**Hints on Creating a Dialog for your Game**

**Background**: When you get a complaint that the compiler doesn’t understand a ‘using…’ reference, it means that you have to tell the compiler that it needs to refer to a specific library when building the project (for efficiency, it doesn’t look at every library on your machine). The class documentation will tell you what ‘assembly’ contains a particular class. For example, to use Forms in my xna project, I had to add a reference to the System.Windows.Forms assembly. To do this, in Solution Explorer, click Right on the References folder. Pick ‘Add Reference’. The Forms assembly will be in the .Net tab in the ‘Add Reference’ window. You’ll need to follow this process any time you get the ‘using…’ complaint.

**Creating a Dialog**

1. Create another project (a Windows Forms project). This will be a test project for building your dialog box.
2. Add a dummy button on your **Form1**. Create a click handler. Once you’ve created the dialog box (see instructions below), you’ll add code here to display/test the dialog.
3. In Solution Explorer, click right on the project, select ‘Add Windows Form’, and add another form (in addition to Form1) to your project. This will be the form for your dialog. Let’s name it **FormSettings**.
4. Drag two buttons onto **FormSettings**. Set the text to **OK** and **Cancel** for the two buttons. In the properties for the **OK** button, set **DialogResult** to **OK**. Similarly, set your **Cancel** buttons’s **DialogResult** to **Cancel**.
5. Back in your **Form1** button click handler, add code to display the dialog and test the dialog result. The code would look something like:

FormSettings dlg;

dlg = new FormSettings();

if ( dlg.ShowDialog() == DialogResult.OK )

{

// save settings in your application

}

1. Now that you have your basic Dialog set up, you can add the necessary controls to it. After returning from the ShowDialog, you can check the controls to extract the appropriate values. For example, add a combo box to your FormSettings. In its properties, set the Modifiers to Public, and set Items to the list of items you want to show. Check the value by doing something like:

if ( dlg.ShowDialog() == DialogResult.OK )

{

int choice = dlg.cmbLevel.SelectedIndex; // user’s choice

// use choice…

}

1. In a settings dialog such as this, the dialog should always display the *current* settings when it is displayed. To handle this, add parameters to the FormSettings constructor that will allow you to pass in values indicating the current settings. The FormSettings constructor can use these values to initialize the controls to the appropriate values, so when it displays it will show the current settings. Make sure all of your initialization call is AFTER the call to **InitializeComponent**().

**Using the dialog in your XNA application.**

Once you have tested your FormSettings in your Forms project, save everything, exit Visual Studio, and then do the following:

1. Copy the following files from your Forms test project into your Xna project. ***Put the files in the same folder with the other source files in the XNA project.*** The files to copy are: **FormSettings.cs**, **FormSettings.resx**, **and FormSettings.Designer.cs**. All three files must be copied!
2. Open your **xna** project in Visual Studio. Click right on the project in Solution Explorer, select **Add Existing**, and add the three files to the project.
3. Edit FormSettings.cs and FormSettings.Designer.cs as needed to make the namespace match with your xna namespace name.
4. You should now be able to use FormSettings in your Game1 button click event handler just like you did in your Forms project.

An assembly is a generic term for an executable or a DLL.

DLL stands for “Dynamic Link Library” – Dynamic is always a runtime thing as opposed to a compile time thing

Copy three source codes: .cs, .designer, .resx and put into place where the rest of your source code is and include them in the project.

You will have to change the namespace manually