VRMeta 1.2 Contents

Support - EMail 72361,2107@compuserve.com or the 3rdParty section of the CIS Delphi forum subject VRMETA.

Introduction

Welcome and thank you for purchasing VRMeta 1.2!

VR Meta is a descendant of TMetafile that provides a Canvas property for you to draw on using Delphi and WinAPI commands. Additional methods and properties open up the world of Metafiles for your applications.

Components

Reference

TVRMetafile Object

Properties Methods

A TVRMetafile object contains a metafile graphic (WMF file format). A TVRMetafile encapsulates a Windows HMETAFILE.

The Canvas of the TVRMetafile is a TCanvas object specified by the Canvas property.

The unscaled height and width of the image in pixels are specified in the Height and Width properties, respectively.

To load a metafile from file use the LoadFromFile method. To save a metafile, call SaveToFile.

To draw a metafile on a canvas call draw or stretchdraw methods of a TCanvas object, passing the TVRMetafile as a parameter.

When the metafile is modified, an OnChange event occurs.

See Delphi Help on Tmetafile for more information

Properties

EmptyInch
Height
Width
Canvas
Handle
IsDiskBased
Set MM
TmpFileName
OnPlayRecord

Empty Property

Applies to
TVRMetafile objects

Declaration
property Empty: Boolean;

Description
Read-only. The Empty property specifies whether the graphics object contains a graphic. If Empty is True, no graphic has been loaded into the graphics object. If Empty is False, a graphic is contained by the graphics object.

Inch Property

Applies to
TVRMetafile objects

Declaration
property Inch: Word;

Description
The Inch property defines the number of metafile units per inch used in determining the Height and Width properties. It is also the notional resolution of the image and should match the scale used for all dimensions in drawing commands.

Don't confuse this with either the resolution of your screen at design time or the resolution of the intended output device, Metafiles are independent of any device resolution, that is you can design them at any notional resolution and then they will be scaled to match the resolution of the device that they will be played on.

That is on a 96DPI screen you design a WMF at a notional resolution of 1000DPI and then have it playback acurately on any device from a 600DPI printer to a 96DPI screen.

MS notes that to avoid numeric overflow the value must be less than 1240.
It appears all MS Draw WMF's use a value of 576 and all MicroGrafix WMF's use 1000.

By default we follow the Micrografix standard, but you can alter this if you wish.

The Height and Width properties are entered as PIXELS based on the current Screen resolution but internally they are converted by multiplying by the inch property and stored as Metafile units.

Unless you have a good reason leave this property at its default value.

Height Property

Applies to
TVRMetafile objects

Declaration
property Height:integer;

Description
The height property defines the theoretical unscaled height of the picture in Pixels.
Theoretical because it does not actualy place any limitation on where you can draw or how large the picture is, it serves mainly as a basis for scaling calculations and is used by Delphi to AutoSize the metafile in a TImage.

Width Property

Applies to
TVRMetafile objects

Declaration
property Width:integer;

Description
The width property specifies the theoretical unscaled width of the picture in PIXELS.
Theoretical because it places no actual restriction on where you can draw nor how large the picture is, it serves mainly as a basis for scaling calculations. This should be set to the value you would expect the metailfe to occupy in a TImage at the Screen resolution your using at design time.

Canvas Property

Applies to
TVRMetafile objects

Declaration
property Canvas:TCanvas;

Description
Run-time and read only. The canvas property gives you access to the drawing surface that represents the Metafile. When you draw on the canvas you are in effect recording GDI commands in the underlying metafile.
See Delphi Help on TCanvas and API reference for drawing comands.

Handle Property

Applies to
TVRMetafile objects

Declaration
property Handle:HMETAFILE;

Description
The handle is the handle to a completed Metafile.
Accessing this property will automatically close the canvas of the metafile so ending the current drawing session.

IsDiskBased Property

Applies to
TVRMetafile objects

Declaration
property IsDiskBased:Boolean;

Description
Determines wether the metafile when created will be disk based. This should not be confused with wether you have loaded the WMF from disk or wish to save it to disk. This property allows the creation of a WMF that does not exist in memory except for the location of a small memory block holding the handle. All drawing commands are sent direct to disk and when the handle is required they are read back from disk. This significantly slows down the rendering process but can be of use in low memory conditions or where there are a large number of metafiles being created or an extremely large and complex image that may not othewise fit in memory.

SetMM Property

Applies to
TVRMetafile objects

Declaration
property SetMM:boolean;

Description
By default when a TVRMetafile is created the inch property is set to 1000, the Mapping Mode is set to ANISOTROPIC, the Window origin is set to 0,0 , the Window extent is set equal to the height and width properties and the canvas is assigned the current default font. If this property is set to False this default action is not carried out, thereby allowing you freedom to set whatever values you wish.

