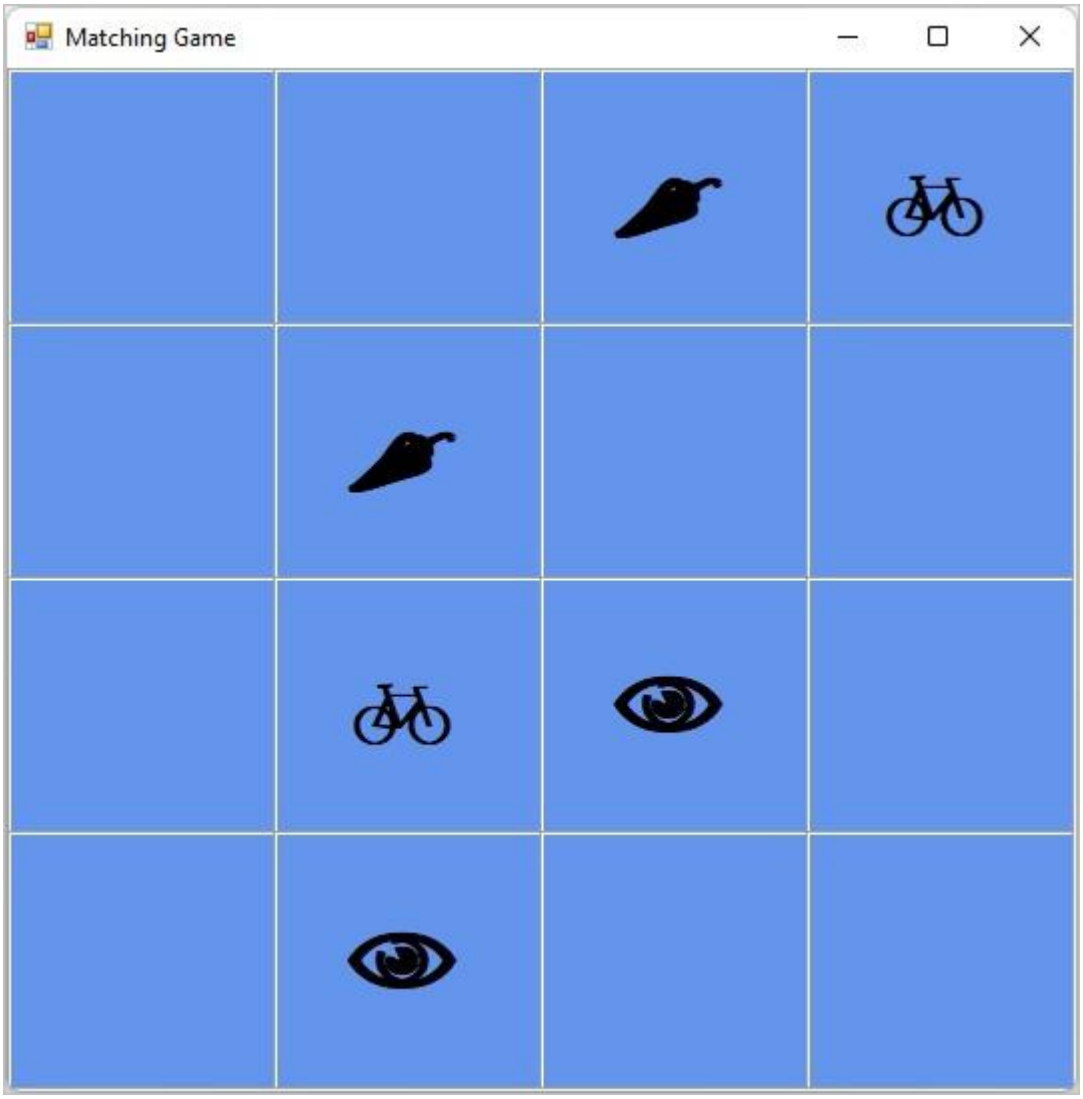
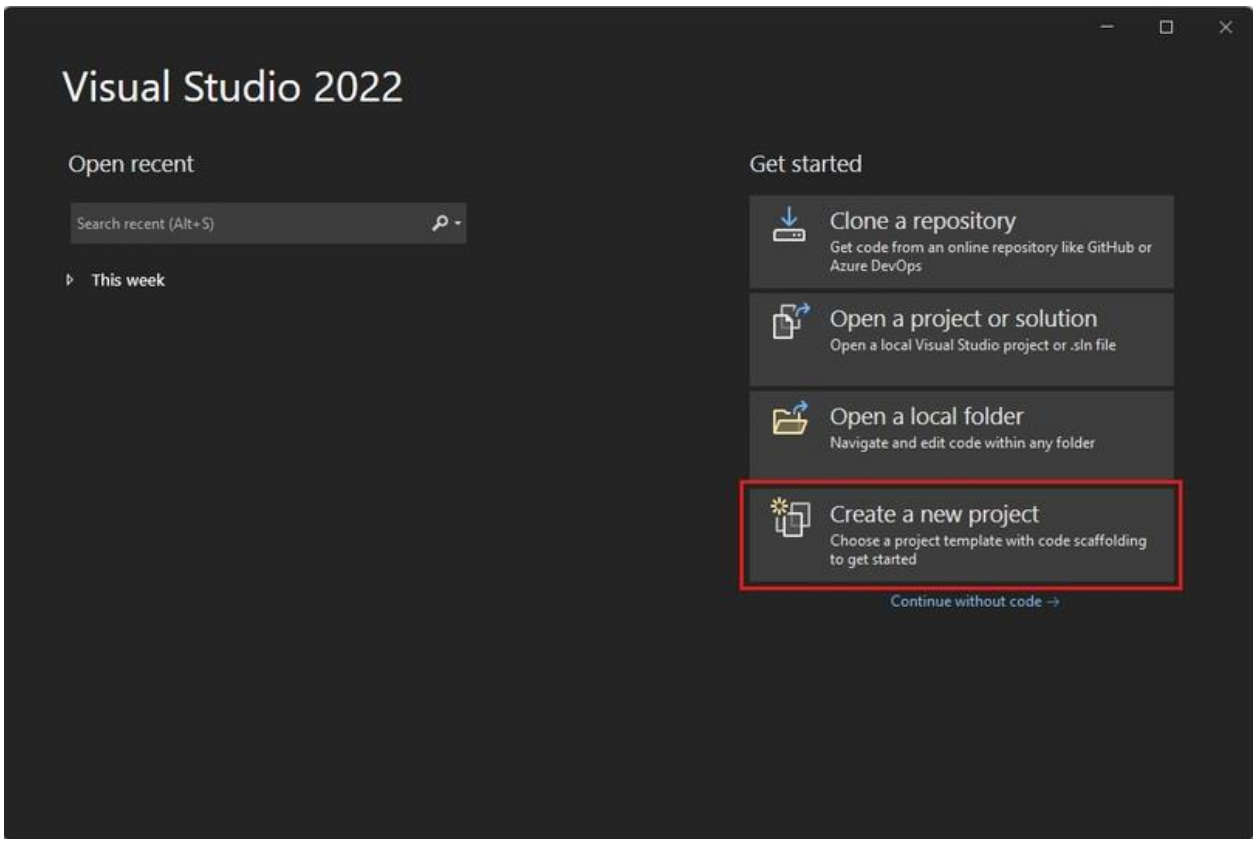


