EMF2LEO Useful Documentation

Contents

2. General Information

EMF2LEO is a command-line tool used to convert vector graphics into the Leo-scripting- l anguage. Vector graphics that should be converted have to be stored in the Windows Enhanced Metafile format (EMF). Every modern graphic program is able to save graphics in this format. Nevertheless, vector graphic programs like Corel Draw 9 are more likely to produce good results than e.g. Paint Shop Pro. After all, the emf2leo converter facilitates the work of the ADOxx® customising staff in the way, that simple figures can be drawn outside of the ADOxx® Administration Toolkit and then be easily converted into the Leo-scripting-language. You should be aware that there will still be some work to do after having converted the Enhanced Metafile into Leo. EMF2LEO's output is a Leo-file whose content must be pasted into in the GraphRep of a class attribute of a ADOxx® standard library. Then the converted graphic elements have to be adapted to the customers' expectations. Due to the different ways of storing an image in an Enhanced Metafile the usability of the Leo-script can vary a lot. In some cases, very good results are obtained. In other cases, the results only represent a better starting base for the customising staff. In the following passages, it will be shown how to obtain the best possible results while working with EMF2LEO. At first, the structure of an Enhanced Metafile is discussed in detail.

2.1. The Windows Enhanced Metafile format (EMF)

The Enhanced Metafile format is used by Microsoft® to provide a portable application independent specification for storing images. Images can be stored in a metafile by using the Win32-GDI which is a part of the Windows-API. An image created by a vector based painting program consists of polygons, lines, ellipses, rectangles, arcs etc. When combined, those geometric figures can form new figures and you can draw almost every imaginable thing by using them. The figures created are drawn with a pen, who has a specified colour, width and style. So if the pen has a red colour attribute, all figures drawn after the definition of the pen are outlined by a red line. Figures can also be filled with colour.

A metafile is a list of records which contain the information required to draw e.g. a polygon or a line. There exist about 100 different record types. The different records are discussed in detail later. In every metafile, you find the same first record, called "header" which stores general information about the metafile, e.g. the overall size of the file, the number of records and the dimensions of the device on which the metafile has been created. The last record in an enhanced metafile is always the EOF (end of file) record. Between those two records, all other records are found. Their types vary according to the information they store. E.g. They contain information which geometrical figures have to be drawn and which pens and brushes have to be set. In fact, an Windows Enhanced Metafile is a static list of instructions. The painting operations are carried out in the same order as they are found in the metafile.

The Leo-scripting-language has some limitations that do not allow the use of all features normally included in a metafile. So, you should take care of the instructions describing how to obtain the best results when you want to use EMF2LEO.

2.2. The records of an EMF

As described above, there are different types of records. In the following list, you find an almost complete list of records and a brief description. All records containing relevant drawing information are identified by a grey background. These records are converted into Leo.

No	Name	Leo	Description / Corresponding Leo-element
1	Header		file header, contains general information and is the first record in every metafile
2	PolyBezier (32 Bit variant)		used to store information on drawing multiple bezier curves
3	Polygon (32 Bit variant)	POLYGON	draws a polygon, outlined with the current pen and filled with current brush
4	Polyline (32 Bit variant)	POLYLINE	draws multiple connected line segments using the current pen
5	PolyBezierTo (32 Bit variant)		draws multiple bezier curves, starting on the current position, the current position is then set to the ending point of the last
6	PolyLineTo (32 Bit variant)	POLYLINE	draws multiple connected line segments using the current pen and updates the current position
7	PolyPolyLine (32 Bit variant)	POLYLINE	draws at least one polyline, normally a whole set of polylines is
8	PolyPolyGon (32 Bit variant)	POLYGON	draws at least one polygon, normally a whole set of polygons is
9	SETWINDOWEXTEX		information about the window extensions
10	SETWINDOWORGEX		which logical point maps to the point (0/0) in device units
11	SETVIEWPORTEXTEX		information on the extensions of the visible region of the window
12	SETVIEWPORTORGEX		which point in device units is identical with the upper left corner of the visible window region
13	SETBRUSHORGEX		sets the brush origin that GDI assigns to the next brush an application selects into the specified device context
14	EOF		last record in every EMF-File
15	SETPIXELV	POINT	a pixel is set to the specified position using the current pen
16	SETMAPPERFLAGS		how logical fonts are mapped to physical fonts
17	SETMAPMODE		which measuring units were used by the device that created the EMF, the mapping mode determines the orientation of the x and y
18	SETBKMODE		background colour of the current clipping region
19	SETPOLYFILLMODE		how polygons are filled using the current brush
20	SETROP2		how are colours, which are already present in the clipping region, combined with the colour of the pen or brush
21	SETSTRETCHBLTMODE		how are pixels of the new image combined with the pixels in the clipping region.
22	SETTEXTALIGN	TEXT	how is text aligned (centred, left, right)
23	SETCOLORADJUSTMENT		how are colours adjusted (brightness, contrast)