TmpFileName Property

Applies to
TVRMetafile objects

Declaration
property TmpFileName:string;

Description
The TmpFileName is used when a DiskBased metafile is created, the WMF will be stored in this file while in use.When the object is freed then the TmpFile will be erased. If you wish to keep the metafile you must use the SaveToFIle methods.
This property is read only!

OnPlayRecord Property

Applies to
TVRMetafile objects

Declaration
property OnPlayRecord:TOnPlayRecord

Description
Specifies the user function to handle the OnPlayRecord event raised by the EnumMeta method.

TOnPlayRecord= function(hdc: HDC; PHTable:PHandleTable; MFR:PMetaRecord; Handles:word) :word of object;
You can call the WinAPI procedure PlayMetaFileRecord to play a record for example PlayMetaFileRecord(hdc, PHTable^, MFR^, Handles);

The TOnPlayRecord should set the Result variable to 1 to continue enumerating through a metafile.

See EnumMeta Method for more information.

Methods

Free Image
Release Handle
Close
Play
Print
Merge
ScaleMerge
ScaleMergeOffset
StretchMerge
StretchMergeOffset
EnumMeta
CopyToClipboard

Free Image Method

Applies to
TVRMetafile objects

Declaration
procedure FreeImage;

Description
The FreeImage procedure clears all memory associated with a metafile ie returning it to an empty state. The same state can be achieved by setting the Handle property to nil.

CopyToClipboard Method

Applies to
TVRMetafile objects

Declaration
procedure CopyToClipboard;

Description
The CopyToClipboard method copies the Metafile to the clipboard. It corrects for Delphi's errors in writing the MetafilePict header. By default Delphi writes the MapMode as HiEnglish and the extents as the metafile width and height in metafile units, VRMeta corrects this to ANISOTROPIC and the extents to the Height and Width in HiMetric units.

Release Handle Method

Applies to
TVRMetafile objects

Declaration
function ReleaseHandle:HMETAFILE;

Description
The ReleaseHandle method release the association of the underlying metafile to this particular object. That is after using this method you can free the object but continue using the Metafile handle returned by this function using WinAPI commands. Of course you are then responsible for ultimately deleting the Metafile.

Close Method

Applies to
TVRMetafile objects

Declaration
procedure Close;

Description
The close method closes a Metafile that is currently open for drawing. It releases the object from the GDI heap and returns in effect a handle to an object on the global heap.
If you have finished drawing you need to call this method before using the metafile as you may otherwise get a blank metafile.
Closing a metafile does not prevent you from subsequently editing the metafile.

Play Method

Applies to
TVRMetafile objects

Declaration
procedure Play (hdc:HDC);

Description
There is generaly no need to use this method. It is a simple wrapper round the WINAPI PlayMetafile procedure and plays the Metafile into the DisplayContext identified in the hdc parameter. Provided for advanced user who may need to manualy control the playing of a metafile in special circumstances

Print Method

Applies to
TVRMetafile objects

Declaration
procedure Print(hdc:HDC);

Description
The print method is currently identical to the Play method, in earlier beta versions extra functionality was provided for printing but this has since been rendered obsolete. Additional functionality may be added in future.

ScaleMerge Method

Applies to
TVRMetafile objects

Declaration
procedure ScaleMerge(MF: TMetafile)

Description
Use this method to import a third party metafile into your metafile and maintain the imported metafile at its designed size (simply "playing" the metafile in causes the incoming metafile to stretch to fit the size of your metafile as specified by the width and height properties).

ScaleMergeOffset Method

Applies to
TVRMetafile objects

Declaration
procedure ScaleMergeOffset(MF: TMetafile; Offset:TPoint);

Description
Use this method to import a third party metafile into your metafile at a desired location and maintain the imported metafile at its designed size (simply "playing" the metafile in causes the incoming metafile to stretch to fit the size of your metafile as specified by the width and height properties).
The Offset parameter specifies the offset into your metafile in scale units as set in the inch property.

StretchMerge Method

Applies to
TVRMetafile objects

Declaration
procedure StretchMerge(MF: TMetafile; ImageWidth,ImageHeight: Integer)

Description
Use this method to import a third party metafile into your metafile and size it to the dimensions given in the ImageWidth and ImageHeight parameters, the dimension are given in metafile units as set in the inch property, (simply "playing" the metafile in causes the incoming metafile to stretch to fit the size of your metafile as specified by the width and height properties).

StretchMergeOffset Method

Applies to
TVRMetafile objects

Declaration
procedure StretchMergeOffset(MF: TMetafile; Offset:TPoint ; ImageWidth, ImageHeight : integer);

Description
Use this method to import a third party metafile into your metafile at a desired location and size it to the dimensions given in the ImageWidth and ImageHeight parameters, dimensions are given in metafile units as set in the inch property (simply "playing" the metafile in causes the incoming metafile to stretch to fit the size of your metafile as specified by the width and height properties).
The Offset parameter specifies the offset into your metafile in metafile units.

