Доктор технических наук
Валерий Алексеевич Жарков

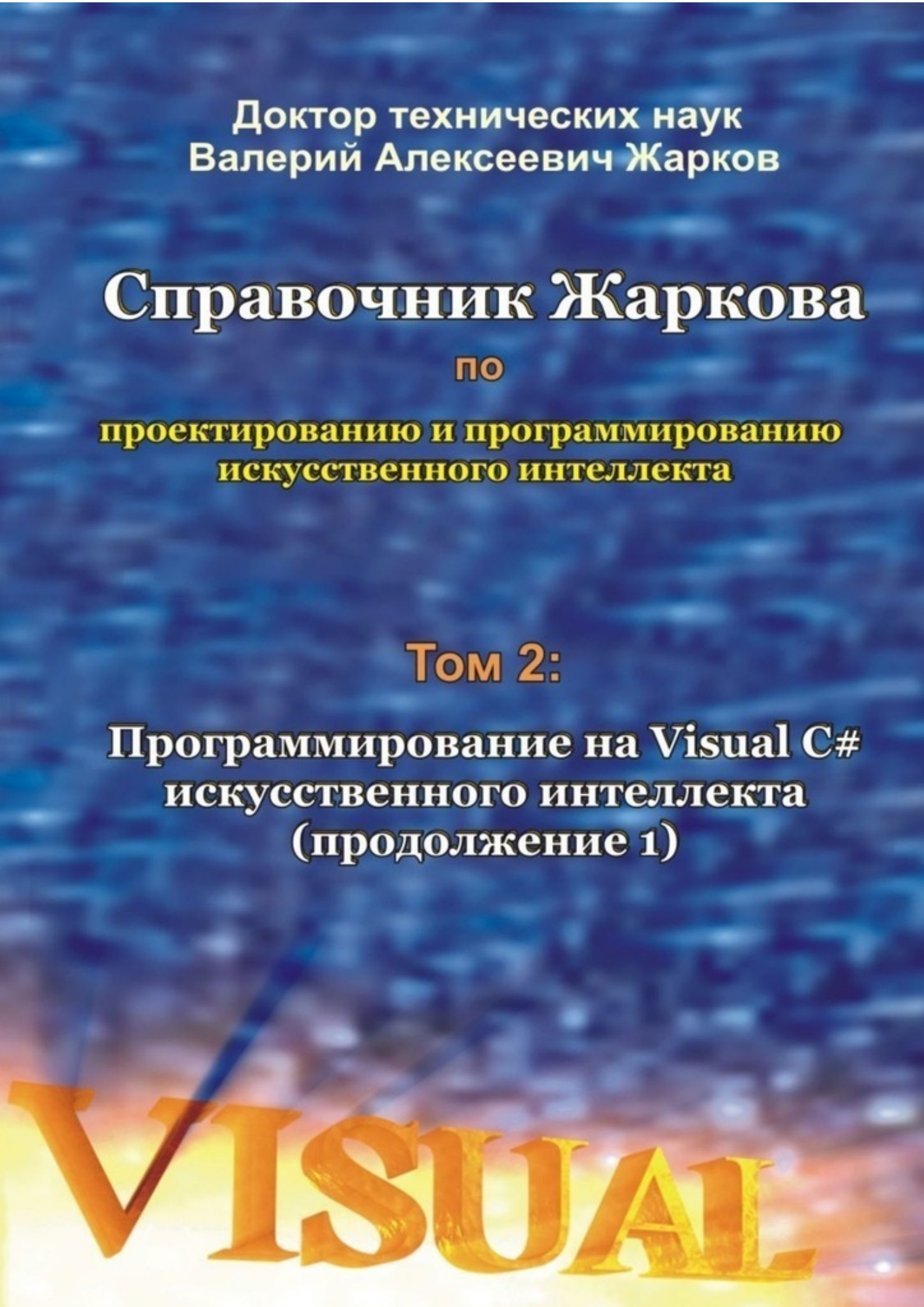
Справочник Жаркова

по

проектированию и программированию
искусственного интеллекта

Том 2:

Программирование на Visual C# искусственного интеллекта (продолжение 1)



Валерий Жарков

Справочник Жаркова по проектированию и программированию искусственного интеллекта. Том 2: Программирование на Visual C# искусственного интеллекта (продолжение 1)

«Издательские решения»

Жарков В. А.

Справочник Жаркова по проектированию и программированию искусственного интеллекта. Том 2: Программирование на Visual C# искусственного интеллекта (продолжение 1) / В. А. Жарков — «Издательские решения»,

ISBN 978-5-00-568043-3

В серии книг «Справочник Жаркова по проектированию и программированию искусственного интеллекта» собрано лучшее программирование искусственного интеллекта (ИИ) в двух- и трёхмерных играх и приложениях, разработанных как автором, так и накопленных за многие годы в Интернете, для настольных компьютеров, ноутбуков, планшетов, смартфонов. Даны инструкции по программированию ИИ с текстами программ, а также адреса в Интернете, где можно загрузить программы, графические и звуковые файлы.

ISBN 978-5-00-568043-3

Содержание

Справочник Жаркова по проектированию и программированию искусственного интеллекта. Том 2: Программирование на Visual C# искусственного интеллекта (продолжение 1)

Валерий Алексеевич Жарков

ISBN 978-5-0056-8043-3 (т. 2)
ISBN 978-5-0056-6546-1
Создано в интеллектуальной издательской системе Ridero

Введение

Это первый и единственный в мире «Справочник Жаркова по проектированию и программированию искусственного интеллекта» из нескольких томов по методологии разработки искусственного интеллекта в двухмерных и трёхмерных играх и приложениях со звуковым сопровождением для настольных компьютеров, ноутбуков и планшетов на основе самого популярного, совершенного и перспективного языка программирования высокого уровня Visual C# самой мощной в мире в области программирования корпорации Microsoft (США).

Искусственный интеллект (ИИ) – это интеллектуальная компьютерная программа, способная разумно выполнять функции, задачи и действия, обычно характерные для разумных существ и свойственные человеческому интеллекту. ИИ в игре или приложении, например, в игре между Компьютером и Человеком, умеет не только проигрывать, но и выигрывать у хорошего Игрока-человека с визуализацией результатов выигрыша. Хорошо известно, что компьютер с ИИ обыгрывает в шахматы любого гроссмейстера. Компьютер с ИИ также легко обыгрывает многих хороших игроков в карты. Если программа в виде ИИ размещена, например, в роботе или другом устройстве, то после того, как ИИ решил заданную ему задачу, ИИ выдаёт команду на выполнение устройством определённого действия. При программировании ИИ важно правильно подобрать среду разработки ИИ и язык программирования.

Среда разработки (иначе, платформа, студия) Visual Studio (или коротко VS) для визуального объектно-ориентированного проектирования приложений дают уникальную возможность быстро разрабатывать высокотехнологичные и наукоёмкие программные продукты с использованием ИИ, а также компьютерные игры с двухмерной и трёхмерной графикой также с использованием ИИ. Важно отметить, что на основе VS мы программируем не закрытые «чёрные ящики», как это делают другие известные компьютерные фирмы, а мы создаём открытые любому пользователю (для постоянного дополнения и совершенствования) программы на базе самых мощных в мире алгоритмических языков высокого уровня Visual C#, Visual Basic и Visual C++. В данном чрезвычайно насыщенном томе (заменяющей несколько других книг) мы последовательно и всесторонне, идя от простого к сложному, излагаем методологию программирования ИИ в играх и приложениях с использований двухмерных и трёхмерных изображений.

Наша основная цель – дать читателю ту информацию, которую он больше нигде не найдёт. Поэтому мы не будем дублировать известные книги по языку программирования Visual C# и давать подробные объяснения по теории языка VC#. Если у читателя возникнуть вопросы, он легко отыщет книгу по данному языку (некоторые полезные по данной тематике книги с сайта ZharkovPress.ru приведены в нашем списке литературы) и там найдёт ответ, так как терминология по всем тематикам у нас общая. Мы будем давать лишь краткие пояснения в виде комментариев в программах, чтобы начинающий пользователь постепенно осваивал базовые дисциплины программирования ИИ на языке VC#, по возможности не используя другие книги; опытному пользователю также будут полезны эти пояснения, поскольку книги по разработке ИИ на основе новых версий языка Visual C# в мире ещё не изданы. К достоинствам нашей книги, рассчитанной на широкий круг новичков и опытных специалистов, мы относим практическую направленность, простоту изложения (без описания сложных теорий, но давая ссылки на книги, в которых эти сложные теории можно изучить), наличие подробных методик и пошаговых инструкций, большое количество примеров и иллюстраций. Теперь читателям может не потребоваться изучать сложные теоретические книги, посещать длительные и дорогостоящие учебные курсы и покупать много отдельных книг. Автор это сделал за них. Читателю необходимо лишь открыть данную книгу в интересующем его разделе (мало кто будет изучать книгу от корки до корки, хотя это и желательно) и по аналогии с разделом (по прин-

ципу: делай, как я) самостоятельно программировать ИИ в практическом приложении или игре с использованием VS. И именно при проектировании ИИ в своём конкретном и профессионально интересном приложении или игре (а не отвлечённых примеров в других книгах) читатель будет изучать базовые дисциплины по данной тематике. Создавая ИИ в своём приложении или игре по методике данной и других наших книг из списка литературы (а также используя справку Help из Visual Studio, как правило, заменяющей все учебники по всем языкам), читатель сможет в одиночку работать за конструктора, технолога, математика и программиста одновременно (при разработке практических приложений) или за сценариста, режиссёра, оператора, дизайнера, художника, музыкального редактора и программиста одновременно (при разработке игр) и сэкономить недели упорного труда. Если в начальных главах инструкции по разработке ИИ в играх и приложениях на базе VS подробны (в интересах новичков), то инструкции в каждой последующей главе мы даём всё короче и короче, чтобы не повторяться и экономить место в книге.

Приводим краткое содержание данного тома справочника.

Многие приложения и игры в книге основаны на программах, или разработанных корпорацией Microsoft, или опубликованных на сайте корпорации Microsof. Поэтому эти программы являются очень мощными и могут быть использованы не только при разработке ИИ в самых разнообразных играх, но и на практике для разработки различных приложений. Структура книги продумана таким образом, чтобы читатели могли создавать на профессиональном уровне (по методологиям и программам из данной и предыдущих наших книг) свои приложения, игры и открытые графические и вычислительные системы с применением двухмерных и трёхмерных изображений и звуковых эффектов, могли вводить разнообразные исходные данные и на выходе приложения или игры получать те результаты, которые необходимы именно им и характерны для их профессиональных или непрофессиональных (игровых) интересов.

Книга предназначена для всех желающих быстро изучить основы программирования искусственного интеллекта в разнообразных двухмерных и трёхмерных компьютерных играх и приложениях на базе самого популярного, совершенного и перспективного (в мире программирования) языка высокого уровня **Visual C#** последних версий для настольных компьютеров и ноутбуков, на этих основах сразу же проектировать ИИ в сложных играх и приложениях и применять их на практике или на отдыхе в разнообразных сферах профессиональной и непрофессиональной деятельности. Также адресована начинающим и опытным пользователям, программистам любой квалификации, а также учащимся и слушателям курсов, студентам, аспирантам, учителям, преподавателям и научным работникам.

В следующем томе автор (доктор технических наук Жарков Валерий Алексеевич) продолжит описывать программирование ИИ в следующих играх и приложениях.

Вопросы, замечания и предложения по тематике книги можно направлять по email:
valery-zharkov@mtu-net.ru
или с сайта нашего «Издательства «Жарков Пресс»:
www.ZharkovPress.ru.

На этом же сайте можно познакомиться с новыми компьютерными книгами «Издательства «Жарков Пресс» (город Москва).

Часть III. Методология программирования искусственного интеллекта в играх в «Крестики-нолики» для Игрока с Компьютером и двух Игроков

Глава 11. Методика программирования искусственного интеллекта в игре в «Крестики-нолики» на сетке 3x3

11.1. Общие сведения

В данной главе мы опишем методику разработки игры, которая предусматривает сразу оба варианта игры: Игрок 1 – Игрок 2 и Игрок – Компьютер. Данную игру мы будем разрабатывать, следуя проекту из книги [Philip Conrod, Lou Tylee. Programming Games with Visual C#. Publisher: Kidware Software LLC; 15 edition (11 July 2017)], но с нашими усовершенствованиями для современной версии VS.

В игре не будут использоваться графические файлы. Поэтому игроки будут воздействовать на надписи типа Label. Перед началом и после победного или ничейного окончания игры звучит соответствующая музыка. Программа состоит из трёх частей:

1) модель игры, которой в программе соответствует файл Form1.cs [Design] *;

2) интерфейс пользователя (user interface – UI), которому в программе соответствует файл Form1.cs;

3) искусственный интеллект (artificial intellect – AI) в виде метода ComputerTurn (), который размещён в файле Form1.cs.

Для основного и самого сложного варианта игры Игрок – Компьютер (для которого надо выбрать второй переключатель) используются все три части программы.

А для упрощённого варианта игры Игрок 1 – Игрок 2 (который установлен по умолчанию) используются только первые две части программы (вместо искусственного интеллекта Компьютера используется интеллект Игрока 2).

11.2. Правила игры

1. После разработки игры по описанной далее методике выполняем проект (нажимаем F5 или Ctrl+F5). Игра появится в своём «остановленном» состоянии, сетка очищена и ожидает Вас для выбора игровых опций (один или два игрока и, если один игрок, который идёт сначала и как умный Вы хотите, чтобы компьютер был). Сетка отключена – никакие метки не могут быть сделаны (путём щелчка по сетке). Я выбрал одну игру для одного игрока, куда я иду сначала (предоставление мне X) и умный компьютер (рис. 11.1).

2. Нажимаем кнопку Start Game, чтобы начать играть. Его заголовок изменится (теперь читаем как Stop Game) и групповые блоки опций, и кнопка Exit станет отключённой (рис. 11.2). Метка Label наверху сетки говорит, что очередь за X. X всегда идёт сначала в этой игре (является ли это Вами, как человеческим игроком, или компьютером).

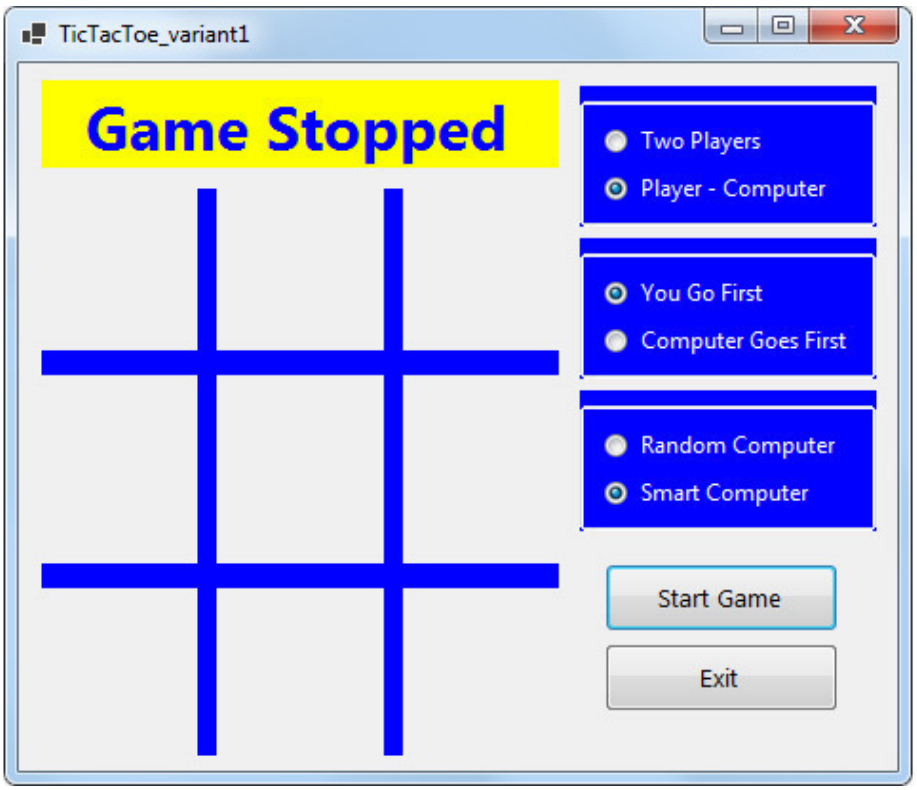


Рис. 11.1.

