NEWTONE

イメージ処理コンポーネント

ImageKit10 VCL

プログラミングガイド

目次

1. 動作環境 3
2. お問い合わせについて 5
3. インストール 6
4. サンプルプログラムについて 15
5. イメージとメモリハンドルの操作について 16
6. エフェクト処理の使用例 19
7. 描画機能を持つコントロール 28
7－1. イメージキットコントロールの使用法 29
7－2. イメージ編集ツールバー(ラスタ)について 34
7－3. イメージ編集ツールバー(ベクトル)について 42
イメージの色について 47
SXF 形式使用時の注意事項 48
索引 49

1. 動作環境

1) 対応プラットフォーム
クライアント OS: 日本語版 Windows 8.1/10
サーバーOS: 日本語版 Windows Server 2012/Server 2012 R2/Server 2016/Server 2019

2) 対応開発コンテナ
Embarcadero RAD Studio XE3(C++Builder XE3/Delphi XE3)
Embarcadero RAD Studio XE4(C++Builder XE4/Delphi XE4)
Embarcadero RAD Studio XE5(C++Builder XE5/Delphi XE5)
Embarcadero RAD Studio XE6(C++Builder XE6/Delphi XE6)
Embarcadero RAD Studio XE7(C++Builder XE7/Delphi XE7)
Embarcadero RAD Studio XE8(C++Builder XE8/Delphi XE8)
Embarcadero RAD Studio 10 Seattle(C++Builder 10 Seattle/Delphi 10 Seattle)
Embarcadero RAD Studio 10.1 Berlin(C++Builder 10.1 Berlin/Delphi 10.1 Berlin)
Embarcadero RAD Studio 10.2 Tokyo(C++Builder 10.2 Tokyo/Delphi 10.2 Tokyo)
Embarcadero RAD Studio 10.3 Rio(C++Builder 10.3 Rio/Delphi 10. 3 Rio)
(すべて日本語版)

3) 対応スキャンデバイス
TWAIN 対応イメージスキャナ(+ ADF)
TWAIN 対応デジタルカメラ
TWAIN 対応フィルムスキャナ

4) 対応 DirectX(DirectShow)
DirectX 9 以上(Web カメラ用のコントロールを使用する場合)

5) 対応イメージファイル形式(読込・保存)
ラスタイメージ

ﾋﾞｯﾄ/ﾋﾟｸｾﾙ		1	4	4	8	8	16	24	32
ﾌｧｲﾙ形式	色数	白黒2値	Color	Gray	Color	Gray			
WindowsBMP/DIB(*1)		○	○	○	○	○	○	○	○
WindowsBMP/DIB(RLE4)			○	○					
WindowsBMP/DIB(RLE8)					○	○			
Exif(JPEG)(*2)						○		○	
FPX(基本機能)(*3)						○		○	
GIF(GIF87a,GIF89a)(*4)		○	○	○	○	○			
JPEG(基本 DCT,ﾌﾟﾛｸﾞﾚｯｼﾌﾞ DCT)						○		○	
JPEG2000(Part1,Code Stream)						○		○	
PCX		○	○	○	○	○		○	
PNG		○	○	○	○	○		○	
TIFF/非圧縮		○	○	○	○	○	△	○	○
TIFF/CCITTRLE		○							
TIFF/GROUP3-1D(*5)		○							
TIFF/GROUP3-2D(*5)		○							
TIFF/GROUP4(*5)		○							
TIFF/LZW		○	○	○	○	○	△	○	○
TIFF/PACKBITS		○	○	○	○	○	△	○	○
TIFF/JPEG(*6)						○		○	

○ 読込・保存とも可能
△ 読込はできるが、保存はできない
(*1)OS/2 BMP は読込のみサポート(16,32 は除く)
(*2)バージョン 2.1 に対応(読込のみ)

(*3)オプション機能は除く
(*4)保存時は GIF89a のみ
(*5)GROUP3-1D は MH、GROUP3-2D は MR、GROUP4 は MMR と同じ形式です。
(*6)読み込みできないファイルも存在します。

ベクトルイメージ
DXF(*7),EMF,SVG(非圧縮)(*8),SXF(SFC/P21),WMF

(*7)DXF のサポート項目
○ヘッダ部
EXTMAX,EXTMIN
○テーブル部
LAYER 色,線種

○エンティティ部
LINE,POINT,CIRCLE,ARC
TEXT　　オプション項目は未対応。
3DFACE　　読込のみ対応。
POLYLINE　曲線には非対応。

基本的にはバージョン GXⅢに対応しておりますが、GXⅢの中でも下記の項目には対応しておりません。
○ヘッダ部
EXTMAX,EXTMIN 以外は非対応
○テーブル部
LTYPE
STYLE
UCS
VIEW
VPORT
○エンティティ部
ATTDEF
ATTRIB
DIMENSION
SHAPE
TRACE
VIEWPORT

(*8)SVG のサポート項目
円、線分、ポリゴン、ポリライン、文字列、矩形

2. お問い合わせについて

弊社にお問い合わせされる場合は、以下の項目をお知らせいただきますようお願いいたします。

ユーザ情報

1)ユーザ ID(ユーザ登録後に弊社より送付(送信)される書類(FAX やメール)に記載されている番号 UID-XXXXXX)
2)お名前(法人登録の場合は、会社名も併せてお知らせください。)
3)ユーザ登録している電話番号、FAX 番号もしくはメールアドレス

製品に関する情報

1)製品名
2)バージョン
3)製品のシリアル No.
4)BPL もしくは DLL ファイルのタイムスタンプ

開発環境および使用機器に関する情報

1)OS
2)開発コンテナ
3)利用形態(VCL or DLL)
4)スキャナもしくはデジタルカメラに関するご質問の場合は、TWAIN デバイスおよびドライバのバージョンなど
5)Web カメラに関するご質問の場合は、Web カメラの名称および DirectX のバージョンなど

ご質問の内容

どのようにしたらその症状が発生するのか、あるいはどのような事柄が聞きたいのか、できるだけ簡潔にご説明ください。文章で説明するのが難しい場合は、再現性のあるプログラムをメールにて添付していただいた方がよりスムーズに解決する場合もございますので、その場合はプログラムを LZH もしくは ZIP 形式で圧縮していただき、お送りください。
また、製品とは関係のないプログラミング技法や WindowsAPI などに関するご質問にはお答えいたしかねますので、予めご了承ください。

問い合わせ先

「お使いになる前に」(パッケージ版は同梱の折込案内、ダウンロード版は別の PDF ファイル)の**サポートについて**をご覧ください。

注意

1PC 開発ライセンスはユーザ登録された方がサポート対象となります。
PC 開発ライセンスパックはユーザ登録された方もしくは同一法人の方(法人登録の場合)がサポート対象となります。

3. インストール

●はじめに
ImageKit10 VCL を使用するために必要なものが二つあります。
(1)「.NET Framewrok 4.0」: ライセンス認証プログラムを実行するために必要
(2)「Microsoft Visual C++ 2019 再頒布可能パッケージ」: ImageKit10 VCL の DLL を使用するために必要
*1 「.NET Framewrok 4.0」は ImageKit10 VCL をインストールする前に適用してください。
*2 (1)(2)とも(x86),(x64)の 2 種類ありますので用途に応じて選択してください。

●インストール方法
管理者権限で該当する開発環境に応じたインストールプログラムを実行して(タスクバーのメニューから「ファイル名を指定して実行」など)、指示される説明に従ってインストールを進めてください。

●インストール後のフォルダの構成
ImageKit10 VCL のインストール後のフォルダ構成は以下の通りです。(ここでは Delphi XE3 を例に上げておりますので、他の開発環境の場合は Delphi XE3 のところを差し替えてお読みください。)

■インストール時指定したフォルダ
```
|          ReadMe
├────■Activation
|          ライセンス認証プログラム(.NET Framewrok 4.0 が必須)
├────■DelphiXE3
|          Delphi XE3 用のモジュール
├────■Help
|          ヘルプファイル
├────■Include
|          DLL 用関数定義ファイル
├────■Manual
|          PDF ファイル
└────■Samples
           サンプルプログラム
```

●DLL を利用する場合
インストール時指定したフォルダ¥XXXX¥System の DLL ファイルをパスの通ったフォルダもしくはアプリケーションと同じフォルダにコピーし、Include フォルダの DLL 用関数定義ファイルをご利用ください(XXXX は C++Builder XE3 や Delphi XE3 など)。64 ビットの DLL ファイルも含まれますので作成するアプリケーションに応じてご利用ください。

●コンポーネントを利用する場合
1.旧バージョンの削除
ImageKit6 VCL/ImageKit7 VCL/ImageKit8 VCL/ImageKit9 VCL のいずれかがインストールされている場合は、「コンポーネント」ー「パッケージのインストール」メニューから ImageKit6 VCL/ImageKit7 VCL/ImageKit8 VCL/ImageKit9 VCL のコンポーネントを削除してください。
削除するタイミングはImageKit10 VCLをインストールする前でも後でも構いませんが、ImageKit10 VCLのコンポーネントを開発環境に組み込む前に行ってください。

2.開発環境への組み込み
コンポーネントを利用して開発を行うには、開発環境へコンポーネントを組み込む必要があります。(手順は各開発環境の 1 から順番に実行してください。)
下記で説明している「Windows のシステムフォルダ」は、Windows¥System32 フォルダを指し、Delphi と C++Builder の Bin、Include、Lib フォルダとは Delphi や C++Builder をインストールしたフォルダの階層化にあるフォルダを指します。
例:Delphi XE3 を C:¥Program Files (x86)¥Embarcadero¥RAD Studio¥10.0 にインストールした場合
Bin フォルダは C:¥Program Files (x86)¥Embarcadero¥RAD Studio¥10.0¥bin
Lib フォルダは C:¥Program Files (x86)¥Embarcadero¥RAD Studio¥10.0¥lib

Delphi XE3:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥DelphiXE3¥Bin フォルダ(以下 DelphiXE3¥Bin、他も同様)のファイルを Delphi

XE3 の bin フォルダに、DelphiXE3¥Lib¥Win64 フォルダのファイルを Delphi XE3 の lib¥win64¥release フォルダに、DelphiXE3¥Lib¥Win32 フォルダのファイルを Delphi XE3 の lib¥win32¥release フォルダに、DelphiXE3¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、DelphiXE3¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。

2. Delphi XE3(もしくは Embarcadero RAD Studio XE3)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi XE3 の bin フォルダにコピーした DclIk10D17.bpl,DclIk10ThumbD17.bpl,DclIk10WebCamD17.bpl を指定し、「開く」ボタンを選択します(成功すると、実行時パッケージにもパッケージが登録されます)。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合

A の場合と同じ手順で OK ですが、A の 1 番で記載しております DelphiXE3¥Lib¥Win64 フォルダのファイルと DelphiXE3¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、DelphiXE3¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi XE4:

A.64 ビット Windows にインストールしている場合

1. インストール時指定したフォルダ¥DelphiXE4¥Bin フォルダ(以下 DelphiXE4¥Bin、他も同様)のファイルを Delphi XE4 の bin フォルダに、DelphiXE4¥Lib¥Win64 フォルダのファイルを Delphi XE4 の lib¥win64¥release フォルダに、DelphiXE4¥Lib¥Win32 フォルダのファイルを Delphi XE4 の lib¥win32¥release フォルダに、DelphiXE4¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、DelphiXE4¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi XE4(もしくは Embarcadero RAD Studio XE4)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi XE4 の bin フォルダにコピーした DclIk10D18.bpl,DclIk10ThumbD18.bpl,DclIk10WebCamD18.bpl を指定し、「開く」ボタンを選択します(成功すると、実行時パッケージにもパッケージが登録されます)。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合

A の場合と同じ手順で OK ですが、A の 1 番で記載しております DelphiXE4¥Lib¥Win64 フォルダのファイルと DelphiXE4¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、DelphiXE4¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi XE5:

A.64 ビット Windows にインストールしている場合

1. インストール時指定したフォルダ¥DelphiXE5¥Bin フォルダ(以下 DelphiXE5¥Bin、他も同様)のファイルを Delphi XE5 の bin フォルダに、DelphiXE5¥Lib¥Win64 フォルダのファイルを Delphi XE5 の lib¥win64¥release フォルダに、DelphiXE5¥Lib¥Win32 フォルダのファイルを Delphi XE5 の lib¥win32¥release フォルダに、DelphiXE5¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、DelphiXE5¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi XE5(もしくは Embarcadero RAD Studio XE5)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi XE5 の bin フォルダにコピーした DclIk10D19.bpl,DclIk10ThumbD19.bpl,DclIk10WebCamD19.bpl を指定し、「開く」ボタンを選択します(成功すると、実行時パッケージにもパッケージが登録されます)。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合

A の場合と同じ手順で OK ですが、A の 1 番で記載しております DelphiXE5¥Lib¥Win64 フォルダのファイルと DelphiXE5¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、DelphiXE5¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi XE6:

A.64 ビット Windows にインストールしている場合

1. インストール時指定したフォルダ¥DelphiXE6¥Bin フォルダ(以下 DelphiXE6¥Bin、他も同様)のファイルを Delphi XE6 の bin フォルダに、DelphiXE6¥Lib¥Win64 フォルダのファイルを Delphi XE6 の lib¥win64¥release フォルダに、DelphiXE6¥Lib¥Win32 フォルダのファイルを Delphi XE6 の lib¥win32¥release フォルダに、

DelphiXE6¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、DelphiXE6¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi XE6（もしくは Embarcadero RAD Studio XE6）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi XE6 の bin フォルダにコピーした DclIk10D20.bpl,DclIk10ThumbD20.bpl,DclIk10WebCamD20.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております DelphiXE6¥Lib¥Win64 フォルダのファイルと DelphiXE6¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、DelphiXE6¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi XE7：
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥DelphiXE7¥Bin フォルダ（以下 DelphiXE7¥Bin、他も同様）のファイルを Delphi XE7 の bin フォルダに、DelphiXE7¥Lib¥Win64 フォルダのファイルを Delphi XE7 の lib¥win64¥release フォルダに、DelphiXE7¥Lib¥Win32 フォルダのファイルを Delphi XE7 の lib¥win32¥release フォルダに、DelphiXE7¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、DelphiXE7¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi XE7（もしくは Embarcadero RAD Studio XE7）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi XE7 の bin フォルダにコピーした DclIk10D21.bpl,DclIk10ThumbD21.bpl,DclIk10WebCamD21.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております DelphiXE7¥Lib¥Win64 フォルダのファイルと DelphiXE7¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、DelphiXE7¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi XE8：
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥DelphiXE8¥Bin フォルダ（以下 DelphiXE8¥Bin、他も同様）のファイルを Delphi XE8 の bin フォルダに、DelphiXE8¥Lib¥Win64 フォルダのファイルを Delphi XE8 の lib¥win64¥release フォルダに、DelphiXE8¥Lib¥Win32 フォルダのファイルを Delphi XE8 の lib¥win32¥release フォルダに、DelphiXE8¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、DelphiXE8¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi XE8（もしくは Embarcadero RAD Studio XE8）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi XE8 の bin フォルダにコピーした DclIk10D22.bpl,DclIk10ThumbD22.bpl,DclIk10WebCamD22.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております DelphiXE8¥Lib¥Win64 フォルダのファイルと DelphiXE8¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、DelphiXE8¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi 10 Seattle：
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥Delphi10Seattle¥Bin フォルダ（以下 Delphi10Seattle¥Bin、他も同様）のファイルを Delphi 10 Seattle の bin フォルダに、Delphi10Seattle¥Lib¥Win64 フォルダのファイルを Delphi 10 Seattle の lib¥win64¥release フォルダに、Delphi10Seattle¥Lib¥Win32 フォルダのファイルを Delphi 10 Seattle の lib¥win32¥release フォルダに、Delphi10Seattle¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、Delphi10Seattle¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。

