BLAISE PASCAL MAGAZINE 114/115

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Blaise Pascal

Free Pascal
Lazarus
version 3.0
PAS2JS V3
Write Once
Compile Anywhere

RAD Studio 12

BLAISE PASCAL MAGAZINE 114/115

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

Blaise Pascal

CONTENT ARTICLES

ADVERTISING

Niklaus Wirth

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

Publisher: PRO PASCAL FOUNDATION in collaboration © Stichting Ondersteuning Programmeertaal Pascal

CONTRIBUTORS

Stephen Ball http://delphiaball.co.uk DelphiABall	Dmitry Boyarintsev dmitry.living @ gmail.com	Michaël Van Canneyt ,michael @ freepascal.org	Marco Cantù www.marcocantu.com marco.cantu @ gmail.com
David Dirkse www.davdata.nl mail: David @ davdata.nl	Benno Evers b.evers @ everscustomtechnology.nl	Bruno Fierens www.tmssoftware.com bruno.fierens @ tmssoftware.com	Holger Flick holger @ flixments.com
Mattias Gärtnernc- gaertnma@netcologne.de	Max Kleiner www.softwareschule.ch max @ kleiner.com	John Kuiper john_kuiper @ kpnmail.nl	Wagner R. Landgraf wagner @ tmssoftware.com
Vsevolod Leonov vsevolod.leonov@mail.ru	Andrea Magni www.andreamagni.eu andrea. magni @ gmail.com www.andreamagni.eu/wp		Helmut Elsner Korrektor der Deutschen Ausgabe helmut.elsner@live.com
		Paul Nauta PLM Solution Architect CyberNautics paul.nauta @ cybernautics.nl	
Kim Madsen www.component4developers.com kbmMW		Boian Mitov mitov @ mitov.com	
	Jeremy North jeremy.north @ gmail.com	Detlef Overbeek - Editor in Chief www.blaisepascal.eu editor @ blaisepascal.eu	
Anton Vogelaar ajv @ vogelaar-electronics.com	Danny Wind dwind @ delphicompany.nl	Jos Wegman Corrector / Analyst	Siegfried Zuhr siegfried @ zuhr.nl

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

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 (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Ondersteuning Programmeertaal Pascal)
Subscription department Edelstenenbaan 21 / 3402 XA IJsselstein, Netherlands Mobile: + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Trademarks All trademarks used are acknowledged as the property of their respective owners.
Caveat Whilst we endeavor 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.

Member of the Royal Dutch Library KB KONINKLIJKE BIBLIOTHEEK Member and donor of WIKIPEDIA

SUBSCRIPTIONS (2023 prices)	Internat. excl. VAT	Internat. incl. 9% VAT	Shipment	TOTAL
Printed Issue (8 per year) **±60 pages :**	**€ 200**	**€ 218**	**€ 130**	**€ 348**
Electronic Download Issue (8 per year) **±60 pages :**	**€ 64,22**	**€ 70**		

COPYRIGHT NOTICE

From your editor

Greetings, esteemed readers,
Wishing you all a joyful new year.
Despite the formidable challenge, achieving global peace remains a paramount objective.

End all hostilities. Furthermore, these antiquated individuals who consistently strive for superiority and grandiosity inevitably create problems. However, hope is always present.
Optimism for improvement and the future.

Spring is approaching.

I had intended to address this issue prior to the Christmas holidays, but the extensive workload required to complete this task - a total of 154 pages - necessitated its extension over multiple days.
Only now have I managed to bring it to completion.
There have been numerous occurrences that required me to work tirelessly, almost around the clock, to complete.

The rationale behind this decision is that we had an additional objective to accomplish:
our magazine is going to be published in another language, specifically Brazilian Portuguese.
Upon completing this task, I will proceed to develop two other versions:
the German Edition and the Brazilian Edition.
This achievement has been facilitated by Geuze (GDK Software),
who have provided assistance in translating it to Portuguese due to their establishment in Brazil, where they employ native speakers of the language.

Greetings to our readers in Brazil.
This news is of utmost significance.
This item highlights the inclusion of the current version of Delphi (12) Athens and, of course, the most recent version of Lazarus 3.0.
The release of PAS2JS version 3 is scheduled for this week.
The Pascal family is expanding, and I intend to undertake further endeavours.
The University of Cologne will host a significant Pascal Conference in Cologne on October 2024.
The conference will take place over two days, specifically on Thursday 10 and Friday 11.
The event will consist of both presentations and workshops.

In March, I will be hosting another Pascal Cafe event in Holland.
I request your presence from all nations, as was the case in the previous year.
The experience was great.
In the subsequent item, specifically item number 116, I will provide further elucidation regarding this matter and outline our intended course of action.
Currently, I extend my desire for tranquilly and a pleasant year 2024 to all of you.

Your Editor:

Detlef

From our technical Advisor, Jerry King

"That's the only letter you received in the mail. Kids no longer write letters. They email or text."

TEST INSIGHT IN LAZARUS
BY MICHAEL VAN CANNEYT

ABSTRACT

The **FPCUnit** support in **Lazarus** has received an upgrade:
The **FPCUnit unit testing framework** now can communicate directly with the **Lazarus IDE**, making it even more easy to fix errors in your code.

1 INTRODUCTION

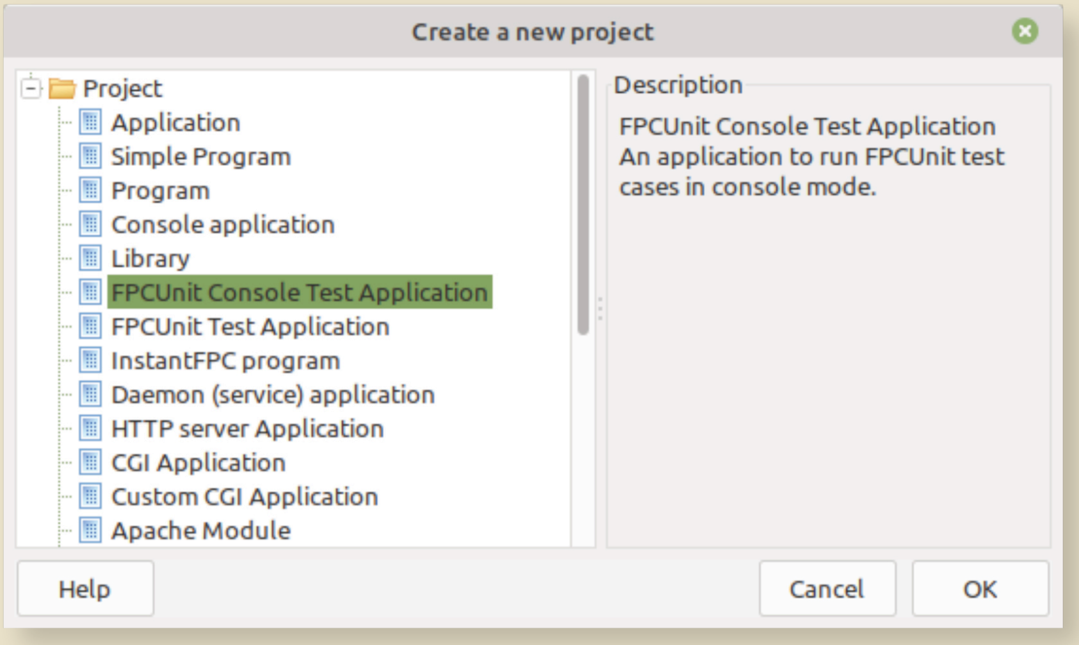
Free Pascal has had a unit testing framework since almost 20 years: `FPCUnit` is in use for the testing of many of the packages included in **Free Pascal**. It is roughly compatible to `DUnit`, the **Delphi unit testframework**: **Vanilla** test code written for `Dunit` will compile with **Free Pascal**.

The **Lazarus IDE** has some wizards to make a **FPCUnit test case** and to start a **FPC unit test program**.
2 kinds of test program can be created: a console test program, and a **GUI** test program.
The former shows the results of all tests on the console.

The latter uses a **GUI** window with a treeview to show all tests and the result of the tests.
Both these programs have the same drawback: they do not interact with the **IDE**.

It would be much nicer if you could just click on the test name and be taken to the implementation of that test in the **IDE.** Or, if there is a stack trace, on the error location and be taken to the location where the error is raised.

Well, now you can: In the trunk version of **Lazarus**, the **LazFPCUnit** package has been extended with '**Test Insight**'.
The idea for this feature has been taken from the similarly named Delphi plugin by **Stefan Glienke**.



Create a new project

Project
- Application
- Simple Program
- Program
- Console application
- Library
- FPCUnit Console Test Application
- FPCUnit Test Application
- InstantFPC program
- Daemon (service) application
- HTTP server Application
- CGI Application
- Custom CGI Application
- Apache Module

Description

FPCUnit Console Test Application
An application to run FPCUnit test cases in console mode.

Figure 1: Creating a FPCUnit console program

2 ARCHITECTURE

The **FPCUnit unit test** system works like all other test systems.
There is a test registry, and there are test listeners.
When the tests from the registry are run, the progress and results are reported by the registered listeners.
The console listeners just write to the console in one of several formats (*you can register your own*), while the **UI listener** adds nodes to a treeview.
The testinsight listener sends the results with HTTP requests to a server.
The location of this server can be specified in the sources, or in a small configuration file.
The **TestInsight** support in the **Lazarus IDE** launches a small HTTP server which listens for the HTTP requests with the test results.
The http server is started when you open the '**Test insight**' window from the **'View - Test insight**' menu in the **Lazarus IDE**.
When the test program exists, it starts it with a special option, to get a list of tests.

When you ask for a test run, the tests are run, and the window displays the test results.
Double clicking on a test will use the **IDE** code tools to jump to the correct method in the test project.
If the test program cannot reach the testinsight **HTTP** server, then it will fall back to running as a regular **FPCUnit test** console program.
You could do the same with a **UI program** if so desired, but this option has not been activated for the **FPCUnit UI** test program

3 USAGE

For this to work, **you need to have the latest IDE sources**, and you must of course install the **`lazfpcunit`** package. it is part of the standard list of packages in the IDE.

The '**New project**' → '**FPCUnit Console Test Application**' menu item (*figure 1 on page 5*) has now a project wizard which offers some options. The following options are available:

Run all tests by default
The standard **FPC unit console program** shows a help when running it without command-line options. If this option is checked, the program will run the tests instead when no command-line options are specified.

Default output format
In this checkbox, you can set the default output format for the console output. You can choose between **XML**, plain text (*with or without time info*) or **LaTeX**.

Use Testinsight to communicate results to the IDE.
When this option is set, the test results are sent to the IDE with testinsight.

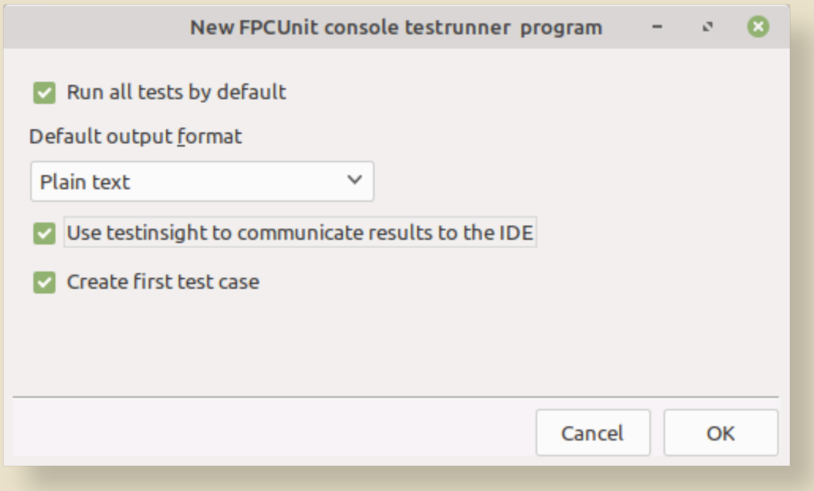
Create first test case
When checked, the IDE immediately launches the '**Create FPCUnit testcase**' wizard when the project is created.
The options dialog is shown in figure 2 on page 7.
Once your project is created with the '**Use testinsight**' option, you should save and compile it at once.
This will allow the IDE testinsight support to get a list of tests.

Once the project is compiled, you can open the **Testinsight** window. The window looks the same as the window of the **graphical FPCUnit test program** - not surprisingly, since it was copied from that window. But some elements were added, and it behaves differently from the original window.

When the window is opened, it will attempt to run the current project with the appropriate options to get a list of test, and the result looks like *figure 3 on article page 3* if all went well.

The '**Refresh**' button (*Left*) can be used to refresh the list of tests if you want. The list of tests will be refreshed in each case when the test program is actually run.
When the active project is changed, then the list of tests will also be automatically refreshed: the path of the **current FPC unit test executable** is always shown in the **status bar** at the bottom of the window.

3 USAGE (CONTINUATION)



New FPCUnit console testrunner program

☑ Run all tests by default

Default output format: Plain text

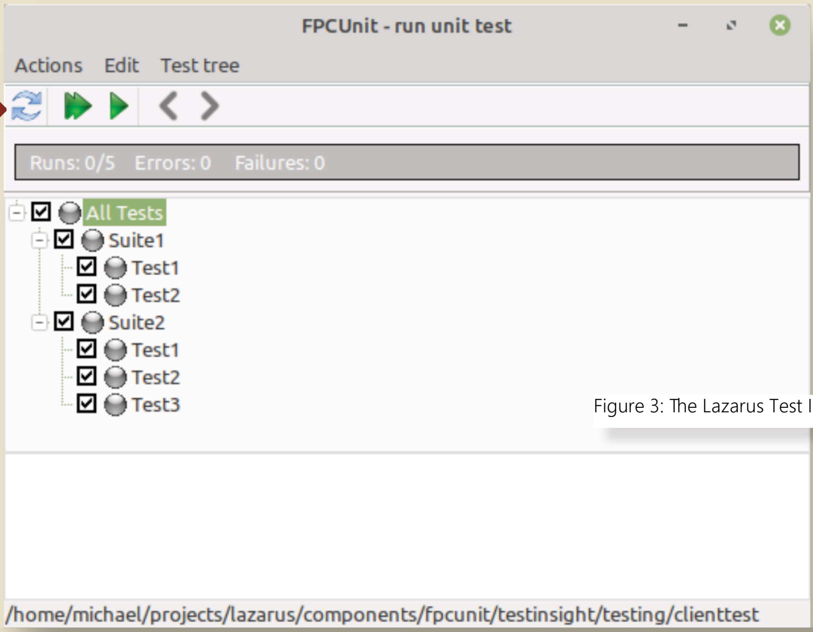
☑ Use testinsight to communicate results to the IDE

☑ Create first test case

Figure 2: The FPCUnit console program options

When you click the '**Run all tests**' button (*the green double arrow*), the **test program** will be executed. As the test results come in, the colored status indicator in front of each test will change color according to the result of the test.
If there is an error, more info will be displayed in nodes below the test node, as shown in *figure 4 on page 8*. Double clicking on the test or an error node, will **attempt to locate the code of the test** and **open the source file**. For most cases, this will work without problem.
However, the mechanism can fail in some cases:

❶ The currently active project in the IDE is not a **unit test** project.
❷ Item the sources of the test are not part of the project.
❸ If you are using some more advanced features of the test framework (*creating tests dynamically*). **TestInsight** supposes that the test names are method names of a `class`.



