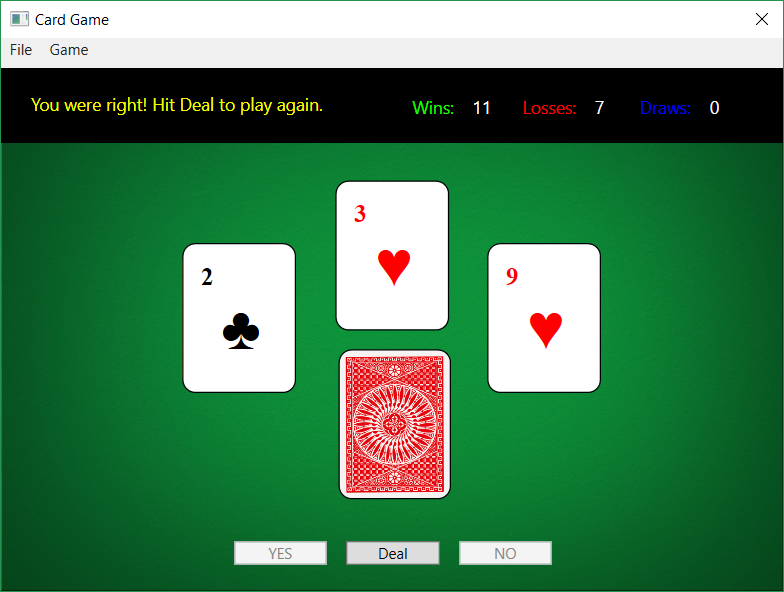
GUI Lab – Windows Presentation Foundation (WPF) and Windows Designer (Part 1)

For our final lab, we will build a GUI-based program to play the Acey Deucey card game.

