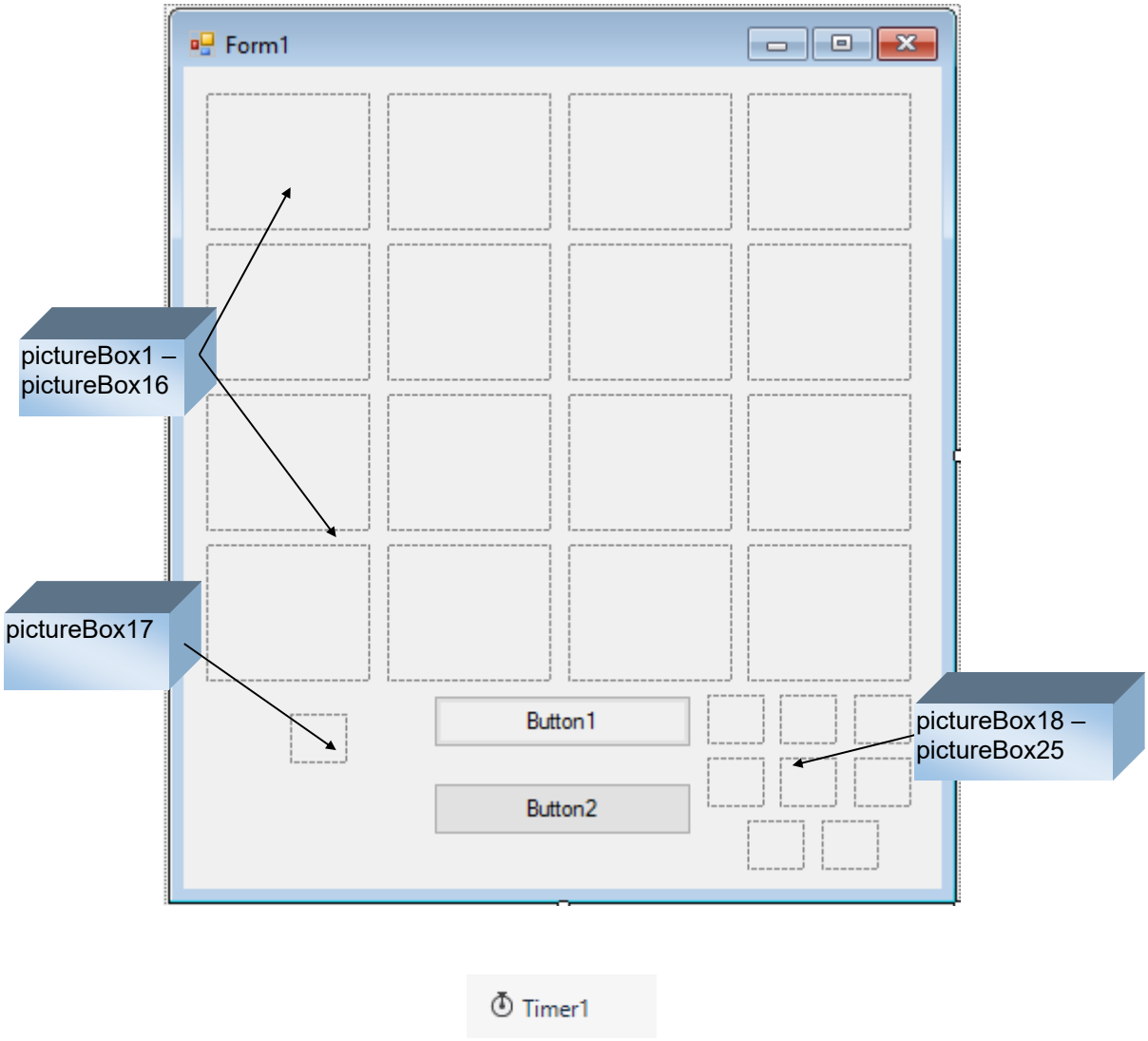
Project 5 – Memory Game

Project Design

In this game for little kids, sixteen squares are used to hide eight different pairs of pictures. The player chooses two squares on the board and the pictures behind them are revealed. If the pictures match, those squares are removed from the board. If there is no match, the pictures are recovered and the player tries again. The play continues until all eight pairs are matched up. The game is saved as **MemoryGame** in the project folder (**\BeginVCS\BVCS Projects**).

