

TCAD xp.b

for Borland Delphi & C++ Builder & Kylix & Vcl.net



User Manual

1998-2005 HongDi science & technology development co.,ltd. of Huzhou,ZheJiang,China

TCAD for Delphi & C++Builder & Kylix &

Vcl.net

1998-2005 HongDi science & technology development co.,ltd.
of Huzhou,ZheJiang,China

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed in China

Publisher

Crystal Component

Technical Editors

hongbin.fei

jun.shi

Cover Designer

hongbin.fei

Team Coordinator

hongbin.fei

Production

hongbin.fei

Special thanks to:

All the people who contributed to this document, to mum and dad , my mothers in law, to our secretary Rain, to the graphic artist who created this great product logo on the cover page ,to the copy shop where this document will be duplicated, and and and...

Table of Contents

Foreword	1
Part I License	3
Part II Introduction	7
1 What is TCAD	7
2 Application ScreenShot	9
Part III Defines	14
Part IV TGridObject	16
1 GridColor	16
2 GridHeight	17
3 GridPenSize	17
4 GridShow	17
5 GridType	18
6 GridWidth	18
Part V TMyCAD	20
1 ClassDiagram	20
Part1 Properties	20
Part2 Events	21
Part3 Methods	22
2 Events	23
OnActionToolToSelecting	23
OnClick	23
OnDblClick	23
OnDeleteLayer	23
OnDragDrop	24
OnDragOver	24
OnMouseDown	25
OnMouseEnter	25
OnMouseEnterShape	25
OnMouseLeave	26
OnMouseLeaveShape	26
OnMouseMove	26
OnMouseUp	26
OnNewLayer	27
OnPaint	27
OnShapeAdded	27
OnShapeCodeDragging	27
OnShapeCodeRotating	28
OnShapeDeleted	28
OnShapeMouseDragged	29
OnShapeMouseDragging	29
OnShapeMouseResized	29
OnShapeMouseResizing	30
OnShapeMouseRotated	30
OnShapeMouseRotating	30

OnShapeSelected	31
3 Methods	31
AddBlockfromTCADFile	31
AddImageShapeByCode	32
AddShapeByCode	32
AddUserDefineShapefromLib	33
AlignBottom	34
AlignHorizontalCenter	34
AlignLeft	34
AlignRight	35
AlignTop	35
AlignVerticalCenter	35
BringToFront	35
BringToFrontByStep	36
ClearAllUndoStuff	36
ClosePolygon	36
Copy	36
CopyToClipboardAsWmf	37
Create	38
CreateLink	38
Cut	39
DeleteAllLayers	39
DeleteAllShapes	39
DeleteLayerByID	40
DeleteLayerByName	40
DeleteSelectedShape	40
DeleteShapeByID	41
DeSelectedAllShapesByCode	41
Destroy	41
DrawAllShape	42
FlipHoriz	42
FlipVert	42
GetLayerIdByName	43
GetLayerIdByNo	43
GetLayerNameByID	43
GetLayerNoById	44
GetLayerNoByName	44
GetLayersCount	45
GetMaxLayerId	45
GetMemShapesCount	46
GetSelectedShape	46
GetSelectedShapes	47
GetSelectedShapesCount	47
GetShapebyID	47
GetShapeByName	48
GetShapebyNo	48
GetShapeNoById	49
GetShapesCount	49
GetShapesCountInALayer	49
GroupWorkingShape	50
InVisibleLayerById	50
InVisibleLayerByName	50
IsLinked	50
IsTCADFile	51
IsVisibleLayerById	51
LoadFromFile	51
LoadFromStream	52
LockUnLockforShapes	52

MergeLayers	53
Move	53
NewLayer	53
Paste	54
PopfromUndoRedoShapeList	54
Print	54
PrintPreview	56
RenameShapename	57
Rotate	57
SaveToBmp	57
SaveToBmp-2	58
SaveToDxf	58
SaveToFile	58
SaveToJpg	59
SaveToStream	59
SaveToWmf	60
SelectAllShapes	60
SelectAllShapesByLayerId	61
SelectShapeByCode	61
SendtoBack	62
SendToBackByStep	62
SetLayerNameById	62
SetLayerNameByName	63
SetMyImage	63
SizeShape	64
UnGroupShape	64
VisibleAllLayer	65
VisibleLayerById	65
VisibleLayerByName	65
4 Properties	66
ArrowAngle	66
ArrowLength	66
ArrowOffset	66
ArrowStyle	67
BkBitmap	67
BkBitmapMode	68
Brush	68
Canvas	68
ColorOfBackground	69
ColorOfHot	69
CrossLine	70
CurrentLayerId	70
DiskFileVersion	70
DragMode	71
Enable	71
Font	71
GridOperation	71
HotShow	72
HotSize	72
LabelValue	72
LabelXY	73
LinkLineDrawStyle	74
OperateAllLayer	75
PageFoot	75
PageFootAlignment	75
PageFootFont	76
PageFootToBottom	76
PageHead	76

PageHeadAlignment	77
PageHeadFont	77
PageHeadToTop	77
PageHeight	78
PageOrientation	78
PageStyle	78
PageWidth	79
Pen	79
PrintABorder	79
PrintABordertoBottom	79
PrintABordertoLeft	80
PrintABordertoRight	80
PrintABordertoTop	80
PrintBackground	81
Ratio	81
ResizeEnable	81
ReturnToSelecting	82
RotateConstraintDegree	82
RotateEnable	82
ShapeTool	83
ShowHint	83
ShowHotLink	84
Snap	84
SnapPixels	84
SnapShape	85
TheUnit	85
UndoRedoSize	85
Version	86
Visible	86
XYMode	86
Zoom	86

Part VI Shape Class Inherited Diagram 88

Part VII TMyShape 91

1 ClassDiagram	91
2 Fields	92
CenterPoint	92
ChildShapesNo	92
LayerID	92
ParentShapesNo	92
Shapeld	92
ShapeNo	93
TextOutPoint	93
ThePoints	93
3 Methods	93
Assign	93
ComputerCenterPoint	94
Create	94
Destroy	94
Draw	94
GetCenterPoint	94
GetCenterPointInZoom	95
GetHeight	95
GetLeftBottom	95
GetLeftTop	95
GetLinkPoint	96

GetLinkPointInZoom	96
GetMyHeight	96
GetMyWidth	96
GetPoint	97
GetPointInZoom	97
GetPointsCount	97
GetRightBottom	97
GetRightTop	97
GetShapeld	98
GetWidth	98
HasChildShapes	98
HasLinkShapes	98
HasParentShape	99
LoadFromStream	99
SaveToStream	99
4 Properties	100
Angle	100
Brush	100
Caption	101
CaptionShow	101
ColorBegin	101
ColorEnd	101
Font	102
GradientStyle	102
Info	102
IsFlipHorz	102
IsFlipVert	103
Locked	103
Name	103
Owner	103
Pen	104
Tag	104
UserData	104
Visible	104
Part VIII TMyCombine	106
1 ClassDiagram	106
Part IX TMyEllipse	108
1 ClassDiagramofTMyEllipse	108
2 Methods	108
Draw	108
GetCenterPoint	108
GetCenterPointInZoom	109
Part X TMyGroup	111
1 ClassDiagram	111
2 Methods	111
Draw	111
Part XI TMyImage	113
1 ClassDiagram	113
2 Properties	114
Bitmap	114

Border	114
Brightness	114
Contrast	115
Grayscale	115
Transparent	115
3 Methods	116
Assign	116
Create	116
Destroy	116
Draw	116
LoadFromStream	117
SaveToStream	117

Part XII TMyLinkLine 119

1 ClassDiagram	119
2 Methods	119
Draw	119

Part XIII TMyEllliArc 121

1 ClassDiagram	121
2 Properties	121
ArcMode	121
ArcStyle	122
3 Methods	123
Assign	123
Create	123
Draw	123
GetCenterPoint	124
GetCenterPointInZoom	124
LoadFromStream	124
SaveToStream	124

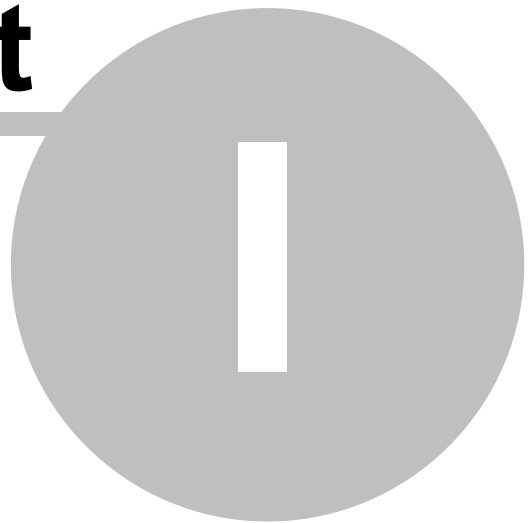
Part XIV TMyLinkLine 126

1 ClassDiagram	126
2 Properties	127
LinkLineDrawStyle	127
StartSpPtId	127
StartSpNo	127
EndSpNo	128
EndSpPtId	128
3 Methods	128
Assign	128
Create	128
Draw	128
LoadFromStream	129
CreateDestLink	129
CreateSrcLink	129
GetEndPoint	130
GetEndShape	130
GetStartPoint	130
RemoveDestLink	130
RemoveSrcLink	130
GetStartShape	131
RemoveAllLink	131

SaveToStream	131
Part XV TMyLine	133
1 ClassDiagram	133
2 Properties	134
ArrowAngle	134
ArrowLength	134
ArrowOffset	134
ArrowStyle	134
3 Methods	135
Assign	135
Create	135
Draw	135
GetInfo	135
IsClickedMeBeforeWhichPoint	135
LoadFromStream	136
SaveToStream	136
Part XVI TMyPolyBezier	138
1 ClassDiagram	138
2 Methods	138
Draw	138
Part XVII TMyPolygon	141
1 ClassDiagram	141
2 Methods	141
Draw	141
Part XVIII TMyPolyLine	143
1 ClassDiagram	143
2 Methods	143
Create	143
Part XIX TMyText	145
1 ClassDiagram	145
2 Properties	145
HAlignment	145
IsBorder	146
IsSolid	146
Lines	146
VAlignment	146
WordWrap	147
3 Methods	147
Assign	147
SaveToStream	147
LoadFromStream	148
Create	148
Destroy	148
Draw	148
Part XX TMyLinkPoint	150
1 ClassDiagram	150

2 Properties	150
Size	150
3 Methods	150
SaveToStream	150
LoadFromStream	151
Create	151
Draw	151
Part XXI TMyRuleLine	153
1 ClassDiagram	153
2 Properties	153
UserInfo	153
ShowUserInfo	154
TickStyle	154
3 Methods	154
Assign	154
Create	154
LoadFromStream	155
Draw	155
SaveToStream	155
Part XXII TMyRectangle	157
1 ClassDiagram	157
2 Methods	157
Draw	157
GetCenterPoint	157
GetCenterPointInZoom	158
GetInfo	158
Part XXIII TUserData	160
1 ClassDiagram	160
2 Property	160
UserDataRecords	160
3 Methods	160
Create	160
AddKeyAndValue	161
Assign	161
ChangeValueByKey	162
ClearAll	162
DeleteRecordByKey	162
GetCount	162
GetKeyByNo	163
GetValueByKey	163
ReNameKey	163
Part XXIV About Crystal Component	165
Index	166

Part



1 License

SOFTWARE LICENSE AGREEMENT

NOTICE---READ BEFORE USING THIS SOFTWARE

CAREFULLY READ THE TERMS AND CONDITIONS OF THIS AGREEMENT BEFORE USING THIS PACKAGE, USING THIS PACKAGE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS.

DEFINITIONS:

The Software Product. The Software Product licensed under this Agreement consists of computer programs, data compilation(s), and documentation referred to as TCAD for Delphi & C++ Builder (the "Software Product").

The Software Product is licensed (not sold) to You, and HongDi science & technology development co.,ltd. of Huzhou ,ZheJiang,China ("Vendor") owns all copyright, trade secret, patent and other proprietary rights in the Software Product.

LICENSE:

1. Evaluation Version License Grant. If You have downloaded or otherwise received an evaluation version of the Software Product, You are authorized to use the Software Product on a royalty-free basis for evaluation purposes during the initial evaluation period of thirty (30) days. During the evaluation period, You may copy the Software Product for archival purposes, provided that any copy must contain the original Software Product's proprietary notices in unaltered form, and You may distribute and/or transmit as many copies to others as You wish. You have the option to register for full use of the Software Product at any time during the evaluation period by following the instructions in the accompanying documentation, including the payment of the required license fee. Your use of the Software Product for any purpose after the expiration of the initial evaluation period is not authorized.

2. Registered Version License Grant For Single Copies (Non-Network Use). If You are a registered user of the Software Product, You are granted non-exclusive rights to install and use the Software Product in accordance with either one of the following authorized uses, but not both: (i) by a single person who uses the Software Product only on one or more computers or workstations, or (ii) as installed on any single computer or workstation, provided the single computer or workstation is used non-simultaneously by multiple persons. You may copy the Software Product for archival purposes, provided that any copy must contain the original Software Product's proprietary notices in unaltered form.

3. Registered Version License Grant For Network Use. If You are a registered user of the Software Product, You are granted non-exclusive rights to install and use the Software Product and/or transmit the Software Product over an internal computer network, provided You acquire and dedicate a licensed copy of the Software Product for each user who may access the Software Product concurrently with any other user. If a copy of the Software Product is used concurrently, then You must have some software mechanism which locks out any concurrent users in excess of the number of licensed copies of the Software Product. You may copy the Software Product for archival

purposes, provided that any copy must contain the original Software Product's proprietary notices in unaltered form.

4. Purchase of Additional Licenses. Registered users of the Software Product may purchase license rights for additional authorized use of the Software Product in accordance with Vendor's then-current volume pricing schedule. Such additional licenses shall be governed by the terms and conditions hereof. You agree that, absent Vendor's express written acceptance thereof, the terms and conditions contained in any purchase order or other document issued by You to Vendor for the purchase of additional licenses, shall not be binding on Vendor to the extent that such terms and conditions are additional to or inconsistent with those contained in this Agreement.

RESTRICTIONS:

You may not: (i) permit others to use the Software Product, except as expressly provided above for authorized network use; (ii) modify the Software Product; (iii) copy the Software Product, except as expressly provided above; or (iv) remove or obscure any proprietary rights notices or labels on the Software Product.

TRANSFERS:

You may not transfer the Software Product or any rights under this Agreement without the prior written consent of Vendor, which consent shall not be unreasonably withheld. A condition to any transfer or assignment shall be that the recipient agrees to the terms of this Agreement. Any attempted transfer or assignment in violation of this provision shall be null and void.

LIMITED SOFTWARE PRODUCT WARRANTY:

Vendor does not warrant the contents of the Software Product or that it will be error free. The Software Product is furnished "AS IS" and without warranty as to the performance or results You may obtain by using the Software Product. The entire risk as to the results and performance of the Software Product is assumed by You.

SPECIFIC EXCLUSION OF OTHER WARRANTIES:

The warranty provided above is in lieu of all other warranties, and there are no other warranties, representations or guarantees of any kind whatsoever, either express or implied, whether arising by statute, agreement, tort, product liability or otherwise, regarding the Software Product, or any other materials to be supplied by Vendor, including warranties as to merchantability, fitness for purpose, design, condition or quality.

NO CONSEQUENTIAL LOSS:

In no event will Vendor or its third party suppliers be liable to You for lost profits, lost savings or any punitive, exemplary, incidental, consequential or special damages arising out of the possession or use of the Software Product, or any other materials to be supplied hereunder.

LIMITATION OF DAMAGES:

If, despite the foregoing, for any reason Vendor becomes liable to You, Vendor's liability will be limited to the amounts paid by You to Vendor for those units of Software Product licensed which have given rise to such liability.

PERMITS:

You are exclusively responsible for obtaining any approvals, permits, licenses or other permissions necessary for You to export, import, possess, install, use or operate the Software Product in a territory, unless otherwise agreed to in writing by Vendor. This includes, but is not limited to, in the case of telecommunications products, obtaining applicable licenses from any telecommunications agencies, authorities or companies having jurisdiction before installing, interfacing, interconnecting or operating the Software Product.

EXCLUSIONS:

The Software Product is not specifically designed, manufactured or intended for use as parts, components or assemblies for the planning, construction, maintenance, operation or use of any nuclear facility nor for the flight safety or navigation of aircraft or ground support equipment, nor for use in any medical device or life-sustaining application. If You are using the Software Product for these applications You agree that, the Vendor is not liable, in whole or in part, for any claims or damages arising from such use and You agree to indemnify and hold Vendor harmless from any claims for lost, cost, damage, expense or liability arising out of or in connection with the use and performance of the Software Product in such excluded applications.

TERMINATION:

This Agreement is effective until terminated. You may terminate it at any time by destroying the Software Product, including all computer programs and documentation, and erasing any copies residing on computer equipment. This Agreement also will terminate if You do not comply with any terms or conditions of this agreement. Upon such termination You agree to destroy the Software Product and erase all copies residing on computer equipment.

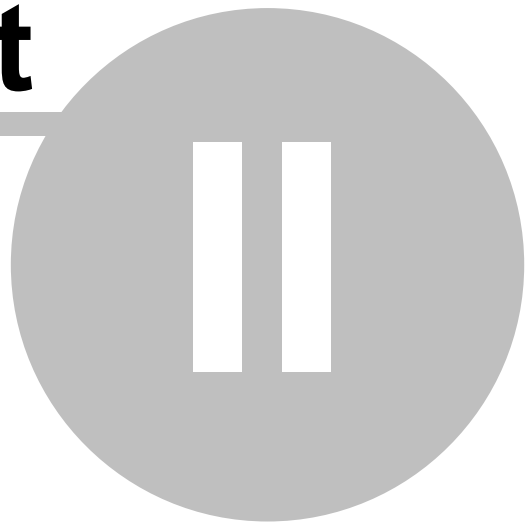
ENTIRE AGREEMENT:

This Agreement constitutes the entire Agreement between the parties as to the subject matter hereof, and supersedes and replaces all prior or contemporaneous agreements, written or oral, regarding such subject matter, and shall take precedence over any additional or conflicting terms which may be contained in Your purchase orders or Vendor's acknowledgement thereof.

Copyright (c) 2004 HongDi science & technology development co.,ltd. of Huzhou
All Rights Reserved

<http://www.codeidea.com>

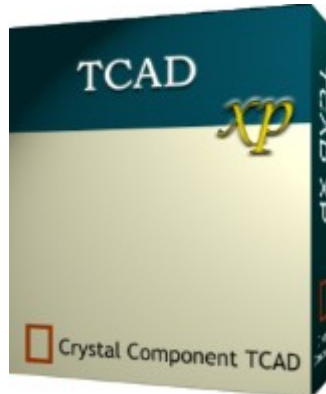
Part



2 Introduction

2.1 What is TCAD

TCAD is a component that will help you write vector graphics applications. Shapes can be interacted with by mouse or code. It is easy to use, effective and powerful. It will save you valuable time.



If you want add some CAD-Drawing function into application , using OLE mode, it is too tired to understand its concept..., programming it from zero, There is more work waiting for you! Now, You can using TCAD to help you write application.Easy to create and use VECTOR shape in your application developing ,only controlled by mouse. Now, You can using TCAD to help you writing application.

Key Features

Shape Types

- Line
- RuleLine
- Polyline
- Polygon
- PolyBezier
- Rectangle
- Arc
- Ellipse
- Text
- Bitmap
- Userdefine shapes

Grouping/Ungrouping

Multi-Layers

Create/ Move /Rotate shape by code

Save/Load to/From DiskFile/Database

Easy to create user-define shape

Support 4 mode coordinates

Detail infomation

Drawing shapes on the designer canvas by mouse actions or code.

Modifying the drawn shapes.

Support multi-layers, printing/deleting/visible invisible layer(s).

Using all colors possible.

Supporting link line shape

Using different style of pens ,different style of brushes if you need.

Creating text objects with any font installed in the system.

Necessarily shape action related events published.

Using page formats like (A0,A1,A2,A3,A4,letter, etc.) or custom sizes.

Can Undo and set undo step size

Cutting, copying, pasting and deleting the shapes.

Ordering the shapes(SendToBack, BringToFront, etc.)

Rotating, Dragging and Scaling the shapes by mouse or code.

Aligning the shape in any style.

Easy to create user-define combine shape

Snapping the mouse point to grids,set grid width or height.

Support 24 gradient style fill mode

Locking/Unlocking Shape

Showing HotSpot of a shape or hiding.

Grouping and ungrouping the shapes.

Zooming and panning, viewing the drawing in any scale.

Showing hints when mouse enter a shape.

Saving the drawing as disk file or stream(database) and opening it.

