Use Generic Types and Native Libraries

Contents

Introduction

To create a standalone Windows program, Limnor Studio uses Microsoft .Net Framework libraries, or called “managed libraries”. To use “unmanaged libraries”, usually created in C/C++, managed libraries

are created to wrap the unmanaged libraries. For example, Emgu CV is such a managed library for Intel Open CV libraries. See http://sourceforge.net/projects/emgucv/

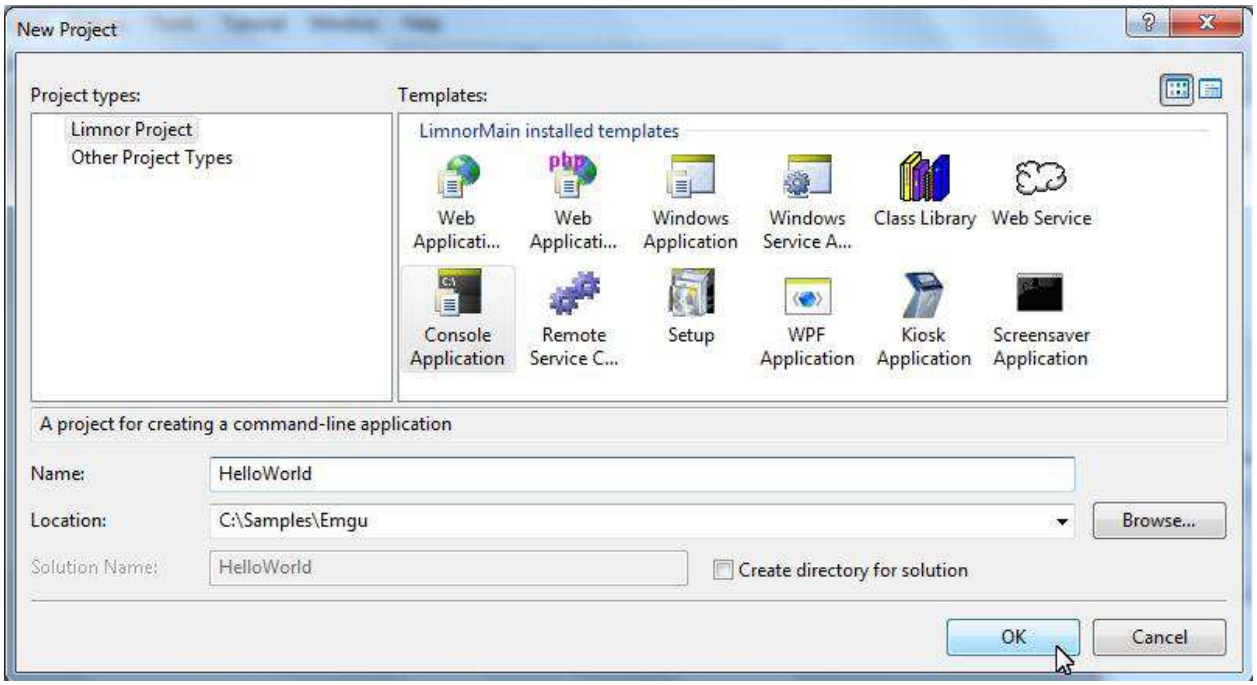
This document re-creates the samples from the Emgu CV tutorial C# samples, using Limnor Studio. It shows how to use generic types in the Emgu CV library.

It shows how to add native libraries files to the project so that those unmanaged library files can be collected into the distribution folder by Limnor Studio. It shows the use of Action Groups to make it easy to show program logic.

Sample 1 – Hello World!

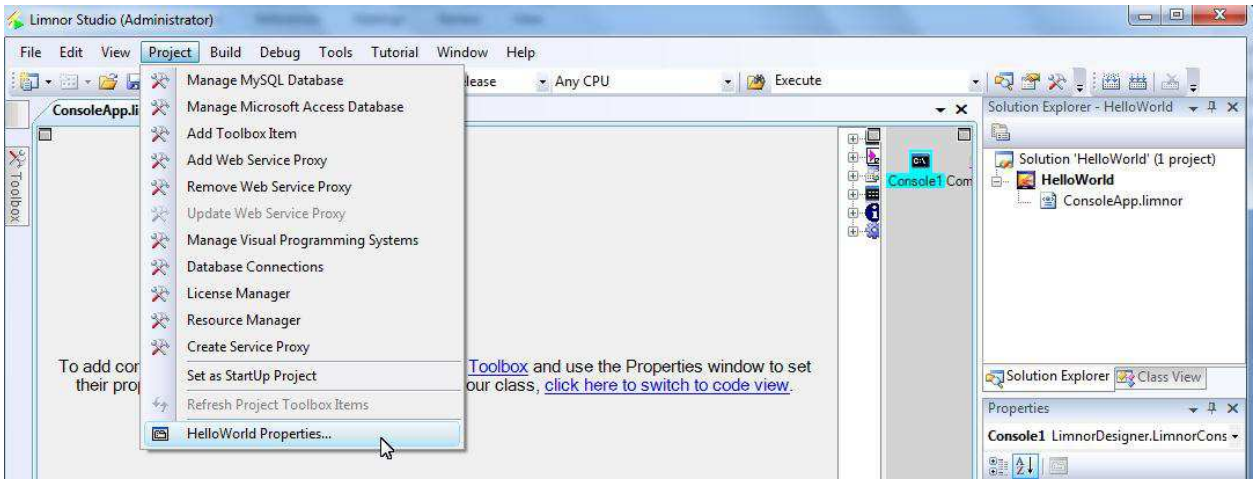
This sample is based on an Emgu sample found at folder "Emgu.CV.Example\HelloWorld" under your Emgu installation folder.

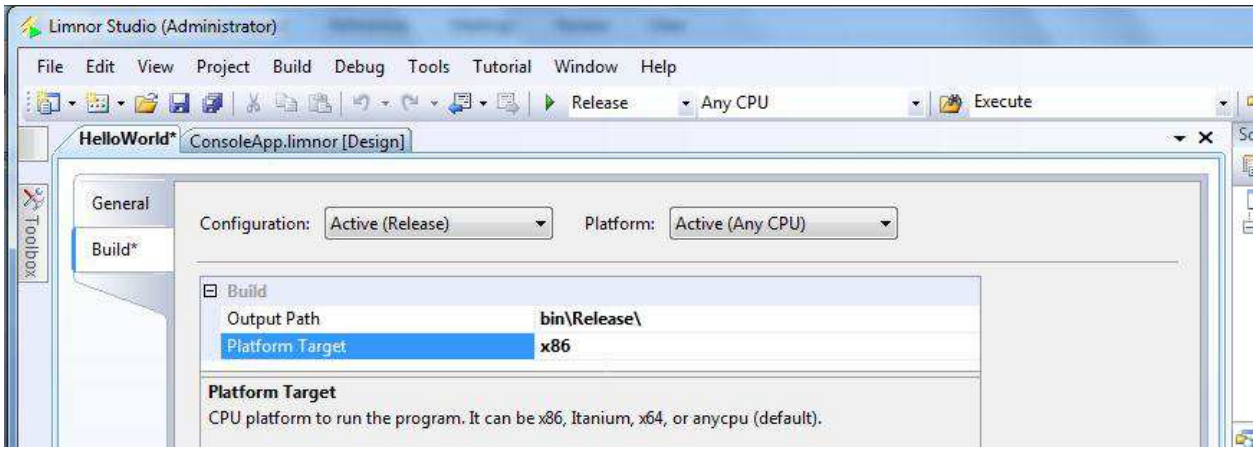
Create a console project:



Target Platform

Set the platform to x86, because we downloaded 32-bit Open CV C++ library.



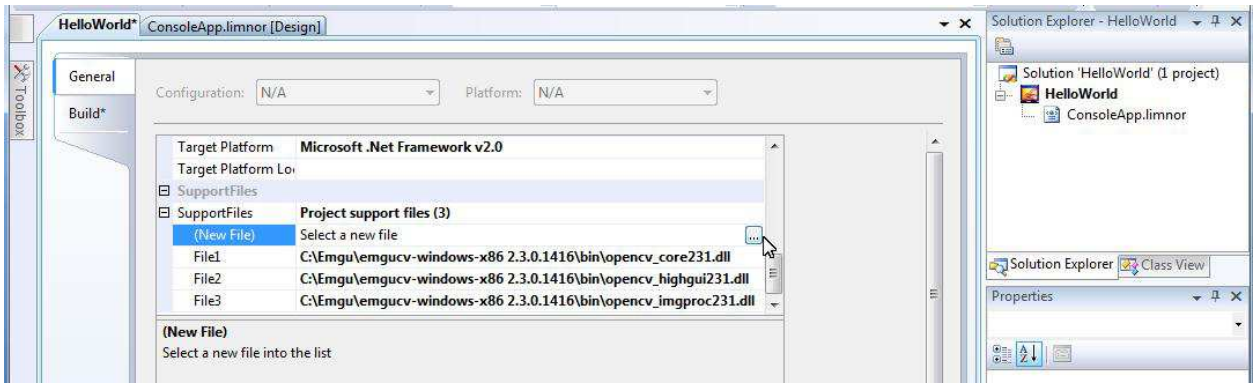


Support files

Because Emgu is a wrapper on unmanaged libraries, we need to tell Limnor Studio which support files the wrapper needs. You need to get such information from the library manufacturer or library documentation. In this sample, the following support files are needed:

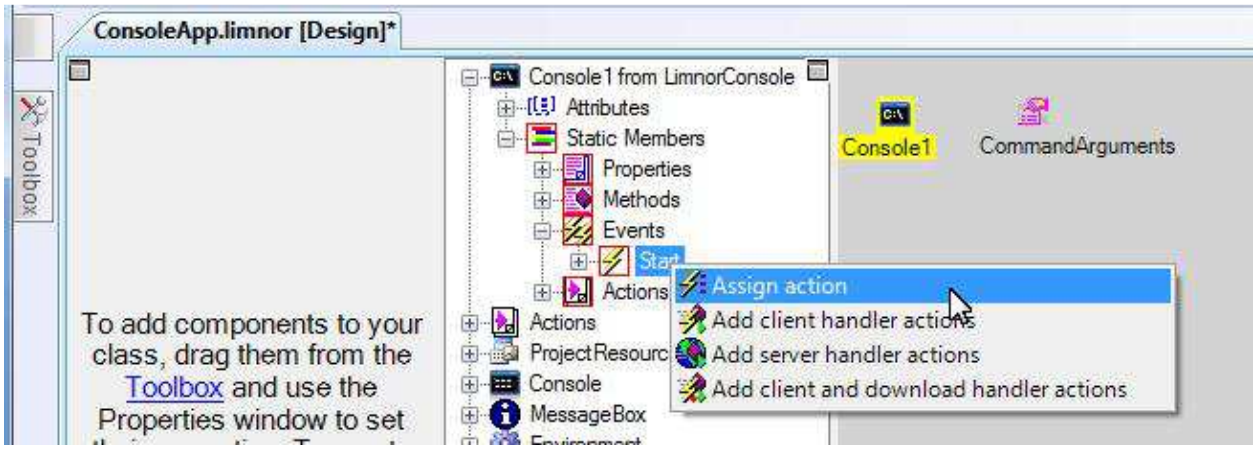
opencv_core231.dll
opencv_highgui231.dll
opencv_imgproc231.dll

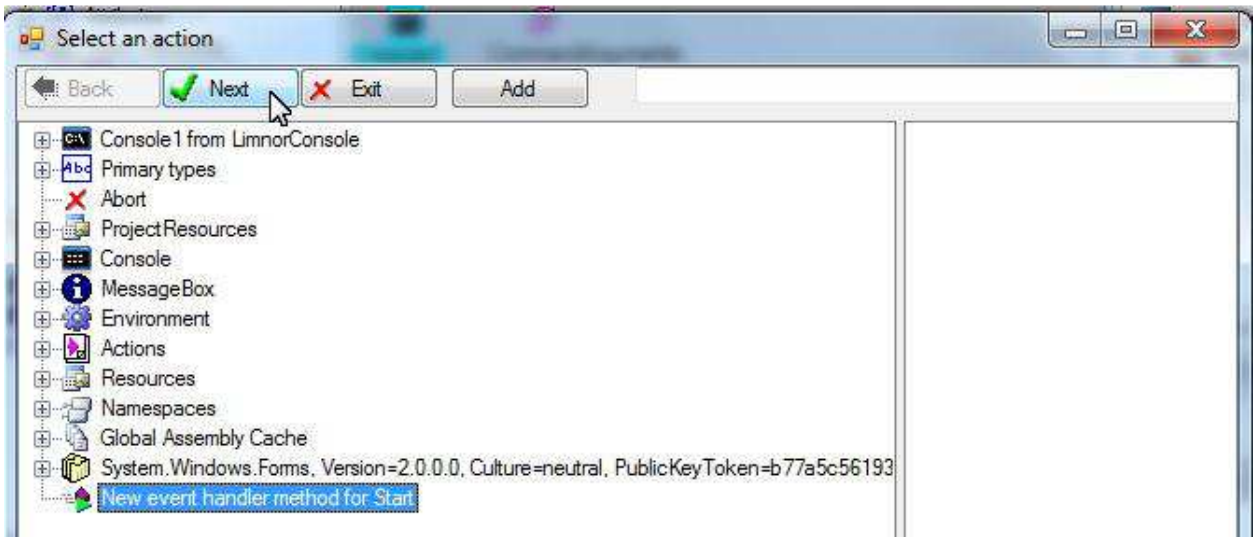
We need to add them to the support file list:



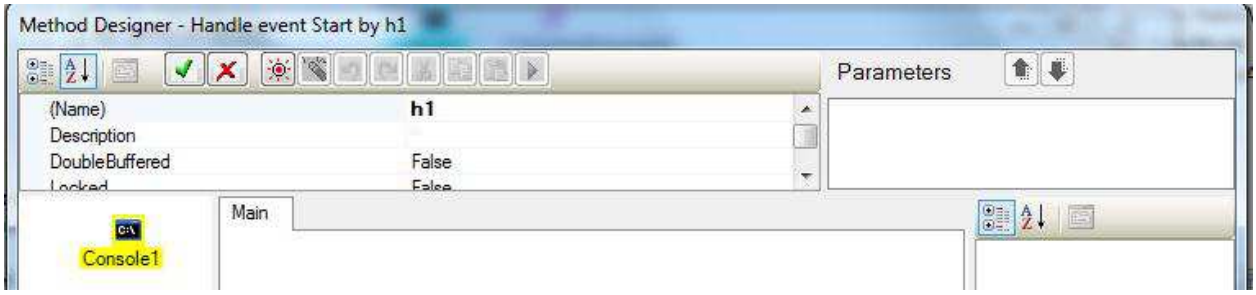
Create method

In a method, we may create variables using the classes from the Emgu CV library. For this sample, we simply create an event handler method:



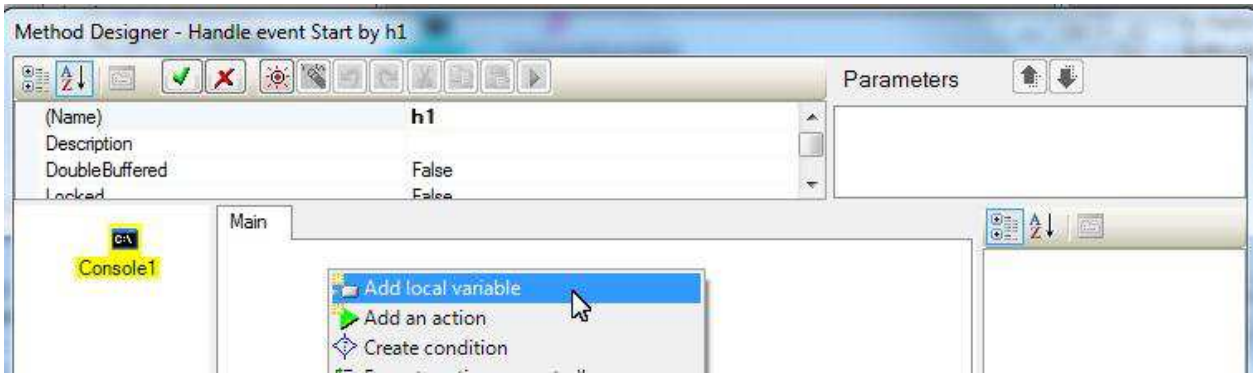


A method editor appears.



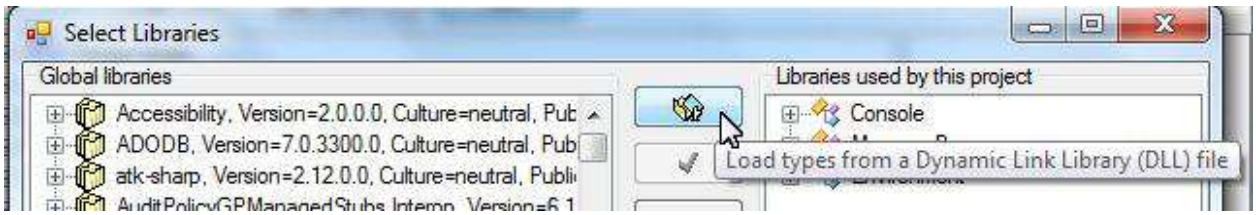
Create Image object

Emgu uses Image object to draw text on the screen. Let's create an image variable.

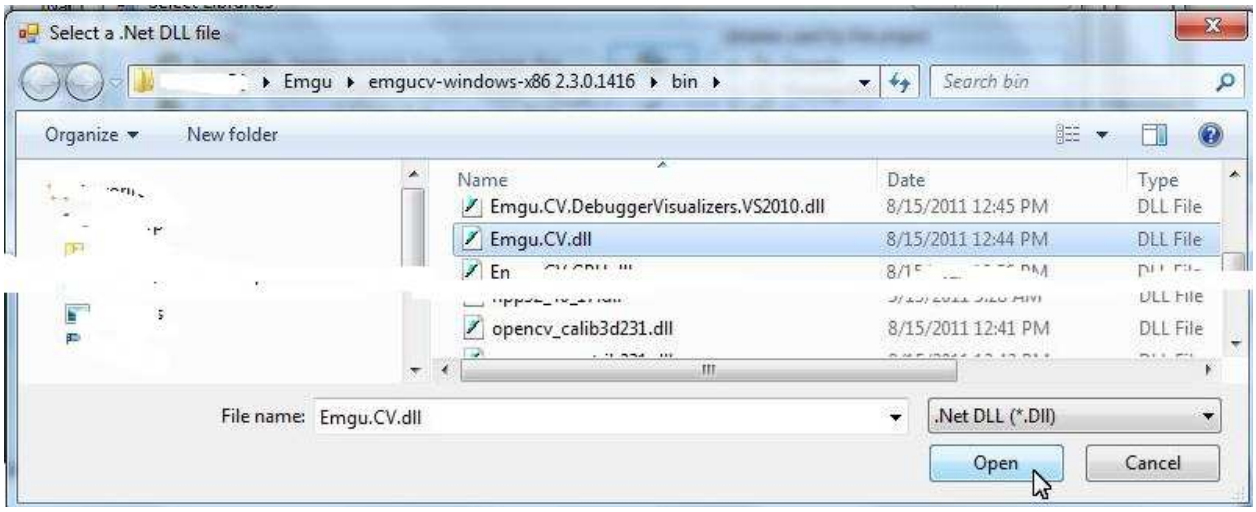


Click Add to add a class from an external library. In this sample, it is the Emgu library.

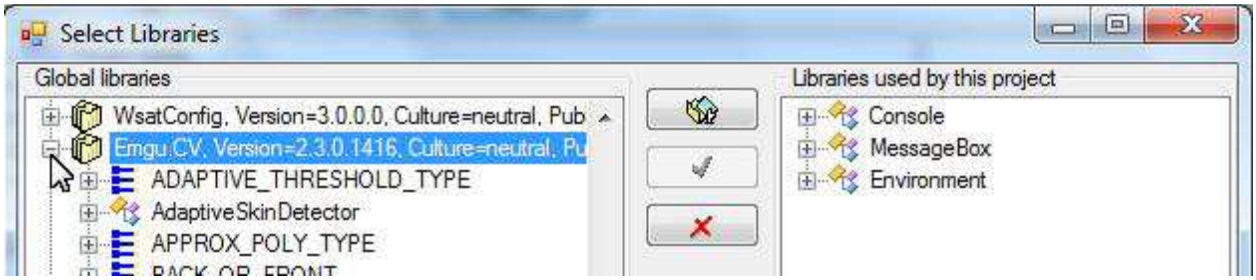
Click  to load the external library:

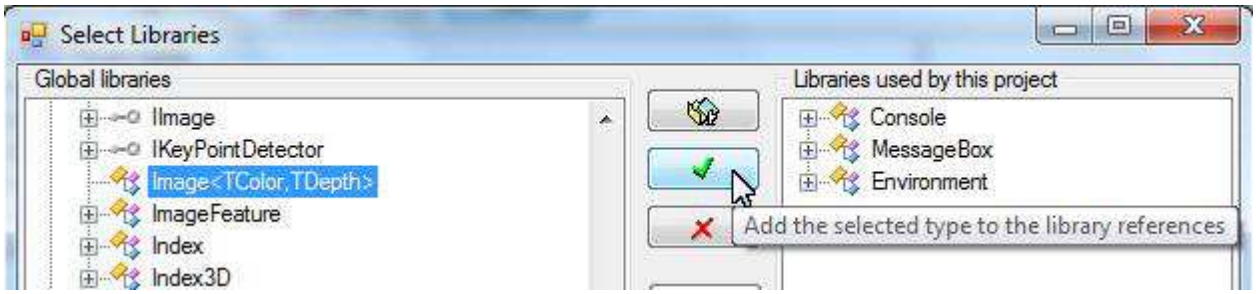


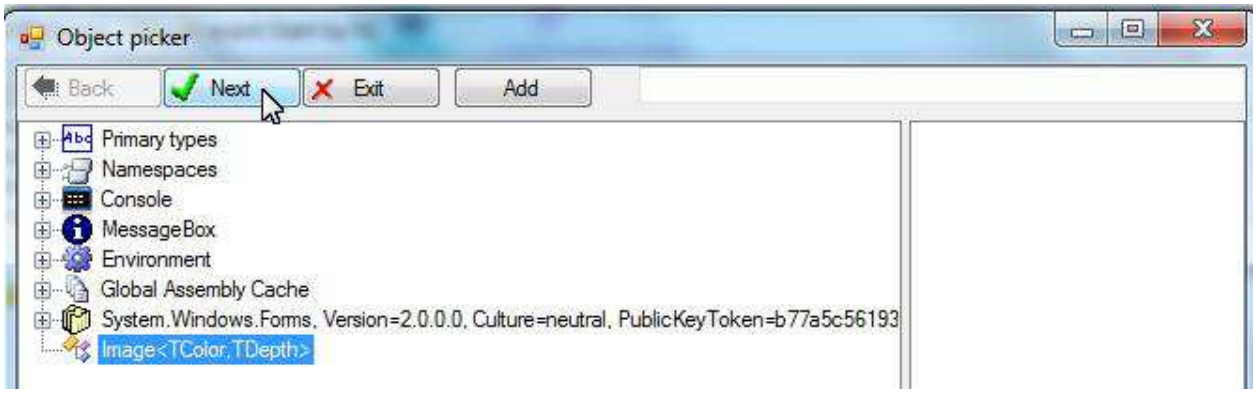
Select Emgu.CV.dll from the bin folder under the Emgu installation folder:



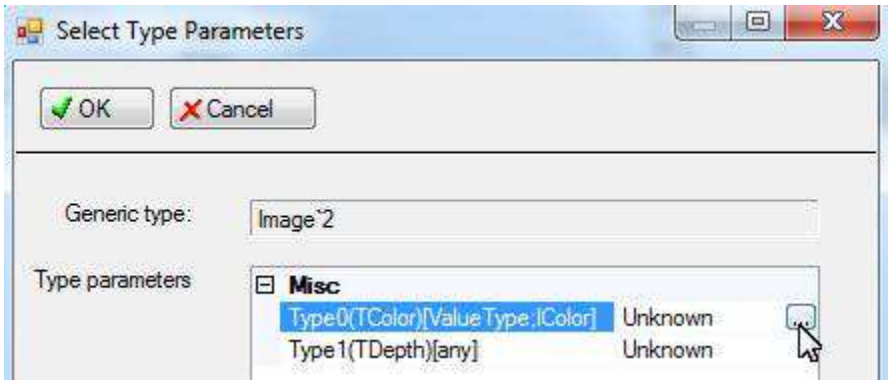
Select the Image class from the Emgu library:

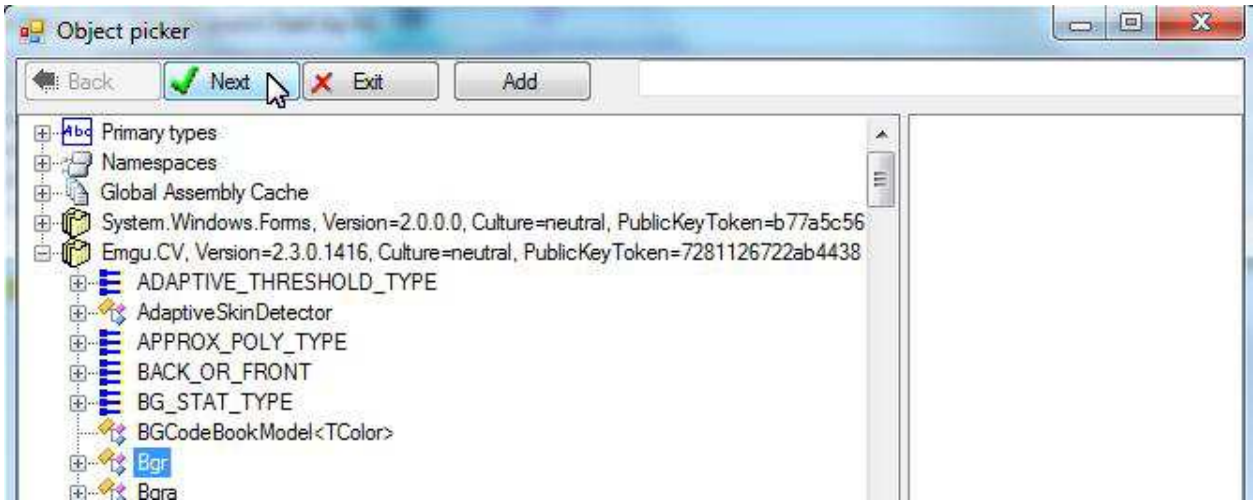


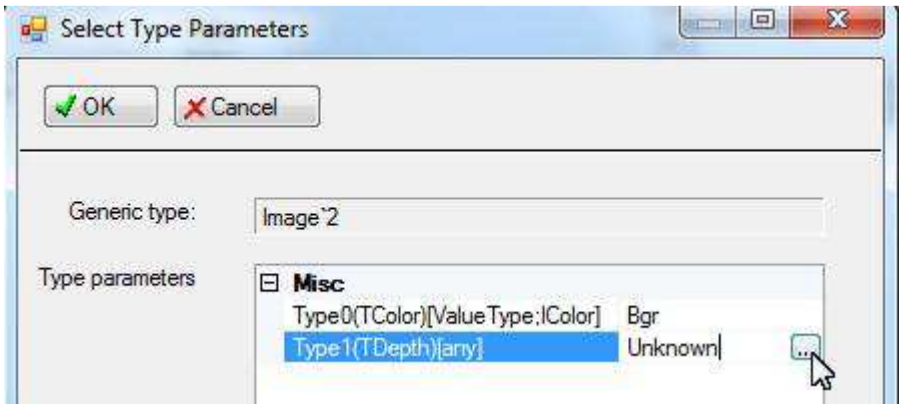


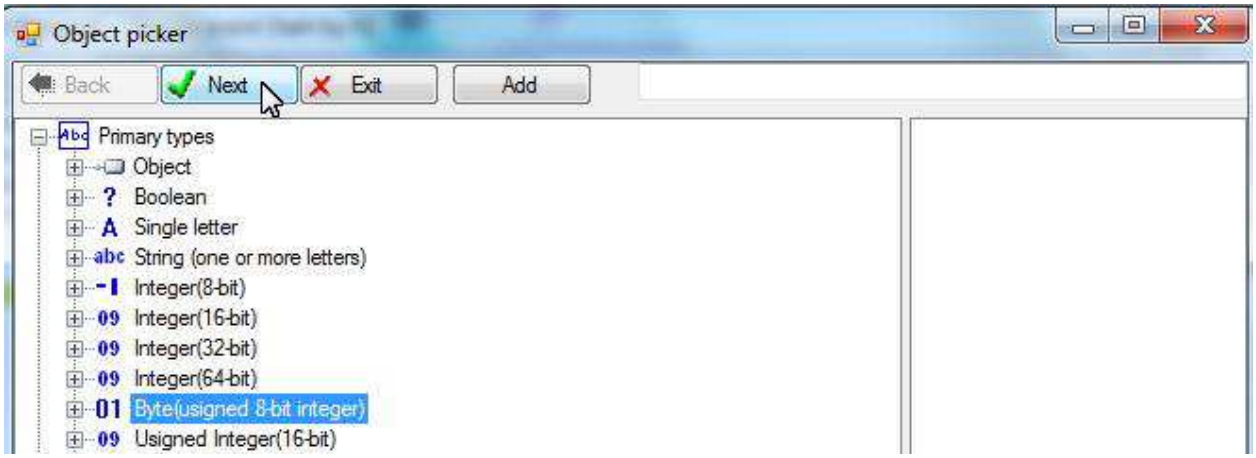


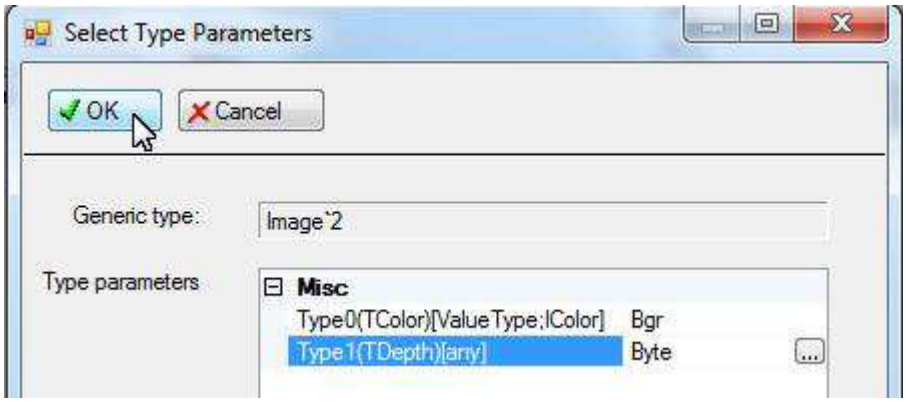
The Image class contains two generic types, TColor and TDepth. According to the original Emgu C# sample, TColor should be class Bgr from the Emgu library, and TDepth should be byte.


Select Type Parameters
OK
Cancel
Generic type:
Image`2
Type parameters
Misc
Type0(TColor)[ValueType;IColor]
Unknown
Type1(TDepth)[any]
Unknown

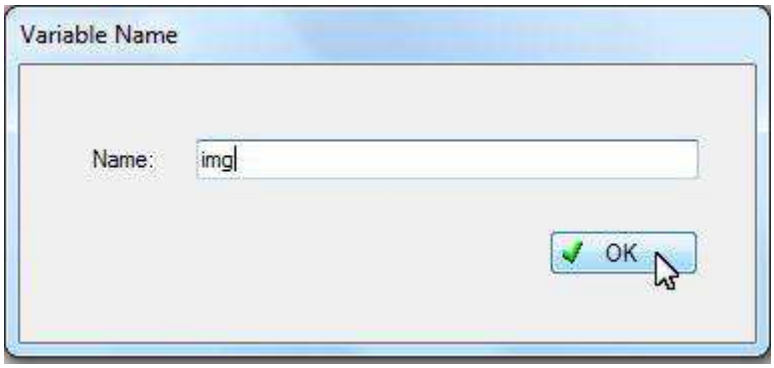

Object picker
Back
Next
Exit
Add
Primary types
Namespaces
Global Assembly Cache
System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c56
Emgu.CV, Version=2.3.0.1416, Culture=neutral, PublicKeyToken=7281126722ab4438
ADAPTIVE_THRESHOLD_TYPE
AdaptiveSkinDetector
APPROX_POLY_TYPE
BACK_OR_FRONT
BG_STAT_TYPE
BGCodeBookModel<TColor>
Bgr
Bgra


Select Type Parameters
OK
Cancel
Generic type:
Image`2
Type parameters
Misc
Type0(TColor)[ValueType;IColor]
Bgr
Type1(TDepth)[any]
Unknown

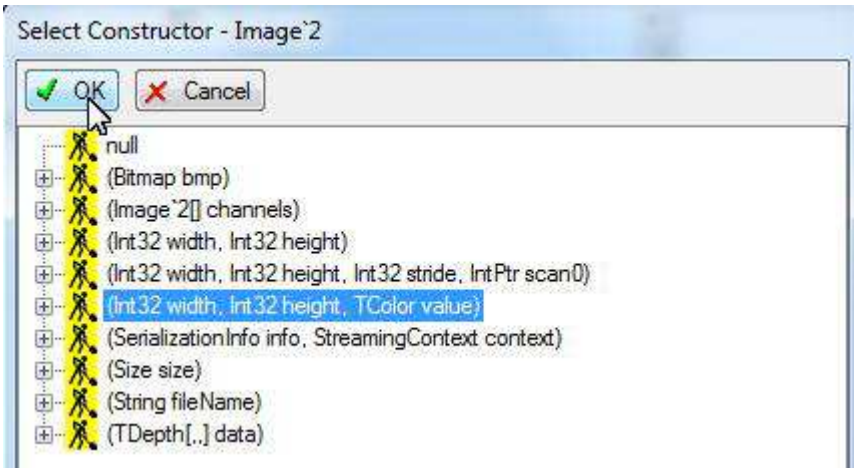




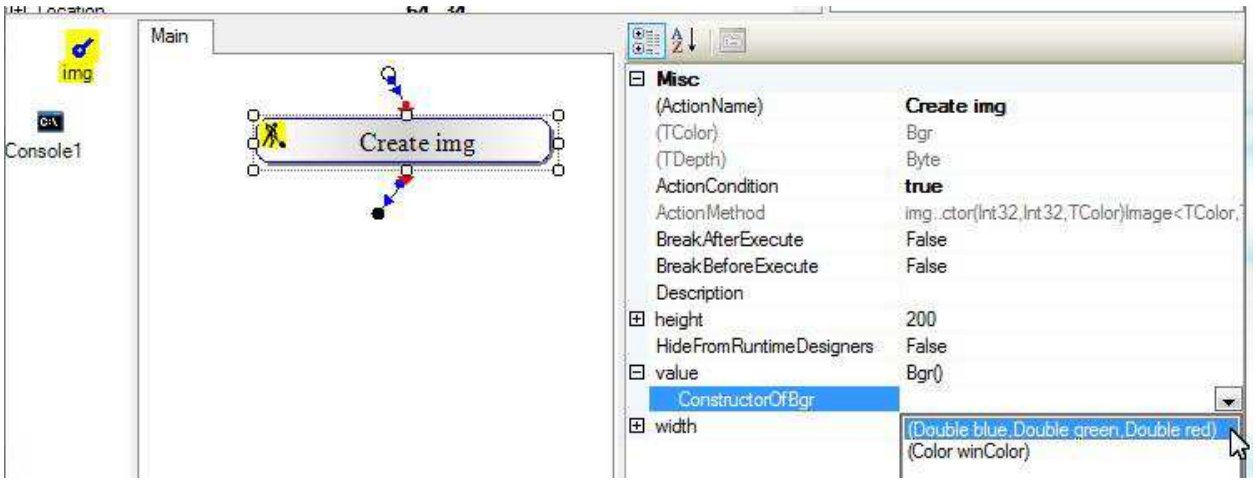
Give a name to the image variable:

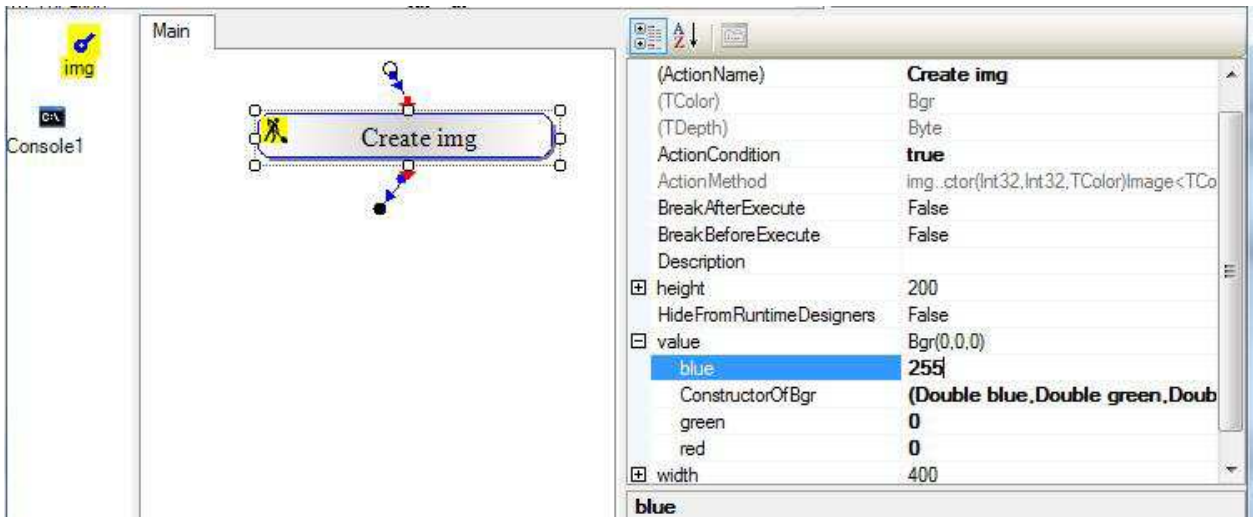


Select a constructor for the variable. Choose the constructor using size and color:

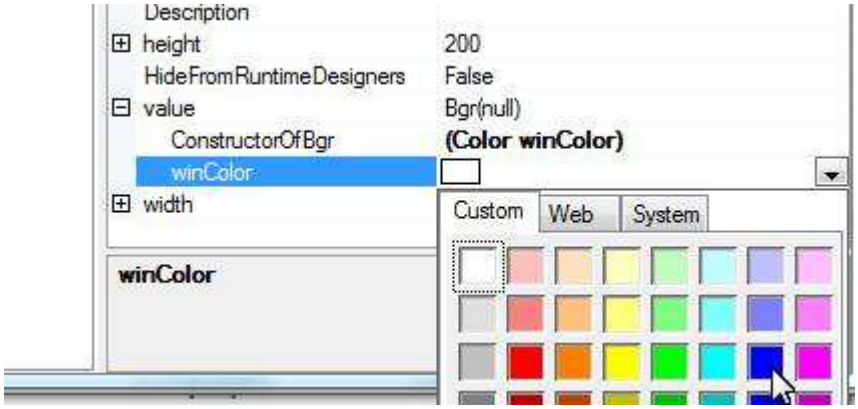


The image variable appears in the Variable Pane. An action appears in the Action Pane. The action constructs the variable. We need to specify the size to be (400,200) and color to blue for the action:



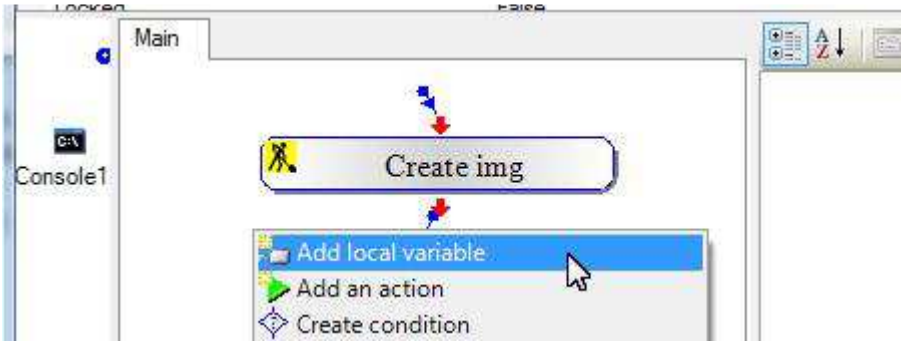


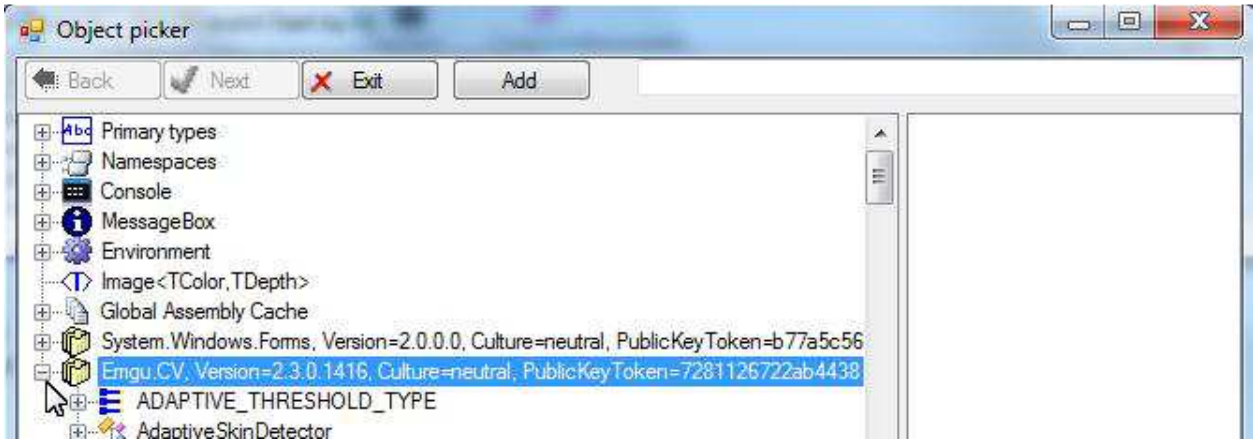
You may also specify the color visually:

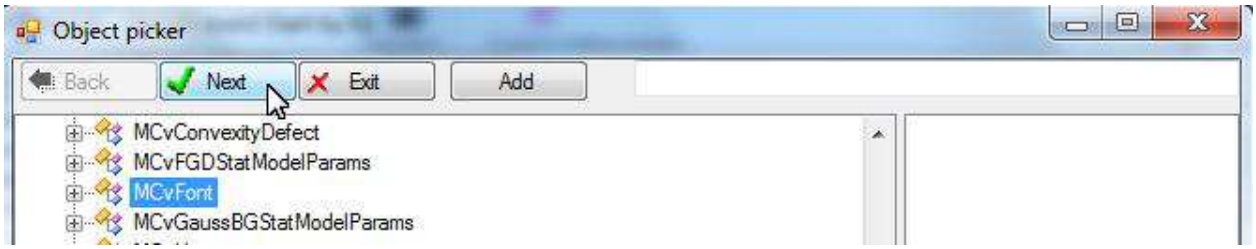


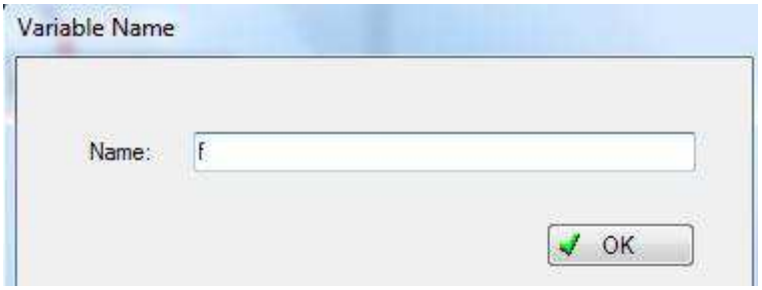
Create font

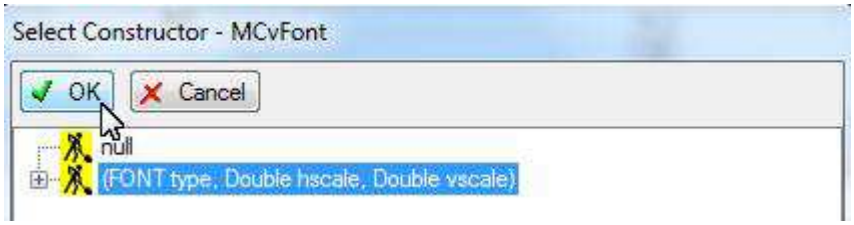
To draw text, we need to create a font.



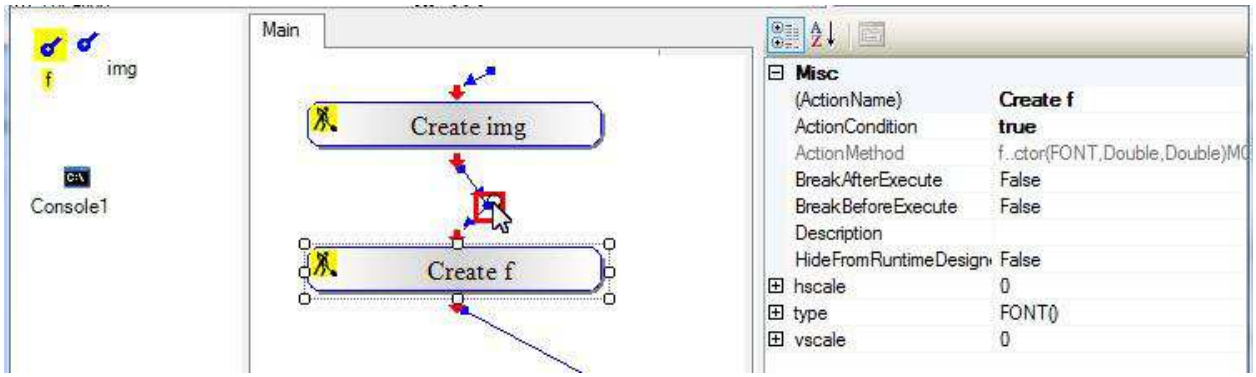




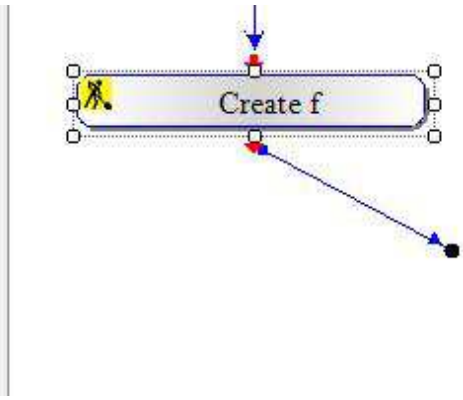


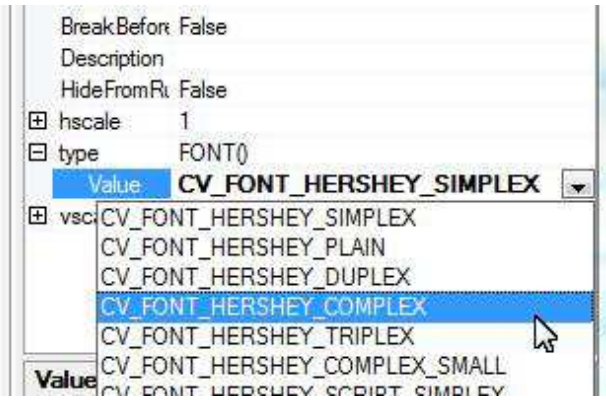


A font variable appears. A new action appears for creating the font. Link the action with the existing one:



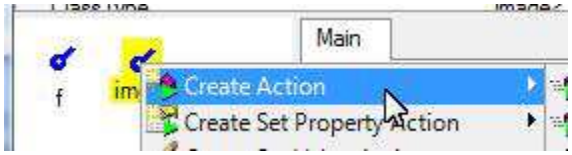
Specify the parameter values for the action:

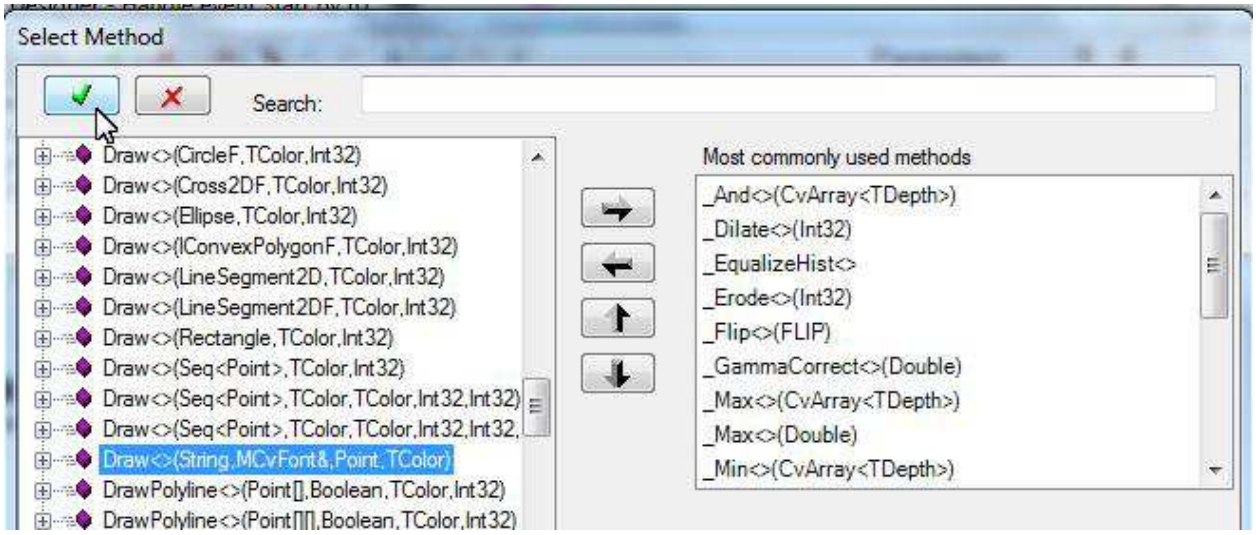




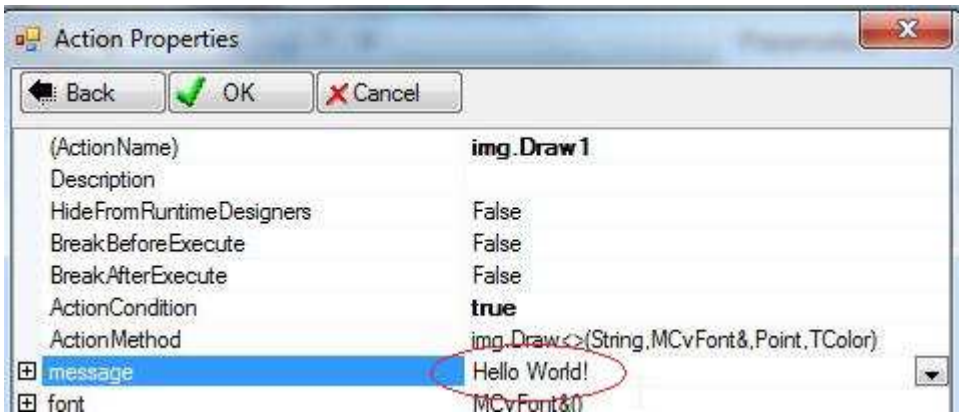
Draw text

The image object has many Draw methods. We may use one of it to create an action to draw “Hello World!” on the image.

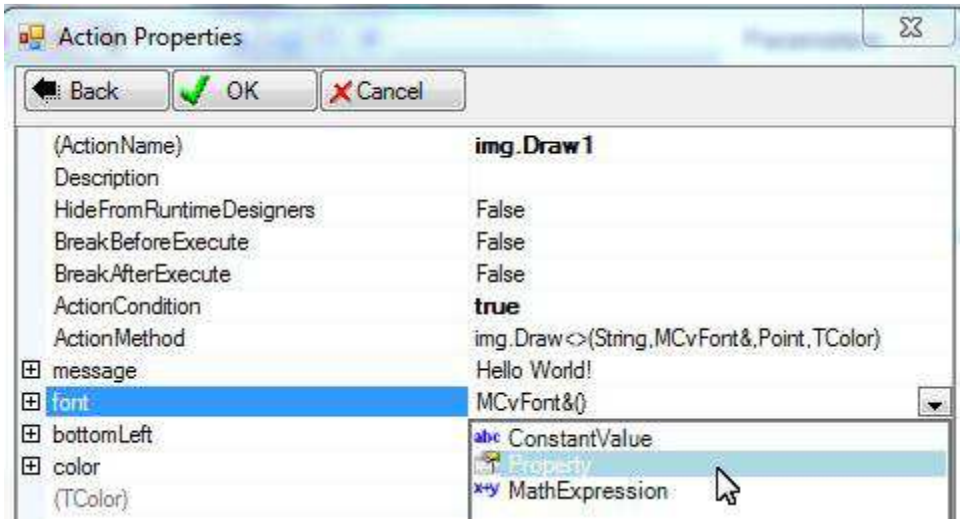


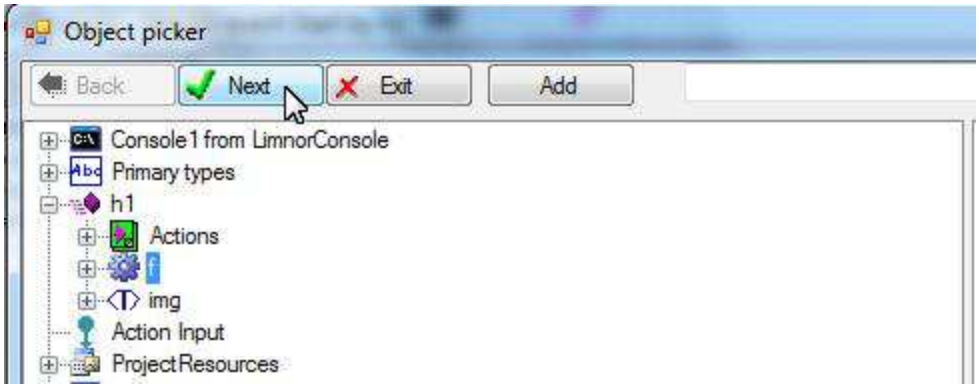


Specify the text to draw:

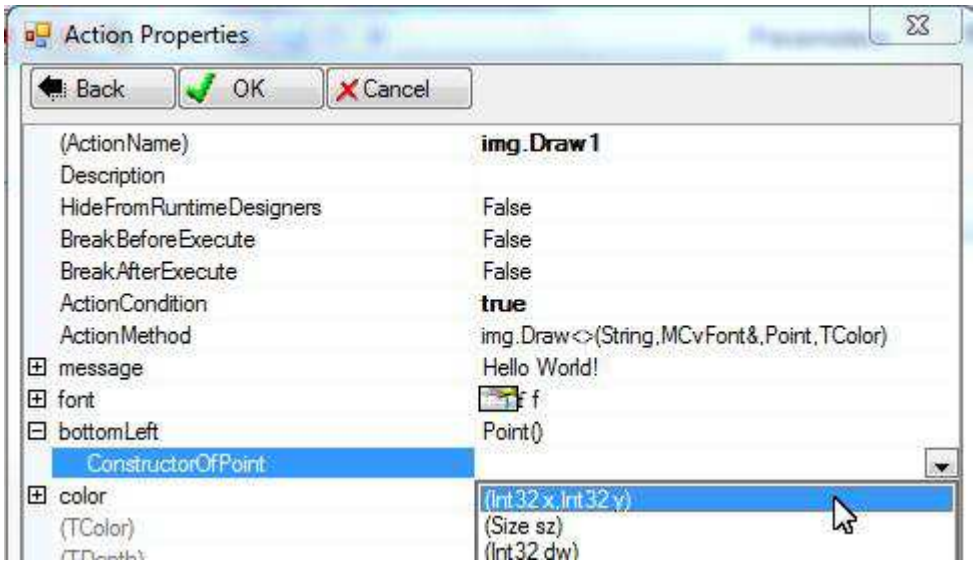


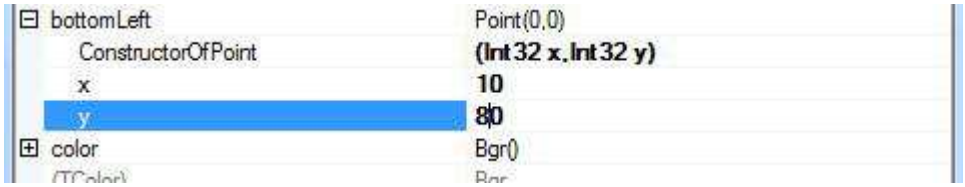
Use the font we have created:



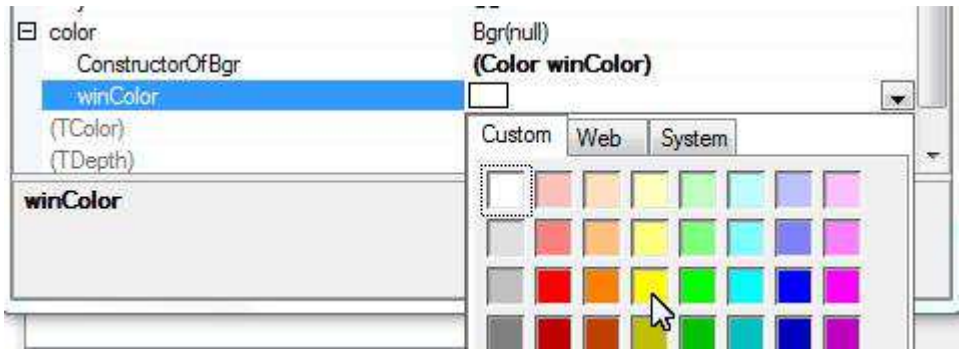


Specify the drawing location:

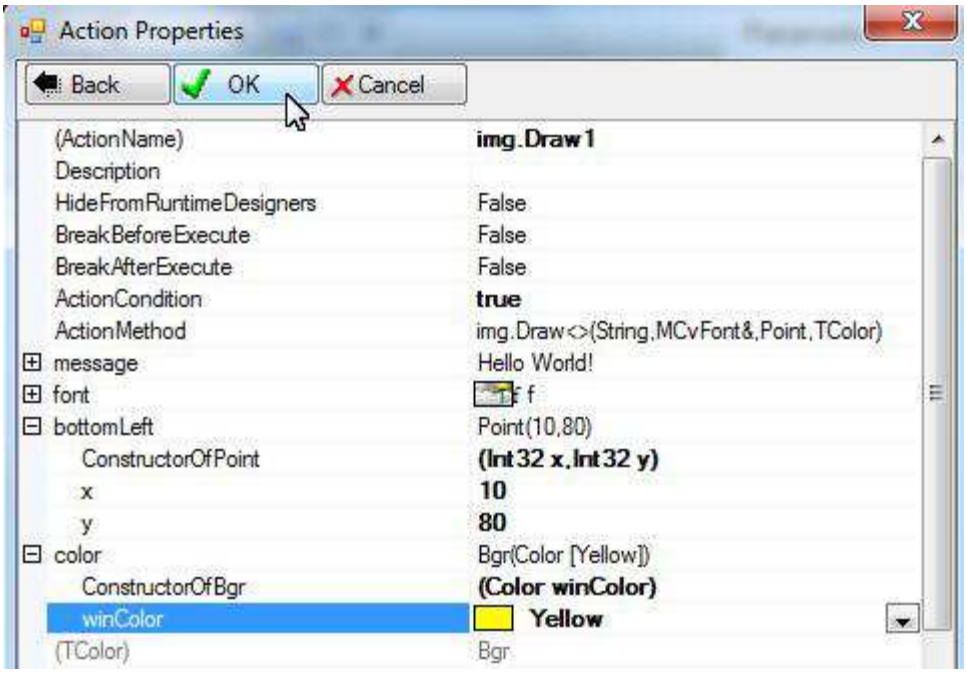


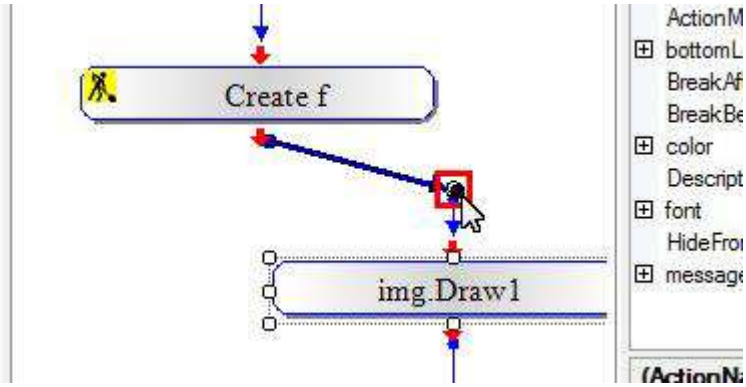


Specify the drawing color:



Click OK to finish creating this action:



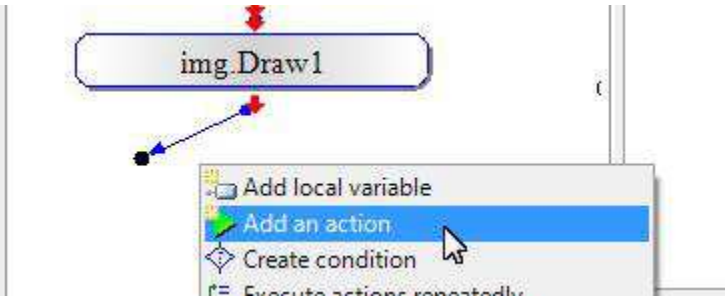


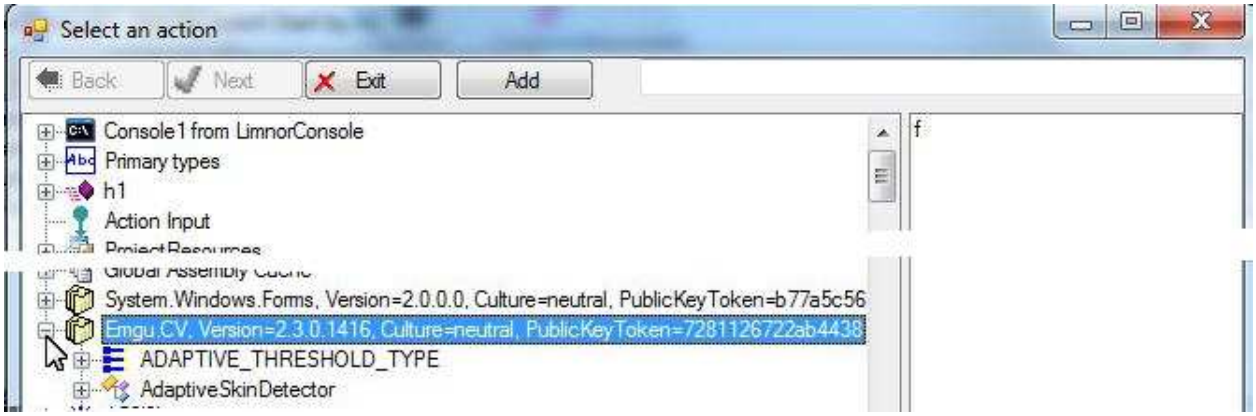
Create a window to display image

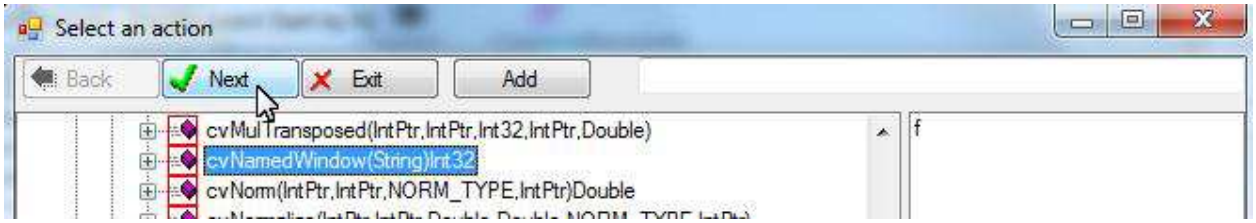
Emgu uses a class, CvInvoke, to provide many static methods. This sample uses following methods:

cvNamedWindow -- create a window
cvShowImage -- show an image on a window
cvWaitKey -- Wait for the key pressing event
cvDestroyWindow -- Destory a window

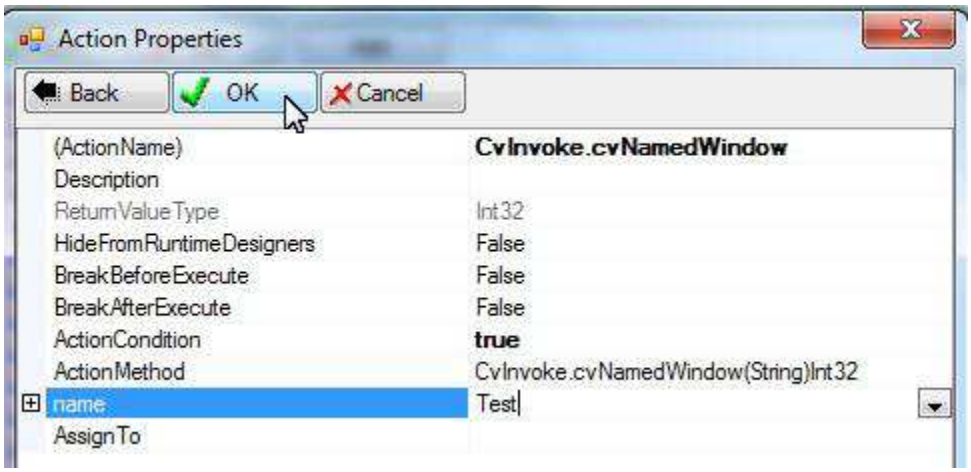
Let's create 4 actions using the above methods.



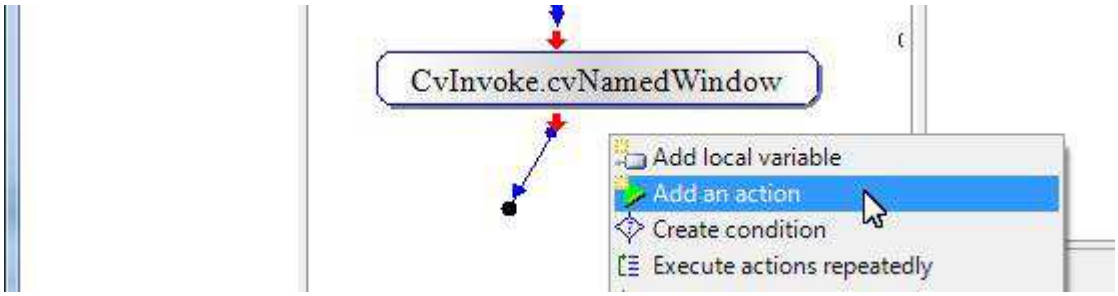


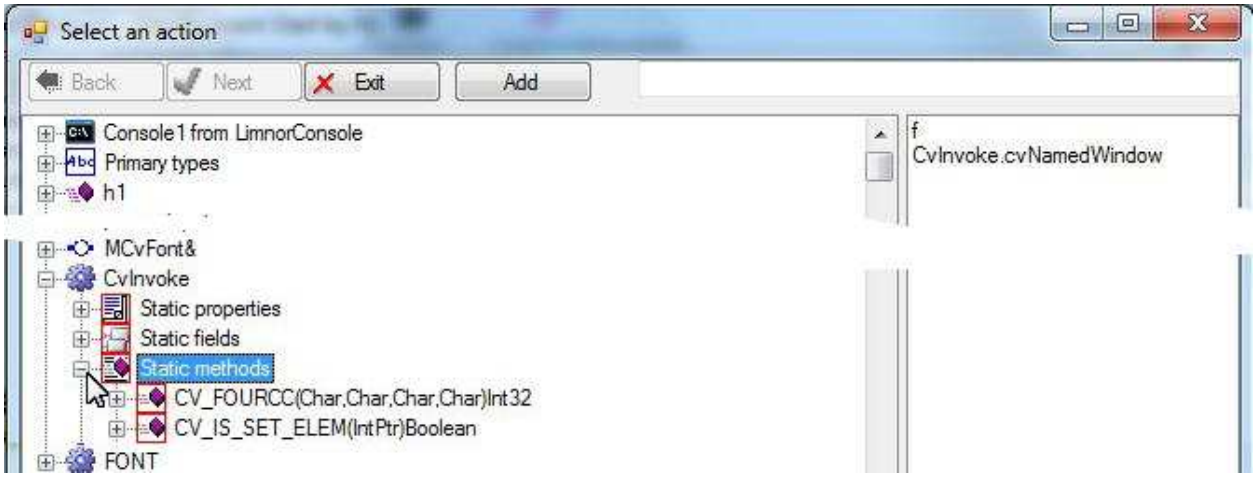


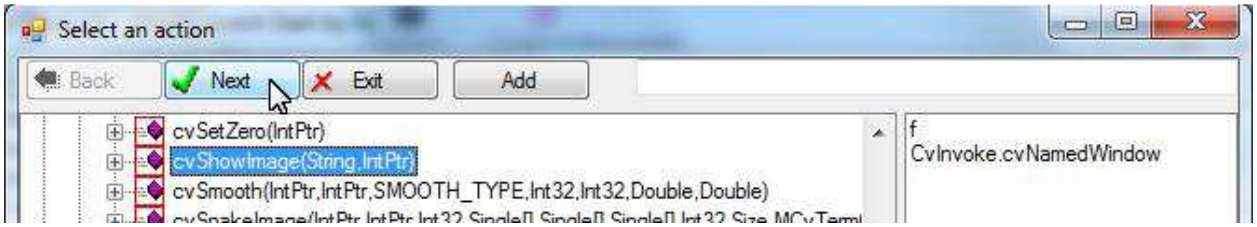
Give a name to the new window. Click OK to finish creating the action.



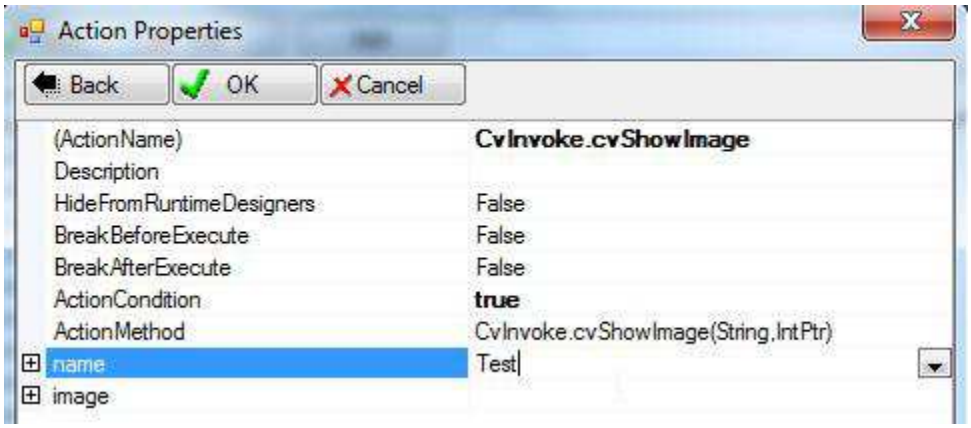
Show image



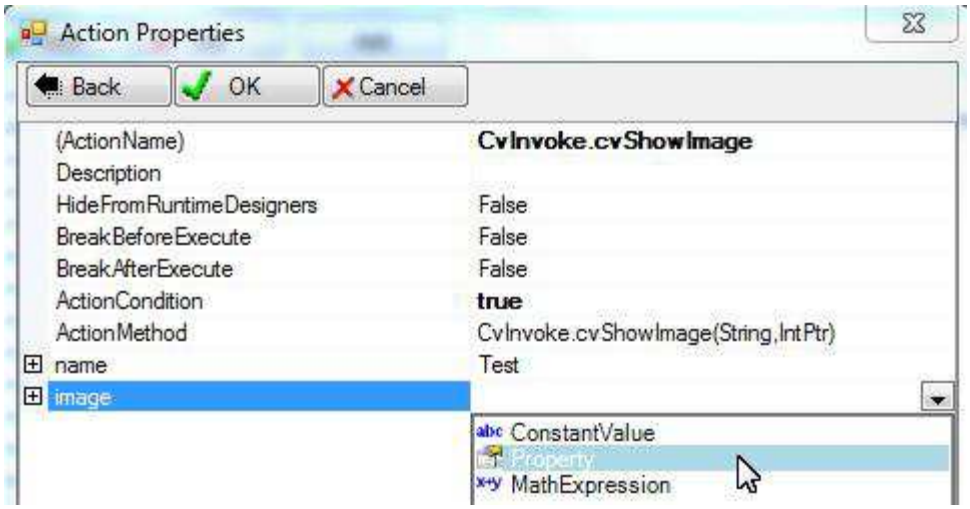


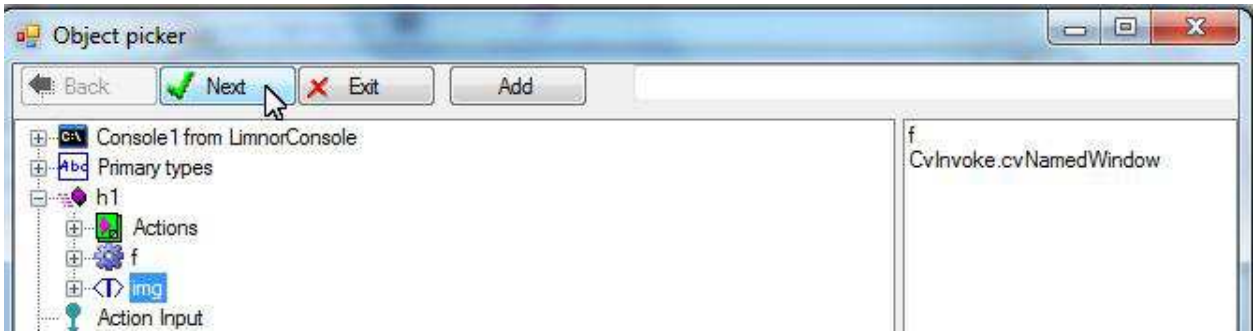


Specify the window name:

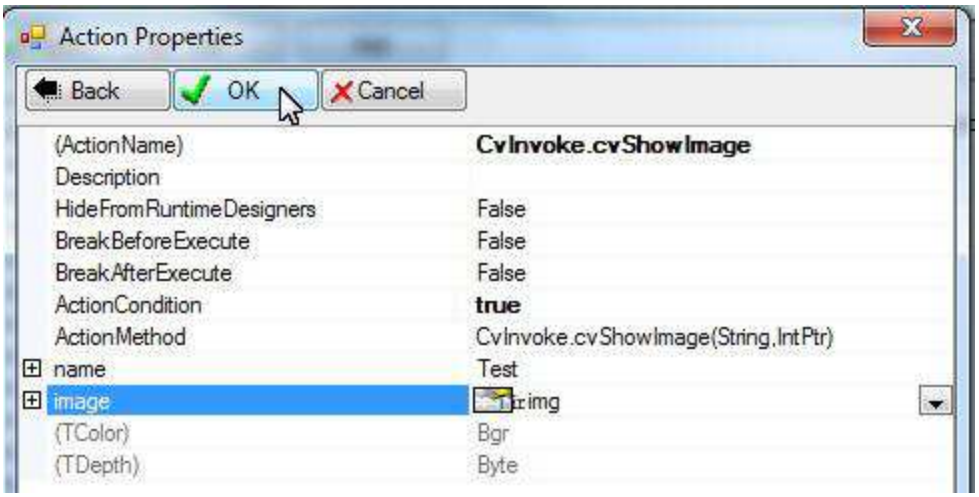


Select the image variable we have created:

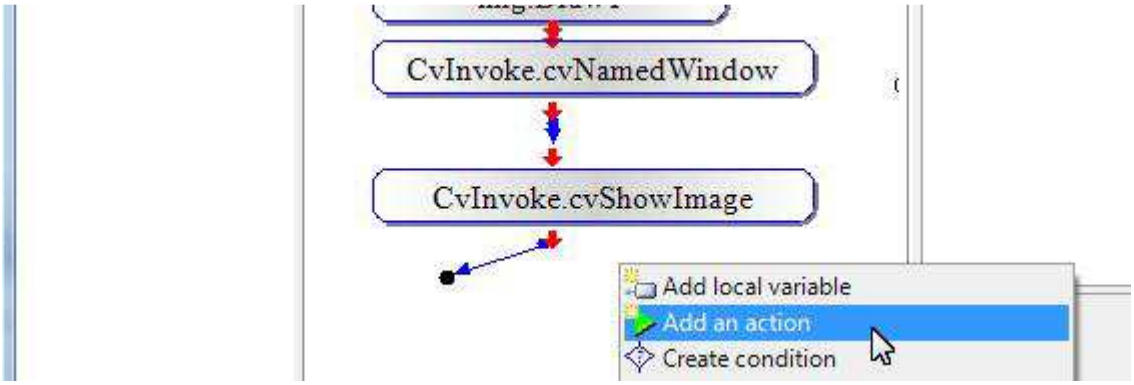


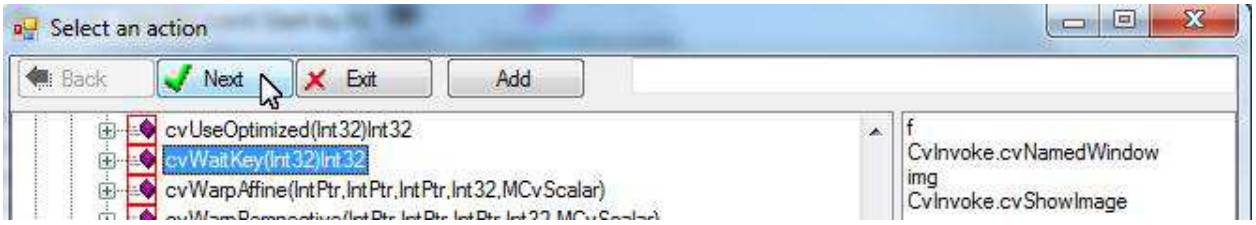


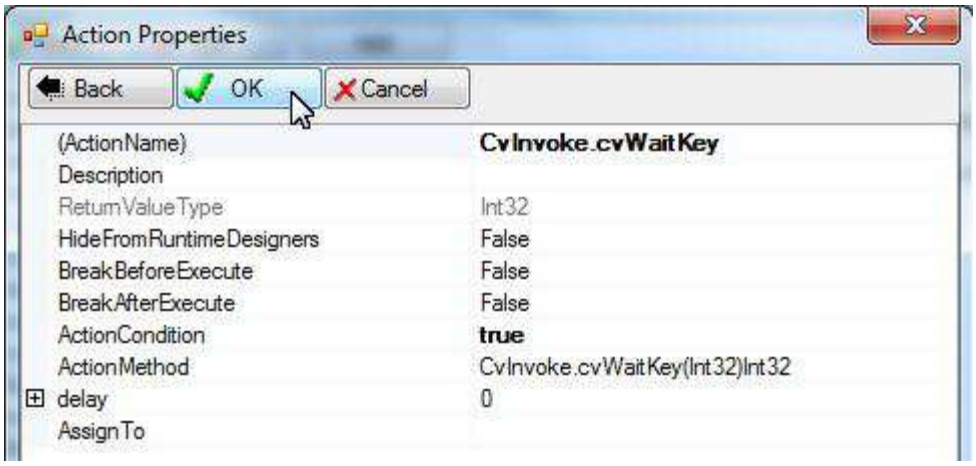
Click OK to finish creating the action:



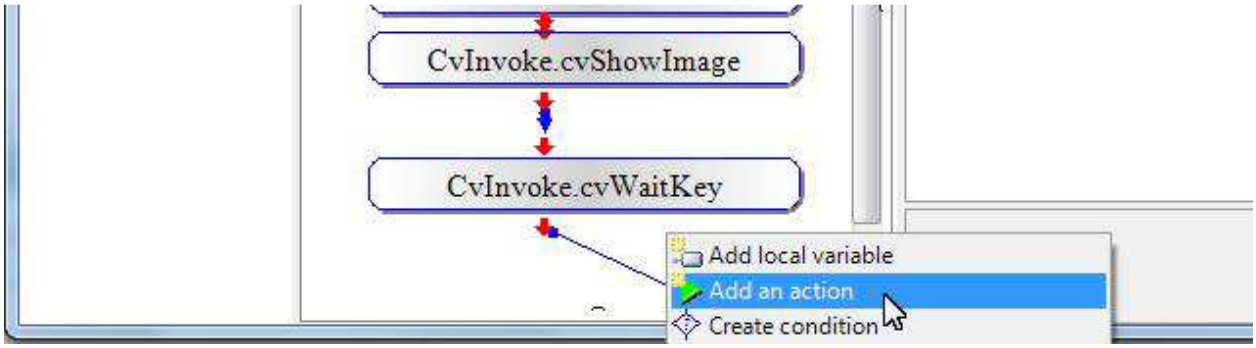
Wait for key press

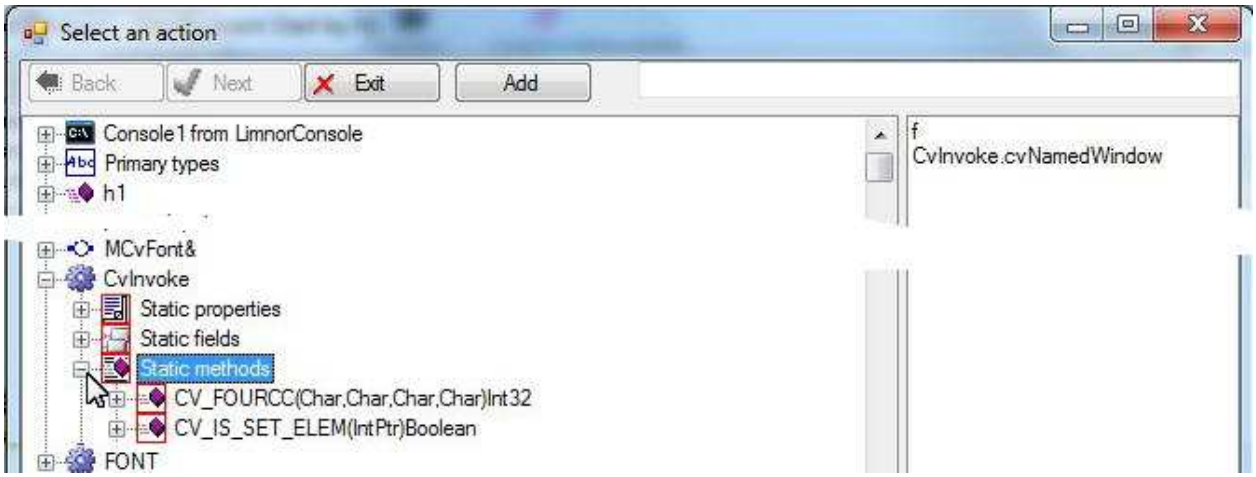


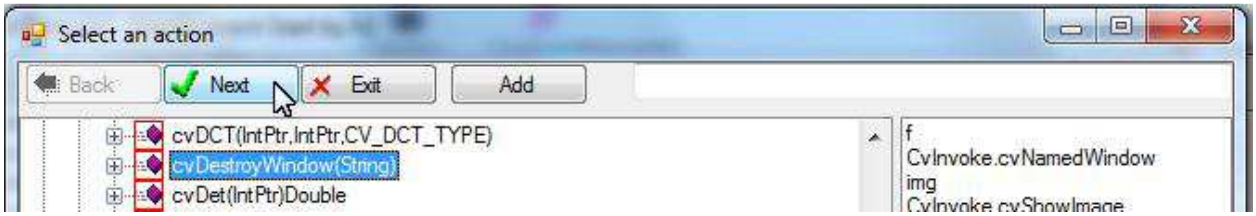




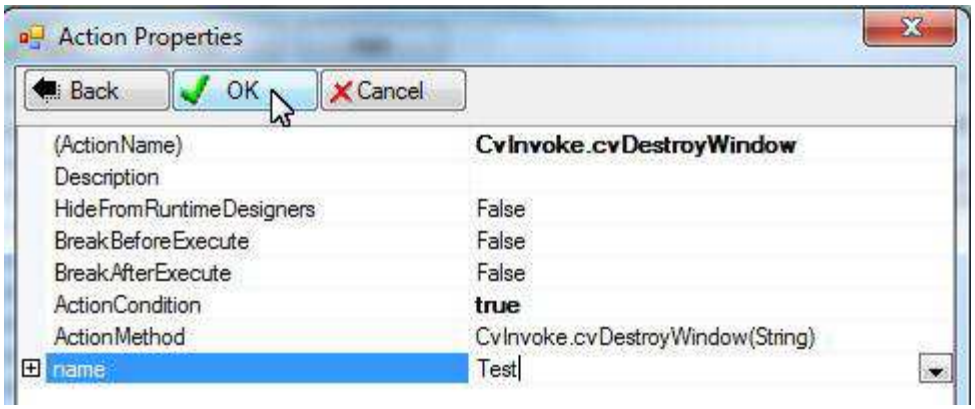
Destroy window





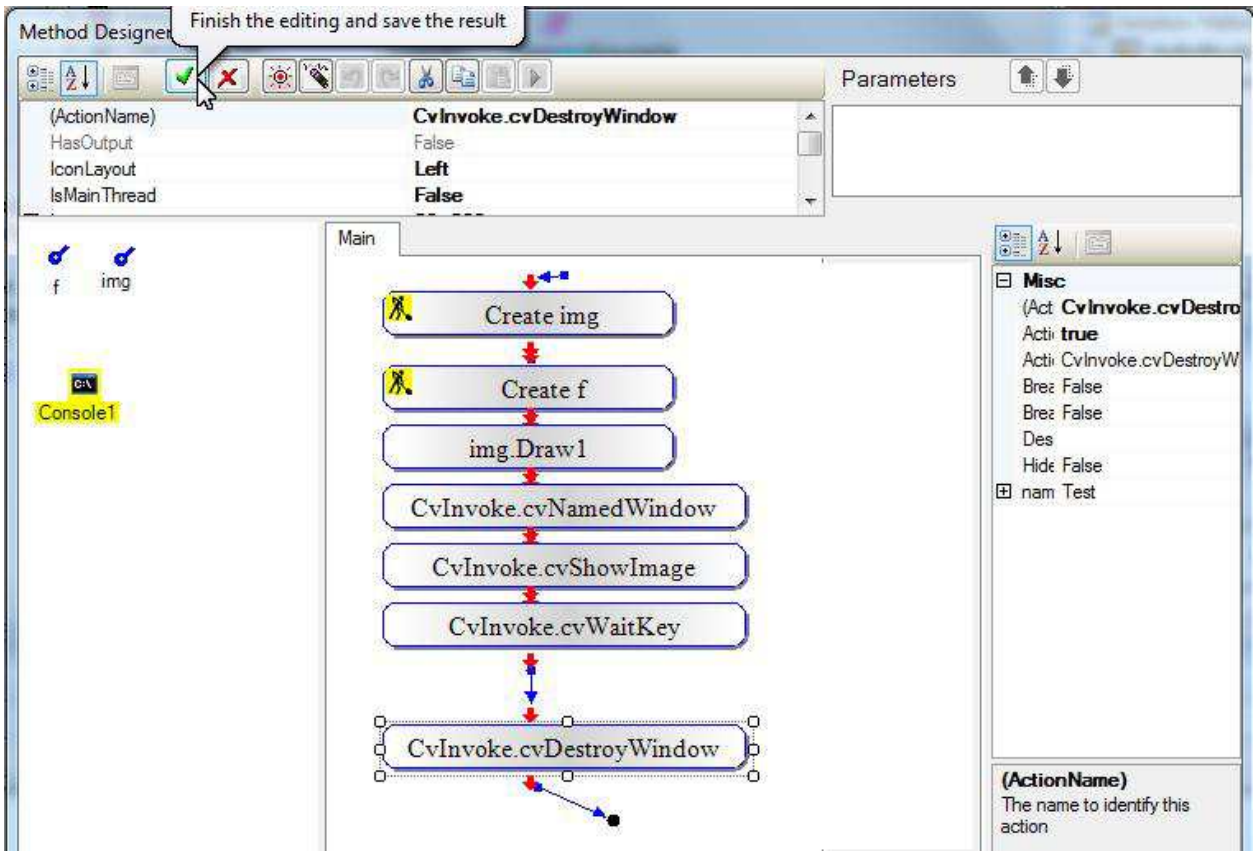


Specify the name for the window to be destroyed. Click OK.

