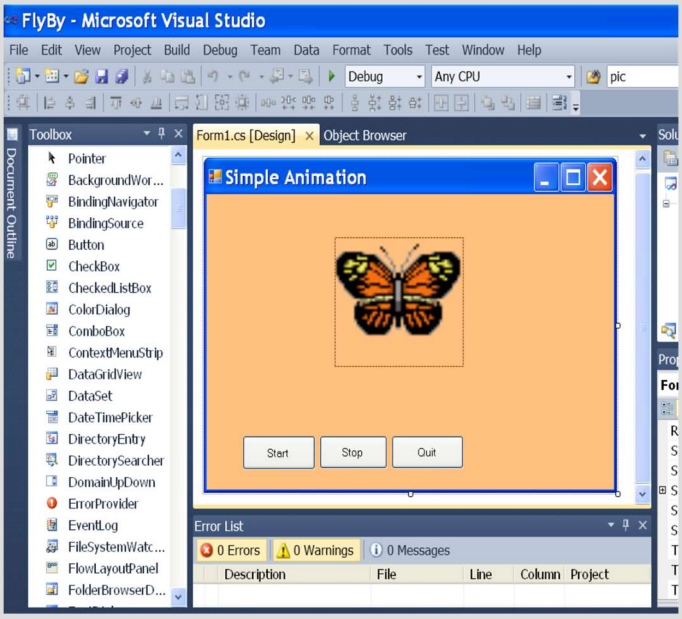
Windows Programming

bucks new university

.NET Windows Programming Using C#



Application Programming (CO453)

Part B Weeks 5-8

C# Windows Programming
Unit 3

MessageBoxes, Timers and other .NET controls

Some Other .NET Objects (or controls)

There are many other types of objects available to the Visual C# programmer as well as forms, buttons, text boxes, labels, etc. We shall now take a look at some of them.

3.1 Radio Buttons Revisited

Radio buttons are useful for allowing users to make choices on a form.
In this Currency Converter example we use 3 radio buttons
(note that they are mutually exclusive .. clicking one button deselects the others)

- Open the existing project called **Currency Converter**. You will see:
  - a text box named **txtUKMoney**
  - a label named **lblConverted** to hold the converted UK money
  - a **Panel** that acts as a Container for some radio buttons
  - 3 radio buttons, named **rbnDollars**, **rbnEuros** and **rbnRupees**.
  - Some other labels and a **Quit** button
- Run the project and try selecting different radio buttons.

Adding Code to the Radio Buttons

We want to get the program to take UK money from the **txtUKMoney** textbox, multiply by the rate and display the result in **lblConverted**. This needs to be done each time we check one of the radio buttons.

- Double-click the Dollars radiobutton and add some code to the **rbnDollars_CheckedChanged()** method:

```
double amount;                                  // variable for calculation
amount = Convert.ToDouble(txtUKMoney.Text) * 1.8;   // 1.8 dollars to the pound
lblConverted.Text = amount.ToString() + " Dollars";  // show the result
```

Task 3.1: Converter

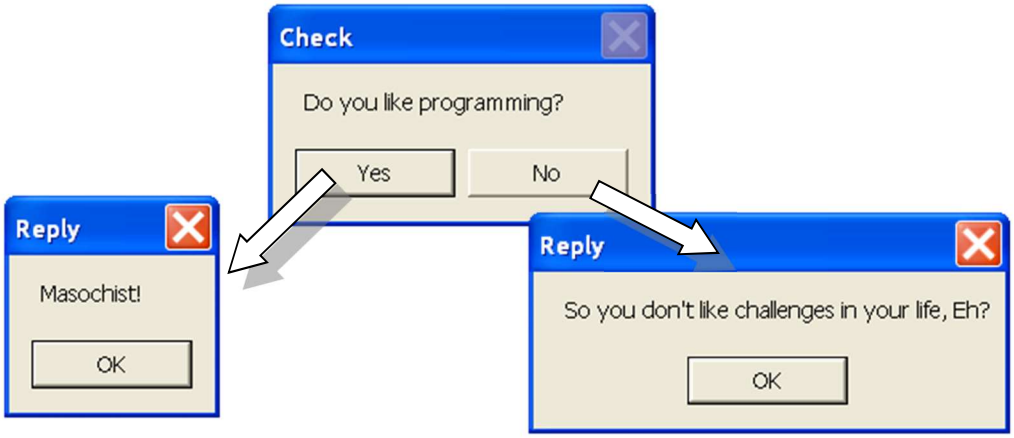
1. Complete the above task to convert UK pounds to dollars
2. Add similar code to the other radio buttons .. note that there are about 1.4 euros to the pound and about 80 rupees to the pound (revise if you wish)
3. Complete the **Converter** project so that the interface looks good and all radio buttons etc. work as they should and the form **StartPosition** is **CenterScreen**
4. Add a suitable background colour to your form
5. Improve your **Quit** button so that it shows a MessageBox which asks:
   Are You Sure?
   and gives you the chance to reply **Yes** or **No** with appropriate actions