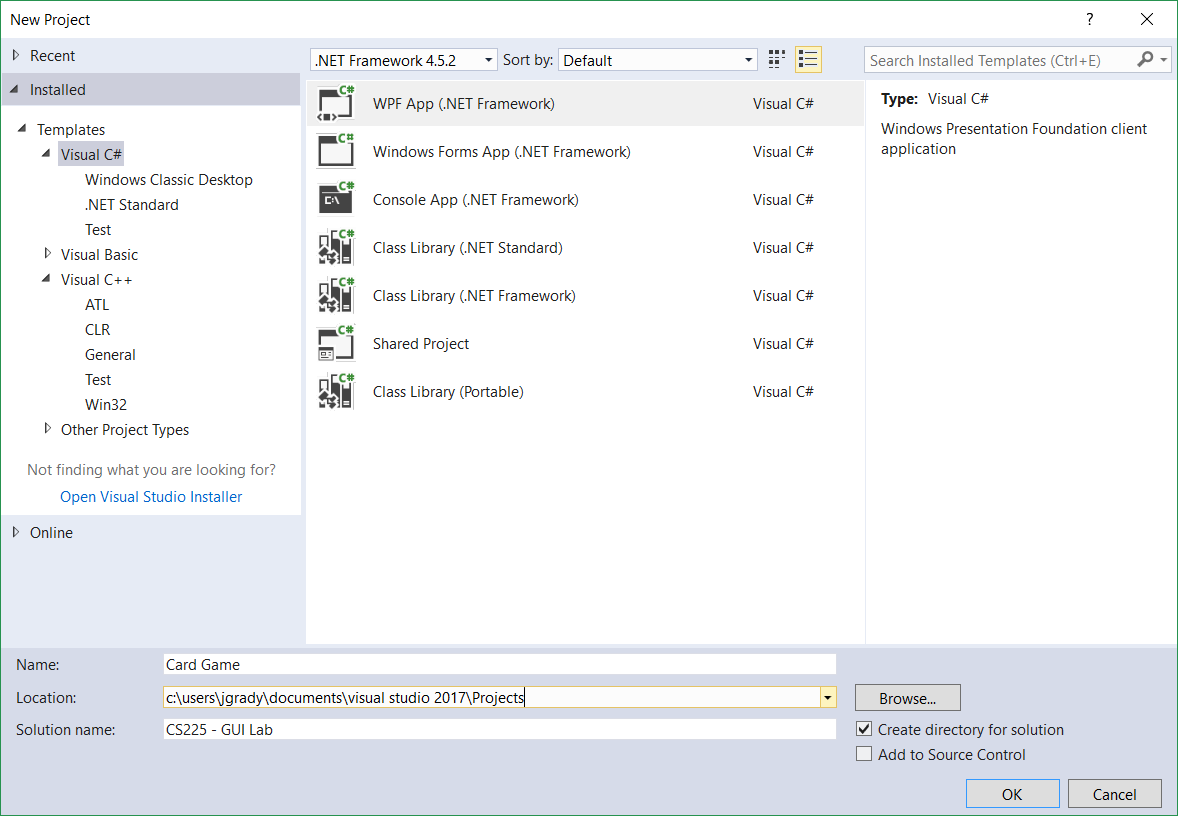


The card game itself is simple:

* Draw two cards from a standard 52-card deck.
  + If the cards are the same rank, or differ by only 1, then the player either loses the hand (normal rules), or draws (nice-nice rules)
  + Otherwise, the player decides if the rank of the next card will be between the two drawn cards or not
    - If they say “Yes”, and the next card it is between the two, they win; otherwise, they lose
    - If they say “No”, and the next card is not between the two, they win; otherwise, they lose
* Continue to draw pairs of cards from the deck until there are fewer than 3 cards remaining. At that point, shuffle the deck of cards and continue with a fresh deck

And while the GUI may not look like much, there’s a lot of code behind the scenes that’s necessary to make the game look and behave the way it does. This lab will walk you through the process of building the game, explaining pieces along the way that you can carve out and use in your own projects.

# 1. Getting Started



Open Visual Studio and create a new Visual C# WPF App project with the following settings:

* Project name: **Card Game**
* Solution name: **CS225** **– GUI Lab**

Next we’ll import the *Card Library* class library we used in Assignment 5, as well as a pair of images for the card deck and application background.

* Download the **Card Library.zip** file from Moodle. Unzip the contents and place the *Card Library* folder inside your *CS225 - GUI Lab* solutionfolder.
* Download the **GUI Lab images.zip** file from Moodle. Unzip the contents and place the *img* folder inside your *CS225 - GUI Lab* solution folder.

If you need help finding your project folder, click on the “**Card Game**” icon in the **Solution Explorer**, then look at the **Properties** panel. The Project Folder property gives you the full path to your project folder.

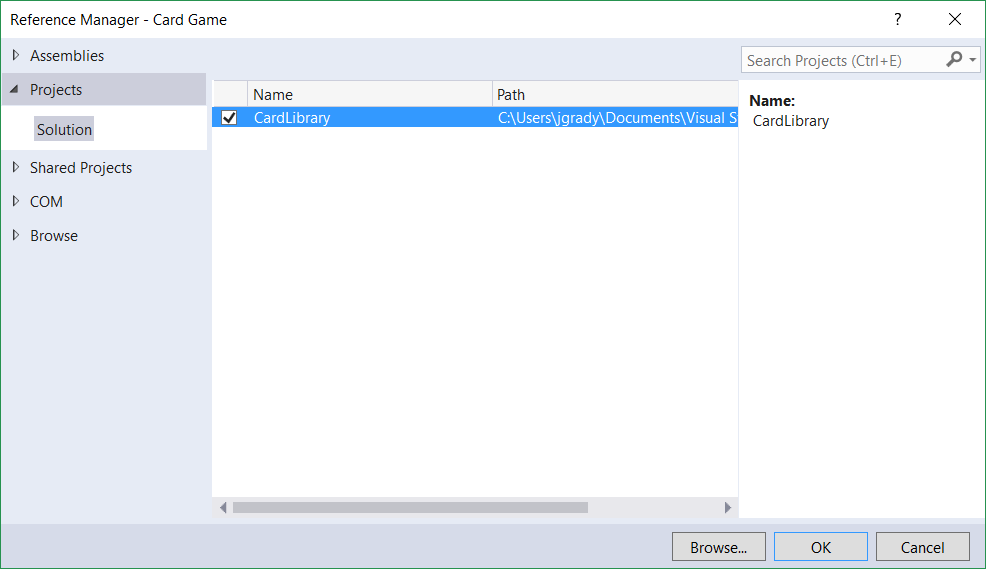
## Adding the Card Library project

In this lab, we’ll use a slightly-modified version of the Card Library project we used in Assignment 5. To do so, we need to add it to our solution:

1. Right-click the “CS225 – GUI Lab” solution icon and choose **Add > Existing Project**.
2. Open the *Card Library* folder
3. Select the *CardLibrary.csproj* project file, then click **Open**

The Card Library project now appears in your solution, but it’s not completely *linked* to your solution.