24	SETTEXTCOLOR	FONT	sets the colour for text
25	SETBKCOLOR		sets the background colour of the clipping region
26	OFFSETCLIPRGN		position of the clipping region is offset by the number of specified
27	MOVETOEX		current position is set to a new point (e.g. starting point for lines)
28	SETMETARGN		intersects the current clipping region for the specified device context with the current metaregion and saves the combined region as the new metaregion for the specified device context
29	EXCLUDECLIPRECT		creates a new clipping region that consists of the existing clipping region minus the specified rectangle
30	INTERSECTCLIPRECT		creates a new clipping region from the intersection of the current clipping region and the specified rectangle
31	SCALEVIEWPORTEXTEX		modifies the viewport for a device context using the ratios formed by the specified multiplicands and divisors
32	SCALEWINDOWEXTEX		modifies the window for a device context using the ratios formed by the specified multiplicands and divisors
33	SAVEDC		saves the state of the currently selected device context
34	RESTOREDC		restores the a saved device context
35	SETWORLDTRANSFORM		two-dimensional transformation between “world-space” and
36	MODIFYWORLDTRANSFORM		changes the world transformation for a device context using the specified mode
37	SELECTOBJECT		chooses a new object (a new object to be drawn “begins”)
38	CREATEOPEN	PEN	a pen is created with the specified attributes
39	CREATEBRUSHINDIREC	FILL	a brush with specified attributes is created
40	DELETEOBJECT		all system resources used to draw an object are freed
41	ANGLEARC	ARC	an arc is drawn using the current pen
42	ELLIPSE	ELLIPSE	an ellipse is drawn
43	RECTANGLE	RECTANG	a rectangle is drawn
44	ROUNDRECT	CURVE COMPOU	draws an rectangle with rounded corners
45	ARC	ARC	draws an elliptical arc
46	CHORD		draws an area defined by the intersection of an ellipse with a
47	PIE	PIE	draws a filled piece of an ellipse
48	SELECTPALETTE		selects the specified logical palette into a device context
49	CREATEPALETTE		creates a logical palette
50	SETPALETTEENTRIES		sets RGB (red, green, blue) colour values and flags in a range of entries in a logical palette
51	RESIZEPALETTE		increases or decreases the size of a logical palette
52	REALIZEPALETTE		maps palette entries from the current logical palette to the system
53	EXTFLOODFILL		fills an area of the display surface with the current brush
54	LINETO	LINE	draws a line from the current position, current position is updated
55	ARCTO	ARC	draws an arc from the current position, current position is updated
56	POLYDRAW		a set of lines and bezier curves is drawn
57	SETARCDIRECTION		are arcs drawn in clockwise or counter-clockwise direction
58	SETMITERLIMIT		sets the limit for the length of miter joins for the specified device
59	BEGINPATH		opens a path bracket in the specified device context

60	ENDPATH		closes a path bracket in the specified device context
61	CLOSEFIGURE		closes an open figure in a path
62	FILLPATH		all figures in the path are closed and the region is filled with the current brush
63	STROKEANDFILLPATH		closes any open figures in a path, strokes the outline of the path by using the current pen, and fills its interior by using the current
64	STROKEPATH		renders the specified path by using the current pen
65	FLATTENPATH		transforms any curves in the path that is selected into the current device context, turning each curve into a sequence of lines
66	WIDENPATH		redefines the current path as the area that would be painted if the path were stroked using the pen currently selected into the
67	SELECTCLIPPATH		selects the current path as a clipping region for a device context
68	ABORTPATH		closes all paths in the current device context
69	---		---
70	GDICOMMENT		copies a comment from a buffer into a specified EMF
71	FILLRGN	RECTANGLE FILL	fills a specified region with the current brush
72	FRAMERGN		outlines the defined region with the current pen
73	INVERTRGN		inverts the colours in the specified region
74	PAINTRGN		paints the specified region by using the brush currently selected into the device context
75	EXTSELECTCLIPRGN		combines the specified region with the current clipping region
76	BITBLT		performs a bit-block transfer of the colour data corresponding to a rectangle of pixels from the specified source device context into a destination device context
77	STRETCHBLT		copies a bitmap from a source rectangle into a destination
78	MASKBLT		combines the colour data for the source and destination bitmaps using the specified mask and raster operation
79	PLGBLT		performs a bit-block transfer of the bits of colour data from the specified rectangle in the source device context to the specified parallelogram in the destination device context
80	SETDIBITSTODEVICE	POINT PEN	sets the pixels in the specified rectangle on the device that is associated with the destination device context using colour data from a DIB
81	STRETCHDIBITS	POINT	copies the colour data for a rectangle of pixels in a DIB to the specified destination rectangle.
82	EXTCREATEFONTINDIR	FONT	creates a logical font that has the specified characteristics
83	EXTTEXTOUTA	TEXT	draws text in the currently selected font style
84	EXTTEXTOUTW	TEXT	draws text in the currently selected font style
85	POLYBEZIER (16 bit		used to store information on drawing multiple bezier curves
86	POLYGON16	POLYGON	draws a polygon, outlined with the current pen and filled with current brush
87	POLYLINE16	POLYLINE	draws multiple connected line segments using the current pen
88	POLYBEZIERTO16		draws multiple bezier curves, starting on the current position, the current position is then set to the ending point of the last
89	POLYLINETO16	POLYLINE	draws multiple connected line segments using the current pen and updates the current position
90	POLYPOLYLINE16	POLYLINE	draws at least one polyline, normally a whole set of polylines is
91	POLYPOLYGON16	POLYGON	draws at least one polygon, normally a whole set of polygons is

92	POLYDRAW16		draws a set of polylines and bezier curves
93	CREATEMONOBRUSH		creates a logical brush that has the pattern specified by the
94	CREATEDIBPATTERNBRUS		creates a logical brush that has the pattern specified by a DIB
95	EXTCREATEPEN		creates a logical cosmetic or geometric pen that has the specified style, width, and brush attributes
96	POLYTEXTOUTA		draws a set of strings
97	POLYTEXTOUTW		draws a set of strings

There is a huge number of records, but only some of them contain relevant drawing information which can be used to convert a file to Leo. While working with EMF2LEO you should be aware that is not possible to convert every EMF. You shouldn't use complicated images. These constraints are due to limitations of Leo.

