Creating a slideshow viewer in C#

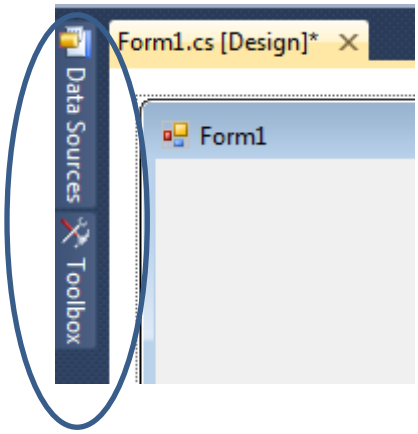
This will create a slideshow viewer that lets you click manually between images or use a timer to automatically progress. It will demonstrate;

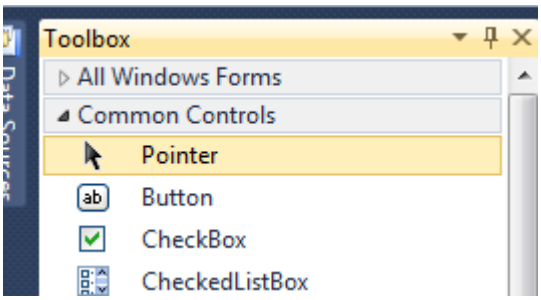
- Loops
- Selection (if statements)
- Events (Button presses)
- Loading images
- A bit about file handling and using libraries and classes

Create a new project in C# that is a **windows form application** and give it a suitable name. (I called mine SlideshowTut). We've mostly used simple GUI based programs before but this requires a GUI.

A form in C# should look familiar, the basic panel that many programs are comprised of, you then add tools, controls and other objects to them and attach code that does something. For instance, when you add a button you add an **"Event listener"** this then listens in the background for you to press the button it's linked to and does some code when you do.

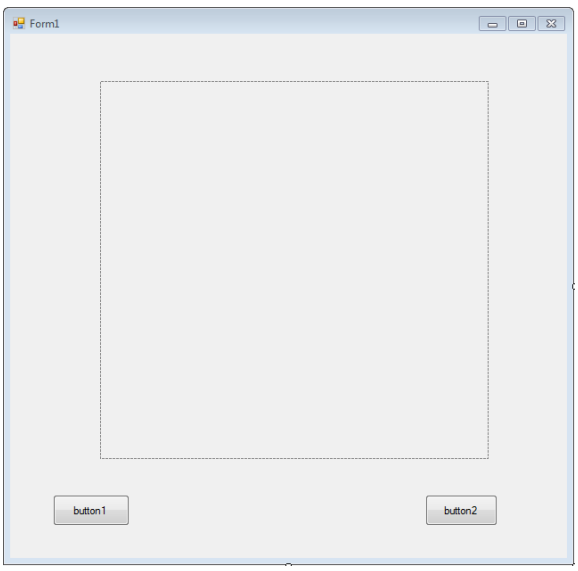
Resize your form to make it a bit bigger (just as you'd resize a picture in say word) and ensure the toolbox is visible. If it's not, look to the left and click on toolbox, then click on the little pin, which will force it visible.



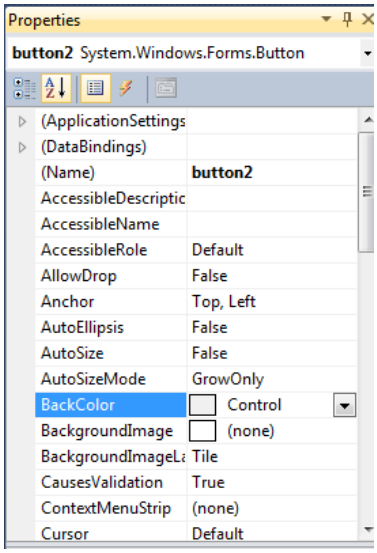


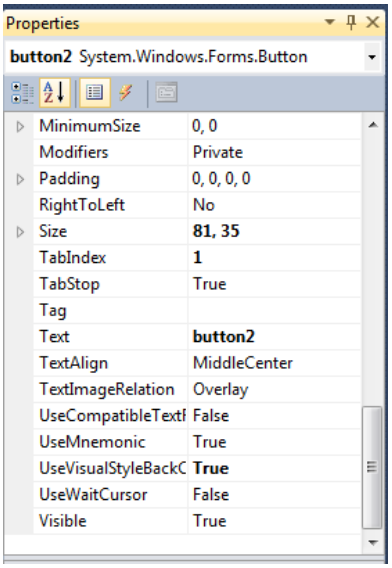
Now, from the toolbox you're going to add a PictureBox (that's used to store pictures funnily enough) and a couple of buttons.

Click button from the toolbox and draw two buttons anywhere on screen. Then do the same with a picturebox. It should look a bit like this.



Click on a button, Right click, go to properties, these are all the properties behind the button object, here you can change all sorts, colours, fonts, styles, but importantly you can change its object name and the text inside. Here, (name) is the name by which you can access the button, it's name in code so to speak. This will be more important later.





Change the name of one button to leftButton, and it's Text (shown in second screenshot) to left or <, then change the name of the other to rightButton and it's Text field to right or <. Be careful about your use of capitals, it matters!

