СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире программирование стало неотъемлемой частью нашей жизни. Мы видим его абсолютно во всех сферах жизни, поэтому будет полезно овладеть навыками программирования не только для заработка денег, но и для реализации своих идей и возможностей в различных сферах.

В ходе данного курсового проекта было необходимо сделать windows приложение, компьютерную игру в жанре «ККИ» (коллекционная карточная игра).

Для реализации проекта было использовано программное обеспечение Microsoft Visual Studio 2022 на языке 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 [1].

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].

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ

Коллекционная карточная игра (ККИ, англ. Collectible Card Game, Trading Card Game) – разновидность настольных и компьютерных игр. В отличие от традиционных карточных игр, коллекционные карточные игры используют специальные карты, схожие с коллекционными карточками. В крупных коллекционных карточных играх могут существовать тысячи различных карт. Нельзя приобрести все существующие карты одновременно; вместо этого от игроков ожидается, что они будут приобретать карты небольшими наборами и составлять свои индивидуальные колоды. Собственно игра между игроками ведётся с использованием различных правил, которые могут сильно различаться для разных игр [2].

Правила игры: вначале игры, каждому из игроков выдаётся четыре карты в руку. Максимальное число карт в руке - восемь. Игроки ходят по очереди, выставляя карты на стол. сначала первый игрок, потом второй. Каждая карат имеет стоимость (ману), урон и жизни. Мана начинает с единицы и каждый ход восстанавливается и увеличивает своё максимальное количество на один. Так же игроки берут по одной карте в каждом начале хода, пока у них не закончиться колода. По завершению хода происходит сражение, в котором выложенные карты на столе сражаются, друг с другом, теряя столько жизней, сколько было урона у противника. На каждой линии свои сражения. Если на какой-то линии не будет карты оппонента, то удар пройдёт по самому игроку. Проиграет тот, кто первый лишиться всего здоровья.

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

- коллекционирование карт;
- интеллект противника;
- возможность выкладывания карт на стол;
- процесс сражения;
- возможность поставить игру на паузу.

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 [3].

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 [4].

При запуске программы на экране будет появляться главное меню, где будут настройки, коллекция, кнопка выходи, и начало игры, это всё есть на рисунке 4.1. После нажатия начала игры, игроку и его оппоненту выдадут по четыре карты в руки, и противник сделает первый ход. Игровое поле можно увидеть на рисунке 4.2. На нем должна отображаться мана равное единице, а также жизни игрока и противника равные тридцати. На экране должна быть кнопка паузы, кнопка следующего раунда, игровое поле, рука игрока и противника и карта взятая в руку.

Для выполнения поставленной задачи должны быть спроектированы следующие основные чисти программы:

- initializ – функция инициализации карт;
- PlHand, EnemyHand – функции создания карт в руках у игроков;
- Card_Click, Line_Click – функции выкладывания карт на стол;
- EnemyTurn – функция хода противника;
- EndTurn_Click – функция окончания раунда;
- Fight – функция сражения.

3 РЕАЛИЗАЦИЯ ПРОГРАММЫ

3.1 Функция initializ

Функция initializ представлена в листинге 3.1, в ней происходит инициализация карт, здесь записываться характеристики карт такие как, hp, mana, dmg, а также картинка карты и была ли она использована. Функция FromFile выгружает картинку из файлов.

Листинг 3.1 – Функция initializ()

```
void initializ()
        {
            Line[0].busy = true;
            Line[1].busy = true;
            Line[2].busy = true;
            Line[3].busy = true;
            Line[4].busy = true;
            EnemyLine[0].busy = true;
            EnemyLine[1].busy = true;
            EnemyLine[2].busy = true;
            EnemyLine[3].busy = true;
            EnemyLine[4].busy = true;

            Card[0].Using = true;
            Card[0].PicterCard =
Image.FromFile("../kki/Cards/test.jpg");

            Card[1].Using = true;
            Card[1].PicterCard =
Image.FromFile("../kki/Cards/Ilya.jpg");
            Card[1].HP = 9;
            Card[1].DMG = 9;
            Card[1].MANA = 9;

            Card[2].Using = true;
            Card[2].PicterCard =
Image.FromFile("../kki/Cards/рыцарь3.jpg");
            Card[2].HP = 8;
            Card[2].DMG = 4;
            Card[2].MANA = 6;

            Card[3].Using = true;
            Card[3].PicterCard =
Image.FromFile("../kki/Cards/рыцарь2.jpg");
            Card[3].HP = 5;
            Card[3].DMG = 4;
            Card[3].MANA = 4;
```

3.2 Функции PlHand, EnemyHand

Функции PlHand и EnemyHand по структуре одинаковые. Они используя структуру Randm() генерируют случайное расположение карт в колоде и выдают первые четыре карты в руки игрокам. Код функции PlHand можно посмотреть в листинге 3.2

Листинг 3.2 – Функция PlHand()

```
void PlHand()
        {
            Random rnd = new Random();
            int check = 0;
            bool flag = true;
            for (int i = 0; i < 29; i++)
            {
                check = rnd.Next(1, 28);
                for (int j = 0; j != 28; j++)
                {
                    if (deck[j] == check && flag)
                    {
                        check = 1;
                        j = 0;
                        flag = false;
                    }
                    if (deck[j] == check && !flag)
                    {
                        check++;
                        j = -1;
                    }
                }
                flag = true;
                deck[i] = check;
            }
            for (int i = 0; i < 4; i++)
            {
                HandCard[i] = Card[deck[i]];
            }
        }
```

3.3 Функции Card_Click, Line_Click

Функций Card_Click (листинг 3.3) в коде восемь на каждую карту в руке, а функций Line_Click (листинг 3.4) пять на каждую линию. Они отвечают за выбор и выкладывание карт на игровое поле.

