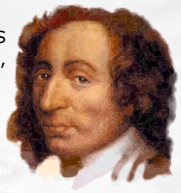
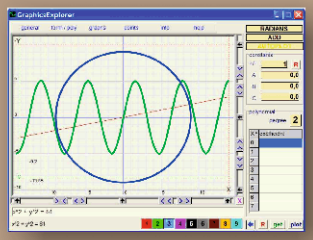
FOR DELPHI, LAZARUS, AND PASCAL RELATED LANGUAGES
PAS2JS / WEB APPS, INTERNET, ANDROID, IOS, MAC,
WINDOWS & LINUX

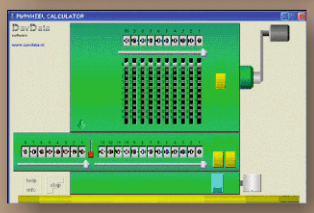

Blaise Pascal

BLAISE PASCAL MAGAZINE 75/76

TECTONICS game By David Dirkse
COCOA Lazarus and future development on Apple platforms By Dmitry Boyarintsev
Creating a subversioning system for DELPHI and LAZARUS By Detlef Overbeek
LAZARUS 2.0 NEWS AND OVERVIEW WITH DETAILS By Detlef Overbeek
How to use PAS2JS under Lazarus By Detlef Overbeek
IDE Scout for Lazarus 2.0 A new intelligent feature: Created by Michael van Canneyt
Video Effects: mix, split and merge, videos, implement Picture in Picture effect and how to perform video transitions: By Boain Mitov
REST easy with kbmMW: Authorization and login management #14 – DB Controlled login By Kim Madsen
REST easy with kbmMW #15 – Handling HTTP POST By Kim Madsen
Sense and Nonsense of AI: Artificial Intelligence explained By Detlef Overbeek
The PowerPDF experience By Marcel HorstHuis

DAVID DIRKSE





```
procedure ;
var
begin
 for i := 1 to 9
do
  begin
```

GRAPHICS COMPUTER MATH & GAMES IN PASCAL



POCKET EDITION

Printed in full color.
A fully indexed PDF book is included + 52 projects

CREDITCARD LIBRARY STICK 16 GB

All issues 1-76on the USB stick complete
searchable 4300 pages -fully indexed including all code



COMBINATION: 3 FOR 1

BOOK INCLUDING THE LIBRARY STICK EXCL. SHIPPING
INCLUDING 1YEAR DOWNLOAD FOR FREE
GET THE BOOK INCLUDING THE NEWEST LIBRARY STICK
INCLUDING 1 YEAR DOWNLOAD OF BLAISE PASCAL MAGAZINE

€ 100

https://www.blaisepascalmagazine.eu/product-category/special-offer/

BLAISE PASCAL MAGAZINE 75/76

FOR DELPHI, LAZARUS, AND PASCAL RELATED LANGUAGES
PAS2JS / WEB APPS, INTERNET, ANDROID, IOS, MAC,
WINDOWS & LINUX



Content

Articles

TECTONICS game
By David Dirkse — **Page 6**

COCOA Lazarus and future development on Apple platforms
By Dmitry Boyarintsev — **Page 13**

Creating a subversioning system for DELPHI and LAZARUS
By Detlef Overbeek — **Page 16**

LAZARUS 2.0 NEWS AND OVERVIEW WITH DETAILS
By Detlef Overbeek — **Page 42**

How to use PAS2JS under Lazarus
By Detlef Overbeek — **Page 44**

IDE Scout for Lazarus 2.0 A new intelligent feature:
Created by Michael van Canneyt — **Page 57**

Video Effects: mix, split and merge, videos, implement Picture in Picture effect and how to perform video transitions:
By Boain Mitov — **Page 62**

REST easy with kbmMW: Authorization and login management #14 – DB Controlled login
By Kim Madsen — **Page 94**

REST easy with kbmMW #15 – Handling HTTP POST
by Kim Madsen — **Page 99**

Sense and Nonsense of AI: Artificial Intelligence explained
By Detlef Overbeek — **Page 102**

The PowerPDF experience
By Marcel HorstHuis — **Page 107**

iss056e162819 (Sept. 14, 2018) --- Hurricane Florence is pictured from the International Space Station as a category 1 storm as it was making landfall near Wrightsville Beach, North Carolina.

ADVERTISERS
Barnsten — Page 93
Books — Page 2
LibStick — Page 5
Mitov Software — Page 60
Components 4 Developers — Page 120



Pascal is an imperative and procedural programming language, which Niklaus Wirth designed in 1968–69 and published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. A derivative known as Object Pascal designed for object-oriented programming was developed in 1985. The language name was chosen to honour the Mathematician, Inventor of the first calculator: Blaise Pascal (see top right).

Publisher: PRO PASCAL FOUNDATION
in collaboration with the Pascal User Group
© Stichting Ondersteuning Programmeertaal Pascal

Stephen Ball
http://delphiaball.co.uk
@DelphiABall

Miguel Bebensee
mbebensee@ibexpert.biz
http://devstructor.com

Peter Bijlsma -Editor
peter @ blaisepascal.eu

Dmitry Boyarintsev
dmitry.living @ gmail.com

Michaël Van Canneyt,
michael @ freepascal.org

Marco Cantù
www.marcocantu.com
marco.cantu @ gmail.com

David Dirkse
www.davdata.nl
E-mail: David @ davdata.nl

Benno Evers
b.evers
@ everscustomtechnology.nl

Bruno Fierens
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

Holger Flick
holger@flixments.com

Primož Gabrijelčič
www.primoz @ gabrijelcic.org

Mattias Gärtner
nc-gaertnma@netcologne.de

Peter Johnson
http://delphidabbler.com
delphidabbler@gmail.com

Max Kleiner
www.softwareschule.ch
max @ kleiner.com

John Kuiper
john_kuiper @ kpnmail.nl

Wagner R. Landgraf
wagner @ tmssoftware.com

Vsevolod Leonov
vsevolod.leonov@mail.ru

Andrea Magni www.andreamagni.eu
andrea.magni @ gmail.com
www.andreamagni.eu/wp

Paul Nauta PLM Solution Architect
CyberNautics
paul.nauta@cybernautics.nl

Kim Madsen
www.component4developers

Boian Mitov
mitov @ mitov.com

Jeremy North
jeremy.north @ gmail.com

Detlef Overbeek - Editor in Chief
www.blaisepascal.eu
editor @ blaisepascal.eu

Howard Page Clark
hdpc @ talktalk.net

Heiko Rompel
info@rompelsoft.de

Wim Van Ingen Schenau -Editor
wisone @ xs4all.nl

Peter van der Sman
sman @ prisman.nl

Rik Smit
rik @ blaisepascal.eu
www.romplesoft.de

Bob Swart
www.eBob42.com
Bob @ eBob42.com

B.J. Rao
contact@intricad.com

Daniele Teti
www.danieleteti.it
d.teti @ bittime.it

Anton Vogelaar
ajv @ vogelaar-electronics.com

Siegfried Zuhr
siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: +31 (0)30 890.66.44 / Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to editor@blaisepascal.eu

Editors
Peter Bijlsma, W. (Wim) van Ingen Schenau, Rik Smit

Correctors
Howard Page-Clark, Peter Bijlsma

Trademarks All trademarks used are acknowledged as the property of their respective owners.
Caveat Whilst we endeavour to ensure that what is published in the magazine is correct, we cannot accept responsibility for any errors or omissions. If you notice something which may be incorrect, please contact the Editor and we will publish a correction where relevant.

Subscriptions (2017 prices)

	Internat. excl. VAT	Internat. incl. VAT	Including Shipment
Printed Issue ±80 pages	€ 235	€ 266,50	€ 85,00
Electronic Download Issue 80 pages	€ 50	€ 60,50	———
Printed Issue Inside Holland(Netherlands) ±80 pages	———	€ 180	€ 30,00

Member and donator of WIKIPEDIA

Subscriptions can be taken out online at www.blaisepascal.eu or by written order, or by sending an email to office@blaisepascal.eu
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
Subscriptions run 365 days. Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to:
ABN AMRO Bank Account no. 44 19 60 863 or by credit card or Paypal
Name: Pro Pascal Foundation-Foundation for Supporting the Pascal Programming Language (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Programmeertaal Pascal)
Subscription department
Edelstenenbaan 21 / 3402 XA IJsselstein, The Netherlands
Mobile: + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Copyright notice

All material published in Blaise Pascal is copyright © SOPP Stichting Ondersteuning Programeertaal Pascal unless otherwise noted and may not be copied, distributed or republished without written permission. Authors agree that code associated with their articles will be made available to subscribers after publication by placing it on the website of the PGG for download, and that articles and code will be placed on distributable data storage media. Use of program listings by subscribers for research and study purposes is allowed, but not for commercial purposes. Commercial use of program listings and code is prohibited without the written permission of the author.

From the editor

This time it will be a short story. There are so many things happened since the last issue that it is sheer impossible to make an overview of all the things:
We created a new Version of Lazarus 2.0.
I personally have been working on it and on the Lazarus Handbook that hopefully soon will become available - and also did the layout of this issue. It is 120 pages.
I created as well a lot of the articles, some really took quite some time because I always have a learning curve: I need to read the books, other articles, try to make sure I am not writing nonsense and be aware of FAKE news. Don't we all?
So I decided to keep this editorial short because on Wednesday 14th November I need to go to Paris and Thursday 15th I' ll be back again. But before that, I simply wanted to finish this issue, because I know a lot of you are waiting for it.... And it will be published in Paris

Last weekend I spend on writing the article about AI - artificial intelligence.
Everybody talks about it, a lot of people do not even understand the basic facts.
So I thought it might be helpful if I give you a hint and wrote an article about it.
I tried to make it easy to understand because I think this is something big. It will not ever more go away.
Please let me know if you would like to read more about it. For now I have not gone into technical details, only briefly. Maybe you could give me some idea about what you expect and what would be your interest.

Detlef



All issues 1-76 on the USB stick complete searchable 4300 pages -fully indexed including all code



Including 1 year subscription
Only € 75 ex vat
ex shipping € 5

Schools and students can get a free subscription:
Simply ask at: office@blaisepascal.eu



INTRODUCTION AND DATA STRUCTURES

Tectonic is a number puzzle. It is also called Suguru. Below are examples of an original puzzle (left) and the solution (right).

Tectonic puzzles come in different sizes, the example above has 6 rows and 6 columns.

These 36 fields are grouped as a single- or as 2, 3, 4 or 5 fields. Fields have fat borders.

In the solved puzzle each group must contain the numbers from 1 (2,3,4,5 up to it's group size) just once. So, a field in a group of one must hold the number 1. Neighbouring (horizontal, vertical or diagonal) fields must also have different numbers. In the original puzzle some fields already contain a number (painted brown).

The player has to find the numbers in the empty fields by using logical deduction.

					3
5					
					4
		3			
				2	3

Original

1	4	1	2	1	3
5	2	3	4	5	2
3	4	1	2	3	4
1	2	3	4	5	1
3	4	5	1	2	3
2	1	2	4	5	1

Solved

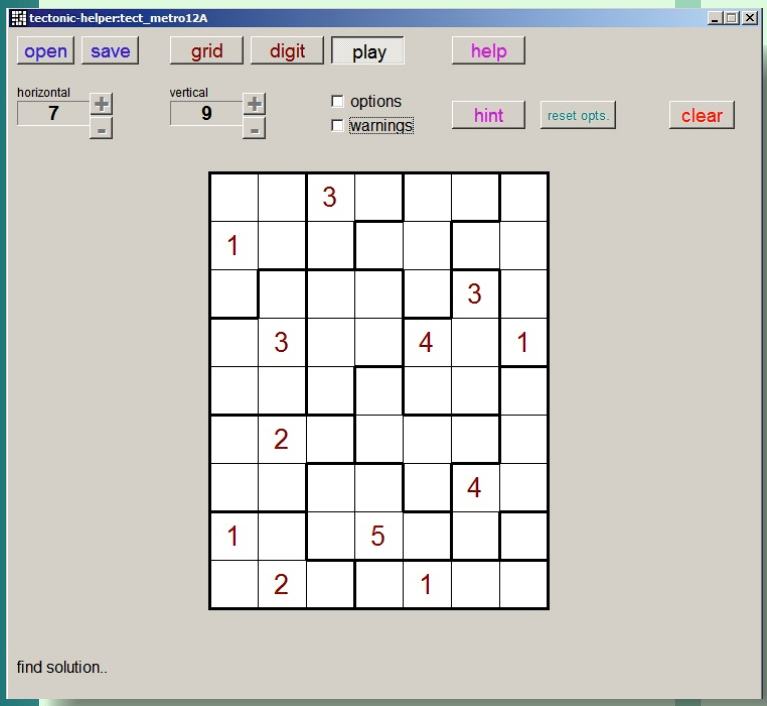
When writing a program, the first step is to define what we want. In this case we want assistance in solving Tectonic puzzles.

This includes:

- using the computer screen as a replacement for pencil and paper
- displaying field options for a quick overview
- supply warnings
- provide hints
- save and load games

It is our intention to increase knowledge. Therefore no direct solutions are calculated because this adds nothing to our undertanding. Instead hints are given which explain the single steps towards a solution.

First a look at the Tectonic program at work:



The menu buttons are at the top of the form.

- **Open** and **Save** are **BitButtons** to save or load puzzles from disc.
- **Grid / Digit / Play** are **Speedbuttons** with common group number 1.
- **Grid** selects the painting of fat lines in a new puzzle.
- **Digit** enables adding original (brown) numbers in a new puzzle.
- **Play** enables adding black (player) numbers or removal of redundant options.
- **Help** opens on-line help information.
- The **options-** and **warning** checkboxes provide option display or check for fields with missing options which inhibits a solution.
- The **Hint BitButton** generates four types of hints during the play time.
- **Reset options** recalculates all options.
- The **Clear BitButton** action depends on the grid/digit/play buttons.
- **Grid:** erase all lines drawn.
- **Digit** : erase all brown (original) numbers.
- **Play** : if any: erase all error information, else erase (black) player numbers.

At the bottom of the form a **Static Text** is placed to display messages.

The description of the Tectonic program comes in four parts

1. introduction and data structures
2. drawing
3. game control
4. play- and hint procedures

DATA STRUCTURES
Programming is writing procedures to manipulate data.
First a field is defined: (see game_unit)

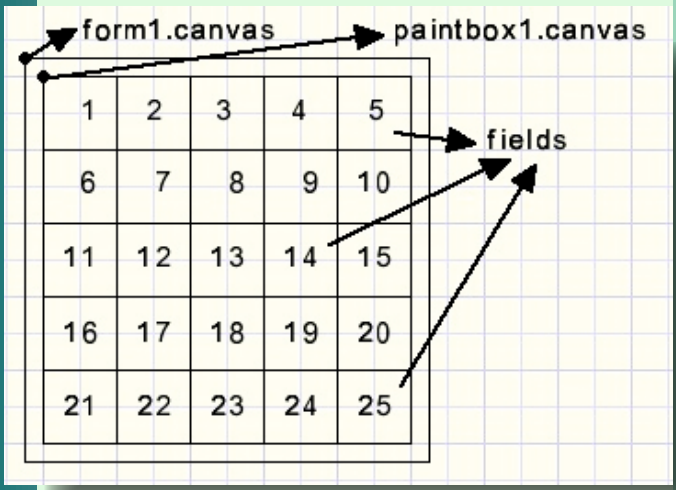
```
type TnrType=(ntNone,ntOrg,ntPlay);
  TField=record
       nrType :TnrType;
       nr   :byte;
       options:byte;
       groupNr:byte;
       grid  :byte;
       marked :boolean;
      end;
```

ntOrg : original (brown) number. 0 if none.
NtPlay : player (black) number, 0 if none.
Options : bit n set for option n present.
GroupNr : 1,2,3.... the group to which field belongs
grid : bit 1..4 set if left, top, right, bottom edge is group boundary.
Marked : true if error marker is drawn in field as warning)

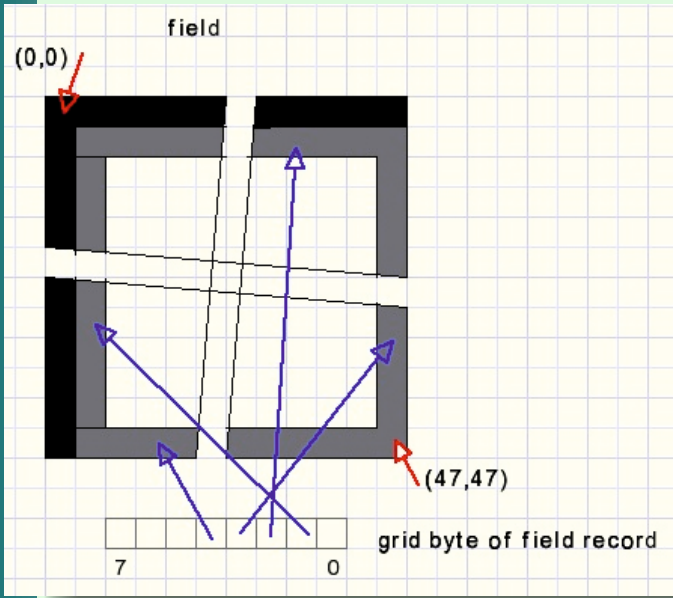
```
const grouplimit=75;
var HSize:byte=15;
  VSize:byte=10;
  field:array[1..150]of TField;
  maxfield:byte=150;
  maxGroup:byte;
  groupSize:array[1..grouplimit]of byte;
  groupMember:
     array[1..5,1..grouplimit]of byte;
     //group fields
```

Hsize: horizontal size. (fields)
Vsize: vertical size. (fields)
For hints and warnings:
Groupsize array: the number of fields in each group
GroupMember array: fields listed per group.

DRAWING
All drawing is done directly in Paintbox1 on Form1.
A field is 48*48 pixels in size.
Horizontal and vertical black lines of 1 pixel wide run through (0,0) coordinates of each field. Picture below shows the field numbers in a 5x5 puzzle.

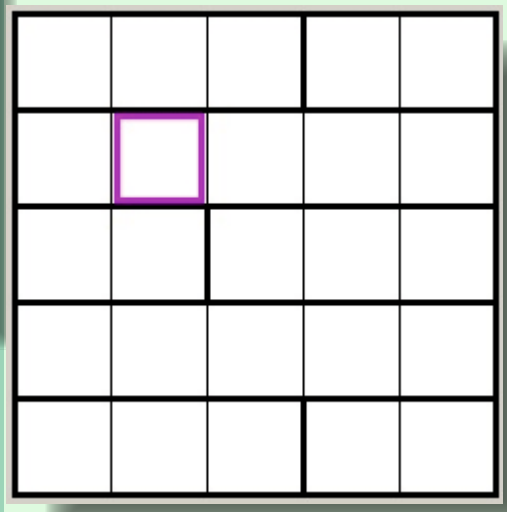


The paintbox lies within a rectangle painted on the canvas of form1.
To mark group boundaries, within a field lines are drawn through coordinates (1,1) , (47,1) ,(1,47) and (47,47).
See the picture below showing an enlarged field and grid bits assignment.
If the grid bit is -1- , the line is drawn.



So groups are enclosed within fat lines of 3 pixels wide.
On occasion a canvas needs repainting because the pixels may be overwritten.
It is convenient to link the line painting on the form canvas to the paintbox1 paint event because repainting of form1 causes flickering of the complete form including the menu buttons. Invalidating paintbox1 also must repaint the lines on the form1 canvas.

To enter a number a field has to be selected. This is done by the marker, a purple rectangle painted within the field, line width = 3 pixels:



The following functions and procedures are needed for above painting: (see unit1 for details)

`function XY2fieldNr(x,y:smallInt): byte;`
x,y are mouse coordinates of paintbox1. The function returns the field number of the (x,y) coordinates.

`procedure fieldNr2XY(var x,y : smallInt; n : byte);`
n is a field number, the left top coordinates of field n are calculated in x,y.

`procedure initGridEdges;`
Sets grid values in field[..].grid for left,top,right,bottom fields. Does not paint.

`procedure paintformEdges(col : Tcolor);`
paints a 1 pixel wide line on the canvas of form1, just around paintbox1. If form1.color is provided as color, the line actually is erased.

`procedure paintgameGrid;`
Paint game with empty fields, no groups

`procedure paintFieldEdges(f : byte);`
paints edges for group marking of field f

procedure paintformEdges(col : Tcolor);
paints a 1 pixel wide line on the canvas of form1, just around paintbox1. If form1.color is provided as color, the line actually is erased.

procedure paintgameGrid;
Paint game with empty fields, no groups

procedure paintFieldEdges(f : byte);
paints edges for group marking of field f

procedure paintfield(f : byte);
paint field f. This procedure calls helpers, depending on the field values:

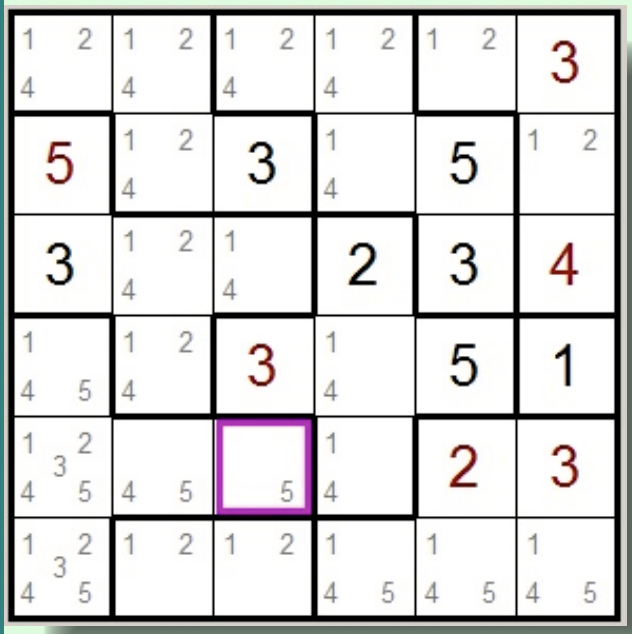
procedure paintEmptyField(f : byte);
paints empty field but does not touch the lines for group marking.

procedure paintfieldNr(f : byte);
paint the number in field f (field[..].nr colored brown or black

procedure paintfieldOptions(f : byte);
paints the options in field f (field[..].options

procedure paintMarker(f : byte; switch : boolean);
paint the marker in field f. If switch is true: purple, if switch is off : white (erase)

procedure setmarker(n : byte);
write marker in field n.
If marker variable <> 0 : first clear the old marker by calling the paintmarker procedure with switch = false;
So, the onPaint event for paintbox1 calls all procedures mentioned before.
Below is a picture with fields holding numbers, options and the marker pointing to a field with just one option.



All actions are initiated by events, mainly from mouse and keyboard.
But mouse- and keyboard event actions depend on the circumstances.
Here we have 3 game states:

painting the groups
adding some original (brown) numbers
searching for solutions by adding (black) numbers

GROUP PAINTING
When the grid menubutton is down a fat edge is painted when the mouse pointer is moved over the edge and the left mousebutton is pressed down.
If the right mousebutton is down, the fat line is erased.

So, we need mouse-down, mouse-move and mouse-up events.
On mouse-down the mousebutton is examined. If left, a variable called paintmode is set to pmPaint else paintmode is set to pmErase.

```
type TPaintMode=
      (pmNone,pmPaint,pmErase);
var paintMode:TpaintMode;
```

On **mouse-move** and **(paintmode <> pmNone)**
procedure setEdge(x,y : smallInt);
is called where (x,y) are paintbox1 coordinates.
This procedure calls
procedure paintEdgeData(f,m : byte);
to update the edge bits m in field[f].grid
which calls **paintFieldEdges(f : byte);**
to paint the lines in field f.

The group lines at the left, top, right and bottom of the puzzle may not be altered.
This is simply done by checking for x,y > 10 and x,y < width-12,height-12.(paintbox1)

The option display uses preset constants for the position of the options relative to the field:

```
const hop:array[1..5]of byte=(5,32,20,5,32);
  vop:array[1..5]of byte=(5,5,16,28,28);
```

hop is for the x-offset, vop for the y-offset fo options 1..5.

GAME CONTROL

```
type TGameStatus =
(gsGrid,gsDigit,gsPlay,gsEnd);
var gameStatus:TgameStatus;
```

gsGrid: if grid menu button is down: resize puzzle, paint the grid (groups)
gsDigit: if digit button is down : add numbers .
GsPlay: if play button is down: enter numbers to solve puzzle.

RESIZING

Resizing is done by pressing the + and – horizontal and vertical speed buttons. These buttons share the mouse-down and mous-up events.
The tag property (1,2,3,4) is used to identify a button.
On mouse-down a timer is started and variable timercode is set to the tag value.
On time out, the timer event method uses the timercode to increment or decrement the horizontal or vertical puzzle dimensions.
Then initGridEdges and setPaintbox1Dimensions are called.
SetPaintbox1dimensions sets width, height , left and top properties of paintbox1 and calls the invalidate method for repainting.
A mouse-up event from the +/. speedbuttons disables the timer.
So, we may resize the puzzle by holding a +/- speedbutton down. This is more convenient then clicking repeatingly.

MENU BUTTONS

These buttons control all events from mouse and keybord.
Instead of handling game control by the event methods a central procedure is used.
An onClick event from the grid/digits/play buttons generates a message to

```
type TGameMessage=
(gmGridBtn,gmDigitBtn,gmPlayBtn,gmClearBtn,gmDelete,gmEnd);
```

```
procedure GameControl(gm : TgameMessage);
```

The gmClearBtn message is from the clear button, gmDelete from the delete key and gmEnd comes from the procedure that checks for a solved puzzle.
Procedure gameControl calls

```
procedure setGameStatus(gs:TgameStatus);
```

to generate messages, enable/disable buttons, to set or clear the marker.

Changing game status from gsGrid to gsDigits is allowed only if all groups are drawn properly which includes:

1. the maximal number of groups is 75
2. groups have no more than 5 fields.

A check is made by

```
function builtGroups : boolean;
```

which returns true if the check was OK.

BuiltGroups calls

```
function initGroups:byte; // (game_unit)
```

which returns <> 0 in case of errors.
This function fills the groupsize[..] and groupMember[.. ,..] arrays. (game_unit)
So now we know

1. the number of fields in each group
2. the fields that make each group

To gather all fields in a group a search over te fields has to be made.
This is done by

```
procedure fillGroup(f,grp : byte);
```

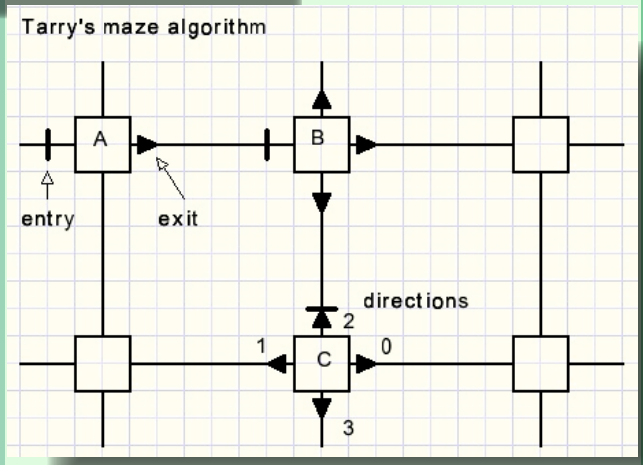
which uses the maze escape algorithm of Tarry.
This procedure sets the groupNr (grp) value for each field. The search starts at field f.

Maze algorithm was originally designed to find an escape route from a maze. The algorithm was designed in 1895. Mazes were very popular at that time.
It provides a systematic way to traverse a maze without leaving any part unvisited. When no exit exists the visitor returns at the starting location.

A maze has roads connecting squares where roads split / join and a decision must be made which direction to go.
Two signs are needed:
1. an entry sign to mark the road into a new square
2. an exit mark for a road to another square.

These are the rules:
1. never take a road twice in the same direction
2. only take the entry road back to the previous square when all exit roads have been traversed

See picture below:



The entry into square A is marked. The first exit to B is taken and the exit is marked. Etc.
In this application the squares are the puzzle fields and the roads are horizontal and vertical connections. There is no exit so after the search we return to the field where the search started.

The fillgroup(...) procedure uses this variables:

```
var f1:byte;//the field where the search started
  game:array[1..150]of byte;//remember entry direction 0..3
  dir:byte;//currect direction of movement
```

This procedure places the group numbers in de field[f].groupNr .
Field[f].grid is examined for existing roads.
Note that the opposite direction is obtained by dir xor 1.

CHECKS
Pressing the play menu button causes some checks to be made:
1. all empty fields should have at least one option
2. all fields of a group together must have all options.

Note: a field with number n has one option bit set : (1 shl n).

`function testGameOK : word;// game_unit`
does the job. The result is a non zero error code if options are missing.
During play time this function is called only when checkbox warningbox is checked.
Also during play time
`function checkEnd : boolean;`
returns true if the puzzle is solved (all numbers filled in).

PLAYING AND HINT PROCEDURES
When entering a number in a field
1. this option must be removed from the fields of the same group
2. this option must be removed from neighbour fields in other groups

`procedure enterDigit(f,n : byte; tp : TnrType);`
enters digit n in field f where tp = ntPlay if in play time.

`procedure dropFieldOption(f : byte);`
is called to drop the option of field f in all neighbour options.

`function neighbourfields(a,b : byte) : boolean;`
returns true if fields a and b are neighbours
There are four types of hints which are examined when the hint BitButton is pressed.

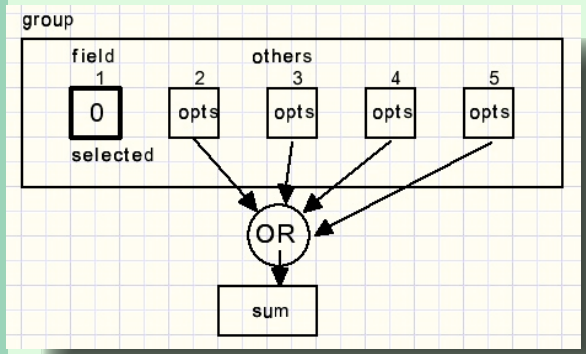
HINT 1
Check for a field without number and only one option.

```
Type THint1=record
      fld:byte;
      nr :byte;
     end;
```

HINT 2
Check for a field that holds the only option for this group.

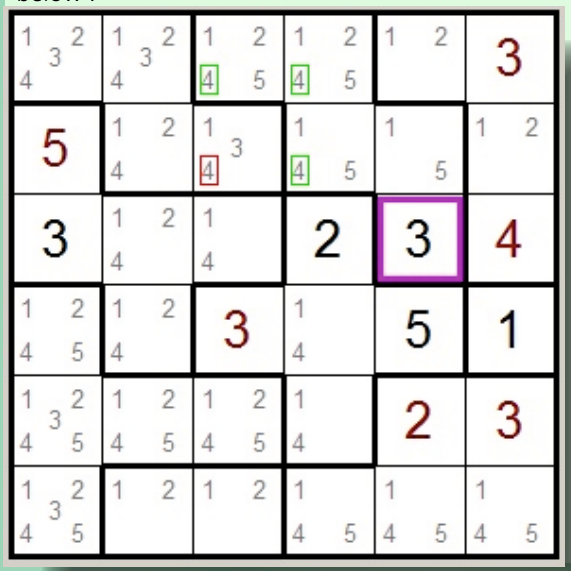
```
Type THint2=record
      fld:byte;
      nr :byte;
     end;
```

(sorry, same type as THint1)
`function procHint2(var h : THint2) : boolean;`
This is a somewhat lengthy routine containing two nested repeat statements and a for loop.
The outer repeat loop steps through the groups. The inner repeat loop selects a zero number field from the group. Then the for loop ORs all options of the other group fields, see picture:



`function popcount(m : byte) : byte;`
returns the number of -1- bits in byte m. If this number is equal to the grouplength -1 then fields 2..5 are missing 1 option, which must be present in field 1. So, hint2 checks for a single option in a field that is missing in the other group fields.

HINT3
Hint3 is only examined when the options display is selected. This becomes more difficult. See the picture below :



If number 4 is put in field 9, all -4- options from field 2 would be dropped which makes a solution impossible. Therefore, the 4 option in field 9 is illegal and should be removed.
The **hint3** procedures indicate these options in red and the neighbour group fields options that need to be saved are put in a green rectangle.

```
type TOptionField=record
       nr  :byte;
       green:byte;
       red :byte;
      end;
  THint3=array[1..6]of TOptionField;
```

function procHint3(var h : THint3) : boolean;
returns true in case of this hint and supplies Thint3 data in h.
The green and red bytes contain bit coded options:
If option 3 is present then bit 3 is set to
-1- (byte value = 8)

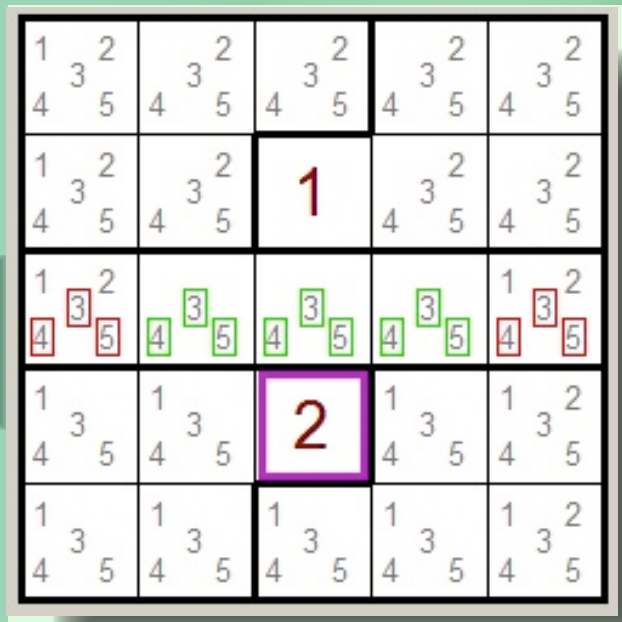
The structure is quite similar to **hint2**.
The outer repeat loop now selects a field (fld)
An inner repeat loop selects a neighbour group (grp).
A for loop then sums the neighbour group option fields in variable

dopt : if the grp field and fld are neighbours
dnopt : if the fields are no neighbours

If dopt has options that are not present in dnopt but are present in field fld then
the options in field fld are illegal.

HINT4
This checks for redundant options within a group.
If field options are (1,2) (1,2) (1,2,3,4,5) (1,2,3,4,5) (1,2,3,4,5) then the 1 and 2 numbers must appear in fields 1 and 2 and therefore these options may be removed from the other fields making the group options (1,2) (1,2) (3,4,5) (3,4,5) (3,4,5).

So far I have never encountered a hint4 in a tectonic puzzle.
When I started to program the hints, hint4 was made before hint3. But now I can only produce these hints by maybe unrealistic examples such as:



The green options fill three groups, so these options 3,4,5 cannot appear in fields 1 and 5.

This is the end of the Tectonic puzzle description.
For details please refer to the source code.





Apple is getting rid of 32-bit completely, so the Carbon API will become history, leaving Cocoa as the only future option.
It is recommended to use FPC 3.0.4a for Mojave macOS 10.14

WHAT IS COCOA?

Cocoa is the macOS system API, analogous to the WinAPI on Windows, which means it is the API you must use to create both neat applications and system extensions for the MacOS.
It is also possible to write command-line utilities for the Mac using Unix/Posix APIs (they also work), but they're not what is expected when talking about macOS.

Cocoa is based on Objective-C, a legacy from Steve Job's NextStep system, which has been merged back into the Apple OS. When you move from a classic MacOS to MacOSX, you find both the "classic MacOS API" (Carbon) and the "NextStep-based-API" (Cocoa) coexist in the Mac world. Since that merge, Carbon has not evolved much, and all new system functionality has been added as a part of Cocoa.

With the release of MacOSX 10.5, Carbon has been deprecated; and we expect Carbon (together with 32-bit support) to be gone completely in macOS 10.15. This means that using Cocoa libraries is the only way to continue future development for this platform.

Cocoa is system API, meaning that it is not a stand-alone library by itself, but rather a collection of different libraries. In macOS such libraries are called "frameworks" and this is the term that will be used going forward. Cocoa, then, is an umbrella framework, and this umbrella covers (or includes) multiple libraries, such as Foundation and AppKit – two libraries which are of paramount importance for Lazarus developers using the LCL.



Apple Park in Cupertino, California, April 2018



"Cocoa" tends to be used as a rather vague generalization. Often when developers say "Cocoa" it would be better to be more specific, and actually talk about "AppKit", or "Foundation", or some other Cocoa framework.

FREE PASCAL AND OBJECTIVE-C

Cocoa is Objective-C based. Apple users might also note, that it's Swift-based. The Objective-C language comes with a C-language runtime library, a library which lets you work with Objective-C language entities, such as classes, protocols and extensions. The presence of this library has made binary support of Objective-C possible in the Free Pascal compiler (FPC).
In order to support all features of Objective-C, FPC introduced a new mode switch. When this mode switch is enabled, FPC's Object Pascal is extended to support an almost complete set of Objective-C features. Any Objective-C framework and any Objective-C class can be used by Free Pascal without use of extra wrapper libraries. Objective-C libraries, then, have straightforward Pascal support. Unfortunately this is not true for Apple classes written in C++.

No C++ implementation provides a run-time library similar to the Objective-C runtime library, making integration of C++ classes with Free Pascal rather more complicated. In order to interact with C++ classes in Pascal, C-language wrapper functions must be provided first, and implementing support for object-oriented features such as inheritance, can be even more complicated.

Jonas Maebe, who added Objective-C support to Free Pascal, has documented all the similarities and differences between Object Pascal and the Objective-C language in detail. You can read the full story on the official wiki:

http://wiki.freepascal.org/FPC_PasCocoa

The Objective-C language itself has evolved over time. Objective-C version 2.0 (released together with MacOSX 10.5) introduced additional language features such as automatic reference counting, and automatic properties. These features are not yet in Free Pascal, because integration with Objective-C started before the 10.5 release. However, the absence of those features is not a show-stopper. One can interact fully with Objective-C using FPC as it is today. Support for "blocks" (closures) which are an Objective-C and a C-language Apple extension have been available in Free Pascal's "trunk" version for some time.
Overall, native support for Objective-C in FPC makes Pascal development straightforward, and by using Free Pascal's syntax for Objective-C it is very easy to switch from Object Pascal. Apple also provides a lot of documentation about its classes, and makes headers for its standard libraries available.

Cocoa and the LCL

The Lazarus Component Library (LCL) is the backbone of Lazarus, providing a cross-platform WinAPI-derived interface to native controls and native behaviour on all supported platforms. The AppKit and Foundation frameworks are two integral parts of the Cocoa framework which deal with what the LCL is designed to cover: user interface elements.
AppKit provides an object-oriented approach. Analogous to the LCL's TApplication, AppKit provides the NSApplication class, which provides "run" and similar methods. Likewise there is NSWindow (an analogue of the LCL TForm), and NSButton (an analogue of the LCL TButton, with "title" instead of "Caption"). Unfortunately, those similarities don't greatly help, and wrapping Cocoa into the LCL requires provision of many C-like interfaces (with a few exceptions for widgetset classes). In this way, each Objective-C class becomes an LCL handle.

Implementing each LCL function means casting a handle to the Objective-C class and calling a specific method of that class. In many cases, such interfacing is straightforward. Alas, in other cases it is not.

Difficulties arise because of particular architectural differences between Cocoa's AppKit framework and the LCL:

- The LCL is message-queue based, but AppKit is not. There are AppKit "events" that behave similarly to messages in the LCL.
  But propagation of an AppKit event stops at a window (the LCL form) and further propagation of the eventhappens through calling specific methods.
  For example, when a user moves her mouse, there's no EVENT_MOUSEMOVE message in Cocoa. The event reaches an NSWindow object, and the window object would call the "mouseMove" method of all the window object's contained controls. The LCL needs to convert all such event calls into messages.

- All mouse or keyboard input events are processed by a chain of controls in AppKit.
  If one control cannot handle the event, it passes it to its parent. In the LCL only certain messages (for example opening a context menu), would be passed to the parent.
  A focused control that does not handle a particular keypress and key-event would simply do nothing. In Cocoa, by default, the event the LCL control ignores gets passed to the control's parent, unless explicitly stopped.
  The LCL needs to take care of processing such "event chains" in a cross-platform manner.

- AppKit is designed for user interface composition, and the controls provided by the system have much less functionality, compared to a typical LCL control. For example, in the LCL a text editing control such as TMemo is expected to scroll the text.
  In Cocoa, the NSTextView text editing control has no scrolling functionality at all. It must be placed within a special scrolling control (NSScrollView) in order to show scroll bars.
  Thus complex controls are composed of several simpler controls. Often, then, a single LCL control must be represented by two or more AppKit controls composed together.
  Hiding the fact that the LCL control is in reality multiple AppKit controls under a single "Handle" is another task that LCL wrappers around Cocoa have to undertake.

- Various other differences include these:
  - ◇ Cocoa implements a different coordinate system (from bottom-to-top in Cocoa, whereas the LCL is top-to-bottom) which can be switched on-the-fly on a per-control basis in Cocoa.
  - ◇ MacOS offers no "per window" menus at all: all menus are at the top of the main window.
  - ◇ The default screen resolution differs: MacOS is natively 72 dpi, while the LCL expects 96 dpi.
  - ◇ Dialog behaviour on MacOS differs from the Windows-derived dialog behaviour the LCL expects.
  - ◇ Hot-key processing in Cocoa differs markedly from what the LCL expects.
  - ◇ There are many more differences, large and small. See the bug tracker for the latest complaints!

DEVELOPMENT PREREQUISITES

The minimum MacOS version the LCL supports is MacOS 10.6, since there are number of API calls introduced in 10.6 that the LCL uses explicitly. Running an LCL app compiled on 10.5 will likely be problematic, although perhaps possible with some patching. Running an LCL app compiled on MacOS 10.4 is not possible, because Objective-C 2.0 features used in the LCL are not available in MacOS 10.4.
The latest feature released for MacOS 10.14 Mojave – Dark Mode – is supported. So, you can get the Dark interface and color scheme, if appropriate LCL color counterparts are used (`clWindow`, `clControlText`).
Apple has been deprecating certain interfaces through successive MacOS releases. The LCL developers take care of keeping backwards compatibility, yet utilizing the new APIs and features where possible. This means that the LCL supports full-screen mode, although it differs between 10.6 and 10.7. Likewise the LCL now supports both view-based and cell-based processing. So, effective from version 10.7, the LCL has a new view-based (rather than cell-based) NSTextView, which you can see in the `TListBox` and `TListView` controls.

NAMING

Classic MacOS - Apple's system prior to the new generation of MacOSX. The last version was MacOS 9.0
MacOSX – the newer (unix based) Apple's operating system up. Starting with version 10.0 up to 10.7. Starting with 10.8 and up to 10.10 it was more common to refer to it as OS X
macOS – the name for the same system (Mac OS X), yet rebranded by Apple (to match iOS watchOS). This is the latest proper spelling of the system.
So the way you spell MacOSX, OSX, macOS might refer to a particular age/version of the system. Yet, if there's no context of a particular version all of those refer to the same Apple's operating system.

BUG OR FEATURE?

Owing to Cocoa's "composite" approach it is sometimes hard to determine if a certain feature is a system limitation or not. For instance, in the LCL, tab-controls are expected to have as many tabs as needed. If there are too many tabs to fit within the window, you expect that an additional widget will be shown to enable the user to scroll through further hidden tabs.

In Apple's world a tab control should have only a limited number of tabs. All tabs should be visible all the time, which places constraints on the tab-control size and on the window itself. From Apple's perspective this is a feature. From the perspective of the LCL this is a bug. In order to satisfy the LCL's cross-platform needs, an extended functionality has been composed for tab-controls.

By intelligent combination of Objective-C support with the LCL's object-oriented interface (which means that the LCL can override methods and implement them in the way it wants) almost any LCL requirement can potentially be implemented. The only risk with implementing non-Mac features in the LCL (so that they work on the Mac) is that they are foreign to Mac users.

So far, there's only one feature that cannot be fully supported. This is the z-order of overlapping controls. However, it is quite a rare case in interface design, and Cocoa has fully documented this feature.
The LCL is also limited compared to MacOS in some respects. The MacOS interface provides some UI elements that have no corresponding control in LCL: for example "Sheets", a non-modal dialog window. In order to address that issue an additional LCL extension has been started:
http://wiki.freepascal.org/macOS_extensions
The idea is to make an LCL application even more Mac-like in look and feel, without affecting the other target operating systems.

SUMMARY

Apple leaves us no option, but to use Cocoa. The Free Pascal compiler provides enough support to code with Objective-C. The Lazarus IDE itself can currently be compiled to run on Cocoa and is usable. The Lazarus IDE is a sufficiently complex app to expose bugs in the LCL's Cocoa implementation that simpler apps may not disclose. A good few bugs remain, which are steadily being fixed.
There's another year left to finish porting the project to 64-bit, which for MacOS targets means Cocoa. The sooner tests begin the better the outcome will be. If you are a Mac user, please help us with the testing that is needed, using both the Lazarus IDE and your own apps!

It is recommended to use FPC 3.0.4a for Mojave macOS 10.14. EXTRA INFO:
https://sourceforge.net/projects/freepascal/files/Mac%20OS%20X/3.0.4/fpc-3.0.4a.intel-macosx.dmg/download
https://sourceforge.net/projects/freepascal/files/Mac%20OS%20X/3.0.4/)

BY DETLEF OVERBEEK

starter expert

Introduction

In this article I want to start explaining what it means to use Tortoise/SVN and what its purpose is:
In short it is creating a backup up system that has the advantage of sub-versioning and even be able to work together on a project with a team, or simply work on it and be able to go back in time once you may need that.

In this first series we do the first step: Have the latest version of Lazarus always at hand and make it possible to have even many different Versions of Lazarus at hand. You of course can do this for Programs and projects of every kind. Please realize that with more projects usually you will need to invest more time to do so.
These first steps will show you how easy it is to this. It might even become a habit.



Figure 1: Logo

Abstract:

What is SVN Tortoise - Installing - Purposes

SVN Tortoise

SVN is a system of sub-versioning and exists for Linux, Mac and Windows.
For Windows there is also a Graphical environment: TortoiseSVN.

Software versioning

is the process of assigning either unique *VERSION NAMES* or unique *VERSION NUMBERS* to unique states of computer software. Simply said: creating and managing of several versions of programs.
Within a given version number category (major, minor), these numbers are generally assigned in increasing order and correspond to new developments in the software. At a fine-grained level, revision control is often used for keeping track of incrementally different versions of information, whether or not this information is computer software.
Computer software (Lazarus, Delphi) is often tracked using two different software versioning schemes - internal version number that may be incremented many times in a single day, such as a revision control number, and a *released version* that typically changes far less often, such as *semantic versioning* (linguistic - by name)or a project code name.

SVN - for Linux, Mac and Windows.

This is the abbreviation of Apache Subversion - its command name is *svn*) is a software versioning and revision control system distributed as open source under the Apache License.
Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly compatible successor to the widely used Concurrent Versions System (CVS).

The open source community has used Subversion widely: for example in projects such as Apache Software Foundation, Free Pascal, FreeBSD, GCC and SourceForge. CodePlex*1 used to offer access to Subversion as well as to other types of clients. Subversion was created by CollabNet Inc. in 2000, and is now a top-level Apache project being built and used by a global community of contributors.

*1 ***CodePlex*** *was a forge website by Microsoft. While it was active, it allowed shared development of open-source software. Its features includes wiki pages, source control based on Mercurial, Team Foundation Server (TFS), Subversion (also powered by TFS) or Git, discussion forums, issue tracking, project tagging, RSS support, statistics, and releases. While CodePlex once encompassed a wide variety of projects, including SQL Server, WPF and Windows Forms-related projects, its major activities were focused on .NET Framework (including ASP.NET) and SharePoint. The most prominent and used project that was born inside CodePlex, the AJAX Control Toolkit, is a joint project between the community and Microsoft.*

TortoiseSVN

is a Subversion client, implemented as a Microsoft Windowsshell extension *2, that helps programmers manage different versions of the source code for their programs. It is free software released under the GNU General Public License*3.

Version control is the art of managing changes to information. It has long been a critical tool for programmers, who typically spend their time making small changes to software and then undoing or checking some of those changes the next day. Imagine a team of such developers working concurrently - and perhaps even simultaneously on the very same files! - and you can see why a good system is needed to *manage the potential chaos*.

*2 ***Microsoft Windowsshell extension.*** *The Windows UI provides users with access to a wide variety of objects necessary for running applications and managing the operating system. The most numerous and familiar of these objects are the folders and files that reside on computer disk drives. There are also a number of virtual objects that allow the user to perform tasks such as sending files to remote printers or accessing the Recycle Bin. The Shell organizes these objects into a hierarchical namespace and provides users and applications with a consistent and efficient way to access and manage objects.*

If you want to work correctly wit Tortoise SVN you have to create an other directory. That way it assures nothing can **really** go wrong. If anything happens you Still have your latest version of Lazarus or what other version you might have installed. That means you install an extra version of Lazarus in the new Directory. So the next step is to download the Tortoise SVN: **`Go to https://sourceforge.net/projects/tortoisesvn/`**

Choose your own version: 32or 64 bit. To make the right choice her some extra information:

System requirements

TortoiseSVN runs on Windows 10 / 8 / 7 Vista or higher and is available in both 32-bit and 64-bit flavours. The installer for 64-bit Windows also includes the 32-bit extension parts. Which means you don't need to install the 32-bit version separately to get the TortoiseSVN context menu and overlays in 32-bit applications. Support for Windows 98, Windows ME and Windows NT4 was dropped in version 1.2.0, and Windows 2000 and XP up to SP2 support was dropped in 1.7.0. Support for Windows XP with SP3 was dropped in 1.9.0. You can still download and install older versions if you need them.

See the Language Packs and Spell Checkers for more information on how to install these. If you encounter any problems during or after installing TortoiseSVN please refer to our online FAQ at **`https://tortoisesvn.net/faq.html`**.

The standard installer has support only for English, but you can download separate language packs and spell check dictionaries separately after installation.

Language Packs

The TortoiseSVN user interface has been translated into many different languages, so you may be able to download a language pack to suit your needs. You can find the language packs on our translation status page . And if there is no language pack available, why not join the team and submit your own translation.



Figure 2: Website

Installation

TortoiseSVN comes with an easy to use installer. Double click on the installer file and follow the instructions. The installer will take care of the rest. Don't forget to reboot after installation.
Note You need Administrator privileges to install TortoiseSVN. The installer will ask you for Administrator credentials if necessary. Language packs are available which translate the TortoiseSVN user interface into many different languages.

Each language pack is packaged as a **`.msi`** installer. Just run the install program and follow the instructions. After the installation finishes, the translation will be available. The documentation has also been translated into several different languages. You can download translated manuals from the support page on our website.

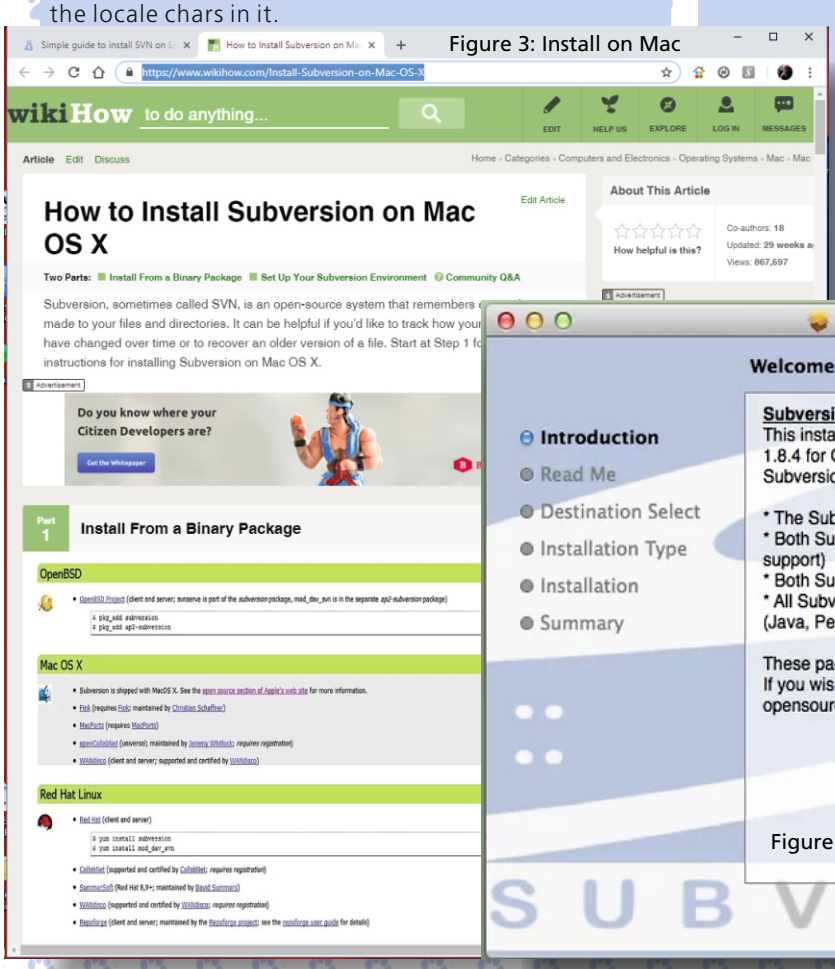
Spellchecker

TortoiseSVN uses the Windows spell checker if it's available (Windows 8 or later). Which means that if you want the spell checker to work in a different language than the default OS language, you have to install the spell checker module in the Windows settings (Settings > Time & Language > Region & Language). TortoiseSVN will use that spell checker if properly configured with the **tsvn:projectlanguage** project property. In case the Windows spell checker is not available, TortoiseSVN can also use spell checker dictionaries from OpenOffice and Mozilla. The installer automatically adds the US and UK English dictionaries. If you want other languages, the easiest option is simply to install one of TortoiseSVN's language packs. This will install the appropriate dictionary files as well as the TortoiseSVN local user interface. After the installation finishes, the dictionary will be available too. You can also install the dictionaries yourself. If you have OpenOffice or Mozilla installed, you can copy those dictionaries, which are located in the installation folders for those applications. Otherwise, you need to download the required dictionary files from **http://wiki.services.openoffice.org/wiki/Dictionaries** .
Once you have got the dictionary files, you probably need to rename them so that the filenames only have the locale chars in it.

Example:
en_US.aff
en_US.dic

Then just copy them into the **%APPDATA%\TortoiseSVN\dic** folder. If that folder isn't there, you have to create it first.
TortoiseSVN will also search the Languages sub-folder of the TortoiseSVN installation folder (normally this will be **C:\Program Files\TortoiseSVN\Languages)**; this is the place where the language packs put their files. However, the **%APPDATA%**-folder doesn't require administrator privileges and, thus, has higher priority. The next time you start TortoiseSVN, the spell checker will be available. If you install multiple dictionaries, TortoiseSVN uses these rules to select which one to use. Check the **tsvn:projectlanguage** setting. Refer to the section called "Project Settings" for information about setting project properties. If no project language is set, or that language is not installed, try the language corresponding to the Windows locale. If the exact Windows locale doesn't work, try the "Base" language, e.g. de_CH (Swiss-German) falls back to de_DE (German). If none of the above works, then the default language is English, which is included with the standard installation.

