**Лабораторная работа № 9  
«Паззл»**

Для реализации приложения‑игры «Паззл» будем использовать возможности технологии *Drag&Drop*: она хорошо известна любому пользователю компьютера иреализована как на уровне операционной системы, так и в самых разнообразных программах. Эта технология позволяет перемещать объекты (в ОС чаще всего файлы и папки, в программах – текст и изображения) посредством перетаскивания их мышью.

В процессе перетаскивания выделяют два типа объектов. Это источник, т. е. перетаскиваемый объект, и приемник – объект, на который при удачном завершении процесса перетаскивания помещается объект‑источник.

В языке *C*# технология *Drag&Drop* базируется на использовании для каждого объекта набора свойств, методов и событий. Мы рассмотрим основные из них.

Для объекта‑приемника используют логическое свойство *AllowDrop*, указывающее, может ли данный объект принимать данные, перетаскиваемые на него пользователем.

Для инициализации процесса перетаскивания используется метод *DoDragDrop(Object data,* [*DragDropEffects*](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/fxref_system.windows.forms/html/5a0ac9b5-457a-2c7f-f511-4452c40d2f60.htm) *allowedEffects).* Здесь в параметре *data* необходимо задать перетаскиваемые данные. Параметр *allowedEffects* позволяет определить допустимые операции с перетаскиваемым объектом и может принимать одно из значений перечисления [*DragDropEffects*](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/fxref_system.windows.forms/html/5a0ac9b5-457a-2c7f-f511-4452c40d2f60.htm):

* *None* – объект‑приемник не принимает данные;
* *Copy* – данные из объекта‑источника копируются в целевой объект;
* *Move* – данные из объекта‑источника перемещаются в целевой объект;
* *Link* – данные из объекта‑источника связываются с целевым объектом;
* *Scroll* – объект‑приемник можно прокрутить во время операции перетаскивания, если нужная область целевого объекта в настоящий момент не отображается;
* *All* – сочетание результатов операций *Copy*, *Link*, *Move* и *Scroll*.

Отметим, что в зависимости от выбранной операции внешний вид указателя мыши меняется.

Событие **DragEnter** происходит, когда курсор мыши в процессе операции перетаскивания попадает в пределы объекта‑приемника.

Событие **DragOver** происходит при перемещении указателя мыши в пределах объекта‑приемника.

Событие **DragLeave** происходит при перетаскивании объекта‑источ­ни­ка за пределы объекта‑приемника.

Событие **DragDrop** генерируется для объекта‑приемника при завершении процесса перетаскивания.

Каждый из обработчиков этих событий получает параметр *e*, предоставляющий информацию о положении курсора мыши, о перетаскиваемом объекте, а также позволяющий задать допустимые операции перетаскивания.

**Упражнение 9.1.** Напишите приложение, реализующее игру «Паззл».

**Решение**

Прежде чем приступить к разработке приложения, вам потребуется подобрать картинку, которую будем использовать в качестве основы для паззла. Далее в любом графическом редакторе необходимо разрезать выбранную картинку на равные части (в нашем случае – 16) и каждую часть сохранить под своим именем.

Перейдем непосредственно к разработке приложения. Создайте новый проект *Puzzle* в решении *Lab9*.

**1‑й этап.** Создадим визуальный интерфейс приложения (рис. 9.2).

Измените значение свойства *Text* формы на «Паззл», размеры формы 600; 600.

Поместите на форму компонент TabledLayoutPanel. Этот компонент служит контейнером для других компонентов и облегчает задачу их упорядоченного расположения в виде таблицы. Измените свойство *Name* этого компонента на *Field*.

