LibreOffice
Community

Regina Henschel

Custom Shape Tutorial

7

LibreOffice is a registered trademark of The Document Foundation
Further Information is available at libreoffice.org

Custom Shape Tutorial

How to Create Your Own Custom Shapes

by Regina Henschel

Preface

Copyright

This document is Copyright © 2022 by Regina Henschel. This document maybe distributed and/or modified under the terms of either the GNU General Public License (https://www.gnu.org/licenses/gpl.html), version 3 or later, or the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0/), version 4.0 or later.

All trademarks within this guide belong to their legitimate owners.

Contributors

To this edition.

Regina Henschel

Revisions

March 2022: Olivier Hallot has corrected numerous linguistic errors.

Feedback

Please direct comments or suggestions about this document to Regina Henschel [rb.henschel@t-online.de]
or use the discussion page of
https://wiki.documentfoundation.org/User:Regina/Custom_Shape_Tutorial
or write a mail to documentation@global.libreoffice.org.

Software

The shapes have been tested in LibreOffice version 7.1.5, only extruded custom shapes (chapter 10) need version 7.4 and for custom shapes in text path mode (chapter 9) you should use at least version 7.3.

This tutorial contains images of the shapes. The shapes themselves are collected in a separate file “Shapes from Custom Shape Tutorial.odg”
https://wiki.documentfoundation.org/images/e/ed/Shapes_from_Custom_Shape_Tutorial.odg

Table of Contents

1 Introduction

The term **custom shape** is used as umbrella term in this tutorial for those shapes, which are written as element `<draw:custom-shape>` in the document source file. LibreOffice provides a lot of predefined custom shapes. They are grouped to the sets ‘Basic Shapes’, ‘Block Arrows’, ‘Symbol Shapes’, ‘Stars and Banners’, ‘Callouts’, and ‘Flowchart’. And all shapes from the ‘Fontwork Gallery’ are custom shapes too.

To distinguish a custom shape from other kind of shapes select the shape and look at the tool ‘Toggle Extrusion’  . It is only enabled, if the shape is a custom shape.

LibreOffice provides no tools to create own custom shapes. This tutorial shows you how to create own custom shapes by defining them directly in the source file. Once they are defined, they can be used in other documents without manipulating the source file again.

Warning: Self-made custom shapes are mainly useful, if you stick to LibreOffice and OpenDocument Format. Converting self-made custom shapes into OOXML format does not work in all cases. Microsoft Office currently cannot read all kind of self-made custom shapes in OpenDocument format.

As you will work directly in the source file with XML markup, you need some tools. Find more about editing the source file on page ODF_Markup[1] in the Wiki. In case you need help with the tools, you can ask on the associated “Discussion” page, so that the page can be improved, or you write a mail to documentation@global.libreoffice.org.

The tutorial assumes that you know general handling of shapes as described in the Draw Guide.[2]

The tutorial contains some exercises. The solutions can be found at the end of each chapter. The exercise has a link to its solution. The exercise number in the solution is a link back to the text of the exercise.

The tutorial contains images of the shapes. The discussed shapes themselves are provided in a separate document “Shapes from Custom Shape Tutorial”[3]. That way they are available even if you use a printed or a PDF version of the tutorial.

1 https://wiki.documentfoundation.org/Documentation/ODF_Markup , [last called 2021-08-28]
2 https://documentation.libreoffice.org/en/english-documentation/ [last called 2021-08-29]
3 https://wiki.documentfoundation.org/images/e/ed/Shapes_from_Custom_Shape_Tutorial.odg

2 Basic Parts of a Custom Shape

2.1 Examine Markup of Predefined Shape ‘Right Triangle’

Start a new Draw document and insert the shape ‘Right Triangle’ at position x = 50 mm, y = 20 mm with size width = 40 mm and height = 60 mm. The shape belongs to the set ‘Basic Shapes’. In case you work with the packed file format ‘.odg’, enter the package and open the file `content.xml` in an editor. In case you work with the flat file format ‘.fodg’, open it directly in an editor. Search for the element `<draw:custom-shape>`. You should see something similar to the following markup

```
<draw:custom-shape draw:style-name="gr1"
                   draw:text-style-name="P1"
                   draw:layer="layout"
                   svg:width="4cm"
                   svg:height="6cm"
                   svg:x="6cm"
                   svg:y="3cm">
    <text:p/>
    <draw:enhanced-geometry … />
</draw:custom-shape>
```

2.1.1 Common Parts

The attributes of a `<draw:custom-shape>` element are similar to other shape types.

`draw:style-name="gr1"`	The shape uses the graphic properties which are contained in a style with name ‘gr1’. Those determine the fill style and line width, for example.
`draw:text-style-name="P1"`	The text in the shape uses a paragraph style with name ‘P1’. Here, our shape has no text. That can be seen by the empty element `<text:p/>`.
`draw:layer="layout"`	The shape is placed on the layer ‘layout’. Layers exist only in Draw and Impress. It is irrelevant for this tutorial.
`svg:width="4cm" svg:height="6cm"`	The attributes `svg:width` and `svg:height` determine the size of the shape. LibreOffice has an internal precision of 1/100 mm. Because of rounding errors between non-metric system (inch, point) and metric system (mm, cm) the values are sometimes by 1 or 2 hundredth millimeter too small or too large. Don’t worry about it.
`svg:x="6cm" svg:y="3cm"`	The attributes `svg:x` and `svg:y` determine the position of the shape. For details see section 2.5.

In case the shape is rotated or sheared, there will be a `draw:transform` attribute instead of `svg:width`, `svg:height` and `svg:x`, `svg:y`.

The part which is specific for a custom shape is the element `<draw:enhanced-geometry … />`

2.1.2 First Look at Element `<draw:enhanced-geometry>`

```
<draw:enhanced-geometry
    svg:viewBox="0 0 21600 21600"
    draw:glue-points="0 0 0 10800 0 21600 10800 21600 21600 21600 10800 10800"
    draw:text-areas="1900 12700 12700 19700"
    draw:type="right-triangle"
    draw:enhanced-path="M 0 0 L 21600 21600 0 21600 0 0 Z N"
/>
```

`draw:type="right-triangle"`

LibreOffice uses this attribute to decide, how to treat the shape in import from, and export to Microsoft file formats. The ODF standard allows you to omit this attribute, but LibreOffice needs it to decide how to handle the shape.

For your own shapes you must use `draw:type="non-primitive"`.

`draw:enhanced-path="…"`

The drawing of the shape consists of geometric elements like lines, curves or circles. The `draw:enhanced-path` attribute contains the definition of the drawing. It is a sequence of commands and coordinates of points. This attribute is mandatory.

`svg:viewBox="0 0 21600 21600"`

The definitions of the geometric elements use a local, unit-less coordinate system for the points. As usual in 2D computer graphics the x-axis points right and the y-axis points down. This attribute is mandatory.

Find more about this attribute in section 2.5.

`draw:glue-points="…"`

Each shape has four default glue points and you can add your own glue points. Custom shapes may have additional shape specific glue points. They are listed in this optional attribute.

`draw:text-areas="…"`

Set explicitly the rectangular area into which text writing is possible. The values determine the left, top, right and bottom edges of this area. This attribute is optional. If omitted, the entire custom shape is used.

In case the shape is mirrored, attributes `draw:mirror-horizontal` and `draw:mirror-vertical` will be added.

Exercise 2.1

Determine the coordinates of points A, B and C in Figure 1.
→ solution

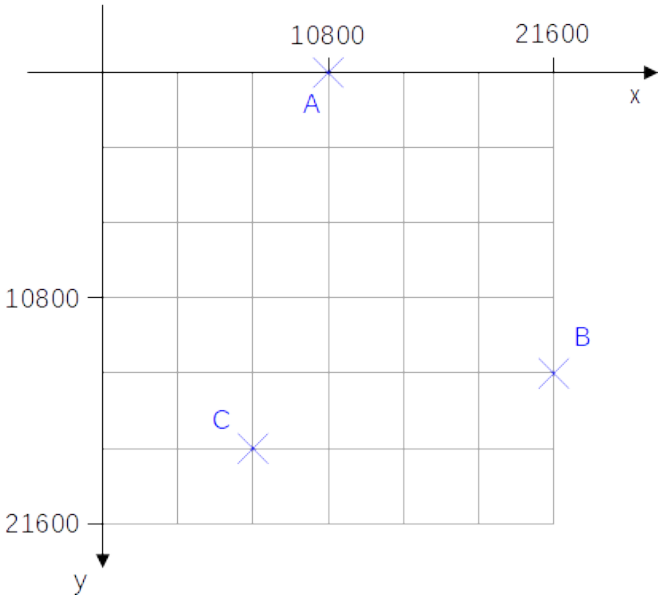


Figure 1: Locale coordinate system

2.1.3 First Look at Attribute `draw:enhanced-path`

The enhanced-path is the essential part of the enhanced-geometry of a custom shape. It describes how to draw the geometric elements. It uses commands like

`M x y`	Lift off the pen, move to point (x|y), put down the pen. That starts a new sub-path. Point (x|y) is then the 'current point' and it is the 'initial point' of the sub-path.
`L x y`	Draw a line from the current point to point (x|y). Point (x|y) is then the new 'current point'.
`Z`	Close the sub-path by drawing a line from the 'current point' to the 'initial point' of the sub-path.
`N`	End the sub-path set and lift off the pen. The 'current point' is then undefined.

More details about the above commands are in chapter 5.1
If several identical commands follow each other directly, only the first one needs to be written down.
So the attribute `draw:enhanced-path="M 0 0 L 21600 21600 0 21600 0 0 Z N"` has the following meaning:

`M 0 0`	Put down the pen at point (0|0). This is the initial point of the sub-path.
`L 21600 21600`	Draw a line to point (21600|21600).

`0 21600`	Draw a line to point (0|21600). Command L is repeated,
`0 0`	Draw a line to point (0|0). Command L is repeated.
`Z`	Draw a line to the initial point of the sub-path.
`N`	Lift off the pen. This ends the sub-path set.

Exercise 2.2

What shape do you get with the following path?
`draw:enhanced-path="M 10800 0 L 21600 10800 10800 21600 0 10800 Z N"`

→ solution

Exercise 2.3

Write a `<draw:enhanced-path>` element for the shape in Figure 2.
→ solution

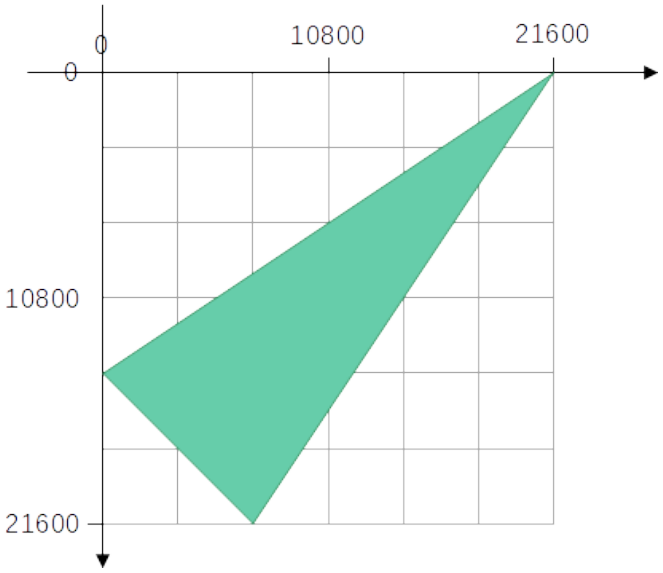


Figure 2: Triangle in local coordinate system

Exercise 2.4

Do the following paths give the same shape?

(a) `draw:enhanced-path="M 0 0 L 21600 21600 0 21600 0 0 Z N"`

(b) `draw:enhanced-path="M 0 0 L 21600 21600 0 21600 Z N"`

→ solution

Exercise 2.5

Why is the following path invalid in regard to the ODF standard although applications might be fault tolerant and show something?

`draw:enhanced-path="L 21600 21600 0 10800 10800 0 Z N"`

→ solution

2.2 Your first Custom Shape

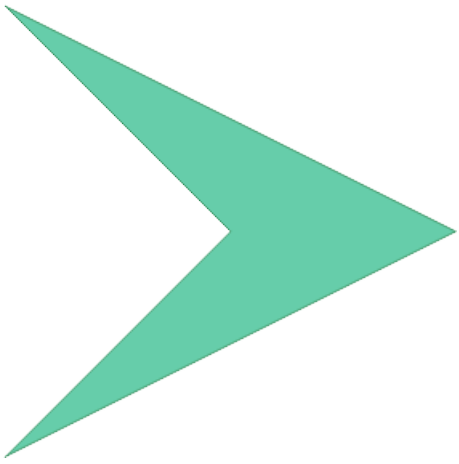


Figure 3: Desired shape

Let’s create the shape shown in Figure 3. In general, this process has the following steps.

1. Put the desired shape into a coordinate system and determine the coordinates of its corners.

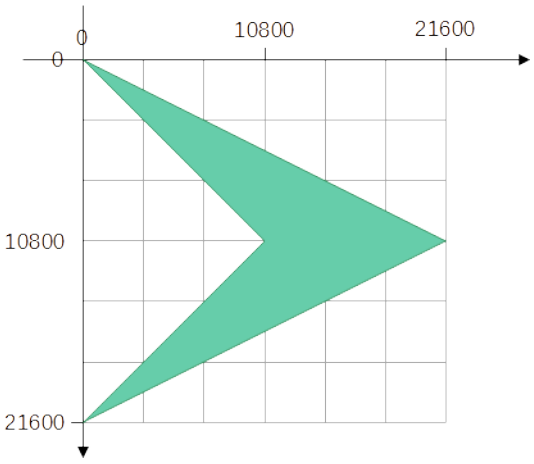


Figure 4: Desires shape in coordinate system

   You get the corners `(0|0)`, `(21600|10800)`, `(0|21600)` and `(10800|10800)`.

2. Write the necessary enhanced-path down. For example, you can use
   `"M 0 0 L 21600 10800 0 21600 10800 10800 Z N"`

3. Prepare the document

   Start with a new Draw document. Then insert an existing custom shape – here use the ‘Right Triangle’ shape. Starting with an existing custom shape has the advantage, that you need not manipulate the surrounding document code – the styles for the graphic and text properties are already defined – and you don't need to locate where to put the shape markup in the document.

   Start with a square size. Set area fill and line style as you like. Close the document.

4. Edit the document

   Enter the package and open sub-file `content.xml` in an editor or open the flat format file in an editor. Search for the element `<draw:custom-shape>`.

   Change the value of attribute `draw:type` to "`non-primitive`".

   Exchange the value of attribute `draw:enhanced-path` with your path from step 2.

   Save the changes and close the document.

5. Test your shape

   Open the file in LibreOffice. Does the shape look as desired? If not, look for errors in steps 1, 2 or 4. Look for Typos too.

   The editor should have you already warned about structural errors in the markup. To detect errors regarding to the ODF standard, you can use the validator[4].

6. Document your shape

   Create a text document that contains

   – (useful for this tutorial) Write down to which section of the tutorial this shape belongs.

   – The new created shape. Copy and paste from the Draw-document. If you put the shape at the beginning of the document, then it is shown in the thumbnail of the document. That makes it easier to find it in your library.

   – The figure and coordinates from steps 1 and 2.

   – Write down mathematical equations which you have made to find the coordinates. (You likely need no equation for this example.)

   – Write down the meaning of the equations used in the custom shape. (We have no equations in this custom shape.)

   – The markup of the final `draw:enhanced-geometry` element from step 4.

   Save this documentation into a library. You will need the document if you will change the shape or extend its features.

7. Drag the shape into your Gallery for use in other documents (do it later, because some changes are still missing).

4 https://odfvalidator.org/ [last called 2021-08-31]
Item *auto-detect* of *ODF Version* is not reliable, choose *OASIS ODF 1.3 (extended conforming)*.

2.3 Attribute `draw:text-areas`

Make a document with your shape created in section 2.2. Enter some text.

The text added to the shape in Figure 5 consists of a sequence of ‘single character followed by space’ and has a small font size. Word wrap is enabled and text anchor is set to left/middle by the ‘Text attributes’ dialog. That makes the text area more visible.

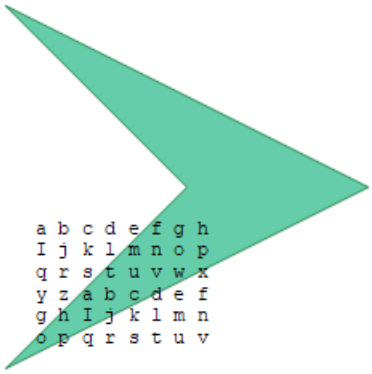


Figure 5: Wrong text area

Notice the text position and size. The text area from the ‘Right Triangle’ shape is used. We need to adapt it to the new enhanced path. Figure 6 shows the desired area as hatched rectangle.

The area shall have 1/6 of the shape height and shall be vertically centered. The area shall not cover the shape. You need the left, top, right, and bottom coordinates of the area edges.

$\text{left} = 0,\ \text{top} = \frac{1}{2} \cdot \frac{5}{6} \cdot 21600 = 9000$

$\text{right} = \text{top} = 9000$ and

$\text{bottom} = 21600 - 9000 = 12600.$

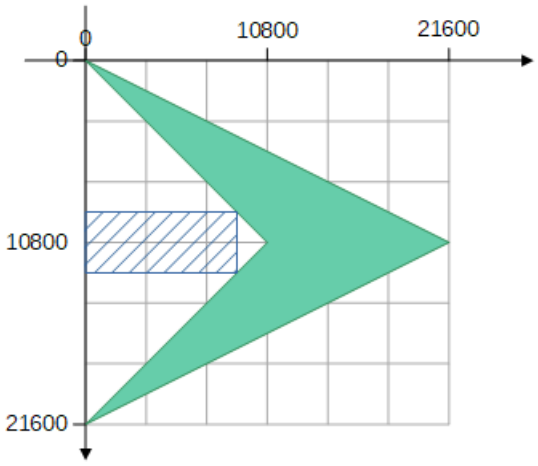


Figure 6: Desired text area

So, to get this rectangle you need `draw:text-areas="0 9000 9000 12600"` in the source file. Edit it accordingly and update your documentation.

The actual text is not restricted to the text-area rectangle and can overflow. The overflow directions can be controlled in the ‘Text Attributes’ dialog. You can find some hints on using the dialog in the wiki page ‘Text in Custom Shapes’.[5]

Coordinates can be negative. You can define a text-area, which is extended to the left, for example. Figure 7 shows the same shape – now with the attribute
`draw:text-areas="-6000 6000 6000 15600"`

5 https://wiki.documentfoundation.org/Documentation/Text_in_Custom_Shapes [last called 2021-08-29]

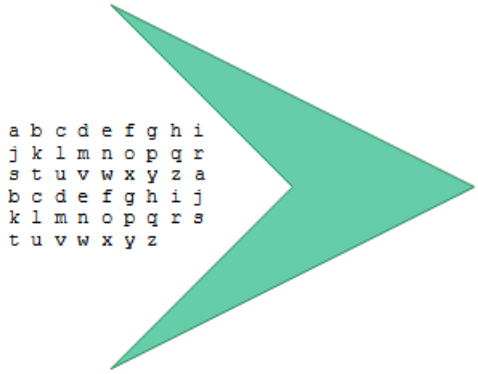


Figure 7: Negative position of left edge of text area

Why the attribute name has a plural ‘s’? Because the attribute can specify one or two text areas. If a second text area exists it is used in case the text style sets a writing mode “top-bottom”. Such writing mode is used for some East Asian languages.

Rotate the shape. Notice that the text area rotates together with the path.

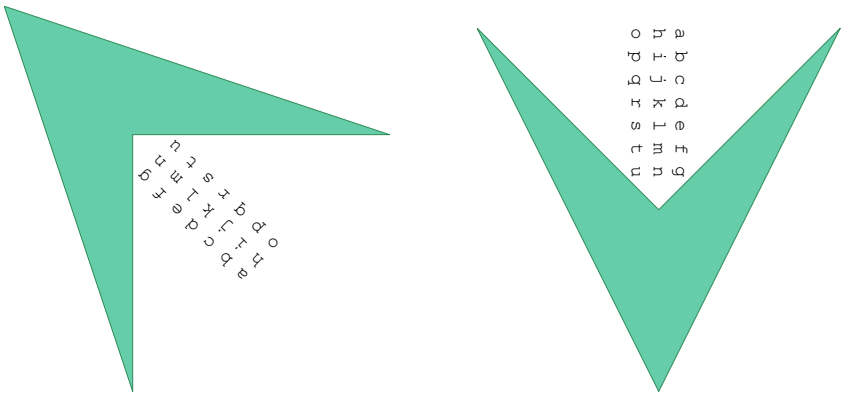


Figure 8: Text in rotated shape

The enhanced geometry of a custom shape provides the attribute `draw:text-rotate-angle` to add a rotation to the text area. The text area is rotated around its center and the rotation is clockwise. If the angle value has no unit, it defaults to degrees. You can determine the angle unit as `deg`, `grad` or `rad`. This attribute can be used to get an upright text in a rotated shape, for example.

Change the markup for the right shape in figure 8 as this:

```
<draw:enhanced-geometry draw:type="non-primitive" svg:viewBox="0 0 21600 21600"
    draw:text-areas="0 9000 9000 12600" draw:text-rotate-angle="90deg"
    draw:glue-points="0 0 21600 10800 0 21600"
    draw:enhanced-path="M 0 0 L 21600 10800 0 21600 10800 10800 Z N" />
```

and you will get the results in the shape of figure 9.

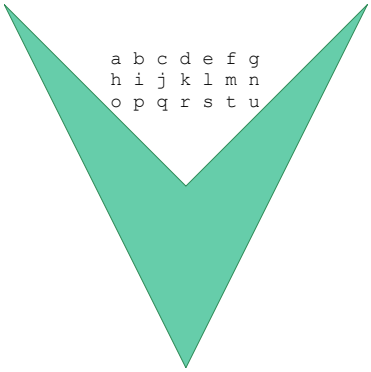


Figure 9: Rotated shape with additional rotation of text area

There is no way to make the additional text area rotation to depend on the shape rotation. You need to adapt the text area rotation manually, when changing the shape rotation. Unfortunately, LibreOffice has yet no user interface for this attribute[6]. You have to edit the markup in the document source or use a macro.

You can add a ‘Text Box’ to the shape in Writer. This is generated with the position and size defined in the attribute `draw:text-areas`. But in fact, this ‘Text Box’ is a frame, and the shape will act as a frame. That means, the shape can no longer rotate and the text in this ‘Text Box’ cannot overflow but you get a red triangle that indicates that there is more text not shown.

2.4 Attribute `draw:glue-points`

The shape ‘Right Triangle’ has glue points in the middle of its edges and in its corners. These positions are not always suitable for the new shape. It should get only glue points in its three outer corners. A glue point is given by x-coordinate and y-coordinate, separated by space. Multiple glue points are listed with a space between them.

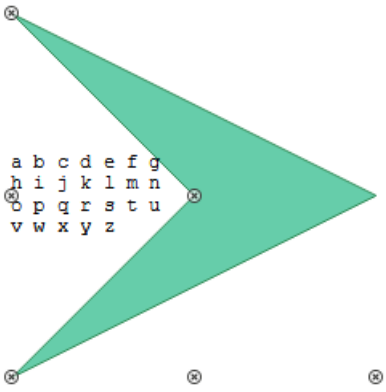


Figure 10: Wrong glue points

> **Exercise 2.6:**
>
> Determine the needed attribute value.
>
> → solution

You have surely found that you need `draw:glue-points="0 0 21600 10800 0 21600"`. Change it in the new shape and update your documentation. Your shape is ready now.

6 https://bugs.documentfoundation.org/show_bug.cgi?id=144942

The ODF standard defines the attribute `draw:glue-point-leaving-directions` to restrict the glue point so that a connector can leave it only in one direction. That is not implemented in LibreOffice. And the ODF standard defines the attribute `draw:glue-point-type`. That allows to auto-generate glue points or to forbid glue points. That is only partly implemented in LibreOffice. We therefore skip these attributes.

2.5 Attribute `svg:viewBox`

The attribute `svg:viewBox` in ODF is derived from the similar attribute in the 'Scalable Vector Graphics' (SVG) language[7]. The meaning of the parameters are svg:viewBox="left top width height". We examine it in this section.

2.5.1 Preparation

Start with a new document in Draw. Set Draw (temporarily) to use measurement unit 'Centimeter'. That is in menu Tools > Options > LibreOffice Draw > General > Settings. Set the grid to 'Resolution' 0.5 cm and 'Subdivision' 5. That is in Tools > Options > LibreOffice Draw > Grid.

Set the page to 12 cm width and 8 cm height in menu Page > Page Properties > tab 'Page'…

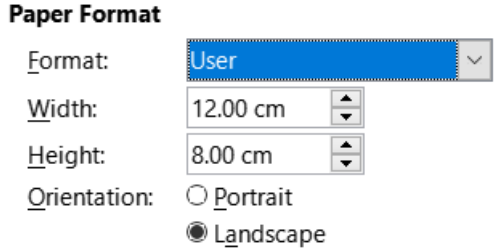


Figure 11: User defined page size

… and left margin to 2 cm and top margin to 1 cm.

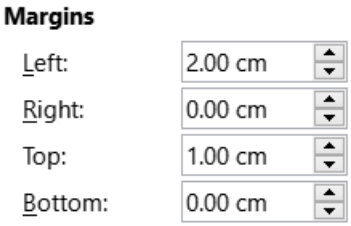


Figure 12: Page margins

Insert the shape 'Right Triangle'. Save the file. Open it in an editor to manipulate element `<draw:custom-shape>`.

Change the element to the given values below. Keep the brown parts unchanged. Delete the attributes `draw:glue-point` and `draw:text-areas`, they are not relevant here.

```
<draw:custom-shape draw:style-name="gr1"
                   draw:text-style-name="P1"
                   draw:layer="layout"
                   svg:width="6cm"
                   svg:height="4cm"
                   svg:x="4cm"
                   svg:y="2cm">
    <text:p/>
```

7 https://www.w3.org/TR/SVG11/coords.html#ViewBoxAttribute

```
    <draw:enhanced-geometry svg:viewBox="10 40 180 80"
                            draw:type="non-primitive"
                            draw:enhanced-path="M 40 80 L 160 60 N"/>
</draw:custom-shape>
```

Save the changed file, but do not open it in LibreOffice now.

The markup above will generate a straight-line segment. Determine the coordinates of its start and end point in the document. Please only read further, after you have tried to calculate them.

2.5.2 Explanation – Snap-Rectangle

So let see, how the calculation must be done.

The attributes `svg:width="6cm"`, `svg:height="4cm"`, `svg:x="4cm"` and `svg:y="2cm"` determine position and size of the rectangle of the resize-handles, which you get, when you click on the shape. This rectangle is called the '**snap-rectangle**'.

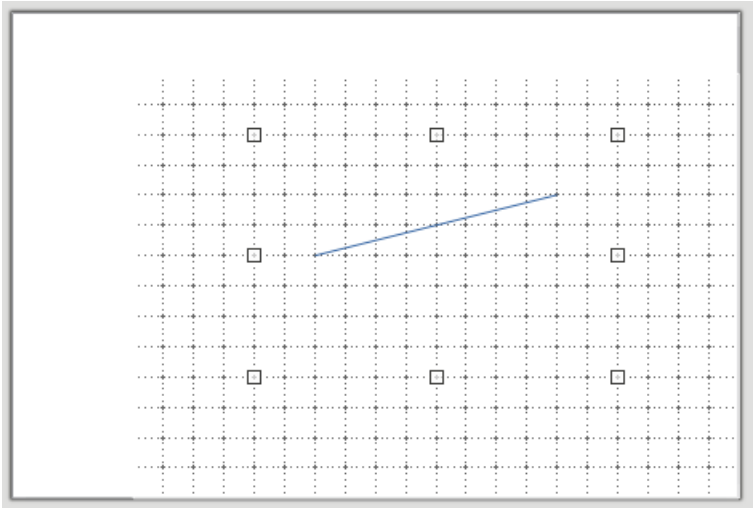


Figure 13: Snap-rectangle

Look at the status bar…

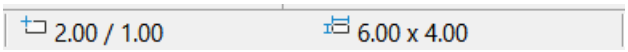


Figure 14: Status bar

… or open the 'Position and Size' dialog. The snap-rectangle has its left-top corner at (left | top) = (2.00 cm | 1.00 cm), and its size is w × h = 6 cm × 4 cm.

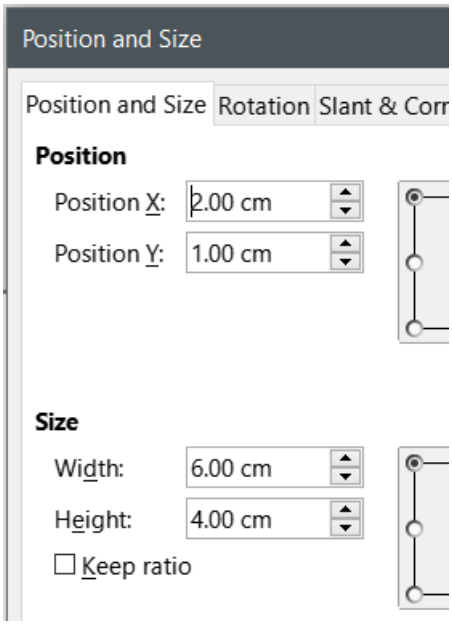


Figure 15: 'Position and Size' dialog

The position values `svg:x="4cm"` and `svg:y="2cm"` refer to the page including margins, but the values in the UI refer to the content area excluding margins.

snap-rectangle left = `svg:x="4cm"` − left page margin 2 cm = 2 cm in UI
snap-rectangle top = `svg:y="2cm"` − top page margin 1 cm = 1 cm in UI

2.5.3 Explanation – Attribute `svg:viewBox`

The attribute `svg:viewBox="10 40 180 80"` maps the local coordinate system of the shape to position (2 cm | 1 cm) and size 6 cm × 4 cm of the snap-rectangle.

Table 1: Example of manual mapping of snap-rectangle to locale coordinate system

snap-rectangle in UI	local coordinate system
left 2 cm	10
width 6 cm	180
1 cm distance in x-direction	180 / 6 = 30 distance in x-direction
top 1 cm	40
height 4 cm	80
1 cm distance in y-direction	80 / 4 = 20 distance in y-direction
	left point (40|80)
left point x = left 2 cm + 1 · 1 cm = 3 cm	40 = 10 + 1 · 30
left point y = top 1cm + 2 · 1 cm = 3 cm	80 = 40 + 2 · 20
left point (3 cm | 3 cm)	
	right point (160|60)

right point x = left 2 cm + 5 · 1 cm = 7 cm	160 = 10 + 5 · 30
right point y = top 1 cm + 1 · 1 cm = 2 cm	60 = 40 + 1 · 20
right point (7 cm | 2 cm)	

Figure 16 shows the position of the axes of the local coordinate system of the shape related to the UI rendering.

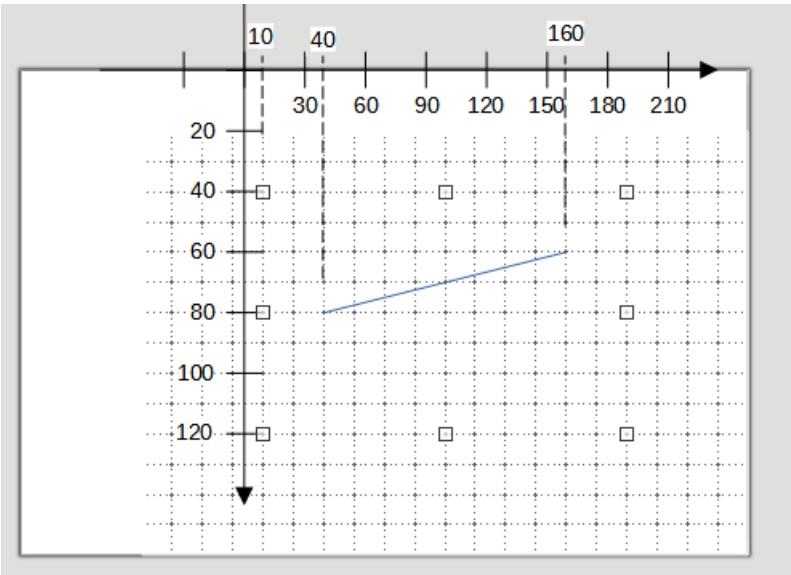


Figure 16: Local coordinate system of the shape

2.5.4 Further Remarks

Values for left and top in the `svg:viewBox` attribute may be negative in the ODF standard and LibreOffice can use it. However, foreign applications might not be able to handle it and will show nothing or fall back to zero.

Values for width and height cannot be strictly negative but may be zero. The attribute `svg:viewBox="0 0 0 0"` would disable display of the shape, if the attribute was used in SVG. But it has a special meaning in LibreOffice as it is used for import and export filter with MS Office file formats. So, do not use it for own shapes.

The attribute `svg:viewBox="0 0 21600 21600"` is used in most of the predefined custom shapes of LibreOffice. The reason is, that these shapes are designed for compatibility with the old, binary file format of MS Office, which uses these values. An advantage of value 21600 is, that it allows a lot of fractions with integer result. But you don't need to stick to it.

Other places in LibreOffice – shapes of type Polygon for example – use values in 1/100 mm directly derived from the contained points.

2.6 Exercise Solutions

2.6.1 Exercise 2.1

`A(10800|0), B(21600|14400), C(7200|18000)`

2.6.2 Exercise 2.2

It is a rhombus.

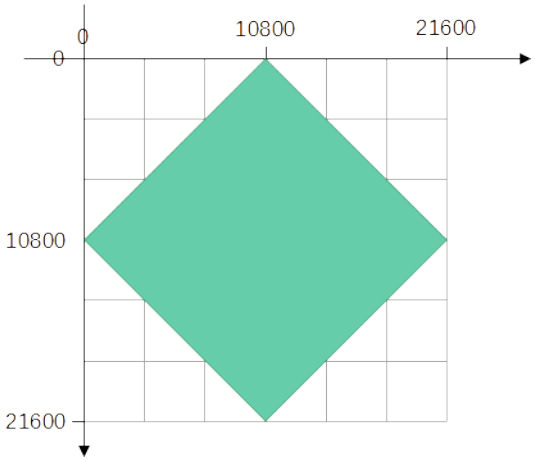


Figure 17: Rhombus

2.6.3 Exercise 2.3

The coordinates of the corner of the triangle have the coordinates `(0|14400)`, `(21600|0)` and `(7200|21600)`. There are multiple solutions depending on where you start the sub-path and whether you visit the corners clockwise or counter-clockwise. For example:

```
draw:enhanced-path="M 0 14400 L 21600 0 7200 21600 Z N"
```

```
draw:enhanced-path="M 7200 21600 L 21600 0 0 14400 Z N"
```

2.6.4 Exercise 2.4

At first glance the shapes look the same, but shape (a) – that is the predefined 'Right Triangle' – consists of four points, and shape (b) consists of three points.

(a) It has the points `(0|0)`, `(21600|21600)`, `(0|21600)` and `(0|0)`.

(b) It has the points `(0|0)`, `(21600|21600)`, `(0|21600)`.

You will notice that shape (a) has indeed four points, if you convert the shape to a polygon and then look into the status bar or drag the points of the polygon.

2.6.5 Exercise 2.5

The command L requires existence of a ‘current point’. The pen needs to be already down. That is not the case when starting the drawing of the enhanced path.

2.6.6 Exercise 2.6

The shape ‘Right Triangle’ has the glue point attribute
`draw:glue-points="0 0 0 10800 0 21600 10800 21600 21600 21600"`

For the new shape we need points left-top (0|0), right-middle (21600|10800), and left-bottom (0|21600),

```
draw:glue-points="0 0 21600 10800 0 21600"
```

3 Introduction to Handles and Equations

3.1 Linear Handle and Modifiers

Start making a new document in Draw, insert a shape ‘Right Triangle’ and save the file. Open the file in an editor and locate the element `<draw:custom-shape>`. Change the markup to that shown below, thereby keep the brown parts. Save the modified file and open it in LibreOffice.

```
<draw:custom-shape draw:name="Introductory example"
                   draw:style-name="gr1"
                   draw:text-style-name="P1"
                   draw:layer="layout"
                   svg:width="4cm"
                   svg:height="3cm"
                   svg:x="5cm"
                   svg:y="6cm">
    <text:p/>
    <draw:enhanced-geometry svg:viewBox="0 0 100 100"
                            draw:type="non-primitive"
                            draw:modifiers="50"
                            draw:enhanced-path="M $0 0 L 100 100 0 100 Z N">
       <draw:handle draw:handle-position="$0 top" />
    </draw:enhanced-geometry>
</draw:custom-shape>
```

The blue, bold parts are new. What do they mean?