Figure 3: Install on Mac



MacOS X

In this article we do not have the possibility to discuss versions for Mac and for Linux. That is why we simply show the websites where you can find your version.
Create a subversion on MacOS X : **https://www.wikihow.com/Install-Subversion-on-Mac-OS-X**

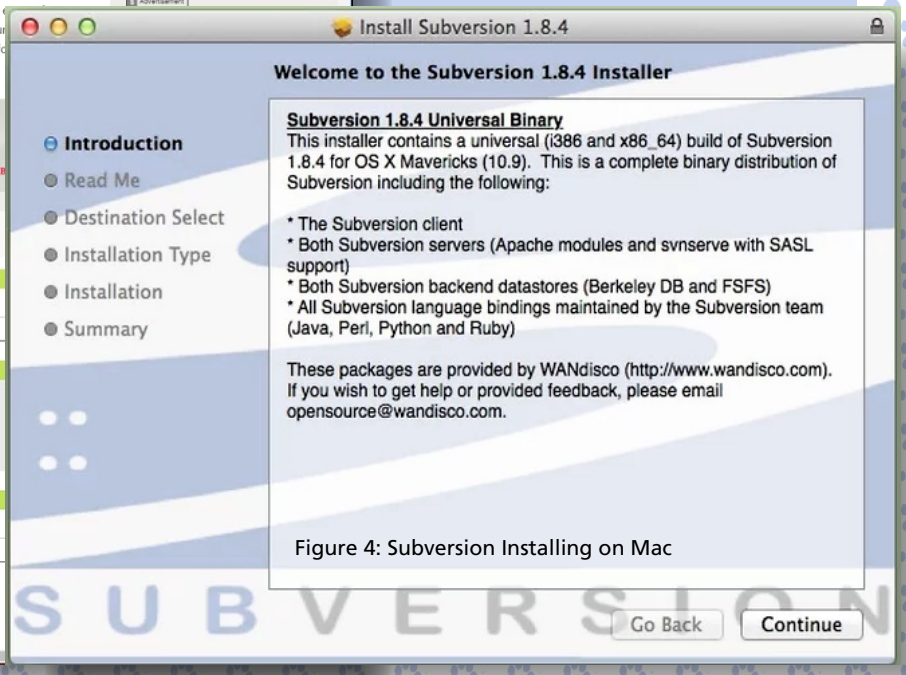


Figure 4: Subversion Installing on Mac

Linux install
`https://linuxtechlab.com/simple-guide-to-install-svn-on-linux-apache-subversion/`

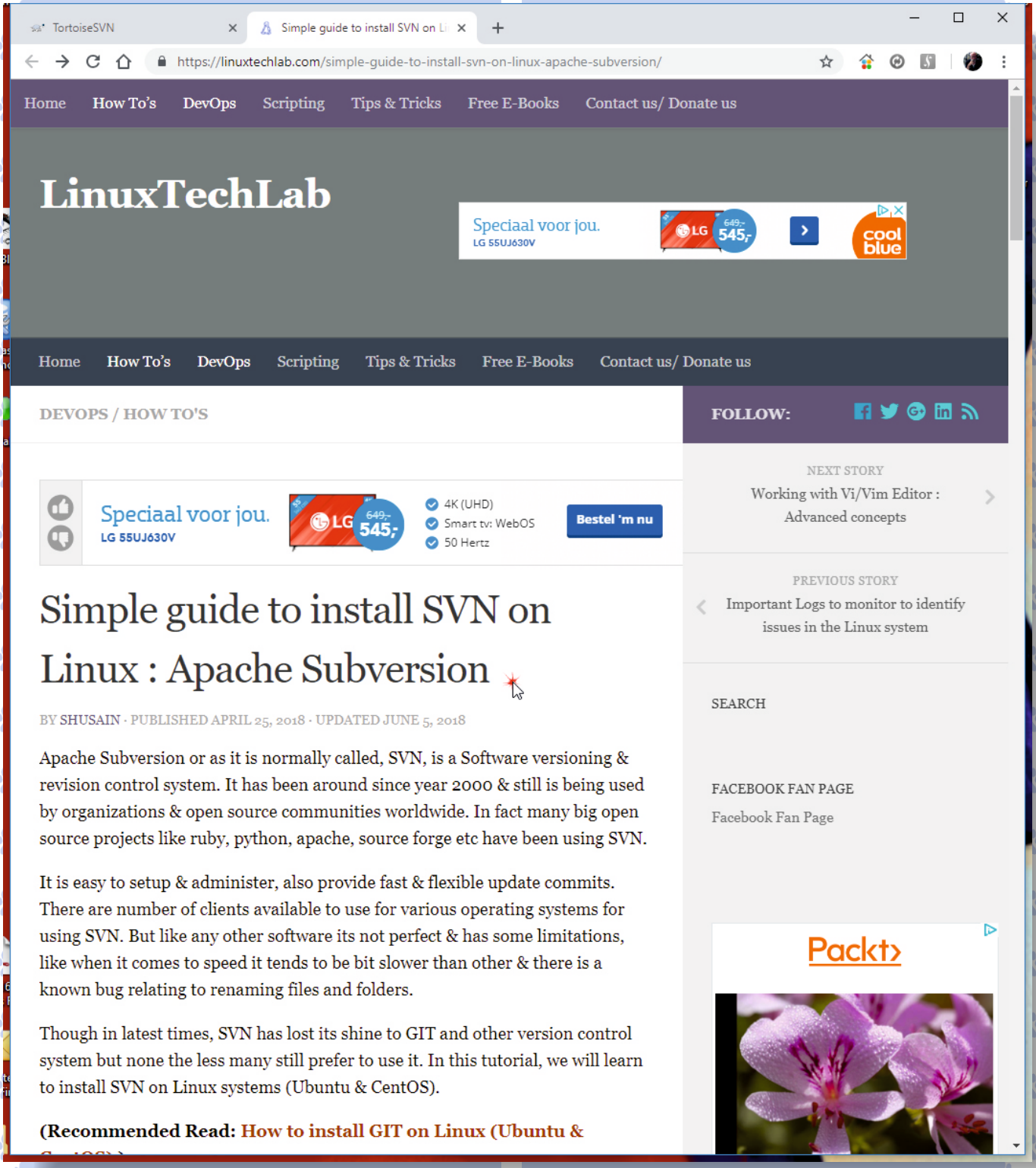
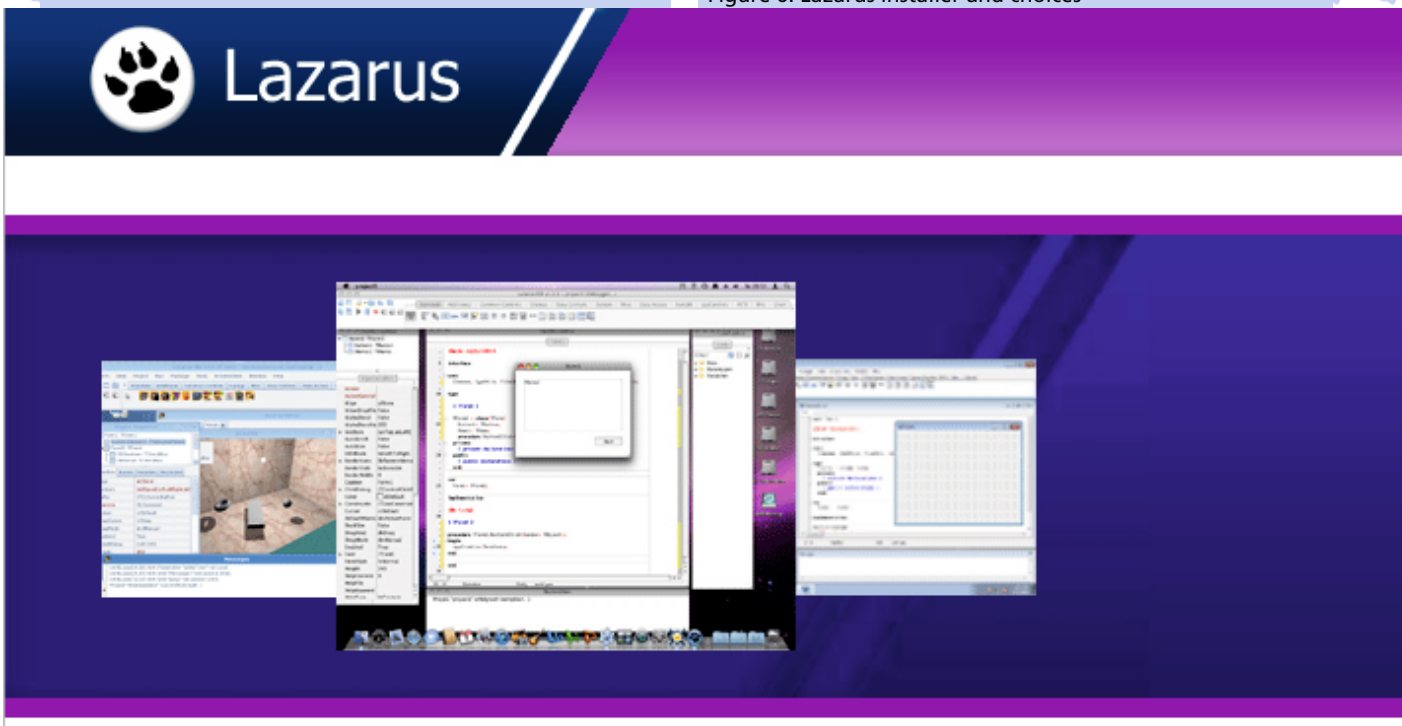


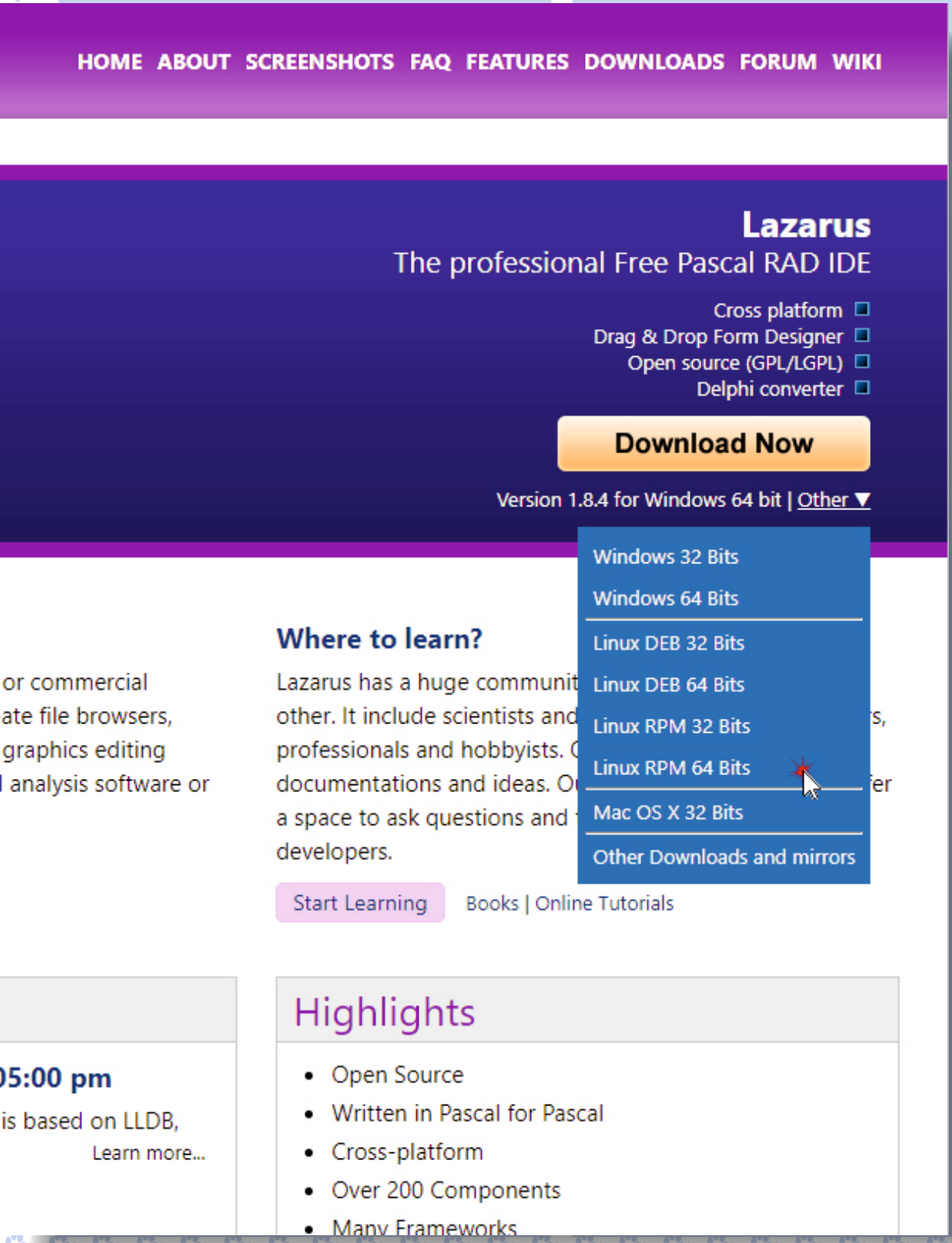
Figure 5: Subversion Installing on Linux

Quick start to install Lazarus under Windows Tortoise SVN

I now will show you how to install a Lazarus version and create the Tortoise SVN system and how to work with it.

Figure 6: Lazarus installer and choices



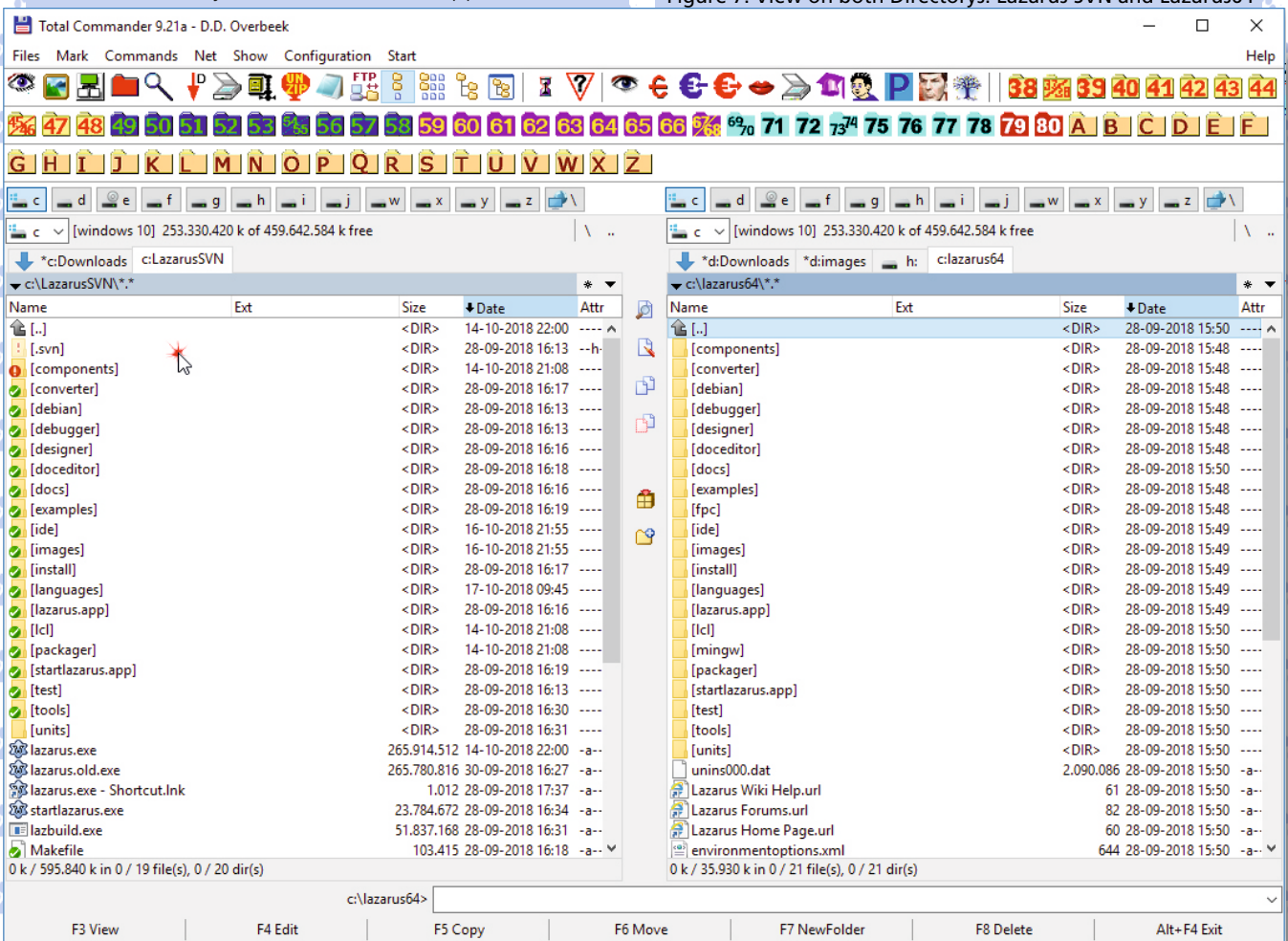

HOME ABOUT SCREENSHOTS FAQ FEATURES DOWNLOADS FORUM WIKI
Lazarus
The professional Free Pascal RAD IDE
Cross platform
Drag & Drop Form Designer
Open source (GPL/LGPL)
Delphi converter
Download Now
Version 1.8.4 for Windows 64 bit | Other
Windows 32 Bits
Windows 64 Bits
Linux DEB 32 Bits
Linux DEB 64 Bits
Linux RPM 32 Bits
Linux RPM 64 Bits
Mac OS X 32 Bits
Other Downloads and mirrors
Where to learn?
Start Learning
Books | Online Tutorials
Highlights
Open Source
Written in Pascal for Pascal
Cross-platform
Over 200 Components
Many Frameworks

Getting the necessary files

1. We of course need the Lazarus version to be installed on your computer. If you have that already installed you can skip the next step:
Create a new directory where the Lazarus projects will be hosted. For Example: **"Laz Projects".**
In that directory you might create several sub-dir's with other Lazarus versions.
Or take your own version and keep it that way - do not change the name. That means you might loose your version if things do not work as expected.
We will start this simple way and later on explain everything you could do with either a Lazarus- or Delphi version. (Delphi will be handled at the end of this file.)
NOTE: for Lazarus we create the project as being a special version of the program. You should realize that you can use your TortoiseSVN for organizing versions of your Project.
Delphi is not a project where we have release candidates. We have have of course several versions and those can be handled in the same way.
But with Delphi you can do Projects for apps as well. Lazarus can of course also run projects and so you can, parallel to what your doing here create an SVN Tortoise Project for each Lazarus app.

For downloading the latest Lazarus stable release go to **http://www.lazarus-ide.org/** and choose one of the available options. You can of course also download the release-candidates which are usually very often updated.
In the overview (Total Cammander at the bottom of the page) you can see the two directory's.
The left contains the new version you can use with Svn Tortoise, the other is still available.
So if you would use your shortcut - you would start your old version of Lazarus, but if you create a new shortcut in your Lazarus SVN you will start your svn - Lazarus version:
C:\LazarusSVN\lazarus.exe.
See page 15 of 25

Figure 7: View on both Directorys: Lazarus SVN and Lazarus64



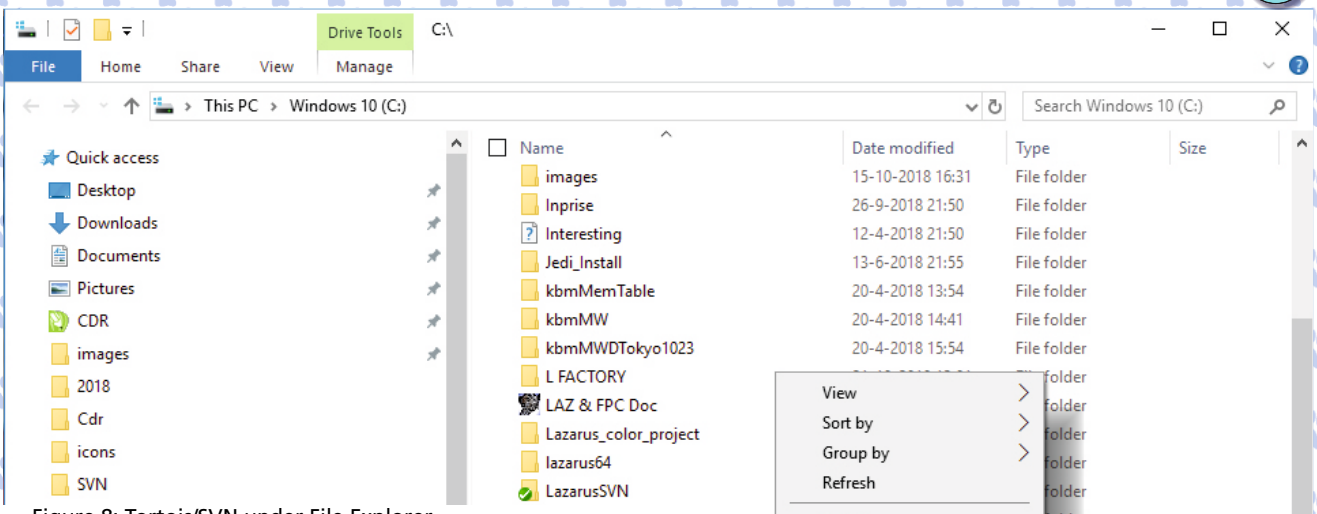


Figure 8: Tortois/SVN under File Explorer

Installing the LazarusSVN version

The installing of Tortoise SVN is a separate issue: simply just install it and after that you will find that the program available if you right-click with your mouse. Her is an example for the Windows File Explorer.

You can see two main items: SVN Checkout and TortoiseSVN. Now before we start on explaining how to work with it, lets try to get your new Lazarus running. First of all: you still have the "normal" version in which you wore working. If you take a closer look at the files in the directory "LazarusSVN" it is still empty.

Lets start by solving this: you need to do an **SVN Checkout**. The example you can see at the top. This is shown in the File Explorer, but it also works quite nice in the Total Commander (page before). Now the first screen shows up and in this you need to fill in the line of the URL of Repository you will have to give the exact same name **`https://svn.freepascal.org/svn/lazarus/trunk/`** and in the heckout directory the name of the directory where you want to put your checkout files: **`C:\LazarusSVN.`** Of course it can be put anywhere: **`F:\MyOwnNameOfTheDirectory.`** Simply click OK and the checkout will start:

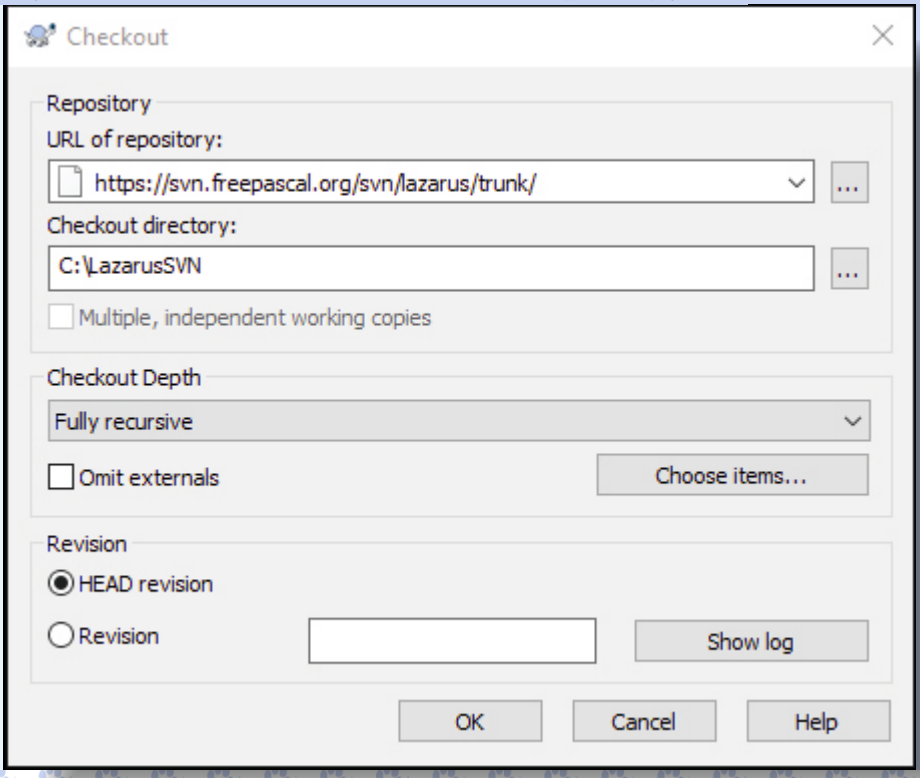


Figure 9: Addresses of the URL of the server and your Project

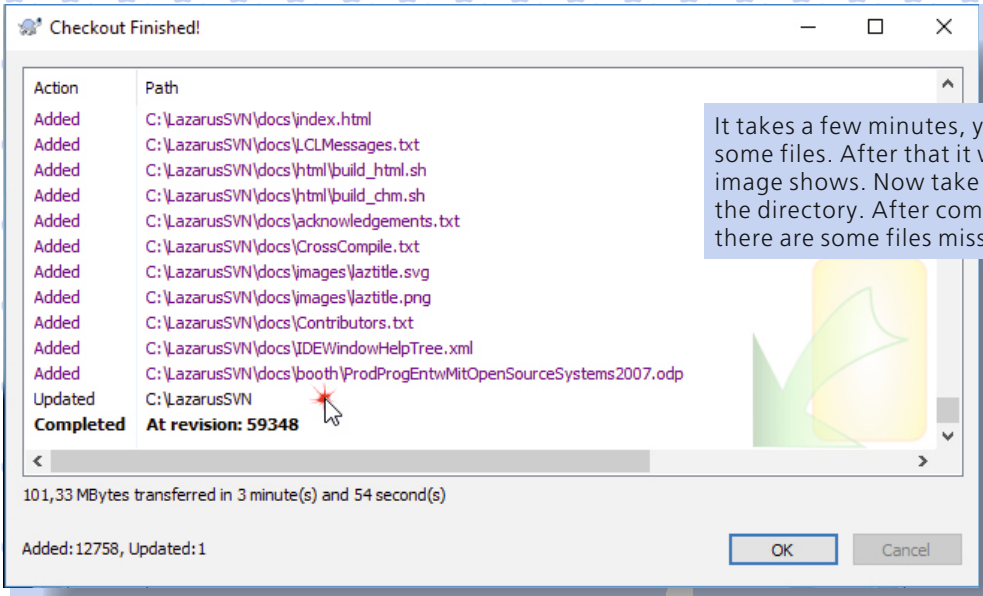


Figure 10: Start of the checkout

It takes a few minutes, you are copying quite some files. After that it will be similar to what this image shows. Now take a look at the content of the directory. After comparing you will find that there are some files missing:

Figure 11:
Creating an extra project

In this case its about something very important like the **Lazarus.exe** file. We need to do some tricks to get the installation done. Since we have no **Lazarus.exe.**
We will have to copy that from the original installation:

1. **Copy and paste** it from the original **Lazarus64** in my case) to the **LazarusSVN** directory.

2. To get it working we need also a lazarus.cfg file. That is the **configuration file** we need to create ourselves.

3. Before you do this please **create an extra Directory** where you will put the **lazarus.cfg** file.
   This is to make sure you will not loose that file after an eventual error or after de-installing.
   Name the Directory **C:\LazarusSVNConfig**

3. Open a text editor (notepad or some other app).
   - In this to be configure-file you will have to put some text:

```
--primary-config-path=C:\LazarusSVNConfig\
```

that file needs to be copied into the **c:\LazarusSVN** directory, so it will be available in two directory's:
C:\LazarusSVNConfig and **C:\LazarusSVN\.**
(That is because the exe will read the configuration file.) Then copy the lazarus.cfg into **C:\LazarusSVN\.** (Figure xx)
The Laxzarus.exe file was alraedy copied.
Start the exe file and compile Lazarus. You will see an information window showing up:

It shows the problems that occur. No need to be worried. It has found Lazarus, but needs to to find the Compiler. You will see that the problems are solved as soon as you provide the path:

c:\lazarus64\fpc\3.0.4\bin\i386-win32\fpc.exe.

NOTE: You can use the directory of your own (old) project because you do not need to install fpc twice. (In my case Lazarus64). (Figure x and on the next pages figure xxx)

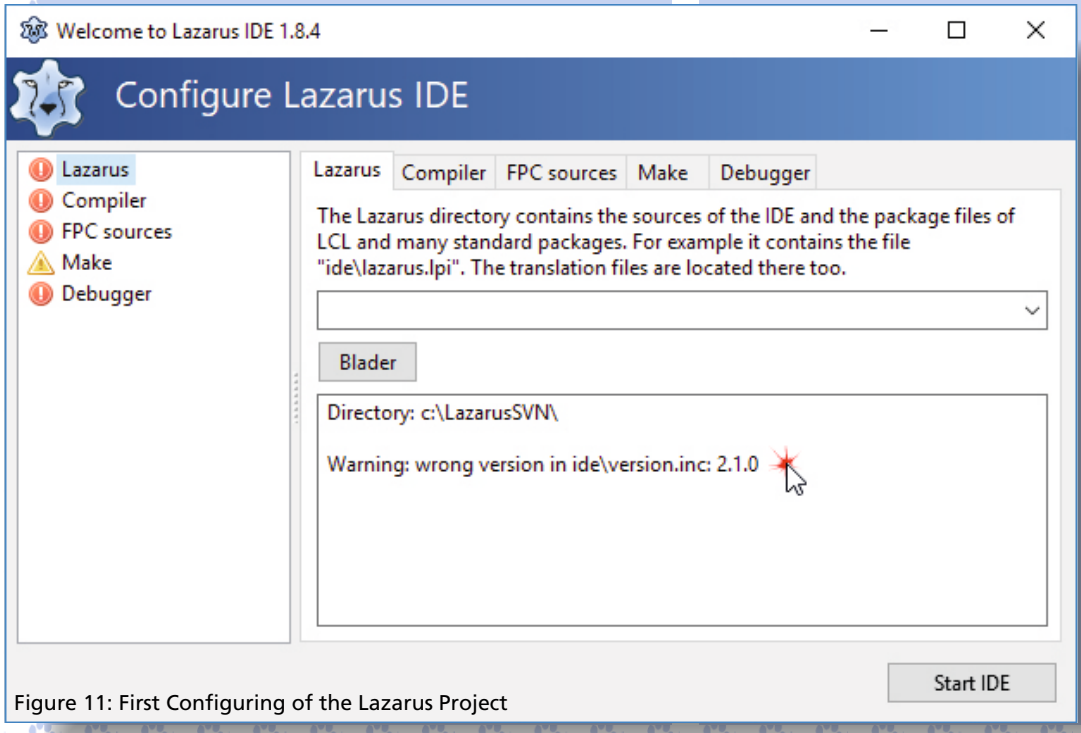


Figure 11: First Configuring of the Lazarus Project

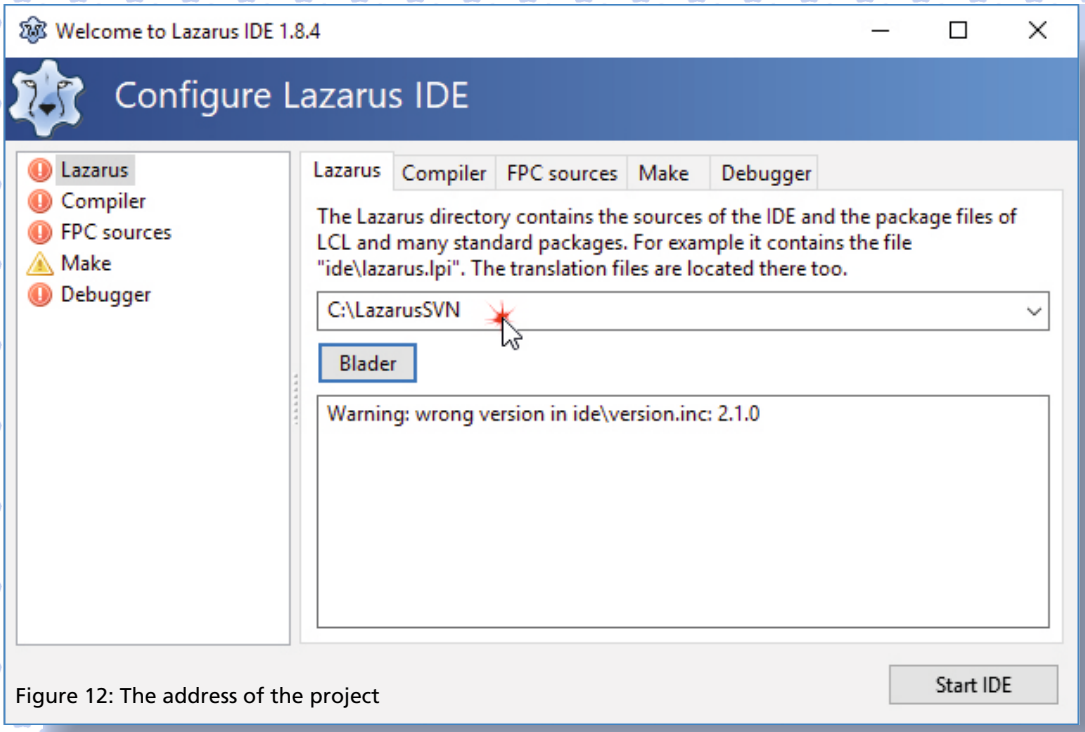


Figure 12: The address of the project

The tab lazarus-tab is active: that means the correct path is needed. It willshow an error. The conflict lies between version1.8.4 and 2.1.0. Once the compilation has been doen at the end odf installing you will finde this problem solved. We firts need to finish all steps.

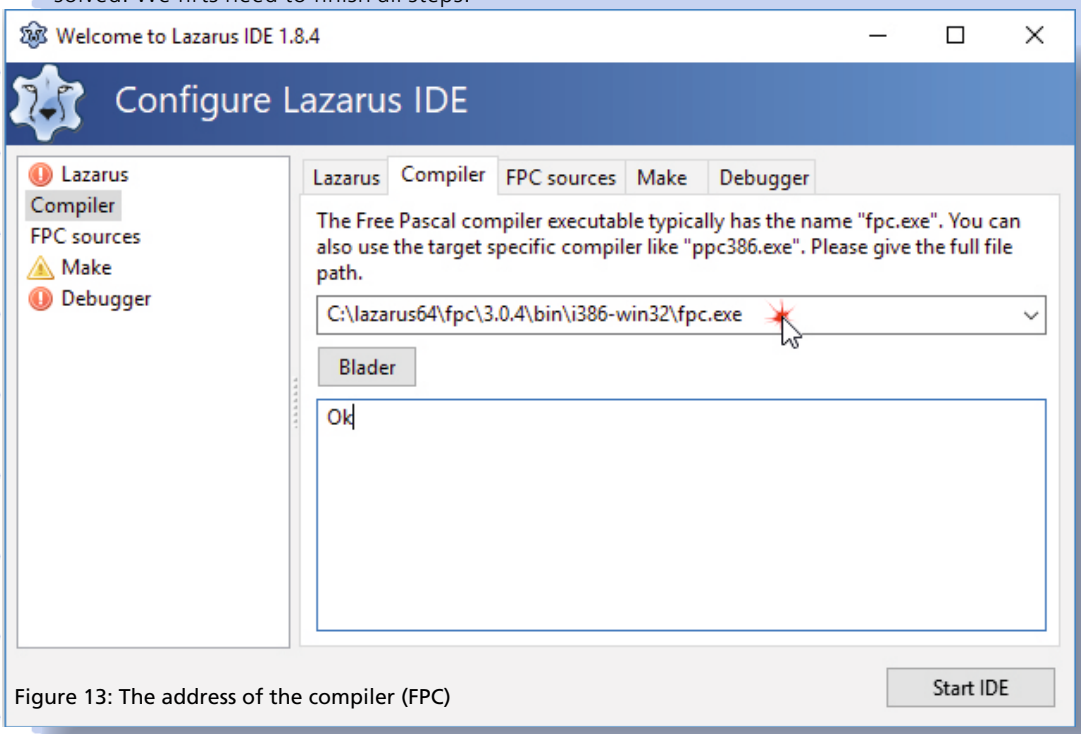


Figure 13: The address of the compiler (FPC)

The Compiler Tab is active: Here you should enter again the path in Lazarus64
`c:\lazarus64\fpc\3.0.4\bin\i386-win32\fpc.exe`
You dont have to do an extra install of **FPC**. It can be done but for the readabilty I chose not todo so. (It makes it rather complicated and we can explain that in an extra article). You can use this version of **FPC**. Because of giving in the path you will find the icons updated...

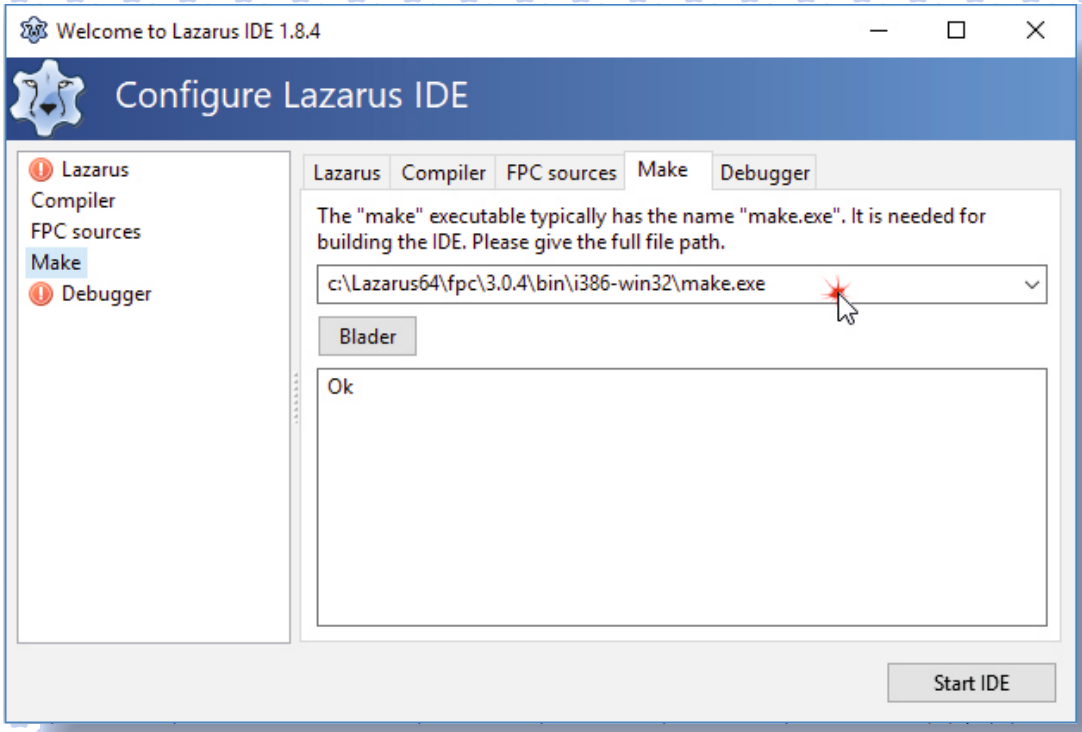


Figure 14: The Make-tab is active: the make file needs to be found – we again will use it from the old directory: C:\Lazarus64. The path is:
`c:\lazarus64\fpc\3.0.4\bin\i386-win32\make.exe`

NOTE: It might be that there is a path found which sends you to the **`Delphi Make.exe`**. Delete that and put the corrcet path in.

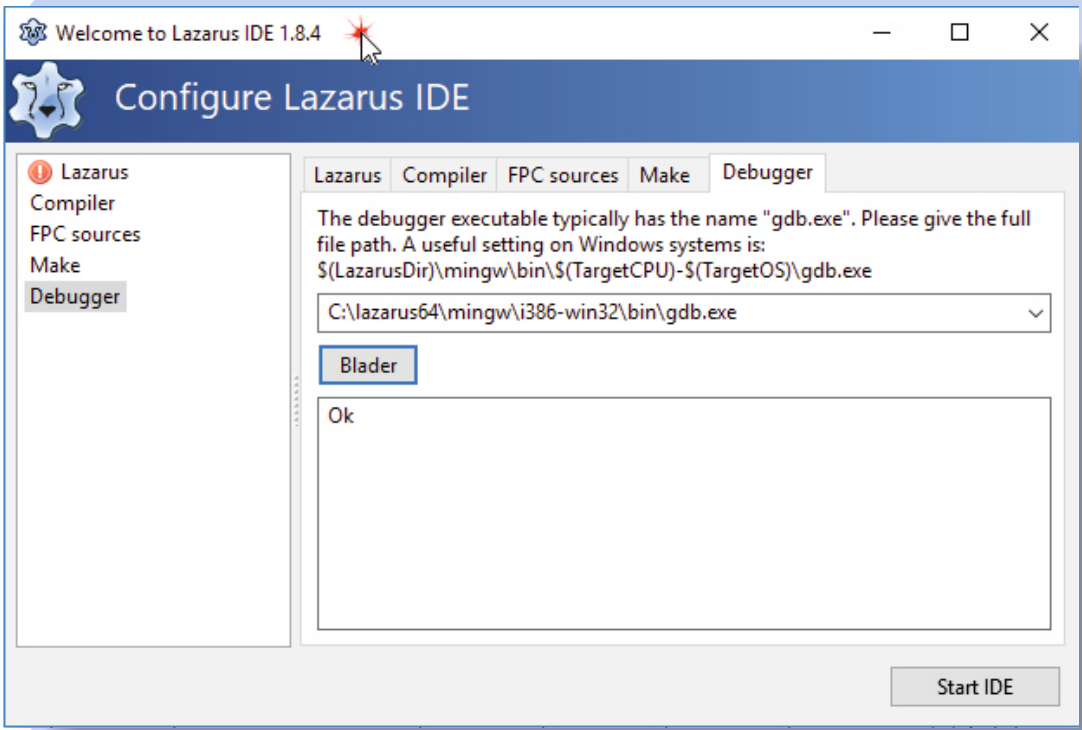


Figure 15: The Debug tab is active: The debugger file needs to be found:
`c:\lazarus64\mingw\i386-win32\bin\gdb.exe.` This is the last stage

Now click Start IDE.
The firts thing happening will be:

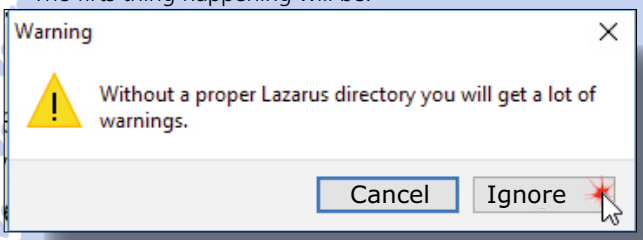
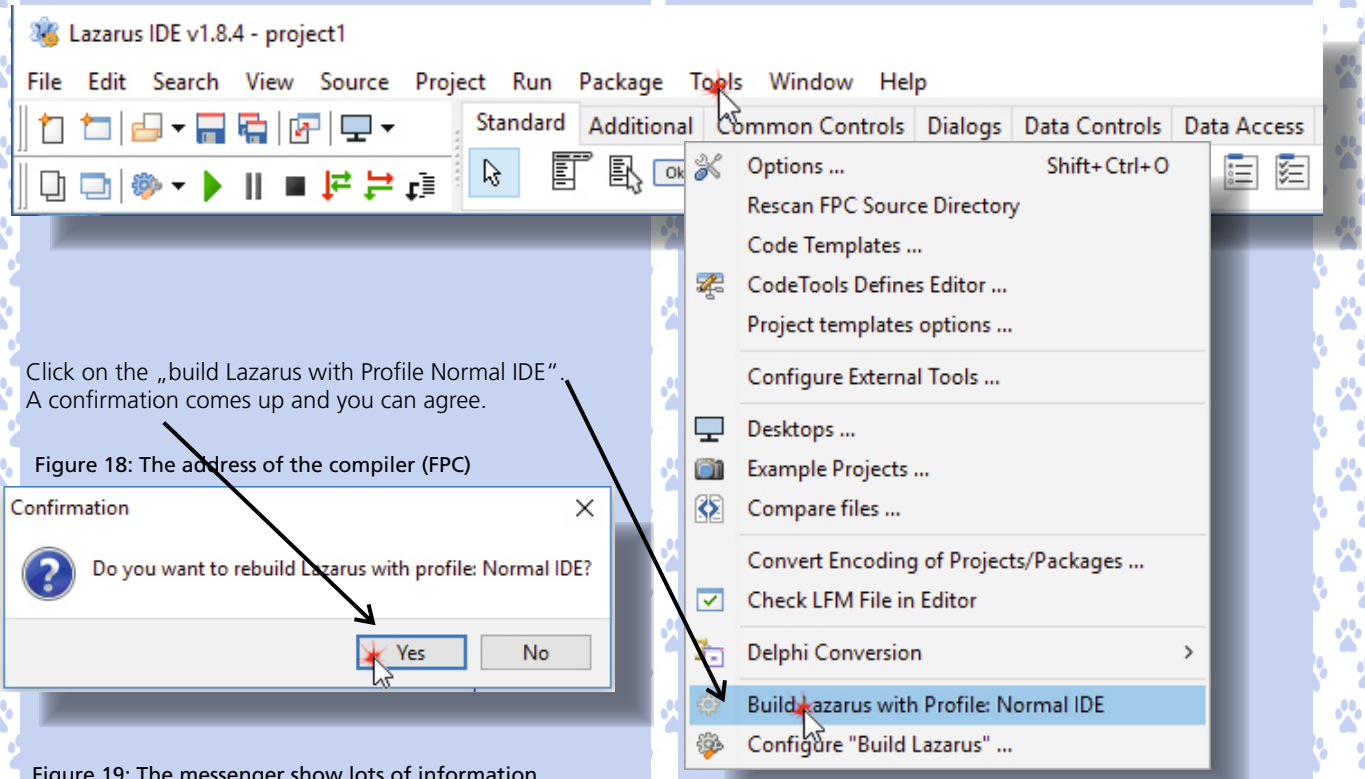


Figure 16: Warning

Simply click Ignore. Lazarus will start after this without problems. Since we have now a working version we need to recompile the program to cretae the newest version from the program.

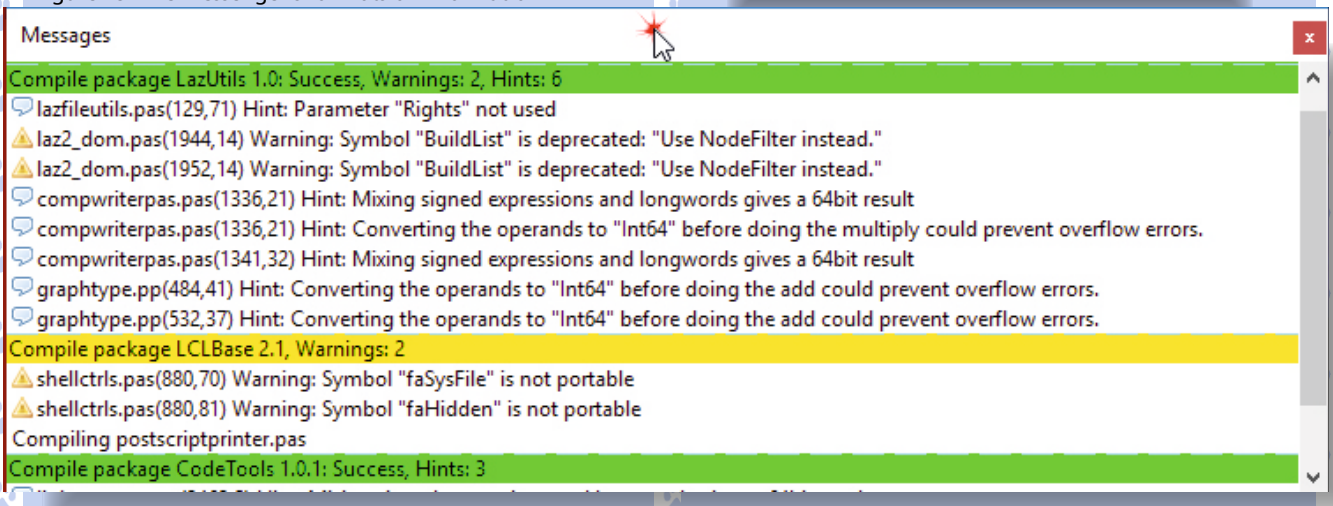
Figure 17: Build Lazarus



Click on the „build Lazarus with Profile Normal IDE".
A confirmation comes up and you can agree.

Figure 18: The address of the compiler (FPC)

Figure 19: The messenger show lots of information.



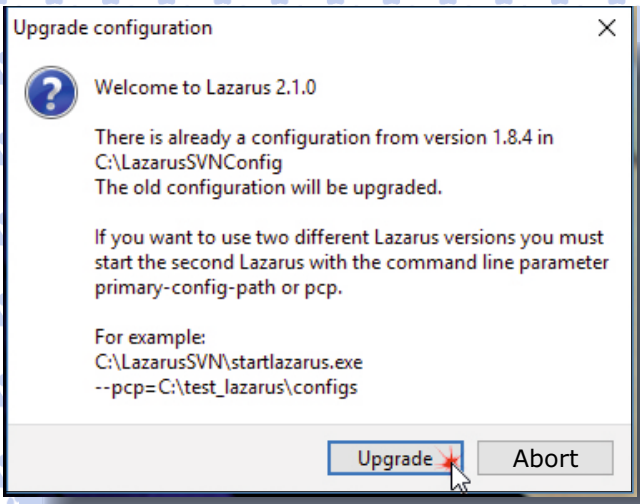


Figure 20: Getting upgrade information:
Click on upgrade. Lazarus will not start because of this warning. Just Cancel.

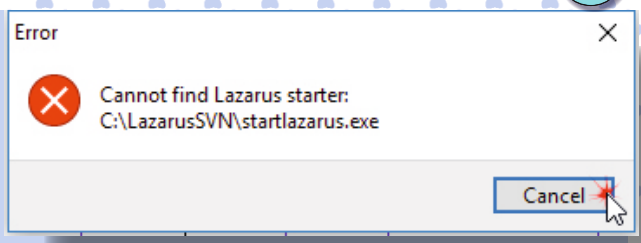
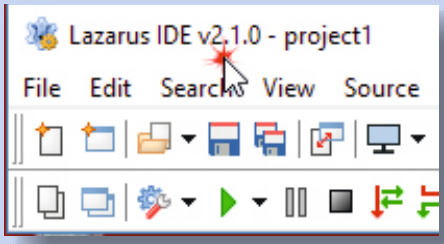


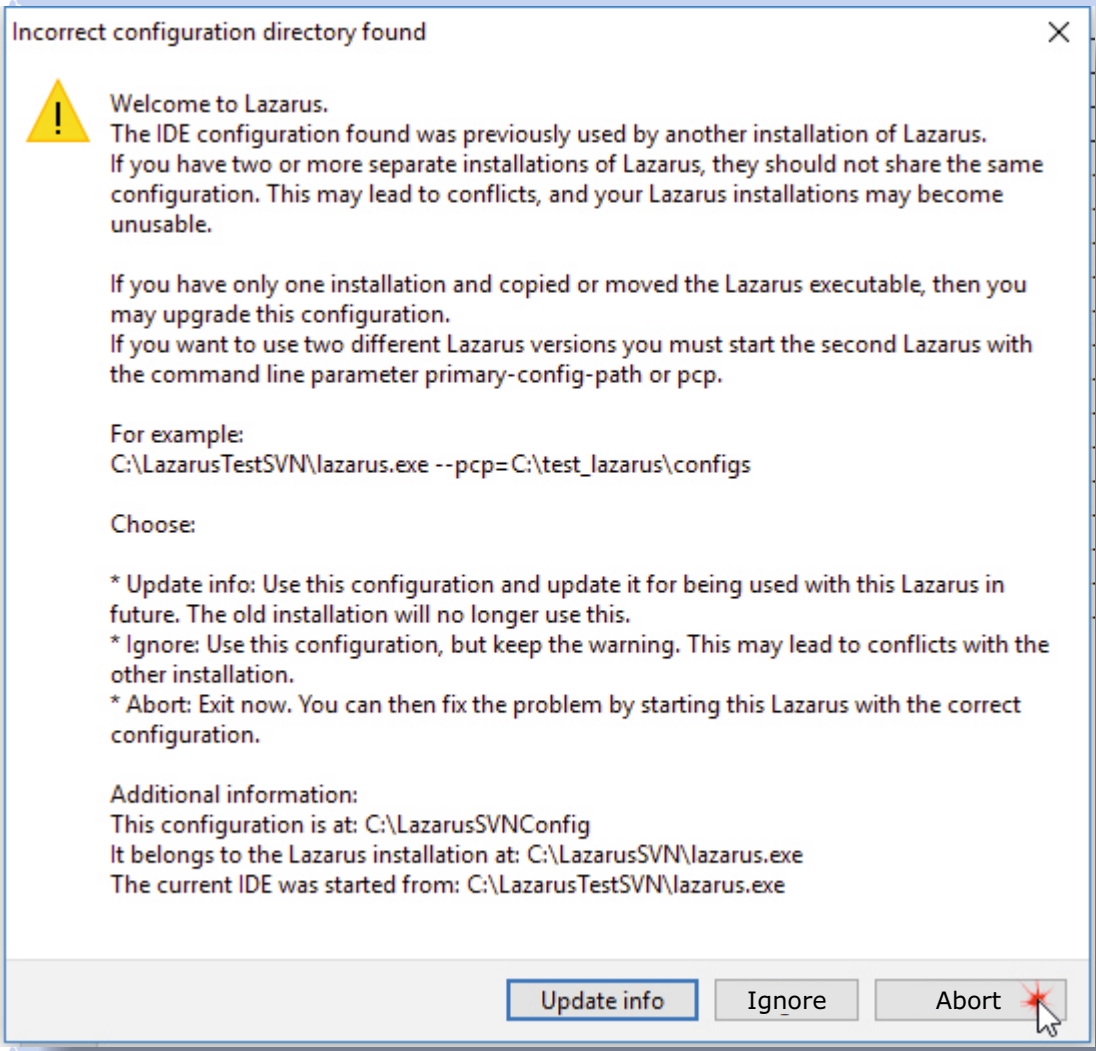
Figure 21: startlazarus is not found

And now finally go to Lazarus .exe and start again.



Finally the new version starts: 2.1

Figure 22: I delibarately created a wrong and duplicate version and the the following message shows up:



A good suggestion would be to create a Shortcut for this version of Lazarus. You can also give the shortcut a nice Icon:

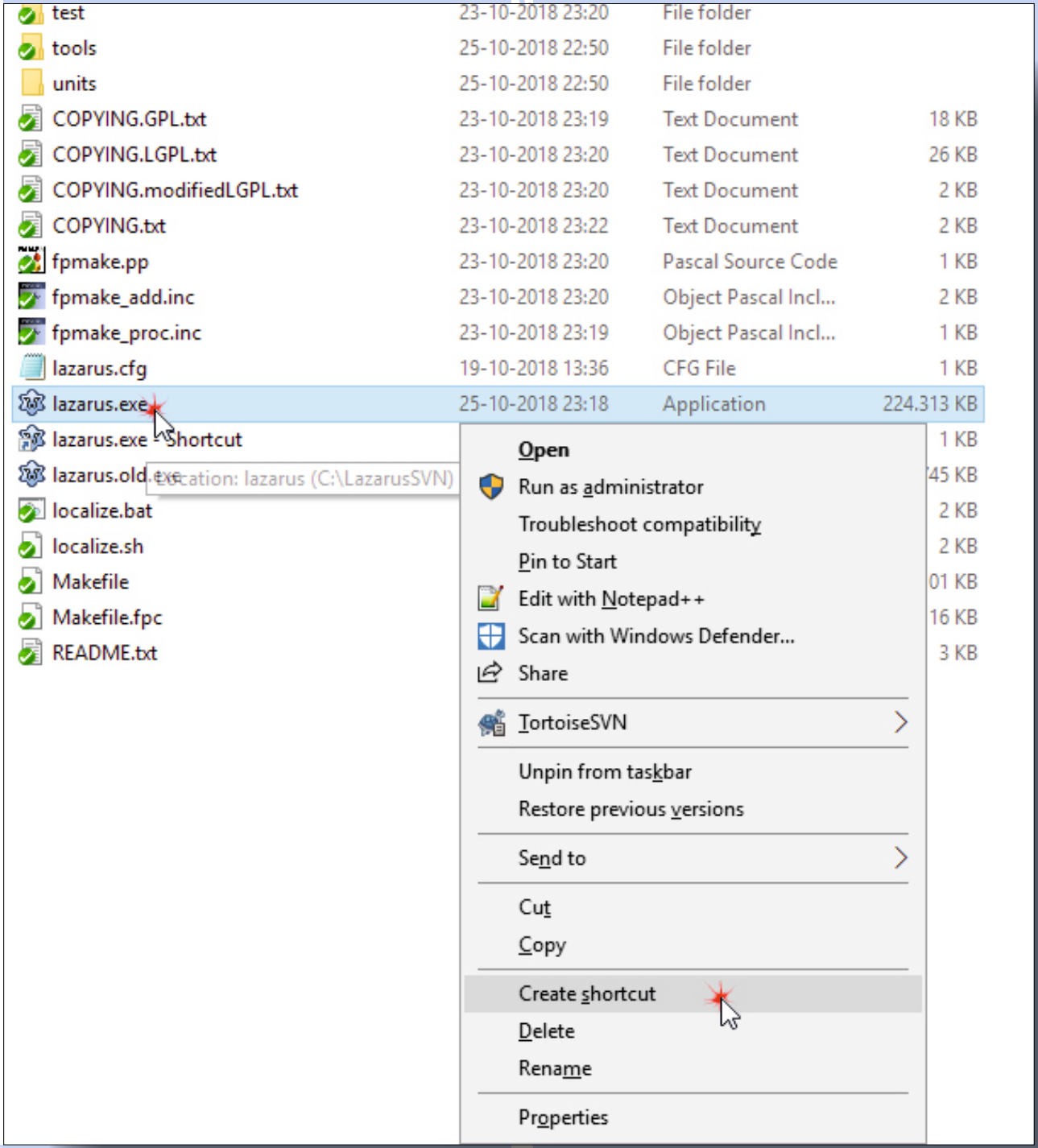


Figure 22: here is an example how to. Right click on the exe and you will get this menu:

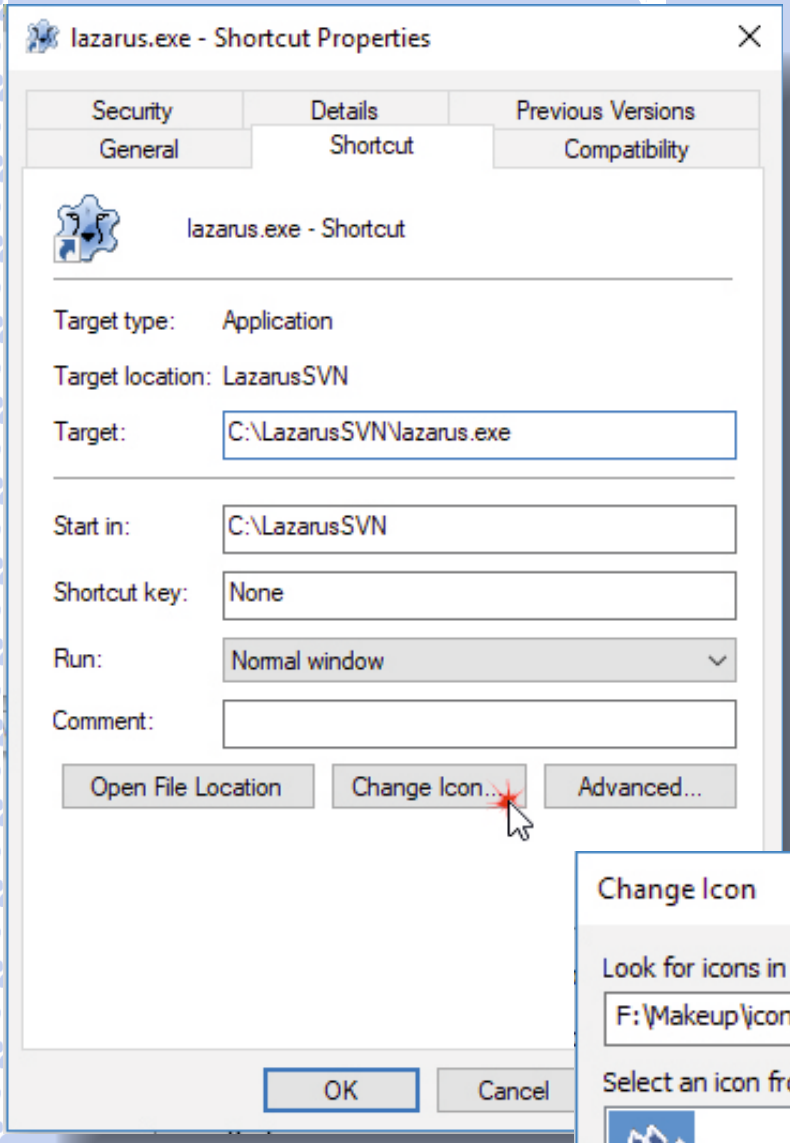


Figure 23: Change Icon

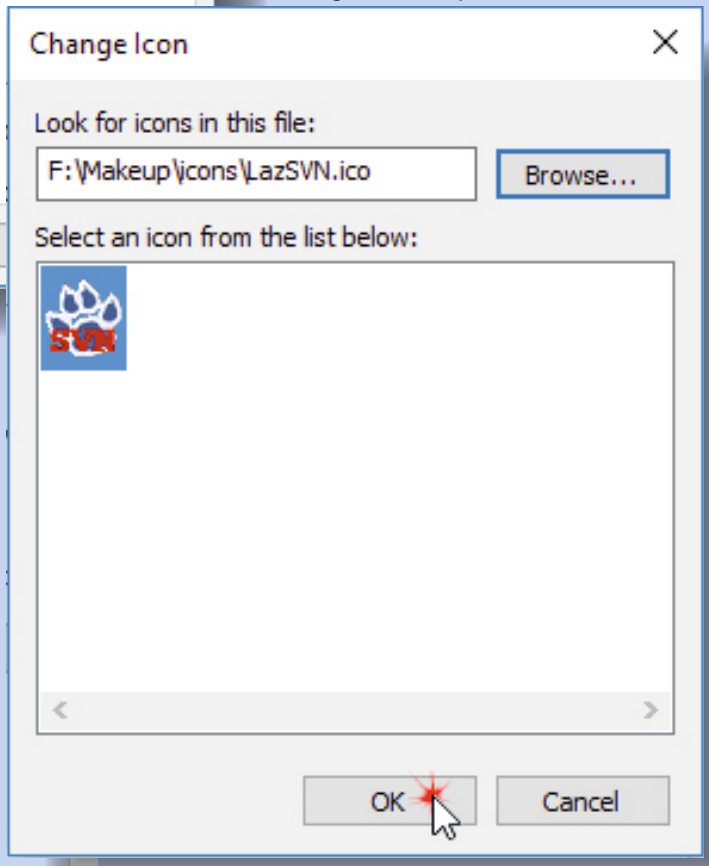


Figure 24: the path for the created ico



Figure 25: Icon created with the XN resource editor we provided some time before (Blaise 71)

General Features

This section describes some of the features of TortoiseSVN which apply to just about everything in the manual. Note that many of these features will only show up within a Subversion working copy.