Листинг 3.3 – Функция Card_Click()

```
private void card2_Click(object sender, EventArgs e)
        {
            cardclick = true;
            TempCard.BackgroundImage = card2.BackgroundImage;
            TempCard.Visible = true;
            card2.Visible = false;
            if (num != 1) PlayCard(num);
            num = 1;
            MANAupdate();
        }
```

Листинг 3.4 – Функция Line_Click()

```
private void line2_Click(object sender, EventArgs e)
        {
            if (cardclick)
            {
                if (Line[1].busy)
                {
                    if (mana >= HandCard[num].MANA)
                    {
                        HandCard[num].Using = false;
                        line2.Cursor = DefaultCursor;
                        line2.BackgroundImage =
TempCard.BackgroundImage;
                        TempCard.Visible = false;
                        Line[1].busy = false;
                        cardclick = false;
                        Line[1].LineDmg = HandCard[num].DMG;
                        Line[1].LineHp = HandCard[num].HP;
                        LineDmg2.Text = Line[1].LineDmg.ToString();
                        LineHp2.Text = Line[1].LineHp.ToString();
                        LineDmg2.Visible = true;
                        LineHp2.Visible = true;
                        mana -= HandCard[num].MANA;
                        MANAupdate();
                    }
                    else
                    {
                        Mana.Text = "No";
                        Mana.Location = new Point(109, 344);
                    }
                }
            }
         }
```

В функциях Card_Click и Line_Click используются функции PlayCard(int numr) (листинг 3.5) и MANAupdate (листинг 3.6) которые правильно отображают карты в руке и обновляют ману соответственно.

Листинг 3.5 – Функция PlayCard()

```
void PlayCard()
        {
            if (numr == 0) card1.Visible = HandCard[0].Using;
            if (numr == 1) card2.Visible = HandCard[1].Using;
            if (numr == 2) card3.Visible = HandCard[2].Using;
            if (numr == 3) card4.Visible = HandCard[3].Using;
            if (numr == 4) card5.Visible = HandCard[4].Using;
            if (numr == 5) card6.Visible = HandCard[5].Using;
            if (numr == 6) card7.Visible = HandCard[6].Using;
            if (numr == 7) card8.Visible = HandCard[7].Using;
            MANAupdate();
        }
```

Листинг 3.6 – Функция MANAupdate()

```
void MANAupdate()
        {
            if (mana < 10)
            {
                Mana.Location = new Point(137, 343);
                Mana.Text = mana.ToString();
            }
            if (mana >= 10)
            {
                Mana.Location = new Point(123, 343);
                Mana.Text = mana.ToString();
            }
        }
```

3.4 Функция EnemyTurn

Фунция EnemyTurn отвечает за интеллект противника и представлен на листинге 3.7, интеллект является алгоритмом. Если у игрока нет никаких карт на столе то противник ищет самую сильную карту с помощью функции FoundPower() (листинг 3.8) и играет её на свободную линию. Если у игрока на столе одна и больше карт то противник ищет самую слабую карту функцией FoundWeakness() (листинг 3.9) и играет её перед картой игрока, если все карты перед игроком заставлены, а у противника осталась мана, он ищет самую сильную и играет её на свободную линию.

Листинг 3.7 – Функция EnemyTurn()

```
void EnemyTurn(){
            int mem = 0, temp = -1, EnemyMem = 0;
            int[] HaveUnit = new int[5];
            int[] EnemyNoHaveUnits = new int[5];
            for (int i = 0, j = 0, k = 0; i < 5; i++)
            {
```

```
                if (!Line[i].busy && EnemyLine[i].busy)
                {
                    mem++;
                    HaveUnit[j] = i;
                    j++;
                }
                else if(EnemyLine[i].busy)
                {
                    EnemyMem++;
                    EnemyNoHaveUnits[k] = i;
                    k++;
                }

            }
            if (EnemyMem == 0) { }
            else
            {
                if (mem == 0)
                {
                    temp = FoundPower();
                    if (temp == -1) { }
                    else
                    {
                        if (EnemyMem > 0)
                        {
                            EnemyLine[EnemyNoHaveUnits[0]].LineDmg
= EnemyHandCard[temp].DMG;
                            EnemyLine[EnemyNoHaveUnits[0]].LineHp =
EnemyHandCard[temp].HP;
                            EnemyLine[EnemyNoHaveUnits[0]].busy =
false;
mana -= EnemyHandCard[temp].MANA;
    FindPlace(EnemyNoHaveUnits[0], temp);
                AmountEnemyCard--;
                EnemyMem--;
                        }
                    }

                }
                else if (mem >= 1)
                {
                    for (int i = 0; i < mem; i++)
                    {
                        if (EnemyMem > 0)
                        {
                            temp = FoundWeakness();
                            if (temp == -1) { }
                            else
                            {
```

