Limnor Studio – User's Guide

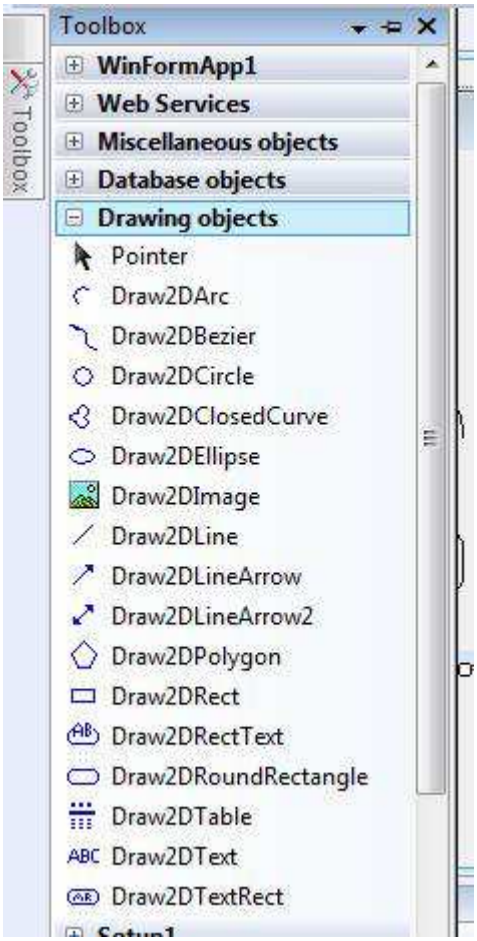
Part – VII

2D Drawing

Contents

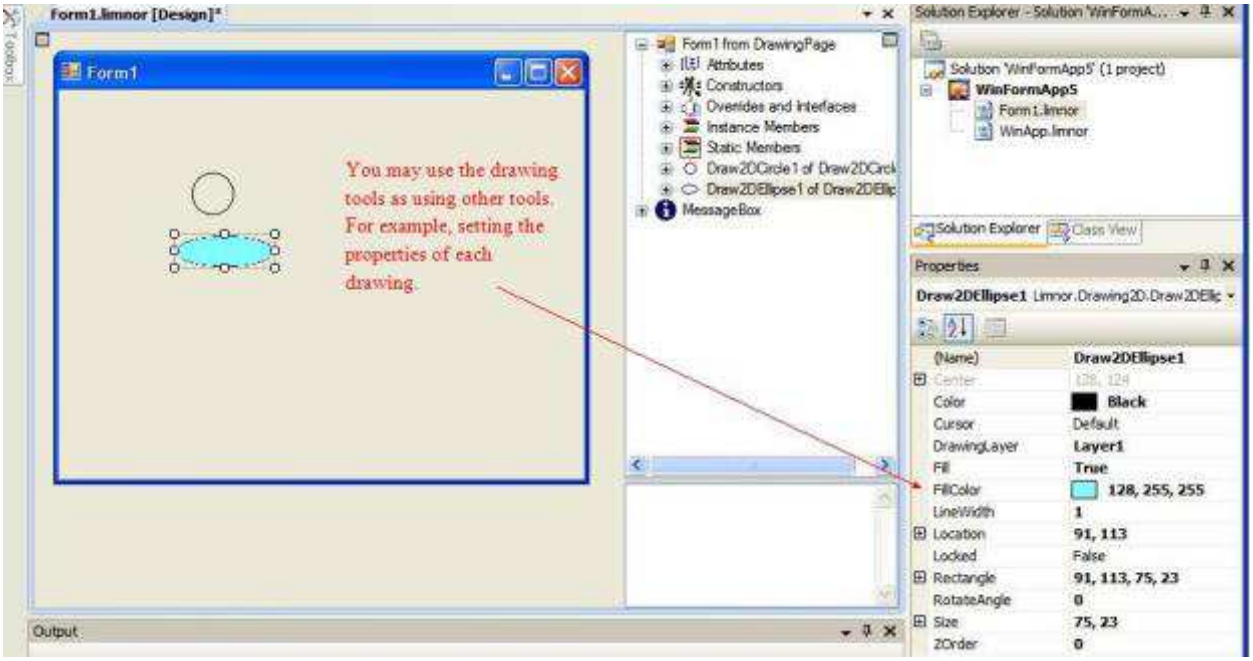
1 Drawing Tools

Drawing tools are in the Toolbox under Drawing Objects tab:



The drawing tools look like other controls at design time. They are controls at design time for doing design like other controls. At runtime they are not controls. They are displayed by drawing on the graphics surface at runtime. Thus at runtime they consume very little system resources comparing to controls.

You may use the drawing tools as using other tools, for example, setting the properties of each drawing.



This document describes basic usages of drawings. For additional information, see following documents.

Use drawing objects to build What You See Is What You Print data entry form - http://www.limnor.com/support/DataEntryWithDrawings.pdf

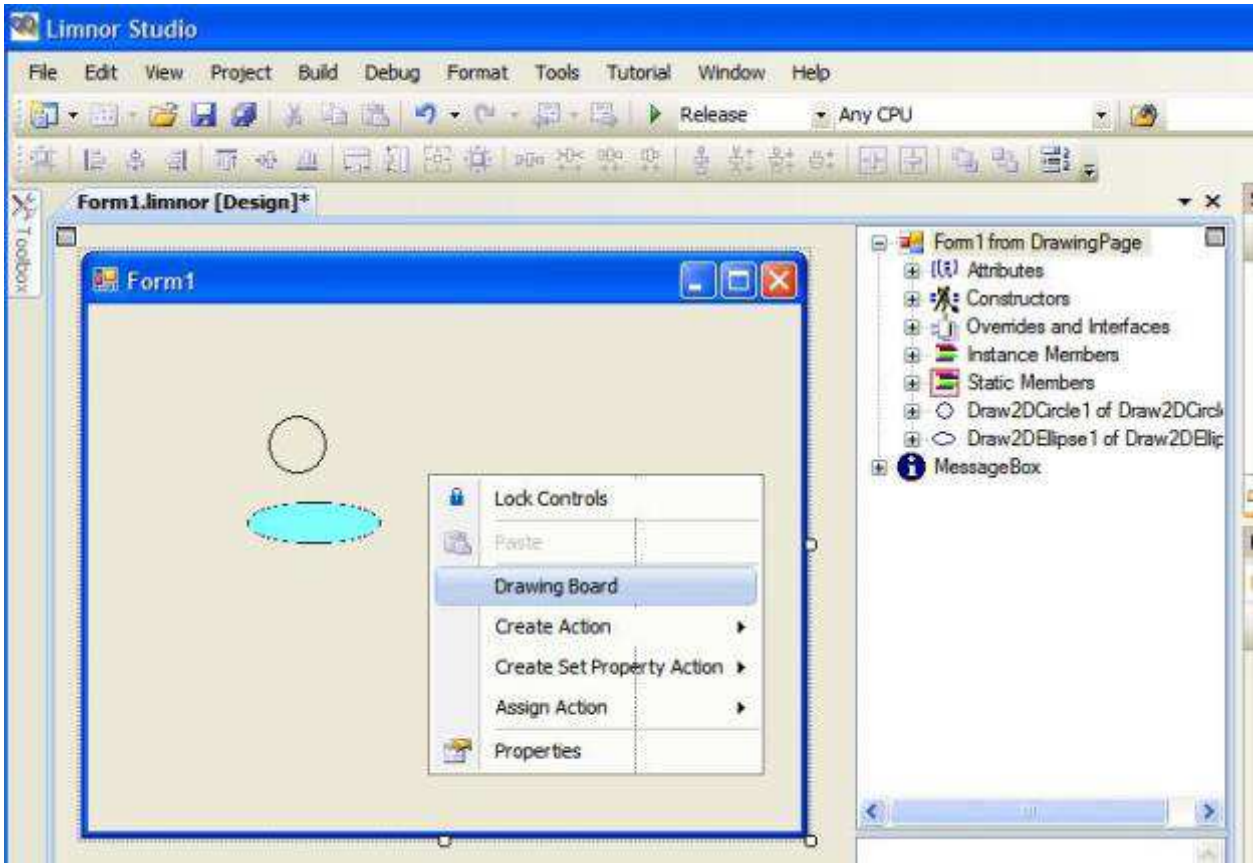
Use drawing groups and drawing repeaters - http://www.limnor.com/support/DrawGroupRepeater.pdf

2 Drawing Board

2.1 Launch Drawing Board

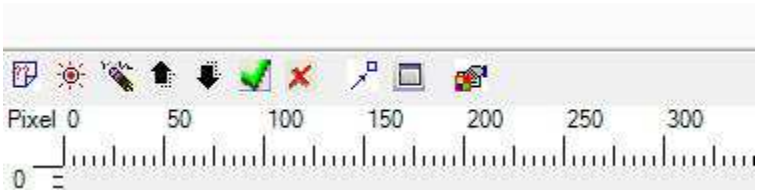
Drawing Board is a graphic tool for editing drawings. It provides more editing capabilities than editing drawings as controls.

To use the Drawing Board, right-click on a form, choose “Drawing Board”:

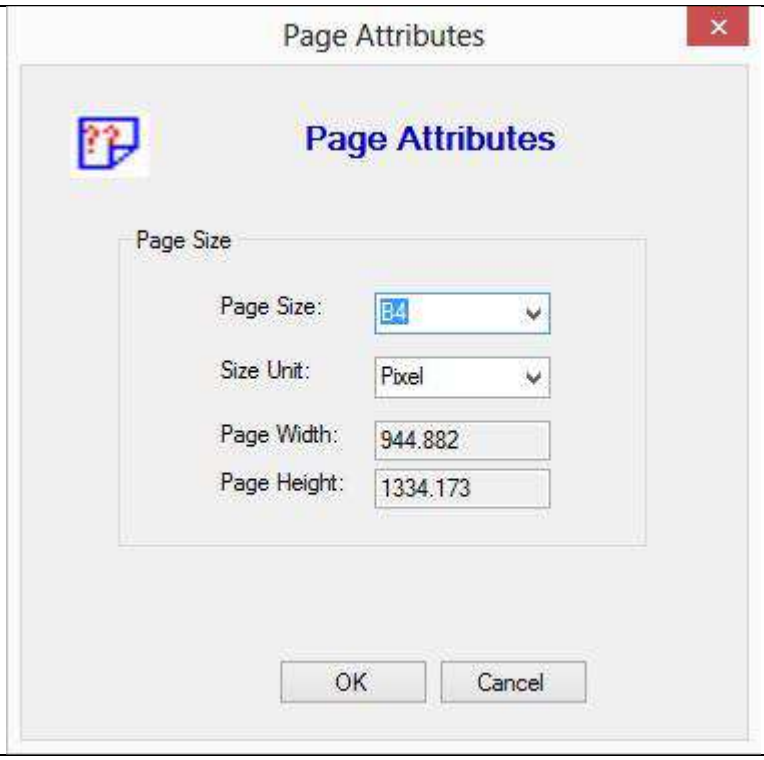


2.2 Toolbar buttons

Toolbar buttons are on top of the drawing board:



	It shows page attributes.

	
	It shows/hides drawing types
	It deletes selected drawing item
	It brings selected drawing item to front
	It sends selected drawing item to back
	It finishes editing the drawings
	It cancels editing the drawings
	It moves drawing-types windows and drawing properties windows to top right position.
	It un-maximize and maximize the drawing board.
	It remembers properties of selected drawing item. A new drawing item of the same type will use remembered properties as default properties.

2.3 Manipulate Drawings

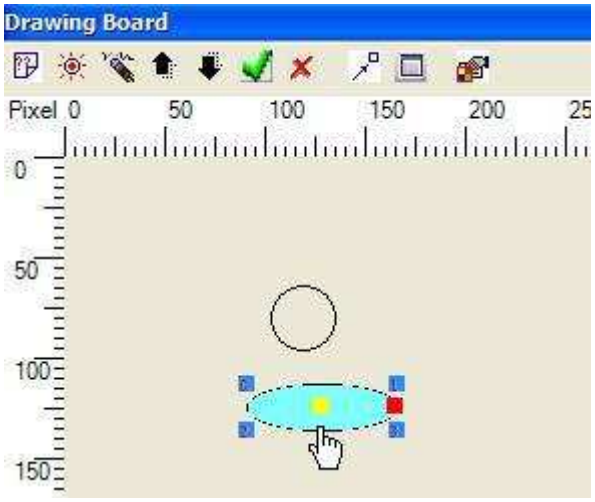
To select a drawing, move mouse over the drawing, the mouse pointer will become a hand indicating it detected the drawing under it. Click the mouse to select it.

When a drawing is selected, it is marked with some blue marks, one yellow mark and a red mark.

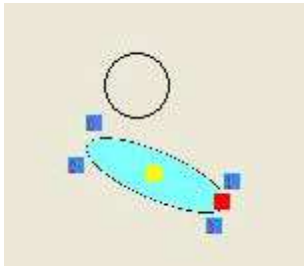
Drag the blue marks ■ to resize the drawing.

Drag the yellow mark ■ to move the drawing.

Drag the red mark ■ to rotate the drawing.

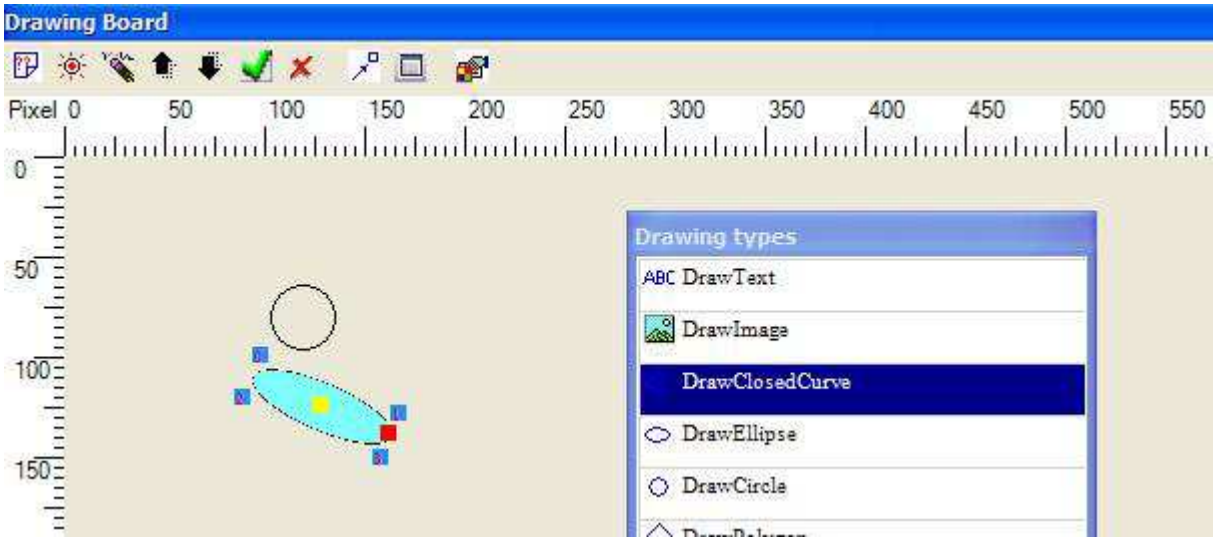


Drag the red mark to rotate the drawing:

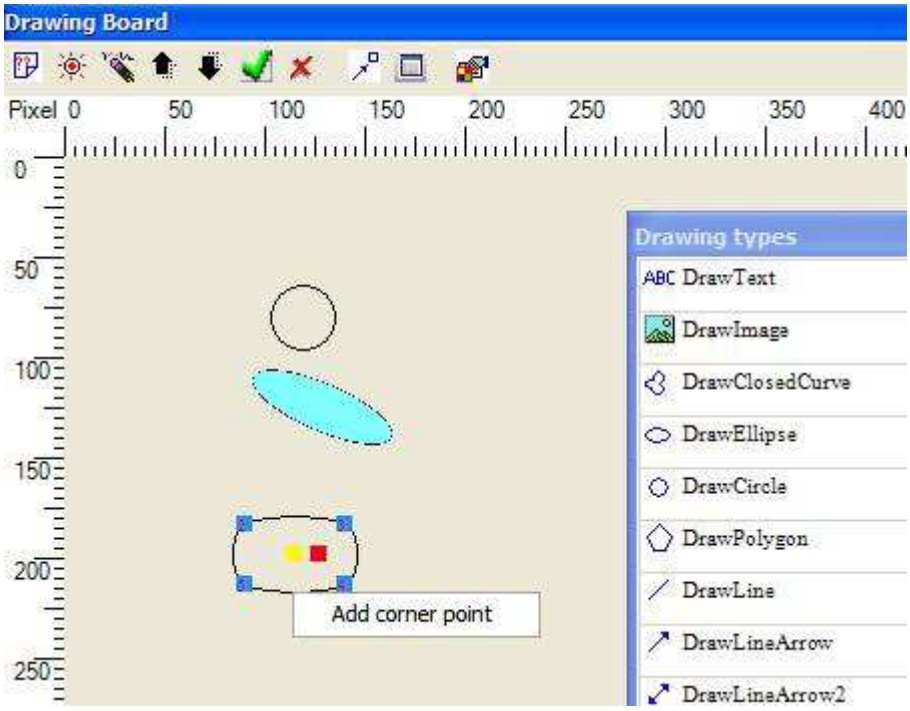


2.4 Create new drawing

To create a new drawing, select a drawing type, click on the drawing board where you want the new drawing to appear.

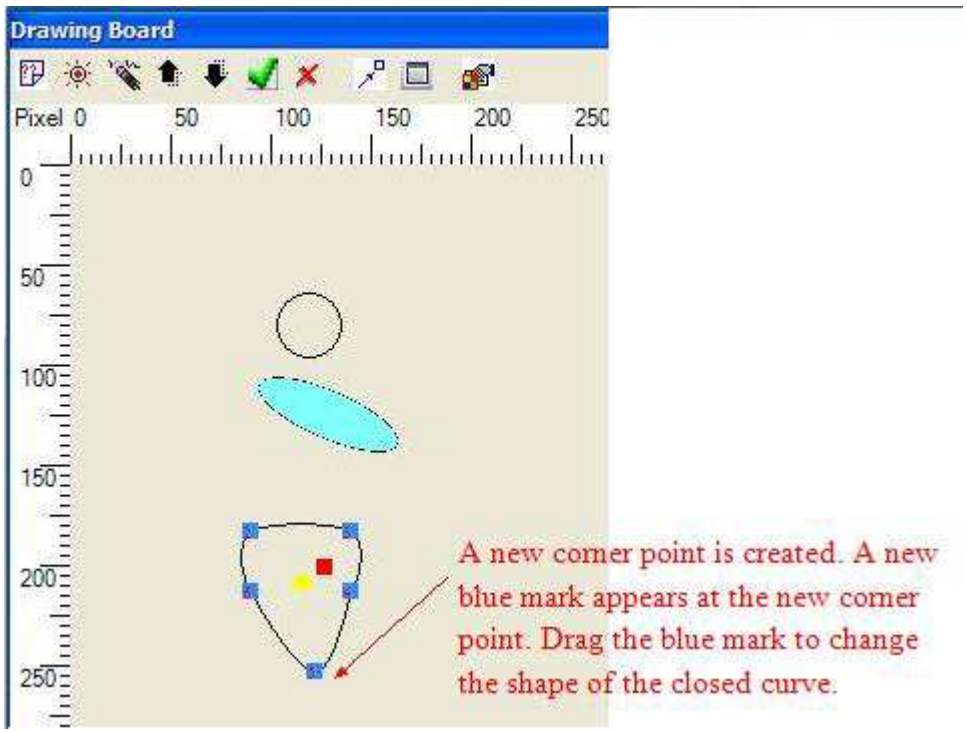


Because we selected "DrawClosedCurve", clicking the drawing board generates a new closed curve:



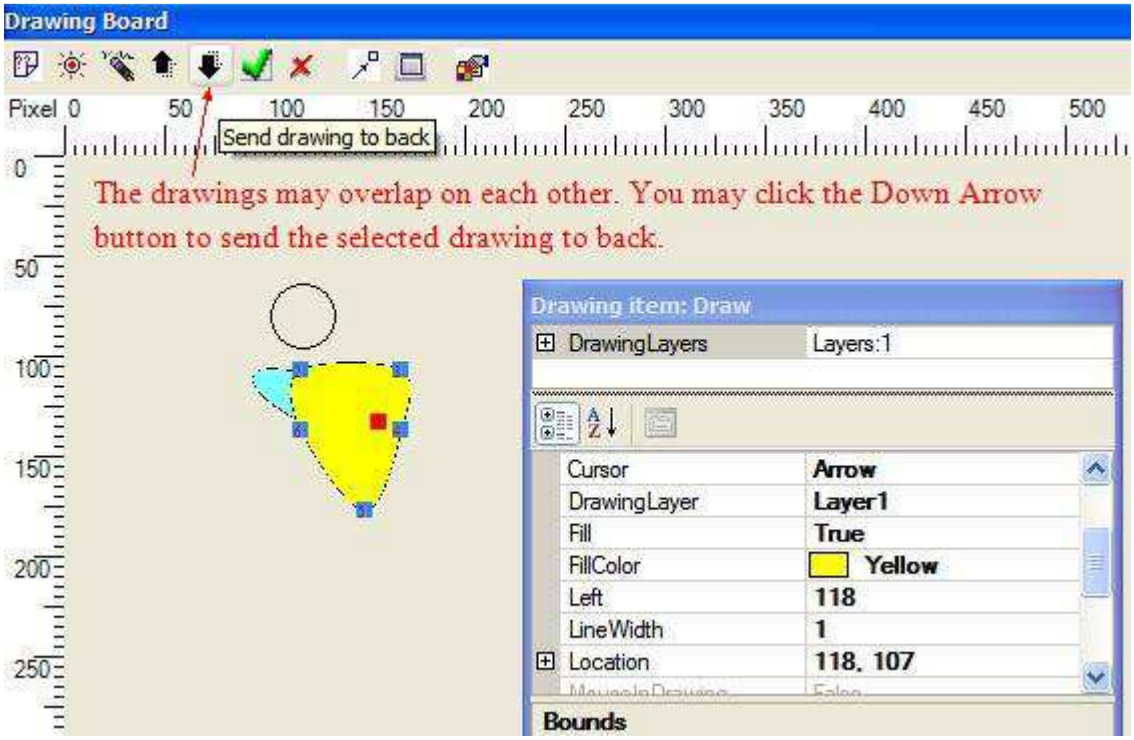
For a closed curve, right-click on the curve, a context menu appears allowing adding a new corner point at the mouse point.

When a new corner point is created a new blue mark appears at the new corner point. Drag the blue mark to change the shape of the closed curve:

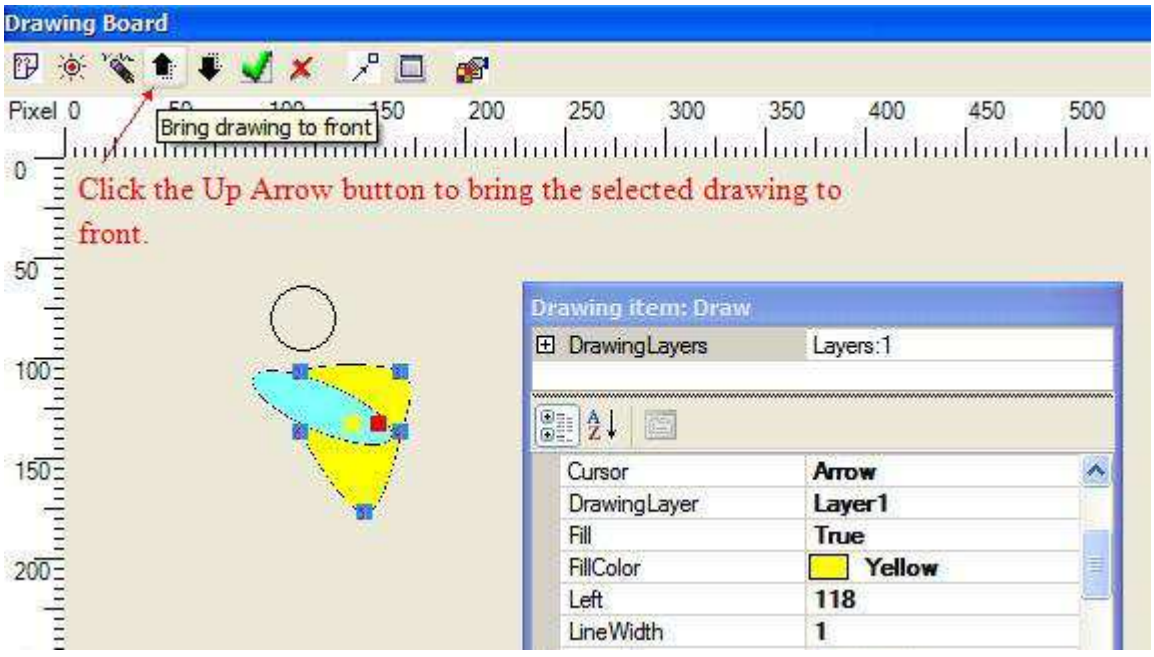


2.5 Change Drawing Z-order

The drawings may overlap on each other. You may click the Down Arrow button to send the selected drawing to back.