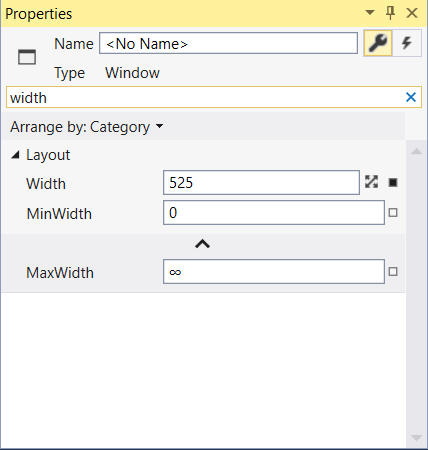
1. Right-click on the “Card Game” project icon; choose **Add > Reference…**
2. The Reference Manager window appears (see the next page for a snapshot). Expand **Solution**, if it is not expanded already.
3. Check the box next to Card Library
4. Hit OK.



## Adding the images to your project

1. Right-click on the “Card Game” project icon again; choose **Add > Existing Item…**
2. Open the *img* folder
3. Change the file filter (it’s the drop-down box above the Add and Cancel buttons) from *Visual C# Files* to *Image Files***.**
4. Hold down the CTRL key and select the 3 images inside the folder.
5. **PAY ATTENTION**. Click the drop-down arrow next to the **Add** button. Choose **Add as Link**. (If you simply click Add, Visual Studio makes copies of the images and places them loose inside our project folder. This is inefficient and could potentially cause errors.)

# 2. Creating the Layout

When you create a WPF project, Visual Studio automatically creates an empty Window object in a *MainWindow.xaml* file. We’ll start by changing the Window’s properties

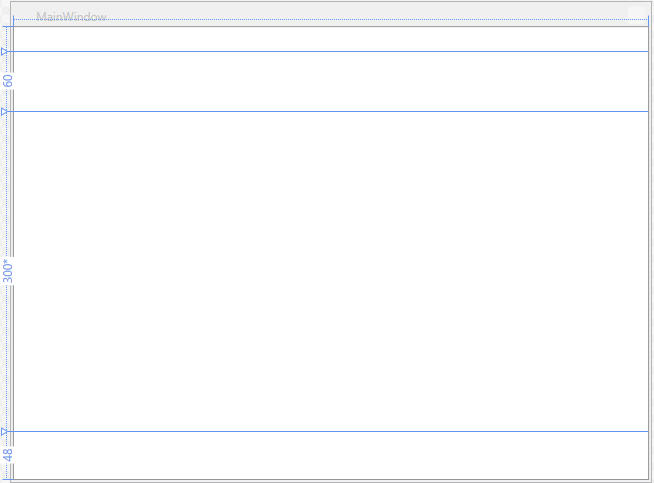
* In Designer, click on the Window object’s title bar.
* The **Properties** panel should be visible in the bottom-right corner screen. *If not*, hit the **F4** key or select **View** menu > **Properties Window**
  + In the **Search Properties** box, type “width”. You’ll get a filtered list of Window properties that contain your search string. The Search Properties box makes finding properties a lot easier than scrolling through the entire list!
  + Change the value of the **Width** property to *640*
  + Find the **Height** property, and change its value to *480*
  + Find the **ResizeMode** property, and change its value to *NoResize*. As the name implies, this prevents the user from resizing the window.
  + Find the **Title** property, and change it to *Card Game*.

Next, we’ll set up the window’s Grid with four (4) rows. We’ll start by adding the rows in Designer view, but then updating their heights in XAML view.

* In Designer, click anywhere inside the body of the Window object to select its Grid.
* Scroll over the top left-edge of the Grid box until you see a yellow horizontal line across the grid. Position the line as shown at the top of the next page.
* Click to create a new Grid row.



* Add three more rows as shown below. Don’t worry about getting the rows perfectly aligned. We’ll adjust the heights in the next step.



Before we go into XAML view to adjust the heights, let’s get to know our Visual Studio interface a little better. Between the Design and XAML panes are the controls for the two:



Hitting the **Design** and **XAML** buttons switches the *focus* of the window to one or the other. Hitting the double-arrow **Swap Panes** button swaps the position of the two panes top-or-bottom.