So in Leo, only polygons, rectangles, pies and figures created using the keyword "compound" are filled with the current brush. On the other side, all geometric elements that form a closed figure are normally filled within an EMF. You should read the last chapter of this documentation to get further information how to produce good results while working with EMF-files.

It is also possible to include bitmap-graphics in an EMF-file. EMF2LEO offers some functions to convert pixel graphics into Leo. Nevertheless, in Leo every pixel is represented by a POINT-command. So you can imagine, that large bitmaps easily create Leo-files larger than 32 kilobytes. The ADOxx® GraphRep only accepts Leo-scripts not exceeding a size of 32 kilobytes. So everything that is larger than this limit, cannot be completely pasted into the GraphRep.

3. Several functionalities

3.1. Specifying a zoom factor

It is possible to change the size of the original graphic stored in an EMF for the output in Leo-script. There are no restrictions or limits for zooming. Nevertheless you should consider to obtain Leo-images that have a reasonable size.

Here are some examples regarding different zooming factors:

100 the size is not changed (100% of the original image)

50 the size of the Leo-image is reduced to the half of the EMF-image

25 the size of the Leo-image is reduced to one quarter

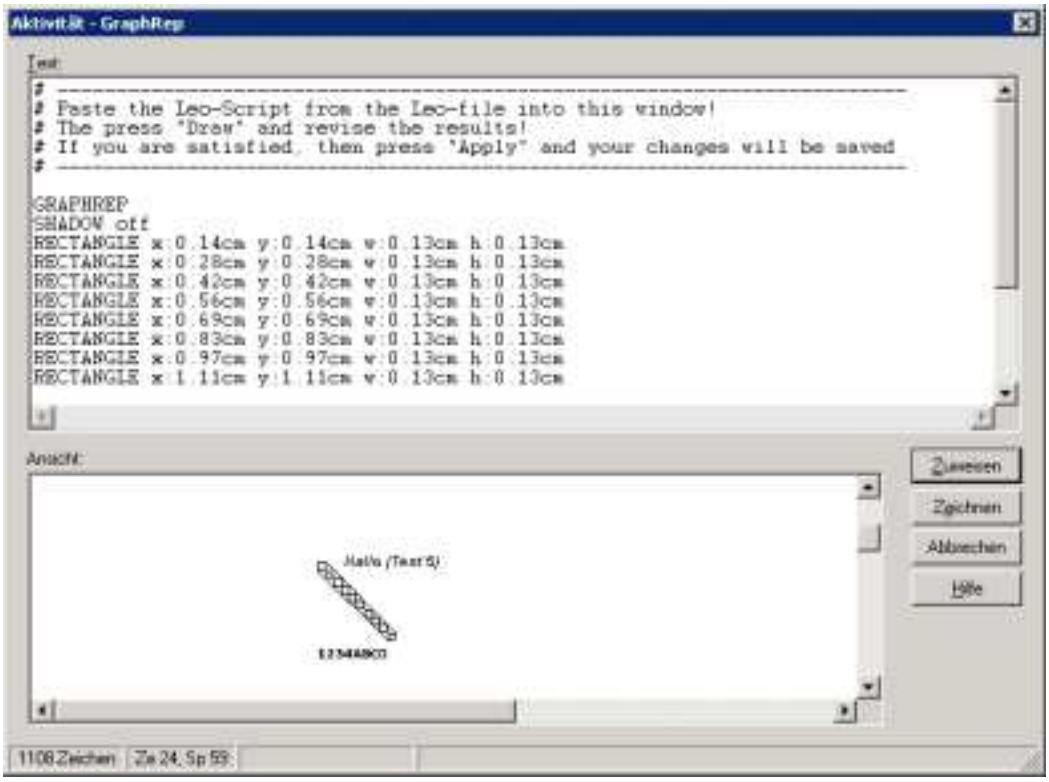
0 a Leo-script is created and all co-ordinates, used to position the image are set to zero

800 the size of the Leo-image is eight times bigger than the original EMF-image

3.2. How to get Leo-Script in the ADOxx® GraphRep

EMF2LEO creates a Leo-output-file. This file is saved in the specified directory or in the same directory in which the original EMF is stored. You should open the file in a text editor (e.g. Notepad) and then copy the whole script into the clipboard.

Then, open the ADOxx® Administration Toolkit and chose the class attribute you want to work on in the class library. In the GraphRep window you paste the new Leo-scriptand then you press the button “draw” to check the result. In almost every case, it isnecessary to revise the results and to rework them.



4. Examples

First of all, you should be aware that EMF2LEO is just a tool that should support the work of the BOC customising staff. It only facilitates creating images that represent class attributes in ADOxx®. Due to the properties of graphic programs the results obtained by EMF2LEO can differ a lot. I tested Metafile Companion and Corel Draw 9 and I have to admit that even Corel Draw is not a very good tool to create good EMFs. The problem is, that often elements are saved as bitmaps within the EMF instead of using records provided for polygons, ellipses etc. Those parts are lost when converting a file from EMF to Leo. Fill commands are also replaced by bitmap-pattern-brushes and so it will probably occur that a rectangle which has been filled in a graphic program won't be filled in Leo. Here, the customizer's work begins. He has to add FILL instructions and in some cases replace polygons formed by LINE instructions into real polygons, so that they can be filled.

On the other hand, the shareware Metafile Companion Version 1.11 is an excellent tool to edit and to create Enhanced Metafiles. It is strongly recommended to make use of this tool, if you want to paint images for the ADOxx® GraphRep. Following the description how to use Corel Draw 9, you will find detailed information on Metafile Companion.

4.1. Corel Draw 9 and EMF2LEO

While working with Corel Draw, you should pay attention to the fact, that curves are not saved in the appropriate records for ellipses or arcs in the EMF-file. So, they are not converted into LEO. If you use other vector graphic programs than Corel Draw 9, you should test yourself whether it stores EMFs in a "clean" way.