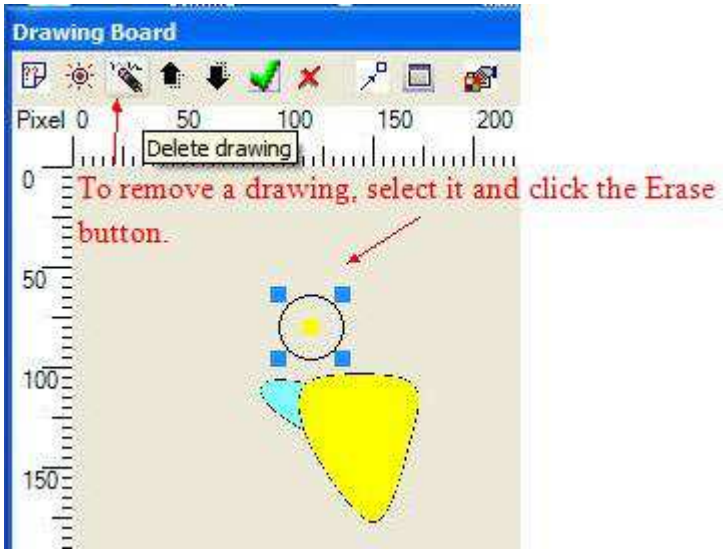


Click the Up Arrow button to bring the selected drawing to front.



2.6 Delete a drawing

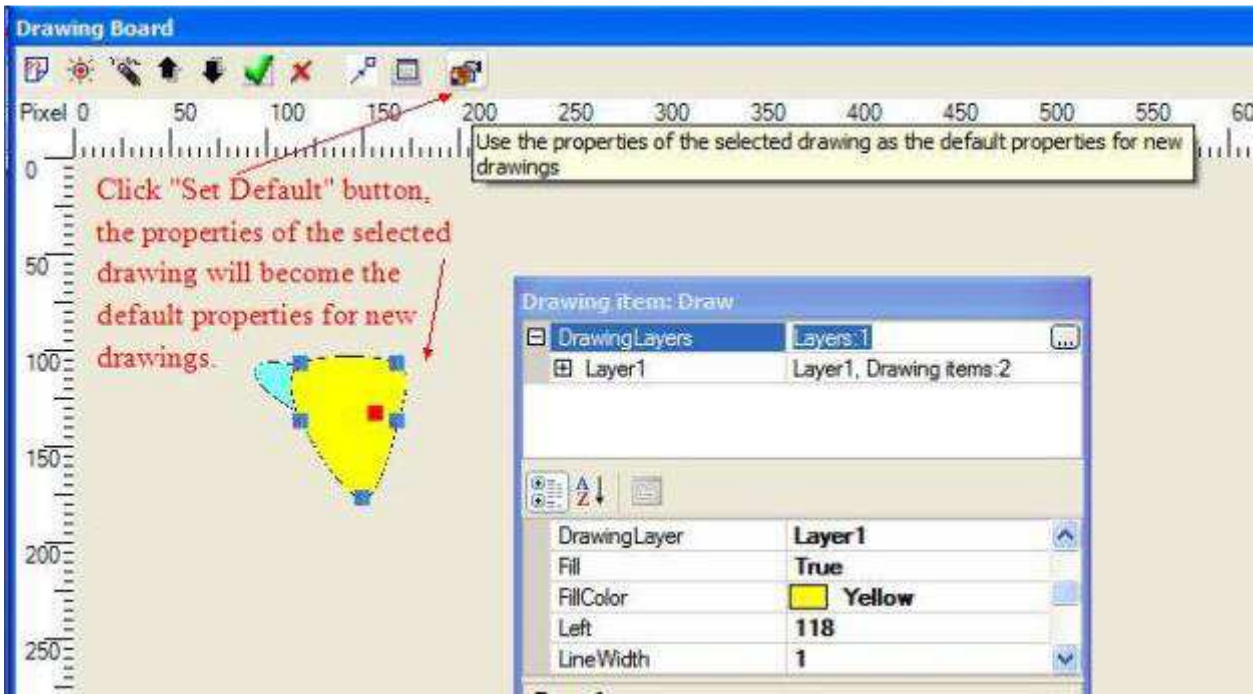
To remove a drawing, select it and click the Erase button.



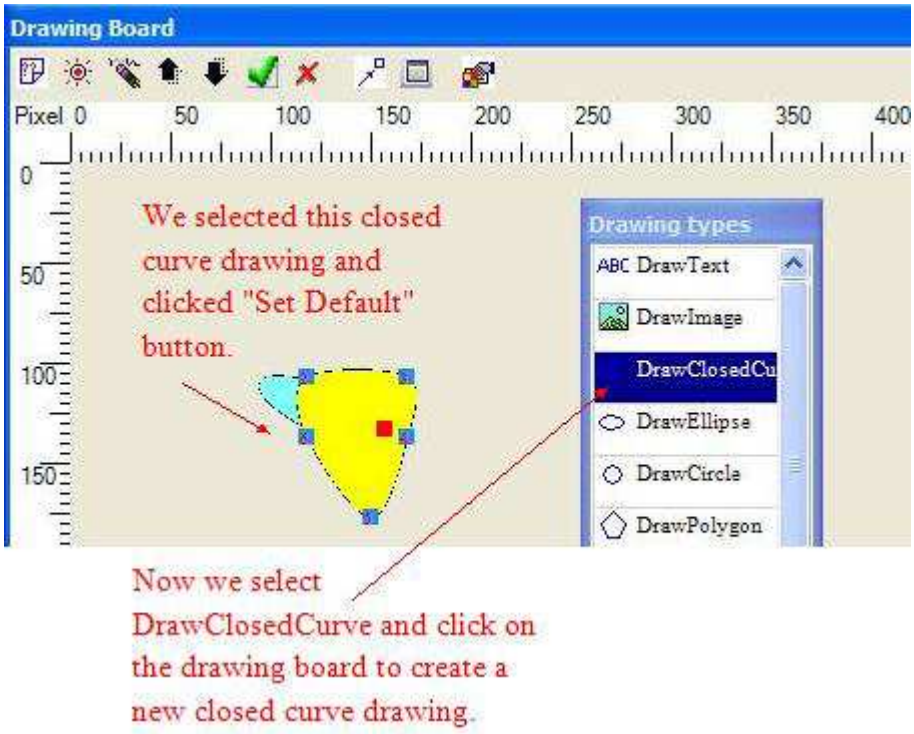
2.7 Set default properties for new drawings

When creating a new drawing we may want it to have properties we want without manually setting its properties. This can be done by creating default properties.

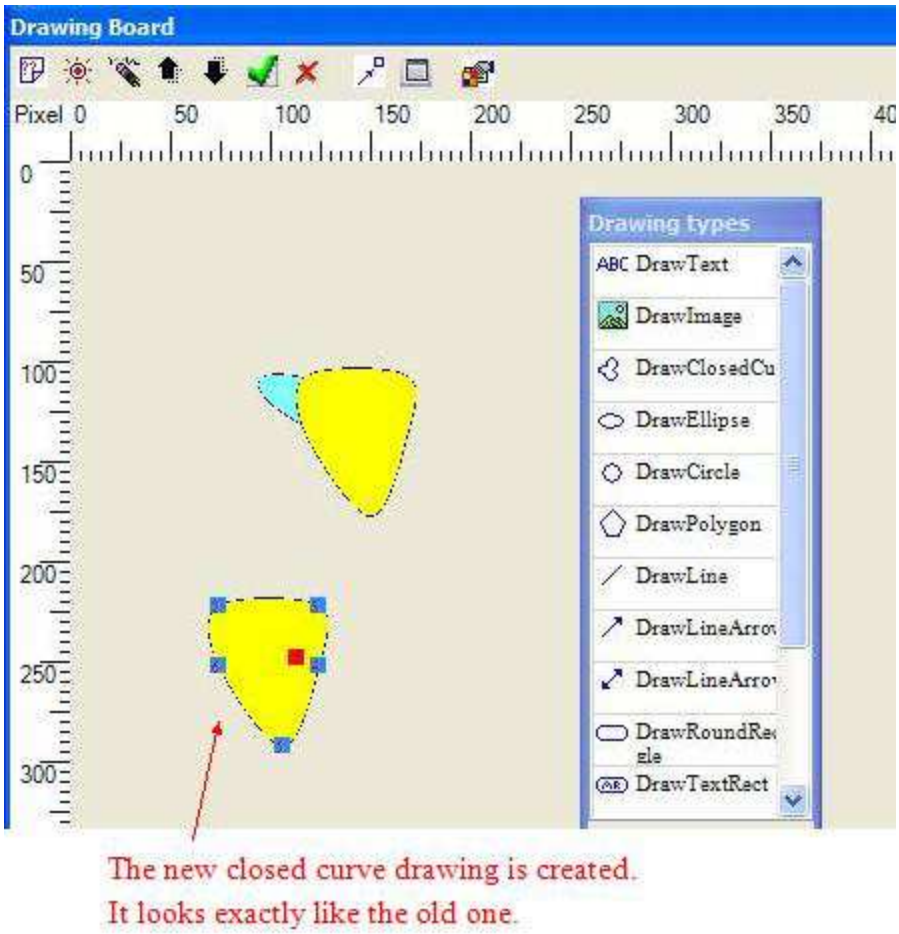
Click "Set Default" button, the properties of the selected drawing will become the default properties for new drawings.



For example, we select a closed curve drawing and clicked "Set Default" button. Then we select DrawClosedCurve and click on the drawing board to create a new closed curve drawing:



The new closed curve drawing is created. It looks exactly like the old one:

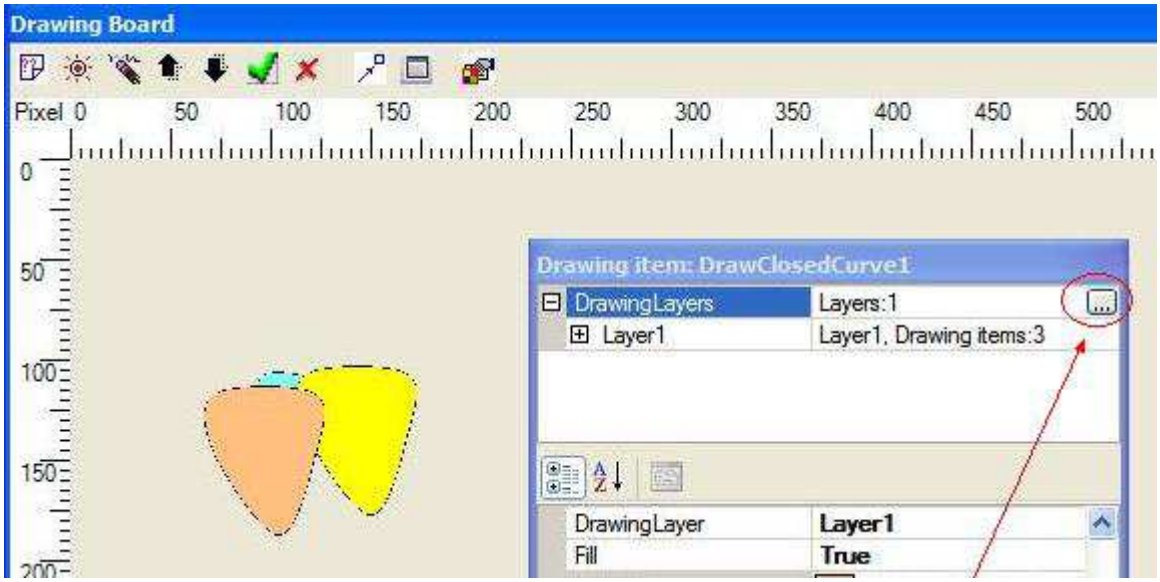


3 Drawing Layers

Every drawing belongs to a Drawing Layer. There is a default drawing layer. By default all drawings belong to the default drawing layer.

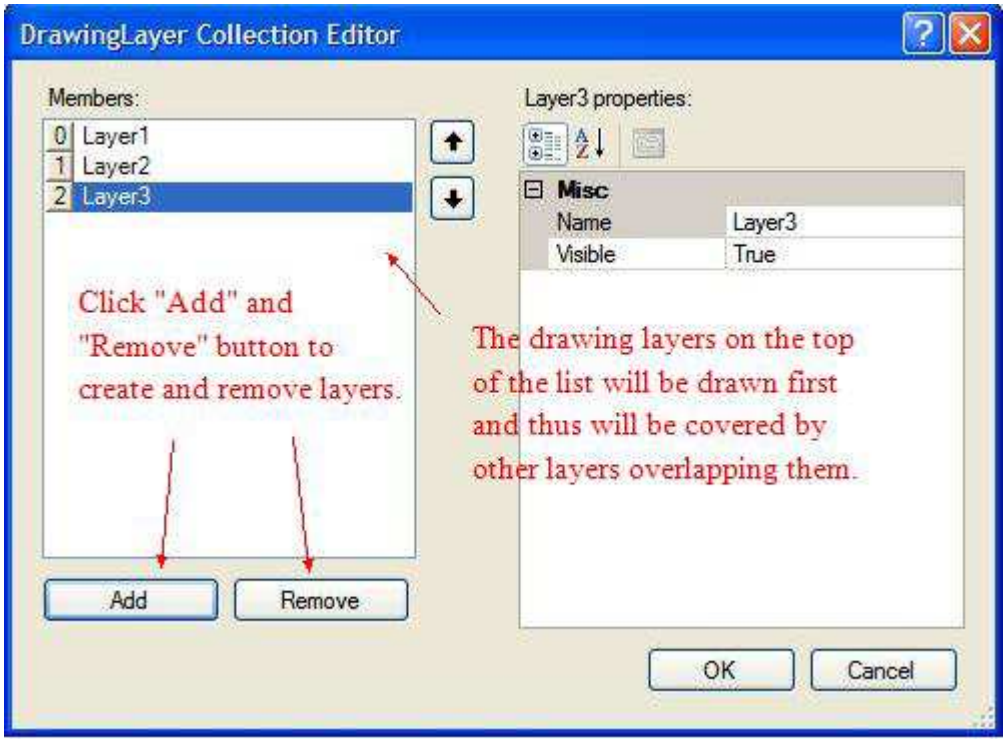
3.1 Create a new drawing layer

To create a new drawing layer, click "…":



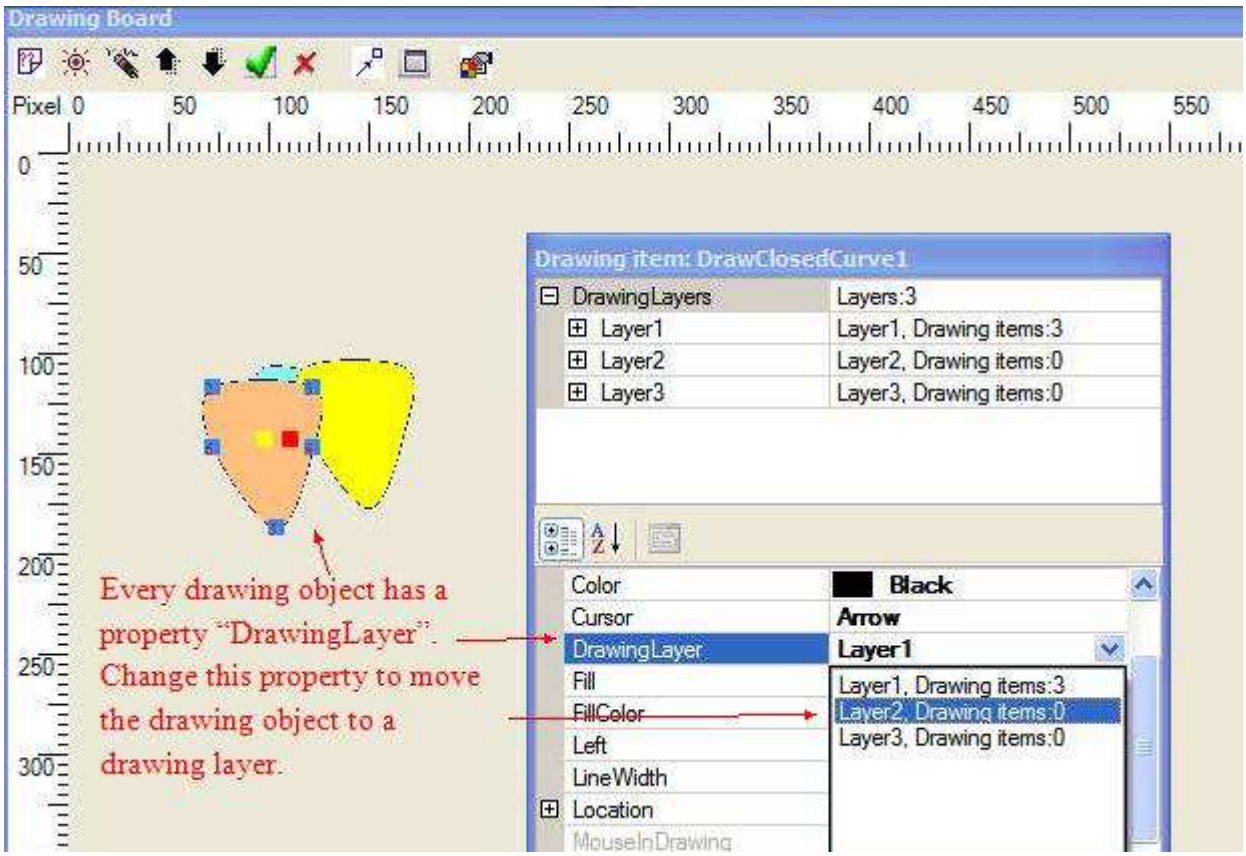
Click "Add" and "Remove" button to create and remove layers.

The drawing layers on the top of the list will be drawn first and thus will be covered by other layers overlapping them



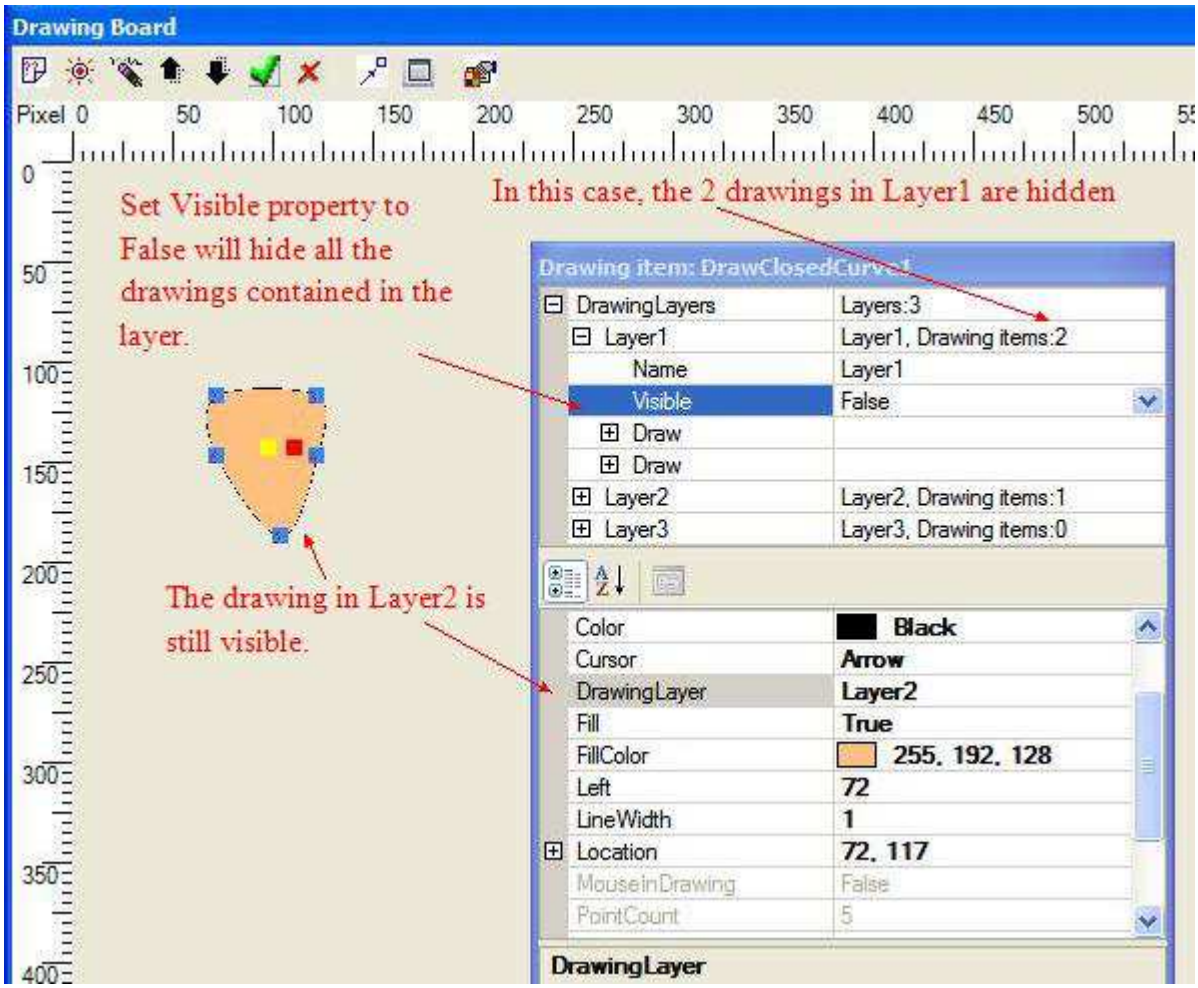
3.2 Change drawing layer for a drawing

Every drawing object has a property “DrawingLayer”. Change this property to move the drawing object to a drawing layer.



3.3 Control drawing visibility through drawing layer

Set Visible property to False will hide all the drawings contained in the layer at runtime. At the design time all drawing layers are visible even the Visible property is False.

