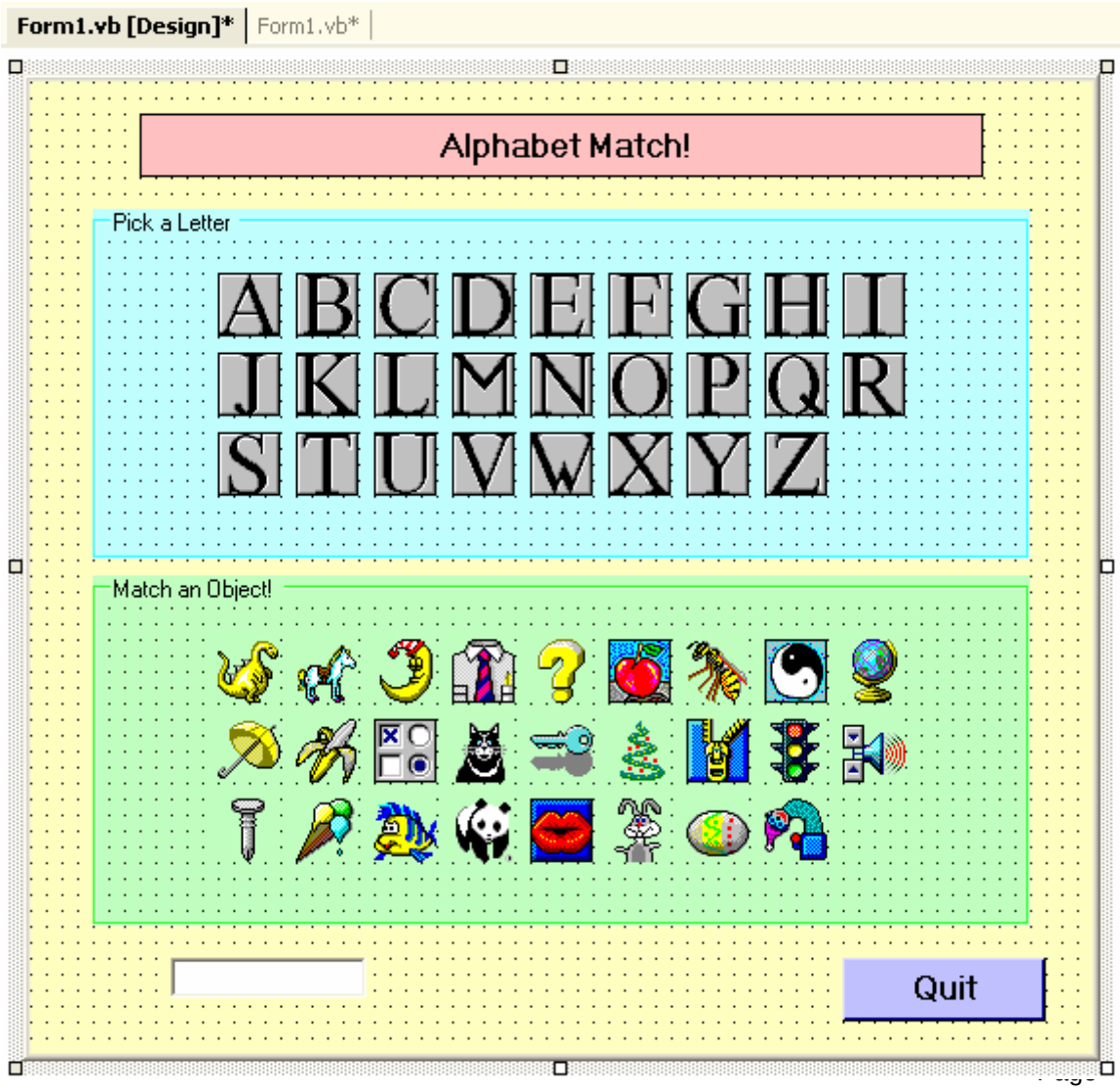
Early Learning Tool 2

Problem Description: This tutorial is a follow on from the Early Learning Tool tutorial in the beginners section. The aim is to design a program that can be used by a toddler (aged 2-4 years). To provide the greatest appeal, it should be easy to use, colourful and provide instant feedback. In addition to this, the program should also be educational. You may choose to base the program around the alphabet, counting, colours, shapes, animals or any early learning theme you wish, as long as what is required is not too complex.