Responding to MessageBoxes

There are different kinds of MessageBoxes. The simplest just have an **OK** button (see above) but others can have **Yes** and **No** buttons, **Yes**, **No** and **Cancel** buttons, etc. When the MessageBox has more than one button, it is possible to detect which one has been clicked. The following code below demonstrates an example of responding to choosing Yes from a MessageBox:

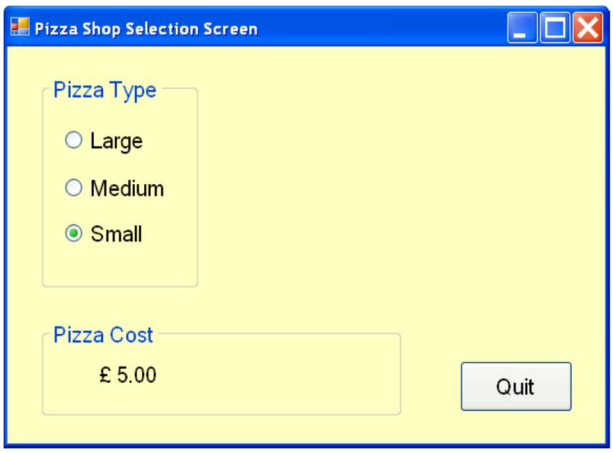
```
DialogResult response;              // define a variable for the result

response = MessageBox.Show("Do you like programming?",  "Check",
                                                    MessageBoxButtons.YesNo);
if (response == DialogResult.Yes)
{
      MessageBox.Show("Masochist!", "Reply");
}
```



3.2 Check Boxes

Check boxes are very similar to radiobuttons, **but** they allow multiple selections to be made .. and each box displays a tick when selected.



- Open the existing **PizzaShop** application.
- Run it to see how the radio buttons work.
- Also look at the code to see how the totalcost is worked out and displayed

Adding Check Boxes to the Form

- First of all add another **GroupBox** to the Form and change its Text property to **Toppings** as shown here
- Then put 4 **CheckBoxes** into the GroupBox and name them **cbx1**, **cbx2**, **cbx3** and **cbx4**
- Choose any 4 pizza toppings and change checkbox **Text** properties to display these, as shown.

Adding Code to the Check Boxes

- Each Topping costs an extra £2.50
- Double-click the first check box and add some code to the **cbx1_CheckedChanged()** method

```
if (cbx1.Checked)          // has this box been checked?
{
      // code to add 2.50 to the topping cost
       // code to display the new totalcost in the label lblCost
}
```

Task 3.2: Check Boxes

1. Complete the Pizza Shop application as described above.
2. Add code to all 4 of the CheckBoxes
3. Test the program to see that everything works correctly and that each topping adds an extra £2.50 to the cost
4. If you deselect a topping can you get the cost to reduce by £2.50?

5. **Complete a Test Plan for the PizzaShop program**

3.3 List Boxes

The list box allows users to select from a list of choices, many more than are possible with radio buttons. A list box is basically an array of strings. Scroll bars are provided for long lists.

- Open the existing Windows application called **PoshNosh**
- This has a PicureBox and a Quit button
- It also has 2 GroupBox controls: **Starter** and **Total Cost**
- There is a **ListBox** called **lstStarter** with 7 items on the list

- **Run** the program and select different items from the list .. there is a label called **lblTotal** that displays the total cost

Adding another ListBox

- First add another **GroupBox** and change its **Text** property to **Main Course**
- Add a **ListBox** control to the GroupBox, name it **lstMain** and change its **Sorted** property to **True** so the list will be in alphabetical order.
- Select the **Items** property and add 4 main course items (one per line): e.g. Steak and Chips, Fish and Chips, Vegetable Curry, Chicken Lasagne
- Now double-click the ListBox and add some code to the **lstMain_SelectedIndexChanged()** method … for example :

```
switch (lstMain.Text)              // what is the chosen item from lstMain  ?
{
        case "Steak and Chips"   :  maincoursecost = 12.50;   break;
        case "Fish and Chips"    :  maincoursecost = 7.50;    break;
        case "Vegetable Curry"   :  maincoursecost = 8.00;    break;
        case "Chicken Lasagne"   :  maincoursecost = 7.25;    break;
}

totalcost = startercost + maincoursecost + dessertcost;        // calculate total
lblTotal.Text = "£" + totalcost.ToString("0.00");              // display total
```