Создание проекта игры "Подбери пару" Windows Forms

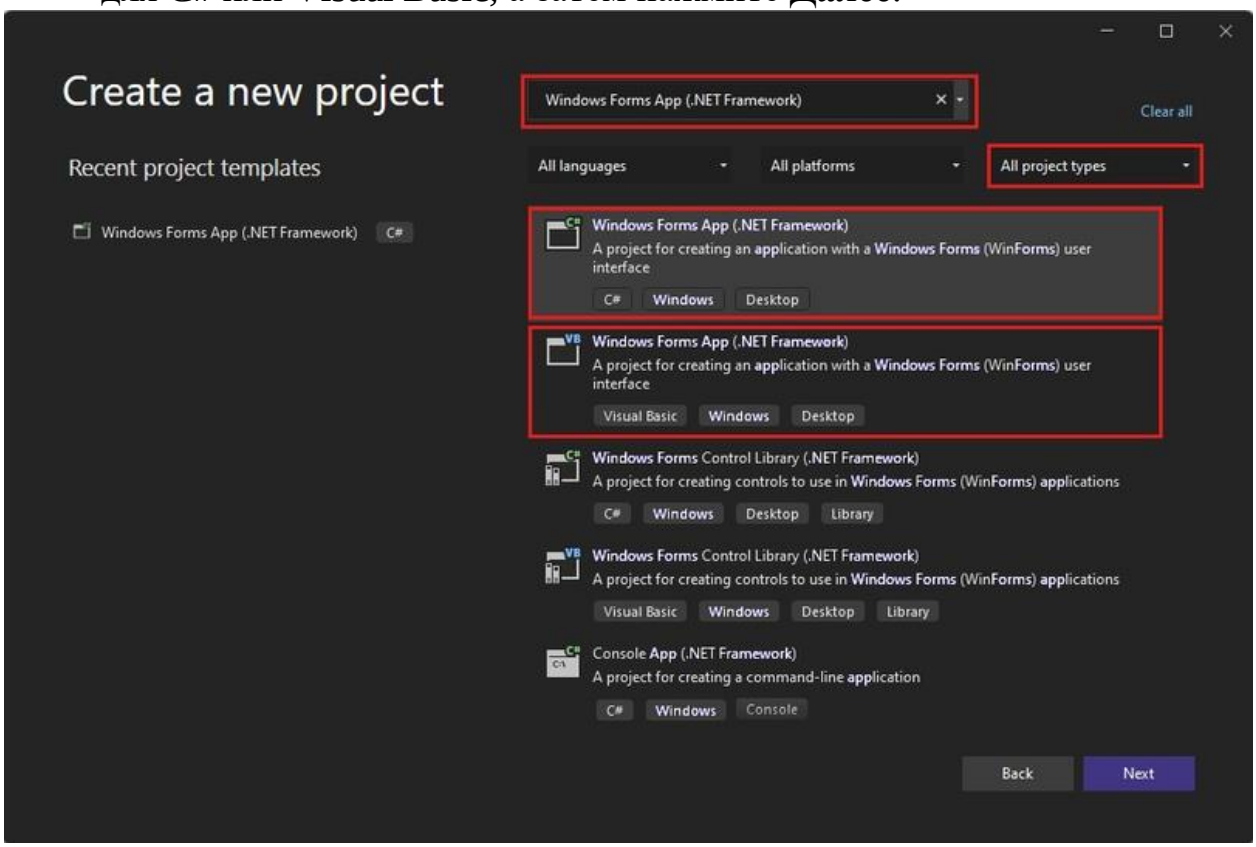
Первый шаг в создании игры "Подбери пару" — это создание проекта приложения Windows Forms.

1. Запустите Visual Studio.
2. В окне запуска выберите **Создание нового проекта**.



3. В окне **Создать проект** выполните поиск по фразе *Windows Forms*. Затем выберите пункт **Классические** в списке **Все типы проектов**.

4. Выберите шаблон **Приложение Windows Forms (.NET Framework)** для C# или Visual Basic, а затем нажмите **Далее**.



5. В окне **Настроить новый проект** назовите проект *MatchingGame*, а затем выберите **Создать**.

Visual Studio создает решение для приложения. Решение является контейнером для всех проектов и файлов, необходимых приложению.

На этом этапе Visual Studio отображает пустую форму в **конструкторе Windows Forms**.

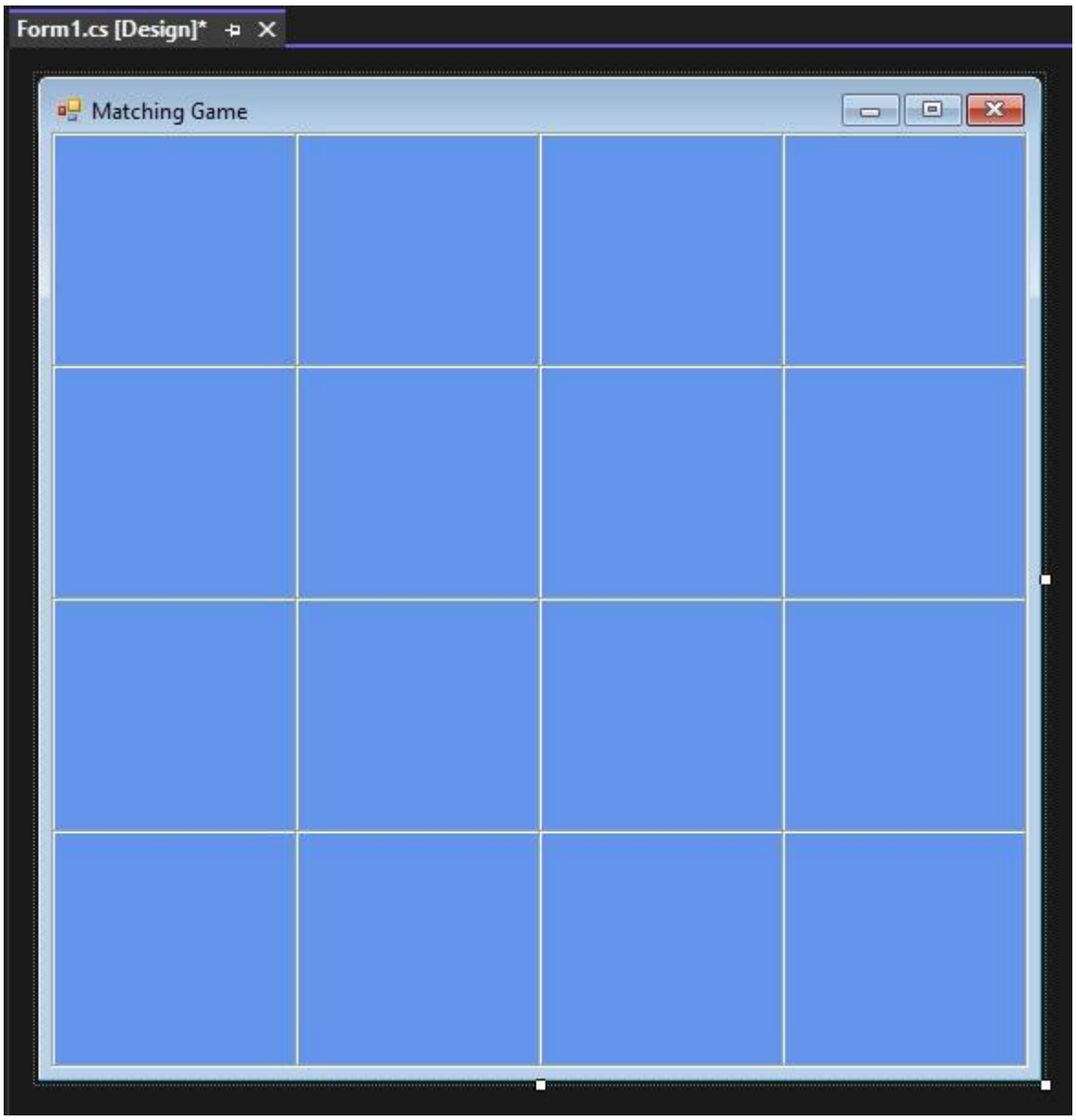
Создание макета для игры

В этом разделе вы создадите для игры сетку "четыре на четыре".

