Contributions on Lineogrammer

Chu-Chi Liu

Master's Project Report 2008

Department of Computer Science

Brown University

There are two parts in this master's project report. Part 1 describes the features implemented in Lineogrammer paper submitted in April 2008. Part 2 introduces the features implemented since then.

Part 1:

1. Polygon Recognizer

Initially, the underlying geometrical structure of the lineogrammer only has lines. In order to manipulate polygons more conveniently, the need to recognize polygons becomes essential. Once a polygon is recognized, it will be filled with the color according to its type of symmetry.

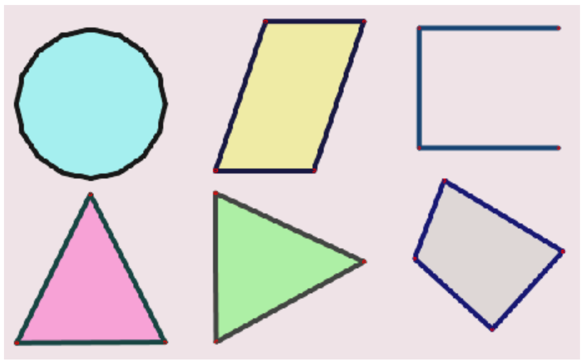


Figure 1.1: Polygons with colors that correspond to their types of symmetry

We use three types of symmetry: horizontal, vertical and rotational to determine the default color of a polygon.

Polygons that contain all three symmetries are color coded light blue. In Figure 1.1, the circle positioned on the top left hand corner fits this description.

The parallelogram on the right hand side of the circle is rotationally symmetrical and therefore is coded by the color yellow.

The bottom row of figure 1.1 contains two triangles, one with a vertical symmetry and one with a horizontal symmetry. The one with vertical symmetry is color coded pink whereas the one with horizontal symmetry is color coded green.

Shapes like the quadrilateral on the bottom left hand corner of figure 1.1 that contains no symmetry whatsoever are assigned the color gray.

The line segments above the quadrilateral do not form a cycle and, therefore, they will not be recognized as a polygon.

Scribbling on to, therefore deleting any segments from a polygon will result in the incompleteness of a cycle. Since an acyclic shape is not recognized as a polygon, the modifying options that polygons allow are not applicable.

Basically, the polygon recognizer is trying to check whether there is a cycle generated after each line stroke or not. If this is the case, a polygon will be created in the underlying geometrical structure of lineogrammer and, according to its type of symmetry, a corresponding color will fill in its fill-area. However, if no cyle is generated, nothing will occur.

2. Select/Deselect

After a polygon is recognized, it is easy to tell whether the stylus is located inside or outside the polygon. Since the fill area of a polygon presents a relatively large surface area for the stylus to tap, tapping the fill area is perhaps one of the easier methods of selecting.

Re-tapping a selected polygon will deselect it. In situations where there are multiple objects, such as other polygons, lines, or labels enclosed within a polygon's fill-area, selecting the polygon will also select the objects enclosed by this polygon. Deselecting a polygon works in the same fashion.

However, the tapping of overlapped areas of multiple polygons creates several problems. After testing several beta versions, we finally came up with the methodology below.

In Figure 2.1, the black dot indicates the tapping point. A polygon is tapped if the tapping point is within its fill area. Subsequently, all the polygons in the enclosed space are tapped.

Of the tapped polygons, the algorithm will select the smallest polygon and any polygons that intersect with it. This implies that our black dot will select the polygons outlined in red.

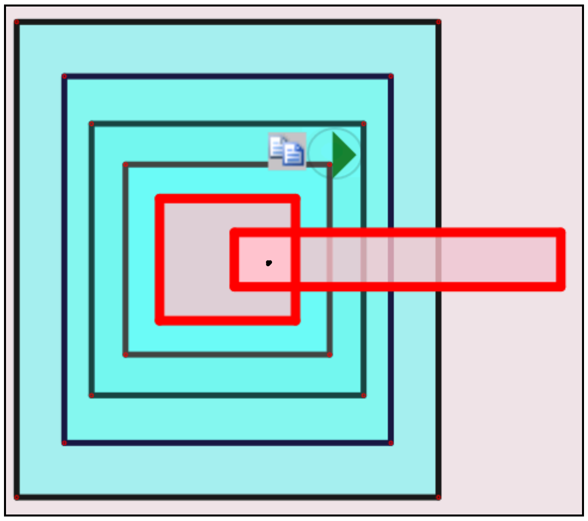


Figure 2.1: The position of the tapping point and the polygons selected

Re-tapping the black dot again will only affect the previously selected polygons. This follows the same algorithm described above.

3. Movement

To move polygons, selecting them first is required. Once a polygon is selected, dragging its fill area with the stylus essentially moves the polygon around.

4. Stretch/Shrink

To stretch polygons, selecting them first is required. Once the polygon is selected, dragging the edge of the polygon with the stylus essentially stretches/shrinks the polygon. Thus the control of the shrinking and stretching of the polygon is easier.

5. Rotation

Imagine a card is pinned on the table with a nail. Then, the card can be easily rotated. The rotation movement in our design is inspired by this idea.

Similarly, selecting the object first before rotation is essential. After the polygon is selected, flicking the stylus will create a "nail" on the drawing canvas.

Notice that the "nail" acts as a pivot/rotation point of the shape.

Once a nail is created, dragging the fill area will rotate the polygon rather than move it (Figure 5.1). It is vital that the distance between the rotation point and stylus is kept constant for the reason that the polygon may enlarge or compress otherwise.

Moreover, the nail can be dragged around as well. This adds flexible maneuverability to the relocating of the rotation point.

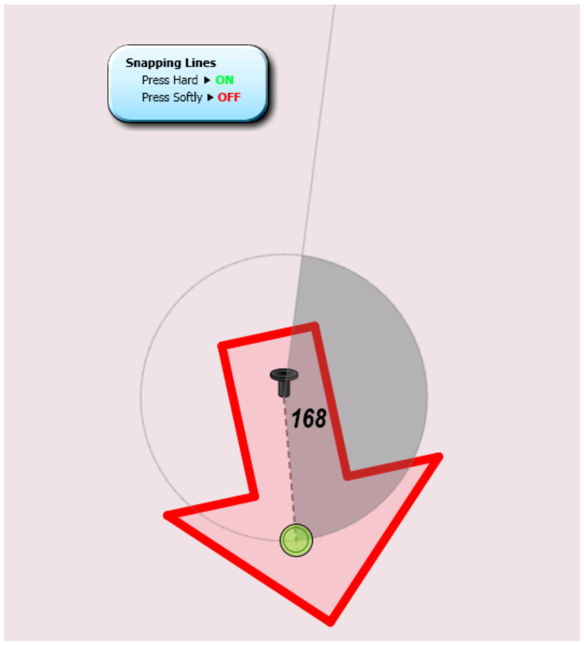


Figure 5.1: An arrowhead is rotated clockwise by 168 degrees

6. Stretch with Nail

If a nail already exists while a polygon is being stretched, then the polygon is stretched according to the location of the nail (stretching center). If multiple polygons are being stretched, the nail location will act as their common stretching center. Meanwhile the nail is still draggable.

7. Curve

Curves can be selected, moved, and rotated, like regular line segments. A complex curve can be formed by joining together multiple simple curves. The curve can be adjusted with the movement of the polygon it's connected to.

Figure 7.1 and 7.2 shows the position/locus of the curves before and after the movement.

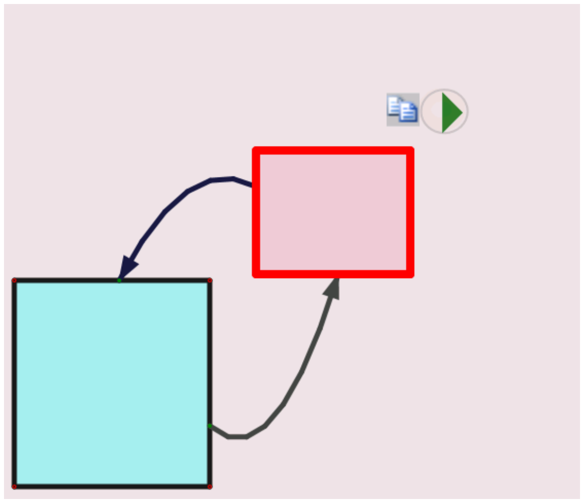


Figure 7.1: How the curves look before moving the polygon in red

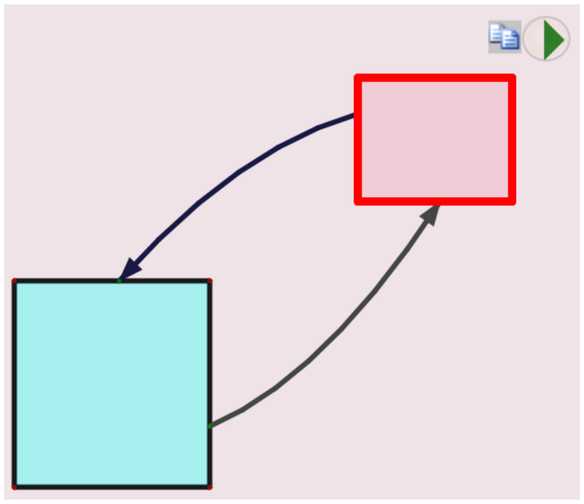


Figure 7.2: How the curves look after moving the polygon in red

8. Polygonized Label

A label can be selected by lassoing it (or by tapping it if it's in edit mode). After being selected, it can then be moved, rotated, or stretched similar to a polygon. Its property, such as color, can also be changed likewise. Hence, labels can be edited just like polygons. This is illustrated in Figure 8.1.

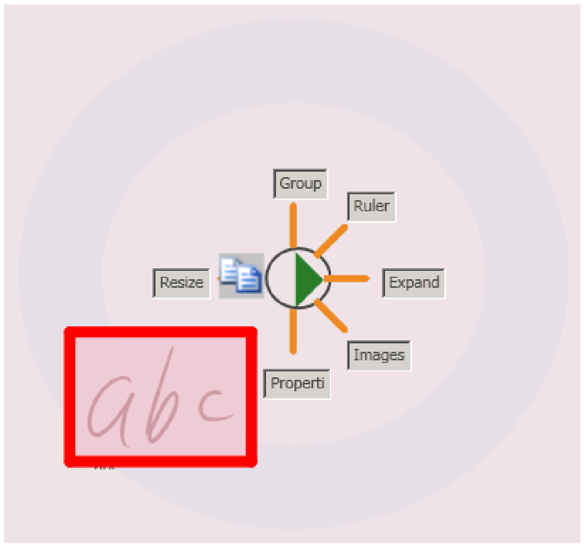


Figure 8.1: A selected label with its marking menu being shown

9. Changing Properties of Polygons/Lines/Labels

After selecting multiple objects such as polygons, lines, or labels, the properties of the selected, for example color, can be edited simultaneously through the "Properties" option in the marking menu (Figure 9.1).

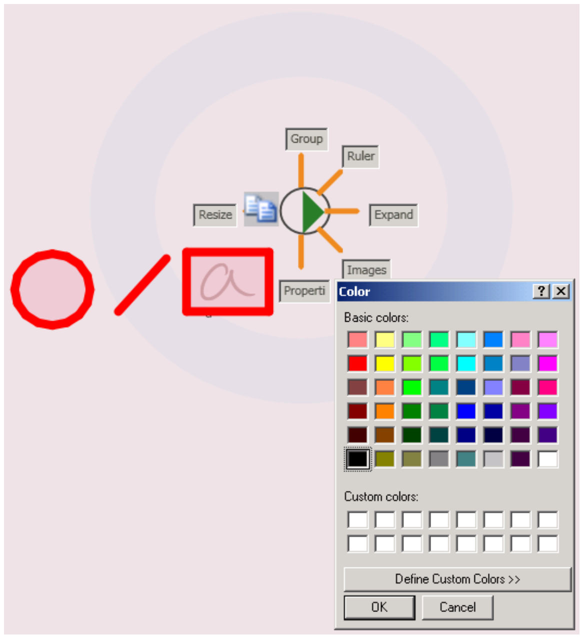


Figure 9.1: Changing Properties via the marking menu

10. Fill an Image for a Polygon

For the selected polygon, its fill area can be filled with an image instead of a color through the "Image" option in the marking menu as shown in Figure 10.1.

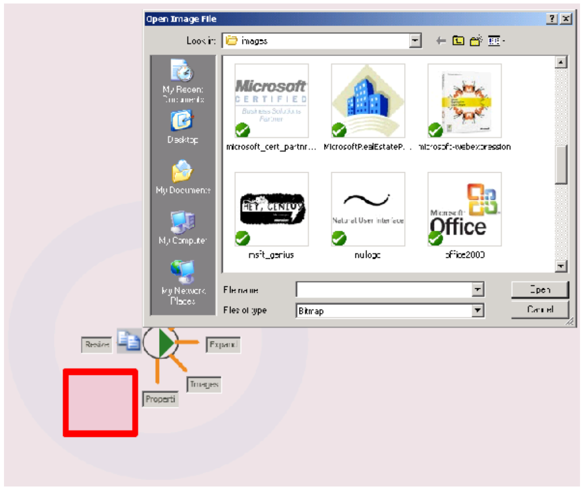


Figure 10.1: Filling the polygon with an image

11. Grouping

When multiple objects such as polygons, lines, and labels are selected, the grouping operation within the marking menu can mark the selected objects as belonging to the same group. After grouping them, selecting any one element in the group will select the other elements in the group too.

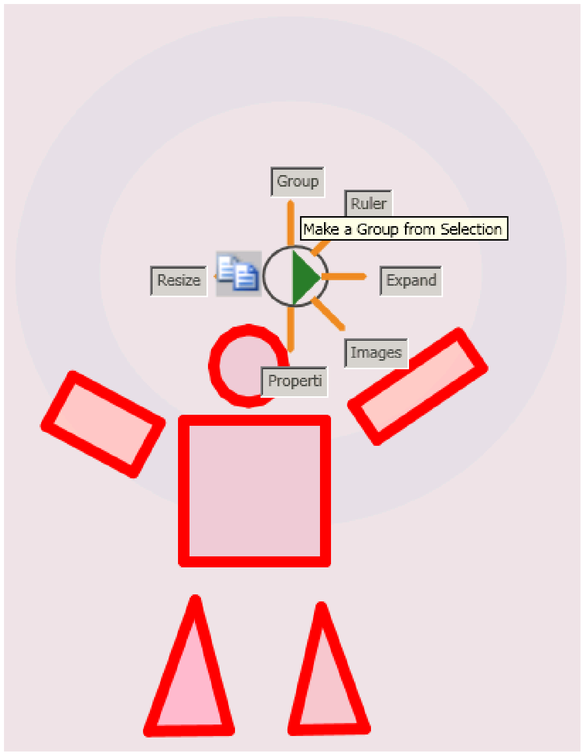


Figure 11.1: Polygons in red are grouped via the grouping option in the marking menu.

12. Connecting Lines

In the previous design, in order to connect two polygons through a line, the endpoints of the connecting line must remain fixed on each side of its connected polygons.

Therefore, any movement of the connected polygons may result in the line dissecting them.

In our new design, the endpoints of the connecting line are fixed inside the polygons rather than at the edge. Moreover, the parts of the connecting line that are inside polygons are made invisible for a neater visual effect. The connecting line looks more natural by not cutting through the polygons as it moves around.

Figure 12.1 and 12.2 shows what the line looks like before and after the polygon movement.