The method used in the previous tutorial was very inefficient. This tutorial will make use of some of the object orientated methods available to us to make the program much smaller and easier to construct.

Skills Covered: Form properties, grouping objects, Tag property, variable declarations, event handlers, looping through controls on a form, Right string function.

This problem can be approached in a variety of different ways. What follows is a suggestion for one program that focuses on the letters of the alphabet. You could create this program and then add other learning elements using a similar coding style if you wish.



The program asks the user to click on a letter and then to try to pick the object which begins with the same letter. If an incorrect choice is made, the program allows the user to pick again. When a correct match is made, a congratulatory message is displayed.

As the first part of this tutorial is essentially the same as the Early Learning Tool tutorial, if you have completed this already, simply load it back up and proceed to enter the code shown later in this document.

Setting up the objects for our program

Follow these steps to create the program.

1. Start a new Visual Basic.NET program called 'EarlyLearn'.

2. Select the Label tool from the toolbox. Use it to draw a rectangular box in the center and at the top of the form. This will be our main heading. Name it 'lblTitle'.

3. Use the Button tool to draw a button in the bottom right hand corner of the form. Name the button 'btnQuit'. This button will be used to end the program.

4. Use the GroupBox tool to draw two large group boxes on the form. These will be used to hold all of the letters of the alphabet and the images of the objects which the user will be trying to match. Make sure the group boxes are large enough so that we can fit in all of the objects that we need to. Name the first group box 'grpLetters' and the second 'grpObjects'.

5. Use the PictureBox tool to create a single rectangular box in the first group box called 'grpLetters'. Even though there are a large number of picture boxes that we are going to use, we are only going to create one to start with. By doing this, we can set it up and copy it as many times as we need to, and save ourselves some time. Set the SizeMode property of the picture box to 'AutoSize'. Click on the Image property of the picture box and then browse to the 'Icons' folder in the same folder as this document. Locate the file named 'A.ico' and select this. The letter A should now be displayed on the picture box. Copy and paste the picture box to make up all the letters of the alphabet, and locate each letter image in the 'Icons' folder. Rename all of the picture boxes appropriately. For example, 'picA', 'picB', etc.

6. Use the PictureBox tool to create another rectangular box, but this time in the second group box called 'grpObjects'. As in the previous step, set the SizeMode property of the picture box to 'AutoSize'. Click on the Image property of the picture box and then browse to the 'Icons' folder in the same folder as this document. Locate the file named 'Apple.ico' and select this. An apple should now be displayed on the picture box. Copy and paste the picture box 26 times. Clicking on the picture boxes in a random fashion, select all of them in turn and set their Image property to be one of the object image files located in the 'Icons' folder. There is one object image per letter of the alphabet. Rename all of the picture boxes appropriately. For example, 'picObjA', 'picObjB', etc.

7. Use the TextBox tool to add a small text box at the bottom of the form. Name it 'txtSelectedLetter'. This text box will be used to hold the letter that has been clicked on by the user.

Changing the properties of the objects

Click on each object in turn and set the following properties:

Form1

StartPosition: CenterScreen
Text: *Blank*
MaximizeBox: False
MinimizeBox: False
ControlBox: False

By setting the forms properties in this way, we can create a borderless form that appears on the screen that cannot be moved, closed or resized by the user. Given that this program will be used by small children, doing this ensures that the program is not ended prematurely. This method can also be used to create forms of a different shape. A different shaped form can be designed in a paint program and then placed onto the form using the BackGroundImage property. If the TransparentColor property of the form is then set to the surrounding colour of the background image, the form will appear to be of a different shape.

lblTitle

Text: Alphabet Match!
TextAlign: MiddleCenter
BorderStyle: FixedSingle
Set the Font and BackColor properties yourself, so that the heading is colourful and easy to read.

btnQuit

Text: Quit
TextAlign: MiddleCenter
Set the Font and BackColor properties yourself, so that the button is colourful and easy to read.

grpLetters

Text: Pick a Letter
BackColor: A colour of your choosing.

grpObjects

Text: Match an Object!
BackColor: A colour of your choosing, different from the colour chosen for grpLetters.