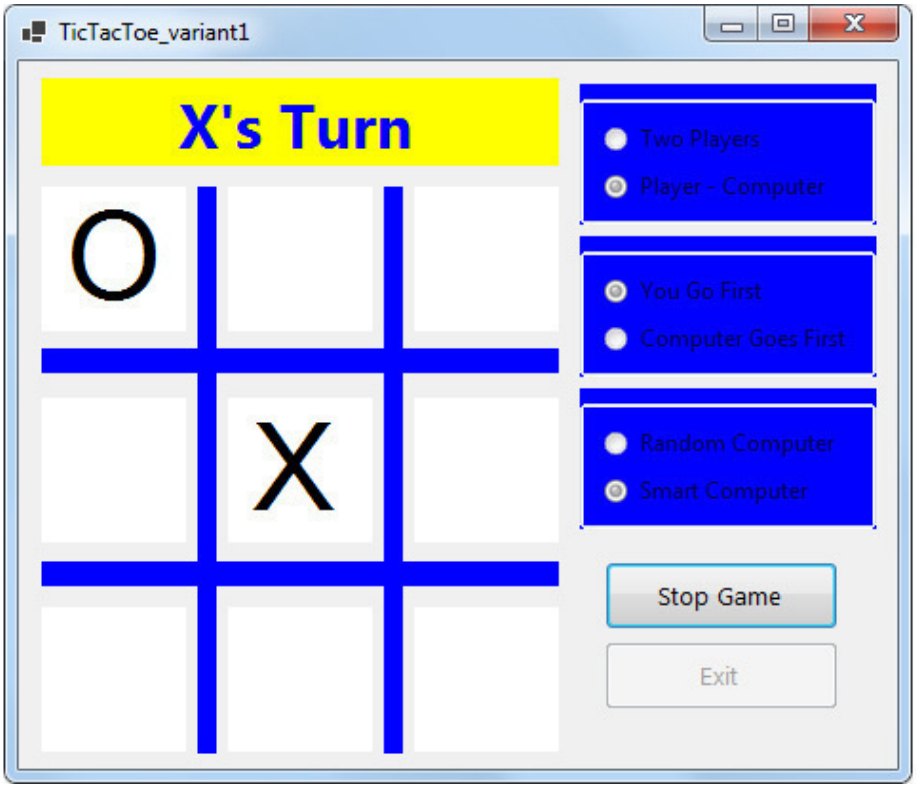


Рис. 11.2.

В этом состоянии игры мы нажимаем на желаемый квадрат. Компьютер тогда поместит свою метку, передавая нам очередь. После каждой метки плата исследована на победу или ничью. Вы продолжаете чередовать игру, пока нет победы, сетка полна или пока Вы не нажимаете **Stop Game**. Игра работает таким же путём для двух игроков, с этими двумя игроками, чередующими щелчки, отмечающие сетку.

3. Вводим метку (X); я выбрал центральный квадрат, и компьютер сразу поместил свою метку (O) в левом верхнем углу (рис. 11.2).

4. Я пробую левый нижний угол, а компьютер блокирует меня от победы (рис. 11.3).

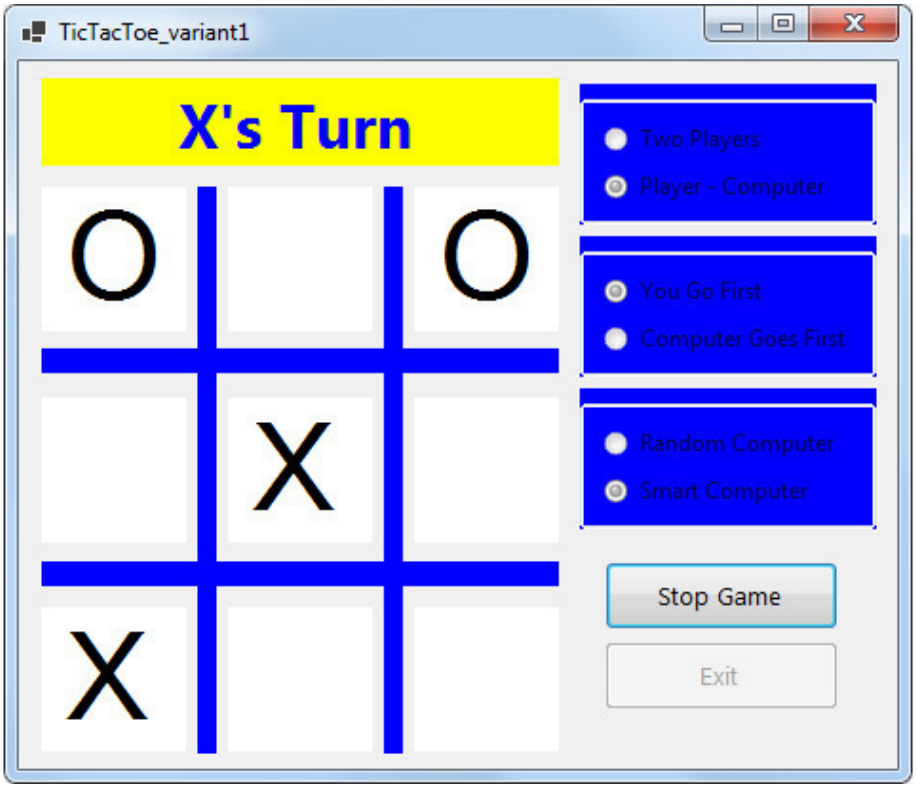


Рис. 11.3.

5. Я блокирую возможную победу компьютера путём помещения X в среднее поле верхнего ряда. Затем компьютер блокирует меня с перемещением к середине нижнего ряда (рис. 11.4).

Похоже, здесь никто не собирается победить.

6. Я продолжаю играть, ставлю X направо от средней строки.

7. Компьютер блокируем меня, ставя O слева в среднем ряду (рис. 11.5).

8. Я ставлю X в оставшейся клетке, мы закончили вничью (рис. 11.6).

Похоже, компьютер довольно умён, у компьютера сложно выиграть.

Так по очереди Игрок и Компьютер стараются получить как можно больше побед за установленное заранее Игроком количество игр.

После любой победы Компьютера в игру может вступить новый (из очереди) Игрок.

9. Чтобы в игре участвовали два игрока (для варианта игры Игрок 1 – Игрок 2), следует выбрать переключатель Two Players. Далее Игрок 1 играет против Игрока 2 по приведённым выше правилам. Если выбрать переключатель Random Computer или Two Players, то можно победить компьютер или второго игрока (рис. 11.7).

10. Нажмите Exit для остановки игры.

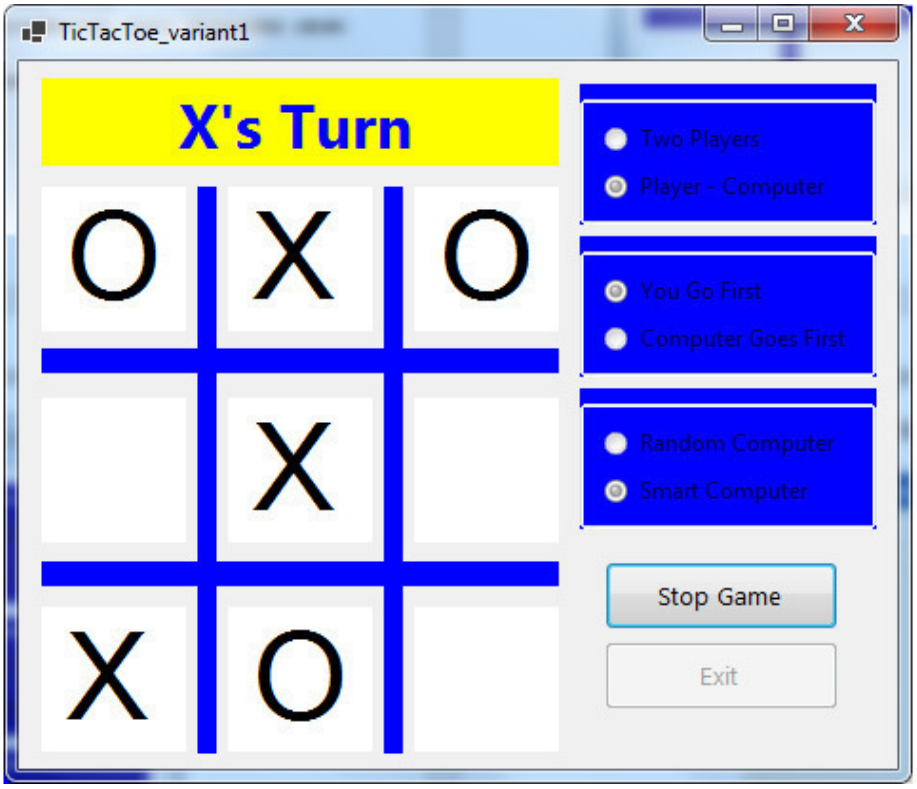


Рис. 11.4.

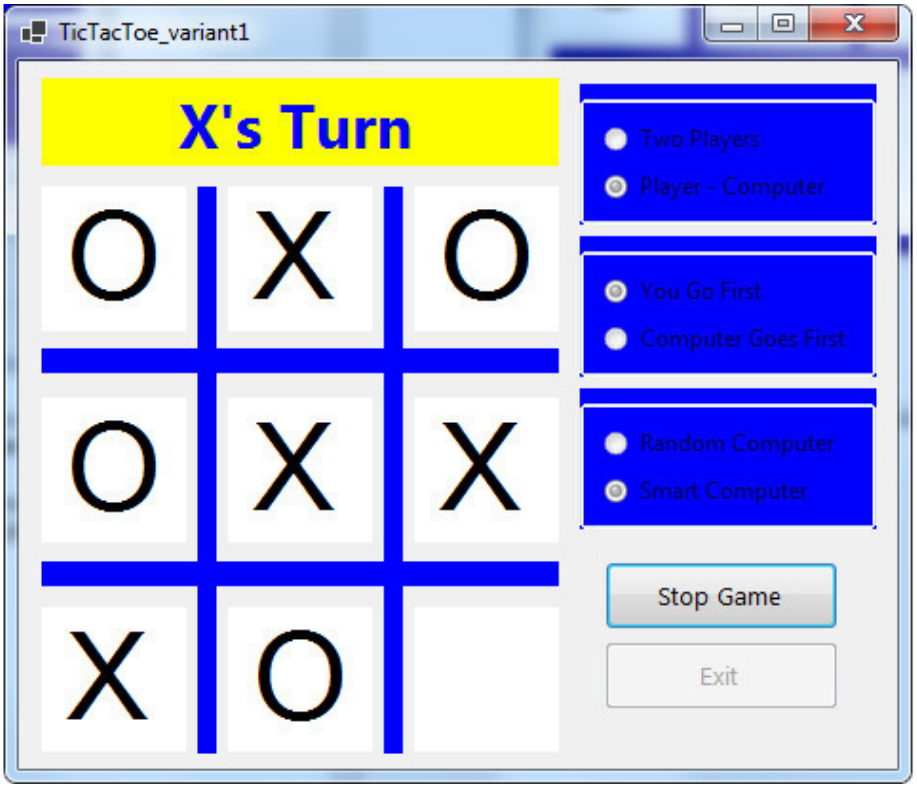


Рис. 11.5.

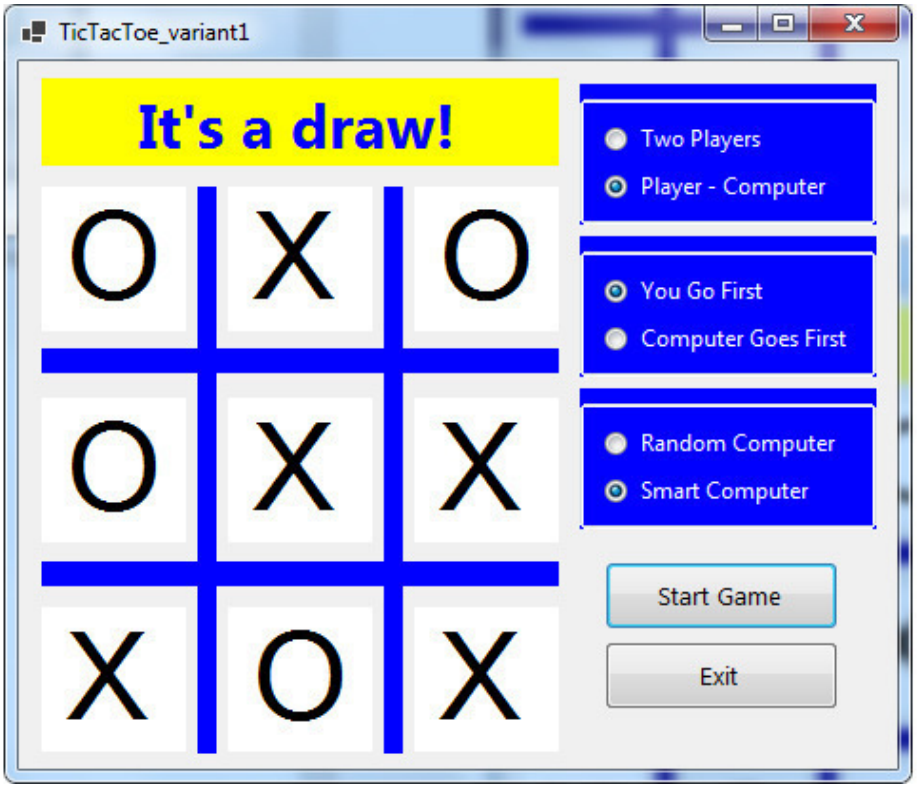


Рис. 11.6.

