

Open Office Draw Nutshells

Nicole Scholz

Vienna University of Economics and Business Administration

Reg. No. 0251817

E-Mail: h0251817@wu-wien.ac.at

May 29, 2007

seminar course paper

Institute for Management Information Systems

Prof. Dr. Rony G. Flatscher

Table of Contents

Table of Contents	2
1 Introduction.....	5
2 Description of the Main Elements.....	6
2.1 Open Office.org.....	6
2.2 Object Rexx.....	7
2.3 BSF4Rexx	8
3 Draw Examples.....	9
3.1 Draw Object Examples.....	10
3.1.1 Example 01 – Line.....	10
3.1.2 Example 02 – objects – create	12
3.1.3 Example 03 – object – remove	13
3.1.4 Example 04 – object – solid.....	13
3.1.5 Example 05 – object – group.....	14
3.1.6 Example 06 – object – cornerradius	15
3.1.7 Example 07 – object – shear	16
3.1.8 Example 08 – object – rotation	16
3.1.9 Example 09 – object – gradient	17
3.1.10 Example 10 – object – hatch	19
3.1.11 Example 11 – object – solidtransparence.....	20
3.1.12 Example 12 – object – solidlinestyle	21
3.1.13 Example 13 – object – shadow.....	22
3.1.14 Example 14 – object – ellipse	23
3.2 Draw Text Examples	24
3.2.1 Example 15 – text – adjust	25
3.2.2 Example 16 – text – fitproportional	26
3.2.3 Example 17 – text – animation	27
3.2.4 Example 18 – text – animationscroll	28
3.2.5 Example 19 – text – properties	29
3.3 Draw Diagram Examples.....	30
3.3.1 Example 20 – diagram – connect	30

3.3.2 Example 21 – diagram.....	31
4 Conclusion.....	35
5 References.....	36
6 Attachment.....	38

Figures

figure 1: Example 01 – Line.....	12
figure 2: Example 02 – objects – create.....	13
figure 3: Example 04 – object – solid.....	14
figure 4: Example 05 – object – group.....	15
figure 5: Example 06 – object – cornerradius.....	16
figure 6: Example 07 – object – shear.....	16
figure 7: Example 08 – object – rotation.....	17
figure 8: Example 09 – object – gradient.....	18
figure 9: Example 10 – object – hatch.....	20
figure 10: Example 11 – object – solidtransparence.....	21
figure 11: Example 12 – object – solidlinestyle.....	22
figure 12: Example 13 – object – shadow.....	23
figure 13: Example 14 – object – ellipse.....	24

figure 14: Example 15 – text – adjust	26
figure 15: Example 16 – text – fitproportional	27
figure 16: Example 17 – text – animation	28
figure 17: Example 18 – text – animationscroll	28
figure 18: Example 19 – text – properties	30
figure 19: Example 20 - diagram - connect.....	31
figure 20: Example entry.....	31
figure 21: Example 21 – diagram – three levels.....	34
figure 22: Example 21 - diagram - one level	34
figure 23: Example 21 - diagram - two levels.....	34

1 Introduction

This paper is about the automation with Draw of Open Office.org and Object Rexx. To automate Draw three elements are needed Open Office.org, Object REXx and BSF4Rexx. These programs are all open source software and can be downloaded from the internet. The first chapter give a brief overview of these elements and provides links where they can be downloaded.

In the third chapter the Draw Nutshells are described. In this chapter it will be illustrated the three elements mentioned above can be used to automate Draw. Therefore parts of the example program code is shown and explained. The whole program code of the examples is added as attachment. The third chapter is divided into three parts.

In the first part the examples showing how to automate the creation of objects are described. It shows how to create shapes like lines, rectangles and circles and how to change them. The line for example can be set dash, the colour of the objects and the appearance of the shapes can be changed. There is also the possibility to add shadows or let the shapes rotate.

The second part shows how text can be added to shapes. Furthermore it will be displayed how the text can be animated, how the size and colour of the text can be changed and how the font can be modified.

In the third part the connection to other shapes is described and how a diagram can be constructed with this connectors. This example is more complex than the others because it includes a lot of features of the previous examples and combines them to create and format the diagram.

2 Description of the Main Elements




The following chapters give a brief overview about the elements that were used to create the Draw nutshells. Three elements are necessary to create the nutshells: Open Office.org, Object Rexx and BSF4Rexx.




2.1 Open Office.org

In 1980 StarDivision created StarOffice. In 1999 Sun Microsystems bought the StarOffice software and enhanced it to Open Office.org. Open Office can be downloaded for free. The latest version is available at: <http://de.openoffice.org/downloads/quick.html> [OOOorga].

Open Office includes programs supporting the common office work. It is maintained by an open source community which has the goal that the programs are running on all major platforms and providing access to all data through APIs and XML-based file format [OOOorga].

The Open Office.org programs are all suitable with other major office software packages. It offers five office programs [OOOorgb]:

-  **Writer:** The Writer is the word processor of the office software. It can be used for writing letters, papers or books. The Writer offers a lot of possibilities to format the written text and to insert tables and diagrams [OOOorgb].
-  **Calc:** The Calc program is a spreadsheet which provides all necessary tools for making calculations, analysis or creating diagrams and reports [OOOorgb].
-  **Impress:** Impress can be used to create multimedia presentations [OOOorgb].

-  **Draw:** With Draw shapes can be painted in different ways and diagrams can be made [OOOorgb].
-  **Base:** Base is used to create databases. Base offers a lot of ways to modify the tables, forms and queries within a database [OOOorgb].
-  **Math:** Can be used to create mathematical equations [OOOorgb].

2.2 Object Rexx

Object Rexx is a programming language and is an open source project. The Rexx Language Association provides a free implementation of Object Rexx [ooRex07].

Object Rexx is an extension of the classic Rexx language. It is an object-orientated program language and therefore includes classes, objects and methods. But this extension does not replace the function of the classic Rexx language. It only provides new functions additionally to classic Rexx [ooRex07].

The following list shows the main aspects of Object Rexx:

- **An English-like language:** A lot of instructions of Object Rexx are English words like SAY, PULL and EXIT. This makes using this programming language easier [ooRex07].
- **Fewer rules:** There are only few rules about format for example there is the possibility to add multiple instructions to a single line and instructions can begin in any column [ooRex07].
- **Interpreted, not compiled:** To start the program it must not be compiled the language processor reads each line from the source file and executes it [ooRex07].
- **Built-in functions and methods:** Rexx provides many built-in functions like comparison operations for text and numbers [ooRex07].

- **Typeless variables:** In Rexx every data is treated as object of different kind so variables can hold any type of object. The type of a variable must not be declared to use it the proper way [ooRex07].
- **String handling:** Rexx offers a functionality to manipulate character strings. This enables programs to read and separate characters, numbers and mixed input [ooRex07].
- **Decimal Arithmetic:** The arithmetic operations in Rexx are based on decimal arithmetic [ooRex07].
- **Clear error messages and powerful debugging:** When the Rexx program encounters an error messages with meaningful explanations are displayed [ooRex07].

The latest version can be downloaded at: <http://www.oorexx.org/download.html>.

2.3 BSF4Rexx

BSF4Rexx is the Bean Scripting Framework for Object Rexx which offers the possibility to use Java objects and methods.

It was developed by Peter Kalendar and professor Rony G. Flatscher in 2000 and was called Essener Version. In 2003/2004 the Augsburgener Version was developed and offered the possibility to start Java from Rexx. Three years later the Vienna Version was developed. This version contains new functions for the automation of Open Office.org and no strict scripting is required [Flat06].

The latest version of BSF4Rexx and the installation guide consisting of two readme files - the readmeBSF4Rexx.txt and readmeOOo.txt - are available at: <http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/>.

3 Draw Examples

The following "nutshells" are short examples which demonstrate the automation of Open Office Draw. They show a lot of possibilities to automate the work with Draw. The examples should be recognised as an incentive to create other programs with a higher complexity.

The examples are divided into three categories. The first category represents the automation of objects with Draw. The second category presents the automation of text with Draw. The last category is about inserting a diagram in Draw. The order of the programs within these categories is settled like the creation order of the examples.

All examples are developed on a Windows XP computer. Most of the examples are created with Open Office.org 2.1 and the others with Open Office.org 2.2. But all examples are tested with Open Office.org 2.2.

In all examples the following program code lines are needed to automate Draw with Object Rexx. First a connection to Open Office.org has to be implemented to ensure Open Office could be automated with Object Rexx. Therefore the UNO service from Open Office.org is needed because it provides bridges to objects written in different programming language. So method calls can be send and return values can be received between different programming languages [OOOAPIa]. The UNO service can be get with the following program code.

```
::requires UNO.cls -- get UNO support
```

The "com.sun.star.frame.Desktop" service is the environment for the components. It is needed to load the required components to create a Draw document [OOOAPIc].

```
/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/  
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object  
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface  
/* open the blank file */  
url = "private:factory/sdraw"  
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,   
                .UNO~noProps)
```

The `MultiServiceFactory` is the main interface from the "com.sun.star.lang.ServiceManager" which is the main factory of every UNO application. The `MultiServiceFactory` contains the `createInstance()` method. This method is used to create an instance of a service which supports every interface specified in the service [OOOAPIa].

```
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory
```

To get the `DrawPage` which contains the drawings the "com.sun.star.drawing.XDrawPageSupplier" is needed [OOOAPId]. This service includes the method `getDrawPages` which returns the `DrawPage` [OOOAPIe].

```
/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage
```

3.1 Draw Object Examples

Draw offers a lot of possibilities to automate objects for example draw a rectangle or an ellipse, change the colour, add a shadow and change the kind of a circle.

The following examples give an overview about some options of automation.

3.1.1 Example 01 – Line

The first example shows how to draw a line in an empty Draw – document and how to change the shape of the painted line.

```
/* create a line and add it to the shape */
Line = xDocumentFactory~createInstance("com.sun.star.drawing.LineShape")~XShape
Line~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 2000))
Line~setSize(.bsf~new("com.sun.star.awt.Size", 15000, 0))
xDrawPage~add(Line)
```

To draw a line an instance of the "com.sun.star.drawing.LineShape" has to be created. Creating the "com.sun.star.drawing.LineShape" service and the `XShape` interface is possible with the help of the `XmultiServiceFactory` of the `xDrawComponent`.

Before the `XShape` can be added to the `XDrawPage` it needs a size and a position. To get the size of a line the "com.sun.star.awt.Size" service is required.

This service needs two parameters. The first parameter is the size of the length of the line. The second parameter is the width. The width is usually set zero to get a straight line as you can see in the program code above. To get a diagonale the size of the second parameter can be set greater than zero.

With the "com.sun.star.awt.Point" service the position of the shape can be set. This service requires two parameters, the x coordinate and the y coordinate. The coordinates start in the left corner. So the coordinates are increasing from left to right and from the top to bottom [OOOAPIb]. The y coordinate should be set 11 minimum that the line appears on the document.

After adding the size and the position the XShape can be added to the XDrawPage with the command add.

```
xShapeProps = Line~XPropertySet
xShapeProps~setProperty("LineStyle",
    bsf.getConstant("com.sun.star.drawing.LineStyle", "DASH")) -- set line style dash
```

The program code above shows how to paint a dash line. To set the property value of the line the interface XPropertySet is required. The method setPropertyValue sets the value of the property with the specified name. To set the linestyle dash the "com.sun.star.drawing.LineStyle" service is needed.

```
xShapeProps1 = xLine1~XPropertySet
xShapeProps1~setProperty("LineColor", box("int", "228B22"x ~c2d)) -- paint the line green
xShapeProps1~setProperty("LineWidth", box("int", "500")) -- set line width 5 millimeter
xShapeProps1~setProperty("LineTransparence", box("int", "80")) -- set transparency 80 per cent
```

The program code above displays how to change the colour and the width of the line. It also shows how to make the line colour transparent. To change the colour of the line the property LineColor is required. This property needs to be passed the requested colour in hexadecimal. In the program code above the haxadecimal of the colour green is given as parameter.

With the property LineWidth the width of the line can be changed. The width of the line is stated in hundredth millimetres. The width of the line in the example below is five millimetres [Sun07].

The transparency is set with the property LineTransparence. This property needs a parameter between 1 and 100 per cent. If the parameter is 100 per cent the

line is fully transparent. In the example the transparency is set 80 per cent [Sun07].

The figure below shows the output of the program.

