УДК 004.032.26

Разработка приложения - пазла в Visual Studio

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассматривается процесс разработки приложения-пазла в среде Visual Studio. Описывается использование различных элементов управления, таких как кнопки и изображения, а также методы работы с графикой и анимацией. Целью данной статьи является, применение Windows Presentation Foundation (WPF) и языка программирования C#, для создания приложения-пазла. Практическим результатом является разработанное приложение.
Ключевые слова: C#, Visual Studio, приложение-пазл, игра

Developing a puzzle application in Visual Studio

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article discusses the process of developing a puzzle application in the Visual Studio environment. It describes the use of various controls, such as buttons and images, as well as methods of working with graphics and animation. The purpose of this article is to use the Windows Presentation Foundation (WPF) and the C# programming language to create a puzzle application. The practical result is a developed application.
Keywords: C#, Visual Studio, puzzle application, game

1. Введение

1.1 Актуальность исследования

Разработка игровых приложений стала важной темой в современном мире. Игры уже давно перестали ассоциироваться только с развлечением и зрелищем, они также могут быть использованы для обучения и развития умственных способностей, а также как способ расслабления и улучшения психического здоровья. Поэтому все больше людей интересуются созданием собственных игр и приложений.

Среда Visual Studio является одним из наиболее популярных инструментов для разработки приложений на языке C#. С помощью данного языка программирования можно создавать игры различной сложности и

функциональности. Игры-головоломки, такие как пазлы, также остаются популярными среди пользователей.

1.2 Обзор исследований

В своей работе М.А. Скороходов рассмотрел применение WPF для разработки викторины по программированию [1]. В статье А.М. Снопкова, В.В. Данилова, В.А. Шабзон описана работа универсальной программы - викторины, которая хорошо справляется с диагностикой знаний обучающихся [2]. В своей работе А. В. Курдидис описал процесс разработки мобильного приложения - викторины по биотехнологиям [3]. А.А. Журавлев в статье описал создание приложение для людей, занимающихся контролем продукции в магазине [4]. А.В. Бартновская показала принцип планирования и разработки приложений в WPF на примере игры [5].

1.3 Цель исследования

Цель исследования – применяя возможности среды разработки Visual Studio и языка программирования C#, создать простое приложение пазла, со всеми, необходимыми для полноценной игры, механиками.

2. Материалы и методы

Для создания игры будем использовать программное обеспечение Visual Studio, а также язык программирования C#.

Для проекта использовалась Windows Presentation Foundation (WPF), платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Эта платформа является частью платформы .NET.

3 Результаты и дискуссия

Начнём разработку приложения, с создания WPF приложения. Сразу назовем проект «Picture Puzzle» (рис. 1).