Merge Method

Applies to
TVRMetafile objects

Declaration
procedure Merge(MF: TMetafile)

Description
Use this method to import a third party metafile into your metafile and automaticaly stretch the imported metafile from its designed size to fit the size of your metafile as specified by the width and height properties.
This is equivalent to simply "playing" the metafile in but adds commands to save and restore the context so that subsequent commands are based on your original settings.

EnumMeta Method

Applies to
TVRMetafile objects

Declaration
procedure EnumMeta(hmf: HMETAFILE)

Description
The EnumMeta is a highly specialised method that can be used to enumerate each record whilst importing a metafile.
By setting the OnPlayRecord property to point to your function, your function will be called for each record in the imported metafile allowing you to alter those records or add additional records. Intended for serious API freaks.
As an example the ScaleMergeOffset method enumerates the imported metafile and amends records setting the Window Origin to correctly position the metafile.

Installation

Copy the VRMETA.DCU to your VCL Library directory. There is no need to use install components (because its not a component) nor is there a need to rebuild the library.

Now that was easy, wasn't it!

Using VRMeta

Add VRMETA to the uses clause of your unit and then create a TVRMetafile object just as you would a TBitmap object and use it in the same way ie.The following code creates a 1" x1" image on a VGA 640 x 480 display (96DPI) and saves the image to disk.

```
procedure MakeIt;
begin

	with TVRMetafile.Create do
	begin
		Inch:=1000;
		Height:=96;
		Width:=96;
		with Canvas do
		begin
			TextOut( 0, 0, 'Hello World');
			Ellipse( 0, 0, 1000, 1000);
		end;
		Close:
		SaveToFile( 'MyWMF.WMF');
		Free;
	end;
end;
```

FAQ

What scale does a metafile use?

A metafiles designed resolution is independent of the screen resolution at design time and the resolution of any device it is being played back on. A metafiles designed resoltuion is specified in the Inch property and is by custom 1000 DPI.
This scale must then be used to specify all dimension in commands you store in the metafile.ie if you want to draw a 1" rectangle you would draw a rectangle with the dimensions 1000 by 1000.
If you are trying to draw an object like a TLabel that is on the screen into a metafile you will need to convert the coordinates, ie if the TLabel is at the point 96,96 on the screen then you adjust by dividing by Screen.PixelsPerInch to turn the location into inches then multiply by the Inch property to turn into metafile units that is 1000,1000 in metafile units (assuming the screen resolution was 96DPI).

I have just created a metafile but when I assign it to a TImage nothing appears?

You need to use the CLOSE method before assigning the TVRMetafile to a TImage. Whilst drawing on a metafile canvas the metafile is open if you access any method that requires the Metafile handle then the metafile is automatically closed, however assigning MyImage.Picture.Metafile:=MyTVRMetafile does not access the metafile handle and will therefore fail. You must explicitly close the metafile.

How should I setup the properties of a TImage if I wish to assign a TVRMetafile to it

The most common settings would be Autosize:=True, AutoStretch:=True;. This will cause the TImage to size to the unscaled size of the metafile when assigned, you can then scale the image as you wish by simply changing the height and width properties of the TImage.
Note DO NOT change the width and height properties of the Metafile, only the height width of the TImage component. You can always refer to the metafile height width properties to find the original size. ie if the Metafile Height is 100, then setting the TImage height to 150 gives 50% increase in size.

How do I scale the image onto a printer canvas

Set the Mapping Mode to ANISOTROPIC then set the ViewPortExtents to a multiple of the the width and height properties adjusting for the different resolutions of screen and printer.
ie This creates a metafile equal in size to a page at screen resolution and then prints to the printer adjusting to the printer resolution.

```
Create a Metafile
Page:=TVRMetafile.Create;
with Page do
begin
Height:=longint(Printer.PageWidth)*Screen.PixelsPerInch div WinProcs.GetDeviceCaps(Printer.Handle,
LOGPIXELSX);
Width:=longint(Printer.PageHeight)*Screen.PixelsPerInch div WinProcs.GetDeviceCaps(Printer.Handle,
LOGPIXELSY);
end;

To scale to the printer
PrnDc:=Printer.Handle;
xPage := GetDeviceCaps (PrnDC, HORZRES) ;
yPage := GetDeviceCaps (PrnDC, VERTRES) ;

SetMapMode(PrnDC,MM_ANISOTROPIC);
SetViewportOrg(PrnDC,0,0);
SetViewportExt(PrnDC,xPage,yPage);

Page.Print(PrnDC); {Draw metafile on the printer canvas}
```

FAQ - 2

I imported a metafile and recorded some additional commands but the scale of the additional commands seems wrong

You must remember that a metafile is a series of GDI commands, when you import a metafile your actualy inserting a number of commands into the current file, the state of the scaling, fonts or colors or anything in the GDI is as it set by the last command in the imported file. To ensure that the GDI is returned to its default state you should use the API command SaveDC before you import the metafile and RestoreDC after importing to return the context to its "normal" state for your metafile. The ScaleMerge and ScaleMergeOffset method handle this automatically for you.

