Доктор технических наук
Жарков Валерий Алексеевич

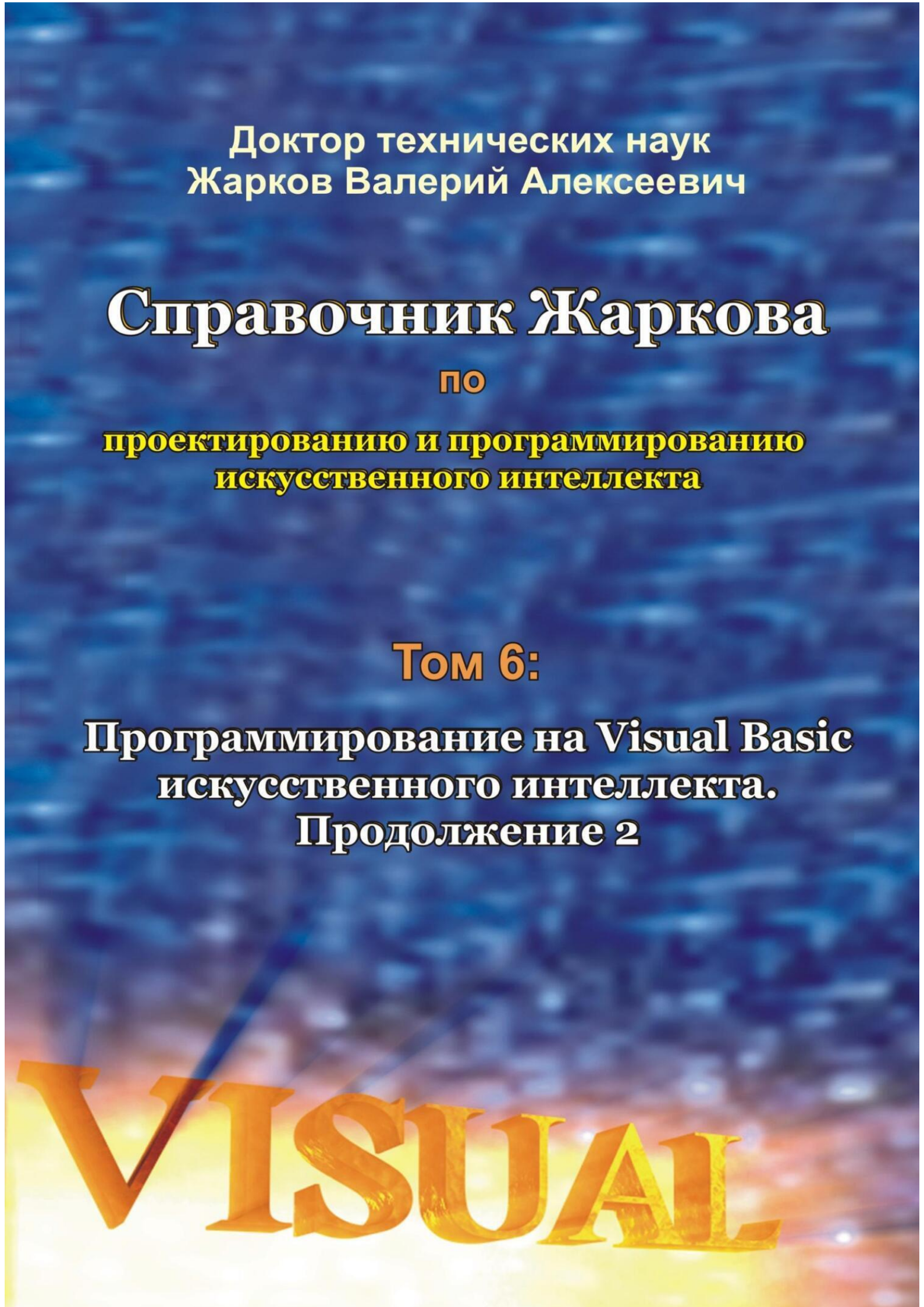
# Справочник Жаркова

по

## проектированию и программированию искусственного интеллекта

## Том 6:

## Программирование на Visual Basic искусственного интеллекта. Продолжение 2



Валерий Жарков

# Справочник Жаркова по проектированию и программированию искусственного интеллекта. Том 6: Программирование на Visual Basic искусственного интеллекта. Продолжение 2

«Автор»
2023

**Жарков В. А.**

Справочник Жаркова по проектированию и программированию искусственного интеллекта. Том 6: Программирование на Visual Basic искусственного интеллекта. Продолжение 2 / В. А. Жарков — «Автор», 2023

В серии книг “Справочник Жаркова по проектированию и программированию искусственного интеллекта” в нескольких томах собрано лучшее программирование искусственного интеллекта (ИИ) в двух- и трёхмерных играх и приложениях, разработанных как автором, так и взятые из Интернета за многие годы и доработанные автором. Программирование ИИ на Visual Basic написано в XVIII частях, которые разделены на три тома 4, 5 и 6. В данном томе 6 приведены части IX-XVIII. Методология программирования искусственного интеллекта: в играх по сборке и выбиванию фигур одинакового цвета или геометрии, в ролевых сюжетных играх, в играх с воздушными боями ракетами с участием самолётов и вертолётов, в спортивных играх, в играх в кости, в играх с летающими объектами, уничтожающимися после столкновения, в математических играх, в трёхмерных играх по управлению автомобилем при езде по дороге с препятствиями. Книги предназначены для желающих изучить программирование ИИ в играх и приложениях на базе VB и других языков.

# Содержание

# Валерий Жарков
# Справочник Жаркова по проектированию и программированию искусственного интеллекта. Том 6: Программирование на Visual Basic искусственного интеллекта. Продолжение 2

## 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, планшетов и смартфонов на основе популярного, совершенного и перспективного языка программирования высокого уровня Visual Basic 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 даёт 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 (заменяющим 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 Basic и давать подробные объяснения по теории языка VB. Если у читателя возникнуть вопросы, он легко отыщет книгу по данному языку (некоторые полезные по данной тематике книги и журналы 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 пояс-

нения в виде комментариев в программах, чтобы начинающий пользователь постепенно осваивал базовые дисциплины программирования ИИ на языке VB, по возможности не используя другие книги; опытному пользователю также будут полезны эти пояснения, поскольку книги по разработке ИИ на основе новых версий языка Visual Basic 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 принципу: делай, как я) самостоятельно программировать ИИ в практическом приложении или игре на основе 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 и журналов из списка литературы и с сайта ZharkovPress.ru (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 серии книг 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 всех XVIII частей из трёх томов 4, 5 и 6 по программированию ИИ на Visual Basic. Введение. Часть I. Краткие основы Visual Basic. Глава 1. Основные определения книги. Глава 2. Методика разработки приложений для выполнения расчётов с эффектами анимации. Глава 3. Методика разработки приложений на нескольких формах и передачи данных с одной формы на другую. Часть II. Учебная методология программирования игр и приложений с подвижными объектами. Глава 4. Методика анимации и управления подвижными объектами. Глава 5. Методика обнаружения столкновений, программирования уничтожений летающих объектов и подсчёта очков. Глава 6. Методология воспроизведения звуковых файлов. Глава 7. Методика программирования жизней, уровней сложности и вывода лучшего результата. Глава 8. Методика улучшения графики и добавления фона экрана. Глава 9. Методика программирования игры с летающими объектами на основе спрайтов. Глава 10. Игра с летающими объектами на основе спрайтов, двух форм и возможности приостановки и повторного запуска игры. Глава 11. Игра с изменяемой траекторией летающих объектов. Часть III. Методология программирования искусственного интеллекта в карточных играх. Глава 12. Методика программирования искусственного интеллекта в карточных играх на примере игры в покер. Часть IV. Методология программирования искусственного интеллекта в играх по сборке картины из её частей. Глава 13. Методика программирования искусственного интеллекта в игре по сборке разрезанной картины заменой местами её масштабируемых частей. Часть V. Методология программирования искусственного интеллекта в играх типа змейки, которая поедает куски пищи. Глава 14. Методика программирования искусственного интеллекта в игре типа змейки, которая поедает куски пищи и за счёт этого увеличивается по длине, на основе одного файла. Глава 15. Методика программирования искусственного интеллекта в игре типа змейки, которая поедает куски пищи и за счёт этого увеличивается по длине, на основе четырёх файлов. Часть VI. Методология программирования искусственного интеллекта

в играх типа “Тетрис” по сборке сплошных полос из разнообразных фигур. Глава 16. Методика программирования искусственного интеллекта в игре по сборке сплошных прямых полос из сторон фигур 12 типов. Глава 17. Методика программирования искусственного интеллекта в игре по сборке сплошных прямых полос из сторон фигур 7 типов с формами для имени игрока, результатов и справкой по игре. Часть VII. Методология программирования искусственного интеллекта в играх в “Крестики-нолики” для Игрока с Компьютером и двух Игроков. Глава 18. Методика программирования искусственного интеллекта в игре в “Крестики-нолики”. Часть VIII. Методология программирования искусственного интеллекта в играх типа “Поле чудес” по угадыванию слова по буквам. Глава 19. Методика программирования искусственного интеллекта в игре по угадыванию слова по буквам при заданном количестве попыток. Часть IX. Методология программирования искусственного интеллекта в играх по сборке и выбиванию фигур одинакового цвета или геометрии. Глава 20. Методика программирования искусственного интеллекта в игре по выбиванию фигур одинакового цвета. Глава 21. Методика программирования искусственного интеллекта в игре по сборке прямых из 5 и более объектов одинакового цвета. Часть X. Методология программирования искусственного интеллекта в ролевых сюжетных играх. Глава 22. Методика программирования искусственного интеллекта в сюжетных играх на примере сюжета о пещерных людях Адаме и Еве. Часть XI. Методология программирования искусственного интеллекта в играх с воздушными боями ракетами с участием самолётов и вертолётов.Глава 23. Методика программирования искусственного интеллекта в игре воздушного боя ракетами вертолёта с самолётами и вертолётами различных типов. Часть XII. Методология программирования искусственного интеллекта в спортивных играх. Глава 24. Методика программирования искусственного интеллекта в игре в теннис на основе элементов управления с уничтожением их после удара мячом. Часть XIII. Методология программирования искусственного интеллекта в играх в кости. Глава 25. Методика программирования искусственного интеллекта в игре в кости на примере игры с двумя кубиками. Часть XIV. Методология программирования искусственного интеллекта в играх с летающими объектами, уничтожающимися после столкновения. Глава 26. Методика программирования искусственного интеллекта в игре с генерированием летающих объектов, отскакивающих от границ и уничтожающихся после столкновения друг с другом. Часть XV. Методология программирования искусственного интеллекта в математических играх. Глава 27. Методика программирования искусственного интеллекта в игре на арифметические действия. Часть XVI. Методология программирования искусственного интеллекта в трёхмерных играх по управлению автомобилем при езде по дороге с препятствиями. Глава 28. Методика программирования искусственного интеллекта в игре по управлению автомобилем. Глава 29. Создание двух проектов игры. Глава 30. Запуск игры. Часть XVII. Проектирование вспомогательных объектов для игр и приложений с искусственным интеллектом. Глава 31. Методика проектирования цифровых часов. Часть XVIII. Развёртывание, публикация и распространение разработанной игры или приложения с искусственным интеллектом. Глава 32. Методика распространения игры или приложения. Заключение. Список литературы.

Приводим краткое содержание VIII-XVIII частей данного тома 6 по программированию ИИ на Visual Basic. Введение. Часть IX. Методология программирования искусственного интеллекта в играх по сборке и выбиванию фигур одинакового цвета или геометрии. Глава 20. Методика программирования искусственного интеллекта в игре по выбиванию фигур одинакового цвета. Глава 21. Методика программирования искусственного интеллекта в игре по сборке прямых из 5 и более объектов одинакового цвета. Часть X. Методология программирования искусственного интеллекта в ролевых сюжетных играх. Глава 22. Методика программирования искусственного интеллекта в сюжетных играх на примере сюжета о пещерных людях Адаме и Еве. Часть XI. Методология программирования искусственного интеллекта в играх с воздушными боями ракетами с участием самолётов и вертолётов. Глава 23. Методика програм-

мирования искусственного интеллекта в игре воздушного боя ракетами вертолёта с самолётами и вертолётами различных типов. Часть XII. Методология программирования искусственного интеллекта в спортивных играх. Глава 24. Методика программирования искусственного интеллекта в игре в теннис на основе элементов управления с уничтожением их после удара мячом. Часть XIII. Методология программирования искусственного интеллекта в играх в кости. Глава 25. Методика программирования искусственного интеллекта в игре в кости на примере игры с двумя кубиками. Часть XIV. Методология программирования искусственного интеллекта в играх с летающими объектами, уничтожающимися после столкновения. Глава 26. Методика программирования искусственного интеллекта в игре с генерированием летающих объектов, отскакивающих от границ и уничтожающихся после столкновения друг с другом. Часть XV. Методология программирования искусственного интеллекта в математических играх. Глава 27. Методика программирования искусственного интеллекта в игре на арифметические действия. Часть XVI. Методология программирования искусственного интеллекта в трёхмерных играх по управлению автомобилем при езде по дороге с препятствиями. Глава 28. Методика программирования искусственного интеллекта в игре по управлению автомобилем. Глава 29. Создание двух проектов игры. Глава 30. Запуск игры. Часть XVII. Проектирование вспомогательных объектов для игр и приложений с искусственным интеллектом. Глава 31. Методика проектирования цифровых часов. Часть XVIII. Развёртывание, публикация и распространение разработанной игры или приложения с искусственным интеллектом. Глава 32. Методика распространения игры или приложения. Заключение. Список литературы.

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 и журналов с сайта ZharkovPress.ru) 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 с использованием ИИ те результаты, которые необходимы именно им и характерны для их профессиональных или непрофессиональных интересов.

**Книга предназначена** для всех желающих быстро изучить основы программирования искусственного интеллекта в разнообразных двухмерных и трёхмерных компьютерных играх и приложениях на базе популярного, совершенного и перспективного (в мире программирования) языка высокого уровня **Visual Basic** последних версий для настольных компьютеров, ноутбуков, планшетов и смартфонов, на этих основах сразу же проектировать ИИ в сложных играх и приложениях и применять ИИ 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 наших книг и журналов можно направлять по email с сайта ZharkovPress.ru.

# Часть IX. Методология программирования искусственного интеллекта в играх по сборке и выбиванию фигур одинакового цвета или геометрии

# Глава 20. Методика программирования искусственного интеллекта в игре по выбиванию фигур одинакового цвета

# 20.1. Общие сведения

Опишем методику проектирования и программирования типичной и широко распространённой игры, когда на форме сначала искусственный интеллект произвольным образом (при помощи генератора случайных чисел – г.с.ч. класса Random) строит разноцветную палитру строк и столбцов из плоских геометрических фигур, в данном примере, из разноцветных кругов.

Затем игрок при помощи указателя мыши быстро выбирает тот цвет, который охватывает как можно большее количество кругов (площадь палитры) и нажимает кнопку мыши (чтобы выбить эти круги из палитры). Круги одинакового цвета, соединённые между собой по горизонтали (по строке) и вертикали (по столбцу) удаляются, а игроку начисляются по 10 очков за каждый выбитый круг. По такой схеме игрок быстро щёлкает мышью по кругам, стараясь за отведённое время выбить как можно больше кругов и соответственно очков.

Искусственный интеллект же периодически дополняет палитру новыми разноцветными кругами (произвольным образом).

Данную игру мы будем разрабатывать, следуя игре Game с сайта microsoft.com. Авторы игры разработали её на устаревшей версии Visual Studio. Поэтому автор данной книги разработал эту игру на современной версии Visual Studio, исправил ошибки и дополнил её недостающими для типичной игры элементами, например, счётчиком секунд на форме и мелодией по окончании времени игры.

В этой игре для отображения поля игры не используются графические файлы (такие файлы формата (.bmp) применяются только для подсчёта очков), поэтому разноцветные круги (по-английски: circle) рисуются при помощи метода FillEllipse класса Graphics из пространства имён System.Drawing в строке:

```
graphics.FillEllipse(brush, New Rectangle(transTopLeft, _
New Size(transwidth, transheight)))
```

Напомним, что, если у эллипса (Ellipse) задать две одинаковые по длине оси, то эллипс будет рисоваться в виде окружности, а закрашенное поле внутри окружности – это круг.

# 20.2. Правила игры

1. После запуска игры на мониторе появляется основная форма (рис. 20.1).

2. Для начала игры щёлкаем или элемент управления PictureBox с рисунком в виде надписи New или в меню Game выбираем команду New Game.

На форме с белым фоном (типа Window) искусственный интеллект выводит палитру из 12 столбцов и 7 рядов (строк) кругов, которые случайным образом (при помощи г.с.ч.) закрашены в 4 цвета: Red – красный, Blue – синий, Green – зелёный и Gray – серый (рис. 20.2).

Если не предпринимать никаких действий, то постепенно через каждый заданный нами (при помощи первого таймера Timer1) интервал времени (в данном проекте, через 7000 миллисекунд или 7 секунд) палитра увеличится до максимального размера из 12 столбцов и 12 рядов разноцветных кругов (рис. 20.3).

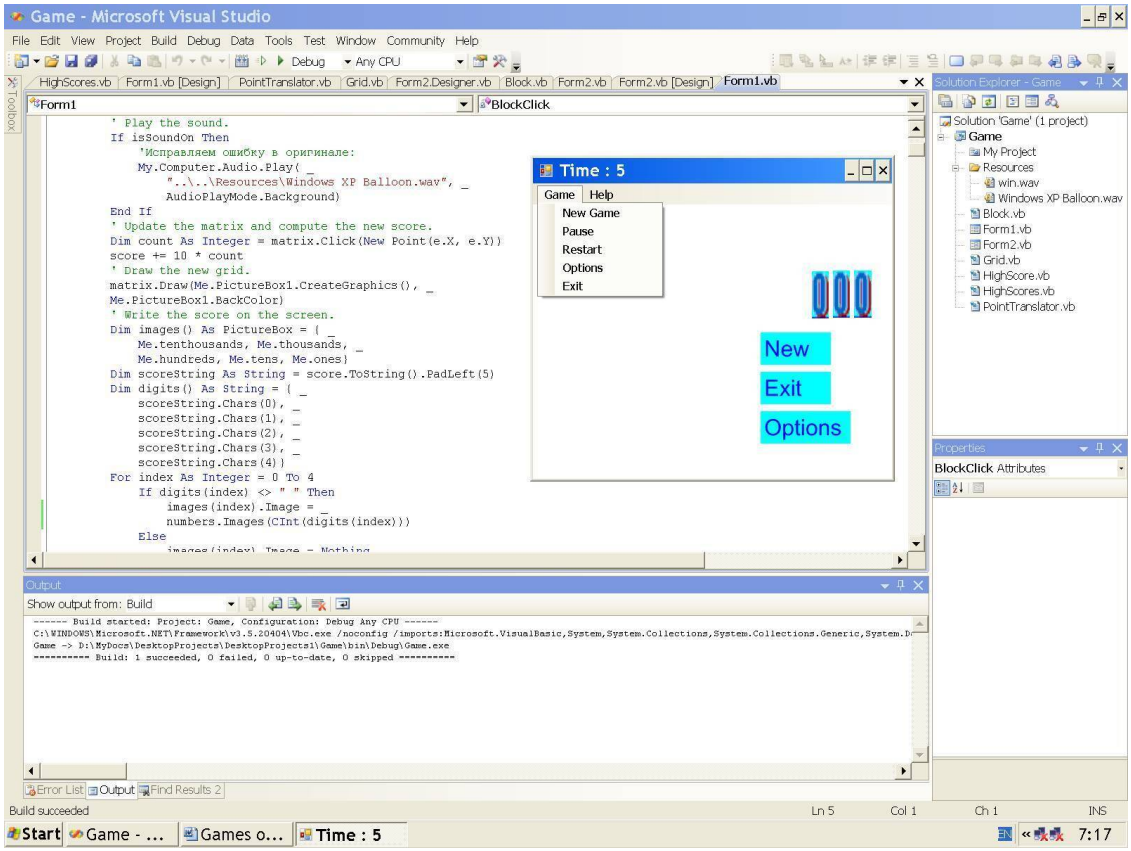



**Рис. 20.1.** Исходная форма.

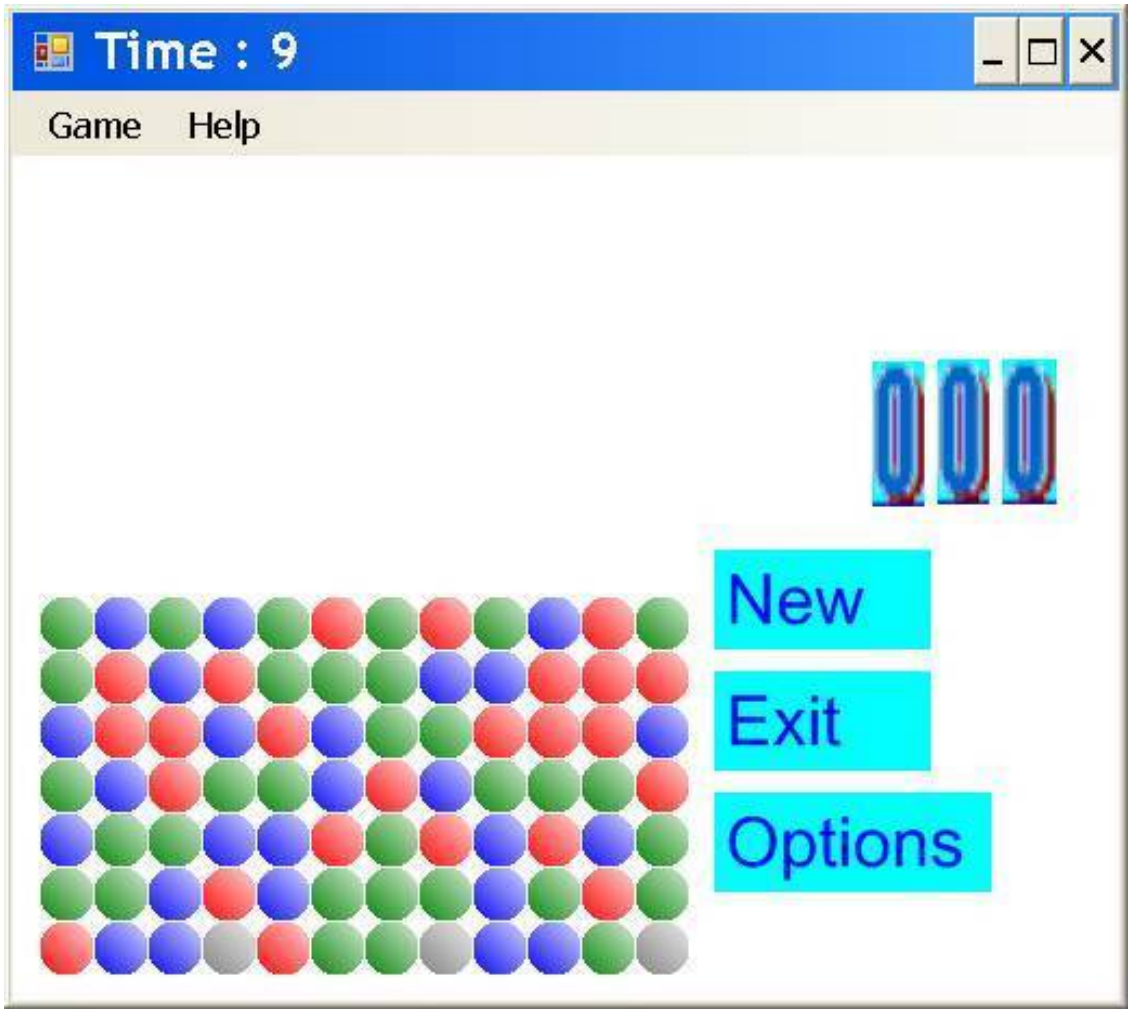



**Рис. 20.2.** Исходная палитра из разноцветных кругов.

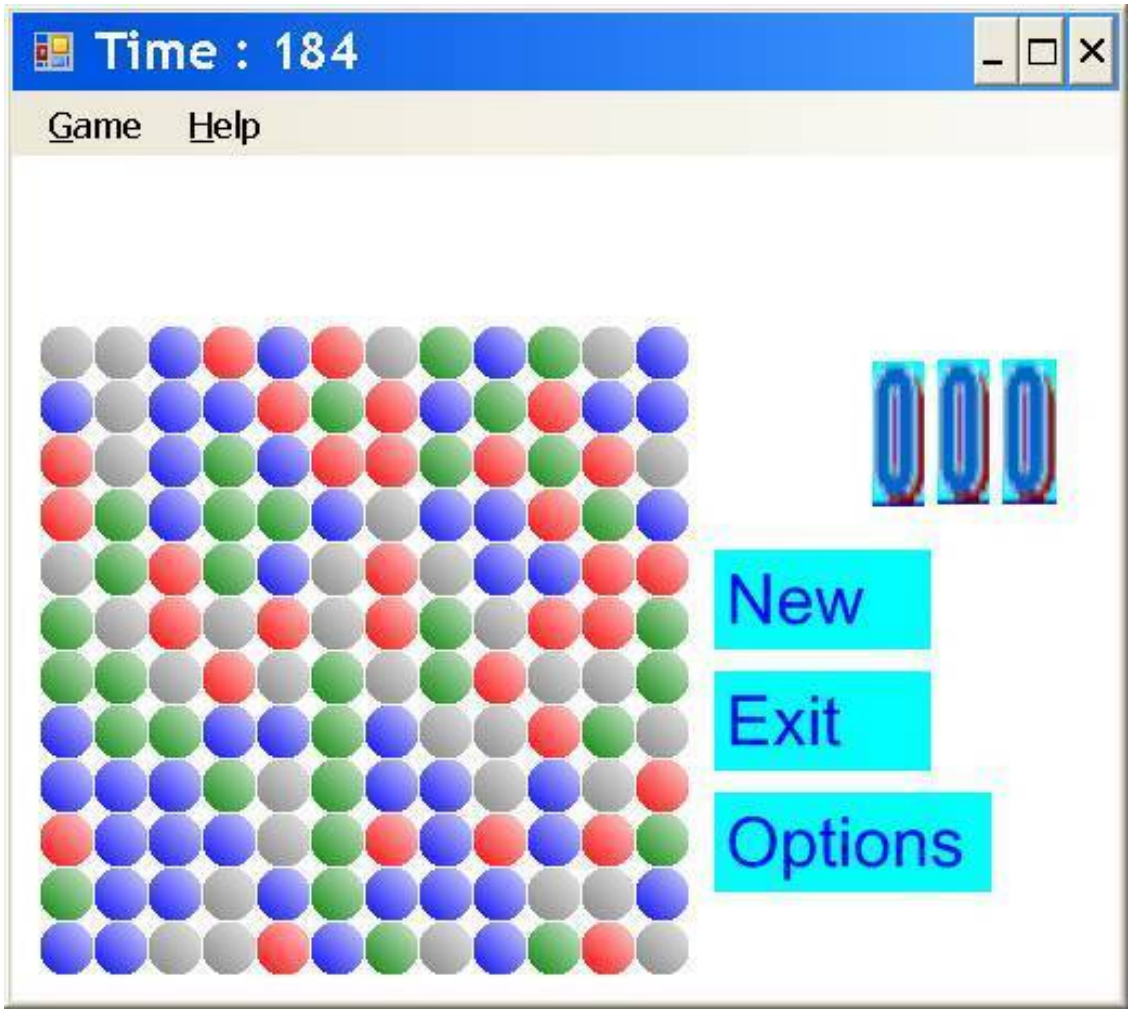



**Рис. 20.3.** Максимальная палитра из разноцветных кругов.

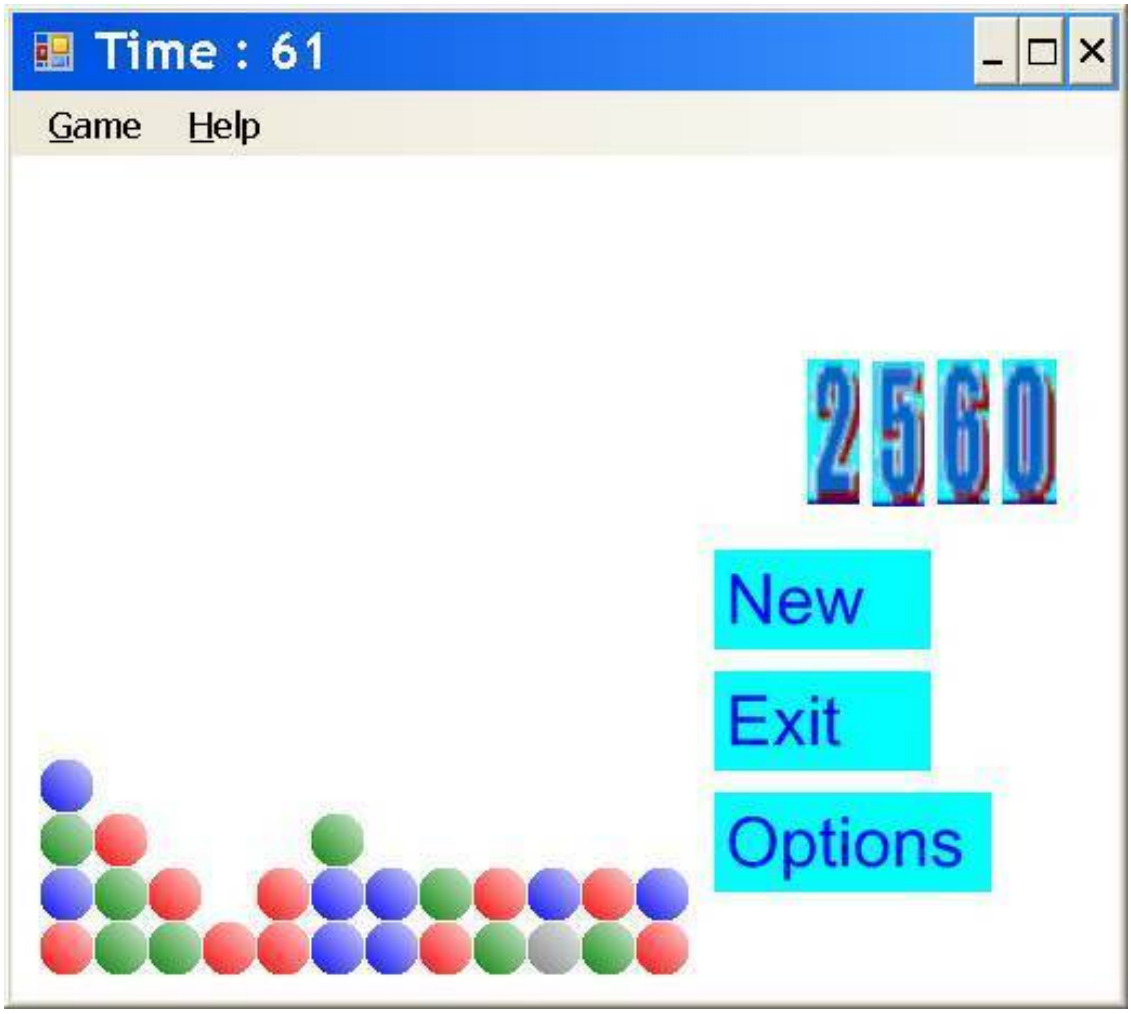



**Рис. 20.4.** Конец игры.

Сразу же после начала игры начинается отсчёт времени (Time) в секундах в верхней части формы на поле для свойства Text при помощи второго таймера Timer2.

