# Building Forms with PowerShell – Part 1 (The Form)

Stephanie Peters April 23, 2012

When I teach PowerShell workshops, I'm frequently asked if there are any good resources for learning about how to build forms in PowerShell. There are a number of good resources out there to be sure, but you can never have too many. This is my contribution to the education of the scripting masses. I'm going to start very simple, and over the next several posts, we'll build up an advanced form repertoire.

This first post is about the form itself. The form is just a window on which we can put useful (or whatever) things to build a GUI. For the most part, the form is the easiest of the elements to work with, but you can get somewhat fancy with it. We'll explore some different form scenarios.

## The Simplest Form

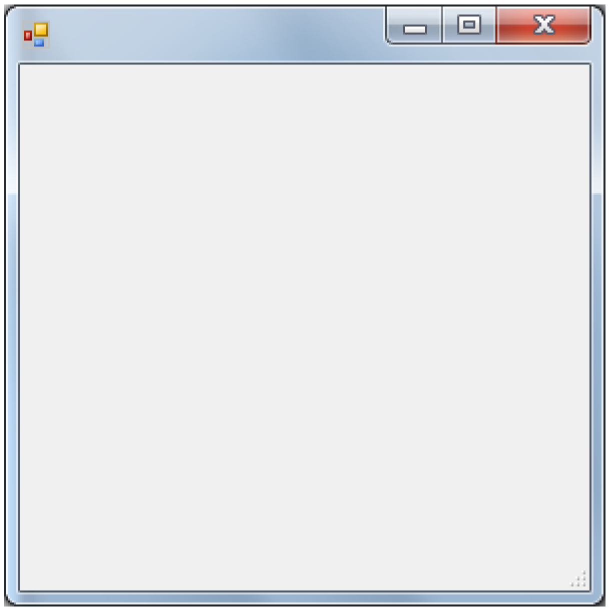
There are only three things required to launch a form from PowerShell:

1. You must load the System.Windows.Forms assembly (if not already done);
2. You must create a new object of type system.windows.forms.form; and,
3. You must call the ShowDialog() method on your new object. Note—don't try calling the Show() method, unless you want your script to hang.

For example:

```
Add-Type -AssemblyName System.Windows.Forms

$Form = New-Object system.Windows.Forms.Form

$Form.ShowDialog()
```

Depending on your Windows shell environment, this is what your form might look like:



Of course, that's not terribly useful, so we'll keep going.

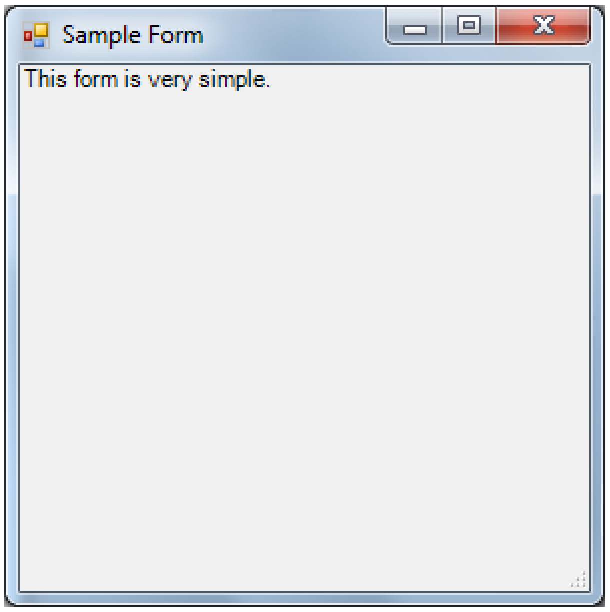
## Adding Some Text

Text will be helpful in adding some purpose to our form. There are a couple of pieces of text we'll add.

First, let's add a window title for the form by setting the form's Text property. This will show up in the top of the form and will also serve as the window title as shown in the task bar.