Продолжение листинга 3.7

```
 EnemyLine[HaveUnit[i]].LineDmg = EnemyHandCard[temp].DMG;
                                    EnemyLine[HaveUnit[i]].LineHp =
EnemyHandCard[temp].HP;
                                    EnemyLine[HaveUnit[i]].busy =
false;
                                    mana -= EnemyHandCard[temp].MANA;
                                    FindPlace(HaveUnit[i], temp);
                                    AmountEnemyCard--;
                                    EnemyMem--;
                                }
                            }
                        }
                        if (EnemyMem > 0)
                        {
                            temp = FoundPower();
                            if (temp == -1) { }
                            else
                            {
                                EnemyLine[EnemyNoHaveUnits[0]].LineDmg
= EnemyHandCard[temp].DMG;
                                EnemyLine[EnemyNoHaveUnits[0]].LineHp =
EnemyHandCard[temp].HP;
                                EnemyLine[EnemyNoHaveUnits[0]].busy =
false;
                                mana -= EnemyHandCard[temp].MANA;
                                AmountEnemyCard--;
                                FindPlace(EnemyNoHaveUnits[0], temp);
                            }
                        }
                    }
                }}
```

Листинг 3.8 – Функция FoundPower()

```
int FoundPower()
        {
            int powerful = -1, memory = -1;
            for (int i = 0; i < 8; i++)
            {
                if (EnemyHandCard[i].MANA <= mana)
                {
                    if (EnemyHandCard[i].DMG > powerful &&
EnemyHandCard[i].DMG > 0)
                    {
                        powerful = EnemyHandCard[i].DMG;
                        memory = i;
                    }
                }
            }
             return memory;
        }
```

Листинг 3.9 – Функция FoundWeakness()

```
{
            int powerful = 99, memory = -1;
            for (int i = 0; i < 8; i++)
            {
                if (EnemyHandCard[i].MANA <= mana)
                {
                    if (EnemyHandCard[i].DMG < powerful &&
EnemyHandCard[i].DMG > 0)
                    {
                        powerful = EnemyHandCard[i].DMG;
                        memory = i;
                    }
                }
            }
         return memory;
        }
```

3.5 Функция EndTurn_Click

Функцию осуществляет переход на следующий раунд, обновляет и увеличивает максимальное число маны, вызывает основные функции такие как Fight(), EnemyTurn(), GiveCard(), ShowEnemyCard() и MANAupdate(). Функция представлена в листинге 3.10.

Листинг 3.10 – Функция EndTurn_Click()

```
private void EndTurn_Click(object sender, EventArgs e)
        {
            mana = manatem++;
            Fight();
            if (NumCard < 28) GiveCard();
            if (NumCard < 20) GiveCardEnemy();
            EnemyTurn();
            ShowEnemyCard(AmountEnemyCard);
            AmountEnemyCard++;
            mana = manatem;
            MANAupdate();
        }
```

По правилам игроки каждый ход берут по одной карте. Это делает функция GiveCard() (листинг 3.11) и её аналог для противника GiveCardEnemy(). Функция fixPic делает правильное отображение картинок.

Листинг 3.11 – Функция GiveCard()

```
void GiveCard()
        {
            int mem = 0;
            for (int i = 0; i < 8; i++)
            {
                if (HandCard[mem].Using == false)
                {
                    break;
                }
                mem++;
            }
            if (mem < 8)
            {
                HandCard[mem] = Card[deck[NumCard]];
                fixPic(mem);
                NumCard++;
            }
            mem = 0;
        }
```

Для того чтобы правильно отображать карты противника используется функция ShowEnemyCard(). Она регулирует количество карт в руке у противника и представлена в листинге 3.11.

Листинг 3.11 – Функция ShowEnemyCard()

```
  void ShowEnemyCard(int num)
        {
            switch (num)
            {
                case 0:
                    Enemycard1.Visible = true;
                    break;
                case 1:
                    Enemycard2.Visible = true;
                    break;
                case 2:
                    Enemycard3.Visible = true;
                    break;
                case 3:
                    Enemycard4.Visible = true;
                    break;
                case 4:
                    Enemycard5.Visible = true;
                    break;
                case 5:
                    Enemycard6.Visible = true;
                    break;
                case 6:
                    Enemycard7.Visible = true;
                    break;
```

```
                    case 7:
                    Enemycard8.Visible = true;
                    break;
                default:
                    break;
            }
        }
```

3.6 Функция Fight

Данная функция представлена на листинге 3.12, она производит бой между выложенными картами на столе, а также за завершение игры в случае поражения одного из игроков. При завершении игры запускается форма, которая говорит о результате игры и предлагает выйти в главное меню.

