ADOBE® INDESIGN® CS3

ADOBE INDESIGN CS3 SCRIPTING: WORKING WITH TRANSFORMATIONS IN APPLESCRIPT

Adobe® InDesign® CS3 Scripting: Working with Transformations in AppleScript

Adobe InDesign CS3 Scripting: Working with Transformations in AppleScript

Operations that change the geometry of items on an InDesign page are called *transformations*. Transformations include scaling, rotation, shearing (skewing), and movement (or translation). In Adobe InDesign CS3 scripting, you apply transformations using the `transform` method. This one method replaces the `resize`, `rotate`, and `shear` methods used in previous versions of InDesign.

This document shows you how to transform objects in InDesign CS3 and discusses some of the technical details behind the new transformation architecture.

This document assumes you have a basic working knowledge of InDesign scripting.

Using the Transform Method

The `transform` method requires a transformation matrix (`transformation matrix`) object that defines the transformation or series of transformations to apply to the object. A transformation matrix can contain any combination of scale, rotate, shear, or translate operations, in any order.

The order in which transformations are applied to an object is important. Applying transformations in differing orders can produce very different results.

To transform an object, you follow two steps:

1. Create a transformation matrix.

2. Apply the transformation matrix to the object using the `transform` method. When you do this, you also specify the coordinate system in which the transformation is to take place. For more on coordinate systems, see "Coordinate Spaces" on page 6. In addition, you specify the center of transformation, or transformation origin. For more on specifying the transformation origin, see "Transformation Origin" on page 7.

The following scripting example demonstrates the basic process of transforming a page item. (For the complete script, see TransformExamples.)

```
--Rotate a rectangle "myRectangle" around its center point.
set myRotateMatrix to make transformation matrix with properties {counterclockwise
rotation angle:27}
transform myRectangle in pasteboard coordinates from center anchor with matrix
myRotateMatrix
--Scale a rectangle "myRectangle" around its center point.
set myScaleMatrix to make transformation matrix with properties {horizontal scale
factor:.5, vertical scale factor:.5}
transform myRectangle in pasteboard coordinates from center anchor with matrix
myScaleMatrix
--Shear a rectangle "myRectangle" around its center point.
set myShearMatrix to make transformation matrix with properties {clockwise shear
angle: 30}
transform myRectangle in pasteboard coordinates from center anchor with matrix
myShearMatrix
--Rotate a rectangle "myRectangle" around a specified ruler point ({72, 72}).
set myRotateMatrix to make transformation matrix with properties {counterclockwise
rotation angle:27}
```

```
transform myRectangle in pasteboard coordinates from {{72, 72}, center anchor}
with matrix myRotateMatrix with considering ruler units
--Scale a rectangle "myRectangle" around a specified ruler point ({72, 72}).
set myScaleMatrix to make transformation matrix with properties {horizontal scale
factor:.5, vertical scale factor:.5}
transform myRectangle in pasteboard coordinates from {{72, 72}, center anchor}
with matrix myScaleMatrix with considering ruler units
```

For a script that "wraps" transformation routines in a series of easy-to-use handlers, refer to the Transform script.

Working with Transformation Matrices

A transformation matrix cannot be changed once it has been created, but a variety of methods can interact with the transformation matrix to create a new transformation matrix based on the existing transformation matrix. In the following examples, we show how to apply transformations to a transformation matrix and replace the original matrix. (For the complete script, see TransformMatrix.)

```
--Scale a transformation matrix by 50% in both
--horizontal and vertical dimensions.
set myTransformationMatrix to scale matrix myTransformationMatrix
horizontally by .5 vertically by .5
--Rotate a transformation matrix by 45 degrees.
set myTransformationMatrix to rotate matrix by angle 45
--Shear a transformation matrix by 15 degrees.
set myTransformationMatrix to shear matrix by angle 15
```

When you use the `rotate matrix` method, you can use a sine or cosine value to transform the matrix, rather than an angle in degrees, as shown in the RotateMatrix script.

```
set myRectangle to rectangle 1 of page 1 of document 1
set myTransformationMatrix to make transformation matrix
--rotate matrix can take the following parameters: byAngle, byCosine, bySine;
--The following statements are equivalent (0.25881904510252 is the sine of 15
degrees, 0.96592582628907 is the cosine).
set myTransformationMatrix to rotate matrix myTransformationMatrix by angle 15
set myTransformationMatrix to rotate matrix myTransformationMatrix by cosine
0.965925826289
set myTransformationMatrix to rotate matrix myTransformationMatrix by sine
0.258819045103
--Rotate the rectangle by the rotated matrix--45 degrees.
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

When you use the `shear matrix` method, you can provide a slope, rather than an angle in degrees, as shown in the ShearMatrix script.