Task 3.3: List Boxes

1. Complete the ListBox exercise above so that Starter and Main Course selections from the ListBoxes all work correctly .. with correct totals
2. Examine the code and make sure you understand how it all works
3. Try altering the height of a ListBox and see how this affects the display.

3.4 Combo Boxes

The drop-down Combo Box is like a cross between a ListBox and a TextBox because you can add new items and also display Text

- First add a new **GroupBox** to the previous form and change its Text property to **Dessert**
- Add a ComboBox to the GroupBox and give it a name: **cbxDessert**
- Change its Text property to **Dessert Choices**
- Now add a selection of your favourite desserts to the **Items** property.
- Run the program and see your selections revealed when you click the drop-down arrow:

- Now in design mode, double-click the combobox and add code to the **cbxDessert_SelectedIndexChanged()** method so that a dessert cost is added to the total and displayed correctly (use the previous listbox code to help you)

Task 3.4: Combo Boxes

1. Complete the **Posh Nosh** project so that all totals work correctly using two ListBoxes and a ComboBox to select the 3 courses
2. Design and use a Test Plan for the Posh Nosh project

3.5 Monkey Bash: Detecting Mouse Clicks

- Open the existing **Monkey** project and look at the form .. it has a PictureBox with an image of a cute little monkey.
- There are 3 buttons at the bottom of the form that don't do anything yet.
- Also notice a **timer** control has been added (it shows as **timer1** under the form)
- The program doesn't do much at the moment but we shall improve it to develop a 'Bash the Cute Little Monkey' game!

The timer control

- The **timer** is a control that can be set to **Tick** at a certain rate. It has a **Tick()** method that can be used to run some code each time it ticks.
- Double-click on the timer control to see the code that has already been added to its **Tick()** method. Each time the timer ticks its sets up new random values for the position of the pictureBox.

```
private  void  timer1_Tick(object sender, EventArgs e)
{
        x = rand.Next(this.Width - 100);        // pick a random x
        y = rand.Next(this.Height - 100);       // pick a random y
        pbxPicture.Left = x;                    // set picture to these new values
        pbxPicture.Top = y;
        Refresh();                              // redraw in the new position
}
```

- note: **this.Width** and **this.Height** are the width and height of the form. These are being used as a guide for the random x and y values)
- Also note that **x**, **y** and **rand** have been defined earlier in the program :

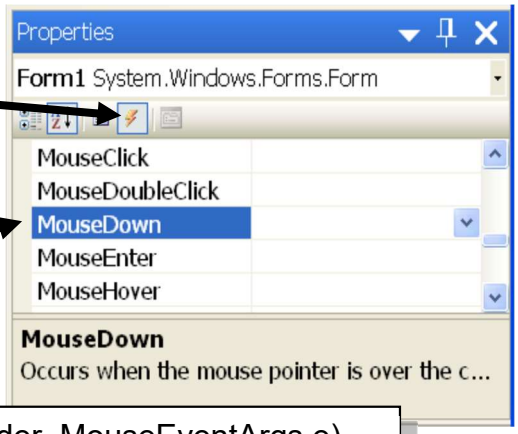
```
int x, y;
Random rand = new Random();     // create new Random object called rand
```

Task 3.5: Monkey Bash 1

1. Double-click the Start button and enter code into its click() method:
   timer1.enabled = true; *// enable the timer control*
2. Double-click the Stop button and enter code into its click() method:
   timer1.enabled = false; *// disable the timer control*
3. Now run the program and use these 2 buttons to start and stop the timer
4. Get the **Quit** button to work too
5. Now add code to display a MessageBox saying **"Ouch! You HIT Me!"** when the Monkey picture is clicked with the mouse.
6. Also get the picture to change to a frowning monkey (monkey2.gif) when it is clicked and then back again
7. Change the **Cursor** property of the form to a **hand**
8. Now play the game.
   - Also try changing the **Interval** property of **timer1** to 2000 and then to 500 ..notice the difference when playing the game and adjust your settings to find a speed that is most suitable