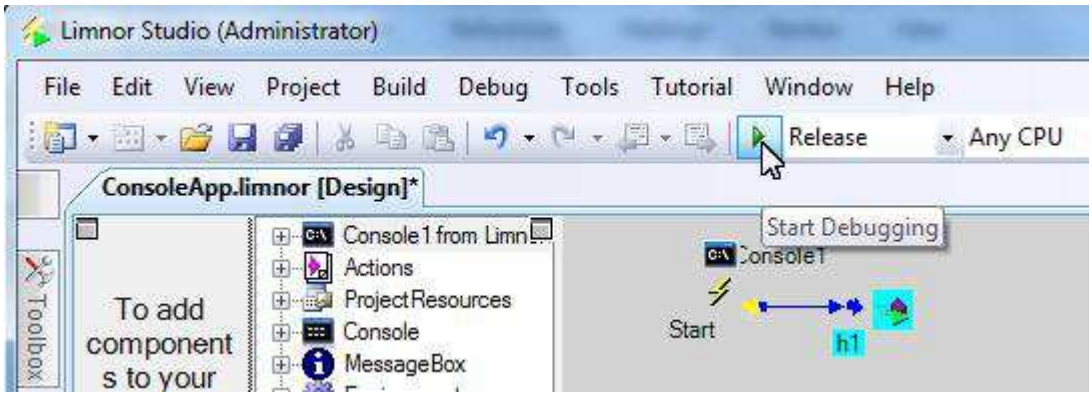


Test

We are done creating this event handler method.



We may test the project now.

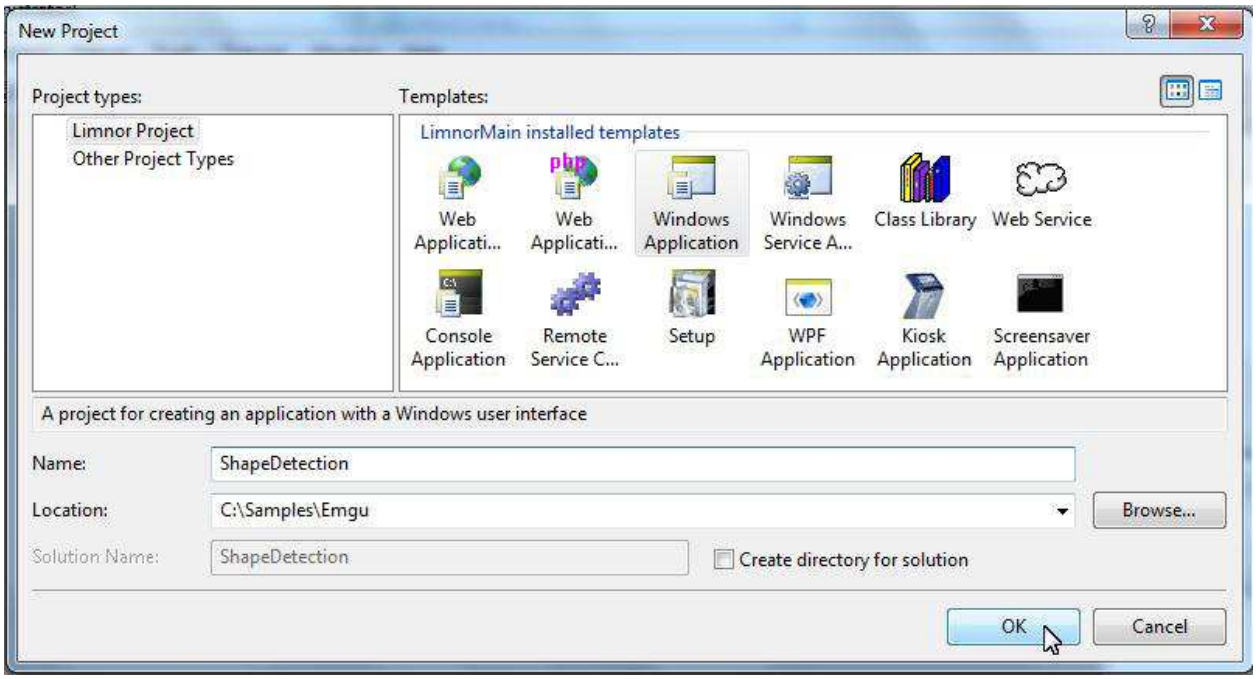


A window appears showing “Hello World!”

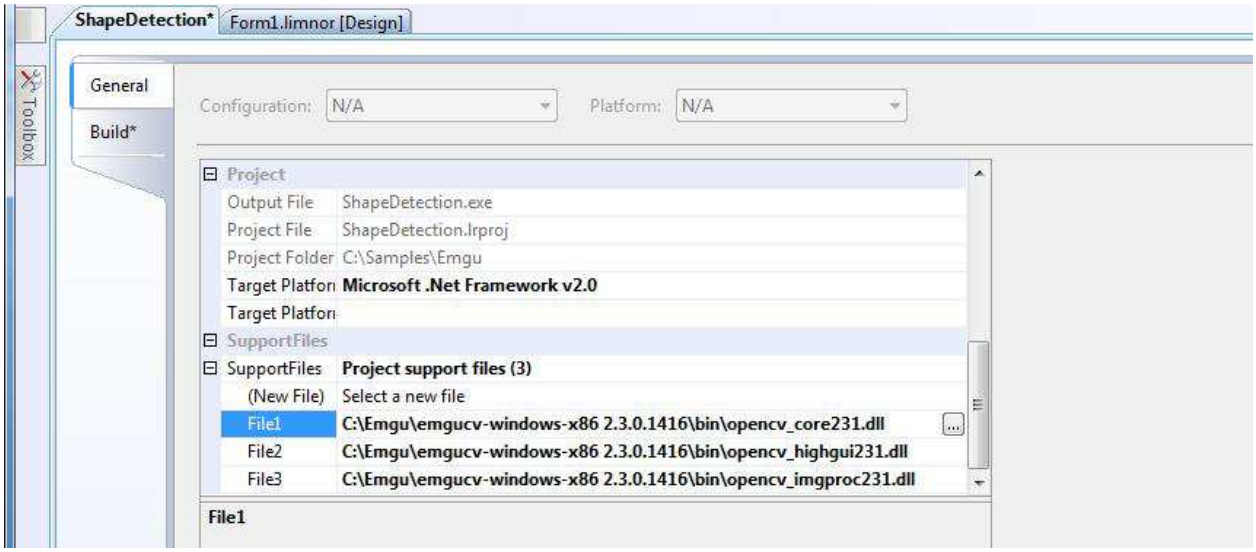
Sample 2 – Shape Detection

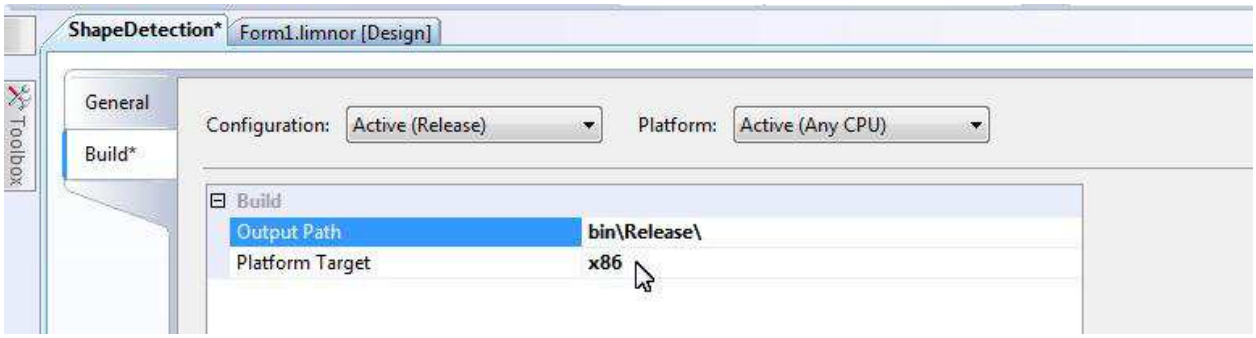
This sample is based on an Emgu sample found in folder “Emgu.CV.Example\ShapeDetection” under the Emgu installation folder.

Create a Windows Form project for this sample:

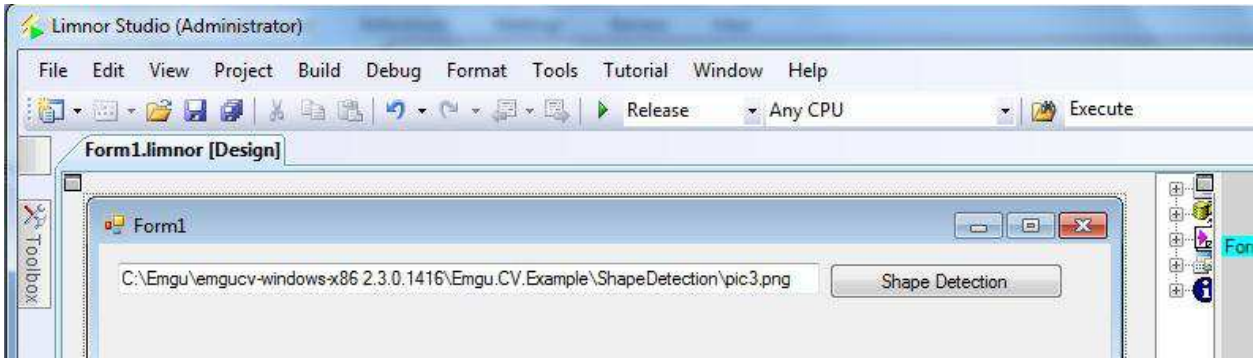


Set the target platform and add the support files to the project, see the last sample for details. Do not forget this step!



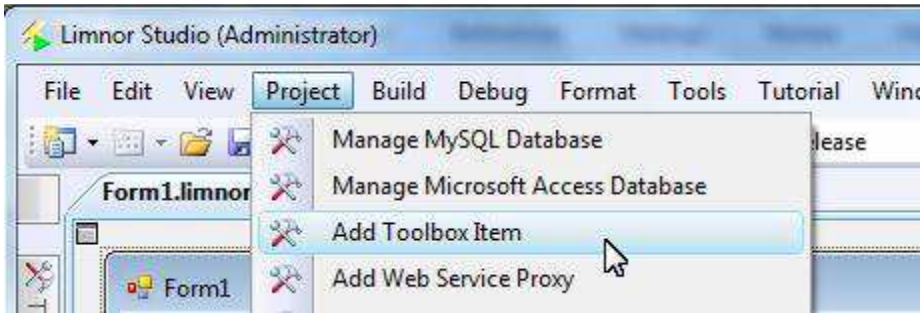


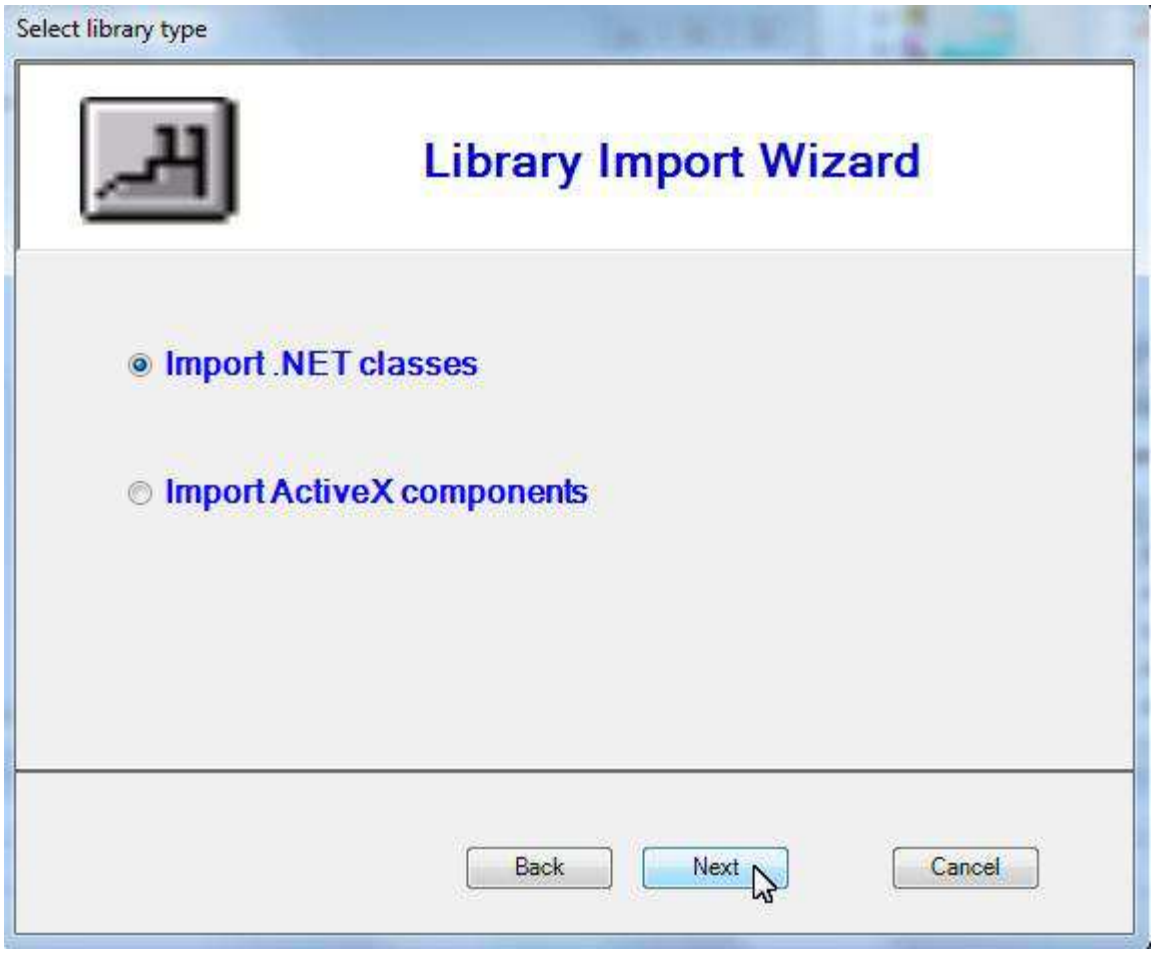
Use a text box for the image file name. Use a button to start shape detection.



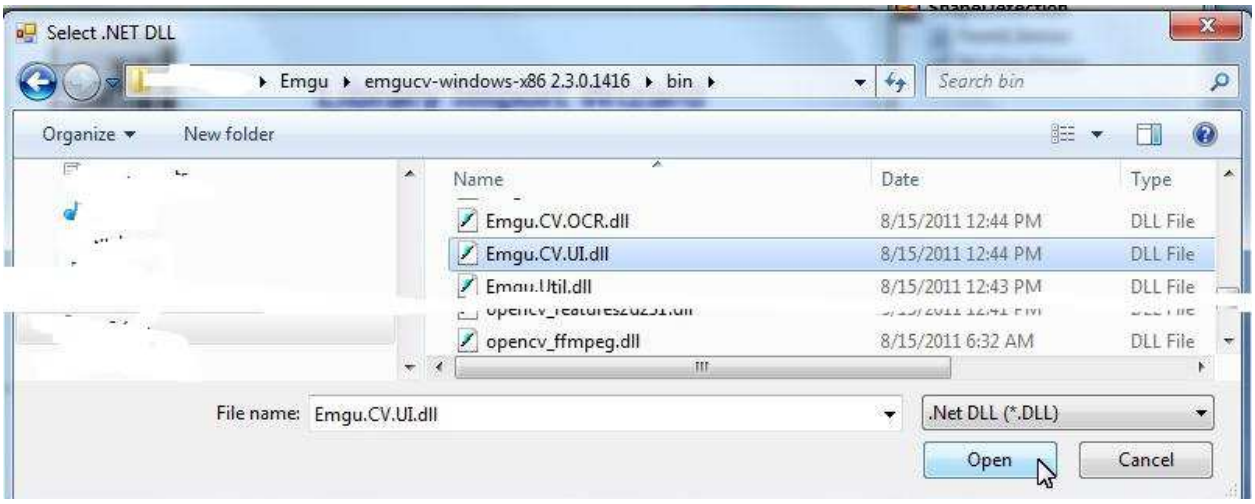
Use Image Box

This sample uses Image Box to display images. Add Image Box to the toolbox so that we may add it to the form.

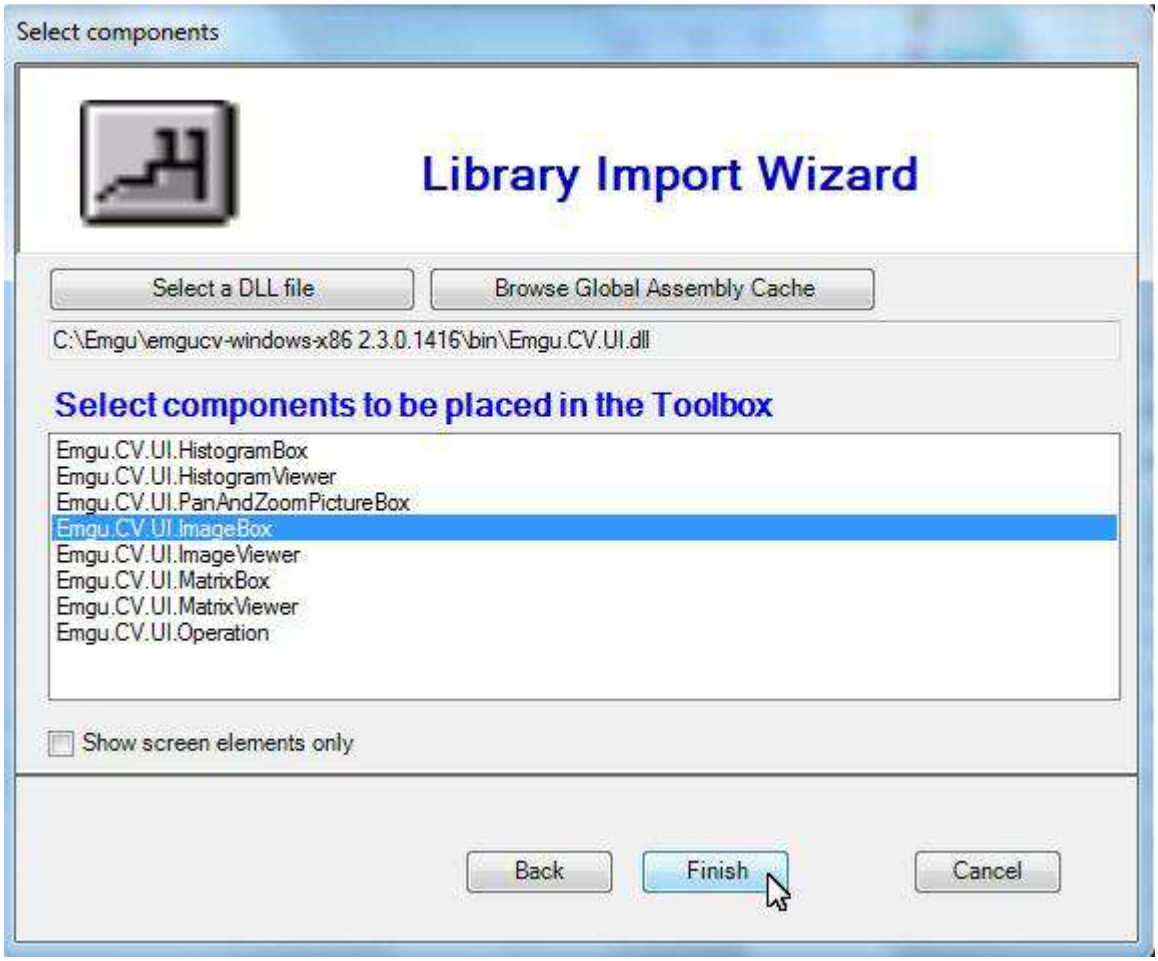




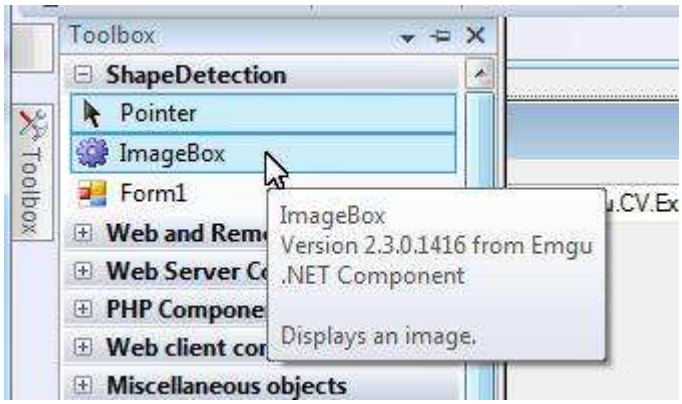
The Image Box is in Emgu.CV.UI.DLL



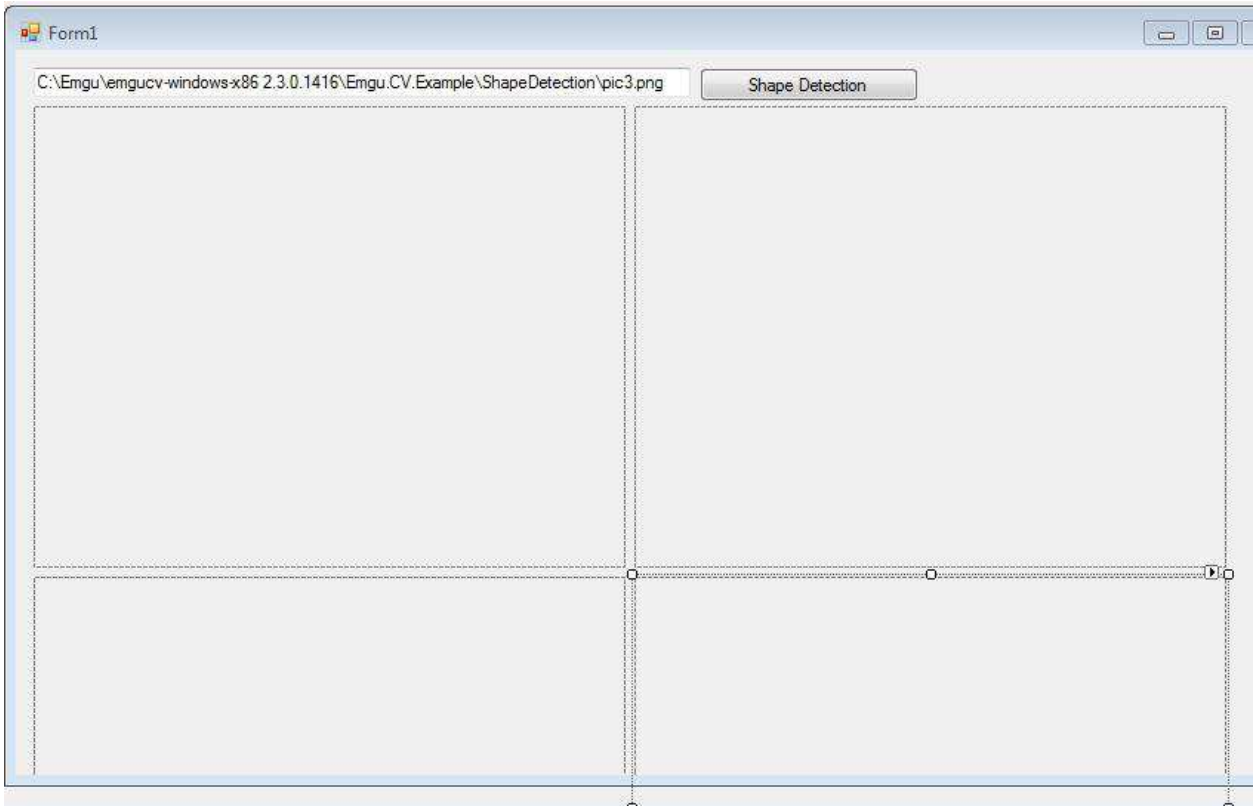
Select the Image Box, click Finish.



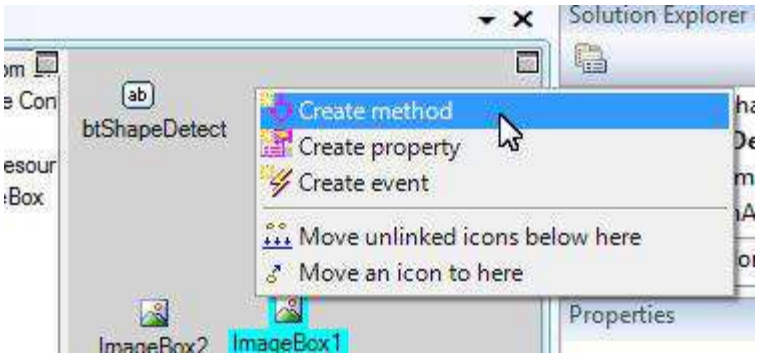
The Image Box appears in the Toolbox:



Add 4 image boxes to the form. One is for displaying the original image. The others are for displaying the detected shapes.



Create Shape Detection Method

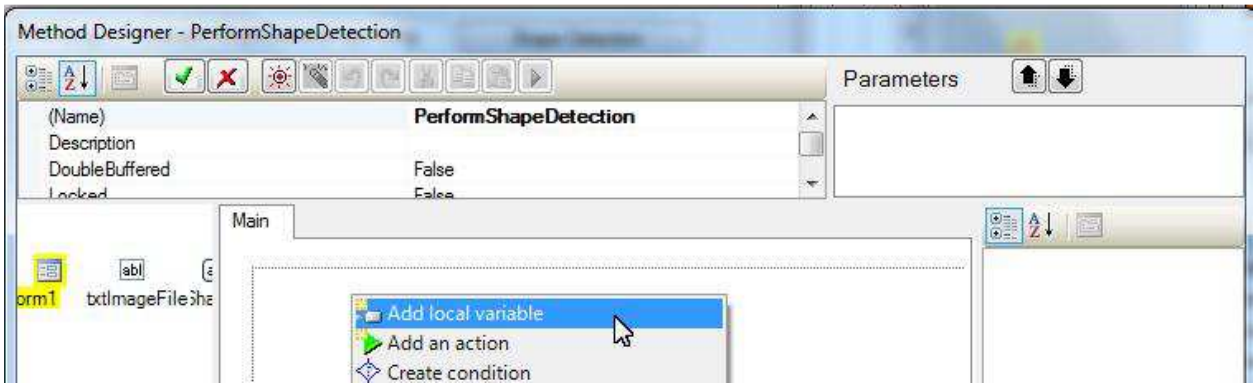


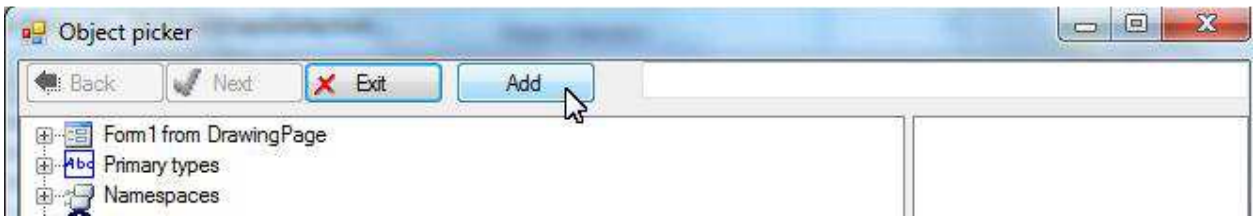
Name the method PerformShapeDetection:

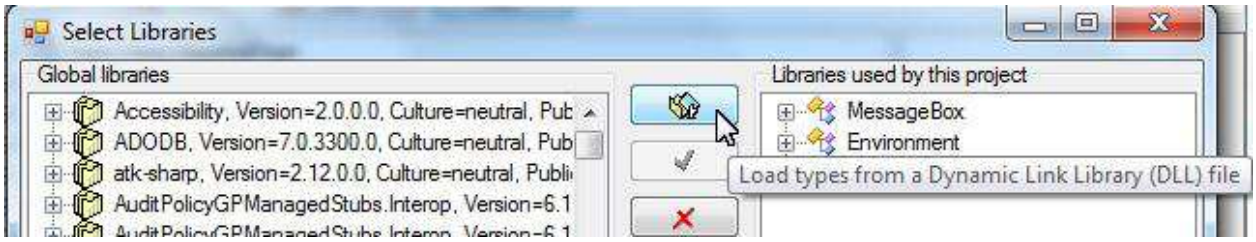
Load image from file

Create image object

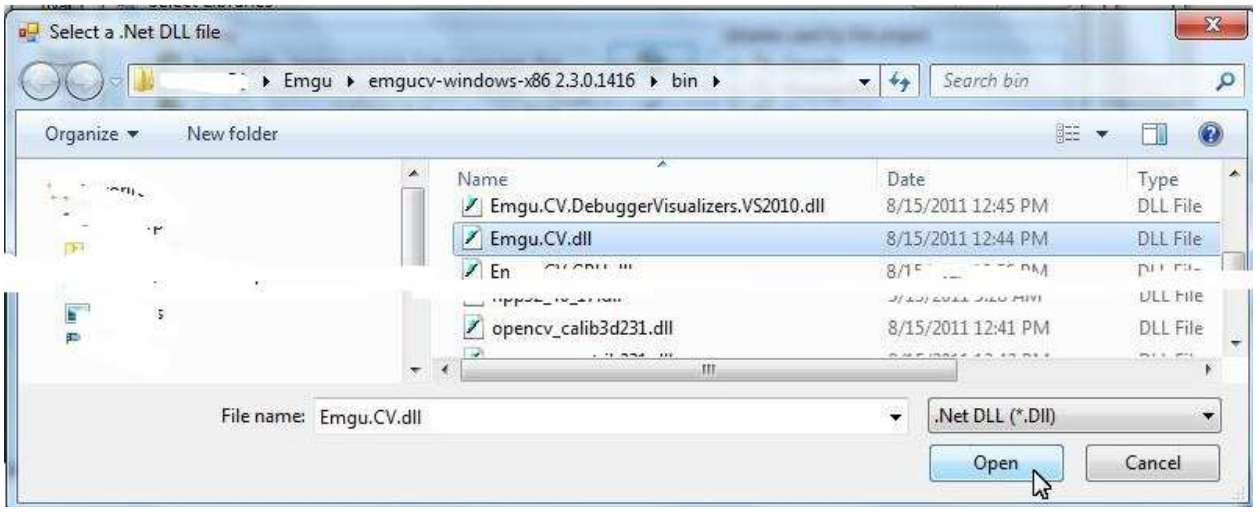
Create an image variable using the image file specified in the text box. It is a similar process as we did in the last sample, but using a different constructor.



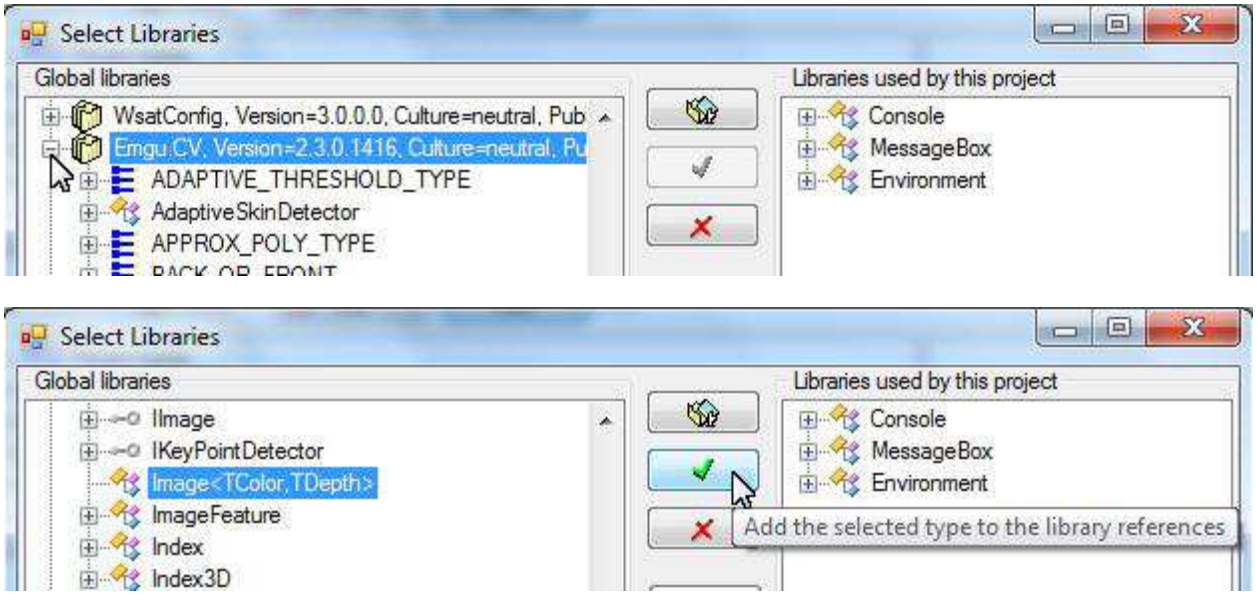


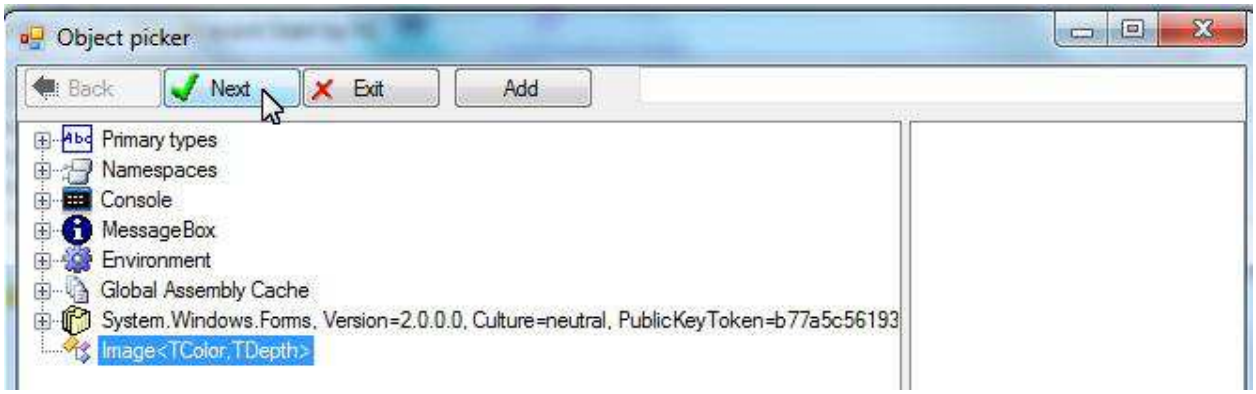


Select Emgu.CV.dll from the Emgu installation folder:

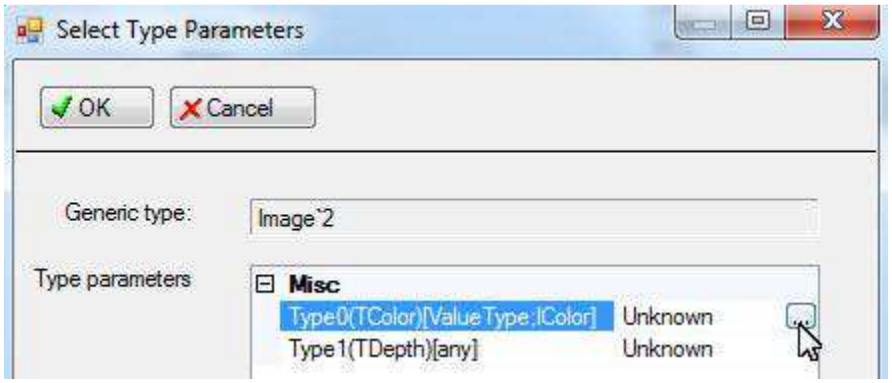


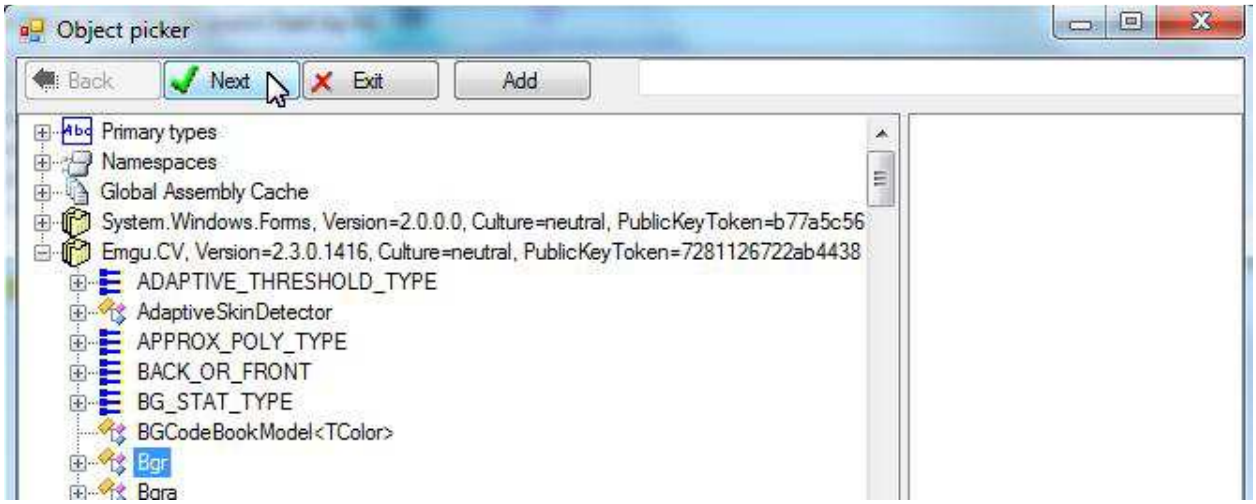
Select the Image class from the Emgu library:

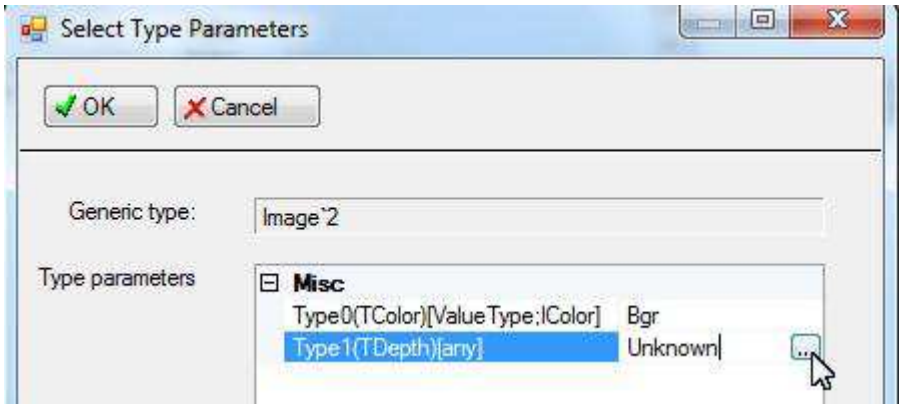


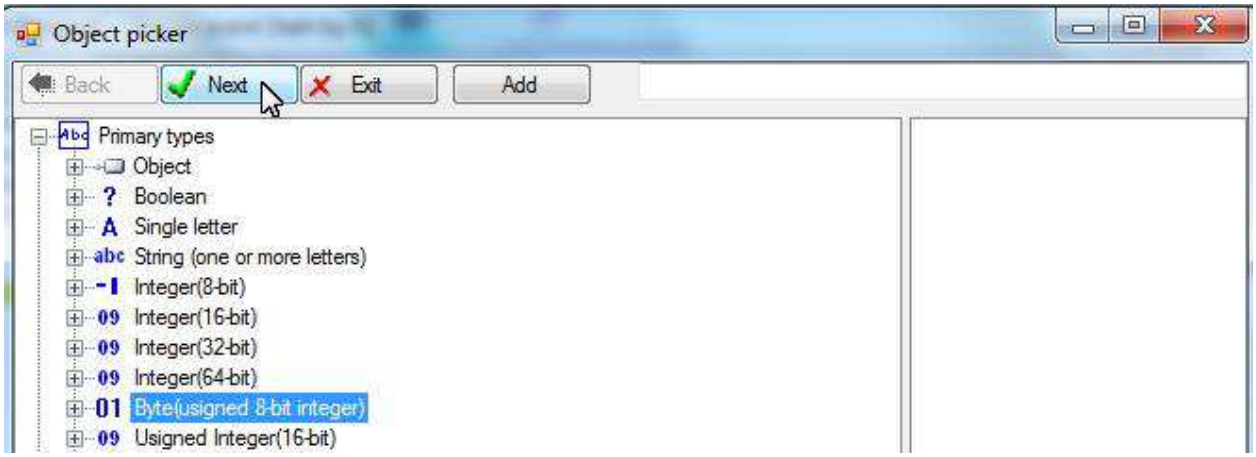


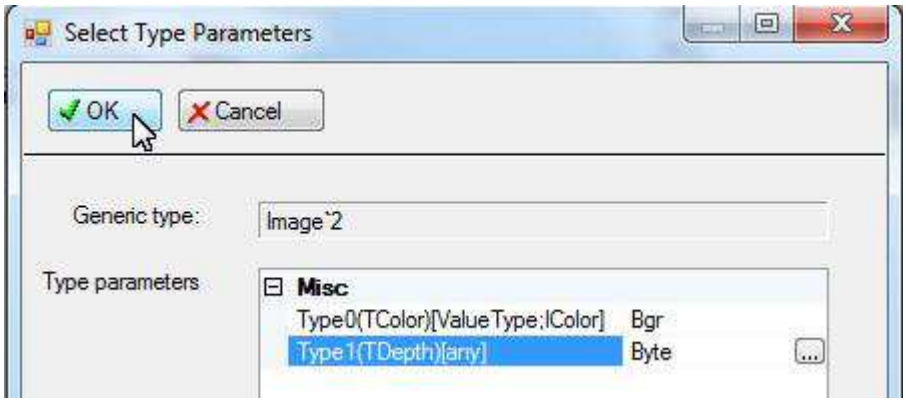
The Image class contains two generic types, TColor and TDepth. According to the original Emgu sample, TColor should be Bgr from the Emgu library, and TDepth should be byte.



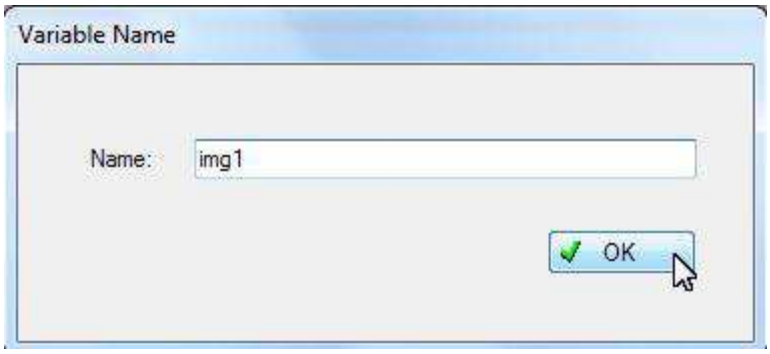




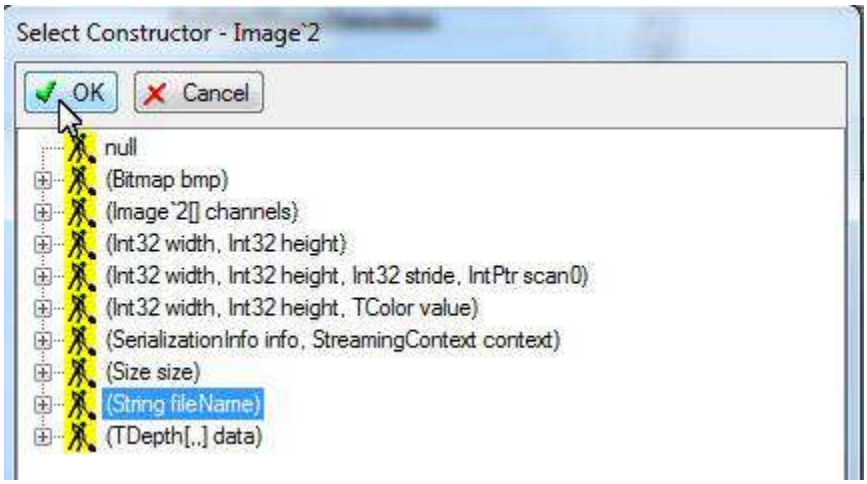




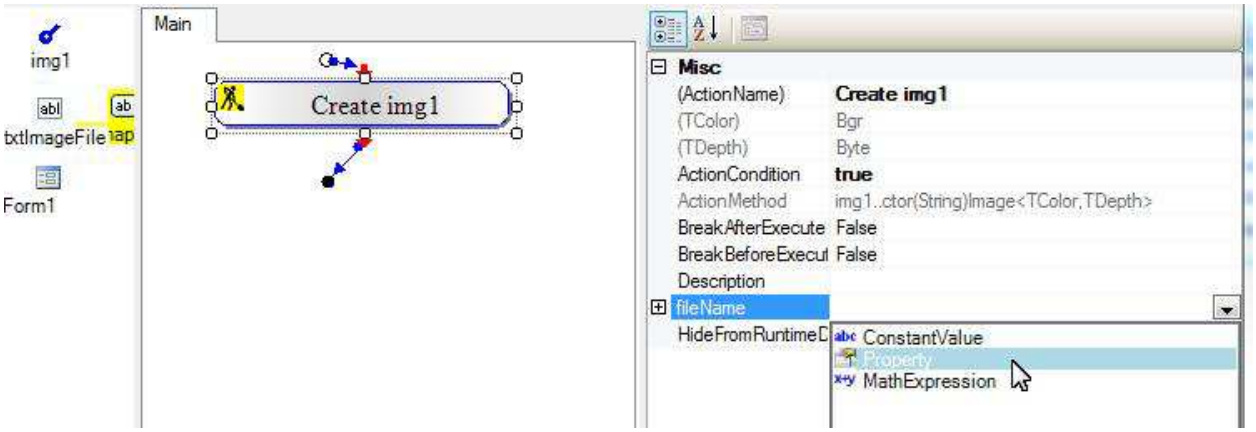
Give a name to the image variable:

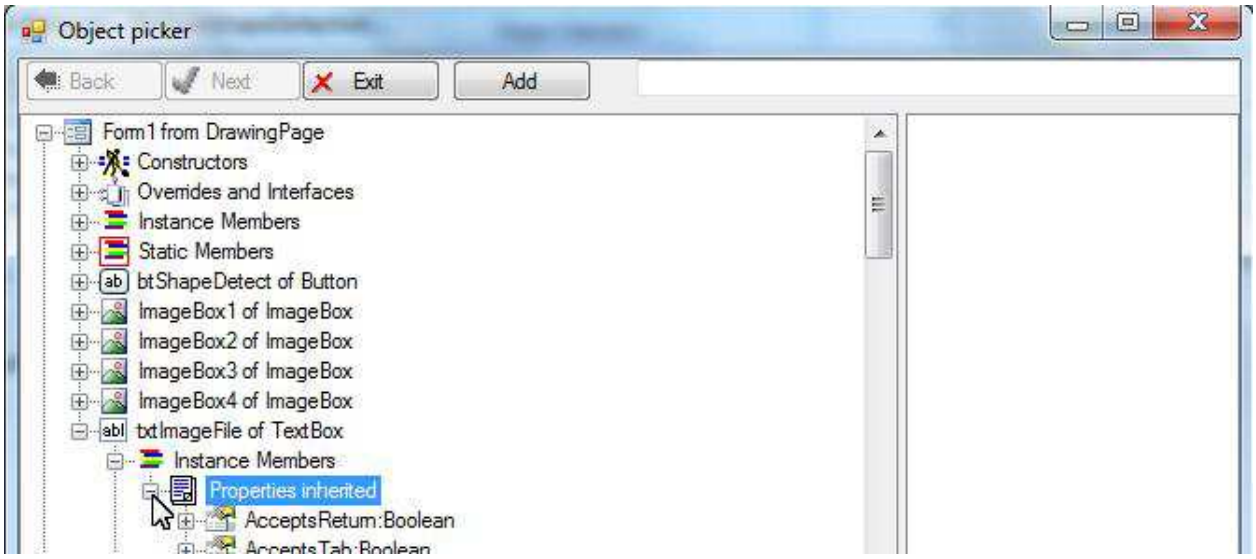


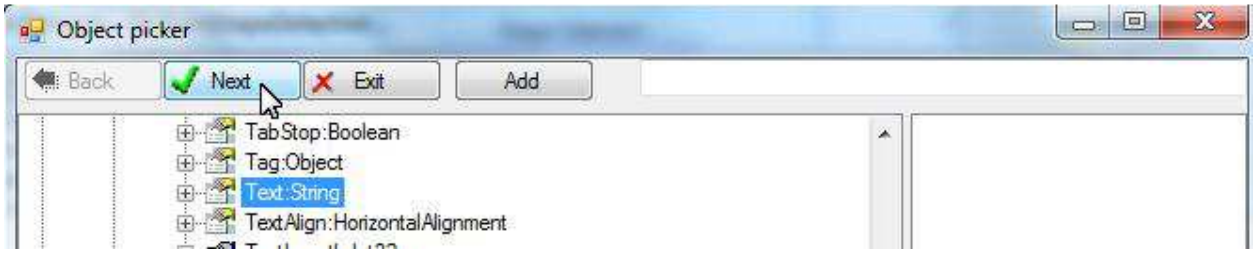
Select the constructor which takes the image file name:



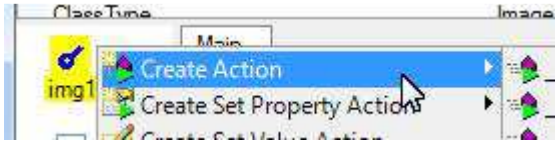
An image variable is created. An action appears. The action creates the image variable. We need to provide the image file name for the action.

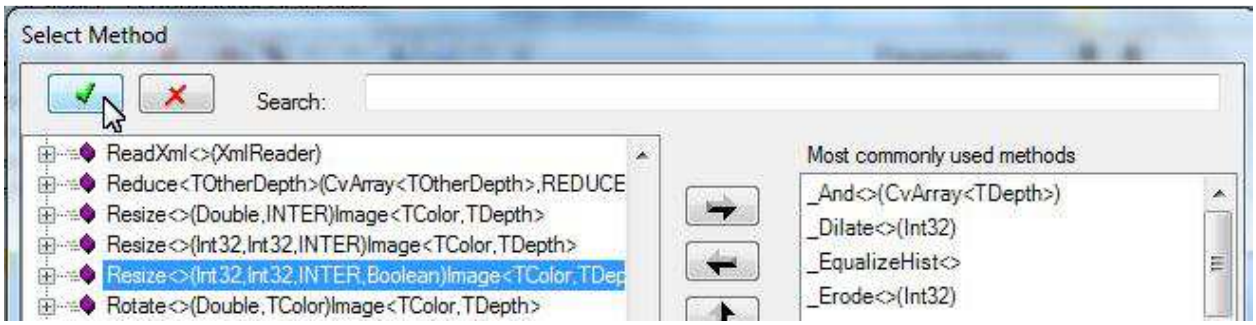




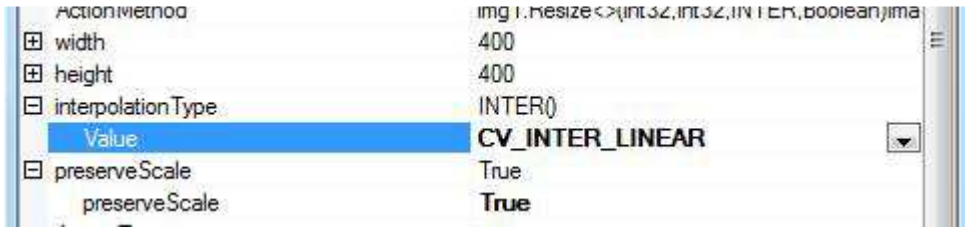


Resize the image to 400x400:

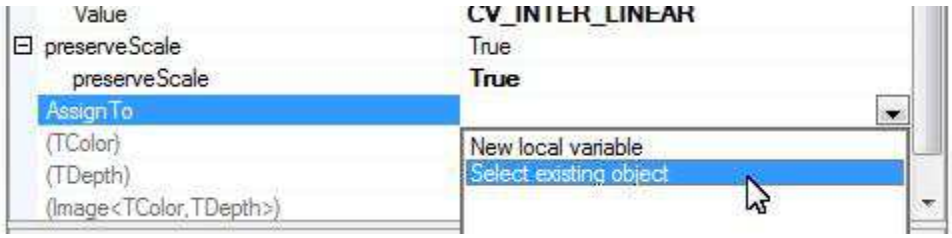




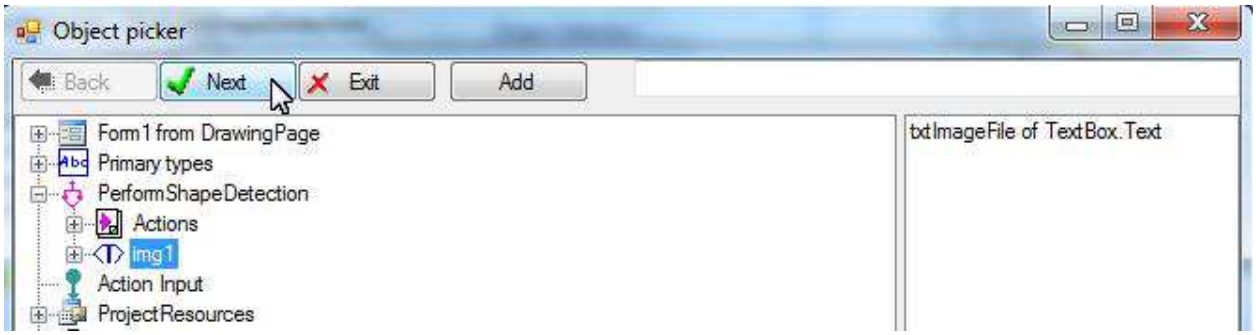
Provide the action parameters values:



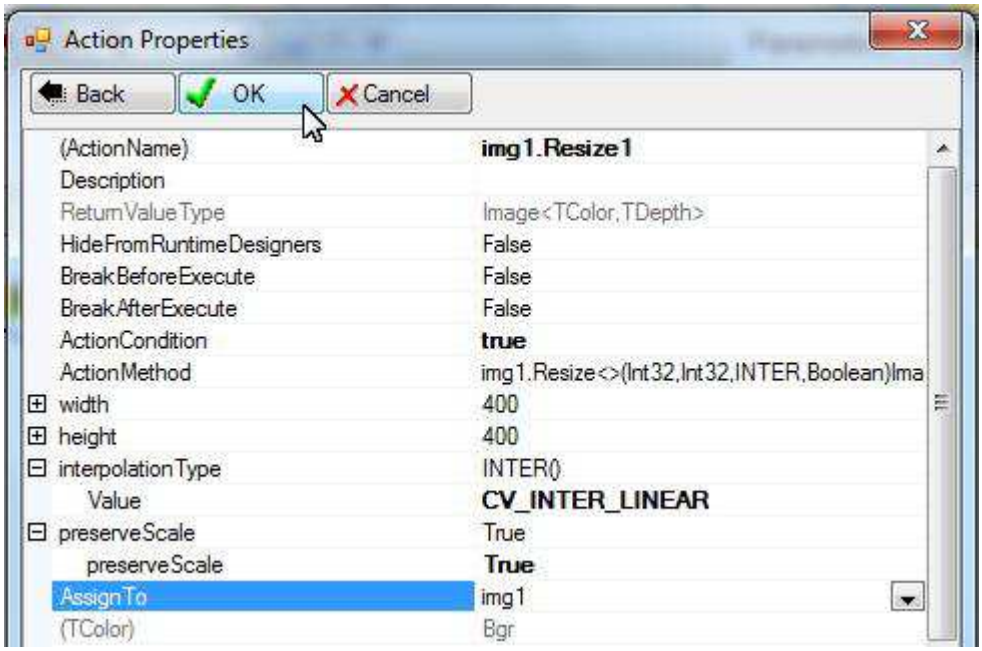
For “AssignTo”, choose “Select existing object”



Select the image variable itself:

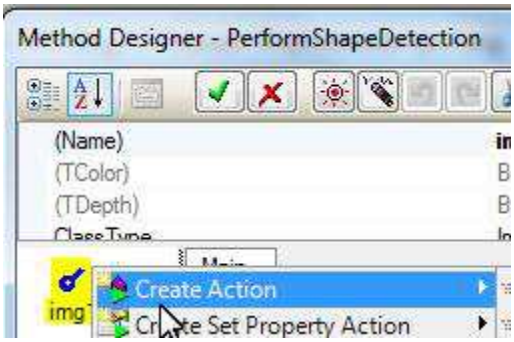


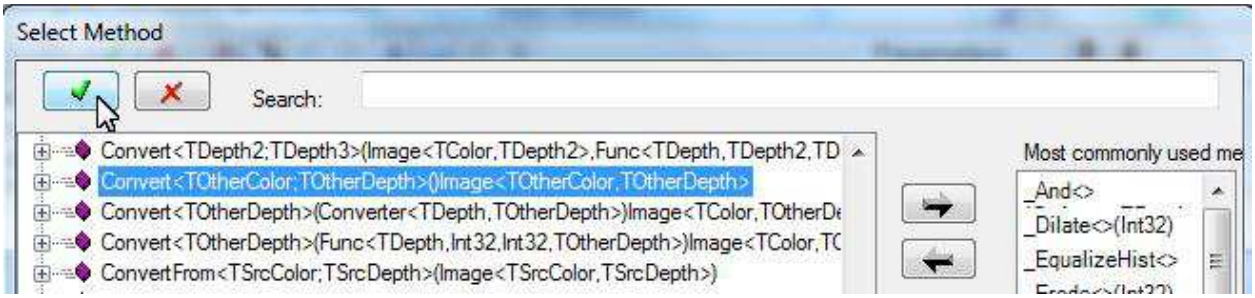
Click OK to finish creating this action:



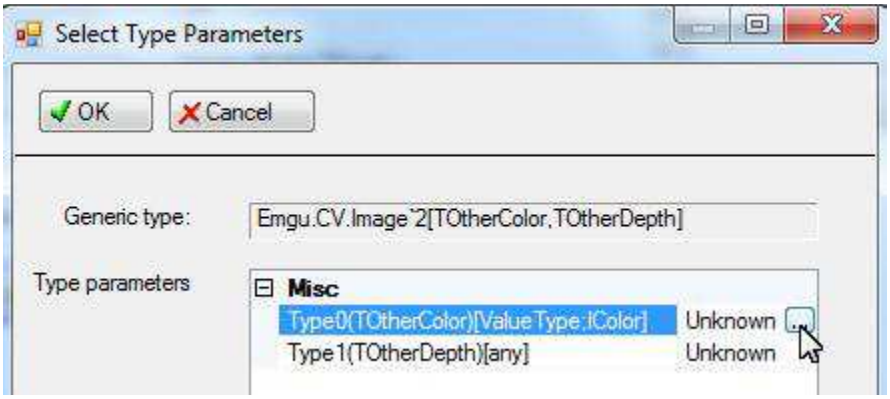
Convert Image to Gray

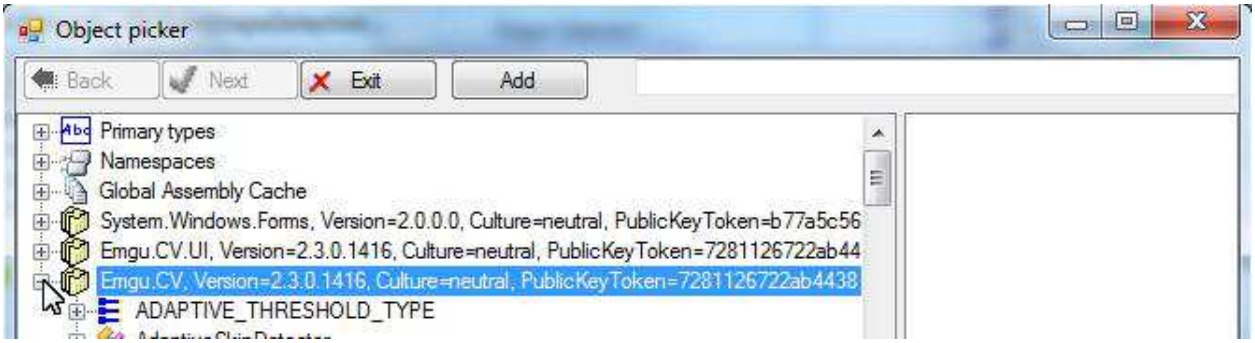
Shape detection needs gray image. A gray image can be generated from an existing image.

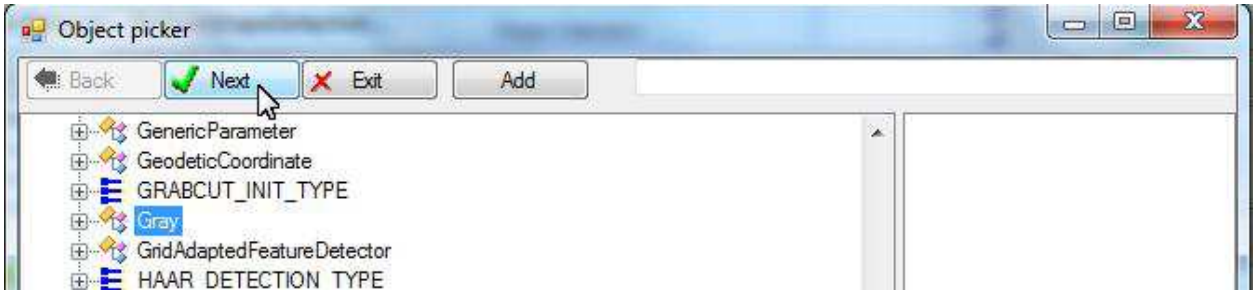




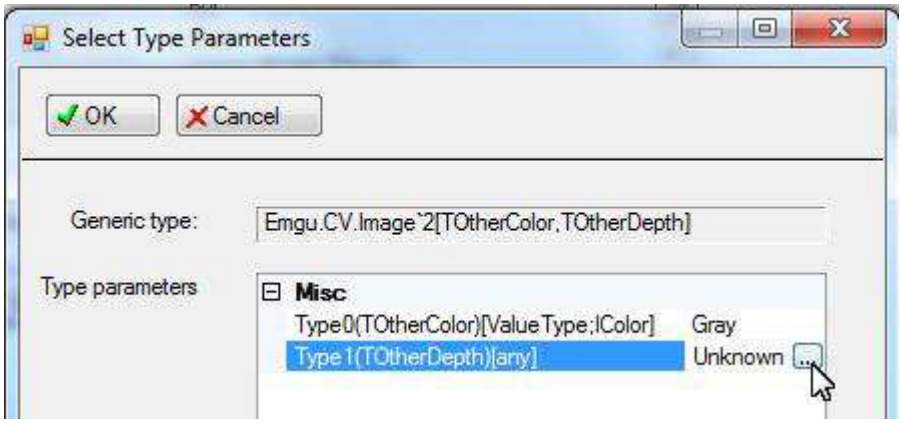
Select Gray as the color:

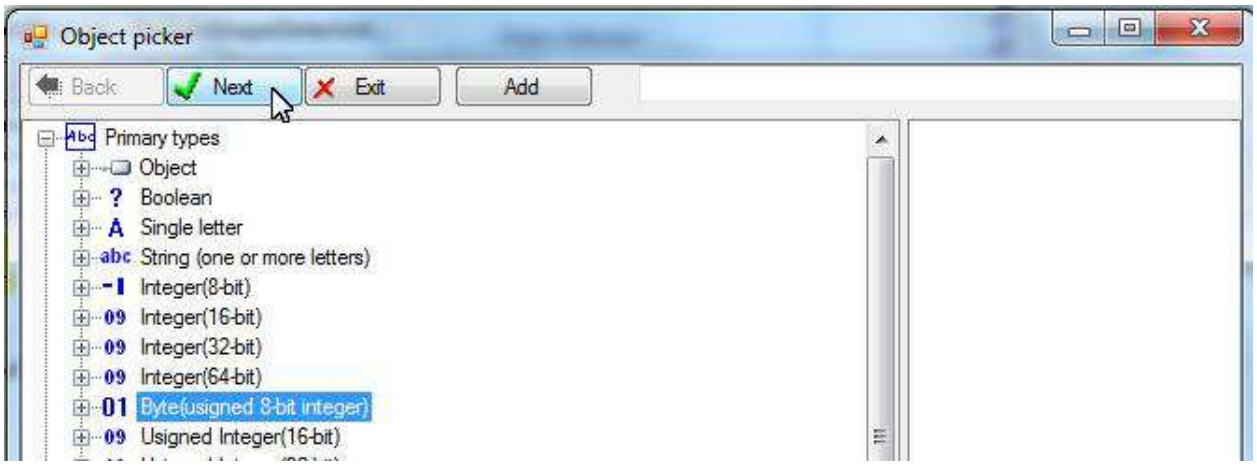


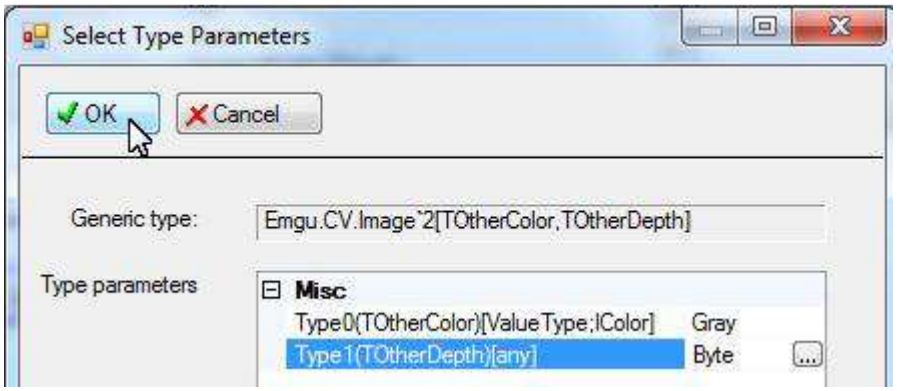




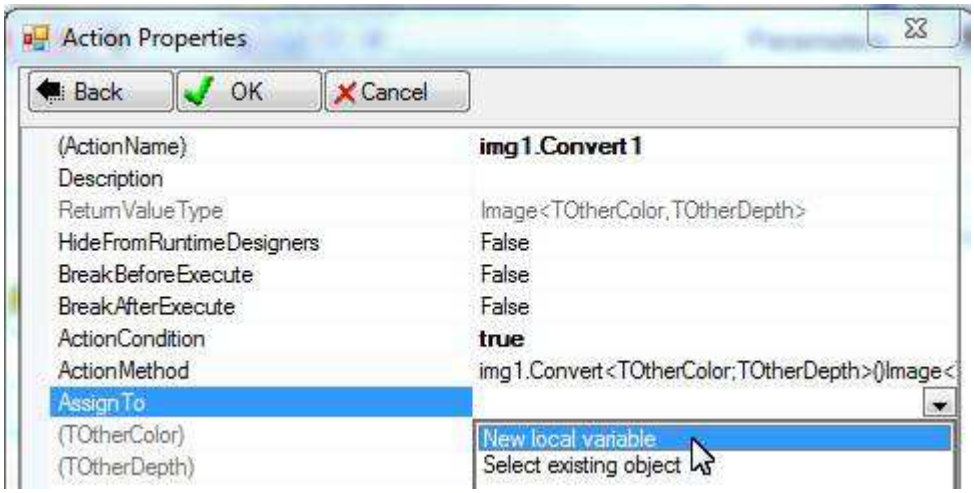
"TOtherDepth" is Byte:



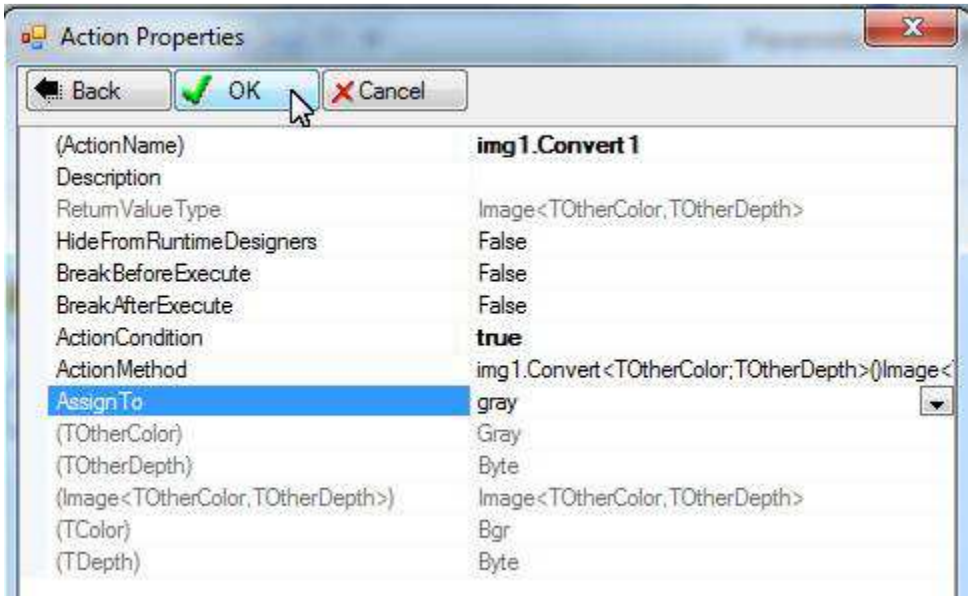




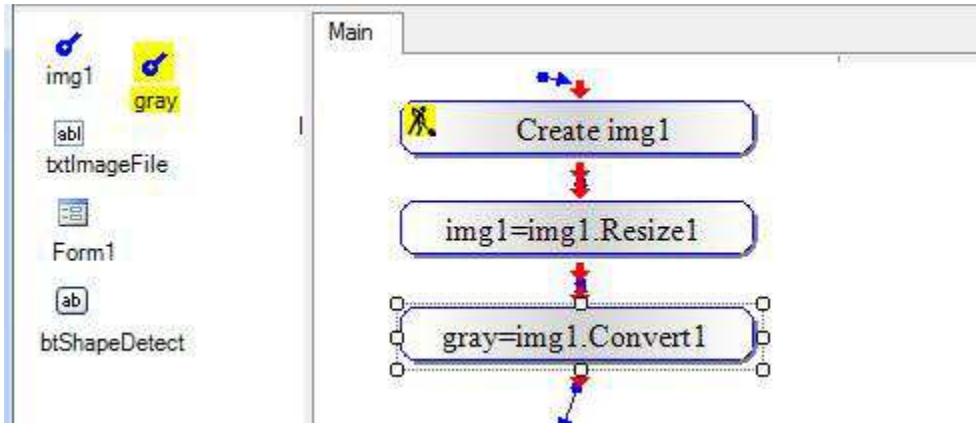
For “AssignTo”, select “New local variable”:



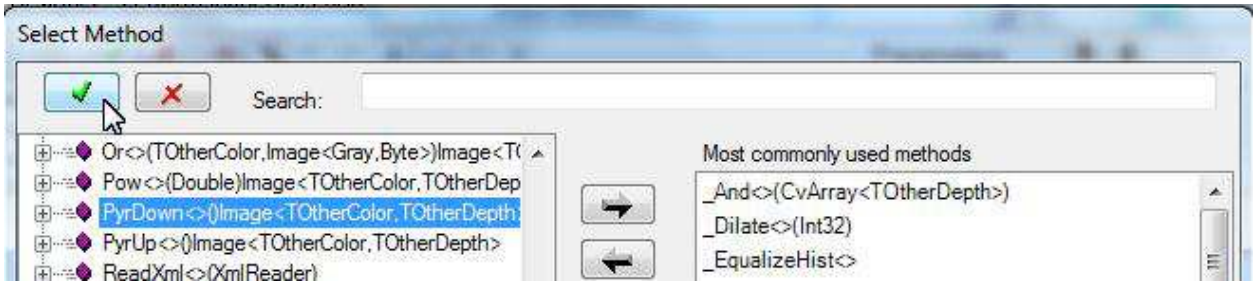
Click OK:



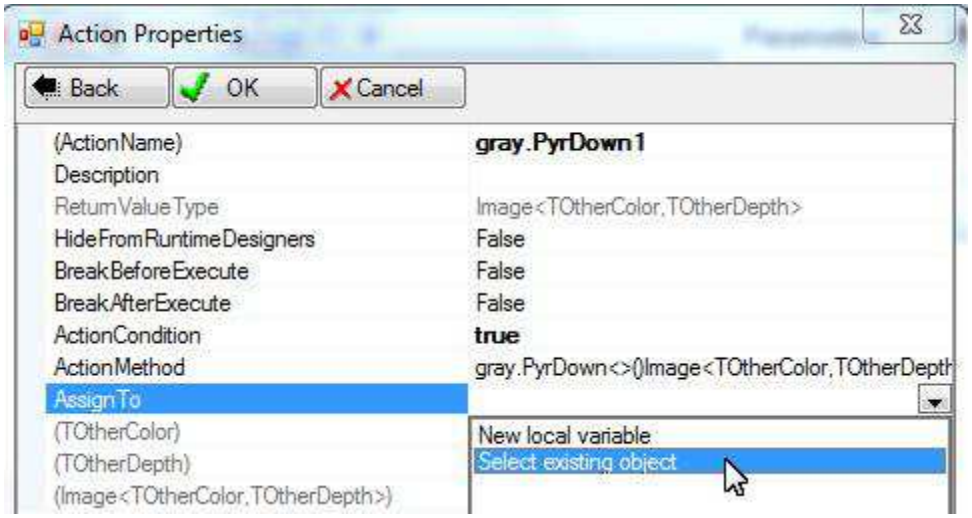
A variable named “gray” appears in the Variable Pane. An action for creating “gray” appears in the Action Pane.

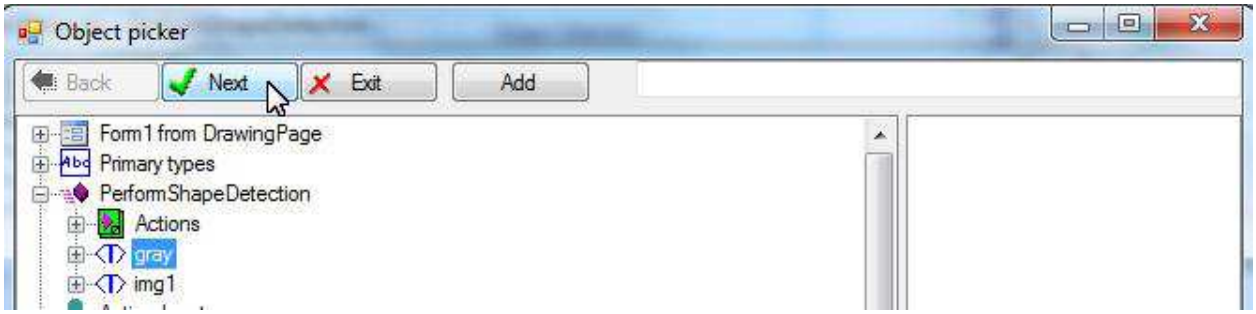


Use PyrDown and PyrUp to create two actions to filter out the noise.