```
set myRectangle to rectangle 1 of page 1 of document 1
set myTransformationMatrix to make transformation matrix
--shear matrix can take the following parameters: by angle, by slope
--The following statements are equivalent. slope = rise/run
--so a 45 degree slope is 1.
set myTransformationMatrix to shear matrix myTransformationMatrix by slope 1
--Shear the rectangle.
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

You can get the inverse of a transformation matrix using the `invert matrix`method, as shown in the following example. (For the complete script, see InvertMatrix.) You can use the inverted transformation matrix to undo the effect of the matrix.

```
set myRectangle to rectangle 1 of page 1 of document 1
set myTransformationMatrix to make transformation matrix with properties
{counterclockwise rotation angle:30, horizontal translation:12, vertical
translation:12}
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
set myNewRectangle to duplicate myRectangle
--Move the duplicated rectangle to the location of the original
--rectangle by inverting, then applying the transformation matrix.
set myTransformationMatrix to invert matrix myTransformationMatrix
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

You can add transformation matrices using the `catenate matrix`method, as shown in the following example. (For the complete script, see CatenateMatrix.)

```
set myTransformationMatrixA to make transformation matrix with properties
{counterclockwise rotation angle:30}
set myTransformationMatrixB to make transformation matrix with properties
{horizontal translation:72, vertical translation:72}
set myRectangle to rectangle -1 of page 1 of document 1
set myNewRectangle to duplicate myRectangle
--Rotate the duplicated rectangle.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrixA
set myNewRectangle to duplicate myRectangle
--Move the duplicate (unrotated) rectangle.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrixB
--Merge the two transformation matrices.
set myTransformationMatrix to catenate matrix myTransformationMatrixA with matrix
myTransformationMatrixB
set myNewRectangle to duplicate myRectangle
--The duplicated rectangle will be both moved and rotated.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

When an object is transformed, you can get the transformation matrix that was applied to it, using the `transform values of` method, as shown in the following script fragment. (For the complete script, see TransformValuesOf.)

```
set myRectangle to rectangle -1 of page 1 of document 1
--Note that transform values of always returns a list containing a
--single transformation matrix.
set myTransformArray to transform values of myRectangle in pasteboard coordinates
set myTransformationMatrix to item 1 of myTransformArray
set myRotationAngle to counterclockwise rotation angle of myTransformationMatrix
set myShearAngle to clockwise shear angle of myTransformationMatrix
set myXScale to horizontal scale factor of myTransformationMatrix
set myYScale to vertical scale factor of myTransformationMatrix
set myXTranslate to horizontal translation of myTransformationMatrix
set myYTranslate to vertical translation of myTransformationMatrix
set myString to "Rotation Angle: " & myRotationAngle & return
set myString to myString & "Shear Angle: " & myShearAngle & return
set myString to myString & "Horizontal Scale Factor: " & myXScale & return
```

```
set myString to myString & "Vertical Scale Factor: " & myYScale & return
set myString to myString & "Horizontal Translation: " & myXTranslate & return
set myString to myString & "Vertical Translation: " & myYTranslate & return &
return
set myString to myString & "Note that the Horizontal Translation and" & return
set myString to myString & "Vertical Translation values are the location" & return
set myString to myString & "of the center anchor in pasteboard coordinates."
display dialog (myString)
```

Note: The values in the horizontal- and vertical-translation fields of the transformation matrix returned by this method are the location of the upper-left anchor of the object, in pasteboard coordinates.

Coordinate Spaces

In the transformation scripts we presented earlier, you might have noticed the `pasteboard coordinates` enumeration provided as a parameter for the transform method. This parameter determines the system of coordinates, or *coordinate space*, in which the transform operation occurs. The coordinate space can be one of three values:

- **`pasteboard coordinates`** is a coordinate space that uses points as units and extends across all spreads in a document. It does not correspond to InDesign's rulers or zero point. Transformations applied to objects have no effect on this coordinate space (e.g., the angle of the horizontal and vertical axes do not change).

- **`parent coordinates`** is the coordinate space of the parent of the object. Any transformations applied to the parent affect the parent coordinates; for example, totating the parent object changes the angle of the horizontal and vertical axes of this coordinate space. In this case, the parent object refers to the group or page item containing the object; if the parent of the object is a page or spread, parent coordinates are the same as pasteboard coordinates.

- **`inner coordinates`** is a coordinate space based on the object itself.

The following script shows the differences between the three coordinate spaces. (For the complete script, see CoordinateSpaces.)

```
set myRectangle to rectangle 1 of group 1 of page 1 of document 1
set myString to "The page contains a group which has been" & return
set myString to myString & "rotated 45 degrees (counterclockwise)." & return
set myString to myString & "The rectangle inside the group was" & return
set myString to myString & "rotated 45 degrees counterclockwise before" & return
set myString to myString & "it was added to the group." & return & return
set myString to myString & "Watch as we apply a series of scaling" & return
set myString to myString & "operations in different coordinate spaces."
display dialog myString
set myTransformationMatrix to make transformation matrix with properties
{horizontal scale factor:2}
--Transform the rectangle using inner coordinates.
transform myRectangle in inner coordinates from center anchor with matrix
myTransformationMatrix
--Select the rectangle and display an alert.
select myRectangle
display dialog "Transformed using inner coordinates."
--Undo the transformation.
tell document 1 to undo
--Transform using parent coordinates.
transform myRectangle in parent coordinates from center anchor with matrix
myTransformationMatrix
```

```
select myRectangle
display dialog "Transformed using parent coordinates."
tell document 1 to undo
--Transform using pasteboard coordinates.
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
select myRectangle
display dialog "Transformed using pasteboard coordinates."
tell document 1 to undo
```

Transformation Origin

The transformation origin is the center point of the transformation. The transformation origin can be specified in several ways:

- Bounds space:

  - anchor — An anchor point on the object itself.

    ```
    center anchor
    ```

- Ruler space:

  - (x, y), page index — A point, relative to the ruler origin on a specified page of a spread.

    ```
    {{72, 144}, 1}
    ```

  - (x, y), location — A point, relative to the parent page of the specified location of the object. Location can be specified as an anchor point or a coordinate pair. It can be specified relative to the object's geometric or visible bounds, and it can be specified in a given coordinate space.

    ```
    {{72, 144}, center anchor}
    ```

- Transform space:

  - (x, y) — A point in the pasteboard coordinate space.

    ```
    {72, 72}
    ```

  - (x, y), coordinate system — A point in the specified coordinate space.

    ```
    {{72, 72}, parent coordinates}
    ```

  - ((x, y)) — A point in the coordinate space given as the `in` parameter of the `transform` method.

    ```
    {{72, 72}}
    ```

The following script example shows how to use some of the transformation origin options. (For the complete script, see TransformationOrigin.)

```
set myRectangle to rectangle 1 of document 1
set myString to "Watch as we rotate the rectangle using different anchor points," &
return
set myString to myString & "bounds types, and coordinate spaces." & return & return
set myString to myString & "You might have to drag the alert aside to" & return
set myString to myString & "see the effect of the transformation."
set myNewRectangle to duplicate myRectangle
set myTransformationMatrix to make transformation matrix with properties
{counterclockwise rotation angle:30}
--Rotate around the duplicated rectangle's center point.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
--Select the rectangle and display an alert.
select myNewRectangle
```

```
display dialog "Rotated around center anchor."
--Undo the transformation.
tell document 1 to undo
--Rotate the rectangle around the ruler location [-100, -100]. Note that the anchor
point specified here specifes the page
--containing the point--*not* that transformation point itself. The transformation
gets the ruler coordinate [-100, -100] based
--on that page. Setting the considerRulerUnits parameter to true makes certain that
the transformation uses the current
--ruler units.
transform myNewRectangle in pasteboard coordinates from {{-100, -100}, top left
anchor} with matrix myTransformationMatrix with considering ruler units
--Move the page guides to reflect the transformation point.
tell guide 1 of page 1 of document 1 to set location to -100
tell guide 2 of page 1 of document 1 to set location to -100
--Select the rectangle and display an alert.
select myNewRectangle
display dialog "Rotated around -100x, -100y."
--Undo the transformation and the guide moves.
tell document 1 to undo
tell document 1 to undo
tell document 1 to undo
```

Resolving Locations

Sometimes, you need to get the location of a point specified in one coordinate space in the context of another coordinate space. To do this, you use the `resolve` method, as shown in the following script example. (For the complete script, see ResolveLocation.)

```
set myRectangle to rectangle 1 of group 1 of page 1 of document 1
--Get ruler coordinate {72, 72} in pasteboard coordinates.
set myPageLocation to resolve myRectangle location {{72, 72}, top right anchor} in
pasteboard coordinates with considering ruler units
--resolve returns a list containing a single item.
set myPageLocation to item 1 of myPageLocation
display dialog "Pasteboard Coordinates:" & return & return & "X: " & item 1 of
myPageLocation & return & "Y: " & item 2 of myPageLocation
--Get ruler coordinate {72, 72} in parent coordinates.
set myPageLocation to resolve myRectangle location {{72, 72}, top right anchor} in
parent coordinates with considering ruler units
--resolve returns a list containing a single item.
set myPageLocation to item 1 of myPageLocation
display dialog "Parent Coordinates:" & return & return & "X: " & item 1 of
myPageLocation & return & "Y: " & item 2 of myPageLocation
```

Transforming Points

You can transform points as well as objects, which means scripts can perform a variety of mathematical operations without having to include the calculations in the script itself. This is particularly useful for AppleScript, which lacks the basic trigonometric functions (sine, cosine) required for most transformations. The ChangeCoordinates sample script shows how to draw a series of regular polygons using this approach:

```
--General purpose routine for drawing regular polygons from their center point.
```

```
    on myDrawPolygon(myParent, myCenterPoint, myNumberOfPoints, myRadius,
myStarPolygon, myStarInset)
        local myPathPoints, myTransformedPoint
        tell application "Adobe InDesign CS3"
            set myPathPoints to {}
            set myPoint to {0, 0}
            if myStarPolygon is true then
                set myNumberOfPoints to myNumberOfPoints * 2
            end if
            set myInnerRadius to myRadius * myStarInset
            set myAngle to 360 / myNumberOfPoints
            set myRotateMatrix to make transformation matrix with properties
            {counterclockwise rotation angle:myAngle}
            set myOuterTranslateMatrix to make transformation matrix with properties
            {horizontal translation:myRadius}
            set myInnerTranslateMatrix to make transformation matrix with properties
            {horizontal translation:myInnerRadius}
            repeat with myPointCounter from 0 to myNumberOfPoints - 1
                --Translate the point to the inner/outer radius.
                if myStarInset = 1 or my myIsEven(myPointCounter) is true then
                    set myTransformedPoint to change coordinates
                    myOuterTranslateMatrix point myPoint
                else
                    set myTransformedPoint to change coordinates
                    myInnerTranslateMatrix point myPoint
                end if
                --Rotate the point.
                set myTransformedPoint to change coordinates
                myRotateMatrix point myTransformedPoint
                copy myTransformedPoint to the end of myPathPoints
                set myRotateMatrix to rotate matrix myRotateMatrix by angle myAngle
            end repeat
            --Create a new polygon.
            tell myParent
                set myPolygon to make polygon
            end tell
            --Set the entire path of the polygon to the array we've created.
            set entire path of path 1 of myPolygon to myPathPoints
            --If the center point is somewhere other than [0,0],
            --translate the polygon to the center point.
            if item 1 of myCenterPoint is not equal to 0 or item 2 of
            myCenterPoint is not equal to 0 then
                set myTransformationMatrix to make transformation matrix with properties
                {horizontal translation:item 1 of myCenterPoint,
                vertical translation:item 2 of myCenterPoint}
                transform myPolygon in pasteboard coordinates from {myCenterPoint,
                center anchor} with matrix myTransformationMatrix with considering
                ruler units
            end if
        end tell
    end myDrawPolygon
    --This function returns true if myNumber is even, false if it is not.
    on myIsEven(myNumber)
        set myResult to myNumber mod 2
        if myResult = 0 then
            set myResult to true
        else
```

```
        set myResult to false
    end if
    return myResult
end myIsEven
```

You also can use the `change coordinates` method to change the positions of curve control points, as shown in the FunWithTransformations sample script.

Transforming Again

Just as you can apply a transformation or sequence of transformations again in the user interface, you can do so using scripting. There are four methods for applying transformations again:

*   `transform again`

*   `transform again individually`

*   `transform sequence again`

*   `transform sequence again individually`

The following script fragment shows how to use `transform again`. (For the complete script, see TransformAgain.)

```
set myTransformationMatrix to make transformation matrix with properties
{counterclockwise rotation angle:45}
transform myRectangleA in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
set myRectangleB to duplicate myRectangleA
transform myRectangleB in pasteboard coordinates from {{0, 0}, top left anchor}
with matrix myTransformationMatrix with considering ruler units
set myRectangleC to duplicate myRectangleB
set myResult to transform again myRectangleC
set myRectangleD to duplicate myRectangleC
set myResult to transform again myRectangleD
set myRectangleE to duplicate myRectangleD
set myResult to transform again myRectangleE
set myRectangleF to duplicate myRectangleE
set myResult to transform again myRectangleF
set myRectangleG to duplicate myRectangleF
set myResult to transform again myRectangleG
set myRectangleH to duplicate myRectangleG
set myResult to transform again myRectangleH
transform myRectangleB in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
set myResult to transform again myRectangleD
set myResult to transform again myRectangleF
set myResult to transform again myRectangleH
```