Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

«Игра на C#»

Выполнил:
студент группы РТ5-31Б:
Савельева В. О.

Подпись и дата:

Проверил:
преподаватель кафедры ИУ5
Гапанюк Ю. Е.

Подпись и дата:

Москва, 2024 г.

Постановка задачи

Разработать игру на языке C# с использованием Windows Forms.

Текст программы

Program.cs

```
using Hello;
namespace hw;

static class Program
{
    [STAThread]
    static void Main()
    {
        // запуск стартовой формы (меню)
        ApplicationConfiguration.Initialize();
        Application.Run(new StartForm());
    }
}
```

Form1.cs

```
using Game;
using entity;
using Hello;

namespace hw;

public partial class Form1 : Form
{
    System.Media.SoundPlayer gam = new System.Media.SoundPlayer();
    System.Media.SoundPlayer ending = new System.Media.SoundPlayer();
    Image wolfPng;   // спрайты персонажа
    Image groundPng;  // картинка земли
    Image homePng;     // картинка дома
    Image skyPng;      //небо
    Image goosePng;    // враги

    Player player;     // игрок
    List<Enemy> geese;     // список врагов
    private bool lastKey = false;    // нажатие клавиши
    int sideObj;    // размер объектов карты
    const int h = 9;   // длина карты
    const int w = 5;    // ширина карты
    int countDeath = 0;   // определяет первую истинную смерть
    int countMute = 0;    // определяет состояние звука
    int xHome;    // координата дома
    int omg = 0;  // количество очков
```

```
    int bornTimer = 0;    // счетчик для создания гусей
    int born = 0;    // количество создаваемых гусей
    bool muteS = false;    // состояние звука
    Random rnd = new Random();    // псевдорандом
    int [,] map;       // массив для карты
    public Form1(bool mute)
    {
        InitializeComponent();

        gam.SoundLocation = "souce\\do.wav";
        ending.SoundLocation = "souce\\cursedPart.wav";

        muteS = mute;

        StartPosition = FormStartPosition.CenterScreen;                 //
установка положения формы на экране
        // загрузка изображений
        wolfPng = new Bitmap("souce\\wolf.png");
        groundPng = new Bitmap("souce\\ground.png");
        skyPng = new Bitmap("souce\\sky.png");
        homePng = new Bitmap("souce\\home.png");
        goosePng = new Bitmap("souce\\dobrogoose.png");

        player = new Player(new Size(192, 115), 400, 450, wolfPng);     // объект
класса игрок

        geese = [];

        // установка параметров таймеров анимации
        gameTimer.Interval = 100;
        gameTimer.Tick += new EventHandler(update);
        gameTimer.Start();
        // установка параметров таймеров движения
        gameTimerMove.Interval = 20;
        gameTimerMove.Tick += new EventHandler(moveUpdate);
        gameTimerMove.Start();

        // установка параметров таймера движения врагов

        gooseTimer.Interval = 20;
        gooseTimer.Tick += new EventHandler(moveGoose);
        gooseTimer.Start();

        // размер объектов карты
        sideObj = 140;

        // карта

        map = new int[w, h] {{4, 1, 1, 1, 1, 1, 1, 1, 4},
                             {4, 1, 1, 1, 1, 1, 1, 1, 4},
                             {4, 1, 1, 1, 1, 1, 1, 1, 4},
```

```
                            {4, 1, 1, 1, 0, 1, 1, 1, 4},
                            {2, 2, 2, 2, 2, 2, 2, 2, 2}
                           };

        // события нажатия на кнопку

        KeyDown += new KeyEventHandler(presskey);
        KeyUp += new KeyEventHandler(freekey);

        // двойная буферизация

        DoubleBuffered = true;
        SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

    }

    // клавиши не нажаты
    private void freekey(object? sender, KeyEventArgs e){
        lastKey = false;
        if (player.life){
            switch(e.KeyCode.ToString()){
                case "A":                       // последняя нажатая клавиша A,
анимация статичный влево
                    if (player.life) player.currAnim = 3;
                    else player.currAnim = 1;
                    break;
                case "D":                       // последняя нажатая клавиша D,
анимация статичный вправо
                    if (player.life) player.currAnim = 8;
                    else player.currAnim = 6;
                    break;
                case "Left":                    // последняя нажатая клавиша Left,
анимация статичный влево
                    if (player.life) player.currAnim = 3;
                    else player.currAnim = 1;
                    break;
                case "Right":                   // последняя нажатая клавиша Right,
анимация статичный вправо
                    if (player.life) player.currAnim = 8;
                    else player.currAnim = 6;
                    break;
            }
            player.maxFrame = 3;
            player.currFrame = 0;
        }
        else player.maxFrame = 4;
        player.roarR = false;
        player.roarL = false;

    }
```

```
    // клавиша нажата, смена анимаций
    private void presskey(object? sender, KeyEventArgs e){
        if (player.life){
            switch(e.KeyCode.ToString()){
                case "A":
                    player.currAnim = 4;
                    player.maxFrame = 4;
                    break;
                case "D":
                    player.currAnim = 9;
                    player.maxFrame = 4;
                    break;
                case "Left":
                    player.currAnim = 0;
                    player.maxFrame = 3;
                    player.roarL = true;
                    break;
                case "Right":
                    player.currAnim = 5;
                    player.maxFrame = 3;
                    player.roarR = true;
                    break;
                }
            lastKey = true;
        }
        else{
            player.currAnim = 1;
            player.maxFrame = 4;
        }
        // нажатие кнопки Escape
        if (e.KeyCode.ToString() == "Escape"){
            // остановка всех таймеров
            gameTimer.Stop();
            gameTimerMove.Stop();
            gooseTimer.Stop();
            // остановка воспроизведения музыки
            gam.Stop();
            ending.Stop();
            // включение композиции стартовой формы, если необходимо
            if (!StartForm.mut) StartForm.opening.PlayLooping();
            //закрытие формы игры
            Close();
        }
    }

    // обновление кадров
    private void update(object? sender, EventArgs e){
        if(lastKey) {
