к семье языков с C – подобным синтаксисом, из них его синтаксис наиболее близок к C# и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты и др.

C# — это язык с C-подобным синтаксисом. Здесь он близок в этом отношении к C++ и Java.

Язык программирования С# был разработан в 1993—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft.

К 2000 году у Microsoft были готовы промышленные версии новых технологий и решений для обмена сообщениями и данными, а также для создания Internet-приложений. Была выпущена и новая платформа для разработки под новые решения — .NET. В ней объединились сразу несколько языков программирования, что было в новинку для того времени.

Ещё одним новшеством платформы .NET была технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно было относительно быстро разработать веб-приложения, взаимодействующие с базами данных. Специально для ASP.NET был создан язык программирования C#. Да и сама ASP.NET была полностью написана на нём.

Будучи объектно-ориентированным языком, он много перенял у Java и С++. Как и Java, C# изначально предназначался для веб-разработки, и примерно 75% его синтаксических возможностей такие же, как у  Java. C# также называют «очищенной версией Java». Ещё 10% наш герой позаимствовал из C++ и 5% – из Visual Basic. Оставшиеся 10% C# — это реализация собственных идей разработчиков. Объектно-ориентированный подход позволяет строить с помощью C# крупные, но в то же время гибкие, масштабируемые и расширяемые приложения.

C# уже давно поддерживает много полезных функций:

* инкапсуляция,
* наследование,
* полиморфизм,
* перегрузка операторов,
* статическая типизация.

При этом он всё ещё активно развивается, и с каждой новой версией появляется всё больше интересного — например лямбды, динамическое связывание, асинхронные методы и т.д.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

* Поддержка нескольких языков. В основе  .NET — общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему платформа поддерживает несколько языков: наряду с C# это VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. Код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) — своего рода ассемблер платформы .NET. Поэтому можно сделать отдельные модули одного приложения на разных языках.
* Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. Какое бы приложение мы ни собирались писать на C# — текстовый редактор, чат или сложный веб-сайт — так или иначе мы задействуем библиотеку классов .NET.
* Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов — это основа для целого стека технологий, которые разработчики могут задействовать при создании разных приложений. Например, для баз данных в этом стеке имеется технология ADO.NET и Entity Framework Core. Для графических приложений с насыщенным интерфейсом — технологии WPF и UWP. Для более простых графических приложений — Windows Forms. Для разработки мобильных приложений — Xamarin. Для создания веб-сайтов - ASP.NET и т.д.

.NET долгое время развивался под названием .NET Framework — преимущественно как платформа для Windows. Но с 2019 она больше не развивается — последней версией этой платформы стала .NET Framework 4.8.

В 2014 Microsoft начал выпускать альтернативную платформу - .NET Core, которая должна была вобрать в себя все возможности устаревшего .NET Framework и добавить новую функциональность. Поэтому следует различать .NET Framework, который предназначен преимущественно для Windows, и кроссплатформенный .NET Core.

C# на протяжении долгого времени уверенно держит позиции в рейтинге наиболее востребованных на рынке разработки языков. Сначала им интересовались только разработчики под Windows, но затем C# научился работать на Mac OS, Linux, iOS и Android. А после того как код платформы открыли для всех желающих, были сняты практически все возможные ограничения в применении C#. В результате язык активно развивается и применяется всё шире. Его часто рекомендуют к изучению в качестве одного из базовых для разработчиков любого профиля.

Инструментарий C# позволяет решать широкий круг задач, язык действительно очень мощный и универсальный. На нём часто разрабатывают:

* веб-приложения,
* игры,
* мобильные приложения для Android или iOS,
* программы под Windows.

Перечень возможностей разработки практически не имеет ограничений благодаря широчайшему набору инструментов и средств. Конечно, всё это можно реализовать при помощи других языков. Но некоторые из них узкоспециализированные, а в некоторых придётся использовать дополнительные инструменты сторонних разработчиков. В C# решить широкий круг задач возможно быстрее, проще и с меньшими затратами времени и ресурсов.

**Платформа Windows Forms**

Windows Forms - платформа пользовательского интерфейса для создания разнообразных клиентских приложений рабочего стола в Windows. Платформа разработки Windows Forms поддерживает широкий набор функций для разработки приложений, включая элементы управления, графику, привязку данных и ввод пользователя. Характерной чертой Windows Forms является использование визуального конструктора с функцией перетаскивания в Visual Studio для упрощения создания приложений Windows Forms.

Windows Forms — это технология пользовательского интерфейса для .NET, представляющая собой набор управляемых библиотек, которые упрощают выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью среды разработки, такой как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод пользователя и взаимодействуют с удаленными компьютерами по сети. В Windows Forms форма — это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем добавления элементов управления в формы и создания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных. При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события, как задано в коде, и обрабатывает события при их возникновении.

В Windows Forms предусмотрено множество элементов управления, которые можно добавлять в формы. Например, элементы управления могут отображать текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Если предусмотренные элементы управления не подходят для ваших целей, в Windows Forms можно создавать собственные пользовательские элементы управления с помощью класса [UserControl](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.usercontrol).