A good possibility to obtain at least the structures of an image, is to save it as WMF (Windows Metafile – ancestor of EMF), then reload the WMF-file in Corel Draw and save the WMF-file as an EMF-file. When you convert this EMF-file to Leo, you obtain an usable image, which mainly consists of LINE instructions. So FILL instructions are ignored by the Leo interpreter, because in Leo only polygons, rectangles, ellipses, pies and figures composed by 'compound' are filled.

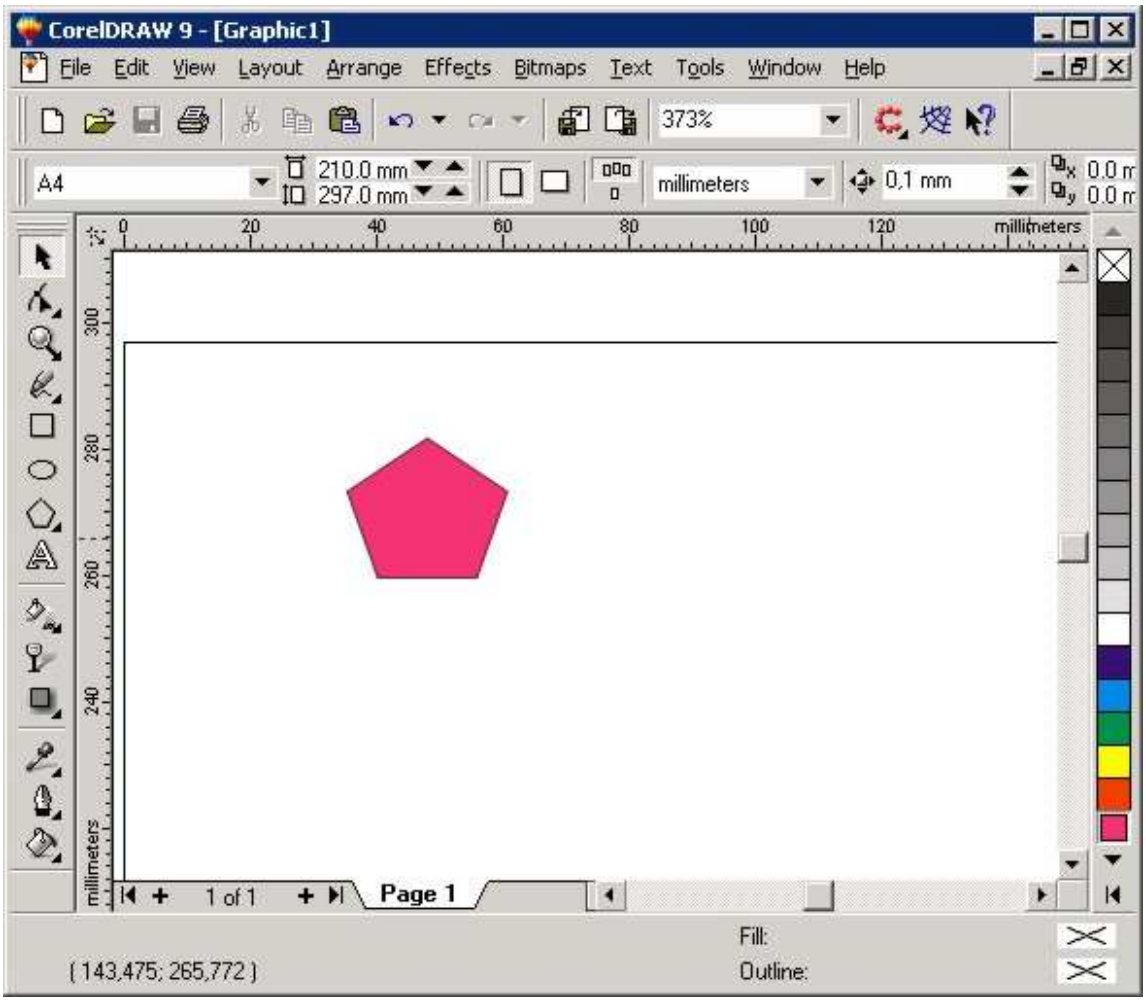
The main advantage of EMF2LEO is that it provides reference points. You do not need to calculate co-ordinates. You paint a raw picture e.g. in Corel Draw 9 and then you convert it into Leo. Although probably not all figures will be filled or even painted, you will receive some reference points you can use to create your own Leo instructions and to position the figures that compose the whole image.

In the following passages, it is going to be shown how you should proceed when you create little images with Corel Draw 9 and Paint Shop Pro 5.