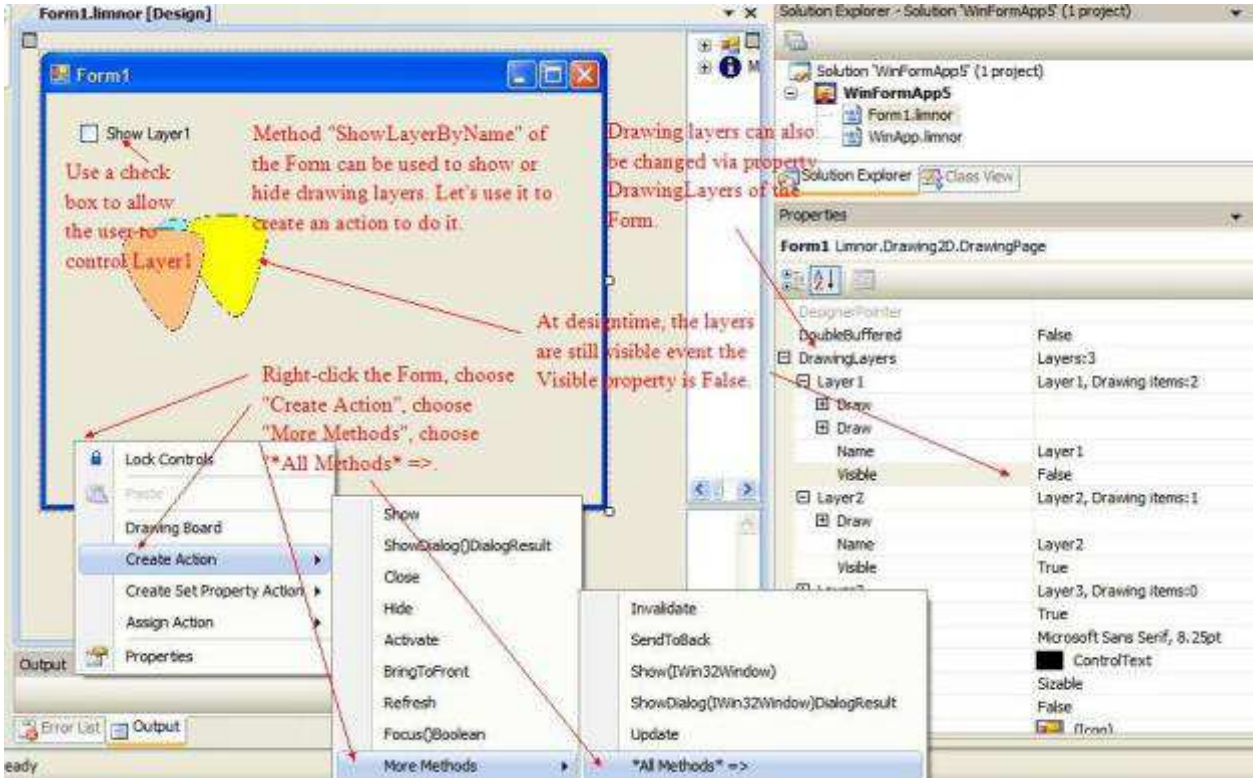


3.4 Change drawing layer at run time

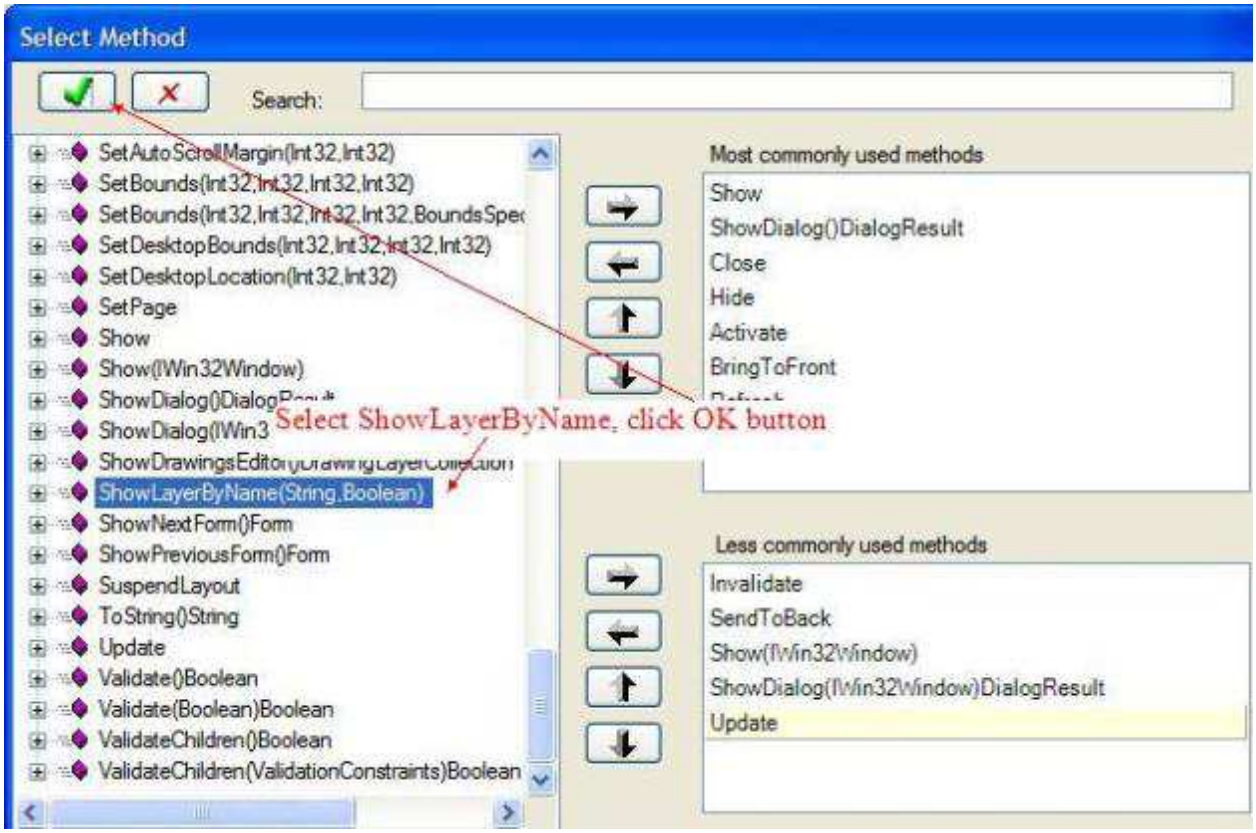
As an example, use a check box to allow the user to control Layer1

Method "ShowLayerByName" of the Form can be used to show or hide drawing layers. Let's use it to create an action to do it.

Right-click the Form, choose "Create Action", choose "More Methods", choose "*All Methods* =>"



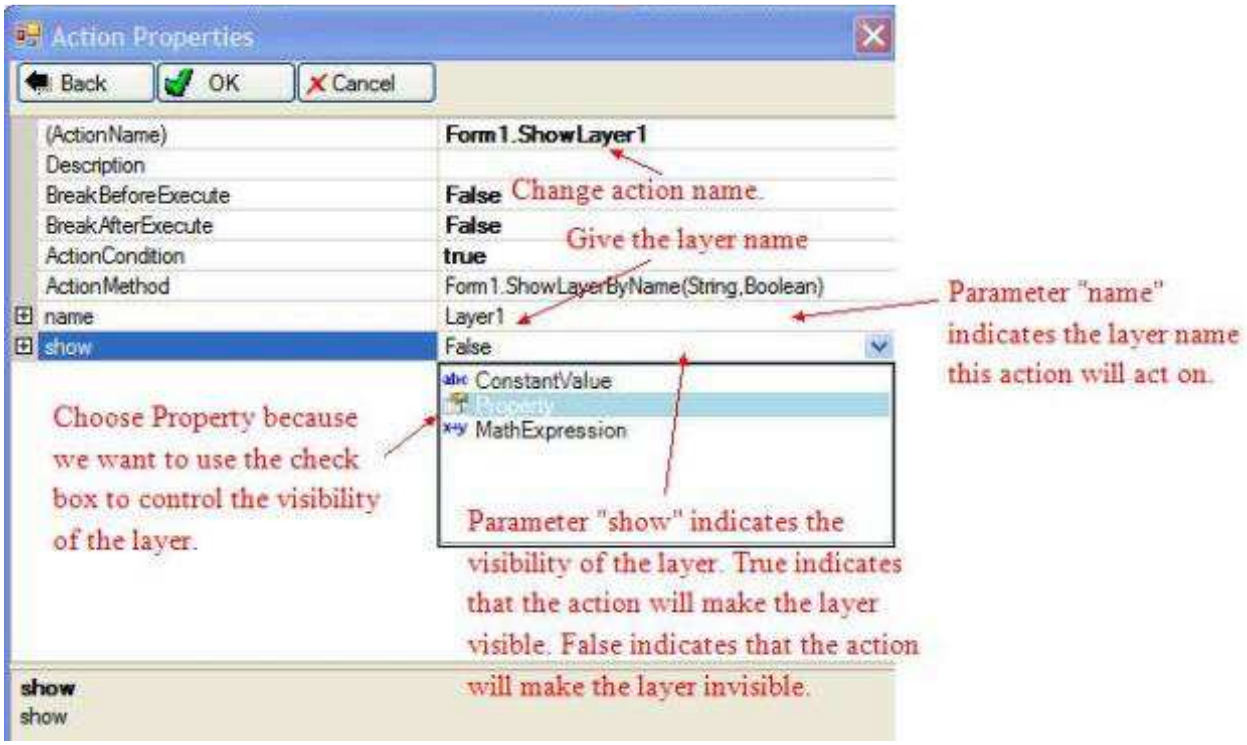
Select ShowLayerByName, click OK button



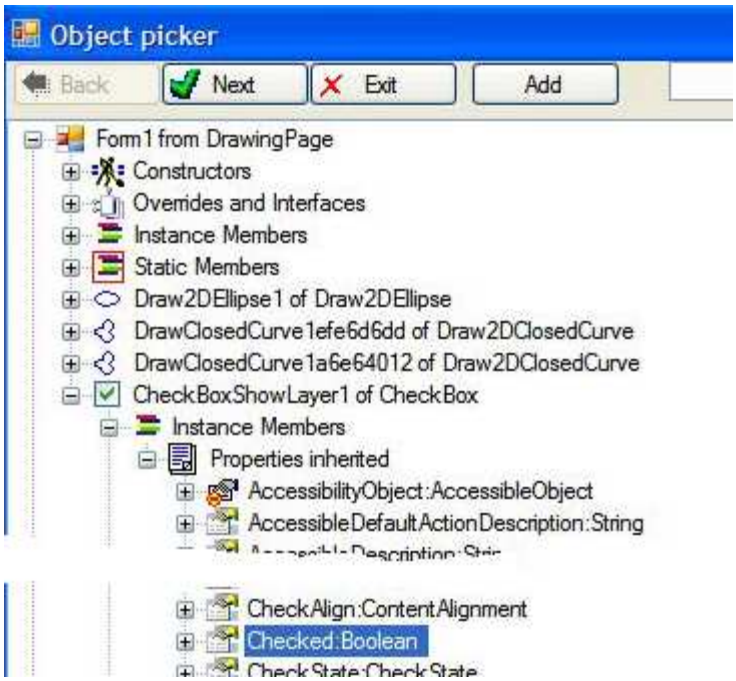
Parameter "name" indicates the layer name this action will act on. In our case it is “Layer1”.

Parameter "show" indicates the visibility of the layer. True indicates that the action will make the layer visible. False indicates that the action will make the layer invisible.

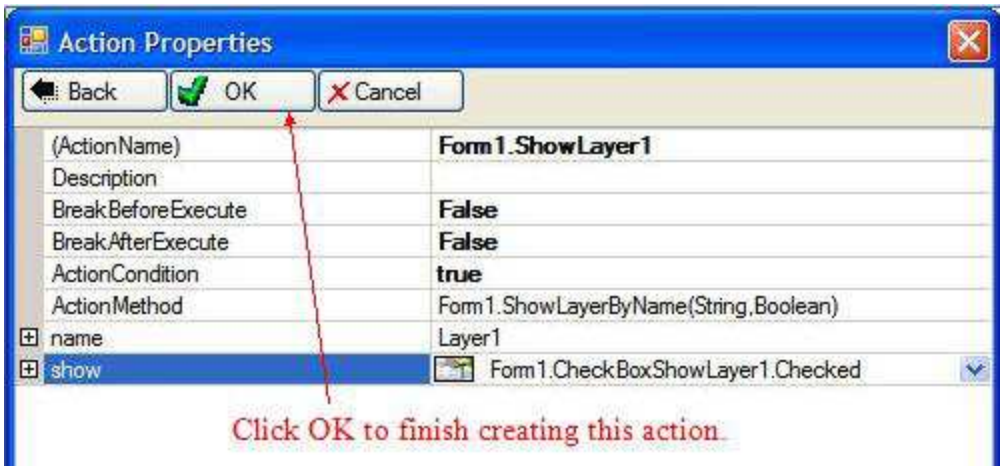
Choose Property for "show" because we want to use the check box to control the visibility of the layer.



Select Checked property of the CheckBox. Click Next.

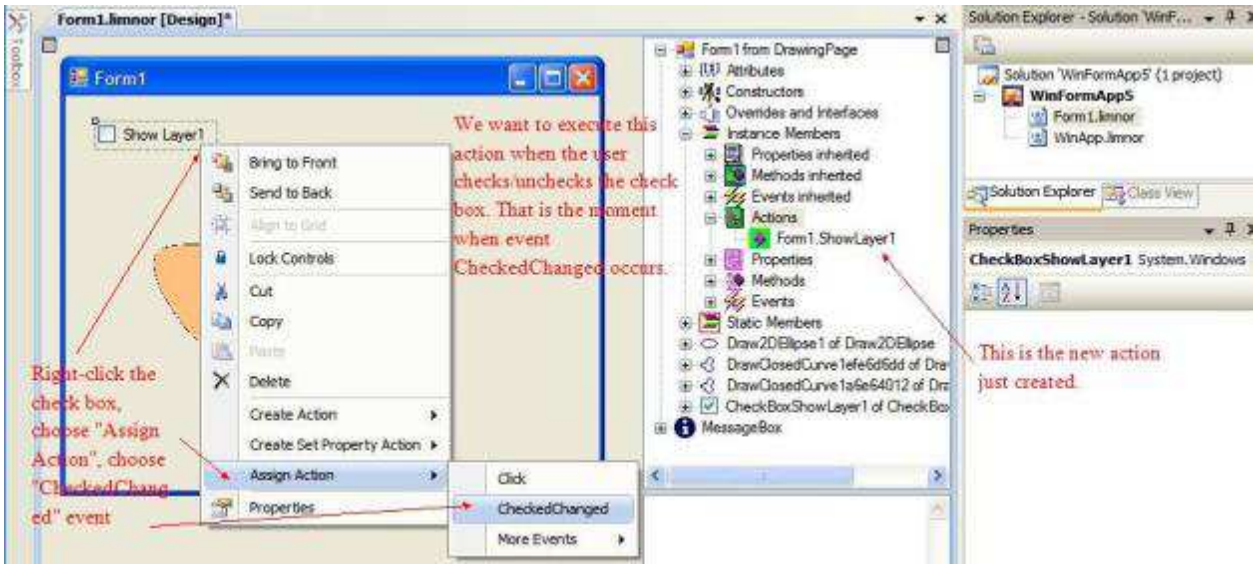


This action will show Layer1 if the check box is checked; it will hide Layer1 if the check box is unchecked.

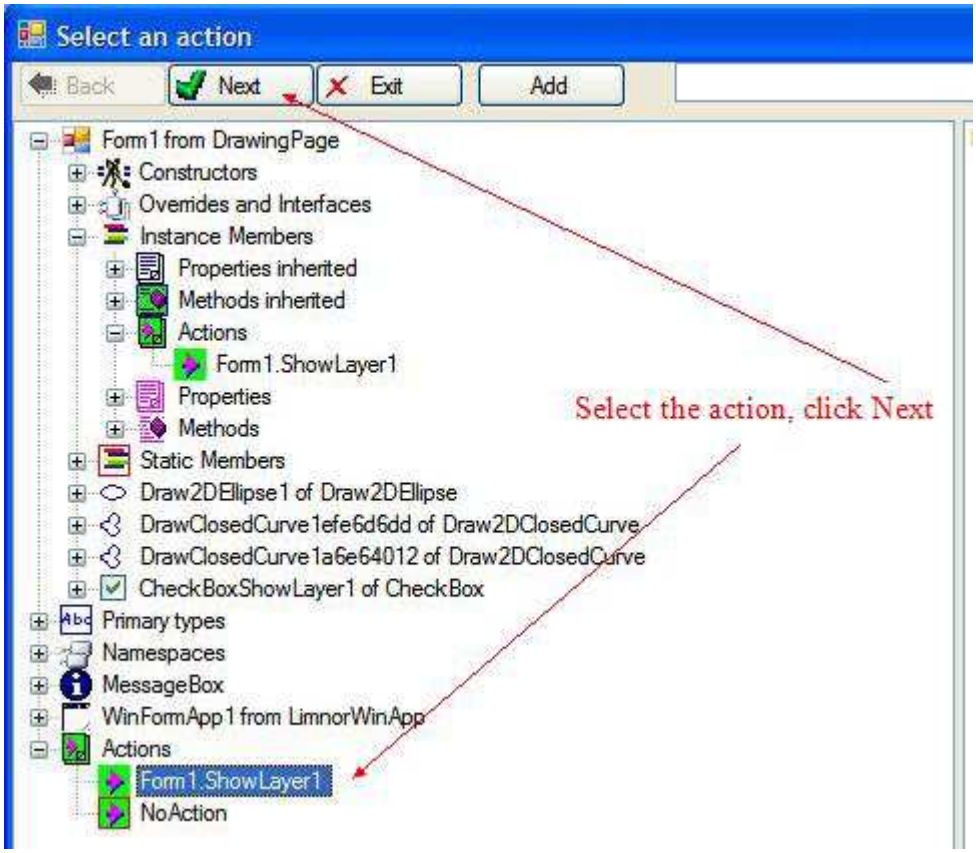


We want to execute this action when the user checks/unchecks the check box. That is the moment when event CheckedChanged occurs.

Right-click the check box, choose "Assign Action", choose "CheckedChanged" event:

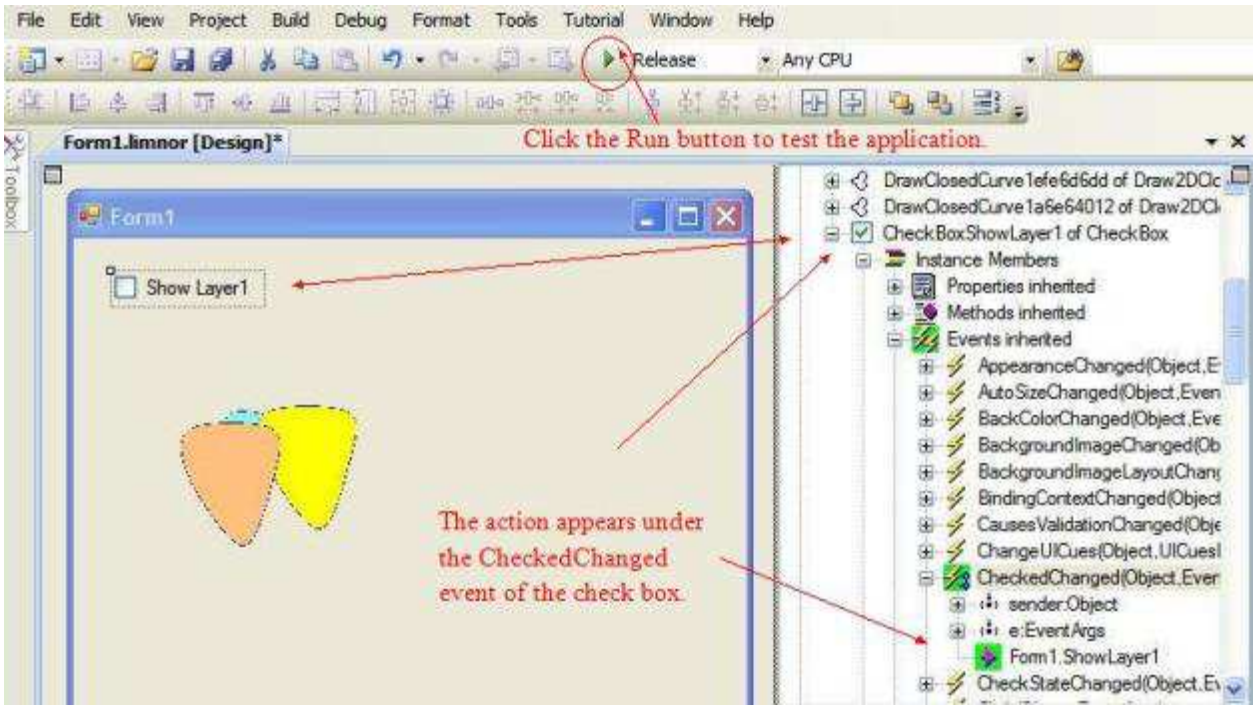


Select the action. Click Next.

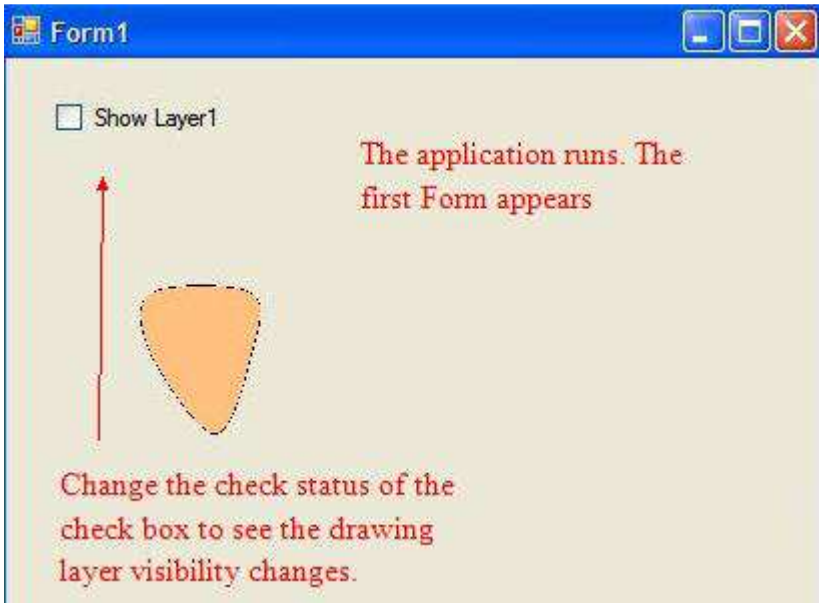


The action appears under the CheckedChanged event of the check box.

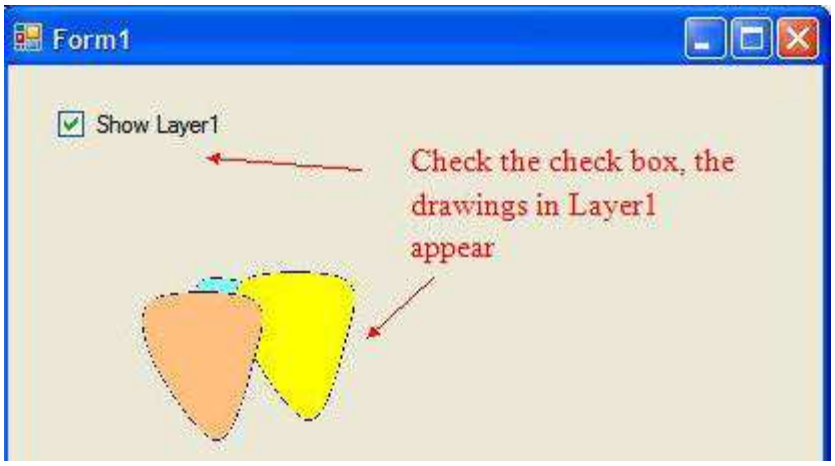
Click the Run button to test the application.



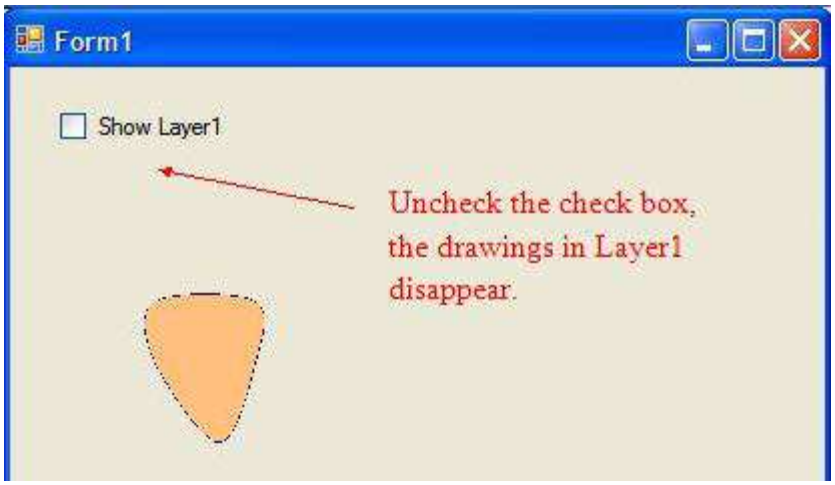
The application runs. The first Form appears. Change the check status of the check box to see the drawing layer visibility changes.