You should get a shape like shown in Figure 18. Click on it and then drag the handle. You can move it horizontally and the triangle changes the position of its top vertex.

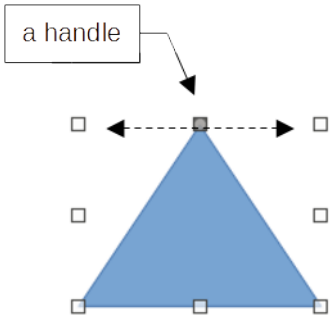


Figure 18: Custom shape with handle

The handle is generated by the element
`<draw:handle draw:handle-position="$0 top" />`

The element `<draw:handle>` is a sub-element of the `<draw:enhanced-geometry>` element. Therefore there is no longer the markup `/>` for an empty element, but there is the closing tag `</draw:enhanced-geometry>`.

The attribute `draw:handle-position` of the handle has two parameters. The first one is the x-coordinate, the second one the y-coordinate.

The value `top` for the second parameter is a constant and means the value `top` of the `svg:viewBox` attribute. So instead of `top` you could use the value `0` here as well. Of course, there are constants `left`, `right` and `bottom` for the other edges of the viewBox rectangle too. All possible constants are listed in ‘Table 13 - Handle position constants’ in section 19.179 draw:handle-position[8] in the ODF standard.

8 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_handle-position [called 2021-08-31]

The value `$0` for the first parameter is a reference to the first value in the `draw:modifiers` attribute. The value 50 in the source file is the value at time of saving. If you move the handle, this value changes. The same value is then used in the attribute `draw:enhanced-path`. As result, the start point in the ‘moveto’ command changes when you drag the handle. The values in the attribute `draw:modifiers` are counted from left to right starting with zero.

The value in the `draw:modifiers` attribute is not an absolute length, but it is a coordinate in the local coordinate system of the shape. Hence the value 50 generates an isosceles triangle here.

> **Exercise 3.1**
>
> Which value in the `draw:modifiers` attribute is needed to get a right triangle? Will it remain to be a right angle, when you change the size of the shape?
>
> → solution

The position of the handle can be set numerically in the UI. Select the shape, open the ‘Position and Size’ dialog and go to tab ‘Slant & Corner Radius’. The setting is neither a ‘slant’ nor a ‘corner radius’. But when the fields for the handle position have been implemented, this tab had enough room for the new controls.

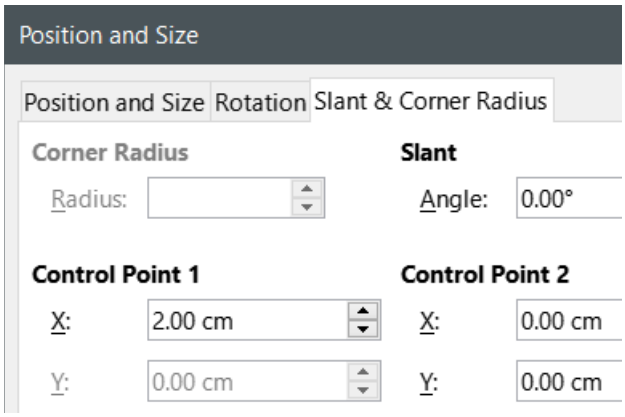


Figure 19: Fields for position of handles

‘Control Point 1’ is the handle of the shape. ‘Control Point 2’ is for the case, that the shape has a second handle. (The fact, that the fields of ‘Control Point2’ are not grayed out here, is an error in that dialog.)

You see that the value in the field is not a value of the local coordinate system of the shape, but the coordinate of the handle in the UI. Thereby, the value is measured from the left edge of the snap-rectangle of the shape.

> **Exercise 3.2**
>
> Which value do you need to enter to get a right triangle?
>
> → solution

> **Exercise 3.3**
>
> There are two more values beside ‘2cm’, which make the triangle an isosceles triangle. Calculate them. Tip: use Pythagoras’ theorem.
>
> Does the triangle remain a right triangle when you change height or change width of this shape?
>
> → solution

A reference to a value in the `draw:modifiers` attribute may be used in attributes `draw:glue-points` and `draw:text-areas` too. To get a glue point that remains at the tip of the rectangle you can use `draw:glue-points=" $0 0"`, for example.

Shapes with handles exist in the OOXML standard too. There the modifiers are called '**adjustment values**'. This term is used in the code of LibreOffice as well.

3.2 Equations

Now we will change the triangle of section 3.1 so that the text-area is bound to the top vertex of the triangle and moves along with it, when the top is moved. The width of the text-area shall be 50% of the width of the triangle and the height 25% of the height of the triangle. It shall be placed so, that the tip of the triangle is in the middle of the bottom edge of the text-area.

Figure 20 shows a sketch including local coordinate system and viewBox.

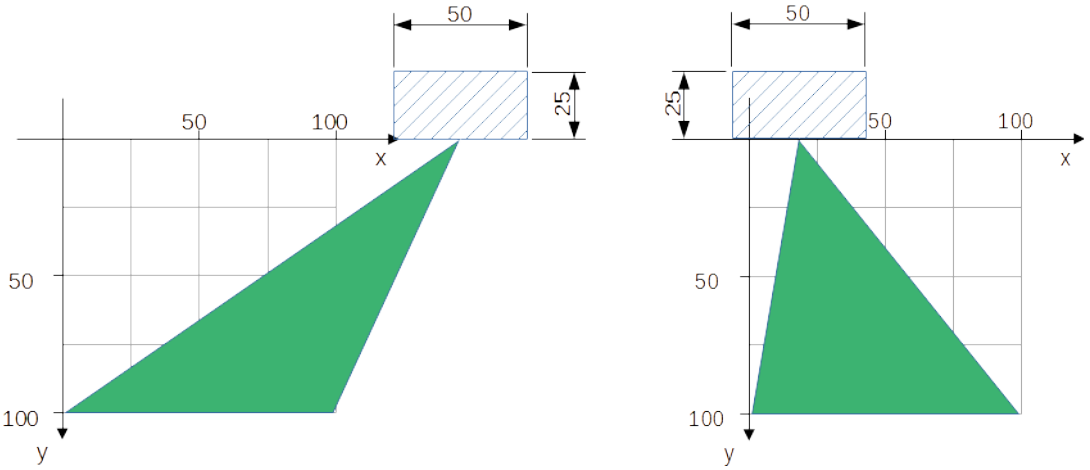


Figure 20: Desired text-area is hatched

Top edge coordinate = −25, bottom edge coordinate = 0.

Left edge coordinate = `$0` − 25, right edge coordinate = `$0` + 25.

The left and right edge coordinate require a mathematical term. But the attribute `draw:text-areas` – and attributes `draw:enhanced-path` and `draw:glue-point` neither – allow mathematical terms in the point coordinates.

The solution is the element `<draw:equation>`. It is a sub-element of the `<draw:enhanced-geometry>`.

`<draw:equation>` elements for the terms `$0` − 25 and `$0` + 25 could e.g. be written as

```
<draw:equation draw:name="f0" draw:formula="$0-25"/>
<draw:equation draw:name="f1" draw:formula="$0+25"/>
```

So a `<draw:equation>` element has the parameter `draw:name`, which provides an identifier for the formula, and parameter `draw:formula`, which contains the mathematical expression to be evaluated.

The result of the evaluation can be used in point coordinates and in other expressions by writing a reference to the formula. A reference is constructed by the character `?` immediately followed by the identifier of the equation.

The desired text-area is then defined by `draw:text-areas="?f0 -25 ?f1 0"` (left top right bottom).

The complete `<draw:enhanced-geometry>` element for the triangle is then

```
<draw:enhanced-geometry svg:viewBox="0 0 100 100"
                        draw:text-areas="?f0 -25 ?f1 0"
                        draw:type="non-primitive"
                        draw:modifiers="128"
                        draw:enhanced-path="M $0 0 L 100 100 0 100 Z N">
    <draw:equation draw:name="f0"
                   draw:formula="$0-25"/>
    <draw:equation draw:name="f1"
                   draw:formula="$0+25"/>
    <draw:handle draw:handle-position="$0 top"/>
 </draw:enhanced-geometry>
```

You can place as many of the `<draw:equation>` elements as you need before the `<draw:handle>` elements. The `<draw:equation>` elements are evaluated in the order they are written in the source file.

The ODF standard allows to use arbitrary identifiers, and using meaningful identifiers helps to understand the purpose of a formula. Isn't the following easier to understand?

```
draw:text-areas="?left -25 ?right 0"
<draw:equation draw:name="left" draw:formula="$0-25"/>
<draw:equation draw:name="right" draw:formula="$0+25"/>
```

Sadly, when LibreOffice opens the file, it throws away the existing identifiers and enumerates the equation as `f0`, `f1`, `f2`, …. when it saves the file. Therefore it is very important to document your own shape as described in section 2.2.

The OOXML standard uses equations in its shapes too. In this case, the element is called **guide.**

3.3 Restriction of Handle Movement

Start a new document. Insert the shape created in section 3.2 and insert the predefined shape 'Isosceles Triangle' from the 'Basic Shapes' set. Drag the handle far away from the base of the triangle. Notice that this is possible with your own shape but not with the predefined one. The handle can only be dragged to the left or right edge of the snap-rectangle in the shape 'Isosceles Triangle'.

Such restrictions are done by the attributes `draw:handle-range-x-maximum` and `draw:handle-range-x-minimum` for the horizontal direction and `draw:handle-range-y-maximum` and `draw:handle-range-y-minimum` for the vertical direction. These are attributes of the `<draw:handle>` element.

Possible values of these attributes are the same as in a parameter of the `draw:handle-position` attribute. That is, you can use a number, one of the above-mentioned constants, a reference to a modifier or a reference to a formula.

With the restriction of the handle movement to the range of the snap-rectangle, the complete `<draw:enhanced-geometry>` element would be

```
<draw:enhanced-geometry svg:viewBox="0 0 100 100"
                        draw:text-areas="?f0 -25 ?f1 0"
                        draw:type="non-primitive"
                        draw:modifiers="128"
                        draw:enhanced-path="M $0 0 L 100 100 0 100 Z N">
    <draw:equation draw:name="f0"
                   draw:formula="$0-25"/>
    <draw:equation draw:name="f1"
                   draw:formula="$0+25"/>
    <draw:handle draw:handle-position="$0 top"
                 draw:handle-range-x-minimum="left"
                 draw:handle-range-x-maximum="right" />
 </draw:enhanced-geometry>
```

Exercise 3.4

With the predefined parallelogram in the ‘Basic Shapes’ set, you can move the top edge horizontally. We want a similar parallelogram but for vertical shearing.

Create an `<draw:enhanced-geometry>` element, so that the right edge can be moved up and down. The parallelogram shall become a rectangle at the topmost position of the handle and a line segment at the lowest position.

→ solution

Figure 21: Parallelogram with vertical edge-movement

3.4 Overflow

In the shear taught in mathematics class, neither the base nor the opposite edge of the parallelogram change their length, but the opposite edge is only shifted. To get this kind of shear, you could make the design as shown in Figure 22

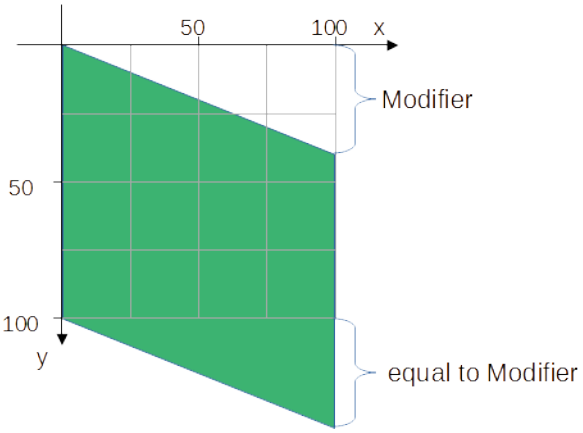


Figure 22: Parallelogram with fix base length

and use the following markup

```
<draw:enhanced-geometry
            svg:viewBox="0 0 100 100" draw:type="non-primitive"
            draw:modifiers="40"
            draw:enhanced-path="M 0 0 L 100 $0 100 ?f0 0 100 Z N">
    <draw:equation draw:name="f0" draw:formula="100+$0"/>
    <draw:handle draw:handle-position="right $0"/>
</draw:enhanced-geometry>
```

As you can see in Figure 22, the drawn parallelogram overflows the snap-rectangle. That is no problem in Draw/Impress but might be in text documents or spreadsheets.

Create this shape and insert it in a text document with lot of dummy text. Set the shape anchor type to ‘as character’. Notice that the automatic recalculation of the line height is based on the snap-rectangle. The parallelogram overflows the text above or below. You can tweak the wrap behavior by adding a top or bottom spacing in the wrap properties.

Figure 23: Spacing in wrap properties

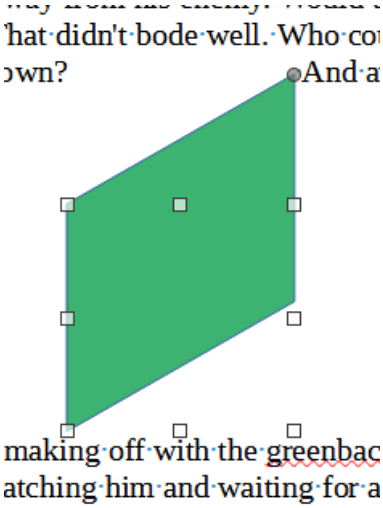


Figure 24: Wrap spacing top 2 cm

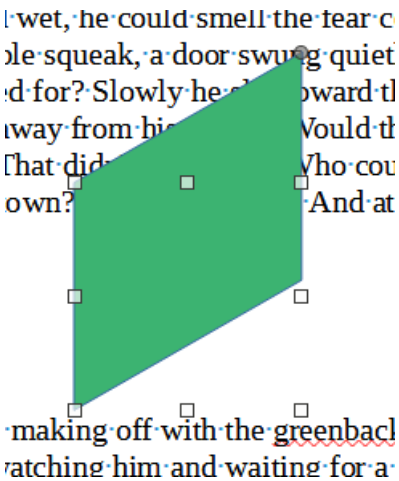


Figure 25: no wrap spacing

A similar problem exists in spreadsheets. Insert the shape into a cell and anchor it ‘to cell’. Then use tool ‘Fit to Cell Size’ from the context menu of the shape. The size is adapted so that the snap-rectangle fits into the cell. The overflowing drawing is not considered.

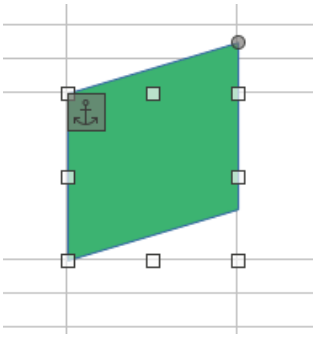


Figure 26: Fit to Cell Size

3.5 Functions

We will create a right triangle with horizontal base by using a Thales' circle.

In the first version we use only the upper half circle. We set the viewBox rectangle twice as wide as high. Of cause the snap-rectangle needs the same ratio for to get really a right triangle.

The sketch in Figure 27 indicates the needed calculations. For the coordinates of point $P(x_P \mid y_p)$ in respect to the center of the circle as origin we get $y_p = \sqrt{50^2 - (x_P)^2}$.

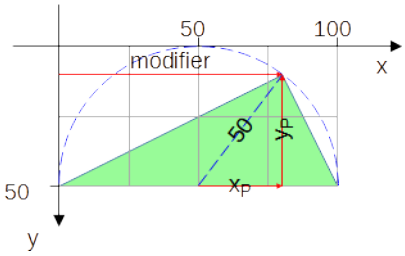


Figure 27: Thales' half circle

The Table 2 lists the functions, which are available for use in formulas. It is cited[9] from the ODF standard.

Table 2: Available functions for attribute draw:formula

Markup	Meaning
`abs(n)`	returns the absolute value of n
`sqrt(n)`	returns the positive square root of n
`sin(n)`	returns the trigonometric sine of n, where n is an angle specified in radians
`cos(n)`	returns the trigonometric cosine of n, where n is an angle specified in radians
`tan(n)`	returns the trigonometric tangent of n, where n is an angle specified in radians
`atan(n)`	returns the arc tangent of n in radians
`min(x,y)`	returns the smaller of two values
`max(x,y)`	returns the greater of two values
`atan2(y,x)`	returns the angle in radians of the vector (x,y) with the x-axis
`if(c,x,y)`	conditional testing: if c is greater than zero then the result of evaluating x is returned, otherwise the result of evaluating y is returned.

For our triangle with point P as handle the idea is now:

modifier `$0`	x-coordinate of handle
`f0=$0-50`	value x_P = modifier − 50

9 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_formula [called 2021-09-01]
It is from Table 12 in section '19.171 draw:formula' in part 2.

`f1=sqrt(50*50-?f0*?f0)`	value y_P. There exists no power-function, we need to multiply
`f2=50-?f1`	y-coordinate of handle = 50 − y_P

The x-coordinate of the handle needs to be restricted to the range [0;100]. Otherwise the value x_P will be outside the range [−50;50] and the radicand of the square root would be negative.

> **Exercise 3.5**
>
> Create the markup of a suitable `<draw:enhanced-geometry>` element.
>
> → solution

In the second version we use a full circle. This cannot be solved with a horizontal modifier because you would not be able to differentiate, whether the vertex of the triangle is in the upper or lower half circle.

The sketch in Figure 28 shows a suitable idea. The handle is now freely movable, which implies, that there need to be two modifiers. The vertex of the triangle is then the intersection point of the ray from the center through the handle position.

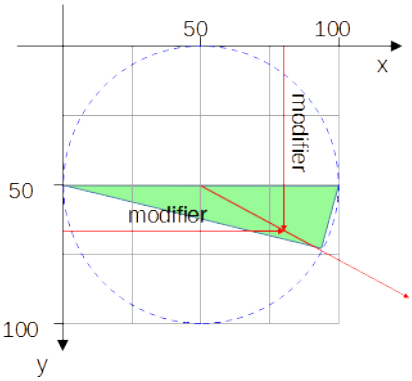


Figure 28: Full Thales' circle

Calculation is firstly straight forward. Get an angle α from the handle position using the `atan2` function. Then calculate the position of the vertex using cos(α) and sin(α).

But atan2(0,0) is not defined. The ODF standard does not specify a replacement value for this case. We could set the tip of the triangle same as for α=0 in this case. The challenge is to detect this case. There exist no comparison operators and no logic operators. The only possibility is to use the `if` function with its ‘is positive’ condition.

I have added the idea here, because we will see the same idea later, when we discuss commands to draw circles and arcs, and because it introduces you to the formula language of custom shapes.

> **Exercise 3.6**
>
> Create the markup of a suitable `<draw:enhanced-geometry>` element. You should at least be able to do so, if you do not consider the special case of atan2(0,0). You can afterwards find an idea for the special case in the exercise solution.
>
> → solution

3.6 Polar Handles

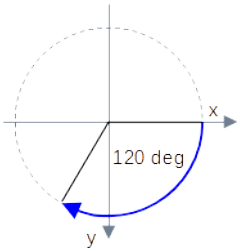
A polar handle does not move horizontally or vertically but on a circular path around a center. Its position is not specified by Cartesian coordinates (x|y) but by polar coordinates (radius, angle). A handle is a ‘polar handle’ if it has the attribute `draw:handle-polar`.

Markup example for a polar handle

```
<draw:handle
        draw:handle-polar="20 40"
        draw:handle-position="30 120" />
```

The `draw:handle-polar` attribute specifies the center of the circle. In the example the center has the coordinates (20|40).

The `draw:handle-position` attribute specifies the radius of the circle in its first parameter and the angle in degree in its second parameter. The angle is measured from the positive x-axis towards the positive y-axis. Because the y-axis points down, this is clockwise on screen.



Caution, the order radius/angle is different form the current contents in section ‘19.179 draw:handle-position’ in ODF 1.3, part 3. But this order is actually used not only by LibreOffice but also by other applications.

The radius can be restricted by the attributes `draw:handle-radius-range-minimum` and `draw:handle-radius-range-maximum`. The angle is used modulo 360. An angle of 230 deg results in the same handle position as the angle −130 deg, for example.

We use the Thales’ circle problem again. If you tried the solution from Exercise 3.6, you noticed that the handle movement is not intuitive in that solution. You will see that a polar handle moves much better along the Thales’ circle.

The markup of the `<draw:enhanced-geometry>` could be

```
<draw:enhanced-geometry svg:viewBox="0 0 100 100" draw:type="non-primitive"
            draw:modifiers="120"
            draw:enhanced-path="M 0 50 L ?f0 ?f1 100 50 Z N">
    <draw:equation draw:name="f0"
                   draw:formula="50+50*cos($0*pi/180)" />
    <draw:equation draw:name="f1"
                   draw:formula="50+50*sin($0*pi/180)" />
    <draw:handle draw:handle-polar="50 50"
                 draw:handle-position="50 $0"/>
</draw:enhanced-geometry>
```

To draw the line segment of the triangle, we need Cartesian coordinates. We get them the same way as in Exercise 3.6 only that the result of function `atan2` is an angle in radians and here we have an angle in degrees. To fit the trigonometric functions `sin` and `cos`, the values in degree needs to be converted. This is done by the factor `pi/180`. The identifier `pi` is a ready to use constant.

The above markup has no glue points defined. Consider adding glue points for the center of the circle and for the vertex of the triangle. That would allow to add a connector to indicate the radius line segment.

You have seen that the ‘Position and Size’ dialog shows positions of linear handles. Does it show the Cartesian coordinates of a polar handle? Can you enter coordinates, so that a desired angle is stored? Create a shape with the above markup, set its size to 4 cm width and 4 cm height and test it. You will find that the ‘Position and Size’ dialog it not yet usable with polar handles.

3.7 Further Handle Attributes

The ODF standard specifies the attributes `draw:handle-mirror-horizontal`[10] and `draw:handle-mirror-vertical`[11]. These attributes are preserved by LibreOffice, but not evaluated.

The ODF standard specifies the attribute `draw:handle-switched`[12]. If `true`, then the handle directions are swapped if shape height exceeds shape width. I do not have a meaningful example yet.

3.8 Exercise Solutions

3.8.1 Exercise 3.1

You get a right triangle for modifier values `0` and `100`. Because resizing is a transformation in vertical or horizontal direction and the arms of the right angle are vertical and horizontal too, it remains a right angle.

3.8.2 Exercise 3.2

The triangle becomes a right triangle if the handle is it at the left or at the right edge of the snap-rectangle. So you need to enter 0 cm or 4 cm to get a right triangle.

3.8.3 Exercise 3.3

$x = \sqrt{4^2 - 3^2} = \sqrt{7} \approx 2.646$

The needed value for the left solution is $-2.646\,\text{cm}$. You can enter the negative value in the dialog, but on reopening the dialog, it will show zero. The dialog needs obviously some improvements.

For the right solution you get 6.646 cm. Remember that the values in the field are relative to the left edge of the snap-rectangle.

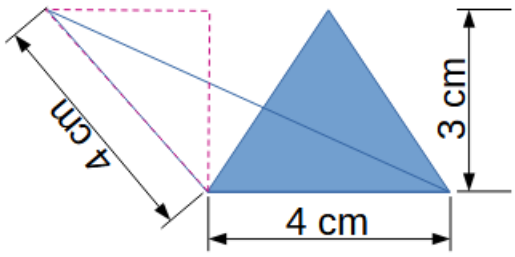


Figure 29: Triangle for Pythagoras' theorem

If you change either width or height, the hypotenuse will change with a different factor than width or height. Therefore the triangle will not stay isosceles. It becomes obvious when you drag the shape to a huge width or height. You would need to calculate a new value for the handle position.

10 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_handle-mirror-horizontal

11 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_handle-mirror-vertical

12 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_handle-switched

3.8.4 Exercise 3.4

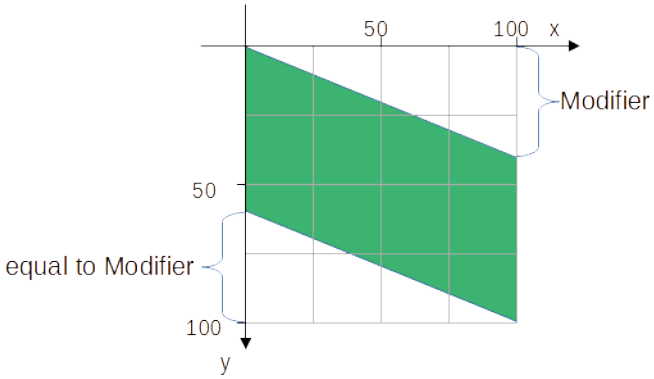


Figure 30: height of parallelogram = height of snap-rectangle

The length of the vertical edges are reduced when shearing becomes larger.

```
<draw:enhanced-geometry svg:viewBox="0 0 100 100" draw:type="non-primitive"
            draw:modifiers="40"
            draw:enhanced-path="M 0 0 L 100 $0 100 100 0 ?f0 Z N">
    <draw:equation draw:name="f0" draw:formula="100-$0"/>
    <draw:handle draw:handle-position="right $0"
                 draw:handle-range-x-minimum="top"
                 draw:handle-range-x-maximum="bottom"/>
</draw:enhanced-geometry>
```

3.8.5 Exercise 3.5

```
<draw:enhanced-geometry svg:viewBox="0 0 100 50" draw:type="non-primitive"
            draw:modifiers="70"
            draw:enhanced-path="M 0 50 L $0 ?f2 100 50 Z N">
    <draw:equation draw:name="f0" draw:formula="$0-50"/>
    <draw:equation draw:name="f1" draw:formula="sqrt(50*50-?f0*?f0)"/>
    <draw:equation draw:name="f2" draw:formula="50-?f1"/>
    <draw:handle draw:handle-position="$0 ?f2"
                 draw:handle-range-x-minimum="left"
                 draw:handle-range-x-maximum="right"/>
</draw:enhanced-geometry>
```

3.8.6 Exercise 3.6

Modifier values are given in the local coordinate system. Hence you need to change them to a coordinate system, where the center of the circle is the origin. And after doing the trigonometric calculations, you need to bring the values back to the local coordinate system of the shape.

To convert the condition ($x \neq 0 \vee y \neq 0$) into the formula language is a challenge. Table 3 below gives some general hints how to translate conditions to the possibilities of the formula language of custom shapes.

```
<draw:enhanced-geometry svg:viewBox="0 0 100 100" draw:type="non-primitive"
            draw:modifiers="60 80"
            draw:enhanced-path="M 0 50 L ?f3 ?f4 100 50 Z N">

    <draw:equation draw:name="f0" draw:formula="$0 - 50" />
    <draw:equation draw:name="f1" draw:formula="$1 - 50" />
    <draw:equation draw:name="f2"
```

```
                    draw:formula="if(abs(?f0)+abs(?f1),atan2(?f0,?f1),0) />
    <draw:equation draw:name="f3" draw:formula="50+50*cos(?f2)"/>
    <draw:equation draw:name="f4" draw:formula="50+50*sin(?f2)"/>
    <draw:handle draw:handle-position="$0 $1"/>
</draw:enhanced-geometry>
```

Table 3: Translate conditions to formula language

Condition	Translation
if (a > b) then A else B	$a>b \Leftrightarrow a-b>0$ `if(a-b,A,B)`

if (a ≥ b) then A else B	$a \geq b \Leftrightarrow \neg(a<b) \Leftrightarrow \neg(b>a) \Leftrightarrow \neg(b-a>0)$
	For negation exchange the branches.
	if (Cond.) then A else B ⇔ if (¬Cond.) then B else A
	`if(b-a,B,A)`

if (a ≠ b) then A else B	$a \neq b \Leftrightarrow a-b \neq 0 \Leftrightarrow \lvert a-b \rvert >0$ `if(abs(a-b),A,B)`
if (a = b) then A else B	Negation of (a ≠ b) `if(abs(a-b),B,A)`
if (a ≠ b ∨ c ≠ d) then A else B	Sum of non-negative values is only positive if at least one value is positive `if(abs(a-b)+abs(c-d),A,B)`
if (a = b ∧ c = d) then A else B	Negation of (a ≠ b ∨ c ≠ d) `if(abs(a-b)+abs(c-d),B,A)`
if (a ≠ b ∧ c ≠ d) then A else B	Product of non-negative values is only positive of both factors are positive `if(abs(a-b)*abs(c-d),A,B)`
if (a = b ∨ c = d) then A else B	Negation of (a ≠ b ∧ c ≠ d) `if(abs(a-b)*abs(c-d),B,A)`

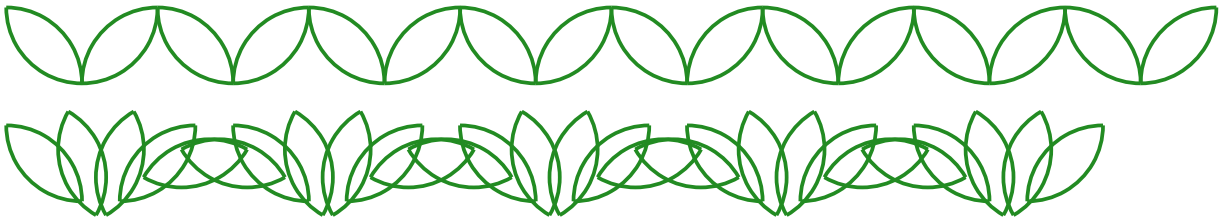
4 Quadrants

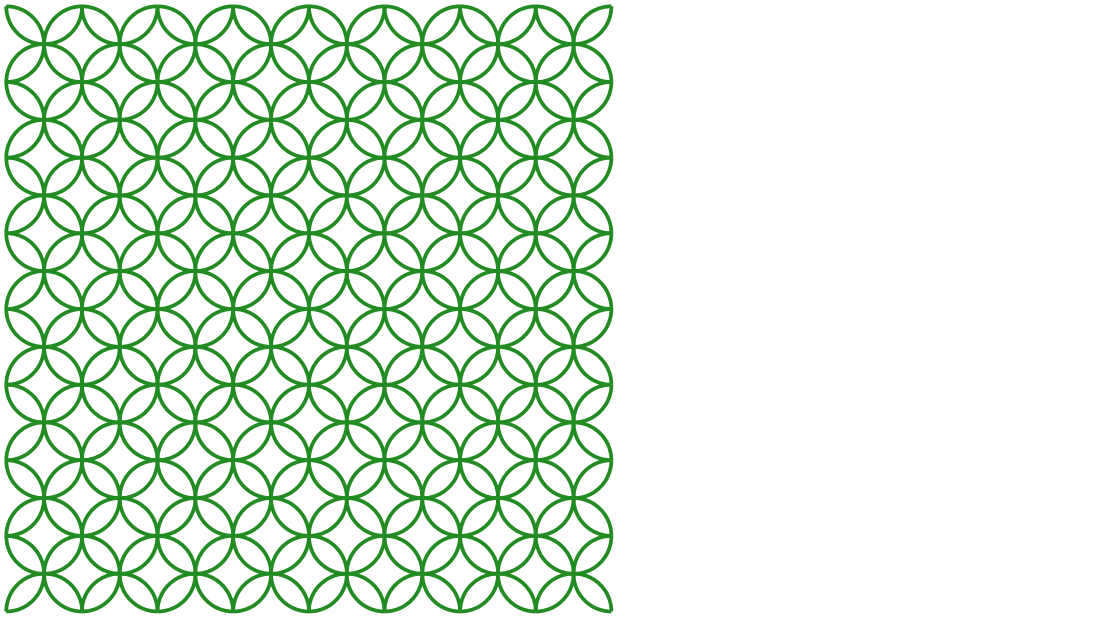
4.1 Introduction

After a lot of mathematics in chapter 3 the examples in this chapter are artistic. Let’s create tiles like this  , for example.

You can mirror and rotate tiles and combine them to borders and wallpapers. The tool Edit > Duplicate in Draw can help you in that.

Examples:





The quadrants will be used in addition to the already known lines in chapter 5 “Fill and Stroke”.

4.2 Single Quadrant

The ODF standard provides the commands X and Y for drawing quadrants. Their definition is in “Table 10 - Enhanced path commands”[13]

Table 4: Description of commands X and Y

Command	Name	Parameters	Description
X	elliptical-quatrantx	(x y) +	Draws a quarter ellipse, whose initial segment is tangential to the x-axis from the current point to (x, y). For each additional quarter ellipse command, the axis to which the segment is tangential to switches from x to y and from y to x.
Y	elliptical-quadranty	(x y) +	Draws a quarter ellipse, whose initial segment is tangential to the y-axis from the current point to (x, y). For each additional quarter ellipse command, the axis to which the segment is tangential to switches from y to x and from x to y

Some examples will show how they are used. Start with a custom shape ‘Rectangle’, set its area fill to ‘None’ and line width to 0.5 mm. Use a square size, edge length 4 cm will work well.

Open the file for editing the element `<draw:enhanced-geometry>`. Use the markup

```
<draw:enhanced-geometry
        svg:viewBox="0 0 100 100"
        draw:type="non-primitive"
        draw:enhanced-path=" ... "/>
```

The commands X and Y expect a ‘current point’. Hence our path starts with a ‘moveto’-command. Set the pen to the center of the viewBox rectangle. Our first target point is (0|25). With command X the markup is then

```
draw:enhanced-path="M 50 50 X 0 25"
```

What is drawn? The wording “tangential to the x-axis“ might not be clear. It means, that the segment leaves the current point horizontally, namely to the side where the target point is placed.

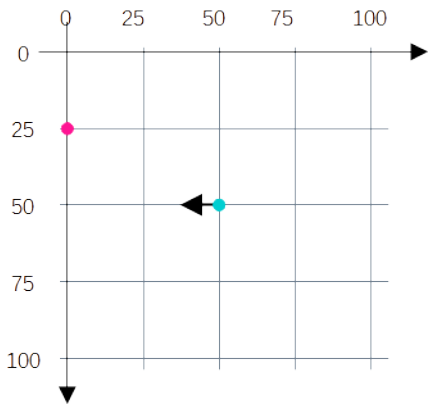


Figure 31: Start of command X

The current point and the target point determine the center of an ellipse. The ellipse meets current point and target point. The drawn segment is the short elliptic arc from current point to target point.

13 in section 19.145 draw:enhanced-path in part3, ODF 1.3. https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_enhanced-path [called 2021-09-04]

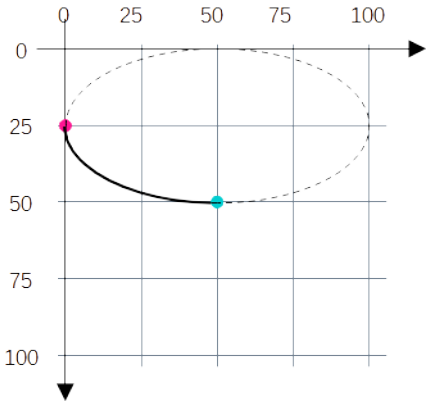


Figure 32: Elliptic arc with 90deg center angle

Exercise 4.1

Note down the complete value for the `draw:enhanced-path` attribute to get the segment in Figure 33. Which point is center of the circle? Which point needs to become ‘current point’ and which is target point, if you use the X command?
→ solution

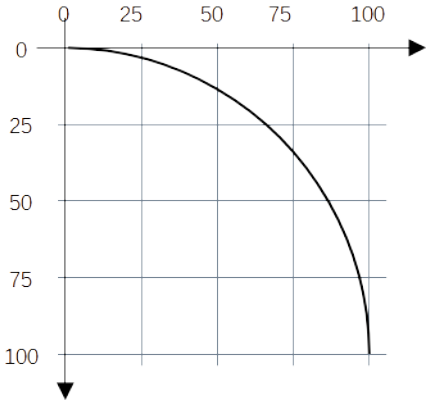


Figure 33: Quadrant of a circle

The Y command works similar, only that the segment leaves the current point vertically. So you can get the quadrant of Figure 33 using the Y command as well. Start in point (100|100), go vertically up towards point (0|0). The according markup would be
`draw:enhanced-path="M 100 100 Y 0 0"`

The difference between the two solutions become visible, if you apply a line style with arrowhead.