            gameTimer.Interval = 80;
        }
        else {
```

```
            gameTimer.Interval = 100;
        }
        // обнуляет/фиксирует текущий кадр при превышении максимального
количества кадров текущей анимации в зависимости от состояния персонажа
        if (player.currFrame >= player.maxFrame) {
            if (player.life) player.currFrame = 0;
            else player.currFrame = 3;
        }
        // переход к следующему кадру анимации
        player.currFrame ++;
        Invalidate();
    }

    // обновление положения на экране
    private void moveUpdate(object? sender, EventArgs e){
        collision();
        if (player.life){
            switch(player.currAnim){
            case 4:
                player.Left();
                break;
            case 9:
                player.Right();
                break;
            }
        }
        else{
            player.currAnim = 1;
            player.maxFrame = 4;
        }
        Invalidate();
    }

    // движение врагов

    private void moveGoose(object? sender, EventArgs e){
        if(player.life){
            // производство гусей
            bornTimer += 20;

            if (bornTimer % 3000 == 0){
                // невероятная формула для расчета количества появляющихся врагов
                born = rnd.Next(1, 5) + bornTimer % 3000 / (omg + 1);
                // добавление создаваемых справа и слева врагов в список врагов
                for (int i = 0; i < born / 2; ++i){
                    geese.Add(new Enemy(new Size(112, 96), -112, 465, goosePng,
true, 3 + rnd.Next(3) + omg / 10));
                }
                for (int i = born / 2; i <= born; ++i){
                    geese.Add(new Enemy(new Size(112, 96), 1400, 465, goosePng,
false, 3 + rnd.Next(3) + omg / 10));
```

```
                }
            }
            // обнуление количества рождений
            born = 0;
            // непосредственно движение гусей по экрану
            if (geese.Count!=0){
                for (int i = 0; i < geese.Count; ++i){
                    if (!geese[i].reverse) geese[i].Left();
                    else geese[i].Right();
                }
            }
        }
        // при смерти игрока останавливается производство гусей и их передвижение
по экрану
        else{
            if (geese.Count!=0){
                for (int i = 0; i < geese.Count; ++i){
                    geese[i].currAnim = 4;
                }
            }
        }
        Invalidate();
    }

    // отрисовка гусей на экране

    private void playEnemy(Graphics g){
        if (geese.Count!=0){
            for (int i = 0; i < geese.Count; ++i){
                g.DrawImage(geese[i]._anims, geese[i]._x, geese[i]._y, new
Rectangle(new Point(112 * geese[i].currAnim, 0), new Size(112, 96)),
GraphicsUnit.Pixel);
                if (geese[i].sad && (geese[i]._x > 1400 || geese[i]._x < -110)){
                    geese.RemoveAt(i);
                }
            }
        }
    }

    // проверка на столкновения персонажа с гусями и гусей с домом

    private void collision(){
        if(!muteS && countMute == 0) {
            gam.PlayLooping();
            countMute ++;
        }
        if (geese.Count != 0){
            for (int i = 0; i < geese.Count; ++i){
                // условие для атаки гусей
```

```
                if ((geese[i]._x - player._x <= 210 && geese[i]._x - player._x >=
0 && player.roarR || player._x - geese[i]._x <= 110 && player._x - geese[i]._x >=
0  && player.roarL) && !geese[i].sad){
                    geese[i].pain();
                    omg++;
                    label1.Text = "Очки: " + omg;
                }
                // смерть персонажа при столкновении гуся с домом или персонажа с
гусем
                if ((geese[i]._x >= player._x - 10 && geese[i]._x <= player._x +
80 || geese[i]._x >= xHome + 30 && geese[i]._x <= xHome + 40) && !geese[i].sad){
                    geese[i].currAnim = 4;
                    player.life = false;
                    countDeath++;
                    if (countDeath == 1) player.currFrame = 0;
                    label1.Location = new Point(300, 300);
                    label1.Text = "Вы проиграли. Вы старались и набрали " + omg +
" очков";
                    //включение грустной музыки в случае проигрыша и при
соответствующем состоянии звука
                    if (!muteS){
                        gam.Stop();
                        ending.PlayLooping();
                        muteS = true;
                    }
                }
            }
        }
        // персонаж больше не атакует
        player.roarR = false;
        player.roarL = false;
    }

    // анимация персонажа
    private void playAnimation(Graphics g){
        g.DrawImage(player._anims, player._x, player._y, new Rectangle(new
Point(192 * player.currFrame, 115 * player.currAnim), new Size(192, 115)),
GraphicsUnit.Pixel);
    }

    // отрисовка карты, персонажа, врагов
    private void drawObj(object? sender, PaintEventArgs e){
        Graphics g = e.Graphics;

        createMap(g);
        playAnimation(g);
        playEnemy(g);
    }

    // создание карты из массива
    private void createMap(Graphics g){
```

```
        for (int i = 0; i < w; i++)
            {
                for (int j = 0; j < h; j++)
                {
                    // объект небо
                    if (map[i, j] == 1)
                    {
                        g.DrawImage(skyPng, j * sideObj, i * sideObj, new
Rectangle(new Point(0, 0), new Size(sideObj, sideObj)), GraphicsUnit.Pixel);
                    }
                    // объект земля
                    else if (map[i, j] == 2)
                    {
                        g.DrawImage(groundPng, j * sideObj, i * sideObj, new
Rectangle(new Point(0, 0), new Size(sideObj, sideObj)), GraphicsUnit.Pixel);
                    }
                    // объект дом
                    else if (map[i, j] == 0)
                    {
                        xHome = j * sideObj;
                        g.DrawImage(homePng, j * sideObj, i * sideObj, new
Rectangle(new Point(0, 0), new Size(sideObj, sideObj)), GraphicsUnit.Pixel);
                    }
                    // объект небо
                    else if (map[i, j] == 4)
                    {
                        g.DrawImage(skyPng, j * sideObj, i * sideObj, new
Rectangle(new Point(0, 0), new Size(sideObj, sideObj)), GraphicsUnit.Pixel);
                    }
                }
            }
    }
}
```

