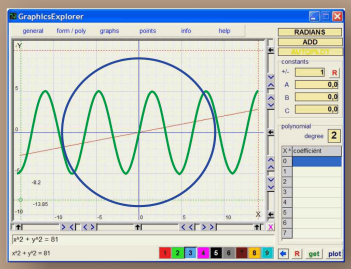
FOR DELPHI, LAZARUS, AND PASCAL RELATED LANGUAGES / ANDROID, IOS, MAC, WINDOWS & LINUX PRINTED, PDF, & ONLINE VIEW

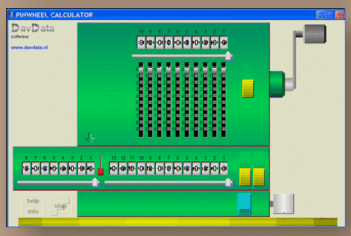
Blaise Pascal

BLAISE PASCAL MAGAZINE 67/68

Tokyo 2.2
By Detlef Overbeek
Video Effects and Animations
By Boian Mitov
CrossVCL and FMXLinux
Two roads to three OS's support
By Vsevolod Leonov
KBMMW Binary Parser
User Defined Functions and kbmMEMSQL
Rest Easy with KBMMW Part 8 Database 3
By Kim Madsen
PAS2JS writing Pascal - creating Javascript,
without knowing the language.
By Detlef Overbeek
PAS2JS Time tracking Web-APP written in Lazarus
By Miguel Bebensee

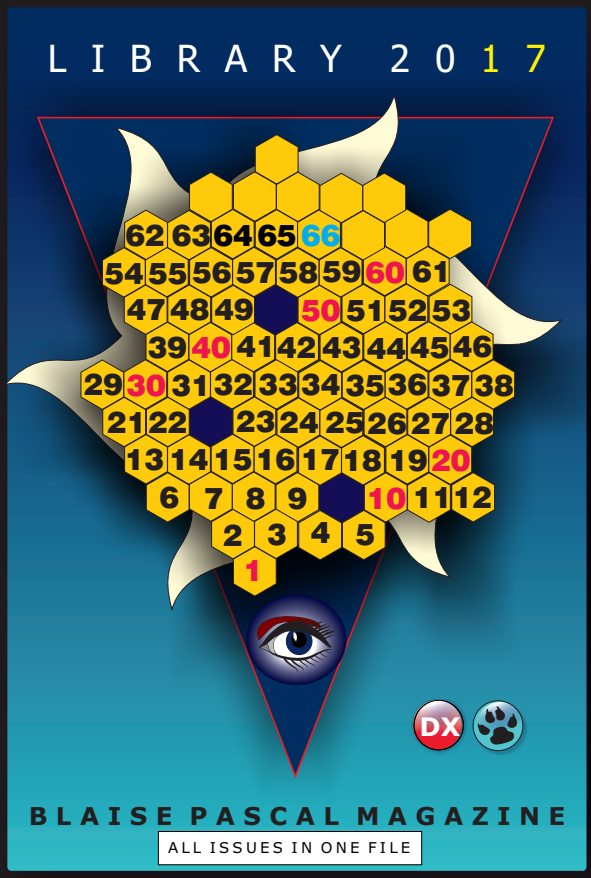
DAVID DIRKSE





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

GRAPHICS COMPUTER MATH & GAMES IN PASCAL

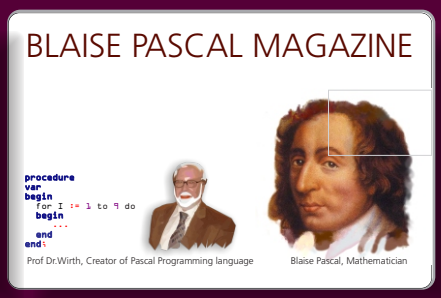


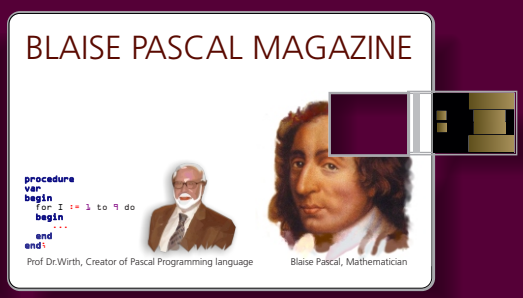
POCKET EDITION

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

CREDITCARD LIBRARY STICK 16 GB

All issues on the USB stick complete searchable 3600 pages - fully indexed





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.blaisepascal.eu/subscribers/UK/UK_CD_DVD_USB_Department.html

BLAISE PASCAL MAGAZINE 67/68

DELPHI, LAZARUS, SMARTMOBILESTUDIO,
AND PASCAL RELATED LANGUAGES
FOR ANDROID, IOS, MAC, WINDOWS & LINUX

CONTENTS

ARTICLES

Cover page: Big, beautiful spiral galaxy NGC 1055 is a dominant member of a small galaxy group a mere 60 million light-years away toward the aquatically intimidating constellation Cetus.
Seen edge-on, the island universe spans over 100,000 light-years, a little larger than our own Milky Way.
The colorful stars in this cosmic close-up of NGC 1055 are in the foreground, well within the Milky Way.
But the telltale pinkish star forming regions are scattered through winding dust lanes along the distant galaxy's thin disk.
With a smattering of even more distant background galaxies, the deep image also reveals a boxy halo that extends far above and below the central bluge and disk of NGC 1055.
The halo itself is laced with faint, narrow structures, and could represent the mixed and spread out debris from a satellite galaxy disrupted by the larger spiral some 10 billion years ago.
NGC 1055 Close-up: Image Credit & Copyright: Processing - Robert Gendler, Roberto Colombari
Data - European Southern Observatory, Subaru Telescope (NAOJ), et al.

ADVERTISERS

Publisher: Foundation for Supporting the Pascal Programming Language
in collaboration with the Dutch Pascal User Group (Pascal Gebruikers Groep)

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.

NEW DETAILS

Subscriptions (2017 prices)

	Internat. excl. VAT	Internat. incl. VAT	Shipment
Printed Normal Issue 44 pages	€ 100	€ 106,50	€ 75
Printed Extended Issue 80 pages	€ 150	€ 159	€ 100
Electronic Download Issue 80 pages	€ 50	€ 60,50	———

Member and donator of
WIKIPEDIA

Printed magazine edition
10 issues per annum, 44-page Delphi-only section: € 100.-- This includes postage, VAT at 6 % and all code and programs accompanying the articles.
Excluding postage the 44-page edition is € 75.-- per annum.
10 issues per annum 80-page Delphi + Lazarus sections: € 150 plus € 100 for postage.
Digital magazine edition (PDF format)
10 issues per annum 80-page Delphi + Lazarus sections: € 50.-- (excluding VAt at 21%).

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

From the editor

2018 is here and I wish you very happy New Year.
I think this is going to be a very good year for Pascal developers.
We will have new opportunities and probably this year Delphi will go up in the TIOBE index.
So we must think of new and even better ways to do things. For Lazarus we can say that its rapidly developing into an environment that will soon be at the level or even beyond Delphi 7.
I can say that because I am a close follower of this program. There are some items to be solved, implementing things like Interfaces, Attributes and Generics. Of course there are quite a few other smaller items to be solved as well.

In this issue we have made an overview of the new features of Delphi: the Theme.
There is little else to mention and I think they really should wait a little longer for their next release to be ready.
So that it won't be necessary to update the program within the moment it was published.

What surprised me was that on the license logo (*the splash screen*) appeared the name of Idera.
I know Idera is the owner of Embarcadero, but I think it's confusing because I understood Embarcadero was the company that owns Delphi.

I like the new colour themes very much and I think they can be very helpful in making difference between working in the dark hours or at day time.
What I found remarkable is that some colour combinations don't help, even hide parts of the IDE. So it will take some time to find combinations of settings are looking good.

I appreciate the new welcome screen because it makes getting information for some items quite a bit more helpful. It's nice to see the direct hit for video etc.

It is also a good idea they now created a new version of the starter version. I think that will be much appreciated, good to know that Bob Swart's Delphi starters book is still available for free.

No news or explanation of the newly acquired firm "Sencha". I will dive into that in the next issue.
I had expected some explanation of what this means for us as developers in Pascal. It's not clear to me, so I will try to find out and explain the next time.

In this issue there is also an article about Cross VCL and FMX Linux. That is all new and very interesting. We will write about that in future articles.

Finally we can tell you something about PAS2JS.
We have quite a long article about that and that is necessary to explain all the quite difficult ways to handle it and to follow some demo projects.
Not quite the RAD development but it's a start.
If you are interested in the Beta-version you can send a request to me and I'll let you know how and what. This Beta testing will officially start after **31st of Januari 2018.** You will be registered and receive the Beta version and demofiles like we have here in the magazine.

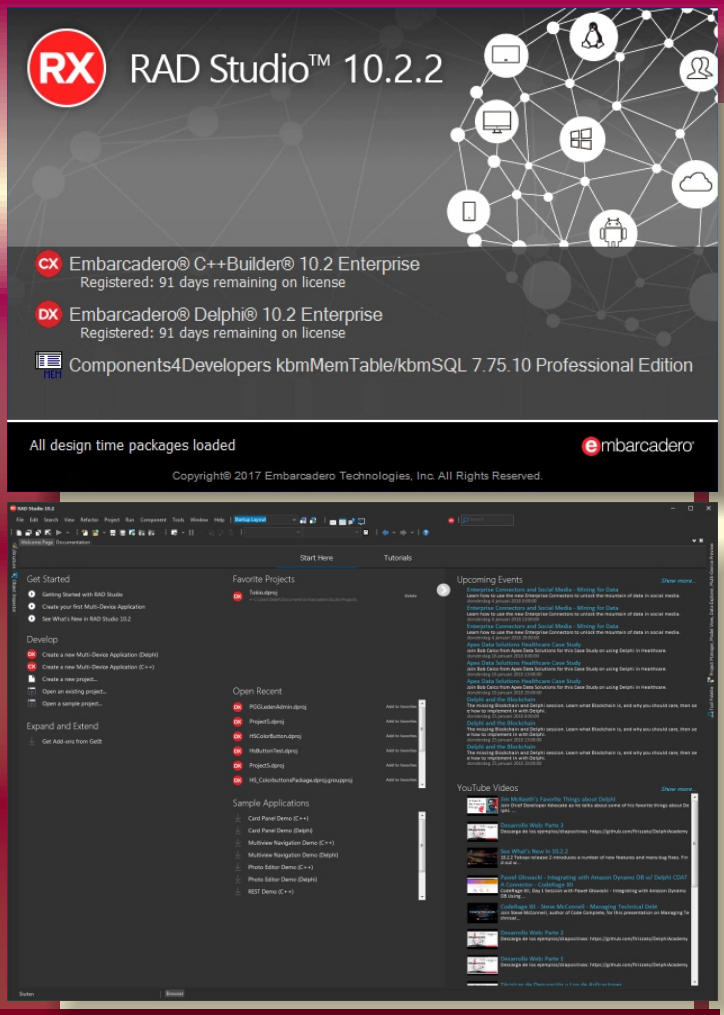
There are still two promises open: the publishing of our new website and creating a new book for Lazarus.
Now that I have finished this fresh item I will put my attention to that.

I hope you will find this a very interesting issue and you will have lots of new ideas to play with...

happy readings...
Detlef Overbeek

IN MEMORIAM PAWEL GLOWACKI

It is with great sadness that we have to inform you about the sudden loss of Pawel Glowacki. He was only 47 years old. Pawel was employed by Embarcadero as Technical Lead but worked very closely with the Barnsten team because he lived in Amsterdam. Pawel had great influence on the Barnsten events and webinars. He always knew flawlessly what was going on in the development tools market. Many new themes and programs have arisen from consultation with him. Most of you have been in direct or indirect contact with him at the live events or via his webinars, books, blogs, white papers etc.His optimism, technical knowledge and positive attitude to life left a deep impression, but also a great emptiness at the loss of such a unique individual. Pawel has been buried 22 December 2017 in his homeland Poland.



TOKYO 2.2 PAGE 1/12
BY DETLEF OVERBEEK

THE EXE FILE: GO TO URL
`http://altd.embarcadero.com/download/radstudio/10.2/radstudio10_2_2_esd_2004.exe`
`(MD5: 40FC2532A2FBE769BA3754AF7432621B)`
THE ISO FILE: GO TO URL
`http://altd.embarcadero.com/download/radstudio/10.2/delphicbuilder10_2_2_2004.iso`
`(MD5: AC1FA2E0E9BE86B5118742B782477B61)`
In your PDF file you can simply click on this address.

For those who want the starter edition, this will be discussed later in this article.
It has to be installed on a separate windows environment because it can not be handled on the same environment as your complete version. Don't do it, it causes a lot of trouble...
So now you will have to make a very basic choice: do you want to download the .exe file or do you want the .ISO? Or did you have already downloaded it?

FIRST THINGS FIRST:
If you have downloaded the .exe file it may have an advantage over the .ISO file. There is no easy way to get the iso file except if you have already a subscription.

If you use the exe file there is nothing to worry about, all you need to know is that the .exe file means a small executable, which will start up the process: download the program (*all the zipped files etc. it does it automatically and is very easy to use*).If you do not have a very fast internet connection you better take the .ISO

The installation for the ISO file works like this:
Drag the ISO file into a virtual drive and then copy the complete contains of that drive into an install directory on the hard disk, hoping this will be a quicker and safer way to run the setup.exe from your own hard disk (See figure 1). Installing it on a win7 OS - which I am still working on, for reasons of not having time to organize all my programs on Win 10 yet.
But to make sure I downloaded the `radstudio10_2_2_esd_2004.exe` file and installed it on a Win 10 version.)
It also has the advantage for you to be able to see if there are any differences on Windows10 where you can see the specific settings of Win 10 colour themes.
It takes quite some time to find a colour setting for me that I really enjoy...but that's for later.

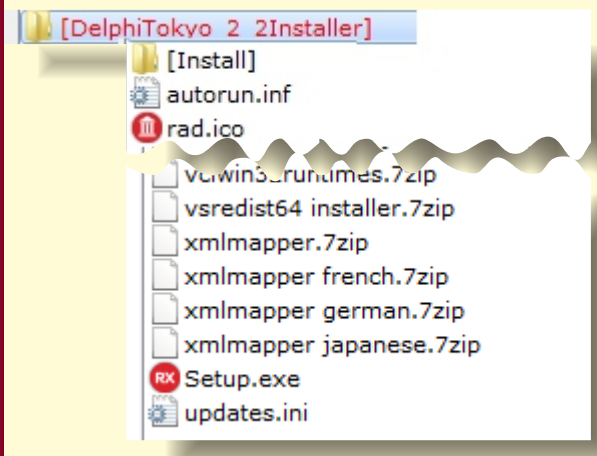


Figure 1. the installer directory

The new Delphi version Tokyo 2.2 was published some days ago and is now available.
After having published a version that had quite some issues, now there is a version that works as far as I know without big problems.
Developers that simply install the program for the first time should also read this article because it gives an insight where you find your details for tuning the program. In this case change colours.

It is obvious you know that your eyes are important and that when you are coding sometimes or often more then 8 hours a day, you might think of a background that is less demanding than pure white... Actually your screen is nothing but a very large neon tube. So your looking into a small sun...

But for the large number of people that use this version as an update: there are still some issues. Problems that actually can be solved.
But it cost a lot of time if you want to find out how it can be done...
So here is an overview how to solve these problems as well find some examples how to arrange things.

One of the first things you need to know:
this still an update. Not a major release.
Get the newest version, you simply can download that from the Embarcadero site, or the addresses in the right column.

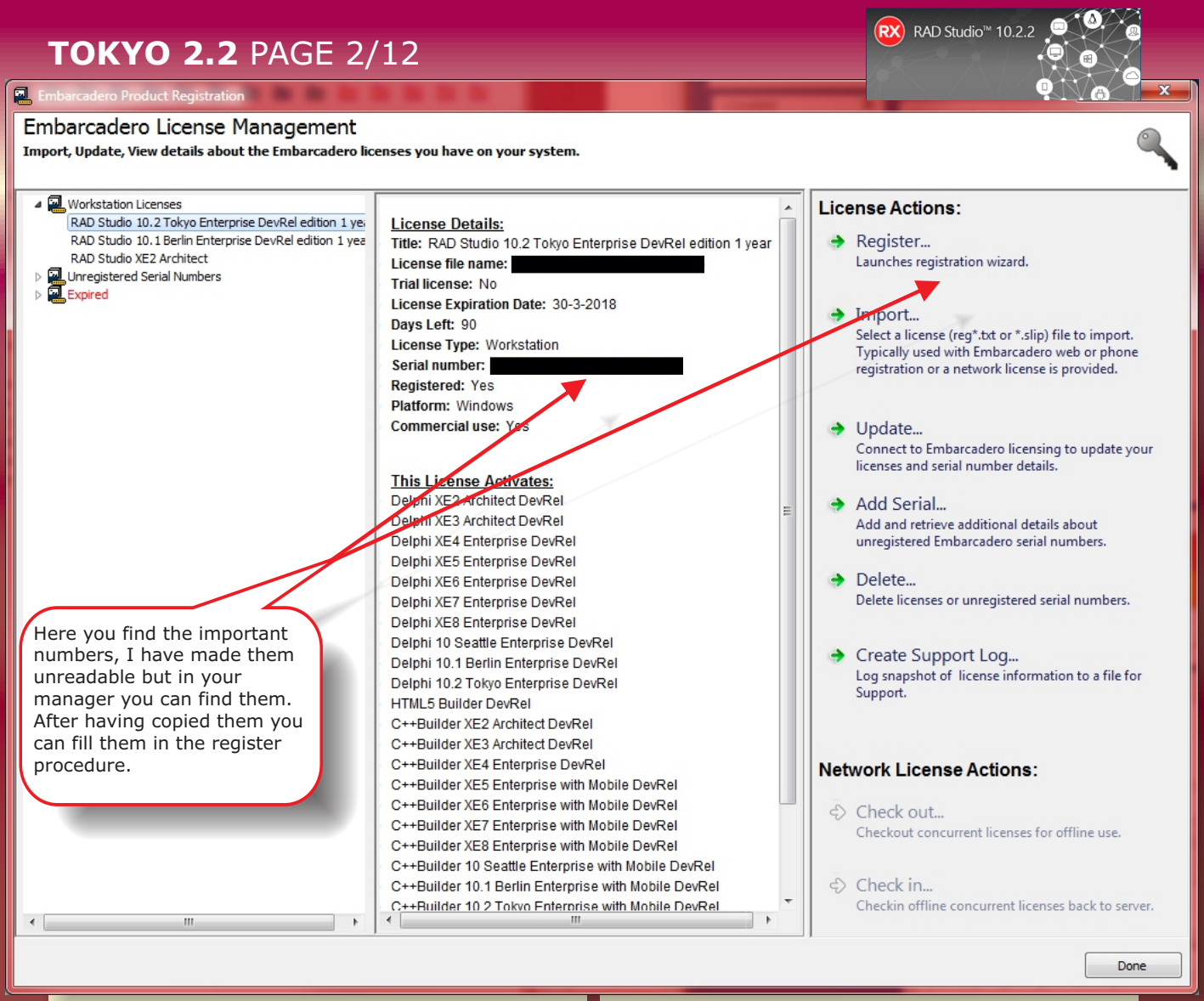
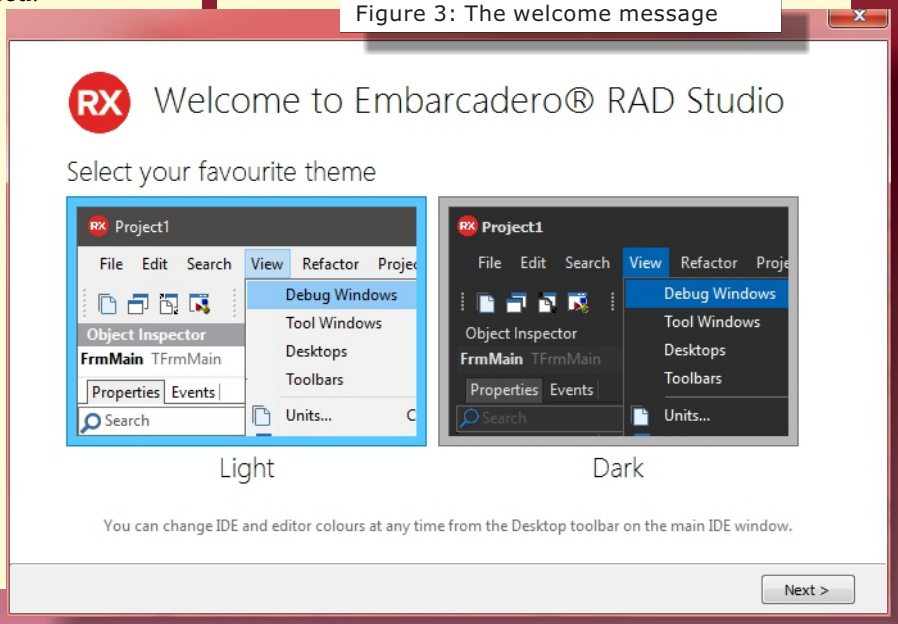


Figure 2: The license manager

After installing I found -because I had the older version of Tokyo already – it is good to know that **it REMOVES the old one and INSTALLS the new version.**
Only a small difficulty.
As soon as it was installed I found that it started and then simply told me it was not registered.
(*I was warned for this, but anyway I thought this to be already handled with the newest update.*)
So restarting didn't help.
Restarting Windows was of no use - so what else?
Trying to find a registering option.
Can't find it.
In earlier versions a License Manager existed in the list of programs…
So where is it?
If you have Delphi installed in the default way, here you can find it;
`c:\Program Files (x86)\Embarcadero\Studio\19.0\bin\LicenseManager.exe` .
As you can see in figure 2 (*the license manager*) you now can register your version by taking out the details for your serial number.
And now, once you start the program again it works and for the first time you see the new Delphi.
The wizard for your choices appears.
It does it only once! But we will find out how to get it working back again...

Figure 3: The welcome message



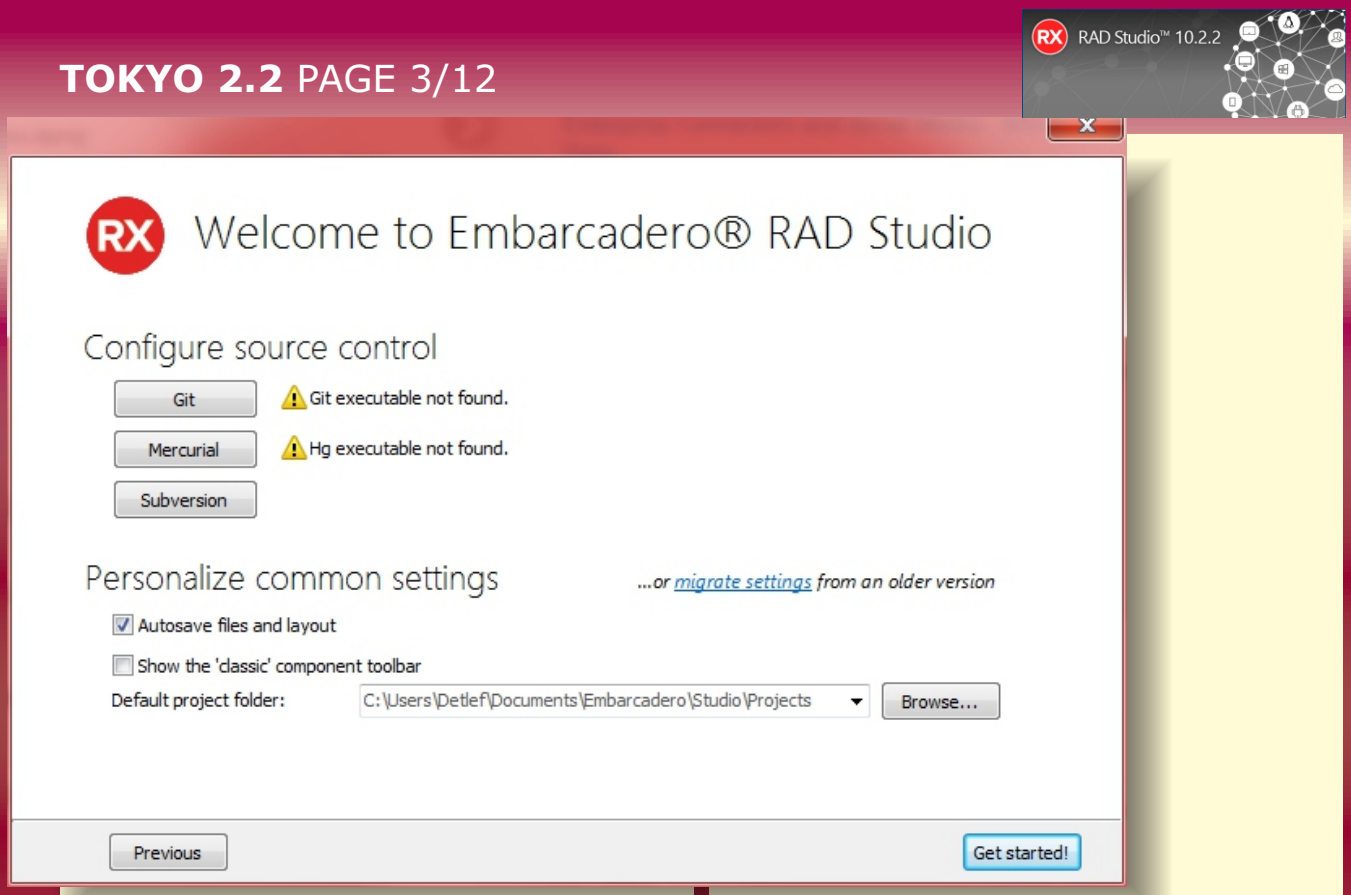


Figure 4: The wizard lets you make your choices

The first time you go by the wizard it takes you to the **Options** page of **Delphi** and opens the window at the point of **Version Control**.
You should make your choice here. If you search on your installation you will find nothing like the **Git** executable. You must download it from the internet yourself. But let's do that later.
It's the same for **Mercurial** as you can see in figure 4.
Here is also the path for your Default project folder. `c:\Users\Detlef\Documents\Embarcadero\Studio\Projects\`.

You can of course create your own environment, by setting this up in your own directory.
So now you can close the Wizard by clicking on the **Get started** button. This opens as I told before the options page: in Delphi that means go to **Tools** and open the **Options** page.
Here there are all the settings you will ever have to look for. Now to change your **colour theme** look for this section: **Theme Manager**.
Here you can actually find all the settings that as it did for me: it dazzles by the number of options.
There are three sections: **Light | Dark and Custom**.
If you use the settings I made, You will see the result is like the images. I chose this because of not having sharp light coming from the screen...

To have a real overview of the working result it is best to open up a program or just start a new application.
If you have your welcome page opened, you will not get the right impression. But see for your self.

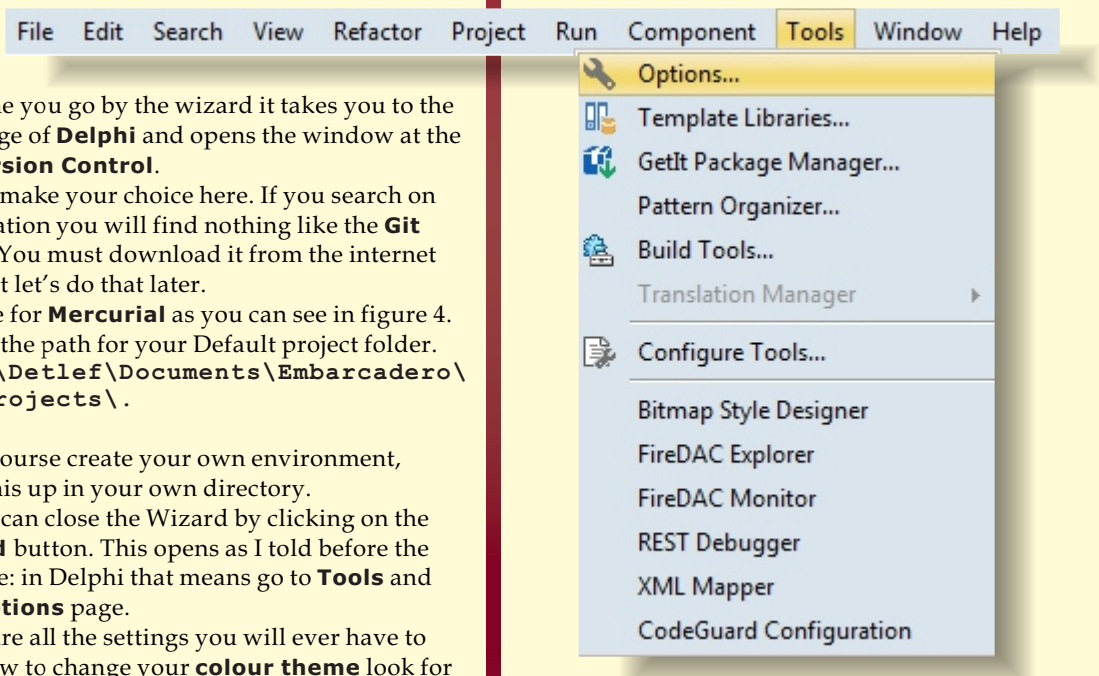


Figure 5: The options menu

GO TO URL

IF YOU SEE THIS BUTTON YOU CAN (IN THE PDF FILE) SIMPLY CLICK ON THE ADDRESS AND THE WEB ADDRESS WILL OPEN.

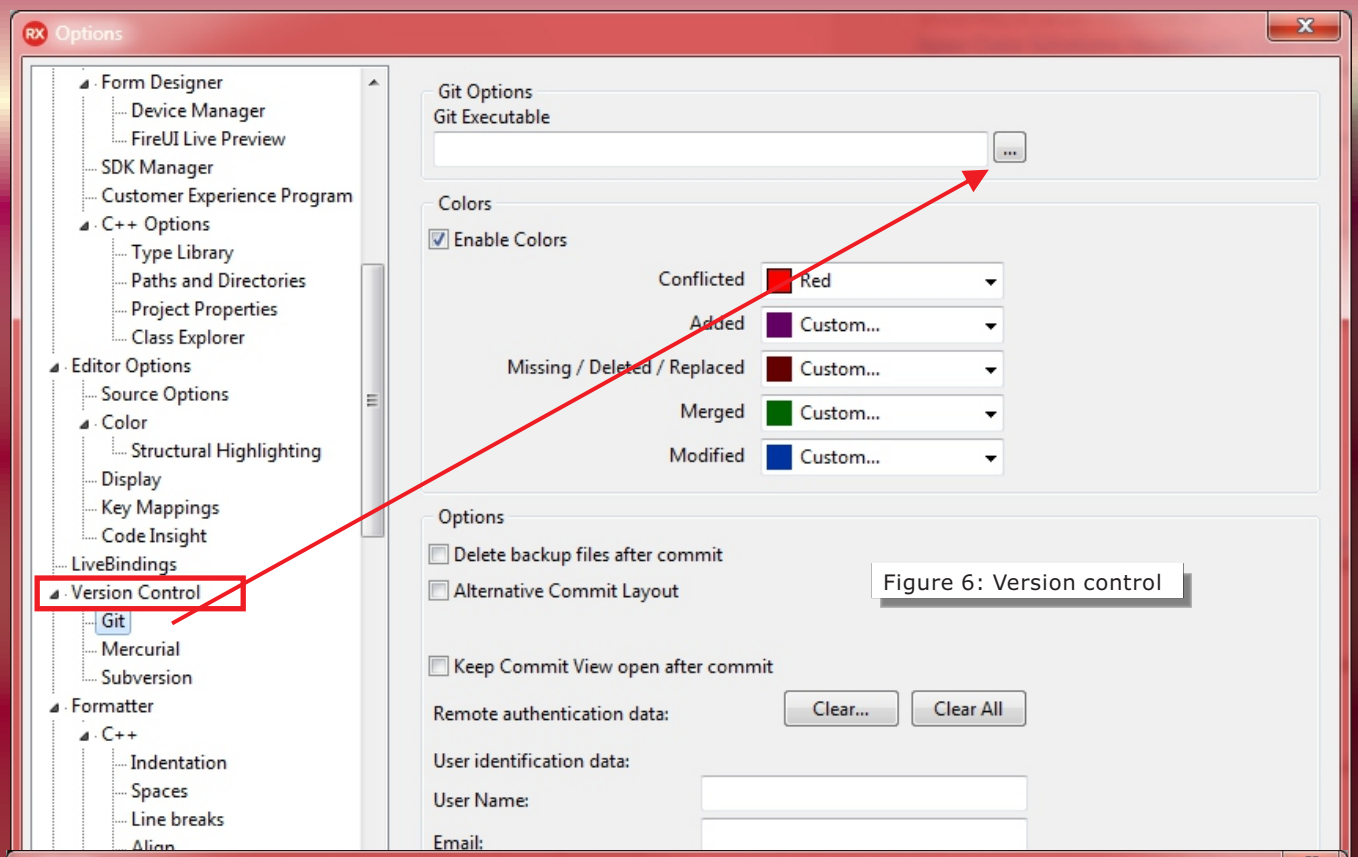


Figure 6: Version control

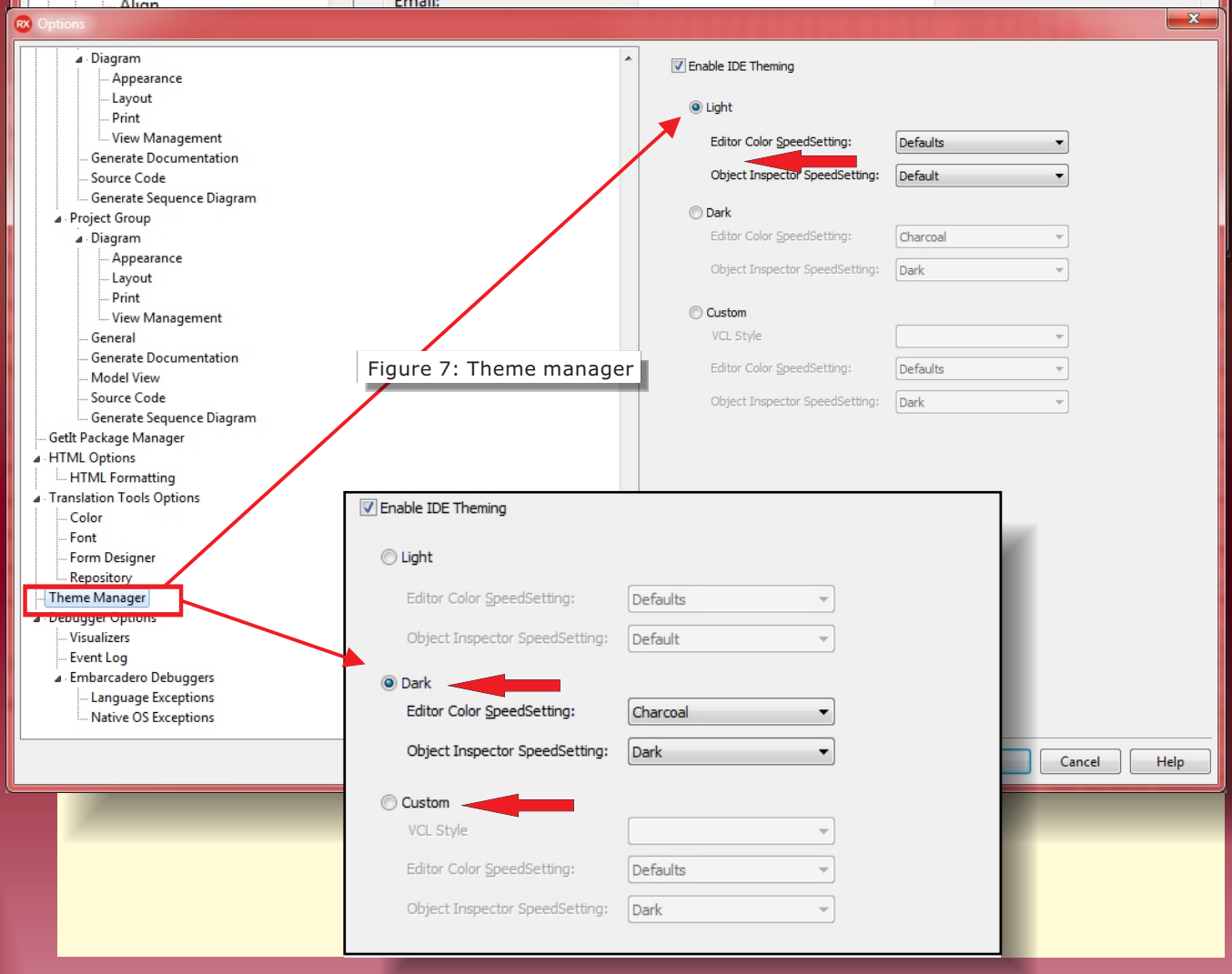


Figure 7: Theme manager

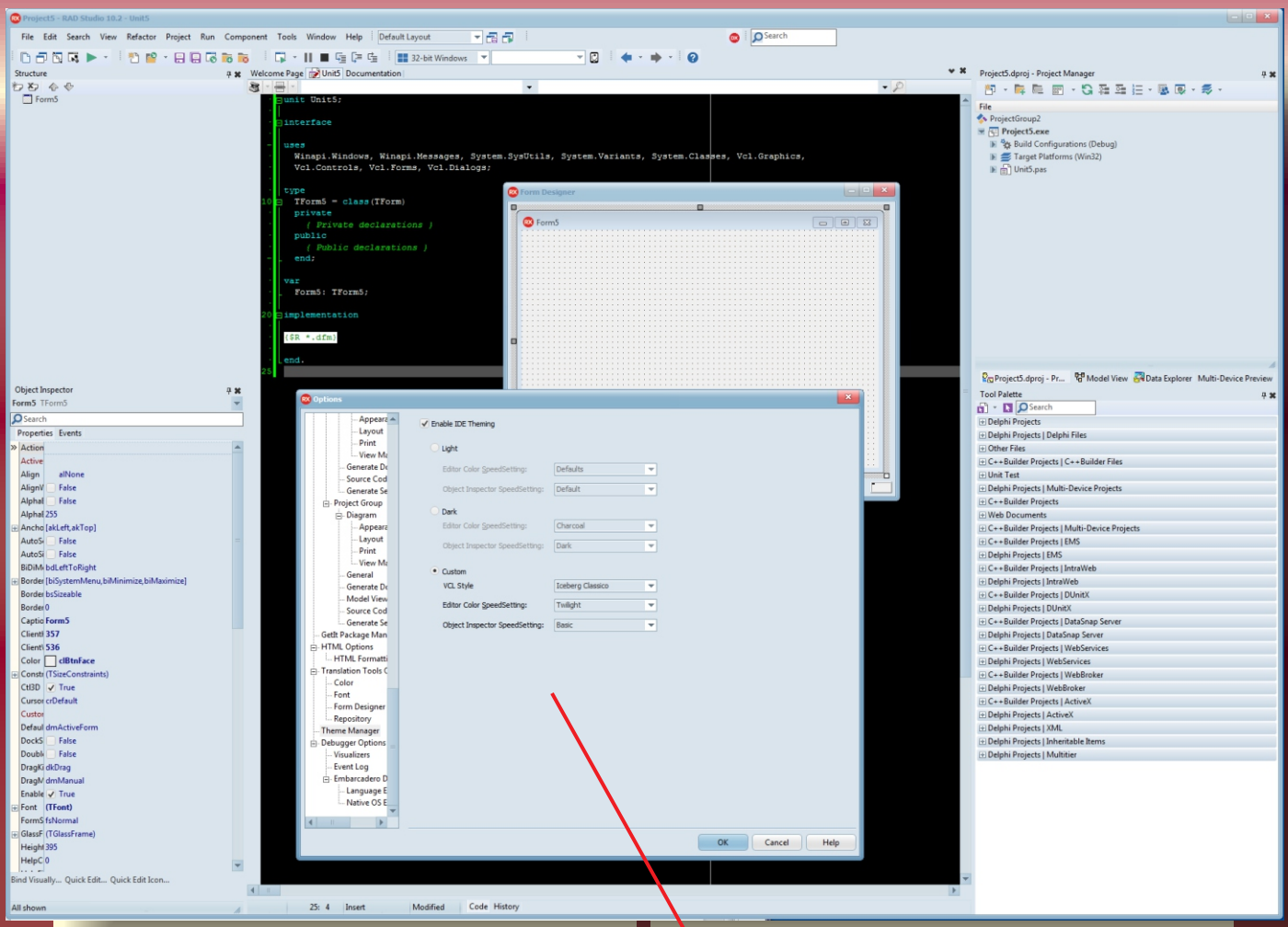


Figure 8. To have a real overview of the working result it is best to open up a program or just start a new application.

I here have chosen for the Dark Code Editor and the light grey-blue-ish color for all other normal white segments: the exact recipe is:

CUSTOM:

VCL Style **Iceberg Classic**
Editor Color SpeedSettings **Twilight**
Object Inspector SpeedSettings **Basic**

It might be a good idea for the near future to have a preview of what you are doing...
Problematic is often that your screen is totally rearranged in size even though you set it back each time . A possibility is to save every trial and give it a special or meaningful name, so you can find the version you created back again.

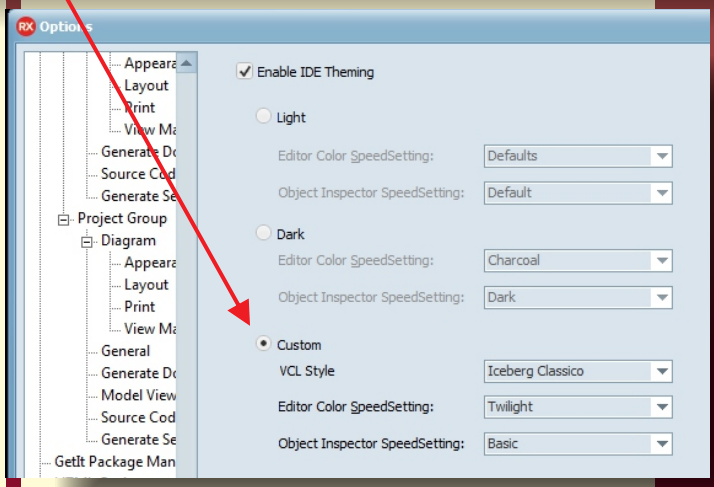


Figure 9. Detail of the settings

GO TO URL

IF YOU SEE THIS BUTTON YOU CAN (IN THE PDF FILE) SIMPLY CLICK ON THE ADDRESS AND THE WEB ADDRESS WILL OPEN.

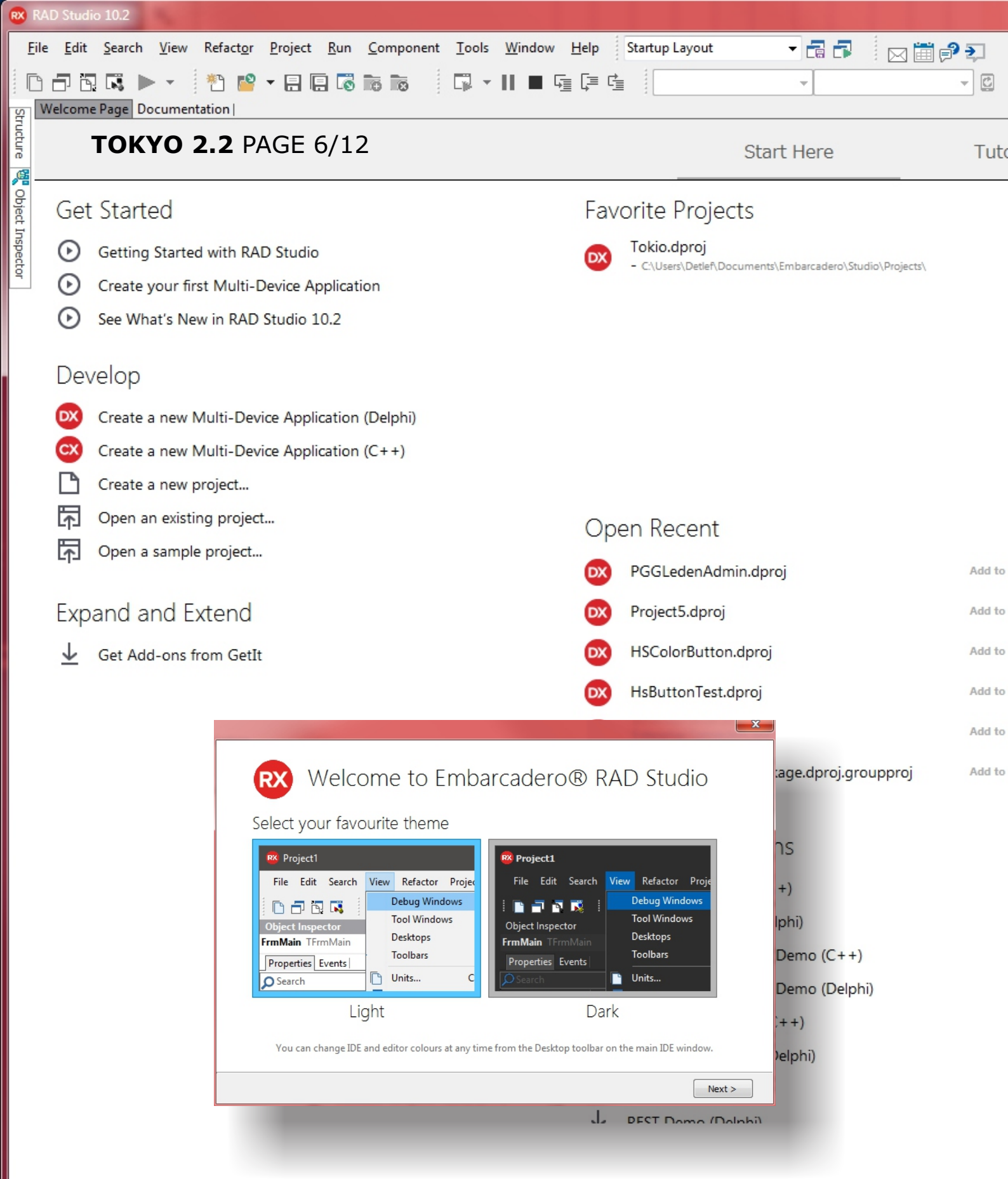


Figure 10: Overview of the first Opening of Delphi Interesting is that there is a better overview, more information and the videos that are available.

Videos are an item itself though they can give al lot of information at first glance.
I personally favor the written article which tells in depth what you do and how to do it.
A video is good for quick info, But to understand it you need to review it often,
and usually doesn't go into details about the subject

Upcoming Events

Show more...

Enterprise Connectors and Social Media - Mining for Data
Learn how to use the new Enterprise Connectors to unlock the mountain of data in social medi a.
donderdag 4 januari 2018 8:00:00

Enterprise Connectors and Social Media - Mining for Data
Learn how to use the new Enterprise Connectors to unlock the mountain of data in social medi a.
donderdag 4 januari 2018 13:00:00

Enterprise Connectors and Social Media - Mining for Data
Learn how to use the new Enterprise Connectors to unlock the mountain of data in social medi a.
donderdag 4 januari 2018 20:00:00

Apex Data Solutions Healthcare Case Study
Join Bob Calco from Apex Data Solutions for this Case Study on using Delphi in Healthcare.
donderdag 18 januari 2018 8:00:00

Apex Data Solutions Healthcare Case Study
Join Bob Calco from Apex Data Solutions for this Case Study on using Delphi in Healthcare.
donderdag 18 januari 2018 13:00:00

Apex Data Solutions Healthcare Case Study
Join Bob Calco from Apex Data Solutions for this Case Study on using Delphi in Healthcare.
donderdag 18 januari 2018 20:00:00

Delphi and the Blockchain
The missing Blockchain and Delphi session. Learn what Blockchain is, and why you should care, then see how to implement in with Delphi.
donderdag 25 januari 2018 8:00:00

Delphi and the Blockchain
The missing Blockchain and Delphi session. Learn what Blockchain is, and why you should care, then see how to implement in with Delphi.
donderdag 25 januari 2018 13:00:00

Delphi and the Blockchain
The missing Blockchain and Delphi session. Learn what Blockchain is, and why you should care, then see how to implement in with Delphi.
donderdag 25 januari 2018 20:00:00

YouTube Videos

Show more...

Jim McKeeth's Favorite Things about Delphi
Join Chief Developer Advocate as he talks about some of his favorite things a bout Delphi. ...

Desarrollo Web: Parte 3
Descarga de los ejemplos/diapositivas: https://github.com/flrizzato/DelphiA cademy

See What's New in 10.2.2
10.2.2 Tokoyo release 2 introduces a number of new features and many bug fi xes. Find out w...

Paweł Głowacki - Integrating with Amazon Dynamo DB w/ Delphi CDATA Connector - CodeRage XII
CodeRage XII, Day 1 Session with Paweł Głowacki - Integrating with Amazon D ynamo DB Using ...

CodeRage XII - Steve McConnell - Managing Technical Debt
Join Steve McConnell, author of Code Complete, for this presentation on Man aging Technical...