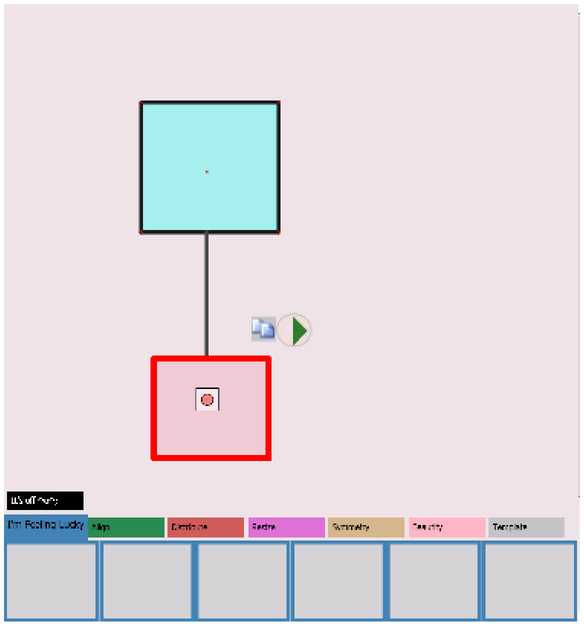


Figure 12.1: A connecting line with its connected polygon before movement

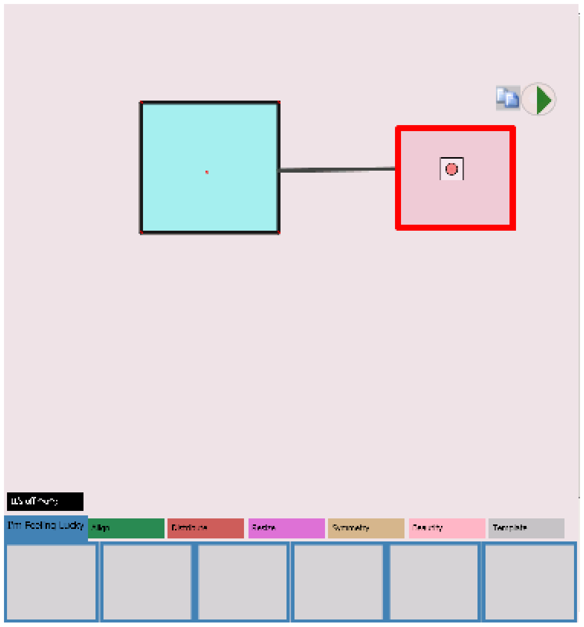


Figure 12.2: A connecting line with its connected polygon after movement

The endpoints inside the polygon can also be relocated for further adjustments as shown in Figure 12.3 and 12.4.

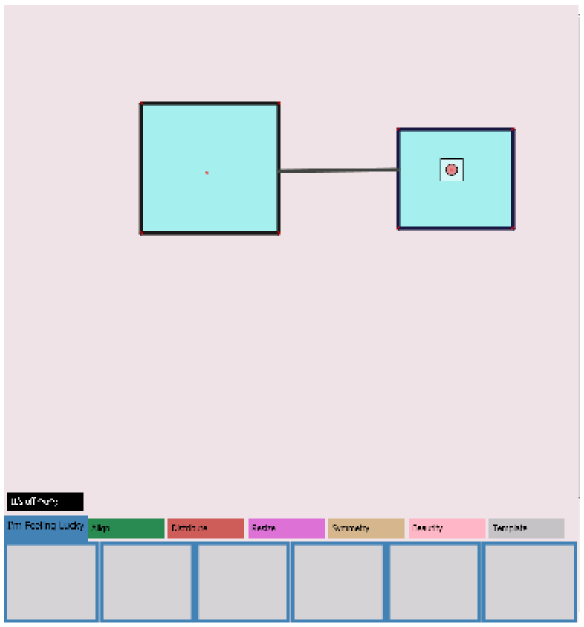


Figure 12.3: Before relocating the right endpoint of the connecting line

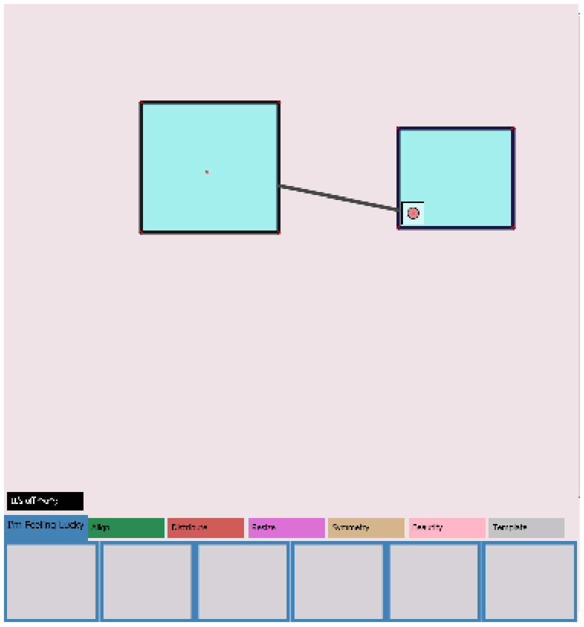


Figure 12.4: After relocating the right endpoint of the conncecting line

13. Alignment Indicator

When the bounding boxes of two polygons are aligned to each other, a line will be shown as the visual feedback indicating the occurrence of the alignment. This feature is commented out presently.

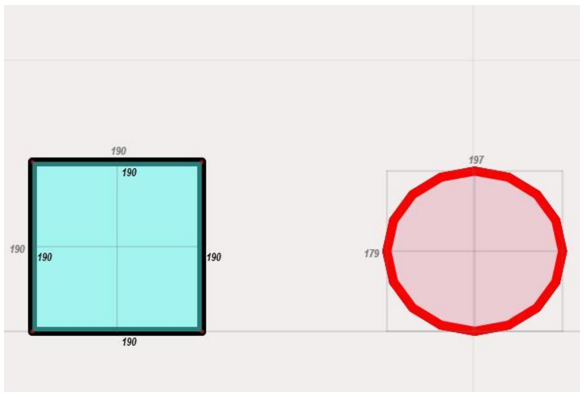


Figure 13.1: Alignmennt indicator showing the occurrence of alignment between the polygons.

14. Length Indicator

While manipulating polygons and lines, the lengths of the polygon edges and the lengths of the lines are shown.

Again, this feature has been commented out presently and can be turned on for further accuracy.

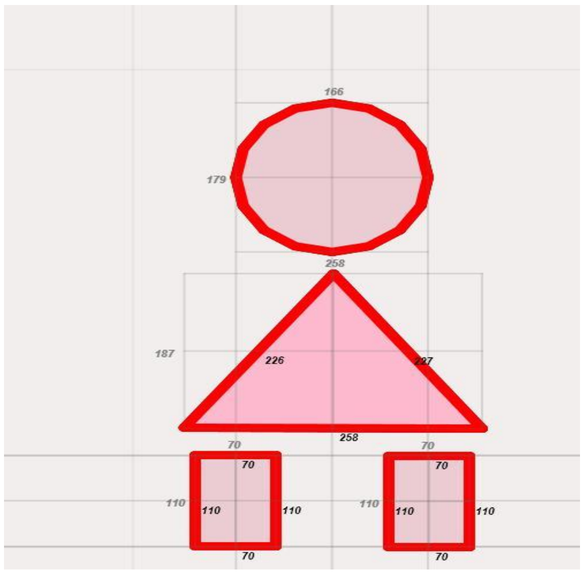


Figure 14.1: Length indicator showing the lengths of the polygon edges

15. Pan

There are two methods that can achieve the effect of panning.

Method one: Selecting all the objects on the canvas will in effect allow movement of the objects as a whole.

Method two: Press the canvas with the stylus and hold for at least half second. Movement of the stylus then will entail movement of every object on the canvas.

This feature is commented out presently.

16. Zoom In/Out

The zoom-in operation is triggered by flattened diagonal zigzag gesture from bottom-left to top-right. This essentially transitions to the interactive zoom rectangle as shown in Figure 16.1. Moreover, the interactive zoom rectangle is controlled by the current position of the stylus.

In order to let the users preview the location of the zoomed-in objects with respect to the screen size after zooming in, an additional rectangular box in light red is displayed on the screen. As a result, users are able to locate the objects with respect to the screen after zooming in.

Figure 16.2 corresponds to the result after zooming in.

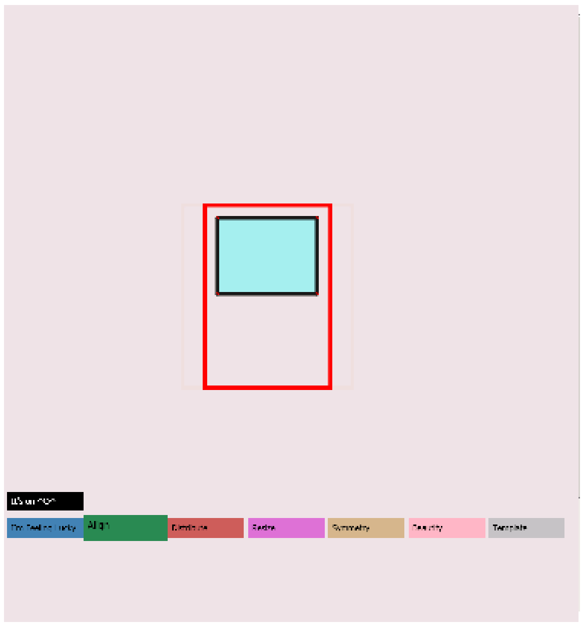


Figure 16.1: Interactive zoom rectangle

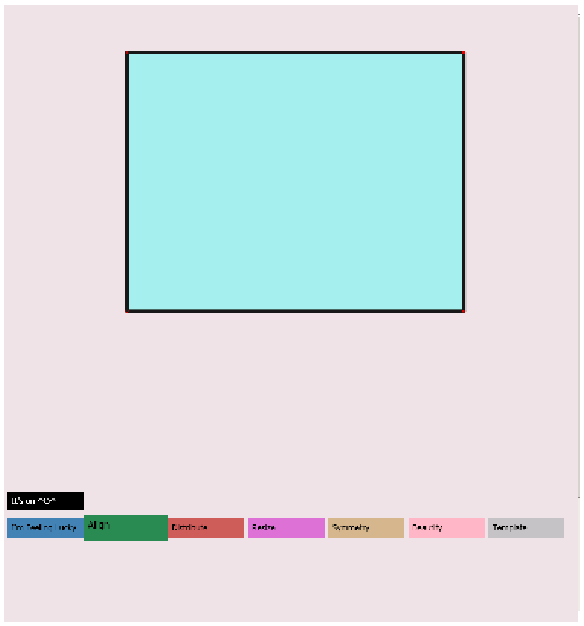


Figure 16.2: The rectangle after zooming in

To be memorable, the zoom-out operation is triggered by performing zoom-in gesture in a reverse way. Moreover, a line for visual feedback will be displayed while the stylus is still on the screen as shown on Figure 16.3 and 16.5.

If the stylus is released while the line is green (Figure 16.3), this restores everything back to the previous zoom level (Figure 16.4).

On the other hand, as the stylus moves away from the starting point, the visual feedback line will turn red after a certain threshold is passed (Figure 16.5). This indicates that interactive zoom-out mode is entered. The amount of zooming out is reversely proportional to the distance between the starting and current position of the stylus.

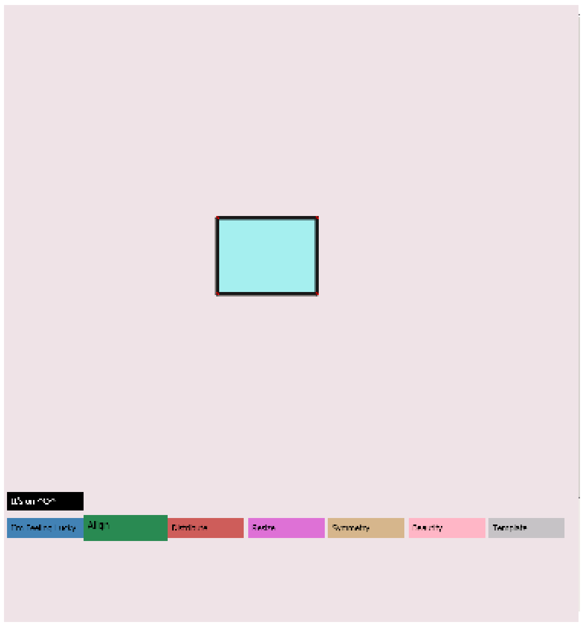


Figure 16.4: The previous zooming level (after normal zooming-out)

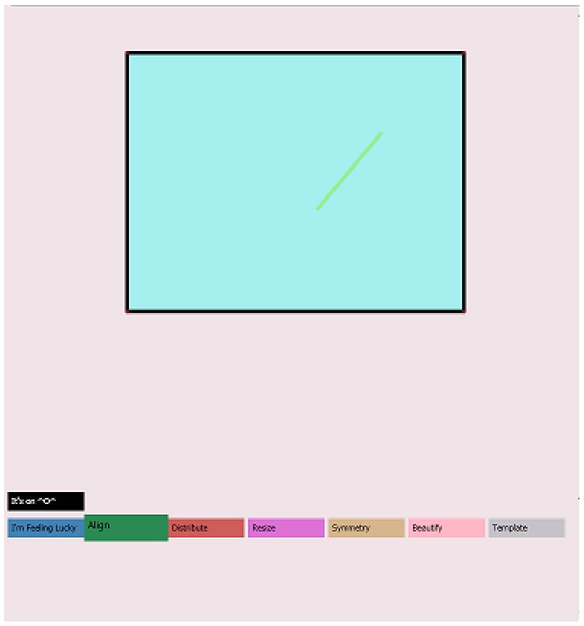


Figure 16.3: A green visual feedback line indicating normal zooming-out.

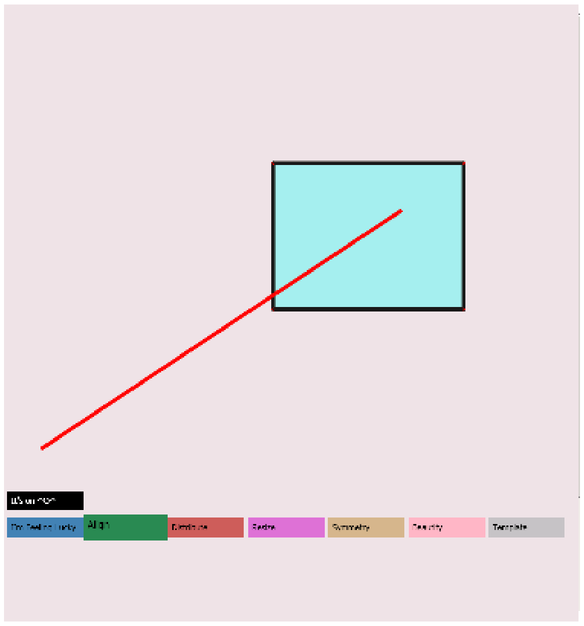


Figure 16.5: A red visual feedback line indicating interactive zooming-out.

Part 2:

On the bottom of the drawing canvas, there are seven operations available: “I’m feeling lucky”, “align”, “distribute”, “resize”, “symmetry”, “beautify”, and “template”. For each operation, as many as six option windows are presented to the user showing how this particular operation is performed. Users can choose an operation and then pick the option window with the most desirable effect.

It is possible to achieve the same result by manual means but this does not fit into our objective of allowing the users to draw with maximum ease and efficiency.

We want the user to be able to focus on their design rather than waste energy exploring the software.

Since only the result, not necessarily the process, is regarded with the foremost importance, methods that allow the same results but require only minimal efforts are optimal.

The objective of this new UI is to reduce the amount of user error and time spent, and avoids unnecessary physical movements.

