## СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Программирование – процесс создания компьютерных программ. По выражению одного из основателей языков программирования Никлауса Вирта, «Программы = алгоритмы + структуры данных» [1][2]. Программирование основывается на использовании языков программирования, на которых записываются исходные тексты программ[3]. Сегодня это возможность реализации своих идей и возможностей в различных сферах. Ранее количество сфер, в которых нужны были информационные технологии, было ограниченно. Сейчас же новые технологии приходят даже в те сферы жизни человека, о которых ранее нельзя было и подумать.

Данный курсовой проект предусматривает написание простейшего windows-приложения, компьютерная игра «Пошаговая стратегия».

Для решения поставленной задачи, разработка программы будет происходить в среде Microsoft Visual Studio 2022 с использованием языка программирования C# и спецификации Windows Forms.

Windows Forms – это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений.

В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

# 1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ

Пошаговая стратегия (англ. Turn-Based Strategy, TBS) – поджанр стратегических игр, в которых игровой процесс состоит из последовательности фиксированных моментов времени, именуемых ходами (или шагами), во время которых игроки совершают свои действия.

Компьютерные пошаговые стратегии происходят от настольных стратегических игр, в которых игроки, как правило, совершали действия по очереди. До 1990 года почти все стратегические компьютерные игры были пошаговыми. Большинство первых пошаговых стратегий были либо вариациями существующих настольных игр, либо в той или иной степени были вдохновлены ими.

Правила игры: Персонаж по имени Боб двигается по круговому подземелью. Персонаж имеет следующие в характеристики: атака, здоровье. Во время хода персонажа случайным образом появляется противник, с которым нужно сыграть в камень-ножницы-бумага. При победе противнику наносится урон, равный показателю атаки персонажа. В случае поражения противник наносит персонажу урон, равный показателю атаки противника. Также при ходе может появиться случайное событие, повышающее или понижающее показатели персонажа(атака,здоровье).При движении на 1 клетку персонаж получает 1 очко.

Следовательно, в нашей системе четко прослеживается необходимость в реализации следующих функций:

- генерация карты;
- движение персонажа;
- появление случайных событий;
- подсчёт счёта;
- игра в камень-ножницы-бумага;

## 2 ПРОЕКТИРОВАНИЕ

Проектирование – процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект – целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы. Наряду с анализом требований, является частью большой стадии жизненного цикла системы, называемой определением системы. Результаты этой стадии являются входной информацией для стадии реализации системы.

Главный этап на стадии проектирования программы – это определение ее ключевых составных частей, каждая из которых несет ответственность за определенные задачи, решаемые в процессе выполнения программы, и является относительно автономной либо частично зависимой от других частей[7].

При запуске программы на экране будет появляться игровое поле с расположенным на нём персонажем.

Для выполнения поставленной задачи должны быть спроектированы следующие основные модули:

- Game – модуль отрисовки игры;
- Menu – модуль появления меню;
- BattleWindow – модуль отрисовки окна битвы;
- ALifeUnit – характеристики основных юнитов;
- GameObject – используемые игровые объекты;
- Bob – характеристики героя;
- Rat – характеристики противника;
- World – генерация карты;
- Tile – генерация плитки карты;
- DebufWindow – генерация диалогового окна случайного события;

# 3 РЕАЛИЗАЦИЯ ПРОГРАММЫ

## 3.1 Модуль Game

В модуле Game находится 4 метода.

Метод Start(object sender, EventArgs e), который представлен в листинге 3.1, регистрирует и создает окно игры, отображая всю необходимую о персонаже информацию в окне.

Листинг 3.1 – Метод Start(object sender, EventArgs e)

```
private void Start(object sender, EventArgs e)
{
var world = new World(3,3,0,0,this);
_bob = new Bob(4, 6, 32, 32, this,world, 10, 3, 0, "Bob");
label2.Text = $"Attack / {_bob.Attack}";
label3.Text = $"Health / {_bob.Health}";
label4.Text = $"Score / {_bob.Score}";
      }
```

Второй метод button1_Click(object sender, EventArgs e) отвечает за передвижение персонажа по карте. При нажатии на кнопку программа проверяет здоровье персонажа, и, если оно больше 0, то персонаж перемещается на следующую клетку, вызывая случайное событие. Если здоровье персонажа равно 0, метод закрывает игровое окно и возвращает в меню. Метод представлен в листинге 3.2

Листинг 3.2 – Метод button1_Click(object sender, EventArgs e).

```
private void button1_Click(object sender, EventArgs e)
{
if (_bob.Health > 0)
{
StepSound.Play();
for (var i = 0; i < _gameObjects.Count; i++)
_gameObjects[i]?.Update();
_bob.Score += 1;
pictureBox1.Invalidate();
label2.Text = $"Attack / {_bob.Attack}";
label3.Text = $"Health / {_bob.Health}";
label4.Text = $"Score / {_bob.Score}";
}
else
{
Menu.AddToScoreList(_bob.Score);
Menu.Show();
Close();
}
}
```

Третий метод Draw(object sender, PaintEventArgs e) представлен в листинге 3.3. Метод отвечает за отрисовку игрового процесса в поле PictureBox.