1. Щелкните форму, чтобы выбрать конструктор Windows Forms. На этой вкладке для C# считывается файл **Form1.cs [Design]**. В окне **Свойства** задайте следующие значения свойств формы.
    a. Измените свойство **Text** с **Form1** на **Matching Game**. Этот текст отображается в верхней части окна игры.
    b. Задайте размер формы. Вы можете изменить его либо задав для свойства **Size** значение **550, 550**, либо перетягивая угол формы до тех пор, пока вы не увидите правильный размер в нижней части IDE Visual Studio.
2. Выберите вкладку **Панель элементов** в левой части интегрированной среды разработки. Если она не отображается, выберите **Представление**>**Панель элементов** в строке меню или нажмите сочетание клавиш **Ctrl**+**Alt**+**X**.
3. Перетащите элемент управления **TableLayoutPanel** из категории **Контейнеры** на панели элементов или дважды щелкните его. В окне **Свойства** задайте следующие свойства для панели.
    a. Задайте для свойства **BackColor** значение **CornflowerBlue**. Чтобы задать это свойство, щелкните стрелку рядом со свойством **BackColor**. В диалоговом окне **BackColor** выберите **Интернет**. Выберите **CornflowerBlue** в списке названий доступных цветов.
    b. Выберите для свойства **Dock** значение **Заливка** из раскрывающегося списка, нажав большую кнопку, расположенную посередине. Этот параметр позволяет растянуть таблицу по всей форме.
    c. Для свойства **CellBorderStyle** установите значение **Inset**. Задание этого значения приведет к тому, что между ячейками поля появятся видимые границы.

d. Нажмите треугольную кнопку в правом верхнем углу элемента управления TableLayoutPanel для отображения меню задач этой панели. В меню задачи щелкните команду **Добавить строку** дважды, чтобы добавить еще две строки. Затем дважды щелкните пункт **Добавить столбец**, чтобы добавить еще два столбца.
e. В меню задач выберите команду **Правка строк и столбцов**, чтобы открыть окно **Стили столбцов и строк**. Для каждого столбца выберите параметр **Процент**, а затем задайте для каждого столбца ширину *25* процентов.
f. Затем в списке в верхней части окна выберите пункт **Строки** и задайте высоту каждой строки равной 25 процентам.
g. Задав все параметры, нажмите кнопку **ОК**, чтобы сохранить изменения.

Элемент управления TableLayoutPanel теперь представляет собой сетку "четыре на четыре" с 16 квадратными ячейками одинакового размера. Эти строки и столбцы задают места, в которых позже появятся значки.



Добавление и форматирование меток для отображения

В этом разделе вы создадите и отформатируете метки, которые будут отображатся во время игры.

1. Убедитесь, что TableLayoutPanel выбран в редакторе формы. Вы должны увидеть элемент управления **tableLayoutPanel1** в верхней части окна **Свойства**. Если он не выбран, выберите элемент управления TableLayoutPanel в форме или из списка в верхней части окна **Свойства**.
2. Откройте панель элементов, как и прежде, а затем — категорию **Стандартные элементы управления**. Добавьте элемент управления

Label в верхнюю левую ячейку TableLayoutPanel. Теперь элемент управления label выбран в интегрированной среде разработки. Задайте для него следующие свойства.Теперь в левой верхней ячейке TableLayoutPanel располагается черный квадрат на синем фоне, который выравнивается по центру.

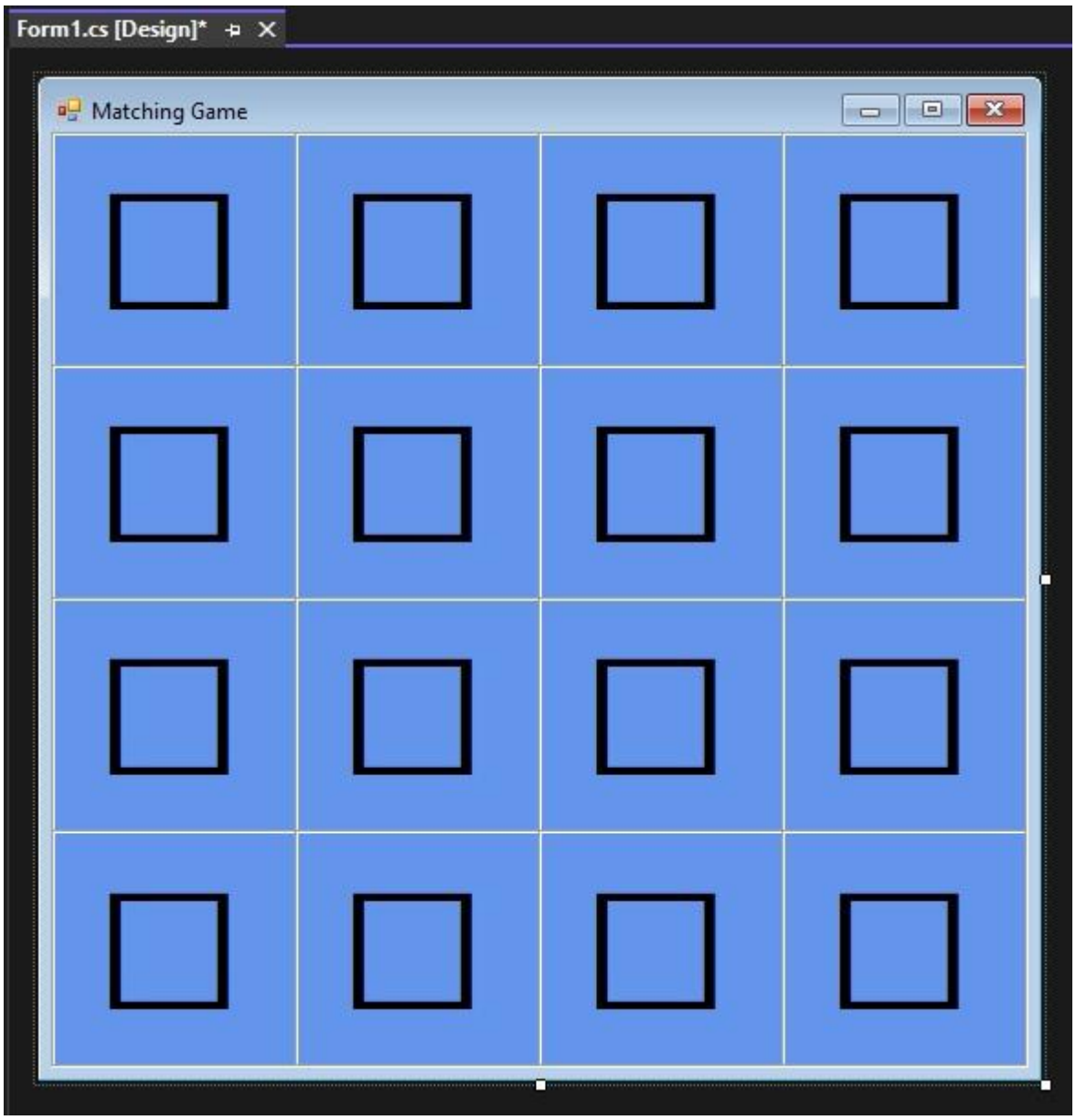
a. Задайте для свойства **BackColor** метки значение **CornflowerBlue**.
b. Задайте свойству **AutoSize** значение **False**.
c. Задайте для свойства **Dock** значение **Fill**.
d. Задайте для свойства **TextAlign** значение **MiddleCenter**, нажав кнопку раскрывающегося списка рядом со свойством, а затем щелкнув среднюю кнопку. Это значение необходимо, чтобы значок отображался в середине ячейки.
e. Выберите свойство **Font**. Появится кнопка с многоточием (**…**). Нажмите многоточие и задайте для параметра **Font** значение **Webdings**, для параметра **Font Style** — значение **Bold**, а для параметра **Size** — значение **48**.
f. Установите свойство **Text** равным букве **c**.