Form1.Designer.cs

```
namespace hw;

partial class Form1
{
    /// <summary>
    ///  Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    ///  Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
    protected override void Dispose(bool disposing)
    {
```

```
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Windows Form Designer generated code

    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        this.gameTimer = new System.Windows.Forms.Timer(this.components);
        this.gameTimerMove = new System.Windows.Forms.Timer(this.components);
        this.gooseTimer = new System.Windows.Forms.Timer(this.components);
        this.label1 = new System.Windows.Forms.Label();
        this.SuspendLayout();

        // label1
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(10, 10);
        this.label1.Font = new System.Drawing.Font("Arial Narrow", 24F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(120, 40);
        this.label1.TabIndex = 4;
        this.label1.BackColor = Color.SkyBlue;
        this.label1.Text = "Очки: 0";

        // timer
        this.gameTimer.Enabled = true;
        this.gameTimerMove.Enabled = true;
        this.gooseTimer.Enabled = true;

        this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 20F);
        this.Controls.Add(this.label1);
        this.ControlBox = false;
        this.components = new System.ComponentModel.Container();
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.AutoScroll = true;
        this.StartPosition = FormStartPosition.CenterScreen;
        this.ClientSize = new System.Drawing.Size(1400, 900);
        this.DoubleBuffered = true;
        this.Paint += new System.Windows.Forms.PaintEventHandler(this.drawObj);
        this.ResumeLayout(false);
    }

    private System.Windows.Forms.Timer gameTimer;
    private System.Windows.Forms.Timer gameTimerMove;
    private System.Windows.Forms.Timer gooseTimer;
    private System.Windows.Forms.Label label1;
```

```
    #endregion
}
```

