Lab 2 Due next lab period – week of April 25-28

Purpose:

1. Create a simple Windows GUI application using Visual Studio and VB.NET
2. Learn UI control event handling
3. Practise VB.NET programming

What to submit:

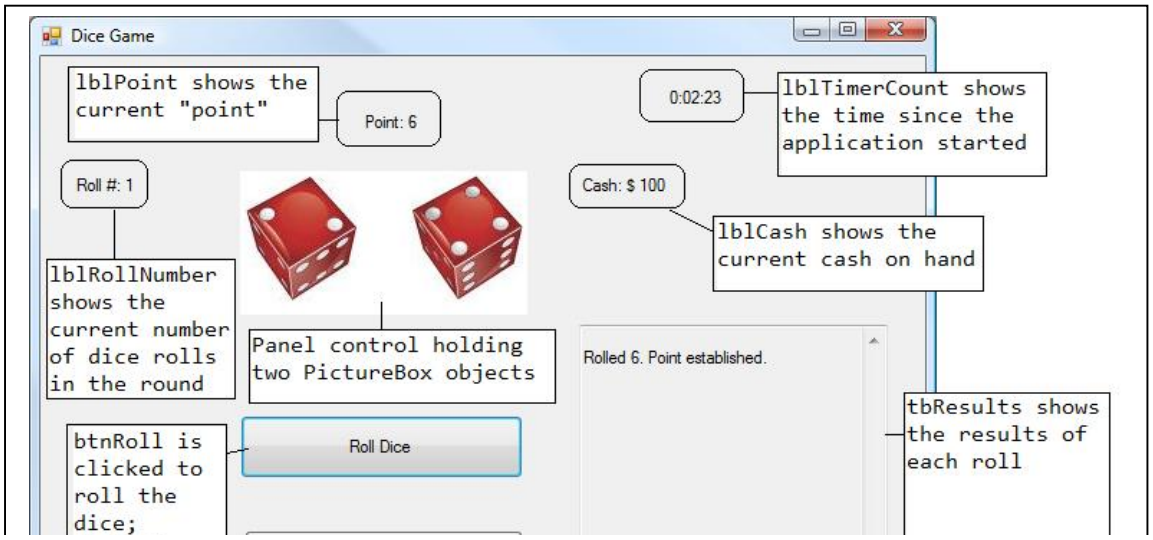
1. At the start of your next lab period, be prepared to demo your lab, and answer 1 or 2 questions
2. Submit your zipped project to D2L before your lab period.

Evaluation (24 marks):

1. Be prepared to demonstrate your solution at the start of next lab. If you are not ready at the start of lab or are not present, a mark of 0 will be given.
2. Your solution will be assessed based on (3 marks each):
    - proper form layout of the controls (no overlapping controls, sufficient white space)
    - coding the roll button to display random dice images
    - coding the results TextBox text display with information each roll
    - coding the labels for the point, roll number and cash
    - coding the Timer and update label correctly
    - coding the quit button to display message
    - VB.NET code style – variables named appropriately, comments used in the code where needed, code is readable and understandable.
    - Correctly answering questions about your solution.
    - 1 bonus mark – adding a control for the bet amount and coding it in the application. By default, the application uses a fixed $10 as the amount which can be won or lost. A new control would allow the player to select the bet amount only at the start of any round. This would be the amount that would be won or lost for that round.
3. **All work must be individual.** Any plagiarism will result in a mark of 0 and possibly additional penalties.