Webdings — шрифт значков, который поставляется с операционной системой Windows. В игре "Подбери пару" игроку нужно подобрать пару для значков. Этот шрифт отображает значки, для которых нужно подобрать пары.

*Вместо **c** попробуйте использовать в свойстве **Text** разные буквы. Восклицательный знак соответствует пауку, прописная буква N — глазу, а запятая — перцу чили.*

3. Выберите элемент управления Label и скопируйте его в следующую ячейку TableLayoutPanel. (Нажмите сочетание клавиш **Ctrl**+**C** или в строке меню выберите **Правка**>**Копировать**.) Затем вставьте его с помощью комбинации клавиш **CTRL**+**V** или выберите **Правка**>**Вставить**.Во второй ячейке элемента управления TableLayoutPanel появится копия первого элемента управления Label. Вставьте его снова, и в третьей ячейке появится еще один элемент управления Label. Продолжайте вставлять элементы управления Label, пока все ячейки не будут заполнены.

Этот шаг завершает создание макета вашей формы.



Добавление случайного объекта и списка значков

В этом разделе вы создадите набор парных символов для игры. Каждый символ добавляется в две случайные ячейки в TableLayoutPanel в форме. Вам понадобится использовать два оператора `new`, создающие два объекта. Первый — это объект Random, который случайным образом выбирает ячейки в элементу управления TableLayoutPanel. Второй объект является объектом List<T>. В нем хранятся случайно выбранные символы.

1. Запустите Visual Studio. Проект игры "Подбери пару" находится в разделе **Открыть последние**.

2. Выберите файл *Form1.cs*, если вы используете C#. Затем выберите **Представление**>**Код**. Также можно нажать клавишу **F7** или дважды щелкнуть **Form1**. В интегрированной среде разработки Visual Studio будет отображатся модуль кода для Form1.
3. В имеющийся код добавьте следующий фрагмент.

```
public partial class Form1 : Form
{
    // Use this Random object to choose random icons for the squares
    Random random = new Random();
    // Each of these letters is an interesting icon
    // in the Webdings font,
    // and each icon appears twice in this list
    List<string> icons = new List<string>()
    {
        "!", "!", "N", "N", ",", ",", "k", "k",
        "b", "b", "v", "v", "w", "w", "z", "z"
    };
```

При написании кода на языке C# убедитесь, что вы помещаете код после открывающей фигурной скобки и сразу после объявления класса (`public partial class Form1 : Form`). При написании кода на языке Visual Basic поместите код сразу после объявления класса (`Public Class Form1`).

Объекты List можно использовать для отслеживания элементов различных типов. Список может содержать числа, значения true или false, текст и другие объекты. В игре "Подбери пару" объект list содержит 16 строк, по одной для каждой ячейки на панели TableLayoutPanel. Каждая строка представляет собой одну букву, соответствующую значкам в метках. Эти символы отображаются в шрифте Webdings в виде автобуса, велосипеда и т. д.
Списки можно уменьшать и увеличивать по мере необходимости, что важно для этой программы.

Назначение каждому элементу управления Label случайного значка

При каждом запуске программы значки назначаются элементам управления Label в форме случайным образом с помощью метода `AssignIconsToSquares()`. Этот код использует ключевое слово `foreach` на C# или `For Each` на Visual Basic.

1. Добавьте метод `AssignIconsToSquares()`.

```
/// <summary>
/// Assign each icon from the list of icons to a random square
/// </summary>
private void AssignIconsToSquares()
{
    // The TableLayoutPanel has 16 labels,
    // and the icon list has 16 icons,
    // so an icon is pulled at random from the list
    // and added to each label
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            int randomNumber = random.Next(icons.Count);
            iconLabel.Text = icons[randomNumber];
            // iconLabel.ForeColor = iconLabel.BackColor;
            icons.RemoveAt(randomNumber);
        }
    }
}
```

Этот код можно ввести непосредственно под кодом, добавленным в предыдущем разделе.
Одна из строк закомментирована специально. Вы добавите ее позже в этой процедуре.

Метод `AssignIconsToSquares()` выполняет итерацию каждого элемента управления label в TableLayoutPanel. Он выполняет одни и те же операторы для каждого из этих элементов управления. Эти операторы запрашивают случайные значки из списка.

- Первая строка преобразует переменную **control** в метку с именем **iconLabel**.
- Вторая строка представляет собой оператор `if`, проверяющий и обеспечивающий успешное выполнение преобразования. Если преобразование выполняется, выполняются операторы в операторе `if`.
- Первая строка в операторе `if` создает переменную с именем **randomNumber**, содержащую случайное число, соответствующее одному из элементов списка значков. Здесь используется метод Next() объекта Random. Метод `Next` возвращает случайное число. Эта строка также использует свойство Count списка **значков** для определения диапазона, из которого выбирается случайное число.
- В следующей строке один из элементов списка значков присваивается свойству Text этой метки.

- Следующая строка скрывает значки. Эта строка будет закомментирована, так что вы сможете проверить оставшуюся часть кода, прежде чем продолжить работу.
- Последняя строка в операторе `if` удаляет из списка значок, добавленный в форму.

1. Добавьте вызов метода `AssignIconsToSquares()` в **конструктор***Form1*. Этот метод заполняет игровую доску значками. Конструкторы вызываются при создании объекта.

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

2. Сохраните и выполните программу. Должна отобразиться форма со случайными значками, которые назначены каждой метке.

Если значки Webdings не отображаются в форме правильно, установите для свойства ***UseCompatibleTextRendering*** *меток в форме значение* ***True****.*

3. Закройте программу, а затем снова запустите ее. Каждой метке назначены разные значки.

Значки видимы, поскольку они не были скрыты. Чтобы скрыть их от игрока, можно задать для свойства **ForeColor** каждого элемента управления Label тот же цвет, что и у свойства **BackColor**.