User error can be reduced in the case that results of the intended operations are always shown to the user pending further confirmation before they are carried out.

The time spent will be reduced as manual manipulations are superseded by automatic drawings by the software itself, similar to that of shortcutting to the final result.

Since all the operations require users to simply tap the screen, excessive physical movements such as repetitive changing of tools can be reduced.

Although as many as six option windows are introduced for each operation, the system only displays the options that are applicable. Therefore, we allow users the option of further adjusting the option windows via selecting polygons on the drawing canvas. For example, in the “align” operation, the polygon selected first is always fixed. All polygons selected later on would be considered for alignment with the first polygon selected.

1. Align

Since free hand drawing can be ambiguous, it is difficult to draw two polygons that align to each other precisely. However, the hand-drawn polygons that are intended to be aligned probably are still very close to each other. Therefore, the algorithm will always try to align the closest two polygons.

The system has six types of alignments: “alignment to the top”, “alignment to the bottom”, “alignment to the left”, “alignment to the right”, “alignment to the horizontal central axis”, and “alignment to the vertical central axis”. In cases where no polygons are selected, the closest two polygons are chosen and the one drawn first is fixed. If there are more than one polygons selected, the first one to be selected is the one fixed.

In Figure 2.1.1, the larger square is drawn first. Hence the larger square is fixed. The system will align the smaller square based on the position of the larger one.

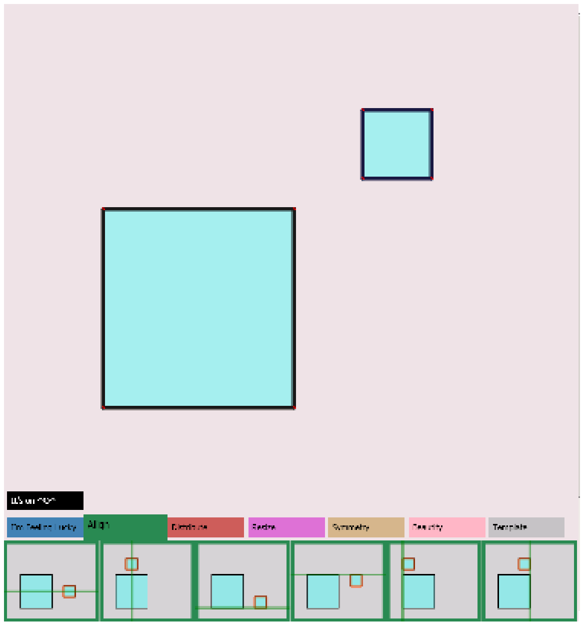
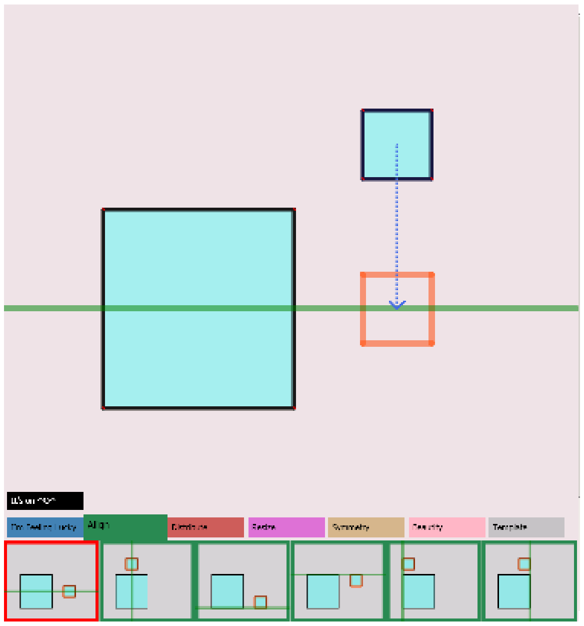


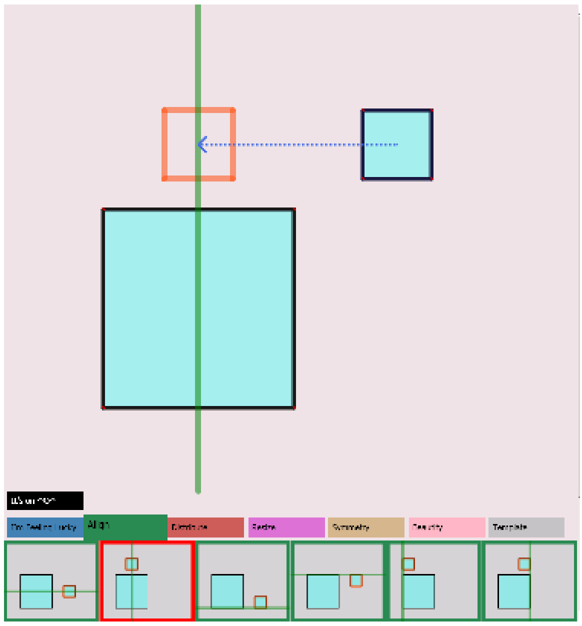
Figure 2.2.1: Two polygons with all the alignment options below.

The following figures illustrate how the six types of alignments are performed. Here, the green line corresponds to the alignment axis and the orange box is the destination.

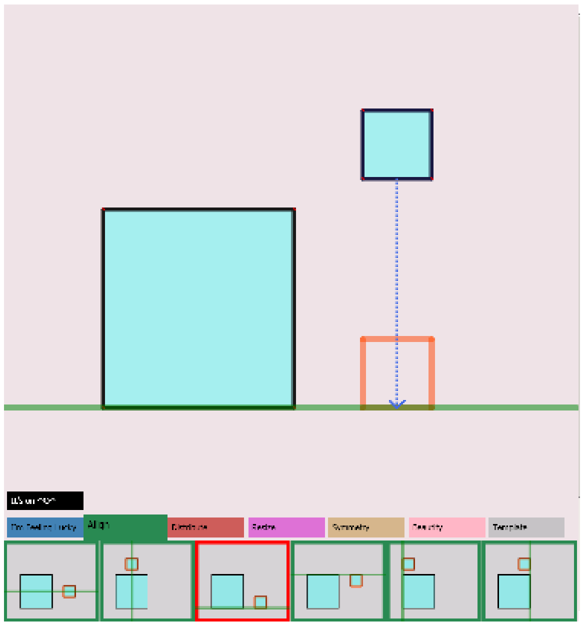
1. Alignment to the horizontal central axis



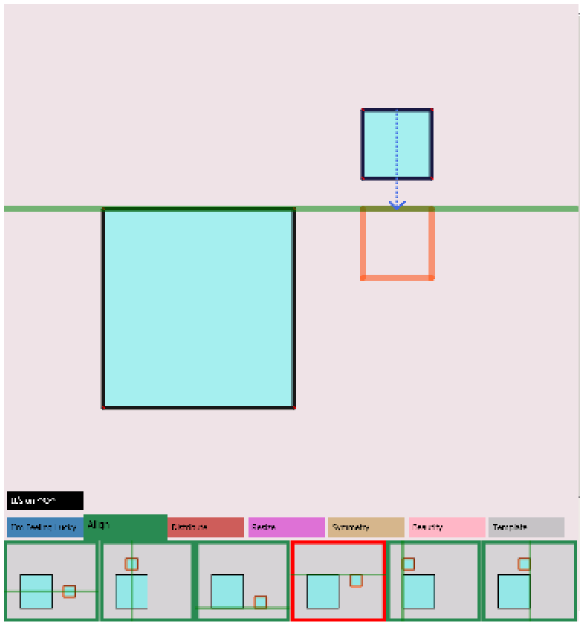
2. Alignment to the vertical central axis



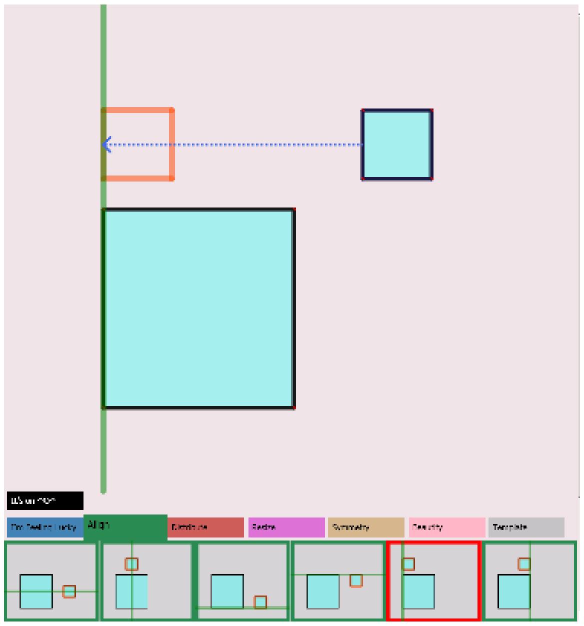
3. Alignment to the bottom



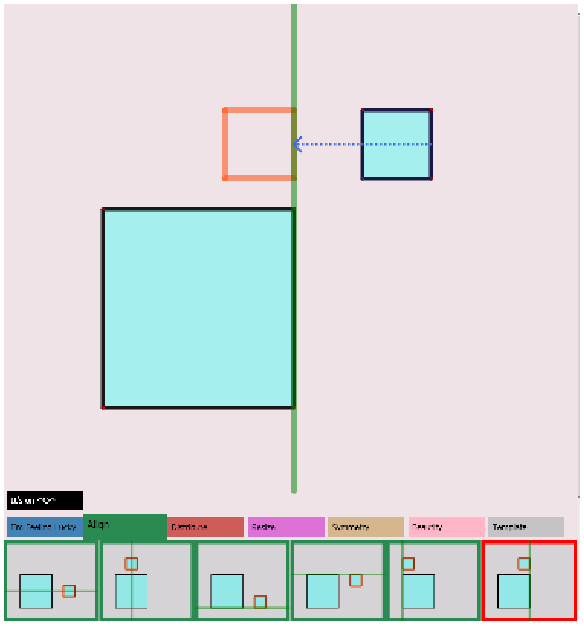
4. Alignment to the top



5. Alignment to the left



6. Alignment to the right



In the situation where one polygon is enclosed by another, if neither of them is selected, the outer polygon will be fixed where as the inner polygon is moved. This is shown in Figure 2.1.2.

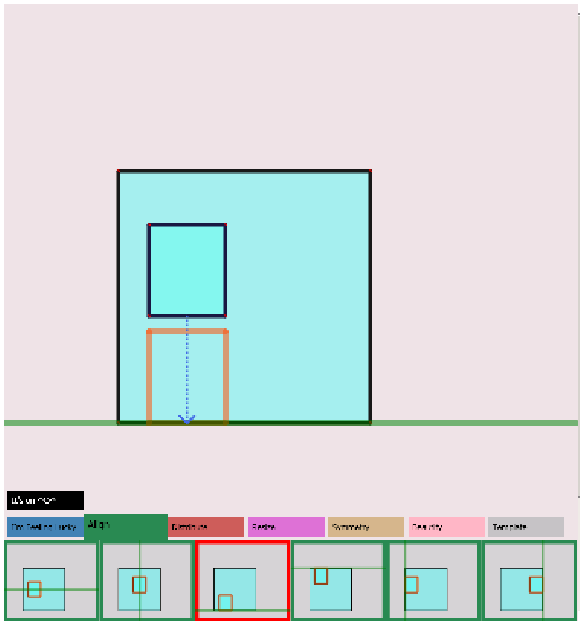


Figure 2.1.2: The outer polygon is fixed and the inner one is aligned to the outer one.

However, if the user wants to keep the inner polygon fixed, this can be achieved by selecting the inner one first followed by the outer one. Figure 2.1.3 illustrates this.

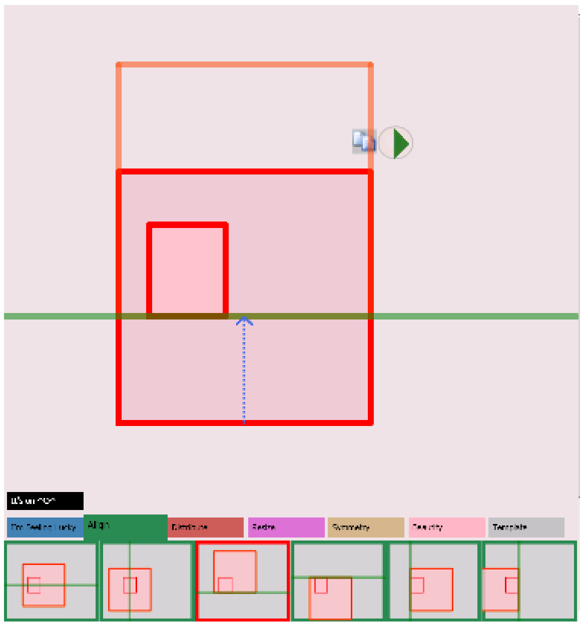


Figure 2.1.3: The inner polygon is fixed and the outer one is aligned to the inner one.

On the other hand, if the outer polygon is selected first, this will be equivalent to the situation where none of the polygons get selected, since the outer polygon is fixed in both cases (Figure 2.1.4).

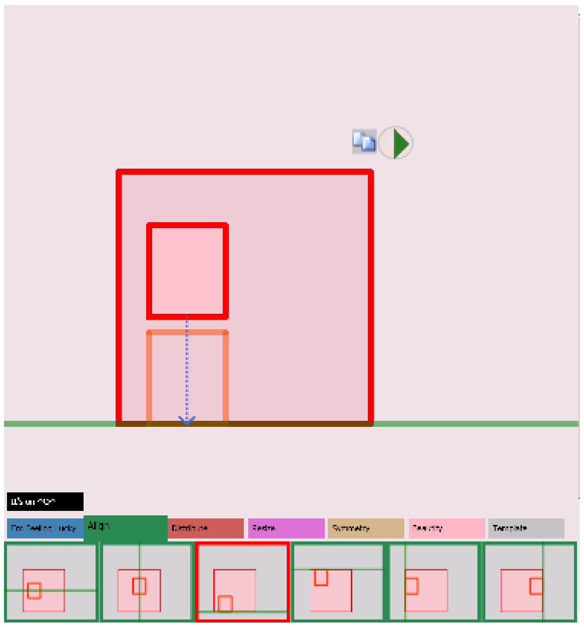


Figure 2.1.4: The outer polygon is fixed and the inner one is aligned to the outer one.

2. Resize

The original Lineogrammer already provides basic resizing operations. To avoid users switching back and forth, the old resizing operation is included in the new design as well. Beside the basic resizing operations, the new design also has a number of enhancements such as making polygons identical, regular, and square, since they comes in handy during the design process.

The algorithm will execute the following routines individually with the results presented in option windows.

Routine 1: The algorithm will pick the two polygons that are most similar, regardless of the number of vertices they have. Then, their bounding boxes will be made identical (Figure 2.2.1).