В Windows Forms имеются многофункциональные элементы управления пользовательского интерфейса, позволяющие эмулировать функции таких сложных приложений, как Microsoft Office. С помощью элементов управления [ToolStrip](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.toolstrip) и [MenuStrip](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.menustrip) вы можете создавать панели инструментов и меню, которые содержат текст и изображения, отображают подменю и размещают другие элементы управления, такие как текстовые поля и поля со списками. Используя функцию перетаскивания **конструктора Windows Forms** в Visual Studio, можно легко создавать приложения Windows Forms. Просто выделите элемент управления с помощью курсора и поместите его на нужное место в форме. Для преодоления трудностей, связанных с выравниванием элементов управления, конструктор предоставляет такие средства, как линии сетки и линии привязки. С помощью элементов управления [FlowLayoutPanel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.flowlayoutpanel), [TableLayoutPanel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.tablelayoutpanel) и [SplitContainer](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer) можно гораздо быстрее создавать сложные макеты форм.

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

**Моделирование интерфейса**

Программа сделана по шаблону Windows Forms и состоит из нескольких форм, каждая из которых отвечает за ту или иную функцию в игре.

На Form2 находится экран главного меню в игре, которая состоит из кнопок Button. При нажатии на эти кнопки пользователь может перемещаться на главную форму с основным игровым сюжетом, а также открыть окно «О программе» и посмотреть рекорд в игре на данный момент.

На Form1 находится основной интерфейс программы, где располагаются восемь муравьев в случайных местоположениях. Наверху находится выпадающее меню. На нем можно нажать на паузу, начать игру заново и выйти из игры. Также наверху есть текстовое поле, в котором идет время по мере присутствия на форме муравьев. На форме в случайном местоположении по вертикали регулярно появляются лягушки и двигаются навстречу муравьям, поедая их при соприкосновении. В правом верхнем углу ведется количество оставшихся муравьев, и когда количество будет ниже четырёх, его цвет будет изменяться, а шрифт увеличиваться. Когда муравьев не остается, пользователя перекидывают на другую форму.

На Form3 находится конечное меню игры, которое состоит из кнопок Button и открывается, когда пользователь проигрывает. На нем расположено текстовое поле со временем, с которым пользователь продержался от начала игры до конца. Здесь же находится кнопка «Играть ещё», при нажатии на которую пользователь опять переносится в основную форму, а также кнопка «Выйти»

**Листинг программы**

Form2

//Метод, который вызывается при открытии формы

private void Form2\_Load(object sender, EventArgs e)

{

FormBorderStyle = System.Windows.Forms.FormBorderStyle.Sizable; //Объект для настройки развернутого окна

WindowState = FormWindowState.Maximized;

}

//Событие по нажатию кнопки «Играть»

private void button1\_Click(object sender, EventArgs e)

{

//Скрытие кнопок-сложности игры

button4.Visible =! button4.Visible;

button5.Visible =! button5.Visible;

button6.Visible =! button6.Visible;

}

//Событие по нажатию кнопки «Рекорд»

private void button3\_Click(object sender, EventArgs e)

{

label1.Visible =! label1.Visible;

StreamReader sr = new StreamReader("Records.txt"); //Объект чтения файла

label1.Text = sr.ReadLine().ToString(); //Запись содержимого файла в Label

sr.Close(); //Закрытие работы с файлом

}

//Событие по нажатию кнопки «О нас»

private void button2\_Click(object sender, EventArgs e)

{

AboutBox1 about = new AboutBox1(); //Объект открытия формы AboutBox1

about.ShowDialog(); //Открытие формы AboutBox1

}

//Поле целочисленной переменной

public int difficult;

//Событие по нажатию кнопки «Сложность: лёгкая»

private void button4\_Click(object sender, EventArgs e)

{

this.Visible = false;

Form1 f1 = new Form1(); //Объект для работы с Form1

f1.labelDIFF.Text = "1"; //Цифра-сложность игры

f1.ShowDialog(); //Открытие Form1

this.Close(); //Закрытие текущей формы

}

//Событие по нажатию кнопки «Сложность: средняя»

private void button5\_Click(object sender, EventArgs e)

{

this.Visible = false;

Form1 f1 = new Form1(); //Объект для работы с Form1

f1.labelDIFF.Text = "2"; //Цифра-сложность игры

f1.ShowDialog(); //Открытие Form1

this.Close(); //Закрытие текущей формы

}

//Событие по нажатию кнопки «Сложность: сложная»

private void button6\_Click(object sender, EventArgs e)

{

this.Visible = false;

Form1 f1 = new Form1(); //Объект для работы с Form1

f1.labelDIFF.Text = "3"; //Цифра-сложность игры

f1.ShowDialog(); //Открытие Form1

this.Close(); //Закрытие текущей формы

}

}

}

Form1

Image FonImg; //Объекты, позволяющие отображать изображения

Image AntImg1;

Image AntImg2;

Image AntImg3;

Image Frog1Img;

Image Frog2Img;

Image Frog3Img;

Image Frog4Img;