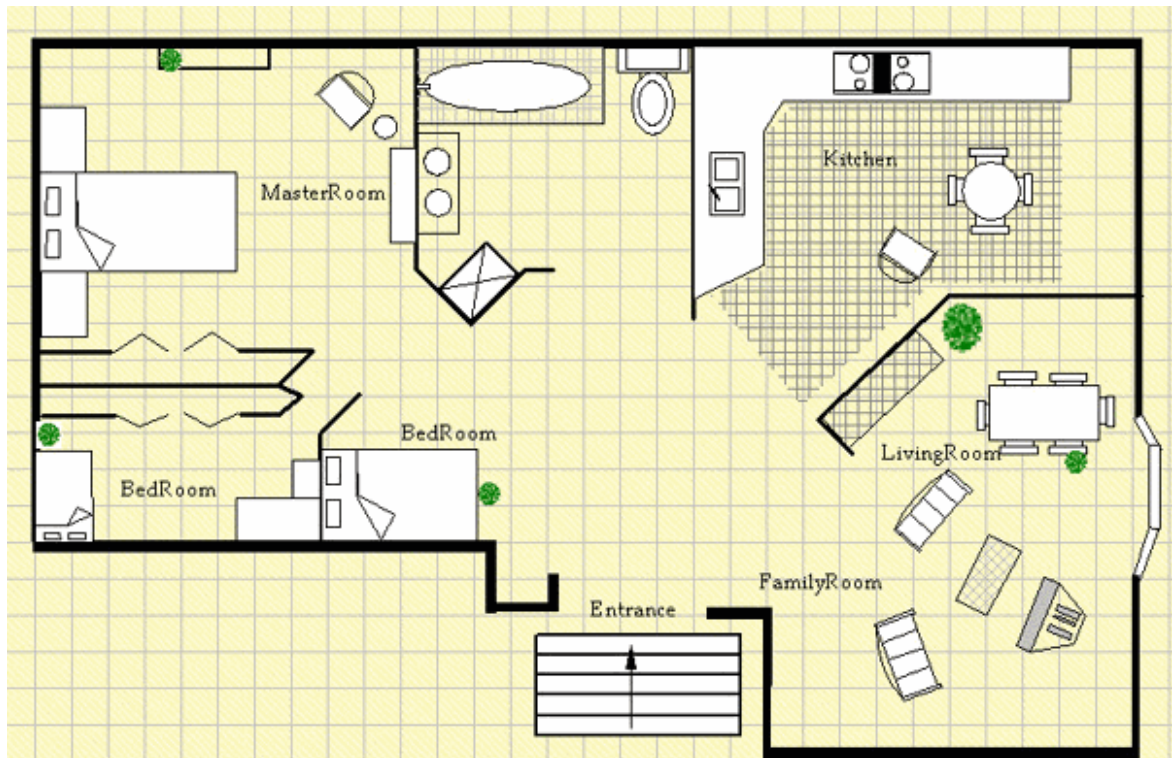
Printing the drawing to the printer and/or plotter.

Inserting bitmaps to the drawing.

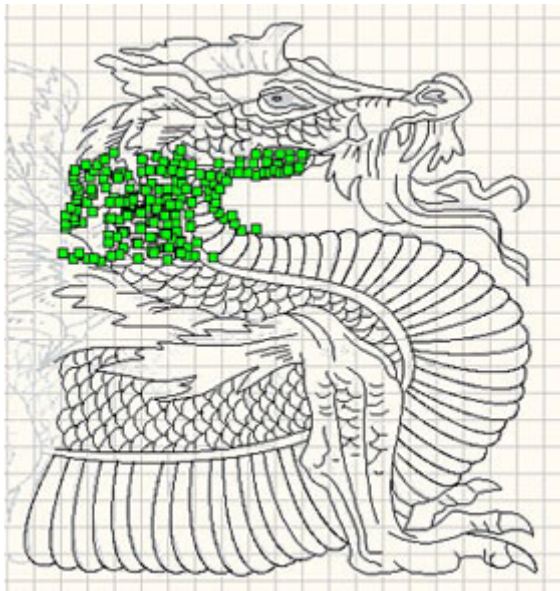
Scaling, rotating,dragging bitmaps like a shape.

Exporting the drawing as WMF,bitmap,Jpg,dxf(R12) file.

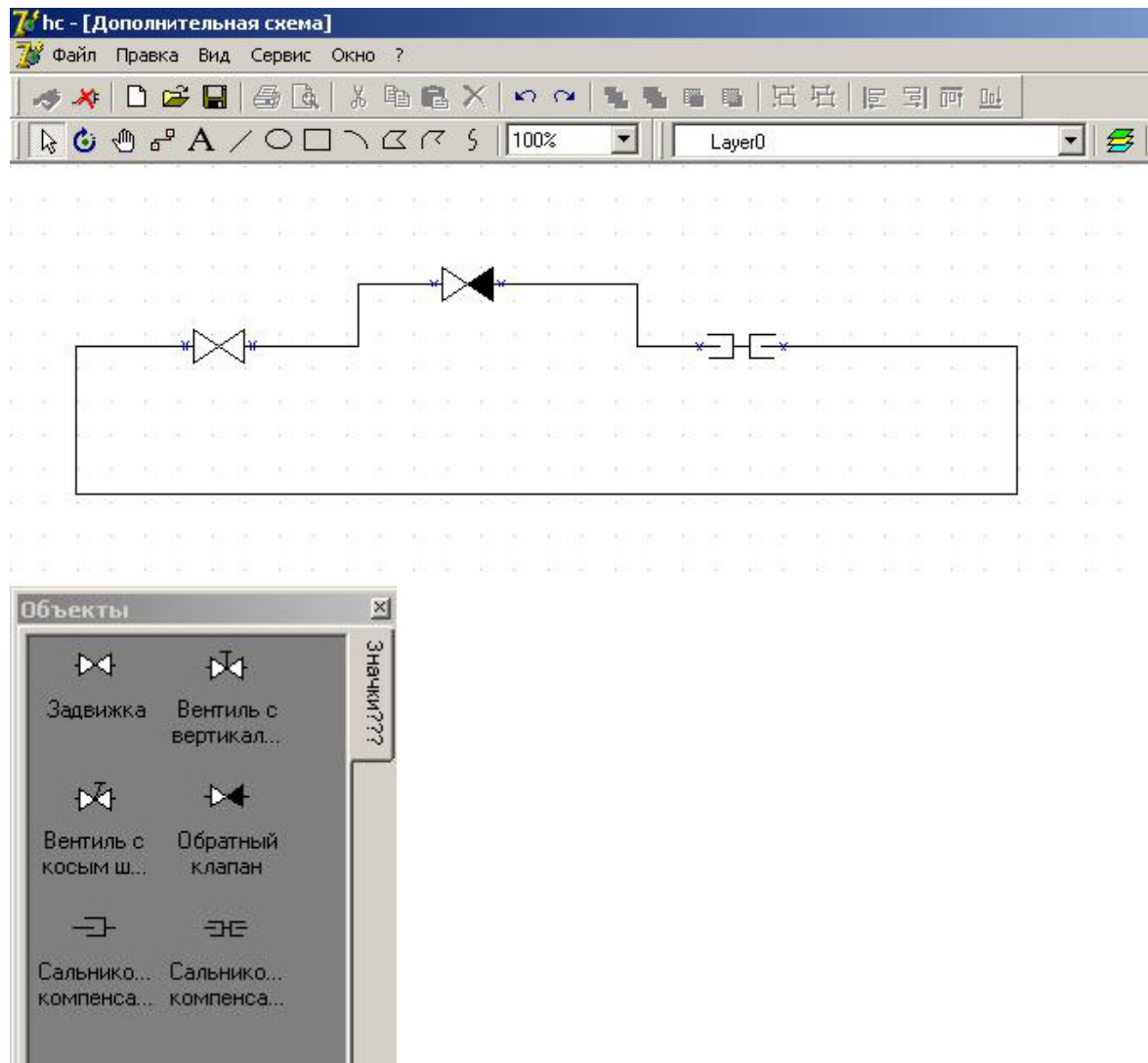
2.2 Application ScreenShot



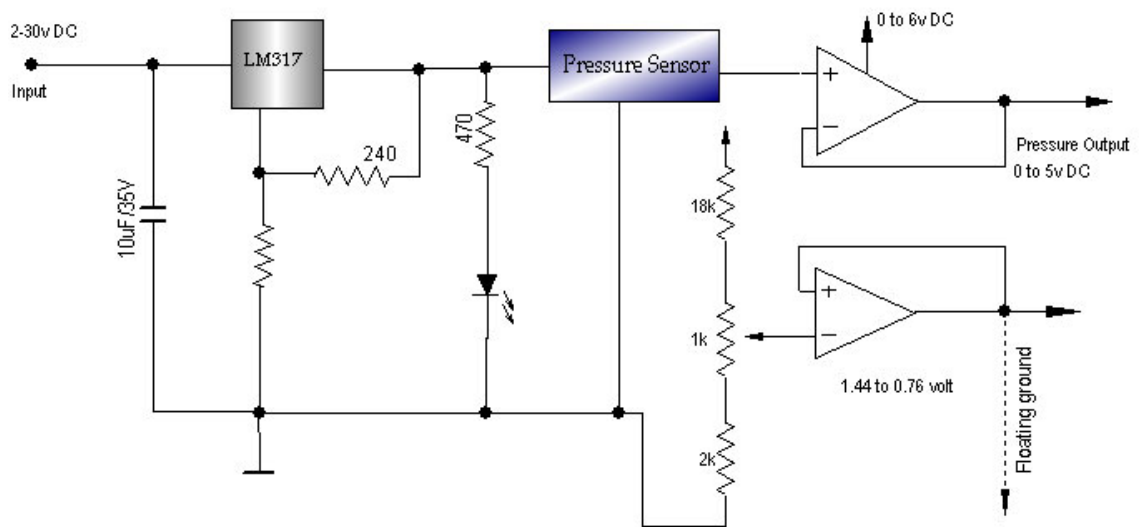
It is a house plan drawing sample,no lib need to draw this,fast and easy.



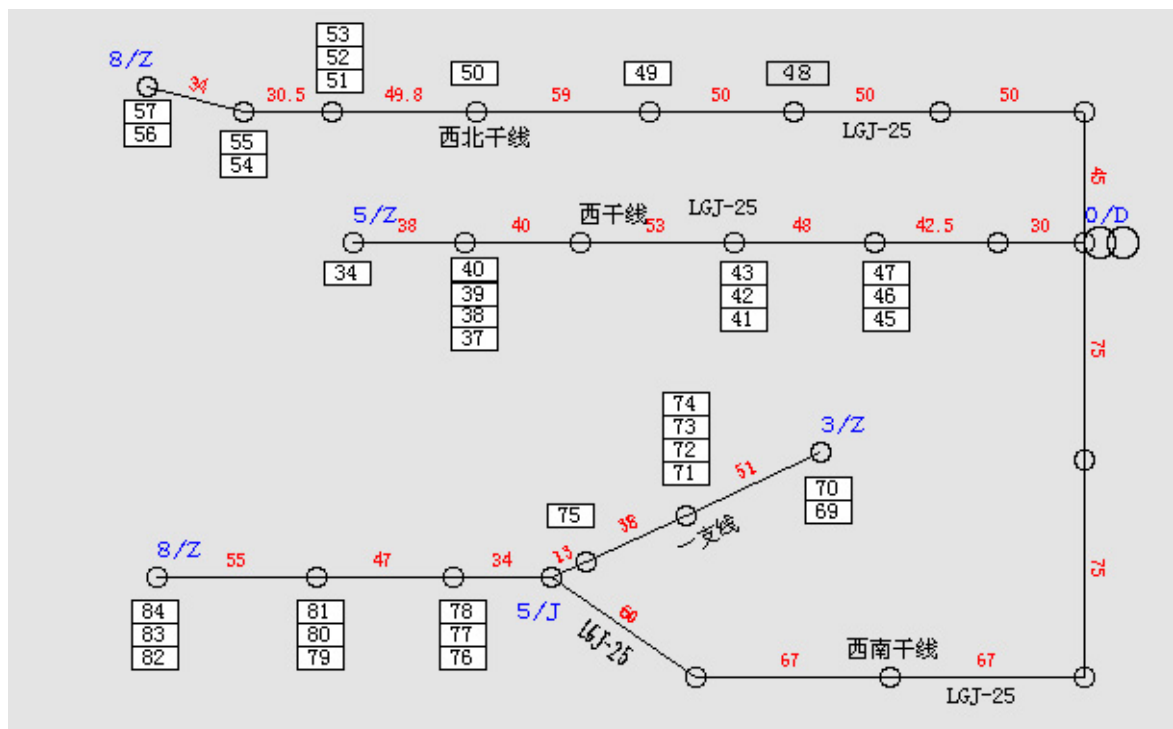
Design of vector drawings from cnc machines .Using TMyElliArc to create this complex dragon.



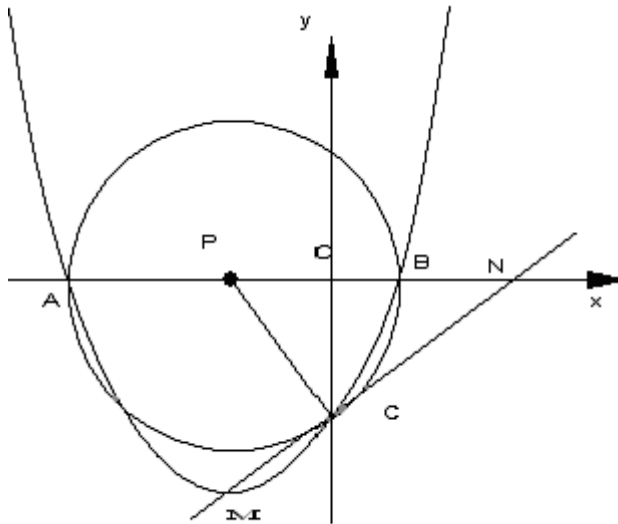
Hydraulic Design



Electric drawing, library need, and support link line, TCAD is a powerful for "link line style" drawing.

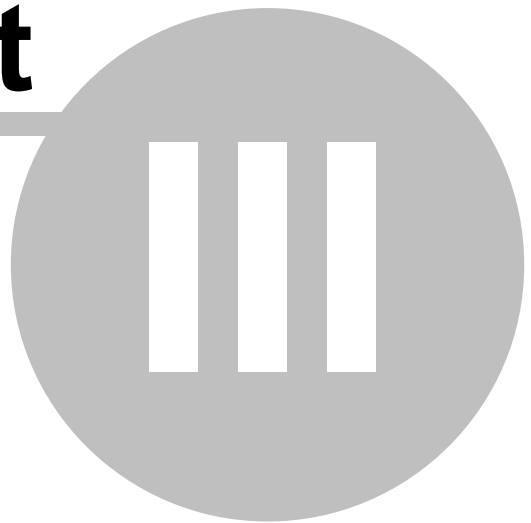


Project: Electric device manager
Company: xian college, China



If you are a teacher or student, TCAD can draw math graph, you can copy the graph to MS Word .
Creating math graph library, can speed your work.

Part



3 Defines

```

TMyPoint = record
    x:single;
    y:single;
end;

TDrawTool = (SpClose,SpSelecting, SpLine, SpRuleLine,SpPolyLine, SpPolygon,
    spPolyBezier,SpRectangle,SpEllipse, SpText ,SpElliArc,Splmage );

TPageStyle = (A0,A1,A2,A3, A4, A5, B3, B4, B5, CustomerPage);

TDragMode= (dgHorz,dgVert,dgBoth);

TXYMode =(Mode0,Mode1,Mode2,Mode3);

TGridType =(gPixel,gLine,gNone);

TUnits = (pixel, mm,dm,m, inch);

TBkBitmapMode = (Tiled,Stretch, Center, LeftTop);

TArrowStyle = (ANone,ALeft,ARight,ADouble);

TBlockLayerMode= (Merge,Import);

TXYMode = (Mode0,Mode1,Mode2,Mode3)

TGradientStyle = ( gsRadialC, gsRadialT, gsRadialB, gsRadialL,
    gsRadialR, gsRadialTL, gsRadialTR, gsRadialBL, gsRadialBR, gsLinearH,
    gsLinearV, gsReflectedH, gsReflectedV, gsDiagonalLF, gsDiagonalLB,
    gsDiagonalRF, gsDiagonalRB, gsArrowL, gsArrowR, gsArrowU, gsArrowD,
    gsDiamond, gsButterfly,gsNone);


TArrUserDataRecord=array of Record
    Key:string;
    Value:String;
END;

```

Part

IV

4 TGridObject

 TGridObject
attributes
* GridColor: TColor * GridHeight: Double * GridPenSize: Byte * GridShow: Boolean * GridType: TGridType * GridWidth: Double
operations
+ Create + Destroy



4.1 GridColor

property GridColor:TColor;

Description:

Set the grid color .

Example:

```
MyCAD1.GridOperation.GridColor:=clBlack;
```

See also:

[GridWidth](#)

[GridHeight](#)

[GridType](#)

[GridPenSize](#)

4.2 GridHeight

property GridHeight:byte;

Description:

The height of grid , it is in pixels unit.

Example:

```
MyCAD1.GridOperation.GridHeight:=12;
```

See also:

[GridWidth](#)

[GirdType](#)

[GridShow](#)

4.3 GridPenSize

property GridPenSize:Byte;

Description:

The grid line or pixel size

Example:

```
MyCAD1.GridOperation.GridPenSize := 3;
```

See also:

[GridWidth](#)

[GridHeight](#)

[GirdType](#)

[GridColor](#)

4.4 GridShow

property GridShow:Boolean;

Description:

The grid show or not.

Example:

```
MyCAD1.GridOperation.GridShow := false;
```

See also:

[GridWidth](#)

[GridHeight](#)

[GirdType](#)

4.5 GridType

property GridType: [TGridType](#)

Description:

The grid is made of pixel or line .

Example:

```
MyCAD1.GridOperation.GridType:=gLine;
```

See also:

[GridWidth](#)

[GridHeight](#)

[GridShow](#)

4.6 GridWidth

property GridWidth:byte;

Description:

The width of grid , the unit is in pixels

Example:

```
MyCAD1.GridOperation.GridWidth:=12;
```

See also:

[GridHeight](#)

[GridType](#)

[GridShow](#)

Part



V

5 TMyCAD

TMyCAD is a powerful 2D drawing component, it is can be used at delphi and c++ Builder.

5.1 ClassDiagram

5.1.1 Part1 Properties

 TMyCAD	
attributes	
+ Global_Changed: Boolean	* PageHead: string
+ IsLoading: Boolean	* PageHeadAlignment: TAlignment
+ MyShapes: TArrMyShape	* PageHeadFont: TFont
+ NextShapeld: Integer	* PageHeadToTop: Byte
+ WorkingShapes: TArrMyShape	* PageHeight: Cardinal
+ Canvas: Integer	* PageOrientation: TPrinterOrientation
+ CurrentLayerId: Integer	* PageStyle: TPageStyle
+ DiskFileVersion: string	* PageWidth: Cardinal
* ArrowAngle: Integer	* Pen: TPen
* ArrowLength: Byte	* PrintABorder: Boolean
* ArrowOffset: Byte	* PrintABordertoBottom: Byte
* ArrowStyle: TArrowStyle	* PrintABordertoLeft: Byte
* BkBitmap: TBitmap	* PrintABordertoRight: Byte
* BkBitmapMode: TBkBitmapMode	* PrintABordertoTop: Byte
* Brush: TBrush	* PrintBackground: Boolean
* ColorOfBackGround: TColor	* Ratio: Double
* ColorOfHot: TColor	* ResizeEnable: Boolean
* CrossLine: Boolean	* ReturnToSelecting: Boolean
* DragMode: TDragMode	* RotateConstraintDegree: Integer
* Enabled: Integer	* RotateEnable: Boolean
* Font: TFont	* ShapeTool: TDrawTool
* GridOperation: TGridObject	* ShowHint: Boolean
* HotShow: Boolean	* ShowHotLink: Boolean
* HotSize: Byte	* Snap: Boolean
* LabelValue: TLabel	* SnapPixels: Byte
* LabelXY: TLabel	* SnapShape: Boolean
* LinklineAroundShape: Boolean	* TheUNIT: TUnits
* LinkLineDrawStyle: TLinkLineDrawStyle	* UndoRedoSize: Byte
* OperateAllLayer: Boolean	* Version: string
* PageFoot: string	* Visible: Integer
* PageFootAlignment: TAlignment	* XYMode: TXYMode
* PageFootFont: TFont	* Zoom: Double
* PageFootToBottom: Byte	

5.1.2 Part2 Events

 TMyCAD
events
* OnActionToolToSelecting: TActionToolToSelecting
* OnClick: TNotifyEvent
* OnDbClick: TNotifyEvent
* OnDeleteLayer: TLayerOp
* OnDragDrop: TNotifyEvent
* OnDragOver: TNotifyEvent
* OnMouseDown: TNotifyEvent
* OnMouseEnter: TNotifyEvent
* OnMouseEnterShape: TEnterLeaveShape
* OnMouseLeave: TNotifyEvent
* OnMouseLeaveShape: TEnterLeaveShape
* OnMouseMove: TNotifyEvent
* OnMouseUp: TNotifyEvent
* OnNewLayer: TLayerOp
* OnPaint: TNotifyEvent
* OnShapeAdded: TShapeAdded
* OnShapeCodeDragging: TShapeCodeDraggingSizing
* OnShapeCodeRotating: TShapeCodeRotating
* OnShapeDeleted: TShapeDeleted
* OnShapeMouseDragged: TShapeMouseDraggingSizingRotating
* OnShapeMouseDragging: TShapeMouseDraggingSizingRotating
* OnShapeMouseResized: TShapeMouseDraggingSizingRotating
* OnShapeMouseResizing: TShapeMouseDraggingSizingRotating
* OnShapeMouseRotated: TShapeMouseDraggingSizingRotating
* OnShapeMouseRotating: TShapeMouseDraggingSizingRotating
* OnShapeSelected: TShapeSelected

5.1.3 Part3 Methods

TMyCAD	
operations	
+ Create(..)	+ GetShapeByNo(..): TMyShape
+ Destroy	+ GetShapeNoByld(..): Integer
+ AddBlockfromTCADFile(..): Boolean	+ GetShapesCount: Integer
+ AddImageShapeByCode(..): Integer	+ GetShapesCountInALayer(..): Integer
+ AddShape(..): Boolean	+ GroupWorkingShape: Integer
+ AddShapeByCode(..): Integer	+ InVisibleLayerByld(..)
+ AddUserDefineShapefromLib(..): Integer	+ InVisibleLayerByName(..)
+ AlignBottom	+ IsLinked(..): Integer
+ AlignLeft	+ IsTCADFile(..): Boolean
+ AlignRight	+ IsVisibleLayerByld(..): Boolean
+ AlignTop	+ LoadFromFile(..): Boolean
+ BringToFront(..)	+ LoadFromStream(..): Boolean
+ BringToFrontByStep(..)	+ LockUnLockforShapes(..)
+ ClearAllUndoStuff	+ MergeLayers(..): Boolean
+ Copy	+ Move(..)
+ CopyToClipboardAsWmf	+ NewLayer(..): Integer
+ CreateLink(..): Integer	+ Paste
+ Cut	+ PopFromUndoRedoShapeList: Integer
+ DeleteAllLayers: Boolean	+ Print(..)
+ DeleteAllShapes	+ PrintPreview(..)
+ DeleteLayerByld(..): Boolean	+ ReleaseCursorClipArea
+ DeleteLayerByName(..): Boolean	+ RenameShapeName(..)
+ DeleteSelectedShapes: Boolean	+ Resize
+ DeleteShapeByld(..): Boolean	+ Rotate(..)
+ DeSelectedAllShapesByCode	+ SaveToBmp(..)
+ DrawAllShape(..)	+ SaveToBmp(..)
+ FlipHoriz(..)	+ SaveToDxf(..)
+ FlipVert(..)	+ SaveToFile(..): Boolean
+ GetLayerldByName(..): Integer	+ SaveToFileold(..): Boolean
+ GetLayerldByNo(..): Integer	+ SaveToJpg(..)
+ GetLayerNameByld(..): string	+ SaveToStream(..): Boolean
+ GetLayerNoByld(..): Integer	+ SaveToWmf(..)
+ GetLayerNoByName(..): Byte	+ SelectAllShapes
+ GetLayersCount: Byte	+ SelectShapeByCode(..): Boolean
+ GetMaxLayerld: Integer	+ SendToBack(..)
+ GetMemShapesCount: Integer	+ SendToBackByStep(..)
+ GetMyPointByMode(..): TMyPoint	+ SetLayerNameByld(..)
+ GetPointByMode(..): TPoint	+ SetLayerNameByName(..)
+ GetRA1A2By3Points(..)	+ SetMyImage(..): Boolean
+ GetSelectedShape: TMyShape	+ SizeShape(..)
+ GetSelectedShapes: TArrMyShape	+ UngroupShape(..)
+ GetSelectedShapesCount: Integer	+ VisibleAllLayer
+ GetShapeByld(..): TMyShape	+ VisibleLayerByld(..)
+ GetShapeByName(..): TMyShape	+ VisibleLayerByName(..)