Check the check box, the drawings in Layer1 appear:



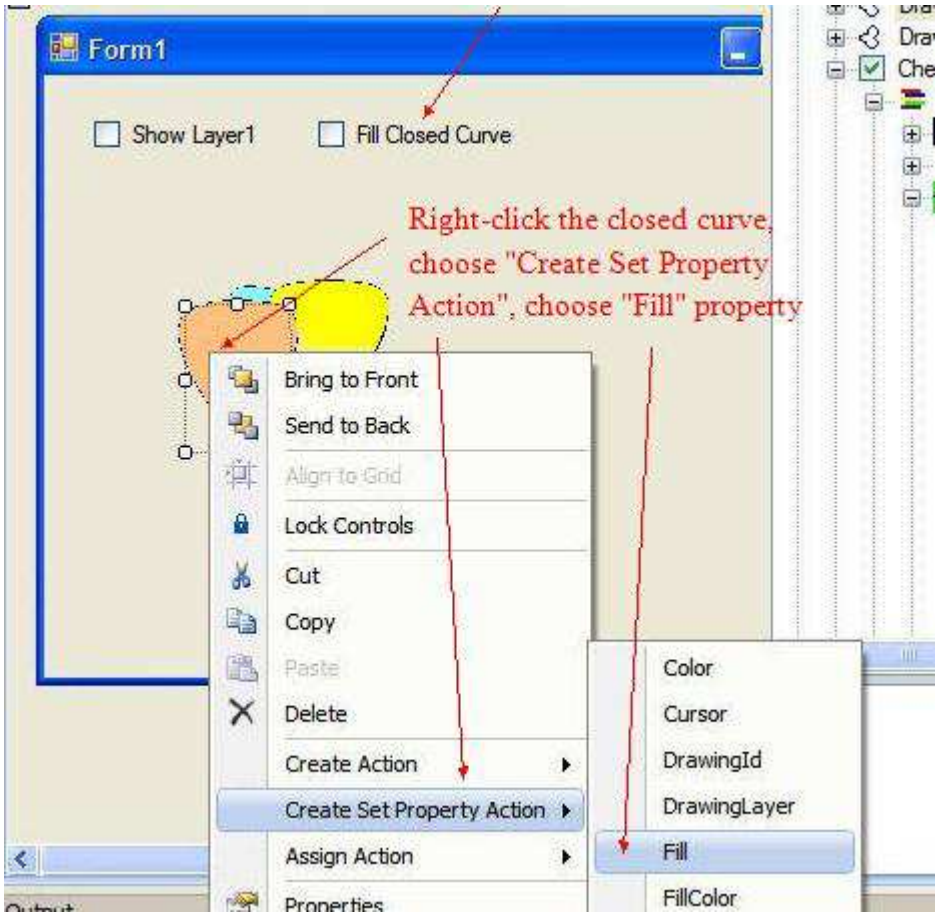
Uncheck the check box, the drawings in Layer1 disappear:



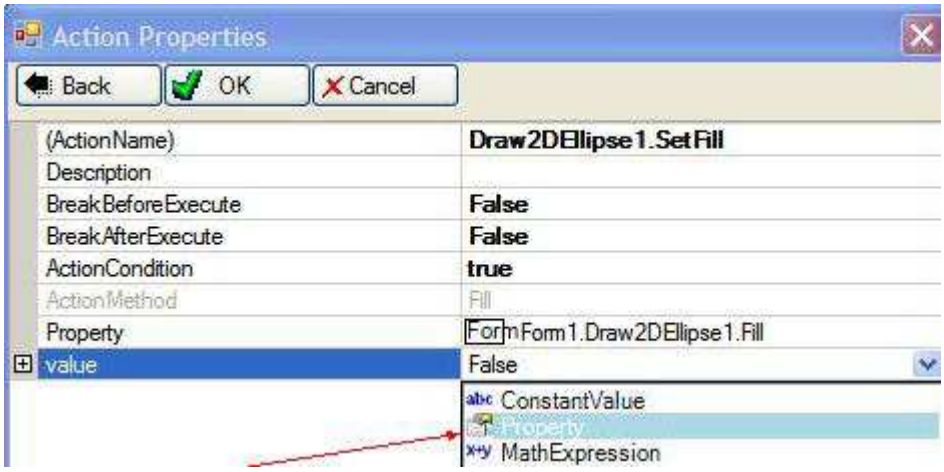
4 Change drawings at runtime

Individual drawing can also be controlled by actions. For example, let's use a check box to control the filling of a closed curve.

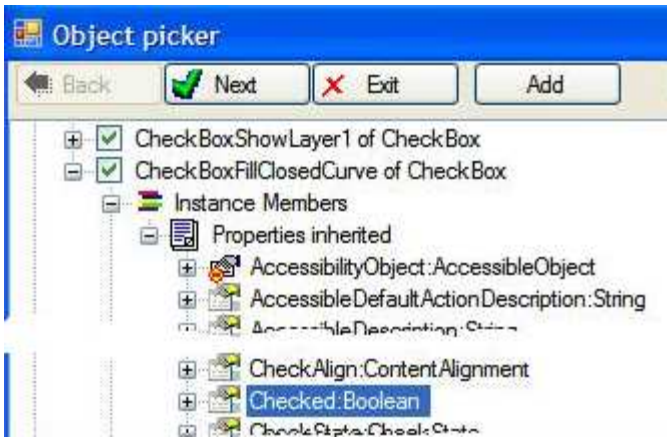
Right-click the closed curve, choose "Create Set Property Action", choose "Fill" property:



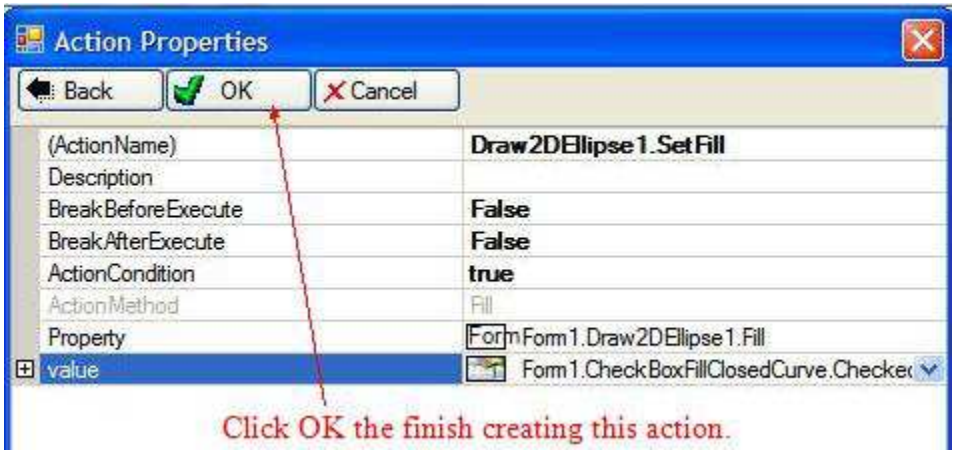
Select Property because we want to use the check box to set the property.



Select Checked property of the Check Box. Click Next.

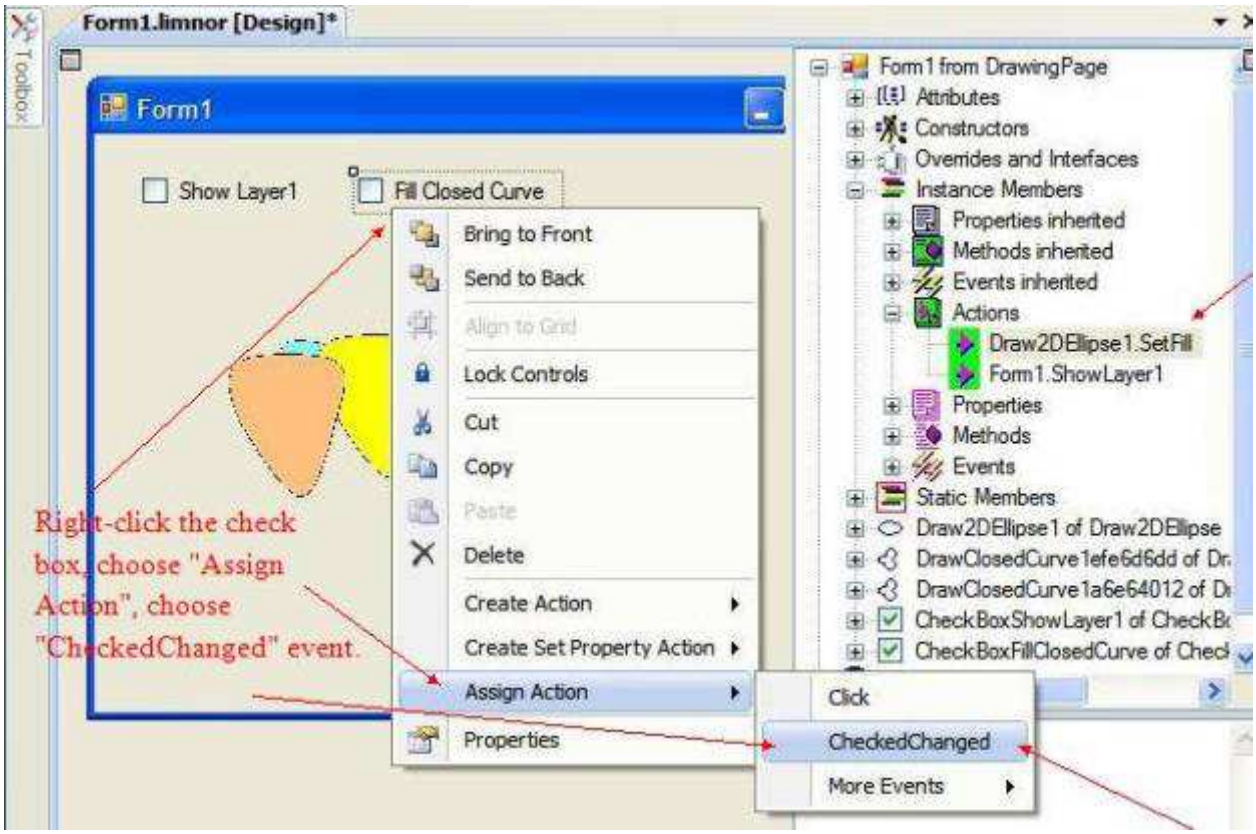


This action assigns the value of Checked property of the check box to the Fill property of the closed curve.

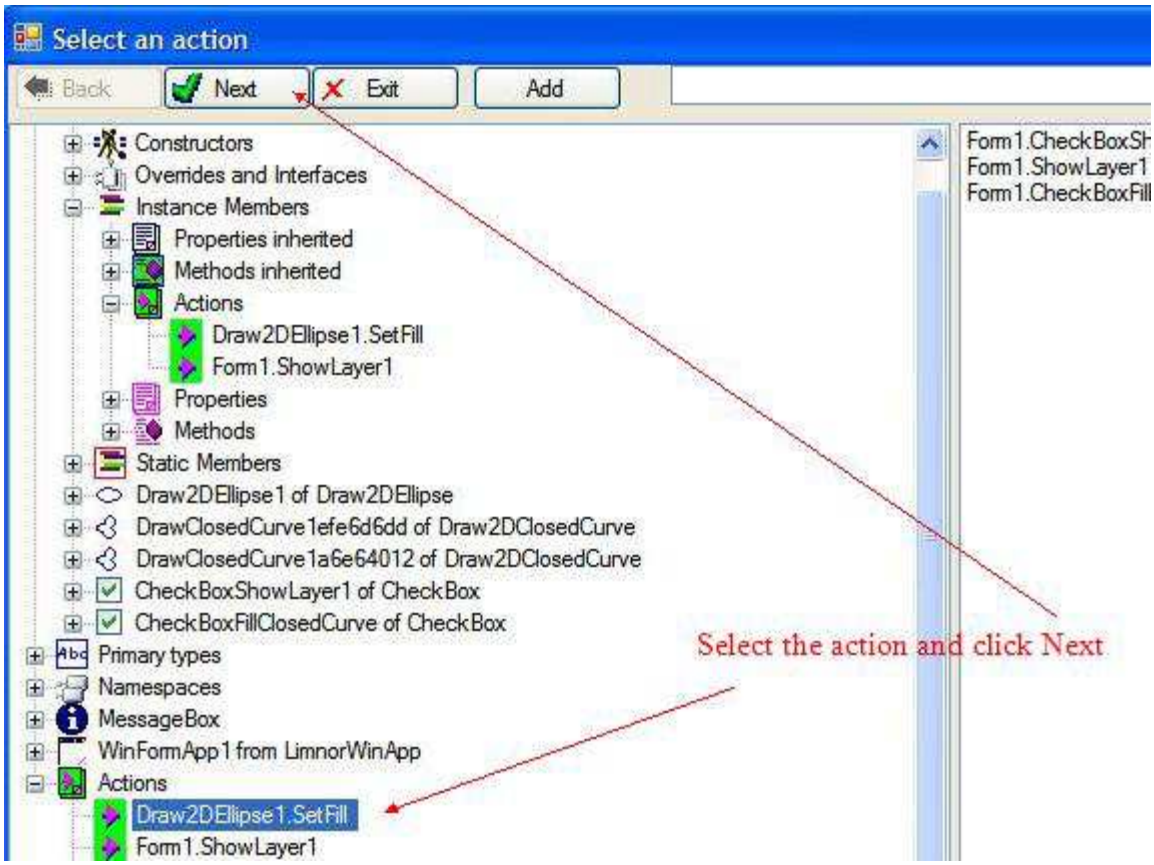


We want to execute this action when the use checks or unchecks the check box. That is, when CheckedChanged event of the check box occurs.

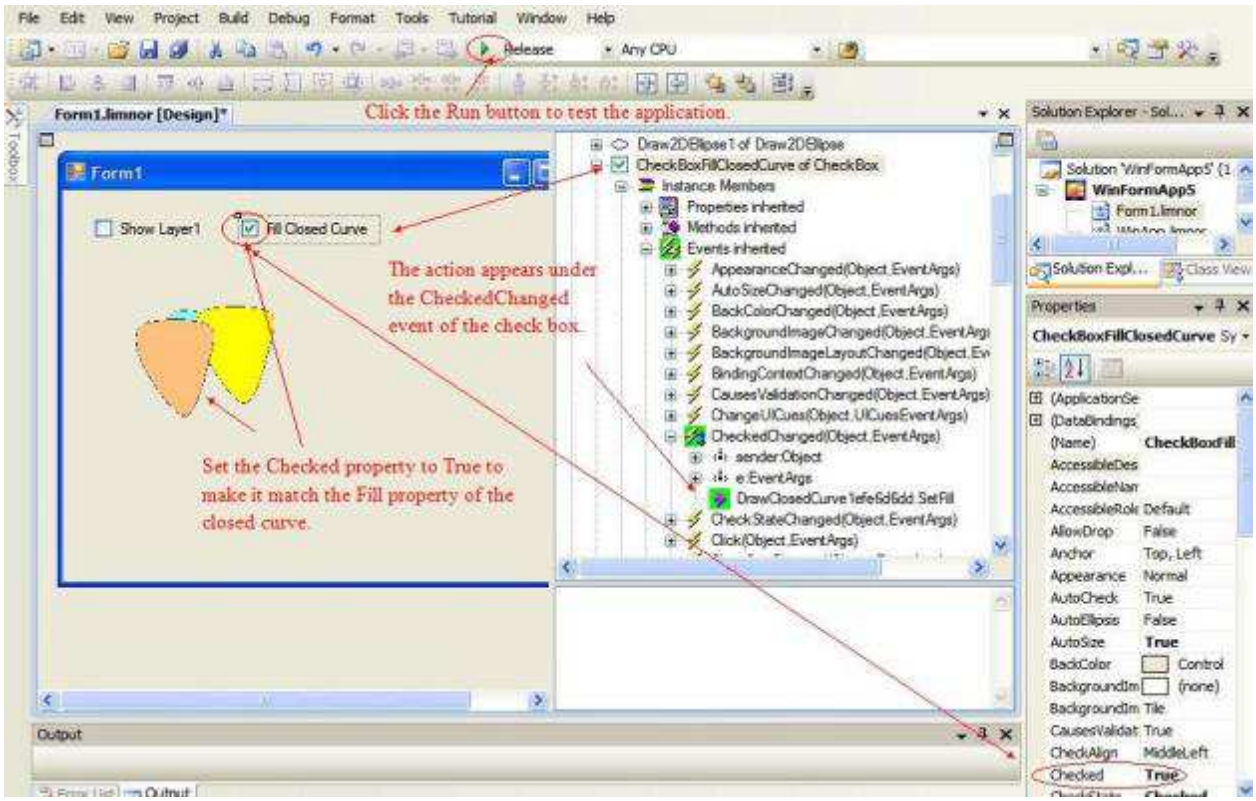
Right-click the check box, choose "Assign Action", choose "CheckedChanged" event.



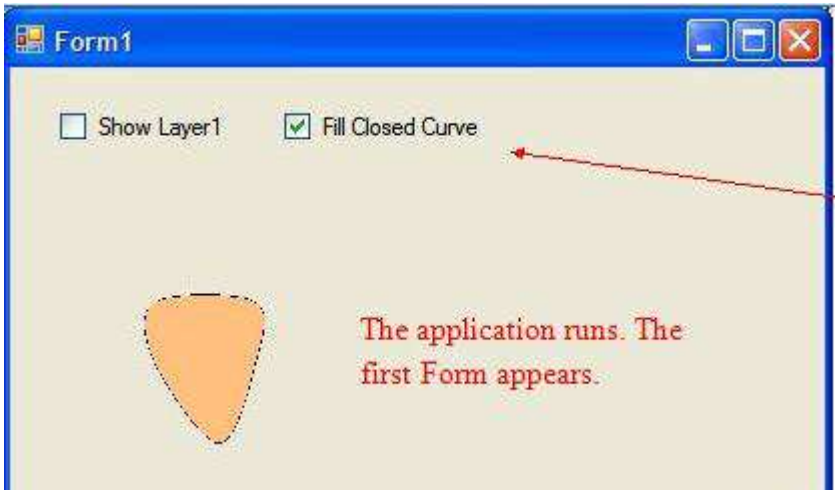
Select the action and click Next:



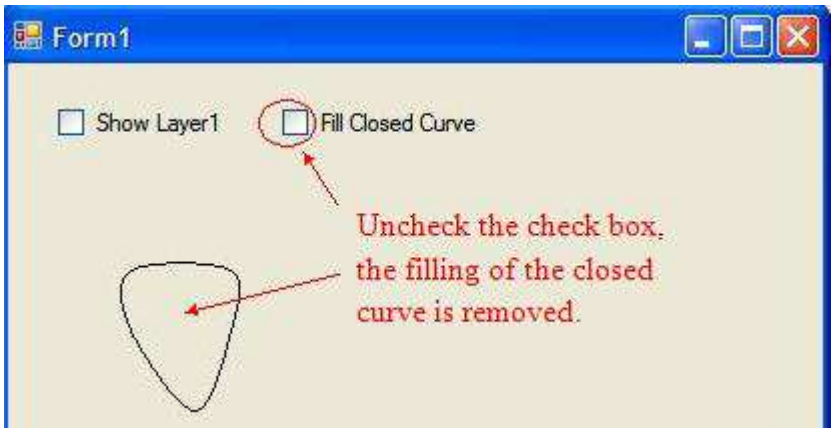
Set the Checked property to True to make it match the Fill property of the closed curve. We may test the application now:



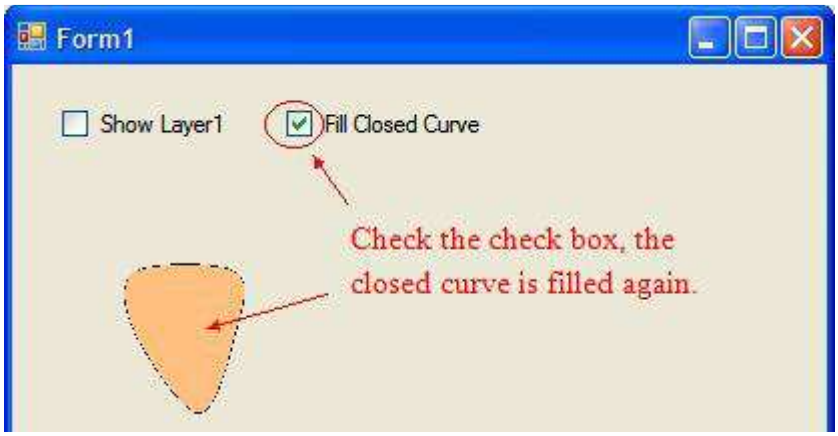
The application runs. The first Form appears. Change the check status of the check box to see how it changes the filling of the closed curve.



Uncheck the check box, the filling of the closed curve is removed.



Check the check box, the closed curve is filled again

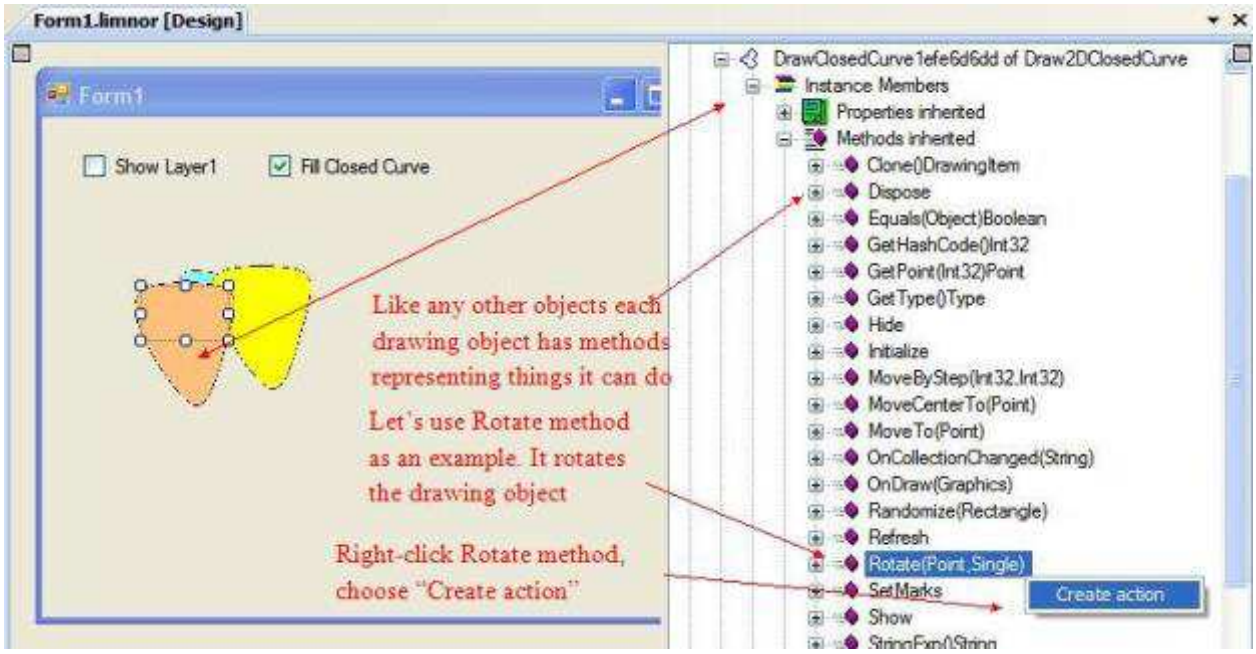


5 Execute drawing actions

Like any other objects each drawing object has methods representing things it can do

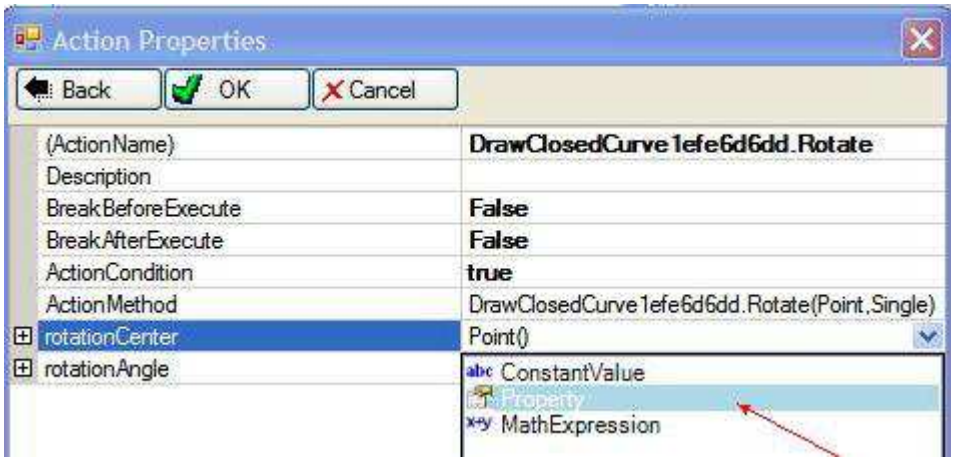
Let's use Rotate method as an example. It rotates the drawing object around a point.