Image Frog5Img;

Image Frog\_eat4;

Rectangle r1; // Объекты, рисующие прямоугольник

Rectangle r2;

//Таким образом происходит объявление Rectangle до r8

Rectangle r1Frog;

Rectangle r1Frog\_2;

Rectangle[] rectangle;

string storona = "";

private int positionFrogX = 20;

private int positionFrogY = 125;

private double positionAntY = 200;

int boost = 0;

int interval\_minute = 1;

int interval\_second = 15;

int Ants = 8;

//Метод движения лягушки

public void MoveFrog()

{

if (Convert.ToInt32(split[1]) >= interval\_minute) //Сравнение количества минут рекорда с количеством минут текущего времени

{

if (labelDIFF.Text == "3")

{

boost += 50; //Повышение скорости лягушки

interval\_minute++;

}

else if (labelDIFF.Text == "2")

{

boost += 25; //Повышение скорости лягушки

interval\_minute++;

}

if (labelDIFF.Text == "1")

{

boost += 20; //Повышение скорости лягушки

interval\_minute++;

}

}

else if (Convert.ToInt32(split[2]) >= interval\_second) //Сравнение количества секунд рекорда с количеством секунд текущего времени

{

if (labelDIFF.Text == "3")

{

boost += 50; //Повышение скорости лягушки

interval\_second += 15;

}

else if (labelDIFF.Text == "2")

{

boost += 25; //Повышение скорости лягушки

interval\_second += 20;

}

if (labelDIFF.Text == "1")

{

boost += 20; //Повышение скорости лягушки

interval\_second += 30;

}

}

if (frogX[0] < -400) //Сравнение местоположения лягушки вдоль оси X с координатами за формой

{

Random posFrog = new Random(); // Случайное расположение лягушки с помощью класса Random

frogX = new int[1];

for (int i = 0; i < frogX.Length; i++)

{

frogX[i] = posFrog.Next(1850, 1900);

}

frogY = new int[1];

for (int i = 0; i < frogY.Length; i++)

{

frogY[i] = posFrog.Next(-150, 650);

}

frogY[0] += positionFrogY;

r1Frog = new Rectangle(frogX[0], frogY[0] + 30, 350, 300); //отрисовывание прямоугольника с лягушкой

frogX[0] -= (positionFrogX + boost);

r1Frog\_2 = new Rectangle(frogX[0], frogY[0], 450, 400);

frogY[0] -= positionFrogY;

}

else

{

frogY[0] += positionFrogY;

r1Frog = new Rectangle(frogX[0], frogY[0] + 30, 350, 300);

frogY[0] -= positionFrogY;

r1Frog\_2 = new Rectangle(frogX[0], frogY[0], 450, 400);

frogX[0] -= (positionFrogX + boost);

}

}

bool IsKeyDown = false;

//Метод удаления элемента из массива

public void Array(ref Rectangle[] rectangle, int index)

{

massX[index] += 100000000; //Перенос муравья за форму по оси X

massY[index] += 100000000; //Перенос муравья за форму по оси Y

IsKeyDown = false;

if (Ants <= 3)

{

if (Ants == 3) //окрашивание количество муравьев в зеленый, и повышение шрифта, если их 3

{

label2.Font = new Font(label2.Font.Name, 10, label2.Font.Style);

label2.Location = new Point(1315, 118);

label2.ForeColor = Color.ForestGreen;

label2.Text = Ants.ToString();

}

else if(Ants == 2)

//окрашивание количество муравьев в оранжевый, и повышение шрифта, если их 2

{

label2.Font = new Font(label2.Font.Name, 20, label2.Font.Style);

label2.Location = new Point(1306, 113);

label2.ForeColor = Color.Orange;

label2.Text = Ants.ToString();

}

else if (Ants == 1)

//окрашивание количество муравьев в красный, и повышение шрифта, если их 1

{

label2.Font = new Font(label2.Font.Name, 22, label2.Font.Style);

label2.Location = new Point(1304, 114);

label2.ForeColor = Color.Red;

label2.Text = Ants.ToString();

}

}

//удаление элемента из массива

Rectangle[] newRectangle = new Rectangle[rectangle.Length - 1];

for (int i = 0; i < index; i++)

{

newRectangle[i] = rectangle[i];

}

for (int i = index + 1; i < rectangle.Length; i++)

{

newRectangle[i - 1] = rectangle[i];

}

rectangle = newRectangle;

}

int index;

bool IsAte = false;

//Метод движения муравьев

public void MoveAnt()

{

if (!IsAte)

{

rectangle = new Rectangle[8] { r1, r2, r3, r4, r5, r6, r7, r8 }; //Объявление массива с переменными Rectangle

}

r1 = new Rectangle(massX[0], massY[0], massSize[0], massSize[0]);

r2 = new Rectangle(massX[1], massY[1], massSize[1], massSize[1]);

//Таким образом задаются параметры оставшихся Rectangle

positionAntY = 200;

for (int i = 0; i < rectangle.Length; i++)

{

if (storona == "Up")