5.2 Events

5.2.1 OnActionToolToSelecting

TActionToolToSelecting = procedure() of object;

property OnActionToolToSelecting: TActionToolToSelecting

Description:

Use OnActionToolToSelecting to handle when the ShapeTool return to SpSelecting state,

Example:

```
procedure TMainFrm.MyCAD1ActionToolToSelecting;  
begin  
    SelectBtn.Down := true;  
end;
```

5.2.2 OnClick

Please read Delphi or C++ Builder Help file.

5.2.3 OnDbClick

Please read Delphi or C++ Builder Help file.

5.2.4 OnDeleteLayer

TLayerOp = procedure(LayerId: integer; LayerName: string; Visible: Boolean) of object;

property OnDeleteLayer: TLayerop

Parameters:

LayerId

The id of this layer.

LayerName

The name of this layer.

Visible

This layer is visible or not.

Description:

Use OnDeleteLayer to handler a layer be deleted.

5.2.5 OnDragDrop

property OnDragDrop: TDragDropEvent;

Delphi syntax:

```
type TDragDropEvent = procedure(Sender, Source: TObject; X, Y: Integer) of object;
property OnDragDrop: TDragDropEvent;
```

C++ syntax:

```
typedef void __fastcall (__closure *TDragDropEvent)(System::TObject* Sender,
System::TObject* Source, int X, int Y);
__property TDragDropEvent OnDragDrop = {read=FOnDragDrop, write=FOnDragDrop};
```

Description:

Occurs when the user drops an object being dragged.

Use the OnDragDrop event handler to specify what happens when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

5.2.6 OnDragOver

Occurs when the user drags an object over a control.

Delphi syntax:

```
type
TDragOverEvent = procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var
Accept: Boolean) of object;
property OnDragOver: TDragOverEvent;
```

C++ syntax:

```
typedef void __fastcall (__closure *TDragOverEvent)(System::TObject* Sender,
System::TObject* Source, int X, int Y, TDragState State, bool &Accept);
__property TDragOverEvent OnDragOver = {read=FOnDragOver, write=FOnDragOver};
```

Description

Use an OnDragOver event to signal that the control can accept a dragged object so the user can drop or dock it.

Within the OnDragOver event handler, change the Accept parameter to false to reject the dragged object. Leave Accept as true to allow the user to drop or dock the dragged object on the control.

To change the shape of the cursor, indicating that the control can accept the dragged object, change the value of the DragCursor property for the control before the OnDragOver event occurs.

The Source is the object being dragged, the Sender is the potential drop or dock site, and X and Y are screen coordinates in pixels. The State parameter specifies how the dragged object is moving over the control.

Note: Within the OnDragOver event handler, the Accept parameter defaults to true. However, if an OnDragOver event handler is not supplied, the control rejects the dragged object, as if the

Accept parameter were changed to false.

5.2.7 OnMouseDown

Please read Delphi or C++ Builder Help file.

5.2.8 OnMouseEnter

property OnMouseEnter:TNotifyEvent

Description:

When the mouse enter TMyCAD, this event trigger.

Example:

```
procedure TForm1.MyCAD1MouseEnter(Sender: TObject);
begin
  Memo1.Lines.Add('-----Enter TCAD Event-----');
end;
```

See Also:

[OnMouseLeave](#)

5.2.9 OnMouseEnterShape

TEnterLeaveShape = procedure (AShape:TMyShape) of object;

property OnMouseEnterShape:TNotifyEvent

Description:

When the mouse enter a shape or a grouped shape, this event trigger.

Example:

```
procedure TForm1.MyCAD1MouseEnterShape(AShape: TMyShape);
begin
  Memo1.Lines.Add('-----Enter Shape Event-----');
  Memo1.Lines.Add('Enter shape, id is : ' + inttostr(AShape.ShapeID));
end;
```

See Also:

[OnMouseLeaveShape](#)

5.2.10 OnMouseLeave

property OnMouseLeave:TNotifyEvent

Description:

When the mouse leave TMyCAD, this event trigger.

Example:

```
procedure TForm1.MyCAD1MouseLeave(Sender: TObject);
begin
  Memo1.Lines.Add('-----Leave TCAD Event-----');
end;
```

See Also:

[OnMouseEnter](#)

5.2.11 OnMouseLeaveShape

TEnterLeaveShape = procedure (AShape:TMyShape) of object;

property OnMouseLeaveShape:TEnterLeaveShape

Description:

When the mouse leave a shape or a grouped shape, this event trigger.

Example:

```
procedure TForm1.MyCAD1MouseLeaveShape(ShapeID, LayId: Integer;
  AShape: TMyShape);
begin
  Memo1.Lines.Add('----- Leave Shape Event-----');
  Memo1.Lines.Add('Leave shape, id is : ' + inttostr(AShape.ShapeID));
end;
```

See also:

[OnMouseEnterShape](#)

5.2.12 OnMouseMove

Please read Delphi or C++ Builder Help file.

5.2.13 OnMouseUp

Please read Delphi or C++ Builder Help file.

5.2.14 OnNewLayer

TLayerOp = procedure(LayerId: integer; LayerName: string; Visible: Boolean) of object;

property OnNewLayer: TLayerOp ;

Description:

Use OnNewLayer to handle a new layer be added.

5.2.15 OnPaint

Please read Delphi or C++ Builder Help file.

5.2.16 OnShapeAdded

TShapeAdded = procedure(LayerId: integer; ShapeID: integer; ShapeName: string; AShape: TMyShape; ShapeCount: integer) of object;

property OnShapeAdded: TShapeAdded;

Description:

Use OnShapeAdded to handler a shape is added

Examples:

```
procedure TMainFrm.MyCAD1ShapeAdded( var AShape: TMyShape; ShapeCount: Integer);
begin
  EventMemo.Lines.Add('-----Add Event');
  EventMemo.Lines.Add('You Add a Shape,it is :' + ShapeName + ',in Layer ' +
    IntToStr(LayerId) + ', it is a ' + AShape.ClassName + ',Now CAD has ' +
    IntToStr(ShapeCount) + ' Shapes');
end;
```

5.2.17 OnShapeCodeDragging

TShapeCodeDraggingSizing = procedure (AShape:TMyShape; dx,dy:Single) of object;

property OnShapeCodeDragging

Description:

When a Shape is Dragging by code , this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeCodeRotating(AShape: TMyShape;dx,dy: single);
begin
  Memo1.Lines.Add('-----code Drag Event-----');
  Memo1.Lines.Add('ShapeId is: '+IntToStr(AShape.ShapeId)+' '+AShape.Name + ' rotating ');
end;
```

See Also:

[Move](#)

5.2.18 OnShapeCodeRotating

TShapeCodeRotating = procedure (AShape:TMyShape; AAngle:Single) of object;

property OnShapeCodeRotating

Description:

When a Shape is Rotating by code , this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeCodeDragging( AShape:TMyShape;AAngle:single);
begin
    Memo1.Lines.Add('-----code Rotate Event-----');
    Memo1.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+' '+AShape.Name +' rotating ');
end;
```

See Also:

[Rotate](#)

5.2.19 OnShapeDeleted

TShapeDeleted = procedure (ShapeID:integer;LayerId:integer;AShape:TMyShape) of object ;

property OnShapeDeleted: TShapeDeleted ;

Description:

Use OnShapeDeleted to handle a shape is deleted.

5.2.20 OnShapeMouseDownDragged

TShapeDraggingSizingRotated = procedure
(AShape:TMyShape;FromPoint:TPoint;ToPoint:TPoint) of object;

property OnShapeMouseDownDragged:TShapeDraggingSizingRotating

Description:

When a Shape is dragging by mouse , this event be trigger.

Example:

```
procedure TMainFrm.MyCAD1ShapeMouseDownDragged(AShape: TMyShape; FromPoint, ToPoint: TPoint);
begin
  Memol.Lines.Add('-----Dragged Event-----');
  Memol.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+AShape.Name + ' dragged ,'+
    'From x:'+ inttostr(FromPoint.x)+' y:' +inttostr(FromPoint.y)+
    'To x:'+ inttostr(ToPoint.x)+' y:' +inttostr(ToPoint.y));
end;
```

See Also:

[Move](#)

5.2.21 OnShapeMouseDownDragging

TShapeDraggingSizingRotated = procedure
(AShape:TMyShape;FromPoint:TPoint;ToPoint:TPoint) of object;

property OnShapeMouseDownDragging:TShapeDraggingSizingRotating

Description:

When a Shape is dragging by mouse , this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeMouseDownDragging( AShape: TMyShape; FromPoint, ToPoint: TPoint);
begin
  Memol.Lines.Add('-----Dragged Event-----');
  Memol.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+AShape.Name + ' are dragging!');
end;

end;
```

See Also:

[Move](#)

5.2.22 OnShapeMouseDownResized

TShapeDraggingSizingRotating= procedure
(AShape:TMyShape;FromPoint:TPoint;ToPoint:TPoint) of object;

property OnShapeMouseDownResizing:TShapeDraggingSizingRotating

Description:

When a shape is resizing by mouse, this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeMouseResized(AShape: TMyShape; FromPoint, ToPoint: TPoint);
begin
    Memol.Lines.Add('-----Resized Event-----');
    Memol.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+AShape.Name + ' resized, '+
        'From x: '+ inttostr(FromPoint.x)+' y: ' +inttostr(FromPoint.y)+
        'To x: '+ inttostr(ToPoint.x)+' y: ' +inttostr(ToPoint.y));
end;
```

5.2.23 OnShapeMouseResizing

TShapeDraggingSizingRotating = procedure
(AShape:TMyShape;FromPoint:TPoint;ToPoint:TPoint) of object;

property OnShapeMouseResizing:TShapeDraggingSizingRotating

Description:

When a shape is resizing by mouse, this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeMouseResizing(AShape: TMyShape; FromPoint, ToPoint: TPoint);
begin
    Memol.Lines.Add('-----Resizing Event-----');
    Memol.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+AShape.Name + ' resizing ,'+
        'From x: '+ inttostr(FromPoint.x)+' y: ' +inttostr(FromPoint.y)+
        'To x: '+ inttostr(ToPoint.x)+' y: ' +inttostr(ToPoint.y));
end;
```

5.2.24 OnShapeMouseRotated

TShapeDraggingSizingRotating= procedure
(AShape:TMyShape;FromPoint:TPoint;ToPoint:TPoint) of object;

property OnShapeMouseRotated:TShapeDraggingSizingRotating

Description:

When a Shape is rotated by mouse, this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeMouseRotated(AShape: TMyShape; FromPoint, ToPoint: TPoint);
begin
    Memol.Lines.Add('-----Rotated Event-----');
    Memol.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+AShape.Name + ' rotated ');
end;
```

See Also:

[Rotate](#)

5.2.25 OnShapeMouseRotating

TShapeDraggingSizingRotating= procedure
(AShape:TMyShape;FromPoint:TPoint;ToPoint:TPoint) of object;

property OnShapeMouseRotating:TShapeDraggingSizingRotating

Description:

When a shape is rotating by mouse, this event be trigger.

Example:

```
procedure TForm1.MyCAD1ShapeMouseRotating(AShape: TMyShape; FromPoint, ToPoint: TPoint);
begin
    Memol.Lines.Add('-----Rotating Event-----');
    Memol.Lines.Add('ShapeId is: '+Inttostr(AShape.ShapeId)+AShape.Name + ' rotating ');
end;
```

See Also:

[Rotate](#)

5.2.26 OnShapeSelected

TShapeSelected = procedure (LastSelectedShape:TMyShape; SelectedShape: TMyShape) of object;

property OnShapeSelected: TShapeSelected ;

Description:

Use OnShapeSelected to handle a shape is selected and also can get the old shape's information.

Examples:

```
procedure TMainFrm.MyCAD1ShapeSelected(LastSelectedShape, SelectedShape: TMyShape);
begin
    Memol.Lines.Add('-----Select Event-----');
    if LastSelectedShape = nil then
        Memol.Lines.Add('No last shape selected')
    else
        Memol.Lines.Add(' Last shape id is ' + inttostr(LastSelectedShape.ShapeID)+
            ',in Layer ' + inttostr(LastSelectedShape.LayerId) +
            ', it is a ' + LastSelectedShape.ClassName);

        Memol.Lines.Add('Now you selected ' + inttostr(SelectedShape.ShapeID)+
            ',in Layer ' + inttostr(LayerId) +
            ', it is a ' + SelectedShape.ClassName);
end;
```

5.3 Methods

5.3.1 AddBlockfromTCADFile

function AddBlockfromTCADFile(const FileName:string;
BlockLayerMode:TBlockLayerMode):Boolean;

Description:

Add block from an exist tcad file , this file can be a template drawing.

Return Value:

true : add success
false : add failed

Parameter

TheName:Image shape's name
 LeftTop:the point of the image's Lfte-Top corner.
 ABitmap:a image that you want add into TCAD

See also:

[efines](#)

5.3.2 AddImageShapeByCode

function AddImageShapeByCode(TheName:String;LeftTop:TMyPoint;ABitmap:TBitmap):integer;

Description:

Add image shape by code, if you want add other shape , Please use procedure [AddShapeByCode](#) or [AddCmbShapeByCode](#)

Return Value:

-1: Add failed
 else(>=0): The new shape's shapeld.

Parameter

TheName:Image shape's name
 LeftTop:the point of the image's Lfte-Top corner.
 ABitmap:a image that you want add into TCAD

Example:

```
var
  myBitmap:TBitmap;
begin
  {Create a temp bitmap to load form a disk file}
  MyBitmap := TBitmap.Create;
  {Load from files}
  Mybitmap.LoadFromFile( ExtractFilePath(Application.ExeName) + 'images\1'+'.bmp');

  {a layer0 already exist by default}
  {Add Image into TMyCAD by code}
  MyCAD1.AddImageShapeByCode( 'MyImage',MyPoint(50,100),MyBitmap);
  {free it because no use}
  MyBitmap.Free;
end;
```

See also:

[AddShapeByCode](#)
[ddUserDefineShapefromLib](#)
[OnShapeAdded](#)

5.3.3 AddShapeByCode

function AddShapeByCode(Owner:TComponent;ShapeStyle:TDrawTool;
 ShapeName:string;ThePoints: array of TMyPoint;TheAngle:Extended=0;OnlyForText:String=""):
 integer;

Description:

Add shape by code, it is useful for automatic drawing . if you want add a image ,Please use AddImageShapeByCode.

Return Value:

0: Ok
-1: incorrect points count

Parameter:

Owner:instance of TMyCAD
ShapeStyle : Please see [DrawTool](#)
ShapeName: shape's name
ThePoints: Points array
TheAngle: The Angle of this new shape
OnlyForText: it is for TMyText shape only

Example:

```
MyCAD1.AddShapeByCode(MyCAD1,spEllipse, 'EllipseShape',[MyPoint(231, 211),
MyPoint(340,211),MyPoint(340, 350),MyPoint(231,350)]);
end;
```

See also:

[ddlImageShapeByCode](#)
[AddCmbShapeByCode](#)
[OnShapeAdded](#)

5.3.4 AddUserDefineShapefromLib

function AddUserDefineShapefromLib(ALibManager:TLibManager;
UDShapeName:string;CenterX,CenterY:integer;OwnerCAD:TMyCAD): Integer;

Description:

Get from library by UDshapename . and add user-defined shape into TMyCAD instance,it is a important function in TMyCAD

Return Value:

-1: Add failed
else(>=0): The new shape's shapeld.

Parameter:

ALibManager:the instance of a library
UDShapeName : the name of a combine shape;
CenterX,CenterY:the shape's center position that you want placed;
OwnerCAD:this shapw will be added into Which TMyCAD

Example:

```
var
  mLibManager:TLibManager;
begin
  if (FileExists(FileName)) and (IsTCADLibFile(FileName)) then
  begin
    mLibManager:=TLibManager.Create;
    mLibManager.LoadFromFile(LibFileName);
```

```
MyCAD1.AddUserDefineShapefromLib( mLibManager, 'MyShapename',100,100,MyCAD1);  
mLibManager.Free;  
end;  
end;
```

See also:

[AddShapeByCode](#)
[ddlImageShapeByCode](#)
[OnShapeAdded](#)

5.3.5 AlignBottom

```
procedure AlignBottom;
```

Description:

When you select more than one shape, this procedure can align them bottom.

Example:

```
MyCAD1.Alignbottom;
```

5.3.6 AlignHorizontalCenter

```
procedure AlignHorizontalCenter;
```

Description:

When you select more than one shape, this procedure can align them horizontal center.

Example:

```
MyCAD1.AlignHorizontalCenter;
```

5.3.7 AlignLeft

```
procedure AlignLeft;
```

Description:

When you select more than one shape, this procedure can align them left.

Example:

```
MyCAD1.AlignLeft;
```

5.3.8 AlignRight

procedure AlignRight;

Description:

When you select more than one shape, this procedure can align them right.

Example:

```
MyCAD1.AlignRight;
```

5.3.9 AlignTop

procedure AlignTop;

Description:

When you select more than one shape, this procedure can align them top.

Example:

```
MyCAD1.AlignTop;
```

5.3.10 AlignVerticalCenter

procedure AlignVerticalCenter;

Description:

When you select more than one shape, this procedure can align them vertical center.

Example:

```
MyCAD1.AlignVerticalCenter;
```

5.3.11 BringToFront

procedure BringToFront(AShape:TMyShape;NeedSave:boolean=true);

Description:

Sent the selected shape to front

Parameter:

AShape:the shape that you will changed;
NeedSave:used by internal.

Example:

```
MyCAD1.BringToFront(MyCAD1.GetSelectedShape);
```

See Also:

[SendToBack](#)
[BringToFrontByStep](#)

5.3.12 BringToFrontByStep

procedure BringToFrontByStep(AShape:TMyShape;NeedSave:boolean=true)

Description:

Sent the selected shape to front by a step

Parameter:

AShape:the shape that you will changed;
NeedSave:used by internal.

Example:

```
MyCAD1.BringToFront(MyCAD1.GetSelectedShape);
```

See Also:

[SendToBack](#)
[BringToFront](#)

5.3.13 ClearAllUndoStuff

procedure ClearAllUndoStuff

Description:

Clear all undo stuff in memory,it is be used at start new drawing.

Example:

```
MyCAD1.ClearAllUndoStuff;
```

See also:

[UndoRedoSize](#)

5.3.14 ClosePolygon

procedure ClosePolygon 

Description:

Finish TMyPolygon drawing.

Example:

```
MyCAD1.ClosePolygon;
```

5.3.15 Copy

procedure Copy();

Description:

Save selected shape to memory, and they also be saved in bitmap format in clipboard.