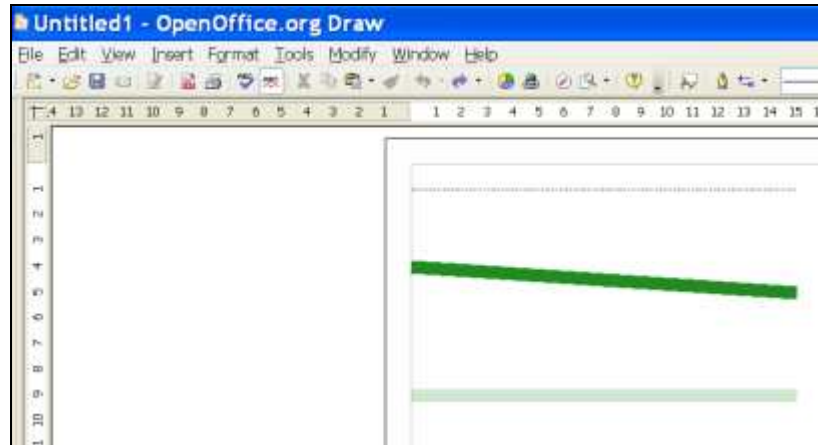


figure 1: Example 01 – Line

3.1.2 Example 02 – objects – create

The second example displays how to create a simple shape, a line, a rectangle and a circle.

```
/* create a rectangle and add it to the shape */
xBox = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xShape
xBox~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 8000))
xBox~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xBox)
```

An instance of "com.sun.star.drawing.RectangleShape" is needed to create a rectangle. Furthermore the size and the position of the rectangle has to be set before the rectangle can be added to the XDrawPage.

```
/* create a circle and add it to the shape */
xCircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xShape
xCircle~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 15000))
xCircle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xCircle)
```

To paint a circle an instance of "com.sun.star.drawing.EllipseShape" has to be created. The size and the position have to be set and then the circle can be added to the document.

If no colour is specified the fill colour of the objects is default light blue and the line colour of the created objects is painted in black as the figure below shows.

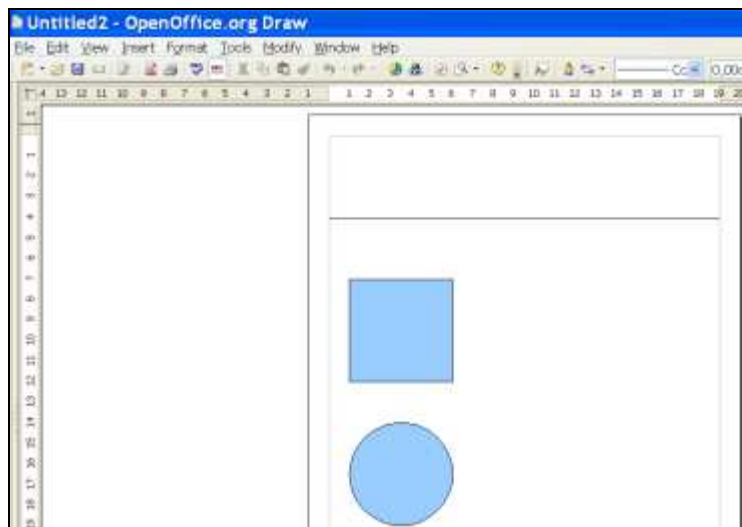


figure 2: Example 02 – objects – create

3.1.3 Example 03 – object – remove

The following program code shows how to remove a painted object. An object that was added to the XDrawPage can be removed with the command remove.

```
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle) -- add rectangle to the draw document

call sysleep 2

xDrawPage~remove(xrectangle) -- remove rectangle from the draw document
```

3.1.4 Example 04 – object – solid

The fourth example shows how to change the line colour and the fill colour of a rectangle and to set the fill style solid.

```
/* colour the rectangle pink solid */
xShapeProps = xrectangle~XPropertySet
/* set fill colour pink */
xShapeProps~setProperty("FillColor", box("int", "ff 00 ff"x ~c2d))
/* set line colour blue */
xShapeProps~setProperty("LineColor", box("int", "00 00 ff"x ~c2d))
/* fill the rectangle */
xShapeProps~setProperty("FillStyle",
  bsf.getConstant("com.sun.star.drawing.FillStyle", "SOLID"))
```

The colour of an object can be changed with the properties `FillColor` and `LineColor`. The properties need the parameter of the colour in hexadecimal. In the example the fill colour is set pink and the line colour blue.

With the service `"com.sun.star.drawing.FillStyle"` the fill style of the rectangle can be set. The fill style of this example is solid. If no fill style is chosen the default fill style is solid.

The following figure shows the output of the fourth example. A rectangle is painted and coloured pink. The style of the line is set dash and the line has been coloured blue.

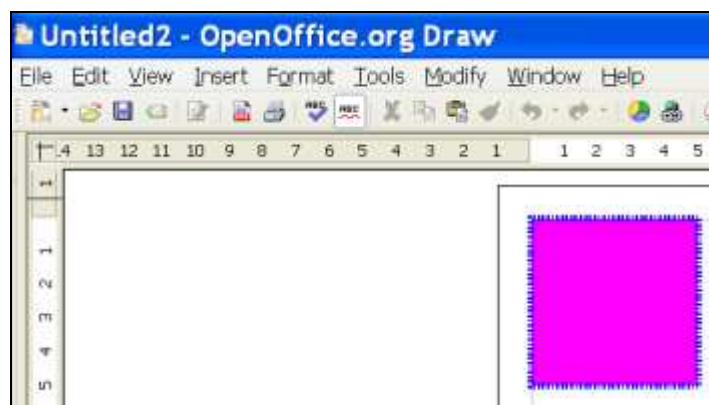


figure 3: Example 04 – object – solid

3.1.5 Example 05 – object – group

If the `XDrawPage` contains more than one shape the shapes can be grouped and handled as one object.

```
xContext = UNO.connect() -- connect to server and retrieve the XContext object
XMcf = xContext~getServiceManager -- retrieve XMultiComponentFactory
```

An important step is to connect to the server and to retrieve the `xContent` object and to the `XMultiComponentFactory`.

```
/* group the 2 shapes */
-- get ShapeCollection from the XMultiComponentFactory
xObj = xMcf~
  createInstanceWithContext("com.sun.star.drawing.ShapeCollection", xContext)
xToGroup = xObj~XShapes
xToGroup~add(xrectangle)
xToGroup~add(xrectangle1)
```

```
xShapeGrouper = xDrawPage~xShapeGrouper
xShapeGroup = xShapeGrouper~group(xToGroup)
```

To group shapes on an XDrawPage the XShapes interface from the "com.sun.star.drawing.ShapeCollection" service is needed. Therefore the method createInstanceWithContext of the interface XMultiComponentFactory is used. After adding the shapes to the XShapes interface the XShapeGrouper is required to group the shapes together. With the XShapeGrouper the objects can be ungrouped as well. Then the whole group can be allocated to another position.

```
xShapeGroup~setPosition(.bsf~new("com.sun.star.awt.Point", 7000, 7000))
```

The figure below shows the output of the example. Two rectangles are grouped together and then allocated as one shape to another position.

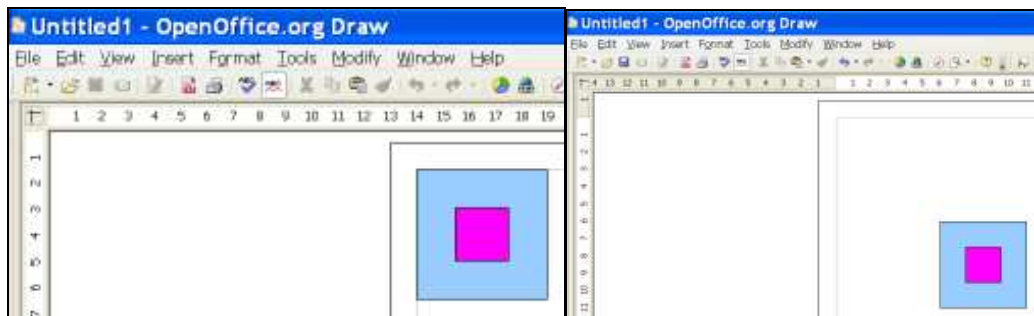


figure 4: Example 05 – object – group

3.1.6 Example 06 – object – cornerradius

The sixth example shows how to change the corner radius of a rectangle.

```
xShapeProps~setProperty("CornerRadius", box("int", 500))
call sys.sleep 2
xShapeProps~setProperty("CornerRadius", box("int", 1000))
call sys.sleep 2
xShapeProps~setProperty("CornerRadius", box("int", 2000))
```

The corner radius can be changed with the property CornerRadius. The CornerRadius is given in hundredth millimetres. The program code above shows the change of the corner radius from five millimetres to one centimetre to two centimetres [Sun07].

The figure displays a rectangle with a corner radius of five millimetres and a rectangle with a corner radius of two centimetres.

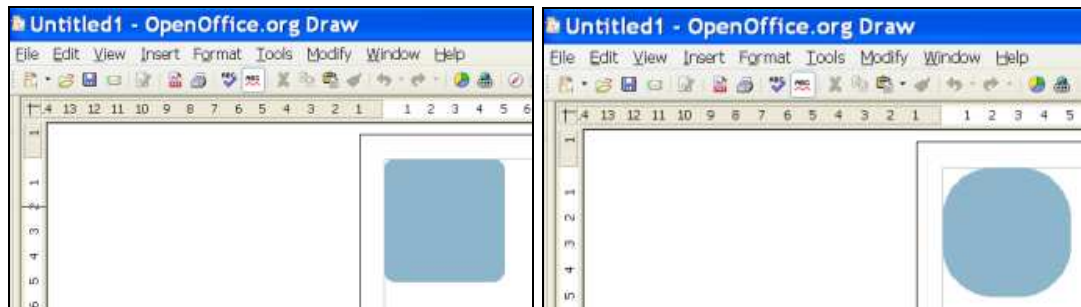


figure 5: Example 06 – object – cornerradius

3.1.7 Example 07 – object – shear

```
/* The shape is sheared counter-clockwise around the center of the bounding box */
call syssleep 1
xShapeProps~setProperty("ShearAngle", box("int", 2000))
call sysleep 1
xShapeProps1~setProperty("ShearAngle", box("int", 10000))
```

The ShearAngle is the amount of shearing for the object. The object is sheared counter-clockwise around the center of the bounding box [OOOAPIf].

The figure below shows two rectangles with a shear angle.

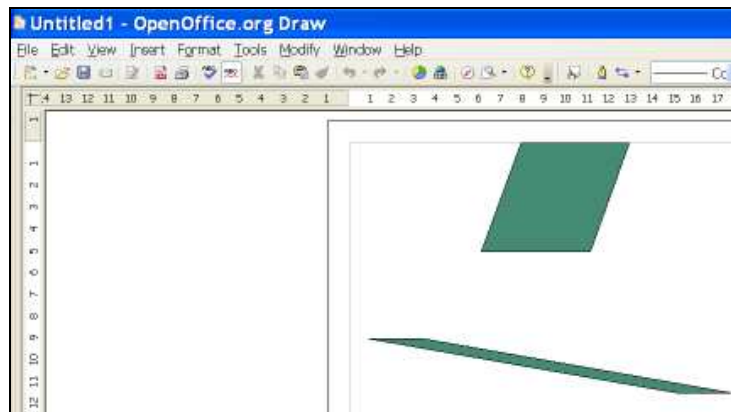


figure 6: Example 07 – object – shear

3.1.8 Example 08 – object – rotation

The program code below shows how to set the rotate angle of an object. The object is rotated counter-clockwise around the center of the bounding box[OOOAPIf].


```

/* rotate the rectangle: counter-clockwise around the center of the bounding box */
call sysssleep 1
xShapeProps~setProperty("RotateAngle", box("int", 10000))

```

The figure below shows a rectangle in normal position and then the same rectangle after the rotation angle is set.

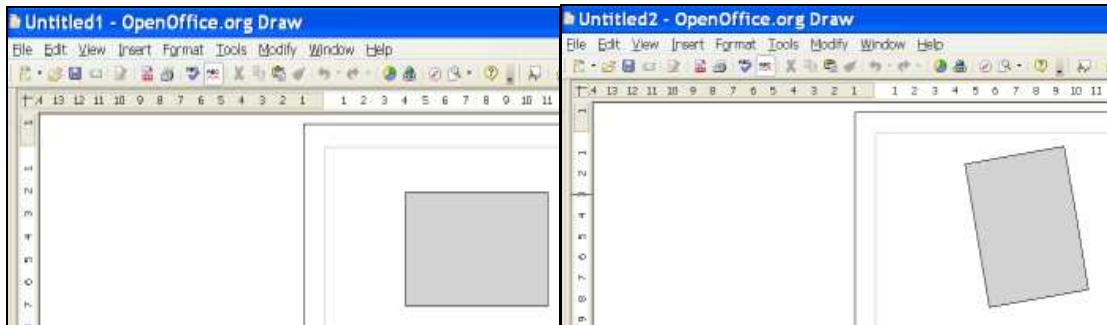


figure 7: Example 08 – object – rotation

3.1.9 Example 09 – object – gradient

The following Draw example creates two rectangles and changes the colour of the rectangles. The fill style of the rectangles is set gradient.

```

/* colour the rectangle gradient linear */
xShapeProps1 = xrectangle1~XPropertySet

oGradient = .bsf~new("com.sun.star.awt.Gradient")
oGradient~Style = bsf.getConstant("com.sun.star.awt.GradientStyle", "LINEAR")
oGradient~StartColor = 8526139
oGradient~EndColor = 238210238
oGradient~Angle = 450 -- angle of 45 degrees
oGradient~StartIntensity = 150
oGradient~EndIntensity = 150
oGradient~StepCount = 100 -- gradient with 100 sinlge colour steps

xShapeProps1~setProperty("FillStyle",
    bsf.getConstant("com.sun.star.drawing.FillStyle", "GRADIENT"))
xShapeProps1~setProperty("FillGradient", oGradient)

```

The program code above shows how a gradient can be customized. To customize the gradient the service "com.sun.star.awt.Gradient" is required. With the "com.sun.star.awt.GradientStyle" service the style of the gradient can be changed. There are two possibilities for the gradient style. The first is to set the style "LINEAR" and the second "RADIAL". In the program code above the gradient style linear is chosen [Sun07].