Листинг 3.12 – Функция Fight()

```
void Fight()
        {
            for (int i = 0; i < 5; i++)
            {
                if (!Line[i].busy)
                {
                    if (!EnemyLine[i].busy){
                        EnemyLine[i].LineHp -= Line[i].LineDmg;
                        Line[i].LineHp -= EnemyLine[i].LineDmg;
                        updat(i);
                    }
                    else
                    {
                        HeroEnemyHp -= Line[i].LineDmg;
                        if (HeroEnemyHp <= 0)
                        {
                            EnemyHp.Location = new Point(1623,
115);
                            EnemyHp.Text = "Dead";
                            EndGame = 1;
                            PauseMenu pause = new PauseMenu(this,
EndGame);
                            pause.ShowDialog();
                            break;
                        }
                        else
                        {
                            if (HeroEnemyHp >= 10)
                            {
```

Продолжение листинга 3.12

```
                        EnemyHp.Location = new Point (1685, 118);
                        EnemyHp.Text = HeroEnemyHp.ToString();
                            }
                            else if (HeroEnemyHp < 10)
                            {
                                EnemyHp.Location = new Point(1701,
115);
                                EnemyHp.Text =
HeroEnemyHp.ToString();
                            }
                        }
                    }
                    else if (!EnemyLine[i].busy)
                    {
                        HeroPlayerHP -= EnemyLine[i].LineDmg;
                        if (HeroPlayerHP <= 0)
                        {
                            Playerhp.Location = new Point(1623, 858);
                            Playerhp.Text = "Dead";
                            EndGame = 2;
                            PauseMenu pause = new PauseMenu(this,
EndGame);
                            pause.ShowDialog();
                            break;
                        }
                        else
                        { if (HeroPlayerHP >= 10)
                            {
                                Playerhp.Location = new Point(1685,
859);
                                Playerhp.Text =
HeroPlayerHP.ToString();
                            }
                            else if (HeroEnemyHp < 10)
                            {
                                Playerhp.Location = new Point(1698,
858);
                                Playerhp.Text =
HeroPlayerHP.ToString();
                            }
                        }
                    }
                }
            }
```

Объединяя все модули, получается программа, в которой есть все необходимые игровые элементы и которая выполняет поставленные задачи проекта.

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 [5].

Тестирование программы проводится с помощью метода черного ящика.

Тестирование методом черного ящика – специальный метод проверки работоспособности программного обеспечения, при котором вся функциональность продукта исследуется без анализа исходного кода. Тестировщики создают логически понятные тест-кейсы, опираясь исключительно на требования из спецификации на проекте [6].

Благодаря этому методу можно быстро находить баги в разработанной функциональности ПО. А также, это удобно, потому что тестировщику не обязательно нужно обладать узкопрофильной специальностью.

При запуске программы появляется меню, в котором пользователь может начать игру или посмотреть коллекцию карт (рисунок 4.1).

Рисунок 4.1 – Меню игры

При нажатии кнопки играть, запуститься игра. Пользователь видит игровой стол (рисунок 4.2).

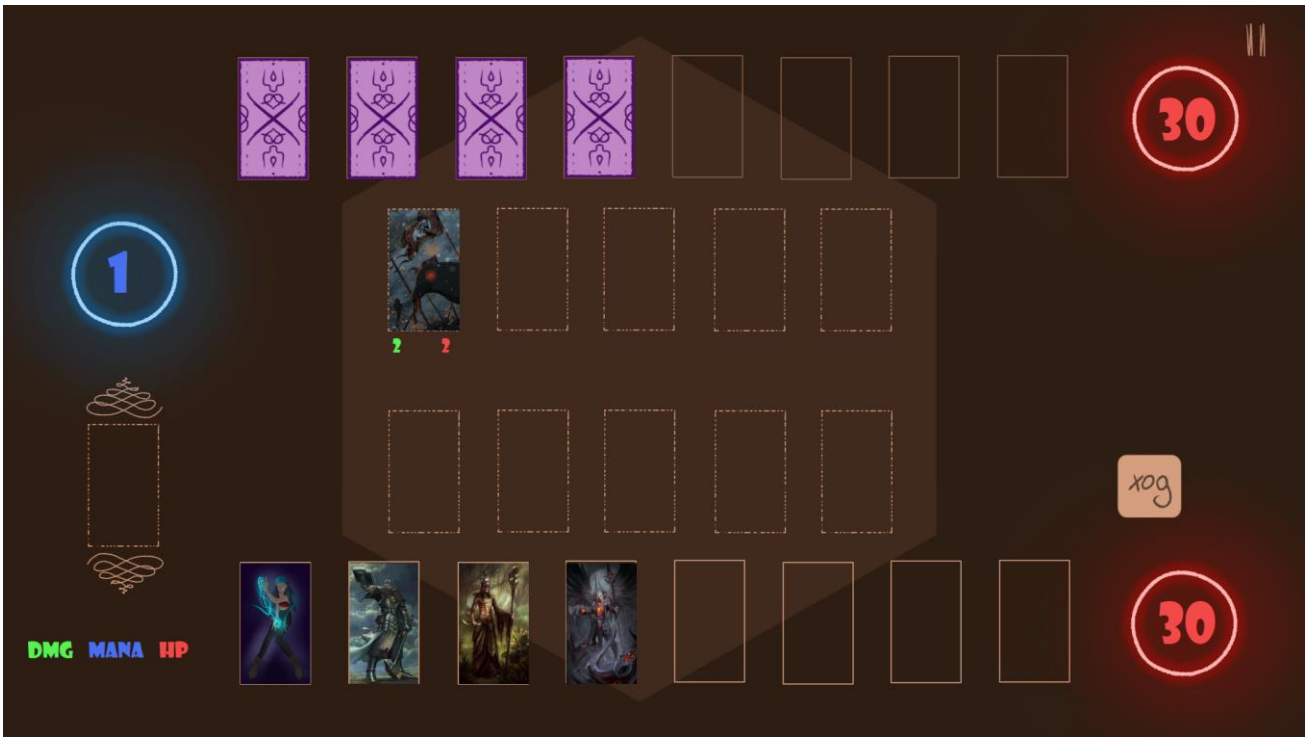

Рисунок 4.2 – Игровой стол

Противник уже выставил карту и теперь игрок может поставить свою стоимостью одну ману. Выбор карты и её выставление с тратой маны видно на рисунок 4.3 и рисунок 4.4.