FPCUnit - run unit test

Runs: 0/5 Errors: 0 Failures: 0

- ☑ All Tests
  - ☑ Suite1
    - ☑ Test1
    - ☑ Test2
  - ☑ Suite2
    - ☑ Test1
    - ☑ Test2
    - ☑ Test3

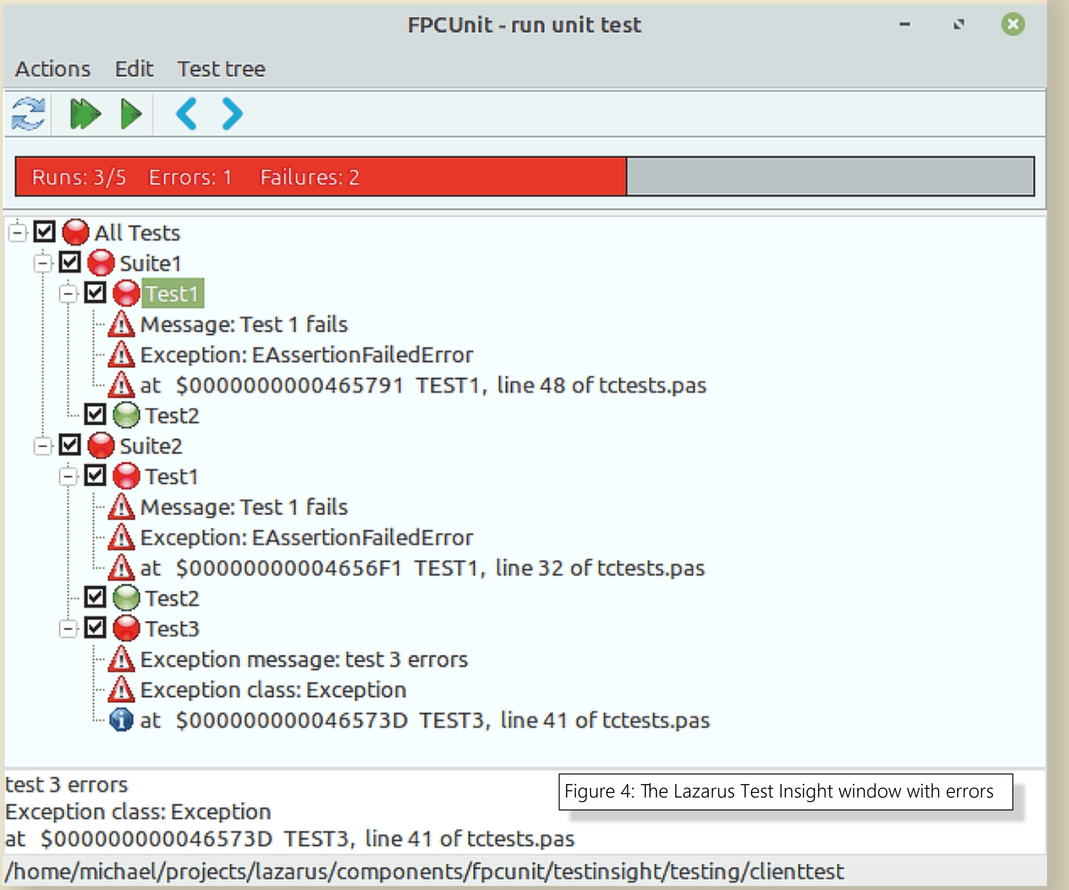
/home/michael/projects/lazarus/components/fpcunit/testinsight/testing/clienttest

Figure 3: The Lazarus Test Insight window

4 CONVERTING AN EXISTING FPCUNIT TEST PROGRAM

to use **TestInsight** the **'New program'** wizard cannot convert an existing program to use **Testinsight**.
If you have an existing **FPCUnit test program** that you wish to convert so it uses testinsight, you need to make some changes to the main project file.
The typical **FPCUnit test console program** has the following main project file source:

```
program clienttest;
{$mode objfpc}{$H+}

uses
  Classes, jsonparser, consoletestrunner, tcTests;

type
  TMyTestRunner = class(TTestRunner)
  end;

var
Application: TMyTestRunner;

begin
  Application := TMyTestRunner.Create(nil);
  Application.Initialize;
  Application.Title := 'FPCUnit Console test runner';
  Application.Run;
  Application.Free;
end.
```



```
Runs: 3/5   Errors: 1   Failures: 2

☑ All Tests
  ☑ Suite1
    ☑ Test1
       Message: Test 1 fails
       Exception: EAssertionFailedError
       at  $0000000000465791  TEST1,  line 48 of tctests.pas
    ☑ Test2
  ☑ Suite2
    ☑ Test1
       Message: Test 1 fails
       Exception: EAssertionFailedError
       at  $00000000004656F1  TEST1,  line 32 of tctests.pas
    ☑ Test2
    ☑ Test3
       Exception message: test 3 errors
       Exception class: Exception
       at  $000000000046573D  TEST3,  line 41 of tctests.pas

test 3 errors
Exception class: Exception
at  $000000000046573D  TEST3,  line 41 of tctests.pas
/home/michael/projects/lazarus/components/fpcunit/testinsight/testing/clienttest
```

Figure 4: The Lazarus Test Insight window with errors

Two things must be done with this project code:

❶ The `fpcunittestinsight` unit must be added to the `uses` clause.
❷ The function `IsTestinsightListening` must be called.
If it returns `True`, then the **`RunRegisteredTests`** routine must be called.

Both are implemented in the `fpcunittestinsight` unit.
If `IsTestinsightListening` returns `False`, then the original code must be executed.

The 2 functions to use are declared as follows:

```
procedure RunRegisteredTests(aConfig : String = '';
baseUrl: string = DefaultUrl);
function IsTestinsightListening(aConfig : String = '';
baseUrl: string = DefaultUrl) : Boolean;
```

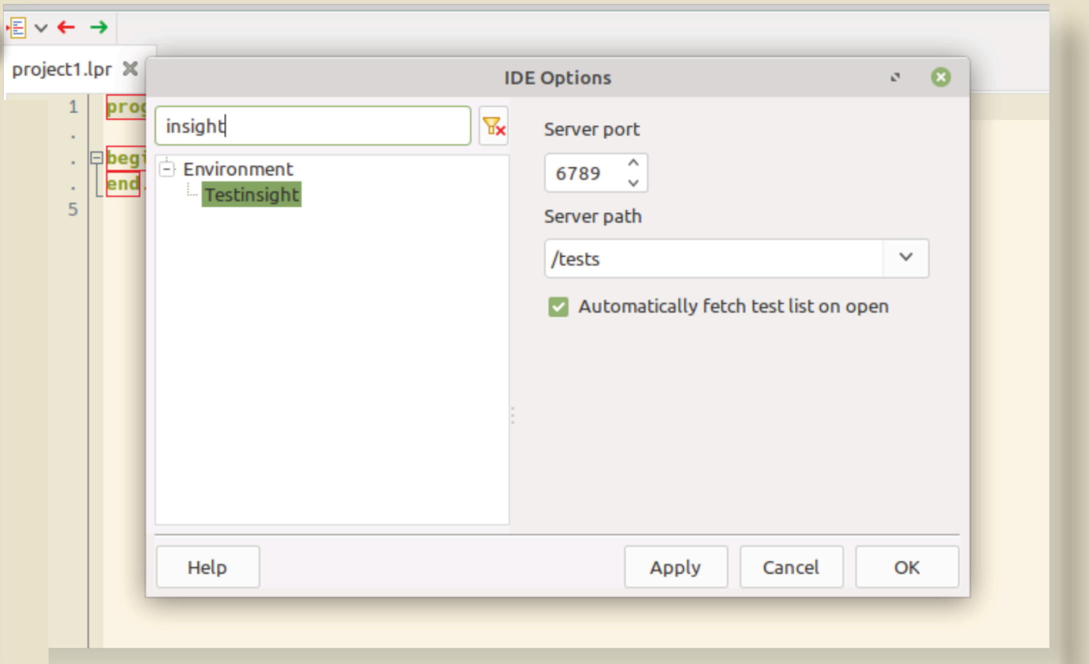
The config argument is the name of an .INI file with the settings for the test run. By default this is `TestInsightSettings.ini`, this is what the IDE uses.
The selected tests will be written in this file, and the port on which the IDE is listening (*The baseURL*)
The **`BaseURL`** argument is the **URL** where the testinsight server is listening.

By default this is **`http://localhost:8081/tests`**,

but this will be set by the IDE in the config file.
You could change these defaults for example to run the test program remotely,
but still receive the results locally on your development PC.
With these changes, the new project code will look like this:

```
program clienttest;

{$mode objfpc}{$H+}

uses
   Classes, jsonparser, consoletestrunner,
   tcTests, fpcunittestinsight;

type
   TMyTestRunner = class(TTestRunner)
   end;

var
   Application: TMyTestRunner;

begin
   if IsTestinsightListening() then
     RunRegisteredTests('','')
   else
   begin
     Application := TMyTestRunner.Create(nil);
     Application.Initialize;
     Application.Title := 'FPCUnit Console test runner';
     Application.Run;
     Application.Free;
   end;
end.
```

Figure 3: The Lazarus Test Insight window



Server port: 6789
Server path: /tests
☑ Automatically fetch test list on open

Figure 5: The Lazarus Test Insight configuration panel

5 CONFIGURATION

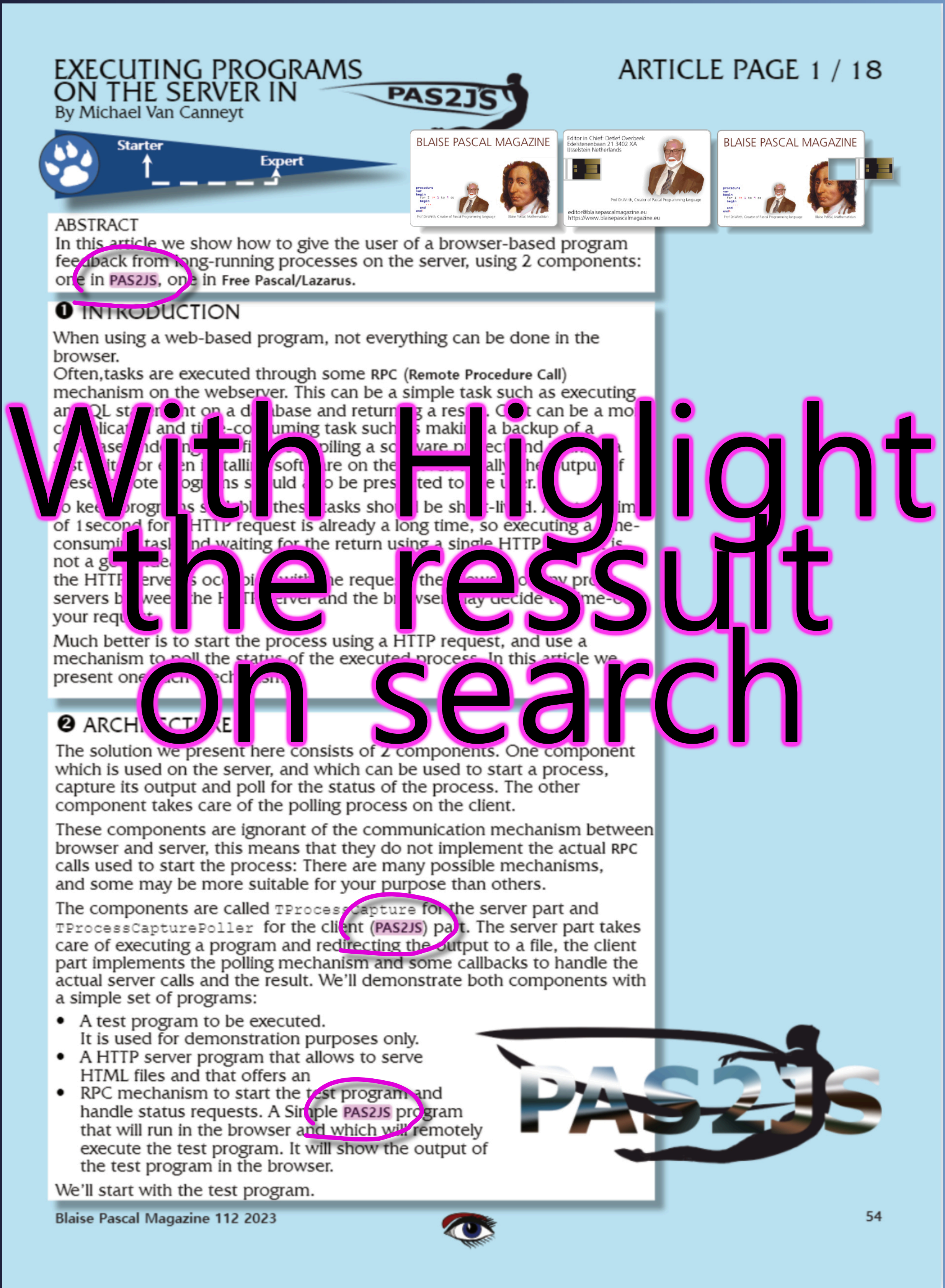
In the IDE, the port and base location on which the server should listen can be set.

In the '**Tools - Options**' menu under **TestInsight**, the following settings can be configured:
Base path this is the path on the HTTP server to which requests must be sent.
The default is `'/tests'`.
Server port this is the path on the HTTP server to which requests must be sent.
The default is 6789
Auto fetch tests When the '**Test Insight**' window is opened, or the current project changes then the list of tests is automatically fetched.
When disabled, you can use the refresh button to get the list of tests.
The settings dialog is shown in *figure 5 on page 10.*

6 CONCLUSION

The '**Test Insight'** functionality makes it easier to do **test-driven development** by allowing you to immediately jump to the implementation of a test or an **`error location`** from within the test result display embedded in the IDE.
The current version of **'Test insight'** is but **an initial version**:
Some extensions are planned, such as **test filtering** and **automatic selection** of tests, and **PAS2JS** support of **testinsight** is also in the works.

LIB-STICK BLAISE PASCAL MAGAZINE



Starter
Expert
BLAISE PASCAL MAGAZINE
Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands
editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu
BLAISE PASCAL MAGAZINE

EXECUTING PROGRAMS ON THE SERVER IN PAS2JS

ARTICLE PAGE 1 / 18

By Michael Van Canneyt

ABSTRACT

In this article we show how to give the user of a browser-based program feedback from long-running processes on the server, using 2 components: one in PAS2JS, one in Free Pascal/Lazarus.

❶ INTRODUCTION

When using a web-based program, not everything can be done in the browser.
Often,tasks are executed through some RPC (Remote Procedure Call) mechanism on the webserver. This can be a simple task such as executing an SQL statement on a database and returning a result. Or it can be a mo[illegible] complicated and time-consuming task such as making a backup of a database, rendering [illegible] files, compiling a software project and [illegible] a test suite, or even installing software on the [illegible]. Ideally the output of these remote programs should also be presented to the user.

To keep programs [illegible] these tasks should be short-lived. A [illegible] time of 1second for HTTP request is already a long time, so executing a time-consuming task and waiting for the return using a single HTTP [illegible] is not a g[illegible] the HTTP server is occupied with the request, the [illegible] or any proxy servers between the HTTP server and the browser may decide to time-out your request.

Much better is to start the process using a HTTP request, and use a mechanism to poll the status of the executed process. In this article we present one such mechanism.

With Higlight the ressult on search

❷ ARCHITECTURE

The solution we present here consists of 2 components. One component which is used on the server, and which can be used to start a process, capture its output and poll for the status of the process. The other component takes care of the polling process on the client.

These components are ignorant of the communication mechanism between browser and server, this means that they do not implement the actual RPC calls used to start the process: There are many possible mechanisms, and some may be more suitable for your purpose than others.

The components are called `TProcessCapture` for the server part and `TProcessCapturePoller` for the client (PAS2JS) part. The server part takes care of executing a program and redirecting the output to a file, the client part implements the polling mechanism and some callbacks to handle the actual server calls and the result. We'll demonstrate both components with a simple set of programs:

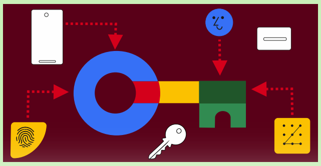
- A test program to be executed. It is used for demonstration purposes only.
- A HTTP server program that allows to serve HTML files and that offers an
- RPC mechanism to start the test program and handle status requests. A Simple PAS2JS program that will run in the browser and which will remotely execute the test program. It will show the output of the test program in the browser.

We'll start with the test program.

Blaise Pascal Magazine 112 2023 54

LIB-STICK USB-CARD: ALL ISSUES / CODE INCLUDED. SAME INTERFACE AS THE INTERNET LIBRARY € 89 CHRISTMAS OFFER

PASSWORDLESS LOGIN WITH PASSKEYS



ABSTRACT

In this article I want to explain the aspects of creating and using passkeys. It looks rather intricate but if you keep following the main goals of it, it is a very interesting and useful way of getting a service that will provide security as well convenience.

PASSKEYS: WHAT ARE THEY?