The start colour set the colour where the gradient begins and the end colour is the colour where the gradient finishes. The colour for these two parameters needs the colour strength from the R/G/B (Red/Green/Blue) schemata. The angle gives the angle of the gradient in tenth of a degree. In the program code above the angle is set 45 degrees [Sun07].

The StartIntensity and the EndIntensity are used to set the colour intensity of the gradient as percentage. The intensity in this program code is set 150 per cent so the colour appears brighter than the value given with the attributes StartColor and EndColor. If the intensity is set under 100 per cent the colour appears darker [Sun07].

The last attribute is the StepCount. The StepCount is the number of single colour steps that are used for the gradient. In the program code above the gradient has 100 single colour steps [Sun07].

If the gradient is not customized the start colour is black and the end colour white. The angle is zero so the lines are parallel to the y axis.

The figure below displays the output of the Example. There are two rectangles with a gradient. The first gradient is linear and the second radial.

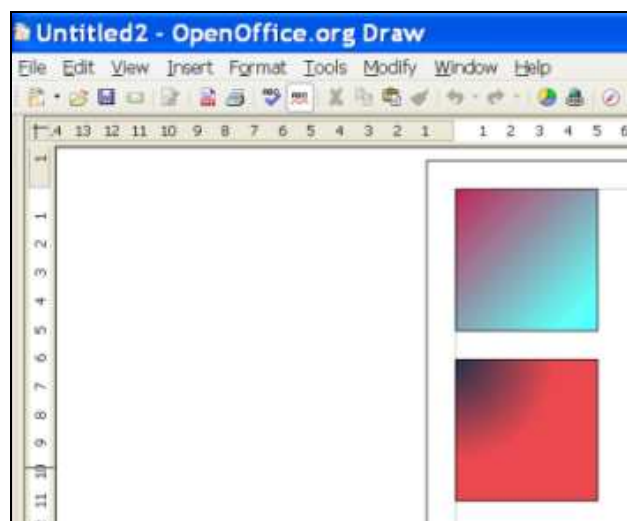


figure 8: Example 09 – object – gradient

3.1.10 Example 10 – object – hatch

The tenth example creates three rectangles and sets the fill style of each rectangle hatch.

```
/* colour the rectangle hatch single */
xShapeProps = xrectangle~XPropertySet
ohatch = .bsf~new("com.sun.star.drawing.Hatch")
ohatch~Style = bsf.getConstant("com.sun.star.drawing.HatchStyle", "SINGLE")
ohatch~Color = 2554848
ohatch~Distance = 1000 -- distance is 1 centimeter
ohatch~Angle = 900 -- angle of 90 degrees

/* fill the rectangle */
xShapeProps~setProperty("FillStyle",
    bsf.getConstant("com.sun.star.drawing.FillStyle", "HATCH"))
xShapeProps~setProperty("FillHatch", ohatch)
```

To set the fill style of a rectangle hatch the service "com.sun.star.drawing.FillStyle" is needed. There is the possibility to customize the hatch. To customize the hatch the "com.sun.star.drawing.Hatch" service is required.

To change the style of the hatch the "com.sun.star.drawing.HatchStyle" service is needed. There are three possible styles of the hatch. The first possibility is single so the object has a horizontal line. The second alternative is to set the style double. If the style is set to double the object has a horizontal and a vertical line. The last possibility is to set the style triple. Then the object has a horizontal, vertical and diagonal line. In the program code above the style is set single [OOOAPIg].

With the attribute Color the line colour is defined. The colour needs three parameters R/G/B: the colour strength of red, green and blue [Sun07].

The distance gives the distance between the lines in hundredth millimetres. In the program code above the distance is 1000 hundredth millimetres so the distance is one centimetre [Sun07].

The angle is given in tenth of a degree. The program code displays an angle of 90 degrees so the lines are parallel to the x axis [Sun07].

If the hatch is not customized there are single hatches and the colour of the lines is black. The angle is zero so the lines are parallel to the y axis.

The figure below shows the output of the tenth example. There are three rectangles. The first rectangle has a single hatch pattern with turquoise lines which are arranged with an angle of 90 degrees. The second rectangle has a double hatch pattern with red lines and the third rectangle has a triple hatch pattern with green lines.

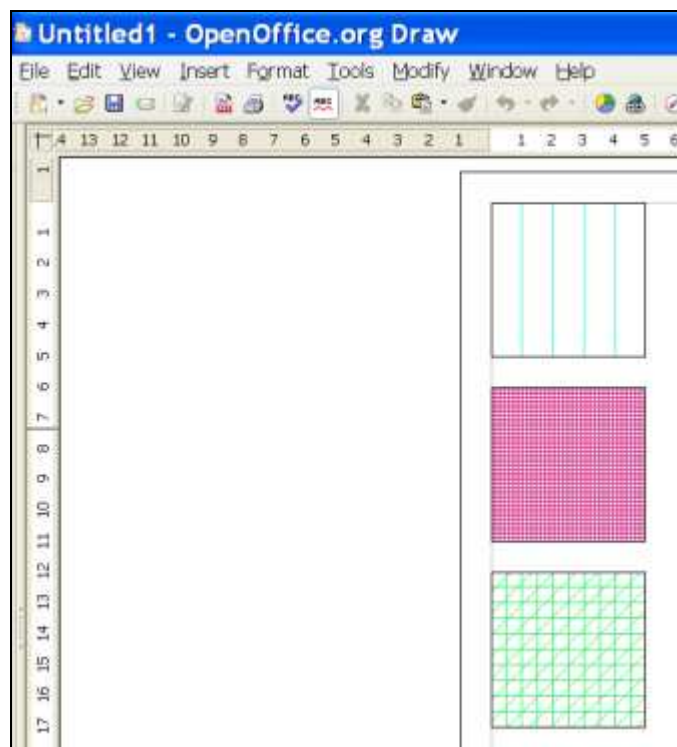


figure 9: Example 10 – object – hatch

3.1.11 Example 11 – object – solidtransparence

This example displays how to get an object transparent. The transparency is set with the property FillTransparence. The parameter is given in per cent. If the object should be fully transparent the parameter has to be set to 100 per cent [Sun07]. The program code below shows that the FillTransparence is set at 50 per cent.

```
xShapeProps1~setPropertyvalue("FillTransparence", box("int", "50")) -- set transparence 50 percent
```

Below you can see the output of the example. It shows two rectangles with a black fill colour and a pink line colour. The transparency of the first rectangle is set to 50 per cent and the transparency of the second rectangle is set to 90 per cent.

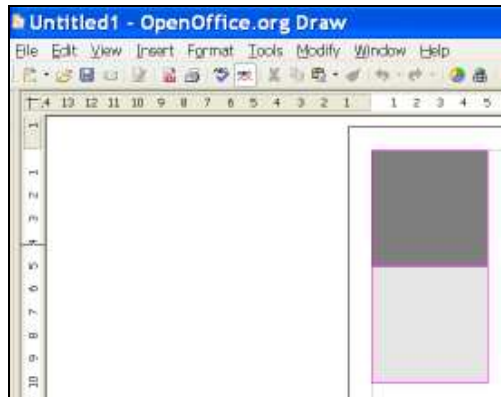


figure 10: Example 11 – object – solidtransparence

3.1.12 Example 12 – object – solidlinestyle

This example shows how to change the line joint of an object. To change the line joint the "com.sun.star.drawing.LineJoint" service is needed. There are five possibilities to change the line joint. One possibility is to use the command "MITER" to set the line joint as the program code displays below. If the command "MITER" is used the lines join at intersections [OOOAPIh].

```
/* change the colour properties */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "191970"x ~c2d)) -- set fill colour blue
xShapeProps~setProperty("LineColor", box("int", "48D1CC"x ~c2d)) -- set line colour turquoise
xShapeProps~setProperty("LineWidth", box("int", "200")) -- set line width 20 millimeter
xShapeProps~setProperty("LineJoint",
    bsf.getConstant("com.sun.star.drawing.LineJoint", "MITER")) -- the lines join at intersections
```

If the command "BEVEL" is used to set the line joints the edges of the lines are joined by lines. Another possibility is to choose the option "NONE". Then the lines will not be connected. The fourth way to join the lines is to use the option "MIDDLE". With this possibilities the middle value between the joints is used to connect the lines. If the value "ROUND" is used to join the lines the lines join with an arc [OOOAPIh].

The following figure shows three rectangles with three different line joints: the first rectangle with "MITER" the second with "BEVEL" and third with "NONE".

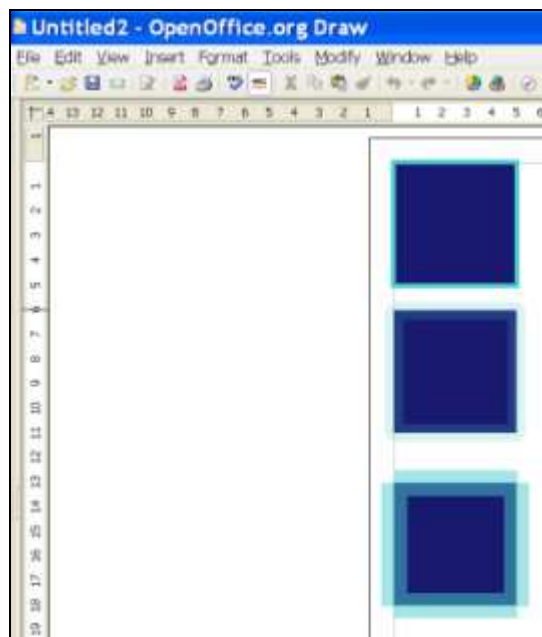


figure 11: Example 12 – object – solidlinestyle

3.1.13 Example 13 – object – shadow

There is the possibility to add a shadow to an object. To add a shadow the shadow property has to be set true.

```
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("Shadow", box("boolean", "True" )) -- set shadow property true
```

If the shadow property has been set to true there is the possibility to customize the shadow. One option to customize the shadow is to change the colour. In the program code below the colour is set to black. The parameter to change the colour has to be given in hexadecimal. The second possibility to vary the shadow is to specify another distance. This can be achieved by setting the the ShadowXDistance and the ShadowYDistance. These parameters are given in hundredth millimetres so the shadow distance of both parameters of the shadow in the example below is two centimetres [Sun07].

```
/* customize shadow */
xShapePropsI~setProperty("ShadowColor", box("int", "000000"x ~c2d )) -- set the shadow colour black

/* set the shadow distance */
xShapePropsI~setProperty("ShadowXDistance", box("int", "2000"))
```

```
xShapeProps1~setProperty("ShadowYDistance", box("int", "2000"))
call sys$sleep 1
xShapeProps1~setProperty("ShadowTransparency", box("int", "90")) -- set transparency to 90 per cent
```

Another way to change the shadow is by setting the ShadowTransparency. The ShadowTransparency has to be assigned in per cent [Sun07].

If the shadow is not customized it is displayed with a small distance of approximately five millimetres to the object and in dark grey.

The following figure shows two rectangles: one with a normal shadow and the other one with a customized shadow.

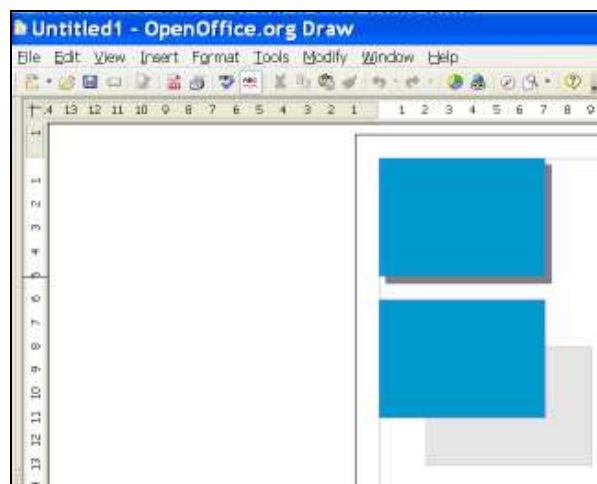


figure 12: Example 13 – object – shadow

3.1.14 Example 14 – object – ellipse

There are a few ways to manipulate an ellipse. This example gives an short overview of some possibilities.

```
/* 1st circle: show the cut circle connected by 2 lines */
xShapeProps~setProperty("CircleStartAngle", box("int", "2000")) -- start angle 20 degrees
xShapeProps~setProperty("CircleEndAngle", box("int", "9000")) -- end angle 90 degrees
xShapeProps~setProperty("CircleKind",
  bsf.getConstant("com.sun.star.drawing.CircleKind", "SECTION"))
```

To manipulate an ellipse the "com.sun.star.drawing.CircleKind" service is needed. For the definition of the kind of the circle two properties have to be set: the "CircleStartAngle" and the "CircleEndAngle". The parameter of these properties has to be given in a tenth of a degree. In this program code the angle of the

circle segment is 70 degrees. This is the difference of the "CircleStartAngle" and the "CircleEndAngle" [Sun07].