Desarrollo Web: Parte 2
Descarga de los ejemplos/diapositivas: https://github.com/flrizzato/DelphiA cademy

Desarrollo Web: Parte 1
Descarga de los ejemplos/diapositivas: https://github.com/flrizzato/DelphiA cademy

Técnicas de Depuración y Log de Aplicaciones
Descarga de los ejemplos/diapositivas: https://github.com/flrizzato/DelphiA

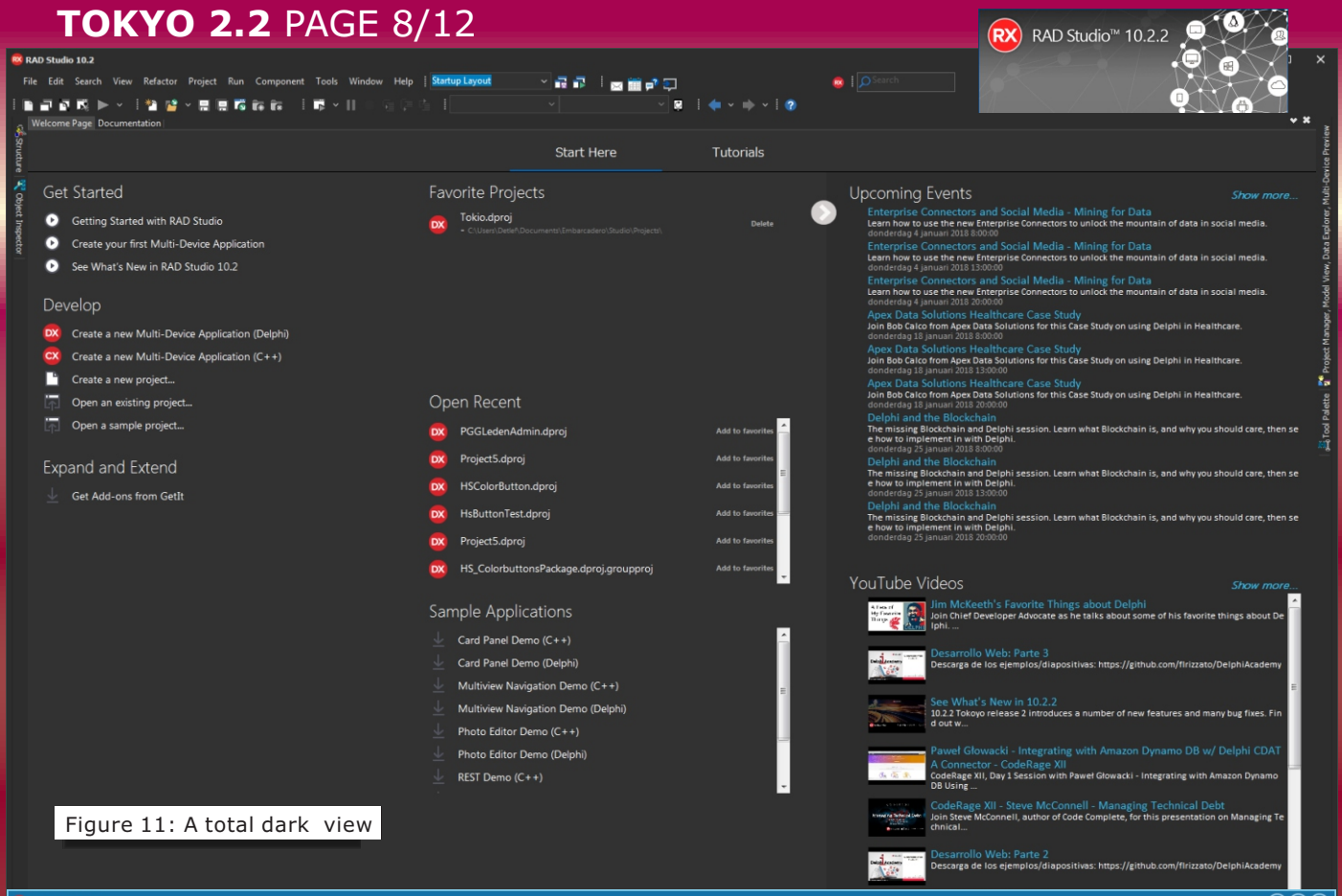


Figure 11: A total dark view

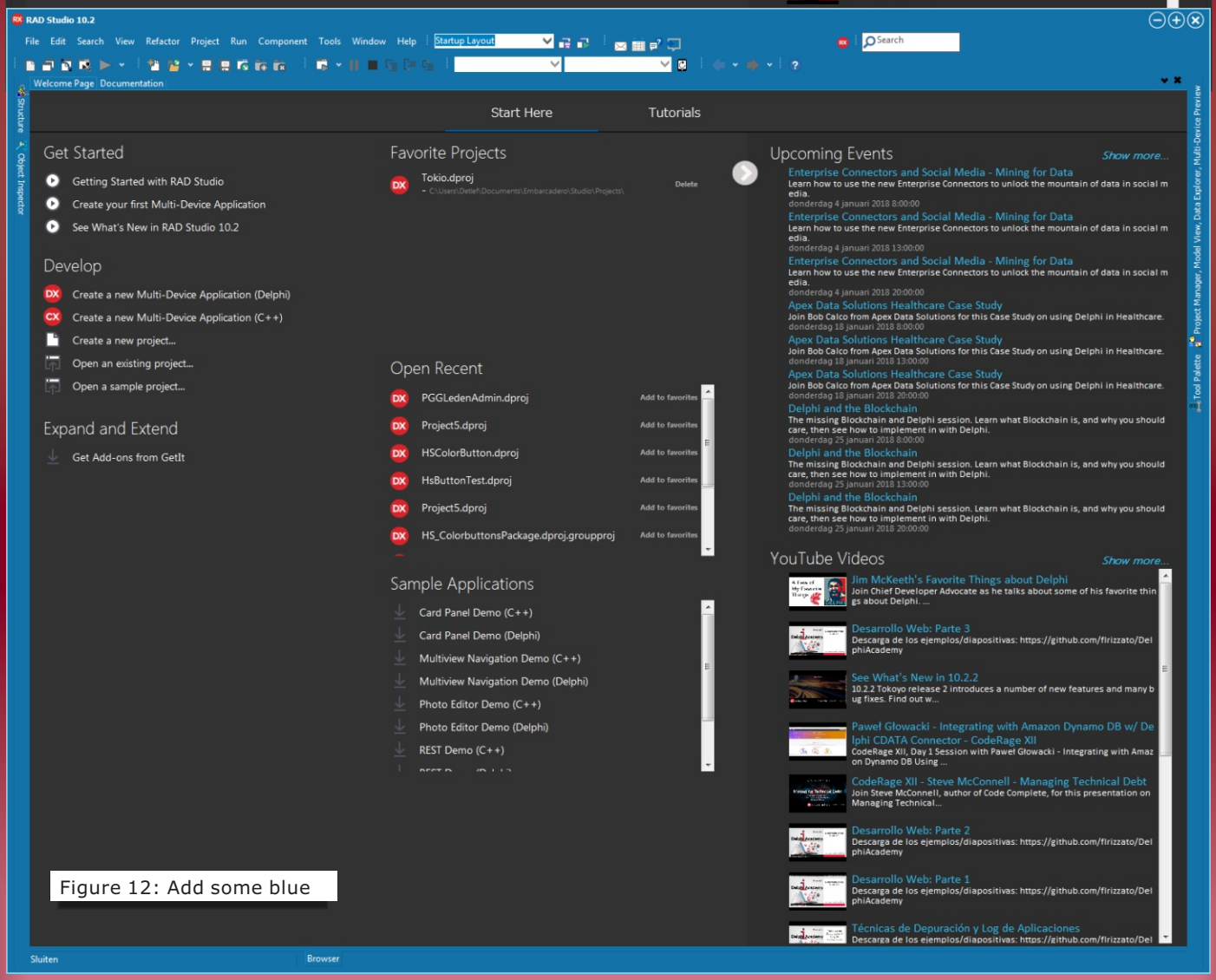


Figure 12: Add some blue

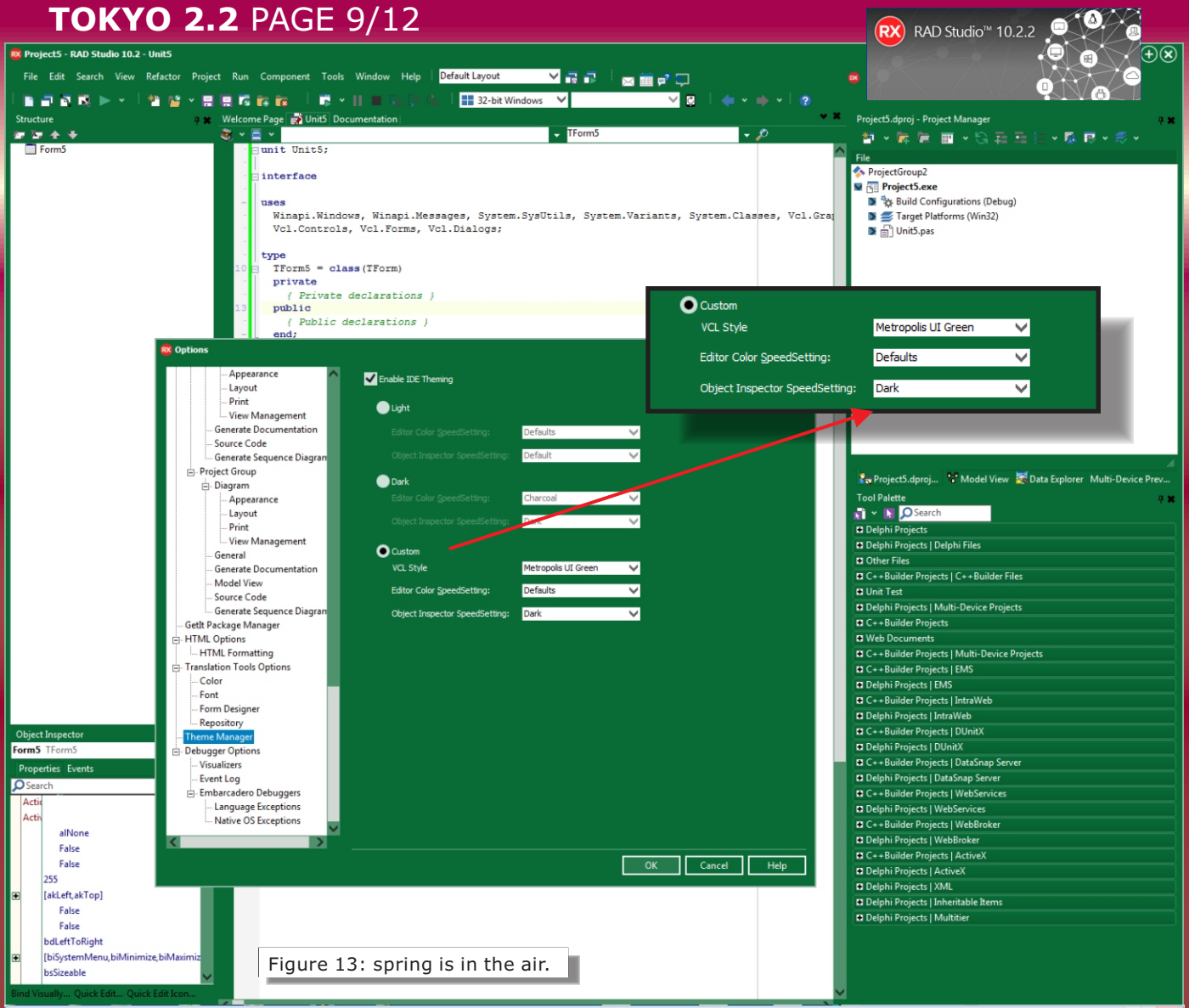


Figure 13: spring is in the air.

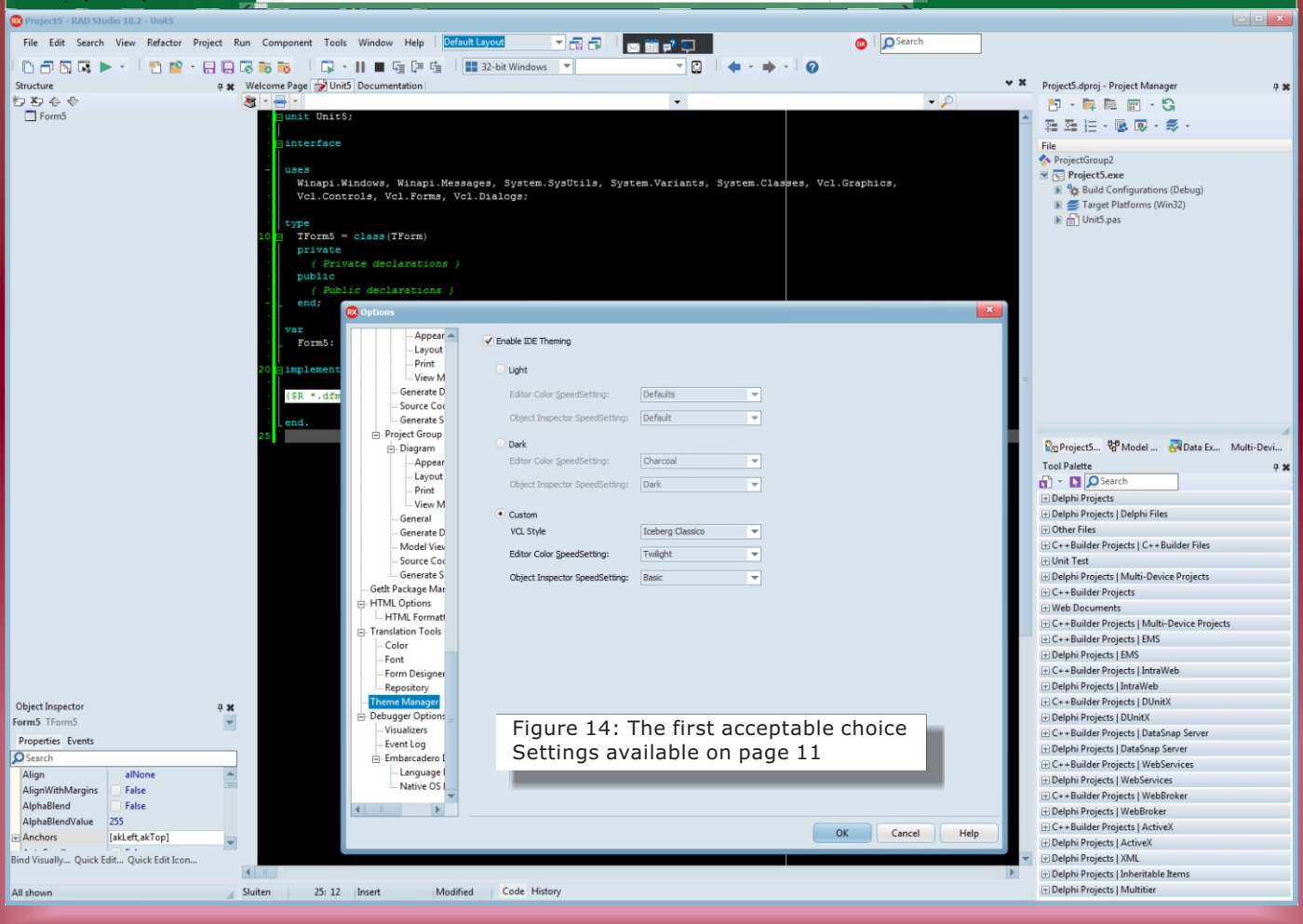
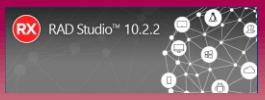
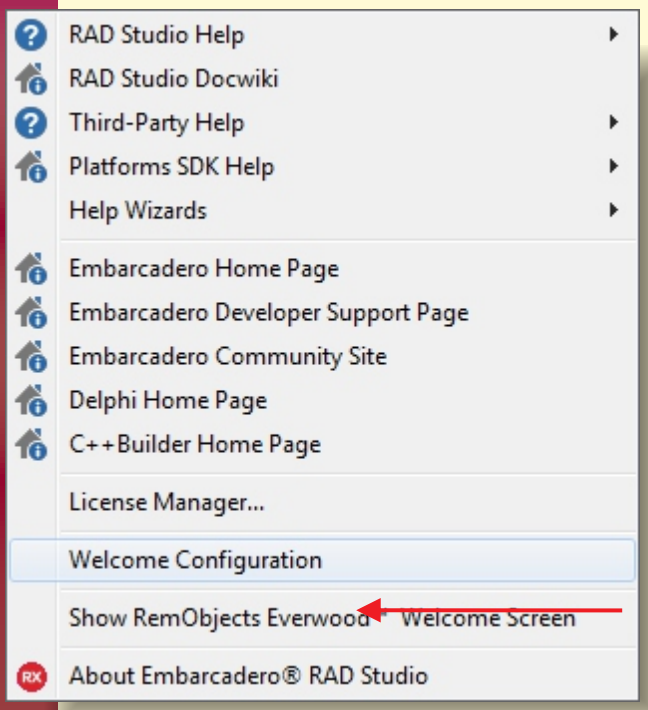


Figure 14: The first acceptable choice
Settings available on page 11



So now we found how to handle this:
the welcome screen is not available any more .
It can be found at the section Help.
On the third line from below:
"Welcome Configuration". But that's not that much important. We had found already where these settings reside.



So now we have a few subjects left:
1. Git Executable
2. Mercurial
3. Subversion
Does it all work with the **newest starter version**?

FINDING THE GIT EXECUTABLE:
It is not automatically installed on your system together with Delphi.
(Git *is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.*
As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.
Git was created by **Linus Torvalds in 2005** *for development of the Linux kernel, with other kernel developers contributing to its initial development.*
Its current maintainer since 2005 is **Junio Hamano.**
As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.
Git is free software distributed under the terms of the GNU General Public License version 2.)

So to find **GIT** we need to go o the web:
`http://gitforwindows.org/` GO TO URL
there is more to be found on that page like an sdk etc.
(*I have placed the* `Git-2.15.1.2-64-bit.exe` *as download in your download page*)
Be aware of the fact that this program comes from Linus Torvald, so for linux it exists as well. There is also a learning webs GO TO URL
`https://try.github.io/levels/1/challenges/1`

FINDING THE MERCURIAL EXECUTABLE:
The official organisation can be foud here:
`https://www.mercurial-scm.org/` GO TO URL
(*I have placed the* `tortoisehg-4.4.2-x64.msi` *as download in your download page*)

(*Mercurial is a distributed revision-control tool for software developers. It is supported on Microsoft Windows and Unix-like systems, such as FreeBSD, macOS and Linux.*

Mercurial's major design goals include high performance and scalability, decentralized, fully distributed collaborative development, robust handling of both plain text and binary files, and advanced branching and merging capabilities, while remaining conceptually simple.

It includes an integrated web-interface. Mercurial has also taken steps to ease the transition for users of other version control systems, particularly Subversion.
Mercurial *is primarily a command-line driven program, but graphical user interface extensions are available, e.g.* **TortoiseHg,** *and several IDEs offer support for version control with* **Mercurial.**
All of **Mercurial's** *operations are invoked as arguments to its driver program hg (*a reference to Hg - the chemical symbol of the element mercury*).*
Matt Mackall *originated Mercurial and serves as its lead developer. Mercurial is released as free software under the terms of the GNU GPL v2.*
It is mainly implemented using the Python programming language, but includes a binary diff implementation written in C.

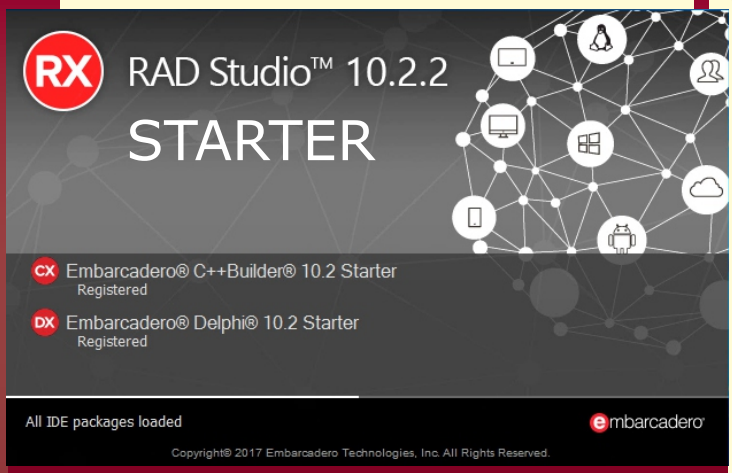
SUBVERSION Apache Subversion
(often abbreviated SVN, after its command name svn) is a software versioning and revision control system distributed as open source under the Apache License.
Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, 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 offers 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.

So to be complete we must explain something about the **CONCURRENT VERSIONS SYSTEM**
The **Concurrent Versions System** *(CVS), is a free software client-server revision control system in the field of software development.*

A version control system keeps track of all work and all changes in a set of files, and allows several developers (potentially widely separated in space and time) to collaborate.
Dick Grune *developed CVS as a series of shell scripts in July 1986.*

In addition to commercial software developers, **CVS** *became popular with the open source software world and was released under the GNU General Public License. While there was regular development to add features and fix bugs in the past, including regular builds and test results, there have been no new releases since 2008.*
So it is quite logic to choose one of the more modern versions.



Embarcadero announced a new version of Delphi Starter:
I have tested if this version has the same colour theme abilities as the big brother: Delphi Pro and up. The answer is yes. You can get it here
`https://www.embarcadero.com/products/delphi/starter/free-download`

Delphi Starter key features include:
- Develop 32-bit Windows applications using the Delphi VCL and FireMonkey frameworks
- IDE and visual development environment
- Hundreds of included components
- License for use until your individual revenue from Delphi applications or company revenue reaches $1,000 US or your development team expands to more than 5 developers

The Delphi Starter Edition is both designed and priced to allow individuals and startups to bootstrap their vision until related revenues reach $1,000 at which point a specially priced Professional Edition license can be purchased.

So, if you're an individual you may use Starter Edition to create apps for your own use and apps that you can sell until your revenues reach $1,000 per year.
If you're a small company or organization without revenue (*or up to $1,000 per year in revenue*), you can also use the **Starter Edition.**

Once your company's total revenue reaches US $1,000, or your team expands to more than 5 developers, you can move up to an unrestricted commercial license with a specially priced Professional edition license. See the Starter Editions FAQs for additional details. (**`https://www.embarcadero.com/products/delphi/starter-faq`**)
For details on the differences between the editions, see the Product Editions page details at the next page and Feature Matrix address (*also on the next page*).

Move up to the Professional edition or above to get additional features including VCL source code, multi-device development for Windows, Mac with the FireMonkey framework, components and drivers for database application development, 64-bit Delphi compiler, additional coding tools, additional debugging features, unit testing, and much more.

Bonus FREE eBook:
Get Delphi Starter and you get access to the 154 page eBook **"Delphi XE Starter Essentials"** by Bob Swart.

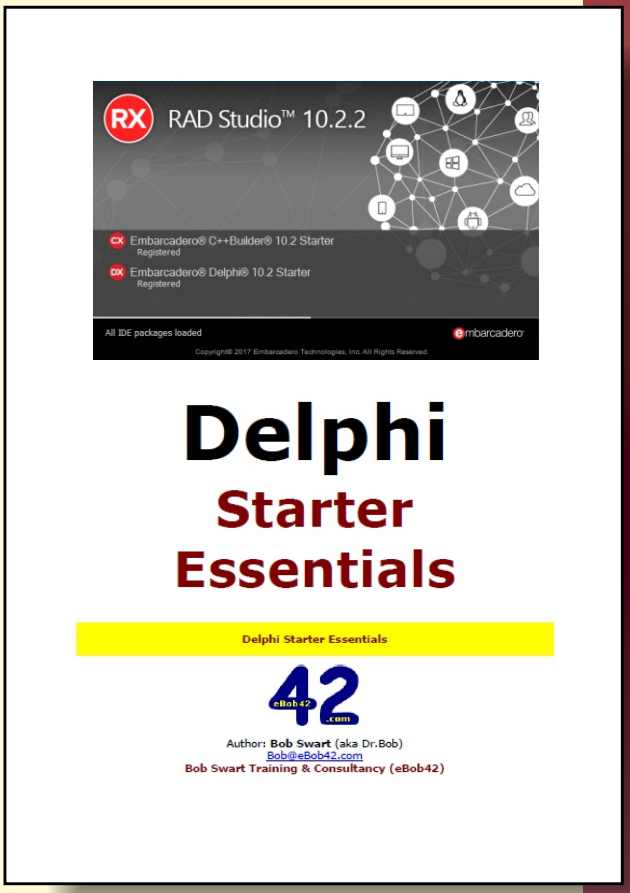


Figure 16: Bob Swarts book

	Starter	Pro	Enterprise	Architect
Develop 32-bit Windows apps	✓	✓	✓	✓
Develop 64-bit Windows apps		✓	✓	✓
Develop Universal 32-bit and 64-bit iOS apps		Optional	✓	✓
Develop Android apps		Optional	✓	✓
Develop OS X apps		✓	✓	✓
Develop Linux apps			✓	✓
Comprehensive VCL and FMX component sets	Limited	✓	✓	✓
Library source code		✓	✓	✓
Build database apps with local/embedded connectivity		✓	✓	✓
Build database apps with client/server connectivity		Optional	✓	✓
Enterprise Mobility Services1 (Developer License)			✓	✓
RAD Server (Single Site Deployment License)			✓	✓
DataSnap multi-tier SDK			✓	✓
SQL database tools				✓
Data modeling tools				✓
Commercial use license	Limited	Full	Full	Full
Access to earlier version licenses		✓	✓	✓

For the detailed Feature Matrix go to:
https://www.embarcadero.com/docs/rad-studio-feature-matrix.pdf

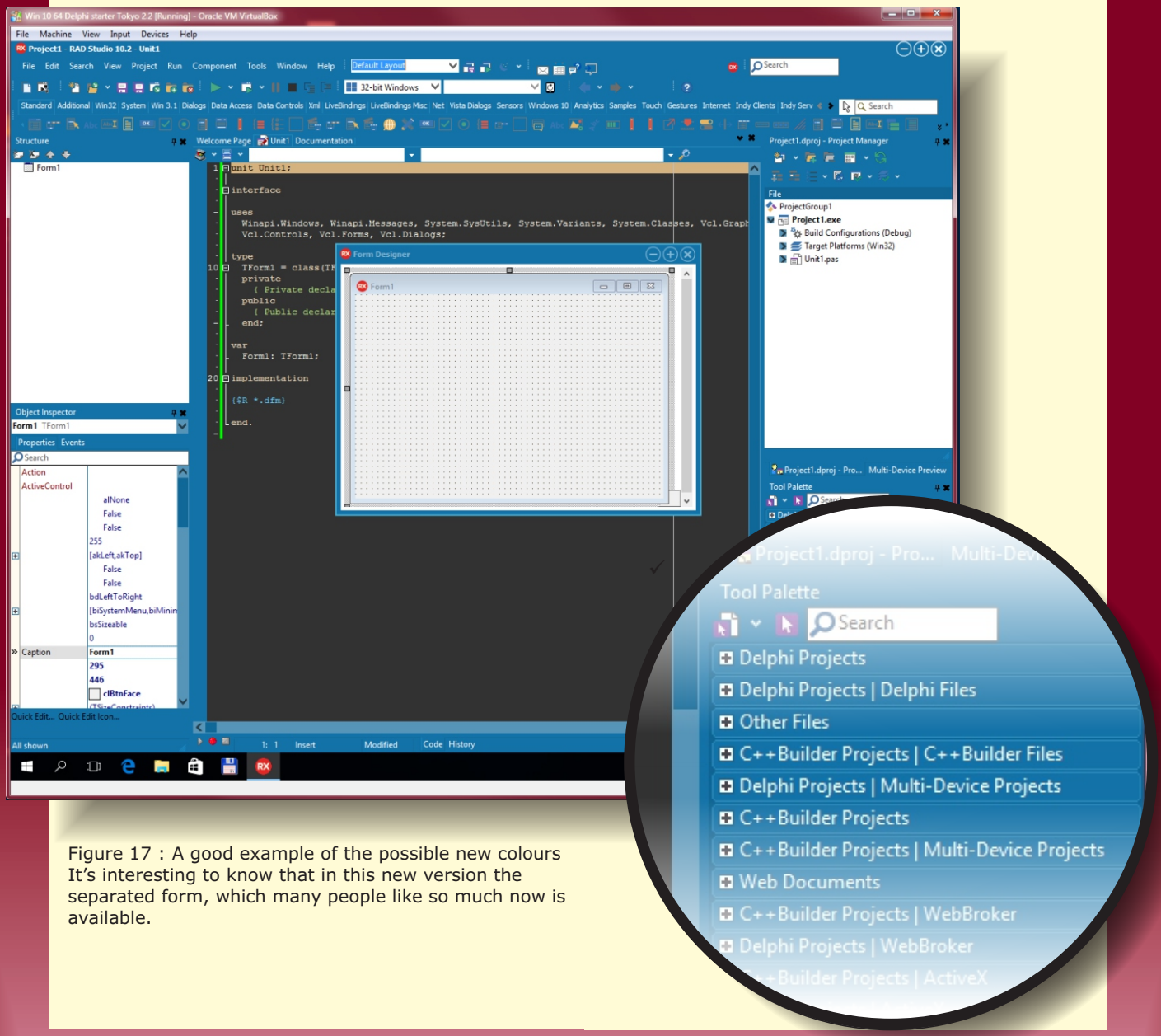
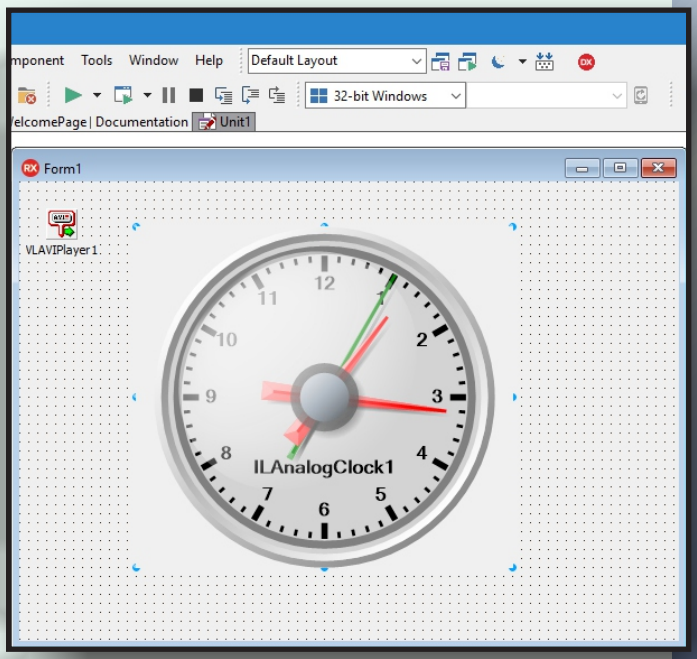


Figure 17 : A good example of the possible new colours
It's interesting to know that in this new version the separated form, which many people like so much now is available.

starter — expert

Delphi



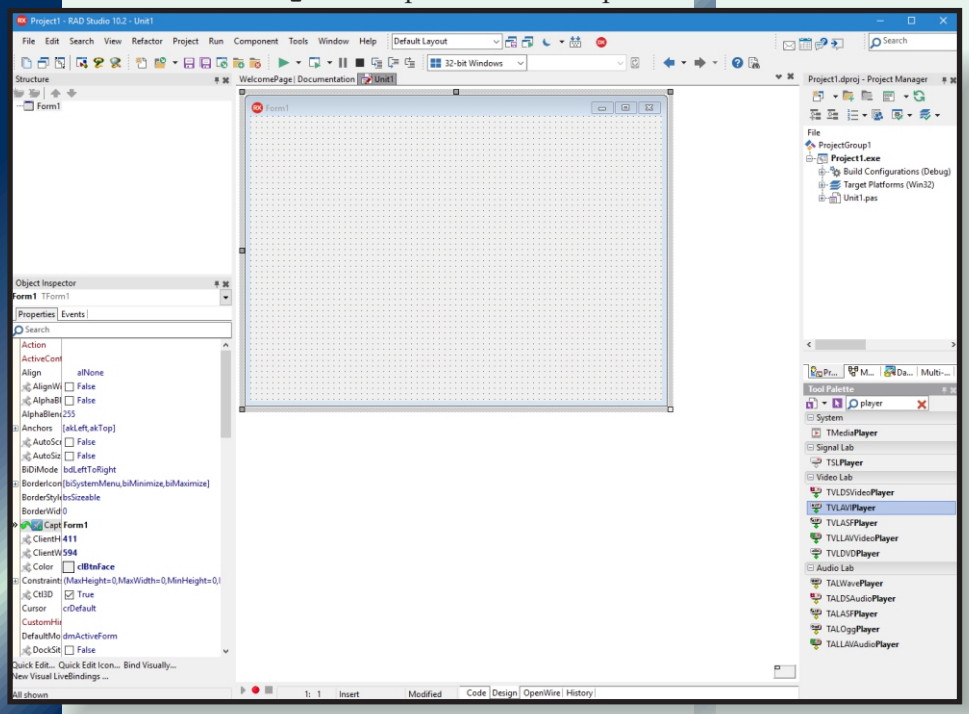
In the previous articles I introduced you to many of the cool features of VideoLab, playing filtering and morphing videos. I also showed you how you can create cool video effects and animate them with AnimationLab. I even showed you how you can render InstrumentLab gauges, and clocks inside the video.

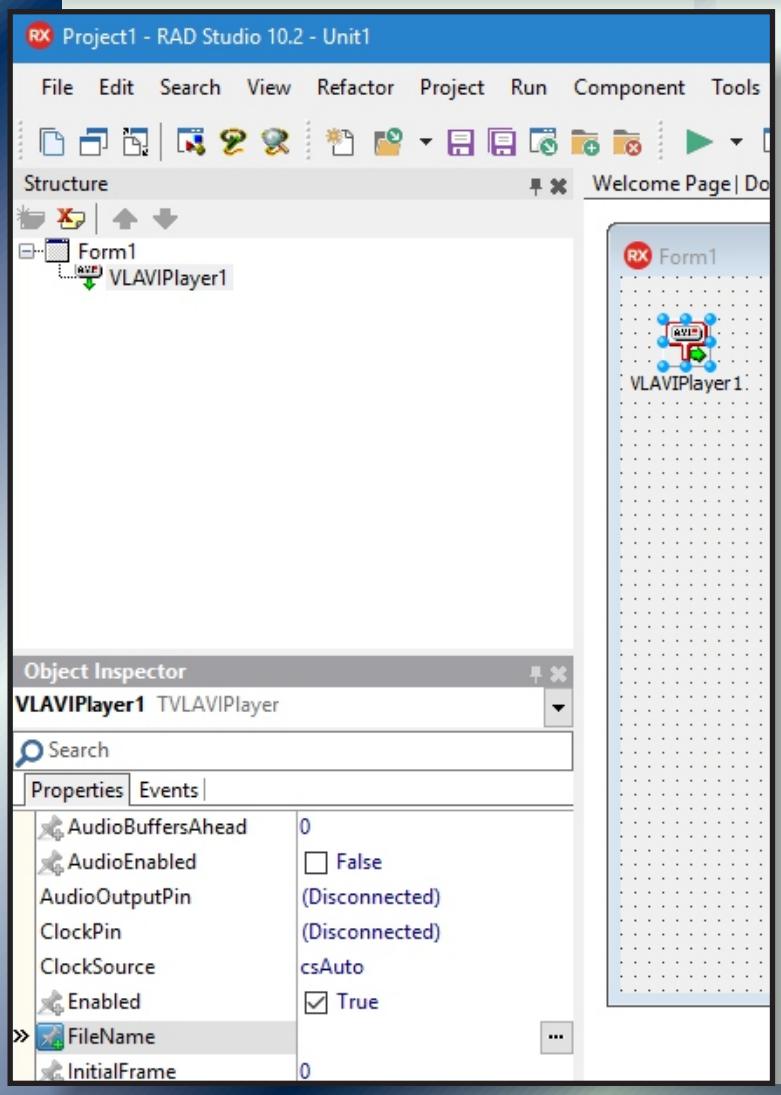
Those are just few of the Videolab capabilities. VideoLab not only can render visual instruments inside the video, but also can display video inside visual instruments, Scope or Waterfall components. It can even render videos in bitmap type elements, such as buttons in the Scope or Waterfall, or even on the surface of 3D shapes such as Round Cube.

First I will show you how you can render the video inside Analog Clock component from InstrumentLab.

Start a new VCL Form application.

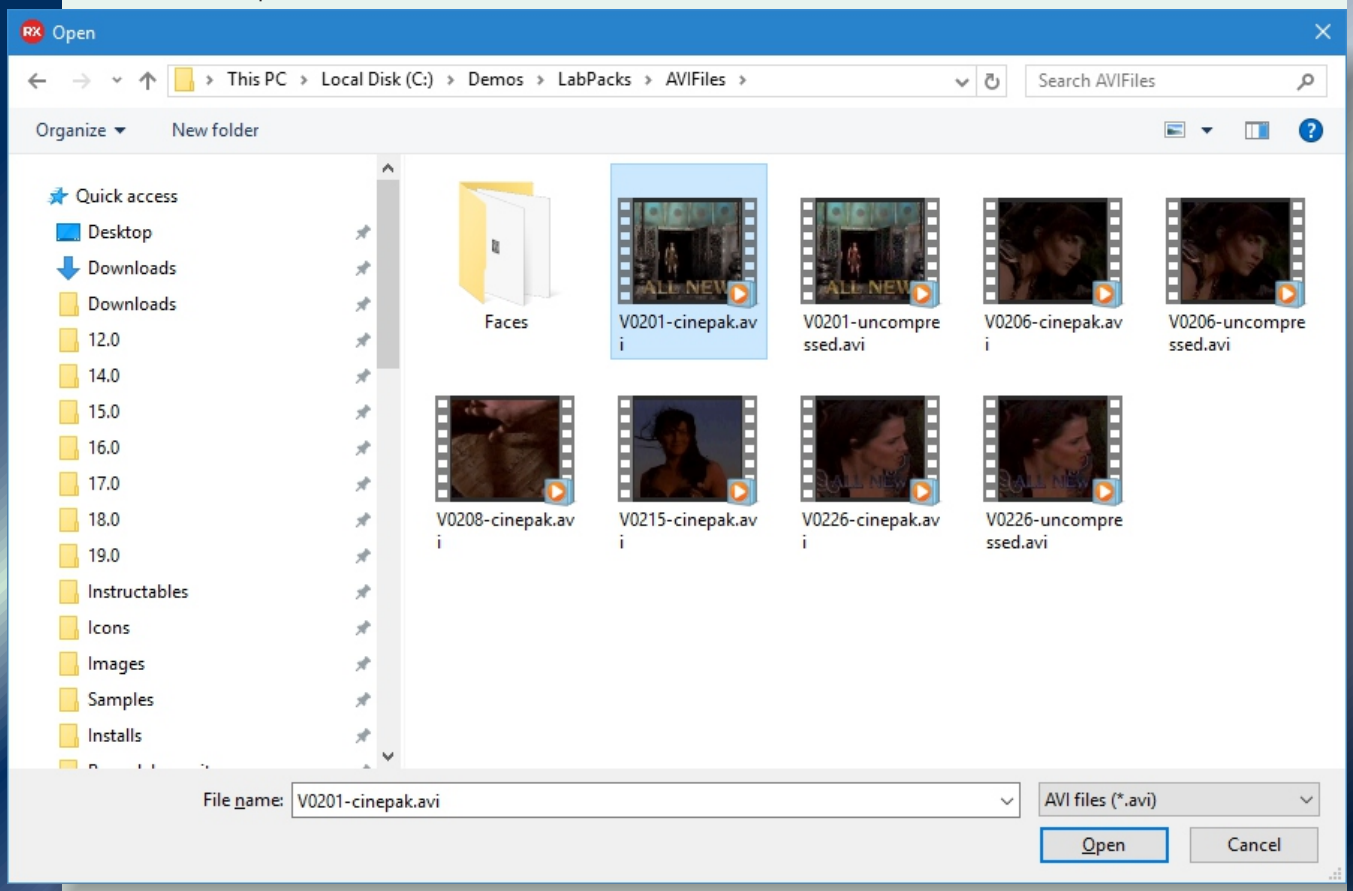
Type "player" in the **Tool Palette** search box, then select **`TVLAVIPlayer`** component from the palette:





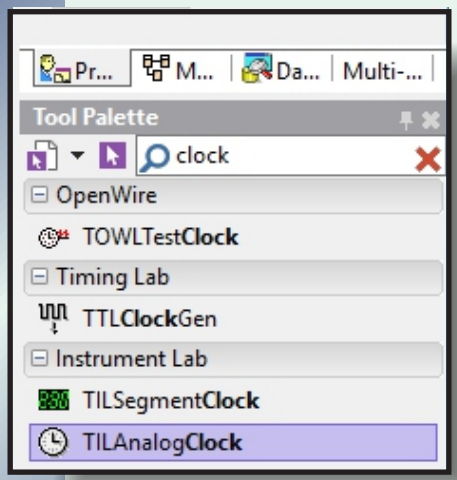
And drop it on the form:
In the **Object Inspector** select the "FileName" property, and click on the "..." *(Ellipsis)* ... button:

In the **File Open Dialog**, select a video file to play,
and click the "Open" button:

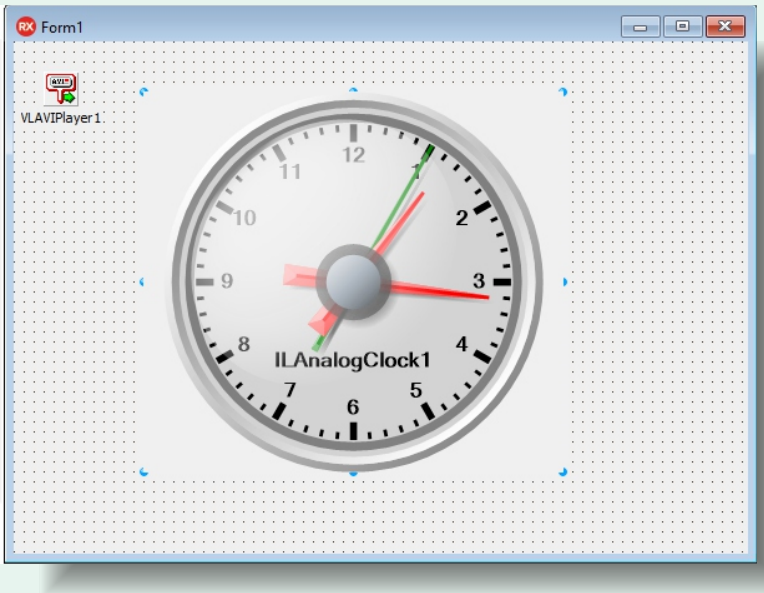


The **AVI Player** can decode only limited number of video types, so to be sure that it will be able to decode the selected video, it is best to use one of the videos included in the **VideoLab installation**.

Type "clock" in the **Tool Palette** search box, then select **`TILAnalogClock`** from the palette:

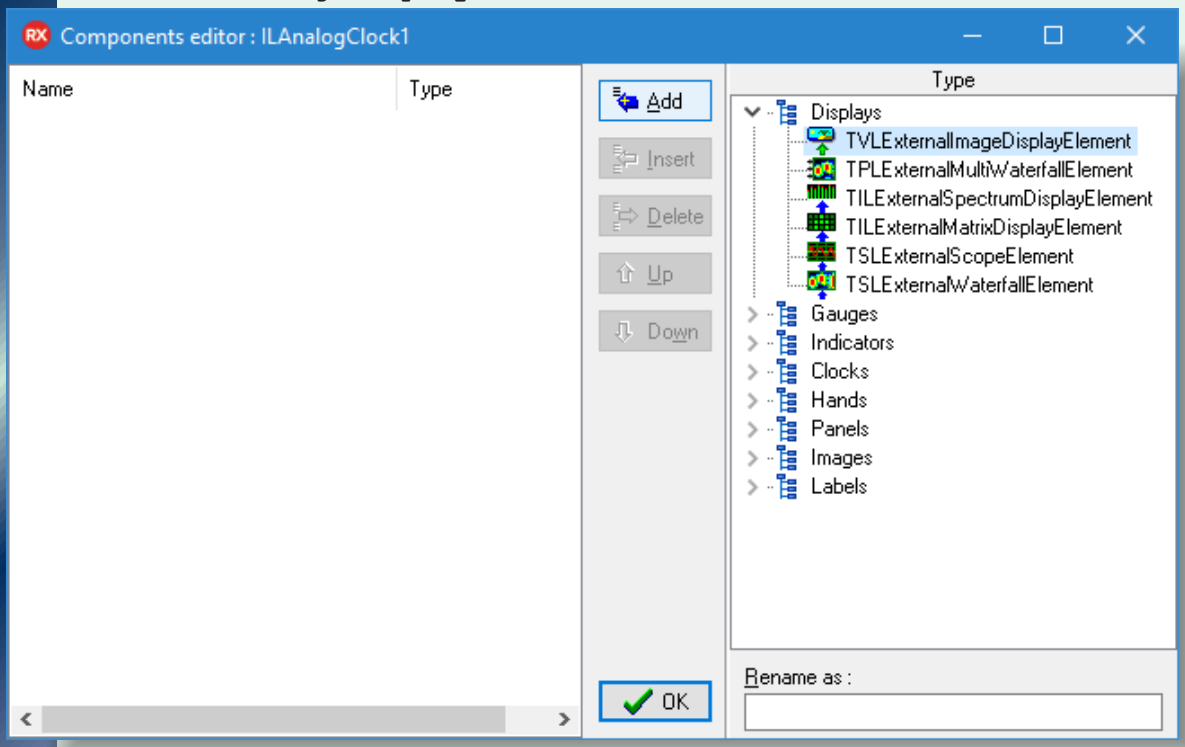


Drop it on the form:

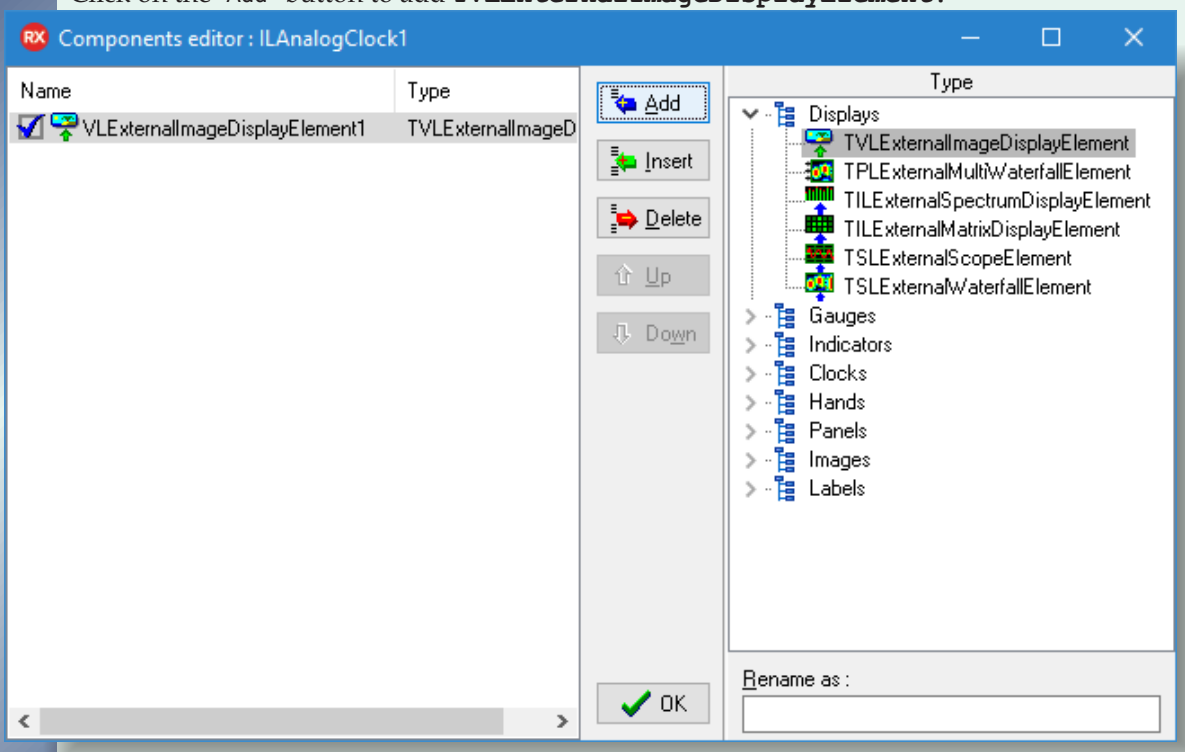


In the Clock, similar to the rest of the **InstrumentLab** controls you can add many different types of elements, such as nested gauges, Thermometers, clocks, LEDs, and even VIDEO DISPLAYS.
To display the video in the Clock, we will add a **Video Display** element.

Double click on the **ILAnalogClock1** component to open the Elements Editor.
In the **Elements Editor**, expand the Displays category in the right view, and select **`TVLExternalImageDisplayElement`.**

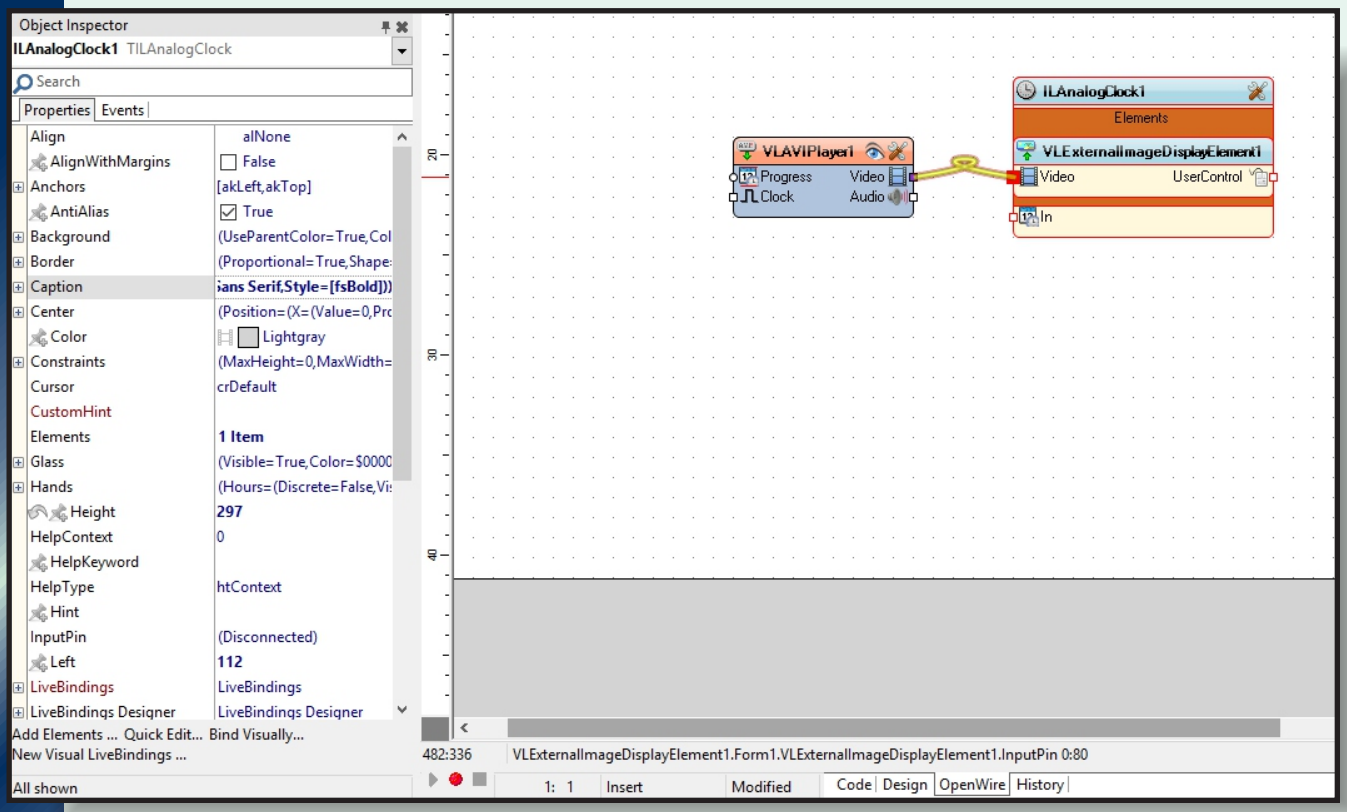


Click on the "Add" button to add **`TVLExternalImageDisplayElement:`**

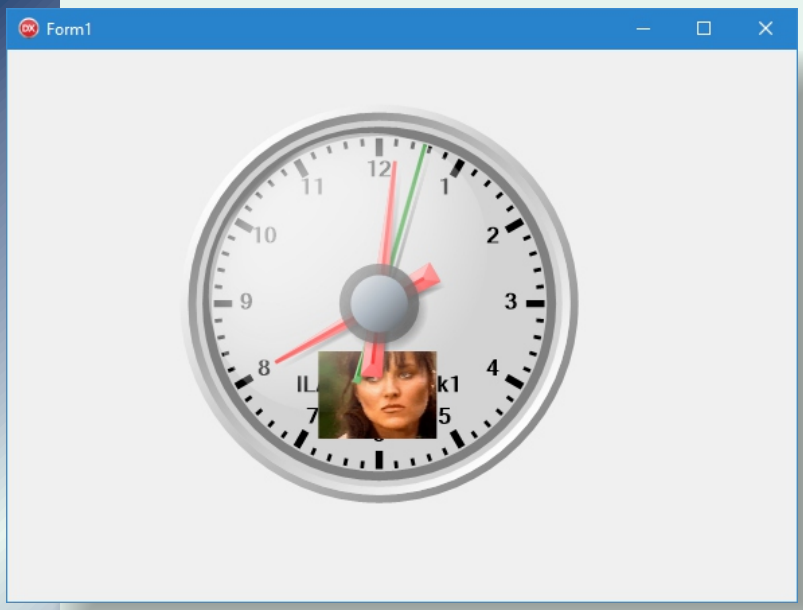


Close the **Elements Editor Dialog.**

Switch to the "Open Wire" tab, and connect the "Video" Output Pin of the **VLAVIPlayer1** to the "Video" Input Pin of the **VLExternalImageDisplayElement1** of the **ILAnalogClock1** component:



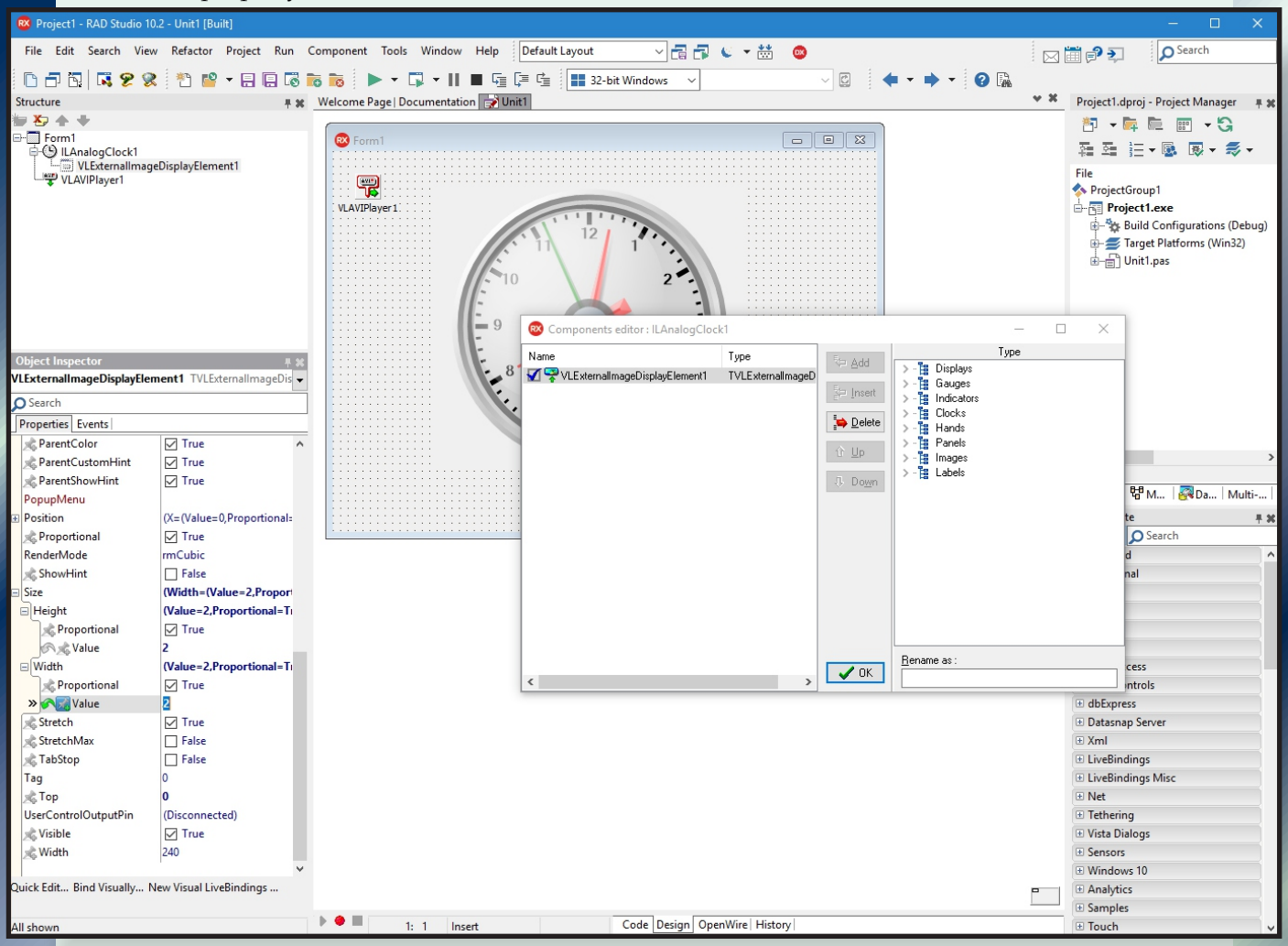
Compile and run the application. You should see the video playing in the small display inside the Clock:



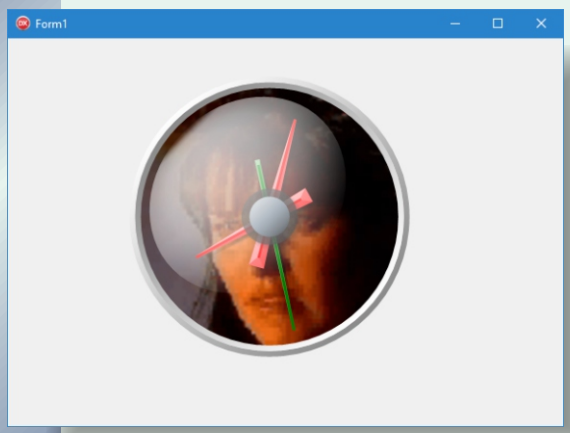
The Video plays in a small display area in the Clock.
We can change the size and position of this area to have the video displayed wherever we want.
As example, we can expand the display to take the entire clock background.