4. остановка программы; Удалите метки комментариев с соответствующей строки кода в цикле.

```
iconLabel.ForeColor = iconLabel.BackColor;
```

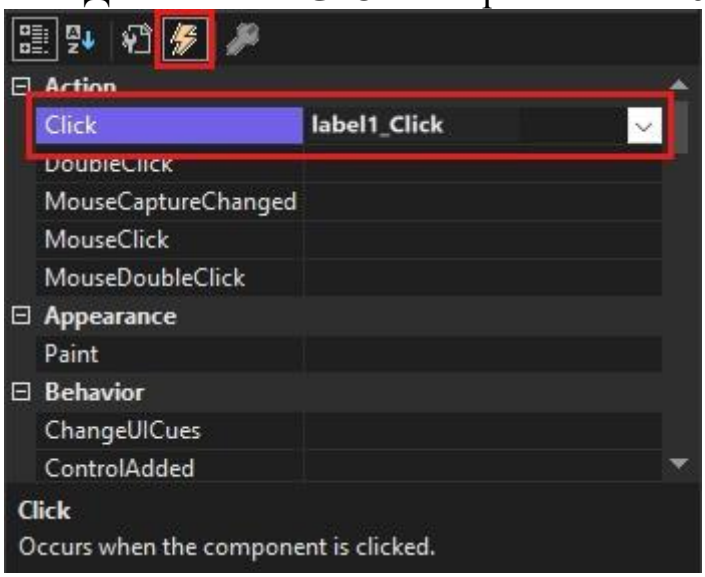
Если вы снова запустите программу, создастся впечатление, что значки исчезли. И будет отображатся только синий фон. Однако значки назначены случайным образом и по-прежнему существуют.

Добавление обработчиков событий в метки

В этой игре "Подбери пару" игрок открывает один скрытый значок, а затем второй. Если значки совпадают, они остаются видимыми. Если нет, оба значка снова скрываются.

Чтобы заставить вашу игру работать таким образом, добавьте обработчик события **Click**, который изменяет цвет выбранной метки в соответствии с фоном.

1. Откройте форму в **конструкторе Windows Forms**. Выберите **Form1.cs** или **Form1.vb**, а затем выберите **Представление**>**Конструктор**.
2. Выберите первый элемент Label, чтобы выделить его. Затем, удерживая нажатой клавишу **Ctrl**, выберите каждую из оставшихся меток. Убедитесь, что выбраны все метки.
3. В окне **Свойства** нажмите кнопку **События**, которая представляет собой молнию. Для события **Click** выберите в поле **label1_Click**.



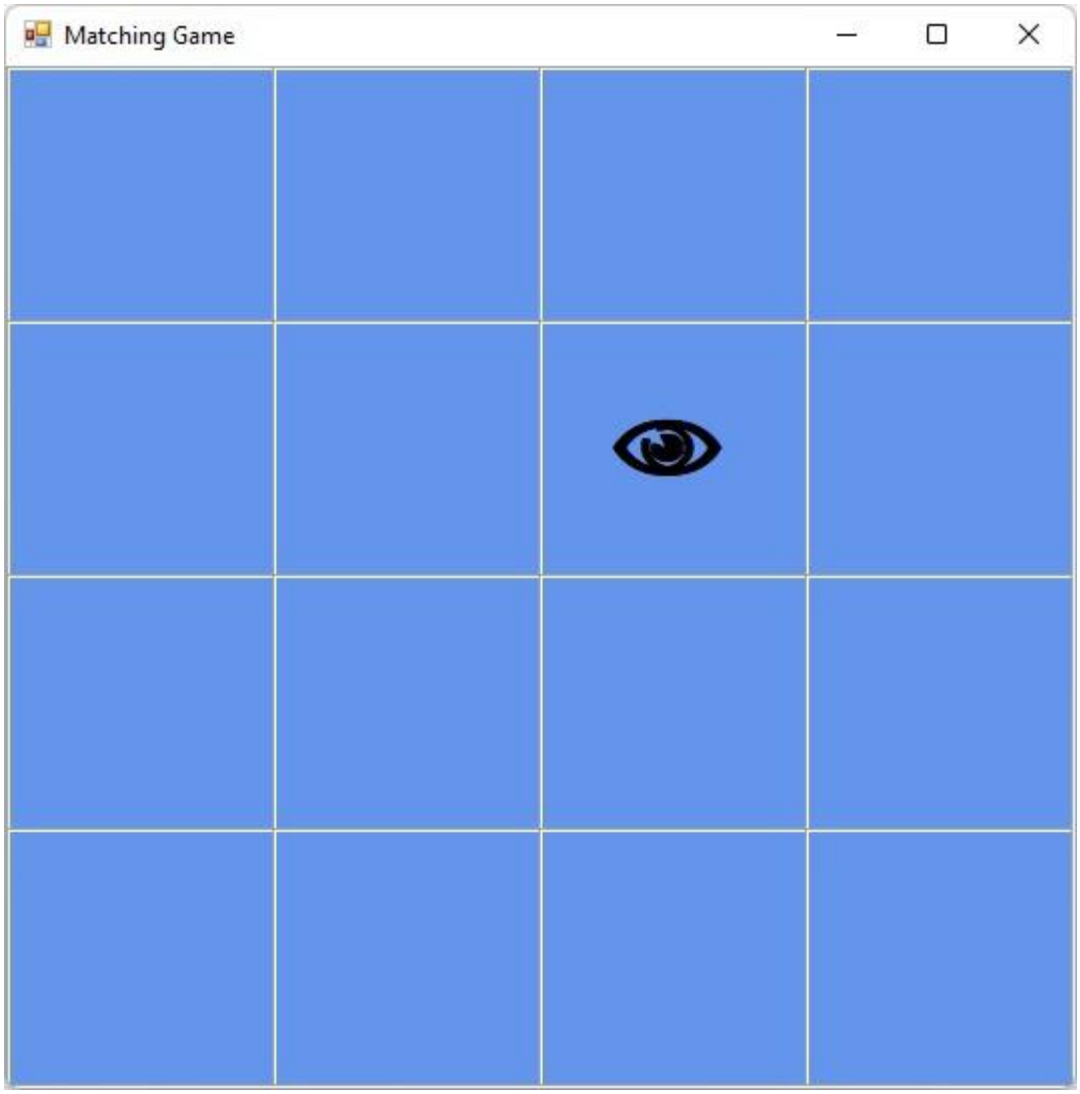
4. Нажмите клавишу **ВВОД**. Интегрированная среда разработки добавляет в код обработчик события `Click` с именем **label1_Click()** . Поскольку вы выбрали все метки, обработчик будет привязан к каждой из этих меток.

5. Добавьте остальную часть кода.