3.6 More Mouse Events

- There are lots of mouse events as well as the MouseClick event. For example, the **MouseDown** event can be used to find out where the mouse is on the form …
- First start a new Windows application and call it **MouseEvents**
- The default event for a form is the Load event so how do we find the MouseDown event?
  - Click on the form to select it
  - Then click the **Events** button in the Properties window.
  - This will show you a list of all the form Events .. scroll down to find the MouseDown event.
- Now double-click **MouseDown** to enter code into the code window for the MouseDown method :




```
private  void  Form1_MouseDown(object sender, MouseEventArgs e)
{
        MessageBox.Show ("The mouse X position is " + e.X);
}
```

- Run the program and press the mouse anywhere on the form to see the MessageBox

Task 3.6: Monkey Bash 2

1. Modify the previous Monkey Bash program so that you get a second message:
   You MISSED! if you click the Form instead of the monkey
   (you will need to use the form's click method for this)
2. A smiley monkey should also be displayed when this happens
3. Add a small **GroupBox** to the form with a **label** inside it.
4. Improve the game by getting the label to display the number of monkey hits
5. Display the number of misses in the GroupBox too
6. Modify the game so the picture never moves into the area of buttons or labels

3.7 MyPad (a simple version of Notepad)

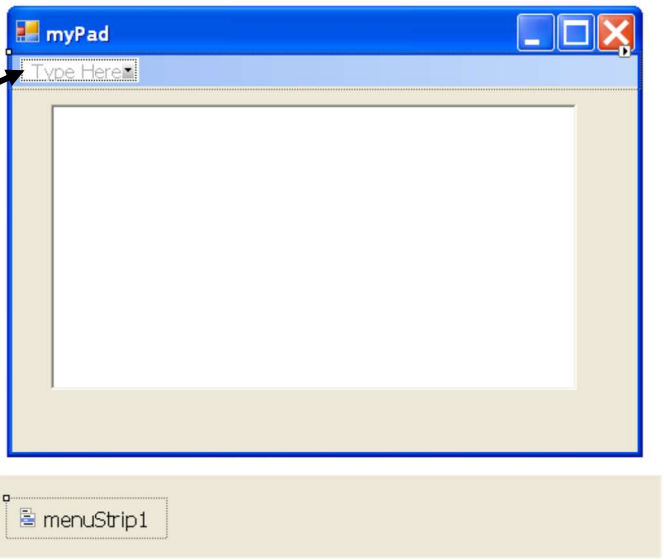
We shall now build a simple notepad clone which will use menus, RichTextboxes, common dialog boxes and also our Scroll bars. Note: You may need to do this project away from the network on a memory stick or another drive to get it to work properly.

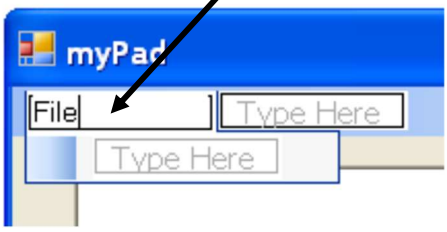
RichTextbox

- Start a new Windows project called **myPad**.
- Add a **RichTextBox** to the project from the ToolBox and call it **txtMain**
  - alter its size to fit most of the form
  - this can be used to enter lots of text which can be formatted, cut and paste etc.
  - Now add a MenuStrip to the project (see next page)

Menus

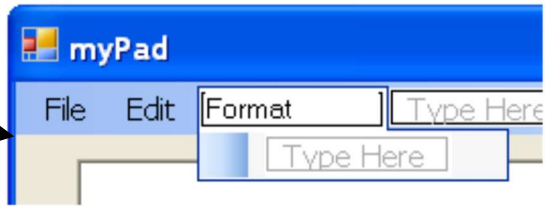
- Add a **MenuStrip** from the ToolBox
- This will appear at the top of the form (and also underneath)
- Where it says **Type Here** type in a menu item **File**



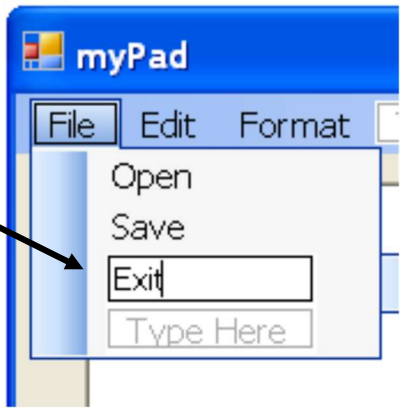


Adding Menu Items

- To the right of **File**, add 2 more items, **Edit** and **Format** (as shown here)