4.3 Sequence of Quadrants

You have learned that it is allowed to omit the character of a command, if several identical commands follow each other immediately. The path `"L 10 30 50 20 10 90"` means the same as the path `"L 10 30 L 50 20 L 10 90"`.

The same rule applies to command X and Y in principle. The path `"M 50 0 X 100 50 50 100 0 50 50 0"` means the same as path `"M 50 0 X 100 50 X 50 100 X 0 50 X 50 0"`.

But the resulting shape is Figure 35 and not Figure 34. Why?

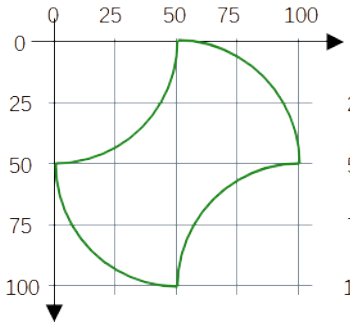


Figure 34: Forced X direction

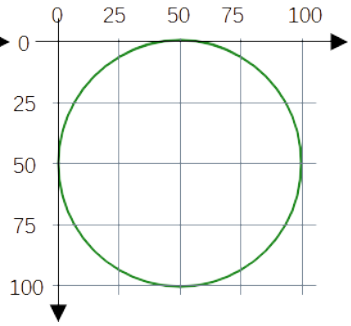


Figure 35: Automatic X Y toggle

That is the sentence “For each additional quarter ellipse command, the axis to which the segment is tangential to switches from x to y and from y to x.” in the ODF standard[13]. So in fact the commands X and Y do the same. They draw a sequence of elliptic quadrants with alternating between horizontal and vertical start direction for the segments. Only that command X starts with horizontal direction and command Y starts with vertical direction.

To get the drawing of Figure 34, you need a different command than X or Y in between. You can use command M for example. That lifts off the pen and you start newly. In fact Figure 34 was created by the path `"M 50 0 X 100 50 M 100 50 X 50 100 M 50 100 X 0 50 M 0 50 X 50 0 N"`.

Exercise 4.2

Write down a complete `<draw:enhanced geometry>` element for a custom shape to show a heart like in Figure 36. Restrict the text area to the hatched square as discussed in chapter 2.3
1. Tip: Use a suitable size of the viewBox
2. Tip: Start with the intersection point of the two half circles.

→ solution

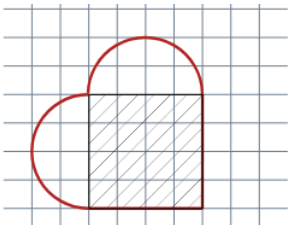


Figure 36: Heart

Exercise 4.3

Write down a complete `<draw:enhanced geometry>` element for a custom shape to show the blue drawing of Figure 37. The inner circle shall touch the outer quadrants. For to accomplish this, the distances as in Figure 37 has to be used. Do not calculate $/\sqrt{2}$ by yourself but use a `<draw:equation>` element as discussed in chapter 3.

→ solution

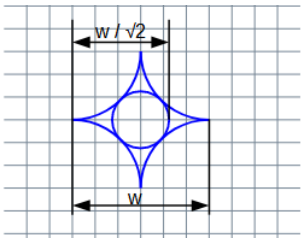
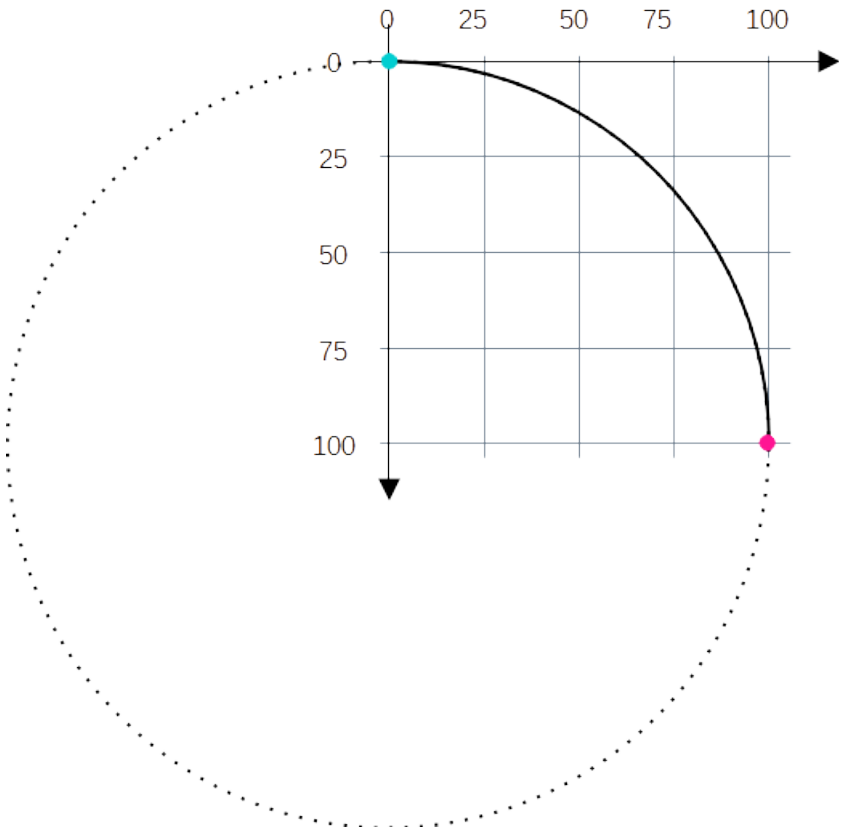


Figure 37: Diamond with circle

4.4 Exercise Solutions

4.4.1 Exercise 4.1




```
draw:enhanced-path="M 0 0 X 100 100"
```

Point (0|100) is center of the circle.

4.4.2 Exercise 4.2

The value for the enhanced-path might be very long. You can insert line breaks and additional spaces. And you can write a comma between parameters – not between command and parameter. These things are not permanent but help you to not lose track while editing.

The needed points are at ⅓ and ⅔ of the width. Therefore, a viewBox width is suitable, which is divisible by 3.

Remember, the values for the text area have the meaning ‘left top right bottom’.

```
<draw:enhanced-geometry
    draw:type="non-primitive"
    svg:viewBox="0 0 60 60"
    draw:text-areas="20 20 60 60"
    draw:enhanced-path="M 20,20 Y 40,0 60,20
                        L 60,60 20,60
                        X 0,40 20,20" />
```

This `<draw:enhanced-geometry>` element has no child elements. Therefore it ends with `/>`.

Other solutions are possible.

4.4.3 Exercise 4.3

This solution draws first the outer quadrants and second the inner circle.

```
<draw:enhanced-geometry
    draw:type="non-primitive"
    svg:viewBox="0 0 60 60"
    draw:enhanced-path="M 0,30 X 30,0 60,30 30,60 0,30
                        M ?f0,30 Y 30,?f0 ?f1,30 30,?f1 ?f0,30" >
    <draw:equation draw:name="f0" draw:formula="60/sqrt(2)" />
    <draw:equation draw:name="f1" draw:formula="60-?f0" />
</draw:enhanced-geometry>
```

This `<draw:enhanced-geometry>` element has child elements. Hence it needs an end tag.

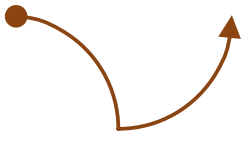
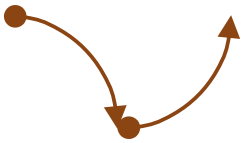
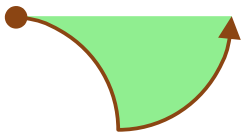
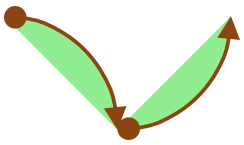
5 Fill and Stroke

5.1 Set of Sub-paths

To get a path with two quadrants with same start direction, you cannot use two commands X following directly each other, but you need a command in-between. You can use a command M or a command L. Does that work the same way? Use a graphic style with no area fill and no line arrowheads and then try it out.

Figure 38: Path with two commands X

Now change the line style of the object so, that it has a ‘Circle’ at start and an ‘Arrow’ at end and set the area fill to some color. You will get this

	`"M 0 0 X 30 30 L 30 30 X 60 0"`	`"M 0 0 X 30 30 M 30 30 X 60 0"`
not filled		
filled		

In the solution with command L the entire path is treated as one sub-path. In the solution with command M it is treated as two sub-paths.

Rule: The command M always starts a new sub-path.

Now look at the fill. In a custom shape you don't need a closed path to fill it. It is different from curve and polygon objects. The start point and the end point of a sub-path are connected by an imaginary straight-line segment and the area of this thus ‘closed’ sub-path is filled.

Rule: Each sub-path determines an own fill area. Thereby the area is bounded by the sub-path itself and a fictive straight-line segment from start point to end point.

Next we will generate the shape in Figure 39.

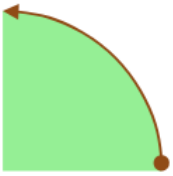


Figure 39: Orientated arc with associated sector

A trial with path value `"M 30 30 Y 0 0 L 30 0 Z"` will not work. The sector gets lines and the arc has no longer arrowheads because the path is closed. The values `"M 30 30 Y 0 0 L 30 0"` and `"M 30 0 L 30 30 Y 0 0"` will work neither. We need something to suppress fill of the arc and lines of the sector. Look at “Table 10 - Enhanced path commands“ in the ODF standard[14]. You will find (in parts cited from there):

14 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#refTable9
Table “ Enhanced path commands” in section “19.145 draw:enhanced-path” in part3 of ODF standard 1.3.

Table 5: Description of commands F and S

Command	Name	Description
F	nofill	Specifies that the current set of sub-paths will not be filled.
S	nostroke	Specifies that the current set of sub-paths will not be stroked.

"stroke" is the technical term for what is called a "line" in the user interface.

We now use value "`M 30 30 Y 0 0 L 30 0 Z`" for the sector and value "`M 30 30 Y 0 0`" for the arc and add the F and S commands.

Try value "`M 30 30 Y 0 0 F M 30 30 Y 0 0 L 30 0 Z S`". You get – nothing. The F and S commands are applied to both arc and sector so that actually nothing is drawn. The description contains “current **set** of sub-paths”. Arc and sector need to be in different sets. To start a new set, use the command N (again partly cited from ODF standard[14]).

Table 6: Description of command N

Command	Name	Description
N	endpath	Ends the current set of sub-paths. The sub-paths will be filled by using the “even-odd” filling rule. Other following subpaths will be filled independently.

(“even-odd” filling rule will be discussed in section 5.2)

Next try is value "`M 30 30 Y 0 0 F N M 30 30 Y 0 0 L 30 0 Z S`". You get a shape like Figure 40. The sector is in front of the arc and partly hides the line ends. A set of sub-paths is always drawn in front of the sub-paths from previous sets. We need it swapped, first draw sector then arc.

Figure 40: Sector in front of arc

Value "`M 30 30 Y 0 0 L 30 0 Z S N M 30 30 Y 0 0 F`" gives the desired shape.

5.2 Overlapping Areas

If the enhance-path value has several sub-paths areas overlap can happen. If the sub-paths belong to the same set, the “even-odd” filling rule is used for the areas. The lines of the paths are always drawn on top of the area fill for sub-paths in the same set.

If sub-paths belong to different sets, that is a command N is in-between, then first the painting of each set is determined, and these paintings are then drawn on top of each other. Thereby the sets are drawn in the order they appear in the attribute value.

5.2.1 Example for ‘even-odd’ Filling Rule

The shape in Figure 41 was generated by the markup

```
draw:enhanced-path=
    "M 30 0 L 0 20 30 40 Z
     M 40 0 10 20 40 40 Z
     M 0 0 40 20 0 40 Z"
```

Figure 41: Example even-odd filling rule

The shape has three large triangles. Figure 42 shows them as hatched triangles to highlight how many areas overlap. If an odd number of areas overlap – including case of one area – then the points in the overlapping part are drawn. If an even number of areas overlap – including case of no area –, then the points in the overlapping part are not drawn, this overlapping part becomes transparent.

If you search internet for “even-odd filling rule”, you will get formal definitions using a ray[15]. For the purpose of making your own custom-shapes it is easier to count the number of areas involved in the overlap.

Figures 43 and 44 show counting the areas involved in an overlap.

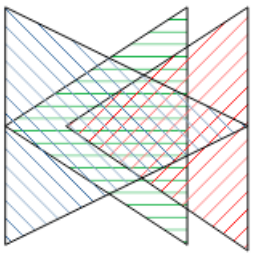


Figure 42: Triangles hatched for counting involved areas

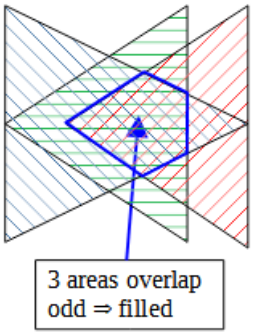


Figure 43: Odd number of involved areas

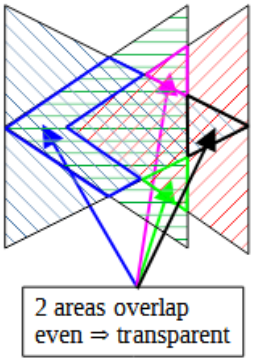


Figure 44: Even number of involved areas

Exercise 5.1

Create a shape with Yin-Yang symbol[16].

The Yin-Yang symbol has two colors – Black and White. That is not possible with one custom-shape. Create the shape so the white areas are transparent instead. Then you get the desired Yin-Yang symbol, if you show the shape in front of a white background. That might be the document background, or you use a white circle, which you place behind the Yin-Yang symbol.

→ solution

15 See https://www.w3.org/TR/SVG11/painting.html#FillProperties, for example.
16 For Yin-Yang symbol see e.g. https://en.wikipedia.org/wiki/Yin_and_yang

5.2.2 Example for Stacked Filled Areas

Change the markup of previous example so, that it has three sets of sub-paths, in fact each set has one sub-path.

```
draw:enhanced-path
="M 30 0 L 0 20 30 40 Z N M 40 0 10 20 40 40 Z N M 0 0 40 20 0 40 Z"
```

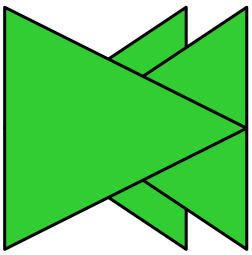


Figure 45: Stacked filled areas

The shape looks as in Figure 45.

The areas behind still exist and are visible, if you set some transparency for the fill, here 75%. The color will then accumulate.

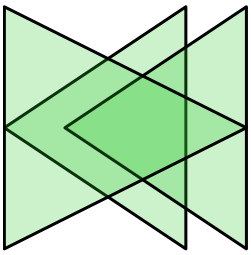


Figure 46: Stacked filled areas with transparency

Exercise 5.2

How looks the shape, if we make only two sets? Try to sketch the result. Then test whether you are right by creating the shape.

a) `draw:enhanced-path`

```
="M 30 0 L 0 20 30 40 Z M 40 0 10 20 40 40 Z N M 0 0 40 20 0 40 Z"
```

b) `draw:enhanced-path`

```
="M 30 0 L 0 20 30 40 Z N M 40 0 10 20 40 40 Z M 0 0 40 20 0 40 Z"
```

→ solution

5.3 Reference for Filling

LibreOffice always uses the bounding box of the shape as reference area for positioning and scaling the filling, regardless of whether the shape has even-odd filling or stacked areas or both. If the shape has structured filling like a gradient or bitmap, it becomes obvious.

Take for example the triangle, which was created in chapter 3. Here it is filled with a gradient with 10 steps. The dashed rectangle indicates the bounding box of the shape.

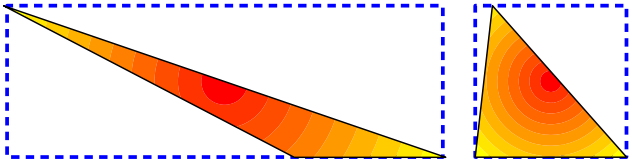


Figure 47: Bounding box for triangle

If the rectangle has the same area filling as the triangle, you can see that the fill indeed uses the bounding rectangle as reference for scaling and position.

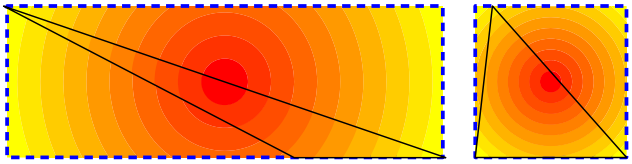


Figure 48: Rectangle equal to bounding box with same filling as shape

The shapes of this chapter look with the same gradient as this:

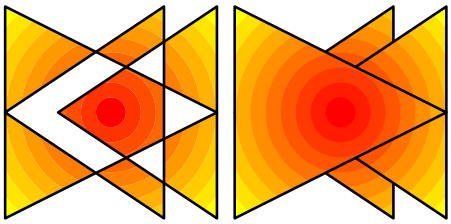


Figure 49: Gradient with even-odd filling and with stacked areas

The `<draw:enhanced-geometry>` element has an attribute `draw:concentric-gradient-fill-allowed`[17] in the ODF standard. That would allow filling to start at the shape path instead of the bounding box, resulting in rendering like what is known from OOXML, for example

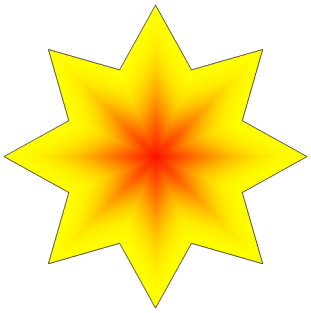


Figure 50: expected rendering if draw:concentric-gradient-fill-allowed="true"

But that attribute is not yet implemented in LibreOffice.

5.4 Lighten and Darken

In Figure 46 you saw you can get more of less light sub-paths, if the filling has transparency. However, transparency is not part of the shape geometry but a property of the graphic style.

17 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_concentric-gradient-fill-allowed

To improve compatibility with OOXML standard, LibreOffice has introduced commands J and K to lighten (blend with White) and commands H and I to darken (blend with Black) the shape color. These commands do not belong to the ODF 1.3 standard and thus must be used in it's own namespace – in this case `drawooo`. If using these commands, you must save in format “1.3 Extended”. To allow other applications to show at least the shape, you must add the ordinary `draw:enhanced-path` attribute. An example shows how to use it.

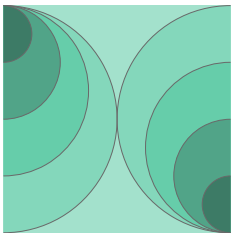


Figure 51: Light and Darken

Example:

The light part in the shape is a square, which covers the entire shape in the back. Then half circles are drawn, becoming darker and smaller from back to front. Note that the shape has no line at the edges. It is done with the S command at the square.

The solution uses `svg:viewBox="0 0 80 80"`. Half circles with same color belong to the same sub-paths set.

`draw:enhanced-path=`	Attribute in normal namespace `draw`. Its value has no extended commands.
`"M 0,0 L 80,0 80,80 0,80 Z S N`	Background rectangle. Command **S** disables lines (technical term ‘stroke’) for the rectangle.
`M 0,0 X 40,40 0,80  M 80,0 X 40,40 80,80 N`	circle radius 40 half circles on left and right side
`M 0,0 X 30,30 0,60  M 80,20 X 50,50 80,80 N`	circle radius 30
`M 0,0 X 20,20 0,40  M 80,40 X 60,60 80,80 N`	circle radius 20
`M 0,0 X 10,10 0,20  M 80,60 X 70,70 80,80 N"`	circle radius 10
drawooo`:enhanced-path=`	Attribute is in proprietary namespace **drawooo**, only used by LibreOffice.
`"M 0,0 L 80,0 80,80 0,80 Z S J N`	command **J**: lighten
`M 0,0 X 40,40 0,80 M 80,0 X 40,40 80,80 K N`	command **K**: lightenless
`M 0,0 X 30,30 0,60 M 80,20 X 50,50 80,80 N`	half circles in original color
`M 0,0 X 20,20 0,40 M 80,40 X 60,60 80,80 I N`	command **I**: darkenless
`M 0,0 X 10,10 0,20 M 80,60 X 70,70 80,80 H N"`	command **H**: darken

Without the extension, that is, as other applications would render it, the shape looks as in Figure 52.

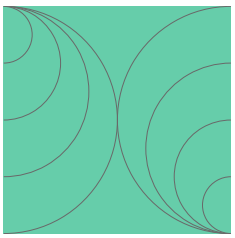


Figure 52: Without darken or lighten

5.5 Exercise Solutions

5.5.1 Exercise 5.1

The symbol contains only the black parts. The lower small circle belongs to the same sub-paths set with the same fill. It becomes transparent because of even-odd fill rule. These parts are drawn without lines. Note the command S in the markup.

Add a circle with perimeter line but no fill to separate the symbol from the background. Note the command F in the markup. Since the black parts are drawn without lines, you can use another color for the perimeter instead of black, light gray for example.

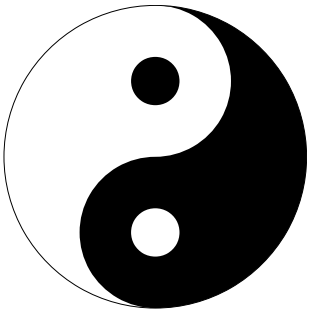


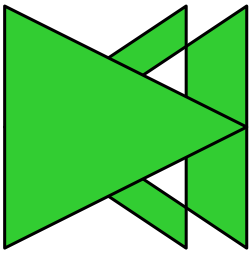
Figure 53: Yin-Yang exercise solution

Value for the `draw:enhanced-path` attribute:

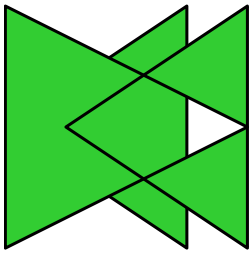
```
"M 50,0 X 100,50 50,100
25,75  50,50 75,25 50,0
M 50,67 X 58,75 50,83 42,75 50,67
M 50,17 X 58,25 50,33 42,25 50,17 S N
M 50,0 X 100,50 50,100 0,50 50,0 F N"
```

right large half circle
inner half circles left and right
small circle in Yin part
small circle in Yang part
additional circle for boundary

5.5.2 Exercise 5.2



a) *Figure 54: First set with even-odd filling*



b) *Figure 55: Last set with even-odd filling*

6 Bézier Curves

6.1 General

This chapter introduces you to commands Q and C which produce a quadratic Bézier curve and cubic Bézier curve, respectively. In literature often all points of a Bézier curve are named ‘control point’. However, we keep the wording of the Draw Guide[18] to use the term ‘control point’ only for those points which are used to determine the shape of the curve.

6.2 Quadratic Bézier Curve

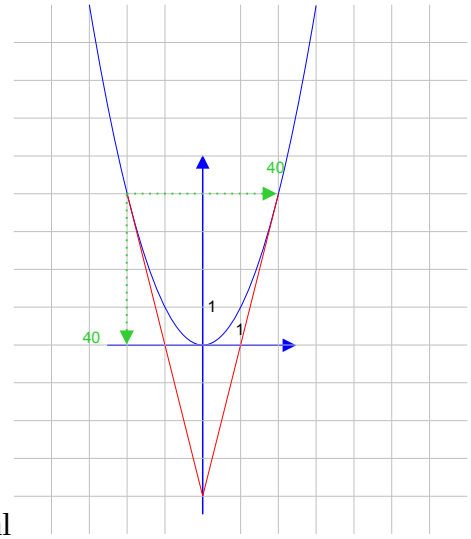
Again, partly cited from ODF standard[14]

Table 7: Description of Command Q

Command	Name	Parameters	Description
Q	quadratic-curveto	(x1 y1 x y)+	Draws a quadratic Bézier curve from the current point to (x,y) using (x1,y1) as the control point. (x,y) becomes the new current point at the end of the command.

The ‘+’ in the parameters syntax description (x1 y1 x y)+ means, that a sequence of several Q commands is possible.

A quadratic Bézier curve is always a parabola segment. Start- and end point of the curve belong to the parabola. The control point is the intersection of the tangents through start- and end point.



So the simple markup `draw:enhanced-path="M 0,0 Q 20,80 40,0"` already creates a parabola. It can be transformed to any needed focal parameter value by changing width or height of the custom shape. For more custom shapes for parabolas see the two documents mentioned in Material for schools#Parabolas[19]

Exercise 6.1

The purpose of this exercise is to brush up your knowledge about handles and equations, see chapter 3. Therefore, the needed mathematics to calculate the control point is given.

Create a custom shape with parabola segment so that (in local coordinate system of the shape) the left most point is (0|0) and the right most point is (40|40). Which part of the parabola is shown is determined by an xy-handle. With the handle in position $H\left(x_H \middle| y_H\right)$ the value for a parameter *b* is then

$$b = \frac{{x_H}^2 - 40\,y_H}{{x_H}^2 - 40\,x_H}$$

18 https://books.libreoffice.org/en/DG71/DG7111-AdvancedDrawTechniques.html#toc94
19 https://wiki.documentfoundation.org/Documentation/Material_for_schools#Parabolas

> And with this parameter *b* the control point *Z* of the quadratic Bézier curve has the coordinates $Z(20|20\,b)$. Because of potential division by zero in the calculation of *b*, the range for x_H shall be restricted to the range $x_H \in [0.001\,;39.999]$.
>
> → solution

6.3 Cubic Bézier curve

You already know cubic Bézier curves from the tools ‘Curve’ and ‘Curve, filled’ in the toolbar ‘Curves and Polygons’ and you have used the ‘Edit Points’ toolbar to edit them. The cubic Bézier curve as part of the `draw:enhanced-path` is a little bit restricted as it has no command for ‘Symmetric Transition’.

Again, partly cited from ODF standard[14]

Table 8: Description of command C

Command	Name	Parameters	Description
C	curveto	(x1 y1 x2 y2 x y) +	Draws a cubic Bézier curve from the current point to (x,y) using (x1,y1) as the control point at the beginning of the curve and (x2,y2) as the control point at the end of the curve.

The notation in the tutorial follows the book “A Primer on Bézier Curves”[20], which is a good start into learning more about Bézier curves.

A cubic Bézier curve is determined by a parametric function

$$B(t)=\begin{pmatrix}B_x(t)\\B_y(t)\end{pmatrix}=(1-t)^3P_0+3(1-t)^2\cdot t\cdot P_1+3(1-t)\cdot t^2\cdot P_2+t^3P_3,\ \ t\in[0\,;1]$$

with start anchor point P_0, control point P_1 (related to start point), control point P_2 (related to end point) and end anchor point P_3[21].

If you have got points $P_0(x_0|y_0)$, $P_1(x_1|y_1)$, $P_2(x_2|y_2)$, $P_3(x_3|y_3)$ then the markup is

`draw:enhanced-path="M x0,y0 C x1,y1 x2,y2 x3,y3"`

Here again a sequence of commands is possible and in such sequence the character C is only needed at the beginning.

So if you have got points $P_0(x_0|y_0)$, $P_1(x_1|y_1)$, $P_2(x_2|y_2)$, $P_3(x_3|y_3)$ for a Bézier curve and $P_3(x_3|y_3)$, $P_4(x_4|y_4)$, $P_5(x_5|y_5)$, $P_6(x_6|y_6)$ for a second Bézier curve continuing the first one, then the markup is

`draw:enhanced-path="M x0,y0 C x1,y1 x2,y2 x3,y3 x4,y4 x5,y5 x6,y6"`

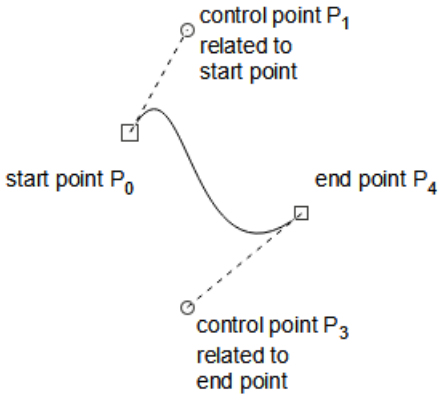


Figure 56: Points of a Bézier curve

20 https://pomax.github.io/bezierinfo/ [called 2021-09-13]

21 Most articles name all four points ‘control’ points, but I follow the naming of the LibreOffice ‘Draw Guide’ which emphasizes the type of the points.

That results in long sequences of point coordinates, and you need to be very careful to list them correctly.

6.3.1 Example ‘Easter egg’

As first example we make a shape for an ‘Easter egg’. The screenshot Figure 57 shows the shape in ‘Edit points’ mode. It was created as path object using the tool “Curve, filled”.

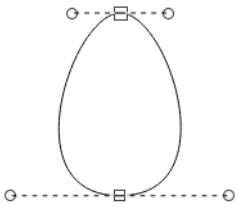


Figure 57: Easter egg created as path object

If your custom shape is static, that is, it has no handles, then you can write the values directly into the `draw:enhanced-path` attribute. How do you calculate the values?

A first idea would be to create the desired object as path object in the UI and then use its points from the source file. But that does not work. A path object uses the attribute `svg:d`. That follows in its syntax the attribute `d` of the `path` element of SVG[22]. Unfortunately LibreOffice writes the control points as relative values and the language for the `draw:enhanced-path` attribute has no commands for relative values.

Hence you need to define the points manually. For the “Easter egg” I have measured the distances in the UI.

	A	B	C	D	E	F	G
1	This spreadsheet belongs to section “Cubic Bézier Curve”						
2							
3	All values are designed for below viewBox						
4	svg:viewBox="0 0 2000 3000"						
5							
6	**Parameters of Easter egg**		**comment**				
7	1000	0	anchor point A, top of egg				
8	790		horizontal distance of top control points from anchor point A				
9	1000	3000	anchor point B , bottom of egg				
10	1820		horizontal distance of bottom control points from anchor point B				
11							
12	**x**	**y**					
13	draw:enhanced-path="						
14	M						
15	1000	0	P0 of 1. Bézier curve = point A				
16	C						
17	210	0	P1 of 1. Bézier curve				
18	-820	3000	P2 of 1. Bézier curve				
19	1000	3000	P3 of 1. Bézier curve = P0 of 2. Bézier curve = B				
20	2820	3000	P1 of 2. Bézier curve				
21	1790	0	P2 of 2. Bézier curve				
22	1000	0	P3 of 2. Bézier curve = point A, explicit close not needed for filling				
23	"						

Figure 58: Spreadsheet to calculate coordinates for the enhanced-path and document it

I suggest using a spreadsheet as documentation of your work. The screenshot Figure 58 shows such spreadsheet. If you organize the spreadsheet in a smart way, you can copy and paste the values into

22 https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/d [called 2021-09-12]

the editor you use to create the custom shape. The marked area was copied directly to Notepad++ to create the custom shape below, for example.

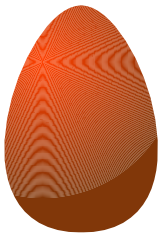


Figure 59: Finished Easter egg as custom shape

The markup in the custom shape is then

```
<draw:enhanced-geometry
    svg:viewBox="0 0 2000 3000" draw:type="non-primitive"
    draw:enhanced-path=
        "M 1000 0 C 210 0 -820 3000 1000 3000 2820 3000 1790 0 1000 0"/>
```

6.3.2 Sine Curve as Custom Shape

Sine curves can be very well approximated by cubic Bézier curves. This section describes how to do it.

If you approximate a function graph on interval $[a_0 ; a_3]$ these three relationships are important:

1. The anchor points of the Bézier curve coincide with the points of the function graph at the edge of the interval.

2. The straight line through control point and anchor point is a tangent to the function graph.

3. In the sequence of Bézier curves, one curve segment will connect to the next. The transition should be smooth. Therefore, the Bézier curves here are so created, that not only the tangent but also its curvature in P_0 and P_3 is the same as for the function. In other contexts, instead of curvature other requirements might be useful, average or maximal deviation, for example.

It is helpful to use a computer algebra system for the calculation. Section 11.1 contains more about the mathematics.

Curve A: For to get the red curve in Figure 60 the points are
$P_0(0|0)$, $P_1\left(\frac{3}{2}\pi-\frac{3}{8}\pi^2-\frac{1}{2}\middle|\frac{3}{2}\pi-\frac{3}{8}\pi^2-\frac{1}{2}\right)$, $P_2(1|1)$ and $P_3\left(\frac{\pi}{2}\middle|1\right)$.
The maximal deviation of Bézier curve to sine curve is less than 0.0006.

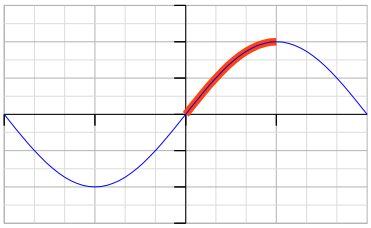


Figure 60: Sine curve on interval [0;π/2]

Curve B: For to get the red curve in Figure 61 the points are
$P_0\left(-\frac{\pi}{2}\middle|-1\right)$, $P_1\left(-\frac{\pi}{2}+\frac{2}{\sqrt{3}}\middle|-1\right)$, $P_2\left(\frac{\pi}{2}-\frac{2}{\sqrt{3}}\middle|1\right)$ and $P_3\left(\frac{\pi}{2}\middle|1\right)$.
The maximal deviation of Bézier curve to sine curve is less than 0.0024.

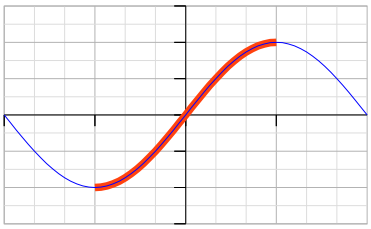


Figure 61: Sine curve on interval [−π/2;π/2]

Now we discuss the steps needed to create the shape for curve A.

The custom shape uses a local coordinate system with y-axis pointing down and origin in the left top corner of the shape.

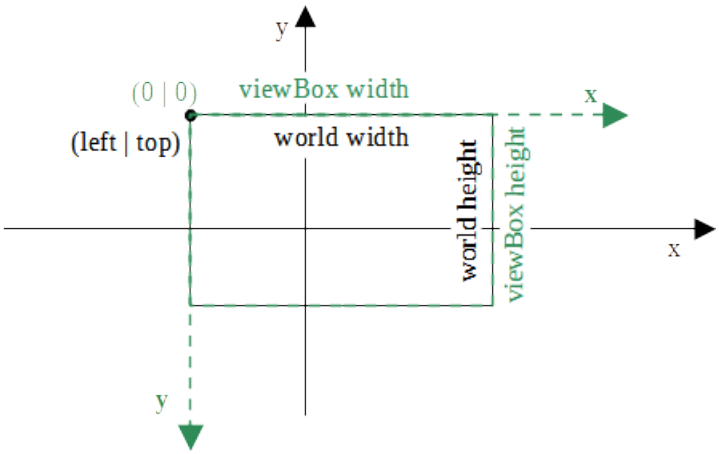


Figure 62: Transformation of 'world' coordinates to shape local coordinates

You need to transform the mathematical 'world' coordinates to coordinates in the local coordinate system. That is done with a homogeneous matrix ***M***, so that
"point in shape local coordinates" = ***M*** · "point in world coordinates"

$$\boldsymbol{M} = \begin{pmatrix} \frac{\text{viewBox width}}{\text{world width}} & 0 & -\frac{\text{viewBox width}}{\text{world width}} \cdot \text{left} \\ 0 & -\frac{\text{viewBox height}}{\text{world height}} & \frac{\text{viewBox height}}{\text{world height}} \cdot \text{top} \\ 0 & 0 & 1 \end{pmatrix}$$

The homogeneous matrix is used for to include translations into the matrix. If you apply the matrix to point coordinates you need to add value 1 as third point coordinate.

Curve A uses the "world" rectangle with (left | top) = (0 | 1) and width × height = π/2 × 1. The shape will use the viewBox value `"0 0 1500 1000"`. The needed transformation matrix is then

$$\boldsymbol{M} = \begin{pmatrix} \frac{3000}{\pi} & 0 & 0 \\ 0 & -1000 & 1000 \\ 0 & 0 & 1 \end{pmatrix}$$

Next you apply the matrix to the points coordinates and get the following results.