Adding the code to make it work

Now, you have a pretty program that does… nothing;

We're going to add the code to make it work. Double click on the right button, see the code "private void rightButton_Click(…)" this is the code that is run when the **EVENT** for the button that will progress to the right hand picture is found, e.g. when the button is pressed. Add the below code;

```
namespace SideshowTut
{
    public partial class Form1 : Form
    {
        int i = 1;
        public Form1()
        {
            InitializeComponent();
            pictureBox1.Image = Image.FromFile(i + ".jpg");
        }
        private void rightButton_Click(object sender, EventArgs e)
        {
            i++;
            pictureBox1.Image = Image.FromFile(i + ".jpg");
        }
    }
}
```

The bits you're adding are circled, an int that is the image number you're currently looking at (starts at 1) and a bit of code when you click the rightButton to add one (go to next image).

The key bit is the pictureBox1.Image = Image.FromFile() bit. pictureBox1 as some of you may have realised is the name of the picturebox we added. It's Image is a **property** or **field** (like I we created above but hidden.) it stores the current image. The Image on the other side of the equals is the **Class** Image, the blueprint that tells C# "this is all the built in code to do with images".

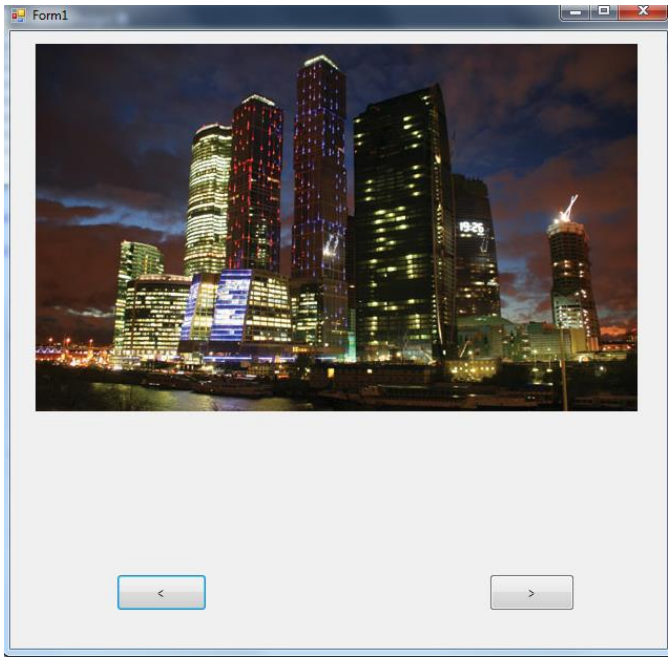
We make use of a C# built in piece of code called FromFile() which just says “get the file at this location as an image”.

Now, a bit about something we’re going to do for simplicity, as this is a fairly simple program we’re going to do a couple things.

First, we’re going to stick all our images in the folder that our program is in, for debugging (which we’re going) this is usually “**documents/Visual Studio 2010/Projects/*ProjectName*/ *ProjectName*/bin/Debug”.**

In my case I called it SlideshowTut, so the path is; “**documents/Visual Studio 2010/Projects/*SlideshowTut*/ *SlideshowTut* /bin/Debug”.** Inside that folder is all the files associated with debugging, (not the code) dump the images in there (I’ve supplied a zip of 5 demo images). Make sure the first one has the name **“1”,** second **“2”** and so on and that they are all **jpg** files.

Now that’s done, give it a try. This was mine;



You will notice though, that if you press too many times the program crashes, that’s because it runs out of images. We need a check to stop it loading an image that doesn’t exist we also need to get the back button working. (Note:  the stop button at the top will exit debugging).

So, add the circled bit, this just checks if we’re above the fifth image (in this demo only the first 5 images will be loaded, if you’ve more simply change it to the amount of numbers you have) if it is, reset back to image 1.

This is an example of **selection**

```
private void rightButton_Click(object sender, EventArgs e)
{
    i++;

    if (i > 5)
    {
        i = 1;
    }

    pictureBox1.Image = Image.FromFile(i + ".jpg");
}
```

Left Button

You need to go back to the form view and double click the left button, the code for this is almost identical.

Instead of adding one, you subtract one (i--), and check if it's below 1, if it is you set I to 5. The last line remains the same. Have a go.

Automatic looping

The last thing we're going to do is add a looping function that just keeps cycling all the images.

For that, we want to add a third button, I named it repeatButton and gave it text Repeat, the code is below. It's simply a loop that counts from 1 to 5, each time it goes around it outputs that current image, waits until that image is visible (to ensure it works on slow and fast computers), then waits 2000 milliseconds, which is 2 seconds. This works by suspending the current "thread" (google this if you want to know more) or suspending the currently running bit of the program.

```
for (int i = 1; i < 6; i++)
{
    /**
     * this is a loop that counts from 1 to 5 and each time it goes
     * round it loads a new picture
     * */
    pictureBox1.Image = Image.FromFile(i + ".jpg");
    Application.DoEvents(); //allow windows to execute all pending tasks including your image show...
    System.Threading.Thread.Sleep(2000); // wait 2000 milliseconds (or 2 seconds)
}
```

It will only loop once, but for our purposes that's fine, it demonstrates proper use of "**iteration**"

Challenges

1. Make it so the program loops forever
2. Make it so the program works with more (or infinite) images
3. Make it so the images the program uses are pulled from a website (this requires thought).