One of the most visible features of **TortoiseSVN** is the icon overlays which appear on files in your working copy. These show you at a glance which of your files have been modified. Refer to Section 4.7.1, "Icon Overlays" in the helpfile to find out what the different overlays represent.

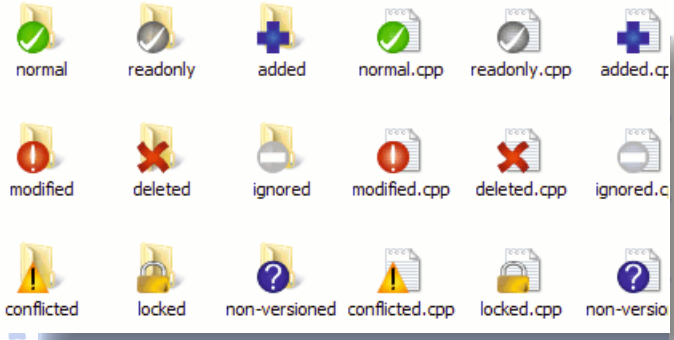


Figure 26: Explorer showing icon overlays

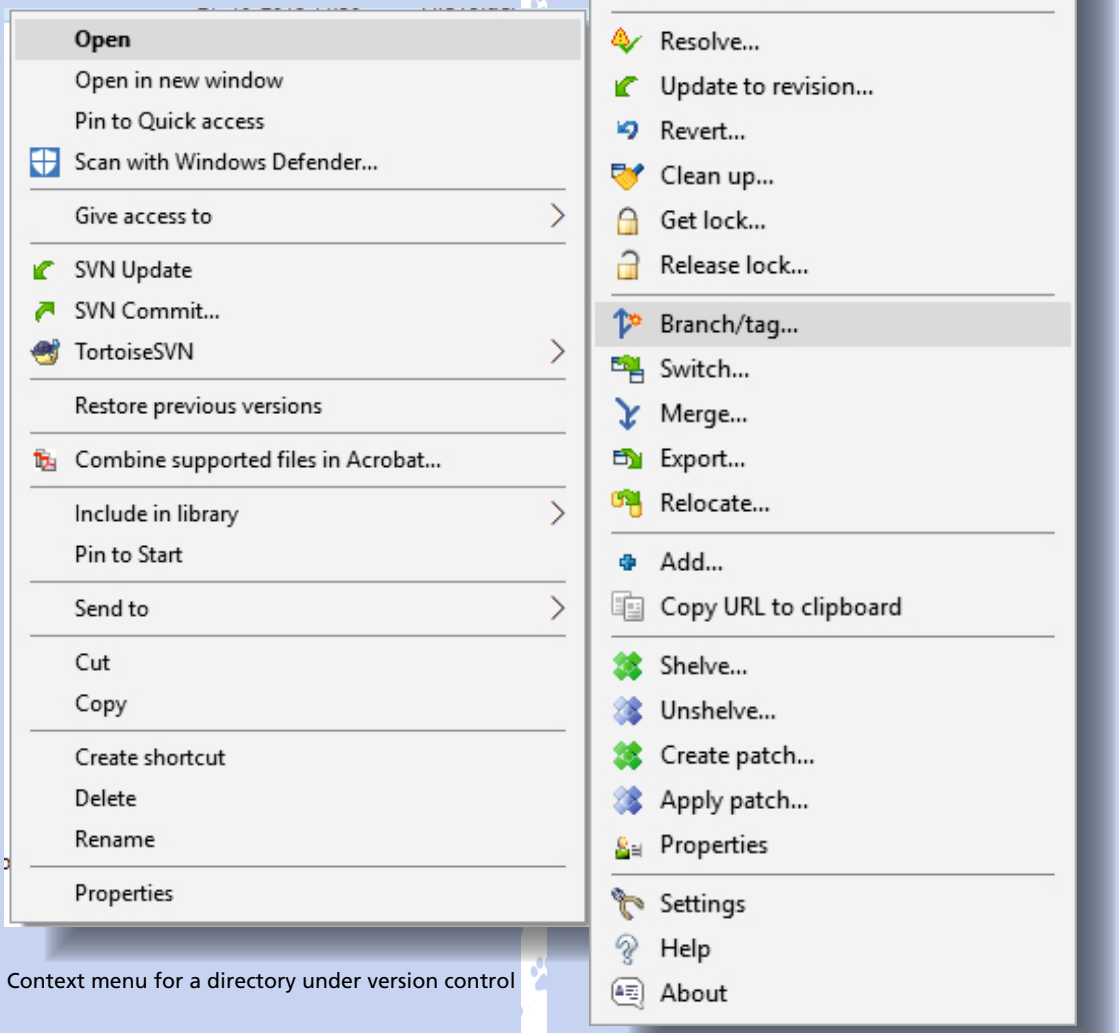


Figure 27: Context menu for a directory under version control

Icon Overlays

Now that you have checked out a working copy from a Subversion repository you can see your files in the windows explorer with changed icons. This is one of the reasons why TortoiseSVN is so popular. TortoiseSVN adds a so called overlay icon to each file icon which overlaps the original file icon. Depending on the Subversion status of the file the overlay icon is different.



A fresh checked out working copy has a green checkmark as overlay. That means the Subversion status is normal.



As soon as you start editing a file, the status changes to modified and the icon overlay then changes to a red exclamation mark.
That way you can easily see which files were changed since you last updated your working copy and need to be committed.



If during an update a conflict occurs then the icon changes to a yellow exclamation mark.



If you have set the svn:needs-lock property on a file, Subversion makes that file read-only until you get a lock on that file. Such files have this overlay to indicate that you have to get a lock first before you can edit that file.



If you hold a lock on a file, and the Subversion status is normal, this icon overlay reminds you that you should release the lock if you are not using it to allow others to commit their changes to the file.



This icon shows you that some files or folders inside the current folder have been scheduled to be deleted from version control or a file under version control is missing in a folder.



The plus sign tells you that a file or folder has been scheduled to be added to version control.



The bar sign tells you that a file or folder is ignored for version control purposes. This overlay is optional.



This icon shows files and folders which are not under version control, but have not been ignored. This overlay is optional.

NOTE: you may find that not all of these icons are used on your system. This is because the number of overlays allowed by Windows is very limited and if you are also using an old version of TortoiseCVS, then there are not enough overlay slots available. TortoiseSVN tries to be a "Good Citizen (TM)" and limits its use of overlays to give other apps a chance too.

I have been searching for a helpfile as PDF. The file on the website does not work. So we created a complete PDF file of all items. As a subscriber you can of course download this from your download page. See next page.
The file is very complete and one might say it is a complete book: 164

The Repository

Subversion is a centralized system for sharing information. At its core is a repository, **which is a central store of data.**
The repository stores information in the form of a filesystem tree - a typical hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others.

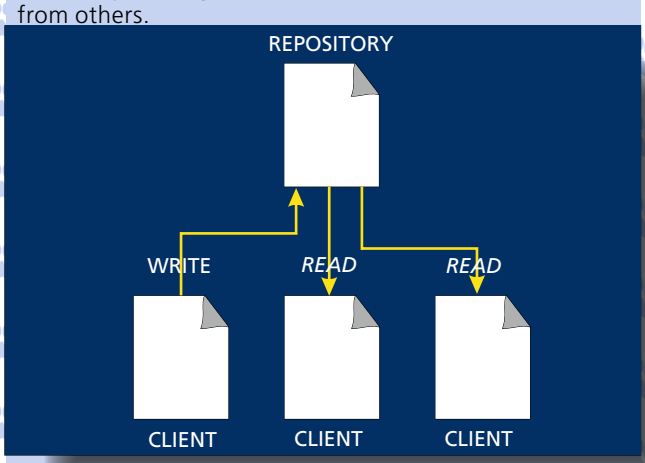


Figure 28: A Typical Client/Server System

What makes it interesting? So far, this seems like the definition of a typical file server. Yes, the repository is a kind of file server, but it's not usual.

ITS SUBVERSION REPOSITORY SPECIAL IS THAT IT REMEMBERS EVERY CHANGE EVER WRITTEN TO IT:

every change to every file, and even changes to the directory tree itself, such as the addition, deletion, and rearrangement of files and directories.

When a client reads data from the repository, it normally sees only the latest version of the filesystem tree.
But in this case the client also has the ability to view previous states of the filesystem.
For example, a client can ask historical questions like, **" What did this directory contain last Wednesday? "**, or **" Who was the last person to change this file, and what changes did they make? "**
This is a system which is designed to record and track changes to data over time.

Parts of this text are taken out of the helpfile

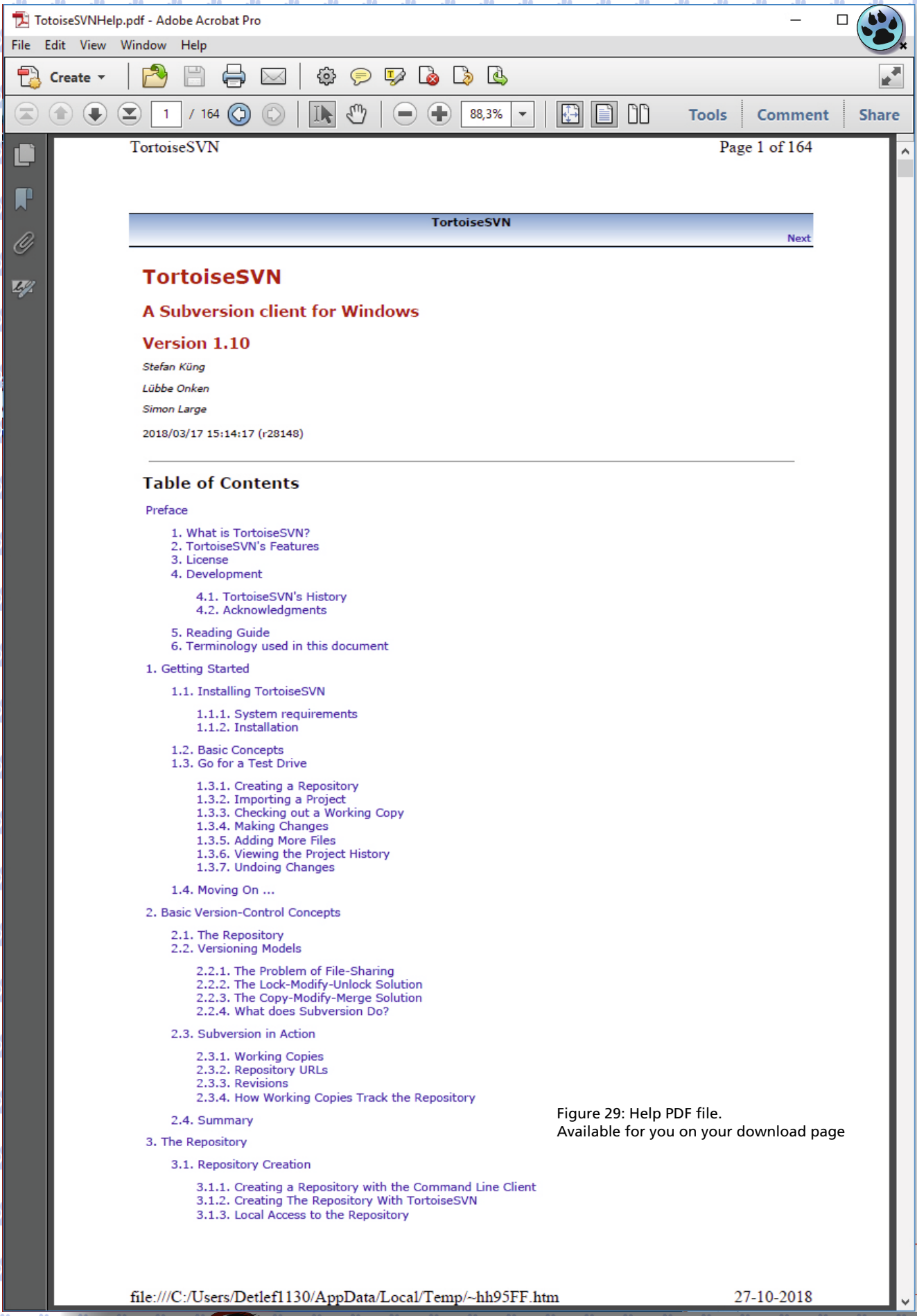


Figure 29: Help PDF file.
Available for you on your download page

Versioning Models

All version control systems have to solve the same fundamental problem: how will the system allow users to share information, but prevent them from accidentally stepping on each other's feet?
It's all too easy for users to **accidentally overwrite** each other's changes in the repository.

The Problem of File-Sharing

Consider this scenario: suppose we have two co-workers, **A** and **B**. They each decide to edit the same repository file at the same time. If **A** saves his changes to the repository first, then it's possible that (a few moments later) **B** could accidentally overwrite them with her own new version of the file. While **A** version of the file won't be lost forever (because the system remembers every change), any changes **A** made won't be present in **B's** newer version of the file, because she never saw **A's** changes to begin with. **A's** work is still effectively lost - or at least missing from the latest version of the file - and probably by accident. This is definitely a situation you would want to avoid!

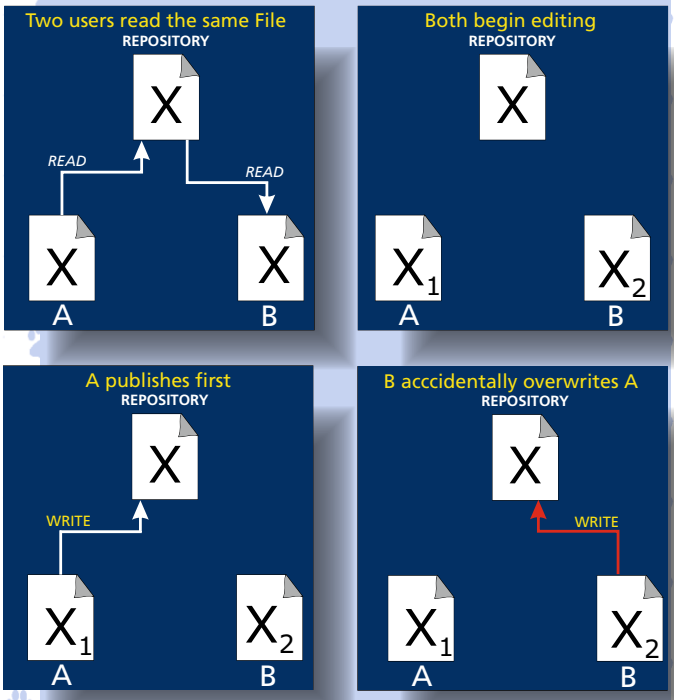


Figure 30: The Problem to Avoid

The Lock-Modify-Unlock Solution

Many version control systems use a lock-modify-unlock model to address this problem, which is a very simple solution. In such a system, the repository allows only one person to change a file at a time. First Harry must lock the file before he can begin making changes to it. Locking a file is a lot like borrowing a book from the library; if Harry has locked a file, then Sally cannot make any changes to it. If she tries to lock the file, the repository will deny the request.

All she can do is read the file, and wait for Harry to finish his changes and release his lock. After Harry unlocks the file, his turn is over, and now Sally can take her turn by locking and editing.

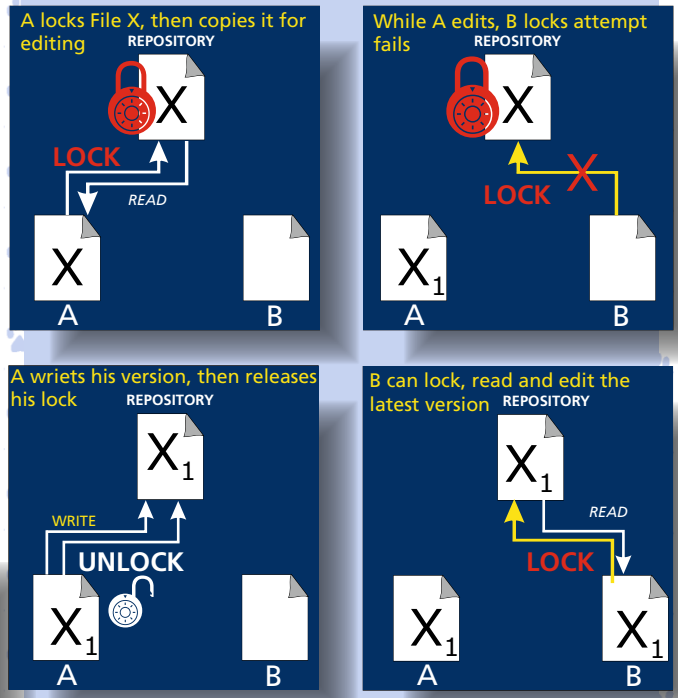


Figure 31: creating copies

The problem with the lock-modify-unlock model is that it's a bit restrictive, and often becomes a roadblock for users:

Locking may cause administrative problems. Sometimes Harry will lock a file and then forget about it. Meanwhile, because Sally is still waiting to edit the file, her hands are tied. And then Harry goes on vacation. Now Sally has to get an administrator to release Harry's lock. The situation ends up causing a lot of unnecessary delay and wasted time.

Locking may cause unnecessary serialization. What if Harry is editing the beginning of a text file, and Sally simply wants to edit the end of the same file? These changes don't overlap at all. They could easily edit the file simultaneously, and no great harm would come, assuming the changes were properly merged together. There's no need for them to take turns in this situation.

Locking may create a false sense of security. Pretend that Harry locks and edits file A, while Sally simultaneously locks and edits file B. But suppose that A and B depend on one another, and the changes made to each are semantically incompatible. Suddenly A and B don't work together anymore. The locking system was powerless to prevent the problem - yet it somehow provided a sense of false security. It's easy for Harry and Sally to imagine that by locking files, each is beginning a safe, insulated task, and thus inhibits them from discussing their incompatible changes early on.

The Copy-Modify-Merge Solution
Subversion, CVS, and other version control systems use a copy-modify-merge model as an alternative to locking. In this model, each user's client reads the repository and creates a personal working copy of the file or project. Users then work in parallel, modifying their private copies. Finally, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately a human being is responsible for making it happen correctly.

Here's an example. Say that A and B each create working copies of the same project, copied from the repository. They work concurrently, and make changes to the same file A within their copies. B saves her changes to the repository first. When A attempts to save his changes later, the repository informs him that his file A is out-of-date. In other words, that file A in the repository has somehow changed since he last copied it. So Harry asks his client to merge any new changes from the repository into his working copy of file A. Chances are that Sally's changes don't overlap with his own; so once he has both sets of changes integrated, he saves his working copy back to the repository.

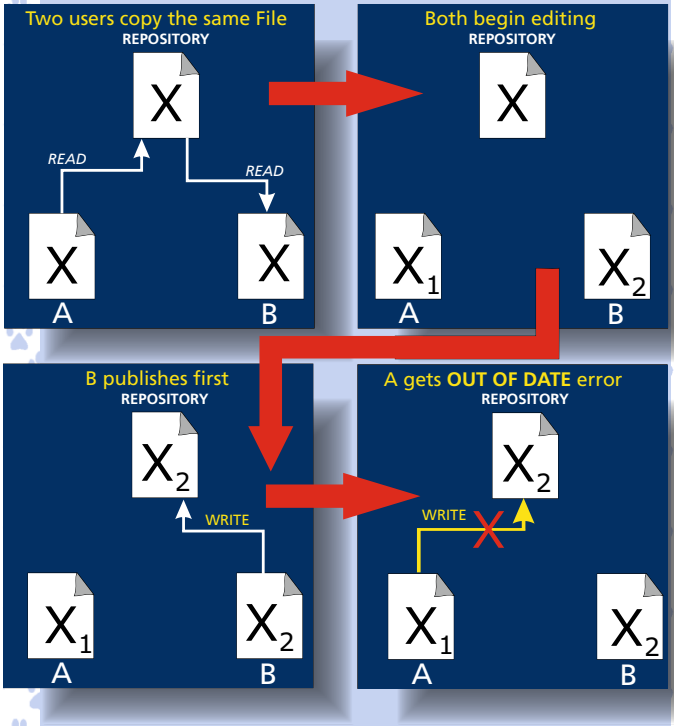


Figure 32 The Copy-Modify-Merge Solution

But what if **B's** changes do overlap with **A's** changes? This situation is called a conflict, and it's usually not much of a problem. When **A** asks his client to merge the latest repository changes into his working copy, his copy of file X is somehow flagged as being in a state of conflict: **he'll be able to see both sets of conflicting changes, and manually choose between them.**

Note that software can't automatically resolve conflicts; only humans are capable of understanding and making the necessary intelligent choices.
Once A has manually resolved the overlapping changes (**perhaps by discussing the conflict with B!**) he can without problem save the merged file back to the repository.
The **copy-modify-merge model** may sound a bit chaotic, it runs extremely smoothly. Users can work in parallel, never waiting for one another. When they work on the same files, it turns out that most of their concurrent changes don't overlap at all; conflicts are infrequent.

And the amount of time it takes to resolve conflicts is far less than the time lost by a locking system.

In the end, it all comes down to one critical factor: user communication. When users communicate poorly, both syntactic and semantic conflicts increase. No system can force users to communicate perfectly, and no system can detect semantic conflicts. So there's no point in being lulled into a false promise that a locking system will somehow prevent conflicts; in practice, locking seems to inhibit productivity more than anything else.
There is one common situation where the lock-modify-unlock model comes out better, and that is where you have unmergeable files. For example if your repository contains some graphic images, and two people change the image at the same time, there is no way for those changes to be merged together. Either A or B will lose their changes.

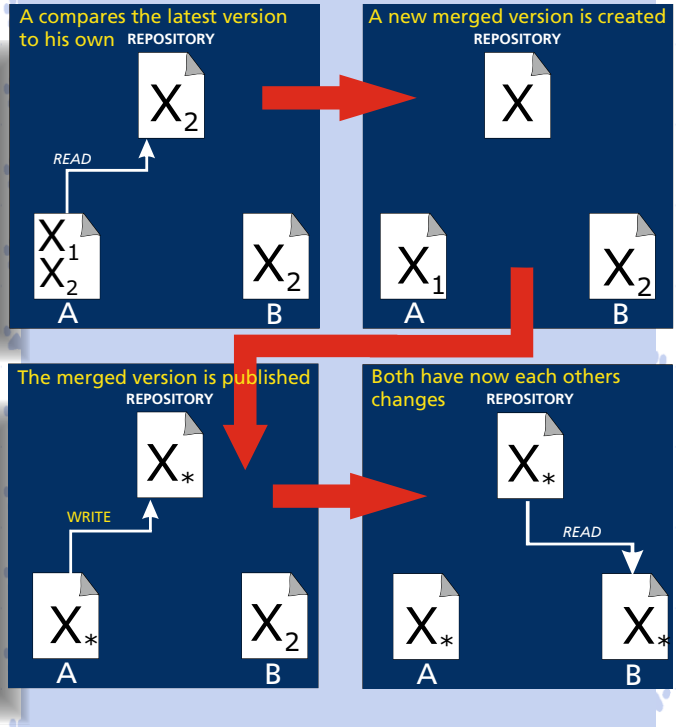


Figure 33 ...Copy-Modify-Merge Continued

What does Subversion Do?

Subversion uses the copy-modify-merge solution by default, and in many cases this is all you will ever need. However, as of Version 1.2, Subversion also supports file locking, so if you have unmergeable files, or if you are simply forced into a locking policy by management, Subversion will still provide the features you need.

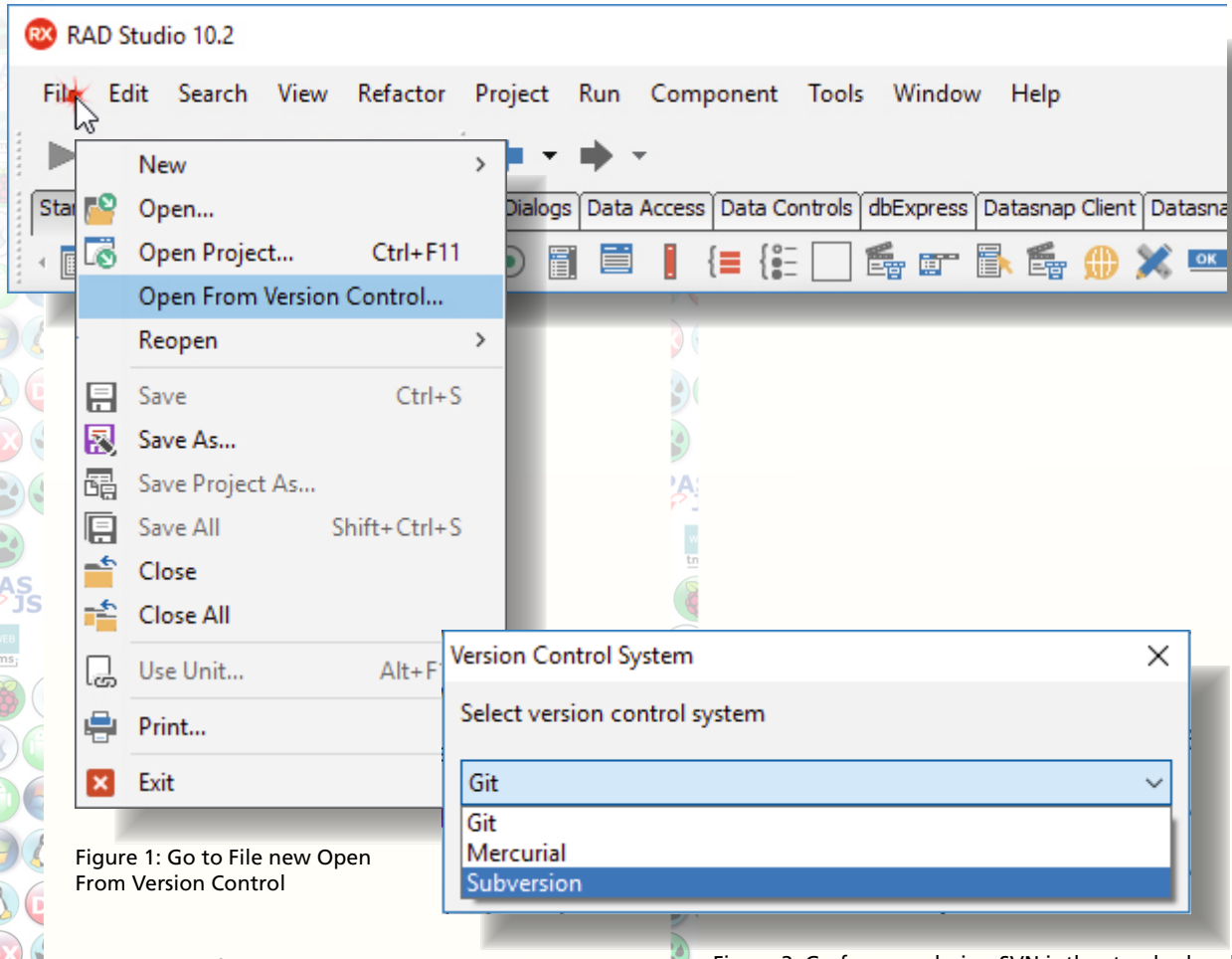


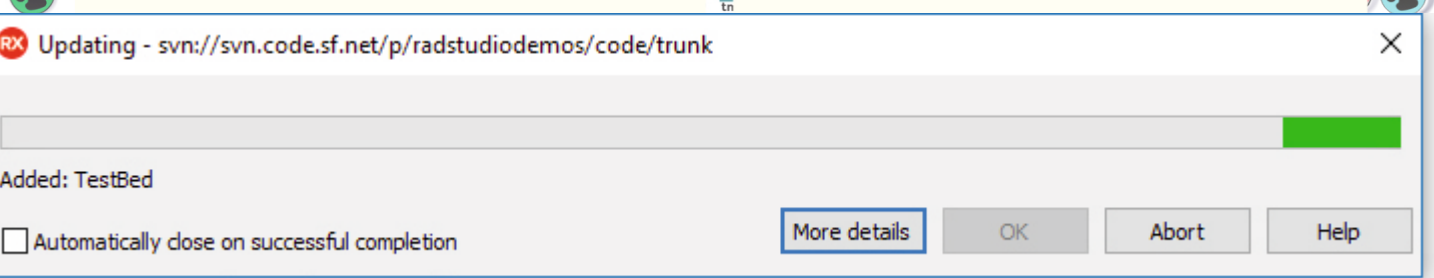
Figure 1: Go to File new Open From Version Control

Figure 2: Go for your choice, SVN is the standard and build in

SubVersioning for Delphi is already built in and very easy to use. Go to File -> Open From Version Control.
A small window (figure 2) will open which gives you the ability to make choices.
Built in is SVN Tortoise and you will have to provide extra details if you would like to add an other one of the other two options: Git and or Mercurial.
In the images you can see the choices that you can make. **For SVN you do not need to give the address of the executable because that is pre defined.**
What you in essence need to do, is give the address of the server where you master version is located, so each time you will do a checkout you have the next step of your program.

It helps tremendously and makes stepping backwards very easy. You could even have several computers with Delphi where you have the passibility of using parts of an older version of your program. That is because you can in this way work with a team - which eventually could be be copies of your selves. (trying to be funny...)

Of course you will have to do some preparations:
If you do not already have that, you need to create a new Directory on your server where you can put you basisc versions of all the programs your working on.
And of course you will have to create your own (local) version of the program.



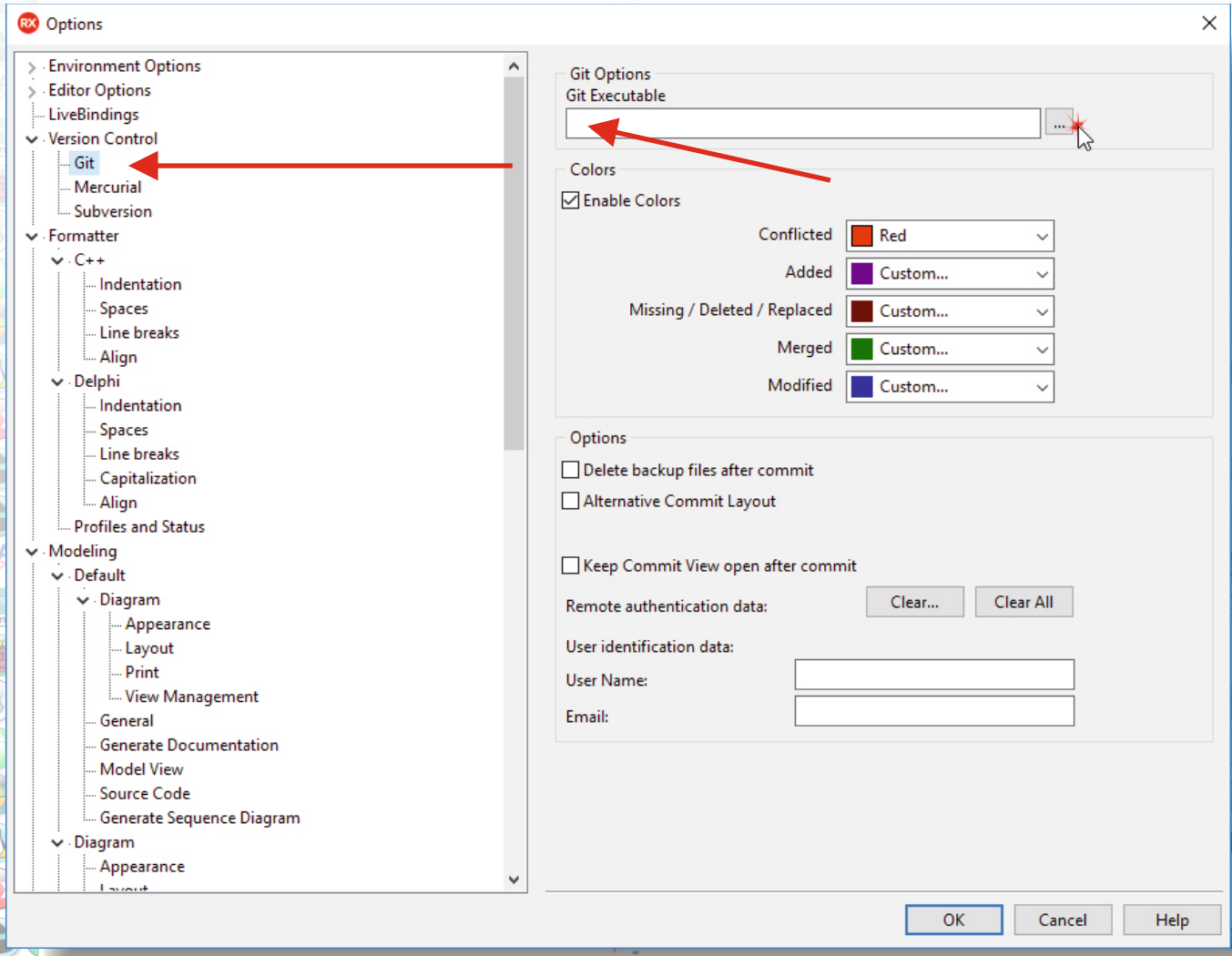
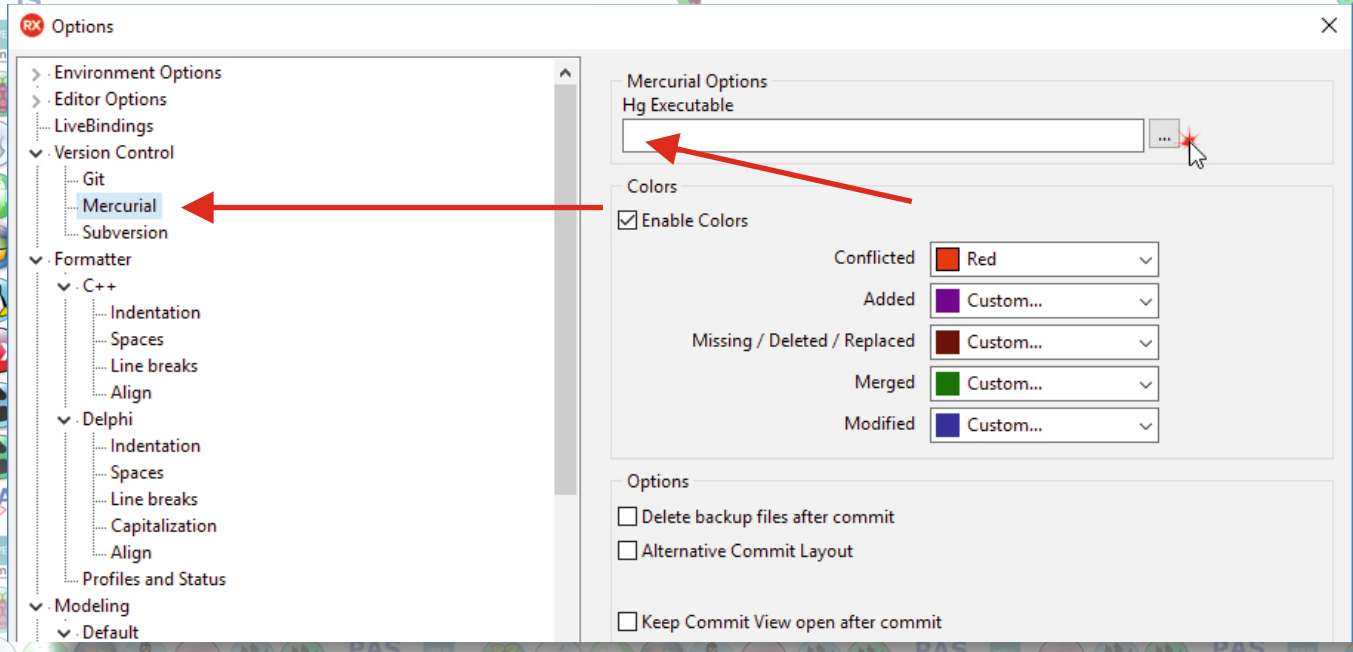


Figure 3: Git and Mercurial options, under tools.
A separate way of making other choices



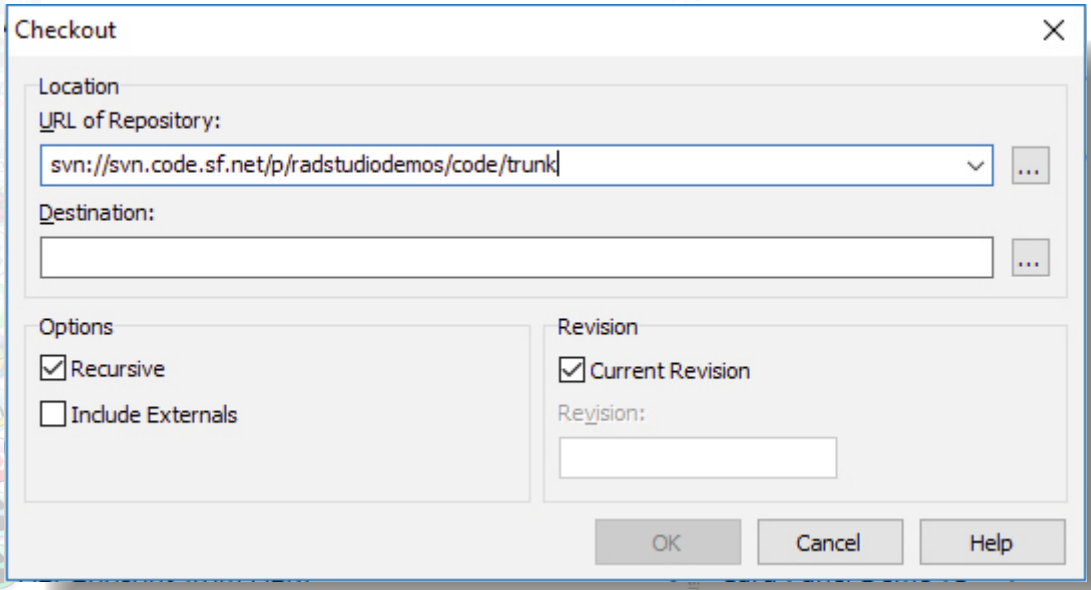
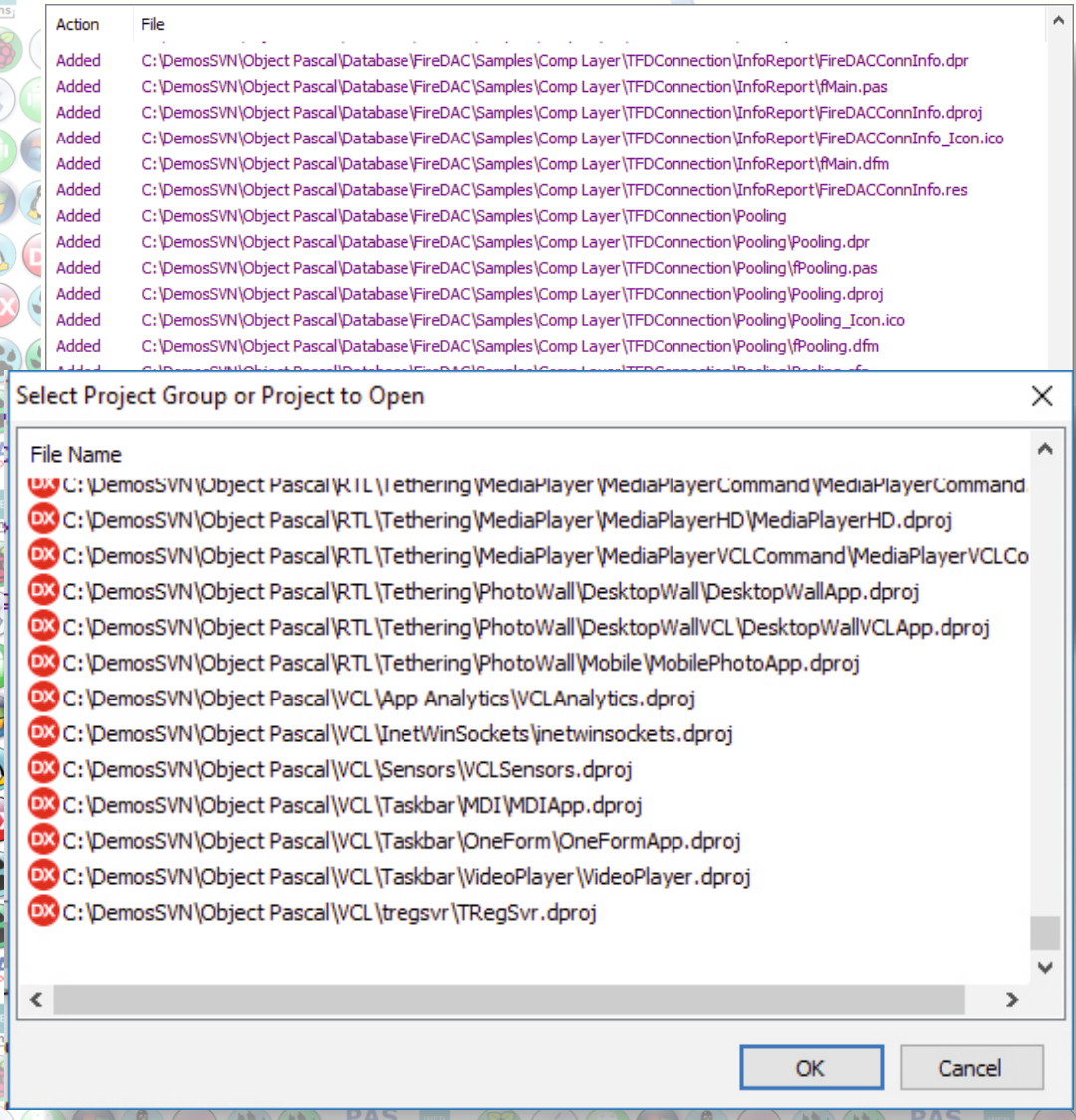


Figure 4: her you find the URL I chose. A good way to see what all is already available to learn lessons from **`svn://svn.code.sf.net/p/radstudiodemos/code/trunk`**.



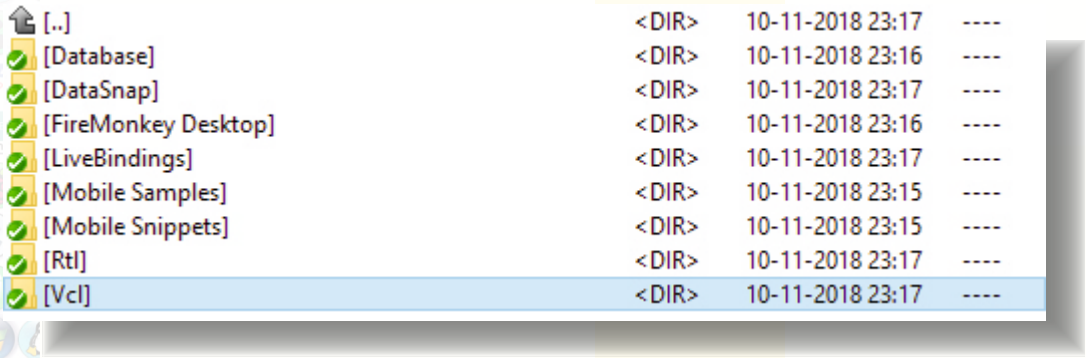


Figure 5: The content of the directory I mentioned

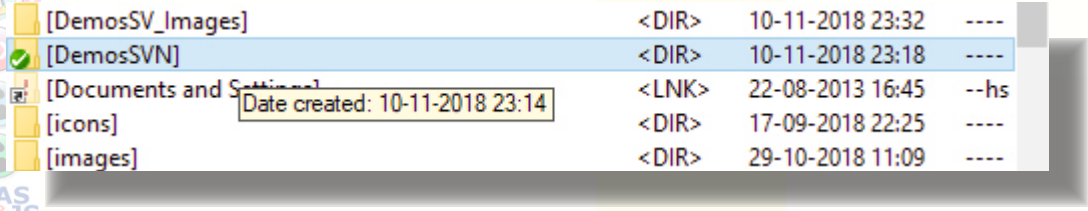
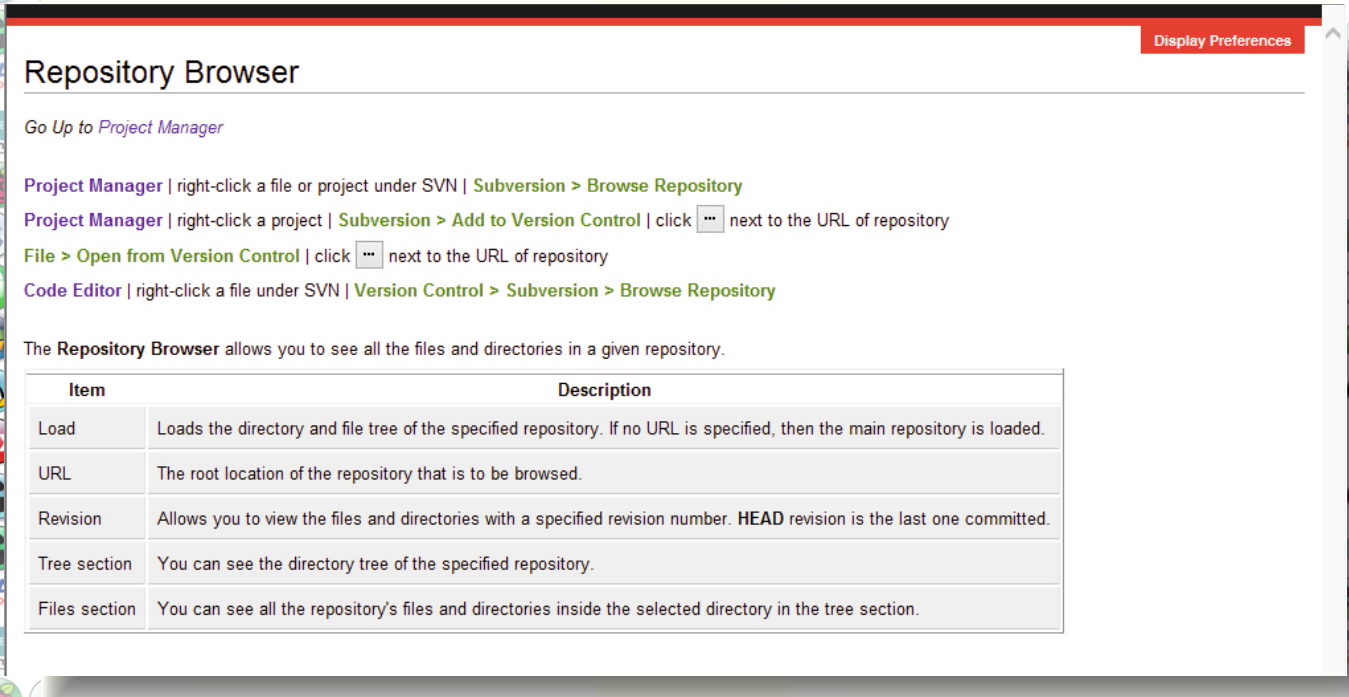


Figure 6: The name of the Dir where I keep the demo projects



Figure 7: Drill down and see the content of the project I chose to have a closer look at...

It is of course necessary to have a close look at the options the help provides. Getting there is very easy: click help and go to **"Repository Browser"**, or go to **"Versioning"**



Display Preferences

Repository Browser

Go Up to Project Manager

Project Manager | right-click a file or project under SVN | **Subversion > Browse Repository**

Project Manager | right-click a project | **Subversion > Add to Version Control** | click ··· next to the URL of repository

File > Open from Version Control | click ··· next to the URL of repository

Code Editor | right-click a file under SVN | **Version Control > Subversion > Browse Repository**

The **Repository Browser** allows you to see all the files and directories in a given repository.

Item	Description
Load	Loads the directory and file tree of the specified repository. If no URL is specified, then the main repository is loaded.
URL	The root location of the repository that is to be browsed.
Revision	Allows you to view the files and directories with a specified revision number. **HEAD** revision is the last one committed.
Tree section	You can see the directory tree of the specified repository.
Files section	You can see all the repository's files and directories inside the selected directory in the tree section.

LAZARUS 2.0 NEWS AND OVERVIEW WITH SOME DETAILS

This is an overview of the new elements ofthe latest lazarus version 2.0(release candidate2), how to install it you can find starting at page 17 of this issue Just to make it easy for you yo can download this also from our website and try it. Theprogram for free nad you can ues it in any way you want. Most people do not realize that it isvery good at using sveral intances, so might use always an other versionfor separate projects. Below you can see what the page looks like: for Linux / Mac / Windows:

(https://sourceforge.net/projects/lazarus/files/
http://www.lazarus-ide.org/index.php?page=checksums

LAZARUS PROFESSIONAL CONFERENCE

NEW LAZARUS CONFERENCE IN BONN (COLOGNE) GERMANY 2019

https://sourceforge.net/projects/lazarus/files/

SOURCEFORGE

Sick and tired of unstable test results?
> CI COULD DO BETTER_
TeamCity – automatic detection and reporting of flaky tests
by JetBrains
Advertisement - Report

Lazarus

Rapid applications development tool and libraries for FPC

Brought to you by: mgaertner, mhess, user4martin, vlx, vsnijders

Summary | Files | Reviews | Support | Wiki | News | External Link | Mailing Lists

Download Latest Version
lazarus-1.8.4-fpc-3.0.4-win32.exe (136.1 MB)

Get Updates

Home

Name	Modified	Size	Downloads / Week
Lazarus Zip _ GZip	2018-10-29		481
Lazarus Linux SRC RPM	2018-10-29		226
Lazarus Linux i386 RPM	2018-10-29		358
Lazarus Linux amd64 DEB	2018-10-29		1,729
Lazarus Linux i386 DEB	2018-10-29		575
Lazarus Linux x86_64 RPM	2018-10-29		558
Lazarus Mac OS X i386	2018-10-29		1,091
Lazarus Documentation	2018-10-29		149
LazarusWindows 64bits	2018-10-28		4,900
LazarusWindows 32bits	2018-10-28		4,116
Lazarus Mac OS X powerpc	2014-10-12		99
Totals: 11 Items			14,282

To start with: all attendees of the last
Lazurus Meetup/Pascon receive automatically a Credit Card USBstick with the latest Lazarus revision. This is standard procedure for those who attend any PasCon / Meetup for Delphi and Lazarus.
This version contains the latest Magazine on Disk as wel the Lzarus 2.0 and the Delphi community Edition.
For the Delphi edition you need to register!

`TScrollingWinControl (TForm, TScrollBox, TFrame)`
`ScreenToClient` and **`ClientToScreen`** are now calculated without scrollbar offset.
Done for Delphi compatibility.
Support for **mouse wheel horizontal scrolling**
Added TControl events for horizontal mouse wheel moving (special mice with horz scroll buttons for PC, and modern mice for Mac). **`OnMouseWheelHorz, OnMouseWheelLeft, OnMouseWheelRight.`**
Events are used in some non native LCL components (TreeView, UpDown, SpinEdit).
Native components do handle horz scrolling already.

Compiler defines to exclude some graphics support
Added flags to exclude support for some graphics formats to create smaller applications:
`-dDisableLCLGIF`
`-dDisableLCLJPEG`
`-dDisableLCLPNM`
`-dDisableLCLTIFF`

`TCustomImageList / TImageList`
the image list now supports multiple resolutions of one image.
See Multiple-resolution TImageList in Lazarus 1.9 and newer for more details.
As a result all LCL controls support High-DPI glyphs on Windows+Linux and Retina on Mac without any additional code.
every LCL control that supports **`ImageList`** has now a new **`[Images]Width`** property to decide what custom width at 96 PPI (100% scale) is to be used.
Example: `TToolBar.Images/ImageWidth, TListView.LargeImages/LargeImagesWidth`
set the **`TCustomImageList.Scaled=True`** property to let the image list automatically pick up the right resolution for your control.

`TSpeedButton, TBitBtn`
New properties Images, ImageIndex and ImageWidth. With them full ImageList support was added.
no need to save the same Glyph in LFM all over the application - automatic high-DPI image handling
`TWinControl.DoubleBuffered, .ParentDoubleBuffered` and **`TApplication.DoubleBuffered`**
Note: DoubleBuffered is a LCLWin32-only feature
Old behavior: DoubleBuffered wasn't properly implemented and it was forced True: **New behavior:**
A Delphi-compatible DoubleBuffered/ParentDoubleBuffered concept was created (it is equal to the Font/ParentFont concept).