Point	“world” coordinates P_W	coordinates in shape $P_S = \boldsymbol{M} \cdot P_w$
P_0	(0|0)	(0|1000)
P_1	$\left(\frac{3}{2}\pi - \frac{3}{8}\pi^2 - \frac{1}{2} \mid \frac{3}{2}\pi - \frac{3}{8}\pi^2 - \frac{1}{2}\right)$	$\left(\frac{3000}{\pi} \cdot (\frac{3}{2}\pi - \frac{3}{8}\pi^2 - \frac{1}{2}) \mid -1000 \cdot (\frac{3}{2}\pi - \frac{3}{8}\pi^2 - \frac{1}{2}) + 1000\right)$
P_2	(1|1)	$(\frac{3000}{\pi}\|0)$
P_3	(π/2|1)	(1500|0)

These coordinates can result in the following markup. Other solutions are possible.

```
<draw:enhanced-geometry
    svg:viewBox="0 0 1500 1000"
    draw:type="non-primitive"
    draw:enhanced-path="M 0 1000 C ?f2 ?f3 ?f0 0 1500 0" >
  <draw:equation draw:name="f0" draw:formula="3000/pi" />
  <draw:equation draw:name="f1" draw:formula="3/2*pi-3/8*pi*pi-1/2" />
  <draw:equation draw:name="f2" draw:formula="?f0*?f1" />
  <draw:equation draw:name="f3" draw:formula="1000*(-?f1+1)" />
</draw:enhanced-geometry>
```

You could also calculate the values outside in a spreadsheet and use them directly (copy&paste!) in the `draw:enhance-path` attribute. The syntax allows floating point values with decimal dot. In that case let the spreadsheet also calculate the transformations.

Exercise 6.2

Create a shape as in Figure 63. For that you put a Bézier curve with the points given in the text to Figure 61 and a mirrored and shifted version of it together in one `draw:enhance-path`. Extending the curve that way makes it possible for the user to create a long sine curve by using the tool ‘Edit > Duplicate’.

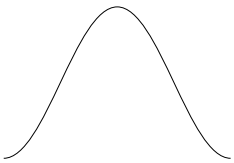


Figure 63: Sine curve segment

Use `"0 0 6000 2000"` as viewBox value and use formulas.

→ solution

6.4 Exercise Solutions

6.4.1 Exercise 6.1

```
<draw:enhanced-geometry
draw:type="non-primitive"

svg:viewBox="0 0 40 40"
```

Do not forget to change it, if you start with a predefined shape

Other values are possible, but using the shape is easier if the shown lines end at the shape snap rectangle.

`draw:modifiers="10 20"`	Without a draw:modifiers attribute a handle would not be movable. The default values are arbitrary, but should be valid in regard to the range given in the draw:handle element
`draw:enhanced-path="M 0 0 Q 20 ?f2 40 40">`	The y-coordinate of the control point depends on the way you have built the expressions. In case you have only one expression used, it would be `"M 0 0 Q 20 ?f0 40 40"`

`<draw:equation`
`  draw:name="f0"`
`  draw:formula="$0*$0" />`
`<draw:equation`
`  draw:name="f1"`
`  draw:formula="(?f0-40*$1)/(?f0-40*$0)" />`
`<draw:equation`
`  draw:name="f2"`
`  draw:formula="20*?f1" />`
	You could have written the complete formula into one equation. ODF allows such long expressions.

`<draw:handle draw:handle-position="$0 $1"`	This handle should be freely movable. Therefore, both coordinates need a reference to a modifier.
`draw:handle-range-x-minimum="0.001"` `draw:handle-range-x-maximum="39.999" />`	Direct translation of the given range.
`</draw:enhanced-geometry>`	The element has child elements and therefor needs a structure with start and end tag.

6.4.2 Exercise 6.2

The sinus curve segment has the “world” rectangle with (left | top) = (−π/2 | 1) and width × height = 2π × 2. Together with the given viewBox value `"0 0 6000 2000"` you get the transformation matrix

$$\boldsymbol{M} = \begin{pmatrix} \frac{3000}{\pi} & 0 & 1500 \\ 0 & -1000 & 1000 \\ 0 & 0 & 1 \end{pmatrix}$$

The point coordinates are given in following table:

Point	"world" coordinates P_W	coordinates in shape $P_S = \boldsymbol{M} \cdot P_w$
P_0	$(-\pi/2 \mid -1)$	$(0 \mid 2000)$
P_1	$\left(-\frac{\pi}{2}+\frac{2}{\sqrt{3}} \middle\vert -1\right)$	$\left(\frac{3000}{\pi}\cdot\frac{2}{\sqrt{3}} \middle\vert 2000\right)$
P_2	$\left(\frac{\pi}{2}-\frac{2}{\sqrt{3}} \middle\vert 1\right)$	$\left(3000-\frac{3000}{\pi}\cdot\frac{2}{\sqrt{3}} \middle\vert 0\right)$
$P_3 = Q_0$	$(\pi/2 \mid 1)$	$(3000 \mid 0)$
Q_1	$\left(\frac{\pi}{2}+\frac{2}{\sqrt{3}} \middle\vert 1\right)$	$\left(3000+\frac{3000}{\pi}\cdot\frac{2}{\sqrt{3}} \middle\vert 0\right)$
Q_2	$\left(\frac{3\pi}{2}-\frac{2}{\sqrt{3}} \middle\vert -1\right)$	$\left(6000-\frac{3000}{\pi}\cdot\frac{2}{\sqrt{3}} \middle\vert 2000\right)$
Q_3	$(3\pi/2 \mid -1)$	$(6000 \mid 2000)$

The expression $\frac{3000}{\pi}\cdot\frac{2}{\sqrt{(3)}}$ (or simplified $\frac{2000\cdot\sqrt{3}}{\pi}$) appears several times in the coordinates. So you should put it into a formula for its own.

```
<draw:enhanced-geometry
    svg:viewBox="0 0 6000 2000"
    draw:type="non-primitive"
    draw:enhanced-path="M 0 2000 C ?f0 2000 ?f1 0 3000 0
                        ?f2 0 ?f3 2000 6000 2000" >
  <draw:equation draw:name="f0" draw:formula="3000/pi*2/sqrt(3)" />
  <draw:equation draw:name="f1" draw:formula="3000-?f0" />
  <draw:equation draw:name="f2" draw:formula="3000+?f0" />
  <draw:equation draw:name="f3" draw:formula="6000-?f0" />
</draw:enhanced-geometry>
```

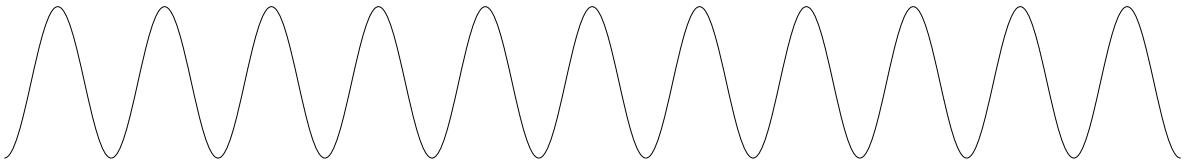


Figure 64: Exercise solution long sine curve by duplicating the shape

7 Shapes with Fixed Distances

7.1 Fixed Height for Sub-path

The first example is inspired by a question on Ask[23]:

Two rectangles are combined in a group. One rectangle holds a heading, the other takes the text body. Requirement: The heading does not change its height, when the group size is changed.

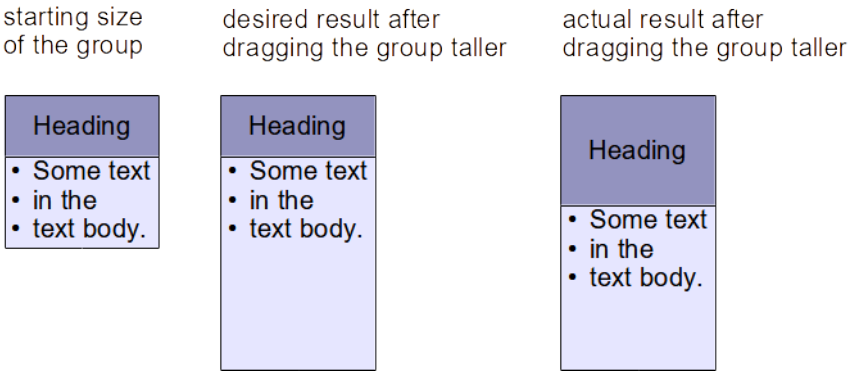


Figure 65: Desired and actual result for making a group higher

The property “Protect Size” does only disable size related settings in the user interface of the shape itself, but it does not protect a shape from being resized, when the containing group is resized.

First approach: We cannot have fixed total height, but we can try to draw a rectangle inside the shape, which has a fixed height and the rest of the shape is transparent.

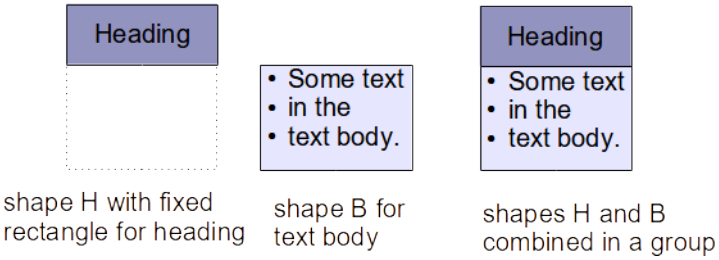


Figure 66: First idea: rectangle with fixed height inside a larger shape

The commands and formulas discussed so far all refer to the viewBox and the values of the viewBox do not change when the shape is resized. Something else is needed, that refers to the actual size of the shape. Such is given in the identifiers `logwidth` and `logheight`.

Quoted from “Table 11 - Enhanced geometry equation identifiers“[24] in the ODF 1.3 standard:

Table 9: Description of identifiers logheight and logwidth

logheight	The height in 1/100th mm as specified by the svg:height 19.543 attribute is used.
logwidth	The width in 1/100th mm as specified by the svg:width 19.575 attribute is used.

We assume shape H is currently 4 cm high and the rectangle for the heading has a fixed height of 1 cm. We assume further, that shape H has the viewBox value `"0 0 2500 6000"`. (The actual

23 https://ask.libreoffice.org/t/protect-size-of-individual-items-in-a-group/67682/4 [called 2021-09-17]

24 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_formula, [called 2021-09-17]

values of attribute viewBox are not essential to the problem, but concrete values make the solutions more understandable.) The desired height of 1 cm is 1/4 of the total height of the shape. So in local coordinate system we need to use 1/4 of the viewBox height, that is 1500.

If the shape is then dragged to a height of 8 cm, the desired 1 cm is 1/8 of the total height of the shape. So, in local coordinate system we need to use 1/8 of the viewBox height, that is 750.

In general we get

$$\text{needed value in local coordinate system} = \frac{\text{desired fixed height in world coordinates}}{\text{shape height in world coordinates}} \cdot \text{viewBox height}$$

This is where the identifier `logheight` comes into play. We switch to values in 1/100th mm and get

$$\text{needed value in local coordinate system} = \frac{\text{desired fixed height in 1/100th mm}}{\text{logheight (which is in 1/100th mm)}} \cdot \text{viewBox height}$$

You can read the rule also as: Scale the desired height with the ratio of viewBox height to logheight.

$$\text{needed value in local coordinate system} = \text{desired fixed height in 1/100th mm} \cdot \frac{\text{viewBox height}}{\text{logheight (which is in 1/100th mm)}}$$

The shape H could have this markup:

```
<draw:enhanced-geometry
    svg:viewBox="0 0 2500 6000"
    draw:text-areas="0 0 2500 ?f0"
    draw:type="non-primitive"
    draw:enhanced-path="M 0 0 L 0 ?f0 2500 ?f0 2500 0 Z">
  <draw:equation draw:name="f0" draw:formula="1000/logheight * height"/>
</draw:enhanced-geometry>
```

Note that the text area has the same rectangle as given by the path. Only the attribute text-area has parameters “left, top, right, bottom”, whereas attribute viewBox has the parameters “left, top, width, height”. You need to use the same rectangle to place the text really into the fixed rectangle.

If you look at the drag handles, you see that the actual shape height is larger as the colored rectangle for the heading.

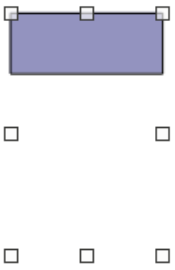


Figure 67: Drag handles of shape H

Now combine shape H in 2.5 width with a shape B for the body text into a group. Use for shape B a rectangle with 2.5 cm width and 3 cm height so that both shapes together result in 2.5 cm width and 4 cm height.

Try out what happens, if you change the height of the group to 6cm, for example.

Shape H for the heading behaves as expected and keeps 1 cm height of its colored rectangle. But shape B for the body text behaves the same as before, so that there is now a gap between the shapes.

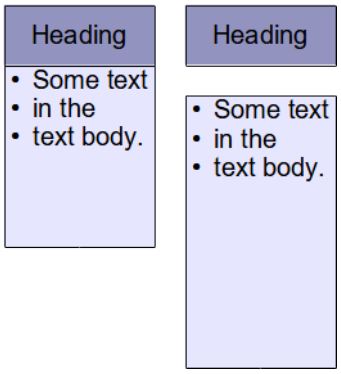


Figure 68: Resized group with fixed heading rectangle

We need to design shape B so the area above the text rectangle is always 1 cm high. That is only possible, if this area belongs to the shape.

Conclusion: Shape B needs a similar solution as shape H. It needs the same total height as shape H. However, now the upper part with the fixed height is transparent, and the colored rectangle and the text area are placed in the lower part.

> **Exercise 7.1**
>
> Create the markup for such shape B.
>
> → solution

In general, to calculate a value *Local* in the local coordinate system from a value *World* in the world coordinate system use the formula $Local = \frac{World \cdot \text{width}}{\text{logwidth}}$ for horizontal direction or $Local = \frac{World \cdot \text{height}}{\text{logheight}}$ for vertical direction.

LibreOffice uses internally the unit “Twip” in Writer and “1/100th mm” in all other modules.
1 Twip = 127/7200 mm. Currently [as of 2021-09-18] LibreOffice has the bug[25], that the identifiers `logheight` and `logwidth` give the value in regard to Twip in Writer.

If your formula has a constant number as length which is used in comparison to identifier `logwidth` or `logheight`, you need to use the length in Twip too. In such case you need a special version of your shape for use in Writer, and you cannot exchange such shapes between Writer and the other modules. That means, to get a rectangle with 1 cm height in Writer in the above example, you have to use 72000/127 for the desired fixed height in the formula instead of 1000.

7.2 Ratio of Shape Width to Height

In this section we will create a shape like the shape “Card” of the category “Flowchart”, but with a handle to determine the size of the cut corner.

As a quick shot we get the following markup

```
<draw:enhanced-geometry svg:viewBox="0 0 600 600"
      draw:type="non-primitive" draw:modifiers="150"
      draw:enhanced-path="M $0 0 L 0 $0 0 600 600 600 600 0 Z">
```

25 The fact, that identifiers `logwidth` and `logheight` are not conform to ODF 1.3 in Writer is tracked as bug in https://bugs.documentfoundation.org/show_bug.cgi?id=144591

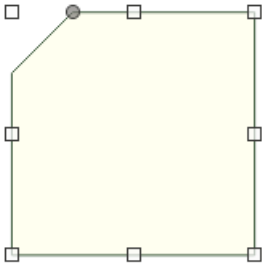


Figure 69: Rectangle with cut corner

```
  <draw:handle draw:handle-position="$0 top"
               draw:handle-range-x-minimum="0"
               draw:handle-range-x-maximum="600"/>
</draw:enhanced-geometry>
```

However, when you change the width or height of the shape the cut has no longer 45° (yellow). But we want a solution where the cut has always 45° regardless shape width or height (green).

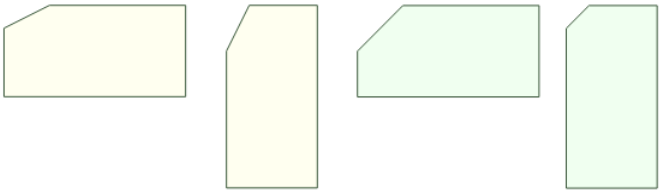


Figure 70: Angle changes if width or height changes

Let us have a second view on identifiers `logwidth` and `logheight`.

Mapping between world and local coordinate systems, can be illustrated with a grid on the shape.

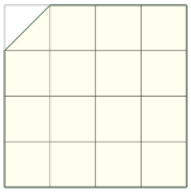


Figure 71: Grid on square shape with same unit on both axes

If the width or height is changed, the units on x-axis and y-axis become different. To get the same angle, one of x- or y-coordinate has to be adapted.

Figure 72: Different units on axes

If the shape is now half of the height of the square size, then $logwidth/logheight = 2$ and the y-coordinate must be doubled. If the shape has a height three times the height of the square size, then $logwidth/logheight = 1/3$ and you need 1/3 of the y-coordinate. Only the ratio of shape width and height is relevant.

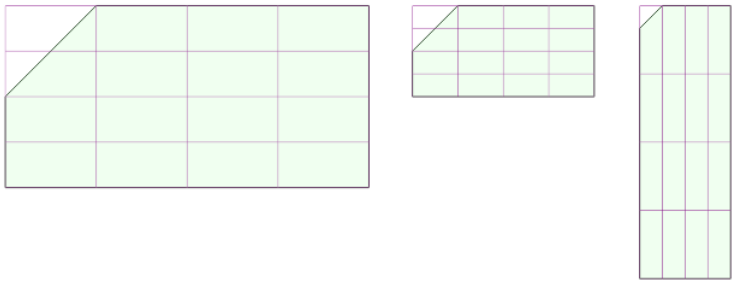


Figure 73: Needed adaption for changed width or height

The scaling factor for the y-coordinate is `logwidth/height`.

And the solution is now:

```
<draw:enhanced-geometry svg:viewBox="0 0 600 600"
      draw:type="non-primitive" draw:modifiers="150"
      draw:enhanced-path="M $0 0 L 0 ?f0 0 600 600 600 600 0 Z">
  <draw:equation draw:name="f0" draw:formula="$0*logwidth/logheight" />
  <draw:handle draw:handle-position="$0 top"
               draw:handle-range-x-minimum="0"
               draw:handle-range-x-maximum="600"/>
</draw:enhanced-geometry>
```

This solution uses implicitly the fact, that the viewBox is a square, that is, `width=height`. If that is not the case, you need to consider ratio in the viewBox size as well. For example

```
<draw:enhanced-geometry svg:viewBox="0 0 600 300"
      draw:type="non-primitive" draw:modifiers="150"
      draw:enhanced-path="M $0 0 L 0 ?f0 0 300 600 300 600 0 Z">
  <draw:equation draw:name="f0"
                 draw:formula="$0*height/width*logwidth/logheight" />
  <draw:handle draw:handle-position="$0 top"
               draw:handle-range-x-minimum="0"
               draw:handle-range-x-maximum="600"/>
</draw:enhanced-geometry>
```

Exercise 7.2

If you drag the handle in the shape having $logwidth/logheight = 2$, then you will get for example

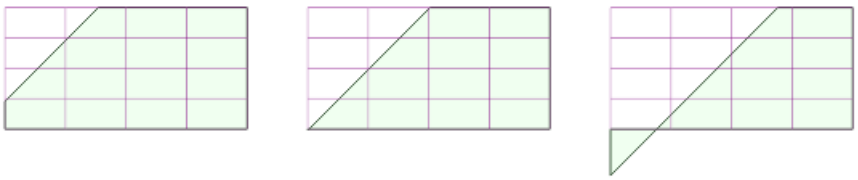


Figure 74: Need for restricting the range for handle movement

Obviously, we need to restrict the range for handle movement to get “nice” results.

Set the value of attribute `draw:handle-range-x-maximum` so, that the cut it not lower then half of the height and not more right than half of the width of the shape. Use the viewBox value `"0 0 600 600"`.

You can use a formula reference instead of a constant value or identifier in the value of attribute `draw:handle-range-x-maximum`.

Test your solution with very wide and with very tall shapes.

→ solution

Remark: The OOXML standard has a similar shape “snap1Rect”.

7.3 Equilateral Triangle

We want a shape with these properties:

- The shape shows an equilateral triangle.
- The width of the triangle is equal to the width of the shape.
- The base of the triangle aligns with the bottom edge of the shape.
- The height of the triangle is computed so the triangle is equilateral, regardless of the height of the shape.

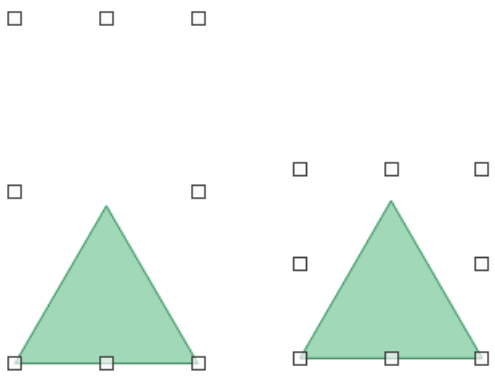


Figure 75: Equilateral triangle inside the snap rectangle of the shape

Figure 76 shows the triangle-shape in a world coordinate system with y-axis pointing down. Left, top corner of the shape is aligned with the origin. The local coordinate system looks similar, only it has `height` instead of `logheight` and `width` instead of `logwidth`.

The mathematical formula for the length *h* is $h = g \cdot \frac{1}{2} \cdot \sqrt{3} = g \cdot \sqrt{\frac{3}{4}}$.

$h = g \cdot \sin(\frac{1}{3}\pi)$ can be used as well.

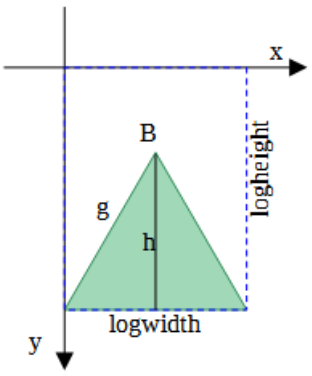


Figure 76: ‘world’ coordinate system for equilateral triangle

A solution is possible with the ideas from section 7.1 as well as with those from section 7.2.

The following solution uses ideas from section 7.1. However now the length, which should be independent from the shape height, is not given anymore by a constant number in the world coordinate system but is calculated from the shape width, and the fixed length is measured from the bottom edge of the shape.

We can use these equations

`f0=logwidth*sqrt(0.75)`	Length *h* in world coordinate system
`f1=?f0*height/logheight`	Value that represents length *h* in local coordinate system.

`f2=height-?f1`	The y-coordinate of tip of the triangle in the local coordinate system.

And this solution uses ideas from section 7.2. It calculates first a distance in local coordinate system and then corrects it for cases with ratio ≠ 1.

`f0=width*sqrt(0.75)`	Distance of triangle tip to triangle base in local coordinate system, if viewBox would be square
`f1=?f0*height/width`	Correction for the case viewBox is not square
`f2=?f1*logwidth/logheight`	Correction for the case the shape is not square
`f3=height-?f2`	The y-coordinate of tip of the triangle in the local coordinate system.

Put for both solutions the expressions into a single formula and simplify it, then you see that the result is the same.

With `svg:viewBox="0 0 6000 4000"` for example, the path would be in first solution
`draw:enhanced-path="M 0 4000 L 3000 ?f2 6000 4000"`

and in second solution
`draw:enhanced-path="M 0 4000 L 3000 ?f3 6000 4000"`

Exercise 7.3

Create two shapes with these properties:

One shape shows a square, the other a cross as diagonals of the square.

The base length of the square is the minimum of width and height of the shape. So the square is never larger than the snap rectangle of the shape.

The square is centered in the shape, that is the place besides the square is the same on opposite sides.

→ solution

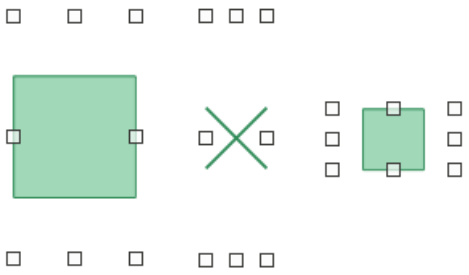


Figure 77: Square centered in snap rectangle of the shape

7.4 Height Independent Length with Modifier

We pick the example “right triangle by Thales’ circle with xy-handle” from section “3.5 Functions”.

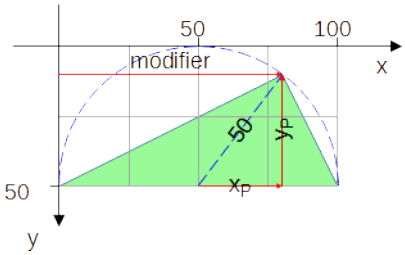


Figure 78: Thales' half circle

The solution there has been

```
<draw:enhanced-geometry svg:viewBox="0 0 100 50" draw:type="non-primitive"
            draw:modifiers="70"
            draw:enhanced-path="M 0 50 L $0 ?f2 100 50 Z N">
    <draw:equation draw:name="f0" draw:formula="$0-50"/>
    <draw:equation draw:name="f1" draw:formula="sqrt(50*50-?f0*?f0)"/>
    <draw:equation draw:name="f2" draw:formula="50-?f1"/>
    <draw:handle draw:handle-position="$0 ?f2"
                 draw:handle-range-x-minimum="left"
                 draw:handle-range-x-maximum="right"/>
</draw:enhanced-geometry>
```

But that creates only a right triangle if the shape is twice as wide as high. To get a right triangle in all cases, the height of the triangle needs to be independent from the shape height.

In addition, as the triangle no longer stretches to the height of the shape, we need to decide where to position the triangle. Here the triangle is aligned with the bottom edge of the shape.

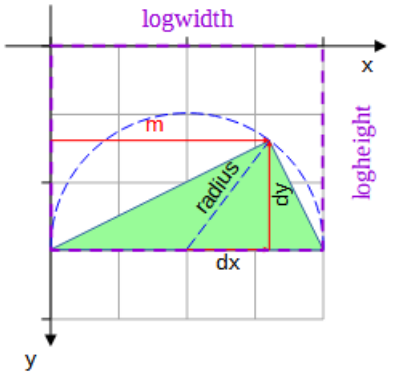


Figure 79: Right triangle aligned with bottom of the shape

We try a solution with ideas from section 7.1. The difference is that here the height needs to be calculated not from a value in the world coordinate system but from a modifier value, which is as such a value in the local coordinate system. The solution is to first calculate backwards a length value in the world coordinate system, which is represented by the modifier.

In detail we make these steps

1. Calculate the value *m* in the world coordinate system, which is represented by the modifier value $0.

   You know from section 7.1, how to calculate a value *Local* in the local coordinate system from a value *World* in the world coordinate system. Use the formula $Local = \frac{World \cdot \text{width}}{\text{logwidth}}$ for horizontal direction or $Local = \frac{World \cdot \text{height}}{\text{logheight}}$ for vertical direction.

   Now we know value *Local* and look for value *World*. We rearrange the formula and get $World = \frac{Local \cdot \text{logwidth}}{\text{width}}$ or $World = \frac{Local \cdot \text{logheight}}{\text{height}}$ respectively.

   Because the modifier value $0 is related to horizontal handle movement we use $m = \frac{\$0 \cdot \text{logwidth}}{\text{width}}$.

2. Calculate the height *dy* of the triangle in world coordinate system.

   $radius = \frac{logwidth}{2}$, $dx = m - radius$ and $dy = \sqrt{radius^2 - dx^2}$

3. Calculate the height *h* in the local coordinate system that represents the value *dy*. This is a length in vertical direction

   $$h = \frac{dy \cdot \text{height}}{\text{logheight}}$$

4. Calculate the y-coordinate of the top of the triangle in the local coordinate system. The x-coordinate is given by the modifier.

   $$\text{y-coordinate of top of the triangle} = \text{height} - h$$

This solution transferred to an `<draw:enhanced-geometry>` element is then

```
<draw:enhanced-geometry
        svg:viewBox="0 0 100 50"
        draw:type="non-primitive"
        draw:modifiers="70"
        draw:enhanced-path="M 0 50 L $0 ?f5 100 50 Z">
  <draw:equation draw:name="f0" draw:formula="$0*logwidth/width"/>
  <draw:equation draw:name="f1" draw:formula="logwidth/2"/>
  <draw:equation draw:name="f2" draw:formula="?f0-?f1"/>
  <draw:equation draw:name="f3" draw:formula="sqrt(?f1*?f1-?f2*?f2)"/>
  <draw:equation draw:name="f4" draw:formula="?f3*height/logheight"/>
  <draw:equation draw:name="f5" draw:formula="height-?f4"/>
  <draw:handle draw:handle-position="$0 ?f5"
               draw:handle-range-x-minimum="left"
               draw:handle-range-x-maximum="right"/>
</draw:enhanced-geometry>
```

If you use other values than 100 and 50 for width and height in the attribute `svg:viewBox`, you need to adapt the values in `draw:enhanced-path` according to the meaning – not literal – `"M 0 bottom L $0 ?f5 right bottom Z"`. Remember you cannot use `bottom` and `right` directly in a value of attribute `draw:enhanced-path`, because these identifiers are not allowed.

> **Exercise 7.4**
>
> Create a solution based on the ideas of section 7.2.
>
> → solution

7.5 Further Path Attributes

The ODF standard provides the attributes `draw:path-stretchpoint-x`[26] and `draw:path-stretchpoint-y`[27]. It seems to me, that the implementation in LibreOffice does not follow the description in the ODF standard. Therefore, I do not discuss the attributes here.

26 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_path-stretchpoint-x

27 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_path-stretchpoint-y

7.6 Exercise Solutions

7.6.1 Exercise 7.1

For shape B you need

```
<draw:enhanced-geometry
    svg:viewBox="0 0 2500 6000"
    draw:text-areas="0 ?f0 2500 6000"
    draw:type="non-primitive"
    draw:enhanced-path="M 0 ?f0 L 0 6000 2500 6000 2500 ?f0 Z">
  <draw:equation draw:name="f0" draw:formula="1000/logheight * height"/>
</draw:enhanced-geometry>
```

Note the formula is not changed in comparison to the solution for shape H. Only path and text area are set to the area below point (0|?f0), whereas shape H has them above that point.

7.6.2 Exercise 7.2

The solution in section 7.2 was

```
<draw:enhanced-geometry svg:viewBox="0 0 600 600"
      draw:type="non-primitive" draw:modifiers="150"
      draw:enhanced-path="M $0 0 L 0 ?f0 0 600 600 600 600 0 Z">
  <draw:equation draw:name="f0" draw:formula="$0*logwidth/logheight" />
  <draw:handle draw:handle-position="$0 top"
               draw:handle-range-x-minimum="0"
               draw:handle-range-x-maximum="600"/>
</draw:enhanced-geometry>
```

The modifier value `$0` marks the horizontal position of the cut and formula value `?f0` the vertical position.

The condition `?f0 < height/2` needs to be converted to use `$0`. Some formula rearrangements:
`?f0<height/2 ^ ?f0=$0*logwidth/logheight`

⇔ `$0*logwidth/logheight<height/2`

⇔ `$0<height/2*logheight/logwidth`

The condition `$0<width/2` can be used directly. To fulfill both conditions use the minimum of the values.

```
<draw:enhanced-geometry svg:viewBox="0 0 600 600"
      draw:type="non-primitive" draw:modifiers="150"
      draw:enhanced-path="M $0 0 L 0 ?f0 0 600 600 600 600 0 Z">
  <draw:equation draw:name="f0" draw:formula="$0*logwidth/logheight" />
  <draw:equation draw:name="f1" draw:formula="height/2*logheight/logwidth" />
  <draw:equation draw:name="f2" draw:formula="min(width/2,?f1)" />
  <draw:handle draw:handle-position="$0 top"
               draw:handle-range-x-minimum="0"
               draw:handle-range-x-maximum="?f2"/>
</draw:enhanced-geometry>
```

7.6.3 Exercise 7.3

Figure 80 shows the square-shape in a world coordinate system with y-axis pointing down. Left, top corner of the shape is aligned with the origin. The local coordinate system is similar, only it has `height` instead of `logheight` and `width` instead of `logwidth`.

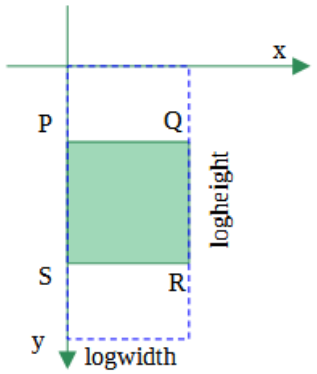


Figure 80: Square in "world" coordinate system

The size of the shape (not of the square) in world coordinate system is logwidth×logheight, the size of the shape in local coordinate system is width×height.

There is no unique solution. Two solutions as example:

Example A

The length of the edge of the square as minimum of `logwidth` and `logheight` can be calculated with

```
f0 = min(logwidth,logheight)
```

Depending on whether `logwidth` > `logheight` (⇔`logwidth-logheight` is positive) or not, the formulas use the representation of the world square width or height. The way is the same as in the previous examples.

```
f1 = if(logwidth-logheight, ?f0/logwidth*width, width)
f2 = if(logwidth-logheight, height, ?f0/logheight*height)
```

To center the square, we calculate the space outside the square and halve it. That gives the coordinates of left, top point P in local coordinate system. Depending on whether `logwidth` > `logheight` or not, this shift has to be done either with width or height.

```
f3 = if(logwidth-logheight,(width-?f1)/2,0)
f4 = if(logwidth-logheight,0,(height-?f2)/2)
```

To get coordinates of point R in local coordinate system we add width and height of the square to the coordinates of point P.

```
f5 = ?f3+?f1
f6 = ?f4+?f2
```

The coordinates of points Q and S are taken from points P and R and need no formulas:
Qx = Rx, Qy = Py, Sx = Px, Sy = Ry

Example B

Here some steps are combined, and case distinctions are omitted.

If `logwidth` > `logheight`, then `logheight` is the minimum and length of edges of the square. If `logwidth` ≤ `logheight`, then `logwidth` is length of edges. We omit above formula `f0` and use these values directly.

```
f0=if(logwidth-logheight,logheight/logwidth*width,width)
```

widths of the square in locale coordinate system

`f1=if(logwidth-logheight,height,logwidth/logheight*height`	height of the square in locale coordinate system
`f2=(width-?f0)/2`	x-coordinate of point P. `width-?0` gives 0 in case `logwidth` ≤ `logheight`
`f3=(height-?f1)/2`	y-coordinate of point P `height-?f1` gives 0 in case `logwidth` > `logheight`
`f4=?f0+?f2`	x-coordinate of point R
`f5=?f1+?f3`	y-coordinate of point R

For example B, the path for the square is
`draw:enhanced-path="M ?f2 ?f3 L ?f4 ?f3 ?f4 ?f5 ?f3 ?f5"`

and for the cross
`draw:enhanced-path="M ?f2 ?f3 L ?f4 ?f5 M ?f4 ?f3 L ?f3 ?f5"`

Note both examples A and B work for any width and height value in attribute `svg:viewBox`.

7.6.4 Exercise 7.4

`f0=width/2`	Radius of Thales' circle in local coordinate system Hypotenuse for Pythagorean theorem
`f1=$0-?f0`	Horizontal leg for Pythagorean theorem
`f2=sqrt(?f0*?f0-?f1*?f1)`	Vertical leg by Pythagorean theorem
`f3=?f2*height/width`	Correction for non-square viewBox
`f4=?f3*logwidth/logheight`	Correction for non-square shape
`f5=height-?f4`	Distance from top

```
<draw:enhanced-geometry
        svg:viewBox="0 0 100 50"
        draw:type="non-primitive"
        draw:modifiers="70"
        draw:enhanced-path="M 0 50 L $0 ?f5 100 50 Z">
  <draw:equation draw:name="f0" draw:formula="width/2"/>
  <draw:equation draw:name="f1" draw:formula="$0-?f0"/>
  <draw:equation draw:name="f2" draw:formula="sqrt(?f0*?f0-?f1*?f1)"/>
  <draw:equation draw:name="f3" draw:formula="?f2*height/width"/>
  <draw:equation draw:name="f4" draw:formula="?f3*logwidth/logheight"/>
  <draw:equation draw:name="f5" draw:formula="height-?f4"/>
  <draw:handle draw:handle-position="$0 ?f5"
               draw:handle-range-x-minimum="left"
               draw:handle-range-x-maximum="right"/>
</draw:enhanced-geometry>
```

8 Arcs

8.1 Defining an Arc by Bounding Box and Radial Rays

Table 10: Description for Commands A, B, V and W.

From 'Table 10 - Enhanced path commands'[28] in the ODF standard.