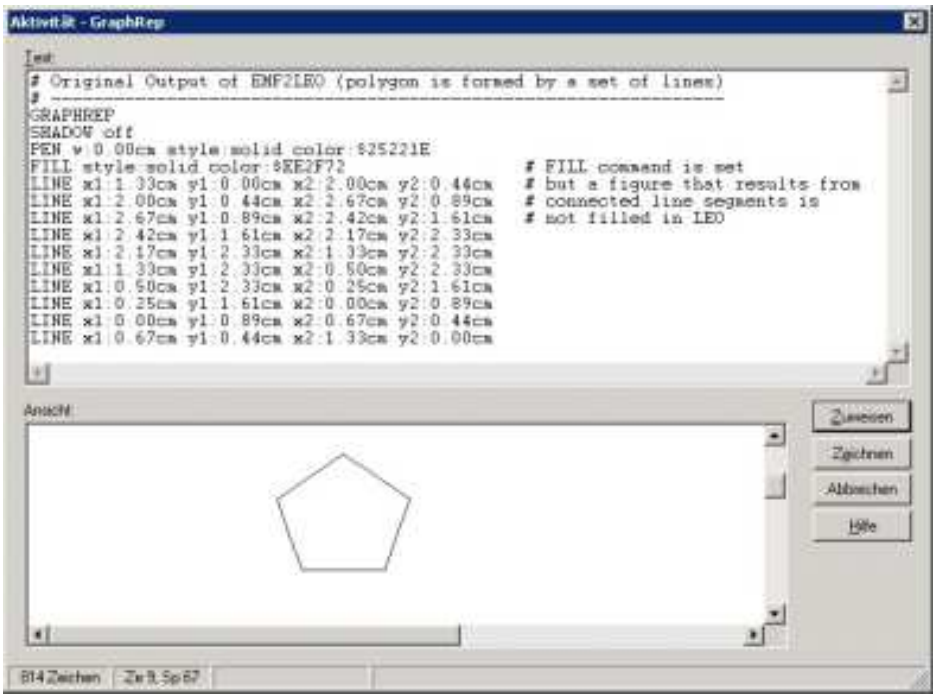
4.1.1. Using Polygons drawn with Corel Draw 9

Not always a polygon is stored as a polygon in Corel Draw. Often, a polygon is created by connecting line segments. The problem consists of the fact that Leo does not fill figures which are composed by line segments.

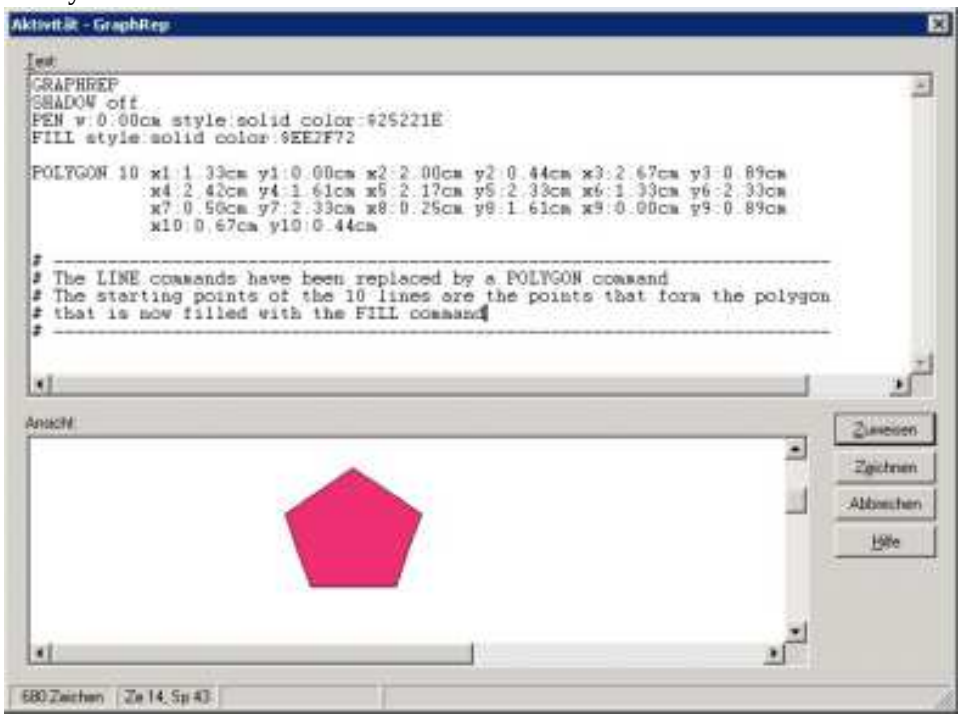
Assuming you want to draw a polygon like it is shown in the following screenshot, you should proceed as it is shown below:



After having drawn the polygon using the "Polygon Tool", you have to export the file to the EMF-format or save it as an EMF-file. Then you have to create a Leo-script. To do so, you use the EMF2LEO program. After having pasted the LEO-Code into the GraphRep of a class attribute, you should get as result an unfilled polygon in the same size:



To obtain a filled polygon, you have to change these 10 LINE instructions into one POLYGON instruction. As reference points for the polygon you should use the x1/y1 co-ordinates of every LINE.



4.1.2. General information about EMF and Corel Draw 9

Of course, EMF2LEO is able to create polygons, too. In this case the problem is, that Corel Draw doesn't recognise a polygon (although the figure had been drawn with the Polygon-Tool!) and saves the figure using the line-to record of an EMF. When converting similar figures, it seems to be the same – Corel Draw likes LINE instructions. Apparently Corel Draw doesn't use the original Windows-API functions provided to work with Enhanced Windows Metafiles.

4.1.3. Corel Draw 9 Tools that work fine with EMF2LEO

- lines created by the freehand tool
- rectangles
- polygons

4.1.4. Corel Draw 9 Tools you shouldn't use

- ellipses
- arcs
- spiral tool
- shadows

In some cases figures are filled with a brush, in other cases a DIB-pattern brush is used. If a DIB-pattern brush is used, filled areas won't be filled in Leo.

If you work with other vector graphic programs, you should test yourself if those programs create "clean" Enhanced Windows Metafiles. EMF2LEO was tested with EMF- files containing geometric figures directly created with Windows-API-functions. EMF2LEO worked fine and is able to generate good Leo-scripts if you have a program that creates metafiles by using the Windows-API-functions (API = Application Programming Interface).