3. Смысл игры заключается в следующем.

Сразу же после начала игры игрок должен быстро щёлкать мышью по кругам.

Если игрок щёлкнет круг, вокруг которого находятся круги с цветом, отличным от цвета данного круга, то круг не уничтожается (не исчезает с палитры), а игроку начисляется 10 очков.

Если игрок щёлкнет круг, вокруг которого находятся другие круги с цветом данного круга, то круги одинакового цвета, соединённые между собой по горизонтали (по строке) и вертикали (по столбцу) удаляются, а игроку начисляются по 10 очков за каждый выбитый круг.

Следовательно, игрок при помощи указателя мыши должен быстро выбивать тот цвет, который охватывает как можно большее количество кругов (площадь палитры).

По такой схеме игрок быстро щёлкает мышью по кругам, стараясь за отведённое время выбить как можно больше кругов и соответственно очков. Каждое выбивание кругов сопровождается воспроизведением звукового файла Windows XP Balloon.wav (типа удара).

Искусственный интеллект же периодически через каждый заданный нами (при помощи первого таймера Timer1) интервал времени (в данном проекте, через 7000 миллисекунд или 7 секунд) дополняет палитру новыми разноцветными кругами (произвольным образом).

Игрой можно управлять не только мышью, но и клавишами клавиатуры. Клавиша M (первая буква английского слова Menu) раскрывает и закрывает меню Game, а клавиша P (первая буква английского слова Pause) приостанавливает и запускает игру вновь. После нажатия кла-

виши Alt вместе с клавишей с подчёркнутой буквой (английского алфавита) в команде меню Game или Help, выполняется соответствующая команда.

4. После начала игры идёт отсчёт времени (Time) в секундах в верхней части формы на поле для свойства Text при помощи второго таймера Timer2.

Для каждого сеанса (попытки) игры одного или нескольких игроков задано определённое время, в данном примере, 60 секунд, по истечении которого звучит мелодия файла win.wav.

Игрок прекращает щёлкать мышью и смотрит на заработанные им очки.

5. В данной игре игрок может не только увидеть, но задокументировать заработанные им очки. Для этого он щёлкает на форме или элемент управления PictureBox с рисунком в виде надписи New или в меню Game выбирает команду New Game.

Только при соблюдении двух условий:

– если это первая, вторая или третья попытка;

– если в данной попытке набрано больше очков, чем в предыдущих более чем трёх попытках,

появляется библиотечная (которую мы не будем проектировать) панель InputBox с информацией о заработанных очках (рис. 20.5).

Во всех остальных случаях панель InputBox не появится, что означает проигрыш в игре.

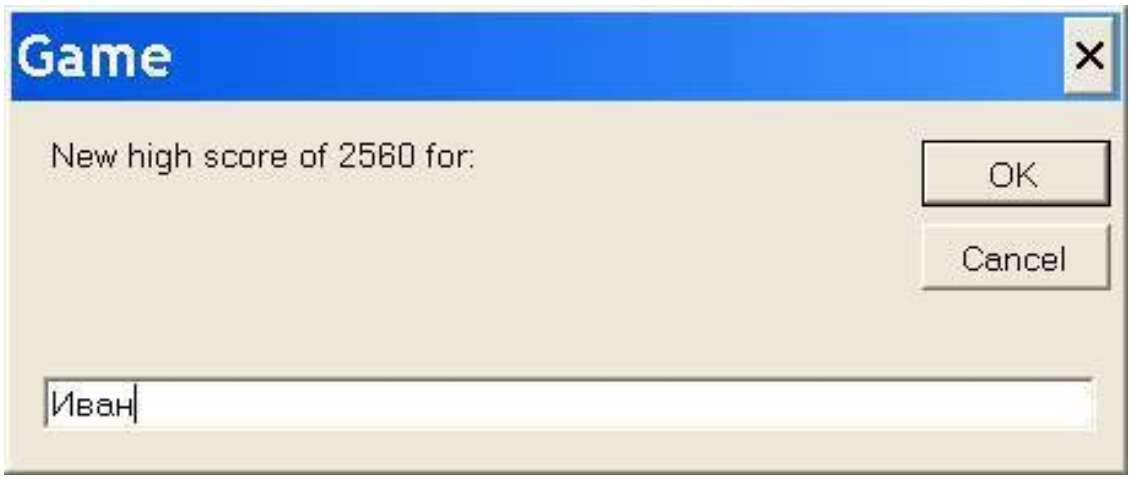



**Рис. 20.5.** Панель InputBox с информацией о заработанных очках.

В эту панель InputBox игрок с радостью (он вошёл в тройку призёров) записывает своё имя (русскими или английскими буквами) и щёлкает кнопку OK. Панель InputBox закрывается.

6. Имя игрока с выбитыми им очками заносятся в таблицу результатов Options (ее мы будем проектировать вместе с основной формой), которую можно увидеть, если на форме выбрать или элемент управления PictureBox с рисунком в виде надписи Options или в меню Game – команду Options. В таблицу Options искусственный интеллект заносит три лучших результата (рис. 20.6), причём на первом месте всегда будет игрок с наибольшим количеством выбитых очков (high score) независимо от количества попыток.

Чтобы очистить таблицу, следует щёлкнуть кнопку Reset. Чтобы выключить музыкальное сопровождение, необходимо снять флажок Sound.

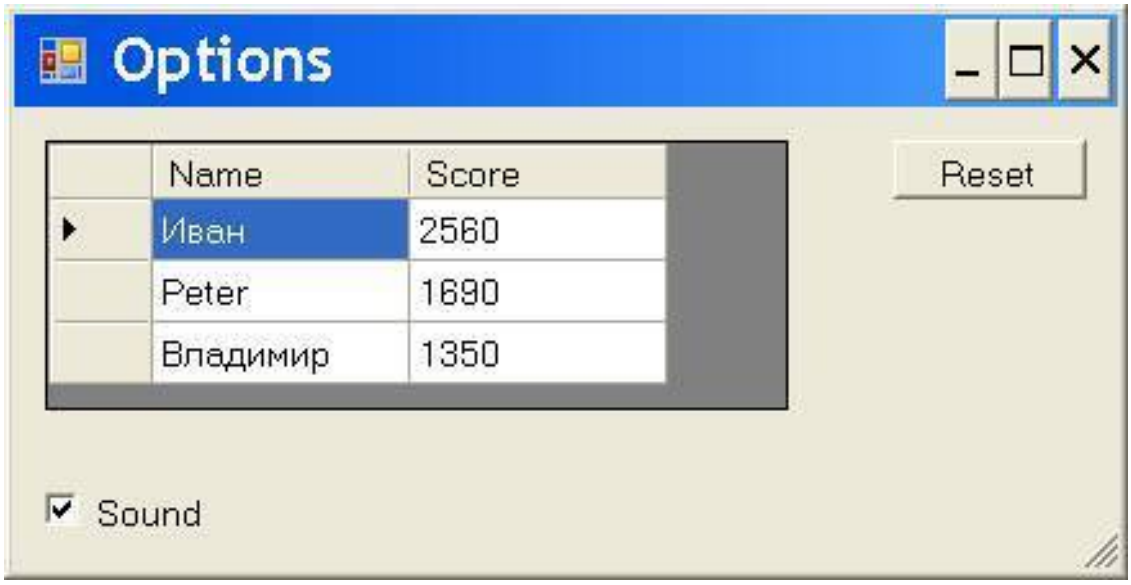



**Рис. 20.6.** Таблица Options с тремя лучшими результатами.

7. Для начала новой попытки игрок снова щёлкает на форме или элемент управления PictureBox с рисунком в виде надписи New или в меню Game выбирает команду New Game.

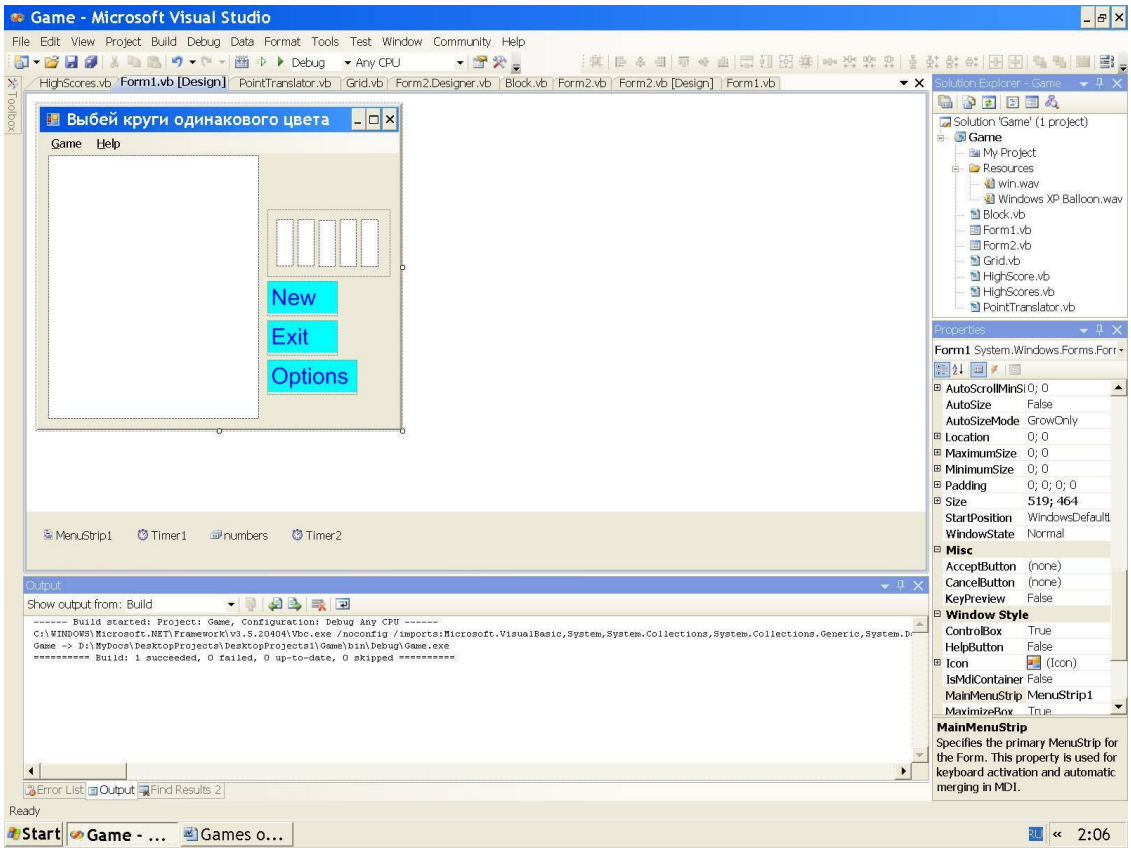
8. Для закрытия игры следует выбрать на форме или элемент управления PictureBox с рисунком в виде надписи Exit или в меню Game выбирает команду Exit

На основании этих правил можно сформулировать другие правила, и любые правила ввести в справочную форму игры, которая появится после выбора команды Contents (Содержание) в меню Help (Помощь) по разработанной нами методике с использованием искусственного интеллекта.

# 20.3. Создание проекта

Создаём проект по обычной схеме: в VS в панели New Project в окне Project types выбираем тип проекта Visual Basic, Windows, в окне Templates выделяем шаблон Windows Forms Application, в окне Name записываем имя проекта Game и щёлкаем OK. Создаётся проект, появляется форма Form1 в режиме проектирования (рис. 20.7). Оставляем по умолчанию или проектируем форму, как подробно описано в параграфе “Методика проектирования формы”. За маркер увеличиваем размеры формы таким образом, чтобы в панели Properties (для Form1) в свойстве Size были значения, например, 519; 464. Белый цвет фона формы мы установим далее в программе (в строке Me.BackColor = Color.White).

Для задания режимов и управления игрой воспользуемся каким-либо элементом управления или компонентом. Как и выше, с панели инструментов Toolbox переносим на форму элемент управления MenuStrip и щёлкаем по нему (ниже формы в режиме проектирования). На форме Form1 появляются окна с надписью Type Here (Печатайте здесь), в которые записываем команды, слева: Game (Игра), New Game (Новая игра), Pause (Пауза), Restart (Перезапуск), Options (Результаты), Exit (Выход), рис. 20.9, справа: Help (Помощь), Contents (Содержание), Index (Указатель), Search (Поиск), About this game (Об этой игре), рис. 20.10.



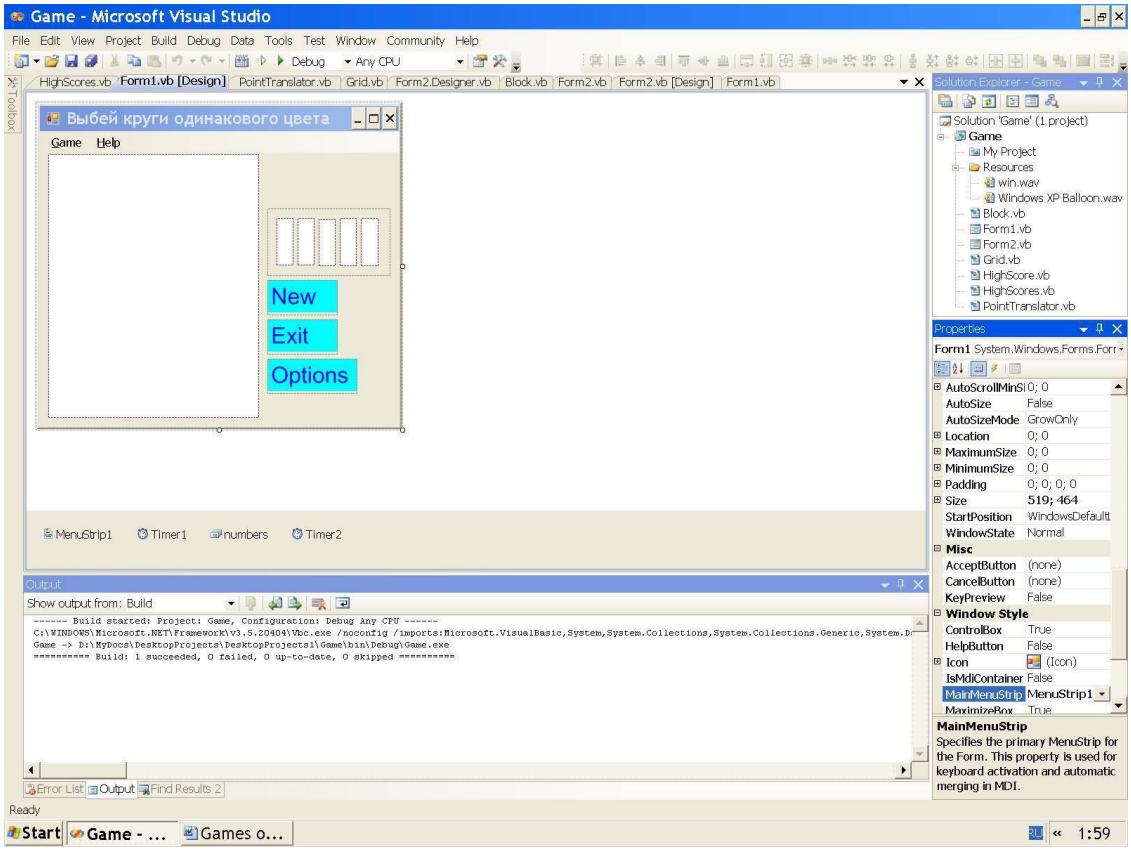
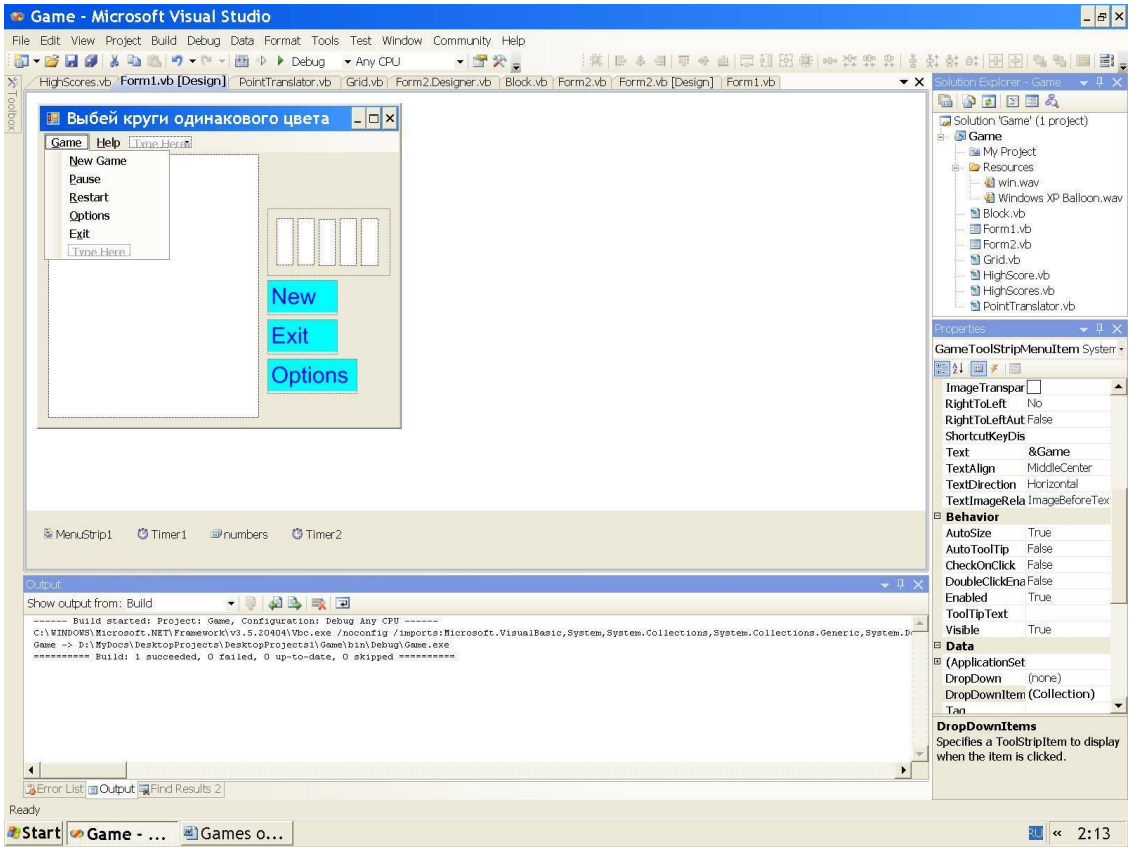



**Рис. 20.7.** Форма Form1 в режиме проектирования. **Рис. 20.8.** SE и Properties.



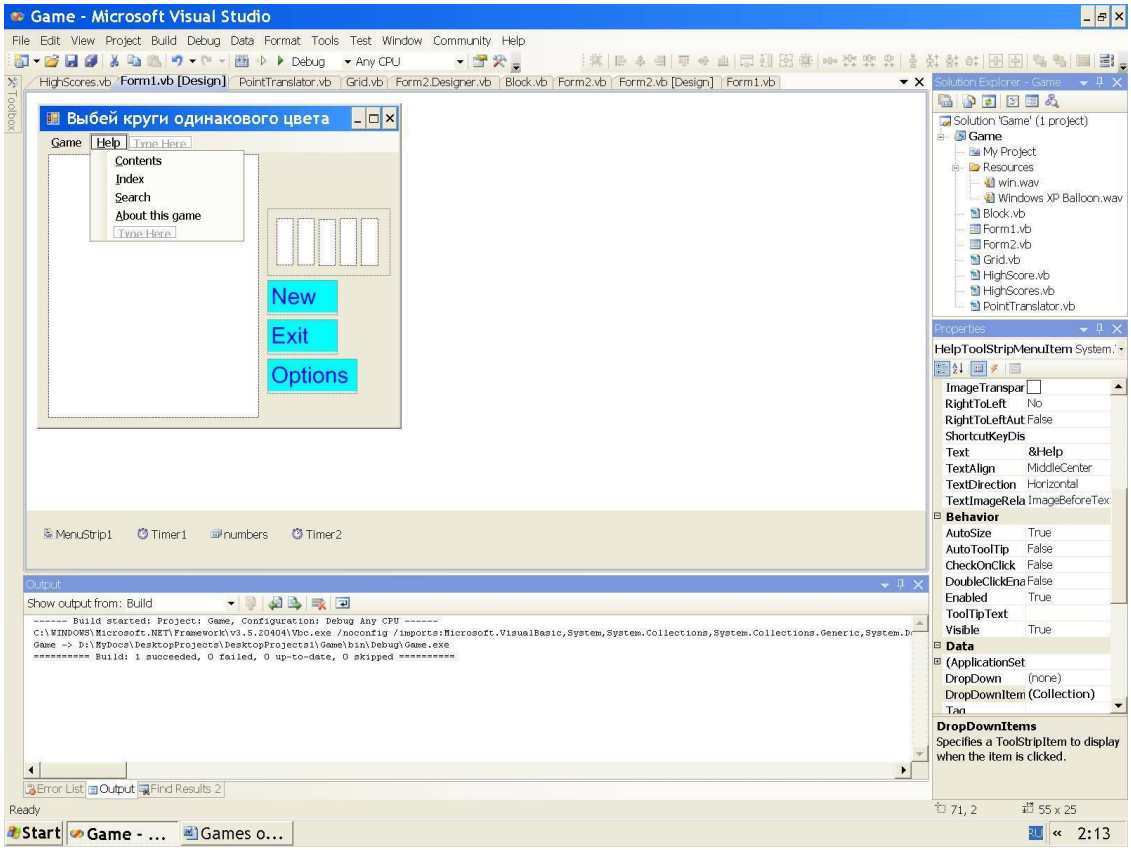



**Рис. 20.9.** Команды меню Game. **Рис. 20.10.** Команды меню Help.

В панели Properties в свойстве Text в имени каждой команды меню MenuStrip перед соответствующей буквой записываем оператор &, после чего на форме эта буква станет подчёркнутой. Напомним, что в режиме выполнения, после нажатия клавиши Alt вместе с клавишей с подчёркнутой буквой (английского алфавита) в команде меню Game или Help, выполняется соответствующая команда.

С панели инструментов Toolbox размещаем на форме основной графический элемент управления PictureBox для поля игры. За маркеры увеличиваем размеры поля, чтобы в панели Properties в свойстве Size были значения 300; 375, а в свойстве BackColor вместо заданного по умолчанию серого цвета Control выбираем белый цвет Window.

Размещаем на форме панель Panel, за её маркеры увеличиваем размеры Size до значений 175; 96.

На этой панели Panel размещаем 5 элементов управления PictureBox с размерами Size (24; 67). В панели Properties в свойстве BackColor для всех этих элементов выбираем белый цвет Window, а в свойстве Name изменяем имена этих 5 элементов на следующие (справа налево):

ones – единицы,

tens – десятки,

hundreds – сотни,

thousands – тысячи,

tenthousands – десятки тысяч.

Ниже на форме размещаем элемент управления PictureBox с заданными по умолчанию размерами Size (100; 50). Чтобы на этом элементе разместить рисунок, в панели Properties щёлкаем свойство Image, в появившейся панели Select Resource выбираем переключатель Local

resource и щёлкаем кнопку Imports (рис. 20.11). В панели Open находим (например, в папке с загруженными из Интернета файлами) графический файл new.bmp (рис. 20.12) и щёлкаем кнопку Open, после чего этот рисунок мы увидим в панели Select Resource, на которой щёлкаем OK. Окончательно, этот рисунок new.bmp разместится в панели Properties в свойстве Image и на форме на поле данного элемента PictureBox.

Аналогично ниже размещаем на форме ещё один элемент управления PictureBox с заданными по умолчанию размерами Size (100; 50), на который добавляем рисунок exit.bmp.

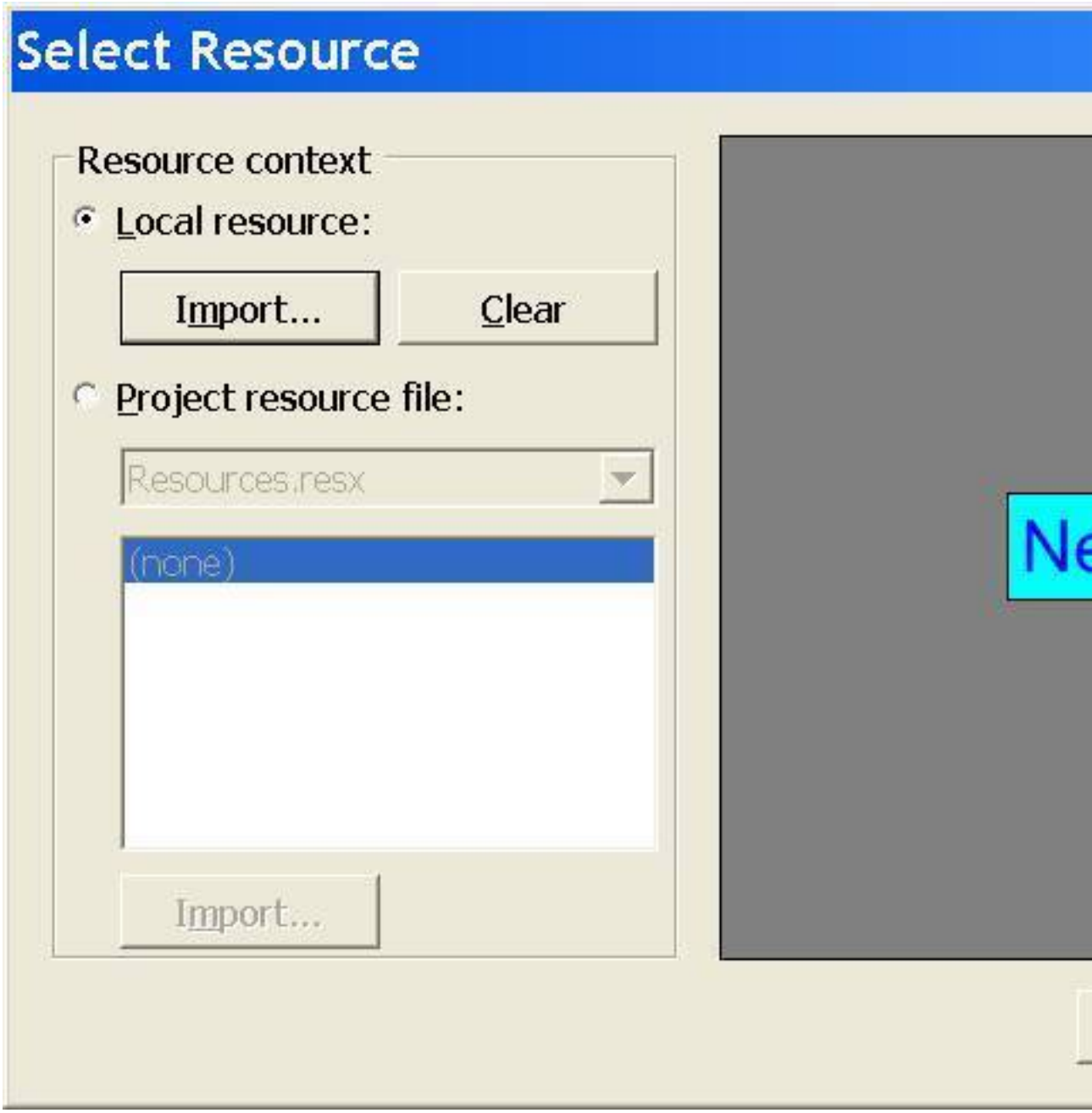



**Рис. 20.11.** В панели Select Resource щёлкаем Imports. **Рис. 20.12.** В панели Open находим файл.

Аналогично ниже размещаем на форме ещё один элемент управления PictureBox, увеличиваем его размеры Size до 128; 50 и добавляем на него рисунок option.bmp.