Command Name	Parameters	Description
A arcto	(x1 y1 x2 y2 x3 y3 x4 y4) +	(x1, y1) and (x2, y2) define the bounding box of an ellipse. A line is then drawn from the current point to the start angle of the arc that is specified by the radial vector of point (x3, y3) and then counter-clockwise to the end-angle that is specified by point (x4, y4).
B arc	(x1 y1 x2 y2 x3 y3 x4 y4) +	The same as the "A" command, except that an implied moveto to the starting point is done.
V clockwisearc	(x1 y1 x2 y2 x3 y3 x4 y4)+	The same as the "A" command, except that an implied moveto to the starting point is done and the arc is drawn clockwise.
W clockwise-arcto	(x1 y1 x2 y2 x3 y3 x4 y4) +	The same as the "A" command except, that the arc is drawn clockwise.

The following drawings only differ in the command used. In all cases the shape has size w×h = 15 cm × 6 cm and attribute `svg:viewBox="0 0 1500 600"`

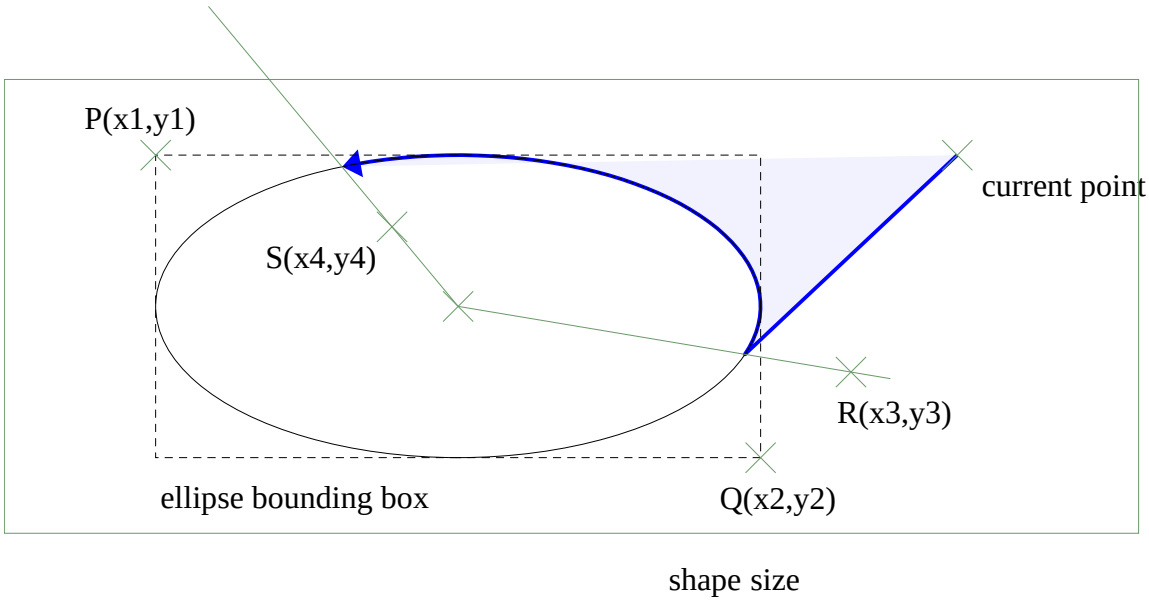


Figure 81: Command **A**

```
draw:enhanced-path="M 1260 100 A 200 100 1000 500 1120 387 513 195"
```

28 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_enhanced-path

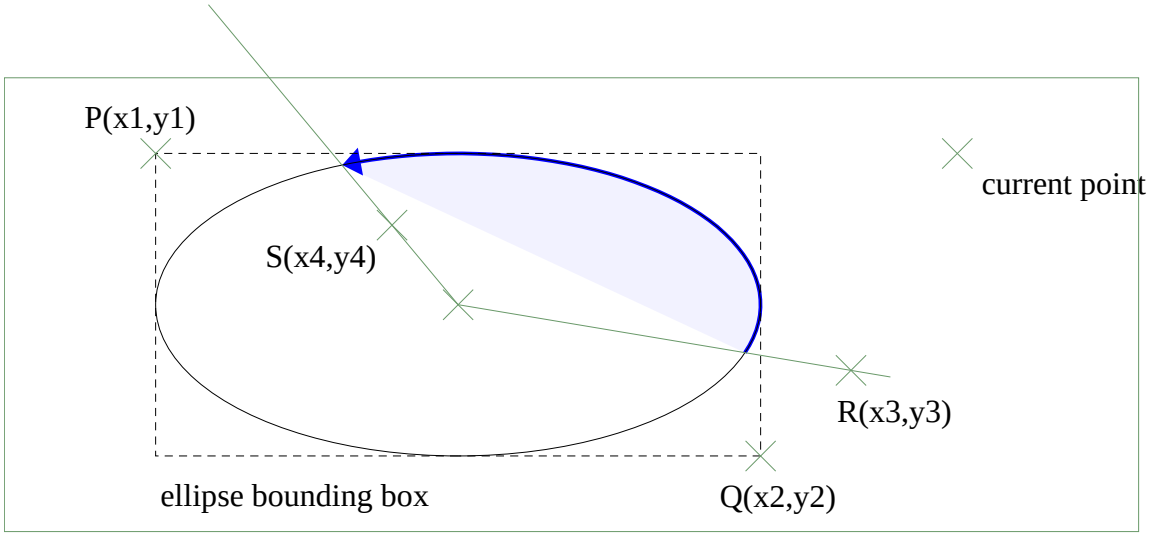


Figure 82: Command ***B***

`draw:enhanced-path="B 200 100 1000 500 1120 387 513 195"`

The command B has an implied ‘moveto’ at the beginning. Therefore, the command `M 1260 100` was omitted here.

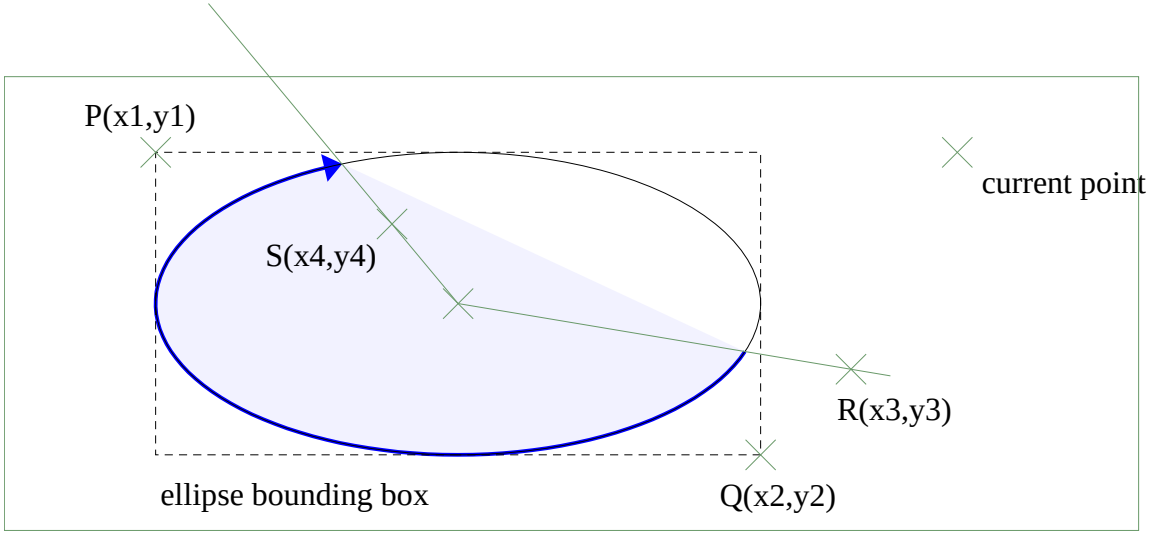


Figure 83: Command ***V***

`draw:enhanced-path="V 200 100 1000 500 1120 387 513 195"`

The command V has an implied ‘moveto’ at the beginning. Therefore, the command `M 1260 100` was omitted here.

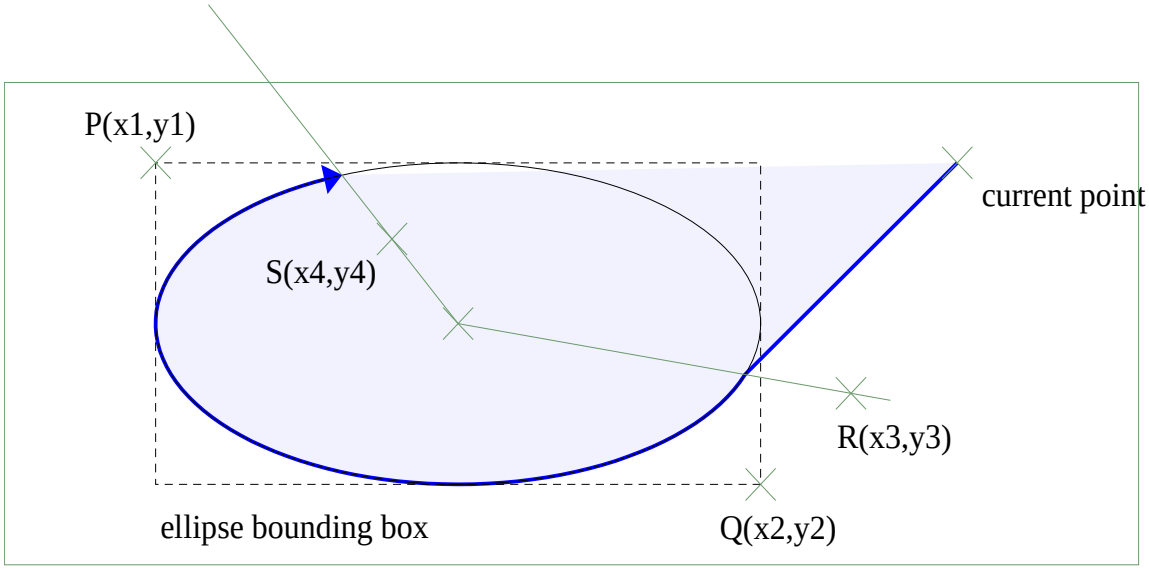


Figure 84: Command ***w***

```
draw:enhanced-path="M 1260 100 W 200 100 1000 500 1120 387 513 195"
```

The arcs have a fill in the examples. Remember, to determine the area to fill, a straight line is drawn from start- to end-point of the sub-path.

The arcs have an arrowhead at the end-point to make the direction of the arcs visible.

Note these arc definitions do not use angles.

> **Exercise 8.1**
>
> Create a circular arc on a square shape so that the underlying circle fills the entire shape. The start point of the arc should be fixed in the middle of the right edge of the shape and the end point should be variable by means of a handle. The arc should be drawn clockwise and should be never filled.
>
> a) Determine the end-point of the arc by an xy-handle.
>
> b) Determine the end-point of the arc by a polar-handle.
>
> → solution

As you noticed, there is no way to bind the xy-handle to the circle line. The visual feedback while dragging the handle is not very good. But in LibreOffice 7.3 you can only set the exact positions in the ‘Position and Size’ dialog for xy-handles, and not for polar handles. Therefore, such a shape is still useful.

> **Exercise 8.2**
>
> Create an arc similar to previous exercise. But now the circle sector which belongs to the arc shall be shown as filled area (if the user has enabled filling). Tip: There has been something similar in chapter ‘Fill and Stroke’.

→ solution

Exercise 8.3

Create a full circle on a square shape.

→ solution

8.2 Defining an Arc by Center, Radii, Start- and End-Angle

Table 11: Description of Commands T and U.

From 'Table 10 - Enhanced path commands'[29] in the ODF standard.

Command Name	Parameters	Description
T angle-ellipseto	(x y wR hR t0 t1) +	Draws a segment of an ellipse. The ellipse is specified by the center(x, y), the width 2*wR, the height 2*hR, and the start-angle t0 in degrees and end-angle t1 in degrees. The segment is drawn clockwise.
U angleellipse	(x y wR hR t0 t1) +	The same as the "T" command, except that an implied moveto to the starting point is done.

The angle is measured clockwise. The first leg is on the positive part of the x-axis. The determined point on the ellipse is the intersection of the second leg with the ellipse line.

So in fact, the angles are used modulo 360°. An angle of −45° is treated the same as an angle of 315°, for example. There is one exception: If t1 = t0 + 360°, a full ellipse is drawn.

If a user touches a handle, then a value in the range [−180°;180] is generated.

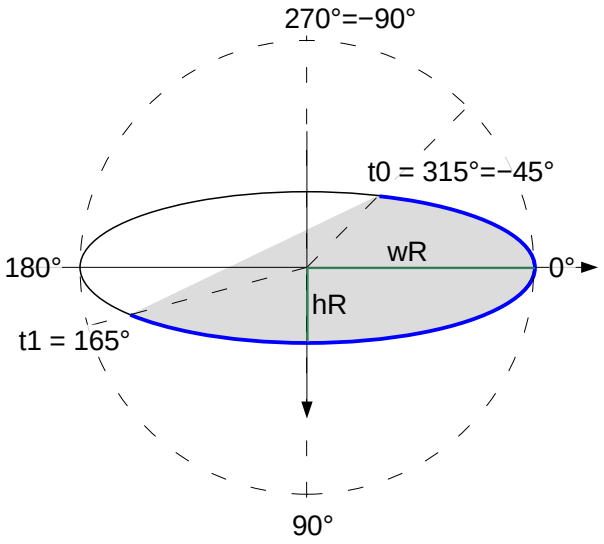


Figure 85: Parameters of commands T and U

Note that this angle orientation is different from the use of start- and end-angle in the shapes `<draw:circle>` and `<draw:ellipse>`, which are available in the 'Legacy Circles and Ovals' toolbar in the UI. For those shapes angles are measured counter-clockwise on screen and in markup.

The angles refer to the local coordinate system. If the shape has a different ratio for width to height than the viewBox of the enhanced-geometry, then the rendered angle will be different from the value in the associated modifier.

29 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_enhanced-path

Exercise 8.4

The Fibonacci spiral can be approximated by quarter circles. An illustration[30] for that is in Wikipedia, and more information on Fibonacci spiral can be found in Wikipedia under topic “Golden spiral”[31].

Create a shape in width to height ratio of 34 to 21 and viewBox value `"0 0 34 21"`, that shows such approximation of the Fibonacci spiral similar to the above-mentioned illustration. Do this by using

a) command T

b) command U

c) command X or Y (see chapter ‘Quadrants’)

→ solution

Now we set a line style with arrowhead to the shapes generated in the exercise and compare them.

Command T or Y *Command U*

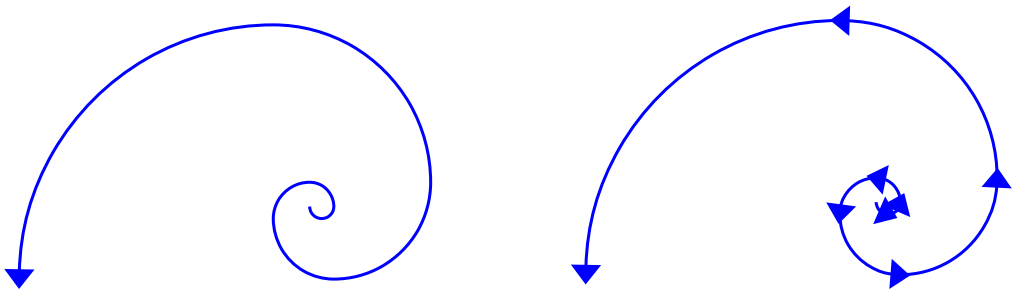


Figure 86: compare commands T or Y with command U in regard to arrowhead

The essential difference between commands T and U is, that a sequence of commands U generates a sequence of several sub-paths because of the implied ‘moveto’ of command U, whereas commands T and Y generate one sub-paths. And because each sub-path gets its own arrowhead in LibreOffice[32], there are arrow heads inside the spiral if using a sequence of commands U.

Some use cases can be easier solved with commands T and U than with commands A, B, V or W. Compare the following solutions with those from exercise 8.1.

Example full circle:

Commands U and T have a special case for full circle

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
      draw:type="non-primitive"
      draw:enhanced-path="U 2000 2000 2000 2000 0 360"/>
```

30 https://commons.wikimedia.org/wiki/File:FibonacciSpiral.svg
31 https://en.wikipedia.org/wiki/Golden_spiral
32 There is ongoing discussion about arrow heads in the ODF TC. But any standardization of the behavior is earliest available in ODF 1.4.
https://issues.oasis-open.org/browse/OFFICE-4111

Example: Shape with handles for start and end point.

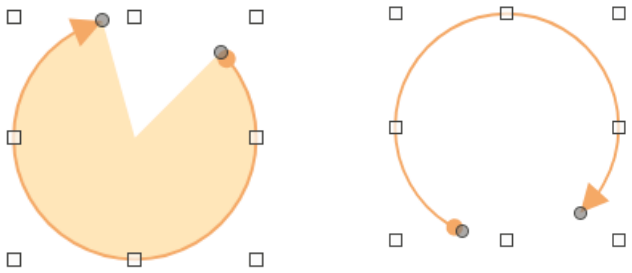


Figure 87: Arc with handle for start and for end point

The modifiers can be used directly, no calculation of circle points is needed.

Arc with sector fill

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
      draw:type="non-primitive" draw:modifiers="-45 -105"
      draw:enhanced-path="M 2000 2000 T 2000 2000 2000 2000 $0 $1 Z S N
                          U 2000 2000 2000 2000 $0 $1 F">
  <draw:handle draw:handle-position="2000 $0" draw:handle-polar="2000 2000"/>
  <draw:handle draw:handle-position="2000 $1" draw:handle-polar="2000 2000"/>
</draw:enhanced-geometry>
```

Arc with only line

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
      draw:type="non-primitive" draw:modifiers="115 50"
      draw:enhanced-path="U 2000 2000 2000 2000 $0 $1 F N">
  <draw:handle draw:handle-position="2000 $0" draw:handle-polar="2000 2000"/>
  <draw:handle draw:handle-position="2000 $1" draw:handle-polar="2000 2000"/>
</draw:enhanced-geometry>
```

8.3 Defining an Arc by Start-Angle and Swing-Angle

This command is an extension to the ODF 1.3 standard and is only usable in a `drawooo:enhanced-path` attribute. This command will be available in the ordinary `draw:enhance-path` attribute in version 1.4 of the ODF standard.

Table 12: Description of the command G.

From issue OFFICE-4026 in the ODF TC bug tracker[33].

Command Name	Parameters	Description
G arcangleto	(wR hR st sw) +	Draws an arc of an ellipse. The ellipse is given by the width 2*wR, and the height 2*hR. The arc is specified by the start angle st in degrees and the swing angle sw in degrees. The ellipse is positioned so that the current point is the start point of the arc. The arc is drawn so that sw is the central angle of the arc. If sw is positive, the arc is drawn clockwise; if sw is negative, the arc is

33 https://issues.oasis-open.org/browse/OFFICE-4026

		drawn counter-clockwise. The endpoint of the arc becomes the new current point. In case the path is filled, the filling is drawn as an ellipse segment. The swing angle may be greater than 360° and may be smaller than −360°. In such cases the fill will overlap.

Command G has essential differences to the commands T and U:

- The arc is attached to the current point and continues the current sub-path. Command T continues the current sub-path but draws an additional straight line segment. Command U starts a new sub-path.
- By using negative swing angles, it is possible to draw arcs counter-clockwise. Commands T and U draw arcs always clockwise. The directions become visible if arrowheads are shown.

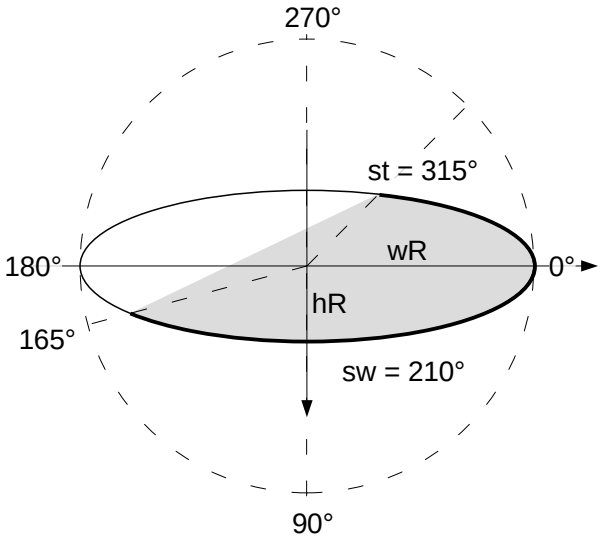


Figure 88: Parameters of command G

As example we will develop the markup for a “wave” shape as in figure 89.

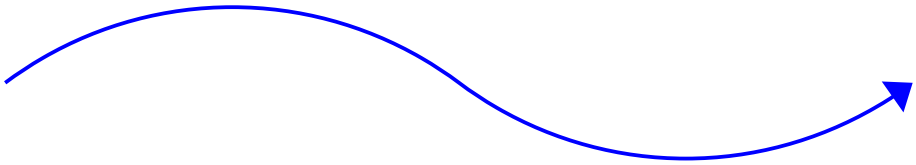


Figure 89: "wave" shape

This shape is combined from two circle arcs. The illustration in figure 90 shows details of the construction.

We use the viewBox value `"0 0 120 20"`. In this coordinate system the start point of the “wave” has coordinates `(0|10)` and the end point `(120|10)`. The center points of the circles are M_1`(30|50)` and M_2`(90|-30)`. Notice, that the center points are outside the viewBox. That is allowed.

To use command G we need the radii of the circles, the angle, which corresponds to the start point of the arc, and the “swing” angle, that is the center angle of the arc.

The radius *r* of the circle is determined by a center and a point on the circle circumference. With Pythagoras’ theorem we get

$$r=\sqrt{(30-0)^2+(50-10)^2}=50$$

Same for the other circle

$$r=\sqrt{(120-90)^2+(10-(-30))^2}=50$$

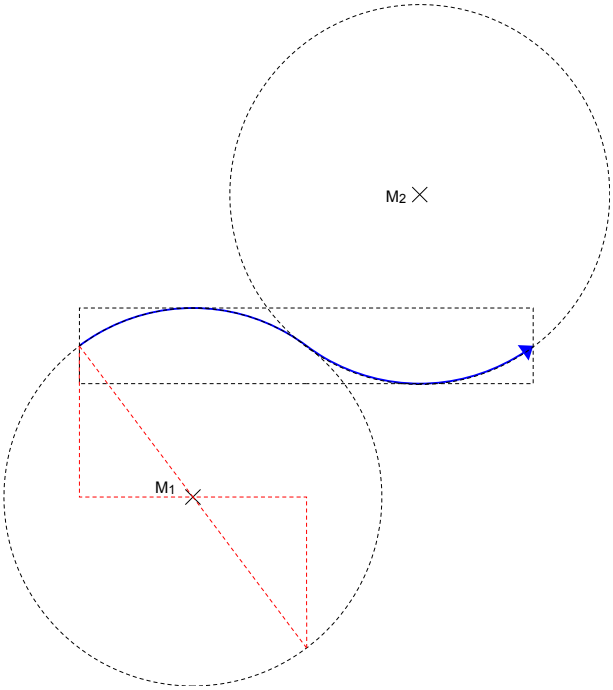


Figure 90: Circles used for "wave" shape

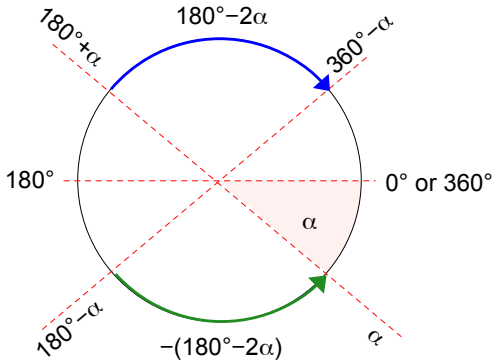


Figure 91: "wave" shape, overview of involved angles

An overview of the involved angles is given in figure 91.

To calculate angle α we use the same right triangle as for the radius.

```
f0=atan2(40,30)*180/pi
```

The other angles can be derived from angle α.

Start angle for first arc (clockwise) `f1=180+?f0`

Swing angle for first arc `f2=180-2*?f0`

Start angle of second arc (counter-clockwise) `f3=180-?f0`

Swing angle for second arc `f4=-?f2`, same angle as for first arc but opposite direction

The enhanced-path is then

```
drawooo:enhanced-path="                command G needs special namespace
    M 0 10                             move to start point
    G 50 50 ?f1 ?f2                    first arc
                                       current point is correct for second arc
    50 50 ?f3 ?f4"                     second arc
```

8.4 Tangent to Circular Arc

We are going to create a shape with a very simple ‘heart’. The heart should have the same size as the shape. A first idea is shown in figure 92. But that solution is not nice, because the transition between the arc and the straight line is not smooth.

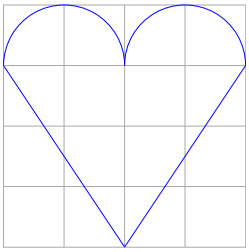


Figure 92: Heart – first, not appealing version

We need to find a point of contact on the circle, so that a straight line through the middle of the bottom edge becomes a tangent to the circle, see figure 94.

The final shape should look like these

Figure 93: Heart with smooth transition, filled and unfilled

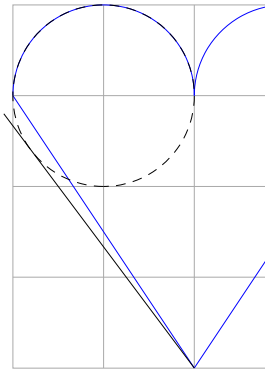


Figure 94: Heart – tangent

To compute a point of contact for a tangent and calculate the associated angles look at some formulas in chapter ‘11 Mathematics’. In the ‘heart’ shape case, it is $x_B = x_M + r$, which makes some formulas simpler.

Use the viewBox value `"0 0 4000 4000"`, which fits to the grid in the illustrations.

M_{left} (1000|1000), M_{right} (3000|1000), radius=1000, B(2000|4000), P(2000|1000)

$k = |PB| = |RB| = 3000$

$$|MB| = \sqrt{1000^2 + 3000^2} = 1000 \cdot \sqrt{10}$$

We calculate the angles as shown in section 11.2.3.

$$\beta = \operatorname{atan2}(3000\ ,\ 1000)$$

$$\gamma = 2 \cdot \beta = 2 \cdot \operatorname{atan2}(3000\ ,\ 1000)$$

The angle corresponding to point R is γ.

We then calculate the coordinates of point R as shown in section 11.2.2.

$$x_R = \frac{1000}{1000^2 + 3000^2} \cdot \left(1000^2 - 3000^2\right) + 1000 = 200$$

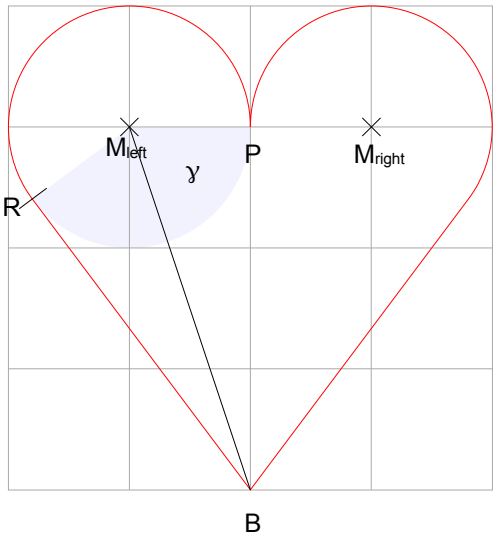


Figure 95: Design a heart with smooth transition

$$y_R = \frac{1000}{1000^2 + 3000^2} \cdot 2000 \cdot 3000 + 1000 = 1600$$

The angle of point P with respect to left circle is 360°, in regard to right circle it is 180°.

The angle and coordinates for the right point of contact can be calculated from symmetry. It is angle = 180°−γ and coordinates $(4000 - x_R \mid y_R) = (3800 \mid 1600)$.

> **Exercise 8.5**
>
> Now develop the markup for this ‘heart’ shape. The heart shall be drawn as one sub-path. Find a solution for each of the three ways to draw arcs. You need to be careful to use correct parameters. All these commands draw an arc, but the parameters are quite different.
>
> → solution

8.5 Binding a Handle to an Ellipse Line

In case of a circular arc it is possible to bind a polar handle to the circle line by putting the circle radius as constant into the `draw:handle-position` attribute. That is not possible in case of an elliptical arc, because an ellipse has no unique radius.

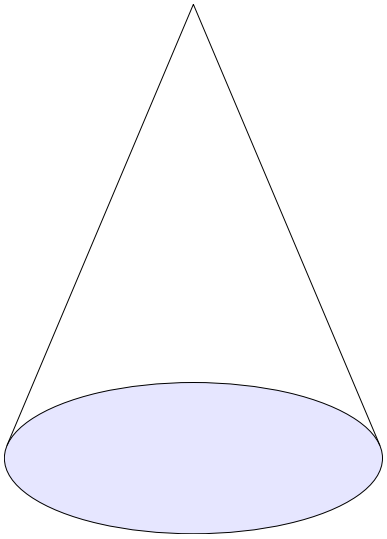


Figure 96: Cone

The problem is shown on an example with a cone. The shape here is 5 cm wide and 7 cm high. The ellipse for the base area is 2 cm high. The shape uses viewBox value `"0 0 5000 7000"`, so the ellipse has semi-major axis 2500 and semi-minor axis 1000, center of the ellipse is (2500|6000) and top of the lines is (2500|0) in local coordinate system.

The lines are tangent to the ellipse. You can create the shape with the help of section 11.2.4. But that is not topic here and therefore the formulas so far are given immediately. Comments on them are in the exercise solutions.

```
f0=2500/100
f1=?f0*6000
f2=sqrt(?f1*?f1-2500*2500)
f3=2500-2500/?f1*?f2
```

```
f4=2500+25007?f1*?f2
f5=2500/?f1*2500/?f0
f6=6000-?f5
```

The enhanced path is then so far

```
draw:enhanced-path="
  U 2500 6000 2500 1000 0 360 N
  M ?f3 ?f6 L 2500 0 ?f4 ?f6 F">
```

Command U can draw a full ellipse. End of set, otherwise the comment F, needed for the lines, would affect the ellipse too.

Command F disables area fill for the lines. It is needed because in custom shapes open paths can be filled too.

The cone is now extended by a triangle. This illustration is common in mathematics teaching to calculate the length of a surface line of a cone. The special feature here is that the position of the triangle can be varied by a handle on the ellipse line.

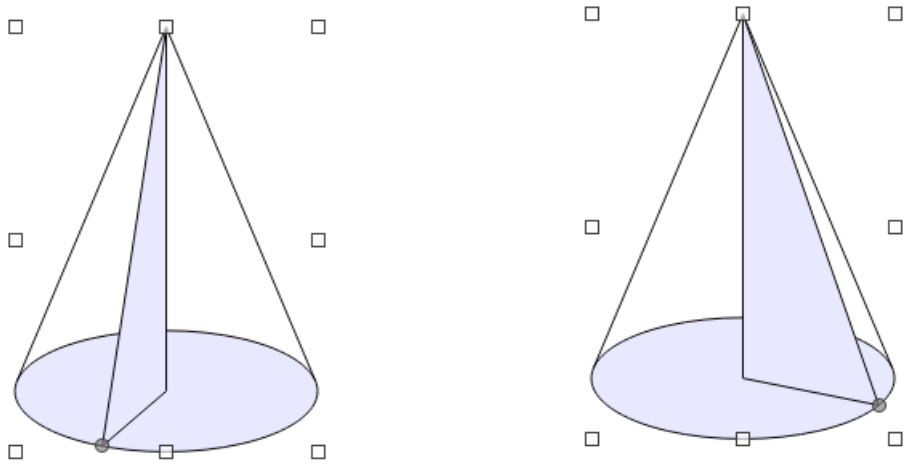


Figure 97: cone with surface line in various positions

A polar handle is determined by the distance *r* to the center and the angle γ. The angle will be connected to a modifier. We will calculate the distance *r* so, that the handle is on the ellipse line.

To compute the distance *r* we can treat the ellipse as if it were centered to origin. Then all points with angle γ have coordinates $(r \cdot \cos(\gamma) \mid r \cdot \sin(\gamma))$. On the other hand, the position of the handle is a point on the ellipse line and as such needs to fulfill the ellipse equation in addition. For the ellipse in the example with radii 2500 and 1000 this is

$$\left(\frac{r \cdot \cos(\gamma)}{2500}\right)^2 + \left(\frac{r \cdot \sin(\gamma)}{1000}\right)^2 = 1$$

We resolve this to $r = \frac{1000 \cdot 2500}{\sqrt{(1000 \cdot \cos(\gamma))^2 + (2500 \cdot \sin(\gamma))^2}}$

We not only need the distance *r* but also the Cartesian coordinates of the handle to be able to draw the triangle.

$x = 2500 + r \cdot \cos(\gamma)$ and $y = 6000 + r \cdot \sin(\gamma)$

The complete markup is given in the exercise solution.

> **Exercise 8.6**
>
> Create the complete `<draw:enhanced-geometry>` element for this shape. This is a repetition exercise for creating polar handles (section 3.6) and for area fill (section 5).
>
> As repetition exercise for section 5.4 you might make the triangle fill lighter than the base area fill.
>
> → solution

8.6 Exercise Solutions

8.6.1 Exercise 8.1

Exercise 8.1 a)

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
      draw:type="non-primitive" draw:modifiers="1000 1000"
      draw:enhanced-path="V 0 0 4000 4000 4000 2000 $0 $1 F N">
  <draw:handle draw:handle-position="$0 $1"
  draw:handle-range-x-maximum="4000"
  draw:handle-range-x-minimum="0"
  draw:handle-range-y-maximum="4000"
  draw:handle-range-y-minimum="0"/>
</draw:enhanced-geometry>
```

Note the command **F** which disables filling.

The advantage of using an xy-handle is that the user can set the exact position in the dialog ‘Position and Size’. The disadvantage is that the handle cannot be bound to the circle line. The handle has no obvious connection to the arc. We discussed this problem already in chapter ‘3.5 Functions’ when a right triangle via Thales’ circle was created.

Exercise 8.1 b)

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
      draw:type="non-primitive" draw:modifiers="-145"
      draw:enhanced-path="V 0 0 4000 4000 4000 2000 ?f0 ?f1 F N">
  <draw:equation draw:name="f0" draw:formula="2000+2000*cos($0*(pi/180))"/>
  <draw:equation draw:name="f1" draw:formula="2000+2000*sin($0*(pi/180))"/>
  <draw:handle draw:handle-position="2000 $0" draw:handle-polar="2000 2000"/>
</draw:enhanced-geometry>
```

Setting a fixed radius of `2000` forces the polar handle to move on the circle line.

Because the handle returns an angle, and the command A requires point coordinates, these have to be calculated. Therefore the unit degree has to be converted to radians to be usable with functions sin and cos.

8.6.2 Exercise 8.2

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
```

```
      draw:type="non-primitive" draw:modifiers="-120"
      draw:enhanced-path=
          "M 2000 2000 W 0 0 4000 4000 4000 2000 ?f0 ?f1 Z S N
           V 0 0 4000 4000 4000 2000 ?f0 ?f1 F">
  <draw:equation draw:name="f0" draw:formula="2000+2000*cos($0*(pi/180))" />
  <draw:equation draw:name="f1" draw:formula="2000+2000*sin($0*(pi/180))" />
  <draw:handle draw:handle-position="2000 $0"
               draw:handle-polar="2000 2000" />
</draw:enhanced-geometry>
```

The filled sector needs a separate set, done by command `N`, otherwise the commands F and S will be applied to both – arc and sector – and nothing is rendered.

