AeroGlass Library

Version 1.2

Contents:

The AeroGlass library accesses the dwmapi dll to extend the frames of windows for use with the C# language, this gives the effect of having more glass areas on a window or making a window look as if it is completely glass. The library aims to make implementing this method easier so that you can achieve this effect with minimal code and will not have to access the dll yourself, version 1.2 of the library is compatible with both windows forms and WPF forms, though the implementation differs slightly. This documentation covers the implementations of both types.

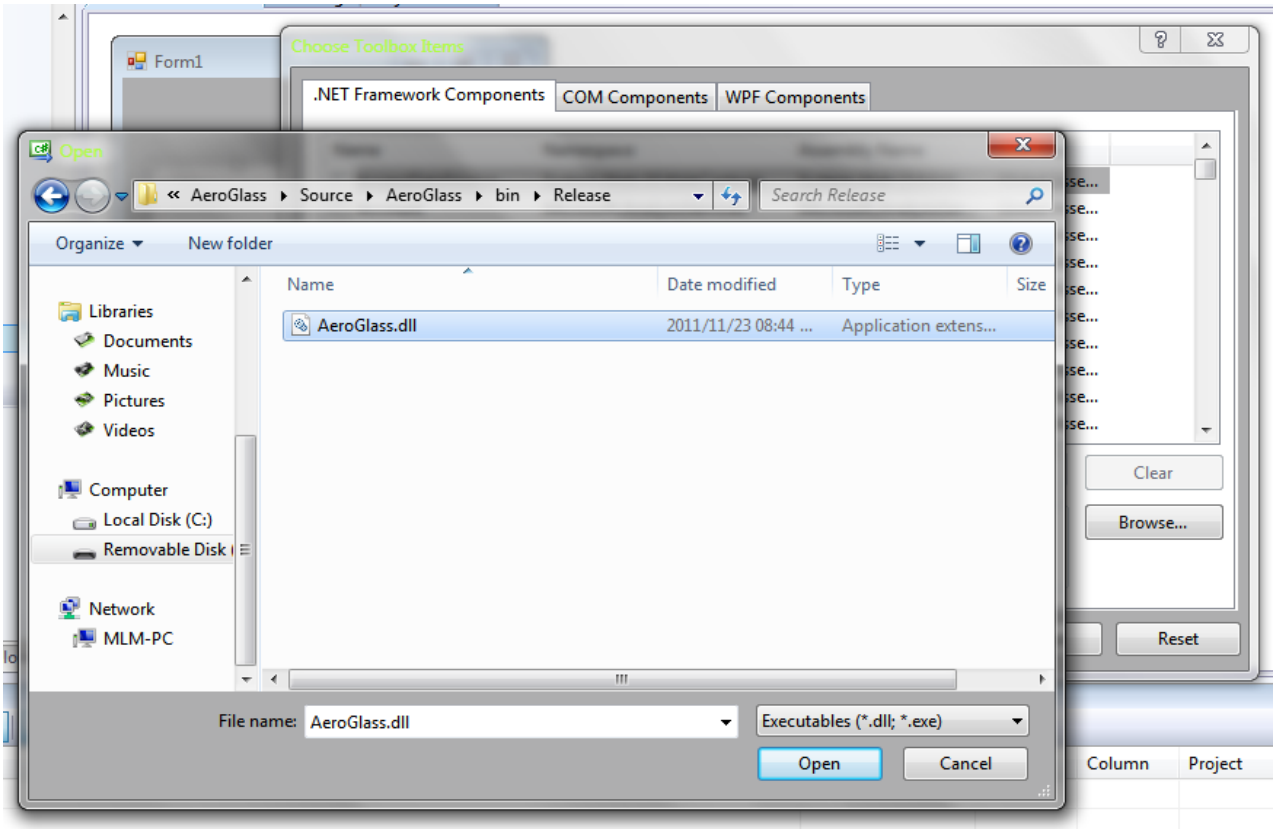
Usage with windows forms

There are 2 ways of implementing glass on a Windows form using the AeroGlass library, one is to use the GlassPane controls AutoExtend property and the other is to use code, let us first take a look at the AutoExtend property method.