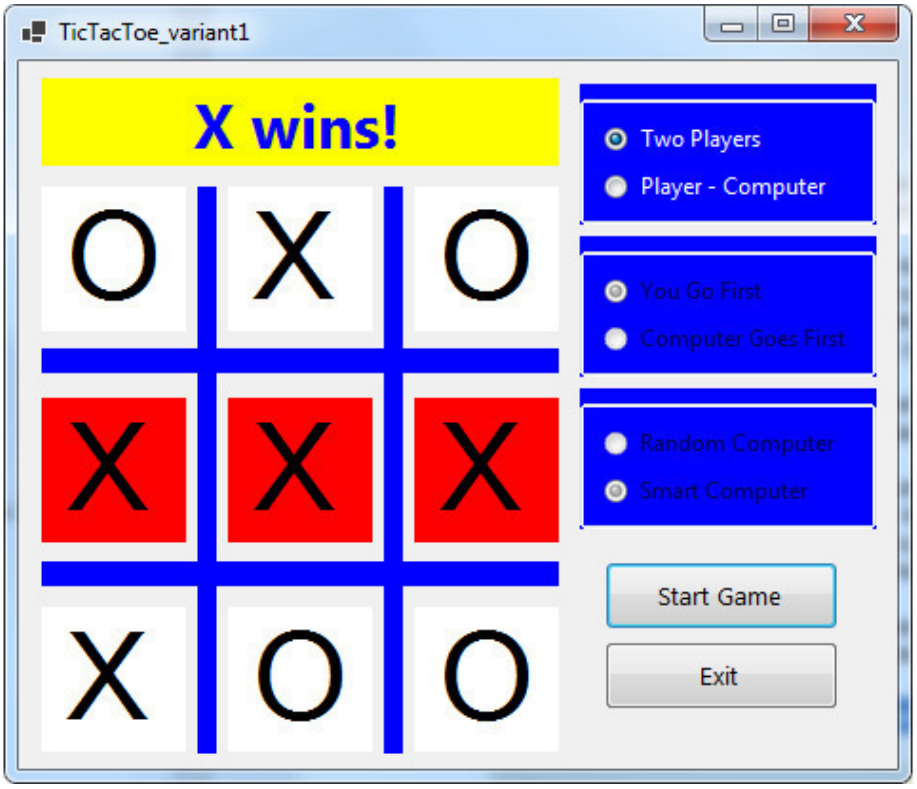


Рис. 11.7.

На основании этих правил можно сформулировать другие правила, и любые правила ввести в справочную форму, которую можно разработать по приведённым в других главах методикам.

11.3. Создание проекта

Создаём проект по обычной схеме: в VS в панели New Project в окне Project types выбираем тип проекта Visual C#, Windows, в окне Templates выделяем шаблон Templates, Visual C#, Windows Classic Desktop, Windows Forms App (.NET Framework), в окне Name записываем любое имя проекта, например, TicTacToe_variant1 и щёлкаем OK. Создаётся проект, появляется форма Form1 на экране в режиме проектирования.

С панели инструментов Toolbox переносим на форму элементы управления Control и устанавливаем для формы и элементов управления следующие свойства:

Form1 Form:
Property Name Property Value
Name frmTicTacToe
BackColor White
Text TicTacToe_variant1
FormBorderStyle Fixed Single
StartPosition CenterScreen
Size 478; 410

label1 Label:
Property Name Property Value
Name lblMessage
AutoSize False
BackColor Yellow
ForeColor Blue
BorderStyle Fixed3D
Font Style Bold
Font Size 24
Text X's Move
TextAlign MiddleCenter
Location 12; 9
Size 272; 46

label2 Label:
Property Name Property Value
Name lblBox1
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 12; 66
Size 76; 76

label3 Label:
Property Name Property Value
Name lblBox2

Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 110; 66
Size 76; 76

label4 Label:
Property Name Property Value
Name lblBox3
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 208; 66
Size 76; 76

label5 Label:
Property Name Property Value
Name lblBox4
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 12; 177
Size 76; 76

label6 Label:
Property Name Property Value
Name lblBox5
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 110; 177
Size 76; 76

label7 Label:
Property Name Property Value
Name lblBox6
Font Microsoft Sans Serif; 48pt

AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 208; 177
Size 76; 76

label8 Label:
Property Name Property Value
Name lblBox7
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 12; 287
Size 76; 76

label9 Label:
Property Name Property Value
Name lblBox8
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 110; 287
Size 76; 76

label10 Label:
Property Name Property Value
Name lblBox9
Font Microsoft Sans Serif; 48pt
AutoSize False
BorderStyle None
Font Size 48
Text X
TextAlign MiddleCenter
Location 208; 287
Size 76; 76

label11 Label (used as grid line – make very skinny):
Property Name Property Value
Name label11
AutoSize False
BackColor Blue

BorderStyle None
Text [blank]
Location 12; 151
Size 272; 13

label12 Label (used as grid line – make very skinny):
Property Name Property Value
Name label12
AutoSize False
BackColor Blue
BorderStyle None
Text [blank]
Location 12; 263
Size 272; 13

label13 Label (used as grid line – make very skinny):
Property Name Property Value
Name label13
AutoSize False
BackColor Blue
BorderStyle None
Text [blank]
Location 94; 66
Size 10; 298

label14 Label (used as grid line – make very skinny):
Property Name Property Value
Name label14
AutoSize False
BackColor Blue
BorderStyle None
Text [blank]
Location 192; 66
Size 10; 298

groupBox1 Group Box:
Property Name Property Value
Name grpPlayers
AutoSize False
BackColor Blue
Text [blank]
Location 295; 12
Size 156; 74

radioButton1 Radio Button:
Property Name Property Value
Name rdoTwoPlayers
AutoSize False
ForeColor White

Font Size 9
Text Two Players
Checked True
Location 13; 19
Size 91; 19

radioButton2 Radio Button:
Property Name Property Value
Name rdoOnePlayer
AutoSize False
ForeColor White
Font Size 9
Text Player – Computer
Location 13; 44
Size 137; 19

groupBox2 Group Box:
Property Name Property Value
Name grpFirst
AutoSize False
BackColor Blue
Text [blank]
Location 295; 92
Size 156; 74

radioButton3 Radio Button:
Property Name Property Value
Name rdoYouFirst
AutoSize False
ForeColor White
Font Size 9
Text You Go First
Checked True
Location 13; 19
Size 91; 19

radioButton4 Radio Button:
Property Name Property Value
Name rdoComputerFirst
AutoSize False
ForeColor White
Font Size 9
Text Computer Goes First
Location 13; 44
Size 137; 19

groupBox3 Group Box:
Property Name Property Value
Name grpComputer

AutoSize False
BackColor Blue
Text [blank]
Location 295; 172
Size 156; 74

radioButton5 Radio Button:
Property Name Property Value
Name rdoRandom
AutoSize False
ForeColor White
Font Size 9
Text Random Computer
Checked True
Location 13; 19
Size 130; 19

radioButton6 Radio Button:
Property Name Property Value
Name rdoSmart
AutoSize False
ForeColor White
Font Size 9
Text Smart Computer
Location 13; 44
Size 115; 19

button1 Button:
Property Name Property Value
Name btnStartStop
AutoSize False
Font Size 10
Text Start Game
Location 308; 263
Size 123; 36

button2 Button:
Property Name Property Value
Name btnExit
AutoSize False
Font Size 10
Text Exit
Location 308; 305
Size 123; 36

Форма приняла вид, показанный на рис. 11.8.

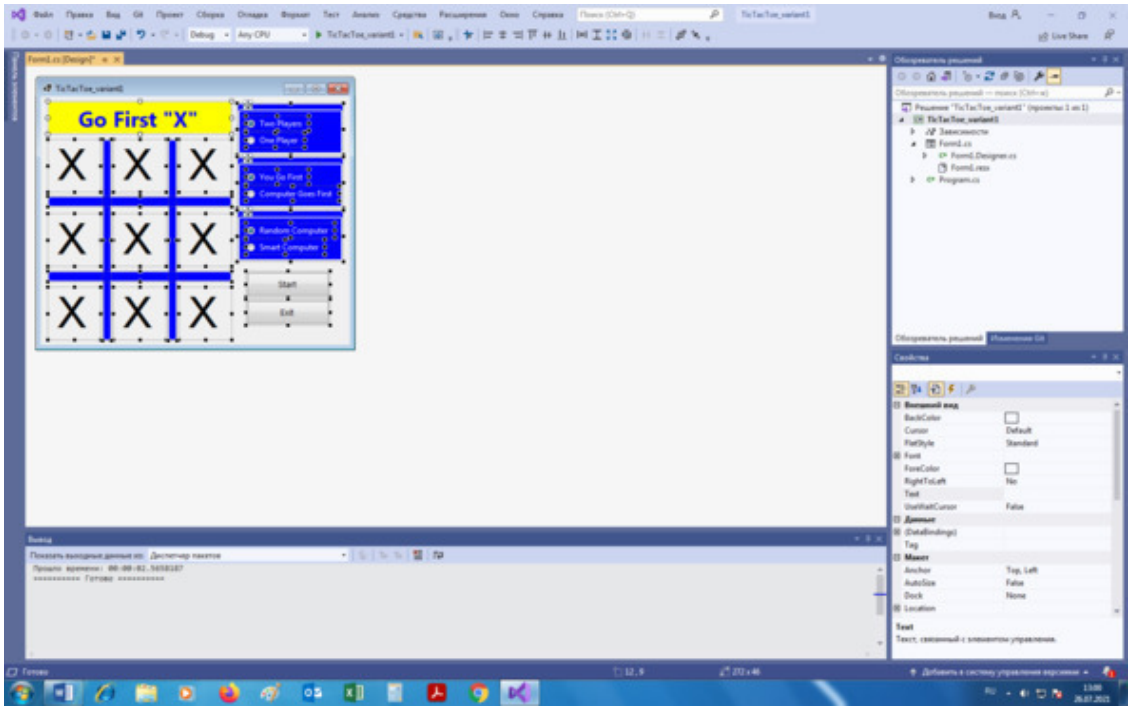
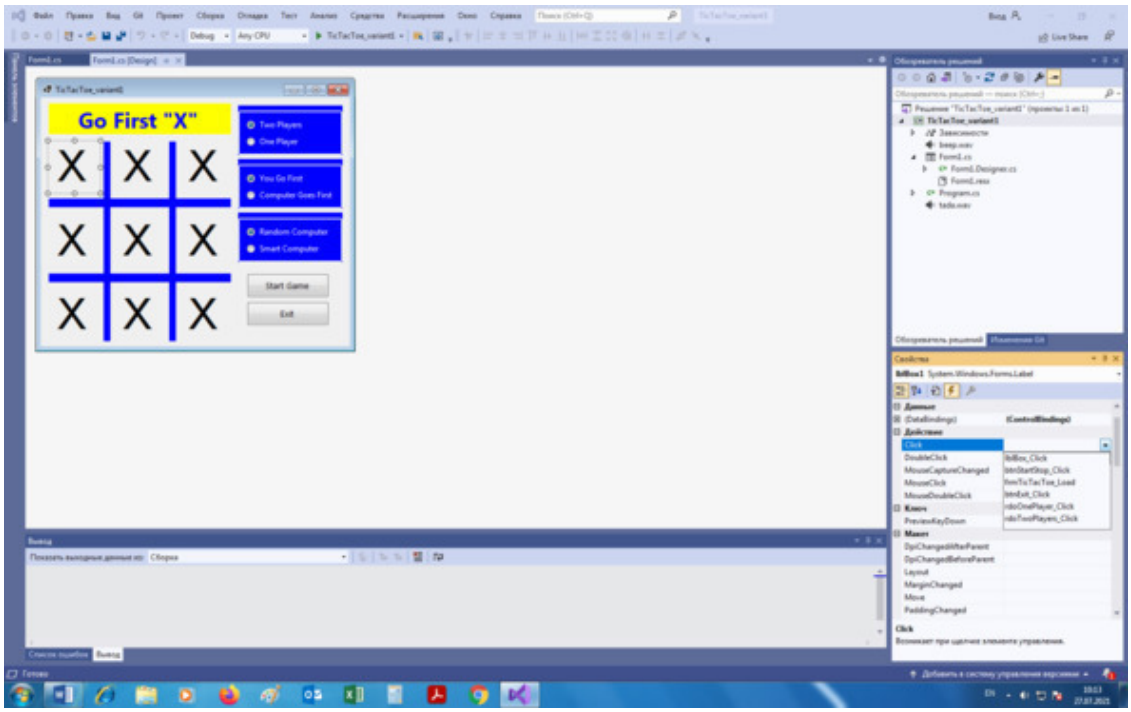


Рис. 11.8. Форма Form1 после проектирования.

Выделяем элемент управления lblBox1, в панели Properties для этого элемента на вкладке Events выделяем событие Click, щёлкаем стрелку и из выпавшего списка выбираем событие lblBox1_Click (рис. 11.9).

Аналогично поступаем для остальных элементов управления lblBox2, lblBox3, lblBox4, lblBox5, lblBox6, lblBox7, lblBox8, lblBox9.



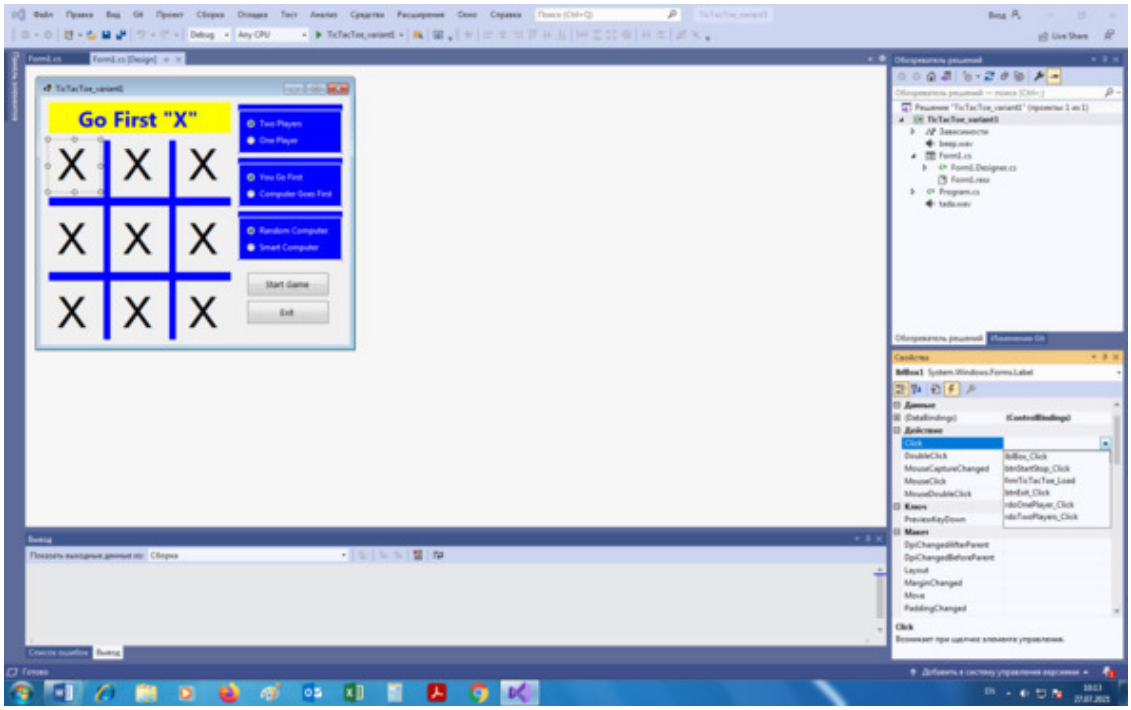


Рис. 11.9. Выделяем lblBox1 и выбираем событие lblBox1_Click.

На рис. 11.12 слева – сетка для игры Tic Tac Toe. Элементы управления Label имеются в большом количестве. Элемент управления Label используется, чтобы сказать Вам ситуацию в игре. Метки используются для маркировки X и O в сетке (несмотря на то, что только X показывают в этом режиме проектирования). Тонкие элементы управления Label используются для формирования тёмно-синей сетки. Справа три средства управления GroupBox и две Button. Каждый GroupBox содержит два средства управления типа переключателя RadioButton, используемые для установления игровых опций. Эти два кнопочных управления Button используются, чтобы запустить и остановить игру и выйти из программы.