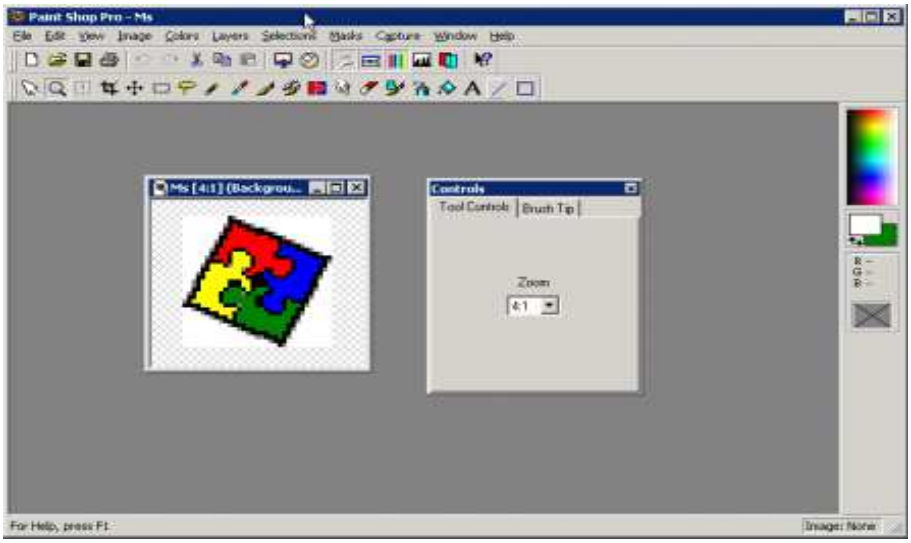
4.2. Device Independent Bitmaps (DIB) stored in an EMF

EMF2LEO is able to generate Leo-script for Device Independent Bitmaps (DIB) stored in an EMF. DIBs are normal bitmaps which can be included in special records of an EMF. If these bitmaps are not compressed, Leo-script will be generated and you can use these bitmaps in ADOxx®. Nevertheless, it is necessary to consider that every pixel of the bitmap will be represented by a POINT instruction in Leo and so it is possible that very large scripts are created. Leo-scripts cannot exceed a size of 32 kilobytes. A POINT instruction has a size of 23 bytes and a PEN instruction which sets the colour has a size of approximately 38 Bytes. So if you use only one colour and you fill an area with this colour, you can use about 1300 POINT instructions, i.e. a bitmap with a size of 36 x 36 pixels is the largest bitmap which would be completely converted into Leo. For every change of the colour, about two POINT instructions less can be used. Nevertheless, you should know that pixels that are white in the original bitmap are not converted into Leo-POINT instructions. So if you have a large white picture, with only some pixels in other colours, the use of EMF2LEO should permit acceptable results.

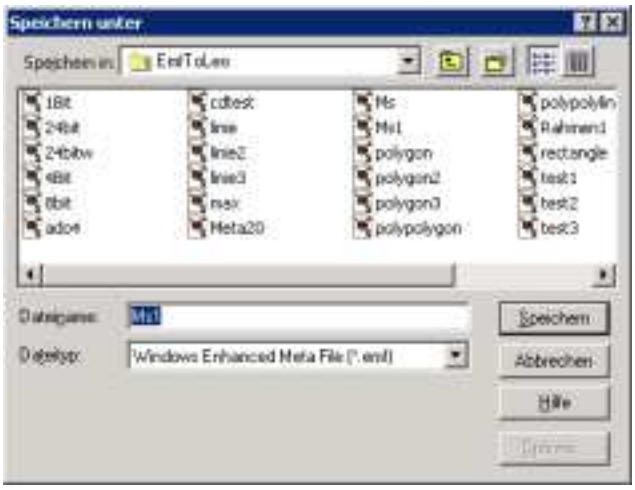
EMF2LEO is able to create Leo-script out of DIBs that use

- 2 colours (monochrome, 1 bit)
- 16 colours (4 bits)
- 256 colours (8 bits)
- 16777216 colours (24 bits).

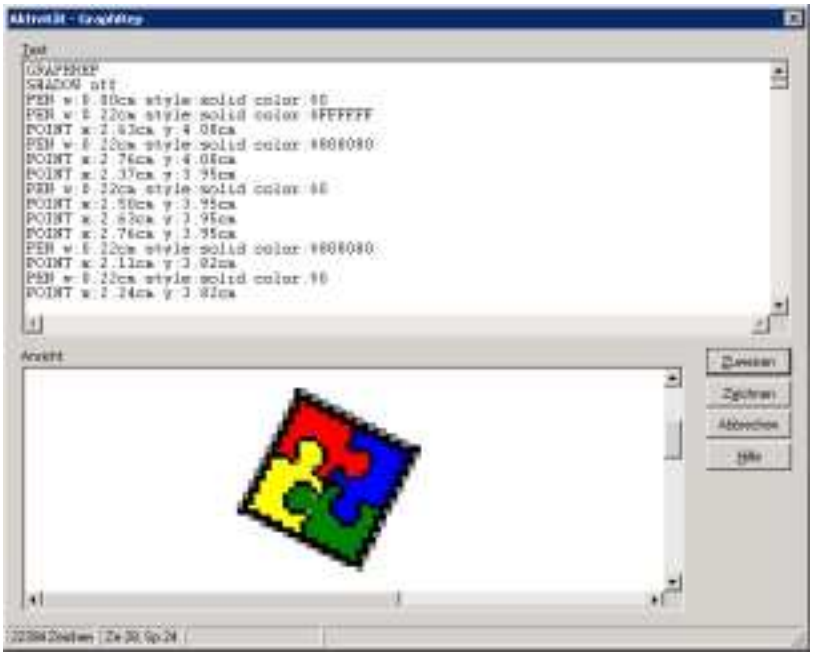
So, when creating images you want to use with ADOxx® you have to consider this fact.
EMF2LEO was tested with bitmaps created by Paint Shop Pro 5.