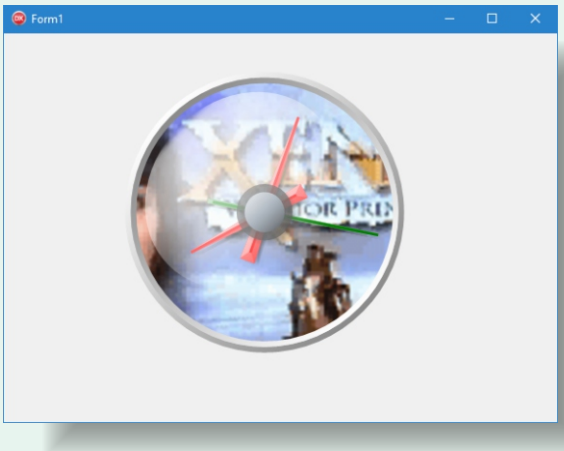
Close the application.

In **Delphi** switch to the **Form Designer**, and double click on the **ILAnalogClock1** component to open the **Elements Editor**. In the **Elements Editor**, select the **VLExternalImageDisplayElement1** in the left view. In the **Object Inspector**, expand the "Size" property, then the "Height" sub-property, and set the "Value" sub-property of "Height" to "2". In the **Object Inspector**, expand the "Width" sub-property, and set the "Value" sub-property of "Width" to "2":



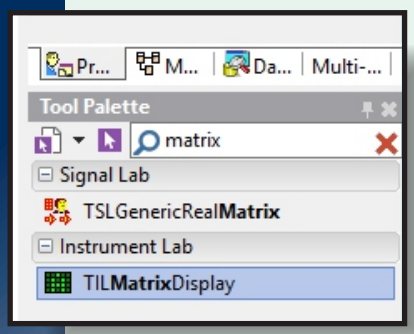
Close the **Elements Editor Dialog**. Compile and run the application. You should see the video playing in the full background of the Clock:



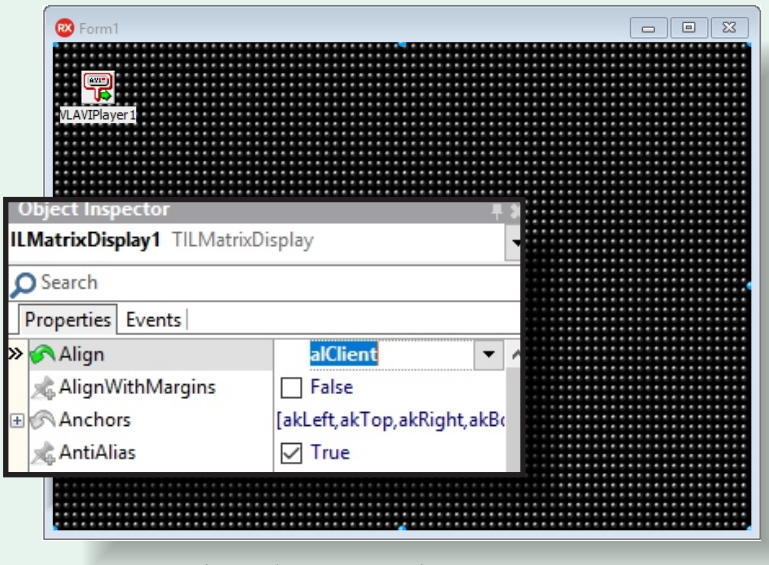


The Video can be rendered on any of the Instruments of **Instrumentlab** as a video layer. **InstrumentLab** however also contains a **LED Matrix** component that can display the video using the LEDs.

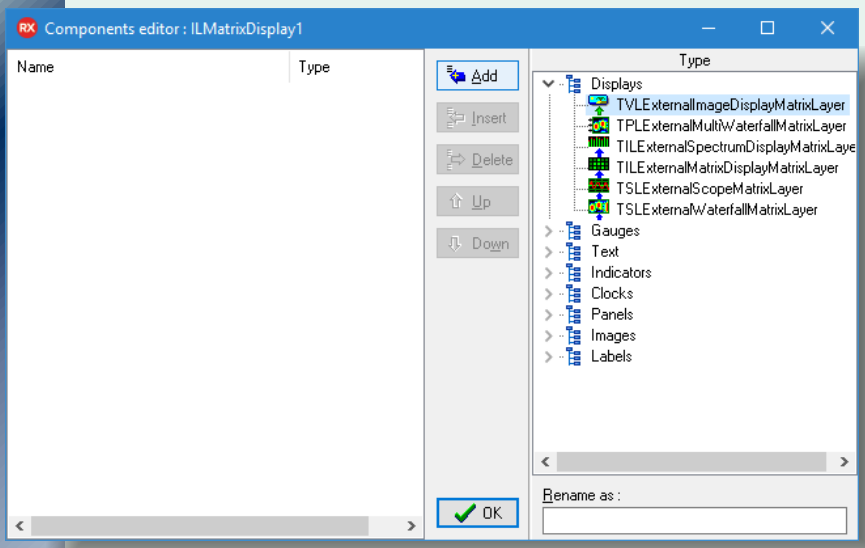
Close the application. In **Delphi,** switch to the **Form Designer**. Remove the **ILAnalogClock1** component. Type "matrix" in the Tool Palette search box, then select **`TILMatrixDisplay`** from the palette:



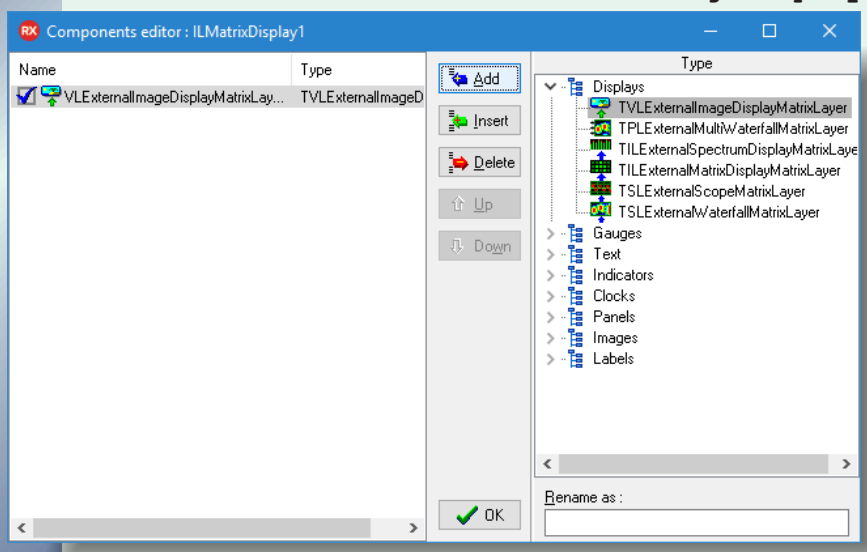
And drop it on the form.
In the **Object Inspector** set the value of the "Align" property to "alClient":



Double click on the **ILMatrixDisplay1** component to open the **Elements Editor**.
In the **Elements Editor**, expand the **Displays** category in the right view, and select **`TVLExternalImageDisplayMatrixLayer`**

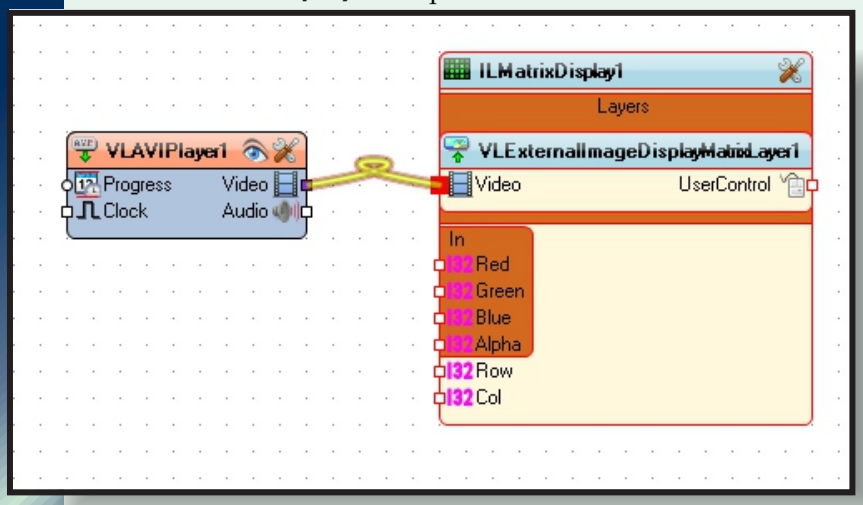


Click on the "Add" button to add **`TVLExternalImageDisplayMatrixLayer:`**

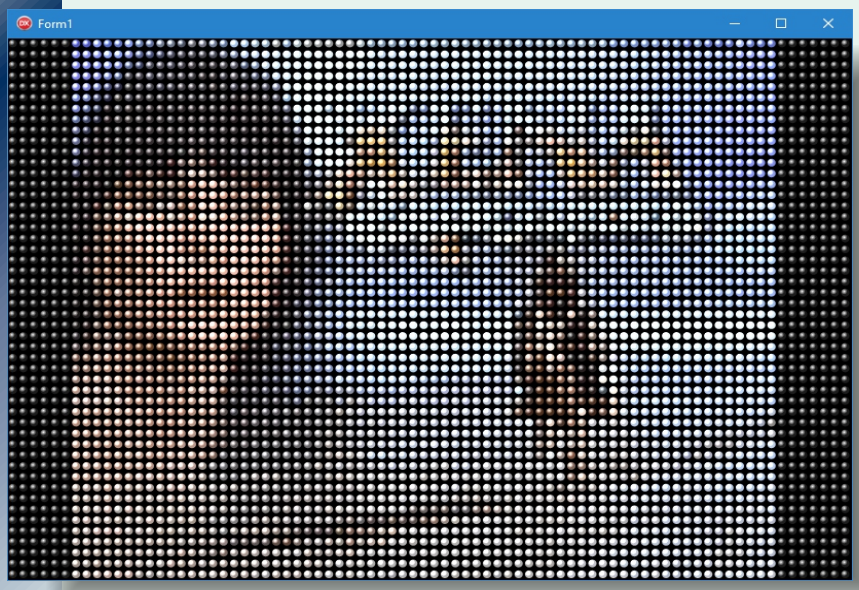


Close the **Elements Editor Dialog.**

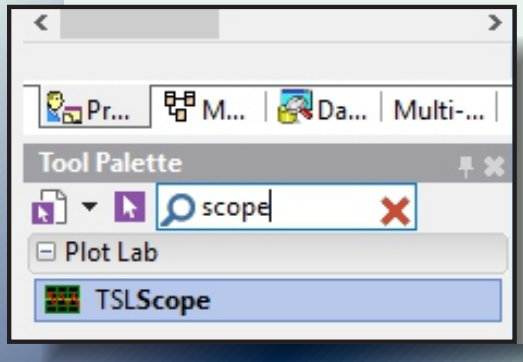
Switch to the "Open Wire" tab, and connect the "Video" Output Pin of the VLAVIPlayer1 to the "Video" Input Pin of the **VLExternalImageDisplayMatrixLayer1** of the **ILMatrixDisplay1** component:

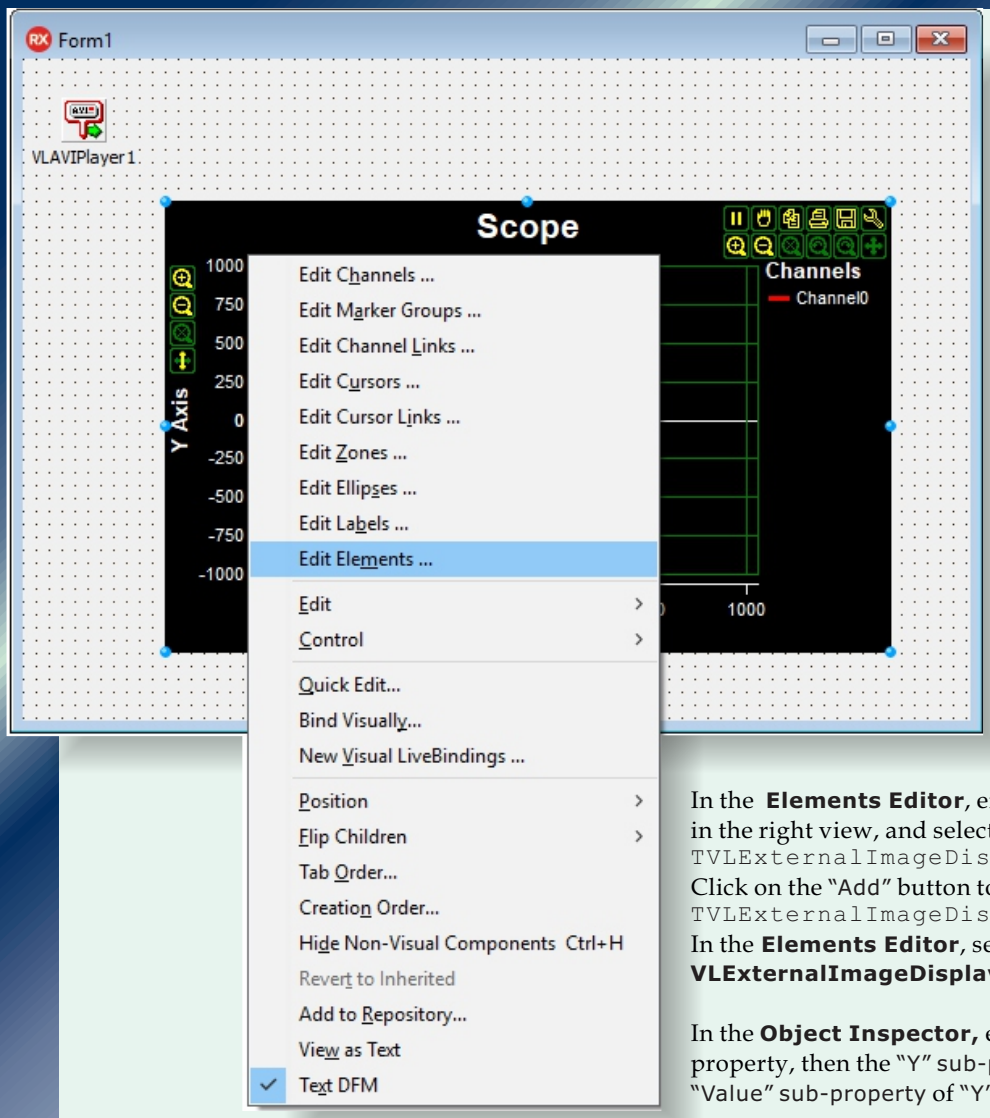


Compile and run the application. You should see the video playing in the **LED Matrix Display** rendered with the individual LEDs:



Video can be rendered not only in the **InstrumentLab** component but also in the **Scope** and **Waterfall** components. Close the application. In **Delphi,** switch to the **Form Designer.** Remove the **ILMatrixDisplay1** component. Type "scope" in the Tool Palette search box, then select **`TSLScope`** from the palette: And drop it on the form.

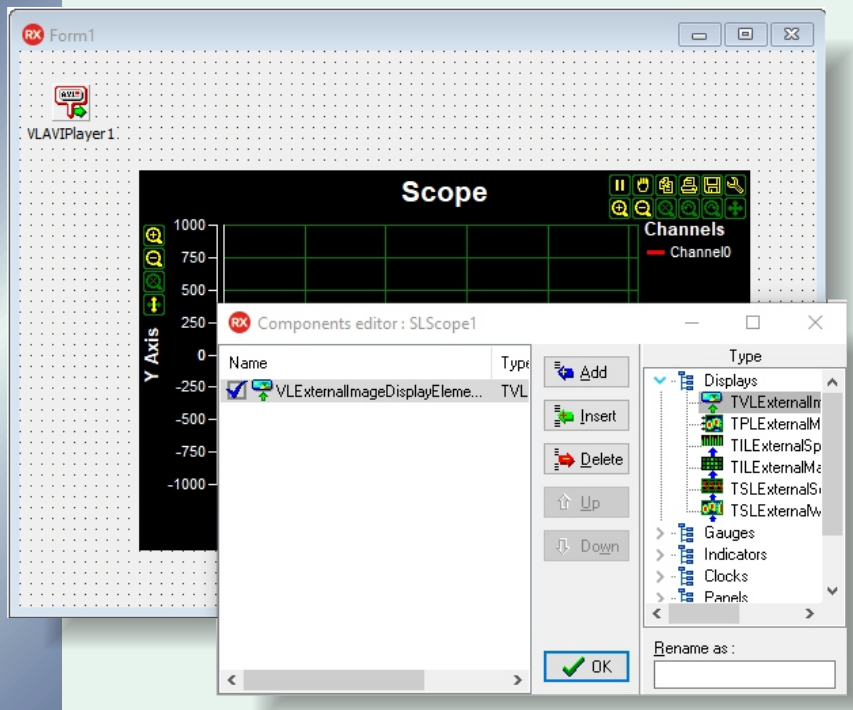


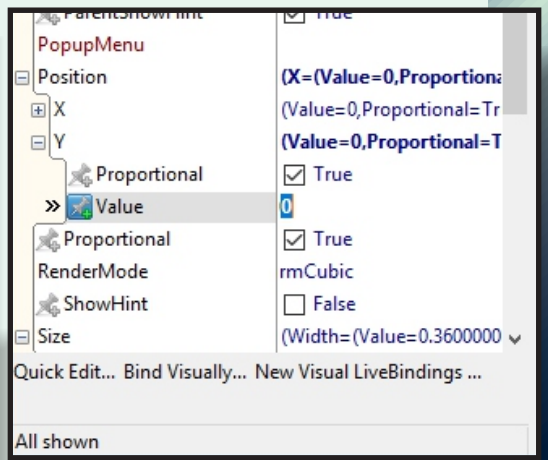


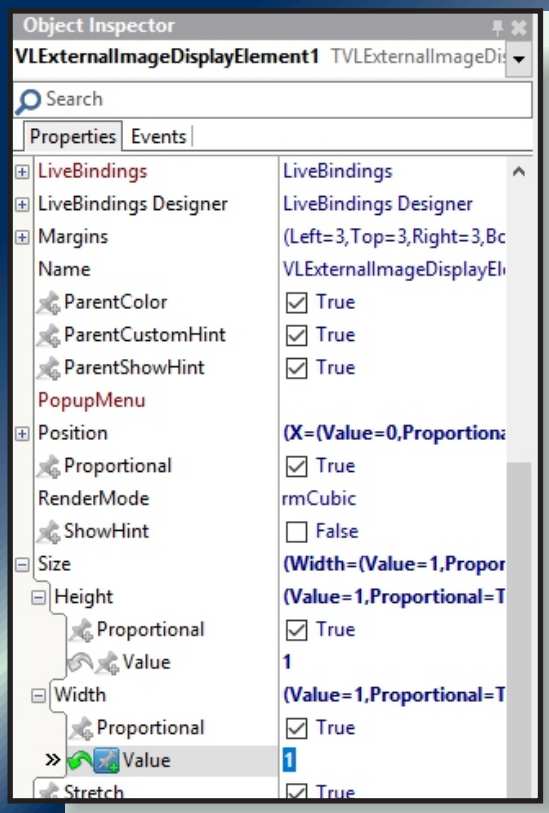
Right click on the **SLScope1** component, and from the **Pop-up Menu** select "Edit Elements..." to open the **Elements Editor:**

In the **Elements Editor**, expand the **Displays category** in the right view, and select `TVLExternalImageDisplayElement`.
Click on the "Add" button to add `TVLExternalImageDisplayElement`.
In the **Elements Editor**, select the newly added **VLExternalImageDisplayElement1** in the left view.

In the **Object Inspector,** expand the "Position" property, then the "Y" sub-property, and set the "Value" sub-property of "Y" to "0":

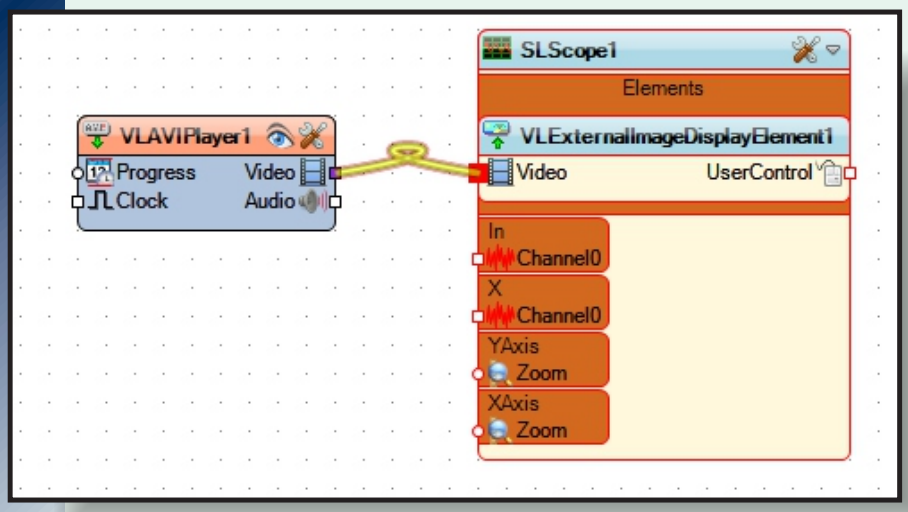




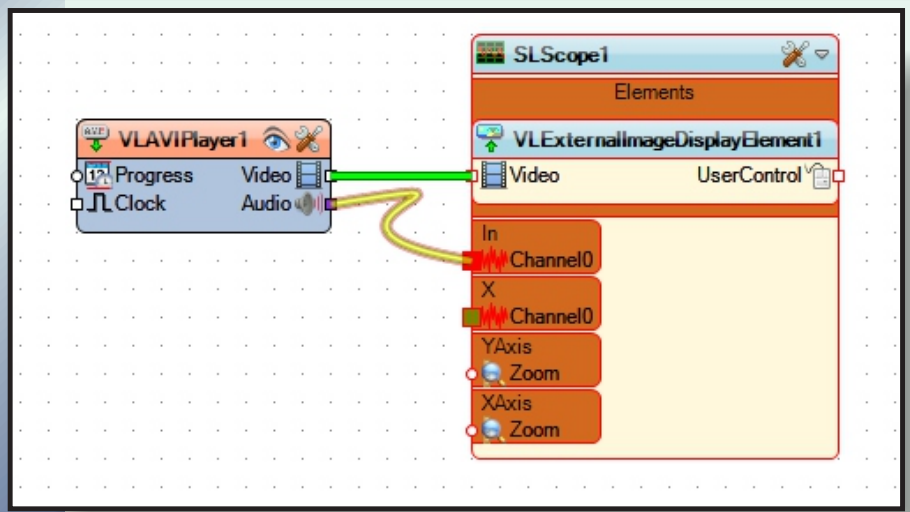


In the **Object Inspector,** expand the "Size" property, then the "Height" sub-property, and set the "Value" sub-property of "Height" to "1".
In the Object Inspector, expand the "Width" sub-property, and set the "Value" sub-property of "Width" to "1":

Close the **Elements Editor Dialog.**



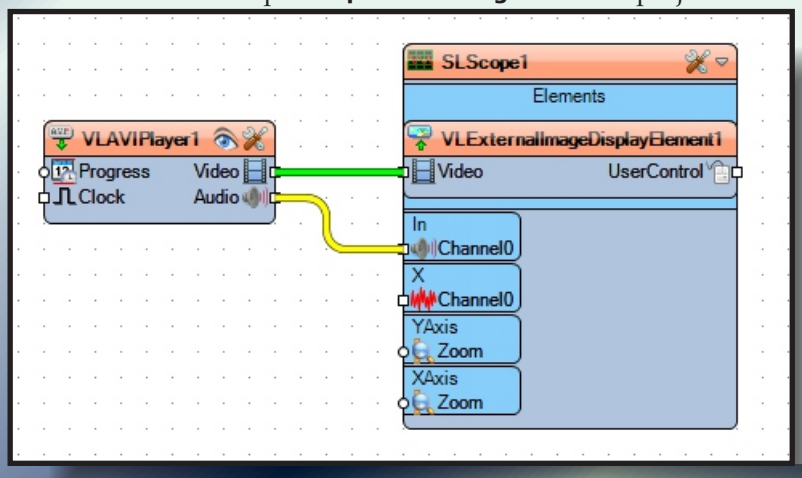
Switch to the "Open Wire" tab, and connect the "Video" Output Pin of the **VLAVIPlayer1** to the "Video" Input Pin of the **VLExternalImageDisplayElement 1** of the **SLScope1** component:



We need some data to be displayed in the **SLScope1** component. The simplest option is to display the **Audio** from the player.
Connect the "Audio" Output Pin of the **VLAVIPlayer1** to the "In" Input Pin of the Channel0 of the **SLScope1** component:

Compile and run the application. You should see the video playing in the full background of the Scope:

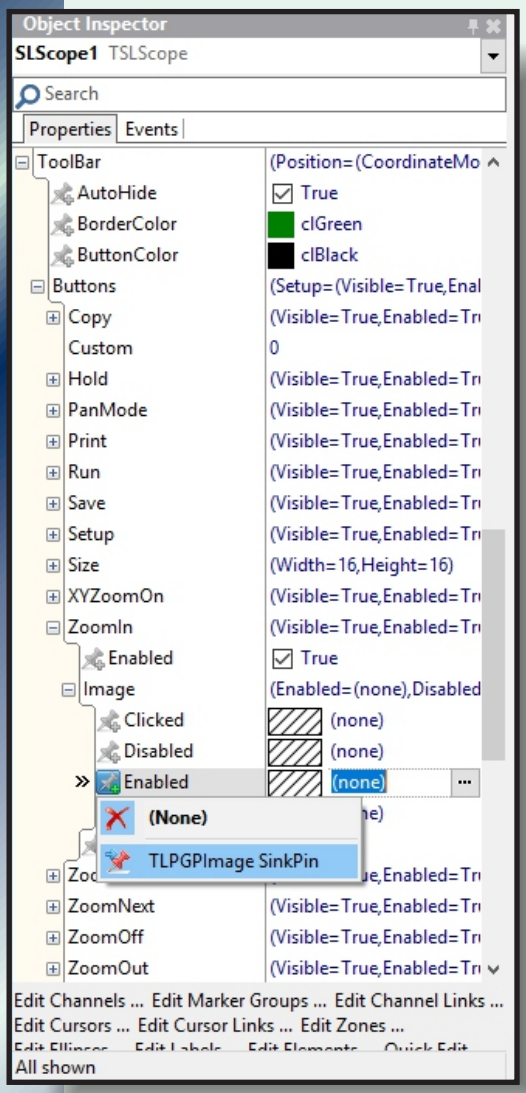
Here is the complete **OpenWire Diagram** of the project:

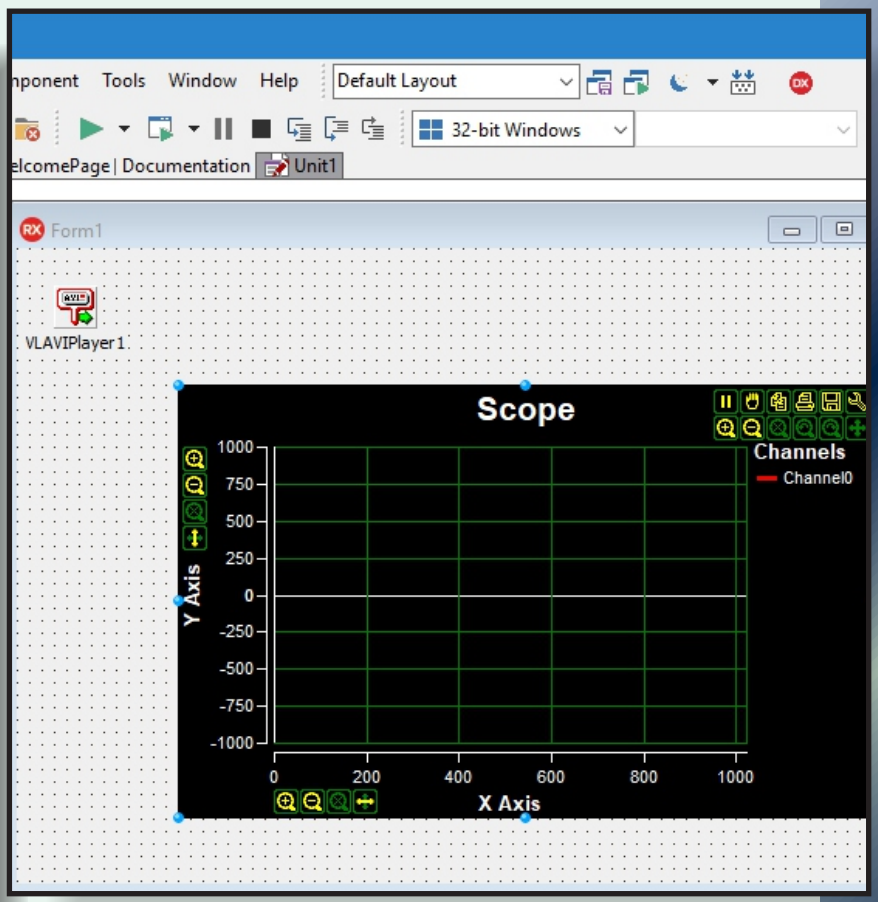


Video can be rendered not only as background of the scope,
but also as images of the toolbar buttons, by using **OpenWire**
Visual Live Binding to add Pins to them.

Close the application.
For simplicity, we will start with new Scope component.
In **Delphi,** switch to the **Form Designer.**
Remove the **SLScope1** component.
Type "scope" in the **Tool Palette** search box,
then select **`TSLScope`** from the palette, and drop it on the form.
In the **Object Inspector Expand** the "Toolbar" property,
then the "Buttons" sub-property, then the "ZoomIn"
sub-property, and finally the "Image" sub-property.

Click on the  button at front of the "Enabled" property.
From the drop down menu, select "**`TLPGImage SinkPin`**":



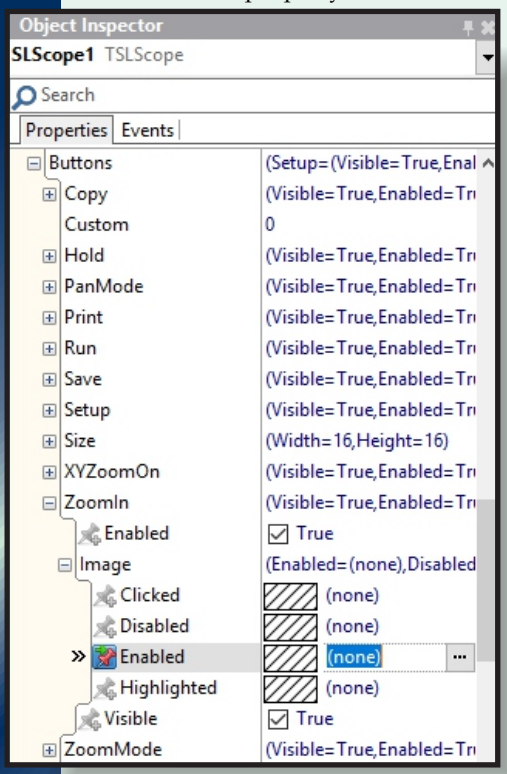


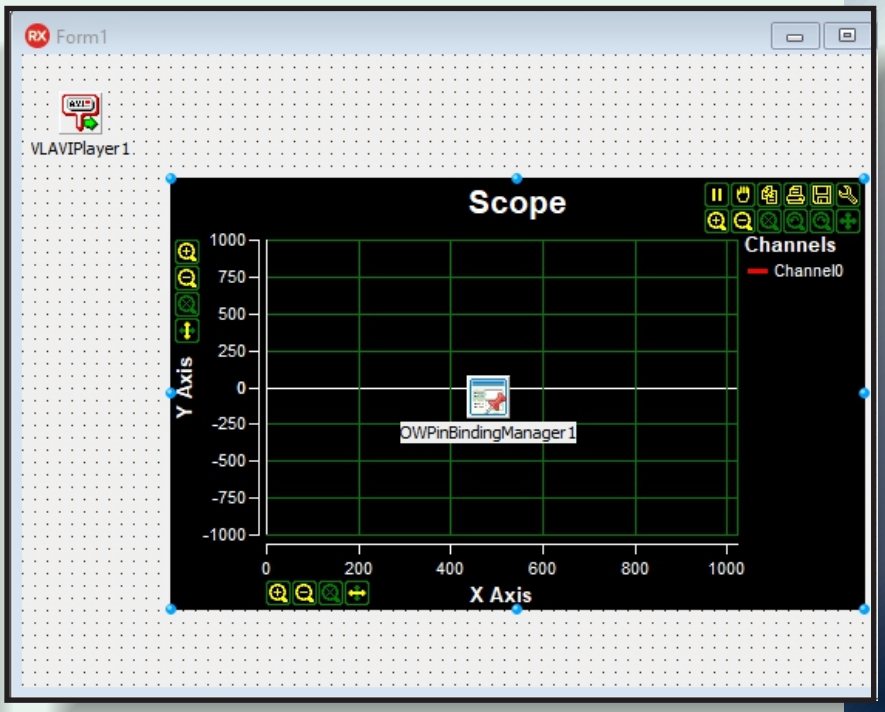
REMEMBER:
YOU CAN DOWNLOAD THE TRIAL VERSION.

IT'S FULLY FUNCTIONAL .
IT'S FREE.

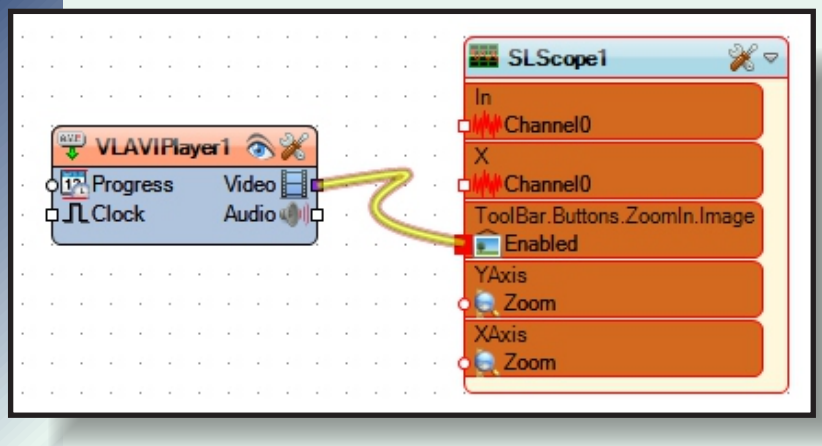
AND HAS NO LIMITATIONS.

A new **OWPinBindingManager1** component will automatically be added to the form. This component will contain the **Open Wire Visual Live Binding** that we added for the property:

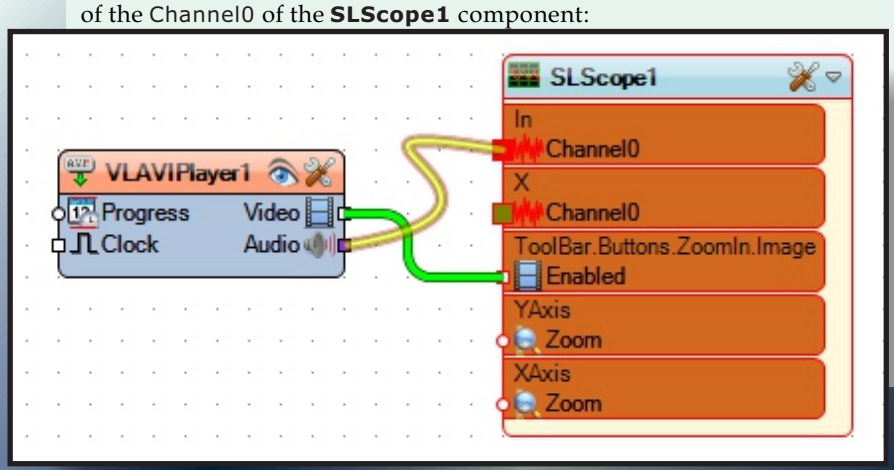




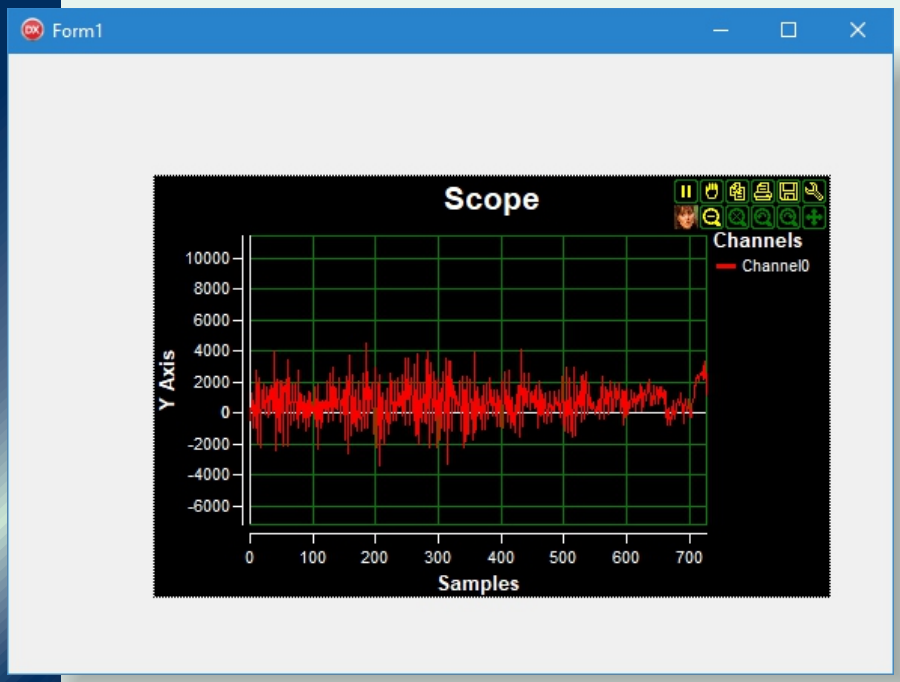
Switch to the **"Open Wire"** tab, and connect the "Video" Output Pin of the **VLAVIPlayer1** to the "Enabled" Input Pin of the "Toolbar.Buttons.ZoomIn.Image" of the **SLScope1** component:

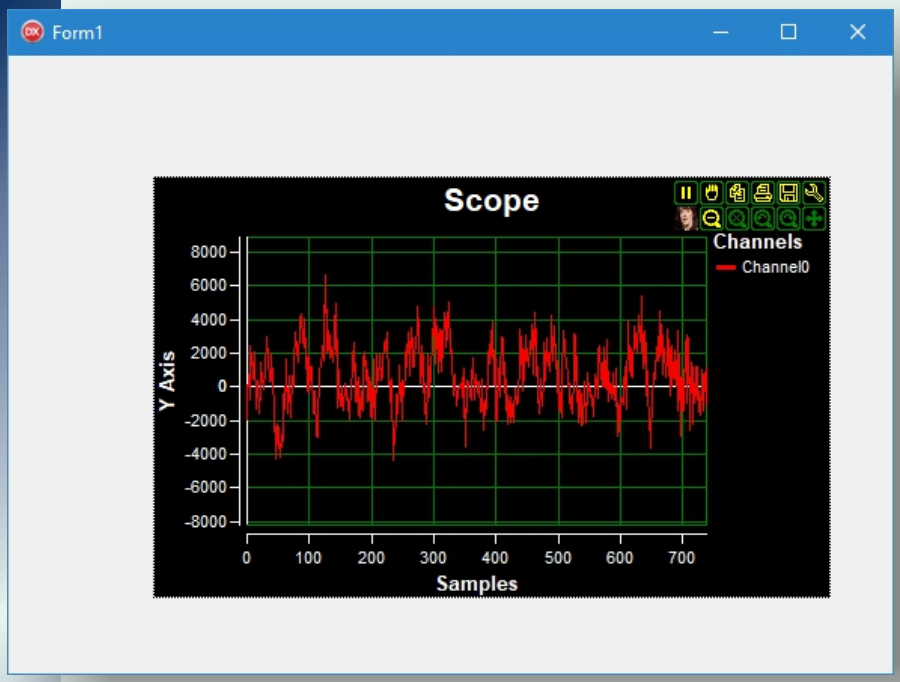


Connect the "Audio" Output Pin of the **VLAVIPLAYER1** to the "In" Input Pin of the Channel0 of the **SLScope1** component:

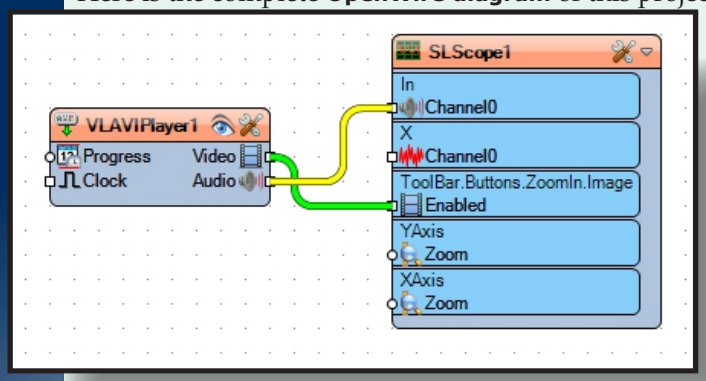


Compile and run the application.
If you move your mouse over the scope to show the Toolbar,
you will see that the **ZoomIn** button
(The first on the second row) will display the video:

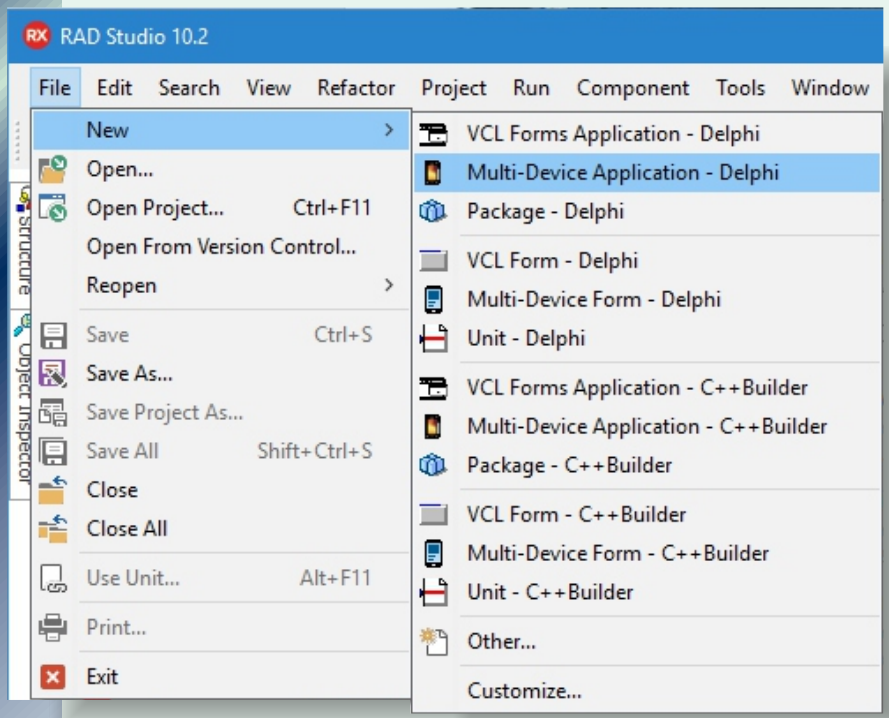




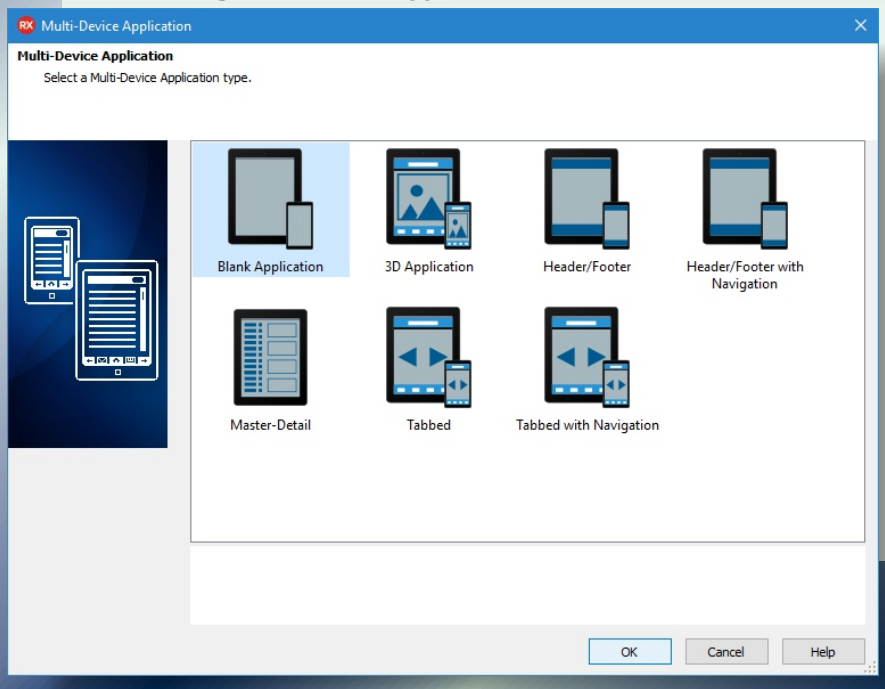
Here is the complete **OpenWire diagram** of this project:



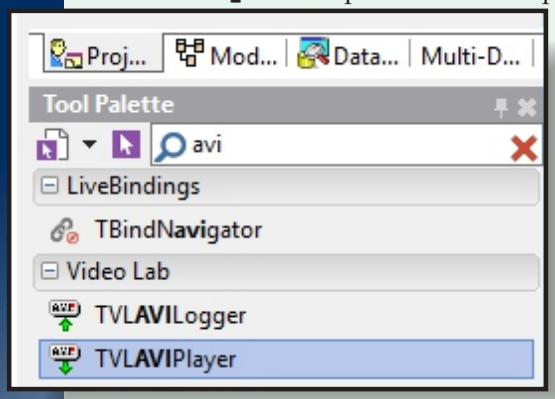
The same way of using **Visual Live Binding** that we used to show the video in the button, can be used to display the video on other surfaces and even 3D objects in **Fire Monkey.** Close the application. In the **Delphi** menu, select "New", then "Multi-Device Application – Delphi":



In the **Dialog,** select **Blank Application**, and click the **OK** Button:

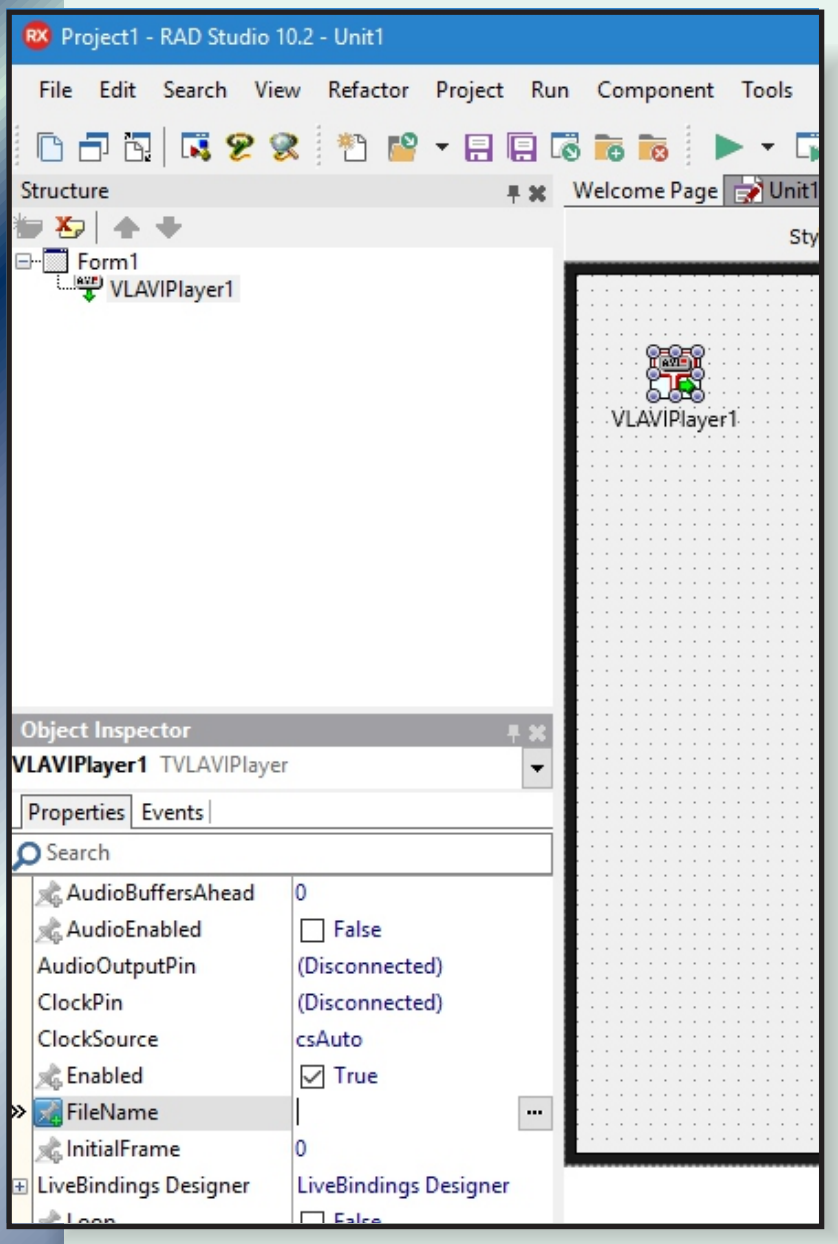


Type "avi" in the **Tool Palette search box**, then select the **`TVLAVIPlayer`** component from the palette:

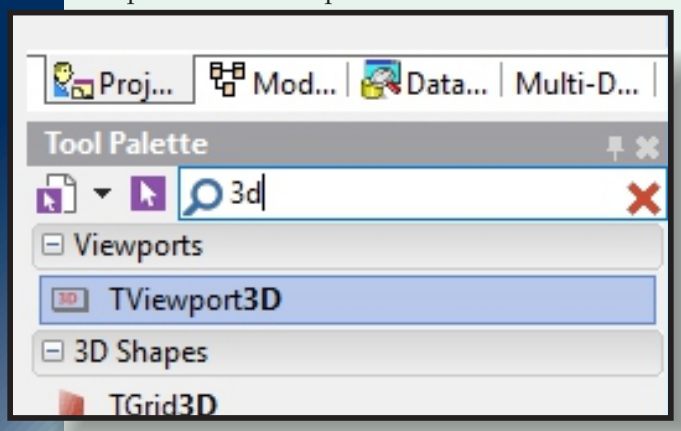


And drop on the form

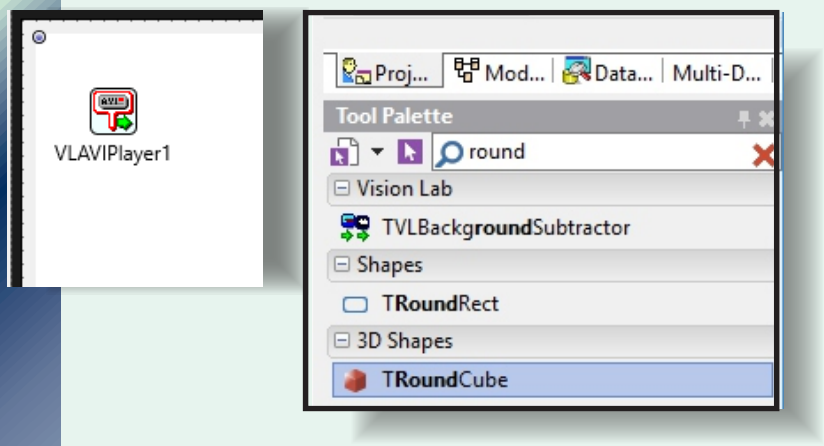
In the **Object Inspector** select the "FileName" property, and click on the "..." *(Ellipsis)* ··· button:



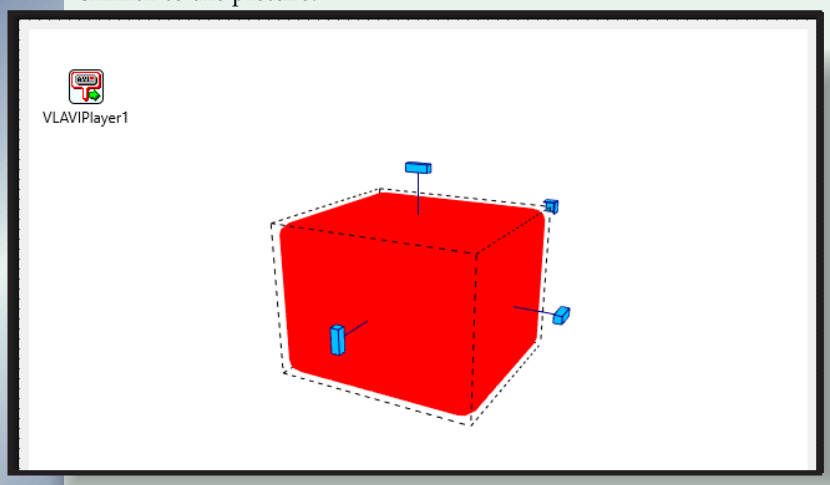
And select file to play. Type "3D" in the **Tool Palette** search box, then select **`TViewport3D`** component from the palette:



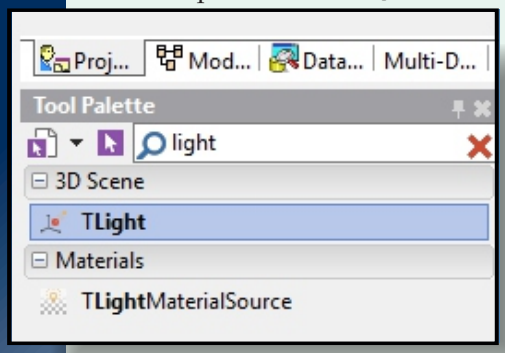
And drop it on the form. Type "round" in the Tool Palette search box, then select **`TRoundCube`** component from the palette:

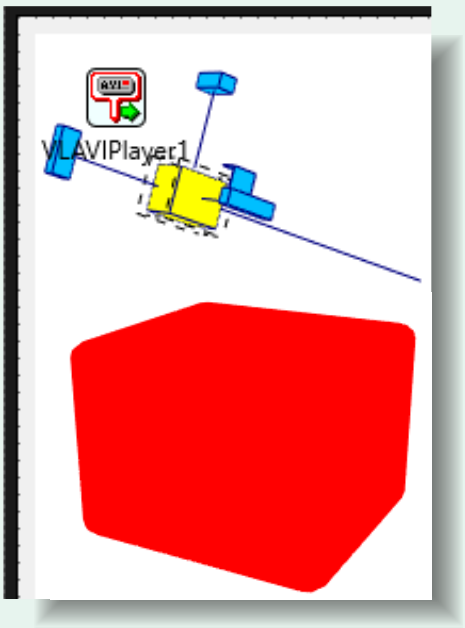


And drop it in the **Viewport3D1,** then rotate and resize it similar to the picture.