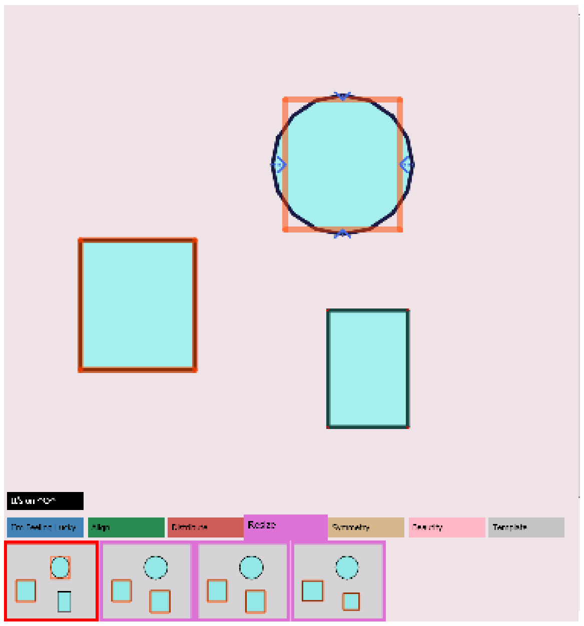


Figure 2.2.1: The rectangle on the left is drawn first, and the circle on the right is resized according to it (numbers of vertices are different)

Routine 2: The algorithm will pick the two polygons that are most similar and have the same number of vertices. Once again, their bounding boxes will be made identical. (Figure 2.2.2)

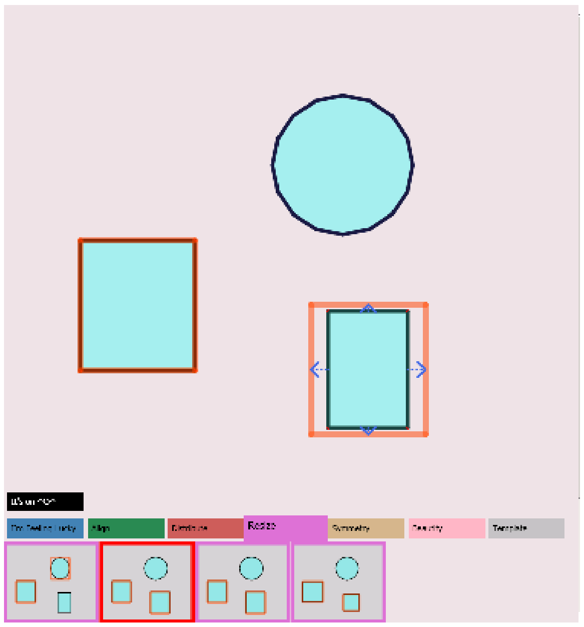


Figure 2.2.2: The rectangle on the left is drawn first, and the rectangle on the right is resized according to it (numbers of vertices are the same)

Routine 3: The algorithm will pick the two polygons that are most similar and have the same number of vertices. Here, Polygons will be made identical. (Figure 2.2.3)

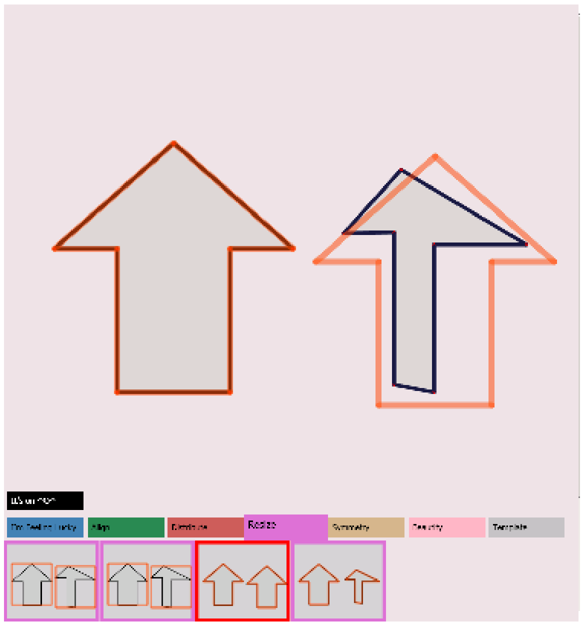


Figure 2.2.3: The arrowhead on the left is drawn first, and the arrowhead on the right is made identical according to the left one.

Routine 4: The algorithm will make the convex non-regular polygons regular. (Figure 2.2.4)

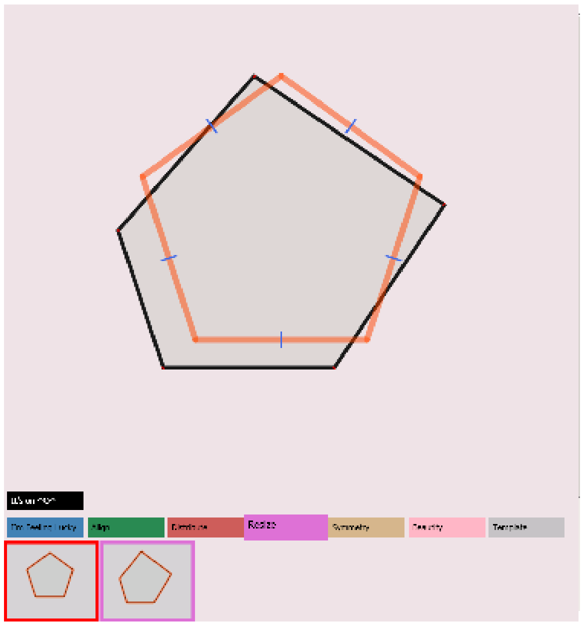


Figure 2.2.4: A pentagon is made regular.

Routing 5: The algorithm will try to find all the polygons with non-square bounding boxes and, then, make their bounding boxes square. (Figure 2.2.5)

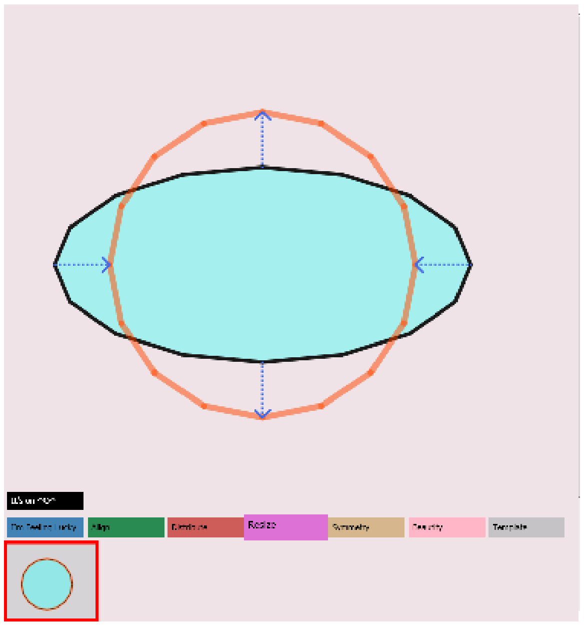


Figure 2.2.5: The bounding box of circle is made square.

In resizing operation, the first selected polygon is fixed, and any further selected polygons will be adjusted according it just like all other operations. However, there are some interesting cases and these are listed below.

Case 1: For a polygon that encloses another polygon, the inner one will be resized according to the outer one if the outer one is selected first. However, the outer polygon will be resized along with the inner one proportionally. Therefore, both polygons will be enlarged. Figure 2.2.6 corresponds to the situation where the outer polygon is selected first. Figure 2.2.7 shows how the polygons will be affected after the resizing operation. Figure 2.2.8 shows the final result.

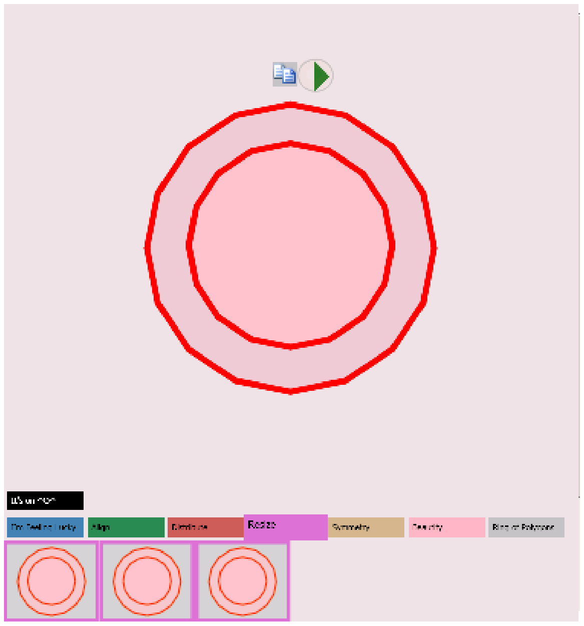


Figure 2.2.6: The outer circle is selected before the inner one

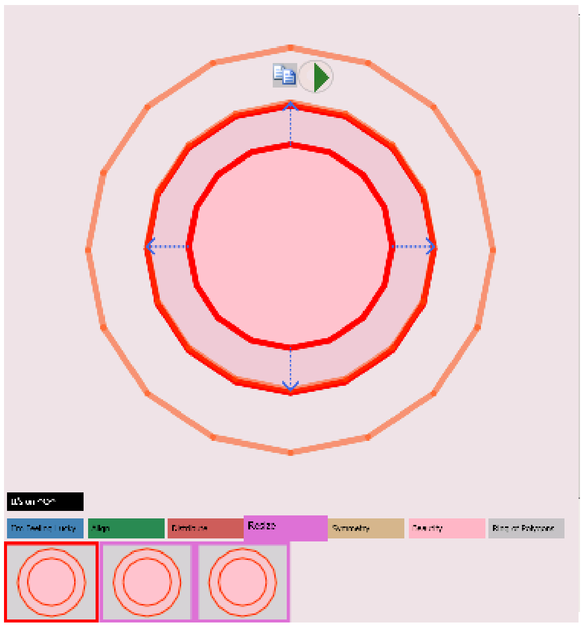


Figure 2.2.7: The inner circle is enlarged and, the outer one is enlarged along with the inner one.

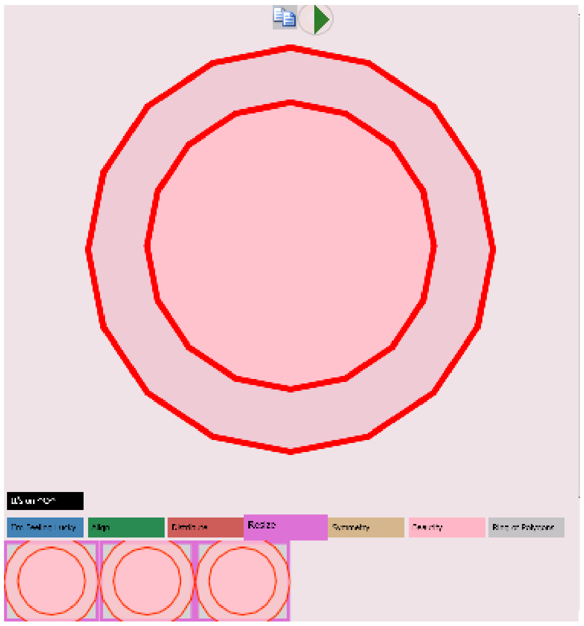


Figure 2.2.8: The result after resizing

Case 2: This is similar to case 1 except that the inner polygon is selected before the outer one. Consequently, the inner polygon will be resized along with the outer one proportionally. In contrast to case 1, both polygons are shrunk here.

Figure 2.2.9 and Figure 2.2.10 represents the situation where the inner polygon is selected before the outer one. Figure 2.2.11 shows how the polygons will be affected after the resizing operation. Figure 2.2.12 shows the final result.

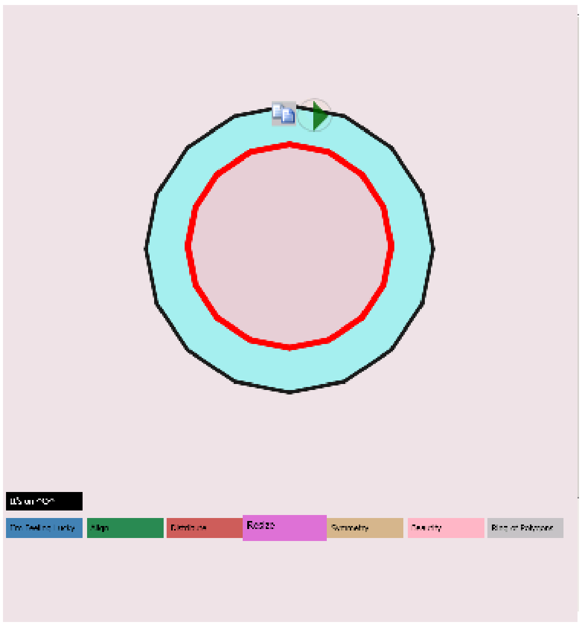


Figure 2.2.9: The inner circle is selected first

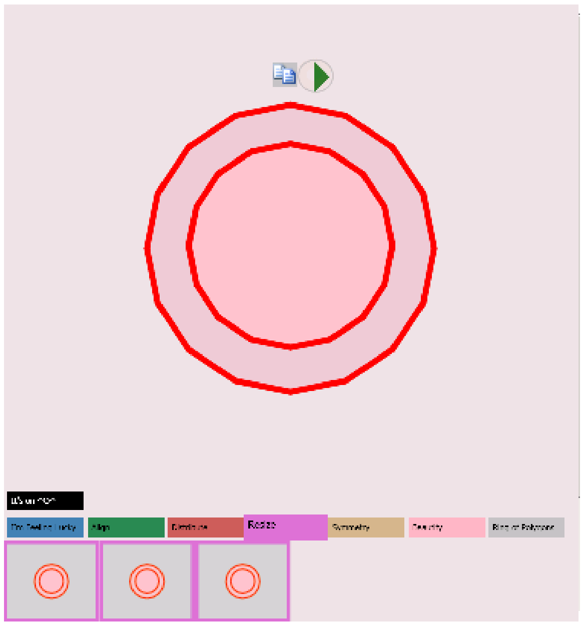


Figure 2.2.10: The outer circle is selected after selecting the inner one.

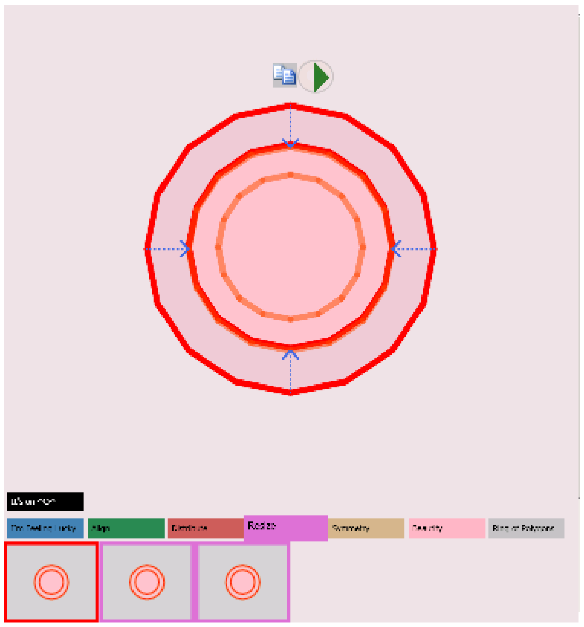


Figure 2.2.11: The outer circle is shrunk and, the inner one is shrunk along with the outer one.

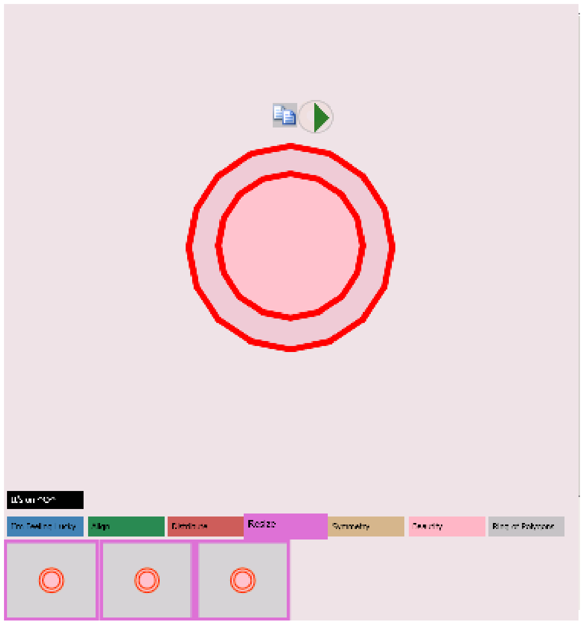


Figure 2.2.12: The result after resizing

Case 3:

Suppose there are three disjoint polygons as shown in Figure 2.2.13, where middle and right polygons have children, the polygons are selected in the following order: A1, B2, B1, C1, C2.