StartForm.cs

```
using hw;
namespace Hello
{
    public partial class StartForm : Form
    {
        static public System.Media.SoundPlayer opening = new
System.Media.SoundPlayer();    // музыка
        static public bool mut = true;  // состояние плеера (по умолчанию звук
выключен)
        public StartForm()
        {
            InitializeComponent();
            opening.SoundLocation = "souce\\start.wav";
            opening.PlayLooping();     // воспроизведение музыки по петле
            if(mut) opening.Stop();    // отключить музыку в зависимости от
состояния

            // события нажатия на кнопки

            button1.Click += button1_Click;
            button2.Click += button2_Click;
            button3.Click += button3_Click;
            button4.Click += button4_Click;
        }

        // открывает форму с игрой, отключает музыку стартовой формы

        private void button1_Click(object? sender, EventArgs e)
        {
            Form1 newForm = new Form1(mut);
            opening.Stop();
            newForm.Show();
        }

        // показывает правила игры и необходимые клавиши для игры

        private void button2_Click(object? sender, EventArgs e)
        {
            // вывод сообщения в виде отдельного окна
            MessageBox.Show("Необходимо защищать домик от гусей!\nХодите с
помощью A и D\nРычите на гусей -> и <-\nВернуться в меню Esc");
        }

        // настройка звука включить/выключить

        private void button3_Click(object? sender, EventArgs e)
        {
```

```
            // меняет состояние плеера в зависимости от предыдущего значения,
вкл/выкл музыки
            if(mut) {
                mut = false;
                opening.PlayLooping();
            }
            else{
                opening.Stop();
                mut = true;
            }
        }

        // закрывает приложение

        private void button4_Click(object? sender, EventArgs e)
        {
            Close();
        }
    }
}
```

StartForm.Designer.cs

```
namespace Hello;

partial class StartForm
{
    /// <summary>
    ///  Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    ///  Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Windows Form Designer generated code

    private void InitializeComponent()
    {
        // создание кнопок
        this.button1 = new System.Windows.Forms.Button();
```

```
        this.button2 = new System.Windows.Forms.Button();
        this.button3 = new System.Windows.Forms.Button();
        this.button4 = new System.Windows.Forms.Button();

        // создание подписи
        this.label1 = new System.Windows.Forms.Label();
        this.SuspendLayout();

        // label1
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(145, 10);
        this.label1.Font = new System.Drawing.Font("Arial Narrow", 24F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(120, 40);
        this.label1.TabIndex = 4;
        this.label1.Text = "Wolf Security";

        // button1
        this.button1.Location = new System.Drawing.Point(180, 100);
        this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(140, 40);
        this.button1.TabIndex = 0;
        this.button1.Text = "Игра";
        this.button1.UseVisualStyleBackColor = true;

        // button2
        this.button2.Location = new System.Drawing.Point(180, 170);
        this.button2.Name = "button2";
        this.button2.Size = new System.Drawing.Size(140, 40);
        this.button2.TabIndex = 0;
        this.button2.Text = "Инфо";
        this.button2.UseVisualStyleBackColor = true;

        // button3
        this.button3.Location = new System.Drawing.Point(180, 240);
        this.button3.Name = "button3";
        this.button3.Size = new System.Drawing.Size(140, 40);
        this.button3.TabIndex = 0;
        this.button3.Text = "Звук";
        this.button3.UseVisualStyleBackColor = true;

        // button4
        this.button4.Location = new System.Drawing.Point(180, 310);
        this.button4.Name = "button4";
        this.button4.Size = new System.Drawing.Size(140, 40);
        this.button4.TabIndex = 0;
        this.button4.Text = "Выход";
        this.button4.UseVisualStyleBackColor = true;
```

```
        this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 20F);
        // добавление кнопок и подписи
        this.Controls.Add(this.label1);
        this.Controls.Add(this.button1);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button3);
        this.Controls.Add(this.button4);
        // параметры формы
        this.ControlBox = false;
        this.components = new System.ComponentModel.Container();
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.AutoScroll = true;
        this.BackColor = Color.LightBlue;
        this.StartPosition = FormStartPosition.CenterScreen;
        this.ClientSize = new System.Drawing.Size(500, 500);
        this.DoubleBuffered = true;
        this.ResumeLayout(false);
    }
    private System.Windows.Forms.Label label1;
    private Button button1, button2, button3, button4;
    #endregion
}
```