Explanation:

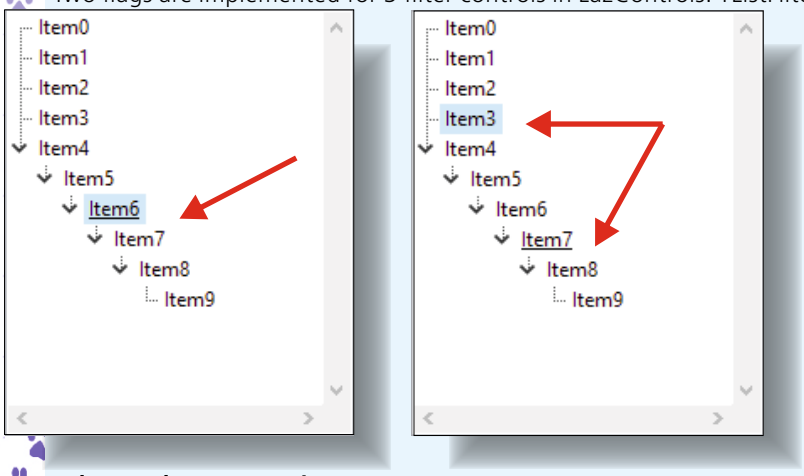
Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first. When DoubleBuffered is false, the windowed control paints itself directly to the window. When DoubleBuffered is true, the windowed control paints itself to an in-memory bitmap that is then used to paint the window. Double buffering reduces the amount of flicker when the control repaints, but is more memory intensive.
When a windowed control is a dock site and has an associated dock manager, it must be double-buffered. Note: Some controls, such as TRichEdit, can't paint themselves into a bitmap. For such controls, DoubleBuffered must be set to false.
The LCL has the TApplication.DoubleBuffered extension over Delphi that allows you to set form's default DoubleBuffered value globally for the whole application
(*set Application.DoubleBuffered before creating the first form*). The value is then applied to all controls on the form with ParentDoubleBuffered:=True.
DoubleBuffered is True by default unless in remote session (this is different to Delphi where DoubleBuffered is False by default).
If you need one specific control to be DoubleBuffered:=False even for Application.DoubleBuffered:=adbDefault, make sure you set control.DoubleBuffered:=False and .ParentDoubleBuffered:=False.

`ListView`
Added CustomSort method like in Delphi.

`TTreeView`
Implemented **HotTrack** property (**show blue underline** for item under cursor).
Implemented auto-scrolling up/down during drag-drop (when mouse reaches the edge of treeview).
FilterOptions in filter controls
Base class for filter controls **`TCustomControlFilterEdit`** **has new property FilterOptions**, which is set of such flags:

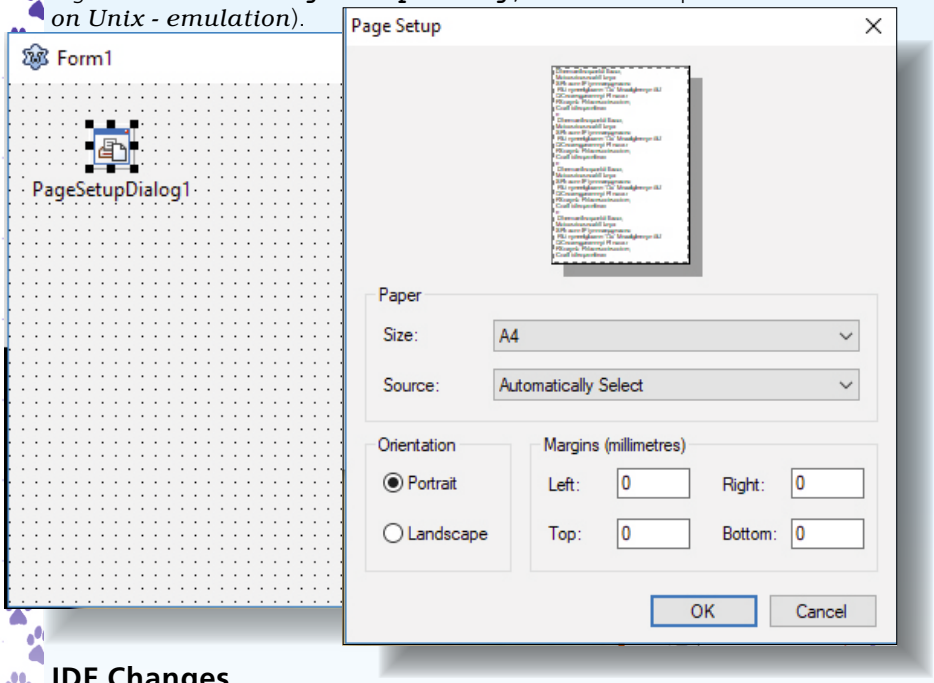
```
TFilterStringOption = (fsoCaseSensitive, fsoMatchOnlyAtStart);
TFilterStringOptions = set of TFilterStringOption;
```

Two flags are implemented for 3 filter controls in LazControls: TListFilterEdit, TListViewFilterEdit, TTreeFilterEdit.



Advanced menus Assign
Supported **`TMenuItem.Assign(TMenuItem):`** this copies all props of menuitem.
Supported **`TMenu.Assign(TMenu):`**
this **copies all items with all nested subitems** from one menu to another (**`TMainMenu, TPopupMenu`**).

`TPageSetupDialog` *(Printer)*
`TPageSetupDialog:` added Margin* and Units properties.
Big rework of Unix **`TPageSetupDialog,`** it is now complete like it's on Windows (*on Windows it was native, on Unix - emulation*).



IDE Changes

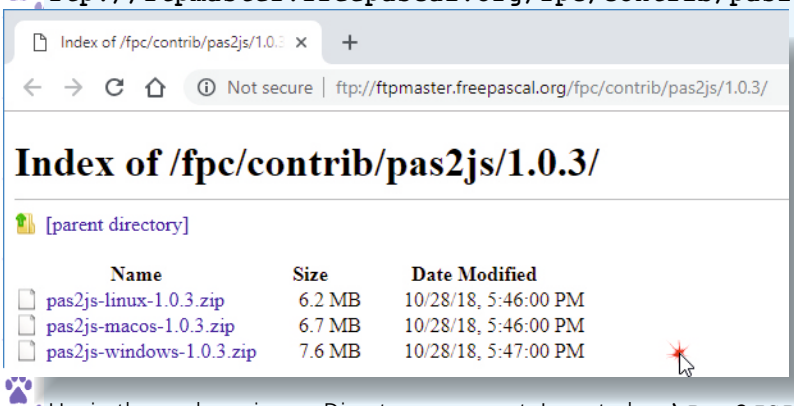
- **Several High-DPI IDE improvements and retina support on Cocoa**
- Delphi **Attributes:** Find declaration, parameter hints, $modeswitch prefixedattributes.
- **The IDE parses the custom compiler options** for the fpc switch -FN<namespaces>

PAS2JS support:

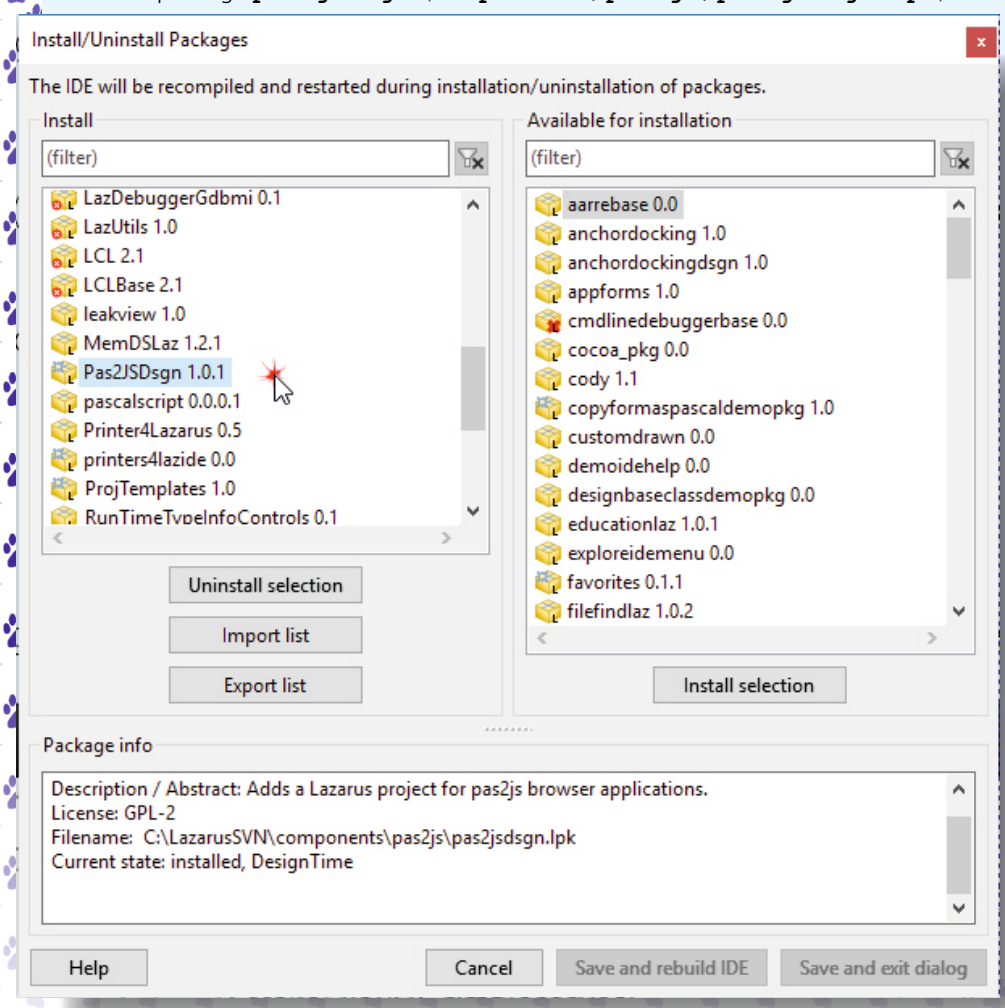
- Added IDE package pas2jsdsgn:
  - create a browser or nodejs webapplication
  - on Run start the webapp in your webbrowser
- Pas2Js **settings are automatically fetched**, same as fpc settings
  quickfixes work with pas2js messages

Lazarus PAS2JS integration

Requires Lazarus 1.9+ and either a PAS2JS snapshot or **svn checkout** (*see pages SVN Tortoise starting at page17*).
To use PAS2JS you need to do the following: Go to:
`ftp://ftpmaster.freepascal.org/fpc/contrib/pas2js/1.0.3/`. Here you can make your choice and download the file(s). You need this otherwise your project will not start.



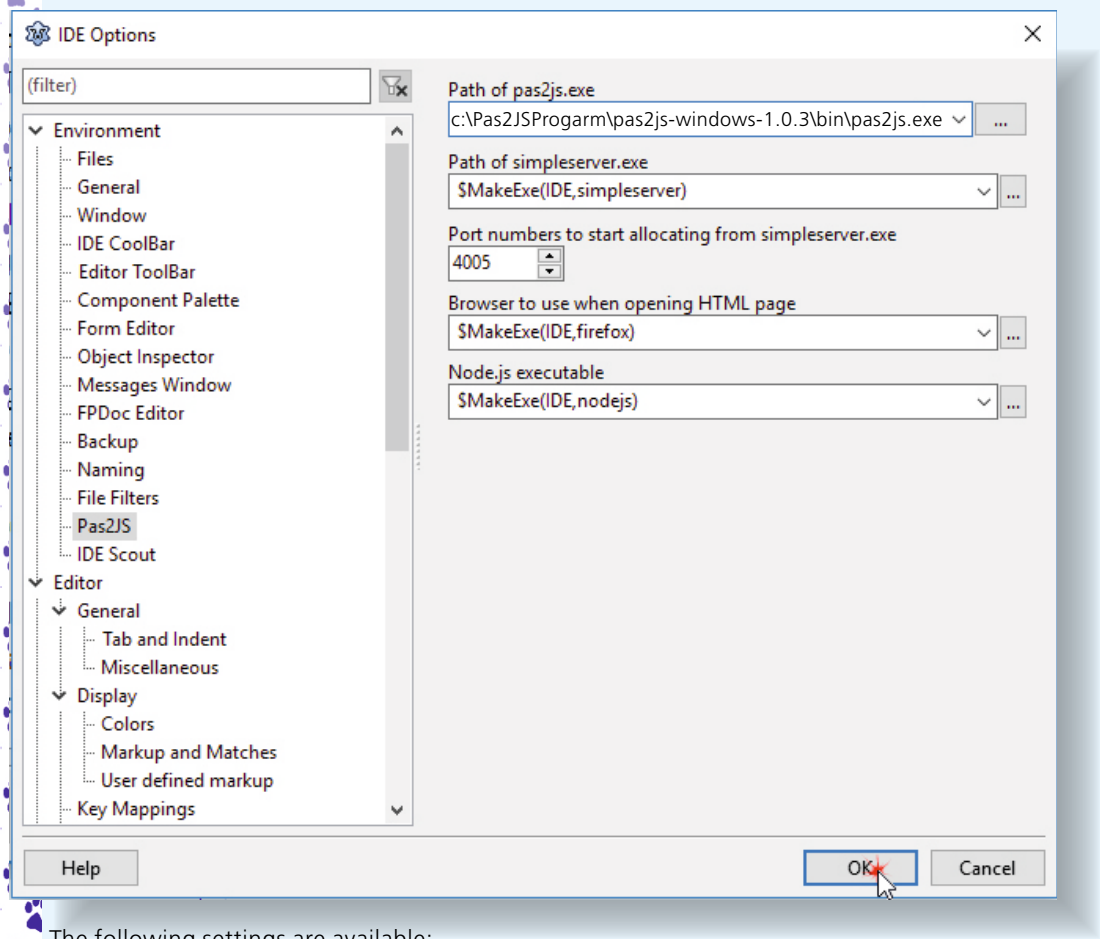
Unzip the package in any Directory you want. I created **`c:\Pas2JSProgram\`** for the **`pas2js.exe file.`**
This path you need for your PA2JS settings and for the saving of the future project :**`c:\LazPas2JsProject\`**
Install the package **`pas2jsdsgn`** (**`components/pas2js/pas2jsdsgn.lpk`**) and **restart the IDE.**



Then set the path of pas2js (**pas2jsfolder\compiler\utils\pas2js\pas2js.exe**)
or in my case **c:\Pas2JSProgarm\pas2js-windows-1.0.3\bin\pas2js.exe**
Go to **Tools / Options / Environment / Pas2JS / Path of pas2js.**
The Lazarus pas2js integration has 3 parts to it:

1. Global options
2. Two project wizards
3. A debug window

Global options
In the Tools/Options... dialog, the **PAS2JS** section allows you to set some options that affect the IDE integration. It looks like this:



The following settings are available:
Path of pas2js.
This is the compiler binary used when setting up a new project.
It will be searched in the PATH if no absolute path is set. Corresponds to IDE macro Pas2JSJS.
Path of simpleserver.
This is the webserver that is started when a project is run that needs a webserver.
This is by default the simpleserver application from the FPC project, but can be another webserver, as long as it accepts the -p option to set the path, and it serves files from the directory in which it was started..
Port numbers to start allocating from. Every time you start a new webserver project, a new port number is allocated for the webserver. (you can still edit this in the new project dialog).
Browser to use when opening HTML page.
The IDE will use this browser to open your HTML page. It will be searched in the PATH if no absolute path is set. Corresponds to IDE macro Pas2JSBrowser.
Node.js executable.
The IDE will use this Node.js executable to start a Node.js project.
It will be searched in the PATH if no absolute path is set. Corresponds to IDE macro Pas2JSNodeJS.

Project Wizards

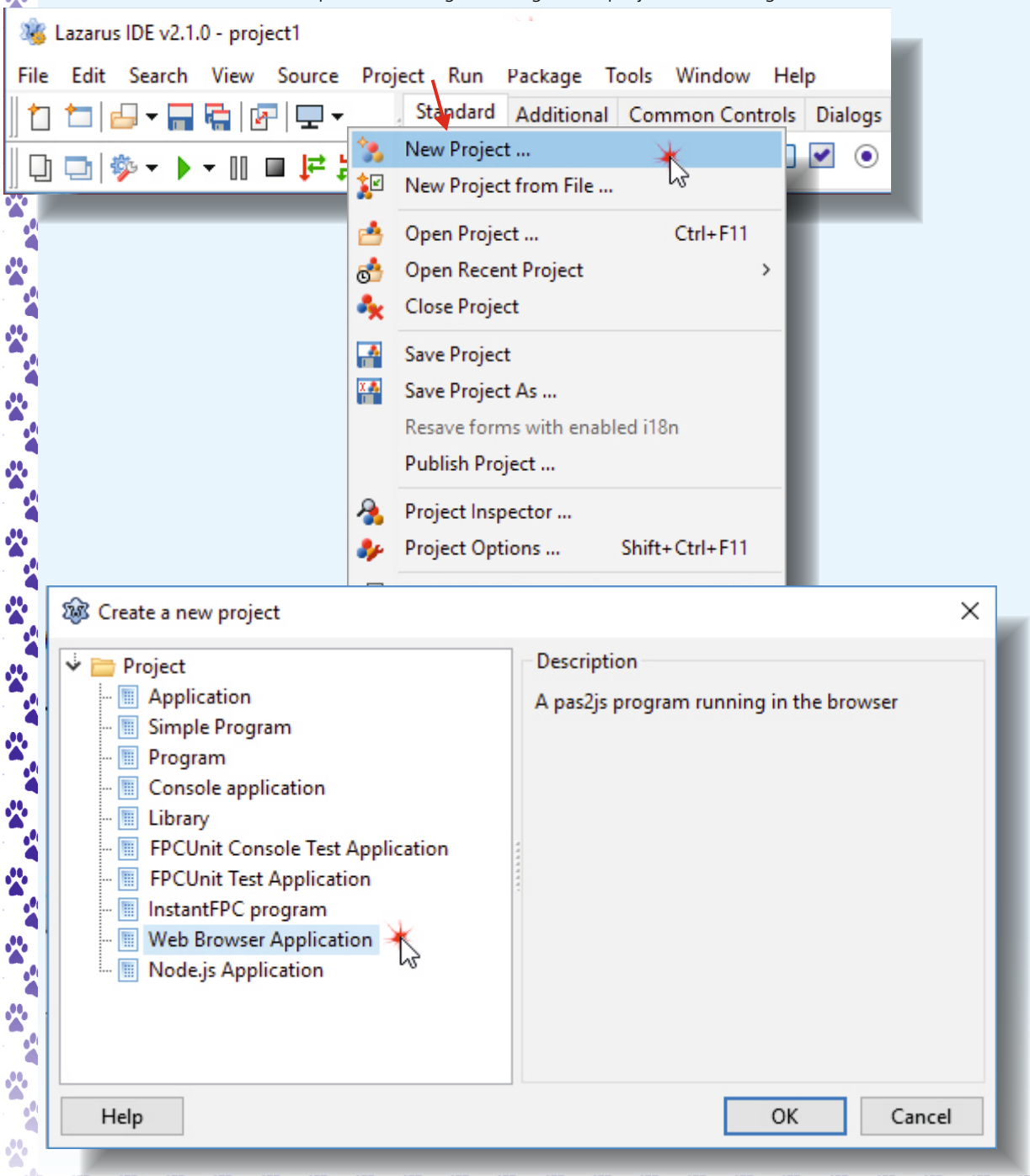
- The PAS2JS support is in the **`pas2jsdsgn.lpk`** package, which you can find in the **`components/pas2js`** directory. It registers 2 wizards in the **'New project'** dialog:
  - Web Browser Application
  - Node.js Application

Besides creating an initial project source, both options will also

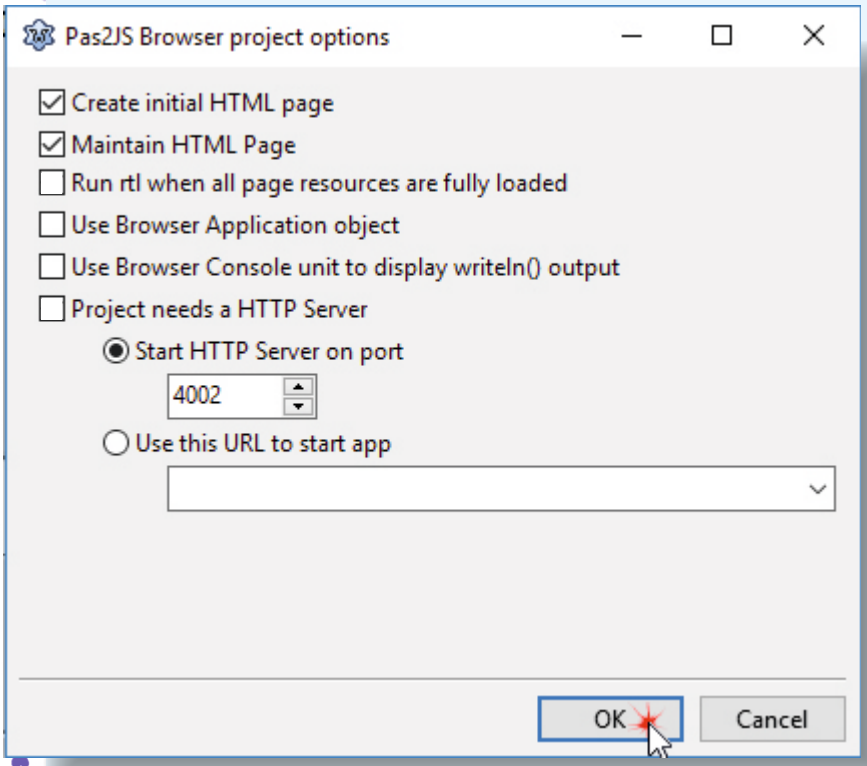
- **set up the compiler command** for compiling with **PAS2JS.**
- **change the 'Run' command** so the **'Run without debugging'** option works: it will open a project in the browser or run it with nodejs.

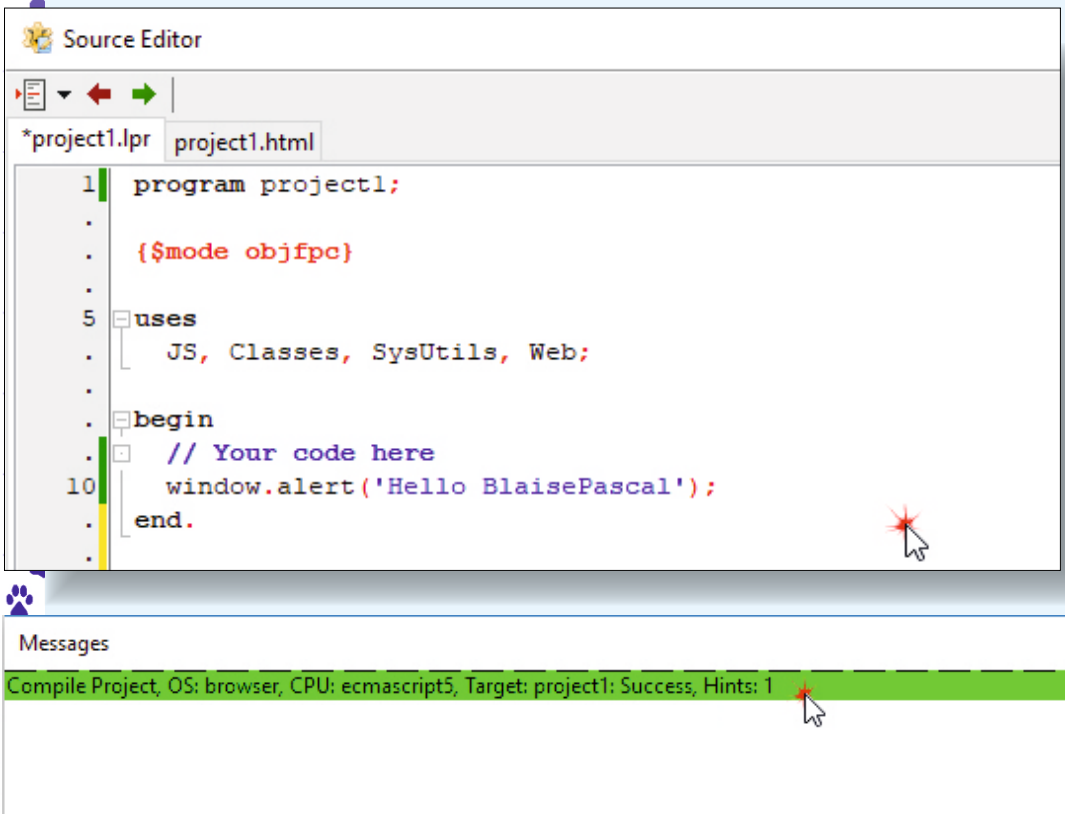
New Web Browser application

This wizard will ask for some options before generating a new project. The dialog is shown below.



PAS2JS

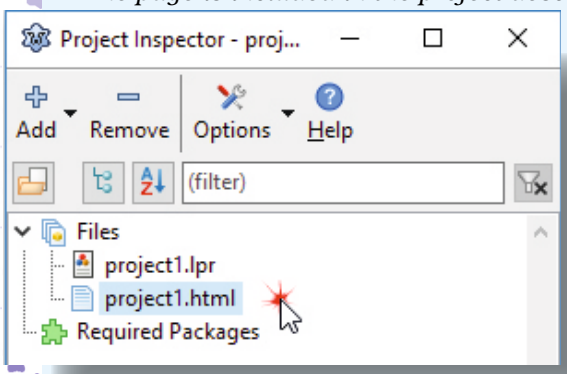

Pas2JS Browser project options
Create initial HTML page
Maintain HTML Page
Run rtl when all page resources are fully loaded
Use Browser Application object
Use Browser Console unit to display writeln() output
Project needs a HTTP Server
Start HTTP Server on port
4002
Use this URL to start app
OK
Cancel

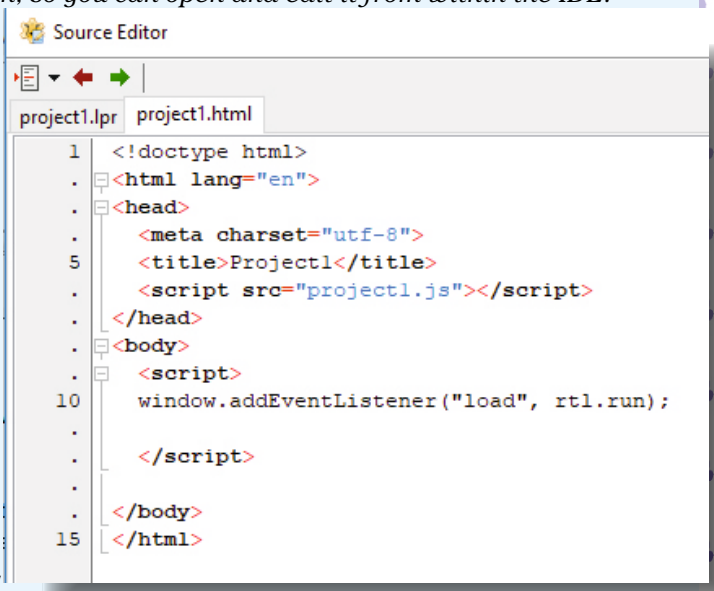

Source Editor
*project1.lpr
project1.html
1 program project1;
{$mode objfpc}
5 uses
JS, Classes, SysUtils, Web;
begin
// Your code here
10 window.alert('Hello BlaisePascal');
end.

Messages
Compile Project, OS: browser, CPU: ecmascript5, Target: project1: Success, Hints: 1

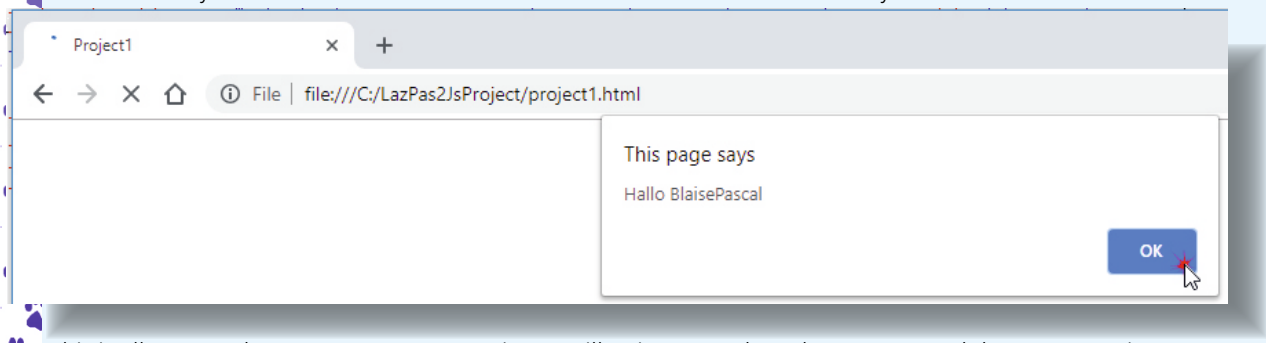
PAS2JS The following options are available:

- **Create initial HTML page.** This is self-explanatory,
  the IDE will **generate a template HTML** page which includes all that is necessary to run the **PAS2JS** code.
  The page is included in the project description, so you can open and edit it from within the IDE.





Just after you have done all the initial settings like I described before you simply create your first project and after compilng, which happens through **Run -> Compile** (not using the symbol that is not yet working) so use **Ctrl + F9** and then save your project and start a browser:
All you need to do is start **Chrome** or **Firefox** is simlpy use the **Ctrl+ O** commad so you can show the Browser where to find your HTML file and then click it and then the browser will show you the result.



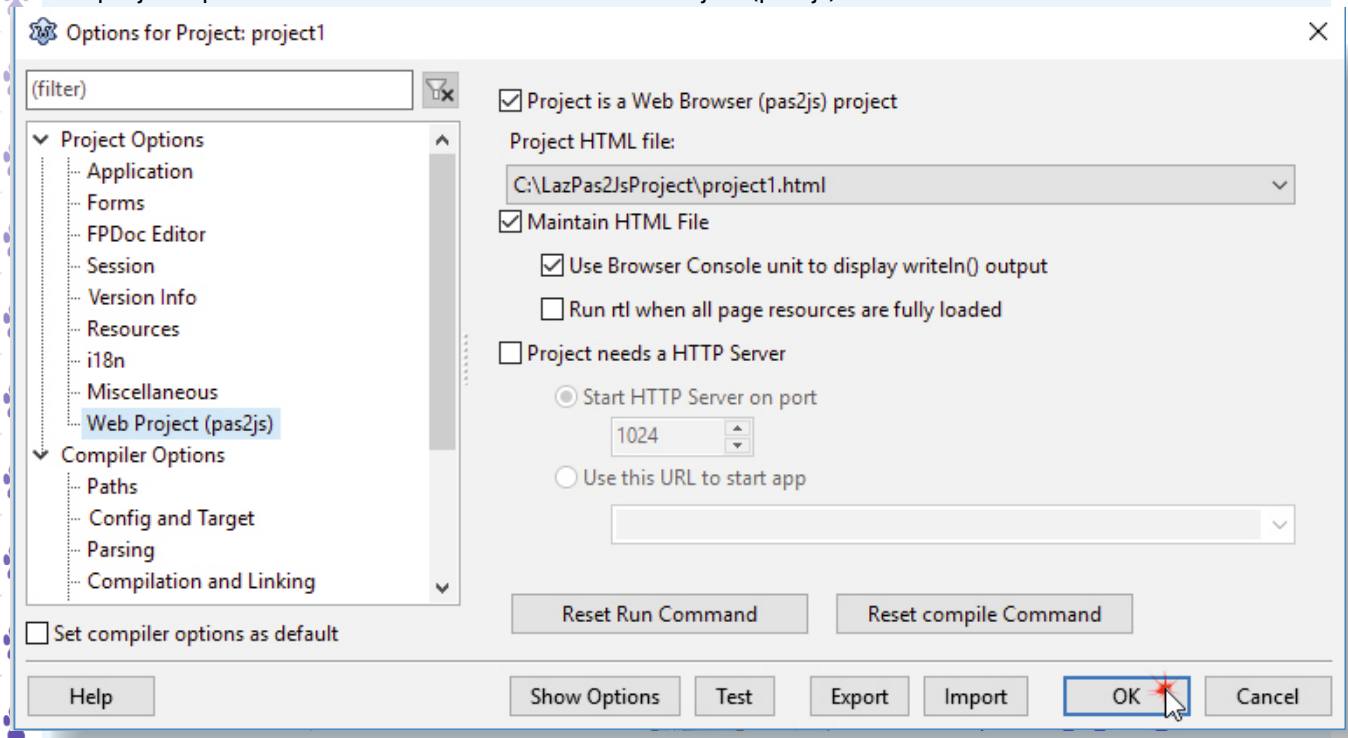
This is all you need to start your Pas2Js Project. I will write more about how to use and do some exercises.
Ther is aslo an example directory written which come wit the zipfile:**`c:\Pas2JSProgram\pas2js-windows-1.0.3\demo\`**

- **Maintain HTML page.**
  If you change the name of the project, the IDE will update the references in the HTML File.
  (all your changes will be lost)
- Run rtl in **`document.onReady`**.
  By default the script tag that starts the ball rolling will call rtl.run().
  Checking this option changes the script, so the **`rtl.run`** is run
  in the **`HTML document.onReady`** event instead.
  This is necessary if your code contains startup code that references elements in the HTML.
  The elements will only be available after the **`onReady`** event.
- **Use Browser Application object.**
  This changes the code to use the **`TBrowserApplication`** object.
  This is a **`TCustomApplication`** descendent which offers support for query parameters etc.
  as if they were command-line parameters.
- **Use Browser Console unit**.
  Checking this will simply include the browserconsole unit in the uses clause.
  This hooks into the **`system units writeln`** command: any output will be appended to the HTML.
  the generated **HTML** has a div with the correct ID to which the output is appended.
- **Project needs a HTTP server.**
  If the project needs a HTTP server, then the IDE can start one for you,
  or you can indicate the URL to use when the 'Run without debugging' command is used.

Project options

You can change the settings chosen in the new project wizard in the Project Settings dialog.
It also allows you to convert any existing (simple) project to a web application project.

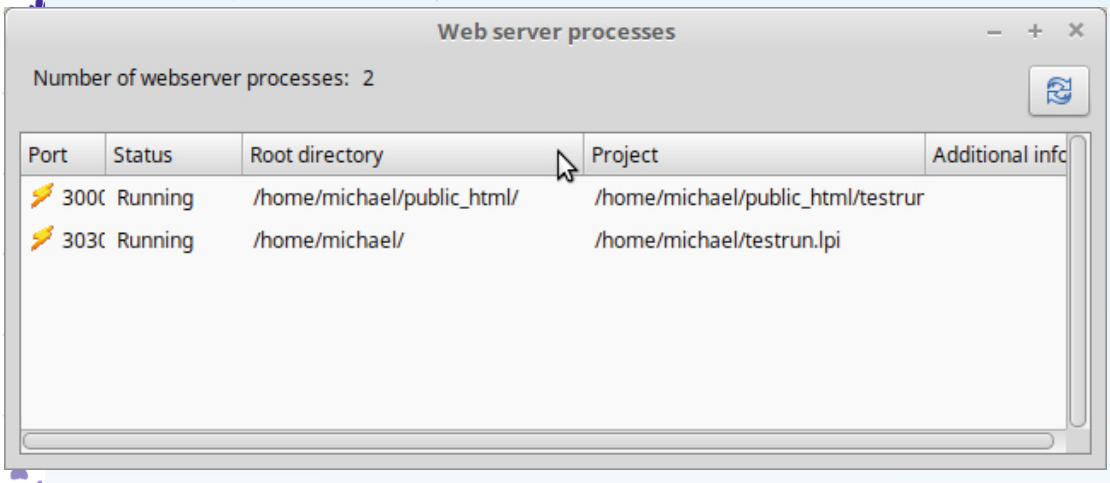
The project options are located at the bottom "Web Project (pas2js)'



The checkbox at the top marks the project as web browser project.
All other options have the same meaning as their counter parts in the 'New project' wizard.

The ```Reset compile command``` and ```Reset run command``` set the respective commands to their default values.

Debug window *(havent started using this one will be in an extra article where and how to use this.)*
As you start projects from various directories, the IDE will start webservers as needed. After some time, there can be several webservers running. An overview of running webservers can be shown using the the View/Debug Windows/Pas2JS Webservers menu item.
It shows a list of processes started by the IDE:

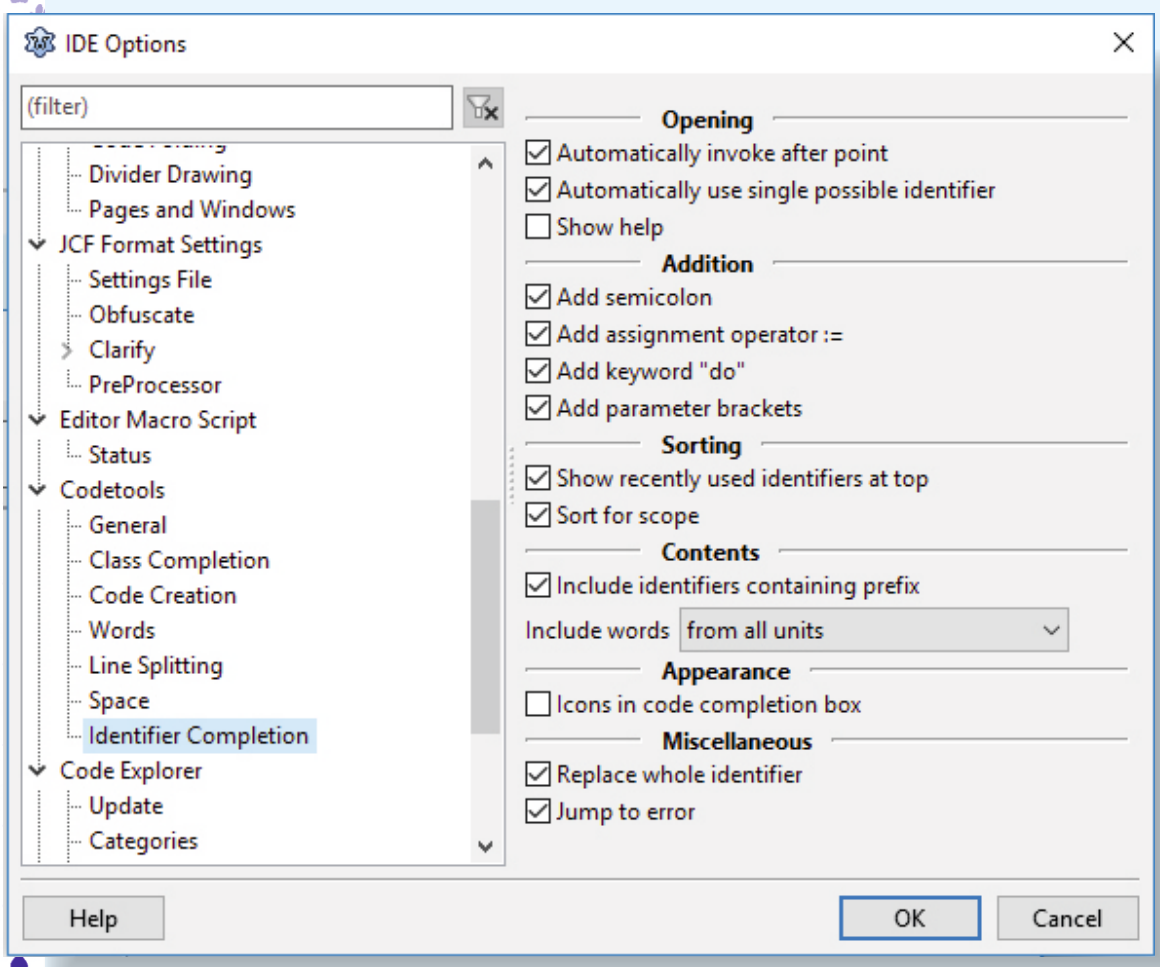


Support for stopping/starting processes will still be added, as well as statistics for number of requests etc.

IDE Changes

- added quickfix for fpc message "inherited method is hidden":
  add modifier override, overload, or reintroduce shortcuts.
- added designer menu item to hide icons for components like TOpenDialog.

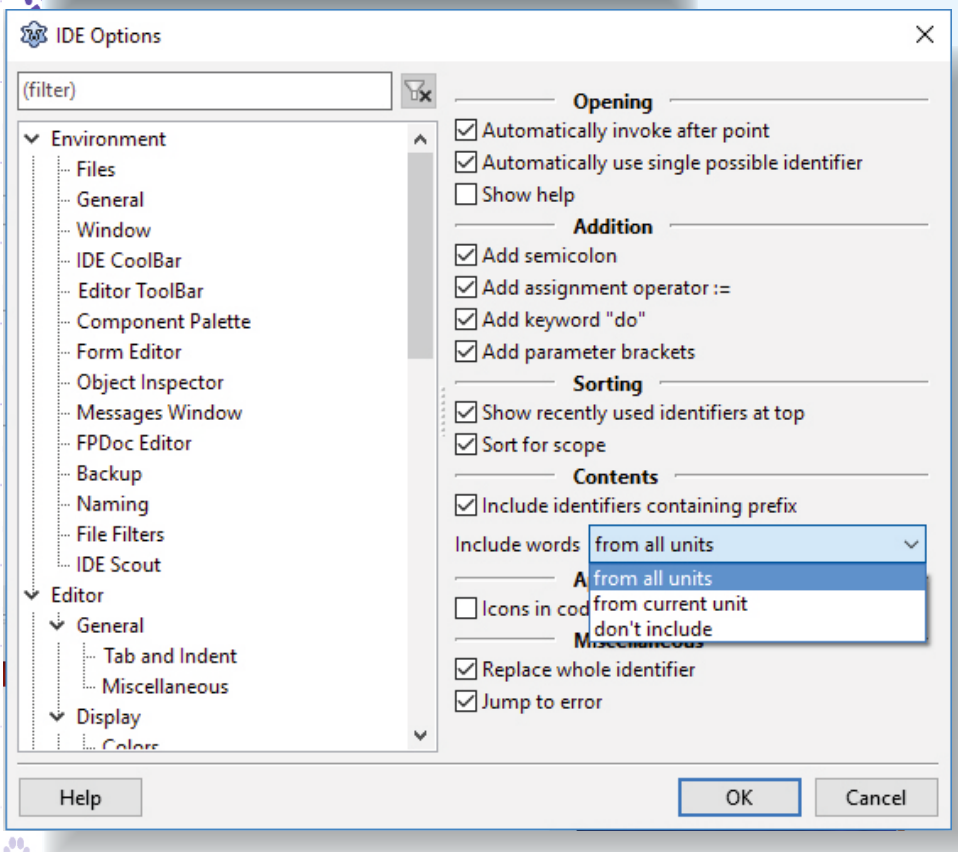
- Option: Show non visual components
  extended filter for the identifier completion window - include identifiers containing prefix.
  (Settings: IDE Options -> Codetools -> Identifier Completion -> Sorting -> Include identifiers containing prefix.)



IDE Changes

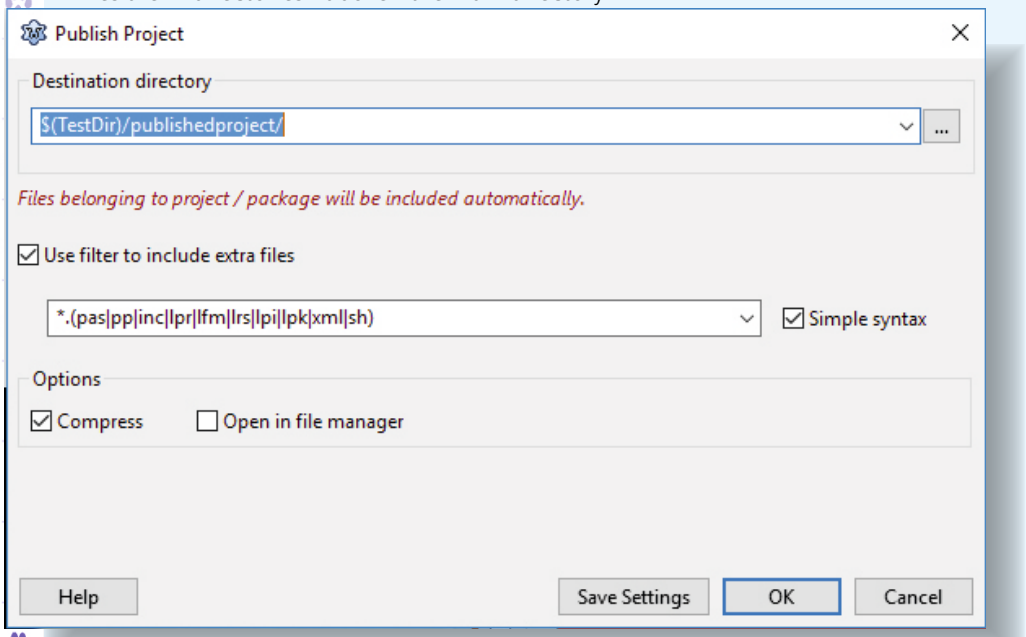
- the identifier completion window includes also text (words) from the currently opened units.
  It can be disabled or set-up in Tools -> IDE Options -> Codetools -> Identifier Completion -> Include words

```
  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private

  public

  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

word
type        Word
type        WordBool
type        WordRec = record
type        LongWord
type        QWord
type        QWordBool
```

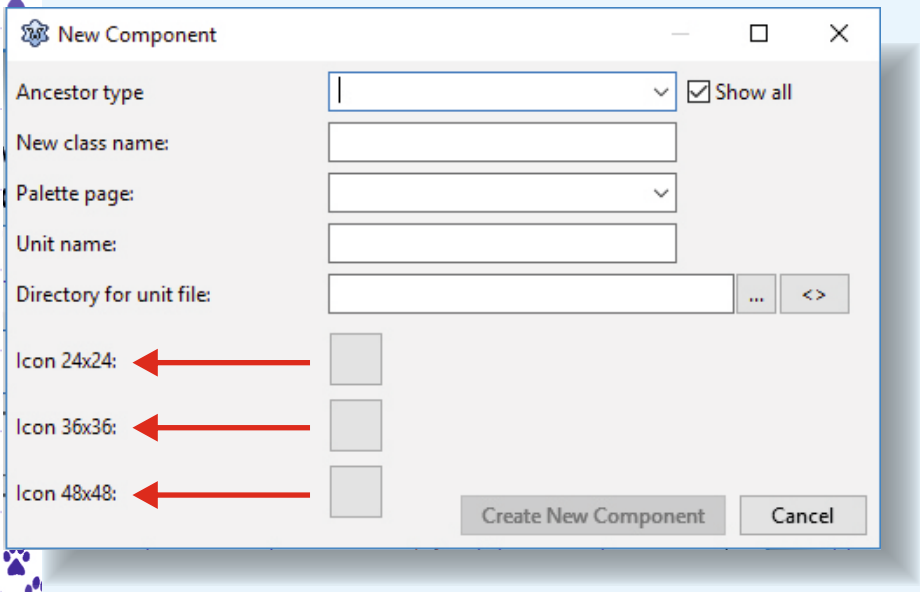


IDE Changes

- the **Publish Project/Package** feature and GUI window was revamped.
  Now all project/package member files are included automatically.
  More files can be added using a filter. The directory structure is maintained even if some files are in directories "above" the main directory.



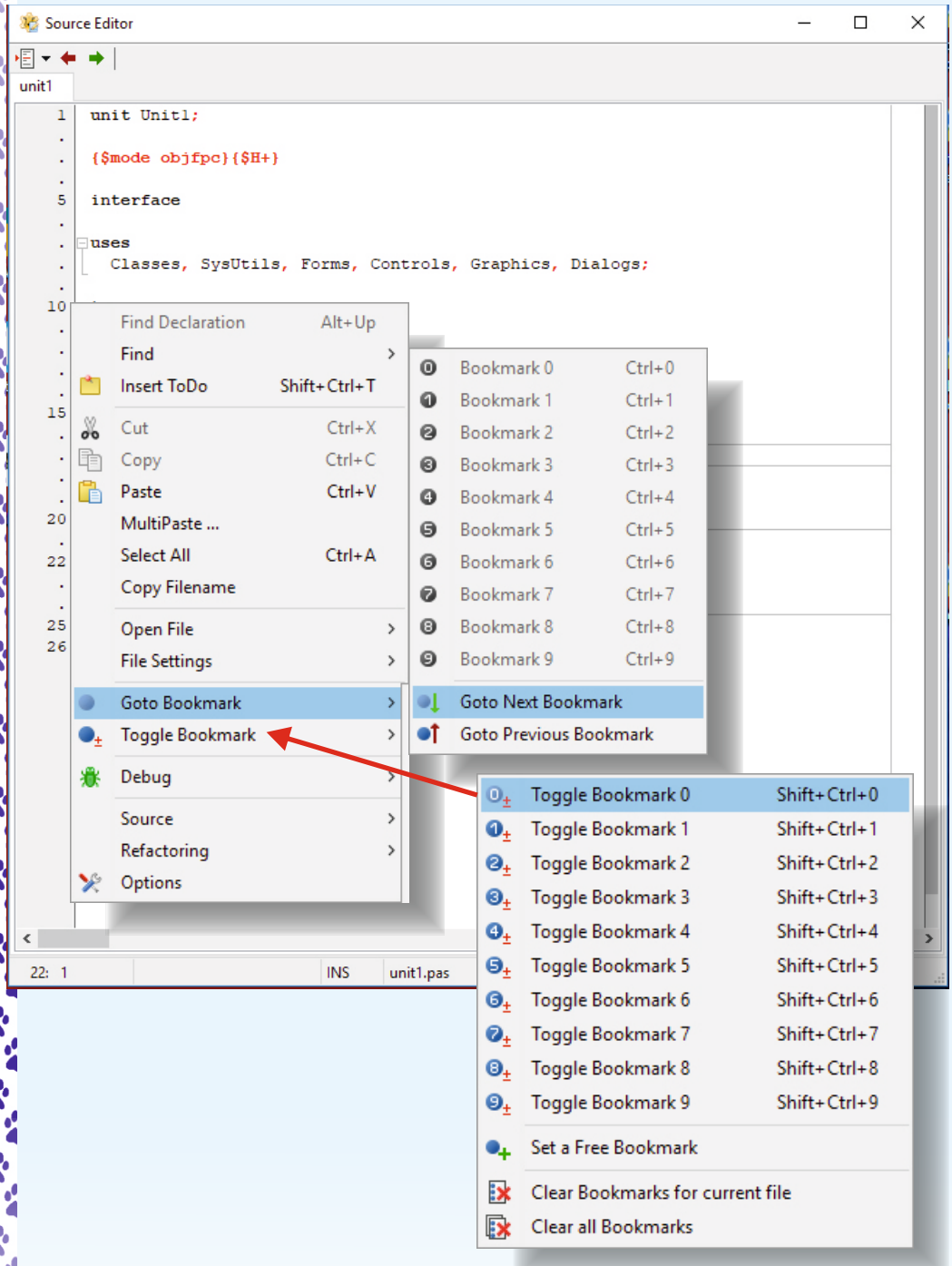
- the "New Component" window and user interface is improved.
  It also now supports 3 icons for different resolutions.



- the **Run -> Compile many Modes**... feature **now remembers** its previously selected modes.
- the **Search -> Procedure List**... window **now remembers** the state of its filter buttons.

Editor

- Added new **"smart move cursor"** moving commands to editor mappings.
  They introduce **word jumping positions** both at word starts and ends.
  They are **useful as alternative Ctrl+Left/+Right/+Shift+Left/+Shift+Right key**
- Added **Goto/Toggle bookmark submenu to toolbar**.
  Ctrl-b pops up "goto bookmark" list, with location for each bookmark.
  Ctrl-Shift-b pops up "toggle bookmark" list.



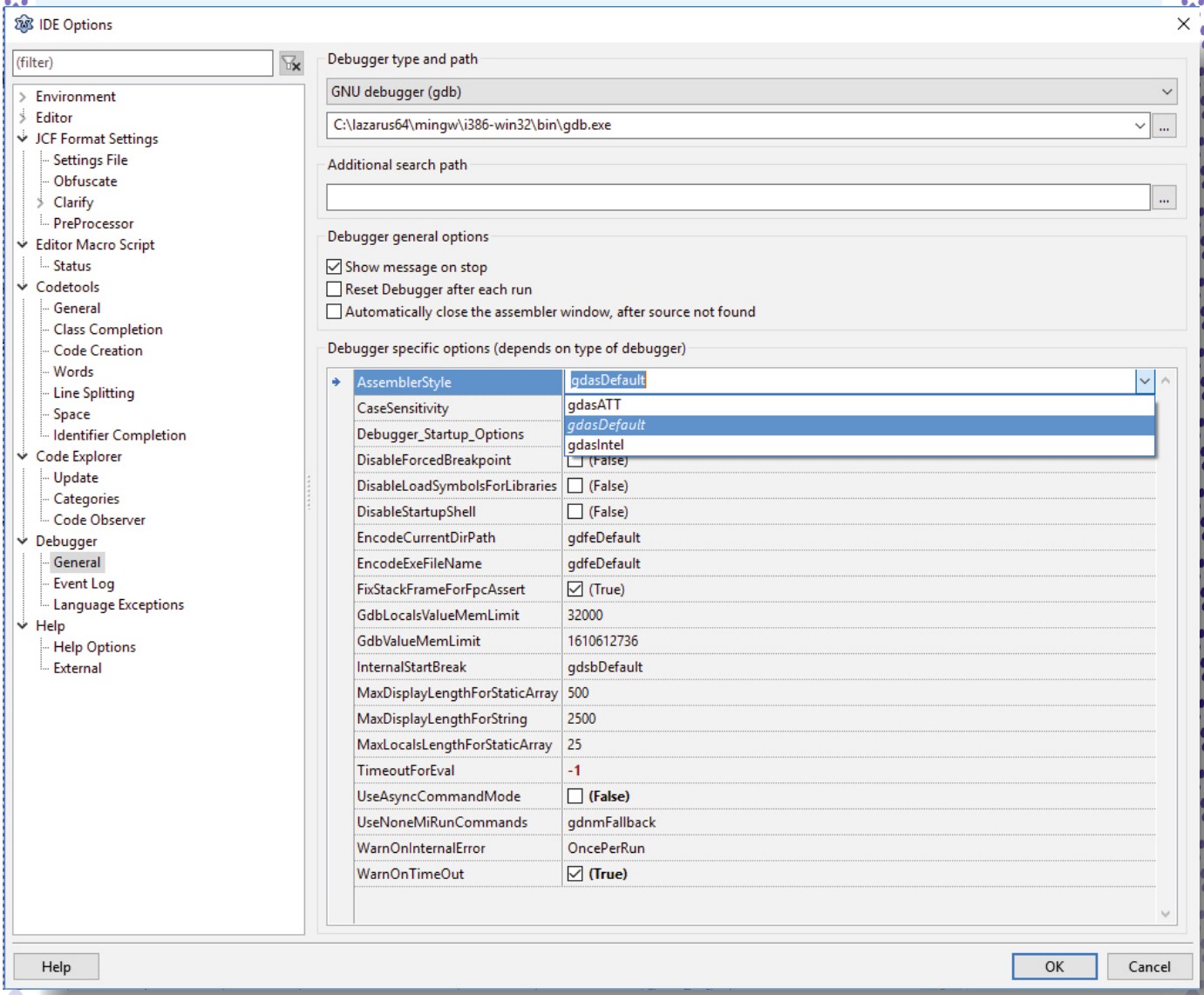
- **Sublime like handling of selection** with right/left navigation keys.
  Remove selection and keep caret at ex-boundary of selection.
  To activate check both: "Caret skips selection" and "Caret left/right clears selection (no move)".
  See mantis 26477. More options to fine tune outline colors

Debugger

Alpha: LLDB based debugger for MacOs (no code-signing required)
Alpha: LLDB + FpDebug based debugger for MacOs (no code-signing required)

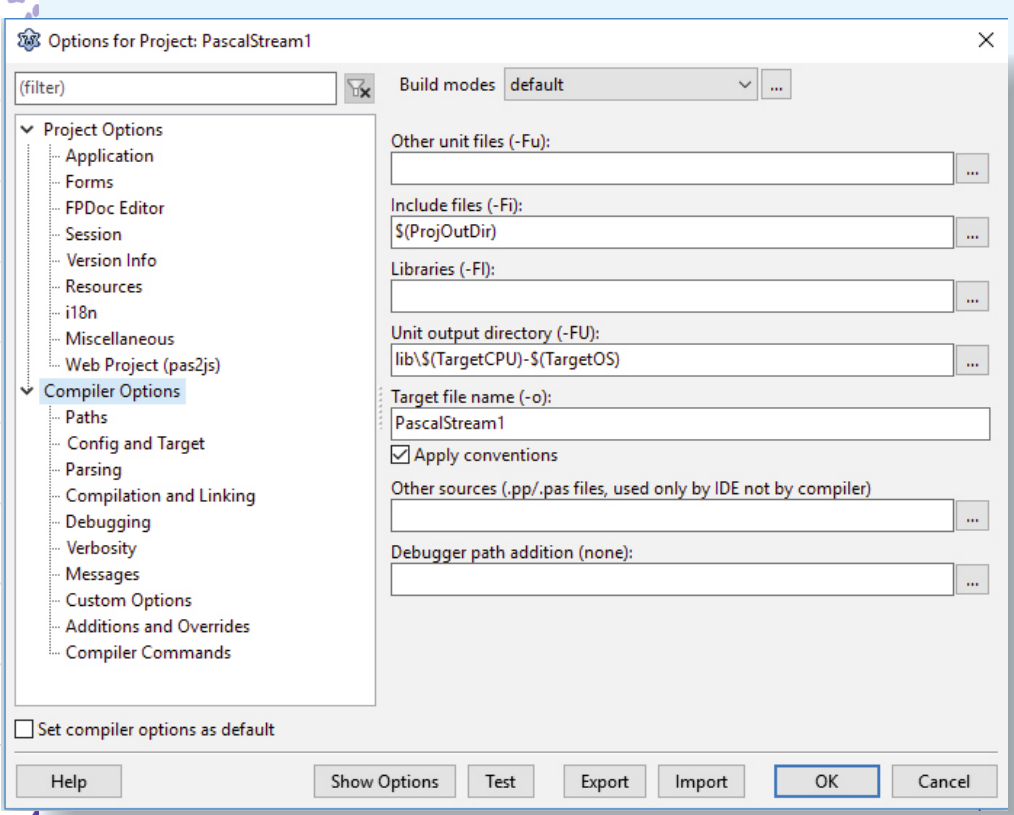
GDB Debugger: new options:

- Added option "**FixStackFrameForFpcAssert**" to workaround fpc wrong frame pointer (display correct line after assert failed)
- Added option "**AssemblerStyle**": ATT vs Intel
- Added option **"DisableStartupShell":** Required on MacOs.
- Added more **size limits for data evaluation** (avoid errors, timeouts and extremely slow responses)
  - **MaxDisplayLengthForStaticArray:** Similar to existing "MaxDisplayLengthForString". This should apply to static array, but it is up to gdb which types it applies it to.
  - **MaxLocalsLengthForStaticArray:** The same, but applied while getting values for the locals window, and the function parameters show in the stack window. Should be more restrictive.
- **GdbValueMemLimit:** For all types. Do not evaluate values, if gdb would need more memory. Setting this to big may crash gdb. Big values may also lead to slow response times (several minutes during which the IDE would be blocked)
- **GdbLocalsValueMemLimit:** For locals and stack.
- **DebugServer:** Added **"target-download"** if remote supports it. Auto closing of the asm window, if it was opened by breaking at a none source line.
- **Dragging selected identifier from source editor to watches window**, to create a watch



IDE Interfaces Changes

- Added FormEditingHook.SaveComponentAsPascal,
  which stores a designer form as Pascal statements using TCompWriterPas.
  You have various options to define the format and it tells you **what units are needed for the Pascal code**.
  There is an example adding a designer menu item to copy the Pascal statements to the clipboard:
  `examples/pascalstream/CopyAsPasPkg/copyaspasdemounit1.pas.`
- Added interface class **`TLazCompilationToolOptions`** with a Command and CompileReasons.
  Used in CompilerOptions.ExecuteBefore and .ExecuteAfter.