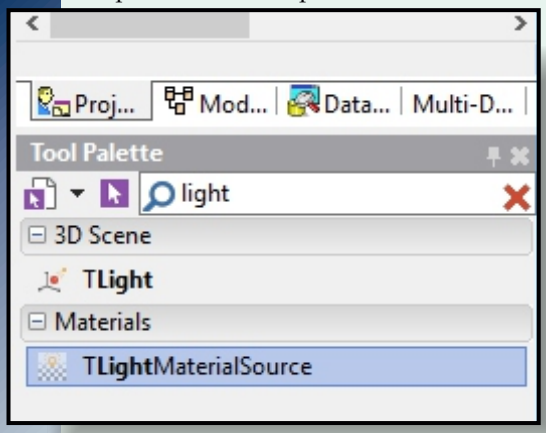


Type "light" in the **Tool Palette search box**, then select **`TLight`** component from the palette:
And drop it in the **Viewport3D1,** then move and rotate it similar to the picture:

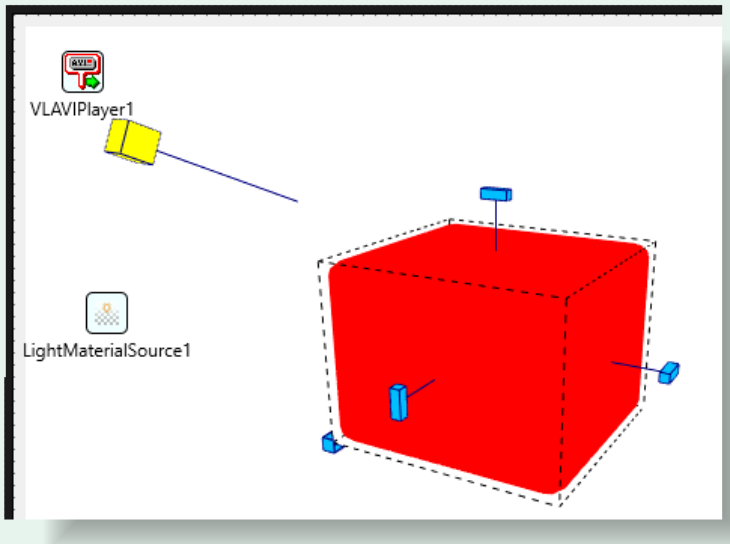


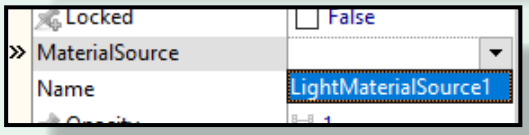


Type "light" in the **Tool Palette** search box, then select **`TLightMaterialSource`** component from the palette:

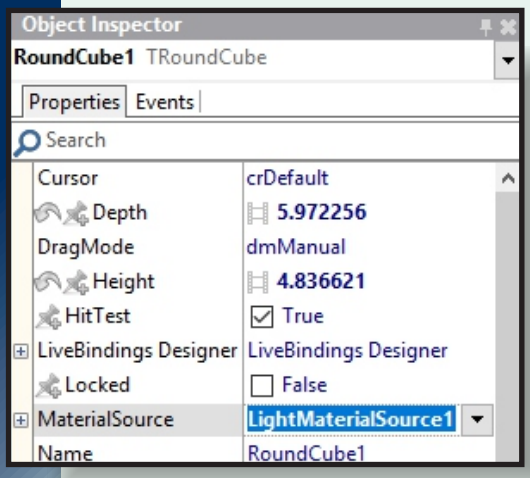


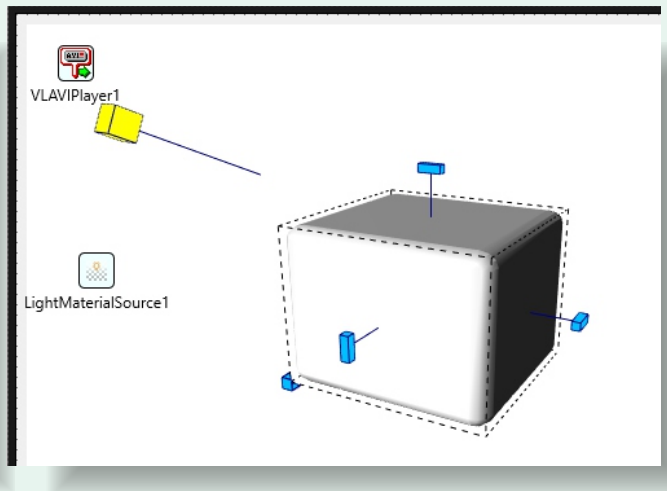
And drop it on the form. Select the **RoundCube1** component. In the **Object Inspector** click on the "Drop Down" button of the "MaterialSource" property:



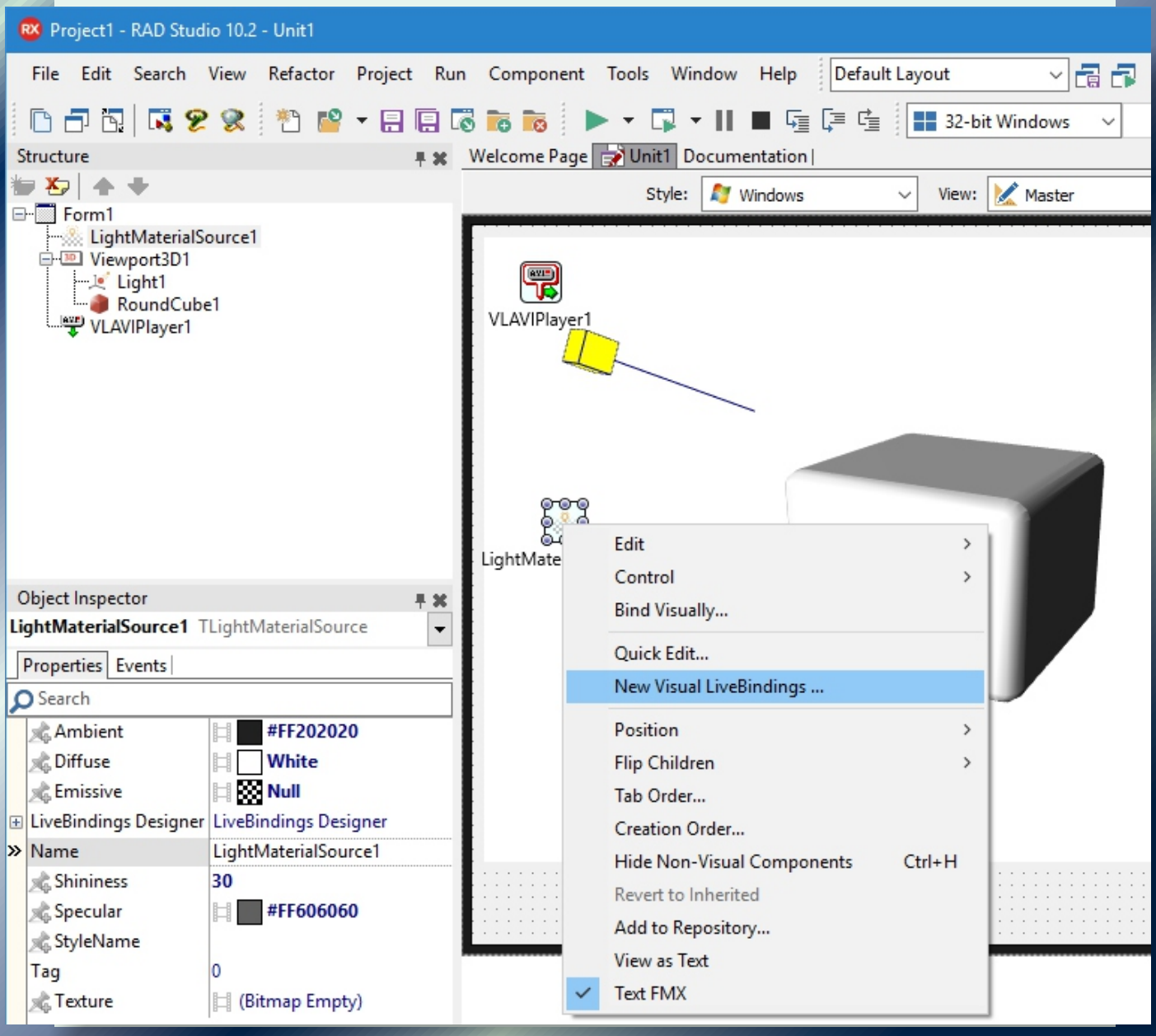


And select **LightMaterialSource1:**

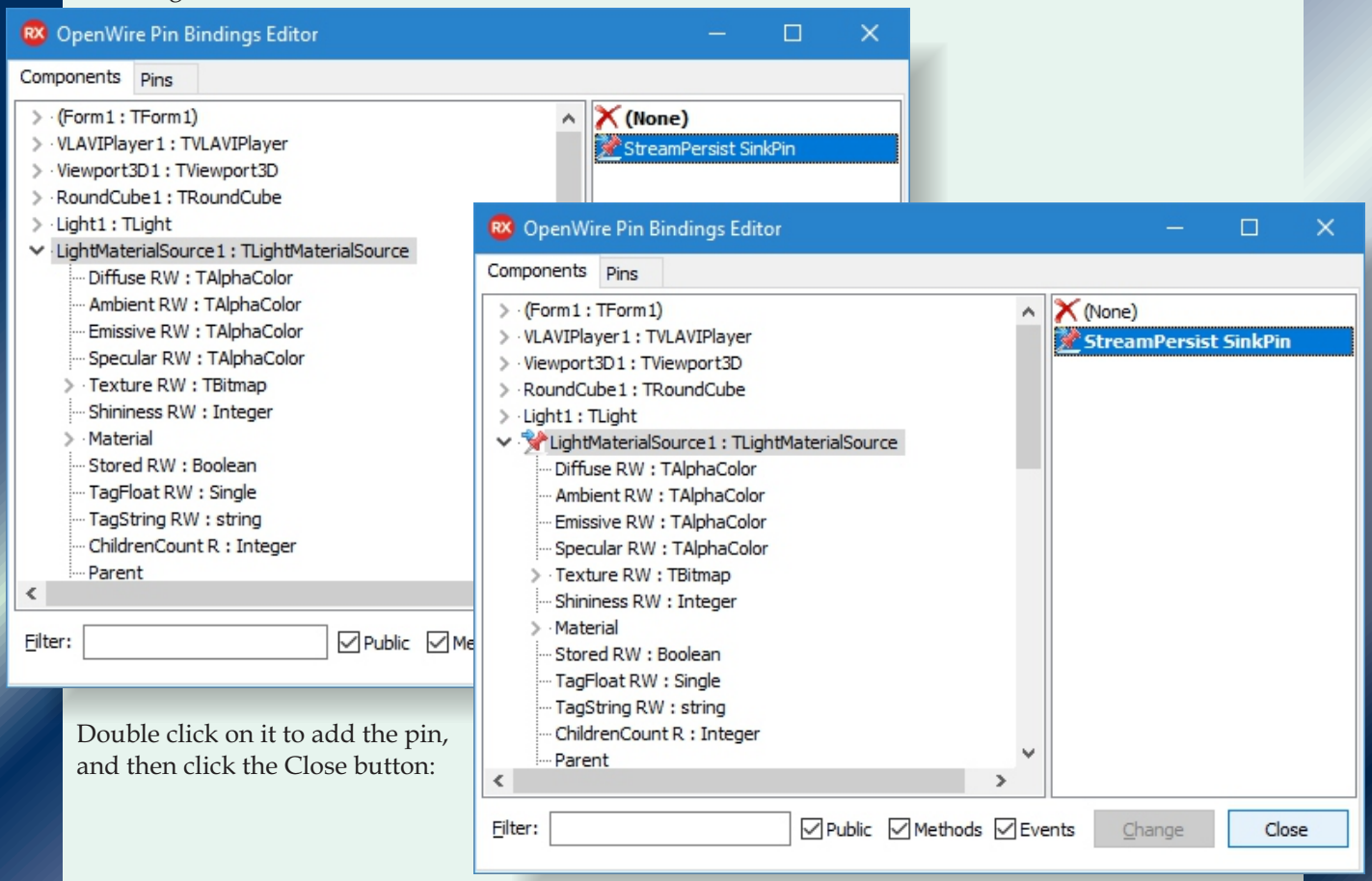




Right-Click on the **LightMaterialSource1** component.
From the Pop-up Menu, select "New Visual LiveBinding...":

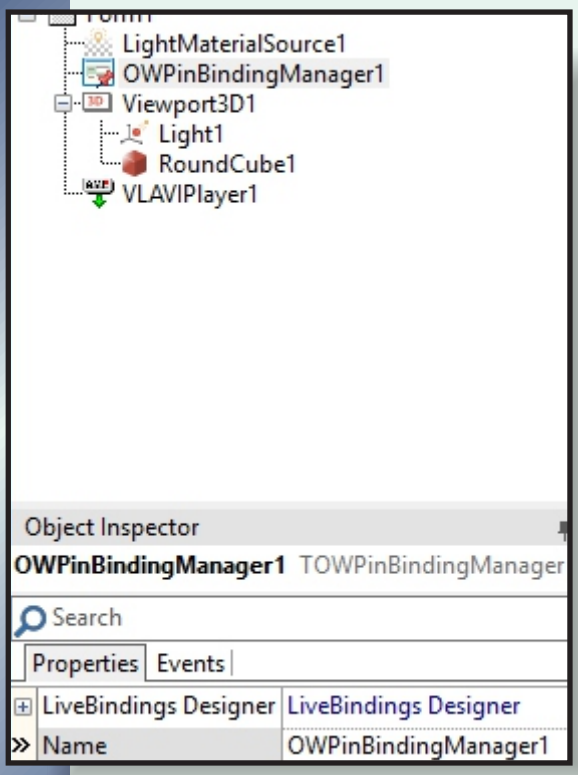


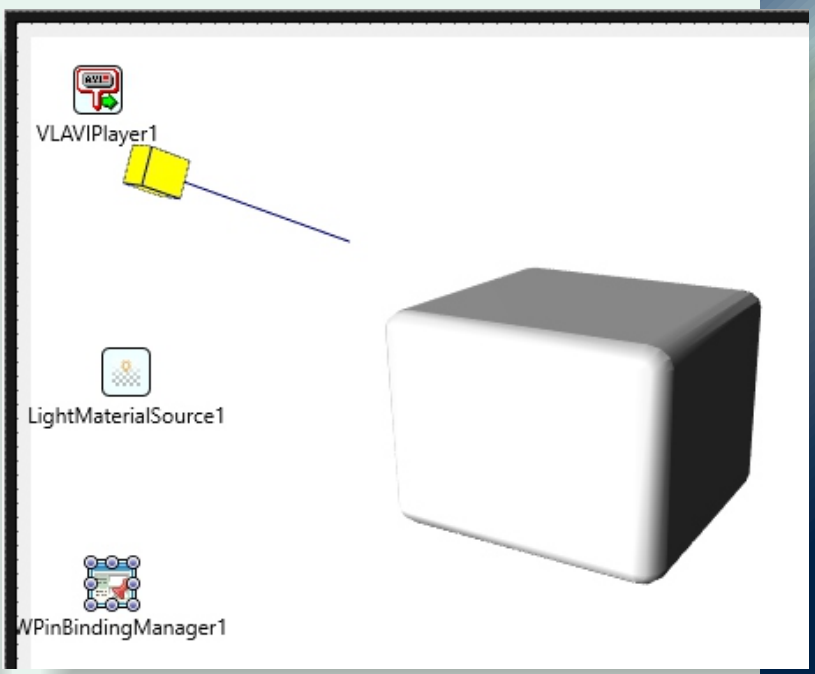
In the right view of the "OpenWire Pin Binding Editor", select "StreamPersist SinkPin":



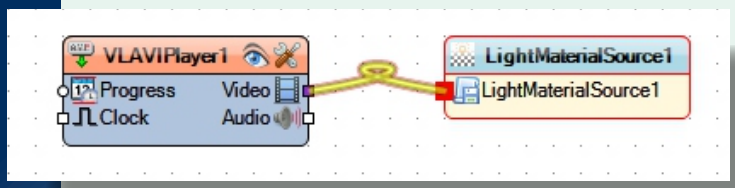
Double click on it to add the pin, and then click the Close button:

A new **OWPinBindingManager1** component will automatically be added to the form. This component will contain the **Open Wire Visual Live Binding** that we added for the property:

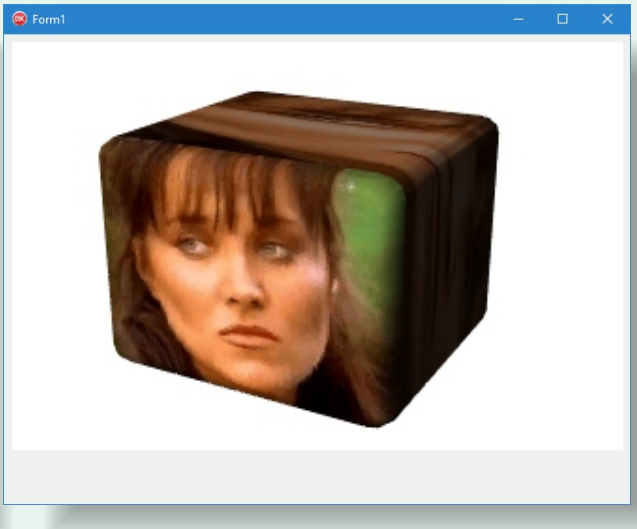




Switch to the "Open Wire" tab, and connect the "Video" Output Pin of the **VLAVIPlayer1** to the "LightMaterialSource1" Input Pin of the **LightMaterialSource1** component:

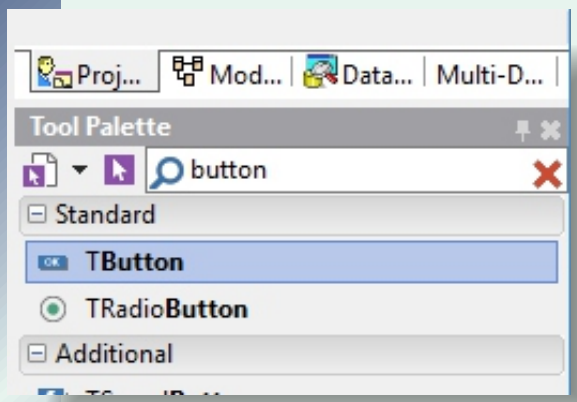


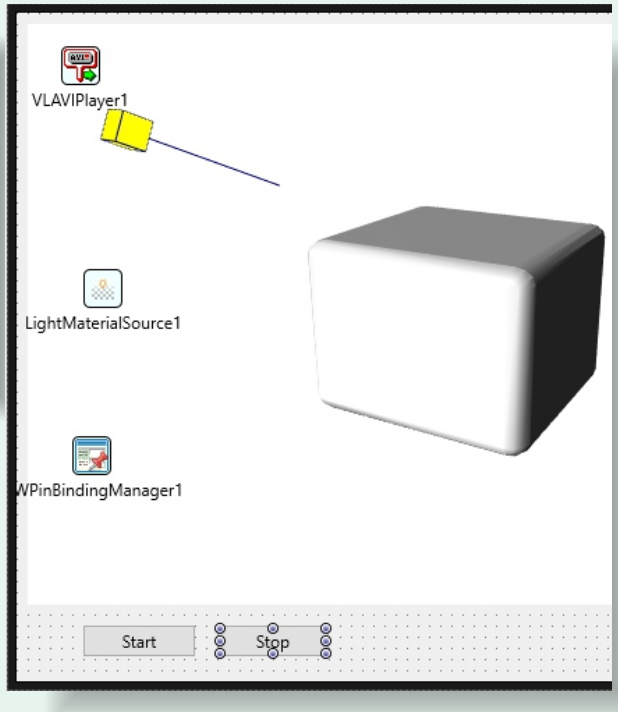
Compile and run the application. You should see the video playing on the surface of the Cube:



In addition to playing the video on the surface of the cube, we can also use **OpenWire Visual Live Bindings** to rotate the cube as the video progresses, and also to Start, Stop, Pause and Resume the video.
Close the application.

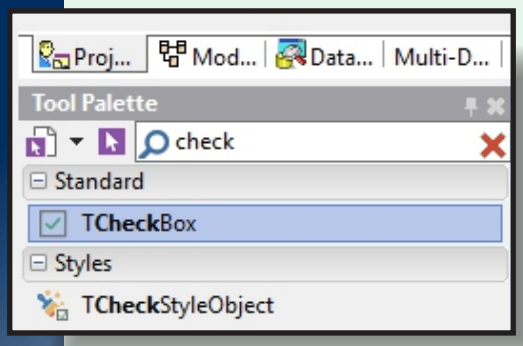
In **Delphi,** switch to the **Form Designer**.
Type "button" in the **Tool Palette search box**, then select `TButton` from the palette:



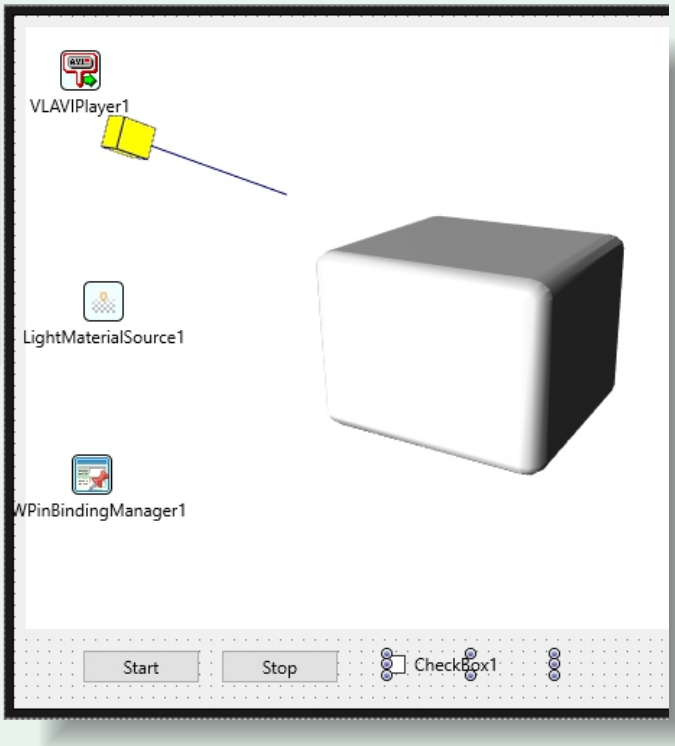


And add 2 of them on the Form.
In the **Object Inspector,** set the "Text" property of the first one to "Start".
In the **Object Inspector,** set the "Text" property of the second one to "Stop".

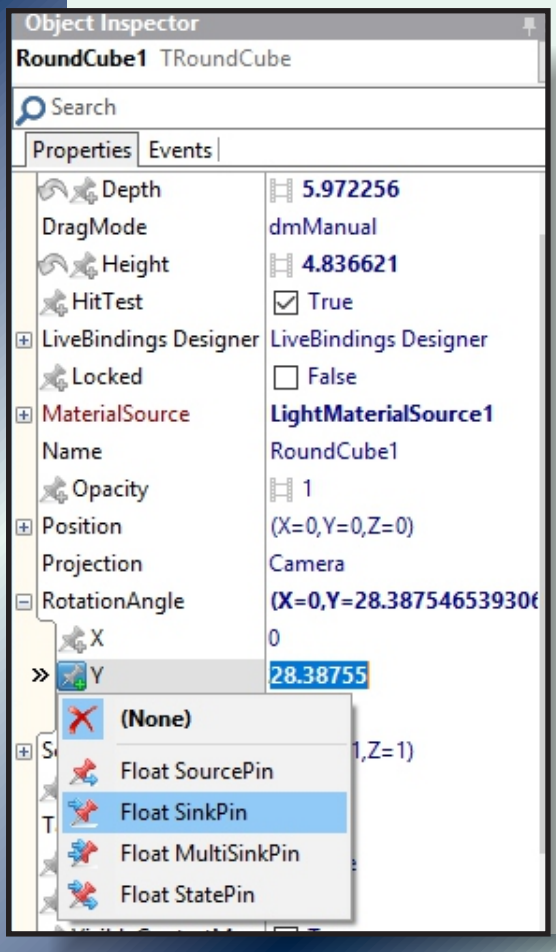
Type "check" in the **Tool Palette** search box, then select **`TCheckBox`** from the palette:

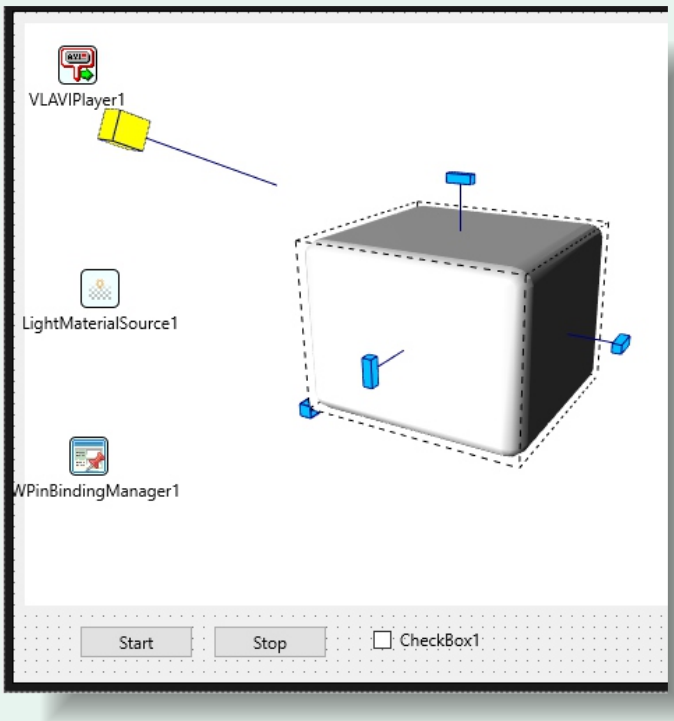


And drop it on the form:

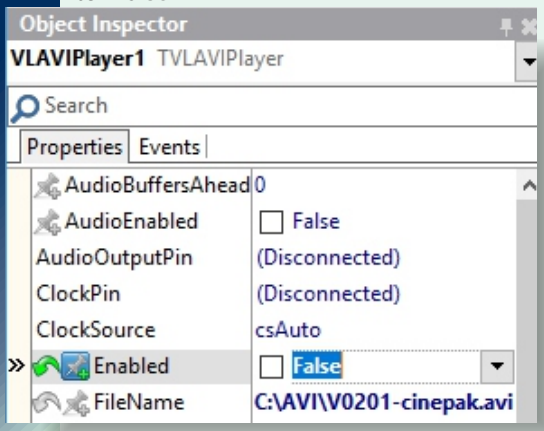


First we will add Pin to rotate the **RoundCube1.** Select the **RoundCube1.** In the **Object Inspector** expand the "RotationAngle" properly. Click on the  button at front of the "Y" sub-property. From the drop down menu, select "Float SinkPin":

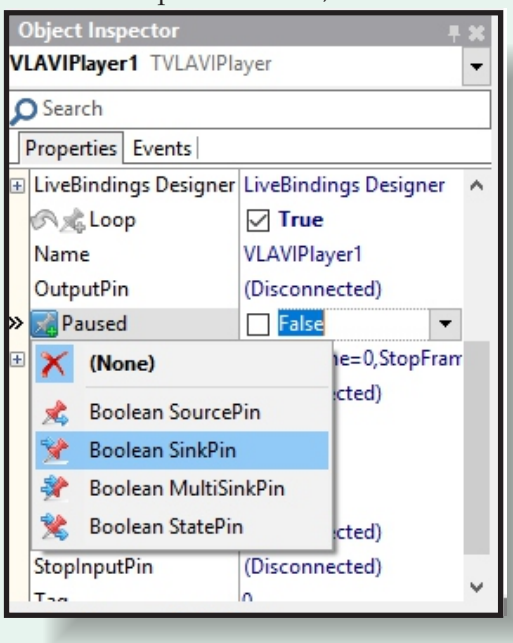




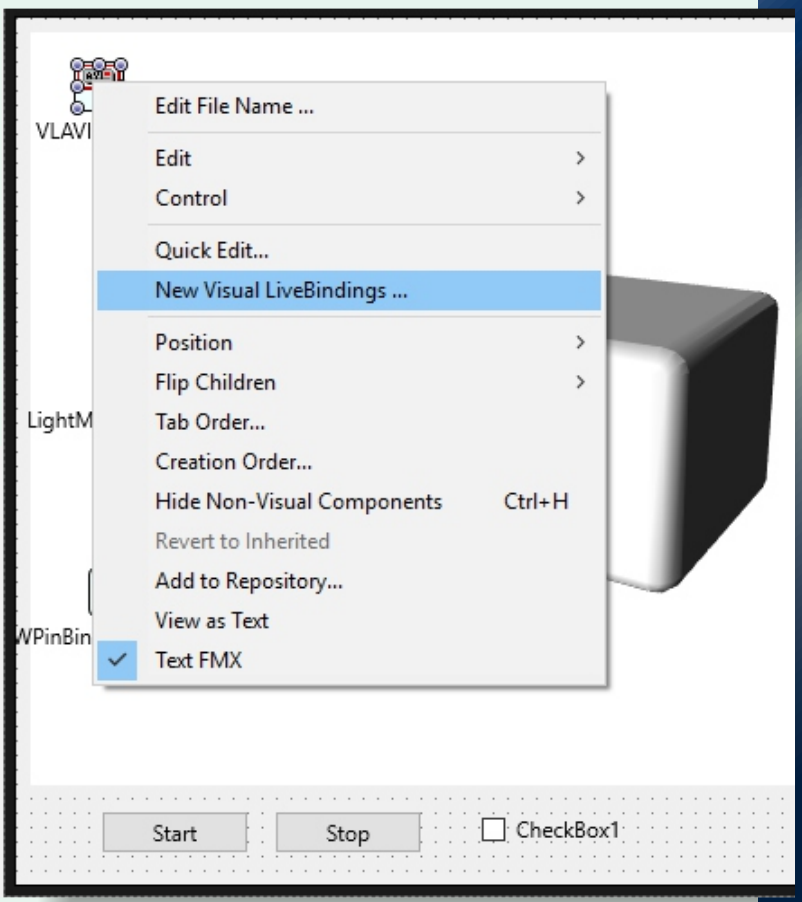
Next we will set the **VLAVIPlayer1** to be initially disabled. Select the **VLAVIPlayer1.** In the **Object Inspector** set the value of the "Enabled" property to "False":



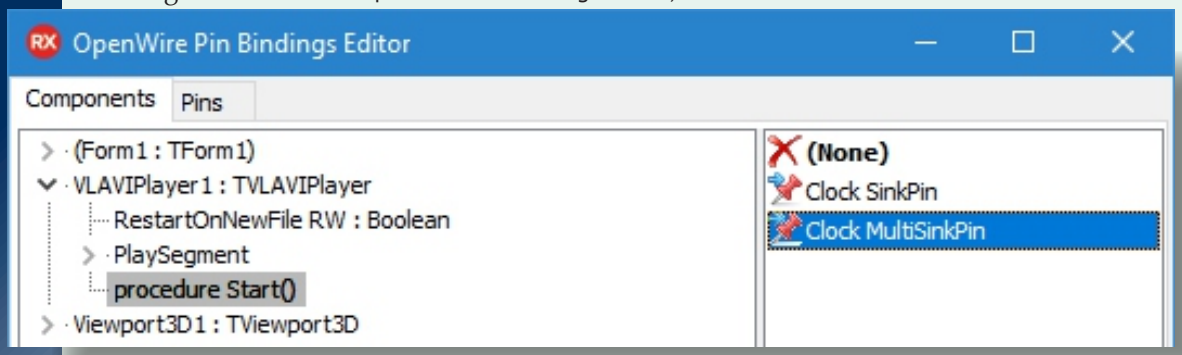
Next we will add Pins to control the **VLAVIPlayer1.** In the **Object Inspector** click on the button at front of the "Paused" sub-property.
From the drop down menu, select "Boolean SinkPin":



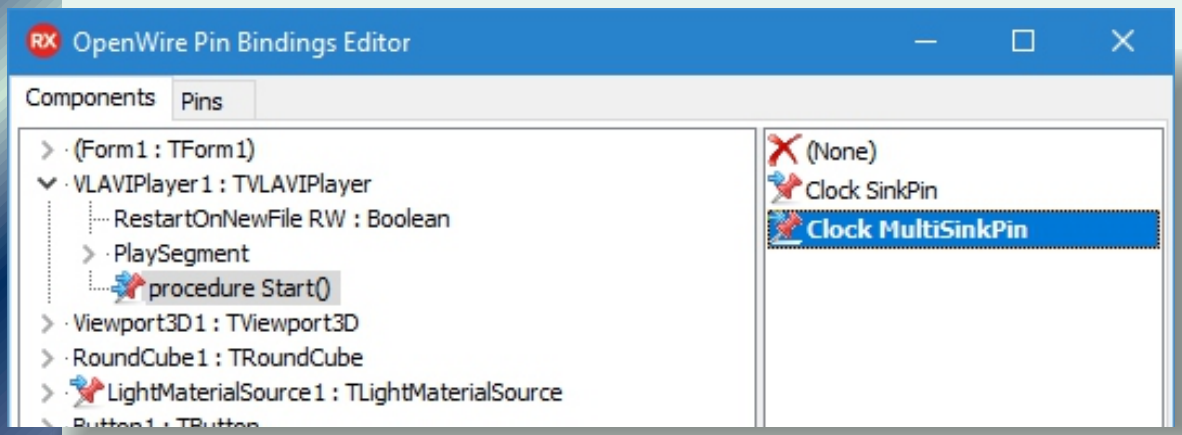
Right-Click on the VLAVIPlayer1 component. From the Pop-up Menu, select "New Visual LiveBinding...":



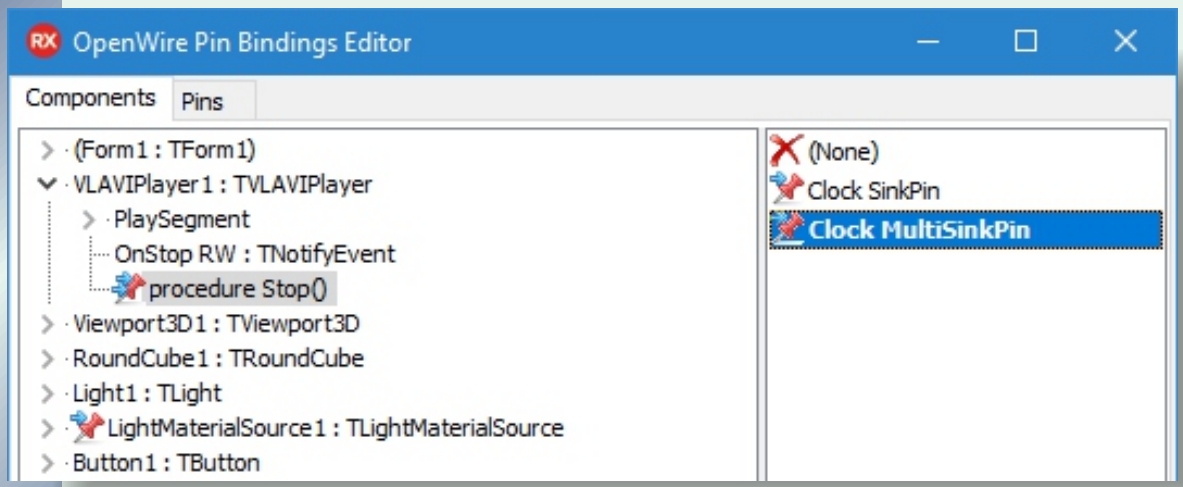
Type "start" in the "Filter" box of the "OpenWire Pin Binding Editor".
In the left view select the "procedure Start()" of the **VLAVIPlayer1.**
In the right view of the "OpenWire Pin Binding Editor", select "Clock MultiSinkPin":



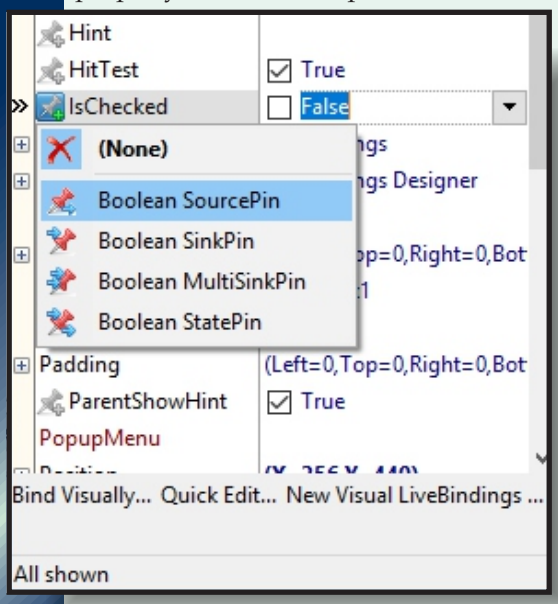
Double click on it to add the pin:

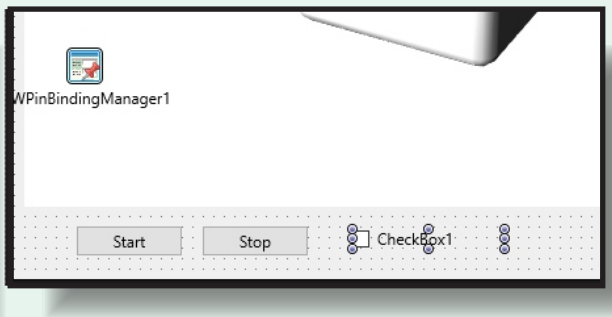


Type "stop" in the "Filter" box of the "OpenWire Pin Binding Editor".
In the left view select the "procedure Stop()" of the **VLAVIPlayer1.**
In the right view of the "OpenWire Pin Binding Editor", double-click on the "Clock MultiSinkPin" add the pin:



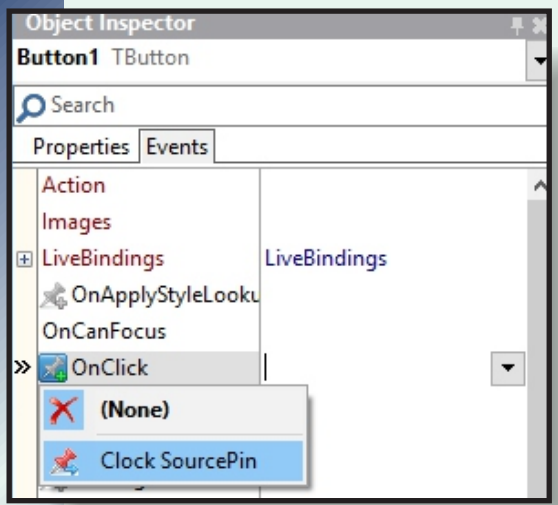
Close the "OpenWire Pin Binding Editor" dialog.
Finally we will add Pins to the **Check Box** and the Buttons so we can control the **VLAVIPlayer1.**
Select the **CheckBox1**. In the **Object Inspector** click on the button at front of the "IsChecked" property. From the drop down menu, select "Boolean SourcePin":

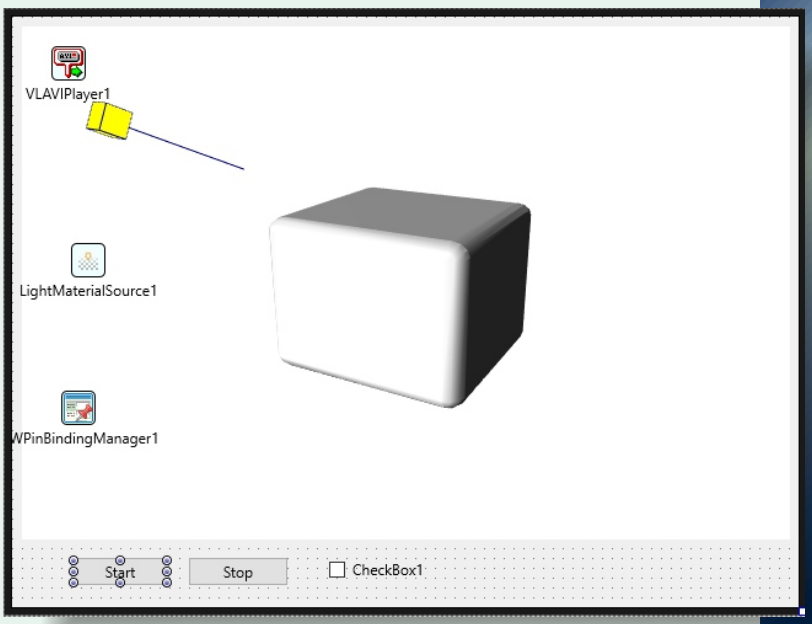




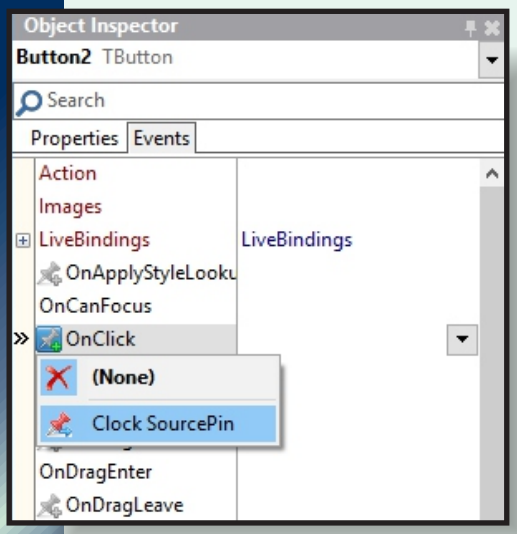
Select the **Button1.** In the **Object Inspector** switch to the "Events" tab.
In the **Object Inspector** click on the button at front of the "OnClick" property.

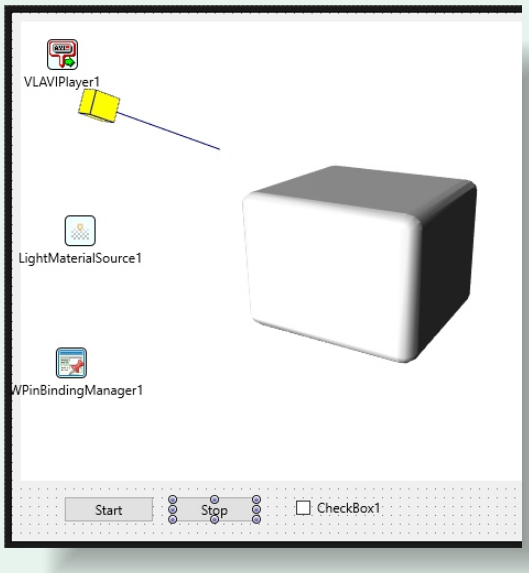
From the drop down menu, select "Clock SourcePin":



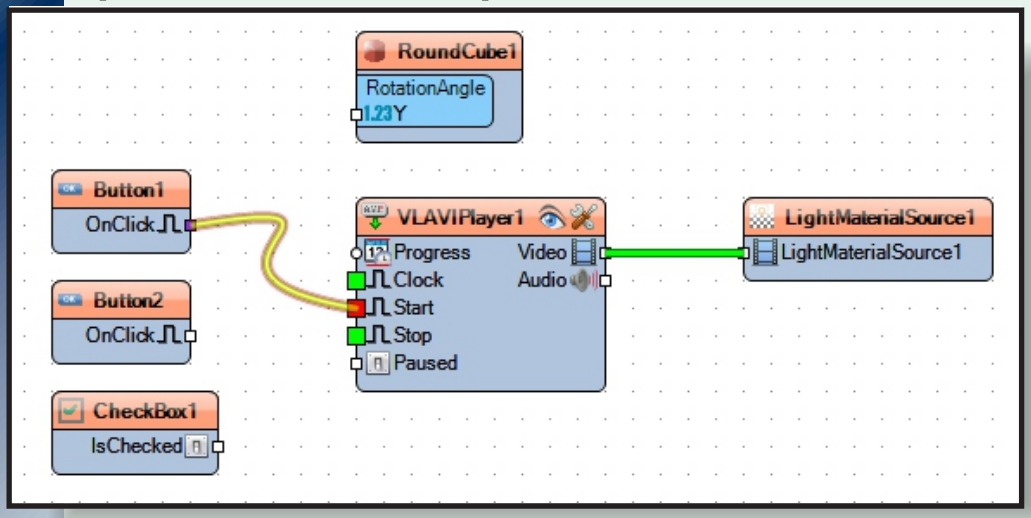


Select the **Button2.** In the **Object Inspector** click on the  button at front of the "OnClick" property. From the drop down menu, select "Clock SourcePin":

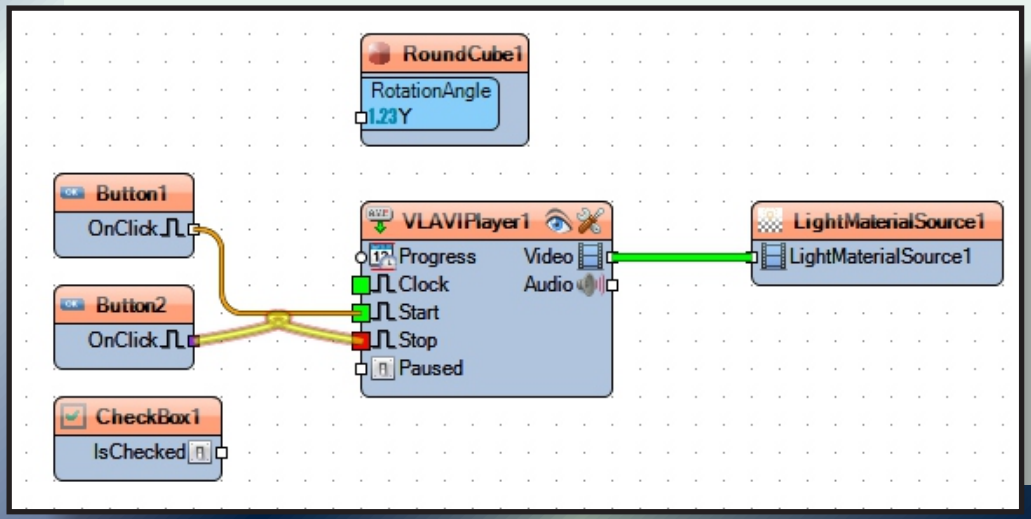




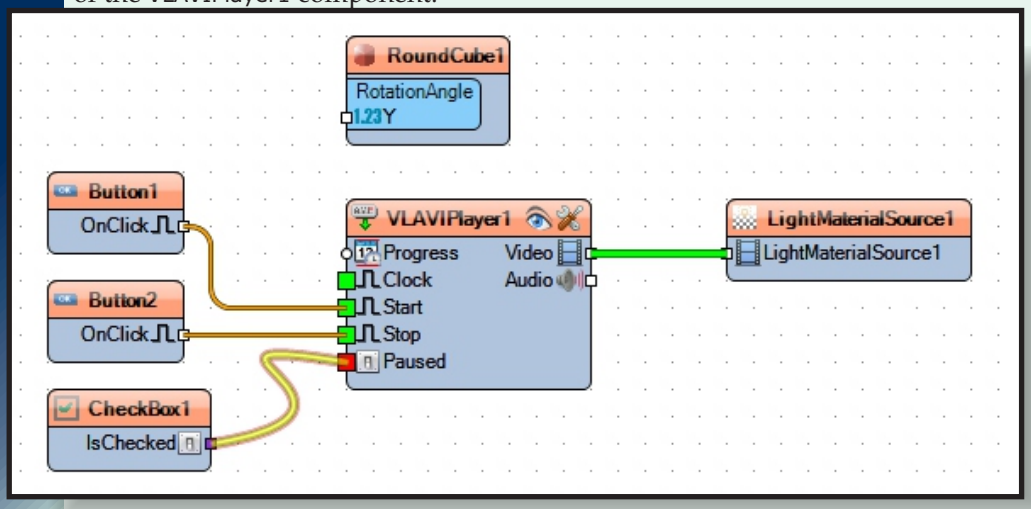
Switch to the "Open Wire" tab, and connect the "OnClick" Output Pin of the **Button1** to the "Start" Input Pin of the **VLAVIPlayer1** component:



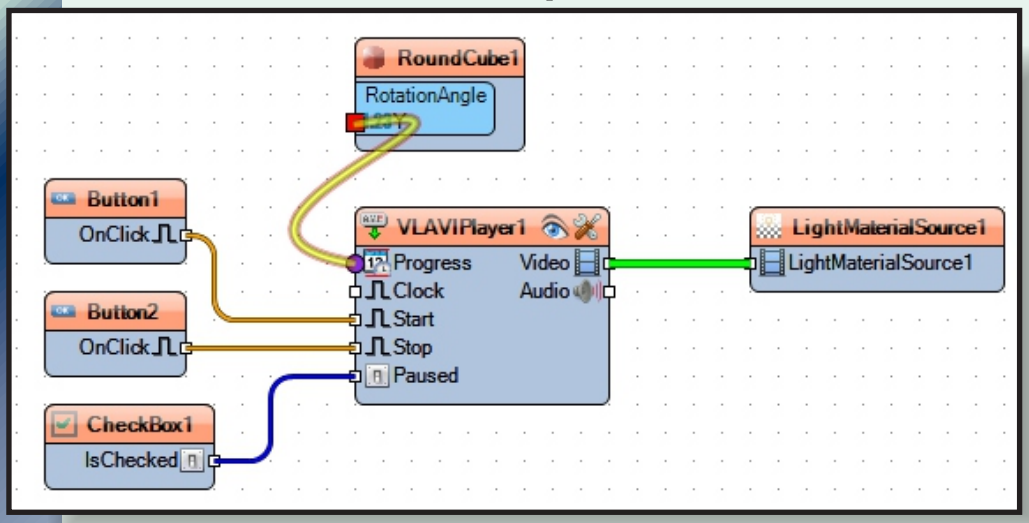
Connect the "OnClick" **Output Pin** of the **Button2** to the "Stop" Input Pin of the **VLAVIPlayer1** component:



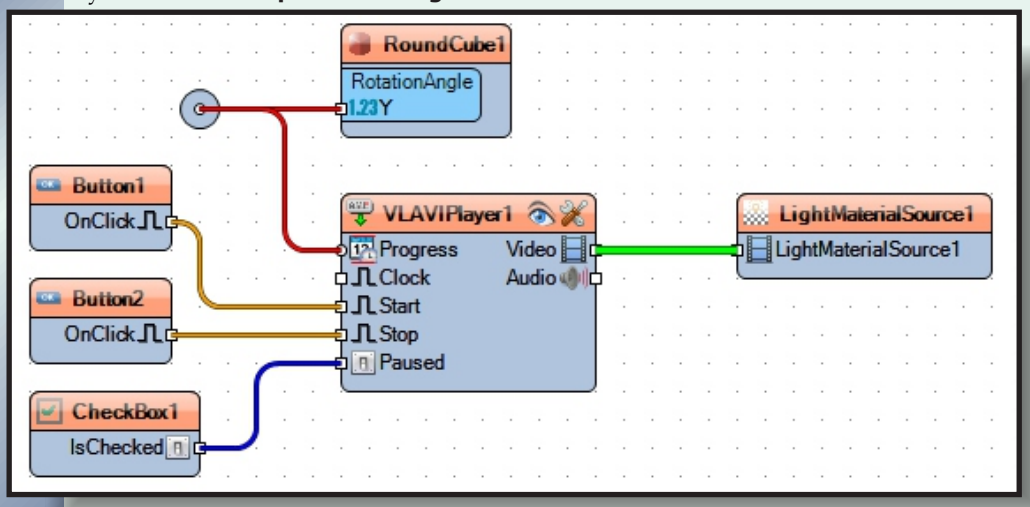
Connect the "IsChecked" Output Pin of the **CheckBox1** to the "Paused" Input Pin of the VLAVIPlayer1 component:



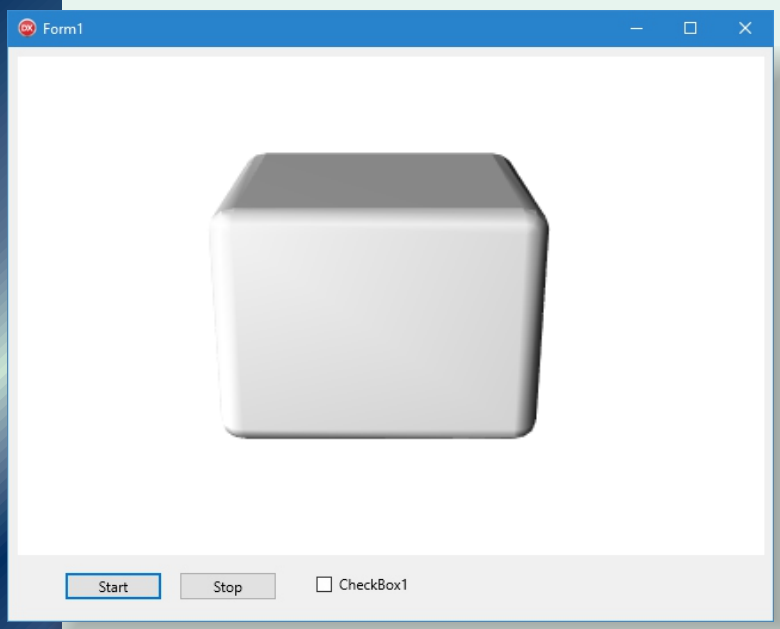
Connect the "Progress" State Pin of the **VLAVIPlayer1** to the "Y" Input Pin of the "RotationAngle" of the **RoundCube1** component:



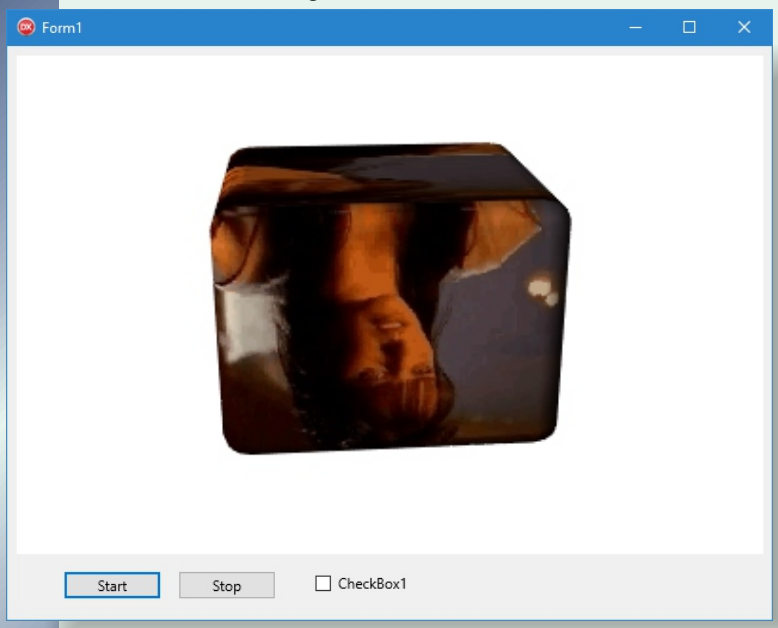
The two pins will be connected together with the help of a state dispatcher represented by a circle in the **OpenWire diagram:**

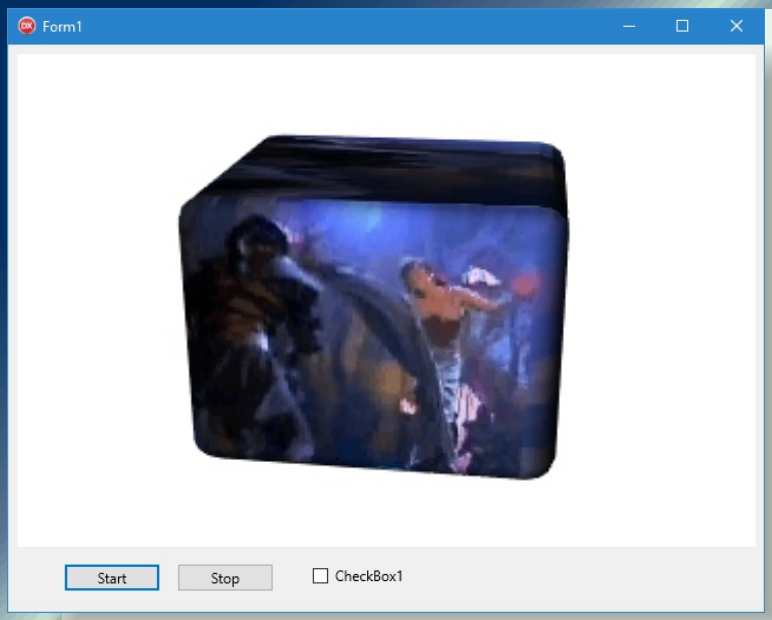


The **State Dispatcher** allows multiple **State Pins** to be connected and to share the same state.
It and also allows **Sink Pins** to be connected and to receive the same state. Since the "Progress" pin is a State Pin and can both receive and send the **Progress position**, it will always be connected through a dispatcher.
Compile and run the application.
Initially the cube will not be rotated, and will have empty surface:

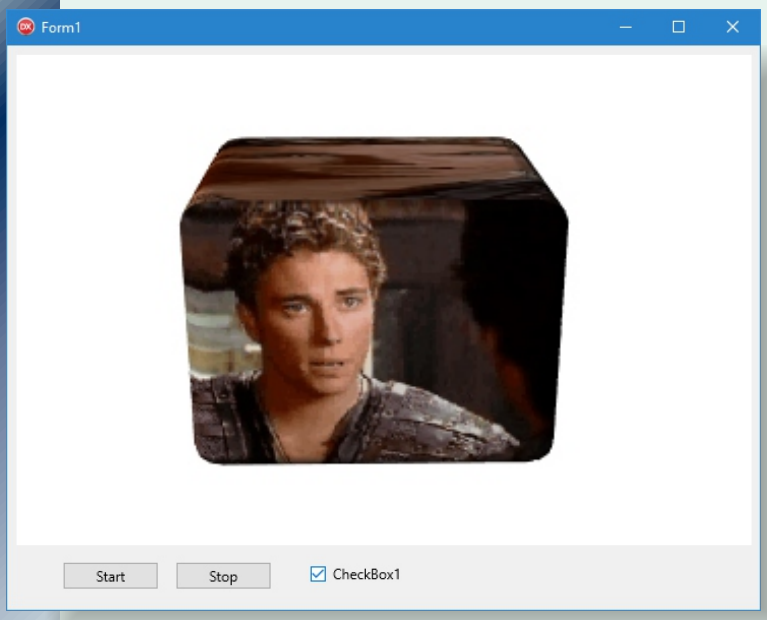


Click on the "Start" Button.
You should see the video playing on the surface of the Cube, and the Cube rotating:





If you Check the **CheckBox1,** the video will pause:



REMEMBER:
YOU CAN DOWNLOAD THE TRIAL VERSION.

IT'S FULLY FUNCTIONAL .
IT'S FREE.

AND HAS NO LIMITATIONS.

And if you uncheck the **CheckBox1,** the video will resume.
If you click the "Stop" button the video will stop, and you can start it again from the beginning after it has stopped, by clicking on the "Start" button.

CONCLUSION:
In this Article I showed you how you can render your videos inside Scope, or Waterfall, Visual Instruments from InstrumentLab, or over LED Matrix component. I also showed you how you can render the video over some bitmap type surfaces such as the buttons of the Scope, or Waterfall, and finally how you can render the video on the surface of 3D object in Fire Monkey Application.
In the following Articles I will show you how you can play video using different video sources such as Video Player, USB or IP Cameras, external devices, or internet streams. I will also show you how you can mix the different streams in variety of ways, and how you can record the video, broadcast it over Internet, or send it to external recording devices.

MITOV SOFTWARE

Video

Follow Yellow Brick

Audio

TO THE WORLD OF TECHNOLOGY


WWW.VISUINO.COM
Signal Processing
Arduino
Communication
the Road
Animation
Computer Vision
Process Control
AI
AND MAGIC
WWW.MITOV.COM

starter expert DELPHI TOKIO 2.2

INTRODUCTION

Since Delphi XE2 Embarcadero provides developers with high-quality tools for multi-platforms rapid application development.
The general paradigm remains the same, while now it needs the use of **FMX platform**. So Delphi developers can implement just the same visual prototyping approach to building applications, while not all the legacy code is the same.

The core is the compiler, so the code for business logic is the same, but forms with controls in VCL and FMX are not compatible. **MIDA converter** is a very good 3rd party form converter and makes the life much easier, while calls for one-way conversion from **VCL project to FMX one.**

The developers are facing the hard choice: stay with VCL and be windows-only developers or migrate to **FMX** with the great opportunity to have multi-platform project later on.

By now Embarcadero has been positioning VCL as best-in-class technology for Windows development. The VCL is updated release by release to support modern OS features, such as new VCL controls for the **Windows 10 UI**.

With Delphi the existing VCL applications are easily migratable to Windows 10 to leverage the latest UI controls and platform APIs. At the same time traditional VCL applications are good enough for many business users, and the support of additional platform either Linux, or Mac OS, or better both of them on the same codebase is the high dream.

Embarcadero has no signs of future VCL support of other-than-Windows platforms in its roadmaps, and we Delphi developers have to look around, hoping Embarcadero technology partners can propose some solutions.

CROSSVCL FOR VCL TO CROSS PLATFORM BOUNDARIES

The truly ideal solution is the technology, which can propagate **VCL** project from **Windows** to **Linux** and **Mac OS** without conversion.
Only in this case an arbitrary legacy codebase can remain the same.
Before May 2017 it seemed impossible, as **VCL** was too bound to **WinAPI** genetically and deep inside. And that was the official reason why Embarcadero introduced a new platform **FMX.**
The very **FMX** at the beginning was unbound from **WinAPI,** while supporting it. **FMX** was designed by **Eugene Kryukov** at the level, sufficiently abstract from **WinAPI,** thus letting different implementation for different **OS's** exist. **FMX** was not accepted by considerable part of **Delphi** developers, as they were not ready to pay the price of project conversion to win another platform.

Thus the **Delphi** world was split into two parts, and in some cases the boundary was across the same person: one month I support legacy project with **VCL,** the other month I develop a new multi-platform project with **FMX.** The problem though was not so severe, as business logic in Object Pascal, non-visual component use code and data access was the same, especially after Embarcadero had included **FireDAC** as multiplatform technology.
However, when a developer is fully dedicated to one project, either as in-house programmer, or independent solution provider, and the project is big, there is no practical way to satisfy his or her users with the support of another platform. **Wine** as a compatibility layer for **Windows** applications to run on Unix-like operating systems is well known, but not technologically perfect to be considered as a universal solution.
Previously, **Delphi** had only **VCL** and **Windows** compiler. Since generation XE2 Delphi has a **Mac OS** compiler and **Linux** compiler since 10.2, so the only obstacle is the **VCL** dependence on **WinAPI.**
The apparently unsolvable problem was finally solved by the release of **CrossVCL.**
CrossVCL is a set of tools for Delphi developers, who can now user **Delphi** with **VCL** to make applications for **Mac OS** and **Linux,** of course, having **Windows** as a primary platform for Delphi IDE itself running on this OS.
The technology doesn't need more than one click for conversion the project into native applications for **Mac OS** or **Linux.**

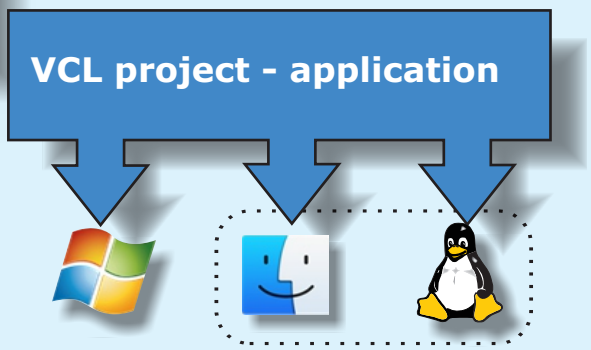


Figure 1 CrossVCL helps Delphi to cover 3 Oss

From the viewpoint of **Delphi** user **CrossVCL** is an IDE plugin, which needs no special tricks. You can easily go to product page `www.crossvcl.com` and download the installation pack. After running the installation pack and completing the process, you can continue using **Delphi** IDE normal way, but now with the capability to build your project for **Linux** and **Mac OS. The latest Delphi version only.**

YOU NEED DELPHI 10.2.2, as at the current stage of the product **CrossVCL** doesn't have backward compatibility and **supports the latest Delphi version only.** The good idea is to give a try to **CrossVCL** by starting a new traditional **VCL** application from scratch normal way. Create a new **VCL** project and save it to some folder. Then sit a button, an edit and a label on a form and write the response

```
Label1.Caption := 'Hello, ' + Edit1.Text;
```

That is really enough, so you can save the project and press **F9** button as we do since Delphi 1. The application will be built and then run on Windows. Then you can simply go to Project Manager pane, right click on "platform" node and add more platforms (*Figure 2*).

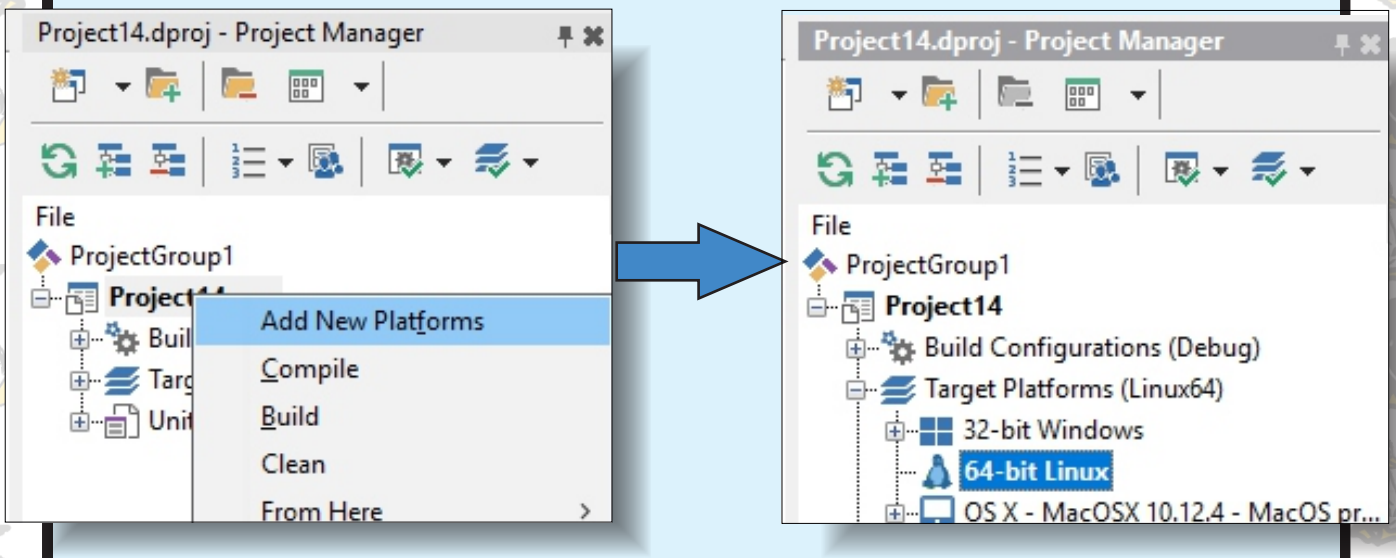


Figure 2 Adding platforms to VCL project with CrossVCL

After the platforms are successfully added, you can select the particular platform and then re-build the application, which this time will be native for the selected platform. For those developers, who still do only **VCL** projects and never tried FMX, please, look at the **Embarcadero's documentations on PAserver: `http://docwiki.embarcadero.com/RADStudio/Tokyo/en/PAServer,_the_Platform_Assistant_Server_Application`.**

I always prefer looking at some schematics before reading manuals as shown in Figure 3.
In case of **VCL** use for **Windows** development you don't think of deployment, as both Delphi IDE, and the resulting application are run on the same OS and typically on the same machine or **VM.**
In case of development on Delphi on Windows machine, but for different OS, the IDE needs a special agent, namely **PAserver** on the machine or **VM** with the different OS.
In this case a **PAserver** instance is up and running either **Mac OS** or **Linux,** and the resulting app is first sent from Delphi IDE to **PAserver,** which then run deploy and run the application, optionally under debugging.

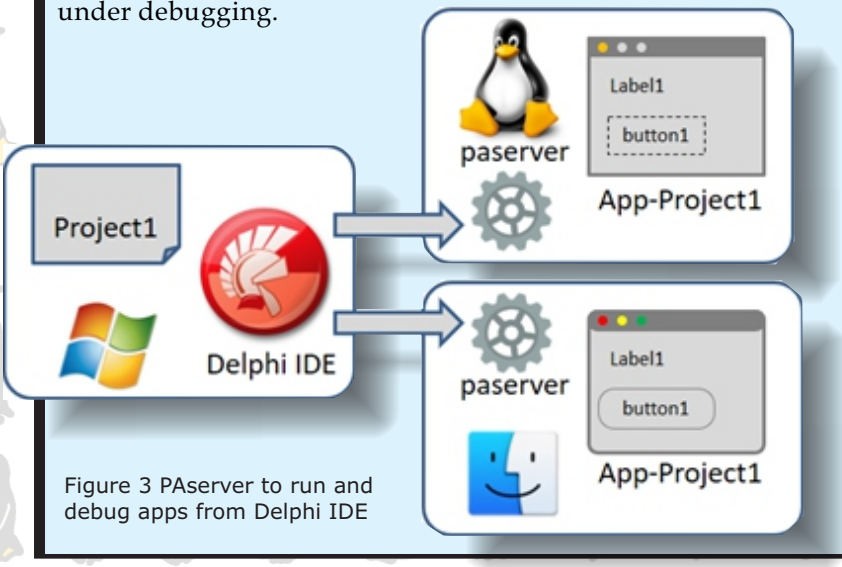


Figure 3 PAserver to run and debug apps from Delphi IDE

The specifics of **PAserver** installation and communication with Delphi IDE is well described in the documentation, while the process itself is very simple – only IP-address of the target OS machine is what you need to know.

When you select a specific target OS in Project Manager, you send a command to Delphi IDE to switch the compiler for the OS and configure the Deployment Manager to pack and send right set of files to the **Paserver.** For more details please refer to the documentation online `http://docwiki.embarcadero.com/RADStudio/XE5/en/Deploying_Applications_Overview`.

Making the multi-platform applications is so easy with **CrossVCL,** that the deployment manager is the only new topic to learn. Once you successfully install the necessary instances of **PAserver** on your **Mac OS** and **Linux** machines and configured connection profiles to let you Delphi IDE be connected to the proper **PAserver** instances, you can not only select, but re-build the application for the selected OS.

The process of building the resulting application takes more time comparing to applications for **Windows,** but not dramatically. If you want to run the application on the selected platform, you can do this. This time the process will take more time, as before running the IDE sends the binary and other complementary files to the **PAserver,** which in turn deploy them and only then run.

If you're starting a new project as we've done recently, you definitely need to be sure the **CrossVCL** mechanisms, integrated in IDE, work file. Later on running the resulting application on the other OS doesn't contribute to your productivity. It should be done not too frequently.

Some acceleration of the deployment can be achieved by using **VM's** on the same workstation, as well as using physical cable instead of **WiFi,** especially in the office.

Currently, I'm using **Mac** machine with two **VM's: Windows** and **Linux,** and can show the result of our exercise (*Figure 4*).

The forms of running applications are the same, but reflects the specifics of a particular OS.

CrossVCL is one of the most expected technology in Delphi world, as needs no special knowledge or tricks to adapt your Windows applications for **Mac OS** or **Linux.** The main advantage is that you can do it right now, and only then think of differences between platforms.

The differences between platforms will appear, but not in the UI, as **CrossVCL** carefully function by function transforms the visual **API.** And this is the way of **CrossVCL:** focus on visual controls, and leave all the other job to multi-platform **RTL** and compiler. Generally speaking, **CrossVCL** is a bridge between the Windows visual **API** and visual **APIs** of **Mac OS** and **Linux.** Conditional compilation will inevitably appear, but not to great extent.

CrossVCL has always limitation, as not any legacy Delphi projects can be re-built for Mac OS and Linux in one click.

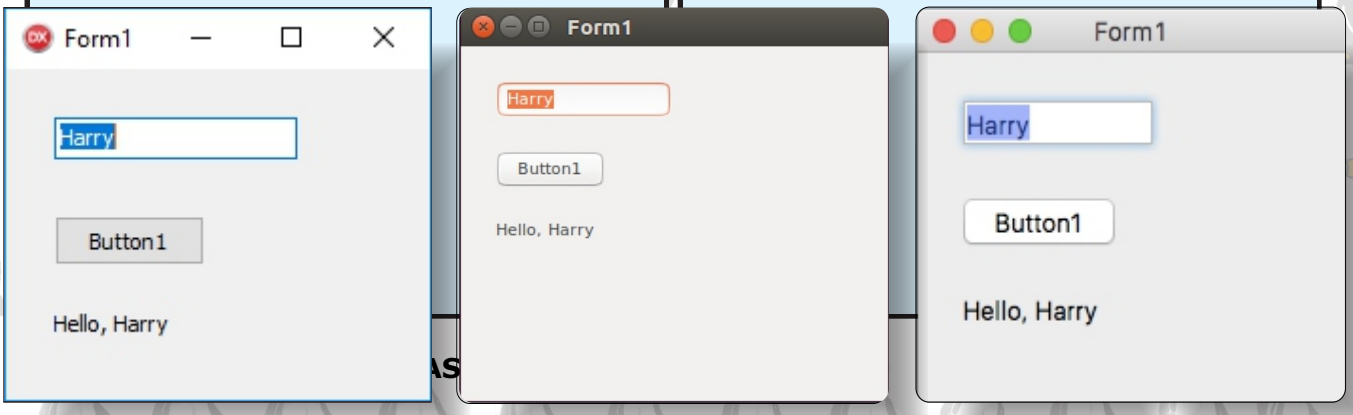
THE MAIN PROBLEM IS THE OBLIGATORY MIGRATION OF PROJECTS TO DELPHI 10.2.2. IT'S BAD NEWS FOR DELPHI 7 FANS, WHILE THEY HAVE ALWAYS BEEN WARNED OF THE NECESSITY TO UPGRADE TO NEWER VERSIONS.

The other thing is the use of **FireDAC,** which is definitely kind of standard of data access now. Anyhow, **Delphi** users first time in their history can think of **Mac OS** and **Linux** as the platforms they can develop for using **VCL.**

The last point to discuss is the support of 3rd party components by **CrossVCL.** Long lasting and developing **Delphi** project in most cases uses some 3rd Party components, extending the basic capability, provided by Borland in the past and Embarcadero now.

CrossVCL moves forward, covering the most used "visual" **API**. That means the chances for easy conversion of some visual components are very high, of course, if the source code is available. Abandoned libraries sadly can't be used, while the active technology companies can easily communicate with **Eugene Kryukov** for the support in checking the compatibility.

Figure 4. The same application for different platforms: Windows, Linux, Mac OS



If the library is compatible, as plenty of **WinAPI's** are bridged to native **Mac OS** or **Linux API's**, it can be approved within the short timeframe. **If not, CrossVCL team will add the necessary WinAPI in the support list with high priority.**
GDI* is implemented quite completely, and **GDI+** to about 90%, so most existing WinAPI-based libraries, as well as **OpenGL** based libraries are covered by the current **CrossVCL** version. (*The Graphics Device Interface (GDI) is a Microsoft Windows application programming interface and core operating system component responsible for representing graphical objects and transmitting them to output devices such as monitors and printers.*)
The logical question at the end can be about **CrossVCL** support of **iOS** and **Android.**
The forecast is now not optimistic, if we put aside "wow" effect.

CrossVCL is aimed to extend the existing **VCL** projects to other platforms. The projects are assumed to be started several years ago and as desktop applications, and the form design corresponds the monitors, as well as the overall user experience reflects the desktop use. So kind of automatic conversion, if imagined, the project from **Windows** to **iOS** or **Android** won't give good results.

And if we talk about a new mobile project, **FMX** is better choice, than **VCL,** and it has mobile UI components like `TListView` and many others which **VCL** hasn't. Thus starting of a new mobile project with **VCL,** even if we could, is not a good idea. I won't expect great demand to this capability, while bridging the **VCL** projects to **Mac OS** and **Linux** is a unique and very attractive feature of **CrossVCL.**

FMXLINUX TO COMPLETE MULTI-PLATFORM SUPPORT

I hope **VCL** users now have lots of optimism, they do know how to continue their favorite projects and support **other-than-Windows** desktop platforms with no efforts.

At the same time, many successful **FMX** projects started for the reasons of multi-platform support. We can define many ways from **VCL** to **FMX,** but not in the opposite direction. **VCL** as an older visual technology has very few advantages over **FMX,** so I never heard of backward transition. "Times of VCL have gone" is not what I'm telling the audience. Better to say is "new projects need new technologies".

A dilemma of VCL versus FMX exists only for the existing projects: **to convert or not to convert.**

Simplicity of **CrossVCL** is a good point, not readiness of 3rd parties is a weak point. One can start a quick **VCL** project, saving time and thinking of "utility"-like application or in-house project. The long-lasting project for a modern enterprise or individuals does need multi-platform support.

Embarcadero provided support of **Mac OS** in XE2 release. **Delphi** users were excited by the perspective of having new platform support every new release or, at least, every year.

The pace was as expected at the beginning, but **Linux** users were held down for the sake of **iOS** and **Android.** The number of mobile users and mobile developers grew faster, than **Linux.** But finally in May 2017 **Embarcadero** provided the support of **Linux,** but in quite funny way.

The **compiler and RTL for Linux** were included into the release, but **FMX** support didn't. **Embarcadero** officials positioned this capability as the way of making server applications for **Linux.**
I leave the discussion of how server-side development for **Linux** is supported in **Delphi 10.2** for those who need it. Reasons of why **Embarcadero** didn't provide **FMX** support for **Linux** could be connected to the cautious marketing position, not to invest much to the direction, which is not so attractive as mobility. Native controls for **Android** and **iOS** looks better to some segments.

The solution of support **Linux** visually was possibly taken in the times of **Embarcadero** acquisition by **IDERA** and **R&D** recombination. Finally, we now have **Eugene Kryukov,** the principal architect and developer of **FMX** concept, outside **Embarcadero,** but not outside the **Delphi** world.
Quite logically Eugene, as independent developer already, continued developing **FMX.** Aside from **CrossVCL, FmxLinux** is also his new project.
FMXLINUX IS AS A SET OF TOOLS FOR DEVELOPMENT OF APPLICATIONS FOR LINUX USING EMBARCADERO DELPHI AND LINUX COMPILER. IT NEEDS DELPHI 10.2.2 AND LATER.
From the viewpoint of design time, **FmxLinux** is an IDE expert for adding **Linux** platform to **FMX** project. From the viewpoint of runtime, **FmxLinux** is a brand-new implementation of **FMX** for **Linux,** tightly integrated with the native **Linux.** One can download the trial version for the project side **`www.fmxlinux.com`** and then build the conventional **FMX** visual project for **Linux.** After installation of **FmxLinux** you can add **Linux** to the list of platform supported in **Project Manager** both for new, and existing project (*Figure 5*).

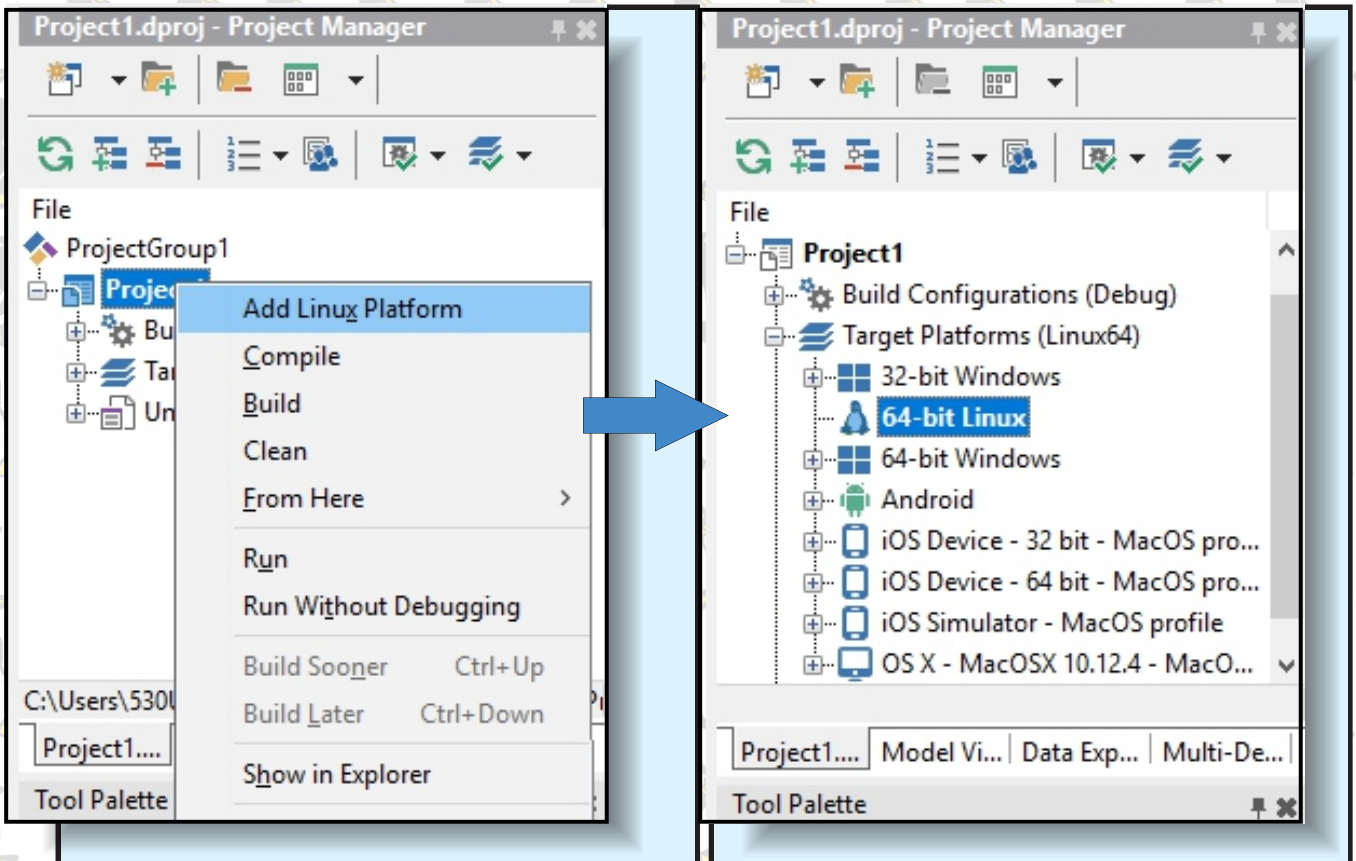


Fig. 5. Adding Linux support to visual FMX project

If you have an available **Linux** machine or **VM** with **PaServer** installed and the corresponding profile ready, you can just select a new target platform in the **Project Manager**, build and run your application on **Linux.**

A good way to see **FmxLinux** in action is the building and running demos, which can be taken both from the demo pack provided with the trial version, and from the standard **Delphi** samples.

At the moment **FmxLinux** was proved by users to support the following list of **Linux** versions:

Ubuntu 16.10 Linux
CentOS 7
Debian 9.1
Fedora 26
Gentoo Linux 2016.07.04
Kali Linux 2017.2
Linux Mint 18.1
Manjaro Linux 17.0
OpenMandriva Lx 3.0
openSUSE Leap 42.2
Parrot Studio 3.8
PCLinuxOS 2017.07
Red Hat Enterprise Linux 7
Zorin OS 12.1 Core

Regarding the 3rd parties, the situation is much better comparing to **CrossVCL,** as libraries for **FMX** were created by the most active companies, and they are the same active and responsive now.

For example, **TMS FMX UI Pack** does support **FmxLinux,** that means if you have these visual components in your project already and then install **FmxLinux,** you won't have problem of rebuilding all the project for **Linux.**

FmxLinux as the **FMX** implementation follows the main concepts of styling. Any visual **FmxLinux** application gets the **Linux** style by default, while you **can change the style dynamically to any of appropriate** (*Figure 6*).

Many interesting styles for **FMX** are available, here are two examples:

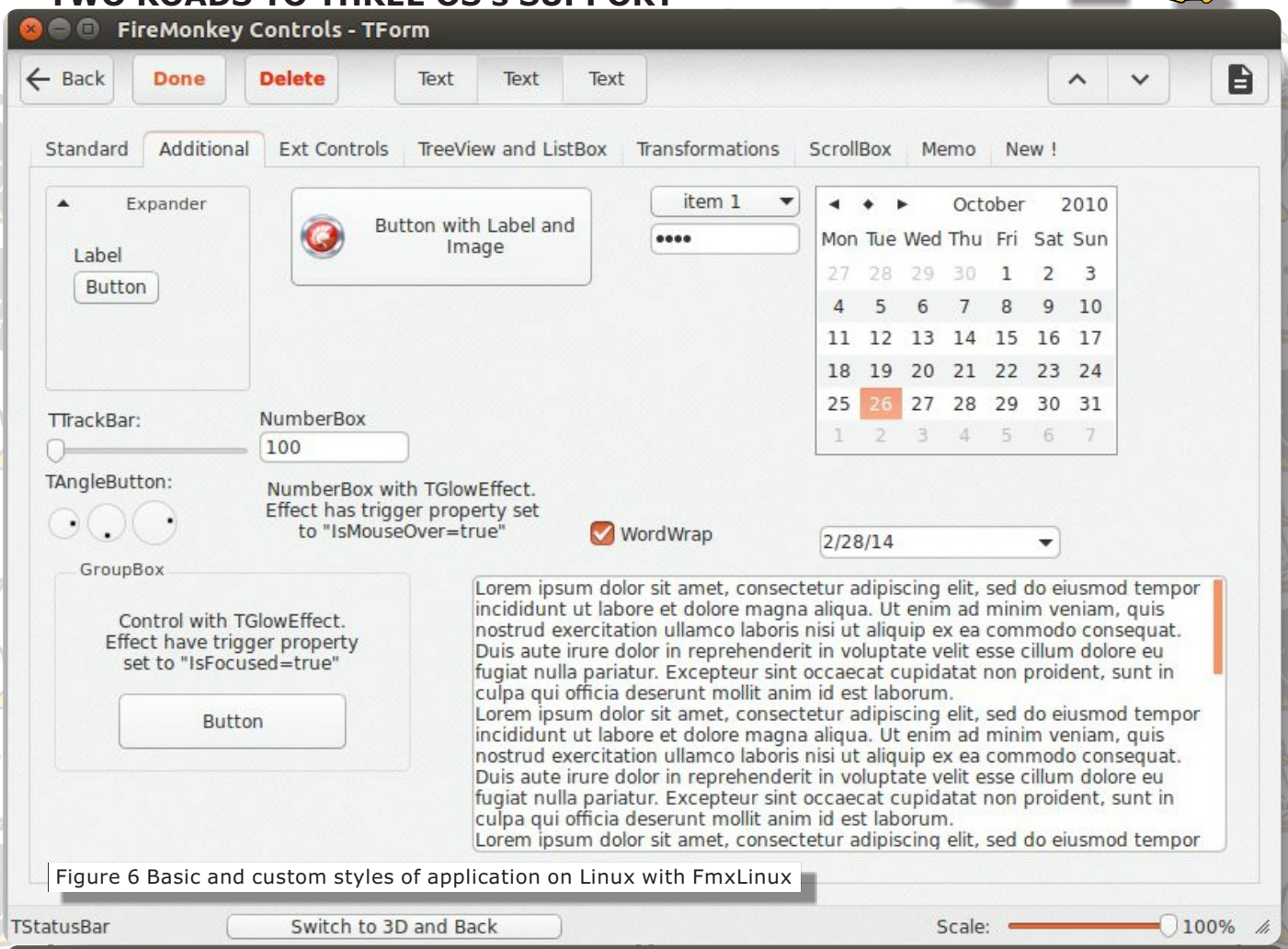
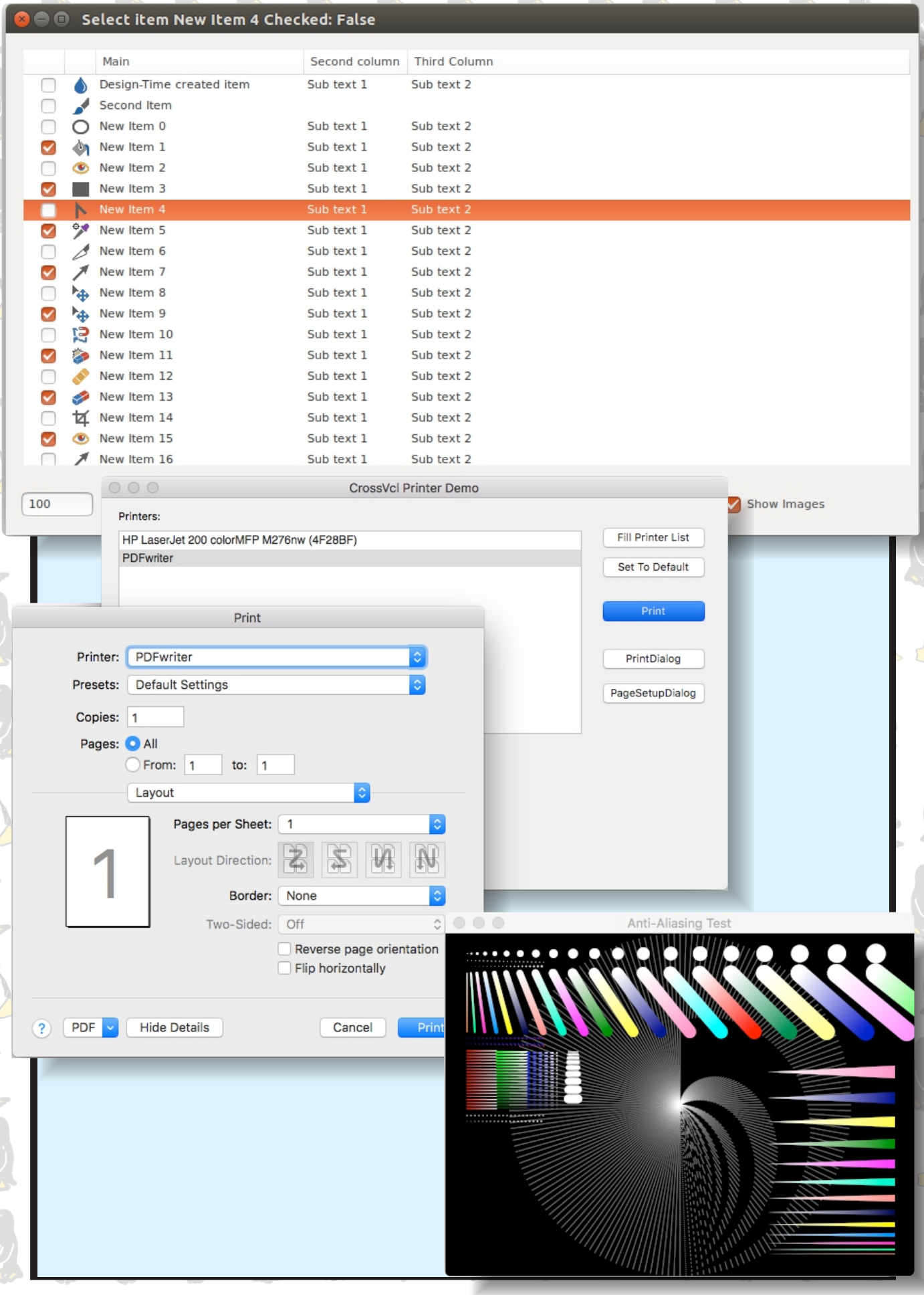
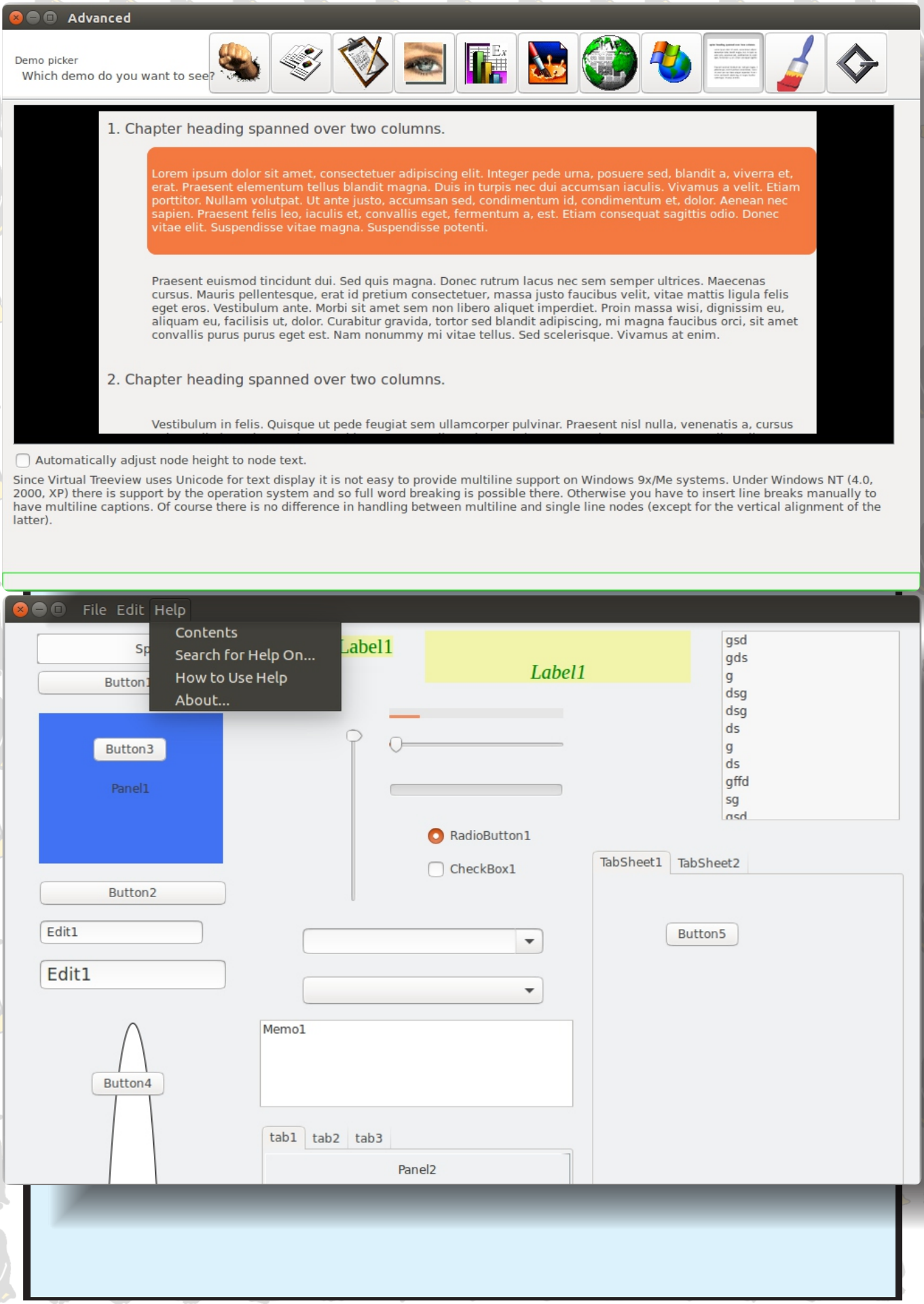
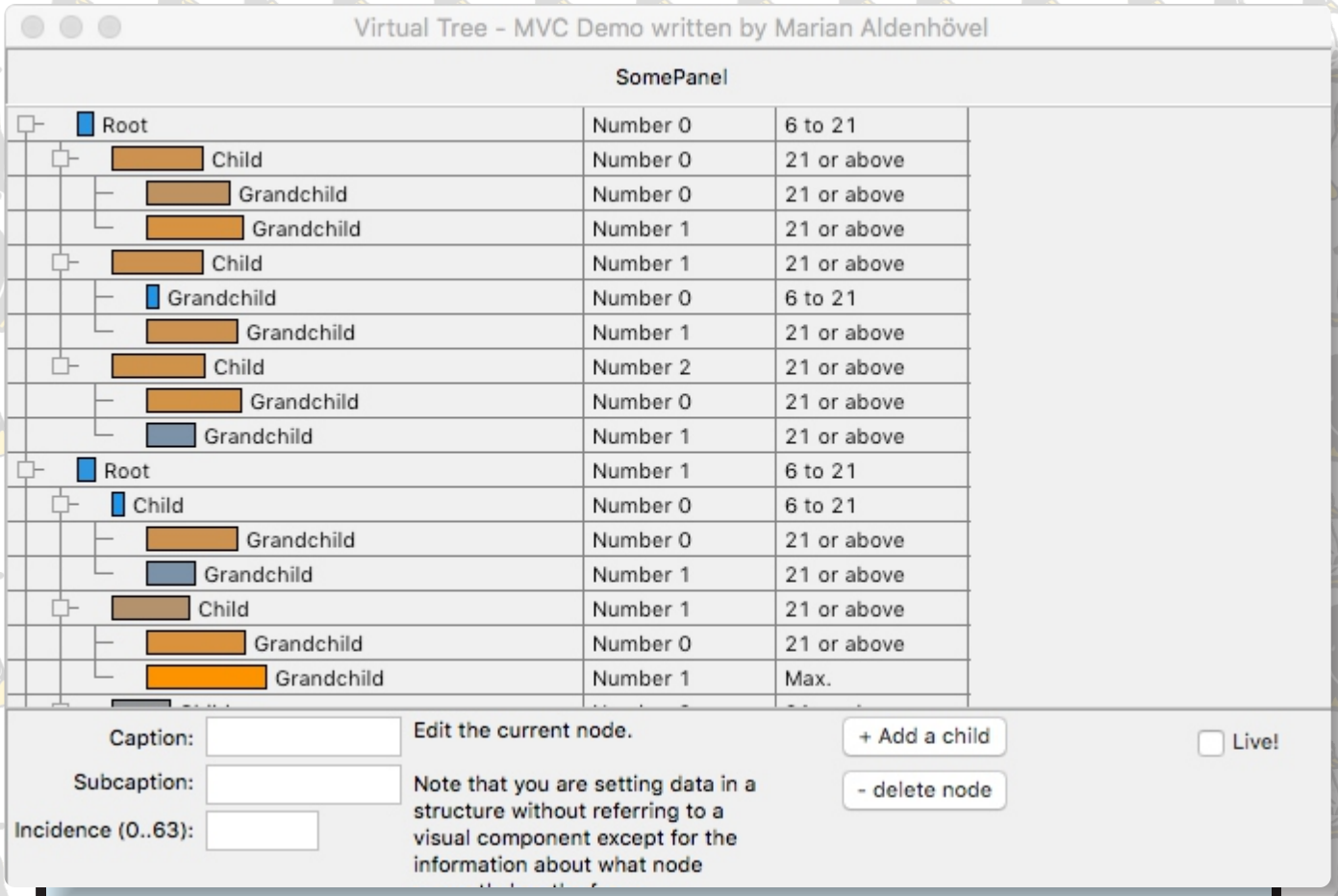
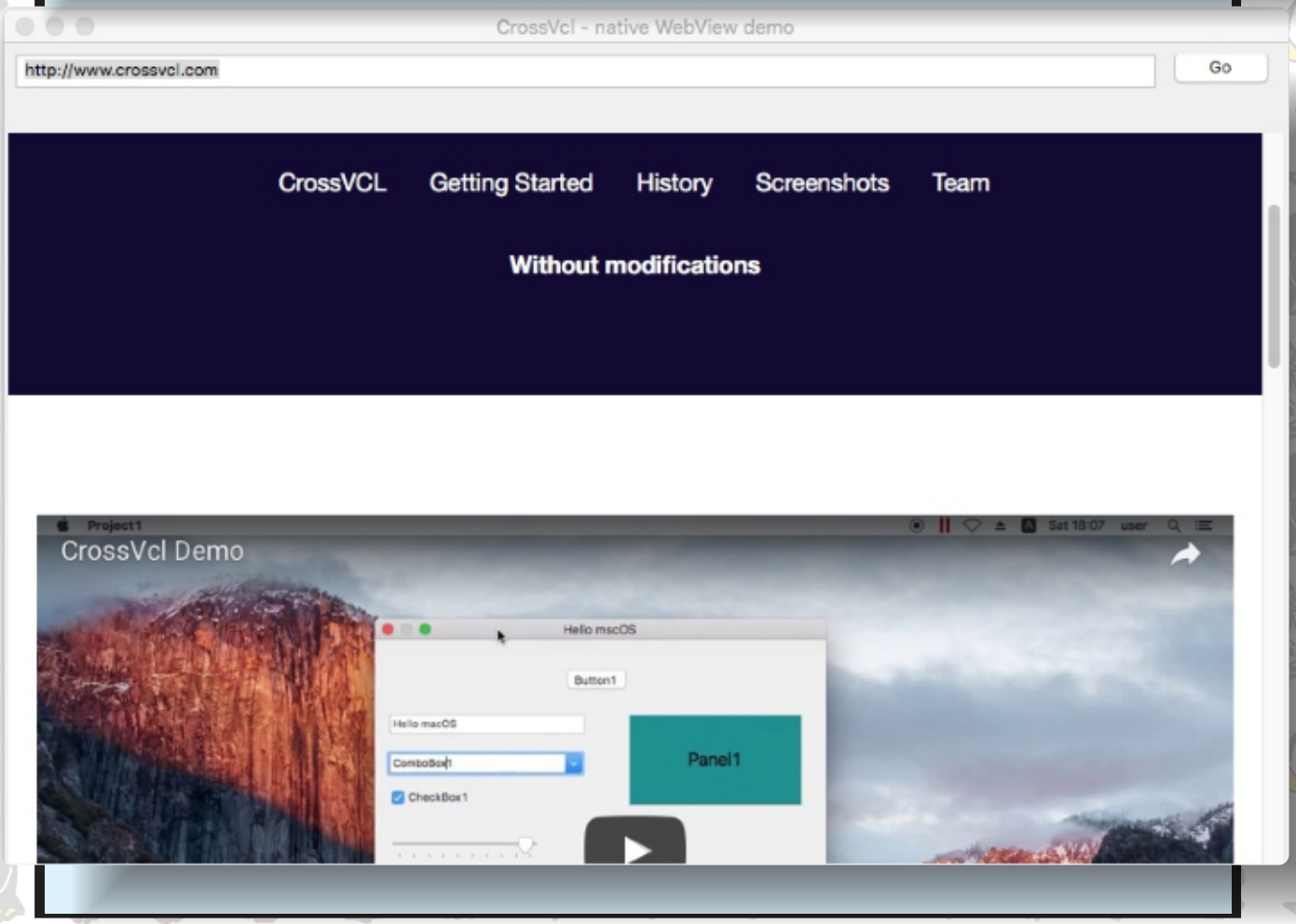