Player.cs

```
namespace Game{
    class Player{
        public Image _anims;   // спрайты анимаций
        public int _x, _y;     // координаты
        public Size _scale;    // размер персонажа
        public int currFrame = 0;   // текущий кадр
        public int maxFrame = 3;    // максимальное количество кадров анимации
        public int currAnim = 3;   // текущая анимация
        public int speed = 10;   // скорость
        public bool life;     // состояние персонажа жив/мертв

        public bool roarR, roarL;   // состояние атаки атака направо/налево
        public Player(Size _scale, int _x, int _y, Image _anims){
            this._scale = _scale;
            this._x = _x;
            this._y = _y;
            this._anims = _anims;
            life = true;
            roarR = false;
            roarL = false;
        }

        // движение налево
        public void Left(){
            _x -= speed;
        }
```

```
        // движение направо
        public void Right(){
            _x += speed;
        }
    }
}
```

Enemy.cs

```
namespace entity{
    class Enemy{
        public Image _anims;  // спрайты
        public int _x, _y;    // координаты
        public Size _scale;   // размер
        public int currAnim = 0;  // текущая анимация
        public int speed;    // скорость передвижения
        public bool reverse;   // начальное направление
        public bool sad;    // состояние врага
        public Enemy(Size _scale, int _x, int _y, Image _anims, bool r, int speed
= 3){
            this._scale = _scale;
            this._x = _x;
            this._y = _y;
            this._anims = _anims;
            this.speed = speed;
            reverse = r;
            sad = false;
            if (reverse) currAnim = 1;
        }

        // перемещение влево
        public void Left(){
            _x -= speed;
        }

        // перемещение вправо
        public void Right(){
            _x += speed;
        }

        // для атаки врага
        public void pain(){
            sad = true;     // состояние "побежден"
            // смена направления движения и анимации в зависимости от начального
движения
            if (reverse) {
                currAnim = 2;
                reverse = false;
            }
            else {
                currAnim = 3;
                reverse = true;
```

```
            }
        }
    }
}
```

Результат

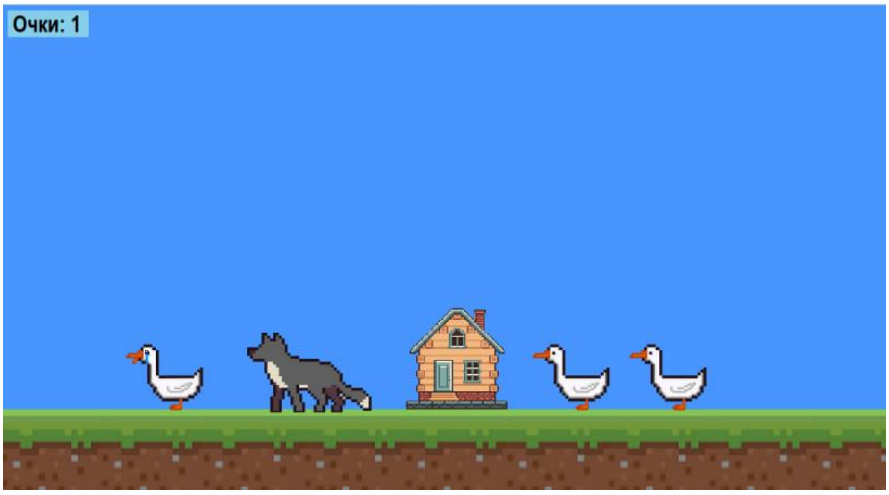
Стартовая форма. Меню игры.