Place Controls on Form

Start a new project in Visual C#. Place sixteen large picture box controls and nine smaller picture box controls on the form. Place two buttons and a timer on the form. When done, you form should resemble this:



Set Control Properties

Set the control properties using the properties window:

Form1 Form:

Property Name	Property Value
Text	Memory Game
FormBorderStyle	FixedSingle
StartPosition	CenterScreen

pictureBox1 Picture Box:

Property Name	Property Value
Name	picHidden0
SizeMode	StretchImage

pictureBox2 Picture Box:

Property Name	Property Value
Name	picHidden1
SizeMode	StretchImage

pictureBox3 Picture Box:

Property Name	Property Value
Name	picHidden2
SizeMode	StretchImage

pictureBox4 Picture Box:

Property Name	Property Value
Name	picHidden3
SizeMode	StretchImage

pictureBox5 Picture Box:

Property Name	Property Value
Name	picHidden4
SizeMode	StretchImage

pictureBox6 Picture Box:

Property Name	Property Value
Name	picHidden5
SizeMode	StretchImage

pictureBox7 Picture Box:

Property Name	Property Value
Name	picHidden6
SizeMode	StretchImage

pictureBox8 Picture Box:

Property Name	Property Value
Name	picHidden7
SizeMode	StretchImage

pictureBox9 Picture Box:

Property Name	Property Value
Name	picHidden8
SizeMode	StretchImage

pictureBox10 Picture Box:

Property Name	Property Value
Name	picHidden9
SizeMode	StretchImage

pictureBox11 Picture Box:

Property Name	Property Value
Name	picHidden10
SizeMode	StretchImage

pictureBox12 Picture Box:

Property Name	Property Value
Name	picHidden11
SizeMode	StretchImage

pictureBox13 Picture Box:

Property Name	Property Value
Name	picHidden12
SizeMode	StretchImage

pictureBox14 Picture Box:

Property Name	Property Value
Name	picHidden13
SizeMode	StretchImage

pictureBox15 Picture Box:

Property Name	Property Value
Name	picHidden14
SizeMode	StretchImage

pictureBox16 Picture Box:

Property Name	Property Value
Name	picHidden15
SizeMode	StretchImage

pictureBox17 Picture Box:

Property Name	Property Value
Name	picBack
SizeMode	StretchImage
Image	Back.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox18 Picture Box:

Property Name	Property Value
Name	picChoice0
SizeMode	StretchImage
Image	Ball.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox19 Picture Box:

Property Name	Property Value
Name	picChoice1
SizeMode	StretchImage
Image	Gift.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox20 Picture Box:

Property Name	Property Value
Name	picChoice2
SizeMode	StretchImage
Image	Bear.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox21 Picture Box:

Property Name	Property Value
Name	picChoice3
SizeMode	StretchImage
Image	Block.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox22 Picture Box:

Property Name	Property Value
Name	picChoice4
SizeMode	StretchImage
Image	Ducky.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox23 Picture Box:

Property Name	Property Value
Name	picChoice5
SizeMode	StretchImage
Image	Burger.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox24 Picture Box:

Property Name	Property Value
Name	picChoice6
SizeMode	StretchImage
Image	Hotdog.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

pictureBox25 Picture Box:

Property Name	Property Value
Name	picChoice7
SizeMode	StretchImage
Image	Cake.gif (in **\BeginVCS\BVCS Projects** folder)
Visible	False

button1 Button:

Property Name	Property Value
Name	btnNew
Text	New Game

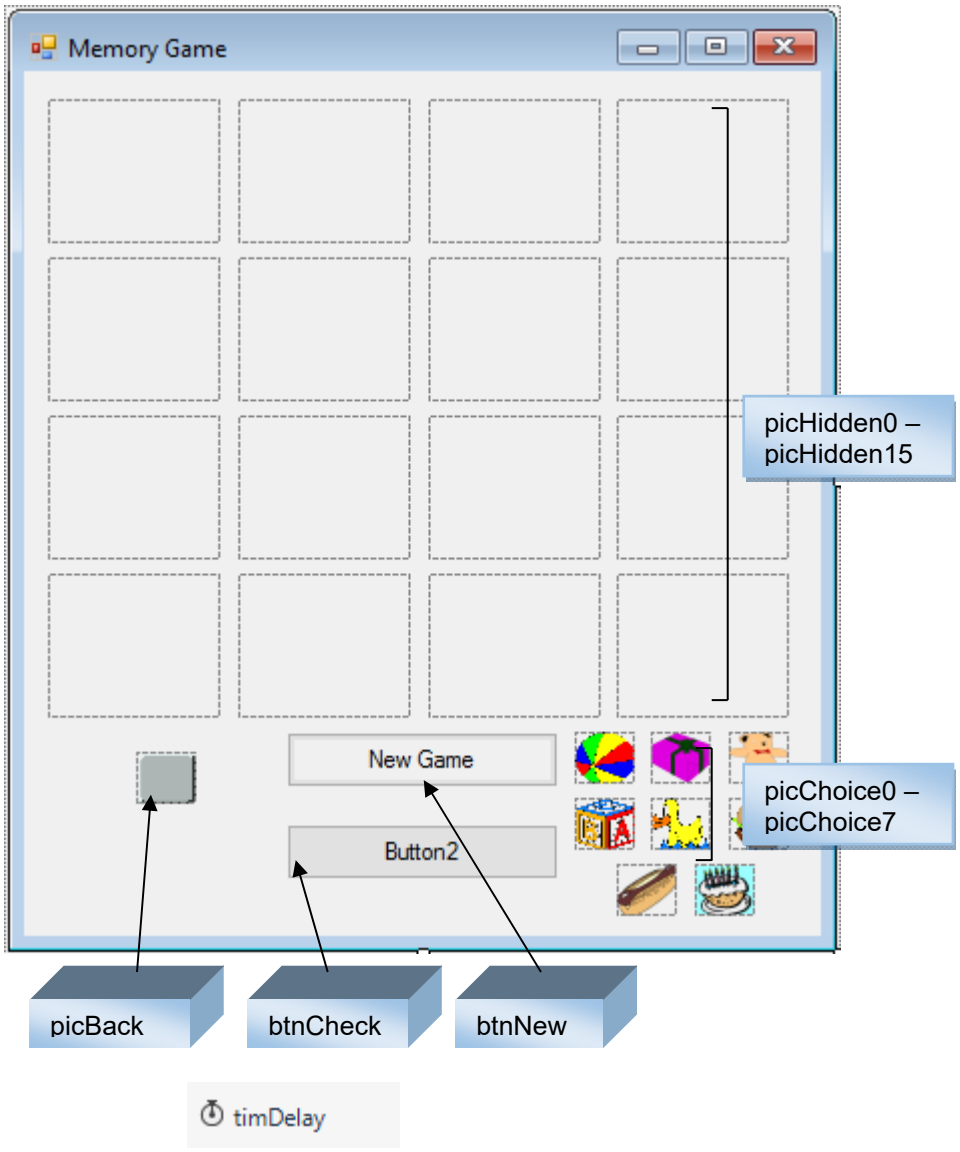
button2 Button:

Property Name	Property Value
Name	btnCheck

timer1 Timer:

Property Name	Property Value
Name	timDelay
Interval	1000

The completed form appears like this:



Before continuing, resize the form to hide **btnCheck**, a button we use for a general method. A few of the graphics will disappear, but that's okay since their Visible property is False anyway.

Write Event methods

When the game starts, pick one box, then another. The game stops when all matching picture pairs have been found. A delay is used to display the pictures for one second before deciding whether or not there is a match. At any time, click **New Game** to start again.