* Hit the Swap Panes button as needed to put the XAML in the top pane.
* In the XAML code, you’ll see four <RowDefinition> elements, one for each row you made inside the Grid. Change the Height properties of each RowDefinition to match the following:

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="24"/>

<RowDefinition Height="60"/>

<RowDefinition Height="300\*"/>

<RowDefinition Height="48"/>

</Grid.RowDefinitions>

</Grid>

The asterisk after “300” in the third RowDefinition means that row has a preferred Height of 300 pixels, but grows or shrinks to fill in any available space

* Hit the Swap Panes button to put the Designer back into the top pane.

## Adding containers to the Grid rows

First, we’ll add an empty Menu bar to the top row. We’ll add menu items to the Menu bar later.

* Go to the **Toolbox** panel – and find the Menu control.
* Click and drag the Menu control onto the top row of the Grid.
* Using the Menu’s resize handles, change the size of the Menu so that it completely fills the top row of the Grid. Visual Studio will help you by snapping the edges of the Menu to the edges of the Grid.
* Finally, click on the small chain links on the right and bottom edges of the Menu bar to lock it to the Grid.
* **OPTIONAL**: In XAML view, locate the <Menu> element and add a Grid.Row *attached property*:

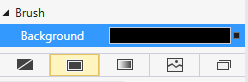
<Menu Grid.Row="0" Height="24"/>

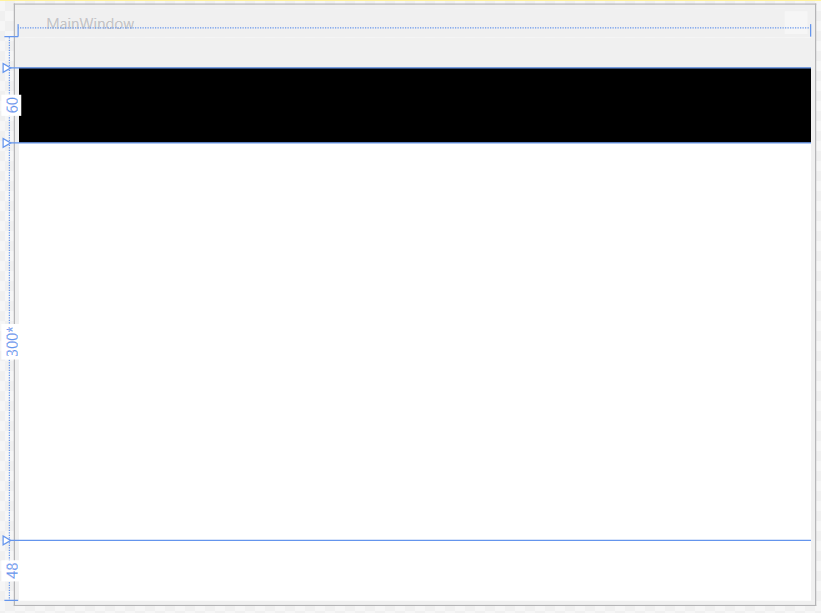
This formally links the Menu to the first row of the Grid. You’d think Designer would have done that for you when you resized the menu, but, no…it did not.

Next, we’ll add three Panel objects to the Window. Panels will help us organize all the components that we’ll add throughout the lab:

* In the **Toolbox** panel, find the Canvas control. Click and drag the Canvas control onto the second row of the Grid
  + Resize the Canvas so it fills the entire second row of the Grid
  + Click on the small chain links on the right and bottom edges of the Menu bar to lock it to the Grid.
* Repeat the same steps to add a Canvas control to the third row of the Grid
* Finally, go back to the **Toolbox** panel and find the WrapPanel control. Click and drag the WrapPanel inside the bottom row of the Grid.
* Change these **Properties**:
  + Name: buttonPanel
  + Width: 290
  + Height: 48
  + Margin
    - Left: 171
    - Right: 171
    - Top: 0

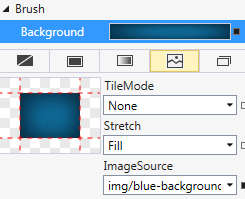
One last thing: let’s change the color of the Canvas in the second row.