Листинг 3.3 – метод Draw(object sender, PaintEventArgs e).

```
private void Draw(object sender, PaintEventArgs e)
{
var graphics = e.Graphics;
for (var i = 0; i < _gameObjects.Count; i++)
_gameObjects[i]?.Draw(graphics);
}
```

### 3.2 Модуль Menu

Данный модуль является стартовым модулем, с которого начинается запуск программы. Модуль содержит 4 метода

Метод button1_Click(object sender, EventArgs e) представлен в листинге 3.4 и является методом, вызываемым при нажатии на кнопку старт.

Листинг 3.4 – Метод button1_Click(object sender, EventArgs e)

```
private void button1_Click(object sender, EventArgs e)
{
new Game(this);
MainSound.Stop();
Hide();
}
```

Второй метод – button2_Click(object sender, EventArgs e)представлен в листинге 3.5.Метод отвечает за выход из программы при нажатии на кнопку.

Листинг 3.5 – Метод button2_Click(object sender, EventArgs e)

```
private void button2_Click(object sender, EventArgs e)
{
Application.Exit();
}
```

Третий метод – AddToScoreList(int score) предназначен для добавления счёта в таблицу лидеров, расположенную в форме Menu. Метод представлен в листинге 3.6

Листинг 3.6 – Метод AddToScoreList(int score)

```
public void AddToScoreList(int score)
{
listBox1.Items.Clear();
if (_scores.ContainsKey(textBox1.Text == string.Empty ? "Unknow" :
textBox1.Text))
_scores[textBox1.Text == string.Empty ? "Unknow" : textBox1.Text] =
_scores[textBox1.Text == string.Empty ? "Unknow" : textBox1.Text] <
score ? score : _scores[textBox1.Text == string.Empty ? "Unknow" :
textBox1.Text];
Else
```

Продолжение листинга 3.6

```
_scores.Add(textBox1.Text == string.Empty ? "Unknow" :
textBox1.Text, score);
foreach (var s in _scores.OrderByDescending(x => x.Value))
listBox1.Items.Add($"{s.Key}:{s.Value}\n");
}
```

Четвёртый метод – checkBox1_CheckedChanged(object sender, EventArgs e) отвечает за фоновую музыку меню.Метод представлен в листинге 3.7

Листинг 3.7 – Метод checkBox1_CheckedChanged(object sender, EventArgs e)

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
if(checkBox1.Checked == true)
{
MainSound.PlayLooping();
}
else
{
MainSound.Stop();
}
}
```

### 3.3 Модуль Battle Window

Данный модуль содержит файл. Battle Window взаимодействует с модулями .файл реализован класс Battle Window, в котором созданы следующие функции:

– void AddUnits(ALifeUnit player, ALifeUnit enemy) – метод добавления персонажей на форму представлен в листинге 3.8;

– void Draw(object sender, PaintEventArgs e) – метод визуального отображения персонажей на форме в поле, представлен в листинге 3.9;

– void button1_Click(object sender, EventArgs e) – метод выбора камня, представлен в листинге 3.10;

– void button2_Click(object sender, EventArgs e)– метод выбора ножниц, представлен в листинге 3.11;

– void button3_Click(object sender, EventArgs e)– метод выбора бумаги, представлен в листинге 3.12;

Листинг 3.8 – метод AddUnits(ALifeUnit player, ALifeUnit enemy)

```
public void AddUnits(ALifeUnit player, ALifeUnit enemy)
{
_player = player;
_enemy = enemy;
button1.Show();
button2.Show();
```

Продолжение листинга 3.8

```
button3.Show();
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
}
```

Листинг 3.9 – метод Draw(object sender, PaintEventArgs e)

```
private void Draw(object sender, PaintEventArgs e)
{
var graphics = e.Graphics;
float playerSize = 256;
float enemySize = 256;
graphics.DrawImage(_ratSprite, 400,0, (int)playerSize,
(int)playerSize);
graphics.DrawImage(_playerSprite, 24, 24, enemySize, enemySize);
}
```

Листинг 3.10 – метод button1_Click(object sender, EventArgs e)