Example:

```
MyCAD1.SelectShapeBycode(0);  
//Select other shape  
MyCAD1.SelectShapeBycode(1,false);  
MyCAD1.Copy;  
MyCAD1.Paste;
```

See also:

[Paste](#)
[Cut](#)

5.3.16 CopyToClipboardAsWmf

procedure CopyToClipboardAsWmf

Description:

Save selected shape be saved in wmf format in clipboard.

Example:

```
MyCAD1.SelectShapeBycode(0);  
//Select other shape  
MyCAD1.SelectShapeBycode(1,false);  
MyCAD1.CopyToClipboardAsWmf;
```

See also:

[Paste](#)
[Cut](#)
[Copy](#)

5.3.17 Create

constructor Create(AOwner: TComponent);

Description:

Create TMyCAD; There are a layer be created,named "Layer0".

Example:

```
var
    MyCAD: TMyCAD;
begin
    MyCAD := TMyCAD.Create(Form1);
    MyCAD.Parent := Form1;
end;
```

See also:

[Destroy](#)

5.3.18 CreateLink

function CreateLink(ALinkShapeName:string;SrcShape:TMyShape; SrcLinkPtId:integer;
DestShape:TMyShape;DestLinkPtId:integer): Integer;

Description:

To create linkline shape between SrcShape and DestShape.

Parameter:

ALinkShapeName : The linkline shape's shapename;

SrcShape: The source shape,it is a userdefine shape, and has linkpoint in it;

SrcLinkPtId: The link point id.

DestShape: The destination shape,it is a userdefine shape, and has linkpoint in it;

DestLinkPtId: The link point id.

Return value:

-1: add failed

else(>=0) : add success, it is the LinkLinkshape's Shapeld.

See Also

[DeleteAllShapes](#)

5.3.19 Cut

procedure Cut();

Description:

Save selected shape to memory, delete selected shape, and they also saved in bitmap format in clipboard.

Example:

```
MyCAD1.SelectShapeBycode(0);  
MyCAD1.Cut;  
MyCAD1.Paste;
```

See also:

[Copy](#)
[Paste](#)

5.3.20 DeleteAllLayers

procedure DeleteAllLayers(): **Boolean**;

Description:

Delete all layer and delete all shapes, CurrentLayerId is -1.

Return value:

true: the all layers was deleted
false: deleted failed

See Also

[DeleteAllShapes](#)

5.3.21 DeleteAllShapes

procedure DeleteAllShapes;

Description:

Delete all shapes , Layer(s) is still exist.

See Also:

[DeleteAllLayers](#)

5.3.22 DeleteLayerByID

function DeleteLayerByID(ALayerID: integer): **Boolean**;

Description:

Delete Layer by the layer's Id , if ALayerId not matched , it returns false.
This function will delete shape(s) which belonged this layer.

See also:

[DeleteLayerByName](#)

5.3.23 DeleteLayerByName

function DeleteLayerByName(ALayerName: string): **Boolean**;

Description:

Delete Layer by the layer's name , if ALayerName not matched , it returns false.
This function will delete shape(s) which belonged this layer.

See also:

[DeleteLayerByID](#)

5.3.24 DeleteSelectedShape

function DeleteSelectedShape:boolean;

Description:

Delete current selected shape(s).

Return Value:

true : delete success
false : delete failed

5.3.25 DeleteShapeByID

function DeleteShapeByID(ShapeID:Integer):**Boolean**;

Description:

Delete a shape or a grouped shape, Shape's ID , start from zero ;when you delete one or more shapes, the shapeld will **not** be chaged.

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
begin
    DeleteShapeByID(1)
end;
```

Return Value:

true : delete success
false : delete failed

See Also:

[DeleteSelectedShape](#)
[DeleteAllShapes](#)

5.3.26 DeSelectedAllShapesByCode

procedure DeselectedAllShapesByCode

Description:

After execute this command, no shape will be in selected state.

Example:

```
MyCAD1.DeselectedAllShapesByCode;
```

See also:

[SelectShapeByCode](#)

5.3.27 Destroy

destructor Destroy;

Description:

Destroy the instance of TMyCAD, all layers,all shapes belonged it will be destoryed too;

See also:

[Create](#)

5.3.28 DrawAllShape

procedure DrawAllShape(MyCanvas:TCanvas;ARect:TRect);

Description:

Redraw all shapes.

Parameter:

MyCanvas : The canvas that you want draw on;

ARect: The region, it shape out of this region, it will not be draw.This is for speed drawing.

5.3.29 FlipHoriz

procedure FlipHoriz(AShape:TMyShape)



Description:

Flip a shape in horizontally .if AShape is a group shape,the childshapes that belonged it will be flipped.

See also:

[TMyShape.IsFlipVert](#)

5.3.30 FlipVert

procedure FlipVert(AShape:TMyShape)



Description:

Flip a shape in vertically . if AShape is a group shape,the childshapes that belonged it will be flipped.

See also:

[TMyShape.IsFlipHorz](#)

5.3.31 GetLayerIdByName

function GetLayerIdByName(ALayerName: string): **integer**;

Description:

Get layer's Id and it's name is ALayerName, if do not match ALayerName, return -1;

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
begin
  ShowMessage(' The Layers name is: '+ 'MyName' + ' its layer Id is: ' +Inttostr(MyCAD1.
  GetLayerIdByName('MyName')));
end;
```

See Also:

[GetLayerNameById](#)

[GetLayerNobyId](#)

5.3.32 GetLayerIdByNo

function GetLayerIdByNo(ALayerNo: integer): **integer**;

Description:

Get layer's Id by No, if do not match No, return -1;

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
var
  a:integer;
begin
  a:=GetLayerIdByNo(4);

  ShowMessage(' The Layers No is: '+ '4' + ' its layer Id is: ' +Inttostr(a));
end;
```

See Also:

[GetLayerNameById](#)

[GetLayerNobyId](#)

5.3.33 GetLayerNameById

function GetLayerNameById(ALayerId: integer): **string**;

Description:

Get layer's Name using it's Id , if do not match id, return "";

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
begin
  ShowMessage(' The Layers Id is: '+ '8' + ' Name is: ' +GetLayerNameById(8);
end;
```

See Also:

[GetLayerIdByName](#)

[GetLayerNobyId](#)

5.3.34 GetLayerNoById

function GetLayerNoById(ALayerId: integer): **integer**;

Description:

Get layer's No by Id, if do not match Id, return -1;

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
begin
    ShowMessage(' The Layers Id is:'+4 + ' its layer No is: '+Inttostr(MyCAD1.
    GetLayerNoById(4)));
end;
```

See Also:

[GetLayerIdByNo](#)

5.3.35 GetLayerNoByName

function GetLayerNoByName(ALayerName: string): **integer**;

Description:

Get layer's No and it's name is ALayerName, if do not match ALayerName, return -1;

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
begin
    with MyCAD1 do
        ShowMessage(' The Layers name is:'+MyName + ' layer No is:
        '+Inttostr(GetLayerNoByName(MyName)));
end;
```

See Also:

[GetLayerIdByNo](#)

[GetLayerNameById](#)

[GetLayerNobyId](#)

5.3.36 GetLayersCount

function GetLayersCount(): **integer**;

Description:

How many layers in TMyCAD.

Sample:

```
procedure Form1.Button1Click(Sender:TObject)
begin
    ShowMessage(' There are '+Inttostr(MyCAD1.GetLayersCount)+' in TMyCAD1!');
end;
```

See Also:

[GetShapesCount](#)
[GetShapesCountInALayer](#)

5.3.37 GetMaxLayerId

function GetMaxLayerId(): **integer**;

Description:

Get max layer's Id, It is **not** the count of layers,if there is no layer existed,it return -1.

Example:

```
procedure Form1.Button1Click(Sender:TObject)
begin
    ShowMessage(' The max layer Id is '+Inttostr(MyCAD1.GetMaxLayerID));
end;
```

See Also:

[GetLayersCount](#)

5.3.38 GetMemShapesCount

function GetMemShapesCount: integer;

Description:

Get the count of memorize shapes in undo list.

Example:

```
EditPastel.Enabled:= MyCAD1.GetMemShapesCount > 0;
```

See also:

[GetShapesCount](#)

5.3.39 GetSelectedShape

function GetSelectedShape: TMyShape;

Description:

Get the selected shape.

Return Value:

When more than two shapes selected , return nil, When a grouped shape selected, return the first shape

Example:

```
Shape1:= MyCAD1.GetSelectedShape;
```

See Also:

[GetSelectedShapes](#)

5.3.40 GetSelectedShapes

function GetSelectedShapes: TArrMyShape;

Description:

Get the selected shapes.

Return Value:

If select more than two shapes selected ,Please use this function to get seleceted shape.

Example:

```
Shape1:= MyCAD1.GetSelectedShapes[0];
```

See Also:

[GetSelectedShape](#)

5.3.41 GetSelectedShapesCount

function GetSelectedShapesCount:integer;

Description:

Get how many shape were selected, a grouped shape is one shape .

See also:

[GetSelectedShape](#)

[GetSelectedShapes](#)

5.3.42 GetShapebyID

function GetShapeById(Id:integer): TMyShape;

Description:

Know a shape's ID , Get the shape.

Return Value:

if Id is not exist, return nil;

Example:

```
Shape1:= MyCAD1.GetShapeById(0);
```

See Also:

[GetShapeByName](#)

[GetShapeByNo](#)

5.3.43 GetShapeByName

```
function GetShapeByName(const AName:String): TMyShape;
```

Description:

Know a shape's name , Get the shape.

Return Value:

if there is no shape named AName , return nil;

Example:

```
Shape1:= GetShapeByName( 'Shape100' );
```

See also:

[GetShapeById](#)

5.3.44 GetShapebyNo

```
function GetShapeByNo(AShapeNo:integer): TMyShape;
```

Description:

Know a shape's No , Get the shape.

Return Value:

if AShapeNo is not exist, return nil;

Example:

```
Shape1:= MyCAD1.GetShapeByNo(10);
```

See Also:

[GetShapeByName](#)

[GetShapeById](#)

5.3.45 GetShapeNoById

function GetShapeNoById(AShapeId: Cardinal): Integer;

Description:

Know a shape's ID , Get the shape no.

Return Value:

if Id is not exist, return -1;

Example:

```
ANo := MyCAD1.GetShapeNoById(0);
```

See Also:

[GetShapeByName](#)

[GetShapeByNo](#)

5.3.46 GetShapesCount

function GetShapesCount: integer;

Description:

Get the count of shapes that you have added .

Example:

```
ShowMessage('There are ' + IntToStr(MyCAD1.GetShapesCount)+ ' in your form!')
```

See also:

[GetShapesCountInALayer](#)

5.3.47 GetShapesCountInALayer

function GetShapesCountInALayer(ALayerId: integer): **integer**;

Description:

get the amount of the shapes on a layerId;

See also:

[GetShapesCount](#)

5.3.48 GroupWorkingShape

function GroupWorkingShape: Integer;

Description:

Group one or more selected shape.

Return Value:

-1: Group shape failed
else:The ShapeID

Example:

```
MyCAD1.GroupWorkingShape;
```

See Also:

[UnGroupShape](#)

5.3.49 InVisibleLayerByld

procedure InVisibleLayerByld(LayerId: integer);

Description:

set the layer hide.

See also:

[InVisibleLayerByName](#)

5.3.50 InVisibleLayerByName

procedure InVisibleLayerByName(LayerName: string);

Description:

set the layer hide.

See also:

[InVisibleLayerByld](#)

5.3.51 IsLinked

function IsLinked(AShape,BShape:TMyShape): Integer;

Description:

To sure there is linking relationshape between AShape and BShape. it there is more than one linking relationship between , it is onyl return the first.

Return Value:

-1:there is no link
>=0 : it is linkline shape id .

5.3.52 IsTCADFile

function IsTCADFile(FileName:String):Boolean;

Description:

Determine a file is in TCAD file format or not .

Example:

```
if MyCAD1.IsTCADFile( OpenFileDialog1.FileName) then
begin
    if not MyCAD1.LoadFromFile(OpenDialog1.FileName) then
        ShowMessage('Error when read file');
    end
else
    ShowMessage(OpenDialog1.FileName + 'is not a TCAD format file');
end;
```

5.3.53 IsVisibleLayerByID

function IsVisibleLayerByID(LayerId: integer): **Boolean**;

Description:

Is the layer is visible or not .

Return Value:

true: it is visible
false:it is invisible or LayerId is not existed.

5.3.54 LoadFromFile

function LoadFromFile(FileName:String):Boolean;

Description:

Load from a file.

Example:

```
MyCAD1.LoadFromFile('c:\SavedFile.tcad');
```

See also:

[SavetoFile](#)

5.3.55 LoadFromStream

function LoadFromStream(Stream:TStream):Boolean;

Description:

Load from the stream, you can read from the database's field or TCP/IP stream.

Example:

```
MyCAD1.LoadFromStream(myStream);
```

See also:

[SaveToStream](#)

5.3.56 LockUnlockforShapes

procedure LockUnlockforShapes(AShape:TMyShape,Value:boolean);

Description:

Prevent the shape or the grouped shape be moved or resized by mouse, but can moved by code.

After execute this command, AShape.Locked is true and disappear the hotspot when you selected this shape.

Parameter:

AShape:a shape;

Vaule: true,lock it;

false,unlock it.

5.3.57 MergeLayers

function MergeLayers(LayerId1,LayerId2:integer): Boolean;

Description:

Merge Layer2 to Layer1 and remove the layer2, CurrentLayerId is Layer1.

Return :

false: failed

true: merge success

Example:

```
if MyCAD1.NewLayer (1,2) then  
  ShowMessage('Merge ok!');
```

See also:

[CurrentLayerId](#)

5.3.58 Move

procedure Move(AShape:TMyShape;Dx,Dy:integer);

Description:

move a shape or a grouped shape by code.OnShapeDragging event tiggered.

Example:

```
AShape:= MyCAD1.GetSelectedShape;  
if AShape <> nil then  
  MyCAD1.Move(AShape,20,40);
```

See Also:

[OnShapeDragging](#)

5.3.59 NewLayer

function NewLayer(ALayerName: string; aVisible: Boolean = true): integer;

Description:

Add a layer for TMyCAD, return new layerid, CurrentLayerid is new layerid also.

Return :

-1: failed

else: return LayerId.

Example:


```
if MyCAD1.NewLayer (' mapLayer')> -1 then  
  ShowMessage('Add ok');
```

See also:

[CurrentLayerId](#)

5.3.60 Paste

procedure Paste();

Description:

Paste saved shapes to TMyCAD, they have 4 pixels offset .

Example:

```
MyCAD1.SelectShapeBycode(0);  
MyCAD1.Cut;  
MyCAD1.Paste;
```

See also:

[Copy](#)

[Cut](#)

5.3.61 PopfromUndoRedoShapeList

procedure PopfromUndoRedoShapeList

Description:

Undo one step .

Example:

```
MyCAD1.PopfromUndoRedoShapeList;
```

See also:

[UndoRedoSize](#)

5.3.62 Print

procedure Print(ALayers: array of Integer; UserScale: double = 1.0);

Description:

Print to printer.

Parameter

ALayer: if you want print all layer, please use [], else [1,2,3], or you create array and set layerId that you want print

UserScale:the scale.

Example:**Delphi**

```
//print all shapes in 50% scale.  
MyCAD1.Print ([],0.5);
```

C++ Builder

```
//print all shapes in 100% scale.  
void __fastcall TMainFrm::BitBtn1Click(TObject *Sender)  
{  
    int p[1];  
    p[0]=NULL;  
    MyCAD1->Print(p,0,1.0);  
}  
  
//print the shapes in 0,1 layers in 100% scale.  
void __fastcall TMainFrm::BitBtn1Click(TObject *Sender)  
{  
    int p[2];  
    p[0]=0; //First Layer Id  
    p[1]=1; //Second Layer Id  
    MyCAD1->Print(p,2,1.0);  
}
```

See also:

[PrintPreview](#)

5.3.63 PrintPreview

procedure PrintPreview(ALayers: array OF Integer; ABitmap:TBitmap; ScaleforPreview:double = 1.0);

Description:

Preview the drawing on the Bitmap.

Parameter

ALayer: if you want print all layer, please use [], else [1,2,3], or you create array and set layerId that you want print

ABitmap: that you draw on.

ScaleforPreview:the scale.

Example:

Delphi:

```
//print preview all shapes in 100% scale.
MyCAD1.PrintPreview ([],Image1.Picture.Bitmap,1.0);

int p[1];
p[0]=0;
MyCAD1->PrintPreview(p,1,Form1->Image1->Picture->Bitmap,1);
Form1->Show();
}
//-----
```

C++ Builder:

```
//print preview all shapes in 100% scale.
void __fastcall TMainFrm::BitBtn1Click(TObject *Sender)
{
    int p[1];
    p[0]=NULL;
    MyCAD1->PrintPreview(p,0,Image1->Picture->Bitmap,1.0);
}
//print preview the shapes in 0,1 layers in 100% scale.
void __fastcall TMainFrm::BitBtn1Click(TObject *Sender)
{
    int p[2];
    p[0]=0; //First Layer Id
    p[1]=1; //Second Layer Id
    MyCAD1->Print(p,2,Image1->Picture->Bitmap,1.0);
}
```

See also:

[Print](#)

5.3.64 RenameShapename

procedure RenameShapeName(const OldName,NewName:String);

Description:

Know a shape's name , Set the shape new name.

Example:

```
MyCAD1.RenameShapeName('Shape100','MyNewShape');
```

5.3.65 Rotate

procedure Rotate (AShape:TMyShape;AAngle:Single);

Description:

Rotate a shape or a combine shape by code. AAngle in radians. This command is more useful for automatic processing.

Example:

This example can rotate shape 30 degree.

```
AShape:= MyCAD1.GetSelectedShape;  
if AShape <> nil then  
    MyCAD1.Rotate(AShape, 30*pi/180);
```

5.3.66 SaveToBmp

procedure SaveToBmp(BmpFileName:String);

Description:

Save TMyCAD drawing to a disk file , it is in BMP format .

Example:

```
MyCAD1.SavetoBMP('c:\SavedFile.bmp');
```

See also:

[LoadFromFile](#)
[SavetoStream](#)
[SaveToWMF](#)
[SaveToJPG](#)

5.3.67 SaveToBmp-2

procedure SaveToBmp(Bmp:TBitmap; NewWidth, NewHeight: integer); overload;

Description:

Save TMyCAD drawing to a disk file in new ratio, it is in BMP format . if NewWidth/ Width > NewHeight / Height , then the zoom according NewWidth/ Width, else according NewHeight / Height.

Example:

```
MyCAD1.SavetoBMP('c:\SavedFile.bmp',100,200);
```

See also:

[LoadFromFile](#)
[SavetoStream](#)
[SaveToWMF](#)
[SaveToJPG](#)

5.3.68 SaveToDxf

procedure SaveToDXF(DXFFileName:String);

Description:

Save CAD drawing to a disk file , it is in AutoCAD R12 DXF format .

Example:

```
CAD1.SavetoDXF('c:\SavedFile.dxf');
```

See also:

[LoadFromFile](#)
[SavetoStream](#)
[SaveToBmp](#)
[SaveToJPG](#)

5.3.69 SaveToFile

function SaveToFile(FileName:String):Boolean;

Description:

Save TMyCAD drawing content to a disk file, it is in TCADformat .

Return value:

true: the drawing saved to the file.
false: the drawing not saved.

Example:

```
CAD1.SaveToFile('c:\SavedFile');
```

See also:

[LoadFromFile](#)
[SaveToStream](#)

5.3.70 SaveToJpg

procedure SaveToJpeg(JpegFileName:String;Quality:integer=80)

Description:

Save CAD drawing to a disk file , it is in JPEG format , set parameter Quality can change the quality of the image,default is 80%.

Example:

```
CAD1.SaveToJPG('c:\SavedFile.jpg');
```

See also:

[LoadFromFile](#)
[SaveToStream](#)
[SaveToBmp](#)
[SaveToWMF](#)

5.3.71 SaveToStream

function SaveToStream(Stream:TStream):Boolean;

Description:

Save to the stream, you can save all content into the database's field or memory stream.

Example:

```
MyCAD1.SaveToStream(myStream);
```

See also:

[LoadFromStream](#)
[SaveToFile](#)

5.3.72 SaveToWmf

procedure SaveToWmf(WMFFileName:String);

Description:

Save CAD drawing to a disk file , it is in WMF format .

Example:

```
CAD1.SavetoWMF('c:\SavedFile.WMF');
```

See also:

[LoadFromFile](#)
[SavetoStream](#)
[SaveToBmp](#)
[SaveToJPG](#)

5.3.73 SelectAllShapes

procedure SelectAllShapes;

Description:

Select all shapes

Notes:

if the shape is grouped with other shapes, the hotspot for that group appear.

Examples:

```
MyCAD1.SelectAllShapes;
```

See also:

[SelectAllShapesByLayerId](#)
[SelectShapeByCode](#)

5.3.74 SelectAllShapesByLayerId

procedure SelectAllShapesByLayerId(const ALayerId:integer);



Description:

Specifies the layer id to select all shapes in this layer.

Notes:

if the shape is grouped with other shapes, the hotspot for that group appear.

Examples:

```
MyCAD1.SelectAllShapesByLayerId(0);
```

See also:

[SelectAllShapes](#)
[SelectShapeByCode](#)

5.3.75 SelectShapeByCode

function SelectShapeByCode(ShapId:integer;RemovePrevSelectedShape:boolean=true):
Boolean;

Description:

Select a shape by code like mouse actions. Use this function if, for example, you want to perform operations on a certain shape, which require the use of GetSelectedShapId, if RemovePrevSelectedShape is true, do not clear last selected shape hot spot.

