PTools Components ver. 1.2

Description

PTools is a set of components and experts that help the designer develop an application without the need for coding. The designer can use HyperTxt component to make a block of text with hyper words, HyperImage component to make a picture do an action, HyperArea to assign any area on the form to an action.

Some times a user needs to tag a place in the project, so that when it is re-started the user can jump directly to he/she left off. This component is called BookMark. It is used in conjunction with any of the previously mentioned components.

In addition, some multimedia components are provided to ease the implementation of a multimedia application. These components include WavePlay component to play an audio file, WavePause is a component used in conjunction with WavePlay so that the user can temporary pause a audio that is playing.

There is a special effect component called fxLabel. It allows the designer to create a shadow, raised or sunken label effect.

In addition, there are some management tools available to make it easier for the designer for complete the job. Like ProjView Expert to group forms. Or Default Component Settings Expert where the designer defines a default settings for the component.

Copyright and Disclaimer

Copyright
This software is developed by Waheed Al-Shahnan as a Masters project, under direction of Dr. Kumar Vadaparty, Computer Engineering and Science Department at Case Western Reserve University, who is currently at Bellcore, N.J. You may use this software for non-commercial applications, such as educational or for free distribution, as long as you have mentioned the software title and my name in the final application at an appropriate position, or in a readme file.

As for commercial applications, you can distribute the application that uses these tools only if you buy the source code and mention the following in your copyright section: "This product was partially developed using Panther Tools by Waheed Al-Sayer."

Also, Delphi is copyrighted to Borland International.

Disclaimer
No one is perfect. I will not be responsible for any damage that occurs by using this software. I will provide reasonable help in maintaining the software. Any comments or problems should be sent to the address mentioned in the Contact Information page.

Thanks and Dedications

I would like to thank Dr. Narasingarao Sreenath, Systems Engineering Department at Case Western Reserve University.
Also, my love, my life, my wife Maryam, whom I will be grateful for her patience and support trying to reach our goal.
I also would like to thank the people of Delphis NewsGroup on the internet, specially Ray Lischner of Tempest Software. And CompuServe Delphi Forum.
Finally, I dedicate these tools in my sons name.

New for this release

I have added a property to TWavePlay and events to THyperTxt components.

- WavePlay has a WorkDir property. If you want to include the wave files in a subdirectory you might have added the subdirectory name to the WaveName property, but now you can put the subdirectory name in a separate property.

- THyperTxt has been improved in three maners. Hyper word has a separate font property. Also, I have added two new events OnOverWord and OnWordClick. These two events will give programmers control over the behavior of the component if the built-in default actions are not enough.

Installation

Installation is divided into two parts

- Installing the complete set of components.
- Or, installing a single component.

Installing the complete set of components.

This is the preferred method, because you will make use of all the capabilities of the software. Some of the components are better installed as a set, like TWavePlay and TWavePause. In order to install the complete set you need to do the following steps:

1. Copy all files with the extension DCU, DFM, DCR in a single directory.
2. Start Delphi. In the Options menu, select Install Components item. A dialog with the list of installed components will shows.
3. Click on the Add button, a dialog will show.
4. In this dialog enter the PTOOLREG.DCU name, or use the browse button to find it.
5. Click on the OK button, then Delphi will start installing the components and the experts.
6. You are done. Now you can explore software using the sample project.

Installing a single component.

If you dont need all the components, or just want to check each one at a time, you can install a component singly. To install a single component follow the next steps for each component:

1. Do as mentioned in steps 1 to 3 above.
2. Then instead of entering PTOOLREG.DCU enter the file name corresponding to the component to be installed. The following table shows the component name and the file name containing the needed parts of that component.

Component	File to install	Additional Files needed
THyperTxt	HTxtReg	HyperTxt, MsgDlgU, StrTool
THyperImage	HImgReg	HyperImage, HImageEd, MsgDlgU, Action, ActDef, WavePly, StrTool
THyperArea	HAreaReg	HyprArea, HAreaEd, Action, ActDef, MsgDlgU, WavePly,
TWavePlay	WavePly	
TWavePause	WavePaus	WavePly
TfxLabel	fxLbl	
TBookMark	BookMark	

Expert	File to install
Component Settings	CompINI
Project Thumbnails	ProjView

***Note**: All files have DCU extention*

Contact information

By E-Mail:
102636.3002@compuserve.com

By Mail (till May 1996)
Waheed Al-Shahnan
30 Severance Cir. Apt 401
Cleveland Hts, OH 44118

or (after May 1996)

Waheed Al-Sayer
P.O.Box 7334,
Hawally, Kuwait 32093
KUWAIT

Future enhancements

Store audio file inside component
This could benefit or hurt the final application. Based on the response it might be implemented.
Store thumbnails in binary file
To avoid the problem of keeping forms open to see a thumbnail in the Project View dialog, we might store a binary representation of the image in a file. Thus requiring the form to be opened only once.
Develop project view as chapter view with page numbering and reorganization
Offer multiple book marks
The user would be able to place multiple marks on any frame. When the user returns to the application, the user will be presented
More special effects for the fxLabel

Well, you tell me what else.

THyperTxt Component

Properties Methods Events Tasks

Unit
HyperTxt

Description
HyperTxt is a special kind of text block with designer defined key Words. If the user wants to look up a certain word, he clicks on it and a pop-up window shows the description or play an audio file. The tool is a descendant of TPaintBox that has special properties.

In addition, the programmer can control the response to the clicks and the mouse shape when it is over the word.

Properties

About	AutoAdjust	Brush
Lines	TabWidth	Words
WordFont		

Methods

Create

Destroy

Events

OnWordClick OnOverWord

WordFont Property

Applies to
THyperTxt component

Declaration
```
property WordFont : TFont;
```

Description
This property is used to define the font of the hyperword. Adjusting the height of the word along with the height of the regular word was taken care of in the program.

OnOverWord Event

Applies to
THyperTxt component

Declaration
```
property OnOverWord: TOverWordEvent;
```

Description
The OnOverWord event occurs when the user moves the cursor over a hyper word. Usually you'll use an OnOverWord event to change the cursor programmatically.
Usually, you will want the cursor to change shape, indicating that the word will perform a certain action if the user clicks on it. You can change the shape of the cursor by changing the value of the Screen.Cursor property for the word at run time when an OnOverWord event occurs.

TOverWordEvent Type

Unit
HyperTxt component

Declaration
```
TOverWordEvent = procedure (Sender: TObject; WordOver: string) of object;
```

Description

The TOverWordEvent type points to a method that handles the movement of the cursor over a hyper word. The Sender is the object the cursor is being moved over, WordOver determines the word the mouse is over.

TOverWordEvent is the type of the OnOverWord event.

OnWordClick Event

Applies to
THyperTxt component

Declaration
```
property OnClickWord: TWordClickEvent;
```

Description
The OnWordClick event occurs when the user clicks on a hyper word. Usually you'll use an OnWordClick event to control the action a word would have.

This event overrides the built-in definition of word action.

TWordClickEvent Type

Unit
HyperTxt component

Declaration
```
TWordClickEvent = procedure (Sender: TObject; WordClicked: string) of
   object;
```

Description

The TWordClickEvent type points to a method that handles the response of a mouse click on a hyper word. The Sender is the object the mouse has clicked, WordClicked determines the word the mouse has clicked on.

TWordClickEvent is the type of the OnWordClick event.

TabWidth Property

Applies to
THyperTxt component

Declaration
```
property TabWidth : integer;
```

Description
This property is used to define the number of spaces replacing the TAB character in the text. In order to type in a TAB in the text, press the Control key when pressing the TAB key.

AutoAdjust Property

Applies to
THyperTxt component

Declaration
```
property AutoAdjust : TAdjustAttrs;
```

Description
This property is used to control the width and height of the component. It checks the maximum width and/or height of the component and changes the Width and/or Height property accordingly. These are the possible values:

Value	Meaning
adBoth	Adjust both the width and height of the component.
adHeight	Adjust only the height of the component based on the number of lines.
adNone	No adjustments is made.
adWidth	Adjust the width according to the length of the longest line.

Words Property

Applies to
THyperTxt component

Declaration
```
property Words : TWords;
```

Description
This property is used to define the list of hyper words and each word action. Similar to THyperArea and THyperImage, each word has an assiciated action that will be performed when the user clicks on the hyper word. To make use of this property look at the Tasks of THyperTxt

TWords Object

Unit

HyperTxt

Description
The TWords object contains definition to all the hyper words and each words action. TWord is a descendant of TStringList object.

TAdjustAttrs Type

Unit

HyperTxt

Declaration

```
TAdjustAttrs = (adBoth, adHeight, adWidth, adNone);
```

Description

The TAdjustAttrs type is the set of adjustment types that is used by AutoAdjust property. AutoAdjust property is used in HyperTxt component.

Using the HyperTxt Tasks

THyperTxt Reference

Purpose
The HyperTxt component provides a way to present text and link words in the text to certain actions. The designer adds the HyperTxt component from the component palette. Similar to any regular visual-component, HyperTxt can be resized and moved around. Once added the designer needs define the block of text in the Lines property, and in Words, the list of words that will show in a special way during program execution.

Tasks
To add lines of text to the component, double-click on the Lines property in the object inspector window. The a dialog box will appear where you can type in text.

Tasks for manipulating Hyper words:
To add hyper words to the component, double-click on the Words property. The designer populates Words property with words using the Word Selection Dialog.

Using the Word Selection Dialog
- To add words, select a word using the mouse or keyboard from Text box, then presses on the right arrow button to add the word to the list in the Words List list box.
- To remove a word from the list, select a word from Words list," and then clicks on the left arrow button. This will remove the word from the list, so it will not be a hyper word.

Defining an action for a word
After defining a list of words the designer defines an action for each one of the words previously selected using the Word Action Dialog. To show the possible action press on the Open Action button. A dialog will extend to show the possible actions.
- To assign an action to a word the designer selects one of the related action from the bottom half of the dialog. The response to the click on the word will either be: showing a definition, showing a frame, playing audio, or close the current frame. These actions are similar to those in HyperArea and HyperImage components.

TBookMark Component

Properties Methods Events

Unit
BookMark

Description
BookMark allows the user to read a book mark that was stored in the project INI file. The INI file name is the project name with .RUN extention. The component should only be placed on the main (starting) form. Once the application starts, the component will look under the [BookMark] section and find the name of the form that was marked.

The component will automatically be visible when there is a book mark. Otherwise, it will be invisible. The component has a Picture property so that the designer can specify a certain tag reminder.

Properties

About

Methods

Create Destroy

Events
None Exist

THyperImage Component

Properties Methods Events

Unit
HyperImg

Description
HyperImage is a component that allows the designer to define an action whenever the user click on the component. The component have a Picture property that hold an image. It also simulates a shadow, which can be show using the AllowShadow property. The designer can assign an action by using the Action property editor.
It is possible to drag and drop on the component a bitmap from the File Manager. This feature bypasses the Picture property editor and directly assigns the file to Picture property.

Properties

About Action AllowShadow CursorShow

Methods

Create Destroy

Events
None Exist

AllowShadow Property

Applies to
THyperImage component

Declaration
```
property AllowShadow : boolean;
```

Description
This property is used to specify whether to allow to simulate a shadow for the image.

THyperArea Component

Properties Methods Events

Unit
HyprArea

Description
HyperArea is a component that allows the designer to define an action whenever the user click on the component. The component is only visible at design-time, thus allowing the designer to use this component over images to define hot areas. The designer can assign an action by using the Action property editor. Unlike HyperImage, THyperArea does not have a picture property.

Properties

About	Action	CursorShow

Methods

Create Destroy

Events
None Exist

Action Property

Applies to
THyperArea, THyperImage components

Declaration
```
property Action : TAction;
```

Description
This property is used to define the kind of action that will be performed when the user clicks on the component. These are the possible values:

Value	Meaning
acFrame	Display another form, which will hide the current form.
acExecute	Execute a program.
acWavePlay	Play a wave file using a WavePlay component.
acDesc	Display a line of text.
acBookMark	Place a bookmark on the current form.
acClose	Close the current form. If the form was the main form it will terminate the application.
acNone	Do nothing.

TAction Object

Unit

Action

Description

The TAction object contains definition to the kind of action to be taken in THyperArea and THyperImage component. TAction consists of subproperties as follows:

Property	Type	Description
ActionType	TActionAttrs	Defines the action type.
Frame	string	The name of the form that will be shown.
Desc	string	The line of text that will be displayed in a message box.
FileToRun	string	The full path name of the file to run.
Param	string	The parameters passed to the file to be run.
WorkDir	string	The working directory of the file to run.
WaveComp	string	The name of the TWavePlay component that contains the audio file.

CursorShow Property

Applies to
THyperArea, THyperImage component

Declaration
```
property CursorShow : boolean;
```

Description
The CursorShow property specifies whether the cursor will change to a Hand whenever the mouse is over the component. It is better to set the property to True in case you are using the HyperArea component, because the component is invisible at run-time.

TActionAttrs Type

Unit

Action

Declaration

```
TActionAttrs = (acFrame, acExecute, acWavePlay, acDesc, acBookMark, acClose,
                acNone);
```

Description

The TActionAttrs type is the set of actions types that is used by Action property. Action property is used in HyperArea and HyperImage components.

TWavePlay Component

Properties Methods Events

Unit
WavePly

Description
WavePlay is component that play wave files. The designer can assign a wave file to the component through the WaveName property. Components like HyperImage, HyperTxt create an instance of TWavePlay to enable the designer to hear the wave file before selecting that component.

The wave file will be played when the program invokes the Play method.

There are no events generated by the component.

Properties

About	Options	WaveName	WorkDir

Methods

Create	Destroy	Open	Status	CloseWave	Play	Pause	Resume

Events
None Exist

About Property

Applies to
TWavePlay, TWavePause, THyperTxt, THyperArea, TfxLabel, TBookMark

Declaration
```
property About : TAboutPTools;
```

Description
Design-time and read only. When double clicked shows a copyright dialog with some information.

Options Property

Applies to
TWavePlay component

Declaration
```
property Options : TWavePlayOptions;
```

Description
This property is used to specify the options that will control some aspects of the component. These are the possible values:

Value	Meaning
wvWaitTillDone	Playing the audio file will hold Windows response till the file is done playing.
wvDefault	If the specified wave file is not found, the default Windows audio is played.
wvAutoPlay	Once this component is created it will play the wave file.

TWavePlayOptions Type

Unit

WavePly

Declaration

```
TWavePlayOption = (wvWaitTillDone, wvDefault, wvAutoPlay);
TWavePlayOptions = set of TWavePlayOption;
```

Description

The TWavePlayOptions type is the set of behaviors the WavePlay object can assume. It defines Options property.

WaveName Property

Applies to
TWavePlay component

Declaration
```
property WaveName : string;
```

Description
Use this property to designate which audio file to Play. If a full path is not specified, the current search path is used.

NOTE: It is advised to keep the wave files in the same directory as the EXE file.

WorkDir Property

Applies to
TWavePlay component

Declaration
```
property WorkDir : string;
```

Description
Use this property to designate a directory where youll keep audio files at. If the directory is a subdirectory of the project, the name should not begin with a back-slash. The subdirectory name followed by a back-slash is all that is needed.

Open Method

Applies to
TWavePlay component

Declaration
```
procedure Open;
```

Description
This method will open the wave file with a unique alias. If it was successful in opening the wave file, the file is ready to be played by the Play method.

Play Method

Applies to
TWavePlay component

Declaration
```
procedure Play;
```

Description
This method will play the wave file which was opened using the Open method. If the file was not opened it will be opened automatically.

Pause Method

Applies to
TWavePlay component

Declaration
```
procedure Pause;
```

Description
This method will pause the wave file being played using the Play method. If the file was not playing this method has no effect.

Rewind Method

Applies to
TWavePlay component

Declaration
```
procedure Rewind;
```

Description
This method will rewind the wave file.

Status Method

Applies to
TWavePlay component

Declaration
```
function Status: string;
```

Description
This method will return a string indicating the status of the component. The string returned is just what mci status command will return.

Resume Method

Applies to
TWavePlay component

Declaration
```
procedure Resume;
```

Description
This method will resume a paused wave file. The device should have been paused using the Pause method. If the file was not playing this method has no effect.

CloseWave Method

Applies to
TWavePlay component

Declaration
```
procedure CloseWave;
```

Description
This method will close the wave file, thus freeing a wave device.

TWavePause Component

Properties Methods Events

Unit
WavePaus

Description
WavePause is component that pauses a playing wave file. The WavePause component is a descendant of SpeedButton. The designer places the button on a form which has a **long** playing wave file. It checks through playing WavePlay components then it will pause call the WavePlay component Pause method.

Once the audio is paused, the button image will change from a pause image to a play image. Simiraly, once the user clicks on the button the audio file will be continued and the image will change from a play to a pause.

Properties

About

Methods

Events
None Exist

TfxLabel Component

Properties Methods Events Tasks

Unit
fxLbl

Description
fxLabel is a special kind of Label component. It is used only for decoration practices, that all there is to it.

Properties

About	EffectType	LightColor
ShadowDistance	ShadowColor	

Methods

Create Destroy

Events
None Exist

ShadowColor Property

Applies to
TfxLabel component

Declaration
```
property ShadowColor : TColor;
```

Description
This property is used to define the color of the shadow of the label.

ShadowDistance Property

Applies to
TfxLabel component

Declaration
```
property ShadowDistance : integer;
```

Description
This property is used to define the distance between the text and its shadow.

LightColor Property

Applies to
TfxLabel component

```
property LightColor : TColor;
```

Description
This property is used to define the color of the lighting, which is usaully white and from upper left.

EffectType Property

Applies to
TfxLabel component

Declaration
```
property EffectType : TEffectTypes;
```

Description
This property is used to define the style in which the visual effect will be. Visual effects, such as Shadow or Raized text can be done by just changing this property. These are the possible values:

Value	Meaning
etCustom	Not a default visual effect.
etDeepSink	Deep sink visual effect.
etRaize	Makes the label looks raized.
etHighRaise	Makes the label looks raized, but with more intence light and shadow.
etShadow	Make the label looks like it has a shadow.
etHighShadow	Make the label looks like it has a shadow, but with more intence light and shadow.
etSink	Makes the label looks as sink or digged.

TEffectTypes Type

Unit

fxLbl

Declaration

```
TEffectTypes = (etCustom, etDeepSink, etRaise, etHighRaise, etShadow,
                etHighShadow, etSink);
```

Description

The TEffectTyps type is the set of available visual effects that is used by EffectType property in TfxLabel component.

Using the fxLabel

TfxLabel Reference

Purpose
The fxLabel component provides only a visual effect of displaying text. The designer adds the fxLabel component from the component palette. Similar to Label component, the designer sets the text using the Caption property. Once added the designer changes the EffectType and Shadow or Light features.

Tasks
To change the label text, click on the Caption property and directly type the text in the object inpector.

Tasks for changing the visual effect:

Adding a shadow
- To make the text appear as if it has shadow, change the EffectType to etShadow or etHighShadow.
- Change the ShadowColor property to match a darker than the background color.

Project View Expert

Description
Project View Expert provides the designer a tool to organize the forms into groups. There is an *important note*, in order to get a thump nail veiw of a form, that form must be opened. This is a restriction that will be addressed in future versions. There are certain tasks in using the Expert, these are:

- Starting the Expert.
- Locating forms previously created.
- Creating New groups.
- Deleting a group.
- Moving a form between groups.
- Finding Newly created forms.
- Reorganizing forms in a group.
- Using the Refresh button.

Tasks

Starting the Expert
Delphi defines new Experts to be in the Help menu. So, to start the expert, under the Help menu youll find an item named Project View Expert. Choose that item, and after a few seconds, depending on the number of forms and your machines speed, youll see the Project View dialog box.

Locating forms previously created the first time
When you start the expert at the first time, all forms previously created will be at the Not Filed group.

Creating New groups
To create a new group, type the name of the group in the edit box above the group list. Then press the **Add Group** button.

Deleting a group
To delete a group, select it, then press on the **Delete Group** button. All form that were in that group will not be deleted, they will be moved to the Not Filed group.

Moving a form between groups
To move a form from one group to another, follow these steps:
Select the from group, a list of forms in that group will show to its right.
Drag the form to be moved from the list and **drop** it over the group you want it to be moved to.

Finding Newly created forms
All forms that are created after the initial invocation of the Expert will be located in the Not Filed group.

Reorganizing forms in a group
You can organize forms in the same group by dragging and dropping the form before the current form. This has no effect on any operation.

Using the Refresh button
The Expert dialog box is shown modal. Which means that you can work on your project while the dialog is shown. Therfore, when you create a form, delete a form, change the cotents of a form; you can press the Refresh button to re-read the forms in the groups.

Component Settings Expert

Description
Component Settings Expert provides the designer a tool to specify initial settings for the component when it is first created. The expert stores those settings in an INI file in the Windows directory by the name PTools.INI. There are certain tasks in using the Expert, these are:

- Starting the Expert.
- Changing a component initial settings.

Tasks

Starting the Expert
Delphi defines new Experts to be in the Help menu. So, to start the expert, under the Help menu youll find an item named Component Settings Expert. Choose that item, and after a few seconds, depending on the number of forms and your machines speed, youll see the Component Settings dialog box.

Changing a component initial settings
Once you start the Expert, you can select the component name from the tabs at the bottom of the dialog. Each component page lists only part of the properties. You can change that property so that the next time you drop a component, the component will read those properties and set the component accordingly.