- Then select **File** again and add some drop-down items below File :
  Open, **Save** and **Exit**
- Then select **Edit** and add some more drop-down items below this:
  Cut, **Copy** and **Paste** and **Undo**
- Finally select **Format** and add two drop-down items **Font** and **Background Colour**



here:

Adding Shortcuts

- All the menu items can now be given shortcuts. Here is one example:
- Select **Cut** from the **Edit** menu
- Go to its **Properties** and find **ShortcutKeys**

- Select the down arrow and choose **Ctrl** and **X**
- Use a similar method to set shortcut keys for all your menu items.

Task 3.7: MyPad 1

1. Set up the myPad project with **RichTextBox** and **MenuStrip** as above
2. Run the project and check that the menu items can be selected (with no functionality as yet!) and you can type into the Text box
3. Test the shortcut keys and see that they work (no functionality yet)

3.8 Adding Functionality to the Menu

We want our menu to actually do something useful. We need to add code to the items.
Note: It may be necessary to put the MyPad folder onto a memory stick or another drive.

Programming the Edit Menu

- Select the **Cut** item from the edit menu and double-click this
- Inside the **cutToolStripMenuItem_Click()** method add some code to cut text from your RichTextBox (named txtMain). This is easy .. it is just :

```
txtMain.Cut();
```

- Now in a similar way program the **Copy** and **Paste** and **Undo** menu items
- Run the program and test you Edit Menu to see that it all works

- You should also be able to program the **Exit** item in the **File** menu

Task 3.8: MyPad 2

- Program the **Edit** menu to work correctly

3.9 The File Menu and Common Dialogs

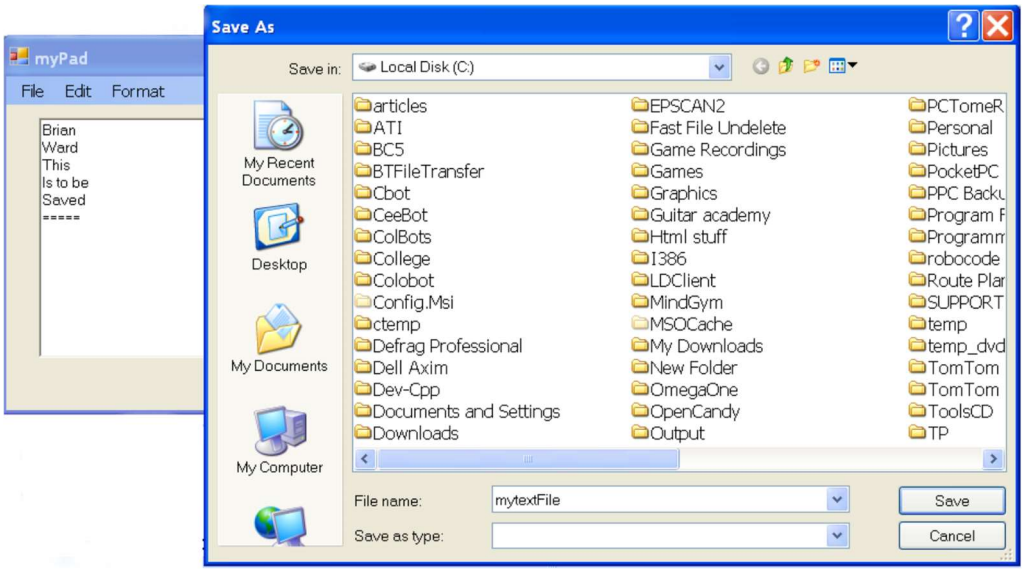
The other menu items are slightly more difficult to program. We shall be using some quite powerful controls called **Common Dialog Boxes** in this section.

Programming the Save Menu item

- Find the **SaveFileDialog** control in the Toolbox and add this to the Project. It will appear underneath the form.
- Change the name to **sfd**
- Now select **Save** from the **File** menu and double-click this
- Inside the **saveToolStripMenuItem_Click()** method add code to use the **sfd** dialog that you just added to the project:

```
sfd.ShowDialog();                    // show dialog to choose a file name
txtMain.SaveFile( sfd.FileName );    // save text using this file name
```

- Now run the project again, type a few sentences into the txtMain and then choose the **Save** option .. you should get a dialog that you will recognise:



- Choose a suitable name for your file and hit the **Save** button.