A **passkey** is a type of digital credential that is connected to an application or website and a user account.
A credential is a document that details a qualification, competence, or authority issued to an individual by a third party with a relevant authority.
Passkeys enable users to log in without requiring any further authentication factors or the entry of a username and password.
Passwords and other antiquated authentication methods are intended to be replaced by this technology.

❶ ADVANTAGES

Passkeys allow you to log in to web services quickly and securely:

- without a password,
- without special hardware
- without the risk of phishing.

I'll try to explain how this password successor works and where you can use passkeys with your PC, (desktop to be precise), smartphone and tablet.

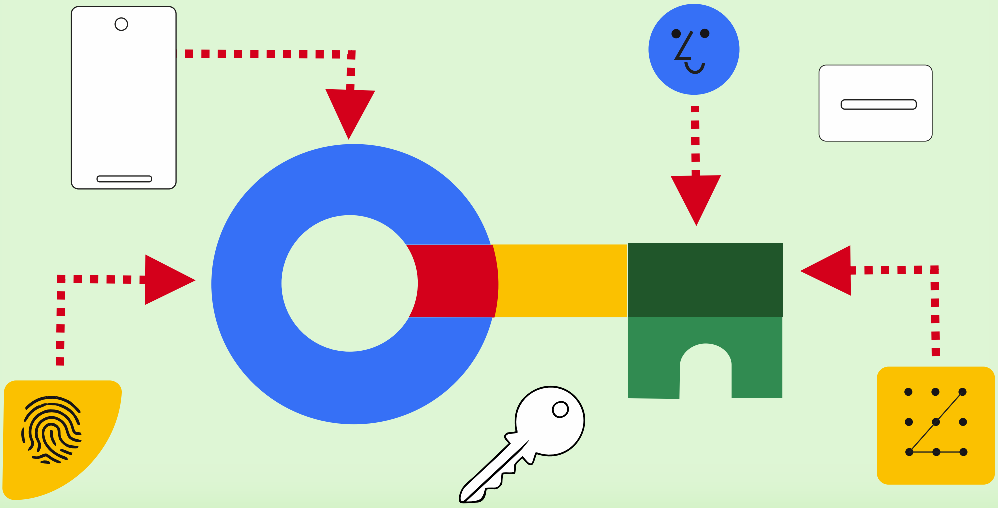
DISADVANTAGES THAT LEAD TO ADVANTAGES

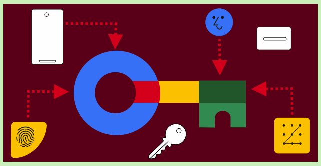
Developers and users both dislike passwords:

- they give a poor user experience,
- they add conversion friction,
- they create security-liability for both users and developers.

The **Password Manager** (*from* **Google**) in **Android** and **Chrome** reduces the friction through autofill; it is good for developers looking for even further improvements in conversion and security, **passkeys** and **identity federation** are the best developers modern approaches.

A passkey can meet **multi factor authentication** requirements in a single step, replacing both: a password and **OTP – One Time Password** - (*e.g. 6-digit SMS code*) to deliver robust protection against phishing attacks, avoids the **UX*** – pain of **SMS** (**Short Message/Messaging Service**) or app-based one-time passwords. *The **U**ser e**X**perience (**UX**) is how a user feels while interacting with a product, system or service





ADVANTAGES
Continuation

Since passkeys are standardized, a single implementation enables a password-less experience across all of a users' devices, across different browsers and operating systems.
Right in the middle where these components meet, you'll find the key. That overlap is the way in which your website should be built, structured, and designed in order to form the perfect User Experience.

Passkeys are safer:

- **Developers only save a public key to the server** instead of a password, meaning there's far less value for a bad actor to hack into servers, and far less clean-up to do in the event of a breach.
- **Passkeys protect users from phishing attacks**.
  Passkeys work only on their registered websites and apps;
- **A user cannot be tricked** into authenticating on a deceptive site because the browser or OS handles verification.
- **Passkeys** reduce costs for sending SMS, making them a **safer and more cost-effective** means for **two-factor authentication.**
- Many intricate processes are carried out discreetly to guarantee security.
  But these activities go unnoticed by individuals.

For you, accessing your account with a passkey is as simple as tapping your smartphone's **fingerprint sensor**, for instance.
Keep in mind that a criminal creative mind will easily make a rubber stamp - just an example.
The increased convenience could potentially convince individuals who are not typically concerned with security to use the new login technique.

❷ CRIMINALS

Every day, cybercriminals take over millions of accounts, usually via password.
Analyses of leaked passwords show that many users tend to choose simple passwords. And on top of that use them for multiple web services.
The protective but cumbersome **two-factor authentication** is often ignored out of convenience.
You can't blame anyone for this.

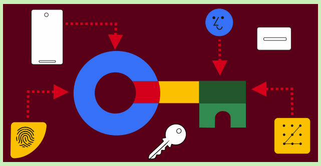
Over time, accounts for websites and apps proliferate because you have to register almost everywhere these days.
If you consistently follow best-practices to secure yourself, you have a lot of work to do.
Not a problem for tech-savvy people, but try explaining it to people of age.

Effectively managing passwords can be challenging due to the fact that the concept predates the internet, is close to the dinosaurs and even far before the invention of the first computer.
Its original purpose did not include the intention of deterring hackers or preventing phishing attempts.
Hence, the utilization of measures such as two-factor authentication is vital in order to continue employing the obsolete procedure in the contemporary era.

PHISHING

Phishing refers to the fraudulent practice of attempting to obtain sensitive information, such as passwords or credit card details, by disguising oneself as a trustworthy entity in that case.
The **EFF Electronic Frontier Foundation has made some comments**
`https://www.eff.org/deeplinks/2023/10/what-passkey`

ELECTRONIC FRONTIER FOUNDATION

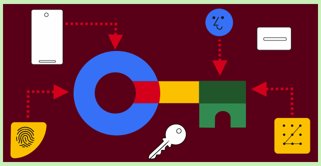


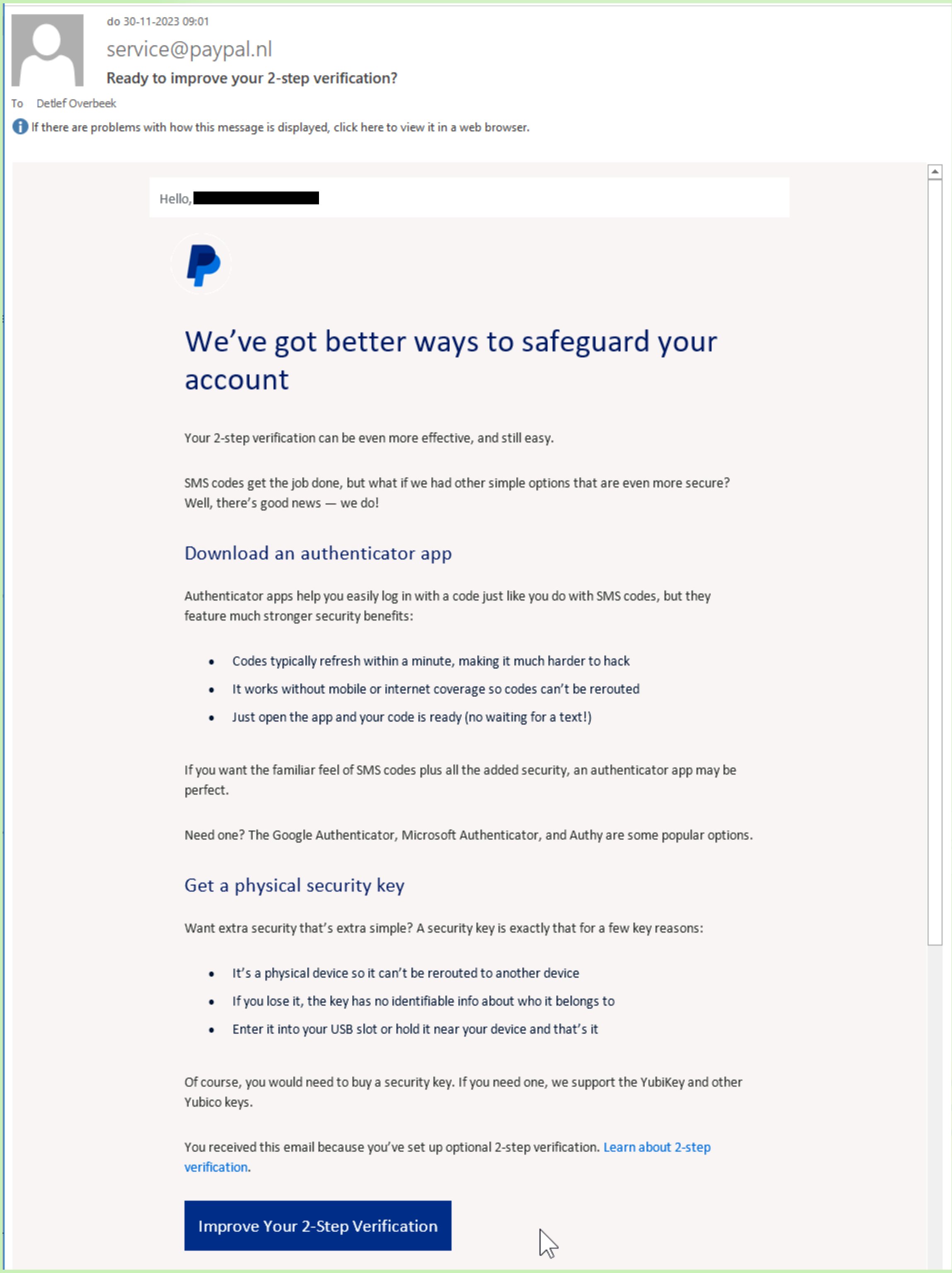
PHISHING - CONTINUATION
Passkeys include information about the specific domain name for which it was generated.
If an individual provides you with an URL to a login page on a domain name that closely resembles the original, you may be deceived.
Your web browser will NOT be, since it has the capability to effortlessly verify for an exact match.
To ensure your safety, your browser will NOT allow transmitting the passkey to the deceptive domain name.
Nevertheless, as long as you retain a memorized password alongside your passkey, a fraudulent website could deceive you by claiming that your passkey is not functioning and prompting you to input the password instead.
In that case entering the password will result in the successful execution of the phishing attack.
Phishing remains a viable threat, but, individuals who habitually use a passkey to access a certain website are more inclined to become wary when prompted to input a password instead. This offers a degree of safeguarding, if not absolute.

PAYPAL EXAMPLE
The authentication inherently encompasses the domain of the website.
This enhances the resilience of passkeys against phishing attempts.
If you access **`PayPal.com`** using a phishing email, you are unable to utilize the passkey that you previously established for **`PayPal.com.`**
Alternatively, you can only generate a fresh passkey for the alternate domain.
This is entirely futile for phishers.
Undoubtedly, it is still imperative that you refrain from entrusting the phishing site with your sensitive information. *See the offer from PayPal on the next page.*

A STAR IS BORN
A new age is emerging with the introduction of passkeys.
Without the need for passwords and devoid of the potential dangers associated with phishing attacks.
The requisite technology is already present in your smartphone, tablet, and computer.
Numerous web services are already equipped for this new advancement.

The Logo for Passkeys





do 30-11-2023 09:01

service@paypal.nl

Ready to improve your 2-step verification?

To Detlef Overbeek

If there are problems with how this message is displayed, click here to view it in a web browser.

Hello, [redacted]

We've got better ways to safeguard your account

Your 2-step verification can be even more effective, and still easy.

SMS codes get the job done, but what if we had other simple options that are even more secure? Well, there's good news — we do!

Download an authenticator app

Authenticator apps help you easily log in with a code just like you do with SMS codes, but they feature much stronger security benefits:

- Codes typically refresh within a minute, making it much harder to hack
- It works without mobile or internet coverage so codes can't be rerouted
- Just open the app and your code is ready (no waiting for a text!)

If you want the familiar feel of SMS codes plus all the added security, an authenticator app may be perfect.

Need one? The Google Authenticator, Microsoft Authenticator, and Authy are some popular options.

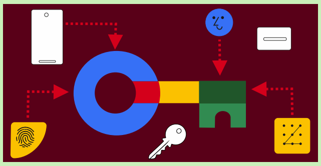
Get a physical security key

Want extra security that's extra simple? A security key is exactly that for a few key reasons:

- It's a physical device so it can't be rerouted to another device
- If you lose it, the key has no identifiable info about who it belongs to
- Enter it into your USB slot or hold it near your device and that's it

Of course, you would need to buy a security key. If you need one, we support the YubiKey and other Yubico keys.

You received this email because you've set up optional 2-step verification. Learn about 2-step verification.



❷ HOW DOES IT WORK?

PASSKEY AUTHENTICATION

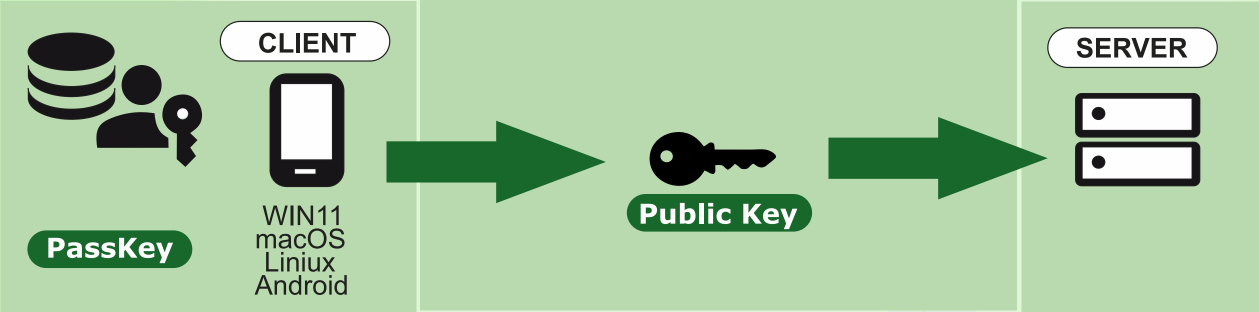
Unlike passwords, with passkey authentication there is no shared secret known to both you and the service that an attacker can intercept. Instead, you have a **private key** and a **public key** for each of your accounts.
Only you hold the important private key.

***1**. When you create a passkey for a web service, the service gets only the public key and links it to your account. With the public key, the service can now verify that you have the private key, without the service ever seeing that private key.

NO SHARED SECRET KNOWN
TO BOTH YOU AND THE SERVICE
THAT AN ATTACKER CAN INTERCEPT

Specifically, this works as follows:
when you log in via a passkey, **the web service provides random data**, called the **challenge**, which your **device signs with your private key**.

The web service checks the digital signature and **can determine with certainty that it comes from your private key.**
That method is called **PUBLIC KEY CRYPTOGRAPHY.**
It has been tried and tested for decades and is also used for **HTTPS** and **e-mail encryption**, for example.



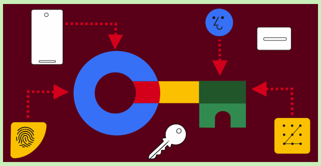
The **EFF** Electronic Frontier Foundation has some comments:

A passkey is **approximately 100-1400 bytes** of **random data,** generated on your device (*like your phone, laptop, or security key*) for the purpose of logging in on a specific website.

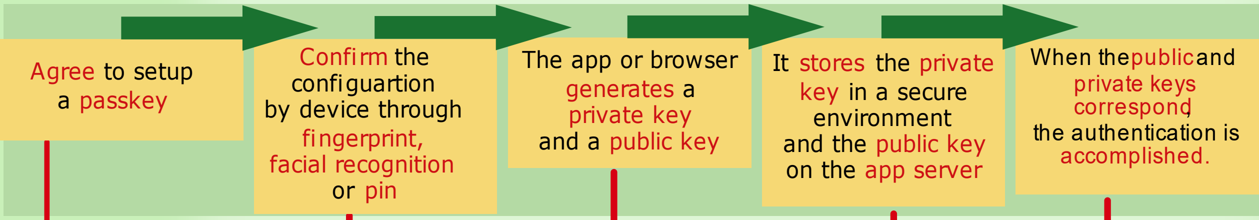
Once the passkey is generated, your browser registers it with the website and it gets stored **somewhere safe** (for instance, your password manager).
I personally would **rather NOT** let this happen and choose an other way to save my key.
Remember this is the part that is private. *See explanation 1**
(The password manager can not copy it to other devices).