Если в игре применяются звуковые файлы, то их можно разместить в одной папке с именем, например, Sounds, а можно разместить непосредственно в проект. Добавляем в проект звуковые файлы beep. wav и tada. wav по стандартной схеме: выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, Existing Item, в панели Add Existing Item в окне «Files of type» выбираем «All Files», в центральном окне находим (в папке файлы, например, из Интернета) и с нажатой клавишей Ctrl выделяем имена файлов и щёлкаем кнопку Add. В панели Solution Explorer мы увидим эти файлы (рис. 11.10).

Дважды щёлкая по имени файла в панели Solution Explorer, любой файл можно открыть, изучить и редактировать.

Схема записи и вывода справочной информации, например, с правилами игры после выбора команды Справка (например, для элемента управления MenuStrip) уже была дана в другой главе, а также в книгах с сайта ZharkovPress.ru.

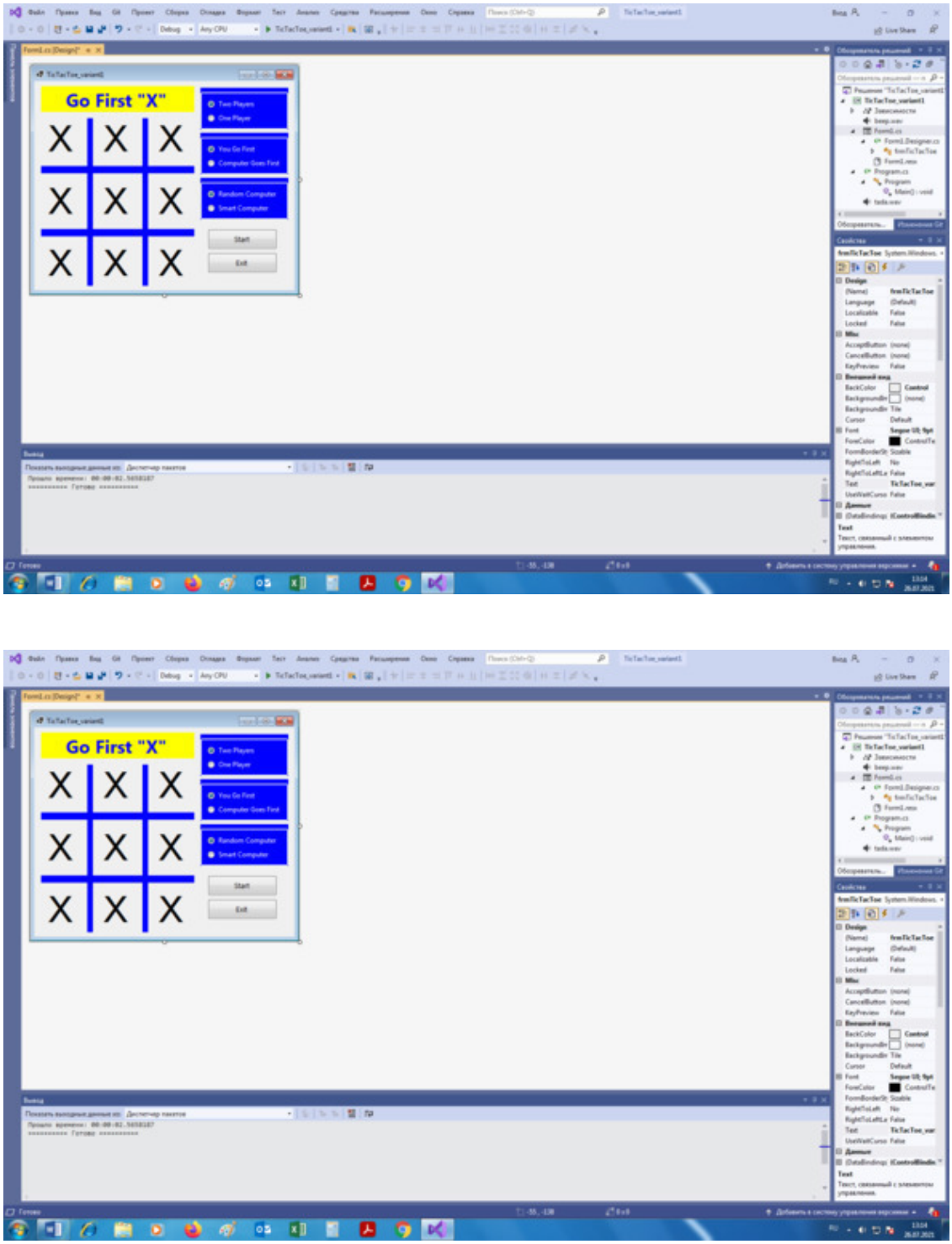


Рис. 11.10. Панели Solution Explorer (слева) и Properties (справа).

11.4. Код и запуск программы

Открываем файл Form1.cs (например, по схеме: File, Open, File) и в классе Form1 записываем следующие переменные и методы.

Листинг 11.1. Переменные и методы.

```
Random myRandom = new Random ();
Label [] boxArray = new Label [9];
bool xTurn;
bool canClick = false;
int numberClicks;
String [] possibleWins = new String [8];
bool gameOver;
//System.Media.SoundPlayer drawSound;
//System.Media.SoundPlayer winSound;

private void lblBox_Click (object sender, EventArgs e)
{
String whoWon = «»;
int i;
if (canClick)
{
// see which box is clicked
Label clickedBox;
clickedBox = (Label) sender;
// last digit of name (-1) is index
i = Convert.ToInt32(clickedBox.Name [
clickedBox.Name. Length – 1].ToString ()) – 1;
// if already clicked then exit
if (boxArray [i].Text!= «»)
return;
numberClicks++;
if (xTurn)
{
boxArray [i].Text = «X»;
xTurn = false;
lblMessage. Text = «O's Turn»;
}
else
{
boxArray [i].Text = «O»;
xTurn = true;
lblMessage. Text = «X's Turn»;
}
// check for win – will establish a value for WhoWon
whoWon = CheckForWin ();
if (whoWon!= «»)
```

```
{
//winSound.Play ();
lblMessage. Text = whoWon + " wins!»;
gameOver = true;
btnStartStop.PerformClick ();
return;
}
else if (numberClicks == 9)
{
// draw
//drawSound.Play ();
lblMessage. Text = «It's a draw!»;
gameOver = true;
btnStartStop.PerformClick ();
return;
}
if (rdoOnePlayer.Checked)
if ((xTurn && rdoComputerFirst.Checked) ||
(!xTurn && rdoYouFirst.Checked))
ComputerTurn ();
}
}
private String CheckForWin ()
{
String winner = «»;
int [] boxNumber = new int [3];
String [] mark = new String [3];
// check all possible for wins
for (int i = 0; i <8; i++)
{
for (int j = 0; j <3; j++)
{
boxNumber [j] = Convert.ToInt32 (
possibleWins[i][j].ToString ());
mark [j] = boxArray [boxNumber [j]].Text;
}
if (mark [0] == mark [1] && mark [0] == mark [2] &&
mark [1] ==
mark [2] && mark [0]!= «»)
{
// we have a winner
winner = mark [0];
for (int j = 0; j <3; j++)
boxArray[boxNumber[j]].BackColor =
Color.Red;
}
}
return (winner);
}
```

```
private void ComputerTurn ()
{
int selectedBox;
int i, n;
int j, k;
String computerMark, playerMark, markToFind;
int [] boxNumber = new int [3];
String [] mark = new String [3];
int emptyBox;
int [] bestMoves = {4, 0, 2, 6, 8, 1, 3, 5, 7};
if (rdoRandom.Checked)
{
// random logic
// put mark in Nth available square
n = myRandom.Next (9 – numberClicks) +1;
i = 0;
for (selectedBox = 0; selectedBox <9;
selectedBox++)
{
if (boxArray [selectedBox].Text == «»)
i++;
if (i == n)
break;
}
// put mark in SelectedBox
lblBox_Click (boxArray [selectedBox], null);
}
else
{
// smart computer
// determine who has what mark
if (rdoComputerFirst.Checked)
{
computerMark = «X»;
playerMark = «O»;
}
else
{
computerMark = «O»;
playerMark = «X»;
}
// Step 1 (K = 1) – check for win —
// see if two boxes hold
// computer mark and one is empty
// Step 2 (K = 2) – check for block —
// see if two boxes hold
// player mark and one is empty
for (k = 1; k <= 2; k++)
{
```

```
if (k == 1)
markToFind = computerMark;
else
markToFind = playerMark;
for (i = 0; i <8; i++)
{
n = 0;
emptyBox = 0;
for (j = 0; j <3; j++)
{
boxNumber [j] = Convert.ToInt32 (
possibleWins[i][j].ToString ());
mark [j] = boxArray [boxNumber [j]].Text;
if (mark [j] == markToFind)
n++;
else if (mark [j] == «»)
emptyBox = boxNumber [j];
}
if (n == 2 && emptyBox!= 0)
{
// mark empty box to win (K = 1) or
// block (K = 2)
lblBox_Click (boxArray [emptyBox], null);
return;
}
}
}
// Step 3 – find next best move
for (i = 0; i <9; i++)
{
if (boxArray [bestMoves [i]].Text == «»)
{
lblBox_Click (boxArray [bestMoves [i]], null);
return;

}
}
}
}
```

Дважды щёлкаем по форме в режиме проектирования или в панели Properties (для формы) на вкладке Events дважды щёлкаем по имени события Load. Появившийся шаблон метода после записи нашего кода принимает следующий вид.

Листинг 11.2. Метод.

```
private void frmTicTacToe_Load (object sender, EventArgs e)
{
//’establish array
```

```
boxArray [0] = lblBox1;
boxArray [1] = lblBox2;
boxArray [2] = lblBox3;
boxArray [3] = lblBox4;
boxArray [4] = lblBox5;
boxArray [5] = lblBox6;
boxArray [6] = lblBox7;
boxArray [7] = lblBox8;
boxArray [8] = lblBox9;
// possible wins
possibleWins [0] = «012»;
possibleWins [1] = «345»;
possibleWins [2] = «678»;
possibleWins [3] = «036»;
possibleWins [4] = «147»;
possibleWins [5] = «258»;
possibleWins [6] = «048»;
possibleWins [7] = «246»;
// clear boxes
for (int i = 0; i <9; i++)
boxArray [i].Text = «»;
lblMessage. Text = «Game Stopped»;
grpFirst. Enabled = false;
grpComputer. Enabled = false;
//drawSound = new System.Media.SoundPlayer (
//Application.StartupPath + "\\beep. wav»);
//winSound = new System.Media.SoundPlayer (
//Application.StartupPath + "\\tada. wav»);

}
```

Дважды щёлкаем по первой кнопке в режиме проектирования или в панели Properties (для этой кнопки) на вкладке Events дважды щёлкаем по имени события Click. Появившийся шаблон метода после записи нашего кода принимает следующий вид.

Листинг 11.3. Метод.

```
private void btnStartStop_Click (object sender, EventArgs e)
{
if (btnStartStop. Text == «Start Game»)
{
btnStartStop. Text = «Stop Game»;
grpPlayers. Enabled = false;
grpFirst. Enabled = false;
grpComputer. Enabled = false;
btnExit. Enabled = false;
xTurn = true;
lblMessage. Text = «X’s Turn»;
// reset boxes
```

```
for (int i = 0; i <9; i++)
{
boxArray [i].Text = «»;
boxArray[i].BackColor = Color. White;
}
canClick = true;
numberClicks = 0;
gameOver = false;
if (rdoComputerFirst.Checked &&
rdoOnePlayer.Checked)
ComputerTurn ();
}
else
{
btnStartStop. Text = «Start Game»;
if (!gameOver)
lblMessage. Text = «Game Stopped»;
grpPlayers. Enabled = true;
if (rdoOnePlayer.Checked)
{
grpFirst. Enabled = true;
grpComputer. Enabled = true;
}
btnExit. Enabled = true;
canClick = false;

}
}
```

Дважды щёлкаем по второй кнопке в режиме проектирования или в панели Properties (для этой кнопки) на вкладке Events дважды щёлкаем по имени события Click. Появившийся шаблон метода после записи нашего кода принимает следующий вид.

Листинг 11.4. Метод.

```
private void btnExit_Click (object sender, EventArgs e)
{
this.Close ();
}
```

В панели Properties для второго переключателя на вкладке Events дважды щёлкаем по имени события Click. Появившийся шаблон метода после записи нашего кода принимает следующий вид.

Листинг 11.5. Метод.

```
private void rdoOnePlayer_Click (object sender, EventArgs e)
{
grpFirst. Enabled = true;
```

```
        grpComputer. Enabled = true;

}
```

В панели Properties для первого переключателя на вкладке Events дважды щёлкаем по имени события Click. Появившийся шаблон метода после записи нашего кода принимает следующий вид.

Листинг 11.6. Метод.

```
        private void rdoTwoPlayers_Click (object sender, EventArgs e)
        {
        grpFirst. Enabled = false;
        grpComputer. Enabled = false;

}
```

Если мы желаем, чтобы при завершении игры с результатом «Ничья» или «Победа» звучала соответствующая музыка, необходимо раскомментировать строки кода, в которых есть слово «Sound», и отладить проект.

Строим и запускаем программу на выполнение обычным образом: Build, Build Selection; Debug, Start Without Debugging.

В ответ Visual Studio выводит форму Form1 с показанным выше полем игры из 9 клеток.

Далее Игрок играет с Компьютером или Игрок 1 играет с Игроком 2 в крестики-нолики согласно приведённым выше правилам.

По методике данной главы можно разрабатывать самые разнообразные игры в крестики-нолики как для игры Игрока с Компьютером, так и для игры Игрока 1 с Игроком 2.

Глава 12. Методика программирования искусственного интеллекта в игре в «Крестики-нолики» на сетке 6x7

12.1. Общие сведения

Разработаем методику проектирования и программирования типичной и широко распространённой игры типа игры в крестики-нолики. Известно много вариантов этой игры как в ручном, так и в компьютерном варианте.

В ручном варианте смысл игры состоит в том, что, например, в тетради в клеточку на поле из квадратов, например, 6 x 7 квадратов (цифры соответствуют количеству квадратов по оси абсцисс «x» и оси ординат «y») два игрока по очереди выделяют квадраты крестиками и ноликами, стараясь первым выделить горизонтальную, вертикальную или диагональную линию из определённого количества крестиков или ноликов.

Данную компьютерную игру мы будем разрабатывать, следуя статье [см. Список литературы: Article from the website: www.codeproject.com/netcf/Connect4AB.asp: Labib B. Connect4 using Alpha-Beta Search algorithm], но с нашими усовершенствованиями для современной версии Visual Studio.

В данной игре (игрока с компьютером) не будут использоваться графические файлы. Поэтому сначала методом FillRectangle класса Graphics на форме рисуется прямоугольник (поле игры), заполненный синим цветом RoyalBlue:

```
e.Graphics.FillRectangle (new SolidBrush(Color.RoyalBlue),
15, 70, 210, 180);
```

где 15, 70 – координаты верхнего левого угла прямоугольника (от верхнего левого угла формы, когда ось «y» направлена вниз);

210, 180 – размеры прямоугольника соответственно по осям «x» и «y».

