УДК 004

Создание игры Шашки в Microsoft Visual Studio на языке программирования C#

Маринчук Александр Сергеевич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания игры Шашки в Microsoft Visual Studio на языке программирования C#. Рассмотрены особенности реализации логики передвижения шашек, заданы правила и описаны условия победы. В проекте используются winforms для отрисовки шахматной доски, а внешний вид шашек подгружается из изображений.
Ключевые слова: игра, шашки, MVS, C#, программирование, winforms.

Creating a Checkers game in Microsoft Visual Studio using the C # programming language

Marinchuk Alexander Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of creating a Checkers game in Microsoft Visual Studio in the C#programming language. The features of the implementation of the logic of the movement of checkers are considered, the rules are set and the conditions of victory are described. The project uses winforms to draw the chessboard, and the view of the checkers is loaded from the images.
Keywords: game, checkers, MVS, C #, programming, winforms.

1. Введение

1.1 Актуальность исследования

Шашки являются довольно старой игрой, которая развивает логическое мышление и раскрывает стратегический склад ума. В данную игру играют миллионы людей на планете ежедневно, но мало кто задумывается о ее программной реализации на том или ином языке программирования. Сами по себе шашки в виде компьютерной игры реализованы достаточно давно, но начинающим программистам стоит обратить на них внимание, так как при написании данной игры рассматриваются различные методы и логика осуществления ходов, которые помогут отточить навыки программирования и применить их в будущем.

1.2 Обзор исследований

В научной работе Э. Э. Сейдаметова и А. Э. Шабанова проведен сравнительный анализа игровых движков, описаны основные критерии на которые стоит обращать внимание при выборе игрового движка, как основного инструмента для разработки игр [1]. И. Д. Ноек и Демичев В. А. свою статью посвящают разработке компьютерной игры на языке C# с помощью методов объектно-ориентированного программирования и кроссплатформенной библиотеки Monogame [2]. R. Bourbia и др. в своей научной работе рассказывают об игре, цель которой улучшение навыков сборки компьютера студентами в виртуальной лаборатории [3]. Разработку игры «Ships» на языке с Sharp описывают Д. В. Борисов и А. В. Виноградов, так же они рассматривают преимущества движка Unity [4]. В работе S. Tesanovic и P. Mitrovic рассматривается реализация игры «Палач» в Microsoft Visual Studio с использованием языка программирования ассемблера, библиотек Kip Irvine's и MASM [5]. А. С. Маринчук в своем исследовании описывает создание игры Flappy Bird в среде Microsoft Visual Studio на языке программирования C# [6]. В статье Д. Ю. Тазабекова и Н.В. Бужинской рассматриваются особенности разработки обучающей компьютерной игры в среде Unity. Данная игра предназначена для обучения школьников информатике в занимательной форме [7].

1.3 Цель исследования

Целью данной статьи является описание процесса создания игры Шашки в Microsoft Visual Studio на языке программирования C#.

2. Методы исследования

Шашки - логическая настольная игра для двух игроков, заключающаяся в передвижении определённым образом фишек-шашек по клеткам шашечной доски. Во время партии каждому игроку принадлежат шашки одного цвета: чёрного или белого. Цель игры — взять все шашки соперника или лишить их возможности хода (запереть) [8].

Для начала следует создать проект и добавить форму. Далее перейти к ее коду и подключить необходимые библиотеки и объявить переменные (Рис. 1).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CheckersGame
{
    Ссылок: 3
    public partial class Form1 : Form
    {
        const int mapSize = 8;
        const int cellSize = 50;

        int currentPlayer;

        List<Button> simpleSteps = new List<Button>();

        int countEatSteps = 0;
        Button prevButton;
        Button pressedButton;
        bool isContinue = false;

        bool isMoving;

        int[,] map = new int[mapSize, mapSize];

        Button[,] buttons = new Button[mapSize, mapSize];

        Image whiteFigure;
        Image blackFigure;
```

Рисунок 1 – Объявление переменных и подключение библиотек

Далее следует написать функцию, создающую игровое поле (Рис. 2).