8.6.3 Exercise 8.3

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000" draw:type="non-primitive"
      draw:enhanced-path=
          "V 0 0 4000 4000 4000 2000 0 2000
             0 0 4000 4000 0 2000 4000 2000" />
```

If the start and end points are the same point of the ellipse, then the center angle of the arc is 0° and nothing is drawn. To circumvent this, draw the circle in two parts.

Currently the path is set to “closed” because start- and endpoint of the arc are the same[34]. Since the arc is “closed” there is no arrowhead. If you need an arrowhead, set the end point a little offset, for example `4000 1999` instead of `4000 2000`.

8.6.4 Exercise 8.4

To use commands U and T you need to start at the outer end of the spiral, because commands U and T draw arcs always clockwise. Using commands X or Y the orientation is automatically correct.

You can compute the point coordinates and radii needed directly from the grid in the illustration.

a) Command U

```
<draw:enhanced-geometry svg:viewBox="0 0 34 21" draw:type="non-primitive"
draw:enhanced-path=
"U 21 21 21 21 180 270
 21 13 13 13 270 360
 26 13 8 8 0 90
 26 16 5 5 90 180
 24 16 3 3 180 270
 24 15 2 2 270 360
 25 15 1 1 0 90
 25 15 1 1 90 180"/>
```

Line breaks are inserted here to make the single quarter arcs more visible. Note the Fibonacci sequence in the radii.

b) Command T

Command T needs a current point; therefore, requires a command M to start.

34 I consider this a bug, see case B in bug https://bugs.documentfoundation.org/show_bug.cgi?id=144390

```
<draw:enhanced-geometry svg:viewBox="0 0 34 21" draw:type="non-primitive"
draw:enhanced-path=
"M 0 21
 T 21 21 21 21 180 270
 21 13 13 13 270 360
 26 13 8 8 0 90
 26 16 5 5 90 180
 24 16 3 3 180 270
 24 15 2 2 270 360
 25 15 1 1 0 90
 25 15 1 1 90 180"/>
```

When comparing with the solution using command U you see the same parameters.

c) Command Y

Command Y needs a current point; therefore, requires a command M to start.

Drawing with start inside:

```
<draw:enhanced-geometry svg:viewBox="0 0 34 21"  draw:type="non-primitive"
draw:enhanced-path="M 24 15 Y 25 16 26 15 24 13 21 16 26 21 34 13 21 0 0 21"/>
```

Drawing with start outside:

```
<draw:enhanced-geometry svg:viewBox="0 0 34 21" draw:type="non-primitive"
draw:enhanced-path="M 0 21 Y 21 0 34 13 26 21 21 16 24 13 26 15 25 16 24 15"/>
```

When comparing with the other solutions, you see here a shorter version. Each quarter circle needs only one coordinate pair.

8.6.5 Exercise 8.5

Using start- and endpoint

We start at the bottom corner of the heart and draw the arcs clockwise. Then command W is suitable. It provides the starting straight line. The ending straight line is drawn by a ‘close’ command.

Command A would be suitable too, but you would need to start with the right arc. Commands B and V are not suitable because they start a new sub-path.

Command W has the parameters:
left, top and right, bottom of bounding box of ellipse, start-point of arc, end-point of arc

```
draw:enhanced-path="M 2000,4000          bottom corner of the heart
       W 0,0 2000,2000                   bounding box of left circle
      200,1600                           start point of arc in left circle
      2000,1000                          end point of arc in left circle
                                         command W again, we omit the character
      2000,0 4000,2000                   bounding box of right circle
      2000,1000                          start point of arc in right circle
      3800,1600                          end point of arc in right circle
```

`Z"`	closing straight line to initial point (2000|4000)

Using start-angle and end-angle

We start again at the bottom corner of the heart and draw the arcs clockwise. Command T is suitable. It provides a starting straight line from current point to start of arc. The ending straight line is drawn by a ‘close’ command.

Command U is not suitable, because it starts a new sub-path.

Command T has the parameters:
center, horizontal radius, vertical radius (of ellipse), start-angle for arc, end-angle for arc.

To get the angles, use the trigonometric functions, therefore we need some formulas.

`f0=2*atan2(3000,1000)`	angle γ in radians
`f1=?f0*180/pi`	angle γ in degrees
`f2=180-?f1`	end-angle in degrees for right arc

And now here is the complete markup, commented

```
<draw:enhanced-geometry
  svg:viewBox="0 0 4000 4000"
  draw:type="non-primitive"
  draw:enhanced-path="M 2000 4000          bottom corner of the heart
      T 1000,1000 1000 1000                center left, horizontal and vertical radius
      ?f1 360                              start-angle and end-angle for left arc
                                           command T again, we omit the character
      3000,1000 1000 1000                  center right, horizontal and vertical radius
      180 ?f2                              start-angle and end-angle for right arc
      Z">                                  closing straight line to (2000|4000), which is
                                           initial point of the sub-path
  <draw:equation draw:name="f0" draw:formula="2*atan2(3000,1000)" />
  <draw:equation draw:name="f1" draw:formula="?f0*180/pi" />
  <draw:equation draw:name="f2" draw:formula="180-?f1" />
</draw:enhanced-geometry>
```

Using start-angle and swing-angle

Command G has the parameters:
horizontal radius, vertical radius, start angle of arc, swing angle of arc

Because command G attaches the arc to the current point, you need to provide the start point of the arc as current point. The start point was above calculated from as special case. Here we show the more general way by $x_R = r \cdot \cos(\gamma) + x_M$ and $y_R = r \cdot \sin(\gamma) + y_M$. Because trigonometric functions are involved, we use formulas for that.

`f0=2*atan2(3000,1000)`	angle γ in radians

`f1=?f0*180/pi`	angle γ in degrees
`f2=1000*cos(?f0)+1000`	x-coordinate of start point of left arc
`f3=1000*sin(?f0)+1000`	y-coordinate of start point of left arc
`f4=360-?f1`	swing-angle in degrees, same for left and right arc

And now here is the complete markup, again commented

```
<draw:enhanced-geometry
  svg:viewBox="0 0 4000 4000"
  draw:type="non-primitive"
  drawooo:enhanced-path="                command G needs special namespace
      M 2000 4000                        bottom corner of the heart
      L ?f2 ?f3                          Line to start point of arc, current point for G
      G 1000 1000                        horizontal and vertical radius
      ?f1 ?f4                            start-angle and swing-angle for left arc
                                         command G again, we omit the character
      1000 1000                          horizontal and vertical radius
      180 ?f4                            start-angle and swing-angle for right arc
      Z">                                closing straight line to (2000|4000), which is
                                         initial point of the sub-path
  <draw:equation draw:name="f0" draw:formula="2*atan2(3000,1000)" />
  <draw:equation draw:name="f1" draw:formula="?f0*180/pi" />
  <draw:equation draw:name="f2" draw:formula="1000*cos(?f0)+1000" />
  <draw:equation draw:name="f3" draw:formula="1000*sin(?f0)+1000" />
  <draw:equation draw:name="f4" draw:formula="360-?f1" />
</draw:enhanced-geometry>
```

8.6.6 Exercise 8.6

Formulas needed for the tangent lines

`f0=2500/100`	scale factor in y-direction to transform the ellipse to a circle
`f1=?f0*6000`	distance from pole to center in regard to the circle
`f2=sqrt(?f1*?f1-2500*2500)`	distance *k* as in section 11 in regard to the circle, that is the distance between pole and point of contact
`f3=2500-2500/?f1*?f2`	x-coordinate of left point of contact. The x-coordinate is not affected by vertical scaling and mirroring.
`f4=2500+25007?f1*?f2`	x-coordinate of right point of contact.

`f5=2500/?f1*2500/?f0`	y-coordinate, still mirrored and translated (see section 11.2.4). Reverse scaling is applied.
`f6=6000-?f5`	final y-coordinate for left and right point of contact. Now revers mirror and translate is applied.

Formulas needed for the handle

The value r is rather complex. Therefore the calculation is divided in smaller parts. You may write it into one formula.

`$0`	modifier, bound to the angle of the handle, value in degrees
`f7=1000*cos($0*pi/180)`	first summand in radicand. Notice the angle conversion from degrees to radians. You might use a separate formula for that. Or you use separate formula for the complete cos expression and the complete sin expression. They are used again in calculating the coordinates.
`f8=2500*sin($0*pi/180)`	second summand in radicand
`f9=1000*2500/sqrt(?f7*?f7+?f8*?f8)`	complete value of radius *r*

Formulas for the position of the handle, needed for the triangle.

`f10=2500 + ?f9*cos($0*pi/180)`	x-coordinate of handle point, corner of the triangle
`f11=6000 + ?f9*sin($0*pi/180)`	y-coordinate of handle point, corner of the triangle

Complete enhanced-geometry element

```
<draw:enhanced-geometry
    svg:viewBox="0 0 5000 7000" draw:type="non-primitive"
    draw:modifiers="140"                    Without this attribute handles will not work.
    draw:enhanced-path="U 2500 6000 2500 1000 0 360 N
    M ?f3 ?f6 L 2500 0 ?f4 ?f6 F
    N                                       End of set, otherwise the previous command F
                                            would affect the triangle too.
    M 2500 0 L ?f10 ?f11 2500 6000 Z        Triangle
    ">                                      End of attribute enhanced-path
<draw:equation draw:name="f0" draw:formula="2500/1000"/>
```

```
<draw:equation draw:name="f1" draw:formula="?f0*6000"/>
<draw:equation draw:name="f2" draw:formula="sqrt(?f1*?f1-2500*2500)"/>
<draw:equation draw:name="f3" draw:formula="2500-2500/?f1*?f2"/>
<draw:equation draw:name="f4" draw:formula="2500+2500/?f1*?f2"/>
<draw:equation draw:name="f5" draw:formula="2500/?f1*2500/?f0"/>
<draw:equation draw:name="f6" draw:formula="6000-?f5"/>
<draw:equation draw:name="f7" draw:formula="1000*cos($0*pi/180)"/>
<draw:equation draw:name="f8" draw:formula="2500*sin($0*pi/180)"/>
<draw:equation draw:name="f9" draw:formula="1000*2500/sqrt(?f7*?f7+?f8*?f8)"/>
<draw:equation draw:name="f10" draw:formula="2500+?f9*cos($0*pi/180)"/>
<draw:equation draw:name="f11" draw:formula="6000+?f9*sin($0*pi/180)"/>
```

```
<draw:handle
```

Each handle is defined by an own element. Be careful to put the handle elements after the equation elements.

```
  draw:handle-polar="2500 6000"
```

Using this attribute make the handle to a polar handle. The values determine the center of the handle movement.

```
  draw:handle-position="?f9 $0"/>
```

The first value determines the distance from center to handle position. Here it is the calculated value *r*. The second value is the angle. Here it is a reference to a modifier. At least one value must be a reference to a modifier, otherwise the handle will not move.

```
</draw:enhanced-geometry>
```

9 Text Path

9.1 Attribute Overview

This chapter is about artistic-shaped text in our custom shapes. LibreOffice provides some predefined shapes in the Fontwork-dialog and allows your own geometries too.

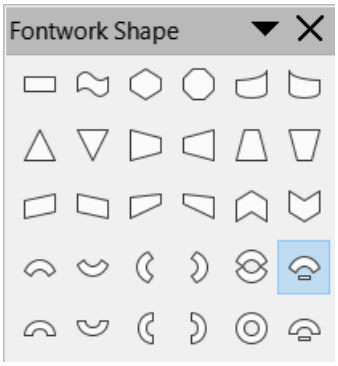


Figure 98: Predefined Fontwork shapes in LibreOffice

Similar shapes exist as “WordArt” in old, binary Microsoft Office. The OOXML standard has a fixed set of “TextWarpDefinitions” for such shapes, which are available as “abc Transform” in modern versions of Microsoft Office. VML has the element “TextPath”.[35]

The ODF standard provides some attributes of the enhanced geometry to describe such shapes:

`draw:text-path`

Its value is `true` or `false`, default to `false`. If `true`, text and path of the custom shape are used in a special way, as explained below.

LibreOffice interprets the `draw:text-path` attribute for all custom-shapes. It has no user interface to let you change its value.

`draw:text-path-allowed`

Values are `true` or `false`, with `false` as default. The definition in ODF 1.3 is, “The draw:text-path-allowed attribute specifies whether the user interface of a consumer that supports the `draw:text-path` 19.221 attribute should allow modification of the value of the `draw:text-path` attribute.”[36]. The definition in ODF 1.1 is, “The `draw:text-path-allowed` attribute specifies if the shape is capable of being rendered as Fontwork object.”[37]

The meaning is unclear to me. LibreOffice does not interpret this attribute.

35 https://docs.microsoft.com/en-us/windows/win32/vml/msdn-online-vml-textpath-element
36 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_text-path-allowed
37 Section “Text Path Allowed” on page 339 in OpenDocument-v1.1-os.odt, 1 February 2007

`draw:text-path-mode`

Accepted values are `shape`, `normal` or `path`, with `normal` as default. The ODF standard[38] defines the meaning as
“`normal`: text is drawn along the path without scaling;
`path`: text is fitted to a path;
`shape`: text is fitted to the bounding box of a shape.”

LibreOffice does not interpret this attribute. The attribute value `path` better corresponds to the way LibreOffice renders Fontwork shapes. Because the default value is `normal`, your shapes should not omit the attribute but have it with value `path`.

`draw:text-path-scale`

Values are `shape` or `path`, with `path` as default value. The ODF standard defines the meaning as “`path`: text scaling is determined by the length of the path from the `draw:enhanced-path` 19.145 attribute.
`shape`: text scaling is determined by the width of a shape.” [39]

More details how LibreOffice interprets this property are given below.

`draw:text-path-same-letter-heights`

Values are `true` or `false`, with `false` as default value. If set to `true`, all letters are scaled in height-direction to have the same height. LibreOffice has a user interface for it in the Fontwork toolbar, so you do not need to care about it in your shapes.

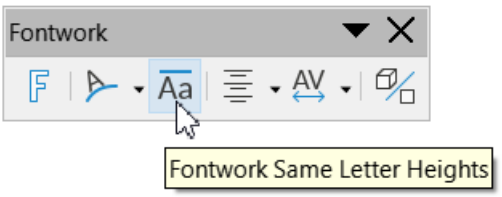


Figure 99: Fontwork Same Letter Heights

38 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_text-path-mode
39 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_text-path-scale

9.2 Common Properties

Figure 100: Compare normal shape with same shape with text path enabled

The normal shape (left) has the markup

```
<draw:enhanced-geometry svg:viewBox="0 0 21600 21600" draw:type="non-primitive"
    draw:enhanced-path="M 0 21600 Y 10800 0 21600 21600 Z N"/>
```

The “Fontwork” shape (right) has the markup

```
<draw:enhanced-geometry svg:viewBox="0 0 21600 21600" draw:type="non-primitive"
    draw:text-path="true"
    draw:enhanced-path="M 0 21600 Y 10800 0 21600 21600 Z N"/>
```

The markup is the same, only that the ‘Fontwork’ shape has the additional attributes to enable text path. We see

Figure 101: Underlying path in 'Fontwork' shape

- Settings from tab ‘Font Effects’ are ignored, here the property ‘outline’ and the brown character color.
- The area fill of the shape is used to fill the letters. In LibreOffice, the entire text gets one common fill. OpenOffice.org versions fill each letter individually.
- The stroke settings of the shape are used in the outline of the letters.
- Word wrap is ignored.
- Paragraph alignment is ignored.
- The path defined in the enhanced-path attribute is no longer rendered as path. It still exists and can be seen, if option ‘Modify Objects with Attributes’ in the ‘Options’ toolbar is disabled and you start rotation, for example, see screenshot in figure 101.

BTW, you get a ‘nicer’ Fontwork shape, if you omit the command Z in the enhanced path and set the alignment to ‘Center’.

9.3 One Path Line

9.3.1 Value `shape` of Attribute `text-path-scale`

The examples here are based on following shape

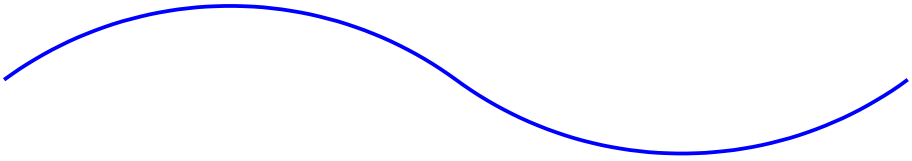


Figure 102: Shape consisting of two circle arcs

The markup is

```
<draw:enhanced-geometry svg:viewBox="0 0 120 20" draw:type="non-primitive"
  draw:enhanced-path="V -20 0 80 100 0 10 60 10 A 40 -80 140 20 60 10 120 10"/>
```

Then add text-path attributes

```
<draw:enhanced-geometry svg:viewBox="0 0 120 20" draw:type="non-primitive"
  draw:text-path="true" draw:text-path-mode="path" draw:text-path-
scale="shape"
  draw:enhanced-path="V -20 0 80 100 0 10 60 10 A 40 -80 140 20 60 10 120 10"/>
```

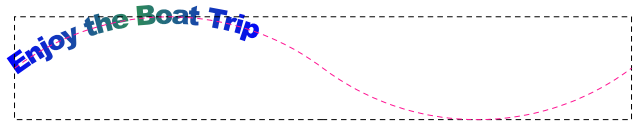
The text is displayed in its original size. Mixing text parts with different sizes is not possible.

Only when the text width exceeds the path length, then the font size is reduced to fit the length. The font resizing affects only the rendering. The nominal font size in the style of the shape is not changed.

The text is then attached to the path. Each letter is rotated to be perpendicular to the path.

Table 13: Different font sizes rendered by text-path-scale attribute value 'shape'

A dashed black line indicates the shape size, a pink dashed line indicates the path.

	15 pt
	24 pt
	40 pt The comparison with the 40 pt large text in the rectangle shows, that the font size in the Fontwork shape is reduced for rendering.

Prior to LO 7.3 the rendering has a bug[40], in which an additional padding of about 10% is introduced in the path length. As result, the text could not fill the whole path.

> **Exercise 9.1**
>
> Use a Fibonacci spiral from exercise 8.4 to create a Fontwork shape for a spiral text.
>
> → solution

9.3.2 Value `path` of Attribute `text-path-scale`

If the value of the attribute `text-path-scale` is `path`, then the font size for rendering adjusted, so that the text attached to the path covers the entire path.

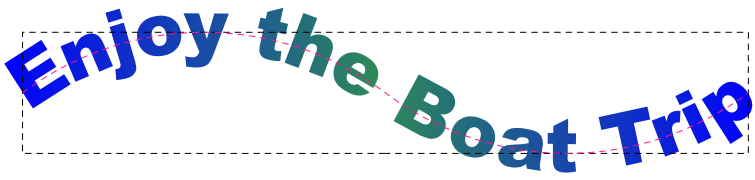


Figure 103: text-path-scale value 'path'

> **Exercise 9.2**
>
> The predefined Fontwork shapes ‘Arch Up (Curve)’, ‘Arch Down (Curve)’, ‘Arch Left (Left)’, ‘Arch Right (Curve)’ and ‘Circle (Curve)’ are shapes of this kind. But all of them have only one handle for the arc and the shapes are symmetric to horizontal or vertical axis.
>
> Create a Fontwork shape, which puts the text on an arc. The start- and end-angle of the arc shall be adjustable by handles.
>
> → solution

9.3.3 Text with Multiple Text Lines

The behavior is similar if the text has multiple lines. The text is then treated as a single block. The middle of the text block is aligned with the path as default, but the anchor can be changed, see section 9.6. The longest line determines the font size to render.

To create multiple lines you must make different paragraphs using the Enter-key. Automatic or manual line break using shortcut Shift-Enter is ignored.

Figure 104: text-path-scale value 'shape' with two paragraphs

The rendering has bugs[41] currently if the text block is wider than the path length. If you get unexpected results, check whether the nominal font size is too large.

40 https://bugs.documentfoundation.org/show_bug.cgi?id=145004
41 https://bugs.documentfoundation.org/show_bug.cgi?id=145004

In the current implementation the distance from the start is calculated for each letter, assuming a straight text line. This distance is measured along the possible curved path and determines a point on the path. Then an imaginary perpendicular line (the normal) is drawn through this point and the character is placed and aligned on it. Thus, characters on the outer side of a curve are more spaced than characters on the inner side. Characters can even overlap in sharp curves.

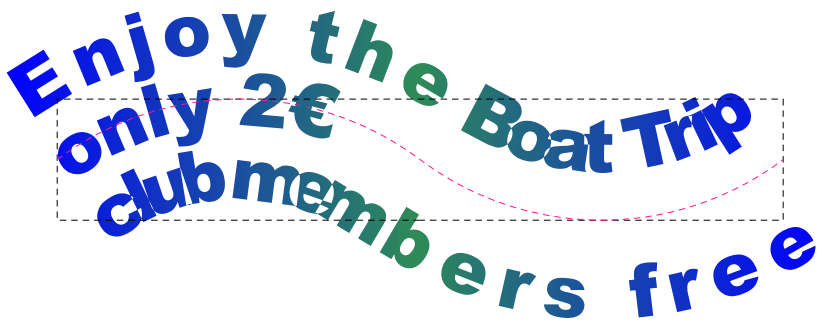


Figure 105: Rendering of multi-line text on a curve

9.4 Two Sub-Paths

If the shape has two sub-paths, the text block is transformed to fit between the two lines. The top of the text block is bound to the first line, the bottom of the text block is bound to the second line. Most of the preset Fontwork shapes have a geometry with two sub-paths.

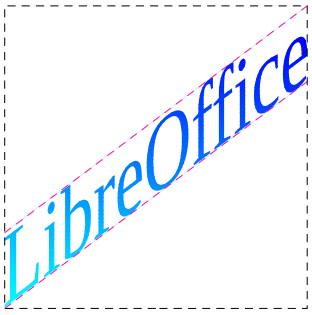
In contrast to shapes with one sub-path, the characters are not perpendicular to the sub-path, but each character is then transformed to fit between corresponding relative path-portions on the two sub-paths. For text with multiple paragraphs the relative vertical position in the bounding box of the text block is relevant too.

Fontwork shapes with two sub-paths still react on attribute `draw:text-path-scale` as described above. The length of the mean of the two sub-path lengths is used as reference. The preset Fontwork shapes with two sub-paths have the value '`path`' for this attribute.

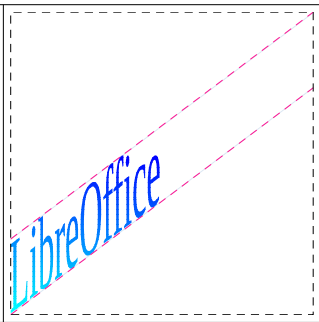
Table 14: Examples for text rendering with two sub-paths

A dashed black line indicates the shape size, pink dashed lines indicate the sub-paths.

Markup	Rendered Shape

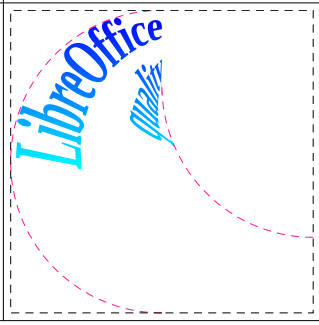
`<draw:enhanced-geometry svg:viewBox="0 0 60 60"`
`  draw:text-path="true"`
`  draw:text-path-mode="path"`
`  **draw:text-path-scale="path"**`
`  draw:type="non-primitive"`
`  draw:enhanced-path="M 0 45 L 60 0`
`  M 0 60 L 60 15"/>`
	

```
<draw:enhanced-geometry svg:viewBox="0 0 60 60"
    draw:text-path="true" draw:text-path-mode="path"
    draw:text-path-scale="shape"
    draw:type="non-primitive"
    draw:enhanced-path="M 0 45 L 60 0
                        M 0 60 L 60 15"/>
```




```
<draw:enhanced-geometry svg:viewBox="0 0 60 60"
    draw:text-path="true" draw:text-path-mode="path"
    draw:text-path-scale="path"
    draw:type="non-primitive"
    draw:enhanced-path="M 30 60 X 0 30 30 0
                        M 60 45 X 30 15"/>
```


```
<draw:enhanced-geometry svg:viewBox="0 0 60 60"
    draw:text-path="true" draw:text-path-mode="path"
    draw:text-path-scale="shape"
    draw:type="non-primitive"
    draw:enhanced-path="M 30 60 X 0 30 30 0
    M 60 45 X 30 15"/>
```



The common part of the markup of the following examples is:
`<draw:enhanced-geometry svg:viewBox="0 0 8 6" draw:text-path="true" draw:text-path-scale="shape" draw:text-path-mode="path" draw:type="non-primitive" draw:enhanced-path="` *see below* `"/>`

Table 15: Examples for text rendering depending on path direction and order

value of `draw:enhanced-path`	rendering
`M 0 0 L 8 0 M 0 6 L 8 6` top and bottom line left to right	LibreOffice Community
`M 0 6 L 8 6 M 0 0 L 8 0` top and bottom line left to right, but first line at bottom, second at top The text block is vertically mirrored.	Community LibreOffice
`M 8 0 L 0 0 M 8 6 L 0 6` top and bottom line right to left The text block is horizontally mirrored.	LibreOffice Community

Obviously, direction and order of the two paths have to be considered.

Note that the filling is applied independent from rendering the characters.

The predefined Fontwork shapes provide already a lot of different forms. You likely need your own Fontwork shapes only to use `draw:text-path-scale="shape"` or if you want to use a sub-path, which creates some mirroring.

Exercise 9.3

Create an ODF conform version of a ‘Ring’. Such shapes are provided by OOXML. Figure 106 shows a ‘Ring’ shape made in PowerPoint.

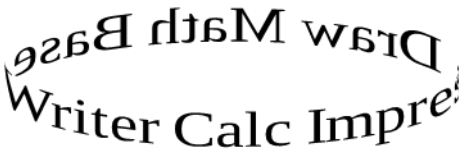


Figure 106: 'Ring' shape made in PowerPoint

You need two ellipses of same size, where one is shifted vertically. The solution will show a very simple version without handles. Of course you could apply all your knowledge from the previous chapters and create a shape with handle for the start of the text and for the height of the ellipses.

→solution

9.5 More Than Two Sub-Paths

If the number of sub-paths is odd, the rendering is similar to the case of one sub-path. If the number of sub-paths is even, each two consecutive sub-paths form a pair and rendering is similar to the case of two sub-paths. The black dashed lines indicate the shape size and the red dashed lines indicate the sub-paths in the examples.

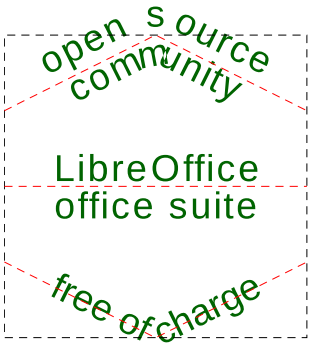


Figure 107: Odd number of sub-paths

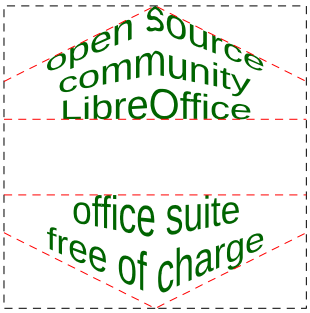


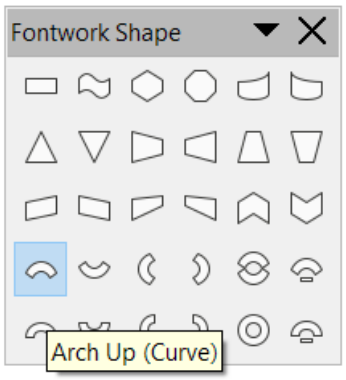
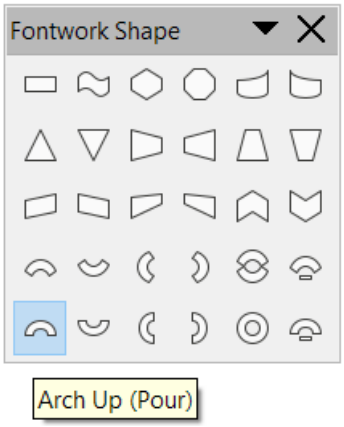
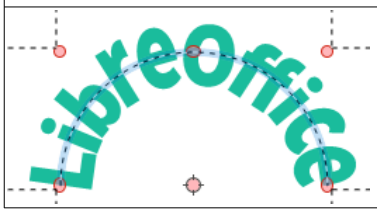
Figure 108: Even number of sub-paths

The paragraphs of the text are evenly distributed over the sub-paths. In figure 107 the five paragraphs are distributed over the three sub-paths as 12-34-5. A distribution 123-4-5 or 12-3-45 is not possible. In figure 108 the 4 sub-paths are combined to 2 pairs and the 5 paragraphs are then distributed as 123-45. Note that there is no text between the second and third sub-path.

It is not possible to mix stretchy rendering between a pair of sub-paths with rendering of text on sub-paths.

The ‘Fontwork Shape’ tool provides shapes with odd number of sub-paths in the last but one row. Their names contain the suffix “(Curve)”. The last row has the same shape names, now with suffix “(Pour)”. These shapes have an even number of sub-paths.

Table 16: 'Arch Up' shape in variants 'Curve' and 'Pour'

‘Arch Up’ shape with one sub-path	‘Arch Up’ shape with two sub-paths
	
	

9.6 Controlling Text Position

The position of the text can be influenced by the style attributes `draw:textarea-horizontal-align` and `draw:textarea-vertical-align`. They are available as 'Anchor' in the dialog 'Text Attributes' in the user interface. Thereby the value `justify` – 'Full Width' in the UI – is not evaluated at all. The horizontal anchor positions are mirrored to the toolbar 'Fontwork Alignment' in the 'Fontwork' toolbar. Changing the horizontal alignment there has no influence on the vertical alignment.

If the attribute `text-path-scale` has the value `path`, then you will see anchor differences for the horizontal anchor only if the text in a sub-paths pair has at least two paragraphs with different length. The vertical anchor position is ignored because the text is fit between pairs of two sub-paths.

If the attribute `text-path-scale` has the value `shape`, the vertical anchor positions `top`, `middle` and `bottom` are evaluated.

Remember that these settings belong to the style of the shape. There are no attributes in the enhanced geometry to determine the position of Fontwork text. In addition, the attributes `draw:text-areas` and `draw:text-rotate-angle` are not evaluated for Fontwork shapes, and you cannot tweak the position by adding leading or trailing spaces or by setting an indent at a paragraph.

You can only act on the position by specifying the enhanced path itself as needed.

9.7 Exercise solutions

9.7.1 Exercise 9.1

```
<draw:enhanced-geometry svg:viewBox="0 0 34 21" draw:type="non-primitive"
    draw:text-path="true" draw:text-path-mode="path" draw:text-path-
scale="shape"
    draw:enhanced-path=
        "M 0 21 Y 21 0 34 13 26 21 21 16 24 13 26 15 25 16 24 15"/>
```

The path of the Fibonacci spiral is drawn from the outside in.

Figure 109: Text on Fibonacci spiral

9.7.2 Exercise 9.2

```
<draw:enhanced-geometry svg:viewBox="0 0 4000 4000"
      draw:type="non-primitive"
      draw:text-path="true" draw:text-path-mode="path"
      draw:text-path-scale="path"
```

```
      draw:modifiers="113 49"
      draw:enhanced-path="U 2000 2000 2000 2000 $0 $1 F N">
  <draw:handle draw:handle-position="2000 $0"
               draw:handle-polar="2000 2000"/>
  <draw:handle draw:handle-position="2000 $1"
               draw:handle-polar="2000 2000"/>
</draw:enhanced-geometry>
```

The size of the characters is adapted such that the text fills the entire path. Drag the handle in the shape[42] to see this feature.

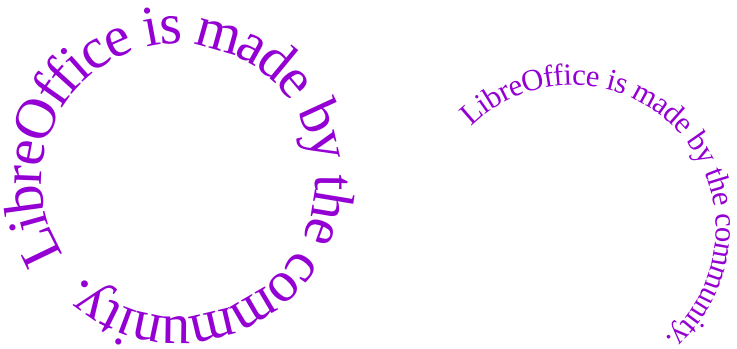


Figure 110: Text on arc with handle and scale 'path'

9.7.3 Exercise 9.3