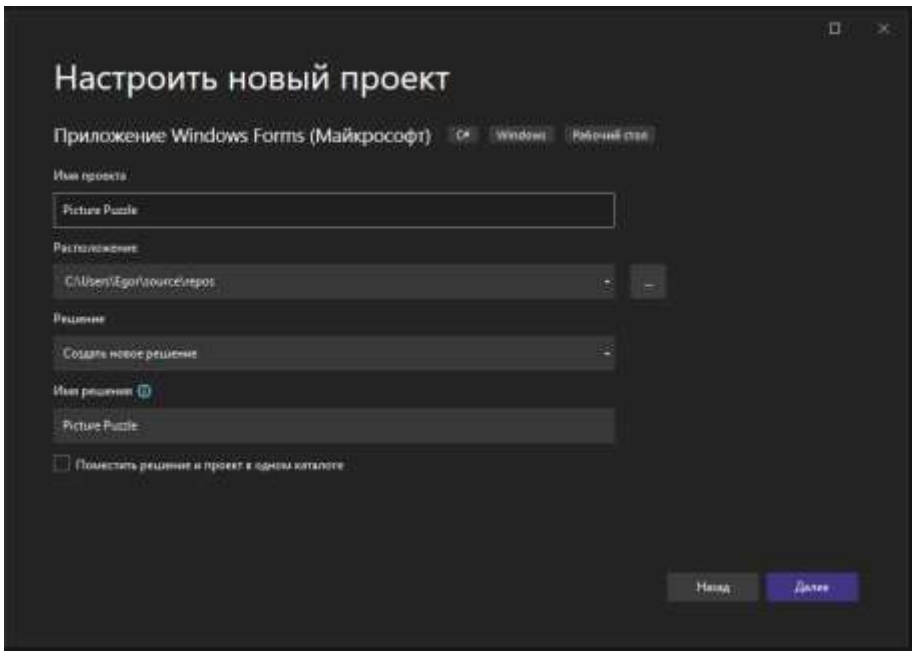

Рисунок 1. Создание проекта

Для начала добавим на форму необходимые элементы. Основным полотном, на котором будут располагаться части пазла, является встроенный элемент - «panel», в качестве частей панели - «button» (рис.2).

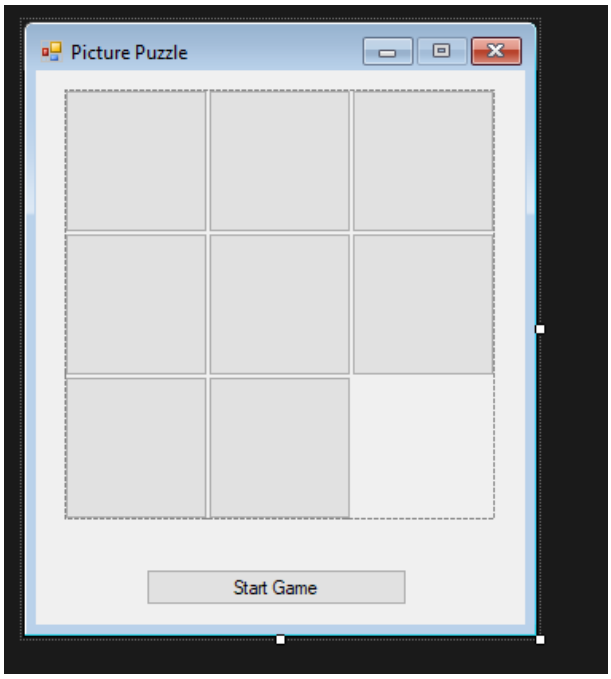

Рисунок 2. Расположение элементов

Переходим к написанию логики игры. Подключаем необходимые библиотеки. Создаем публичный класс (для доступа из других частей приложения), а внутри определяем поля:

- «EmptyPoint» - объект типа «Point», который используется для хранения позиции пустой ячейки в пазле.

- «images» - объект «ArrayList», который используется для хранения изображений в пазле.

Также сразу определим игровое поле (рис.3).