```
private void button1_Click(object sender, EventArgs e)
{
var ratChoose = _random.Next(0,3);
switch (ratChoose) {
case 0:
DrawSound.Play();
new DebufWindow("Ничья!");
break;
case 1:
GoodSound.Play();
new DebufWindow($"Победа! Вы наносите {_enemy.Name}
{_player.Attack} ед. урона");
_enemy.Health -= _player.Attack;
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
break;
case 2:
BadSound.Play();
new DebufWindow($"Поражение! {_enemy.Name} наносит вам
{_enemy.Attack} ед. урона");
_player.Health -= _enemy.Attack;
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
break;
}
if (_enemy.Health <= 0)
{
WinSound.Play();
new DebufWindow("You Win");
Close();
}
else if(_player.Health <=0)
```

```
{
DefeatSound.Play();
new DebufWindow("You Lose((:(");

Close();
}

pictureBox1.Invalidate();
}
```

Листинг 3.11 – метод button2_Click(object sender, EventArgs e)

```
private void button2_Click(object sender, EventArgs e)
{
var ratChoose = _random.Next(0, 3);
switch (ratChoose)
{
case 0:
BadSound.Play();
new DebufWindow($"Поражение! {_enemy.Name} наносит вам
{_enemy.Attack} ед. урона");
_player.Health -= _enemy.Attack;
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
break;
case 1:
DrawSound.Play();
new DebufWindow("Ничья!");
break;
case 2:
GoodSound.Play();
new DebufWindow($"Победа! Вы наносите {_enemy.Name}
{_player.Attack} ед. урона");
_enemy.Health -= _player.Attack;
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
break;
}
if (_enemy.Health <= 0)
{
WinSound.Play();
new DebufWindow("You Win");
Close();
}
else if(_player.Health <=0)
{
DefeatSound.Play();
new DebufWindow("You Lose((:(");
Close();
}
```

```
pictureBox1.Invalidate();
}
```

Листинг 3.12 – метод button3_Click(object sender, EventArgs e)

```
private void button3_Click(object sender, EventArgs e)
{
var ratChoose = _random.Next(0, 3);
switch (ratChoose)
{
case 0:
GoodSound.Play();
new DebufWindow($"Победа! Вы наносите {_enemy.Name}
{_player.Attack} ед. урона");
_enemy.Health -= _player.Attack;
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
break;
case 1:
BadSound.Play();
new DebufWindow($"Поражение! {_enemy.Name} наносит вам
{_enemy.Attack} ед. урона");
_player.Health -= _enemy.Attack;
label1.Text = $"Здоровье: {_player.Health}";
label2.Text = $"Здоровье: {_enemy.Health}";
break;
case 2:
DrawSound.Play();
new DebufWindow("Ничья!");
break;
}
if (_enemy.Health <= 0)
{
WinSound.Play();
new DebufWindow("You Win");
Close();
}
else if(_player.Health <=0)
{
DefeatSound.Play();
new DebufWindow("You Lose((:(");
Close();
}
pictureBox1.Invalidate();
}
```

### 3.4 Модуль ALifeUnit

Данный модуль отвечает за хранение информации об используемых в игре юнитах. В ALifeUnit реализован абстрактный класс ALifeUnit,который связан с модулем GameObject и содержит следующие функции и характеристики.Модуль представлен в листинге 3.13.

Листинг 3.13 – Файл ALifeUnit.cs

```
using System.Media;
namespace RedAllert
{
public abstract class ALifeUnit : GameObject
{
public string Name { get; set; }
internal int Health { get; set; }
internal int Attack { get; set; }
internal int Score { get; set; }
public ALifeUnit(int x, int y, int width, int height, Game form,
int health, int attack,int score, string name) : base(x, y, width,
height, form)
{
Health = health;
Attack = attack;
Name = name;
Score = score;
}
      public abstract void BattleDraw(Graphics graphics);
      }
      }
```

### 3.5 Модуль GameObject

Данный модуль является абстрактным классом от которого должны наследоваться все классы которые могут находиться в игре. Представлен в листинге 3.14

Листинг 3.14 – Абстрактный класс Game Object

```
      public abstract class GameObject
      {
      protected Game Form;
      public int X;
      public int Y;
      public int Width;
      public int Height;
      public GameObject(int x, int y, int width, int height, Game
form)
```