Then we'll add some text in the main window by adding a label control. In order to add a label, we need to create a new Label object, set the relevant properties and add it to the Controls collection of the form. For now, we'll set only two properties—Text and AutoSize. Enabling AutoSize will ensure that our label is large enough to display all our text.

```
Add-Type -AssemblyName System.Windows.Forms
$Form = New-Object system.Windows.Forms.Form
$Form.Text = "Sample Form"
$Label = New-Object System.Windows.Forms.Label
$Label.Text = "This form is very simple."
$Label.AutoSize = $True
$Form.Controls.Add($Label)
$Form.ShowDialog()
```

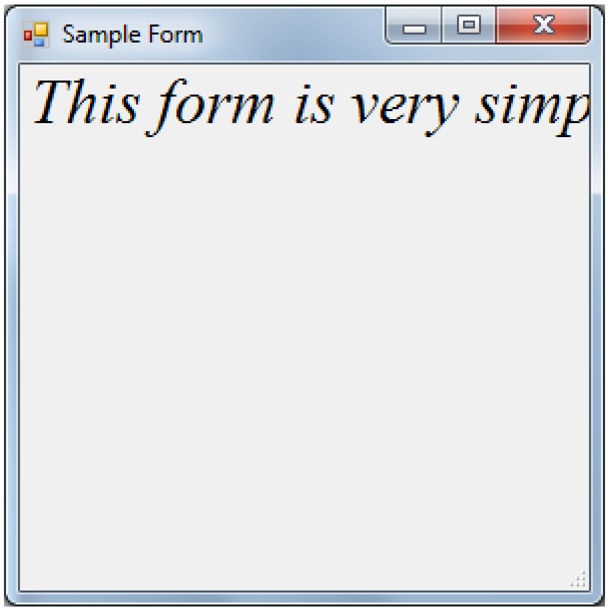


This form is very simple.

Note – Most forms will contain at least several controls. Usually there would be at least one button. We'll add more controls later in this series, but for now we'll just stick with a single label.

We might want to change the font used by the form. In order to change the font, we need to create a new font object with the appropriate properties and assign the new font object to the form's Font property.

```
Add-Type -AssemblyName System.Windows.Forms

$Form = New-Object system.Windows.Forms.Form

$Form.Text = "Sample Form"

$Font = New-Object System.Drawing.Font("Times New
Roman",18,[System.Drawing.FontStyle]::Italic)

# Font styles are: Regular, Bold, Italic, Underline, Strikeout

$Form.Font = $Font

$Label = New-Object System.Windows.Forms.Label

$Label.Text = "This form is very simple."

$Label.AutoSize = $True

$Form.Controls.Add($Label)

$Form.ShowDialog()
```

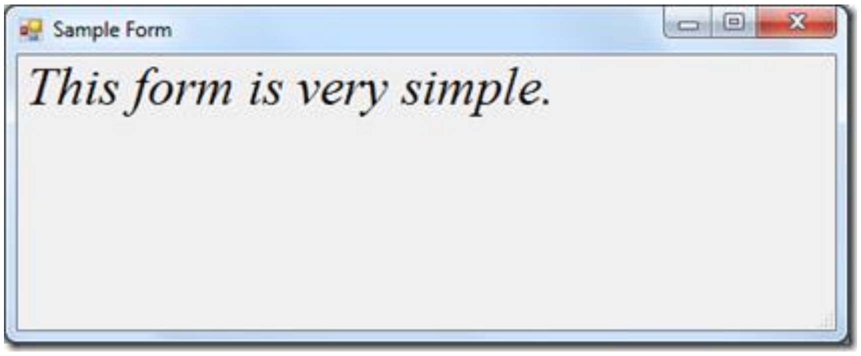


This form is very simp

Now we have a bit of a problem because the text is too large to fit on the form in its default size. There are a few different ways we could address this:

- Specify a form size – we could set a particular height and width that would be large enough. This would work, but it's somewhat difficult to figure out what the appropriate size should be.

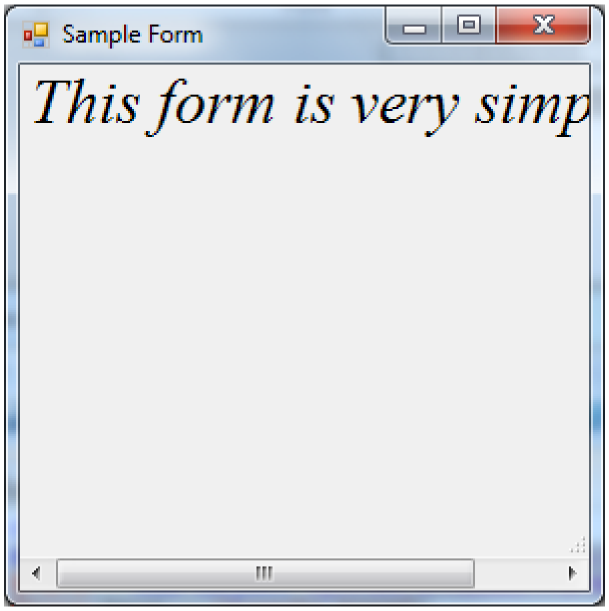
  ```
  $Form.Width = 500
  $Form.Height = 200
  ```

  This form is very simple.