The logo for Microsoft® Office™ has been opened in Paint Shop Pro. The size of this image is 32 x 32 pixels, but there are some white spaces. You can now save this image in the EMF-format.



A Leo-script-file will be created using the EMF2LEO service. You have to open this file and paste the content into the Graphrep component:



Creating little bitmaps, saving them as EMF and using them in ADOxx® is one of the greatest advantages of EMF2LEO. Nevertheless, scaling problems occur, since in ADOxx® the dimensions of a graphic are not device independent. To gain better results you should make use of the zoom option provided by EMF2LEO.

4.3. Metafile Companion and EMF2LEO

The shareware Metafile Companion is a really good tool if you want to create images for use with EMF2LEO. It is vector graphic program specialised to store the produced paintings in the Windows Metafile or Enhanced Windows Metafile format. There, all the records provided for ellipses, rectangles etc. are used and so images can be converted to Leo without important losses of picture data. In the following passages, it is shown how to create a picture with Metafile Companion and how to get Leo-script code out of it.

Nevertheless, you should consider, that Metafile Companion is a shareware software. It can be downloaded from http://www.CompanionSoftware.com. EMF2LEO has been tested with version 1.11. The program could be used freely 30 times and then it had to be registered.

4.3.1. How to create a metafile using Metafile Companion

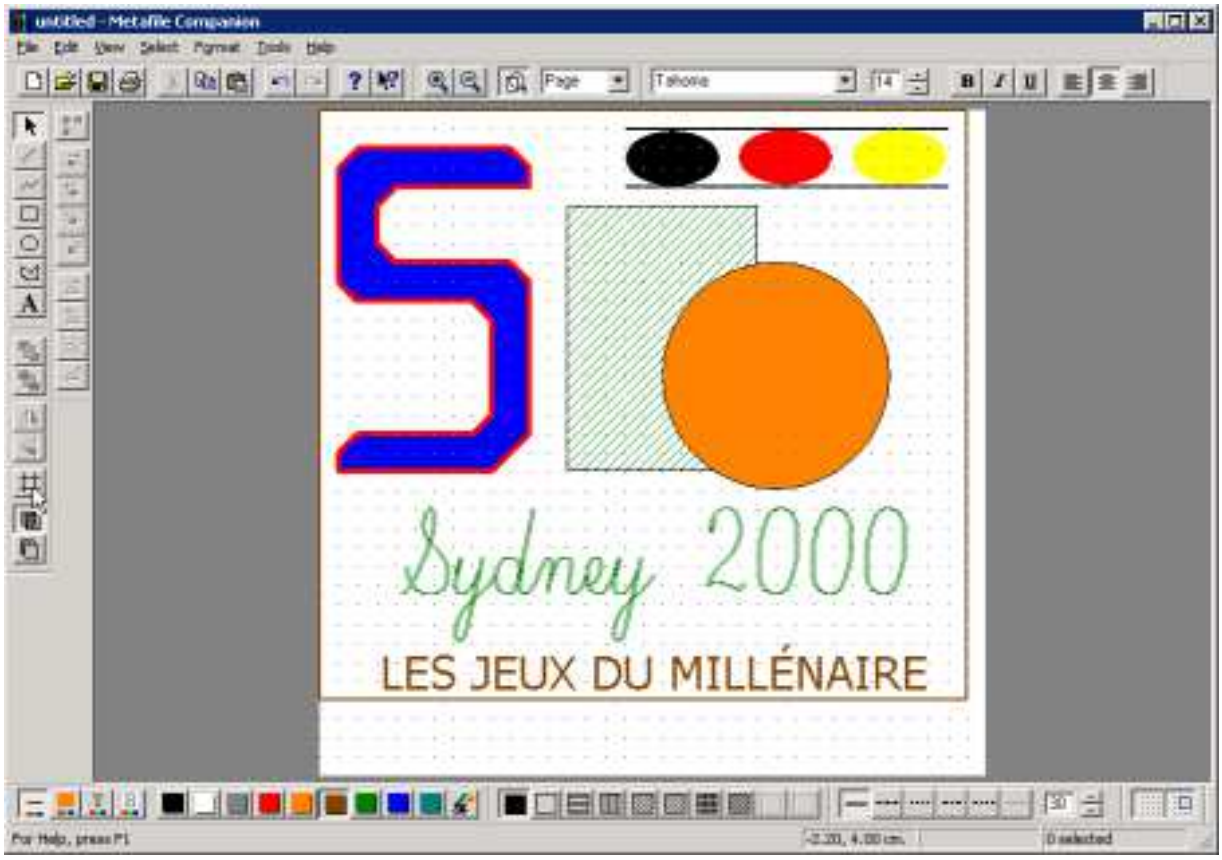
At first, a new file has to be created. Then you should take a look at the properties of the image. To do so, you press ALT+ENTER or you choose "Properties" in the "Edit" menu. The following dialog will appear:

In this dialog, properties for 'Text', 'Fill', 'Line' and 'Picture' can be set. The section 'Other' can be ignored, because everything you set there has no impact on the Leo-script produced later. In this moment, you should specify the dimensions of the image, you are going to create. In this example, it is planned to draw an image that has a width and a height of seven centimetres. Usually the Resize Objects With Picture option is checked meaning that if the entire picture is resized so are all the objects it contains. However,

there are times when you want to change the size of the overall picture without changing the size of the objects. For example, you may want to increase or decrease the "white space" or margin around all the objects without changing the objects themselves. In this case make sure that Resize Objects With Picture is not checked.