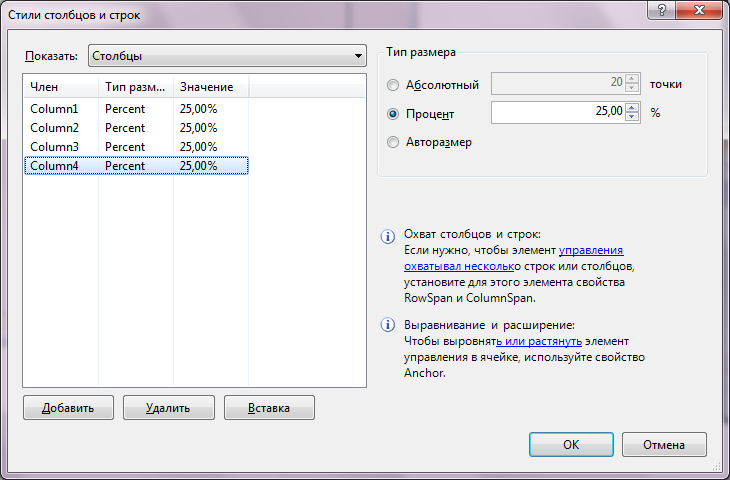
Измените количество строк и столбцов компонента в диалоговом окне Стили столбцов и строк (рис. 9.1). Для этого достаточно в окне Cвойств щелкнуть по кнопке  напротив свойства *Columns*. С помощью кнопки *Добавить* доведите количество столбцов до четырех и задайте их размеры в процентах 25% для каждого столбца относительно размеров компонента. Повторите операцию для строк. В результате компонент TabledLayoutPanel представляет собой таблицу 4×4.

Установите значение свойства *Dock* компонента TabledLayoutPanel равным *Fill* (выберите центральный прямоугольник). Это свойство позволяет задать расположение границ компонента относительно родительского компонента. Значение *Fill* указывает, что компонент занимает всю рабочую область формы.

В верхнюю левую ячейку таблицы поместите компонент PictureBox. Измените значения его свойств следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *picture1* |
| *SizeMode* | *StretchImage* |
| *Dock* | *Fill* |
| *Margin* | 0; 0; 0; 0 |

*Рис. 9.1.* Диалоговое окно Стили столбцов и строк



Добавление столбца / строки

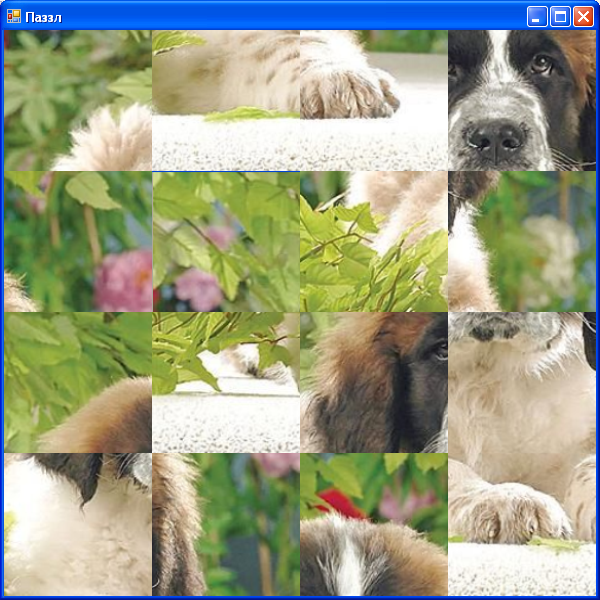
Указание размеров выделенного столбца / строки

Выбор объекта для редактирования – строки или столбцы

Далее скопируйте компонент *picture1* в буфер и вставьте его копию во вторую ячейку первой строки таблицы, измените значение свойства *Name* на *picture2.* Повторите операцию для оставшихся ячеек таблицы.

Теперь в каждый компонент PictureBox загрузите произвольный фрагмент картинки (не повторяясь). Для этого в окне Свойств выберите свойство *Image* и вызовите диалоговое окно Выбор ресурса, в котором выберите *Локальный ресурс*, нажмите на кнопку *Импорт* и укажите путь к одному из подготовленных вами в начале работы файлов.

В результате на форме произвольным образом должны располагаться 16 фрагментов исходной картинки (рис. 9.2).



