TMS TAdvCardList
DEVELOPERS GUIDE

June 2020

Web: https://www.tmssoftware.com
Email: info@tmssoftware.com

Table of Contents

Design

1. Introduction

TAdvCardList is a Delphi package that contains two visual components – TAdvCardList and TDBAdvCardList. Main goal of these components to organize cards of contacts like in Microsoft Outlook with a A..Z bar to quickly browse through cards. Project divides into two phases: first – development of TAdvCardList; second – to change TAdvCardList for working with data base as source of data for cards (TDBAdvCardList).

1.1 Definitions

ContactList – native Delphi VCL package with components for working with cards of contacts.

Item Type – type of item data and editor for modifying. List of data types and editors:

TAdvCardItemType = (actLabeledItem, actItem, actImage, actOwnerDraw);

// actMemo removed – the same as actItem, aceText with LineCount > 1

// actImage needs to specification

TAdvCardItemEditor = (aceText, aceNumber, aceFloat, aceBoolean, aceDropDownList, aceDropDownEdit, aceDate, aceTime, aceCustom);

TAdvCardItemDataType = (cdtString, cdtFloat, cdtInteger, cdtBoolean, cdtDate, cdtTime);

2. Project Design

2.1 Problem Statement

The problem of	At this moment there are no efficient, reliable components for providing client abilities.
Affects	Software developers
the impact of which is	It is needed to develop Delphi VCL design and runtime package uses Win32Api and Delphi VCL. Main problem is need of deep understanding of WinApi, object programming and objects interaction development.
a successful solution would be	Visual components to organize contact cards with specified items written in Delphi 7.

2.2 User View

From user point of view this package must be install in Delphi "ContactList" palette tab and work properly under Windows.

This package must contain three essential components – TAdvCardList & TAdvCardButtonBar (phase 1) and TDBAdvCardList (phase 2) with the next features:

TAdvCardList

- properties for assigning layout of card, count of items, their data and editor type. Should be provided features for assigning data to instances of object, such as CardAppearance, CardTemplateItem, CardTemplate and so on;
- all standard Windows and Delphi events, and events concerned with data changing for all objects should be provided.

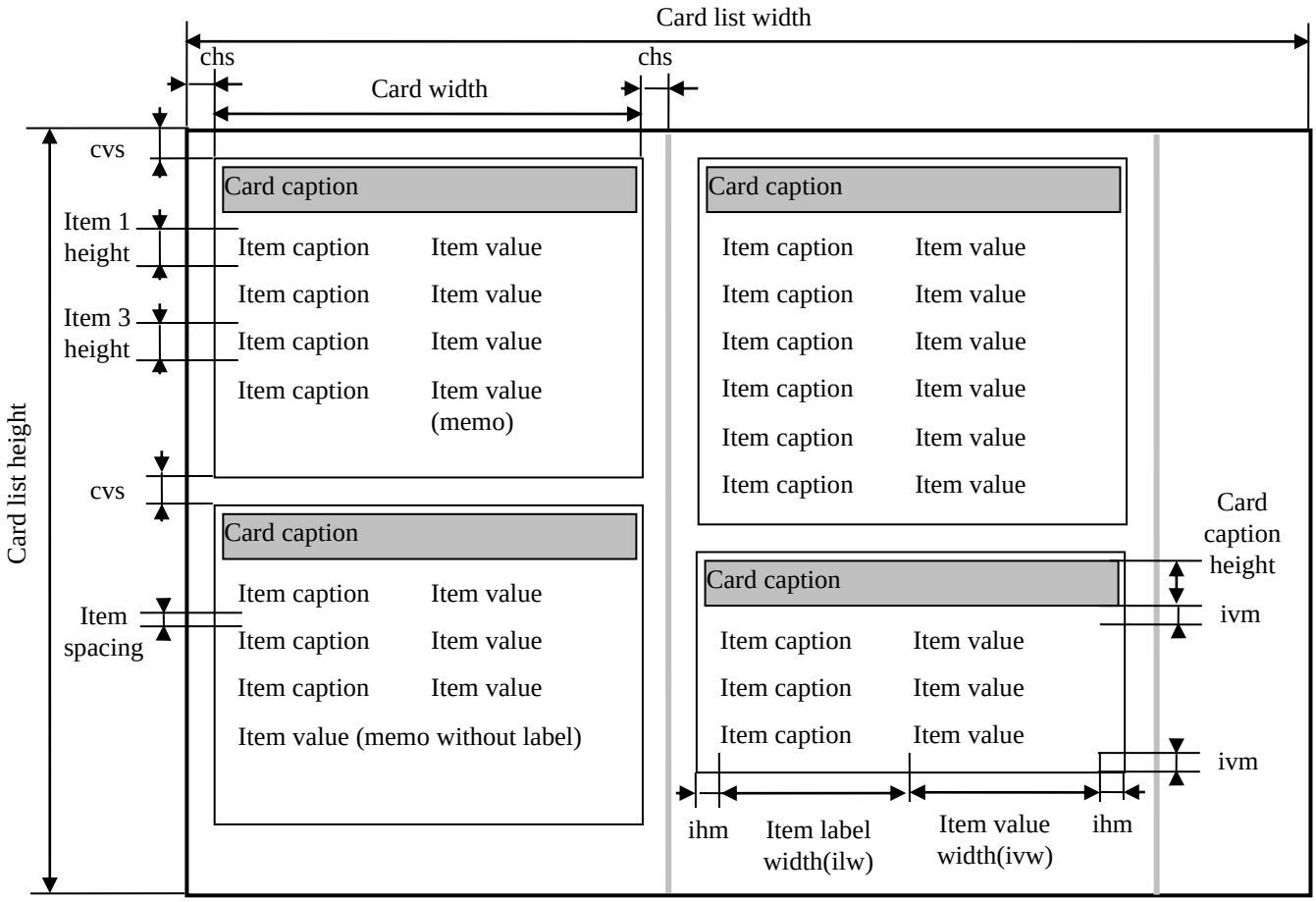
TDBAdvCardList

- based on TAdvCardList;
- additional properties for working with data base as data source of cards content.

2.3 Internal View

For successful development the package we must define essential and additional objects and their ancestors. Also we need to define own data types. Full list of properties and events must be realized in published sections of objects. Some properties maybe be placed in public section and have access only in runtime. From developer point of view main object TAdvCardList (ancestor is TCustomControl) has template of card layout TAdvCardTemplate (TPersistant), which describes all items in the card and their data type. Access to each card and item's data occurs through TAdvCards object (TCollection). Behavior of card in each case describes in instances of TAdvCardAppearance (TPersistant) object. Also we need some additional objects for filtering and sorting features including TAdvButtonsBar (TGraphicControl) for quick browse through cards.

MOCK-UP OF OBJECTS



chs – card horizontal spacing;
cvs – card vertical spacing;
ihm – item horizontal margins;
ivm – item vertical margins;
ivw = card width – ilw – ihm*2 : ilw = configurable by property
column width = card width + chs*2

```
unit AdvCardList;

interface

  TAdvGradientDirection = (gdHorizontal, gdVertical);

  TAdvGradient = class(TPersistent)
  published
    Color: TColor; default clWindow; // if clNone use clear brush
```

```
    ColorTo: TColor; default clNone; // if clNone use solid Color
    Direction: TadvGradientDirection; default gdHorizontal
    { events }
    OnGradientChange(Sender: TObject);
  end;

  TAdvCardAppearance = class(TPersistent)
  Published
    BevelInner : for panel-like appearance
    BevelOuter
    BevelWidth
    BorderColor: TColor; default clNone;
    BorderWidth: Integer; default 1;
    CaptionColor: TAdvGradient;
    CaptionBorderColor: TColor; default clNone;
    CaptionBorderWidth: Integer; default 1;
    CaptionFont: TFont;
    Color: TAdvGradient;
    ItemLabelFont: TFont; // used for replacing all item label fonts in this appearance  : conflicting with
TemplateItem.LabelFont ???
    ItemEditFont: TFont; // used for replacing all item edit fonts in this appearance
    ReplaceLabelFont: Boolean; default False; // if checked ItemLabelFont used for all items ? used for ???
    ReplaceEditFont: Boolean; default False; // if checked ItemEditFont used for all items ? used for ???
    LabelWidth: sets width of the label
  end;

  TAdvCardItemType = (actLabeledItem, actItem, actImage, actOwnerDraw);
  // actMemo removed – the same as actItem, aceText with LineCount > 1
  // actImage needs to specification

  TAdvCardItemEditor = (aceText, aceNumber, aceFloat, aceBoolean, aceDropDownList, aceDropDownEdit,
aceDate, aceTime, aceCustom);

  TAdvCardItemDataType = (cdtString, cdtFloat, cdtInteger, cdtBoolean, cdtDate, cdtTime);

  TAdvCardStringHideCondition = (chcAlwaysShow, chcEmpty, chcNotEmpty, chcCustom);
  TAdvCardDateHideCondition = (chcAlwaysShow, chcNullDate, chcNotNullDate, chcCustom);
  TAdvCardFloatHideCondition = (chcAlwaysShow, chcNull, chcNotNull, chcCustom);
  TAdvCardIntegerHideCondition = (chcAlwaysShow, chcNull, chcNotNull, chcCustom);
  TAdvCardBooleanHideCondition = (chcAlwaysShow, chcTrue, chcFalse, chcCustom);
  // shs changed to chc (card hide condition)
  // chcCustom needs to specification

  TAdvCardTemplateItem = class(TCollectionItem)
  public
  published
    AutoSize: Boolean; // when true, item is autosizing based on textheight
    LabelFont: TFont; ; conflict with appearance LabelFont ?
    Caption: string;
    CaptionColor: TColor; default clNone; // background of caption; clear if clNone
```

```
  DefaultValue: Variant; // default value for new card
  EditFont: TFont;
  ValueColor: TColor; default clNone; // background of value; clear if clNone
  EditColor: TColor; default clWindow; // background of item value in edit mode (for controls)
  ItemType: TAdvCardItemType;default actLabeledItem;
  ItemEditor: TAdvCardItemEditor; default aceText;
  DataType: TAdvCardItemDataType; // removed from TAdvCardItem
  List: TStringList; // for DropDown
  LineCount: Integer; default 1; // for aceText  only to determine or Edit either Memo use; if = 1 – Edit, otherwise – Memo (when wordwrap is choosen, there should be a way to let it autosize according to text, for example with LineCount = -1 ??)
  Indent: Integer; default 0; // indent for item value
  Height: Integer; default 22; // used when AutoSize is False
  MaxHeight: Integer; default 66; // used when AutoSize is True for prevent infinite height growing
  Name: string; // Name of the item for object property editor
  ReadOnly: Boolean;
  HideStringCondition: TAdvCardStringHideCondition; // work also if Visible = True
  HideIntegerCondition: TAdvCardIntegerHideCondition;
  HideDateCondition: TAdvCardDateHideCondition;
  HideFloatCondition: TAdvCardFloatHideCondition;
  Tag: Integer;
  WordWrap: boolean; // when true, memo text is shown wordwrapped, otherwise with end ellipsis
  Format: string; // floating format string if datatype is a float
  Prefix: string; // prefix string for display only, on edit will be hided
  Suffix: string; // suffic string for display only, on edit will be hided
  ShowHint: boolean; // show special item hint
  Visible: Boolean; default True; // for manual hiding of item
  { events }
   OnTemplateItemChange(Sender: TObject; Item: TAdvCardTemplateItem);
 end;

// united with TAdvCardTemplate
{  TAdvCardTemplateItemCollection = class(TCollection)
  public
  published
   Items[index: Integer]: TAdvCardTemplateItem;
  end; }

 TAdvCardTemplate = class(TCollection)
 Public
  Function GetTemplateItemClass: TCollectionItem; virtual; (allow to override to create custom classes)
  GetItemByName(Name: string): TAdvCardTemplateItem;
  ItemValueWidth: Integer; // read only; equal CardWidth – ItemLabelWidth – ItemHorMargins*2
 published
  Items[index: Integer]: TAdvCardTemplateItem;
  CardCaptionHeight: Integer; default 22;
  ItemLabelWidth: Integer; default 100; // see figure
  CardWidth: Integer; default 200;
 { removed from  TAdvCardAppearance }
  ItemSpacing: Integer; default 5; // spacing between two items
```

```
  HorMargins: Integer; default 10; // left and right margins from border to keep for displaying items (ihm in figure)
  VertMargins: Integer; default 10; // top and bottom margins from border to keep for displaying items (ivm in figure)
 { events }
  OnTemplateItemAdd(Sender: TObject; Item: TAdvCardTemplateItem);
  OnBeforTemplateItemDelete(Sender: TObject; Item: TAdvCardTemplateItem);
  OnCardTemplateChange(Sender: TObject);
 end;

 TAdvCardItem = class(TCollectionItem) // or array or TList element
 public
  AsBoolean: Boolean;
  AsString: string;
  AsFloat: double;
  AsInteger: integer;
  AsDate: TDateTime;
  AsTime: TDateTime
  DataType: TAdvCardItemDataType; // removed to TAdvCardTemplateItem
  Hint: string; // specific item hint
  Selected: Boolean;
  Object: TObject;  // to allow assignment of custom objects
  OwnsObject: Boolean; // when true, object is destroyed automatically when item is destroyed
  Tag: Integer;
 end;

 TAdvCardItemList = class(TCollection) // or implemented by array or TList
 Public
  Function GetCardItemClass: TCollectionItem; virtual;  // to allow override of carditem class, default is TAdvCardItem
  Card: TAdvCard;  // owner card
 published
  Items[index: Integer]: TAdvCardItem; // not allowed change item count
 end;

 TAdvCard = class(TCollectionItem) // or array or TList element
 public
  Selected: Boolean;
  SelectedItem: Integer;
 published
  Caption: TCaption;  // what is caption used for , set in template ???
  ItemList: TAdvCardItemList;
  Tag: Integer;
  Hint: string; // full card hint
 end;

 TAdvCards = class(TCollection) // or implemented by array or TList
 public
  Items[index: Integer]: TAdvCard;
  SelectedCard: Integer;
 published
```

```
  { events }
  OnCardAdd(Sender: TObject; Card: TAdvCard);
  OnBeforCardDelete(Sender: TObject; Card: TAdvCard);
  OnCardChange(Sender: TObject; Card: TAdvCard);
 end;

 TSortDirection = (sdAscending, sdDescending);

 TAdvCardSortSettings = class(TPersistent)
  SortIndex: integer; // card item to sort on
  SortDirection: TSortDirection;
  { events }
  OnSortChange(Sender: TObject);
 end;

 TAdvCardFilterSettings = class(TPersistent);
  FilterIndex: integer; // card item to filter on
  FilterCondition: string; // simple condition : if pos(filter, carditem[index]) = 1 then allow
  { events }
  OnFilterChange(Sender: TObject);
 end;

 TAdvCardEvent = procedure(Sender: TObject; CardIndex: Integer; ItemIndex: Integer);
 TAdvCardItemEvent = procedure(Sender: TObject; CardIndex: Integer; ItemIndex: Integer);
 TAdvCardItemAllowEvent = procedure(Sender: TObject; CardIndex: Integer; ItemIndex: Integer; var Allow:
Boolean);

 TAdvCardList = class(TCustomControl)
 protected
  procedure StartEdit(editrect:trect; cardindex, itemindex: integer; value: variant); // method called when custom
editing should start
 public
  Items: TAdvCardItemList;
  SelectedCard: TAdvCard;
  SelectedIndex: Integer;
  function CardAtXY(x,y: integer): TAdvCard;
  function CardItemAtXY(x,y: integer): TAdvCardItem;
  procedure DoneEdit(cardindex, itemindex: integer; value: variant); method called by custom editor to update after
editing ends
  LeftCol: sets the left column index in a scrolled cardlist
 published
  BorderWidth: Integer;
  BorderColor: TColor; // without border if clNone
  CardNormalAppearance: TAdvCardAppearance;
  CardSelectedAppearance: TAdvCardAppearance;
  CardHoverAppearance: TAdvCardAppearance;
  CardEditingAppearance: TAdvCardAppearance;
  CardTemplate: TAdvCardTemplate; // template of cards
  CardVertSpacing: Integer; default 20; // see figure
  CardHorSpacing: Integer; default 20; // see figure
```

```
    ColumnSizing: Boolean; // ability to resize column width in realtime
    Color: TAdvGradient; // background card list color
    Columns: Integer; // nr. of columns; read only
    ColumnWidth: Integer; // if changed card width recalculate and vice versa if card width changed then column
width recalculate
    ReadOnly: Boolean;
    GridLineWidth: Integer; default 3; // line between columns
    GridLineColor: TColor;  // line between columns
    MultiSelect: Boolean; // allow ctrl-click multi card selection
    SortSettings: TAdvCardSortSettings;
    Sorted: Boolean;
    ShowScrollBar: Boolean (show scrollbar or not)
    FilterSettings: TAdvCardFilterSettings;
    Filtered: Boolean;
    { events }
    OnCardStartEdit: TAdvCardItemAllowEvent;
    OnCardEndEdit: TAdvCardItemEvent;
    OnCardItemGetDisplText(Sender, CardIndex, ItemIndex; Integer; var Text: string); // allows virtual text or
dynamic text modifications
    OnDrawCardItem(Sender: TObject; CardIndex; ItemIndex; Card: TAdvCard; Item: TAdvCardItem; Canvas,
Rect); // custom draw event
    OnDrawCardItemProp(Sender: TObject; CardIndex; ItemIndex; Item: TAdvCardItem; AFont: TFont; ABrush:
TBrush); // queries draw properties
    OnShowCardItem: TAdvCardItemAllowEvent;

    OnCardCaptionClick: TAdvCardEvent;
    OnCardCaptionDblClick: TAdvCardEvent;
    OnCardClick: TAdvCardEvent;
    OnCardItemClick: TAdvCardItemEvent;
    OnCardDblClick: TAdvCardEvent;
    OnColumnResizing(Sender: TObject; var NewSize: Integer);

    // all normal window control events
    OnKeyXXXXX
    OnMouseXXXXX
    OnEnter
    OnExit
    OnDragDrop
    OnDragOver
    OnStartDrag
  end;

  TAdvButtonClickEvent = procedure(Sender: TObject; ButtonIndex: integer; ButtonCaption: string) of object;

  TAdvBarAlignment = (baHorizontal, baVertical);
  TAdvBarButtonDirection = (bdVertical, bdHorizontal);

  TAdvButtonsBar = class(TCustomControl)
      BarAlignment: TAdvBarAlignment; default  baVertical; // used to control the alignment of the bar
      ButtonColor: TColor; // color of the buttons in the component
```

```
    ButtonDirection: TAdvBarButtonDirection;
    BorderWidth: integer; // set how wide is the border around the buttons
    ButtonGap: integer; // distance between buttons
    ButtonSize: integer; size of the button (height in vert. mode, width in horiz. Mode)
    CardList: TAdvCardList; // used to set the TAdvCardList component this buttons bar control
    ShowNumButton: Boolean; // show or hide the "123" button in the first position of the bar
    { events }
    OnButtonClick: TAdvButtonClickEvent;
  end;

implementation

end.
```

BEHAVIOR

1. All values display as static, if user select the item dynamic control for editing created.
2. If cards keep more space then card list width horizontal scroll box shows.
3. TAdvButtonBar browse cards by caption.

Note1 : number of buttons on the AdvButtonsBar is determined by height / (ButtonSize + ButtonGap)
if less buttons can be shown on the AdvButtonsBar than alphabet, letters should group, ie. For example:

Abc
def

Ghi

Etc..

Note2:

It should be able to use the keyboard with;

Arrow up/down left/right
Page Up / Page Down (4 items down/up)
Home / End (first / last item)
Letter of alphabet to go to first matching item

2.4 Assumptions and Dependencies

The final package is not stand alone application, it uses only in Delphi IDE.

Additional documents

Full list of properties and events will be provided with project proceeding.

Such documents will be provided with project release:

1. Test plan
2. Project directory tree
3. Test project

3. System Requirements

Windows & Delphi or C++Builder development environment