С панели инструментов Toolbox переносим на форму компонент типа списка рисунков ImageList, который, как компонент, размещается ниже формы. В панели Properties в свойстве

Name изменяем его имя на numbers (цифры для подсчёта очков), а в свойстве ImageSize увеличиваем цифры до размеров 26; 67. Чтобы этот компонент заполнить цифрами для подсчёта очков, в панели Properties щёлкаем свойство Images, в появившейся панели Images Collection Editor щёлкаем кнопку Add (рис. 20.13). В приведённой выше панели Open находим (например, в папке с загруженными из Интернета файлами) графический файл 0.bmp и щёлкаем кнопку Open, после чего этот рисунок мы увидим в панели Images Collection Editor. Аналогично в список рисунков ImageList добавляем остальные цифры 1, 2, 3, …, 9.

Чтобы программа периодически через Interval времени дополняла поле игры новыми разноцветными кругами (взамен выбитых игроком кругов), с панели инструментов Toolbox переносим на форму (точнее, ниже формы) первый таймер Timer1. В панели Properties (для этого таймера) в свойстве Enabled оставляем заданное по умолчанию значение False, т.к. мы включим этот таймер в программе в нужном месте при помощи строки (Timer1.Enabled = True). А в свойстве Interval вместо заданных по умолчанию 100 миллисекунд задаём, например, значение 7000 миллисекунд (равное 7 секундам).

Чтобы в верхней части формы (на синей полоске для свойства Text) после начала игры шел отсчёт времени (Time), на форму переносим второй таймер Timer2. В панели Properties (для этого второго таймера) в свойстве Enabled изменяем заданное по умолчанию значение False на True (включаем таймер), в свойстве Interval вместо заданных по умолчанию 100 миллисекунд задаём значение 1000 (равное 1 секунде), чтобы шел посекундный отсчёт времени.

Если в игре применяются звуковые файлы, то их целесообразно разместить в одной папке с именем, например, Resources. Для добавления в проект этой папки, в панели Solution Explorer выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, New Folder, в поле появившегося значка папки записываем имя папки и нажимаем клавишу Enter.

Добавляем в эту папку первый звуковой файл Windows XP Balloon.wav по стандартной схеме: выполняем правый щелчок по имени этой папки, 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 с загруженными из Интернета файлами) и выделяем имя файла и щёлкаем кнопку Add (или дважды щёлкаем по имени файла). В панели Solution Explorer мы увидим этот файл.

Аналогично добавляем в проект второй файл win.wav.

Напомним, что добавлять в проект указанные выше файлы можно как по одному, так и все сразу (после их выделения или только одной мышью, или мышью с нажатой клавишей Shift – для выделения всех соседних файлов, или мышью с нажатой клавишей Ctrl – для выделения всех файлов в различных местах).

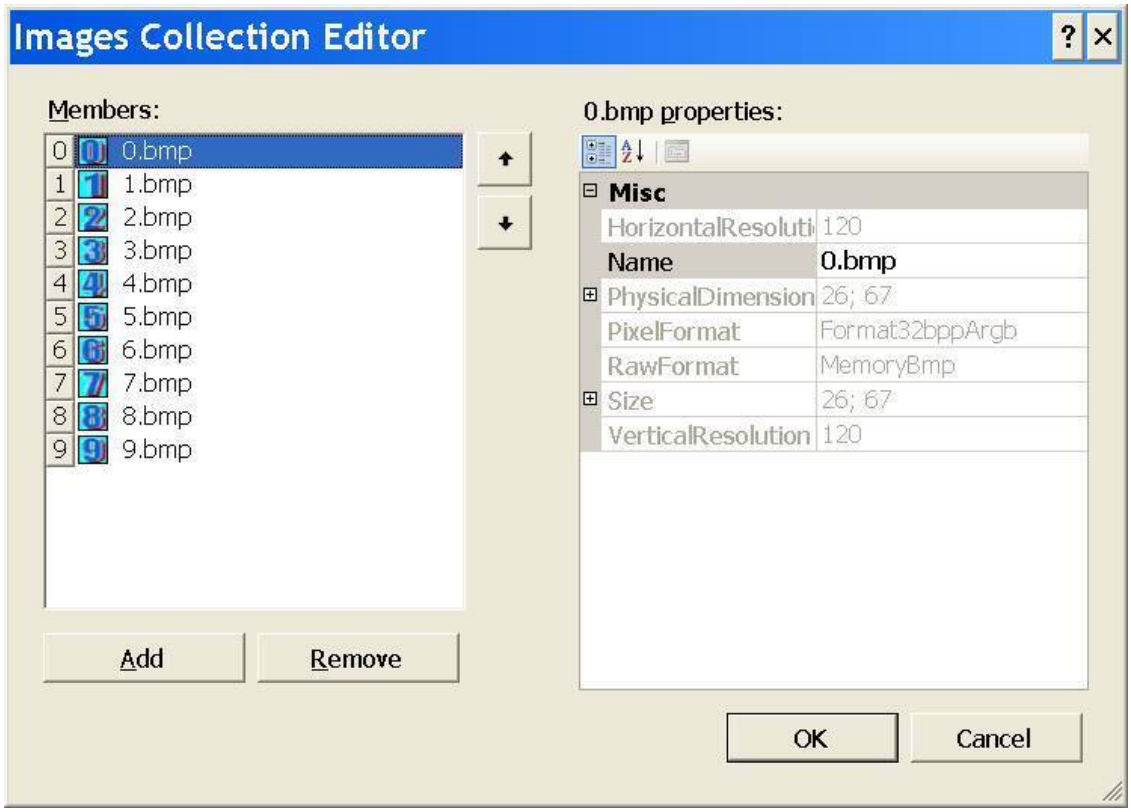



**Рис. 20.13.** В панели Images Collection Editor щёлкаем кнопки Add и OK.

Для ввода в проект новой формы (для таблицы с результатами игры) в меню Project выбираем Add Windows Form, в панели Add New Item оставляем заданные по умолчанию параметры и щёлкаем кнопку Add. В ответ VS выводит новую форму Form2 и добавляет в панель Solution Explorer новый пункт Form2.vb. Аналогично, как первую, проектируем вторую форму (рис. 20.14), за маркеры увеличиваем форму до размеров Size (436; 223) и вводим на форму элементы управления: сетку DataGridView с размерами Size (288; 104), кнопку Button с заголовком Reset (в свойстве Text) и флажок CheckBox, для которого в свойстве Name записываем имя isSoundOn, а в свойстве Checked выбираем значение True (устанавливаем флажок). Свойства этих элементов управления можно стандартно изменять, как описано ранее.

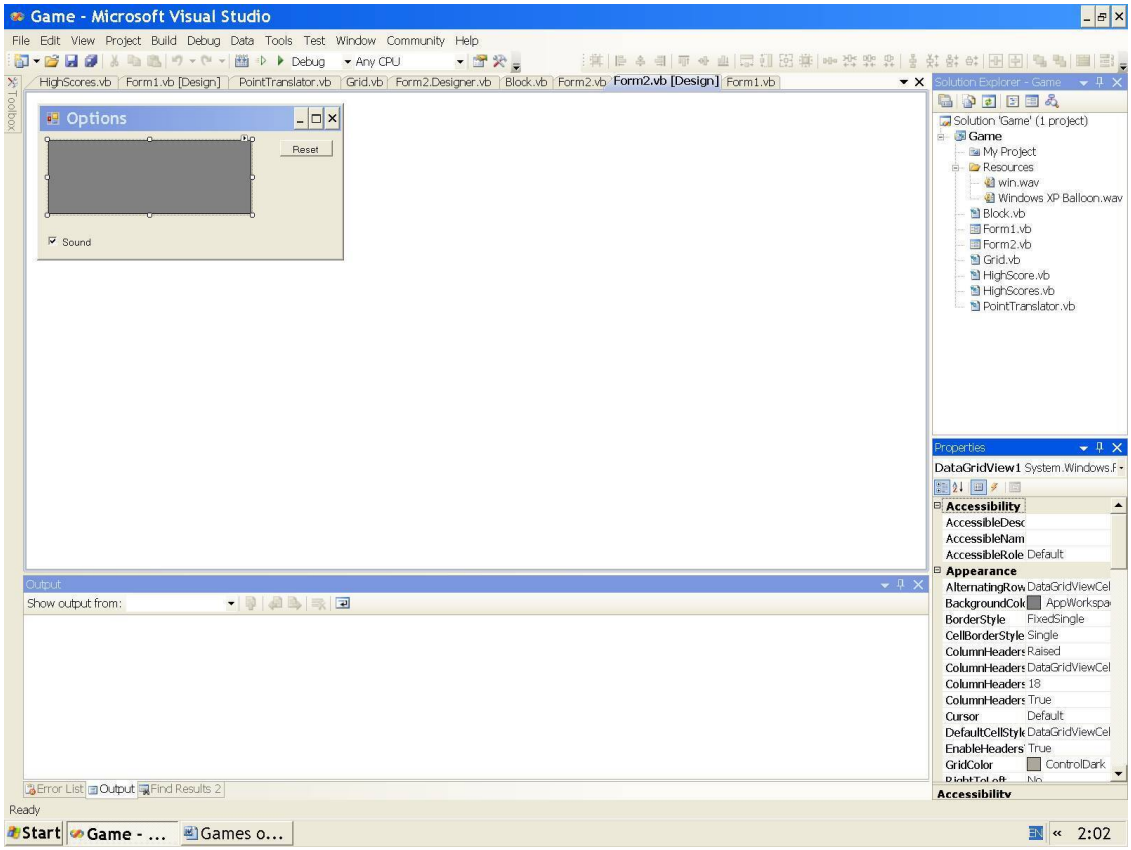



**Рис. 20.14.** Форма Form2 для таблицы с результатами игры.

# 20.4. Код программы

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

**Листинг 20.1.** Переменные и методы.

```
Dim matrix As Grid
Dim score As Integer = 0
Dim mouseOffset As Point
Dim paused As Boolean = False
Dim isSoundOn As Boolean = True
Private Sub BlockClick(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs)
' Play the sound.
If isSoundOn Then
'Исправляем ошибку в оригинале:
My.Computer.Audio.Play( _
"..\..\Resources\Windows XP Balloon.wav", _
AudioPlayMode.WaitToComplete) 'Ждем окончания мелодии.
End If
' Update the matrix and compute the new score.
Dim count As Integer = matrix.Click(New Point(e.X, e.Y))
score += 10 * count
' Draw the new grid.
matrix.Draw(Me.PictureBox1.CreateGraphics(), _
Me.PictureBox1.BackColor)
' Write the score on the screen.
Dim images() As PictureBox = { _
Me.tenthousands, Me.thousands, _
Me.hundreds, Me.tens, Me.ones}
Dim scoreString As String = score.ToString().PadLeft(5)
Dim digits() As String = { _
scoreString.Chars(0), _
scoreString.Chars(1), _
scoreString.Chars(2), _
scoreString.Chars(3), _
scoreString.Chars(4)}
For index As Integer = 0 To 4
If digits(index) <> " " Then
images(index).Image = _
numbers.Images(CInt(digits(index)))
Else
images(index).Image = Nothing
End If
Next
End Sub
Private Sub StartNewGame()
' If a game is already running, check for a new high score.
```

```
If Not matrix Is Nothing Then
Me.Timer1.Enabled = False
HighScores.UpdateScores(score)
End If
Timer1.Enabled = False
matrix = New Grid(6)
score = 0
matrix.Draw(Me.PictureBox1.CreateGraphics(), _
Me.PictureBox1.BackColor)
Timer1.Enabled = True
AddHandler PictureBox1.MouseDown, AddressOf BlockClick
'Обнуляем счётчик секунд:
secondCounter = 0
End Sub
' To pause the game, turn off the timer.
Private Sub Pause()
Timer1.Enabled = False
Me.PauseToolStripMenuItem.Visible = False
Me.RestartToolStripMenuItem.Visible = True
RemoveHandler PictureBox1.MouseDown, AddressOf BlockClick
paused = True
End Sub
Private Sub ShowOptions()
'Dim optionsForm As New Options
Dim optionsForm As New Form2
optionsForm.SoundOn = isSoundOn
optionsForm.ShowDialog()
isSoundOn = optionsForm.SoundOn
optionsForm.Dispose()
End Sub
Private Sub Restart()
Timer1.Enabled = True
Me.PauseToolStripMenuItem.Visible = True
Me.RestartToolStripMenuItem.Visible = False
AddHandler PictureBox1.MouseDown, AddressOf BlockClick
paused = False
End Sub
Private Sub EndGame()
' Get top scores so far.
Me.Timer1.Enabled = False
HighScores.UpdateScores(score)
Me.Close()
End Sub
```

В панели Properties (для Form1) на вкладке Events дважды щёлкаем по имени события Load (Загрузка). Появившийся шаблон метода Form1_Load после записи нашего кода принимает следующий вид.

**Листинг 20.2.** Метод для загрузки объектов.

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
```

```
PointTranslator.Graphics = Me.PictureBox1.CreateGraphics()
Me.PictureBox1.Width = Block.BlockSize * 12
Me.PictureBox1.Height = Block.BlockSize * 15
HighScores.SetUpHighScores()
' Setup the background color and the starting score.
Me.BackColor = Color.White
Me.ones.Image = Me.numbers.Images(0)
Me.tens.Image = Me.numbers.Images(0)
Me.hundreds.Image = Me.numbers.Images(0)
Me.Menu = Nothing
End Sub
```

Дважды щёлкаем по команде New Game для элемента управления MenuStrip. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.3.** Метод-обработчик выбора команды.

```
Private Sub NewGameToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles NewGameToolStripMenuItem.Click
StartNewGame()
End Sub
```

Дважды щёлкаем по команде Pause для элемента управления MenuStrip. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.4.** Метод-обработчик выбора команды.

```
Private Sub PauseToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles PauseToolStripMenuItem.Click
Me.Pause()
End Sub
```

Дважды щёлкаем по команде Restart для элемента управления MenuStrip. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.5.** Метод-обработчик выбора команды.

```
Private Sub RestartToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles RestartToolStripMenuItem.Click
Restart()
End Sub
```

Дважды щёлкаем по команде Options для элемента управления MenuStrip. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.6.** Метод-обработчик выбора команды.

```
Private Sub OptionsToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles OptionsToolStripMenuItem.Click
Dim optionsForm As New Form2
optionsForm.ShowDialog()
End Sub
```

Дважды щёлкаем по команде Exit для элемента управления MenuStrip. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.7.** Метод-обработчик выбора команды.

```
Private Sub ExitToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
```

```
Handles ExitToolStripMenuItem.Click
Me.EndGame()
End Sub
```

Дважды щёлкаем по элементу управления PictureBox с рисунком new.bmp (или в панели Properties для этого элемента на вкладке Events дважды щёлкаем по имени события Click). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.8.** Метод-обработчик щелчка по элементу.

```
Private Sub newGame_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles newGame.Click
StartNewGame()
End Sub
```

Дважды щёлкаем по элементу управления PictureBox с рисунком exit.bmp (или в панели Properties для этого элемента на вкладке Events дважды щёлкаем по имени события Click). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.9.** Метод-обработчик щелчка по элементу.

```
Private Sub exitGame_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles exitGame.Click
EndGame()
End Sub
```

Дважды щёлкаем по элементу управления PictureBox с рисунком options.bmp (или в панели Properties для этого элемента на вкладке Events дважды щёлкаем по имени события Click). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.10.** Метод-обработчик щелчка по элементу.

```
Private Sub options_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles options.Click
ShowOptions()
End Sub
```

Для управления игрой мышью, в панели Properties (для формы Form1) на вкладке Events дважды щёлкаем по имени события MouseDown. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.11.** Метод-обработчик нажатия кнопки мыши.

```
Private Sub Form1_MouseDown(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) _
Handles MyBase.MouseDown
mouseOffset = New Point(-e.X, -e.Y)
End Sub
```

Для управления игрой мышью, в панели Properties (для формы Form1) на вкладке Events дважды щёлкаем по имени события MouseMove. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.12.** Метод-обработчик перемещения мыши.

```
Private Sub Form1_MouseMove(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) _
Handles MyBase.MouseMove
If e.Button = Windows.Forms.MouseButtons.Left Then
Dim mousePos As Point = Control.MousePosition
mousePos.Offset(mouseOffset.X, mouseOffset.Y)
Location = mousePos
End If
```

```
End Sub
```

Для управления игрой клавишами клавиатуры, в панели Properties (для формы Form1) на вкладке Events дважды щёлкаем по имени события KeyPress. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.13.** Метод-обработчик нажатия клавиши.

```
Private Sub Form1_KeyPress(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.KeyPressEventArgs) _
Handles MyBase.KeyPress
Select Case e.KeyChar
Case "p"c, "P"c
If paused Then
Restart()
Else
Pause()
End If
Case "m"c, "M"c
If Me.FormBorderStyle = _
Windows.Forms.FormBorderStyle.Fixed3D Then
Me.FormBorderStyle = _
Windows.Forms.FormBorderStyle.None
Me.Menu = Nothing
Else
Me.FormBorderStyle = _
Windows.Forms.FormBorderStyle.Fixed3D
'Me.Menu = Me.MainMenu1
End If
Case Else
' Do nothing.
End Select
End Sub
```

Чтобы программа периодически через Interval времени дополняла поле игры новыми разноцветными кругами (взамен выбитых игроком кругов), ниже формы дважды щёлкаем по значку для первого таймера Timer1 (или в панели Properties для этого компонента на вкладке Events дважды щёлкаем по имени события Tick). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.14.** Метод, вызываемый через Interval времени.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Timer1.Tick
' Add another row to the grid and update the screen.
matrix.AddRow()
matrix.Draw(Me.PictureBox1.CreateGraphics(), _
Me.PictureBox1.BackColor)
End Sub
```

Чтобы в верхней части формы (на синей полоске для свойства Text) после начала игры шел отсчёт времени (Time), ниже формы дважды щёлкаем по значку для второго таймера Timer2 (или в панели Properties для этого компонента на вкладке Events дважды щёлкаем по имени события Tick). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 20.15.** Метод, вызываемый через Interval времени.

```
'Счётчик секунд, который обнуляем в начале каждой игры
'в методе StartNewGame:
Dim secondCounter As Integer
'Время окончания игры:
Dim EndGameTime As Integer = 60
Private Sub Timer2_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Timer2.Tick
secondCounter = secondCounter + 1
Me.Text = "Time : " & secondCounter.ToString()
'Мелодия окончания игры:
If secondCounter = EndGameTime Then
My.Computer.Audio.Play( _
"..\..\Resources\win.wav", _
AudioPlayMode.Background)
End If
End Sub
```

Схема записи и вывода справочной информации, например, с правилами игры после выбора команды Contents (для элемента управления MenuStrip) и после выбора других команд уже приводилась в наших предыдущих работах.

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

Теперь в наш проект добавляем новые файлы (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 Block.vb 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 20.16.** Новый файл.

```
Imports System.Drawing.Drawing2D
''' <summary>
''' This class represents one of the balls in the game grid.
''' </summary>
''' <remarks></remarks>
Public Class Block
Public Const BlockSize As Integer = 25
Private colorValue As Color
Private deletionValue As Boolean = False
Private Shared rand As New Random
Public Property Color() As Color
Get
Return colorValue
End Get
```

```
Set(ByVal Value As Color)
colorValue = Value
End Set
End Property
Public Property MarkedForDeletion() As Boolean
Get
Return deletionValue
End Get
Set(ByVal Value As Boolean)
deletionValue = Value
End Set
End Property
Public Sub New(ByVal newColor As Color)
colorValue = newColor
End Sub
Public Sub New(ByVal colors() As Color)
Dim ncolors As Integer = colors.Length
Dim pickedColor As Integer
pickedColor = rand.Next(0, ncolors)
colorValue = colors(pickedColor)
End Sub
Public Sub Draw(ByVal graphics As Graphics, ByVal point As Point)
Dim brush As System.Drawing.Drawing2D.LinearGradientBrush = _
CreateTheBrush(point)
DrawTheCircle(graphics, brush, point)
End Sub
Private Sub DrawTheCircle(ByVal graphics As Graphics, _
ByVal brush As LinearGradientBrush, ByVal location As Point)
Dim topleft As Point = location
Dim bottomright As Point = New Point(location.X + _
BlockSize, location.Y + BlockSize)
Dim transTopLeft As Point = PointTranslator.TranslateToBL( _
topleft)
Dim transBottomRight As Point = _
PointTranslator.TranslateToBL(bottomright)
Dim transwidth As Integer = transBottomRight.X – transTopLeft.X
Dim transheight As Integer = _
transBottomRight.Y – transTopLeft.Y
graphics.FillEllipse(brush, New Rectangle(transTopLeft, _
New Size(transwidth, transheight)))
End Sub
Private Function CreateTheBrush(ByVal location As Point) As _
LinearGradientBrush
Dim transLocation As Point = _
PointTranslator.TranslateToBL(location)
Dim brushpt1 As Point = transLocation
Dim brushpt2 As New Point(transLocation.X + Block.BlockSize _
+ 4, transLocation.Y – BlockSize – 4)
Dim brush As New LinearGradientBrush(brushpt1, _
```

```
brushpt2, Me.Color, System.Drawing.Color.White)
Return brush
End Function
End Class
```

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 Grid.vb 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 20.17.** Новый файл.

```
''' <summary>
''' This class represents the grid of blocks. It handles most of
''' the game play.
''' </summary>
''' <remarks></remarks>
Public Class Grid
' The grids is 12 columns and 15 rows of Block objects.
Dim matrix(11, 14) As Block
''' <summary>
''' Creates a few rows of blocks to start the game.
''' Game starts with Red, Blue, and Green blocks.
''' </summary>
''' <param name="nrows">Number of rows of blocks to create
''' to start the game.</param>
''' <remarks></remarks>
Public Sub New(ByVal nrows As Integer)
If nrows > matrix.GetLength(0) Then
Throw New Exception("Must start with " & _
matrix.GetLength(0) & " or fewer rows.")
End If
Dim row As Integer
Dim column As Integer
For row = 0 To nrows – 1
For column = 0 To matrix.GetLength(1) – 1
matrix(row, column) = New Block( _
New Color() {Color.Red, Color.Blue, Color.Green})
Next
Next
For row = nrows To matrix.GetLength(0) – 1
For column = 0 To matrix.GetLength(1) – 1
matrix(row, column) = Nothing
Next
Next
End Sub
''' <summary>
''' A new row may be added at any time. New rows have Gray
''' blocks in addition
''' to Red, Blue, and Green. This makes the game more difficult.
''' </summary>
```

```
''' <remarks></remarks>
Public Sub AddRow()
Dim column As Integer
' Add a new block to each column.
For column = 0 To matrix.GetLength(1) – 1
Dim newBlock As New Block(New Color() _
{Color.Red, Color.Blue, Color.Green, Color.Gray})
' Add the new block at the botttom of the column,

' and push the rest of the
' blocks up one column.
For row As Integer = matrix.GetLength(0) – 1 To 1 Step -1
matrix(row, column) = matrix(row – 1, column)
Next
matrix(0, column) = newBlock
Next
End Sub
''' <summary>
''' Draw the grid of blocks
''' </summary>
''' <param name="graphics"></param>
''' <param name="backColor"></param>
''' <remarks></remarks>
Public Sub Draw(ByVal graphics As Graphics, _
ByVal backColor As Color)
graphics.Clear(backColor)
Dim row As Integer
Dim column As Integer
Dim theBlock As Block
For row = 0 To matrix.GetLength(0) – 1
For column = 0 To matrix.GetLength(1) – 1
theBlock = matrix(row, column)
If Not theBlock Is Nothing Then
Dim pointA As New Point( _
column * Block.BlockSize, _
row * Block.BlockSize)
matrix(row, column).Draw(graphics, pointA)
End If
Next
Next
End Sub
''' <summary>
''' This method responds to a click event in the UI.
''' </summary>
''' <param name="point"></param>
''' <returns>The number of blocks removed from the grid.</returns>
''' <remarks></remarks>
Public Function Click(ByVal point As Point) As Integer
' Figure out row and column.
```

```
Dim total As Integer
Dim transPt As Point = PointTranslator.TranslateToTL(point)
Dim selectedRow As Integer = transPt.Y \ Block.BlockSize
Dim selectedColumn As Integer = transPt.X \ Block.BlockSize
Dim selectedBlock As Block = matrix(selectedRow, _
selectedColumn)
If Not selectedBlock Is Nothing Then
selectedBlock.MarkedForDeletion = True
' Determine if any of the neighboring blocks are
' the same color.
FindSameColorNeighbors(selectedRow, selectedColumn)
' Determine how many blocks would be eliminated.
total = Me.CalculateScore()
If total > 1 Then
Me.CollapseBlocks()
Else
Me.ClearMarkedForDeletion()
End If
End If
Return total
End Function
Private Sub ClearMarkedForDeletion()
Dim row As Integer
Dim column As Integer
For column = matrix.GetLength(1) – 1 To 0 Step -1
' If column is completely empty, then move everthing
' down one.
For row = 0 To matrix.GetLength(0) – 1
If Not matrix(row, column) Is Nothing Then
matrix(row, column).MarkedForDeletion = False
End If
Next
Next
End Sub
''' <summary>
''' Find out how many blocks will be eliminated.
''' </summary>
''' <returns></returns>
''' <remarks></remarks>
Private Function CalculateScore() As Integer
Dim row As Integer
Dim column As Integer
Dim total As Integer = 0
For column = matrix.GetLength(1) – 1 To 0 Step -1
' If column is completely empty, then move everthing
' down one.
For row = 0 To matrix.GetLength(0) – 1
If Not matrix(row, column) Is Nothing Then
If matrix(row, column).MarkedForDeletion Then
```

```
total += 1
End If
End If
Next
Next
Return total
End Function
''' <summary>
''' After the blocks are removed from the columns, there may be
''' columns that are empty. Move columns from right to left to
''' fill in the empty columns.
''' </summary>
''' <remarks></remarks>
Public Sub CollapseColumns()
Dim row As Integer
Dim column As Integer
For column = matrix.GetLength(1) – 1 To 0 Step -1
' If column is completely empty, then all the columns
' over one.
Dim noBlocks As Boolean = True
For row = 0 To matrix.GetLength(0) – 1
If Not matrix(row, column) Is Nothing Then
noBlocks = False
End If
Next
If noBlocks Then
Dim newcol As Integer
For newcol = column To matrix.GetLength(1) – 2
For row = 0 To matrix.GetLength(0) – 1
matrix(row, newcol) = matrix(row, newcol + 1)
Next
Next
newcol = matrix.GetLength(1) – 1
For row = 0 To matrix.GetLength(0) – 1
matrix(row, newcol) = Nothing
Next
End If
Next

End Sub
''' <summary>
''' Remove all the blocks from the grid.
''' </summary>
''' <remarks></remarks>
Public Sub CollapseBlocks()
Dim theBlock As Block
Dim column As Integer
Dim row As Integer
Dim aRow As Integer
```

```
' First remove the blocks from each column.
For column = 0 To matrix.GetLength(1) – 1
For row = matrix.GetLength(0) – 1 To 0 Step -1
theBlock = matrix(row, column)
If (Not theBlock Is Nothing) Then
If theBlock.MarkedForDeletion Then
For aRow = row To matrix.GetLength(0) – 2
matrix(aRow, column) = _
matrix(aRow + 1, column)
Next
matrix(matrix.GetLength(0) – 1, _
column) = Nothing
End If
End If
Next
Next
' Reset the MarkedForDeletion flags.
For row = 0 To matrix.GetLength(0) – 1
For column = 0 To matrix.GetLength(1) – 1
theBlock = matrix(row, column)
If Not theBlock Is Nothing Then
theBlock.MarkedForDeletion = False
End If
Next
Next
' Remove any columns that are now empty.
CollapseColumns()
End Sub
''' <summary>
''' Provides access into the grid.
''' </summary>
''' <param name="row"></param>
''' <param name="column"></param>
''' <value></value>
''' <remarks></remarks>
Default Public Property Item(ByVal row As Integer, _
ByVal column As Integer) As Block
Get
Return matrix(row, column)
End Get
Set(ByVal Value As Block)
matrix(row, column) = Value
End Set
End Property
Private blocksToExamine As ArrayList
''' <summary>
''' Set MarkedForDeletion to True for each neighboring block
''' of the same color.
''' </summary>
```

```
''' <param name="row"></param>
''' <param name="column"></param>
''' <remarks></remarks>
Private Sub FindSameColorNeighbors(ByVal row As Integer, _
ByVal column As Integer)
Dim color As Color = matrix(row, column).Color
blocksToExamine = New ArrayList
blocksToExamine.Add(New Point(row, column))
matrix(row, column).MarkedForDeletion = True
' Each time you find a neighbor, mark it for deletion, and
' add it to the list of blocks to look for neighbors.
' After you
' examine it, remove it from the list. Keep doing this
' until there are no more blocks to look at.
While blocksToExamine.Count > 0
FindNeighbors()
End While
End Sub
''' <summary>
''' Look to the blocks on each side.
''' </summary>
''' <remarks></remarks>
Private Sub FindNeighbors()
' Take the first block out of the arraylist and examine it.
Dim location As Point = CType(blocksToExamine(0), Point)
Dim currentBlock As Block = matrix(location.X, location.Y)
Dim row As Integer = location.X
Dim column As Integer = location.Y
blocksToExamine.RemoveAt(0)
Dim nextRow As Integer
Dim nextCol As Integer
Dim selected As Block
' look up
If row < matrix.GetLength(0) – 1 Then
nextRow = row + 1
selected = matrix(nextRow, column)
ExamineNeighbor(selected, nextRow, column, _
currentBlock.Color)
End If
' look down
If row > 0 Then
nextRow = row – 1
selected = matrix(nextRow, column)
ExamineNeighbor(selected, nextRow, column, _
currentBlock.Color)
End If
' look left
If column > 0 Then
nextCol = column – 1
```

```
selected = matrix(row, nextCol)
ExamineNeighbor(selected, row, nextCol, _
currentBlock.Color)
End If
' look right
If column < matrix.GetLength(1) – 1 Then
nextCol = column + 1
selected = matrix(row, nextCol)
ExamineNeighbor(selected, row, nextCol, _
currentBlock.Color)
End If
End Sub
''' <summary>
''' If the neighbor is the same color, add it to the blocks
''' to examine.
''' </summary>
''' <param name="selected"></param>
''' <param name="row"></param>
''' <param name="column"></param>
''' <param name="color"></param>
''' <remarks></remarks>
Private Sub ExamineNeighbor(ByVal selected As Block, _
ByVal row As Integer, ByVal column As Integer, _
ByVal color As Color)
If Not selected Is Nothing Then
If selected.Color.Equals(color) Then
If Not selected.MarkedForDeletion Then
selected.MarkedForDeletion = True
blocksToExamine.Add(New Point(row, column))
End If
End If
End If
End Sub
End Class
```