Right-click Rotate method, choose "Create action"

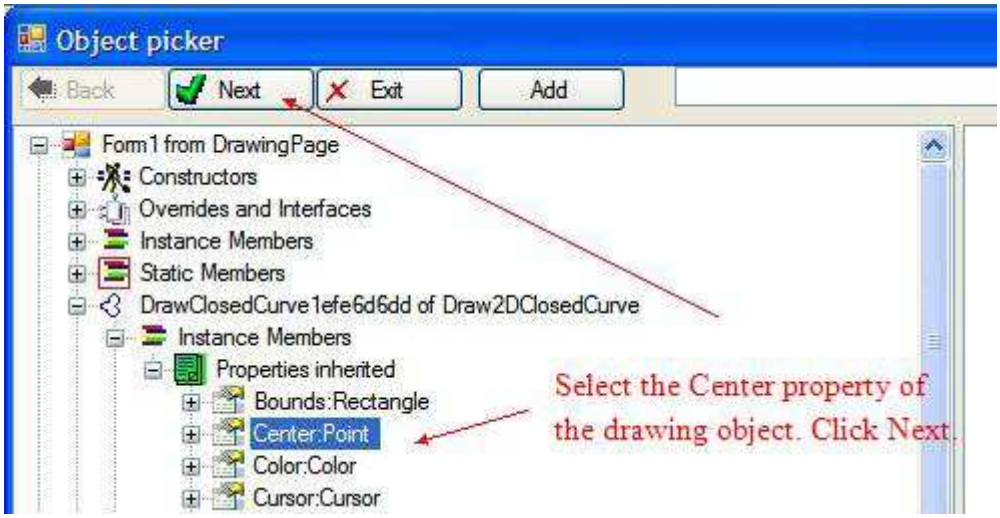


This action requires us to give two parameters: rotationCenter and rotationAngle.

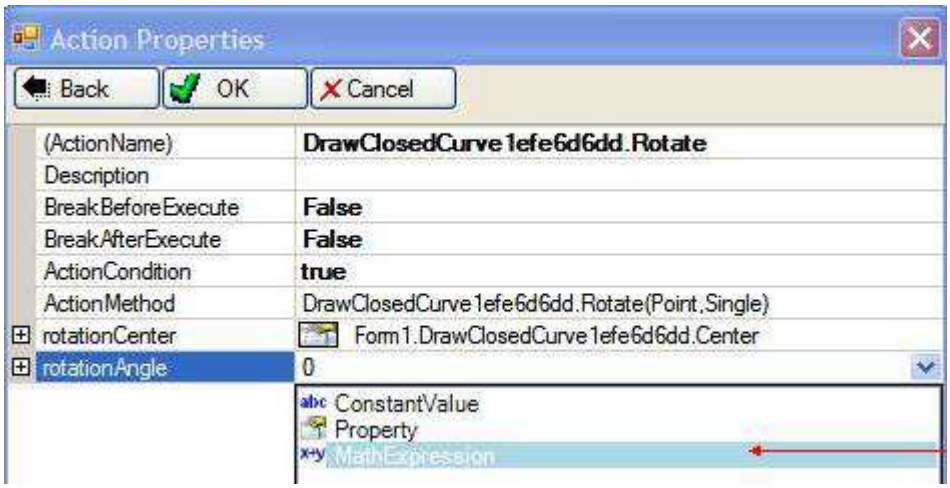
The rotation will be around the rotationCenter for an angle specified by rotationAngle. In this example, let's make the rotation around the center of the drawing object, a close curve. So, choose Property for rotationCenter.



Select the Center property of the drawing object. Click Next.

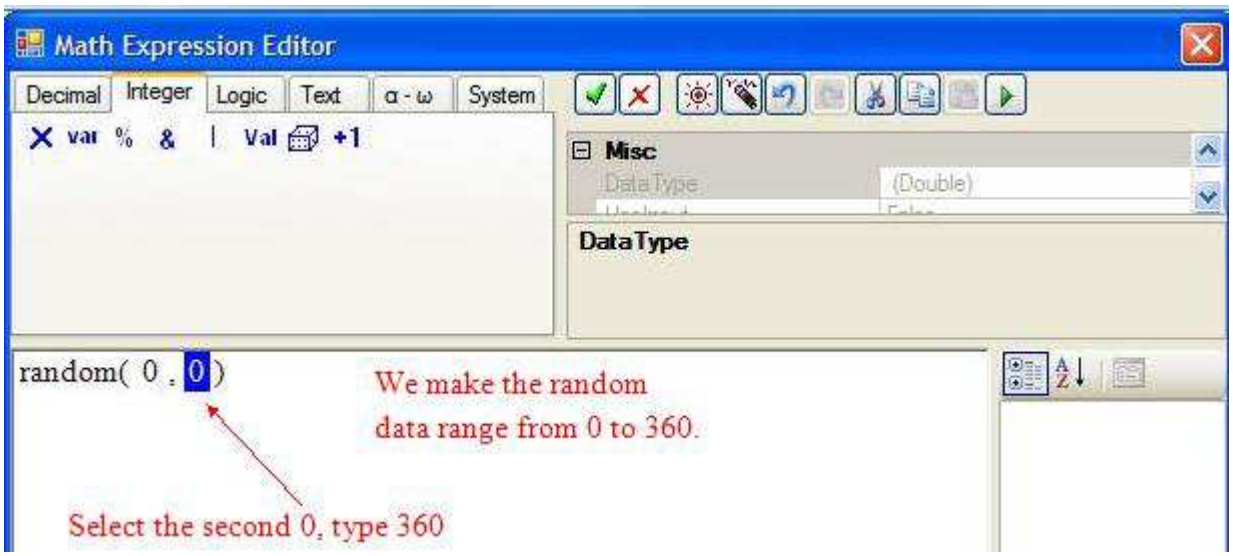


For this example, we want to use a random rotation angle. So, we choose MathExpression for rotationAngle

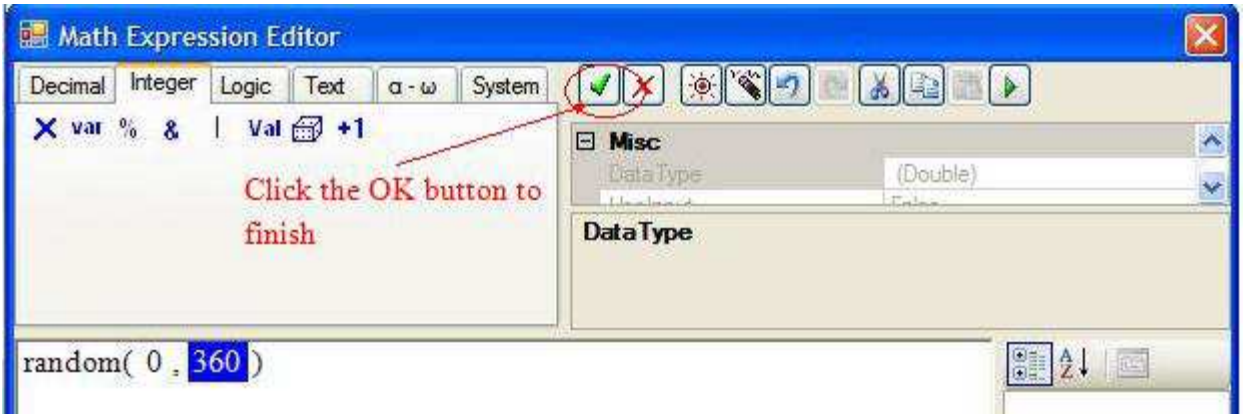


Click the dice button.

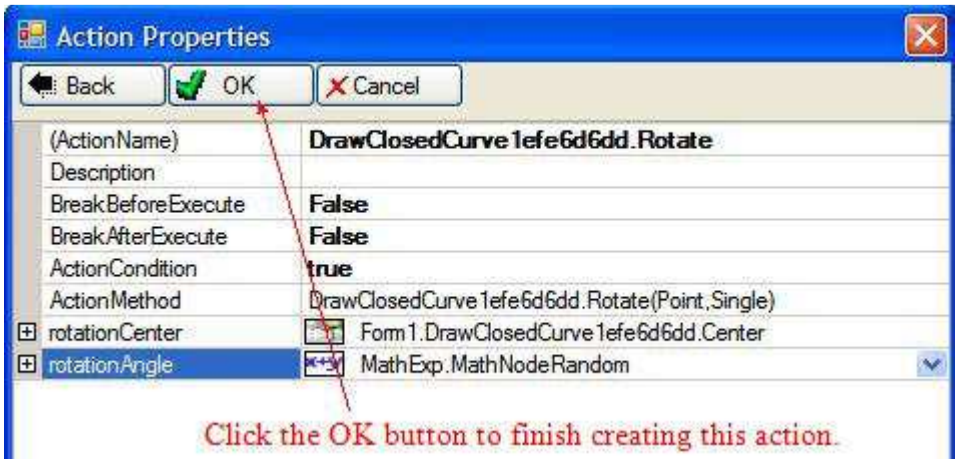
We make the random data range from 0 to 360. Select the second 0, type 360.



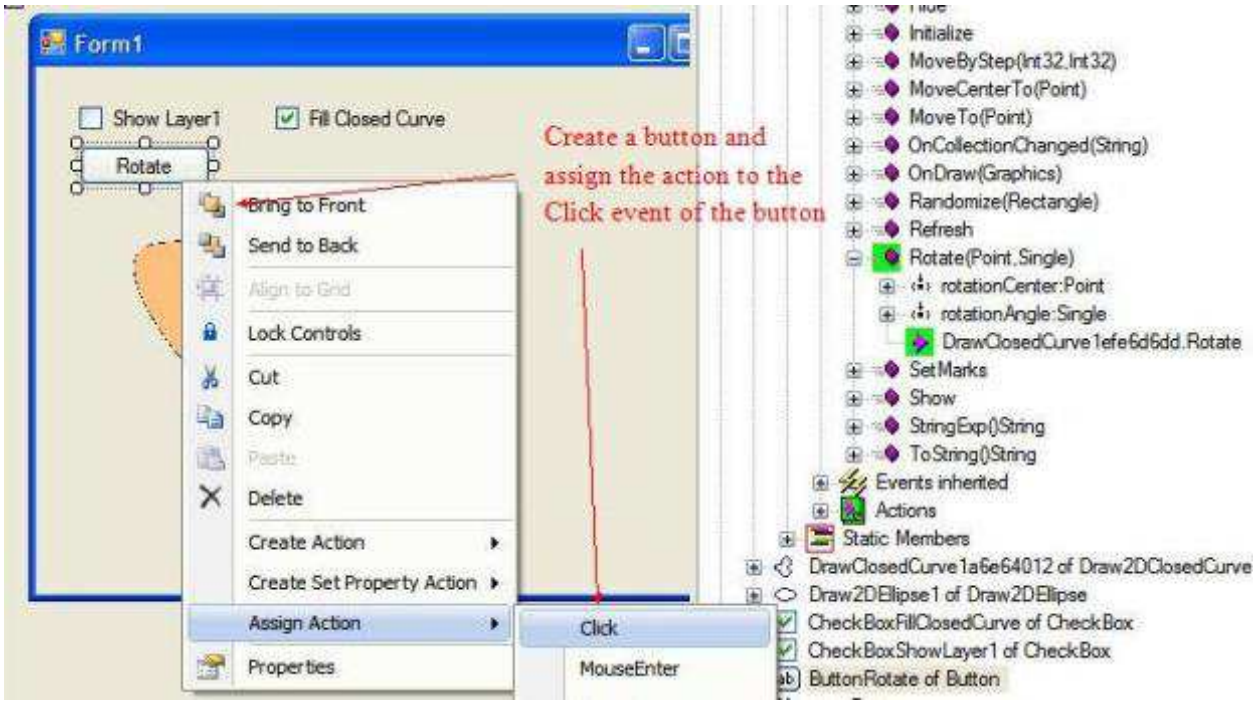
Click OK to finish:



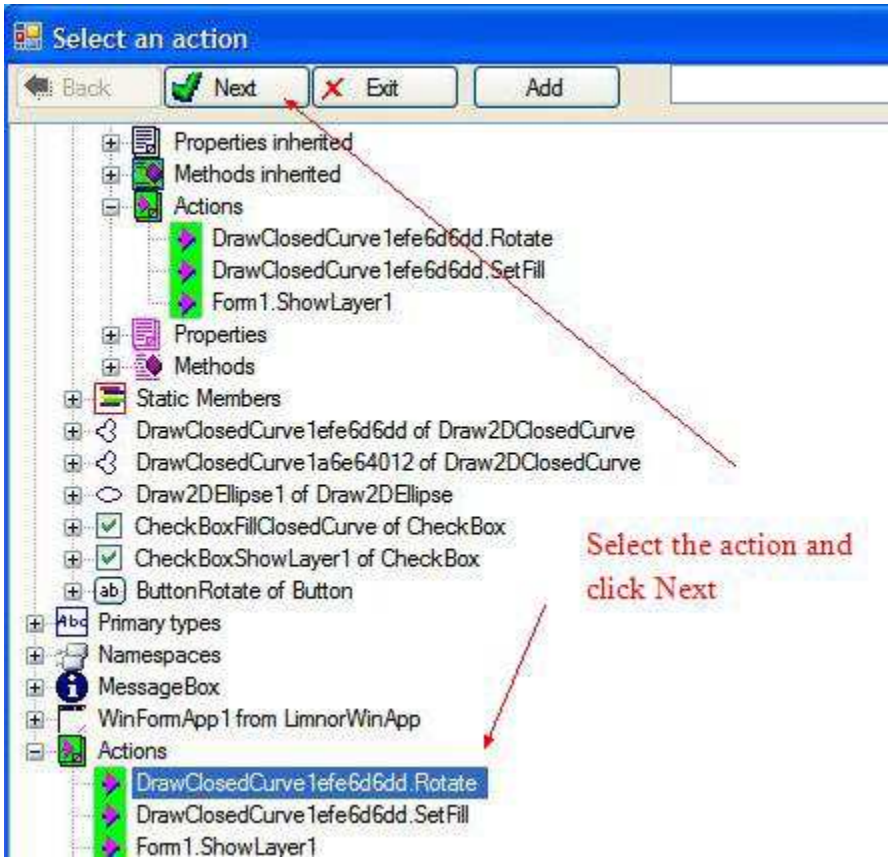
This action will rotate the closed curve around the center of it by a random degree.



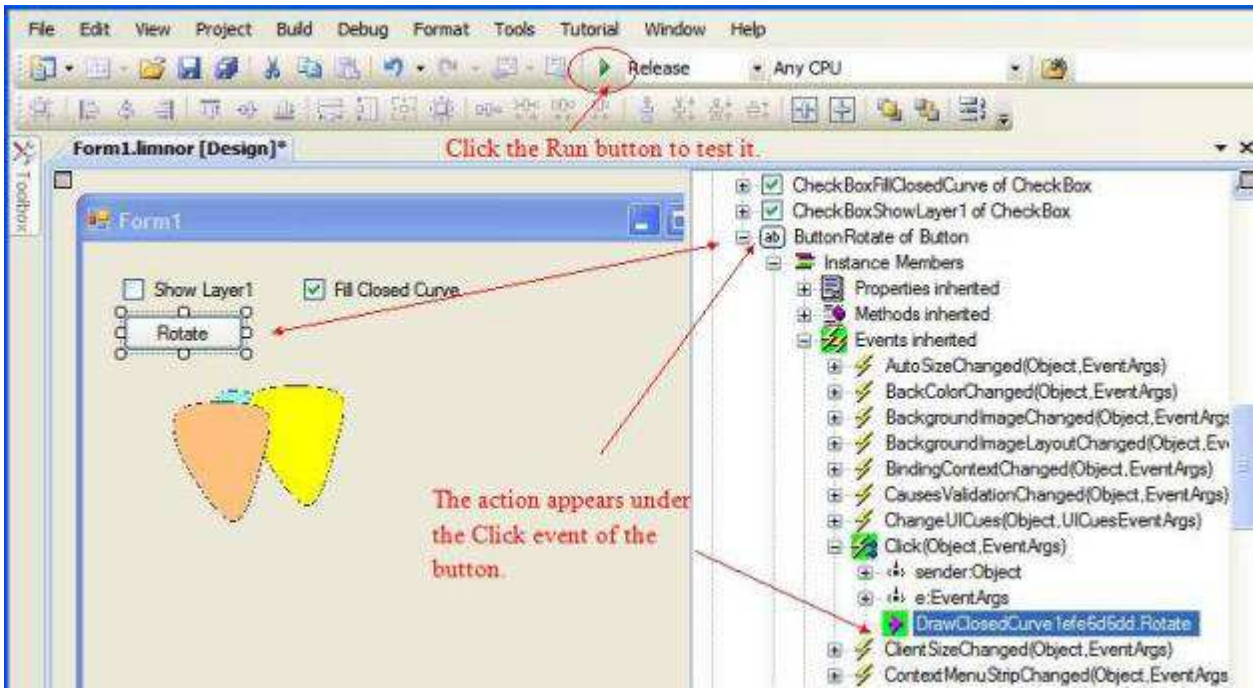
Create a button and assign the action to the Click event of the button



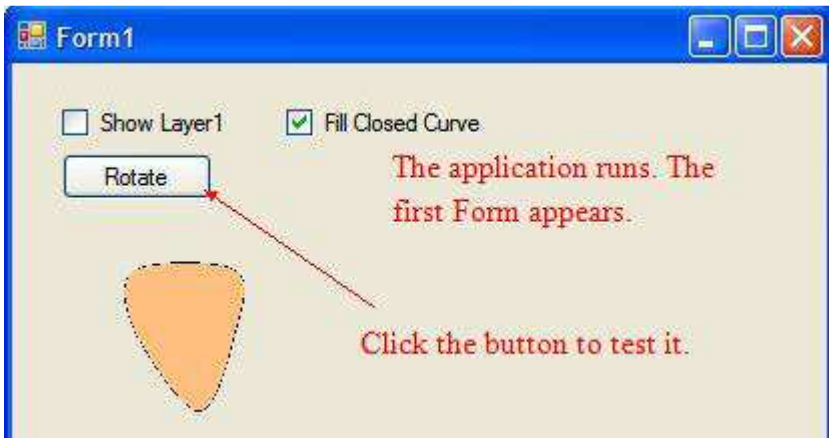
Select the action and click Next.



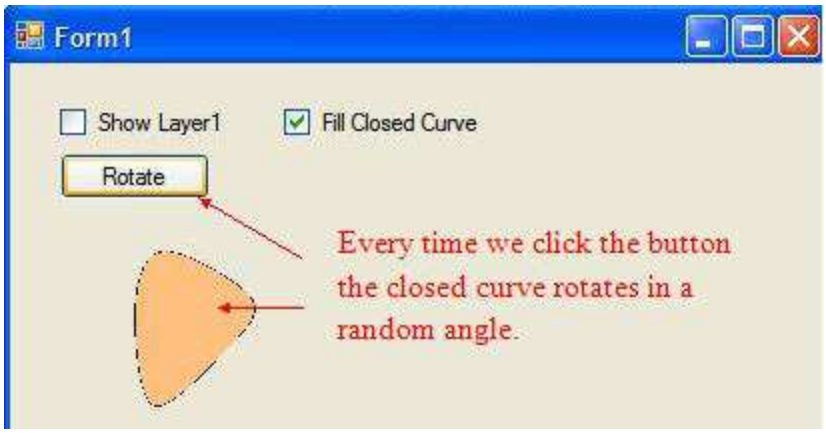
We may test the application now.



The application runs. The first Form appears. Click the button to test it.

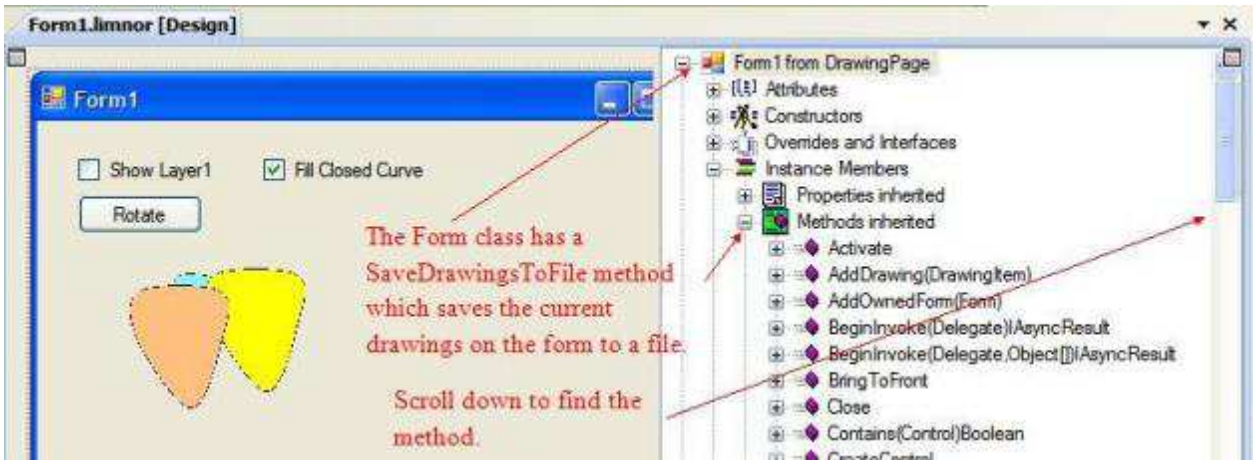


Every time we click the button the closed curve rotates in a random angle.

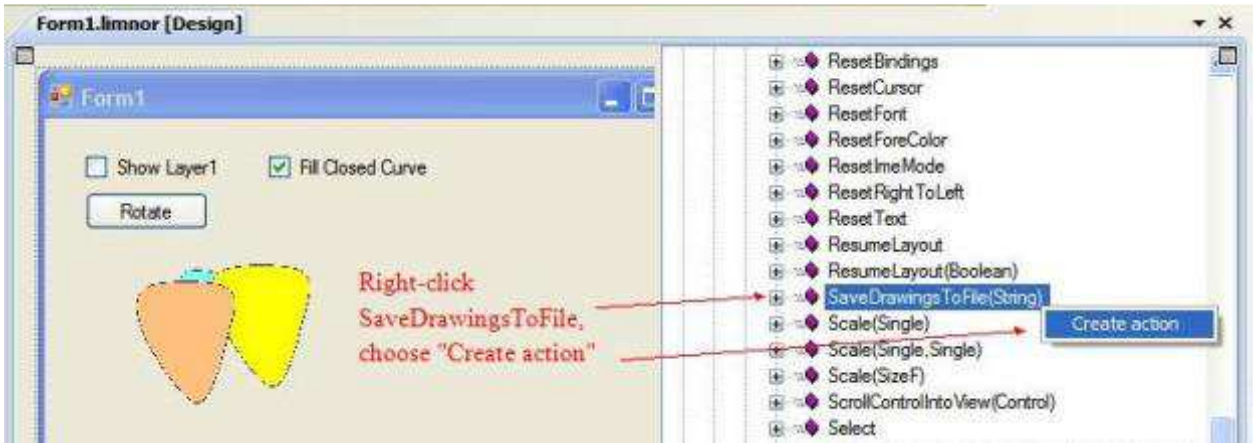


6 Save drawings to xml file

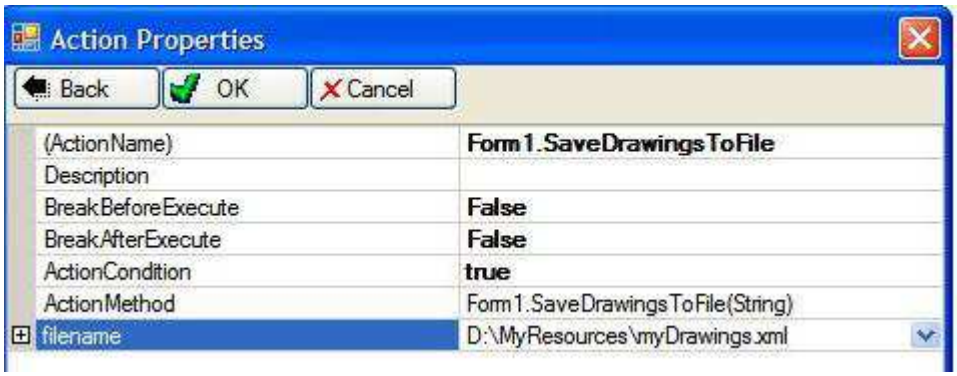
The Form class has a SaveDrawingsToFile method which saves the current drawings on the form to a file.