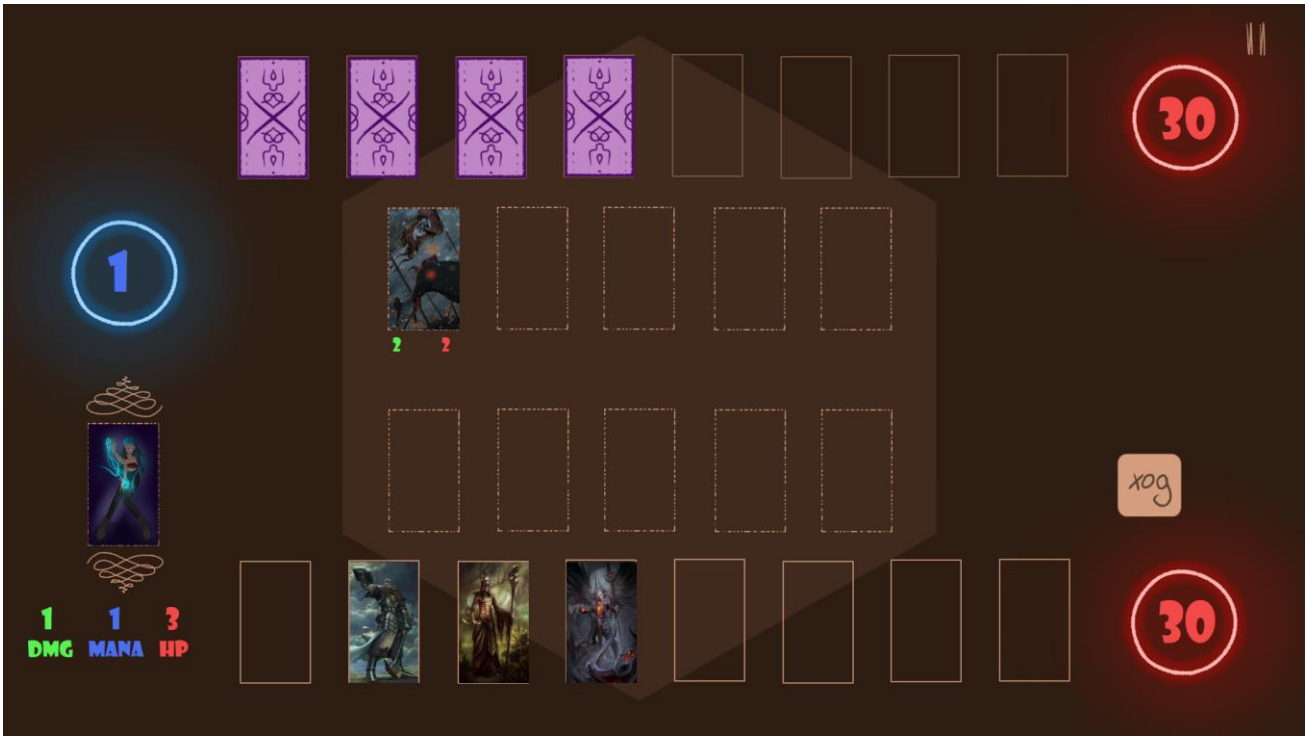

Рисунок 4.3 – Выбора карты

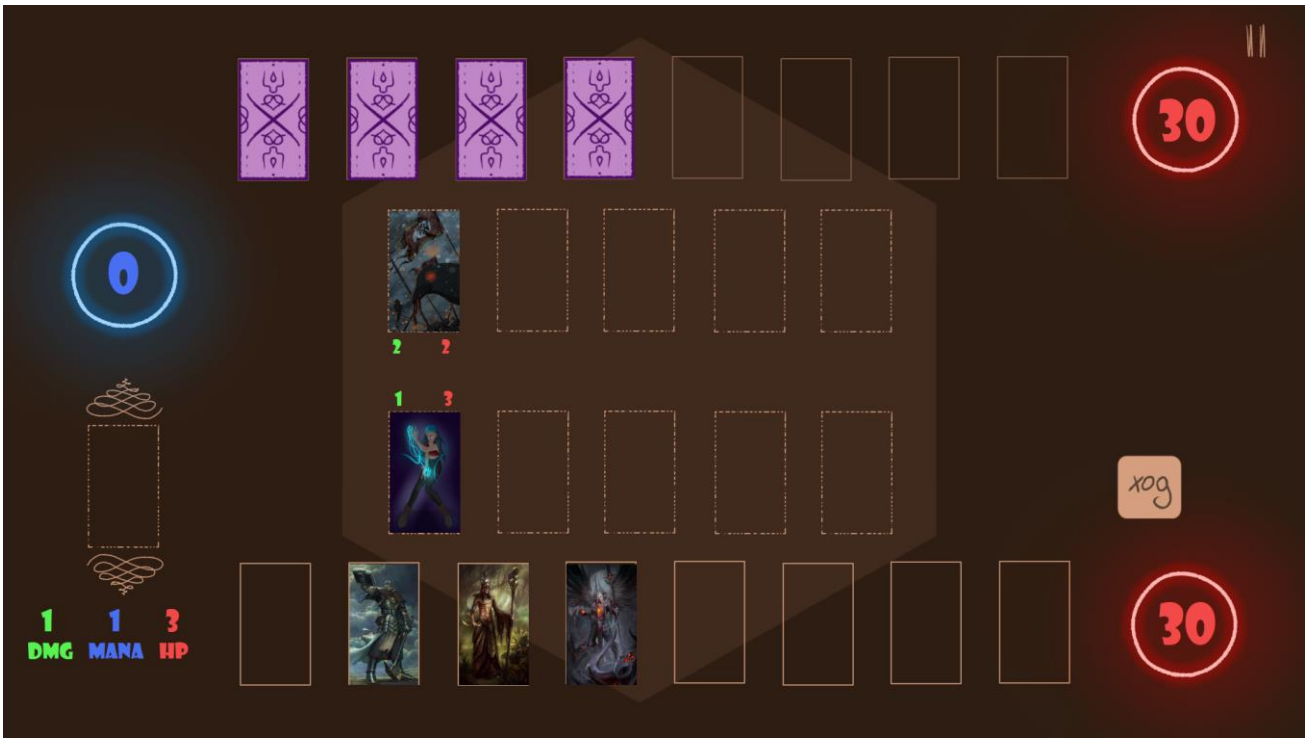

Рисунок 4.4 – Использование карты

После нажатия следующего хода, произошло сражения и карты потеряли значения жизни в соответствии с уроном противника, а также