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label1_Click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;
    if (clickedLabel != null)
    {
        // If the clicked label is black, the player clicked
        // an icon that's already been revealed --
        // ignore the click
        if (clickedLabel.ForeColor == Color.Black)
            return;
        clickedLabel.ForeColor = Color.Black;
    }
 }
```

При копировании и вставке блока кода `label1_Click()` вместо его ввода вручную проследите за тем, что заменить существующий код `label1_Click()`. В противном случае в коде появится дублирующий блок.

Выберите пункт **Отладка**>**Начать отладку**, чтобы запустить программу. Вы должны увидеть пустую форму с синим фоном. Выберите любую из ячеек в форме. После этого должен отобразится один из значков. Продолжайте выбирать различные ячейки формы. Значки будут отображаться при их выборе.



Добавление ссылок на элементы управления Label

В этом разделе вы добавите в код две *ссылочные переменные*. Они будут отслеживать (или ссылаться на) объекты Label.

1. Добавьте ссылки на метки в свою форму, используя следующий код.

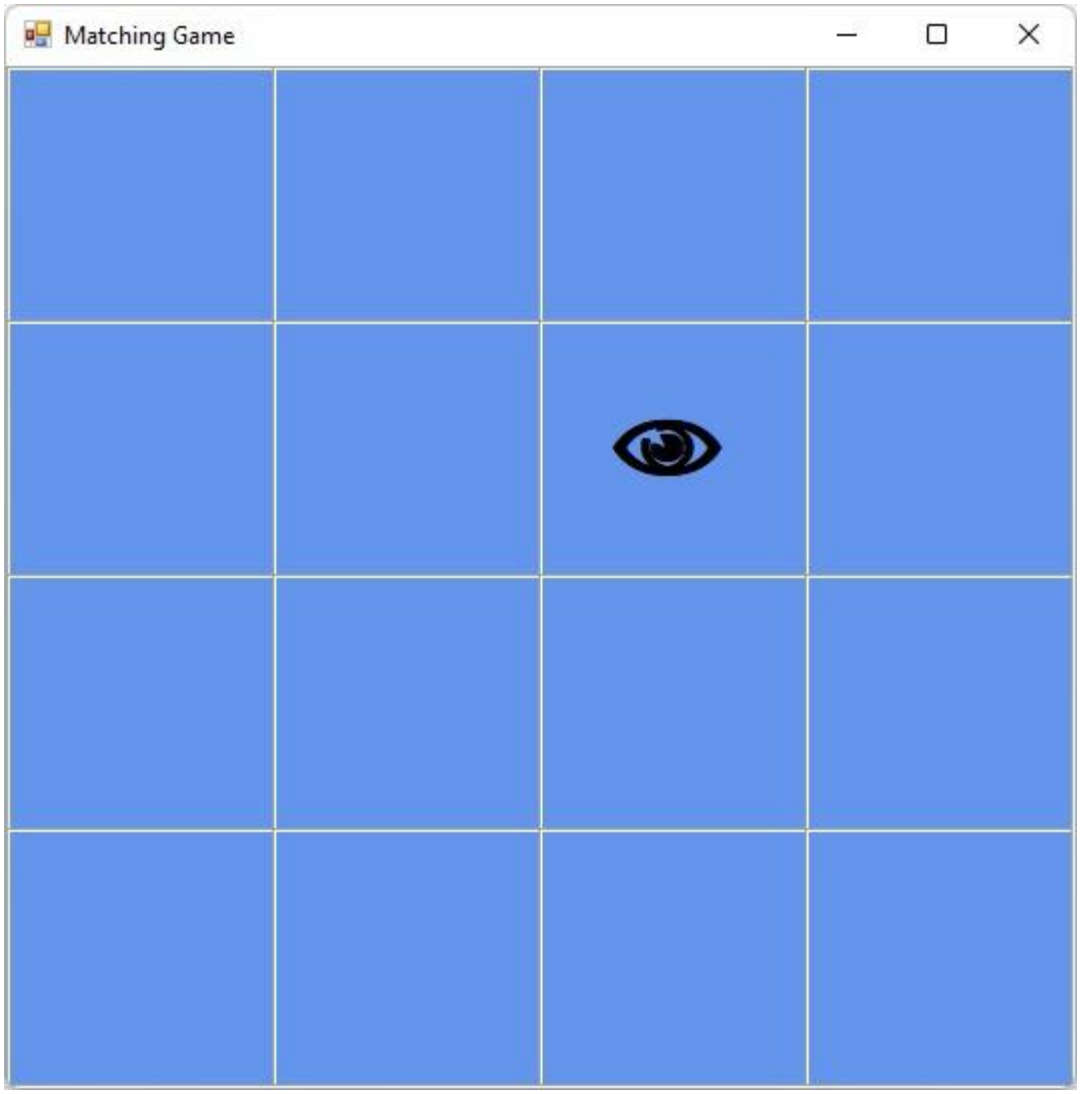
```
public partial class Form1 : Form
{
    // firstClicked points to the first Label control
    // that the player clicks, but it will be null
    // if the player hasn't clicked a label yet
    Label firstClicked = null;
    // secondClicked points to the second Label control
    // that the player clicks
    Label secondClicked = null;
```

Эти операторы не приводят к отображению элементов управления Label на форме, поскольку вы не указываете ключевое слово `new`. Когда программа запускается, и `firstClicked`, и `secondClicked` имеют значение `null` для C# или `Nothing` для Visual Basic.

1. Измените свой обработчик событий Click для использования новой ссылочной переменной `firstClicked`. Удалите последнюю инструкцию в методе обработчика событий `label1_Click()` (`clickedLabel.ForeColor = Color.Black;`) и замените ее выражением `if`, как показано ниже.

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label1_Click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;
    if (clickedLabel != null)
    {
        // If the clicked label is black, the player clicked
        // an icon that's already been revealed --
        // ignore the click
        if (clickedLabel.ForeColor == Color.Black)
            return;
        // If firstClicked is null, this is the first icon
        // in the pair that the player clicked,
        // so set firstClicked to the label that the player
        // clicked, change its color to black, and return
        if (firstClicked == null)
        {
            firstClicked = clickedLabel;
            firstClicked.ForeColor = Color.Black;
            return;
        }
    }
}
```

Сохраните и выполните программу. Выберите одну из меток и появится ее значок. Выберите следующую метку и обратите внимание, что ничего не происходит.



1. При этом черным становится только первый выбранный значок. Другие значки останутся невидимыми.

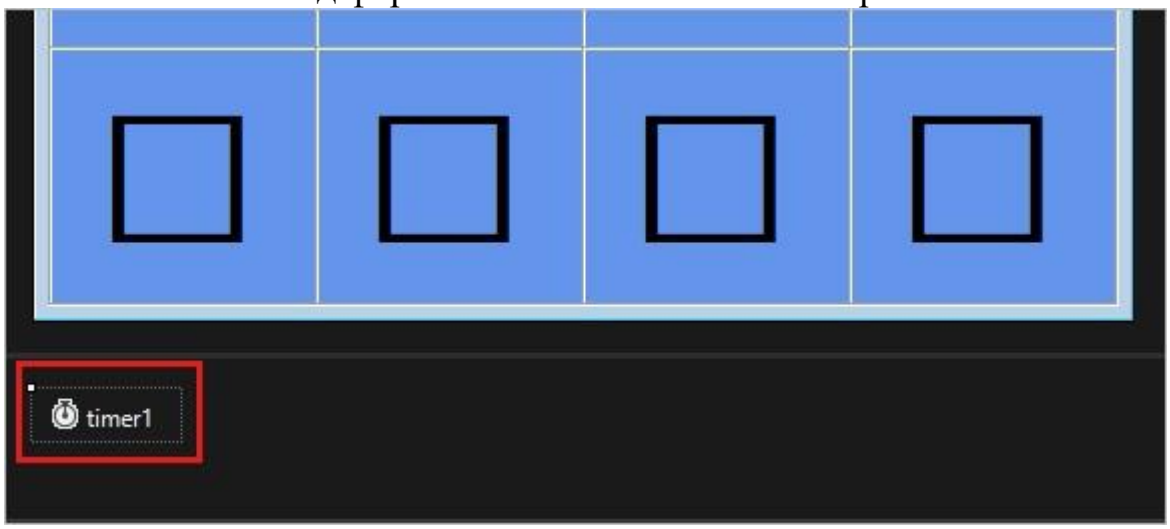
Программа уже отслеживает первую метку, которую выбрал игрок. Поэтому ссылка `firstClicked` не равна `null` на C# или `Nothing` на Visual Basic. Когда выражение `if` обнаруживает, что `firstClicked` не равно `null` или `Nothing`, оно выполняет последующие инструкции.