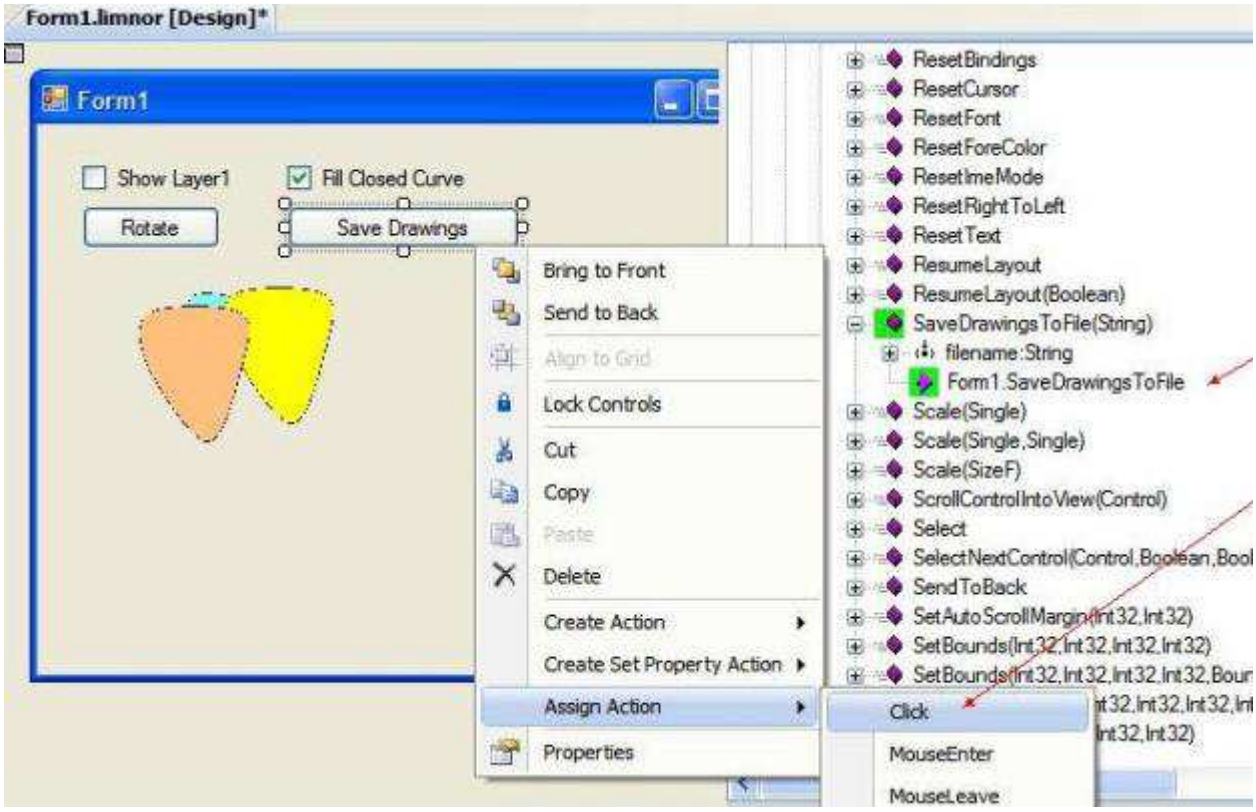
Right-click SaveDrawingsToFile, choose "Create action"



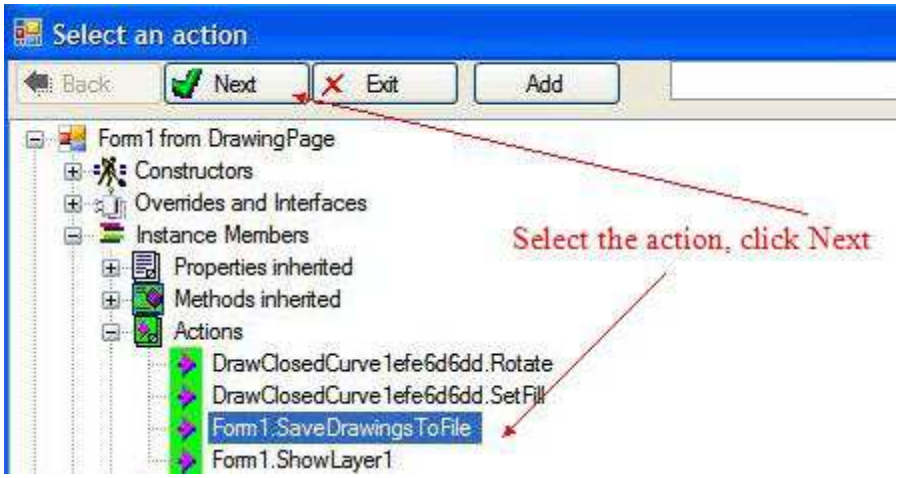
For this sample, we simply use a constant file name for this action.



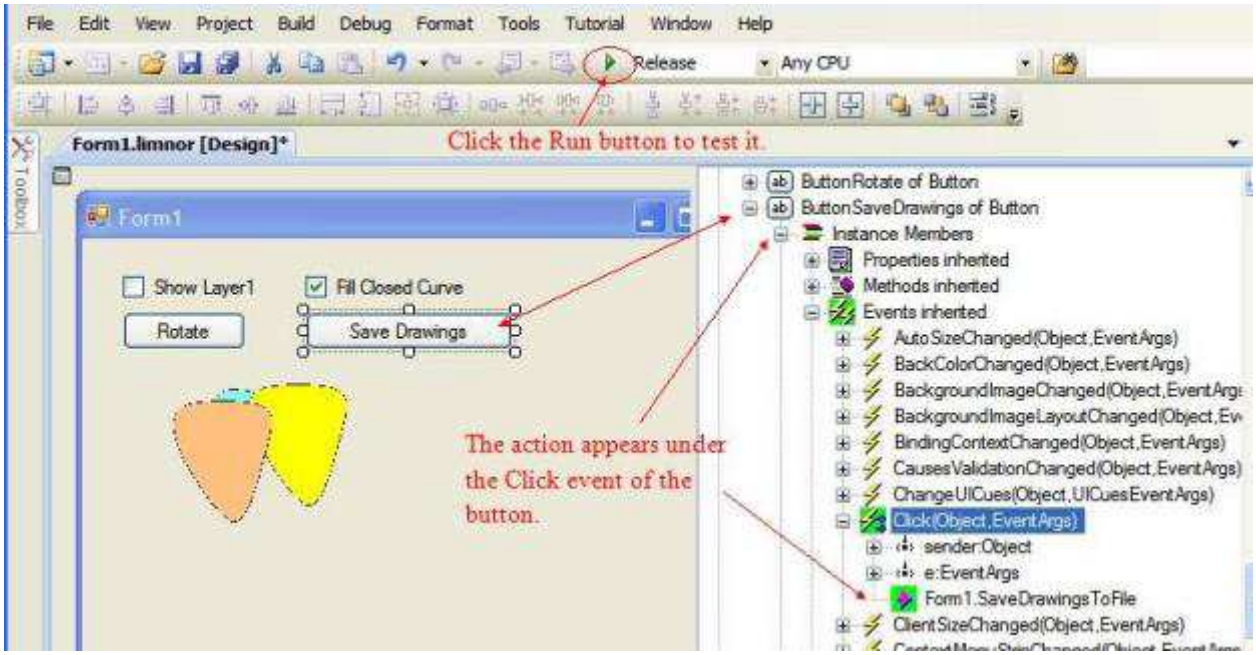
Assign the action the Click event of a button



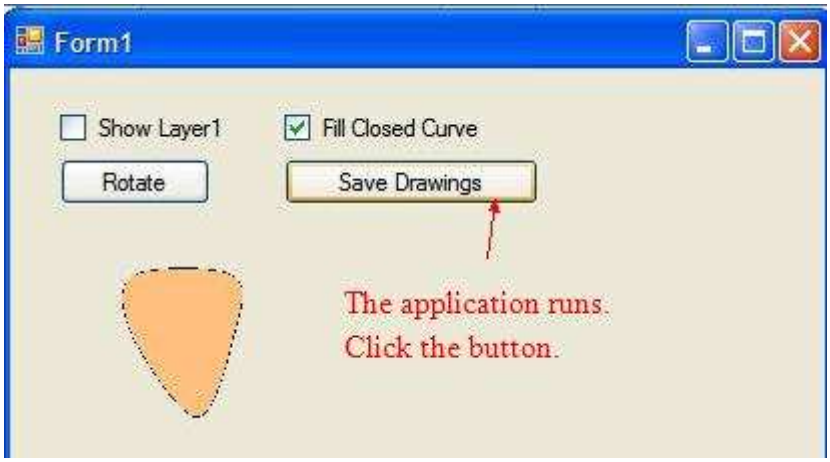
Select the action. Click Next.



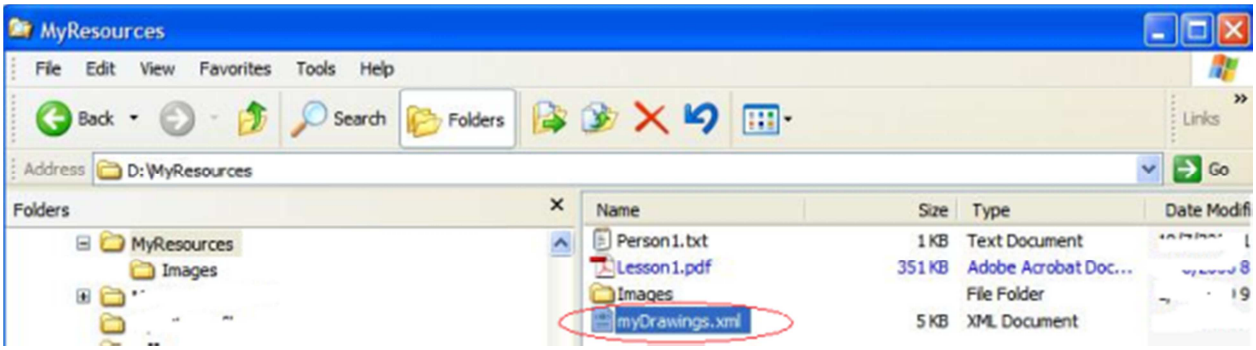
We may test the application now.



The application runs. Click “Save Drawings”:

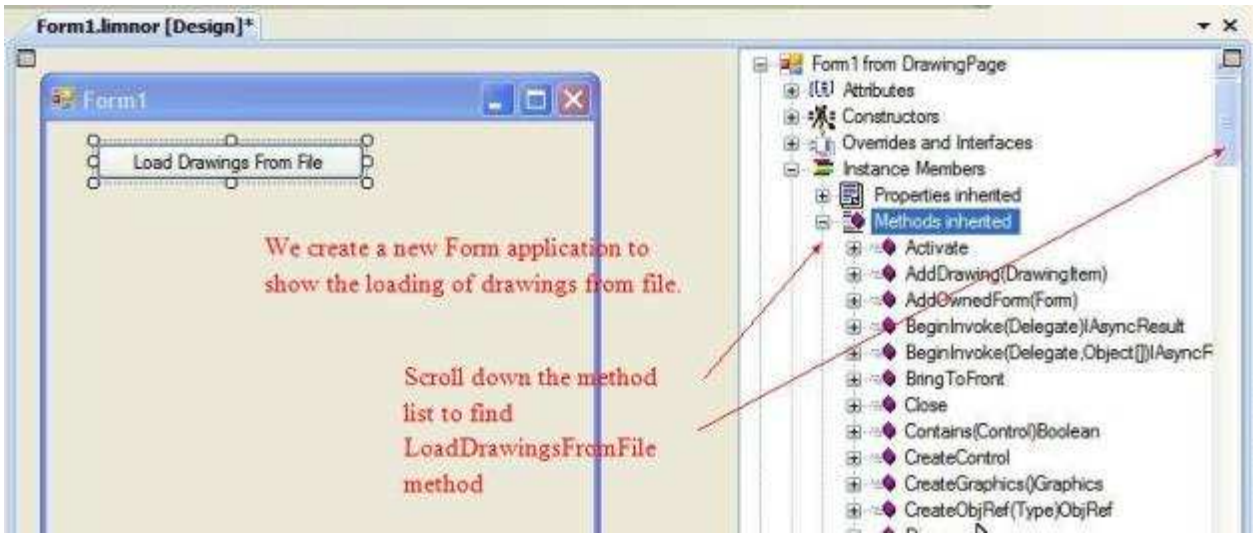


The file is created. This file can be used by other applications.

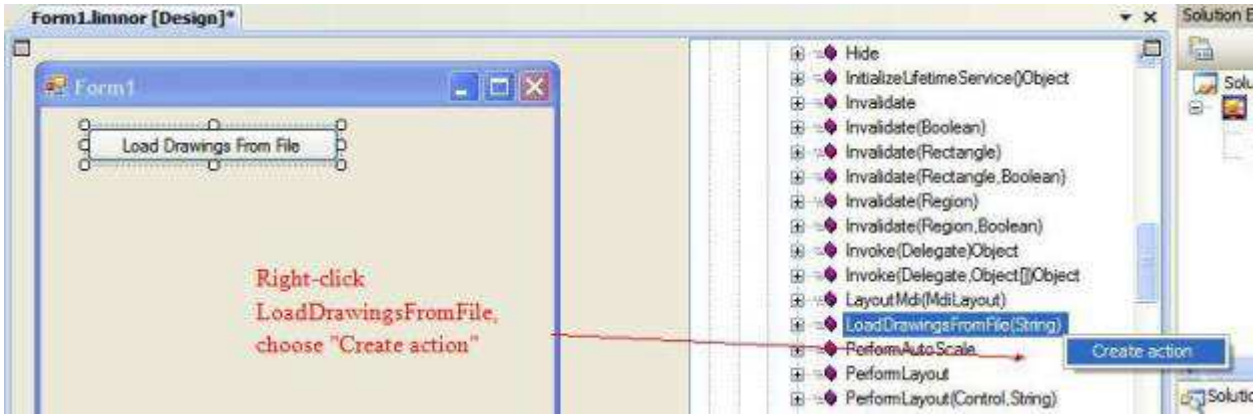


7 Load drawings from xml file

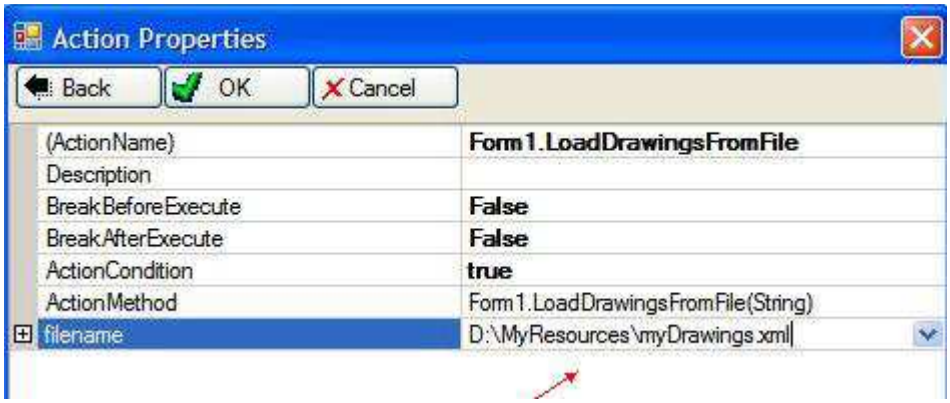
We create a new Form application to show the loading of drawings from file. Scroll down the method list to find LoadDrawingsFromFile method.



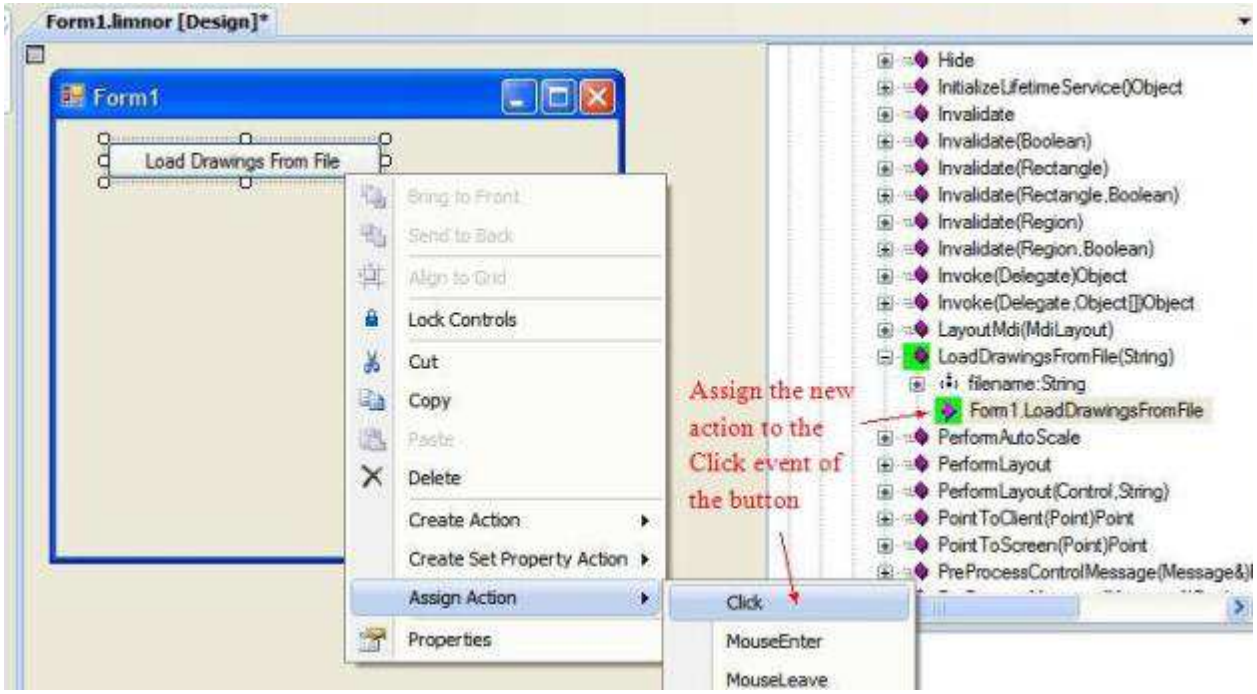
Right-click LoadDrawingsFromFile, choose "Create action"



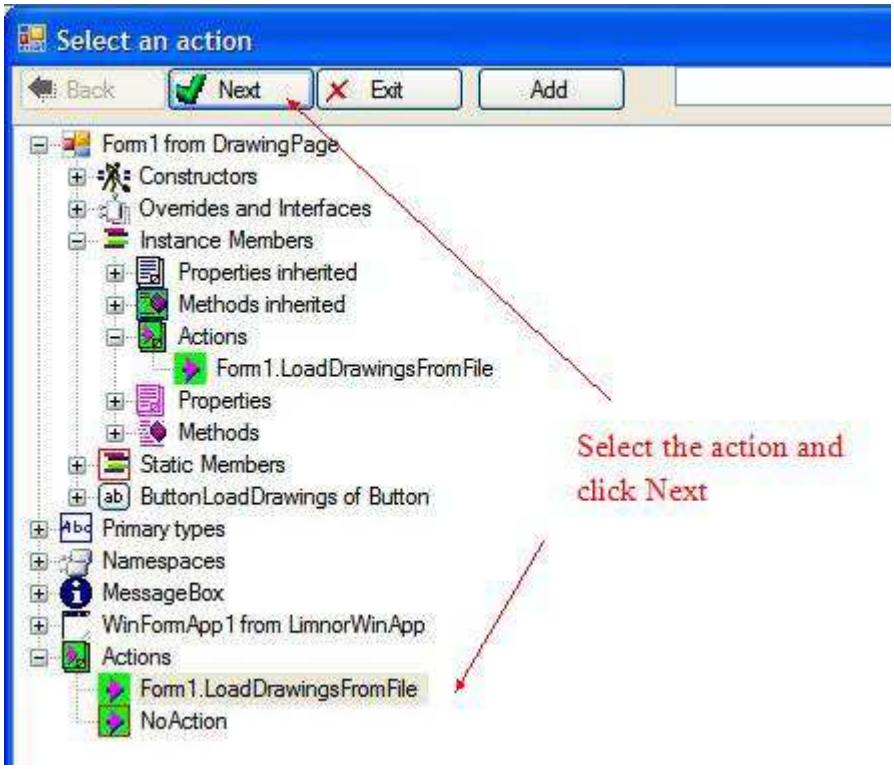
Specify the file name:



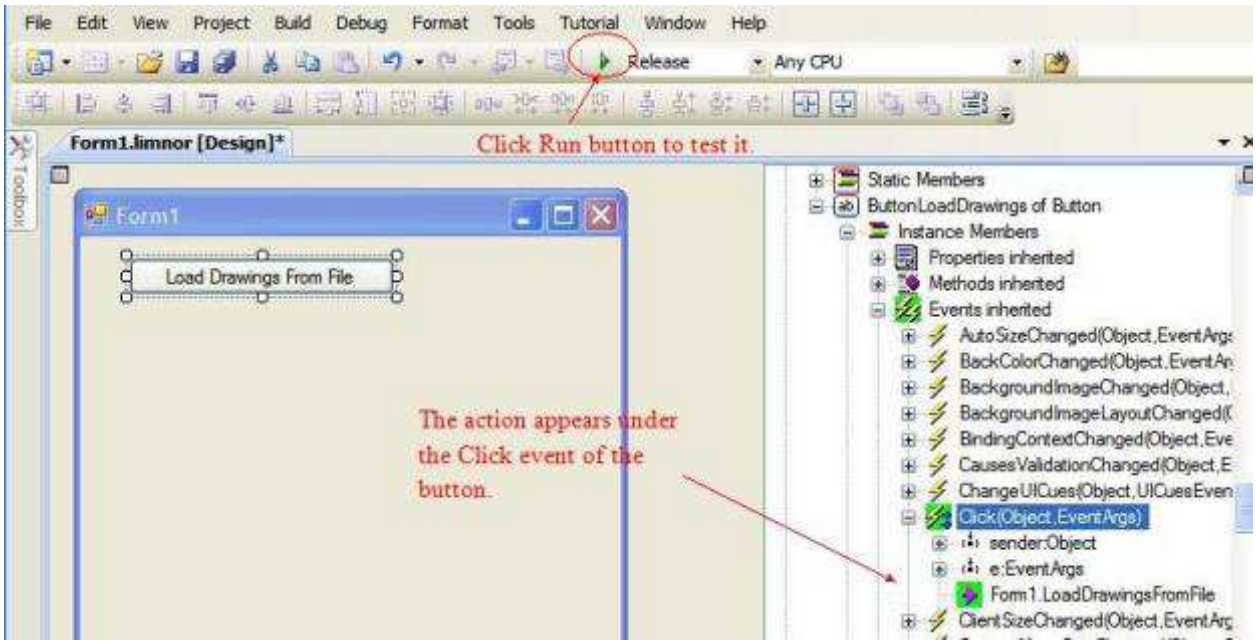
Assign the new action to the Click event of the button



Select the action. Click Next.

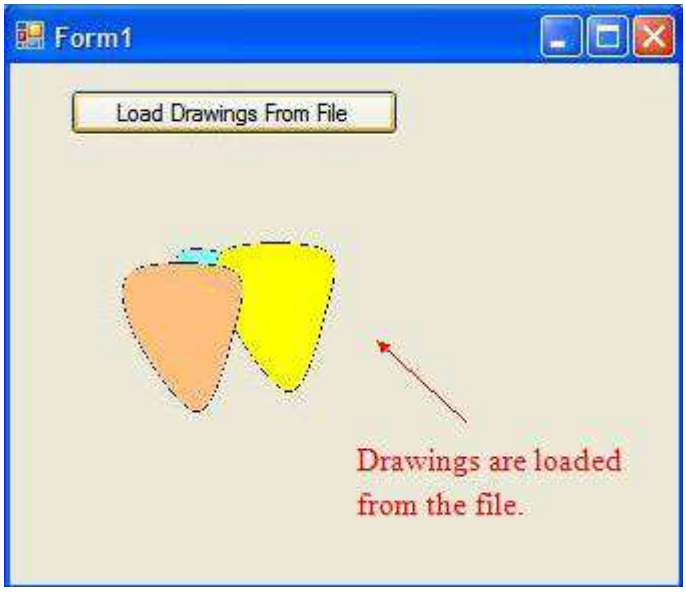


We may test the application now.



Click the button to load drawings from the xml file.

The drawings are loaded from the Xml file



8 Continuous drawing and animation effect

8.1 What is continuous drawing

Keep changing drawings and repainting screen may create animation effects. The Users' Guide for screensaver describes an example of moving text. A text keeps moving from left to right on the screen again and again. See http://www.limnor.com/support/Limnor%20Studio%20-%20User%20Guide%20-%20ScreenSaver.pdf

This feature is enabled by setting property ContinuousDrawingStarted to True.

It works in the following way.

1. Method OnPrepareNextPaint is executed. You should override this method to modify drawings. All your drawing ideas and innovations are implemented in this method.
2. The system draws all the drawings.

3. The system waits for a time indicated by ContinuousDrawingInterval property, in milliseconds. The waiting is in another thread. So, do not worry about hurting performance of the application.
4. The system goes back to step 1.

All your animation ideas and innovations are implemented in method OnPrepareNextPaint.

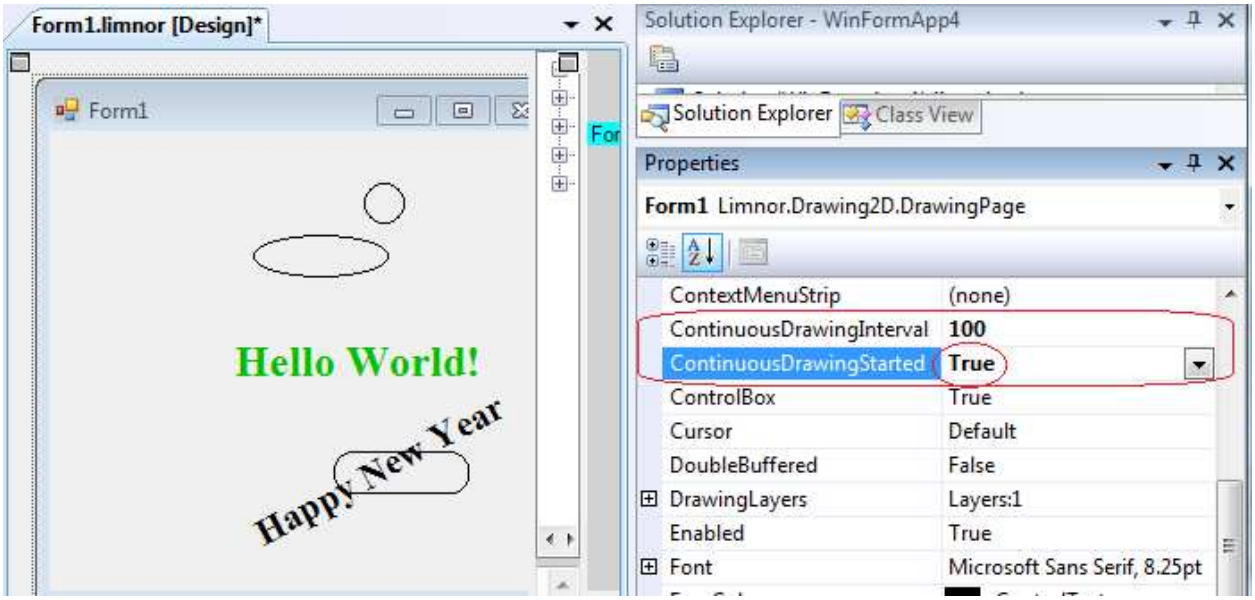
In the moving text example, in method OnPrepareNextPaint the Left property of the Text is increased by some pixels to achieve an effect of moving from left to right. So, every time OnPrepareNextPaint is executed the text moves to right by some pixels. In OnPrepareNextPaint there is also an action to move the text to left side of the screen when the text is moved over the right edge of the screen.

Here we show another example of continuous drawing. A drawing has a Randomize method which randomly modifies the drawing. We add some drawings to the form at design time. In method OnPrepareNextPaint we go through every drawing and execute Randomize action.

We will also show a skill of going through all objects in a collection or an array.

8.2 Enable continuous drawing

Set ContinuousDrawingStarted to True to enable continuous drawing.

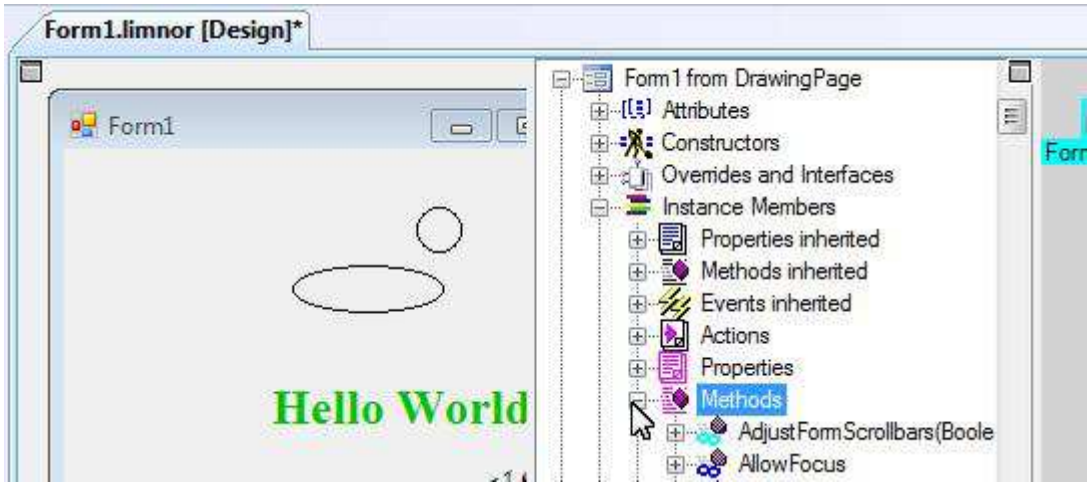


In this sample, we set ContinuousDrawingInterval to 100. Every 0.1 second the form will do a repaint.

8.3 Design continuous drawing

8.3.1 Override OnPrepareNextPaint

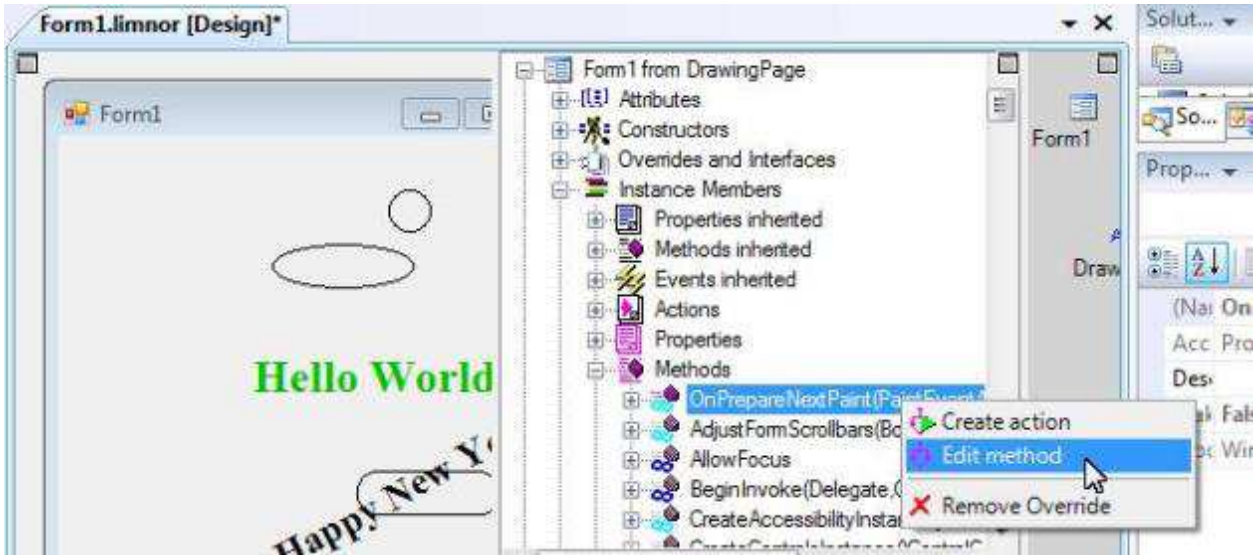
Continuous drawing design is implemented in method OnPrepareNextPaint. Expand Methods node:



Scroll down to find method OnPrepareNextPaint.

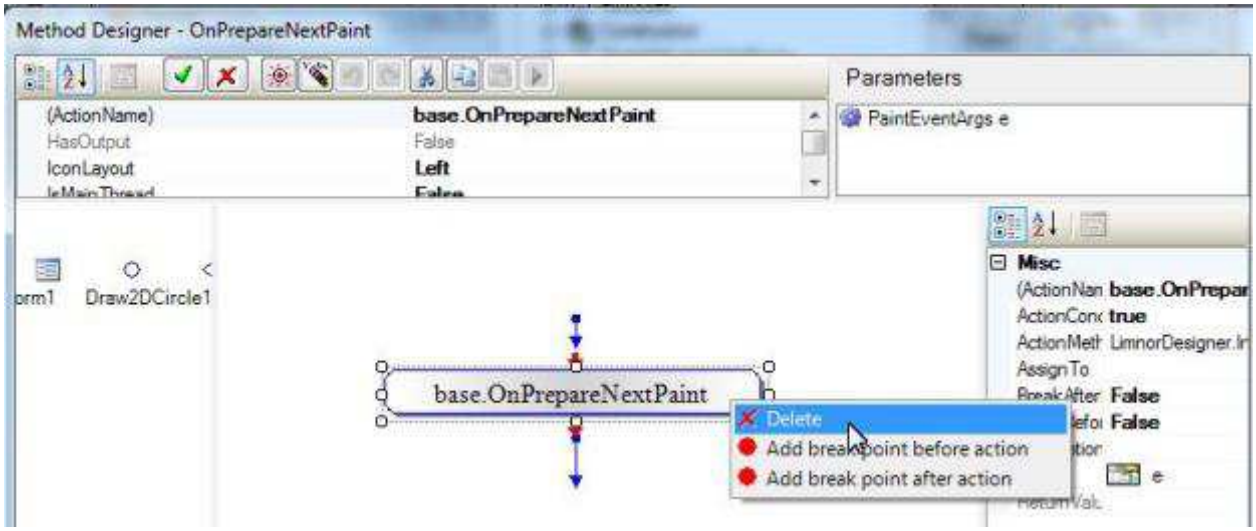
Right-click method OnPrepareNextPaint, choose “Override”:

Overridden methods appear at the top of the list. Right-click overridden OnPrepareNextPaint and choose “Edit method”:



The Method Editor appears.

By default an overridden method contains an action to execute the method implemented by the base class. In our example, we may remove it:

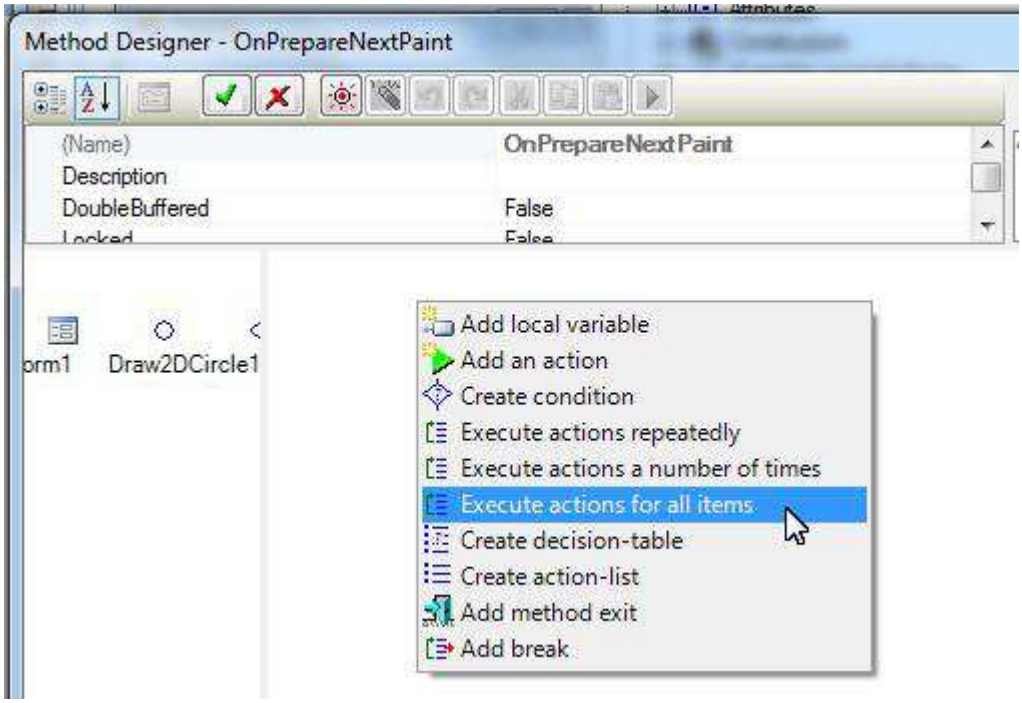


For details of editing a method using the Methd Editor, see Users' Guide
http://www.limnor.com/support/Limnor%20Studio%20-%20User%20Guide%20-%20Part%20V.pdf

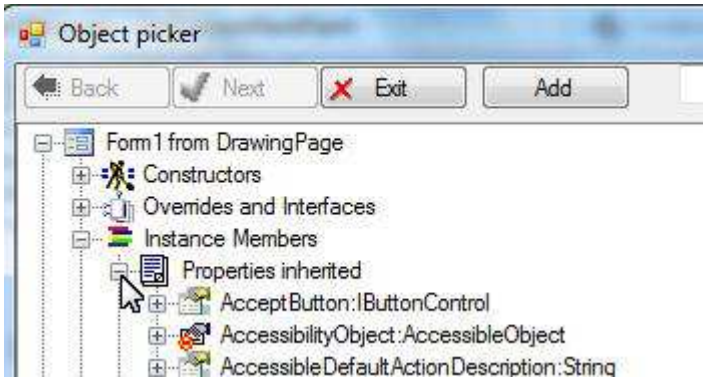
Below we show examples of using the Method Editor.

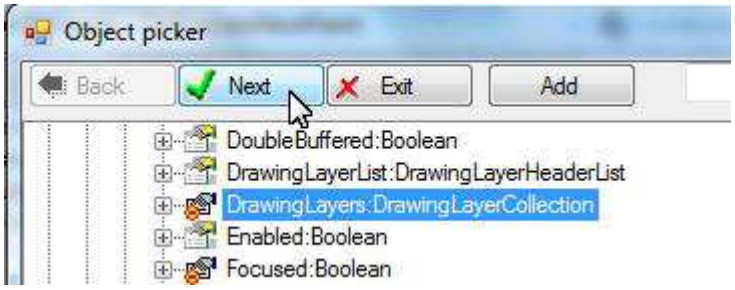
8.3.2 Method example: going through all drawings

Right-click the Action-Pane, choose

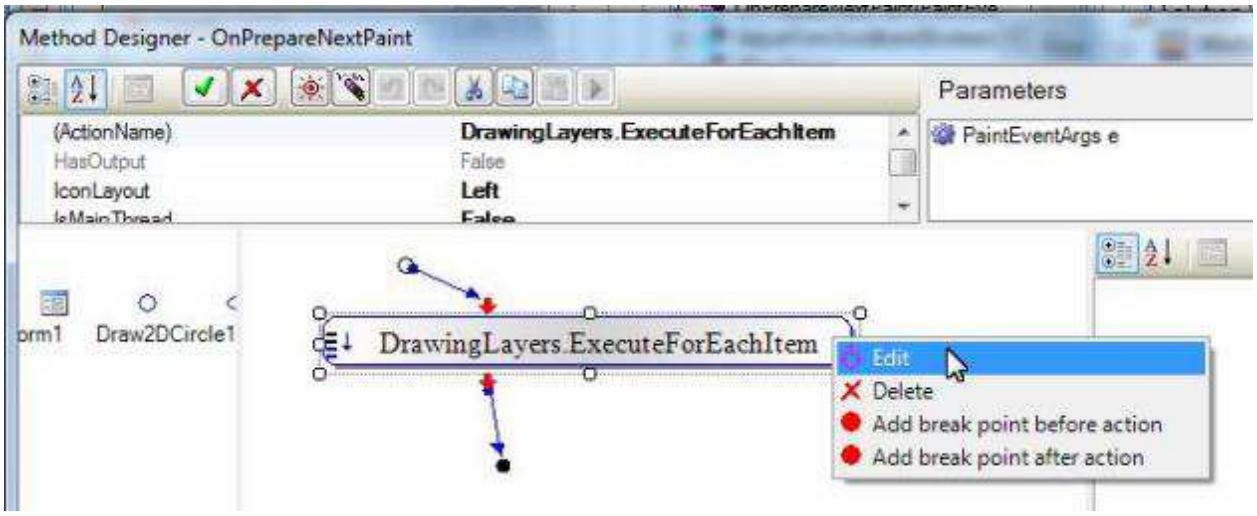


A dialogue box appears for selecting an item holder. An item holder can be an array, a list, or a collection. In our example, we want to select the DrawingLayers property of the form:

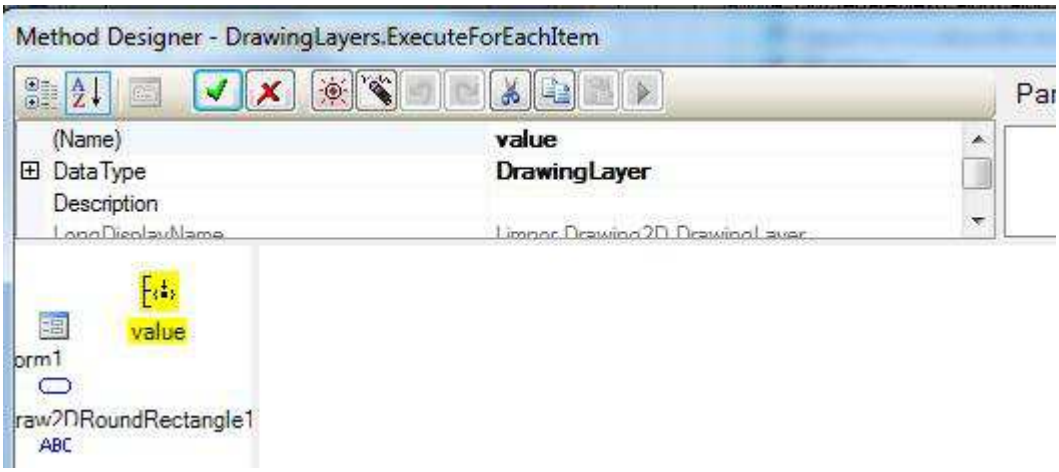




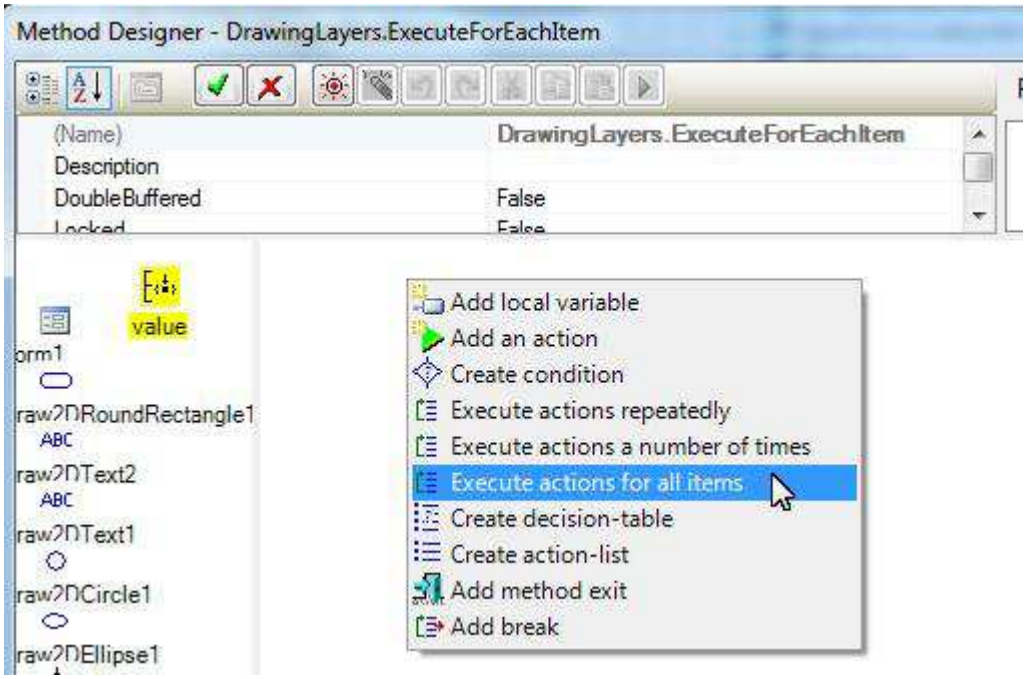
DrawingLayers property holds all drawing layers of the form. This action goes through each drawing layer. Right-click the action; choose “Edit” to launch a new Method Editor to add actions:



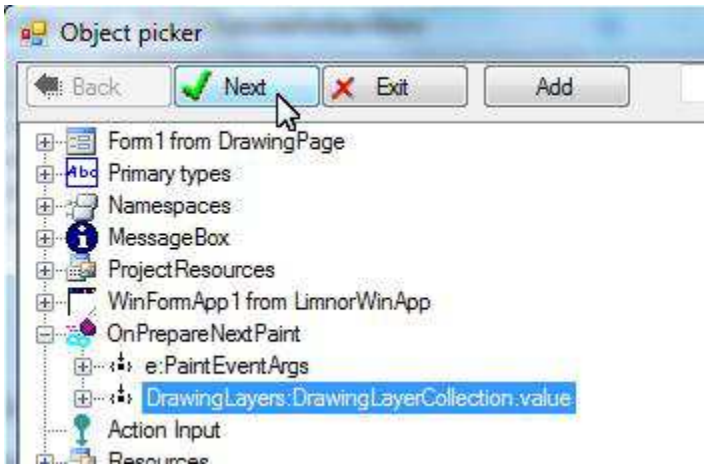
Note that there is a “value” icon in the new Method Editor. It represents the drawing layer as one item for this “Execute actions for all items” action for the DrawingLayers property.