In the section 'Fill' you should always choose "solid" as pattern and you can specify the colour which is used for the next objects that are going to be filled. Pay attention to the fact, that new figures you draw for which you haven't specified a new filling colour are created in Leo with the same colour, the current fill property is set to. I.e. if you draw a blue polygon and then you draw an unfilled ellipse, the ellipse will be shown in blue as long as you do not specify "white" as fill property for the ellipse.

You should not use the 'rotate' property in the section 'Text' because, you won't get any good results in Leo. EMF2LEO cannot transform rotated text.

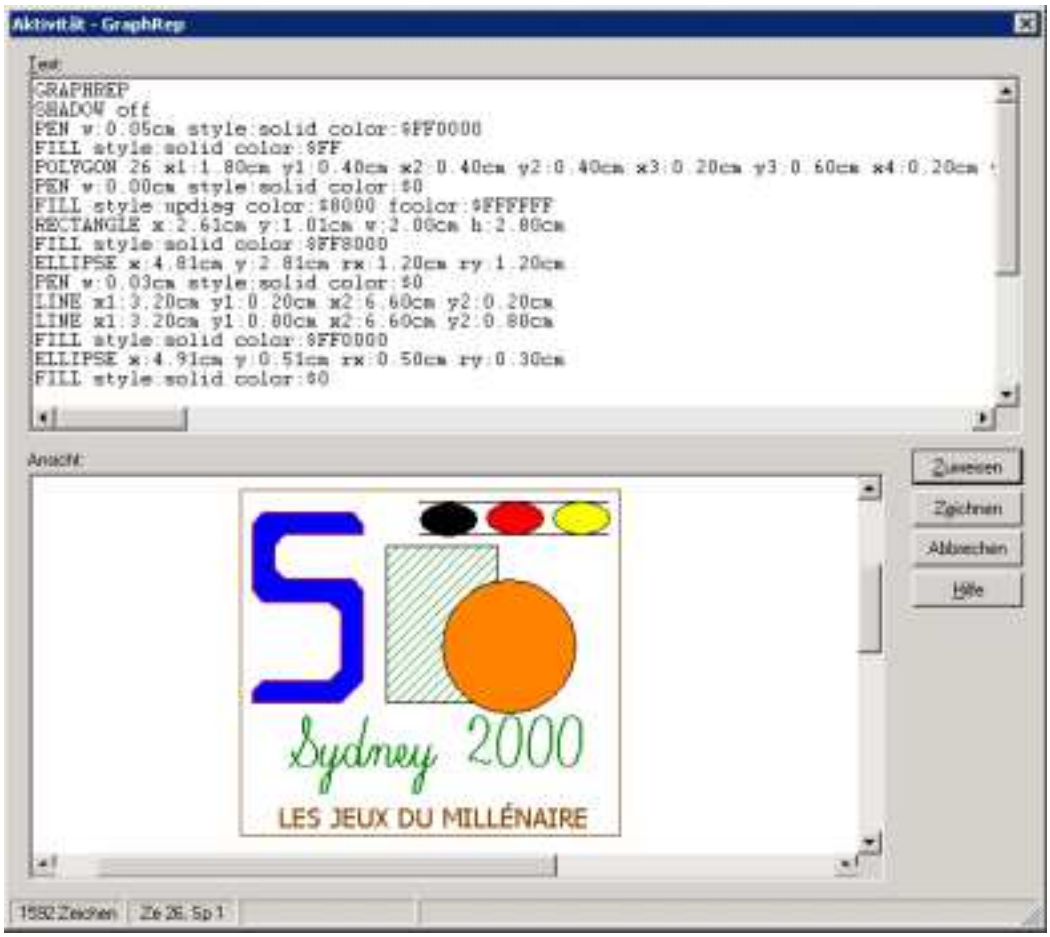


Then you can paint whatever you want, e.g. the following image:

Now, it is suggested to "trim" the picture. This command is used to resize the picture to the smallest size that encloses all the objects in the picture. This removes any extra white space or margin around the objects. Metafile Companion actually leaves a 1%

margin around all the objects so they are not accidentally clipped if the picture is used by another program. To trim the picture, use the symbol on the toolbar the mouse pointer points to on the screenshot.

When you have finished, you save the picture as an EMF-file. In this example, the file has been named "image.emf". To convert it into Leo-script, you use the EMF2LEO web service.



This is the result you get in the ADOxx® GraphRep. The only weakness is the wrong positioning of the text which occasionally occurs. You should be careful when you use a lot of text in an EMF created by Metafile Companion. In almost all the cases you have to reposition the text within the GraphRep.

In this case, you get exactly the same image like its 'ancestor' produced by Metafile Companion.

4.3.2. The following elements and figures can be used

- line
- polygon
- polyline
- text
- ellipse
- rectangle

All figures can be filled. Even if you want to keep some figures white, you should explicitly fill them with white solid colour. Polygons, ellipses and rectangles con also be filled using a horizontal, vertical, forward/backward diagonal or a cross hatch style. You should avoid using “bitmap fill”. Lines can be solid, dashed, dotted, dash-dotted or transparent.

Unfortunately, Metafile Companion does not offer the possibility to draw arcs or pies. These figures can be converted by EMF2LEO, too.

4.4. Cliparts from Microsoft® Office™ (Use of WMF-files with EMF2LEO)

Cliparts included in the Microsoft® Office™ software are normally stored in the WMF-format. To make use of such pictures in ADOxx® you open the desired image in Metafile Companion or another good Windows Metafile editor of your choice. Then you save the opened file in the EMF-format. Now, it is possible to convert office cliparts into Leo-script.

5. Questions and Errors

If you find any errors or unusual behaviour in EMF2LEO or you have questions on the usage of this tool, you should contact:

faq@adoxx.org