- Enable scroll bars – By enabling AutoScroll, we allow the user to scroll over to see the remaining text. This is obviously not an optimum solution, but scroll bars are probably a good idea when necessary.

  ```
  $Form.AutoScroll = $True
  ```



This form is very simp

- Enable AutoSize – This is a much better idea for our purposes. By enabling AutoSize on the form, it will automatically resize to accommodate whatever controls we add to it.

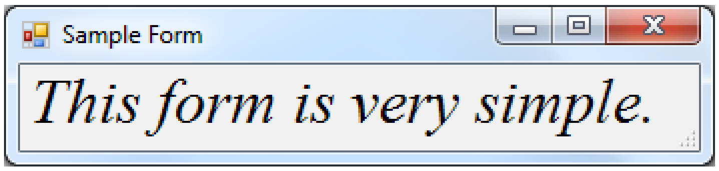
```
$Form.AutoSize = $True
$Form.AutoSizeMode = "GrowAndShrink"
    # or GrowOnly
```

  However, you should know that enabling AutoSize in this way will prevent the user from manually resizing the form. The strategy you choose will depend on your specific requirements.

Here is the new form code:

```
Add-Type -AssemblyName System.Windows.Forms

$Form = New-Object system.Windows.Forms.Form

$Form.Text = "Sample Form"


Form.AutoScroll = $True

$Form.AutoSize = $True

$Form.AutoSizeMode = "GrowAndShrink"

    # or GrowOnly


Font = New-Object System.Drawing.Font("Times New
Roman",24,[System.Drawing.FontStyle]::Italic)

    # Font styles are: Regular, Bold, Italic, Underline, Strikeout

$Form.Font = $Font

$Label = New-Object System.Windows.Forms.Label

$Label.Text = "This form is very simple."
```

```
$Label.AutoSize = $True
$Form.Controls.Add($Label)
$Form.ShowDialog()
```



This form is very simple.

It’s looking better.  Now let’s keep going.

## Window Settings

There are a number of window settings we can customize for our form.  Here we’ll experiment with the following:

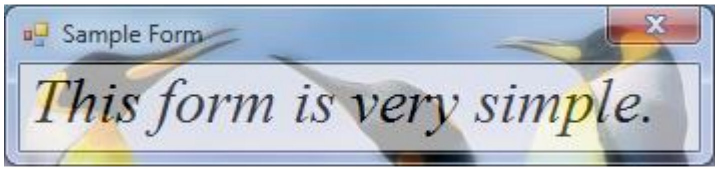
- Disabling the minimize and maximize functionality
- Setting the window state to normal (versus maximized or minimized)
- Hiding the size grip (usually in the lower right corner of the window)
- Setting the opacity to 70%, making the window partially transparent
- Setting the window start position so that the window will open in the middle of the visible screen

```
Add-Type -AssemblyName System.Windows.Forms
$Form = New-Object system.Windows.Forms.Form
$Form.Text = "Sample Form"
Form.AutoScroll = $True
$Form.AutoSize = $True
$Form.AutoSizeMode = "GrowAndShrink"
    # or GrowOnly
```

```
$Form.MinimizeBox = $False
$Form.MaximizeBox = $False
$Form.WindowState = "Normal"
    # Maximized, Minimized, Normal
$Form.SizeGripStyle = "Hide"
    # Auto, Hide, Show
$Form.ShowInTaskbar = $False
$Form.Opacity = 0.7
    # 1.0 is fully opaque; 0.0 is invisible
```

```
$Form.StartPosition = "CenterScreen"
    # CenterScreen, Manual, WindowsDefaultLocation, WindowsDefaultBounds,
CenterParent
```

```
Font = New-Object System.Drawing.Font("Times New
Roman",24,[System.Drawing.FontStyle]::Italic)
    # Font styles are: Regular, Bold, Italic, Underline, Strikeout
$Form.Font = $Font
$Label = New-Object System.Windows.Forms.Label
$Label.Text = "This form is very simple."
$Label.AutoSize = $True
$Form.Controls.Add($Label)
$Form.ShowDialog()
```



This form is very simple.