From then on, you can use that **passkey to log in to that website** without entering a password.

When you go to a website's login page, you'll have the option to "**Sign in with a passkey.**"
If you choose that option you'll get a confirmation prompt from your password manager, and will be logged in after confirming.
For all this to work, there needs to be **passkey support** in the website, your browser, your password manager, and usually **also your operating system**.



PASSKEY INITIATION AND FLOW



Agree to setup a passkey

Confirm the configuartion by device through fingerprint, facial recognition or pin

The app or browser generates a private key and a public key

It stores the private key in a secure environment and the public key on the app server

When thepublicand private keys correspond the authentication is accomplished.

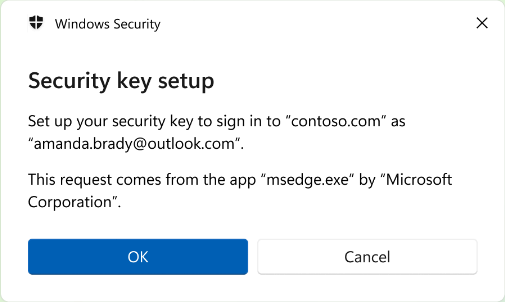
Typically, when **users attempt to log in** to an application that **utilizes passkeys**, a dialogue box will appear, prompting them to save a **passkey (usually a fingerprint, face ID, or pattern**) for future authentication purposes. The user must verify and continue.

Upon the user's **agreement to utilize passkey-based authentication**, the application will proceed to **generate a cryptographic key pair** consisting of a **public key** and a **private key**. This method is **performed locally** on the user's device, which **can include their computer, tablet**, or **smartphone**.

During the procedure, the **private key is securely stored in a wallet**, while the **public key is sent to the application server** for storage.

On the user's next login attempt, the application server will transmit an **encrypted puzzle** to the user's device, utilizing the device's public key. To **verify** the identity of the user attempting to log in, the user's device will **decipher** the puzzle and transmit it back to the server.

After the **completion of the public and private key matching verification** process, the user is able to successfully access the application.

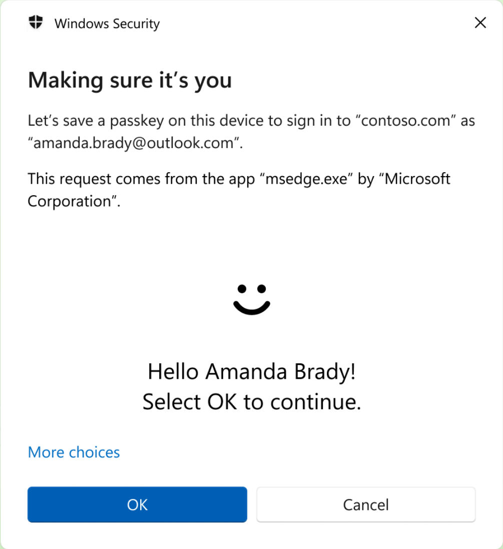


Windows Security

Security key setup

Set up your security key to sign in to “contoso.com” as “amanda.brady@outlook.com”.

This request comes from the app “msedge.exe” by “Microsoft Corporation”.



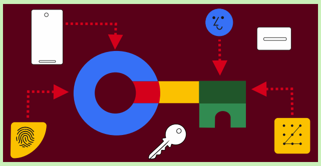
Windows Security

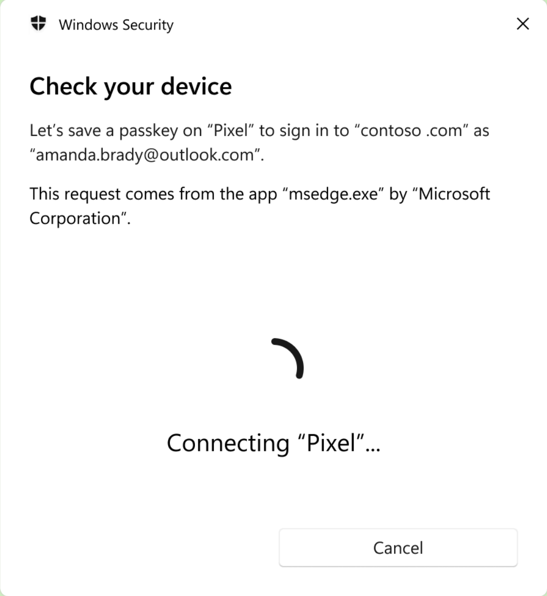
Making sure it’s you

Let’s save a passkey on this device to sign in to “contoso.com” as “amanda.brady@outlook.com”.

This request comes from the app “msedge.exe” by “Microsoft Corporation”.

Hello Amanda Brady!
Select OK to continue.



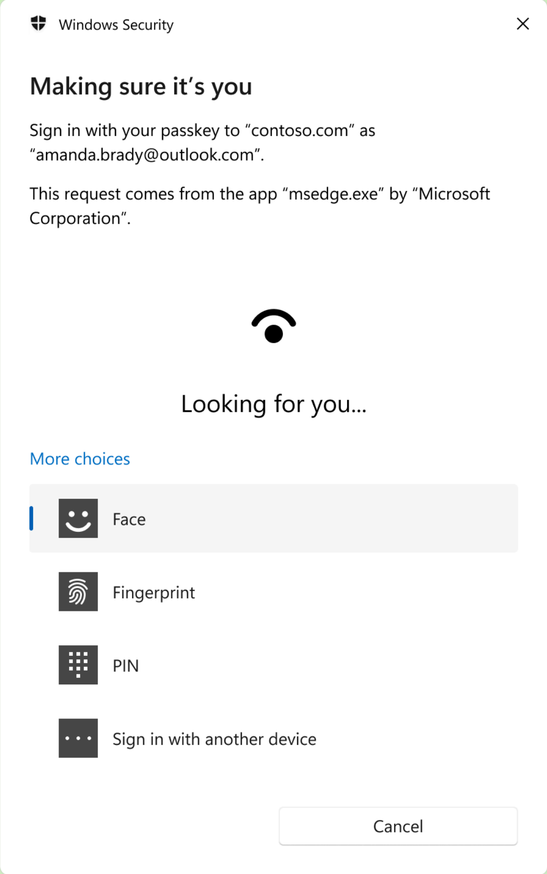


Check your device

Let's save a passkey on "Pixel" to sign in to "contoso .com" as "amanda.brady@outlook.com".

This request comes from the app "msedge.exe" by "Microsoft Corporation".

Connecting "Pixel"...

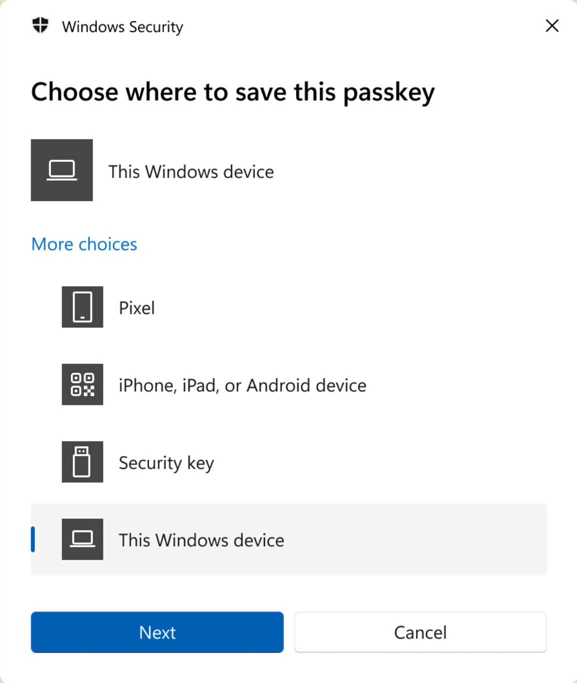


Making sure it's you

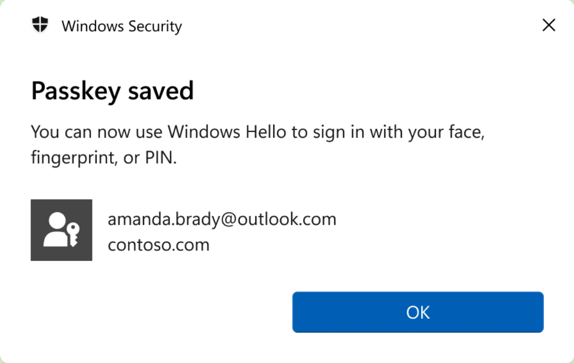
Sign in with your passkey to "contoso.com" as "amanda.brady@outlook.com".

This request comes from the app "msedge.exe" by "Microsoft Corporation".

Looking for you...



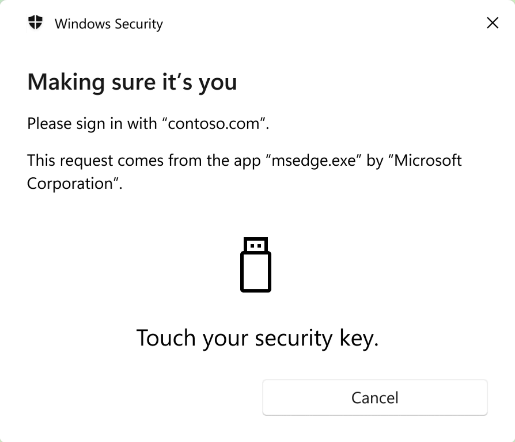
Choose where to save this passkey



Passkey saved

You can now use Windows Hello to sign in with your face, fingerprint, or PIN.

amanda.brady@outlook.com
contoso.com

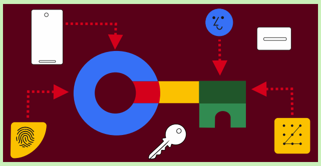


Making sure it's you

Please sign in with "contoso.com".

This request comes from the app "msedge.exe" by "Microsoft Corporation".

Touch your security key.



- You can **create many passkeys:**

each passkey unlocks a single account on a single website.
For multiple accounts on a single website, you can have multiple passkeys for that website.
For instance, if you have a social media account for personal use and one for business, you would have different passkeys for each account.

- You can usually **have both a password and a passkey** on your account, and can log in with either. Logging in with a passkey is generally faster, since your password manager will offer to do it in a single click, instead of the multiple clicks that logging in with a password usually takes.

Also, logging in with a passkey typically lets you skip traditional two-factor authentication (*SMS, authenticator app, or security key*).

Passkeys build in a second factor.
Each time you use the passkey to log in, your browser or operating system may ask you to re-enter your device unlock **PIN**.
If you use a **fingerprint** or **facial recognition** to unlock your device, your browser might instead request you re-enter your fingerprint or show your face, to confirm that it's really you asking to log in.
I personally dislike the facial recognition. It is nowadays very simple to create a picture as a facial mask

That gives two factors of authentication:
the device that stores your passkey is something you have, and it's accompanied by something you know (the **PIN**) or something you are (**a fingerprint or a face**).

Instead of utilizing a password, you will be provided with a unique passkey for each individual account.

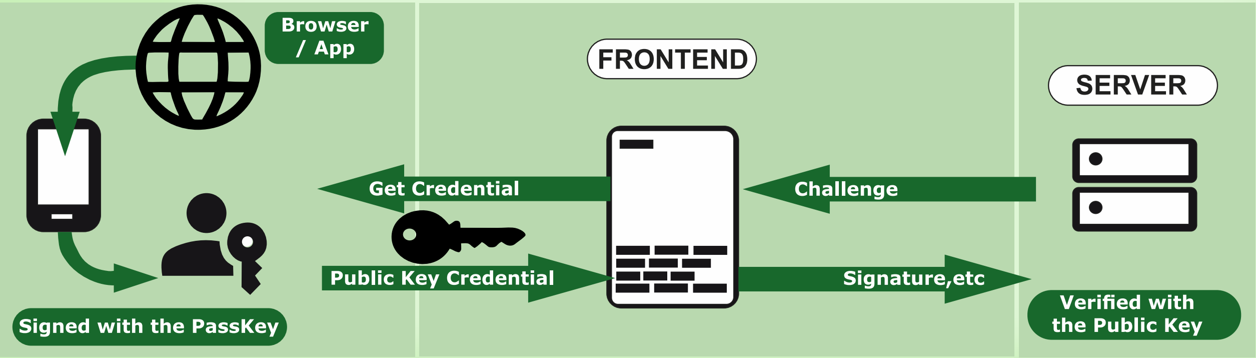
- The **key is saved** on your mobile device, PC, or tablet.
- To log in, simply select the passkey login option on the appropriate page.
- Verify the utilization of the passkey by employing a concise PIN.

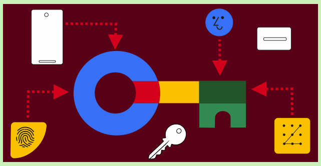
This is the same for all passkeys.

It may be potentially expedited by utilizing your fingerprint or facial scan:
Upon completion, you will be logged in promptly.
Be cautious of these biological characteristics as they can be convincingly counterfeited in modern times.
The **PIN** or **biometrics** you provide are exclusively utilized on-site to access the passkey and are **never transmitted elsewhere**.

The passkey is automatically safeguarded by a second factor, with the first factor being the passkey itself. All this without the need for separate and burdensome **two-factor authentication.**

A passkey is **essentially a cryptographic key pair** that is automatically **produced by a security chip** on your computer or smartphone.
The requirement to generate a new passkey for each registration, in order to adhere to the requirements of the website, will hopefully become obsolete in the near future.





STORAGE AND BACKUP
A passkey stored on just one computer or phone alone isn't that useful.
Can one log in from a different device?
What if your device is stolen, lost or falls apart ?

There are a number of answers here for **storage and or backup**.
But each is a different story

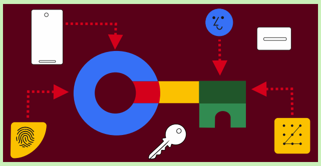
That is a reason for passkeys to be seen as having a special option for circumstances to be solved but I think they are conflicting in their strategy...

❶ You could store them in the **password manager,** which encrypts them, it which makes them vulnerable - if you can crack the storage... and than backs them up to the cloud.
It helps you copy them onto all of your devices.
I have no believe or trust in "the cloud".
It's actually a marketing name for some thing ancient:
the computer via a internet connection or cable. Its out of your hands.
It also is against the rule not to let anyone have the key.

❷ there is always the old fashioned way of saving it under a list of items and keep that safely at your office, but it can of course be found and then stolen. You will have to manually ad the key to the device at being asked

❸ There is also a **physical solution:** a **USB** stick
Passkeys are created and stored in a physical security key that you plug in via USB3.
To log in on an other device, you plug in the security key when prompted.
This can even be done with your mobile
Passkeys created this way can't be copied.
Only recently-made security keys support this.

❹ Passkeys are created and stored on a **high-security chip built into your computer or phone** (for instance, a **TPM** or **Secure Enclave**, available on most devices made in the last few years).
A **TPM (Trusted Platform Module)**
is used to improve the security of your PC.
It's used by services like BitLocker drive encryption, Windows Hello, and others,
to securely create and store cryptographic keys, and to confirm that the operating system
and firmware on your device are what they're supposed to be, and haven't been tampered with
A **Secure Enclave**
Modern computing devices often have a special hardware chip dedicated to
storing critically important information like encryption keys and hashes.
In PCs, this is a **Trusted Platform Module (TPM)**, *while in mobile devices like* **Androids** *and* **iPhones**, *it's called a* **Secure Enclave**..



Like point ❸, these passkeys can't be copied.
Answer 3 and 4 are less convenient (and answer 3 costs some money: to buy a security key).
But they offer a higher level of security against someone stealing your devices.
With Answer 1, someone who steals your computer might be able to copy the passkeys if your password manager is unlocked.

Also, solutions 3 and 4 don't really solve the "device got lost" problem.
If you're using one of those answers, you should have multiple passkeys stored on different devices as backup.
Alternatively you may wind up **relying on email-based account recovery.**

IS IT ADVISABLE TO UTILIZE PASSKEYS?

As with other matters concerning security and privacy, the response is contingent upon various factors.
However, passkeys are generally advisable for the majority of individuals.
If you are currently utilizing a **password manager**, creating lengthy and distinct passwords for every website, and consistently employing the autofill functionality for logging in (*as opposed to manually copying and pasting passwords*), passkeys will offer a somewhat elevated level of protection along with considerably enhanced ease.