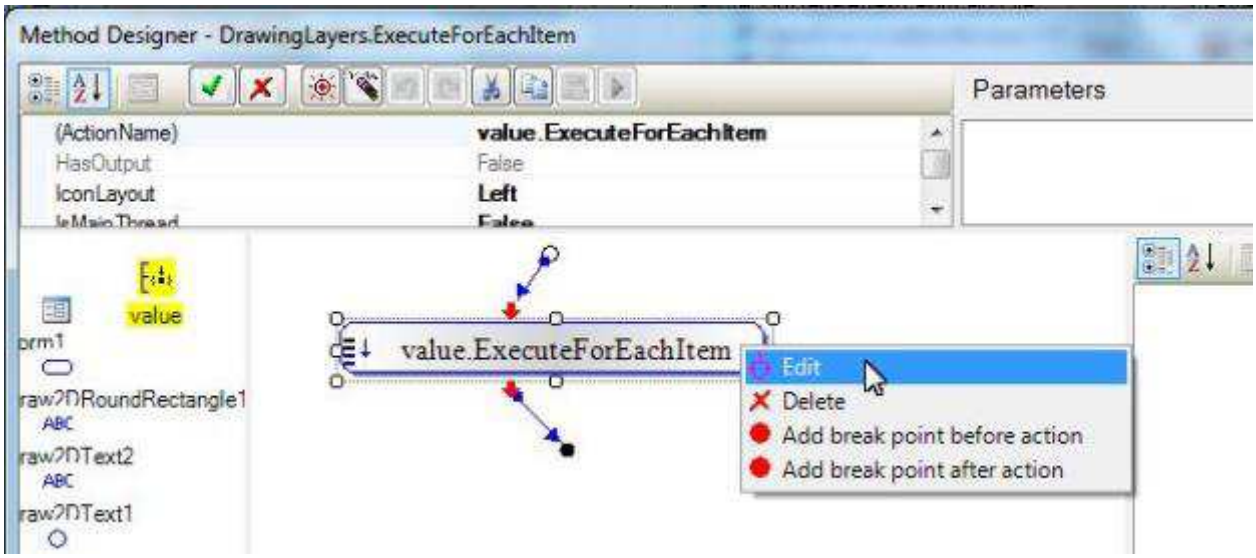
A DrawingLayer is a collection of drawings. We may use an “Execute actions for all items” to go through all drawings. Right-click the Action Pane and choose “Execute actions for all items”



Choose the “value” (as a DrawingLayer it is a collection of drawings):

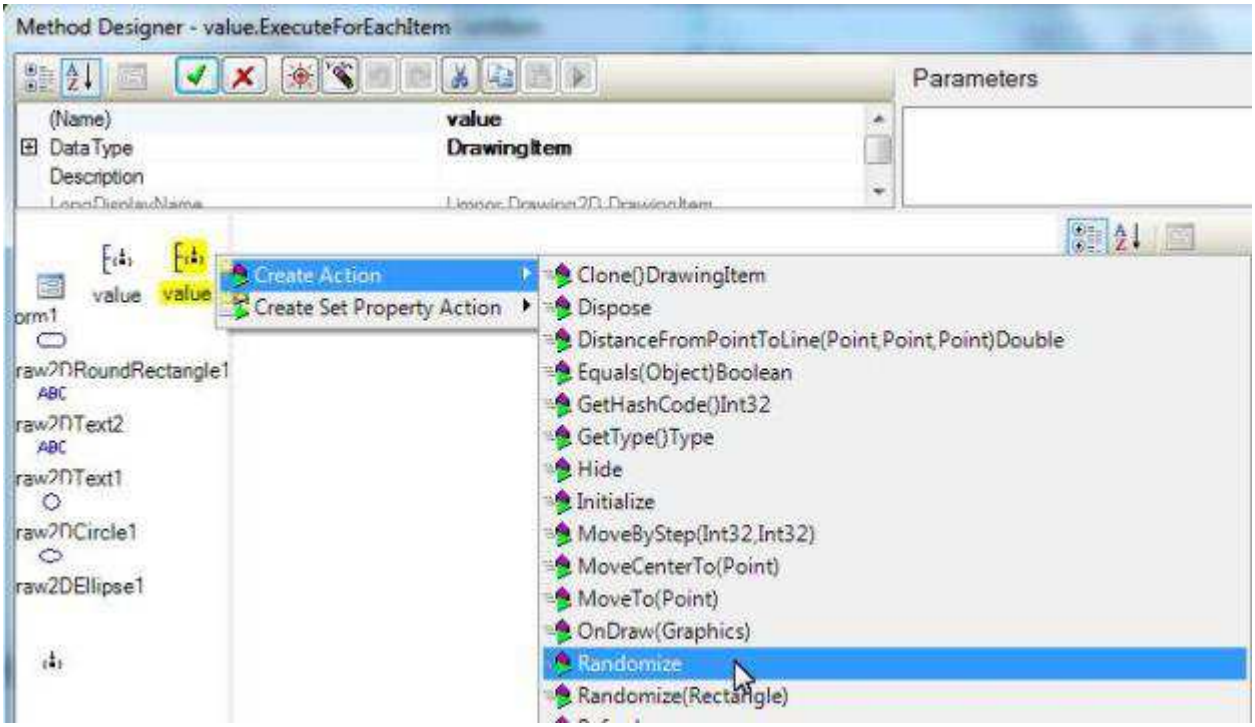


The action is created and appears in the action Pane. Right-click it and choose “Edit” to add actions:

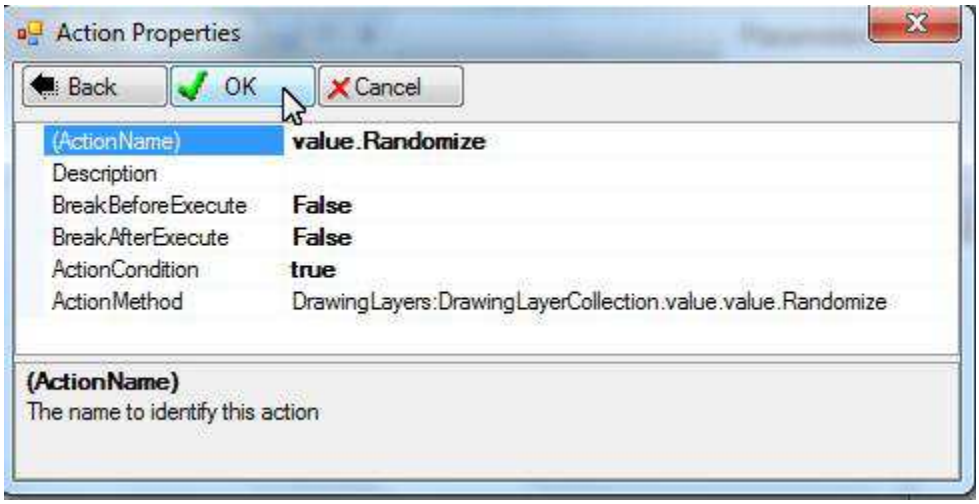


Another Method Editor appears for adding the actions. Note that in this Method Editor there is a “value” icon representing a drawing object and it is the item for this “Execute actions for all items” action for the drawing layer which is represented by the other “value” icon:

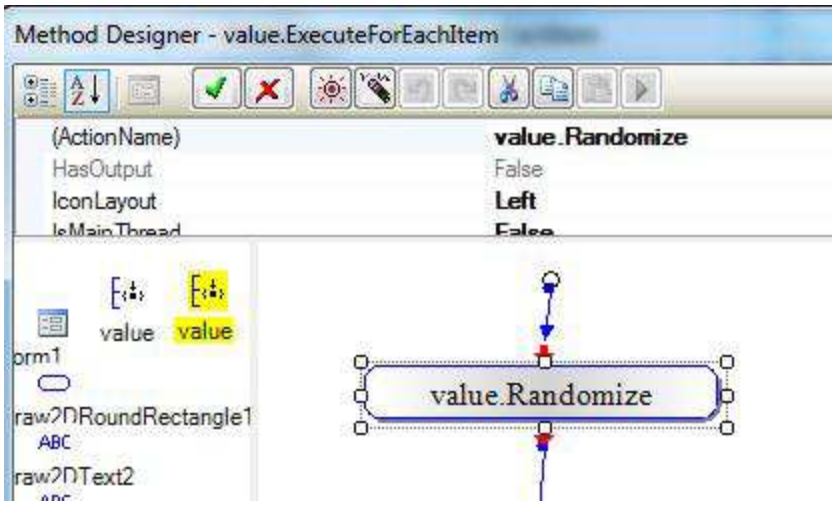
Right-click the drawingItem “value”; select “Create Action”; choose Randomize method:



Click OK:



The action appears in the Action Pane:



By this programming when the method OnPrepareNextPaint executes it goes through all drawing layers; for each drawing layer it goes through all drawings; for each drawing it execute Randomize action.

8.3.3 Method example: processing specific types of collection items

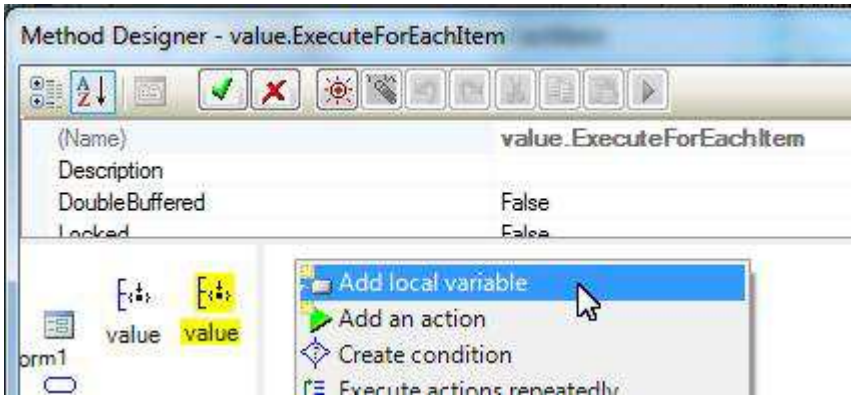
In the previous example we used "value" object to create a Randomize action. This can be done because all kinds of drawing objects have a Randomize method.

Suppose we want to set the angle of all text drawings to 30 degree. This can be done by setting the TextAngle property of text drawing.

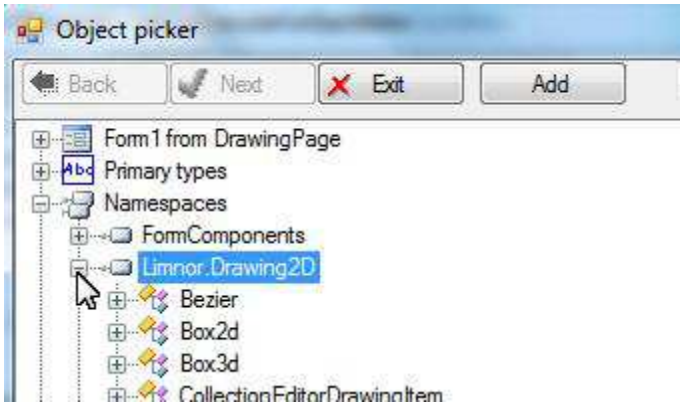
But we cannot do it using the "value" icon because the "value" icon represents a generic drawing. A generic drawing does not have a TextAngle property.

We can do it by creating a text drawing variable and use a "Use it as" action to assign "value" icon to the text drawing.

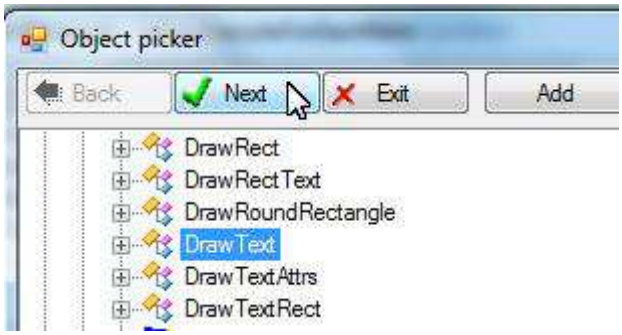
Right-click the Icon Pane or the Action Pane; choose "Create local variable":



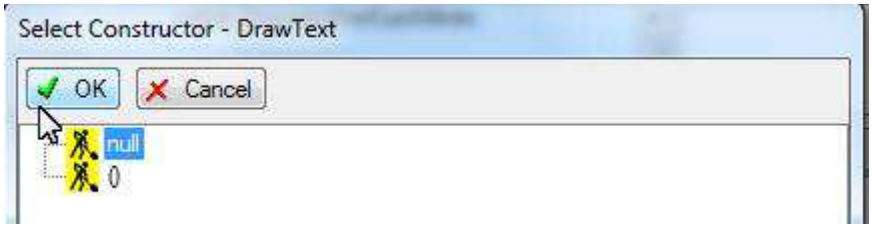
Expand the Limnor.Drawing2D namespace:



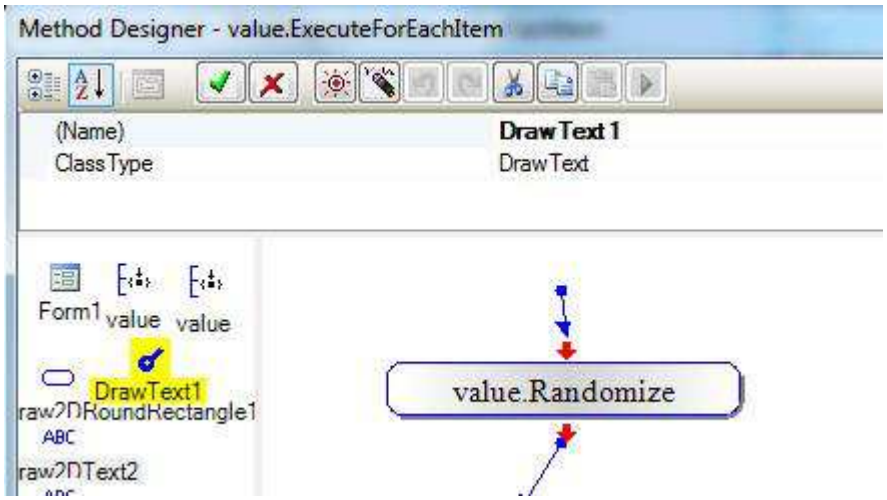
Select DrawText:



Select null because we do not want to create a new drawing:

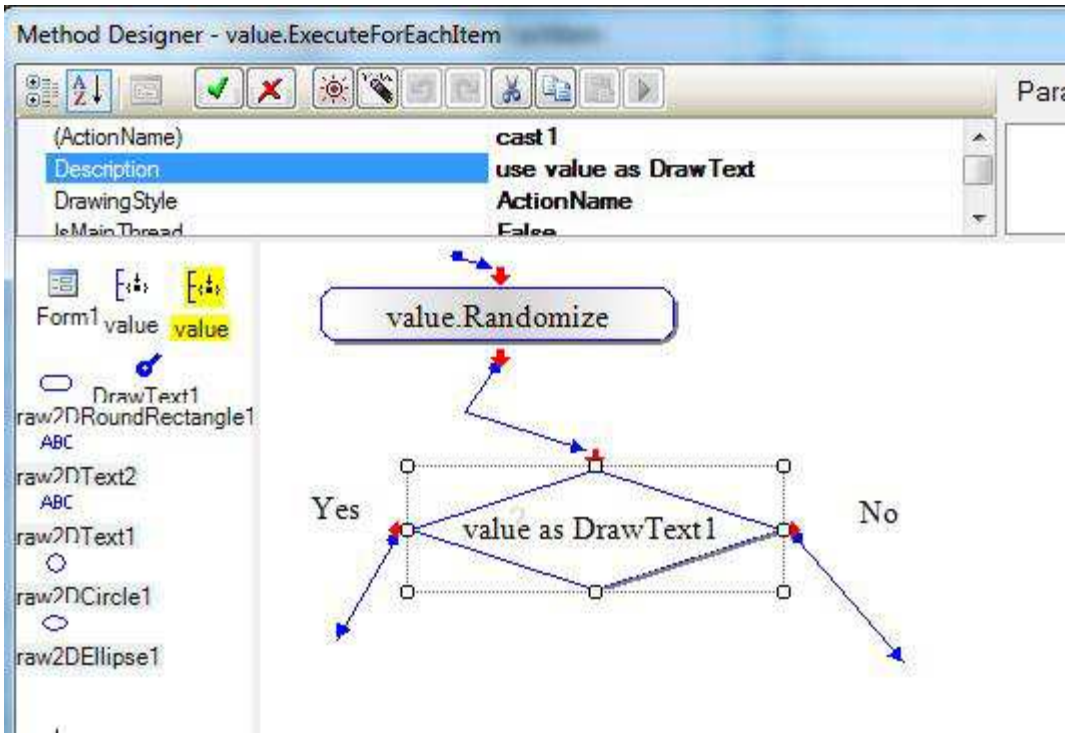


The variable appears in the Icon Pane:



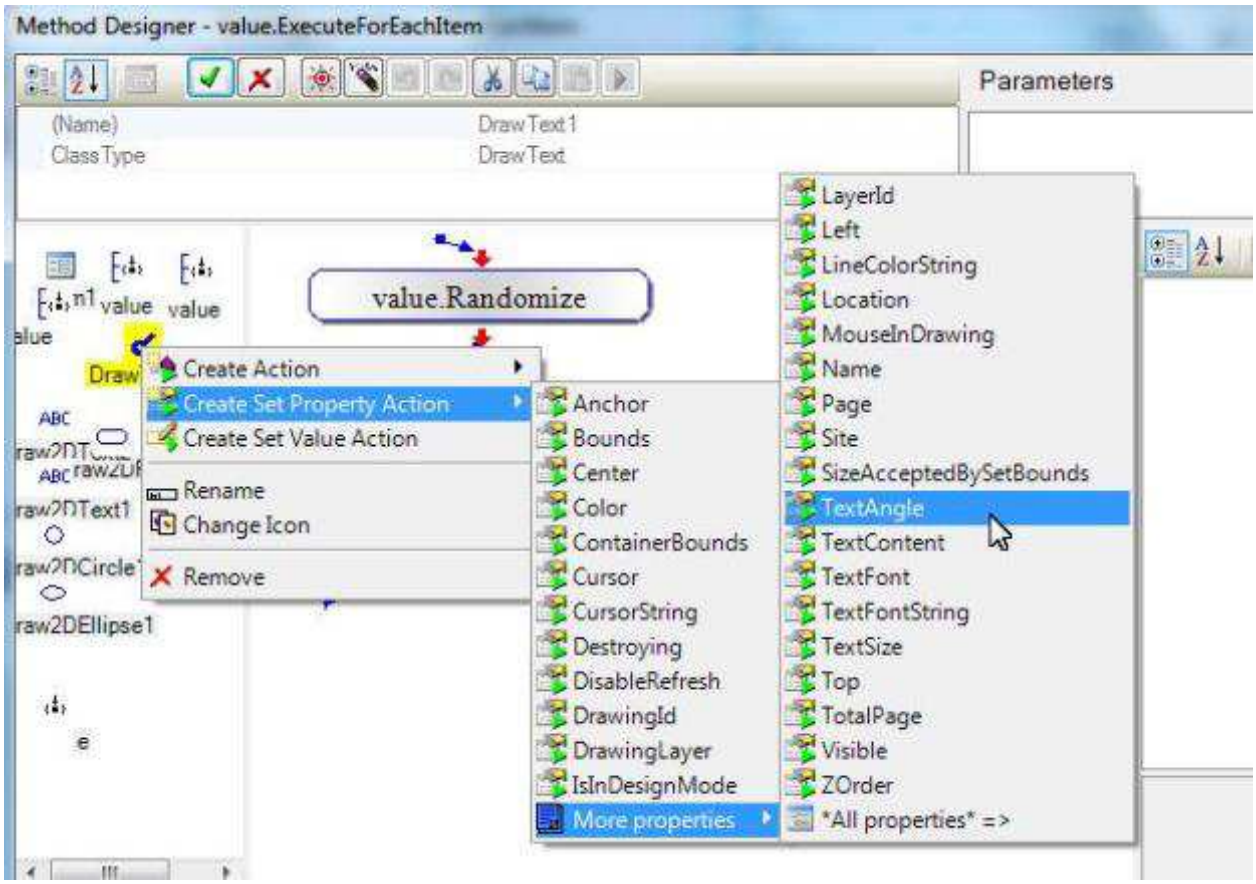
Right-click the “value” icon for the drawing item. The new variable appears under a “Use it as” menu:

Select the variable under the “Use it as” menu. A cast-action appears. Link it to the existing action:

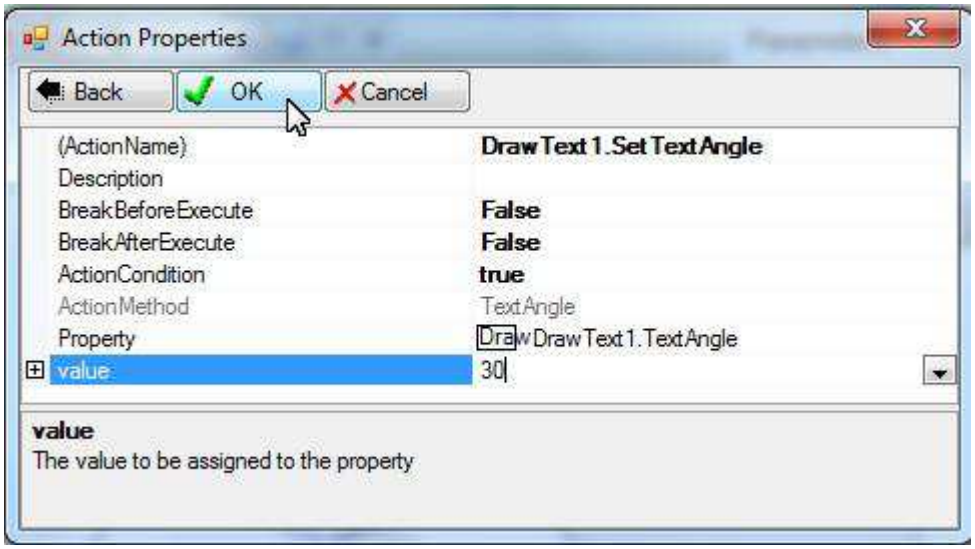


This action has two outputs. “Yes” port indicates that “value” is a DrawText object and it is assigned to the variable DrawText1. “No” port indicates that “value” is not a DRAWText and the variable DrawText1 is not given a valid value.

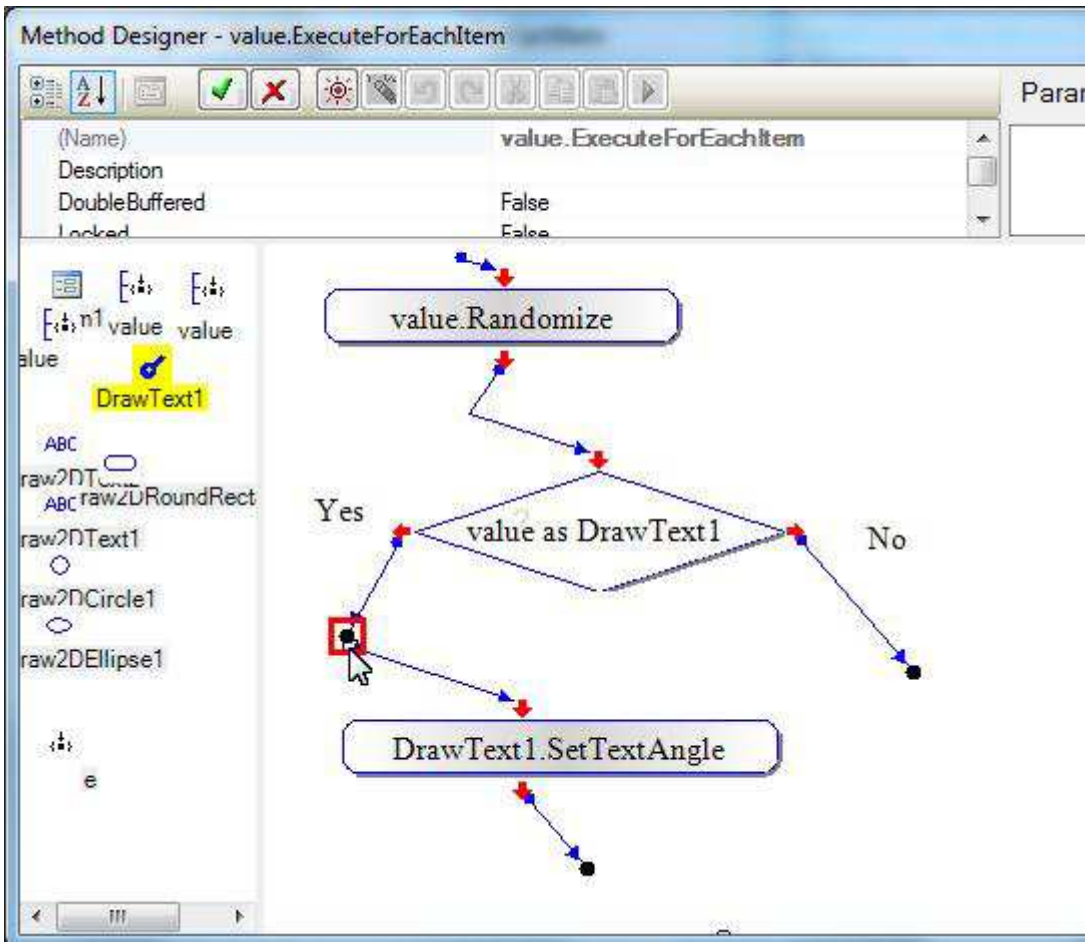
Right-click variable DrawText1; choose “Create set property action”; choose “TextAngle” property.



Give a constant 30 for "value" property of the action:



Link this action to the "Yes" port of the "Use it as" action:



We may close all the Method Editors and test the application now. We may see drawings randomly changing all the time but the two text drawings are always drawn at 30 degree.

9 Feedback

Please send your feedback to support@limnor.com