2. Delphi 10 Seattle(もしくは Embarcadero RAD Studio 10 Seattle)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi 10 Seattle の bin フォルダにコピーした DclIk10D23.bpl,DclIk10ThumbD23.bpl,DclIk10WebCamD23.bpl を指定し、「開く」ボタンを選択します(成功すると、実行時パッケージにもパッケージが登録されます)。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております Delphi10Seattle¥Lib¥Win64 フォルダのファイルと Delphi10Seattle¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、Delphi10Seattle¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi 10.1 Berlin:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥Delphi10_1Berlin¥Bin フォルダ(以下 Delphi10_1Berlin¥Bin、他も同様)のファイルを Delphi 10.1 Berlin の bin フォルダに、Delphi10_1Berlin¥Lib¥Win64 フォルダのファイルを Delphi 10.1 Berlin の lib¥win64¥release フォルダに、Delphi10_1Berlin¥Lib¥Win32 フォルダのファイルを Delphi 10.1 Berlin の lib¥win32¥release フォルダに、Delphi10_1Berlin¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、Delphi10_1Berlin¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi 10.1 Berlin(もしくは Embarcadero RAD Studio 10.1 Berlin)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi 10.1 Berlin の bin フォルダにコピーした DclIk10D24.bpl,DclIk10ThumbD24.bpl,DclIk10WebCamD24.bpl を指定し、「開く」ボタンを選択します(成功すると、実行時パッケージにもパッケージが登録されます)。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております Delphi10_1Berlin¥Lib¥Win64 フォルダのファイルと Delphi10_1Berlin¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、Delphi10_1Berlin¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi 10.2 Tokyo:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥Delphi10_2Tokyo¥Bin フォルダ(以下 Delphi10_2Tokyo¥Bin、他も同様)のファイルを Delphi 10.2 Tokyo の bin フォルダに、Delphi10_2Tokyo¥Lib¥Win64 フォルダのファイルを Delphi 10.2 Tokyo の lib¥win64¥release フォルダに、Delphi10_2Tokyo¥Lib¥Win32 フォルダのファイルを Delphi 10.2 Tokyo の lib¥win32¥release フォルダに、Delphi10_2Tokyo¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、Delphi10_2Tokyo¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi 10.2 Tokyo(もしくは Embarcadero RAD Studio 10.2 Tokyo)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi 10.2 Tokyo の bin フォルダにコピーした DclIk10D25.bpl,DclIk10ThumbD25.bpl,DclIk10WebCamD25.bpl を指定し、「開く」ボタンを選択します(成功すると、実行時パッケージにもパッケージが登録されます)。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております Delphi10_2Tokyo¥Lib¥Win64 フォルダのファイルと Delphi10_2Tokyo¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、Delphi10_2Tokyo¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

Delphi 10.3 Rio:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥Delphi10_3Rio¥Bin フォルダ(以下 Delphi10_3Rio¥Bin、他も同様)のファイルを Delphi 10.3 Rio の bin フォルダに、Delphi10_3Rio¥Lib¥Win64 フォルダのファイルを Delphi 10.3 Rio の lib¥win64¥release フォルダに、Delphi10_3Rio¥Lib¥Win32 フォルダのファイルを Delphi 10.3 Rio の lib¥win32¥release フォルダに、Delphi10_3Rio¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、Delphi10_3Rio¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. Delphi 10.3 Rio(もしくは Embarcadero RAD Studio 10.3 Rio)を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。

3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、Delphi 10.3 Rio の bin フォルダにコピーした DclIk10D26.bpl,DclIk10ThumbD26.bpl,DclIk10WebCamD26.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、ツールパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております Delphi10_3Rio¥Lib¥Win64 フォルダのファイルと Delphi10_3Rio¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、Delphi10_3Rio¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder XE3:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++BuilderXE3¥Bin フォルダ（以下 C++BuilderXE3¥Bin、他も同様）のファイルを C++Builder XE3 の bin フォルダに、C++BuilderXE3¥Include フォルダのファイルを C++Builder XE3 の include¥windows¥vcl フォルダに、C++BuilderXE3¥Lib¥Win64 フォルダのファイルを C++Builder XE3 の lib¥win64¥release フォルダに、C++BuilderXE3¥Lib¥Win32 フォルダのファイルを C++Builder XE3 の lib¥win32¥release フォルダに、C++BuilderXE3¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++BuilderXE3¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder XE3（もしくは Embarcadero RAD Studio XE3）を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder XE3 の bin フォルダにコピーした DclIk10D17.bpl,DclIk10ThumbD17.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++BuilderXE3¥Lib¥Win64 フォルダのファイルと C++BuilderXE3¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++BuilderXE3¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder XE4:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++BuilderXE4¥Bin フォルダ（以下 C++BuilderXE4¥Bin、他も同様）のファイルを C++Builder XE4 の bin フォルダに、C++BuilderXE4¥Include フォルダのファイルを C++Builder XE4 の include¥windows¥vcl フォルダに、C++BuilderXE4¥Lib¥Win64 フォルダのファイルを C++Builder XE4 の lib¥win64¥release フォルダに、C++BuilderXE4¥Lib¥Win32 フォルダのファイルを C++Builder XE4 の lib¥win32¥release フォルダに、C++BuilderXE4¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++BuilderXE4¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder XE4（もしくは Embarcadero RAD Studio XE4）を起動し「コンポーネント」ー「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder XE4 の bin フォルダにコピーした DclIk10D18.bpl,DclIk10ThumbD18.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++BuilderXE4¥Lib¥Win64 フォルダのファイルと C++BuilderXE4¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++BuilderXE4¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder XE5:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++BuilderXE5¥Bin フォルダ（以下 C++BuilderXE5¥Bin、他も同様）のファイルを C++Builder XE5 の bin フォルダに、C++BuilderXE5¥Include フォルダのファイルを C++Builder XE5 の include¥windows¥vcl フォルダに、C++BuilderXE5¥Lib¥Win64 フォルダのファイルを C++Builder XE5 の lib¥win64¥release フォルダに、C++BuilderXE5¥Lib¥Win32 フォルダのファイルを C++Builder XE5 の lib¥win32¥release フォルダに、C++BuilderXE5¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++BuilderXE5¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder XE5（もしくは Embarcadero RAD Studio XE5）を起動し「コンポーネント」ー「パッケージのインストール」メ

ニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder XE5 の bin フォルダにコピーした DcllIk10D19.bpl,DcllIk10ThumbD19.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++BuilderXE5¥Lib¥Win64 フォルダのファイルと C++BuilderXE5¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++BuilderXE5¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder XE6:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++BuilderXE6¥Bin フォルダ（以下 C++BuilderXE6¥Bin、他も同様）のファイルを C++Builder XE6 の bin フォルダに、C++BuilderXE6¥Include フォルダのファイルを C++Builder XE6 の include¥windows¥vcl フォルダに、C++BuilderXE6¥Lib¥Win64 フォルダのファイルを C++Builder XE6 の lib¥win64¥release フォルダに、C++BuilderXE6¥Lib¥Win32 フォルダのファイルを C++Builder XE6 の lib¥win32¥release フォルダに、C++BuilderXE6¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++BuilderXE6¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder XE6（もしくは Embarcadero RAD Studio XE6）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder XE6 の bin フォルダにコピーした DcllIk10D20.bpl,DcllIk10ThumbD20.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++BuilderXE6¥Lib¥Win64 フォルダのファイルと C++BuilderXE6¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++BuilderXE6¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder XE7:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++BuilderXE7¥Bin フォルダ（以下 C++BuilderXE7¥Bin、他も同様）のファイルを C++Builder XE7 の bin フォルダに、C++BuilderXE7¥Include フォルダのファイルを C++Builder XE7 の include¥windows¥vcl フォルダに、C++BuilderXE7¥Lib¥Win64 フォルダのファイルを C++Builder XE7 の lib¥win64¥release フォルダに、C++BuilderXE7¥Lib¥Win32 フォルダのファイルを C++Builder XE7 の lib¥win32¥release フォルダに、C++BuilderXE7¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++BuilderXE7¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder XE7（もしくは Embarcadero RAD Studio XE7）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder XE7 の bin フォルダにコピーした DcllIk10D21.bpl,DcllIk10ThumbD21.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++BuilderXE7¥Lib¥Win64 フォルダのファイルと C++BuilderXE7¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++BuilderXE7¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder XE8:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++BuilderXE8¥Bin フォルダ（以下 C++BuilderXE8¥Bin、他も同様）のファイルを C++Builder XE8 の bin フォルダに、C++BuilderXE8¥Include フォルダのファイルを C++Builder XE8 の include¥windows¥vcl フォルダに、C++BuilderXE8¥Lib¥Win64 フォルダのファイルを C++Builder XE8 の lib¥win64¥release フォルダに、C++BuilderXE8¥Lib¥Win32 フォルダのファイルを C++Builder XE8 の lib¥win32¥release フォルダに、C++BuilderXE8¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++BuilderXE8¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。

2. C++Builder XE8（もしくは Embarcadero RAD Studio XE8）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder XE8 の bin フォルダにコピーした DclIk10D22.bpl,DclIk10ThumbD22.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++BuilderXE8¥Lib¥Win64 フォルダのファイルと C++BuilderXE8¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++BuilderXE8¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder 10 Seattle:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++Builder10Seattle¥Bin フォルダ（以下 C++Builder10Seattle¥Bin、他も同様）のファイルを C++Builder 10 Seattle の bin フォルダに、C++Builder10Seattle¥Include フォルダのファイルを C++Builder 10 Seattle の include¥windows¥vcl フォルダに、C++Builder10Seattle¥Lib¥Win64 フォルダのファイルを C++Builder 10 Seattle の lib¥win64¥release フォルダに、C++Builder10Seattle¥Lib¥Win32 フォルダのファイルを C++Builder 10 Seattle の lib¥win32¥release フォルダに、C++Builder10Seattle¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++Builder10Seattle¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder 10 Seattle（もしくは Embarcadero RAD Studio 10 Seattle）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder 10 Seattle の bin フォルダにコピーした DclIk10D23.bpl,DclIk10ThumbD23.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++Builder10Seattle¥Lib¥Win64 フォルダのファイルと C++Builder10Seattle¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++Builder10Seattle¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder 10.1 Berlin:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++Builder10_1Berlin¥Bin フォルダ（以下 C++Builder10_1Berlin¥Bin、他も同様）のファイルを C++Builder 10.1 Berlin の bin フォルダに、C++Builder10_1Berlin¥Include フォルダのファイルを C++Builder 10.1 Berlin の include¥windows¥vcl フォルダに、C++Builder10_1Berlin¥Lib¥Win64 フォルダのファイルを C++Builder 10.1 Berlin の lib¥win64¥release フォルダに、C++Builder10_1Berlin¥Lib¥Win32 フォルダのファイルを C++Builder 10.1 Berlin の lib¥win32¥release フォルダに、C++Builder10_1Berlin¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++Builder10_1Berlin¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder 10.1 Berlin（もしくは Embarcadero RAD Studio 10.1 Berlin）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder 10.1 Berlin の bin フォルダにコピーした DclIk10D24.bpl,DclIk10ThumbD24.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合
A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++Builder10_1Berlin¥Lib¥Win64 フォルダのファイルと C++Builder10_1Berlin¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++Builder10_1Berlin¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder 10.2 Tokyo:
A.64 ビット Windows にインストールしている場合
1. インストール時指定したフォルダ¥C++Builder10_2Tokyo¥Bin フォルダ（以下 C++Builder10_2Tokyo¥Bin、他も同様）のファイルを C++Builder 10.2 Tokyo の bin フォルダに、C++Builder10_2Tokyo¥Include フォルダのファイルを C++Builder 10.2 Tokyo の include¥windows¥vcl フォルダに、C++Builder10_2Tokyo¥Lib¥Win64 フォルダのファイル

を C++Builder 10.2 Tokyo の lib¥win64¥release フォルダに、C++Builder10_2Tokyo¥Lib¥Win32 フォルダのファイルを C++Builder 10.2 Tokyo の lib¥win32¥release フォルダに、C++Builder10_2Tokyo¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++Builder10_2Tokyo¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。

2. C++Builder 10.2 Tokyo（もしくは Embarcadero RAD Studio 10.2 Tokyo）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder 10.2 Tokyo の bin フォルダにコピーした Dcllk10D25.bpl,Dcllk10ThumbD25.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合

A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++Builder10_2Tokyo¥Lib¥Win64 フォルダのファイルと C++Builder10_2Tokyo¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++Builder10_2Tokyo¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

C++Builder 10.3 Rio:

A.64 ビット Windows にインストールしている場合

1. インストール時指定したフォルダ¥C++Builder10_3Rio¥Bin フォルダ（以下 C++Builder10_3Rio¥Bin、他も同様）のファイルを C++Builder 10.3 Rio の bin フォルダに、C++Builder10_3Rio¥Include フォルダのファイルを C++Builder 10.3 Rio の include¥windows¥vcl フォルダに、C++Builder10_3Rio¥Lib¥Win64 フォルダのファイルを C++Builder 10.3 Rio の lib¥win64¥release フォルダに、C++Builder10_3Rio¥Lib¥Win32 フォルダのファイルを C++Builder 10.3 Rio の lib¥win32¥release フォルダに、C++Builder10_3Rio¥System¥Win64 フォルダのファイルを Windows のシステムフォルダに、C++Builder10_3Rio¥System¥Win32 フォルダのファイルを SysWOW64 フォルダにそれぞれコピーします。
2. C++Builder 10.3 Rio（もしくは Embarcadero RAD Studio 10.3 Rio）を起動し「コンポーネント」－「パッケージのインストール」メニューを選択します。
3. 「設計時パッケージ」の「追加(A)」ボタンを選択し、C++Builder 10.3 Rio の bin フォルダにコピーした Dcllk10D26.bpl,Dcllk10ThumbD26.bpl を指定し、「開く」ボタンを選択します（成功すると、実行時パッケージにもパッケージが登録されます）。
4. 「OK」ボタンを選択してダイアログを閉じると、コンポーネントパレットの「ImageKit」にアイコンが表示されます。

B.32 ビット Windows にインストールしている場合

A の場合と同じ手順で OK ですが、A の 1 番で記載しております C++Builder10_3Rio¥Lib¥Win64 フォルダのファイルと C++Builder10_3Rio¥System¥Win64 フォルダ内に含まれるファイルをコピーする必要はありません。また、C++Builder10_3Rio¥System¥Win32 フォルダのファイルは Windows のシステムフォルダにコピーします。

●ビルドバージョン

ImageKit10 VCL は次の環境でビルドしてあります。

Delphi XE3: Update 2
Delphi XE4: Update 1
Delphi XE5: Update 2 Hotfix 5
Delphi XE6: Update 1
Delphi XE7: Update 1
Delphi XE8: Subscription Update 1
Delphi 10 Seattle: Subscription Update 1
Delphi 10.1 Berlin:Update 2
Delphi 10.2 Tokyo: Update 3
Delphi 10.3 Rio: Update 3
C++Builder XE3: Update 2
C++Builder XE4: Update 1
C++Builder XE5: Update 2 Hotfix 5
C++Builder XE6: Update 1
C++Builder XE7: Update 1
C++Builder XE8: Subscription Update 1
C++Builder 10 Seattle: Subscription Update 1
C++Builder 10.1 Berlin: Update 2
C++Builder 10.2 Tokyo: Update 3
C++Builder 10.3 Rio: Update 3

●弊社 HP からアップデートモジュールをダウンロードしてモジュールを更新する場合

Delphi もしくは C++Builder を起動していない状態で、ダウンロードしたモジュールを既存のモジュールに上書きコピーします。

●アンインストール方法

(1)インストールプログラムを使用して最新版へ入れ替える場合

コントロールパネルの「プログラム」-「プログラムのアンインストール」で ImageKit10 VCL を選択して削除してください。
次に最新版のインストールプログラムを実行して ImageKit10 VCL をインストールしてください。その後で、既存のファイルに最新のファイルを上書きしてください。上書きに関してはインストール方法の各開発環境の 1 番を参照してください。