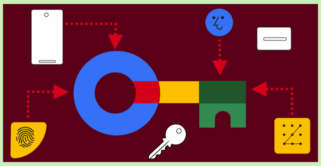
If you are not currently using a password management, implementing passkeys will significantly enhance your security measures (*and will also necessitate the adoption of a password manager*).

When using two factor authentication (**2FA**) on websites, passkeys offer greater convenience and potentially enhanced security.

The **SMS** or authenticator app **2FA** solutions are susceptible to phishing attacks as fraudulent websites can request the one-time code from users and subsequently transmit it, along with the phished password, to the legitimate website.

Passkeys offer enhanced security compared to **SMS** or authenticator app **2FA** because to their immunity to phishing attacks.
Unlike **SMS** or authenticator apps, passkeys are not susceptible to being deceived by fraudulent websites as the browser can accurately associate each passkey with its corresponding site.

Security key **2FA** is not susceptible to phishing attacks, hence transitioning from security key **2FA** to a passkey primarily offers convenience by eliminating an additional login step and the need to remember an extra password.
By storing your passkeys on a security key that is safeguarded with a PIN or biometric authentication, you can attain comparable outcomes to using security key two-factor authentication (**2FA**).
Storing passkeys in a **password manager** reduces security to some extent, as unauthorized individuals who obtain access to the password manager can utilize the passkeys without requiring actual possession of the security key.



As recently as 2023, the level of support for passkeys is highly inconsistent, especially when it comes to **synchronisation**.
Adam Langley (*Senior staff engineer at Google*) provides examples of the limitations of different password management systems.
He states that **Windows Hello** does not synchronize at all, **Google Password Manager** only synchronizes between **Android** devices, and **iCloud Keychain** exclusively functions on **Apple** devices. Even after resolving those issues, the synchronisation of data between different ecosystems (such as iOS and Windows) will continue to be a significant challenge. Third-party password organizers such as 1Password, Bitwarden, and Dashlane offer passkey functionality and can synchronize data across several platforms. However, it should be noted that not all platforms are currently supported by these services. For example, as of October 2023, 1Password does not provide complete support for passkeys on Android.

If you wish to experiment with passkeys on a disposable account, you have the option to create one on **`passkeys.io`** or **`webauthn.io.`**
If you enjoy being at the forefront of technological advancements, I encourage you to give passkeys a try. During your journey, you may encounter obstacles and be forced to resort to the longstanding and contested method of using a password.

Users can select an account to sign in with.
Typing the username is not required.

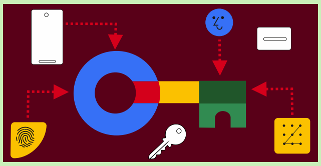
- Users can authenticate using device's screen lock such as a fingerprint sensor, facial recognition or PIN.
- Once a passkey is created and registered, the user can seamlessly switch to a new device and immediately use it without needing to re-enroll (unlike traditional biometric auth, which requires setup on each device).

When a user wants to sign in to a service that uses passkeys, their browser or operating system will help them select and use the right passkey.
The experience is similar to how saved passwords work today.

To make sure only the rightful owner can use a passkey, the system will ask them to unlock their device. This may be performed with a biometric sensor as explained before, PIN, or pattern. You can either choose a non biological base or a pin or pattern
To create a passkey for a website or application, a user first must register with that website or application.

1. Go to the website/application and sign in using the existing sign-in method.
2. Click Create a passkey button.
3. Check the information stored with the new passkey.
4. Use the device screen unlock to create the passkey.

Actually this all happens if you use a mobile phone.
For WINDOWS is a different approach...



PASSKEY SUPPORT ON ANDROID AND CHROME

Passkeys can be **synchronized across devices** in the same ecosystem.
For example, passkeys created on **Android** are stored in the **Google Password Manager**.
NOTE: Starting from **Android 14,** users can opt to use **third-party credential management apps** to store their passkeys.

Passkeys are an emerging technology and supported environments are still evolving.
As of August 2023, **Chrome** on **macOS** and **Windows** stores passkeys on the **local device only.**

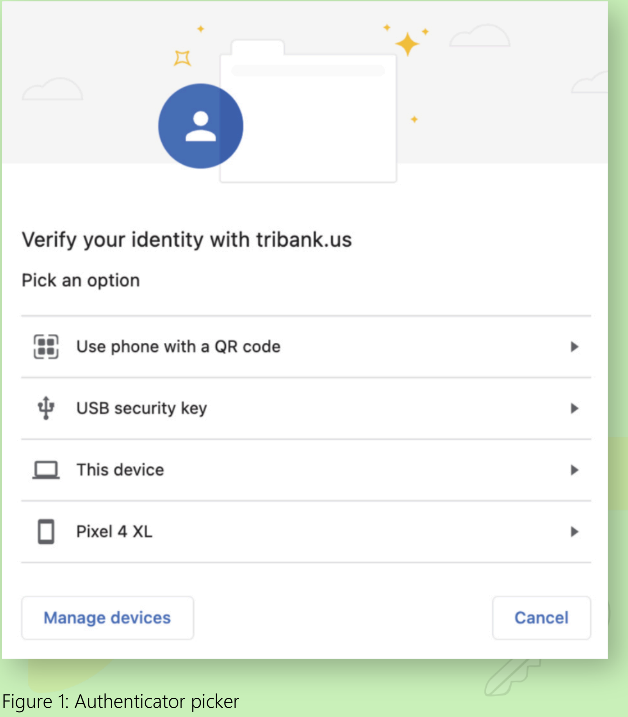
GOOGLE PASSWORD MANAGER

Google Password Manager stores, serves and synchronizes passkeys on **Android** and **Chrome**.
Passkeys from **Google Password Manager** are available to all **Android** apps, including **Chrome** and **other browsers**.
When the user creates a passkey on an **Android** device it's stored and synchronized with their other **Android** devices, and their **passkey secrets are encrypted end-to-end**.

This makes **passkeys available** to the user **across all Android devices** that use **Google Password Manager** and are signed in with the same **Google Account.**
Google Password Manager on **Chrome** helps ***create*** and ***sign in*** with **passkeys**.

Depending on the desktop operating system (*e.g.* **Chrome OS**, **iOS**, **macOS**, **Windows**) users may be presented with a **QR code** to securely use a **passkey** stored on their **mobile device**, or a **notification** may be **displayed** prompting the user to **unlock their phone** to use **the relevant passkey.**

CHROME'S PASSKEY SUPPORT ON DIFFERENT OPERATING SYSTEMS



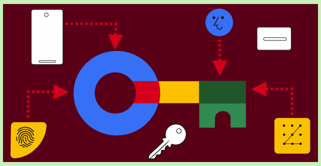
Verify your identity with tribank.us

Pick an option

- Use phone with a QR code ▸
- USB security key ▸
- This device ▸
- Pixel 4 XL ▸

Manage devices   Cancel

Figure 1: Authenticator picker



Chrome on all desktop platforms supports using passkeys from mobile devices. To use a passkey from your Android or iOS device, select the appropriate option when asked. *See Figure 1: on page xx Article page.* To learn more about how to use a phone to sign in, read **Sign-in with a phone**.
https://developers.google.com/identity/passkeys/use-cases#sign-in-with-a-phone
The following sections outline **Chrome** behaviour on different operating systems.

ANDROID
Chrome on **Android OS 9** or later supports passkeys. **Passkeys** generated in **Chrome** on **Android** are stored in the **Google Password Manager**. These **passkeys** are available on all other **Android** devices as long as **Google Password Manager** is available and the same user's **Google Accoun**t is signed in.

WINDOWS
Chrome on **Windows** stores passkeys in **Windows Hello**, which **doesn't synchronize** them to other devices as of October 2023.
When a user tries to **sign in to a website for the first time** on Chrome on Windows, **they should scan a QR code** with another device **that already has a passkey**. After that, they can create a passkey on the local **Windows device** for future use there.

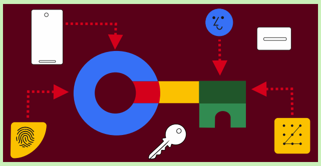
MACOS
Chrome on **macOS 13.5** and later can use **iCloud Keychain** to store passkeys. **Passkeys** in **iCloud Keychain** are **synchronized across the user's Apple devices** and **can be used by other browsers** and apps.
Chrome on **macOS** can also store passkeys in a **local profile**, which means **they aren't synchronized** to other devices. Storing passkeys in a local profile is available in earlier versions of macOS.

IOS / IPADOS
Chrome on **iOS 16** and **iPadOS 16** uses **iCloud Keychain** to store passkeys. **Passkeys** in **iCloud Keychain** are synchronized across the user's Apple devices and can be used by other browsers and apps.

LINUX
Chrome on **Linux** **doesn't support passkeys with a built-in platform authenticator.**
Linux users can **use passkeys from another device** such as an **Android phone** or an **iPhone** by scanning a **QR code**

Operating System	**Android**	**macOS**	**iOS/iPadOS**	**Windows**	**Linux**
Local user verification	☑	☑	☑	☑	🚫
Passkey sync	☑	☑[1]	☑[1]	🕓[3]	🚫
Autofill	☑	☑	☑	☑[2]	🚫
Can Sign with a mobyle	☑	☑	☑	☑	☑

☑ :Supported, 🕓 : Planned, 🚫 : No plans
[1] :Syncs with iCloud [2] : Requires Windows 11 22H2 [3] :Depends on Windows Hello



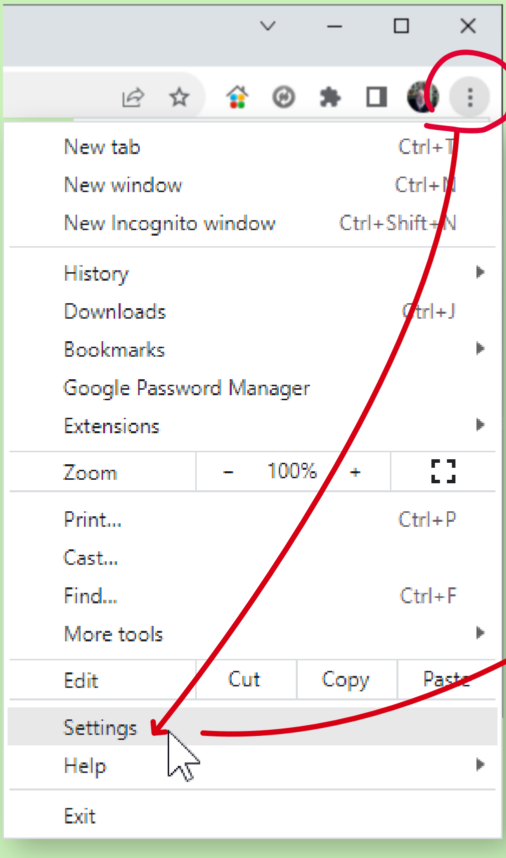


Figure 1 Button and Settings

To find the right settings follow these steps:

- open your browser → at the top-right you will see a button with **three dot**s. If you click on that the window drops down. See the left image.
- You see **Settings**.
- Click on that and the next page shows up:
- Here is **Security.** Choose **Security**: a new small window pops up. Here you need to click on Manage your account

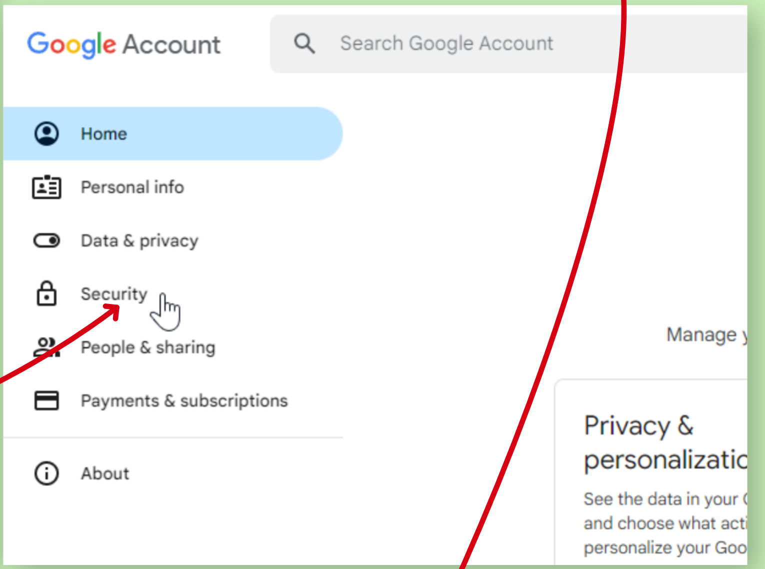


Figure 2 Security must be chosen

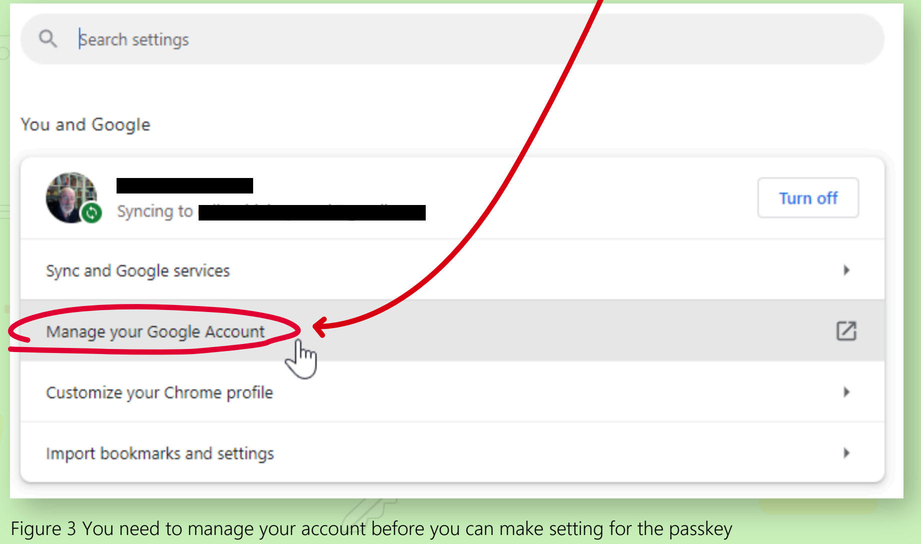
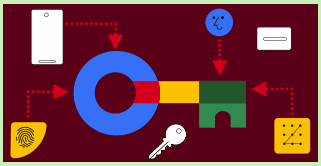
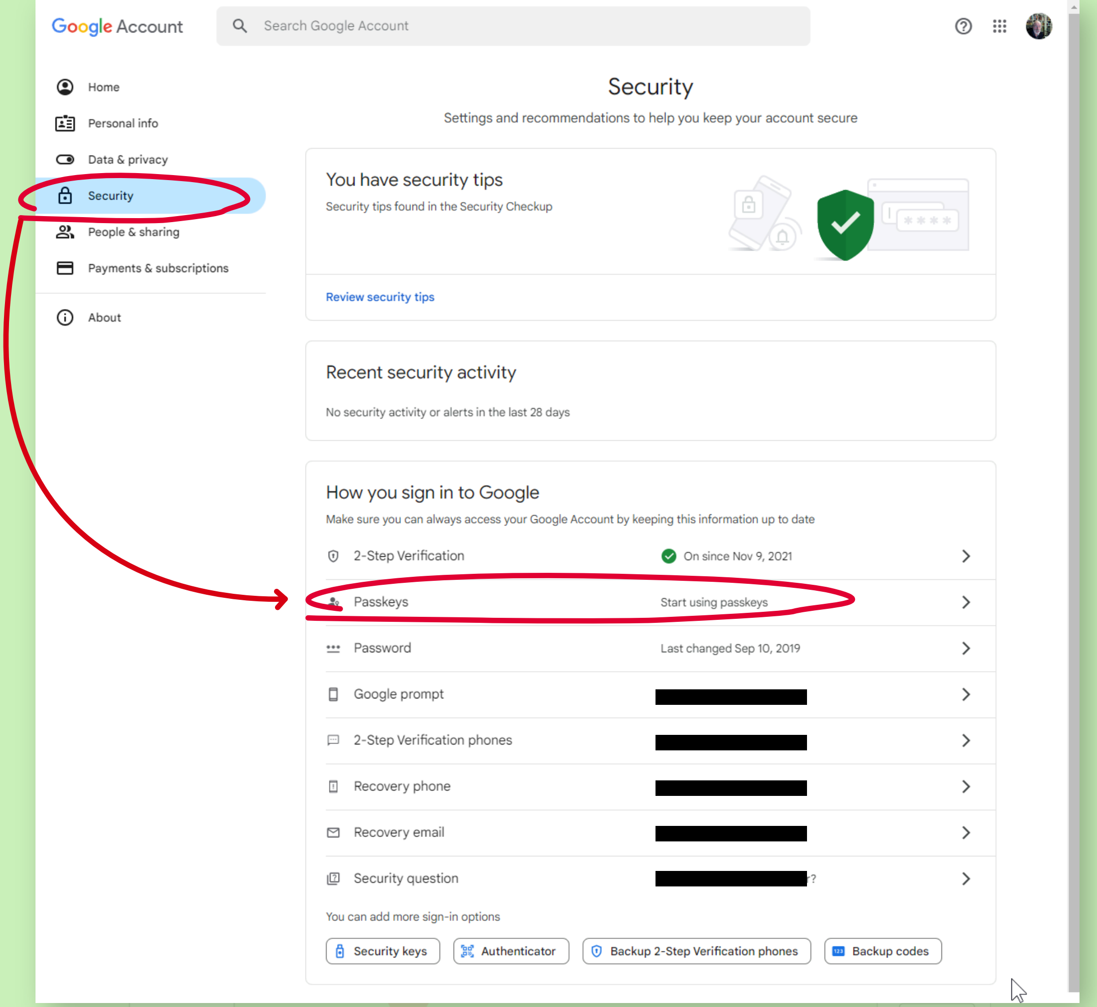