```
public void CreateMap()
{
    this.Width = (mapSize + 1) * cellSize;
    this.Height = (mapSize + 1) * cellSize;

    for(int i = 0; i < mapSize; i++)
    {
        for (int j = 0; j < mapSize; j++)
        {
            Button button = new Button();
            button.Location = new Point(j * cellSize, i * cellSize);
            button.Size = new Size(cellSize, cellSize);
            button.Click += new EventHandler(OnFigurePress);
            if (map[i, j] == 1)
                button.Image = whiteFigure;
            else if (map[i, j] == 2) button.Image = blackFigure;

            button.BackColor = GetPrevButtonColor(button);
            button.ForeColor = Color.Red;

            buttons[i, j] = button;

            this.Controls.Add(button);
        }
    }
}
```

Рисунок 2 – Функция CreateMap

Теперь напишем функцию смены игрока после завершения хода (Рис. 3).

```
ссылка: 1
public void SwitchPlayer()
{
    currentPlayer = currentPlayer == 1 ? 2 : 1;
    ResetGame();
}
```

Рисунок 3 – Функция смены игрока

Следующим шагом будет написание функции, отвечающей за выбор той или иной шашки при котором остальные шашки начинают светиться серым (Рис. 4).

```
Ссылок: 5
public Color GetPrevButtonColor(Button prevButton)
{
    if ((prevButton.Location.Y/cellSize % 2) != 0)
    {
        if ((prevButton.Location.X / cellSize % 2) == 0)
        {
            return Color.Gray;
        }
    }
    if ((prevButton.Location.Y / cellSize) % 2 == 0)
    {
        if ((prevButton.Location.X / cellSize) % 2 != 0)
        {
            return Color.Gray;
        }
    }
    return Color.White;
}
```

Рисунок 4 – Функция GetPrevButtonColor

Далее напишем функцию OnFigurePress, которая будет отвечать непосредственно за логику передвижения шашек по игровому полю (Рис. 6).

```
public void OnFigurePress(object sender, EventArgs e)
{
    if (prevButton != null)
        prevButton.BackColor = GetPrevButtonColor(prevButton);

    pressedButton = sender as Button;

    if(map[pressedButton.Location.Y/cellSize,pressedButton.Location.X/cellSize] != 0 && map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] == currentPlayer)
    {
        CloseSteps();
        pressedButton.BackColor = Color.Red;
        DeactivateAllButtons();
        pressedButton.Enabled = true;
        countEatSteps = 0;
        if(pressedButton.Text == "D")
        ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize,false);
        else ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize);

        if (isMoving)
        {
            CloseSteps();
            pressedButton.BackColor = GetPrevButtonColor(pressedButton);
            ShowPossibleSteps();
            isMoving = false;
        }
        else
            isMoving = true;
    }
    else
    {
        if (isMoving)
        {
            isContinue = false;
              if (Math.Abs(pressedButton.Location.X / cellSize - prevButton.Location.X/cellSize) > 1)
            {
                isContinue = true;
                DeleteEaten(pressedButton, prevButton);
            }
            int temp = map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize];
            map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] = map[prevButton.Location.Y / cellSize, prevButton.Location.X / cellSize];
            map[prevButton.Location.Y / cellSize, prevButton.Location.X / cellSize] = temp;
            pressedButton.Image = prevButton.Image;
            prevButton.Image = null;
            pressedButton.Text = prevButton.Text;
            prevButton.Text = "";
            SwitchButtonToCheat(pressedButton);
            countEatSteps = 0;
            isMoving = false;
            CloseSteps();
            DeactivateAllButtons();
            if (pressedButton.Text == "D")
                ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize, false);
            else ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize);
            if (countEatSteps == 0 || !isContinue)
            {
                CloseSteps();
                SwitchPlayer();
                ShowPossibleSteps();
                isContinue = false;
            }else if(isContinue)
            {
                pressedButton.BackColor = Color.Red;
                pressedButton.Enabled = true;
                isMoving = true;
            }
        }
    }

    prevButton = pressedButton;
}
```

Рисунок 6 – Функция OnFigurePress

Теперь опишем функцию показа возможных ходов при выборе шашки (Рис. 7).