I want to import a metafile but I want it at the same size as it appears after I stretched it not at its original size

Use the StretchMerge and StretchMergeOffset methods to adjust a third party metafile from its original size to any desired size.

Metafiles Reference Material

Metafiles Defined

A metafile is a mechanism for storing a graphics device interface (GDI) "picture" -- a series of GDI functions that are used to draw an image. A metafile consists of a series of records, each representing a GDI function. When the metafile is played back, each stored function is executed using its recorded parameters.

In effect, a metafile is a journal of GDI operations, and because all GDI primitives can be recorded, any image that can be drawn can be stored in a metafile. Because a metafile is in a standard format, applications can exchange metafiles and use them for image storage.

The mapping mode of a metafile can be altered during playback. Thus, the image can be scaled arbitrarily, with every component scaling separately, which minimizes the loss of information for the image as a whole and which is not characteristic of bitmaps. In addition, if the image is sparse, a metafile uses less memory than does a bitmap of the same image.

Because of their device independence and scaling abilities, metafiles are useful for transferring images between applications, and most applications support the Clipboard format associated with metafiles (CF_METAFILEPICT). When treated as a single graphics primitive, a metafile is easy to paste into an application without that application needing to know about the specific content of the picture.

Metafiles Usage

Creating a metafile is as simple as calling the CreateMetaFile function (called automatically whne you access the Canvas property). An application can store a metafile in global memory or to disk; using a memory metafile is faster, but it does use up memory.
The CreateMetaFile function returns a handle to a metafile device context (DC) (The Canvas.Handle property). To record into a metafile any function that performs an output operation or sets a drawing attribute, use this handle in place of a normal DC handle when calling that function.

When the desired picture is stored in the metafile DC, the application calls the CloseMetaFile function (The Close method). As its name implies, the CloseMetaFile function closes the metafile DC so that it can no longer
be used for recording. The function returns a handle to a metafile (The Handle property).

Now the metafile is ready for playback. The PlayMetaFile function is the simplest way to play back a metafile. It accepts a destination DC, which is where the image is to be drawn, and the metafile itself. In this function, GDI recalls every stored instruction in the metafile and executes it to the destination DC. The application can place the image anywhere in the destination DC and scale it to the desired size by altering the logical coordinate system (see below).

When the application is through with the metafile and before terminating, the application must free GDI memory used by the metafile by means of the DeleteMetaFile function. If the metafile is stored on disk, the file remains untouched; only GDI memory associated with the metafile is freed. GDI deletes all objects created during a metafile playback as soon the playing is complete.

Metafiles Capabilities

In Microsoft® Windows™ version 3.0, some major improvements greatly increased the practicality of metafiles. The size restriction on metafiles was essentially removed, and now their size is limited to 2^32 bytes of information (a DWORD value for the size). The size of each record is no longer limited to 64K, so large bitmap operations can now be handled successfully. This size increase would be useless, however, if the number of objects were still limited by the size of GDI's heap, so the META_DELETEOBJECT record was added to allow object cleanup during playback.

For the object deletion to work correctly when recording to a metafile, first deselect the object being deleted. Deleting an object that is currently selected into a metafiling DC will work, but no META_DELETEOBJECT record is created for that object. The "stranded" object is deleted when the metafile playback is complete, but it remains on the system throughout the playback.

Using device-independent bitmaps (DIBs) to store all bitmap information significantly improves device independence. Using the new DIB-based records, an application can place color bitmaps in metafiles without losing information for functions such as BitBlt and StretchBlt. The conversion to a DIB while recording, and from a DIB during playback, is automatic.

The ability to use a metafile DC as the destination DC of a playback is also new; a metafile can now be played into another metafile. Thus, you can easily embed metafile information inside another metafile or copy pieces of one metafile into another. A word of caution: Windows version 3.0 crashes if you use PlayMetaFile and the destination DC is a memory-based metafile. This situation is corrected in Windows version 3.1, and you can use PlayMetaFileRecord for both memory-basedand disk-based metafiles in Windows versions 3.0 and 3.1.

Metafiles Limitations

Some limitations in GDI APIs do not permit metafiles to be fully functional. Because Windows version 3.0 lacked a scaling font technology, fonts used in metafiles did not scale nicely as the metafile was sized. The addition of TrueType™ in Windows version 3.1 eliminates this problem. Because no curve primitive is defined beyond the basic Ellipse functionality, including a complex curve in a metafile is not possible. Although you can use the Polygon function to draw a curve, it will not scale smoothly. Regions do not scale at all, rendering them virtually useless for any sort of complex clipping within a metafile.

Under Windows version 3.0, when an application passes a handle to a DC, determining whether the DC is real or a metafile is not possible. Under Windows version 3.1, GetDeviceCaps(hDC, TECHNOLOGY) correctly identifies a DC as a metafile DC (return value is DT_METAFILE) when appropriate.