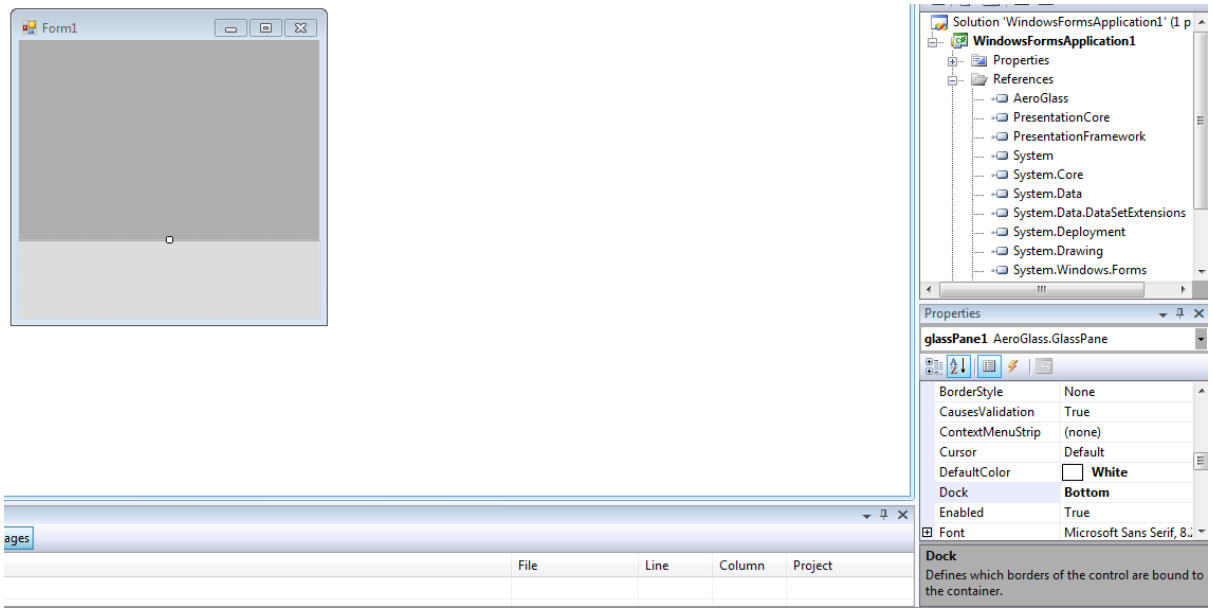
AutoExtend

Before we begin it is worth noting that if you use the AutoExtend method you can only place one GlassPane, if you place another it will not work or errors may occur, it is for this reason that AutoExtend is false by default.

For AutoExtend to work, the glass pane must be docked, this will be explained later in the code section. To make a form with an extended glass frame using this method, add the control to your toolbox by right clicking the toolbox and selecting choose items



Following this, place the GlassPane control , dock it as you wish and set the AutoExtend property to true, here I have docked mine on the bottom of the form.



Now running the application should produce the following result:

Using Code

Now let us take a look at using code to implement an extended glass frame. When using code ,you are still required to use the GlassPane, this is because all the code actually does is extend the frame of the window inwards, if we just use the code and not the GlassPane the inner part of the window will still cover the extended frame. This can be seen if you look carefully at the image below.

To use the library ,we must first add a reference to the AeroGlass dll, placing a GlassPane on the form will automatically do this, then we add the using statement to use the Classes in the library.

```
using AeroGlass;
```

Now, in the form class, create a new instance of the Glass class:

```
Glass g = new Glass();
```

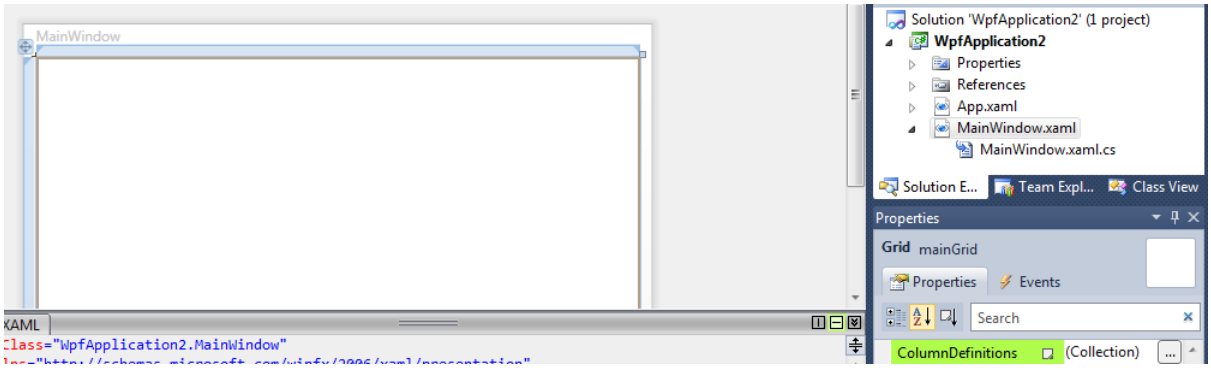
Place a GlassPane on your form and dock it as you desire, I have docked mine on the bottom as I did previously, but this time AutoExtend is set to false. Then in the Form_Load event, call the extendFrame function:

```
g.extendFrame(0, glassPane1.Height, 0, 0, this);
```

The extendFrame function has the following parameters: int Top , int Bottom, int Left, int Right Form form. ”this” refers to the current form. Note that I used the height of the GlassPane as my margin width for the bottom margin. Running this code will yield the same result as seen previously

Usage with WPF forms

Using the AeroGlass Library with a WPF form is a little different but the concept is still the same. Starting with a Blank WPF form, select the main grid in the center of the form and give it a name ,have named mine mainGrid.



Next ,add a reference to the AeroGlass dll and include the following using statement in the code of your window:

```
using AeroGlass;
```

Then in the class:

```
WpfGlass wg = new WpfGlass();
```

Finally we add this line to the Window_loaded event:

```
wg.extendFrame((int)mainGrid.Height, 0, 0, 0, this);
```

As you can see, the structure of this last statement is the same as that used in the Windows Form, the only difference is that the height of mainGrid had to be cast as an integer. Running this should result in the following:

Don’t like a completely glass window, simply change the size of the grid or choose different properties and parameters to parse or overlay the window with different controls.

Here I have overlaid a semi-transparent rectangle to give a nice effect and added a semi-transparent textbox control: