Resizing, Translating and Rotating Shapes in Excel

by George Lungu

<excelunusual.com>

Resizing Shapes

The distance between two points of coordinates (x_A,y_A) and (x_B,y_B) can be increased by a factor of "n" through multiplication of all its coordinates by "n":

$$\boxed{Dist(A,B)=\sqrt{(x_A-x_B)^2+(y_A-y_B)^2}}$$

After resizing by a factor of "n" the distance between the points becomes:

$$Dist(A',B')=\sqrt{(x'_A-x'_B)^2+(y'_A-y'_B)^2}=$$

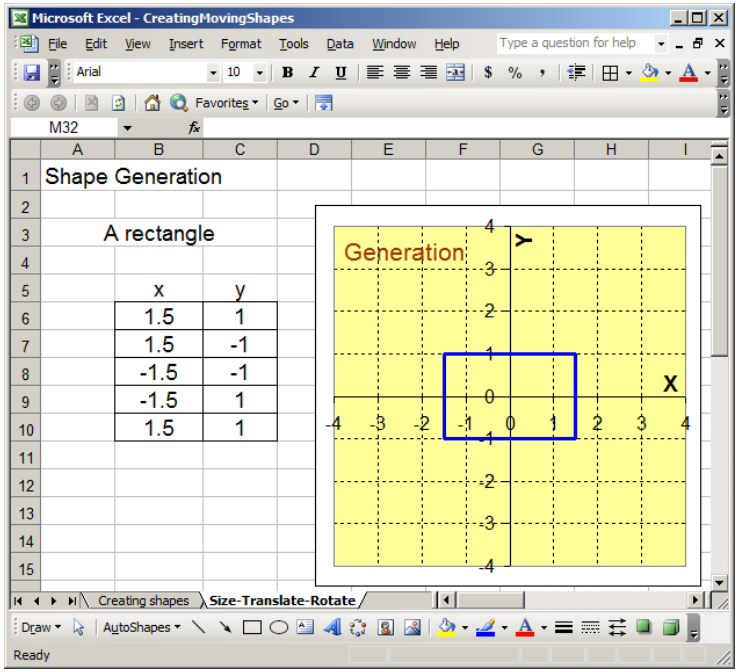
$$=\sqrt{(n\cdot x_A-n\cdot x_B)^2+(n\cdot y_A-n\cdot y_B)^2}$$

$$=|n|\cdot\sqrt{(x_A-x_B)^2+(y_A-y_B)^2}=|n|\cdot Dist(A,B)$$

More general, resizing can also be performed by separate factors on each axis independently as we can see in the following example:

Shape generation:

- Insert a Worksheet named: “Size_Translate_Rotate”
- Create a 2x1 rectangle centered in origin
- Display the rectangle on a 2D scatter plot with axes sized from -4 to 4



Shape resizing buttons:

- The resize factors will be contained in cells:

“C20” for *X-dimension* and

“C23” for *Y-dimension* using the following macros:

```
Private Sub Size_Y_Change()
    Range("C23") = Size_Y.Value / 10
End Sub
```

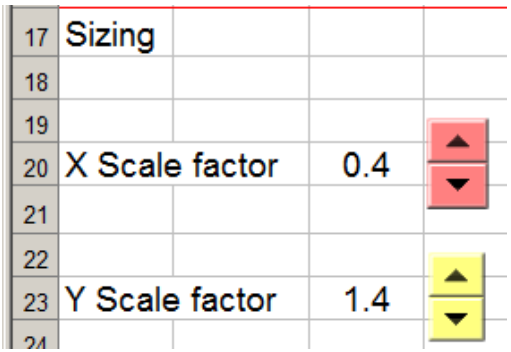
```
Private Sub Size_X_Change()
    Range("C20") = Size_X.Value / 10
End Sub
```

Button properties:

The names of the buttons are “*Size_X*” and “*Size_Y*”

I arbitrarily changed their colors to red and yellow respectively

Min=0, Max=20 for both buttons



Shape resizing – continuation

Create new data for the resized rectangle:

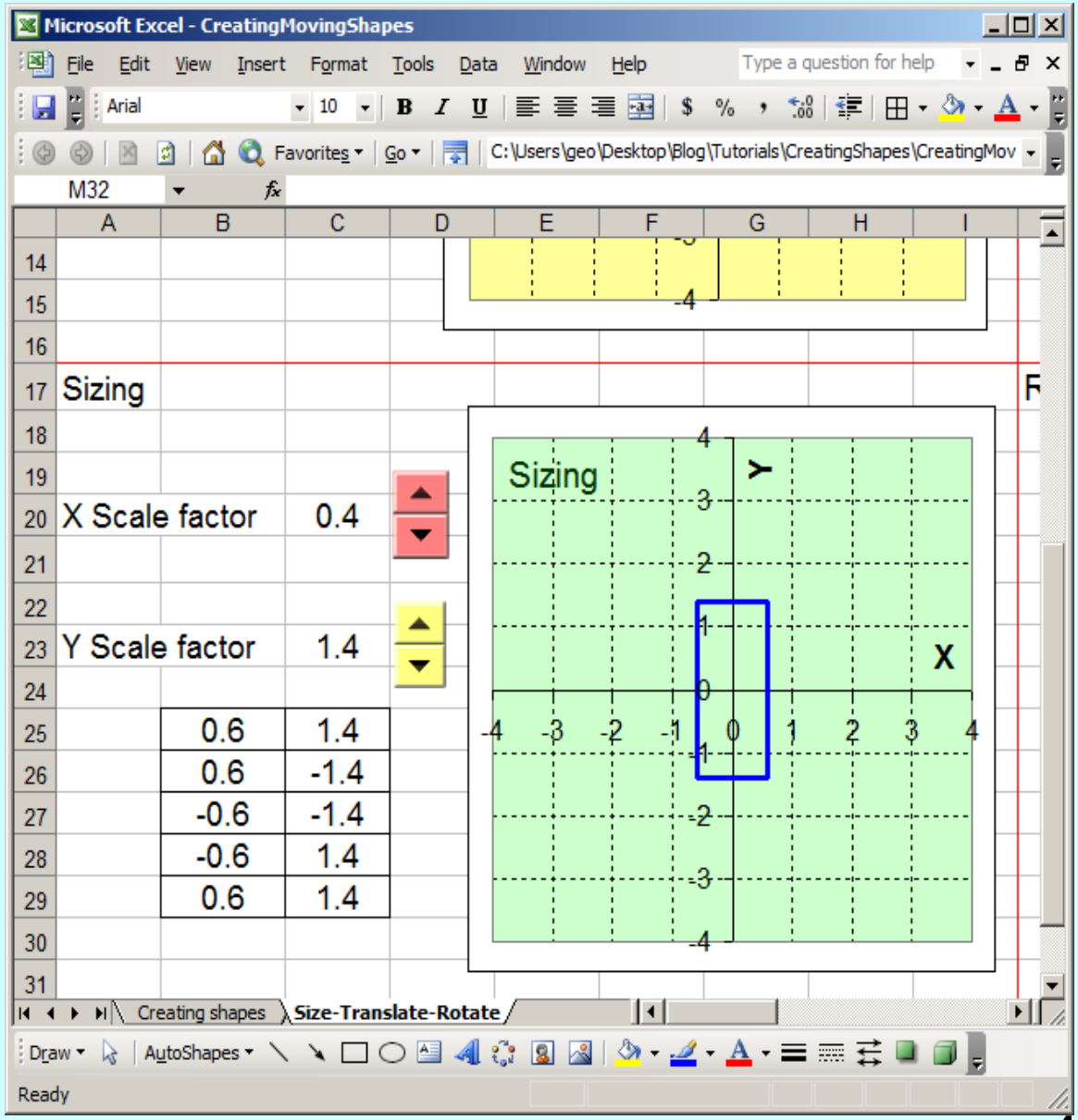
- Cell B25: “=C$20*B6”
- Copy B25 down to B29
- Cell C25: “=C$23*C6”
- Copy C25 down to C29

Plot the shifted rectangle:

Use the data in the area “B25:C29” to create a plot of the resized rectangle (the green chart in the picture)

Verify the functionality:

Click two spin buttons and verify that the rectangle is indeed scaled independen-tly on either X axis or Y axis

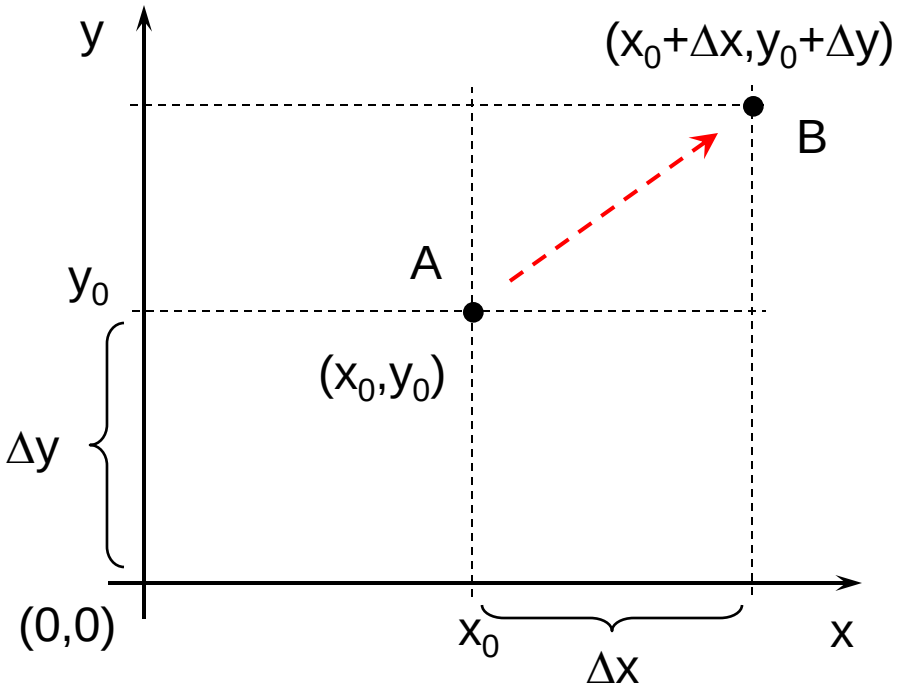


Moving shapes (spatial translation)

- A shape can be moved along an axis by adding a certain increment to the respective coordinate.
- For instance if an object is located at the coordinate (x_0,y_0) by adding Δx to x_o we can move the object along the x axis by Δx (which can be either positive or negative).
- Similarly if an object is located at the coordinate (x_0,y_0) by adding Δy to y_o we can move the object along the y axis by Δy (which can be either positive or negative).
- We can always move an object in any direction from point A to point B by changing to x_o to y_o accordingly

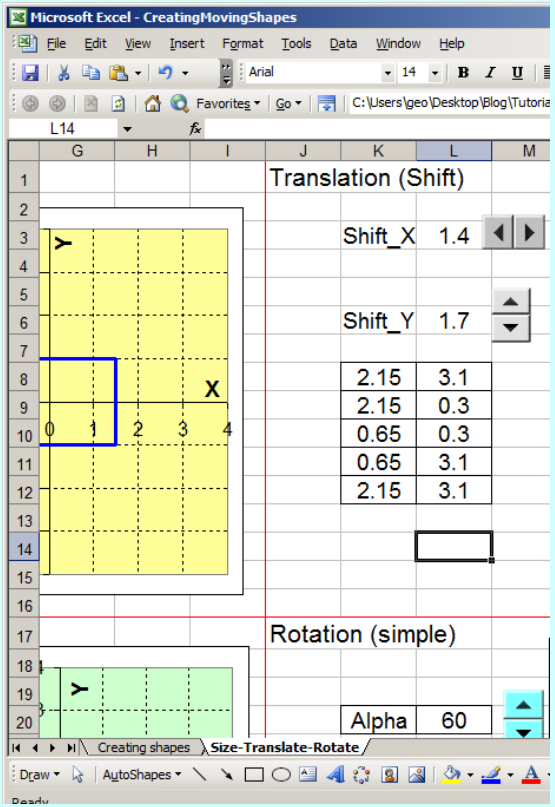
Translation

$$(x_0, y_0) \rightarrow (x_0 + \Delta x, y_0 + \Delta y)$$



Translation of the previously sized rectangle:

- Create a new table with the translated rectangle data:
- Cell K8: “=B25+$L$3”
- Copy K8 down to K12
- Cell L8: “=C25+$L$6”
- Copy L8 down to L12



Shape resizing buttons:

- The resize factors will be contained in cells:

“L3” for *X-dimension* and

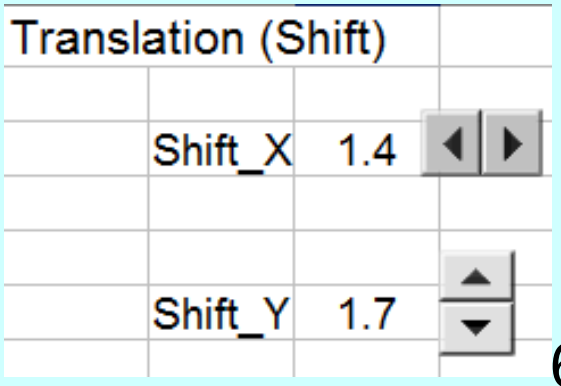
“L6” for *Y-dimension* using the following macros:

```
Private Sub Shift_X_Change()
    Range("L3") = Shift_X.Value / 10
End Sub
```

```
Private Sub Shift_Y_Change()
    Range("L6") = Shift_Y.Value / 10
End Sub
```

Button properties:

- The names of the buttons are “*Shift_X*” and “*Shift_Y*”
  - I arbitrarily changed their colors to grey
  - Min=0, Max=20 for both buttons



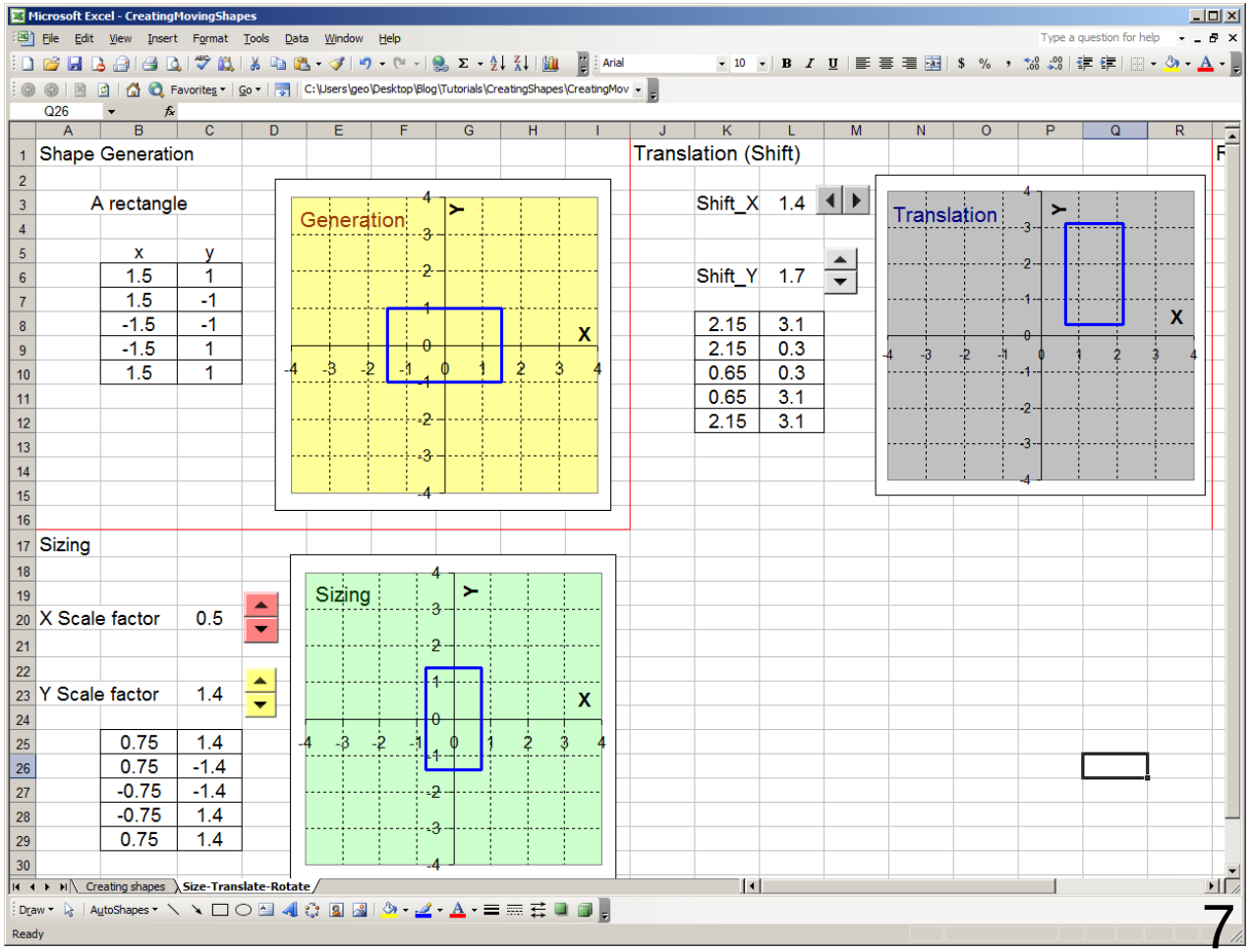
Translation - continuation

Plot the shifted rectangle:

- Use the data in the area “K8:L12” to create a plot of the resized rectangle (the grey chart in the picture)

Verify the functionality:

- Click the two spin buttons and verify that the rectangle is indeed translated independently on either X axis or Y axis

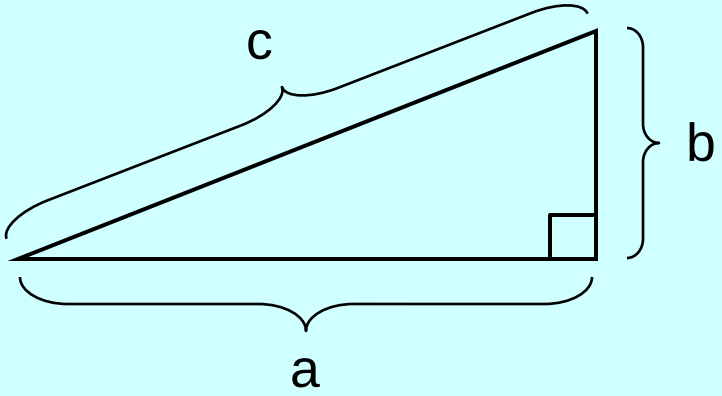


Rotating shapes

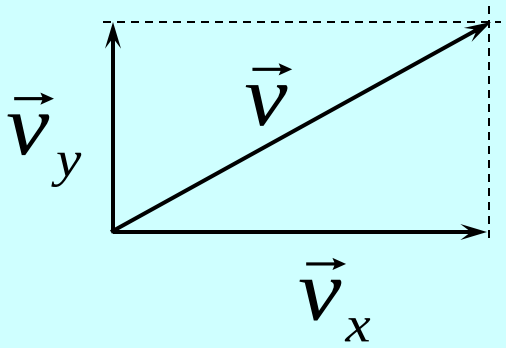
Let's review the definitions of two basic trigonometric functions on a ***right triangle***:

$$Sin(\alpha) = \frac{b}{c}$$

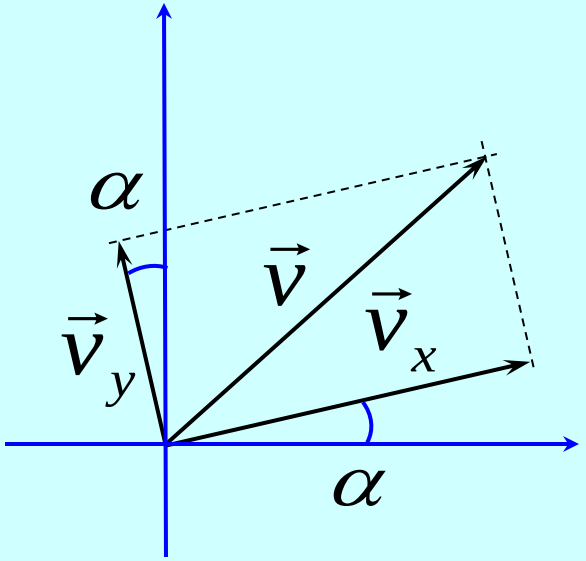
$$\cos(\alpha) = \frac{a}{c}$$



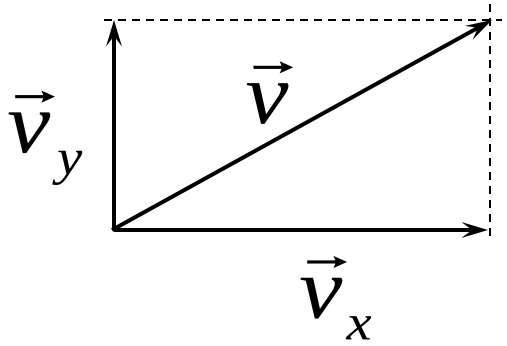
Let's take a vector with its (x,y) components:



and let's rotate it by angle α:



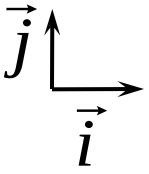
Before the rotation, $\vec{v}_x$ and $\vec{v}_y$ were parallel with the x and y axes respectively and could be expressed function of the axes unit vectors $\vec{i}$, $\vec{j}$:



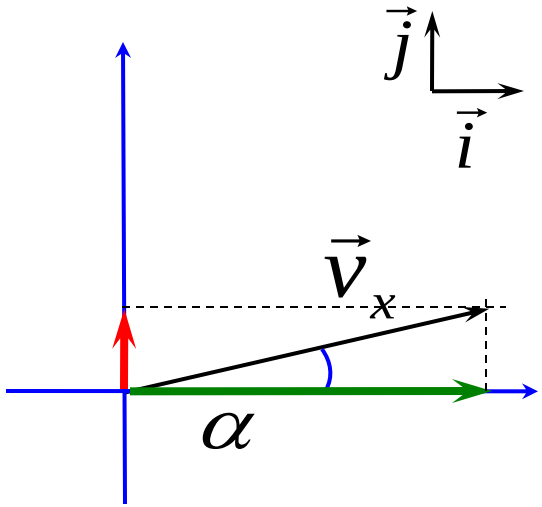
$$\vec{v}_x = \vec{i} \cdot v_x$$

$$\vec{v}_y = \vec{j} \cdot v_y$$

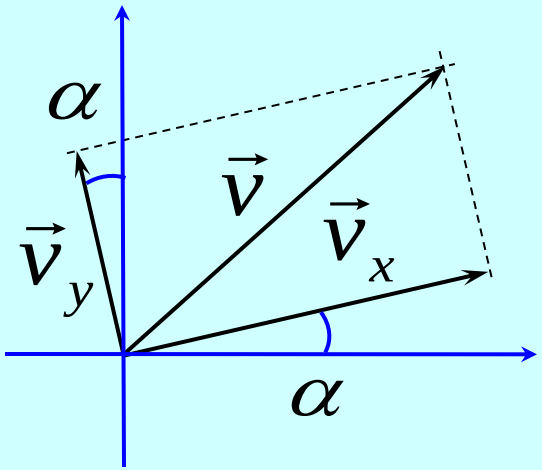
Where $\vec{i}$ and $\vec{j}$ are the x and y unit vectors respectively

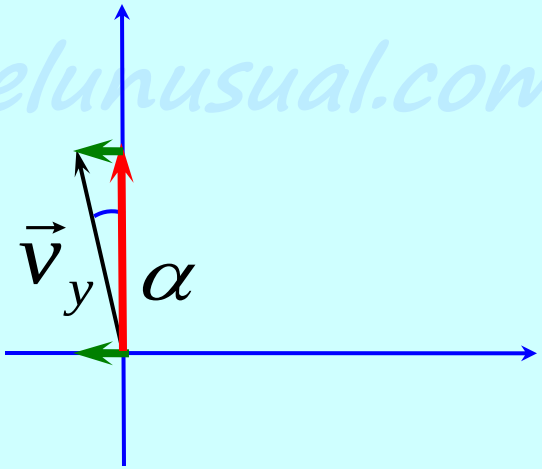


After the rotation, $\vec{v}_x$ and $\vec{v}_y$ are no longer parallel with the x and y axes respectively and can be expressed function of the axes unit vectors $\vec{i}$, $\vec{j}$:



$$\vec{v}_x = \underbrace{\vec{i} \cdot \cos(\alpha) \cdot v_x}_{} + \underbrace{\vec{j} \cdot \sin(\alpha) \cdot v_x}_{}$$





$$\vec{v}_y = \underbrace{-\vec{i}\cdot\sin(\alpha)\cdot v_y}_{} + \underbrace{\vec{j}\cdot\cos(\alpha)\cdot v_y}_{}$$

We can express $\vec{v}$ as the vector sum of $\vec{v}_x$ and $\vec{v}_y$

$$\vec{v} = \vec{v}_x + \vec{v}_y =$$

$$= \vec{i}\cdot\cos(\alpha)\cdot v_x + \vec{j}\cdot\sin(\alpha)\cdot v_x - \vec{i}\cdot\sin(\alpha)\cdot v_y + \vec{j}\cdot\cos(\alpha)\cdot v_y$$

Grouping around $\vec{i}$ and $\vec{j}$ results in:

$$\vec{v} = \vec{i}\cdot[\cos(\alpha)\cdot v_x - \sin(\alpha)\cdot v_y] + \vec{j}\cdot[\sin(\alpha)\cdot v_x + \cdot\cos(\alpha)\cdot v_y]$$

Calling v_x' and v_y' the new coordinates of the rotated point around origin we obtain the following formulas:

$$\begin{cases} v_x' = \cos(\alpha)\cdot v_x - \sin(\alpha)\cdot v_y \\ v_y' = \sin(\alpha)\cdot v_x + \cos(\alpha)\cdot v_y \end{cases}$$

(where a is the rotation angle and v_x and v_y are the original coordinates before the rotation)

Don't memorize but be able to derive this at any time!