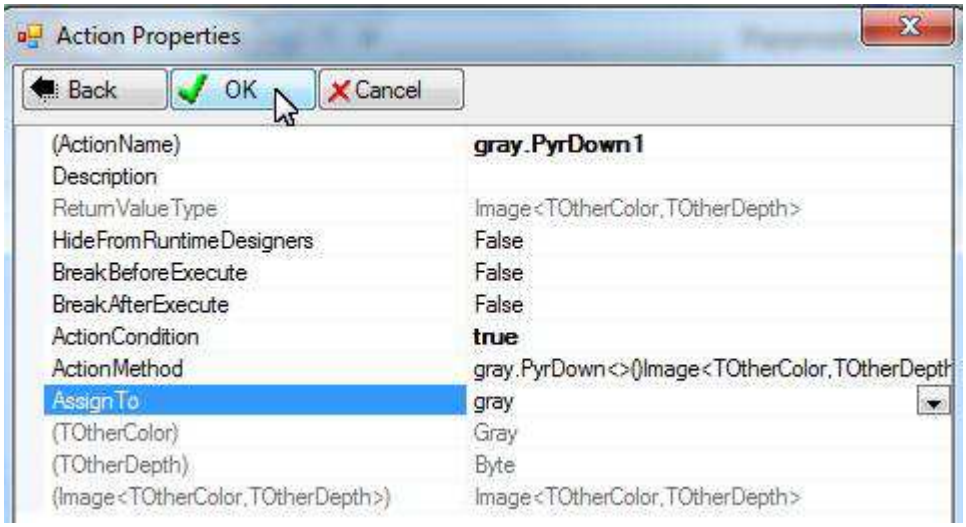


For “AssignTo”, use “Select existing object” and select variable “gray”:

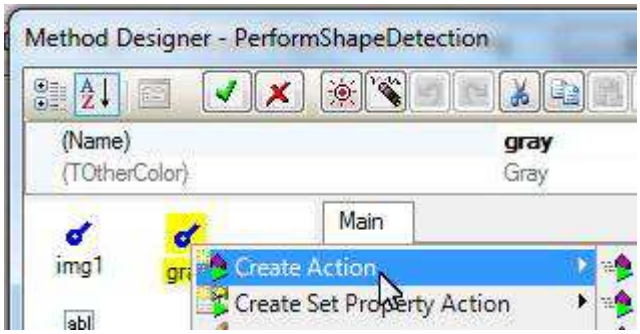


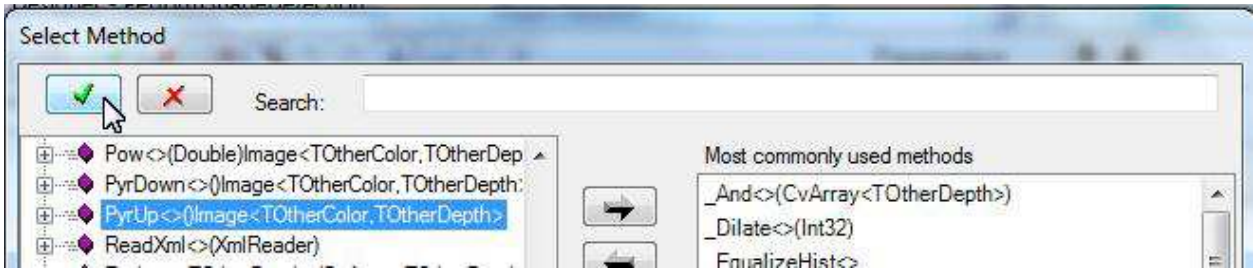


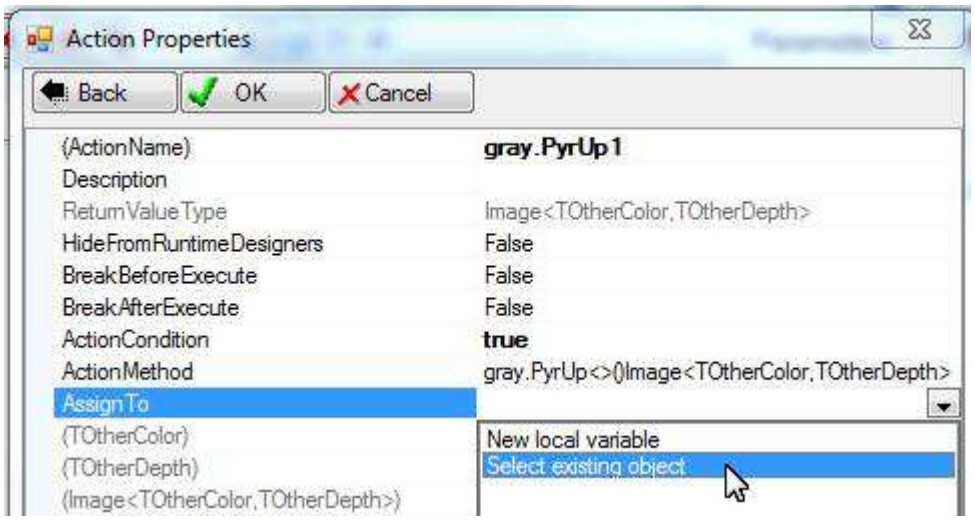
Click OK to finish creating this action:

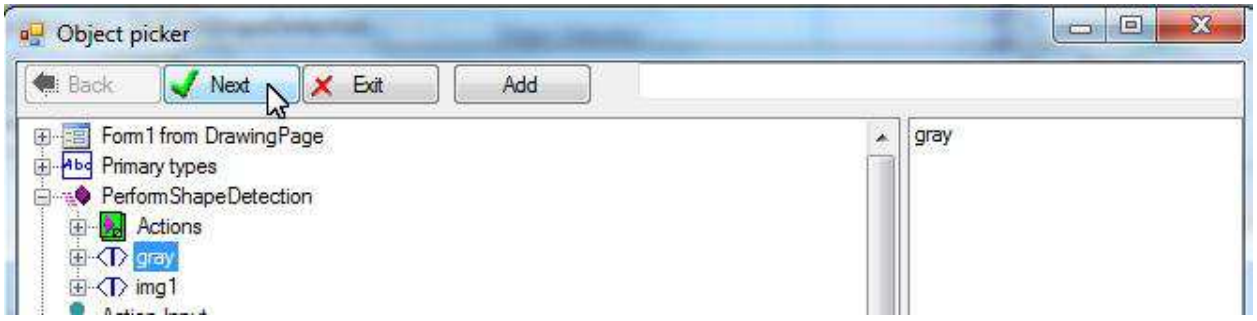


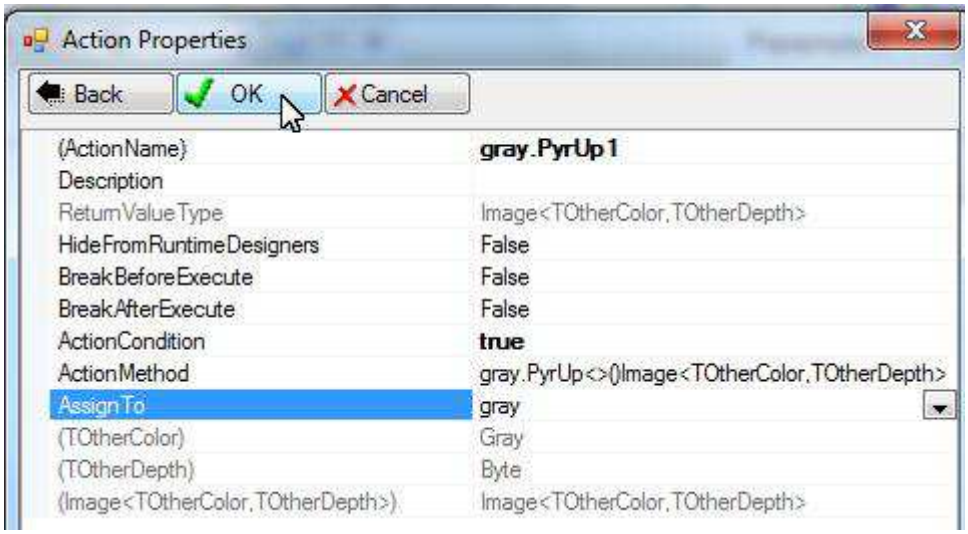
Create PyrUp action:


Method Designer - PerformShapeDetection
(Name)
gray
(TOtherColor)
Gray
Main
img1
Create Action
Create Set Property Action


Select Method
Search:
Pow<>(Double)Image<TOtherColor,TOtherDep
PyrDown<>()Image<TOtherColor,TOtherDepth:
PyrUp<>()Image<TOtherColor,TOtherDepth>
ReadXml<>(XmlReader)
Most commonly used methods
_And<>(CvArray<TOtherDepth>)
_Dilate<>(Int32)


Action Properties
Back
OK
Cancel
(ActionName)
gray.PyrUp1
Description
ReturnValueType
Image<TOtherColor,TOtherDepth>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
gray.PyrUp<>()Image<TOtherColor,TOtherDepth>
Assign To
(TOtherColor)
(TOtherDepth)
(Image<TOtherColor,TOtherDepth>)
New local variable
Select existing object

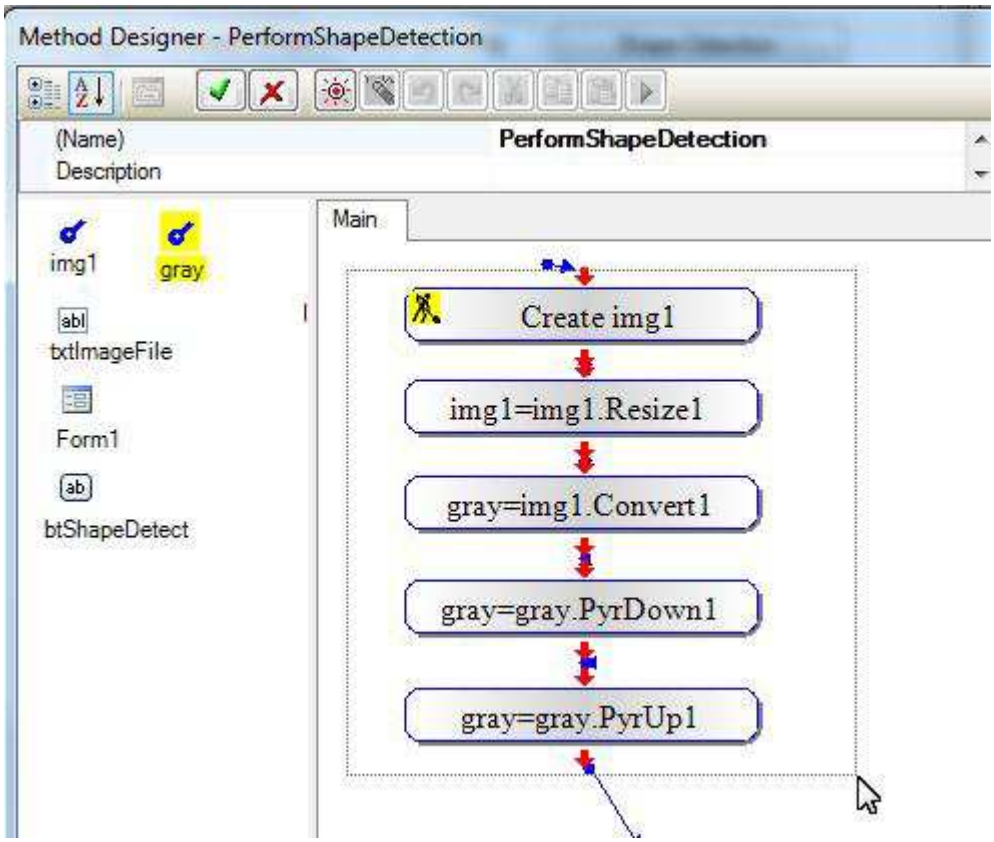

Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
gray
img1
gray



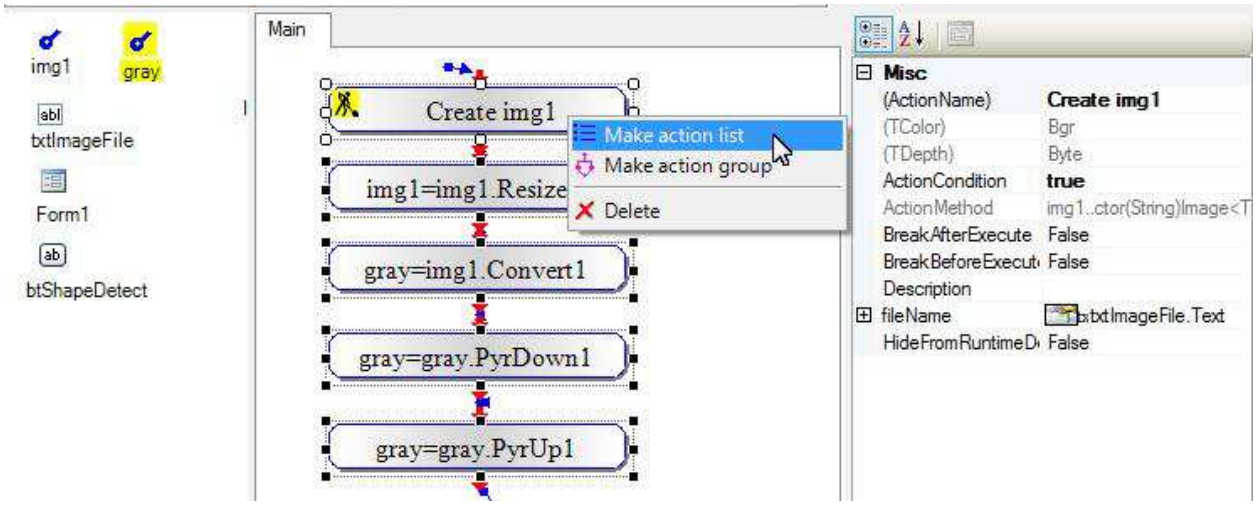
Create Action List

By now the Action Pane is pretty much full. We may create an action list to include all actions.

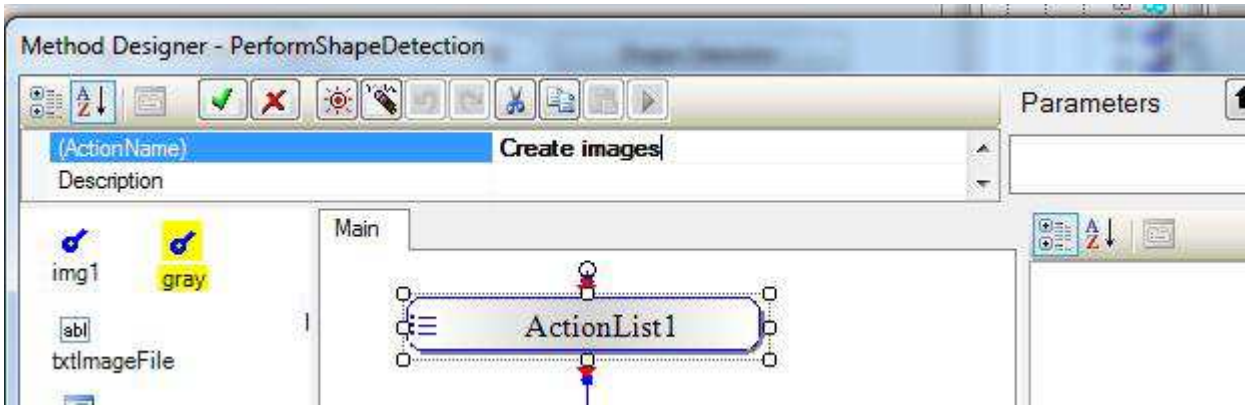
Select all the actions:



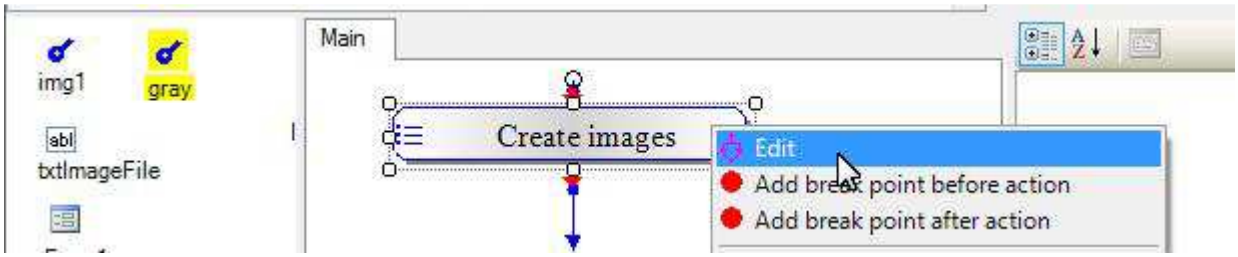
Right-click selected actions; choose “Make action list”:



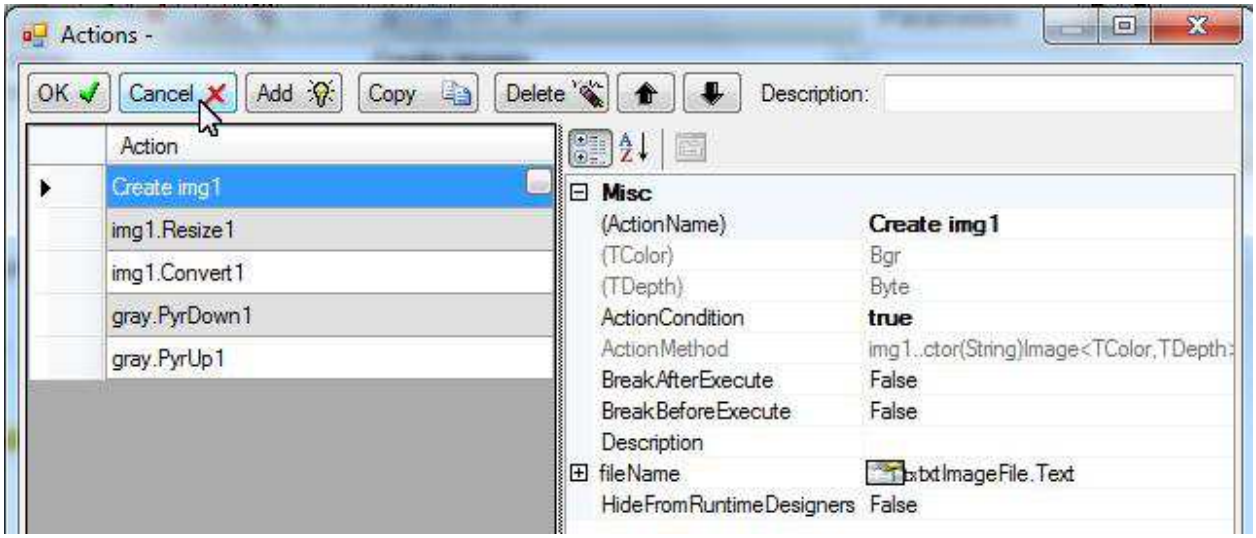
An action list is created. We may rename it to “Create images”:



To modify it, right-click it and choose “Edit”:

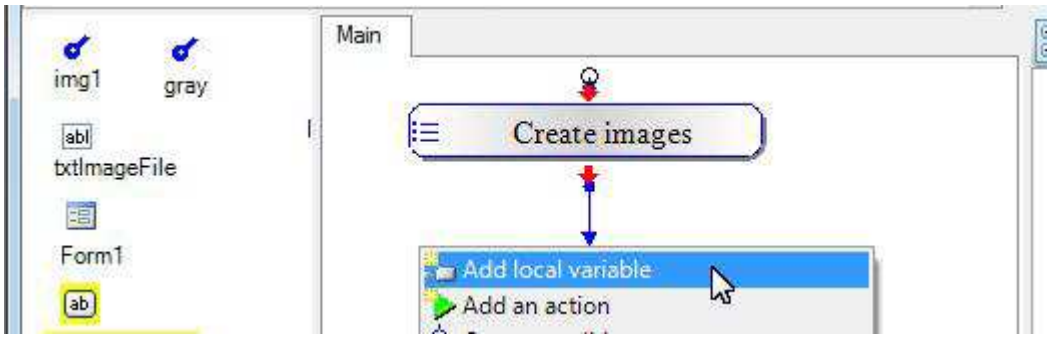


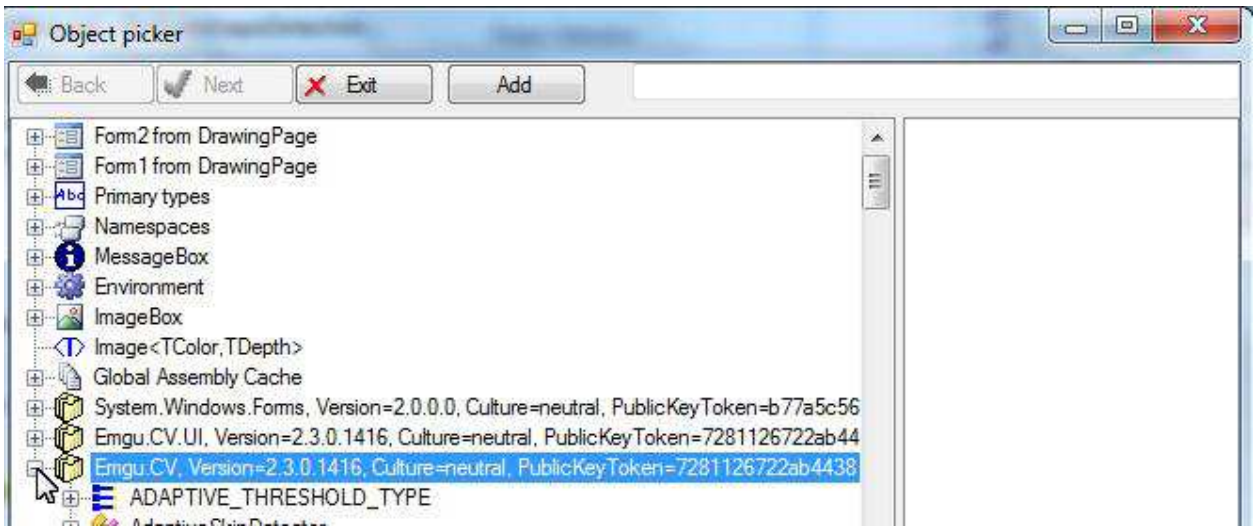
A large number of actions can be included. You may change the action order and modify each action.

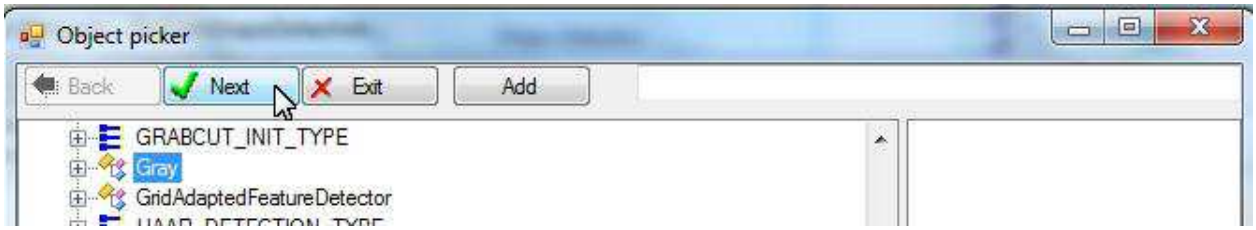


Circles Detect

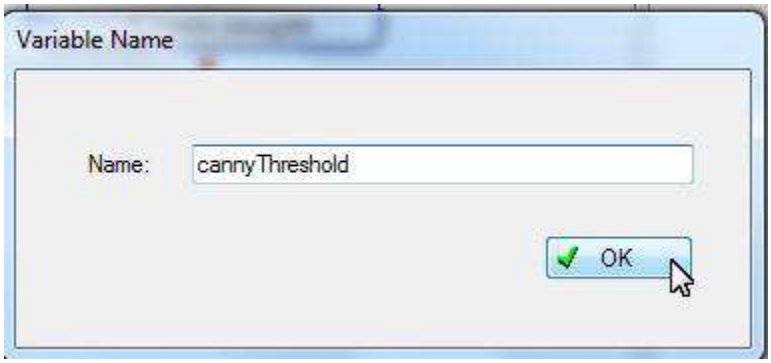
A gray image's HoughCircles method is used to detect circles. This method needs two Gray objects. Let's create two Gray objects first.



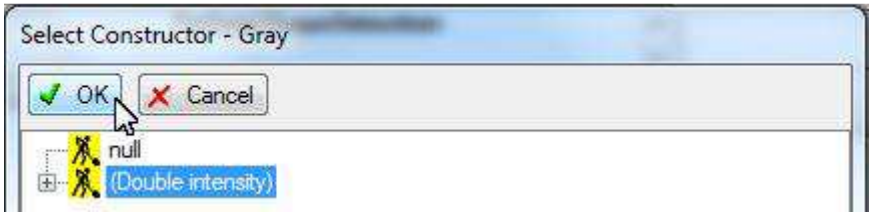




Give a variable name:

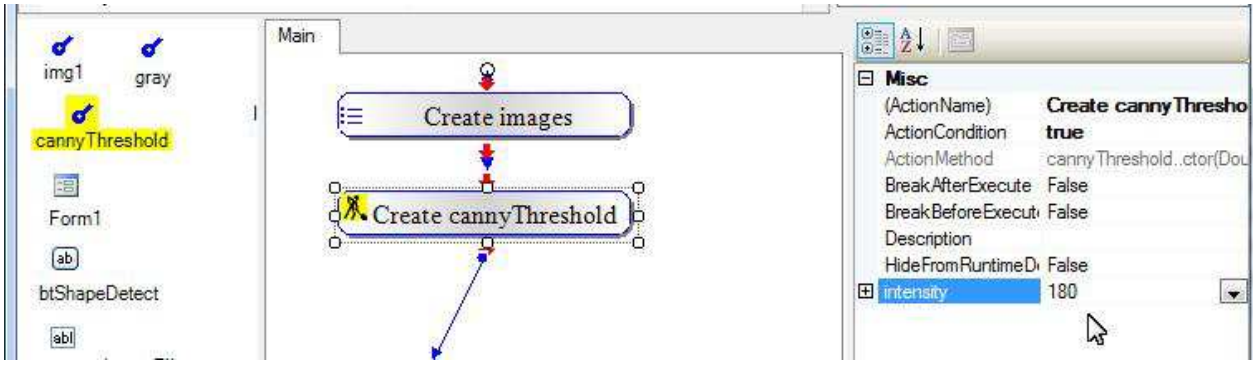


Select the constructor:

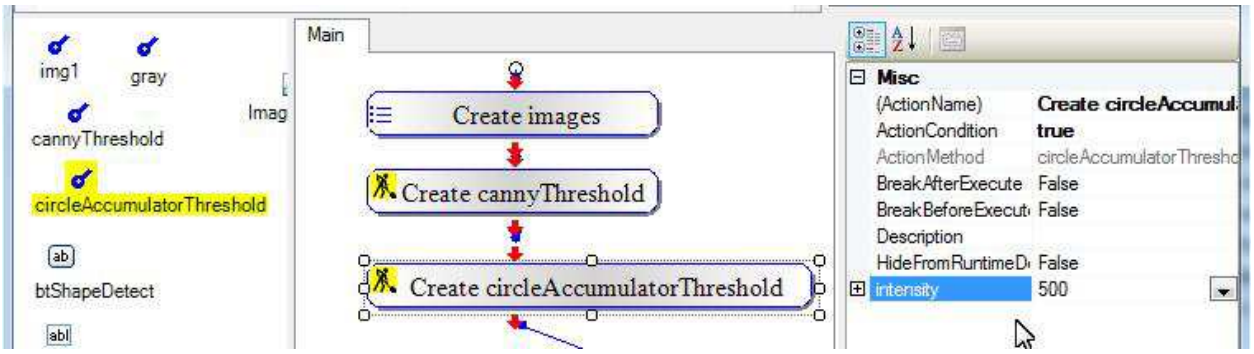


The variable “cannyThreshold” appears in the Variable Pane. An action appears in the Action Pane for creating the variable.

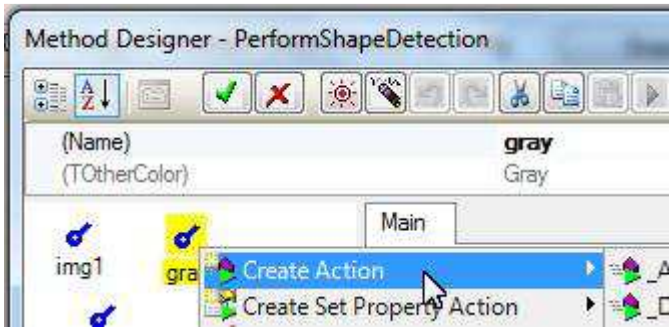
According to the tutorial from Emgu, 180 is used for “cannyThreshold”:



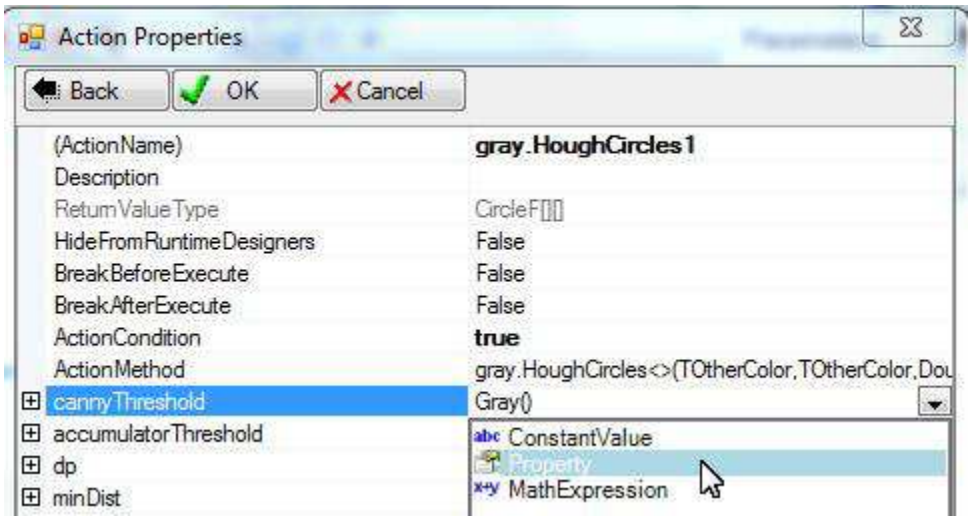
Create another Gray object with value 500:

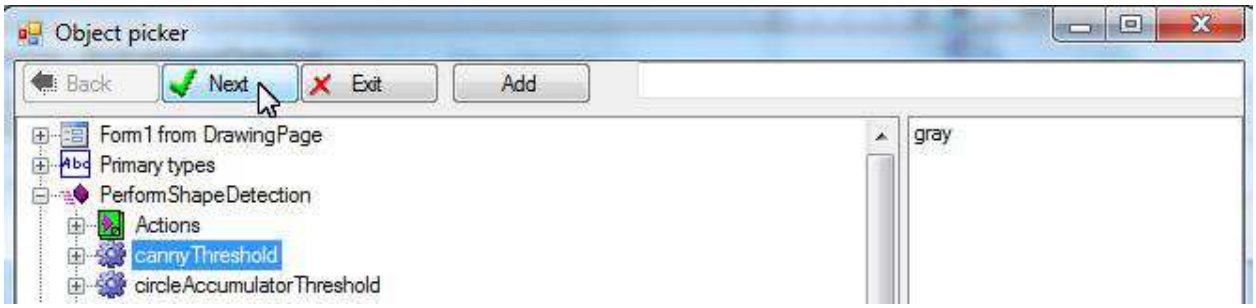


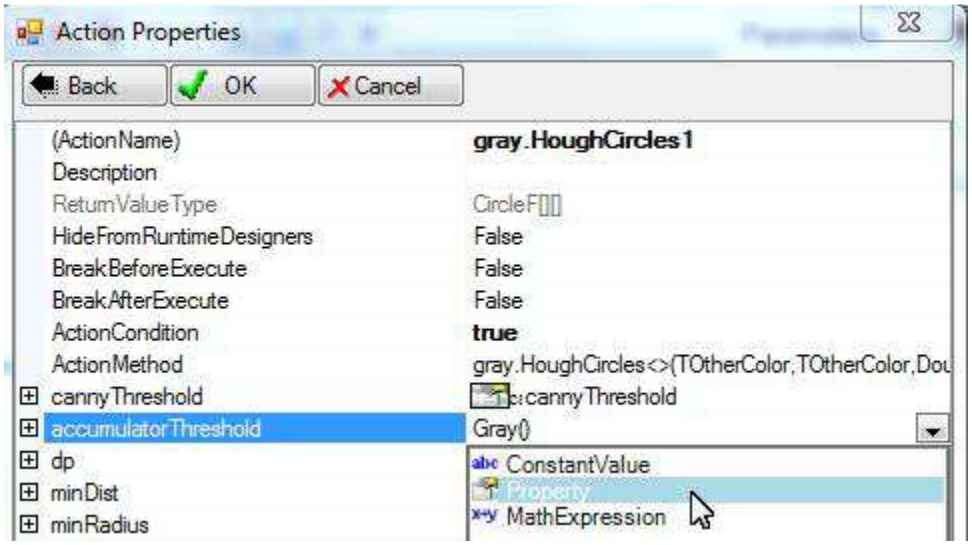
Now we may use variable “gray” to create HoughCircles action:

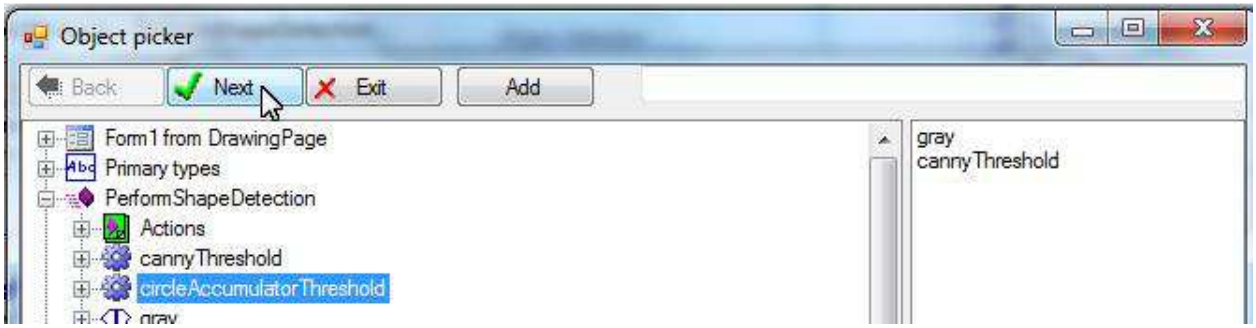


Assign to two Gray objects to the action:



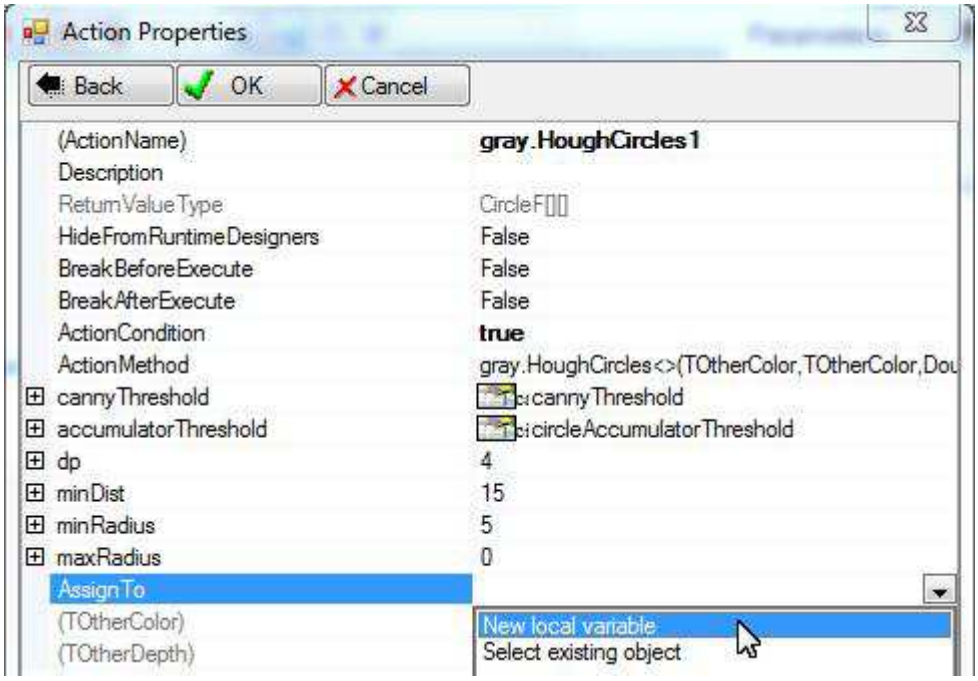




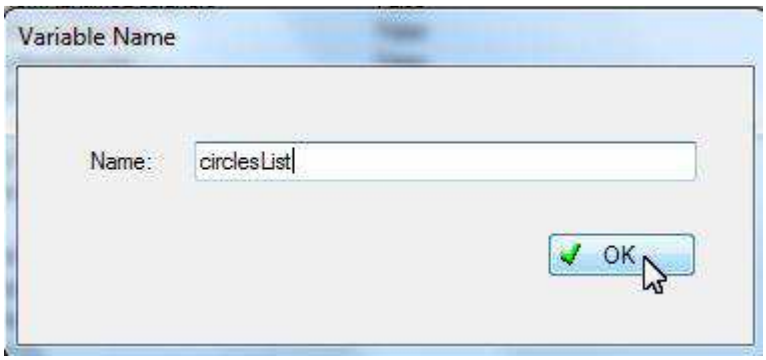


Set other parameter values according to the Emgu tutorial.

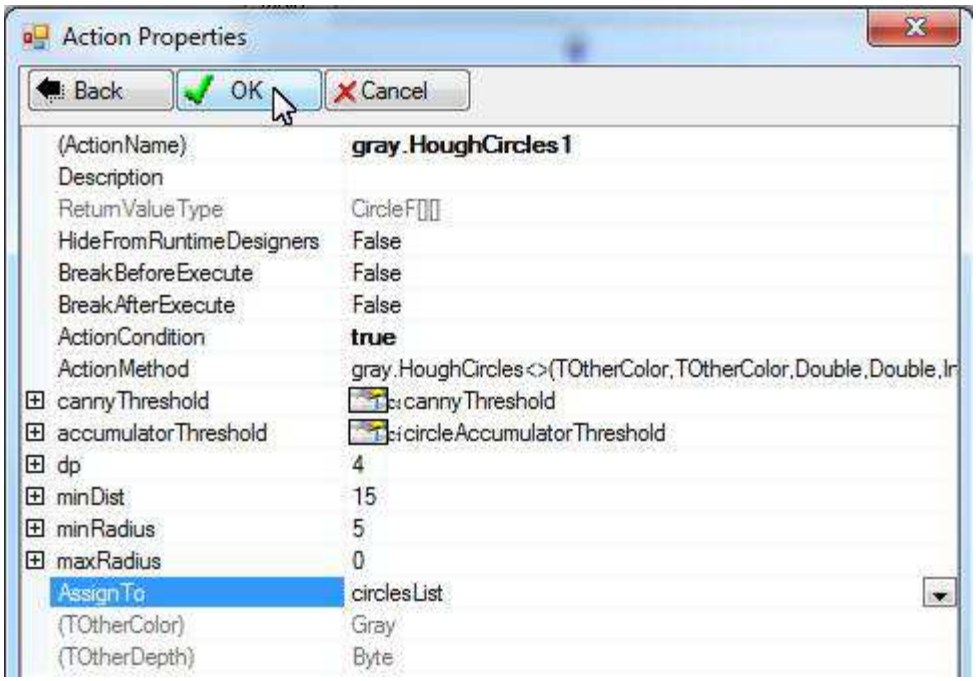
For “AssignTo”, select “New local variable”:



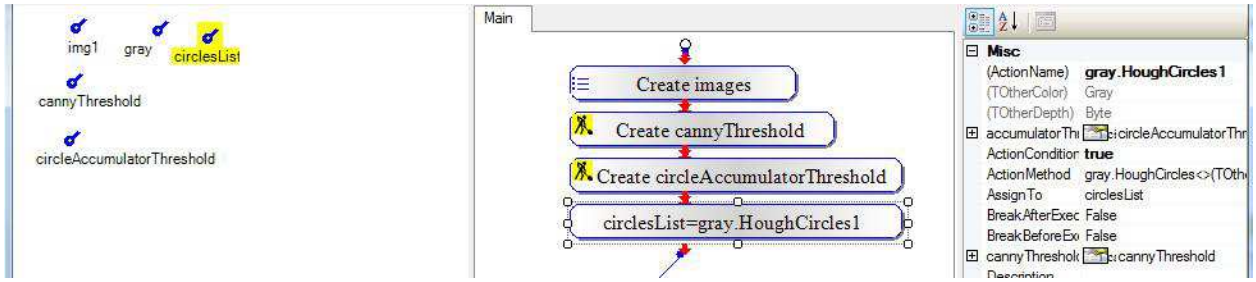
Give a new variable name:



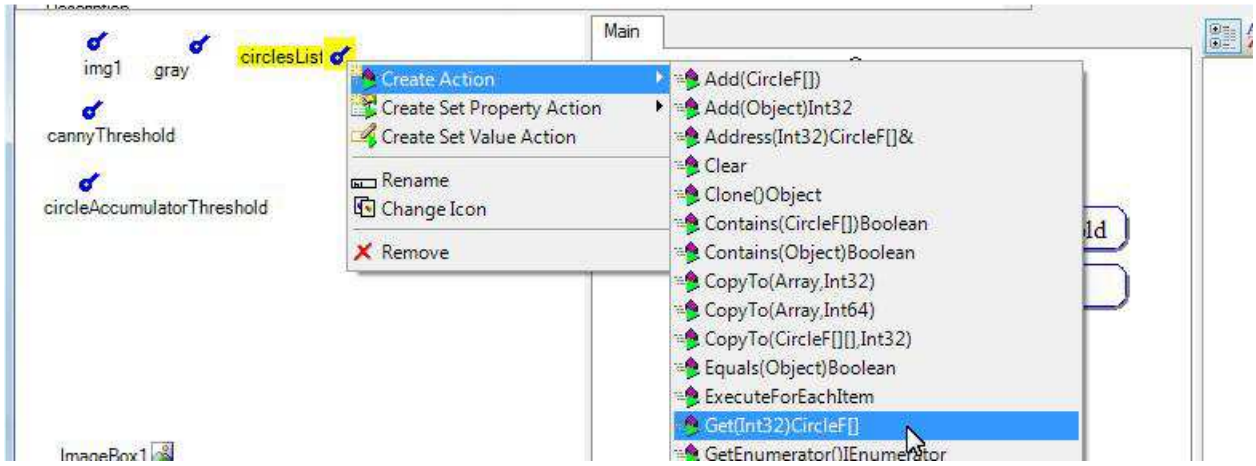
Click OK to finish creating this action:

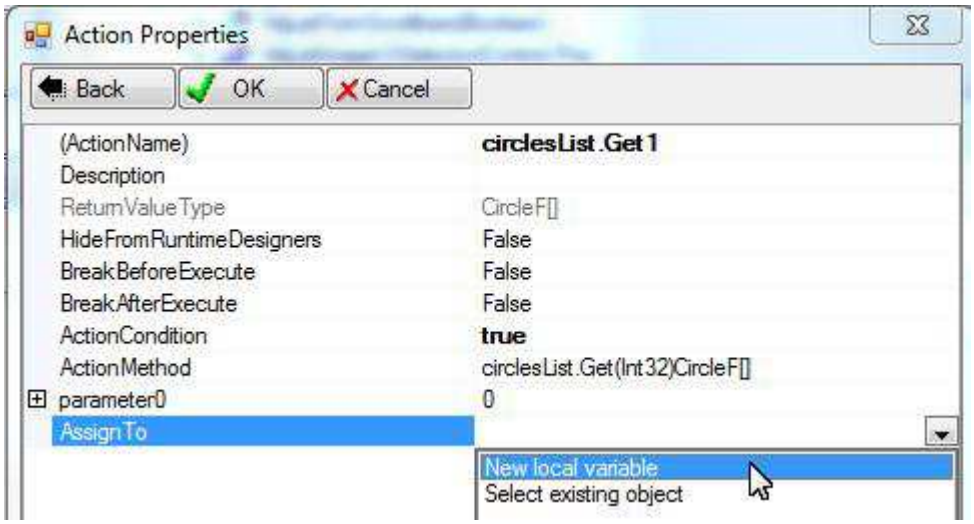


A new variable, “circlesList”, appear in the Variable Pane. A new action creating “circlesList” appears in the Action Pane.

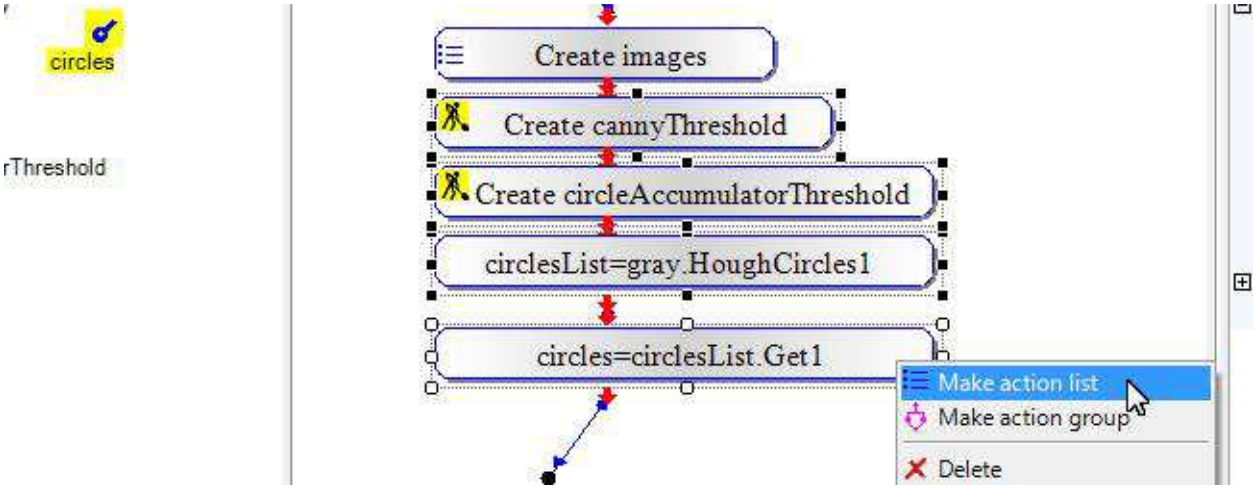


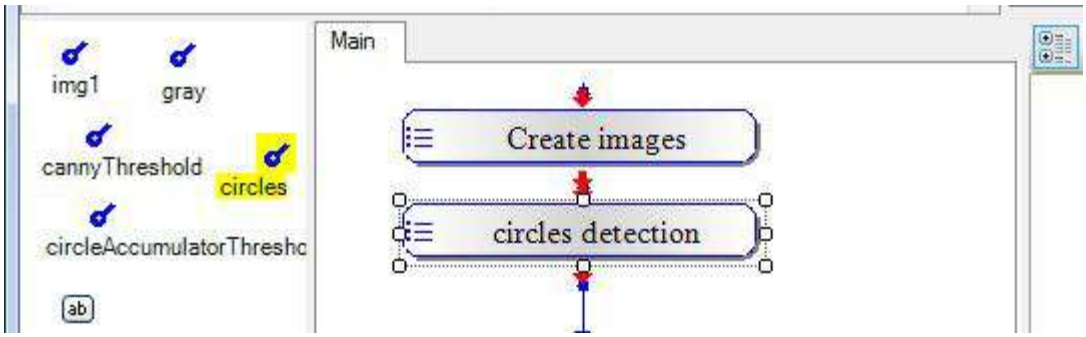
Get the first circle array from “circlesList”:





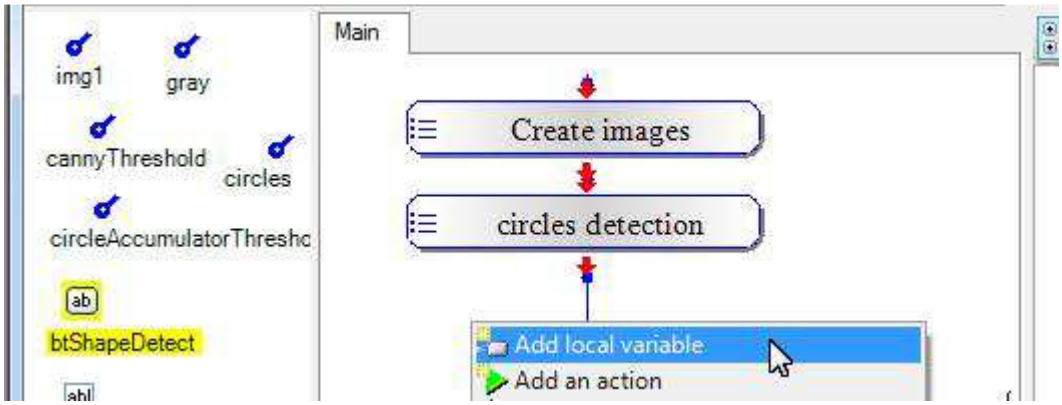
We may group the above 4 action into a “circles detection” action list to make room for the editor:

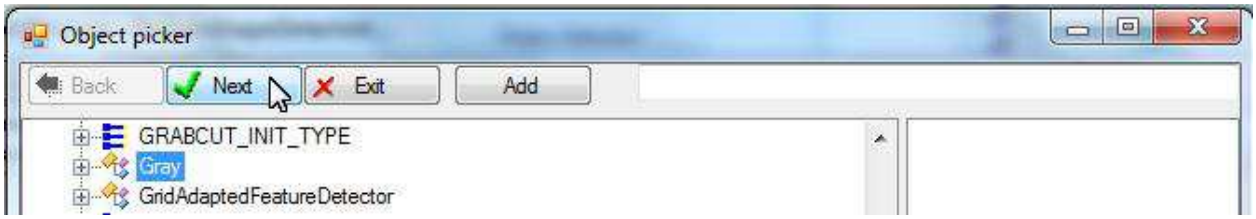


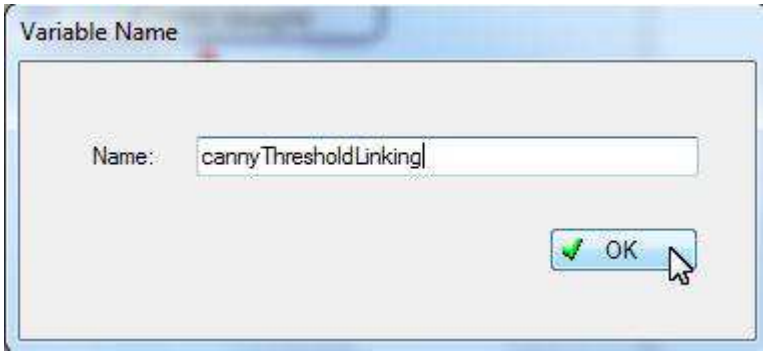


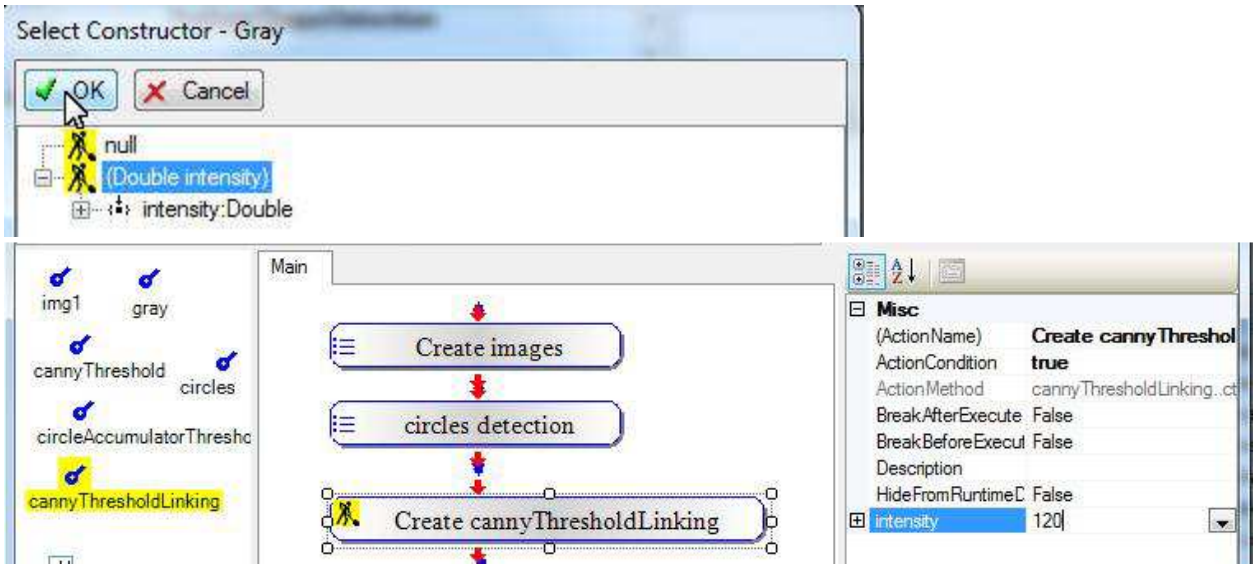
Lines Detection

Create a Gray object named "cannyThresholdLinking"

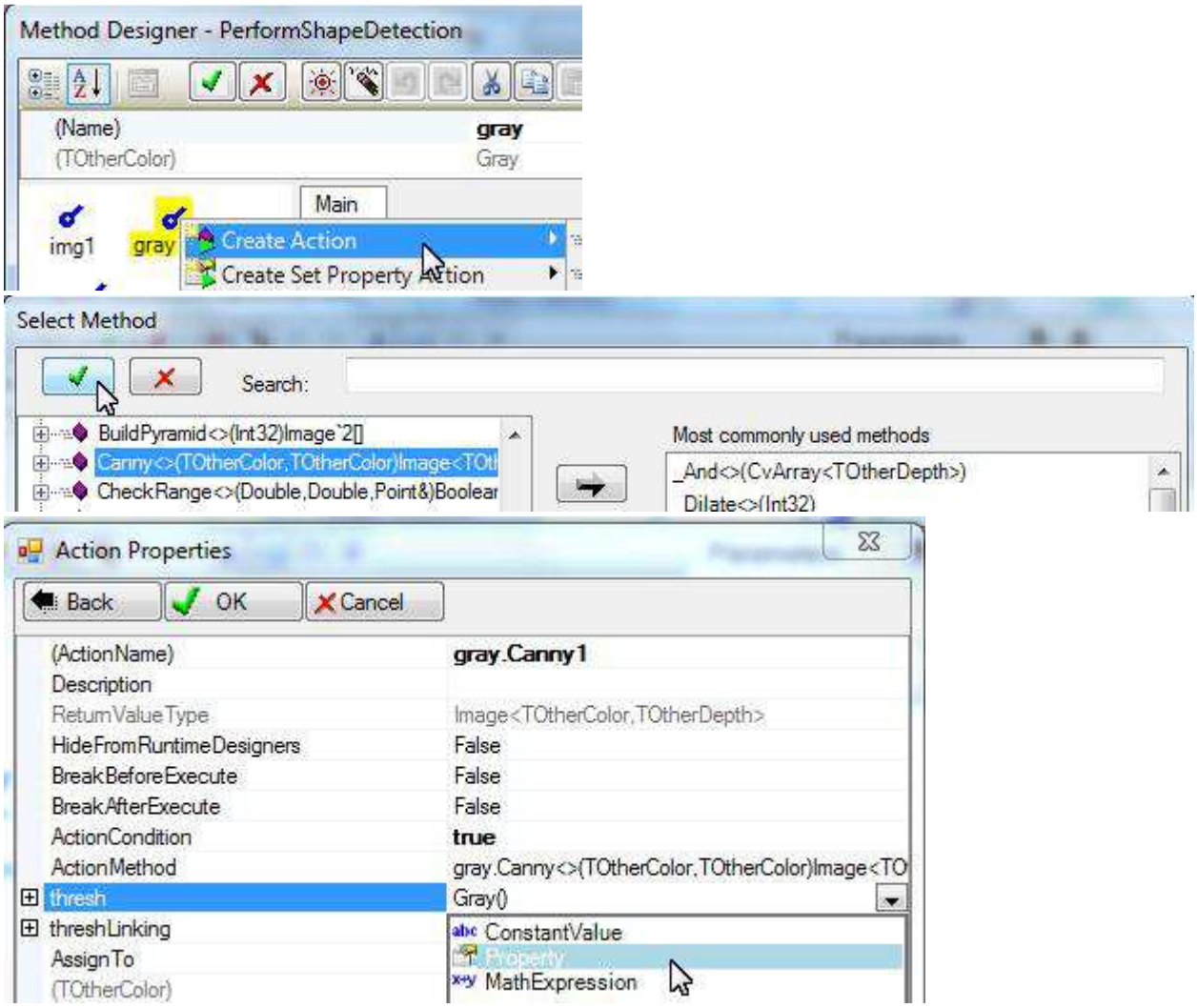


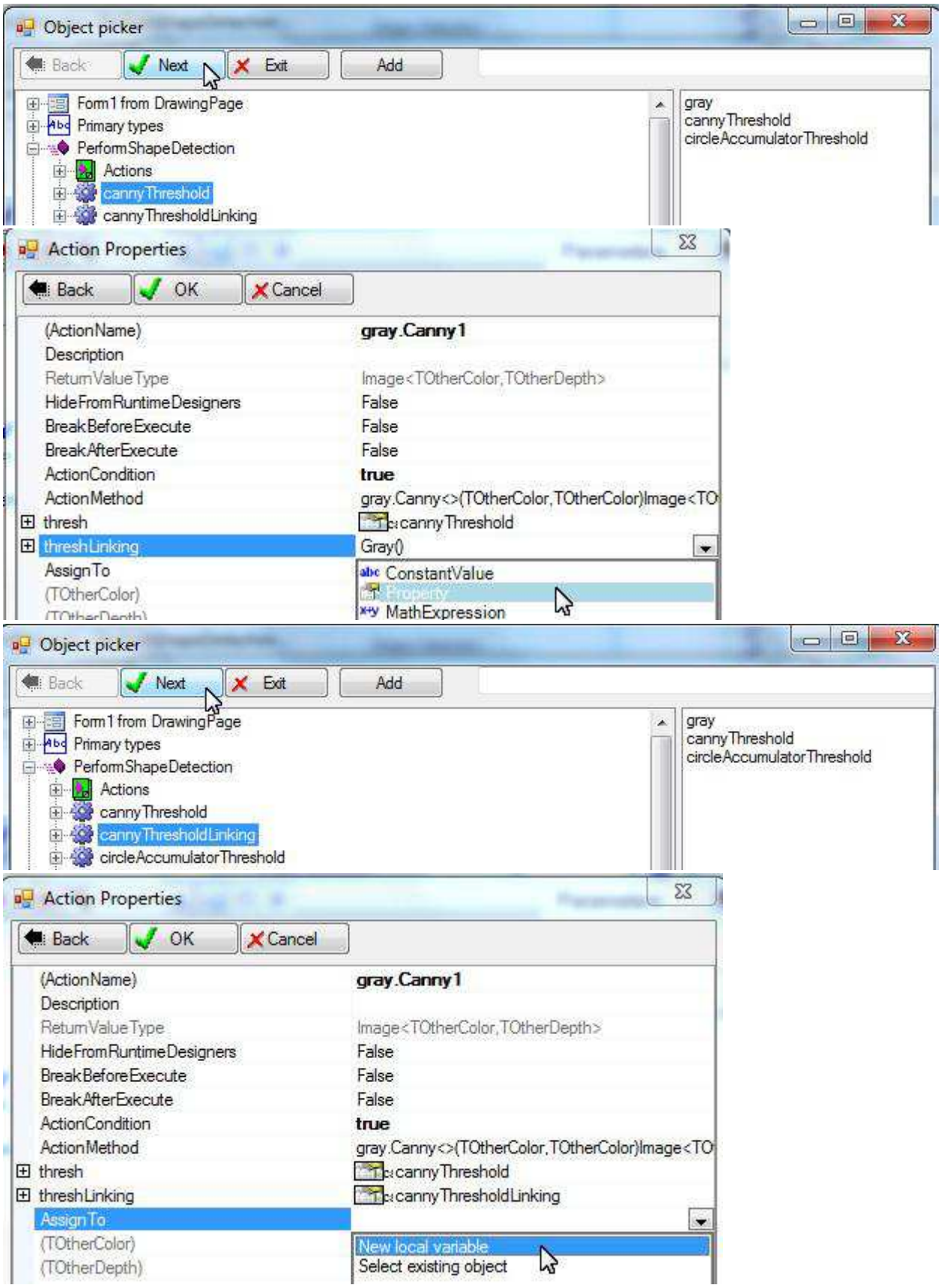


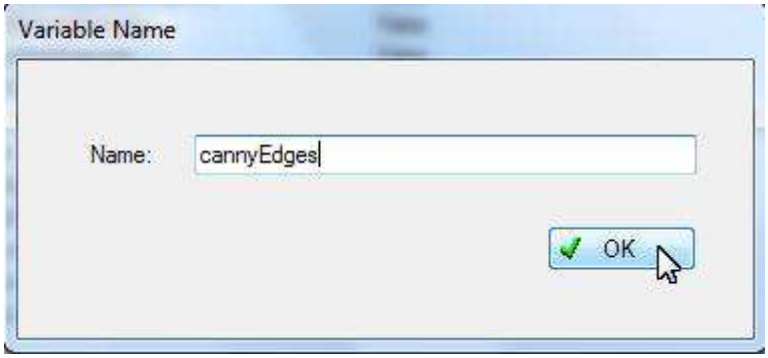


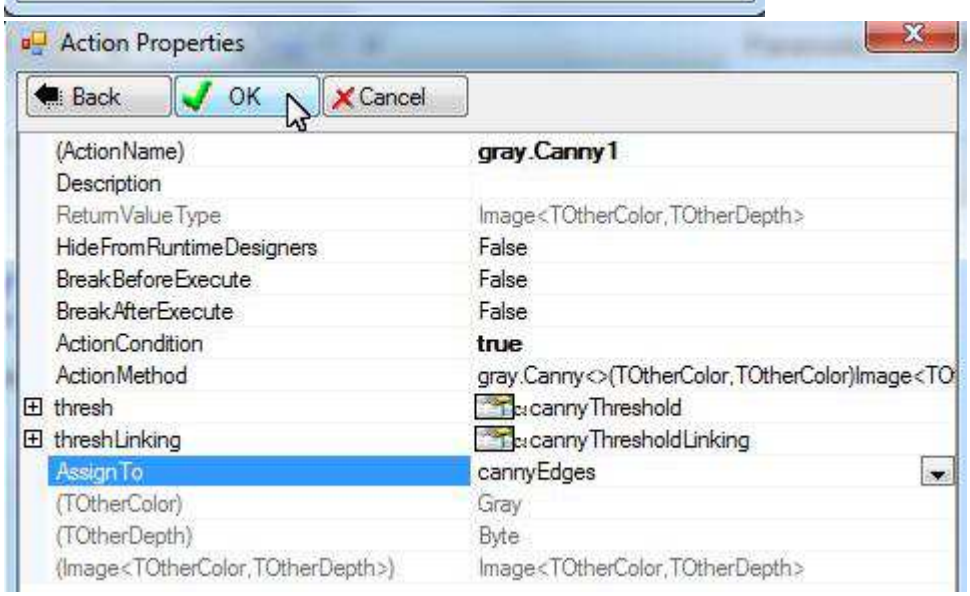


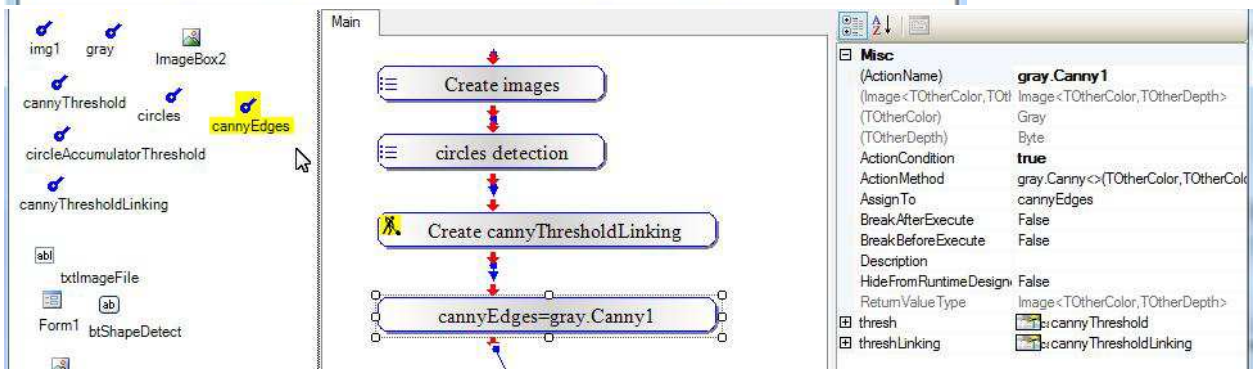
Create a Gray image named "cannyEdges".



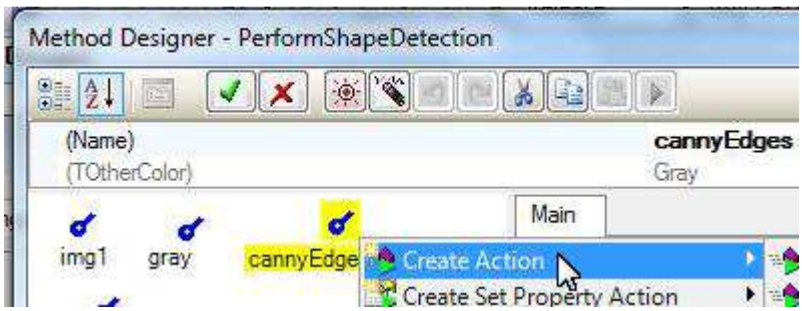

Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
cannyThreshold
cannyThresholdLinking
gray
cannyThreshold
circleAccumulatorThreshold
Action Properties
Back
OK
Cancel
(ActionName)
gray.Canny1
Description
ReturnValueType
Image<TOtherColor,TOtherDepth>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
gray.Canny<>(TOtherColor,TOtherColor)Image<TO
thresh
cannyThreshold
threshLinking
Gray()
AssignTo
(TOtherColor)
ConstantValue
Property
MathExpression
Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
gray
cannyThreshold
circleAccumulatorThreshold
Action Properties
Back
OK
Cancel
(ActionName)
gray.Canny1
Description
ReturnValueType
Image<TOtherColor,TOtherDepth>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
gray.Canny<>(TOtherColor,TOtherColor)Image<TO
thresh
cannyThreshold
threshLinking
cannyThresholdLinking
AssignTo
(TOtherColor)
(TOtherDepth)
New local variable
Select existing object

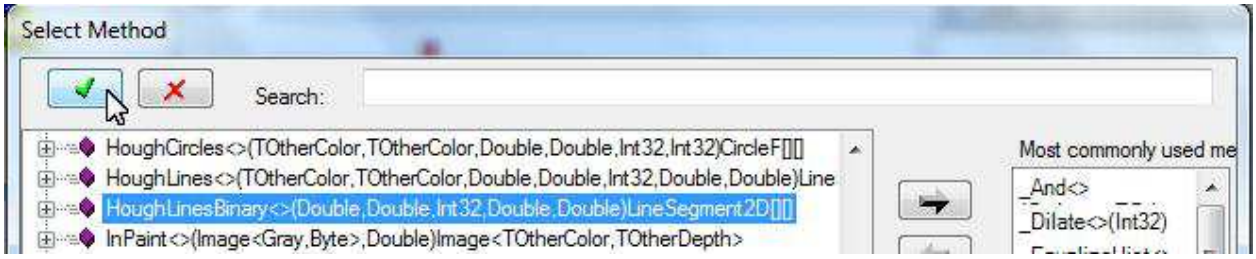




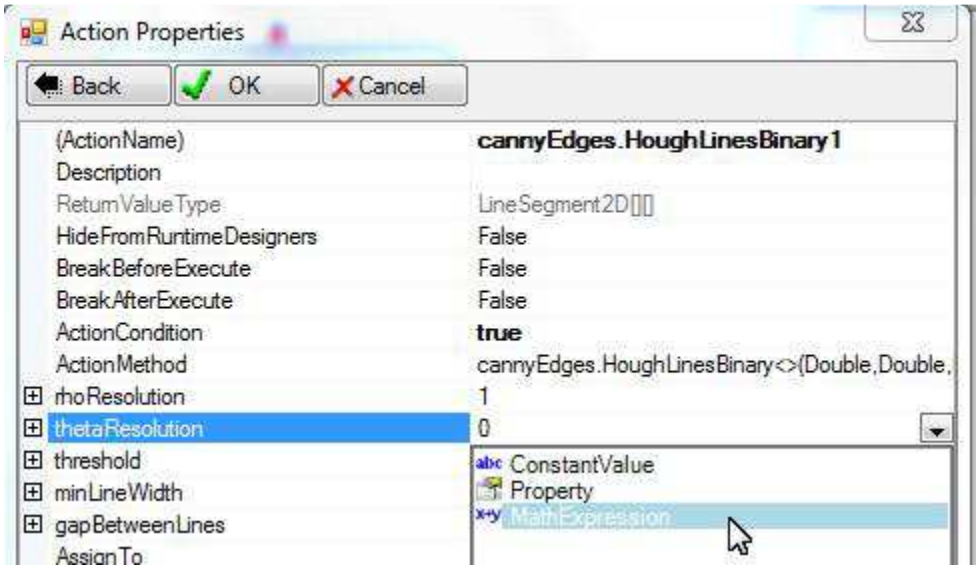


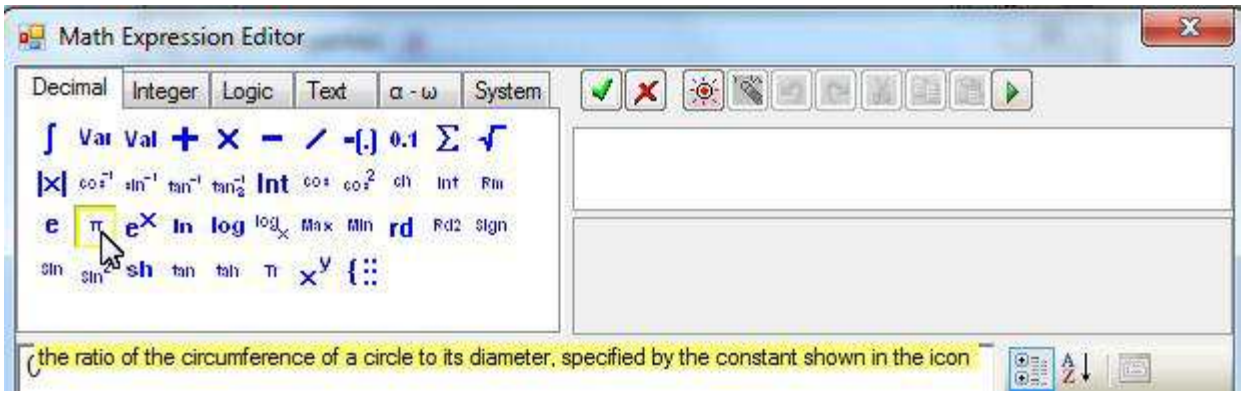
Detect lines

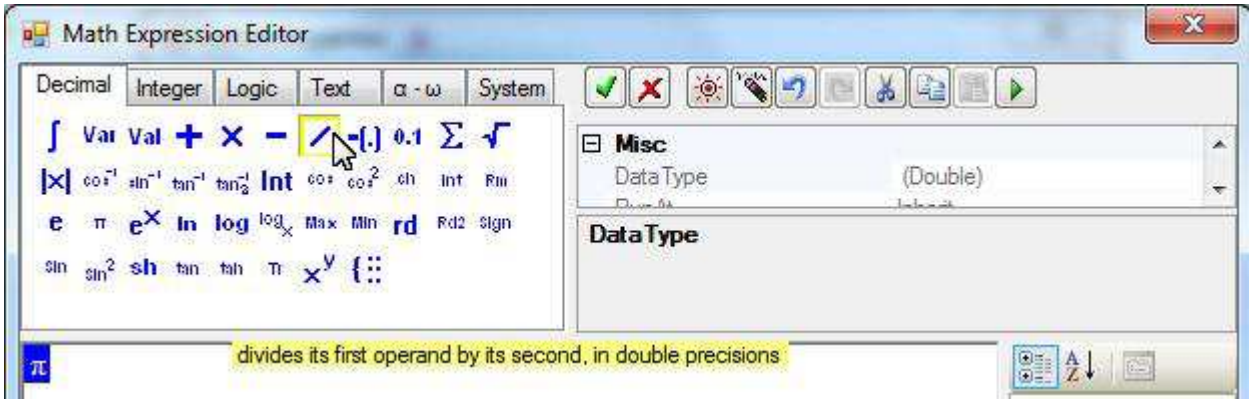


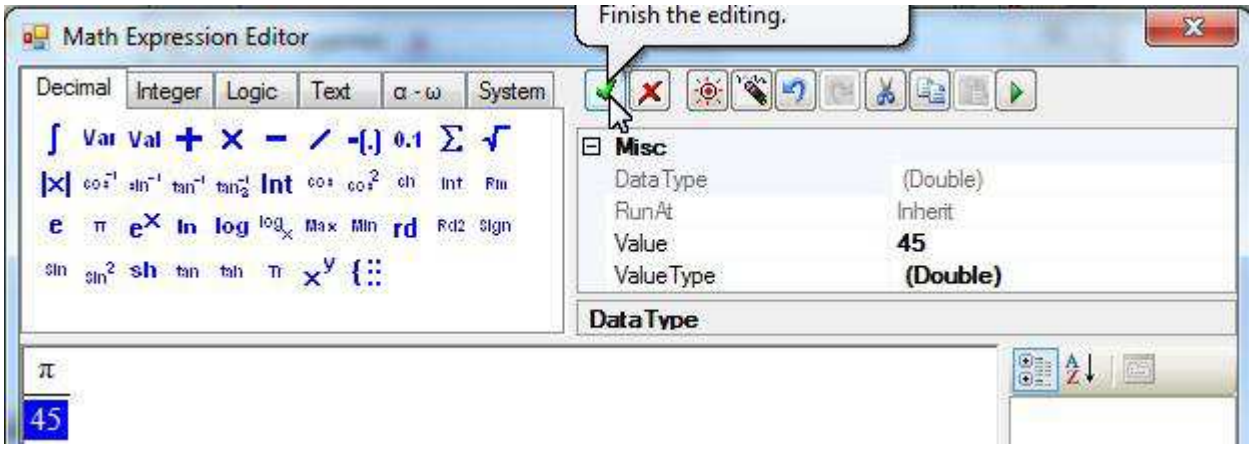


Provide action parameter values according to the Emgu tutorial. For thetaResolution, use an expression:

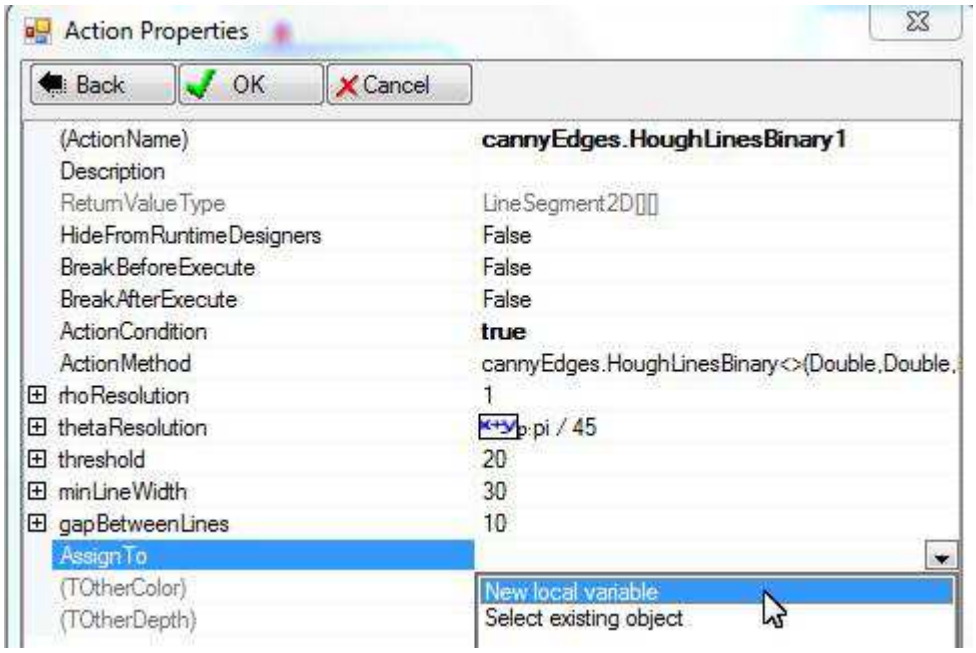


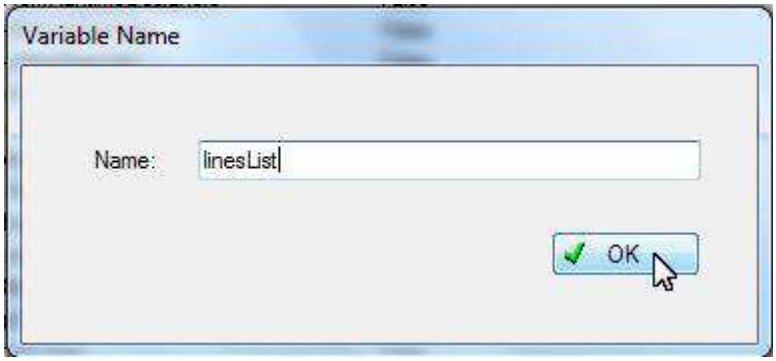




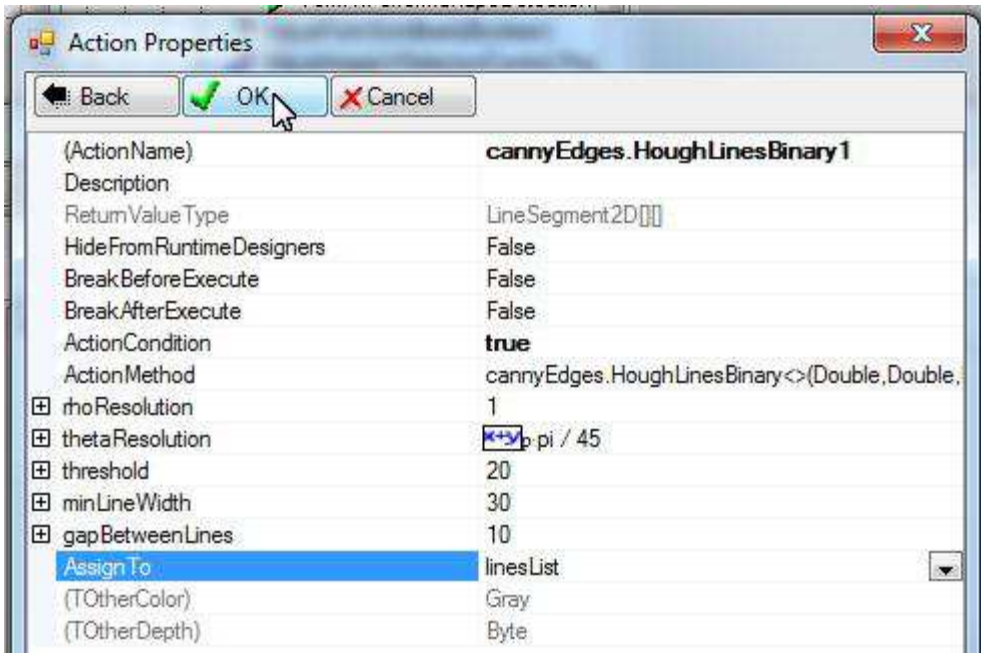


Use a new variable to get the results:

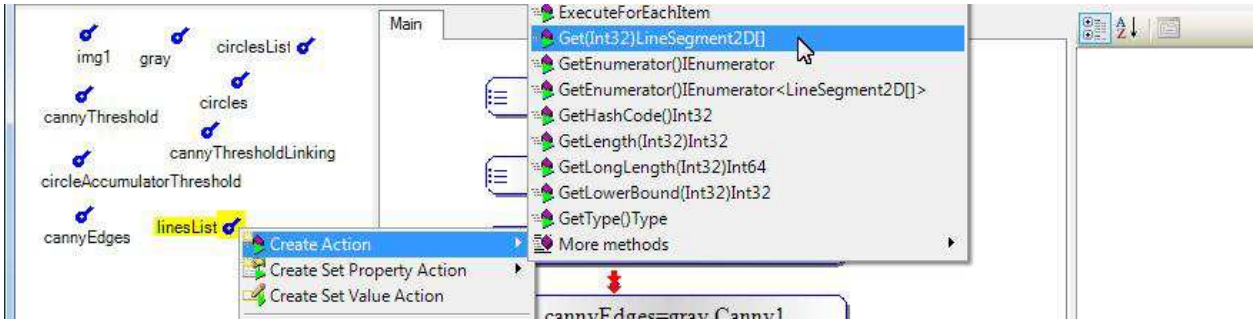


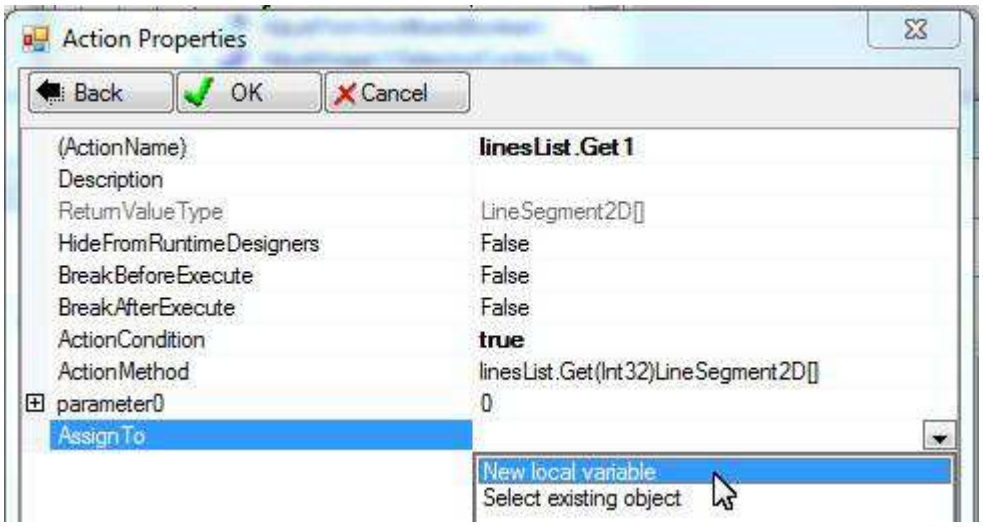


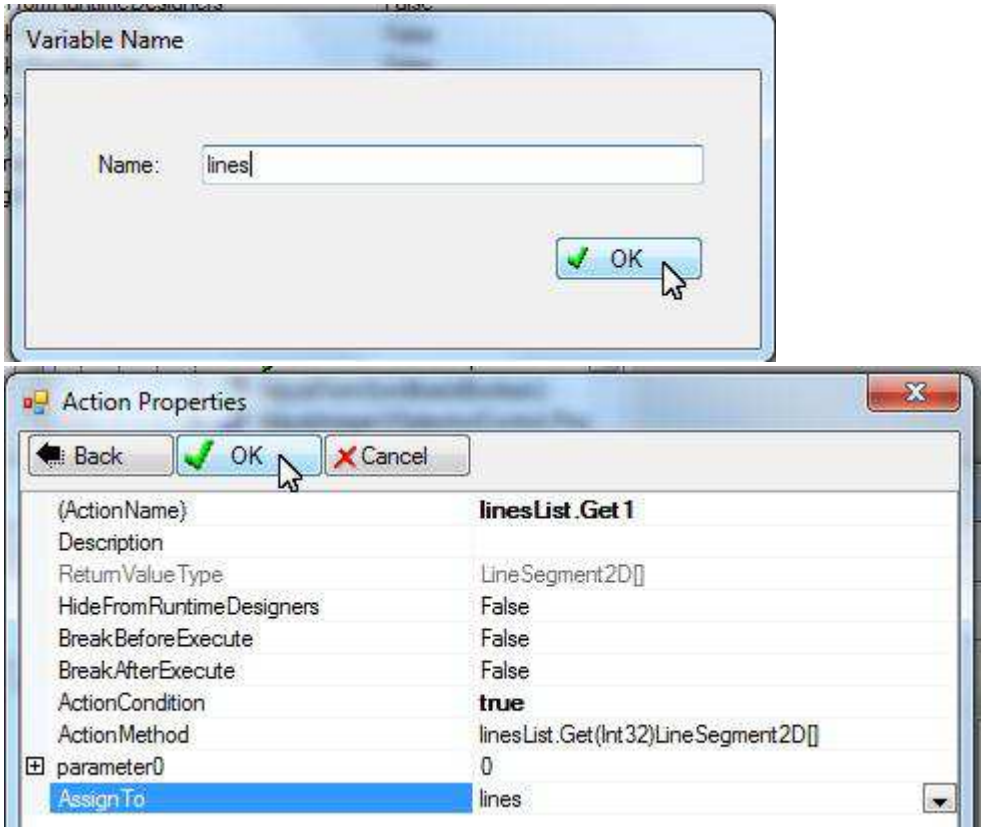
Click OK to finish creating the action:



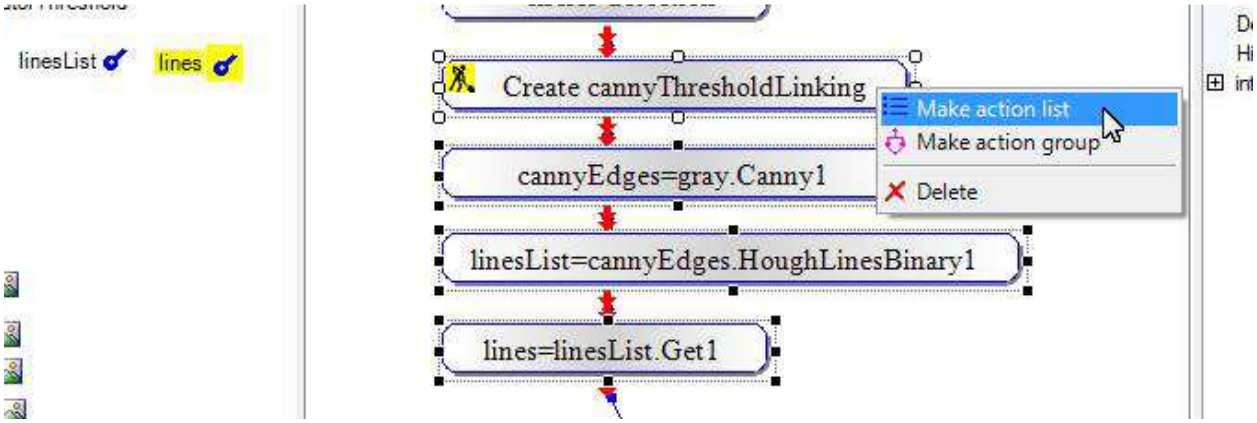
Get lines from the first item of “linesList”:

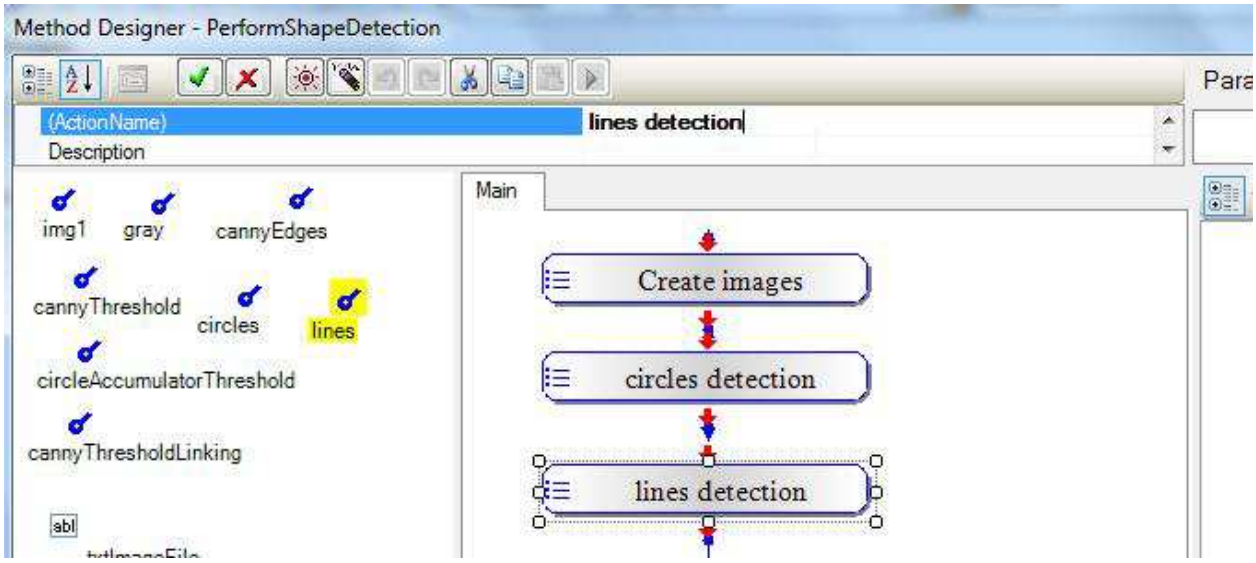






We may group the above 4 action into a “lines detection” action list to make room for the editor:



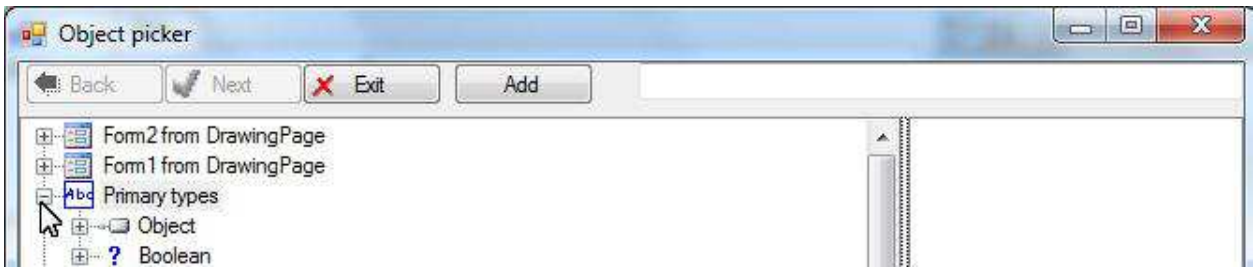


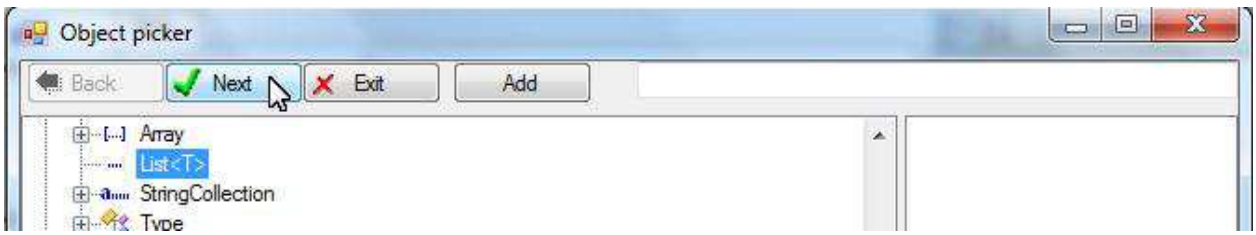
Triangles and Rectangles Detection

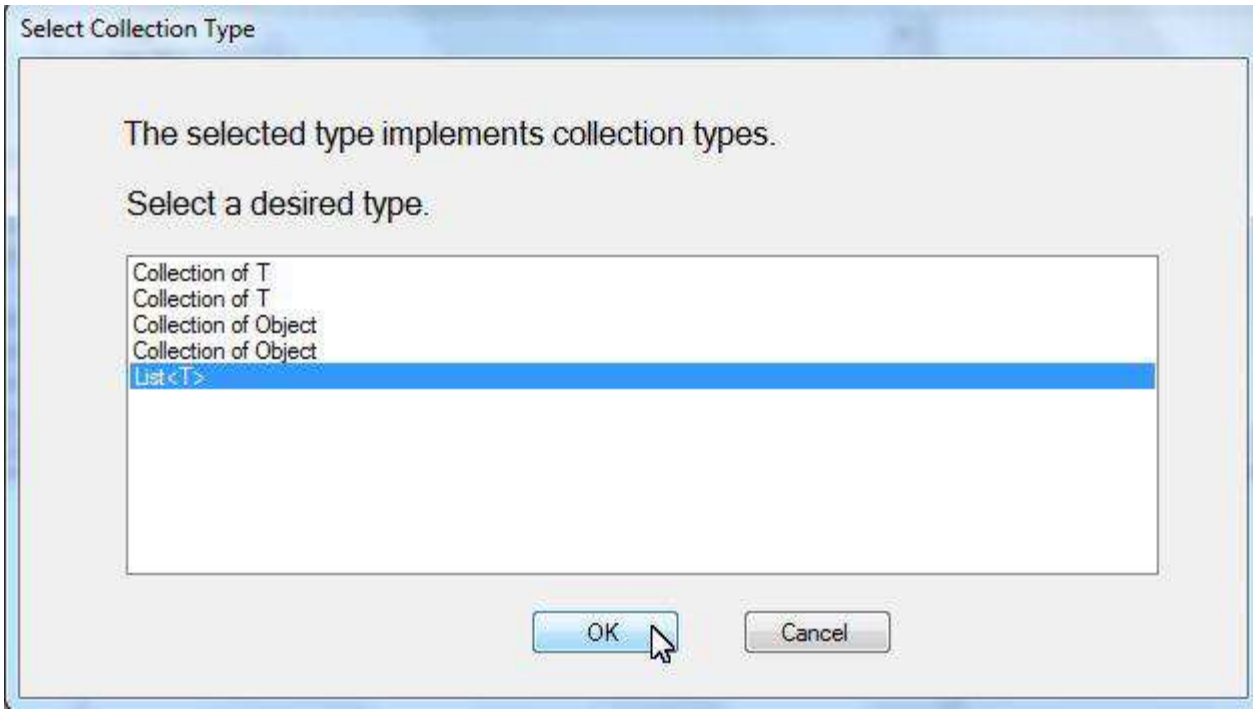
The "cannyEdges" variable has a `FindContours` method for finding contours. A triangle is a contour with 3 vertices and a rectangle is a contour with 4 vertices.

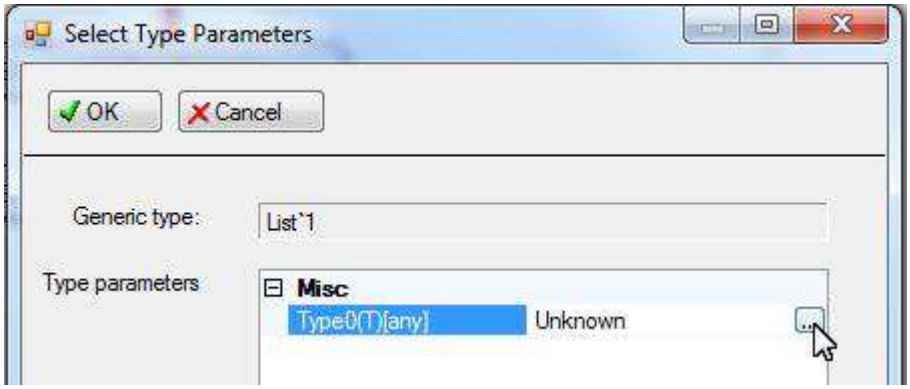
Create variable triangle list

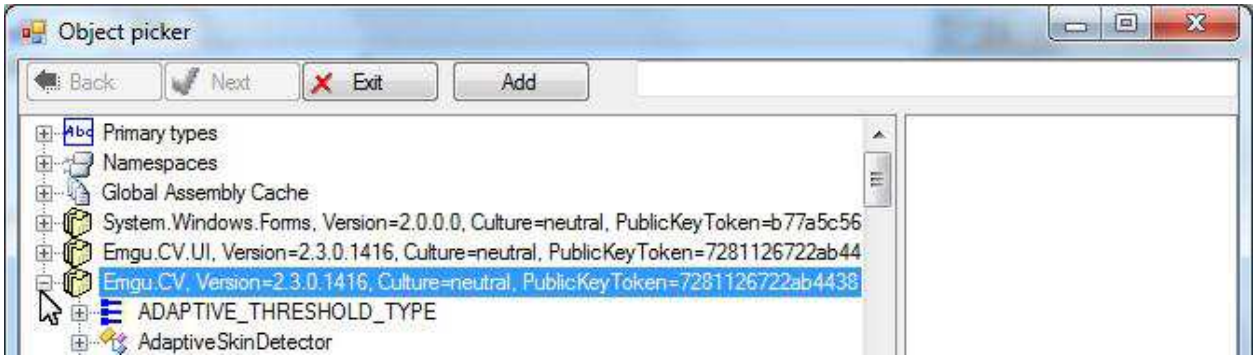
We use a variable for triangles.

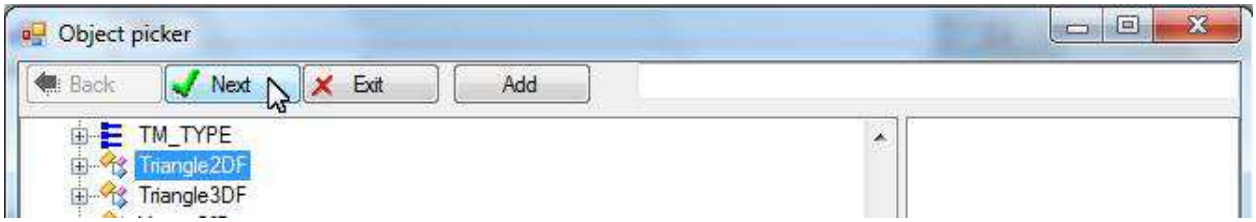


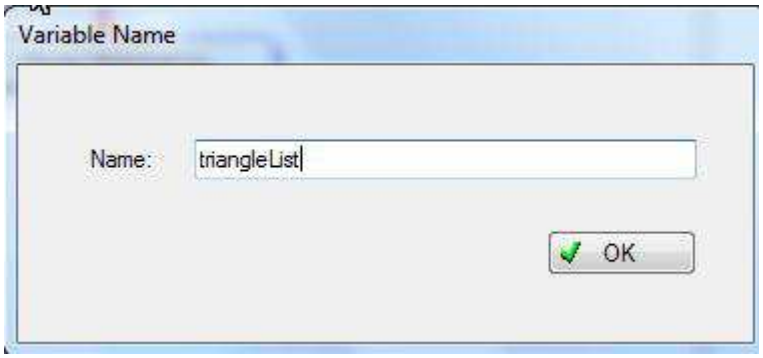



Select Collection Type
The selected type implements collection types.
Select a desired type.
Collection of T
Collection of T
Collection of Object
Collection of Object
List<T>
OK
Cancel


Select Type Parameters
OK
Cancel
Generic type:
List`1
Type parameters
Misc
Type0(T)[any]
Unknown


Object picker
Back
Next
Exit
Add
Primary types
Namespaces
Global Assembly Cache
System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c56
Emgu.CV.UI, Version=2.3.0.1416, Culture=neutral, PublicKeyToken=7281126722ab44
Emgu.CV, Version=2.3.0.1416, Culture=neutral, PublicKeyToken=7281126722ab4438
ADAPTIVE_THRESHOLD_TYPE
AdaptiveSkinDetector



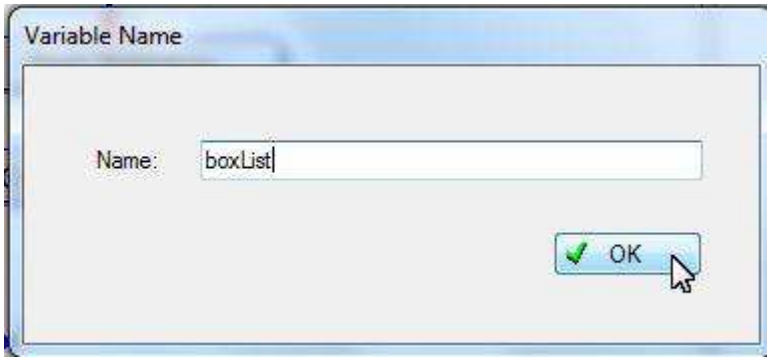


Create variable for rectangle list

The steps are mostly the same as above. The different steps are listed below.

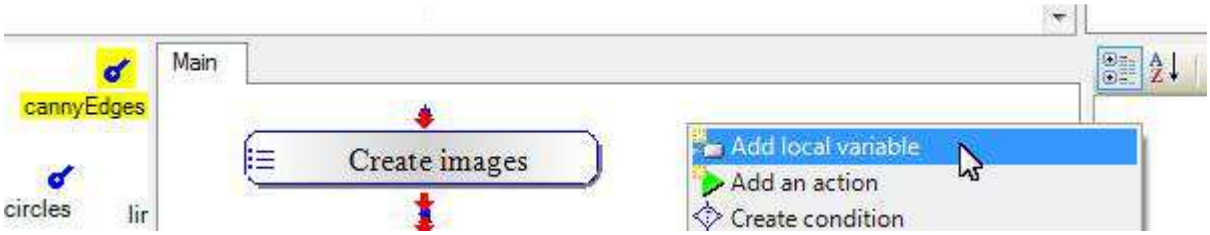
Select McvBox2D instead of Triangle2DF:

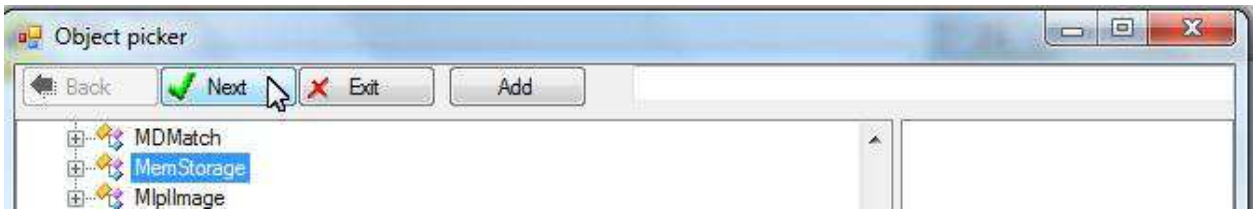
Name it “boxList”:

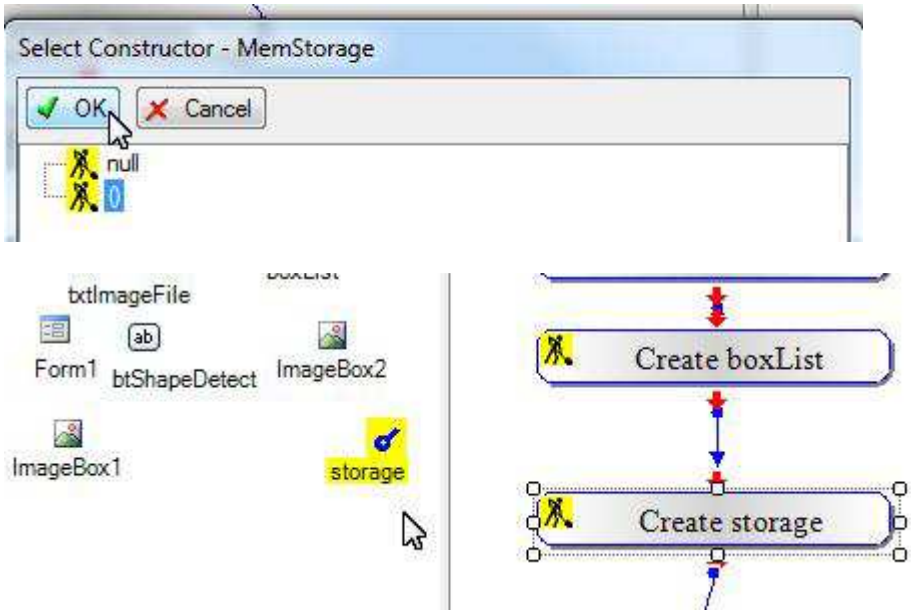


Create storage variable

A storage variable is needed for various actions.

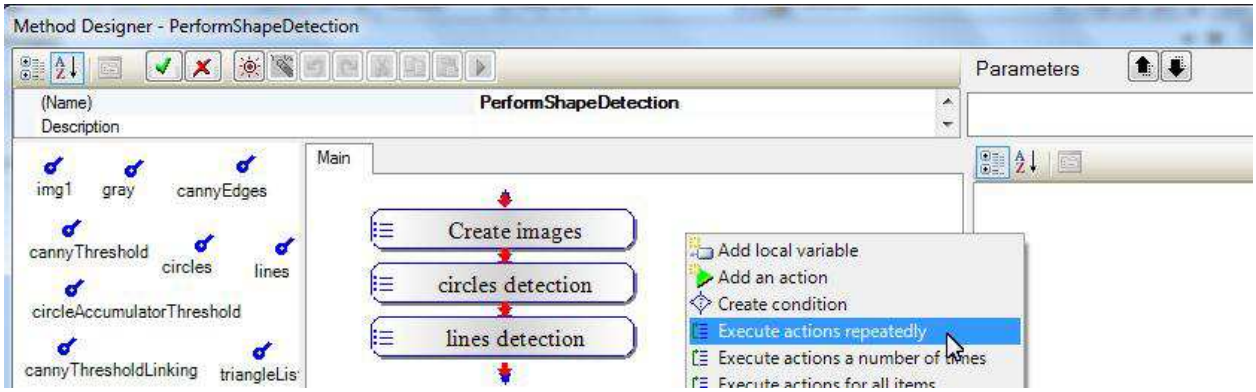




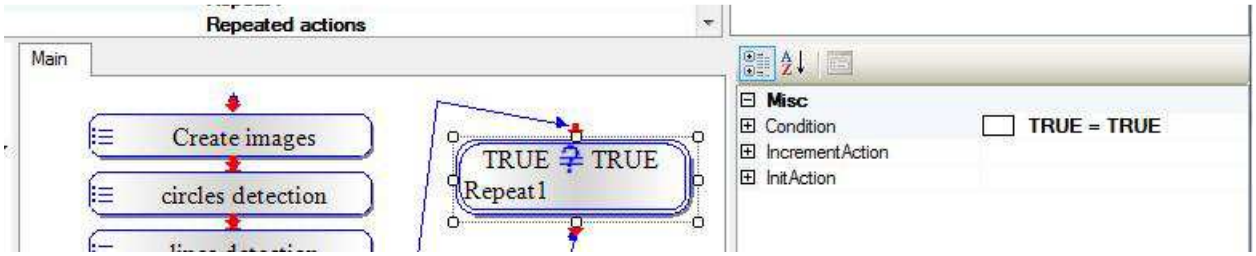


Go through shapes

We use a loop action to go through contours.

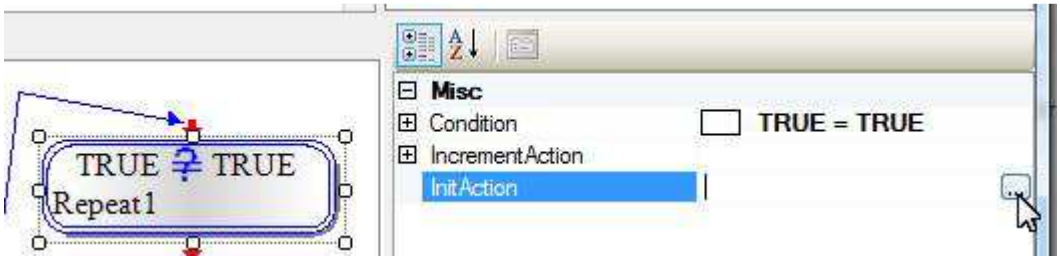


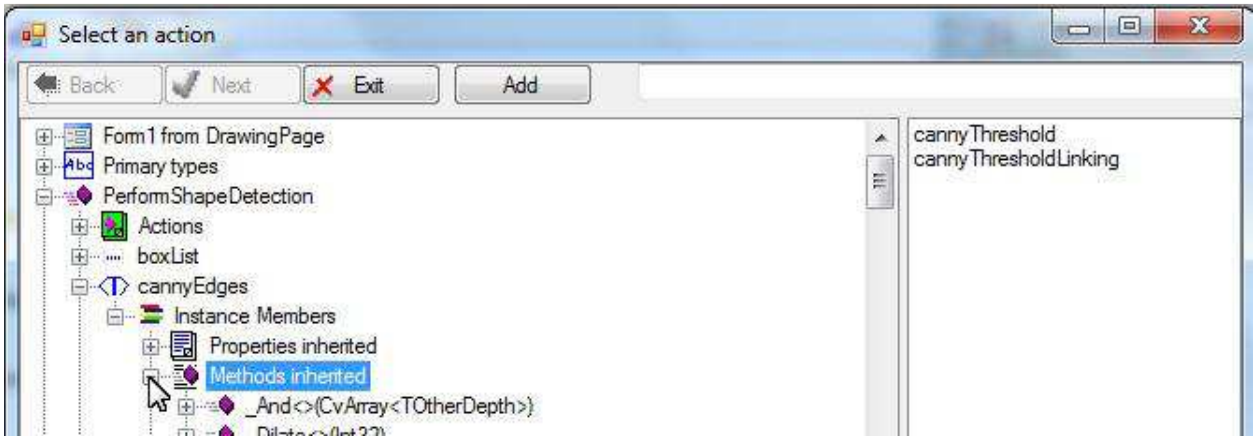
A loop action has an InitAction property indicating the initialization action executed before the loop starts. It has an IncrementAction property indicating the action to be executed after each loop. It has a Condition property indicating the condition to break the loop. If the Condition evaluates to False then the loop breaks.

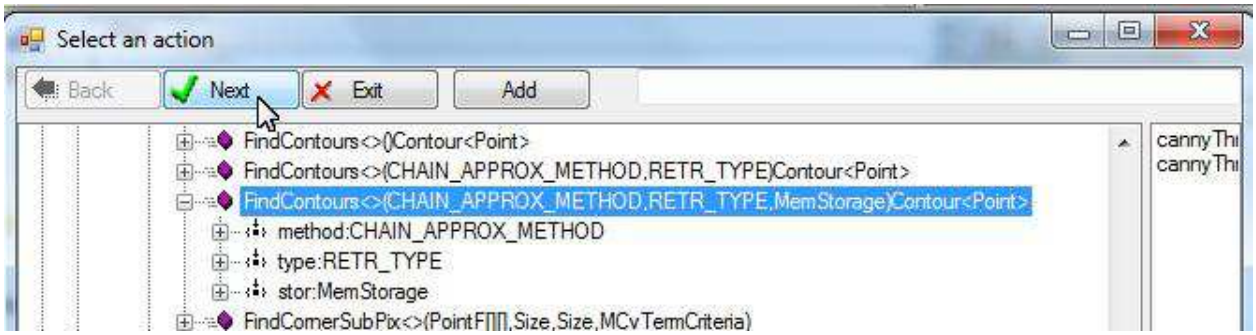


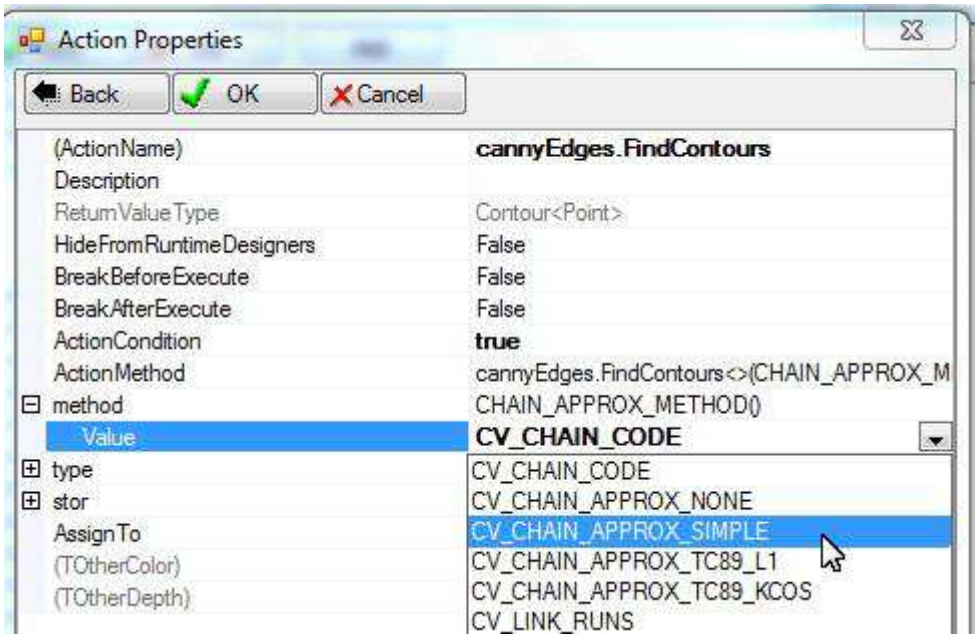
Setup initialization action

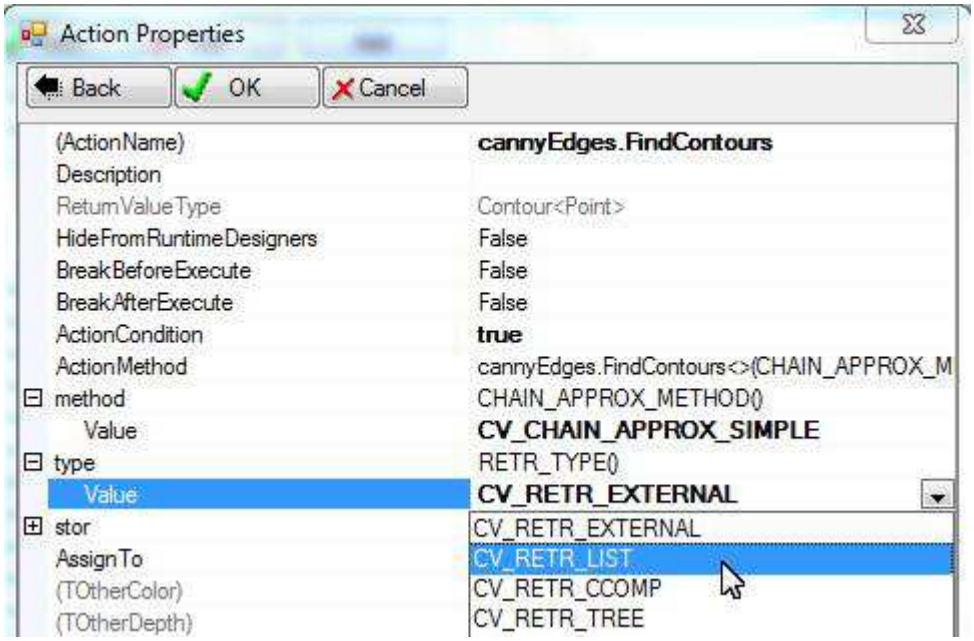
For this sample, the initialization action is to use the FindContours method of “cannyEdges” to get all shapes so that the loop action may go through each shape. This method returns the first shape. From the first shape we can get the next shape.

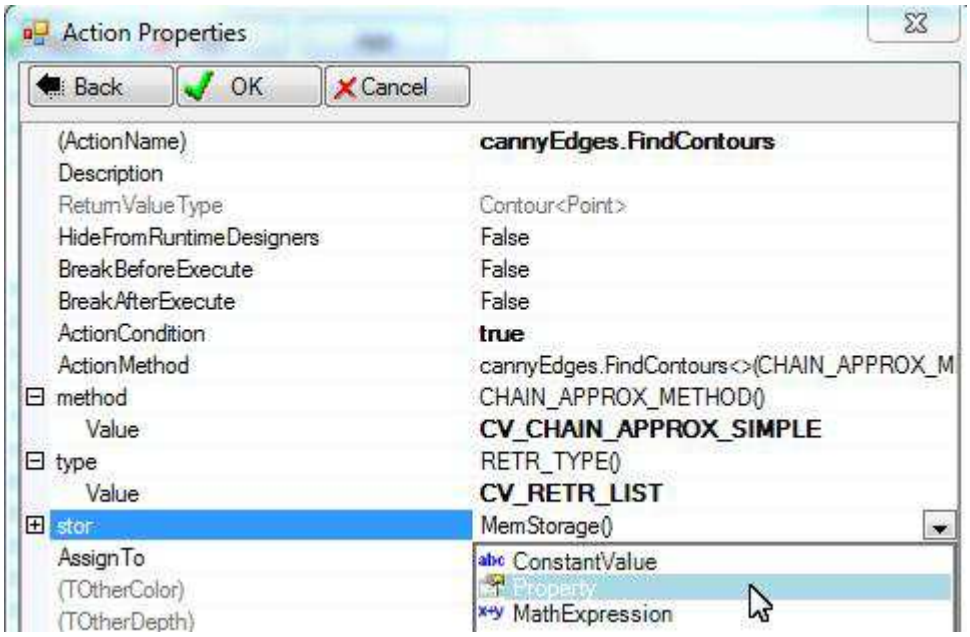


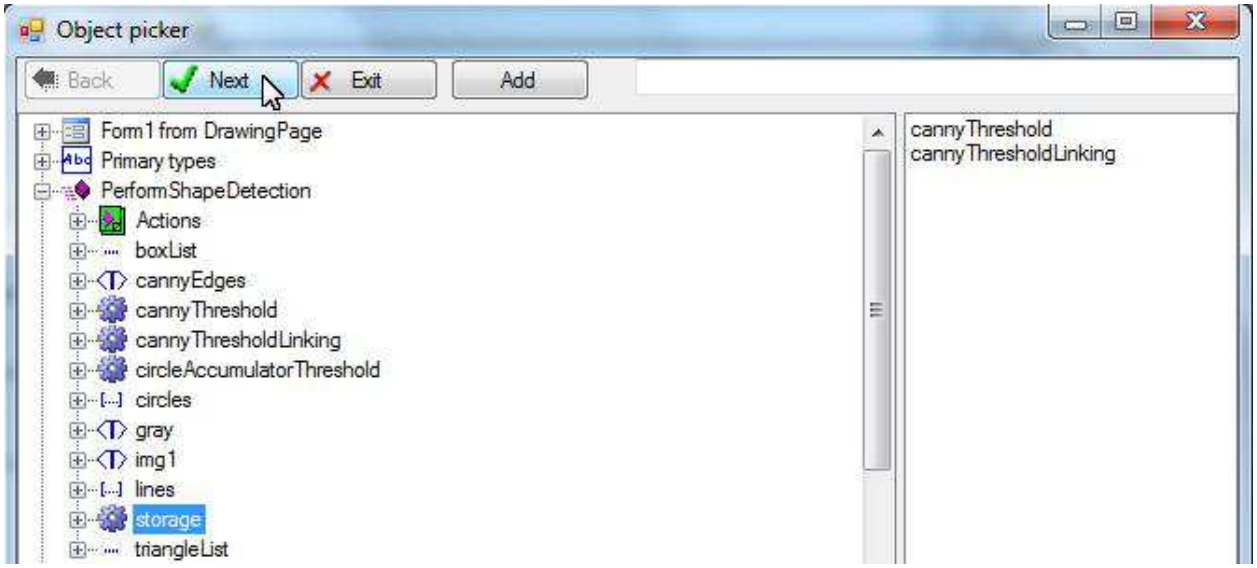


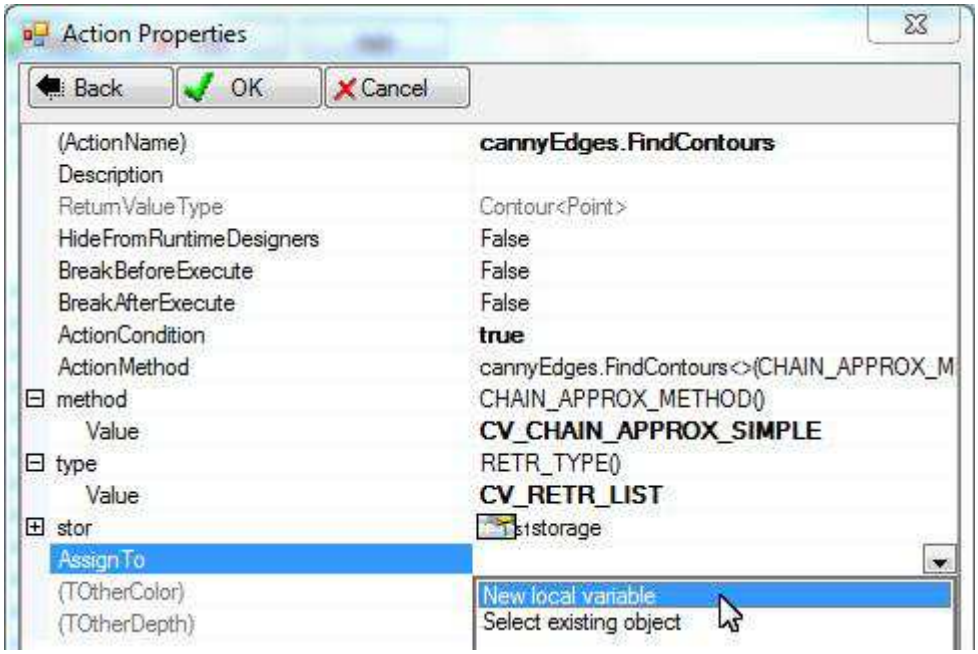


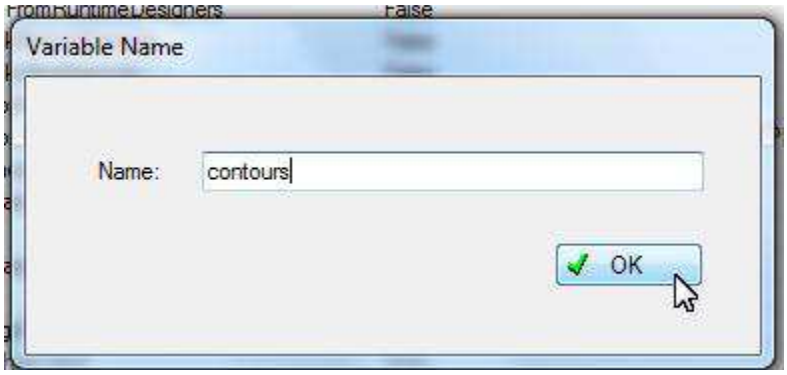

Action Properties
Back
OK
Cancel
(ActionName)
cannyEdges.FindContours
Description
ReturnValueType
Contour<Point>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
cannyEdges.FindContours<>(CHAIN_APPROX_M
method
CHAIN_APPROX_METHOD()
Value
CV_CHAIN_CODE
type
stor
AssignTo
(TOtherColor)
(TOtherDepth)
CV_CHAIN_CODE
CV_CHAIN_APPROX_NONE
CV_CHAIN_APPROX_SIMPLE
CV_CHAIN_APPROX_TC89_L1
CV_CHAIN_APPROX_TC89_KCOS
CV_LINK_RUNS

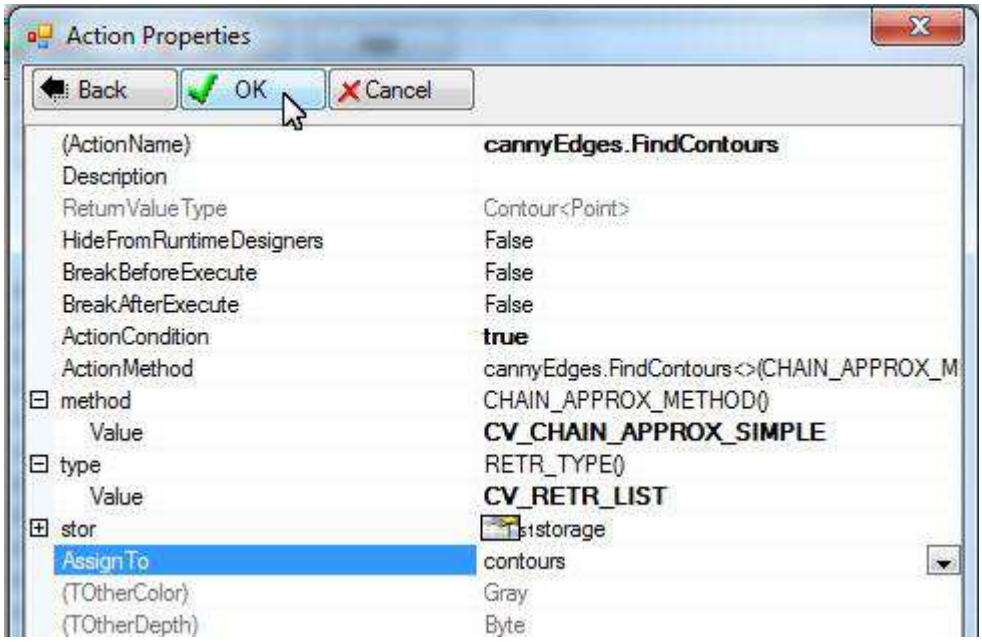

Action Properties
Back
OK
Cancel
(ActionName)
cannyEdges.FindContours
Description
ReturnValueType
Contour<Point>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
cannyEdges.FindContours<>(CHAIN_APPROX_M
method
CHAIN_APPROX_METHOD()
Value
CV_CHAIN_APPROX_SIMPLE
type
RETR_TYPE()
Value
CV_RETR_EXTERNAL
stor
AssignTo
(TOtherColor)
(TOtherDepth)
CV_RETR_EXTERNAL
CV_RETR_LIST
CV_RETR_CCOMP
CV_RETR_TREE


Action Properties
Back
OK
Cancel
(ActionName)
cannyEdges.FindContours
Description
ReturnValueType
Contour<Point>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
cannyEdges.FindContours<>(CHAIN_APPROX_M
method
CHAIN_APPROX_METHOD()
Value
CV_CHAIN_APPROX_SIMPLE
type
RETR_TYPE()
Value
CV_RETR_LIST
stor
MemStorage()
AssignTo
(TOtherColor)
(TOtherDepth)
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
gray
img1
lines
storage
triangleList
cannyThreshold
cannyThresholdLinking

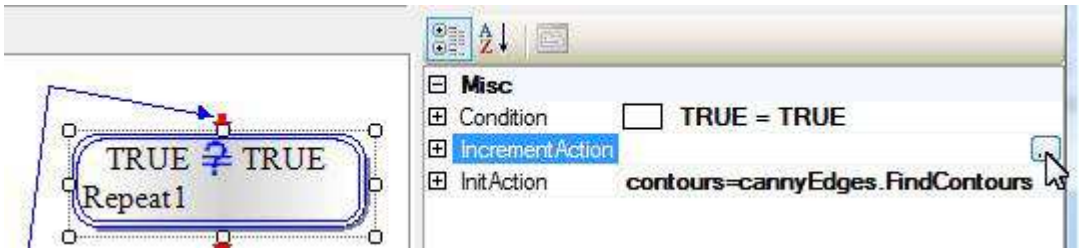

Action Properties
Back
OK
Cancel
(ActionName)
cannyEdges.FindContours
Description
ReturnValueType
Contour<Point>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
cannyEdges.FindContours<>(CHAIN_APPROX_M
method
CHAIN_APPROX_METHOD()
Value
CV_CHAIN_APPROX_SIMPLE
type
RETR_TYPE()
Value
CV_RETR_LIST
stor
s1storage
AssignTo
(TOtherColor)
(TOtherDepth)
New local variable
Select existing object

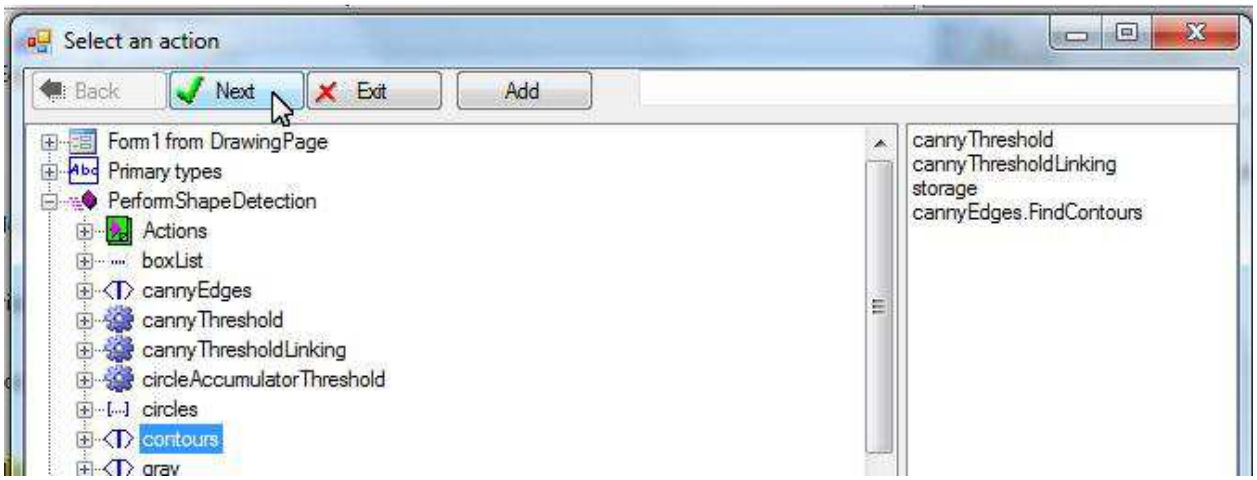

Variable Name
Name:
contours
OK

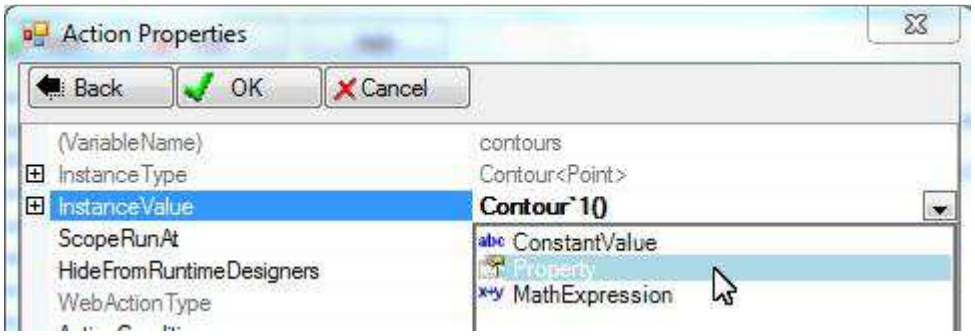

Action Properties
Back
OK
Cancel
(ActionName)
cannyEdges.FindContours
Description
ReturnValueType
Contour<Point>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
cannyEdges.FindContours<>(CHAIN_APPROX_M
method
CHAIN_APPROX_METHOD()
Value
CV_CHAIN_APPROX_SIMPLE
type
RETR_TYPE()
Value
CV_RETR_LIST
stor
s1storage
AssignTo
contours
(TOtherColor)
Gray
(TOtherDepth)
Byte

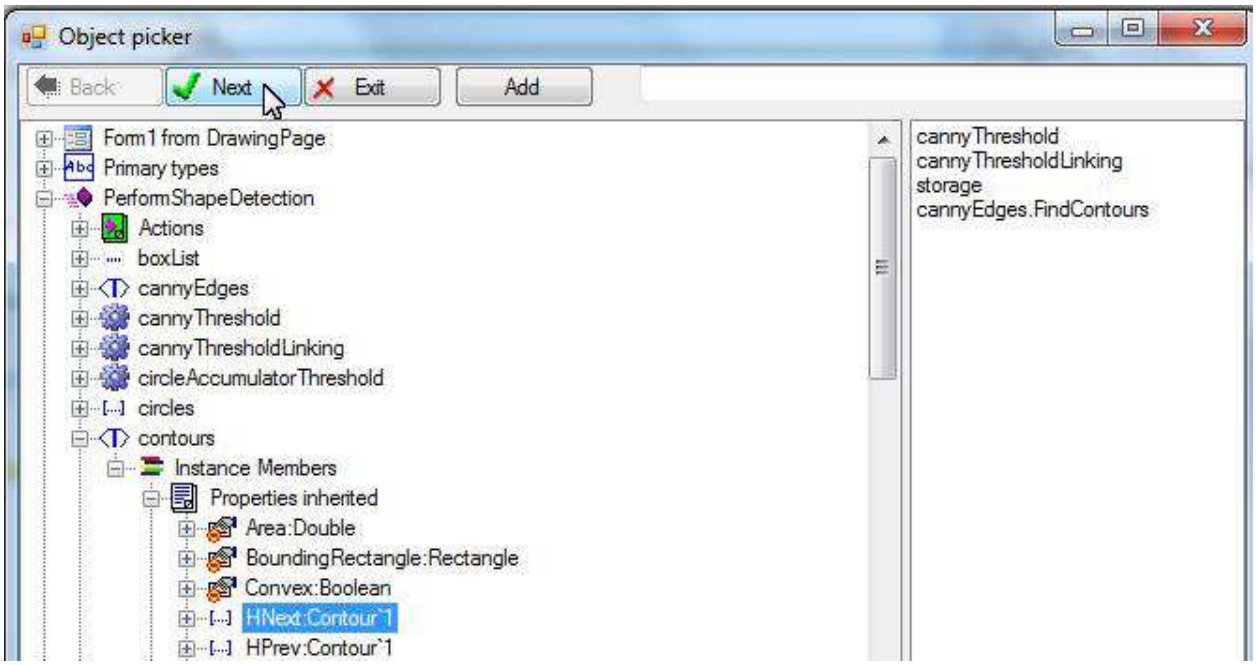
Setup increment action

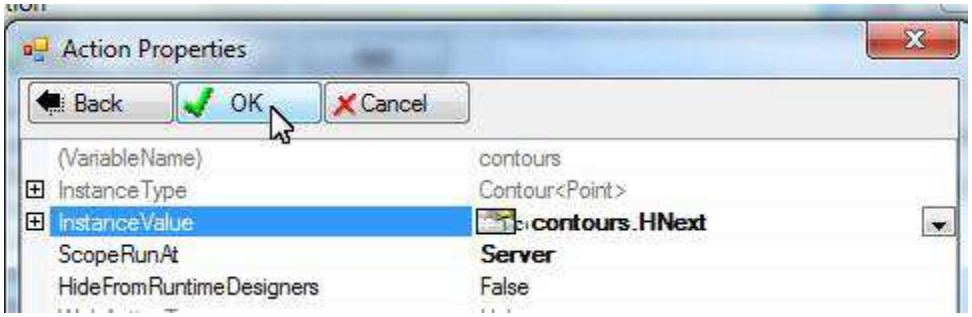
The “contours” has a HNext property for getting the next contours. We may use it to create an action to be executed after each loop.







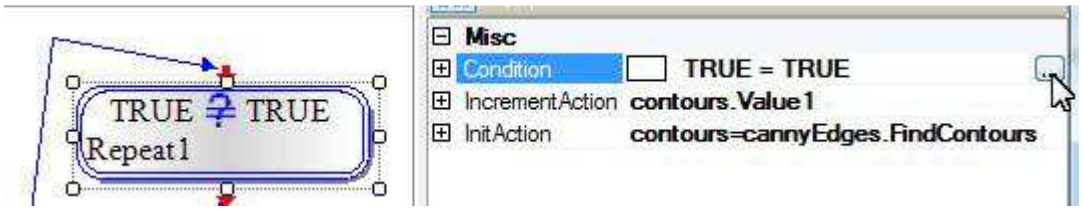


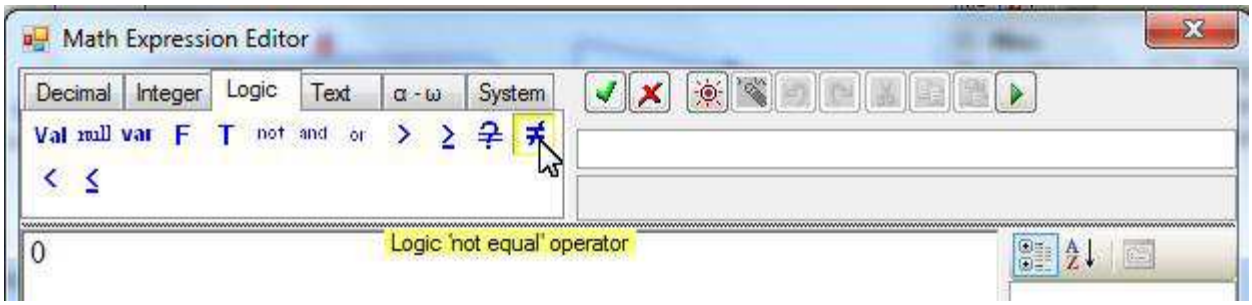


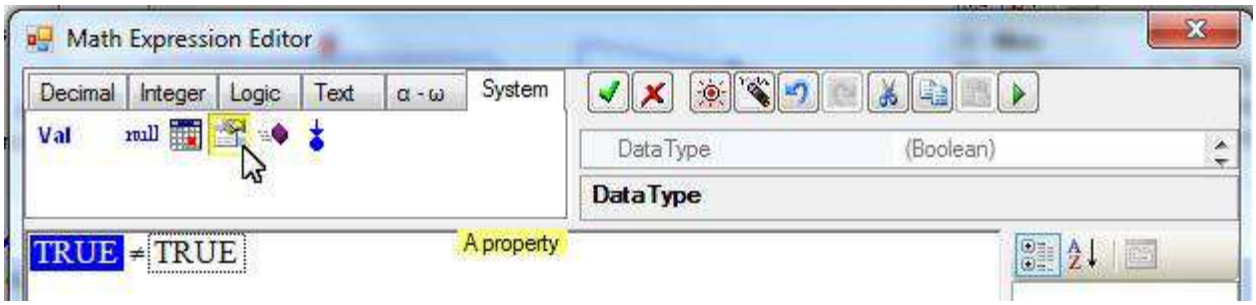
After executing this action, the “contours” variable will become the next shape.

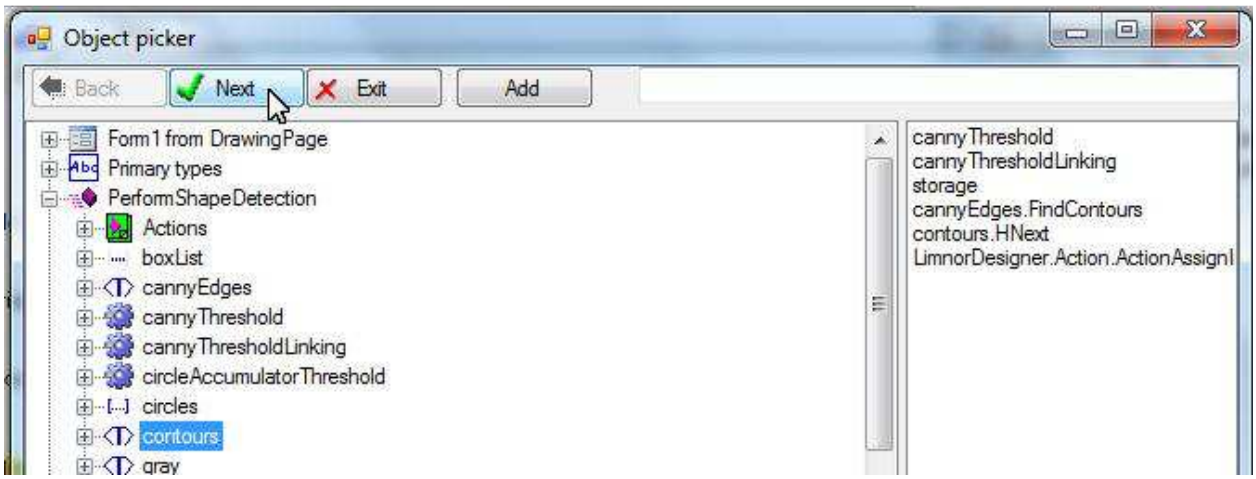
Setup loop condition

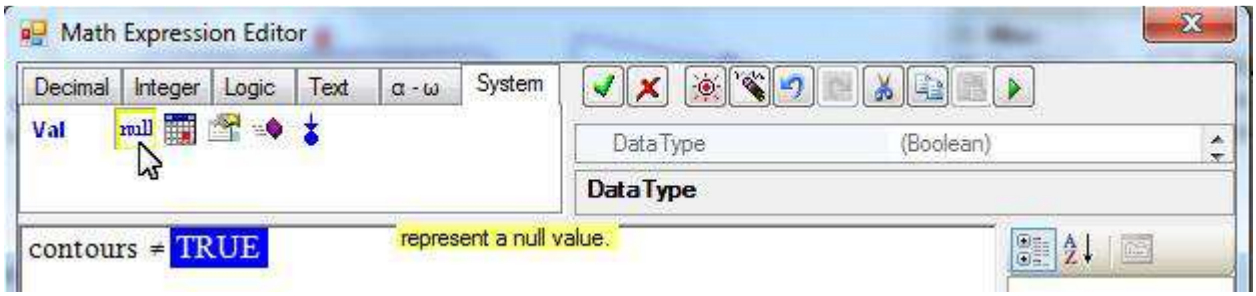
The “contours” variable represents the current shape. If there are no more shapes then “contours” variable will be null. So, the loop condition should be that the “contours” variable is not null.

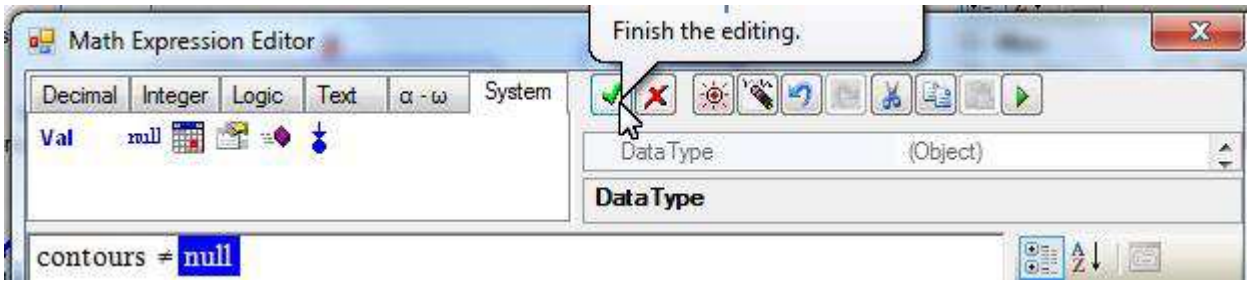



Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
0
Logic 'not equal' operator

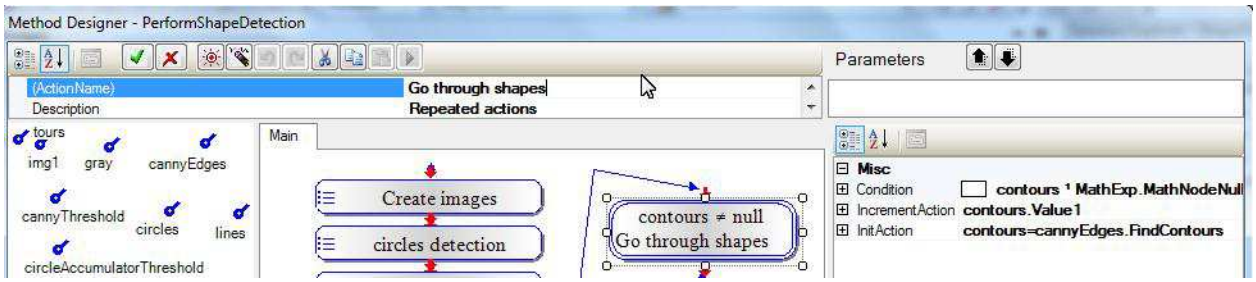

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
TRUE ≠ TRUE
A property


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
gray
cannyThreshold
cannyThresholdLinking
storage
cannyEdges.FindContours
contours.HNext
LimnorDesigner.Action.ActionAssignI


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
contours = TRUE
represent a null value.

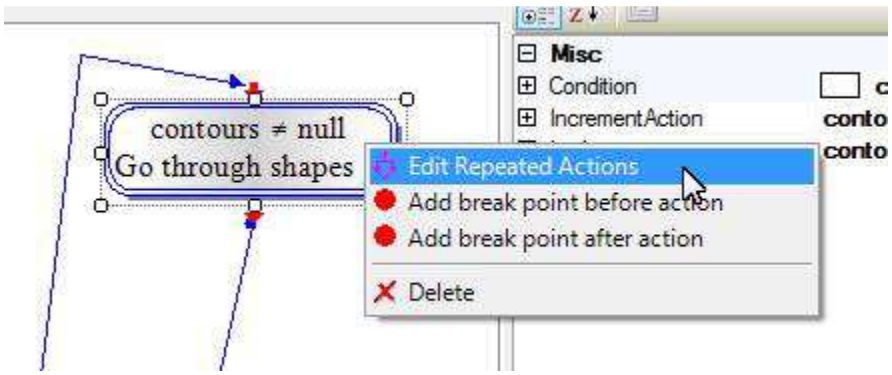


We may give a description to the loop action:

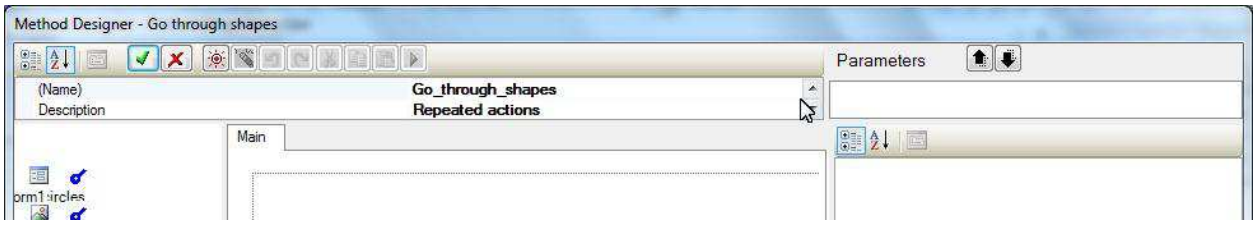


Process Each Shape

To process each shape, right-click the loop action, choose “Edit Repeated Actions”:

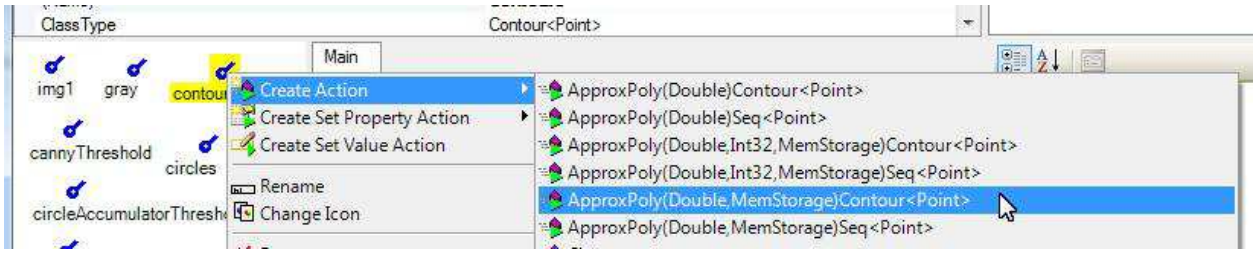


A new Method Editor Dialogue box appears for us to add actions to it:

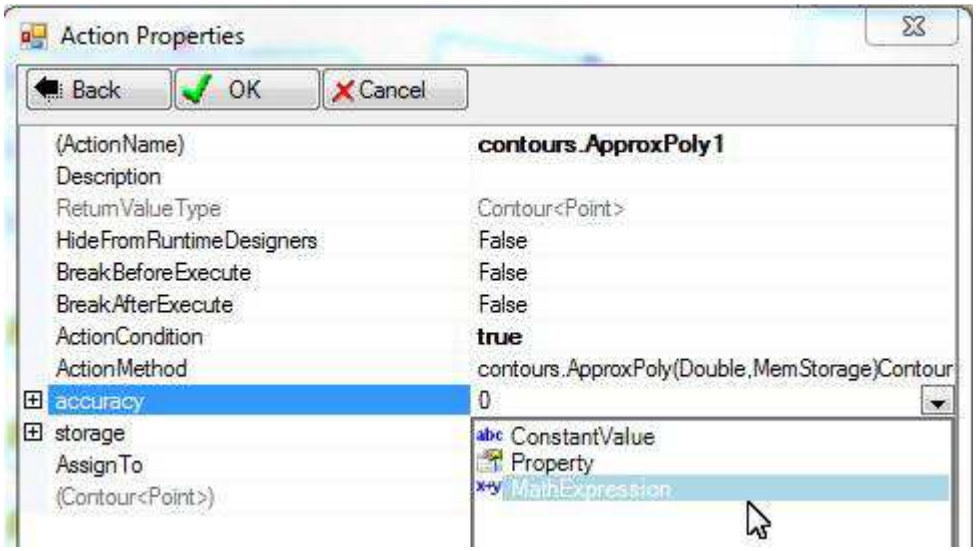


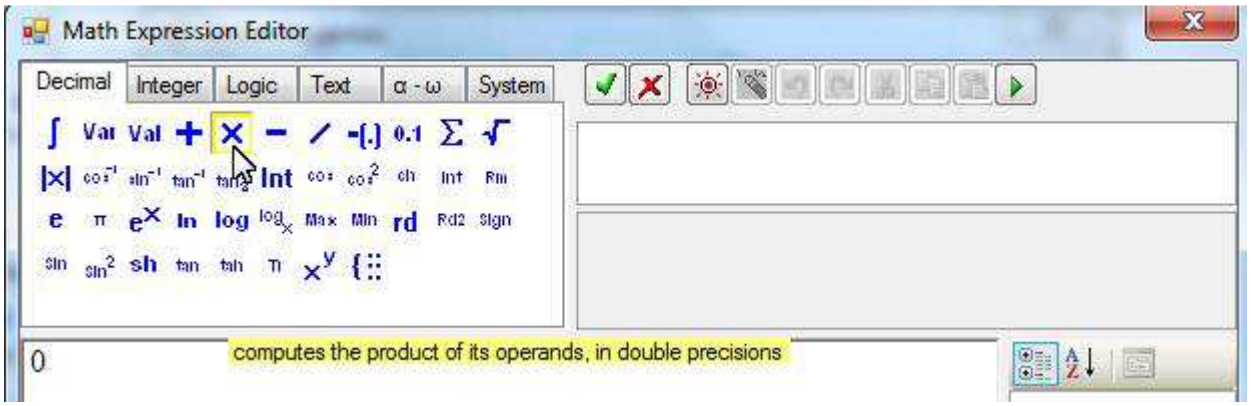
Get current contour

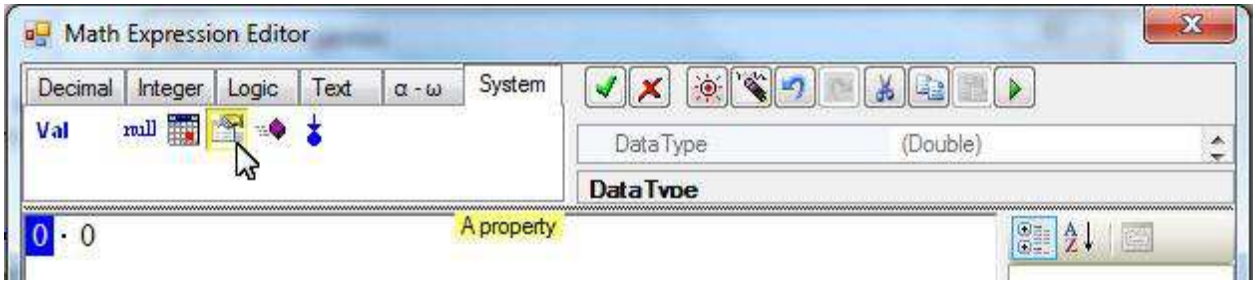
“contours” has a `ApproxPoly` method for getting the current contour.

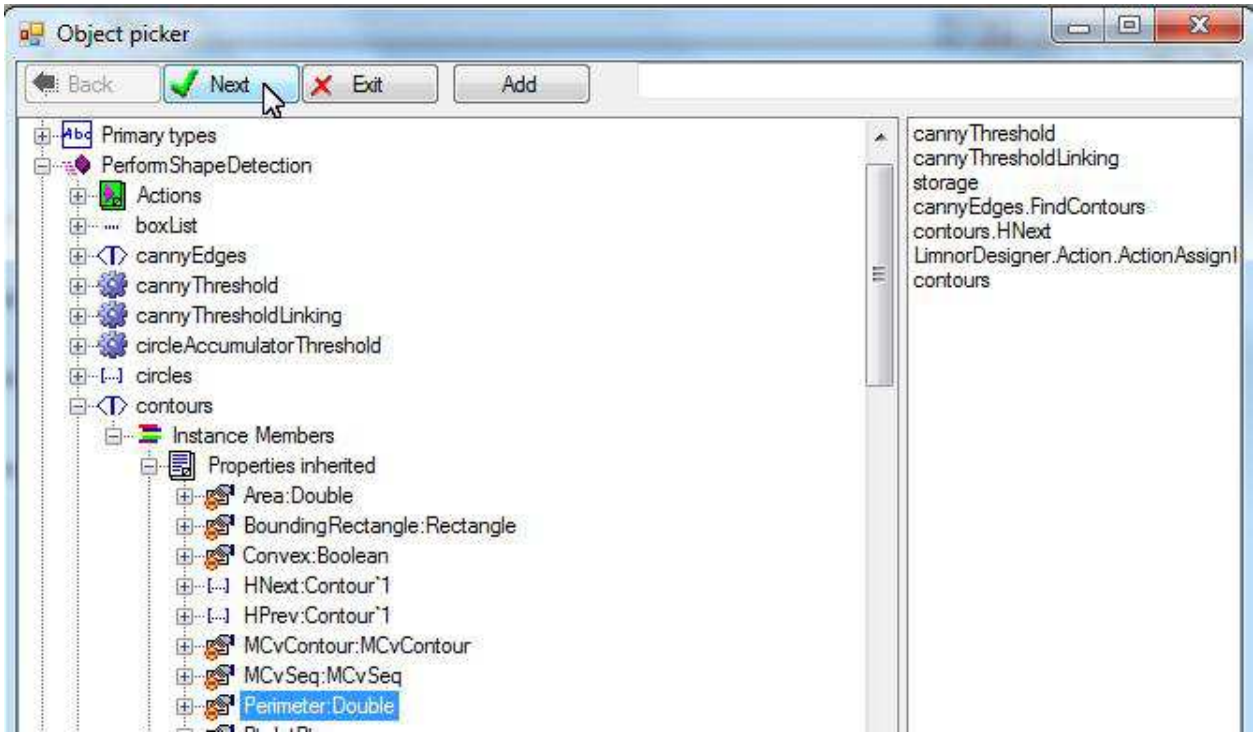


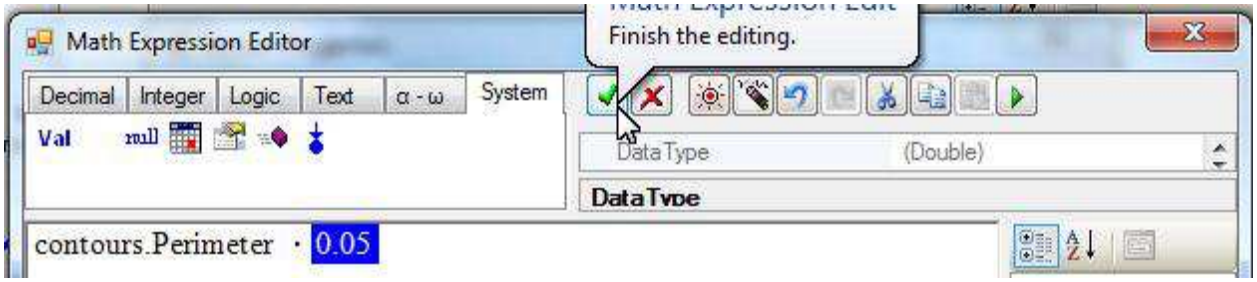
Set the “accuracy” to 5% of the contour perimeter:



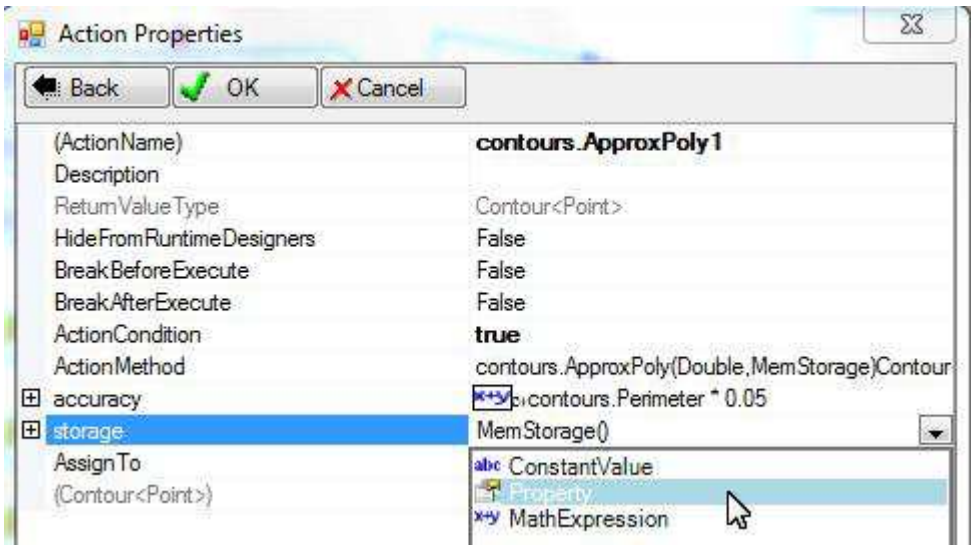


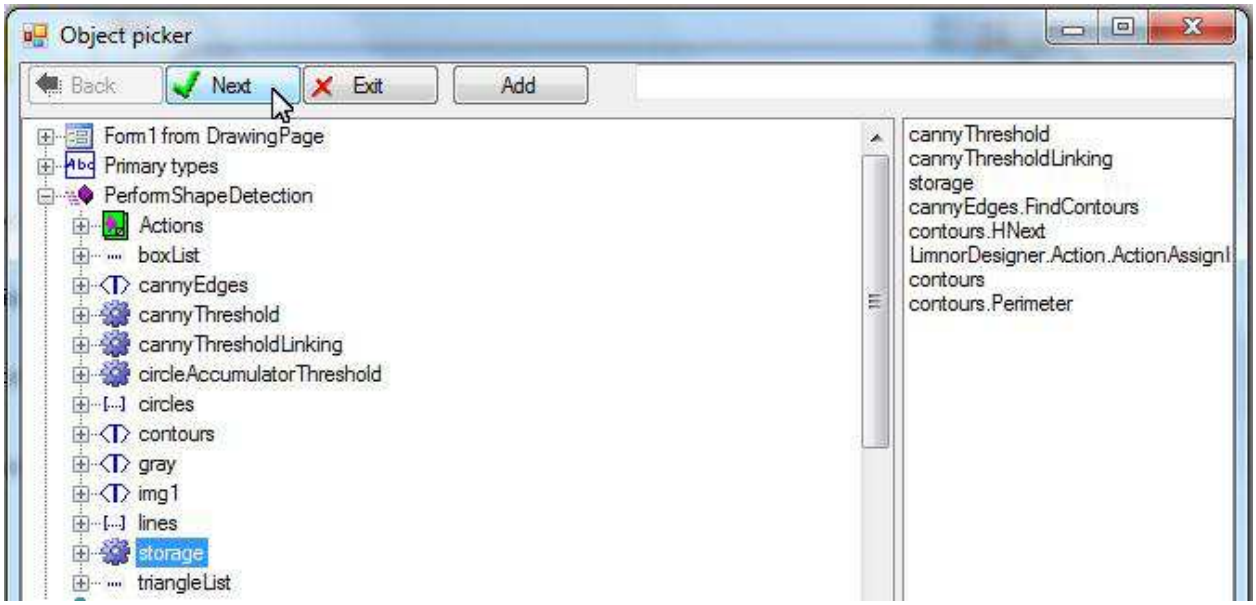




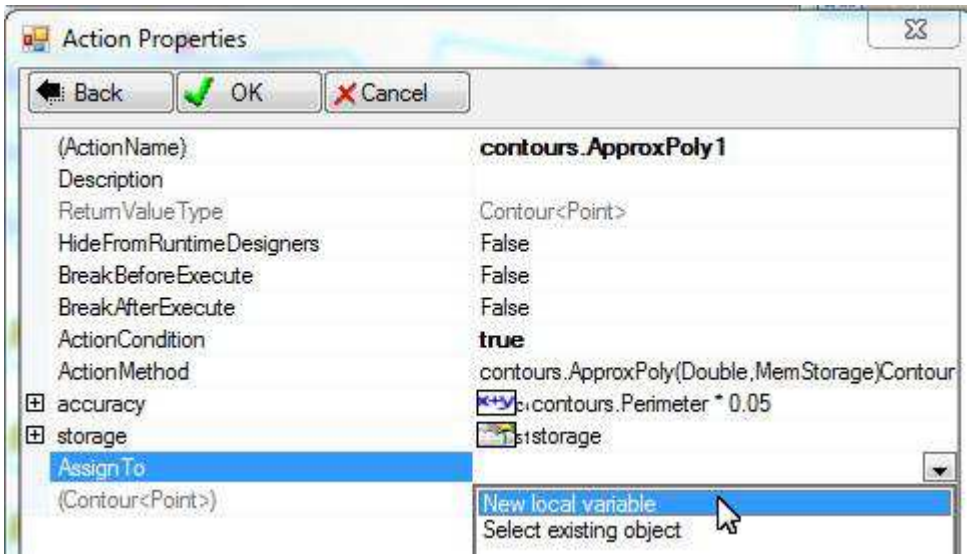


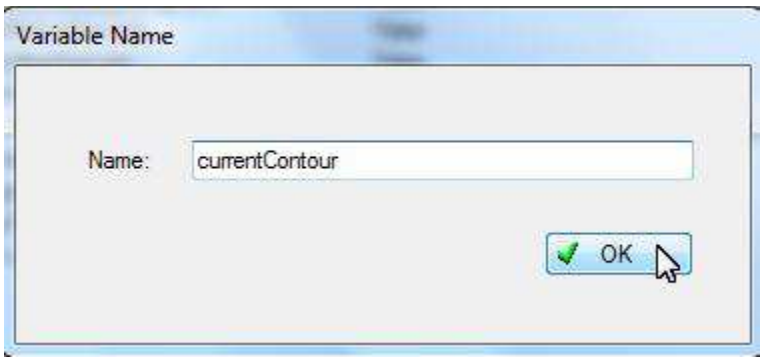
Set the storage to the variable “storage”:



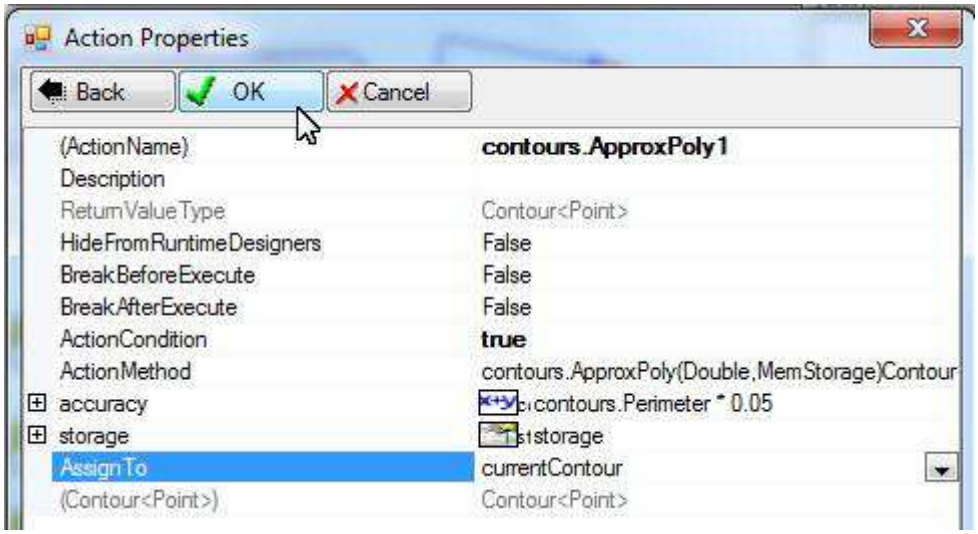


Assign the result to a new variable:

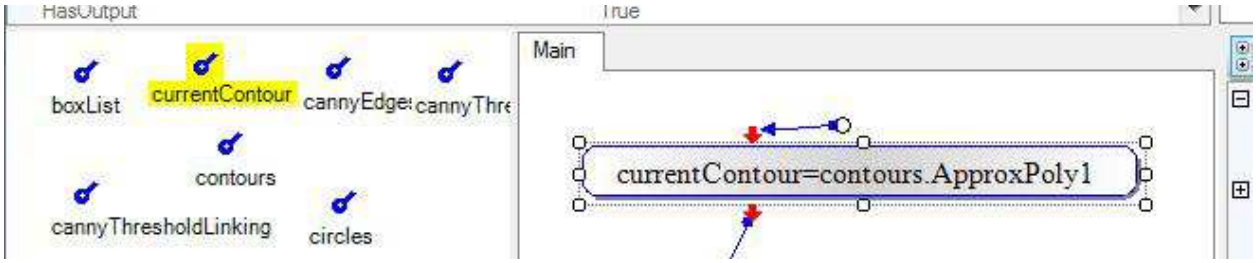




Click OK to finish creating the action.