Добавление таймера

Игра "Подбери пару" использует для управления приложением **Timer**. Таймер ожидает, а затем вызывает событие, называемое *тактом*. Он может запускать действие или регулярно повторять его.

При использовании в программе таймер позволяет игроку выбрать два значка. Если эти значки не совпадают, программа скроет их по истечении короткого периода времени.

1. Выберите вкладку **Панель элементов**, в категории **Компоненты** дважды щелкните или перетащите компонент **Таймер** в свою форму. В области под формой появится значок таймера с именем **timer1**.



2. Щелкните значок **Timer1**, чтобы выбрать таймер. В окне **Свойства** выберите кнопку **Свойства**, чтобы просмотреть соответствующие параметры.

3. Задайте для свойства **Interval** значение **750**, то есть 750 миллисекунд.Свойство **Interval** задает время ожидания между *тактами* таймера во время активации события Tick. Ваша программа вызывает метод Start() для запуска таймера только после того, как игрок выберет вторую метку.

4. Выберите значок элемента управления timer, а затем нажмите **Ввод** или дважды щелкните таймер. Затем интегрированная среда разработки добавит пустой обработчик событий Tick. Замените код следующим кодом.

```
/// <summary>
/// This timer is started when the player clicks
/// two icons that don't match,
/// so it counts three quarters of a second
/// and then turns itself off and hides both icons
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void timer1_Tick(object sender, EventArgs e)
{
    // Stop the timer
    timer1.Stop();
    // Hide both icons
    firstClicked.ForeColor = firstClicked.BackColor;
    secondClicked.ForeColor = secondClicked.BackColor;
    // Reset firstClicked and secondClicked
    // so the next time a label is
    // clicked, the program knows it's the first click
    firstClicked = null;
    secondClicked = null;
}
```

Обработчик события Tick выполняет три действия.

- Останавливает таймер, вызывая метод Stop().
- Использует две ссылочные переменные, `firstClicked` и `secondClicked`, чтобы снова сделать невидимыми значки двух меток, которые выбрал игрок.
- Сбрасывает значения ссылочных переменных `firstClicked` и `secondClicked` на `null` в C# и `Nothing` в Visual Basic.

1. Перейдите в редактор кода и добавьте в начало и конец метода обработчика событий `label1_Click()` следующий код. Этот код проверяет, включен ли таймер, задает ссылочную переменную `secondClicked` и запускает таймер. Метод обработчика событий `label1_Click()` теперь выглядит так:

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label1_Click(object sender, EventArgs e)
{
    // The timer is only on after two non-matching
    // icons have been shown to the player,
    // so ignore any clicks if the timer is running
    if (timer1.Enabled == true)
        return;
    Label clickedLabel = sender as Label;
    if (clickedLabel != null)
    {
        // If the clicked label is black, the player clicked
        // an icon that's already been revealed --
        // ignore the click
        if (clickedLabel.ForeColor == Color.Black)
            return;
        // If firstClicked is null, this is the first icon
        // in the pair that the player clicked,
        // so set firstClicked to the label that the player
        // clicked, change its color to black, and return
        if (firstClicked == null)
        {
            firstClicked = clickedLabel;
            firstClicked.ForeColor = Color.Black;
            return;
        }
        // If the player gets this far, the timer isn't
        // running and firstClicked isn't null,
        // so this must be the second icon the player clicked
        // Set its color to black
        secondClicked = clickedLabel;
        secondClicked.ForeColor = Color.Black;
        // If the player gets this far, the player
        // clicked two different icons, so start the
        // timer (which will wait three quarters of
        // a second, and then hide the icons)
        timer1.Start();
    }
}
```

- Код в начале метода проверяет, запущен ли таймер, обращаясь к значению свойству **Enabled**. Если игрок выбирает первый и второй элемент управления Label и таймер запускается, выбор третьего элемента управления Label ни к чему не приведет.
- Код в конце метода задает ссылочную переменную `secondClicked` для отслеживания второго элемента управления Label. И затем присваивает значку ярлыка черный цвет, чтобы сделать его видимым. Затем таймер запускается в однократном режиме, то есть

ожидает 750 миллисекунд и после этого вызывает одно событие Tick. Обработчик события Tick таймера скрывает два значка и сбрасывает ссылочные переменные `firstClicked` и `secondClicked`. Теперь форма готова к тому, чтобы игрок выбрал другую пару значков.

При копировании и вставке блока кода `label1_Click()` вместо его ввода вручную проследите за тем, что заменить существующий код `label1_Click()`. В противном случае в коде появится дублирующий блок.

1. Сохраните и выполните программу. Нажмите квадрат, и значок станет видимым. А теперь нажмите другой квадрат. Значок появиться на короткое время, а затем оба значка исчезнут.

Теперь ваша программа может отслеживать выбор первого и второго значков. А также использует таймер для приостановки перед исчезновением значков.

Отмена исчезновения пар значков

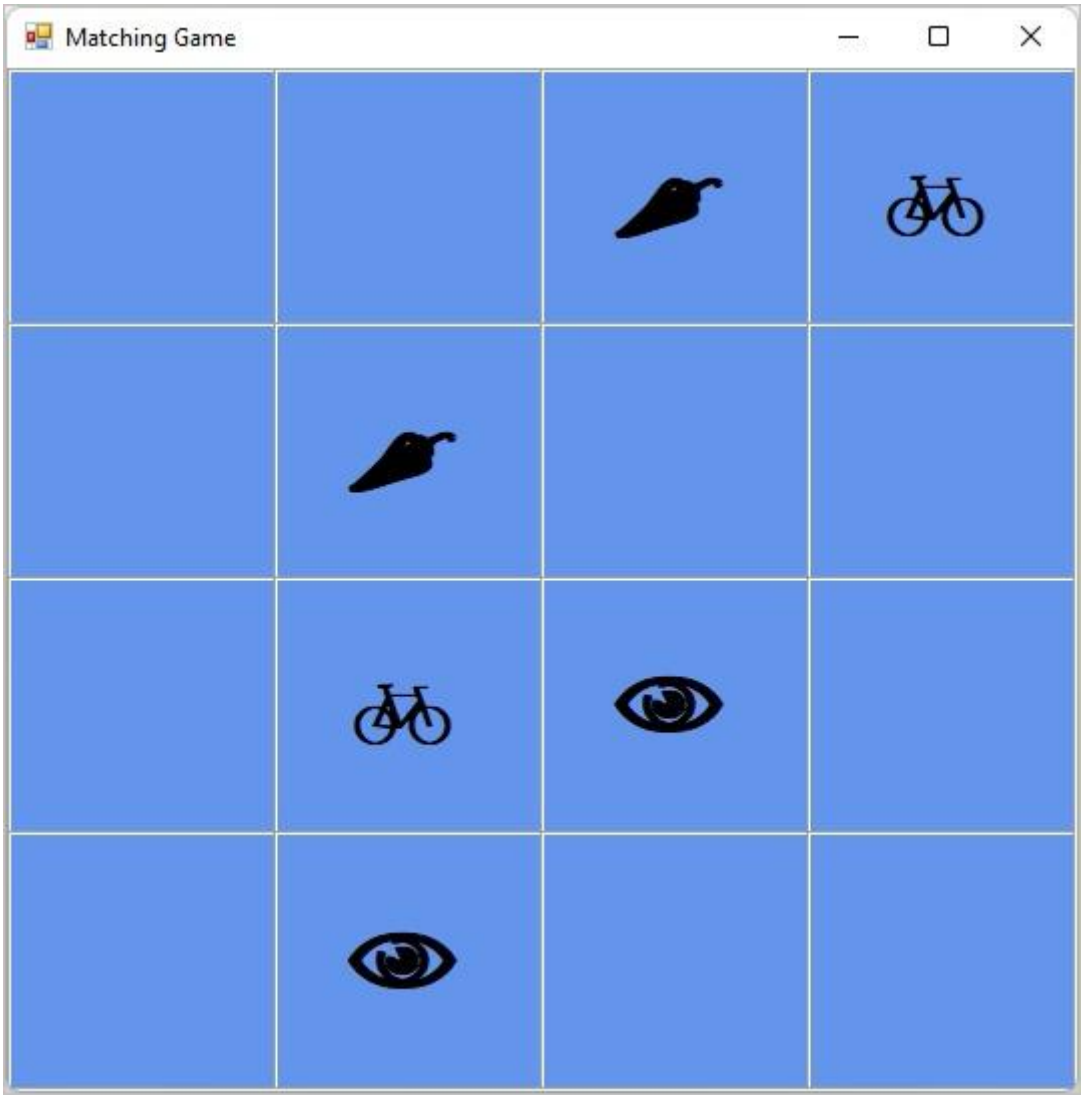
Если игрок подобрал пару, игра должна перезагрузиться, чтобы больше не отслеживать метки, использующие ссылочные переменные `firstClicked` и `secondClicked`. Однако она не должна сбрасывать цвета для двух совпадающих меток. Эти метки должны и дальше отображаться.

1. Добавьте следующий оператор `if` в метод обработчика событий `label_Click()`. Разместите его ближе к концу кода непосредственно над оператором, запускающим таймер.