With the service "com.sun.star.drawing.CircleKind" four kinds of circles can be defined. First a full circle can be painted. Another possibility is to draw a section of a circle as the program code above shows. The third way to manipulate an ellipse is to display only a segment of a circle. This possibility creates a divided circle whose interfaces are directly connected. The last alternative is to illustrate only the arc of a circle [Sun07].

Below the figure shows three manipulated circles. The first object is a section of a circle. The following object displays a divided ellipse with directly connected interfaces. At last the object shows only the arc of a circle.

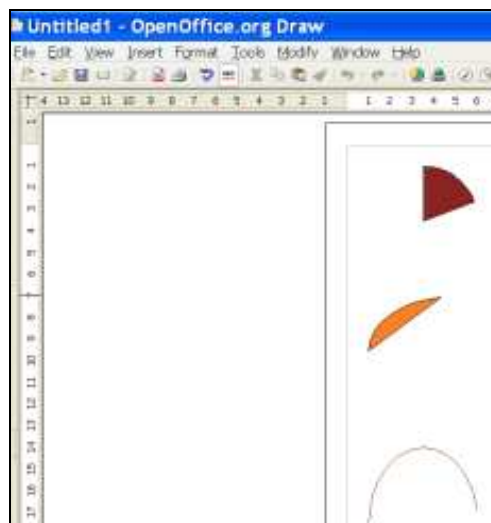


figure 13: Example 14 – object – ellipse

3.2 Draw Text Examples

Open Office Draw provides a variety of functions to automate text. The added text can for example be animated or adjusted.

The following examples give a short view about some possibilities to automate text with Draw.

3.2.1 Example 15 – text – adjust

The following example presents how to add a text to an object and how this text can vary its positions in the object.

```
xCircle~xText~setString("I am a circle!") -- add text to the circle
```

Adding a text to an object is possible with the method `setString("string")`. This method is from the interface `XText`. In the code above the String "I am a circle" is added to the object. The added text is not flexible.

The position of the added text can be changed if a text cursor is added to the `XText`. A text cursor is a moveable text range and so the text position can be modified [OOOAPIb].

To manipulate the text the service "com.sun.star.drawing.TextVerticalAdjust" is needed. This service enables to specify the vertical position of a text inside an object in the relation to the object. With this service the whole text inside a shape is manipulated and not only individual text lines [OOOAPIi].

In this example two values of this service are used. With "TOP" the text in the shape is adjusted to the top edge of the object. If the value "BOTTOM" is used it is the other way round.

```
xShapeProps~setProperty("TextVerticalAdjust",  
    bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "TOP"))  
  
xText = xrectangle~XText  
xTextCursor = xText~createTextCursor  
xTextCursor~gotoEnd(.false)  
xTextRange = xTextCursor~XTextRange  
xTextRange~setString("I am a rectangle!")
```

Another possibility to manipulate a text in an object works with the service "com.sun.star.drawing.TextHorizontalAdjust" as the program code shows below. With this service the horizontal position of the whole text can be changed. In this example two values are used. The first value "LEFT" sets the left edge of the text to the left edge of the object. With the value "RIGHT" the right edge of the text will be set to the right edge of the object [OOOAPIj].

```

/* text is displayed on the left edge of the shape*/
xShapeProps1~setPropertyValue("TextHorizontalAdjust",
    bsf.getConstant("com.sun.star.drawing.TextHorizontalAdjust", "LEFT"))

```

If no position of the text is set the text always appears in the middle of the shape.

The following figure displays the output of the example.

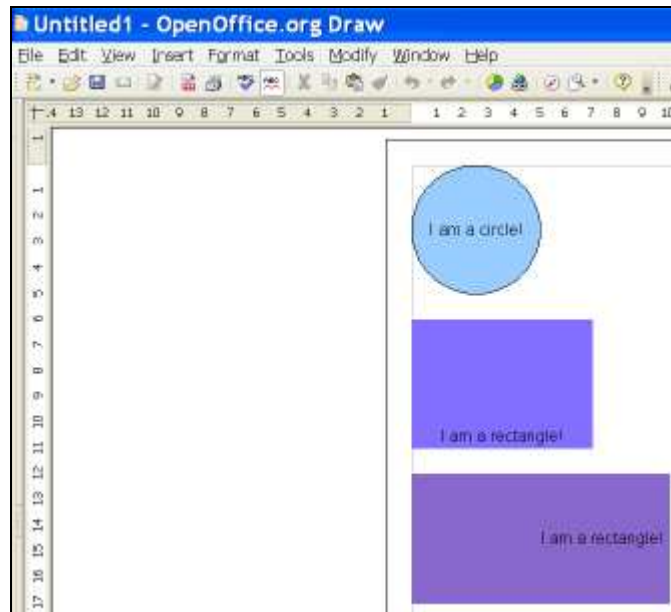


figure 14: Example 15 – text – adjust

3.2.2 Example 16 – text – fitproportional

This example shows the possibility to change the size of the text in an object with the service "com.sun.star.drawing.TextFitToSizeType". This service defines how the text in an object relates to the size of the object. With the value proportional the text size is scaled proportional to the size of the object. To change the size the text cursor is required like in the example 15.

The size of the text can be changed with setting the distance to the edges of the object. The higher the distance to the edges the smaller is the text size.

```

/* set the properties of the text in the rectangle */
xShapeProps~setPropertyValue("TextFitToSize",
    bsf.getConstant("com.sun.star.drawing.TextFitToSizeType", "PROPORTIONAL"))
xShapeProps~setPropertyValue("TextLeftDistance", box("int", 1000))
xShapeProps~setPropertyValue("TextRightDistance", box("int", 1000))

```

```
xShapeProps~setProperty("TextUpperDistance", box("int", 1000))
xShapeProps~setProperty("TextLowerDistance", box("int", 1000))
```

The following figure shows the output of the example.

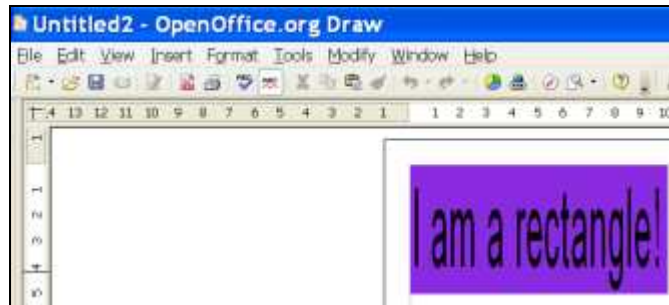


figure 15: Example 16 – text – fitproportional

3.2.3 Example 17 – text – animation

This example shows how to animate the text in an object. For the animation of a text the text cursor has to be set and the "com.sun.star.drawing.TextAnimationKind" service is needed. This service offers five ways to animate the text: no animation, let the text blink, scroll the text, scrolling the text from one edge of the object to the other and sliding the text to its end position. The program code below shows the possibility to let the text switch its state between visible and invisible rotational with the option "BLINK" [OOOAPIk].

```
/* set the properties of the circle */
xShapeProps~setProperty("TextAnimationKind",
  bsf.getConstant("com.sun.star.drawing.TextAnimationKind", "BLINK"))
```

In this example the text changes its animation after three seconds. With the command "ALTERNATE" the text in the object scrolls from one side to the other continuously [OOOAPIk].

The figure below shows a cutout of the output of this example.

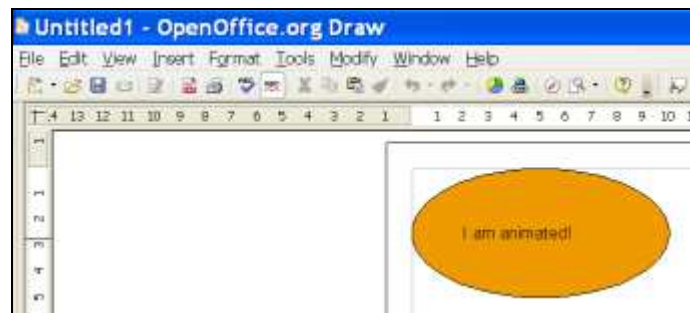


figure 16: Example 17 – text – animation

3.2.4 Example 18 – text – animationscroll

This example displays how the text animation scroll can be customized. First the text cursor has to be set and the service "com.sun.star.drawing.TextAnimationKind" has to be started and the value scroll has to be selected. Then the service "com.sun.star.drawing.TextAnimationDirection" can be used to specify the scroll direction. It is possible to let the text scroll from the bottom edge of the object to the top, from the right edge of the object to the left and the other way round. The program code below shows two different directions which switch after five seconds. If no direction is choosed the text scrolls from the right edge to the left edge of the object.

```
xShapeProps~setProperty("TextAnimationKind",
    bsf.getConstant("com.sun.star.drawing.TextAnimationKind", "SCROLL"))
xShapeProps~setProperty("TextAnimationDirection",
    bsf.getConstant("com.sun.star.drawing.TextAnimationDirection", "UP"))

call SysSleep 5
xShapeProps~setProperty("TextAnimationDirection",
    bsf.getConstant("com.sun.star.drawing.TextAnimationDirection", "RIGHT"))
```

The following figure shows the output of this example.

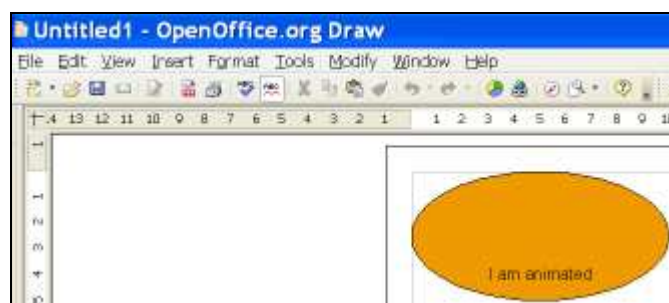


figure 17: Example 18 – text – animationscroll

3.2.5 Example 19 – text – properties

The following example shows how the properties of the text in an object can be changed.

First the text cursor has to be set and the `xPropertySet` interface has to be added to get access to the properties of the text of the object. Then the properties of the text can be changed. The program code below shows how to change the `CharColor`, `CharHeight`, `CharWeight` and the `CharUnderline` property. The `CharWeight` property can be modified with a constant from the `"com.sun.star.awt.FontWeight"` service. But the property `CharWeight` needs float number as value so the constant first has to be casted to float. To set the `CharUnderline` property the constant of the `"com.sun.star.awt.FontUnderline"` service has to be casted to short otherwise the constant cannot be used [OOOAPII].

```
cur = xTextCursor~xPropertySet
cur~setProperty("CharColor", box("int", "8968CD"x ~c2d)) -- set text colour purple
cur~setProperty("CharHeight", box("float", "30")) -- set font size 30
cur~setProperty("CharWeight", box("float", bsf.getConstant("com.sun.star.awt.FontWeight", "BOLD"))) -- set the font bold
cur~setProperty("CharUnderline", box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "DOUBLE"))) -- double underline the text
```

To cross out the text in an object the `CharCrossedOut` property has to be set to true. The font can be modified with the `CharFontName` property. If the `CharWordMode` is set to true the underline and the strike-through properties are not applied to white spaces [OOOAPII].

```
cur~setProperty("CharCrossedOut", box("boolean", "TRUE")) -- text is crossed out
cur~setProperty("CharFontName", "Times") -- change font to times new roman
cur~setProperty("CharWordMode", box("boolean", "TRUE")) -- underline and strike-through properties are not applied to white spaces
```

The program code below shows that a text shadow can be set when the `CharShadowed` property is set to true.

```
cur~setProperty("CharShadowed", box("boolean", "TRUE")) -- set a text shadow
```

The following figure shows the output of the example. First the text colour is changed to purple, the font size is set to 30, the font weight is set to bold and the text is underlined with a double line. Then the text is crossed out, the font

changes to Times New Roman and the CharWordMode is set true. After two seconds the text is not underlined and not crossed out and a text shadow is set.