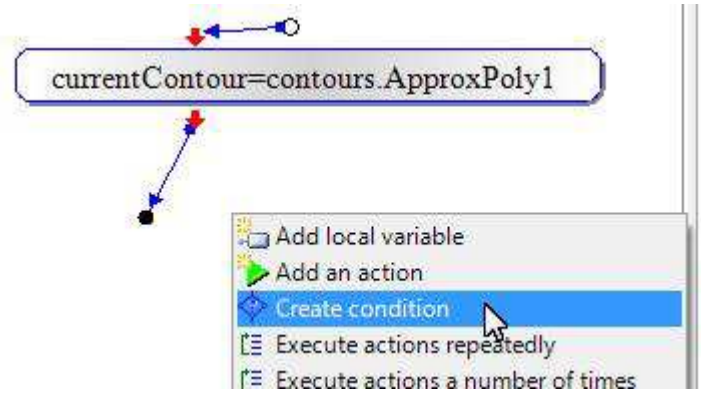


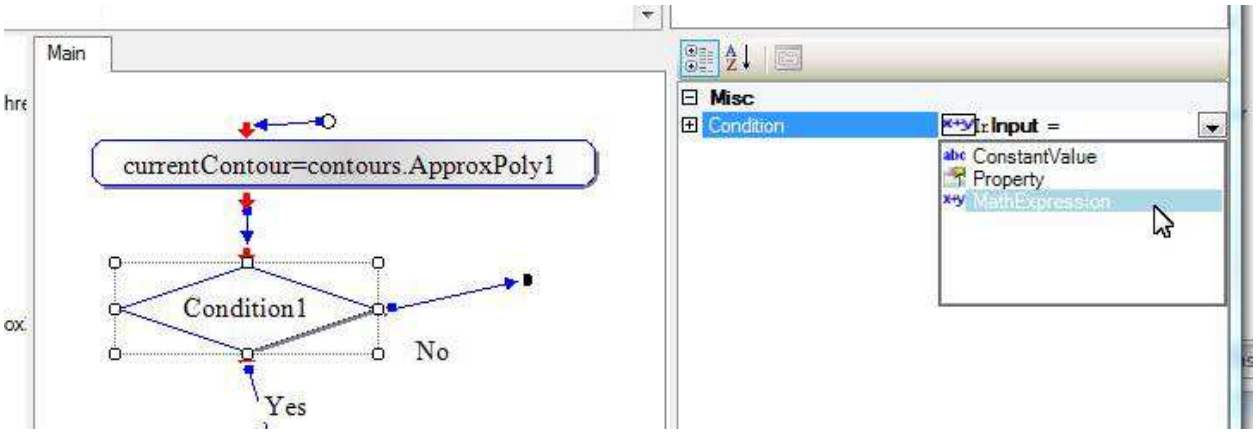
The new variable and action appear:

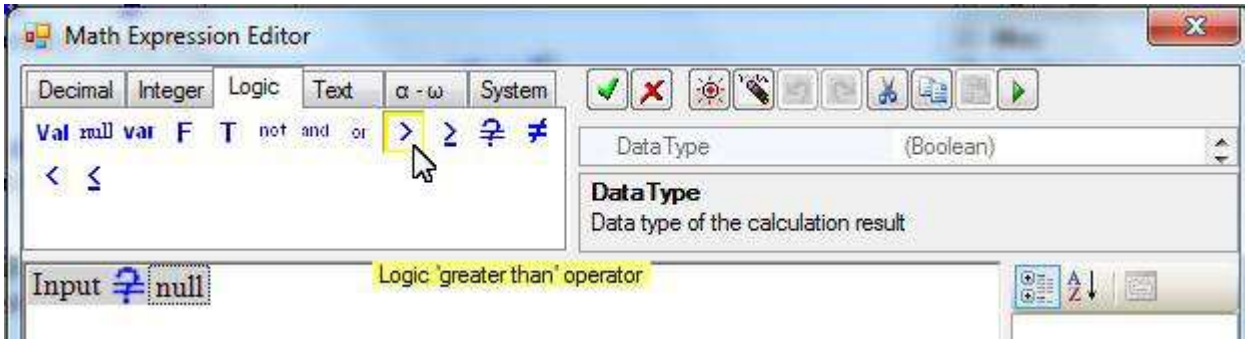


Check contour size

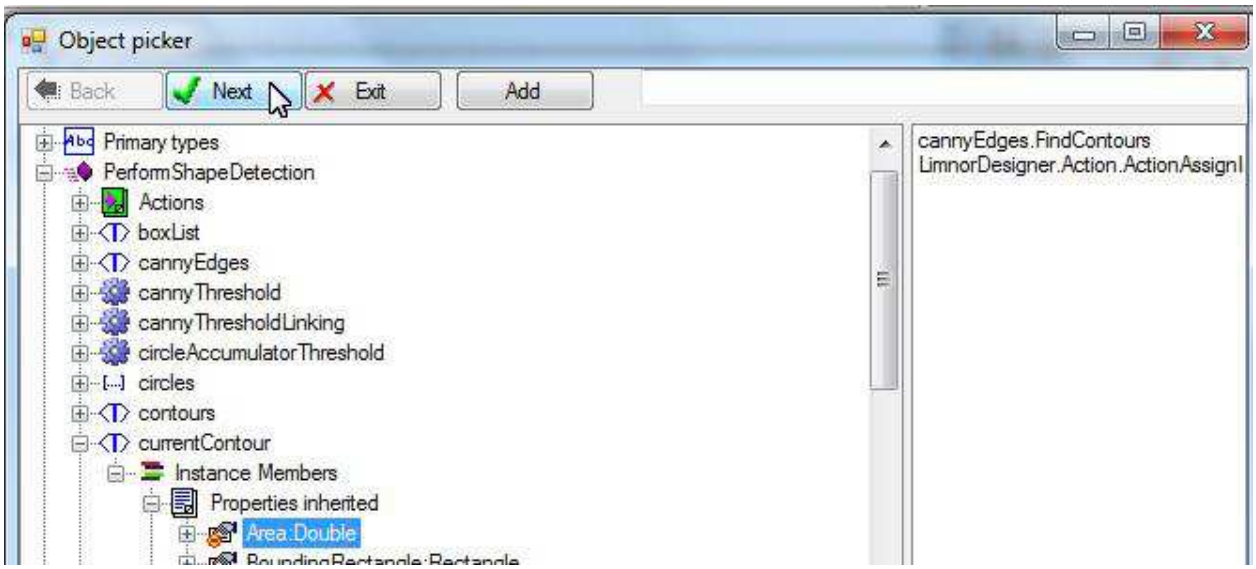
Only a contour's Area property is larger than 250 will be processed.

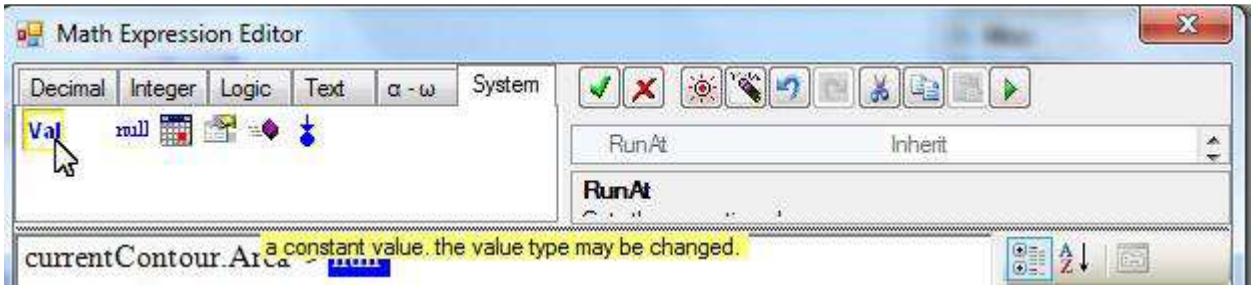


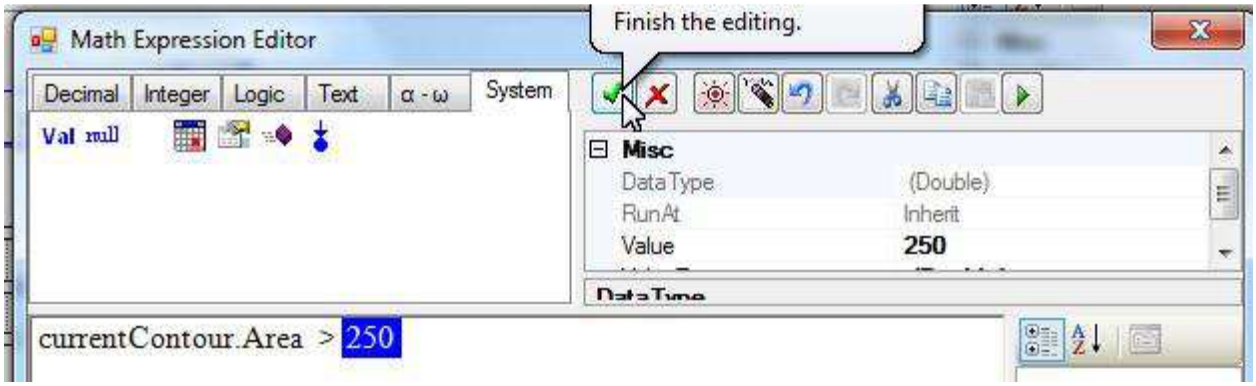

Main
currentContour=contours.ApproxPoly1
Condition1
No
Yes
Misc
Condition
Input =
ConstantValue
Property
MathExpression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val null var F T not and or
DataType
(Boolean)
DataType
Data type of the calculation result
Input null
Logic 'greater than' operator

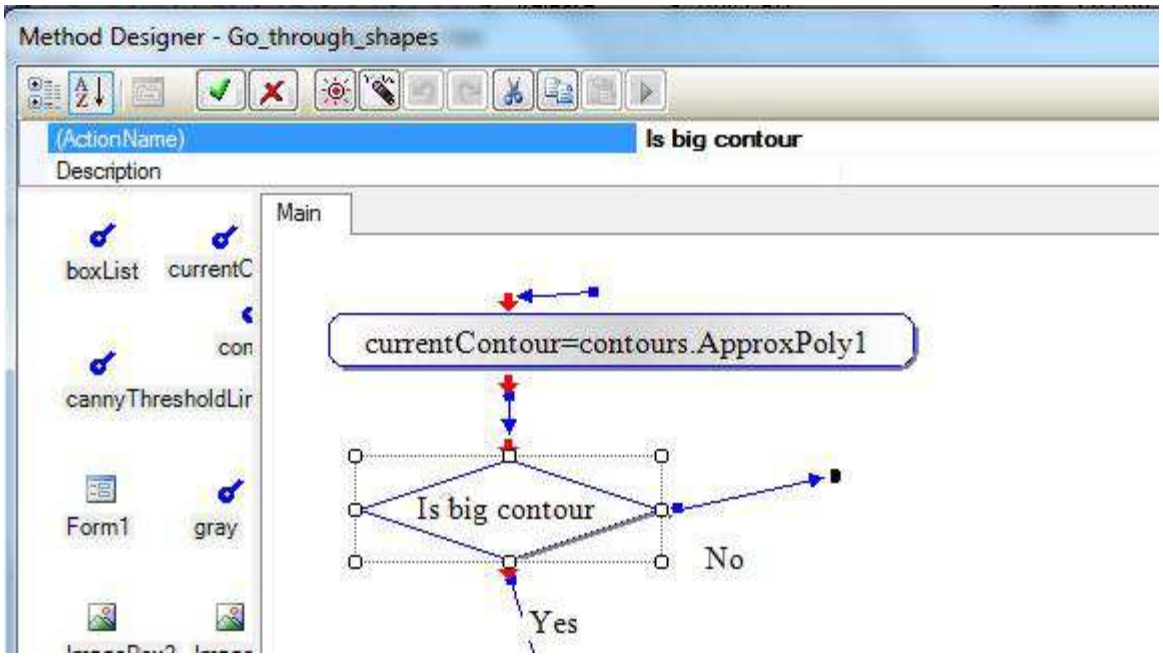

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val null
Misc
RunAt
Gets the execution place
Input > null
A property





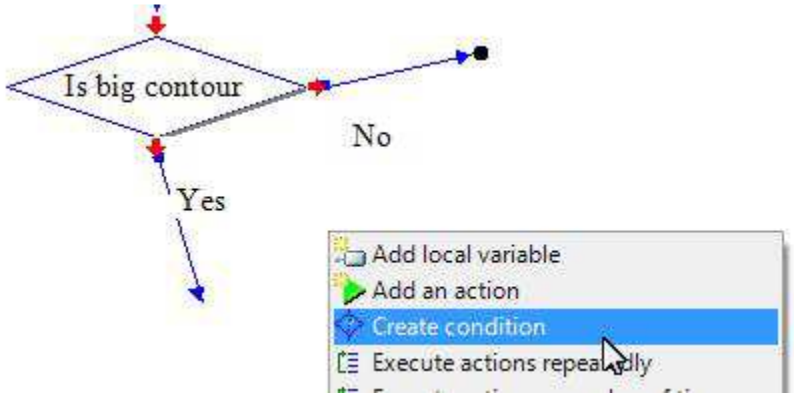


Rename the condition:

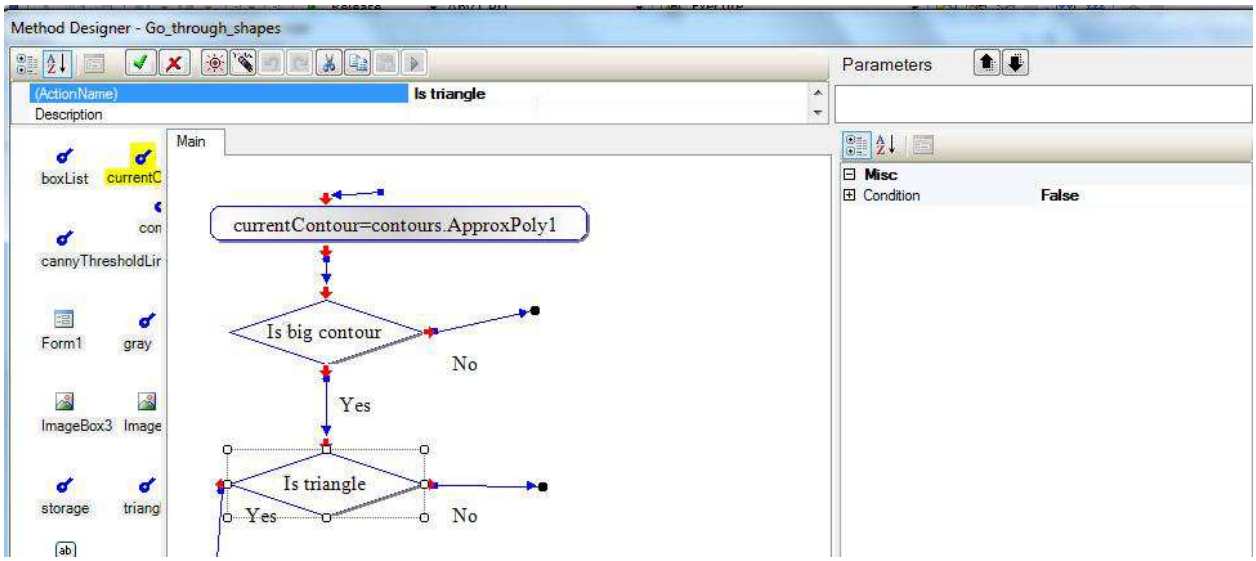


Get triangle

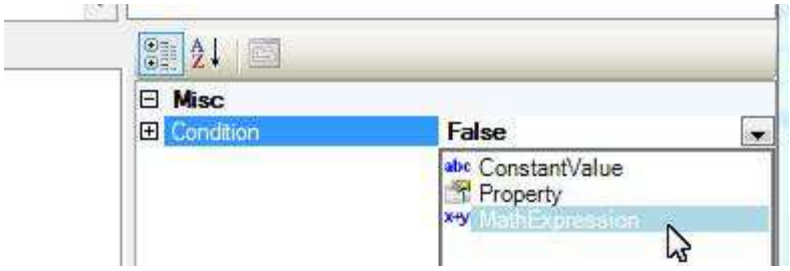
If a contour has 3 vertices then it is a triangle.

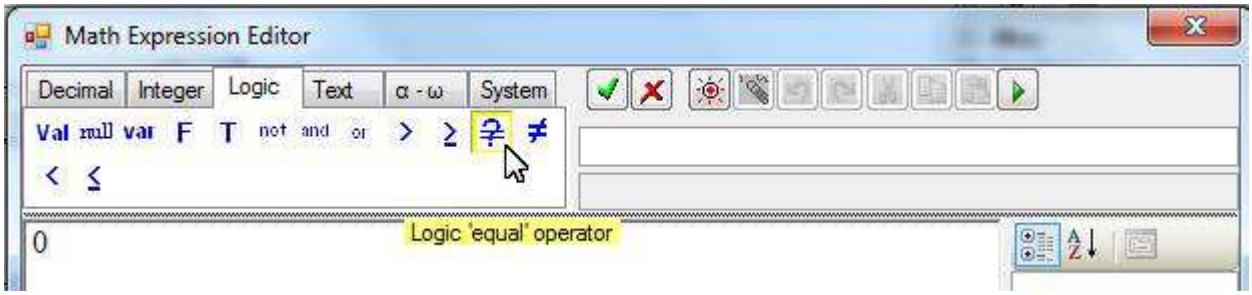


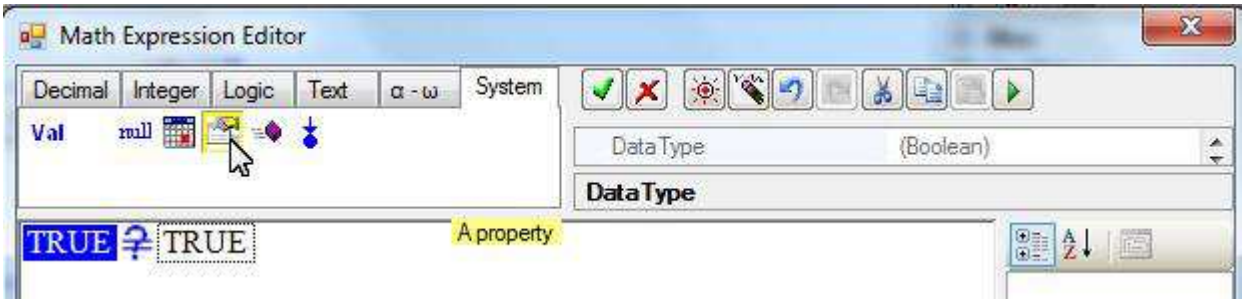
Rename the condition to “Is triangle”:

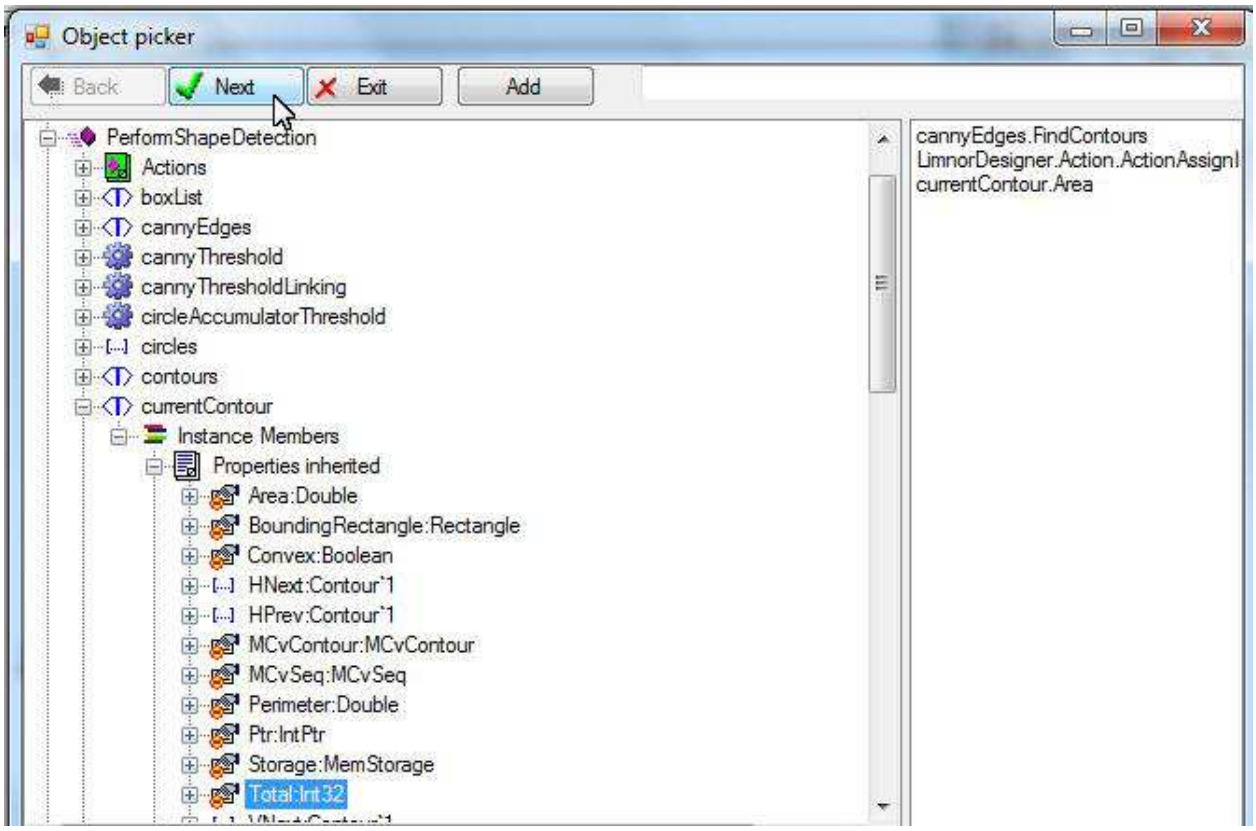


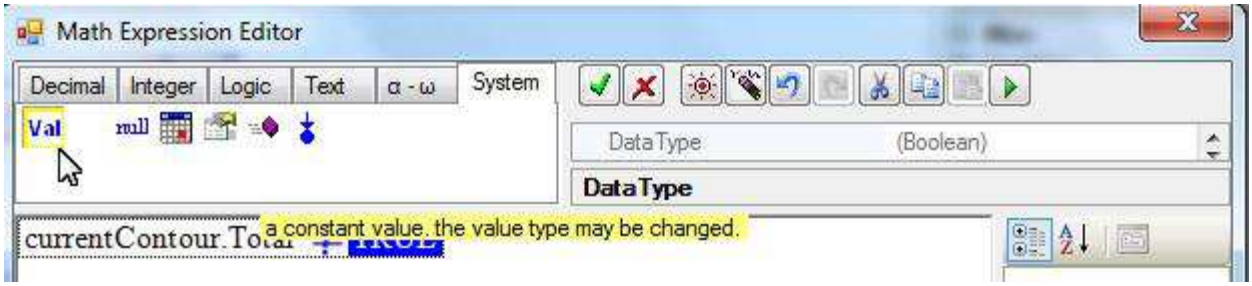
Set its condition to check if the contour has 3 points:

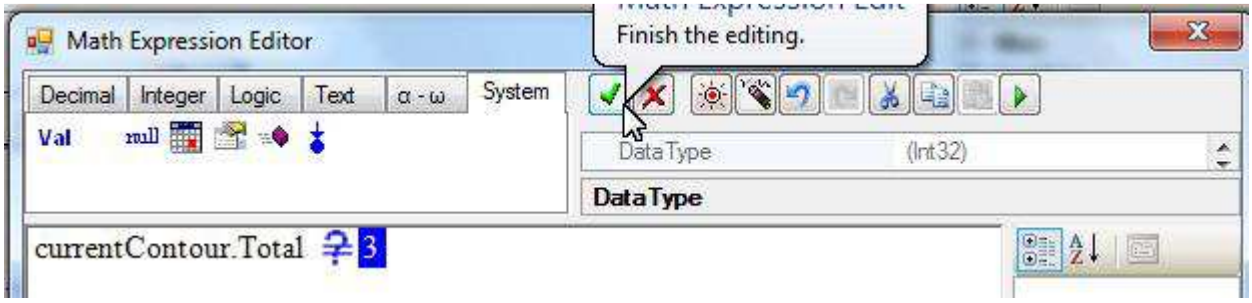




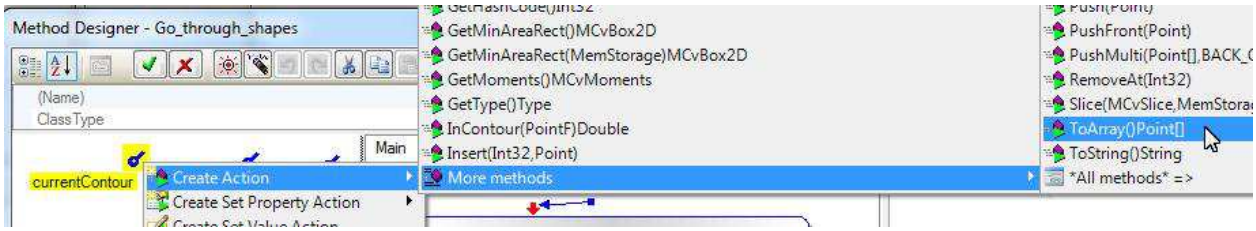


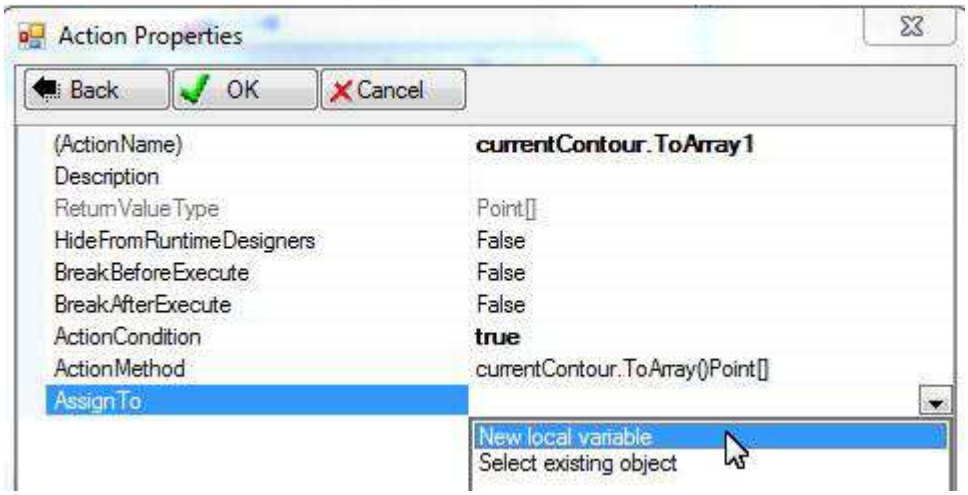


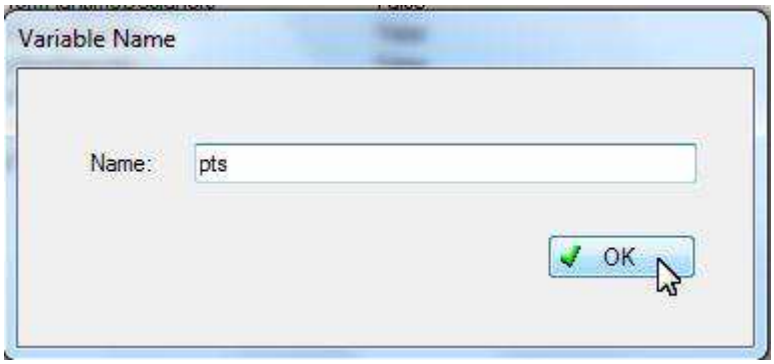


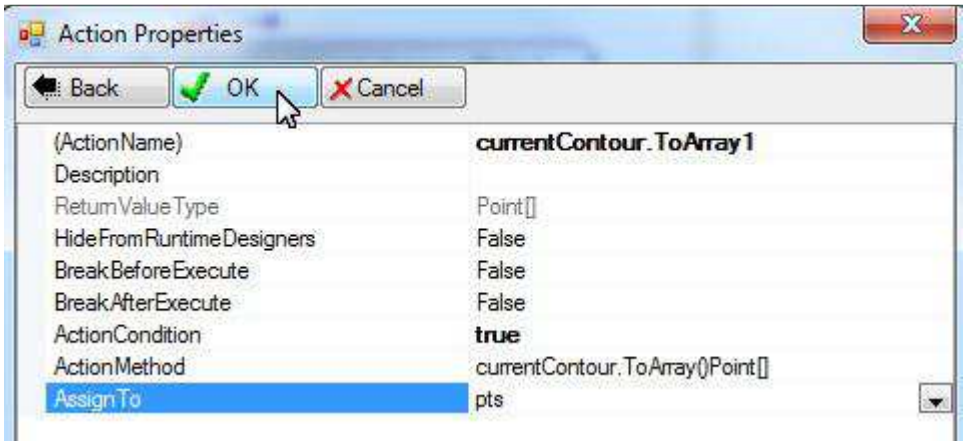


If it is a triangle then get the 3 pints from the contour and use the points to create a triangle.

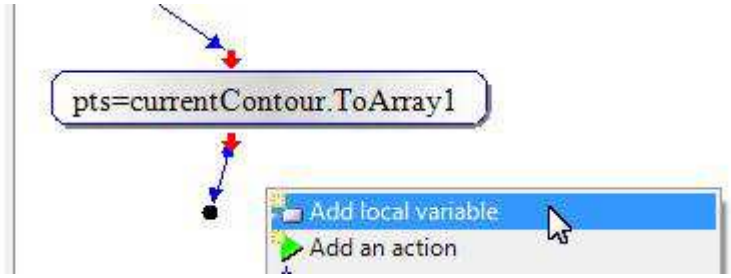


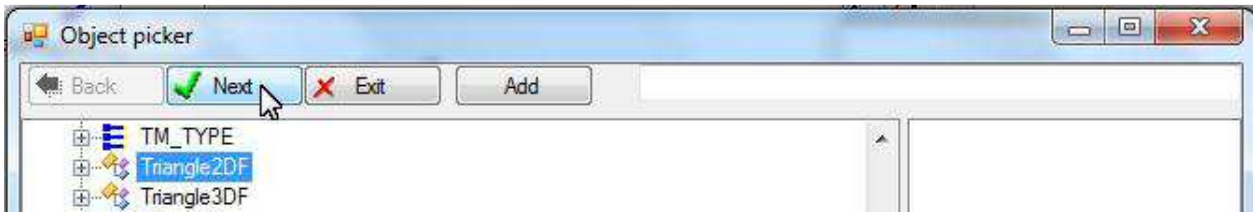


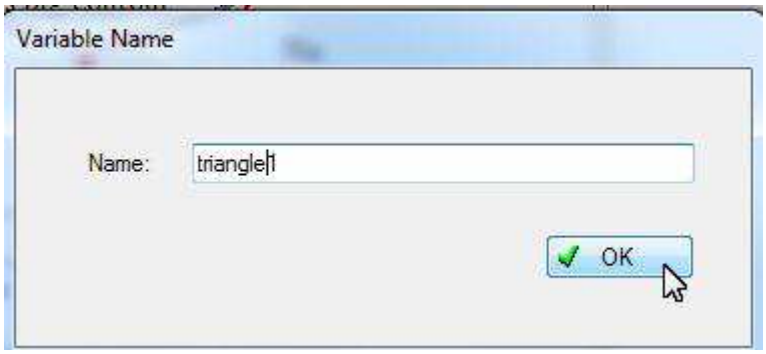


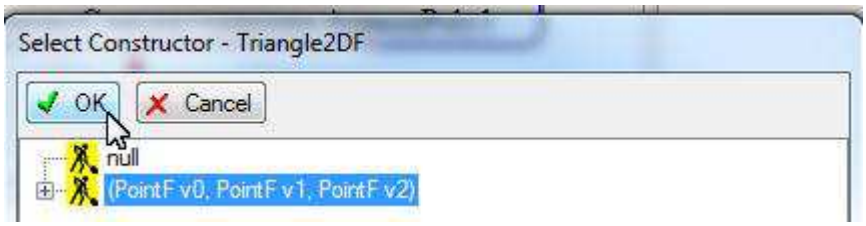


Create triangle:

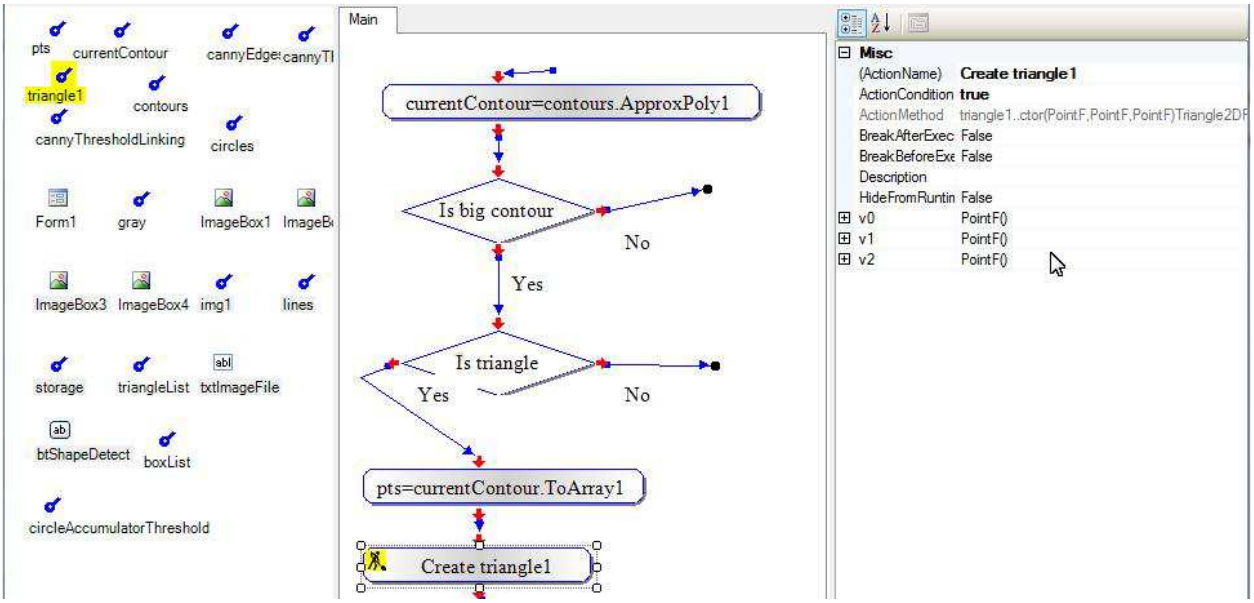


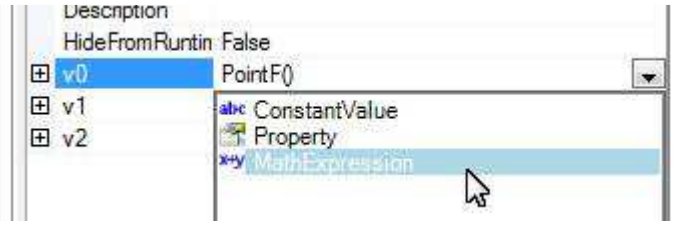


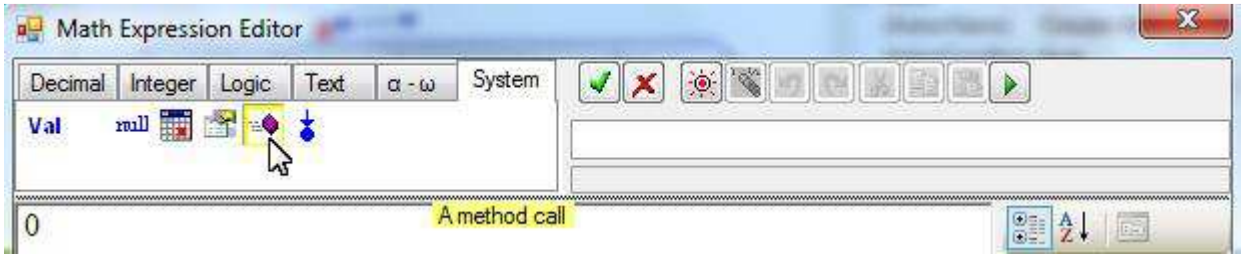


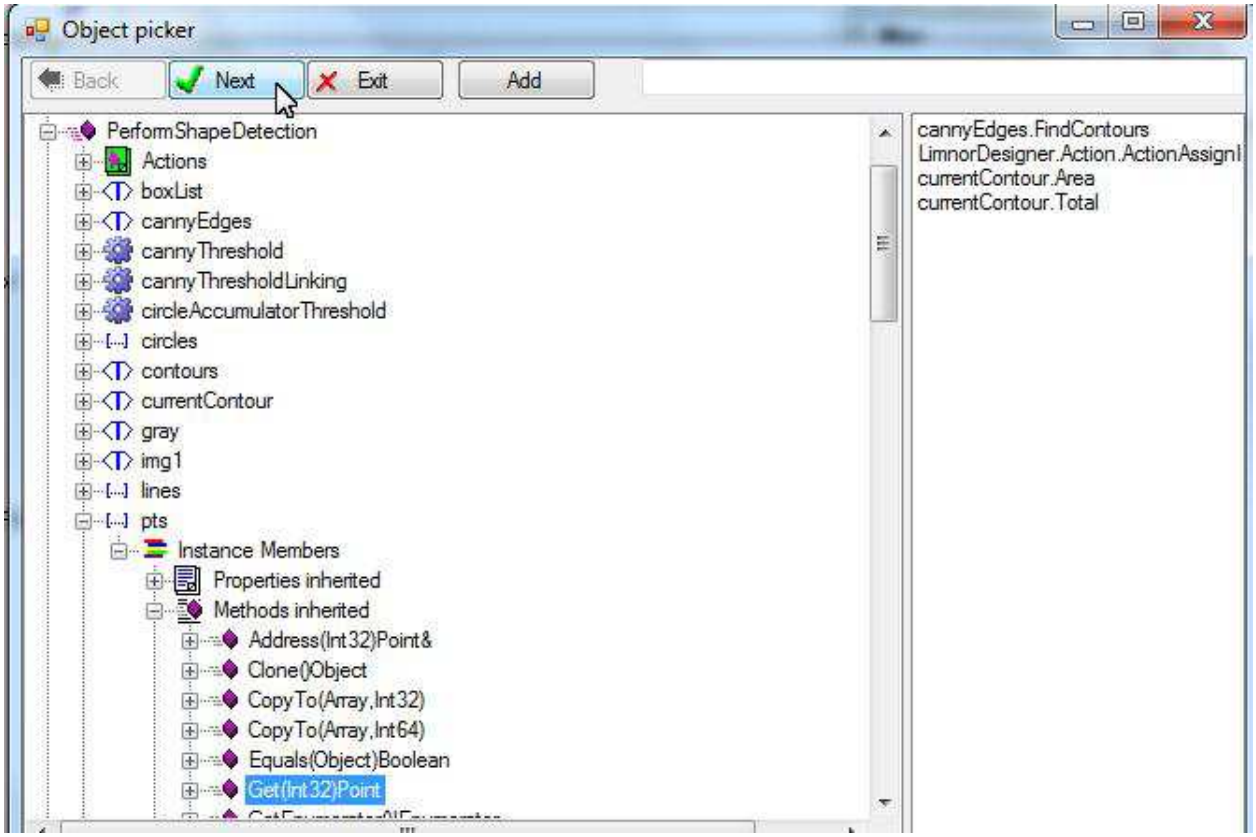


We need to specify the 3 points for the triangle:


pts
currentContour
triangle1
contours
cannyThresholdLinking
circles
Form1
gray
ImageBox1
ImageBox3
ImageBox4
img1
lines
storage
triangleList
txtImageFile
btShapeDetect
boxList
circleAccumulatorThreshold
Main
currentContour=contours.ApproxPoly1
Is big contour
No
Yes
Is triangle
Yes
No
pts=currentContour.ToArray1
Create triangle1
Misc
(ActionName) Create triangle1
ActionCondition true
ActionMethod triangle1..ctor(PointF,PointF,PointF)Triangle2DF
BreakAfterExec False
BreakBeforeExe False
Description
HideFromRuntin False
v0 PointF()
v1 PointF()
v2 PointF()

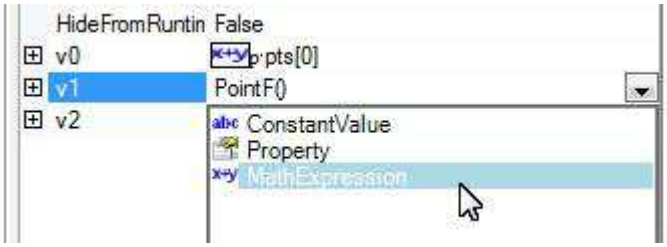

Description
HideFromRuntin False
v0
PointF()
v1
v2
ConstantValue
Property
MathExpression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
0
A method call

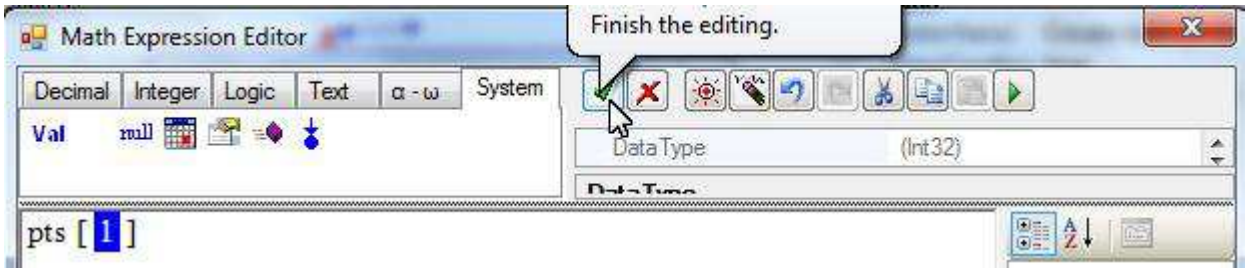


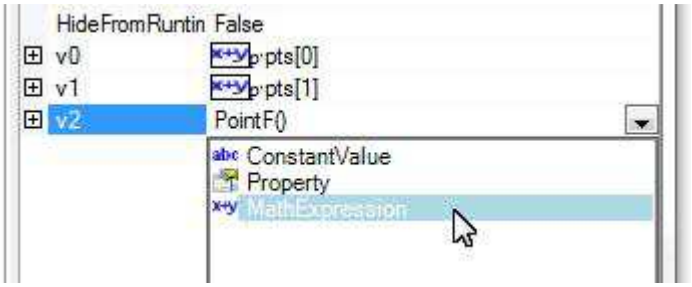
0 indicates the first point in the point array:

Do the same for the other 2 points of the triangle:

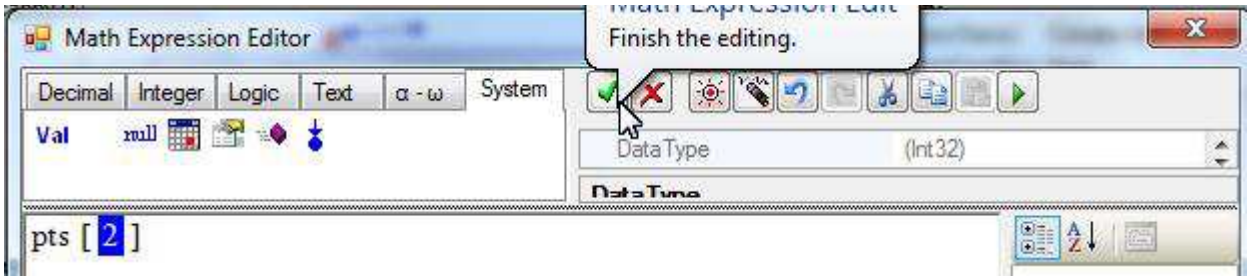


1 indicates the second point in the array:

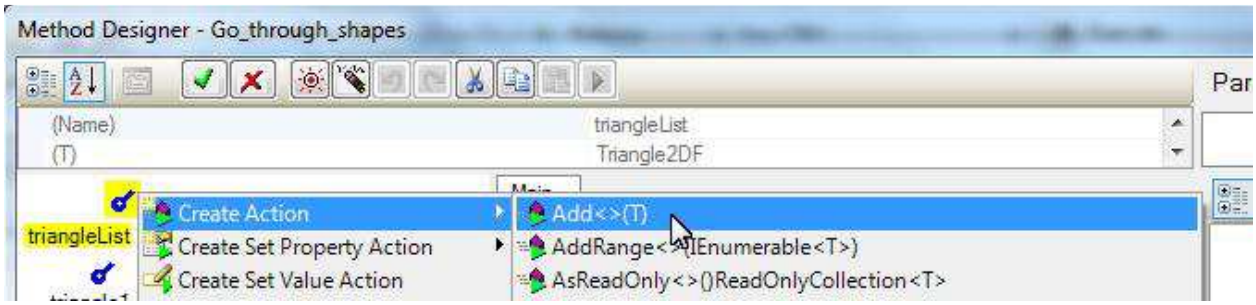


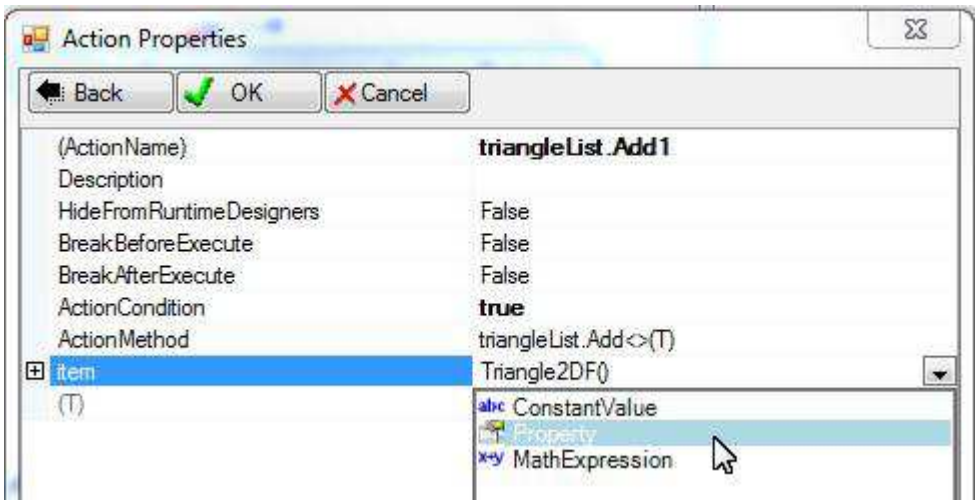


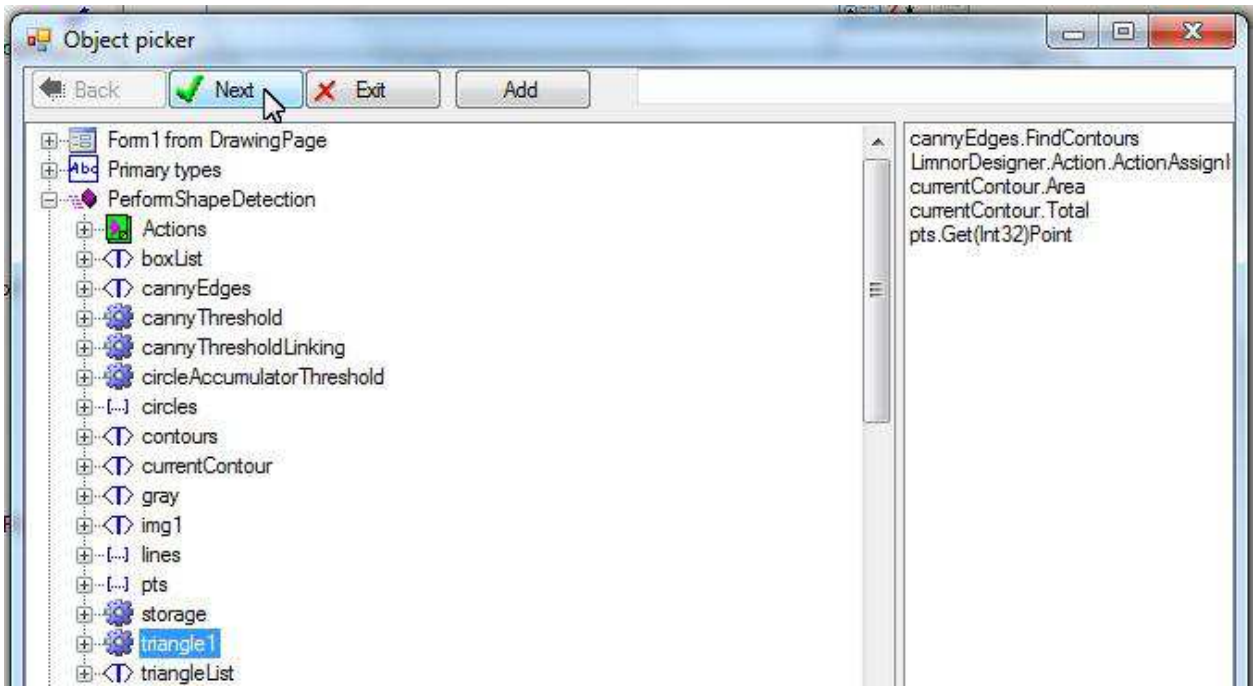
2 indicates the 3rd point in the array:

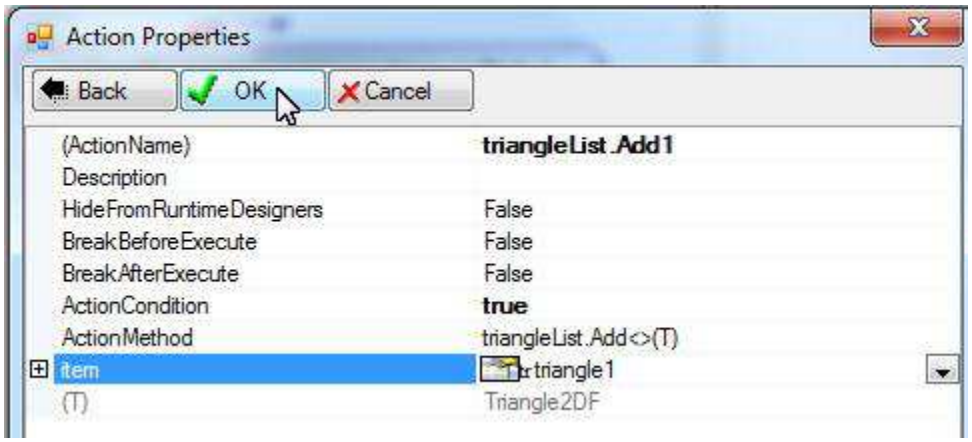


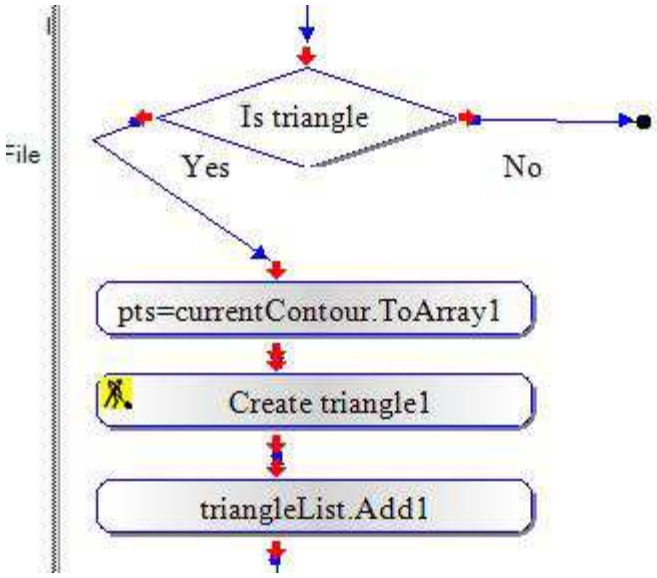
Add the triangle to the list:




Action Properties
Back
OK
Cancel
(ActionName)
triangleList.Add1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleList.Add<>(T)
item
Triangle2DF()
(T)
ConstantValue
Property
MathExpression

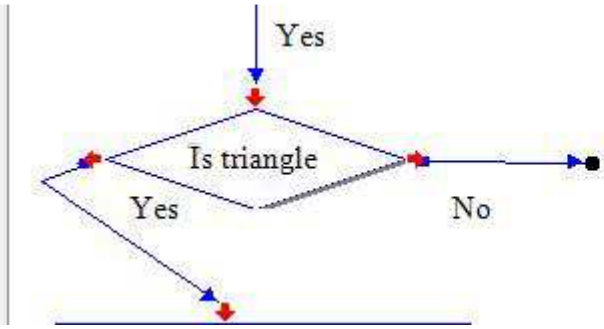

Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
currentContour
gray
img1
lines
pts
storage
triangle1
triangleList
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point

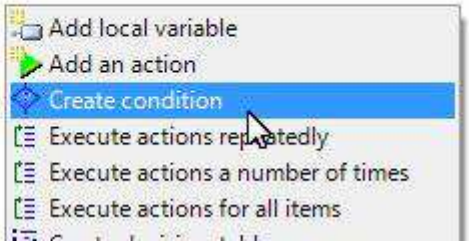

Action Properties
Back
OK
Cancel
(ActionName)
triangleList.Add1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleList.Add<>(T)
item
triangle1
(T)
Triangle2DF



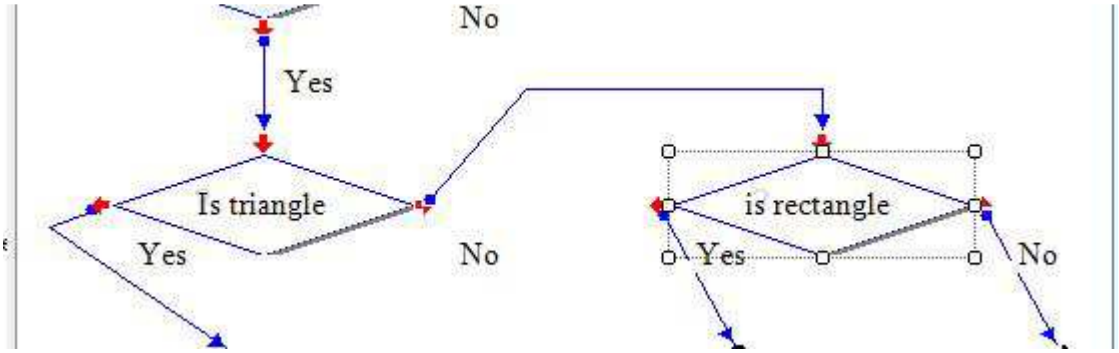
Get rectangle

If a contour has 4 vertices then it is a rectangle.

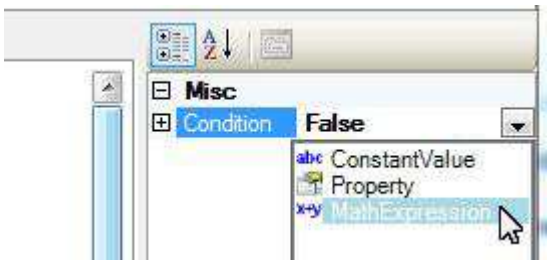


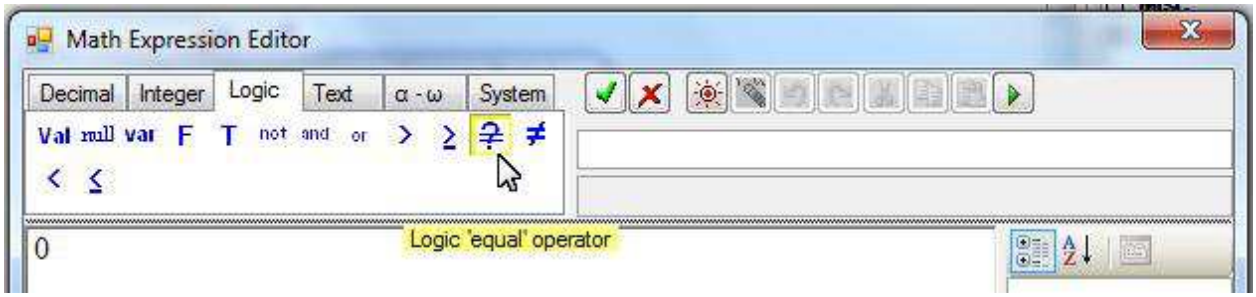


Change its name to “Is rectangle”:

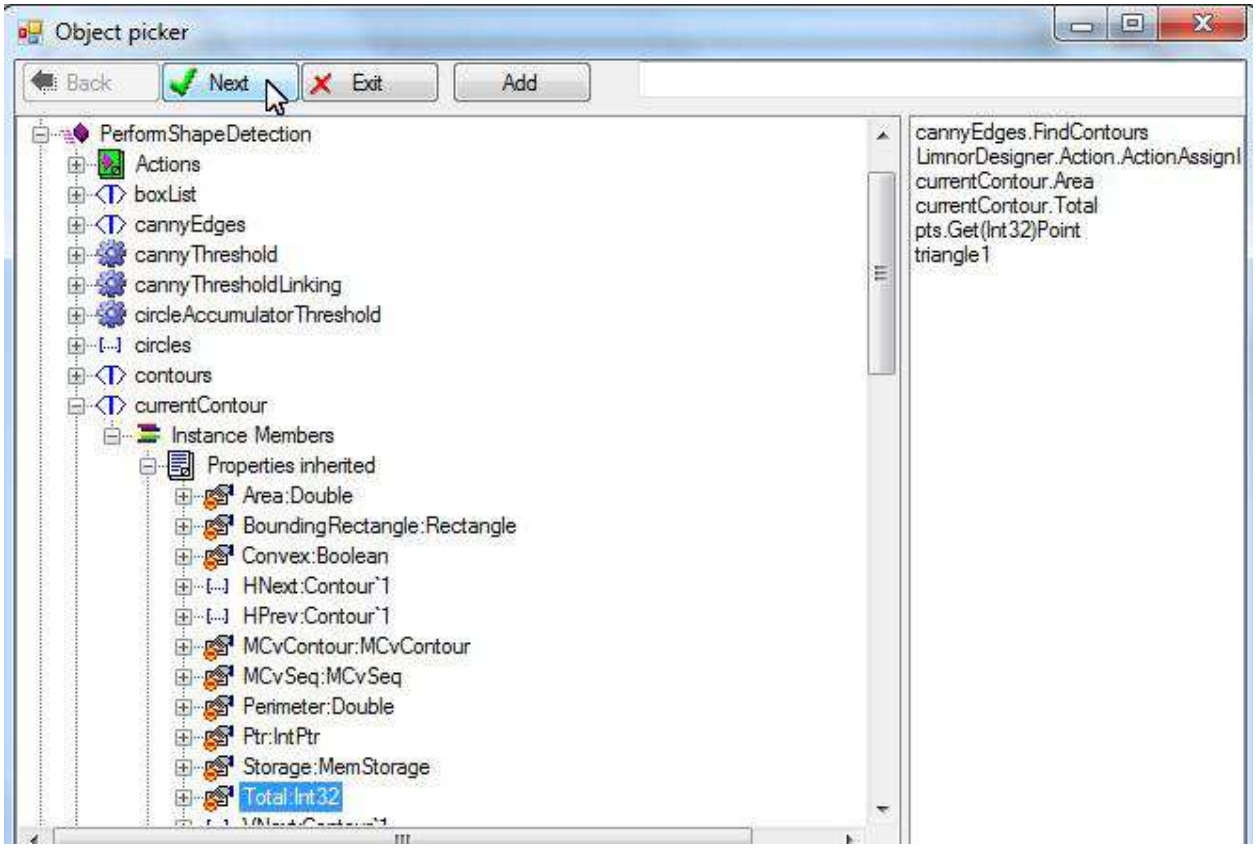


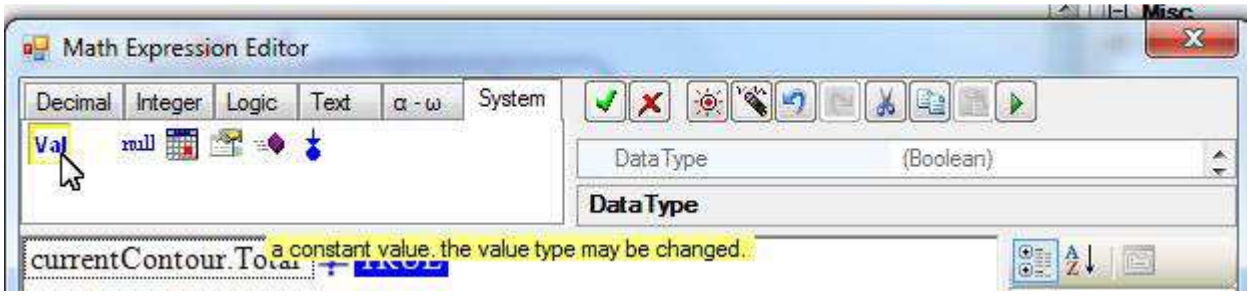
Set its condition to check if the contour has 4 vertices:

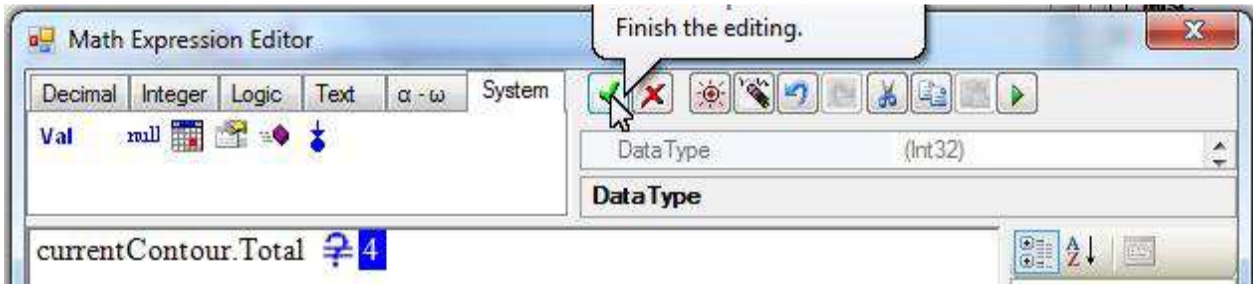

Misc
Condition
False
ConstantValue
Property
MathExpression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
0
Logic 'equal' operator

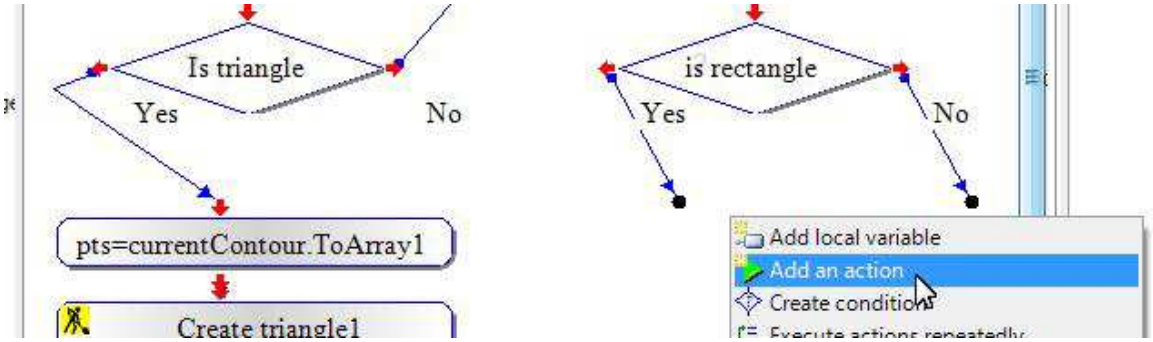

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
Data Type
(Boolean)
DataType
TRUE
TRUE
A property

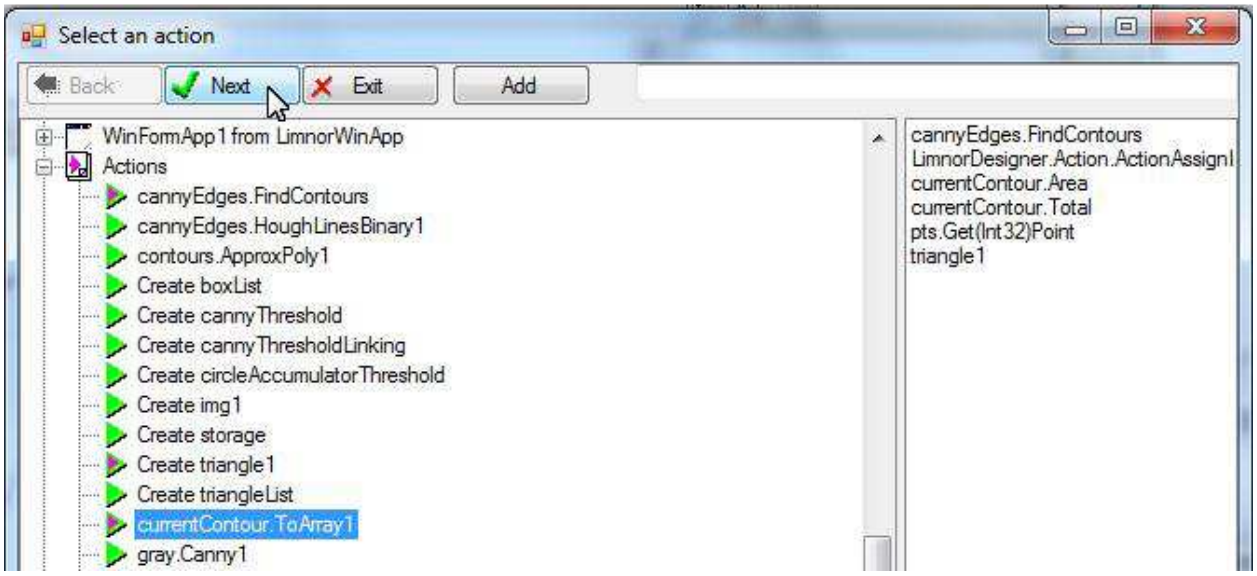


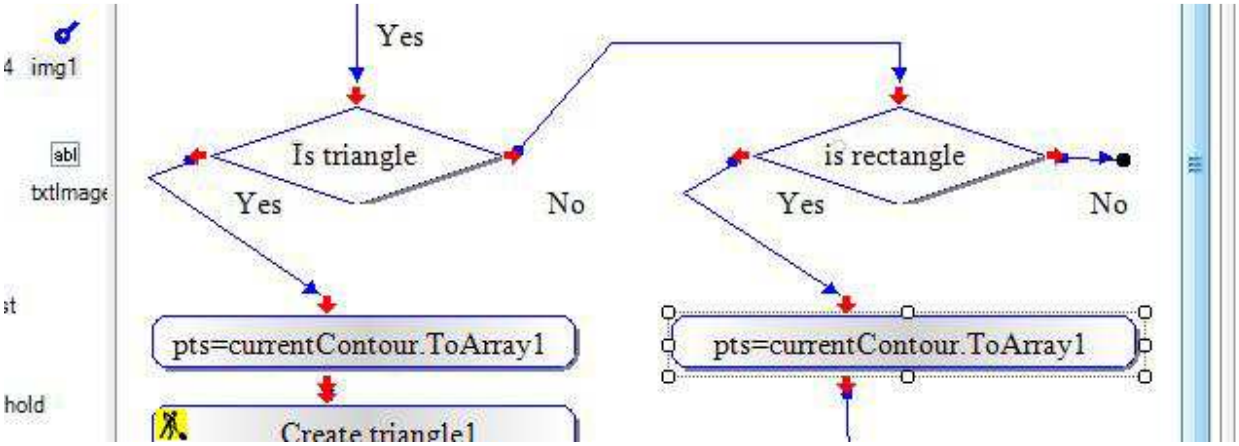




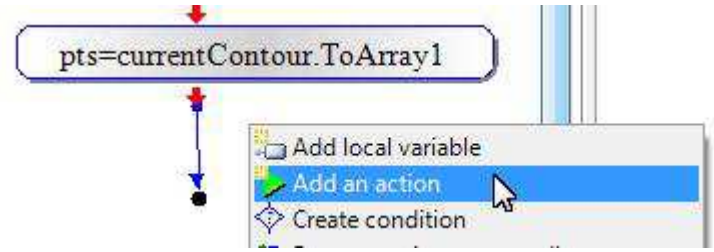
Determine if all the angles in the contour are within [80, 100] degree by forming the line segments from the contour vertices.