Metafiles Scaling

A metafile that is created by an application and then passed to another application is likely to be scaled. Scaling may alter the desired image in a way not anticipated by the creating application that does not scale the image. Every logical measure defined in a logical object is scaled before the object is realized into physical form.

For a logical object such as pens, the width is transformed from logical to physical as an x-scalar value. If the metafile is scaled in y but not in x, the pen width is unchanged. If the metafile is scaled in x but not in y, the pen width does scale. Thus, using a pen of width 1 in a metafile results in a pen that is wider (thick and slow) when the metafile is scaled. If a nominal width pen (width of 1 at all times) is desired, use 0 as the width because it is not affected by mapping modes. A 0-width pen is drawn as having a width of 1.

Font sizing is more complicated. The two values that scale in a logical font are the height and the width. Most applications use a width of 0 to define a font, which results in a physical font with a width that was designed for the given height. As the metafile is stretched in x, the font remains the same size. As the metafile is stretched in y, however, the physical font grows bigger and probably wider. In and of itself, this is not a bad thing, but problems arise when the metafile makes assumptions about the width of the font by placing the characters of a text string individually by using ExtTextOut with a width array or using a TextOut for each character. In either case, the x-placement of each character scales with the metafile, but the font's width does not necessarily scale accordingly, which causes characters to overlap or be widely spaced.

The simplest way to overcome this situation is to not place the characters individually but to use TextOut (or ExtTextOut with no width array) to output the whole string. The text string remains intact, but its size may change in relation to the rest of the image when x and y are not scaled identically. Another possibility is to define the font with a nonzero width so that it scales in x as well as in y. Doing so in Windows version 3.0 is not wonderful because its bitmapped fonts do not scale independently in x and y. Scaling a font's width is possible with TrueType in Windows version 3.1. Unfortunately, anytime a font's width is scaled, the look of the typeface changes in ways not necessarily intended by the designers, and a typographically "incorrect" typeface results.

Metafiles Pitfalls

GDI functions that return data either do not work properly or crash the system if the DC passed in is a metafile DC. This category of functions includes all Get functions as well as RectVisible, PtVisible, EnumFonts, EnumObjects, DPtoLP, and LPtoDP. Any Escape function that involves a data return is recorded in the metafile but returns no meaningful data.

A number of functions in the Windows API appear to the naked eye to be GDI functions with a DC parameter that should be able to be recorded in a metafile; in reality these are functions of the window manager interface and are not recorded in a metafile. They are DrawFocusRect, DrawIcon, DrawText, ExcludeUpdateRgn, FillRect, FrameRect, GrayString, InvertRect, ScrollDC, and TabbedTextOut. Because a metafile DC is not actually associated with a device, you cannot use SetDIBits, GetDIBits, and CreateDIBitmap with a metafile DC. You can use SetDIBitsToDevice and StretchDIBits with a metafile DC as the destination. CreateCompatibleDC, CreateCompatibleBitmap, and CreateDiscardableBitmap are not meaningful with a metafile DC.

Calling SelectObject with a metafile DC does not return the previously selected object in the metafile; it returns either 1 for a successful recording or 0 for a failed recording. Attempting to use SelectObject with a return of 1 to restore the previous object does not work and causes a UAE in Windows version 3.0.

Support is limited for regions in metafiles. Regions do not scale properly and should be avoided.

Metafiles that are created by an application and then passed to another application should avoid altering the viewport extent in order to be easily scalable (see below for a discussion of using mapping modes with metafiles).

Metafiles Internals

Metafiles have several layers of headers. GDI deals with the METAHEADER structure, which sits directly before the bits. It is described in the API reference. The METAFILEPICT structure is associated with a metafile when it is placed in the Clipboard. Its basic function is to identify the mapping mode and drawing size of the image, which are helpful for proper pasting into another application.

Each metafile record consists of two parts, a descriptor and the contents. Both are defined in the METARECORD structure:

rdSize	The size of the record in WORDs. For many records, this value is a constant because the number of parameters is not variable. The size is a DWORD value that allows records of more than 64K, a common occurrence with bitmap manipulations.
rdFunction	Identifies the function being recorded. It can be one of the META_* values defined in WINDOWS.H. That list of values also shows which GDI API functions can actually be recorded in a metafile.
rdParm[]	The space holder for the function's parameters. The size of this array varies to fit as much memory as is needed.

The API reference, in Delphi for Windows version 1.0 describes the ordering of parameters for all possible records, detailing those with nonstandard parameters.
Those GDI functions with a fixed number of parameters (that is, those that have no arrays or strings) are recorded as they are called, with the parameters stored in reverse order from the function definition.
The functions with complex parameters vary in the way they are stored.