Figure 3 You need to manage your account before you can make setting for the passkey





Security

Settings and recommendations to help you keep your account secure

You have security tips

Security tips found in the Security Checkup

Review security tips

Recent security activity

No security activity or alerts in the last 28 days

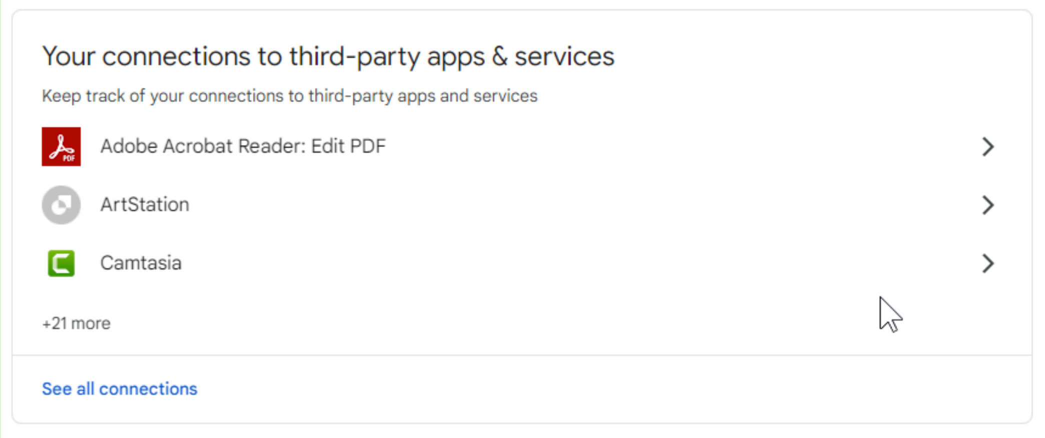
How you sign in to Google

Make sure you can always access your Google Account by keeping this information up to date

2-Step Verification	On since Nov 9, 2021
Passkeys	Start using passkeys
Password	Last changed Sep 10, 2019
Google prompt	[redacted]
2-Step Verification phones	[redacted]
Recovery phone	[redacted]
Recovery email	[redacted]
Security question	[redacted]?

You can add more sign-in options

Figure 4 Choosing security will show a list of items : Choose passkeys



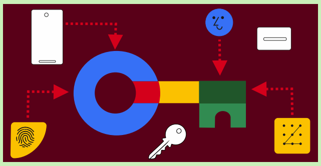
Your connections to third-party apps & services

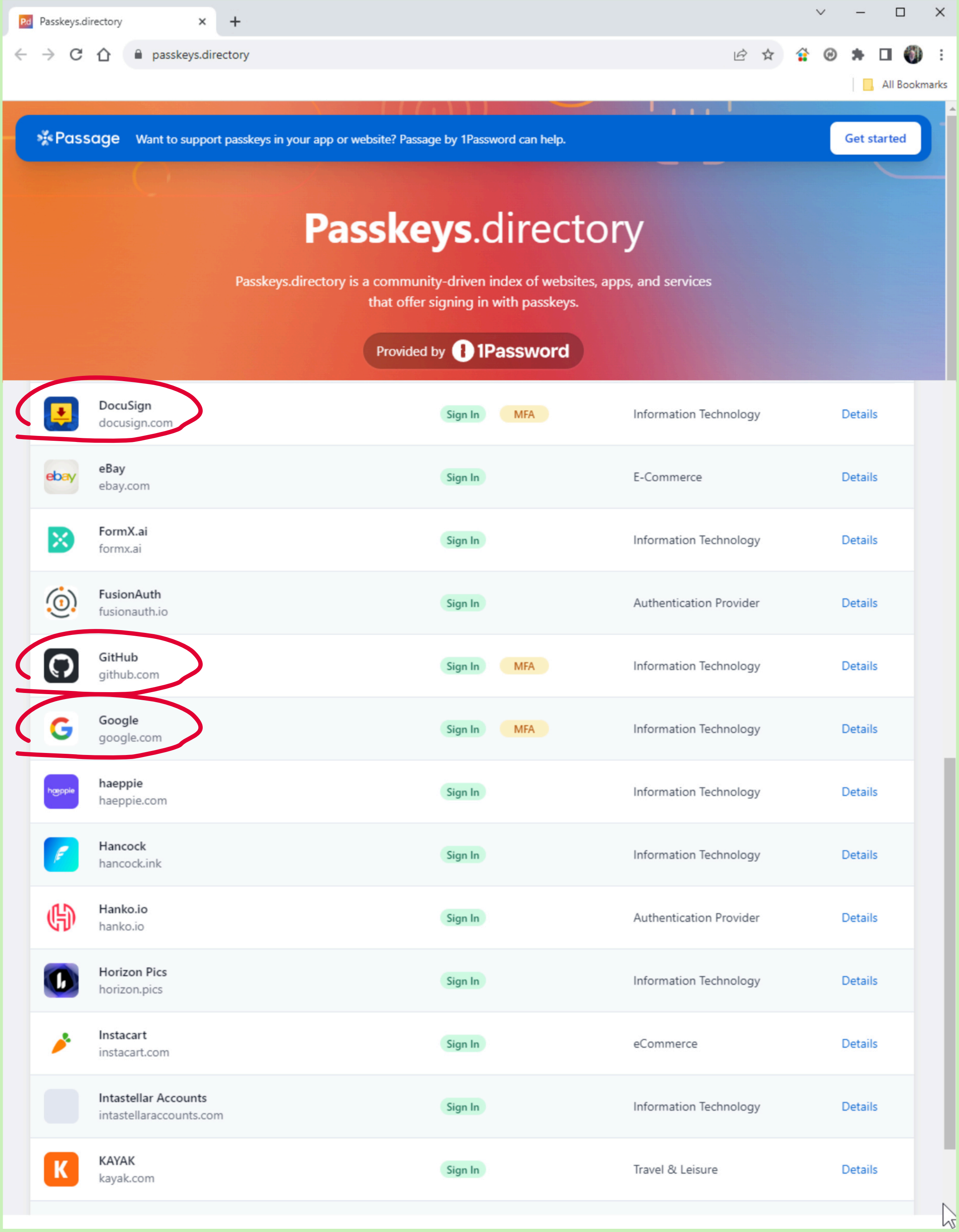
Keep track of your connections to third-party apps and services

- Adobe Acrobat Reader: Edit PDF
- ArtStation
- Camtasia

+21 more

See all connections





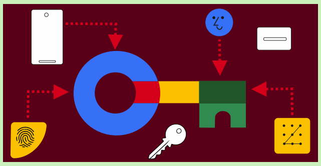
Want to support passkeys in your app or website? Passage by 1Password can help.

Passkeys.directory

Passkeys.directory is a community-driven index of websites, apps, and services that offer signing in with passkeys.

Provided by 1Password

DocuSign docusign.com	Sign In	MFA	Information Technology	Details
eBay ebay.com	Sign In		E-Commerce	Details
FormX.ai formx.ai	Sign In		Information Technology	Details
FusionAuth fusionauth.io	Sign In		Authentication Provider	Details
GitHub github.com	Sign In	MFA	Information Technology	Details
Google google.com	Sign In	MFA	Information Technology	Details
haeppie haeppie.com	Sign In		Information Technology	Details
Hancock hancock.ink	Sign In		Information Technology	Details
Hanko.io hanko.io	Sign In		Authentication Provider	Details
Horizon Pics horizon.pics	Sign In		Information Technology	Details
Instacart instacart.com	Sign In		eCommerce	Details
Intastellar Accounts intastellaraccounts.com	Sign In		Information Technology	Details
KAYAK kayak.com	Sign In		Travel & Leisure	Details



FREQUENTLY ASKED QUESTIONS

- Do passkeys work on devices that don't have a screen lock method set up?
  It depends on the password manager implementation, whether a credential provider allows for a passkey creation and authentication without a user knowledge factor challenge.
  Providers can prompt users to set up a PIN or biometric screen lock before creating a passkey.

- **How can passkeys** registered on one platform (*such as* **Android**) be used to sign in on other platforms (*such as* **web** *or* **iOS**)?
  A passkey registered on Android, for example, can be used to sign-in on other platforms by connecting the Android phone with another device.

https://developers.google.com/identity/passkeys/use-cases#sign-in-with-a-phone

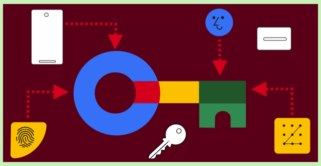
  To establish a connection between the two devices users need to open the site they are trying to sign in to on a device that doesn't have a passkey registered,
  scan a QR code, and then confirm the sign-in on the device they had created the passkey on (*in this case, the* **Android** *device*).

THE PASSKEY NEVER LEAVES THE ANDROID DEVICE,
so typically apps will suggest creating a **new passkey** on the other device to facilitate the sign-in the next time. This flow will work in a similar way for other platforms as well.

- **Can I move synchronized passkeys** from one platform provider to another?
  Passkeys are saved to the credential provider defined by the platform.
  Some platforms, like **Android**, allow users to choose the provider of their choice (*a system or third-party password manager*) **starting in Android 14**,
  which may be able to synchronize passkeys across different platforms.
  Support for **moving passkeys** directly from one platform provider to another is not available at this time.

- **Can a user synchronize their passkeys across non-Google Android devices?**
  Passkeys are only synced within the device's **ecosystem** (*that is*, **Android to Android** *with Google Password Manager by default*), but not across the ecosystem.
  Android is opening up the platform (**starting in Android 14**) to allow users to select which **credential provide**r they want to use (*such as a third-party password manager*).
  That will enable use cases like synchronizing passkeys between different ecosystems (*depending on how open other platforms are*).

- What should developers do about **devices and platforms that don't support passkeys**?
  Developers are recommended to keep the existing sign-in options in their app for the time being so that they will continue to be available for devices and surfaces that do not support passkeys.



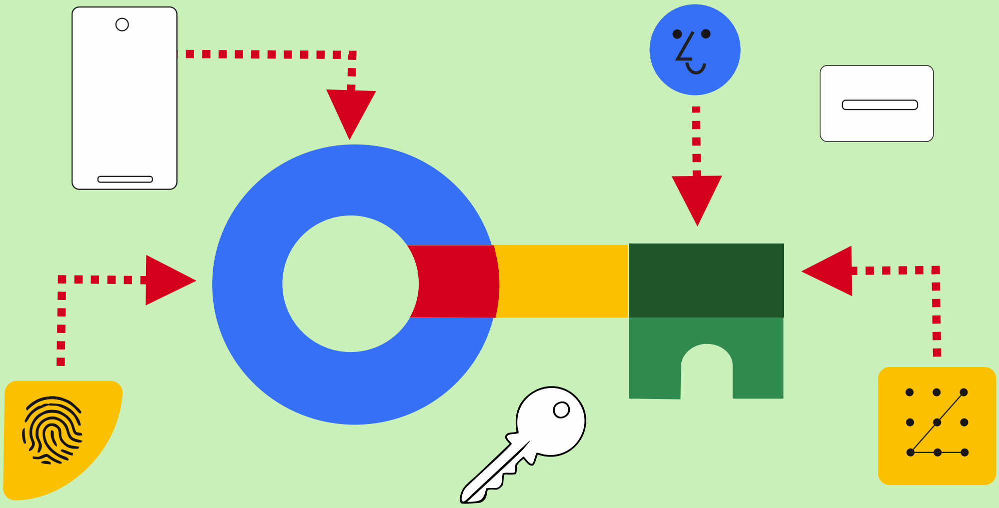
- Can an RelyingParty specify an account for the user to sign in with?
  Relying parties (*third-party apps*) can populate the '**allowCredentials**' with a list of credential IDs sent from their app backend indicating which passkeys should be used to authenticate the user.
  NOTE: The first step to enable passkey support for your **Android** app is to associate your app and the website.
  To do so, host a **Digital Asset Links JSON** file on your website, and add a link to the **Digital Asset Link** file to your app's manifest.
  This demonstrates that you own both the website and the app.
  To learn more, check out the documentation on **Digital Asset Links**.

- For passkeys created in **Chrome** on other platforms:
  If the passkey is created in **Chrome** on other platforms (*Mac, iOS, Windows*), then no.
  Check out the supported environments for more information.
  Meanwhile, **users can use the phone they created the passkey on to sign in.**

https://developer.android.com/training/sign-in/passkeys
https://developers.google.com/identity/passkeys
https://developers.google.com/identity/passkeys/use-cases

TRY IT YOURSELF
You can try passkeys in this demo:

https://passkeys-demo.appspot.com/
https://medium.com/androiddevelopers/bringing-seamless-authentication-to-your-apps-using-credential-manager-api-b3f0d09e0093

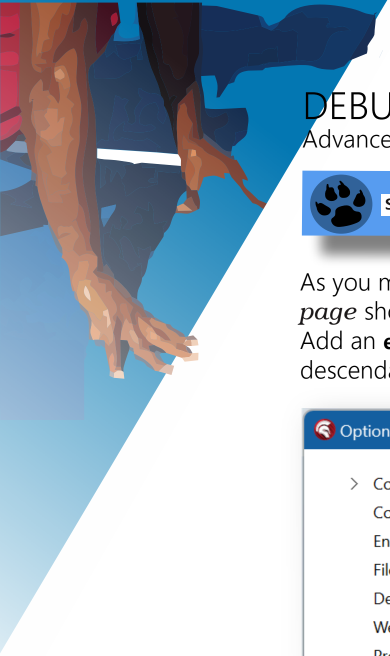


POCKET PACKAGE (2BOOKS) PRICE: € 30,00 EXCLUDING VAT AND SHIPPING

LAZARUS HANDBOOK

CHRISTMAS OFFER 2023

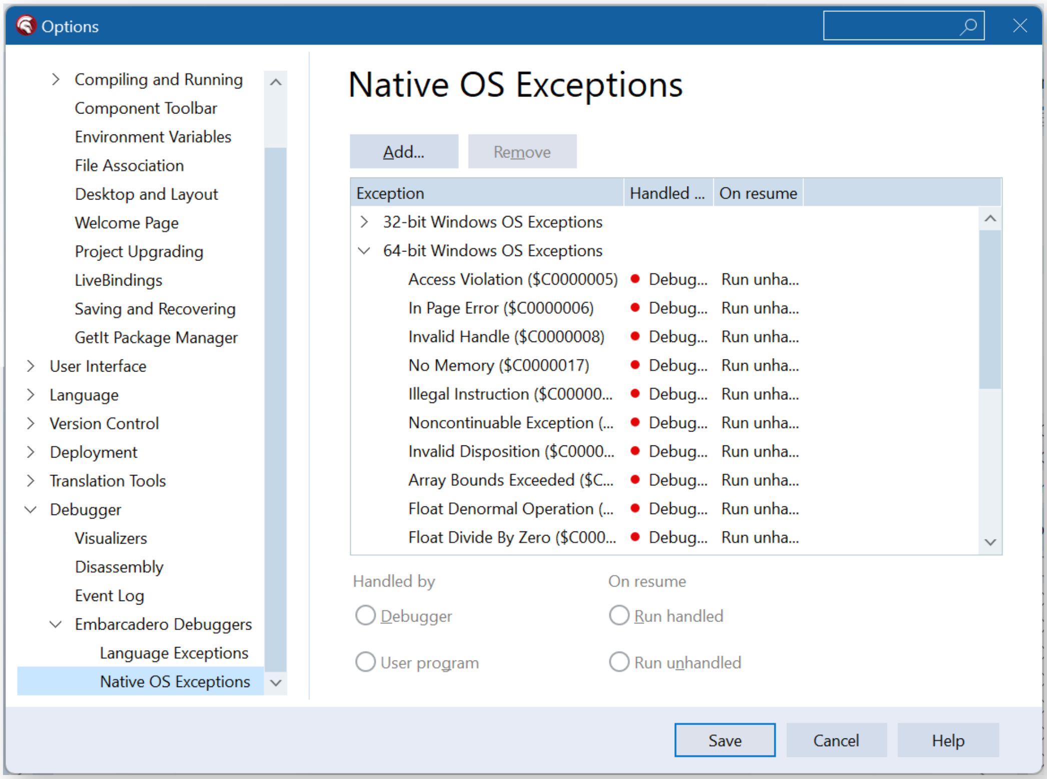
https://www.blaisepascalmagazine.eu/product-category/books/



DEBUGGING A 64-BIT-BOX

Advanced Debugging under the hood with RAD Studio 11.3 Alexandria. bij Max Kleiner

As you my know you can tell the debugger to ignore certain kinds of exceptions. *Figure 1 on this page* shows **Delphi's language-exception** and **native OS exception** options.
Add an **exception class** to the list in **language-exceptions**, and all exceptions of that type and of any descendant types will pass through to your program without **Delphi** interfering.