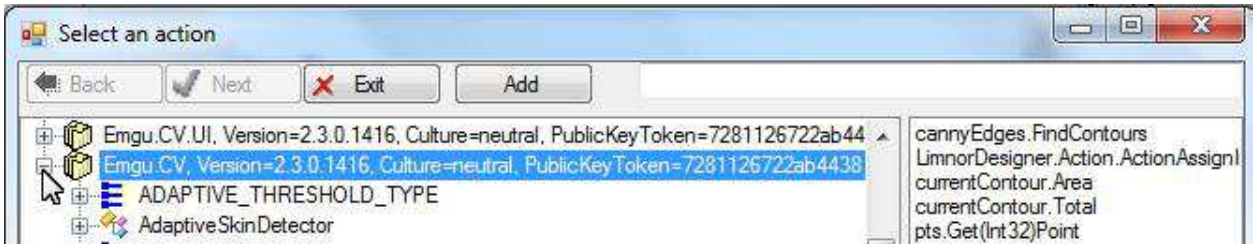


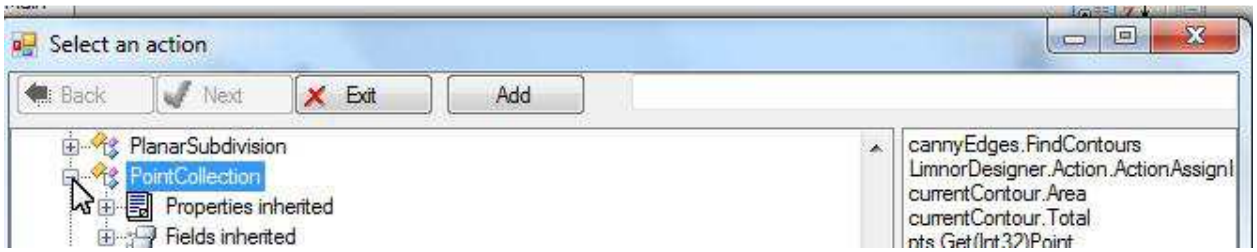


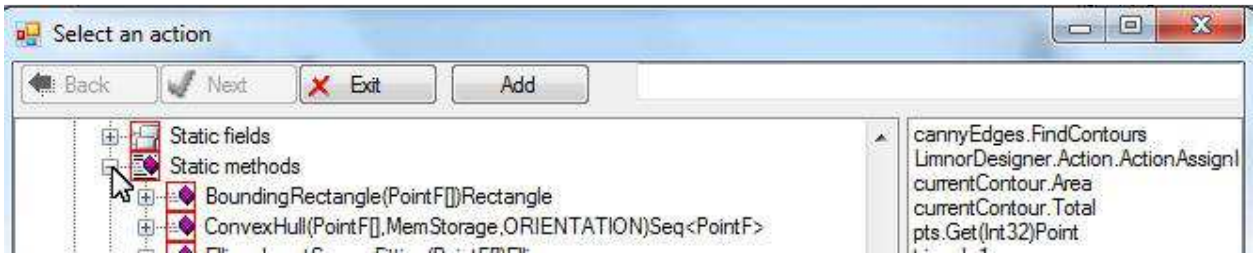


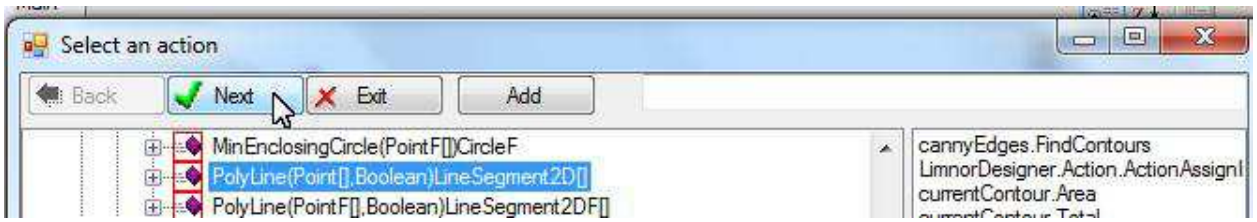
Create line segments from the point array:

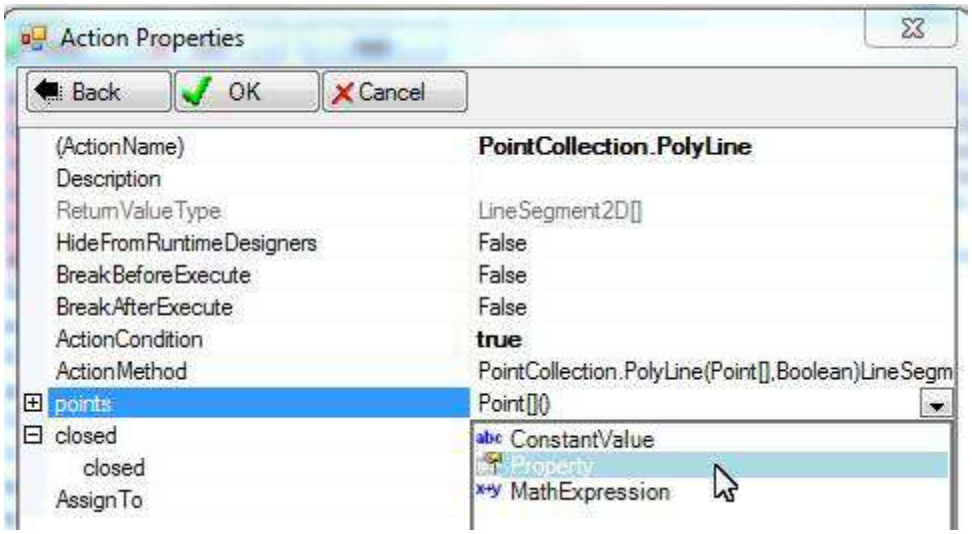

pts=currentContour.ToArray1
Add local variable
Add an action
Create condition

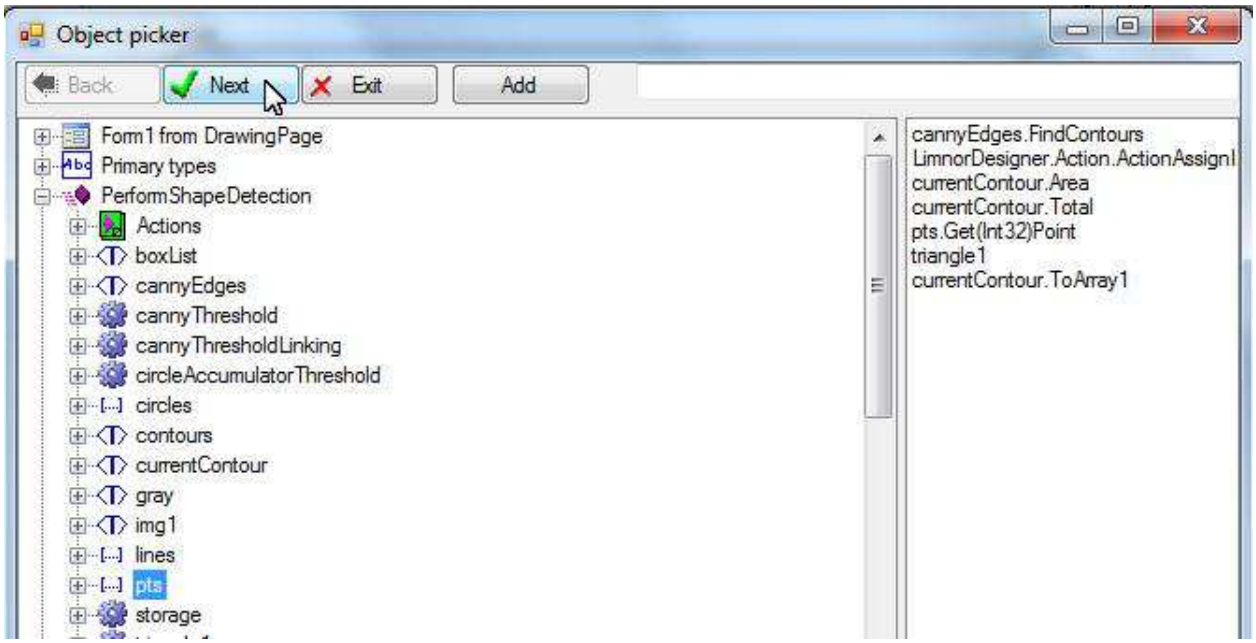

Select an action
Back
Next
Exit
Add
Emgu.CV.UI, Version=2.3.0.1416, Culture=neutral, PublicKeyToken=7281126722ab44
Emgu.CV, Version=2.3.0.1416, Culture=neutral, PublicKeyToken=7281126722ab4438
ADAPTIVE_THRESHOLD_TYPE
AdaptiveSkinDetector
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point

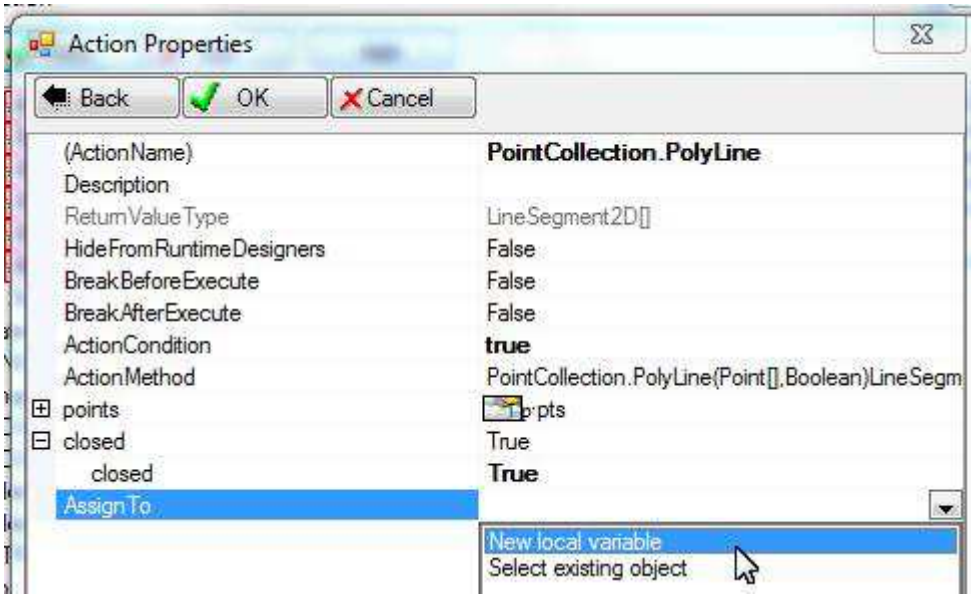

Select an action
Back
Next
Exit
Add
PlanarSubdivision
PointCollection
Properties inherited
Fields inherited
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total


Select an action
Back
Next
Exit
Add
Static fields
Static methods
BoundingRectangle(PointF[])Rectangle
ConvexHull(PointF[],MemStorage,ORIENTATION)Seq<PointF>
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point

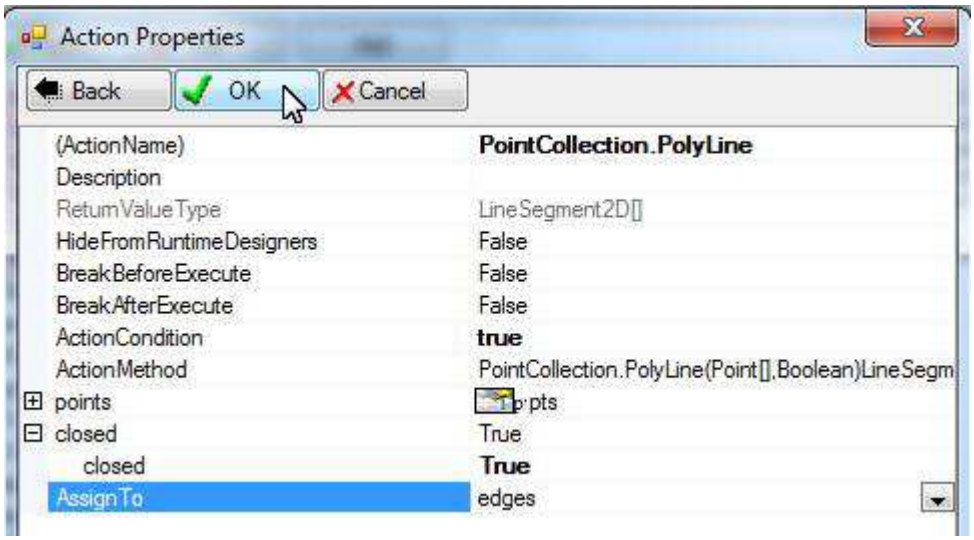

Select an action
Back
Next
Exit
Add
MinEnclosingCircle(PointF[])CircleF
PolyLine(Point[],Boolean)LineSegment2D[]
PolyLine(PointF[],Boolean)LineSegment2DF[]
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area

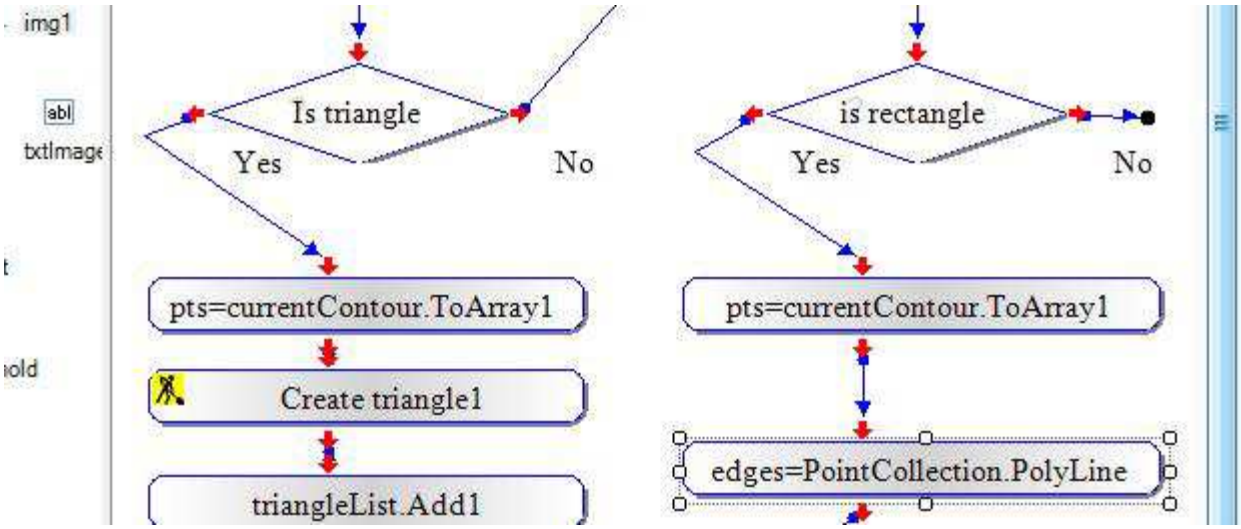

Action Properties
Back
OK
Cancel
(ActionName)
PointCollection.PolyLine
Description
ReturnValueType
LineSegment2D[]
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
PointCollection.PolyLine(Point[],Boolean)LineSegm
points
Point[]()
closed
closed
AssignTo
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
currentContour
gray
img1
lines
pts
storage
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point
triangle1
currentContour.ToArray1

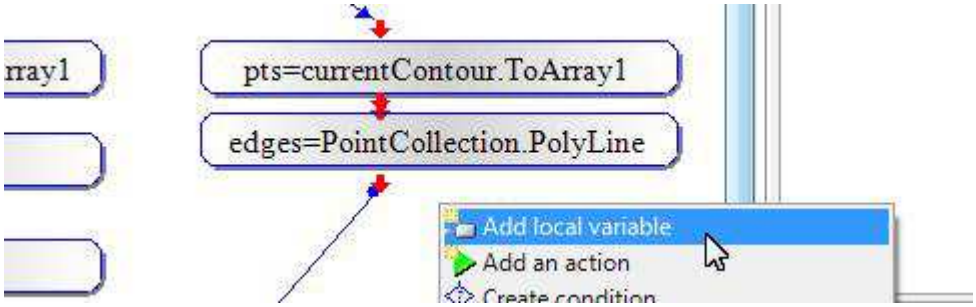

Action Properties
Back
OK
Cancel
(ActionName)
PointCollection.PolyLine
Description
ReturnValueType
LineSegment2D[]
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
PointCollection.PolyLine(Point[],Boolean)LineSegm
points
pts
closed
True
closed
True
AssignTo
New local variable
Select existing object

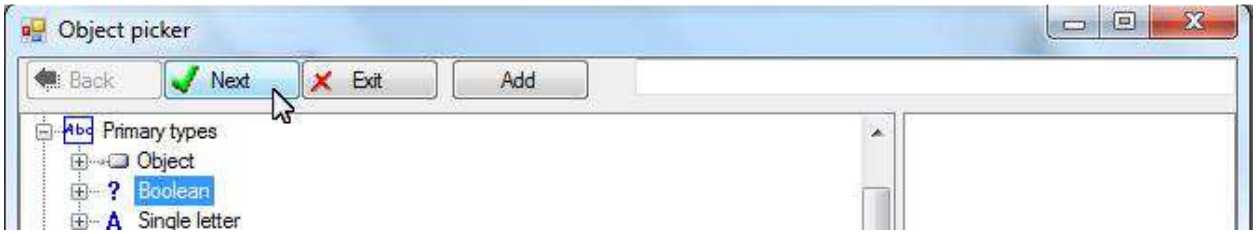

Variable Name
Name:
edges
OK

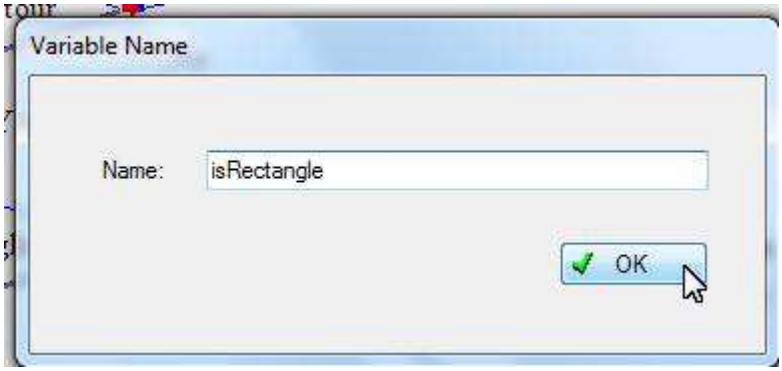

Action Properties
Back
OK
Cancel
(ActionName)
PointCollection.PolyLine
Description
ReturnValueType
LineSegment2D[]
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
PointCollection.PolyLine(Point[],Boolean)LineSegm
points
pts
closed
True
closed
True
AssignTo
edges

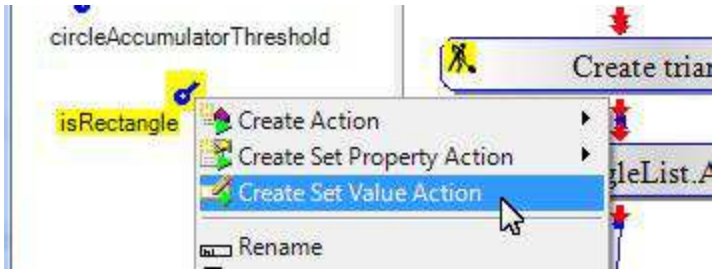


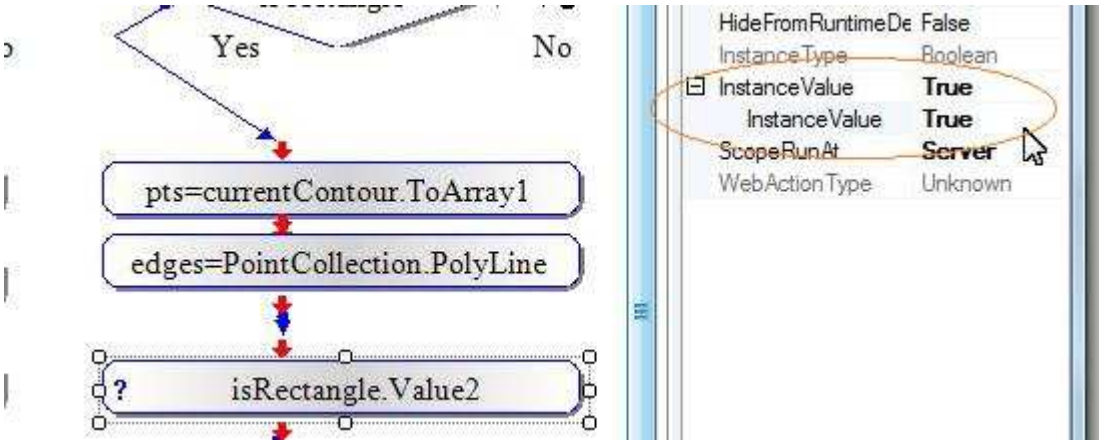
Each angle in a rectangle is about 90 degree. We'll go through the edges to see if there is an angle outside of [80, 100] degree. Before doing that we create a Boolean variable and set it True, indicating that we assume the contour is a rectangle. If we found an angle outside of [80, 100] degree then we set the variable to False, indicating that actually the contour is not a rectangle.



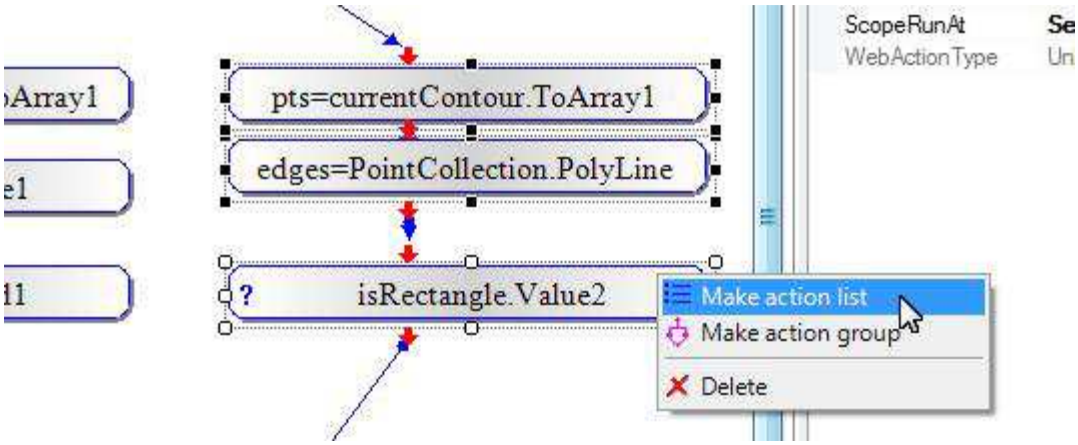




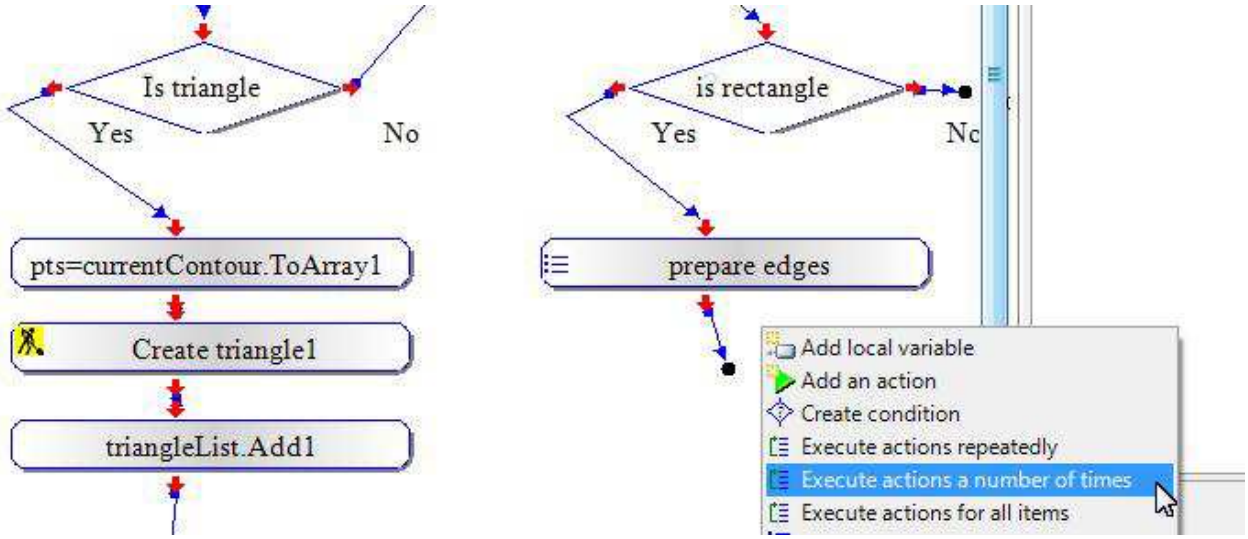


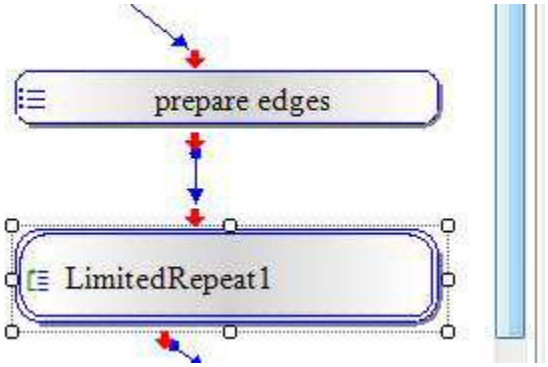


Make the 3 new actions into an action list to make room in the editor:

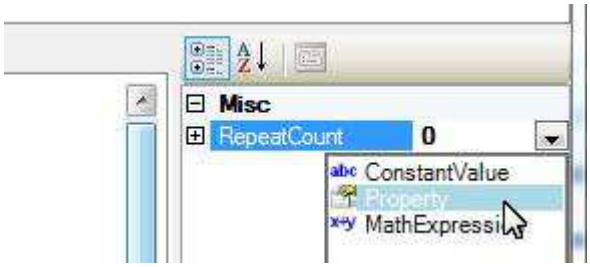


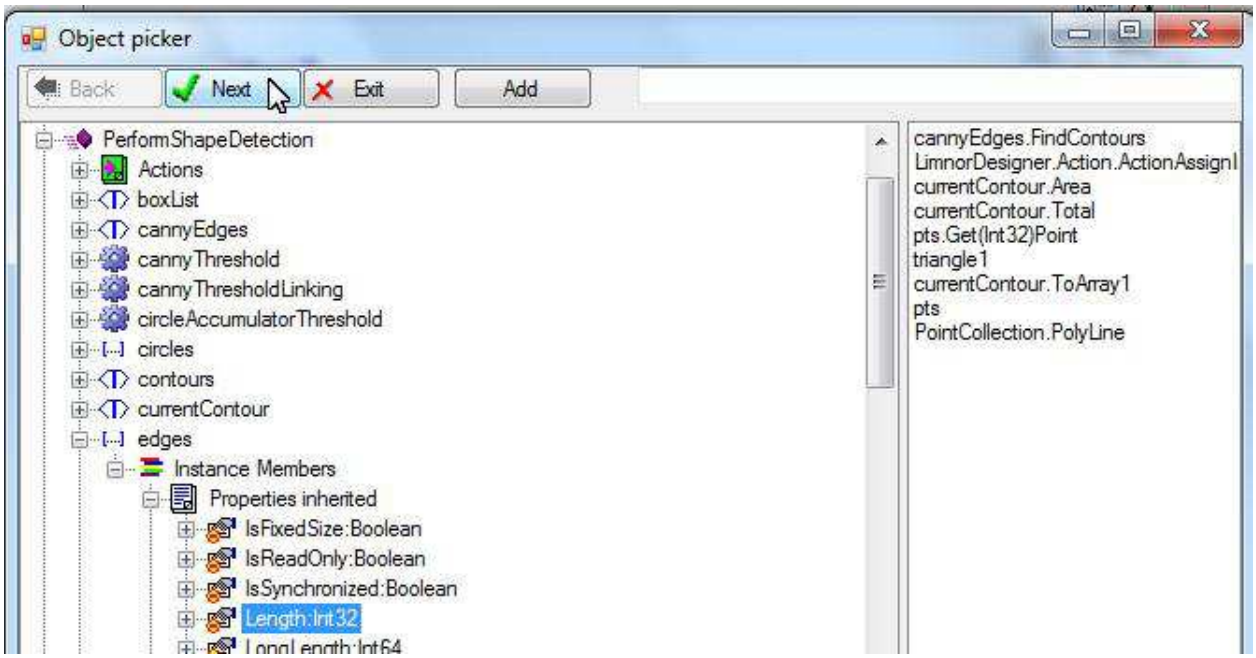
Create an action to go through edges:



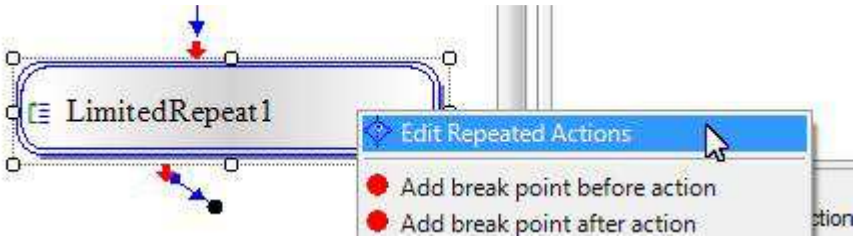


Set its repeat count to the number of edges:

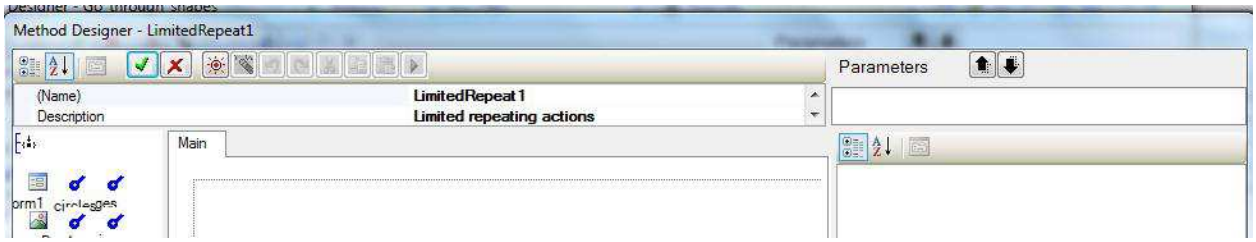




To create actions to process each edge, right-click the loop action, choose “Edit Repeated Actions”:

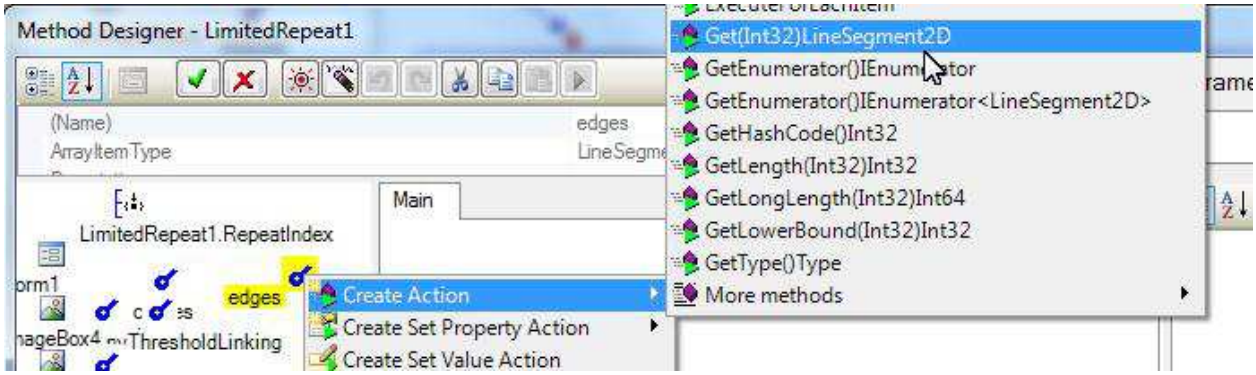


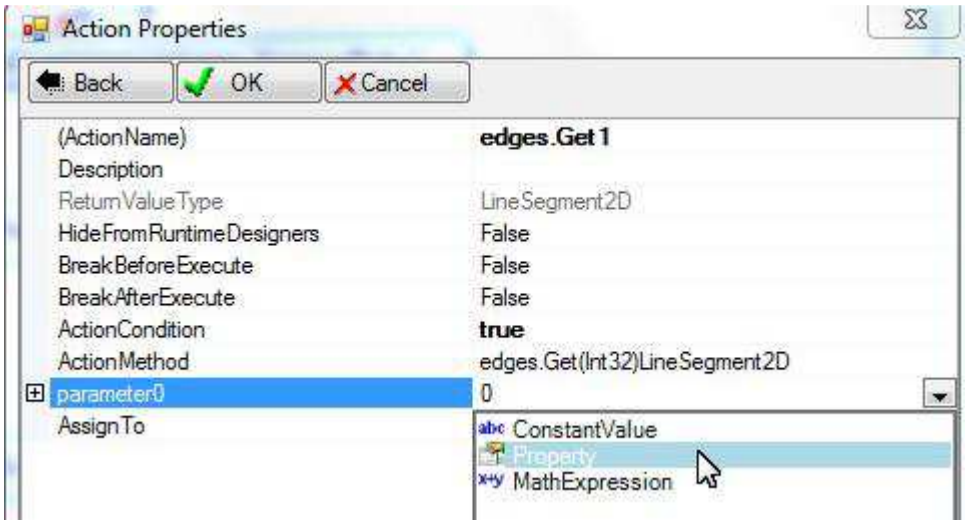
A new Method Editor appears:

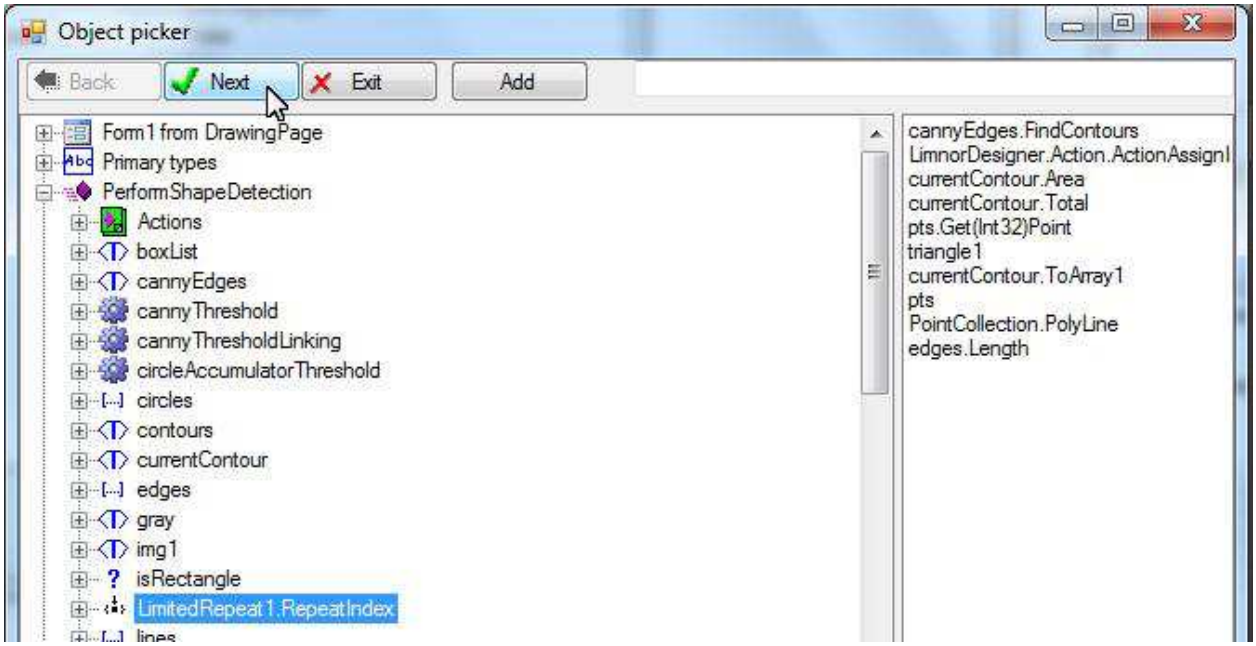


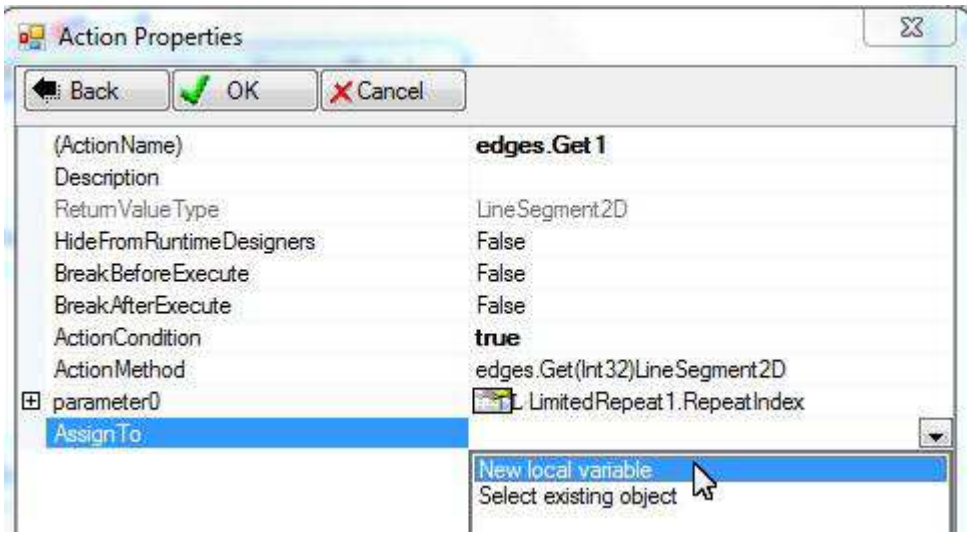
Note the “RepeatedIndex” icon. It represents the edge index, from 0 to number of edges minus 1.

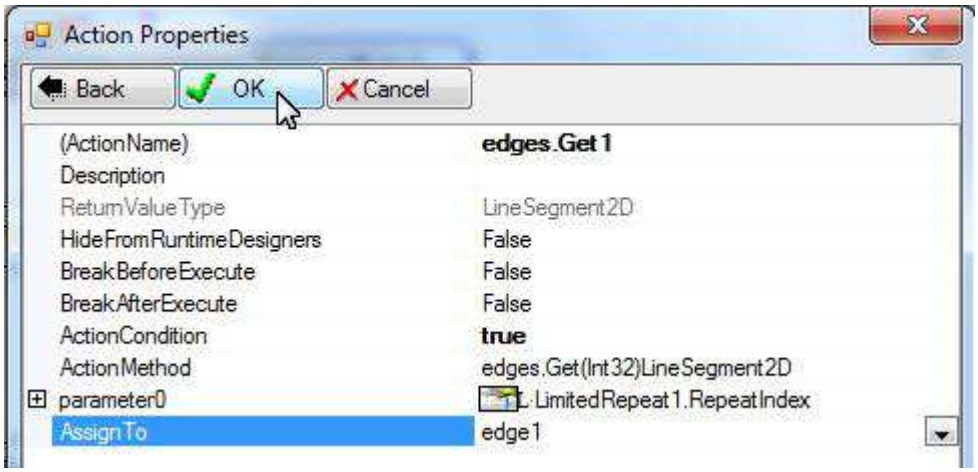
We'll use the "RepeatedIndex" to get the two edges for checking the angle.



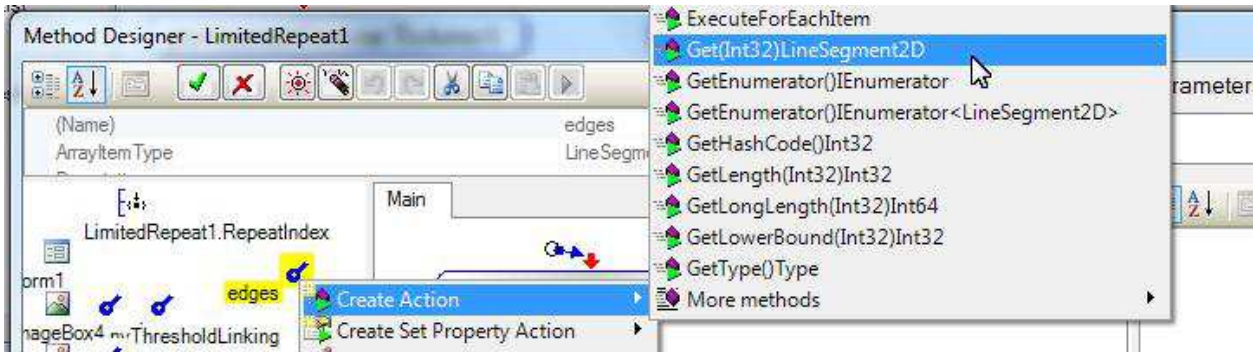


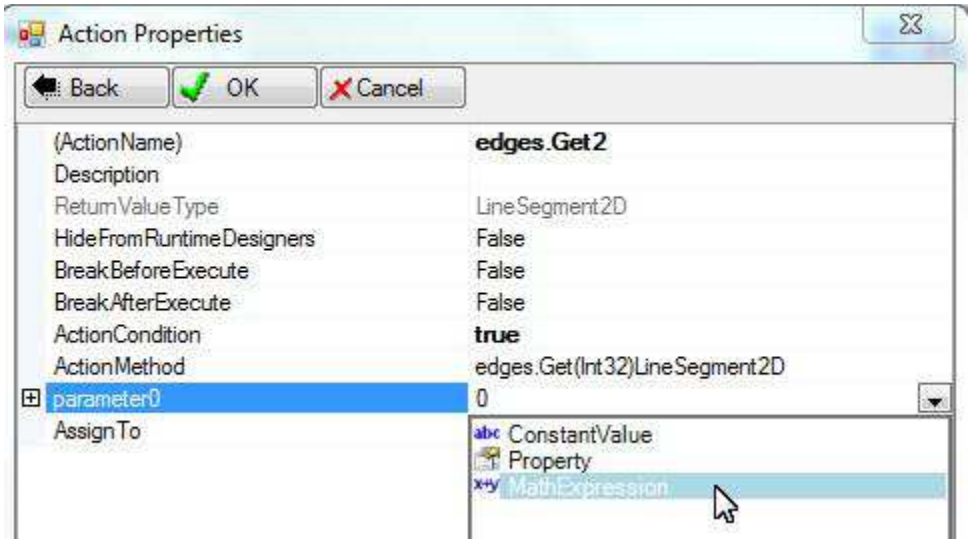


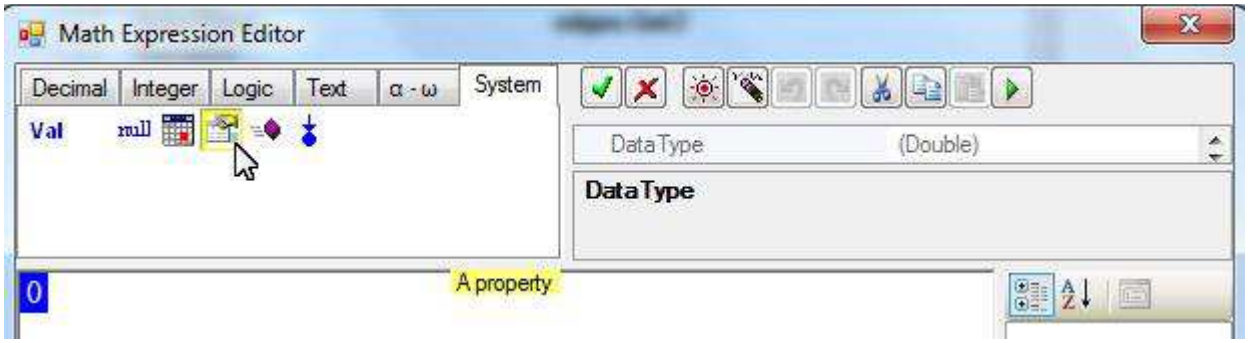


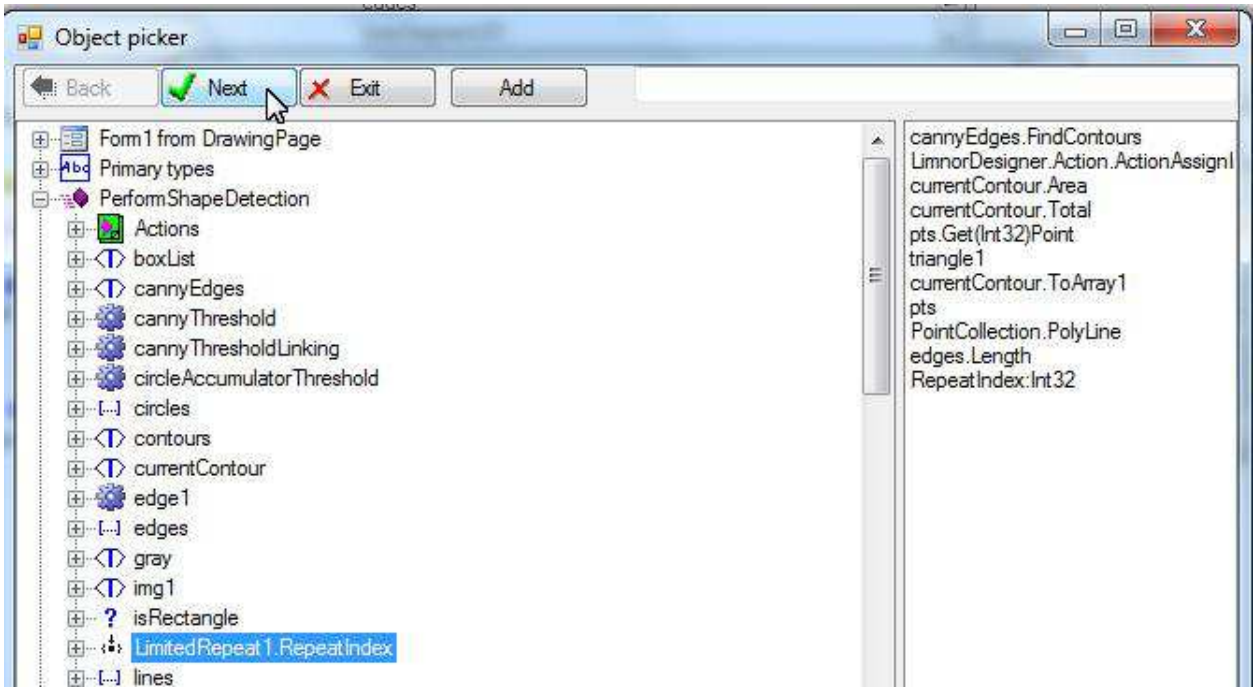


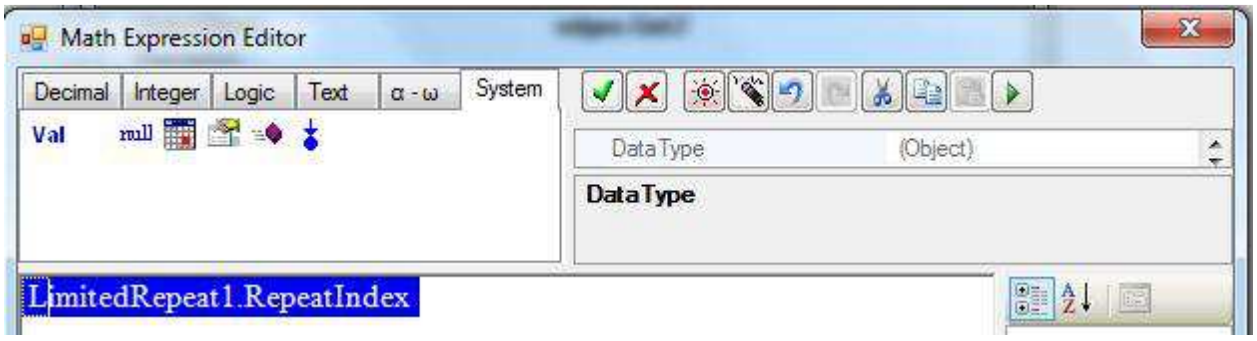
Get the second edge:


Method Designer - LimitedRepeat1
(Name)
ArrayItemType
LimitedRepeat1.RepeatIndex
Main
edges
Create Action
Create Set Property Action
ExecuteForEachItem
Get(Int32)LineSegment2D
GetEnumerator()IEnumerator
GetEnumerator()IEnumerator<LineSegment2D>
GetHashCode()Int32
GetLength(Int32)Int32
GetLongLength(Int32)Int64
GetLowerBound(Int32)Int32
GetType()Type
More methods

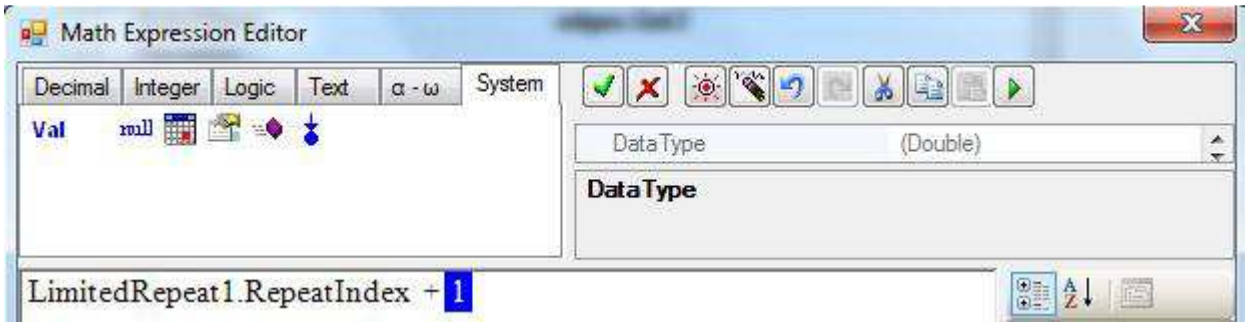

Action Properties
Back
OK
Cancel
(ActionName)
edges.Get2
Description
ReturnValueType
LineSegment2D
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
edges.Get(Int32)LineSegment2D
parameter0
0
AssignTo
ConstantValue
Property
MathExpression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Double)
DataType
0
A property

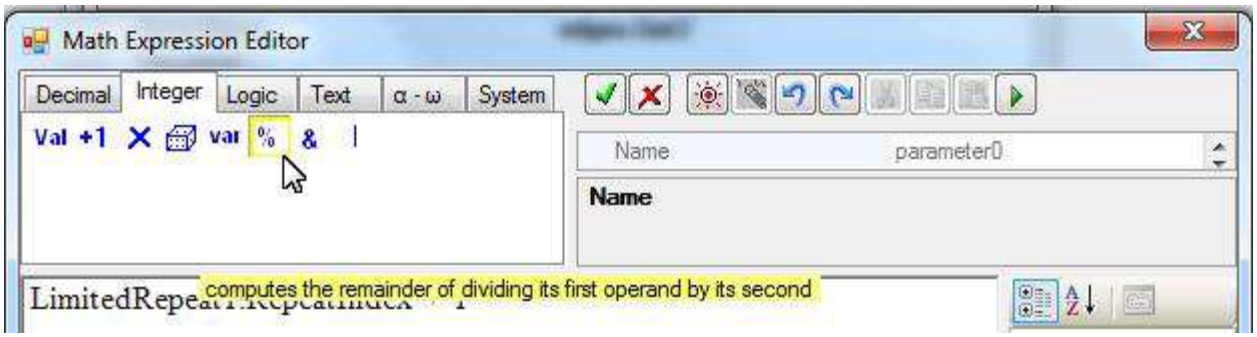




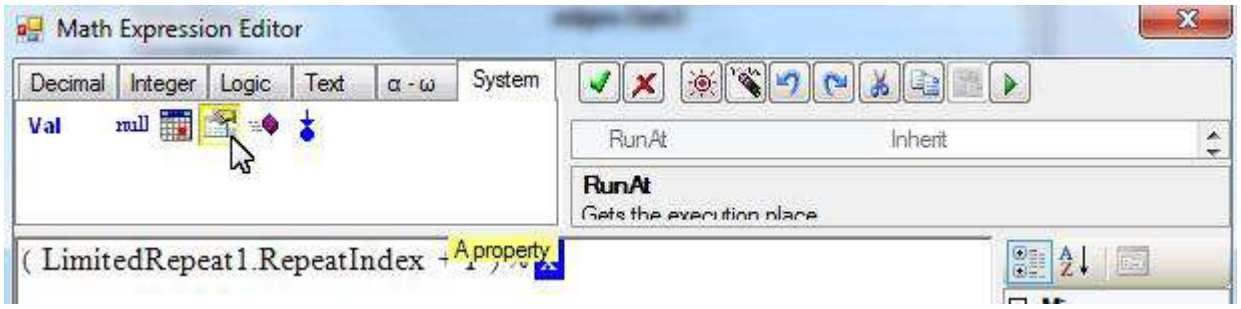
Press “+1”:

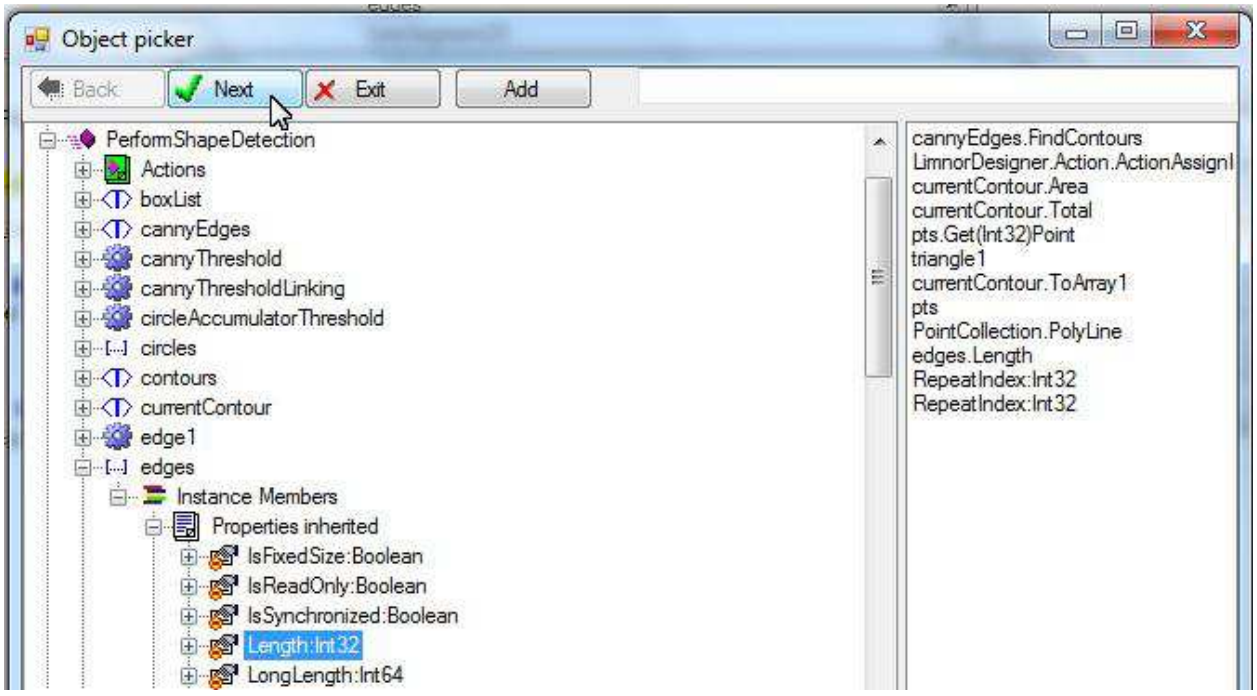


Make sure nothing is highlighted. Click “%”:

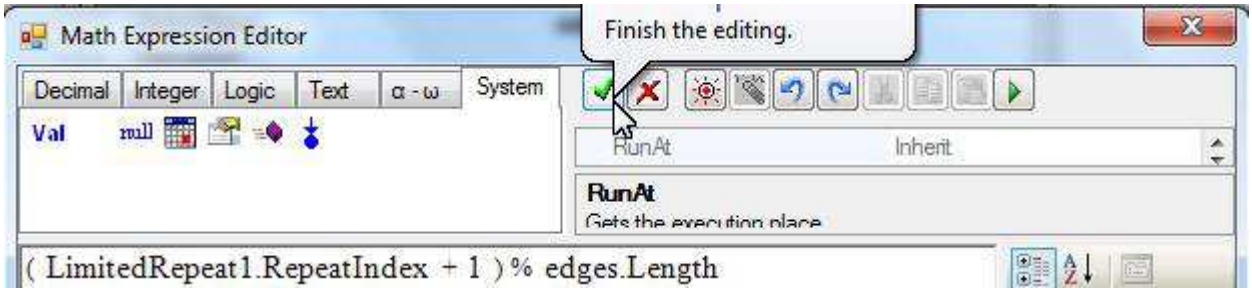

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val +1
var
%
&
Name
parameter0
Name
computes the remainder of dividing its first operand by its second

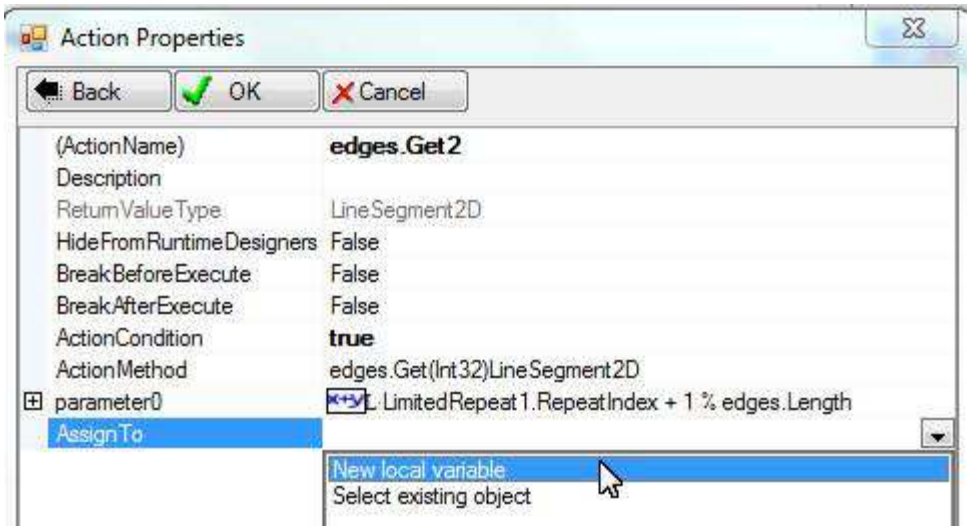

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
RunAt
Inherit
RunAt
(LimitedRepeat1.RepeatIndex + 1) %

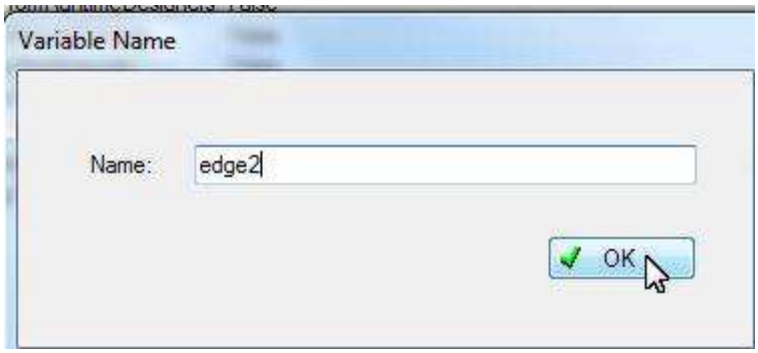

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
RunAt
Inherit
RunAt
(LimitedRepeat1.RepeatIndex
A property

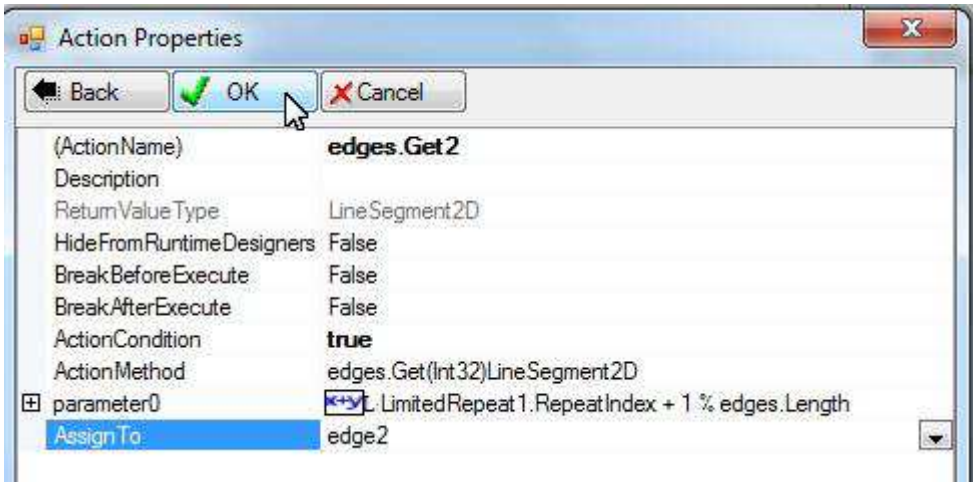

Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
currentContour
edge1
edges
Instance Members
Properties inherited
IsFixedSize:Boolean
IsReadOnly:Boolean
IsSynchronized:Boolean
Length:Int32
LongLength:Int64
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point
triangle1
currentContour.ToArray1
pts
PointCollection.PolyLine
edges.Length
RepeatIndex:Int32
RepeatIndex:Int32

This expression is the array index for the second edge:

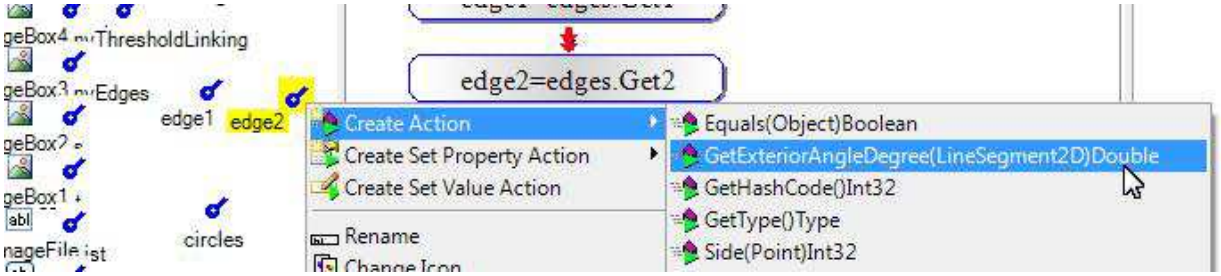




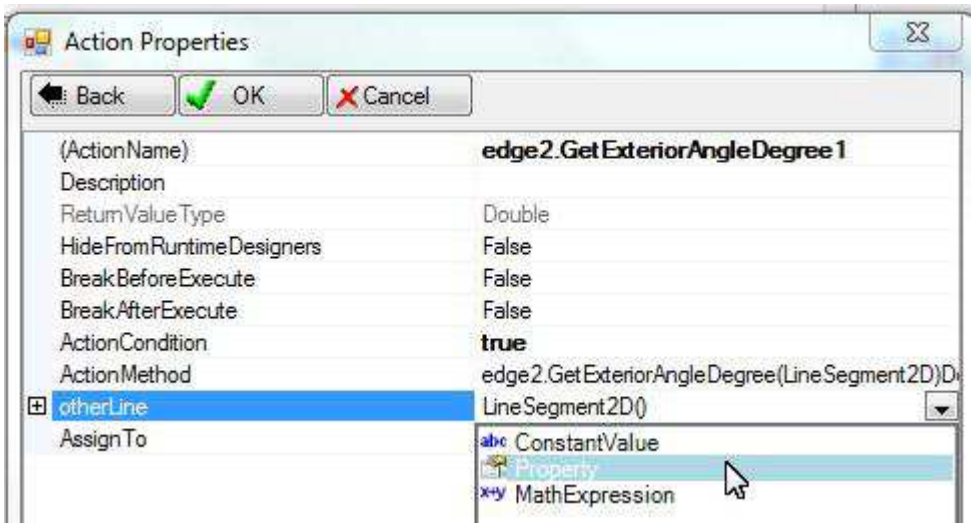


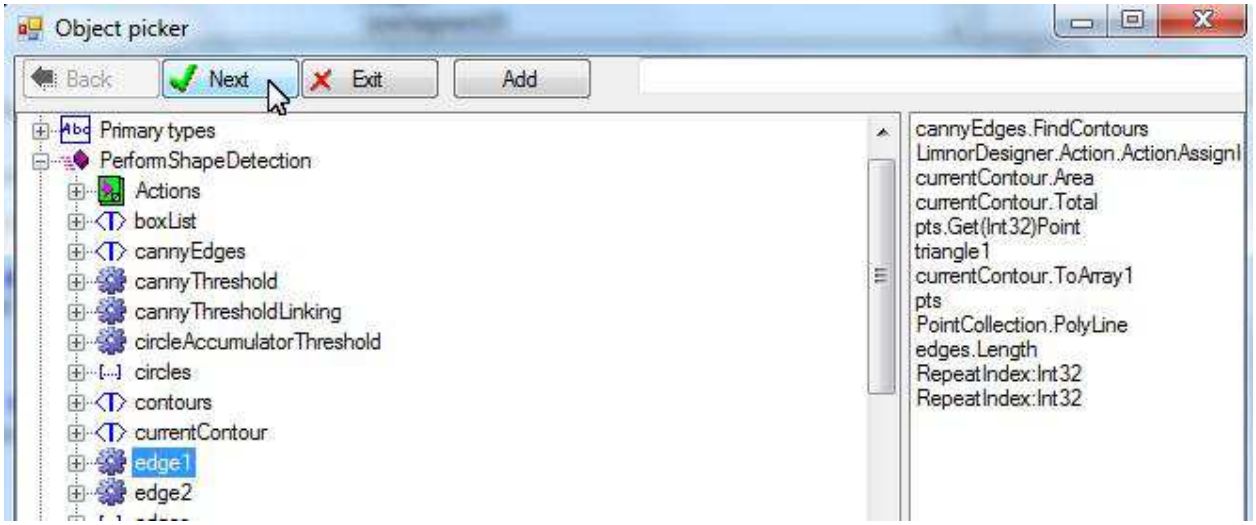


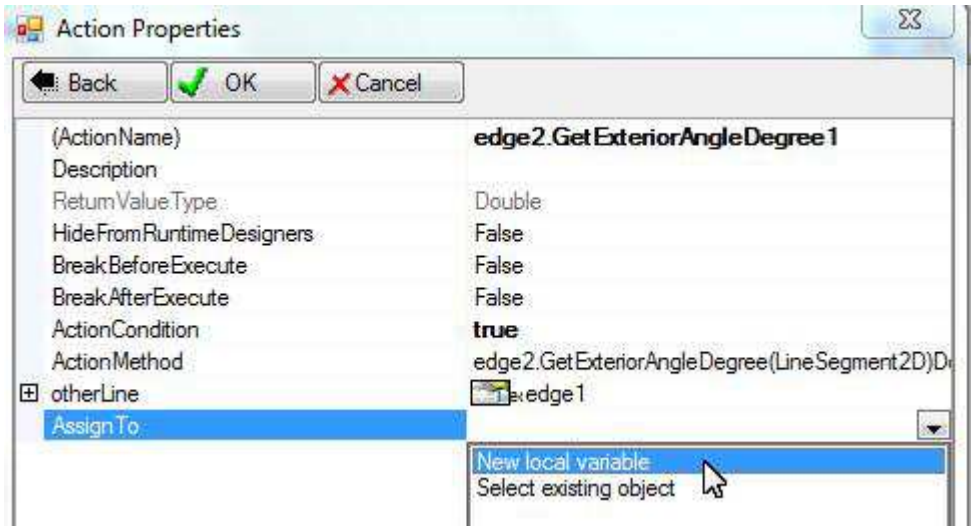
Use the second edge's `GetExteriorAngleDegree` method to get angle:

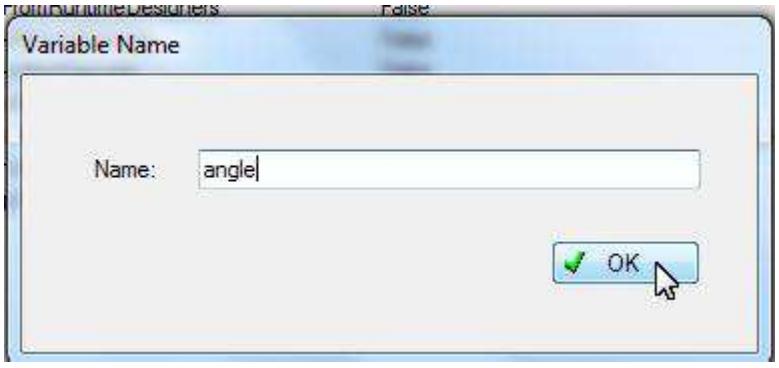


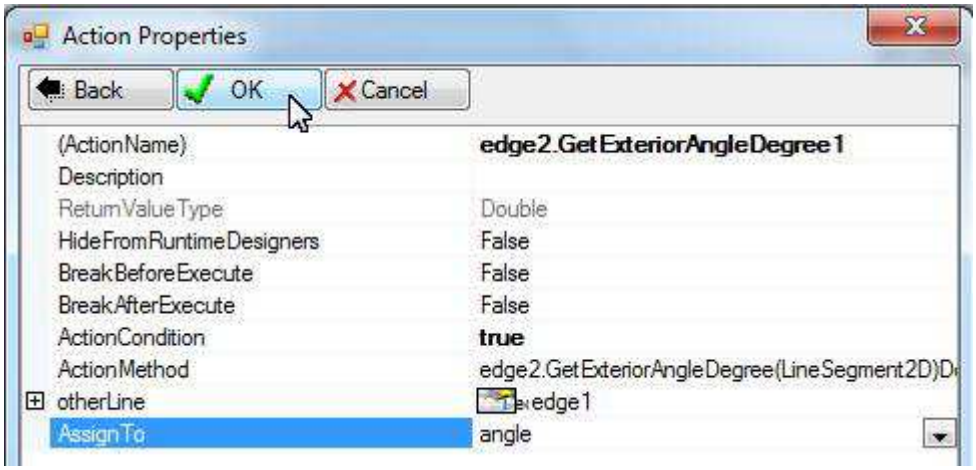
Set "otherLine" to edge1:



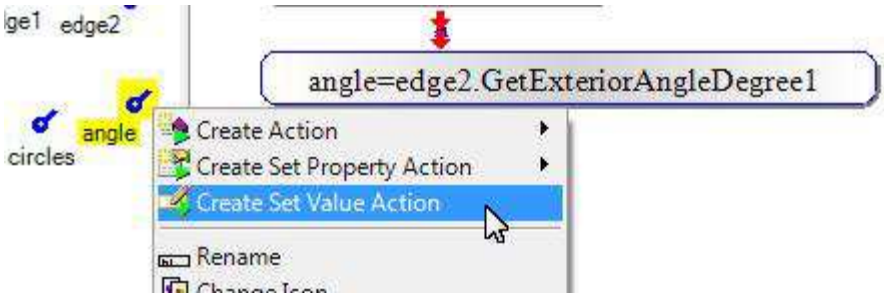

Object picker
Back
Next
Exit
Add
Primary types
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
currentContour
edge1
edge2
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point
triangle1
currentContour.ToArray1
pts
PointCollection.PolyLine
edges.Length
RepeatIndex:Int32
RepeatIndex:Int32

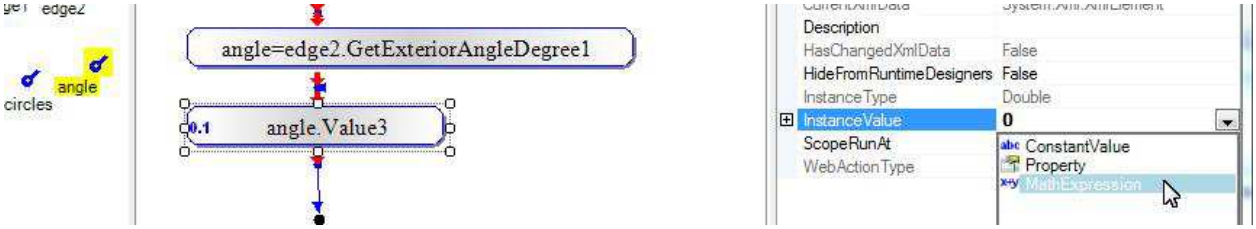

Action Properties
Back
OK
Cancel
(ActionName)
edge2.GetExteriorAngleDegree1
Description
ReturnValueType
Double
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
edge2.GetExteriorAngleDegree(LineSegment2D)D
otherLine
edge1
AssignTo
New local variable
Select existing object

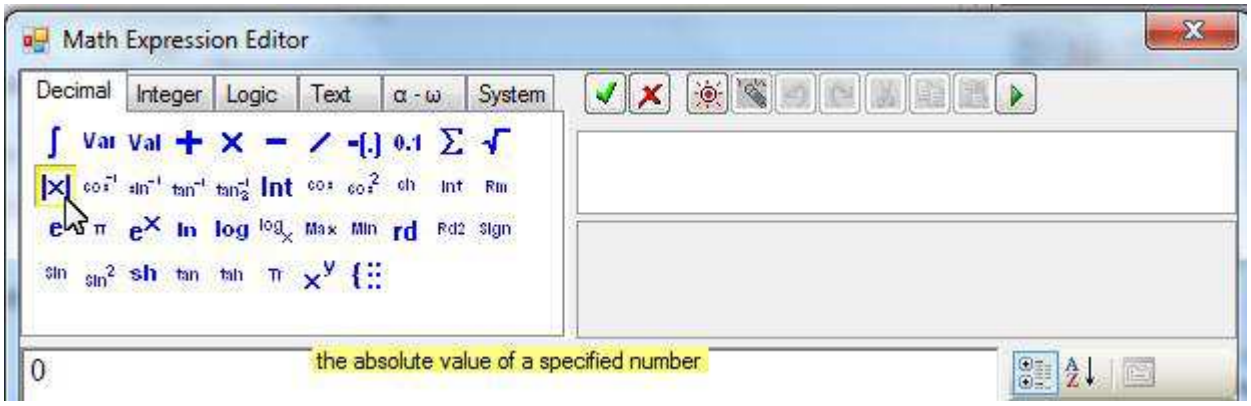

Variable Name
Name:
angle
OK

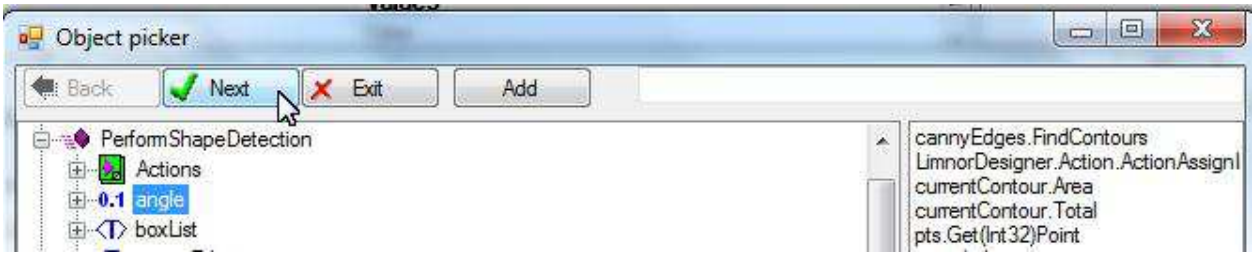


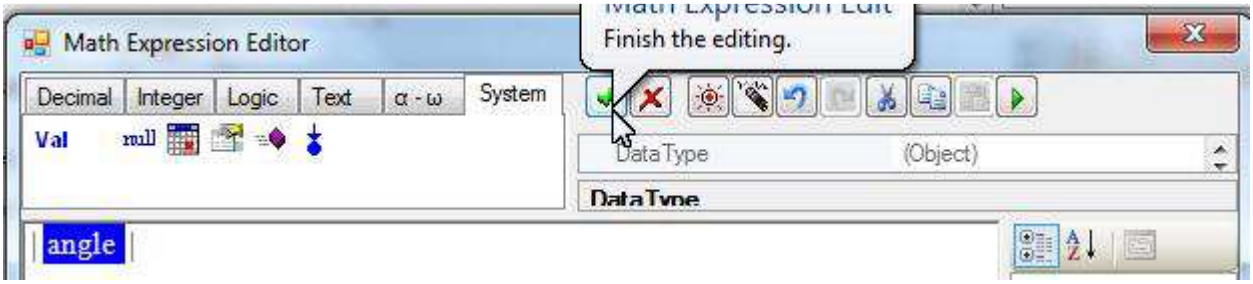
Make the angle into an absolute value:



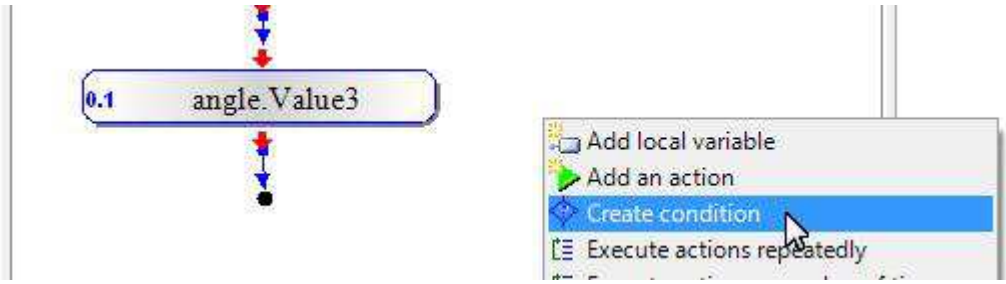




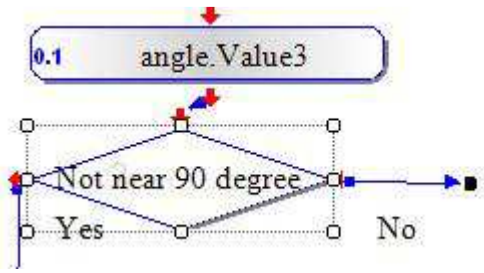




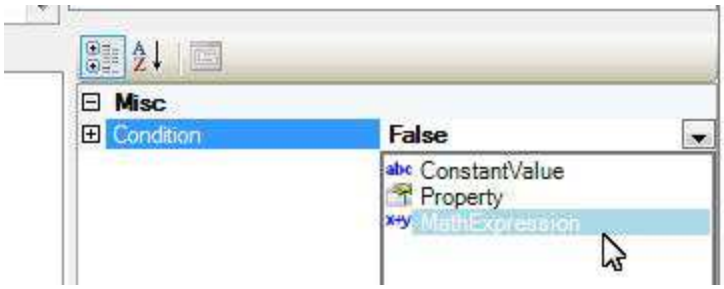
Add a condition to check whether the angle is away from 90 degree too much, that is, the angle is smaller than 80 or larger 100:

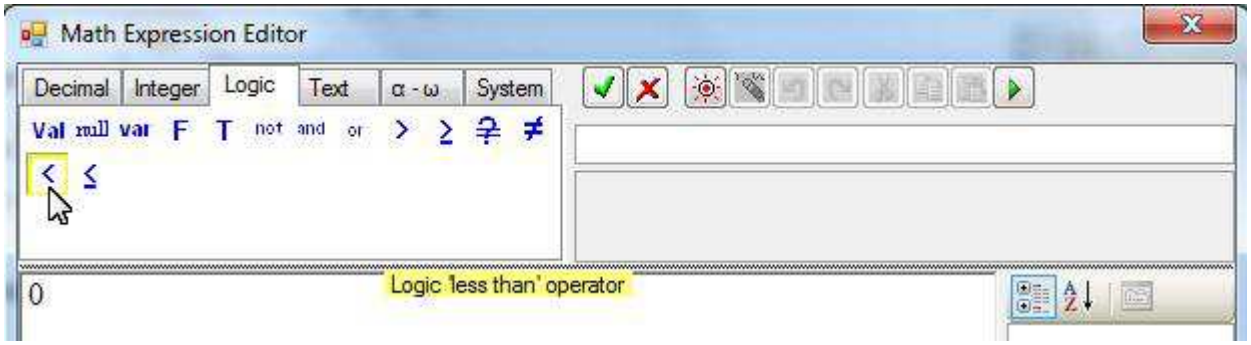


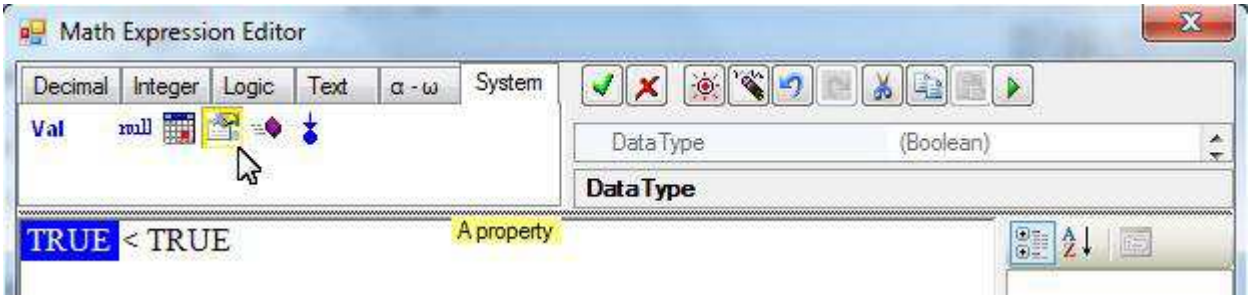
Rename the condition “Not near 90 degree”:

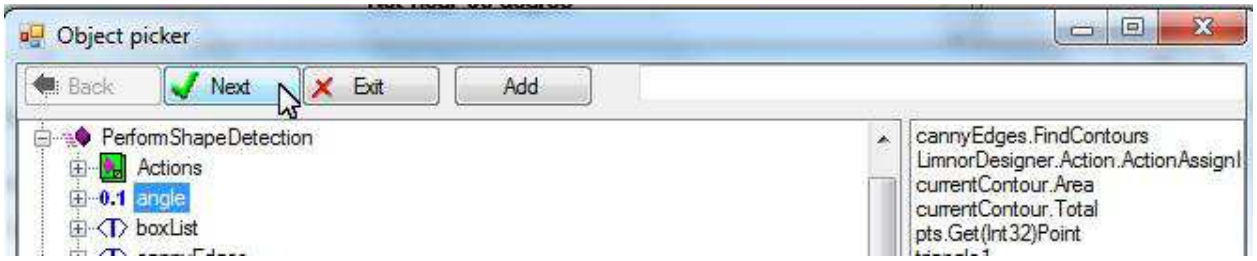


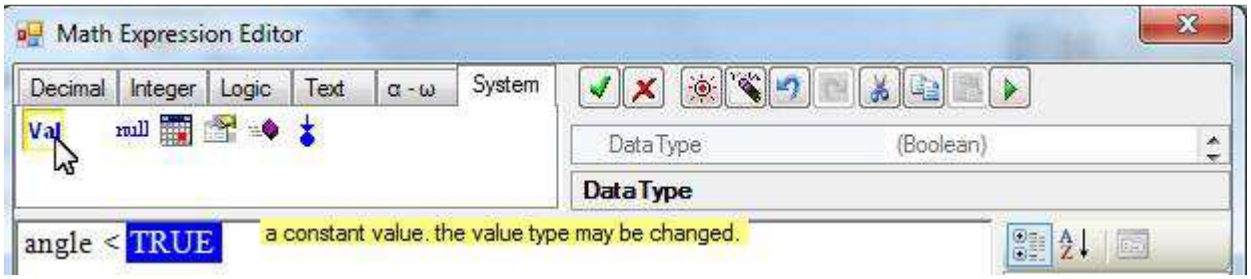
Set its condition:

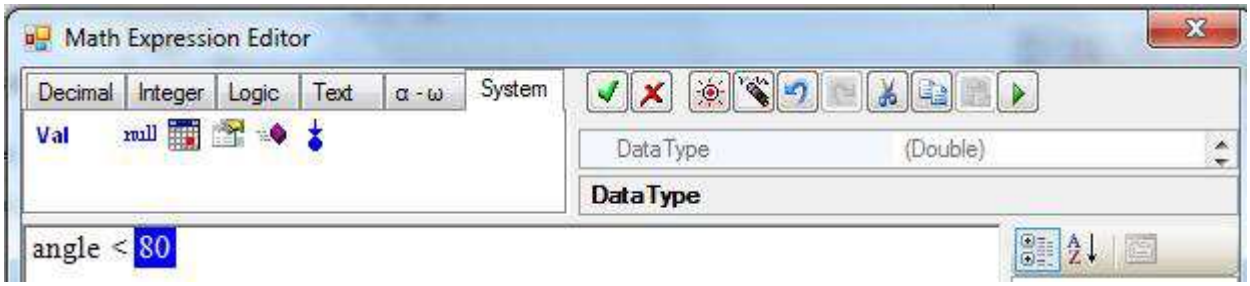

Misc
Condition
False
ConstantValue
Property
MathExpression

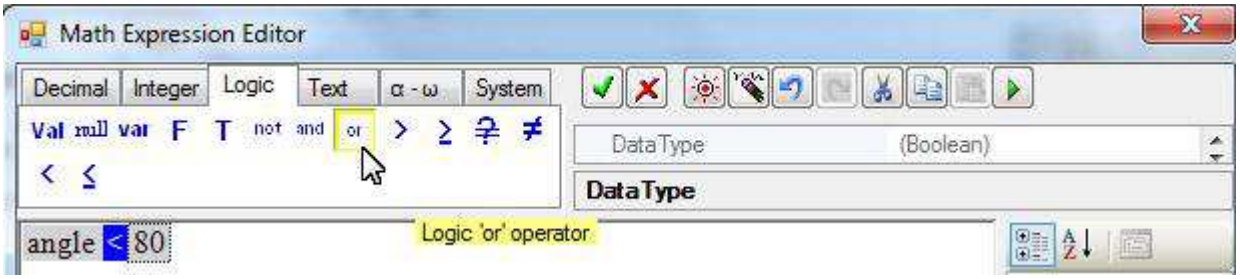

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
0
Logic 'less than' operator

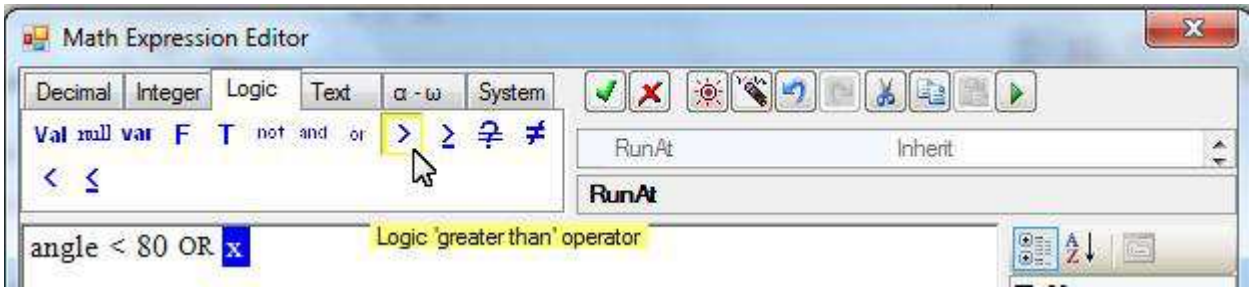

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
TRUE < TRUE
A property


Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
0.1 angle
boxList
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point

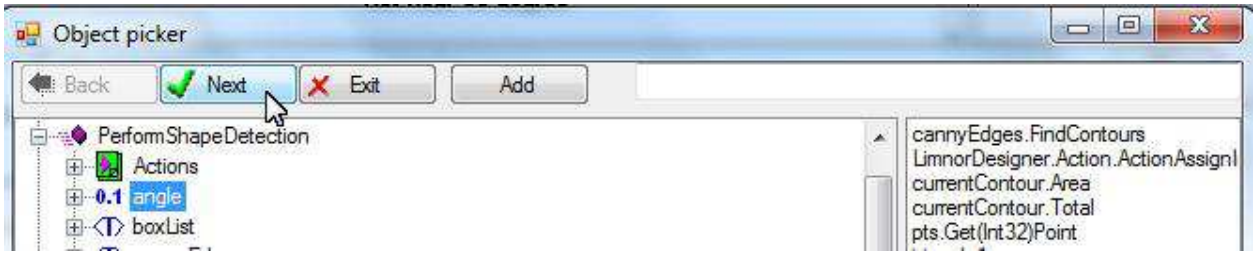

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
angle < TRUE
a constant value. the value type may be changed.

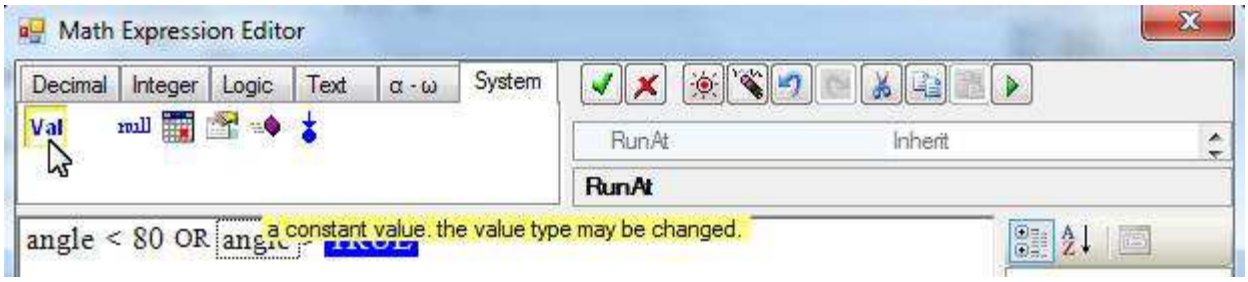

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Double)
DataType
angle < 80


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
DataType
(Boolean)
DataType
angle < 80
Logic 'or' operator

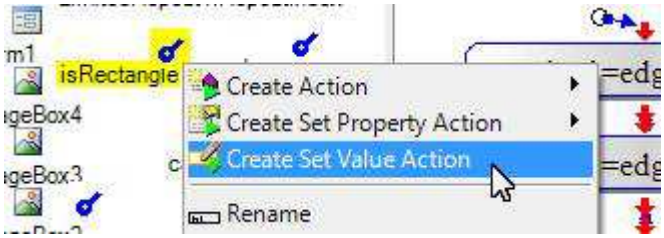

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
RunAt
Inherit
RunAt
angle < 80 OR x
Logic 'greater than' operator

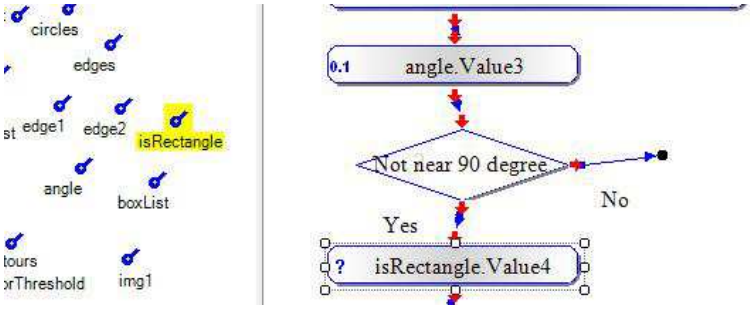

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
RunAt
Inherit
RunAt
angle < 80 OR x > TRUE
A property
Misc


Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
angle
boxList
cannyEdges.FindContours
LimnorDesigner.Action.ActionAssignI
currentContour.Area
currentContour.Total
pts.Get(Int32)Point

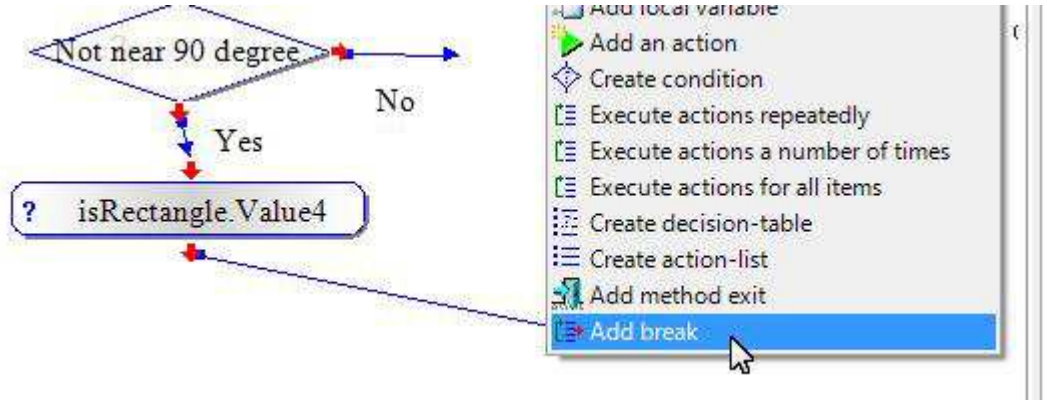


If the angle is away from 90 degree too much then set isRectangle to false and break the loop action.