Components

TOpenGLControl

New property Options of type set, currently with ocoMacRetinaMode as the only member.
If set, **ocoMacRetinaMode** OpenGL controls will use **retina support (high resolution mode).**
Control can use Qt5 widgeset now, Qt4 is also fixed for modern OpenGL contexts.

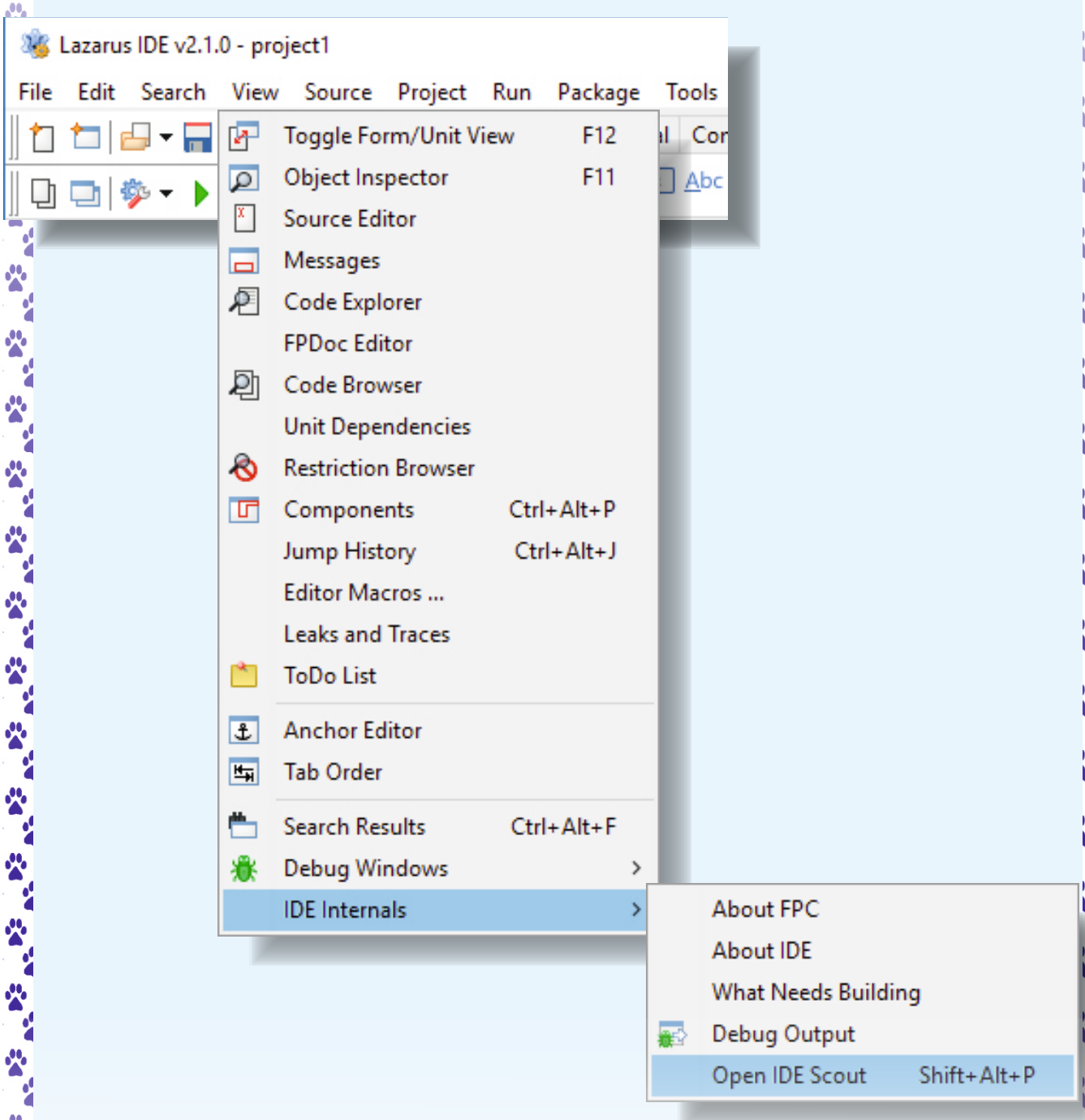
TAChart

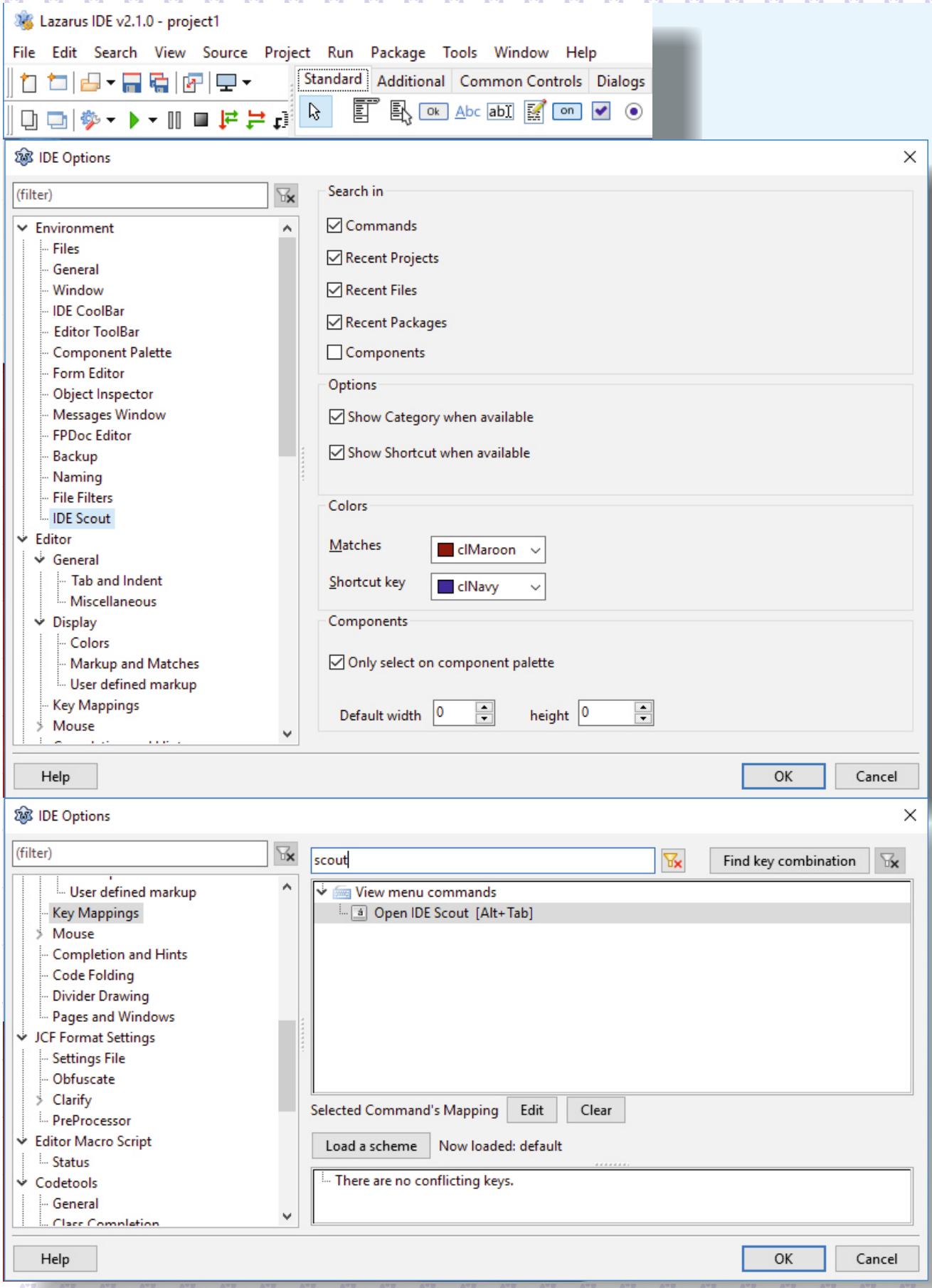
- The new **TExpressionSeries** and **TExpressionColorMapSeries** plot mathematical functions at design-time.
- **TLineSeries** has a new property **"ColorEach"** which can be used to color the line segments individually.
- **TCubicSplineSeries** has a new property **"SplineType"** which allows to select between the **"natural"** and the **"monotone Hermite"** splines (*the latter avoiding overshoot of the interpolation*).
- **TAreaSeris** has a new property **"Banded"** which suppresses painting of the bottom layer of stacked areas and makes the series look like a band or a stack of bands.
- The new **property "MarkDistancePercent"** of **TPieSeries** can be used, for example, to center a label within its pie for all chart sizes.
- **TLineSeries, TCubicSplineSeries, TBSplineSeries** and **TFitSeries** now can display error bars.
- The **TFitSeries** fitting engine was extended to be able to exclude specific parameters from fitting and to do a statistical analysis of the fit results (error estimates of the determined fit parameters, confidence limits, goodness of fit).

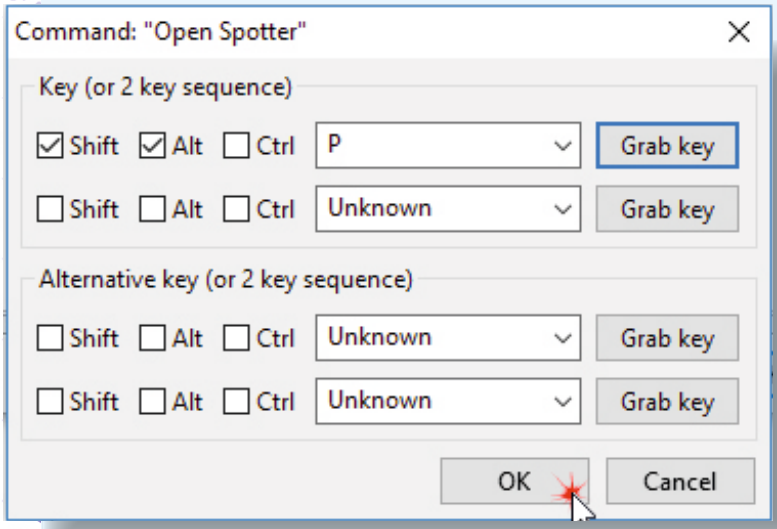
IDE SCOUT

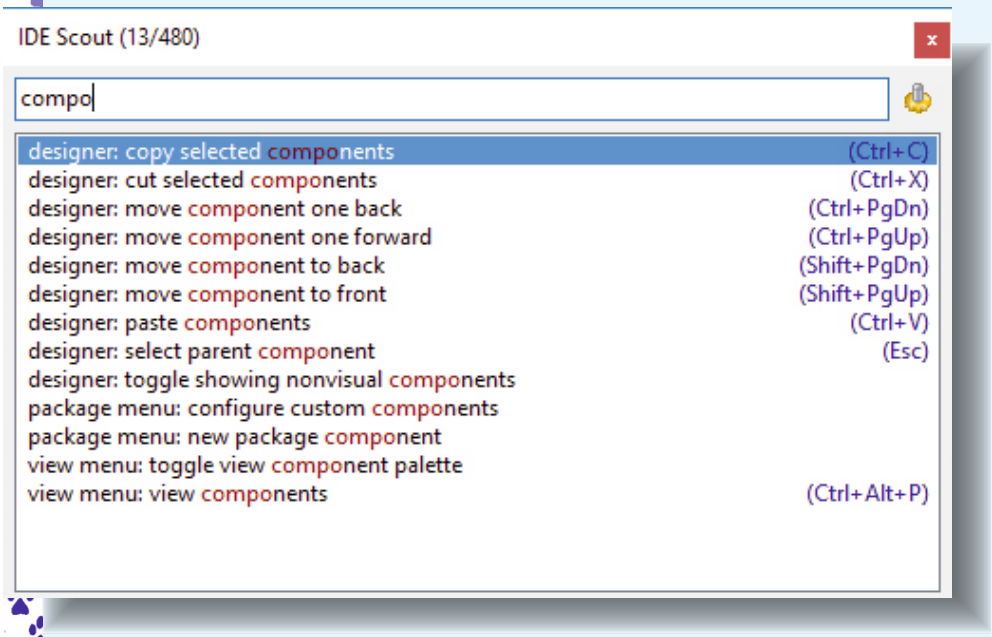
A very new item in the ide is the use of the **IDE Scout**. It' s easy to use and you can set it with your own key mapping. The project was created after an attendee (James Ralston) of the event in Bonn (Germany) asked for this. Michael van canneyt could not sleep and therefore he created it instantly. Mattias Gaertner implementd it during the live session.

IDE Scout is not in 2.0. It is in 2.1. So it is only available via svn/git.



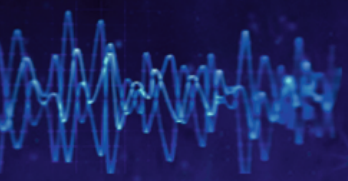

Lazarus IDE v2.1.0 - project1
File Edit Search View Source Project Run Package Tools Window Help
Standard Additional Common Controls Dialogs
IDE Options
(filter)
Environment
Files
General
Window
IDE CoolBar
Editor ToolBar
Component Palette
Form Editor
Object Inspector
Messages Window
FPDoc Editor
Backup
Naming
File Filters
IDE Scout
Editor
General
Tab and Indent
Miscellaneous
Display
Colors
Markup and Matches
User defined markup
Key Mappings
Mouse
Search in
Commands
Recent Projects
Recent Files
Recent Packages
Components
Options
Show Category when available
Show Shortcut when available
Colors
Matches
clMaroon
Shortcut key
clNavy
Components
Only select on component palette
Default width
0
height
0
Help
OK
Cancel
IDE Options
(filter)
User defined markup
Key Mappings
Mouse
Completion and Hints
Code Folding
Divider Drawing
Pages and Windows
JCF Format Settings
Settings File
Obfuscate
Clarify
PreProcessor
Editor Macro Script
Status
Codetools
General
scout
Find key combination
View menu commands
Open IDE Scout [Alt+Tab]
Selected Command's Mapping
Edit
Clear
Load a scheme
Now loaded: default
There are no conflicting keys.
Help
OK
Cancel


Command: "Open Spotter"
Key (or 2 key sequence)
Shift Alt Ctrl P Grab key
Shift Alt Ctrl Unknown Grab key
Alternative key (or 2 key sequence)
Shift Alt Ctrl Unknown Grab key
Shift Alt Ctrl Unknown Grab key
OK
Cancel


IDE Scout (13/480)
compo
designer: copy selected components (Ctrl+C)
designer: cut selected components (Ctrl+X)
designer: move component one back (Ctrl+PgDn)
designer: move component one forward (Ctrl+PgUp)
designer: move component to back (Shift+PgDn)
designer: move component to front (Shift+PgUp)
designer: paste components (Ctrl+V)
designer: select parent component (Esc)
designer: toggle showing nonvisual components
package menu: configure custom components
package menu: new package component
view menu: toggle view component palette
view menu: view components (Ctrl+Alt+P)

MITOV SOFTWARE

WWW.MITOV.COM



Signal Processing



Arduino



Audio

Computer Vision





Video


Communication
Animation
AI
Visual
Instruments
Delphi
Components
Galaxy
Process Control
Mouse Click Away

IMPLEMENT PICTURE IN PICTURE EFFECT AND HOW TO PERFORM VIDEO TRANSITIONS

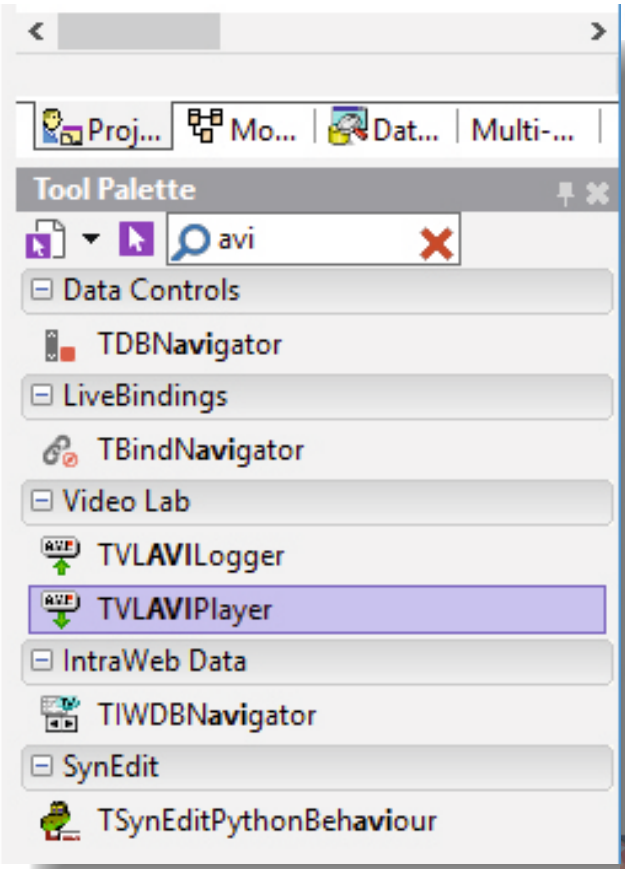
BY BOAIN MITOV

In the previous articles, I showed you how easy it is to play video files, capture from variety of video sources, how to filter and morph the video, apply effects, draw on frames, perform animations, render on variety of 2D and 3D surfaces, and finally how to record and broadcast the video.

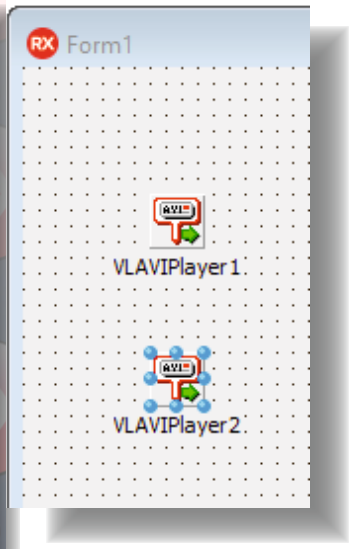
In this article I will show you how easy it is to mix, split and merge, videos, implement Picture in Picture effect, and how to perform video transitions.

In VideoLab there are 2 components that can be used to mix video. TVLCombine and TVLVideoMixer. The TVLCombine is designed specifically to allow 2 videos to be mixed with some advanced features suitable for this mode. The TVLVideoMixer is specifically designed to mix any number of video streams, but as such has less features when mixing only 2 videos.
We will start with a project using the TVLCombine to combine 2 videos.

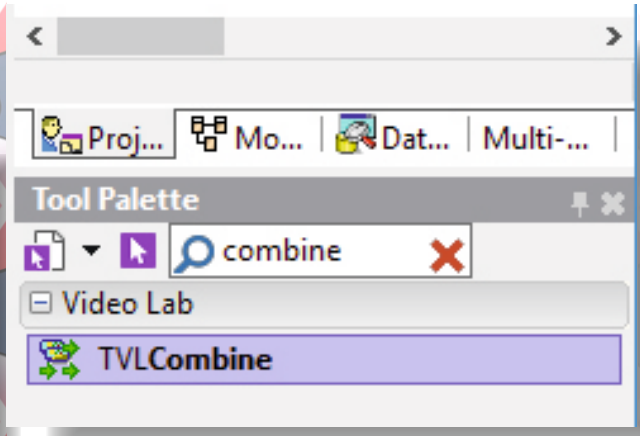
Start a new **VCL** Form application.
Type **"avi"** in the Tool Palette search box, then select **TVLAVIPlayer** component from the palette:



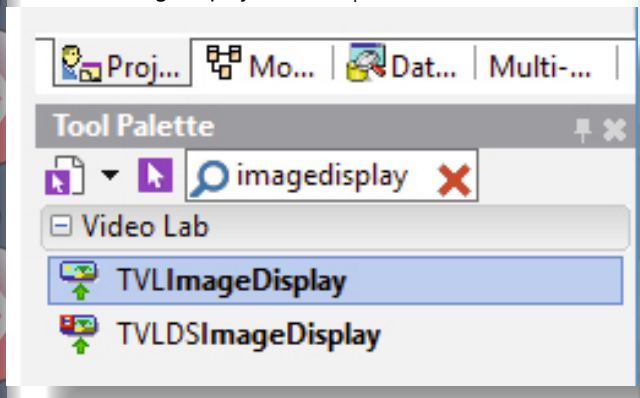
And drop 2 of them on the form.



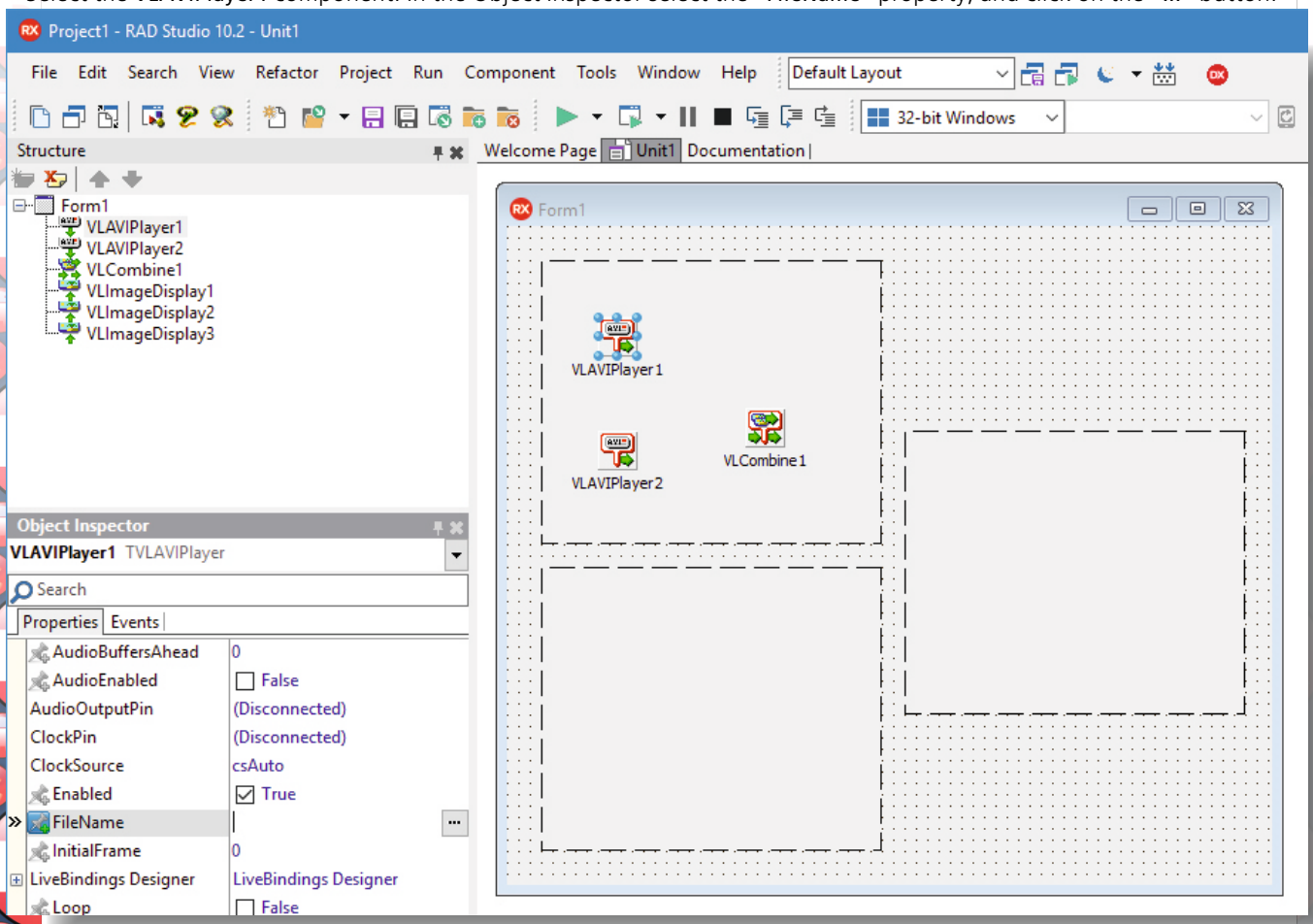
Type **"combine"** in the Tool Palette search box, then select **TVLCombine** component from the palette:



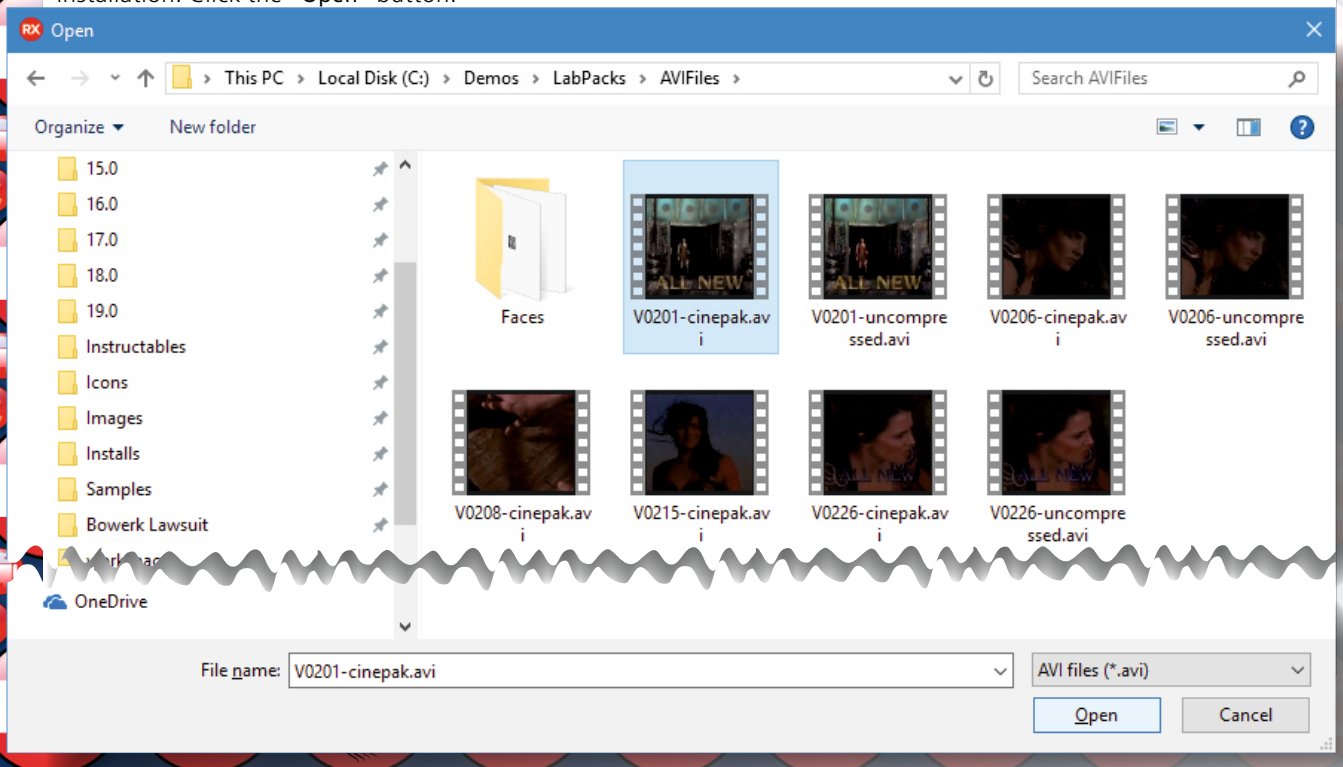
And drop it on the form.
Type **"imagedisplay"** in the Tool Palette search box, then select **TVLImageDisplay** from the palette:



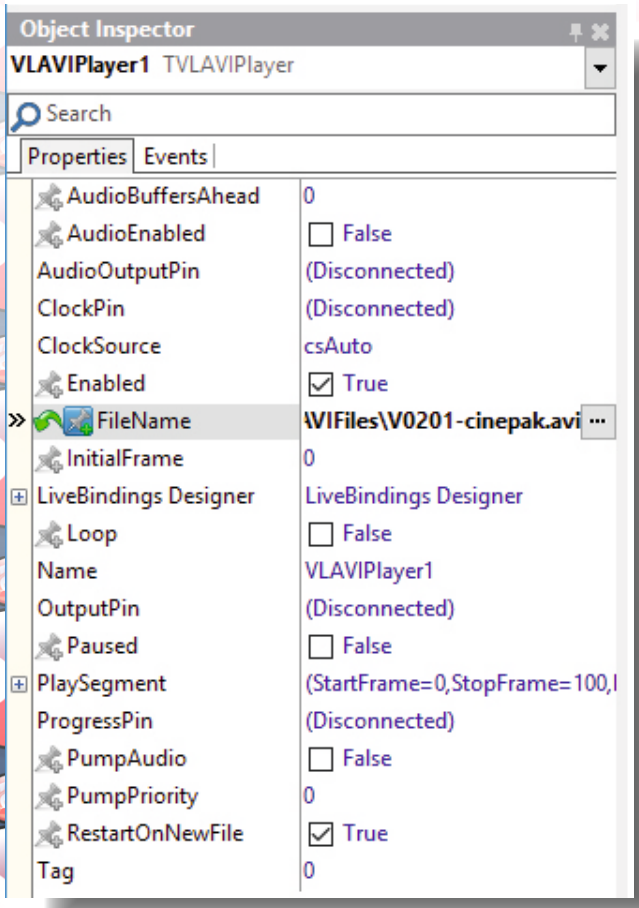
And drop 3 of them on the form.
Select the **VLAVIPlayer1** component. In the Object Inspector select the **"FileName"** property, and click on the **"..."** button:



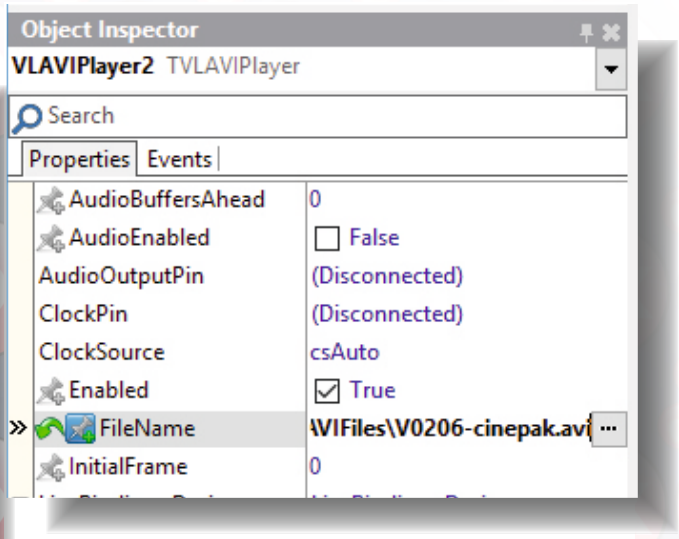
In the **File Open Dialog**, select a video file to play. The **AVI Player** can decode only limited number of video types, so to be sure that it will be able to decode the selected video, it is best to use the videos included in the **VideoLab** installation. Click the **"Open"** button:



This will assign the **FileName** property:



Select the **VLAVIPlayer2** component. In the **Object Inspector** set another file name in the **FileName** property:



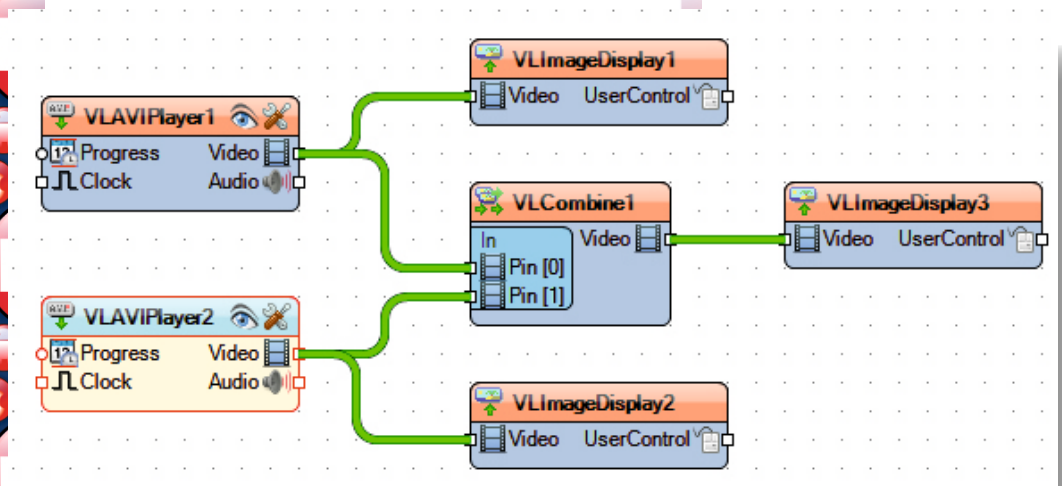
Switch to the **"Open Wire"** tab.
Connect the **"Video"** output pin of the **VLAVIPlayer1** component to the **"Video"** input pin of the **VLImageDisplay1** component.

Connect the **"Video"** output pin of the **VLAVIPlayer1** component to the **"Pin[0]"** input pin of the **"In"** pin list of the **VLCombine1** component.

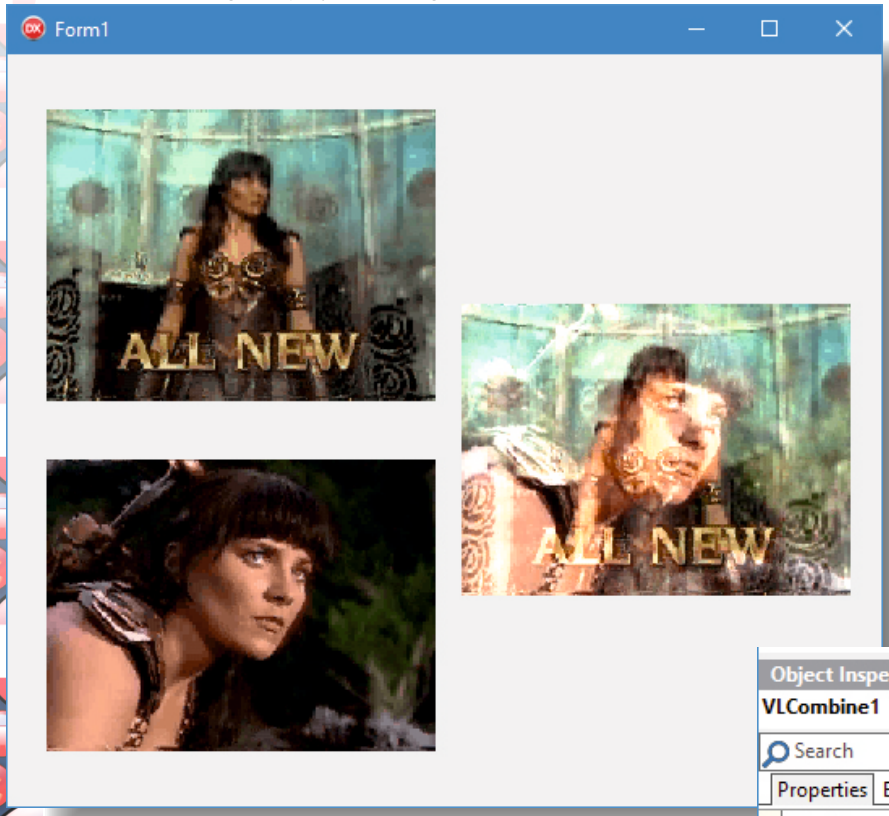
Connect the **"Video"** output pin of the **VLAVIPlayer2** component to the **"Video"** input pin of the **VLImageDisplay2** component.

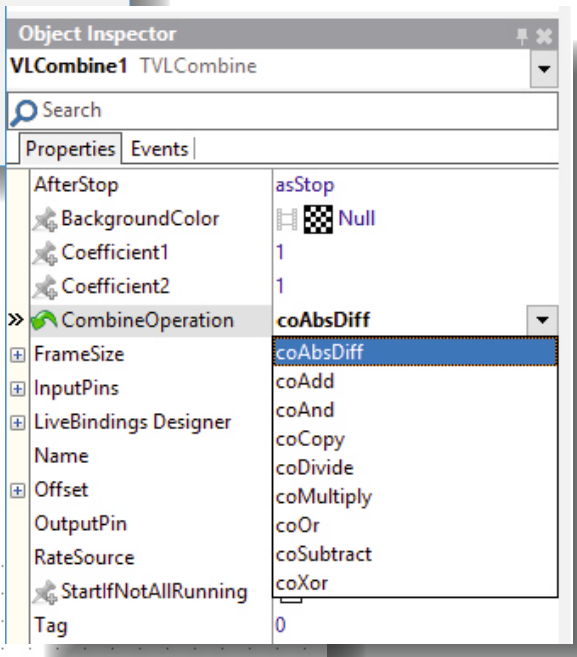
Connect the **"Video"** output pin of the **VLAVIPlayer2** component to the **"Pin[1]"** input pin of the **"In"** pin list of the **VLCombine1** component.

Connect the **"Video"** output pin of the **VLCombine1** component to the **"Video"** input pin of the **VLImageDisplay3** component as shown on the picture:

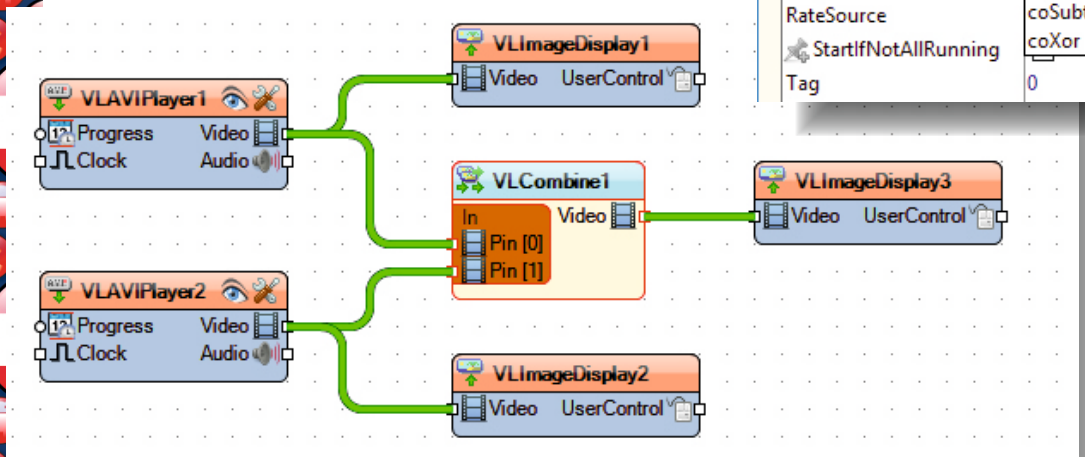


Compile and run the application. You will see the 2 videos playing in the image displays on the left, and the mixed video in the image display on the right:

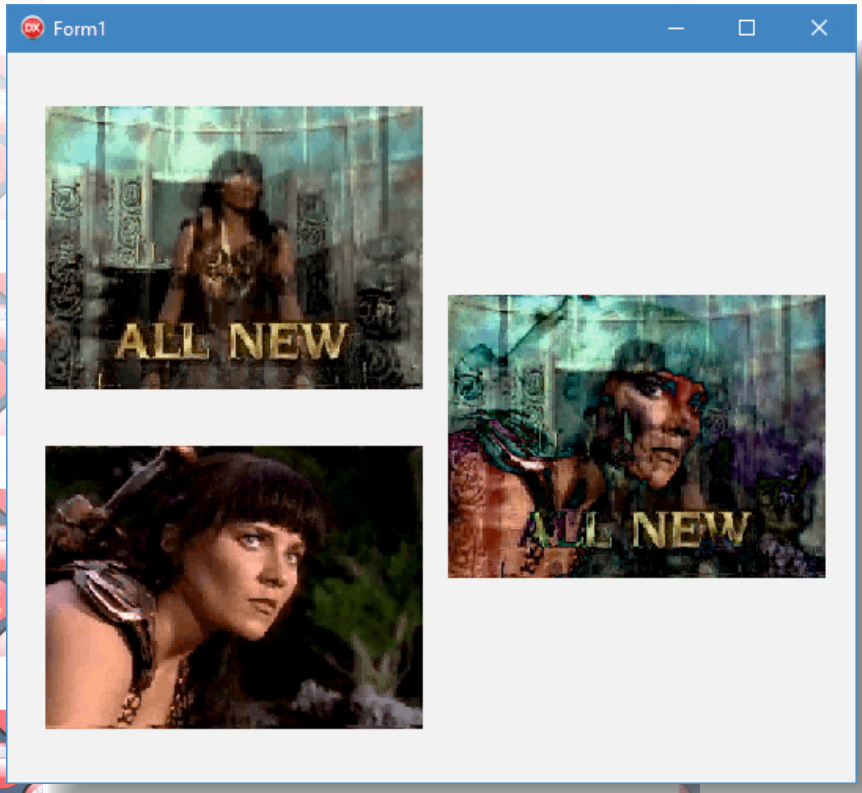




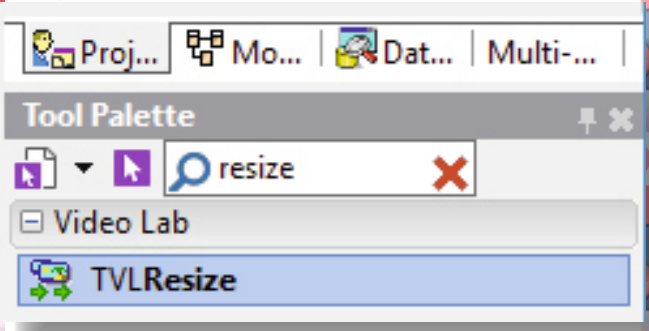
Close the application.
The **TVLCombine** component has many mixing modes. We can easily change it to combine the videos differently. Select the **VLCombine1** component.
In the **Object Inspector** set the **"CombineOperation"** property to **"coAbsDiff"**:



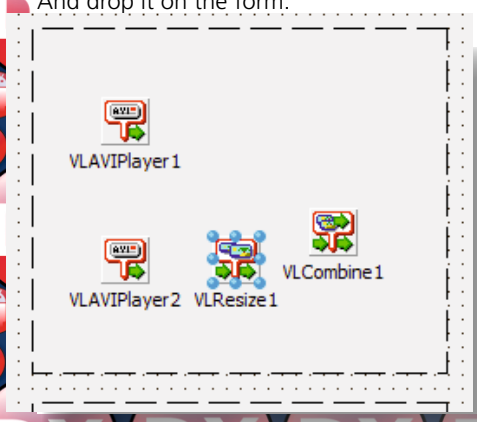
Compile and run the application. You will see the 2 videos combined using absolute difference between the pixels:



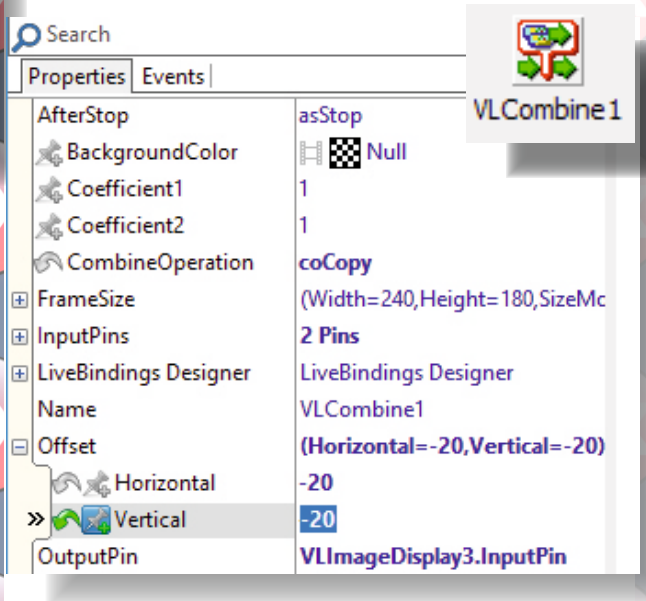
The **TVLCombine** can be used to implement **"picture in picture"** effect. For this we will resize one of the videos and we will offset it to some position. Type "resize" in the Tool Palette search box, then select TVLResize from the palette:

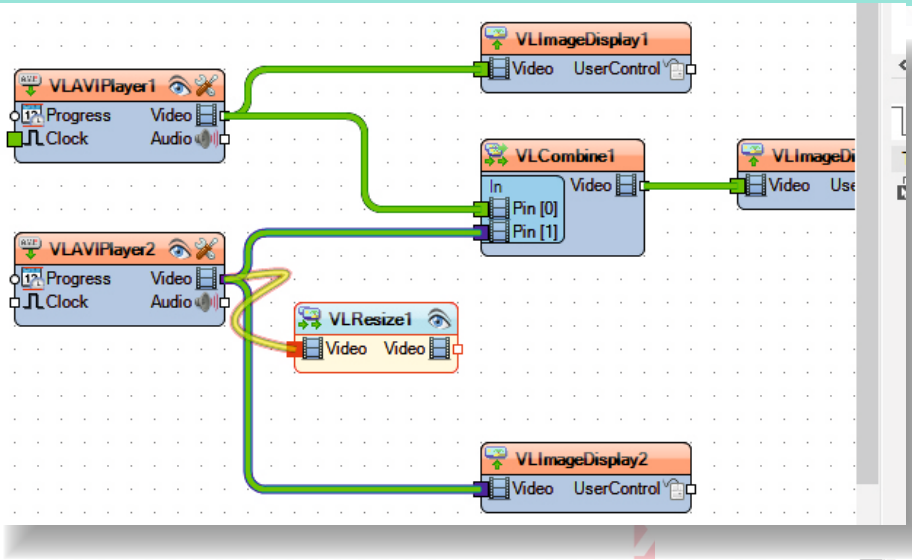


And drop it on the form.

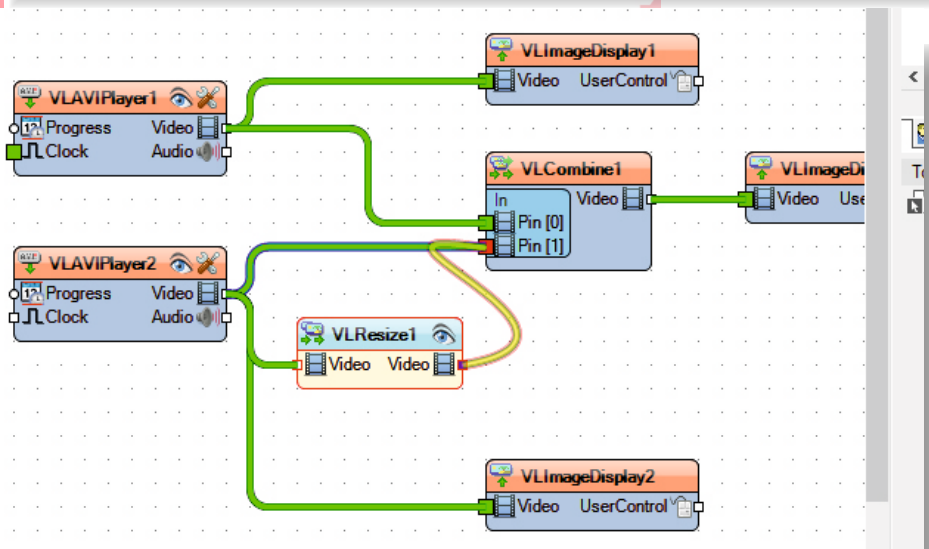


The component will resize one of the videos when we connect it later in the OpenWire tab.
Now we need to set some X/Y offset for the video when we combine.
Select the VLCombine1 component.
In the Object Inspector, set the value of the "CombineOperation" property to "coCopy". This will copy the second image on top of the first one.
In the Object Inspector, expand the "Offset" property, and set the "Horizontal" sub-property to "-20", and the "Vertical" sub-property to "-20":

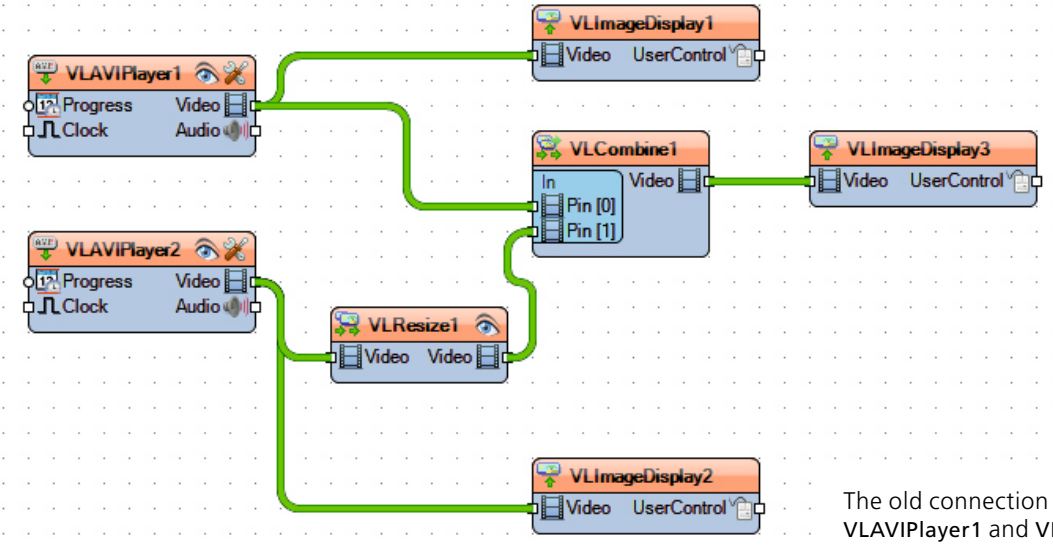




Switch to the **"Open Wire"** tab. Connect the **"Video"** Output Pin of the **VLAVIPlayer2** to the **"Video"** Input Pin of the **VLResize1** component:

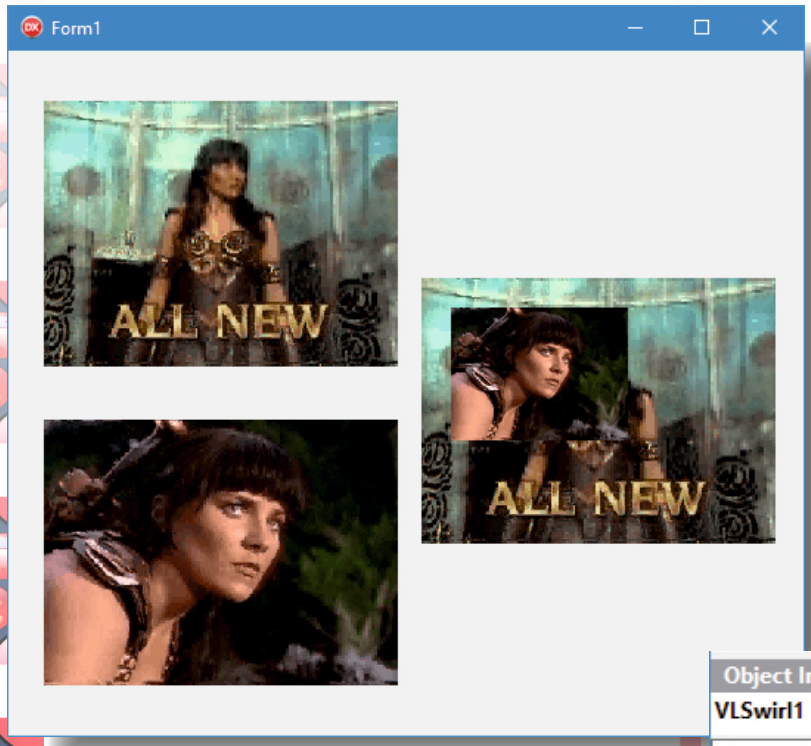


Connect the **"Video"** Output Pin of the **VLResize1** to the **"Pin[1]"** input pin of the "In" pin list of the **VLCombine1** component:



The old connection between **VLAVIPlayer1** and **VLCombine1** will be automatically disconnected. Here is the complete **OpenWire** diagram for the project:

Compile and run the application. You will see the 2 videos combined with **"Picture in Picture"** effect:



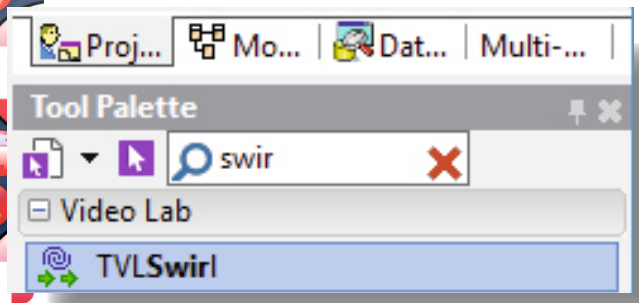
Close the application.

The **TVLCombine** merges the videos using alpha transparency channel, which means that each pixel can have a channel indicating how transparent it is.

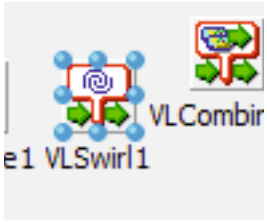
All geometric transformations in **VideoLab** properly set the alpha channel for the pixels, so geometrically transformed videos can be mixed with transparent pixels outside of the transformation area.

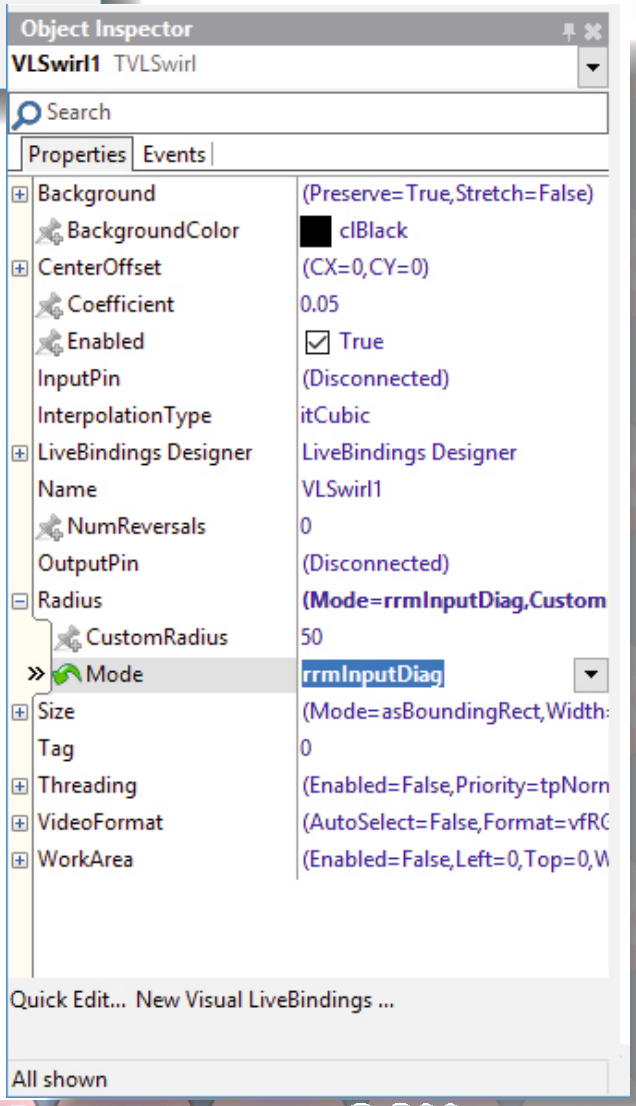
To demonstrate this we will swirl one of the videos, and will copy it on top of the other with alpha transparency.