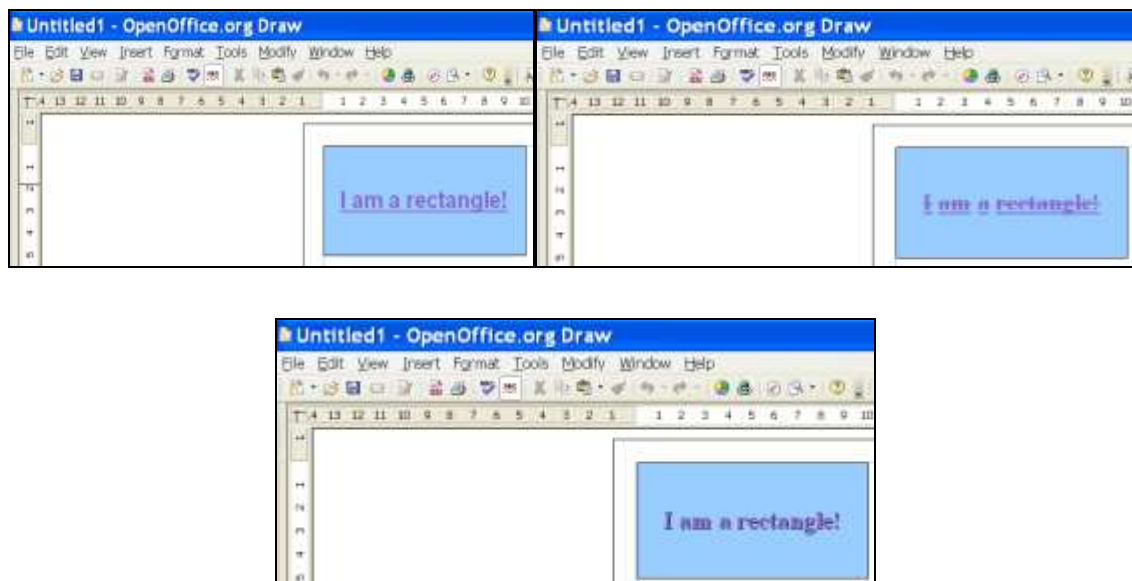


figure 18: Example 19 – text – properties

3.3 Draw Diagram Examples

In Open Office there is the possibility to draw automate diagrams. The following examples show how to automate diagrams.

3.3.1 Example 20 – diagram – connect

This example shows how to connect a shape with itself or with other shapes. Therefore an instance of the "com.sun.star.drawing.ConnectorShape" service has to be created, added to the shape and added to the xDrawPage.

```
xrectangle = xDocumentFactory ~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xrectangle1 = xDocumentFactory ~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xrectangle)
xDrawPage~add(xrectangle1)
```

After creating a connector shape the shapes can be connected by specifying where the line should start and end. In the program code below the start of the connecting line is the xrectangle and the end is the xrectangle1. Now it has to be stated on which edge of the rectangle the line should start and end. The start point is set with the StartGluePointIndex and the end point with the

EndGluePointIndex. The edges of the rectangle are numbered from the right edge to the top edge. So the right edge is specified with the number 1, the bottom edge with 2, the left edge with 3 and the top edge with 4 [OOOAPIm].

```
xprops = xrectangle~xPropertySet
xprops~setProperty("StartShape", xxrectangle)
xprops~setProperty("StartGluePointIndex", box("int", 2)) -- connect from the bottom line of the xxrectangle

xprops~setProperty("EndShape", xxrectangle1)
xprops~setProperty("EndGluePointIndex", box("int", 4)) -- connect to the top line of the xxrectangle1
```

The figure below shows some ways of connecting shapes which are applied in this example.

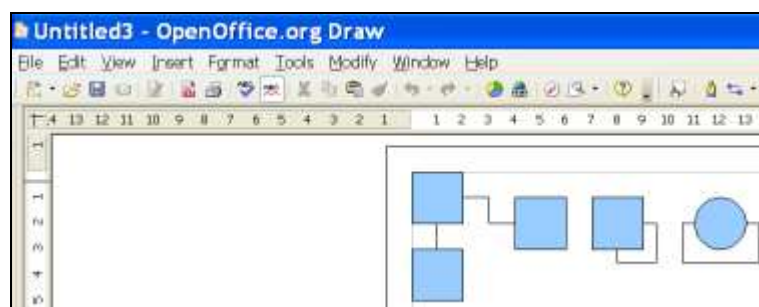


figure 19: Example 20 - diagram - connect

3.3.2 Example 21 – diagram

In this example a diagram can be created individually. The user can enter the desired parameters to the program as the following figure shows.

```
Please insert the title of the diagram:
Manager
*** Please insert the NUMBER of the SECOND organigram level nodes (max 4 nodes):
4
Please insert the NAME of the SECOND level node number 1:
Production
*** Please insert the NUMBER of the THIRD organigram level nodes (max 2 nodes):
2
*** Please insert the NAME of the THIRD level node number 1:
P1
*** Please insert the NAME of the THIRD level node number 2:
P2
Please insert the NAME of the SECOND level node number 2:
Sales
*** Please insert the NUMBER of the THIRD organigram level nodes (max 2 nodes):
2
*** Please insert the NAME of the THIRD level node number 1:
S1
*** Please insert the NAME of the THIRD level node number 2:
S2
Please insert the NAME of the SECOND level node number 3:
Marketing
*** Please insert the NUMBER of the THIRD organigram level nodes (max 2 nodes):
1
*** Please insert the NAME of the THIRD level node number 1:
M1
Please insert the NAME of the SECOND level node number 4:
Purchase
*** Please insert the NUMBER of the THIRD organigram level nodes (max 2 nodes):
0
```

figure 20: Example entry

The program reads the entered information from the user input and takes the number of the tree levels. The level number including the title row is limited to three layers but the user can decide if he likes to have only one level or more. If the user inserts zero for the number of the second level nodes only the first level (the title) is painted. For each level the user has to put in the name of the nodes and how many child nodes they should have themselves.

The names and parent nodes of each level are saved in an array. After the collection of the user input the diagram is painted. Therefore the rectangle shapes has to be created and connected to their parent nodes. For each entry in the array from the second level a rectangle is created. Therefore a loop iterates the array until the requested level nodes number is reached. The size and the position of the rectangles are variable. The x coordinates of the rectangles are calculated with the following formula: The length of the document (19000) plus an extra border of 1250 on the left side of the level nodes minus a factor where the rectangles should be placed. With the next (second) iteration the loop variable is increased and therefore the factor decreased. So the second element is placed more in direction of the right border. It is enough space to place up to four nodes on this second level in one row.

```

DO k=1 to orgnumber2
lv12nodelength=4000
  lv12nodewidth=2000
  --create the rectangles depending on how many should be created
  xfactorlv12=(5)-k

  lv12nodex=19000+1250-(xfactorlv12*4750)
  lv12nodey=8000

xrectanglelv12.k = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
xxrectanglelv12.k = xrectanglelv12.k~xshape
xxrectanglelv12.k~setPosition(.bsf~new("com.sun.star.awt.Point", lv12nodex, lv12nodey))
xxrectanglelv12.k~setSize(.bsf~new("com.sun.star.awt.Size", lv12nodelength, lv12nodewidth))
xDrawPage~add(xxrectanglelv12.k) -- adds the rectangle to the draw document

```

The building of the connection of the shapes is done in two steps. First the second level nodes are connected to the first level node (title). Therefore the created array with the information of the second level nodes has to be iterated. The nodes of the second level are always connected to the first level node (title) so the StartShape property of the ConnectorShape can be set fix to xrectangletitle. The EndShape has to be changed to the current iterated element of the second

level which should be also connected with the title node. The EndShape variable is for example xxrectanglelv12.k as the program code shows below.

```
xrectanglelv12.k = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xrectanglelv12.k)

xproplvl2.k = xrectanglelv12.k~xPropertySet

xproplvl2.k~setProperty("StartShape", xrectanglelv12.k)
xproplvl2.k~setProperty("StartGluePointIndex", box("int", 2))

xproplvl2.k~setProperty("EndShape", xxrectanglelv12.k)
xproplvl2.k~setProperty("EndGluePointIndex", box("int", 4))
```

To connect the second level nodes and the third level nodes the StartShape and the EndShape has to be changed for the rectangles in opposite to the title node before. So the StartShape and the EndShape changes for every rectangle that will be created as the program code shows below. In this example the current node is named with two loop variables, k and m, where k is the parent node (second level) and m the left or right child of k (the third level node).

```
lv13suffix=xxrectanglelv12.k

xrectanglelv13.k.m = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xrectanglelv13.k.m)

xproplvl3.k.m = xrectanglelv13.k.m~xPropertySet

xproplvl3.k.m~setProperty("StartShape", lv13suffix)
xproplvl3.k.m~setProperty("StartGluePointIndex", box("int", 2))

xproplvl3.k.m~setProperty("EndShape", xxrectanglelv13.k.m)
xproplvl3.k.m~setProperty("EndGluePointIndex", box("int", 4))
```

The following figure shows the output of the example input above. The font of each node is coloured purple, the font size gets smaller from level to level to allow each second level node to have two children and to offer enough place in the current row. The font of the first level is set to bold to mark this node as title element.

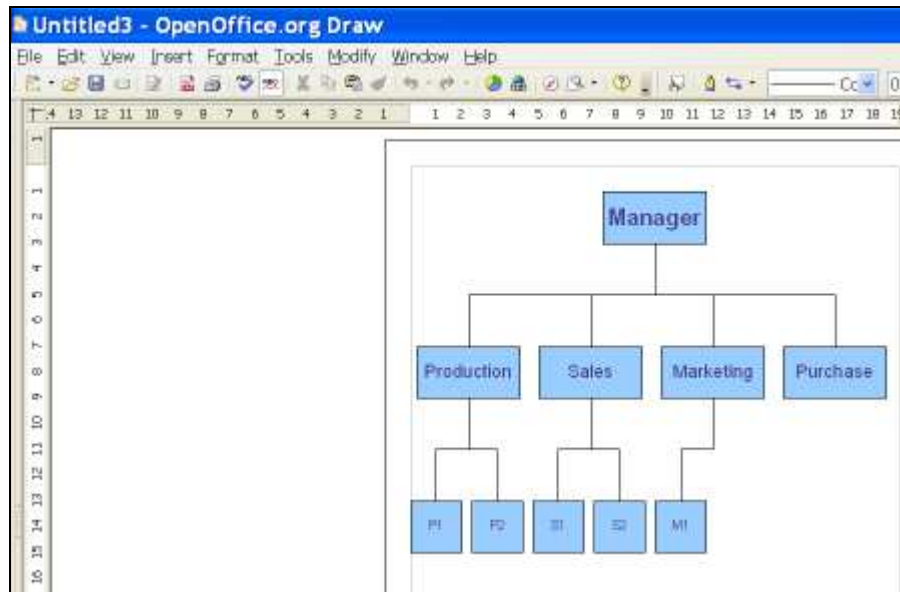


figure 21: Example 21 – diagram – three levels

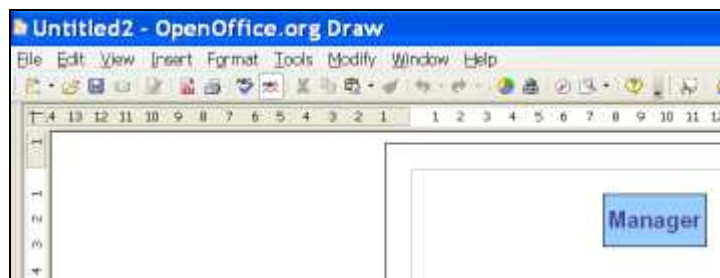


figure 22: Example 21 - diagram - one level

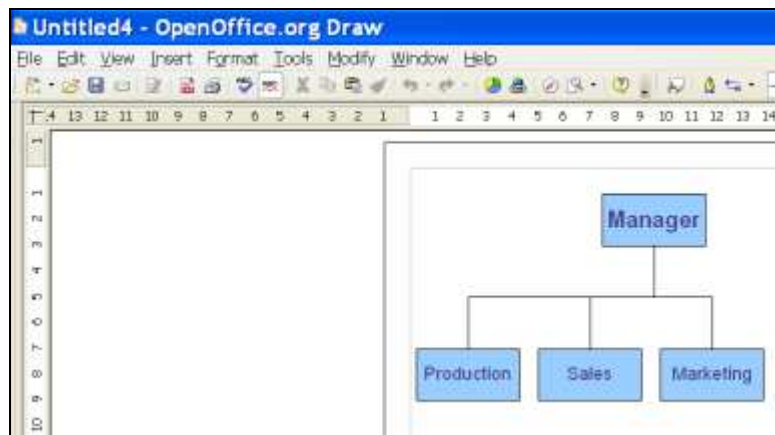


figure 23: Example 21 - diagram - two levels

4 Conclusion

The automation of Draw of Open Office.org is a great possibility to draw effectively and efficiently diagrams, objects or objects with text. There is a wide range of possibilities to write useful applications which can be used in job or for private use.

Particularly the needed programs for the automation are free available on internet and everyone can download and install them and start to write programs to automate functions they need.

Draw offers great variety of possible features to manipulate shapes and text within the shapes. One important thing to know for drawing a shape is that the x and y coordinate start in the left corner of the draw document and so the coordinates increase from left to right and from top to bottom. This is important to set the right position of the created shape. This is a good example how essential it is in programming to deal with the basics first and climb the ladder up step by step.

Another important thing is that the shape and the text is formatted or changed by using the properties of them. So before the objects can be manipulated the properties have to be available. It is essential to understand the system and the usage of the interfaces and methods. The usage of them is not always intuitive and trivial. People writing code for Open Office automation have to distinguish and recognize on which opportunities they have to use methods and when they have to solve the problem or reach their aim with properties.

These important hints to save time and efforts while handling such tasks can be found on developer guides or community forums. Also the documentation and API's can be a helpful to solve problems while programming with Object Rexx and when stuck in writing code for Open Office automation.

Summarizing it can be said that programming Draw Nutshells with Object Rexx is interesting, challenging and nice to see the programming skills developing and solving more difficult tasks.