The Open Menu Item

- Select the **OpenFileDialog** control in the Toolbox and add this to the Project. Change its Name to **ofd** for short
- Program the **Open** menu item (hint: try using **txtMain.LoadFile(ofd.FileName);** and test it by opening the file you just saved.

Task 3.9: MyPad 3

- Program the **File** menu to work correctly. If necessary you may have to copy MyPad onto a memory stick or other drive and run it from there.

3.10 The Format Menu and Common Dialogs

We shall be use some more **Common Dialog Boxes** in this section.

The Font Menu Item

- Add the **FontDialog** control to the Project. Call it **fd** for short.
- Add code to the **Font** item to show the FontDialog and then select the appropriate font to use in txtMain:

```
fd.ShowDialog();
txtMain.Font = fd.Font;
```

- Test it to see if it works OK

The Background Colour Menu Item

- This time add the **ColorDialog** control to the Project. Call it **cd** for short.
- Can you see how to program the **Background Colour** item to first show the dialog and then use the **Color** selected as the **BackColor** for **txtMain** (see the code above for clues)

Foreground Colour Menu Item

- Add a third item to the **Format** menu called **Foreground Colour**
- Program this so it changes the **ForeColor** used by **txtMain** and test it

Task 3.10: MyPad 4

- Program the **Format** menu to work correctly

Unit 3: Independent Study

Task 3.11: Hit the Target !

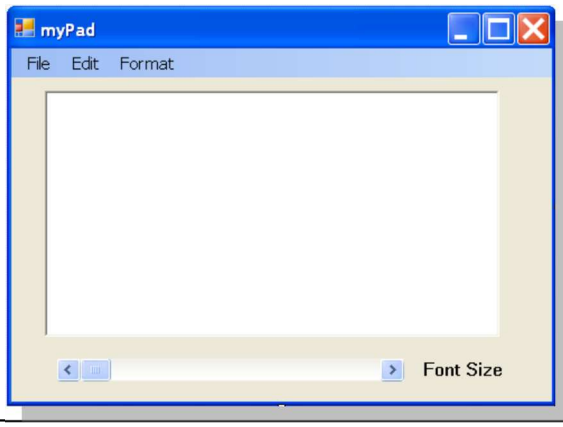
1. Copy the whole Monkey folder and modify it so that a moving Target is displayed. You will find **Target.gif** in the normal Images folder on the L: drive
2. Make the Size of the PictureBox equal to 100, 100 (so its centre will be 50,50)
3. Use labels to display a score and message that depend on your distance from the target centre (see note below)
4. Note: **e.X** and **e.Y** can be used here in the formula for a right-angled triangle
   - distance = square root of ($(\mathbf{e.X} - 50)^2 + (\mathbf{e.Y} - 50)^2$) .. if the target centre is 50, 50
   - This gives us the mouse distance from the centre
   - Math.Pow() and Math.Sqrt() can be used in the formula

Task 3.12: More Monkeys

2. Add 4 more timers and Monkeys to your Monkey project
3. Set different intervals for the timers and create an interesting variation of the game

Task 3.13: MyPad Scrollbar

- Continue the myPad project, and add a scrollbar which changes the font size of selected text, as below



Adding a Scroll Bar

Add a horizontal ScrollBar. Also add a **Font Size** label as shown here.

- Change the **Minimum** property of Scrollbar to **8** and **Maximum** to **36**
- Now add code to change the Font Size of the txtMain font
  e.g. this example changes the size to 20:

```
Font  f = new  Font(txtMain.Font.Name, 20, FontStyle.Regular);  // change size to 20
txtMain.SelectionFont = f;                                      // use this font in text box
```

- Modify this code so it uses the value generated by the Scroll bar as the size

Task 3.14: Choose a Picture

1. Create a **new** Windows application and add to the form:
   a. a ComboBox (name: cbxChoice)
   b. a PictureBox (name: pbxChoice)
   c. a Quit button (name: btnQuit)
2. Look at the **monsters** folder in the **images** folder and choose at least 6 monster images .. copy them to your project **Debug** folder (in **bin**)
3. Look at the combobox **Items** property and add some item names .. the names of your selected monster pictures. Also change the **Text** property.
4. Now add code so that when a particular Text is chosen a corresponding image is displayed in the PictureBox

Note:

pbxChoice.Image = Image.FromFile("kong.gif");
puts an image into the **pbxChoice** PictureBox