(2)コンピュータから抹消する場合

ライセンス認証プログラムを起動し解除手続きを行ってください。オンラインで解除手続きを行う場合はアンインストール時にも行うことができます。
VCL でご利用の場合、ご利用の開発環境から ImageKit10 VCL のパッケージの削除を行ってください。
コントロールパネルの「プログラム」-「プログラムのアンインストール」で ImageKit10 VCL を選択して削除してください。
コンピュータから完全に ImageKit10 VCL 関連のファイルを削除される場合は、アンインストール後に Windows¥System32 や Windows¥SysWOW64 フォルダから ImageKit10 VCL の実行時用パッケージファイル（拡張子 bpl）と dll ファイルを削除してください。実行時用パッケージファイルと dll ファイルについては**コンポーネントリファレンス**の「作成したアプリケーションの配布」の実行時パッケージのところをご覧ください。また、各開発環境の bin や include および lib フォルダにコピーした ImageKit10 VCL のファイルも削除してください。ファイルの更新時刻が 10:00 のものが対象でファイル名は Dcllk10*.*、Ik*.hpp、ImageKit*.hpp、Ik*.res、Ik*.dfm、ImageKit*.*、ImgKit10*.dcp です。

※コントロールパネル内の表現については Windows 10 などを参考にしております。

4. サンプルプログラムについて

インストール時指定したフォルダの階層化の¥Samples の中に対応コンテナのフォルダがあります。
お使いのコンテナに応じて該当するプロジェクトファイルを使用して、動作を確認してください。

スキャン関連のサンプルを実行する場合には、TWAIN ドライバが必要になりますので、下記の説明をお読みください。

●スキャンデバイスメーカ提供の TWAIN ドライバのインストール
（スキャンデバイスを利用される場合）

ご使用のイメージスキャナやデジタルカメラ、フィルムスキャナに添付されているディスクまたは CD-ROM などより、ご使用のコンピュータへ「TWAIN ドライバ」のインストールが行われていることが前提となります。
既に、インストールされている場合は、そのままお使いください。
TWAIN ドライバのインストールについては、メーカのマニュアルなどをご覧ください。

なお、TWAIN ドライバに関して、例えば 16 ビット用のドライバしかメーカが提供していない場合でも、32 ビットで使用できるような TWAIN の 32 ビット、16 ビット相互コンバータ(Twunk_32.exe,Twunk_16.exe)がメーカにより用意されていますので、詳しくは各メーカへお問い合わせください。

また、取り込み中にリソースが減少するなどの現象が発生した場合は、
1.TWAIN ドライバを最新のものに更新する
2.Twain_32.dll を最新のものに更新する
3.Twunk_32.exe,Twunk_16.exe の二つのファイル名をリネームする
の順にそれぞれのケースでお試しください。ただし、2 番と 3 番については Windows のバージョンによっては有効にならない場合があります。

64 ビット Windows で 64 ビット TWAIN ドライバをご利用される場合は、TWAINDSM.dll が必要です。TWAIN ドライバのインストールを行っても該当ファイルが存在しない場合は、twain.org の Web サイトより 64 ビット版の TWAINDSM.dll をダウンロードしてください。

5. イメージとメモリハンドルの操作について

ImageKit10 VCL は、イメージをメモリ上の「メモリハンドル」として操作・管理します。
「メモリハンドル」は、メモリ上に保存してあるイメージひとつひとつと関連づけられた「識別番号」の様なものと考えてください。
ImageKit10 VCL は、この「メモリハンドル」を介して実際のイメージを操作するわけです。
次に**イメージキットコントロール(TVImageKit)**を中心にそれぞれの場合について簡単に解説いたします。

1.基本イメージと階層イメージについて
基本イメージ:ImageKit.ImageHandle
階層イメージ:ImageKit.Layer[n].ImageHandle　　(n:0～99)
基本イメージにイメージが設定してある場合のみ、階層イメージが使用できます。
基本イメージを使用する場合は ImageKit.LayerNo に-1 を設定し、階層イメージを使用する場合は 0～99 の値を設定します。
デフォルトは-1 です。

基本イメージにイメージを読み込む

```
VImageKit1.LayerNo := -1;
VImageKit1.FileIO.FileName := 'イメージファイル';
VImageKit1.FileIO.LoadFile(vikLoad);
```

階層イメージ(0 番目)にイメージを読み込む

```
VImageKit1.LayerNo := 0;
VImageKit1.FileIO.FileName := 'イメージファイル';
VImageKit1.FileIO.LoadFile(vikLoad);
```

LayerNo プロパティが有効なメソッドは
(1)ImageKit の次のメソッド
GetFromClipBrd()
SetToClipBrd()
DibFromBitmap()
BitmapFromDib()
(2)ImageKit.FileIO の次のメソッド
LoadFile(), SaveFile()
LoadFileMem(), SaveFileMem()
RGBBmpPlaneFileLoad(), RGBBmpPlaneFileSave()
CMYKBmpPlaneFileLoad(), CMYKBmpPlaneFileSave()
YCCBmpPlaneFileLoad(), YCCBmpPlaneFileSave()
(3)ImageKit.Effect の全メソッド
となります。

2. 明示的なイメージメモリの解放は不要
例えば下記のコードで、イメージファイルの読み込みを行う場合(VImageKit1.FileIO.LoadFile の実行)、その都度前回読み込んだイメージのメモリハンドルが自動的に解放され、新しく読み込んだイメージのメモリハンドルが設定されます。

```
(Delphi でのイメージファイルの読み込みの例)
procedure TForm1.Button1Click(Sender: TObject);
begin
    if VImageKit1.FileIO.OpenFileDialog = False then        //ファイルの選択
        Exit;
    VImageKit1.DisplayMode := vikScale;                     //イメージの表示
    VImageKit1.FileIO.LoadFile(vikLoad);                    //イメージファイルの読み込み
    VImageKit1.DisplayImage();                              //イメージの表示
end;
```

従って、使用メモリ量がどんどん増えることはありません。

次のコードは

```
VImageKit1.ImageHandle := VImageKit1.CopyImage(ImageHandle)
```

(1)以前の VImageKit1.ImageHandle を解放する

(2)VImageKit1.ImageHandle に新たに ImageHandle の複製のイメージブロックのハンドルをセットする
となります。

また次のコードは
VImageKit1.ImageHandle := ImageHandle
(1)以前の VImageKit1.ImageHandle を解放する
(2)VImageKit1.ImageHandle に新たに ImageHandle のハンドルをセットする
となります。

なお、次のコードは(通常は行いませんが)
VImageKit1.ImageHandle := VImageKit1.ImageHandle
(1)以前の VImageKit1.ImageHandle を解放する
(2)VImageKit1.ImageHandle に無効になった VImageKit1.ImageHandle をセットする
となり、無効なハンドルが設定されます。

3. エフェクト処理
エフェクト処理を行うと ImageKit.ImageHandle が更新されます。従ってオリジナルのイメージデータが必要な場合は
Image = ImageKit.CopyImage() で ImageKit.ImageHandle の複製を作成しておいてください。
なお、CopyImage メソッドの引数を省略した場合は LayerNo プロパティに応じたメモリハンドルの複製を指示します(LayerNo = −1 であれば ImageKit.ImageHandle の複製となります)。

```
VImageKit1.LayerNo := -1;
VImageKit1.Effect.SelectMode := vikEffectAll;
VImageKit1.DisplayMode := vikScale;
ret := VImageKit1.Effect.Emboss(0, 3, 128);          //エンボス処理
VImageKit1.DisplayImage();
```

また、CopyImage メソッドによって作成したイメージは必要に応じて VImageKit.FreeMemory メソッドで解放します。
(不要になったメモリをそのまま放置しておきますとメモリ不足の原因になりますので、不要になったら適宜メモリを解放してください。)

4. スキャンデバイスからのイメージの取り込み
＜コンポーネントを使用＞
取り込み実行 → イベント発生 → メモリのコピー or ファイルに保存
＜DLL を使用＞
取り込み実行 → ユーザ関数発生 → メモリのコピー or ファイルに保存

読み取り実行時のAfterScanイベント(コンポーネントを使用)やユーザ関数(DLLを使用)の中で引数となるメモリは、イベントやユーザ関数実行後ImageKit10 VCLで引数のメモリを解放するため、予めイベントやユーザ関数でメモリのコピーを作成するか、もしくはファイルに保存する必要があります。

5. メモリの参照とコピー
＜参照＞
A ・・・ イメージのメモリ
B = A

とするとAとBが同じ1つのイメージのメモリを指し、A → イメージのメモリ ← Bとなります。

(VCL)

```
VImageKit1.ImageHandle := B;
VImageKit1.FreeMemory; (メソッドの引数を省略した場合は LayerNo プロパティに応じたメモリハンドルを指します)
```

(DLL)

```
IKFreeMemory(B)
```

とするとBのメモリを解放することにより、Aから参照できなくなります。
また、**メモリの 2 重解放**を防ぐために解放した後にAとBにそれぞれ0を代入するようにしてください。

＜コピー＞
A ・・・ イメージのメモリ
(VCL)

```
B := VImageKit1.CopyImage(A);
```

(DLL)

B := IKCopyImage(A);

A → イメージのメモリ

B → 新しく作成されたイメージのメモリ

この様にコピーを行うと、A と B は独立したメモリとなります。
そのため、不要になった場合は A と B の両方を解放する必要があります。

【参考】

ImageKit10 VCL で管理するラスタイメージのメモリハンドルの内容は、デバイス独立ビットマップ（DIB）へのポインタのハンドルです。メモリハンドルの値を Windows API の GlobalLock 関数の引数にした戻り値が、DIB へのポインタとなります。

デバイス独立ビットマップ（DIB）の形式を以下に示します。

詳細は、各関連マニュアル、書籍などをご覧ください。

6. エフェクト処理の使用例

エフェクト処理を使用すると、ラスタイメージに対して高度な処理が行うことができます。
次に代表的なエフェクト処理の例を記載します。

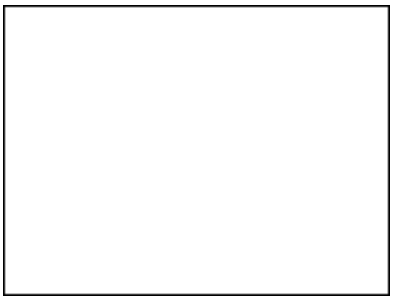
○選択処理 (SelectImage)

ImageKit10 ではイメージを選択する方法として、全体、多角形、円形の 3 パターンがあります。いずれの場合もイメージとマスクを生成し、VCL の場合はイメージキットコントロールでは ImageHandle プロパティもしくは Layer プロパティ内の ImageHandle プロパティに、DLL の場合は IKSELECT_IMAGE 構造体のメンバー変数に出力されます。ここではそれぞれ 2 つの方法で選択した場合に、出力されるイメージとマスクを載せながら説明していきます。

・イメージ全体を選択する場合(SelectMode = 1)

イメージの出力

マスクの出力



※イメージ全体を選択する場合でも、マスクは出力されます。不要な場合はマスクハンドルを解放してください。

【Delphi の例(VCL)】

```
VImageKit1.LayerNo := -1;                          //基本イメージを処理
VImageKit1.ImageHandle := ImgHandle;               //イメージをセット
VImageKit1.Effect.MaskImageHandle := 0;
VImageKit1.Effect.SelectMode := vikEffectAll;      //選択モードを全体に設定
Ret := VImageKit1.Effect.SelectImage(255, 0, 0);   //選択処理実行
```

【Delphi の例(DLL)】

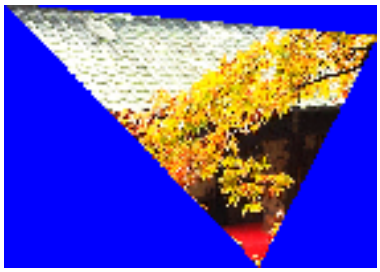
```
// 選択処理実行
IKSelectImageEx(ImgHandle, DstHandle, 1, pt(0), 0, rc, True, 255, 0, 0, 0, '選択', '選択', 'キャンセル');
IKFreeMemory(ImgHandle);// 入力ハンドルを解放、ハンドルを解放しないとイメージがいつまでもメモリに常駐します
```

・多角形で選択する場合(SelectMode = 2)

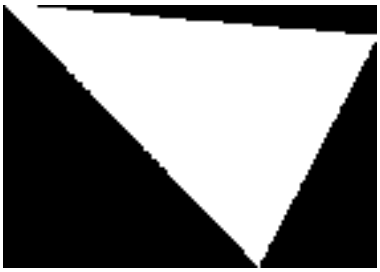
多角形の頂点の座標を引数に渡すことにより多角形選択ができます。
出力されるイメージとマスクのサイズは選択された最小の矩形になります。

・InOut が TRUE の場合

イメージの出力



マスクの出力

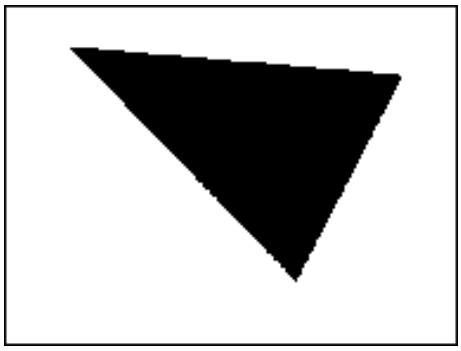


エフェクト処理の使用例

・InOut が FALSE の場合

イメージの出力

マスクの出力



※イメージサイズは 640×480 とします。

【Delphi の例(VCL)】

```
Pt: array[0..2] of TPoint;
Pt[0].X := 90; Pt[0].Y := 60;                    // ポイントの入力
Pt[1].X := 560; Pt[1].Y := 100;
Pt[2].X := 410; Pt[2].Y := 390;
VImageKit1.LayerNo := -1;                        // 基本イメージを処理
VImageKit1.ImageHandle := ImgHandle;             // イメージをセット
VImageKit1.Effect.MaskImageHandle := 0;
VImageKit1.Effect.InOut := vikInside;            // vikOutside,vikInside で選択できます
VImageKit1.Effect.SelectMode := vikEffectPolygon;      // 選択モードを多角形に設定
Ret = VImageKit1.Effect.SelectImage(Pt, 255, 0, 0);    // 選択処理実行
```

【Delphi の例(DLL)】

```
Pt: array[0..2] of TPoint;
Pt[0].X := 90; Pt[0].Y := 60;                    // ポイントの入力
Pt[1].X := 560; Pt[1].Y := 100;
Pt[2].X := 410; Pt[2].Y := 390;
// 選択処理実行
IKSelectImageEx(ImgHandle, DstHandle, 2, Pt(0), 3, rc, True, 255, 0, 0, 0, '選択', '選択', 'キャンセル');
IKFreeMemory(ImgHandle);          // 入力ハンドルを解放
                                  // ハンドルを解放しないとイメージがいつまでもメモリに常駐します
ImgHandle := 0;
```

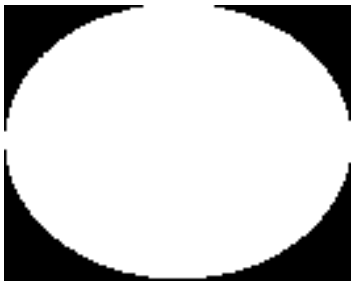
・円形で選択する場合(SelectMode = 3)

円に外接する矩形の頂点座標を VCL の場合は RectLeft・RectTop・RectRight・RectBottom プロパティに設定し、DLL の場合は RECT 構造体に入れ引数に渡すことにより円形選択ができます。
出力されるイメージとマスクのサイズは選択された最小の矩形になります。