Native OS Exceptions

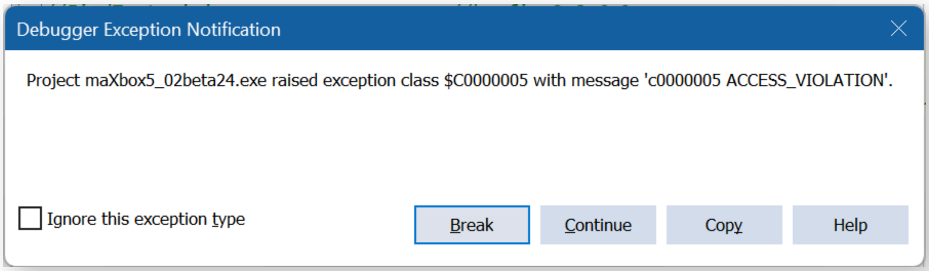
Exception	Handled ...	On resume
32-bit Windows OS Exceptions		
64-bit Windows OS Exceptions		
Access Violation ($C0000005)	● Debug...	Run unha...
In Page Error ($C0000006)	● Debug...	Run unha...
Invalid Handle ($C0000008)	● Debug...	Run unha...
No Memory ($C0000017)	● Debug...	Run unha...
Illegal Instruction ($C00000...	● Debug...	Run unha...
Noncontinuable Exception (...	● Debug...	Run unha...
Invalid Disposition ($C0000...	● Debug...	Run unha...
Array Bounds Exceeded ($C...	● Debug...	Run unha...
Float Denormal Operation (...	● Debug...	Run unha...
Float Divide By Zero ($C000...	● Debug...	Run unha...

Handled by
○ Debugger
○ User program

On resume
○ Run handled
○ Run unhandled

Figure 1

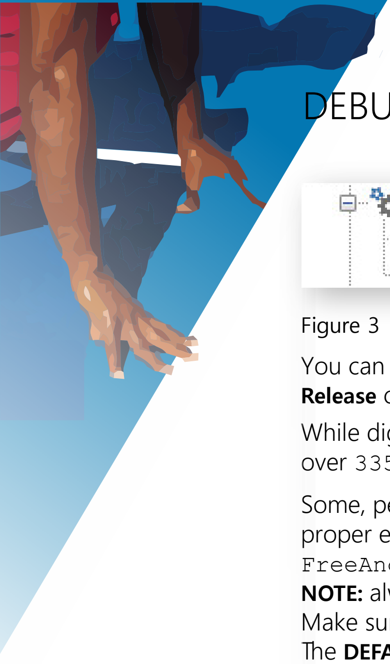
In its default settings, the **Delphi IDE** notifies you whenever an `exception` occurs in your program, as in *Figure 2 on this page*. What's important to realize is at that point, none of your program's `exception-handling` code has run yet.
It's all **Delphi** itself; its special status as a debugger allows it to get first notification of any `exception` in your program, even before your program knows about it.
So its not that easy open your **64-bit application in the IDE,** add and activate the **64-bit debugger options**, and compile your application as a **64-bit Windows application** with the right debugging.

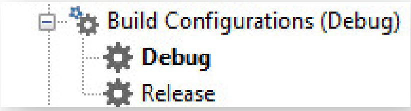


Project maXbox5_02beta24.exe raised exception class $C0000005 with message 'c0000005 ACCESS_VIOLATION'.

□ Ignore this exception type

Figure 2





```
Build Configurations (Debug)
    Debug
    Release
```

Figure 3

You can change option values in any configuration, including Base. You can delete the **Debug** and **Release** configurations, but you cannot delete the **Base** configuration or move it.

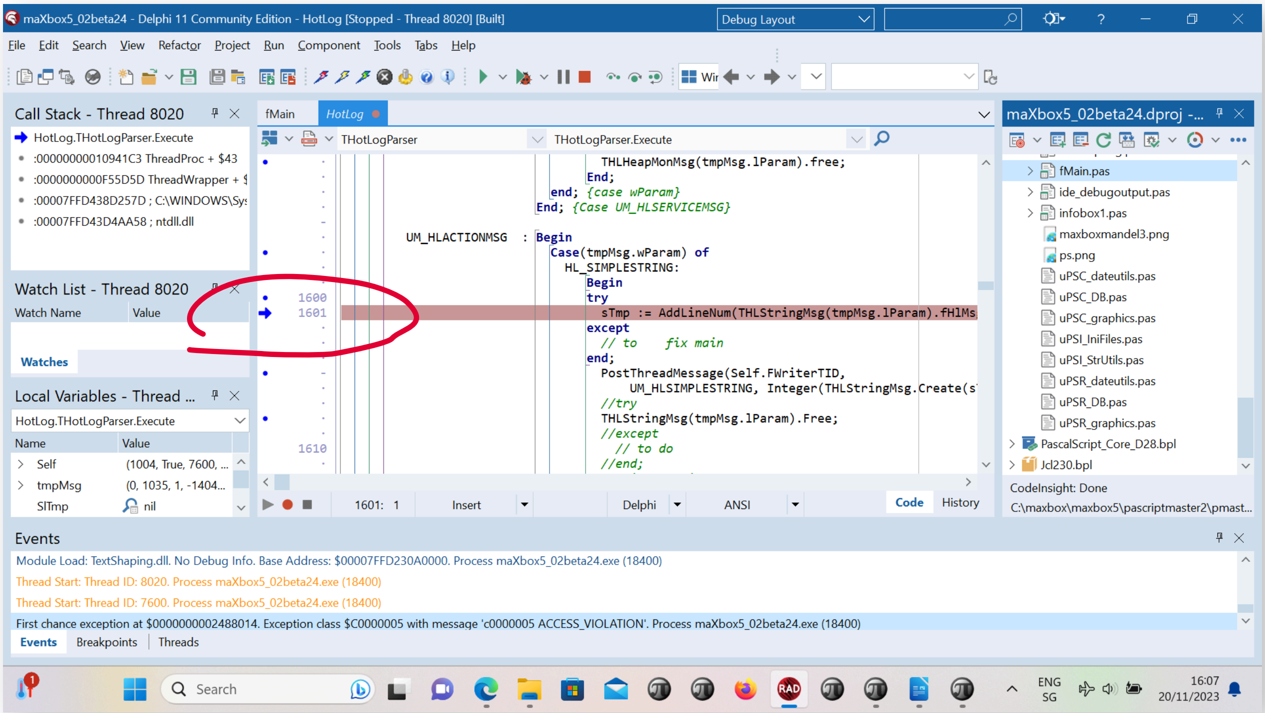
While digging or diving across the source code of **maXbox4** it seems to be impossible to migrate over `3354 units` in a decent and proper way to **maXbox5** aka **64-bit Version**.

Some, people were complaining about it like it creates problems, but without actually providing a proper example. Additionally, **Embarcadero** recently added the recommendation to use `FreeAndNil` in their manual (*finally!*).
NOTE: always compile the application in **Release and Debug mode**.
Make sure the **Project Options** are correctly set for Debug mode.
The **DEFAULT** settings for **Debug mode are NOT correct/complete** - at least not in **Delphi XE7** and **Tokyo**. Once upon a time they will set the correct options for Debug mode if any exists.
So, enable and study subjects like:

- **"Stack frames"**
- **"Map file generation (detailed)"**
- **"Range checking",**
- **"Symbol reference info"**
- **"Debug information"**
- **"Overflow checking"**
- **"Assertions"**
- **"Debug DCUs"**

Most of the time you get some **first chance exceptions**. After you break the **debugger exception notification** in *the figure* you get exactly the line to fix the bug (*it was a Nil pointer*):

Figure 4



maXbox5_02beta24 - Delphi 11 Community Edition - HotLog [Stopped - Thread 8020] [Built]

Call Stack - Thread 8020

```
HotLog.THotLogParser.Execute
:00000000010941C3 ThreadProc + $43
:0000000000F55D5D ThreadWrapper + $
:00007FFD438D257D ; C:\WINDOWS\Sys
:00007FFD43D4AA58 ; ntdll.dll
```

Watch List - Thread 8020

Watch Name	Value

Watches

Local Variables - Thread ...

HotLog.THotLogParser.Execute

Name	Value
Self	(1004, True, 7600, ...
tmpMsg	(0, 1035, 1, -1404...
SlTmp	nil

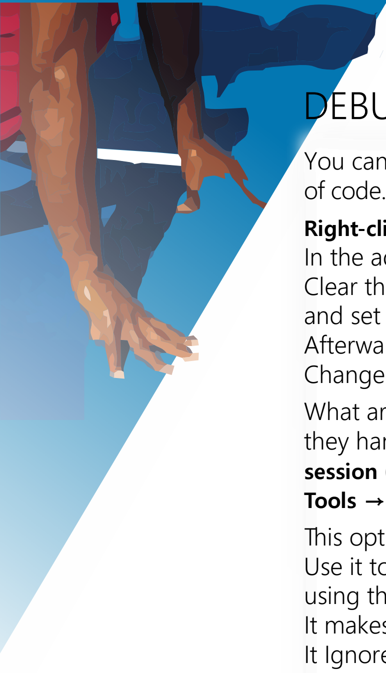
```
                THLHeapMonMsg(tmpMsg.lParam).free;
              End;
            end; {case wParam}
          End; {Case UM_HLSERVICEMSG}

UM_HLACTIONMSG  : Begin
            Case(tmpMsg.wParam) of
              HL_SIMPLESTRING:
                Begin
                try
                  sTmp := AddLineNum(THLStringMsg(tmpMsg.lParam).fHlMs
                except
                  // to    fix main
                end;
                  PostThreadMessage(Self.FWriterTID,
                      UM_HLSIMPLESTRING, Integer(THLStringMsg.Create(s
                  //try
                  THLStringMsg(tmpMsg.lParam).Free;
                  //except
                    // to do
                  //end;
```

1601: 1 Insert Delphi ANSI

CodeInsight: Done
C:\maxbox\maxbox5\pascriptmaster2\pmast...

Events

```
Module Load: TextShaping.dll. No Debug Info. Base Address: $00007FFD230A0000. Process maXbox5_02beta24.exe (18400)
Thread Start: Thread ID: 8020. Process maXbox5_02beta24.exe (18400)
Thread Start: Thread ID: 7600. Process maXbox5_02beta24.exe (18400)
First chance exception at $0000000002488014. Exception class $C0000005 with message 'c0000005 ACCESS_VIOLATION'. Process maXbox5_02beta24.exe (18400)
```



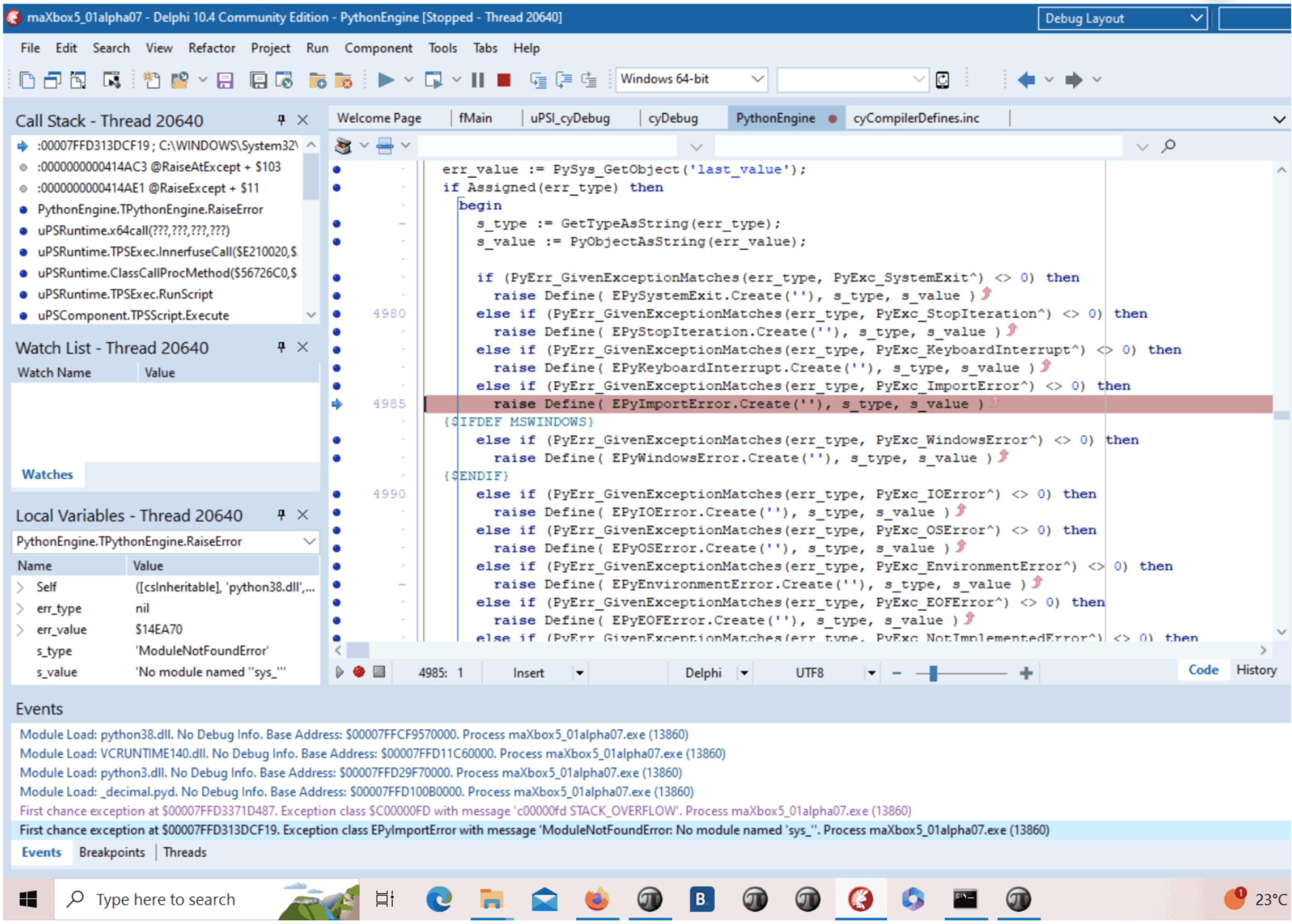
You can use **Delphi's "advanced breakpoints"** to disable `exception handling` around a region of code. To begin, set a **breakpoint** on the line of code where you want the **IDE** to ignore exceptions.