Note – I've positioned this form over the well-known sample image penguins.jpg to demonstrate that the form is 30% transparent.

## Visual Stimulation

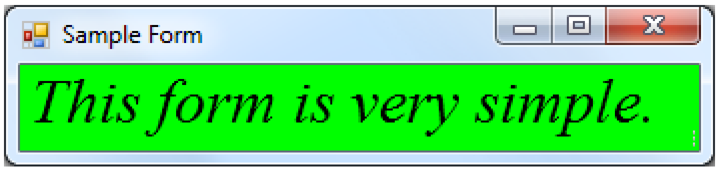
Note – for this section, we'll revert the window settings changes we made to the form in the previous section.

There are a few things we can do to make our form pop visually.  The easiest way to change the look and feed is to change the background color of the form.

```
Add-Type -AssemblyName System.Windows.Forms
$Form = New-Object system.Windows.Forms.Form
$Form.Text = "Sample Form"
$Form.AutoSize = $True
$Form.AutoSizeMode = "GrowAndShrink"
    # or GrowOnly
```

```
$Form.BackColor = "Lime"
    # color names are static properties of System.Drawing.Color
    # you can also use ARGB values, such as "#FFFFEBCD"
```

```
$Font = New-Object System.Drawing.Font("Times New
Roman",24,[System.Drawing.FontStyle]::Italic)

    # Font styles are: Regular, Bold, Italic, Underline, Strikeout

$Form.Font = $Font

$Label = New-Object System.Windows.Forms.Label

$Label.Text = "This form is very simple."

$Label.AutoSize = $True

$Form.Controls.Add($Label)

$Form.ShowDialog()
```



This form is very simple.

Hmmm, interesting, but maybe not quite what we were looking for.

Let’s try adding a visually pleasing background image for our form.  In order to do that, we need to do the following:

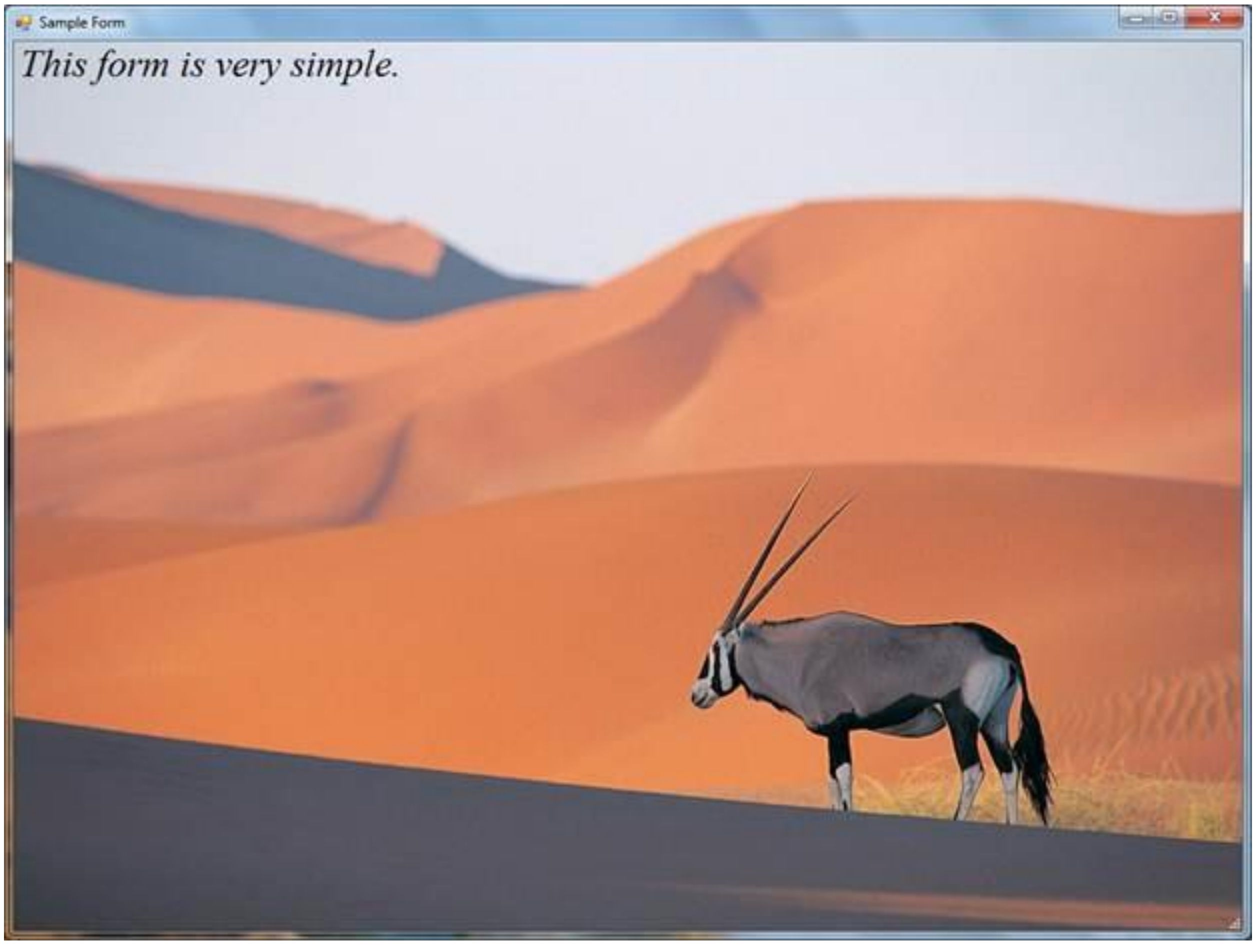
- Create a new system.drawing.image object from an image file (in this the sample picture “Oryx Antelope.jpg) and assign it to the BackgroundImage property of the form
- Set the BackgroundImageLayout to None (The default is Tile, which tiles the image across the form as many times as it will fit.)
- Remove the AutoSize feature from the form and instead set the size of the form to mage the image dimensions.
- Configure the label so that it’s background is transparent, so as not to obstruct the picture

```
Add-Type -AssemblyName System.Windows.Forms

$Form = New-Object system.Windows.Forms.Form

$Form.Text = "Sample Form"


$Image = [system.drawing.image]::FromFile("$($Env:Public)\Pictures\Sample
Pictures\Oryx Antelope.jpg")

$Form.BackgroundImage = $Image

$Form.BackgroundImageLayout = "None"

    # None, Tile, Center, Stretch, Zoom


$Form.Width = $Image.Width

$Form.Height = $Image.Height
```

```
$Font = New-Object System.Drawing.Font("Times New
Roman",24,[System.Drawing.FontStyle]::Italic)

    # Font styles are: Regular, Bold, Italic, Underline, Strikeout

$Form.Font = $Font

$Label = New-Object System.Windows.Forms.Label

$Label.Text = "This form is very simple."
```

```
$Label.BackColor = "Transparent"
```

```
$Label.AutoSize = $True

$Form.Controls.Add($Label)

$Form.ShowDialog()
```



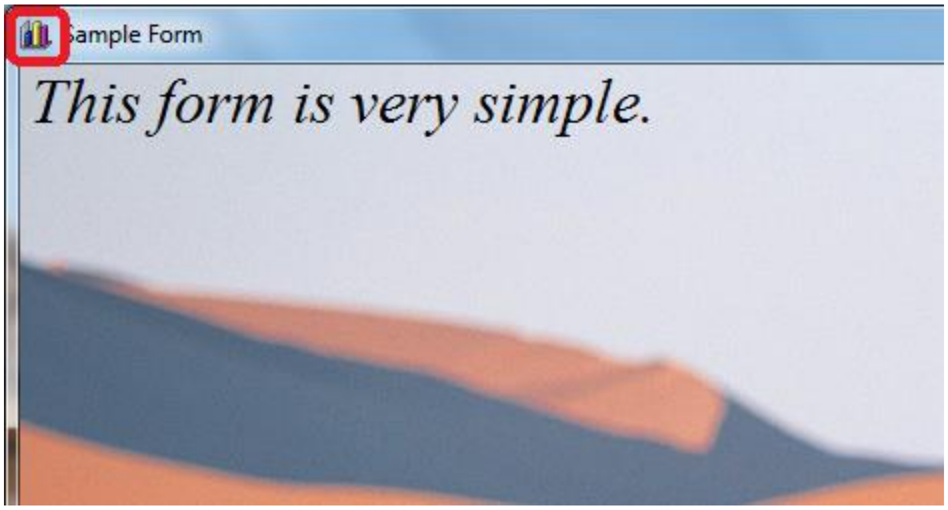
*This form is very simple.*

Very nice.