*Рис. 9.2.* Форма приложения «Паззл»

**2‑й этап.** Создание программного кода.

Процесс собирания паззла удобно реализовать с использованием технологии *Drag&Drop*. В этом случае пользователь перетаскивает фрагменты картинки с одного места формы на другое с помощью мыши, при этом два фрагмента картинки меняются местами. При таком способе реализации каждый компонент PictureBox может являться одновременно как источником, так и приемником, поэтому необходимо задать свойство *AllowDrop* для каждого компонента PictureBox. Поскольку это свойство в окне Свойств не отображается, зададим его значение программным путем в обработчике события Load формы.

private void Form1\_Load(object sender, EventArgs e)

{

this.picture1.AllowDrop = true;

this.picture.AllowDrop = true;

this.picture3.AllowDrop = true; this.picture4.AllowDrop = true;

this.picture5.AllowDrop = true; this.picture6.AllowDrop = true;

this.picture7.AllowDrop = true; this.picture8.AllowDrop = true;

this.picture9.AllowDrop = true; this.picture10.AllowDrop = true;

this.picture11.AllowDrop = true; this.picture12.AllowDrop = true;

this.picture13.AllowDrop = true; this.picture14.AllowDrop = true;

this.picture15.AllowDrop = true; this.picture16.AllowDrop = true;

}

Эти операторы можно заменить на цикл:

for (int i = 1; i <= 16; i++)

((PictureBox)(Controls.Find("picture" +

i,true)[0])).AllowDrop = true;

Процесс перетаскивания инициируется, когда пользователь щелкает мышью по любому из компонентов PictureBox. Создадим обработчик события MouseDown для компонента *рicture1*. Здесь мы должны инициализировать процесс перетаскивания с помощью метода *DoDragDrop*(). Это можно сделать следующим образом:

DoDragDrop(picture1, DragDropEffects.Move);

Обратите внимание, что в качестве перетаскиваемых данных мы запоминаем компонент *picture1*. Однако при таком способе решения нам потребуется написать еще 15 аналогичных обработчиков события MouseDown для каждого из фрагментов картинки. Такой способ решения является достаточно трудоемким.

Поэтому в методе *DoDragDrop*() будем указывать именно тот компонент PictureBox, для которого возникло событие MouseDown. В результате обработчик будет иметь следующий вид:

private void picture1\_MouseDown

(object sender, MouseEventArgs e)

{

DoDragDrop((PictureBox)sender, DragDropEffects.Move);

}

Далее создадим обработчик события DragEnter для компонента picture1. Напомним, что это событие возникает, когда в процессе перетаскивания курсор мыши попадает в пределы объекта‑приемника. В этом случае нам необходимо задать допустимую операцию перетаскивания – перемещение (*Move*).

private void picture1\_DragEnter

(object sender, DragEventArgs e)

{

e.Effect = DragDropEffects.Move;

}

Создадим обработчик события DragDrop, завершающего процесс перетаскивания. Здесь нам нужно поменять местами картинки в объекте‑исто­чнике и объекте‑приемнике:

private void picture1\_DragDrop(object sender, DragEventArgs e)

{

//объект‑приемник

PictureBox receiver = (PictureBox)sender;

//объект‑источник

PictureBox source =

(PictureBox)e.Data.GetData((typeof(PictureBox)));

//Temp ‑ вспомогательная переменная

//для хранения изображения

Image Temp = receiver.Image;

//меняем картинки местами

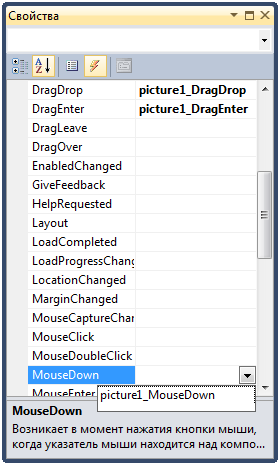
receiver.Image = source.Image;

source.Image = Temp;

}

Обратите внимание, что объект‑источник определяется с помощью метода *GetData*(), предоставляющего доступ к данным об объекте‑источнике, которые мы запоминали при инициализации операции перетаскивания.