```
<draw:enhanced-geometry svg:viewBox="0 0 6000 2000"
    draw:type="non-primitive" draw:text-path="true"
    draw:text-path-mode="path" draw:text-path-scale="path"
    draw:enhanced-path="B 0 0 6000 1200 0 600 6000 600
                        A 0 0 6000 1200 6000 600 0 600
                        B 0 800 6000 2000 0 1400 6000 1400
                        A 0 800 6000 2000 6000 1400 0 1400"/>
```

To get a full ellipse, draw two arcs. To get the lower part of the text left-to-right and the higher part of the text mirrored, right to left, the arcs are drawn counter-clockwise. Therefore, the commands A and B are used.

Both examples have nominal font size 18pt. In case of `scale="path"` the letters are enlarged so that the text fills the whole ellipse. In both cases the letters are stretched vertically to fit between the two ellipses.

Figure 111: font size increased to fit the ellipse

`scale="path"`

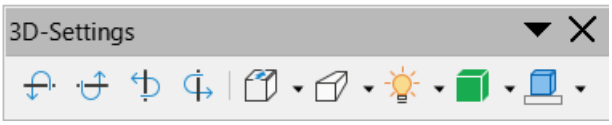
Figure 112: only increased height

`scale="shape"`

42 Shapes a contained in the separate file “Shapes from Custom Shape Tutorial.odg”.

10 Custom Shapes in 3D

The command ‘Toggle Extrusion’ brings a custom shape into 3D-mode or back. The shape is extruded perpendicular to the shape to generate a solid. The same engine is used internally for rendering such extruded shapes as for “real” 3D-scenes. However, the ‘3D Effects’ dialog is not usable with an extruded custom shape. Instead, the ‘3D-Settings’ toolbar is used to manipulate 3D properties via the user interface. The toolbar provides a simplified way to specify properties, but does not provide all possible settings of the document format.



ODF specifies 24 attributes in the `<draw:enhanced-geometry>` element for custom-shapes, that are related to the 3D-mode. Besides the attributes `dr3d:projection` and `dr3d:shade-mode` all of them have a naming structure `draw:extrusion-*`. The ODF specification was inspired by similar shape modes in the binary file format of MS Office[43], by the attributes of the extrusion element in VML[44] and 3D-shapes in Rich Text Format[45].

The properties are usable in macros, with service EnhancedCustomShapeExtrusion.[46] To examine the current, non default properties applied, the ‘Developer Tools’ are helpful. The extrusion properties are located in item ‘Extrusion’ in property ‘CustomShapeGeometry’.

This chapter will discuss these attributes. You already know how to edit the document source, therefore this chapter has no further exercises, but explains these attributes.

10.1 Enable 3D-mode

10.1.1 Attribute `draw:extrusion`

The attribute value `"true"` switches 3D-mode on and value `"false"` switches it off. The attribute is available as command ‘Toggle Extrusion’ in the user interface, so there is no need to edit it in file format.

10.1.2 Attribute `draw:extrusion-allowed`

With the attribute `draw:extrusion-allowed` a document producer can specify, that the application shall or shall not provide tools for extruding a shape. This attribute is not implemented in LibreOffice. All custom shapes – which includes Fontwork shapes – can be extruded in LibreOffice.

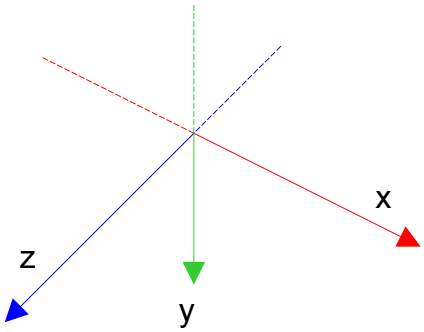
43 see chapters ‘2.3.15 3D Object’ and ‘2.3.16 3D Style’ in [MS-ODRAW] ‘Office Drawing Binary File Format’, available from https://docs.microsoft.com/en-us/openspecs/office_file_formats

44 See https://www.w3.org/TR/NOTE-VML and chapter ‘VML Extrusion Element’ in https://docs.microsoft.com/en-us/windows/win32/vml/msdn-online-vml-extrusion-element

45 The specification is no longer directly available from microsoft.com. It is archieved for example in https://web.archive.org/web/20190708132914/http://www.kleinlercher.at/tools/Windows_Protocols/Word2007RTFSpec9.pdf

46 https://api.libreoffice.org/docs/idl/ref/servicecom_1_1sun_1_1star_1_1drawing_1_1EnhancedCustomShapeExtrusion.html

10.2 Coordinate System



The x-axis points right and the y-axis points down in the internal 2D coordinate-system of custom shapes. That is the same for the 3D coordinate-system of an extruded custom shape. The z-axis is perpendicular to the xy-plane and points towards the observer. So extruded custom shapes use a left-handed coordinate-system. That is different from “true” 3D objects, which use a right-handed coordinate-system.

The origin of the 3D coordinate-system is the center of the snap rectangle of the shape. But for x- or y-coordinates you will not use absolute values. The attributes are defined to use values, which are somehow relative to the shape.

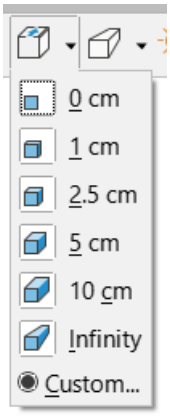
10.3 Text

Custom shapes can have text. Those text is currently not included in the 3D-scene of the extruded shape but is drawn on top of the projection of the shape.

If the shape has set the attribute `draw:text-path="true"` (see chapter 9, Text Path) you get a ‘Fontwork’ shape. In that case the text itself is the outline of the shape and reacts on extrusion.

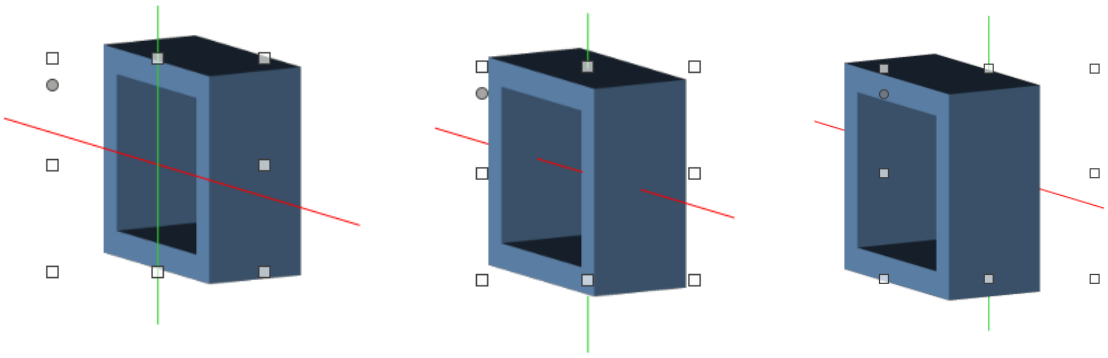
10.4 Object Geometry

10.4.1 Attribute `draw:extrusion-depth`



The attribute value has two components. The first component specifies the extrusion depth. It is a number with unit. This component is available as drop-down list ‘Depth’ in the ‘3D-settings’ toolbar. The second component is a number in the range [0;1], which is interpreted as fraction of the depth. It specifies the amount of the extrusion that lies before a shape. This second component has no user interface. The default value is `"36pt 0"`.

Value 0 for the second component means that the shape is the front face of the solid, with value 1 it is the back face of the solid. The impact of the second component becomes visible, when the object is rotated. Figure 113 shows objects with second component value 0, 0.5 and 1 respectively. The objects have same rotation angles and same rotation center. Note the location of the solid relative to the snap rectangle of the shape.



position 0 position 0.5 position 1

Figure 113: Rotated shape with different extrusion positions

10.4.2 Attribute `draw:extrusion-rotation-angle`

The attribute value has two components. The first component specifies the angle for the rotation around the x-axis, the second component the angle for the rotation around the y-axis. The angles may have the unit `deg`, `grad`, or `rad`. If no unit is used, the value is interpreted as angle in degrees. The default value is `"0 0"`.

The user interface provides four buttons to rotate the solid in 5 deg steps.

“Tilt Down” Rotation around a horizontal axis. Parts of the solid above the axis come nearer to you. The rotation angle decreases.

“Tilt Up” Rotation around a horizontal axis. Parts of the solid below the axis come nearer to you. The rotation angle increases.

“Tilt Left” Rotation around a vertical axis. Parts of the solid right from the axis come nearer to you. The rotation angle increases.

“Tilt Right” Rotation around a vertical axis. Parts of the solid left from the axis come nearer to you. The rotation angle decreases.

The user interface has no limit for rotation. You can end up with rotation angles larger than 360 deg or smaller than −360 deg. Editing the document source might be useful to correct such angles.

The rotation on the z-axis is not specified as extrusion property but by the `draw:transform` attribute of the `<draw:custom-shape>` element. The corresponding user interface is the ‘Rotation’ section in the ‘Position and Size’ dialog.

The rotations are applied in the order first z-axis, then vertical axis (y-axis), last horizontal axis (x-axis).

10.4.3 Attribute `draw:extrusion-rotation-center`

The attribute `draw:extrusion-rotation-center` specifies the center of rotation, the intersection point of the rotation axes. The user interface has no tool for changing its values. The default

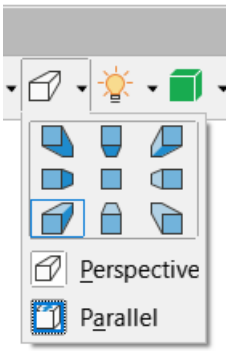
position of the rotation center is the center of the shape. That need not be the front face of the solid, see attribute `draw:extrusion-depth`.

The attribute value is written in the form "`(dx dy dz)`", for example "`(0.5 -0.5 1000)`". The first and second component are fractions of the shape size. The third component is interpreted as hundredth millimeter. The shape center as default rotation center would be written as value "`(0 0 0)`".

These values are relative to the default rotation center. A positive value for *dx* moves the center to the right, a positive value for *dy* moves it down, and a positive value for *dz* moves it towards the observer. The value "`(0.5 -0.5 1000)`" will set the rotation center to the right-top of the shape, 1 cm towards the observer, for example. The position of the rotation center effects the position of the actual rendered drawing.

Old versions of LibreOffice have this wrongly implemented. You need at least version 7.3 to use other than the default value of this attribute.

10.5 Projection



To render the three-dimensional solid to the two-dimensional paper or screen either parallel projection or central projection is used. For extruded custom shapes the projection plane (image plane, film plane) is always the xy-plane.

A custom shape has a defined two-dimensional size, which is indicated by the resize handles. The result of the projection is not fit into this rectangle but might exceed it. That is different to “true” 3D-objects.

10.5.1 Attribute `dr3d:projection`

The attribute `dr3d:projection` specifies which projection kind to use. It has the value `parallel` for parallel projection and the value `perspective` for central projection. The value `parallel` is the default value.

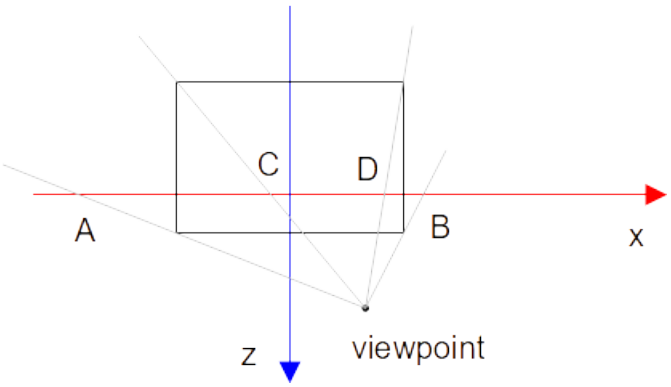
You find this setting in the user interface in the lower part of the drop-down ‘Direction’ in the ‘3D-Settings’ toolbar.

When you change the projection type, LibreOffice remembers related values you have set and will also write them to file. This allows you to easily switch between projection types.

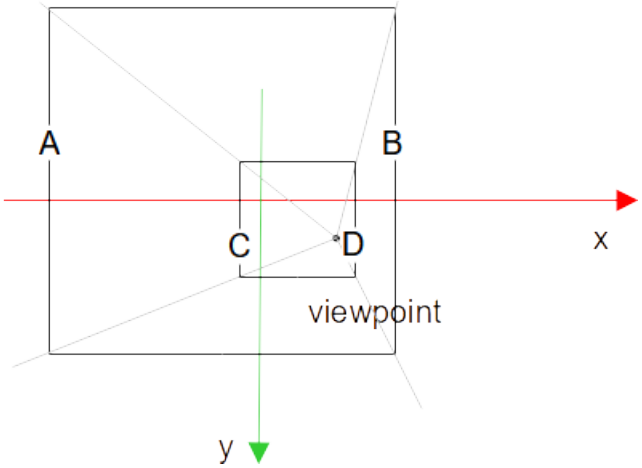
10.5.2 Central Projection

A central projection is determined by a viewpoint (eye-point), that is the position of the observer. A straight line is drawn through viewpoint and object point. The intersection of this line with the xy-plane is the result of the projection.

The illustration shows an extruded square in a view from top to bottom. The square is extruded so that ¼ of the depth lies before the shape.



Looking at the same extruded shape from front to back, you see the known effect, that nearer parts are larger than parts far away. And you see that the viewpoint becomes the so called ‘vanishing point’.



10.5.3 Attributes `draw:extrusion-viewpoint` and `draw:extrusion-origin`

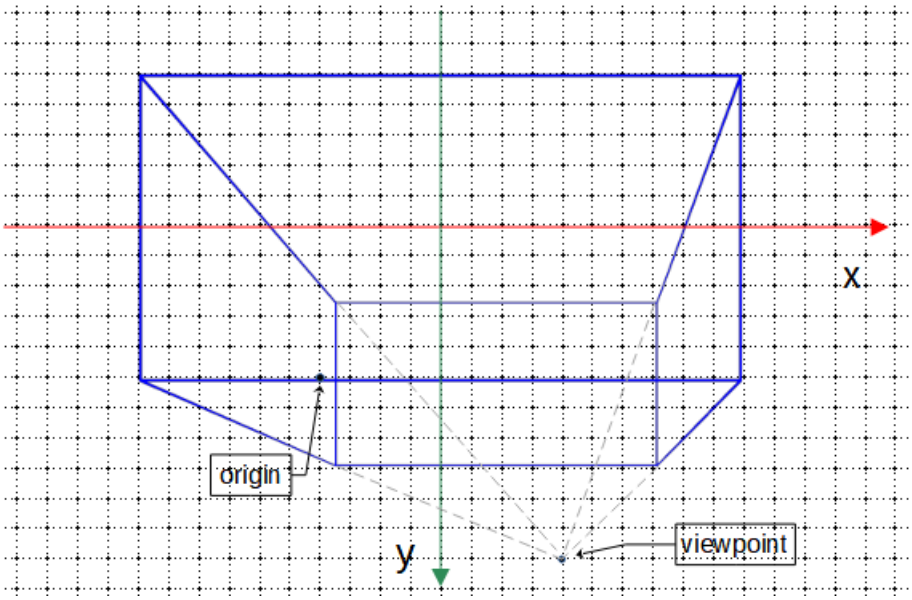
The viewpoint is specified by the attributes `draw:extrusion-viewpoint` and `draw:extrusion-origin` in the document source. These attributes are always used together. They are specific to central projection and ignored if the projection type is set to value `"parallel"`.

The value of attribute `draw:extrusion-viewpoint` has the form `"(vx vy vz)"` where each of `vx`, `vy` and `vz` is a length consisting of a number immediately followed by a length unit. The default value is `"(3.5cm -3.5cm 25cm)"`. But LibreOffice uses `"(-3.472cm 3.472cm 25cm)"` as default, which is the default value in MS Office file formats. The attribute is not available in the user interface.

The component `vz` determines the z-coordinate of the viewpoint directly. The components `vx` and `vy` are not coordinates in the coordinate system, but they describe a shift from the viewpoint origin. The viewpoint origin is specified by the attribute `draw:extrusion-origin`. Its value has the form `"dx dy"`. The components `dx` and `dy` are fractions of width and height of the *bounding* rectangle of the 2D shape with applied 2D transformations. Values are in the range `[-0.5, 0.5]`. The center of the bounding rectangle is given by the value `"0 0"`. The default value is `"0.5 -0.5"`, the right-top corner of the bounding rectangle.

Example: Take a shape ‘rectangle’ with width 10 cm and height 5 cm, without any transformations. With attributes

`draw:extrusion-origin="-0.2 0.5"` and
`draw:extrusion-viewpoint="(4cm, 3cm, 10cm)"`
you get the absolute coordinates of the viewpoint as
x-coordinate = −0.2 · 10 cm + 4 cm = 2 cm and y-coordinate = 0.5 ·5 cm + 3 cm = 5.5 cm



LibreOffice has no tool in the user interface to set arbitrary values. It only offers nine presets in the upper part of the drop-down ‘Direction’ in the ‘3D-Settings’ toolbar.

North-East: `draw:extrusion-viewpoint="(-3.472cm 3.472cm 25cm)"` `draw:extrusion-origin="-0.5 0.5"`

East: `draw:extrusion-viewpoint="(-3.472cm 0cm 25cm)"` `draw:extrusion-origin="-0.5 0"`

South-East: `draw:extrusion-viewpoint="(-3.472cm -3.472cm 25cm)"` `draw:extrusion-origin="-0.5 -0.5"`

South: `draw:extrusion-viewpoint="(0cm -3.472cm 25cm)"` `draw:extrusion-origin="0 -0.5"`

South-West: `draw:extrusion-viewpoint="(3.472cm -3.472cm 25cm)"` `draw:extrusion-origin="0.5 -0.5"`

West: `draw:extrusion-viewpoint="(3.472cm 0cm 25cm)"` `draw:extrusion-origin="0.5 0"`

North-West: `draw:extrusion-viewpoint="(3.472cm 3.472cm 25cm)"` `draw:extrusion-origin="0.5 0.5"`

North: `draw:extrusion-viewpoint="(0cm 3.472cm 25cm)"` `draw:extrusion-origin="0 0.5"`

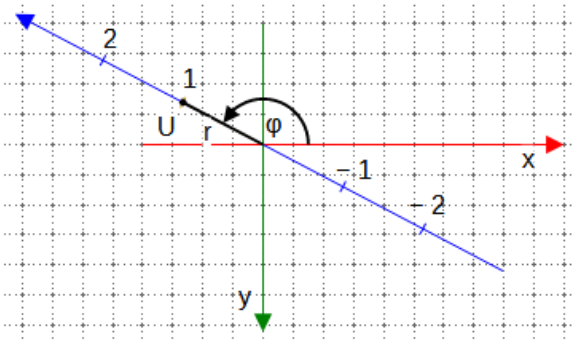
Backwards : `draw:extrusion-viewpoint="(0cm 0cm 25cm)"`
`draw:extrusion-origin="0 0"`

Note that the bounding rectangle might be different from the snap rectangle. Both shapes in the following illustration have set `draw:extrusion-viewpoint="(0cm 0cm 10cm)"` `draw:extrusion-origin="0 0"` and `svg:width="12cm", svg:height="8cm"` and `draw:extrusion-depth="25mm 0"`. But the bounding boxes are different and therefore the positions of the viewpoint origin are different.

Old versions of LibreOffice have not used the bounding rectangle but the snap rectangle. To get the correct rendering as specified in ODF and as used by MS Office, you need to use at least version 7.3.

10.5.4 Parallel Projection

In general, a parallel projection can be determined by a projection direction. To calculate the image of a point, we need to know the image point U of the unit point (0|0|1) of the z-axis. ODF uses the polar coordinates (r , φ) of this point. Section 11.3 contains, how to create them from known projection direction.



The angle φ is positive in the shown direction.

10.5.5 Attribute draw:extrusion-skew

The attribute `draw:extrusion-skew` specifies the skew amount (redaction factor) and the skew angle. Its value has the form `"amount angle"`. The `amount` is given as percentage without % sign of the z-coordinate of a point. With above notations it is 100 r. The number of component `angle` is

the above notated angle φ converted to unit degree. The value of the angle is used modulo 360°, so value `-30` has the same result as value `330`. The default value is `"50 45"` in ODF.

The `draw:extrusion-skew` attribute is only used for parallel projection and is ignored in case of central projection.

The buttons in the upper part of the drop-down ‘Direction’ of the ‘3D-Settings’ toolbar work too for parallel projection. The value for the `angle` component is set as indicated in following table:

North-West `135`	North `90`	North-East `45`
West `180`	Backwards `0`	East `-360`
South-West `-135`	South `-90`	South-East `-45`

All buttons but ‘Backwards’ set the `amount` component to `50`.

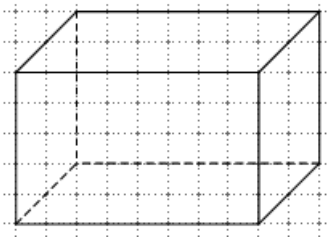
The button ‘Backwards’ is special because it produces an orthogonal projection, whereas the other buttons produce oblique projections. The ‘Backwards’ button sets the `amount` component to `0`.

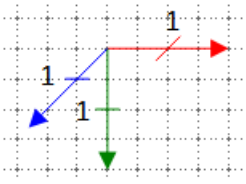
LibreOffice writes the value `-360` for the direction ‘East’, which is basically the same as value `0`. But it indicates that the user has set ‘East’ and not ‘Backwards’.

LibreOffice uses internal `-135` (‘South-West’) as default skew angle, because MS Office formats use that value as default. So you need to make sure, that the value of `draw:extrusion-skew` is not omitted in document source but always written. To force this when using the ‘3D-Settings’ toolbar switch from the default direction ‘South-West’ to a different direction and then back, in case ‘South-West’ is intended.

10.5.6 Examples of oblique parallel projection

Oblique parallel projections are often called “cavalier projection” or “cabinet projection”. However, these terms are not standardized and depend on country. Surely you have used such projections in school mathematics to sketch a cuboid on squared paper.

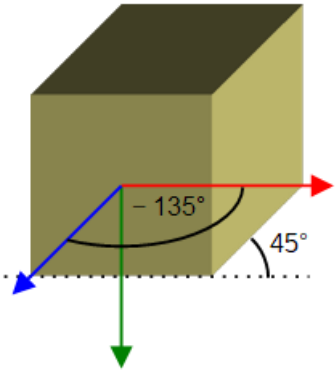




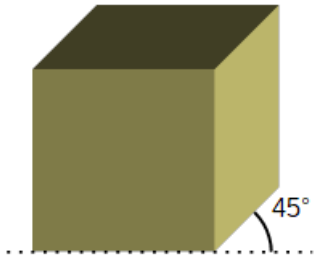
To use a coordinate system as shown in the illustration, you need to set `draw:skew="70.71 -135"`. Using a skew angle `-135` results in a projection, where those lines, which go from the observer perpendicular to the xy-plane are drawn with a 45° angle to the x-axis. The reduction factor of $70.71 \triangleq \sqrt{0.5} \approx 0.7071$ corresponds to the length of the diagonals in a grid square.

The illustrations show a cube with edge length 3 cm and extrusion behind the front face.

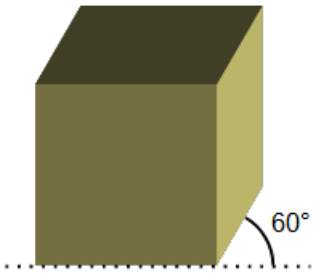
`draw:skew="70.71 -135"`



`draw:skew="50 -135"`



`draw:skew="50 -120"`



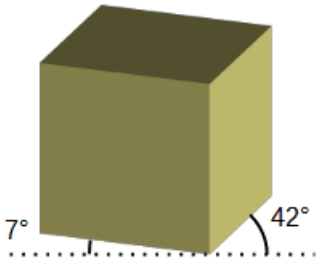
10.5.7 Examples for orthographic projection

In an “orthographic”, also called “orthogonal”, parallel projection, the projection direction is perpendicular to the xy-plane. Two of such projections are standardized in ISO 5456-3, “isometric projection” and “dimetric” projection.

Because the direction is perpendicular to the xy-plane, you must set `draw:skew="0 0"` (‘Backwards’). But to get “isometric” or “dimetric”, you need to rotate the solid in addition.

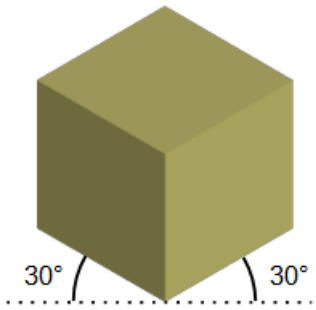
To get a nearly “dimetric” projection, rotate the solid by 20° about the y-axis and by 20° about the x-axis. You can do that in the ‘3D-Settings’ toolbar by 4 times ‘Tilt Left’ and 4 times ‘Tilt Down’. That produces about 7° and 42° angles of receding lines against a horizontal line. Some “Geodreieck”[47] tools have marks for 7° and 42°. The illustration shows again a cube. The markup for this cube is

```
draw:extrusion-skew="0 0"
draw:extrusion-rotation-angle="-20 20"
```



The “isometric” projection needs a rotation of 45° about the y-axis and a rotation about the x-axis of about 35°. The cube in the illustration has the markup

```
draw:extrusion-skew="0 0"
draw:extrusion-rotation-angle="-35 45"
```



If you want the angles more precise, you need to edit the document source. You can use for “dimetric” projection

```
draw:extrusion-skew="0 0"
draw:extrusion-rotation-angle="-19.4205368891 20.2680671953"
```

and for “isometric” projection