* Click inside the second row of the Window to select its Canvas
* In the **Properties Panel**, make change the Background color to black using the **Solid Color Brush**



# 3. Adding a Background Image

In this part, we’ll add a background image to the window

* In the **Document Outline** panel, select the Window. If your Document Outline panel isn’t visible, hit **Ctrl+Alt+D**, or select **View** menu > **Other Windows** > **Document Outline Window**.
* In the **Properties Panel**…
  + Set the Background to use the **Tile Brush**
  + Click the ImageSource drop-down menu; choose either the blue background (*blue-background.png*) or the green one (*green-background.png*).

# 4. Adding Controls to the GUI

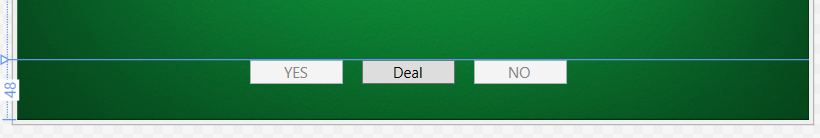
In this section, we’ll add labels to the second row; a pair of rounded rectangles to the third row as placeholders for the dealt cards; and three buttons to the fourth row.

## Adding the status and score labels

* In the **Toolbox** panel, find the Label control and drag it onto the second Grid row.
* Make these changes to its **Properties**
  + Name: statusLabel
  + Width: 300
  + Height: 36
  + Content: Hit “Deal” to Play
  + Foreground: Solid Color Brush: yellow (choose a color from the color picker)
  + Text Size: 14
* **YOUR TURN:** Add the “Wins”, “Losses”, and “Draws” labels. You don’t need to set specific names, widths, or heights for these three labels. Their Text Sizes should be 14, and their Foreground colors to green, red, and cyan (sky blue) respectively.
* Add the three “0” labels as shown on the previous page.
  + All three should have a Width of 36
  + Set the Name property of each one to the following:
    - First label’s Name: winsCountLabel
    - Second label’s Name: lossesCountLabel
    - Third label’s Name: drawsCountLabel

## Adding the game control buttons

Next, we’ll add the three game control buttons to the bottom row of the Grid.



* In the **Toolbox** panel, find the Button control and drag it onto the WrapPanel inside the last Grid row.
* Set these **Properties** of the Button
  + Name: yesButton
  + Content: YES
  + IsEnabled: false (uncheck the box)
  + Margin
    - Left: 15
* Add the second button to the last Grid row and set the following **Properties**
  + Name: dealButton
  + Content: Deal
  + Margin
    - Left: 15
* Add the third button to the last Grid row and set the following **Properties**
  + Name: noButton
  + Content: NO
  + IsEnabled: false (uncheck the box)
  + Margin
    - Left: 15

## Adding the card position placeholders

We’re almost done adding controls to the game window. Our next step is to add three Rectangle controls to the window for the card position placeholders:



We’ll start with the left placeholder, then the right, ending with the middle placeholder that represents the deck of undealt cards.

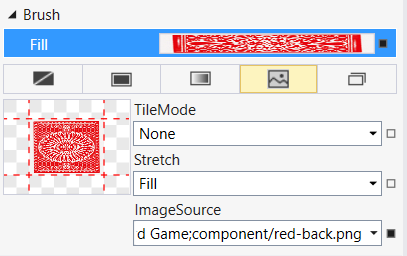
* In the **Toolbox** panel, find the Rectangle control and drag it onto the third Grid row.
* Set these **Properties** of the Rectangle
  + Width: 90
  + Height: 120
  + Canvas.Left: 145 (**HINT**: this property appears as “Left” under the “Layout” section. Typing “canvas” in the search bar takes you to this property)
  + Canvas.Top: 80
  + Fill: No Brush
  + Stroke (color): #FFE6DD30
  + RadiusX: 10
  + RadiusY: 10

As you could tell, RadiusX and RadiusY properties round off the corners of the Rectangle.

* Add the right-hand Rectangle and set these **Properties**:
  + Width: 90
  + Height: 120
  + Canvas.Left: 390
  + Canvas.Top: 80

Note that Visual Studio applied the same Fill, Stroke, RadiusX, and RadiusY settings to the new Rectangle, saving us some time.