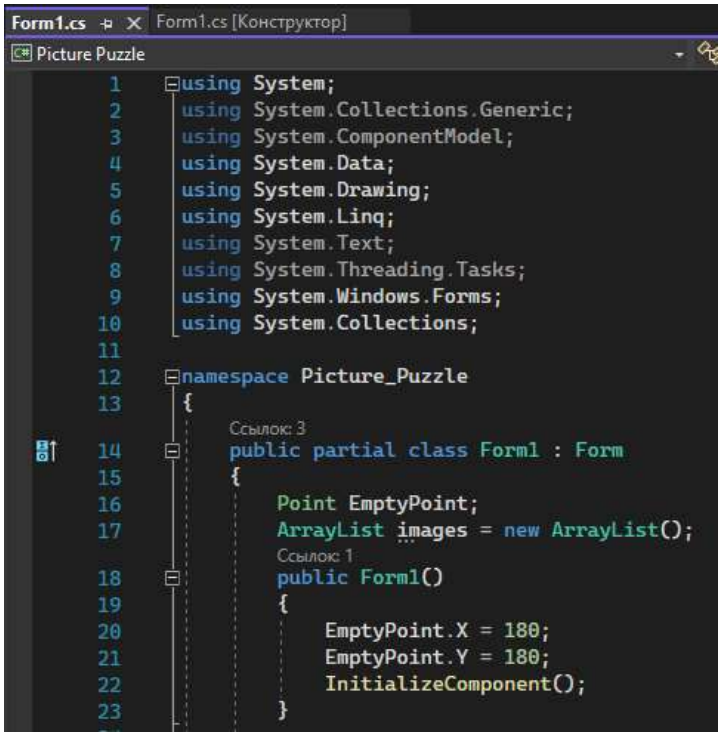

Рисунок 3. Добавление необходимых переменных и определение размера поля

Создаем событие (метод) на щелчок мыши по кнопке «Start Game», начинающий генерацию пазла. Этот метод вызывает другие функции для создания и перемешивания изображений, затем добавляет изображения в кнопки. Метод «private void AddImagesToButtons(ArrayList images)» добавляет изображения в кнопки пазла. Каждая кнопка на форме представляет часть изображения, которая составляет пазл. Эта функция берет 8 изображений, перемешивает и распределяет по кнопкам в случайном порядке. Далее метод «private int[] suffle(int[] arr)» перемешивает индексы массива с помощью генератора случайных чисел и метода «OrderBy» класса «Enumerable». Объекты, используемые в этом коде:

- Image - класс для работы с изображениями
- Button - класс для создания кнопок на форме.
- ArrayList - коллекция, используемая для хранения изображений в пазле (рис.4).

```
Ссылок: 1
private void button9_Click(object sender, EventArgs e)
{
    foreach (Button b in panel1.Controls)
        b.Enabled = true;

    Image orginal = Image.FromFile(@"img\img.jpg");

    cropImageTomages(orginal, 270, 270);

    AddImagesToButtons(images);
}

Ссылок: 1
private void AddImagesToButtons(ArrayList images)
{
    int i = 0;
    int[] arr = { 0, 1, 2, 3, 4, 5, 6, 7 };

    arr = suffle(arr);

    foreach (Button b in panel1.Controls)
    {
        if (i < arr.Length)
        {
            b.Image = (Image)images[arr[i]];
            i++;
        }
    }
}

Ссылок: 1
private int[] suffle(int[] arr)
{
    Random rand = new Random();
    arr = arr.OrderBy(x => rand.Next()).ToArray();
    return arr;
}
```

Рисунок 4. Добавление метода обработки щелчка мыши и генерации частей пазла

Далее создаем метод «private void cropImageTomages(Image orginal, int w, int h)», который будет разбивать изображение на 8 частей, каждая из которых будет представлять собой кусок изображения, выбранный из оригинала с помощью параметров ширины и высоты. Эта функция создает новый битмап размером w*h, затем используя класс «Graphics» отрисовывает оригинальное изображение в битмапе. Затем для каждой части пазла функция создает битмап размером 90*90 и копирует пиксели из битмапа оригинала. Полученные куски изображения добавляются в коллекцию изображений. Теперь создаем функцию «private void button1_Click(object sender, EventArgs e)» отвечающую за обработку события нажатия на конкретную часть (кнопку) в пазле. Этот метод вызывается при нажатии на любую кнопку в пазле. Если выбранная часть пазла рядом с пустой клеткой, они меняются местами. Объекты, используемые в этом коде:

- «Image» - класс для работы с изображениями.
- «Bitmap» - класс для создания битмапов.