```
        {
        X = x;
        Y = y;
```

Продолжение листинга 3.14

```
        Width = width;
        Height = height;
        Form = form;
        Form.CreateNewObject(this);
        }
        public abstract void Draw(Graphics graphics);
        public abstract void Update();
        public static Image RotateImage(Image img, float
rotationAngle)
{
//create an empty Bitmap image
Bitmap bmp = new Bitmap(img.Width, img.Height);
//turn the Bitmap into a Graphics object
Graphics gfx = Graphics.FromImage(bmp);
//now we set the rotation point to the center of our image
gfx.TranslateTransform((float)bmp.Width / 2, (float)bmp.Height /
2);
//now rotate the image
gfx.RotateTransform(rotationAngle);
gfx.TranslateTransform(-(float)bmp.Width / 2, -(float)bmp.Height /
2);
//set the InterpolationMode to HighQualityBicubic so to ensure a
high
//quality image once it is transformed to the specified size
gfx.InterpolationMode = InterpolationMode.HighQualityBicubic;
//now draw our new image onto the graphics object
gfx.DrawImage(img, new Point(0, 0));
//dispose of our Graphics object
gfx.Dispose();
//return the image
return bmp;
}
}
```

### 3.6 Модуль Bob

Модуль содержи файл Bob.cs. В файле содержится класс Bob. Этот класс используется только Game.cs.

Файл содержит информацию о герое и необходимые для создания персонажа характеристики

В файле реализованы следующие методы:

– void Draw(Graphics graphics) представлен в листинге 3.15;

– void BattleDraw(Graphics graphics) представлен в листинге 3.16;
– void Update() представлен в листинге 3.17;

Листинг 3.15 – метод void Draw(Graphics graphics)

```
public override void Draw(Graphics graphics)
{
graphics.DrawImage(Sprite, new Point(X * Width, Y * Height));
}
```

Листинг 3.16 – метод void BattleDraw(Graphics graphics)

```
public override void BattleDraw(Graphics graphics)
{
graphics.DrawImage(Sprite, X * Width, Y * Height);
}
```

Листинг 3.17 – метод void Update()

```
public override void Update()
{
_world.MoveToNextTile(this);
var random = new Random();
var cube = random.Next(1, 10);
switch (cube)
{
case 1:
if (Attack > 0)
{
Attack -= 1;
DebufSound.Play();
new DebufWindow("О нет... Вы утомились. Ваша атака понижается на 1
ед.");
}
break;
case 2:
Attack += 1;
BufSound.Play();
new DebufWindow("Вы нашли странное зелье. Ваша атака повышается на
1 ед.");
break;
case 3:
Health += 1;
BufSound.Play();
new DebufWindow("Вы нашли исцеляющий свиток. Ваше здоровье
повышается на 1 ед.");
break;
case 4:
if (Health > 1)
{
Health -= 1;
DebufSound.Play();
new DebufWindow("О нет... Вас укусил паук. Ваше здоровье падает на
1 ед.");
```

```
}
break;
case 5:
if ((Health == 1 && this.Attack < 2) || (this.Health < 4 &&
this.Attack == 0))
{
Health = 1;
Attack = 10;
new DebufWindow("Вы в ярости.Атака увеличена");
}
break;
case 6:
AttentionSound.Play();
new DebufWindow("На вас напала крыса");
var battleWindow = new BattleWindow();
battleWindow.AddUnits(this, new Rat(0, 0, 0, 0, Form, 0, 0, 0,
"Nafanya"));
break;
}
if (Health <= 0)
{
DefeatSound.Play();
new DebufWindow("You died");
Form.Menu.AddToScoreList(Score);
Form.Menu.Show();
Form.Close();
}
}
```

### 3.7 Модуль Rat

Данный модуль отвечает за хранение информации о противнике и . Содержит в себе конструктор Rat, представленный в листинге 3.18.

Листинг 3.18 – конструктор Rat

```
public Rat(int x, int y, int width, int height, Game form, int
health, int attack,int score, string name) : base(x, y, width,
height, form, health, attack,score, name)
{
var random = new Random();
Health = random.Next(6, 8);
Attack = random.Next(2, 4);
}
```

### 3.8 Модуль World

Модуль World используется программой для генерации карты, по которой будет передвигаться персонаж. Содержит 3 метода

Первый метод – SetToTile(GameObject gameObject) для подключения клетки хода к карте. Представлен в листинге 3.19.

Второй метод – MoveToNextTile(GameObject gameObject) используется программой для перемещения персонажа на следующую клетку.Представлен на листинге 3.20.

Третий метод – InitMap() используется для генерации карты. Карта представляет собой прямоугольник с заданными случайными значениями длины и ширины. Метод представлен в листинге 3.21.

Листинг 3.19 – метод SetToTile(GameObject gameObject)

```
public void SetToTile(GameObject gameObject) {
var index = _random.Next(_tiles.Count);
_gameObjectInWorld.Add(new Tuple<GameObject, int>(gameObject,
index));
_gameObjectInWorld.Last().Item1.X = _tiles[index].X;
_gameObjectInWorld.Last().Item1.Y = _tiles[index].Y;
}
```

Листинг 3.20 – метод MoveToNextTile(GameObject gameObject)