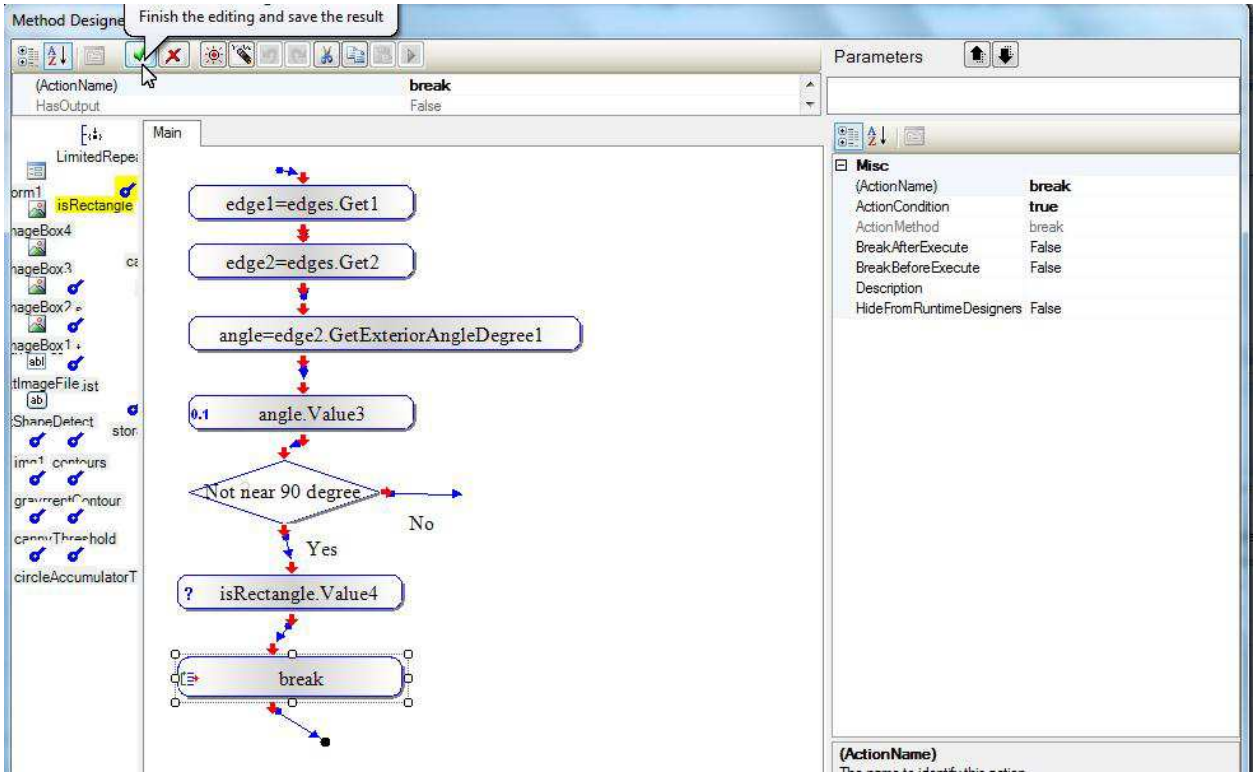




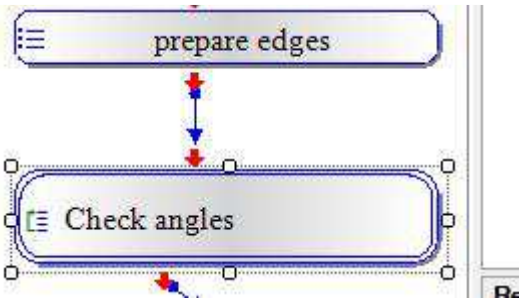
Break the loop action:



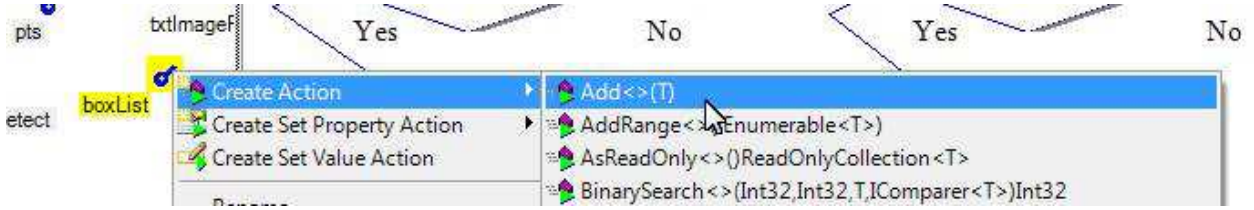
These are the angle checking actions:



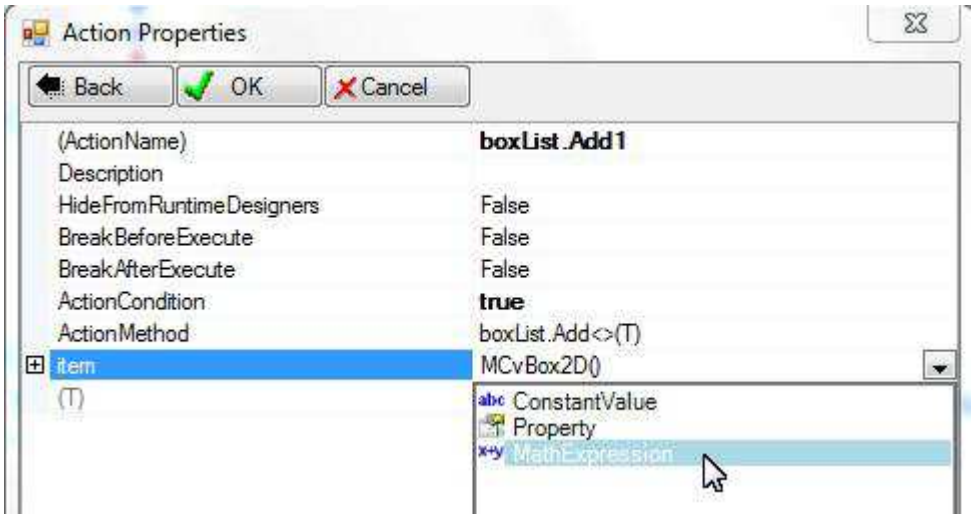
Change the loop action name to “check angles”:

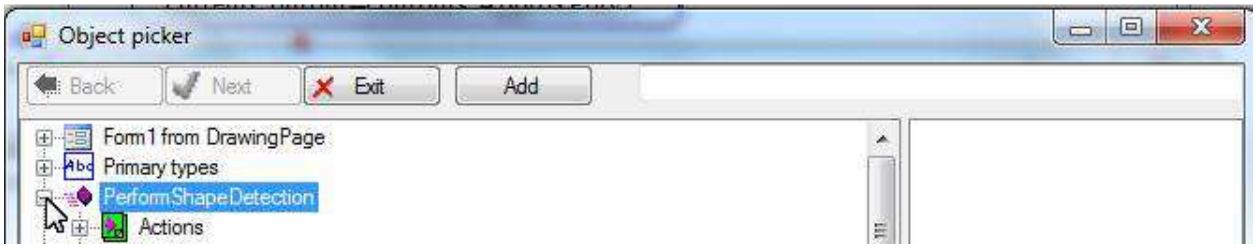


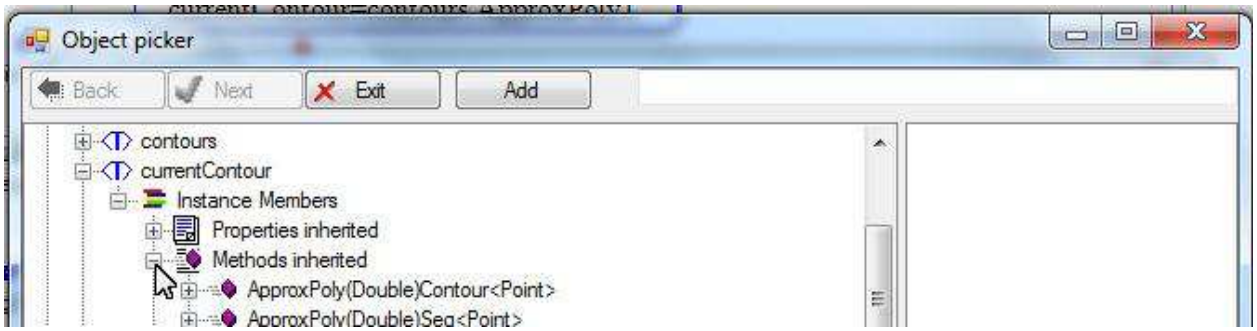
If “Check angles” does not set isRectangle to false then add the contour to the rectangle list:

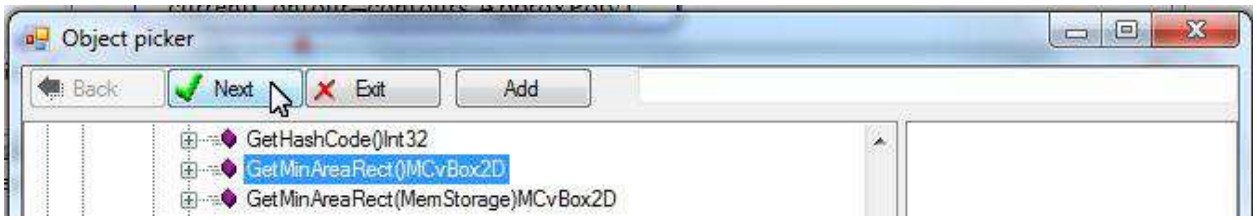


Use an expression to create a rectangle from the contour:

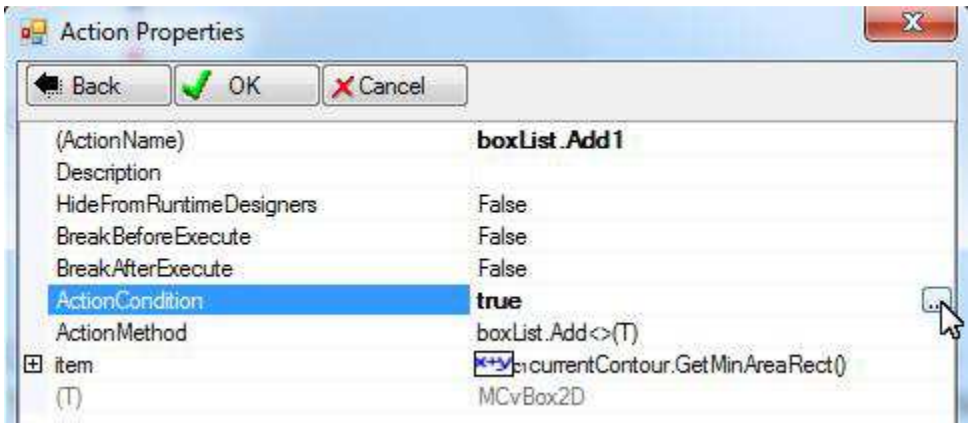


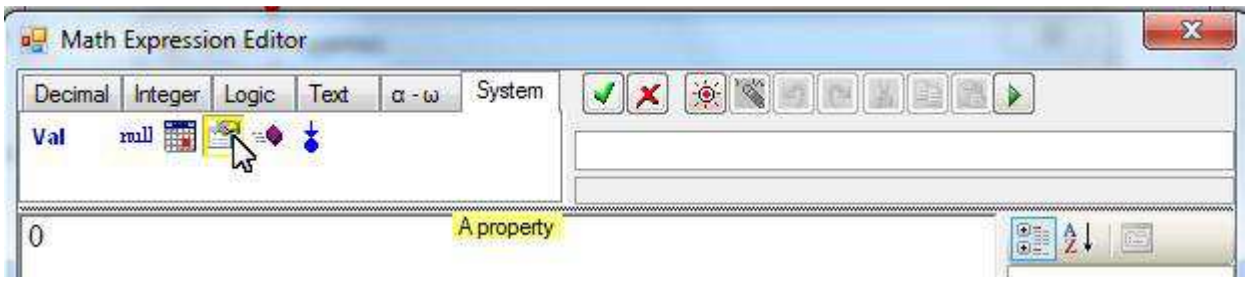


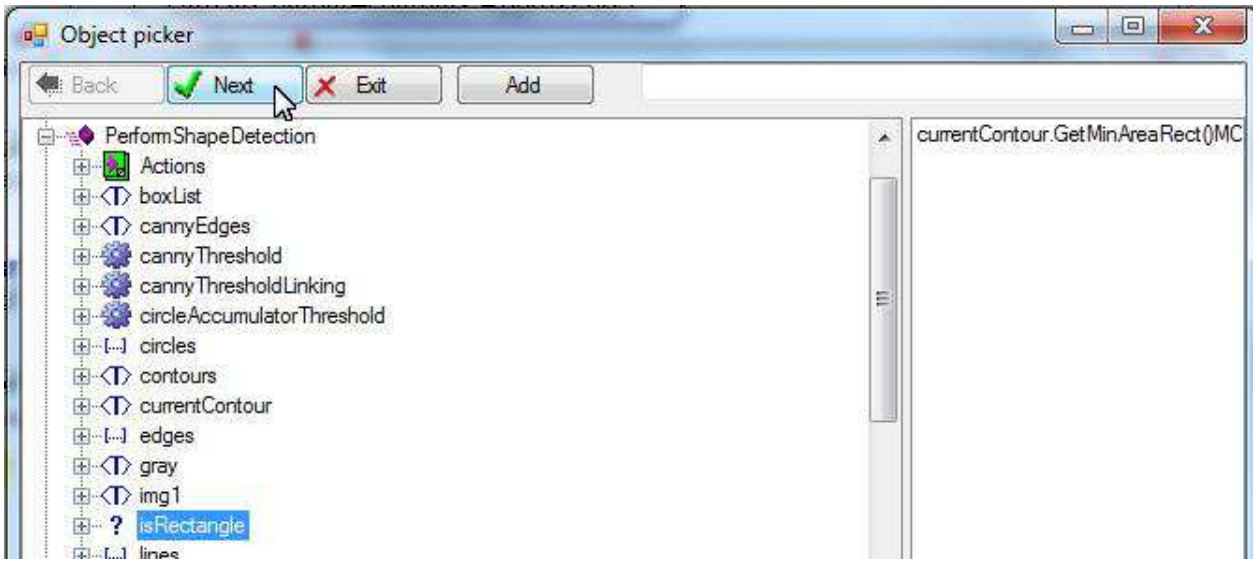


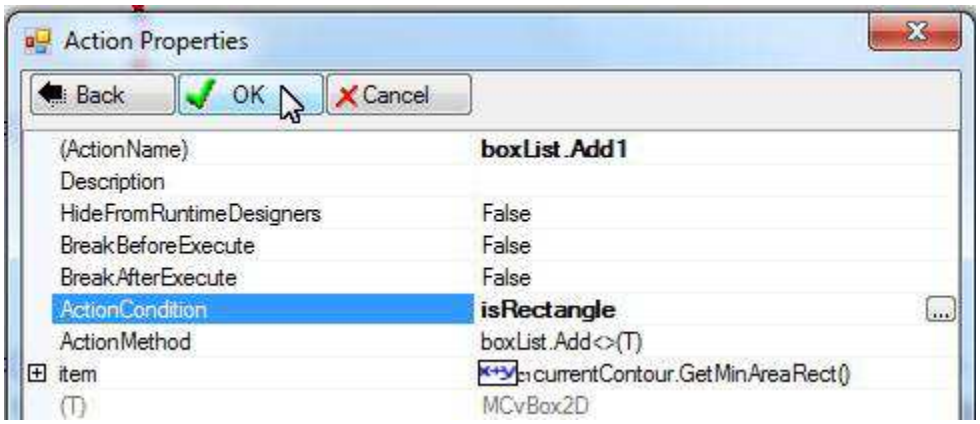


Set the ActionCondition to isRectangle:

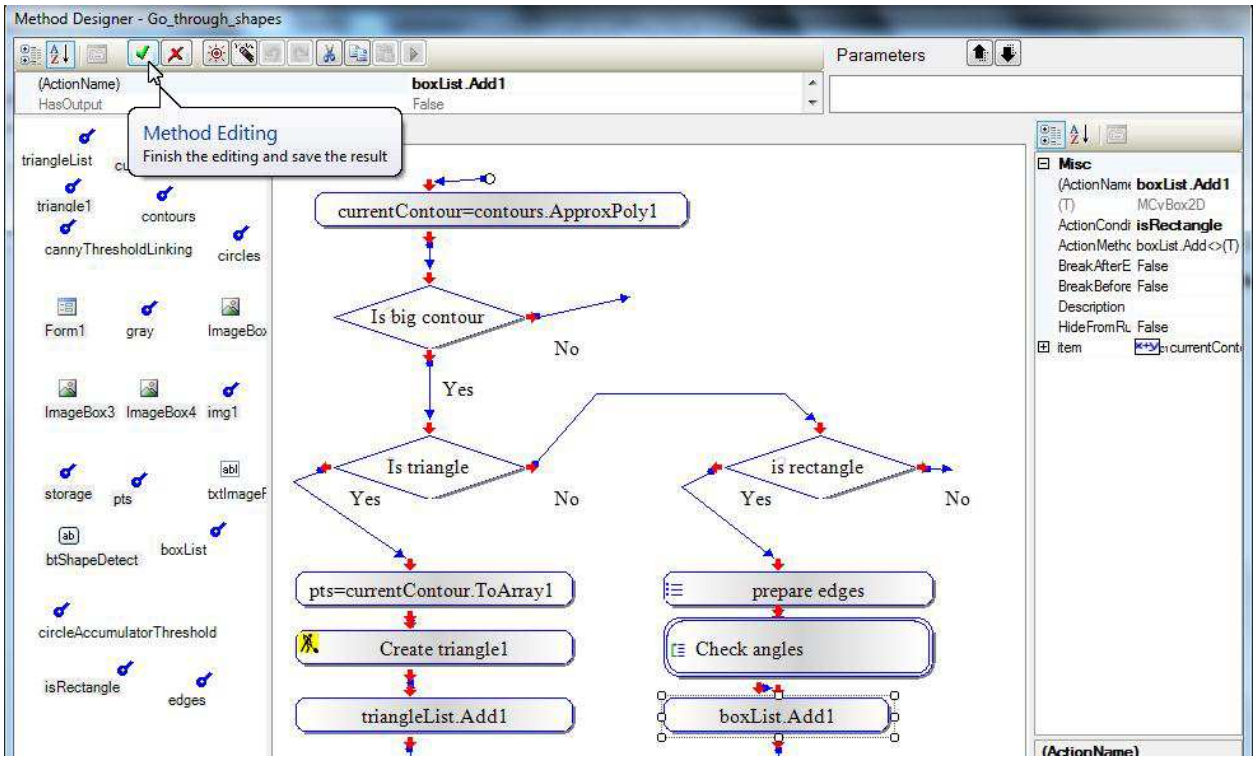






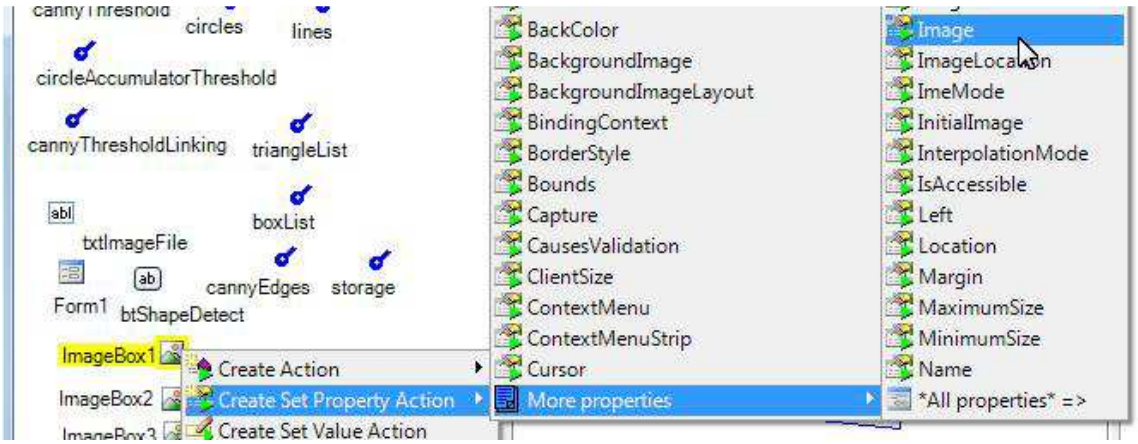


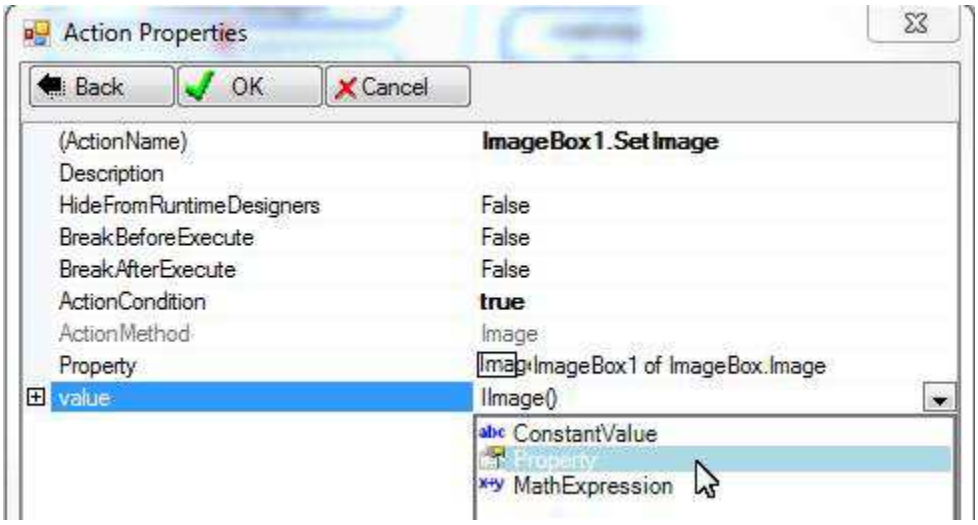
This completes the triangle and rectangle detection:

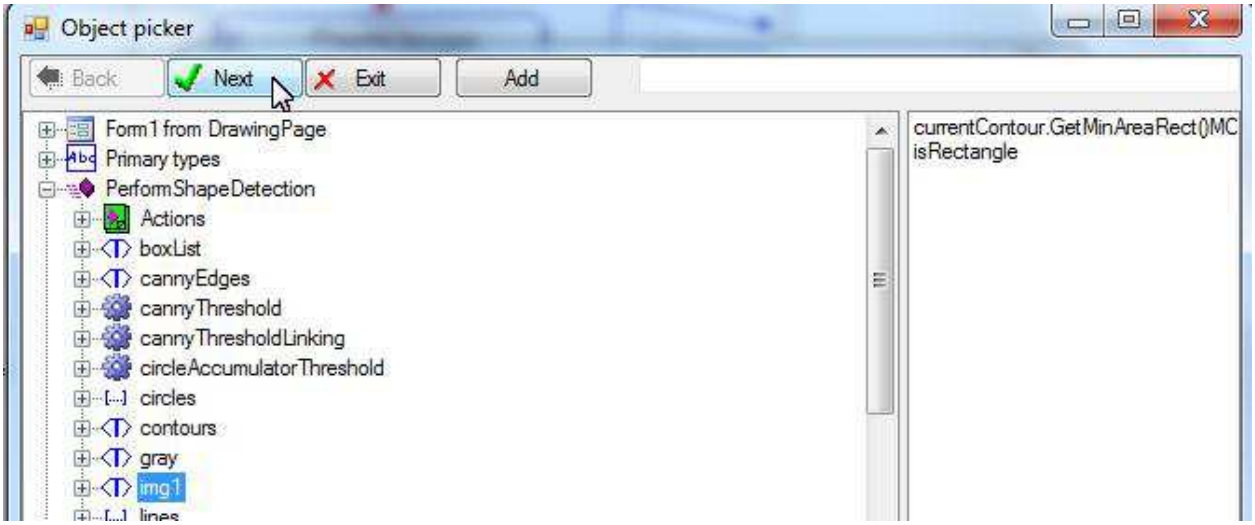


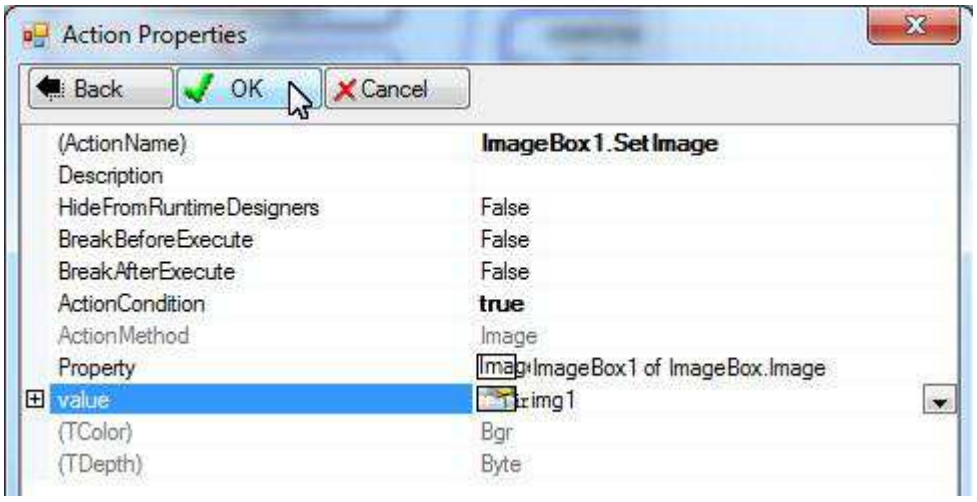
Show original image

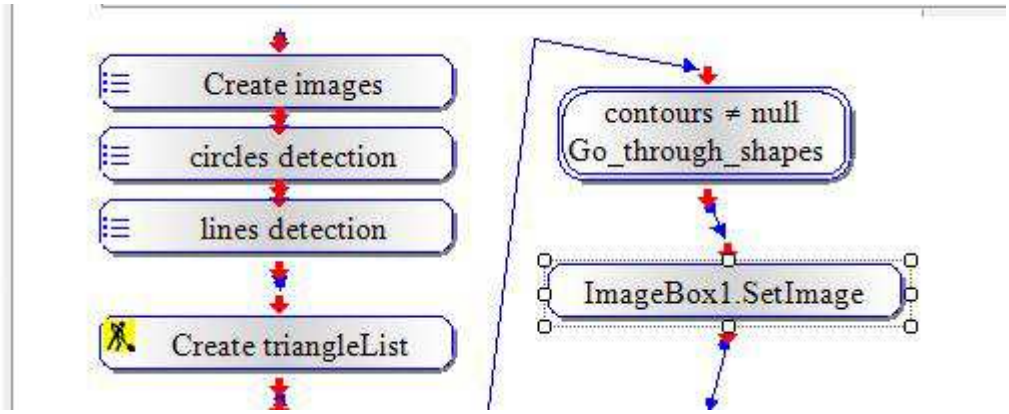
Display the original image on the first image box.




Action Properties
Back
OK
Cancel
(ActionName)
ImageBox1.SetImage
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
Image
Property
ImageBox1 of ImageBox.Image
value
IImage()
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
gray
img1
currentContour.GetMinAreaRect()MC
isRectangle


Action Properties
Back
OK
Cancel
(ActionName)
ImageBox1.SetImage
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
Image
Property
ImageBox1 of ImageBox.Image
value
img1
(TColor)
Bgr
(TDepth)
Byte

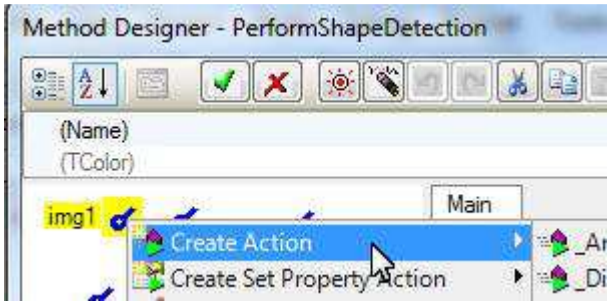


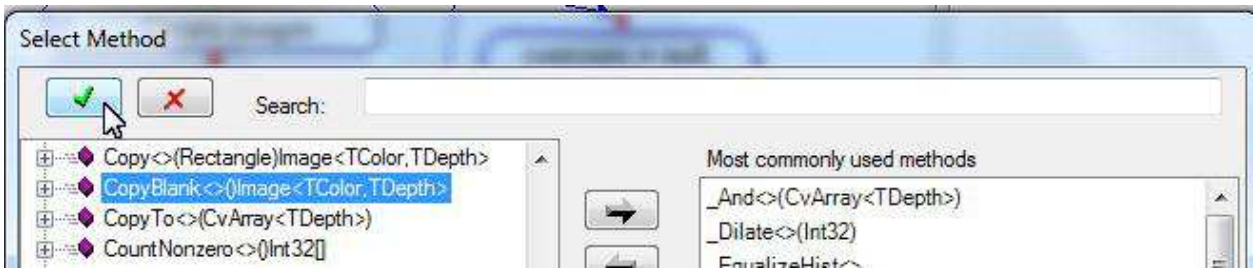
Display triangles and rectangles

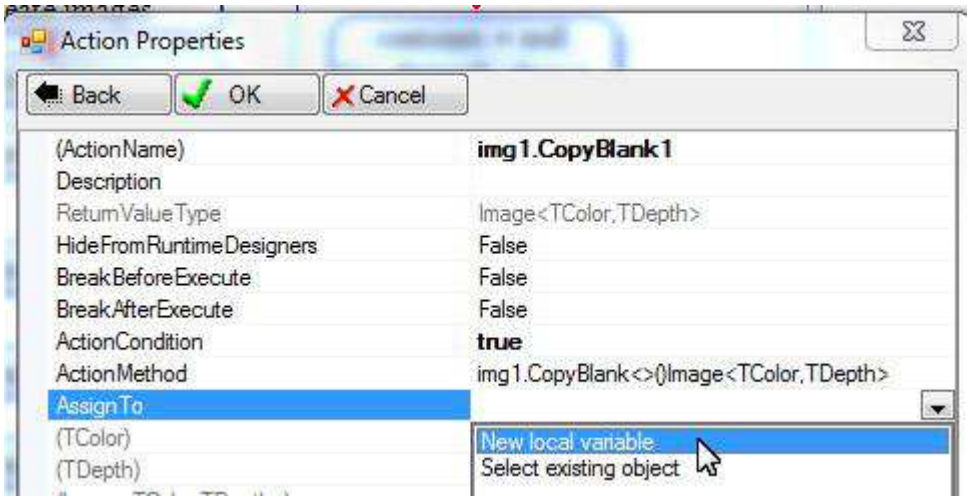
Triangles and rectangles are displayed on the second image box.

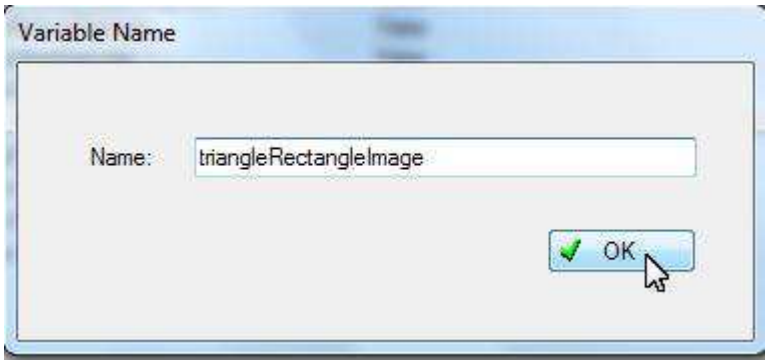
Create a blank image. Draw all the triangles and rectangles to the blank image. Assign the image to the second image box.

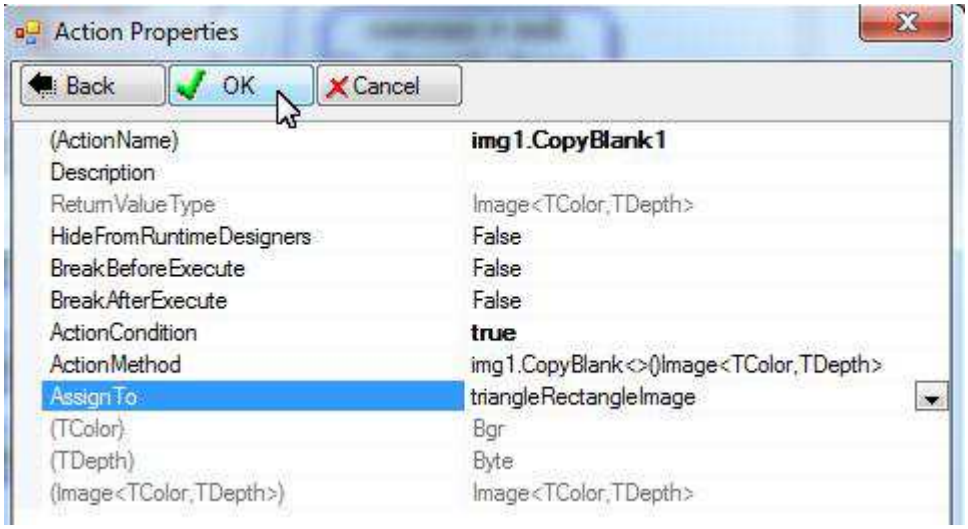
Create a blank image





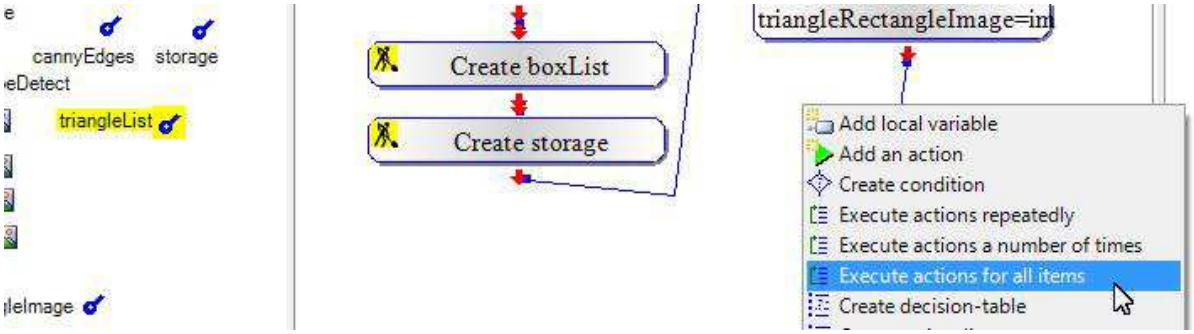


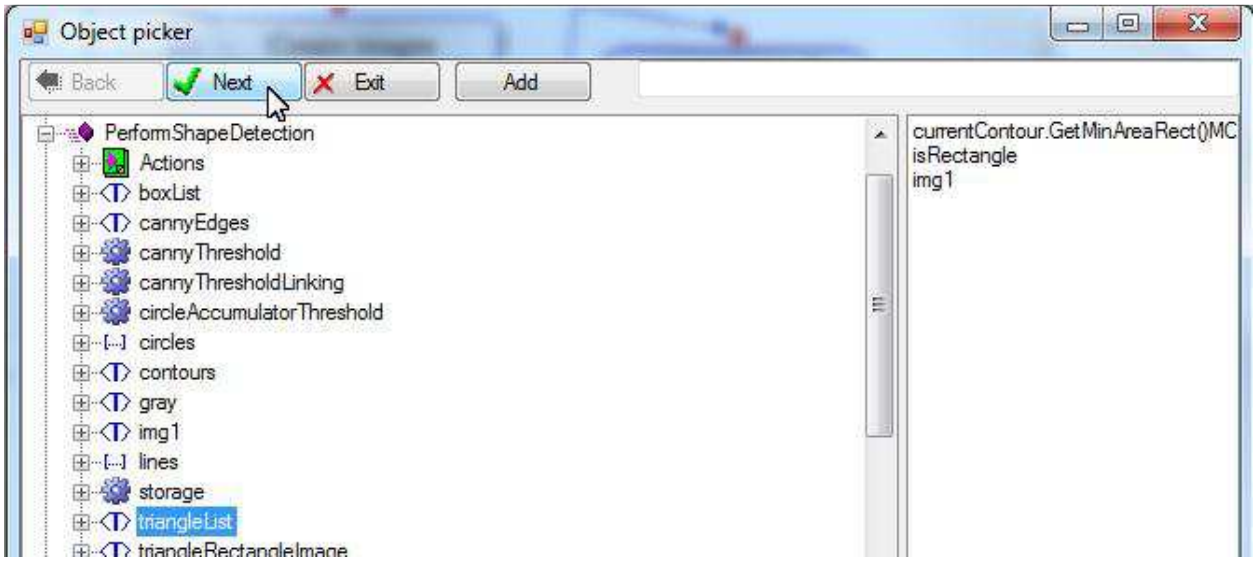


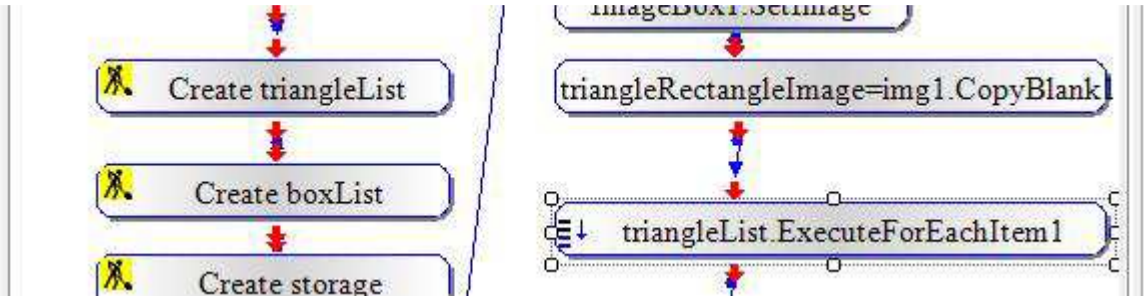


Go through all triangles

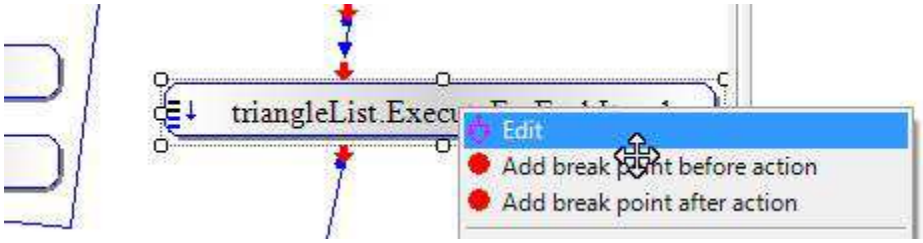
Create a loop action to go through all triangles.



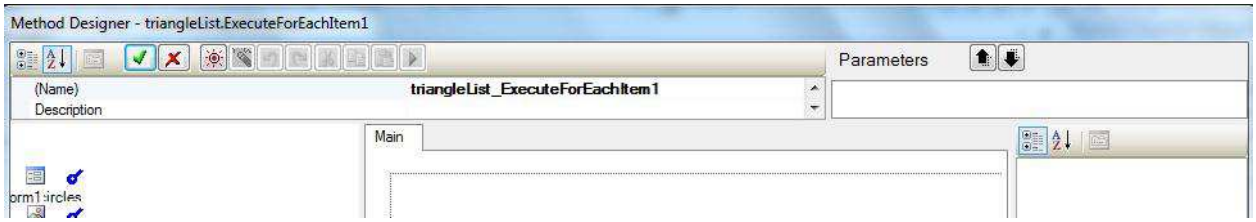




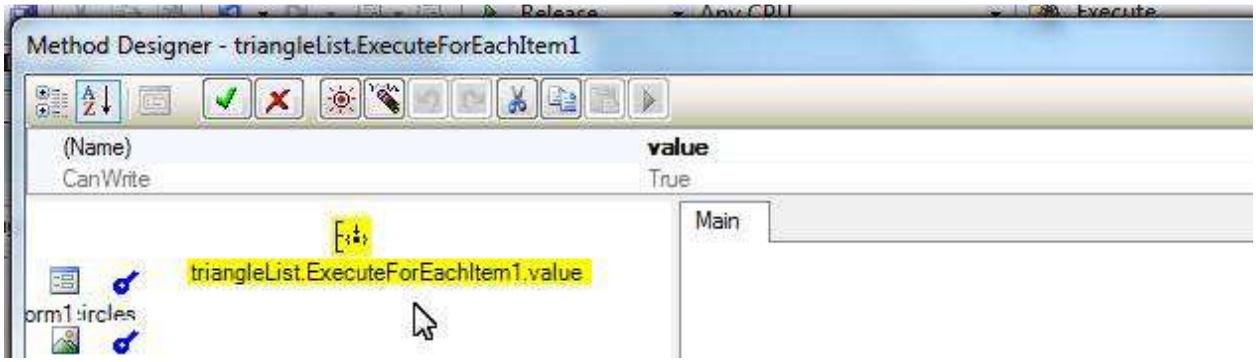
To show each triangle, right-click the loop action, select “Edit”:



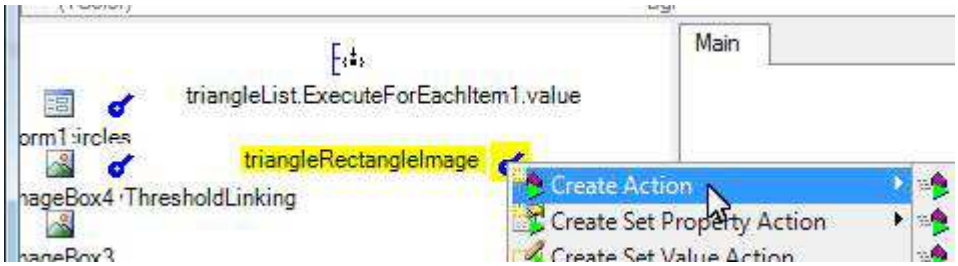
A new Method Editor appears.

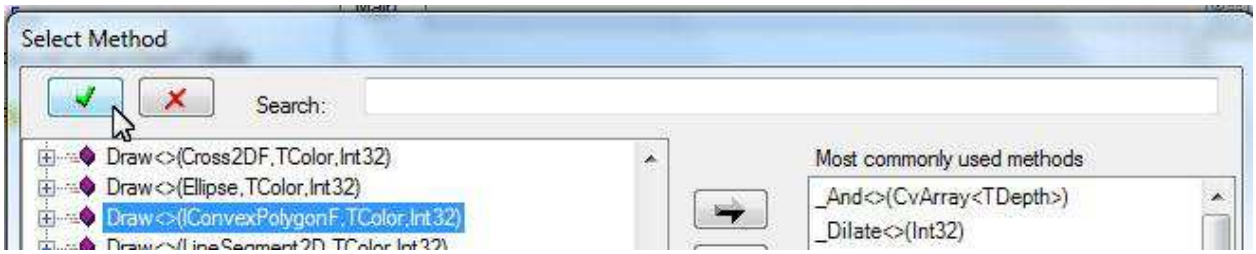


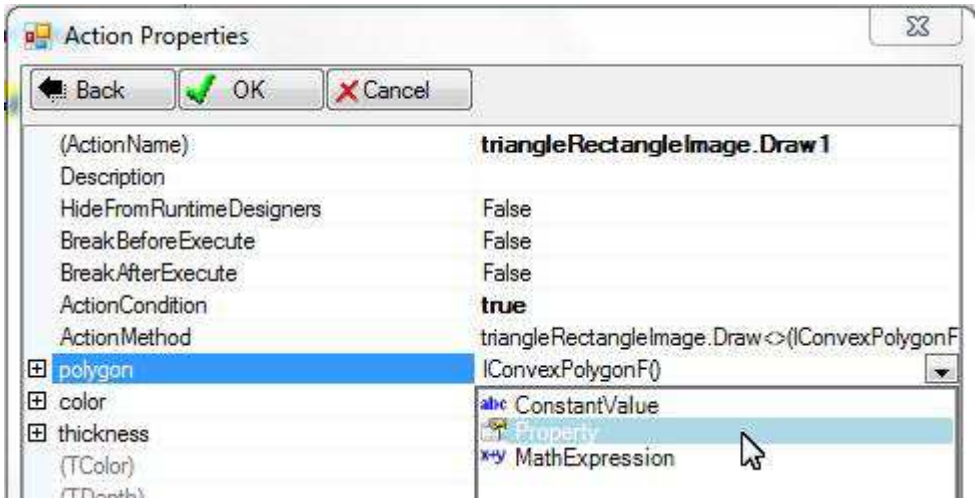
Note the “value” icon. It represents the triangle to be used. We’ll draw it to the blank image.

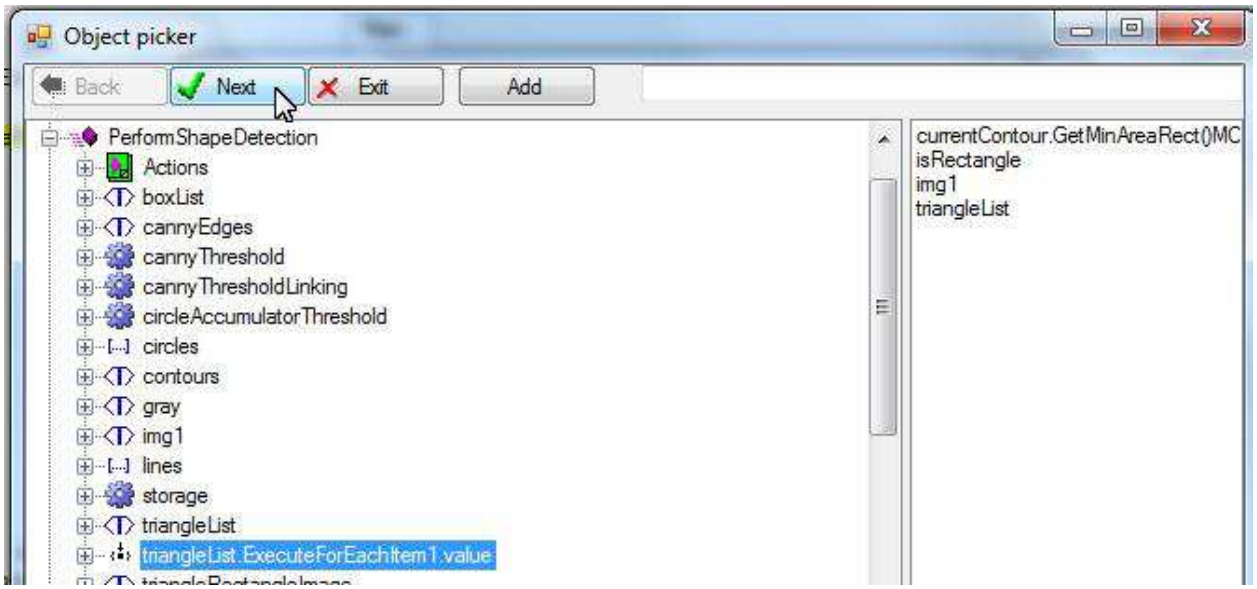


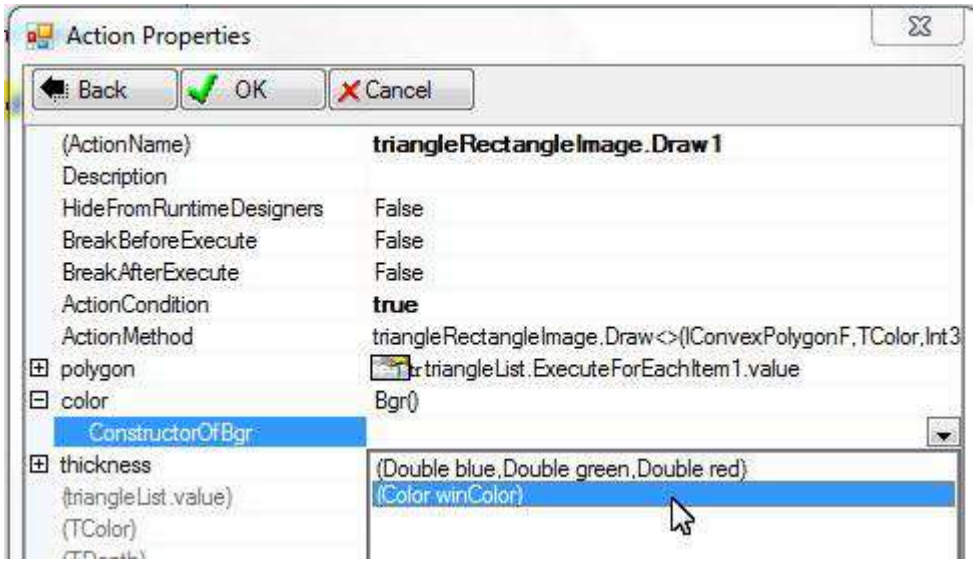
Right-click the blank image named “`triangleRectangleImage`”; select its Draw method.

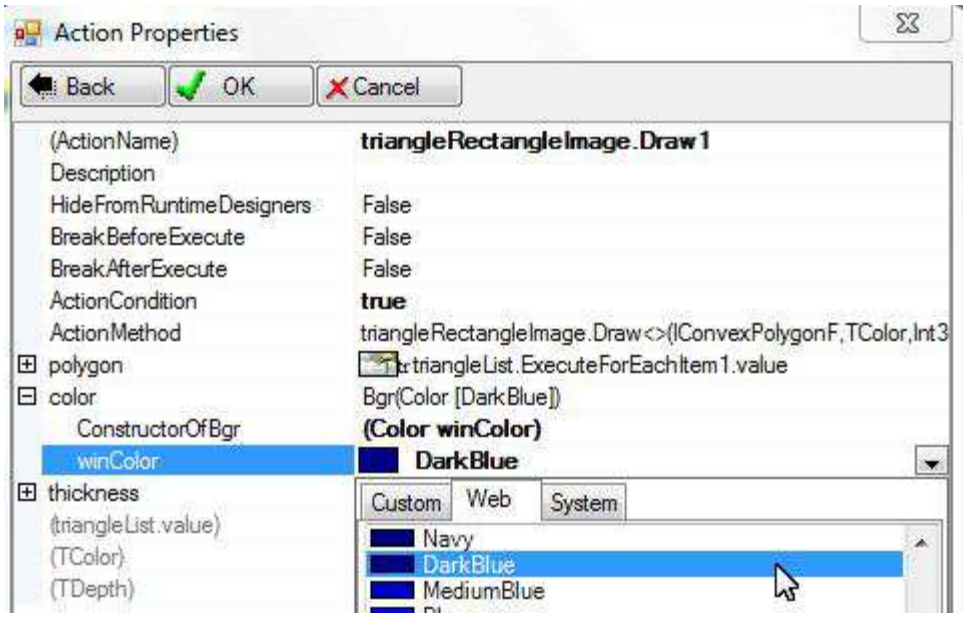


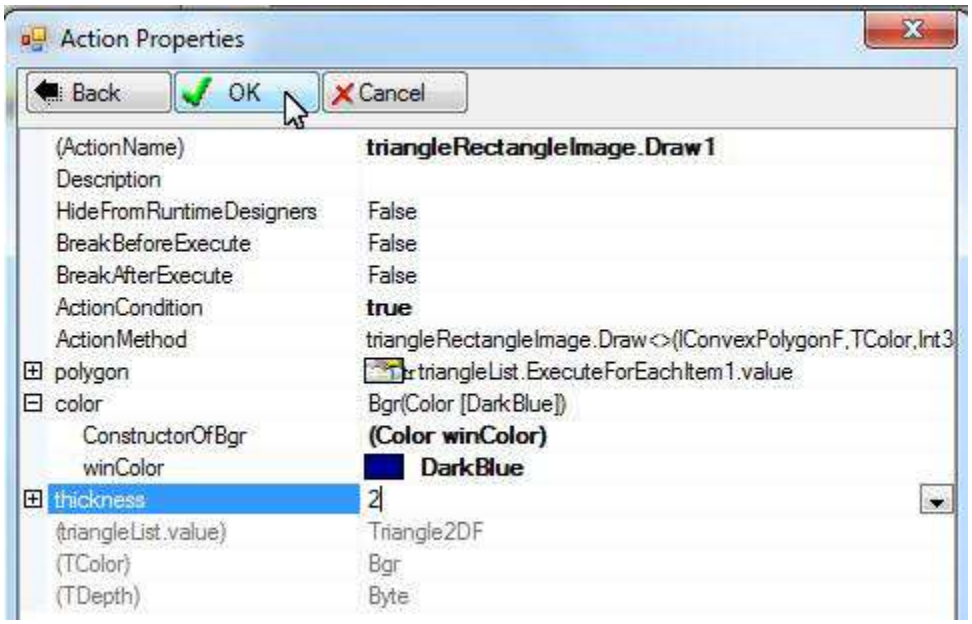


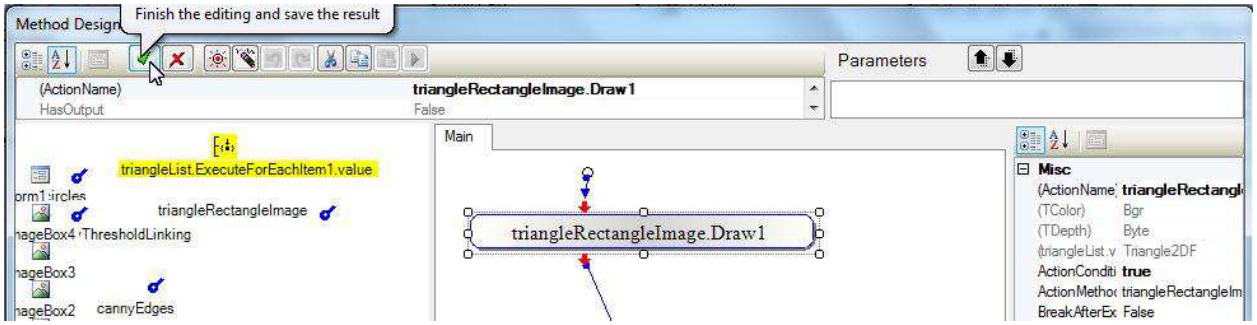

Action Properties
Back
OK
Cancel
(ActionName)
triangleRectangleImage.Draw1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleRectangleImage.Draw<>(IConvexPolygonF
polygon
IConvexPolygonF()
color
thickness
(TColor)
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circles
contours
gray
img1
lines
storage
triangleList
triangleList.ExecuteForEachItem1.value
currentContour.GetMinAreaRect()MC
isRectangle
img1
triangleList


Action Properties
Back
OK
Cancel
(ActionName)
triangleRectangleImage.Draw1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleRectangleImage.Draw<>(IConvexPolygonF,TColor,Int3
polygon
triangleList.ExecuteForEachItem1.value
color
Bgr()
ConstructorOfBgr
thickness
(triangleList.value)
(TColor)
(Double blue,Double green,Double red)
(Color winColor)

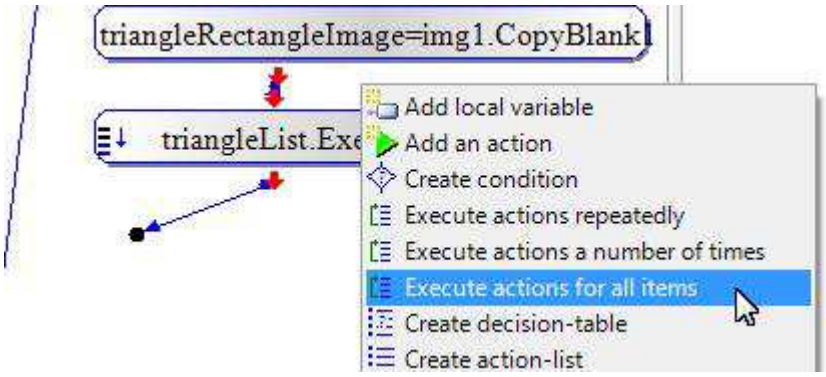


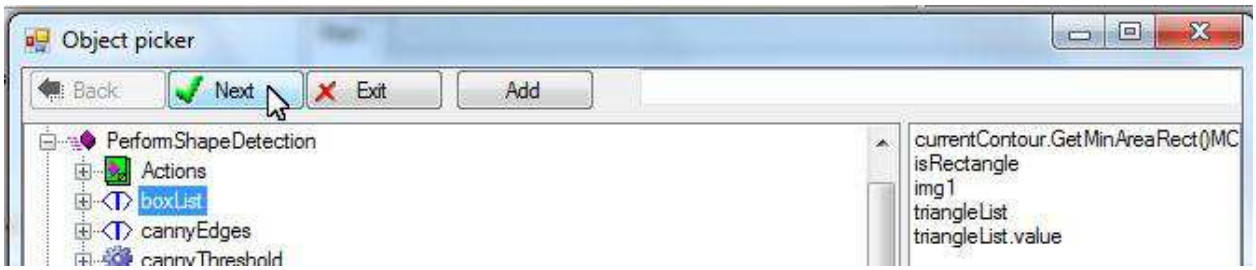


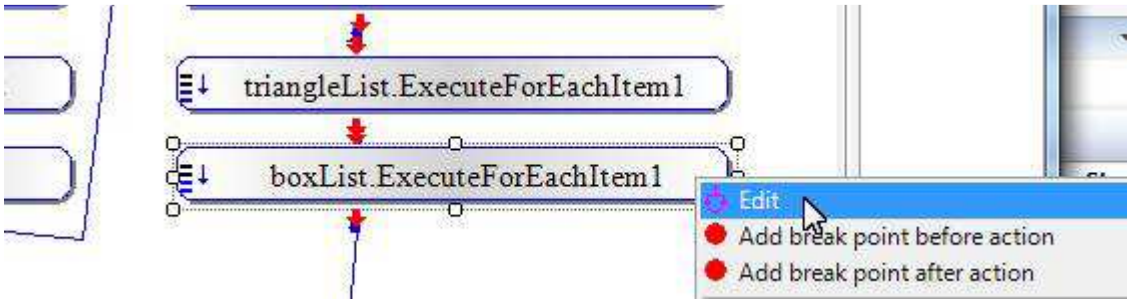


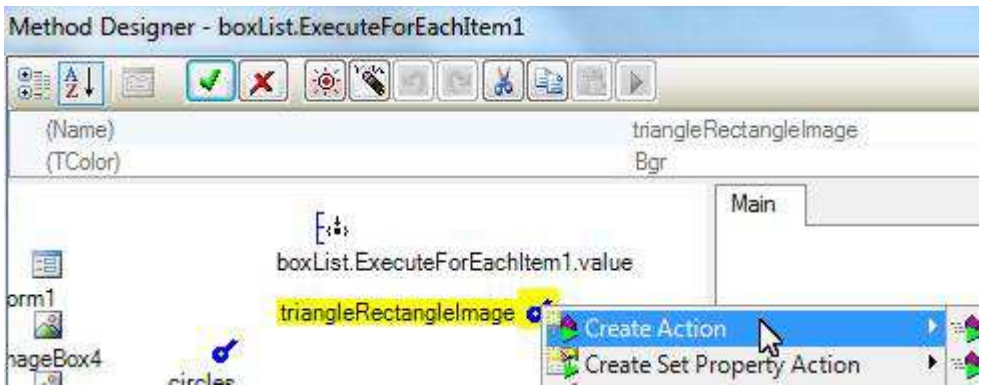
Go through all rectangles

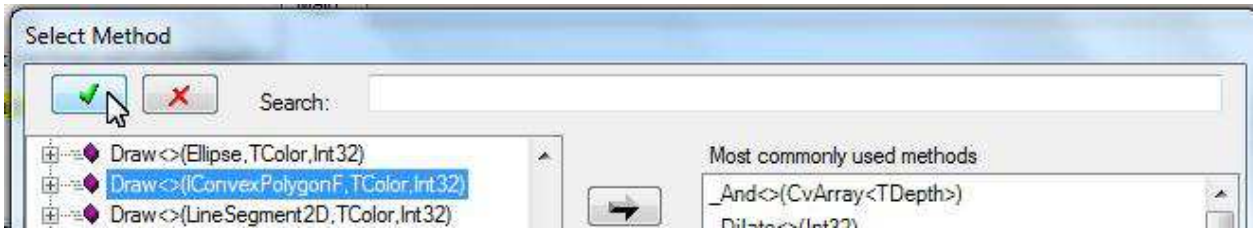
Create a loop action to go through all rectangles.

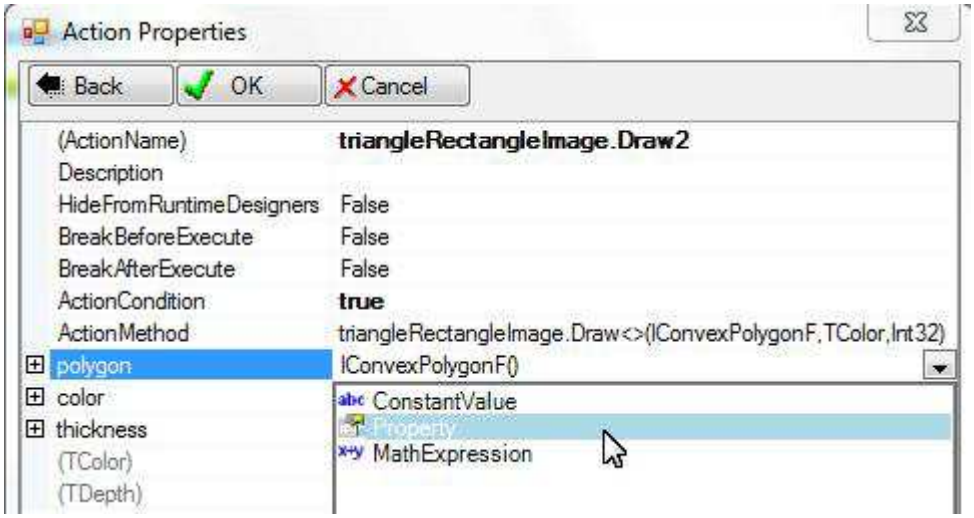

triangleRectangleImage=img1.CopyBlank
triangleList.Exe
Add local variable
Add an action
Create condition
Execute actions repeatedly
Execute actions a number of times
Execute actions for all items
Create decision-table
Create action-list

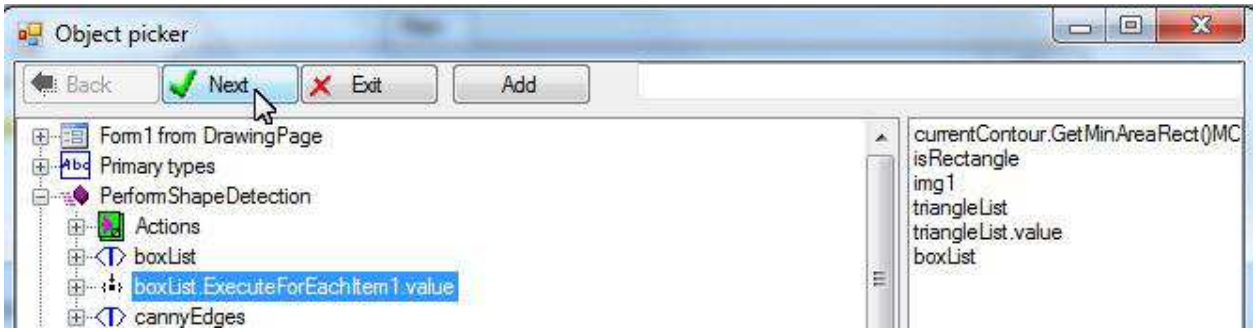

Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
currentContour.GetMinAreaRect()MC
isRectangle
img1
triangleList
triangleList.value

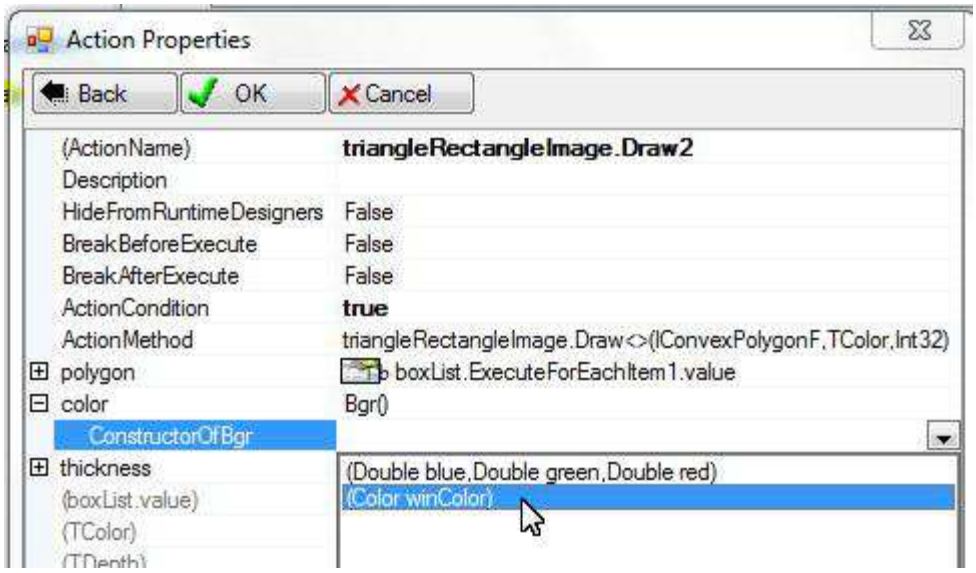

triangleList.ExecuteForEachItem1
boxList.ExecuteForEachItem1
Edit
Add break point before action
Add break point after action

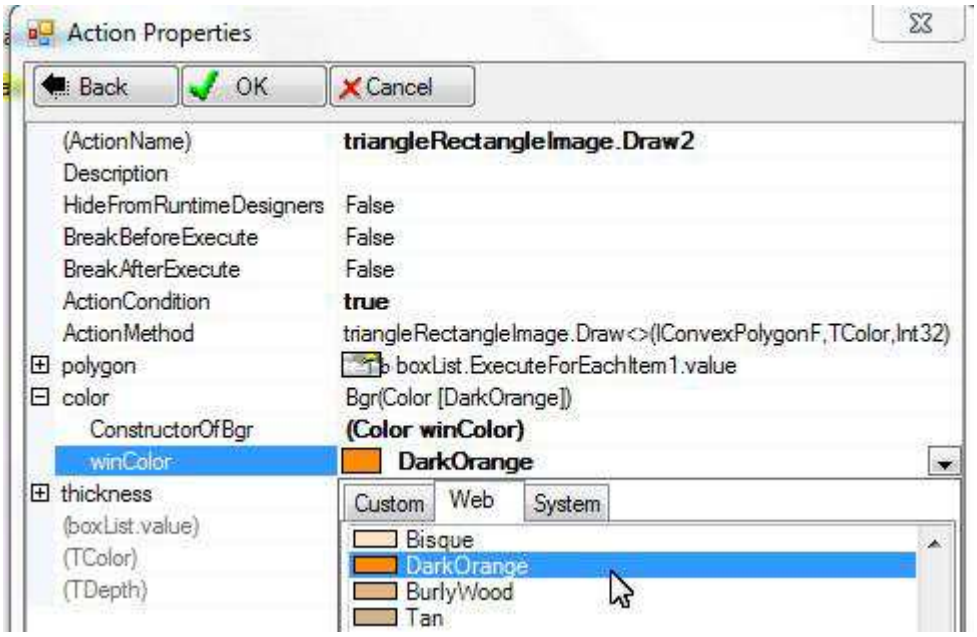

Method Designer - boxList.ExecuteForEachItem1
(Name)
triangleRectangleImage
(TColor)
Bgr
Main
boxList.ExecuteForEachItem1.value
triangleRectangleImage
Create Action
Create Set Property Action

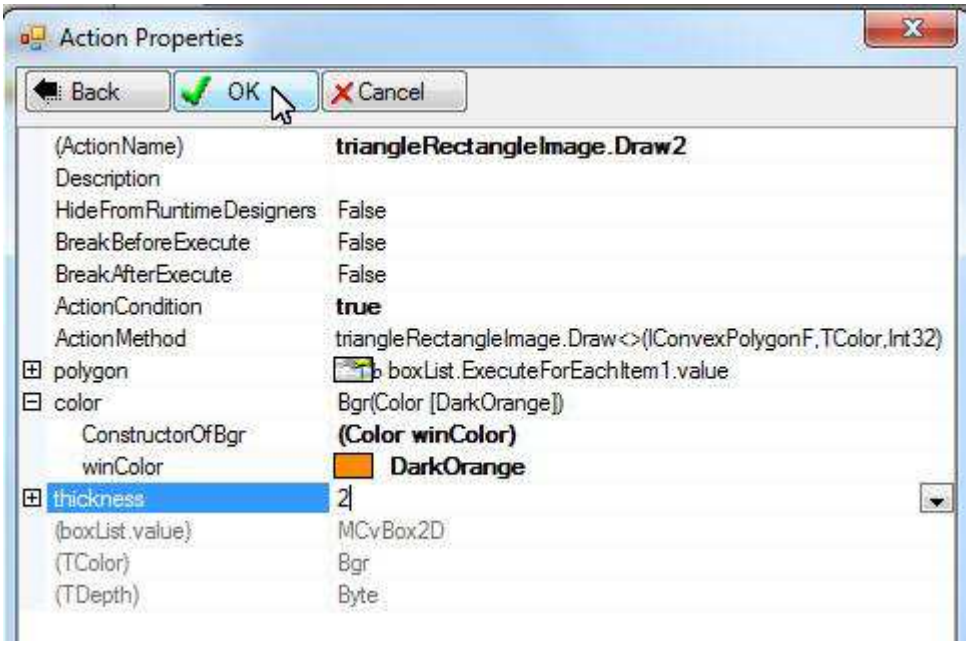

Select Method
Search:
Draw<>(Ellipse,TColor,Int32)
Draw<>(IConvexPolygonF,TColor,Int32)
Draw<>(LineSegment2D,TColor,Int32)
Most commonly used methods
_And<>(CvArray<TDepth>)

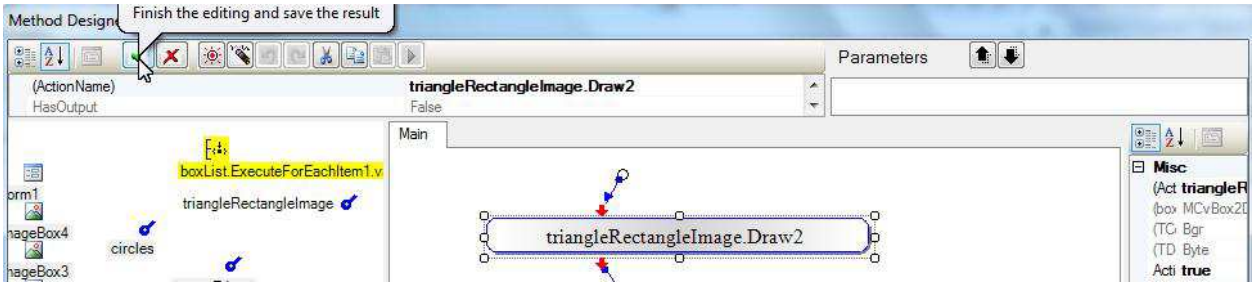

Action Properties
Back
OK
Cancel
(ActionName)
triangleRectangleImage.Draw2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleRectangleImage.Draw<>(IConvexPolygonF,TColor,Int32)
polygon
IConvexPolygonF()
color
thickness
(TColor)
(TDepth)
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
PerformShapeDetection
Actions
boxList
boxList.ExecuteForEachItem1.value
cannyEdges
currentContour.GetMinAreaRect()MC
isRectangle
img1
triangleList
triangleList.value
boxList


Action Properties
Back
OK
Cancel
(ActionName)
triangleRectangleImage.Draw2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleRectangleImage.Draw<>(IConvexPolygonF,TColor,Int32)
polygon
boxList.ExecuteForEachItem1.value
color
Bgr()
ConstructorOfBgr
thickness
(boxList.value)
(TColor)
(TDepth)
(Double blue,Double green,Double red)
(Color winColor)

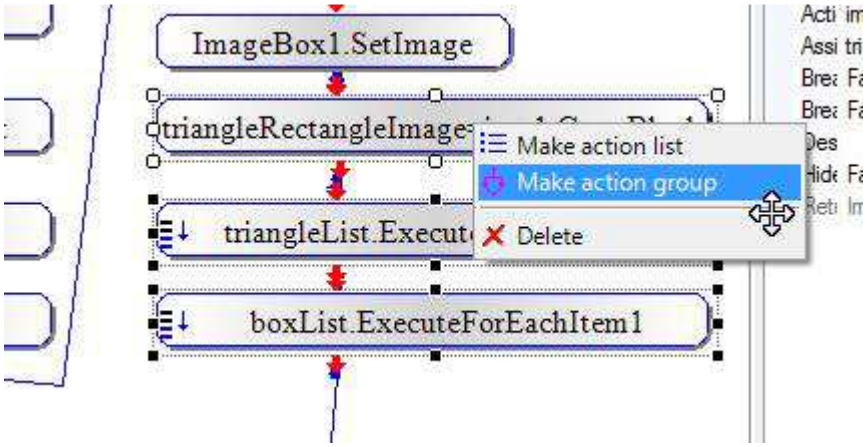

Action Properties
Back
OK
Cancel
(ActionName)
triangleRectangleImage.Draw2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleRectangleImage.Draw<>(IConvexPolygonF,TColor,Int32)
polygon
boxList.ExecuteForEachItem1.value
color
Bgr(Color [DarkOrange])
ConstructorOfBgr
(Color winColor)
winColor
DarkOrange
thickness
(boxList.value)
(TColor)
(TDepth)
Custom
Web
System
Bisque
DarkOrange
BurlyWood
Tan


Action Properties
Back
OK
Cancel
(ActionName)
triangleRectangleImage.Draw2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
triangleRectangleImage.Draw<>(IConvexPolygonF,TColor,Int32)
polygon
boxList.ExecuteForEachItem1.value
color
Bgr(Color [DarkOrange])
ConstructorOfBgr
(Color winColor)
winColor
DarkOrange
thickness
2
(boxList.value)
MCvBox2D
(TColor)
Bgr
(TDepth)
Byte

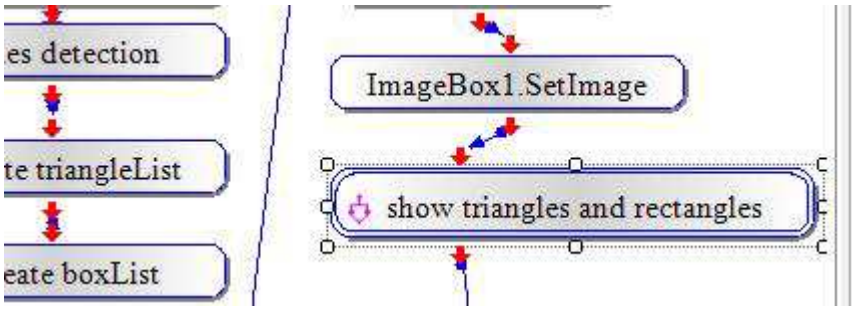

Finish the editing and save the result
Parameters
(ActionName)
triangleRectangleImage.Draw2
HasOutput
False
Main
boxList.ExecuteForEachItem1.v
triangleRectangleImage
circles
triangleRectangleImage.Draw2
Misc
true

Make action group

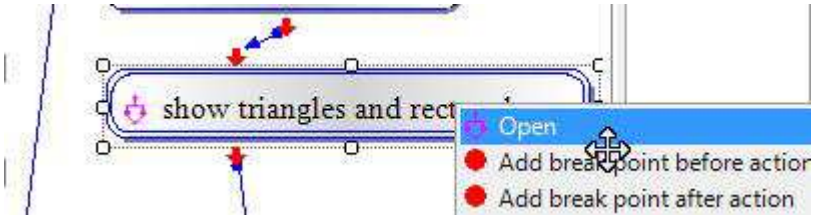
We may group the actions to make room in the editor. This step is optional.