Return value:

true: the shape is selected
false: there is no shape with this Id,

Notes:

if the shape is grouped with other shapes, the hotspot for that group appear.
This function can effect the GetSelectedShapId

Examples:

```
MyCAD1.SelectShapeByCode(2);
```

See also:

[SelectAllShapes](#)
[SelectAllShapesByLayerId](#)

5.3.76 SendtoBack

procedure SendToBack(AShape:TMyShape;NeedSave:Boolean=true);

Description:

Sent the selected shape to background .

Parameter:

AShape:the shape that you will changed;
NeedSave:used by internal.

Example:

```
MyCAD1.SendToBack(MyCAD1.GetSelectedShape);
```

See Also:

[BringToFrontByStep](#)
[BringToFront](#)

5.3.77 SendToBackByStep

procedure SendToBackByStep(AShape:TMyShape;NeedSave:boolean=true)

Description:

Sent the selected shape to background by a step

Parameter:

AShape:the shape that you will changed;
NeedSave:used by internal.

Example:

```
MyCAD1.SendToBackByStep(MyCAD1.GetSelectedShape);
```

See Also:

[SendToBack](#)
[BringToFront](#)

5.3.78 SetLayerNameById

procedure SetLayerNameById(const NewLayerName: string; ALayerID: integer);

Description:

Change the layer's name by layer's id.

5.3.79 SetLayerNameByName

procedure SetLayerNameByName(const NewLayerName, OldLayerName: string);

Description:

Change the layer's name by old layer's name.

5.3.80 SetMyImage

function SetMyImage(ShapeID: integer; ABitmap: TBitmap): **Boolean**;

Description:

Add a bitmap to TMyImage's shape, it is important.

Example:

```
if AShape.ClassName = 'TMyImage' then
begin
  if OpenPictureDialog1.Execute then
  begin
    myBitmap := TBitmap.Create;
    myBitmap.LoadFromFile(OpenPictureDialog1.FileName);
    if MyCAD1.SetMyImage(ShapeId, MyBitmap) then
      Memol.Lines.Add('Bitmap be loaded into Shape' + Inttostr(ShapeID))
    else
      Memol.Lines.Add('Bitmap NOT be loaded into Shape' + Inttostr(ShapeID));
    MyBitmap.Free;
  end;
end;
```

Return value:

true: add image success

false:add image failed.

5.3.81 SizeShape

procedure SizeShape(AShape:TMyShape;ASelectedHotId:integer;AMovPt:TMyPoint);

Description:

Size a shape or a grouped shape, it is like mouse action of resizing a shape.

Parameter:

AShape: shape that you will size it.

ASelectedHotId: it is hotspot id.

AMovPt: the destination point;

Example:

```
MyCAD1.SizeShape(MyCAD1.GetSelectedShape,0,MyPoint(100,100));
```

See also:

[Move](#)

5.3.82 UngroupShape

procedure UngroupShape(AShape: TMyShape;NeedSaved:boolean= true);

Description:

unGroup have grouped shapes.

Parameter:

AShape: shape that you will ungroup it.

NeedSaved: it is for undo procedure, normal is true, do not change it, it is used by internal.

Example:

```
MyCAD1.UngroupShape(MyCAD1.GetSelectedShape);
```

See also:

[GroupWorkingshape](#)

5.3.83 VisibleAllLayer

`procedure VisibleAllLayer;`

Description:

Make all layer(s) visible

See Also:

[VisibleLayerById](#)
[VisibleLayerByName](#)
[InVisibleLayerById](#)
[InVisibleLayerByName](#)

5.3.84 VisibleLayerById

`procedure VisibleLayerById(LayerID: integer);`

Description:

Make a layer visible by layer's id.

See Also:

[VisibleAllLayer](#)
[VisibleLayerByName](#)
[InVisibleLayerById](#)
[InVisibleLayerByName](#)

5.3.85 VisibleLayerByName

`procedure VisibleLayerByName(LayerName: string);`

Description:

Make a layer visible by layer's name.

See Also:

[VisibleAllLayer](#)
[VisibleLayerById](#)
[InVisibleLayerById](#)
[InVisibleLayerByName](#)

5.4 Properties

5.4.1 ArrowAngle

property ArrowAngle:integer

Description:

Set the Line and PolyLine 's arrow angle. value is between: 0 - 359

Example:

```
MyCAD1.ArrowAngle := 180 ;
```

See also:

[ArrowOffset](#)

[ArrowStyle](#)

5.4.2 ArrowLength

property ArrowLength:Byte

Description:

Set the Line and PolyLine 's arrow Length. value is between: 10-50

See also:

[ArrowAngle](#)

[ArrowOffset](#)

[ArrowStyle](#)

5.4.3 ArrowOffset

property ArrowOffset:byte

Description:

When a Line shape is drawn with an arrow, the Arrowoffset specifies how many pixels from the end of the line the arrow is drawn.

value is between: 0 - 255 , default is 0.

Example:

```
MyCAD1.ArrowOffset :=0;
```



```
MyCAD1.ArrowOffset :=16;
```



See also:

[ArrowAngle](#)

[ArrowStyle](#)

5.4.4 ArrowStyle

property ArrowStyle: [TArrowStyle](#)

Description:

Set the Line and PolyLine 's arrow style;

ANone: it is a line;

ALeft: one arrow at the left end of the line or polyline;

ARight: one arrow at the right end of the line or polyline;

ADouble: a double-arrow line or polyline

See also:

[ArrowAngle](#)

[ArrowOffset](#)

5.4.5 BkBitmap

property BkBitmap: TBitmap

Description:

Set the background bitmap for TMyCAD, for clear it , BkBtmap:=nil;

Example:

The example show assign a bitmap that you load to MyCAD1.

```
var
  mybitmap:TBitmap;
begin
  if OpenPictureDialog1.Execute then
    begin
      myBitmap:=TBitmap.Create;
      mybitmap.LoadFromFile(OpenPictureDialog1.FileName);
      MyCAD1.BkBitmap:=mybitmap;
      MyBitmap.Free;
    end;
end;
```

See also:

[BkBitmapMode](#)

5.4.6 BkBitmapMode

property BkBitmapMode:TBkBitmapMode

TBkBitmapMode = (Tiled,Stretch,Center,LeftTop);

Description:

Set the background bitmap show style.

Tiled : if the bitmap size low than TMyCAD's size , two or more this bitmap drawn on TMyCAD

Stretch:Bitmap be stretch to fit at TMyCAD

Center: Bitmap is in center of TMyCAD

LeftTop:Bitmap be showed from TMyCAD's Left,Top

See Also:

[TBkBitmapMode](#)

5.4.7 Brush

property Brush:TBrush;

Description:

Set the brush of TMyCAD that you need;

5.4.8 Canvas

property Canvas:TCanvas;

Description:

Canvas allows drawing on the TMyCAD by providing a TCanvas object for this purpose. Any canvas operation is valid on TMyCAD, and draw operation doesn't change TMyShapes object, the content will be clear when paint event occurred.A canvas object is created automatically for the TMyCAD and the property is read-only.

See also:

[OnPaint](#)

5.4.9 ColorOfBackground

property ColorOfBackground:TColor

Description:

specify the background color .

Example:

```
MyCAD1.ColorOfBackground := clBule;
```

See Also:

[ColorOfHot](#)

5.4.10 ColorOfHot

property ColorOfHot:TColor

Description:

Specify the color of hot square;

Example:

```
MyCAD1.ColorofHot:=clRed;
```

See also:

[ColorofBackground](#)

5.4.11 CrossLine

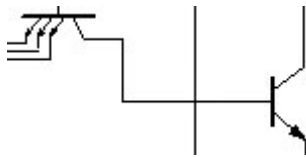
property CrossLine:Boolean;

Description:

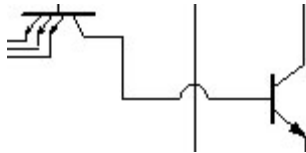
It is used for TMyLinkLine drawing feature, when it is true,the drawing speed lower than false;

Example:

```
MyCAD1.CrossLine:=false;
```



```
MyCAD1.CrossLine:=true;
```



See also:

TMyLinkLine

5.4.12 CurrentLayerId

property CurrentLayerId :integer;

Description:

Get the current layer's ID, it is start from 0; New shape(s) will belong to this layer.

Example:

```
MyCAD1.CurrentLayerID:=2;  
ShowMessage (IntToStr('You are drawing on layerID:'+MyCAD1.CurrentLayerID));  
MyCAD1.CurrentLayerID:=3;  
ShowMessage (IntToStr('You are drawing on layerID:'+MyCAD1.CurrentLayerID));  
// if there is no layerid is 3,MyCAD1.CurrentLayerID still equal 2
```

See also:

[NewLayer](#)

5.4.13 DiskFileVersion

It is readonly field,for TMyCAD version compatibility.

5.4.14 DragMode

property DragMode: [TDragMode](#)

Description:

It can help you drag shape, when value is dgHorz then the shape drag in horizontal when value is dgVert then the shape drag in vertical.

Example:

```
MyCAD1.DragMode:=dgHorz;
```

5.4.15 Enable

property Enabled: Boolean;

Description

Whether the TCAD control responds to mouse, keyboard, and timer events. but still effected by code action.

Use Enabled to change the availability of the TMyCAD to the user. To disable a control, set Enabled to false. Disabled TMyCAD appear dimmed. If Enabled is false, the TMyCAD ignores mouse, keyboard, and timer events.

To re-enable a control, set Enabled to true. The TMyCAD is no longer dimmed, and the user can use the TMyCAD.

See also

[Visible](#)

5.4.16 Font

property Font: TFont;

Description:

Set the font.

5.4.17 GridOperation

property GridOperation: TGridOperation 

Description:

set the grid of TMyCAD.

5.4.18 HotShow

property HotShow:boolean;

Description:

true: the hotspot showed normally
false: the hotspot hide, even a shape be selected.

Example:

Delphi syntax:

```
MyCAD1.HotShow := true;
```

C++ syntax:

```
MyCAD1->HotShow = true;
```

5.4.19 HotSize

property HotSize:byte

Description:

can set the hotspot big or small.

Delphi syntax:

```
MyCAD1.HotSize := 6;
```

C++ syntax:

```
MyCAD1->HotSize = 6;
```

5.4.20 LabelValue

property LabelValue:TLabel

Description:

Show the parameter about the current shape. It can show the length of a TMyLine or height and width of a TMyRectangle

Example:

```
MyCAD1.LabelValue:=Form1.Label2;
```

See also:

[LabelXY](#)

5.4.21 LabelXY

property LabelXY:TLabel

Description:

Show current mouse cursor 's coordinate, it is changed by mouse moving;

Example:

Delphi syntax:

```
MyCAD.LabelXy:=Form1.Label1;
```

C++ syntax:

```
MyCAD->LabelXy =Form1->Label1;
```

See also:

[LabelValue](#)

5.4.22 LinkLineDrawStyle

property LinkLineDrawStyle:TLinkLineDrawStyle

Description:

define linkline style.

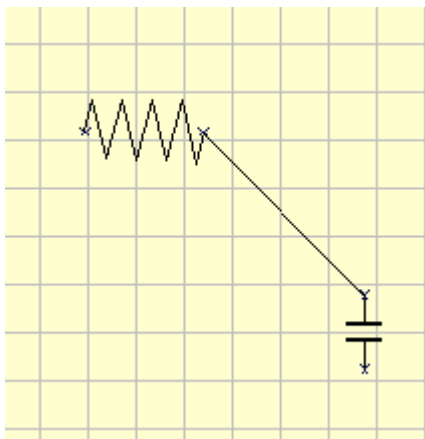
Example:

Delphi syntax:

```
MyCAD1.LinkLineDrawStyle:=lldsFree;
```

C++ syntax:

```
MyCAD1->LinkLineDrawStyle =lldsFree;
```

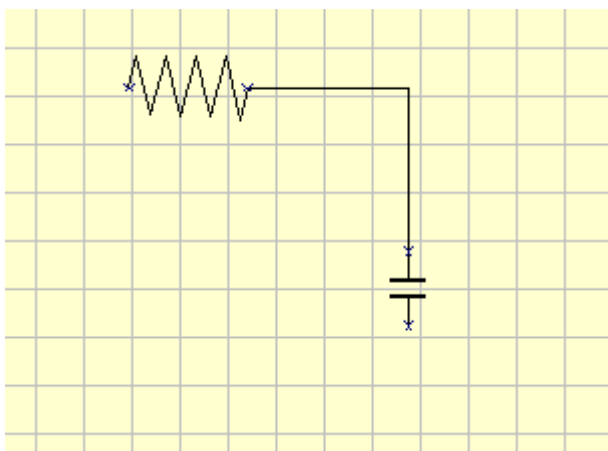


Delphi syntax:

```
MyCAD1.LinkLineDrawStyle:=lldsHV;
```

C++ syntax:

```
MyCAD1->LinkLineDrawStyle =lldsHV;
```



5.4.23 OperateAllLayer

property OperateAllLayer:boolean;

Description:

Is you want mouse action effects only to current layer or all layers. default is true;

Example:

```
MyCAD1.OperateAllLayer :=true;
```

5.4.24 PageFoot

property PageFoot:string;

Description:

Set the printing page foot.

Example:

Delphi syntax:

```
MyCAD1.PageFoot := 'Designed by hongbin.fei, 2004.12';
```

C++ syntax:

```
MyCAD1->PageFoot = "Designed by hongbin.fei, 2004.12";
```

See also:

[PageHead](#)

5.4.25 PageFootAlignment

property PageFootAlignment: TAlignment;

Description:

Determines how the text is aligned within the page foot.

Use Alignment to change the way the text is formatted by the TMyCAD control. Alignment can take one of the following values:

Value Meaning

taLeftJustify	Text is left-justified: Lines all begin at the left edge of the control.
taCenter	Text is centered in the control.
taRightJustify	Text is right-justified: Lines all end at the right edge of the control.

See also:

[PageHeadAlignment](#)

5.4.26 PageFootFont

property PageFootFont:TCADFont;

Description:

Set the font of page foot.

5.4.27 PageFootToBottom

property PageFootToBottom: integer;

Description:

The space of page foot to bottom, it is in pixel unit.

See also:

[PageHeadToTop](#)

5.4.28 PageHead

property PageHead:string;

Description:

Set the page title.

Example:

Delphi syntax:

```
MyCAD1.PageHead := 'TCAD magic drawing';
```

C++ syntax:

```
MyCAD1->PageHead = "TCAD magic drawing";
```

See also:

[PageFoot](#)

5.4.29 PageHeadAlignment

property PageHeadAlignment: TAlignment;

Description:

Determines how the text is aligned within the page head.

Use Alignment to change the way the text is formatted by the TCAD control. Alignment can take one of the following values:

Value Meaning

taLeftJustify	Text is left-justified: Lines all begin at the left edge of the control.
taCenter	Text is centered in the control.
taRightJustify	Text is right-justified: Lines all end at the right edge of the control.

See also:

[PageFootAlignment](#)

5.4.30 PageHeadFont

property PageHeadFont:TCADFont;

Description:

Set the font style of page head.

5.4.31 PageHeadToTop

property PageHeadToTop: integer;

Description:

The space of page title to top, it is in pixel unit.

See also:

[PageFootToBottom](#)

5.4.32 PageHeight

property PageHeight : Cardinal

Description:

Set the Canvas(Page)'s Height, it is in unit cm. If you change the PageHeight ,PageStyle is changed automatic.

Example:

```
MyCAD1.PageHeight := 210;
```

See also:

[PageStyle](#)
[PageWidth](#)

5.4.33 PageOrientation

property PageOrientation:TPrinterOrientation ;

Description:

The Canvas's direction is Portrait or Landscape.then PageWidth and PageHeight will be changed(swaped) unless if PageStyle is CustomerPage.

Example:

```
MyCAD1.PageOrientation := poLandscape;
```

See also:

[PageStyle](#)

5.4.34 PageStyle

property PageStyle: TPageStyle;

Description:

Set the Canvas(Page)'s style,(A0,A1,A2,A3,A4,A5,B3,B4,B5,CustomerPage), then PageWidth and PageHeight will be changed.

Example:

```
MyCAD1.PageStyle:= A4;
```

See also:

[Defines](#)**5.4.35 PageWidth**

property PageWidth :Cardinal

Description:

Set the Canvas(Page)'s Width, it is in unit cm. If you change the PageStyle,then it is changed automatic.

Example:

```
MyCAD1.PageWidth := 120;
```

See also:[PageStyle](#)[PageHeight](#)**5.4.36 Pen**

property Pen:TPen

Description:

Set the pen that you need;

5.4.37 PrintABorder

property PrintABorder:Boolean ;

Description:

Is a border when printpreview or print.

See Also:[PrintBackground](#)**5.4.38 PrintABordertoBottom**

property PrintABordertoBottom: integer;

Description:

The space of the border to bottom, it is in pixel unit.

See also:

[PrintABordertoLeft](#)
[PrintABordertoTop](#)
[PrintABordertoRight](#)

5.4.39 PrintABordertoLeft

property PrintABordertoLeft: integer;

Description:

The space of the border to left, it is in pixel unit.

See also:

[PrintABordertoTop](#)
[PrintABordertoBottom](#)
[PrintABordertoRight](#)

5.4.40 PrintABordertoRight

property PrintABordertoRight: integer;

Description:

The space of the border to right, it is in pixel unit.

See also:

[PrintABordertoLeft](#)
[PrintABordertoTop](#)
[PrintABordertoBottom](#)

5.4.41 PrintABordertoTop

property PrintABordertoTop: integer;

Description:

The space of the border to top, it is in pixel unit.

See also:

[PrintABordertoLeft](#)
[PrintABordertoBottom](#)
[PrintABordertoRight](#)

5.4.42 PrintBackground

property PrintBackground:Boolean ;

Description:

Print canvas's background,if true,print will like the screen.

Example:

```
MyCAD1.PrintBackground:=false;
```

See Also:

[PrintABorder](#)

5.4.43 Ratio

property Ratio:double

Description:

Define the ratio , it is very useful,It will be appear at property [LabelValue](#), that is Line's Length and area for TMyLine,TMyRectangle,TMyEllisple oth other shape.

Example:

```
MyCAD1.Ratio:=100; // 1:100
```

See also:

[TheUnit](#)

5.4.44 ResizeEnable

property ResizeEnable:boolean;

Description:

Can resizing shape or not, it is very useful for the case of don't allow resize shape

See also:

[RotateEnable](#)

5.4.45 ReturnToSelecting

property ReturnToSelecting: **Boolean**

Description:

if it is true, mean The ShapeTool will return to [selecting] status automatically after you drew a shape, trigger a event OnActionToolToSelecting, else, ReturnToSelecting is false , the ShapeTool still on draw status, You can draw same shapes consecutively.

Example:

```
MyCAD1.ReturnToSelecting:=false;
```

You can draw same shapes consecutively.

See Also:

[OnActionToolToSelecting](#)

5.4.46 RotateConstraintDegree

property RotateConstraintDegree:Integer;

xp.a

Description:

Set the constraint degree when rotating,if value as zero,it can rotate freely.

5.4.47 RotateEnable

property RotateEnable:boolean;

Description:

Can rotating shape or not, it is very useful for the case of don't allow rotate shape

See also:

[ResizeEnable](#)

5.4.48 ShapeTool

property ShapeTool: [TDrawTool](#)

Description:

Set the current's Tool, you can change it when you need.

Example:

```
// Set Line draw tool
MyCAD1.ShapeTool:=SpLine;

// Set Link Line draw tool
MyCAD1.ShapeTool:=SpLinkLine;
// this code only allowed under TCADxp-PRO or TCADxp-ENT
```

Note:

If set shapetool =spClose , to close TCAD's draw,drag,resize,rotate fucntion;

See also:

[Defines](#)

5.4.49 ShowHint

property ShowHint: Boolean;

Description:

Determines whether the TMyCAD displays a Shape Caption when the mouse pointer rests momentarily on it.

Example:

Delphi syntax:

```
MyCAD1.ShowHint := true ;
```

C++ syntax:

```
MyCAD1->ShowHint = true ;
```

5.4.50 ShowHotLink

property ShowHotLink: Boolean;

Description:

Whether show the TMyCAD displays hot link point for a shape.

Example:

Delphi syntax:

```
MyCAD1.ShowHotLink := true ;
```

C++ syntax:

```
MyCAD1->ShowHotlink = true ;
```

5.4.51 Snap

property Snap: Boolean ;

Description:

Snap mouse to grid or not, help you drawing .

Example:

```
MyCAD1.Snap := true;
```

See also:

[SnapPixels](#)

[GridWidth](#)

[GridHeight](#)

5.4.52 SnapPixels

property SnapPixels: Byte

Description:

The mouse point will be capture to the grid when the pixels low than this value between mouse point and nearest grid .It can not big than gridwidth or gridheight.

Example:

```
MyCAD1.SnapPixels := 3 ;
```

See also:

[Snap](#)

[GridWidth](#)

[GridHeight](#)

5.4.53 SnapShape

property SnapShape:Boolean ;

Description:

When Drag or Resize shape, whether snap mouse to other shape , it can help you drawing .

Example:

```
MyCAD1.SnapShape := true;
```

See also:

[SnapPixels](#)

5.4.54 TheUnit

property TheUnit:[TUnits](#)

Description:

Which unit do you use, It will be appear at property [LabelValue](#), that is Line's Length or area for TMyLine,TMyRectangle,TMyEllisple...