One set of records does more than merely store parameters: the one dealing with the creation, selection, and deletion of objects. Instead of recording actual handles, which is not useful for playback, a SelectObject function generates two records. The first is a creation record (for example, META_CREATEPENINDIRECT). The second is META_SELECTOBJECT, which has a parameter that is an index into the object table. This object table is associated with the metafile and grows as objects are added. Each new object gets a new entry in the table and, hence, an index into the table. If an object is reselected into a metafile, the corresponding META_SELECTOBJECT record references the initial object table entry. When an application calls DeleteObject for an object that was in the metafile, a META_DELETEOBJECT record is added. It references the entry, and that entry is marked as open. The next object that is created for the metafile reuses that entry and its index. Object creation, selection, and deletion depend on proper ordering during playback to achieve the proper results. For PlayMetaFileRecord and the EnumMetaFile callback, this handle table becomes a third component of the metafile. It is used invisibly when PlayMetaFile is used.

Metafiles Mapping Modes

The METAFILEPICT structure contains information about the desired size of the metafile. When an application pastes a metafile, it can use this information to control the size of the metafile output. For this to work cleanly between applications, be aware that:

- The metafile is responsible for the window part of the mapping mode.

- The player of the metafile is responsible for the viewport part of the mapping mode.

To perform a simple playback of the metafile, the application sets the mapping mode to the mode specified in the METAFILEPICT structure, sets the viewport origin to the desired placing of the metafile, and calls PlayMetaFile. How big is this output image? Because the x-extent and y-extent are given in logical units based on the specified mapping mode, you can use LPtoDP to calculate the size of the image in pixels.

If the mapping mode is MM_ANISOTROPIC or MM_ISOTROPIC, sizing is not quite so simple. Because the x-extent and y-extent of the image are given in MM_HIMETRIC coordinates, first convert them to pixel values.
You can use LPtoDP after setting the mapping mode to MM_HIMETRIC or use the HORZSIZE/HORZRES and VERTSIZE/VERTRES ratios (values obtained using GetDeviceCaps) to convert manually. Before playback, the application needs to set the viewport origin to the desired location, set the mapping mode to the specified mode, and set the viewport extents to the values calculated above. A properly created metafile that uses MM_ANISOTROPIC or MM_ISOTROPIC mapping modes sets the window extent at the start of the metafile to complete the mapping mode equation. (The viewport itself is not sufficient; using any of the other mapping modes sets appropriate values for the window extents.)
If no desired extents are provided in the METAFILEPICT structure, the application doing the playback can arbitrarily choose a size.

Scaling a metafile that uses MM_ANISOTROPIC or MM_ISOTROPIC is easy -- merely change the viewport extents to the desired size before playback. The viewport defines the size of the metafile image. To scale metafiles that use any other mapping mode, first transform the metafile to use MM_ANISOTROPIC. You don't need to change the metafile itself, but you do need to change the mapping mode setup before beginning the playback. Here is some simple code to do this:

```
SetMapMode(hDC, lpMetaFilePict->mm);
SetMapMode(hDC, MM_ANISOTROPIC);
```

The first call sets up the viewport and window for the desired mapping mode. The second changes the mapping mode to be scalable but doesn't change the viewport and the window information. Thus, the window setting is in line with the mapping mode of the metafile and the logical coordinates within while leaving the viewport ready for scaling as desired.

Metafiles Manipulating During Playback

You don't need to blindly play back metafiles. Windows has a mechanism that allows an application to inspect every record before it is played, change that record, or even invent a record of its own. EnumMetaFile calls a callback routine with every record found in the metafile. The application then calls PlayMetaFileRecord to play an individual record. In the simplest case, the information passed to the callback can be sent directly to PlayMetaFileRecord to simulate

PlayMetaFile. In more complicated scenarios, the application can change the colors of objects or text of a TextOut, omit certain records, or simply add new records to the playback.

Note: Altering the order of object creation, selection, or deletion calls can adversely affect object management during playback. It is important to keep in mind how the object table works when manipulating object-based records.

Applications that want to store private information (for retrieval during playback) in the metafile can do so by calling the Escape function with MFCOMMENT. The function to be performed does nothing on a regular DC but records that information in the metafile for a metafile DC. Of course, EnumMetaFile must be used during playback for the application to see and use it.

Note: If the metafile is placed in the Clipboard for transferring to another application, the record is ignored during normal playback. To ensure that two commenting applications do not become confused and attempt to interpret the other's private data, place some kind of signature (in the form of a few identifier bytes at the start) in the comment.