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 HighScore.vb 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 20.18.** Новый файл.

```
''' <summary>
''' Represents one high score.
''' </summary>
''' <remarks></remarks>
Public Class HighScore
Implements IComparable
Public nameValue As String
Public scoreValue As Integer
Public Property Name() As String
```

```
Get
Return nameValue
End Get
Set(ByVal Value As String)
nameValue = Value
End Set
End Property
Public Property Score() As Integer
Get
Return scoreValue
End Get
Set(ByVal Value As Integer)
scoreValue = Value
End Set
End Property
Public Overrides Function ToString() As String
Return Name & ":" & Score
End Function
Public Sub New(ByVal saved As String)
Name = saved.Split(":".ToCharArray)(0)
Score = CInt(saved.Split(":".ToCharArray)(1))
End Sub
Public Function CompareTo(ByVal obj As Object) As Integer Implements System.IComparable.CompareTo
Dim other As HighScore
other = CType(obj, HighScore)
Return Me.Score – other.Score
End Function
End Class
```

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 HighScores.vb 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 20.19.** Новый файл.

```
Imports Microsoft.Win32
''' <summary>
''' Reads and writes the top three high scores to the registry.
''' </summary>
''' <remarks></remarks>
Public Class HighScores
''' <summary>
''' Read scores from the registry.
''' </summary>
''' <returns></returns>
''' <remarks></remarks>
Public Shared Function GetHighScores() As HighScore()
Dim tops(2) As HighScore
Dim scoreKey As RegistryKey = Registry.CurrentUser. _
```

```
CreateSubKey("Software\VBSamples\Collapse\HighScores")
For index As Integer = 0 To 2
Dim key As String = "place" & index.ToString
Dim score As New HighScore(CStr(scoreKey.GetValue(key)))
tops(index) = score
Next
scoreKey.Close()
Return tops
End Function
''' <summary>
''' Update and write the high scores.
''' </summary>
''' <param name="score"></param>
''' <remarks></remarks>
Public Shared Sub UpdateScores(ByVal score As Integer)
Dim tops(3) As HighScore
Dim scoreKey As RegistryKey = Registry.CurrentUser. _
CreateSubKey("Software\VBSamples\Collapse\HighScores")
tops(0) = New HighScore(scoreKey.GetValue("Place0").ToString)
tops(1) = New HighScore(scoreKey.GetValue("Place1").ToString)
tops(2) = New HighScore(scoreKey.GetValue("Place2").ToString)
If score > tops(2).Score Then
Dim name As String = InputBox("New high score of " & _
score & " for:")
tops(3) = New HighScore(" :0")
tops(3).Name = name
tops(3).Score = score
Array.Sort(tops)
Array.Reverse(tops)
scoreKey.SetValue("Place0", tops(0).ToString)
scoreKey.SetValue("Place1", tops(1).ToString)
scoreKey.SetValue("Place2", tops(2).ToString)
End If
scoreKey.Close()
End Sub
''' <summary>
''' Set up the entries for new scores.
''' </summary>
''' <remarks></remarks>
Shared Sub SetUpHighScores()
Dim scoreKey As RegistryKey = Registry.CurrentUser. _
CreateSubKey("Software\VBSamples\Collapse\HighScores")
If scoreKey.GetValue("Place1") Is Nothing Then
scoreKey.SetValue("Place0", " :0")
scoreKey.SetValue("Place1", " :0")
scoreKey.SetValue("Place2", " :0")
End If
scoreKey.Close()
End Sub
```

```
''' <summary>
''' Reset scores.
''' </summary>
''' <remarks></remarks>
Shared Sub ResetScores()
Dim scoreKey As RegistryKey = Registry.CurrentUser. _
CreateSubKey("Software\VBSamples\Collapse\HighScores")
scoreKey.SetValue("Place0", " :0")
scoreKey.SetValue("Place1", " :0")
scoreKey.SetValue("Place2", " :0")
scoreKey.Close()
End Sub
End Class
```

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 PointTranslator.vb 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 20.20.** Новый файл.

```
''' <summary>
''' Form coordinates have the top, left as (0,0). For the game grid,
''' it is easier to have the bottom left of the grid as (0,0). This
''' translates the points.
''' </summary>
''' <remarks></remarks>
Public Class PointTranslator
Private Shared graphicsValue As Graphics
Private Shared height As Integer
Public Shared Property Graphics() As Graphics
Get
Return graphicsValue
End Get
Set(ByVal Value As Graphics)
graphicsValue = Value
height = CInt(graphicsValue.VisibleClipBounds.Height())
End Set
End Property
' Translates an (X,Y) point from the top left to
' an (X, Y) point from the bottom left.
Public Shared Function TranslateToBL(ByVal topleft As Point) _
As Point
Dim newPoint As Point
newPoint.X = topleft.X
newPoint.Y = height – topleft.Y
Return newPoint
End Function
Public Shared Function TranslateToTL(ByVal bottomleft As Point) _
As Point
Dim newPoint As Point
```

```
newPoint.X = bottomleft.X
newPoint.Y = height – bottomleft.Y
Return newPoint
End Function
End Class
```

После этих добавлений ( Block.vb, Grid.vb, HighScore.vb, HighScores.vb, PointTranslator.vb) в панели Solution Explorer должны быть файлы, показанные выше. Дважды щёлкая по имени файла, любой файл можно открыть, изучить и редактировать.

Теперь в наш проект добавляем переменные и методы, связанные с формой Form2 для вывода результатов игры.

Открываем файл Form2.vb (например, по схеме: File, Open, File) и в классе Form2 нашего проекта записываем следующее свойство.

**Листинг 20.21.** Свойство.

```
Public Property SoundOn() As Boolean
Get
Return Me.isSoundOn.Checked
End Get
Set(ByVal Value As Boolean)
Me.isSoundOn.Checked = Value
End Set
End Property
```

В панели Properties (для Form2) на вкладке Events дважды щёлкаем по имени события Load (Загрузка). Появившийся шаблон метода Form2_Load после записи нашего кода принимает следующий вид.

**Листинг 20.22.** Метод для загрузки результатов игры.

```
Private Sub Form2_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
Me.DataGridView1.DataSource = HighScores.GetHighScores()
End Sub
```

На форме Form2 дважды щёлкаем по кнопке Button. Появляется шаблон метода (для очистки таблицы результатов), который после записи нашего кода принимает следующий вид.

**Листинг 20.23.** Метод-обработчик щелчка кнопки.

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
HighScores.ResetScores()
Me.DataGridView1.DataSource = HighScores.GetHighScores()
End Sub
```

В случае необходимости, методика добавления в проект звукового сигнала Beep (по-русски: Бип) описана ранее.

## 20.5. Запуск игры

Строим и запускаем программу на выполнение обычным образом:

Build, Build Selection; Debug, Start Without Debugging.

В ответ Visual Studio выводит показанную выше форму, на которой искусственный интеллект произвольным образом (при помощи генератора случайных чисел – г.с.ч. класса Random) строит разноцветную палитру строк и столбцов из плоских геометрических фигур, в данном примере, из разноцветных кругов.

Затем игрок при помощи указателя мыши быстро выбирает тот цвет, который охватывает как можно большее количество кругов (площадь палитры) и нажимает кнопку мыши (чтобы выбить эти круги из палитры).

Круги одинакового цвета, соединённые между собой по горизонтали (по строке) и вертикали (по столбцу) искусственный интеллект удаляет, а игроку начисляются по 10 очков за каждый выбитый круг.

По такой схеме игрок быстро щёлкает мышью по кругам, стараясь за отведённое время выбить как можно больше кругов и соответственно очков (согласно приведённым выше правилам).

По методике данной главы можно разрабатывать самые разнообразные игры по выбиванию фигур одного цвета из разноцветной палитры разнообразных фигур с использованием искусственного интеллекта.

# Глава 21. Методика программирования искусственного интеллекта в игре по сборке прямых из 5 и более объектов одинакового цвета

## 21.1. Общие сведения

Опишем методику проектирования и программирования типичной и широко распространённой игры игрока с искусственным интеллектом в виде компьютера, когда на форме в отдельных квадратах сетки, например, 9 x 9 сначала произвольным образом (при помощи генератора случайных чисел – г.с.ч. класса Random) появляется определённое количество, в данной игре 3 разноцветных объекта, например, 3 больших мяча, которые игрок может перемещать при помощи мыши, и 3 маленьких разноцветных мяча, которые размещает искусственный интеллект, чтобы помешать игроку построить прямую линию из мячей (так как в клетку с маленьким мячом большой мяч уже нельзя разместить).

Затем игрок при помощи указателя мыши быстро выбирает (щёлкает) тот большой мяч, который он желает переместить к другому мячу (или мячам) того же цвета. Мяч, по которому игрок щёлкнул, начинает пульсировать. Игрок второй раз щёлкает на той пустой клетке, в которую должен переместиться мяч, чтобы образовать прямую из мячей одинакового цвета. Мяч перемещается в эту клетку. После этого искусственный интеллект размещает в пустующие клетки следующие 3 больших мяча произвольных цветов.

Аналогично игрок снова щёлкает по выбранному им мячу и по той клетке, в которую мяч перемещается. После этого искусственный интеллект снова размещает в пустующие клетки следующие 3 больших мяча произвольных цветов.

Как только игрок соберёт горизонтальную, вертикальную или диагональную прямую линию из 5 и более мячей одинакового цвета, игроку начисляются очки (по 100 очков за каждый собранный в линию мяч), а линия из собранных мячей исчезает, освобождая клетки для новых мячей.

По такой схеме игрок быстро щёлкает мышью по мячам, стараясь за отведённое время собрать как можно больше линий из мячей одинакового цвета. Искусственный интеллект же периодически (после каждого второго щелчка игрока и следующего за ним перемещения пульсирующего мяча в новую клетку) дополняет клетки 3 новыми разноцветными мячами (произвольным образом).

Данную игру мы будем разрабатывать, следуя игре Line (проект lines_vbnet.zip в заархивированном виде) из Интернета, которая разработана на устаревшей версии Visual Studio. Поэтому автор данной книги разработал эту игру на современной версии Visual Studio, исправил ошибки и дополнил её недостающими для типичной игры элементами, например, счётчиком секунд и мелодиями начала и окончания времени игры.

В панели Solution Explorer (нашего будущего проекта) дважды щёлкаем по имени графического файла BlackBall.png для большого мяча чёрного цвета. Появляется собственный графический редактор Visual Studio с изображением этого мяча, причём инструментами этого редактора можно редактировать это изображение (рис. 21.1) применительно к нашим задачам. Видно, что за счёт раскраски круга различными оттенками соответствующего цвета (в данном случае, чёрного цвета) создаётся впечатление световых бликов, и плоский чёрный круг

нами воспринимается как объёмный резиновый мяч. Аналогично можно увидеть графические файлы больших и маленьких мячей всех цветов в данном проекте.

В начале игры звучит мелодия:

```
My.Computer.Audio.Play("..\..\Sounds\drumpad-crash.wav")
```

Чтобы можно было ограничить игру по времени при помощи определённой мелодии, используем такой код:

```
'Счётчик секунд, который обнуляем в начале каждой игры
'в методе NewGame:
Dim secondCounter As Integer
'Время, через которое звучит мелодия
'возможного окончания игры:
Dim EndGameTime As Integer = 60
Private Sub tmr2_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles tmr2.Tick
DDTime.number += 1
lblTime.Refresh()
'Счётчик секунд:
secondCounter = secondCounter + 1
'Мелодия окончания игры:
If secondCounter = EndGameTime Then
My.Computer.Audio.Play("..\..\Sounds\win.wav", _
AudioPlayMode.Background)
End If
End Sub
```

В этом коде мелодия звучит через 60 секунд. Можно задать любое время, по окончании которого сеанс игры можно прекратить, записать набранные очки и начать новый сеанс игры или допустить к игре следующего игрока.

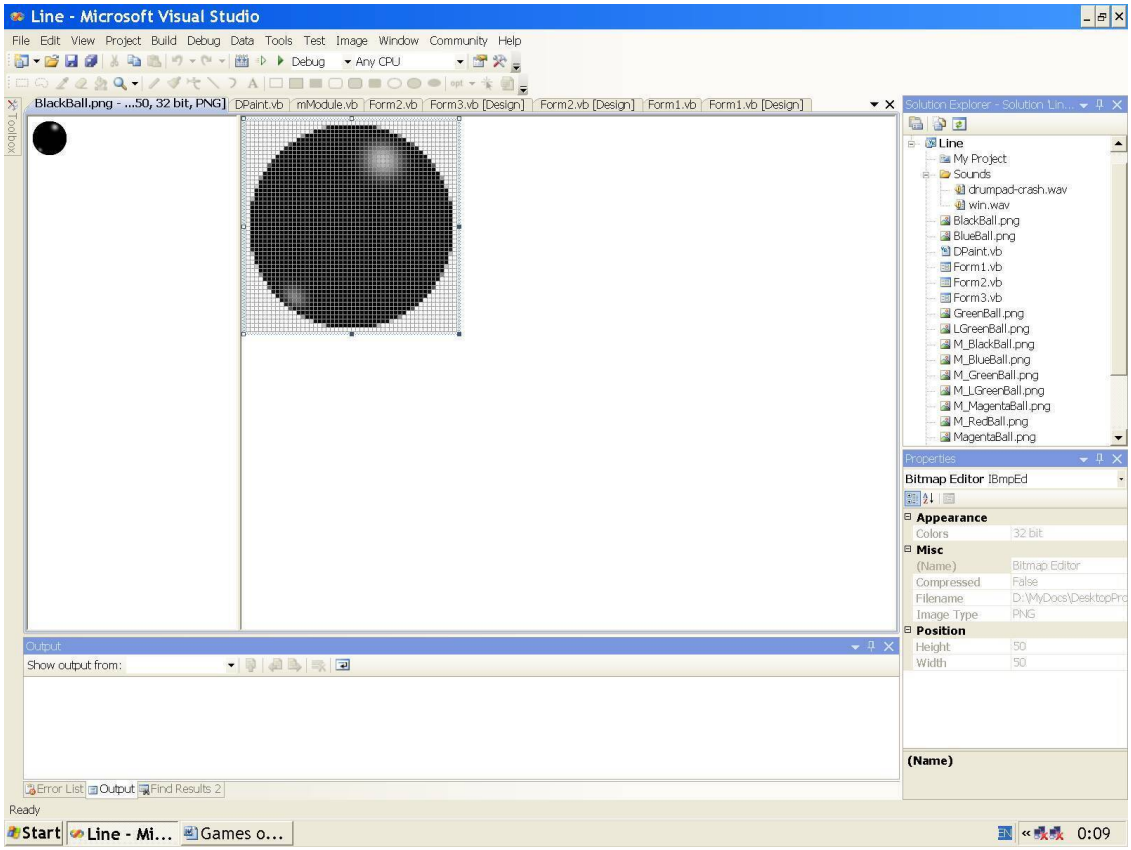



**Рис. 21.1.** Графический редактор Visual Studio с изображением мяча.

## 21.2. Правила игры

1. После запуска данной игры (игрока с компьютером) на мониторе появляется основная форма (рис. 21.2).

2. Для начала игры в меню Игра выбираем команду Новая.

Появляется стандартная (нам не нужно её проектировать) форма InputBox для записи имени игрока (рис. 21.3). Записываем (русскими или английскими буквами) имя и щёлкаем кнопку OK (или нажимаем клавишу Enter, поскольку, как мы видим, эта клавиша выделена по умолчанию).

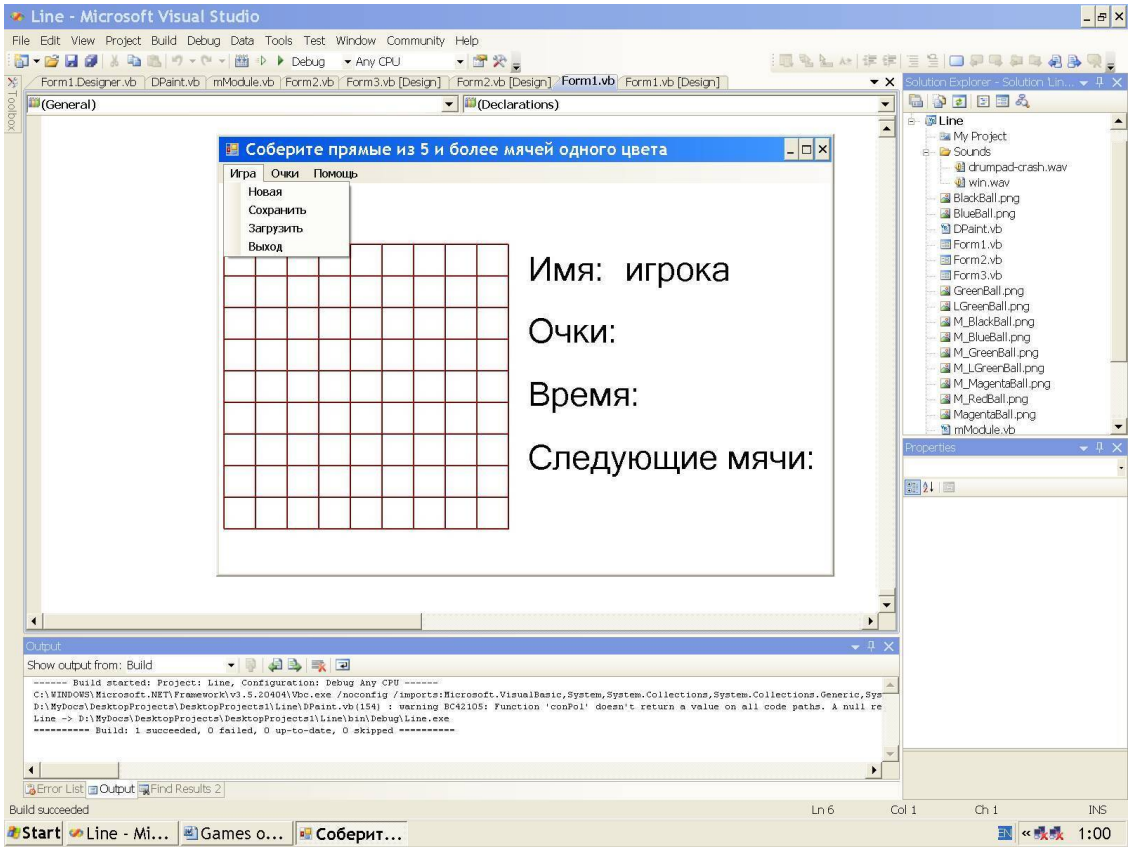



**Рис. 21.2.** Исходная форма.

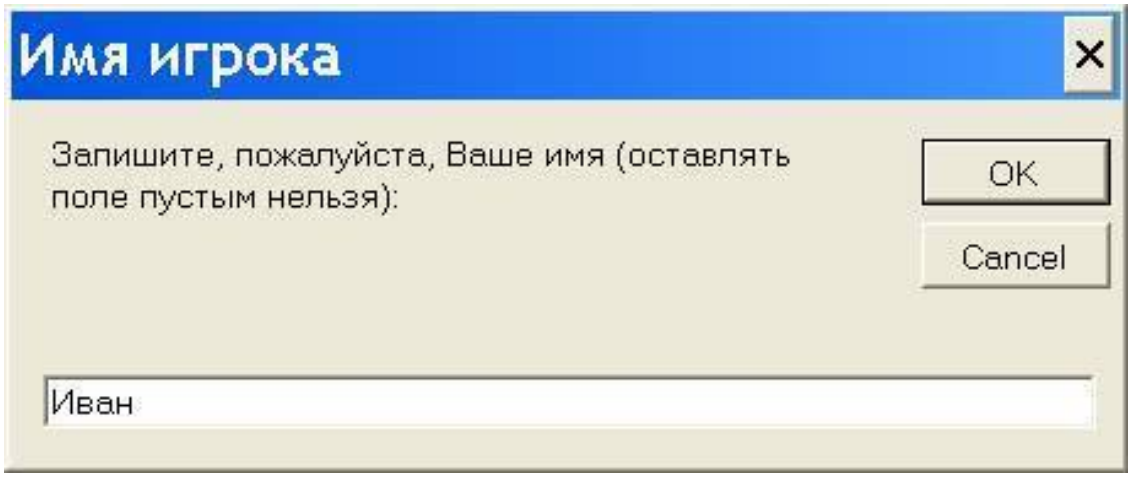



**Рис. 21.3.** Форма для записи имени игрока.

3. На форме в отдельных квадратах сетки 9 x 9 произвольным образом (при помощи г.с.ч. класса Random) появляются 3 разноцветных больших мяча, которые игрок может перемещать при помощи мыши, и 3 маленьких разноцветных мяча, которые размещает искусственный интеллект, чтобы помешать игроку построить прямую линию из мячей (так как в клетку с маленьким мячом большой мяч уже нельзя разместить).

Звучит мелодия начала игры, и пошёл отсчёт времени игры в секундах (рис. 21.4).

4. Если из трёх больших мячей есть мячи одинакового цвета, то игрок при помощи указателя мыши быстро выбирает (щёлкает) тот мяч, который он желает переместить к другому мячу (или мячам) того же цвета.

Если из трёх больших мячей нет мячей одинакового цвета, то игрок при помощи указателя мыши быстро выбирает (щёлкает) тот мяч, который он желает переместить в другой (более удобный, по его мнению) квадрат сетки.


**Рис. 21.4.** 3 больших и 3 маленьких мяча, отсчёт времени игры в секундах.

Мяч, по которому игрок щёлкнул, начинает пульсировать (и игрок видит, какой же мяч он щёлкнул).

Игрок может передумать и щёлкнуть другой мяч. После этого предыдущий мяч перестаёт пульсировать, и начинает пульсировать тот мяч, который игрок щёлкнул последним.

Ниже надписи “Следующие мячи:” игрок видит 3 мяча, которые появятся на поле после его щелчка, и опытный игрок учитывает эти мячи в своих прогнозах.

Теперь игрок щёлкает на тому пустому квадрату, в который должен переместиться мяч, чтобы образовать прямую линию из мячей одинакового цвета.

Мяч перемещается в эту клетку.

Отметим, что игрок может разместить мяч и не вплотную к мячу того же цвета, а через один или несколько пустых квадратов, чтобы прогнозировать сборку как можно более длинной линии из мячей одинакового цвета (так как на место уже имеющихся в сетке мячей компьютер с искусственным интеллектом не может размещать другие мячи). Однако игрок должен помнить, что компьютер со своим искусственным интеллектом (для данной игры) может “разгадать” замысел игрока и разместить свой маленький мяч в пустой квадрат между большими мячами игрока (тем самым, разорвав сплошную линию из мячей игрока).

5. После этого искусственный интеллект размещает в пустующие клетки следующие 3 больших мяча произвольных цветов, а 3 маленьких блокирующих мяча переносит в другие квадраты (чтобы помешать игроку построить прямую линию из 5 и более мячей), рис. 21.5.

Ниже надписи “Следующие мячи:” компьютер с искусственным интеллектом размещает 3 мяча, которые появятся на поле после перемещения игроком мяча из одного квадрата в другой.

6. Аналогично игрок снова щёлкает по выбранному им мячу и по той клетке, в которую мяч перемещается. После этого искусственный интеллект снова размещает в пустующие клетки следующие 3 больших мяча произвольных цветов, а 3 маленьких блокирующих мяча переносит в другие квадраты (чтобы помешать игроку построить прямую линию из 5 и более мячей).


**Рис. 21.5.** Добавляются ещё 3 больших мяча, а 3 маленьких мяча – в новых квадратах.


**Рис. 21.6.** Игрок подготовил вертикаль из 5 чёрных мячей с пустым 6-м квадратом.

7. На рис. 21.6 показано, как игрок подготовил вертикаль из 5 чёрных мячей с пустым 6-м квадратом.

8. На следующем ходе игрок переносит в этом 6-й пустой квадрат чёрный мяч, тем самым собрав вертикальную линию из 6 чёрных мячей.

Как только игрок соберёт горизонтальную, вертикальную или диагональную прямую линию из 5 и более мячей одинакового цвета, игроку начисляются очки (по 100 очков за каждый собранный в линию мяч), а линия из собранных мячей исчезает, освобождая квадраты, в которые искусственный интеллект далее будет размещать новые мячи.

На рис. 21.7 игроку начислено 600 очков.

9. По такой схеме игрок быстро щёлкает мышью по мячам, стараясь за отведённое время собрать как можно больше линий из мячей одинакового цвета.

Искусственный интеллект же периодически (после каждого второго щелчка игрока и следующего за ним перемещения пульсирующего мяча в новую клетку) уничтожает линию из 5 и более мячей (если такая линия собрана игроком), начисляет очки, дополняет квадраты 3 новыми разноцветными мячами (произвольным образом), а 3 маленьких блокирующих мяча переносит в другие квадраты (чтобы помешать игроку построить прямую линию из 5 и более мячей)..


**Рис. 21.7.** Игроку начислено 600 очков.

10. По окончании игры (например, по окончании заданного времени) игрок выбирает в меню Очки команду “Показать средние” очки.

Появляется стандартная панель MessageBox.Show с выходной информацией (рис. 21.8):

Очки:

Время:

Среднее значение: (очков/сек).

11. По среднему значению очков в секунду определяется победитель в игре.

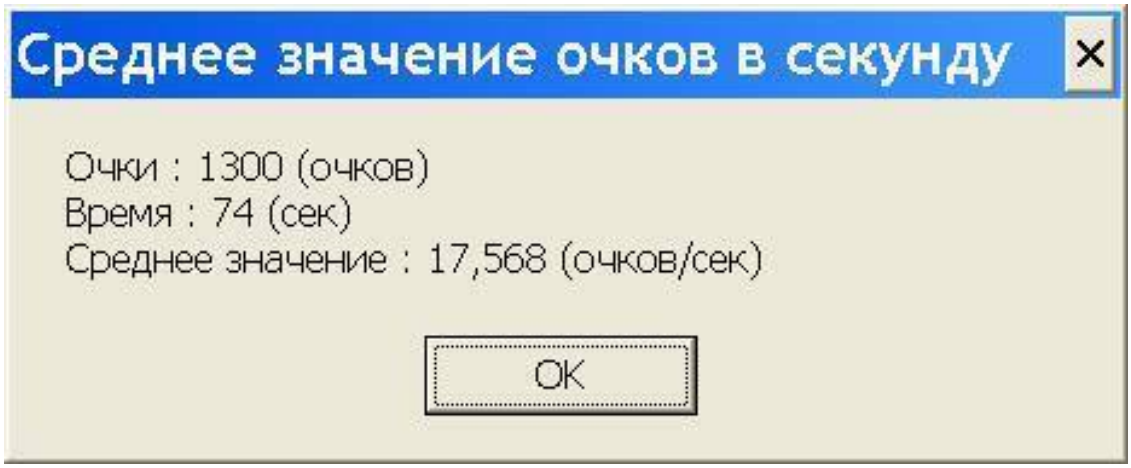



**Рис. 21.8.** Панель MessageBox.Show с выходной информацией.

12. Таким образом, после начала игры идёт отсчёт времени при помощи таймера.

Для каждого сеанса (попытки) игры одного или нескольких игроков задано определённое время, в данном примере, 60 секунд, по истечении которого звучит мелодия файла win.wav.

Игрок прекращает щёлкать мышью и смотрит на заработанные им очки.

13. Для начала новой попытки игрок снова щёлкает команду Новая в меню Игра.

14. Для закрытия игры следует в меню Игра выбрать команду Выход (или на форме щёлкнуть значок Close).

На основании этих правил можно сформулировать другие правила игры с использованием искусственного интеллекта, и любые правила ввести в справочную форму игры, которая появится после выбора команды Справка в меню Помощь по разработанной нами ранее (или в книгах с сайта ZharkovPress.ru) методике.

## 21.3. Создание проекта

Создаём проект по обычной схеме: в VS в панели New Project в окне Project types выбираем тип проекта Visual Basic, Windows, в окне Templates выделяем шаблон Windows Forms Application, в окне Name записываем имя проекта Line и щёлкаем OK. Создаётся проект, появляется форма Form1 в режиме проектирования (рис. 21.9). Оставляем по умолчанию или проектируем форму, как подробно описано в параграфе “Методика проектирования формы”. За маркер увеличиваем размеры формы таким образом, чтобы в панели Properties (для Form1) в свойстве Size были значения, например, 880; 630. Устанавливаем белый цвет Window для фона формы в свойстве BackColor.

Для задания режимов и управления игрой воспользуемся каким-либо элементом управления или компонентом. Как и выше, с панели инструментов Toolbox переносим на форму элемент управления MenuStrip и щёлкаем по нему (ниже формы в режиме проектирования). На форме Form1 появляются окна с надписью Type Here (Печатайте здесь), в которые записываем команды сначала на английском языке:

File, New Game, Save, Load, Exit;

Score, Show Score, Calculate Avg;

Help, Help, About.

Теперь в панели Properties (для каждой команды) в свойстве Text изменяем английские команды на соответствующие русские:

Игра, Новая, Сохранить, Загрузить, Выход;

Очки, Показать, Рассчитать средние;

Помощь, Справка, О программе (рис. 21.10 – 21.12).

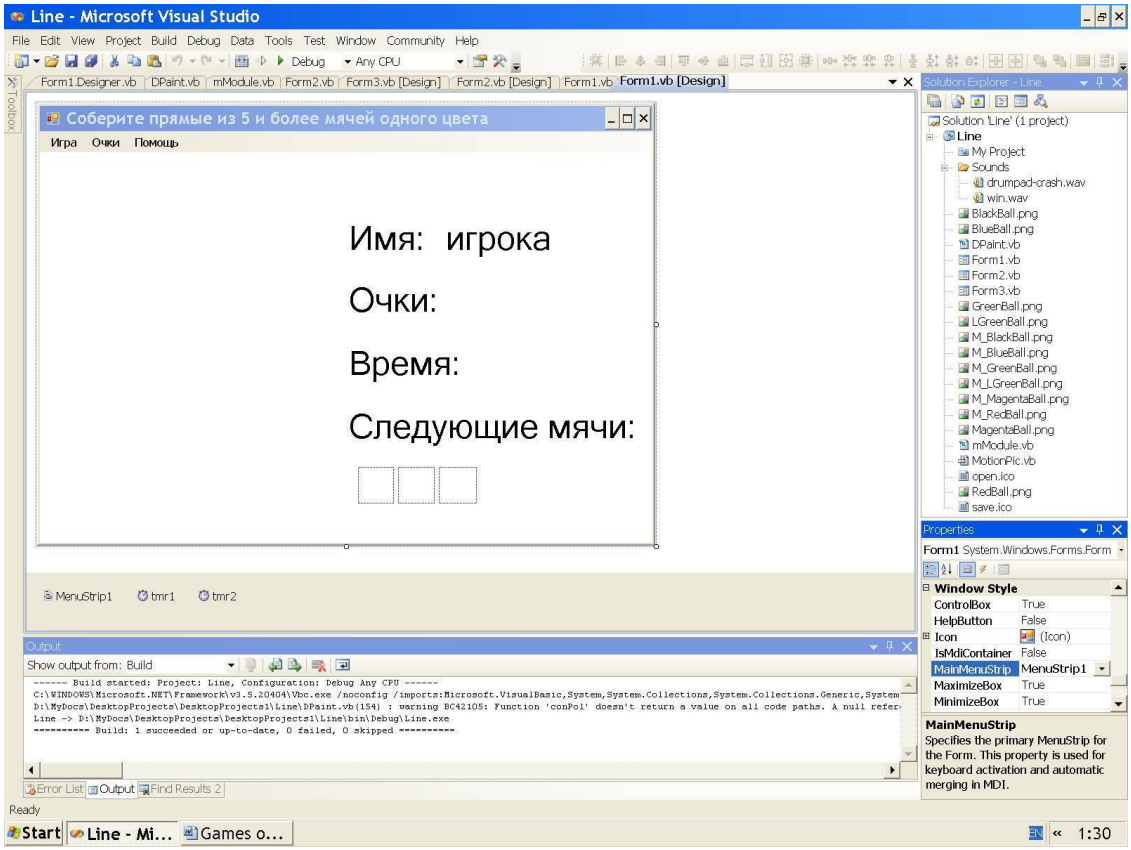



**Рис. 21.9.** Форма Form1 в режиме проектирования.

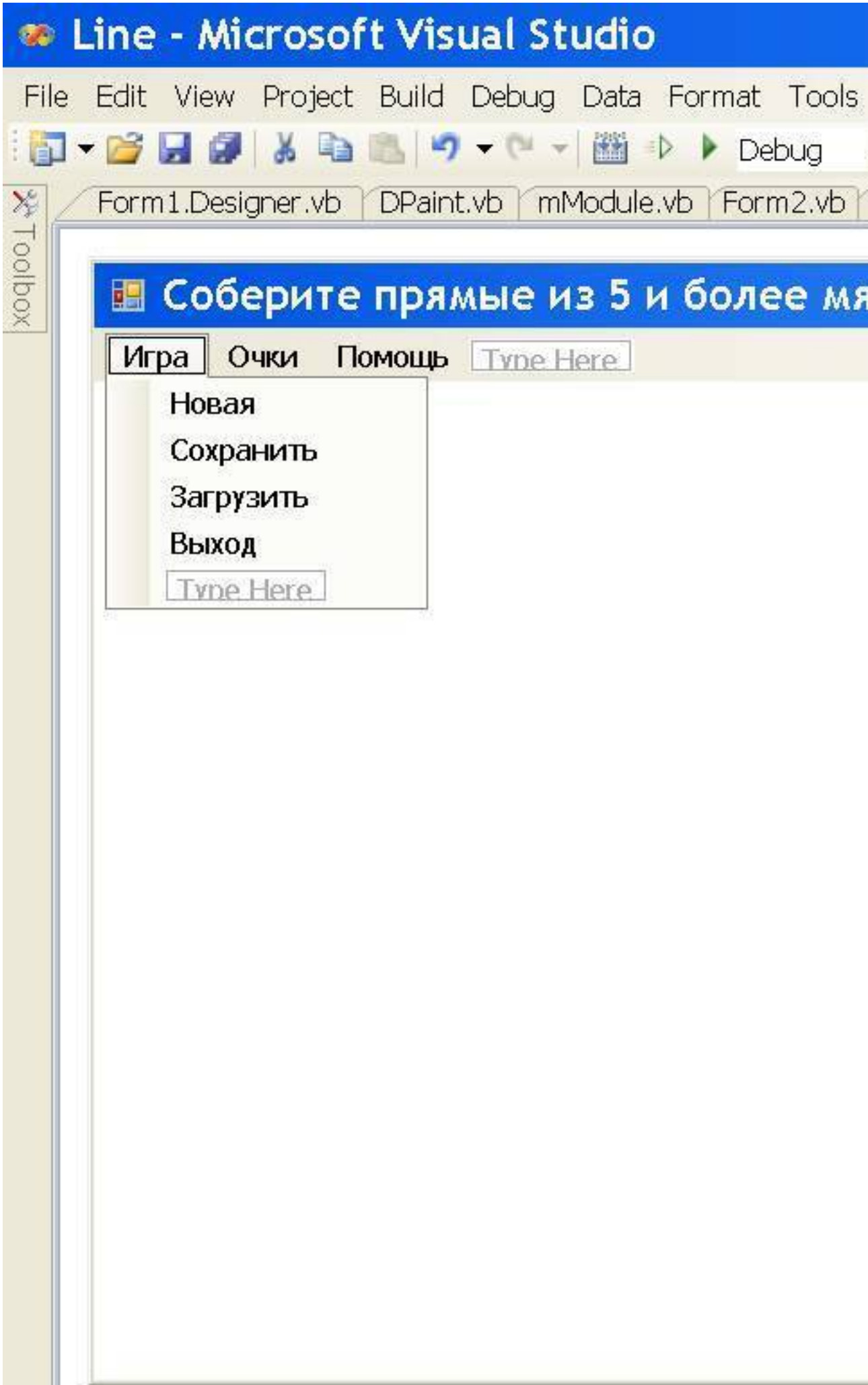

Line - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Tools
Debug
Form1.Designer.vb DPaint.vb mModule.vb Form2.vb
Toolbox
Соберите прямые из 5 и более мя
Игра Очки Помощь Type Here
Новая
Сохранить
Загрузить
Выход
Type Here

**Рис. 21.10.** Команды меню Игра. **Рис. 21.11.** Команды меню Очки. **Рис. 21.12.** Меню Помощь.

С панели инструментов Toolbox переносим на форму первую надпись Label. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 430; 124, в свойстве Name изменяем имя на lblName, в свойстве Font увеличиваем размер шрифта до 28, в свойстве Text записываем “Имя:”.

Правее на этой же горизонтали размещаем вторую надпись Label. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 568; 124, в свойстве Name изменяем имя на lblNameShow, в свойстве Font увеличиваем размер шрифта до 28, в свойстве Text записываем “игрока”.

Ниже размещаем третью надпись Label. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 430; 212, в свойстве Name изменяем имя на lblScore, в свойстве Font увеличиваем размер шрифта до 28, в свойстве Text записываем “Очки: ” (с пробелами для вывода очков в эти пробелы).

Ниже размещаем четвёртую надпись Label. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 430; 302, в свойстве Name изменяем имя на lblTime, в свойстве Font увеличиваем размер шрифта до 28, в свойстве Text записываем “Время: ” (с пробелами для вывода секунд, минут и часов в эти пробелы).

Ниже размещаем пятую надпись Label. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 430; 392, в свойстве Name изменяем имя на lblBallPreview, в свойстве Font увеличиваем размер шрифта до 28, в свойстве Text записываем “Следующие мячи:”.

Ниже размещаем первый графический элемент управления PictureBox. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 453; 477, в свойстве Size устанавливаем размеры 52; 52, в свойстве Name изменяем имя на picBallPre1.

Правее размещаем второй элемент PictureBox. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 510; 477, в свойстве Size устанавливаем размеры 52; 52, в свойстве Name изменяем имя на picBallPre2.

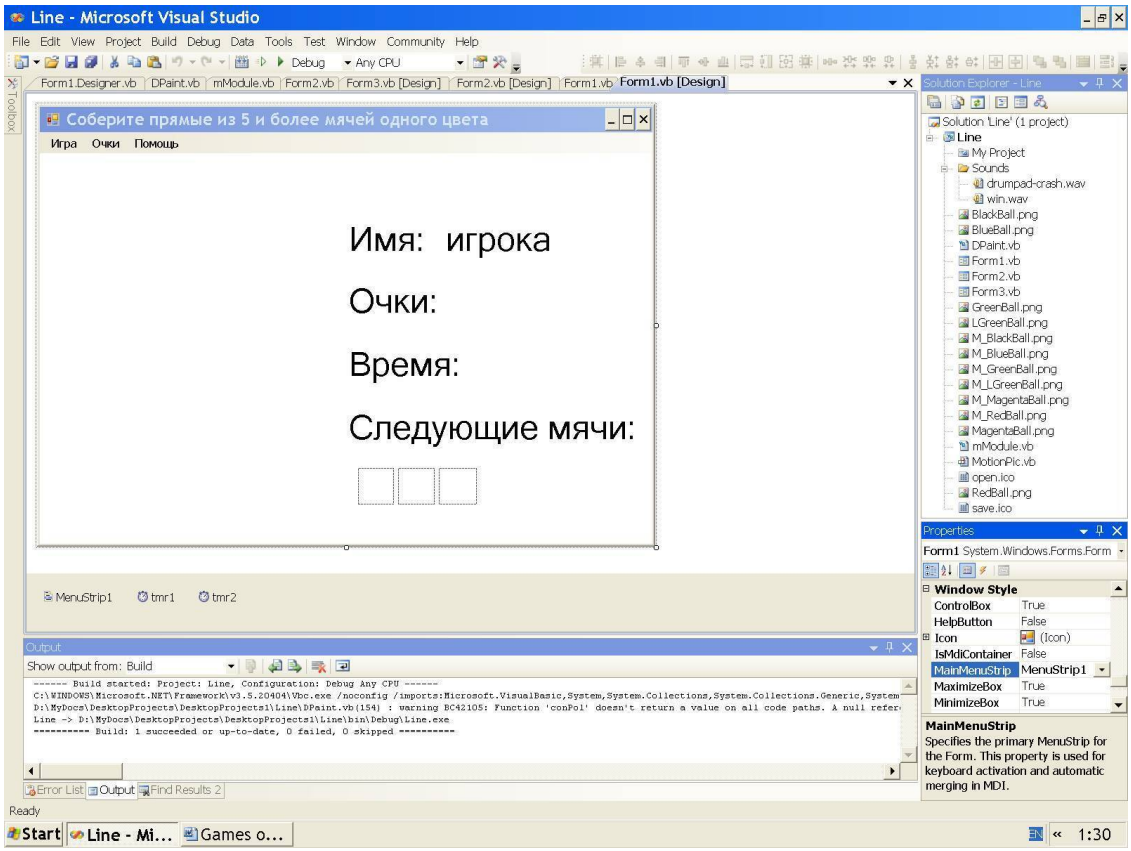
Правее размещаем третий элемент PictureBox. В панели Properties (для этого элемента) в свойстве Location устанавливаем координаты верхнего левого угла элемента 568; 477, в свойстве Size устанавливаем размеры 52; 52, в свойстве Name изменяем имя на picBallPre3.

Чтобы программа периодически через Interval времени дополняла поле игры новыми разноцветными мячами, с панели инструментов Toolbox переносим на форму (точнее, ниже формы) первый таймер Timer. В панели Properties (для этого таймера) в свойстве Name записываем имя tmr1, в свойстве Enabled оставляем заданное по умолчанию значение False, т.к. мы включим этот таймер в программе в нужном месте при помощи строки (Timer1.Enabled = True). А в свойстве Interval вместо заданных по умолчанию 100 миллисекунд задаём, например, значение 150 миллисекунд.

Чтобы после начала игры на форме шел отсчёт времени (Time), на форму переносим второй таймер Timer. В панели Properties (для этого таймера) в свойстве Name записываем имя tmr2, в свойстве Enabled оставляем заданное по умолчанию значение False, а в свойстве Interval задаём 1000 миллисекунд (равные 1 секунде).

Если в игре применяются звуковые файлы, то их целесообразно разместить в одной папке с именем, например, Sounds. Для добавления в проект этой папки, в панели Solution Explorer выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, New Folder, в поле появившегося значка папки записываем имя папки и нажимаем клавишу Enter.

Добавляем в эту папку звуковые файлы drumpad-crash.wav и win.wav по стандартной схеме: выполняем правый щелчок по имени этой папки, 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 с загруженными из Интернета файлами) и с нажатой клавишей Ctrl выделяем имена файлов и щёлкаем кнопку Add. В панели Solution Explorer мы увидим эти файлы (рис. 21.13). В панели Properties свойства этих файлов оставляем заданными по умолчанию.



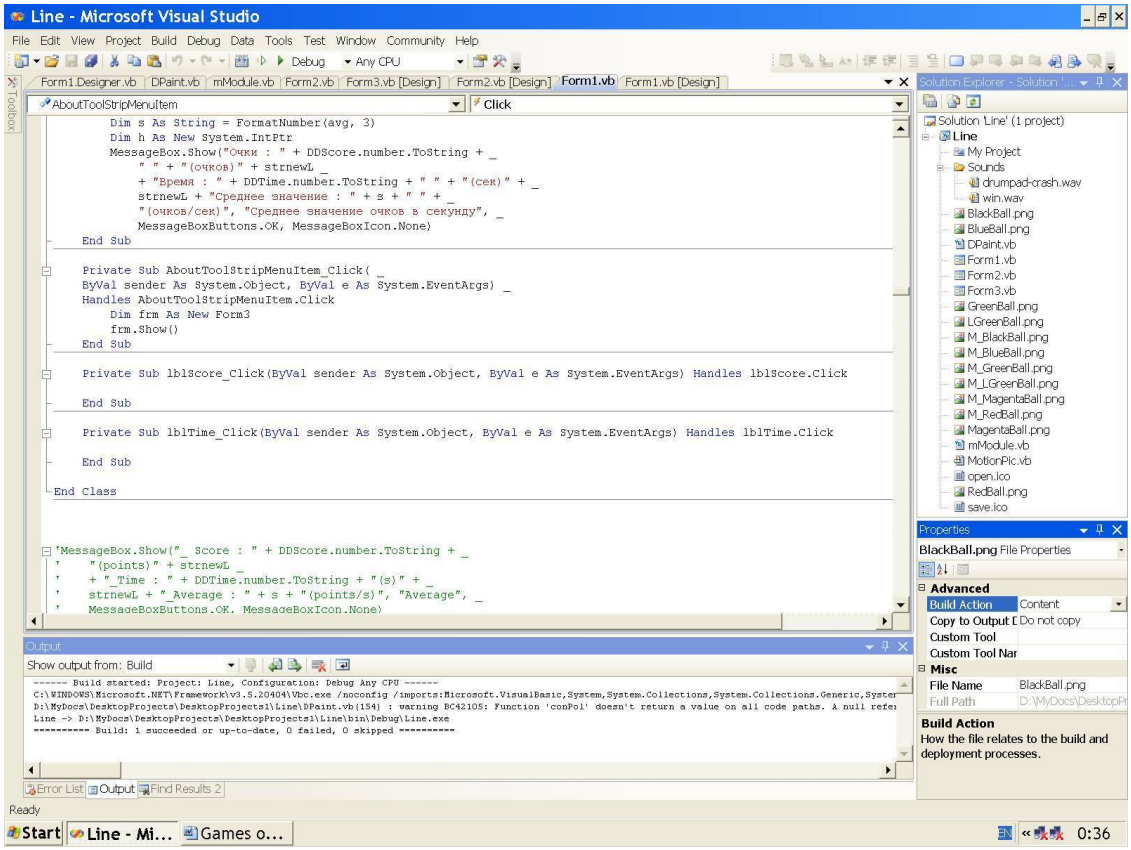



**Рис. 21.13.** Панель Solution Explorer. **Рис. 21.14.** Панель Properties.

Добавляем в проект графические файлы по стандартной схеме: в панели Solution Explorer выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, Existing Item (или Project, Add Existing Item), в панели Add Existing Item в окне “Files of type” выбираем “All Files”, в центральном окне находим (например, в папке с загруженными из Интернета файлами) и с нажатой клавишей Ctrl выделяем имена файлов и щёлкаем кнопку Add. В панели Solution Explorer мы увидим эти файлы (рис. 21.13). В панели Properties свойства этих файлов оставляем заданными по умолчанию (рис. 21.14).

Для ввода в проект новой формы (для таблицы с результатами игры) в меню Project выбираем Add Windows Form, в панели Add New Item оставляем заданные по умолчанию параметры и щёлкаем кнопку Add. В ответ VS выводит новую форму Form2 и добавляет в панель Solution Explorer новый пункт Form2.vb. Эта форма Form2, которая должна выводиться после выбора команды Показать в меню Очки, будет проектироваться программно согласно приведённому далее коду.

С панели инструментов Toolbox переносим на форму Form2 (точнее, ниже формы) первый таймер Timer. В панели Properties (для этого таймера) в свойстве Enabled оставляем заданное по умолчанию значение False, т.к. мы включим этот таймер в программе в нужном месте при помощи строки (Timer1.Enabled = True). А в свойстве Interval вместо заданных по умолчанию 100 миллисекунд задаём, например, значение 20 миллисекунд.

Аналогично переносим на форму Form2 второй таймер Timer. В панели Properties (для этого таймера) в свойстве Enabled оставляем заданное по умолчанию значение False, а в свойстве Interval задаём 40 миллисекунд.

Аналогично добавляем в проект третью форму Form3, которая будет выводиться после выбора команды Справка в меню Помощь. На форме Form3 размещаем какой-либо элемент

управления, например, TextBox, в который записываем справочную информацию, например, правила игры по описанной ранее методике.

Аналогично добавляется и проектируется следующая форма, которая должна выводиться после выбора команды “О программе” в меню Помощь.

Свойства всех элементов управления можно стандартно изменять, как описано ранее.

## 21.4. Код программы

Открываем файл Form1.vb (например, так: File, Open, File) и вверху записываем директиву для подключения требуемого пространства имен:

Imports System.IO 'Для класса StreamWriter.

Напомним, что эту строку можно и не записывать, но тогда нам придётся перед каждым классом записывать это пространство имён.

Теперь в классе Form1 нашего проекта записываем следующие переменные и методы.

**Листинг 21.1.** Переменные и методы.

```
Const intBaseX As Integer = 10
Const intBaseY As Integer = 120
Dim Rand As New Random
'***
Dim playerName As String
Dim playerScore As Double
Dim playerTime As Integer
Dim DDScore As New DPaint
Dim DDTime As New DPaint
'***
Dim intFlag As Integer = -1
Dim flagMadeNew = 0
Dim posMoveTo As Integer
Dim MPBoxes(80) As MotionPic
Dim ThreeBI(2) As Integer
Dim ThreeBP(2) As Integer
Dim prePic(2) As PictureBox
Private Sub InitBoard(ByVal plName As String, _
ByVal plScore As Double, ByVal plTime As Integer)
playerName = plName
playerScore = plScore
playerTime = plTime
If flagMadeNew = 0 Then
Dim i As Integer
Dim intX = intBaseX + 2
Dim intY = intBaseY + 2
For i = 0 To 80
Dim MP As New MotionPic(New Size(36, 36), _
New Point(intX, intY))
MP.SizeMode = PictureBoxSizeMode.StretchImage
intX += 45
If (i + 1) Mod 9 = 0 Then
intY += 45
```

```
intX = intBaseX + 2
End If
AddHandler MP.Click, AddressOf Ball_Click
MPBoxes(i) = MP
Next
Me.Controls.AddRange(MPBoxes)
DDScore.width = lblScore.Height / 2 – 6
DDScore.thick = DDScore.width / 4
DDScore.position = New Point(lblScore.Width – _
(DDScore.width + 2) * 9, lblScore.Height / 2)
DDTime.width = lblTime.Height / 2 – 6
DDTime.thick = DDTime.width / 4
DDTime.position = New Point(lblTime.Width – _
(DDTime.width + 2) * 9, lblTime.Height / 2)
AddHandler lblScore.Paint, AddressOf LabelScore_Paint
lblScore.Refresh()
AddHandler lblTime.Paint, AddressOf LabelTime_Paint
lblTime.Refresh()
For i = 0 To 2
prePic(i) = New PictureBox
prePic(i).SizeMode = PictureBoxSizeMode.StretchImage
prePic(i).Size = New Size(16, 16)
prePic(i).Visible = False
Me.Controls.Add(prePic(i))
AddHandler prePic(i).Click, AddressOf PrePic_Click
prePic(i).BringToFront()
Next
Else
ResetBoard()
End If
lblNameShow.Text = playerName
If playerName.Length > 8 Then
lblNameShow.Text += " "
tmr1.Enabled = True
End If
tmr2.Enabled = True
DDScore.number = plScore
lblScore.Refresh()
DDTime.number = plTime
lblTime.Refresh()
PreShow()
End Sub
Private Sub FindSol(ByVal i As Integer)
If MPBoxes(i).Tag <> "" Or MPBoxes(i).Tag = "Here" Then
Return
Else
MPBoxes(i).Tag = "Here"
End If
Select Case TestABox(i)
```

```
Case 1
FindSol(1)
FindSol(9)
Case 2
FindSol(7)
FindSol(17)
Case 3
FindSol(71)
FindSol(79)
Case 4
FindSol(63)
FindSol(73)
Case 5
FindSol(i + 1)
FindSol(i + 9)
FindSol(i – 1)
Case 6
FindSol(i – 9)
FindSol(i – 1)
FindSol(i + 9)
Case 7
FindSol(i – 1)
FindSol(i – 9)
FindSol(i + 1)
Case 8
FindSol(i – 9)
FindSol(i + 1)
FindSol(i + 9)
Case Else
FindSol(i – 9)
FindSol(i + 9)
FindSol(i + 1)
FindSol(i – 1)
End Select
End Sub
Private Sub ResetAllTag()
For Each Pic As MotionPic In MPBoxes
If Pic.Tag = "Here" Then
Pic.Tag = ""
End If
Next
End Sub
Private Function TestABox(ByVal val As Integer)
Select Case val
Case 0 : Return 1
Case 8 : Return 2
Case 80 : Return 3
Case 72 : Return 4
Case 1 To 7 : Return 5
```

```
Case 73 To 79 : Return 7
Case 17, 26, 35, 44, 53, 62, 71 : Return 6
Case 9, 18, 27, 36, 45, 54, 63 : Return 8
Case Else : Return 0
End Select
End Function

'Serious trouble happened – think more
Private Function GiveThreeBalls() As Boolean
If ThreeBI(1) = -1 Then 'Review for Game over
Return False
Else
For i As Integer = 0 To 2
If ThreeBI(i) = -1 Then
Exit For
Else
If MPBoxes(ThreeBP(i)).MPState = BallState. _
NO_BALL And ThreeBP(i) <> posMoveTo Then
MPBoxes(ThreeBP(i)).Init(ThreeBI(i))
CalWin(ThreeBP(i))
End If
End If
Next
End If
RandomThreeBalls()
PreShow()
Return True
End Function
Private Function IsFullBoard() As Boolean
Dim i As Integer
For Each Pic As PictureBox In MPBoxes
If MPBoxes(i).MPState <> BallState.NO_BALL Then
i += 1
End If
Next
If i = 81 Then
Return True
Else
Return False
End If
End Function
Private Sub RandomThreeBalls()
Dim ArrL As New ArrayList
Dim i As Integer
Dim pos As Integer
Dim ind As Integer
For i = 0 To 80
If MPBoxes(i).MPState = BallState.NO_BALL Or _
MPBoxes(i).MPState = BallState.DESTROYING_BALL Then
```