```
public void MoveToNextTile(GameObject gameObject)
{
var findedObject = _gameObjectInWorld.FindIndex(x => x.Item1 ==
gameObject);
if (_gameObjectInWorld[findedObject].Item2 == _tiles.Count - 1)
{
_gameObjectInWorld[findedObject].Item1.X = _tiles[0].X;
_gameObjectInWorld[findedObject].Item1.Y = _tiles[0].Y;
_gameObjectInWorld[findedObject] = new Tuple<GameObject,
int>(_gameObjectInWorld[findedObject].Item1, 0);
return;
}
_gameObjectInWorld[findedObject].Item1.X =
_tiles[_gameObjectInWorld[findedObject].Item2 + 1].X;
_gameObjectInWorld[findedObject].Item1.Y =
_tiles[_gameObjectInWorld[findedObject].Item2 + 1].Y;
_gameObjectInWorld[findedObject] = new Tuple<GameObject,
int>(_gameObjectInWorld[findedObject].Item1,
_gameObjectInWorld[findedObject].Item2 + 1);
}
```

Листинг 3.21 – метод InitMap()

```
private void InitMap()
{
Random xr = new Random();
var xCount = xr.Next(3,10);
Random yr = new Random();
var yCount = yr.Next(3,8);
```

```
for (int x = X; x <= xCount + X; x++)
_tiles.Add(new Tile(x, Y, 32, 32, Form,0));
for (int y = Y + 1; y <= yCount + Y; y++)
_tiles.Add(new Tile(X + xCount, y, 32, 32, Form, 90));
for (int x = X + xCount; x >= X; x--)
_tiles.Add(new Tile(x, Y + yCount + 1, 32, 32, Form, 0));
for (int y = Y + yCount; y >= Y; y--)
_tiles.Add(new Tile(X, y, 32, 32, Form, 90));
}
```

### 3.9 Модуль Tile

Модуль является наследуемым классом от Game Object. В классе Tile содержатся все характеристики, необходимые программе для объявления клетки и последующем расположении ее на карте.

Листинг 3.22– класс Tile

```
public class Tile : GameObject
{
public Image Sprite =
Image.FromFile("../../../resources/tile.png");
public Tile(int x, int y, int width, int height, Game form,float
rotation) : base(x, y, width, height, form)
{
Sprite = RotateImage(Sprite, rotation);
}
public override void Draw(Graphics graphics)
{
graphics.DrawImage(Sprite, X * Width, Y * Height);
}
public override void Update() {}
}
}
```

### 3.10 Модуль DebufWindow

Модуль DebufWindow отвечает за появление диалоговых окон, содержащих информацию о случайных событиях, появляющихся по ходу игры. DebufWindow содержит в себе один метод void button1_Click(object sender, EventArgs e), используемый программой для выхода из диалогового окна события. Метод представлен в листинге 3.23

Листинг 3.23 – метод void button1_Click(object sender, EventArgs e)