{

bg.Graphics.Clear(BackColor);

if (massY[i] > 80) //При перемещении муравьев по оси Y вверх у каждого имеется своя скорость перемещения

rectangle[i] = new Rectangle(massX[i], massY[i] -= (int)Math.Round(Math.Sqrt(positionAntY)), massSize[i], massSize[i]);

}

if (storona == "Left") //При перемещении муравьев по оси X влево у каждого имеется своя скорость перемещения

{

bg.Graphics.Clear(BackColor); //Очистка графики

if (massX[i] > 20)

rectangle[i] = new Rectangle(massX[i] -= (int)Math.Round(Math.Sqrt(positionAntY)), massY[i], massSize[i], massSize[i]);

}

if (storona == "Right") //При перемещении муравьев по оси X вправо у каждого имеется своя скорость перемещения

{

bg.Graphics.Clear(BackColor); //Очистка графики

if (massX[i] < 1600)

rectangle[i] = new Rectangle(massX[i] += (int)Math.Round(Math.Sqrt(positionAntY)), massY[i], massSize[i], massSize[i]);

}

if (storona == "Space")

{

rectangle[i] = new Rectangle(massX[i], massY[i], massSize[i], massSize[i]);

IsKeyDown = false;

}

if (storona == "Down") //При перемещении муравьев по оси Y вниз у каждого имеется своя скорость перемещения

{

bg.Graphics.Clear(BackColor); //Очистка графики

if (massY[i] < 850)

rectangle[i] = new Rectangle(massX[i], massY[i] += (int)Math.Round(Math.Sqrt(positionAntY)), massSize[i], massSize[i]);

}

if (positionAntY <= 1700)

{

positionAntY \*= 2;

}

else

{

positionAntY /= 5;

}

if (storona == "Up" || storona == "Down" || storona == "Right" || storona == "Left")

{

IsKeyDown = true;

}

}

for (int i = 0; i < rectangle.Length; i++)

{

if (((massX[i] + 10 <= frogX[0] + 80 && massX[i] + massSize[i] - 10 >= frogX[0] + 80) || (massX[i] - 10 + massSize[i] <= frogX[0] + 80 + 400 && massX[i] + 10 >= frogX[0] + 80) || (massX[i] + 10 >= frogX[0] + 80 && massX[i] <= frogX[0] + 400 - 80)) && ((massY[i] + 10 + massSize[i] >= frogY[0] + 40 && massY[i] + 10 + massSize[i] <= frogY[0] + 350) || (massY[i] + 10 >= frogY[0] + 20 && massY[i] + 10 <= frogY[0] +350) || (massY[i] + 10 + massSize[i] >= frogY[0] + 40 && massY[i] + 10 + massSize[i] <= frogY[0] + 350))) //Условие, при котором лягушка соприкасается с каким-то из муравьев

{

index = i;

IsAte = true;

bg.Graphics.Clear(BackColor);

bg.Graphics.DrawImage(Game.Properties.Resources.box, rectangle[i], 50, 100, 600, 500, GraphicsUnit.Pixel); //Вырисовка части окна, в которой ведется количество муравьев

if (Ants != 0) //Условие отсутствия муравьев

{

Ants -= 1;

label2.Text = Ants.ToString();

}

else

label2.Text = Ants.ToString();

for (int j = 0; j < rectangle.Length; j++)

{

if (index != j) //Условие, выполняющееся только с оставшимися муравьями

{

rectangle[j] = new Rectangle(massX[j], massY[j], massSize[j], massSize[j]);

}

}

Array(ref rectangle, index); //Вызов метода удаления элемента из массива

}

}

End(); //Вызов метода проигрыша

}

StreamWriter sw; //Объект записи в файл

StreamReader sr; //Объект чтения файла

//Метод проигрыша

private void End()

{

if (Ants == 0 && window == true)

{

sr = new StreamReader("Records.txt", Encoding.GetEncoding(1251));

while (!sr.EndOfStream)

{

qst = sr.ReadLine();

}

this.Visible = false;

window = false;

timer1.Stop();

timer2.Stop();

Form3 f3 = new Form3();

f3.time2.Text = time.ToString("mm:ss:ff");

f3.ShowDialog();

this.Close();

sr.Close();

if (Convert.ToInt32(split[1]) > Convert.ToInt32(qst.Split(':')[0]))

{

sw = new StreamWriter("Records.txt", false); //запись времени в файл

sw.WriteLine(time.ToString("mm:ss:ff"));

sw.Close(); //закрытие работы с файлами

}

else if (Convert.ToInt32(split[2]) > Convert.ToInt32(qst.Split(':')[1]))

{

sw = new StreamWriter("Records.txt", false);

sw.WriteLine(time.ToString("mm:ss:ff"));

sw.Close(); //закрытие работы с файлами

}

}

}

int[] massX = new int[8]; //массив координат X муравьев

int[] massY = new int[8]; //массив координат Y муравьев

int[] massSize = new int[8]; //массив размеров муравьев

int[] frogX = new int[1]; //массив координат X лягушки

int[] frogY = new int[1]; //массив координат Y лягушки

Graphics g;