Для того чтобы описанные нами операции выполнялись для любого фрагмента паззла, необходимо в окне Свойств на вкладке Событийкаждое из трех событий (MouseDown, DragEnter и DragDrop) для каждого из оставшихся 15 компонентов PictureBox связать с написанными нами обработчиками соответствующих событий. Для быстроты можно выделить все оставшиеся 15 компонентов (нажимая клавишу *Ctrl*), а потом связывать нужные события (рис. 9.3).



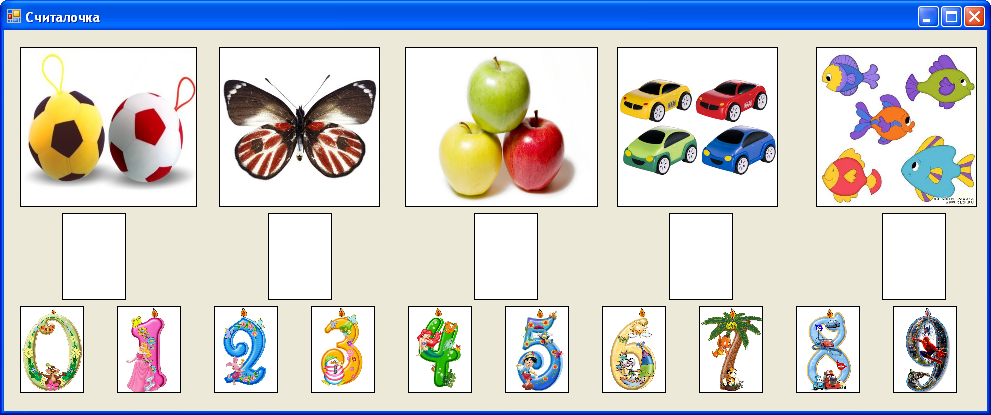
*Рис. 9.3.* Установка связи между событиями и обработчиками событий

**Задания для самостоятельного выполнения**

1. Добавьте в игру «Паззл» следующие возможности:

* в начале игры части картинки переставляются в произвольном порядке;
* когда картинка собрана, выдается сообщение с поздравлением и приложение закрывается;
* добавьте к приложению еще какую‑либо возможность по вашему усмотрению (например, какая‑то реакция, когда кусочек картинки перекладывается на правильное место; отсутствие возможности перекладывать частичку картинки на неправильное место; возможность просмотра полной картинки; возможность выбора из нескольких картинок; возможность разбиения картинки на разное количество кусочков и т. д.).

1. Напишите любое из трех следующих приложений:
   * 1. Приложение «Считалочка» (рис. 9.4). Необходимо сосчитать количество предметов, изображенных на рисунке, и в ячейку под рисунком перетащить нужную цифру. Если цифра выбрана правильно, она помещается в ячейку, в противном случае выводится сообщение об ошибке. При нажатии на кнопку *F*1 выводится сообщение с описанием правил игры. В конце игры вывести сообщение с поздравлениями.



*Рис. 9.4.* Приложение «Считалочка»

* + 1. Приложение «Читарик». Приложение разделено на две части: слева располагаются картинки (не менее пяти), а справа – слова. Необходимо перетащить каждое слово на нужное место.
    2. Приложение «Словарик». Приложение разделено на две части. Слева располагается ряд слов на английском языке (не менее пяти), справа – на русском. Необходимо каждое русское слово перетащить на нужное место.

1. «Ханойские башни». Имеются три колышка *A*, *B*, *C* и *n* дисков разного размера. Сначала все диски надеты на колышек *A* в виде пирамиды. Цель игрока – перенести все диски с колышка *A* на колышек *C*, соблюдая при этом следующие условия: диски можно переносить только по одному, больший диск нельзя ставить на меньший. Написать приложение, реализующее эту игру. Приложение должно подсчитывать количество перемещений, а также число минимально возможных перемещений и показывать эти значения по окончании игры.

**Баллы:**  каждая из трех задач оценивается по два балла.