5 References

- [OOOorga] About US: Open Office.org, 2007, <http://about.openoffice.org/index.html>, retrieved 20 May 2007
- [OOOorgb] Open Office.org 2 – Product Description, <http://www.openoffice.org/product/>, retrieved on 20 May 2007
- [ooRex07] Open Object Rexx – About Open Object Rexx, 2007, <http://www.oorexx.org/>, retrieved 20 May 2007
- [Flat06] Dr. Rony G. Flatscher, Automatisierungen von Java Anwendungen (7) – Course slides, 2006, <http://wi.wu-wien.ac.at/rgf/wu/lehre/autowin/material/folien/>, retrieved November 2006
- [OOOAPIa] Open Office.org, Open Office.org - Developers Guide, 28 October 2005, <http://api.openoffice.org/docs/DevelopersGuide/ProfUNO/ProfUNO.xhtml>, retrieved on 20 May 2007
- [OOOAPIb] Open Office.org, Open Office.org - Developers Guide, 28 October 2005, <http://api.openoffice.org/docs/DevelopersGuide/Drawing/Drawing.xhtml> , retrieved on 20 May 2007
- [OOOAPIc] Open Office.org API, service Desktop, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/frame/Desktop.html>, retrieved on 20 May 2007
- [OOOAPId] Open Office.org API, service DrawPage, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/DrawPage.html>, retrieved on 20 May 2007
- [OOOAPIe] Open Office.org API, service XDrawPageSupplier, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/XDrawPageSupplier.html>, retrieved on 20 May 2007

- [OOOAPIf] Open Office.org API, service RotationDescriptor, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/RotationDescriptor.html>, retrieved 24 May 2007
- [OOOAPIg] Open Office.org API, enum HatchStyle, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/HatchStyle.html>, retrieved 24 May 2007
- [OOOAPIh] Open Office.org API, enum LineJoint, 2003 <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/LineJoint.html>, retrieved 26. May 2007
- [OOOAPIi] Open Office.org API, enum TextVerticalAdjust, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/TextVerticalAdjust.html>, retrieved 26 May 2007
- [OOOAPIj] Open Office.org API, enum TextHorizontalAdjust, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/TextHorizontalAdjust.html>, retrieved 26 May 2007
- [OOOAPIk] Open Office.org API, enum TextAnimationKind, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/TextAnimationKind.html>, retrieved 27 May 2007
- [OOOAPIl] Open Office.org API, enum CharacterProperties, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/style/CharacterProperties.html>, retrieved 27 May 2007
- [OOOAPIm] Open Office.org API, service ConnectorShape, 2003, <http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/ConnectorShape.html>, retrieved 5 June 2007
- [Sun07] Sun Microsystems Documentation, Aufbau von Zeichnungen, 2007, <http://docs.sun.com/app/docs/doc/819-1326/6n3mlokui?l=de&a=view>, retrieved on 24 May 2007

6 Attachment

```

/* Example 01:
   paint 3 lines and change their shapes */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a line and add it to the shape */
Line = xDocumentFactory~createInstance("com.sun.star.drawing.LineShape")~xShape
Line~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 2000))
Line~setSize(.bsf~new("com.sun.star.awt.Size", 15000, 0))
xDrawPage~add(Line)
xShapeProps = Line~XPropertySet
xShapeProps~setProperty("LineStyle",
    bsf.getConstant("com.sun.star.drawing.LineStyle", "DASH")) -- set line style dash

/* create a line and add it to the shape */
xLine = xDocumentFactory~createInstance("com.sun.star.drawing.LineShape")~xShape
xLine~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 5000))
xLine~setSize(.bsf~new("com.sun.star.awt.Size", 15000, 1000))
xDrawPage~add(xLine)

/* create a line and add it to the shape */
xLine1 = xDocumentFactory~createInstance("com.sun.star.drawing.LineShape")~xShape
xLine1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 10000))
xLine1~setSize(.bsf~new("com.sun.star.awt.Size", 15000, 0))
xDrawPage~add(xLine1)

/* colour the 2nd and the 3rd line green */
call syssleep 1
xShapeProps = xLine~XPropertySet
xShapeProps~setProperty("LineColor", box("int", "228B22"x ~c2d)) -- paint the line green
xShapeProps~setProperty("LineWidth", box("int", "500")) -- set line width 5 millimeter

xShapeProps1 = xLine1~XPropertySet
xShapeProps1~setProperty("LineColor", box("int", "228B22"x ~c2d)) -- paint the line green
xShapeProps1~setProperty("LineWidth", box("int", "500")) -- set line width 5 millimeter

/* set the 3rd line transparent */
call sysleep 1
xShapeProps1~setProperty("LineTransparence", box("int", "80")) -- set transparency 80 per cent

::requires UNO.cls -- get UNO support

```

```

/* Example 02:
   create a line, a circle and a rectangle and add it to the shape */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
oDesktop      = UNO.createDesktop()  -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
               .UNO~noProps)
/* need document's factory to be able to insert created objects*/
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a line and add it to the shape */
xLine = xDocumentFactory~createInstance("com.sun.star.drawing.LineShape")~xShape
xLine~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 5000))
xLine~setSize(.bsf~new("com.sun.star.awt.Size", 19000, 0))
xDrawPage~add(xLine)

/* create a circle and add it to the shape */
xCircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xShape
xCircle~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 15000))
xCircle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xCircle)

/* create a rectangle and add it to the shape */
xBox = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xShape
xBox~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 8000))
xBox~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xBox)

::requires UNO.cls  -- get UNO support

```

```

/* Example 03:
   draw a rectangle and add it to the shape and then remove the rectangle from the shape */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop()  -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader
               -- interface
/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
               .UNO~noProps)
/* need document's factory to be able to insert created objects*/
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))

```

```
xDrawPage~add(xrectangle) -- add rectangle to the draw document

call sys.sleep 2

xDrawPage~remove(xrectangle) -- remove rectangle from the draw document

::requires UNO.cls -- get UNO support
```

```
/* Example 04:
   create a rectangle and change the fill colour and the line colour */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))

xDrawPage~add(xrectangle) -- add the rectangle to the draw document

/* colour the rectangle pink solid */
xShapeProps = xrectangle~XPropertySet
/* set fill colour pink */
xShapeProps~setProperty("FillColor", box("int", "ff 00 ff"x ~c2d))
/* set line colour blue */
xShapeProps~setProperty("LineColor", box("int", "00 00 ff"x ~c2d))

/* fill the rectangle */
xShapeProps~setProperty("FillStyle",
    bsf.getConstant("com.sun.star.drawing.FillStyle", "SOLID"))
xShapeProps~setProperty("LineWidth", box("int", "200")) -- set line width 2 millimeter
xShapeProps~setProperty("LineStyle",
    bsf.getConstant("com.sun.star.drawing.LineStyle", "DASH")) -- set line style dash

::requires UNO.cls -- get UNO support
```

```
/* Example 05:
   draw two rectangles, group them and then move them together */

/* initialize connection to server, get XContext */
xContext = UNO.connect() -- connect to server and retrieve the XContext object
XMcf = xContext~getServiceManager -- retrieve XMultiComponentFactory
```



```

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* draw a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 2500, 2500))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 2000, 2000))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document
xShapeProps = xrectangle1~XPropertySet
/* set fill colour pink */
xShapeProps~setProperty("FillColor", box("int", "ff 00 ff"x ~c2d))

/* group the 2 shapes */
/* get ShapeCollection from the XMultiComponentFactory */
xObj = xMcf~
    createInstanceWithContext("com.sun.star.drawing.ShapeCollection", xContext)
xToGroup = xObj~XShapes
xToGroup~add(xrectangle)
xToGroup~add(xrectangle1)
xShapeGrouper = xDrawPage~xShapeGrouper
xShapeGroup = xShapeGrouper~group(xToGroup)

call sys.sleep 1
xShapeGroup~setPosition(.bsf~new("com.sun.star.awt.Point", 7000, 7000)) -- move the group to another
position

::requires UNO.cls -- get UNO support

```

```

/* Example 06:
    create a rectangle, change the colour and set a corner radius */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)

/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

```

```

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* colour the rectangle blue */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "8DB6CD"x ~c2d))
xShapeProps~setProperty("LineColor", box("int", "8DB6CD"x ~c2d))

/* fill the rectangle */
xShapeProps~setProperty("FillStyle",
    bsf.getConstant("com.sun.star.drawing.FillStyle", "SOLID"))

xShapeProps~setProperty("CornerRadius", box("int", 500))
call sysssleep 2
xShapeProps~setProperty("CornerRadius", box("int", 1000))
call sysssleep 2
xShapeProps~setProperty("CornerRadius", box("int", 2000))

::requires UNO.cls -- get UNO support

```

```

/* Example 07:
   create two rectangles, change the colour and set a shear */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects*/
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 8000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* colour the rectangle dark green solid */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "458B74"x ~c2d)) -- set fill colour dark green

/* draw a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 9000, 10000))

```

```
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 2500, 2500))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document

/* colour the rectangle dark green solid */
xShapeProps1 = xrectangle1~XPropertySet
xShapeProps1~setProperty("FillColor", box("int", "458B74"x ~c2d)) -- set fill colour dark green

/* The shape is sheared counter-clockwise around the center of the bounding box */
call sys.sleep 1
xShapeProps~setProperty("ShearAngle", box("int", 2000))
call sys.sleep 1
xShapeProps1~setProperty("ShearAngle", box("int", 10000))

::requires UNO.cls -- get UNO support
```

```
/* Example 08:
   draw a rectangle, colour it, and let it rotate */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 5000, 3000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 7000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* colour the rectangle grey solid */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "D3D3D3"x ~c2d)) -- set fill colour grey

/* rotate the rectangle: counter-clockwise around the center of the bounding box */
call sys.sleep 1
xShapeProps~setProperty("RotateAngle", box("int", 10000))

::requires UNO.cls -- get UNO support
```

```
/* Example 09:
   paint two rectangles and set fill style of them gradient */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
```

```

oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document

/* colour the rectangle gradient linear */
xShapeProps1 = xrectangle1~XPropertySet

oGradient = .bsf~new("com.sun.star.awt.Gradient")
oGradient~Style = bsf.getConstant("com.sun.star.awt.GradientStyle", "LINEAR")
oGradient~StartColor = 8526139
oGradient~EndColor = 238210238
oGradient~Angle = 450 -- angle of 45 degrees
oGradient~StartIntensity = 150
oGradient~EndIntensity = 150
oGradient~StepCount = 100 -- gradient with 100 single colour steps

xShapeProps1~setProperty("FillStyle",
    bsf.getConstant("com.sun.star.drawing.FillStyle", "GRADIENT"))
xShapeProps1~setProperty("FillGradient", oGradient)

/* create a rectangle */
xrectangle2 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle2~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 7000))
xrectangle2~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle2) -- adds the rectangle to the draw document

/* colour the rectangle gradient radial*/
xShapeProps2 = xrectangle2~XPropertySet

o1Gradient = .bsf~new("com.sun.star.awt.Gradient")
o1Gradient~Style = bsf.getConstant("com.sun.star.awt.GradientStyle", "RADIAL")
o1Gradient~StartColor = 250235215
o1Gradient~EndColor = 255222173
o1Gradient~Angle = 500 -- angle of 50 degrees
o1Gradient~StartIntensity = 100
o1Gradient~EndIntensity = 50
o1Gradient~StepCount = 10 -- gradient with 10 single colour steps

xShapeProps2~setProperty("FillStyle",
    bsf.getConstant("com.sun.star.drawing.FillStyle", "GRADIENT"))
xShapeProps2~setProperty("FillGradient", o1Gradient)

::requires UNO.cls -- get UNO support

```

```

/* Example 10:
   draw three rectangles and set fill style hatch */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop()  -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
                                                         .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* colour the rectangle hatch single */
xShapeProps = xrectangle~XPropertySet
ohatch = .bsf~new("com.sun.star.drawing.Hatch")
ohatch~Style = bsf.getConstant("com.sun.star.drawing.HatchStyle", "SINGLE")
ohatch~Color      = 2554848
ohatch~Distance   = 1000 -- distance is 1 centimeter
ohatch~Angle      = 900 -- angle of 90 degrees

/* fill the rectangle */
xShapeProps~setProperty("FillStyle",
                       .bsf.getConstant("com.sun.star.drawing.FillStyle", "HATCH"))
xShapeProps~setProperty("FillHatch", ohatch)

/* draw a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 7000))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document

/* colour the rectangle hatch double */
xShapeProps = xrectangle1~XPropertySet

ohatch1 = .bsf~new("com.sun.star.drawing.Hatch")
ohatch1~Style = bsf.getConstant("com.sun.star.drawing.HatchStyle", "DOUBLE")
ohatch1~Color      = 13771137
ohatch1~Distance   = 50 -- distance is 0.5 millimeter
ohatch1~Angle      = 0 -- angle of 0 degrees

/* fill the rectangle */
xShapeProps~setProperty("FillStyle",
                       .bsf.getConstant("com.sun.star.drawing.FillStyle", "HATCH"))
xShapeProps~setProperty("FillHatch", ohatch1)