(Рис.1)

Запуск игры при нажатии на кнопку «Игра».

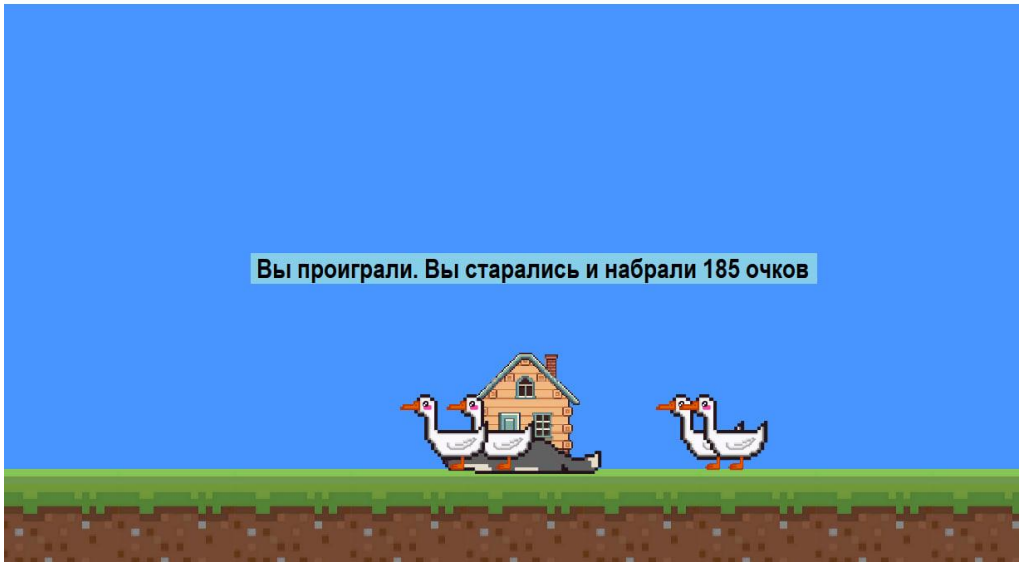
(Рис.2)

Атака врагов и перевод их в новое состояние.



(Рис.3)

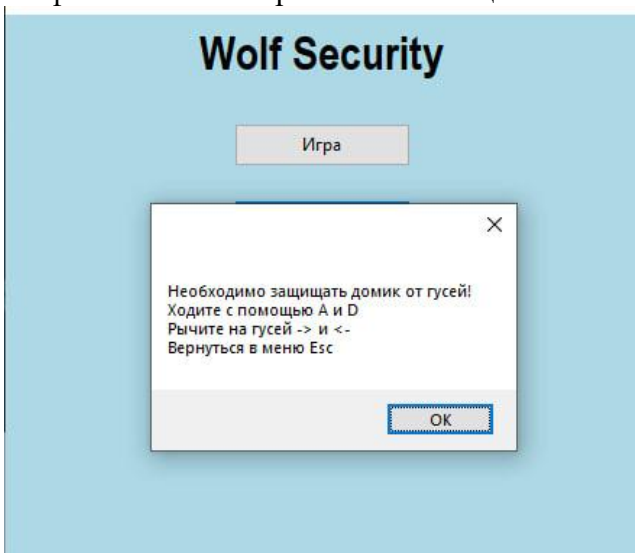
Проигрыш при столкновении персонажа с гусем или при достижении гусем дома.



(Рис.4)

При нажатии Escape происходит закрытие формы игры и переход к меню (см. Рис.1).

При нажатии кнопки «Инфо» на меню открывается сообщение.



(Рис.5)

При нажатии «ОК» или «крест» сообщение закрывается. При нажатии на кнопку «Звук» на форме меню включается или выключается звук приложения. Кнопка «Выход» стартовой формы закрывает приложение.