* Add the middle Rectangle and set these **Properties**:
  + Width: 90
  + Height: 120
  + Canvas.Left: 270
  + Canvas.Top: 165
  + Stroke (color): #FF000000 (black)
  + Fill: (see figure below)
    - Brush: Tile Brush
    - Stretch: Fill
    - ImageSource: red-back.png



**YOUR TURN**: This is a good time to Save your work. Hit **Ctrl+Shift+S**, do **File** menu > **Save All**, or hit the  Save All button in the main toolbar.

# 5. Using ContentControl and Templates

The next phase of the project is to make the graphic of a dealt playing card. Each time the player clicks the “Deal” button to start a new hand, we’ll place two new cards (i.e. Rectangle controls) on the screen. We also plan on using animations, as though we deal cards straight from the deck.

Anytime you have a game that spawns many similar objects of some indeterminate quantity – cards, enemies in a shoot-em-up game, etc. - a smart design choice is to create a *Template* of that object. Think of a Template as a custom component where you group multiple WPF controls together to form a single component. You can then apply that Template to a ContentControl object, which is the abstract parent class of all WPF component controls (Label, TextBox, etc.)

## Adding a ContentControl

First, we’ll add the empty ContentControl to our window:

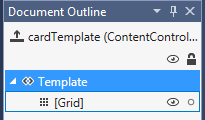
* In the **Toolbox** panel, find the ContentControl control and drag it onto the third Grid row. Place it directly on top of the middle Rectangle that represents the deck of cards
* Set its **Name** property to cardContentControl
* Resize the ContentControl to match the size of the Rectangle beneath it

## Creating a Template

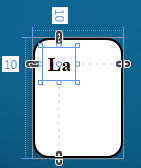
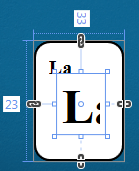
Next, we’ll create a new Template and apply it to our cardContentControl

* Right-click on the ContentControl and choose **Edit Template > Create Empty…**
* Rename the template “cardTemplate”
* **IMPORTANT**: Under **Define In**, click the **This Document** dropdown menu and choose Window: <no name>
* Hit **OK**

We’re now back at the main Designer window. If you take a look at the Document Outline you’ll notice we’re now inside the Template portion of our GUI hierarchy:

* To return to the main document, hit the  up arrow to the left of the template name.
* To get back into the cardTemplate Template, right-click on the ContentControl and choose **Edit Template > Edit Current**

Enough housekeeping - let’s create a generic play card by editing the cardTemplate Template:

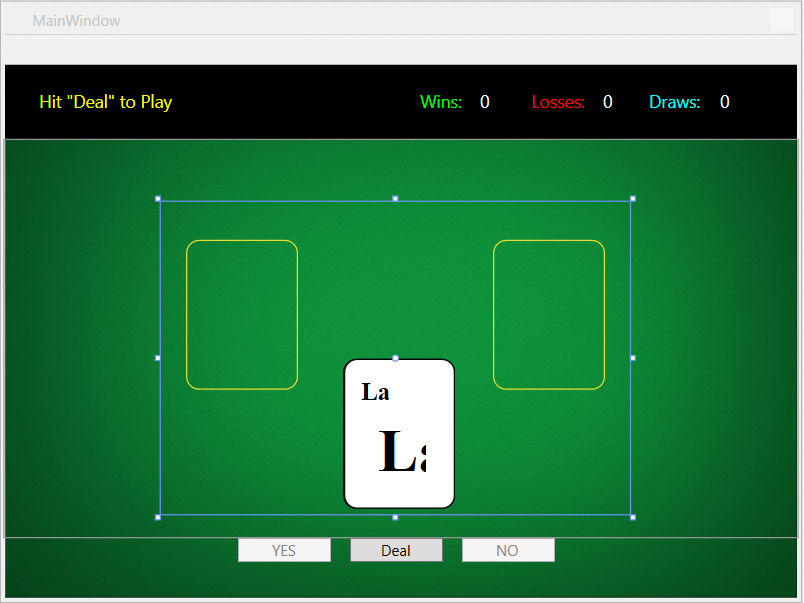
* In your game’s window, select the middle Rectangle (i.e., the deck of cards)
* In the **Toolbox** panel, find the Rectangle control and drag it onto the ContentControl
* Reposition and resize Rectangle so that it matches the Rectangle that represents the deck of cards
* Change the Fill property to a **Solid Color Brush**, white (#FFFFFFFF)
* In the **Toolbox** panel, find the Label control and drag it onto the top left corner of the new Rectangle.
* Set these **Properties** of the Label:
  + Name: rankLabel
  + Width: 32
  + Height: 32
  + Text: Times New Roman, 20px, and Bolded
* Add another label to the ContentControl for the suit, setting these **Properties**
  + Name: suitLabel
  + Width: 48
  + Height: 60
  + Text: Times New Roman, 48px, and Bolded
  + HoriztonalContentAlignment: Center

That should do it for the card template. Click the up arrow in the **Document Outline** panel to get back to the main document.