восстановлена мана, получена карта (рисунок 4.5).

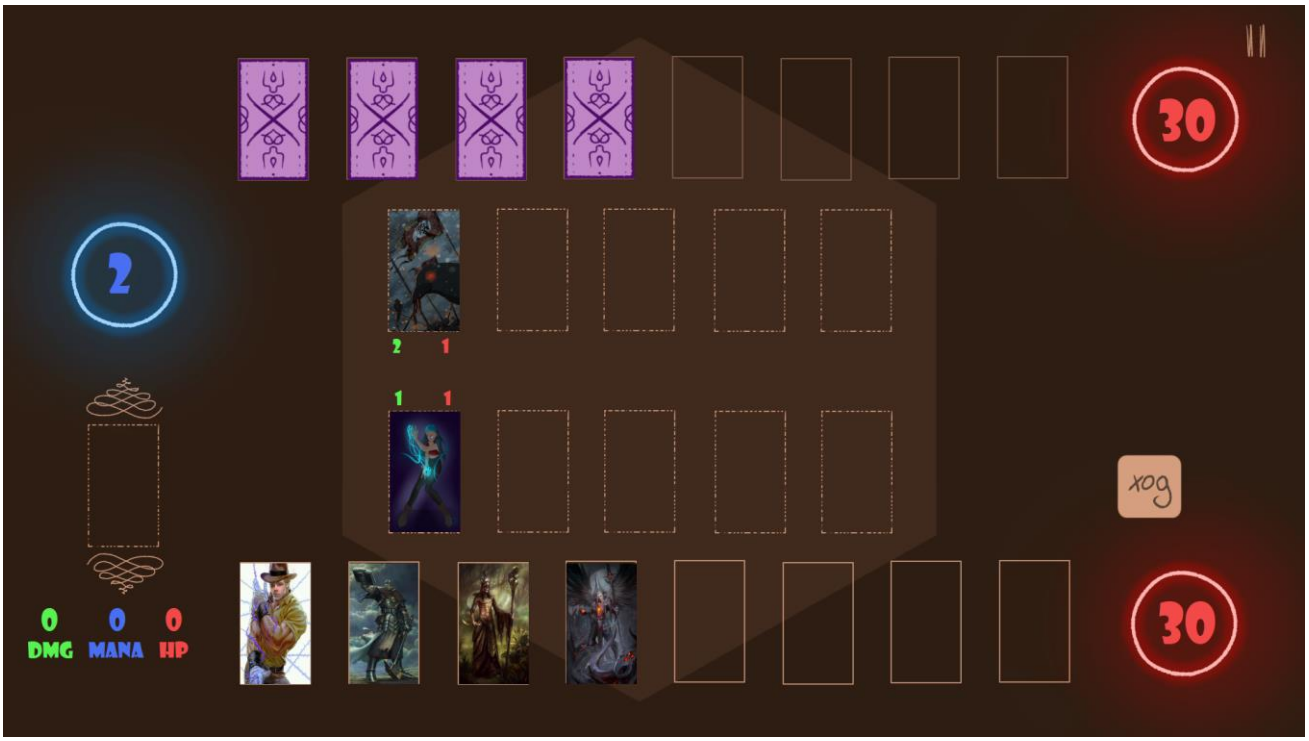

Рисунок 4.5 – Следующий ход

Было произведено несколько раундов в ходе которых произошли сражения, в которых было убито несколько противников и произведены ранения самих игроков (рисунок 4.6).