Type "swir" in the Tool Palette search box, then select TVLSwirl from the palette:

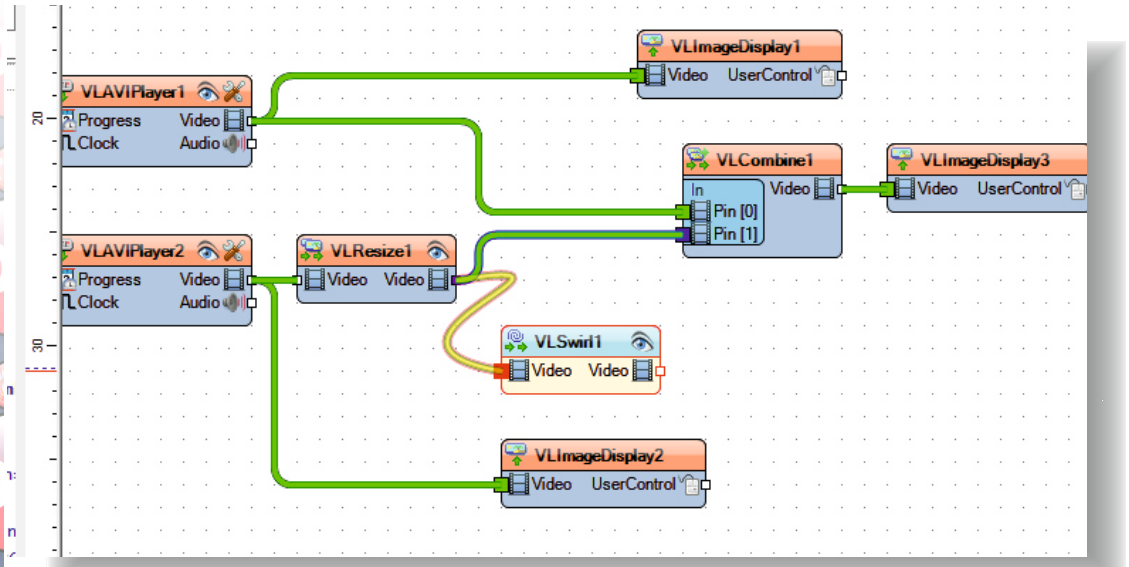


And drop it on the form.
In the Object Inspector, expand the "Radius" property, and set the "Mode" sub-property to "rrmInputDiag":

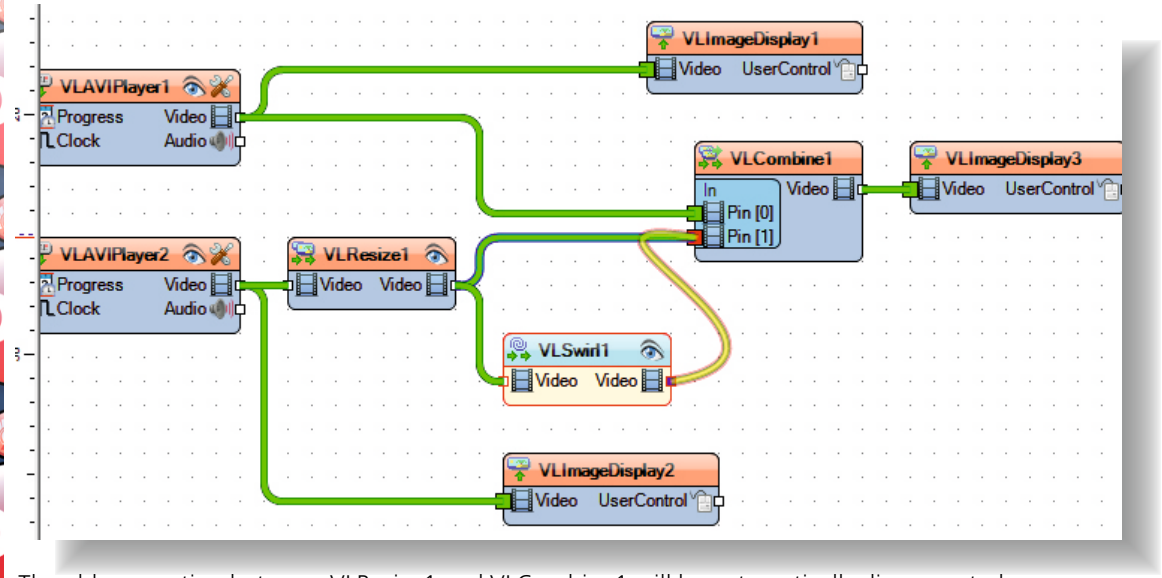




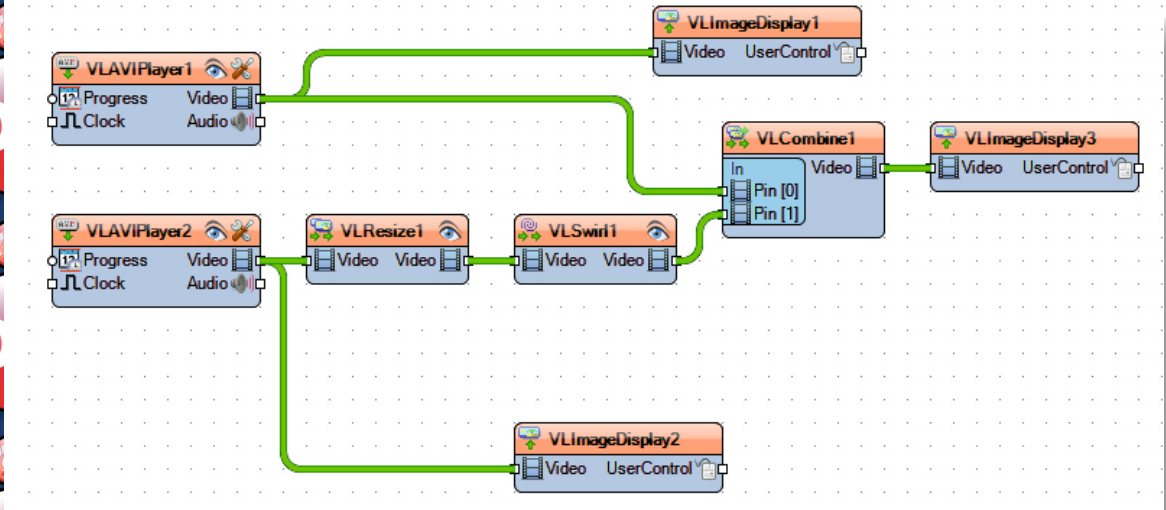
Connect the "Video" Output Pin of the VLResize1 to the "Video" Input Pin of the VLSwirl1 component:



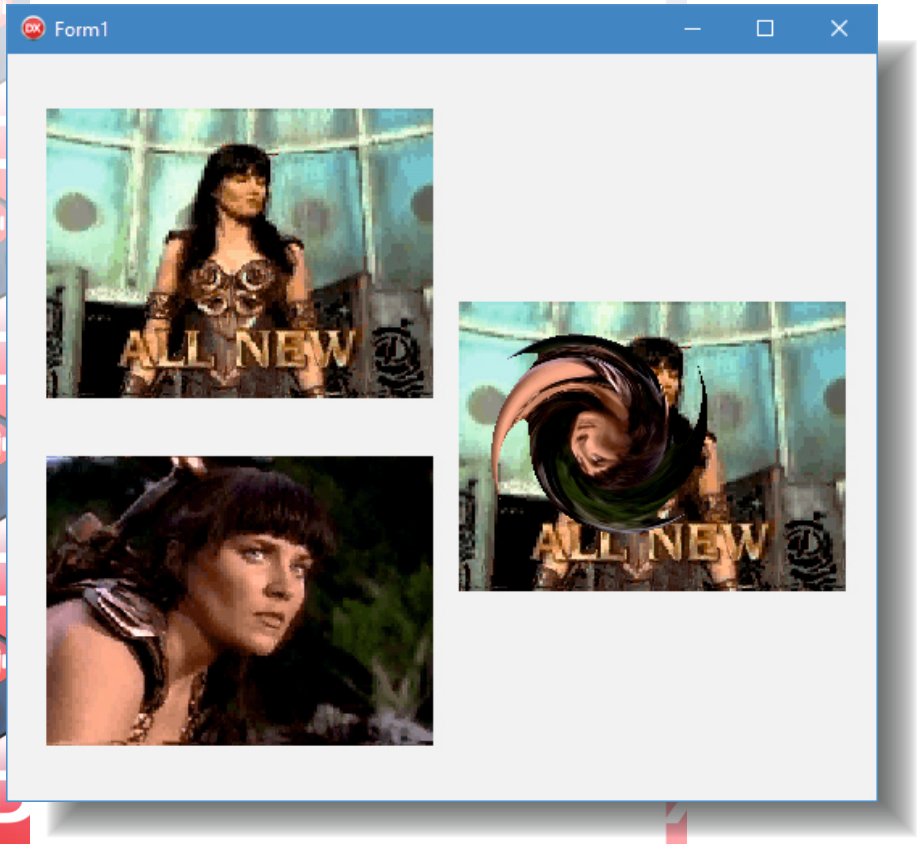
Connect the "Video" Output Pin of the VLSwirl1 to the "Pin[1]" input pin of the "In" pin list of the VLCombine1 component:



The old connection between VLResize1 and VLCombine1 will be automatically disconnected.
Here is the complete OpenWire diagram for the project:



Compile and run the application. You will see the 2 videos combined with the second video swirled and properly applied on top of the first with the areas outside the swirl transparent:

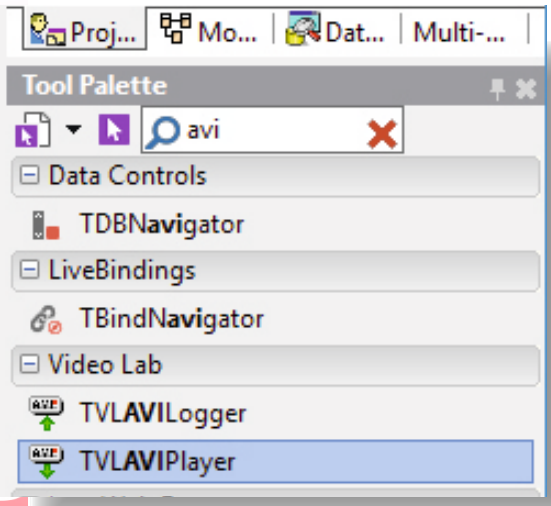


Close the application.

There are many more effects and mixing modes that can be achieved with the **TVLCombine.** Some of them are shown in the demo projects included in the VideoLab installations. You can also experiment on your own and discover the rest.
When you need to mix more than 2 video streams, you can use the **TVLVideoMixer** component. Now I will show you how you can use the **TVLVideoMixer** to mix together 3 videos.

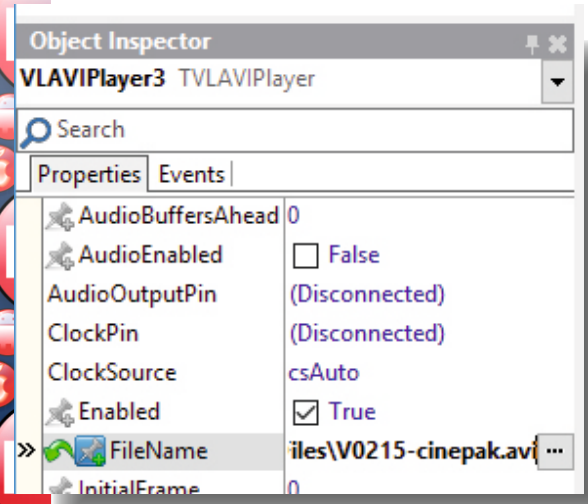
Start a new VCL Form application.

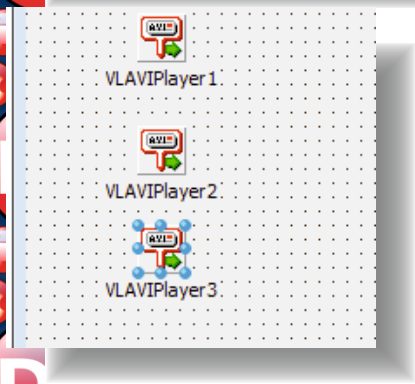
Type "avi" in the Tool Palette search box, then select **TVLAVIPlayer** component from the palette:



And drop 3 of them on the form.

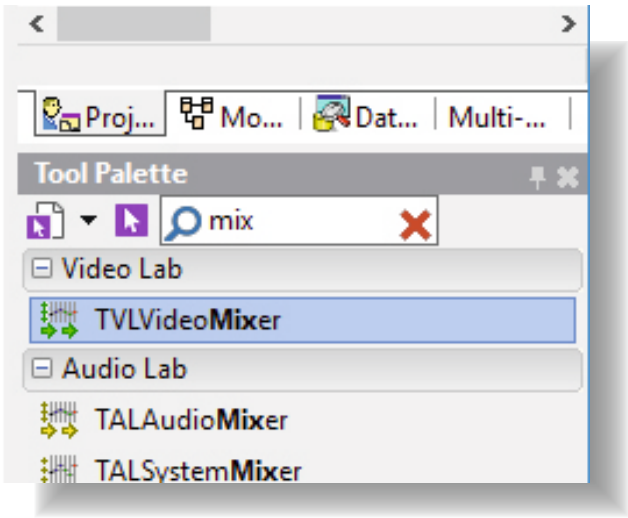
In the **Object Inspector,** set file names in the **"FileName"** properties for the 3 components. Use different video files for each:



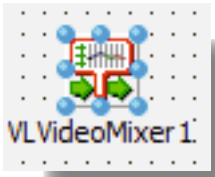


The **AVI Player** can decode only limited number of video types, so to be sure that it will be able to decode the selected video, it is best to use the videos included in the **VideoLab** installation.

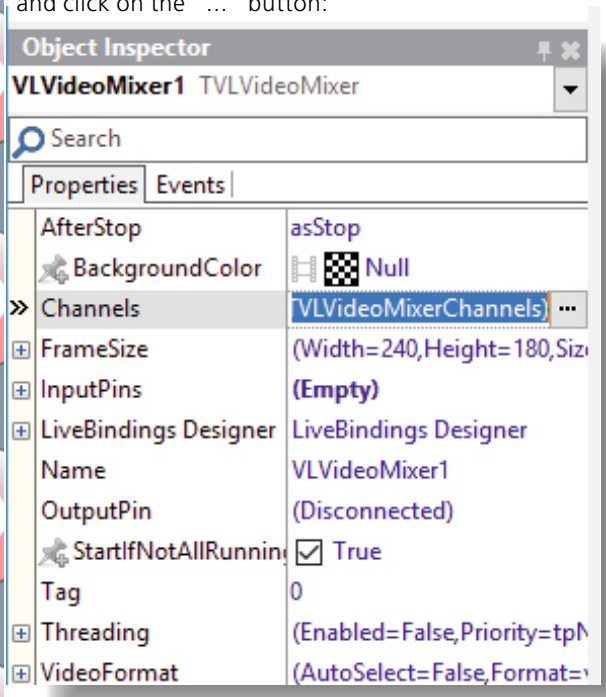
Type "mix" in the Tool Palette search box, then select TVLVideoMixer component from the palette:



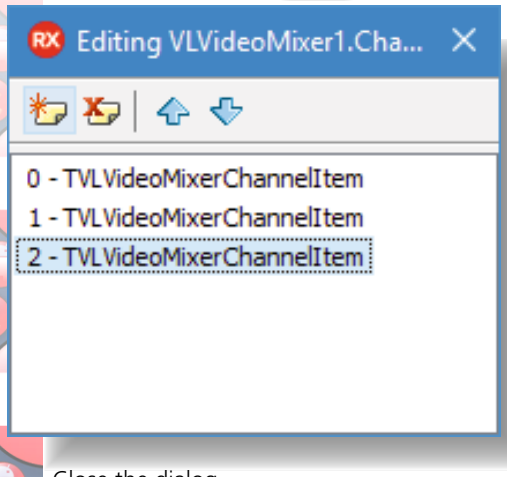
And drop it on the form.



In the Object Inspector select the "Channels" property, and click on the "..." button:

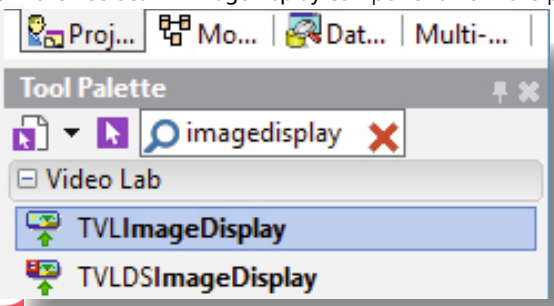


This will open the Channels collection editing dialog.
Click 3 times on the button to add 3 channels:

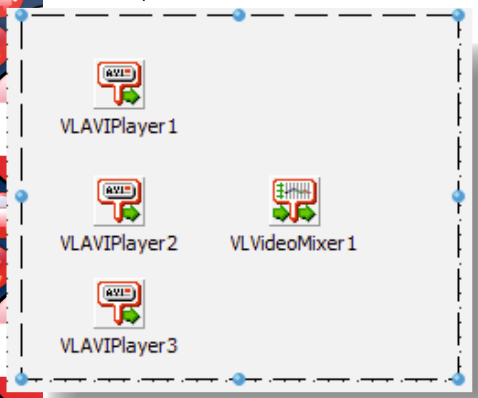


Close the dialog.

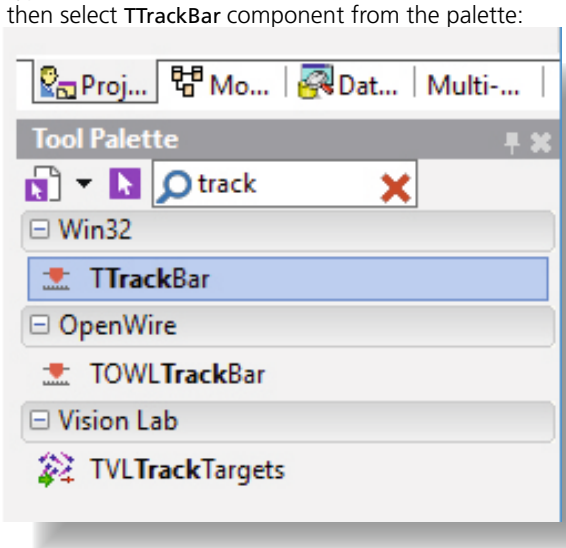
Type **"imagedisplay"** in the Tool Palette search box,
then select **TVLImageDisplay** component from the palette:



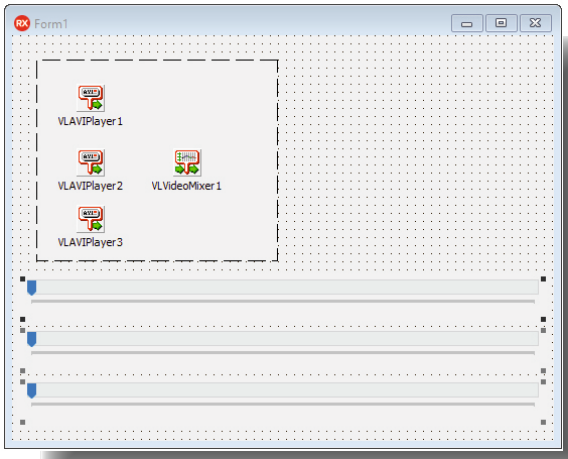
And drop it on the form.



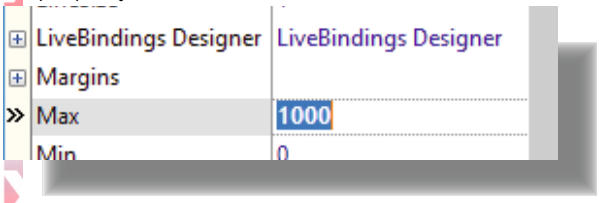
Type **"track"** in the **Tool Palette** search box,
then select **TTrackBar** component from the palette:



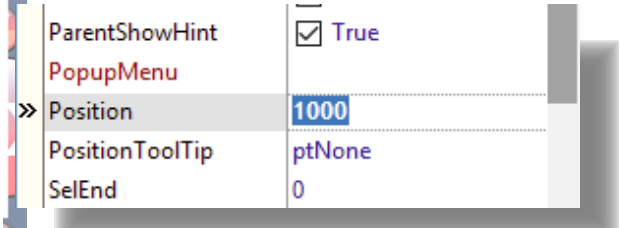
And drop 3 of them on the form.
Select the 3 TTrackBar components on the form.



In the **Object Inspector** set the value of the **"Max"**
property for the 3 track bars to **"1000"**:



In the Object Inspector set the value of the
"Position" property for the 3 track bars to "1000":

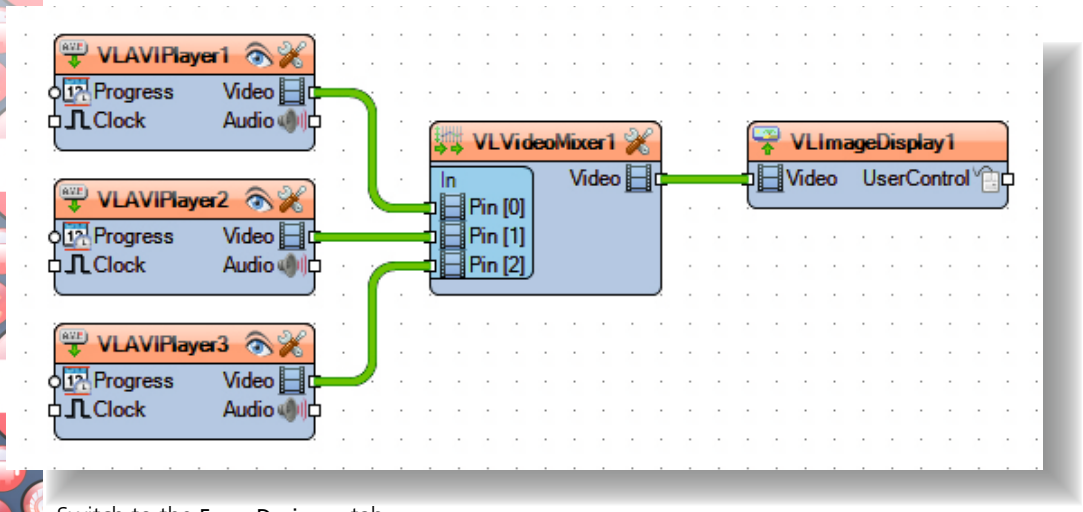


Switch to the **"Open Wire"** tab.
Connect the **"Video"** output pin of the **VLAVIPlayer1** component to the "Pin[0]" input pin of the **"In"** pin list of the **VLVideoMixer1** component.

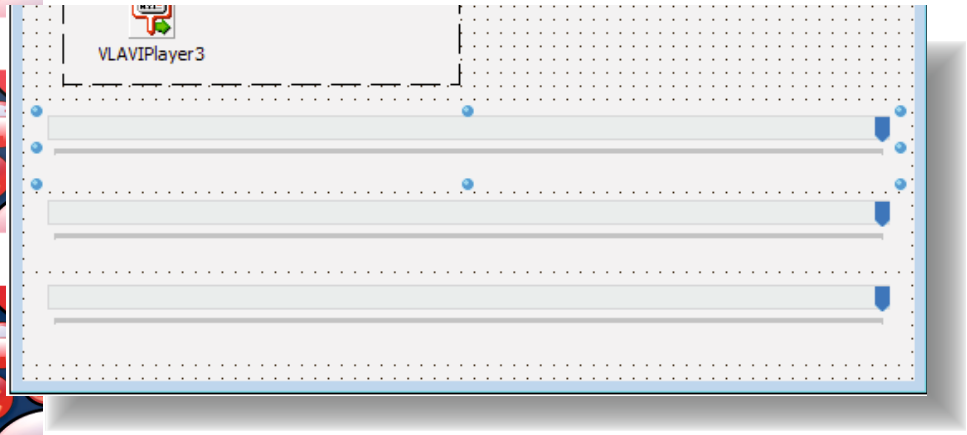
Connect the **"Video"** output pin of the **VLAVIPlayer2** component to the **"Pin[1]"** input pin of the **"In"** pin list of the **VLVideoMixer1** component.

Connect the **"Video"** output pin of the **VLAVIPlayer3** component to the **"Pin[2]"** input pin of the **"In"** pin list of the **VLVideoMixer1** component.

Connect the **"Video"** output pin of the **VLVideoMixer1** component to the **"Video"** input pin of the **VLImageDisplay1** component as shown on the picture:



Switch to the **Form Designer** tab.
Double-click on the **TrackBar1** component to generate **OnChange** event handler:



In the event handler write the flowing code:

```
procedure TForm1.TrackBar1Change(Sender:TObject);
begin
 VLVideoMixer1.Channels[0].Coefficient:=TrackBar1.Position/1000;
end;
```

Here we will set the **"Coefficient"** of one of the channels between 0.0 and 1.0 indicating how much this video will appear in the final mix. 0.0 means the video will be invisible (will not appear), 1.0 means it will fully appear (Will not be transparent). Values between 0.0 and 1.0 make the video more or less transparent.

Switch to the **Form Designer** tab. Double-click on the **TrackBar2** component to generate **OnChange** event handler:

In the even handler write the flowing code:

```
procedure TForm1.TrackBar2Change(Sender:TObject);
begin
 VLVideoMixer1.Channels[1].Coefficient:=TrackBar2.Position/1000;
end;
```

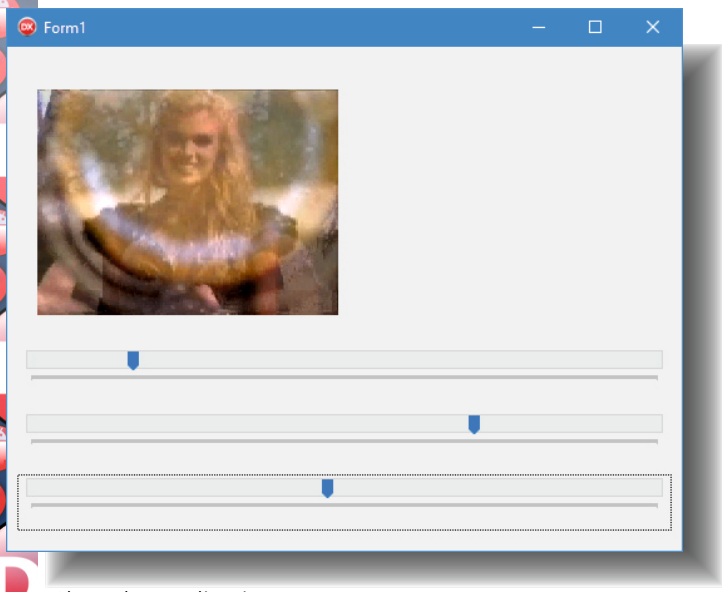
Switch to the Form Designer tab. Double-click on the TrackBar3 component to generate OnChange event handler.

In the even handler write the flowing code:

```
procedure TForm1.TrackBar3Change(Sender:TObject);
begin
 VLVideoMixer1.Channels[2].Coefficient:=TrackBar3.Position/1000;
end;
```

Now the 3 Track Bars will control the mixing level of the 3 channels of the video mixer.

Compile and run the application. You will see the videos mixed, and you can use the track bars to control the mixing of the 3 video channels:



Close the application.

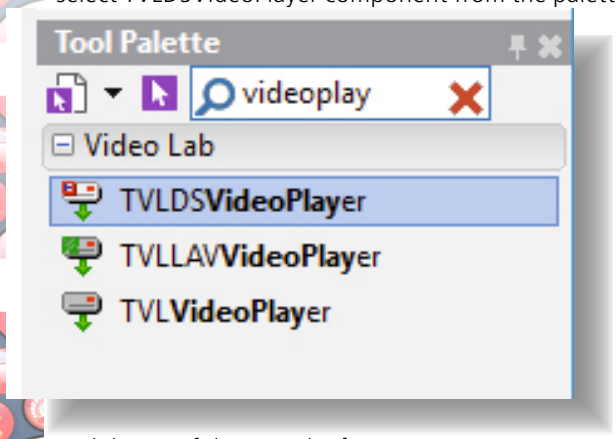
Here is the complete source code for this project:

```
unit Unit1;

interface

uses
 Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
 Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, VLMultiInput, VLVideoMixer,
 Mitov.Types, VLBasicVideoPlayer, VLAVIPlayer, Vcl.ComCtrls, Mitov.VCLTypes,
 VCL.LPControl, SLControlCollection, VLCommonDisplay, VLImageDisplay;

type
 TForm1 = class(TForm)
  VLAVIPlayer1: TVLAVIPlayer;
  VLAVIPlayer2: TVLAVIPlayer;
  VLAVIPlayer3: TVLAVIPlayer;
  VLVideoMixer1: TVLVideoMixer;
  VLImageDisplay1: TVLImageDisplay;
  TrackBar1: TTrackBar;
  TrackBar2: TTrackBar;
  TrackBar3: TTrackBar;
  procedure TrackBar1Change(Sender: TObject);
  procedure TrackBar2Change(Sender: TObject);
  procedure TrackBar3Change(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.TrackBar1Change(Sender: TObject);
begin
 VLVideoMixer1.Channels[0].Coefficient := TrackBar1.Position/1000;
end;

procedure TForm1.TrackBar2Change(Sender: TObject);
begin
 VLVideoMixer1.Channels[1].Coefficient := TrackBar2.Position/1000;
end;

procedure TForm1.TrackBar3Change(Sender: TObject);
begin
 VLVideoMixer1.Channels[2].Coefficient := TrackBar3.Position/1000;
end;

end.
```

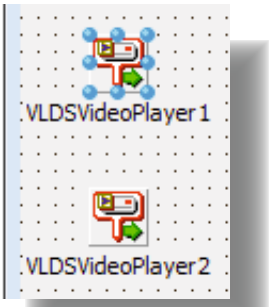
In addition to simply mixing video channels, the TVLCombine and TVLVideoMixer can be used to implement smooth video transitions from video to video.

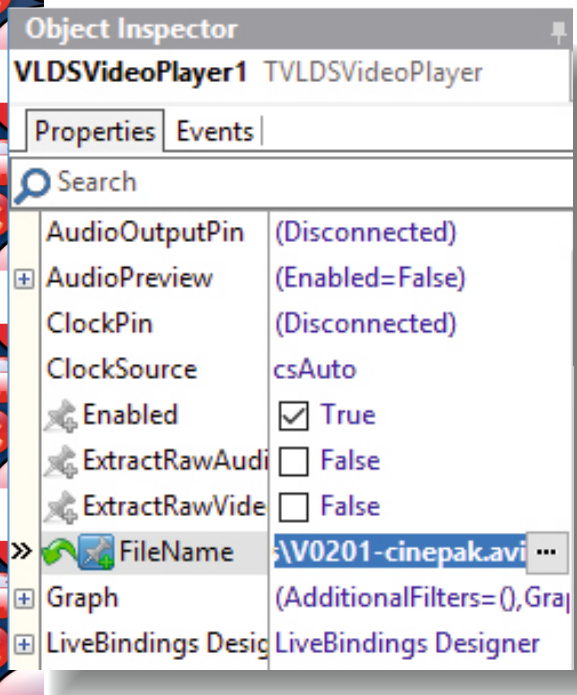
Start a new VCL Form application.

Type "videoplayer" in the Tool Palette search box, then select TVLDSVideoPlayer component from the palette:

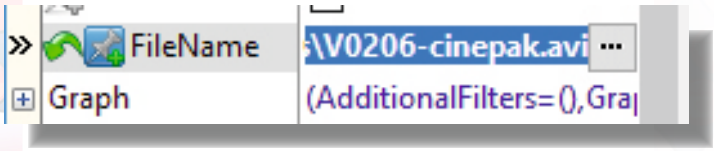


And drop 2 of them on the form.
Select the **VLDSVideoPlayer1** component and in the **Object Inspector** set a file name for the **"FileName"** property:

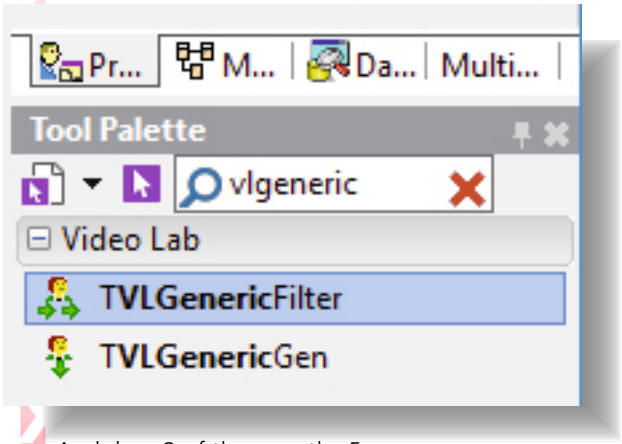




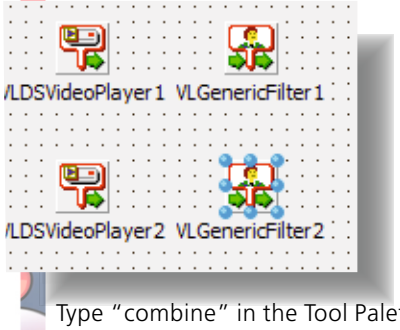
Select the VLDSVideoPlayer2 component and in the Object Inspector set a different file name for the "FileName" property:



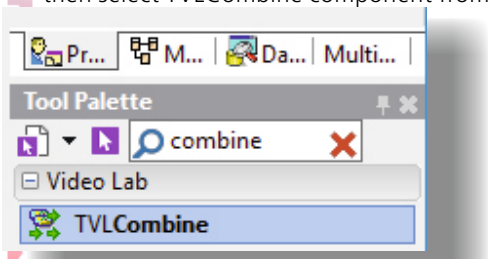
Type "vlgeneric" in the Tool Palette search box, then select TVLGenericFilter component from the palette:



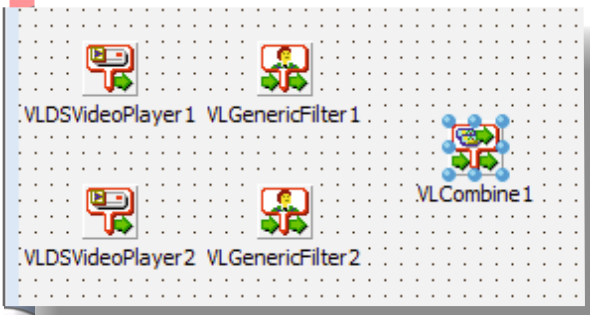
And drop 2 of them on the Form.



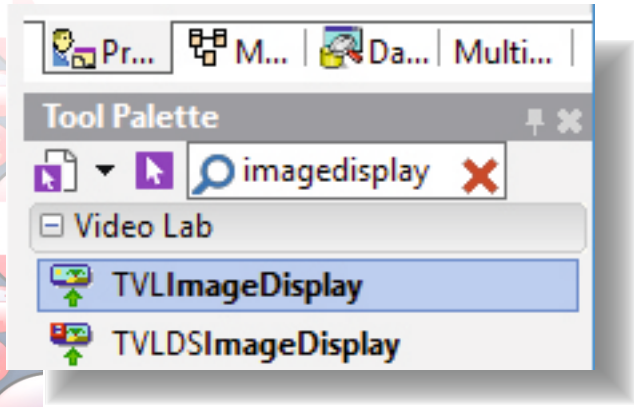
Type "combine" in the Tool Palette search box, then select TVLCombine component from the palette:



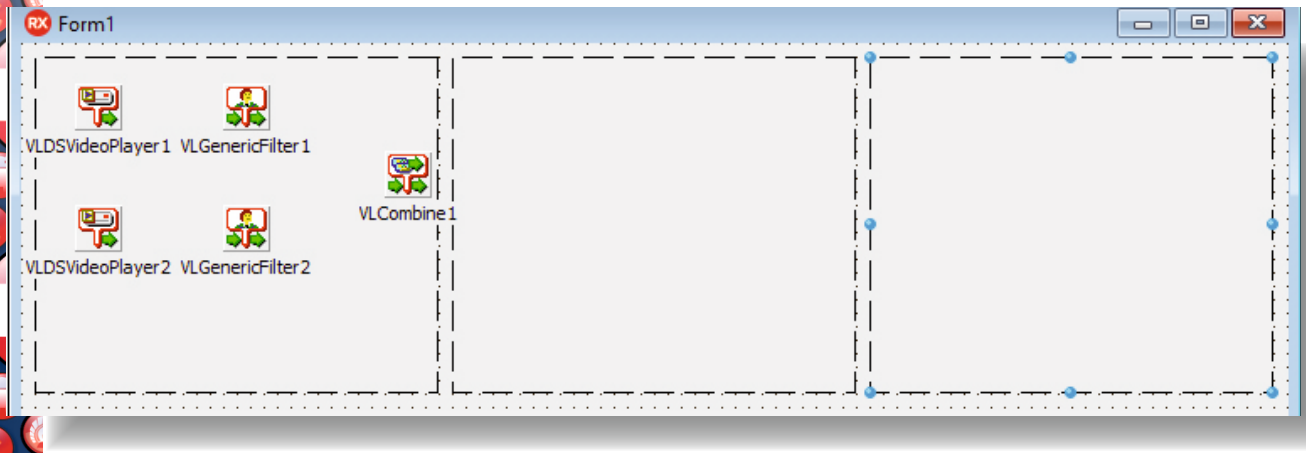
And drop it on the Form.



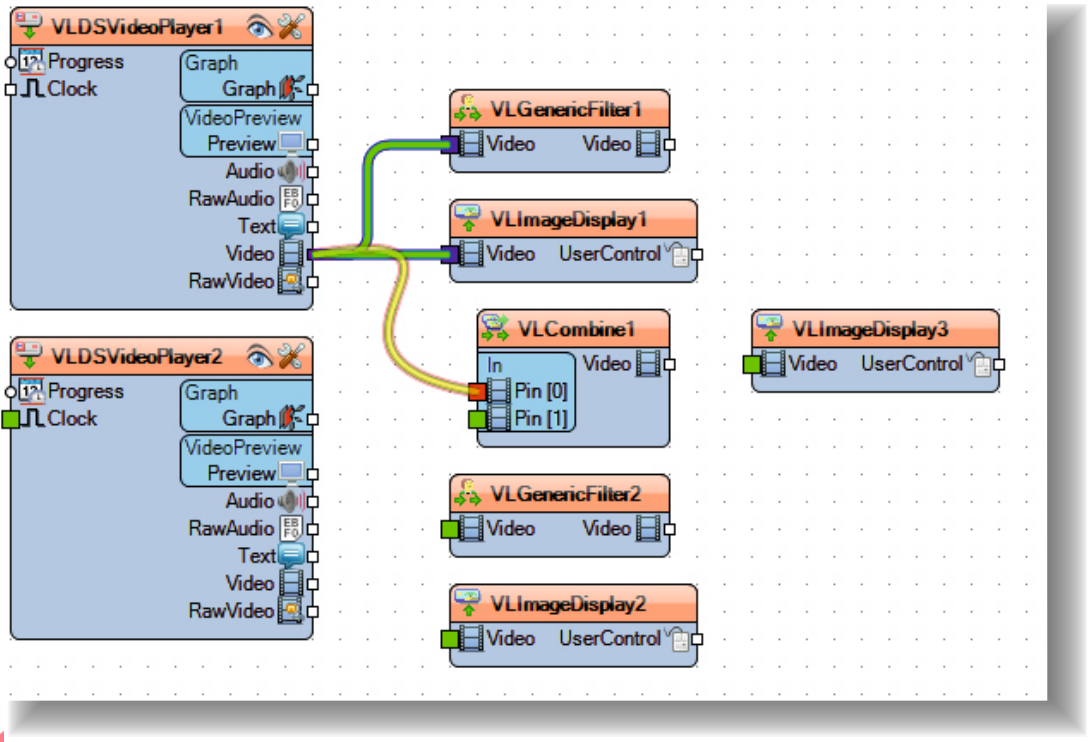
Type "imagedisplay" in the Tool Palette search box, then select TVLImageDisplay component from the palette:



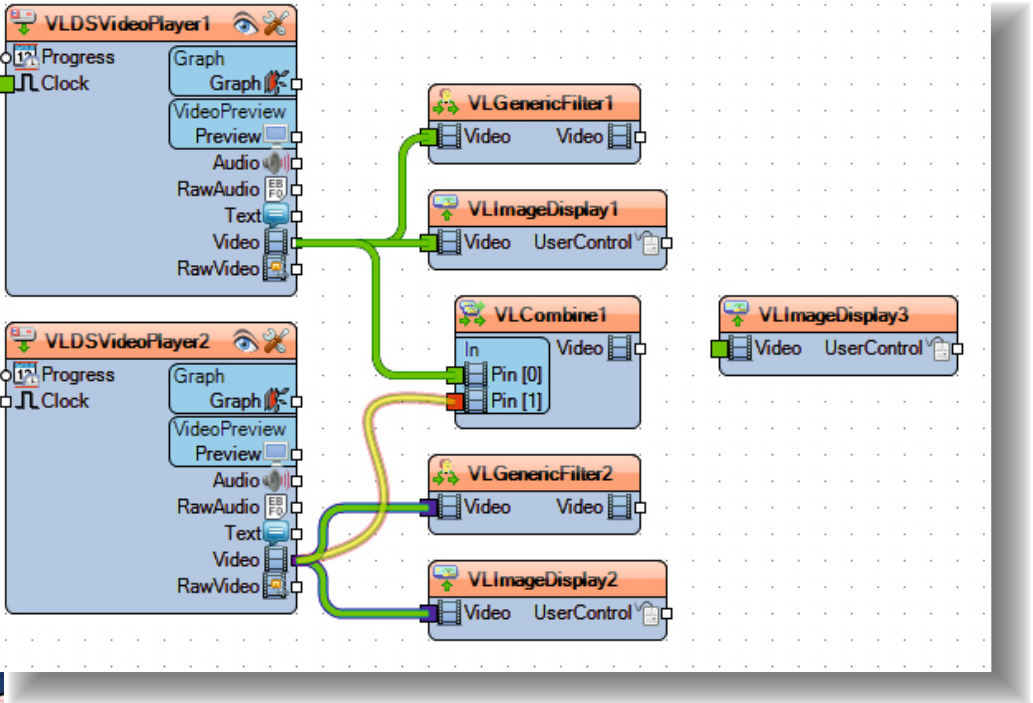
And place 3 of them on the Form:



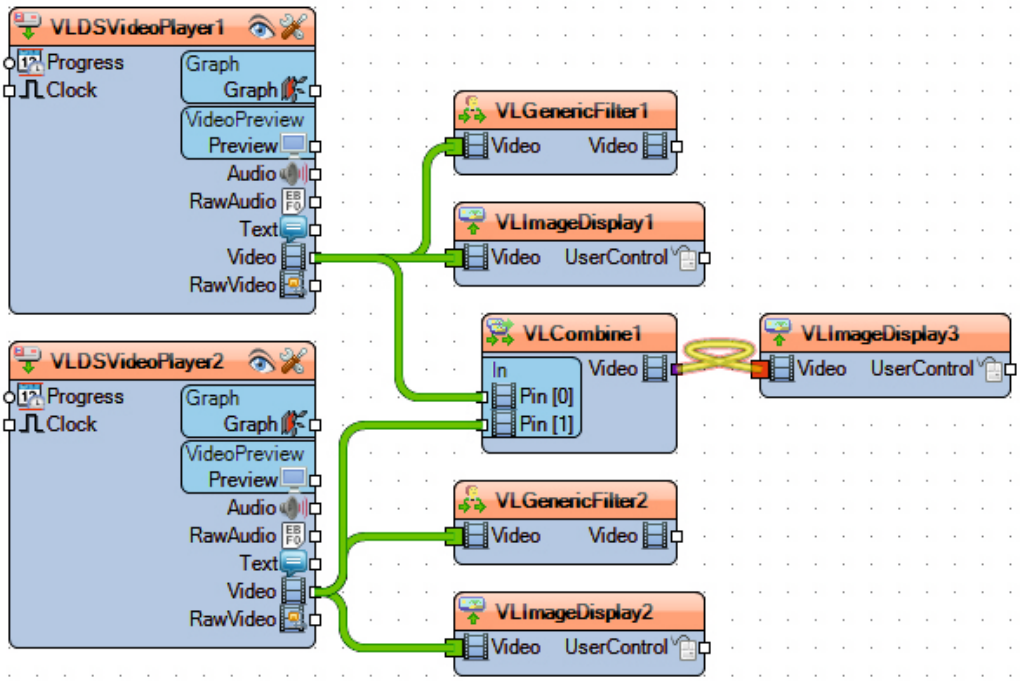
Switch to the **"Open Wire"** tab.
Connect the **"Video"** output pin of the **VLDSVideoPlayer1** component to the **"Video"** input pin of the **VLGenericFilter1** component.
Connect the **"Video"** output pin of the **VLDSVideoPlayer1** component to the **"Video"** input pin of the VLImageDisplay1 component.
Connect the **"Video"** output pin of the **VLDSVideoPlayer1** component to the **"Pin[0]"** input pin of the **"In"** pin list of the **VLCombine1** component as shown on the picture:



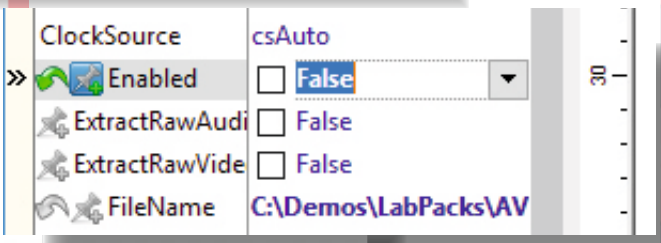
Connect the **"Video"** output pin of the **VLDSVideoPlayer2** component to the **"Video"** input pin of the **VLGenericFilter2** component.
Connect the **"Video"** output pin of the **VLDSVideoPlayer2** component to the **"Video"** input pin of the **VLImageDisplay2** component.
Connect the **"Video"** output pin of the **VLDSVideoPlayer2** component to the **"Pin[1]"** input pin of the **"In"** pin list of the **VLCombine1** component as shown on the picture:

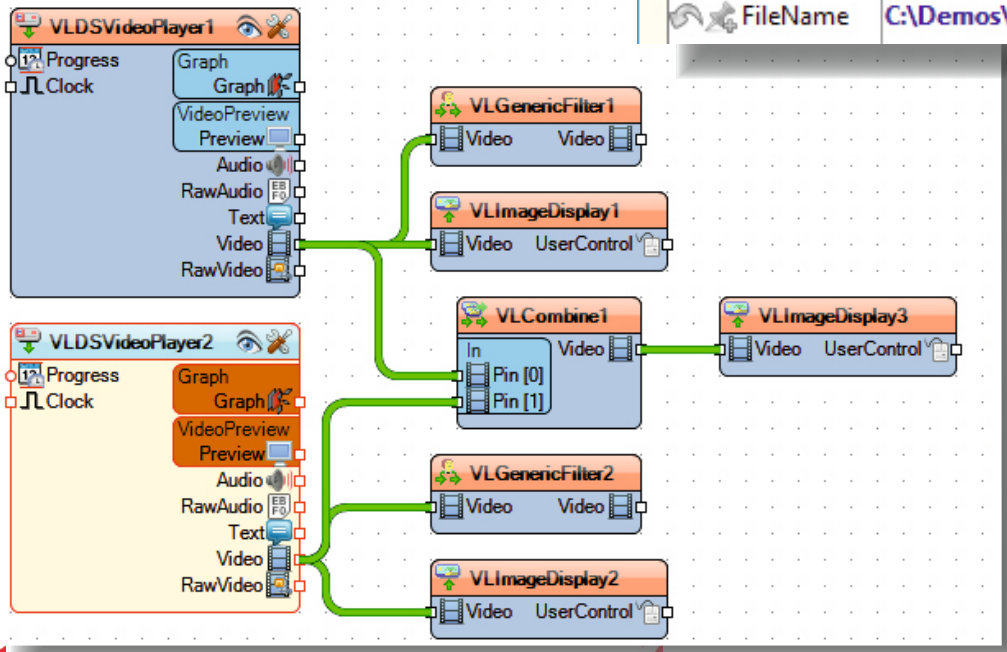


Connect the "Video" output pin of the VLCombine1 component to the "Video" input pin of the VLImageDisplay3 component:



We want the first player to start playing when we start the application, and to have the second one waiting to start later. For this we will disable the second player.
Select the VLDSVideoPlayer2 component.
In the Object Inspector set the value of the "Enabled" property to "False":





Switch to the "Code" tab and add:

```
FFile1Counter : Integer;
FFile2Counter : Integer;
```

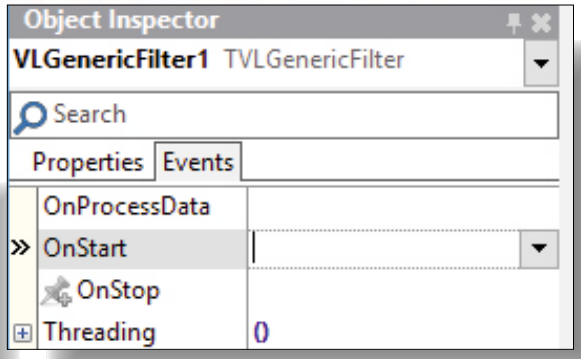
to the **"private"** section of the TForm1 class:

```
TForm1 = class(TForm)
 VLGenericFilter1:TVLGenericFilter;
 VLGenericFilter2:TVLGenericFilter;
 VLCombine1:TVLCombine;
 VLImageDisplay1:TVLImageDisplay;
 VLImageDisplay2:TVLImageDisplay;
 VLImageDisplay3:TVLImageDisplay;
 VLDSVideoPlayer2:TVLDSVideoPlayer;
 VLDSVideoPlayer1:TVLDSVideoPlayer;
private
 { Private declarations }
 FFile1Counter:Integer;
 FFile2Counter:Integer;

public
 { Public declarations }
end;
```

We will need to open the second player while the first one is playing. Opening takes time so we will schedule it to execute while the first player is playing. For this we will use the **MainThreadExecute** procedure from **`Mitov.Threading`**. In the **"Implementation"** section of the unit add:

```
uses
  Mitov.Threading;
```

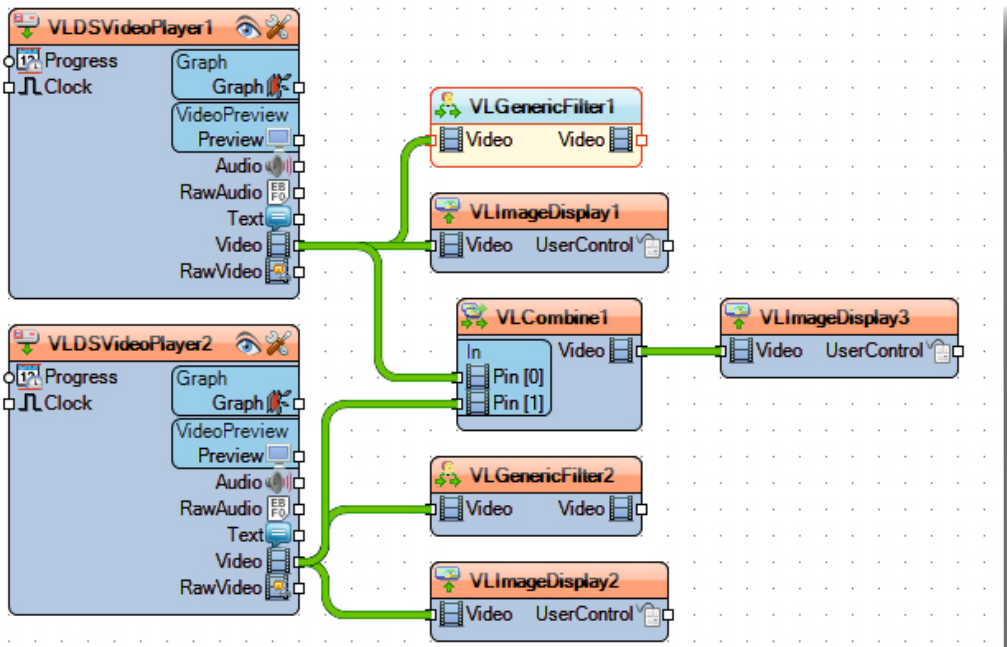
Switch to the **"Open Wire"** tab.
Select the **VLGenericFilter1** component.
In the **Object Inspector** switch to the **"Events"** tab, and select the **`OnStart`** event.
Double-click on the editing area of the **`OnStart`** event to generate the event handler:



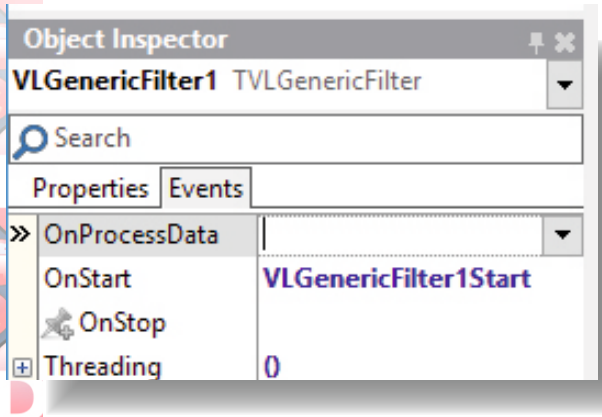
In the event handler add the following code:

```
procedure TForm1.VLGenericFilter1Start(
   Sender:TObject;var AWidth,
   AHeight:Integer;AFrameDelay:Real);
begin
  FFile1Counter:=0;
  MainThreadExecute(Self,True,
   procedure()
    begin
     VLDSVideoPlayer2.Open();
    end
  );

end;
```

Here we use **MainThreadExecute** to schedule **`VLDSVideoPlayer2.Open();`** to be executed as soon as possible in the main thread. The second **"True"** parameter instructs the function, that if the call is made from the Main thread, the execution to be done not right away, but after the currently scheduled tasks of the main thread are handled.
We also set the **FFile1Counter** to 0 so we can start counting the frames as they play.



Switch to the **"Open Wire"** tab.
In the Object Inspector select the **OnProcessData** event.
Double-click on the editing area of the **OnProcessData** event to generate the event handler:



In the event handler add the following code:

```
procedure TForm1.VLGenericFilter1ProcessData(Sender:TObject;
 InBuffer:IVLImageBuffer; var OutBuffer:IVLImageBuffer;
 var SendOutputData:Boolean);
begin
 Inc(FFile1Counter);
 if(FFile1Counter=VLDSVideoPlayer1.FramesCount-100)then
  VLDSVideoPlayer2.Start();
end;
```

Here we increment the **FFile1Counter** and if we reach the last 100 frames, start the second player.

Switch to the "Open Wire" tab.
Select the VLGenericFilter2 component. In the Object Inspector select the OnStart event.
Double-click on the editing area of the OnStart event to generate the event handler:

In the event handler add the following code:

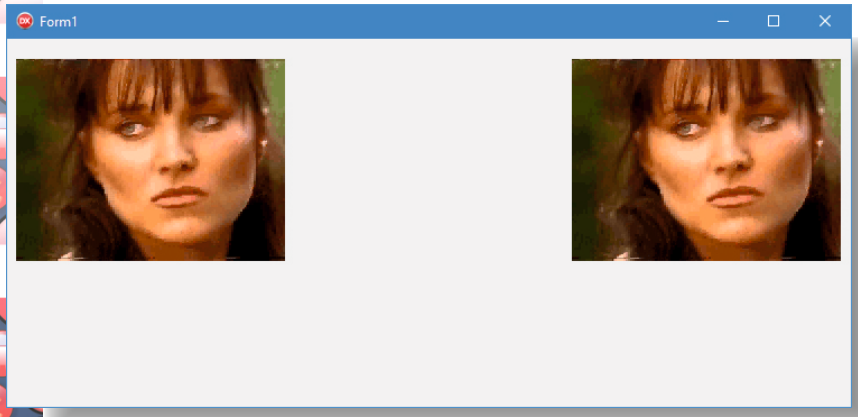
```
procedure TForm1.VLGenericFilter2Start(Sender:TObject;
var AWidth, AHeight:Integer;AFrameDelay:Real);
begin
 FFile2Counter:=0;
 MainThreadExecute(Self,True,
   procedure()
   begin
    VLDSVideoPlayer1.Open();
   end
  );

end;
```

Here we do the same as for the other player, and setting FFile2Counter, instead of FFile1Counter.