Add this code to the **general declarations** area:

```
int choice;
int[] picked = new int[2];
int[] behind = new int[16];
PictureBox[] displayed = new PictureBox[16];
PictureBox[] choices = new PictureBox[8];
Random myRandom = new Random();
```

The **Form1_Load** event establishes images to pick from

```
private void Form1_Load(object sender, EventArgs e)
{
    //establish display and choices picture boxes
    displayed[0] = picHidden0;
    displayed[1] = picHidden1;
    displayed[2] = picHidden2;
    displayed[3] = picHidden3;
    displayed[4] = picHidden4;
    displayed[5] = picHidden5;
    displayed[6] = picHidden6;
    displayed[7] = picHidden7;
    displayed[8] = picHidden8;
    displayed[9] = picHidden9;
    displayed[10] = picHidden10;
    displayed[11] = picHidden11;
    displayed[12] = picHidden12;
    displayed[13] = picHidden13;
    displayed[14] = picHidden14;
    displayed[15] = picHidden15;
    choices[0] = picChoice0;
    choices[1] = picChoice1;
    choices[2] = picChoice2;
    choices[3] = picChoice3;
    choices[4] = picChoice4;
    choices[5] = picChoice5;
    choices[6] = picChoice6;
    choices[7] = picChoice7;
    // start new game
    btnNew.PerformClick();
}
```

The **btnNew_Click** event method sets up the hidden pictures:

```
private void btnNew_Click(object sender, EventArgs e)
{
    for (int i = 0; i < 16; i++)
    {
        // replace with card back
        displayed[i].Image = picBack.Image;
        displayed[i].Visible = true;
        behind[i] = i;
    }
    // Randomly sort 16 integers (0 to 15) using Shuffle
routine from Class 9
    // behind array contains indexes (0-7) for hidden
pictures
    // Work through remaining values
    // Start at 16 and swap one value
    // at each for loop step
    // After each step, remaining is decreased by 1
    for (int remaining = 16; remaining >= 1; remaining--)
    {
        // Pick item at random
        int itemPicked = myRandom.Next(remaining);
        // Swap picked item with bottom item
        int tempValue = behind[itemPicked];
        behind[itemPicked] = behind[remaining - 1];
        behind[remaining - 1] = tempValue;
    }
    for (int i = 0; i < 16; i++)
    {
        if (behind[i] > 7)
        {
            behind[i] = behind[i] - 8;
        }
    }
    choice = 0;
}
```

The **Click** event methods for the 16 picture boxes:

```
private void picHidden0_Click(object sender, EventArgs e)
{
    picked[choice] = 0;
    btnCheck.PerformClick();
}

private void picHidden1_Click(object sender, EventArgs e)
{
    picked[choice] = 1;
    btnCheck.PerformClick();
}

private void picHidden2_Click(object sender, EventArgs e)
{
    picked[choice] = 2;
    btnCheck.PerformClick();
}

private void picHidden3_Click(object sender, EventArgs e)
{
    picked[choice] = 3;
    btnCheck.PerformClick();
}

private void picHidden4_Click(object sender, EventArgs e)
{
    picked[choice] = 4;
    btnCheck.PerformClick();
}

private void picHidden5_Click(object sender, EventArgs e)
{
    picked[choice] = 5;
    btnCheck.PerformClick();
}

private void picHidden6_Click(object sender, EventArgs e)
{
    picked[choice] = 6;
    btnCheck.PerformClick();
}
```

```
private void picHidden7_Click(object sender, EventArgs e)
{
    picked[choice] = 7;
    btnCheck.PerformClick();
}

private void picHidden8_Click(object sender, EventArgs e)
{
    picked[choice] = 8;
    btnCheck.PerformClick();
}

private void picHidden9_Click(object sender, EventArgs e)
{
    picked[choice] = 9;
    btnCheck.PerformClick();
}

private void picHidden10_Click(object sender, EventArgs e)
{
    picked[choice] = 10;
    btnCheck.PerformClick();
}

private void picHidden11_Click(object sender, EventArgs e)
{
    picked[choice] = 11;
    btnCheck.PerformClick();
}

private void picHidden12_Click(object sender, EventArgs e)
{
    picked[choice] = 12;
    btnCheck.PerformClick();
}

private void picHidden13_Click(object sender, EventArgs e)
{
    picked[choice] = 13;
    btnCheck.PerformClick();
}
```

```
private void picHidden14_Click(object sender, EventArgs e)
{
    picked[choice] = 14;
    btnCheck.PerformClick();
}

private void picHidden15_Click(object sender, EventArgs e)
{
    picked[choice] = 15;
    btnCheck.PerformClick();
}
```