Sometimes people like to put it in matrix form but I prefer the previous expression:

$$\begin{pmatrix} v_x' \\ v_y' \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

There are two ways in which we can use the rotation in practice:

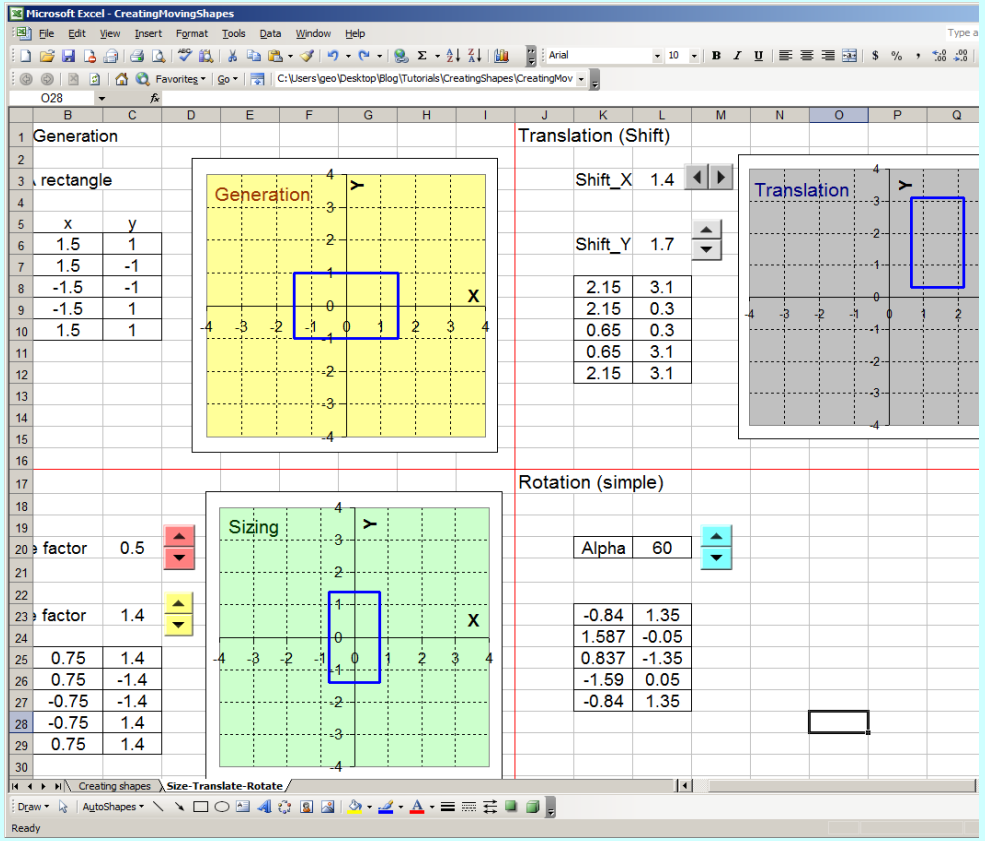
1. Rotate the object around it's own center (keeping the center of the object at the same coordinate)

2. Rotate the whole scene around a "scene pole"

see next page =>

Case#1: Simple Rotation:

- Create a new table with the rotated rectangle data:
- Cell K23: “=B25*COS(RADIANS(L$20))-C25*SIN(RADIANS(L$20))”
- Cell L23: “=B25*SIN(RADIANS(L$20))+C25*COS(RADIANS(L$20))”
- Copy K23 down to K27
- Copy L23 down to L27



Rotation button:

- The rotation angle “Alpha” will be contained in cell: “L20”
- The name of the button is “Rot_simple”
- Min = -1, Max = 75
- The VBA macro is shown below in blue

```
Private Sub Rot_simple_Change()
    If Rot_simple > 71 Then Rot_simple = 0
    If Rot_simple < 0 Then Rot_simple = 71
    Range("L20") = 5 * Rot_simple.Value
End Sub
```

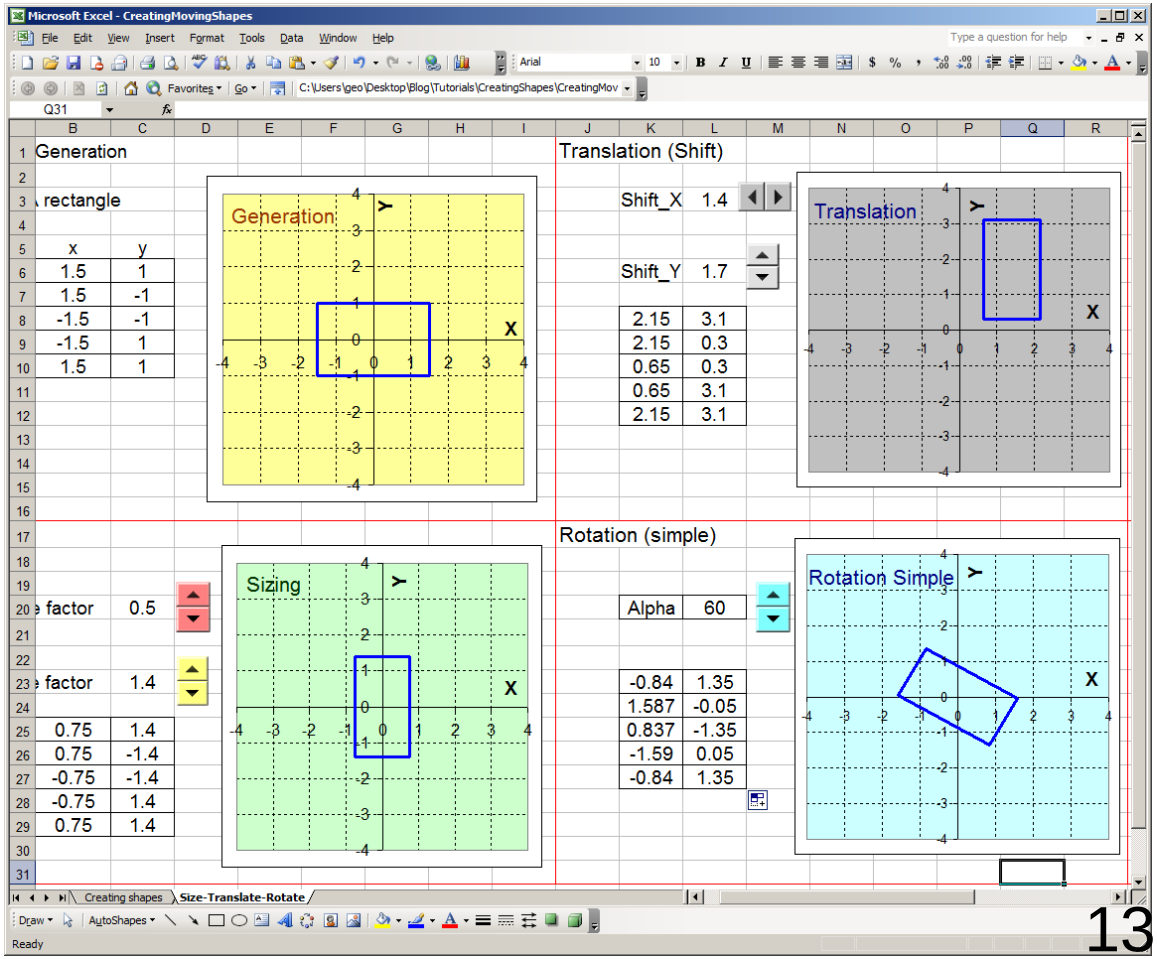
Case#1: Simple Rotation (continuation)