## Adding an animation staging area

Finally, we’ll add a new Canvas control, which acts as the staging area for the playing card animations.

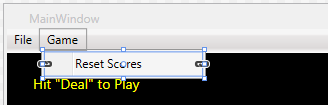
* In the **Toolbox** panel, find the Canvas control and drag it onto the third Grid row.
* Set these **Properties** of the Canvas:
  + Name: playArea
  + Width: 375
  + Height: 250
  + Left: 125
  + Top: 50



This staging area is an important cog in the game. Not only does it give us a “sandbox” for adding and removing playing cards from the GUI, but it will also help us define the starting and stopping points of the card animations later down the line.

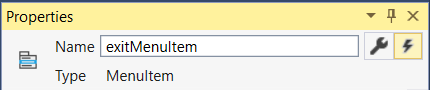
# 6. Adding Menu Items to a Menu

We’ve already added a menu bar to the GUI, but not the menus. We’ll add two menus – “File” and “Game” - each with a single menu item that performs an action.



* To add a menu item…
  + Click on the empty Menu bar in your GUI
  + Right-click on the Menu bar and choose **Add MenuItem**
  + Resize the new MenuItem so that it’s the same height as the Menu bar
  + Right-click on the new MenuItem and choose **Edit Text**
  + Type “File”, then click anywhere outside of the Menu
  + In the **Properties** panel, change the following:
    - Name: fileMenuItem
    - Width: Auto (**HINT**: hit the  button next to the Width text field)
    - Height: Auto
  + Repeat the same steps to add a second MenuItem:
    - Change its text to “Game”
    - Change its Name to “gameMenuItem”
    - Change its Width and Height to “Auto”
* To add choices to a menu item…
  + First, click on the “File” menu in your GUI.
  + Right-click the menu item and choose **Add MenuItem**.
  + In the **Properties** panel…
    - Change the Name of the MenuItem to “exitMenuItem”
    - Change the Header of the MenuItem to “Exit”
    - Change its Width and Height to “Auto”
  + Repeat the same steps to add a MenuItem under the “Game” menu
    - Change its Name to “resetMenuItem”
    - Change its Header to “Reset Scores”
    - Change its Width and Height to “Auto”

Those new menu items won’t do much unless they subscribe to *event handlers*. Windows Designer makes working with existing menus very difficult, so we’ll have to select the “Exit” and “Reset Scores” menu items through the **Document Outline** panel

* In the **Document Outline**, locate and click on the “exitMenuItem” menu item in the Menu.
* In the **Properties** panel…
  + Click the **Event Handler (lightning bolt)** button
  + Double-click in the box next to the Click event to create an empty event handler.

Visual Studio switches you over to the code-behind file. Notice that Visual Studio automatically adds skeletal code for the event handler, exitMenuItem\_Click.

When the user chooses this menu item, the GUI application should exit. Add this line of code to the body of the event handler:

Application.Current.Shutdown();

* + Close the *MainWindow.xaml.cs* code-behind file
  + **YOUR TURN**: Following the same steps as above, add an event handler to the “Reset Scores” menu item.

We won’t add any code to the event handler at this point. Later in the lab, we’ll add code that resets the value of the wins, losses, and draws to 0.

**YOUR TURN**: Once again, this a good time to Save your work. Hit **Ctrl+Shift+S**, do **File** menu > **Save All**, or hit the  Save All button in the main toolbar.

**YOUR TURN**: This is also a good time to compile and run your code.

* Go to the **Build** menu and choose **Build Solution**.
* Go to the **Debug** menu and choose **Start Without Debugging**.

**Congratulations!** You’re finished with Part 1. You’ve added all the game’s GUI components, including a handful of menu items. If all went well, your game should look like the screenshot below.

In **Part 2**, we’ll work on the data model and logic of the game.