・InOut が TRUE の場合

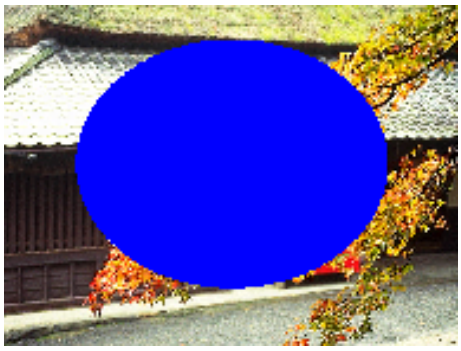
イメージの出力

マスクの出力

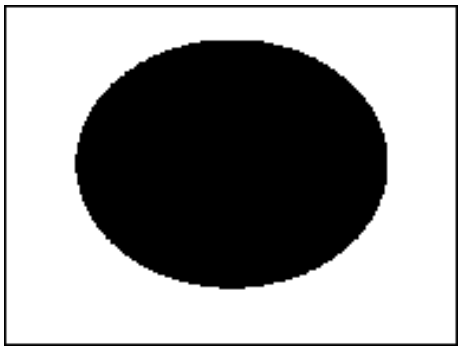


・InOut が FALSE の場合

イメージの出力



マスクの出力



※イメージサイズは 640×480 とします。

【Delphi の例(VCL)】

```
VImageKit1.Effect.RectLeft := 100;   VImageKit1.Effect.RectTop := 50;          // ポイントの入力
VImageKit1.Effect.RectRight := 540;  VImageKit1.Effect.RectBottom := 400;
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := ImgHandle;                    //イメージをセット
VImageKit1.Effect.MaskImageHandle := 0;
VImageKit1.Effect.InOut := vikInside;                   //ikOutside,ikInside で選択できます
VImageKit1.Effect.SelectMode := vikEffectEllipse;       //選択モードを円形に設定
Ret := VImageKit1.Effect.SelectImage(255, 0, 0)         //選択処理実行
```

【Delphi の例(DLL)】

```
rc: TRect;
rc.Left := 100;   rc.Top := 50;          // RECT 構造体に座標を代入
rc.Right := 540; rc.Bottom := 400;
//  選択処理実行
IKSelectImageEx(ImgHandle, DstHandle, 3, pt(0), 0, rc, True, 255, 0, 0, 0, '選択', '選択', 'キャンセル');
IKFreeMemory(ImgHandle); //  入力ハンドルを解放、ハンドルを解放しないとイメージがいつまでもメモリに常駐します
ImgHandle := 0;
```

○エンボス処理 (Emboss)

・マスクを使用して処理する場合(SelectMode = 0)

ImageKit10 ではマスクを使用してイメージの選択範囲を指定できます。
以降イメージハンドルを ImgHandle、マスクハンドルを MskHandle として説明いたします。
ImgHandle は 1,4,8,16,24,32 ビットイメージで処理の対象となるイメージです。
(Emboss の場合は 8G,16,24,32 ビットイメージ)
MskHandle は白黒 2 値のイメージで、白を選択ピクセル、黒を非選択ピクセルとする処理範囲を選択するイメージです。

	入力マスク	入力イメージ	出力イメージ
VCL	MaskImageHandle	ImageHandle	ImageHandle
DLL	hMskBmh	hImgBmh	DstHandle

【Delphi の例(VCL)】

```
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := ImgHandle;                    //イメージとマスクをセット
VImageKit1.Effect.MaskImageHandle := MskHandle;
VImageKit1.Effect.SelectMode := vikEffectMask;
```

エフェクト処理の使用例

```
Ret := VImageKit1.Effect.Emboss(0, 3, 128);     //エンボス処理実行
```

【Delphi の例(DLL)】

```
Src: IKSELECT_IMAGE;
Src.hImgBmh := ImgHandle;
Src.hMskBmh := MskHandle;
// エンボス処理実行
DstHandle := IKEmboss(Src, 0, pt(0), 0, rc, True, 0, 3, 128, 0, 'エンボス', 'エンボス', 'キャンセル');
IKFreeMemory(ImgHandle);            // 入力ハンドルを解放
IKFreeMemory(MskHandle);            // ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
ImgHandle := 0;
MskHandle := 0;
```

・イメージ全体を処理する場合(SelectMode = 1)

処理結果

【Delphi の例(VCL)】

```
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := ImgHandle;                    //イメージをセット
VImageKit1.Effect.MaskImageHandle := 0;
VImageKit1.Effect.SelectMode := vikEffectAll;
Ret := VImageKit1.Effect.Emboss(0, 3, 128);     //エンボス処理実行
```

【Delphi の例(DLL)】

```
Src: IKSELECT_IMAGE;
Src.hImgBmh := ImgHandle;
Src.hMskBmh := 0;
// エンボス処理実行
DstHandle := IKEmboss(Src, 1, pt(0), 0, rc, True, 0, 3, 128, 0, 'エンボス', 'エンボス', 'キャンセル');
IKFreeMemory(ImgHandle); // 入力ハンドル(イメージ)を解放、ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
ImgHandle := 0;
```

・イメージの一部を多角形で選択して処理する場合(SelectMode = 2)

処理結果

InOut = TRUE の時

処理結果

InOut = FALSE の時

※InOut が TRUE の場合は多角形の内側、FALSE の場合は多角形の外側を選択します。
※イメージサイズは 640×427 とします。

【Delphi の例(VCL)】

```
Pt: array[0..2] of TPoint;
Pt[0].X := 90; Pt[0].Y := 60;                    // ポイントの入力
Pt[1].X := 560; Pt[1].Y := 100;
Pt[2].X := 410; Pt[2].Y := 390;
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := ImgHandle;             //イメージをセット
VImageKit1.Effect.MaskImageHandle := 0;
VImageKit1.Effect.InOut := vikInside;            //vikInside,vikOutside で選択できます
VImageKit1.Effect.SelectMode := vikEffectPolygon;
Ret := VImageKit1.Effect.Emboss(Pt, 0, 3, 128)   //エンボス処理実行
```

【Delphi の例(DLL)】

```
Src: IKSELECT_IMAGE;
Pt: array[0..2] of TPoint;
Pt[0].X := 90; Pt[0].Y := 60;                    // ポイントの入力
Pt[1].X := 560; Pt[1].Y := 100;
Pt[2].X := 410; Pt[2].Y := 390;
Src.hImgBmh := ImgHandle;                        // 入力ハンドルにイメージをセット
Src.hMskBmh := 0;
// エンボス処理実行
DstHandle := IKEmboss(Src, 2, pt(0), 0, rc, True, 0, 3, 128, 0, 'エンボス', 'エンボス', 'キャンセル');
IKFreeMemory(ImgHandle);                         // 入力ハンドル(イメージ)を解放
                                                 // ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
ImgHandle := 0;
```

・イメージの一部を円形で選択して処理する場合(SelectMode = 3)

処理結果

InOut = TRUE の時

処理結果

InOut = FALSE の時

※InOut が TRUE の場合は多角形の内側、FALSE の場合は多角形の外側を選択します。
※イメージサイズは 640×427 とします。

【Delphi の例(VCL)】

```
VImageKit1.Effect.RectLeft := 100; VImageKit1.Effect.RectTop := 50;        //ポイントの入力
VImageKit1.Effect.RectRight := 540; VImageKit1.Effect.RectBottom := 400;
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := ImgHandle;                                       //イメージをセット
VImageKit1.Effect.MaskImageHandle := 0;
VImageKit1.Effect.InOut := vikInside;                                      //vikInside,vikOutside で選択できます
VImageKit1.Effect.SelectMode := vikEffectEllipse;
Ret := VImageKit1.Effect.Emboss(0, 3, 128);     //エンボス処理実行
```

【Delphi の例(DLL)】

```
Src: IKSELECT_IMAGE;
rc: TRect;
rc.Left := 100;  rc.Top := 50;          // RECT 構造体に座標を代入
rc.Right := 540; rc.Bottom := 400;
Src.hImgBmh := ImgHandle;                        // 入力ハンドルにイメージをセット
Src.hMskBmh := 0;
```

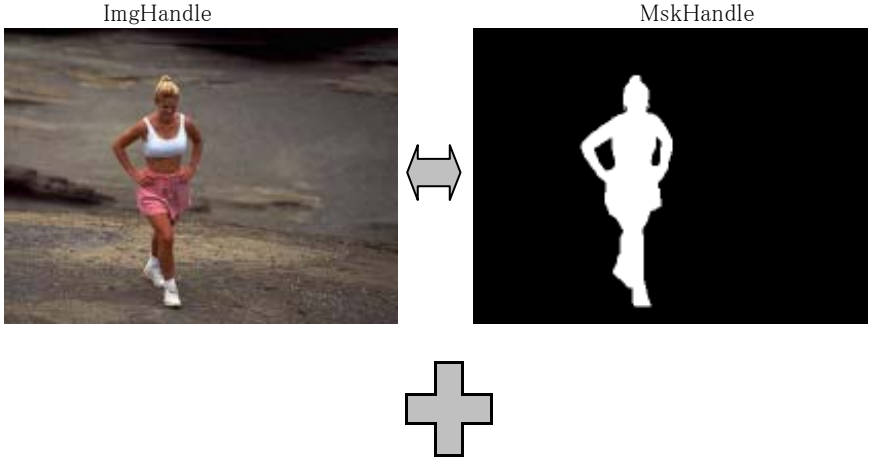
```
// エンボス処理実行
DstHandle := IKEmboss(Src, 3, pt(0), 0, rc, True, 0, 3, 128, 0, 'エンボス', 'エンボス', 'キャンセル');
IKFreeMemory(ImgHandle); // 入力ハンドル(イメージ)を解放、ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
ImgHandle := 0;
```

※Emboss はイメージとマスクのハンドルを受けて、一つのハンドルを返すという形で、下記のメソッド・関数と受け渡しの形は同じです。下記のメソッド・関数を使用の際にもこの説明を参考にしてください。

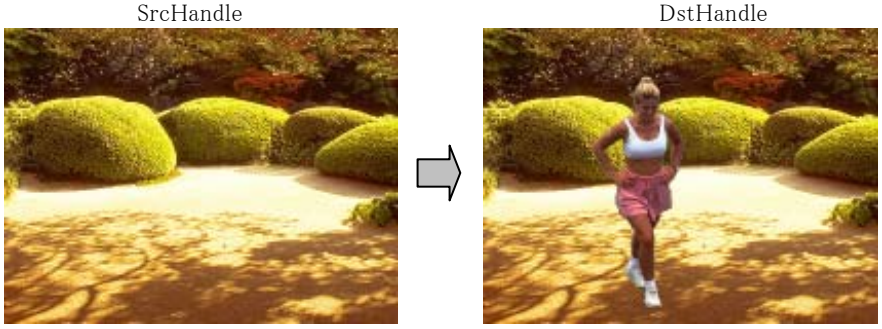
メソッド(関数)名	機能
AntiAlias(IKAntiAlias)	ラスタイメージのエッジを滑らかにします。
Blur(IKBlur)	ラスタイメージにぼかしを施します。
Canvas(IKCanvas)	ラスタイメージにキャンバス地効果を施します。
Chroma(IKChroma)	ラスタイメージの彩度を調整します。
CustomFilter(IKCustomFilter)	ラスタイメージに対してユーザ独自のフィルタ処理を施します。
GlassTile(IKGlassTile)	ラスタイメージにガラスタイル効果を施します。
Lens(IKLens)	ラスタイメージにレンズ効果を施します。
Mosaic(IKMosaic)	ラスタイメージに対してモザイク処理を施します。
MotionBlur(IKMotionBlur)	ラスタイメージにモーションぼかし効果を施します。
OilPaint(IKOilPaint)	ラスタイメージに油絵風効果を施します。
Outline(IKOutline)	ラスタイメージから輪郭部分を抽出します。
RemoveNoise(IKRemoveNoise)	ラスタイメージのノイズを除去します。
RGBGamma(IKRGBGamma)	ラスタイメージの RGB 値のガンマ補正を行います。
RGBLevel(IKRGBLevel)	ラスタイメージの RGB 値の加減処理を行います。
RGBRev(IKRGBRev)	ラスタイメージの RGB 値の反転処理を行います。
RGBSpline(IKRGBSpline)	ラスタイメージの RGB 値のスプライン補正を行います。
Ripple(IKRipple)	ラスタイメージに波紋効果を施します。
Sharp(IKSharp)	ラスタイメージの輪郭を強調してシャープにします。
UnifyColor(IKUnifyColor)	ラスタイメージの色のばらつきを修正します。
Waves(IKWaves)	ラスタイメージにさざ波効果を施します。
WhirlPinch(IKWhirlPinch)	ラスタイメージにねじりつまみ効果を施します。
YCCGamma(IKYCCGamma)	ラスタイメージの YCrCb 値のガンマ補正を行います。
YCCLevel(IKYCCLevel)	ラスタイメージの YCrCb 値の加減処理を行います。
YCCRev(IKYCCRev)	ラスタイメージの YCrCb 値の反転処理を行います。
YCCSpline(IKYCCSpline)	ラスタイメージの YCrCb 値のスプライン補正を行います。

○貼り付け処理 (PasteImage)

貼り付け処理では、マスク(MskHandle)を使用してイメージ(ImgHandle)を選択し、それを別のイメージ(SrcHandle)に貼り付けることができます。



SrcHandle　　DstHandle



※イメージサイズが 384×288 の場合

【Delphi の例(VCL)】

```
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := ImgHandle;
VImageKit1.Effect.MaskImageHandle := MskHandle;
Ret := VImageKit1.Effect.PasteImage(SrcHandle, 0, False, False, 255, False, 0, 0, 0, 255, 255, 255, 192, 144, False);
// 貼り付け処理
if Ret = False then Exit;
// SrcHandle を解放
VImageKit1.FreeMemory(SrcHandle);
SrcHandle := 0;
```

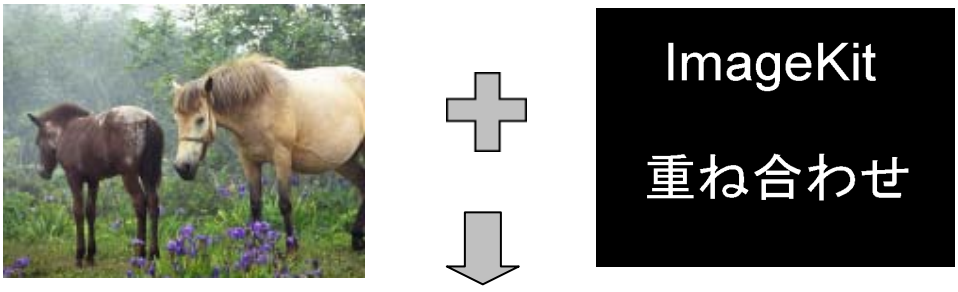
【Delphi の例(DLL)】

```
Src: IKSELECT_IMAGE;
Src.hImgBmh := ImgHandle;
Src.hMskBmh := MskHandle;
DstHandle := IKPasteImage(SrcHandle, Src, 0, FALSE, FALSE, 255, FALSE, 0, 0, 0, 255, 255, 255, 192, 144,
FALSE, 0, '貼り付け', '貼り付け', 'キャンセル'); // 貼り付け処理
IKFreeMemory(SrcHandle);          // 入力ハンドル(イメージ)を解放
IKFreeMemory(Src.hImgBmh);        // ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
IKFreeMemory(Src.hMskBmh);
SrcHandle := 0;
Src.hImgBmh := 0;
Src.hMskBmh := 0;
```

○重ね合わせ処理 (Layer[Image])

重ね合わせ処理では、透過色を指定して二つのイメージを重ね合わせることができます。

ImgHandle1　　ImgHandle2



DstHandle

【Delphi の例(VCL)】

```
VImageKit1.LayerNo := -1;
Ret := VImageKit1.Effect.LayerImage(ImgHandle1, ImgHandle2, 255, True, 0, 0, 0, 255, 255, 255, 0, 0, False);  //重ね合わせ処理
if Ret = False then Exit;
// ImgHandle1, ImgHandle2 を解放
VImageKit1.FreeMemory(ImgHandle1);
ImgHandle1 := 0;
VImageKit1.FreeMemory(ImgHandle2);
ImgHandle2 := 0;
```