```
public void ShowPossibleSteps()
{
    bool isOneStep = true;
    bool isEatStep = false;
    DeactivateAllButtons();
    for(int i = 0; i < mapSize; i++)
    {
        for (int j= 0; j < mapSize; j++)
        {
            if (map[i, j] == currentPlayer)
            {
                if (buttons[i, j].Text == "D")
                    isOneStep = false;
                else isOneStep = true;
                if (IsButtonHasEatStep(i, j, isOneStep, new int[2] { 0, 0 }))
                {
                    isEatStep = true;
                    buttons[i, j].Enabled = true;
                }
            }
        }
    }
    if (!isEatStep)
        ActivateAllButtons();
}
```

Рисунок 7 – Функция ShowPossibleSteps

Если шашка дошла до противоположной стороны, то она превращается в «дамку» и логика ее движения меняется. Опишем данные действия в следующей функции (Рис. 8).

```
public void SwitchButtonToCheat(Button button)
{
    if (map[button.Location.Y / cellSize, button.Location.X / cellSize] == 1 && button.Location.Y / cellSize == mapSize - 1)
    {
        button.Text = "D";

    }
    if (map[button.Location.Y / cellSize, button.Location.X / cellSize] == 2 && button.Location.Y / cellSize == 0)
    {
        button.Text = "D";
    }
}
```

Рисунок 8 – Функция SwitchButtonToCheat

Союзная шашка при «перепрыгивании» через вражескую «съедает» ее и она покидает игровое поле. Опишем эту логику в функции DeleteEaten (Рис. 9).

```
public void ShowDiagonal(int IcurrFigure, int JcurrFigure, bool isOneStep = false)
{
    int j = JcurrFigure + 1;
    for (int i = IcurrFigure - 1; i >= 0; i--)
    {
        if (currentPlayer == 1 && isOneStep && !isContinue) break;
        if (IsInsideBorders(i, j))
        {
            if (!DeterminePath(i, j))
                break;
        }
        if (j < 7)
            j++;
        else break;

        if (isOneStep)
            break;
    }

    j = JcurrFigure - 1;
    for (int i = IcurrFigure - 1; i >= 0; i--)
    {
        if (currentPlayer == 1 && isOneStep && !isContinue) break;
        if (IsInsideBorders(i, j))
        {
            if (!DeterminePath(i, j))
                break;
        }
        if (j > 0)
            j--;
        else break;

        if (isOneStep)
            break;
    }

    j = JcurrFigure - 1;
    for (int i = IcurrFigure + 1; i < 8; i++)
    {
        if (currentPlayer == 2 && isOneStep && !isContinue) break;
        if (IsInsideBorders(i, j))
        {
            if (!DeterminePath(i, j))
                break;
        }
        if (j > 0)
            j--;
        else break;

        if (isOneStep)
            break;
    }

    j = JcurrFigure + 1;
    for (int i = IcurrFigure + 1; i < 8; i++)
    {
        if (currentPlayer == 2 && isOneStep && !isContinue) break;
        if (IsInsideBorders(i, j))
        {
            if (!DeterminePath(i, j))
                break;
        }
        if (j < 7)
            j++;
        else break;

        if (isOneStep)
            break;
    }
}
```

Рисунок 9 – Функция DeleteEaten

Следующая функция описывает логику хода «дамки» по диагонали. Также она может «съедать» вражеские шашки, осуществляя ход под углом 90 градусов (Рис. 10).

```
private void update(object sender, EventArgs e)
{
    MapController.ResetArea();
    if (!MapController.Collide())
    {
        MapController.currentShape.MoveDown();
    }
    else
    {
        MapController.Merge();
        MapController.SliceMap(label1,label2);
        timer1.Interval = MapController.Interval;
        MapController.currentShape.ResetShape(3,0);
        if (MapController.Collide())
        {
            MapController.ClearMap();
            timer1.Tick -= new EventHandler(update);
            timer1.Stop();
            MessageBox.Show("Ваш результат: " + MapController.score);
            Init();
        }
    }
    MapController.Merge();
    Invalidate();
}
```

Рисунок 10 – Функция логики движения «дамки»

Далее описаны две функции, отвечающие за «съедание» союзной шашкой нескольких вражеских шашек. Если таковые ходы имеются, то остальные ходы блокируются (Рис. 11).