Далее методом FillEllipse класса Graphics в этом прямоугольнике рисуются 6 x 7 кружочков (ноликов) диаметром 27 пикселей белого цвета White:

```
g.FillEllipse (new SolidBrush (Color. White),
x +3, y +3, 27, 27);
```

Когда игрок щёлкает мышью по белому кружочку, этот белый кружочек заменяется таким же кружочком, но жёлтого цвета Yellow:

```
g.FillEllipse (new SolidBrush (Color. Yellow),
x +3, y +3, 27, 27);
```

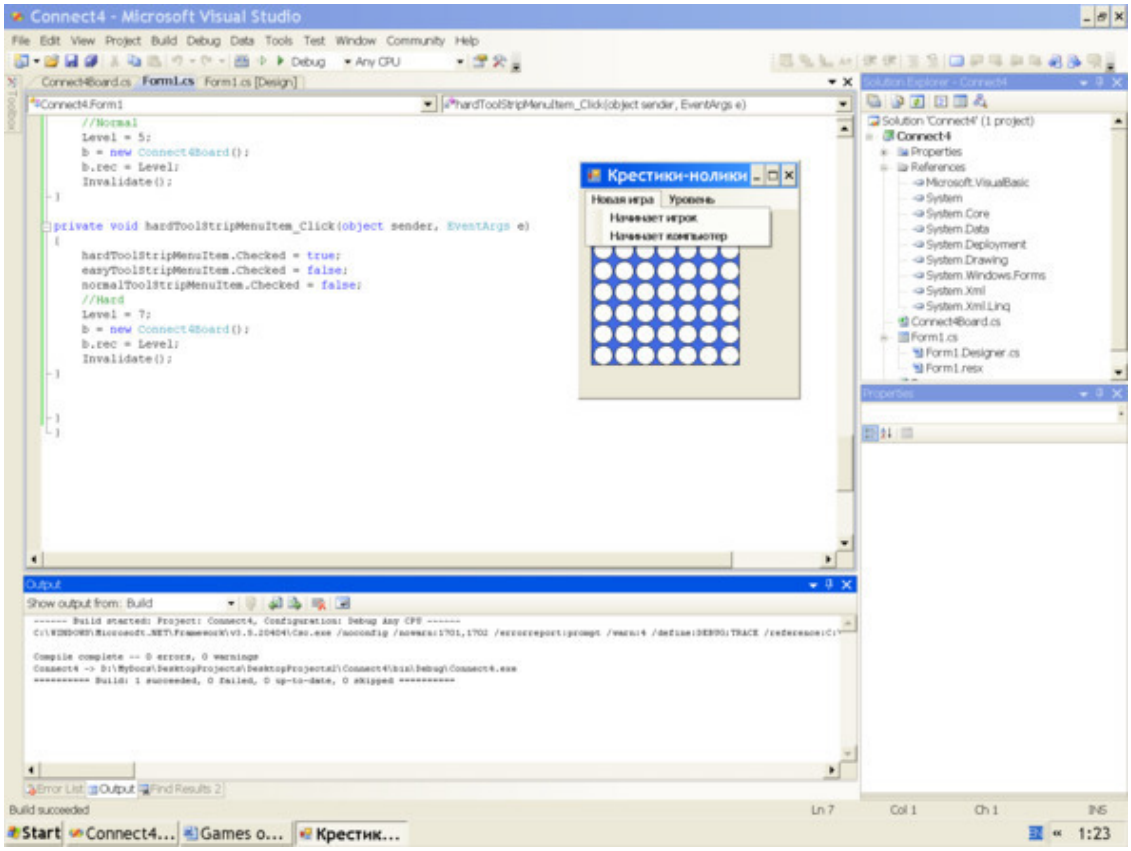
После расчёта вариантов (по определённому математическому алгоритму, реализованному в показанной далее программе) компьютер заменяет один белый кружочек таким же кружочком красного цвета Red:

```
g.FillEllipse (new SolidBrush(Color.Red),
x +3, y +3, 27, 27);
```

Так по очереди игрок и компьютер выделяют жёлтые и красные кружочки, стараясь первым выделить горизонтальную, вертикальную или диагональную линию из 4-х кружочков одинакового цвета. После каждого щелчка игроком поля игры (мышью) он слышит сигнал Beep (по-русски: Бип).

12.2. Правила игры

1. После запуска игры на форме появляется поле игры, состоящее из 7 x 6 белых кружочков (рис. 12.1). Напомним, что цифры соответствуют количеству квадратов по осям «x» и «y». В меню «Новая игра» по умолчанию установлен режим «Начинает игрок», но можно выбрать «Начинает компьютер» (рис. 12.1). В меню Уровень по умолчанию установлен режим Обычный, но можно выбрать Нормальный или Сложный (рис. 12.2).



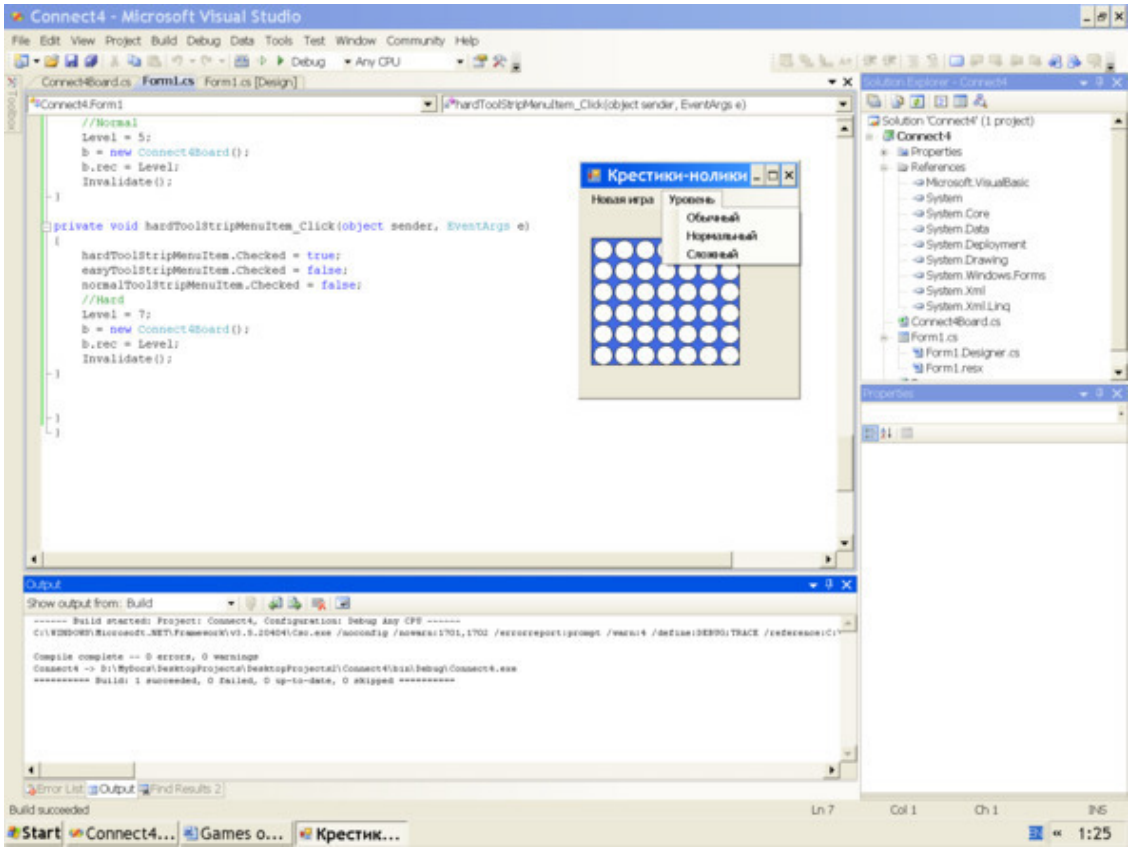


Рис. 12.1. Меню Новая игра. **Рис. 12.2.** Меню Уровень (Level).

2. Игрок щёлкает мышью по выбранному им белому кружочку, этот белый кружочек заменяется жёлтым кружочком. После каждого щелчка игроком поля игры он слышит сигнал Beep.

После расчёта вариантов (по определённому математическому алгоритму, реализованному в показанной далее программе) компьютер заменяет один белый кружочек таким же кружочком красного цвета (рис. 12.3).

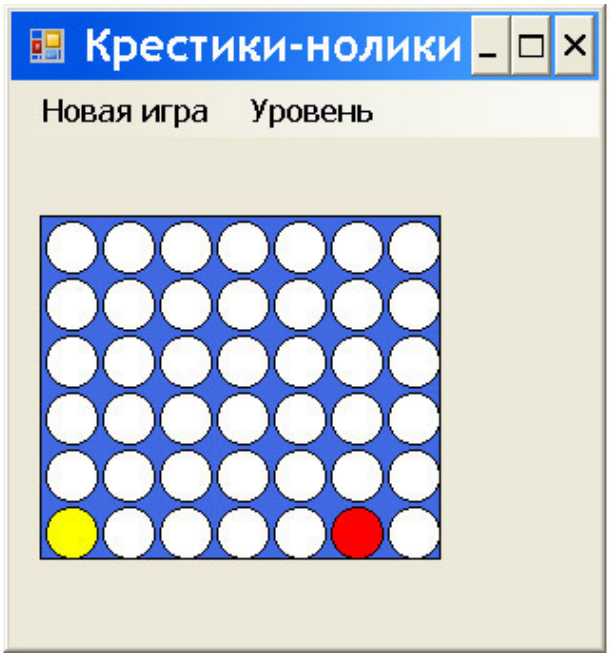
3. Так по очереди игрок и компьютер выделяют жёлтые и красные кружочки, стараясь первым выделить горизонтальную, вертикальную или диагональную линию из 4-х кружочков одинакового цвета. Но если игрок или компьютер «видят», что следующим ходом его соперник соберёт линию из 4-х кружочков одинакового цвета, он должен (чтобы не проиграть) на этот квадрат поместить свой кружок (даже в ущерб своей тактике игры). Середина игры показана на рис. 12.4.

4. Как только игрок или компьютер первым соберёт линию из 4-х кружочков своего цвета, появляется информационная панель Show класса MessageBox с сообщением для игрока, либо «Вы победили» (рис. 12.5), либо «Вы проиграли» (рис. 12.6).

На этой панели нужно щёлкнуть OK.

5. Для начала новой игры следует в меню «Новая игра» выбрать режим «Начинает игрок» (или «Начинает компьютер»).

6. Для закрытия игры следует на форме выбрать значок Close.



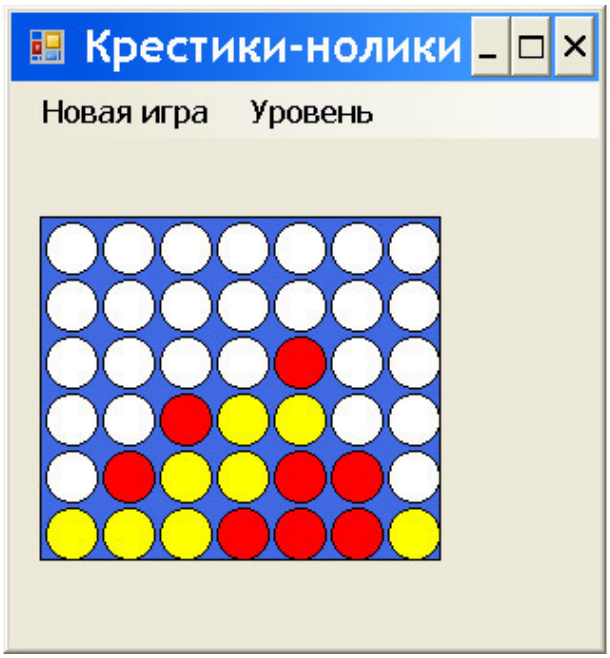
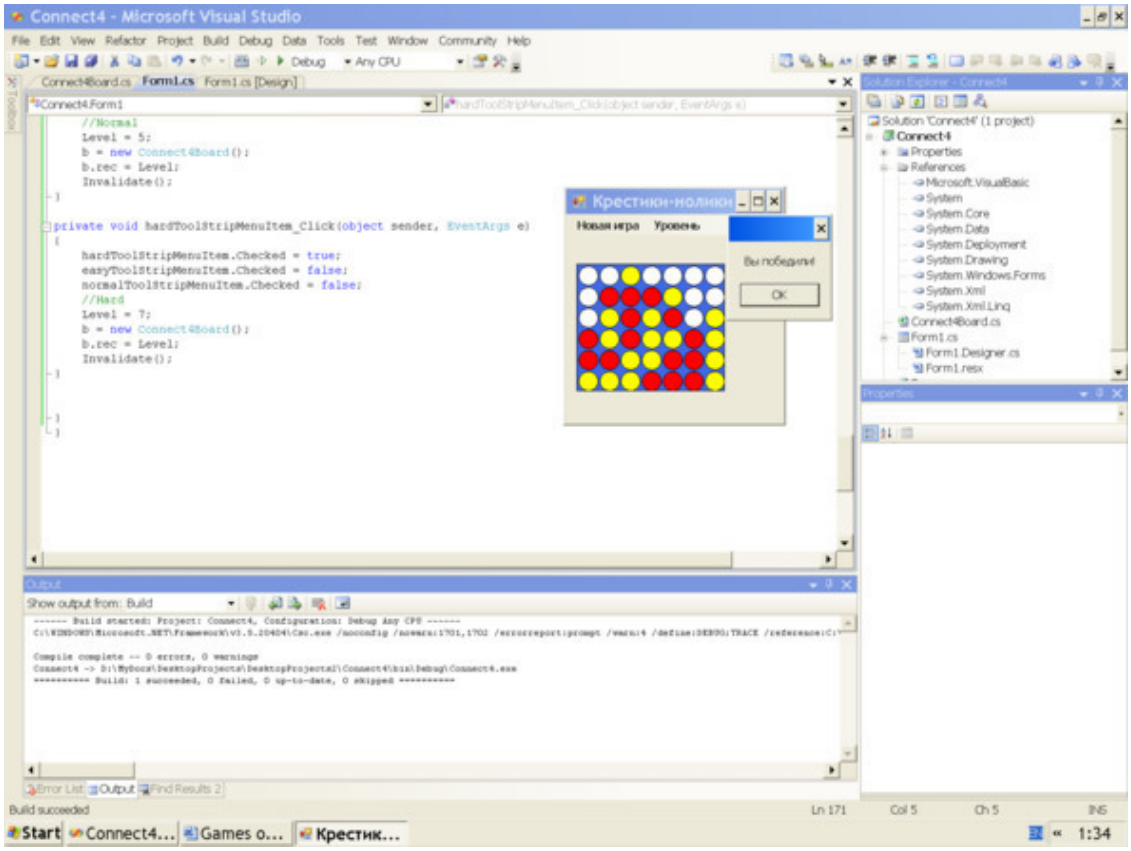


Рис. 12.3. Начало игры. **Рис. 12.4.** Середина игры.