【Delphi の例(DLL)】

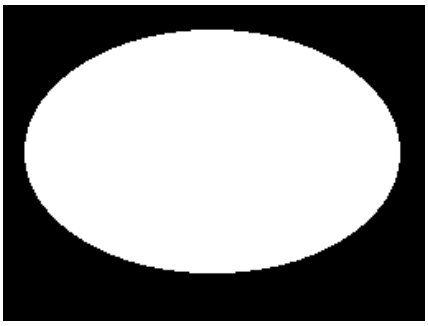
```
DstHandle: THandle;
DstHandle := IKLayer(ImgHandle1, ImgHandle2, 255, True, 0, 0, 0, 255, 255, 255, 0, 0, False, 0, '重ね合わせ処理', '重ね合わせ処理', 'キャンセル');      // 重ね合わせ処理
IKFreeMemory(ImgHandle1);          // 入力ハンドル(イメージ)を解放
IKFreeMemory(ImgHandle2);          // ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
ImgHandle1 := 0;
ImgHandle2 := 0;
```

○拡大縮小処理 (Resize)

リサイズ処理では、イメージとマスクを同時に拡大縮小処理することが可能です。

InImgHandle

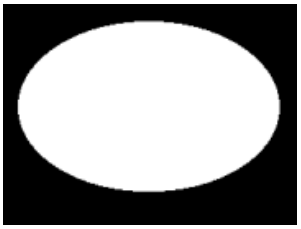
InMskHandle



70%に縮小

OutImgHandle

OutMskHandle



※イメージサイズが 256×192 の場合

【Delphi の例(VCL)】

```
VImageKit1.LayerNo := -1;
VImageKit1.ImageHandle := InImgHandle;
VImageKit1.Effect.MaskImageHandle := InMskHandle;
Ret := VImageKit1.Effect.Resize(179, 134, True);          // リサイズ処理
```

【Delphi の例(DLL)】

```
Src, Dst: IKSELECT_IMAGE;
Src.hImgBmh := InImgHandle;          // 入力ハンドルにイメージとマスクをセット
Src.hMskBmh := InMskHandle;
IKResizeEx(Src, Dst, 179, 134, True, 0, '拡大縮小処理', '拡大縮小処理', 'キャンセル');  // リサイズ処理
IKFreeMemory(InImgHandle);           // 入力ハンドル(イメージ)を解放
IKFreeMemory(InMskHandle);           // ハンドルを解放しないとイメージがいつまでもメモリに常駐します。
InImgHandle := 0;
InMskHandle := 0;
```

※Resize はイメージとマスクのハンドルを受けて、イメージとマスクのハンドルを返すという形で、下記のメソッド・関数と受け渡しの形は同じです。下記のメソッド・関数を使用の際にもこの説明を参考にしてください。

メソッド(関数)名	機能
Rotation(IKRotationEx)	イメージとマスクの回転を行います。

7. 描画機能を持つコントロール

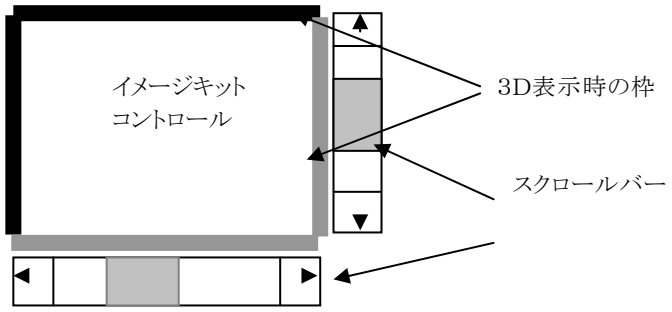
ImageKit10 には 5 つの描画機能を持ったコントロールが実装されています。旧バージョンから継承されているサムネイル、ImageKit7 から実装されたイメージキットコントロール、そして ImageKit9 から実装された Web カメラ用のプレイ/プレビュー/レコードコントロールです。
イメージキットコントロールは ImageKit6 より以前のコモン、ディスプレイ、エフェクト、ファイル、プリント、スキャンコントロールを一つに統合したコントロールです。そのため、ImageKit6 より以前のディスプレイコントロールでサポートしている機能は全てイメージキットコントロールに含まれます。本章ではイメージキットコントロールに含まれる「イメージ編集ツールバー」について簡単に説明していきます。

7－1. イメージキットコントロールの使用法

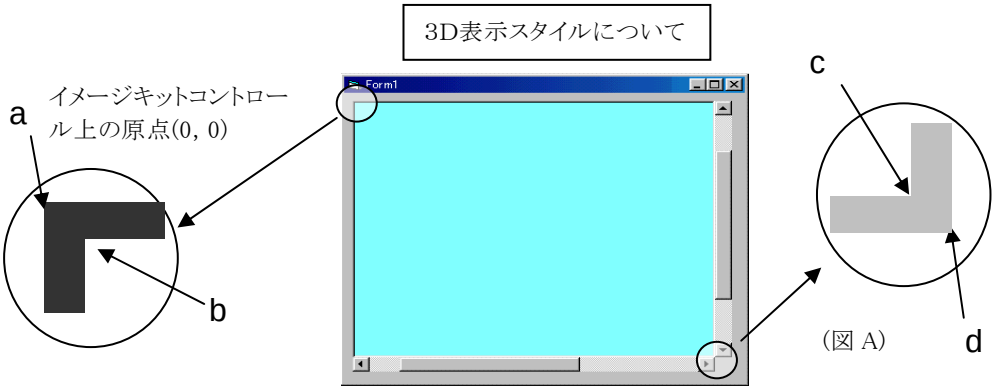
1.イメージキットコントロールの基本構成

イメージキットコントロールは基本的に以下の様な構成になっています。
デザインは Bordervisible,Appearance,Color,ScrollBar プロパティの設定で行います。

Bordervisible	:イメージキットコントロールの枠の描画
Appearance	:イメージキットコントロールの枠のフラット,凹,凸の指定
Color	:イメージキットコントロールの背景色の指定
ScrollBar	:スクロールバーの表示の有無



2.スクリーンの座標について



(図 A)

(図 B) (図 C) (図 D)

e : イメージキットコントロール上のイメージ表示開始座標 OriginX, OriginY(a からの座標値)

(1) BorderVisible:False　　Appearance:(設定値は無効)
　　a --- x = 0, y = 0
　　b --- a と同じ
　　d --- イメージキットコントロールの大きさ
　　c --- d と同じ

(2) BorderVisible:True　　Appearance:フラット
　　a --- x = 0, y = 0
　　b --- x = 1, y = 1

d --- イメージキットコントロールの大きさ
c --- x = d の幅 － 1, y = d の高さ － 1

(3) BorderVisible：True　　Appearance：凹，凸
a --- x = 0, y = 0
b --- x = 2, y = 2
d --- イメージキットコントロールの大きさ
c --- x = d の幅 － 2, y = d の高さ － 2

※(1)から(3)においてストレッチ表示と実寸表示の場合は、Grad を vikNone 以外にすると b の x,y はそれぞれ 20 増えます。スケール表示の場合は、イメージのサイズとコントロールのサイズにより決定されます。詳しくは EndDispImage イベントの OriginX,OriginY で確認してください。

3.イベントの書式

イメージの表示後,スクロール後に発生

EndDispImage(OriginX, OriginY, ALeft, ATop, ARight, ABottom, ScaleWidth, ScaleHeight)
OriginX,OriginY　　：イメージキットコントロール内でのイメージの表示開始座標
ALeft,ATop　　：表示されている実イメージの原点からの位置（左上）
ARight,ABottom　　：表示されている実イメージの原点からの位置（右下）
ScaleWidth, ScaleHeight：表示イメージのスケール

【例】
（図 B）では…実寸表示
BorderVisible：False のとき OriginX = 0, OriginY = 0 (Grad：vikNone 以外のとき OriginX = 20, OriginY = 20)
Appearance：凹又は,凸のとき OriginX = 2, OriginY = 2 (Grad：vikNone 以外のとき OriginX = 22, OriginY = 22)
ALeft, ATop = 表示されている実イメージの原点からの位置（左上）
ARight, ABottom = 表示されている実イメージの原点からの位置（右下）
ScaleWidth, ScaleHeight = DispScaleX,DispScaleY プロパティ値
（図 C）では…スケール表示
BorderVisible：False のとき OriginX = 0, OriginY = 30
Appearance：凹又は,凸のとき OriginX = 2, OriginY = 32(30+2)
ALeft = 0, ATop = 0
ARight = 実イメージの幅-1, ABottom = 実イメージの高さ-1
ScaleWidth = 0.55, ScaleHeight = 0.55
（図 D）では…ストレッチ表示
BorderVisible：False のとき OriginX = 0, OriginY = 0 (Grad：vikNone 以外のとき OriginX = 20, OriginY = 20)
Appearance：凹又は,凸のとき OriginX = 2, OriginY = 2 (Grad：vikNone 以外のとき OriginX = 22, OriginY = 22)
ALeft = 0, ATop = 0
ARight = 実イメージの幅-1, ABottom = 実イメージの高さ-1
ScaleWidth = 0.55, ScaleHeight = 0.69

マウスボタンを押した後に発生

MouseDownImage(Button, Shift, OriginX, OriginY, ALeft, ATop, ARight, ABottom, ScaleWidth, ScaleHeight, X, Y)
Button　　：mbLeft-左ボタン,mbRight-右ボタン,mbMiddle-中ボタン
Shift　　：ssShift-Shift キー,ssCtrl-Ctrl キー,ssAlt-Alt キー など
OriginX,OriginY　　：イメージキットコントロール内でのイメージの表示開始座標
ALeft,ATop　　：表示されている実イメージの原点からの位置（左上）
ARight,ABottom　　：表示されている実イメージの原点からの位置（右下）
ScaleWidth, ScaleHeight：表示イメージのスケール
X, Y　　：マウスボタンを押したイメージ上の座標

マウスボタンを離した後に発生

MouseUpImage(Button, Shift, OriginX, OriginY, ALeft, ATop, ARight, ABottom, ScaleWidth, ScaleHeight, X, Y)
Button　　：mbLeft-左ボタン,mbRight-右ボタン,mbMiddle-中ボタン
Shift　　：ssShift-Shift キー,ssCtrl-Ctrl キー,ssAlt-Alt キー など
OriginX,OriginY　　：イメージキットコントロール内でのイメージの表示開始座標
ALeft,ATop　　：表示されている実イメージの原点からの位置（左上）
ARight,ABottom　　：表示されている実イメージの原点からの位置（右下）
ScaleWidth, ScaleHeight：表示イメージのスケール
X, Y　　：マウスボタンを離したイメージ上の座標

マウスを移動後に発生

MouseMoveImage(Shift, OriginX, OriginY, ALeft, ATop, ARight, ABottom, ScaleWidth, ScaleHeight, X, Y)

Shift	:ssShift-Shift キー,ssCtrl-Ctrl キー,ssAlt-Alt キー など
OriginX,OriginY	:イメージキットコントロール内でのイメージの表示開始座標
ALeft,ATop	:表示されている実イメージの原点からの位置(左上)
ARight,ABottom	:表示されている実イメージの原点からの位置(右下)
ScaleWidth, ScaleHeight	:表示イメージのスケール
X, Y	:マウスを移動させたイメージ上の座標

4.イメージの表示

イメージの表示

イメージの表示は下記のように**イメージハンドルのセット**,**表示**の順で行ってください。

[Delphi]

```
{コントロール}.LayerNo := -1;
{コントロール}.FileIO.FileName := 'c:¥Image¥001.jpg';
{コントロール}.DisplayMode := vikScale; //or vikStrech,vikActualSize
{コントロール}.FileIO.LoadFile(vikLoad); //ImageHandle に設定される
{コントロール}.DisplayImage();
```

[C++Builder]

```
{コントロール}->LayerNo = -1;
{コントロール}->FileIO->FileName = "c:¥¥Image¥¥001.jpg';
{コントロール}->DisplayMode = vikScale; //or vikStrech,vikActualSize
{コントロール}->FileIO->LoadFile(vikLoad); //ImageHandle に設定される
{コントロール}->DisplayImage();
```

指定した位置からのイメージの表示(実寸表示でのみ有効)

(例) イメージの横 10 ピクセル,縦 20 ピクセルからの表示

この時 **DispCenterX**,**DispCenterY** のプロパティは **0** に設定してください。

```
DispStartX := 10;
DispStartY := 20;
DisplayMode := vikActualSize;
DisplayImage();
```

各コンテナでの記述は前記を参照してください。

指定した位置をイメージキットコントロールの中心に表示(実寸表示でのみ有効)

(例) イメージの横 200 ピクセル,縦 100 ピクセルをイメージキットコントロールの中心に表示

この時 **DispStartX**,**DispStartY** のプロパティは **0** に設定してください。

```
DispCenterX := 200;
DispCenterY := 100;
DisplayMode := vikActualSize;
DisplayImage();
```

各コンテナでの記述は前記を参照してください。

イメージのスケール表示(実寸表示でのみ有効)

(例) イメージを 1.5 倍に表示

```
DispScaleX := 1.5;                  // 1.5 倍表示
DispScaleY := 1.5;                  // 1.5 倍表示
DisplayMode := vikActualSize;
DisplayImage();
```

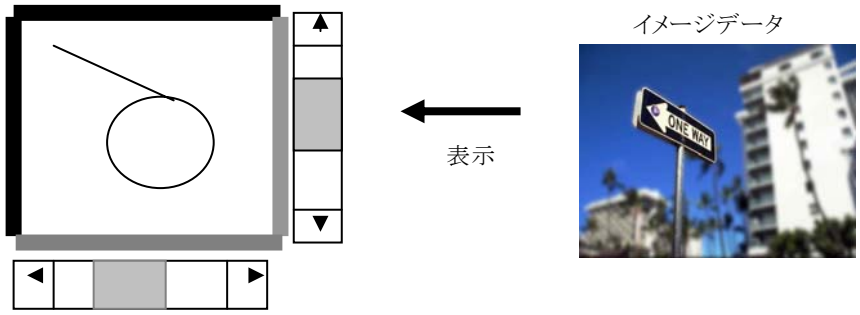
各コンテナでの記述は前記を参照してください。

5.イメージキットコントロールのデバイスコンテキストについて

(1)Canvas.Handle

スクリーンのデバイスコンテキストです。**PrintDraw** プロパティに実装されているメソッド(DLL の場合は関数)もしくは Canvas プロパティのメソッドを使用することにより、スクリーンへ図形を描画する事が出来ます。

ただし、これはスクリーンへの描画であり、イメージ自体への描画ではありません。



【Delphi の例】

```
VImageKit1.LayerNo := -1;
VImageKit1.FileIO.FileName := 'c:¥Image¥001.jpg';
VImageKit1.DisplayMode := vikScale;        //or vikStrech,vikActualSize
VImageKit1.FileIO.LoadFile(vikLoad);       //ファイルからイメージを読み込む
VImageKit1.DisplayImage();
// 直線
VImageKit1.PrintDraw.ClearProperty;
VImageKit1.PrintDraw.PenWidth := 1;
VImageKit1.PrintDraw.PenStyle := vikPenSolid;
Ret := VImageKit1.PrintDraw.Line(VImageKit1.Canvas.Handle, 10, 10, 300, 200, vikScreen);
// 円弧
VImageKit1.PrintDraw.ClearProperty;
VImageKit1.PrintDraw.PenWidth := 2;
VImageKit1.PrintDraw.PenStyle := vikPenSolid;
VImageKit1.PrintDraw.BrushStyle := vikBrushHatchBdiagonal;
VImageKit1.PrintDraw.BrushColor := clRed;
VImageKit1.PrintDraw.Transparent := True;
Ret := VImageKit1.PrintDraw.Ellipse(VImageKit1.Canvas.Handle, 300, 200, 500, 400, vikScreen);
```

各コンテナでの記述は前記を参照してください。

(2)ImageHdc

イメージメモリのデバイスコンテキストです。このイメージメモリのデバイスコンテキストに **PrintDraw** プロパティに実装されているメソッド(DLL の場合は関数)を用いてイメージに直接図形を描画する事が出来ます。

【Delphi の例】

```
VImageKit1.LayerNo := -1;
VImageKit1.FileIO.FileName := 'c:¥Image¥001.jpg';
VImageKit1.DisplayMode := vikScale;        //or vikStrech,vikActualSize
VImageKit1.FileIO.LoadFile(vikLoad);       //ファイルからイメージを読み込む
VImageKit1.DisplayImage();
VImageKit1.Edit.EditEnable := True;        //編集可
// 直線
VImageKit1.PrintDraw.ClearProperty;
VImageKit1.PrintDraw.PenWidth := 1;
VImageKit1.PrintDraw.PenStyle := vikPenSolid;
Ret := VImageKit1.PrintDraw.Line(VImageKit1.ImageHdc, 10, 10, 300, 200, vikScreen);
```

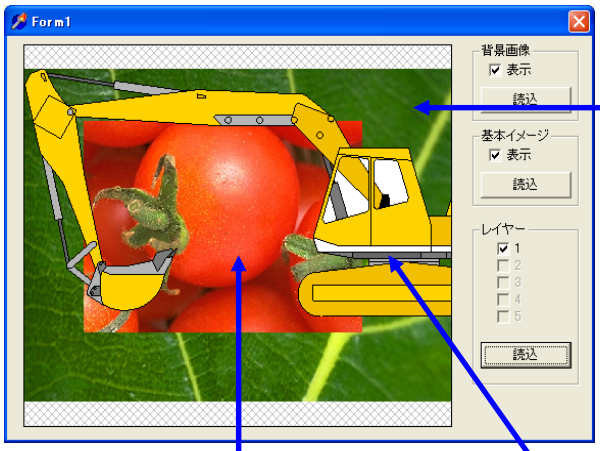
```
// 円弧
VImageKit1.PrintDraw.ClearProperty;
VImageKit1.PrintDraw.PenWidth := 2;
VImageKit1.PrintDraw.PenStyle := vikPenSolid;
VImageKit1.PrintDraw.BrushStyle := vikBrushHatchBdiagonal;
VImageKit1.PrintDraw.BrushColor := clRed;
VImageKit1.PrintDraw.Transparent := True;
Ret := VImageKit1.PrintDraw.Ellipse(VImageKit1.ImageHdc, 300, 200, 500, 400, vikScreen);
//イメージメモリへの描画を有効に(VImageKit1.ImageHdc の内容を VImageKit1.ImageHandle へセット)
VImageKit1.Edit.Modify
```

各コンテナでの記述は前記を参照してください。

6.イメージキットコントロールの階層構造について

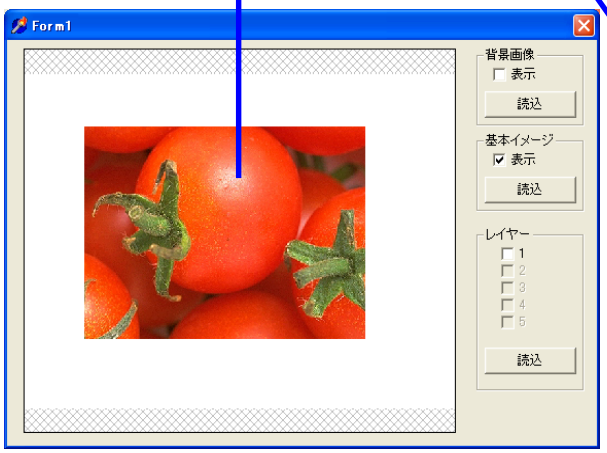
背景イメージ、基本イメージ、階層イメージの順に描画されます。階層イメージは最大で 100 個あり、インデックスの小さいものが下側に配置されます。目盛やグリッドを表示する場合は、対象イメージの上(最上層)に描画されます。注意として階層イメージに設定されているラスタイメージは透過処理を行うため、イメージの数が多くなると処理時間がかかります。ベクトルイメージも同様です。背景イメージは基本イメージの透過を有効にすると表示されます。

背景、基本、階層イメージの表示

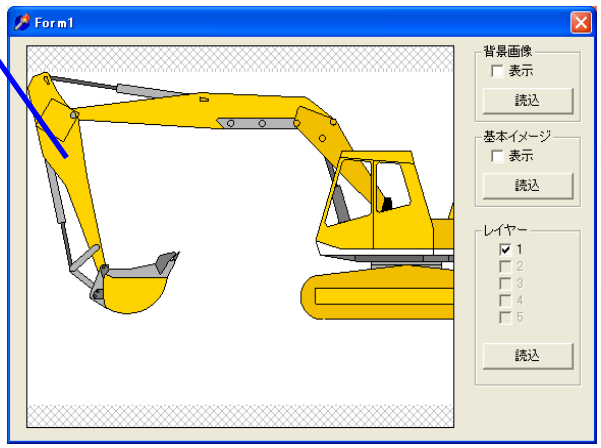


背景イメージ(基本イメージの下部に表示)

基本イメージ



階層イメージ(基本イメージの上部に表示)



7－2. イメージ編集ツールバー(ラスタ)について

■「ズーム」ボタン

イメージの拡大・縮小を行います。

【イメージのズーム倍率】

イメージ編集ツールバーのズームボタンで対応しているズーム倍率は 1/20 倍～20 倍の範囲です。
つまり、最も拡大した状態で 20 倍の大きさ、最も縮小した場合で 1/20 倍の大きさとなります。

【操作方法】

イメージキットコントロールの対象となるイメージを拡大・縮小するにはマウスの左ボタンや右ボタンを使います。拡大表示するには、イメージを左クリックしてください。クリックするたびにズーム倍率が 1 段階上がり、クリックされた位置が中心になります。
縮小表示するには、イメージを右クリックしてください。ズーム倍率が 1 段階下がります。
なお、このズーム機能はイメージキットコントロールもしくはディスプレイコントロールの表示モードが「実寸」以外の場合は、(用途がありませんので)使用できません。

■「矩形範囲選択」ボタン

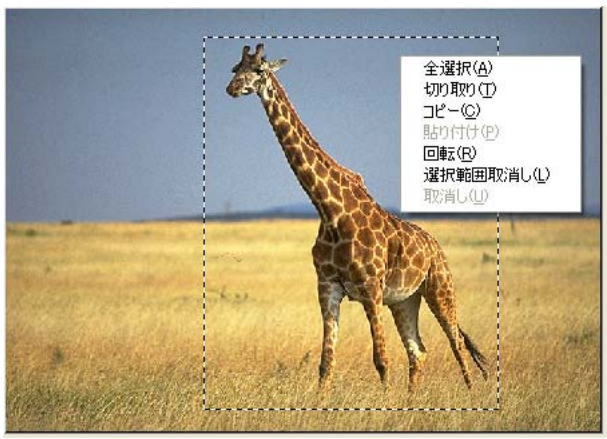
イメージ上に任意の矩形(長方形)範囲を指定します。
通常は、その指定した範囲をコピーして別の位置や他のイメージに貼り付けたり、指定範囲の移動、といった操作に使われます。

【操作方法】

マウスでイメージ上の選択したい矩形(長方形)範囲の開始位置をクリックします。
開始位置は矩形の左右、上下のどのコーナーからでも構いません。
そのままマウスをドラッグして選択範囲を広げます。ドラッグ操作に応じて選択範囲の枠線が変わります。選択範囲を決定する位置でマウスのボタンを離します。選択範囲の枠線がマーキー(動く線)になり、選択範囲が決定されます。
選択範囲決定後に選択範囲内でマウスを左クリックすると範囲を選択する前の状態に戻ります。
決定した範囲選択内をドラッグすると選択範囲の移動ができます。その場合は、移動によって空いた領域は自動的にバックカラー(背景色)で埋められます。

【関連操作】

選択範囲決定後にマウスを右クリックするとメニューリストが表示されます。



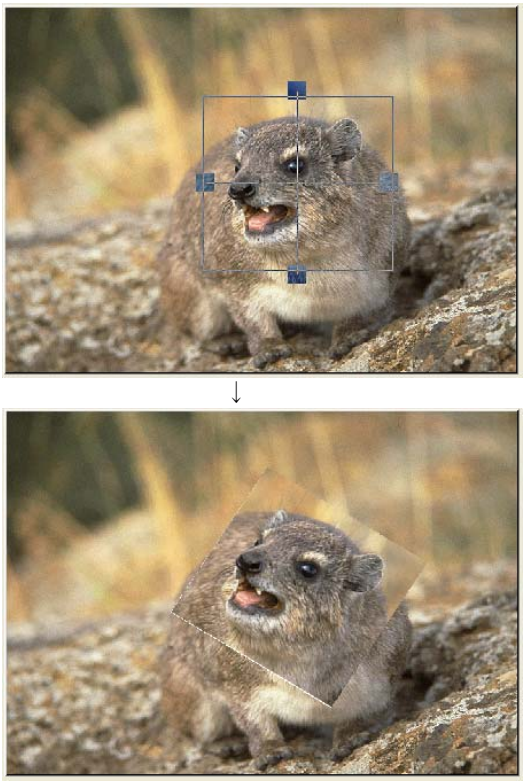
［全選択］　　部分的な領域ではなく、イメージ全体を選択範囲とします。

[切り取り]　選択範囲を切り取り、切り取ったイメージをクリップボードにコピーします。切り取りで、空いた領域は自動的にバックカラー(背景色)で埋められます。

[コピー]　選択範囲のイメージをクリップボードにコピーします。

[貼り付け]　クリップボードにあるイメージを現在のイメージ上に貼り付けます。貼り付けた直後に貼り付け位置をドラッグで決定します。

[回転]　選択範囲のイメージを回転します。
メニューリストから「回転」を選択すると下図のような回転ガイド枠が表示されます。回転ガイド枠の中央を中心として、上下左右のいずれかの■マークを円弧を描くようにドラッグして、回転ガイド枠を回転させます。目的の角度になったらマウスのボタンを離します。



[選択範囲取消し]　選択して決定した範囲を取消し、解除します。

[取消し]　一つ前に行った操作を取消し、元の状態に戻します。

■「自由範囲選択」ボタン

イメージ上に任意の自由線(フリーハンド)で囲まれた領域範囲を指定します。
自由線ですから、マウスでなぞったとおりの領域が指定できます。
通常は、その指定した範囲をコピーして別の位置や他のイメージに貼り付けたり、指定範囲の移動、といった操作に使われます。

【操作方法】

マウスをイメージ上の選択したい領域範囲の開始位置をクリックします。
そのままマウスをドラッグして囲みたい選択範囲の軌跡を描きます。選択範囲を決定する最後の位置でマウスのボタンを離します。選択範囲の枠線がマーキー(動く線)になり、選択範囲が決定されます。なお、マウスのボタンを離す直前の最後の位置と自由線の選択を開始した最初の位置は自動的に直線で結ばれ(閉じられ)ます。
選択範囲決定後に選択範囲内でマウスを左クリックすると範囲を選択する前の状態に戻ります。
決定した範囲選択内をドラッグすると選択範囲の移動ができます。その場合は、移動によって空いた領域は自動的にバックカラー(背景色)で埋められます。

【関連操作】

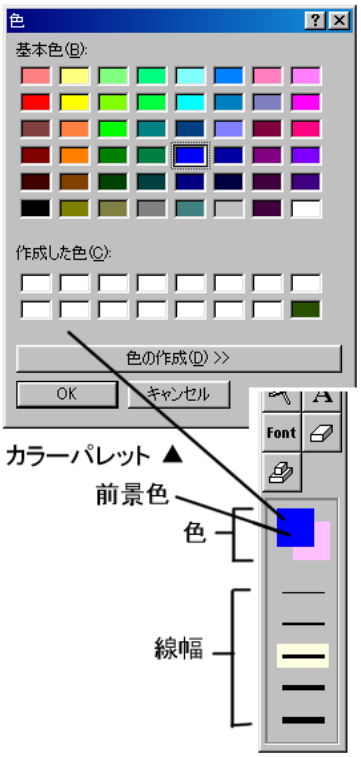
選択範囲決定後にマウスを右クリックするとメニューリストが表示されます。
詳細は、「矩形範囲選択」ボタンの【関連操作】と同様ですのでそちらをご覧ください。

■「ペン」ボタン

イメージ上にペンでフリーハンドの線を描きます。

【操作方法】

1 「ペン」ボタンを選択します。
2 色選択エリアで前景色を選択します。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 線幅選択エリアで線幅を選択します。
4 イメージ上の線を開始する位置をクリックします。
5 そのまま、マウスボタンをドラッグしてフリーハンドの線を描きます。
6 マウスボタンを離すとフリーハンドを終了します。



【ワンポイント】

対象となるイメージキットコントロール上のイメージの表示モードが「実寸」以外の場合は、細い線幅を選択して線を描いた場合、描いた線がつながっていないように表示されることがあります。これは、表示モードが「実寸」以外の場合は、イメージキットコントロールにイメージを表示する時点で、表示モードの「スケール」や「ストレッチ」の指定に合わせてイメージキットコントロールが自動的にイメージの縮小や拡大をしていることに起因します。この場合でも表示モードを「実寸」にすれば、線がつながっていることを確認でき、その状態が本来の正しいイメージということになります。

■「色選択」ボタン

イメージ上の 1 ポイントの色を取得し、パレットの前景色に設定します。

【操作方法】

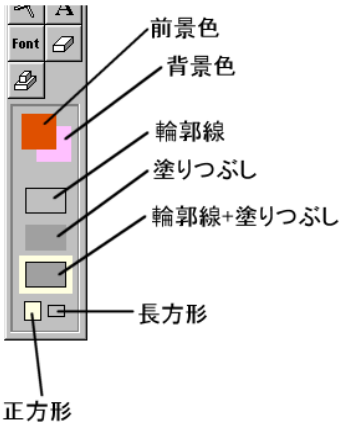
イメージ上の選択したい色の位置をマウスでクリックします。
色選択エリアの前景色が選択した色に変わります。

■「四角形」ボタン

イメージ上に四角形を描きます。

【操作方法】

1 「四角形」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 形状エリアで形状(正方形、長方形)を選択します。
5 マウスでイメージ上の描画したい四角形の開始位置をクリックします。開始位置は四角形の左右、上下のどのコーナーからでも構いません。
6 そのまま、マウスボタンをドラッグして四角形を描きます。
7 マウスボタンを離すと四角形が決定されます。



【ワンポイント】

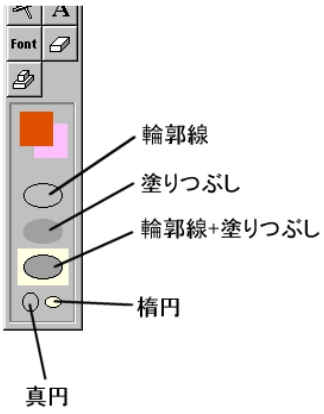
対象となるイメージキットコントロール上のイメージの表示モードを「実寸」にして当機能を利用するようにします。(詳しくは、「ペン」ボタンの【ワンポイント】をご覧ください。)

■「円」ボタン

イメージ上に円を描きます。

【操作方法】

1 「円」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 形状エリアで形状(真円、楕円)を選択します。
5 マウスでイメージ上の描画したい円の中心位置をクリックします。
6 そのまま、マウスボタンをドラッグして円描きます。
7 マウスボタンを離すと円が決定されます。