Since A1 is the first polygon selected, it is always fixed. B2 and C1 are the first selected within the group B and C. Therefore, B2 and C1 will be resized according to A1 such that their bounding boxes are identical. Furthermore, both B1 and C2 will be adjusted with respect to B2 and C1, as shown in Figure 2.2.14.

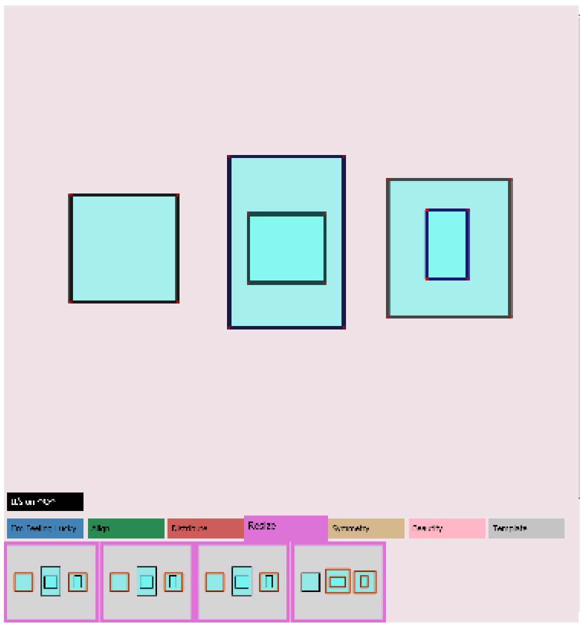


Figure 2.2.13: The polygons will be selected in the order: A1, B2, B1, C1, and C2, where
A1 is the polygon on the left,
B1 is the outer polygon in the middle,
B2 is the inner polygon in the middle,
C1 is the outer polygon on the right, and
C2 is the inner polygon on the right.

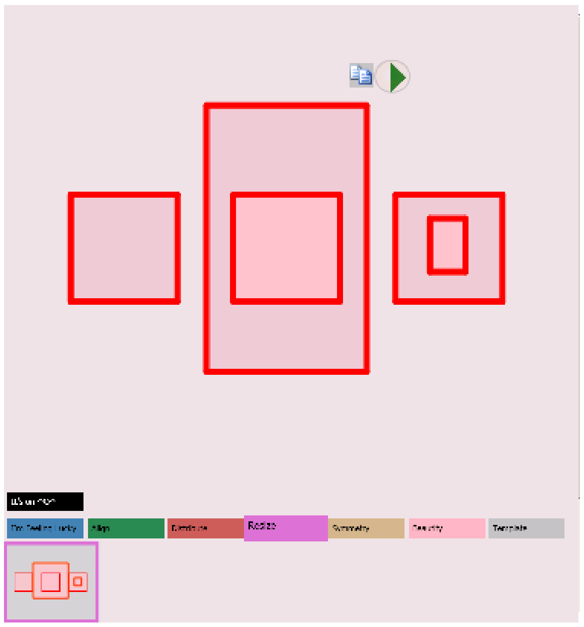


Figure 2.2.14: The result after resizing

3. Distribute

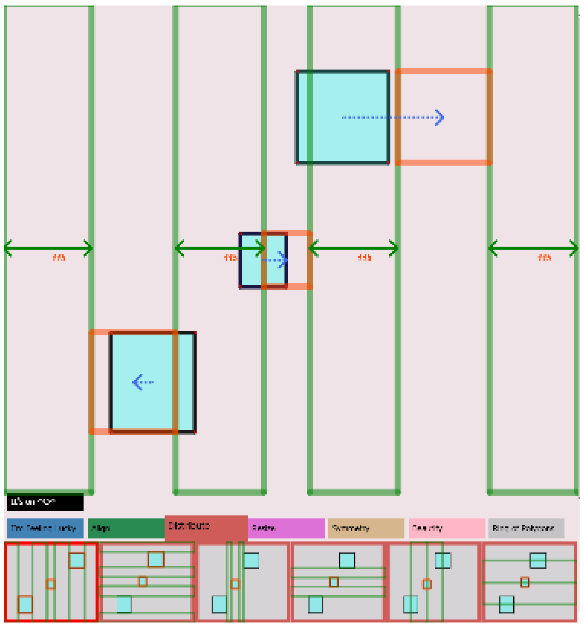
Most of the diagramming software provides distributing feature since this is frequently used. However, each has different user interface design.

Actually, there are not so many ways to distribute. Thus, the distributing operation should be kept simple. To achieve this, our approach is to show the most common result to the user beforehand.

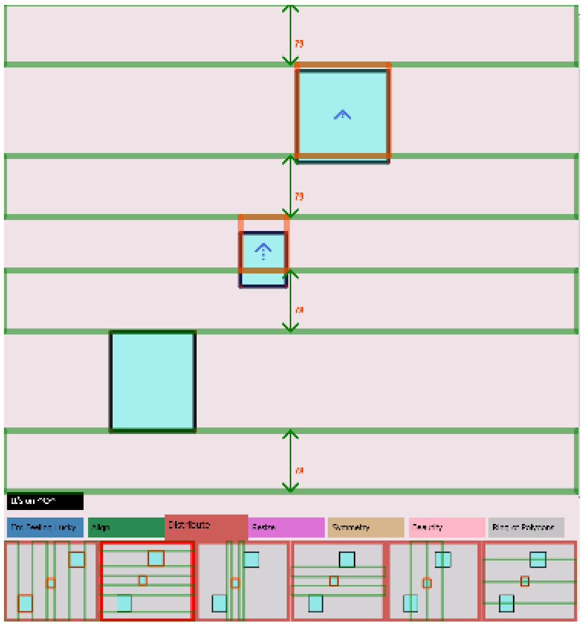
From the observation, most tasks that need distributing operation can be accomplished by 8 basic types of distribution. Therefore, it is possible for us to pre-compute the results for the user. Furthermore, the distribution operation in the new design only requires the users to choose a pre-computed option window by tapping. Compared with other software, this new distribution method significantly reduced the amount of time user spent in learning how to perform distribution.

The 8 types of distribution operations are described as follows.

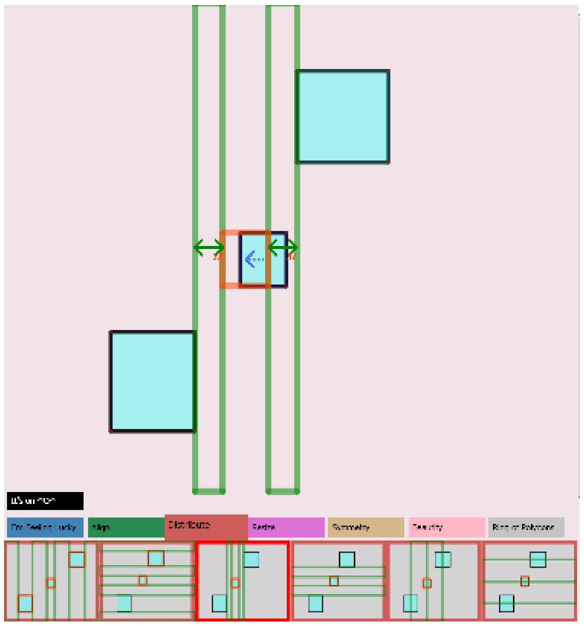
1. Horizontally distribute the polygons evenly between the left and right boundaries of the drawing canvas, where the intervals between the polygons are the same.



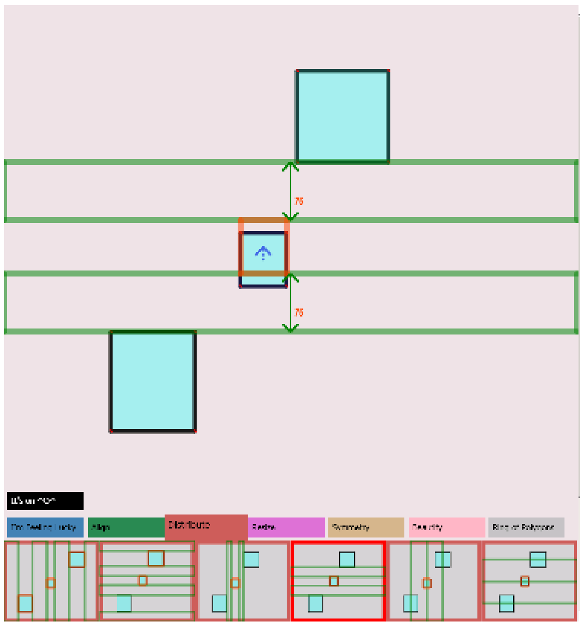
2. Vertically distribute the polygons evenly between the top and bottom boundaries of the drawing canvas, where the intervals between the polygons are the same.



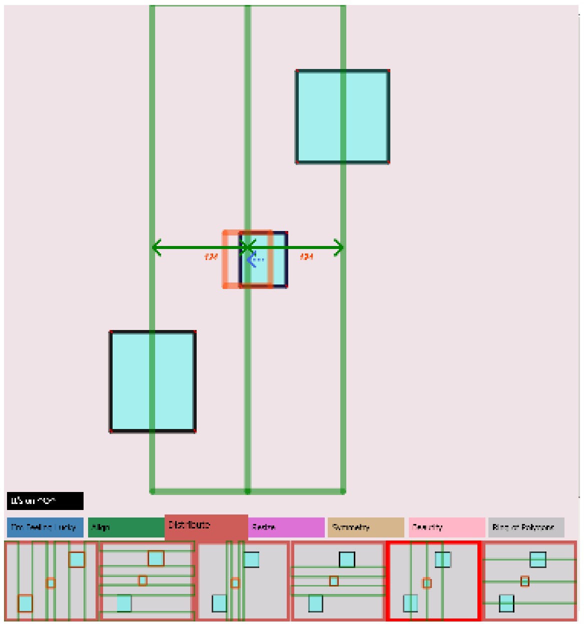
3. Horizontally distribute the polygons evenly between the left most polygon and the right most polygon, where the intervals between the polygons are the same.



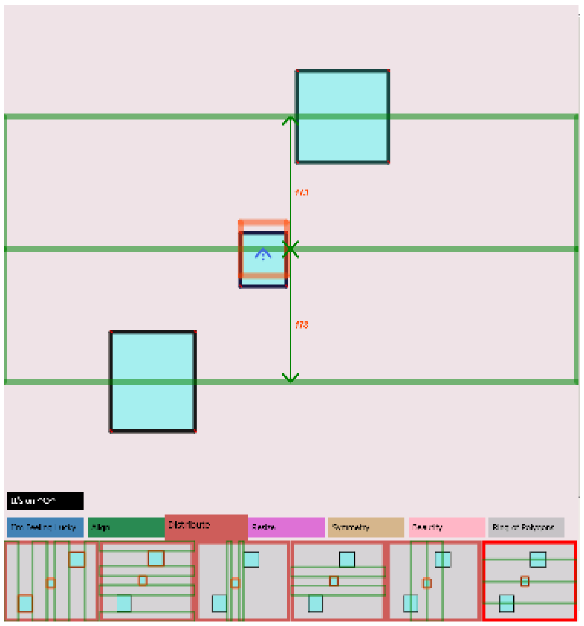
4. Vertically distribute the polygons evenly between the top most polygon and the bottom most polygon, where the intervals between the polygons are the same.



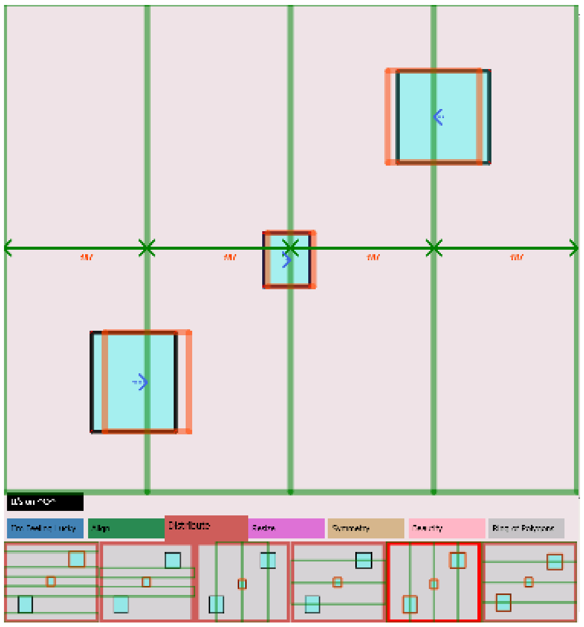
5. Horizontally distribute the polygons evenly between the left most polygon and the right most polygon, where the intervals between the centers of the polygons are the same.



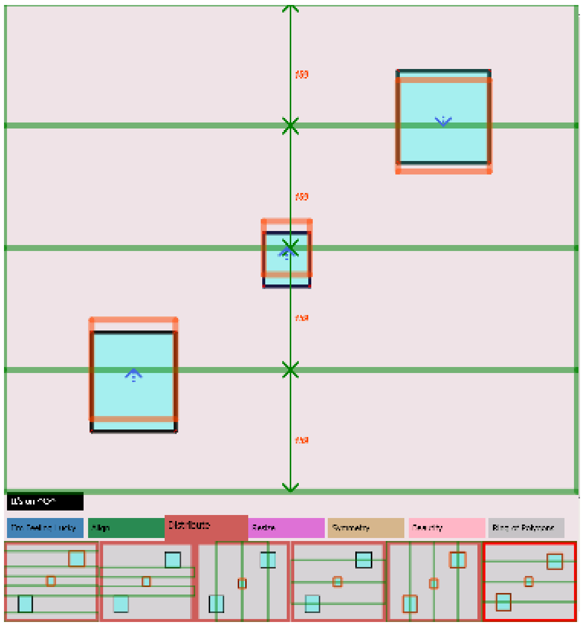
6. Vertically distribute the polygons evenly between the top most polygon and the bottom most polygon, where the intervals between the centers of the polygons are the same.



7. Horizontally distribute the polygons evenly between the left and right boundary of the drawing canvas, where the intervals between the centers of the polygons are the same.



8. Vertically distribute the polygons evenly between the top and bottom boundary of the drawing canvas, where the intervals between the centers of the polygons are the same



In addition, "Distribution" allows the user to have more control by selecting polygons. After selection, the algorithm will first try to find the smallest common parent polygon – a polygon that encloses all the selected.

If such a polygon exists, the algorithm will distribute all the selected polygons within this polygon. Otherwise, the selected polygon will be distributed in the entire drawing canvas.

Figure 2.3.1 gives an example of how selected polygons are distributed within the smallest common parent polygon.