Example:

```
MyCAD1.TheUnit := inch
```

5.4.55 UndoRedoSize

property UndoRedoSize:byte

Description:

Set the size (steps) of undo action saving. It costs system resource.

Example:

```
MyCAD1.Zoom:= 0.50 ;
```

See also

[PopfromUndoRedoShapeList](#)

5.4.56 Version

property Version: string;

Description:

It shows the version of TMyCAD, it is read only.

Example:

```
ShowMessage( 'Installed TMyCAD version is:'+MyCAD1.Version);
```

5.4.57 Visible

property Visible: Boolean;

Description

Determines whether the TMyCAD appears on screen.

Use the Visible property to control the visibility of the control at runtime. If Visible is true, the control appears. If Visible is false, the control is not visible.

Calling the Show method sets the control's Visible property to true. Calling the Hide method sets it to false.

See also

[Enable](#)

5.4.58 XYMode

property XYMode

Description:

There 4 mode to choose, to fit your need.

Please read [TXyMode](#) in defines .

5.4.59 Zoom

property Zoom:Double

Description:

Set the zoom value,the width,height, shape and background will resize automatically.When Zoom is 1 , TMyCAD is showed in 100%.

Example:

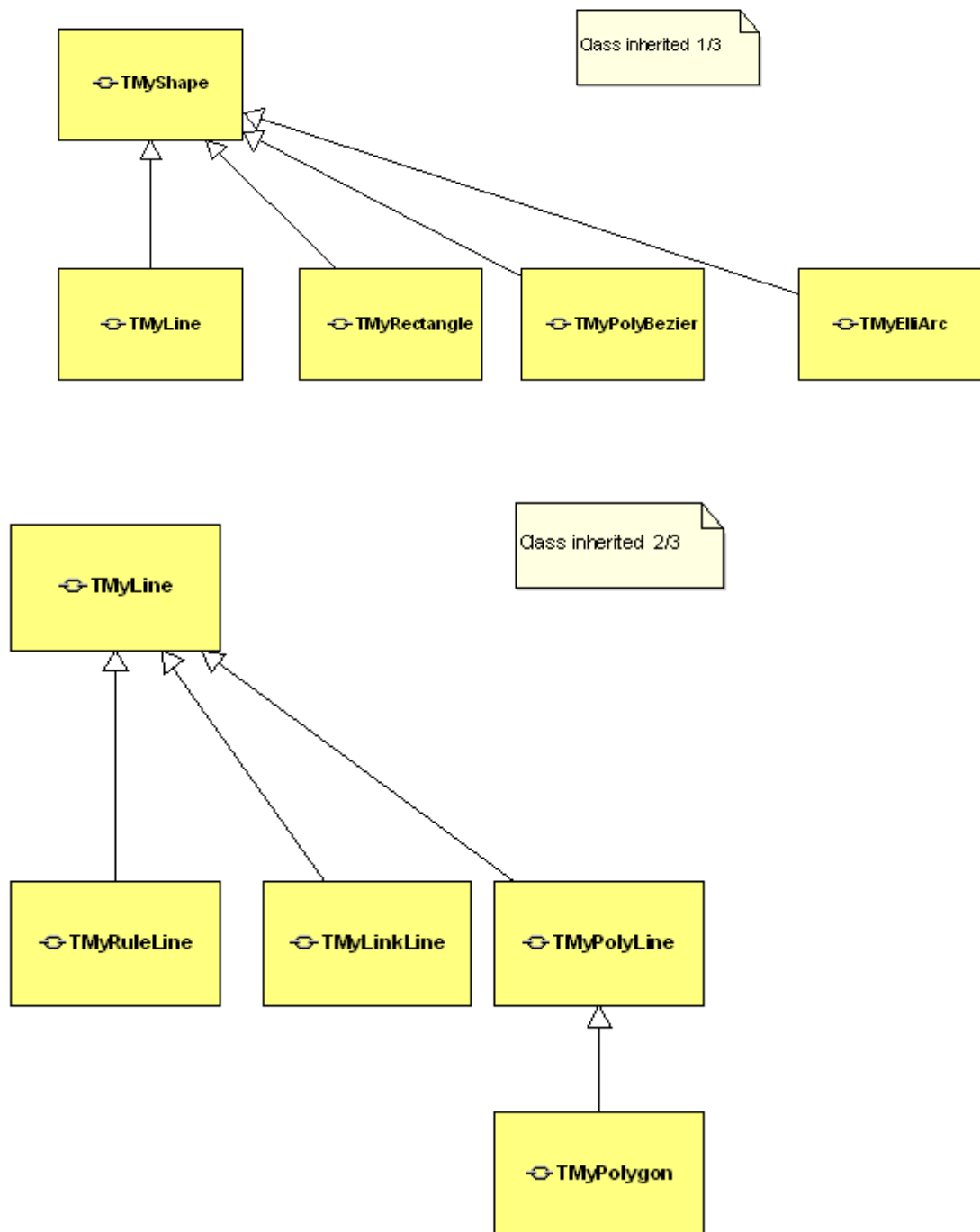
```
MyCAD1.Zoom:= 0.50 ;
```

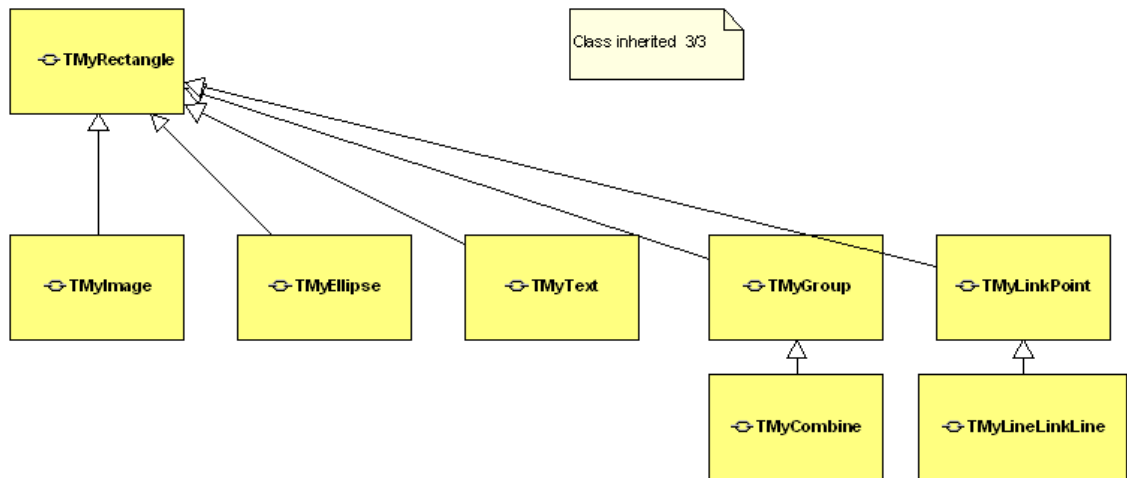
Part



VI

6 Shape Class Inherited Diagram





Part

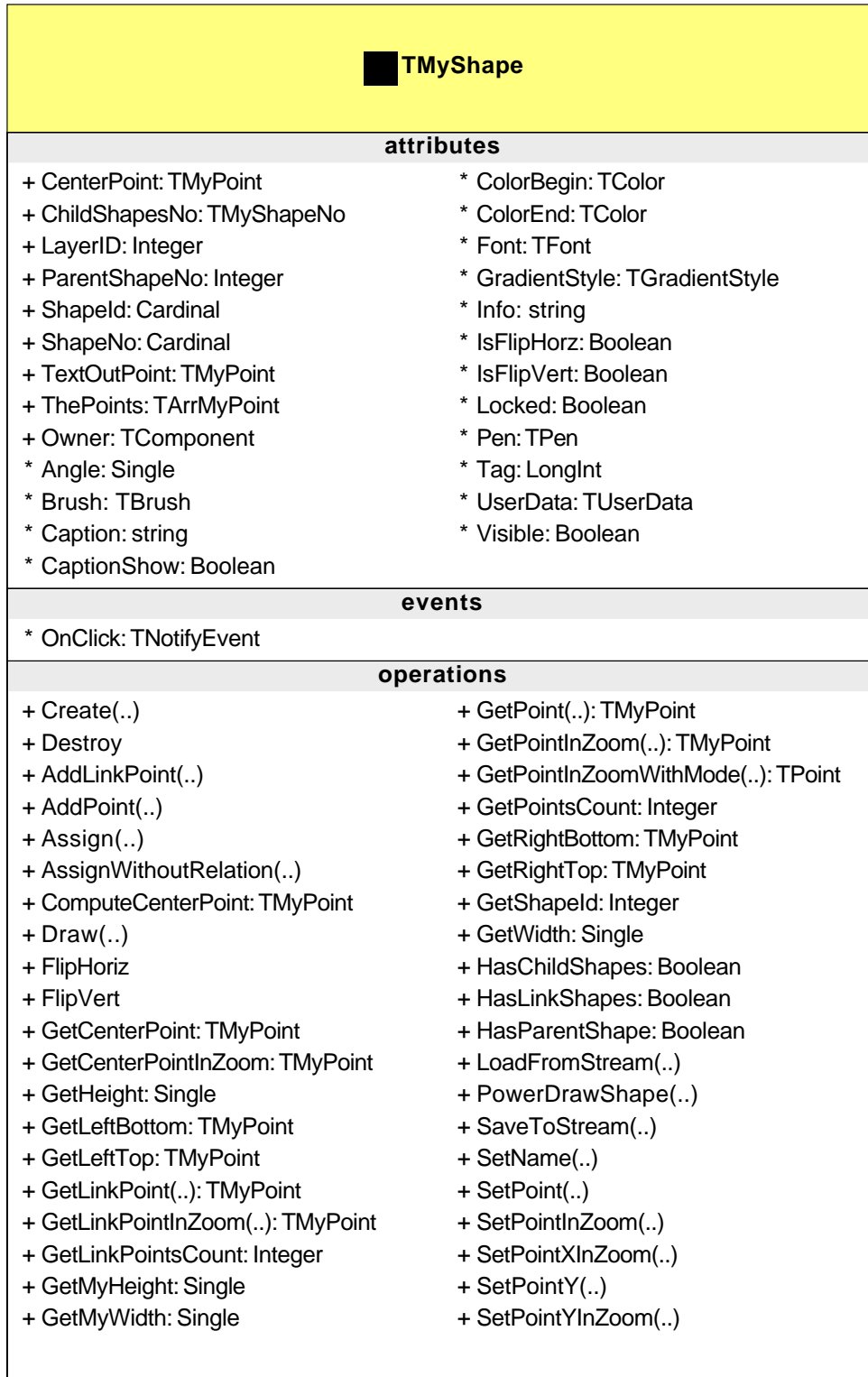


VII

7 TMyShape

It is a base class of shapes, it defines common properties ,events,methods for all shape(s) class.

7.1 ClassDiagram



7.2 Fields

7.2.1 CenterPoint

CenterPoint:TMyPoint;

Description:

It is the CenterPoint for a shape, and maintained by TMyCAD.

7.2.2 ChildShapesNo

ChildShapesNo:TMyShapeNo;

Description:

If the shape is a single shape , this field is empty;if it is a grouped or combined shape, this field will store the child shape no.

7.2.3 LayerID

LayerID:integer

Description:

It indicate the shape belonged which layer,one shape must belonged one layer.

7.2.4 ParentShapesNo

ParentShapesNo:integer;

Description:

It is the parent shape's no., if a shape has not a parent , it is -1.

7.2.5 Shapeld

Shapeld:integer;

Description:

It is the shape's Id, Delete or add a shape , it is not changed, it is a auto-increment field. it is maintaced by TMyCAD and readonly for you.

7.2.6 ShapeNo

ShapesNo:integer;

Description:

It is the order of MyShapes, delete, add, sendback,bringtofront,sendbackbystep,bringtofrontbystep ,will change it. it is maintaced by TMyCAD and readonly for you.

7.2.7 TextOutPoint

TextOutPoint:TMyPoint;

Description:

It is the point that text out position, whan a shape created, it is at center - bottom of a shape, you can change it by mouse or code.

7.2.8 ThePoints

ThePoints:TArrMyPoint

Description:

It is the points of a shape, different shape has different points count.it is not related with XyMode. LeftTop is (0,0). it is base on [TMyPoint](#) type.

7.3 Methods

7.3.1 Assign

procedure Assign(Source: TMyShape); virtual;

Description:

Copy a source shape properties.

Example:

AShape.Assign(BShape);

See also:

[Create](#)

7.3.2 ComputerCenterPoint

function ComputeCenterPoint: TMyPoint;

Description:

It can compute the shape's centerpoint.

Returns:

The center point of a shape.

7.3.3 Create

constructor Create(AOwner: TMyCAD); virtual;

Description:

Internal data structure is initialized.

See also:

[Destroy](#)

7.3.4 Destroy

destructor Destroy; override;

Description:

First all owned fields be released, finally inherited Destroy is called.

See also:

[Create](#)

7.3.5 Draw

procedure Draw(MyCanvas:TCanvas); virtual;

Description:

Draw a shape on the MyCanvas.

7.3.6 GetCenterPoint

function GetCenterPoint: TMyPoint;

Description:

It returns the shape's centerpoint.

Returns:

The center point of a shape.

See Also:

[GetCenterPointInZoom](#)

7.3.7 GetCenterPointInZoom

function GetCenterPointInZoom: TMyPoint;

Description:

It returns the shape's centerpoint in current zoom.

See Also:

[GetCenterPoint](#)

7.3.8 GetHeight

function GetHeight: Single;

Description:

It return a shape's Outside-Rectangle 's Height,it is in pixel unit.

7.3.9 GetLeftBottom

function GetLeftBottom: Single;

Description:

It return a shape's Outside-Rectangle's Left-Bottom position,it is in pixel unit.

7.3.10 GetLeftTop

function GetLeftTop: Single;

Description:

It return a shape's Outside-Rectangle's Left-Top position,it is in pixel unit.

7.3.11 GetLinkPoint

function GetLinkPoint(PointID: integer): TMyPoint;

Description

Get the shape's link point by PointId.

Parameter:

PointID : the point id of the link point that you will retrieve.
it is PointId>=0 and PointId<= GetLnkPointsCount-1

Returns:

the link point.

Example:

Get the first link point of a shape.

```
IF AShape.GetLinkPointsCount >0 THEN  
    APoint:=AShape.GetLinkPoint(0);
```

7.3.12 GetLinkPointInZoom

function GetLinkPointInZoom(PointID: integer): TMyPoint;

Description

Get the link point in current zoom value.Please see [GetLinkPoint](#) to get more information.

7.3.13 GetMyHeight

function GetMyHeight: Single;

Description:

It return a shape's actual height,it leave out the angle, the value is in pixel unit.

7.3.14 GetMyWidth

function GetMyWidth: Single;

Description:

It return a shape's actual width,it leave out the angle, the value is in pixel unit.

7.3.15 GetPoint

function GetPoint(PointID: integer): TMyPoint; public

Description

Get the shape's point by PointId.

Parameter:

PointID : the point id of the point that you will retrieve. it is PointId>=0 and PointId<= GetPointsCount-1

Returns:

the point.

Example:

Get the first point of a shape.

```
IF AShape.GetPointsCount > 0 THEN  
    APoint := AShape.GetPoint(0);
```

7.3.16 GetPointInZoom

function GetPointInZoom(PointID: integer): TMyPoint; public

Description

Get the point in current zoom value. Please see [GetPoint](#) to get more information.

7.3.17 GetPointsCount

function GetPointsCount: Integer; public

Description:

Get how much points of a shape.

7.3.18 GetRightBottom

function GetRightBottom: Single;

Description:

It return a shape's Outside-Rectangle's Right-Bottom position, it is in pixel unit.

7.3.19 GetRightTop

function GetRightTop: Single;

Description:

It return a shape's Outside-Rectangle's Right-Top position, it is in pixel unit.

7.3.20 GetShapeld

function GetShapeld:integer

Description:

To get the shape id of a shape.

Note:

The Shapeld is start from Zero.

7.3.21 GetWidth

function GeWidth: Single;

Description:

It return a shape's Outside-Rectangle's width,it is in pixel unit.

7.3.22 HasChildShapes

function HasChildShapes: Boolean;

Description

To know a shape has child shape(s) or not.

Return:

true : a shape has child shape(s).

false : a shape has no child shape(s).

7.3.23 HasLinkShapes

function HasLinkShapes: Boolean;

Description

To know a shape has linkline shape(s) or not.

Return:

true : a shape has linkline shape(s).

false : a shape has no linkline shape(s).

7.3.24 HasParentShape

function HasParentShape: Boolean;

Description

To know a shape has parent shape(s) or not.

Return:

true : a shape has parent shape(s).

false : a shape has no parent shape(s).

7.3.25 LoadFromStream

procedure LoadFromStream(AStream:TStream); virtual;

Description:

Load a shape from a stream, other class such as TMyline override it.

See also:

[SaveToStream f MyShape](#)

7.3.26 SaveToStream

procedure SaveToStream(AStream:TStream); virtual;

Description:

It can save a shape to a stream, other class such as TMyline override it.

See Also:

[LoadFromStream f MyShape](#)

7.4 Properties

7.4.1 Angle

property Angle:single;

Description:

Set the Angle of TMyShape that you need; it is in rad.

Example:

AShape.Angle:=0.222;

See Also:

[Rotate\(TMyCAD\)](#)

7.4.2 Brush

property Brush:TBrush;

Description:

Set the brush of TMyShape that you need;

Note: 

In tcad of xp.b edition,there's a new function that support brush.bitmap to fill a closed shape.

Code example:

```
var
  tmpBitmap:TBitmap;
begin
  tmpBitmap := TBitmap.Create;
  tmpBitmap.LoadFromFile('d:\test\tmp.bmp');
  MyCAD1.GetSelectedShape.Brush.Bitmap := tmpBitmap;
  MyCAD1.Repaint;
end;
```

7.4.3 Caption

property Caption:string;

Description:

When a shape be created, it is same as name,you can change it.

Example:

AShape.Caption:='it is a line';

See Also:

[CaptionShow](#)

7.4.4 CaptionShow

property CaptionShow:boolean;

Description:

Set the hint show or not. the hint is caption.

Example:

AShape.CaptionShow:= true;

See Also:

[Caption](#)

7.4.5 ColorBegin

property ColorBegin: TColor;

Description:

Set the begin color if you set the gradient style.

See Also:

[ColorEnd](#)
[GradientStyle](#)

7.4.6 ColorEnd

property ColorEnd: TColor;

Description

Set the end color if you set the gradient style.

See Also:

[ColorBegin](#)
[GradientStyle](#)

7.4.7 Font

property Font:TCADFont;

Description:

TCADFont describes font characteristics used when displaying text. TCADFont defines a set of characters by specifying the logheight and logwidth, font family (typeface), attributes (such as bold or italic) and so on. TCADFont encapsulates a windows logical font. it is used for display info field.

See Also:

[Info](#)

7.4.8 GradientStyle

property GradientStyle:TGradientStyle;

Description:

To set gradient style

Example:

AShape.GradientStyle:=gsRadialC;

7.4.9 Info

property Info:string;

Description:

It store string, you can set it as your need.

Example:

AShape.Info:='The is a fun shape'

7.4.10 IsFlipHorz

property IsFlipHorz: boolean;



Description

change the shape flipping state, when it is true, the shape will be flip in horizontally.

See also

[TMyCAD.FlipHorz](#)

7.4.11 IsFlipVert

property IsFlipVert: boolean; 

Description

change the shape flipping state,when it is true, the shape will be flip in vertically.

See also

[TMyCAD.FlipVert](#)

7.4.12 Locked

property Lock:boolean;

Description:

If a shape be locked,the hotspot show gray style.

7.4.13 Name

property Name: TComponentName;

Specifies the name of the component as referenced in code.

Description:

Use Name to change the name of a shape to reflect its purpose in the current application. By default, the new shape will be named "shape1","shape2", and so on.

Warning:

Changing Name at runtime causes any references to the old name to become undefined. Any subsequent code that uses the old name will cause an exception.

7.4.14 Owner

property Owner : TMyCAD;

Description:

The shape is belonged with which instance of TMyCAD.

7.4.15 Pen

property Pen:TPen

Description:

Set the pen of TMyShape that you need;

7.4.16 Tag

property Tag: Longint;

Description

Tag has no predefined meaning. The Tag property is provided for the convenience of developers. It can be used for storing an additional integer value or it can be typecast to any 32-bit value such as a component reference or a pointer.

7.4.17 UserData

property UserData:TUserData;

Description

You can add yourself data , it is very useful.

Example:

```
UserData.AddKeyandValue('Weight','20kg');
```

See also:

[TUserData](#)

7.4.18 Visible

property Visible: boolean;



Description

when it is false,the shape cannot be select,resize and rotate and group.

Part



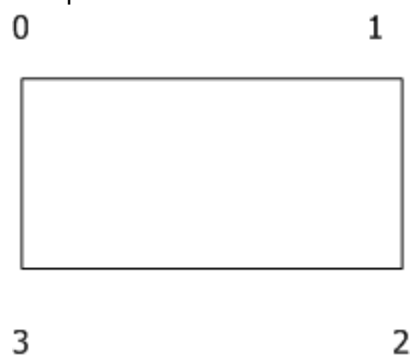
8 TMyCombine

It is a class of Combine shape, it defines properties ,events,methods .