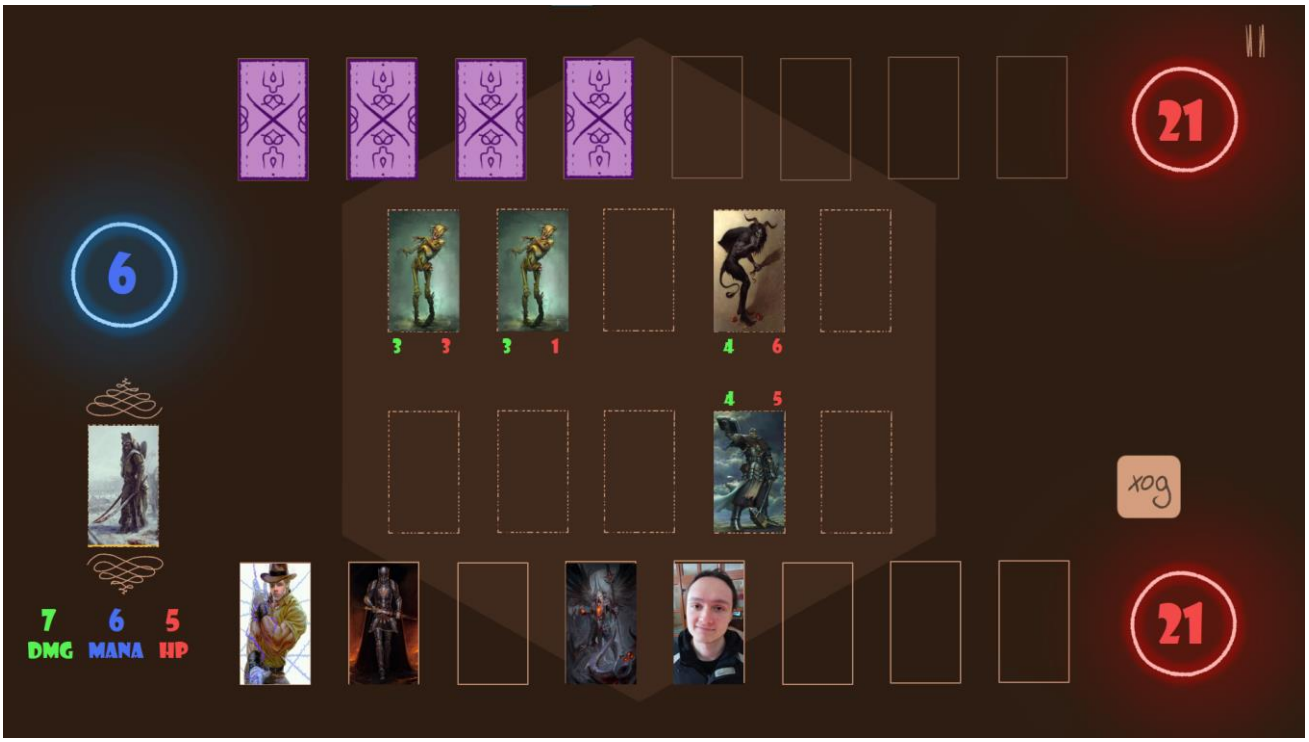

Рисунок 4.6 – Дальнейшие сражения

При нажатии на значок паузы, появляется меню паузы (рисунок 4.7).