```

```

/* draw a rectangle */
xrectangle2 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle2~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 13000))
xrectangle2~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle2) -- adds the rectangle to the draw document

/* colour the rectangle hatch double */
xShapeProps = xrectangle2~XPropertySet

ohatch2 = .bsf~new("com.sun.star.drawing.Hatch")
ohatch2~Style = bsf.getConstant("com.sun.star.drawing.HatchStyle", "TRIPLE")
ohatch2~Color      = 2552150
ohatch2~Distance   = 500 -- distance is 5 millimeter
ohatch2~Angle      = 0 -- angle of 0 degrees

/* fill the rectangle */
xShapeProps~setProperty("FillStyle", bsf.getConstant("com.sun.star.drawing.FillStyle", "HATCH"))
xShapeProps~setProperty("FillHatch", ohatch2)

::requires UNO.cls -- get UNO support

```

```

/* Example 11:
   paint two rectangles, set a colour and set a transparency */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, UNO~noProps)

/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document

/* colour the rectangle black solid */
xShapeProps1 = xrectangle1~XPropertySet
xShapeProps1~setProperty("FillColor", box("int", "000000"x ~c2d)) -- set fill colour black
xShapeProps1~setProperty("LineColor", box("int", "ff 00 ff"x ~c2d)) -- set line colour pink

/* draw a rectangle */
xrectangle2 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle2~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 6000))
xrectangle2~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))

```

```
xDrawPage~add(xrectangle2) -- adds the rectangle to the draw document

/* colour the rectangle black solid */
xShapeProps2 = xrectangle2~XPropertySet
xShapeProps2~setPropertyvalue("FillColor", box("int", "000000"x ~c2d)) -- set fill colour black
xShapeProps2~setPropertyvalue("LineColor", box("int", "ff 00 ff"x ~c2d)) -- set line colour pink

call sysleep 1
xShapeProps1~setPropertyvalue("FillTransparence", box("int", "50")) -- set transparence 50 percent

call sysleep 1
xShapeProps2~setPropertyvalue("FillTransparence", box("int", "90")) -- set transparence 90 percent

::requires UNO.cls -- get UNO support
```

```
/* Example 12:
   draw three rectangles, set colour blue and change the line joint of the rectangles */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
                                                       .UNO~noProps)

/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* change the colour properties */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setPropertyvalue("FillColor", box("int", "191970"x ~c2d)) -- set fill colour blue
xShapeProps~setPropertyvalue("LineColor", box("int", "48D1CC"x ~c2d)) -- set line colour turquoise
xShapeProps~setPropertyvalue("LineWidth", box("int", "200")) -- set line width 20 millimeter
xShapeProps~setPropertyvalue("LineJoint",
                             .bsf.getConstant("com.sun.star.drawing.LineJoint", "MITER")) -- the lines join at intersections

/* draw a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 7000))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document

/* change the colour properties */
xShapeProps1 = xrectangle1~XPropertySet
xShapeProps1~setPropertyvalue("FillColor", box("int", "191970"x ~c2d)) -- set fill colour blue
xShapeProps1~setPropertyvalue("LineColor", box("int", "48D1CC"x ~c2d)) -- set line colour turquoise
```



```

xShapeProps1~setProperty("LineWidth", box("int", "700")) -- set line width 7 millimeter
xShapeProps1~setProperty("LineTransparency", box("int", "80")) -- set transparency 80 per cent
xShapeProps1~setProperty("LineJoint",
    bsf.getConstant("com.sun.star.drawing.LineJoint", "BEVEL")) -- edges of the lines are joined by
lines

/* draw a rectangle */
xrectangle2 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle2~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 14000))
xrectangle2~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))

xDrawPage~add(xrectangle2) -- adds the rectangle to the draw document

/* change the colour properties */
xShapeProps2 = xrectangle2~XPropertySet
xShapeProps2~setProperty("FillColor", box("int", "191970"x ~c2d)) --set fill colour blue
xShapeProps2~setProperty("LineColor", box("int", "48D1CC"x ~c2d)) -- set line colour turquoise
xShapeProps2~setProperty("LineWidth", box("int", "1000")) -- set line width 1 centimeter
xShapeProps2~setProperty("LineTransparency", box("int", "50")) -- set transparency 50 per cent
xShapeProps2~setProperty("LineJoint",
    bsf.getConstant("com.sun.star.drawing.LineJoint", "NONE")) -- the joint of the lines is not
connected

::requires UNO.cls -- get UNO support

```

```

/* Example 13:
   paint two rectangles, change the colour, set a shadow and set shadow transparency */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 7000, 5000))
xDrawPage~add(xrectangle) -- adds the rectangle to the draw document

/* colour the rectangle blue solid and set shadow */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("Shadow", box("boolean", "True" )) -- set shadow property true
xShapeProps~setProperty("FillColor", box("int", "009ACD"x ~c2d )) -- set fill colour blue
xShapeProps~setProperty("LineColor", box("int", "7B68EE"x ~c2d)) -- set line colour blue

/* draw a rectangle */

```



```

xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 7000))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 7000, 5000))
xDrawPage~add(xrectangle1) -- adds the rectangle to the draw document

/* colour the rectangle blue solid and set shadow */
xShapeProps1 = xrectangle1~XPropertySet
xShapeProps1~setProperty("Shadow", box("boolean", "True" ))
xShapeProps1~setProperty("FillColor", box("int", "009ACD"x ~c2d )) -- set fill colour blue
xShapeProps1~setProperty("LineColor", box("int", "7B68EE"x ~c2d)) -- set line colour blue

/* customize shadow */
xShapeProps1~setProperty("ShadowColor", box("int", "000000"x ~c2d )) -- set the shadow colour
black

call sysleep 2
/* set the shadow distance */
xShapeProps1~setProperty("ShadowXDistance", box("int", "2000"))
xShapeProps1~setProperty("ShadowYDistance", box("int", "2000"))

call sysleep 1
xShapeProps1~setProperty("ShadowTransparence", box("int", "90")) -- set transparency to 90 per
cent

::requires UNO.cls -- get UNO support

```

```

/* Example 14:
   paint three circles, change the colour and the shape of the circles */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
oDesktop      = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
                                                         .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a circle and add it to the shape */
xCircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xShape
xCircle~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 2000))
xCircle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xCircle)

/* create a circle and add it to the shape */
xCircle1 = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xShape
xCircle1~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 8000))
xCircle1~setSize(.bsf~new("com.sun.star.awt.Size", 8000, 5000))
xDrawPage~add(xCircle1)

/* create a circle and add it to the shape */
xCircle2 = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xShape

```

```

xCircle2~setPosition(.bsf~new("com.sun.star.awt.Point", 2000, 15000))
xCircle2~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 7000))
xDrawPage~add(xCircle2)

/* set the colours of the circles */
xShapeProps1 = xCircle1~XPropertySet
xShapeProps1~setProperty("FillColor", box("int", "FF7F24"x ~c2d)) -- colour the 1st circle orange
xShapeProps = xCircle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "8B2323"x ~c2d)) -- colour the 2nd circle dark red
xShapeProps2 = xCircle2~XPropertySet
xShapeProps2~setProperty("LineColor", box("int", "8B4513"x ~c2d)) -- colour the 3rd circle line brown

/* change the shape of the circles */
call sysleep 1
/* 1st circle: show the cut circle connected by 2 lines */
xShapeProps~setProperty("CircleStartAngle", box("int", "2000")) -- start angle 20 degrees
xShapeProps~setProperty("CircleEndAngle", box("int", "9000")) -- end angle 90 degrees
xShapeProps~setProperty("CircleKind", $
    bsf.getConstant("com.sun.star.drawing.CircleKind", "SECTION"))

call sysleep 1
/* 2nd circle: show the cut circle connectet by 1 line */
xShapeProps1~setProperty("CircleStartAngle", box("int", "10000")) -- start angle 100 degrees
xShapeProps1~setProperty("CircleEndAngle", box("int", "18000")) -- end angle 180 degrees
xShapeProps1~setProperty("CircleKind", $
    bsf.getConstant("com.sun.star.drawing.CircleKind", "CUT"))

call sysleep 1
/* 3rd circle: show the circle with an open cut */
xShapeProps2~setProperty("CircleStartAngle", box("int", "1000")) -- start angle 10 degrees
xShapeProps2~setProperty("CircleEndAngle", box("int", "18000")) -- end angle 180 degrees
xShapeProps2~setProperty("CircleKind", $
    bsf.getConstant("com.sun.star.drawing.CircleKind", "ARC"))

::requires UNO.cls -- get UNO support

```

```

/* Example 15:
   create three objects, add a text and adjust the text */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, $
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a circle and add it to the shape */
xCircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xShape

```

```

xCircle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xCircle~setSize(.bsf~new("com.sun.star.awt.Size", 5000, 5000))
xDrawPage~add(xCircle)

xCircle~xText~setString("I am a circle!") -- add text to the circle

/* create a Rectangle and add it to the shape */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 7000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 7000, 5000))
xDrawPage~add(xrectangle)

/* set the properties of the rectangle shape */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "836FFF"x ~c2d))
xShapeProps~setProperty("LineColor", box("int", "836FFF"x ~c2d))
/* text is displayed on the top edge of the shape*/
xShapeProps~setProperty("TextVerticalAdjust",
    bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "TOP"))

xText = xrectangle~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString("I am a rectangle!")

/* create a Rectangle and add it to the shape */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 13000))
xrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 10000, 5000))
xDrawPage~add(xrectangle1)

/* set the properties of the rectangle shape */
xShapeProps1 = xrectangle1~XPropertySet
xShapeProps1~setProperty("FillColor", box("int", "8968CD"x ~c2d))
xShapeProps1~setProperty("LineColor", box("int", "8968CD"x ~c2d))
/* text is displayed on the left edge of the shape*/
xShapeProps1~setProperty("TextHorizontalAdjust",
    bsf.getConstant("com.sun.star.drawing.TextHorizontalAdjust", "LEFT"))

xText = xrectangle1~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString("I am a rectangle!")

call sysssleep 1
/* text is displayed on the bottom edge of the shape*/
xShapeProps~setProperty("TextVerticalAdjust",
    bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "BOTTOM"))

call sysssleep 1
/* text is displayed on the right edge of the shape*/
xShapeProps1~setProperty("TextHorizontalAdjust",
    bsf.getConstant("com.sun.star.drawing.TextHorizontalAdjust", "RIGHT"))

::requires UNO.cls -- get UNO support

```

```

/* Example 16:
   draw a rectangle, add a text and change the size of the text proportional to the shape */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop()  -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
                                                         .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a Rectangle and add it to the shape */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 10000, 5000))
xDrawPage~add(xrectangle)

/* set the properties of the rectangle shape */
xShapeProps = xrectangle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "8A2BE2"x ~c2d)) -- set fill colour purple
xShapeProps~setProperty("LineColor", box("int", "8A2BE2"x ~c2d)) -- set fill colour purple

/* set the properties of the text in the rectangle */
xShapeProps~setProperty("TextFitToSize",
                       .bsf~getConstant("com.sun.star.drawing.TextFitToSizeType", "PROPORTIONAL"))
xShapeProps~setProperty("TextLeftDistance", box("int", 1000))
xShapeProps~setProperty("TextRightDistance", box("int", 1000))
xShapeProps~setProperty("TextUpperDistance", box("int", 1000))
xShapeProps~setProperty("TextLowerDistance", box("int", 1000))

xText = xrectangle~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString("I am a rectangle!")

call Sys.sleep 1
xShapeProps~setProperty("TextFitToSize",
                       .bsf~getConstant("com.sun.star.drawing.TextFitToSizeType", "PROPORTIONAL"))
xShapeProps~setProperty("TextLeftDistance", box("int", 100))
xShapeProps~setProperty("TextRightDistance", box("int", 100))
xShapeProps~setProperty("TextUpperDistance", box("int", 100))
xShapeProps~setProperty("TextLowerDistance", box("int", 100))

::requires UNO.cls -- get UNO support

```

```

/* Example 17:
   draw a circle, add text and animate the text */

```

```

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop()  -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)

/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a Rectangle and add it to the shape */
xcircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xshape
xcircle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xcircle~setSize(.bsf~new("com.sun.star.awt.Size", 10000, 5000))
xDrawPage~add(xcircle)

/* set the properties of the rectangle shape */
xShapeProps = xcircle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "EE9A00"x ~c2d)) -- set fill colour orange

/* set the properties of the circle */
xShapeProps~setProperty("TextAnimationKind",
    bsf.getConstant("com.sun.star.drawing.TextAnimationKind", "BLINK"))

xText = xcircle~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString("I am animated!")

call sysssleep 3
xShapeProps~setProperty("TextAnimationKind",
    bsf.getConstant("com.sun.star.drawing.TextAnimationKind", "ALTERNATE"))

::requires UNO.cls  -- get UNO support

```

```

/* Example 18:
   draw a circle, add text and animate the text */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop      = UNO.createDesktop()  -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)

/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

```

```

/* create a Rectangle and add it to the shape */
xcircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")~xshape
xcircle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xcircle~setSize(.bsf~new("com.sun.star.awt.Size", 10000, 5000))
xDrawPage~add(xcircle)

/* set the properties of the rectangle shape */
xShapeProps = xcircle~XPropertySet
xShapeProps~setProperty("FillColor", box("int", "EE9A00"x ~c2d)) -- set fill colour orange

/* set the properties of the circle */
xText = xcircle~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString("I am animated")

xShapeProps~setProperty("TextAnimationKind",
    bsf.getConstant("com.sun.star.drawing.TextAnimationKind", "SCROLL"))
xShapeProps~setProperty("TextAnimationDirection",
    bsf.getConstant("com.sun.star.drawing.TextAnimationDirection", "UP"))

call Sys.sleep 5
xShapeProps~setProperty("TextAnimationDirection",
    bsf.getConstant("com.sun.star.drawing.TextAnimationDirection", "RIGHT"))

::requires UNO.cls -- get UNO support

```

```

/* Example 19:
   draw a rectangle, add text and change the text properties */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface

/* open the blank file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* create a Rectangle and add it to the shape */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")~xshape
xrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 10000, 5000))
xDrawPage~add(xrectangle)

xText = xrectangle~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString("I am a rectangle!")

