

Forms and Controls

C# Programming

January 17

Office Hours

Ravi:

Wednesdays, 11-12am

Amanda:

Tuesdays, 11-12am

Thursdays, 3-4pm

All office hours will be held in Moore 100B

Custom Controls

Any questions?

Part I

Forms and Controls

Good Design

- It is easy to quickly put together a GUI application using the Form Designer
- But it is important to make sure the overall design of the GUI is attractive and easy to use
- Today we will look at a few things to help “polish” your apps

Window Resizing

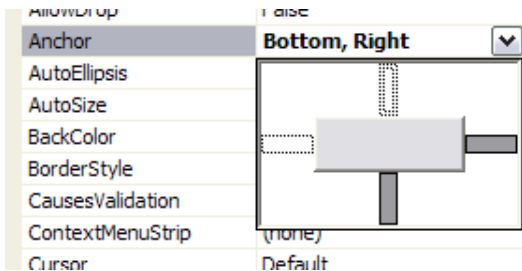
- Some applications – like Calculator – are designed to be a fixed size
- If this is the case, you should actually make the window a fixed size!
- To prevent the form's border from being resized, set its `FormBorderStyle` property to `FixedSingle` or `Fixed3D`
- If your window's border is fixed, you probably don't want to allow it to be maximized either
- Set the `MaximizeBox` property to `False`

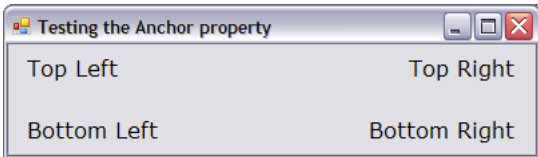
Window Resizing

- The majority of applications *should* be resizable
- When a window is resized, we want the controls to stretch and move appropriately to maintain the same organization and usefulness
- The `Anchor` property of a control defines how it is **repositioned** when the window's border changes
- By default, a control's `Anchor` is set to `AnchorStyles.Top | AnchorStyles.Left`
- Note the use of the **bitwise-OR** operator (often used when setting multiple flags)
- This can be read as “the control is anchored to the top and to the left” – meaning the distance from the control to the left and top borders will remain fixed

Window Resizing

- From the Designer, you can choose which sides to anchor to

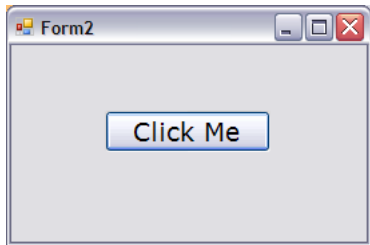






Window Resizing

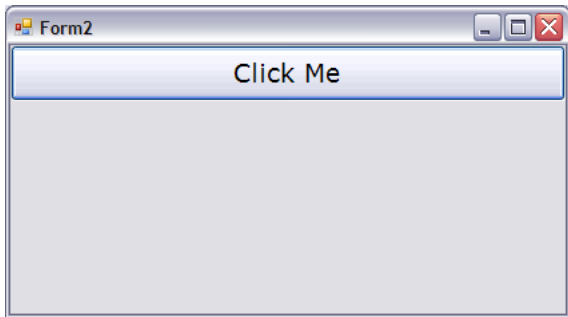
- Depending on the type of control, it can also be **resized** when the window's size changes
- For example, if you set the `Anchor` of a button to all four sides, these distances will remain constant by changing the size of the button





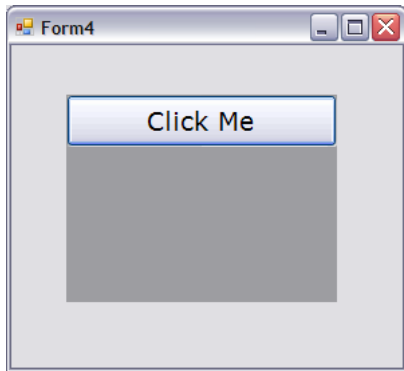
Window Resizing

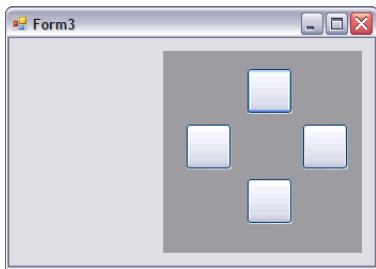
- Another possibility is to have a control fill the entire side of a window
- Setting a control's Dock property to Top, for example, will bind the control the top so that it extends the entire distance from the left to right borders
- Setting a control's Dock property to Fill will bind it to occupy the entire area of its container

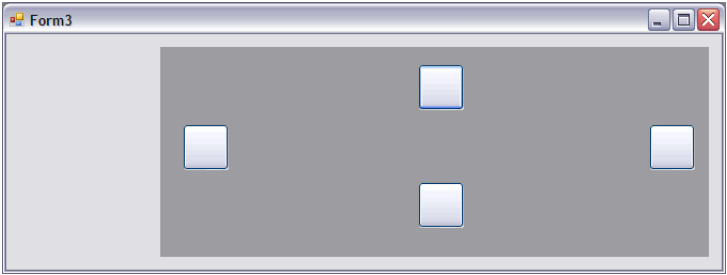


Panels

- A panel control is a container – it holds other controls
- A panel has properties like `BackColor` and `Font` that correspond to the area of the panel
- You can use panels to group similar GUI widgets
- Controls can be added to a panel instead of the form directly
- For controls added to a panel, the values of properties `Location`, `Anchor`, `Dock`, etc. are **in relation to the panel**, not the form