## Finishing Touch

One last thing we can do to refine our form is to add a customized icon. We can load an icon from an image file (in this case GRAPH.ICO which is part of Office). A good resource for creating custom icon files is http://www.xiconeditor.com/.

```
$Icon = New-Object system.drawing.icon ("C:\Program Files\Microsoft
Office\Office14\GRAPH.ICO")

$Form.Icon = $Icon
```



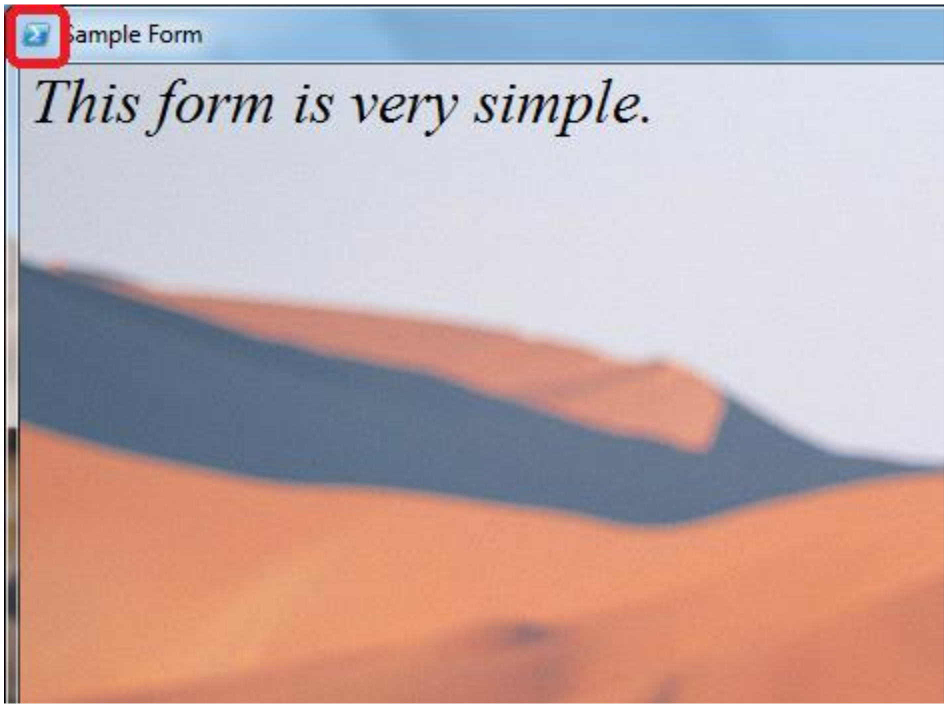
Alternatively, we can extract an icon from an associated file, such as PowerShell.exe.

```
Add-Type -AssemblyName System.Windows.Forms

$Form = New-Object system.Windows.Forms.Form

$Form.Text = "Sample Form"
```

```
$Icon = [system.drawing.icon]::ExtractAssociatedIcon($PSHOME + "\powershell.exe")

$Form.Icon = $Icon
```

```
$Image = [system.drawing.image]::FromFile("$($Env:Public)\Pictures\Sample
Pictures\Oryx Antelope.jpg")

$Form.BackgroundImage = $Image

$Form.BackgroundImageLayout = "None"

    # None, Tile, Center, Stretch, Zoom

$Form.Width = $Image.Width

$Form.Height = $Image.Height

$Font = New-Object System.Drawing.Font("Times New
Roman",24,[System.Drawing.FontStyle]::Italic)

    # Font styles are: Regular, Bold, Italic, Underline, Strikeout
```

```
$Form.Font = $Font
$Label = New-Object System.Windows.Forms.Label
$Label.Text = "This form is very simple."
$Label.BackColor = "Transparent"
$Label.AutoSize = $True
$Form.Controls.Add($Label)
$Form.ShowDialog()
```



This form is very simple.

Beautiful.

So we've seen most of the commonly used functionality of Windows Forms.  For more esoteric customization, start with[http://msdn.microsoft.com/en-us/library/system.windows.forms.form.aspx](http://msdn.microsoft.com/en-us/library/system.windows.forms.form.aspx).