Placing objects inside a group box can be very beneficial. During design time, all of the objects inside the group box can be moved around simply by moving the group box on the form. During the execution of a program, if the group box is made invisible, all the objects within the group box are also invisible. Likewise, if the group box is disabled, all the objects inside the group box will act as if they are also disabled. This can save a great deal of time. As you will also see when we begin coding our program, it is very easy to access all of the objects within a group box and change their properties via programming code.

picA – picZ

Image: The revelant image file from the 'Icons' folder.
SizeMode: AutoSize
Tag: The respective letter of the alphabet. For example 'A', 'B', etc.

The Tag property provides a method by which information can be stored in the object for future reference. In this case, by storing the corresponding letter in the tag property of the picture box, we can easily refer to the property in code to work out which letter has been clicked on by the user.

picObjA – picObjZ

Image: The revelant image file from the 'Icons' folder.
SizeMode: AutoSize

txtSelectedLetter

Text: *Blank*.
Visible: False

Adding code to our program

If you have previously created the Early Learning Tool from the beginners section of this resource, then you should replace all of the code you typed in with the code shown below:

1. Most of our code will be concerned with the Click events of the different picture boxes on our form. In the beginner solution to this problem, each of the picture boxes were coded separately to achieve the desired result, but this is inefficient and time consuming. An easier method is to create an event handler.

   Switch to code view, and type the following line which will set up our event handler:

```
Private Sub LetterSelect(ByVal sender As Object, _
      ByVal e As System.EventArgs) _
      Handles picA.Click, picB.Click, picC.Click, picD.Click, _
      picE.Click, picF.Click, picG.Click, picH.Click, picI.Click, _
      picJ.Click, picK.Click, picL.Click, picM.Click, picN.Click, _
      picO.Click, picP.Click, picQ.Click, picR.Click, picS.Click, _
      picT.Click, picU.Click, picV.Click, picW.Click, picX.Click, _
      picY.Click, picZ.Click

End Sub
```

The event handler that we have set up is called ‘LetterSelect’ and it will be triggered when any of the events listed after the ‘handles’ keyword is executed. In other words, whenever the user clicks on any of the picture boxes with letters on them, this subroutine will be run.

The ‘sender’ parameter can be used to refer to the actual object that triggered the event handler, and the ‘e’ parameter can be used to access any special parameters related to the event that was triggered. Using these parameters can be complicated, and not all of their features will be discussed in this tutorial.

2. Type the following code into the ‘LetterSelect’ subroutine:

```
Dim Letter As Control
If txtSelectedLetter.Text = "" Then
      For Each Letter In grpLetters.Controls
            Letter.Visible = False
      Next
      sender.SizeMode = PictureBoxSizeMode.StretchImage
      sender.Height = 64
      sender.Width = 64
      sender.Left = sender.Left - 16
      sender.Top = sender.Top - 16
      sender.Visible = True
      txtSelectedLetter.Text = sender.Tag
End If
```

All of the controls within a group box, panel or a form, can be looped through using the first part of this code. Firstly, the variable ‘Letter’ is declared as a control. The loop ‘For Each’ is used to cycle through all of the controls that are present in the set defined by ‘grpLetters.Controls’. In other words, the loop will cycle through all of the controls in the group box called ‘grpLetters’, and the ‘Letter’ variable will point to each one in turn. The next line of code makes each picture box in turn invisible.

The ‘If’ statement checks to see if a letter has already been selected. If it has, the letter that was clicked on would have had its value copied to the ‘txtSelectedLetter’ text box, and the code will not be executed.

As the event handler is triggered when the user clicks on any of the picture boxes, you may be wondering how we can refer to the one that was clicked on in our code. The way to do this, is using the ‘sender’ parameter, which is holding a reference to the picture box that was clicked. In the code above, the picture box that has been clicked on is made larger than the others and visible. The Tag property is copied to the ‘txtSelectedLetter’ text box.