//Обработчик события при открытии формы

public void Form1\_Load(object sender, EventArgs e)

{

//Шрифт и цвет количества муравьев по умолчанию

label2.Text = Ants.ToString();

label2.BackColor = Color.White;

pictureBox1.BackColor = Color.White;

label2.Location = new Point(1317, 120);

Window(); //Вызов метода расширения экрана

g = CreateGraphics(); //Объект для элемента управления графикой

bg = buff.Allocate(CreateGraphics(), ClientRectangle); //Создание графического буфера

bg.Graphics.SmoothingMode = SmoothingMode.HighQuality;

Random pos = new Random(); //Координаты муравьев

massX = new int[8];

for (int i = 0; i < 8; i++)

{

massX[i] = pos.Next(300);

}

massY = new int[8];

for (int i = 0; i < 8; i++)

{

massY[i] = pos.Next(88, 900);

}

Random posFrog = new Random(); // Координаты лягушек

frogX = new int[1];

for (int i = 0; i < frogX.Length; i++)

{

frogX[i] = posFrog.Next(1800, 1850);

}

frogY = new int[1];

for (int i = 0; i < frogY.Length; i++)

{

frogY[i] = posFrog.Next(0, 640);

}

Random random\_size = new Random(); //Размеры муравьев

massSize = new int[8];

for (int i = 0; i < 8; i++)

{

massSize[i] = pos.Next(90, 120);

}

AntImg1 = new Bitmap(Game.Properties.Resources.ANT);

AntImg2 = new Bitmap(Game.Properties.Resources.ANT21);

FonImg = new Bitmap(Game.Properties.Resources.Background1);

//Таким образом, работает форматирование всех переменных Image в изображения, определяемые пикселями

//Запуск Timer

timer1.Start();

timer2.Start();

}

//Метод для развернутого окна приложения

private void Window()

{

FormBorderStyle = System.Windows.Forms.FormBorderStyle.Sizable;

WindowState = FormWindowState.Maximized; //Развернутое окно приложения

}

//Обработчик события при нажатии на клавишу

private void Form1\_KeyDown\_1(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Up: //Нажата стрелка наверх

storona = "Up";

break;

case Keys.Space: //Нажат пробел

storona = "Space";

break;

case Keys.Right: //Нажата стрелка направо

storona = "Right";

break;

case Keys.Left: //Нажата стрелка налево

storona = "Left";

break;

case Keys.Down: //Нажата стрелка вниз

storona = "Down";

break;

case Keys.Escape: //Нажат Escape

FormBorderStyle = System.Windows.Forms.FormBorderStyle.Sizable;

WindowState = FormWindowState.Normal; //Свернутое окно приложения

break;

}

MoveAnt(); // Вызов метода движения муравьев

}

bool dir = true;

BufferedGraphicsContext buff = BufferedGraphicsManager.Current; //Двойная буфферизация

BufferedGraphics bg;

int num = 0;

Rectangle fon;

Rectangle box;

Rectangle miniAnt;

bool window = true;

//Обработчик события, предназначенный для исполнения действий через равные промежутки времени

private void timer1\_Tick(object sender, EventArgs e) {

MoveAnt(); // Вызов метода движения муравьев

MoveFrog(); // Вызов метода движения лягушек

fon = new Rectangle(0, 0, 1956, 1036); //Отрисовка фона основной формы

box = new Rectangle(1800, 25, 210, 150); //Отрисовка поля счета муравьев

miniAnt = new Rectangle(1840, 45, 100, 100); //Отрисовка маленького муравья в поле счета

Random monsters\_r = new Random();

timer1.Interval = 70;

bg = buff.Allocate(CreateGraphics(), ClientRectangle); //Создание графического буфера

bg.Graphics.SmoothingMode = SmoothingMode.HighQuality;

bg.Graphics.Clear(BackColor);

num++;

if (dir)

{

//Вырисовка анимации по кадрам

if (num == 1)

{

bg.Graphics.DrawImage(FonImg, fon, 0, 0, 1900, 1000, GraphicsUnit.Pixel);

if (IsAte == true) //Условие, при котором лягушка съедает муравья

{

bg.Graphics.DrawImage(Frog\_eat4, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel); //Вырисовка изображения лягушки при поедании

IsAte = false;

}

else

{

bg.Graphics.DrawImage(Frog1Img, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel);

}

if (IsKeyDown) //Условие отвечающее за остановку анимации при остановке муравьев

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

else

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

}

else if (num == 2)

{

bg.Graphics.DrawImage(FonImg, fon, 0, 0, 1900, 1000, GraphicsUnit.Pixel);

if (IsAte == true)

{

bg.Graphics.DrawImage(Frog\_eat4, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel);

IsAte = false;

}

else

{

bg.Graphics.DrawImage(Frog2Img, r1Frog\_2, 5, 10, 1300, 1000, GraphicsUnit.Pixel); //Вырисовывается изображение лягушки

}

if (IsKeyDown)

{

bg.Graphics.DrawImage(AntImg2, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg2, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

else

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

}

else if (num == 3)