```

```

cur = xTextCursor~xPropertySet
cur~setProperty("CharColor", box("int", "8968CD"x ~c2d)) -- set text colour purple
cur~setProperty("CharHeight", box("float", "30")) -- set font size 30
cur~setProperty("CharWeight", box("float", bsf.getConstant("com.sun.star.awt.FontWeight", "BOLD"))) -- set the font bold
cur~setProperty("CharUnderline", box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "DOUBLE"))) -- double underline
the text

Call sys.sleep 2
cur~setProperty("CharCrossedOut", box("boolean", "TRUE")) -- text is crossed out
cur~setProperty("CharFontName", "Times") -- change font to times new roman
cur~setProperty("CharWordMode", box("boolean", "TRUE")) -- underline and strike-through
properties are not applied to white spaces

Call sys.sleep 2
cur~setProperty("CharCrossedOut", box("boolean", "FALSE"))
cur~setProperty("CharUnderline", box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "NONE"))) -- text is not underlined
cur~setProperty("CharShadowed", box("boolean", "TRUE")) -- set a text shadow

::requires UNO.cls -- get UNO support

```

```

/* Example 20:
   three rectangles and connect them with a line */

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document*/
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface */
/* open the blank *.sxd - file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

/* draw a rectangle */
xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
xxrectangle = xrectangle~xshape
xxrectangle~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 1000))
xxrectangle~setSize(.bsf~new("com.sun.star.awt.Size", 2000, 2000))
xDrawPage~add(xxrectangle) -- adds the rectangle to the draw document

/* draw a rectangle */
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
xxrectangle1 = xrectangle1~xshape
xxrectangle1~setPosition(.bsf~new("com.sun.star.awt.Point", 1000, 4000))
xxrectangle1~setSize(.bsf~new("com.sun.star.awt.Size", 2000, 2000))
xDrawPage~add(xxrectangle1) -- adds the rectangle to the draw document

xrectangle = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xrectangle1 = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape

```



```
xDrawpage~add(xrectangle)
xDrawPage~add(xrectangle1)

xprops = xrectangle~xPropertySet
xprops~setProperty("StartShape", xrectangle)
xprops~setProperty("StartGluePointIndex", box("int", 2)) -- connect from the bottom line of the
xrectangle

xprops~setProperty("EndShape", xrectangle1)
xprops~setProperty("EndGluePointIndex", box("int", 4)) -- connect to the top line of the
xrectangle1

/* draw a rectangle */
xrectangle2 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
xxrectangle2 = xrectangle2~xshape
xxrectangle2~setPosition(.bsf~new("com.sun.star.awt.Point", 8000, 2000))
xxrectangle2~setSize(.bsf~new("com.sun.star.awt.Size", 2000, 2000))
xDrawPage~add(xxrectangle2) -- adds the rectangle to the draw document

xrectangle2 = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xrectangle2)

xprops2 = xrectangle2~xPropertySet
xprops2~setProperty("StartShape", xxrectangle2)
xprops2~setProperty("StartGluePointIndex", box("int", 1)) -- connect from the right edge of the
xxrectangle2

xprops2~setProperty("EndShape", xxrectangle2)
xprops2~setProperty("EndGluePointIndex", box("int", 2)) -- connect to the left edge of the
xxrectangle2

/* draw a rectangle */
xrectangle3 = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
xxrectangle3 = xrectangle3~xshape
xxrectangle3~setPosition(.bsf~new("com.sun.star.awt.Point", 5000, 2000))
xxrectangle3~setSize(.bsf~new("com.sun.star.awt.Size", 2000, 2000))
xDrawPage~add(xxrectangle3) -- adds the rectangle to the draw document

xrectangle3 = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xrectangle3)

xprops3 = xrectangle3~xPropertySet
xprops3~setProperty("StartShape", xxrectangle3)
xprops3~setProperty("StartGluePointIndex", box("int", 4))

xprops3~setProperty("EndShape", xxrectangle)
xprops3~setProperty("EndGluePointIndex", box("int", 1))

/* create a circle and add it to the shape */
xCircle = xDocumentFactory~createInstance("com.sun.star.drawing.EllipseShape")
xxcircle = xcircle~xShape
xxCircle~setPosition(.bsf~new("com.sun.star.awt.Point", 12000, 2000))
xxCircle~setSize(.bsf~new("com.sun.star.awt.Size", 2000, 2000))
xDrawPage~add(xxCircle)

xcircle = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xcircle)
```



```
xprops4 = xcircle~xPropertySet
xprops4~setPropertyValue("StartShape", xxcircle)
xprops4~setPropertyValue("StartGluePointIndex", box("int", 1))

xprops4~setPropertyValue("EndShape", xxcircle)
xprops4~setPropertyValue("EndGluePointIndex", box("int", 3))

::requires UNO.cls -- get UNO support
```

```
/* Example 21:
   create a diagram */

titleinput:
say "Please insert the title of the diagram: "
parse pull title
orgtitle = title

SELECT
when orgtitle == "" THEN do
    say "No title specified. Please insert the TITLE of the diagram: "
    CALL titleinput
END
OTHERWISE
END

/* Retrieve the Desktop object, we need its XComponentLoader interface to load a new document */
oDesktop = UNO.createDesktop() -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface */

/* open the blank *.sxd - file */
url = "private:factory/sdraw"
xDrawComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0,
    .UNO~noProps)
/* need document's factory to be able to insert created objects */
xDocumentFactory = xDrawComponent~XMultiServiceFactory

/* get draw page by index */
xDrawPage = xDrawComponent~XDrawPagesSupplier~getDrawPages~getByIndex(0)~XDrawPage

level2input:
say "*** Please insert the NUMBER of the SECOND organigram level nodes (max 4 nodes):"
parse pull number
orgnumber2 = number

SELECT
when orgnumber2 == 0 | orgnumber2 == 1 | orgnumber2 == 2 | orgnumber2 == 3 | orgnumber2 == 4
    THEN do
        orgname2 = .array~new(orgnumber2)
        do i = 1 to orgnumber2 by 1
            say " "
            say " Please insert the NAME of the SECOND level node number" i ":"
```

```

parse pull name
orgname2[i] = name
-- Third Level
say "**** Please insert the NUMBER of the THIRD organigram level nodes (max 2
nodes):"

parse pull number3
orgnumber3 = number3

SELECT
when orgnumber3 == 0 | orgnumber3 == 1 | orgnumber3 == 2
THEN do
orgname3.i = .array~new(orgnumber3)
orgparents3.i = i
do j = 1 to orgnumber3
say "**** Please insert the NAME of the THIRD level node number" j":"
parse pull name3
orgname3.i[j] = name3
end
END
OTHERWISE
SAY "Invalid input - ending program!"
EXIT
END

end
END
OTHERWISE say "No valid number. Please insert a number: "
CALL level2input
END

```

-- Create Title Node place it in middle of the document

```

titlenodelength=4000
titlenodewidth=2000
--Center over the width of the page, mid point of the sheet - length of rectangle +1000 left frame
space
titlenodex=((19000/2)-(titlenodelength/2))+1000
titlenodey=2000

xrectangletitle = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
xxrectangletitle = xrectangletitle ~xshape
xxrectangletitle ~setPosition(.bsf~new("com.sun.star.awt.Point", titlenodex, titlenodey))
xxrectangletitle ~setSize(.bsf~new("com.sun.star.awt.Size", titlenodelength, titlenodewidth))
xDrawPage~add(xxrectangletitle) -- adds the rectangle to the draw document

xText = xrectangletitle~XText
xTextCursor = xText~createTextCursor
xTextCursor~gotoEnd(.false)
xTextRange = xTextCursor~XTextRange
xTextRange~setString(orgtitle)
cur = xTextCursor~xPropertySet
cur~setProperty("CharColor", box("int", "483D8B"x ~c2d)) -- set text colour blue
cur~setProperty("CharHeight", box("float", "25")) -- set font size 25
cur~setProperty("CharWeight",
box("float", bsf.getConstant("com.sun.star.awt.FontWeight", "BOLD"))) -- set the font bold
--xrectangletitle ~xText~setString(orgtitle)

z=1

```

```

-- Navigates through the lvl2 Node-Array and created the nodes, max 4
DO k=1 to orgnumber2
  lvl2nodelength=4000
  lvl2nodewidth=2000
  --create the rectangles depending on how many should be created
  xfactorlvl2=(5)-k

  lvl2nodex=19000+1250-(xfactorlvl2*4750)
  lvl2nodey=8000

  xrectanglelvl2.k = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape")
  xxrectanglelvl2.k = xrectanglelvl2.k~xshape
  xxrectanglelvl2.k~setPosition(.bsf~new("com.sun.star.awt.Point", lvl2nodex, lvl2nodey))
  xxrectanglelvl2.k~setSize(.bsf~new("com.sun.star.awt.Size", lvl2nodelength, lvl2nodewidth))
  xDrawPage~add(xxrectanglelvl2.k) -- adds the rectangle to the draw document

  xText = xrectanglelvl2.k~XText
  xTextCursor1 = xText~createTextCursor
  xTextCursor1~gotoEnd(.false)
  xTextRange1 = xTextCursor1~XTextRange
  xTextRange1~setString(orgname2[k])
  cur1 = xTextCursor1~xPropertySet
  cur1~setProperty("CharColor", box("int", "483D8B"x ~c2d)) -- set text colour blue
  cur1~setProperty("CharHeight", box("float", "20")) -- set font size 20

  xrectanglelvl2.k = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
ape
  xDrawpage~add(xrectanglelvl2.k)

  xproplvl2.k = xrectanglelvl2.k~xPropertySet

  xproplvl2.k~setProperty("StartShape", xrectangletitle)
  xproplvl2.k~setProperty("StartGluePointIndex", box("int", 2))

  xproplvl2.k~setProperty("EndShape", xxrectanglelvl2.k)
  xproplvl2.k~setProperty("EndGluePointIndex", box("int", 4))

-- Navigates through the lvl3 Node-Array and created the nodes, max 2

  DO m=1 to orgname3.k~items

  lvl3parentnode=orgparents3.k
  lvl3suffix=xxrectanglelvl2.k

  lvl3nodelength=2000
  lvl3nodewidth=2000

  --create the rectangles depending on how many should be created
  xfactorlvl3=(9)-z

  lvl3nodex=19000+1000-(xfactorlvl3*2375)
  lvl3nodey=14000

  xrectanglelvl3.k.m = xDocumentFactory~createInstance("com.sun.star.drawing.RectangleShape
")
  xxrectanglelvl3.k.m = xrectanglelvl3.k.m~xshape
  xxrectanglelvl3.k.m~setPosition(.bsf~new("com.sun.star.awt.Point", lvl3nodex, lvl3nodey))
  xxrectanglelvl3.k.m~setSize(.bsf~new("com.sun.star.awt.Size", lvl3nodelength, lvl3nodewidth))
  xDrawPage~add(xxrectanglelvl3.k.m) -- adds the rectangle to the draw document

```

```
xText = xxrectanglelv13.k.m~XText
xTextCursor3 = xText~createTextCursor
xTextCursor3~gotoEnd(.false)
xTextRange3 = xTextCursor3~XTextRange
xTextRange3~setString(orgname3.k[m])
cur3 = xTextCursor3~xPropertySet
cur3~setProperty("CharColor", box("int", "483D8B"x ~c2d)) -- set text colour blue
cur3~setProperty("CharHeight", box("float", "14")) -- set font size 14

xrectanglelv13.k.m = xDocumentFactory~createInstance("com.sun.star.drawing.ConnectorShape")~xshape
xDrawpage~add(xrectanglelv13.k.m)

xproplv13.k.m = xrectanglelv13.k.m~xPropertySet

xproplv13.k.m~setProperty("StartShape", lv13suffix)
xproplv13.k.m~setProperty("StartGluePointIndex", box("int", 2))

xproplv13.k.m~setProperty("EndShape", xxrectanglelv13.k.m)
xproplv13.k.m~setProperty("EndGluePointIndex", box("int", 4))
z=z+1

END
END

::requires UNO.cls -- get UNO support
```