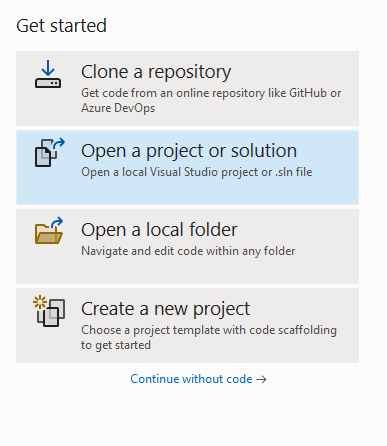
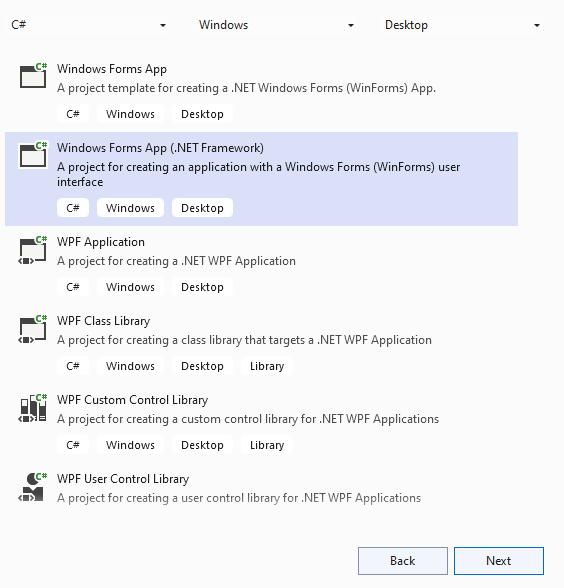
**Windows Forms Orientation**

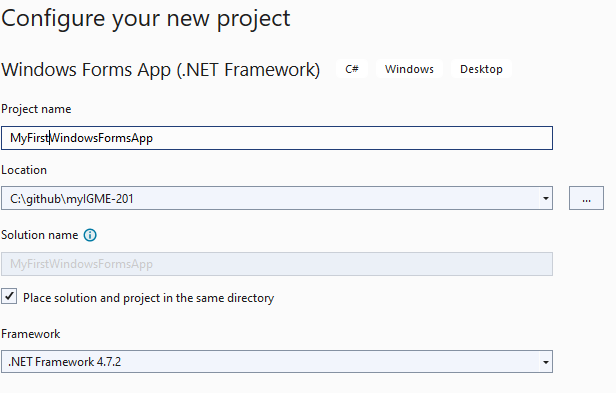
1. Start Visual Studio 2022 and select Create New Project



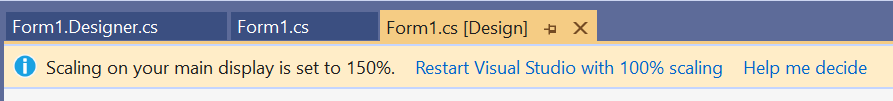
1. Search for C# / Windows / Desktop and select “Windows Forms App (.NET Framework)”. You must select the **.NET Framework** for compatibility purposes. Click Next.



1. Change the Project name to “MyFirstWindowsFormsApp” and set the location to your myIGME-201 folder. Click Create.

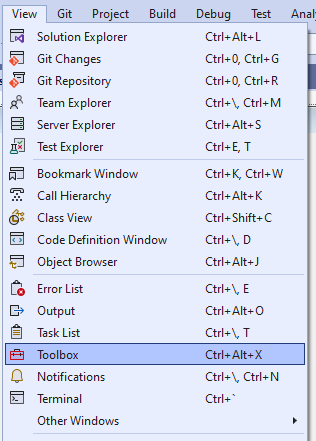
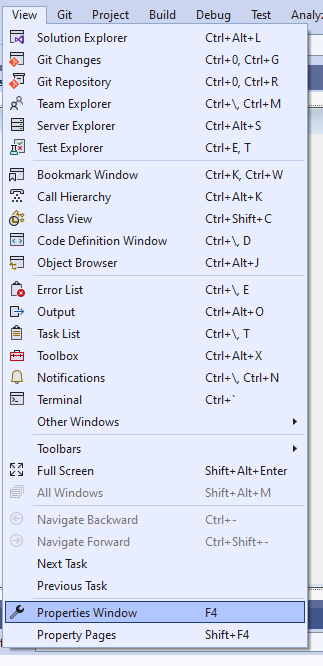