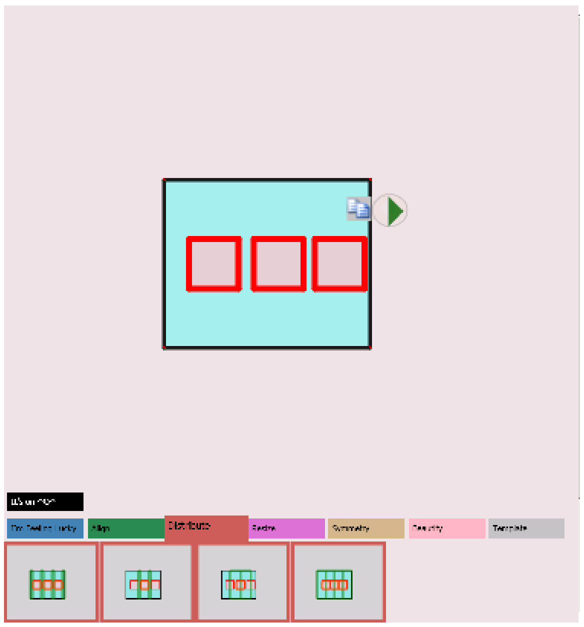


Figure 2.3.1: Distribution within a polygon

4. I'm Feeling Lucky

It is observed that transforming the polygons in Figure 2.4.1 to the polygons in Figure 2.4.2 is very time consuming. To make the process more efficient, it would be nice if the software can simplify some of the works. This is where "I'm Feeling Lucky" comes into play.

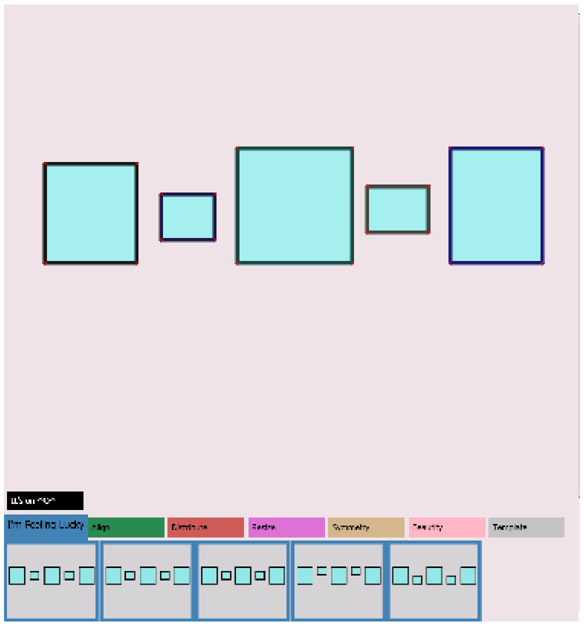


Figure 2.4.1

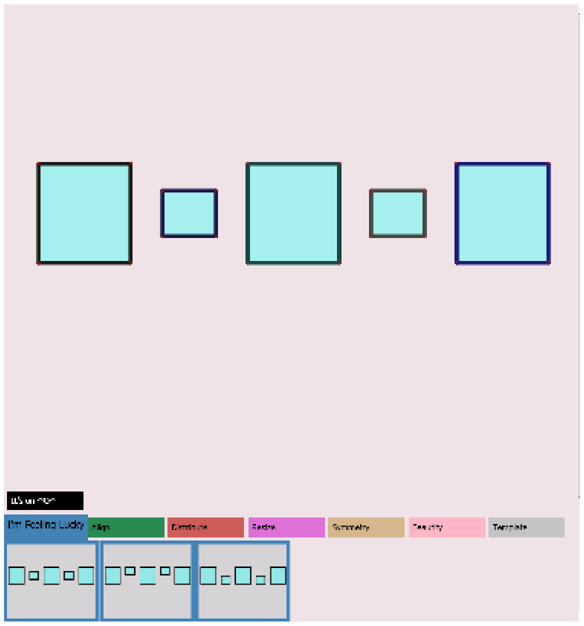


Figure 2.4.2

The purpose of "I'm Feeling Lucky" is to speed up tedious and repetitive works such as a series of alignment, resizing and distribution operations.

In short, "I'm Feeling Lucky" executes a number of operations such as resizing, distribution, and alignment in a sophisticated sequence to achieve the desired result.

There are 10 kinds of "I'm Feeling Lucky" in the current system but only six of them will be shown. With high probability, at least one option window will match the user's intention.

Below are the figures that briefly show how the current scene can be transformed into the alternative.

Figure 2.4.3 shows what the drawing canvas originally looks like.

Figures 2.4.3.1 to 2.4.3.6 show how the current drawing canvas is transformed after "I'm Feeling Lucky" is applied.

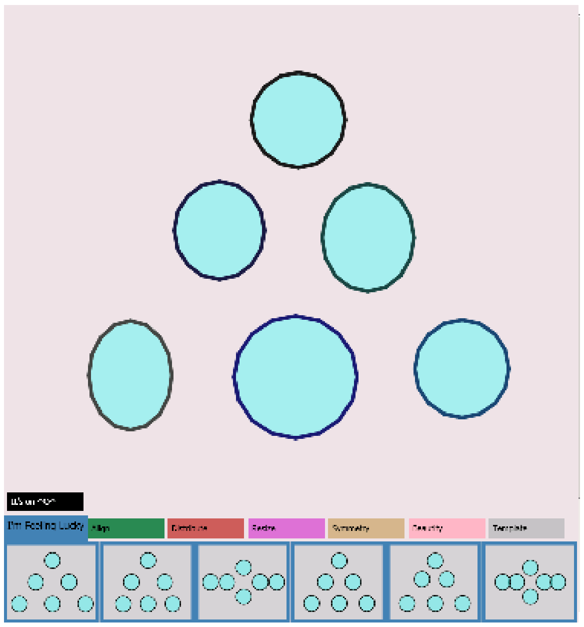


Figure 2.4.3: The original sketch from the users

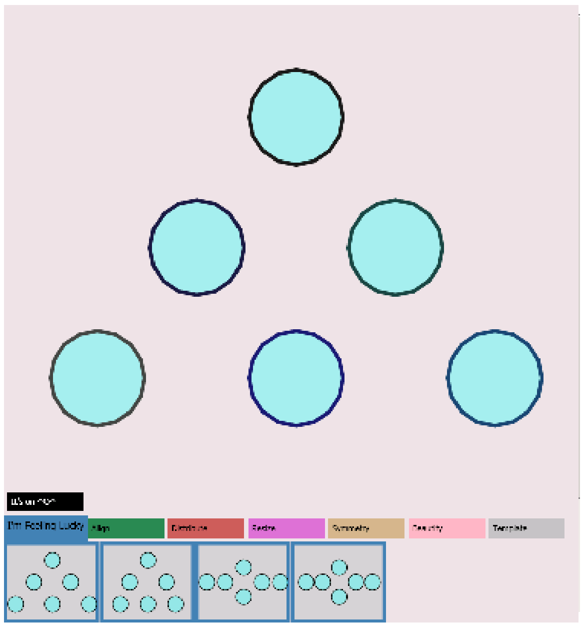


Figure 2.4.3.1: The result after tapping on option window 1

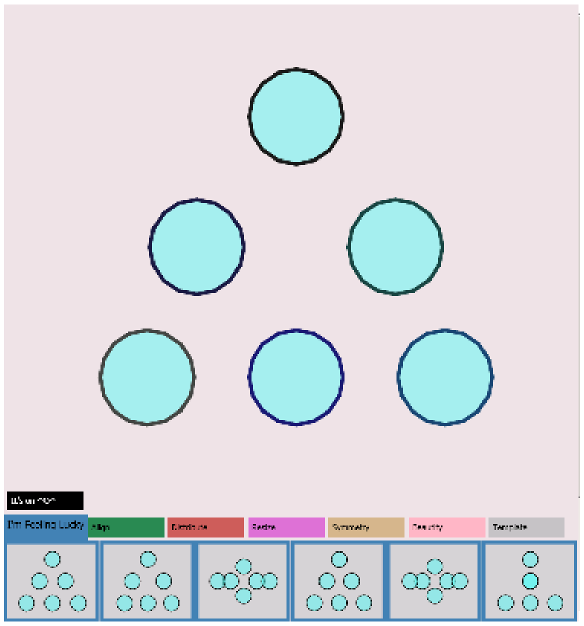


Figure 2.4.3.2: The result after tapping on option window 2

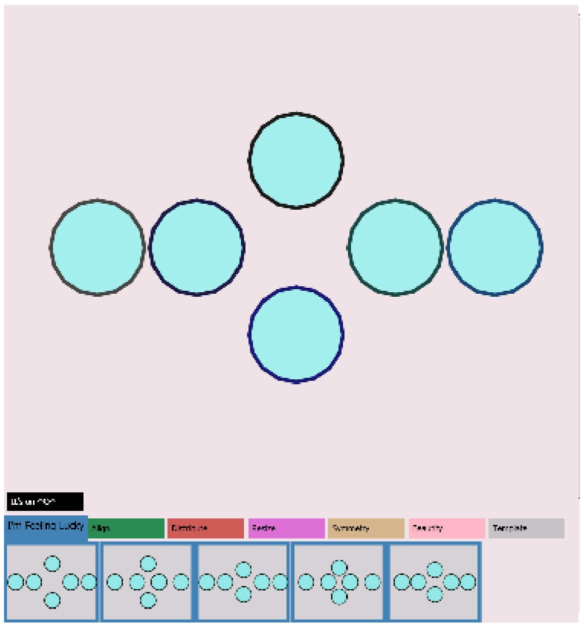


Figure 2.4.3.3: The result after tapping on option window 3

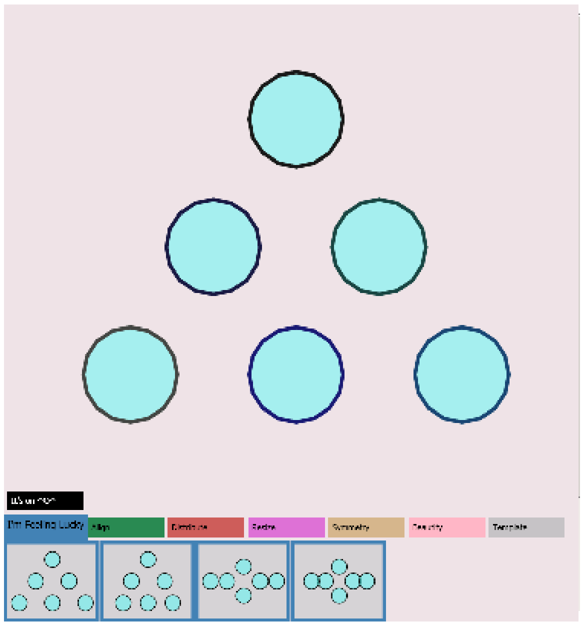


Figure 2.4.3.4: The result after tapping on option window 4

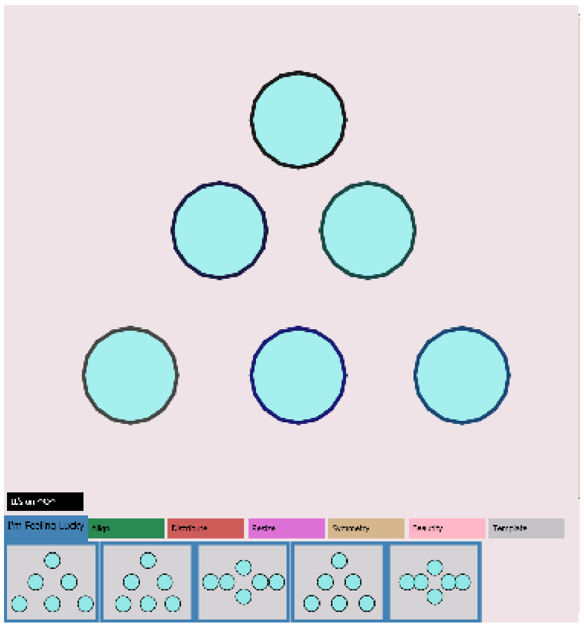


Figure 2.4.3.5: The result after tapping on option window 5

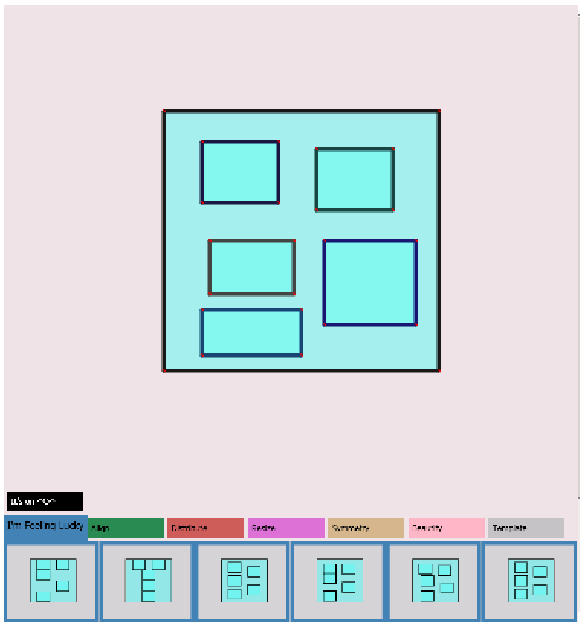


Figure 2.4.4: How the enclosed polygons look before "I'm Feeling Lucky"

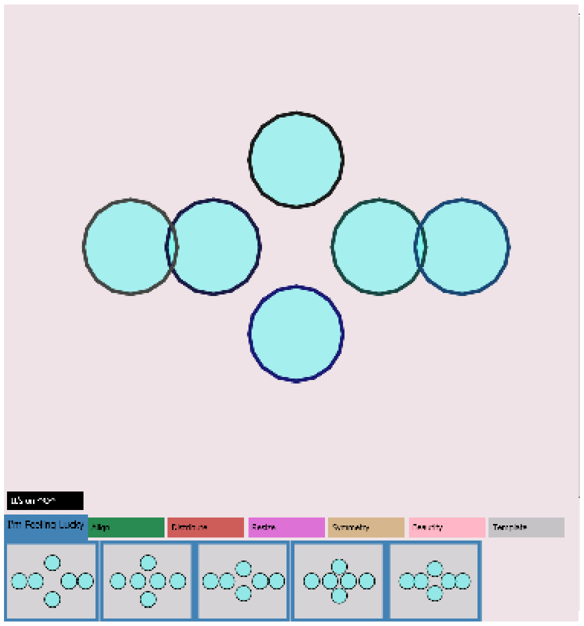


Figure 2.4.3.6: The result after tapping on option window 6

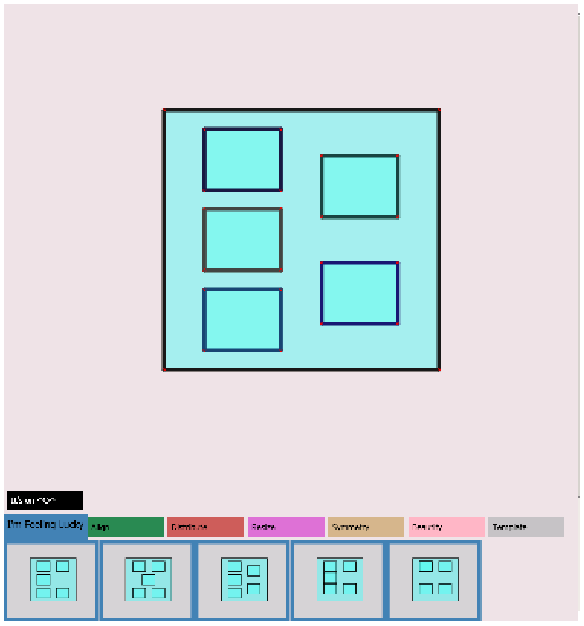


Figure 2.4.5: How the enclosed polygons look after "I'm Feeling Lucky"

Moreover, the "I'm Feeling Lucky" operation can also be applied to the polygons that are within another polygon. Figure 2.4.4 and Figure 2.4.5 show how the enclosed polygons look before and after the "I'm Feeling Lucky" is applied.

5. Template

One way or another, Lineogramer is able to provide most options in the SmartArt of Microsoft Office. However, cyclic structures shown in Figure 2.5.1 are difficult to achieve by Lineogrammer. Consequently, a new feature "Template" is added to handle this kind of task.