Plot the shifted rectangle:

- Use the data in the area “K23:L27” to create a plot of the resized rectangle (the blue chart in the picture)

Verify the functionality:

- The plotted rectangle must be a rotated version of the rectangle in the green graph
- Click the spin button and verify that the rectangle rotates smoothly around origin

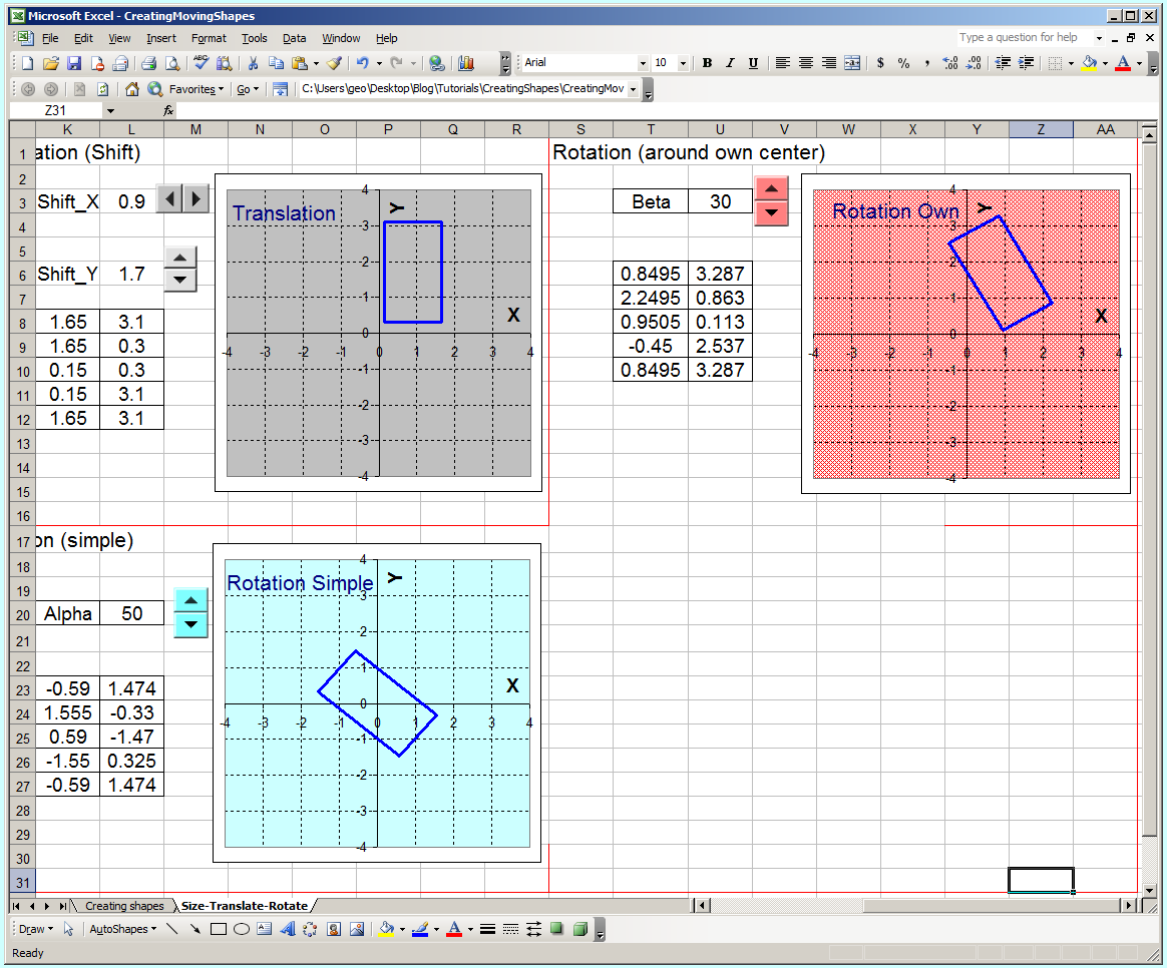


Case#2: Shape rotation around its own center:

In this case the rotation is done while the shape is centered in the origin and followed by the translation to the final position. On our spread sheet the object is rotated in origin then translated afterward but in real life (let's say a game) the object is applied a translation to origin, then a rotation and then a translation back to its normal position.