Metafiles Valid Functions

The following list of functions that are valid in a Windows 3.1 metafile is taken from page 107 of the "Windows SDK: Programmer's Reference, Volume 1: Overview":

AnimatePalette	OffsetViewportOrg	SetBkMode
Arc	OffsetWindowOrg	SetDIBitsToDevice
BitBlt	PatBlt	SetMapMode
Chord	Pie	SetMapperFlags
CreateBrushIndirect	Polygon	SetPixel
CreateDIBPatternBrush	Polyline	SetPolyFillMode
CreateFontIndirect	PolyPolygon	SetROP2
CreatePatternBrush	RealizePalette	SetStretchBltMode
Ellipse	RestoreDC	SetTextColor
Escape	RoundRect	SetTextJustification
ExcludeClipRect	SaveDC	SetViewportExt
ExtTextOut	ScaleViewportExt	SetViewportOrg
FloodFill	ScaleWindowExt	SetWindowExt
IntersectClipRect	SelectClipRgn	SetWindowOrg
LineTo	SelectObject	StretchBlt
MoveTo	SelectPalette	StretchDIBits
OffsetClipRgn	SetBkColor	TextOut

The following table lists additional functions that are valid in a metafile under Windows 3.1 but were missed in the MS Documentation:

CreateDIBitmap	CreateSolidBrush	Rectangle
CreateFont	DeleteObject	ResizePalette
CreateHatchBrush	ExtFloodFill	SetPaletteEntries
CreatePalette	FillRgn	SetTextAlign
CreatePen	InvertRgn	SetTextCharacterExtra
CreatePenIndirect	PaintRgn	

Although the following functions might function correctly in a metafile, they should not be used in a metafile:

AbortDoc	EndPage	StartPage
EndDoc	StartDoc	ResetDC

Windows has a number of functions that are not supported in metafiles directly because they are used to put an object into a metafile. The region functions are a common example of this function category. An application can create an arbitrary region, either directly or by combining existing regions. When an application selects the handle to the region into a metafile DC (display context), Windows records a CREATEREGION record into the metafile.

An application can select a bitmap into a memory DC that is compatible with the display (not directly into the metafile), and then then call the BitBlt function to move the bitmap from the memory DC into the

metafile DC. Windows saves the bitmap image in the metafile as a DIB (device-independent bitmap).

Placeable Metafiles

A placeable Windows metafile is a standard Windows metafile that has an additional 22-byte header. The header contains information about the aspect ratio and original size of the metafile, permitting applications to display the metafile in its intended form.
All metafiles can be assumed to be of this format. Note that Windows itself cannot handle Placeable Metafiles (curious isnt it) instead the application first reads the Header then discards it and treats the remainder of the file as a standard metafile, dont worry all this is handled for you by Delphi. We note it for reference only.
The header for a placeable Windows metafile has the following form:

```
typedef struct {
    DWORD    key;
    HANDLE   hmf;
    RECT     bbox;
    WORD     inch;
    DWORD    reserved;
    WORD     checksum;
} METAFILEHEADER;
```

Following are the members of a placeable metafile header:
key

Specifies the binary key that uniquely identifies this file type. This member must be set to 0x9AC6CDD7L.

hmf

Unused; must be zero.

bbox

Specifies the coordinates of the smallest rectangle that encloses the picture. The coordinates are in metafile units as defined by the inch member.

inch

Specifies the number of metafile units to the inch. To avoid numeric overflow, this value should be less than 1440. Most applications use 576 or 1000.

reserved

Unused; must be zero.

checksum

Specifies the checksum. It is the sum (using the XOR operator) of the first 10 words of the header.

The actual content of the Windows metafile immediately follows the header. The format for this content is identical to that for standard Windows metafiles. For some applications, a placeable Windows metafile must not exceed 64K.

Note:

Placeable Windows metafiles are not compatible with the GetMetaFile function. Applications that intend to use the metafile functions to read and play placeable Windows metafiles must read the file by using an

input function (such as _lread), strip the 22-byte header, and create a standard Windows metafile by using the remaining bytes and the SetMetaFileBits function.

Registration

This version of VR Meta v1.2. is limited to use for the next 90 days, you MUST register to receive the unrestricted version and to be able to legally distribute applications containing this component.

COST - **US$25.00**

This version of the software package is available via compuserve only.

You can register this software via Compuserve CIS SWREG.

GO SWREG and follow the instructions for registering shareware.

The Registration ID for VRMeta v1.2 is 9524.

We will send the full package via EMail to your CIS address as soon as we are advised of your registration.

Registered users will receive:

1 Full unrestricted version of VRMeta v1.2 .
2 Full component source code.
3 Licence agreement, allowing you to legally distribute applications containing the VRMeta object, plus allowing the help file or its contents to be distributed or copied or included in your own application help file. Help file RTF source documents available upon request.
4 Free future minor version upgrades to VRMeta when available.

Support

Support is available via Compuserve

EMail to CIS ID 72361,2107@compuserve.com - Rob Edgar

or send Email to the CIS DELPHI forum. 3RDParty section. with the subject VRMETA.

License Agreement - Release Version

Important

By using this software you accept the following terms of this License Agreement. If you do not agree with these terms, you should not use the software and promptly return it for a refund.

Ownership

Visual Solutions Ltd. retains the ownership of this copy of the enclosed software package. It is licensed to you for use under the following conditions:

You May

You may transfer this software to another party if the other party agrees to the terms and conditions of the agreement and completes and returns a registration card to Visual Solutions Ltd. The registration card is available by writing to Visual Solutions Ltd. If you transfer the software, you must simultaneously transfer all documentation and related disks.
You may merge this software with your own software or code provided that the primary purpose of your software is not database access control, such software may be distributed without payment of a royalty fee.
You may copy any part of this help file, with the exception of this license agreement, for the sole purpose of informing the users of your software on how to use this unit.