The **btnCheck_Click** “hidden” general method that displays the choices for a match:

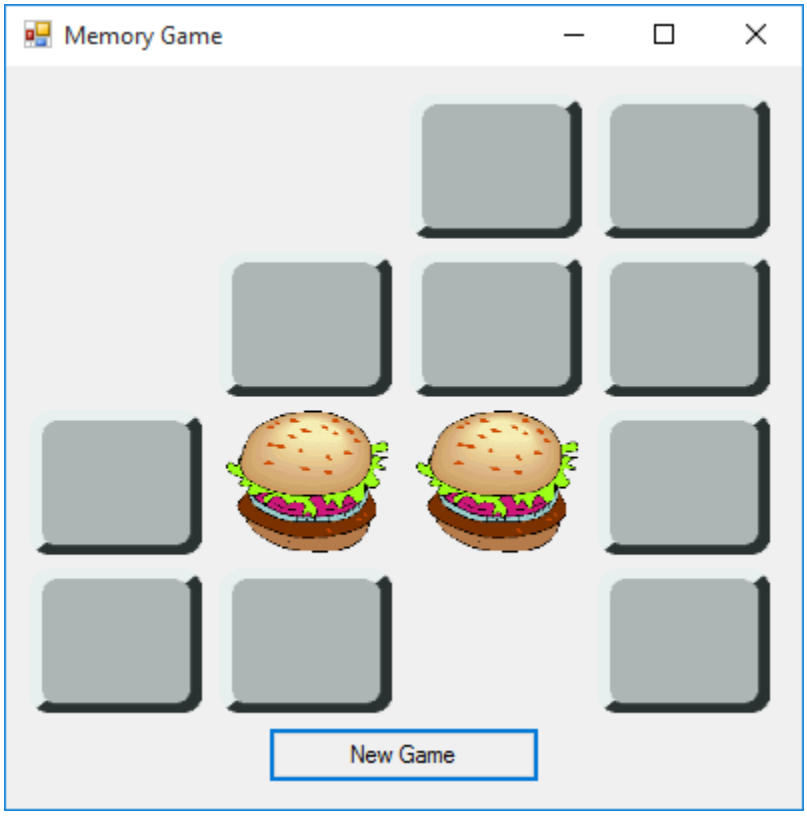
```
private void btnCheck_Click(object sender, EventArgs e)
{
    // Only execute if not trying to pick same box
    if (choice == 0 || (choice == 1 && picked[0] !=
picked[1]))
    {
        // Display selected picture
        displayed[picked[choice]].Image =
choices[behind[picked[choice]]].Image;
        displayed[picked[choice]].Refresh();
        if (choice == 0)
        {
            // first choice - just display
            choice = 1;
        }
        else
        {
            // Delay for one second before checking
            timDelay.Enabled = true;
        }
    }
}
```

The **timDelay_Tick** method that checks for matches after a delay:

```
private void timDelay_Tick(object sender, EventArgs e)
{
    timDelay.Enabled = false;
    // After delay, check for match
    if (behind[picked[0]] == behind[picked[1]])
    {
        // If match, remove pictures
        displayed[picked[0]].Visible = false;
        displayed[picked[1]].Visible = false;
    }
    else
    {
        // If no match, blank picture, restore backs
        displayed[picked[0]].Image = picBack.Image;
        displayed[picked[1]].Image = picBack.Image;
    }
    choice = 0;
}
```

Run the Project

Save your work. Run the project. Sixteen boxes appear. Click on one and view the picture. Click on another. If there is a match, the two pictures are removed (after a delay). If there is no match, the boxes are restored (also after a delay). Once all matches are found, click **New Game** to play again. Here's the middle of a game I was playing (notice the form has been resized at design time to hide the lower of the two button controls:



Other Things to Try

Some things to help improve or change this game: add a scoring system to keep track of how many tries you took to find all the matches, make it a two player game where you compete against another player or the computer, or set it up to match other items (shapes, colors, upper and lower case letters, numbers and objects, etc.).