8.1 ClassDiagram



The points Id order:



Part

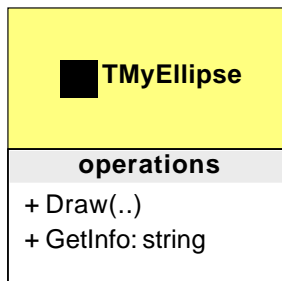


IX

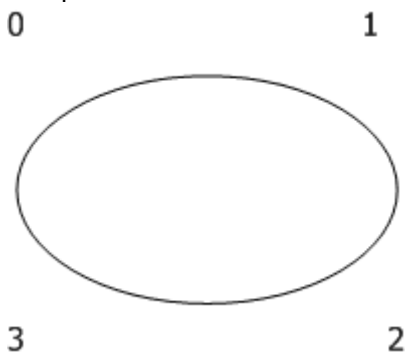
9 TMyEllipse

It is a class of Ellipses(Circle) shape, it defines properties ,events,methods for a Ellipses.

9.1 ClassDiagramofTMyEllipse



The points Id order:



9.2 Methods

9.2.1 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw a ellipse shape on the MyCanvas.

9.2.2 GetCenterPoint

function GetCenterPoint: TMyPoint;override

Description:

function GetCenterPoint overrides inherited GetCenterPoint.,It returns the ellipse shape's centerpoint .

See Also:

[GetCenterPointInZoom f MyEllipse](#)

9.2.3 GetCenterPointInZoom

function GetCenterPointInZoom: TMyPoint;override

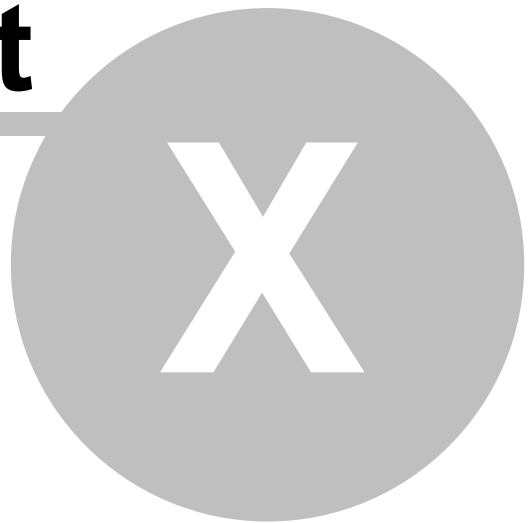
Description:

function GetCenterPointInZoom overrides inherited GetCenterPointInZoom,It returns the ellipse shape's centerpoint in current zoom.

See Also:

[GetCenterPoint f MyEllipse](#)

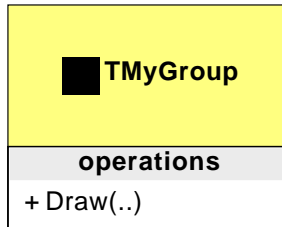
Part



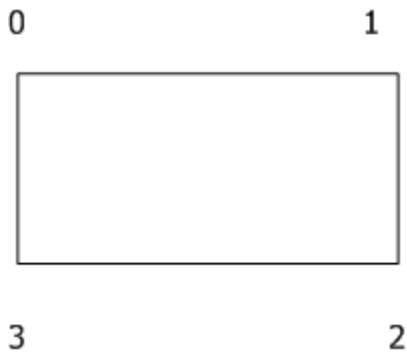
10 TMyGroup

It is a class of group shape, it defines properties ,events,methods .

10.1 ClassDiagram



The points Id order:



10.2 Methods

10.2.1 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw a group shape on the MyCanvas. This method calls child shape to draw on the MyCanvas.

Part

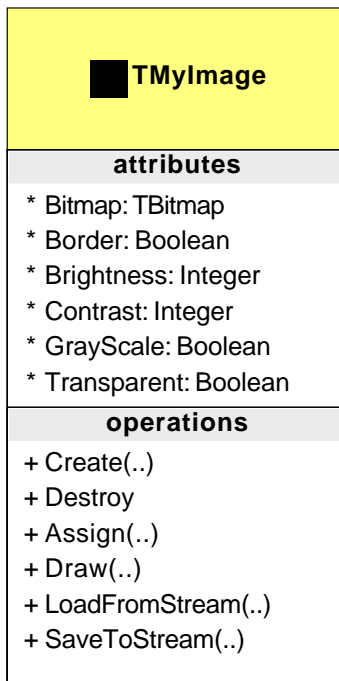


XI

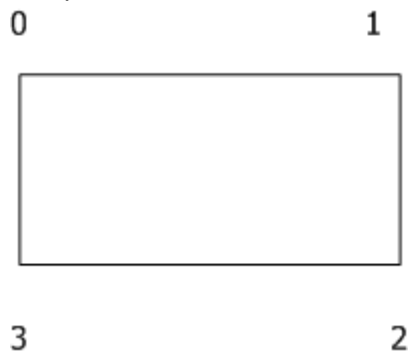
11 TMyImage

It is a class of image shape, it defines properties ,events,methods for a image shape.

11.1 ClassDiagram



The points Id order:



11.2 Properties

11.2.1 Bitmap

property Bitmap:TBitmap

Description:

Set the bitmap for TMyImage shape, for clear it , Btmap:=nil;

Example:

The example show assign a bitmap that you load to TMyImage.

```
var
    mybitmap:TBitmap;
begin
    if OpenPictureDialog1.Execute then
    begin
        myBitmap:=TBitmap.Create;
        mybitmap.LoadFromFile(OpenPictureDialog1.FileName);
        (AShape as TMyImage).Bitmap:=mybitmap;
        MyBitmap.Free;
    end;
end;
```

11.2.2 Border

property Border:boolean



Description:

Set the a border around the bitmap ;

Example:

```
AShape.Pen.Width:=2;
AShape.Pen.Color:=clRed;
AShape.Border:=true;
```

11.2.3 Brightness

property Brightness:integer;



Description:

Set the the brightness for a bitmap , it is -255<= Brightness<=255;

Example:

```
AShape.Brightness:=45;
```

See also:

[Contrast](#)

11.2.4 Contrast

property Contrast:integer;



Description:

Set the the contrast for a bitmap , it is $-100 \leq \text{contrast} \leq 100$;

Example:

```
AShape.constrast:=45;
```

See also:

[Brightness](#)

11.2.5 Grayscale

property Grayscale:boolean;



Description:

Grayscale a color bitmap.

Example:

```
AShape.Grayscale:=true;
```

See also:

[Brightness](#)

[Contrast](#)

11.2.6 Transparent

property Transparent:Boolean

Description:

Set the Transparent for bitmap of TMyImage shape.

11.3 Methods

11.3.1 Assign

procedure Assign(Source: TMyShape); override;

Description:

Copy a source image shape properties.

Example:

AShape.Assign(BShape);

11.3.2 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and Bitmap be created.

11.3.3 Destroy

destructor Destroy; override;

Description:

First all owned fields be released, finally inherited Destroy is called.

See also:

11.3.4 Draw

procedure Draw(MyCanvas: TCanvas); override;

Description:

Draw a image shape on the MyCanvas.

11.3.5 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a image shape from a stream.

See also:

[SaveToStream](#)

11.3.6 SaveToStream

procedure SaveToStream(AStream:TStream); override;

Description:

It can save a shape to a stream.

See also:

[LoadFromStream](#)

Part

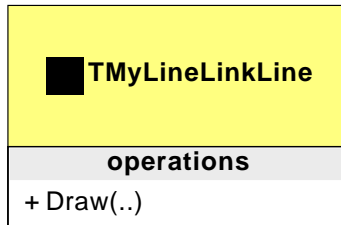


XII

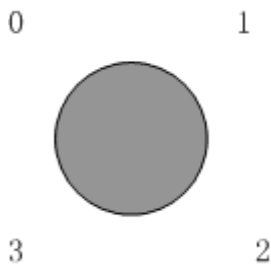
12 TMyLineLinkLine

It is a class of LineLinkLine shape, it defines properties ,events,methods.

12.1 ClassDiagram



The points Id order:



12.2 Methods

12.2.1 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw a line link line shape on the MyCanvas.

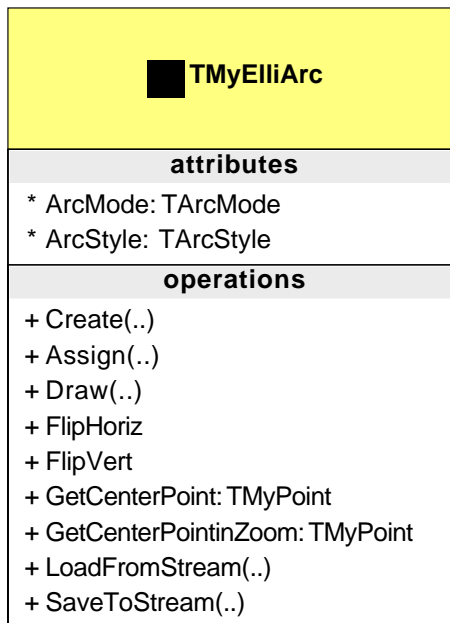
Part



13 TMyElliArc

It is a class of arc shape, it defines properties ,events,methods.

13.1 ClassDiagram



The points Id order:

□



13.2 Properties

13.2.1 ArcMode

property ArcMode: TArcMode;

Description:

There are two choice for aec mode, amCircle and amEllipse.

amCircle:



amEllipse:



13.2.2 ArcStyle

property ArcStyle:TArcStyle;

Description:

There are 3 choices for this property only when ArcMode is amCircle.

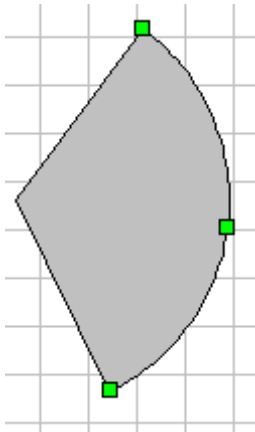
asArc:



asChord:



asSector:



13.3 Methods

13.3.1 Assign

procedure Assign(Source: TMyShape); override;

Description:

Copy a source arc shape properties.

Example:

AShape.Assign(BShape);

13.3.2 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and ArcStyle is asArc ;ArcMode is amCircle;

13.3.3 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw a arc hape on the MyCanvas.

13.3.4 GetCenterPoint

function GetCenterPoint: TMyPoint;override

Description:

function GetCenterPoint overrides inherited GetCenterPoint.,It returns the shape's centerpoint .

See Also:

13.3.5 GetCenterPointInZoom

function GetCenterPointInZoom: TMyPoint;override

Description:

function GetCenterPointInZoom overrides inherited GetCenterPointInZoom,It returns the shape's centerpoint in current zoom.

See Also:

13.3.6 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a arc shape from a stream.

See also:

13.3.7 SaveToStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a arc shape from a stream.

See also:

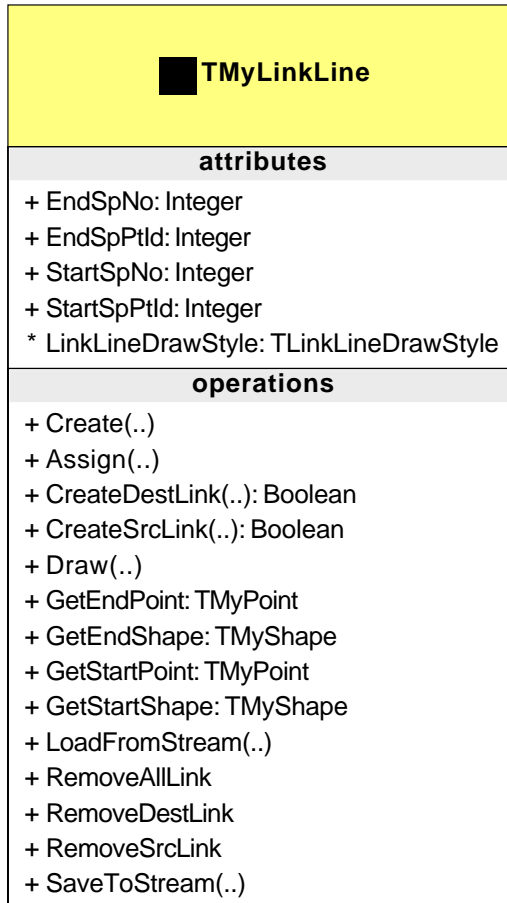
Part

XIV

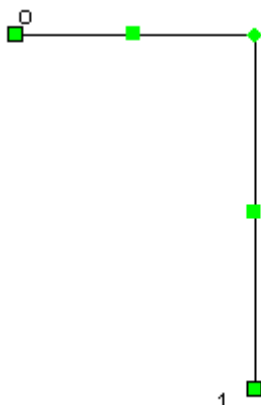
14 TMyLinkLine

It is a class of link line shape, it defines properties ,events,methods.

14.1 ClassDiagram



The points Id order:



14.2 Properties

14.2.1 LinkLineDrawStyle

property LinkLineDrawStyle:TLinkLineDrawStyle

Description:

define linkline style.

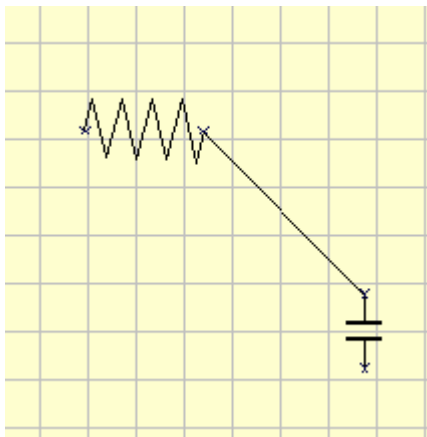
Example:

Delphi syntax:

```
AShape.LinkLineDrawStyle:=lldsFree;
```

C++ syntax:

```
AShape->LinkLineDrawStyle =lldsFree;
```



Delphi syntax:

```
AShape.LinkLineDrawStyle:=lldsHV;
```

C++ syntax:

14.2.2 StartSpPtd

Property StartSpPtd:integer;

Description:

it is read / write, at run time only.The link point id of the start shape.
if StartSpNo is -1,it is must -1.

14.2.3 StartSpNo

Property StartSpNo:integer;

Description:

It is a pointer of start shape. -1:mean no start shape linked.

14.2.4 EndSpNo

Property EndSpNo:integer;

Description:

It is a pointer of end shape. -1:mean no end shape linked.

14.2.5 EndSpPtId

Property EndSpPtId:integer;

Description:

it is read / write, at run time only.The link point id of the end shape.
if EndSpNo is -1,it is must -1.

14.3 Methods

14.3.1 Assign

procedure Assign(Source: TMyShape); override;

Description:

Copy a source line shape properties.procedure Assign overrides inherited Assign.

Example:

AShape.Assign(BShape);

14.3.2 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Constructor Create overrides the inherited Create. First inherited Create is called, then the internal data structure is initialized.LinkLineDrawStyle is lldshv.

14.3.3 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw on the MyCanvas.

14.3.4 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a link line shape from a stream.

14.3.5 CreateDestLink

function CreateDestLink(AShapeId:integer;AShapeLinkPtID:integer): Boolean;

Description:

AShapeId : the end shape id;

AShapeLinkPtID : the link id of the shape,the shape'id is AShapeid.

If can link two shapes, and the path is vertical or horizontal . it is used for flow drawing,electric drawing and more.

Returns:

true : created success;

false : created failed.

See also:

[CreateSrcLink](#)

14.3.6 CreateSrcLink

function CreateSrcLink(AShapeId:integer;AShapeLinkPtID:integer): Boolean;

Description:

it can build a (start) relation ship with exist shape.

Parameter:

AShapeId : the start shape id;

AShapeLinkPtID : the link id of the shape,the shape'id is AShapeid.

Returns:

true : created success;

false : created failed.

See also:

[CreateDestLink](#)

14.3.7 GetEndPoint

function GetEndPoint: TMyPoint;

Description:

Get the link point in end link shape.

Return(s):

if there is no end shape linked,it returns the last point of self;
else it returns the Link point id of the linked shape.

14.3.8 GetEndShape

function GetEndShape: TMyShape;

Description:

Get the end link shape.

Return(s):

nil : there is no end link shape;
else : the instance of the link shape.

14.3.9 GetStartPoint

function GetStartPoint: TMyPoint;

Description:

Get the link point in start link shape.

Return(s):

if there is no start shape linked,it returns the last point of self;
else it returns the Link point id of the linked shape.

14.3.10 RemoveDestLink

procedure RemoveDestLink;

Description:

remove the target linked shape.

14.3.11 RemoveSrcLink

procedure RemoveSrcLink;

Description:

Remove the source linked shape.

14.3.12 GetStartShape

function GetStartShape: TMyShape;

Description:

Get the start link shape.

Return(s):

nil : there is no start link shape;

else : the instance of the link shape.

14.3.13 RemoveAllLink

procedure RemoveAllLink;

Description:

remove the all linked shape.

14.3.14 SaveToStream

procedure SaveToStream(AStream:TStream); override;

Description:

It can save a link line shape to a stream.

Part

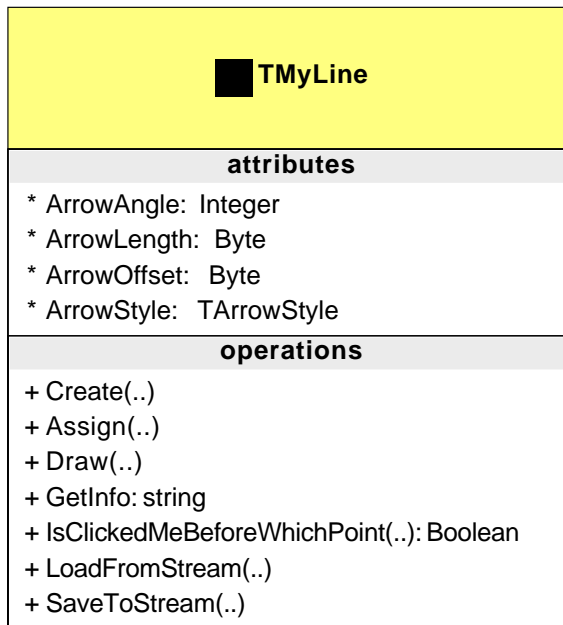


XV

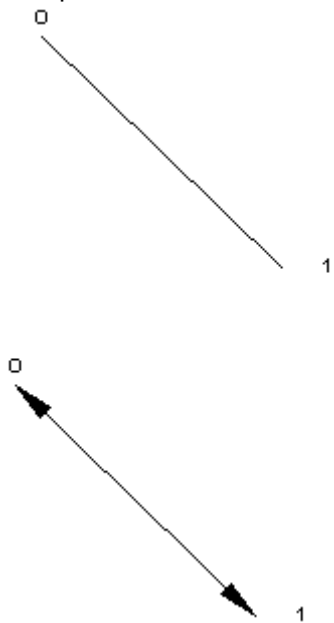
15 TMyLine

It is a class of line shape, it defines properties ,events,methods.

15.1 ClassDiagram



The points Id order:



15.2 Properties

15.2.1 ArrowAngle

property ArrowAngle:integer

Description:

Set the Line and PolyLine 's arrow angle. value is between: 0 - 359

Example:

```
MyCAD1.ArrowAngle := 10 ;
```

15.2.2 ArrowLength

property ArrowLength:Byte

Description:

Set the Line and PolyLine 's arrow Length. value is between: 10-50

15.2.3 ArrowOffset

property ArrowOffset:byte

Description:

When a Line shape is drawn with an arrow, the Arrowoffset specifies how many pixels from the end of the line the arrow is drawn.

value is between: 0 - 255 , default is 0.

15.2.4 ArrowStyle

property ArrowStyle:[TArrowStyle](#)

Description:

Set the Line and PolyLine 's arrow style;

ANone: it is a line;

ALeft: one arrow at the left end of the line or polyline;

ARight: one arrow at the right end of the line or polyline;

ADouble: a double-arrow line or polyline

15.3 Methods

15.3.1 Assign

procedure Assign(Source: TMyShape); override;

Description:

Copy a source line shape properties.

Example:

AShape.Assign(BShape);

15.3.2 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and ArrowOffset, ArrowLength, ArrowAngle, ArrowStyle will be equal TMyCAD's.

15.3.3 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw on the MyCanvas.

15.3.4 GetInfo

function GetInfo: string;override

Description:

function GetInfo overrides inherited GetInfo.,It returns the line shape's length.Change the owner's TheUnit, the area will be changed.

See Also:

[TheUnit f MyCAD](#)

15.3.5 IsClickedMeBeforeWhichPoint

function IsClickedMeBeforeWhichPoint(var BeforeWhichPointID:integer;APoint:

TPoint): Boolean;

Description:

judge the mouse click positon on the line?if not no the line,it return false.

15.3.6 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a line shape from a stream.

See also:

15.3.7 SaveToStream

procedure SaveToStream(AStream:TStream); override;

Description:

Save a line shape to a stream.

See also:

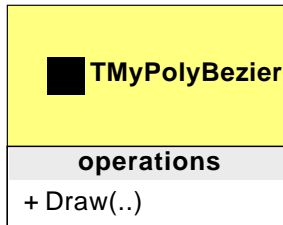
Part

XVI

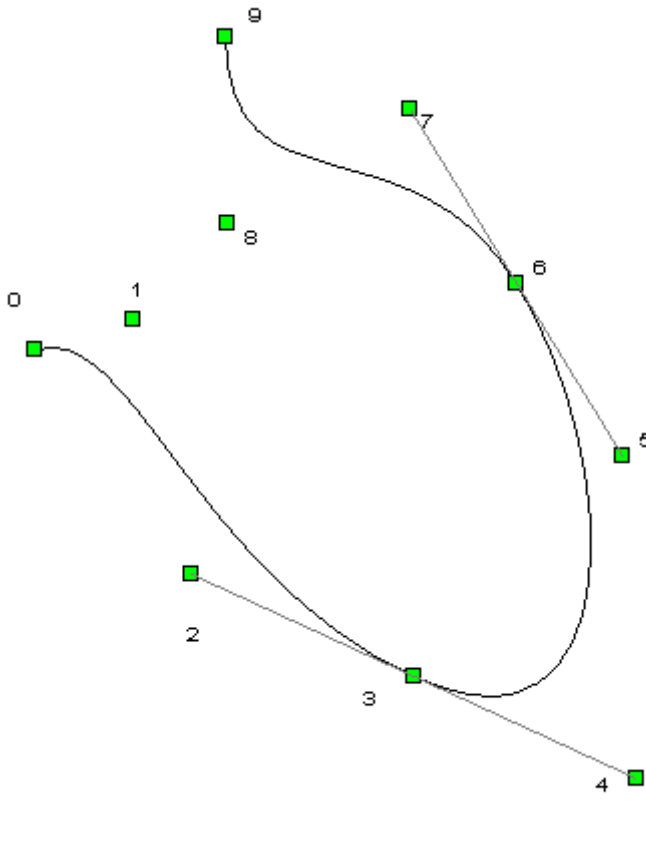
16 TMyPolyBezier

It is a class of polybezier shape, a continuous bezier shape. it defines properties, events, methods.

16.1 ClassDiagram



The points Id order like TMyLine, and it has more than two point.



16.2 Methods

16.2.1 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw on the MyCanvas.

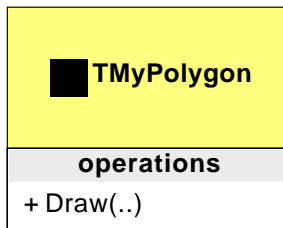
Part



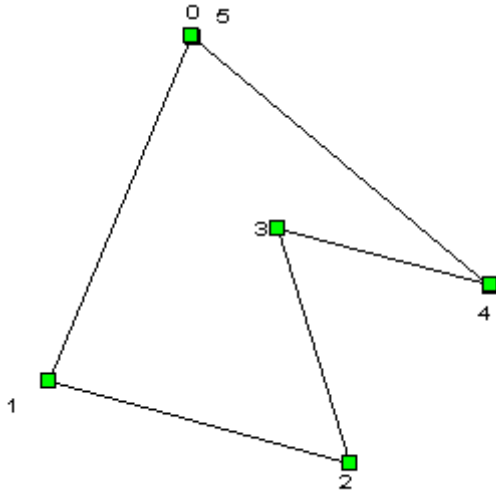
17 TMyPolygon

It is a class of polygon shape, it defines properties ,events,methods.

17.1 ClassDiagram



The points Id order like TMyPolyline, and it has more than two point.



17.2 Methods

17.2.1 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw on the MyCanvas.

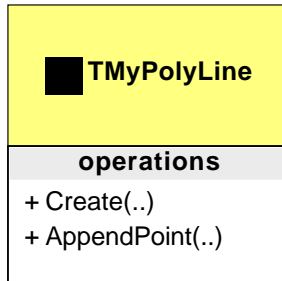
Part



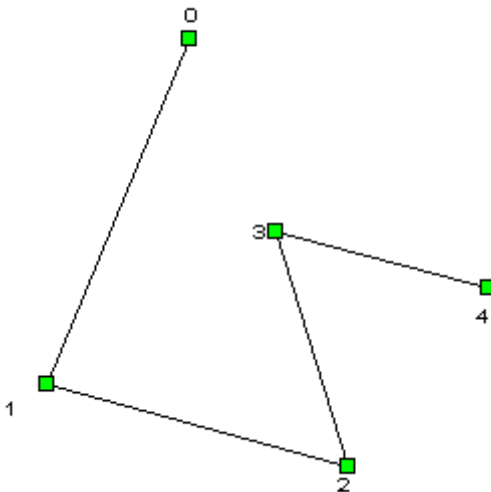
18 TMyPolyLine

It is a class of polyline shape, it defines properties ,events,methods.

18.1 ClassDiagram



The points Id order like TMyLine, and it has more than two point.



18.2 Methods

18.2.1 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and ArrowStyle default is asNone.

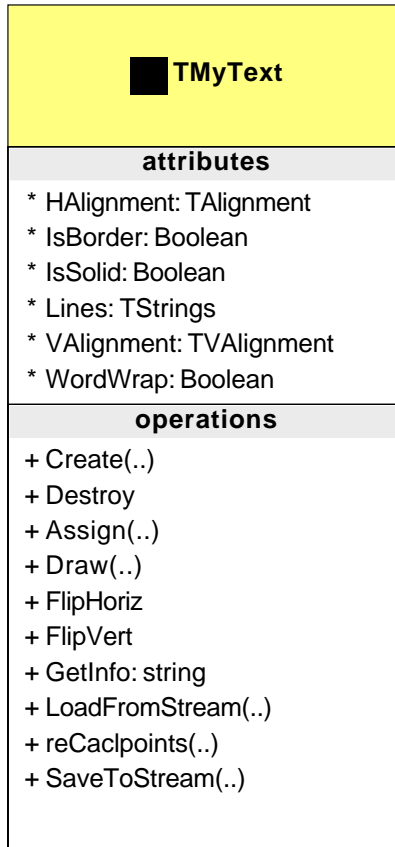
Part

XIX

19 TMyText

It is a class of text shape, it defines properties ,events,methods.

19.1 ClassDiagram



The points Id order:

0

1



3

2

19.2 Properties

19.2.1 HAlignment

property HAlignment:TAlignment;

Description:

Determines how the text is horizontal aligned within the outer rectangle.

Use Alignment to change the way the text is formatted by the TMyText control. Alignment can take one of the following values:

Value Meaning

taLeftJustify	Text is left-justified: Lines all begin at the left edge of the control.
taCenter	Text is centered in the control.
taRightJustify	Text is right-justified: Lines all end at the right edge of the control.

See also:

[VAlignment](#)

19.2.2 IsBorder

property IsBorder:Boolean;

Description:

when it is true,there a rectangle around the text.


19.2.3 IsSolid

property IsSolid:Boolean;

Description:

when it is true,there is a solid text.

19.2.4 Lines

property Lines: TStrings; 

Description

Use Lines to manipulate text in an outer rectangle on a line-by-line basis. Lines is a TStrings object, so the TStrings methods may be used for Lines to perform manipulations such as counting the lines of text, adding new lines, deleting lines, or replacing lines with new text.

Note: if Lines' value is nil,TMyText's info property is enabled,you can use it.

19.2.5 VAlignment

property VAlignment:TVAAlignment; 

Description:

Determines how the text is vertical aligned within the outer rectangle.

Use VAlignment to change the way the text is formatted by the TMyText control. VAlignment can

take one of the following values:

Value	Meaning
vaTop	Text is top: Lines all begin at the top edge of the control.
vaMiddle	Text is centered between top and bottom in the control.
vaBottom	Text is bottom: Lines all end at the bottom edge of the control.

See also:

[HAlignment](#)

19.2.6 WordWrap

property WordWrap: Boolean; 

Description:

Set WordWrap to true to allow the text to display multiple line of text. When WordWrap is true, text that is too wide for the outer rectangle control wraps at the right margin and continues in additional lines.

Set WordWrap to false to limit the label to a single line. When WordWrap is false, text that is too wide for the outer rectangle appears outside.

19.3 Methods

19.3.1 Assign

procedure Assign(Source: TMyShape); override; 

Description:

Copy a source text shape properties.

Example:

```
AShape.Assign(BShape);
```

19.3.2 SaveToStream

procedure SaveToStream(AStream: TStream); override;

Description:

Save a text shape to a stream.

See also:

[LoadFromStream](#)

19.3.3 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a text shape from a stream.

See also:

[SaveToStream](#)

19.3.4 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and IsBorder is false, IsSolid is true.

See also:

[Destroy](#)

19.3.5 Destroy

destructor Destroy; override;

Description:

First all owned fields be released, finally inherited Destroy is called.

See also:

[Create](#)

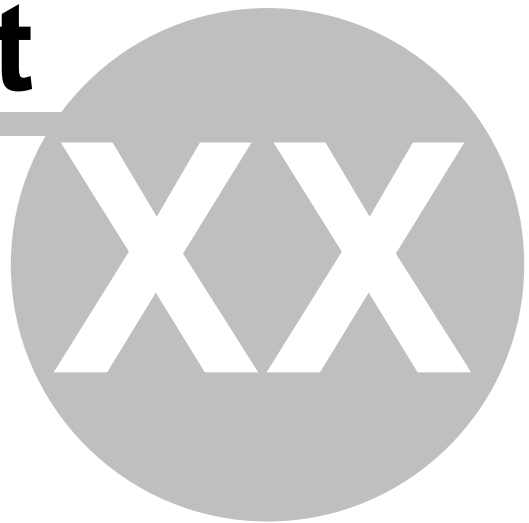
19.3.6 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw a text shape on the MyCanvas.

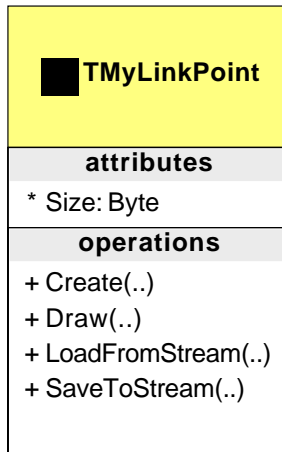
Part



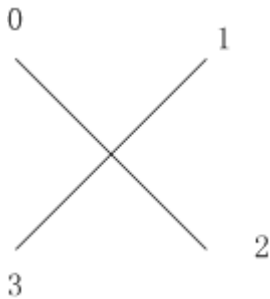
20 TMyLinkPoint

It is a class of linkpoint shape, it defines properties ,events,methods ,it is ONLY for library manager tool.

20.1 ClassDiagram



The points Id order:



20.2 Properties

20.2.1 Size

property Size:byte

Description:

set the size .

20.3 Methods

20.3.1 SaveToStream

procedure SaveToStream(AStream:TStream); override;

Description:

Save a linkpoint shape to a stream.

See also:

20.3.2 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a linkpoint shape from a stream.

See also:

20.3.3 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and size is 8 pixel.

20.3.4 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw a linkpoint shape on the MyCanvas.

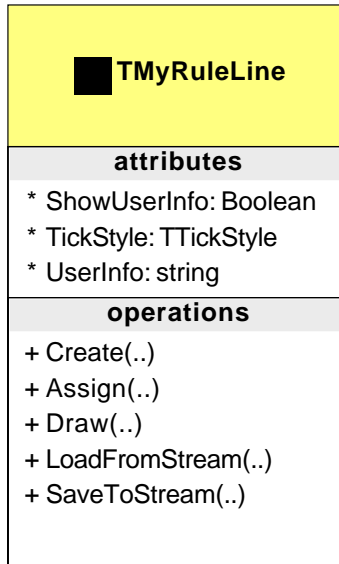
Part

XXI

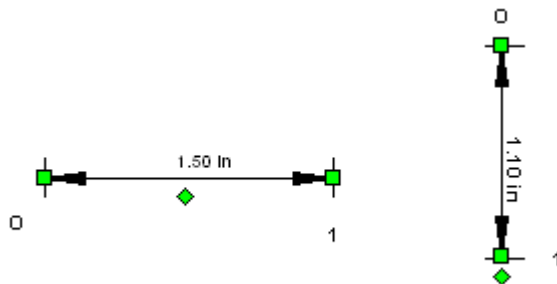
21 TMyRuleLine

It is a class of ruleline shape, it defines properties ,events,methods for a rectangle.

21.1 ClassDiagram



The points Id order:



21.2 Properties

21.2.1 UserInfo

property UserInfo:string;

Description:

set the userinfo.

See also:

[ShowUserInfo](#)

21.2.2 ShowUserInfo

property ShowUserInfo:boolean;

Description:

When it is true,UserInfo showed on the rule line, else show the length string computed by TMyRuleline.

21.2.3 TickStyle

property TickStyle:TTickStyle;

Description:

TTickStyle=(tsLine,tsNone);

When it is tsLine, the TMyRuleline has a two small line beside two end point;
when it is tsNone, the TMyRuleline has no small line beside two end point;

21.3 Methods

21.3.1 Assign

procedure Assign(Source: TMyShape); override;

Description:

Copy a source line shape properties.

Example:

AShape.Assign(BShape);

21.3.2 Create

constructor Create(AOwner: TMyCAD); override;

Description:

Internal data structure is initialized and ArrowStyle is ADouble, UserInfo is "", ShowUserInfo is False, TickStyle is tsLine;

21.3.3 LoadFromStream

procedure LoadFromStream(AStream:TStream); override;

Description:

Load a rule line shape from a stream.

See also:

21.3.4 Draw

procedure Draw(MyCanvas:TCanvas); override;

Description:

Draw on the MyCanvas.

21.3.5 SaveToStream

procedure SaveToStream(AStream:TStream); override;

Description:

Save a rule line shape to a stream.

See also:

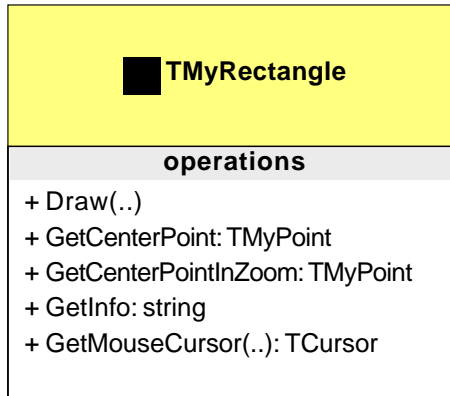
Part



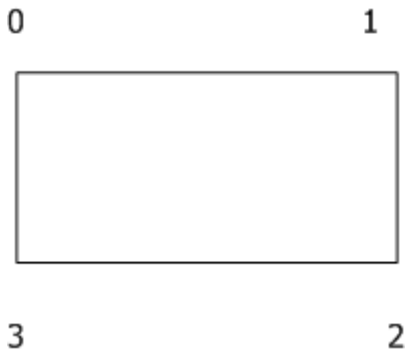
22 TMyRectangle

It is a class of rectangle shape, it defines properties ,events,methods for a rectangle.

22.1 ClassDiagram



The points Id order:



22.2 Methods

22.2.1 Draw

procedure Draw(MyCanvas: TCanvas); override;

Description:

Draw a rectangle shape on the MyCanvas.

22.2.2 GetCenterPoint

function GetCenterPoint: TMyPoint; override

Description:

function GetCenterPoint overrides inherited GetCenterPoint.,It returns the rectangle shape's centerpoint .

See Also:

[GetCenterPointInZoom f MyRectangle](#)

22.2.3 GetCenterPointInZoom

function GetCenterPointInZoom: TMyPoint;override

Description:

function GetCenterPointInZoom overrides inherited GetCenterPointInZoom,It returns the rectangle shape's centerpoint in current zoom.

See Also:

[GetCenterPoint f MyRectangle](#)

22.2.4 GetInfo

function GetInfo: string;override

Description:

function GetInfo overrides inherited GetInfo.,It returns the rectangle shape's area.Change the owner's TheUnit, the area will be changed.

See Also:

[TheUnit f MyCAD](#)

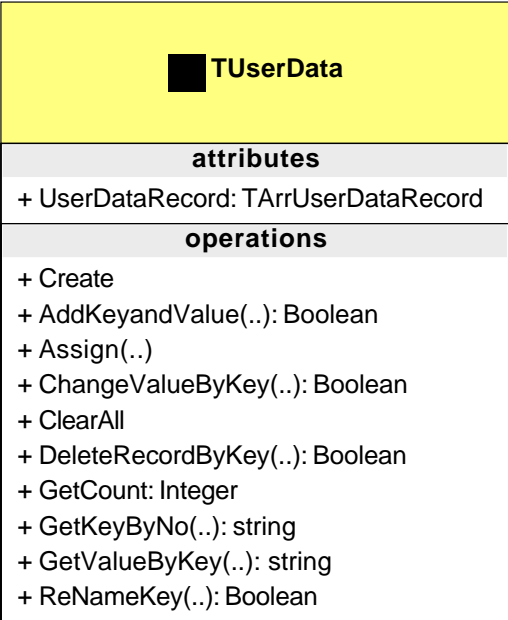
Part

XII

23 TUserData

This is class lets you store yourself data for a (single or group) shape.

23.1 ClassDiagram



23.2 Property

23.2.1 UserDataRecords

property UserDataRecords:TArrUserDataRecord;

Description:

Store userdefine properties

See also:

[Defines](#)

23.3 Methods

23.3.1 Create

constructor Create;

Description:

Internal data structure is initialized ;

23.3.2 AddKeyAndValue

funcrion AddKeyAndValue(Const AKey,AValue:string):boolean;

Description:

Add key and value to the TUserData instance;

Parameter:

AKey: a key for the user define property, it is a string;
AValue: the value, it is a string;

Returns:

true: add success;
false: add failed, if the key is existed, it will return false;

Examples:

```
UserData1.AddKeyandValue('Name','John');
```

23.3.3 Assign

procedure Assign(Source:TUserData)

Description:

Copy a instance of TUserData.

Parameter:

Source: other exist instance of TUserData

Examples:

```
UserData1.Assign(UserData0);  
//Userdata1's data will be lost and rewrite
```

23.3.4 ChangeValueByKey

function ChangeValueByKey(const AKey:string;AValue:string):boolean;

Description:

Change the value by key. if AKey do not match , it return false else return true;

Parameter:

AKey: exist key;
AValue: new value ;

Examples:

```
If  UserData1.ChangeValueByKey( 'Name' , 'Rose' ) THEN  
  ShowMessage( 'Changed!' );
```

23.3.5 ClearAll

procedure ClearAll;

Description:

Clear all data;

23.3.6 DeleteRecordByKey

function DeleteRecordByKey(AKey:string): Boolean

Description:

Delete a record by a key;

Parameter:

AKey: the exist key;

Returns:

true: delete success;
false:delete failed, a key cannot be found,

23.3.7 GetCount

function GetCount: Integer;

Description:

Get how many records existed.

23.3.8 GetKeyByNo

function GetKeyByNo(const ANo:integer): string;

Description:

Get the key string by no. if no do not match , it return false else return true;

Parameter:

ANo: the record id in the records, it is from zero.

Examples:

```
ShowMessage('the first key name is: ' + UserData1.GetKeyByNo(0));
```

23.3.9 GetValueByKey

function GetValueByKey(const AKey:string): string;

Description:

Get the value by key. if no do not match , it return false else return true;

Parameter:

AKey: the key string in the records.

Examples:

```
ShowMessage('the first key name is: ' + UserData1.GetKeyByNo(0)+' Value: '+UserData1.GetValueByKey(UserData1.GetKeyByNo(0) ));
```

23.3.10 ReNameKey

function ReNameKey(OldKey,NewKey:string):boolean;

Description:

Rename a key name. if no do not match , it return false else return true;

Parameter:

AKey: the key string in the records.

Part



24 About Crystal Component

Our goal is to provide you with useful components and we hope makes it easier for you to create great application with us. we have been serving the image and graphic components since 1998.

Our website:

You can get all information about our products from our web.

<http://www.codeidea.com>

<http://cad.codeidea.com>

E-Mail:

hongbin.fei

webmaster@codeidea.com

yuefen.yao

support@codeidea.com

Telephone:

+86 572 7281888 (Office)

+86 572 2607144(Fax)

+86 (0)13335721372 (Mobile)

Address:

Room 303#,304#
699# Road QingTong
HuZhou, ZheJiang
China
313000

Index

- A -

About Crystal Component 165
AddShapeByCode 32
AlignBottom 34
AlignLeft 34
AlignRight 35
AlignTop 35

- B -

BkBitmapMode 68
BringToFront 35
Brush 68

- C -

ColorOfBackground 69
ColorOfHot 69
Create 38
CurrentLayerId 70

- D -

defines 14
DeleteAllLayers 39
DeleteAllShapes 39
DeleteLayerById 40
DeleteLayerByName 40
DeleteSelectedShape 40
DeleteShape 41
Destroy 41
DrawAllShape 42

- G -

GetLayerIdByName 43
GetLayerIdByNo 43
GetLayerNameById 43
GetLayerNoById 44
GetLayerNoByName 44
GetLayersCount 45
GetMaxLayerId 45
GetShapesCount 49

GetShapesCountInALayer 49
GridHeight 17
GridShow 17
GridWidth 18
GroupWorkingShape 50

- I -

InVisibleLayerById 50
InVisibleLayerByName 50
IsTCADFile 51
IsVisibleLayerById 51

- L -

LabelXY 73
LableValue 72
LoadFromFile 51
LoadFromStream 52

- N -

NewLayer 53

- P -

PageHeight 78
PageOrientation 78
PageStyle 78
PageWidth 79
Pen 79
Print 54
PrintABorder 79
PrintBackground 81
PrintPreview 56

- R -

Ratio 81
ReturnToSelecting 82

- S -

SaveToFile 58
SaveToStream 59
SendtoBack 62
SetLayerNameById 62
SetLayerNameByName 63
SetMyImage 63

ShapeTool 83

Snap 84

SnapPixels 84

- T -

TheUnit 85

- U -

UnGroupShape 64

- V -

VisibleAllLayer 65

VisibleLayerByID 65

- Z -

Zoom 86



www.codeidea.com