- «Graphics» - класс для работы с рисованием на битмапах.
- «Button» - класс для создания кнопок на форме (рис.5).

```
Ссылок: 1
private void cropImageTomages(Image orginal, int w, int h)
{
    Bitmap bmp = new Bitmap(w, h);

    Graphics graphic = Graphics.FromImage(bmp);

    graphic.DrawImage(orginal, 0, 0, w, h);

    graphic.Dispose();

    int movr = 0, movd = 0;

    for (int x = 0; x < 8; x++)
    {
        Bitmap piece = new Bitmap(90, 90);

        for (int i = 0; i < 90; i++)
            for (int j = 0; j < 90; j++)
                piece.SetPixel(i, j,
                    bmp.GetPixel(i + movr, j + movd));

        images.Add(piece);

        movr += 90;

        if (movr == 270)
        {
            movr = 0;
            movd += 90;
        }
    }

}


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

Рисунок 5. Добавление метода разбивки изображения

Переходим к реализации функции перемещения частей пазла на поле, для этого создаем метод «private void MoveButton(Button btn)». Эта функция принимает объект части пазла в качестве аргумента. Если расстояние между частью пазла и пустой клеткой составляет 90 пикселей по горизонтали или вертикали, то кнопка перемещается на место пустой кнопки. Координаты перемещаемой кнопки и пустой кнопки затем меняются местами. Если пустая

клетка перемещается на последнее место (координаты 180, 180), запускается функция «CheckValid()». Данная функция проверяет все ли части в пазле находятся на своих местах. Эта функция подсчитывает количество кнопок, которые находятся на своих местах. Все кнопки в пазле подразделяются на 8 частей, и каждая часть имеет свой уникальный индекс, который определяется через формулу (y/90)*3 + x/90, где y и x - это координаты кнопки в пикселях. Если все 8 кнопок находятся на своих местах, то запускается сообщение о победе (рис.6).

```
Ссылок: 1
private void MoveButton(Button btn)
{
    if (((btn.Location.X == EmptyPoint.X - 90 || btn.Location.X == EmptyPoint.X + 90)
        && btn.Location.Y == EmptyPoint.Y)
        || (btn.Location.Y == EmptyPoint.Y - 90 || btn.Location.Y == EmptyPoint.Y + 90)
        && btn.Location.X == EmptyPoint.X)
    {
        Point swap = btn.Location;
        btn.Location = EmptyPoint;
        EmptyPoint = swap;
    }