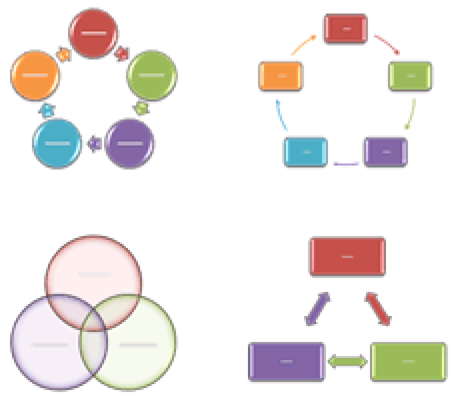


Figure 2.5.1: http://office.microsoft.com/en-us/help/HA100570651033.aspx

First, the algorithm will create a polygon by connecting all the centers of the polygons on the scene. Once this is obtained, we will check if this polygon is convex and non-regular. If not, nothing will be done. Otherwise, the algorithm will make it regular. As a result, all the polygons on the scene will be rearranged in such a way that their centers, now, form a regular polygon.

Furthermore, the algorithm will execute the following routines individually and the result generated by each routine will be shown in an option window.

For routine 1, the polygons remain unchanged (Figure 2.5.2).

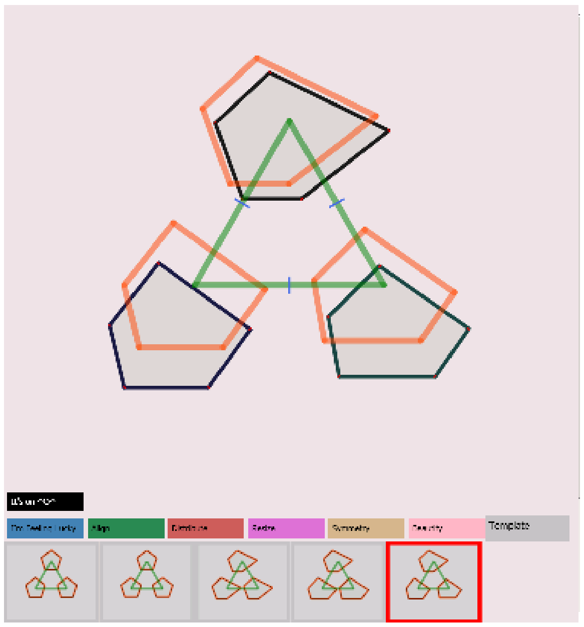


Figure 2.5.2: Option window 5 corresponding to routine 1

For routine 2, if the sizes of the bounding boxes of the polygons are similar, the bounding boxes will be made identical based on the polygon drawn first (Figure 2.5.3).

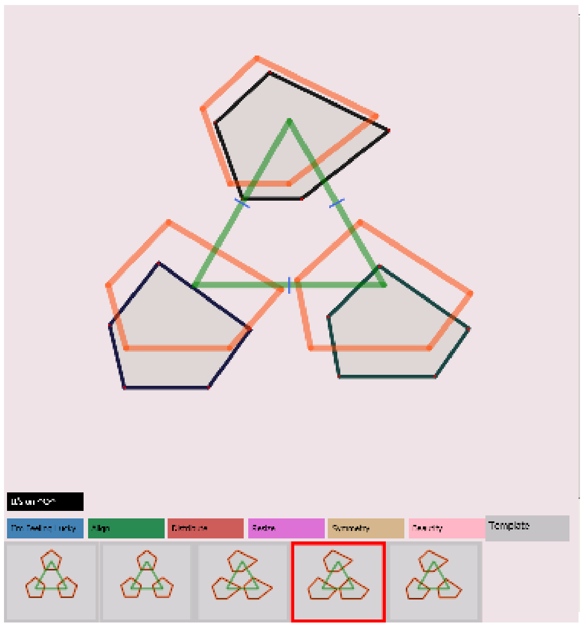


Figure 2.5.3: Option window 4 corresponding to routine 2

For routine 3, if the sizes of the bounding boxes of the polygons are similar, the polygons will be made identical based on the polygon drawn first (Figure 2.5.4).

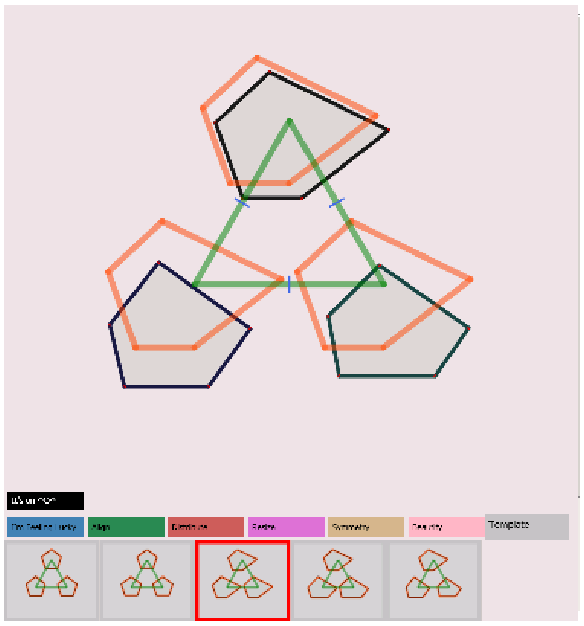


Figure 2.5.4: Option window 3 corresponding to routine 3

For routine 4, if the sizes of the bounding boxes of the polygons are similar, the polygons will be made regular based on the polygon drawn first (Figure 2.5.5).

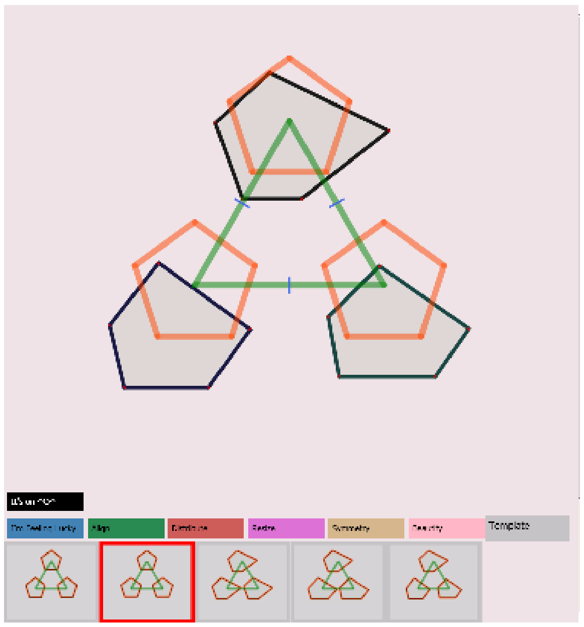


Figure 2.5.5: Option window 2 corresponding to routine 4

For routine 5, if the sizes of the bounding boxes of the polygons are similar, the polygons will be made regular and, then, their bounding boxes will be made square based on the polygon drawn first (Figure 2.5.6).

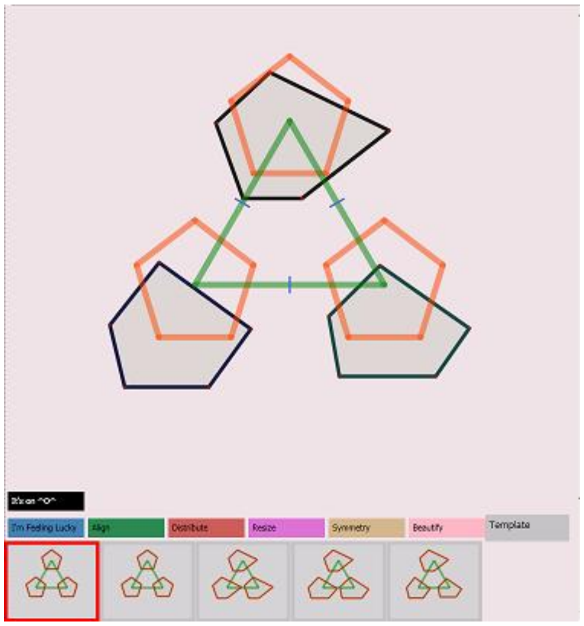


Figure 2.5.6: Option window 1 corresponding to routine 5

6. Beautify

Although sketching a polygon can be done quickly, the resulting polygon may not look professional. Hence, considerable amount of adjustments on the sketched polygon are probably still needed.

Here, the purpose of the "Beautify" operation is to cut down the amount of manual adjustments performed by the user. Even though "Beautify" operation may not always give what the user expects, it's likely that it will still provide the options that are close enough.

The algorithm of the "Beautify" operation is described as follows.

1. Make almost vertical/horizontal lines vertical/horizontal exactly.

2. Make vertices that have similar x/y coordinates the same.

3. Make edges that have similar lengths the same.

Note: When an arrowhead is recognized, it is processed separately.

Essentially, the option windows in "Beautify" show each intermediate step of the algorithm.

The following figures demonstrate how "Beautify" works on 3 different types of drawing. Figures 2.6.1 (stair shape), 2.6.2 (letter F), and 2.6.3 (double arrowhead) are the quick sketch made by the user. Figures 2.6.1.1., 2.6.2.1, and 2.6.3.1 show what Beautify is about to do. Figures 2.6.1.2, 2.6.2.2, and 2.6.3.2 show the final result generated.