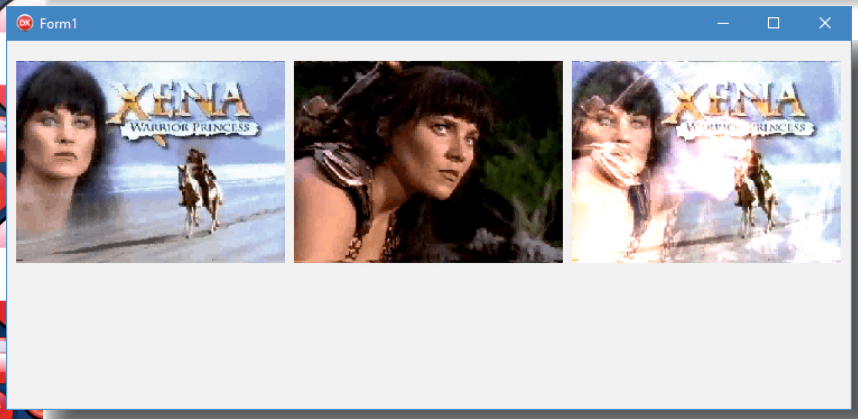
Switch to the "Open Wire" tab. In the Object Inspector select the OnProcessData event. Double-click on the editing area of the OnProcessData event to generate the event handler: In the event handler add the following code:

```
procedure TForm1.VLGenericFilter2ProcessData(Sender:TObject;
 InBuffer:IVLImageBuffer;var OutBuffer:IVLImageBuffer;
 var SendOutputData:Boolean);
begin
 Inc(FFile2Counter);
 if(FFile2Counter=VLDSVideoPlayer2.FramesCount-100)then
  VLDSVideoPlayer1.Start();

end;
```

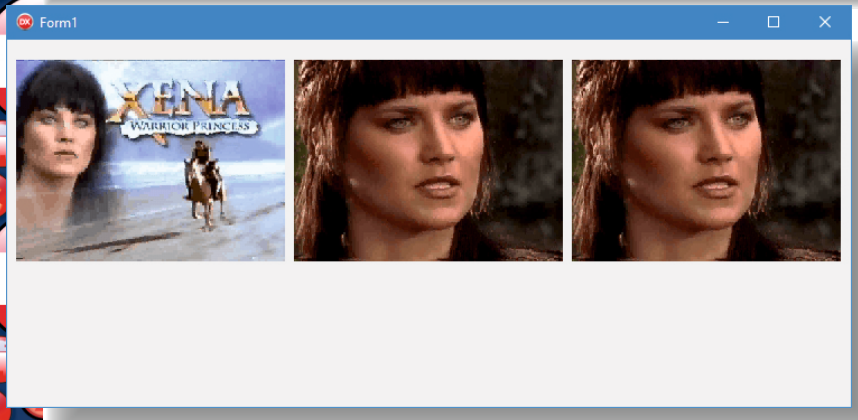
The code is again identical to the one we did for the other player.



Compile and run the application. You should see the first file staring to play both in the first image display and the third one:



When the file reaches the last 100 frames, the second video will start playing and will be mixed with the first one:



When the first file finishes playing only the second will continue until it reaches the last 100 frames, then it will start the first one, and the process will repeat:

Close the application.
Here is the complete source code for this project:

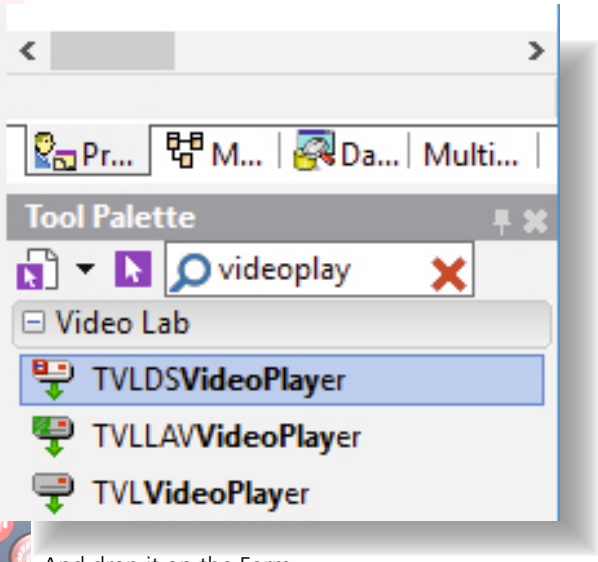
```
unit Unit1;

interface

uses
 Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
 Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Mitov.VCLTypes, VCL.LPControl,
 SLControlCollection, VLCommonDisplay, VLImageDisplay, VLMultiInput,
 VLCommonCombine, VLCombine, LPComponent, SLCommonFilter, VLCommonFilter,
 VLBasicGenericFilter, VLGenericFilter, Mitov.Types, MLDSPlayer, VLDSVideoPlayer;

type
 TForm1 = class(TForm)
  VLDSVideoPlayer1: TVLDSVideoPlayer;
  VLDSVideoPlayer2: TVLDSVideoPlayer;
  VLGenericFilter1: TVLGenericFilter;
  VLGenericFilter2: TVLGenericFilter;
  VLCombine1: TVLCombine;
  VLImageDisplay1: TVLImageDisplay;
  VLImageDisplay2: TVLImageDisplay;
  VLImageDisplay3: TVLImageDisplay;
  procedure VLGenericFilter1Start(Sender: TObject; var AWidth,
   AHeight: Integer; AFrameDelay: Real);
  procedure VLGenericFilter1ProcessData(Sender: TObject;
   InBuffer: IVLImageBuffer; var OutBuffer: IVLImageBuffer;
   var SendOutputData: Boolean);
  procedure VLGenericFilter2Start(Sender: TObject; var AWidth,
   AHeight: Integer; AFrameDelay: Real);
  procedure VLGenericFilter2ProcessData(Sender: TObject;
   InBuffer: IVLImageBuffer; var OutBuffer: IVLImageBuffer;
   var SendOutputData: Boolean);
 private { Private declarations }
  FFile1Counter: Integer;
  FFile2Counter: Integer;

 public { Public declarations }
 end;

var  Form1: TForm1;

implementation

{$R *.dfm}

uses Mitov.Threading;

procedure TForm1.VLGenericFilter1ProcessData(Sender: TObject; InBuffer:
IVLImageBuffer; var OutBuffer: IVLImageBuffer;
 var SendOutputData: Boolean);
begin
 Inc(FFile1Counter);
 if(FFile1Counter = VLDSVideoPlayer1.FramesCount - 100) then
  VLDSVideoPlayer2.Start();

end;

procedure TForm1.VLGenericFilter1Start(Sender: TObject; var AWidth,
 AHeight: Integer; AFrameDelay: Real);
begin
 FFile1Counter := 0;
 MainThreadExecute(Self, True,
   procedure()
   begin
    VLDSVideoPlayer2.Open();
   end
  );

end;
```

```
procedure TForm1.VLGenericFilter2ProcessData(Sender:TObject;
 InBuffer:IVLImageBuffer;var OutBuffer:IVLImageBuffer;
 var SendOutputData:Boolean);
begin
 Inc(FFile2Counter);
 if(FFile2Counter=VLDSVideoPlayer2.FramesCount-100)then
  VLDSVideoPlayer1.Start();

end;

procedure TForm1.VLGenericFilter2Start(Sender:TObject;var AWidth,
 AHeight:Integer;AFrameDelay:Real);
begin
 FFile2Counter:=0;
 MainThreadExecute(Self,True,
   procedure()
   begin
    VLDSVideoPlayer1.Open();
   end
  );

end;

end.
```

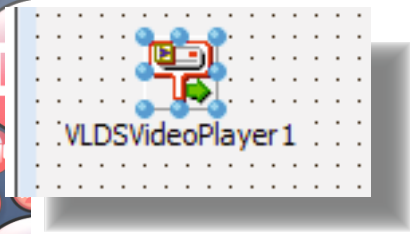
Often we need to split a single video file into multiple shorter video files. We can do this quite easily with VideoLab. For this we can use logger to record a file, and from time to time change the FileName property. The old file will be closed, and the logger will start recording into a new file.

Start a new VCL Form application.

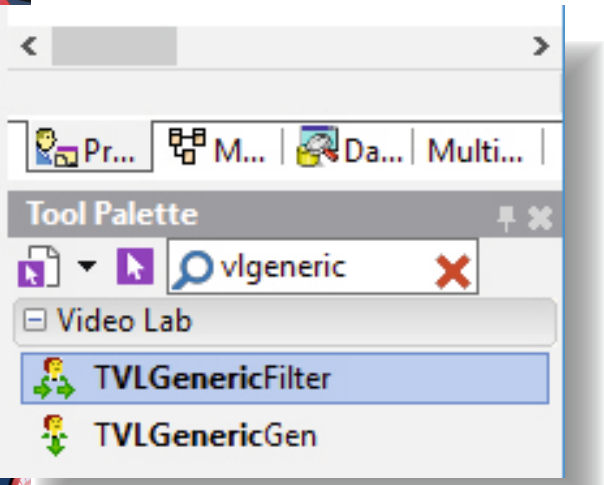
Type "videoplayer" in the Tool Palette search box, then select TVLDSVideoPlayer component from the palette:



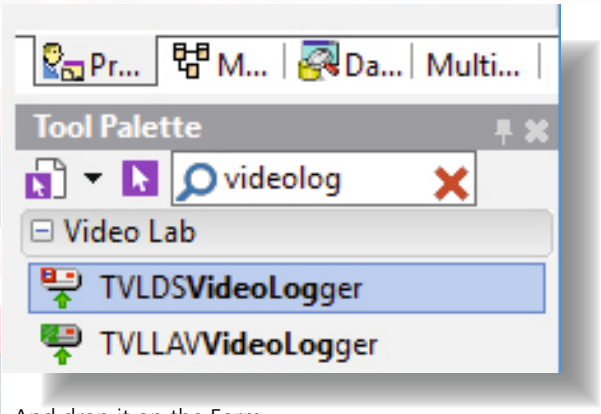
And drop it on the Form.



Type "vlgeneric" in the Tool Palette search box, then select TVLGenericFilter component from the palette:



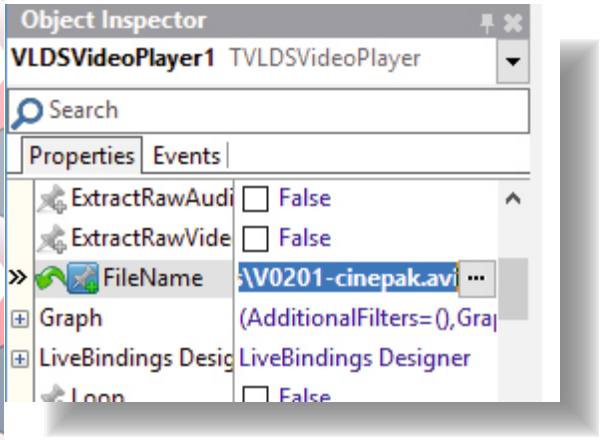
And drop it on the Form.
Type "videolog" in the Tool Palette search box, then select TVLDSVideoLogger component from the palette:



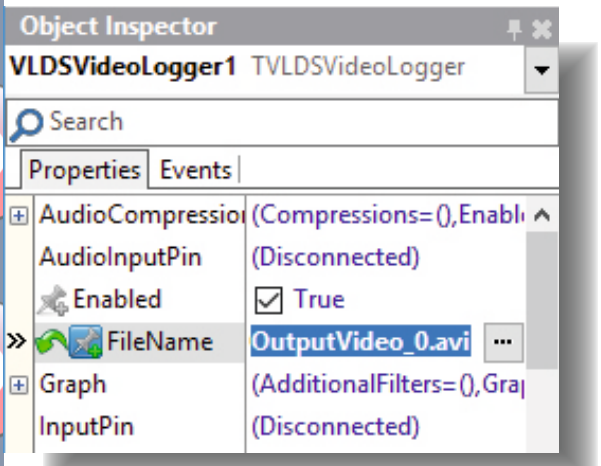
And drop it on the Form.
Type "imagedisplay" in the Tool Palette search box, then select TVLImageDisplay from the palette:

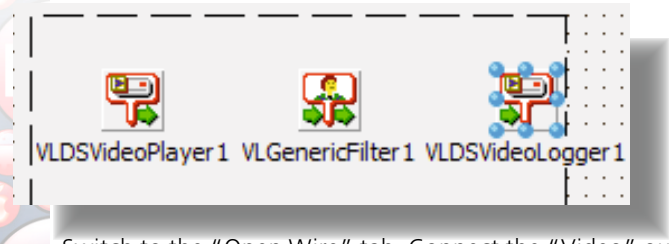
And drop it on the Form.

Select the VLDSVideoPlayer1 component and in the Object Inspector set a file name for the "FileName" property:

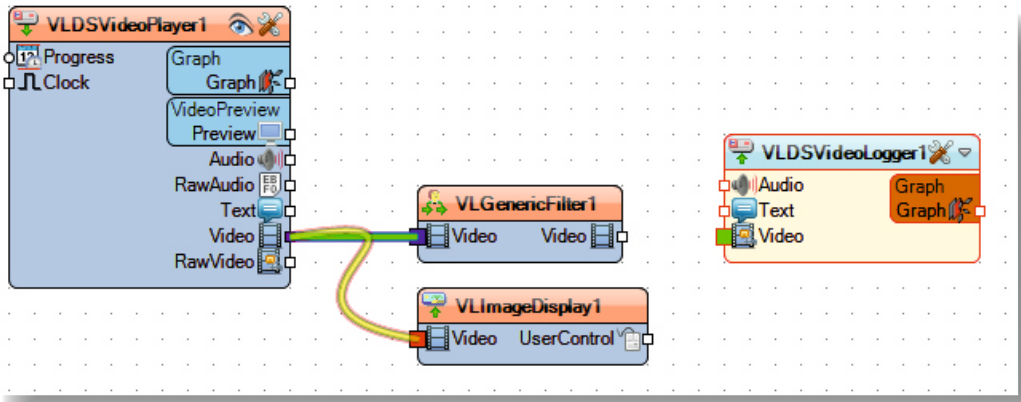


Select the VLDSVideoLogger1 component and in the Object Inspector set the value of the "FileName" property to "OutputVideo_0.avi":

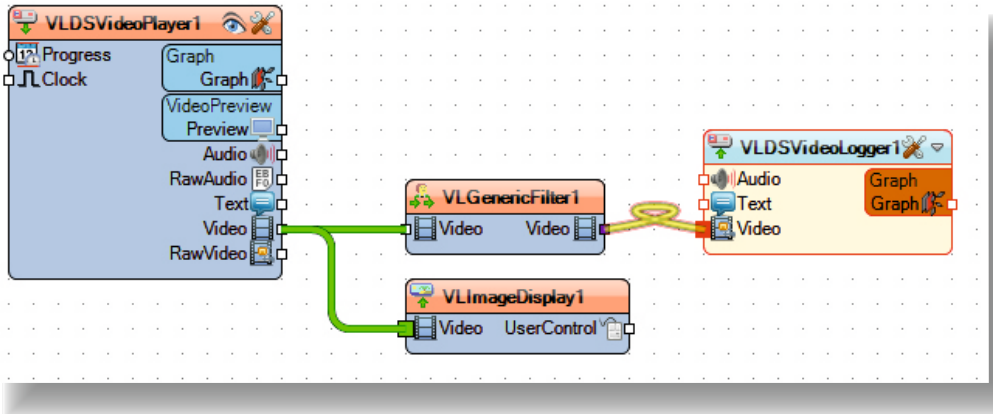




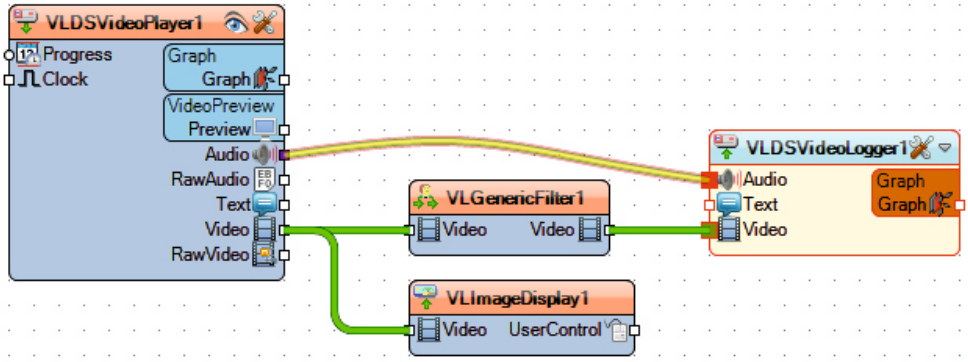
Switch to the "Open Wire" tab. Connect the "Video" output pin of the VLDSVideoPlayer1 component to the "Video" input pin of the VLGenericFilter1 component. Connect the "Video" output pin of the VLDSVideoPlayer1 component to the "Video" input pin of the VLImageDisplay1 component:



Connect the "Video" output pin of the VLGenericFilter1 component to the "Video" input pin of the VLDSVideoLogger1 component:



Connect the "Audio" output pin of the VLDSVideoPlayer1 component to the "Audio" input pin of the VLDSVideoLogger1 component:



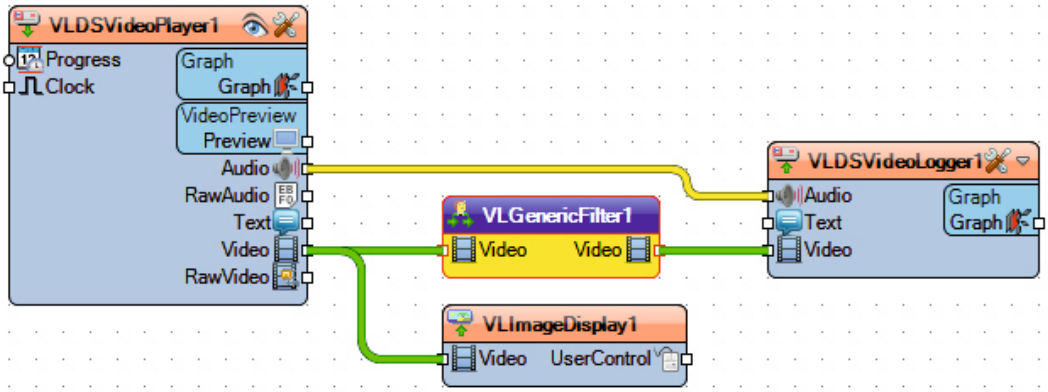
Switch to the "Code" tab and add:

```
FFrameNo  : Integer;
```

to the "private" section of the TForm1 class:

```
TForm1 = class(TForm)
 VLDSVideoPlayer1: TVLDSVideoPlayer;
 VLGenericFilter1: TVLGenericFilter;
 VLDSVideoLogger1: TVLDSVideoLogger;
 VLImageDisplay1: TVLImageDisplay;
private
 { Private declarations }
 FFrameNo : Integer;

public
 { Public declarations }
```

Switch to the "Open Wire" tab, and double-click on the VLGenericFilter1 component to generate the OnProcessData event handler:

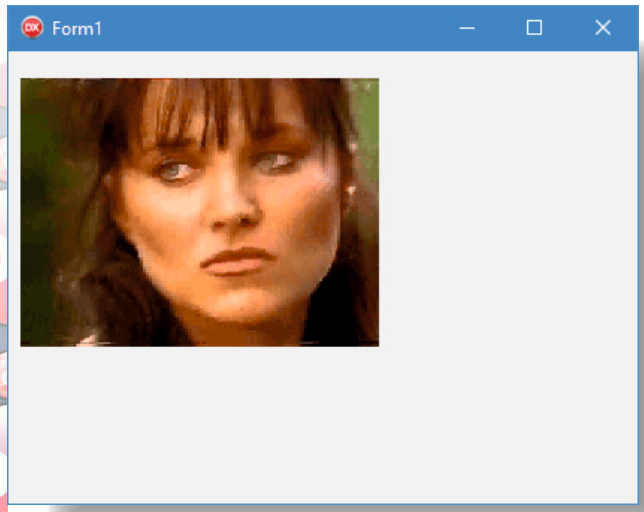


In the event handler, add the following code:

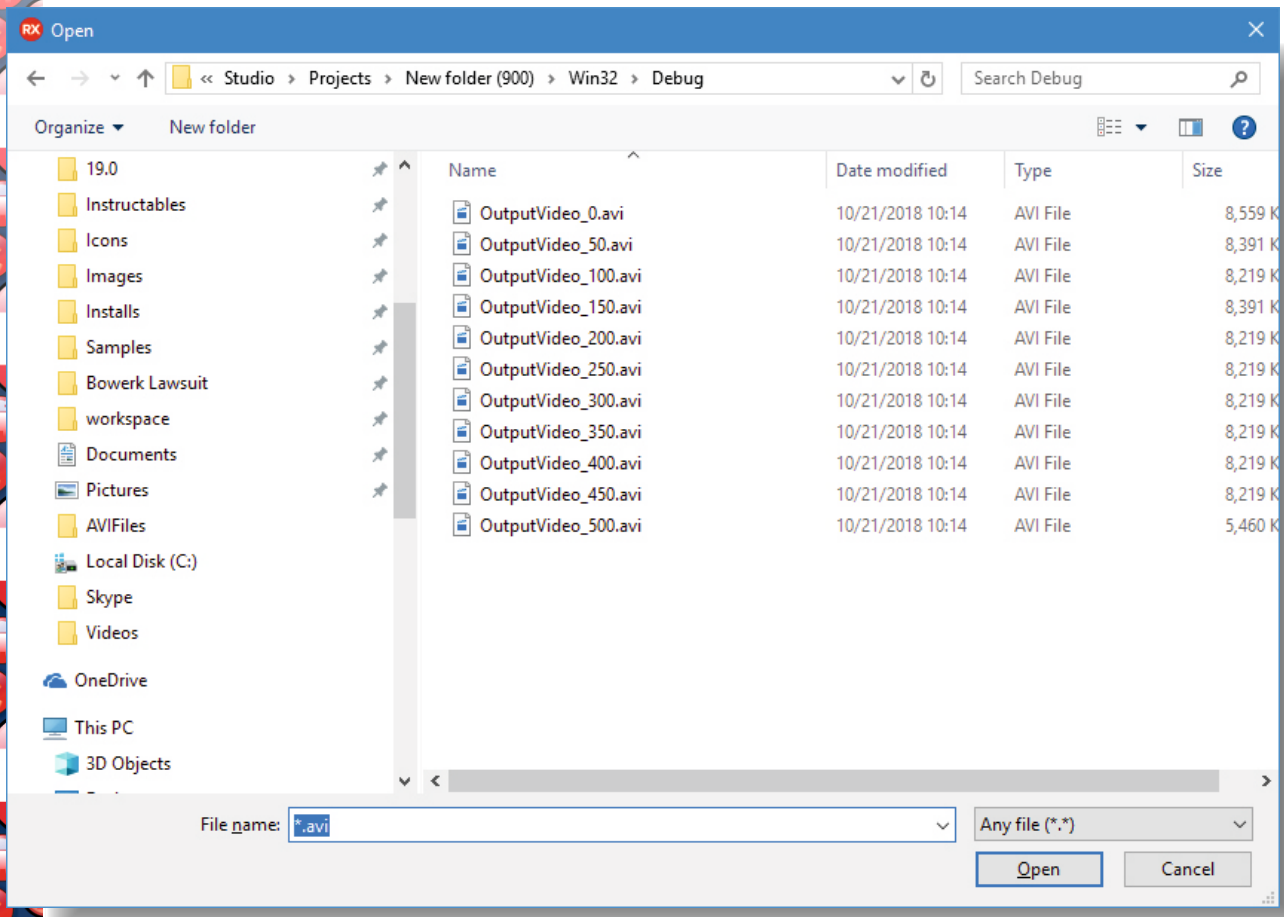
```
procedure TForm1.VLGenericFilter1ProcessData(Sender: TObject;
 InBuffer: IVLImageBuffer; var OutBuffer: IVLImageBuffer;
 var SendOutputData: Boolean);
begin
 Inc(FFrameNo);

 if ((FFrameNo mod 50) = 0) then
  VLDSVideoLogger1.FileName := 'OutputVideo_' + FFrameNo.ToString() + '.avi';

end;
```

Here we will change the recorded file name every 50 frames, breaking the video into smaller files. Compile and run the application.

You will see the video playing:



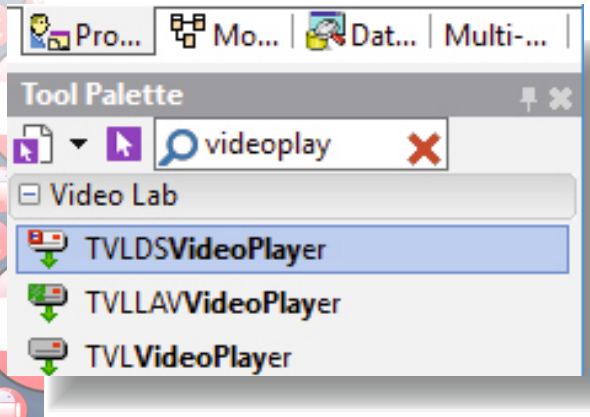
After the video finishes playing, multiple smaller files will be created:



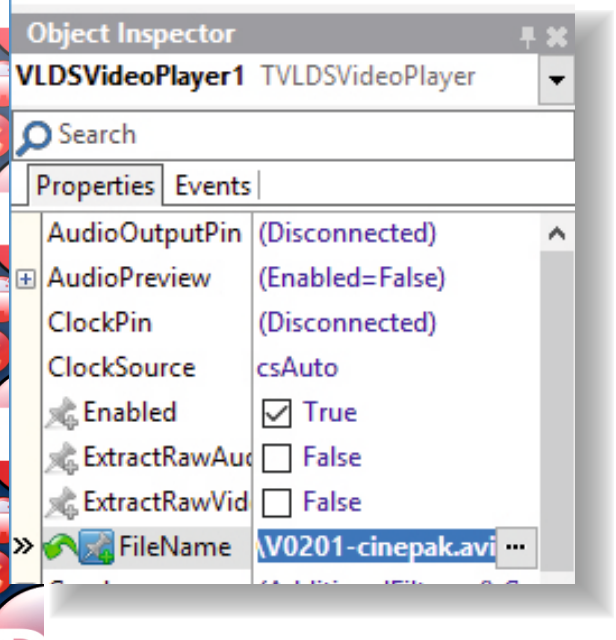
Close the application.

VideoLab can also be used to merge smaller files into a single big file.

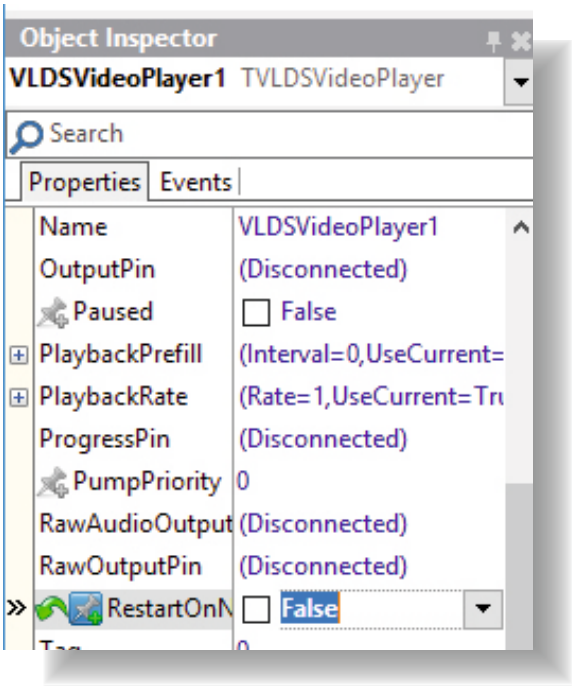
Start a new VCL Form application.
Type "videoplayer" in the Tool Palette search box, then select TVLDSVideoPlayer component from the palette:



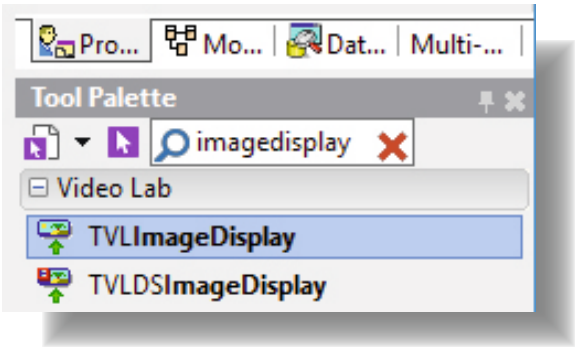
And drop it on the form.
In the Object Inspector set a file name for the "FileName" property:



In the Object Inspector set the value of the "RestartOnNewFile" property to "False":

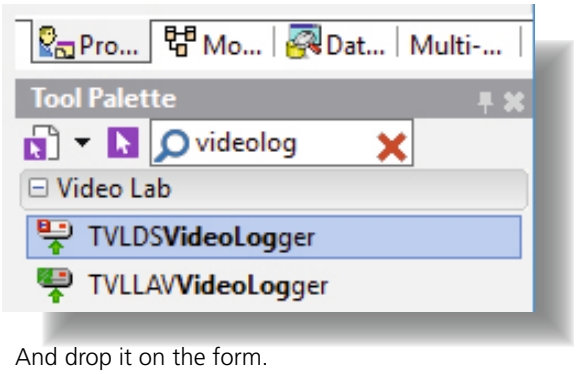


Type "imagedisplay" in the Tool Palette search box, then select TVLImageDisplay from the palette:



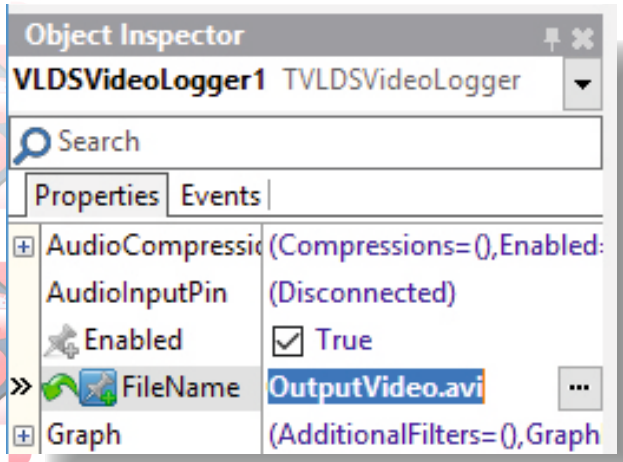
And drop it on the form.

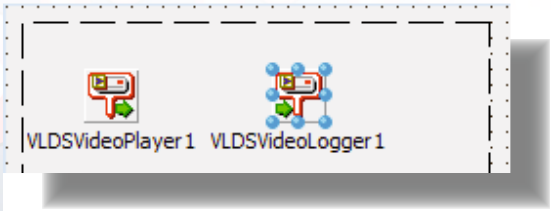
Type "videolog" in the Tool Palette search box, then select TVLDSVideoLogger component from the palette:



And drop it on the form.

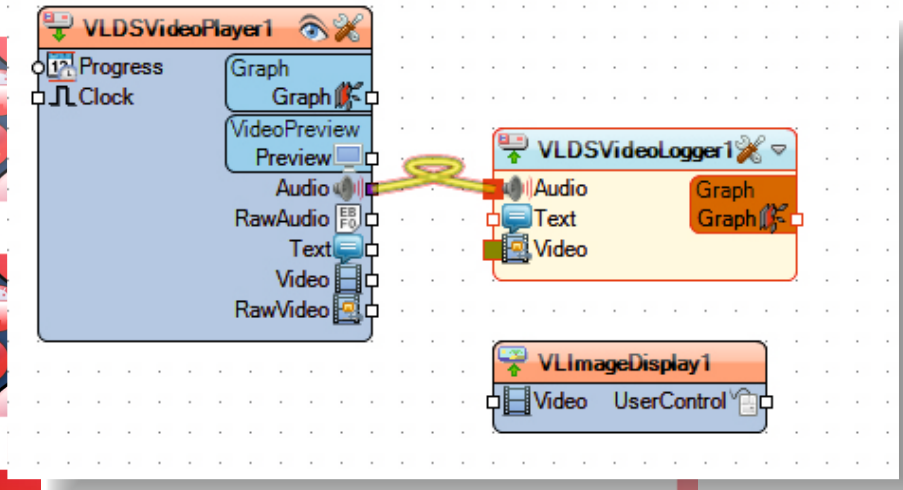
In the Object Inspector set the value of the "FileName" property to "OutputVideo.avi":

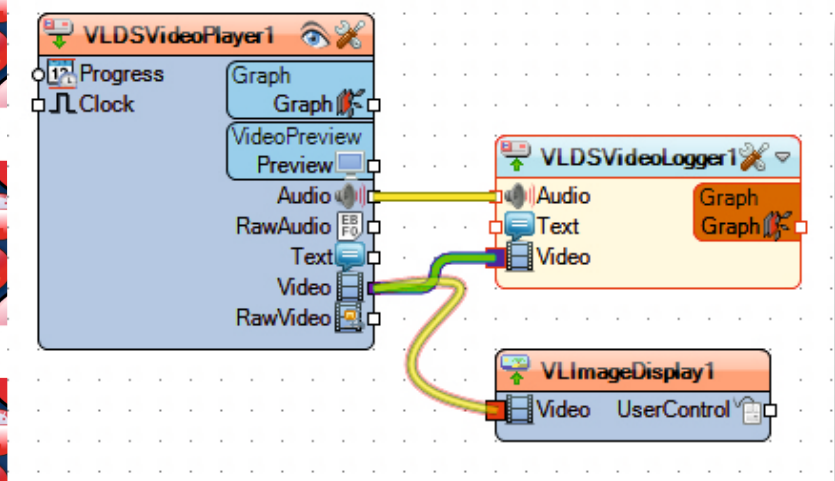




Switch to the **"Open Wire"** tab.

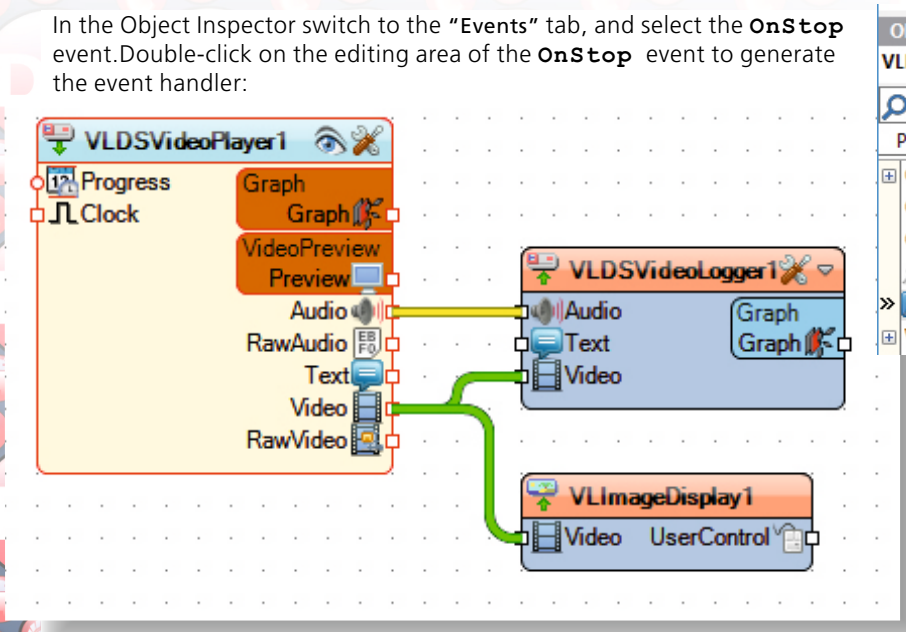
Connect the "Audio" output pin of the VLDSVideoPlayer1 component to the "Audio" input pin of the VLDSVideoLogger1 component:

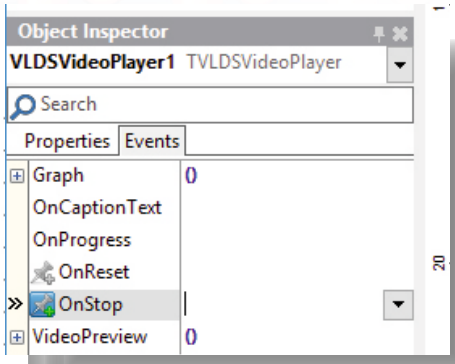




Connect the "Video" output pin of the **VLDSVideoPlayer1** component to the "Video" input pin of the **VLDSVideoLogger1** component. Connect the "Video" output pin of the **VLDSVideoPlayer1** component to the "Video" input pin of the **VLImageDisplay1** component:

In the Object Inspector switch to the **"Events"** tab, and select the **OnStop** event.Double-click on the editing area of the **OnStop** event to generate the event handler:



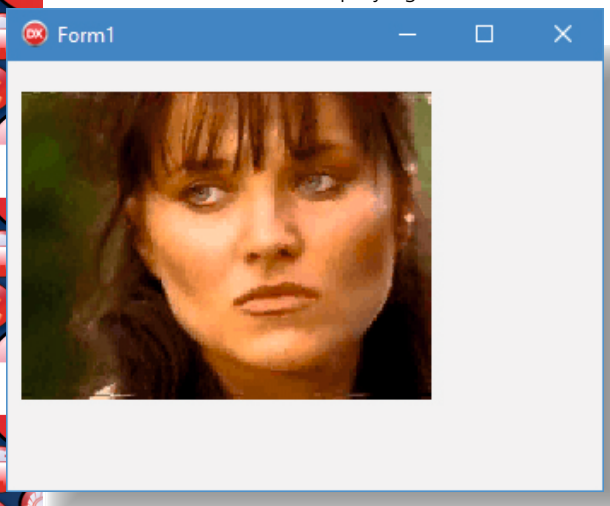


In the event handler add the following code:

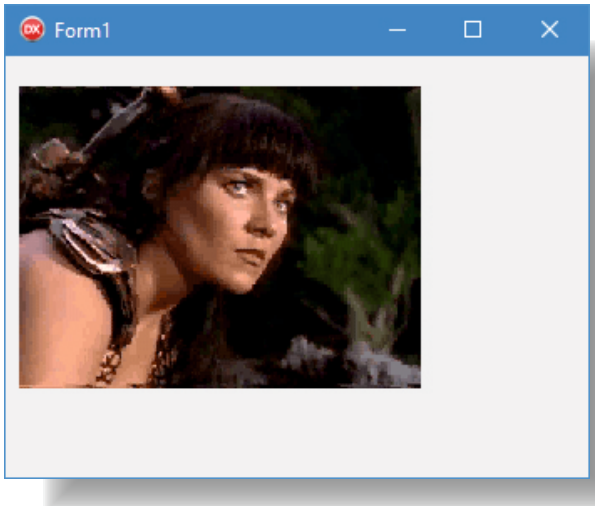
```
procedure TForm1.VLDSVideoPlayer1Stop(Sender:TObject);
begin
 VLDSVideoPlayer1.OnStop:=NIL;
 VLDSVideoPlayer1.FileName:=
'C:\Demos\LabPacks\AVIFiles\V0206-cinepak.avi';
 VLDSVideoPlayer1.Start();
end;
```

Here we make sure the **VLDSVideoPlayer1Stop** will not be called again by setting **VLDSVideoPlayer1.OnStop** to **NIL**, set new file name for the **VLDSVideoPlayer1**, and start the **VLDSVideoPlayer1**.

Compile and run the Application.
You will see the first video playing:



And when it finishes the second will start:



A file containing both videos one after another will be created. Close the application.

Here is the complete source code for this project:

```
unit Unit1;

interface

uses
 Winapi.Windows,Winapi.Messages,System.SysUtils,System.Variants,
 System.Classes,Vcl.Graphics, Vcl.Controls,Vcl.Forms,Vcl.Dialogs,LPComponent,
 SLCommonFilter, VLCommonLogger,VLDSVideoLogger,Mitov.VCLTypes,
 VCL.LPControl, SLControlCollection,VLCommonDisplay,VLImageDisplay,
 Mitov.Types,MLDSPlayer, VLDSVideoPlayer;

type
 TForm1 = class(TForm)
  VLDSVideoPlayer1: TVLDSVideoPlayer;
  VLImageDisplay1: TVLImageDisplay;
  VLDSVideoLogger1: TVLDSVideoLogger;
  procedure VLDSVideoPlayer1Stop(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.VLDSVideoPlayer1Stop(Sender: TObject);
begin
 VLDSVideoPlayer1.OnStop := NIL;
 VLDSVideoPlayer1.FileName := 'C:\Demos\LabPacks\AVIFiles\V0206-
cinepak.avi';
 VLDSVideoPlayer1.Start();
end;

end.
```

Conclusion:

In this article I showed you how you can combine and mix video sources, Implement Picture In Picture effect, perform video transitions, split and merge videos. For video sources we used variety of video players, but the same can be done with IP or Web camera, TV Tuner, internet stream, or any other video source.

In the following articles I will show you how you can study and analyze the video and audio, and how you can use video or audio with Artificial Intelligence classifiers, or perform computer vision tasks recognizing or tracking objects in the video.



DESCRIPTION
Venez au Congrès Delphi à Paris le 15 Novembre!
Nous organisons une présentation exceptionnelle de la nouvelle version de RAD Studio 10.3 Rio ! Le Congrès est organisé par Barnsten, le représentant officiel d'Embarcadero en France. Nous ne sommes pas seuls ....

Ce jour-là, vous pourrez apprendre les dernières évolutions techniques et bénéficier des conseils et astuces présentés par des experts Delphi en France.

Embarcadero commencera la présentation avec leur dernière vision, Road Map et continuera ensuite par la présentation technique de 10.3 Rio.

Agenda :

9:00 - 9:30 Café d'accueil

9:30 - 10:00 Embarcadero/Barnsten - Maxime Capellot

10:00 - 12:00 Présentation de toutes les nouveautés RAD Studio 10.3 Patrick Premartin (Olf Software)

12:00 - 12:30 Présentation de 3 applications utilisant les derniéres technologies Delphi - Maxime Capellot (Barnsten)
12:30 - 14:00 Repas

14:00 - 15:00 Une nouvelle méthode RADicale pour développer des applications web modernes - Bruno Fierens (TMS Software)

15:00 - 16:00 Plongée dans les LiveBindings - Serge Girard

16:00 - 16:20 Pause gourmande

16:20 - 17:20 Comment protéger vos données
Marion Candau (MVP Embarcadero)

Cliquez ici pour un aperçu de l'ensemble du programme.
Cette journée vous donnera amplement l'occasion de vous familiariser avec l'équipe Barnsten, les experts Delphi et bien sûr differents partenaires et développeurs Delphi.

Nous nous réjouissons de pouvoir vous rencontrer personnellement.

L'equipe Barnsten

Adresse l'événement:
Paris Story
11 bis rue Scribe
75009 Paris
France

starter expert DX

Intro
There have been some questions about how to build a server, with authorization and login management, where the users and their roles are defined in a database.

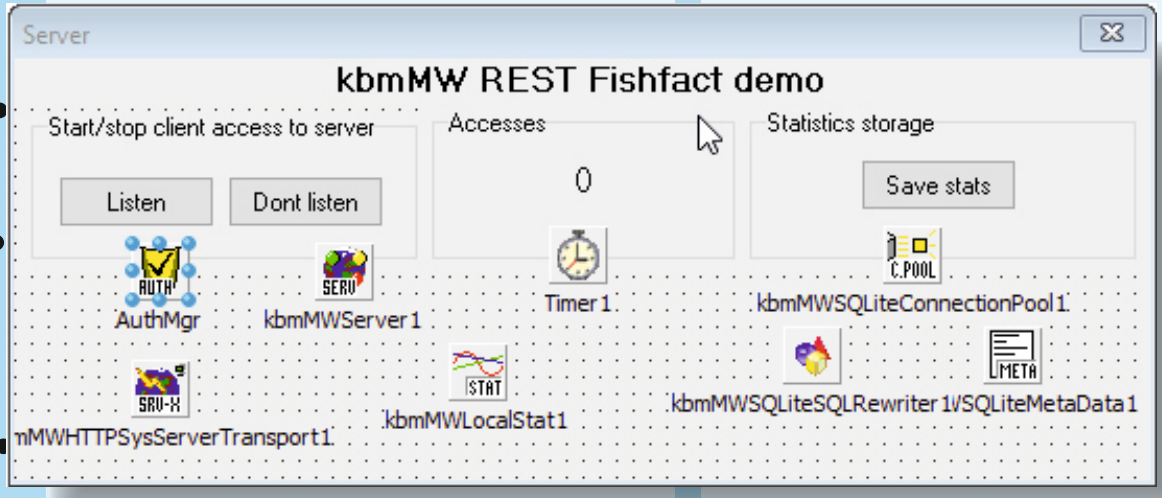
This blog post explains one way of doing that using the TkbmMWAuthorizationManager. Please refer to the previous post (REST easy with kbmMW #4 – Access management) for additional general information.

First we should have some server that needs login support. For this sample, I have chosen the FishFact REST server, which where built in the blog
REST easy with kbmMW #12 – Fishfact demo using HTTP.sys transport.

Adding login security

Based on that server we add a **`TjkbmMWAuthorizationManager`** to the main form (Unit1).

Then we need to determine how to store and access user information from the database. Since this sample already use the ORM to access the database, it makes sense to continue to do so for the user management.




```
[kbmMW_Table('name:user')]
TUser = class
private
  FID:kbmMWNullable<string>;
  FName:kbmMWNullable<string>;
  FPassword:kbmMWNullable<string>;
  FRole:kbmMWNullable<string>;
public
  [kbmMW_Field('name:"id", primary:true, generator:shortGuid',ftString,38)]
  property ID:kbmMWNullable<string> read FID write FID;

  [kbmMW_Field('name:"name"',ftString,50)]
  [kbmMW_NotNull]
  property Name:kbmMWNullable<string> read FName write FName;

  // A secure system should never store plain text passwords, but only SHA256 hashed ones.
  // In that case make room for 64 characters.
  [kbmMW_Field('name:"password"',ftString,50)]
  property Password:kbmMWNullable<string> read FPassword write FPassword;

  [kbmMW_Field('name:"role"',ftString,30)]
  property Role:kbmMWNullable<string> read FRole write FRole;
end;
```

Lets add a class describing a user:

Notice the warning about the password. In this sample we store the unhashed plaintext password in the database. That is a NO NO in a production system. Instead one should store a hashed and salted version of the password… I'll explain later how to modify the code to do that.

For now we accept that the password is unhashed in the database.

In the already existing Form.OnCreate event handler, we should let the ORM ensure that the user table is made available. In addition we also should define the roles that are accepted by this server.

In our sample, there are only two types of users… the anonymous ones and the ones that are logged in with administrator rights. Most of the functionality is made available for anonymous users to use, except one REST call, which require the administrative role. But first things first:

```
procedure TfrmMain.FormCreate(Sender: TObject);
begin
   FORM:=TkbmMWORM.Create;
   FORM.OpenDatabase(kbmMWSQLiteConnectionPool1);
   FORM.CreateOrUpgradeTable(TUser);

   // Add the one single role this application server knows about except anonymous.
   AuthMgr.AddRole('AdminRole');

   kbmMWServer1.AutoRegisterServices;
end;
```

The interesting parts here is the CreateOrUpgradeTable call, which ensures there is a table named user in the database, and the definition of a role called AdminRole.

In Unit1 we should also remember to register the TUser class so kbmMW is aware about its existance. One place to do that is the initialization section of the form unit.

```
...
initialization
 TkbmMWRTTI.EnableRTTI([TUser]);
 kbmMWRegisterKnownClasses([TUser]);

end.
```

Now we must link the knowledge about the user table with the login process of the authorization manager. The crucial point is that the authorization manager is the supreme authority in relation to logins, and as such must know about the actors that are allowed to login. Thus the actors needs to be defined. It can either be done at startup time of the application server, where a complete list of known users are defined as actors towards the authorization manager, or alternatively it can be done on the fly on a need to know bases, which is what I have chosen to show here.

We use the OnLogin event of the authorization manager:

We use the OnLogin event of the authorization manager:

```
procedure TfrmMain.AuthMgrLogin(Sender: TObject; const AActorName,
 ARoleName: string; var APassPhrase: string;  var AActor: TkbmMWAuthorizationActor;
 var ARole: TkbmMWAuthorizationRole;  var AMessage: string);
var
 user:TUser;
begin
  // Lookup user with given name and password.
  user:=ORM.Query<TUser>(['Name','Password'],[AActorName,APassPhrase]);
  if user<>nil then
    try
      // Check if users role is defined. If not complain.
      ARole:=AuthMgr.Roles.Get(user.Role.Value);
      if ARole=nil then
        AMessage:='Role not supported'
      else
      begin
        // Check if actor exists, use it, else create one.
        AActor:=AuthMgr.GetActor(AActorName);
        if AActor=nil then
          AActor:=AuthMgr.AddActor(AActorName,APassPhrase,ARoleName);
        AMessage:='User found and is allowed login';
      end;
    finally
      user.Free;
    end
  else
    AMessage:='User not found';
end;
```

Basically it use the ORM to lookup a user with the given name and password in the database. If one is found, it checks to see if the role that has been defined in the database on the user, exists.

If it does, then it attempts to figure out if the user has already been defined as an actor in the authorization manager. If not then one is defined and all is well.

What if the database is changed… For example if a user changes password? In such case you should not only update the password in the database but also update it in the actor representation in memory.

You can do something like this:

```
procedure TUnit1.UpdateUserPassword(const AUserName, ANewPassword:string);
var
 user:TUser;
begin
 AuthMgr.ChangeActorPassword(AUserName,ANewPassord);
 user:=ORM.Query<TUser>(['Name'],[AUserName]);
 if user<>nil then
   try
    user.Password:=ANewPassword;
    ORM.Update(user);
   finally
    user.Free;
   end;
end;
```

And if the user is to be deleted:

```
procedure TUnit1.RemoveUser(const
AUserName:string);
begin
 AuthMgr.DeleteActor(AUserName);
 ORM.Delete<TUser>(['Name'],[AUserName]);
end;
```

Finally we should define exactly what should be protected by login.

For that we open Unit2.pas which contains the REST service, and choose one or more of the methods to protext. In this case GetSpecieByCategory will now require login as an administrative role, for it to be used.

```
[kbmMW_Rest('method:get, path:"specieByCategory/{category}"')]
[kbmMW_Auth('role:[AdminRole], grant:true')]
function GetSpecieByCategory([kbmMW_Rest('value:"{category}"')]
    const ACategory:string):TBiolifeNoImag
```

Also remember to add kbmMWSecurity to the interface uses clause of the unit. kbmMWSecurity.pas contains the definition of the kbmMW_Auth attribute.

Now we are done. Make sure to add a user with a password and a role of AdminRole to the user table, run the application server and try out the various REST calls.

The moment you try to this call:

http://localhost:1111/biolife/specieByCategory/Butterflyfish

You will be requested for a login by the browser. If you provide the user name and password matching a user in the database having the role AdminRole, you will be shown the result of the request. Otherwise you will only have access to all other REST calls which have no kbmMW_Auth attribute and as such are allowed to be called anonymously.

Hashing passwords

Remember that I mentioned storing (and transferring) plaintext passwords is a no no in production environments?

Hence we should encrypt the password before storage and transfer. However encryption typically means its possible to reverse the encryption, provided the encryption key can be guessed or hacked, which would reveal the password in plain text again. Since users may sometimes reuse the same password on multiple servers, we should make it as hard as possible for potential hackers to get back to the plaintext version of the password.

In a REST setup, its usually a web browser that decides how usernames and passwords are sent. The typical way is actually to leave all encryption to a SSL and user certificates and such to ensure that not only transmitted login data is scrambled, but also all other traffic between the browser and the server. Check the REST easy with kbmMW #3 – SSL blog post for information about one way to secure your REST application server with SSL and certificates.

And that is all fine and good, but we also have the storage part. We really shouldn't store the password in plaintext.

So we will use one way encryption… also known as hashing. It basically calculates a (complex) sum of the original password. Since its a "sum", its usually impossible to reverse the calculation back to the original password, provided a good secure hashing algorithm is used. Fortunately kbmMW provides native support for several secure hashing algorithms. One of the most used ones, that is generally considered secure, is called SHA256.

Now when we receive a password in the OnLogin event, we need to hash it before we do anything with it. It is super simple to do so.

Include **kbmMWHashSHA256** in your uses clause.

```
var
 hashed:string;
begin
 hashed:=TkbmMWHashSHA256.HashAsString(APassPhrase,'somesaltvalue');
...
end;
```

somesaltvalue is some "secret" value you have in your application and that is unique for your application. It can be anything, but prefer a length string of scrambled random characters.

The idea behind a "salt" is that it makes it extremely difficult to attempt to bruteforce guessing the correlation between a plaintext attempt and a calculated SHA256 value. If you simply use the password by itself, then attackers has a much easier time attempting to guess the password, simply because they can try out all combinations of characters and match the hashed result with the sniffed hash value that you have hashed. Adding a salt, ensures that the attacker will have no chance in brute force attacking by trying out all combinations, because regardless of what the attacker attempts to find, it will never be the same as the value you have stored in the database due to the secret salt.

Now we have a hashed string, and that one can be stored in a database, and similarly every time we need to figure out if person has typed in the correct password, we first need to hash it server side (with the correct salt) and then attempt to look the hashed password (and username) up in the database.

PROLOGUE

There are many more features in the authorization manager, which I have not explained here, but visit our site at http://www.components4developers.com, and look for the kbmMW documentations section for whitepapers.

If you like this, please share the word about kbmMW wherever you can and feel free to link, like, share and copy the posts of this blog to where you find they could be useful.

Oh... and whats about that featured image? It's an image of the spiral galaxy M51a, also known as the whirlpool galaxy. Whirlpool is also the name chosen for one of the stronger hashing algorithms, for which there has still not been found any significant weaknesses or attack vectors.





Intro

Ive been asked questions about how to handle POST via kbmMW's smart service based REST.

This blog post explains the typical POST variations, and how to handle them in kbmMW.

POST variants

A web/REST client can POST data in multiple ways:

As value data in the path part of the URL eg.
POST **http://localhost/myservice/myfunction/10/20**
As key/value data in the query part of the URL eg.
POST **http://localhost/myservice/myfunction?arg1=10&arg2=20**
As FORM key/value data in the body. eg.
POST **http://localhost/myservice/myfunction**
and then body contains:
name1=value1&name2=value2
As XML or JSON data in the body.
eg.
POST **http://localhost/myservice/myfunction**
and then body contains:
{"name1 :"value1 ,"name2 :"value2"}
As Multipart body typically as part of a file upload.
You extract data in different ways depending on the way the client has chosen to send them.

1. You use:

```
[kbmMW_Rest('method:post, path: "myfunction/{value1}/{value2}"')]
 function MyFunction( [kbmMW_Rest('value:"{value1}"')] AValue1:integer;
 [kbmMW_Rest('value:"{value2}"')] AValue2:integer):string;
```

2. You use:

```
[kbmMW_Rest('method:post, path: "myfunction"')]
 function MyFunction( [kbmMW_Rest('value:"$value1"')] AValue1:integer;
 [kbmMW_Rest('value:"$value2"')] AValue2:integer):string;
```

3. You use:

```
[kbmMW_Rest('method:post, path: "myfunction"')]
 function MyFunction( [kbmMW_Rest('value:body')] ABody:string):string;
```

and in MyFunction's implementation:

```
var
 qv:TkbmMWHTTPQueryValues; // Found in kbmMWHTTPUtils.pas
begin
 qv:=TkbmMWHTTPQueryValues.Create;
 try
  qv.AsString:=ABody; // If the body is URL encoded you can use qv.AsEncodedString:=ABody
  value1:=qv.ValueByName['value1'];
  value2:=qv.ValueByName['value2'];
 finally
  qv.Free;
 end;
end;
```

There are multiple ways:

If the data is "known" you can define a class to receive the data. Eg.

```
[kbmMW_Root('data',[mwrfIncludeOnlyTagged])]
 TData = class
 private
   FValue1:string;
   FValue2:string;
 public
   [kbmMW_Attribute('name1')]
   property Value1:string read FValue1 write FValue1;

   [kbmMW_Attribute('name2')]
   property Value2:string read FValue2 write FValue2;
 end;
...
   [kbmMW_Rest('method:post, path: "myfunction"')]
   function MyFunction([kbmMW_Rest('value: "body"')]const AData:TData):integer;
```

An instance of AData will then be provided to the function MyFunction and automatically freed upon exit of MyFunction.