{

bg.Graphics.DrawImage(FonImg, fon, 0, 0, 1900, 1000, GraphicsUnit.Pixel);

if (IsAte == true)

{

bg.Graphics.DrawImage(Frog\_eat4, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel);

IsAte = false;

}

else

{

bg.Graphics.DrawImage(Frog3Img, r1Frog\_2, 5, 10, 1300, 1000, GraphicsUnit.Pixel); //Вырисовывается изображение лягушки

}

if (IsKeyDown)

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

else

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

}

else if (num == 4)

{

bg.Graphics.DrawImage(FonImg, fon, 0, 0, 1900, 1000, GraphicsUnit.Pixel);

if (IsAte == true)

{

bg.Graphics.DrawImage(Frog\_eat4, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel);

IsAte = false;

}

else

{

bg.Graphics.DrawImage(Frog4Img, r1Frog\_2, 5, 10, 1300, 1000, GraphicsUnit.Pixel); //Вырисовывается изображение лягушки

}

if (IsKeyDown)

{

bg.Graphics.DrawImage(AntImg3, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg3, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

else

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовывается графика до r8

}

}

else if (num == 5)

{

bg.Graphics.DrawImage(FonImg, fon, 0, 0, 1900, 1000, GraphicsUnit.Pixel);

if (IsAte == true)

{

bg.Graphics.DrawImage(Frog\_eat4, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel);

IsAte = false;

}

else {

bg.Graphics.DrawImage(Frog5Img, r1Frog\_2, 5, 10, 1280, 1000, GraphicsUnit.Pixel); //Вырисовывается изображение лягушки

}

if (IsKeyDown)

{

bg.Graphics.DrawImage(AntImg2, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg2, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

else

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

}

else dir = false;

if (dir)

{

bg.Render(); //Возврат готового изображения

}

}

else

{

num = 0;

bg.Graphics.DrawImage(FonImg, fon, 0, 0, 1900, 1000, GraphicsUnit.Pixel);

if (IsAte == true)

{

bg.Graphics.DrawImage(Frog\_eat4, r1Frog, 50, 150, 1000, 700, GraphicsUnit.Pixel);

IsAte = false;

}

else

{

bg.Graphics.DrawImage(Frog1Img, r1Frog, 100, 50, 1000, 700, GraphicsUnit.Pixel); //Вырисовывается изображение лягушки

};

if (IsKeyDown)

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

else

{

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовываются изображения муравьев до r8

}

dir = true;

bg.Render(); //Возврат готового изображения

}

timer2.Interval = 1;

}

// Структура, представляющая текущее время

DateTime time = new DateTime(0, 0);

string[] split = new string[3]; //разделение массива со временем на минуты, секунды и милисекунды

string qst = "";

//Обработчик события, предназначенный для исполнения действий через равные промежутки времени

private void timer2\_Tick(object sender, EventArgs e)

{

time = time.AddMilliseconds(20); //добавление 20 милисекунд в каждый промежуток времени выполнения Timer

label1.Text = time.ToString("mm:ss:ff"); //Запись текущего времени в Label

split = time.ToString().Split(':'); //Деление времени на минуты, секунды и милисекунды

}

//Обработчик события при нажатии «Пауза»

private void паузаToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Приостановка работы Timer

timer1.Enabled = false;

timer2.Enabled = false;

}

//Обработчик события при нажатии «Выйти»

private void выйтиToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Close(); //Закрытие текущей формы

}

//Обработчик события при нажатии «Продолжить»

private void продолжитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Возобновление работы Timer

timer1.Enabled = !timer1.Enabled;

timer2.Enabled = !timer2.Enabled;

}

//Обработчик события при нажатии «Начать заново»

private void начатьЗановоToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Возврат исходных параметров игры

//Настройка шрифта и цвета количества муравьев по умолчанию

label2.Font = new Font(label2.Font.Name, 8, label2.Font.Style);

label2.Location = new Point(1317, 120);

label2.ForeColor = Color.Black;

label2.Text = Ants.ToString();

Rectangle[] rectangle = new Rectangle[8] { r1, r2, r3, r4, r5, r6, r7, r8 };

IsAte = false;

r1 = new Rectangle(massX[0], massY[0], massSize[0], massSize[0]);

r2 = new Rectangle(massX[1], massY[1], massSize[1], massSize[1]);

//Таким образом задаются параметры оставшихся Rectangle

Ants = 8;

label2.Text = Ants.ToString();

positionAntY = 200;

Random pos = new Random();

massY = new int[8];

for (int i = 0; i < 8; i++)

{

massY[i] = pos.Next(88, 900);

}

massX = new int[8];

for (int i = 0; i < 8; i++)

{

massX[i] = pos.Next(300);

}

Random posFrog = new Random(); //Настройка случайного позиционирования лягушек

frogX = new int[1];

for (int i = 0; i < frogX.Length; i++)

{

frogX[i] = posFrog.Next(1300, 1350);

}

frogY = new int[1];

for (int i = 0; i < frogY.Length; i++)

{

frogY[i] = posFrog.Next(88, 700);