```
draw:extrusion-skew="0 0"
draw:extrusion-rotation-angle="-35.2643896828 45"
```

47 https://en.wikipedia.org/wiki/Set_square#Geodreieck [last called 2022-03-03]

The calculations for these angles are shown in section 11.4.

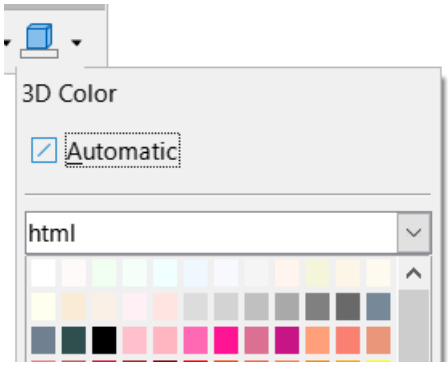
10.6 Object Surface

Rendering object surface and lighting of extruded shapes have changed significantly in LibreOffice 7.4. The descriptions here are of limited use for earlier versions.

The surface of an object is determined by the fill of front face and back face and by the color of the extruded facets. The fill of front and back face is specified as the fill of the two-dimensional shape. The color of the extruded facets can be specified explicitly or is determined automatically by LibreOffice.

Another aspect is how the surface of an object reacts on the lights of the 3D-scene.

10.6.1 Attribute `draw:extrusion-color`



This attribute is available in the ‘3D Color’ part of the ‘3D-Settings’ toolbar. No need to edit the document source for that.

The attribute value `false` means, that no explicit extrusion color is used. That is setting ‘Automatic’ in the ‘3D Color’ tool. In case the shape has a color fill, LibreOffice uses this color for the extrusion facets too. In other cases, LibreOffice sets a color following some internal rules. Other applications will have own rules. For interoperability you should set the color explicitly in these cases.

The attribute value `true` means, that the color is used, that is defined in the style attribute `draw:secondary-fill-color`. That color can be set by the common tool for color choosing in the lower part of the ‘3D Color’ tool. When you chose a color there, then attribute `draw:extrusion-color="true"` is automatically set in the `<draw:enhanced-geometry>` element and the attribute `draw:secondary-fill-color` is generated in the associated `<style:style>` element. So there is no need to edit the document source for this attribute.

10.6.2 Attribute `draw:extrusion-light-face`

This attribute toggles whether the front face of the extruded shape reacts on lights. This attribute is not implemented in LibreOffice. In LibreOffice a front face always reacts on lights.

10.6.3 Attribute `draw:extrusion-diffusion`

This attribute specifies how much of diffuse light is reflected by the shape. To use this attribute, you need at least version 7.4. ODF allows values from 0% to 100% with default value 0%. The value 0% is not interpreted by LibreOffice, because existing shapes in documents created with older versions of LibreOffie have not set the attribute, but act like having 100% set. A value smaller than 100% reduces the brightness of the rendered object. To force writing the attribute to document source switch extrusion off and then back to on.

This attribute acts only on solid colors. Filling with gradient, bitmap, hatch or pattern is not affected.

Diffusion 100% Diffusion 50%

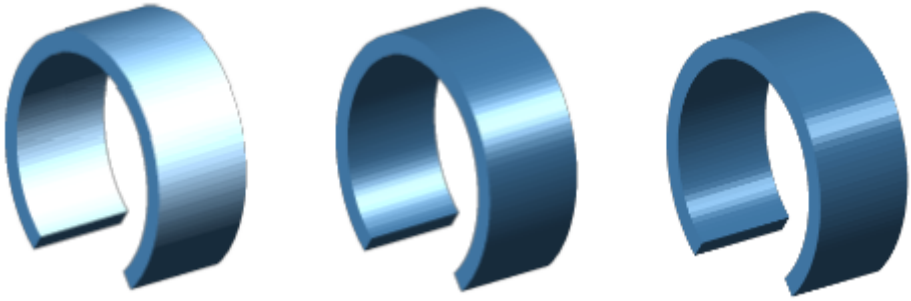
Figure 114: Diffusion

The ‘Metal MS compatible’ surface preset uses about 67% for the `diffusion` attribute. Switching to ‘Matte’, ‘Plastic’ or ‘Metal ODF’ surface preset then sets the `diffusion` attribute to 100%.

10.6.4 Attribute `draw:extrusion-shininess`

This attribute specifies how much the shape material behaves like a mirror with respect to lights. To use this attribute, you need at least version 7.4. Older versions interpret the attribute incorrectly.

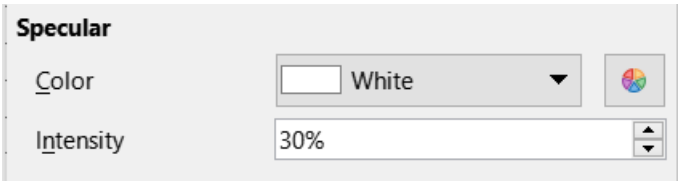
This attribute is similar to the setting ‘Specular Intensity’ in the ‘Material’ tab of the ‘3D Effects’ dialog of “true” 3D-scenes. ODF allows values from 0% to 100%, but values below 3% result in strange rendering and values larger than 66% will not increase the effect. In most cases there is no need to change the default value of 50%.



Shininess 15% Shininess 50% Shininess 66%

Figure 115: Shininess

10.6.5 Attribute `draw:extrusion-specularity`



To use this attribute, you need at least version 7.4. Older versions interpret this attribute incorrectly.

The description of the attribute is very vague in the standard. LibreOffice uses it similar to the attribute ‘Specular Color’ in the ‘Material’ tab of the ‘3D Effects’ dialog of a “true” 3D-scenes.

The value is a percent value. With value `0%` the first light source is no longer treated as specular light, but as a directional, but diffuse light. That results in a matte surface of the rendered shape. In LibreOffice, specularity is evaluated only for the first light.

ODF 1.3 allows only values in range [0%, 100%]. But when importing a document from MS binary formats often a value of about 122% is used. Saving in ‘ODF 1.3 extended’ will keep this value, saving in ‘ODF 1.3’ will clamp it to 100%.

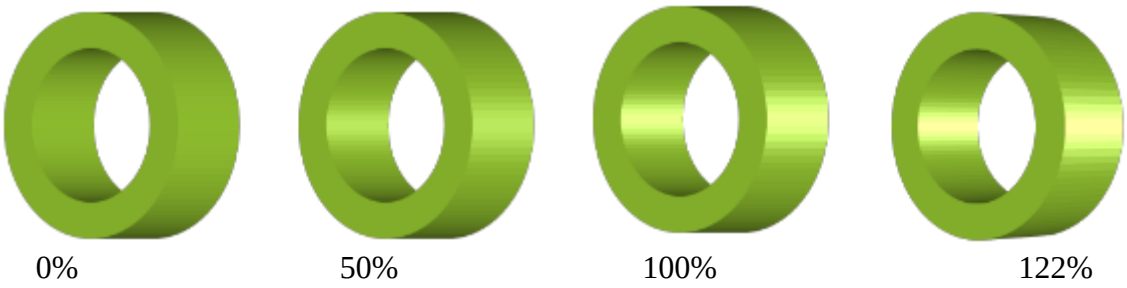


Figure 116: Several values of attribute `draw:extrusion-specularity`

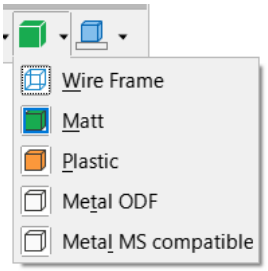
Note that only the extruded facets of the solid are glossy, front face and back face are always matte.

There is no general user interface for this attribute. The ‘Surface’ drop-down list in the ‘3D Settings’ toolbar set a value of `122%` when switching from ‘Matte’ surface to ‘Plastic’, ‘Metal ODF’ or ‘Metal MS compatible’, and it sets a value of `0%` when switching to ‘Matte’ surface. In other cases, an existing value is kept. So if you set an own value in the document source, it might be overwritten when you use the ‘Surface’ drop-down list, and you will need to set the value again.

10.6.6 Attributes `draw:extrusion-metal` and `loext:extrusion-metal-type`

The ODF standard specifies for values of the `draw:extrusion-metal` attribute:
“• `false`: the specular color for the shading of an extruded shape is white.
• `true`: the specular color for the shading of an extruded shape is gray (red green and blue values of 200) instead of white and 15% is added to the specularity.”[48]

To achieve better compatibility with Microsoft Office formats, LibreOffice introduces an additional `loext:extrusion-metal-type` attribute. Its possible values are `MetalODF` and `MetalMSCompatible`. With value `MetalODF`, rendering uses the way specified in ODF. With value `MetalMSCompatible`, rendering uses the shape fill or extrusion color respectively instead of Gray. That is similar to rendering in MS Office. This attribute cannot be saved in ‘ODF 1.3’ file format, you must use ‘ODF 1.3 extended’. A document saved in ‘ODF 1.3’ will always display a shape with `draw:extrusion-metal="true"` as if `loext:extrusion-metal-type="MetalODF"` was set.



You can set surface ‘Metal ODF’ and ‘Metal MS compatible’ in the ‘Surface’ drop-down list in the ‘3D-Settings’ toolbar. These two entries replace the ‘Metal’ entry in versions prior to 7.4.

The ‘Metal MS compatible’ surface preset additionally sets the value for `draw:diffusion` to about 67%. This darkens the shape similar to rendering in MS Office.

48 https://docs.oasis-open.org/office/OpenDocument/v1.3/os/part3-schema/OpenDocument-v1.3-os-part3-schema.html#attribute-draw_extrusion-metal

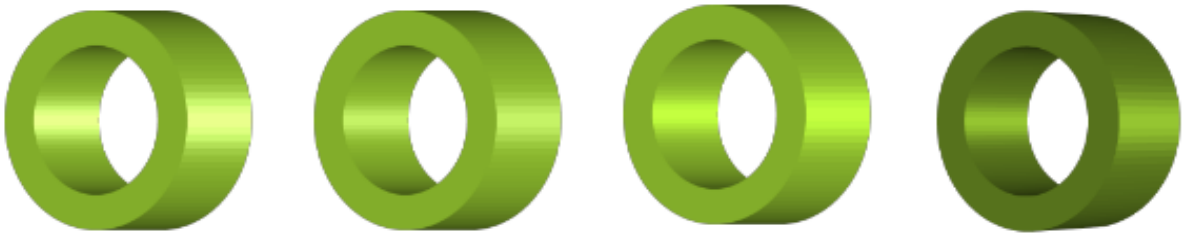


Figure 117: Rendering with and without surface Metal

The ‘Matt’ and ‘Plastic’ surface presets have no corresponding attribute in file format. The ‘Wire Frame’ preset is part of the `dr3d:shade-mode` attribute.

10.7 Rendering Quality

10.7.1 Attribute `dr3d:shade-mode`

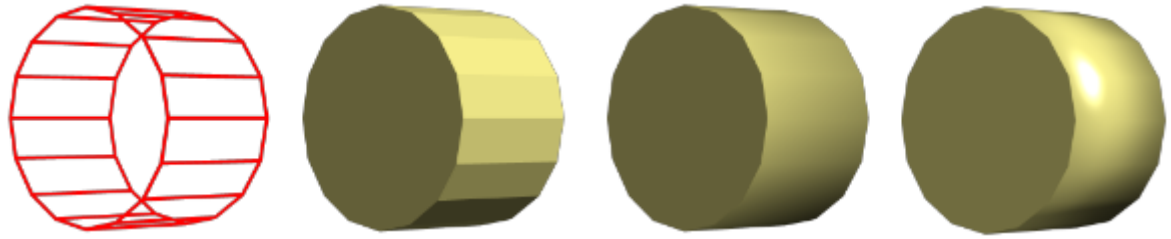
The `dr3d:shade-mode` attribute has the possible values `draft`, `flat`, `gouraud` and `phong`.

The `draft` value is set, when you select the ‘Wire Frame’ surface preset. In this case, no surface is drawn, but the otherwise invisible lines are drawn as solid lines in the thickness and color set as shape properties. For curves, lines are drawn between the ends of the line segments created to interpolate the curve.

The `flat` value is the default value. It is set, when you switch from ‘Wire Frame’ surface preset to any of the other surface presets or when a shape is newly created and extruded.

There is no user interface to set the values `gouraud` and `phong`. They require to calculate of interpolations for each point of an extruded face and therefore rendering is more expensive. If MS Office exports an extruded shape to ODF, it uses the `gouraud` value. The `gouraud` value is known as SMOOTH in the API. In `gouraud` mode, the faces are not rendered in a single color, but with gradients so that the color transitions between faces are smoother.

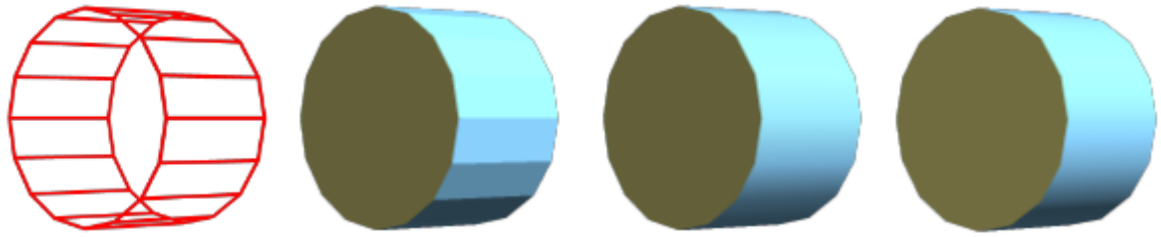
Figure 118 shows how the shape at the right side is rendered if it is extruded.



draft flat gouraud phong

Figure 118: shade mode variants with no extrusion color

If an extrusion color is set, then LibreOffice treats the extruded facets internally as a separate drawing object. The rendering is more symmetrical in this case.



draft flat gouraud phong

Figure 119: shade mode variants with blue extrusion color

10.7.2 Attribute `draw:extrusion-number-of-line-segments`

This attribute specifies the number of line segments, which have to be used to interpolate a curve. This attribute is not implemented in LibreOffice.

10.8 Scene Lights

Rendering object surface and lighting of extruded shapes have changed significantly in LibreOffice 7.4. The descriptions here are of limited use for earlier versions.

A 3D-scene of an extruded custom shape has three lights: one ambient light with no direction and two directional lights. Ambient light is reflected by every part of the surface of an object in the same extent, directional lights are only reflected with an amount that depends on the direction of the facets of the extruded shape in relation to the direction of the light. That is, the angle between the normal of the facet surface and direction of the light is taken into account. The most light is received when the reflected light is directed towards the observer.

10.8.1 Attributes `draw:extrusion-first-light-direction` and `draw:extrusion-second-light-direction`

Normally, the light rays of a directional light can be parallel or they start in a common point. The ODF format does not use the latter but has only parallel light rays.

The value of a direction attribute has the form `"(dx dy dz)"`. Note that the separator is a space, not a comma. The components are not absolute values, only the ratio between them is relevant.

Value "(5 0 1)" has the same meaning as value "(50000 0 10000)". The default direction for the first light source is "(5 0 1)", for the second light source "(-5 0 1)".

The upper part of the ‘Lighting’ tool in the ‘3D-Settings’ toolbar has nine buttons for setingt a preset. The images show the direction of the first light, but the direction of the second light is automatically set as well. The directions used are listed in the table below. The values are the same values as used by MS Office in RTF or binary format. If you have set own values in the document source, you will have to set them again when you change the lighting direction using the toolbar.

From	First Light	Second Light
top-left	(-50000 -50000 10000)	(50000 0 10000)
top	(0 -50000 10000)	(0 50000 10000)
top-right	(50000 -50000 10000)	(-50000 0 10000)
left	(-50000 0 10000)	(50000 0 10000)
center	(0 0 10000)	(0 0 10000)
right	(50000 0 10000)	(-50000 0 10000)
bottom-left	(-50000 50000 10000)	(50000 0 10000)
bottom	(0 50000 10000)	(0 -50000 10000)
bottom-right	(50000 50000 10000)	(-50000 0 10000)

The implementation in LibreOffice versions prior to 7.4 used the wrong, opposite sign for the z-coordinate. This causes the light to fall on the shape from behind. To get around the fact that the shapes are much too dark as a result, an additional light with rays parallel to the z-axis coming from the observer was introduced. Since the lighting was corrected in version 7.4, the lighting of the old shapes looks different in version 7.4.

10.8.2 Attribute `draw:extrusion-brightness`

The `draw:extrusion-brightness` attribute determines the amount of non-directional, ambient light. The values range from 0% to 100% with a default value of 33%. The ambient light is not colored light. LibreOffice multiplies the percent value with 255 and sets the result to all three channels of an RGB color, effectively setting a degree of Gray. Example: For percent value 33%, LibreOffice calculates the value $0.33 \cdot 255 = 84.15$, rounds it to 84 and sets the color RGB(84,84,84).

10.8.3 Attributes `draw:extrusion-first-light-level` and `draw:extrusion-second-light-level`

These attributes determine the amount of directional light. The values range from 0% to 100%. It is converted to a RGB color as described above. The default value is 66% for both light sources.

The ‘Bright’, ‘Normal’ and ‘Dim’ intensity presets in the ‘Lighting’ tool in the ‘3D-Settings’ toolbar set these values:

‘Bright’:
```
draw:extrusion-brightness="33%"
draw:extrusion-first-light-level="66%"
draw:extrusion-second-light-level="66%"
```

‘Normal’:
```
draw:extrusion-brightness="15%"
draw:extrusion-first-light-level="67%"
draw:extrusion-second-light-level="37%"
```

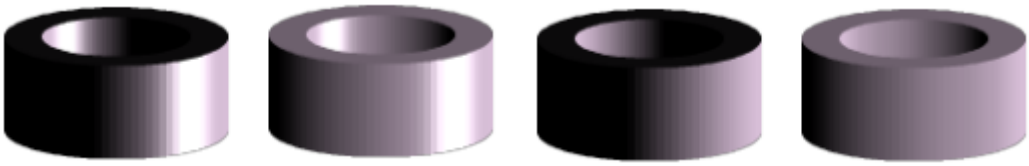
‘Dim’:
```
draw:extrusion-brightness="6%"
draw:extrusion-first-light-level="79%"
draw:extrusion-second-light-level="21%"
```

If you have set own values in the document source, you will have to set them again when you change the light intensity using the toolbar.

10.8.4 Attributes `draw:extrusion-first-light-harsh` and `draw:extrusion-second-light-harsh`

These attributes determine for the first and second light, respectively, whether it is a harsh or a soft light. The ODF specification leaves it up to implementations to determine how the light is softened. LibreOffice adds four light directions at a 60° angle to the original light direction and distributes some of the light amount to these additional lights. This satisfactorily emulates how MS Office uses a “soft” light. These additional lights are not included in the document source or shape properties, but are only used when rendering the shape. In default settings, the first light is always harsh. The second light is soft in ‘Bright’ and ‘Normal’ presets and harsh in ‘Dim’ preset.

All following examples use `draw:extrusion-brightness="0%"`, `draw:extrusion-second-light-level="0%"` and `draw:extrusion-first-light-level="100%"`.



harsh, specular soft, specular harsh, matte soft, matte

Figure 120: harsh vs. soft combined with specular vs. matte

Although a “soft” light is more diffuse than a “harsh” light, it still has a direction. That is different from ambient light, which has no direction.

11 Mathematics

11.1 Approximate Sine Curve by Cubic Bézier Curve.

In almost all cases it will not be possible to use a single Bézier curve for an entire function graph, but it will be a sequence of Bézier curves, each approximating a section of the function graph. So the more precise task is: Approximate a function graph over interval [a;b] by a Bézier curve.

1st requirement: The anchor points of the Bézier curve coincide with the points of the function graph at the edge of the interval.

general f(x)	sin(x) on [-π/2; π/2]	sin(x) on [0;π/2]
$P_0(a_0\|f(a_0))$	$P_0\left(-\frac{\pi}{2}\middle\|-1\right)$	$P_0(0\|0)$
$P_3(a_3\|f(a_3))$	$P_3\left(\frac{\pi}{2}\middle\|1\right)$	$P_3\left(\frac{\pi}{2}\middle\|1\right)$

2nd requirement: The straight line through the control point and the anchor point is a tangent to the function graph.
The tangent line at the point $(x_0\|f(x_0))$ generally has in the extression $t(x)=f'(x_0)\cdot(x-x_0)+f(x_0)$.

general f(x)	sin(x) on [-π/2; π/2]	sin(x) on [0;π/2]
$P_1(a_1\|f'(a_0)\cdot(a_1-a_0)+f(a_0))$	$P_1(c_1\|-1)$	$P_1(d_1\|d_1)$
$P_2(a_2\|f'(a_3)\cdot(a_2-a_3)+f(a_3))$	$P_2(c_2\|1)$	$P_2(d_2\|1)$

3rd requirement: In the sequence of Bézier curves, one curve will touch the next. The transition should be smooth. Therefore, the Bézier curves are created so that their curvature at points P_0 and P_3 is the same as for the function.

For a function, the curvature (“kappa”) is calculated as $\kappa_f=\frac{f''(x)}{(1+f'(x)^2)^{3/2}}$

For a Bézier curve it is $\kappa_B(t)=\frac{B_x'(t)\cdot B_y''(t)-B_x''(t)\cdot B_y'(t)}{((B_x'(t))^2+(B_y'(t))^2)^{3/2}}$

Therefore you have to solve $\kappa_B(0)=\kappa_f(a_0)\wedge\kappa_B(1)=\kappa_f(a_3)$. It is highly recommended to use a computer algebra system (CAS) for this.

In case “sin(x) on [-π/2; π/2]” we get the solutions $c_1=-\frac{\pi}{2}+\frac{2}{\sqrt{3}}$ and $c_2=\frac{\pi}{2}-\frac{2}{\sqrt{3}}=-c_1$.

A table of values shows that the maximum deviation of the y-values between Bézier curve and sine curve is less than 0.002344623573828.

In case “sin(x) on [0;π/2]” we get the solutions $d_1=\frac{3}{2}\pi-\frac{3}{8}\pi^2-\frac{1}{2}\approx 0.511287329976$ and $d_2=1$.

This results in a maximal deviation of less than 0.000580425227085. Even using the rounded values $P_1(0.5|0.5)$ only results a maximal deviation of less than 0.000748891084773.

In other context, other third requirements may be useful instead of curvature, such as considering average or maximal deviation.

There is a lot of literature on Bézier curves. For starters, "A Primer on Bézier Curves"[49] might be useful.

11.2 Mathematics for Tangents of Ellipses

The tangents through a point B outside the circle touch the circle at points S and R. The straight line through points R and S is the 'polar' to 'pole' B.

This chapter contains a method to calculate the points of contact S and R and angles associated with these points. Section 11.2.4 discusses how to solve a similar problem for an ellipse. If you search for the terms 'pole' and 'polar' you will find some more methods.

The following calculations and figures show the case where point B lies below the center of the circle. Formulas for positions above the center can be derived by mirroring.

The formulas already take into account the coordinates of the center of the circle. Instead, you can translate the drawing so that the center is at the origin, and first determine the solution for it. Afterwards you move it back.

11.2.1 Tangents to the Circle in General

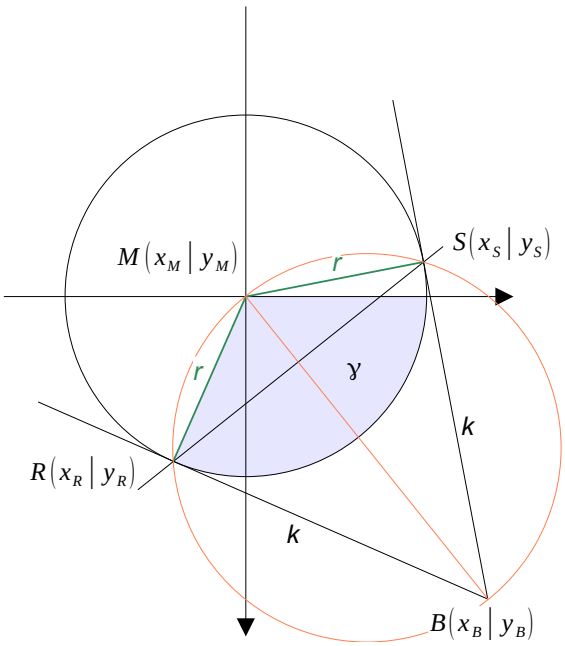


Figure 121: Tangents to the Circle

The points S and R belong to the circle with center M, which gives the equation

$$(x-x_M)^2+(y-y_M)^2=r^2$$

The triangles MSB and MBR are right triangles and so we get the equation

49 https://pomax.github.io/bezierinfo/ [called 2021-09-13]

$|MB|^2=(x_B-x_M)^2+(y_B-y_M)^2$ and then $k=|SB|=|RB|=\sqrt{|MB|^2-r^2}=\sqrt{(x_B-x_M)^2+(y_B-y_M)^2-r^2}$.

Since the point B is outside the circle, the radicand is never negative.

Depending on the task, you may need the Cartesian coordinates $(x_R \mid y_R)$ of a point of contact or the associated angle γ or both. Calculating one from the other is the well-known problem of converting between Cartesian and polar coordinates. If the angle γ is known, the Cartesian coordinates are $(r\cdot\cos(\gamma)+x_M \mid r\cdot\sin(\gamma)+y_M)$ and from known coordinates $(x_R \mid y_R)$ the angle can be calculated by $\gamma=\operatorname{atan2}(y_R-y_M\ ,\ x_R-x_M)$.

11.2.2 Coordinates for Points of Contact

The coordinates of points S and R are (using a CAS)

$$x_S=\frac{r}{|MB|^2}\cdot\left(r\cdot(x_B-x_M)+(y_B-y_M)\cdot k\right)+x_M$$

$$y_S=\frac{r}{|MB|^2}\cdot\left(r\cdot(y_B-y_M)-(x_B-x_M)\cdot k\right)+y_M$$

$$x_R=\frac{r}{|MB|^2}\cdot\left(r\cdot(x_B-x_M)-(y_B-y_M)\cdot k\right)+x_M$$

$$y_R=\frac{r}{|MB|^2}\cdot\left(r\cdot(y_B-y_M)+(x_B-x_M)\cdot k\right)+y_M$$

The formulas are unwieldy in the general case. But they can be simplified in some special cases.

For the special case $x_B=x_M$ the formulas can be reduced and you get $|MB|^2=(y_B-y_M)^2$ and $k=\sqrt{(y_B-y_M)^2-r^2}$.

$$x_S=\frac{r}{y_B-y_M}\cdot\sqrt{(y_B-y_M)^2-r^2}+x_M \quad \text{and} \quad y_S=\frac{r}{y_B-y_M}\cdot r+y_M$$

$$x_R=\frac{r}{y_B-y_M}\cdot\left(-\sqrt{(y_B-y_M)^2-r^2}\right)+x_M \quad \text{and} \quad y_R=\frac{r}{y_B-y_M}\cdot r+y_M$$

In special case $x_B=x_M+r$, that is, when the tangent is parallel to the y-axis, the point S has coordinates $(x_M+r \mid y_M)$ and the point R has coordinates

$$x_R=\frac{r}{r^2+(y_B-y_M)^2}\cdot\left(r^2-(y_B-y_M)^2\right)+x_M \quad \text{and} \quad y_R=\frac{r}{r^2+(y_B-y_M)^2}\cdot 2r\cdot(y_B-y_M)+y_M$$

Analogous, in special case $x_B=x_M-r$, point R has coordinates $(x_M-r \mid y_M)$ and S has coordinates

$$x_S=\frac{r}{r^2+(y_B-y_M)^2}\cdot\left(-r^2+(y_B-y_M)^2\right)+x_M \quad \text{and} \quad y_S=\frac{r}{r^2+(y_B-y_M)^2}\cdot 2r\cdot(y_B-y_M)+y_M$$

11.2.3 Angles for Points of Contact

Point R is determined by $\gamma = \alpha + \beta$, point S is determined by $\omega = \alpha - \beta$

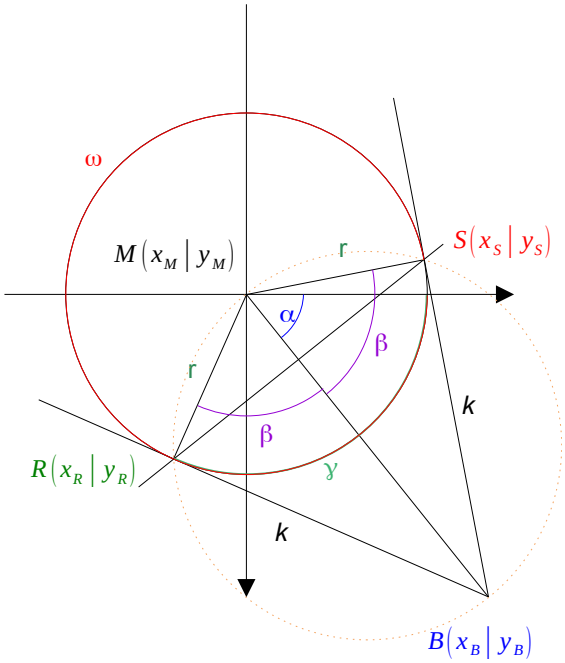


Figure 122: Illustration of tangents to circle including angles

We need to calculate the angles α and β. The formula language in custom shapes has only atan2 as inverse trigonometric function.

Angle α is determined by point B, $\alpha = \text{atan2}(y_B - y_M , x_B - x_M)$.

Angle β is determined by right triangle SBM or MBR, $\beta = \text{atan2}(k , r)$.

In special case $x_B = x_M + r$ you get $\alpha = \beta$ and so $\gamma = 2 \cdot \beta$ and $\omega = 360°$.

In special case $x_B = x_M - r$ you get $\alpha + \beta = 180°$ and so $\gamma = 180°$ and $\omega = 180° - 2 \cdot \beta$.

In special case $x_B = x_M$ you get $\alpha = 90°$ and so $\gamma = 90° + \beta$ and $\omega = 90° - \beta$.

11.2.4 Tangents to Ellipse

Translate the drawing so that the center of the ellipse is at the origin. Mirror on the x-axis if the pole is above the center of the ellipse.

Stretch the drawing so that the ellipse becomes a circle with semi-major axis of the ellipse as radius, keeping the center of the ellipse fixed. Note that the pole point C of the ellipse also moves and becomes a new point B to be a pole of the circle.

In most use cases, the direction in which to stretch is known. Case distinctions are then not necessary.

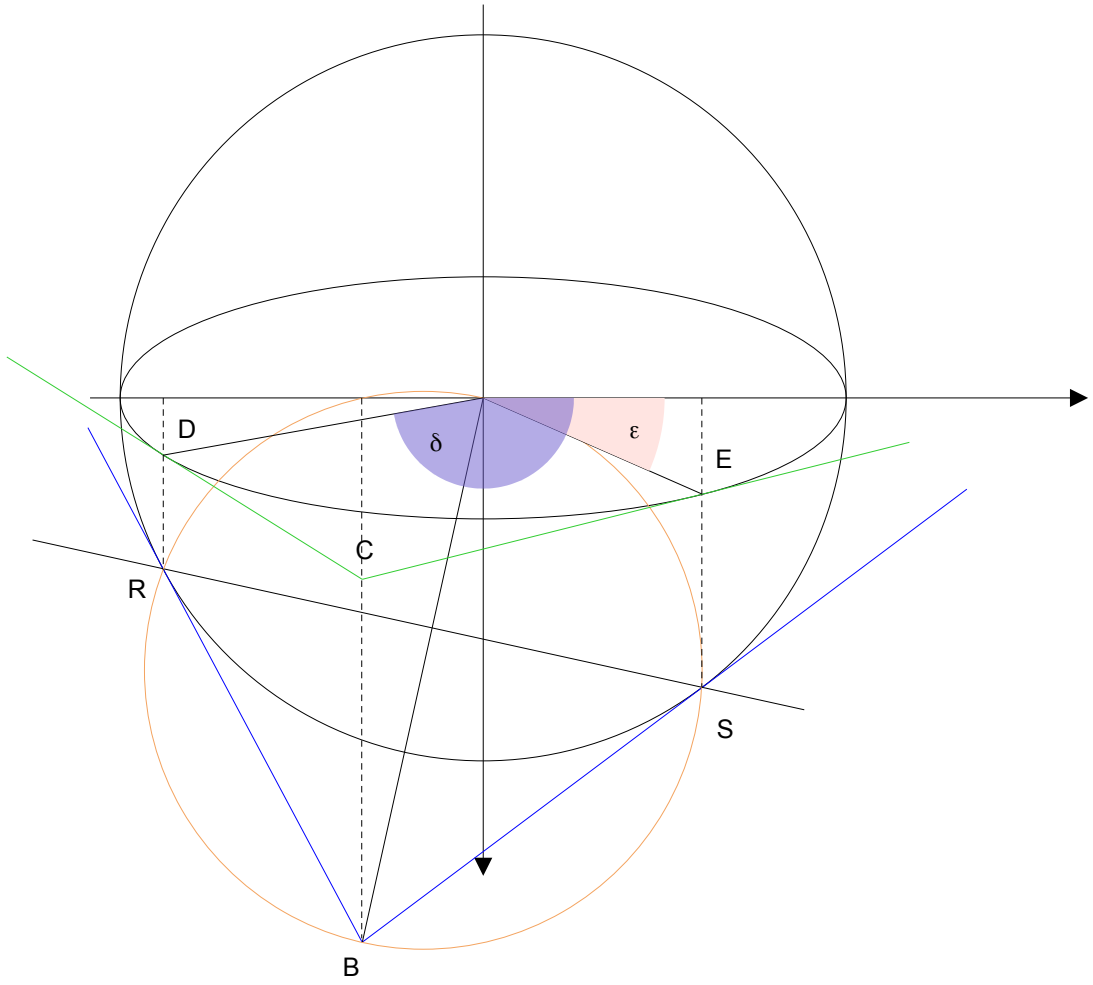


Figure 123: Relationship between tangents to circle and to ellipse

You can calculate the coordinates of point R and S first:

1. Calculate the points of contact R and S of the tangents of the pole B with respect to the circle, as shown in the previous sections.
2. Stretch back the drawing. This will move points R and S on the circle to points D and E on the ellipse. Mirror back if you mirrored in the beginning.
3. Calculate the angles δ and ε from the point coordinates of points D and E using the function atan2 if you need angles in addition.
4. Translate the drawing back to the original position.

Or you calculate the angles α and ω first:

1. Calculate the angles α and ω for pole B with respect to the circle, as shown in the previous sections.
2. Mirror back if you have mirrored in the beginning.
3. Calculate the angles δ and ε using “de La Hire's point construction”[50] for an ellipse.

50 https://en.wikipedia.org/wiki/Ellipse#Drawing_ellipses

$R\left(wR\cdot\cos(\gamma) \middle| wR\cdot\sin(\gamma) \right)$

$A\left(hR\cdot\cos(\gamma) \middle| hR\cdot\sin(\gamma) \right)$

$D\left(wR\cdot\cos(\gamma) \middle| hR\cdot\sin(\gamma) \right)$

The angle δ is then obtained from the Cartesian coordinates of point D

$\delta = \operatorname{atan2}\left(hR\cdot\sin(\gamma) , wR\cdot\cos(\gamma) \right)$

The calculation for angle ε is then analogous.

$\varepsilon = \operatorname{atan2}\left(hR\cdot\sin(\omega) , wR\cdot\cos(\omega) \right)$

4. If you need the Cartesian coordinates of the points of contact, translate the drawing back to the original position.

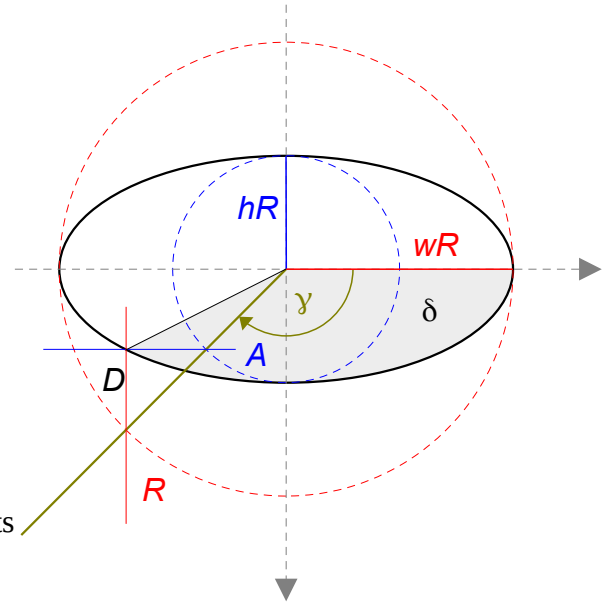


Figure 124: De La Hire's point construction for an ellipse

11.3 Skew Values for Oblique Parallel Projection

Let us assume that the projection direction is determined by "4 right, 2 down, 3 towards observer".

11.3.1 Calculate Skew Values

To calculate the values for the `draw:extrusion-skew` attribute, we describe the direction by a vector in a right-handed coordinate system with y-axis up.

$\vec{b} = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$, in the example $\vec{b} = \begin{pmatrix} 4 \\ -2 \\ 3 \end{pmatrix}$

We now calculate the point $U(u_x|u_y|0)$ on the xy-plane that is the image of the unit point $(0|0|1)$ of the z-axis.

For this point we get the equation

$\begin{pmatrix} u_x \\ u_y \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \lambda\cdot \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$, in the example $\begin{pmatrix} u_x \\ u_y \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \lambda\cdot \begin{pmatrix} 4 \\ -2 \\ 3 \end{pmatrix}$.

It has the solution

$\lambda = -\frac{1}{b_z}$, $u_X = -\frac{b_x}{b_z}$, $u_y = -\frac{b_y}{b_z}$, in the example $\lambda = -\frac{1}{3}$, $u_X = -\frac{4}{3}$, $u_y = \frac{2}{3}$.

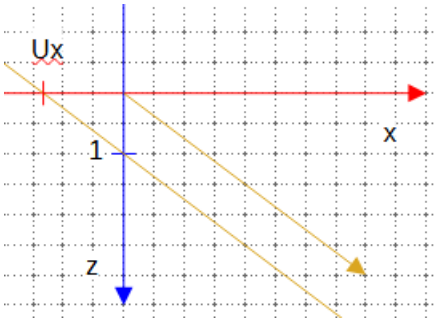
We exclude case $b_z = 0$, because the projection direction would be parallel to the xy-plane.

$r = \frac{1}{|b_z|}\sqrt{b_x^2 + b_y^2}$ and $\varphi = \operatorname{atan2}\left(-\frac{b_y}{b_z} ; -\frac{b_x}{b_z}\right)$

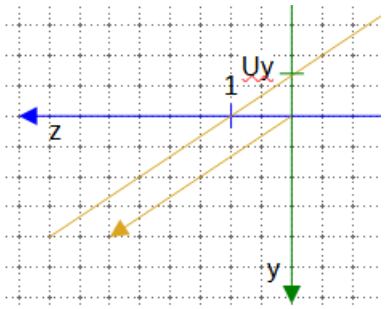
In the example $r \approx 1.4907$ and $\varphi \approx 2.677945\,rad \approx 153.43495\,deg$

In case $b_x = 0 \wedge b_y = 0$ atan2 is not defined. The projection direction would not be oblique, but orthogonal to the xy-plane. In this case we use $r = 0$ and $\varphi = 0$.

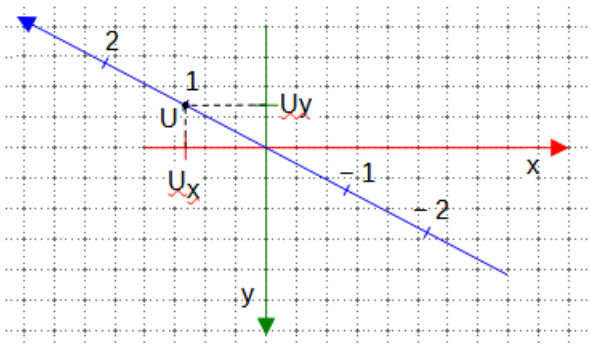
11.3.2 Construct Skew Values



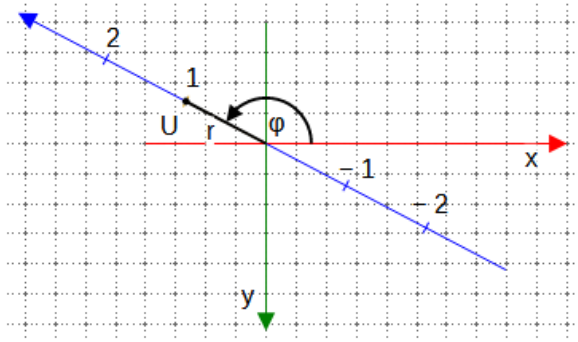
First, we look at the 3D-scene from top to bottom. The grid has 0.5 line spacing. For this view we need only the x-coordinate and the z-coordinate. The yellow arrow is the projection direction. The yellow straight line is parallel to the projection direction. This line meets the xy-plane in the x-coordinate Ux.



Now we look at the 3D-scene from right to left. For this view we only need the y-coordinate and the z-coordinate. The yellow arrow is again the projection direction. Again we construct a parallel straight line through the unit point of the z-axis. It intersects the xy-plane in y-coordinate Uy.



As last step, we look on the 3D-scene from front to back and identify the point U(Ux|Uy). In fact, we get not only the point U. We get the whole image of the z-axis and a scale on this image.

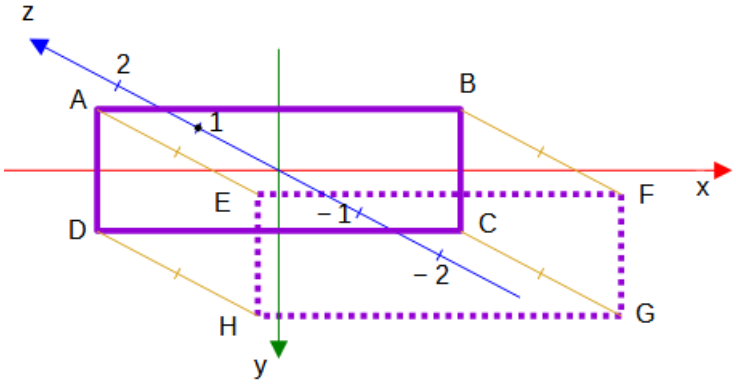


Now you can measure the distance r and the angle φ.

11.3.3 Construct the Projection of an Object

Let’s take as an example an extruded rectangle with 6 cm width, 2 cm height and 2 cm depth. The front face lies in the xy-plane. The center of the rectangle is at the origin of the coordinate system. So the vertices of the front face are A(−3|−1|0), B(3|−1|0), C(3| 1|0) and D(−3|1|0).

The back face lies behind the front face.
So the back face has the vertices E(−3|−1|−2), F(3|−1|−2), G(3| 1|−2) and H(−3|1|−2).



The solid purple rectangle is the front face, the dotted one is the back face. To get the point E, take a copy of the segment from 0 to −2 of the image of the z-axis and move it to the point A. Do the same for all points of the front face.

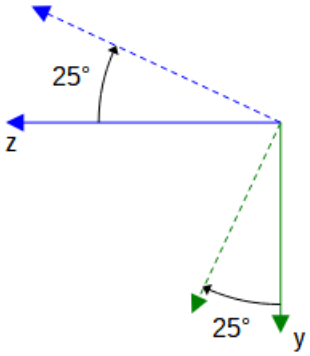
11.4 Angles in Orthographic Projection

11.4.1 From Rotation Angles to Projection

First, consider the example with attribute `draw:extrusion-rotation-angle="25 15"`.

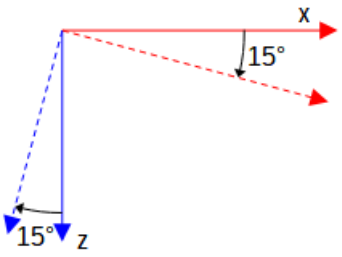
A rotation of 25° around the x-axis corresponds to using 5 times “Tilt up”. The illustration shows this rotation in a view from right to left. This rotation is described by the matrix

$$\boldsymbol{M}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(25°) & -\sin(25°) \\ 0 & \sin(25°) & \cos(25°) \end{pmatrix}.$$



A rotation of 15° around the y-axis corresponds to using 3 times “Tilt Left”. The illustration shows this rotation in a view from top to bottom. This rotation is described by the matrix

$$\boldsymbol{M}_y = \begin{pmatrix} \cos(15°) & 0 & -\sin(15°) \\ 0 & 1 & 0 \\ \sin(15°) & 0 & \cos(15°) \end{pmatrix}.$$



We apply both rotations, first the rotation around the y-axis, then the rotation around the x-axis. So we get the total matrix

$$\boldsymbol{M} = \boldsymbol{M}_x \cdot \boldsymbol{M}_y = \begin{pmatrix} \cos(15°) & 0 & -\sin(15°) \\ -\sin(25°)\cdot\sin(15°) & \cos(25°) & -\sin(25°)\cdot\cos(15°) \\ \cos(25°)\cdot\sin(15°) & \sin(25°) & \cos(25°)\cdot\cos(15°) \end{pmatrix}$$

More generally, the rotation given by the `draw:extrusion-rotation-angle="α β"` attribute is described by the matrix

$$\boldsymbol{M}=\boldsymbol{M}_x\cdot\boldsymbol{M}_y=\begin{pmatrix}\cos(\beta) & 0 & -\sin(\beta)\\ -\sin(\alpha)\cdot\sin(\beta) & \cos(\alpha) & -\sin(\alpha)\cdot\cos(\beta)\\ \cos(\alpha)\cdot\sin(\beta) & \sin(\alpha) & \cos(\alpha)\cdot\cos(\beta)\end{pmatrix}.$$

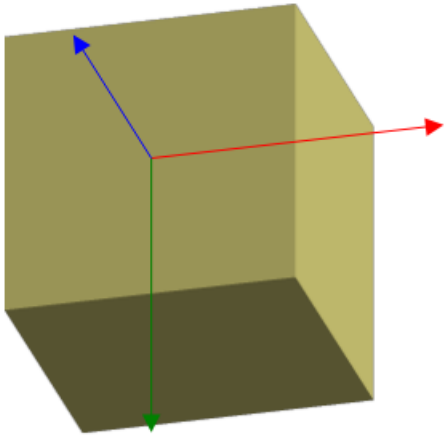
We now look at the object in an orthogonal parallel projection in a front-to-back view. We determine the images of the three coordinate axis in this view. Differences in the z-coordinate are not visible, so we ignore the last row in the matrix. We find the coordinates of the images of the unit points of the axes as columns in the matrix

$$\begin{pmatrix}\cos(\beta) & 0 & -\sin(\beta)\\ -\sin(\alpha)\cdot\sin(\beta) & \cos(\alpha) & -\sin(\alpha)\cdot\cos(\beta)\end{pmatrix}.$$

For the above example, this is the matrix $\begin{pmatrix}0.97 & 0 & -0.26\\ -0.11 & 0.91 & -0.41\end{pmatrix}$ in rounded values.

Image of point $(1|0|0)$ of x-axis is $(0.97|-0.11)$.
Image of point $(0|1|0)$ of y-axis is $(0|0.91)$.
Image of point $(0|0|1)$ of z-axis is $(-0.26|-0.41)$.

The illustration shows a cube with an edge length of 5 cm. The extrusion is behind the shape and the center of rotation is the center of the shape. The axis arrows are drawn so long that they would be 5 cm long in the 3D-scene. Note that the edges of the cube in the view are parallel to the images of the axes and the projected lengths of the cube correspond to the projected lengths of the axis arrows. Remember that in the two-dimensional view of the shape, the y-axis points downward.



11.4.2 From Projection to Rotation Angles

Orthographic projections are usually described in terms of the two-dimensional drawing by the angles to the horizontal x-axis and the reduction factors for axis units.

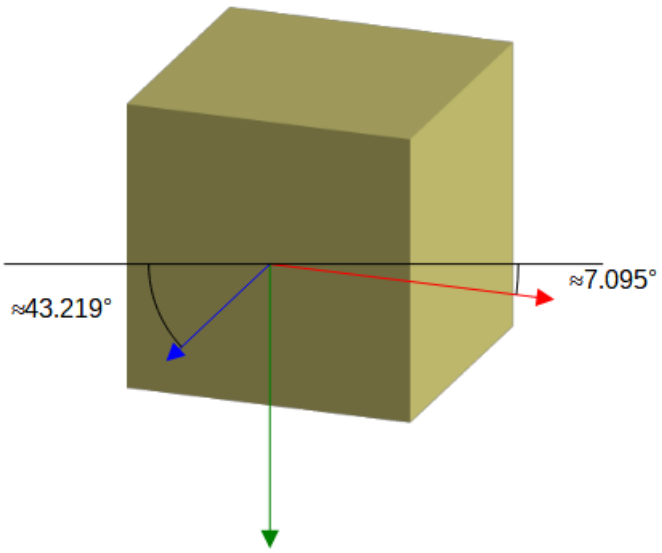
Rotation angle α and β produce the images

$(1|0|0) \longrightarrow U(\cos(\beta)\,|\,-\sin(\alpha)\cdot\sin(\beta))$

$(0|1|0) \longrightarrow V(0\,|\,\cos(\alpha))$

$(0|0|1) \longrightarrow W(-\sin(\beta)\,|\,-\sin(\alpha)\cdot\cos(\beta))$

The “dimetric” projection requires angles of 7° and 42° according to ISO 5456-3. Using the approximated angels $\alpha=-20°$ and $\beta=20°$ gives about $U(0.94|0.12)$, $V(0|0.94)$ and $W(-0.34|0.32)$.



This results in angles $\arctan\left(\frac{0.32}{0.34}\right)\approx 43.219°$ and $\arctan\left(\frac{0.12}{0.94}\right)\approx 7.095°$.

So to get 7° and 42° we need

$$\frac{-\sin(\alpha)\cdot\cos(\beta)}{-(-\sin(\beta))}=\tan(42°) \text{ and } \frac{-\sin(\alpha)\cdot\sin(\beta)}{\cos(\beta)}=\tan(7°)$$

Simplify these equations and you get

$$\frac{-\sin(\alpha)}{\tan(\beta)}=\tan(42°) \text{ and } -\sin(\alpha)\cdot\tan(\beta)=\tan(7°)$$

and then $\sin(\alpha)=-\tan(42°)\cdot\tan(\beta)$ and $(\tan(\beta))^2=\frac{\tan(7°)}{\tan(42°)}$.

Finally, since we already know that $\beta > 0$ and $\alpha < 0$, we get

$$\beta = \arctan\left(\sqrt{\frac{\tan(7°)}{\tan(42°)}}\right) \approx 20.2680671953° \text{ and}$$

$$\alpha = \arcsin\left(-\tan(42°)\cdot\sqrt{\frac{\tan(7°)}{\tan(42°)}}\right) = -19.4205368891° .$$

The "isometric" projection requires angles of 30° and 30° in the 2D drawing. We obtain the necessary 3D rotation angles in a similar way as for the "dimetric" projection.

$$\beta = \arctan\left(\sqrt{\frac{\tan(30°)}{\tan(30°)}}\right) = 45° \text{ and } \alpha = \arcsin\left(-\tan(30°)\cdot 1\right) \approx -35.2643896828°$$

12 Indexes

12.1 Figures

12.2 Tables

12.3 Index of Elements, Attributes and Path Commands

Tip: An alphabetic index as above, does not allow hyperlinks. You can use the tool ‘Go to page’ instead. It is available in the Navigator in the side bar or via short-cut Ctrl+G.

LibreOffice
Community

7

Fantastic Community - Fun Project - Free Office Suite