```
ArrL.Add(i)
End If
Next
For i = 0 To IIf(ArrL.Count > 2, 2, ArrL.Count – 1)
pos = Rand.Next(0, ArrL.Count)
pos = CInt(ArrL(pos))
ArrL.Remove(pos)
ThreeBP(i) = pos
ind = Rand.Next(0, 12)
ind = (ind \ 2) * 2
ThreeBI(i) = ind
Next
For j As Integer = i To 2
ThreeBI(j) = -1
ThreeBP(j) = -1
Next
End Sub
'#Region "Check for Calculate Score"
Private Function CheckHor(ByVal pos As Integer) As Integer
Dim type As Integer = MPBoxes(pos).MPIndex
Dim i As Integer = (pos \ 9) * 9
Dim count As Integer
Dim startpos As Integer = i
Dim endpos As Integer = i
While i < (pos \ 9) * 9 + 9
If MPBoxes(i).MPIndex = type Then
endpos += 1
count = endpos – startpos
Else
If count > 4 Then
While MPBoxes(pos).MPState = _
BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j).Destroy()
Next
Return count
End If
If i >= (pos \ 9) * 9 + 5 Then
Return count
End If
startpos = i + 1
endpos = i + 1
End If
i += 1
End While
If count > 4 Then
While MPBoxes(pos).MPState = BallState.ZOOMING_BALL
```

```
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j).Destroy()
Next
Return count
End If
End Function
Private Function CheckVer(ByVal pos As Integer) As Integer
Dim type As Integer = MPBoxes(pos).MPIndex
Dim i As Integer = pos Mod 9
Dim count As Integer
Dim startpos As Integer = i
Dim endpos As Integer = i
While i < (pos Mod 9) + 73
If MPBoxes(i).MPIndex = type Then
endpos += 9
count = (endpos – startpos) / 9
Else
If count > 4 Then
While MPBoxes(pos).MPState = _
BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j * 9).Destroy()
Next
Return count
End If
If i >= (pos Mod 9) + 36 Then
Return count
End If
startpos = i + 9
endpos = i + 9
End If
i += 9
End While
If count > 4 Then
While MPBoxes(pos).MPState = BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j * 9).Destroy()
Next
Return count
End If
End Function
Private Function CheckLR(ByVal pos As Integer) As Integer
If pos = 5 Or pos = 6 Or pos = 7 Or pos = 8 Or pos = 15 _
```

```
Or pos = 16 Or pos = 17 _
Or pos = 25 Or pos = 26 Or pos = 35 Or pos = 45 _
Or pos = 54 Or pos = 55 _
Or pos = 63 Or pos = 64 Or pos = 65 Or pos = 72 _
Or pos = 73 Or pos = 74 Or pos = 75 Then
Return 0
End If
Dim type As Integer = MPBoxes(pos).MPIndex
Dim i As Integer = pos Mod 10
i = IIf(i = 8, 18, IIf(i = 7, 27, IIf(i = 6, 36, i)))
Dim count As Integer
Dim startpos As Integer = i
Dim endpos As Integer = i
Dim tempi As Integer = i + 1
Dim temp As Integer
If i < 9 Then
temp = 9 – i
Else
temp = 9 – (i \ 9)
End If
While i < tempi + (temp – 1) * 10
If MPBoxes(i).MPIndex = type Then
endpos += 10
count = (endpos – startpos) \ 10
Else
If count > 4 Then
While MPBoxes(pos).MPState = _
BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j * 10).Destroy()
Next
Return count
End If
If i >= pos + 40 Then
Return count
End If
startpos = i + 10
endpos = i + 10
End If
i += 10
End While
If count > 4 Then
While MPBoxes(pos).MPState = BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j * 10).Destroy()
```

```
Next
Return count
End If
End Function
Private Function CheckRL(ByVal pos As Integer) As Integer
If pos = 0 Or pos = 1 Or pos = 2 Or pos = 3 Or pos = 9 _
Or pos = 10 Or pos = 11 _
Or pos = 18 Or pos = 19 Or pos = 27 Or pos = 53 _
Or pos = 61 Or pos = 62 _
Or pos = 69 Or pos = 70 Or pos = 71 Or pos = 77 _
Or pos = 78 Or pos = 79 Or pos = 80 Then
Return 0
End If
Dim type As Integer = MPBoxes(pos).MPIndex
Dim i As Integer = pos Mod 8
If i = 0 Then
i = 8
ElseIf i < 4 Then
i = (i + 1) * 8 + i
ElseIf pos \ 8 >= 5 Then
i = 45
End If
Dim count As Integer
Dim startpos As Integer = i
Dim endpos As Integer = i
Dim tempi As Integer = i + 1
Dim temp As Integer
If i < 9 Then
temp = i + 1
Else
temp = 9 – (i \ 8)
End If
While i < tempi + temp * 8
If MPBoxes(i).MPIndex = type Then
endpos += 8
count = (endpos – startpos) \ 8
Else
If count > 4 Then
While MPBoxes(pos).MPState = _
BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j * 8).Destroy()
Next
Return count
End If
If i >= pos + 32 Then
Return count
```

```
End If
startpos = i + 8
endpos = i + 8
End If
i += 8
End While
If count > 4 Then
While MPBoxes(pos).MPState = BallState.ZOOMING_BALL
Application.DoEvents()
End While
For j As Integer = 0 To count – 1
MPBoxes(startpos + j * 8).Destroy()
Next
Return count
End If
End Function
Private Function CalWin(ByVal pos As Integer) As Integer
Dim point As Integer = CheckHor(pos)
If point < 4 Then
point = CheckVer(pos)
End If
If point < 4 Then
point = CheckLR(pos)
End If
If point < 4 Then
point = CheckRL(pos)
End If
If point > 4 Then
Dim dpoint As Double = point * 100 + (dpoint \ 6) * 100
Dim n As Double = DDScore.number + dpoint
For i As Double = DDScore.number To n Step 10
DDScore.number = i
lblScore.Refresh()
Next
DDScore.number = n
Return point
Else
Return 0
End If
End Function
'#Region "Ball Event And Paint Board"
Private Sub Ball_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs)
CType(sender, MotionPic).Jump()
If CType(sender, MotionPic).MPState <> _
BallState.NO_BALL Then
If intFlag <> -1 Then
MPBoxes(intFlag).Jump()
End If
```

```
'intFlag = MPBoxes.IndexOf(MPBoxes, sender)
'Исправляем предупреждение:
intFlag = Array.IndexOf(MPBoxes, sender)
ElseIf intFlag <> -1 Then
Dim tempS As String = MPBoxes(intFlag).Tag
MPBoxes(intFlag).Tag = ""
FindSol(intFlag)
If sender.tag = "Here" Then
'posMoveTo = MPBoxes.IndexOf(MPBoxes, sender)
'Исправляем предупреждение:
posMoveTo = Array.IndexOf(MPBoxes, sender)
If posMoveTo = ThreeBP(0) Then
prePic(0).SendToBack()
ElseIf posMoveTo = ThreeBP(1) Then
prePic(1).SendToBack()
ElseIf posMoveTo = ThreeBP(2) Then
prePic(2).SendToBack()
End If
CType(sender, MotionPic).Init(MPBoxes(intFlag). _
MPIndex)
MPBoxes(intFlag).Destroy()
While MPBoxes(intFlag).MPState = _
BallState.DESTROYING_BALL
Application.DoEvents()
End While
If CalWin(posMoveTo) = 0 Then
If GiveThreeBalls() = False Then
playerScore = DDScore.number
Dim frm As Form2 = New Form2
frm.AddPlayer() = _
New Player(playerName, playerScore)
frm.Show()
frm.PlashScreen()
frm.drawTable()
ResetBoard()
End If
End If
PreShow()
intFlag = -1
Else
MPBoxes(intFlag).Tag = tempS
End If
ResetAllTag()
End If
End Sub
Private Sub ResetBoard()
ThreeBI(0) = -1
tmr1.Enabled = False
tmr2.Enabled = False
```

```
playerScore = 0
playerTime = 0
DDScore.number = 0
DDTime.number = 0
Me.Refresh()
PreShow()
lblNameShow.Text = ""
picBallPre1.Image = Nothing
For i As Integer = 0 To 80
MPBoxes(i).Reset()
Next
End Sub
Private Sub DrawBoard(ByVal sender As Object, _
ByVal e As System.Windows.Forms.PaintEventArgs) _
Handles MyBase.Paint
Dim g As Graphics = e.Graphics
'Рисуем сетку линиями красного (Red) цвета:
Dim p1 As New Pen(Color.Red)
Dim p2 As New Pen(Color.Black)
For i As Integer = 0 To 9
g.DrawLine(p1, intBaseX + 45 * i – 4, intBaseY – 5, _
intBaseX + 45 * i – 4, intBaseY + 45 * 9 – 5)
g.DrawLine(p2, intBaseX + 45 * i – 3, intBaseY – 4, _
intBaseX + 45 * i – 3, intBaseY + 45 * 9 – 4)
g.DrawLine(p1, intBaseX – 4, intBaseY + 45 * i – 5, _
intBaseX + 45 * 9 – 4, intBaseY + 45 * i – 5)
g.DrawLine(p2, intBaseX – 3, intBaseY + 45 * i – 4, _
intBaseX + 45 * 9 – 3, intBaseY + 45 * i – 4)
Next
End Sub
Private Sub LabelScore_Paint(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.PaintEventArgs)
DDScore.showNumber(e.Graphics)
End Sub
Private Sub LabelTime_Paint(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.PaintEventArgs)
DDTime.showTime(e.Graphics)
End Sub
Private Sub tmr1_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles tmr1.Tick
Dim s As String = lblNameShow.Text
lblNameShow.Text = s.Substring(1) + s.Substring(0, 1)
End Sub
'Счётчик секунд, который обнуляем в начале каждой игры
'в методе NewGame:
Dim secondCounter As Integer
'Время, через которое звучит мелодия
'возможного окончания игры:
Dim EndGameTime As Integer = 60
```

```
Private Sub tmr2_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles tmr2.Tick
DDTime.number += 1
lblTime.Refresh()
'Счётчик секунд:
secondCounter = secondCounter + 1
'Мелодия окончания игры:
If secondCounter = EndGameTime Then
My.Computer.Audio.Play("..\..\Sounds\win.wav", _
AudioPlayMode.Background)
End If
End Sub
'#Region "Preview Balls"
Private Sub PreShow()
Dim col, row As Integer
For i As Integer = 0 To 2
If ThreeBI(i) = -1 Then
prePic(i).Visible = False
Else
prePic(i).Visible = True
prePic(i).Image = Image.FromFile(ImgList(ThreeBI(i)))
col = ThreeBP(i) Mod 9
row = ThreeBP(i) \ 9
prePic(i).Location = New Point(col * 45 + _
intBaseX + (38 – prePic(i).Width) / 2, _
row * 45 + intBaseY + (38 – prePic(i).Height) / 2)
prePic(i).Visible = True
prePic(i).BringToFront()
End If
If ThreeBI(0) <> -1 Then
picBallPre1.Visible = True
picBallPre1.Image = _
Image.FromFile(ImgList(ThreeBI(0)))
End If
If ThreeBI(1) <> -1 Then
picBallPre2.Visible = True
picBallPre2.Image = _
Image.FromFile(ImgList(ThreeBI(1)))
Else
picBallPre2.Visible = False
End If
If ThreeBI(2) <> -1 Then
picBallPre3.Visible = True
picBallPre3.Image = _
Image.FromFile(ImgList(ThreeBI(2)))
Else
picBallPre3.Visible = False
End If
Next
```

```
End Sub
Private Sub PrePic_Click(ByVal sender As Object, _
ByVal e As System.EventArgs)
Dim i As Integer = Array.IndexOf(prePic, sender)
Dim MP As MotionPic = MPBoxes(ThreeBP(i))
Call Ball_Click(MP, e)
End Sub
'#Region "Save and Load Game"
Private Sub SaveGame()
Dim strNewLine = Chr(13) + Chr(10)
Dim s As String = Nothing
playerScore = DDScore.number
playerTime = DDTime.number
s += "#Assignment Line" + strNewLine
s += playerName.ToString + strNewLine
s += playerScore.ToString + strNewLine
s += playerTime.ToString + strNewLine
For i As Integer = 0 To 2
s += ThreeBI(i).ToString + ";" + ThreeBP(i).ToString
If i < 2 Then
s += ","
End If
Next
s += strNewLine
For i As Integer = 0 To 80
s += MPBoxes(i).MPState.ToString + ";" + _
MPBoxes(i).MPIndex.ToString
If i < 80 Then
s += ","
End If
Next
Dim SW As StreamWriter = Nothing
Try
SW = New StreamWriter("LSF.vmt")
SW.Write(s)
Catch IOE As IOException
MessageBox.Show("Can't save File !", "Error", _
MessageBoxButtons.OK, MessageBoxIcon.Error)
Catch EX As Exception
MessageBox.Show("Some Error occurs while Saving" + _
strNewLine + "Error :" + EX.ToString, _
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
Finally
SW.Close()
End Try
End Sub
Private Sub LoadGame()
Dim strRead(4) As String
Dim strBigArr() As String
```

```
Dim strSmallArr() As String
Dim SR As StreamReader = Nothing
If Not File.Exists("LSF.vmt") Then
MessageBox.Show("Save File doesn't Exists", _
"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
Exit Sub
End If
Try
SR = New StreamReader("LSF.vmt")
If SR.ReadLine <> "#Assignment Line" Then
MessageBox.Show("Invalid Loaded File", "Error", _
MessageBoxButtons.OK, MessageBoxIcon.Error)
SR.Close()
Exit Sub
End If
For i As Integer = 0 To 4
strRead(i) = SR.ReadLine
Next
Catch IOE As IOException
MessageBox.Show("Can't load File !", "Error", _
MessageBoxButtons.OK, MessageBoxIcon.Error)
Catch EX As Exception
MessageBox.Show("Some Error occurs while Loading" + _
Chr(13) + Chr(10) + "Error :" + EX.ToString, _
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
Finally
SR.Close()
End Try
'Information Loaded
playerName = strRead(0)
playerScore = CDbl(strRead(1))
playerTime = CInt(strRead(2))
ReDim strBigArr(2)
ReDim strSmallArr(1)
strBigArr = strRead(3).Split(",")
For i As Integer = 0 To 2
strSmallArr = strBigArr(i).Split(";")
ThreeBI(i) = CInt(strSmallArr(0))
ThreeBP(i) = CInt(strSmallArr(1))
Next
InitBoard(playerName, playerScore, playerTime)
ReDim strBigArr(80)
strBigArr = strRead(4).Split(",")
For i As Integer = 0 To 80
strSmallArr = strBigArr(i).Split(";")
If CInt(strSmallArr(0)) <> BallState.NO_BALL Then
MPBoxes(i).Init(CInt(strSmallArr(1)))
End If
Next
```

```
End Sub
```

В меню Игра дважды щёлкаем по команде Новая (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.2.** Метод-обработчик выбора команды.

```
Private Sub NewGameToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles NewGameToolStripMenuItem.Click
'Обнуляем счётчик секунд:
secondCounter = 0
'Мелодия начала игры:
My.Computer.Audio.Play("..\..\Sounds\drumpad-crash.wav")
Dim plName As String = Nothing
While Trim(plName) = ""
plName = InputBox("Запишите, пожалуйста, Ваше имя " + _
"(оставлять поле пустым нельзя):", "Имя игрока")
End While
InitBoard(plName, 0, 0)
RandomThreeBalls()
GiveThreeBalls()
flagMadeNew = 1
SaveToolStripMenuItem.Enabled = True
End Sub
```

В меню Игра дважды щёлкаем по команде Сохранить (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.3.** Метод-обработчик выбора команды.

```
Private Sub SaveToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles SaveToolStripMenuItem.Click
SaveGame()
End Sub
```

В меню Игра дважды щёлкаем по команде Загрузить (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.4.** Метод-обработчик выбора команды.

```
Private Sub LoadToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles LoadToolStripMenuItem.Click
If flagMadeNew = 1 Then
ResetBoard()
End If
LoadGame()
flagMadeNew = 1
SaveToolStripMenuItem.Enabled = True
End Sub
```

В меню Игра дважды щёлкаем по команде Выход (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.5.** Метод-обработчик выбора команды.

```
Private Sub ExitToolStripMenuItem_Click( _
```

```
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles ExitToolStripMenuItem.Click
Me.Close()
End Sub
```

В меню Очки дважды щёлкаем по команде Показать (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.6.** Метод-обработчик выбора команды.

```
Private Sub ShowScorToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles ShowScorToolStripMenuItem.Click
Dim frm As Form2 = New Form2
frm.Show()
frm.drawTable()
End Sub
```

В меню Очки дважды щёлкаем по команде “Рассчитать средние” (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.7.** Метод-обработчик щелчка по элементу.

```
Private Sub CalculateAvgToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles CalculateAvgToolStripMenuItem.Click
Dim strnewL = Chr(13) + Chr(10)
Dim avg As Double = DDScore.number / DDTime.number
Dim s As String = FormatNumber(avg, 3)
Dim h As New System.IntPtr
MessageBox.Show("Очки : " + DDScore.number.ToString + _
" " + "(очков)" + strnewL _
+ "Время : " + DDTime.number.ToString + " " + "(сек)" + _
strnewL + "Среднее значение : " + s + " " + _
"(очков/сек)", "Среднее значение очков в секунду", _
MessageBoxButtons.OK, MessageBoxIcon.None)
End Sub
```

В меню Помощь дважды щёлкаем по команде “О программе” (для элемента управления MenuStrip). Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.8.** Метод-обработчик выбора команды.

```
Private Sub AboutToolStripMenuItem_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles AboutToolStripMenuItem.Click
Dim frm As New Form3
frm.Show()
End Sub
```

Схема записи и вывода справочной информации, например, с правилами игры после выбора команды Справка (для элемента управления MenuStrip) и после выбора других команд уже приводилась в наших предыдущих работах.

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

Теперь в наш проект добавляем новые файлы (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 DPaint.vb 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 21.9.** Новый файл.

```
Public Class DPaint
Private _number As String
Private _position As Point
Private _color As Color
Private _pen As Pen
Private _thick As Integer
Private _width As Integer
Private _brush As SolidBrush
#Region "Property Declaration"
Public Sub New()
'_color = Color.Yellow
'Для большей чёткости задаём красный цвет цифр для очков
'и времени:
_color = Color.Red
_brush = New SolidBrush(_color)
_number = 0
End Sub
Public Sub New(ByVal number As Integer)
Me.number = number
_color = Color.Yellow
_brush = New SolidBrush(_color)
End Sub
Public Property number() As Double
Get
Return CDbl(_number)
End Get
Set(ByVal Value As Double)
_number = Value.ToString
_number = _number.PadLeft(8, "0")
End Set
End Property
Public Property position() As Point
Get
Return _position
End Get
```

```
Set(ByVal Value As Point)
_position = Value
End Set
End Property
Public Property thick() As Integer
Get
Return _thick
End Get
Set(ByVal Value As Integer)
_thick = Value
End Set
End Property
Public Property width() As Integer
Get
Return _width
End Get
Set(ByVal Value As Integer)
_width = Value
End Set
End Property
#End Region
#Region "Show digital Number"
Private Function conPol(ByVal i As Integer) As Point()
Dim c, h As Integer
Dim x As Integer = _position.X
Dim y As Integer = _position.Y
Dim poly(3) As Point
Select Case i
Case 1
c = x
h = y – _width – 2
poly(0).X = c
poly(0).Y = h
poly(1).X = c + _width
poly(1).Y = h
poly(2).X = c + _width – _thick
poly(2).Y = h + _thick
poly(3).X = c + _thick
poly(3).Y = h + _thick
Return poly
Case 2
c = x + _width
h = y – _width – 1
poly(0).X = c
poly(0).Y = h
poly(1).X = c
poly(1).Y = h + _width
poly(2).X = c – _thick
poly(2).Y = h + _width – _thick / 2
```

```
poly(3).X = c – _thick
poly(3).Y = h + _thick
Return poly
Case 3
c = x + _width
h = y + 1
poly(0).X = c
poly(0).Y = h
poly(1).X = c
poly(1).Y = h + _width
poly(2).X = c – _thick
poly(2).Y = h + _width – _thick
poly(3).X = c – _thick
poly(3).Y = h + _thick / 2
Return poly
Case 4
c = x + _width
h = y + _width + 2
poly(0).X = c
poly(0).Y = h
poly(1).X = c – _width
poly(1).Y = h
poly(2).X = c – _width + _thick
poly(2).Y = h – _thick
poly(3).X = c – _thick
poly(3).Y = h – _thick
Return poly
Case 5
c = x
h = y + _width + 1
poly(0).X = c
poly(0).Y = h
poly(1).X = c
poly(1).Y = h – _width
poly(2).X = c + _thick
poly(2).Y = h – _width + _thick / 2
poly(3).X = c + _thick
poly(3).Y = h – _thick
Return poly
Case 6
c = x
h = y – 1
poly(0).X = c
poly(0).Y = h
poly(1).X = c
poly(1).Y = h – _width
poly(2).X = c + _thick
poly(2).Y = h – _width + _thick
poly(3).X = c + _thick
```

```
poly(3).Y = h – _thick / 2
Return poly
Case 7
ReDim poly(5)
c = x
h = y
poly(0).X = c
poly(0).Y = h
poly(1).X = c + _thick
poly(1).Y = h – _thick / 2
poly(2).X = c + _width – _thick
poly(2).Y = h – _thick / 2
poly(3).X = c + _width
poly(3).Y = h
poly(4).X = c + _width – _thick
poly(4).Y = h + _thick / 2
poly(5).X = c + _thick
poly(5).Y = h + _thick / 2
Return poly
End Select
End Function

Private Sub show(ByVal g As Graphics, ByVal led1 As Boolean, _
ByVal led2 As Boolean, ByVal led3 As Boolean, _
ByVal led4 As Boolean, ByVal led5 As Boolean, _
ByVal led6 As Boolean, ByVal led7 As Boolean)
led(g, 1, led1)
led(g, 2, led2)
led(g, 3, led3)
led(g, 4, led4)
led(g, 5, led5)
led(g, 6, led6)
led(g, 7, led7)
End Sub
Private Sub led(ByVal g As Graphics, ByVal led As Integer, _
ByVal sta As Boolean)
If (sta) Then
g.FillPolygon(_brush, conPol(led))
End If
End Sub
Private Sub showANum(ByVal g As Graphics, ByVal num As Integer)
Select Case num
Case 0
show(g, True, True, True, True, True, True, False)
Case 1
show(g, False, True, True, False, False, False, False)
Case 2
show(g, True, True, False, True, True, False, True)
Case 3
```

```
show(g, True, True, True, True, False, False, True)
Case 4
show(g, False, True, True, False, False, True, True)
Case 5
show(g, True, False, True, True, False, True, True)
Case 6
show(g, True, False, True, True, True, True, True)
Case 7
show(g, True, True, True, False, False, False, False)
Case 8
show(g, True, True, True, True, True, True, True)
Case 9
show(g, True, True, True, True, False, True, True)
End Select
End Sub
Public Sub showNumber(ByVal g As Graphics)
Dim tempnum As Integer
Dim tempPos As Point = _position
For i As Integer = 0 To _number.Length – 1
tempnum = _number.Substring(i, 1)
showANum(g, CInt(tempnum))
_position.X += _width + 2
Next
_position = tempPos
End Sub
#End Region
#Region "Show Time"
Private Sub show2Points(ByVal g As Graphics)
Dim r1 As Integer = _position.Y + _width / 2 – 2
Dim r2 As Integer = _position.Y – _width / 2 – 2
Dim c As Integer = _position.X + _width / 2 – 2
g.FillEllipse(_brush, c, r1, _width \ 3, _width \ 3)
g.FillEllipse(_brush, c, r2, _width \ 3, _width \ 3)
End Sub
Public Sub showTime(ByVal g As Graphics)
Dim num As Integer = CInt(_number)
Dim tempPos As Point = _position
Dim l As Integer = IIf(num = 3600, 8, 5)
Dim h As Integer = num \ 3600
Dim m As Integer = (num Mod 3600) \ 60
Dim s As Integer = num Mod 60
showANum(g, h \ 10)
_position.X += _width + 2
showANum(g, h Mod 10)
_position.X += _width + 2
show2Points(g)
_position.X += _width + 2
showANum(g, m \ 10)
_position.X += _width + 2
```

```
showANum(g, m Mod 10)
_position.X += _width + 2
show2Points(g)
_position.X += _width + 2
showANum(g, s \ 10)
_position.X += _width + 2
showANum(g, s Mod 10)
_position = tempPos
End Sub
#End Region
End Class
```

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 mModule.vb 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 21.10.** Новый файл.

```
Public Module mModule
Private s As String = Application.StartupPath & "..\..\"
'Public ImgList() As String = {s & "BlackBall.png",
's & "M_BlackBall.png", s & "BlueBall.png" _
', s & "M_BlueBall.png", s & "GreenBall.png", s &
'"M_GreenBall.png", s & "LGreenBall.png" _
', s & "M_LGreenBall.png", s & "MagentaBall.png", s &
'"M_MagentaBall.png" _
', s & "RedBall.png", s & "M_RedBall.png"}
'Исправляем ошибку:
Public ImgList() As String = _
{"..\..\BlackBall.png", "..\..\M_BlackBall.png", _
"..\..\BlueBall.png", "..\..\M_BlueBall.png", _
"..\..\GreenBall.png", "..\..\M_GreenBall.png", _
"..\..\LGreenBall.png", "..\..\M_LGreenBall.png", _
"..\..\MagentaBall.png", "..\..\M_MagentaBall.png", _
"..\..\RedBall.png", "..\..\M_RedBall.png"}
Public Structure Player
Private _PlayerName As String
Private _PlayerScore As String
Public Sub New(ByVal PlayerName As String, _
ByVal PlayerScore As String)
_PlayerName = PlayerName
_PlayerScore = PlayerScore
End Sub
Public Property PlayerName() As String
Get
Return _PlayerName
End Get
Set(ByVal Value As String)
_PlayerName = Value
End Set
```

```
End Property
Public Property PlayerScore() As String
Get
Return _PlayerScore
End Get
Set(ByVal Value As String)
_PlayerScore = Value
End Set
End Property
End Structure
End Module
```

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 MotionPic.vb 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 21.11.** Новый файл.