1. A message will be displayed if your monitor’s scaling is not set to 100%

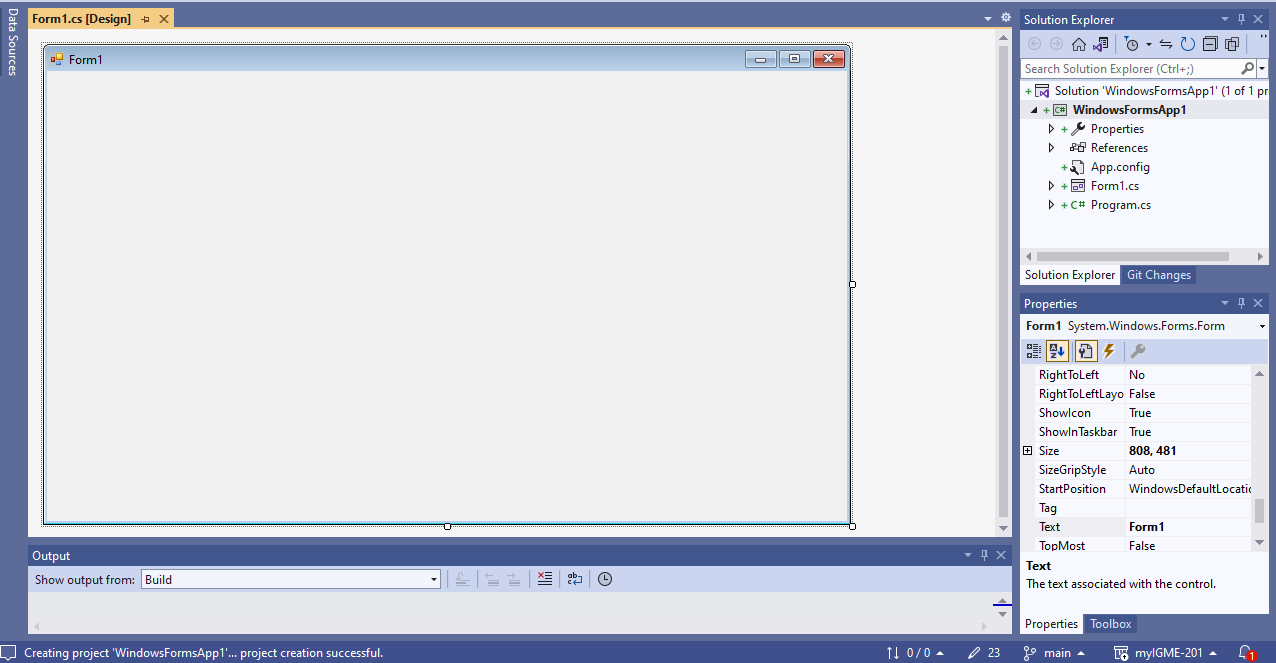


Follow the instruction in the “Scaling Issues” section of the “Special Topics” on page 7 of the “Windows Forms Controls.docx” document in myCourses Content to set your monitor to 100% scaling while using the Visual Studio Form Designer.

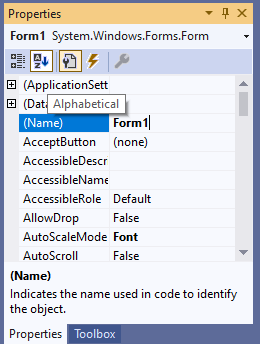
1. Select View/Solution Explorer, View/Git Changes, View/Toolbox and View/Properties Window

Arrange the panels so that the IDE looks similar to the following, with the “Solution Explorer” and “Git Changes” panels tabbed and docked to the top right, and the “Properties” and “Toolbox” panels tabbed and docked to the bottom right:



1. Click “Form1” in the form title bar to select the Form Control (NOT “Form1.cs [Designer]”). In the “Properties” panel tab, click the A-Z icon (the second icon from the left) to display the properties in alphabetical order. The default is to display the properties organized by category, but that makes it very hard to find the properties. Furthermore, the lightning bolt icon gives access to the event handlers of the selected control. We will NOT use this tab, but will instead explicitly code the event handlers ourselves. Refer to “Event Handlers” in the “Windows Forms Controls.docx” document in myCourses Content.



Set the following properties for the Form:

* (Name): HoudiniForm
* Size: 251, 375
* Text: Houdini

Forms are classes and therefore are defined using PascalCase, whereas all controls on a form are member variables of the Form class, and therefore use camelCase.