    if (EmptyPoint.X == 180 && EmptyPoint.Y == 180)
        CheckValid();
}


private void CheckValid()
{
    int count = 0, index;
    foreach (Button btn in panel1.Controls)
    {
        index = (btn.Location.Y / 90) * 3 + btn.Location.X / 90;
        if (images[index] == btn.Image)
            count++;
    }
    if (count == 8)
        MessageBox.Show("well done you win!");
}
```

Рисунок 6. Добавление метода движения частей пазла и проверка собранного пазла

Осталось только проверить игру. Жмем кнопку «Start Game» (рис.7-9).

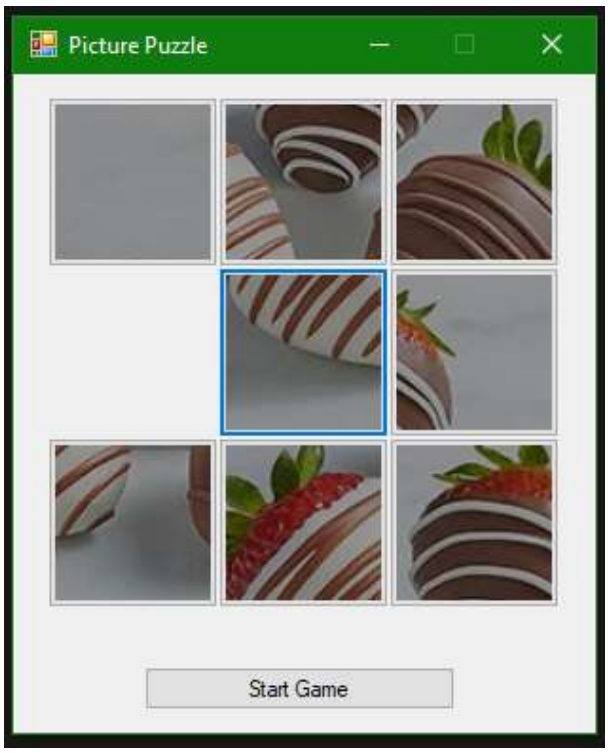

Рисунок 7. Демонстрация пазла

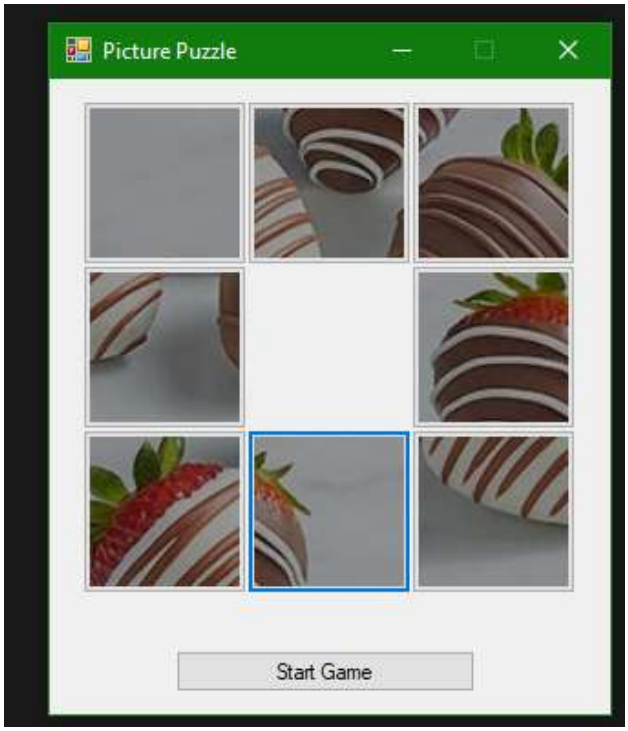

Рисунок 8. Демонстрация пазла

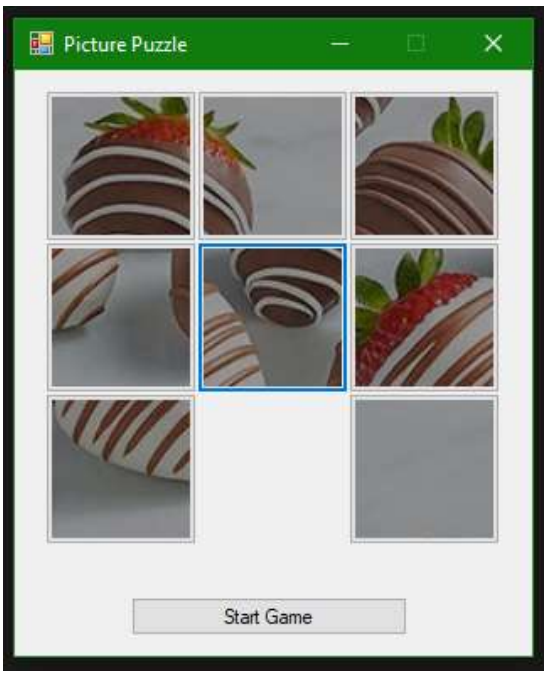

Рисунок 9. Демонстрация пазла

4 Выводы

В рамках данного исследования описана разработка приложения-пазла в среде разработки Visual Studio. В дальнейшем планируется расширения функционала приложения, улучшение интерфейса и основных механик, а также добавление соревновательного мультиплеера.

Библиографический список

1. Скороходов М.А. Разработка викторины по программированию с использованием WPF// Форум молодых ученых. 2021. С. 145-149.
2. Снопкова А.М., Данилова В.В., Шабзон В.А. Разработка универсальной программы - викторины на языке программирования python// Междисциплинарность науки как фактор инновационного развития. 2018. №4. С. 104-107.
3. Курдидис А. В. Разработка мобильного приложения-викторины по биотехнологиям// Юный ученый. 2023. № 4 (67-1). С. 25-26.
4. Журавлев А.А. Разработка приложения для контроля продукции магазина с помощью WPF// Modern science. 2021. С. 527-530.
5. Бартновская А.В. Приложение, реализующее игру "линейные гонки" с использованием WPF и графики Directx// Актуальные вопросы физики и техники. 2020. С. 256-259.