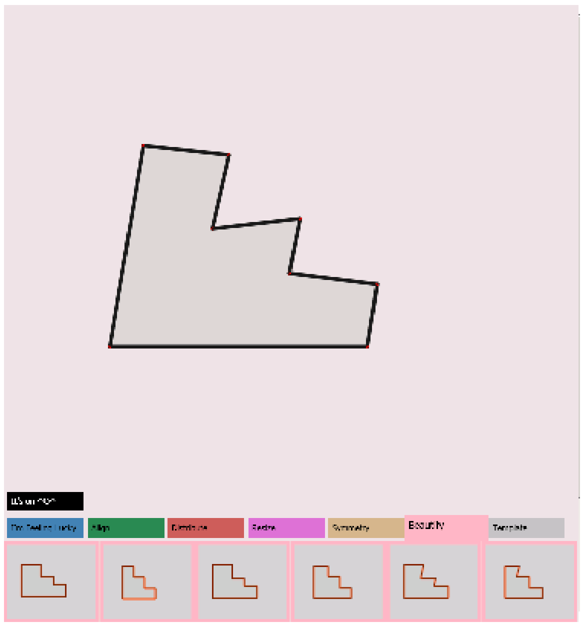


Figure 2.6.1: The quick sketch of a stair shape

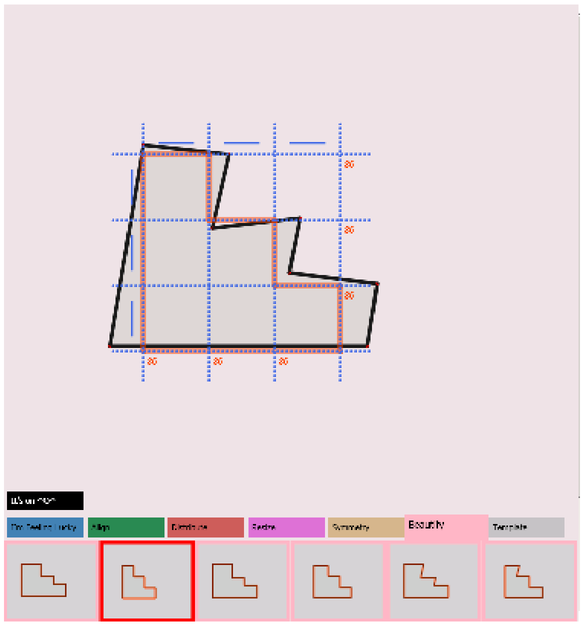


Figure 2.6.1.1: Showing the intended beautification of option window 2

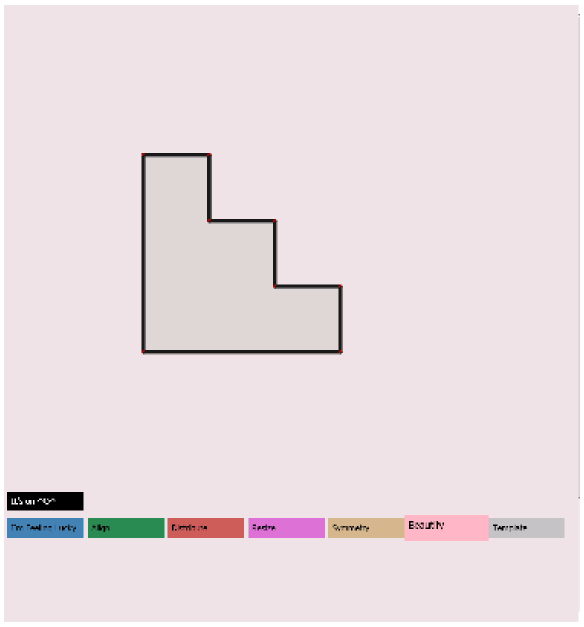


Figure 2.6.1.2: The beautified stair shape

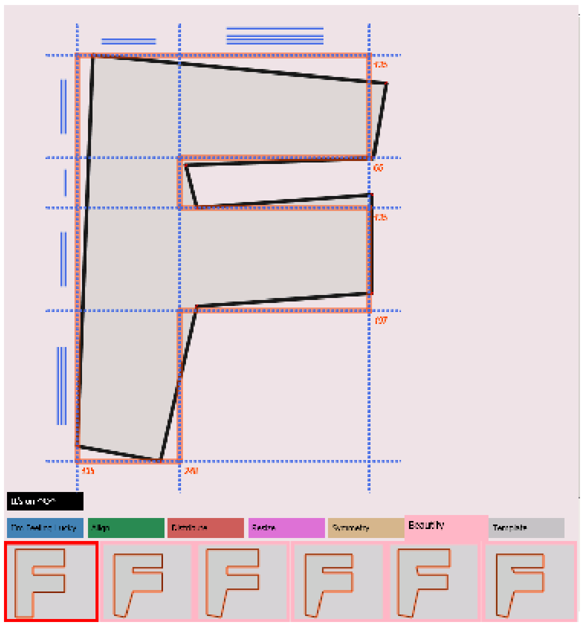


Figure 2.6.2.1: Showing the intended beautification of option window 1

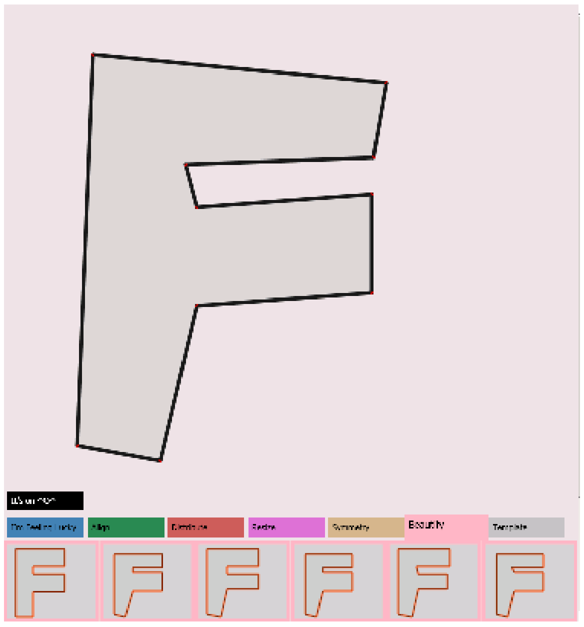


Figure 2.6.2: The quick sketch of a letter F

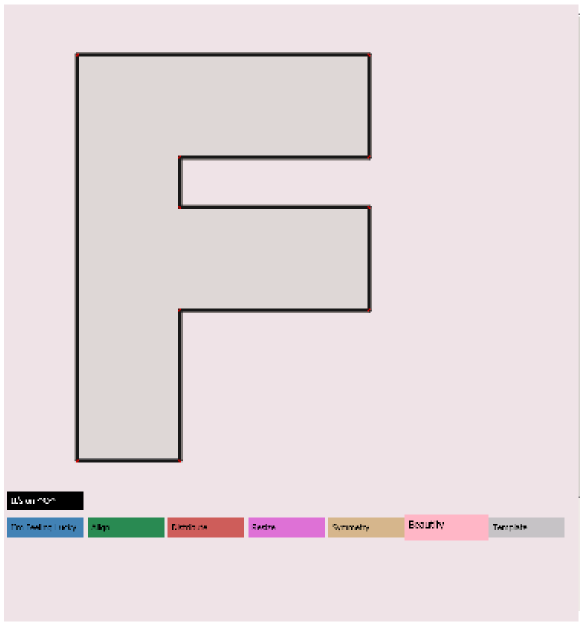


Figure 2.6.2.2: The beautified letter F

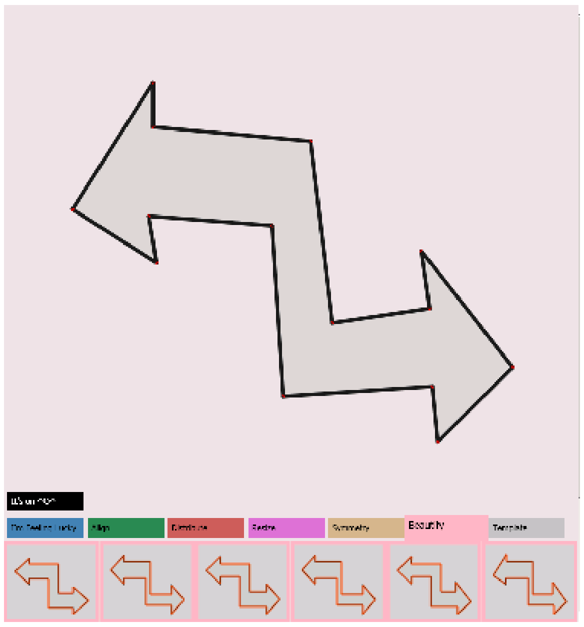


Figure 2.6.3: The quick sketch of a double arrowhead

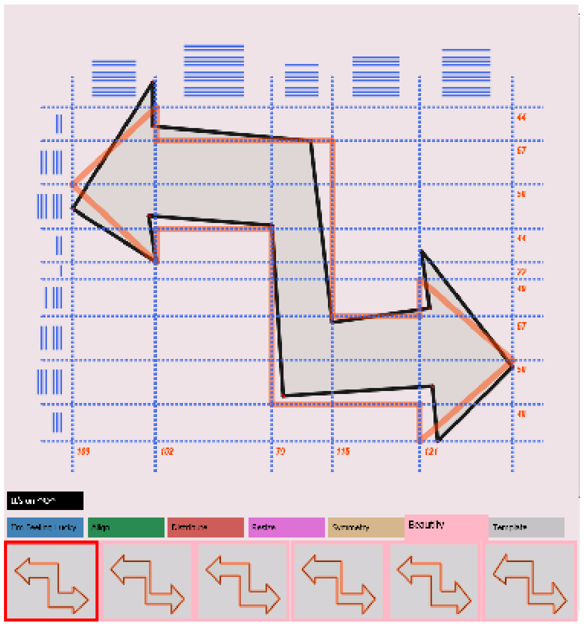


Figure 2.6.3.1: Showing the intended beautification of option window 1

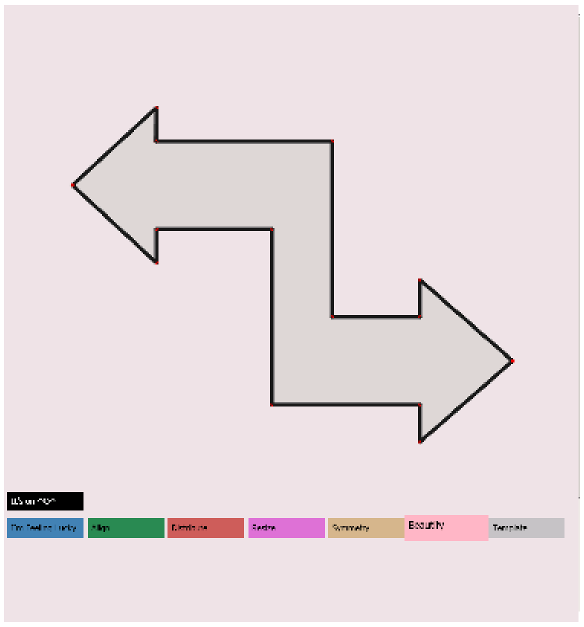


Figure 2.6.3.2: The beautified double arrowhead

7. Symmetry

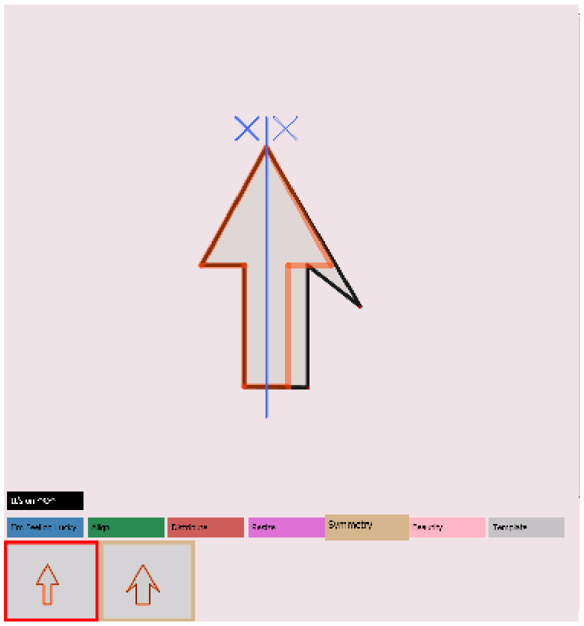
"Symmetry" is a special kind of "Beautify". Both of them try to reduce the amount of manual manipulations done by users, but what differentiates "Symmetry" from "Beautify" is that "Symmetry" will only explore all kinds of applicable symmetries.

There are 8 kinds of symmetries in total. However, only the first six applicable symmetries will be provided. The symmetry option windows are ordered in such a way that the most common type of symmetry will be presented to the user first.

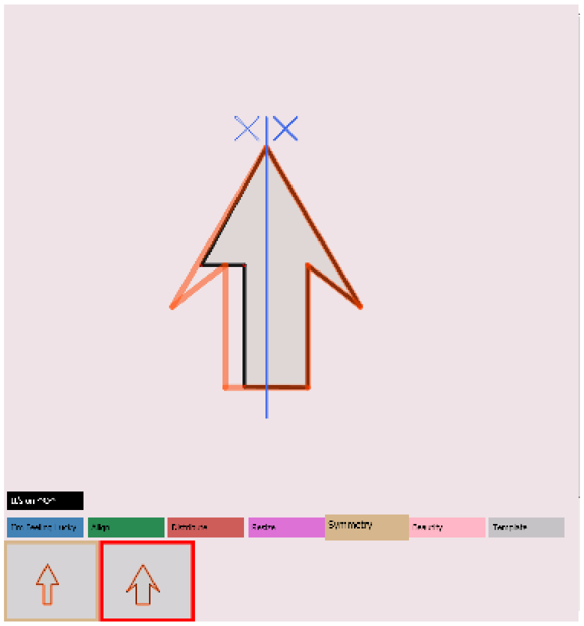
Here is a list of symmetries provided: "symmetry to the right", "symmetry to the left", "symmetry to the top", "symmetry to the bottom", "rotational symmetry to the right", "rotational symmetry to the left", "rotational symmetry to the top", and "rotational symmetry to the bottom".

The following figures illustrate how the eight types of symmetries are performed. Here, the blue line is the symmetry axis, the blue solid cross indicates the side that is fixed, the blue dashed cross indicates the side that is modified, and the orange polygon shows the resulting shape.

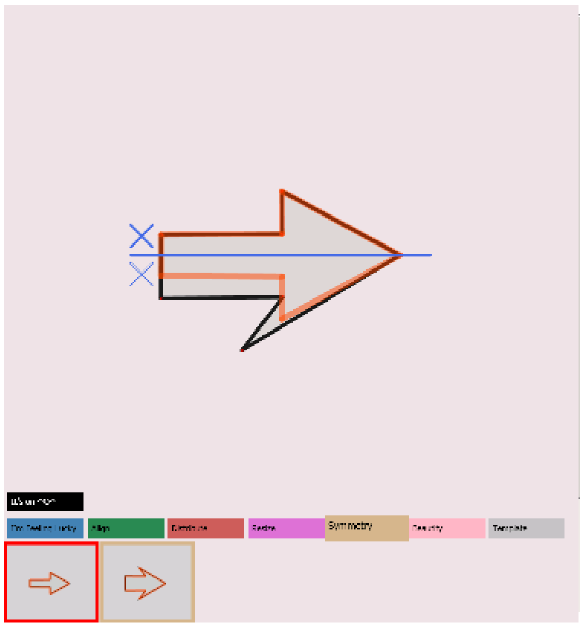
1. Symmetry to the left



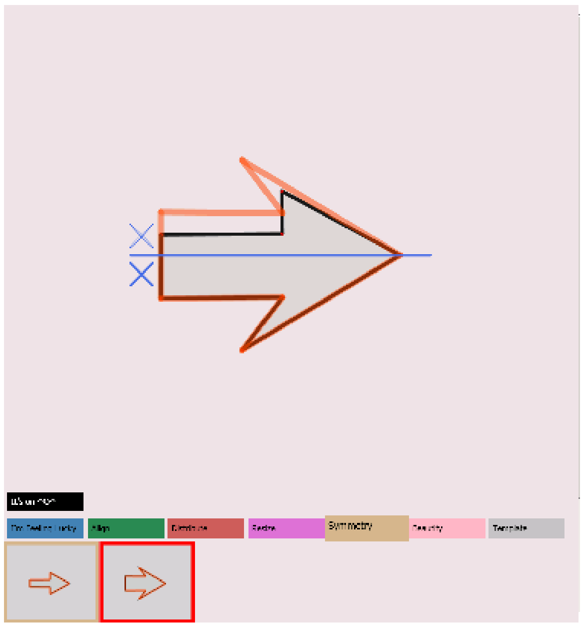
2. Symmetry to the right



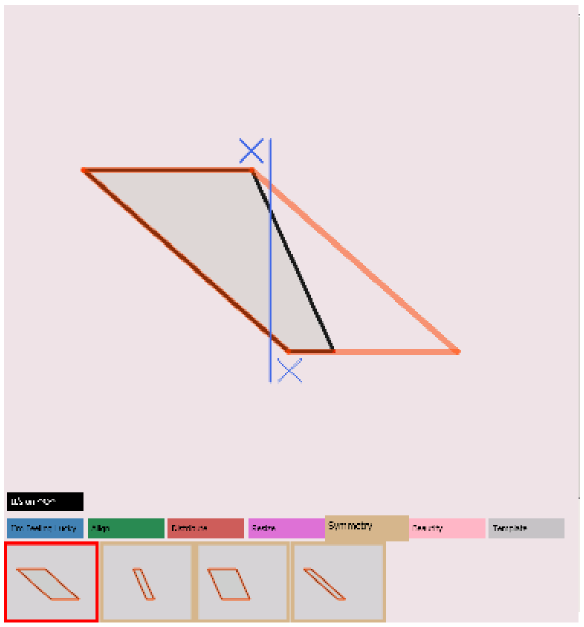
3. Symmetry to the top



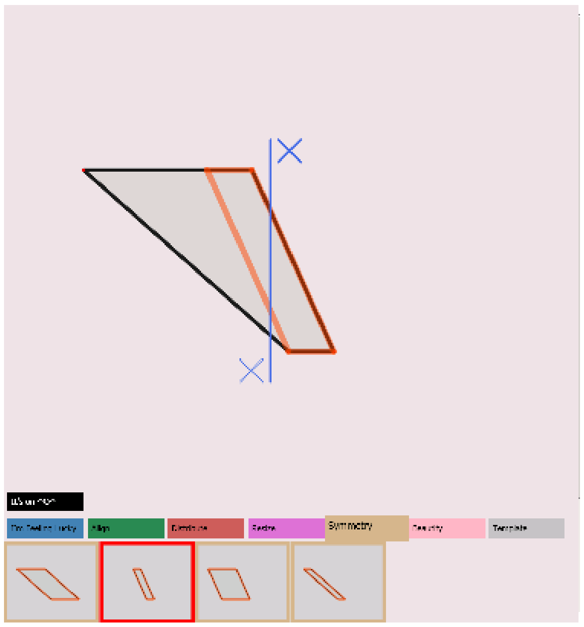
4. Symmetry to the bottom



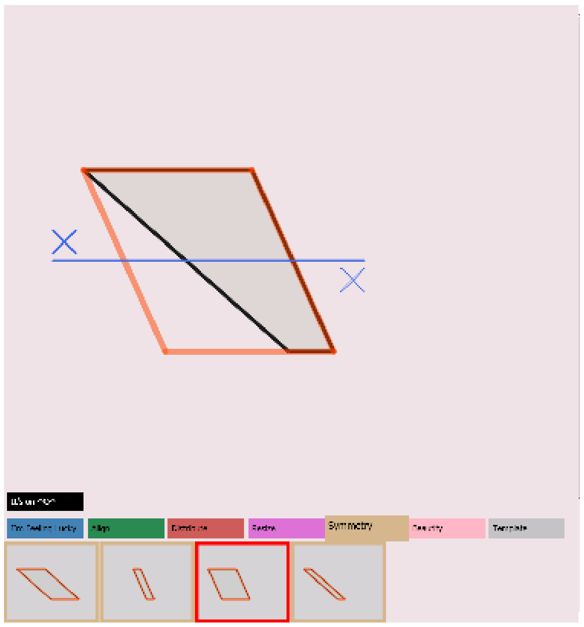
5. Rotational symmetry to the left



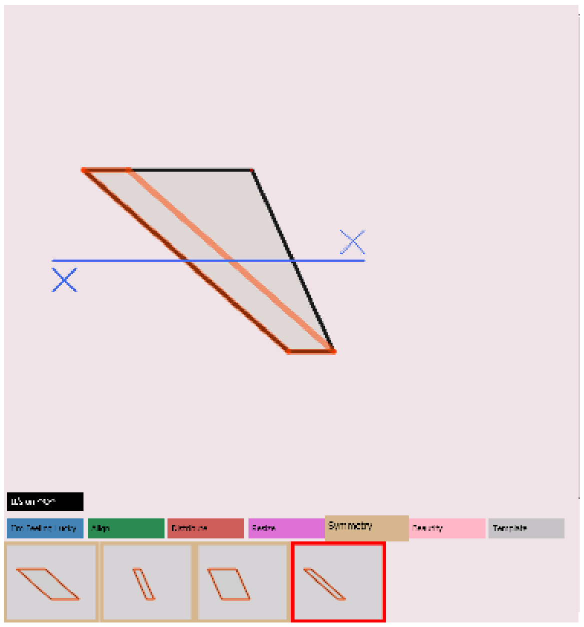
6. Rotational symmetry to the right



7. Rotational symmetry to the top



8. Rotational symmetry to the bottom


I'm Feeling Lucky
Align
Distribute
Symmetry
Template