You May Not

You may not copy the documentation or software except as described in the installation section of this manual. You may not distribute, rent, sub-license or lease the software or documentation, including translating, decompiling, disassembling, or creating derivative works. You may not reverse-engineer any part of this software, or produce any derivative work. You may not make telecommunication transmittal of this software.

Termination

This license and your right to use this software automatically terminates if you fail to comply with any provision of this license agreement.

Rights

Visual Solutions Ltd. retains all rights not expressly granted. Nothing in this license agreement constitutes a waiver of Visual Solutions Ltd.'s rights under the H.K. copyright laws or any other law.

Limited Warranty

If you discover physical defects in the media, Visual Solutions Ltd. will replace the media or documentation at no charge to you, provided you return the item to be replaced with proof of payment to Visual Solutions Ltd. during the 90-day period after having taken delivery of the software.

License Agreement

Visual Solutions Ltd.excludes any and all implied warranties, including warranties of merchantability and fitness for a particular purpose and limits your remedy to return the software and documentation to Visual Solutions Ltd. for replacement.
Although Visual Solutions Ltd. has tested the software and reviewed the documentation, Visual Solutions Ltd. MAKES NO WARRANTY OF REPRESENTATION, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS SOFTWARE OR DOCUMENTATION, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS SOFTWARE AND DOCUMENTATION ARE LICENSED "AS IS" AND YOU, THE LICENSEE, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.

IN NO EVENT WILL Visual Solutions Ltd. BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL

OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR DOCUMENTATION, even if advised of the possibility of such damages. In particular, Visual Solutions Ltd. shall have no liability for any data stored or processed with this software, including the costs of recovering such data.
THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESSED OR IMPLIED. No Visual Solutions Ltd. dealer, agent, or employee is authorized to make any modifications or additions to this warranty.

Information in this document is subject to change without notice and does not represent a commitment on the part of Visual Solutions Ltd. The software described in this document is furnished under this license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the written permission of Visual Solutions Ltd.

Some countries do not allow the exclusion of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from country to country.

Index

≡
#
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Glossary

≡
#
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

License Agreement

Important

By using this software you accept the following terms of this License Agreement. If you do not agree with these terms, you should not use the software and promptly return it for a refund.

Ownership

Visual Solutions Ltd. retains the ownership of this copy of the enclosed software package. It is licensed to you for use under the following conditions:

You May

You may transfer this software to another party if the other party agrees to the terms and conditions of the agreement and completes and returns a registration card to Visual Solutions Ltd. The registration card is available by writing to Visual Solutions Ltd. If you transfer the software, you must simultaneously transfer all documentation and related disks.

You May Not

You may not copy the documentation or software except as described in the installation section of this manual. You may not distribute, rent, sub-license or lease the software or documentation, including translating, decompiling, disassembling, or creating derivative works. You may not reverse-engineer any part of this software, or produce any derivative work. You may not make telecommunication transmittal of this software.

Termination

This license and your right to use this software automatically terminates if you fail to comply with any provision of this license agreement.

Rights

Visual Solutions Ltd. retains all rights not expressly granted. Nothing in this license agreement constitutes a waiver of Visual Solutions Ltd.'s rights under the H.K. copyright laws or any other law.

Limited Warranty

If you discover physical defects in the media, Visual Solutions Ltd. will replace the media or documentation at no charge to you, provided you return the item to be replaced with proof of payment to Visual Solutions Ltd. during the 90-day period after having taken delivery of the software.

License Agreement

Visual Solutions Ltd.excludes any and all implied warranties, including warranties of merchantability and fitness for a particular purpose and limits your remedy to return the software and documentation to Visual Solutions Ltd. for replacement.
Although Visual Solutions Ltd. has tested the software and reviewed the documentation, Visual Solutions Ltd. MAKES NO WARRANTY OF REPRESENTATION, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS SOFTWARE OR DOCUMENTATION, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS SOFTWARE AND DOCUMENTATION ARE LICENSED "AS IS" AND YOU, THE LICENSEE, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.

IN NO EVENT WILL Visual Solutions Ltd. BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR DOCUMENTATION, even if advised of the possibility of such damages. In particular, Visual Solutions Ltd. shall have no liability for any data stored or processed with this software, including the costs of recovering such data.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESSED OR IMPLIED. No Visual Solutions Ltd. dealer, agent, or employee is authorized to make any modifications or additions to this warranty.

Information in this document is subject to change without notice and does not represent a commitment on the part of Visual Solutions Ltd. The software described in this document is furnished under this license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the written permission of Visual Solutions Ltd.

Some countries do not allow the exclusion of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from country to country.

Title