```
private void button1_Click(object sender, EventArgs e)
{Close();}
```

# 4 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Тестирование программы – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Тестирование программы проводится с помощью метода черного ящика.

Тестирование методом черного ящика – метод тестирования программного обеспечения, который предполагает, что внутренняя структура системы неизвестны тестировщику. Выбираются входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же известен, каким должен быть результат этой обработки. Тестирование черного ящика – углубление во внутренне устройство системы, за пределы ее внешних интерфейсов

Недостатки:

– тестирование не может производиться на ранних этапах: имеется необходимость ожидание создания пользовательского интерфейса;

– нельзя провести более тщательное тестирование, с покрытием большого количества путей выполнения программы.

Преимущества:

– для выполнения тестирования черного ящика не требуется большое количество специальных знаний;

– при использовании автоматизации тестирования на этом уровне, поддержка тестовых скриптов может оказаться достаточно полезной, если программа часто изменяется.

При запуске программы появляется меню. Пользователь видит фон с кнопками старта игры и выхода из игры,таблицу лидеров и поле для ввода имени (**Ошибка! Источник ссылки не найден.**).

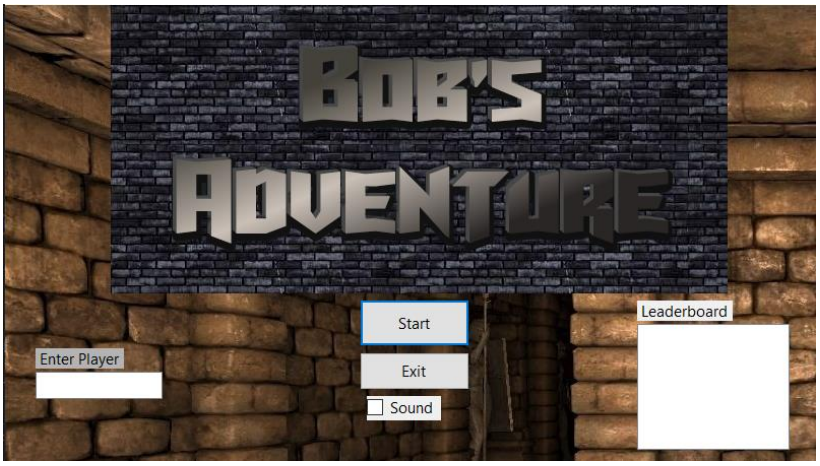


Рисунок 4.1 – Меню игры

При нажатии на кнопку «Start» появляется игровое поле,показатели персонажа(Рисунок 4.2)

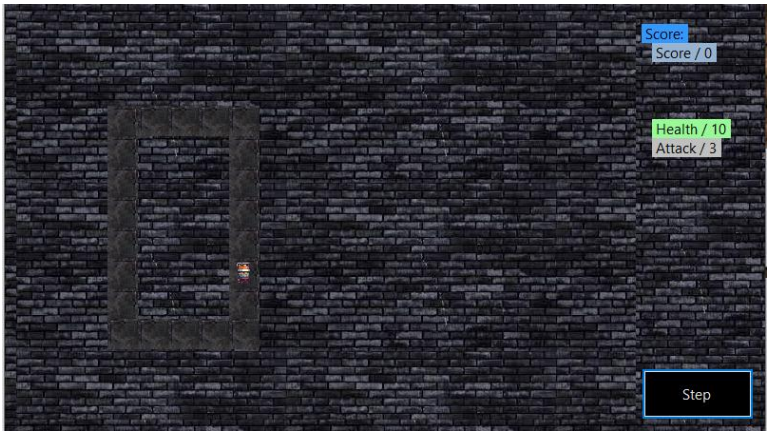


Рисунок 4.2 – Игровое поле

При нажатии на кнопку «Step» игрок перемещается на 1 клетку по часовой стрелке и ему случайным образом выпадает событие(Рисунок 4.3)

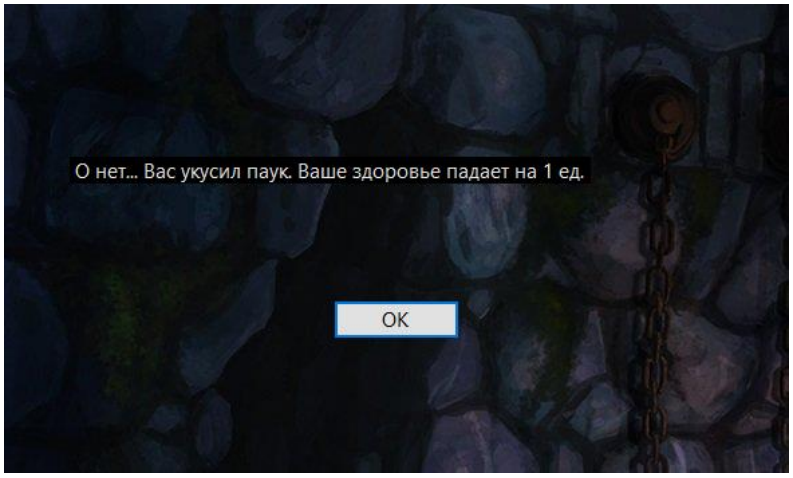


Рисунок 4.3 – Диалоговое окно случайного события

В процессе игры появляется случайное событие – нападение крысы(Рисунок 4.4)

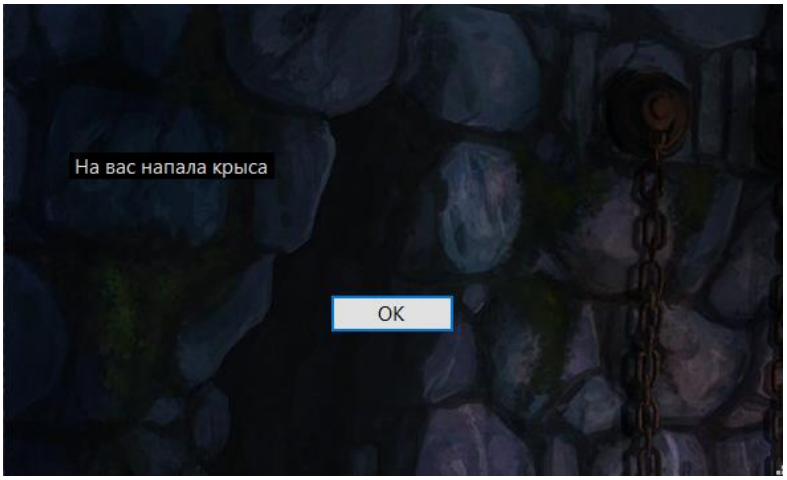


Рисунок 4.4 – Нападение крысы