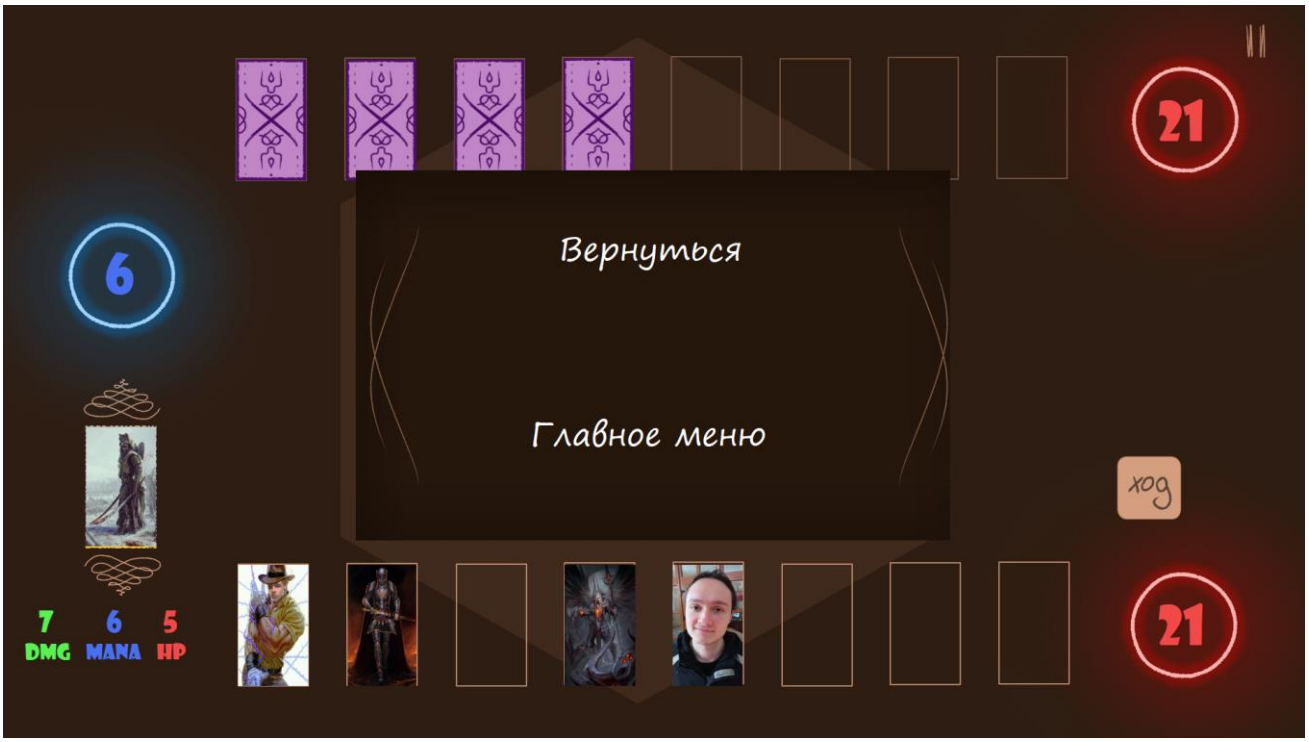

Рисунок 4.7 – Меню паузы

Как только у одного из игроков заканчиваться жизни игра заканчивается и появляется меню окончания игры (рисунок 4.8).

Рисунок 4.8 – Меню окончания игры

При нажатии на меню коллекции откроется коллекция игровых карт, которую можно увидеть на рисунке 4.9.

Рисунок 4.9 – Меню коллекции

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

В ходе тестирования программа вела себя исправно не было аварийного завершения работы ПО, что говорит о полной функциональности игрового приложения.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была поставлена задача – создать полностью работоспособное программное обеспечение, представляющее собой игровое приложение в жанре «ККИ» (коллекционная карточная игра).

Проект сделан и играется с установленными правилами. В игре присутствует мотивация переигрывать, так как после каждой победы у игрока добавляется в колоду новая карта. Полученные карты сохраняться при перезаходе в игру. Реализовано рабочее меню с настройками, где можно изменить рубашку карты, выключить и включить музыку, сбросить прогресс. Присутствует коллекция, где отображаться все игровые карты. В меню паузы можно остановить бой, перейти в главное меню, остановить музыку. Из основных механик в игре реализовано:

- коллекционирование карт;
- интеллект противника;
- возможность выкладывания карт на стол;
- процесс сражения;
- возможность поставить игру на паузу.

В ходе тестирования, не было обнаружено ошибок, программа вела себя исправно не было аварийного завершения работы ПО, что говорит о полной функциональности игрового приложения и выполнении главной задачи курсовой работы.

Программа реализована в соответствии всем требованиям, была протестирована, работает стабильно и может использоваться для решения поставленной задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Microsoft: руководство по классическим приложениям Windows Forms .NET [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0. Дата доступа: 28.11.2022;

2. Wikipedia: коллекционная карточная игра [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Коллекционная_карточная_игра. Дата доступа: 01.12.2022;

3. Wikipedia: проектирование [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Проектирование. Дата доступа: 01.12.2022;

4. Липпман Б. Язык программирования C++. Базовый курс/ Б. Липпман, 2001 – 1104с.

5. Wikipedia: тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения. Дата доступа: 02.12.2022;

6. Testmatick: тестирование методом черного ящика [Электронный ресурс]. – Режим доступа: https://testmatick.com/ru/testirovanie-metodom-serogo-yashhika-osnovnye-ponyatiya-i-osobennosti/. Дата доступа: 02.12.2022.

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 программы	Вход в приложение	Открытие меню приложение	Тест пройден успешно
Выход из программы	Нажатие на кнопку «Выход»	Выход из программы	Тест пройден успешно
Заход в меню коллекции	Нажатие на кнопку «Коллекция»	Открытие меню коллекции	Тест пройден успешно
Заход в меню настройки	Нажатие на кнопку «Настройки»	Открытие меню настройки	Тест пройден успешно
Запуски игры	Нажатие на кнопку «Играть»	Запуск игры	Тест пройден успешно
Остановка игры	Нажатие на кнопку «Пауза»	Остановка игры, появления меню паузы	Тест пройден успешно
Возвращение в игру из меню паузы	Нажатие на кнопку «Вернуться»	Продолжение игры	Тест пройден успешно
Выход в главное меню	Нажатие на кнопку «Главное меню»	Выход в главное меню	Тест пройден успешно
Выставить карту	Нажать на необходимую карту, нажать на линию куда выставить карту	Карта поставлена на нужную линию	Тест пройден успешно
Сражение карт, с последующим убиранием с поля	Карты на столе, нажать на кнопку «Следующий раунд»	Карты правильно посчитали значения, и исчезли все карты со значениями жизни равным нулю	Тест пройден успешно
Ход противника	Нажатие на кнопку «Следующий ход»	Противник выставит карты, в выгодном для него положении	Тест пройден успешно

Продолжение таблицы А.1

Тестовый вариант	Входные данные	Ожидаемый результат	Результат тестирования
Начала следующего хода	Нажатие на кнопку «Следующий ход»	Обновление значений маны	Тест пройден успешно
Добавление карты в руку	Нажатие на кнопку «Следующий ход»	Карта появиться в руке	Тест пройден успешно
Окончание игры	Здоровья игрока или противника равное нулю	Запуск меню окончания игры	Тест пройден успешно
Выход в главное меню через меню окончания игры	Нажатие на кнопку «Главное меню»	Выход в главное меню	Тест пройден успешно