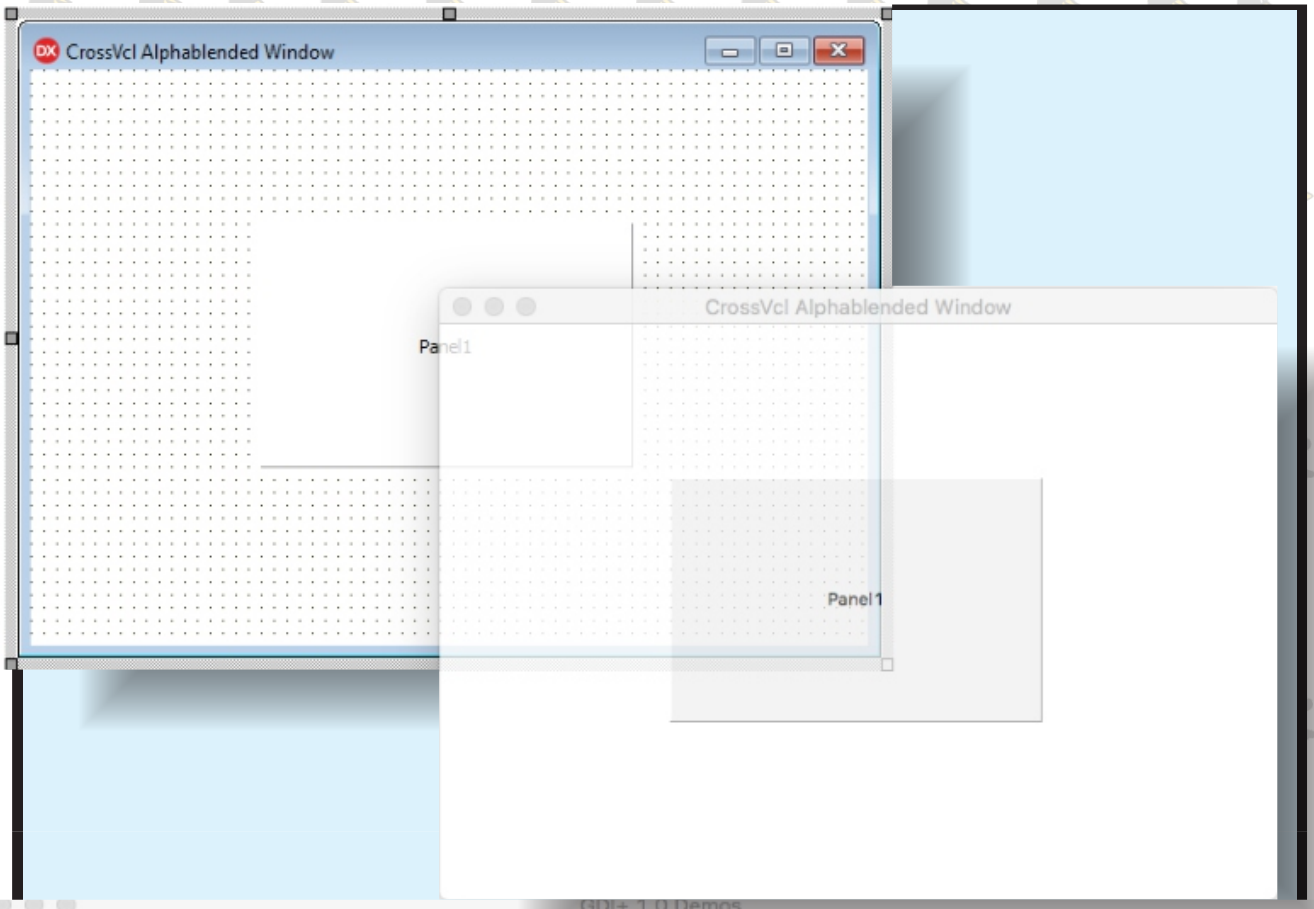
Figure 6 Basic and custom styles of application on Linux with FmxLinux

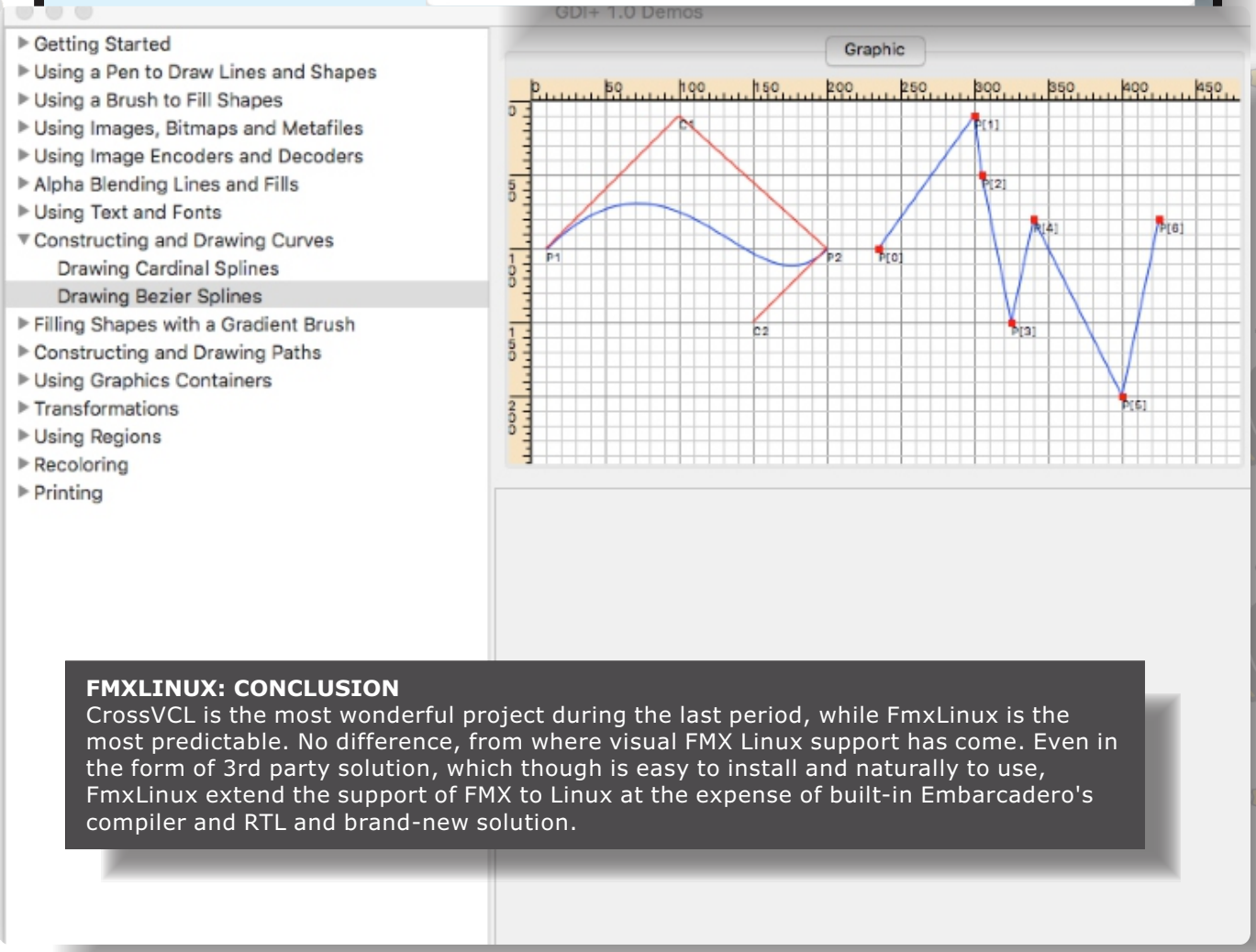

Select item New Item 4 Checked: False
Main
Second column
Third Column
Design-Time created item
Second Item
New Item 0
New Item 1
New Item 2
New Item 3
New Item 4
New Item 5
New Item 6
New Item 7
New Item 8
New Item 9
New Item 10
New Item 11
New Item 12
New Item 13
New Item 14
New Item 15
New Item 16
Sub text 1
Sub text 2
100
Show Images
CrossVcl Printer Demo
Printers:
HP LaserJet 200 colorMFP M276nw (4F28BF)
PDFwriter
Fill Printer List
Set To Default
Print
PrintDialog
PageSetupDialog
Print
Printer:
PDFwriter
Presets:
Default Settings
Copies:
Pages:
All
From:
to:
Layout
Pages per Sheet:
Layout Direction:
Border:
None
Two-Sided:
Off
Reverse page orientation
Flip horizontally
PDF
Hide Details
Cancel
Print
Anti-Aliasing Test


Advanced
Demo picker
Which demo do you want to see?
1. Chapter heading spanned over two columns.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer pede urna, posuere sed, blandit a, viverra et, erat. Praesent elementum tellus blandit magna. Duis in turpis nec dui accumsan iaculis. Vivamus a velit. Etiam porttitor. Nullam volutpat. Ut ante justo, accumsan sed, condimentum id, condimentum et, dolor. Aenean nec sapien. Praesent felis leo, iaculis et, convallis eget, fermentum a, est. Etiam consequat sagittis odio. Donec vitae elit. Suspendisse vitae magna. Suspendisse potenti.
Praesent euismod tincidunt dui. Sed quis magna. Donec rutrum lacus nec sem semper ultrices. Maecenas cursus. Mauris pellentesque, erat id pretium consectetuer, massa justo faucibus velit, vitae mattis ligula felis eget eros. Vestibulum ante. Morbi sit amet sem non libero aliquet imperdiet. Proin massa wisi, dignissim eu, aliquam eu, facilisis ut, dolor. Curabitur gravida, tortor sed blandit adipiscing, mi magna faucibus orci, sit amet convallis purus purus eget est. Nam nonummy mi vitae tellus. Sed scelerisque. Vivamus at enim.
2. Chapter heading spanned over two columns.
Vestibulum in felis. Quisque ut pede feugiat sem ullamcorper pulvinar. Praesent nisl nulla, venenatis a, cursus
Automatically adjust node height to node text.
Since Virtual Treeview uses Unicode for text display it is not easy to provide multiline support on Windows 9x/Me systems. Under Windows NT (4.0, 2000, XP) there is support by the operation system and so full word breaking is possible there. Otherwise you have to insert line breaks manually to have multiline captions. Of course there is no difference in handling between multiline and single line nodes (except for the vertical alignment of the latter).
File Edit Help
Contents
Search for Help On...
How to Use Help
About...
Label1
Label1
Button3
Panel1
Button2
Edit1
Edit1
Button4
RadioButton1
CheckBox1
gsd
gds
g
dsg
dsg
ds
g
ds
gffd
sg
gsd
TabSheet1
TabSheet2
Button5
Memo1
tab1
tab2
tab3
Panel2


Virtual Tree - MVC Demo written by Marian Aldenhövel
SomePanel
Root
Child
Grandchild
Caption:
Subcaption:
Incidence (0..63):
Edit the current node.
Note that you are setting data in a structure without referring to a visual component except for the information about what node
+ Add a child
- delete node
Live!


CrossVcl - native WebView demo
http://www.crossvcl.com
Go
CrossVCL
Getting Started
History
Screenshots
Team
Without modifications
CrossVcl Demo
Hello mscOS
Button1
Panel1
CheckBox1





FMXLINUX: CONCLUSION
CrossVCL is the most wonderful project during the last period, while FmxLinux is the most predictable. No difference, from where visual FMX Linux support has come. Even in the form of 3rd party solution, which though is easy to install and naturally to use, FmxLinux extend the support of FMX to Linux at the expense of built-in Embarcadero's compiler and RTL and brand-new solution.

KBMMW BINARY PARSER PART 1

BY KIM MADSEN

starter — expert DX Delphi

A universal binary parser made available as part of kbmMW Enterprise Edition

What does it do? Well.. it parses well formed binary (or textual) streams to extract telegrams and their contents. It functionality wise can be compared to a regular expression, just for bit and byte level information, although with simple scripting and calculation capabilities.

A telegram is in this sense a fixed length bunch of bytes, which may contain bit fields or byte fields or ASCII type string data.

The definition file defines how the telegrams are looking, what subparts they consist of, and what to do when a matching part has been found.

The outcome is typically a match along with a number of keys/values, or a failure to match with anything. The actual naming and use of the keys and their values is up to the developer to decide.

A definition file is default written in YAML and consists of 3 main sections:

1

- **VALUES**
- **TELEGRAMS**
- **TAGS**

The **VALUES** section can contain a list of predefined values to be available before any attempt to match anything. This can for example be used for defining "constants" which your application understands or default values.

The **TELEGRAMS** section contains an array of the telegram masks to look for, and the **TAGS** section contains an optional number of named sub parts referenced from either the **TELEGRAMS** section or from within the **TAGS** section.

It may seem a bit vague right now, but it probably makes more sense when I show a sample in a moment.

The **TELEGRAMS** section and the **TAGS** section both contains masks and optional expressions to be executed when a mask match. They also both define if the mask is a bytes mask, a bits mask or a string mask. Masks which has been defined as bytes masks, always operates on the byte level. Similarly masks which has been defined as bits masks, always operates on the bit level (*currently maximum 8 bit per mask*).

String masks are similar to bytes masks, except that they compare **ASCII** strings. Let us look at a sample definition file.

As **YAML actively uses indentation** to determine if something belongs to current definition or is a new definition, it is of high importance that the indentation is correct. **YAML also recognizes lines starting with – as an entry in an array**, unless the dash seems to be part of another property.

In fact **YAML is pretty complex** in what it understands, but it does read easier for the human eye, why I chose it as the default definition file format.

The sample definition file is for a standard scale format called **Toledo** deviced by a company called **Mettler Toledo** many years ago.

YAML wise, the document actually contains an object with one single property named **TOLEDO**, which has 3 properties, **VALUES**, **TELEGRAMS** and **TAGS**.
The **VALUES** property has a number of properties with values. The **TELEGRAMS** object has one property with an array of objects each having a mask and optional **expr property**
The **TAGS** property is an object which has a number of properties (**SWA**, **SWB**, **SWC**, **DP** etc) which each are objects containing a property named **bytes/bits/string** which is an object containing either a single mask and optional expr property, or an array of such.

It may take a while getting used to read and write **YAML** documents, but perseverance makes experts.

LINES STARTING WITH # ARE COMMENTS.

```
# This is a sample file showing how to parse Toledo telegrams
# using kbmMW Binary Parser

TOLEDO:
  VALUES:
    # Unit constants
    C_UNIT_GRAM:                  2000
    C_UNIT_UK_POUND:              2001
    C_UNIT_KILOGRAM:              2002
    C_UNIT_METRIC_TON:            2003
    C_UNIT_OUNCE:                 2004
    C_UNIT_TROY_OUNCE:            2005
    C_UNIT_PENNY_WEIGHT:          2006
    C_UNIT_UK_TON:                2007
    C_UNIT_CUSTOM:                2008

    # Status constants
    C_STATUS_OK:                  1000
    C_STATUS_DATA_ERROR:          1001
    C_STATUS_SCALE_ERROR:         1002
    C_STATUS_SCALE_OVERLOAD:      1003
    C_STATUS_IN_MOTION:           1004
    C_STATUS_TARE_ERROR:          1005
    C_STATUS_TRANSMISSION_ERROR:  1006
    C_STATUS_INVALID_COMMAND:     1007
    C_STATUS_INVALID_PARAMETER:   1008

    # Tare constants
    C_TARE_PRESET:                3000
    C_TARE_AUTO:                  3001
    C_TARE_NONE:                  3002

   # Default values
    STATUS:                       @C_STATUS_OK
    TARE:                         0
    GROSS:                        0
    NET:                          0
    INCREMENT_SIZE:               1
    IS_POWER_NOT_ZEROED:          false
    IS_SETTLED:                   false
    IS_OVERLOAD:                  false
    IS_NEGATIVE:                  false
    IS_CHECKSUM_OK:               false
    WEIGHT_FACTOR:                1
    TARE_FACTOR:                  1
    TARE_CODE:                    @C_TARE_NONE
    TERMINAL_NO:                  0
    WEIGHT_UNIT:                  @C_UNIT_KILOGRAM
    TARE_UNIT:                    @C_UNIT_KILOGRAM

TELEGRAMS:
    bytes:
       - mask: [ 0x2, @SWA, @SWB, @SWC, 6*@W, 6*@T, 0xD, @CHK ]
         expr: - "WEIGHT_UNIT=IF(IS_UNIT_UK_POUND=1,C_UNIT_POUND, IF(IS_UNIT_KILOGRAM,C_UNIT_KILOGRAM,WEIGHT_UNIT))"
               - "TARE_UNIT=WEIGHT_UNIT"
               - "STATUS=IF(IS_CHECKSUM_OK=1,IF(IS_OVERLOAD,C_STATUS_OVERLOAD,C_STATUS_OK),C_STATUS_DATA_ERROR)"
               - "WEIGHT=WEIGHT*WEIGHT_EXPANSION*F(WEIGHT_FACTOR<1,WEIGHT_FACTOR,1)"
               - "TARE=TARE*TARE_EXPANSION*IF(TARE_FACTOR<1,TARE_FACTOR,1)"
               - "GROSS=  IF(IS_NETTO=0,WEIGHT,0)"
               - "NET=    IF(IS_NETTO=1,WEIGHT,0)“

 TAGS:
    SWA:
    bits:
        # bit offset 0
        mask: [ 0, 0, 1, 2*@IS, 3*@DP ]
   DP:
    bits:
   # bit offset 0, 3 bits
       - mask: [ 0, 0, 0 ]
         expr: [ WEIGHT_FACTOR=100, TARE_FACTOR=100 ]
       - mask: [ 0, 0, 1 ]
         expr: [ WEIGHT_FACTOR=10, TARE_FACTOR=10 ]
       - mask: [ 0, 1, 0 ]
         expr: [ WEIGHT_FACTOR=1, TARE_FACTOR=1 ]
```

```
    -mask: [0,1,1]
     expr: [WEIGHT_FACTOR=0.1, TARE_FACTOR=0.1]
    -mask: [1,0,0]
     expr: [WEIGHT_FACTOR=0.01, TARE_FACTOR=0.01]
    -mask: [1,0,1]
     expr: [WEIGHT_FACTOR=0.001, TARE_FACTOR=0.001]
    -mask: [1,1,0]
     expr: [WEIGHT_FACTOR=0.0001, TARE_FACTOR=0.0001]
    -mask: [1,1,1]
     expr: [WEIGHT_FACTOR=0.00001, TARE_FACTOR=0.00001]

IS:
 bits:
    # bit offset 3, 2 bits
    -mask: [0,1]
     expr: INCREMENT_SIZE=1
    -mask: [1,0]
     expr: INCREMENT_SIZE=2
    -mask: [1,1]
     expr: INCREMENT_SIZE=5
CHK:
 bytes:
     expr:
    "IS_CHECKSUM_OK=IF(CHK2COMP7(0,17)=VALUE,1,0)"
SWB:
 bits:
     mask: [0, IS_POWER_NOT_ZEROED, 1,
     IS_UNIT_UK_POUND/
     IS_UNIT_KILOGRAM, !IS_SETTLED, IS_OVERLOAD,
     IS_NEGATIVE, IS_NETTO]
SWC:
 bits:
     mask:
   [0, IS_HANDTARE, 1, @EW, IS_PRINTREQUEST, 3*@WF]
WF:
 bits:
    -mask: [0,0,0]
    -mask: [0,0,1]
     expr: [WEIGHT_UNIT=C_UNIT_GRAM, TARE_UNIT=C_UNIT_GRAM]
    -mask: [0,1,0]
     expr: [WEIGHT_UNIT=C_UNIT_METRIC_TON, TARE_UNIT=C_UNIT_METRIC_TON]
    -mask: [0,1,1]
     expr: [WEIGHT_UNIT=C_UNIT_OUNCE, TARE_UNIT=C_UNIT_OUNCE]
    -mask: [1,0,0]
     expr: [WEIGHT_UNIT=C_UNIT_TROY_OUNCE, TARE_UNIT=C_UNIT_TROY_OUNCE]
    -mask: [1,0,1]
     expr: [WEIGHT_UNIT=C_UNIT_PENNY_WEIGHT, TARE_UNIT=C_UNIT_PENNY_WEIGHT]
    -mask: [1,1,0]
     expr: [WEIGHT_UNIT=C_UNIT_UK_TON,TARE_UNIT=C_UNIT_UK_TON]
    -mask: [1,1,1]
     expr: [WEIGHT_UNIT=C_UNIT_CUSTOM, TARE_UNIT=C_UNIT_CUSTOM]
EW:
 bits:
    -mask: 0
     expr: [WEIGHT_EXPANSION=1, TARE_EXPANSION=1]
    -mask: 1
     expr: [WEIGHT_EXPANSION=10, TARE_EXPANSION=10]

W:
 string:
     expr: WEIGHT=VALUE
T:
 string:
     expr: TARE=VALUE
```

When the **kbmMW Binary Parser** is provided this definition file, it compiles it to build a parse tree, which efficiently can parse whatever you throw at it as a file or a stream.

We can see that one telegram mask has been defined in the `TELEGRAMS/bytes array`. It contains a mask that consists of 8 parts.

Each part is, unless a '*' is included, exactly 1 byte wide. The first part is 0x2 which simply means that the data must start with the hexadecimal value 2, which is `STX` (*start of transmission*) in the **ASCII** character set.

The second part is `@SWA,` which means that there must be one byte, which will be parsed by the tag called `SWA.`

The `@SWB` and `@SWC` also match one byte, that each of them must be parsed by a named tag. Then we have the `6*@W` part. That means that there are 6 bytes which must be parsed by the `W` tag.

You get the picture?

Let's look at the `SWA` tag. It is defined as a bits tag. Hence it only parses bits and at most 8 of them. It has a mask defined as `0, 0, 1, 2*@IS, 3*@DP.` That means that most significant bit should be `0`, next one should also be `0`, next should be `1`, and then comes `2` bits which should be parsed by the `IS` tag, and then 3 lowest significant bits should be parsed by the `DP` tag.

Looking at the `DP` tag, you will see that is also a bits type tag, which makes sense since we are parsing a subset of bits from the `SWA` tag.

There are defined a number of possible `DP` bit masks, which, when matched, result in one or more expressions being executed.

So let's say that the 3 bits matches `1 0 1.` Then the expressions `WEIGHT_FACTOR=0.001` and `TARE_FACTOR=0.001` are both executed, essentially setting some values we can use later on, or explore from our program.
Notice the **[]**?
In `YAML` that is called an inline array, where each element is separated by a comma. That is the reason why I mention that two expressions are executed in this case, when the match is successful.

The `IS` tag follow a similar procedure as the `DP` tag. The `SWB` tag is an interesting one. It is used for parsing the 3rd byte of the data.
It is also a bits type mask, and contains 8 parts, one for each bit in the matched byte.
The most significant bit should be 0. .

Whatever the next bit is, is set in the value `IS_POWER_NOT_ZEROED`, which can then be used in other expressions or by the developer later on. Then a 1 bit must be available.
Next comes a bit, which if set, sets
`IS_UNIT_UK_POUND to 1`
and
`IS_UNIT_KILOGRAM to 0,`
else it sets `IS_UNIT_KILOGRAM to 1`
and `IS_UNIT_UK_POUND to 0.`

The next bit is set negated to the value `IS_SETTLED.`
So if the bit was 1, then `IS_SETTLED` is set to 0 and visa versa.

The 3 remaining bits sets
`IS_OVERLOAD,`
`IS_NEGATIVE` and `IS_NETTO` values.

Simple stuff, right?

Now let us look at the `W` tag.
It's defined to be a string type tag, which means that any masks we write must be written as strings, and any value matched is seen as a string (*a collection of bytes*). As the `W` tag does not have any masks defined, the `tag mask` is considered a match, and any optional expression on that `tag` is run. In this case we just set the value `WEIGHT` equal to the complete matched value.

That introduces the magic word `VALUE.`
It is a special variable, which always contains the latest match, regardless if it is a „bits" or „strings" match.
In this case, it is how we get the „Value"

When all matches have been successful, we have a matching `telegram,` and only then will all the matching `telegrams` expressions be run.
Internally **kbmMW Binary Parser** uses the **kbmMemTable SQL**- and **Expression parser**, and as such can do all the things that the expression parser can do, including calling functions etc.

We miss the code to run the parser:

```
rd:=TkbmMWBPFileReader.Create(eDefFile.Text);
  try
   rd.OnSkipping:=procedure(var AByteCount:integer)
         begin
          Log.Info('Skipping '+inttostr(AByteCount));
         end;

   // If you want to see the parsed values on a positive match.
   rd.OnMatch:=procedure(AValues:IkbmMWBPValues; var ABreak:boolean; const ASize:integer)
        var
         a:TArray;
         i:integer;
         row:integer;
        begin
          grid.DefaultRowHeight:=25;
          grid.RowCount:=AValues.Count+1;
          row:=1;
          a:=AValues.Names;
          TArray.Sort(a);
          for i:=0 to High(a) do
          begin
            grid.Cells[0,row]:=a[i];
            grid.Cells[1,row]:=VarToStr(AValues.Value[a[i]]);
            inc(row);
          end;
          if AValues.Value['IS_OVERLOAD'] then
           Log.Info('Overload')
          // else if AValues.Value['IS_SETTLED'] then
          // Log.Info('Unsettled gross:'+VarToStr(AValues.Value['GROSS']))
          else
           Log.Info('Gross:'+VarToStr(AValues.Value['GROSS']) +'
                          Settled:'+vartostr(AValues.Value['IS_SETTLED']));
          // ABreak:=true; // Only return first match.
        end;

   rd.Run(eDataFile.Text);
   Log.Info('Found '+inttostr(rd.MatchCount)+
       ' matches. Skipped '+inttostr(rd.SkippedBytes)+' bytes');
  finally
   rd.Free;
  end;
```

We take advantage of that a file reader is made available, that makes it easy to parse large files. But one could just as easily have created any other type readers, descending from **`TkbmMWBPCustomReader`**.
Each time the **parser** is not able to parse something successfully, it will attempt to skip past it, until either a match is made, or all data has been processed.
The OnSkipping event is called on those occasions.
When a match is made, the **`OnMatch`** event is called. The developer can choose what to do with the values and if the parsing should continue when the event is done.

The **file reader** accepts one argument, the name of the definition file.
And what is being read, is the file with the filename given in the Run statement.
Run will continue to run, until either all data has been read, or the process is interrupted, by either setting a zero value for **`AByteCount`** in **`OnSkipping`**, or setting **`ABreak`** to true in **`OnMatch`**.
After the execution ends, you can explore how many bytes were skipped and how many telegrams were read in total.

The parser is likely to evolve as new requirements appear, and I encourage users of it to play an active role in extending it so we all can benefit from a very versatile binary parser.

USER DEFINED FUNCTIONS AND KBMMEMSQL BY KIM MADSEN

starter — expert

Delphi

INTRODUCTION

As you may know, the widely used kbmMemTable also supports querying and more using a SQL92 like syntax via the component TkbmMemSQL. This article explains how to create UDF (user defined functions) that can be used as part of the SQL statement. Although UDF has been supported for a long time, it has been somewhat expanded in the upcoming release, why this article do reference features not available in the current kbmMemTable v. 7.76.

Standard UDF functions:
kbmMemTable already comes with a fairly extensive set of standard UDF functions which can be used as part of your query, but you can expand and enable and disable which UDF functions should be available for your purpose.

Include kbmSQLStdFunc in your projects uses clause to get access to all the standard UDF functions, which includes the following, grouped by their logical category names:

MATH.TRIG

- SIN(x)
- COS(x)
- TAN(x)
- LOG(x)
- LOG2(x)
- EXP(x)

MATH

TRUNC(x)	Returns integer part
FRAC(x)	Returns fractional part
MOD(x,y)	Return remainder after integer division
DIV(x,y)	Integer division
SQRT(x)	Square root
SQR(x)	Square. Same as x*x
ROOT(x,y)	Calculate y'th root of x.
MIN(x1..,xn)	Return minimum argument value
MAX(x1..,xn)	Return maximum argument value
AVG(x1..xn)	Return average value of all arguments
SUM(x1..,xn)	Return sum of all arguments
ABS(x)	Return absolute (*Non negative*) value of x
POW(x,y)	Return x in the power of y

STRING

UPPER(x)	Return uppercase value
LOWER(x)	Return lowercase value
TRIM(x)	Return value trimmed for leading and trailing spaces
MID(x,y,z)	Return z characters from x starting at y (*first character is 1*)
LEFT(x,y)	Return y leftmost characters of x
RIGHT(x,y)	Return y rightmost characters of x
LENGTH(x)	Return the length in characters of x

LEFTPAD(x,y,z)
Pad left of x with the character y until the complete result has a length of z

RIGHTPAD(x,y,z)
Pad right of x with the character y until the complete result has a length of z

CHR(x)
Convert the integer value x to a unicode character

POS(x,y)
Return the position in y, where the substring x is to be found. 0 is returned if x is not in y

REPLACE(x,y,z,v,w)
Search x for value y. When found, replace y with z. If v *(optional)* is true, then all occurrences are replaced. If w (optional) is true, then search is case insensitive

SPLIT(x,y,OUT z)
Search x for the sub string y. When found return the leading part as result or null if nothing found. If z is provided *(optional)* then the trailing part is returned in the variable given by z. *NEW IN 7.77*

DATETIME

NOW
Returns current date and time as a Delphi TDateTime type floating point value.

DATE(x)
Returns the date part of x which is a Delphi TDateTime type floating point value, as an integer. Same as TRUNC(somedate)

TIME(x)
Returns the time part of x which is a Delphi TDateTime type floating point value as a float. Same as FRAC(somedate)

YEAR(x)
Returns the year as an integer (2017) of the Delphi TDateTime type floating point value.

MONTH(x)
Returns the month as an integer (1 = Jan, 12=Dec) of the Delphi TDateTime type floating point value.

DAY(x)
Returns the day of month as an integer (1..31) of the Delphi TDateTime type floating point value.

HOURS(x)
Returns the hour value as an integer (0..23) of the Delphi TDateTIme type floating point value.

MINUTES(x)
Returns the minutes part as an integer (0..59) of the Delphi TDateTime type floating point value.

SECONDS(x)
Returns the seconds part as an integer (0..59) of the Delphi TDateTime type floating point value.

DATESTRING(x)
Returns the date of a Delphi TDateTime type floating point value as a formatted string according to the FormatSettings defined on the TkbmMemSQL component.

TIMESTRING(x)
Returns the time of a Delphi TDateTime type floating point value as a formatted string according to the FormatSettings defined on the TkbmMemSQL component.

CAST

CASTTODATETIME(x)
Casts x to a Delphi TDateTime floating point value. If x is a string, it will be parsed according to the FormatSettings.
CASTTOSTRING(x)
Casts x to a string.
If x is a TDateTime value, it will be converted to a date/time string value according to the FormatSettings.
CASTTONUMBER(x)
Casts x to a number. If x is a string, it will be converted to a floating point value according to the FormatSettings.

CONDITIONAL

IF(x,y,z,v)
Depending on x, either y, z or v *(optional)* will be returned. If x is true, then y will be returned. If x is false, then z will be returned and if x is null then v alternative z will be returned if v is not specified.
ISNULL(x)
Return true or false depending on if x is null.

CONVERSION

DATATYPE(x,OUT y, OUT z)
Parses the SQL style datatype (eg. VARCHAR(30)) given in x, and returns the Delphi TFieldType best matching as an integer or NULL if the given SQL datatype is invalid. If y is provided (optional), the size of the declaration (eg 30) is returned in the referenced variable. If z is provided (optional), the precision of the declaration (eg 0) is returned in the referenced variable. *NEW IN 7.77*

Please take notice to the functions category names. They can be used to enable or disable whole groups of functions. E.g.

```
kbmSQLFunctionRegistrations.DisableGroup('MATH.TRIG')
```

would result in all the trigonometrical functions being unavailable for SQL expressions.

Obviously there is also an EnableGroup function. All registered functions are default enabled. If you want to only disable a certain function, you can use:

```
kbmSQLFunctionRegistrations.DisableFunction('MONTH')
```

Now the MONTH function will not be available for SQL expressions.

CREATING A NEW CUSTOM UDF FUNCTION

It is pretty easy to create a custom UDF function. Basically you will need to create a globally accessible function, and register it to kbmMemSQL.

Typically an UDF takes zero or more arguments and until and including v. 7.76 returns one single value. The UDF function is usually called many times depending on the actual SQL statement.

Let us look at how the SIN standard function is implemented.

```
function SQLSin(const
    AOperation:TkbmSQLCustomOperation;
    const ASituation:TkbmSQLFunctionSituation;
    const AArgs:TkbmSQLNodes;
    var AResult:variant):boolean;
begin
  kbmSQLCheckArgs(AArgs,1);
  case ASituation of  fsWidth:
     AResult:=0;
  fsExecute:
    begin
       AResult:=AArgs.Node[0].Execute;
    if not VarIsNull(AResult) then
       AResult:=Sin(AResult);
    end;
  fsDataType:
    AResult:=ftFloat;
  end;
  Result:=true;
end;
```

The SIN UDF function takes 4 arguments:

AOperation is the current operation the function is being called from (it can be a select, insert etc. operation). This argument can usually be ignored, unless you want to get to the FormatSettings which is available as a property in the operation instance.

ASituation which indicates which situation the UDF is being called in. It can be one of fsWidth, fsDataType or fsExecute. The UDF will be called during compilation of the SQL statement to determined what type of data it will return, and what it expects the width of the returned data to be. And it will finally be called a number of times to execute the actual function at SQL execution time.

AArgs provides the arguments for the UDF function. Your function may require a minimum number of arguments, and to check for that it is recommended to use the kbmSQLCheckArgs function as shown in the example. The SIN function only expects one argument.

AResult is the parameter that should receive the outcome of the function. It must adhere to the data type which the UDF function provided at compilation time.

The SIN function is registered to kbmMemSQL by calling RegisterFunction, typically in the units Initialization section. E.g:

```
initialization

kbmSQLFunctionRegistrations.RegisterFunction(
'MATH.TRIG','SIN',@SQLSin);
```

Check the kbmSQLStdFunc.pas unit to see the how other standard functions has been implemented and registered. The SIN function can for example be used like this:

```
SELECT SIN(field1) AS "Sineof", field1
FROM sometable
```

But it can also be part of the condition and any other place in the SQL statement, where an expression can be given. But what if you want to split a value into multiple values using an UDF function, how is that done? For example an UDF splitting a string in two parts.

SPLIT(x) Lets say we want to split x into two parts, exactly where there is a colon (:), but how will we get 2 strings back when we can only return one? From kbmMemTable v. 7.77 it is possible *(optionally)* to use a sligltly different syntax in the SQL statement when calling UDF functions.

```
SELECT SPLIT(field1, OUT $var1)
AS "Left", $var1 as "Right"
FROM sometable
```

The trick is to use the new OUT syntax. Now we will let the split function return all the data left of the colon (:), or if no colon is found all the data in x, and send the remaining data back as the contents of a variable called var1.

And register it:

```
initialization
kbmSQLFunctionRegistrations.RegisterFunction('
    MYFUNCS','SPLIT',@SQLSplit);
```

You can have as many OUT variables as you need, but remember that the function MUST return one value in the old fashioned way.

Also notice that the variable names must start with $, and that the variable values are not available until the UDF function has been called. Usually it's thus recommended to call the UDF function as early as possible in your SQL statement, so the variables are available for the remainder of the current statement processing.

The variables are automatically cleared before attempting to process another record.

If you create some cool, can't live without, UDF functions you think ought to be part of the standard SQL function library, ping us.

```
function SQLSplit(const AOperation:TkbmSQLCustomOperation;
    const ASituation:TkbmSQLFunctionSituation;
    const AArgs:TkbmSQLNodes; var AResult:variant):boolean;
var v: variant; s:string; i:integer;
begin
  kbmSQLCheckArgs(AArgs,2);
  case ASituation of fsWidth:
    begin
      AResult:=AArgs[0].Width;
    end;
    fsExecute:
    begin
      v:=AArgs[0].Execute;
      if VarIsNull(v)
      then Result:=Null
      else
        begin
        s:=v;
        i:=pos(':',s);
        if i<0
        then AResult:=s
        else
          begin
            AResult:=copy(s,1,i-1);
            kbmSQLSetVariableValue(AArgs[1],copy(s,i+1,length(s)));
          end;
        end;
    end;
  fsDataType:
    begin
      kbmSQLSetVariableMetaData(AArgs[1],ftString,AArgs[0].Width);
      AResult:=ftString;
    end;
  end;
  Result:=true;
end;
```

REST EASY WITH KBMMW PART 8
DATABASE 3 BY KIM MADSEN

starter — expert

Delphi

In previous articles we have seen how easy it is to use kbmMW's ORM to maintain database structures and access and manipulate data.

Next release of kbmMW continues to extend on the ORM with additional features designed to make typical chores easy.

Most often, you want records to disappear from a table the moment you call ORM's delete methods.

But sometimes, for example when you are referencing the record from other tables, it would be nice to retain the record silently, but keep it out of the result set of most queries.

Some databases supports this the hard way by using referential integrity, which means you will simply not be allowed to delete the record if some other table reference it.

That leaves it entirely up to the developer to manually manage when to show the record and when not to.

kbmMW's ORM takes a slightly different approach to the problem. What if you define a field in the class that will flag if the record is to be considered deleted or not, and tell kbmMW that it should take that field into account when operating that particular class?

Next version will allow you to do just that. So instead of actually deleting the record, kbmMW will automatically flag it as deleted, AND automatically keep the record out of the queries result sets, regardless if you use higher level methods to operate the data, or lowerlevel kbmMW SQL statements.

The following is a sample class holding images and other stuff. Notice the kbmMW_Table attribute. It now also contains some new settings:

defaultDeleteMethod:mark
deleteMarkProperty:Default
deleteMarkValue:true

```
[kbmMW_Table('name:image,
defaultDeleteMethod:mark,
deleteMarkProperty:Deleted,
deleteMarkValue:true')]
TImage = class // this unit is called uData
private
  FID:kbmMWNullable;
  FDescription:kbmMWNullable;
  FPersonID:string;
  FBlob:TMemoryStream;
  FDeleted:boolean;
protected
  procedure SetBlob(AValue:TMemoryStream);
virtual;
public
  constructor Create; virtual;
  destructor Destroy; override;

  [kbmMW_Field('name:id, primary:true,
        generator:shortGuid',ftString,40)]
  [kbmMW_NotNull]
  property ID:kbmMWNullable read FID write FID;

  [kbmMW_Field('name:personid',ftString,40)]
  [kbmMW_NotNull]
  [kbmMW_Null('')]
  property PID:string read FPersonID
                      write FPersonID;

  [kbmMW_Field('name:description',ftString,30)]
  property Description:kbmMWNullable
        read FDescription write FDescription;

  [kbmMW_Field('name:blob',ftGraphic)]
  [kbmMW_NotNull]
  property Blob:TMemoryStream
        read FBlob write SetBlob;

  [kbmMW_Field('name:deleted',ftBoolean)]
  property Deleted:boolean
        read FDeleted write FDeleted;
end;
```

Default value for **defaultDeleteMethod** is delete or default which means the same… just delete the record (*the usual way*).

However when setting it to mark, we tell **kbmMW** that we want **kbmMW** to mark the deletion of records by setting the value of a specific field to a specific value. Basically it can be most (non blob) types of field, and the value can also be chosen at will. In our case, we indicate that we want the Deleted boolean property to be the one that indicates if a record is deleted or not.

The moment we use **ORM.Delete(…)** on an TImage instance, the corresponding record will be marked as deleted, and if we search for TImage instances in the database using **ORM,** the deleted ones will not be returned to us.

So from the perspective of the developer, the delete seems to operate exactly the same as before, with the big exception, that the records are in fact still available in the table. The **ORM** just hides the deleted ones from us.

In other situations, you might actually want to delete the record the old fashioned way, but you would like also to have a copy of the deleted record in a backup or audit table.
This can also easily be done in kbmMW by setting defaultDeleteMethod to move.

```
[kbmMW_Table('name:image,
defaultDeleteMethod:move,
deleteMoveToTable:uData.TBackupImage')]
TImage = class
...
end;

[kbmMW_Table('name:backupImage')]
TBackupImage=class(TImage)
public
  [kbmMW_Field('name:id,
primary:true',ftString,40)]
  [kbmMW_NotNull]
  property ID:kbmMWNullable read FID write FID;
end;
```

Notice that we also provide a `deleteMoveToTable` setting, which points to the fully qualified name of the class that should act as the backup/audit table.

Each time we use **kbmMW** to delete a record, it will first be inserted into the database table named backupImage, and only then be deleted from the image table.

All this will run in a single transaction, and if things goes wrong, a rollback will be forced to ensure the consistency between the backup table and the actual table is not violated during the insert/delete operation.

In the above example we have redefined the id field's attributes, to ensure that there is no generators defined on it.
The reason is that using **kbmMW's SQL** statements to delete a record will automatically be rewritten to select into and delete.
If any generator fields were defined, those would not be populated in such scenario, that's why kbmMW would raise an exception if generator fields were found.

However if you do not use **kbmMW's SQL** statements to delete records, its perfectly legal to have additional generator fields in the backup table. Those will then be populated automatically the usual way.

If you really really want to delete a record on a table where the class are using one of these alternative delete methods, you can add an option to make kbmMW just delete the record.

```
ORM.Delete(o,nil,[mwqoIgnoreDeleteModification]);
```

Assuming o is an instance of **uData.TImage,** and o is actually to be found in the table, then the record corresponding to o will be deleted from the table the old fashioned way.

The same option can be given to the Query methods, to allow showing a record that otherwise would not be returned, because it was marked as deleted.

If the move method was used, then you will have to query using the **TBackupImage** class to get to the deleted *(but backed up)* records.

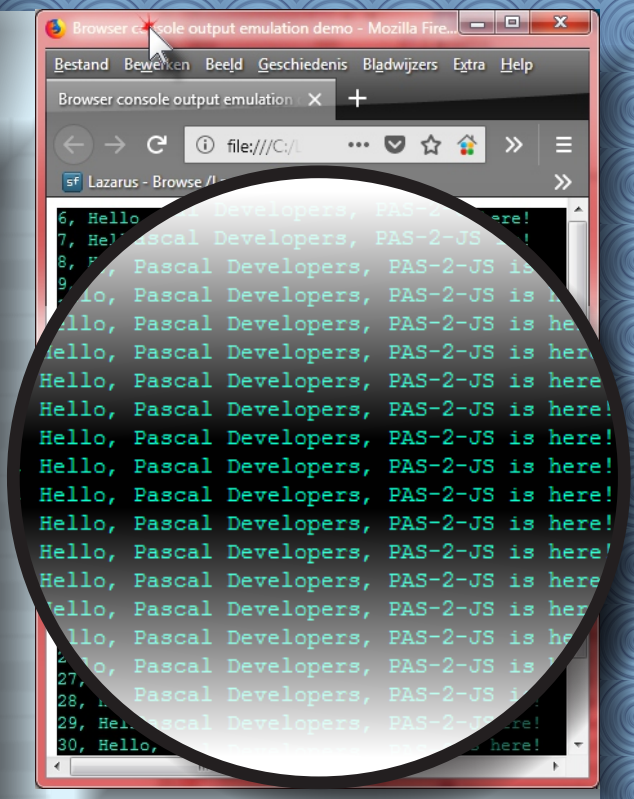
BY DETLEF OVERBEEK
TECHNIQUE BY MICHAEL VAN CANNEYT
AND MATTIAS GÄRTNER

INTRODUCTION

This is about the first edition of the beta version of Pas2JS. It's quite a long story, but here it is.
You can use it and test it. All you need is FPC, Lazarus and the compiler.
Since it all works under FPC without a graphical User Interface (for the moment) – the components will soon be available – you will have to create it all on the fly. But to make it easy for you we (Mattias Gärtner and Michael van Canneyt) have written a number of sample projects so you can try them and create your own projects.
In this article I will try to explain how to do that.

INSTALLATION

First simply start by downloading the zip file from your Download Page at our website:
downloads.blaisepascal.eu/pas2js.i386-win32.zip
(this is a download address). For future purposes and updates of the windows version you can go to:
ftp://ftpmaster.freepascal.org/fpc/contrib/pas2js/0.8.41/pas2js.i386-w
for future changes and updates you can go to:
http://wiki.freepascal.org/Pas2JS_Version_Changes
If you want to find the project you can do so at
http://wiki.freepascal.org/pas2js



If you want to become an official Beta user for Pas2JS please let us know:
editor@blaisepascalmagazine.eu
After you have downloaded the file you can start to unpack the zip file. To make it easier you better follow for the first time exactly what we did here: We created a directory on Disk C because that is the easiest for standard purposes. Here is an overview of the content of the zip file:

```
c:\LazPas2JS\                                  // Main dir
c:\LazPas2JS\bin\
            \i386-win32
            \i386-win32\backup\
            \i386-win32\pas2js.cfg             // the file you can see as code in listing 1
            \i386-win32\pas2js.exe             // the file you need for compiling
            \i386-win32\pas2jslib.dll
c:\LazPas2JS\examples\                         //← here are the sub dirs for examples
c:\LazPas2JS\examples\pas2js\
c:\LazPas2JS\examples\pas2js\fcldb\
c:\LazPas2JS\examples\pas2js\fpcunit\
c:\LazPas2JS\examples\pas2js\fpreport\
c:\LazPas2JS\examples\pas2js\hotreload\
c:\LazPas2JS\examples\pas2js\jquery\
c:\LazPas2JS\examples\pas2js\rtl\              //← all the projects you will be shown in this
article
c:\LazPas2JS\fpmkinst\                         //← only for pas2Js projects needed
c:\LazPas2JS\pas2js\                           //← only for pas2Js projects neededPlease set
                                                  it up as we did here.
```

Create a directory called "LazPas2JS" **(C:/LazPas2JS)**.
All the content of the zip file needs to be put in this Dir and it will look after that like this:

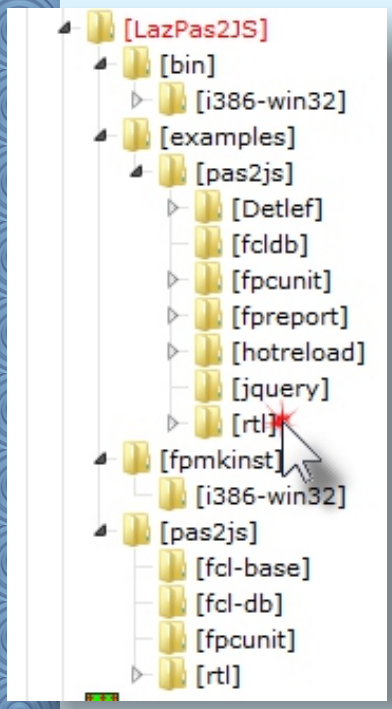


Figure 2:
the logo overview of the Directories of the whole project.

The example directory you find at the mouse cursor. We will show the content and examples in a later stage, in the chapter Examples.
To follow this exactly has a reason: the **pas2js.cfg** file has been adapted for this:

listing 1

```
#
# Minimal config file for pas2js compiler
#
# not yet implemented: -d is the same as #DEFINE
# not yet implemented: -u is the same as #UNDEF
# Write always a nice logo ;)
-l
# Display Hints, Warnings and Notes
-vwnh
# If you don't want so much verbosity use
#-vw
-FuC:\LazPas2JS\pas2js\rtl  // this is as you can see the path
-FuC:\LazPas2JS\pas2js\fcl-base
-FuC:\LazPas2JS\pas2js\fcl-db
-FuC:\LazPas2JS\pas2js\fpcunit
#IFDEF nodejs
-Jirtl.js
#ENDIF
# end.
# end
```

There is of course a simple way to make the path variable.

```
-Fu$CfgDir/..\..\..\pas2js\..\fpc-pack\pas2js/rtl
-Fu$CfgDir/..\..\..\pas2js\..\fpc-pack\pas2js/fcl-base
-Fu$CfgDir/..\..\..\pas2js\..\fpc-pack\pas2js/fcl-db
-Fu$CfgDir/..\..\..\pas2js\..\fpc-pack\pas2js/fpcunit
```

LAZARUS CONFIGURATION

To be able to work with pas2js, the Lazarus IDE needs to be told where the source files are. This can be done by opening several packages that are part of pas2js.
Its a simple procedure:

First of all start **Lazarus,** if you still need to select a language : go to tools, choose **Options General,** you'll find **Language;** choose the one you want.➔ **Ok.**

Now select the **'Package'** menu from the main menu, and choose **'Open Package File':**

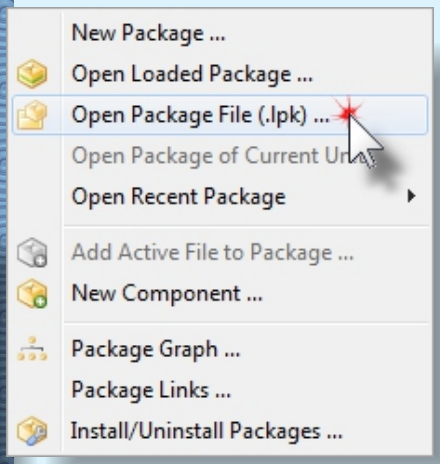


Figure 3: Open the Package file.

Click Open Package file:

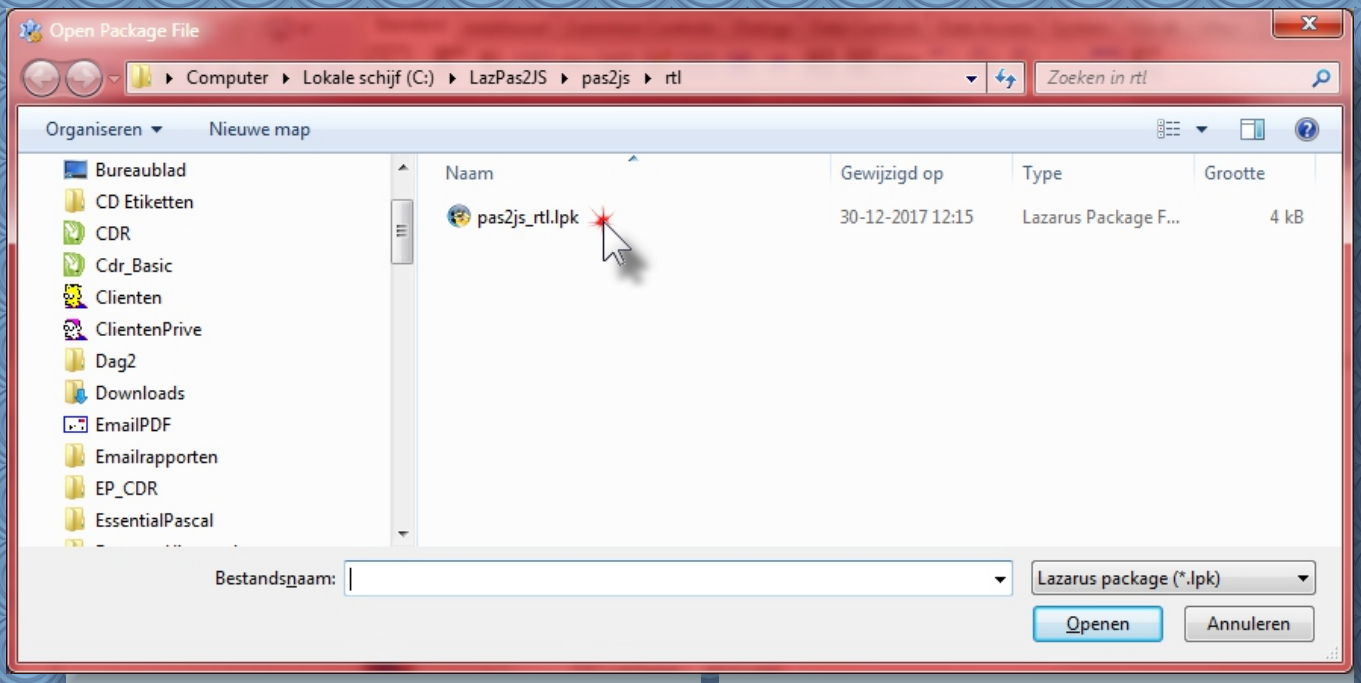


Figure 4: The Package file.