```
public bool DeterminePath(int ti,int tj)
{

    if (map[ti, tj] == 0 && !isContinue)
    {
        buttons[ti, tj].BackColor = Color.Yellow;
        buttons[ti, tj].Enabled = true;
        simpleSteps.Add(buttons[ti, tj]);
    }else
    {

        if (map[ti, tj] != currentPlayer)
        {
            if (pressedButton.Text == "D")
                ShowProceduralEat(ti, tj, false);
            else ShowProceduralEat(ti, tj);
        }

        return false;
    }
    return true;
}

Ссылок: 2
public void CloseSimpleSteps(List<Button> simpleSteps)
{
    if (simpleSteps.Count > 0)
    {
        for (int i = 0; i < simpleSteps.Count; i++)
        {
            simpleSteps[i].BackColor = GetPrevButtonColor(simpleSteps[i]);
            simpleSteps[i].Enabled = false;
        }
    }
}
```

Рисунок 11 – Функции DeterminePath и CloseSimpleSteps

Три следующие функции описывают рамки хода шашки, чтобы она не могла выйти за игровое поле, активацию шашек при снятии выбора с фигуры и деактивацию шашек (Рис. 12).

```
public bool IsInsideBorders(int ti,int tj)
{
    if(ti>=mapSize || tj >= mapSize || ti<0 || tj < 0)
    {
        return false;
    }
    return true;
}

ссылка: 1
public void ActivateAllButtons()
{
    for (int i = 0; i < mapSize; i++)
    {
        for (int j = 0; j < mapSize; j++)
        {
            buttons[i, j].Enabled = true;
        }
    }
}

Ссылок: 3
public void DeactivateAllButtons()
{
    for (int i = 0; i < mapSize; i++)
    {
        for (int j = 0; j < mapSize; j++)
        {
            buttons[i, j].Enabled = false;
        }
    }
}
```

Рисунок 12 – Функции рамок поля, активации и деактивации шашек

С помощью функции InitializeComponent придадим фигурам вид шашек, используя заранее приготовленные изображения. В Init проинициализируем проект и запустим его, создавая поле с шашками, расположенными как в массиве map. Также напишем функцию рестарта игры (Рис. 13).

```
public Form1()
{
    InitializeComponent();

    whiteFigure = new Bitmap(new Bitmap(@"C:\Users\admin\Downloads\CheckersGame-master\CheckersGame-master\CheckersGame\Sprites\w.png"), new Size(cellSize - 10, cellSize - 10));
    blackFigure = new Bitmap(new Bitmap(@"C:\Users\admin\Downloads\CheckersGame-master\CheckersGame-master\CheckersGame\Sprites\b.png"), new Size(cellSize - 10, cellSize - 10));

    this.Text = "Шашки";

    Init();
}

Ссылок: 2
public void Init()
{
    currentPlayer = 1;
    isMoving = false;
    prevButton = null;

    map = new int[mapSize,mapSize] {
        { 0,1,0,1,0,1,0,1 },
        { 1,0,1,0,1,0,1,0 },
        { 0,1,0,1,0,1,0,1 },
        { 0,0,0,0,0,0,0,0 },
        { 0,0,0,0,0,0,0,0 },
        { 2,0,2,0,2,0,2,0 },
        { 0,2,0,2,0,2,0,2 },
        { 2,0,2,0,2,0,2,0 }
    };