Note:

For this case I won't give detailed Excel instructions. The implementation is left to the reader. The macro is almost identical to the previous macro and the formulas for translation and simple rotation were covered previously.

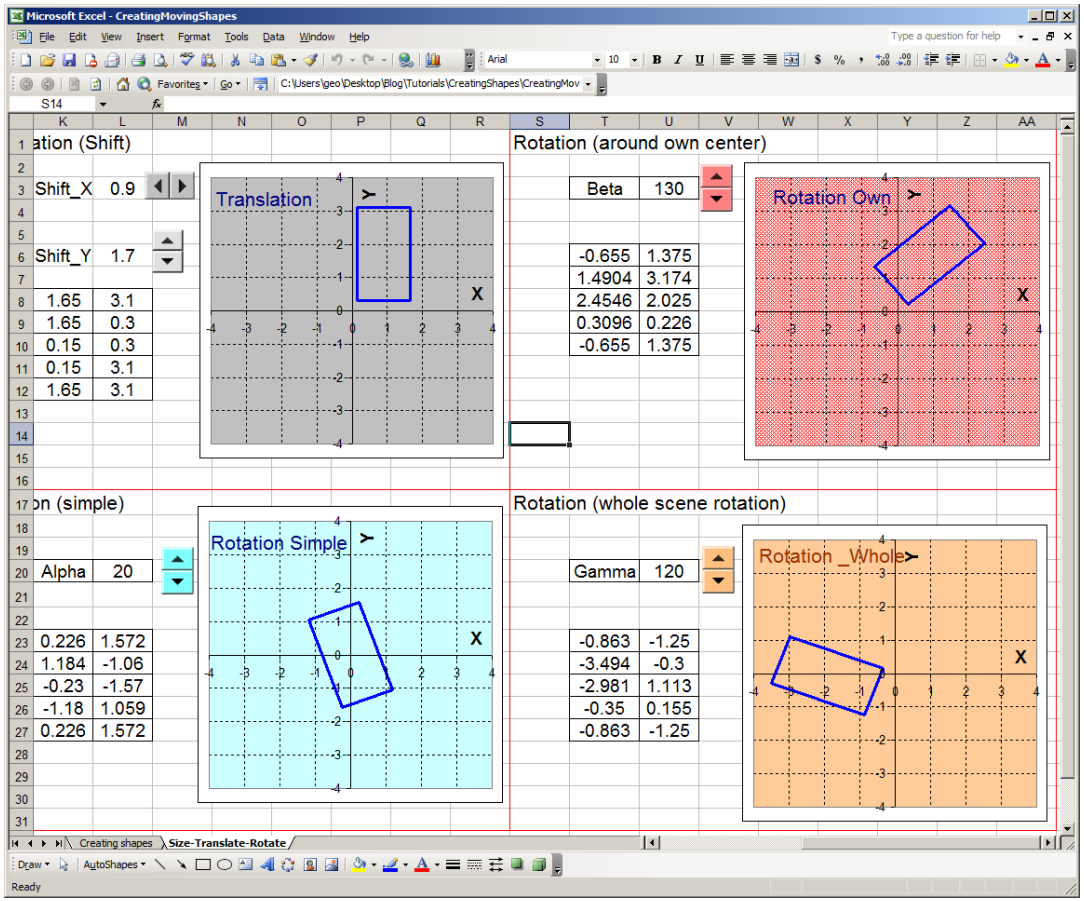


Case#3: Whole frame rotation:

Here the rotation is performed after all the necessary translations while all the shapes are in the right place (if there are more than one shape). If the scene rotation needs to be done around a different pole we need to translate the whole scene to that pole, then perform the rotation after which the opposite translation will be used to bring the scene in its original place

Note:

For this case I won't give detailed Excel instructions. The implementation is left to the reader. The macro is almost identical to the previous macro and the formulas for translation and simple rotation were covered before



Conclusions: Five different shape alterations were shown in this presentation:

1. ***Resizing*** – which can be done independently on each axis by multiplying the coordinates by a certain factor while the shape is centered in the origin.
2. ***Translation*** – which can be done independently on each axis by adding or subtracting a term to each coordinate.
3. ***Simple rotation*** – formulas were derived for rotation around the origin by an angle α of a shape centered in origin.
4. ***Rotation of a shape around its own axis*** – the exact derivation was not given but the recipe is to translate the shape back to origin, then rotate it and then translate the shape to it's final position (translation + rotation + translation)
5. ***Scene rotation*** – this is just a simple rotation but the shapes in the scene are held fixed at their own coordinates.

A MS Excel 2003 work book with all the examples can be downloaded. The reader is advised to examine the functionality of each section of the work sheet and try to rebuild it from scratch reading as little as possible. This presentation should be seen as a lure or a lifeline. The farther you can reach with the least reading, the more knowledge you'll gain. Your only real teacher lives in yourself. **Spend few days, don't rush!**