【ワンポイント】

対象となるイメージキットコントロール上のイメージの表示モードを「実寸」にして当機能を利用するようにします。(詳しくは、「ペン」ボタンの【ワンポイント】をご覧ください。)

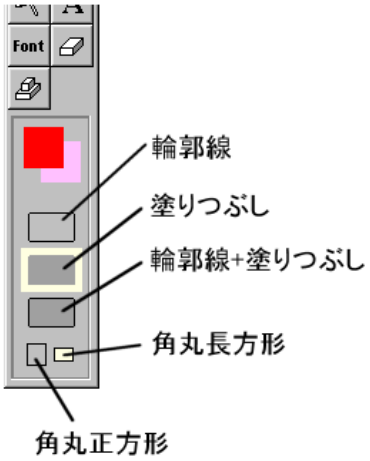
■「角丸矩形」ボタン

イメージ上に角丸矩形(四角形)を描きます。

イメージ編集ツールバー(ラスタ)

【操作方法】

1 「角丸矩形」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 形状エリアで形状(角丸正方形、角丸長方形)を選択します。
5 マウスでイメージ上の描画したい角丸矩形の開始位置をクリックします。開始位置は角丸矩形の左右、上下のどのコーナーからでも構いません。
6 そのまま、マウスボタンをドラッグして角丸矩形を描きます。
7 マウスボタンを離すと角丸矩形が決定されます。



【ワンポイント】

対象となるイメージキットコントロール上のイメージの表示モードを「実寸」にして当機能を利用するようにします。(詳しくは、「ペン」ボタンの【ワンポイント】をご覧ください。)

■「直線」ボタン

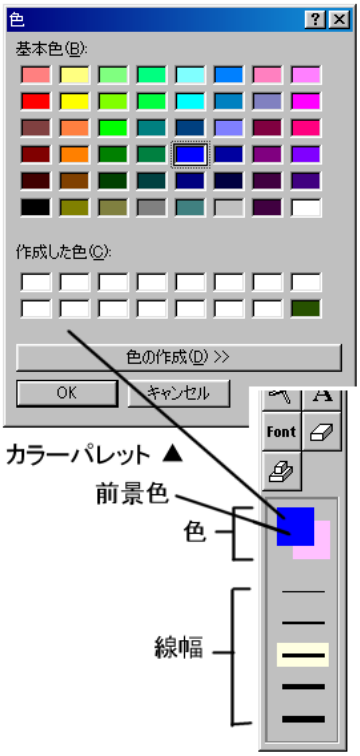
イメージ上に直線を描きます。

【操作方法】

1 「直線」ボタンを選択します。
2 色選択エリアで前景色を選択します。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 線幅選択エリアで線幅を選択します。
4 イメージ上の直線を開始する位置をクリックします。
5 そのまま、マウスボタンをドラッグして直線を描きます。
6 マウスボタンを離すと終了します。

【ワンポイント】

対象となるイメージキットコントロール上のイメージの表示モードを「実寸」にして当機能を利用するようにします。(詳しくは、「ペン」ボタンの【ワンポイント】をご覧ください。)

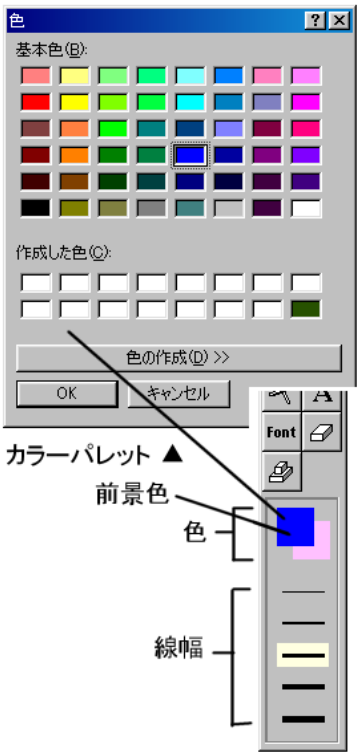


■「曲線」ボタン

イメージ上にベジェ曲線を描きます。

【操作方法】

1 「曲線」ボタンを選択します。
2 色選択エリアで前景色を選択します。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 線幅選択エリアで線幅を選択します。
4 イメージ上のベジェ曲線を開始する位置をクリックします。
5 そのまま、マウスボタンをドラッグして直線を描き、ベジェ曲線の終了する位置でマウスボタンを離します。
6 直線から離れた任意の位置でマウスのクリック・ドラッグを 2 回繰り返してベジェ曲線を作成します。1 回目のクリック・ドラッグで開始位置からの角度と目的位置を設定し、2 回目のクリック・ドラッグで終了位置からの角度と目的位置を設定します。
7 2 回目のクリック・ドラッグでマウスのボタンを離すと、曲線の形状を決定します。



【ワンポイント】

対象となるイメージキットコントロール上のイメージの表示モードを「実寸」にして当機能を利用するようにします。(詳しくは、「ペン」ボタンの【ワンポイント】をご覧ください。)

■「塗りつぶし」ボタン

領域や図形を前景色で塗りつぶします。

【操作方法】

1 「塗りつぶし」ボタンを選択します。
2 色選択エリアで前景色を決めます。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 塗りつぶす領域や図形の上をマウスでクリックします。

【ワンポイント】

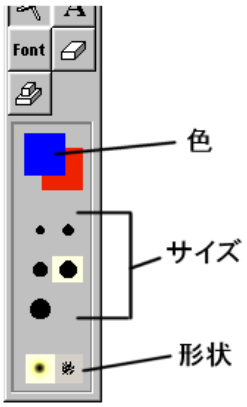
この塗りつぶしを実行した際に、塗りつぶす領域や図形の輪郭線が閉じていないと、ほかの領域まで色が広がってしまい意図しない領域も塗られてしまいます。輪郭線の切れ目にご注意ください。

■「エアーブラシ」ボタン

エアーブラシを使ってスプレーを吹き付けます。

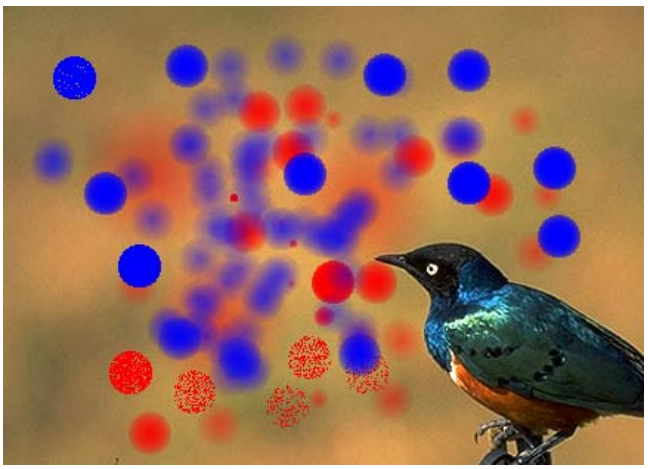
【操作方法】

1 「エアーブラシ」ボタンを選択します。
2 色選択エリアで前景色を決めます。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 サイズ選択エリアでスプレーのサイズを選択します。
4 形状選択エリアでスプレーの形状を選択します。
5 スプレーする位置をマウスでクリックします。そのままドラッグをすると連続してスプレーすることができます。



【ワンポイント】

このエアーブラシを利用する場合は、イメージ上の同一位置でマウスボタンを押している時間でスプレーの濃度が変わります。長い時間押しているほど濃いスプレーを吹き付けることになります。ぼかす感覚でスプレーする場合は、同一位置では短い時間しかボタンを押さないようにします。



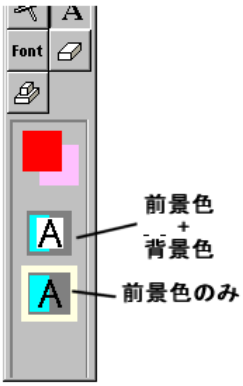
エアーブラシ
を使った
スプレーの例

■「テキスト」ボタン、「フォント」ボタン

テキストボックスを使って文字列を入力します。

【操作方法】

1 「テキスト」ボタンを選択します。
2 表示されるテキストボックスにイメージ上に表示したい文字列を入力します。改行する場合は、「Ctrl」+「Enter」キーを押します。
3 文字列の入力が済んだら、カーソルがテキストボックス内にある状態で「Enter」キーを押します。
4 テキストボックスの枠線がマーキー(動く線)になり背景のイメージが透過され見えるようになります。この段階で、次のことができます。
   ●枠線マーキー内をドラッグしてテキストボックスを移動できます。このとき、マーキー外をクリックしてしまうと文字列が確定してしまいますので注意してください。
   ●イメージ編集ツールバーの前景色/背景色指定エリアで指定した色を文字列の背景色とすることもできます。
   ●枠線マーキー内をダブルクリックすると、再度文字列の編集ができます。
   ●イメージ編集ツールバーの色選択エリアで前景色を変更すると枠線マーキー内の文字列の色も変わります。
   ●イメージ編集ツールバーの「フォント」ボタンで「フォントの指定」ダイアログを表示し、フォント名やスタイル、サイズ、文字飾りなどを変更できます。
5 文字列を確定したい場合は枠線マーキー外をクリックするか、イメージ編集ツールバーのほかのボタンを押します。

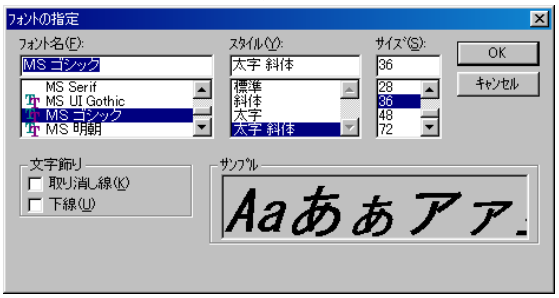


【ワンポイント】

上記の 4 で記載されている機能は必ずしも、テキストボックスの枠線がマーキー(動く線)になったタイミングだけでできるとは限りません。たとえば、先に文字列の色やフォントを決めておいてから「テキスト」ボタンを押して文字

列を入力してもかまいません。それらの順序については、必要に応じてお試しください。

▼フォント指定ダイアログ

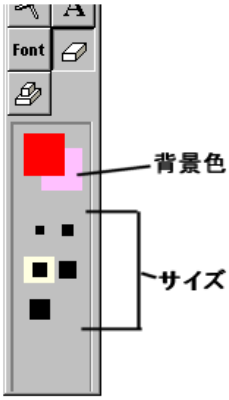


■「消しゴム」ボタン

イメージ上を現在の背景色で塗りつぶします。

【操作方法】

1 「消しゴム」ボタンを選択します。
2 色選択エリアで背景色を選択します。背景色エリアをクリックするとカラーパレットの選択が表示されます。
3 サイズ選択エリアで消しゴムのサイズを選択します。
4 消しゴムで消したい部分をドラッグします。
5 マウスボタンを離すと終了します。



■「スタンプ」ボタン

イメージ上に特定のイメージデータをスタンプとして押します。
スタンプへのイメージデータのセットは、

VImageKit1.Edit.StampBmpFile := CurrentDir + 'Stamp.bmp';

のようにプログラムで定義します。

【操作方法】

1 「スタンプ」ボタンを選択します。
2 スタンプを配置したい位置でマウスをクリックします。
3 マウスボタンを離すと終了します。

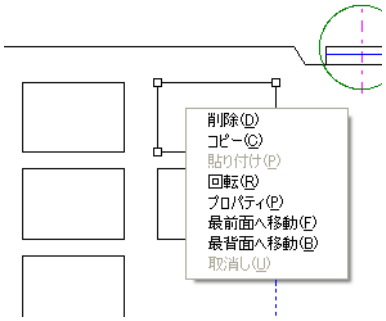
7－3. イメージ編集ツールバー(ベクトル)について

■「カーソル」ボタン

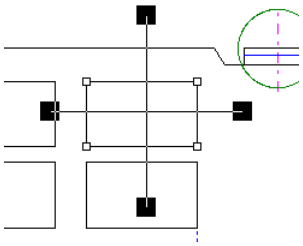
ベクトルデータの要素を選択したり、「ドラッグスクロール」が有効な場合のマウスによるイメージのドラッグを行います。

【操作方法】
[ベクトルデータの要素の選択]

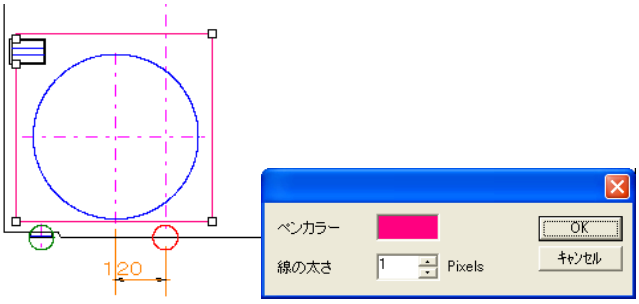
1 「カーソル」ボタンを選択します。
2 選択したいベクトルデータの要素をクリックします。該当する要素が小さな正方形で強調表示されます。複数の要素を連続して選択する場合は、[Shift]キーを押しながら対象をクリックします。



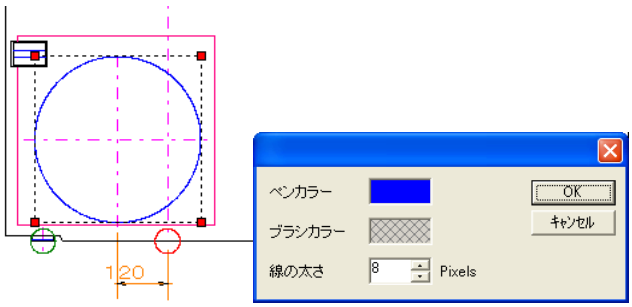
3 要素を選択した状態で右クリックすると、メニューリストが表示されます。ここで、必要な処理を選択します。
4 要素を削除する場合は「削除」、回転させる場合は「回転」を選択します。
▼回転時の回転ガイド枠



5 メニューリストから「プロパティ」を選択するとその要素のプロパティを確認・変更できます。
▼直線や四角形の場合のプロパティ(ペンカラー、線の太さ)

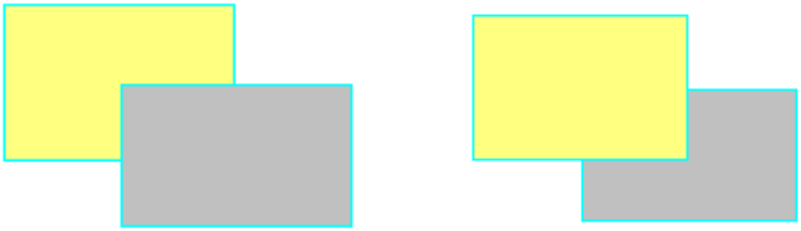


▼円の場合のプロパティ(ペンカラー、ブラシカラー、線の太さ)



6 メニューリストから「最前面へ移動」や「最背面へ移動」を選択することによって、その要素を他の要素に対して前面や背面に指定することができます。背面になった要素は前面の要素と重なった部分は隠れて見えなくなります。

▼右下図形が前面になった場合　　▼左上図形が前面になった場合



【ワンポイント】
メニューリストから「プロパティ」を選択して線の太さを 0 にすると、その要素を拡大縮小しても線の太さは変わりません。

■「ズーム」ボタン

イメージの拡大・縮小を行います。

【イメージのズーム倍率】
イメージ編集ツールバーのズームボタンで対応しているズーム倍率は 1/20 倍～20 倍の範囲です。
つまり、最も拡大した状態で 20 倍の大きさ、最も縮小した場合で 1/20 倍の大きさとなります。

【操作方法】
イメージキットコントロールの対象となるイメージを拡大・縮小するにはマウスの左ボタンや右ボタンを使います。拡大表示するには、イメージを左クリックしてください。クリックするたびにズーム倍率が 1 段階上がり、クリックされた位置が中心になります。縮小表示するには、イメージを右クリックしてください。ズーム倍率が 1 段階下がります。
なお、このズーム機能はイメージキットコントロールの表示モードが「実寸」以外の場合は、（用途がありませんので）使用できません。