3. We now need to write another event handler, this time for when the user clicks on a picture box with an object on it. Type in the following event handler definition, noting that this time the events relate to the object picture boxes instead of the letter picture boxes.

```
Private Sub ObjectSelect(ByVal sender As Object, _
      ByVal e As System.EventArgs) _
      Handles picObjA.Click, picObjB.Click, picObjC.Click, _
      picObjD.Click, picObjE.Click, picObjF.Click, picObjG.Click, _
      picObjH.Click, picObjI.Click, picObjJ.Click, picObjK.Click, _
      picObjL.Click, picObjM.Click, picObjN.Click, picObjO.Click, _
      picObjP.Click, picObjQ.Click, picObjR.Click, picObjS.Click, _
      picObjT.Click, picObjU.Click, picObjV.Click, picObjW.Click, _
      picObjX.Click, picObjY.Click, picObjZ.Click
End Sub
```

4. Type the following code into the 'ObjectSelect' event handler:

```
      Dim ObjTile As Control
      Dim Letter As Control
      If txtSelectedLetter.Text <> "" Then
            For Each ObjTile In grpObjects.Controls
                  ObjTile.Visible = False
            Next
            sender.SizeMode = PictureBoxSizeMode.StretchImage
            sender.Height = 64
            sender.Width = 64
            sender.Left = sender.Left - 16
            sender.Top = sender.Top - 16
            sender.Visible = True
            If txtSelectedLetter.Text = _
                  Microsoft.VisualBasic.Right(sender.Name, 1) Then
                  MsgBox("Correct!!", MsgBoxStyle.OKOnly + _
                        MsgBoxStyle.Exclamation)
            Else
                  MsgBox("Bad luck! Try again!", MsgBoxStyle.OKOnly _
                        + MsgBoxStyle.Exclamation)
            End If
            sender.SizeMode = PictureBoxSizeMode.AutoSize
            sender.Left = sender.Left + 16
            sender.Top = sender.Top + 16
            For Each ObjTile In grpObjects.Controls
                  ObjTile.Visible = True
            Next
            For Each Letter In grpLetters.Controls
                  If Letter.Visible = True Then
                        Letter.Height = 32
                        Letter.Width = 32
                        Letter.Left = Letter.Left + 16
                        Letter.Top = Letter.Top + 16
                  Else
                        Letter.Visible = True
                  End If
                  txtSelectedLetter.Text = ""
            Next
      End If
```

The first part of this code checks to see whether the user has already clicked on a letter. If they have not, none of the code is executed.

The object that was clicked on is made larger and the other objects turned invisible, in the same fashion as the letters were in the previous event handler.

After this is done, the code checks to see whether the selection that has been made is correct or not. The letter that was clicked by the user is contained in the 'txtSelectedLetter' text box. This is compared to the object that the user has clicked, by examining the name of the picture box. As each of the picture boxes are named according to the first letter of the object they represent, this can be used to our advantage. The 'Microsoft.VisualBasic.Right' command can be used to extract any number of characters from the right hand side of a string. In this case, one character is extracted from the name of the picture box that was clicked. If it is the same as the letter in the text box, a congratulations message is displayed, otherwise a message encouraging the user to try again is shown.

Regardless of whether the user achieved a match or not, the picture boxes are reset to their starting positions, and all of them are made visible again. The text box is also cleared.

5. Type the following code into the click event of the button 'btnQuit', which will end the program:

```
Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e _
   As System.EventArgs) Handles btnQuit.Click
   End
End Sub
```

Testing the program

Run and test the program and ensure that it behaves as expected. Test that it is possible to match two picture boxes and receive the congratulatory message. Test that the alternative message is displayed when the two picture boxes do not match. Also check to see that all of the picture boxes are reset once the selections have been made.

Further ideas to develop

- Incorporate sound effects and animations to make the game more interesting.
- Create a program which uses the same principles based around the numbers 1-20, basic shapes, farm animals or a related theme.
- As each pair is selected, make them invisible so that the user can see what pairs are still remaining. When all the pairs have been selected, display a congratulatory message and give the user the option of resetting the program so that they can start again.