}

bg.Graphics.DrawImage(AntImg1, r1, 50, 100, 600, 500, GraphicsUnit.Pixel);

bg.Graphics.DrawImage(AntImg1, r2, 50, 100, 600, 500, GraphicsUnit.Pixel);

//Таким образом вырисовывается графика до r8

bg.Graphics.DrawImage(Frog1Img, r1Frog, 100, 50, 1000, 700, GraphicsUnit.Pixel);

storona = "";

IsKeyDown = false;

timer1.Start(); //Возобновление работы Timer

timer2.Start();

time = new DateTime(0, 0);

bg.Render(); //Возврат готового изображения

}

Form 3

private void Form3\_Load(object sender, EventArgs e) //Обработчик события при открытии формы

{

FormBorderStyle = System.Windows.Forms.FormBorderStyle.Sizable; //Настройка развернутого окна

WindowState = FormWindowState.Maximized;

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close(); //Закрытие текущей формы

}

private void button2\_Click(object sender, EventArgs e)

{

this.Visible = false;

Form1 f1 = new Form1(); //Объект для открытия Form1

f1.ShowDialog(); //Открытие Form1 в новом окне

this.Close(); //Закрытие текущей формы

}

**Инструкция пользователя**

1. Открыть приложение.
2. Для старта игры нажать кнопку «Играть» и далее выбрать сложность игры.
3. Задача пользователя – управлять восьми муравьями при помощи стрелочек на клавиатуре и спасти их от надвигающихся лягушек. При нажатии на пробел, муравьи останавливаются.   
   В течении игры можно наблюдать за количеством муравьев на специальном поле в правом верхнем углу.  
   С течением времени лягушки начинают ускорятся и изворачиваться от них будет становиться сложнее.  
   Сверху находится выплывающее меню, в котором можно нажать на паузу, продолжить игру, начать игру заново и выйти из игры. Также сверху ведется время, в течении которого муравьи остаются живыми.

Когда же их не остается, игра заканчивается.

1. На последней форме показано время удержания муравьев живыми, кнопка «Выйти», чтобы выйти из игры, и кнопка «Играть ещё», чтобы начать игру заново. Если поставлен рекорд по времени, он записывается в файл и будет показываться при нажатии кнопки «Рекорд».

**Тестовый пример**

Начальная форма является экраном главного меню игры. Здесь расположены основные кнопки для ознакомления пользователя с текущим рекордом игры и её разработчиком. Здесь же пользователь выбирает сложность и начинает игру. На рисунке 1 изображен интерфейс экрана главного меню.