Description

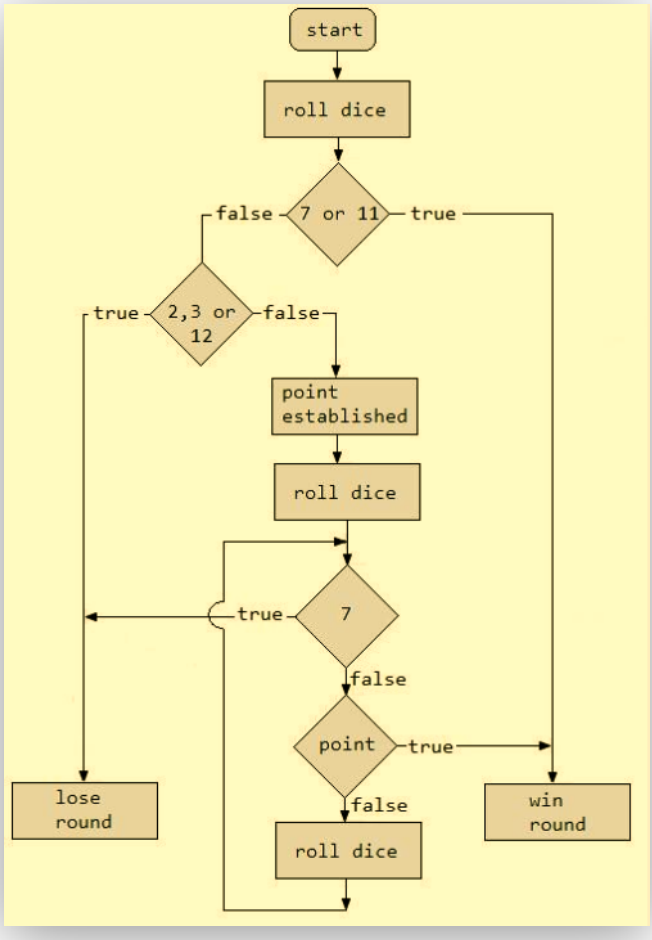
For this 'craps' lab you will build a Windows application similar to the following:



This Windows form application simulates a simplified version of a common wagering dice game called “craps”. In this application the dice images show the results of ‘rolling’ two six sided dice when the button “Roll Dice” is clicked. The player (aka ‘user’) begins with $100 cash, shown in the Form’s `lblCash` Label control. The player’s goal is to roll one of the “point” numbers: 4, 5, 6, 8, 9, or 10 (the sum of the two dice), then roll it again before a seven is rolled. If the point number is rerolled, the player wins the round and gains a bet amount of $10. However, if a seven is rolled before the point number, the player loses the round and loses $10 from the cash amount. The player can click the “Cash out” button to end the application and see how much cash remains. The player can continue to play new rounds so long as the cash position is more than the bet amount of $10. You cannot borrow money in this application.

When the player starts a round, a roll of 2, 3 or 12 before the point is established means the player has automatically lost the round. A roll of 7 or 11 before a point is established is called “natural” and the player wins the round.

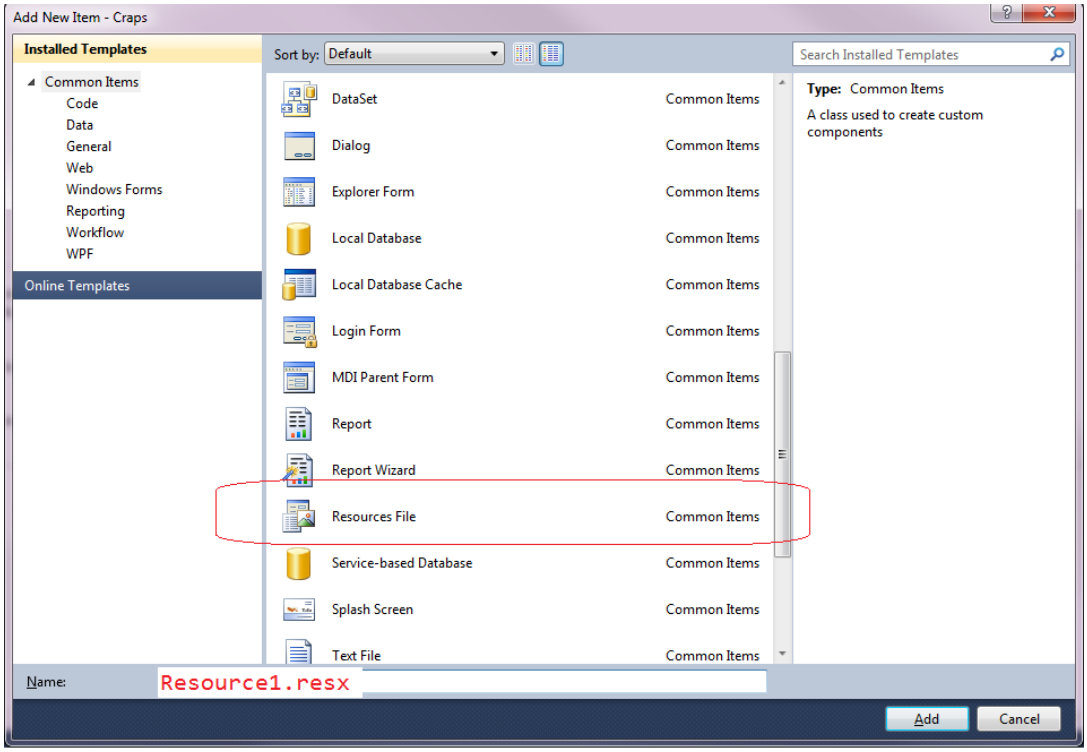
Once a point has been established, any roll but that point (or 7 which ends the round) is considered a pass and the player can reroll. A flowchart describing the roll mechanics is shown below. Recall that the rectangle shapes denote a process and diamond shapes denote a decision.