    CreateMap();
}

ссылка: 1
public void ResetGame()
{
    bool player1 = false;
    bool player2 = false;

    for(int i = 0; i < mapSize; i++)
    {
        for (int j = 0; j < mapSize; j++)
        {
            if (map[i, j] == 1)
                player1 = true;
            if (map[i, j] == 2)
                player2 = true;
        }
    }
    if (!player1 || !player2)
    {
        this.Controls.Clear();
        Init();
    }
}
```

Рисунок 13 – Инициализация проекта и функция рестарта игры

Запустим проект и посмотрим на внешний вид игры (Рис. 14-15).

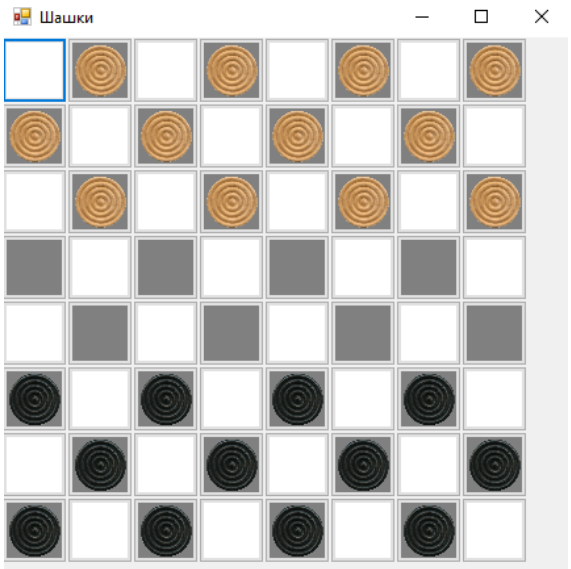

Рисунок 14 – Вид готовой игры при запуске проекта

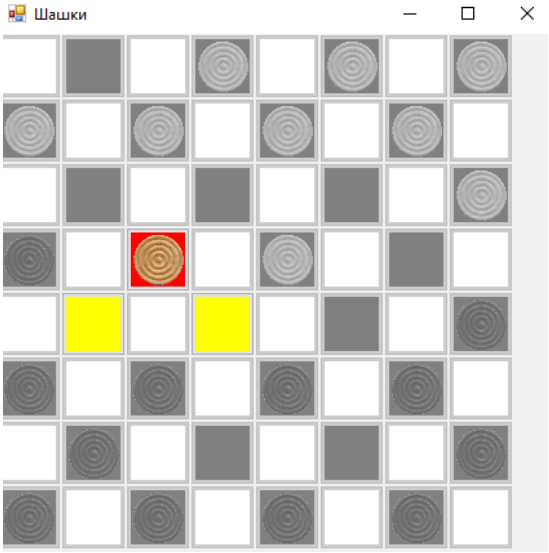

Рисунок 15 – Вид игры при выборе шашке

3. Выводы

Таким образом, была написана игра Шашки в Microsoft Visual Studio на языке программирования C#, рассмотрена логика передвижения шашек, описаны условия «съедания» вражеских фигур и представлены методы построения игрового поля.

Библиографический список

1. Сейдаметов Э. Э., Шабанов А. Э. Современные средства разработки игр //Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2020. №. 1. С. 54-62.
2. Ноек И. Д., Демичев В. А. Разработка компьютерной игры на языке программирования c# с применением основных принципов объектно-ориентированного программирования и свободного программного обеспечения Monogame //Материалы Ежегодной межвузовской студенческой научной конференции ОЧУ ВО «Еврейский университет». 2019. С. 251-259.
3. Bourbia R. et al. Development of serious game to improve computer assembly skills //Procedia-Social and Behavioral Sciences. 2014. Т. 141. С. 96-100.
4. Борисов Д. В., Виноградов А. В. Разработка игры «Ships» на языке с Sharp. 2017.
5. Tešanovic S., Mitrovic P. Development of the game Hangman in assembly programming language //Telfor Journal. 2018. Т. 10. №. 2. С. 134-138
6. Маринчук А. С. Создание игры Flappy Bird в Microsoft Visual Studio на языке программирования C //Постулат. 2021. №. 1.
7. Тазабеков Д. Ю., Бужинская Н. В. Применение среды Unity для разработки обучающих компьютерных игр //Тенденции развития науки и образования. 2020. №. 58-2. С. 34-37.
8. Википедия - Шашки URL: https://ru.wikipedia.org/wiki/Шашки (дата обращения: 29.01.2021).