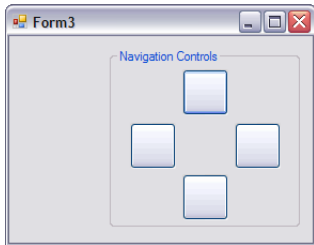






Group Boxes

- A group box is similar to a panel
- It draws a visible border around the controls within it
- It also has a caption



Splitter

- You can divide your window into resizable panels
- Play with `SplitContainer` and `Splitter` if you're interested

Fonts

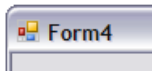
- If you want to change the font of all controls on a form, the hard way is to change the `Font` property of each control individually
- The easy way is to set the `Font` of the container – either the form itself or a panel or group box that contains controls
- This should set the `Font` property of all controls within the container

Transparency

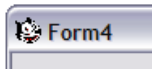
- To allow a control's `BackColor` to take the color of its container, set it to `Color.Transparent`

Icons

- A default window icon is used if none is specified



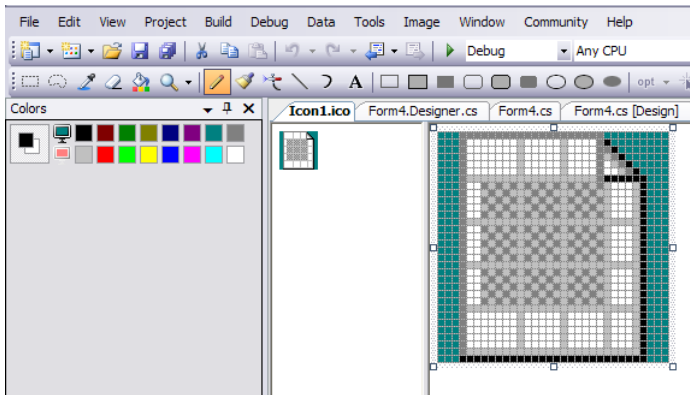
- To change this, set the form's Icon property to an icon file (.ico)



- Note that this also changes the icon used in the Task Switcher

Icons

- There is freeware that converts images to icon files
- Visual Studio also has an Icon Designer
- File→New→File→Icon File



Icons

- You can also change the icon associated with the compiled .exe
- By default, the icon is:



Demo.exe

- To change this, right-click the project in the Solution Explorer and select Properties
- In the Application tab, you can select an icon



Demo.exe

Part II

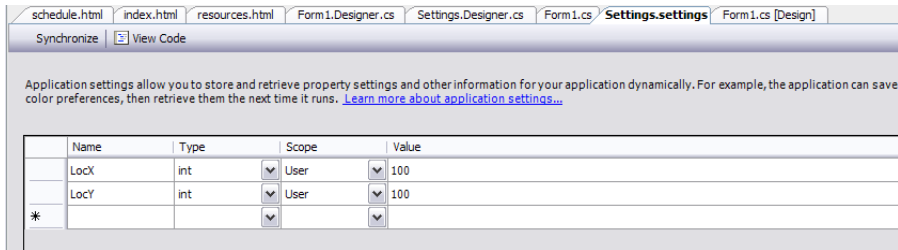
Settings Files

Settings

1. When an application is run, it is useful to allow certain properties – like size, location, color – to be configurable
 2. When an application terminates, it is often useful to save information to be used next time it is run
- Both of these can be accomplished using **settings files**

Settings

- A settings file is usually created with a new project, called `Settings.settings`
- You can add settings files to any project
- Through the Designer, you can add variables to be stored in the settings file:



Synchronize | View Code

Application settings allow you to store and retrieve property settings and other information for your application dynamically. For example, the application can save color preferences, then retrieve them the next time it runs. [Learn more about application settings...](#)

	Name	Type	Scope	Value
	LocX	int	▼ User	▼ 100
	LocY	int	▼ User	▼ 100
*			▼	▼

Settings

- Each variable can be assigned one of two scopes
- **Application Scope:** These get stored in a common `app.config` file and can only be read at run-time
- **User Scope:** These get stored in `user.config` that is specific to the user running the application can be read and written at run-time
- The location of `app.config` is usually in the same directory as the project
- The location of `user.config` is something like `\Documents and Settings\\Local Settings\Application Data\\<appname+id>\<appversion>`

Example

An example that saves the location of a form before exiting and positions the next instance of the form in the same location...

```
private void Form1_FormClosing(
    object sender, FormClosingEventArgs e) {
    Properties.Settings settings =
        Properties.Settings.Default;
    settings.LocX = this.Location.X;
    settings.LocY = this.Location.Y;
    settings.Save();
}
```

```
private void Form1_Load(object sender, EventArgs e) {
    Properties.Settings settings =
        Properties.Settings.Default;
    this.Location =
        new Point(settings.LocX, settings.LocY);
    this.Text = "" + settings.LocX + " " + settings.LocY;
}
```

Resources

- You can also add **resource** files to your project to save strings, images, and other files
- Check the Resources webpage for links to learn more about settings and resource files