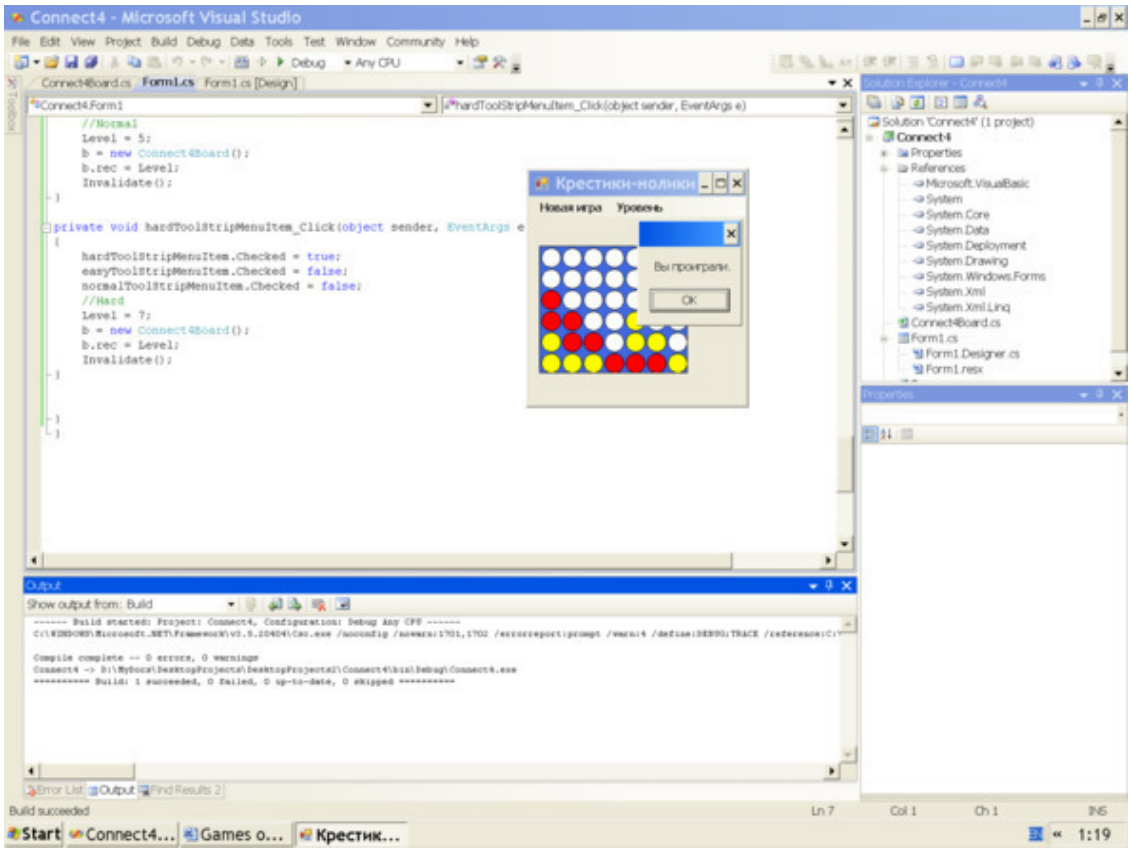
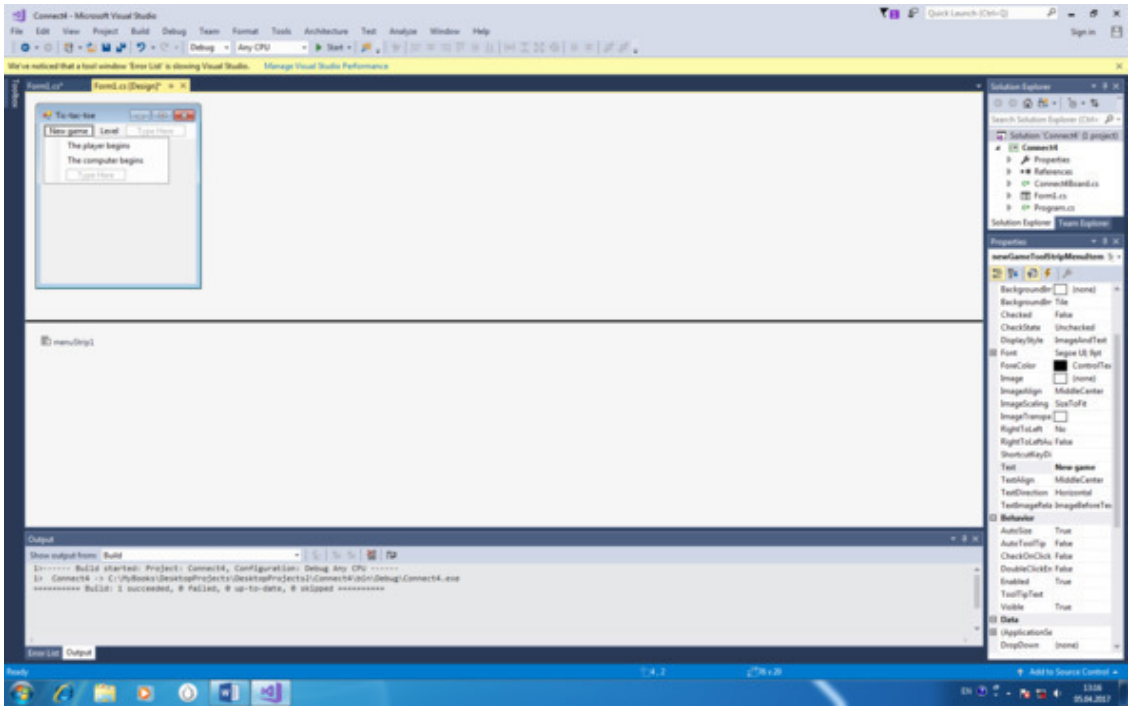


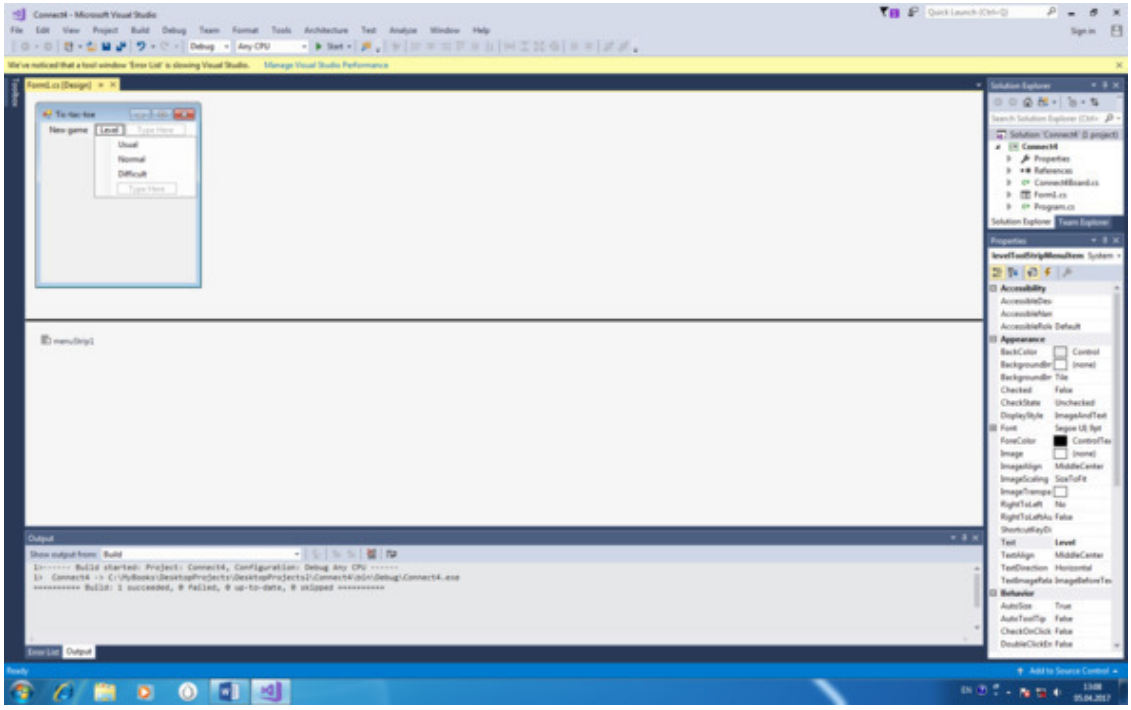
Рис. 12.5. Победил игрок. **Рис. 12.6.** Победил компьютер.

На основании этих правил можно сформулировать другие правила, и любые правила ввести в справочную форму игры, которую можно разработать по приведённым в других главах методикам.

12.3. Создание проекта

Создаём проект по обычной схеме: в VS в панели New Project (показанной выше) слева на вкладке Templates выбираем шаблон (тип проекта) Visual C#, Windows Classic Desktop и проверяем, чтобы в окне Templates был выделен (как правило, по умолчанию) шаблон Windows Forms App (.NET Framework); в окне Name записываем (или оставляем по умолчанию) имя проекта, например, Connect4 (рис. 12.7, 12.8, 12.9) и щёлкаем OK.





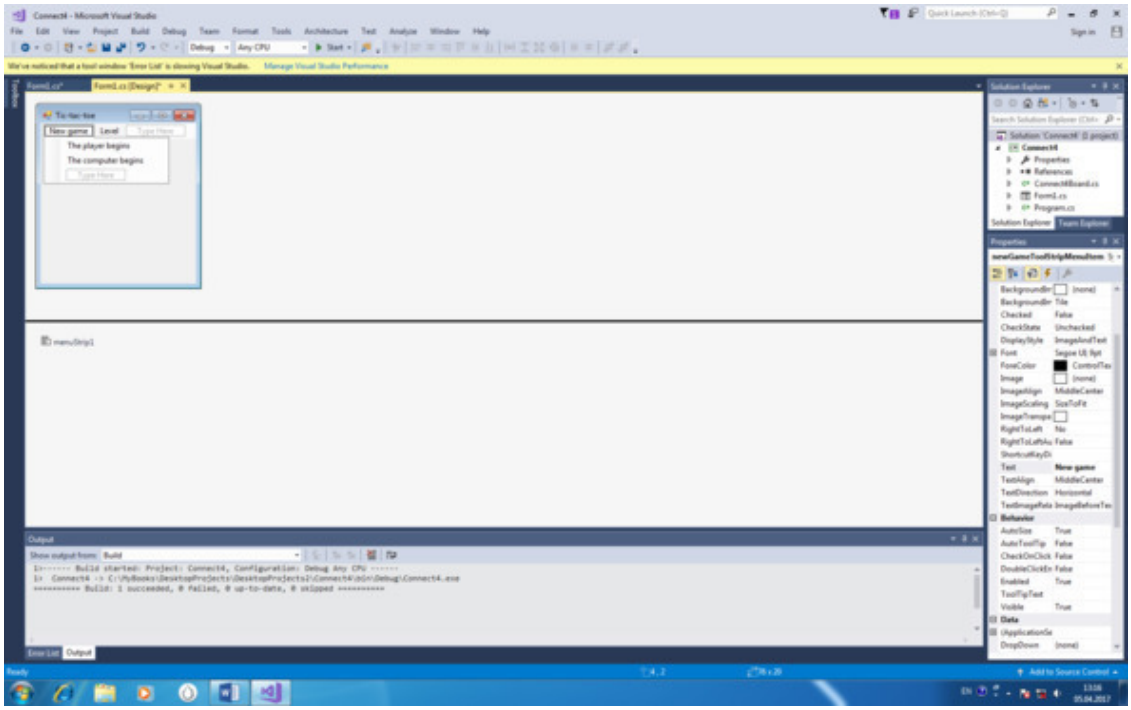


Рис. 12.7. Форма Form1. **Рис. 12.8.** Меню Level. **Рис. 12.9.** SE и Properties.

Создаётся проект, появляется форма Form1 на экране в режиме проектирования (рис. 12.7).

Оставляем по умолчанию или проектируем форму, как подробно описано выше и в наших книгах с сайта ZharkovPress.ru в параграфе «Методика проектирования формы».

Для задания режимов игры, как и выше, с панели инструментов Toolbox переносим на форму элемент управления MenuStrip (рис. 12.7).

На форме Form1 появляются окна с надписью Type Here (Печатайте здесь), в которые записываем команды, слева: New game (Новая игра), Begins the player (Начинает игрок), Begins the computer (Начинает компьютер), справа: Level (Уровень), Usual (Обычный), Normal (Нормальный), Difficult (Сложный), рис. 12.8. Если мы записали команды в виде русских слов, то теперь в панели Properties (для всех команд элемента управления MenuStrip) в свойстве Name вместо этих русских слов записываем английские слова, а именно, для команд слева: newGame, playerFirst, computerFirst, для команд справа: level, easy, normal, hard.

Если мы хотим подавать звуковой сигнал Beep в различные моменты выполнения каких-либо действий, например, после каждого щелчка экрана мышью, то поступаем следующим образом. По 1-му варианту, согласно разработанной выше методике использования в нашем приложении метода (функции) из любого другого языка, на первом этапе необходимо создать ссылку на тот язык, например, на Visual Basic. Для этого в меню Project выбираем команду Add Reference, в панели Reference Manager выбираем ссылку на соответствующую динамически подключаемую библиотеку (dynamic link library) формата (.dll), в данном случае, на Microsoft.VisualBasic и щёлкаем кнопку OK. А в тех местах кода, где нам нужен этот сигнал, например, в метод-обработчик Form1_MouseDown нажатия левой кнопки мыши, иначе, щелчка мышью по экрану, запишем далее строку:

```
Microsoft.VisualBasic.Interaction.Beep ();
```

По 2-му варианту, в тех местах кода, где нам нужен этот сигнал Beep, например, в обработчик события в виде нажатия клавиши Enter, записываем следующую строку:

```
System.Media.SystemSounds.Beep.Play ();
```

12.4. Код программы

Открываем файл Form1.cs (например, по схеме: File, Open, File) и в классе Form1 записываем следующие переменные и метод.

Листинг 12.1. Переменные и метод.

```
public int cx = 15;
public int cy = 70;
public int Level = 3;
public Connect4Board b = new Connect4Board ();

public void Draw (Graphics g)
{
Pen p = new Pen (Color. Black);
g. DrawRectangle (p, 15, 70, 210, 180);
int q = 0;
int w = 5;
for (int y = cy; y <cy +180; y += 30)
{
for (int x = cx; x <cx +210; x += 30)
{
if (b.arr [q, w] == 0)
g.FillEllipse (new SolidBrush (Color. White),
x +3, y +3, 27, 27);
if (b.arr [q, w] == 1)
g.FillEllipse (new SolidBrush (Color. Yellow),
x +3, y +3, 27, 27);
if (b.arr [q, w] == 2)
g.FillEllipse (new SolidBrush(Color.Red),
x +3, y +3, 27, 27);
g. DrawEllipse (p, x +3, y +3, 27, 27);
q++;
}
q = 0;
w – ;

}
}
```

В панели Properties (для Form1) на вкладке Events дважды щёлкаем по имени события Paint. Появившийся шаблон метода Form1_Paint после записи нашего кода принимает следующий вид. Напомним, что другие варианты вывода изображения, например, на элемент управления PictureBox и после щелчка по какому-либо элементу управления уже приводились ранее.

Листинг 12.2. Метод для рисования изображений.

```
private void Form1_Paint (object sender, PaintEventArgs e)
{
```

```
            e.Graphics.FillRectangle (new SolidBrush(Color.RoyalBlue),
            15, 70, 210, 180);
            Draw(e.Graphics);
}
```

В панели Properties (для Form1) на вкладке Events дважды щёлкаем по имени события MouseDown. Появившийся шаблон метода Form1_MouseDown после записи нашего кода принимает следующий вид.

Листинг 12.3. Метод для обработки щелчка экрана мышью.

```
            private void Form1_MouseDown (object sender, MouseEventArgs e)
            {
            Microsoft.VisualBasic.Interaction.Beep ();
            if (b. endt == 0)
            {
            if ((e.X> cx) && (e.X <cx +210))
            {
            if ((e.Y> cy) && (e.Y <cy +180))
            {
            int selection = (e.X – cx) / 30;
            if ((selection> = 0) && (selection <7))
            {
            if (b.add (selection, b.plr) == 0)
            {
            Draw(this.CreateGraphics ());
            b. turncheck ();
            b. turn = b. cpu;
            if (b. endt == 0)
            {
            int ck;
            b.col = b.Think ();
            ck = b.add(b.col, b. cpu);
            if (ck == 0)
            {
            b.lin = 6 – b.tops[b.col];
            b. turn = b.plr;
            }
            }
            Draw(this.CreateGraphics ());
            b. turncheck ();

}
}
}
}
}
}
```

Дважды щёлкаем по команде Begins the player (Начинает игрок) для элемента управления MenuStrip (или в панели Properties на вкладке Events выбираем событие Click). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

Листинг 12.4. Метод-обработчик выбора команды.

```
private void playerFirstToolStripMenuItem_Click (
object sender, EventArgs e)
{
//Player First
b = new Connect4Board ();
b.rec = Level;
Invalidate ();

}
```

Дважды щёлкаем по команде Begins the computer (Начинает компьютер) для элемента управления MenuStrip. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

Листинг 12.5. Метод-обработчик выбора команды.

```
private void computerFirstToolStripMenuItem_Click (
object sender, EventArgs e)
{
//CPU First
b = new Connect4Board ();
b.rec = Level;
Draw(this.CreateGraphics ());
b. turn = b. cpu;
if (b. endt == 0)
{
int ck;
b.col = b.Think ();
ck = b.add(b.col, b. cpu);
if (ck == 0)
{
b.lin = 6 – b.tops[b.col];
b. turn = b.plr;
}
}
Draw(this.CreateGraphics ());
b. turncheck ();

}
```

Дважды щёлкаем по команде Usual (Обычный). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

Листинг 12.6. Метод-обработчик выбора команды.

```
private void easyToolStripMenuItem_Click (
object sender, EventArgs e)
{
easyToolStripMenuItem.Checked = true;
hardToolStripMenuItem.Checked = false;
normalToolStripMenuItem.Checked = false;
//Easy
Level = 3;
b = new Connect4Board ();
b.rec = Level;
Invalidate ();

}
```

Дважды щёлкаем по команде Normal (Нормальный). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

Листинг 12.7. Метод-обработчик выбора команды.

```
private void normalToolStripMenuItem_Click (
object sender, EventArgs e)
{
hardToolStripMenuItem.Checked = false;
easyToolStripMenuItem.Checked = false;
normalToolStripMenuItem.Checked = true;
//Normal
Level = 5;
b = new Connect4Board ();
b.rec = Level;
Invalidate ();

}
```

Дважды щёлкаем по команде Difficult (Сложный). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

Листинг 12.8. Метод-обработчик выбора команды.

```
private void hardToolStripMenuItem_Click (
object sender, EventArgs e)
{
hardToolStripMenuItem.Checked = true;
easyToolStripMenuItem.Checked = false;
normalToolStripMenuItem.Checked = false;
//Hard
Level = 7;
b = new Connect4Board ();
```

```
b.rec = Level;
Invalidate ();

}
```

Схема записи и вывода справочной информации, например, с правилами игры после выбора дополнительной команды Справка (для элемента управления MenuStrip) дана ранее и в наших книгах на сайте ZharkovPress.ru в параграфе «Методика добавления информации в справочную форму».

Мы закончили написание программы в главный класс Form1 (для формы Form1 с пользовательским интерфейсом игры).

Теперь в наш проект добавляем новый файл (для программирования соответствующих игровых действий).

Добавить в проект файл можно по двум вариантам.

По первому варианту, добавляем в проект нужный файл по обычной схеме: в панели Solution Explorer выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, Existing Item, в панели Add Existing Item в окне «Files of type» выбираем «All Files», в центральном окне находим (например, в папке компьютера файл, скопированный из Интернета), выделяем имя этого файла и щёлкаем кнопку Add (или дважды щёлкаем по имени этого файла).

По второму варианту, в панели Solution Explorer выполняем правый щелчок по имени проекта и в контекстном меню выбираем Add, New Item, в панели Add New Item выделяем шаблон Code File, в окне Name записываем имя Connect4Board. cs и щёлкаем кнопку Add. В проект (и в панель Solution Explorer) добавляется этот файл, открывается пустое окно редактирования кода, в которое записываем код со следующего листинга.

Листинг 12.9. Новый файл.

```
using System;
using System. Drawing;
using System.Collections;
using System.Windows.Forms;

namespace Connect4
{
public class Connect4Board
{
public Random random = new Random ();
public int [,] arr = new int [7, 6];
public int [,] thn = new int [7, 6];
public int [] tops = new int [7];
public int player, computer, endt = 0;
public int plr = 1, cpu = 2, rec = 3, turn = 1;
public int m, n, r, temp, so, ch, col, t, y;
public int plrcoin, cpucoin;
public int lin;
public Connect4Board ()
{
}
```

```
public int Think ()
{
int i;
i = rec;
return check (i, -9999999, 9999999);
}
public void turncheck ()
{
int temp;
char [] toto = new char [20];
temp = checkwin ();
if (temp == plr)
{
MessageBox.Show («You won!»);
endt = 1; return;
}
if (temp == cpu)
{
MessageBox.Show («You lost.»);
endt = 1; return;
}
if (temp == 0)
{
for (t = 0; t <= 6; t++)
if (tops [t] <6) temp = 1;
if (temp == 0)
{
//drawn ();
endt = 1;
return;
}
}
}
public int check (int i, int alpha, int beta)
{
int co, score, t, g, j = 0, p;
i – ;
if (i == -1) {score = position (); return score;}
if (i % 2 == 0)
{
int max = 0, k;
j = 0; co = 0;
for (t = 0; t <7; t++)
{
g = add (t, cpu);
if (g == 0)
{
if (checkwin () == cpu)
{
```

```
sub (t);
if (i == rec – 1)
return t;
else return 9000;
}
k = check (i, alpha, 999999);
if (k> alpha) alpha = k;
sub (t);
if (k> beta) return k;
if (co == 0) {max = k; co = 1; j = t;}
if (k == max)
{
p = (random.Next (6)) +1;
if (p> 4) j = t;
}
if (k> max) {max = k; j = t;}
}
}
score = max;
}
else
{
int min = 0, k = 0;
co = 0;
for (t = 0; t <7; t++)
{
g = add (t, plr);
if (g == 0)
{
if (checkwin () == plr)
{
sub (t);
/*if (i==rec-1) return t; else*/
return -10000;
}
k = check (i, -99999, beta);
if (k <beta) beta = k;
sub (t);
if (k <alpha) return k;
if (co == 0) {min = k; co = 1; j = t;}
if (k <min) {min = k; j = t;}
}
}
score = min;
}
if (i == rec – 1) return j;
return score;
}
public int add (int c, int coin)
```

```
{
if (tops [c] <6)
{
arr [c, tops [c]] = coin;
tops [c] ++; return 0;
}
return 1;
}
public int sub (int c)
{
tops [c] – ;
arr [c, tops [c]] = 0;
return 0;
}
public int position ()
{
int u, o, x, y, j, score;
int gh = 0, hg = 0;
score = 0;
//Empty the think array

for (x = 0; x <7; x++)
{
for (y = 0; y <6; y++)
{
thn [x, y] = 0;
}
}
//Sum the score of every opportunity to the score
for (y = 0; y <6; y++)
{
for (x = 0; x <7; x++)
{
if (arr [x, y] == 0)
score = score + checkhole (x, y);
if (y> 0)
{
if ((thn [x, y] == cpu) &&
(arr [x, y – 1]!= 0)) gh++;
if ((thn [x, y] == plr) &&
(arr [x, y – 1]!= 0))
{hg++; score = score – 4000;}
}
else
{
if (thn [x, y] == cpu) gh++;
if (thn [x, y] == plr)
{hg++; score = score – 4000;}
}
```

```
}
}
if (gh> 1) score = score + (gh – 1) * 500;
if (gh == 1) score = score – 100;
if (hg> 1) score = score – (hg – 1) * 500;
for (x = 0; x <7; x++)
{
gh = 0;
for (y = 1; y <6; y++)
{
/*if (gh==0)
if ((thn [x,y]> 0) && (arr [x,y-1] ==0)) {
gh=1;

} */
if ((thn [x, y] == cpu) &&
(thn [x, y – 1] == cpu))
{
u = 0; j = 0;
for (o = y – 1; o> -1; o – )
{
if (thn [x, o] == plr) u = 1;
if (arr [x, o] == 0) j++;
}
if (u == 0) score = score +1300 – j * 7;
if (u == 1) score = score +300;
}
if ((thn [x, y] == plr) &&
(thn [x, y – 1] == plr))
{
u = 0; j = 0;
for (o = y – 1; o> -1; o – )
{
if (thn [x, o] == cpu) u = 1;
if (arr [x, o] == 0) j++;
}
if (u == 0) score = score – 1500 + j * 7;
if (u == 1) score = score – 300;
}
if (thn [x, y] == plr)
{
u = 0;
for (o = y – 1; o> -1; o – )
{
if (thn [x, o] == cpu) u = 1;
}
if (u == 1) score = score +30;
}
if (thn [x, y] == cpu)
```

```
{
u = 0;
for (o = y – 1; o> -1; o – )
{
if (thn [x, o] == plr) u = 1;
}
if (u == 1) score = score – 30;
}
}
}
return score;
}
public int checkhole (int x, int y)
{
int score = 0;
int max, min;
int d0 = 0, d1 = 0, d2 = 0, d3 = 0;
if (((x +1) <7) && ((y – 1)> -1))
{
if (arr [x +1, y – 1] == cpu)
{
d1++;
if (((x +2) <7) && ((y – 2)> -1))
{
if (arr [x +2, y – 2] == cpu)
{
d1++;
if (((x +3) <7) && ((y – 3)> -1))
{
if (arr [x +3, y – 3] == cpu)
d1++;
}
}
}
}
}
if (((x – 1)> -1) && ((y +1) <6))
{
if (arr [x – 1, y +1] == cpu)
{
d1++;
if (((x – 2)> -1) && ((y +2) <6))
{
if (arr [x – 2, y +2] == cpu)
{
d1++;
if (((x – 3)> -1) && ((y +3) <6))
{
if (arr [x – 3, y +3] == cpu) d1++;
```

```
}
}
}
}
}
if (((x – 1)> -1) && ((y – 1)> -1))
{
if (arr [x – 1, y – 1] == cpu)
{
d2++;
if (((x – 2)> -1) && ((y – 2)> -1))
{
if (arr [x – 2, y – 2] == cpu)
{
d2++;
if (((x – 3)> -1) && ((y – 3)> -1))
{
if (arr [x – 3, y – 3] == cpu) d2++;
}
}
}
}
}
if (((x +1) <7) && ((y +1) <6))
{
if (arr [x +1, y +1] == cpu)
{
d2++;
if (((x +2) <7) && ((y +2) <6))
{
if (arr [x +2, y +2] == cpu)
{
d2++;
if (((x +3) <7) && ((y +3) <6))
{
if (arr [x +3, y +3] == cpu) d2++;
}
}
}
}
}
if ((y – 1)> -1) if (arr [x, y – 1] == cpu)
{
d0++;
if ((y – 2)> -1) if (arr [x, y – 2] == cpu)
{
d0++;
if ((y – 3)> -1)
if (arr [x, y – 3] == cpu) d0++;
```

```
}
}
if (x – 1> -1)
{
if (arr [x – 1, y] == cpu)
{
d3++;
if (x – 2> -1)
{
if (arr [x – 2, y] == cpu)
{
d3++;
if (x – 3> -1)
if (arr [x – 3, y] == cpu) d3++;
}
}
}
}
if (x +1 <7)
{
if (arr [x +1, y] == cpu)
{
d3++;
if (x +2 <7)
{
if (arr [x +2, y] == cpu)
{
d3++;
if (x +3 <7)
if (arr [x +3, y] == cpu) d3++;
}
}
}
}
max = d0;
if (d1> max) max = d1;
if (d2> max) max = d2;
if (d3> max) max = d3;
if (max == 2) score = score +5;
if (max> 2)
{
score = score +71; thn [x, y] = cpu;
if ((d1 <3) && (d2 <3) && (d3 <3))
score = score – 10;
}
if (((x +1) <7) && ((y – 1)> -1))
{
if (arr [x +1, y – 1] == plr)
{
```

```
d1++;
if (((x +2) <7) && ((y – 2)> -1))
{
if (arr [x +2, y – 2] == plr)
{
d1++;
if (((x +3) <7) && ((y – 3)> -1))
{
if (arr [x +3, y – 3] == plr) d1++;
}
}
}
}
}
if (((x – 1)> -1) && ((y +1) <6))
{
if (arr [x – 1, y +1] == plr)
{
d1++;
if (((x – 2)> -1) && ((y +2) <6))
{
if (arr [x – 2, y +2] == plr)
{
d1++;
if (((x – 3)> -1) && ((y +3) <6))
{
if (arr [x – 3, y +3] == plr) d1++;
}
}
}
}
}
if (((x – 1)> -1) && ((y – 1)> -1))
{
if (arr [x – 1, y – 1] == plr)
{
d2++;
if (((x – 2)> -1) && ((y – 2)> -1))
{
if (arr [x – 2, y – 2] == plr)
{
d2++;
if (((x – 3)> -1) && ((y – 3)> -1))
{
if (arr [x – 3, y – 3] == plr) d2++;
}
}
}
}
```

```
}
if (((x +1) <7) && ((y +1) <6))
{
if (arr [x +1, y +1] == plr)
{
d2++;
if (((x +2) <7) && ((y +2) <6))
{
if (arr [x +2, y +2] == plr)
{
d2++;
if (((x +3) <7) && ((y +3) <6))
{
if (arr [x +3, y +3] == plr) d2++;
}
}
}
}
}
if ((y – 1)> -1) if (arr [x, y – 1] == plr)
{
d0++;
if ((y – 2)> -1) if (arr [x, y – 2] == plr)
{
d0++;
if ((y – 3)> -1)
if (arr [x, y – 3] == plr) d0++;
}
}
if (x – 1> -1)
{
if (arr [x – 1, y] == plr)
{
d3++;
if (x – 2> -1)
{
if (arr [x – 2, y] == plr)
{
d3++;
if (x – 3> -1)
if (arr [x – 3, y] == plr) d3++;
}
}
}
}
if (x +1 <7)
{
if (arr [x +1, y] == plr)
{
```

```
d3++;
if (x +2 <7)
{
if (arr [x +2, y] == plr)
{
d3++;
if (x +3 <7)
if (arr [x +3, y] == plr) d3++;
}
}
}
}
min = d0;
if (d1> min) min = d1;
if (d2> min) min = d2;
if (d3> min) min = d3;
if (min == 2) score = score – 4;
if (min> 2)
{
score = score – 70; thn [x, y] = plr;
if ((d1 <3) && (d2 <3) && (d3 <3))
score = score +10;
}
return score;
}
public int checkwin ()
{
int r, x, y;
r = 0;
for (y = 2; y> -1; y – )
{
for (x = 0; x <7; x++)
{
checku (x, y, ref r);
}
}
for (y = 0; y <6; y++)
{
for (x = 0; x <4; x++)
{
check2r (x, y, ref r);
}
}
for (y = 2; y> -1; y – )
{
for (x = 0; x <4; x++)
{
checkr (x, y, ref r);
}
```

```
}
for (y = 2; y> -1; y – )
{
for (x = 3; x <7; x++)
{
checkl (x, y, ref r);
}
}
return r;
}
public void checku (int x, int y, ref int r)
{
if ((arr [x, y] == 2) && (arr [x, y +1] == 2) &&
(arr [x, y +2] == 2) &&
(arr [x, y +3] == 2)) r = 2;
if ((arr [x, y] == 1) && (arr [x, y +1] == 1) &&
(arr [x, y +2] == 1) &&
(arr [x, y +3] == 1)) r = 1;
}
public void check2r (int x, int y, ref int r)
{
if ((arr [x, y] == 2) && (arr [x +1, y] == 2) &&
(arr [x +2, y] == 2) &&
(arr [x +3, y] == 2)) r = 2;
if ((arr [x, y] == 1) && (arr [x +1, y] == 1) &&
(arr [x +2, y] == 1) &&
(arr [x +3, y] == 1)) r = 1;
}
public void checkr (int x, int y, ref int r)
{
if ((arr [x, y] == 2) && (arr [x +1, y +1] == 2) &&
(arr [x +2, y +2] == 2) &&
(arr [x +3, y +3] == 2)) r = 2;
if ((arr [x, y] == 1) && (arr [x +1, y +1] == 1) &&
(arr [x +2, y +2] == 1) &&
(arr [x +3, y +3] == 1)) r = 1;
}
public void checkl (int x, int y, ref int r)
{
if ((arr [x, y] == 2) && (arr [x – 1, y +1] == 2) &&
(arr [x – 2, y +2] == 2) &&
(arr [x – 3, y +3] == 2)) r = 2;
if ((arr [x, y] == 1) && (arr [x – 1, y +1] == 1) &&
(arr [x – 2, y +2] == 1) && (
arr [x – 3, y +3] == 1)) r = 1;

}
}
}
```

После этого добавления в панели Solution Explorer появляется этот файл, как показано выше. Дважды щёлкая в SE по имени файла, любой файл можно открыть, изучить и редактировать.

12.5. Запуск игры

Строим и запускаем программу на выполнение обычным образом:

Build, Build Selection; Debug, Start Without Debugging.

В ответ Visual C# выводит форму Form1 с показанным выше полем игры из кружочков (ноликов).

Далее мы играем с компьютером в крестики-нолики согласно приведённым выше правилам.

По методике данной главы можно разрабатывать самые разнообразные игры в крестики-нолики, однако при следующем условии.

Часть IV. Методология программирования искусственного интеллекта в спортивных играх на примере игры в теннис для Игрока с Компьютером или двух Игроков

Глава 13. Методика программирования искусственного интеллекта в игре в теннис на основе элементов управления на одной форме

13.1. Общие сведения

Разработаем методику проектирования и программирования типичной и широко распространённой игры в теннис с мячом и двумя ракетками. Если в предыдущей книге (с сайта ZharkovPress.ru) игру в теннис мы разработали на основе графических файлов, то в данной главе эту же игру разработаем на основе стандартных элементов управления с панели инструментов Toolbox, следуя проекту [см. Список литературы: Game Tiny Tennis from the website of Microsoft corporation], но с нашими усовершенствованиями для современной версии Visual Studio.

Кратко, сюжет игры заключается в следующем. После старта игры появляется форма Form1, слева на форме (точнее, в клиенткой области формы) находится ракетка Игрока 1, справа – ракетка Игрока 2 (по первому режиму игры) или Компьютера в роли Игрока 2 (по второму режиму игры). Мяч произвольным образом прыгает в пределах 4-х границ формы, отскакивая от верхней и нижней границ формы и двух ракеток. Левая граница формы – это ворота Игрока 1, а правая граница формы – это ворота Компьютера в роли Игрока 2 или самого Игрока 2. Если мяч коснётся левой или правой границ формы, считается, что один игрок забил мяч в ворота другого игрока, счёт увеличивается в пользу забившего мяч игрока, а мяч снова появляется в произвольной точке формы и летит в произвольном направлении, заданном генератором случайных числе класса Random.

В задачу пользователя в роли Игрока 2 (по первому режиму игры, заданному по умолчанию) входит – при помощи клавиш со стрелками перемещать правую ракетку по вертикали снизу вверх и сверху вниз таким образом, чтобы отбить мяч и не дать мячу коснуться правой границы формы, так как после каждого такого касания (пропущенного в ворота мяча) Игроку 1 начисляется 1 очко. Пользователь в роли Игрока 2 должен стараться (по возможности) ракеткой отбить мяч таким образом, чтобы забить его в ворота Игрока 1.

Левая ракетка Игрока 1 старается отбить мяч и не дать мячу коснуться левой границы формы, так как после каждого такого касания (пропущенного в ворота мяча) Компьютеру в роли Игрока 2 начисляется 1 очко (как забившему мяч).

Таким образом, для заданного по умолчанию первого режима игры (Игрок 1 против пользователя в роли Игрока 2), в методе-конструкторе Form1 используется строка:

```
_player2 = new Bat (pictureBox2, ClientSize. Width – 30 —
Bat. Width, Keys. Up, Keys.Down, 0, ClientSize. Height);
```

Видно, что пользователь (Игрок 2) перемещает ракетку вверх при помощи клавиши Keys. Up, а вниз – Keys.Down.

По второму режиму игры (Игрок 1 против Компьютера в роли Игрока 2), в методе-конструкторе Form1 комментируем приведённую выше строку и раскомментируем строку:

```
_player2 = new Bat (pictureBox2, ClientSize. Width – 30 —
Bat. Width, 0, ClientSize. Height);
```

Теперь Компьютер (в роли Игрока 2) будет перемещать правую ракетку по вертикали снизу вверх и сверху вниз и отбивать ракеткой мяч, чтобы не пропустить его мимо ракетки в правые ворота.

Победителем считается игрок, который за отведённое на игру время большее число раз поразил ворота соперника.

В данной игре также была поставлена и решена математическая задача о перемещении объекта в замкнутом пространстве, сформулированная в предыдущей главе с большим количеством допущений.

13.2. Правила игры

1. После запуска игры появляется форма Form1, на клиентской области которой слева находится ракетка Игрока 1, справа – ракетка Игрока 2 (по первому режиму игры) или Компьютера в роли Игрока 2 (по второму режиму игры). Вверху находится табло для подсчёта очков (в виде двух элементов управления Label); в начале игры счёт 0:0, рис. 13.1.

Две ракетки и мяч разработаны на основе элемента управления PictureBox.

Мяч произвольным образом прыгает в пределах 4-х границ формы, отскакивая от границ формы и двух ракеток. После каждого такого отскока мы слышим звук добавленного в проект файла beep. wav. Левая граница формы – это ворота Игрока 1, а правая граница формы – это ворота Игрока 2 (или Компьютера в роли Игрока 2).

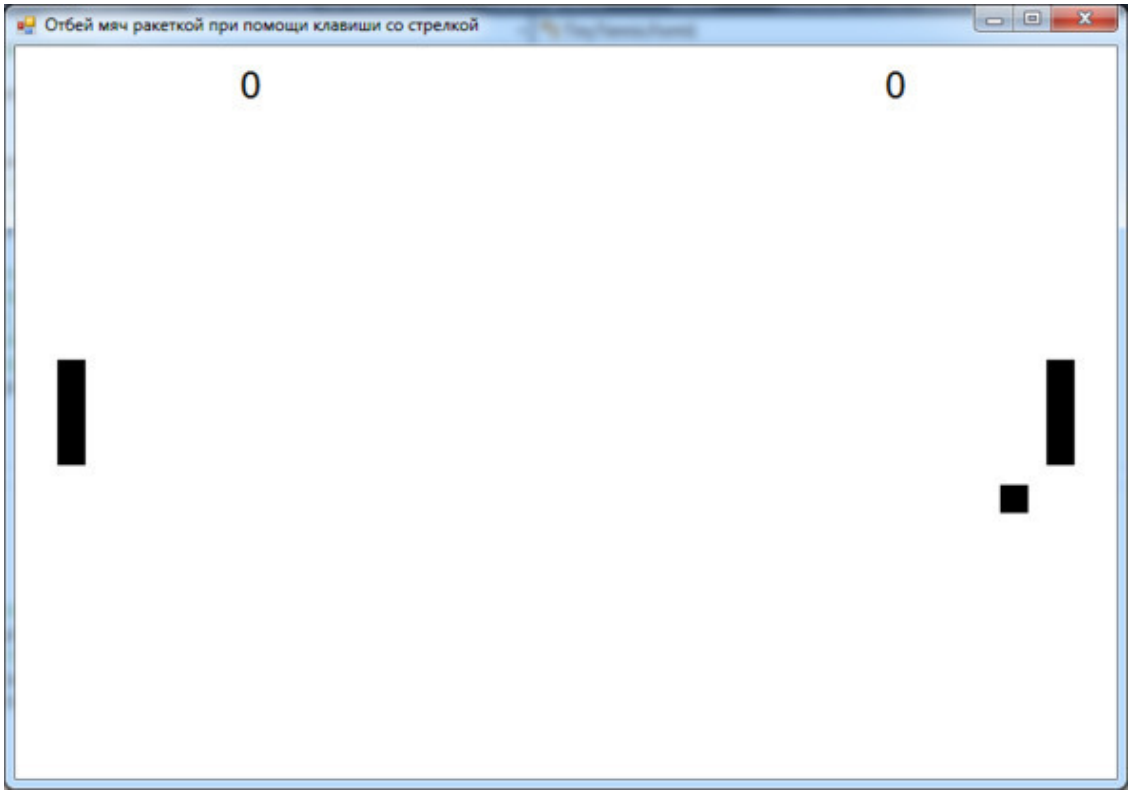


Рис. 13.1. Перед началом игры (слева – ракетка Игрока 1, справа – ракетка Игрока 2).

2. В задачу пользователя в роли Игрока 2 (по первому режиму игры, заданному по умолчанию) входит – перемещать правую ракетку по вертикали снизу вверх и сверху вниз таким образом, чтобы отбить мяч и не дать мячу коснуться правой границы формы, так как после каждого такого касания (пропущенного в ворота мяча) Игроку 1 начисляется 1 очко за забитый гол. И Игрок 2, и Компьютер в роли Игрока 2 стараются ракеткой отбить мяч таким образом, чтобы забить его в ворота Игрока 1.

Левая ракетка Игрока 1 старается отбить мяч и не дать мячу коснуться левой границы формы, так как после каждого такого касания (пропущенного в ворота мяча) пользователю в роли Игрока 2 (или Компьютеру в роли Игрока 2) начисляется 1 очко за забитый гол.

На рис. 13.2 показан счёт 4:11 в пользу Игрока 2.

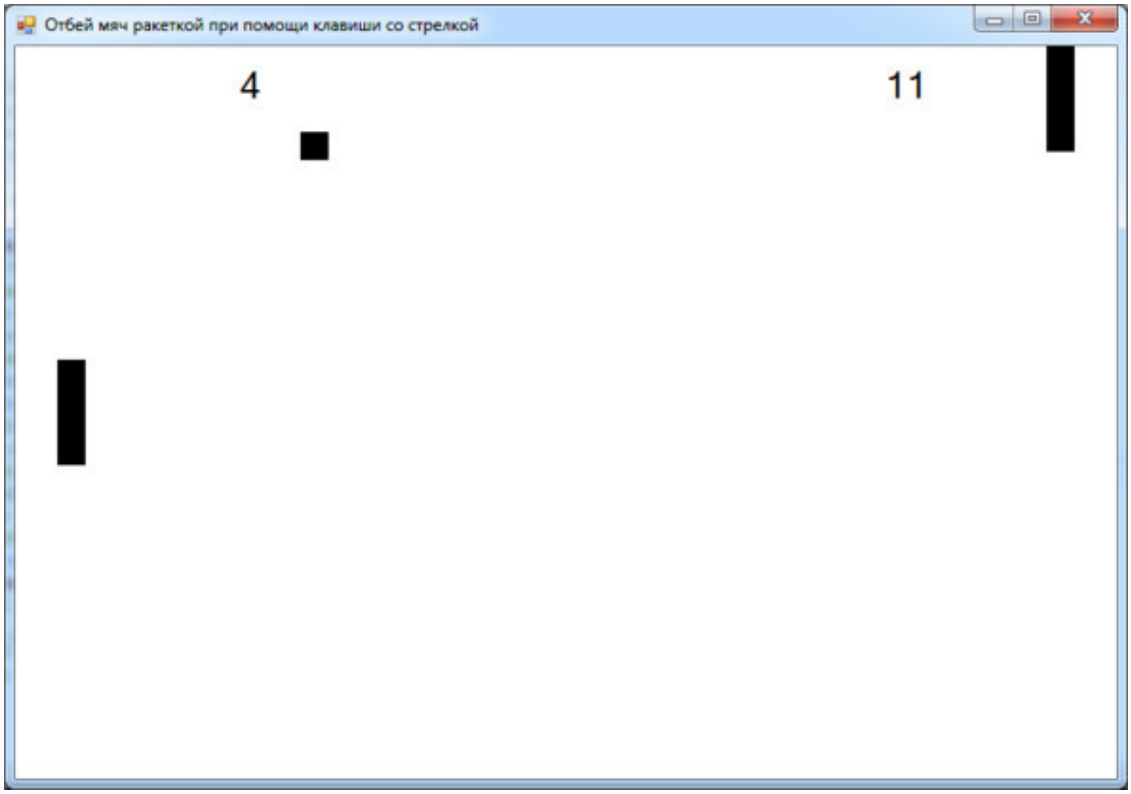


Рис. 13.2. Игра идёт, счёт 4:11 (слева – ракетка Игрока 1, справа – ракетка Игрока 2).

3. Победителем считается игрок, который за отведённое на игру время большее число раз поразил ворота соперника. Как правило, Компьютер (в роли Игрока 2) побеждает.

4. Если в игре участвует несколько человек, то победителем считается тот, у кого лучшая разница между забитыми и пропущенными мячами.

5. Форму с игрой закрываем стандартно, щёлкая на ней значок Close.

На основании этих правил можно сформулировать другие правила, и любые правила ввести в справочную форму, которую можно разработать по описанной ранее методике.

И, естественно, в качестве графических изображений объектов и звуковых файлов игры можно использовать различные элементы управления и файлы различных форматов (с внесением соответствующих изменений в приведённую далее программу).

13.3. Создание проекта

Создаём проект по обычной схеме: в VS в панели New Project (показанной выше) слева на вкладке Templates выбираем шаблон (тип проекта) Visual C#, Windows Classic Desktop и проверяем, чтобы в окне Templates был выделен (как правило, по умолчанию) шаблон Windows Forms App (.NET Framework); в окне Name записываем (или оставляем по умолчанию) имя проекта, например, TinyTennis и щёлкаем OK.

Создаётся проект, появляется форма Form1 в режиме проектирования (рис. 13.3 и рис. 13.4). Оставляем по умолчанию или проектируем форму, как подробно описано ранее в параграфе «Методика проектирования формы» в наших книгах с сайта ZharkovPress.ru. За маркеры увеличиваем размеры формы таким образом, чтобы в панели Properties в свойстве Size были, например, такие величины: 800; 559.

Конец ознакомительного фрагмента.