```
// If the player gets this far, the timer isn't
        // running and firstClicked isn't null,
        // so this must be the second icon the player clicked
        // Set its color to black
        secondClicked = clickedLabel;
        secondClicked.ForeColor = Color.Black;
        // If the player clicked two matching icons, keep them
        // black and reset firstClicked and secondClicked
        // so the player can click another icon
        if (firstClicked.Text == secondClicked.Text)
        {
            firstClicked = null;
            secondClicked = null;
            return;
        }
        // If the player gets this far, the player
        // clicked two different icons, so start the
        // timer (which will wait three quarters of
        // a second, and then hide the icons)
        timer1.Start();
    }
}
```

Первый оператор `if` проверяет, совпадает ли значок в первой метке, которую выбрал игрок, со значком во второй метке. Если значки одинаковы, программа выполняет три оператора. Первые два оператора сбрасывают ссылочные переменные `firstClicked` и `secondClicked`. После этого они перестают отслеживать метки. Третий оператор — оператор `return`, который пропускает оставшиеся в методе операторы, позволяя не выполнять их.

1. Запустите программу, а затем начните выбирать квадраты в форме.



Если вы выбрали пару, которая не совпадает, произойдет событие таймера Tick и оба значка исчезнут.
Если выбрать совпадающую пару, будет выполнен оператор `if`. Оператор return заставляет метод пропустить код, который запускает таймер. При этом значки остаются видимыми.

Проверка того, выиграл ли игрок

Вы создали увлекательную игру, которая должна завершиться, как только игрок победит. В этом разделе содержатся сведения о том, как добавить метод для проверки того, выиграл ли игрок.

1. Добавьте метод `CheckForWinner()` в конец кода, под обработчиком событий `timer1_Tick()`.

```
/// <summary>
/// Check every icon to see if it is matched, by
/// comparing its foreground color to its background color.
/// If all of the icons are matched, the player wins
/// </summary>
private void CheckForWinner()
{
    // Go through all of the labels in the TableLayoutPanel,
    // checking each one to see if its icon is matched
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            if (iconLabel.ForeColor == iconLabel.BackColor)
                return;
        }
    }
    // If the loop didn't return, it didn't find
    // any unmatched icons
    // That means the user won. Show a message and close the form
    MessageBox.Show("You matched all the icons!", "Congratulations");
    Close();
}
```

В этом методе используется еще один цикл `foreach` для C# или цикл `For Each` для Visual Basic, чтобы пройти по каждой метке в **TableLayoutPanel**. Он проверяет цвет значка каждой метки, чтобы убедиться, что он совпадает с фоном. Если цвета совпадают, значок остается невидимым, а значит игрок не подобрал пару оставшимся значкам.
В этом случае программа использует оператор `return`, чтобы пропустить оставшуюся часть метода. Если цикл прошел через все метки без выполнения оператора `return`, значит, всем значкам в форме была подобрана пара. Программа отображает окно MessageBox с поздравлением победителя, а затем вызывает метод `Close()` для завершения игры.
После этого обработчик событий Click метки вызывает новый метод `CheckForWinner()`.

```
// If the player gets this far, the timer isn't
// running and firstClicked isn't null,
// so this must be the second icon the player clicked
// Set its color to black
secondClicked = clickedLabel;
secondClicked.ForeColor = Color.Black;
// Check to see if the player won
CheckForWinner();
// If the player clicked two matching icons, keep them
// black and reset firstClicked and secondClicked
// so the player can click another icon
if (firstClicked.Text == secondClicked.Text)
{
    firstClicked = null;
    secondClicked = null;
    return;
}
```

Убедитесь, что программа проверяет наличие победителя сразу после отображения второго значка, который выбирает игрок. Найдите строку, где задается цвет второму значку, который вы выбрали, и вызовите метод `CheckForWinner()` сразу после этой строки.

1. Сохраните и выполните программу. Сыграйте в игру и подберите пару всем значкам. В случае победы программа выводит сообщение с поздравлением.

2. Если выбрать **OK**, игра "Подбери пару" закроется.

Изучение других возможностей

Теперь ваша игра "Подбери пару" завершена. Вы можете добавить дополнительные возможности, чтобы сделать эту игру более сложной и интересной. Ниже приведены некоторые варианты.

- Замените значки и цвета на другие.Попробуйте проверить свойство ForeColor метки.
- Добавьте таймер игры, который отслеживает время, необходимое игроку для победы.Вы можете добавить метку, чтобы отобразить время, затраченное на форму. Разместите ее над **TableLayoutPanel**. Добавьте в форму еще один таймер, чтобы иметь возможность отслеживать время. Следующий код служит для запуска таймера, когда игрок начинает игру, и остановки таймера после сопоставления последних двух значков.

- Добавьте звуки, которые будут воспроизводиться при нахождении игроком пары, отображении двух несовпадающих значков и повторном сокрытии значков программой.Для воспроизведения звуков можно использовать пространство имен System.Media. Дополнительные сведения см. в руководстве по воспроизведению звуков в приложении Windows Forms (C#).
- Сделайте игру труднее, увеличив игровое поле. Необходимо сделать больше, чем просто добавить строки и столбцы в TableLayoutPanel. Вы должны также учитывать количество создаваемых значков.
- Сделайте игру интереснее, скрывая первый значок, если игрок медлителен.