■「範囲選択」ボタン

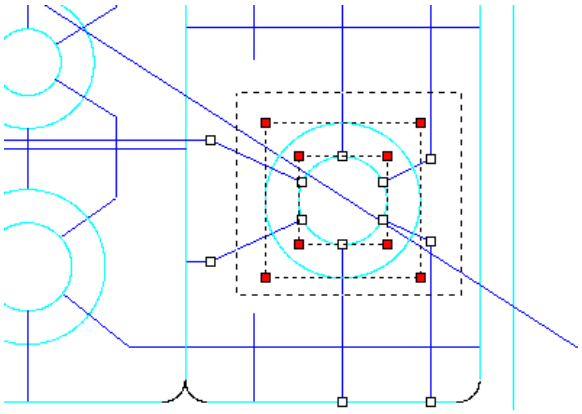
イメージ上に任意の矩形（長方形）範囲を指定します。
通常は、その指定した範囲をコピーして別の位置や他のイメージに貼り付けたり、指定範囲の移動、といった操作に使われます。

【操作方法】
マウスでイメージ上の選択したい矩形（長方形）範囲の開始位置をクリックします。
開始位置は矩形の左右、上下のどのコーナーからでも構いません。
そのままマウスをドラッグして選択範囲を広げます。ドラッグ操作に応じて選択範囲の枠線が変わります。選択範囲を決定

する位置でマウスのボタンを離します。選択範囲の枠線がマーキー(動く線)になり、選択範囲が決定されます。
選択範囲決定後に選択範囲外でマウスを左クリックすると範囲の選択を解除します。

【ワンポイント】

選択範囲を指定すると、マーキー(動く線)の枠線のほかに選択範囲に含まれるベクトルデータの要素(線分や円など)も同時に強調表示されます。円の場合は、円が収まる四角形の枠が点線で表示され、その 4 角が赤い小さな正方形で表示されます。線分の場合は、その両端に白い小さな正方形が表示されます。また、選択した範囲内にその要素の一部が入る場合はその要素全体が強調表示され、選択範囲とみなされます。たとえば選択した範囲内に直線の片端しかなかったとしても、その直線の両端に白い小さな正方形が表示され、その選択範囲をドラッグして移動すればその直線全体の移動となります。



■「四角形」ボタン

イメージ上に四角形を描きます。ただし、DXF 形式のイメージの場合、塗りつぶしはできません。

【操作方法】

1 「四角形」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 形状エリアで形状(正方形、長方形)を選択します。
5 マウスで描画したい四角形の開始位置をクリックします。開始位置は四角形の左右、上下のどのコーナーからでも構いません。
6 そのまま、マウスボタンをドラッグして四角形を描きます。
7 マウスボタンを離すと四角形が決定されます。

■「円」ボタン

イメージ上に円を描きます。ただし、DXF 形式のイメージの場合、塗りつぶしはできません。

【操作方法】

1 「円」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 形状エリアで形状(真円、楕円)を選択します。
5 マウスで描画したい円の中心位置をクリックします。
6 そのまま、マウスボタンをドラッグして円描きます。
7 マウスボタンを離すと円が決定されます。

■「角丸矩形」ボタン

イメージ上に角丸矩形(四角形)を描きます。ただし、DXF 形式のイメージの場合、使用できません。

【操作方法】

1 「角丸矩形」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 形状エリアで形状(角丸正方形、角丸長方形)を選択します。

5 マウスで描画したい角丸矩形の開始位置をクリックします。開始位置は角丸矩形の左右、上下のどのコーナーからでも構いません。
6 そのまま、マウスボタンをドラッグして角丸矩形を描きます。
7 マウスボタンを離すと角丸矩形が決定されます。

■「直線」ボタン

イメージ上に直線を描きます。

【操作方法】
1 「直線」ボタンを選択します。
2 色選択エリアで前景色を選択します。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 線幅選択エリアで線幅を選択します。
4 直線を開始する位置をクリックします。
5 そのまま、マウスボタンをドラッグして直線を描きます。
6 マウスボタンを離すと終了します。

■「連続線」ボタン

連続した直線を描きます。

【操作方法】
1 「連続線」ボタンを選択します。
2 色選択エリアで前景色を選択します。前景色エリアをクリックするとカラーパレットの選択が表示されます。
3 線幅選択エリアで線幅を選択します。
4 連続した直線の描画を開始する位置をクリックします。
5 そのまま、マウスボタンをドラッグして直線を描きます。
6 マウスボタンを離して直線の終点(次の直線の始点)を指定します。
7 上記の 5、6 を繰り返します。

■「多角形」ボタン

連続した直線で多角形を描きます。ただし、DXF 形式のイメージの場合、塗りつぶしはできません。

【操作方法】
1 「多角形」ボタンを選択します。
2 色選択エリアで前景色(塗りつぶしの場合は、背景色も)を決めます。
3 スタイルエリアでスタイル(輪郭線、塗りつぶし、輪郭線+塗りつぶし)を選択します。
4 多角形を開始する位置をクリックします。
5 そのまま、マウスボタンをドラッグして直線を描きます。
6 マウスボタンを離して直線の終点(次の直線の始点)を指定します。
7 上記の 5、6 を繰り返します。
8 最後にマウスをドラッグせずにポインタだけを移動させると自動的に最初に描画した直線の始点と結んで閉じた多角形を描画してくれます。

■「イメージ」ボタン

ラスタイメージを読み込みます。読み込み時の操作方法は二通りあります。
ただし、DXF 形式のイメージの場合、ラスタイメージを読み込むことはできません。

●[指定した大きさで読み込む場合]
【操作方法】
1 「イメージ」ボタンを選択します。
2 読み込みたいイメージの大きさをマウスでドラッグして四角形の枠で指定します。四角形の左右、上下のどのコーナーからでも構いません。
3 そのまま、マウスボタンをドラッグして四角形を描きます。
4 マウスボタンを離すと「ファイルを開くダイアログ」が表示され、読み込みたいイメージファイルを選択します。
5 指定した四角形の枠に収まるようにイメージが読み込まれ、表示されます。

●[そのイメージファイルのオリジナルな大きさで読み込む場合]
【操作方法】
1 「イメージ」ボタンを選択します。
2 読み込んだイメージを表示する位置をマウスのクリックで指定します。この位置が読み込んだイメージの左上になります。
3 マウスボタンを離すと「ファイルを開くダイアログ」が表示され、読み込みたいイメージファイルを選択します。
4 選択したイメージがオリジナルな大きさで表示されます。

■「テキスト」ボタン、「フォント」ボタン

テキストボックスを使って文字列を入力します。

【操作方法】
1 「テキスト」ボタンを選択します。
2 表示されるテキストボックスにイメージ上に表示したい文字列を入力します。
3 文字列の入力が済んだら、カーソルがテキストボックス内にある状態で「Enter」キーを押します。
4 テキストボックスの枠線がマーキー(動く線)になり背景のイメージが透過され見えるようになります。この段階で、次のことができます。
●枠線マーキー内をドラッグしてテキストボックスを移動できます。このとき、マーキー外をクリックしてしまうと文字列が確定してしまいますので注意してください。
●イメージ編集ツールバーの前景色/背景色指定エリアで指定した色を文字列の背景色とすることもできます。
●枠線マーキー内をダブルクリックすると、再度文字列の編集ができます。
●イメージ編集ツールバーの色選択エリアで前景色を変更すると枠線マーキー内の文字列の色も変わります。
●イメージ編集ツールバーの「フォント」ボタンで「フォントの指定」ダイアログを表示し、フォント名やスタイル、サイズ、文字飾りなどを変更できます。
5 文字列を確定したい場合は枠線マーキー外をクリックするか、イメージ編集ツールバーのほかのボタンを押します。

【ワンポイント】
上記の 4 で記載されている機能は必ずしも、テキストボックスの枠線がマーキー(動く線)になったタイミングだけでできるとは限りません。たとえば、先に文字列の色やフォントを決めておいてから「テキスト」ボタンを押して文字列を入力してもかまいません。それらの順序については、必要に応じてお試しください。

イメージの色について

Windows 上でのイメージの色の取り扱いについて考慮しておくべき点を簡単に説明します。

1)パレット化する 16・256 色とパレット化しないフルカラー

画面に同時に表示できる色数は、動作環境の設定によって異なります。

16 色を同時に表示する環境では、表示可能な数千色の中からそれぞれのイメージに必要な 16 色分を自動的にパレットにセット(パレット化)して表示します。
256 色の場合もパレットが 256 色分になる点を除けば、16 色の場合と同様と考えられます。

それに対して、フルカラー(1670 万色)の場合は、16・256 色の場合と異なりパレット化せずにイメージそのままの色に極めて近い色を同時に表示できます。
それは、フルカラーのイメージでは、ピクセルごとのカラー情報をパレットを使用せずに直接データに保持していることで実現されています。
反面、16・256 色に比べ使用するメモリやファイルの容量が大きく、イメージの操作(表示やファイル化など)に時間がかかるなどのデメリットがあり、ハードウェアに依存する割合が大きくなります。

2)複数のイメージを同時に表示する場合の現象

Windows の仕様として、パレットの異なる複数のイメージを同時に表示している場合、現在フォーカスを持つイメージのパレットを使用して、同時に表示されている他のイメージを表示し直します。
それにより、フォーカスのない他のイメージが予期しない色で表示される現象が生じます。
この現象は、16・256 色などの環境で生じ、フルカラー環境下では特別問題はありません。

3)固定パレットによる減色

複数のフルカラー(1670 万色)イメージを、16 または 256 色環境下で表示する時に、16 または 256 色の固定パレット減色処理を施すと、前記の「2」複数のイメージを同時に表示する場合の現象」は、解決します。

4)16・256 色環境でフルカラーイメージを表示する場合の現象

16・256 色環境でフルカラーイメージを表示する場合、Windows はフルカラーイメージを自動的にパレット化して表示します。
このため、表示色によっては、元のフルカラーイメージとかなり異なった色で表示される場合があります。
逆に、フルカラー環境で 16・256 色のイメージを表示する場合は問題ありません。

SXF 形式使用時の注意事項

本製品では財団法人日本建設情報総合センターの CAD データ交換標準ソフトウェアを利用しております。
そのため、お客様がイメージファイルの SXF 形式を利用できるようなアプリケーションを開発・販売・運用される場合は、財団法人日本建築情報総合センターとのライセンス契約（無料）がお客様サイドで必要になります。
詳しくは
　http://www.cals.jacic.or.jp/cad/index.html
をご覧ください。

索引

.

.NET Framewrok 4.0 6

C

C++Builder 10 Seattle 12
C++Builder 10.1 Berlin 12
C++Builder 10.2 Tokyo 12
C++Builder 10.3 Rio 13
C++Builder XE3 10
C++Builder XE4 10
C++Builder XE5 10
C++Builder XE6 11
C++Builder XE7 11
C++Builder XE8 11

D

DclIk10D17.bpl 7, 10
DclIk10D18.bpl 7, 10
DclIk10D19.bpl 7, 11
DclIk10D20.bpl 8, 11
DclIk10D21.bpl 8, 11
DclIk10D22.bpl 8, 12
DclIk10D23.bpl 9, 12
DclIk10D24.bpl 9, 12
DclIk10D25.bpl 9, 13
DclIk10D26.bpl 10, 13
DclIk10ThumbD17.bpl 7, 10
DclIk10ThumbD18.bpl 7, 10
DclIk10ThumbD19.bpl 7, 11
DclIk10ThumbD20.bpl 8, 11
DclIk10ThumbD21.bpl 8, 11
DclIk10ThumbD22.bpl 8, 12
DclIk10ThumbD23.bpl 9, 12
DclIk10ThumbD24.bpl 9, 12
DclIk10ThumbD25.bpl 9, 13
DclIk10ThumbD26.bpl 10, 13
DclIk10WebCamD17.bpl 7
DclIk10WebCamD18.bpl 7
DclIk10WebCamD19.bpl 7
DclIk10WebCamD20.bpl 8
DclIk10WebCamD21.bpl 8
DclIk10WebCamD22.bpl 8
DclIk10WebCamD23.bpl 9
DclIk10WebCamD24.bpl 9
DclIk10WebCamD25.bpl 9
DclIk10WebCamD26.bpl 10
Delphi 10 Seattle 8
Delphi 10.1 Berlin 9
Delphi 10.2 Tokyo 9
Delphi 10.3 Rio 9
Delphi XE3 6
Delphi XE4 7
Delphi XE5 7
Delphi XE6 7
Delphi XE7 8
Delphi XE8 8
DLL を利用する場合 6

M

Microsoft Visual C++ 2019 再頒布可能パッケージ 6

S

SXF 形式使用時の注意事項 48

T

Twain_32.dll 15
TWAINDSM.dll 15
TWAIN ドライバのインストール 15

あ

アンインストール 14

い

イメージキットコントロールの使用法 29
イメージとメモリハンドルの操作 16
イメージの色 47
イメージ編集ツールバー(ベクトル)について 42
イメージ編集ツールバー(ラスタ)について 34
インストール 6
インストール後のフォルダの構成 6

え

エフェクト処理の使用例 19

か

開発環境への組み込み 6

き

旧バージョンの削除 6

こ

コンポーネントを利用する場合 6

さ

サンプルプログラム 15

た

対応イメージファイル形式 3
対応開発コンテナ 3
対応スキャンデバイス 3

対応 DirectX.....3
対応プラットフォーム.....3

は

パレット化.....47

ひ

描画機能を持つコントロール.....28
ビルドバージョン.....13

ふ

複数のイメージを同時に表示.....47

め

メモリの参照とコピー.....17
メモリハンドル.....16

り

リソースの減少が発生したら.....15

The ImageKit incorporates the following software:

●JPEG capability
This software is based in part on the work of the Independent JPEG Group.

●TIFF capability
This software is based on the "libtiff" which has the following copyrights:

●JasPer License Version 2.0

Permission is hereby granted, free of charge, to any person (the "User") obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

1. The above copyright notices and this permission notice (which includes the disclaimer below) shall be included in all copies or substantial portions of the Software.

2. The name of a copyright holder shall not be used to endorse or promote products derived from the Software without specific prior written permission.

THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF THE SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER. THE SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. NO ASSURANCES ARE PROVIDED BY THE COPYRIGHT HOLDERS THAT THE SOFTWARE DOES NOT INFRINGE THE PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS OF ANY OTHER ENTITY. EACH COPYRIGHT HOLDER DISCLAIMS ANY LIABILITY TO THE USER FOR CLAIMS BROUGHT BY ANY OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR OTHERWISE. AS A CONDITION TO EXERCISING THE RIGHTS GRANTED HEREUNDER, EACH USER HEREBY ASSUMES SOLE RESPONSIBILITY TO SECURE ANY OTHER INTELLECTUAL PROPERTY RIGHTS NEEDED, IF ANY. THE SOFTWARE IS NOT FAULT-TOLERANT AND IS NOT INTENDED FOR USE IN MISSION-CRITICAL SYSTEMS, SUCH AS THOSE USED IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION OR COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL SYSTEMS, DIRECT LIFE SUPPORT MACHINES, OR WEAPONS SYSTEMS, IN WHICH THE FAILURE OF THE SOFTWARE OR SYSTEM COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE ("HIGH RISK ACTIVITIES"). THE COPYRIGHT HOLDERS SPECIFICALLY DISCLAIM ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR HIGH RISK ACTIVITIES.

●SXF 共通ライブラリ

（財）日本建設情報総合センターの CAD データ交換標準ソフトウェアを利用しています。

ImageKit10 VCL

プログラミングガイド

発行 株式会社ニュートン

〒940-0076
新潟県長岡市本町 2-2-15 シャングリラ本町 1F
TEL 0258-86-6954
FAX 0258-86-6964
http://www.newtone.co.jp/