The path for the file to open is `c:\LazPas2JS\pas2js\rtl\pas2js_rtl.lpk`. Double click and the package overview will appear:

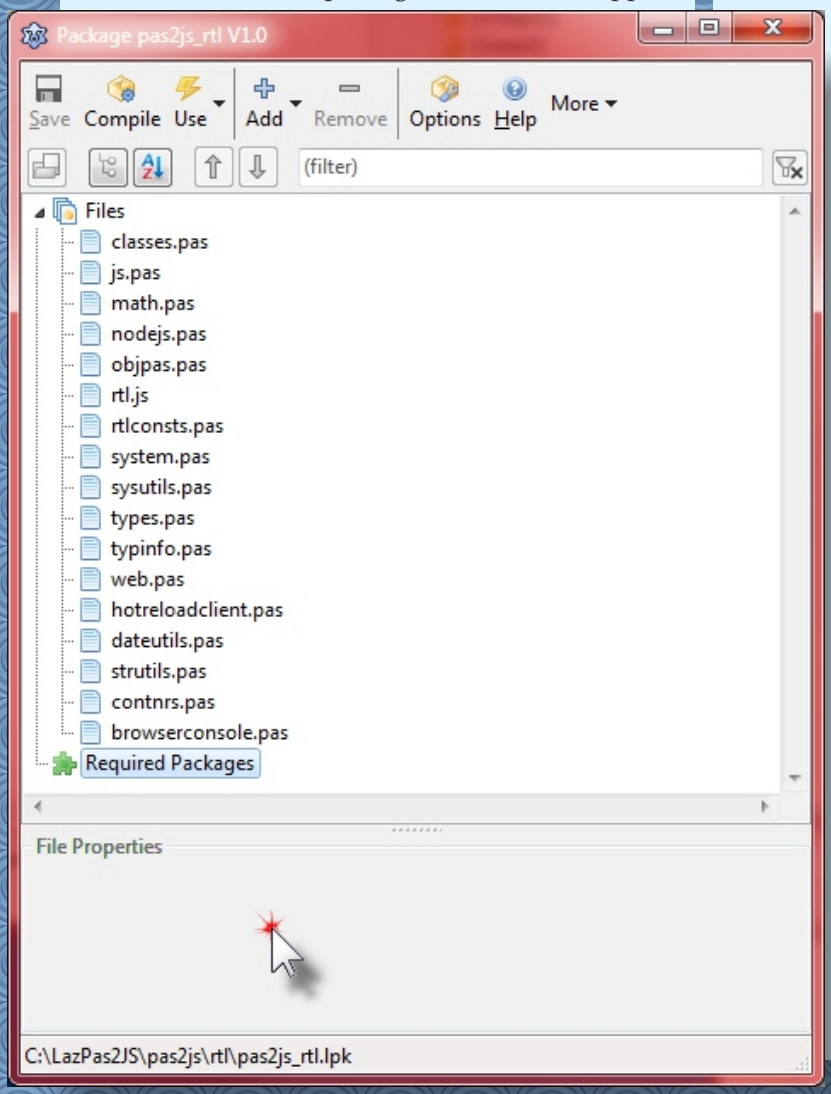


Figure 5: Overview of the needed pas files

This must be done for 3 more packages:

```
c:\LazPas2JS\pas2js\fcl-base\fcl_base_pas2js.lpk.
c:\LazPas2JS\pas2js\fcl-db\pas2js_fcldb.lpk.
c:\LazPas2JS\pas2js\fpcunit\fpcunit_pas2js.lpk.
```

After this, you can add these packages as dependencies to your **pas2js** projects, and **lazarus** will know where to find the units that you use in your project.

CREATING A NEW PROJECT

Creating a **Pas2JS** project has been completely automated in the development version of lazarus (*IDE package pas2jsdsgn*). The steps outlined here show how to do all the needed work manually.
A **Pas2JS** project is actually a simple program.
So, start Lazarus and go to → **File** → **New** and select **Simple Program.**

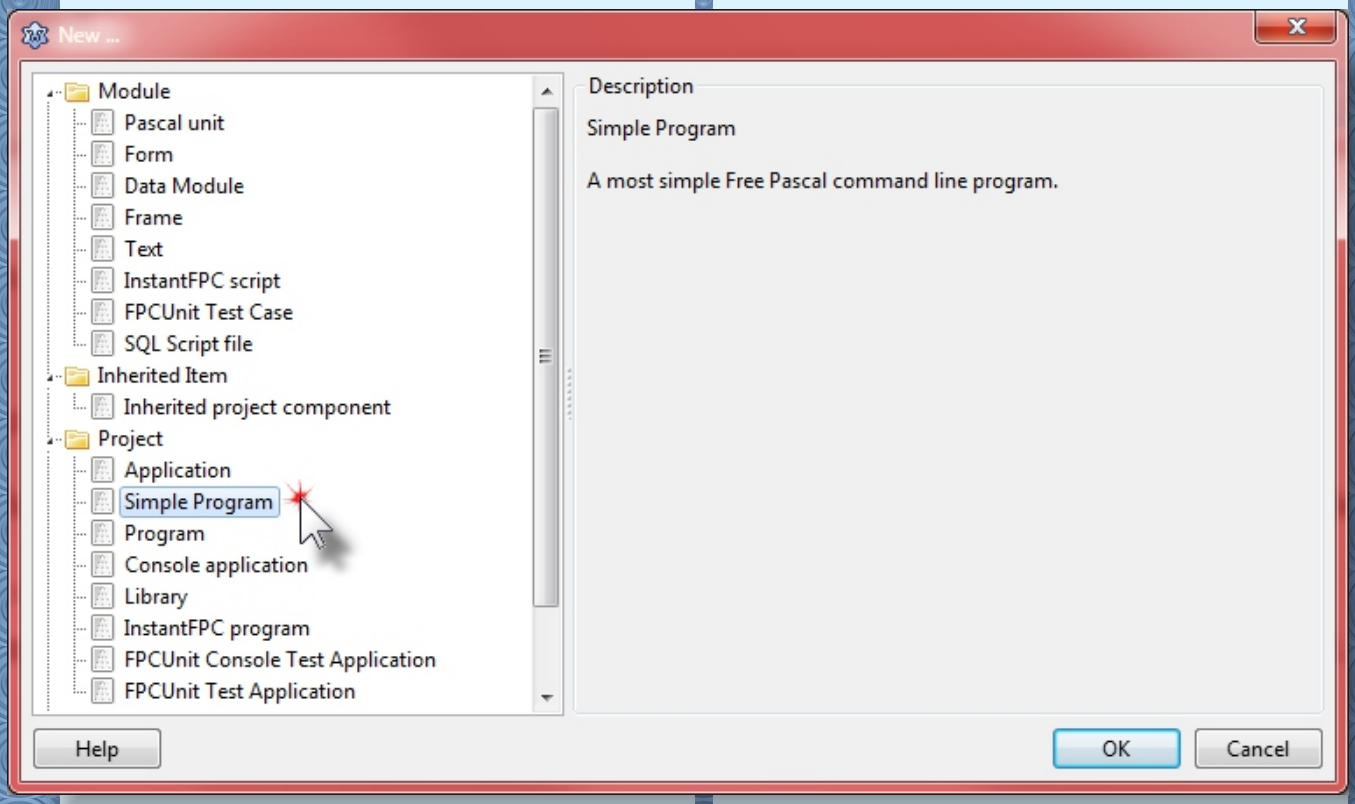


Figure 6: Choose a simple program

It's best to save the new project at once, preferably in a separate directory.
You best create a Directory of your own choice: What I did was first create a directory in the already known examples/rtl directory:
`C:\LazPas2JS\examples\pas2js\rtl\Detlef.`

CONFIGURING THE PROJECT COMPILER.

A **Pas2JS project is NOT COMPILED using the regular FPC compiler,** as configured in the global options of the IDE. Instead, we need to tell the IDE that for this particular project, it should do this with **Pas2JS,** and that **it should not bother to call the regular compiler.** This can be done using the **'Compiler commands'** configuration of the project.

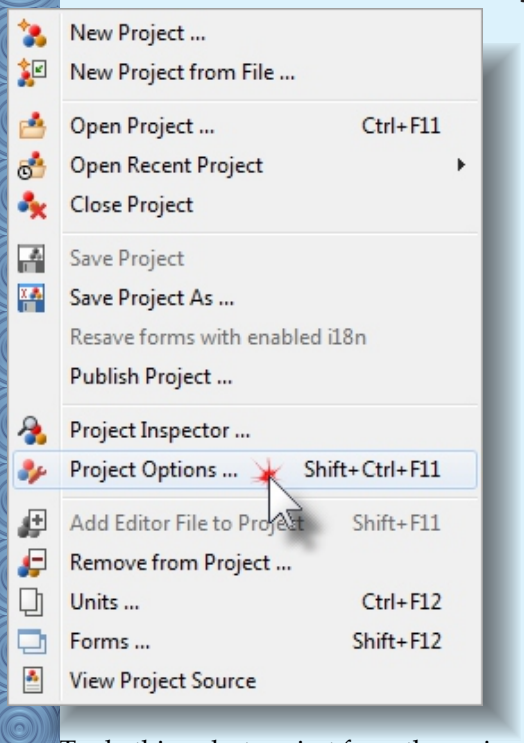


Figure 6: Go to Project Options

To do this, select project from the main menu, and choose **Project Options**. From here a new window will open:

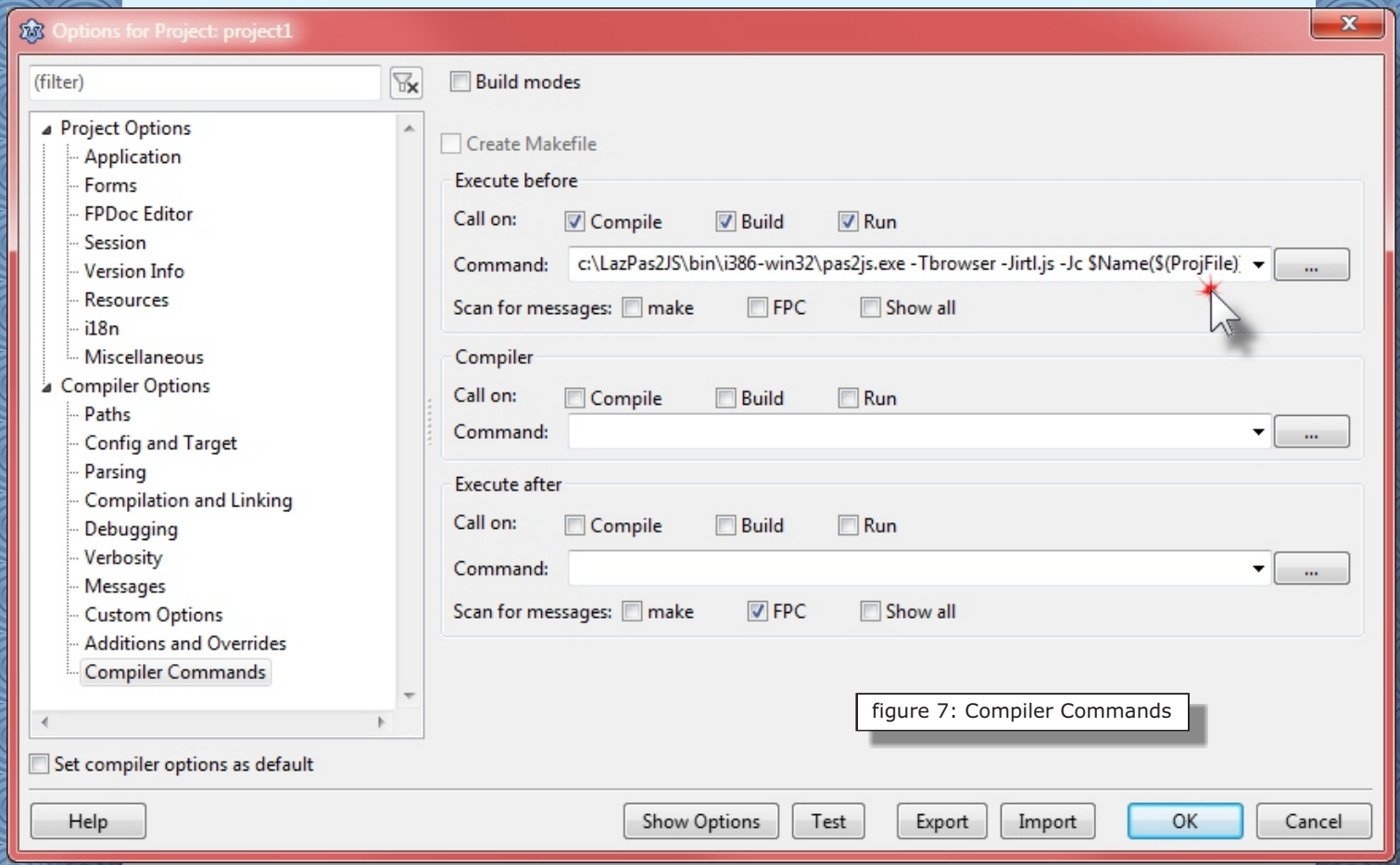


figure 7: Compiler Commands

The section we need is located under **Compiler Options**. One of the last line at the bottom is **Compiler Commands.** Here, 2 things mush be configured:

1. **Under 'Compiler',**
   the edit box **'Command'** must be cleared, and the 3 **checkboxes** after **"Call on"** must be **unchecked.**
2. **Under 'Execute before',**
   the 3 **checkboxes** after **"Call on"** (**Compile, Build, Run**) must be checked.

The command to enter in "execute before" is:

```
c:\LazPas2JS\bin\i386-win32\pas2js.exe -Tbrowser -Jirtl.js -Jc $Name($(ProjFile))
```

The actual path to the **Pas2Js** executable can differ if you installed it in another location.

ADDING THE PAS2JS COMPILER TO THE PATH

If you are tired of putting in the path all the time you might also change your Operating System PATH variable:
For **WIN7**: go to → **System** → **Advanced Options** → **Environment Variables** → **Edit**
Insert the lines you want after what is already there: add a **semicolon** (;) then add

```
c:\LazPas2JS\bin\i386-win32\
```

at the "command" line remove all text and replace it with

```
$MakeExe(pas2js) -Tbrowser -Jirtl.js -Jc $Name($(ProjFile))
```

For WIN10: go to → **System** → **Advanced options** → **Environment Variables** → **Edit**
Insert the lines you want after what is already there: add a semicolon (;) then add

```
c:\LazPas2JS\bin\i386-win32\
```

If you have done this, the "command" line to be entered in the 'Execute before' options of the project can be replaced with

```
$MakeExe(pas2js) -Tbrowser -Jirtl.js -Jc $Name($(ProjFile))
```

FOR WIN 10: SOME CONFIGURATION - ILLUSTRATIONS

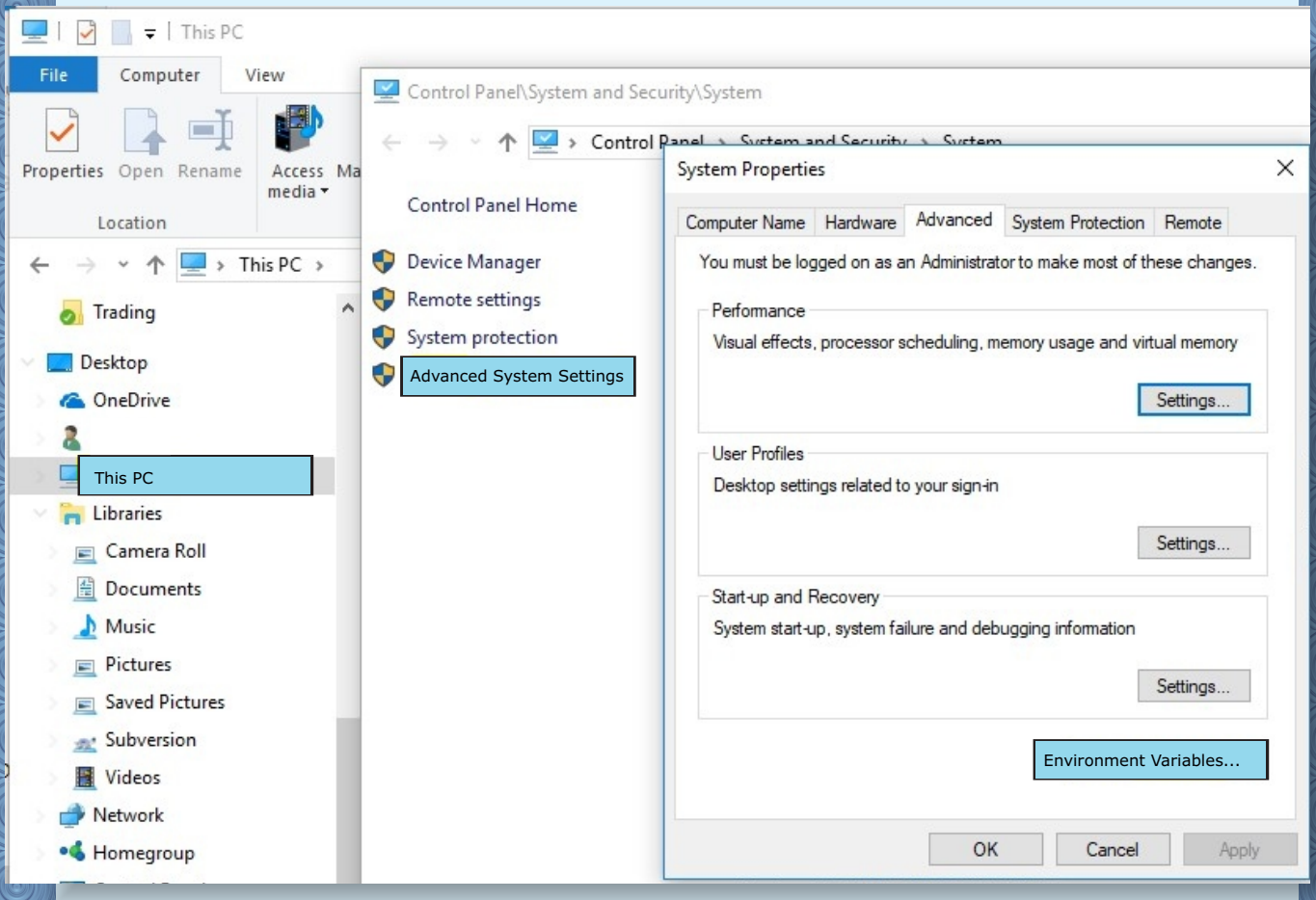


Figure 8: Advanced System Settings -Environment Variables

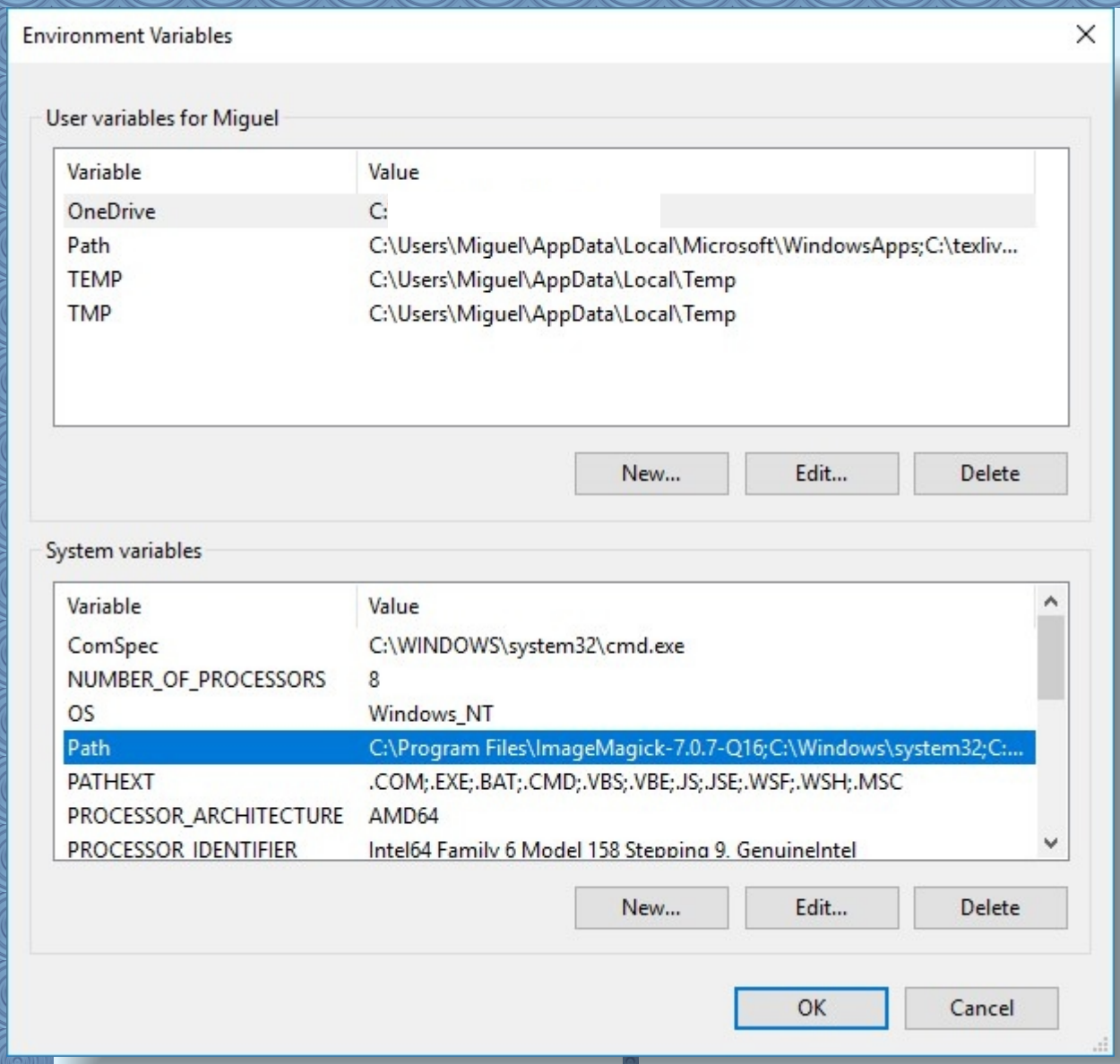


Figure 9: The Path you want to alter.

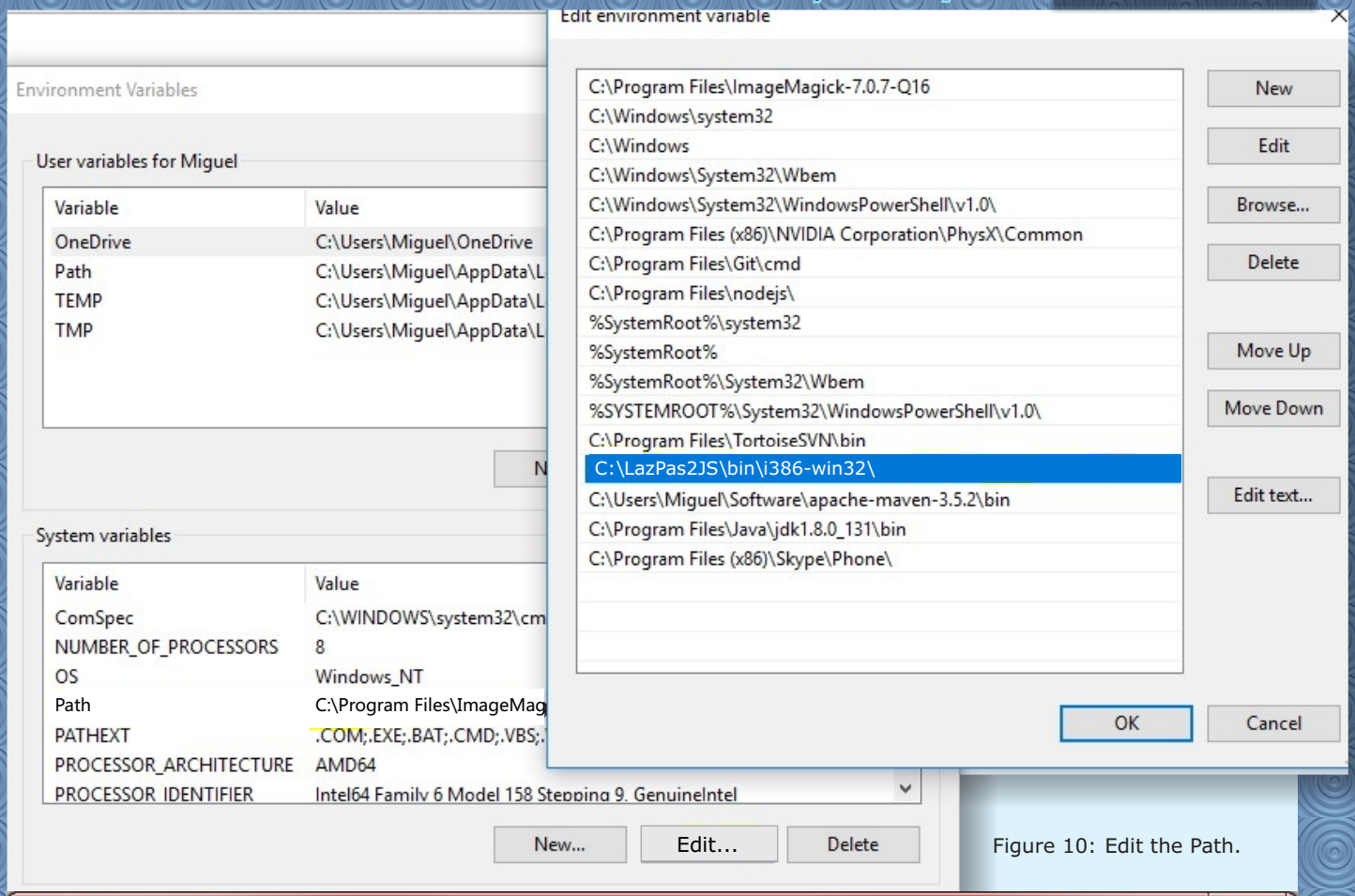


Figure 10: Edit the Path.

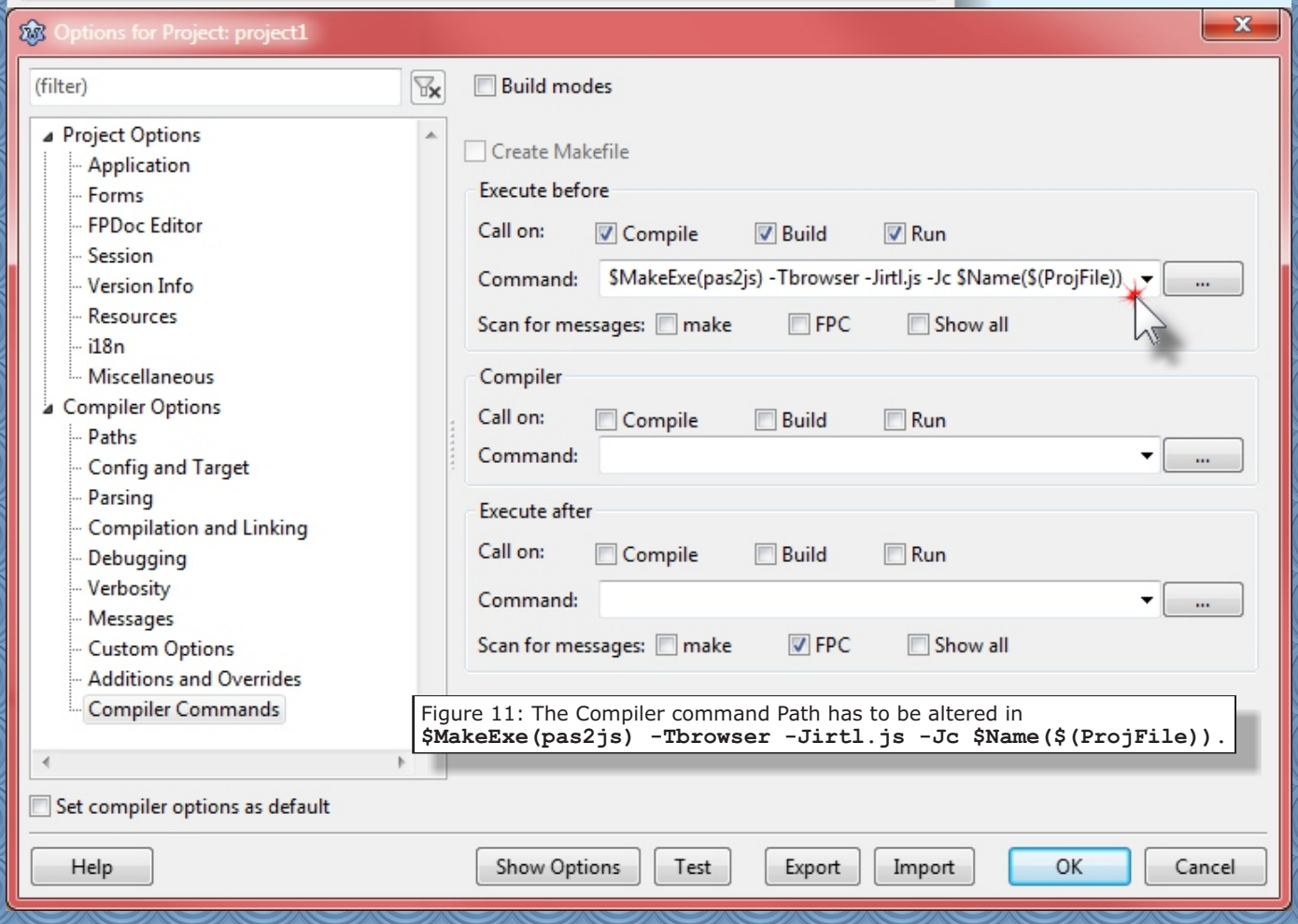


Figure 11: The Compiler command Path has to be altered in `$MakeExe(pas2js) -Tbrowser -Jirtl.js -Jc $Name($(ProjFile))`.

COMPILING THE PROJECT

Once this is all done, you can compile the project using the **'Compile'** or **'Build'** commands under the **Run** menu. If all goes well, a green line should appear. If you attempt to run the project, this will fail, **because there is no .exe generated, only a `.js` file.**

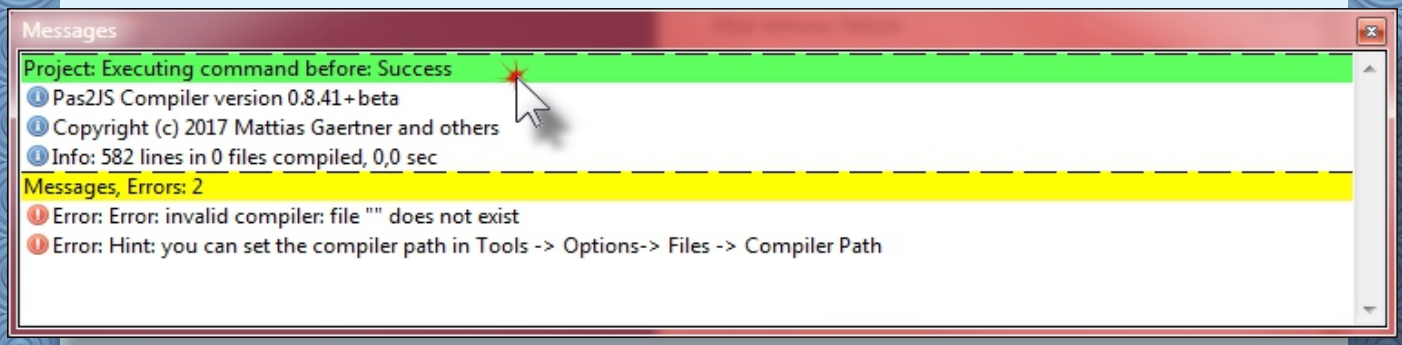


Figure 12: The result: it works

This can be easily checked in a **file explorer**.

If the project compiled correctly, the project directory should look like this:

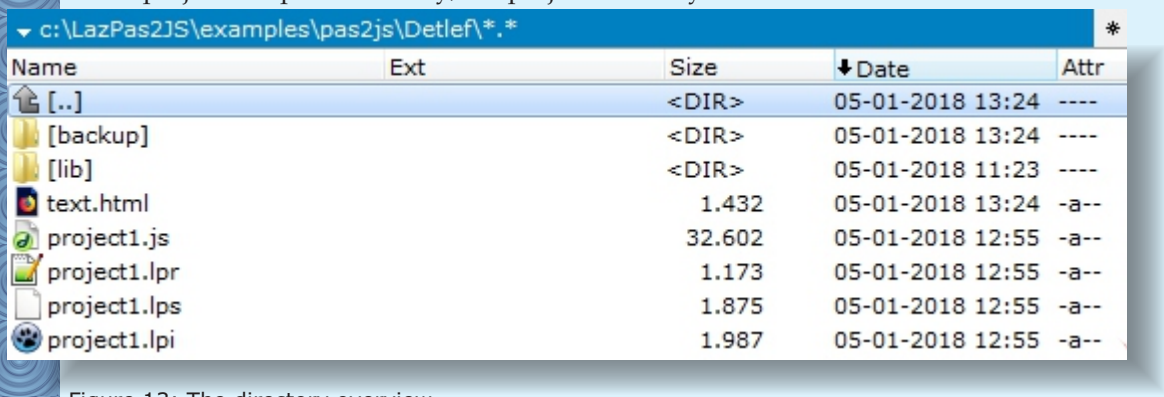


Figure 13: The directory overview

Notice that a **`project.js`** has been created. You also can see that there is a file called text.html.

THE PROJECT HTML FILE

CREATING A HTML FILE

A **Pas2JS project** runs in the browser. To be able to run it in the browser, a **HTML** page is needed.

You should consider this **HTML** file as part of your project. To create a **HTML** file and add it to the **Lazarus project**, go to the First create a text file using **File → New**, and choose **'Text'** below **'Module':**

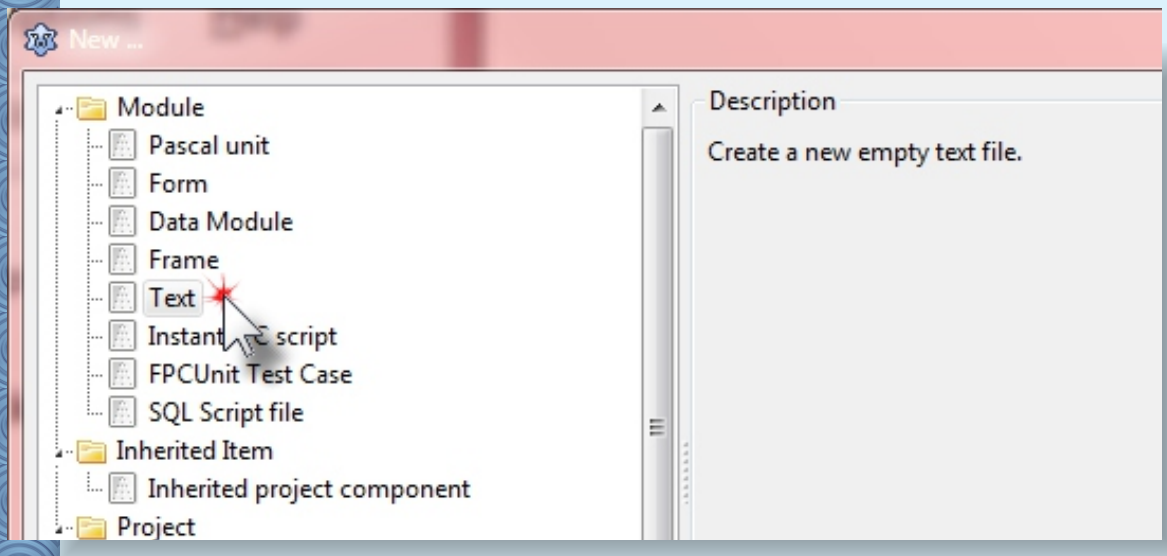


Figure 14: Now there will be a text file created.

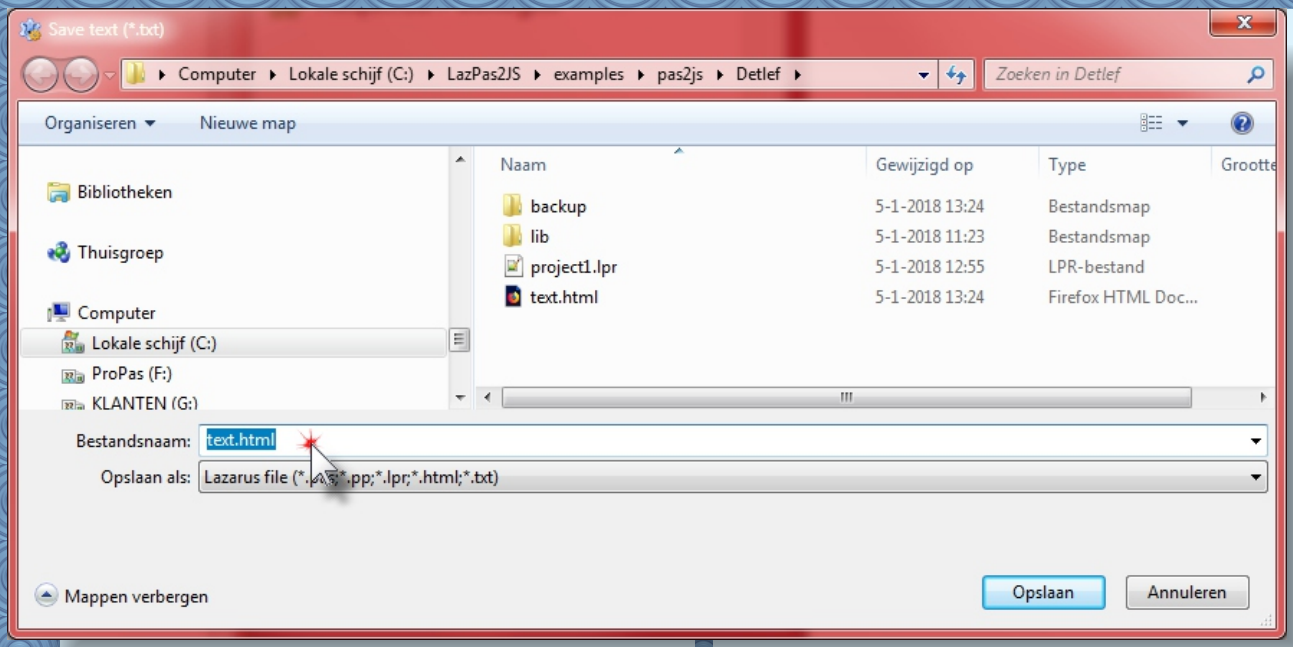


Figure 15: Save as HTML

After that save the file as text.html.
The **Project Inspector** will now show the file:

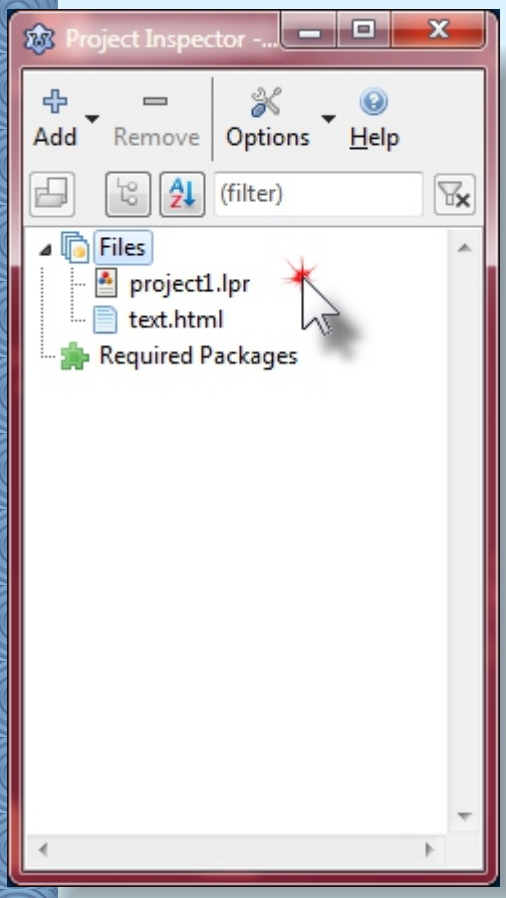


Figure 16: Project overview

HTML CONTENTS

The **HTML** file of course needs to have some content. You can edit the file in the Lazarus IDE (*it has highlighting for HTML and CSS*) , but of course you can use any text editor. The minimal content required to run a pas2JS program is as follows:

```
<html>
  <head>
    <title>Collection demo</title>
    <script type="application/javascript" src="project1.js"></script>
  </head>
  <body>
    <script type="application/javascript">
     rtl.run();
    </script>
  </body>
</html>
```

There are 2 important things in the **HTML** file

1. The **javascript** file generated by the compiler must be included, using a script tag:

```
<script type="application/javascript" src="project1.js"></script>
```

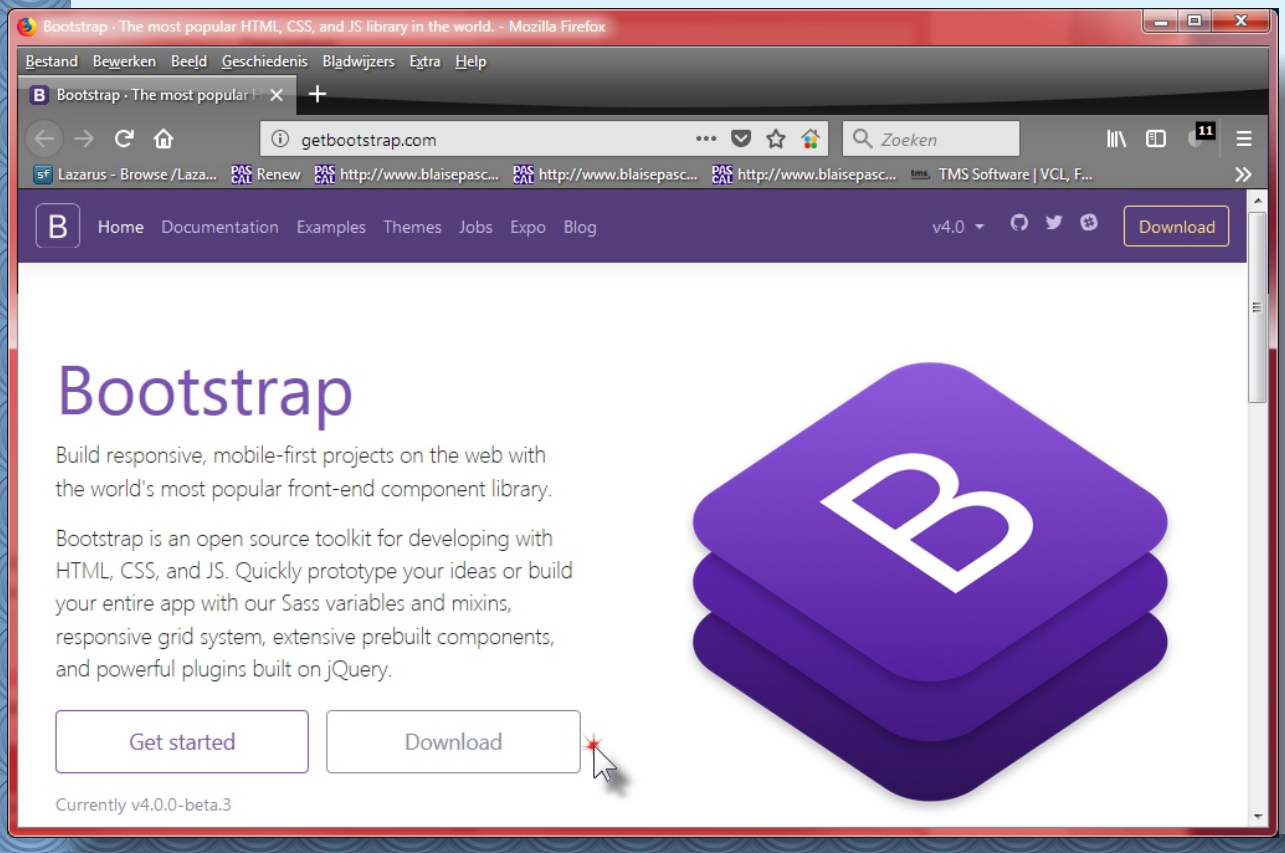
2. The program must be started in the body of the HTML document by invoking rtl.run():

```
<script type="application/javascript">
     rtl.run();
    </script>
```

That is all that needs to be done. To run the program, you can just open the HTML file in the browser: it does not need to be on a web server, it can be opened in the explorer.

ADDING A STYLE

A typical pas2js program will create HTML on the fly. To make the HTML look good, it must be styled. For style, if you want you can use Bootstrap for CSS. It has many advantages : it has lots of options, looks good and supports responsive design. if you want to find out go to **http://getbootstrap.com/.**



BOOTSTRAP
is a free and open-source front-end **web framework for designing websites** and **web applications**.
It contains **HTML-** and **CSS-**based design templates for typography, forms, buttons, navigation and other interface components, as well as optional **JavaScript** extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Features
Bootstrap 3 supports the latest versions of the **Google Chrome, Firefox, Internet Explorer, Opera,** and **Safari** .
It additionally supports back to **IE8** and the latest **Firefox Extended Support Release (ESR)**.
Since 2.0, **Bootstrap** supports responsive web design.
This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (*desktop, tablet, mobile phone*).
Starting with version 3.0, Bootstrap adopted a mobile-first design philosophy, emphasizing responsive design by default.
The version 4.0 release added Sass and flexbox support. If you would like to dive in:
just download it.

NOTEPAD ++
To edit **CSS** or **HTML** there are quite a lot of options:
One of my favourites is **Notepad ++** which can be downloaded from here:
`https://notepad-plus-plus.org/.`
I have created some illustrations to allow you to see the ease of use for this: If you really want to create some gorgeous designs, you can do it with this one, go and play with it. For creating good looking stylesheets I will write in the next issue.

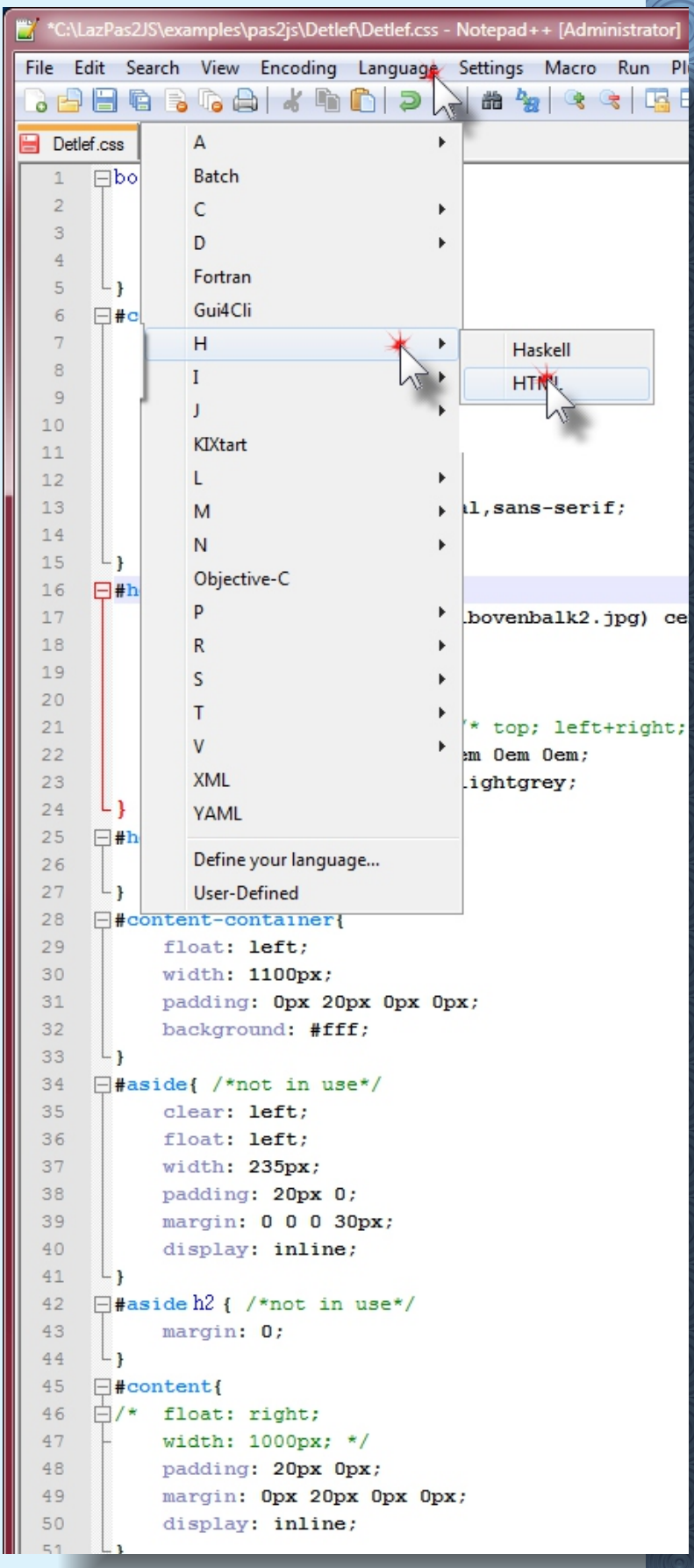


Figure 18: Choose The Notepad ++ HTML page

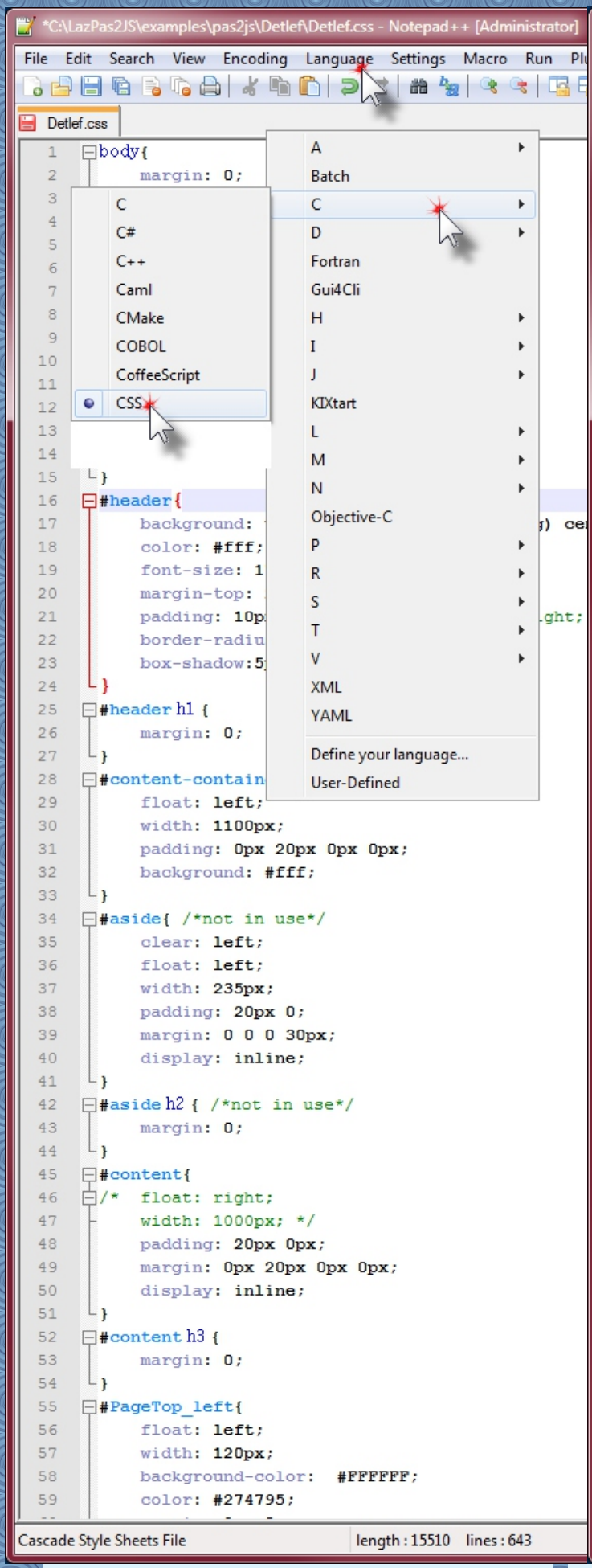


Figure 19: Choose The Notepad ++ CSS page

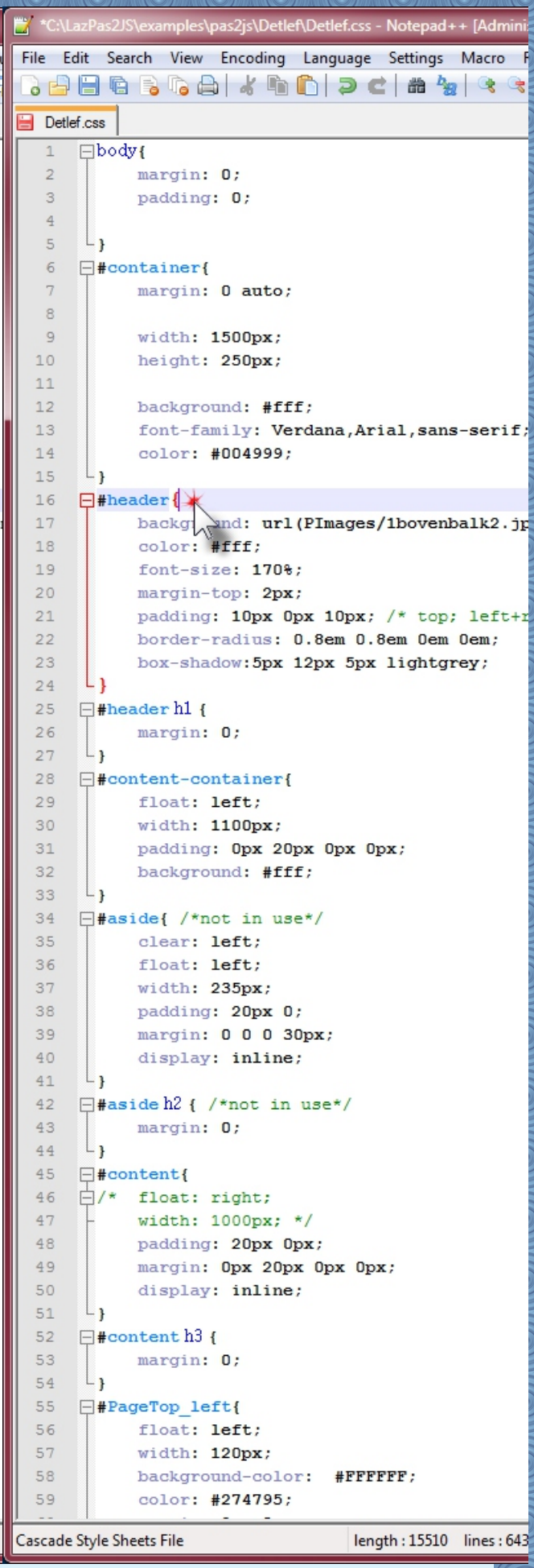


Figure 20: Detail of the CSS page

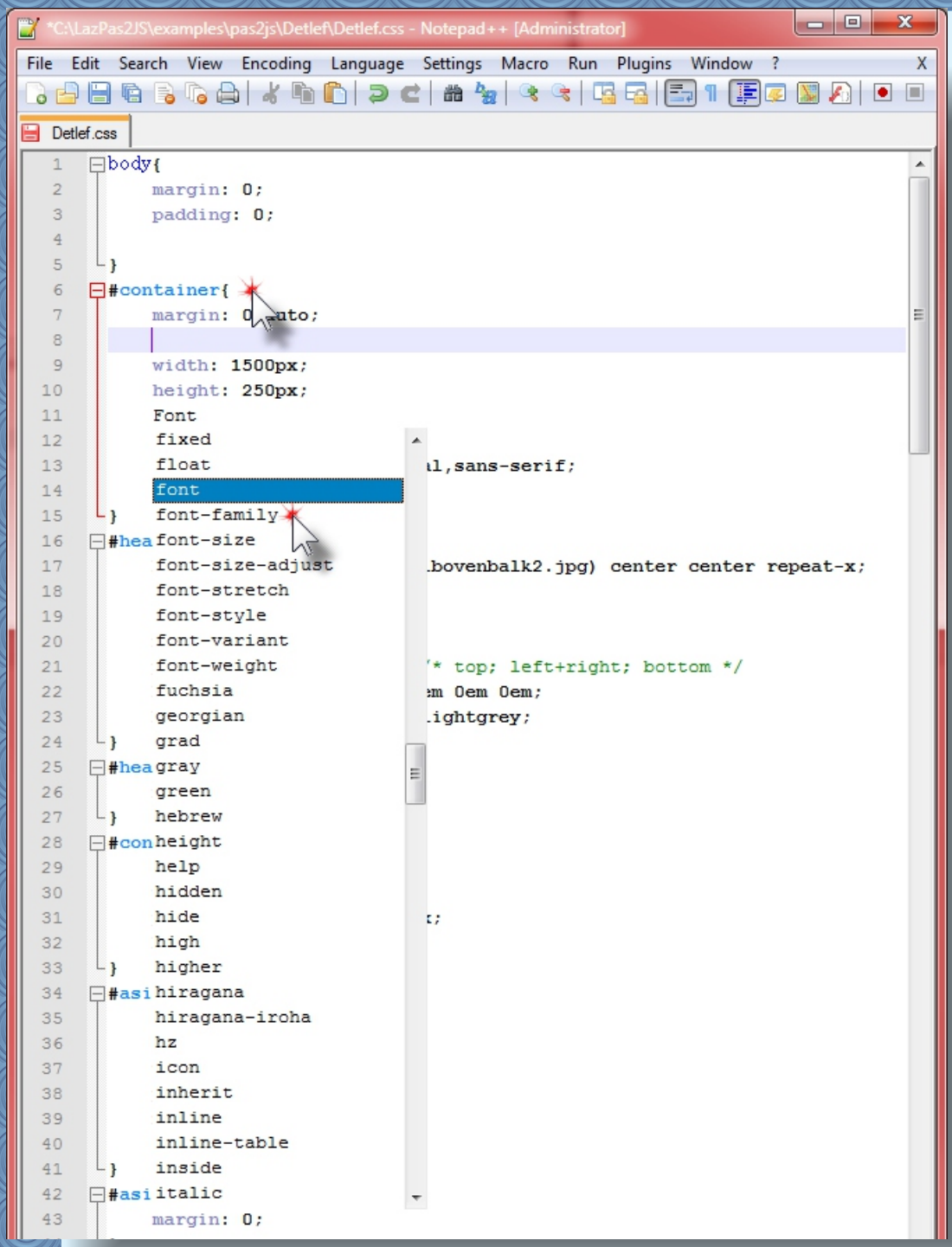


Figure 21: Getting additional options

If you use like in Lazarus or Delphi the Key- combination **CTRL+Space-bar** you will find additional information you might need for the Style Sheet.