```
Option Strict On
Public Enum BallState
ZOOMING_BALL = -2
NO_BALL = -1
NORMAL_BALL = 0
JUMPING_BALL = 1
DESTROYING_BALL = 2
End Enum
Public Class MotionPic
Inherits System.Windows.Forms.PictureBox
Private myTimer As New System.Windows.Forms.Timer
Private picWidth As Integer
Private picHeight As Integer
Private picTop As Integer
Private picLeft As Integer
Private picState As Integer
Private picIndex As Integer
Private sign As Integer
#Region "Property Declaration"
Public Sub New(ByVal eSize As Size, ByVal eLocation As Point)
Me.Size = eSize
Me.Location = eLocation
picWidth = Me.Width
picHeight = Me.Height
picTop = Me.Top
picLeft = Me.Left
picState = BallState.NO_BALL
picIndex = -1
End Sub
Public ReadOnly Property MPState() As Integer
Get
Return picState
```

```
End Get
End Property
Public ReadOnly Property MPIndex() As Integer
Get
Return picIndex
End Get
End Property
Public Sub Init()
Me.Init(CInt(Rnd() * 12))
End Sub
#End Region
Public Sub Init(ByVal value As Integer)
picIndex = value
picState = BallState.ZOOMING_BALL
Dim i As Integer = ImgList(value).LastIndexOf("\")
Me.Tag = ImgList(value).Substring(i + 1, _
ImgList(value).Length – i – 5)
ZoomIn()
End Sub
Private Sub ZoomIn()
Me.Top = picTop + (picHeight – 4) \ 2
Me.Left = picLeft + (picWidth – 4) \ 2
Me.Width = 4
Me.Height = 4
Me.Image = Image.FromFile(ImgList(picIndex))
AddHandler myTimer.Tick, AddressOf TimerEventZoomIn
myTimer.Interval = 10
myTimer.Start()
End Sub
Private Sub TimerEventZoomIn(ByVal myObject As Object, _
ByVal myEventArgs As EventArgs)
If Me.Top > picTop And Me.Width < picWidth Then
Me.Top -= 2
Me.Left -= 2
Me.Width += 4
Me.Height += 4
Else
myTimer.Enabled = False
picState = BallState.NORMAL_BALL
RemoveHandler myTimer.Tick, AddressOf TimerEventZoomIn
End If
End Sub
Public Sub Jump()
If picState = BallState.NORMAL_BALL Then
sign = 1
picState = BallState.JUMPING_BALL
AddHandler myTimer.Tick, AddressOf TimerEventJump
myTimer.Interval = 20
myTimer.Start()
```

```
ElseIf picState = BallState.JUMPING_BALL Then
StopJump()
End If
End Sub
Public Sub StopJump()
If picState = BallState.JUMPING_BALL Then
picState = 0
myTimer.Enabled = False
RemoveHandler myTimer.Tick, AddressOf TimerEventJump
Me.Top = picTop
Me.Left = picLeft
Me.Height = picHeight
Me.Width = picWidth
End If
End Sub
Private Sub TimerEventJump(ByVal myObject As Object, _
ByVal myEventArgs As EventArgs)
Me.Height -= sign * 1
Me.Top = picTop + (Me.Height – picHeight) \ 4
If Me.Height = picHeight Or Me.Height <= 3 * picHeight / 4 Then
sign *= -1
End If
End Sub
Public Sub Destroy()
If picState = BallState.JUMPING_BALL Then
StopJump()
End If
picState = BallState.DESTROYING_BALL
AddHandler myTimer.Tick, AddressOf TimerEventDestroy
Me.Top = picTop + 1
Me.Left = picLeft + 1
Me.Width = picWidth – 2
Me.Height = picHeight – 2
myTimer.Interval = 10
myTimer.Start()
End Sub
Private Sub TimerEventDestroy(ByVal myObject As Object, _
ByVal myEventArgs As EventArgs)
If Me.Top > picTop And Me.Width > 0 Then
Me.Top += 2
Me.Left += 2
Me.Width -= 4
Me.Height -= 4
Else
Me.Image = Nothing
Me.Top = picTop
Me.Left = picLeft
Me.Width = picWidth
Me.Height = picHeight
```

```
myTimer.Enabled = False
Me.picState = BallState.NO_BALL
Me.picIndex = -1
Me.Tag = ""
RemoveHandler myTimer.Tick, AddressOf TimerEventDestroy
End If
End Sub
Public Sub Reset()
While Me.picState = BallState.DESTROYING_BALL
Application.DoEvents()
End While
If Me.picState = BallState.JUMPING_BALL Then
StopJump()
End If
While Me.picState = BallState.ZOOMING_BALL
Application.DoEvents()
End While
Me.Image = Nothing
Me.picIndex = -1
Me.picState = BallState.NO_BALL
Me.Tag = ""
End Sub
Protected Overrides Sub OnMouseMove(ByVal e As _
System.Windows.Forms.MouseEventArgs)
If picState = BallState.NORMAL_BALL Or _
picState = BallState.JUMPING_BALL Then
Me.Image = Image.FromFile(ImgList(picIndex + 1))
End If
End Sub
Protected Overrides Sub OnMouseLeave( _
ByVal e As System.EventArgs)
If picState = BallState.NORMAL_BALL Or picState = _
BallState.JUMPING_BALL Then
Me.Image = Image.FromFile(ImgList(picIndex))
End If
End Sub
End Class
```

После этих добавлений (DPaint.vb, mModule.vb, MotionPic.vb, open.ico, save.ico) в панели Solution Explorer должны быть файлы, показанные выше. Дважды щёлкая по имени файла, любой файл можно открыть, изучить и редактировать.

Теперь в наш проект добавляем переменные и методы, связанные с формой Form2 для вывода результатов игры.

Открываем файл Form2.vb (например, так: File, Open, File) и вверху записываем директивы для подключения требуемых пространств имен:

```
Imports System.Drawing.Drawing2D
Imports System.Drawing.Text
Imports System.IO
```

Теперь в классе Form2 нашего проекта записываем следующие переменные и методы.

**Листинг 21.12.** Переменные и методы.

```
Dim gr As Graphics
Dim lbrTitle As LinearGradientBrush
Dim lbrBoard As LinearGradientBrush
Dim midPoint As Point
Dim startPoint As PointF
Dim intGradiantStep As Integer = 5
Dim intCurrentGradientShift As Integer = 0
Const colW1 As Integer = 250
Const colW2 As Integer = 150
Const rowH As Integer = 30
Dim AddedPlayer As New Player("", "-1")
Dim ArrPlayer As New ArrayList
Dim intCurrentGradientRow As Integer = 110
Public WriteOnly Property AddPlayer() As Player
Set(ByVal Value As Player)
If Value.PlayerName.Length > 14 Then
Value.PlayerName = _
Value.PlayerName.Substring(0, 14)
End If
AddedPlayer = Value
End Set
End Property
Public Sub drawTable()
Me.BackgroundImage = Nothing
Me.BackColor = Color.Moccasin
Application.DoEvents()
Dim g As Graphics = CreateGraphics()
Dim tpen1 As New Pen(Color.Red, 1)
Dim tpen2 As New Pen(Color.Black, 1)
Dim P1 As New Point(2, 80)
Dim P2 As New Point(400, 80)
For i As Integer = 0 To 11
g.DrawLine(tpen1, P1, P2)
P1.Y += 1
P2.Y += 1
g.DrawLine(tpen2, P1, P2)
P1.Y += rowH – 1
P2.Y += rowH – 1
Next
P1.Y = 80
P2.X = P1.X
P2.Y -= rowH
g.DrawLine(tpen2, P1, P2)
P1.X += 1
P2.X += 1
g.DrawLine(tpen1, P1, P2)
P1.X += colW1
P2.X += colW1
g.DrawLine(tpen2, P1, P2)
```

```
P1.X += 1
P2.X += 1
g.DrawLine(tpen1, P1, P2)
P1.X += colW2 – 3
P2.X += colW2 – 3
g.DrawLine(tpen1, P1, P2)
P1.X += 1
P2.X += 1
g.DrawLine(tpen2, P1, P2)
SaveScore()
LoadScore()
Timer1.Enabled = True
Timer2.Enabled = True
End Sub
Public Sub LoadScore()
If Not File.Exists("Score.dat") Then
GoTo newScore
End If
Dim FSR As New StreamReader("Score.dat")
Dim s As String
s = FSR.ReadLine
If Trim(s) <> "#Assignment Line#" Then
FSR.Close()
GoTo newScore
End If
For i As Integer = 0 To 9
s = FSR.ReadLine
Dim PlayerName As String = s.Split(CChar(";"))(0)
Dim PlayerScore As String = s.Split(CChar(";"))(1)
ArrPlayer.Add(New Player(PlayerName, PlayerScore))
Next
FSR.Close()
Exit Sub
newScore:
ArrPlayer.Add(New Player("AAA", "5000"))
ArrPlayer.Add(New Player("AAA", "4500"))
ArrPlayer.Add(New Player("AAA", "4000"))
ArrPlayer.Add(New Player("AAA", "3500"))
ArrPlayer.Add(New Player("AAA", "3000"))
ArrPlayer.Add(New Player("AAA", "2500"))
ArrPlayer.Add(New Player("AAA", "2000"))
ArrPlayer.Add(New Player("AAA", "1500"))
ArrPlayer.Add(New Player("AAA", "1000"))
ArrPlayer.Add(New Player("AAA", "500"))
End Sub
Public Sub SaveScore()
LoadScore()
For i As Integer = 0 To 9
If CDbl(AddedPlayer.PlayerScore) >= _
```

```
CDbl(CType(ArrPlayer(i), Player).PlayerScore) Then
ArrPlayer.Insert(i, AddedPlayer)
Exit For
End If
Next
Dim FSW As New StreamWriter("Score.dat")
Dim s As String = "#Assignment Line#"
s += Chr(13) + Chr(10)
For i As Integer = 0 To 9
s += CType(ArrPlayer(i), Player).PlayerName + ";" + _
CType(ArrPlayer(i), Player).PlayerScore
s += Chr(13) + Chr(10)
Next
FSW.Write(s)
FSW.Close()
End Sub
Public Sub PlashScreen()
Me.BackgroundImage = Image.FromFile("GameOver.gif")
Application.DoEvents()
Threading.Thread.Sleep(2000)
End Sub
```

Ниже формы Form2 дважды щёлкаем по значку первого таймера Timer. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.13.** Метод-обработчик события Tick.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Timer1.Tick
gr = CreateGraphics()
midPoint = New Point(Me.Width \ 2, 10)
Dim strText As String = "Score Board"
Dim fnt As New Font("Microsoft Sans Serif", 30, _
FontStyle.Bold, GraphicsUnit.Point)
Dim strSize As New SizeF(gr.MeasureString(strText, fnt))
Dim ptfGradientStart As New _
PointF(intCurrentGradientShift, 0)
Dim ptfGradientEnd As New PointF(0, intCurrentGradientRow)
lbrTitle = New LinearGradientBrush(ptfGradientStart, _
ptfGradientEnd, Color.SteelBlue, Color.Brown)
startPoint = New PointF(midPoint.X – _
CInt(strSize.Width / 2), midPoint.Y)
gr.DrawString(strText, fnt, lbrTitle, startPoint)
ptfGradientStart = New PointF(0, intCurrentGradientShift)
ptfGradientEnd = New PointF(intCurrentGradientRow, 0)
lbrTitle = New LinearGradientBrush(ptfGradientEnd, _
ptfGradientStart, Color.MediumSlateBlue, _
Color.GhostWhite)
gr.DrawString(strText, fnt, lbrTitle, startPoint.X – 2, _
startPoint.Y + 2)
intCurrentGradientShift += intGradiantStep
If intCurrentGradientShift = 400 Then
```

```
intGradiantStep = -5
ElseIf intCurrentGradientShift = -400 Then
intGradiantStep = 5
End If
End Sub
```

Ниже формы Form2 дважды щёлкаем по значку второго таймера Timer. Появляется шаблон метода, который после записи нашего кода принимает следующий вид.

**Листинг 21.14.** Метод-обработчик события Tick.

```
Private Sub Timer2_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Timer2.Tick
'showScore()
Dim g As Graphics = CreateGraphics()
Dim fnt As New Font("Courier New", 20, FontStyle.Bold, _
GraphicsUnit.Point)
Dim startPoint As PointF = New PointF(20, 80)
Dim nextPoint As PointF = _
New PointF(startPoint.X + colW1, 80)
Dim ptfGradientStart As New PointF(intCurrentGradientRow, _
startPoint.X)
Dim ptfGradientEnd As New PointF(nextPoint.Y, _
intCurrentGradientRow)
lbrBoard = New LinearGradientBrush(ptfGradientStart, _
ptfGradientEnd, Color.GreenYellow, Color.SlateGray)
Dim PlayerNames As String = "Name" + Chr(13) + Chr(10)
Dim PlayerScores As String = "Score" + Chr(13) + Chr(10)
For i As Integer = 0 To 9
PlayerNames += CType(ArrPlayer(i), Player).PlayerName _
+ Chr(13) + Chr(10)
PlayerScores += CType(ArrPlayer(i), Player).PlayerScore _
+ Chr(13) + Chr(10)
Next
g.DrawString(PlayerNames, fnt, lbrBoard, startPoint)
g.DrawString(PlayerScores, fnt, lbrBoard, nextPoint)
intCurrentGradientRow += intGradiantStep
End Sub
```

В случае необходимости, методика добавления в проект звукового сигнала Beep (по-русски: Бип) описана ранее.

## 21.5. Запуск игры

Строим и запускаем программу на выполнение обычным образом:

Build, Build Selection; Debug, Start Without Debugging.

В ответ Visual Studio выводит показанную выше форму, на которой в отдельных квадратах сетки, например, 9 x 9 сначала произвольным образом (при помощи генератора случайных чисел – г.с.ч. класса Random) искусственный интеллект выводит определённое количество, в данной игре 3, разноцветных объекта, например, 3 больших мяча, которые игрок может перемещать при помощи мыши, и 3 маленьких разноцветных мяча, которые размещает искусствен-

ный интеллект, чтобы помешать игроку построить прямую линию из мячей (так как в клетку с маленьким мячом большой мяч уже нельзя разместить).

Игрок периодически щёлкает по выбранному им мячу и по клетке, в которую мяч перемещается. А искусственный интеллект периодически размещает в пустующие клетки следующие 3 больших мяча произвольных цветов.

Как только игрок соберёт горизонтальную, вертикальную или диагональную прямую линию из 5 и более мячей одинакового цвета, игроку начисляются очки (по 100 очков за каждый собранный в линию мяч), а линия из собранных мячей исчезает, освобождая клетки для новых мячей. По такой схеме игрок играет согласно приведённым выше правилам.

По методике данной главы можно разрабатывать самые разнообразные игры с использованием искусственного интеллекта по сборке линий одного цвета из разноцветной палитры разнообразных фигур.

# Часть X. Методология программирования искусственного интеллекта в ролевых сюжетных играх

# Глава 22. Методика программирования искусственного интеллекта в сюжетных играх на примере сюжета о пещерных людях Адаме и Еве

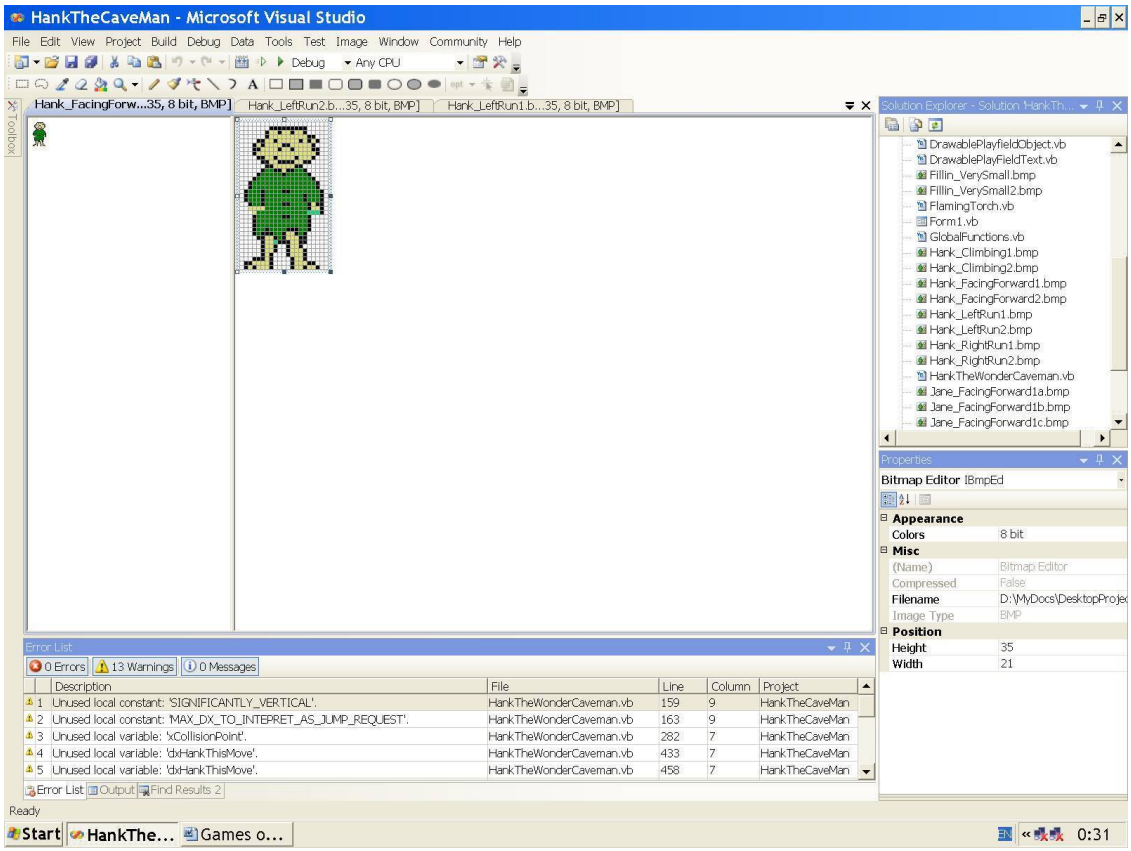
# 22.1. Общие сведения

Опишем методику проектирования и программирования типичной и распространённой ролевой (Role-playing game – RPG) сюжетной игры и использованием искусственного интеллекта, когда игрок в роли пещерного человека Адама собирает на земле горящие факелы для Евы (чтобы согреть её в пещере).

Данную игру мы будем разрабатывать, следуя проекту в заархивированном виде CaveManHank.zip (Пещерный человек Хэнк) автора Ivo Salmre от 27.6.2002 с сайта gotdotnet.com для устаревшей версии Visual Studio, причём, для карманного компьютера (КПК), но с нашими усовершенствованиями для современной версии Visual Studio, причём для настольного компьютера или планшета.

В панели Solution Explorer (нашего будущего проекта) или в папке с именем проекта дважды щёлкаем по имени графического файла Hank_FacingForward1.bmp (загруженного, например, из Интернета) для изображения пещерного мужчины с именем Хэнк (у нас он будет зваться, как Адам). Появляется собственный графический редактор Visual Studio с изображением этого мужчины, причём инструментами этого редактора можно редактировать это изображение (рис. 22.1) применительно к нашим задачам.

Аналогично в панели Solution Explorer дважды щёлкаем по имени графического файла Jane_FacingForward1a.bmp для изображения пещерной женщины с именем Джейн (у нас она будет зваться, как Ева). Появляется редактор Visual Studio с изображением этой женщины (рис. 22.2). Аналогично можно открыть и редактировать остальные графические файлы проекта.



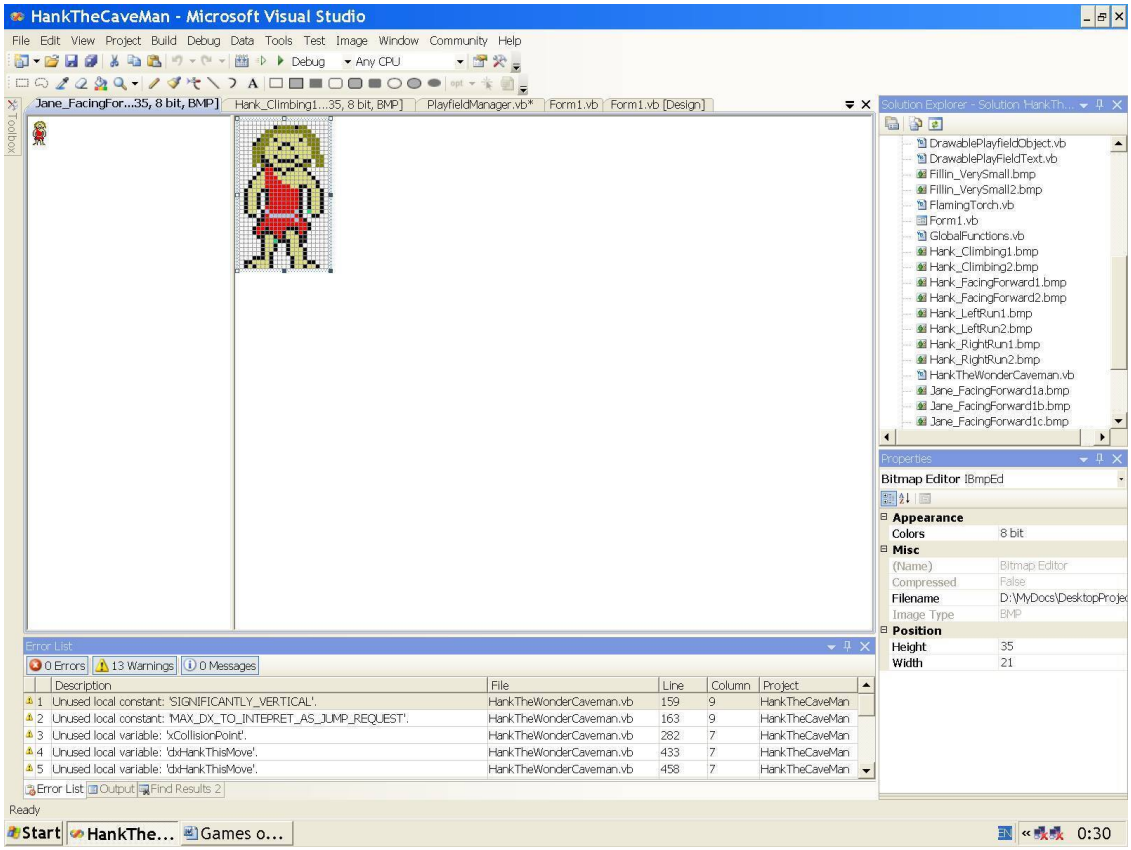



**Рис. 22.1.** Пещерный мужчина Адам. **Рис. 22.2.** Пещерная женщина Ева.

Как уже и ранее было рассказано, и в этой методической игре поле игры состоит из нескольких слоёв (layers) изображений.

Нижний слой – это фоновое изображение (фоновая картинка). Оно рисуется только один раз в начале выполнения программы.

На среднем слое находятся статические изображения, которые не перемещаются (в данной игре – это настилы и лестницы). Они не должны рисоваться каждый раз от кадра к кадру, когда изображение обновляется; они изменяют состояние только при переходе от одного уровня (level) игры к другому.

В главном (переднем) слое находятся динамические изображения, которые изменяются от кадра к кадру и за счёт этого перемещаются на экране. Они должны рисоваться каждый раз, когда обновляются анимированные изображения.

Другие пояснения этой игры можно найти (если необходимо) в книгах с сайта ZharkovPress.ru.

# 22.2. Правила игры

1. После запуска игры на экране появляется форма с пользовательским интерфейсом игры (рис. 22.3).

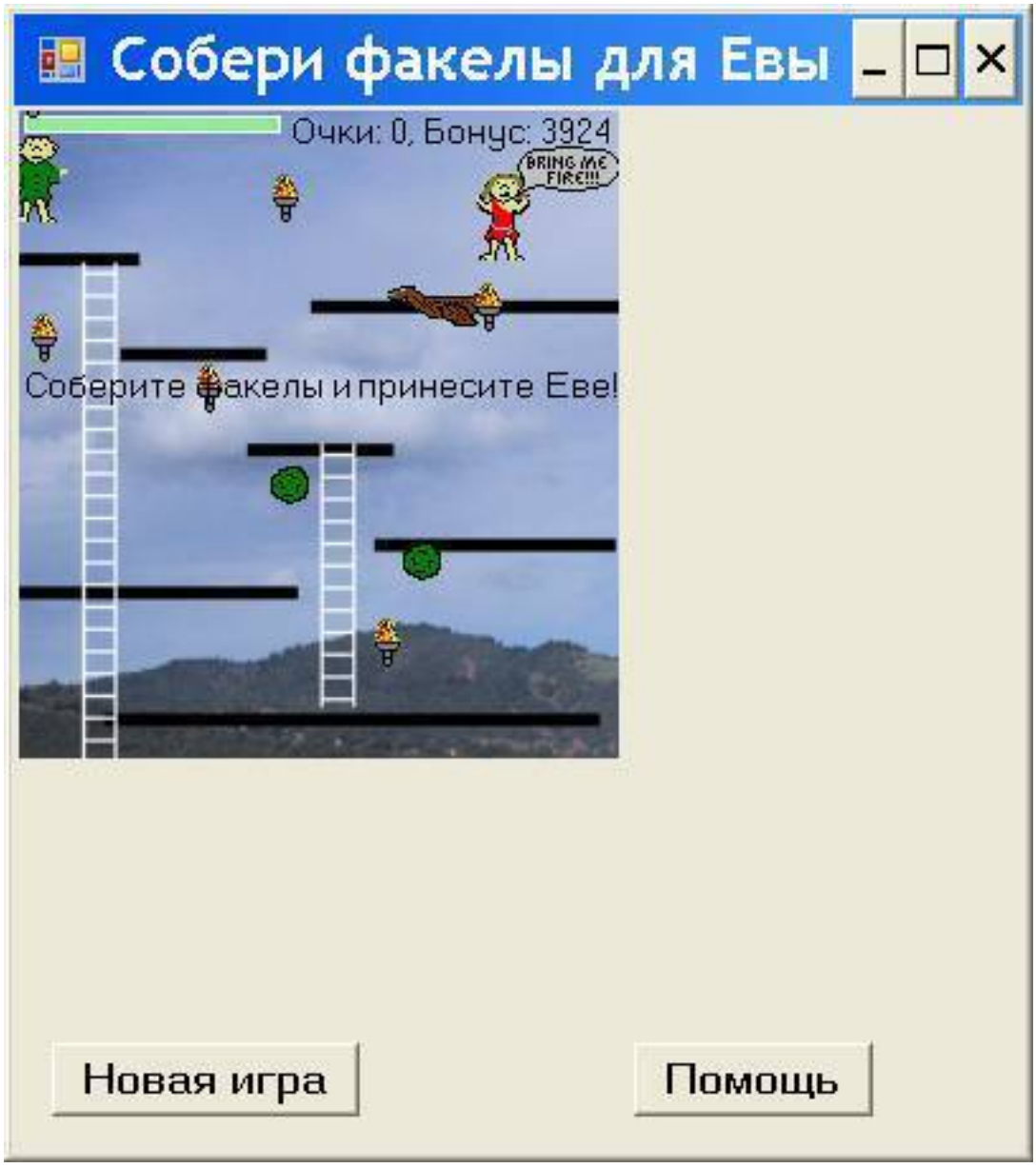



**Рис. 22.3.** Начало игры.

Слева в верхней части экрана мы видим индикатор энергии Адама, которая уменьшается по мере течения времени.

Правее находятся надпись “Очки: 0” и Бонус в виде начального времени, которое стремительно уменьшается.

Ниже расположен рисунок, на котором Ева кричит Адаму: “Bring me fire!!!” (Принеси мне огонь).

Левее находится рисунок Адама с озабоченным видом.

Ниже мы видим указание игроку, управляющему изображением Адама: “Соберите факелы и принесите Еве”.

Управление Адамом состоит в том, чтобы щёлкать по экрану указателем мыши в том месте экрана, куда должен идти Адам, чтобы взять факел. После каждого щелчка появляется пунктирная линия, показывающая, куда пойдёт Адам.

Чтобы Адам подпрыгнул от падающих валунов (которые извергают вулканы, управляемые искусственным интеллектом) или от пролетающей огромной птицы эпохи динозавров (которые видны на предыдущем рисунке), следует нажать клавишу пробела.

Очки засчитываются за каждый собранный факел данного уровня, за оставшееся время бонуса и за энергию игрока, оставшуюся в конце каждого уровня.

Адам потеряет энергию, если он будет поражён валуном или птицей. Адам будет подскакивать, чтобы не быть поражённым этими объектами. Адам также потеряет энергию, если он упадёт с большой высоты.

Звучит мелодия начала игры.

2. Через несколько секунд поясняющая надпись “Соберите факелы и принесите Еве” исчезает (чтобы не заслонять игровое поле).

3. Щёлкаем указателем мыши в том месте формы, куда должен идти Адам, чтобы взять факел. После каждого щелчка появляется пунктирная линия, показывающая, куда пойдёт Адам.

Например, щёлкаем по первому факелу, который расположен по горизонтали справа от Адама. На этом пути нет настилов, поэтому Адам без помех берет этот факел, и игроку начисляются 200 очков (рис. 22.4)


**Рис. 22.4.** Игроку начислено 200 очков.

4. К другим факелам подойти не так просто, так как мешают настилы. И Адам может пройти к факелам только через промежутки в настилах и по лестницам.

Щёлкаем указателем мыши в промежутках между настилами и по лестницам, указывая Адаму путь к факелам.

Мы взяли ещё один факел и получили 500 очков (рис. 22.5).


**Рис. 22.5.** Игроку начислено 500 очков.

5. Напомним, что, чтобы Адам подпрыгнул от падающих валунов (которые извергают вулканы) или от пролетающей огромной птицы эпохи динозавров, следует нажать клавишу пробела.

Мы не успели нажать клавишу пробела, Адам не подпрыгнул и был сбит валуном. Энергия у Адама закончилась (на индикаторе вверху), и игра тоже закончилась (рис. 22.6).

Звучит мелодия окончания игры.

6. Для начала новой игры следует нажать кнопку “Новая игра”.

7. Если нам нужна справка о правилах игры, то щёлкаем кнопку Помощь. Последовательно появляются несколько библиотечных панелей MsgBox с записанными нами в программу текстами (рис. 22.7 – 22.9).

8. Если в игре участвует несколько человек, то победителем считается тот, кто набрал большее количество очков.

На основании этих правил можно сформулировать другие правила игры с использованием искусственного интеллекта, и любые правила ввести в показанные справочные панели.