If the data is "unknown" you can use a XML or JSON streamer. Eg.

```
var
 on:TkbmMWONCustomObject;
begin
 json:=TkbmMWJSONStreamer.Create;
 try
  on:=json.LoadFromUTF16String(ABody);
  if on.IsObject then
 begin
   value1:=TkbmMWONObject(on).AsString['name1'];
   value2:=TkbmMWONObject(on).AsString['name2'];
 end;
 finally
  json.Free;
 end;
end;
```

It is somewhat more complex to handle, but kbmMW contains a TkbmMWHTTPMultiParts class that can be used to decipher multipart data containing multipart boundaries.This time we first need to figure out if its actually a multipart body. Then we need to figure out whats the boundary identification between each part, separately.

```
[kbmMW_Rest('method:post, path: "myfunction"')] function MyFunction:string;

function TMyService.MyFunction:string;
var  i:integer;  mp:TkbmMWHTTPMultiParts; p:TkbmMWHTTPMultiPart;
 f:TkbmMWHTTPMimeHeaderValueFields; helper:TkbmMWHTTPTransportStreamHelper;
 sFileName, sBoundary:string; fs:TFileStream;
begin
 Result:='No data found';

 // First pick out content-type header field.
 helper:=TkbmMWHTTPTransportStreamHelper(RequestTransportStream.Helper);
 f:=helper.Header.ValueFields['Content-Type'];
 if f=nil then
  exit;

 // Check if boundary given. If so parse multiparts.
 sBoundary:=f.ValueByName['boundary'];
 if sBoundary<>'' then
 begin
  mp:=TkbmMWHTTPMultiParts.Create(RequestStream,sBoundary);
  try
   // Loop thru parts.
   for i:=0 to mp.Count-1 do
   begin
    // Check if file upload.
    p:=mp.Parts[i];
    f:=p.Headers.ValueFields['Content-Disposition'];
    sFileName:=f.ValueByName['filename'];
    if sFileName<>'' then
    begin
     ForceDirectories('.\receivedfiles');
     sFileName:='.\receivedfiles\'+sFileName;
     DeleteFile(sFileName);
     fs:=TFileStream.Create(sFileName,fmCreate+fmOpenWrite);
     try
      p.SaveToStream(fs);
     finally
      fs.Free;
     end;
     Result:='Thank you for the file '+sFileName;
    end;
   end;
  finally
   mp.Free;
  end;
 end;
end;
```

Prologue

There are many more features in the kbmMW REST smart service. Some of them are explained in other blog posts in this "REST easy" serie, You may also want to check out the documentation section containing more than 600 pages of d cumentation by visiting our site at http://www.components4developers.com, and look for the kbmMW documentations section.

If you like this, please share the word about kbmMW wherever you can and feel free to link, like, share and copy the posts of this blog to where you find they could be useful.

Oh… and whats about that featured image? It's the mail boxes in Denmark… In danish we call that "Post kasser". Due to the internet, fewer and fewer of these are to be found around the landscape. The last 10 years around 85% of all post offices has disappeared. Instead local grocery shops handle snail mail with various success.

BY DETLEF OVERBEEK

AI, AI , AI, AI, AI...
it sounds like the emergency siren of a police car.
Is it that frightening?
In this article I want to explain a few things about it.

First of all we need to understand the basics of it, but let's start with the history.

WHAT DOES AI ACTUALLY MEAN?
Is it some kind of robot language created by the famous writer Isaac Asimov? Asimov foresaw a lot of our future problems. Not only did he create laws for robotics we should use, but he also was already understanding and stating a very big problem: Discrimination.

I remember as very young person the small Science Fiction Novels he wrote and I bought them for about 75 cents. Of course I read them in bed under the sheets. Unfortunately I did not keep them in a safe place. They got lost. But ever since I have bought all of his books, even a special edition of his first Foundation Trilogies.
Discrimination because these Human like figures he described were an absolute copy of a human,
but were thought to be humanoids: Robots. They had blue coloured skin. This made me think of the many black or coloured people we have, the way they have been oppressed and abused.
A philosophic start to begin thinking of the mechanical robots we want to make, or are creating.
The future has come very close: What rights will we give them? Wil they become our successor?
Aren't they actually just computers based on AI?

Can Artificial intelligence be artificial at all? Intelligence should be proven.

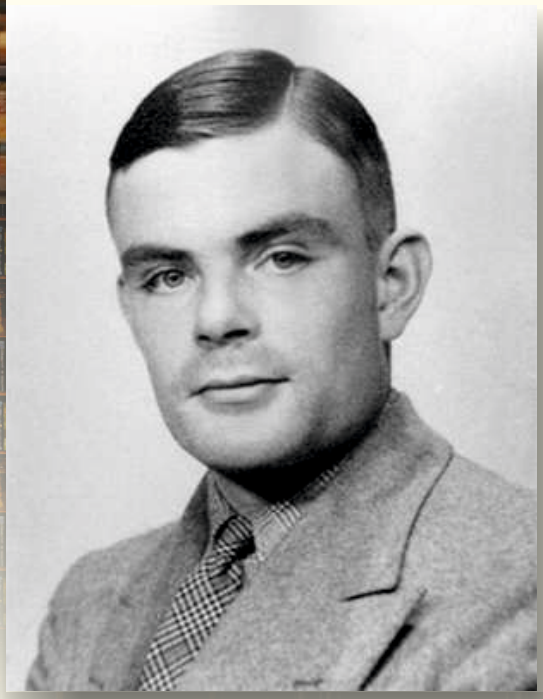
Some of these answers can be explained.
To begin with: AI-Artificial Intelligence,
isn't that simply the wrong term?
Can Artificial intelligence be artificial at all?
Intelligence should be proven.

Often people describe a pause in a computer's operation as: **He's thinking.** I always correct that and explain that a computer can calculate but it can't think.
And that explains a lot: it is NOT intelligent.
It uses algorithms and that is a set of step-by-step instructions so narrowed down that even a literal minded machine like a computer can follow those instructions.
We program how it thinks, but it does not necessarily end up thinking as we do.

So lets start by its practical origins, just as far back as a calculator. And its quite some time ago to begin with. During the end of the first world war it all started already by the German Enigma machine and its counterpart.

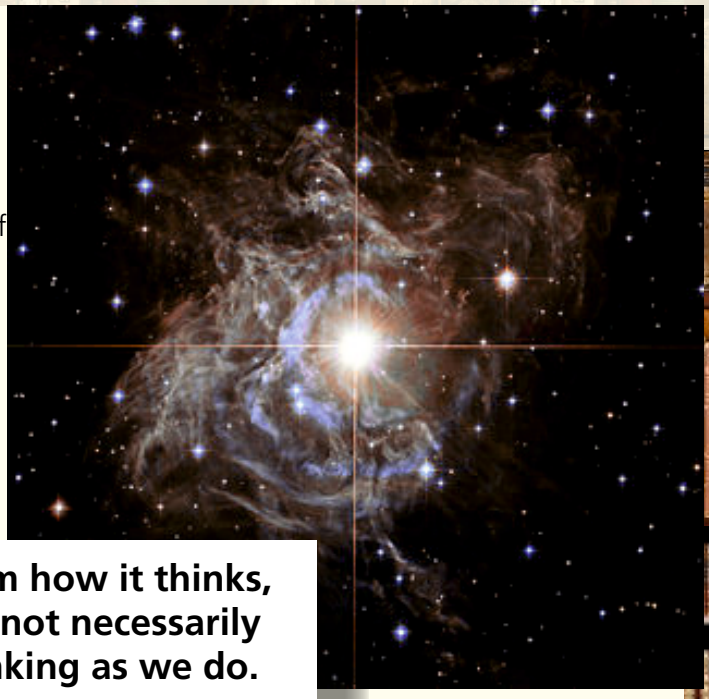


Enigma was invented by the German engineer Arthur Scherbius at the end of World War 1.
The British Mathematician and logician
Alan Turing (who was mistreated because of a very narrow minded people that could not understand his intelligence and because of: DISCRIMINATION) was one of the first to start creating AI.



The earliest successful AI program was written in 1951 by **Christopher Strachey**, later director of the Programming Research Group at the University of Oxford.
Strachey's checkers (draughts) program ran on the Ferranti Mark I computer at the University of Manchester, England.
By the summer of 1952 this program could play a complete game of checkers at a reasonable speed.

So its not so new after all.
And it still will really take some time before we come to the real step: Intelligence. One should understand that intelligence is not only the capability of following an enormous complicated algorithm. It also needs to be something which we think nowadays is only for living beings and especially humans. However the newly constructed computers are because of the size of their CPU's capable of learning.
They nowadays can use the **GPU** (Graphics Processing Unit), and these are available - because of the new Graphical Cards- in an enormous number on any computer.
You can use them for computational services as well, many people don't realize...



We program how it thinks, but it does not necessarily end up thinking as we do.

In AI, recognizing a pattern means fitting an equation to data.
A very interesting basic thought about this is an American Lady that showed us for the first time how really "big" the universe is. Here is a very short story a bout her.

Henrietta Leavitt's prediction rule.
Henrietta Swan Leavitt (1868 - 1921) was an **American lady-astronomer.**
She worked at the Harvard College Observatory as a **"computer",** tasked with examining photographic plates in order to measure and catalogue the brightness of stars.
This work led her to discover the relation between the luminosity and the period of Cepheid variables.

WIKIPEDIA

A Cepheid variable is a type of star that pulsates radially, varying in both diameter and temperature and producing changes in brightness with a well-defined stable period and amplitude. Leavitt's discovery provided astronomers with the first "standard candle", with which to measure the distance to far away galaxies. After her death, ***Edwin Hubble*** *used* ***Leavitt's luminosity–period relation****, together with the galactic spectral shifts first measured by* ***Vesto Slipher*** *at Lowell Observatory, in order to establish that the universe is expanding.*

These complicated patterns should be described with complicated equations. To use them you will need a lot of computational power as well, you will need a lot of data to be able to interpret them reliably.

Only a very short time we have had the capability and technology of doing this at reasonable cost.

A large breakthrough in AI is the use of so-called **Neural Networks.** Not those of your brain of course but because its name-like shape.

A Neural Network consists of very large and complicated equations that are capable of using all the data and create very complicated patterns in these data and recognize them by making mappings from inputs to outputs.

We cannot use these networks because they do what our human brain does, but because it works wonderfully well in predicting tasks, like recognizing of images, learning languages and even videos etc.

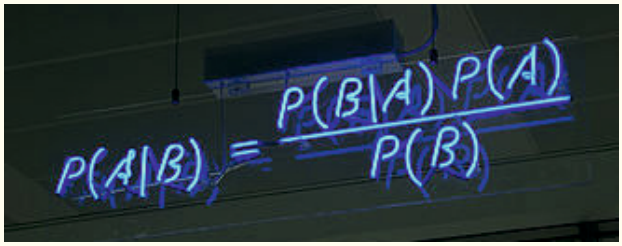
Fitting prediction rules to data you could state the following:

1. In AI a pattern is a prediction rule that maps an input to an output
2. Learning a pattern means fitting a good prediction rule to a dataset.

A very fine rule to start with and learn from is the so called **BAYES RULE.**
In probability theory and statistics, **(Bayes' rule)** describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
For example, if cancer is related to age, then, using Bayes' theorem, a person's age can be used to more accurately assess the probability that they have cancer, compared to the assessment of the probability of cancer made without knowledge of the person's age.

One of the many applications of Bayes' theorem is Bayesian inference, a particular approach to statistical inference.

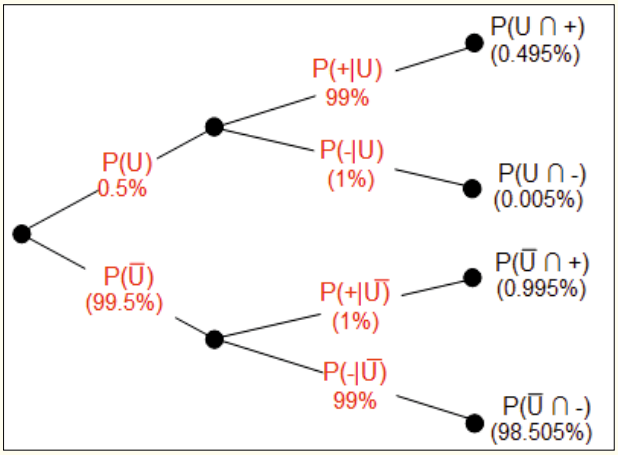
When applied, the probabilities involved in Bayes' theorem may have different probability interpretations. With the Bayesian probability interpretation the theorem expresses how a subjective degree of belief should rationally change to account for availability of related evidence. Bayesian inference is fundamental to Bayesian statistics.



By Gnathan87 - Own work

WIKIPEDIA

*Bayes' theorem is named after Reverend Thomas Bayes; 1701–1761), who first provided an equation that allows new evidence to update beliefs in his "**An Essay towards solving a Problem in the Doctrine of Chances**"(1763).*
*It was further developed by **Pierre-Simon Laplace**, who first published the modern formulation in his 1812 "Théorie analytique des probabilités". Sir Harold Jeffreys put Bayes' algorithm and Laplace's formulation on an axiomatic basis. Jeffreys wrote that Bayes' theorem "is to the theory of probability what the Pythagorean theorem is to geometry". If you might like to understand this rule Have a look at this video:*
https://www.youtube.com/watch?v=E2pOJwSwWDk



And here comes another trick because of its results it has also a highly addictive pleasure. In some way you could predict things and therefore it will give you the opportunity to make the right decisions. So actually is Neural Networking learning a way of predicting.
The question remains if one can trust decision-making of a machine.
But It can be a quite good helper. It will hardly ever be more intelligent then we are.

But it will know the facts better and therefore be able to substantiate the subject and thus make better decisions, and these decisions will not be in any way emotional.

Which is of course not always what we want but it will be up to us if we allow the outcome or not to be used.
Strategic decision-making is often very much instinctive and unconscious, but also can be shaped by deliberate reflection and make an attempt at empathy.

It might force you to overthink a very threatening solution - temperature of the Oceans may become so high that the Ocean might rise a few meters and because of this convincing evidence the president might decide to unite with the Paris Agreement again, even though he will of course say it's not caused by people.
This illustrates in a certain way how much we need this technology.
I personally have a future idea of using AI as a very good extra tool. It will take longer than the next few years to develop the capability of building robots that can do all that humans can do. Building intelligent robots is a very different subject.
Once we would really want that, we will need a lot of time and futuristic techniques that aren't even available yet.

Ever thought of a robot sitting with a shrink?

Mother Nature, so to speak, found out that creating intelligent species is a very different piece of cake: if you simply would like your machine to stay "alive" it (the machine) would need to understand being empathic.
It would need to be able to feel (skin) and touch, and would also have all kinds of urges like having psychic and social feelings.
Ever thought of a robot sitting with a shrink?

So I am very confident of the future, this will help us and not threaten us like the fake news the press wants to make us believe.
There is a serious research being done where the outcome was that **people that are optimistic live longer**. (Done with AI).

Of course we will be more and more helped by these AI-techniques and they will even be implemented into our brain if that is helpful.
Why not?
You might think that Coca Cola or Nike will inject a virus into your brain that that tells you that you will have to by Nike's all the time and even zip Coke out of them?
Well than you have little confidence in what mankind will and can do.
It is as it ever has been: it's a two double-edged sword. You can get hurt.

If we want to keep alive in this massive number of peoples on this earth we better get all the help we can and not be afraid of AI.
Of course there are sharp edges and future problems we have to solve; but they're of a very different kind. If we want to use AI as much as it possible can - and its promises for the future - we need to make better CPU's: smaller and smarter (Nvidia is already doing so).

There is a problem with using AI: the energy it absorbs. Again this sword has two edges: AI will raise sea levels because of the temperature effects it will cause.
It uses because of its ever growing energy hunger - now already 5 % of the total energy on earth, and it will become 20% by 2025 (prediction made by AI).

But this is what keeps the swords edge sharp:
AI will help us to develop GPUs that will be able to use less energy, eventually as it already happens make systems that create hydrogen from sunlight in plants and paint sprayed on walls which contains sun cells that provide electricity.

So especially we who are developers should make use of this very beautiful opportunity to create more and better programs and find out what companies already have patterns for us freely available like Google - they are frighteningly big, but we can't do without them, as well as Amazon and others whose names I do not want to use here. But there is one company for Pascal components that helps us already: Bruno Fierens' TMS software.
And that is how AI-Pascal comes to us as programmers, he made the APIs available for us.
For Pascal: Delphi and Lazarus.
We need more of this.
Lots more. Lets give it a try.

MORE FROM TMS
https://www.tmssoftware.com/site/blog.asp?post=487

tmssoftware.com

Visiting the TMS lab day 2: Adding artificial intelligence to TMS WEB Core apps



There is no doubt about it that artificial intelligence is a hot topic. Wondering how this cool technology could be used by Delphi developers creating web applications, we researched how we could leverage existing technology. In the area of artificial intelligence, the Tensorflow.js library is one of the most popular and powerful libraries for use in web clients. So, we embarked on investigating how we could enable using TensorFlow.js easily from our beloved Pascal language in a TMS WEB Core web client application.

starter expert



Introduction

PDF has been the world standard for electronic documents since 2008. The specifications1 for pdf is now an ISO standard, so documented worldwide.
For years I use MS Word to create pdf files from a Word document and for a while I had the desire to create a pdf document from Delphi but without the intervention of a third program.
Where do you start? Since GetIt Package Manager is part of Delphi (I currently use Delphi Tokyo 10.2.3) that is my first search location.
I came across the PowerPdf package.

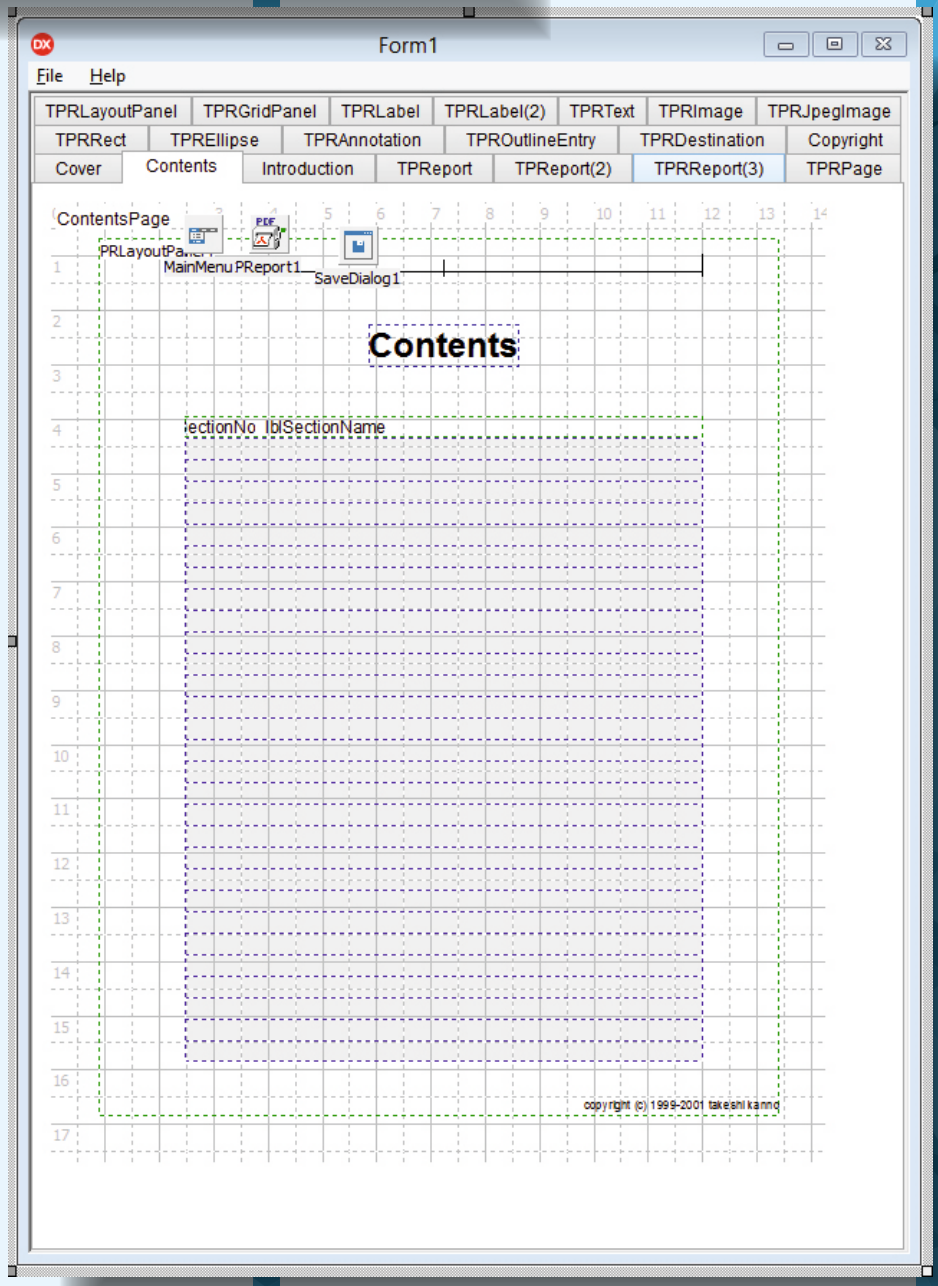
The discovery

Installing a new package is always a kind of exciting discovery, it can be a disappointment whether you have found the right tool. After installation you hope for good documentation or even better, many examples. That turns out to be disappointing. Then just google on PowerPdf.

That's where I end up on a SourceForge1 website. And then it appears that the software has not changed in the last 4 years. That makes you think. But because the code is open source, I'm still curious, sometimes you find some pearls. Further searching on the internet, I came across more examples on a sourceForge / Lazarus2 website A folder higher is the lazarus version of PowerPdf. I tried to install this but there are too many references to the Lazarus libraries.

Analyzing the code

What is striking in the code is the frequent use of the "with" statement.
In the ancient day's it used to be necessary to keep the code compact, with the current self-documenting code this is outdated and unclear. Well I just spend an hour to make the code readable.



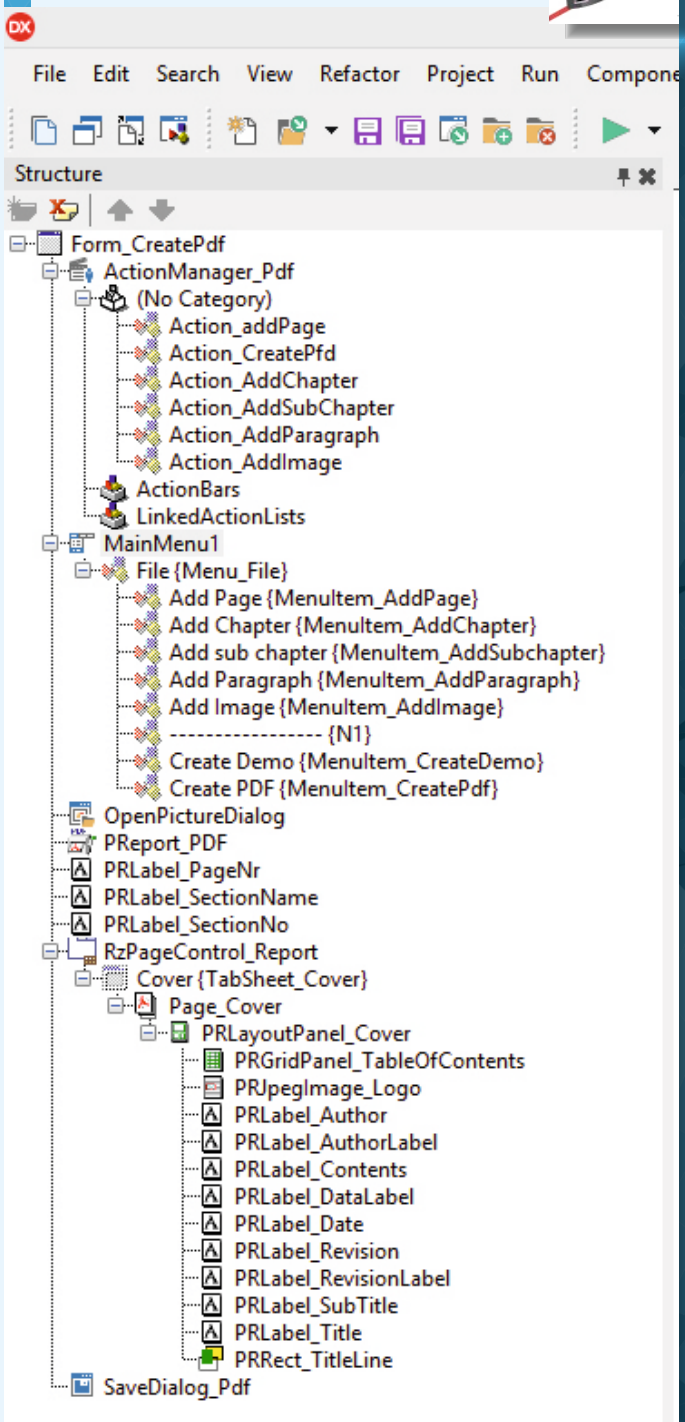
EXAMPLES
An interesting example is **MakeDoc** program.
The program contains only one form, but this form contains the entire documentation of **PowerPdf.**
If you install and run the program, you can create the documentation as a pdf file including tables of contents and an outline via the **File | Create PDF** menu.

On the form is a **`TPageControl`** with per page a tab sheet with the pages of the documentation.

So you can create a pdf in Delphi in design time.
But I am more interested in creating a pdf document in run time. However, the concept appeals to me.
You **can dynamically create several pages.** You have a visual check and if you like it you create the document.

I am going to follow this example and make my own form with which I can make a PDF document.
I call this form **`Form_CreatePdf`** and provide it with properties and methods to create a pdf document.

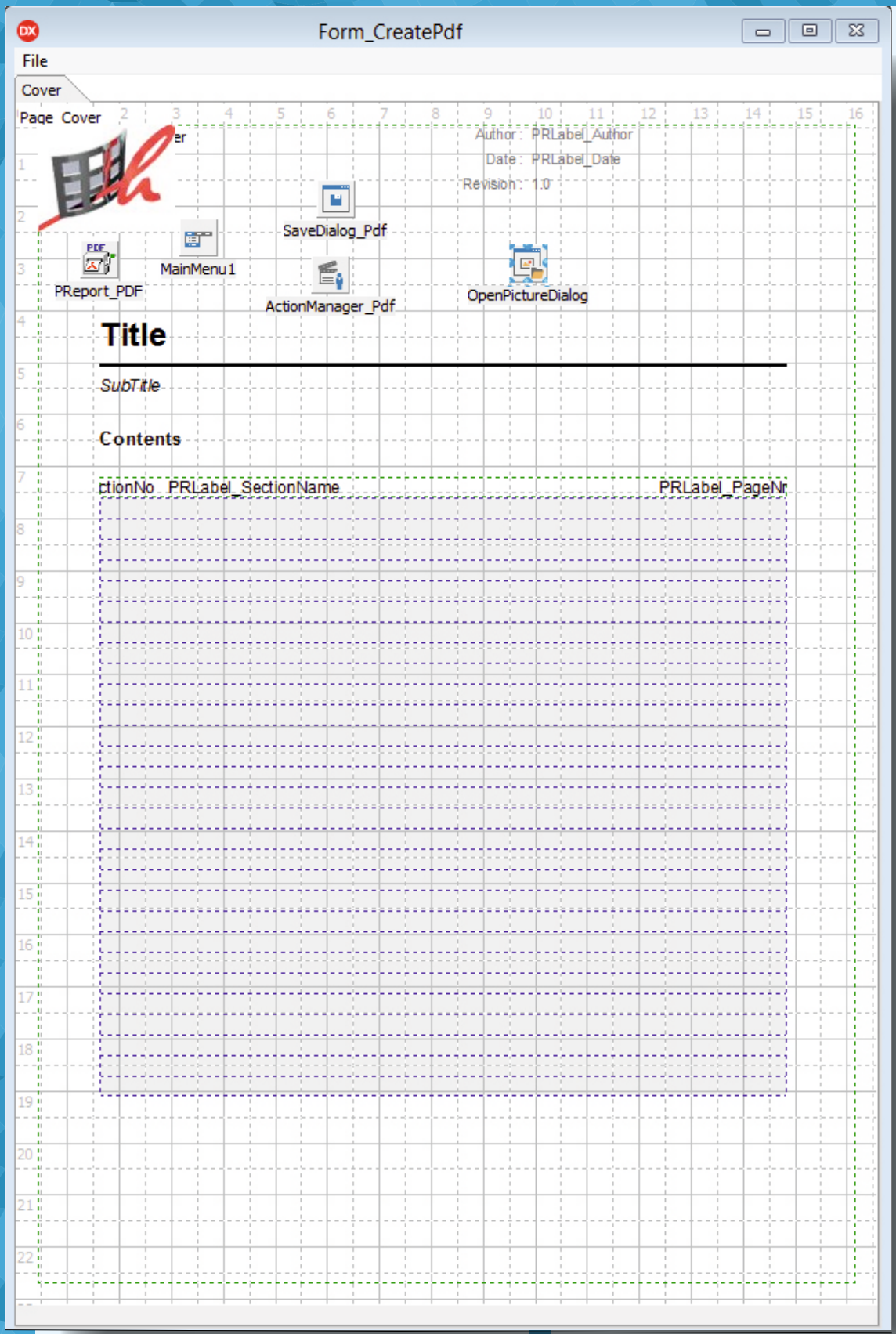
FORM_CREATEPDF
On this form I put a **`RzPagecontrol`** with one tab sheet for the cover of the document. I copied the components of this tab sheet from the MakeDoc program.
In the structure on the right you can see the structure of the form.
All components with the prefix PR are PowerPdf components
To provide the document with document data, I have defined the following properties:




```
Property Title : String;
Property SubTitle : String;
Property Author : String;
Property CreationDate : TDate;
Property Revision : String;
```

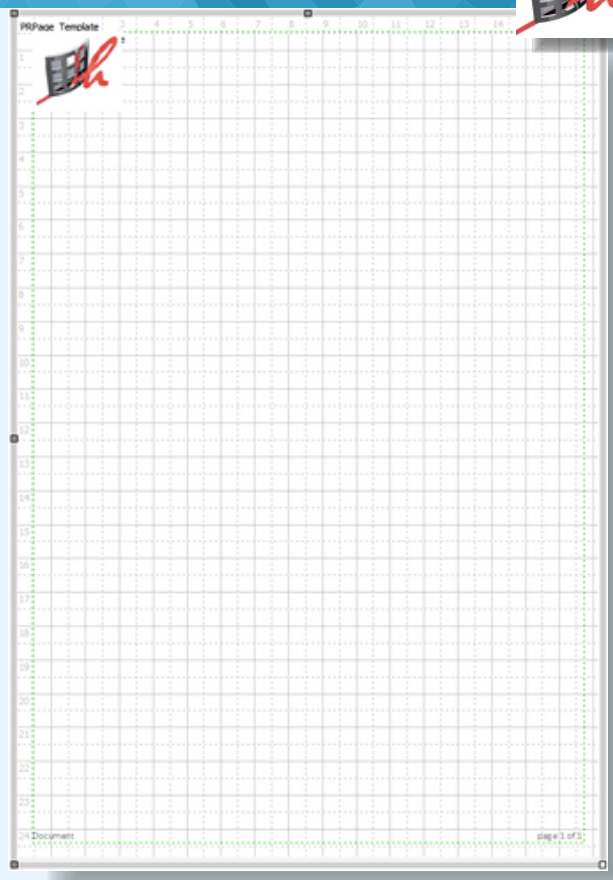
To build the document, the following methods are required:

```
procedure AddPage(Name: String);

Procedure AddChapter(Level : Integer; Title : String);

Procedure AddLine(Line : String; Level : Integer);

Procedure AddParagraph(Lines : WideString);

procedure AddImage(ImageFileName, Caption : String; pHeight, pWidth : Integer; Align : TAlign);
```


Form_CreatePdf
File
Cover
Page Cover
Author : PRLabel_Author
Date : PRLabel_Date
Revision : 1.0
SaveDialog_Pdf
MainMenu1
PReport_PDF
ActionManager_Pdf
OpenPictureDialog
Title
SubTitle
Contents
ctionNo PRLabel_SectionName
PRLabel_PageNr

PAGE TEMPLATE
It is useful if you can use a template for every page where a number of things have already been defined, such as a header with a logo and a footer with page numbering
Here a frame comes in handy. In this Frame you put a PDF page **(TPRPage)** with a layout panel **(TPRLayoutPanel)** on top of it.
At the top you can put a company logo **(TPRimage)** and bottom labels **(TPRlabel)** with page numbering and other stuff that you want to put in the footer.
For the logo, set the **TPRimage** property **SharedImage** to true, this will prevent the logo from being included in the document again and again.

If you add a new page, create a new tab sheet and on that tab sheet you put an instance of the frame.
You define this in an AddPage function in the main form.
You also want to add chapters, sub chapters and paragraphs to the page. I solved this by providing the frame with the **AddChapter** and **AddParagraph** functions.



PAGE FORMAT.
The page size in the **MakeDoc** example is an American format. In the PdfDoc.pas unit you can adjust this, assuming a DPI of 72, to 595 x 842 (w x h) pixels for A4
Here is the code for adding a page.

```
Procedure TForm_CreatePdf.AddPage(Name : String);
  var NewFrame  : TFrame_Template;  NewTabSheet : TRzTabsheet;
begin
  NewTabSheet := TRzTabsheet.Create(RzPageControl_Report);
  NewTabSheet.PageControl := RzPageControl_Report;

  FPageNr := RzPageControl_Report.PageCount - 1;

  NewTabsheet.Tag   := FPageNr;
  NewTabsheet.Caption := 'Page_' + FPageNr.ToString;

  // Create new pdf page from template frame.
  NewFrame := TFrame_Template.Create(NewTabsheet);
  NewFrame.Parent  := NewTabsheet;
  NewFrame.Top     := 0;
  NewFrame.Left    := 0;
  NewFrame.PageNr  := NewTabsheet.Tag;
  NewFrame.Name    := 'Page_' + NewFrame.PageNr.ToString;
  NewFrame.PageMargins    := FPageMargins;
  NewFrame.TextMarginsMm := FTextMarginsMm;
  NewFrame.PagePixWidth  := FPdfPage.PixWidth;
  NewFrame.PagePixHeight := FPdfPage.PixHeight;

  NewFrame.SetPageProperties;
  NewFrame.OnBeforePrintEvent := PRLayoutPanelBeforePrint;

  NrOfPages := RzPageControl_Report.PageCount;

  RzPageControl_Report.ActivePageIndex := RzPageControl_Report.PageCount - 1;

  Action_AddChapter.Enabled   := True;
  Action_AddImage.Enabled     := True;
end;
```

```
procedure TForm_CreatePdf.RzPageControl_ReportChange(Sender: TObject);
begin
 FCurrentPage := Get_CurrentPage(RzPageControl_Report.ActivePageIndex);
end;
```

In order to be able to keep track of which page items have to be added to, the frame on which work must be stored is stored in a variable called FCurrentPage. This is assigned when changing the pageControl.

The Get_CurrentPage function searches the pages of the page control by the ActivePageIndex and copies the found frame to FcurrentPage.
After you have added a page you can provide it with chapters with a title and content. Because I want to create a table of contents on the cover page and an outline in the PDF document, I use a TContentElement. The numbering of the chapters and paragraphs is maintained in a record TChapterNumber. Were Level 1 is a chapter, Level 2 the first paragraph level etc.

```
procedure TForm_CreatePdf.AddChapter(Level: Integer; Title: String);
var
 ContentsElement : TContentsElement;
 ChapterNr : TChapterNumber;
begin

 if FContentsList.Count > 0 then begin
  ChapterNr := TContentsElement(FContentsList.Items[FContentsList.Count -
1]).ChapterNr;

 end else begin
  ChapterNr.Level1 := 0;
  ChapterNr.Level2 := 0;
  ChapterNr.Level3 := 0;
  ChapterNr.Level4 := 0;
 end;

 ContentsElement := TContentsElement.Create;

 ContentsElement.ChapterNr := ChapterNr;
 ContentsElement.AddChapter(level, Title);
 ContentsElement.PageNr   := FPageNr;

 FContentsList.Add(ContentsElement);

 FCurrentPage.AddChapter(Level, Title);

 Action_AddSubChapter.Enabled := True;
 Action_AddParagraph.Enabled  := True;
end;
```

The use of different fonts
In a document you also want to be able to use different (head) styles. For this I use a TList with fonts FFontList in which you can define the different styles.
A bit similar to the heading styles in a html website.
In this example I have defined 5 different fonts:
See next page for the code ->

```
Procedure TForm_CreatePdf.CreateFFontList;
var
 MyFont : TFont;

begin
// Title Font
 MyFont := TFont.Create;

 MyFont.Name  := FTitleFontName;
 MyFont.Style := [];
 MyFont.Size  := 24;
 MyFont.Color := FTitleFontColor;

 FFontList.Add(MyFont);

// Chapter Font
 MyFont := TFont.Create;

 MyFont.Name  := FChapterFontName;
 MyFont.Style := [fsBold];
 MyFont.Size  := 14;
 MyFont.Color := FTitleFontColor;

 FFontList.Add(MyFont);

// SubChapter Font
 MyFont       := TFont.Create;

 MyFont.Name  := FChapterFontName;
 MyFont.Style := [fsBold];
 MyFont.Size  := 12;
 MyFont.Color := FTitleFontColor;

 FFontList.Add(MyFont);

// SubSubChapter MyFont
 MyFont := TFont.Create;

 MyFont.Name  := FChapterFontName;
 MyFont.Style := [fsBold];
 MyFont.Size  := 11;
 MyFont.Color := FTitleFontColor;

 FFontList.Add(MyFont);

// plain Text MyFont
 MyFont := TFont.Create;

 MyFont.Name  := FTextFontName;
 MyFont.Style := [];
 MyFont.Size  := 10;
 MyFont.Color := FTextFontColor;

 FFontList.Add(MyFont);

// Small font
 MyFont       := TFont.Create;

 MyFont.Name  := FTextFontName;
 MyFont.Style := [];
 MyFont.Size  := 8;
 MyFont.Color := FTextFontColor;

 FFontList.Add(MyFont);

end;
```

In the **AddChapter function** you indicate by means of the index which style to use. In the method of the frame **(FcurrentPage.AddChapter)** the chapter is added

```
Procedure TFrame_Template.AddChapter(Level: Integer; Title: String);
begin
 AddLabel(Level, Title);
end;

Procedure TFrame_Template.AddLabel(Level : Integer; Title : String);
var
 PdfChapter : TPRLabel;
begin
 PdfChapter := TPRLabel.Create(PRLayoutPanel_Page);

 PdfChapter.Parent := PRLayoutPanel_Page;

 if Level < 3 then
  PdfChapter.Tag := Level
 else
  PdfChapter.Tag := 0;

 PdfChapter.Left := FTextPosition.X;

 FTextPosition.Y := FTextPosition.Y + (TFont(FFontList[level]).Size);

 PdfChapter.Top := FTextPosition.Y;

 PdfChapter.FontName  := TFont(FFontList[level]).Name;
 PdfChapter.FontBold  := fsBold in TFont(FFontList[level]).Style;
 PdfChapter.FontItalic := fsItalic in TFont(FFontList[level]).Style;
 PdfChapter.FontColor := TFont(FFontList[level]).Color;
 PdfChapter.FontSize  := TFont(FFontList[level]).Size;
 PdfChapter.Height    := TFont(FFontList[level]).Size;

 PdfChapter.Caption := Title;

 FTextPosition.Y := FTextPosition.Y + PdfChapter.Height +
         (TFont(FFontList[level]).Size div 4);
end;
```

In a **TPoint** variable **FTextPosition,** the x and y position of the next item on the page is tracked.
Then you can take into account margins left, right and top. You can also determine when a page is full and then create a new page.
When a new page is made, the y position is made equal to the top margin of the page, the x position equals the ledt margin of the page.

ADDING A PARAGRAPH.
To add text to the page a TPrText box is added to the page. To determine the height of the PrText I use a TMemo, made invisible, where I put the paragraph lines. The height of the box is then determined from the number of lines in the memo multiplied with a constant found by emprirical determination. -> Next Page.

```
procedure TFrame_Template.AddParagraph(Lines: WideString);
var
 pdfParagraph : TPrText;
begin
 pdfParagraph := TPrText.Create(PRLayoutPanel_Page);
 pdfParagraph.Parent := PRLayoutPanel_Page;

 pdfParagraph.Tag          := 0;
 pdfParagraph.Left         := FTextPosition.X;
 pdfParagraph.Top          := FTextPosition.Y;
 pdfParagraph.Width        := FPageWidth;
 pdfParagraph.WordWrap     := True;

 pdfParagraph.FontName     := TFont(FFontList[4]).Name;
 pdfParagraph.FontColor    := TFont(FFontList[4]).Color;
 pdfParagraph.FontBold     := fsBold   In TFont(FFontList[4]).Style;
 pdfParagraph.FontItalic   := fsItalic In TFont(FFontList[4]).Style;

 Memo_Paragraph.Font       := TFont(FFontList[4]);
 Memo_Paragraph.Left       := pdfParagraph.Left;
 Memo_Paragraph.Width      := pdfParagraph.Width;

 Memo_Paragraph.Lines.Text:= Lines;
 pdfParagraph.Text         := Lines;
 pdfParagraph.Height       := Round(Memo_Paragraph.Lines.Count * 11.25);

 FTextPosition.Y := pdfParagraph.Top + pdfParagraph.Height + g_lineMargin;

end;
```

TABLES

No document is complete if you can not create a table. Now PowerPdf has a component TPRGridPanel for that purpose but it is only filled during the creation of the document. And in this component in design time you must define a label per column in terms of position and width and alignment.
I prefer to build a table with separate lines with tabs in run time. For this I use a dynamic array in the template frame that can be filled with TTabDefinitions

```
TTabAlign = (taLeft, taRight, taCenter, taNumeric);
TTabDefinition = record
 Position    : single; //mm
 TabAlign    : TTabAlign;
 Pixels      : Integer;
 FillCharacter : String;
end;
```

The tabs can then be set with the following procedure

```
procedure TFrame_Template.SetTab(TabDef: TTabDefinition; ClearAll: Boolean);
var
 TabListLen : Integer;
begin
 if ClearAll then
  ClearTabList;
 TabListLen := Length(FTabList);
 SetLength(FTabList, TabListLen + 1);
 FTabList[TabListLen] := TabDef;
 FTabList[TabListLen].pixels := mmToDots(FTabList[TabListLen].Position);
end;
```

In this example a PDF is created with a DPI of 72. There is a **mmToDots** *conversion function*

One row of the table can be added as a string with a tab (#9) as afield separator..
With the level you indicate which font to use.
You can then cut a line into labels and use the PdfLabel function **GetTextWidth** to position the labels according to the TabAlign. In the program this could be like this:

```
TabDef.Position := 0.5;
TabDef.TabAlign := taLeft;
FCurrentPage.SetTab(TabDef, True);// Clear the tabs

TabDef.Position := 5;
TabDef.TabAlign := taRight;
FCurrentPage.SetTab(TabDef, False);

TabDef.Position := 10;
TabDef.TabAlign := taCenter;
FCurrentPage.SetTab(TabDef, False);

TabDef.Position := 15;
TabDef.TabAlign := taNumeric;
FCurrentPage.SetTab(TabDef, False);

MyLine := 'Item_Left' + #9 + 'Item_Right' + #9 + 'Item_Center' + #9 + '10.50';
FCurrentPage.AddLine(MyLine, 4);
```

For the table of contents
For the table of contents I do use the **TPRGridPanel** because it always has the same columns

Rest of the program is placed on the next page ->

Inserting an Image

The following code shows the insertion of an image.

From the main form:

```
Procedure TForm_CreatePdf.AddImage(ImageFileName, Caption : String;
    pHeight, pWidth : Integer; Align : TAlign);
var
 ImageElement : TImageElement;
begin
 ImageElement.ImageName := ImageFileName;
 ImageElement.Width   := pWidth;
 ImageElement.Height  := pHeight;
 ImageElement.Align   := alBottom;
 ImageElement.Stretch  := True;
 ImageElement.Printable := True;
 ImageElement.Caption  := Caption;

 FCurrentPage.AddImage(ImageElement);
end;
```

```
procedure TFrame_Template.AddLine(Line: String; Level : Integer);
var
 LineList : TStringDynArray;  i, iTab , TabListLength : Integer;  PdfLabel : TPRLabel;
 LabelWidth, PosDot : Integer;  beforeDot : String;
begin
 if pos(#13, Line) > 0 then Exit;
 if Trim(Line) = '' then begin
  FTextPosition.Y := FTextPosition.Y + TFont(FFontList[level]).Size div 2;
  Exit;
 end;

 LineList := SplitString(Line, #9);
 TabListLength := Length(FTabList);
 if Length(LineList) > 0 then begin
  iTab := 0;

  for i := 0 to Length(LineList) - 1 do begin
   PdfLabel := TPRLabel.Create(PRLayoutPanel_Page);
   PdfLabel.Parent := PRLayoutPanel_Page;
   if iTab < TabListLength then begin
    PdfLabel.FontName   := TFont(FFontList[Level]).Name;
    PdfLabel.FontBold   := fsBold in TFont(FFontList[Level]).Style;
    PdfLabel.FontItalic := fsItalic in TFont(FFontList[Level]).Style;
    PdfLabel.FontColor  := TFont(FFontList[Level]).Color;
    PdfLabel.FontSize   := TFont(FFontList[Level]).Size;
    PdfLabel.Height     := TFont(FFontList[Level]).Size;
    PdfLabel.Caption    := LineList[i];
    LabelWidth          := Round(PdfLabel.GetTextWidth);
    PdfLabel.Width      := LabelWidth;

    case FTabList[iTab].TabAlign of
      taLeft   : PdfLabel.Left := FTextPosition.X + FTabList[iTab].Pixels;
      taRight  : PdfLabel.Left  := FTextPosition.X + FTabList[iTab].Pixels – LabelWidth;
      taCenter : PdfLabel.Left := FTextPosition.X + FTabList[iTab].Pixels – LabelWidth shr 1;
      taNumeric : begin
      PosDot := Pos(FormatSettings.DecimalSeparator, LineList[i]);
      if PosDot > 0 then begin
       beforeDot := Copy(LineList[i], 0, posdot -1);
       PdfLabel.Caption := beforeDot;
       PosDot := Round(PdfLabel.GetTextWidth) + 1;
       PdfLabel.Left  := FTextPosition.X + FTabList[iTab].Pixels - PosDot;
       PdfLabel.Width := Round(PdfLabel.GetTextWidth);
      end;
     end;
    end;

    PdfLabel.Top      := FTextPosition.Y;
    PdfLabel.Caption  := LineList[i];
    inc(iTab);
   end;
  end;
 end else begin     // No Tabs present
  PdfLabel := TPRLabel.Create(PRLayoutPanel_Page);
  PdfLabel.Parent  := PRLayoutPanel_Page;
  PdfLabel.Left    := FTextPosition.X;
  FTextPosition.Y  := FTextPosition.Y + (TFont(FFontList[Level]).Size);
  PdfLabel.Top     := FTextPosition.Y;
  PdfLabel.FontName    := TFont(FFontList[Level]).Name;
  PdfLabel.FontBold    := fsBold  in TFont(FFontList[Level]).Style;
  PdfLabel.FontItalic  := fsItalic in TFont(FFontList[Level]).Style;
  PdfLabel.FontColor   := TFont(FFontList[Level]).Color;
  PdfLabel.FontSize    := TFont(FFontList[Level]).Size;
  PdfLabel.Height      := TFont(FFontList[Level]).Size;
 end;

 FTextPosition.Y := FTextPosition.Y + PdfLabel.Height + (TFont(FFontList[level]).Size div 4);
end;
```

The ImageElement is of type record

```
procedure TFrame_Template.AddImage(ImageElement: TImageElement);
var
 pdfImage : TPRJpegImage;  PdfImageCaption : TPRLabel;

 PdfPicture : TPicture;  DestRect, SrcRect, FitRect : TRect;
begin
 if FileExists(ImageElement.ImageName) then begin
  SrcRect      := Rect(0,0, ImageElement.Width, ImageElement.Height);
  pdfPicture   := TPicture.Create;
  try
   pdfPicture.LoadFromFile(ImageElement.ImageName);
   pdfImage          := TPRJpegImage.Create(PrLayoutPanel_Page);
   pdfImage.Parent   := PRLayoutPanel_Page;
   DestRect := Rect(0,0,PDF_DEFAULT_PAGE_WIDTH, PDF_DEFAULT_PAGE_HEIGHT);
   FitPicture(DestRect, SrcRect, FitRect); //Picture is fit center of the destRect

   case ImageElement.Align of
    alNone  :begin
     pdfImage.Width    := ImageElement.Width;
     pdfImage.Height   := ImageElement.Height;
     pdfImage.Top      := FTextPosition.Y;
     pdfImage.Left     := FTextPosition.X;
    end;
    alTop   :begin
     pdfImage.Width    := FitRect.Right;
     pdfIMage.Height   := FitRect.Bottom - FitRect.Top;
     pdfImage.Top      := FTextPosition.Y;
     pdfImage.Left     := 0;
    end;
    alBottom :begin
     pdfImage.Width    := FitRect.Right;
     pdfIMage.Height   := FitRect.Bottom - FitRect.Top;
     pdfImage.Top      := PDF_DEFAULT_PAGE_HEIGHT - pdfIMage.Height - FTextMarginsDots.Bottom;
     pdfImage.Left     := 0;
    end;
    alLeft  :begin
     pdfImage.Top      := FTextPosition.Y;
     pdfImage.Left     := FTextPosition.X;
    end;
    alRight :begin
     pdfImage.Top      := FTextPosition.Y;
     pdfImage.Left     := PDF_DEFAULT_PAGE_WIDTH - FTextPosition.X - pdfImage.Width;
    end;
    alClient :begin
     pdfImage.Width    := FitRect.Right;
     pdfIMage.Height   := FitRect.Bottom - FitRect.Top;
     pdfImage.Top      := FitRect.Top;
     pdfImage.Left     := FitRect.Left;
    end;
   end;

   pdfImage.Picture    := pdfPicture;
  finally
   pdfPicture.Free;
  end;

  FTextPosition.Y := FTextPosition.Y + pdfIMage.Height + g_LineMargin;

  if ImageElement.Caption <> '' then begin
   FTextPosition.x := pdfImage.Left;
   AddLabel(g_Paragraph, ImageElement.Caption);

   FTextPosition.x := FTextMarginsDots.Left;
  end;
 end;
end;
```

Page administration

The main form also keeps track of the administration of the pages. In the footer of the page is the page number and the number of pages (page 1 of 3). If a page with a frame is added, the old pages must be adjusted to change the number of pages
In the frame a property NrOfPages has been added that adjusts the pages label. This happens in the following procedure:

```
procedure TForm_CreatePdf.SetPagesProperties;
var
 i: Integer;
 ts : TRzTabsheet;
 ii: Integer;
begin

 for i := 0 to RzPageControl_Report.PageCount -1 do begin
  ts := RzPageControl_Report.Pages[i];

  for ii := 0 to ts.ComponentCount - 1 do begin
   if ts.Components[ii] is TFrame_Template then begin
    TFrame_template(ts.Components[ii]).NrOfPages   := FNrOfPages;
    TFrame_template(ts.Components[ii]).DocumentName := FDocumentName;
   end;
  end;
 end;
end;
```

This procedure passes all page frames and updates the number of pages. This procedure is also used to adjust the document name per page

Table of Contents

In the **MakeDoc example** the table of contents is made when creating the pdf document. This involves using a class **`TContentsElement`** that is stored in an **`FContentsList.`** Each chapter is given **`TPrText`** style and a tag.
This tag indicates the chapter level. This is used when creating the table of contents and the links (annotations) in the outline to the correct pages.
This can of course already be set when composing the document at runtime.
I prefer to add every heading / sub heading of a chapter as a **`TPrLabel`**

About the author: **Marcel Horsthuis**
As an aircraft engineer, Marcel worked for 15 years at Fokker Aircraft. In 1997 he started his own software company now called “Horsthouse IT Development”.
From the start of his company he works with Delphi, now 10.2 Tokyo.
He has realized various CD-ROM and CRM / CMS projects. In addition to Delphi, he also has many years of experience with the realization and hosting of websites. Both with his own CMS or with Wordpress.

Marcel works part-time at IBIS-Technologies as a senior software developer. In his free time, Marcel plays the electric bass and does tennis as his sport

CREATING THE PDF DOCUMENT
Finally the creation of the PDF document. First the table of Contents is filled during in the Print function of the cover page. Then a loop over the PageControl handles the print of the pages.

```
procedure TForm_CreatePdf.CreatePdf(FileName: String);
var  i : integer; APage : TPrPage; Page_Template : TFrame_Template;
begin
 if (FileName = '') then begin
    if SaveDialog_Pdf.Execute then begin  PdfDocument := SaveDialog_Pdf.FileName;
    end;
 end else begin
    PdfDocument := FileName;
 end;
 if PdfDocument <> '' then begin
  PReport_PDF.FileName := PdfDocument;
  SetPagesProperties; //Renumber the pages and sets document name
  PReport_PDF.BeginDoc;
  if PReport_PDF.UseOutlines then  FCurrentOutline[0] :=  PReport_PDF.OutlineRoot;
  FPos := 0;
  PRGridPanel_TableOfContents.RowCount := FContentsList.Count;
  PRGridPanel_TableOfContents.Height  := FContentsList.Count * 15;
  while FPos < FContentsList.Count do begin  PReport_Pdf.Print(PRPage_Cover);
  end;
  for i := 1 to RzPageControl_Report.PageCount - 1 do begin
     Page_Template := Get_CurrentPage(i);
     APage := TPRPage(Page_Template.Controls[0]);
     if APage <> nil then begin
        PReport_pdf.Print(Apage);
   end;
  end;
  PReport_PDF.EndDoc;
  FreeContentsList;
  ShowMessage('Document  :  ' + PdfDocument + '  Created.');
 end;
end;
```

Conclusion

This voyage of discovery has given me more insight into the creation of pdf documents. With some adjustments I have made a pdf form that meets my (preliminary) wishes. If necessary, I can also add functionalities in the future.
With this PowerPdf library you create pdf documents according to the old version 1.2, but the documents can be opened in the recent pdf reader.
The final source code contains more than what I have discussed here, but here I show the basics of my method.

More documentation

Googling with the search term "pdf format specification" I came across a link:
"Document management - Portable document format - Part 1: PDF .."
which refers to a pdf document DF32000_2008.pdf with specifications for version 1.7.
This document is still public
The latest version of this document, "Document management - Portable document format - Part 2: PDF 2.0" is now part of the world wide ISO organisation and can be bought via:
https://www.nen.nl/NEN-Shop/Norm/ISO-3200022017-en.htm.
In the link below (3). you will find a mini course on how to create a pdf document with a text editor yourself
Links:

1 https://sourceforge.net/projects/powerpdf/
2 https://sourceforge.net/p/lazarus-ccr/svn/3825/tree/components/powerpdf/Example/
3 https://blog.idrsolutions.com/2013/01/understanding-the-pdf-file-format-overview/

Marcel Horsthuis
www.horsthouse.nl



KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.06.30 BETA RELEASED!

NEW! TKBMMWISAPIRESTSERVERTRANSPORT REST CAPABLE ISAPI SERVER SIDE TRANSPORT.

- **RAD Studio 10.2 Tokyo support including Linux support** (in beta).
- **Huge number of new features** and improvements!
- **New Smart services and clients** for very easy publication of functionality and use from clients and REST aware systems without any boilerplate code.
- **New ORM OPF** (Object Relational Model Object Persistence Framework) to easy storage and retrieval of objects from/to databases.
- **New high quality random functions.**
- **New high quality pronouncable password generators.**
- **New support for YAML, BSON, Messagepack** in addition to JSON and XML.
- **New Object Notation framework which JSON, YAML, BSON and Messagepack** is directly based on, making very easy conversion between these formats and also XML which now also supports the object notation framework.
- **Lots of new object marshalling improvements,** including support for marshalling native Delphi objects to and from YAML, BSON and Messagepack in addition to JSON and XML.
- **New LogFormatter support** making it possible to customize actual logoutput format.
- **CORS support in REST/HTML services.**
- **High performance HTTPSys transport for Windows.**
- Focus on central performance improvements.
- Pre XE2 compilers no longer officially supported.
- Bug fixes
- **Multimonitor** remote desktop V5 (VCL and FMX)
- RAD Studio and Delphi XE2 to 10.2 Tokyo support
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support!
- **Native PHP**, Java, OCX, ANSI C, C#, Apache Flex client support!
- **High performance LZ4 and Jpeg compression**
- **Native high performance** 100% developer defined app server with support for loadbalancing and failover
- **Native improved XSD importer** for generating marshal able Delphi objects from XML schemas.
- **High speed, unified database access (35+ supported database APIs)** with connection pooling, metadata and data caching on all tiers
- **Multi head access** to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- **Full FastCGI hosting support.** Host PHP/Ruby/Perl/Python applications in kbmMW!
- **Native AMQP support** (Advanced Message Queuing Protocol) with AMQP 0.91 client side gateway support and sample.
- **Fully end 2 end secure brandable Remote Desktop** with near REALTIME HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- **Bundled kbmMemTable Professional** which is the fastest and most feature rich in memory table for Embarcadero products.

kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.

- Easily supports large datasets with millions of records
- Easy data streaming support
- Optional to use native SQL engine
- Supports nested transactions and undo
- Native and fast build in M/D, aggregation /grouping, range selection features
- Advanced indexing features for extreme performance

kbmMW SQL functions supported:

LOCALDATETIMETOISO8601, ISO8601TOLOCALDATETIME, DATETIMETOISO8601, ISO8601TODATETIME, UTCDATETIMETOISO8601, ISO8601TOUTCDATETIME, PARSEUTCDATETIME, PARSELOCALDATETIME, FORMATUTCDATETIME, FORMATLOCALDATETIME

Improved support for C++ Builder

COMPONENTS 4 DEVELOPERS

EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER BDS DEVELOPMENT FRAMEWORK FOR WIN 32 / 64, .NET AND LINUX WITH CLIENTS RESIDING ON WIN32 / 64, .NET, LINUX, UNIX MAINFRAMES, MINIS, EMBEDDED DEVICES, SMART PHONES AND TABLETS.