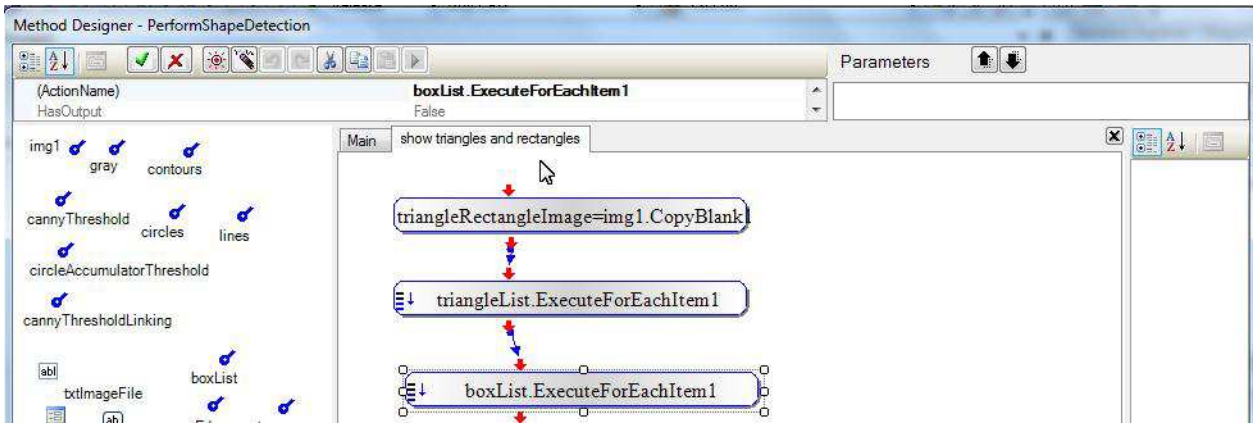
Name the action group “show triangles and rectangles”:



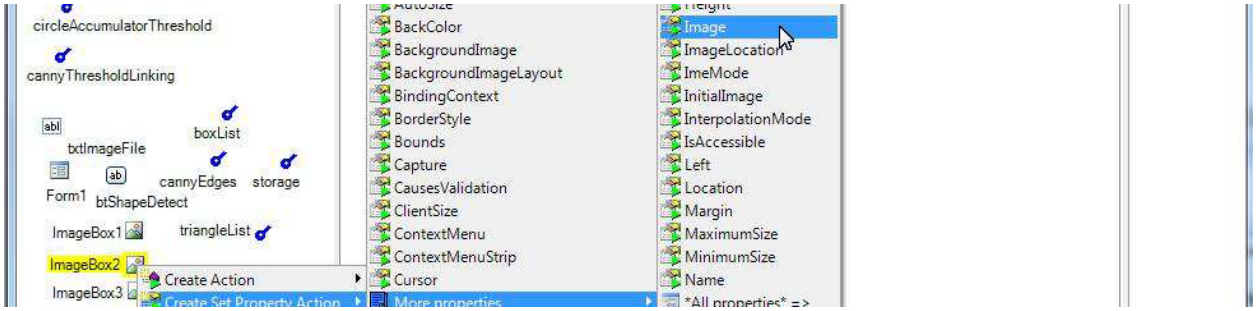
To modify the action group, right-click it and choose “Open”:

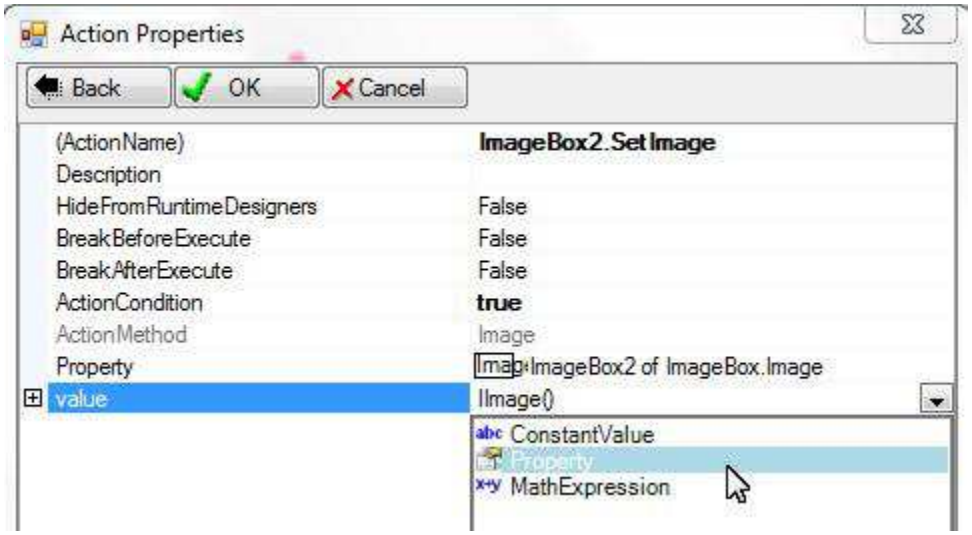


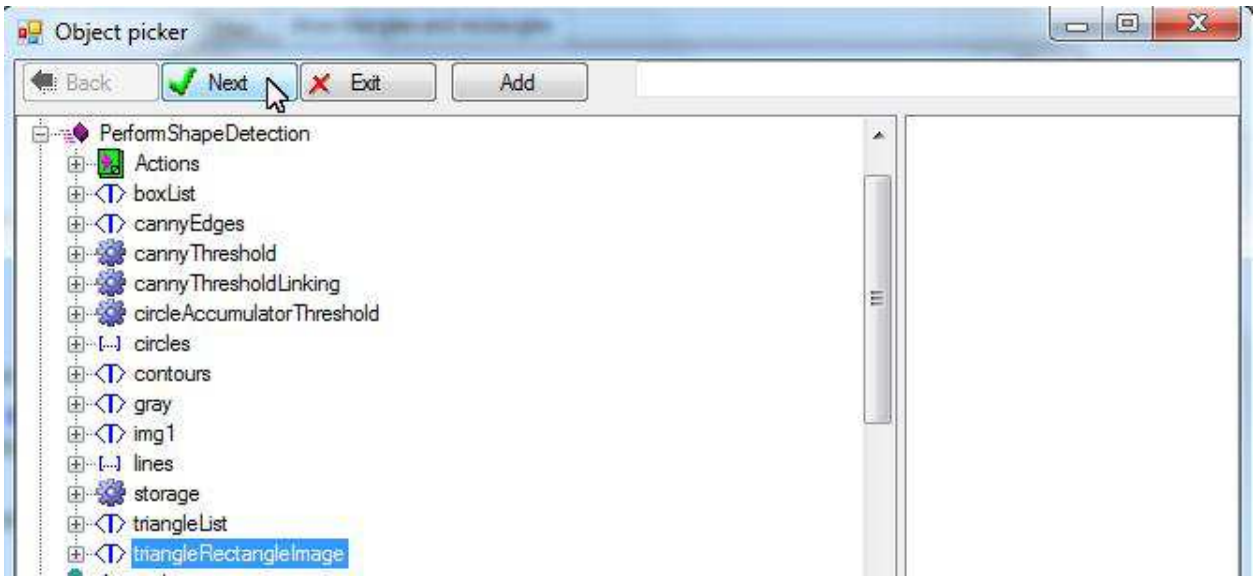
A new tab page is created to show the action group:



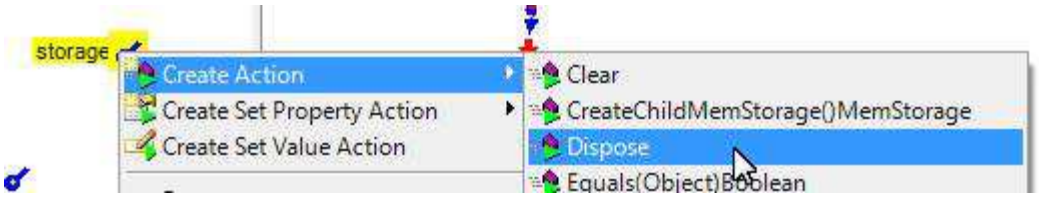
Add an action to display the image on the second image box:



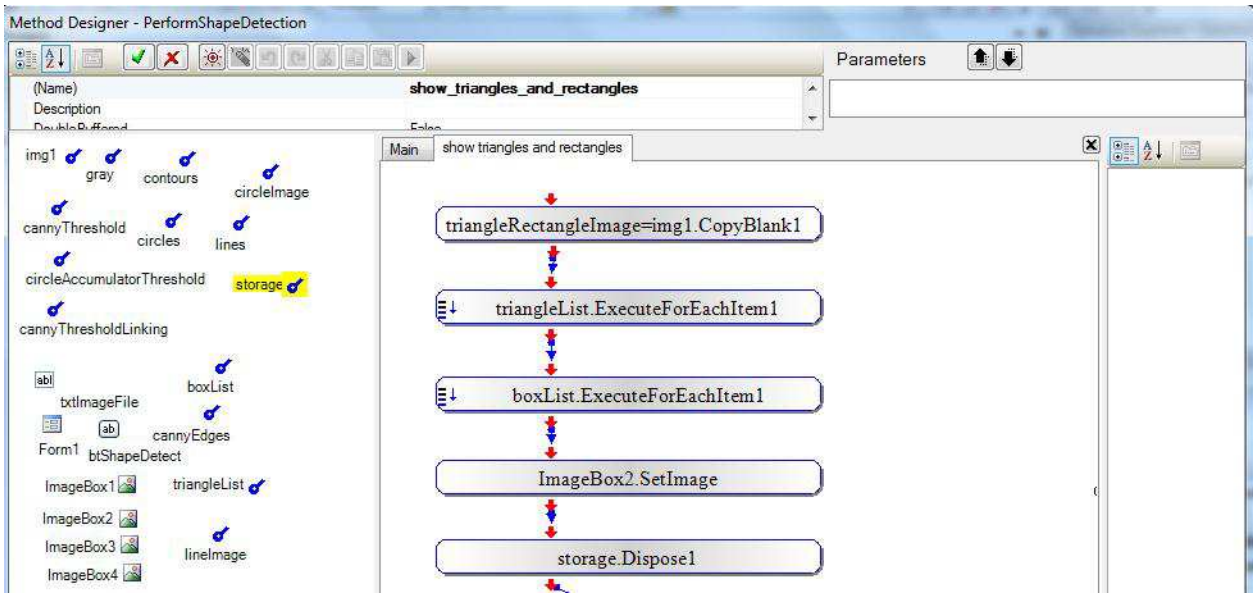




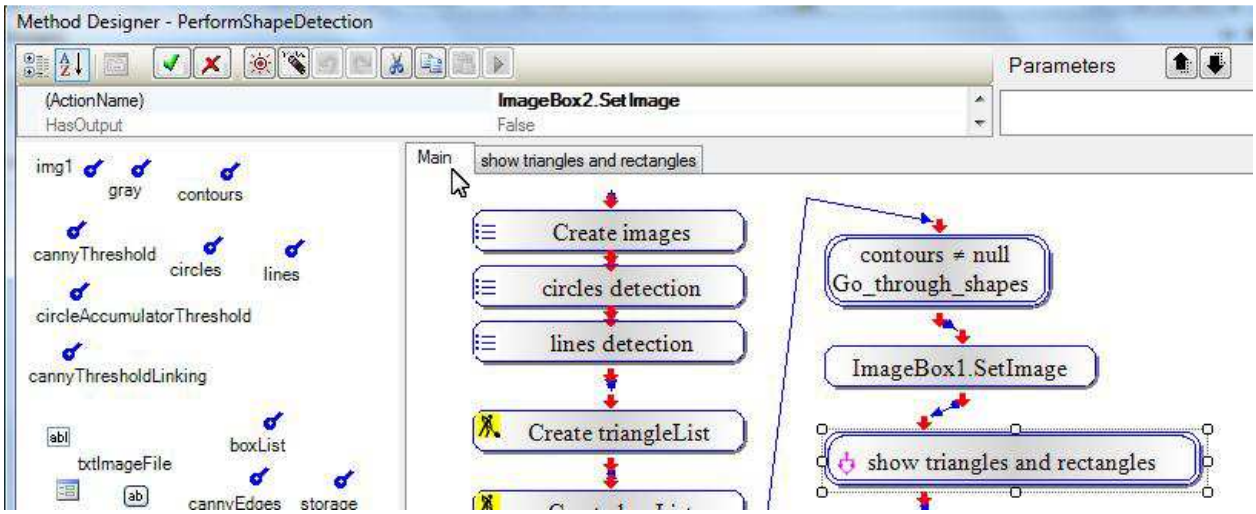
By now the storage object is no longer needed. We may call its Dispose method to free the memory it takes.



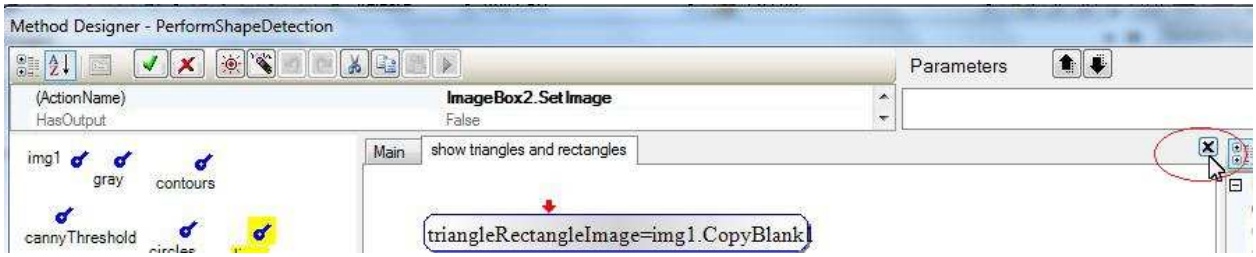
These are the actions for showing triangles and rectangles on the second image box:



We main switch back to the main routine:



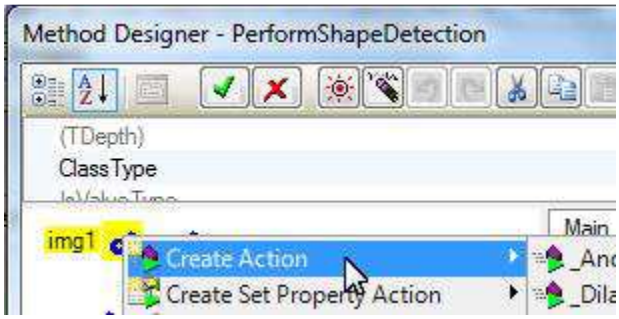
Or close the action group:

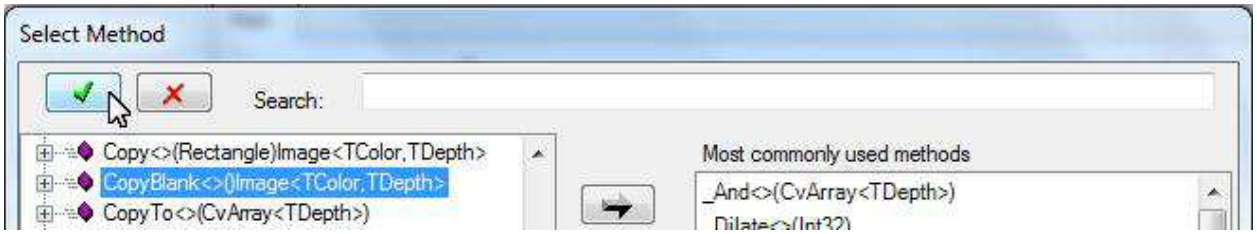


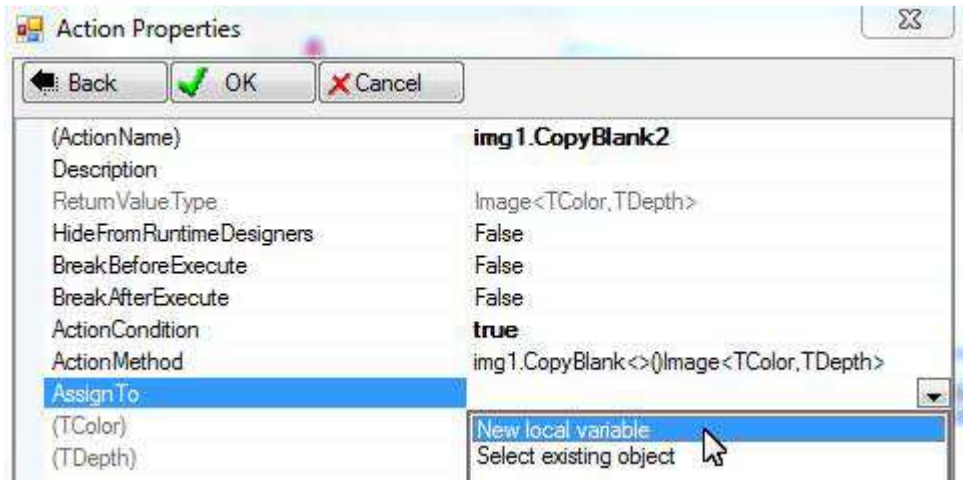
Display Circles

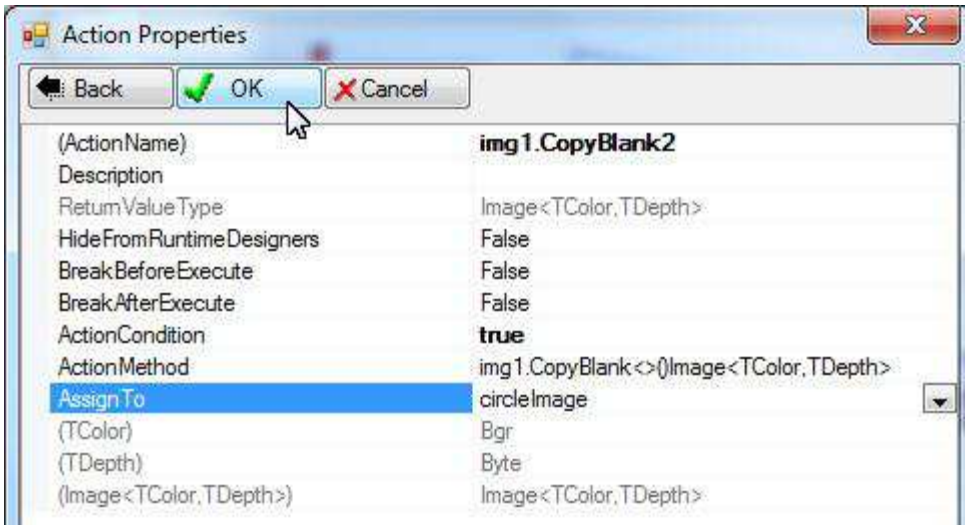
We'll follow the same action pattern: create a blank image; drawing all circles on the image; set the image to an image box.

Create a blank image

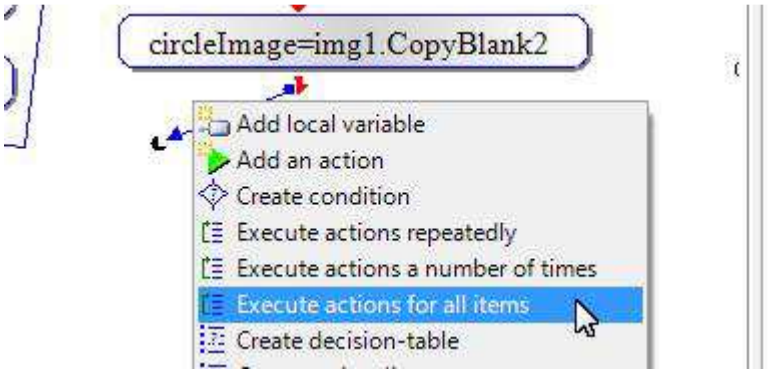


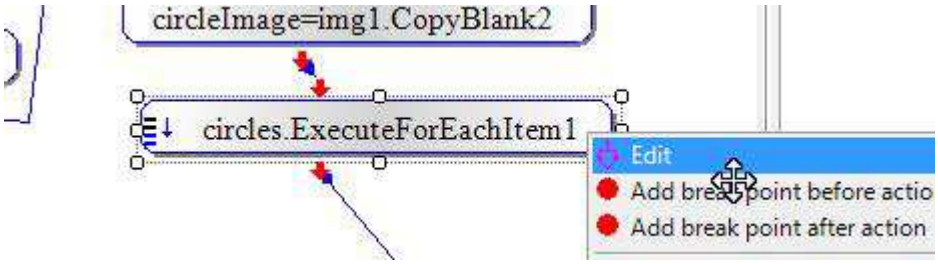


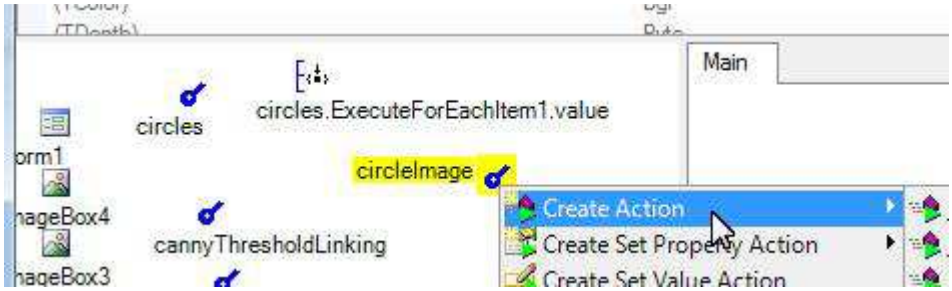


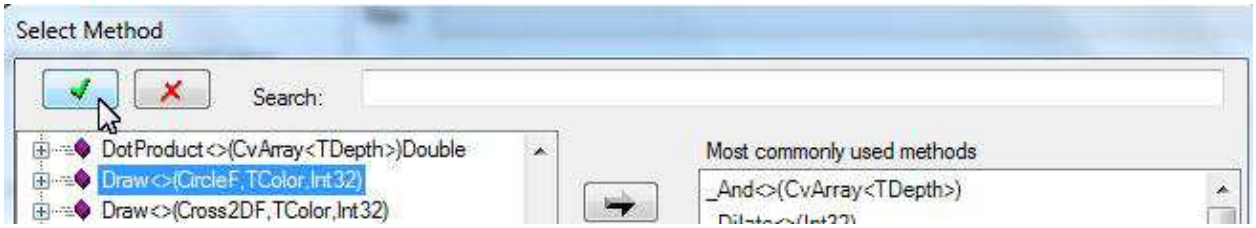


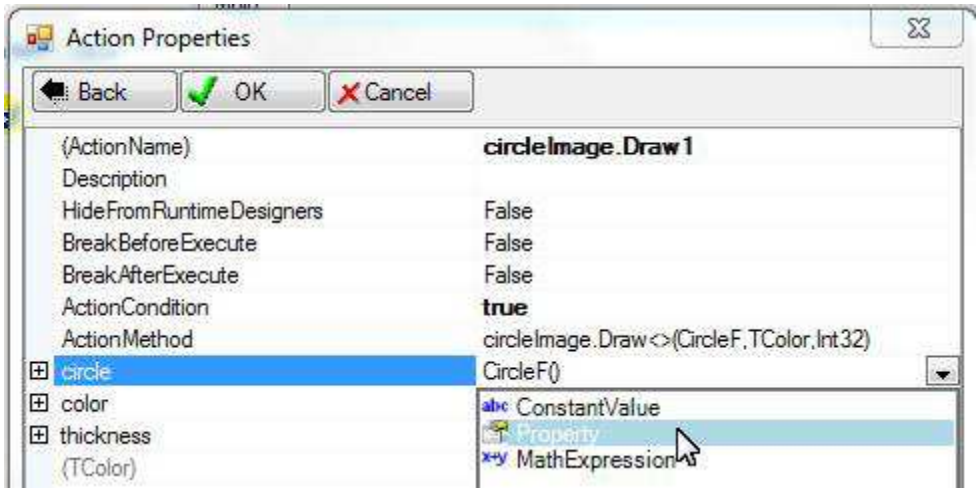
Go through circles

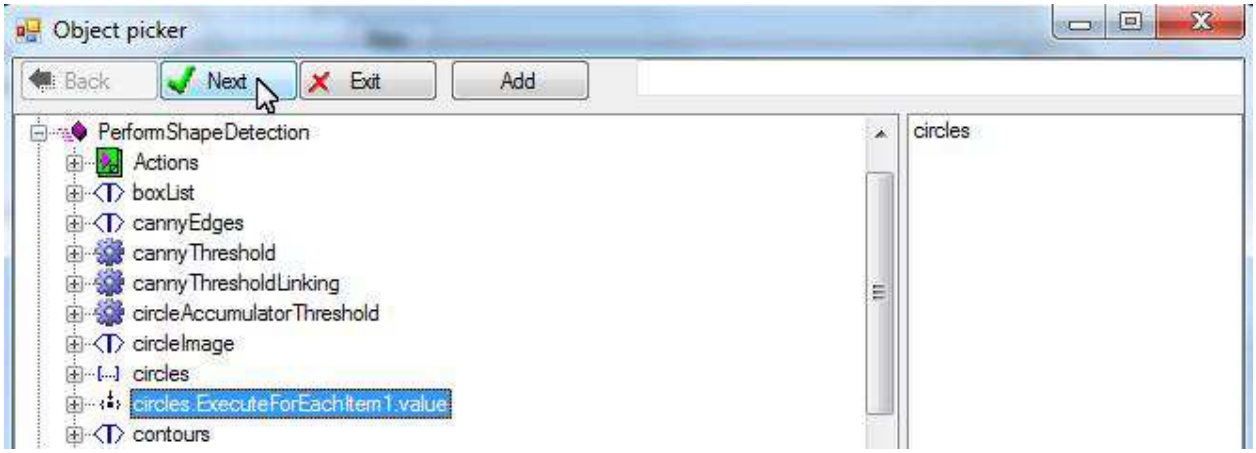


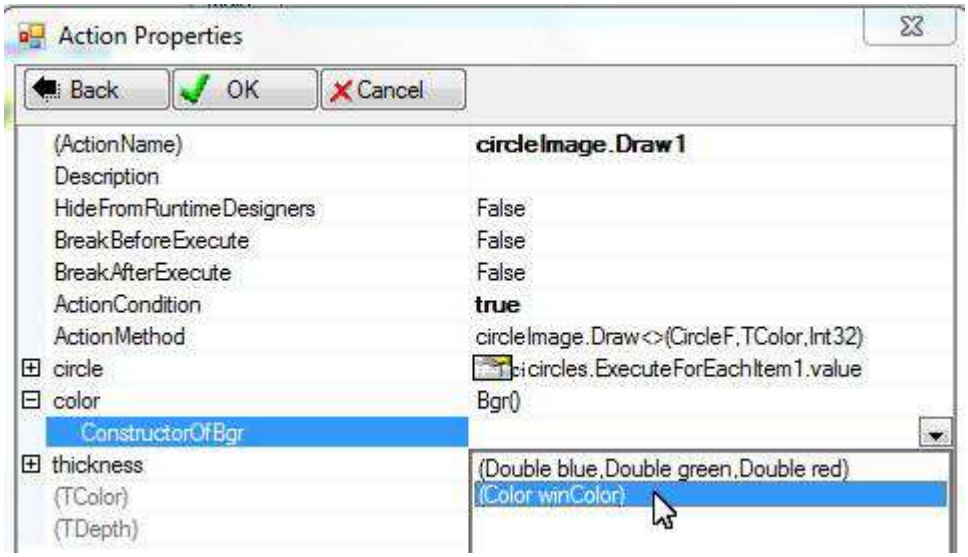

circleImage=img1.CopyBlank2
circles.ExecuteForEachItem1
Edit
Add break point before actio
Add break point after action

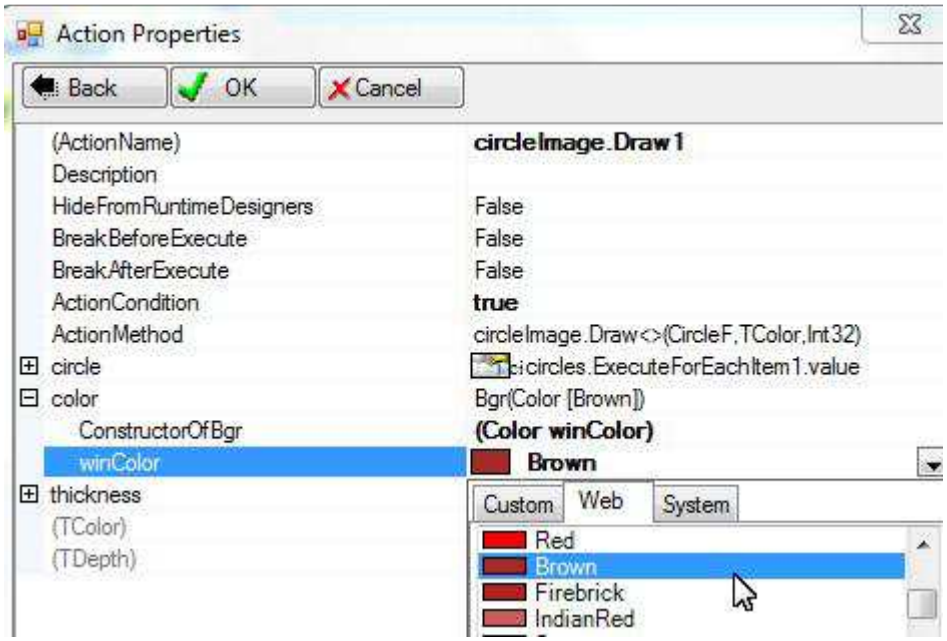

Main
circles
circles.ExecuteForEachItem1.value
circleImage
Create Action
Create Set Property Action
Create Set Value Action
cannyThresholdLinking

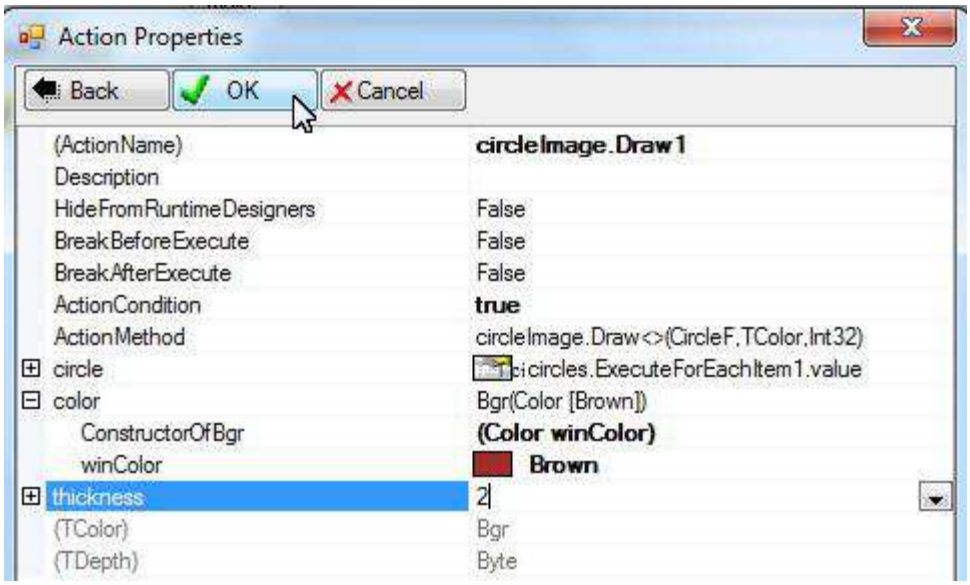

Select Method
Search:
DotProduct<>(CvArray<TDepth>)Double
Draw<>(CircleF,TColor,Int32)
Draw<>(Cross2DF,TColor,Int32)
Most commonly used methods
_And<>(CvArray<TDepth>)

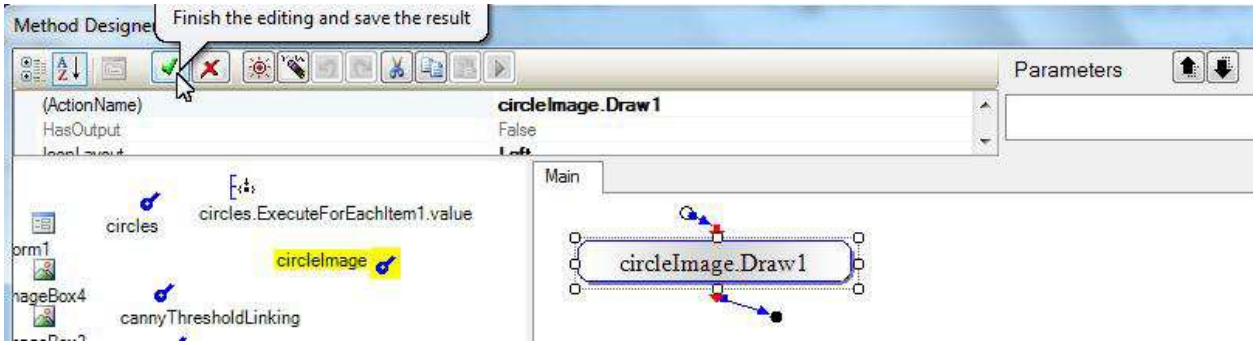

Action Properties
Back
OK
Cancel
(ActionName)
circleImage.Draw1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
circleImage.Draw<>(CircleF,TColor,Int32)
circle
CircleF()
color
thickness
(TColor)
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circleImage
circles
circles.ExecuteForEachItem1.value
contours
circles


Action Properties
Back
OK
Cancel
(ActionName)
circleImage.Draw 1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
circleImage.Draw<>(CircleF,TColor,Int32)
circle
i:circles.ExecuteForEachItem1.value
color
Bgr()
ConstructorOfBgr
thickness
(Double blue,Double green,Double red)
(Color winColor)
(TColor)
(TDepth)

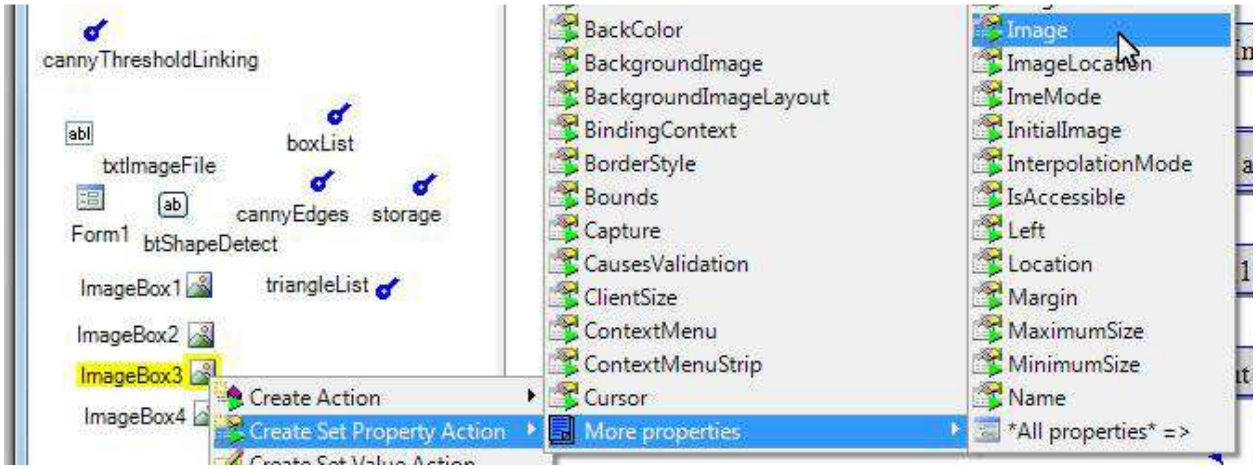

Action Properties
Back
OK
Cancel
(ActionName)
circleImage.Draw 1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
circleImage.Draw<>(CircleF,TColor,Int32)
circle
i:circles.ExecuteForEachItem1.value
color
Bgr(Color [Brown])
ConstructorOfBgr
(Color winColor)
winColor
Brown
thickness
Custom
Web
System
Red
Brown
Firebrick
IndianRed
(TColor)
(TDepth)

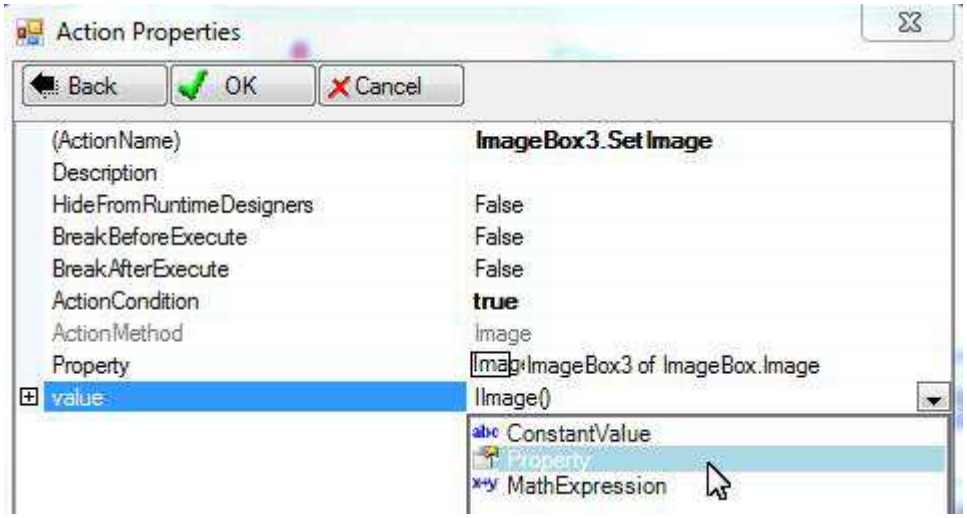


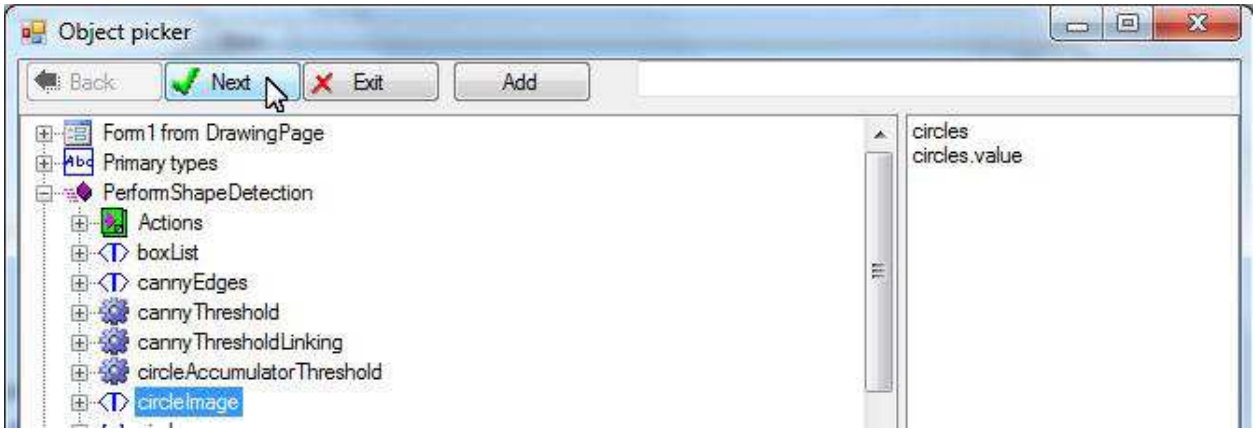


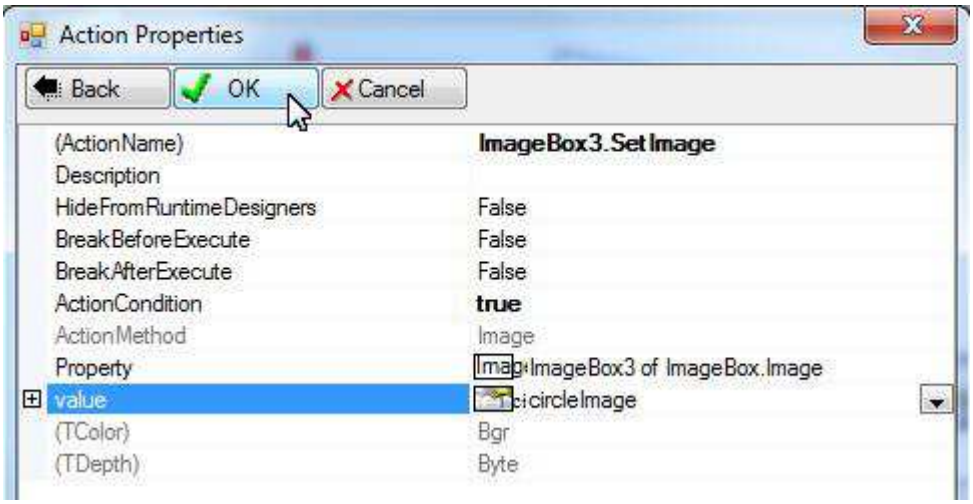
Display image

Display the image on the third image box:



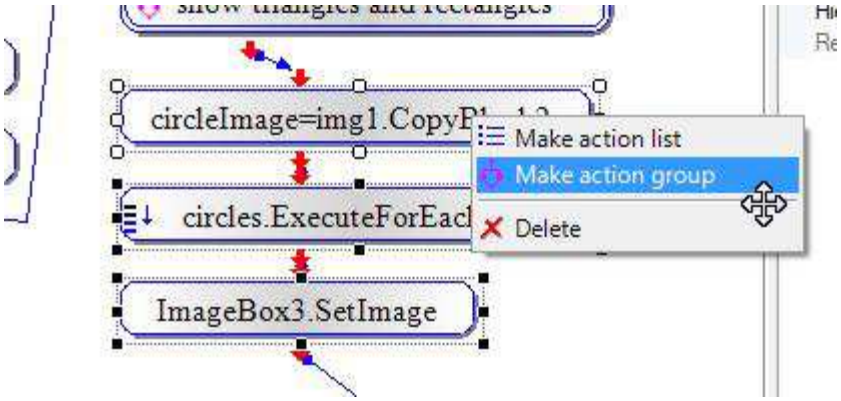




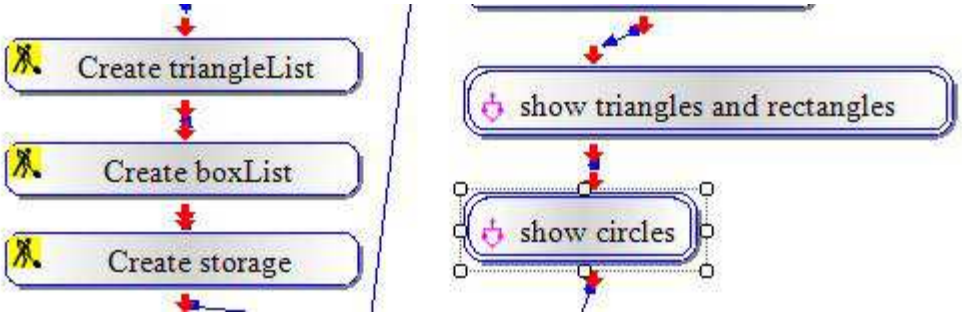


Make action group

This step is not necessary. But it helps make the Editor clean.

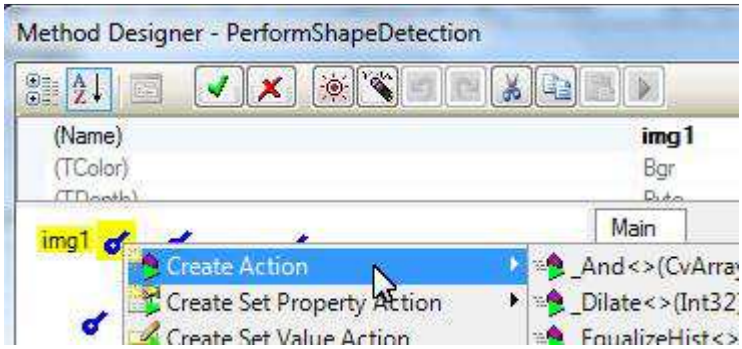


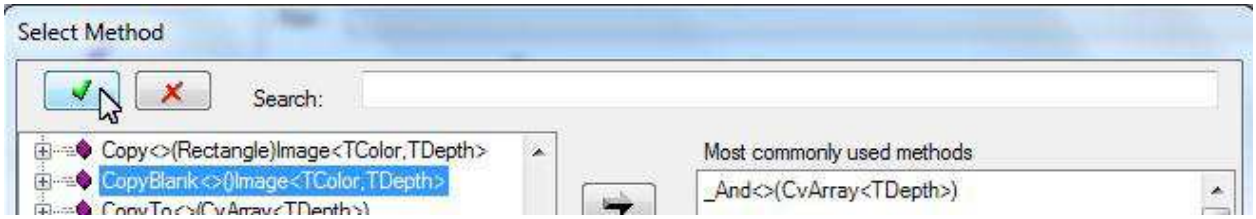
Name the action group “show circles”:

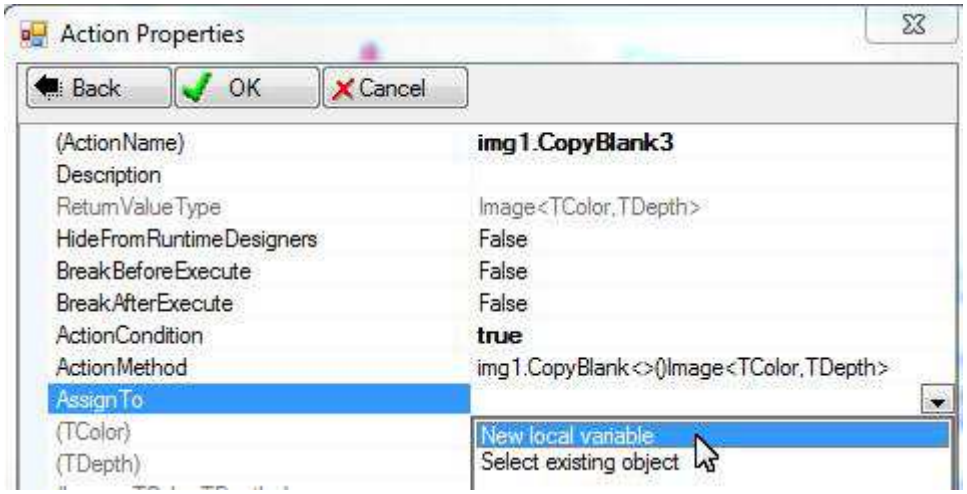


Display Lines

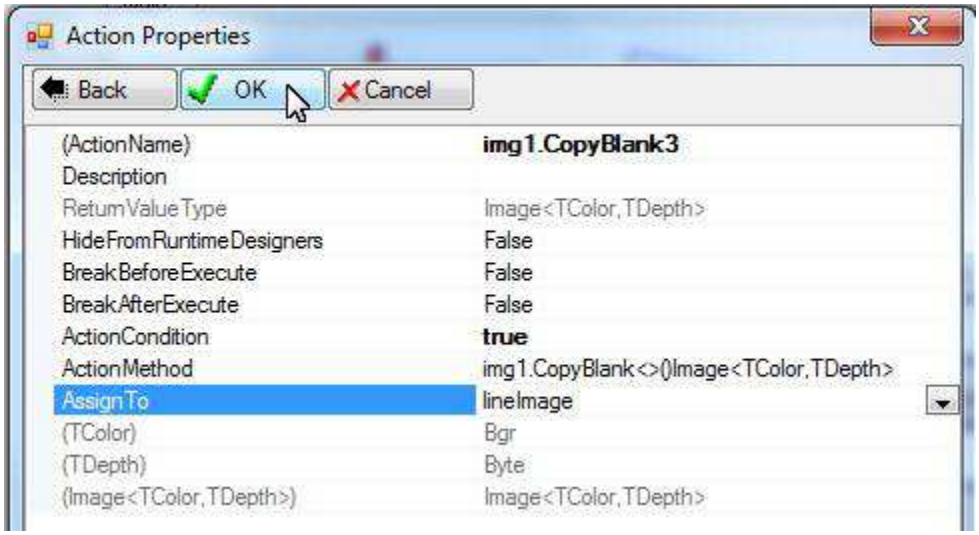
Create blank image



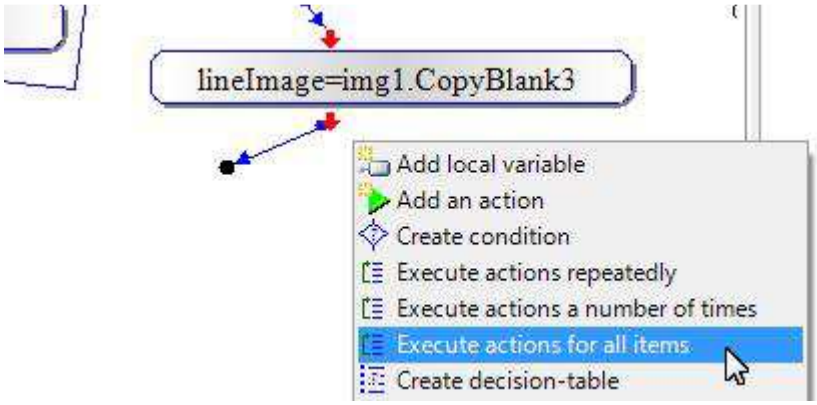


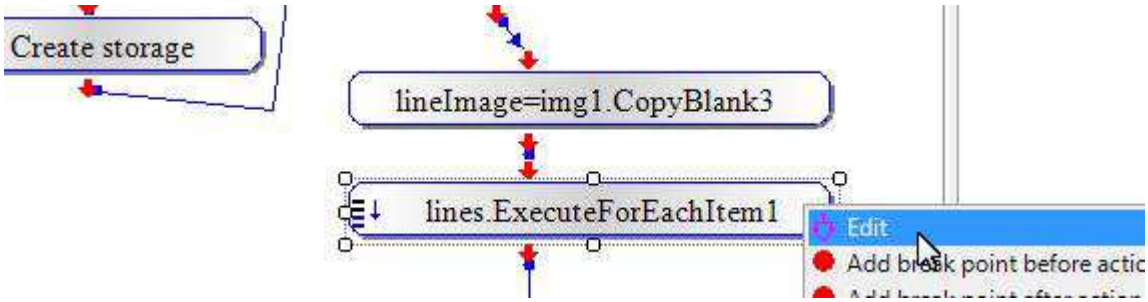

Action Properties
Back
OK
Cancel
(ActionName)
img1.CopyBlank3
Description
ReturnValueType
Image<TColor,TDepth>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
img1.CopyBlank<>()Image<TColor,TDepth>
AssignTo
(TColor)
(TDepth)
New local variable
Select existing object

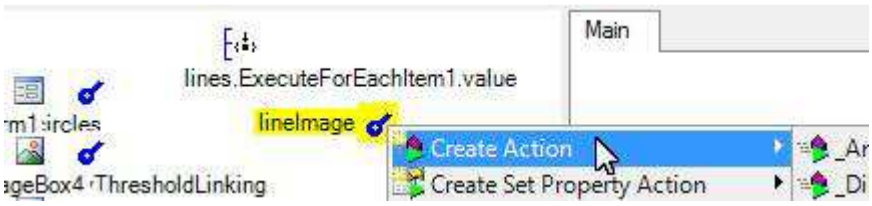

Variable Name
Name:
lineImage
OK

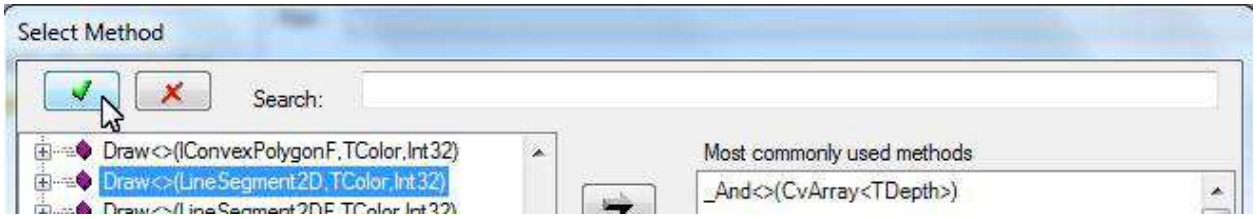

Action Properties
Back
OK
Cancel
(ActionName)
img1.CopyBlank3
Description
ReturnValueType
Image<TColor,TDepth>
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
img1.CopyBlank<>()Image<TColor,TDepth>
AssignTo
lineImage
(TColor)
Bgr
(TDepth)
Byte
(Image<TColor,TDepth>)
Image<TColor,TDepth>

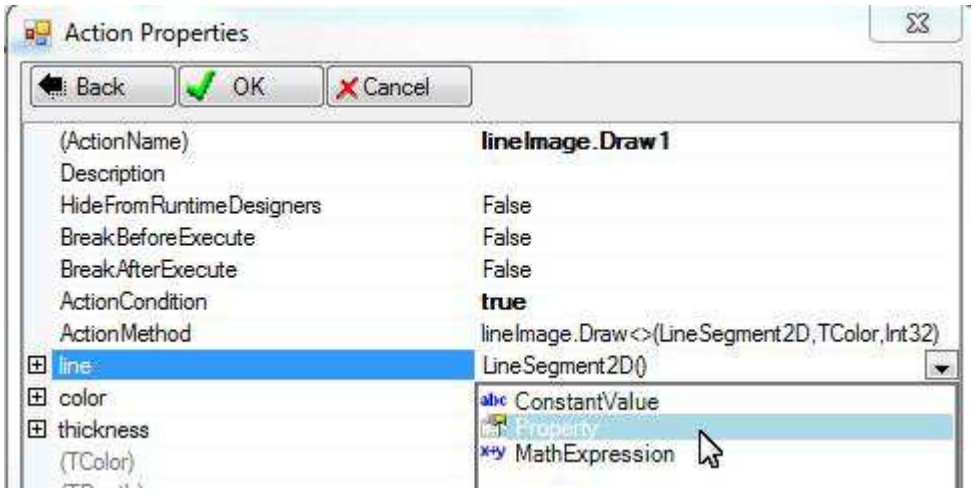
Go through lines

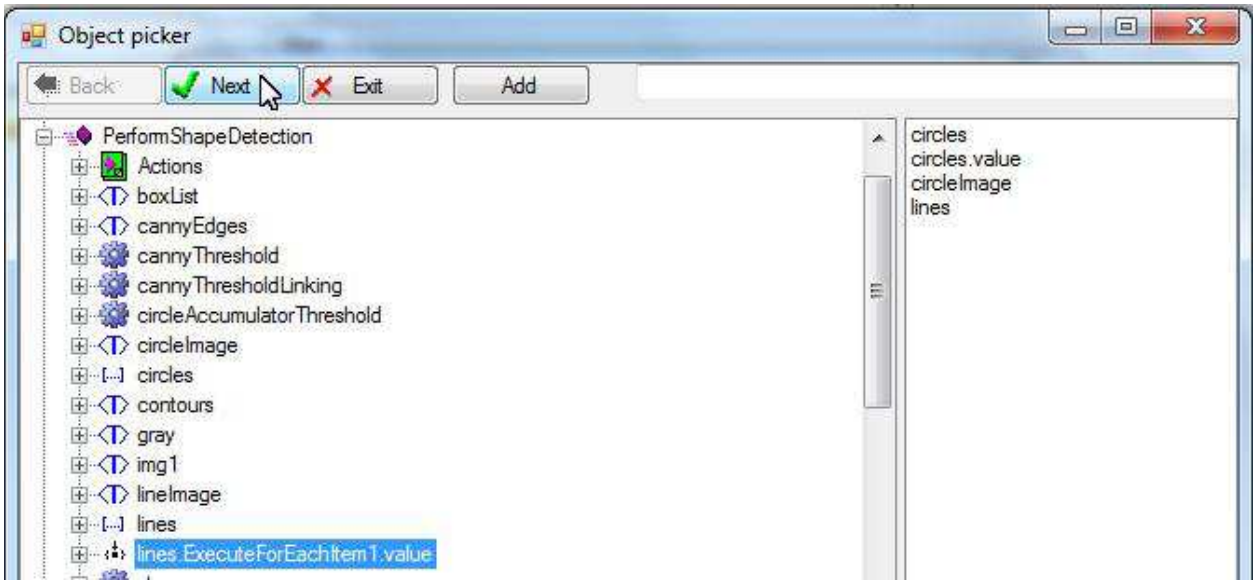


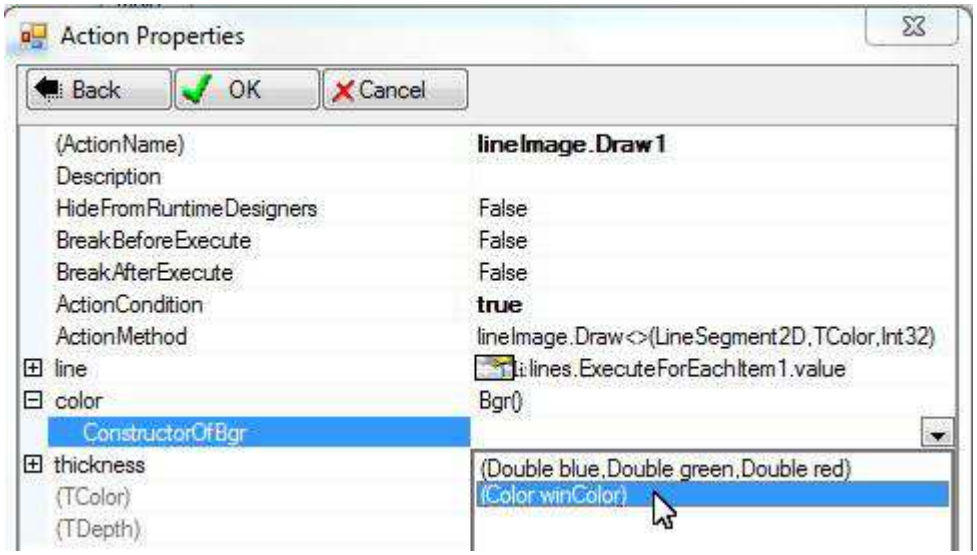


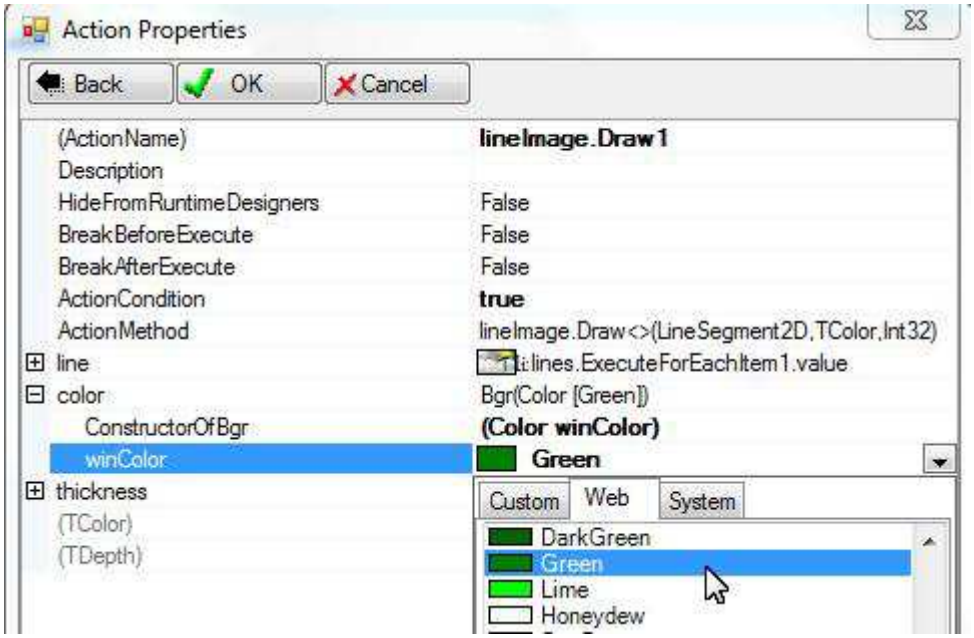


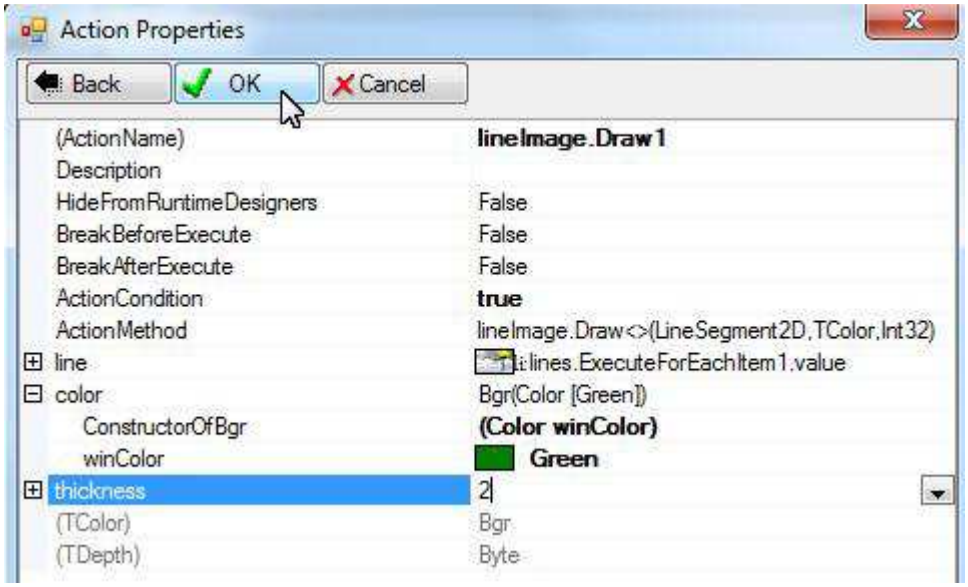


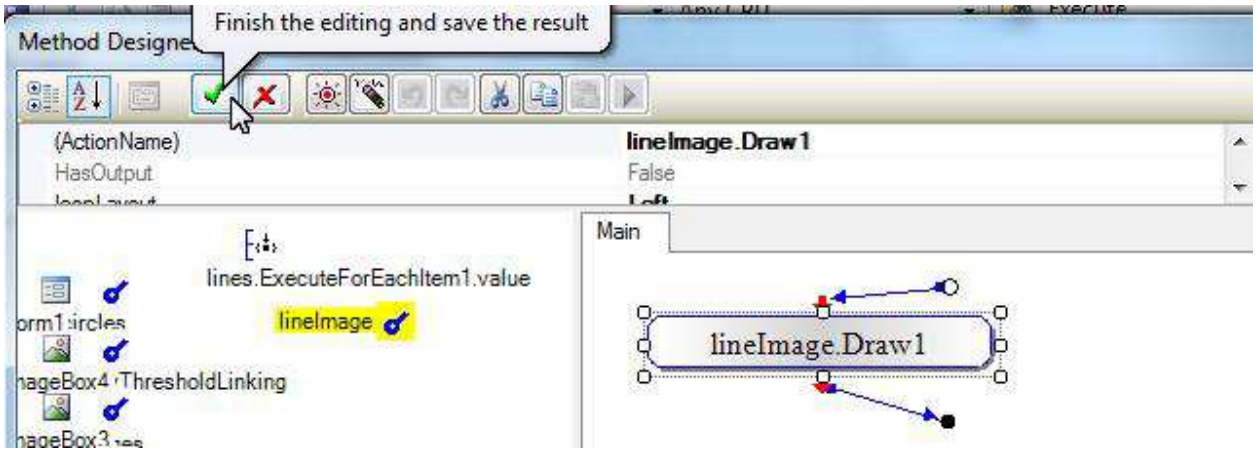

Action Properties
Back
OK
Cancel
(ActionName)
lineImage.Draw 1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
lineImage.Draw<>(LineSegment2D,TColor,Int32)
line
LineSegment2D()
color
thickness
(TColor)
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
PerformShapeDetection
Actions
boxList
cannyEdges
cannyThreshold
cannyThresholdLinking
circleAccumulatorThreshold
circleImage
circles
contours
gray
img1
lineImage
lines
lines.ExecuteForEachItem1.value
circles
circles.value
circleImage
lines


Action Properties
Back
OK
Cancel
(ActionName)
lineImage.Draw 1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
lineImage.Draw<>(LineSegment2D,TColor,Int32)
line
lines.ExecuteForEachItem1.value
color
Bgr()
ConstructorOfBgr
thickness
(TColor)
(TDepth)
(Double blue,Double green,Double red)
(Color winColor)

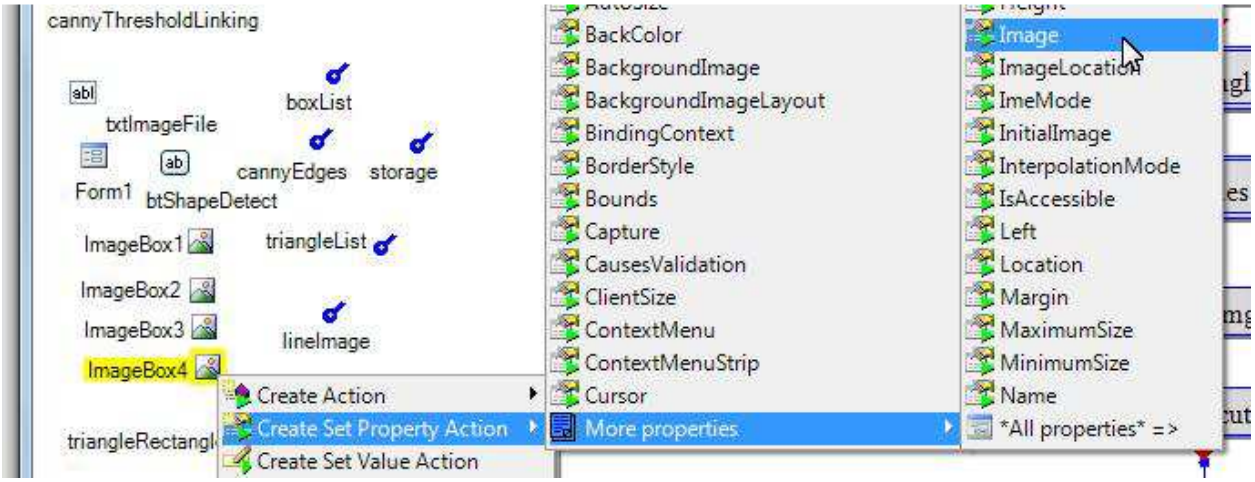

Action Properties
Back
OK
Cancel
(ActionName)
lineImage.Draw1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
lineImage.Draw<>(LineSegment2D,TColor,Int32)
line
i:lines.ExecuteForEachItem1.value
color
Bgr(Color [Green])
ConstructorOfBgr
(Color winColor)
winColor
Green
thickness
(TColor)
(TDepth)
Custom
Web
System
DarkGreen
Green
Lime
Honeydew

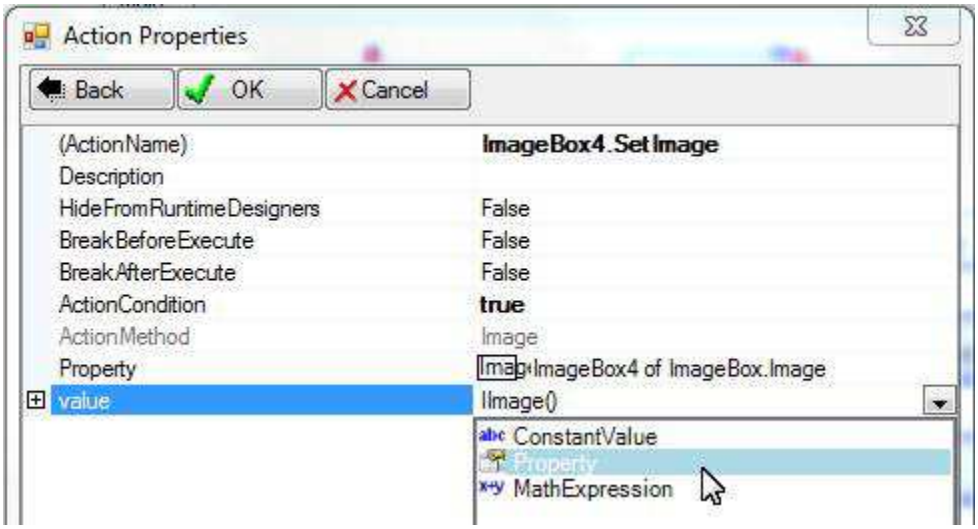

Action Properties
Back
OK
Cancel
(ActionName)
lineImage.Draw1
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
lineImage.Draw<>(LineSegment2D,TColor,Int32)
line
i:lines.ExecuteForEachItem1.value
color
Bgr(Color [Green])
ConstructorOfBgr
(Color winColor)
winColor
Green
thickness
2
(TColor)
Bgr
(TDepth)
Byte

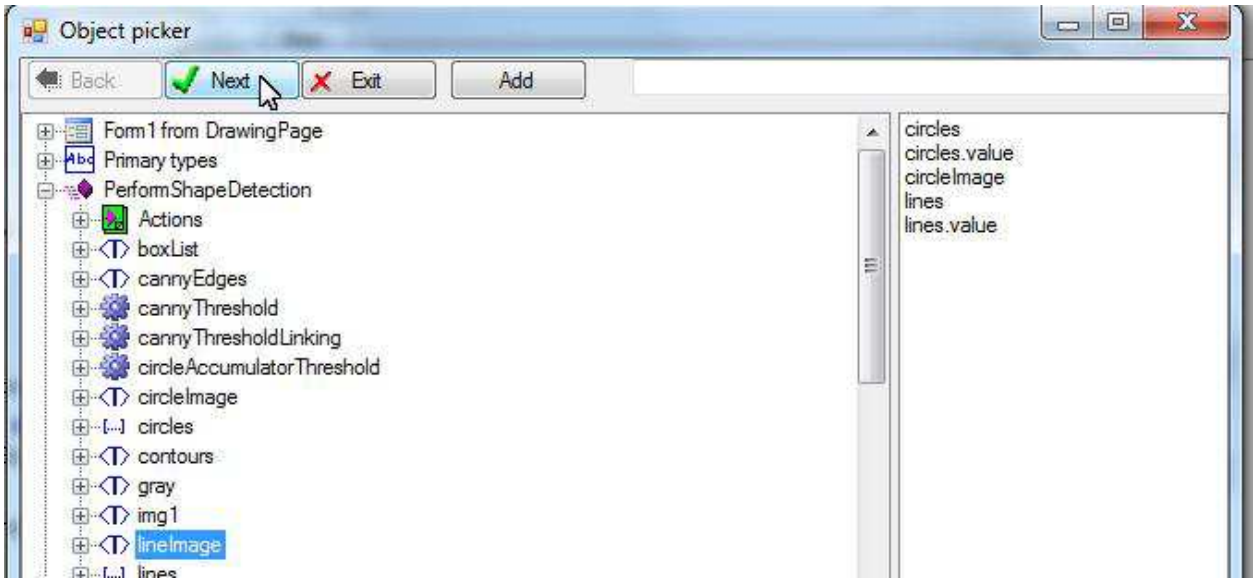

Finish the editing and save the result
Method Designe
(ActionName)
lineImage.Draw1
HasOutput
False
lines.ExecuteForEachItem1.value
lineImage
Main
lineImage.Draw1

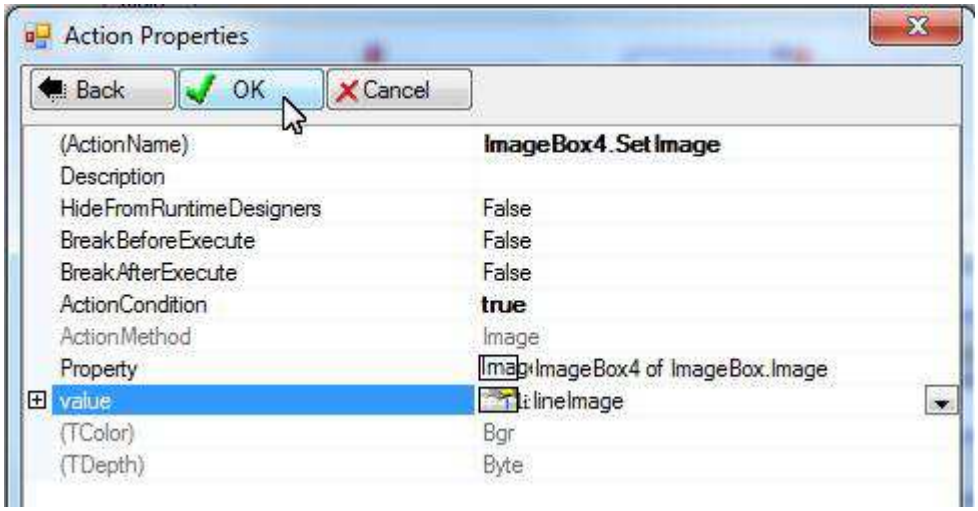
Display image

The lines are displayed on the 4th image box:



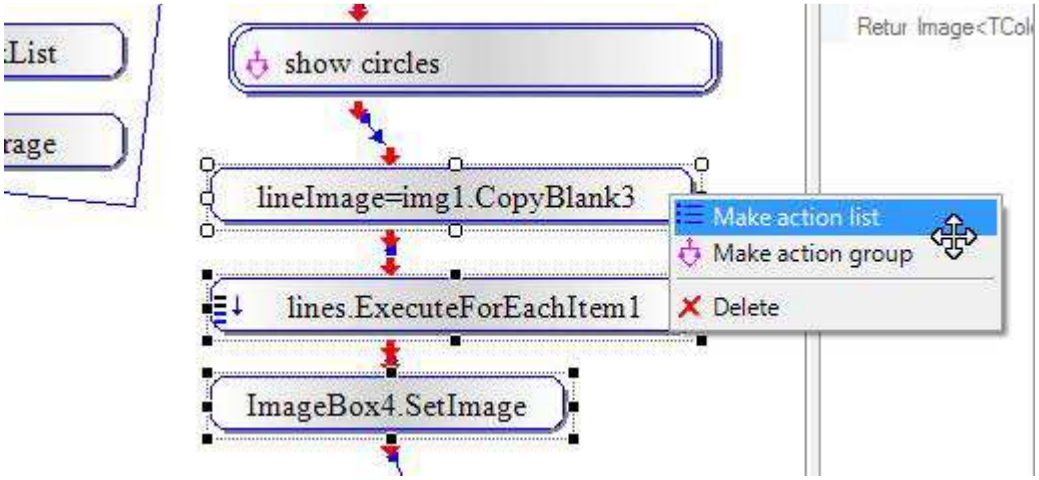




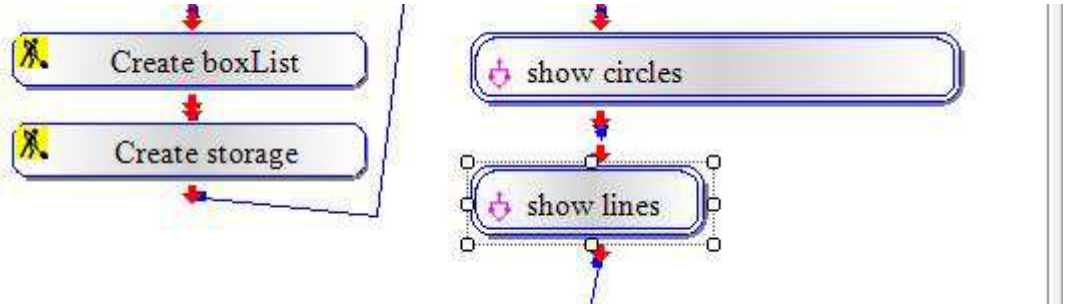


Make action group

This step is optional. It helps to make the diagram cleaner.

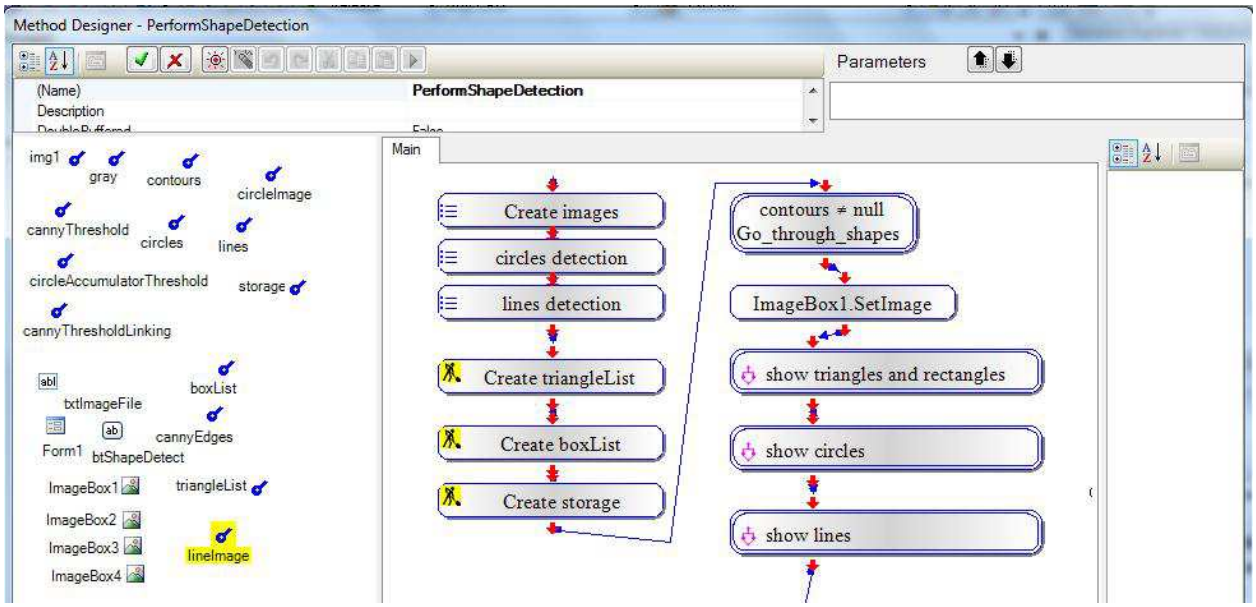


Name the action group “show lines”:

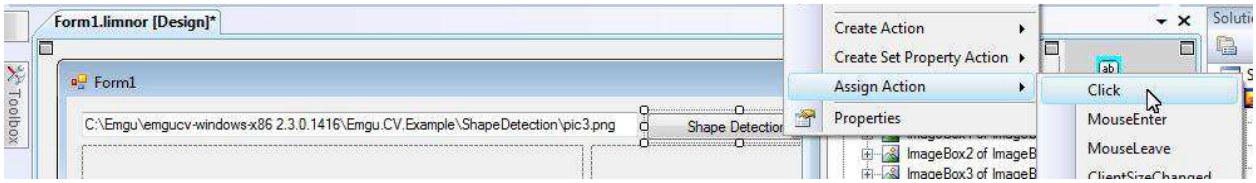


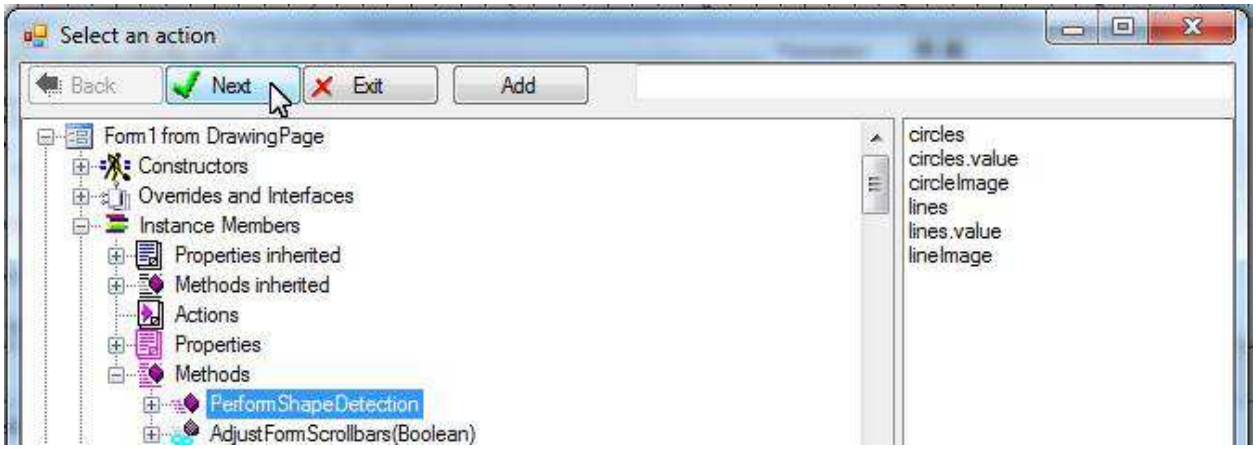
Execute the method

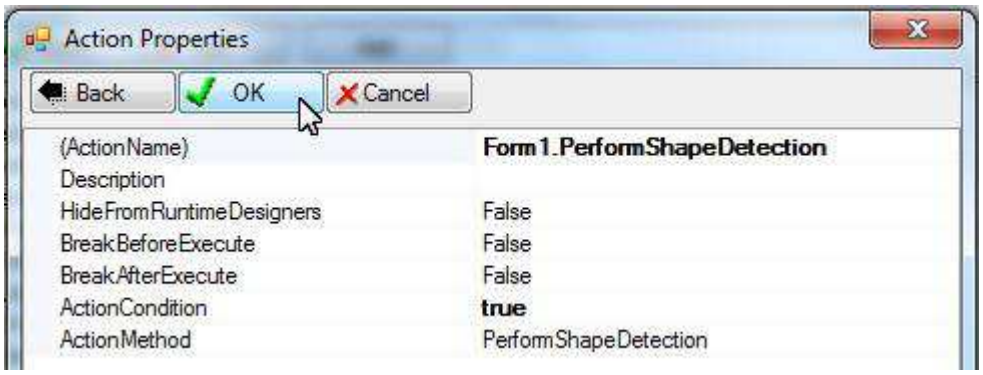
The method is done.



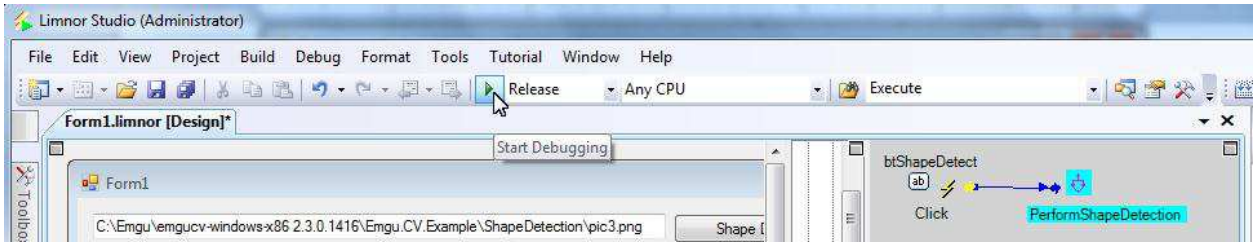
Assign the method to the button:





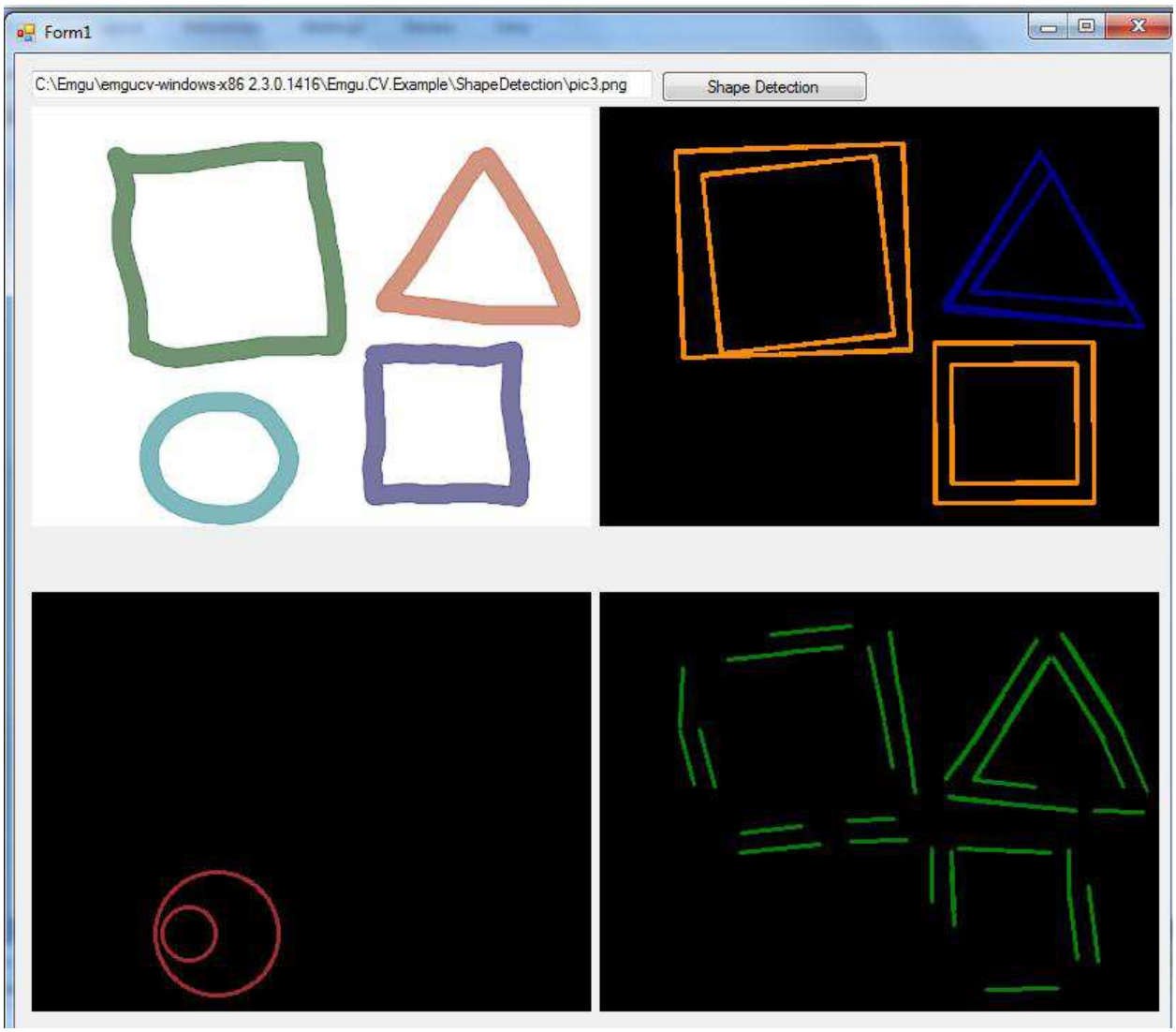


We may test the project now.



The form appears. Click the “Start Detection” button:

The original image and the detected shapes appear on the form.



Feedback

Please send your feedback to support@limnor.com