1. Select the Toolbox panel tab, expand “All Windows Forms” and double-click “PictureBox” to add a PictureBox control to the form.
2. Select the PictureBox and select the “Properties” panel tab and set the following properties for the PictureBox control:

* (Name): houdiniPictureBox
* Location: 31,28
* Size: 180,228
* SizeMode: Zoom

1. Select the Toolbox panel tab and double-click “Button” to add a Button control to the form.
2. Select the Button and select the “Properties” panel tab and set the following properties for the Button control:

* (Name): exitButton
* Location: 86,291
* Size: 75,23
* Text: Exit

1. If Visual Studio automatically generates code, such as either of the following methods:

* private void houdiniPictureBox\_Click(object sender, EventArgs e) { }
* private void exitButton\_Click(object sender, EventArgs e) { }

***Leave the code alone.*** If you delete it without modifying the “Events” tab of the control’s properties (the lightning icon), then the form will break and you will not be able to edit it. It is best to leave the code there. It won’t cause any harm. Visual Studio generates the event handler method using the control name, a single underscore, and the event name. Note that it does not follow the convention that methods should be PascalCase. We will always write our own event handlers using PascalCase and 2 underscores between the control name and the event, as you will see below.

1. Right-click on the white space of the form designer and select “View Code” from the context menu.
2. Modify the source code to match the following:

using System;

using System.Windows.Forms;

namespace MyFirstWindowsFormsApp

{

// Class: HoudiniForm

// Purpose: A form that shows a picture of Houdini, but hides him if the mouse hovers over it

// Restrictions: None

public partial class HoudiniForm : Form

{

// the Form constructor initializes control properties and event handlers

public HoudiniForm()

{

// auto-generated method that creates the controls from the Form Designer

// InitializeComponent() is defined in the auto-generated file Form1.Designer.cs

// THIS MUST BE THE FIRST STATEMENT IN THE FORM CONSTRUCTOR

InitializeComponent();

// every Windows control has a Tag field that can be used for any purpose

// let's use it to determine whether the houdiniPictureBox is visible

this.houdiniPictureBox.Tag = true;

// set the URL of the houdiniPictureBox image location

this.houdiniPictureBox.ImageLocation = "https://people.rit.edu/dxsigm/Houdini.jpg";

// set the event handler when the mouse enters the PictureBox to call HoudiniPictureBox\_\_MouseEnterLeave

this.houdiniPictureBox.MouseEnter += new EventHandler(HoudiniPictureBox\_\_MouseEnterLeave);

// set the event handler when the mouse leaves the PictureBox to call HoudiniPictureBox\_\_MouseEnterLeave

this.houdiniPictureBox.MouseLeave += new EventHandler(HoudiniPictureBox\_\_MouseEnterLeave);

// set the event handler when the exitButton is clicked to call ExitButton\_\_Click and exit the app

this.exitButton.Click += new EventHandler(ExitButton\_\_Click);

}

// Method: HoudiniPictureBox\_\_MouseEnterLeave

// Purpose: Toggle between showing and hiding the PictureBox upon the mouse entering and leaving

private void HoudiniPictureBox\_\_MouseEnterLeave(object sender, EventArgs e)

{

// the PictureBox control is passed as the "sender" variable

PictureBox pb = (PictureBox)sender;

// negate the current boolean value of the houdiniPictureBox Tag property

pb.Tag = !(bool)pb.Tag;

// set the visible property of the houdiniPictureBox to the current boolean value of the Tag property

pb.Visible = (bool)pb.Tag;

}

// Method: ExitButton\_\_Click

// Purpose: Exit the app when the exit button is clicked

private void ExitButton\_\_Click(object sender, EventArgs e)

{

// exit the application

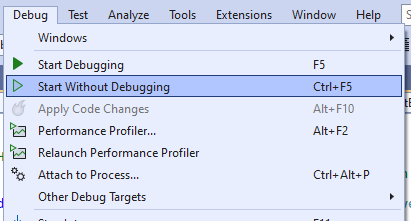
Application.Exit();

}

}

}

13. Select the menu option Debug/Start Without Debugging



When the mouse tries to make contact with Houdini, he hides!

The Exit button exits the app.

1. Submit the GitHub URL of the project folder.

**Important Concepts**

* Setting up the Windows Forms IDE to have easy access to the Solution Explorer, Git Changes, Toolbox and Properties panels
* Configuring the Properties panel to use alphabetical order using the A-Z icon
* Adding controls to the form
* Editing the properties of each control
* Naming conventions for Forms (PascalCase) and their member controls (camelCase)
* We will NOT use the lightning bolt tab of the Properties panel to set event handlers. Instead we will always explicitly code them with the convention of “ControlName\_\_Event” for the delegate method.
* Every class, method and line of code must have a comment
* InitializeComponent() must be the first statement in the Form’s constructor