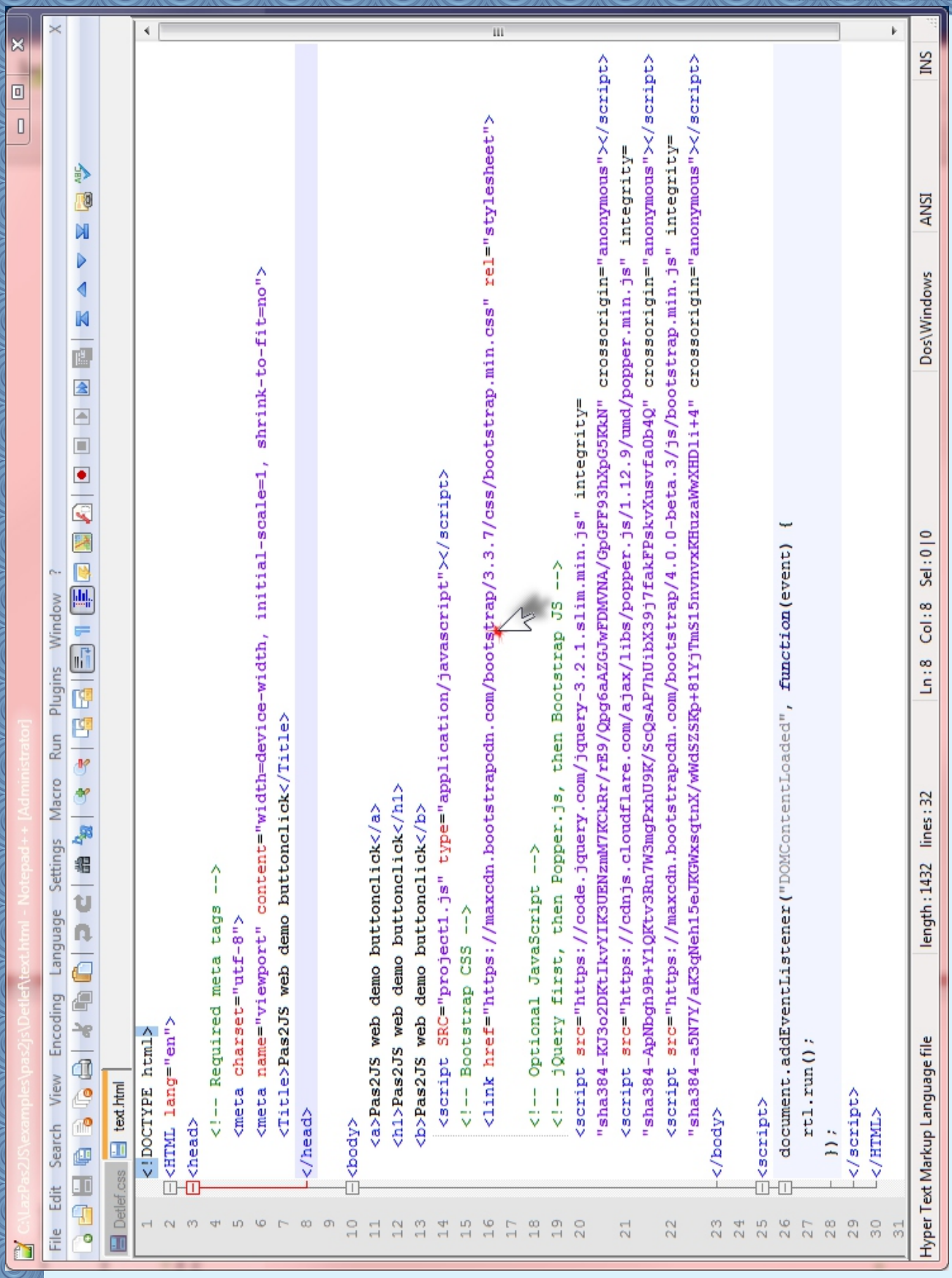


Figure 22: The Html file of the project shown in Notepad++ It is of course funny that we show in Notepad++ the line with the Bootstrap CSS style. I used the style Bootstrap showed to be used as a template and added only necessary things to play around a little bit:
I made some headings to find out what would be changed.

```
<body>
   <a>Pas2JS web demo buttonclick</a>
   <h1>Pas2JS web demo buttonclick</h1>
   <b>Pas2JS web demo buttonclick</b>
```

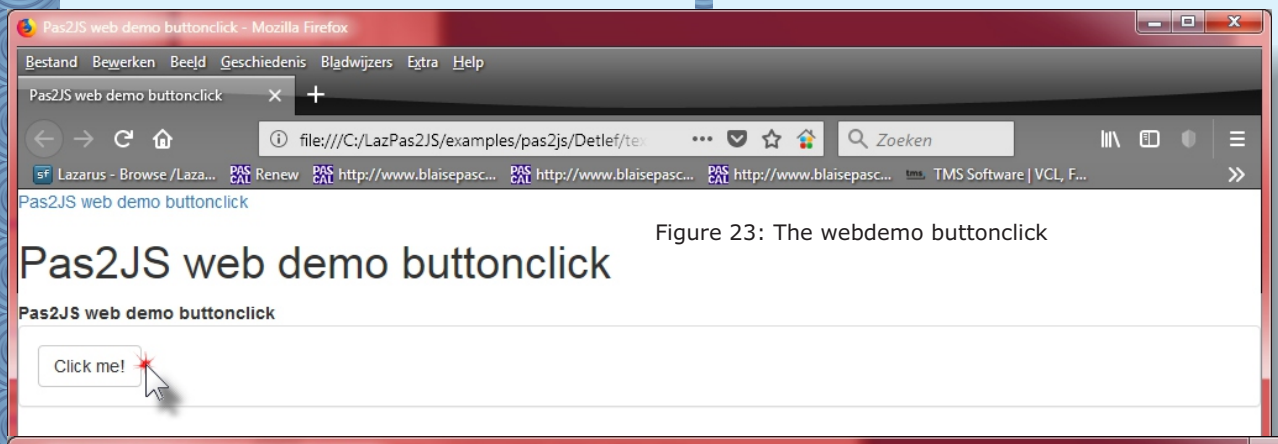


Figure 23: The webdemo buttonclick

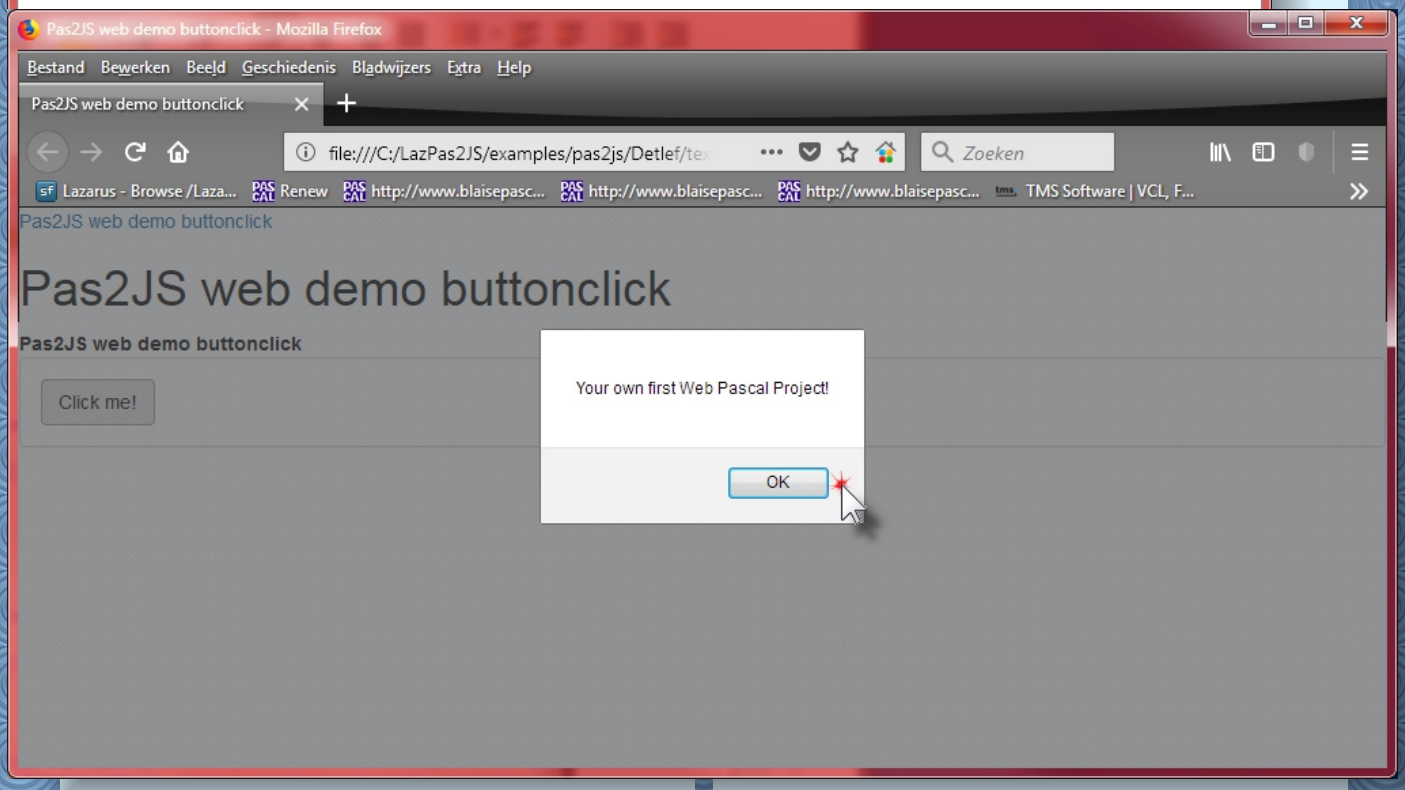


Figure 24: The result

I advise you to do your own alterations.

BACK TO THE PROJECT
here is the listing of this project.

```
program project1;

uses web, classes;

Type
 TForm = Class
  function ButtonClick(Event: TJSMouseEvent): boolean;
  Constructor Create;
 end;

function TForm.ButtonClick(Event: TJSMouseEvent): boolean;
// to create a message
begin
 writeln('ButtonClick ',Event,' in ',className);
 window.alert('Your own first Web Pascal Project!');
 Result:=true;
end;

constructor TForm.Create;
Var
 Panel,PanelContent : TJSElement;
 Button1      : TJSElement;
begin
 Panel := document.createElement('div');
 // attrs are default array property...
 Panel['class']:='panel panel-default';
 PanelContent := document.createElement('div');
 PanelContent['class']:='panel-body';

 Button1:=document.createElement('input');
 Button1['id']   := 'Button1';
 Button1['type'] := 'submit';
 Button1['class'] := 'btn btn-default';
 Button1['value'] := 'Click me!';

 TJSHTMLElement(Button1).onclick := @ButtonClick;
 document.body.appendChild(Panel);
 Panel.appendChild(PanelContent);
 PanelContent.appendChild(Button1);
end;

begin
 TForm.Create;
end.
```

This time we will give you all the projects that came with this first Beta version:
You will find the results in code delivered with the project. A description and explanation follows here.
I need to say one thing more:
Michael van Canneyt and **Mattias Gärtner** are the geniuses that created this.

My role was the organizer and helping hand.

If you really want to do things with this Beta: you can ask Michael van Canneyt at
Michael@FreePascal.org
He will try to help and answer.

CREATING A SIMPLE SERVER

Some programs that run in the browser will need a webserver, to fetch extra files or fetch data.
This can be any webserver: **Apache, IIS, NGINX** or **any other webserver**.
It's NOT even necessary to install a full-blown webserver: FPC comes with 2 webserver utilities.
One of them is **simpleserver,** a small **HTTP** server that serves files from the directory in which it was started.
A version of this program can be found on your own downloadpage (via Login).

`https://www.blaisepascal.eu/loginnew.php`

The project is called `simpleserver.lpi`.
It can be opened in **Lazarus,** and compiled.
By default, it will start serving files from the directory in which it was started, and it will do so on port 3000.
(*both the starting directory and the port can be specified*)

So, if you want to load the `project1` from a webserver, do the following steps:

```
C:\LazPas2JS\examples\pas2js\simpleserver
```

`C:\LazPas2JS\examples\pas2js\simpleserver`

3. **Start** the simpleserver.exe.
   Simply double-clicking on it in the explorer should be enough.

```
CD C:\LazPas2JS\examples\pas2js\simpleserver
```

and type the following 2 commands:

`simpleserver`

4. When **simpleserver** is running, in the browser, enter the following URL:

`http://localhost:3000/test.html`

and the program should be displayed in your browser as it was when you double-clicked.

The default behaviour of **simpleserver** is simply to serve any file it finds on disk, based on the URL request it receives from the browser.

But you can add your own output to it.
For example, the following lines can be inserted at the start of the main program source:

```
Procedure DoJSONRequest(ARequest : TRequest; AResponse : TResponse);

begin
  AResponse.Content:='{ "data" : [ { "name" : "michael" }, '+
  ' { "name": "Detlef" } , { "name": "Mattias" }]  }';
  AResponse.ContentType:='application/json';
  AResponse.SendContent;
end;

begin
  httprouter.RegisterRoute('/jsondata',@DoJSONRequest);
  TSimpleFileModule.RegisterDefaultRoute;
  // Here the rest of the main program follows
```

After recompiling the program and running it, you can enter the following URL in the browser
`http://localhost:3000/jsondata`
If you do this, you should see something like Figure 25 (*see below*) using Firefox:

This is of course very simple **JSON data**, but it can be easily extended to include other, more dynamic, data.
The **JSON** data can then be requested and used in your **HTML** project (*the demoajax and demoxhr sample projects show how to do this*).

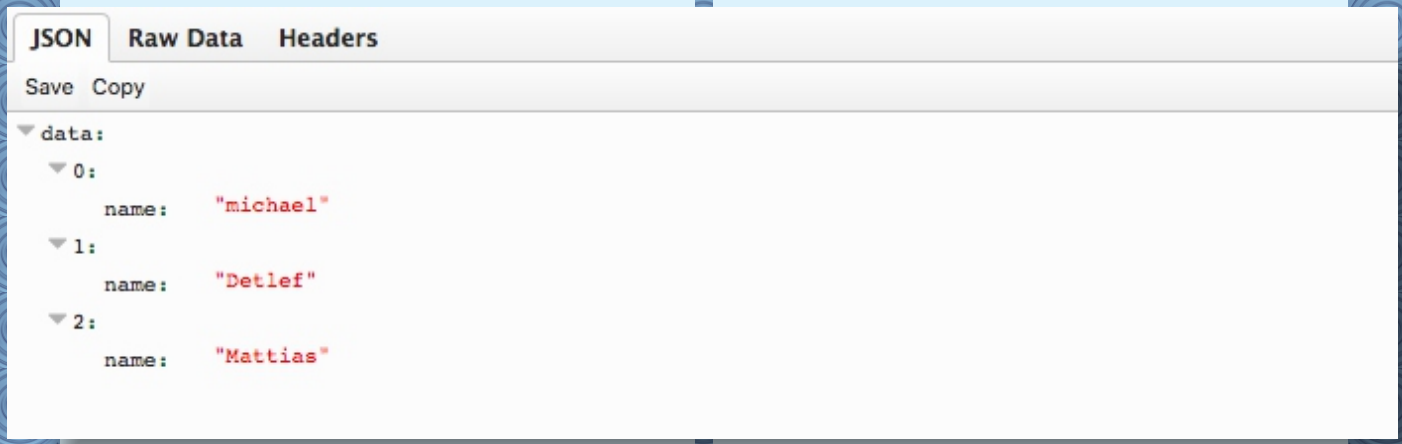
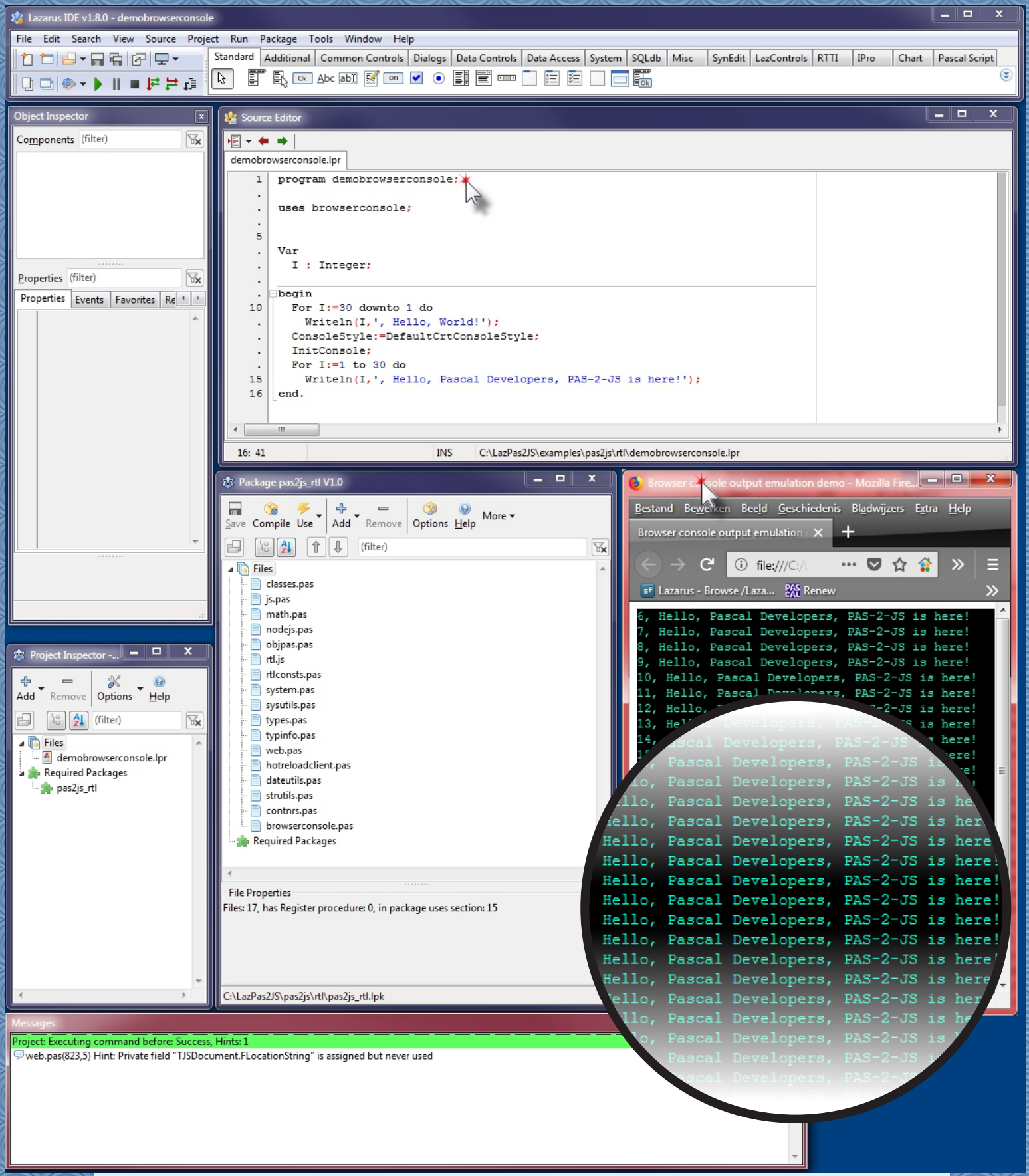
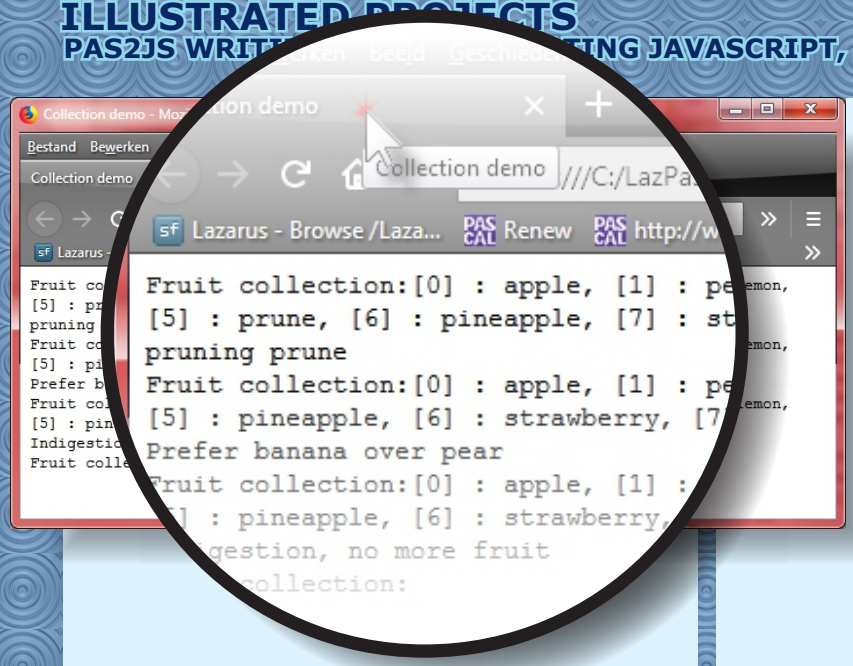


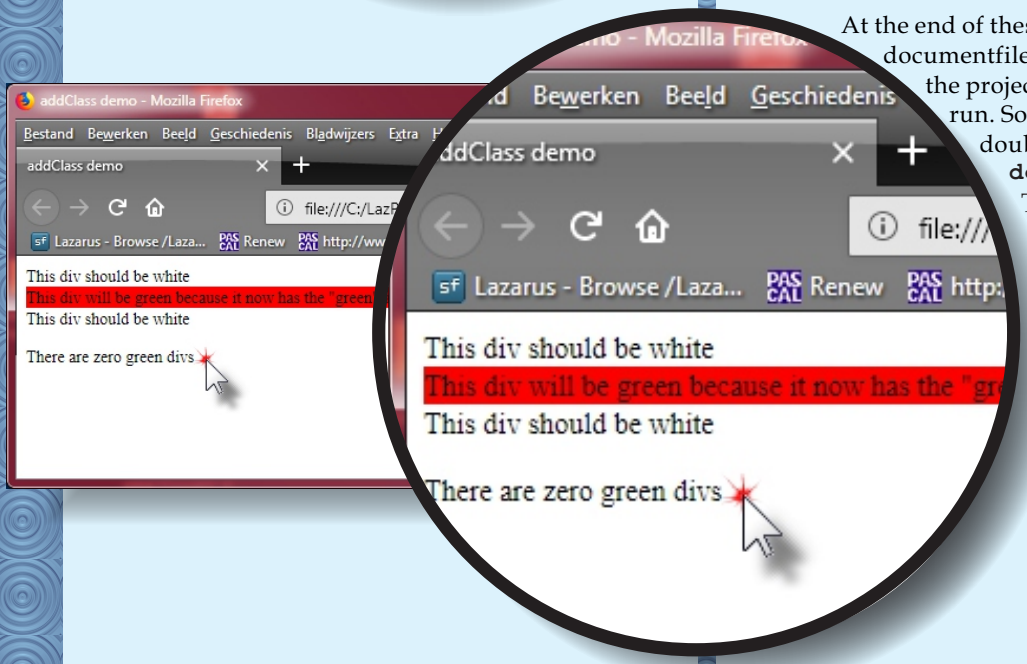
Figure 25 The result



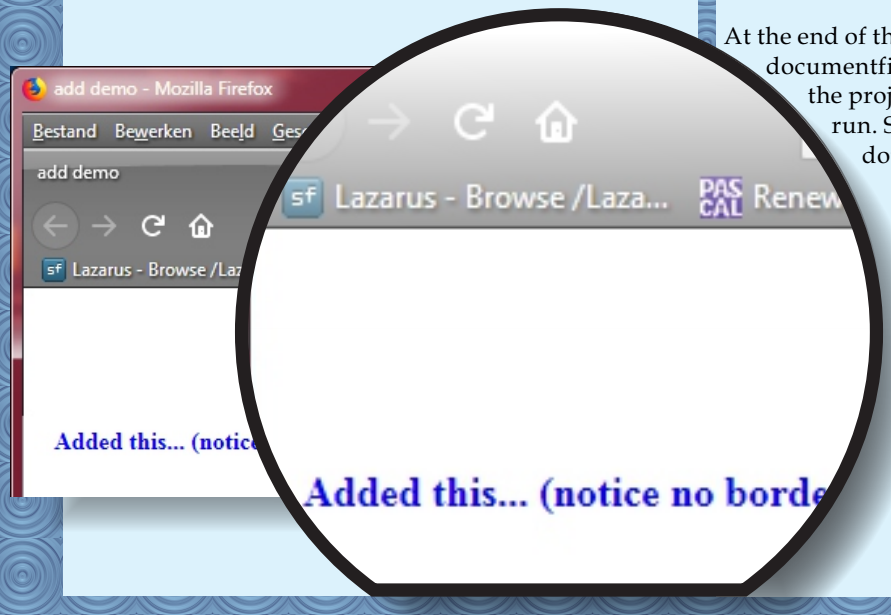
At the end of these lines you will find the documentfiles, there working and the project is already prepared and run. So if your interested doubleclick the **demobrowserconsole.html**. To see it in your own browser double-click on the ***.lpi** file and your project will open. **demobrowserconsole.html demobrowserconsole.js / demobrowserconsole.lpi / demobrowserconsole.lpr demobrowserconsole.lps**



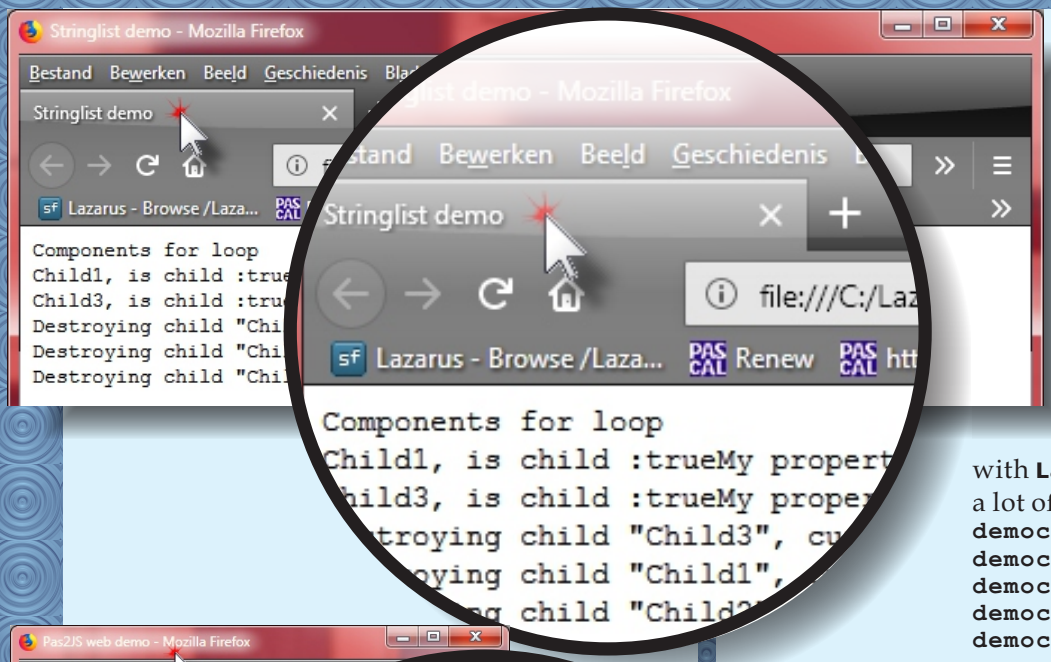
At the end of these lines you will find the documentfiles, there working and the project is already prepared and run. So if your interested doubleclick the **democollection.html**
To see it in your own browser double- click on the ***.lpi** file and your project will open.
democollection.html
democollection.js
democollection.lpi
democollection.lps
democollection.pas



At the end of these lines you will find the documentfiles, there working and the project is already prepared and run. So if your interested doubleclick the **demoaddclass2.html**
To see it in your own browser double- click on the ***.lpi** file and your project will open.
demoadd.html
demoadd.lpi
demoadd.pas
demoaddclass.html
demoaddclass.pas
demoaddclass2.html
demoaddclass2.pas

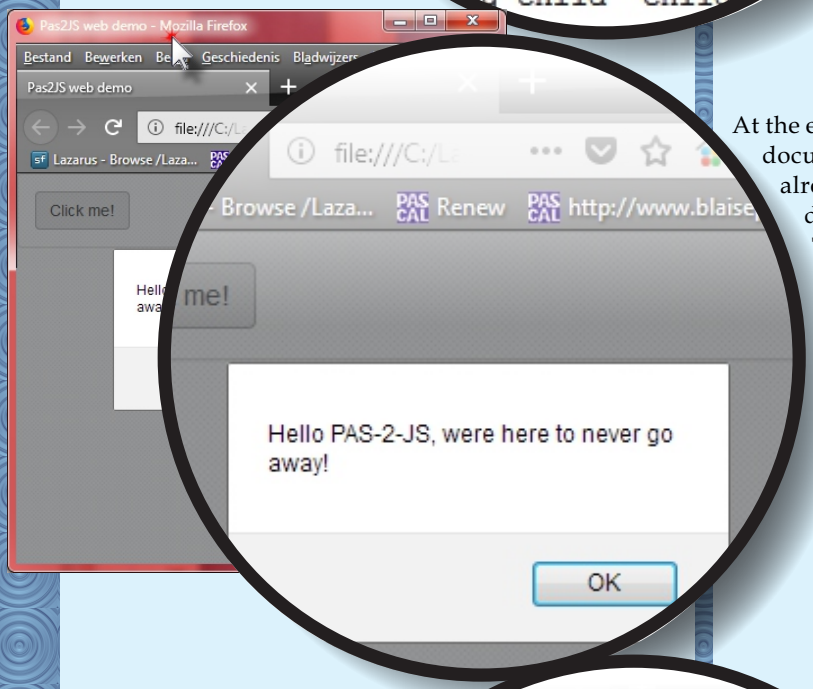


At the end of these lines you will find the documentfiles, there working and the project is already prepared and run. So if your interested doubleclick the **demoadd.html**
To see it in your own browser double- click on the ***.lpi** file and your project will open.
demoadd.html
demoadd.lpi
demoadd.pas



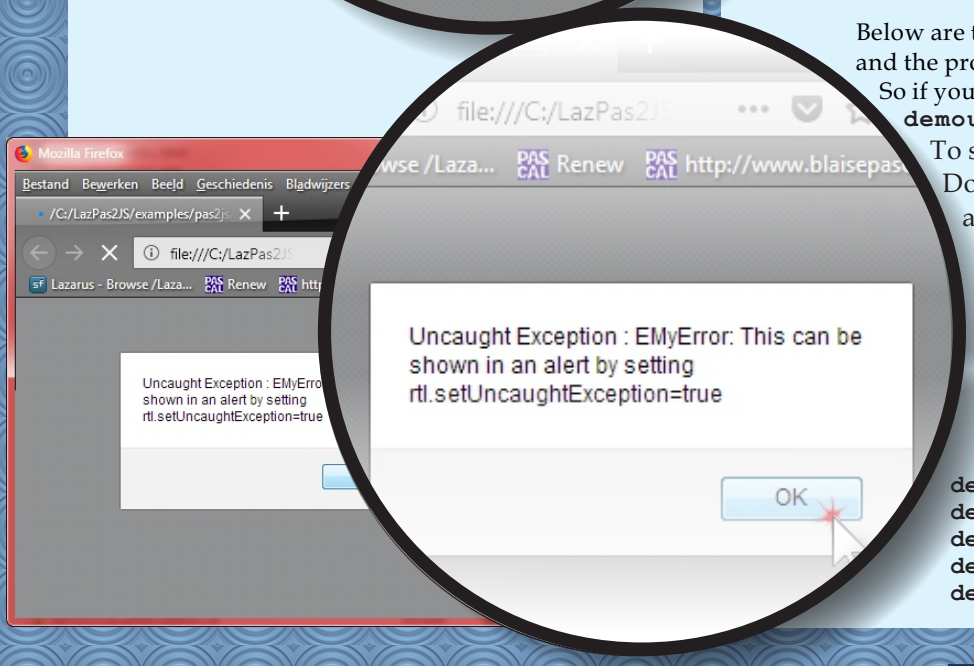
At the end of these lines you will find the documentfiles, there working and the project is already prepared and run so if your interested duobleclick the **democomponents.html**. To see it in your own browser double- click on the ***.lpi** file and your project will open with **Lazarus.** There are quite a lot of examples.

democomponents.html
democomponents.js
democomponents.lpi
democomponents.lpr
democomponents.lps



At the end of these lines you will find the documentfiles, there working and the project is already prepared and run. So if your interested duobleclick the **demodombuttonevent.html** To see it in your own browser double- click on the ***.lpi** file and your project will open.

demodombuttonevent.html
demodombuttonevent.js
demodombuttonevent.lpi
demodombuttonevent.lps
demodombuttonevent.pas



Below are the documentfiles, there working and the project is already prepared and run. So if your interested duobleclick the **demouncaughtexception.html** To see it in your own browser. Double click on the ***.lpi** file and your project will open with **Lazarus.** There are quite a lot of examples.

demouncaughtexception.html
demouncaughtexception.js
demouncaughtexception.lpi
demouncaughtexception.lps
demouncaughtexception.pas

BY MIGUEL BEBENSEE

starter expert

INTRODUCTION

Time tracking applications are usually simple but often-used applications in everyday business. This article presents a simple time-tracking web application which was developed using Lazarus and Pas2JS. This application can be used to record start and end times, which are saved in the browser's local storage. Moreover, the application runs on desktop and mobile devices and can be used without an internet connection on a smartphone. Test it yourself: **devstructor.com/demos/pas2js-time**

The Project

As already mentioned, the project was completely developed in Object Pascal using the Lazarus IDE, Pas2JS, HTML and Bootstrap. Pas2JS has the task of generating a JavaScript file from the Object Pascal source code. The form was designed in HTML, the standard markup language for web pages. HTML allows the placement of content such as buttons and labels on the web page. Finally, Bootstrap is a web framework with the task of making the form look good on all devices.

USING HTML AS A FORM

Although it sounds complicated, using a **HTML** document as a form is actually quite easy. Basically, we can create **HTML** objects like this button and access it in **Object Pascal.**

```
<button class="btn btn-primary"
     id="btnBook">Start</button>
```

To access this button, a reference in the form class and a simple assignment via the ID attribute is sufficient. Please note that the class attributes are predefined **Bootstrap** classes which will give the button a modern blue colour.

```
TTimeForm = class(TObject)
  [...]
  btnBook: TJSHTMLElement;
 private
[...]

constructor TTimeForm.Create;
begin
 [...]
 // Assign Controls
 btnBook:=TJSHTMLElement(document.getElementById('btnBook'));
 btnBook.onclick:=@BookClick;
 [...]
end;
```

After the reference has been set, the button can be used as normal and, for example, the `OnClick` event can be assigned or the caption changed. Of course, it is also possible to create controls at runtime and append them into the **HTML** document tree which is called **DOM.**

PROGRAMMING WEB APPLICATIONS IN OBJECT PASCAL

Whilst the previous step was an unusual one for us **Object Pascal** developers, we now return to the familiar **Object Pascal** programming. At this point, there is not much to report. It is possible to use well-known concepts such as `classes, encapsulation, inheritance` and type safety as in any other desktop application. Especially the last concept of type safety is an enrichment for web development, as it allows the development of scalable applications. **JavaScript** is not type safe. This can quickly lead to type mistakes, especially in growing projects or when working in a team. This project consists of three units: the data model of a booking `(unitModel)`, a controller for access, loading and saving bookings `(unitBookingData)`, and the main program including the `Form (TimeTracking)`.

THE DATA MODEL

The data model consists of one class, the `TBooking`.

```
TBooking = class(TObject)
 private
  FID: Integer;
  FStart: TDateTime;
  FStop: TDateTime;
  function GetIsOpened: Boolean;
  procedure SetID(AValue: Integer);
  procedure SetStart(AValue: TDateTime);
  procedure SetStop(AValue: TDateTime);
  function GenID: Integer;
public
  constructor Create;
  constructor Create(AID: Integer);

  procedure Load;
  procedure Save;
  procedure Delete;

  procedure Book;
  function GetWorkingTimeString: String;
published
  property ID: Integer read FID write SetID;
  property Start: TDateTime read Fstart
                              write SetStart;
  property Stop: TDateTime read Fstop
                              write SetStop;
  property IsOpened: Boolean read GetIsOpened;
end;
```

The booking has some elementary properties. This includes the data ID which is used for loading and saving, the start and stop time, as well as a property if the booking is still open. Moreover, it can be loaded, saved, deleted and booked using the procedures. Thereby, the data is saved into the **HTML5** local storage which is a key-value-database provided by the browser. This database is shown in Figure1 .
Each booking has a start and stop value.
Moreover, the local storage stores a generator which merely holds the last booking ID.

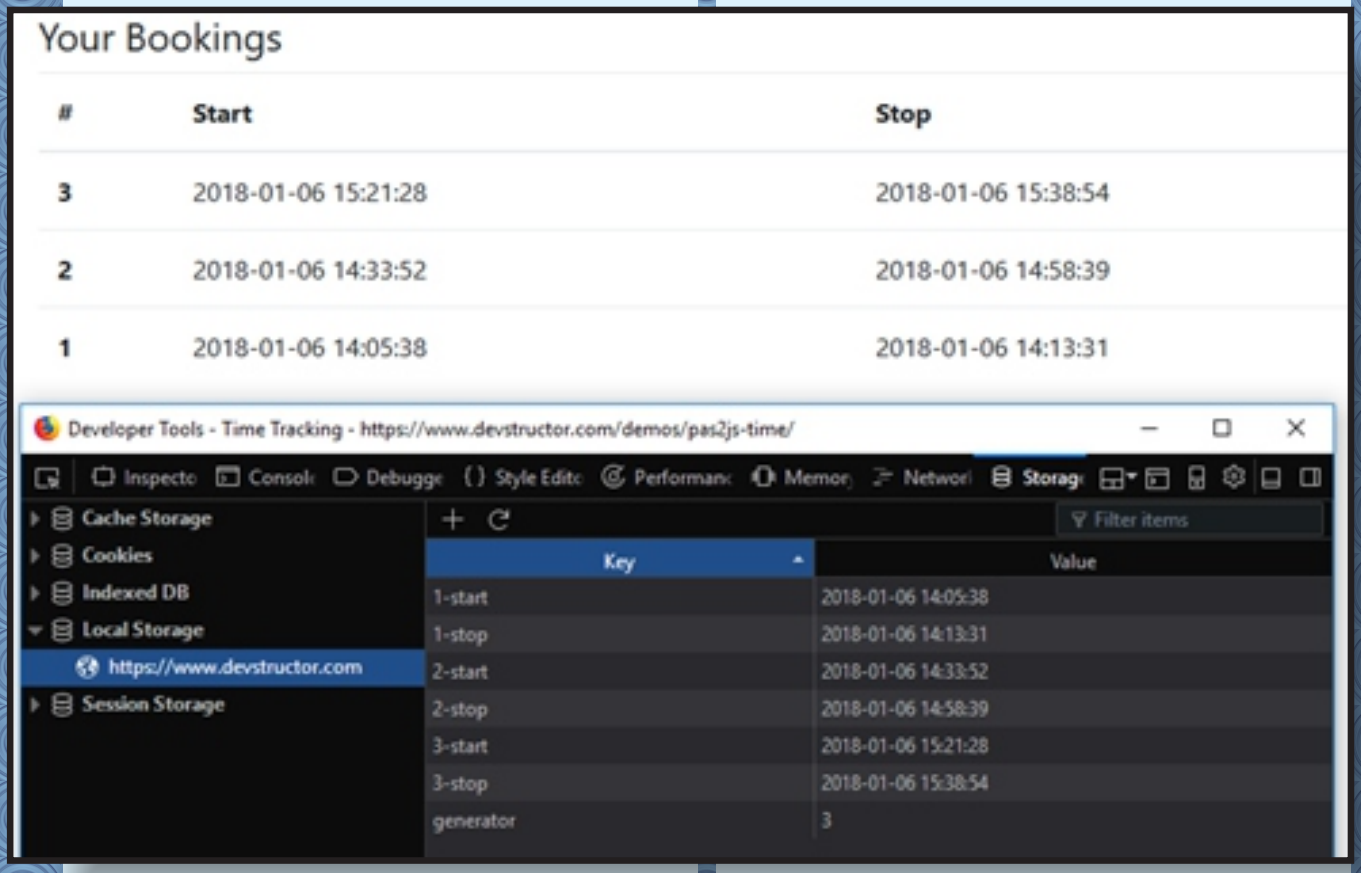


Figure 1: Data Structure of Bookings in the Local Storage

Accessing the local storage is a simple task. Let us take a look at the **function GenID**.

```
function TBooking.GenID: Integer;
var LastID: Integer = 0;
begin
 if window.localStorage.getItem('generator') <> Null then
  LastID:=StrToInt(window.localStorage.getItem('generator'));

 Result:=LastID + 1;
 window.localStorage.setItem('generator', IntToStr(Result));
end;
```

This function returns the next generator value for the ID property. It tries to read the last value from the local storage which is saved using the key generator.
If this value is not found, it will keep with the value zero from the variable declaration. After this, it sets the return value to the next ID and saves the last value to the local storage. That is all there is to it!

ONE BOOKING IS NOT ENOUGH.

Of course, the user should be able to create more than one booking. Therefore, the booking list must also be available in the application. This task is done by the **TBookingData** class.

```
TBookingData = class(TObject)
 private
  FBookings: TList;

  function GetBooking(Index: Integer): TBooking;
  function GetBookingCount: Integer;
  function GetIsOpened: Boolean;

  procedure LoadData;
 public
  constructor Create;

  procedure Book;
  procedure DeleteAll;

  property BookingCount: Integer read GetBookingCount;
  property Booking[Index: Integer]: TBooking read GetBooking;
  property IsOpened: Boolean read GetIsOpened;
 end;
```

This class is a simple connection between the data model and the form. The tasks of this class include access to the booking list. Again, it shows that **Pas2JS** handles complex structures such as the indexed property Booking pretty well. As a result, **Object Pascal** developers hardly have to adapt to the new platform. Furthermore, **TBookingData** allows you to create a booking and to delete all data.
Finally, **LoadData** loads all bookings and is executed in the constructor.

The Form

The last piece of the puzzle is the form. It is not much different from a normal desktop application. It stores references of the graphical controls, has some **OnClick Events** and features some procedures to display information and to update the booking table. The differences can be found in the detail. An important task of the form class is to access the **HTML** form. The next code snippet shows the procedure **RenderBookings** which generates the Booking table.

```
procedure TTimeForm.RenderBookings;
var
 i: Integer;  tblBookings: TJSElement;
 tblRow, colID, colStart, colStop: TJSElement;
begin
 UpdateBookButton;
 UpdateActiveBooking;

 tblBookings:=document.getElementById('tblBookings');
 tblBookings.innerHTML:='';

 for i := Pred(FBookingData.BookingCount) downto 0 do
 begin
  if FBookingData.Booking[i].IsOpened then Continue;

  tblRow:=document.createElement('tr');

  colID:=document.createElement('th');
  colID.innerText:=IntToStr(FBookingData.Booking[i].ID);

  colStart:=document.createElement('td');
  colStart.innerText:=DateTimeToStr(FBookingData.Booking[i].Start);

  colStop:=document.createElement('td');
  colStop.innerText:=DateTimeToStr(FBookingData.Booking[i].Stop);

  tblRow.append(colID);
  tblRow.append(colStart);
  tblRow.append(colStop);
  tblBookings.append(tblRow);
 end;
end;
```

To achieve this goal, the table is first located and assigned via the ID of the **HTML** element. Following this, the old content must be deleted. This can be done by simply setting the inner **HTML** to an empty text. Finally, the table contents are created in a for loop. The `document.createElement` method can be used to create new **HTML** elements. After the required properties have been set, the element can be appended via the `append method`.
Until now, the whole project could be programmed in **Object Pascal** like any other desktop application. Differences can be found in the detail, such as accessing or creating **HTML** elements. When working with frameworks such as **Bootstrap,** sometimes pure **Javascript** can help. In this application, a **Bootstrap confirmation dialog** should appear before the deletion of all data. Following the successful deletion, the web app should automatically close this dialog.

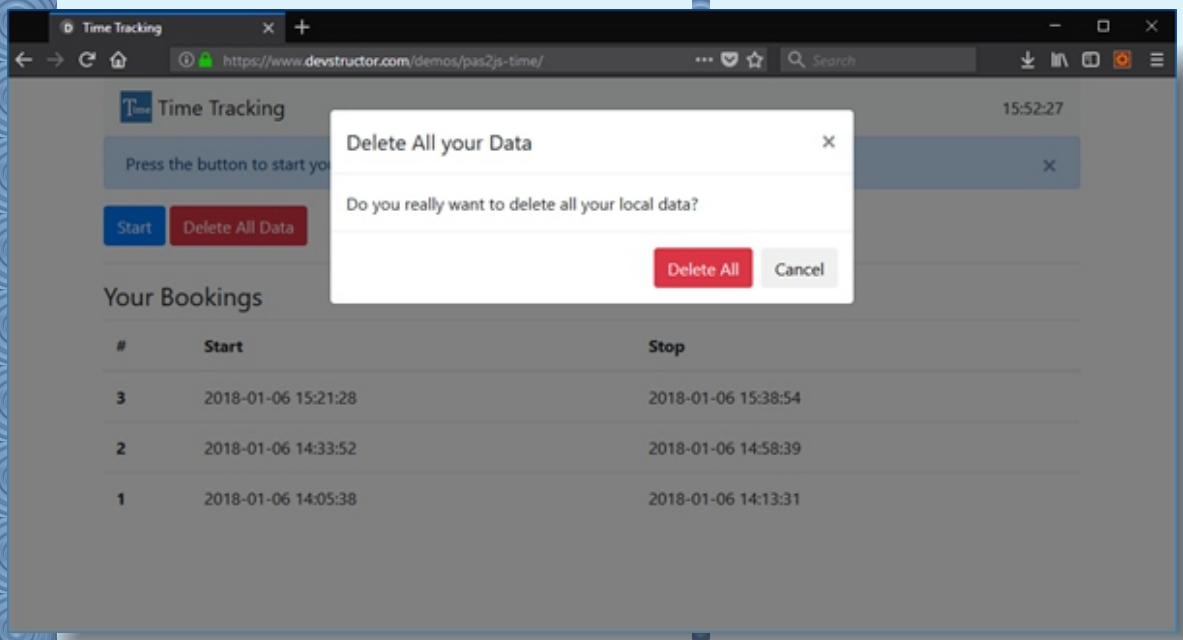


Figure 2: Bootstrap Confirmation Dialog

When I searched the Bootstrap documentation, I found the following simple **JavaScript** line to close the dialog.
`$('#dlgDelete').modal('hide')`
But how to implement this **jQuery** line in **Object Pascal**?
Well you could add the `Pascal wrapper unit` for the **jQuery framework** or just take the simple and lazy way as I did.

```
procedure TTimeForm.HideDlgDelete; assembler;
asm
  $('#dlgDelete').modal('hide');
end;
```

Inline **Assembler** in **JavaScript** is just **JavaScript.** Especially when working with **JavaScript** frameworks, this can be very helpful if you do not have the time and inclination to create wrapper classes for minor things. But creating wrapper classes is really not much work thanks to external classes.

AN OFFLINE WEB APPLICATION?

That sounds weird, but it is very simple in this project. The main task of the web server is to transfer the documents. This includes all required **HTML, JavaScript** and **Bootstrap** files.
After this, there is no need to contact the server again. All data will be saved in the **local storage**.
As a result, it is possible to use the **HTML5** cache manifest. The **cache manifest** is a simple text file which lists all necessary files for the offline usage. The manifest file for this project looks like this.

CACHE:

```
index.html
css/bootstrap.min.css
js/jquery-3.2.1.slim.min.js
js/popper.min.js
js/bootstrap.min.js
TimeTracking.js
img/webtime.png
img/webtime-small.png
```

After creating this file, we simply need to add the file name of the **cache manifest** to the **HTML** file. This is done by adding the following attribute to the **html** tag.

```
<html manifest="webtime.appcache">
```

Please note that the filename of my manifest file is "`webtime.appcache`". The last question is: "How to use the manifest file?" You just need to add the website to your smartphone's home screen. After browsing the site for the first time, the browser will cache the specified files automatically.

ABOUT THE AUTHOR

Hello!
My name is Miguel and I'm a software engineer at IBExpert in Germany. Back in 2008, I started software development with Delphi 7 and decided to work commercially three years later. At IBExpert, the development of database and client/server systems is one of my main tasks. However, I really enjoy the diversity in my everyday life.

IBExpert: ibexpert.net
My website: devstructor.com

Quantum Computation possible with Majorana Fermions

The Majorana fermion is a hypothetical fermionic particle which is its own anti-particle. (Source: Thinkstock)

KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.04.40 RELEASED!

NEW! AUTOMATIC DATABASE STRUCTURE UPDATE
NEW! GENERIC OBJECT ORIENTED CONFIGURATION FRAMEWORK

- **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

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.