После нажатия на кнопку «ОК» появляется окно игры в камень-ножницы-бумага(Рисунок 4.5).Игрок выбирает один из предметов, и в случае победы наносит урон противнику (Рисунок 4.6),в случае поражения противник наносит урон герою(Рисунок 4.7).В случае выбора одинаковых предметов(Ничья) урон не наносится ни противнику, ни игроку(Рисунок 4.8).Каждое из действий сопровождается звуковым сигналом.


Рисунок 4.5 – Окно битвы

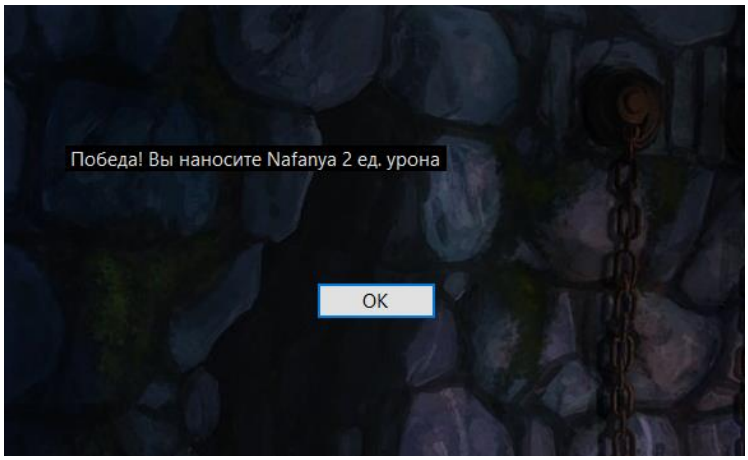


Рисунок 4.6 – Победа

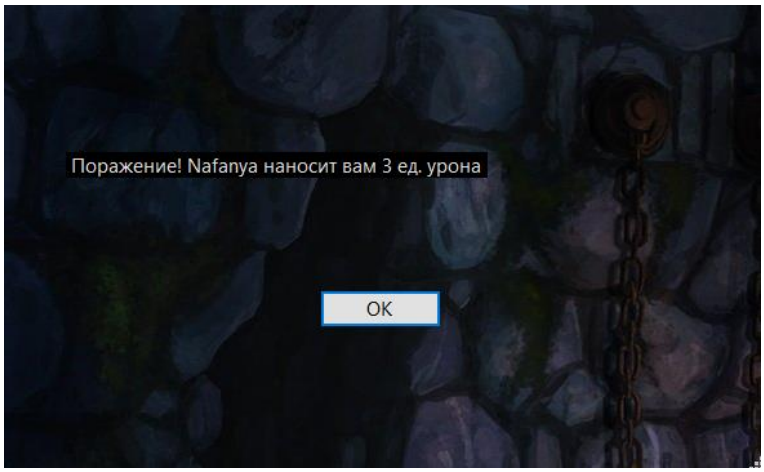


Рисунок 4.7 – Поражение


Рисунок 4.8 – Ничья

В случае, когда здоровье противника становится равным 0 появляется окно победы и игра продолжается. В случае, когда игрок теряет все очки здоровья игра заканчивается (Рисунок 4.9) и отправляет игрока в Главное меню, где в поле LeaderBoard появляется информация об игроке и набранных им очках (Рисунок 4.10).

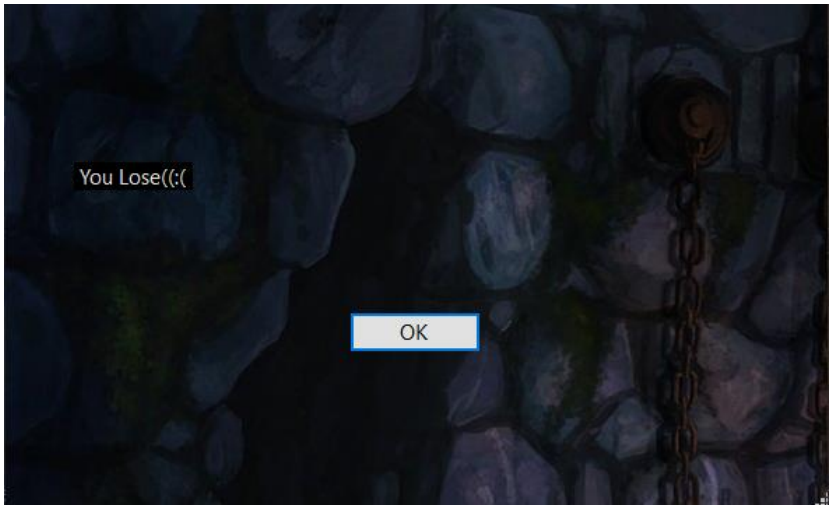


Рисунок 4.9 – Окно поражения

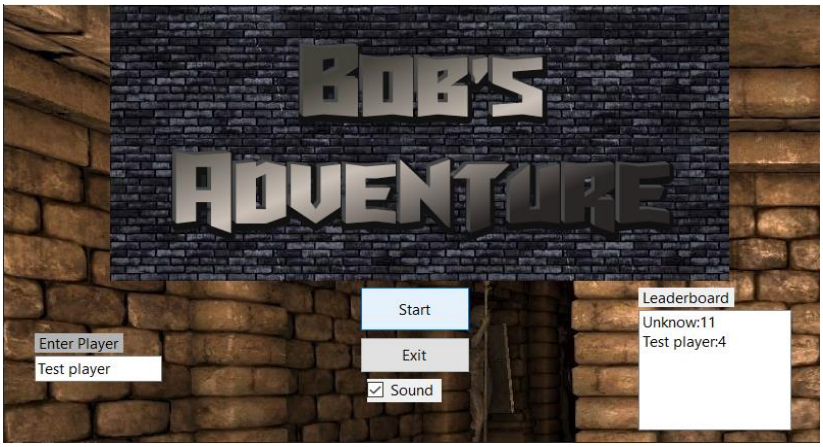


Рисунок 4.10 –Обновленное окно меню

Результаты тестирования показаны в таблице А.1 (приложение А).

В ходе тестирования программы не было замечено сбоев или аварийного завершения работы игрового приложения, что свидетельствует о его полной работоспособности.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе была поставлена задача – на основе полученных в ходе изучения курса «Конструирование программного обеспечения» знаний создать работоспособное программное обеспечение, представляющее собой игровое приложение «Пошаговая стратегия».

В игровом приложении существует возможность:

- генерация карты;
- движение персонажа;
- появление случайных событий;
- набор очков;
- игра в камень-ножницы-бумага;

В ходе тестирования не было обнаружено сбоев в программе или аварийного завершения работы ПО, что говорит о полной функциональности игрового приложения и выполнении главной задачи курсовой работы.

В итоге, можно сказать, что программа реализована в соответствии всем требованиям, протестирована надлежащим образом, работает стабильно и может использоваться для решения поставленной задачи. Поставленная задача выполнена в полной мере.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985;
2. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона + CD. М.: ДМК Пресс, 2010. ISBN 978-5-94074-584-6, 0-13-022005-9;
3. Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Программирование. Дата доступа: 30.11.2022;
4. Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Тестирование. Дата доступа: 30.11.2022;
5. Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Пошаговая_стратегия. Дата доступа: 30.11.2022;
6. Фридман, А. C/C++. Алгоритмы и приемы программирования / А. Фридман. – М. : Бином, 2007. – 560 с.
7. Липпман Б. Язык программирования C++. Базовый курс/ Б. Липпман, 2001 – 1104с.

**ПРИЛОЖЕНИЕ А**
(обязательное)
**Таблица тестирования программы**

Таблица А.1 – Способы проверок с указанием ожидаемых результатов испытаний

| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
|---|---|---|---|
| Запуск клиентского модуля | Вход в игровое приложение | Открытие главного меню | Тест пройден успешно |
| Загрузка игрового процесса | Нажатие на опцию главной страницы «Start» | Открытие окна игры | Тест пройден успешно |
| Выход из игрового приложения | Нажатие на опцию главной страницы «Exit» | Закрытие игрового приложения | Тест пройден успешно |
| Управление игроком | Нажатие на клавишу Step | Движение игрока | Тест пройден успешно |
| Набор очков | Нажатие на клавишу Step | Счетчик очков повышается на одну единицу | Тест пройден успешно |
| Появление событий | Нажатие на клавишу Step | Открытие случайных диалоговых окон | Тест пройден успешно |
| Появление окна битвы | Выпадение события «Rat want’s to kill you» и нажатие на клавишу OK | Открытие окна битвы с противником | Тест пройден успешно |
| Игра в камень-ножницы-бумага | Нажатие на клавиши Камень , Ножницы, Бумага. | Появление диалоговых окон и нанесение урона противнику/персонажу | Тест пройден успешно |
| Сохранение результата в таблицу лидеров | При смерти игрока его счет должен быть больше чем его предыдущий | Имя игрока и его счёт появляются в таблице лидеров в главном меню | Тест пройден успешно |