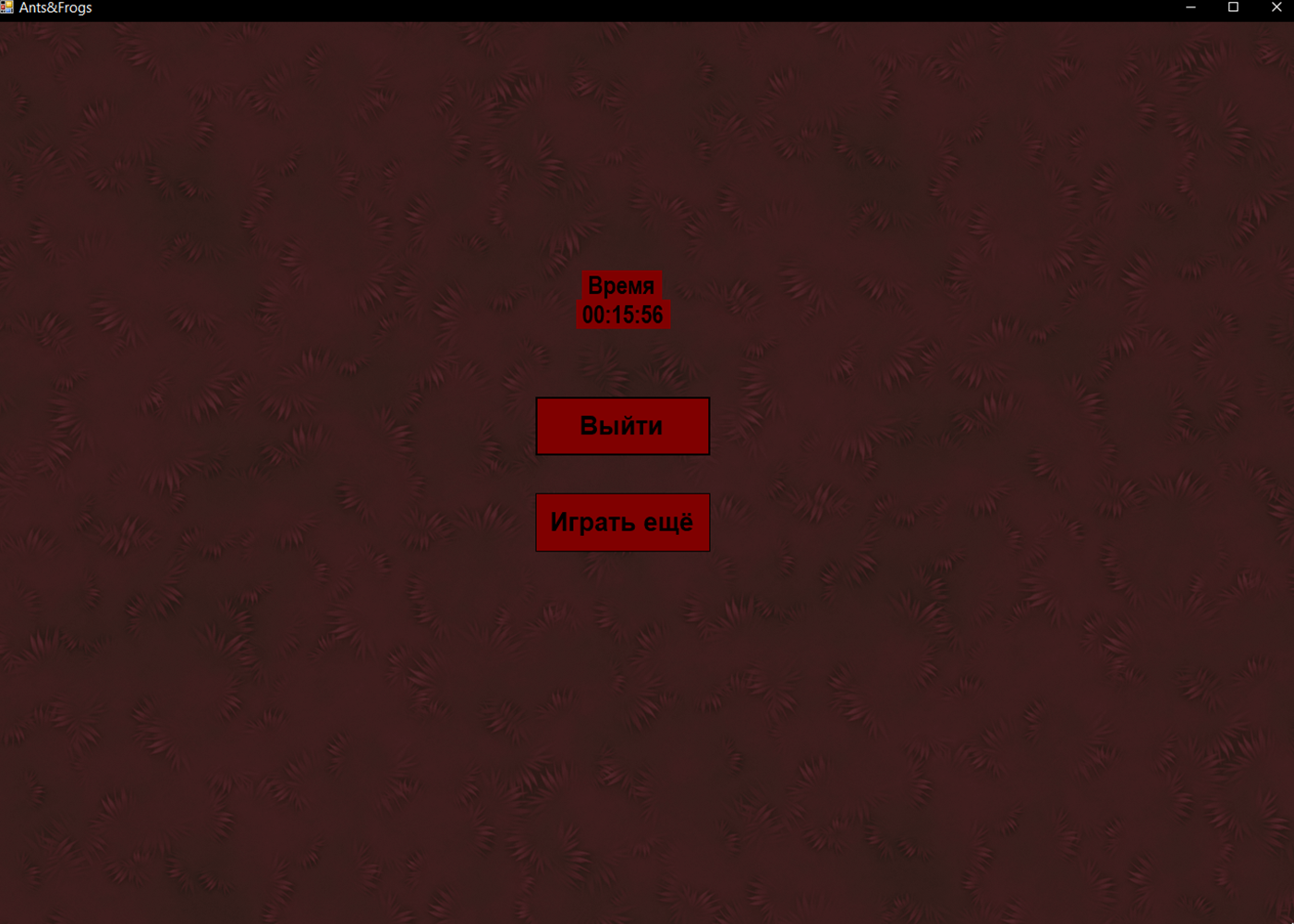
*Рисунок 1 – Экран главного меню*

При нажатии кнопки «Играть» мы перемещаемся на основную форму. На ней происходит сам игровой процесс. На форме расположены муравьи и лягушка. На выплывающем меню можно поставить игру на паузу, продолжить игру, начать заново и выйти. Также ведется счет времени с начала игрового процесса. При соприкосновении лягушки с муравьем, она его съедает. В верхнем правом углу находится поле со счетом оставшихся муравьев. На рисунке 2 изображен интерфейс основной формы игры на паузе.



*Рисунок 2– Основная форма*

Когда муравьев не остается, пользователь проигрывает и перемещается на конечную форму. Здесь показано время удержания муравьев живыми. Также, с помощью кнопок можно начать игру заново или выйти. Интерфейс конечной формы изображен на рисунке 3.



*Рисунок 3 – Экран конца игры*

**Актуальность выбранной темы**

Индустрия компьютерных игр (разработка, издание и продвижение игр) – одна из самых быстро развивающихся отраслей компьютерных технологий и одновременно глобального сектора развлечений. Игры становятся культурными феноменами и признаются произведениями искусства. Число геймеров растет (2,5 млрд игроков в 2019 г.), а сам гейминг становится высокооплачиваемой профессией.

Что касается ключевых тенденций, определяющих развитие глобальной игровой индустрии, то здесь, в первую очередь, необходимо отметить новые технологии: мобильные, облачные, виртуальные, графические, искусственный интеллект и др. Уже в течение нескольких лет компьютерные игры активно интегрируются в соцсети. Также появился и обратный тренд – они сами становятся своеобразными соцсетями. Растет популярность киберспорта. Сейчас киберспортивные турниры уже могут быть сравнимы с чемпионатами в традиционных видах спорта по размеру призовых.

На сегодняшний день тема компьютерных игр является все более актуальной.

**ЗАКЛЮЧЕНИЕ**

В процессе работы над этой программой были сложности с удалением изображений муравьев при их поедании. Проблема была решена путём их перемещения за пределы формы по осям X и Y. Также были сложности с неточным условием, при котором лягушка поедала муравья при их соприкосновении. Эта проблема была решена добавлением нескольких других условий. В целом, все изменения работают корректно. В будущем можно будет продемонстрировать эту игру в 3D, используя, например, такой движок, как Unity. Цель, поставленная в начале работы, достигнута. Программа выполнена.

**СПИСОК ЛИТЕРАТУРЫ**

1. Microsoft Corporation. Принципы проектирования и разработки программного обеспечения. Учеб. курс MCSD. М.: Изд.-торг. дом «Русская редакция», 2000.
2. Windows Forms. Программирование на C#. – Режим доступа: http://csharpcoding.org/category/windows-forms/
3. Агуров П. В. C#. Разработка компонентов в MC Visual studio 2005/2008 Агуров П. В. – СПб.:БХВ-Петербург, 2008. – 479 с.
4. Агуров П. В. C#. Сборник рецептов / Агуров П. В. – СПб.:БХВ-Петербург, 2007. – 412с.
5. Албахари Дж. C# 6.0. Справочник. Полное описание языка [Текст] / Дж . Албахари, Б. Албахари — 6-е изд. — Москва: Вильямc, 2016. - 1040 c.
6. Артамонов И.В. Разработка распределенных сервисно-ориентированных программных средств / И.В. Артамонов – Иркутск: Изд-во БГУЭП, 2012. – 130 с.
7. Биллиг, В. А. Объектное программирование в классах на C# 3.0 В.А. Биллиг. - 2-е изд., испр.. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 391 с. Полный текст находится в ЭБС "Университетская библиотека ONLINE".
8. Биллиг В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008): учебное пособие В. А. Биллиг. — Москва: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2016. - 584 c.