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

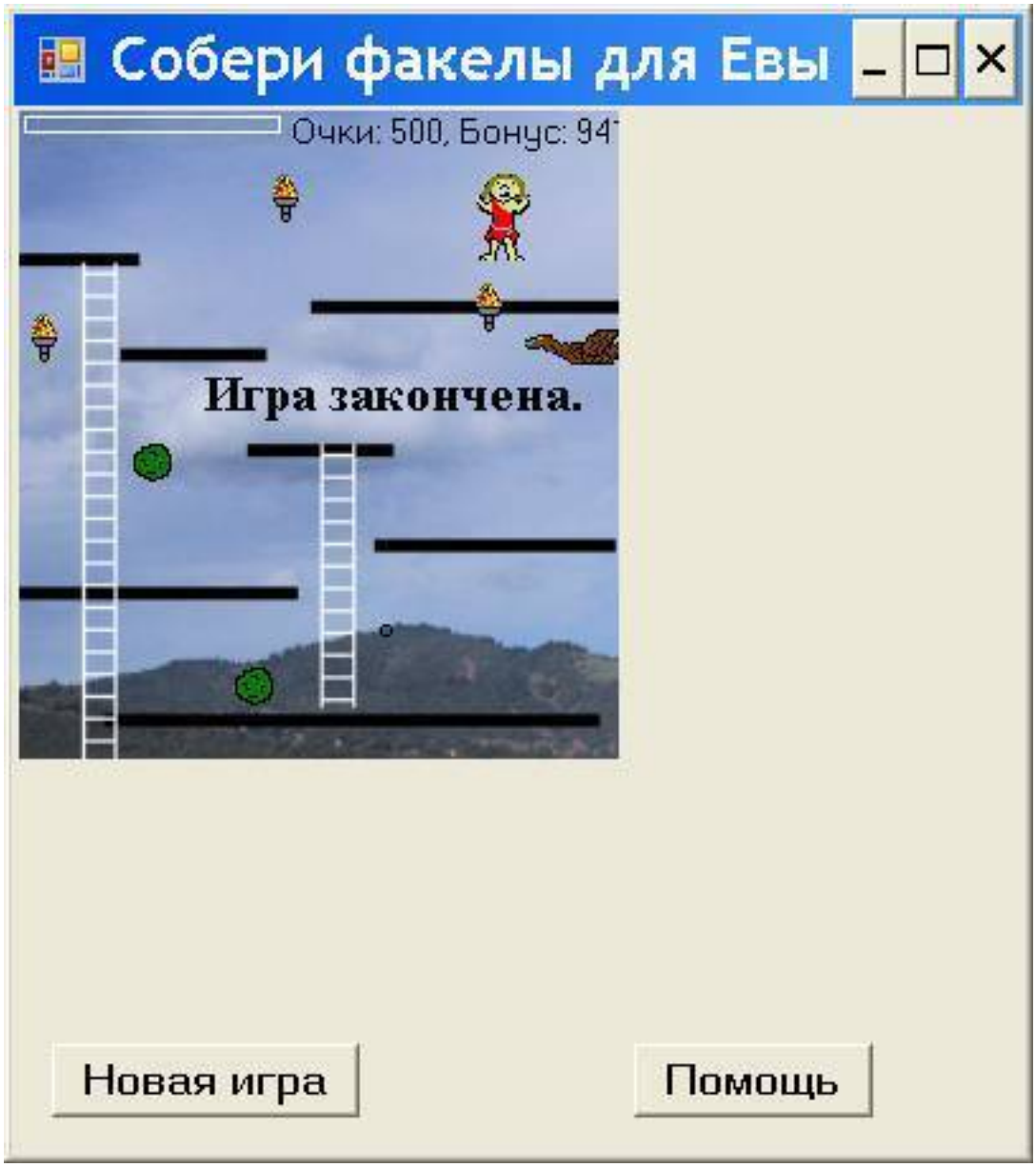



**Рис. 22.6.** Игра закончена.

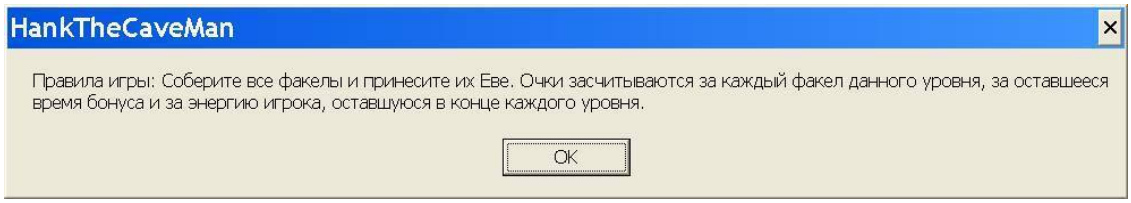



**Рис. 22.7.** Первая часть справки.

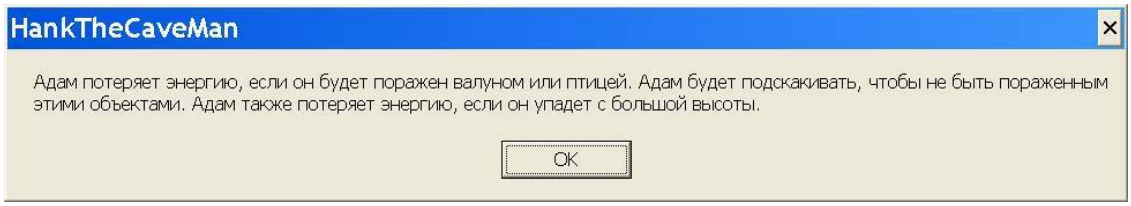



**Рис. 22.8.** Вторая часть справки.

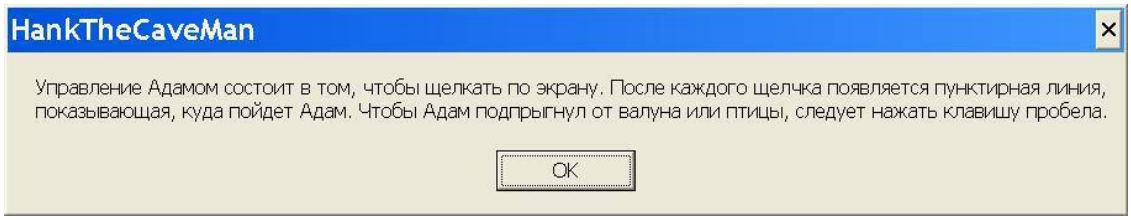



**Рис. 22.9.** Третья часть справки.

# 22.3. Создание проекта

Создаём проект по обычной схеме: в VS в панели New Project в окне Project types выбираем тип проекта Visual Basic, Windows, в окне Templates выделяем шаблон Windows Forms Application, в окне Name записываем имя проекта HankTheCaveMan и щёлкаем OK.

Так как, в отличие от указанного выше оригинала, в нашем проекте при загрузке графических файлов имя проекта будет определяться в программе, то сейчас в окне Name можно записать любое имя. Создаётся проект, появляется форма Form1 в режиме проектирования (рис. 22.10). Оставляем по умолчанию или проектируем форму, как подробно описано в параграфе “Методика проектирования формы”. Например, в панели Properties (для Form1) в свойстве Font увеличиваем размер шрифта до 10. За маркеры увеличиваем размеры формы таким образом, чтобы в свойстве Size были, например, такие величины: 405; 456.

С панели инструментов Toolbox переносим в нижнюю часть формы элемент управления типа окна текста TextBox.

Ниже размещаем кнопку Button с номером 1. В панели Properties (для этого элемента) в свойстве Text записываем “Новая игра”.

Правее размещаем кнопку Button. В панели Properties (для этого элемента) в свойстве Name записываем ButtonNextLevel, а в свойстве Text записываем Уровень.

Правее размещаем кнопку Button. В панели Properties (для этого элемента) в свойстве Name записываем buttonInstructions, а в свойстве Text записываем Помощь.

Свойства и формы, и этих элементов управления можно изменять, как описано ранее.

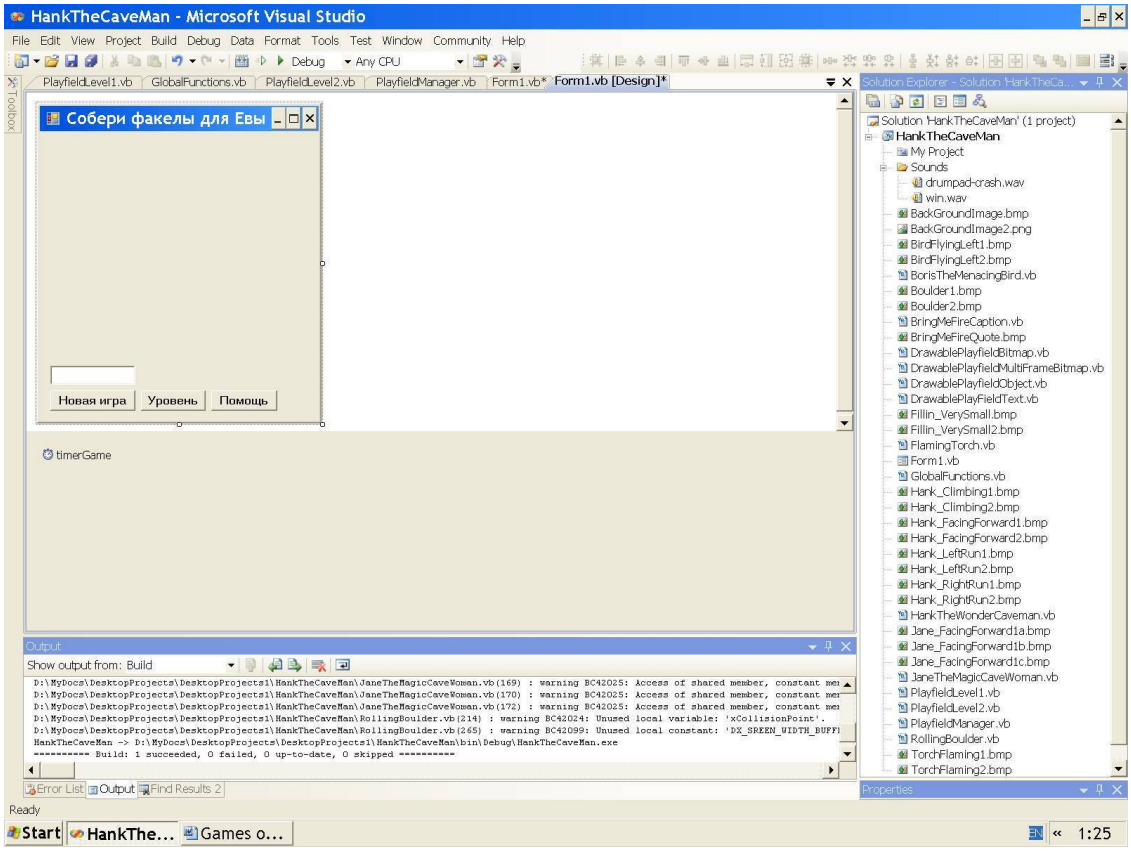
Переносим таймер Timer. В панели Properties (для этого невидимого на форме компонента) в свойстве Name записываем timerGame, в свойстве Enabled выбираем True, а свойству Interval задаём значение 40 миллисекунд, что соответствует 25 кадрам в секунду по стандарту телевещания России.

Теперь добавляем в проект графические файлы по обычной схеме: в панели Solution Explorer выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, Existing Item (или Project, Add Existing Item), в панели Add Existing Item в окне “Files of type” выбираем “All Files”, в центральном окне находим (например, в папке с загруженными из Интернета файлами) и с нажатой клавишей Ctrl выделяем файлы и щёлкаем кнопку Add, чтобы после этого добавления в панели Solution Explorer были файлы, показанные на рис. 22.9.

В панели Solution Explorer выделяем все графические файлы (с нажатой клавишей Ctrl), а в панели Properties в свойстве Build Action (Действие при построении) вместо заданного по умолчанию выбираем значение Embedded Resource (Встроенный ресурс) для всех этих файлов.

Если в игре применяются несколько звуковых файлов, то их целесообразно разместить в одной папке с именем, например, Sounds. Для добавления в проект этой папки, в панели Solution Explorer выполняем правый щелчок по имени проекта, в контекстном меню выбираем Add, New Folder, в поле появившегося значка папки записываем имя папки и нажимаем клавишу Enter.

Добавляем в эту папку звуковые файлы по стандартной схеме: выполняем правый щелчок по имени этой папки, 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 с загруженными из Интернета файлами) и с нажатой клавишей Ctrl выделяем файлы и щёлкаем кнопку Add. В панели Solution Explorer мы увидим эти файлы.


HankTheCaveMan - Microsoft Visual Studio
Собери факелы для Евы
Новая игра
Уровень
Помощь
timerGame
Ready

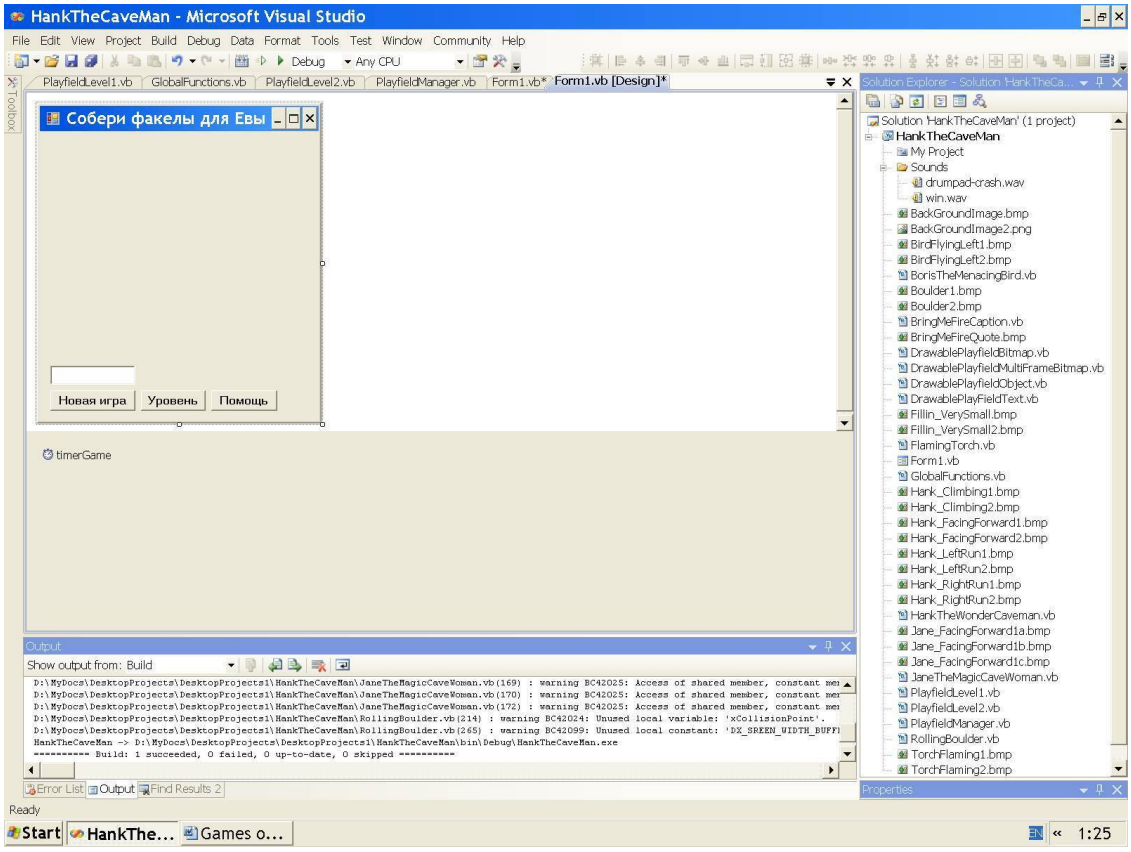

HankTheCaveMan - Microsoft Visual Studio
Собери факелы для Евы
Новая игра
Уровень
Помощь
timerGame
Ready

**Рис. 22.10.** Форма Form1 в режиме проектирования. **Рис. 22.11.** Панель Solution Explorer.

# 22.4. Код и запуск программы

Открываем файл Form1.vb (например, по схеме: File, Open, File) и в самом верху импортируем пространства имён для управления соответствующими классами:

```
Imports System.Reflection 'Для класса Assembly.
```

В классе Form1 нашего проекта записываем следующие переменные и методы.

**Листинг 22.1.** Переменные и методы.

```
'_
'Offsets on form where we should draw the playfield
'_
Const GAME_SCREEN_DX = 2
Const GAME_SCREEN_DY = 1
'Current playfield
Private m_playfieldManager As PlayFieldManager
'Gfx object of the form that we will render the playfield to
Dim m_myFormsGraphics As Graphics
'_
'Does the things we need to get get a round running
'_
Sub InitializeNewLevel()
'Add a game event handler to get updates on the game
AddHandler m_playfieldManager.GameStateChanged, _
AddressOf GameEventOccured
'Start the game
timerGame.Enabled = True
ButtonNextLevel.Visible = False
'_
'Give the text-box focus so we can get keyboard events
'_
TextBox1.Focus()
End Sub
'_
'Starts a new game
'_
Sub StartGame()
'Мелодия начала игры с ожиданием окончания мелодии:
'My.Computer.Audio.Play("..\..\Sounds\drumpad-crash.wav", _
' AudioPlayMode.WaitToComplete)
'Мелодия начала игры без ожидания её окончания:
My.Computer.Audio.Play("..\..\Sounds\drumpad-crash.wav")
'_
'Create Graphics object and cache it
'_
If (m_myFormsGraphics Is Nothing) Then
m_myFormsGraphics = Me.CreateGraphics
End If
'_
```

```
'Create the playfield level we want to start on
'_
m_playfieldManager = New Playfield_Level2()
'_
'Start the level running
'_
InitializeNewLevel()
End Sub
Sub PictureBoxPlayField_MouseDown( _
ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.MouseEventArgs)
'If Hank's around, give him a new destination
'If Not (m_HankTheWonderCaveMan Is Nothing) Then
If Not (m_playfieldManager Is Nothing) Then
m_playfieldManager.HankTheWonderCaveman. _
setHanksDestination(e.X, e.Y)
End If
End Sub
'_
'This event-sink is to get called when a game event occurs
'_
Sub GameEventOccured(ByVal gameUpdateMessage As _
PlayFieldManager.GameUpdate)
'The level was completed succesfully
If (gameUpdateMessage = PlayFieldManager.GameUpdate. _
levelSuccesfullyCompleted) Then
ButtonNextLevel.Visible = True
End If
End Sub
'Загружаем в проект файлы изображений по такой схеме:
'Создаём объект myAssembly класса Assembly и присваиваем ему
'ссылку на исполняемую сборку нашего приложения:
Public myAssembly As Assembly = Assembly.GetExecutingAssembly()
'Создаём объект myAssemblyName
'класса System.Reflection.AssemblyName и присваиваем ему
'имя сборки, которое состоит из имени проекта,
'Version, Culture, PublicKeyToken:
Public myAssemblyName As AssemblyName = myAssembly.GetName()
'Из имени сборки при помощи свойства Name
'выделяем имя проекта типа string:
Public myName_of_project As String = myAssemblyName.Name
```

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

**Листинг 22.2.** Метод для загрузки изображений.

```
Private Sub Form1_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
Me.Visible = False
'_
'Load and cache the global object we need
```

```
'_
g_InitializeGlobals()
Me.Visible = True
'_
'Place the textbox off the screen
'_
TextBox1.Width = 4
TextBox1.Multiline = True
TextBox1.Left = -20
'Start the game
StartGame()
End Sub
```

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

**Листинг 22.3.** Метод для закрытия формы.

```
Private Sub Form1_FormClosing(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.FormClosingEventArgs) _
Handles MyBase.FormClosing
' '_
' 'Cleanup
' '_
'_
'Free up the form based objects
'_
m_playfieldManager = Nothing
If Not (m_myFormsGraphics Is Nothing) Then
m_myFormsGraphics.Dispose()
m_myFormsGraphics = Nothing
End If
'_
'Free up all the global objects
'_
g_DisposeAndFreeGlobals()
End Sub
```

На форме дважды щёлкаем по кнопке Button с номером 1 и текстом “Новая игра” (или в панели Properties, для этого элемента, на вкладке Events дважды щёлкаем по имени соответствующего события). Появившийся шаблон после записи нашего кода принимает следующий вид.

**Листинг 22.4.** Метод-обработчик щелчка кнопки.

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button2.Click
StartGame()
End Sub
```

На форме дважды щёлкаем по кнопке Button с текстом Уровень (или в панели Properties, для этого элемента, на вкладке Events дважды щёлкаем по имени соответствующего события). Появившийся шаблон метода после записи нашего кода принимает следующий вид.

**Листинг 22.5.** Метод-обработчик щелчка кнопки.

```
Private Sub ButtonNextLevel_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
```

```
Handles ButtonNextLevel.Click
'_
'Starts a new level of play
'_
'Ask the current level what it's next level is
m_playfieldManager = m_playfieldManager.GetNextLevel()
'Start the level running
InitializeNewLevel()
End Sub
```

На форме дважды щёлкаем по кнопке Button с текстом Помощь (или в панели Properties, для этого элемента, на вкладке Events дважды щёлкаем по имени соответствующего события). Появившийся шаблон метода после записи нашего кода принимает следующий вид.

**Листинг 22.6.** Метод-обработчик щелчка кнопки.

```
Private Sub buttonInstructions_Click( _
ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles buttonInstructions.Click
MsgBox("Правила игры: Соберите все факелы и " + _
"принесите их Еве. Очки засчитываются " + _
"за каждый факел данного уровня, " + _
"за оставшееся время бонуса и за энергию игрока, " _
+ "оставшуюся в конце каждого уровня.")
MsgBox("Адам потеряет энергию, если он будет поражён " + _
"валуном или птицей. Адам будет подскакивать, " + _
"чтобы не быть поражённым этими объектами. " + _
"Адам также потеряет энергию, " + _
"если он упадёт с большой высоты.")
MsgBox("Управление Адамом состоит в том, " + _
"чтобы щёлкать по экрану. После каждого щелчка " + _
"появляется пунктирная линия, показывающая, " + _
"куда пойдёт Адам. Чтобы Адам подпрыгнул " + _
"от валуна или птицы, " + _
"следует нажать клавишу пробела.")
'Bring the focus back to the textbox
'so it gets the keyboard input
TextBox1.Focus()
End Sub
```

В панели Properties, для элемента TextBox, на вкладке Events дважды щёлкаем по имени события KeyDown. Появившийся шаблон после записи нашего кода принимает следующий вид.

**Листинг 22.7.** Метод-обработчик события.

```
Private Sub TextBox1_KeyDown(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.KeyEventArgs) _
Handles TextBox1.KeyDown
'_
'This textbox is a way to get the keyboard input
'for the game becuase:
' 1. When you hit the "START" button, it get's the focus
'(so the form no longer gets keydown events)
' 2. The Start Button does not get key-down events
```

```
'_
If (m_playfieldManager Is Nothing) Then
Return
End If
Dim i As Integer
i = e.KeyValue
Const PocketPC_BUTTON_PUSHED = 134
Const Keyboard_SPACE_PUSHED = 32
Const PocketPC_UP = 38
Const PocketPC_DOWN = 40
Const PocketPC_LEFT = 37
Const PocketPC_RIGHT = 39
Dim hank As HankTheWonderCaveman = _
m_playfieldManager.HankTheWonderCaveman
Const D_MOVEMENT = 14
Const D_MOVEMENT_UP = 70
If (i = PocketPC_BUTTON_PUSHED) Or _
(i = Keyboard_SPACE_PUSHED) Then
hank.MakeHankJump()
e.Handled = True
ElseIf (i = PocketPC_RIGHT) Then
hank.NudgeHanksDirection(D_MOVEMENT, 0)
ElseIf (i = PocketPC_LEFT) Then
hank.NudgeHanksDirection(-D_MOVEMENT, 0)
ElseIf (i = PocketPC_UP) Then
hank.NudgeHanksDirection(0, -D_MOVEMENT_UP)
ElseIf (i = PocketPC_DOWN) Then
hank.NudgeHanksDirection(0, D_MOVEMENT)
End If
TextBox1.Text = ""
End Sub
```

Дважды щёлкаем по значку для таймера Timer (или в панели Properties, для этого компонента, на вкладке Events дважды щёлкаем по имени соответствующего события). Появившийся шаблон метода после записи нашего кода принимает следующий вид.

**Листинг 22.8.** Метод, вызываемый таймером через каждый интервал Interval времени.

```
Private Sub timerGame_Tick(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles timerGame.Tick
' '_
' 'This timer is the heartbeat if the game.
' 'Every time it gets called we
' 'allow all of the game characters to move,
' 'we check for collisions beween objects,
' 'and we render the board to the screen.
' '_
'_
'If we are not running, exit the sub
'_
If (m_playfieldManager Is Nothing) Then Exit Sub
'_
```

```
'Give all the characters a chance to move
'_
m_playfieldManager.MoveCharactersOnPlayfield()
'_
'Render the playfield and all of the objects onto the form
'_
m_myFormsGraphics.DrawImage(m_playfieldManager. _
RenderPlayField(), GAME_SCREEN_DX, GAME_SCREEN_DY)
End Sub
```

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

**Листинг 22.9.** Метод-обработчик события.

```
Private Sub Form1_MouseDown(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) _
Handles MyBase.MouseDown
'Make sure there is a game we are playing
If Not (m_playfieldManager Is Nothing) Then
If e.Button = MouseButtons.Right Then
m_playfieldManager.HankTheWonderCaveman. _
MakeHankJump()
Else
'_
'Figure out where on the game playfield we clicked
'_
Dim x_GAME As Integer
Dim y_GAME As Integer
x_GAME = e.X – GAME_SCREEN_DX
y_GAME = e.Y – GAME_SCREEN_DY
'Let Hank figure out what to do
m_playfieldManager.HankTheWonderCaveman. _
setHanksDestination(x_GAME, y_GAME)
End If
End If
End Sub
```

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

Теперь в наш проект добавляем новые файлы (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 (или Project, Add Existing Item), в панели Add Existing Item в окне “Files of type” выбираем “All Files”, в центральном окне находим (в папке компьютера с загруженными, например, из Интернета, файлами) и с нажатой клавишей Ctrl выделяем все файлы формата (.vb) и щёлкаем кнопку Add, чтобы после этого добавления в панели Solution Explorer были файлы, показанные на рис. 22.11.

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 BorisTheMenacingBird.vb 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 22.10.** Новый файл.

```
'_
'This class represents one of the characters in the game,
' "BorisTheMenacingBird"
'
'It contains logic for the rendering and movement of the character
'_
Public Class BorisTheMenacingBird
Inherits DrawablePlayfieldMultiFrameBitmap
Private m_yVelocityBoris As Integer      'How fast are _
'we traveling vertically?
Private m_xVelocityBoris As Integer      'How fast are we
'traveling vertically?
Const MAX_DY_BORIS_FLYING = 4
Private m_y_accelerationBorris As Integer
Private m_world_I_Inhabit As PlayFieldManager
Private m_myModeOfMovement As ModeOfMovement
Public Enum ModeOfMovement As Integer
 Flying = 1
End Enum

'When did we last update a flipped image
Private m_lastTickCountWhenImageFlipped As Integer
Const DTIME_TO_FLAP_WINGS = 400      'Every 600 ms we should
'flap our wings
'_
'These are the current image states for Hank
'_
Private Enum BorisImagesIndexes
 flyLeft1 = 1
 flyLeft2 = 2
End Enum
'_
'[in] X,Y : Position to start Hank at
'[in] worldHankInhabits : Playfield in which Hank lives
'_
Sub New(ByVal x As Integer, ByVal y As Integer, _
ByVal world_I_Inhabit As PlayFieldManager)
 '_
 'Get the bitmaps for our character
 '_
 Dim col As Collection
 col = g_FlyingBirdPictureCollection()
 m_world_I_Inhabit = world_I_Inhabit
 ChangeMyMovementState(ModeOfMovement.Flying)
 'Start him off as falling.
 'Initialize our base class with these…
```

```
 MyBase.Initialize_DrawableMultiPlayfiedBitmapObject( _
x, y, col, True)
 'Set the image index
 Me.CurrentFrameIndex = BorisImagesIndexes.flyLeft1
 'Set Boris' speed
 m_xVelocityBoris = -4
 m_y_accelerationBorris = 1
 '_
 'Set the collision rectangle for Hank
 '_
 Const BORIS_COLLISION_STARTX = 11
 Const BORIS_COLLISION_DX = 31
 Const BORIS_COLLISION_STARTY = 6
 Const BORIS_COLLISION_DY = 7
 Me.CollisionRectangle = New Rectangle( _
BORIS_COLLISION_STARTX, BORIS_COLLISION_STARTY, _
BORIS_COLLISION_DX, BORIS_COLLISION_DY)
End Sub
'_
'This function should be called to bring about any change
'of state
'It sets/resets any other variables we need to when we deal
'with state transitions
'_
Private Sub ChangeMyMovementState(ByVal newState As _
ModeOfMovement)
 m_myModeOfMovement = newState
End Sub
'_
'MOVE Boris
'
'This function is called to move the character on the screen
```

# Конец ознакомительного фрагмента.