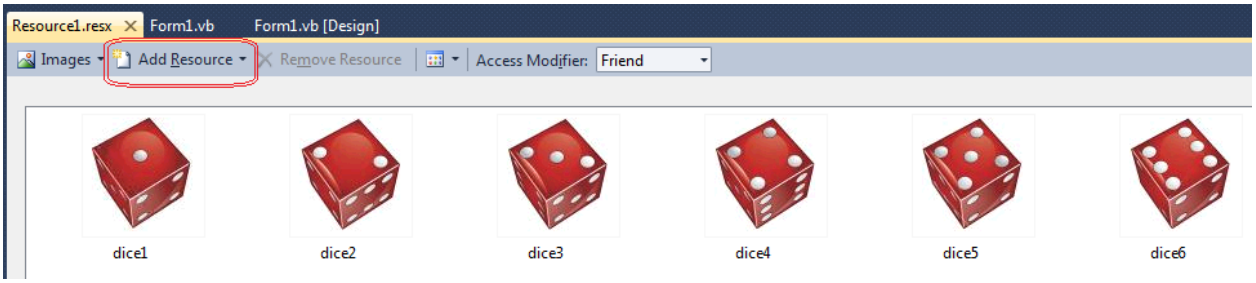


Work out the algorithm carefully in your VB code. Make sure any loops and conditions are correctly defined.

Steps

1. Create a new project.
    a. From the list of installed templates select project type Visual Basic Windows Forms Application with the .NET Framework and name it Lab2.
2. Preliminaries:
    a. Adding Images to project:
        i. The images for the dice sides and the sample executable can be found within a zip file on D2L.
        ii. Download the zip file and extract the images to a folder.
        iii. To load the images into a Resources component for your Windows form project, activate the Solution Explorer window and right click on the project name – the second from the topmost entry – it will be just above the My Project entry. Select the menu option for Add and then select `Component...`. A list of Visual Studio templates will appear in a window. Scroll down the list and select the one labelled Resources File. The name will be shown as Resource1.resx. Click Add to add the component to your solution. Select the resource file in the solution explorer window and double click. The Resource tab should appear in the main window. Find the Add `Resource` menu option at the top of the tab and click the down arrow to the right. Select the "Add Existing File" menu option. Find the first dice side image and add it to the resource file. You will have to do this five more times for each of the dice images.





iv. Once the images your application needs are all in the Resources file, you can reference them easily in VB.NET using the notation `My.Resources.Resource1.`*name* where *name* is the resource name you gave to the image.

v. For example, to assign the "one dot" dice image to a `PictureBox` control named `PictureBox1`, you would use the VB.NET statement

```
PictureBox1.Image = My.Resources.Resource1.dice1
```

vi. There is no need to use Windows pathnames or network names if you use Resource files.

3. Setup the form

a. As there are a few more operational parts to this lab than in the earlier one, you will have to take care to make this process work correctly. If you need help to organize the development of this application, you can use this guide.

b. Design the form layout with all the required controls in place and appropriately named:

i. `Panel` control with a white background
ii. Two `PictureBox` controls inside the `Panel`
iii. Two `button` controls
iv. Multiline readonly `TextBox` control with a vertical scrollbar
v. Label for the roll number
vi. Label for the point
vii. Label for the cash
viii. Label for the clock
ix. `Timer` control

1. For the time display you need to drag a `Timer` control onto the Form in the Design mode. Note that the `Timer` control cannot be seen on the form so in the Design, the control is pushed outside the form. In the Form's `Load` event invoke the `Timer's Start` method to start the timer.

c. Add the images to the Project's Resources

d. In the Form code declare a private class scope constant NUM_SIDES = 6. This represents the number of dice sides.

e. Make sure to layout your form elements in a way that adheres to principles of good design, using appropriate:

i. White space
ii. Alignment
iii. Grouping

4. Add the functionality:

a. Define a function `RollDie()` which returns a random number between 1 and NUM_SIDES+1. Make a class scope variable `rndRoll` of type Random and use it in your function.

```
Private rndRoll as New Random()
```

a. Add class scope variables:

i. Integer variables to keep track of various amounts:

1. intCash, intBetAmount, intRollCounter, intPoint, intSum

ii. Boolean variable to keep track of win/lose states:

1. blnLostRound, blnWonRound, blnPointSet

b. Write the Form's Load event
   i. set the labels for cash, point and roll number.
   ii. You will need to keep track of the current cash, so initialize `intCash` to 100.
   iii. The amount that is won or lost is 10, initialize `intBetAmount` to 10.
   iv. The number of rolls needs to be tracked, initialize `intRollCounter` to zero.
   v. The point number for the round is initialized as `intPoint` = 0.
   vi. We can use flags to tell the application when a condition such as a round won or lost has happened. Initialize `blnLostRound` and `blnWonRound` to false.
   vii. Initialize `blnPointSet` to false indicating that the player has yet to set a point number in the current round.
   viii. Note that these initializations in the Form `Load` are performed only once when the application starts. Some of these variables will need to be reset if the player wants to continue on to a new round (e.g. reset the flags, the point and the roll counter). The time label needs to have a starting time reference, so initialize a class scope variable `datStartTime` to `Date.Now`, which represents the time at that second.

c. Add a tick event to the timer:
   i. The Timer's `Tick` event should fire every second so change its `Interval` property accordingly in the `Property` window. The code for displaying the clock label can make use of the `TimeSpan` object as well as `String.Format` function.

```
Dim timePlayed As TimeSpan = datNow.Subtract(datStartTime)

lblTimerCount.Text = String.Format("{0}:{1:d2}:{2:d2}", _
                                   timePlayed.Hours, _
                                   timePlayed.Minutes, _
                                   timePlayed.Seconds)
```

d. Add a click event for the Roll button:
   i. The `click` event for the `Roll` button will contain the bulk of the application logic as shown in the flowchart. You will need to figure out what the application should do when either of the `blnWonRound` or `blnLostRound` flags is set. Take your time planning this part of the development.
   For the `TextBox` control you can add new text on new lines using the VB.NET constant vbCRLF (or vbNewLine) as in

```
txtResults.Text += vbCrLf + "Rolled " + intSum.ToString + _
                   ". Point established."
```

e. Add code to force the textbox to scroll up automatically:
   i. There is a method for forcing the `TextBox` text to scroll up automatically. You append a blank to its text, then use the ScrollToCaret method to move the cursor the end, then reset the focus back to the previous control. This code can go in the Roll button's click event handler.

```
' Make the TextBox text scroll up
Dim ctr As Control = Me.ActiveControl
txtResults.Focus()
```

```
txtResults.AppendText("")
txtResults.ScrollToCaret()
ctr.Focus()
```

f. Add a dialog box to ask the user if they want to play again:
   i. The `MessageBox` class can be used to accept user input and you can check its value against a `DialogResult` object. The first parameter in the `Show` method is the text inside message box, the second parameter is the box title, the third parameter is what kind of response you require from the user (a Yes/No or other types of buttons), the fourth parameter is the style of icon shown in the message box.

```
If MessageBox.Show("Do you wish to play again?", _
                   "Round over", _
                   MessageBoxButtons.YesNo, _
                   MessageBoxIcon.Question) _
          = Windows.Forms.DialogResult.Yes Then
```

g. Add a click event to the Quit button:
   i. The Quit button uses `MessageBox` to display the amount of remaining cash, then issues the `Application.Exit` statement.

A version of the executable will be made available to you for inspection on D2L in the dice.zip folder.

PictureBox: http://msdn.microsoft.com/en-us/library/system.windows.forms.picturebox.aspx

Timer: http://msdn.microsoft.com/en-us/library/system.timers.timer.aspx

Debugging

How to set a simple breakpoint http://msdn.microsoft.com/en-us/library/k80ex6de.aspx

Optional

The dice images appear just after you click the Roll Dice button. It is possible to employ a second Timer control in the form to create the illusion that the dice is "rolling" by making random dice images appear quickly then gradually slowly until they stop. The `Timer` control's `Interval` property can be modified within the control's `Tick` event. A counter variable would help count the number of `Tick` events that have fired.

Also, you can make a button that allows the user to immediately stop the dice rolling in this optional change (basically a 'cheat mode' where you stop the dice when you see your point appearing).