Right-click on the **breakpoint** dot in the gutter and open the **breakpoint-property dialogue**.
In the advanced section are some check boxes.
Clear the "Break" box to prevent the **debugger** from interrupting your program at that line,
and set the **"Ignore subsequent exceptions"** box.
Afterwards, set another breakpoint where you want the debugger to resume handling exceptions.
Change its properties to handle subsequent exceptions.

What are **subsequent exceptions:**
they handle all subsequent exceptions **raised by the current process** during the **current debug session** (*the debugger will stop on exceptions based on the current exception settings in:* **Tools → Options → Debugger Options → Embarcadero Debuggers → Language Exceptions).**

This option does **stop on all exceptions**.
Use it to turn on **normal exception behaviour** after another breakpoint **disabled normal behaviour** using the **Ignore subsequent exceptions** option.
It makes sense in a block of code with the option Ignore subsequent exceptions.
It Ignores all subsequent exceptions raised by the current process during the current debug session (*the debugger will not stop on any exception*).
Normally you know **Halts** execution: the traditional and default action of a breakpoint).
Use this **Ignore subsequent exceptions** with **Handle subsequent exceptions** as a pair.
You can surround specific blocks of code with the **Ignore/Handle** pair to skip any exceptions which occur in that block of code like in the following *Figure 5 on this page*.

Figure 5



Call Stack - Thread 20640

```
:00007FFD313DCF19 ; C:\WINDOWS\System32\
:0000000000414AC3 @RaiseAtExcept + $103
:0000000000414AE1 @RaiseExcept + $11
PythonEngine.TPythonEngine.RaiseError
uPSRuntime.x64call(???,???,???,???)
uPSRuntime.TPSExec.InnerfuseCall($E210020,$
uPSRuntime.ClassCallProcMethod($56726C0,$
uPSRuntime.TPSExec.RunScript
uPSComponent.TPSScript.Execute
```

Local Variables - Thread 20640

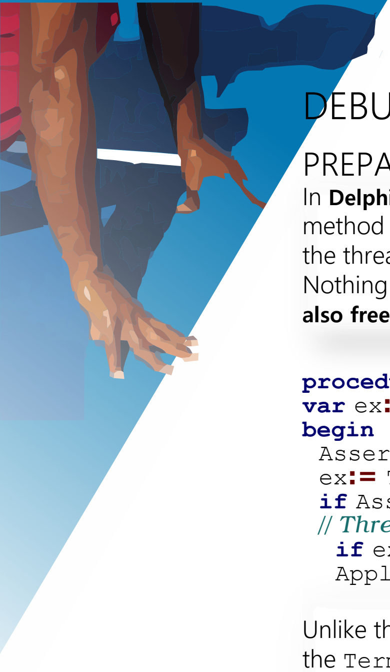
PythonEngine.TPythonEngine.RaiseError

Name	Value
Self	([csInheritable], 'python38.dll',...
err_type	nil
err_value	$14EA70
s_type	'ModuleNotFoundError'
s_value	'No module named ''sys_'''

```
  err_value := PySys_GetObject('last_value');
  if Assigned(err_type) then
    begin
      s_type := GetTypeAsString(err_type);
      s_value := PyObjectAsString(err_value);

      if (PyErr_GivenExceptionMatches(err_type, PyExc_SystemExit^) <> 0) then
        raise Define( EPySystemExit.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_StopIteration^) <> 0) then
        raise Define( EPyStopIteration.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_KeyboardInterrupt^) <> 0) then
        raise Define( EPyKeyboardInterrupt.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_ImportError^) <> 0) then
        raise Define( EPyImportError.Create(''), s_type, s_value )
  {$IFDEF MSWINDOWS}
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_WindowsError^) <> 0) then
        raise Define( EPyWindowsError.Create(''), s_type, s_value )
  {$ENDIF}
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_IOError^) <> 0) then
        raise Define( EPyIOError.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_OSError^) <> 0) then
        raise Define( EPyOSError.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_EnvironmentError^) <> 0) then
        raise Define( EPyEnvironmentError.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_EOFError^) <> 0) then
        raise Define( EPyEOFError.Create(''), s_type, s_value )
      else if (PyErr_GivenExceptionMatches(err_type, PyExc_NotImplementedError^) <> 0) then
```

Events

```
Module Load: python38.dll. No Debug Info. Base Address: $00007FFCF9570000. Process maXbox5_01alpha07.exe (13860)
Module Load: VCRUNTIME140.dll. No Debug Info. Base Address: $00007FFD11C60000. Process maXbox5_01alpha07.exe (13860)
Module Load: python3.dll. No Debug Info. Base Address: $00007FFD29F70000. Process maXbox5_01alpha07.exe (13860)
Module Load: _decimal.pyd. No Debug Info. Base Address: $00007FFD100B0000. Process maXbox5_01alpha07.exe (13860)
First chance exception at $00007FFD3371D487. Exception class $C00000FD with message 'c00000fd STACK_OVERFLOW'. Process maXbox5_01alpha07.exe (13860)
First chance exception at $00007FFD313DCF19. Exception class EPyImportError with message 'ModuleNotFoundError: No module named 'sys_''. Process maXbox5_01alpha07.exe (13860)
```



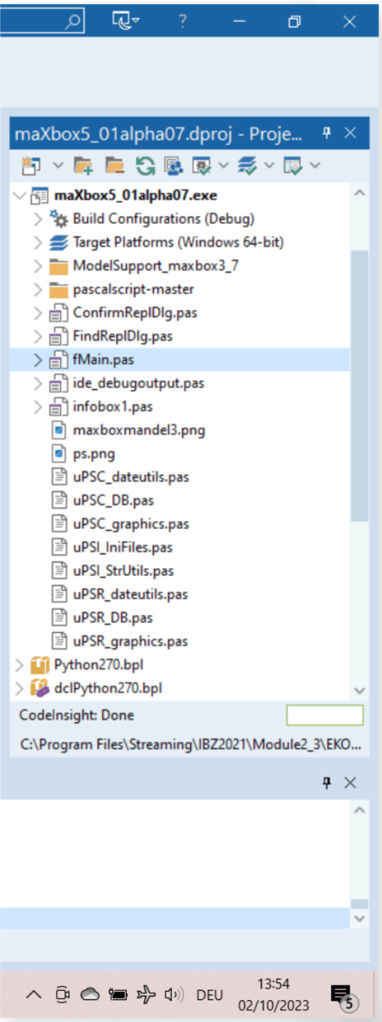
PREPARE TO DEBUG THE DEBUGGER

In **Delphi 11** — and probably most other versions — if an **`exception`** escapes from the **`Execute`** method without being handled, then it is caught by the function that called `Execute` and stored in the thread's **`FatalException`** property. (See in `Classes.pas`, `ThreadProc`.)
Nothing further is done with that exception until the **thread is freed,** at which point the **exception is also freed**.

```
procedure TForm1.Onterminate(Sender: TObject);
var ex: TObject;
begin
 Assert(Sender is TThread);
 ex:= TThread(Sender).FatalException;
 if Assigned(ex) then begin
 // Thread terminated due to an exception
   if ex is Exception then
   Application.ShowException(Exception(ex))
```

Unlike the **Windows API TerminateThread MSDN**, which forces the thread to terminate immediately, the `Terminate` method merely requests that the **thread** terminate.
This allows the thread to perform any cleanup and finalisation before it shuts down.
To catch the `exceptions` that occur inside your `thread` function, add a `try...except` block to the implementation of the `Execute` method!
So I prepared the Project and debugging as Debug mode like this: *Figure 6 on this page*

Figure5 Continuation

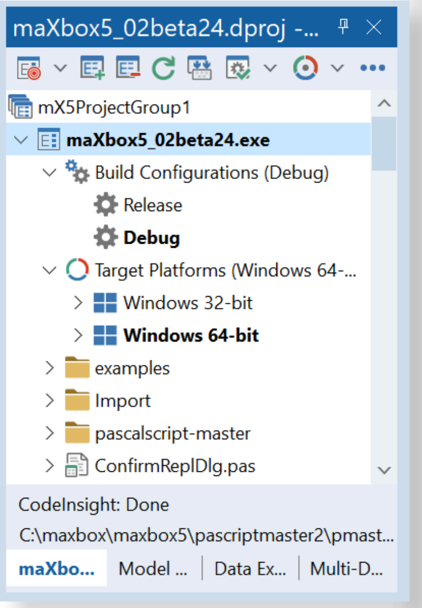


maXbox5_01alpha07.dproj - Proje...

- maXbox5_01alpha07.exe
  - Build Configurations (Debug)
  - Target Platforms (Windows 64-bit)
  - ModelSupport_maxbox3_7
  - pascalscript-master
  - ConfirmReplDlg.pas
  - FindReplDlg.pas
  - fMain.pas
  - ide_debugoutput.pas
  - infobox1.pas
  - maxboxmandel3.png
  - ps.png
  - uPSC_dateutils.pas
  - uPSC_DB.pas
  - uPSC_graphics.pas
  - uPSI_IniFiles.pas
  - uPSI_StrUtils.pas
  - uPSR_dateutils.pas
  - uPSR_DB.pas
  - uPSR_graphics.pas
- Python270.bpl
- dclPython270.bpl

CodeInsight: Done

C:\Program Files\Streaming\IBZ2021\Module2_3\EKO...



maXbox5_02beta24.dproj -...

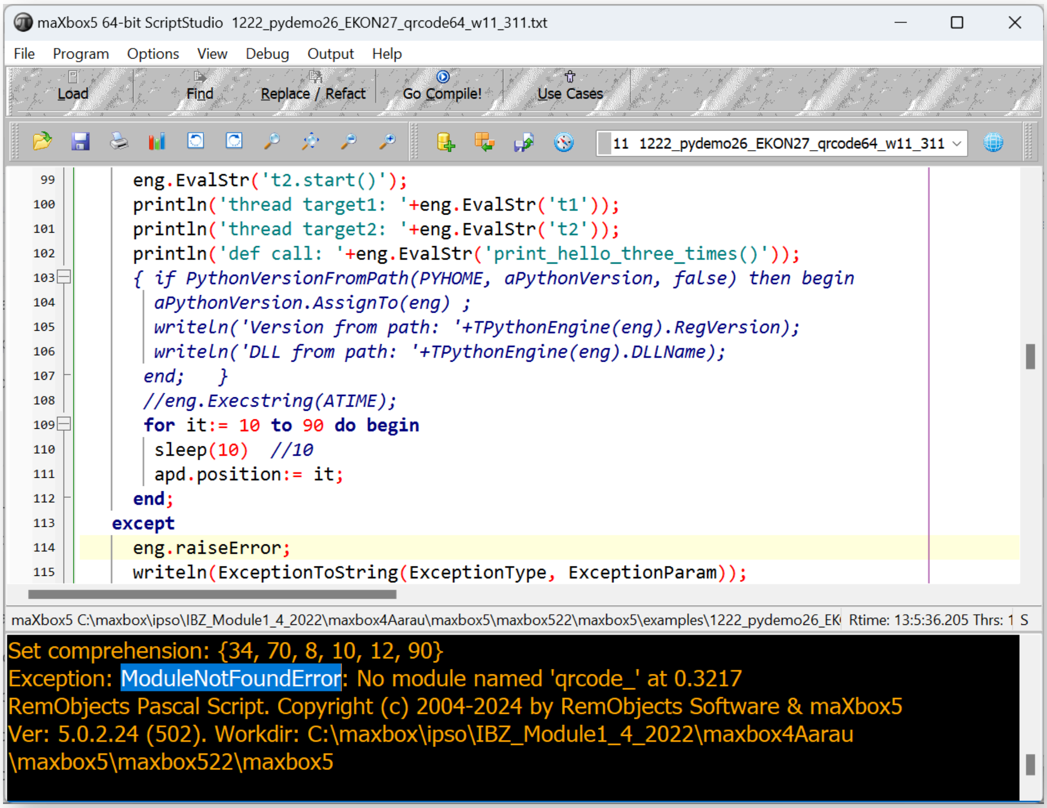
- mX5ProjectGroup1
- maXbox5_02beta24.exe
  - Build Configurations (Debug)
    - Release
    - **Debug**
  - Target Platforms (Windows 64-...
    - Windows 32-bit
    - **Windows 64-bit**
  - examples
  - Import
  - pascalscript-master
  - ConfirmReplDlg.pas

CodeInsight: Done

C:\maxbox\maxbox5\pascriptmaster2\pmast...

Figure 6

The odd thing is that I have wrapped my **Pascal** call in a **`try except`**, which has handlers for **`AccessViolationException, COMException`** and everything else, but when **Delphi 10.4** or **Studio 11.3** intercepts the **`AccessViolationException`**, the debugger breaks on the method call (`doc.OCR`), and if I step through, it continues to the next line instead of entering the catch or `except` block.
So I decided to to catch the exception on the script in a runtime routine to get the most on my console from a **dynamic debugbox maXbox:** → *See next page*


maXbox5 64-bit ScriptStudio 1222_pydemo26_EKON27_qrcode64_w11_311.txt
File Program Options View Debug Output Help
Load
Find
Replace / Refact
Go Compile!
Use Cases
11 1222_pydemo26_EKON27_qrcode64_w11_311

```
      eng.EvalStr('t2.start()');
      println('thread target1: '+eng.EvalStr('t1'));
      println('thread target2: '+eng.EvalStr('t2'));
      println('def call: '+eng.EvalStr('print_hello_three_times()'));
      { if PythonVersionFromPath(PYHOME, aPythonVersion, false) then begin
        aPythonVersion.AssignTo(eng) ;
        writeln('Version from path: '+TPythonEngine(eng).RegVersion);
        writeln('DLL from path: '+TPythonEngine(eng).DLLName);
       end;   }
       //eng.Execstring(ATIME);
       for it:= 10 to 90 do begin
        sleep(10)  //10
        apd.position:= it;
      end;
    except
      eng.raiseError;
      writeln(ExceptionToString(ExceptionType, ExceptionParam));
```

maXbox5 C:\maxbox\ipso\IBZ_Module1_4_2022\maxbox4Aarau\maxbox5\maxbox522\maxbox5\examples\1222_pydemo26_EK(Rtime: 13:5:36.205 Thrs: 1 S

```
Set comprehension: {34, 70, 8, 10, 12, 90}
Exception: ModuleNotFoundError: No module named 'qrcode_' at 0.3217
RemObjects Pascal Script. Copyright (c) 2004-2024 by RemObjects Software & maXbox5
Ver: 5.0.2.24 (502). Workdir: C:\maxbox\ipso\IBZ_Module1_4_2022\maxbox4Aarau
\maxbox5\maxbox522\maxbox5
```

I also activated the **`JITEnable`** variable, it controls when the just-in-time debugger is called. So for the reorganisation of the sources I have the latest revision with patches from `issue #202 (commit 86a057c)` but I am unable to compile the files at first (`Core_D27`) that are part of the **`PascalScript_Core_D27.dpk`** for that platform for **Linux64, Win64** nor **MacOS64**.

Here's some source output at first to show the internal exception handling for the **10.4 dccosx64** or **dcc64** compiler (similar results exist for **dcclinux64**):

```
procedure TPSExec.ExceptionProc(proc, Position: Cardinal; Ex: TPSError;
                                            const s: tbtString; NewObject: TObject);
var
  d, l: Longint;
  pp: TPSExceptionHandler;   //debcnt: integer
begin
  ExProc:= proc;
  ExPos:= Position;
  ExEx:= Ex;
  ExParam:= s;
  inc(debcnt);
  if maxform1.GetStatDebugCheck then
    maxform1.memo2.lines.add('debug: '+inttostr(debcnt)+'-'+s+'
           '+inttostr(proc)+' err:'+inttostr(ord(ex)));    //@fmain
  if ExObject <> nil then
    ExObject.Free;
  ExObject:= NewObject;
 //ShowMessage('We do not get this far: '+exparam);
  if Ex = eNoError then Exit;
   //maxform1.memo2.lines.add(s);
   // halt(1);
   // ShowMessage('We don't want not get this far');

  for d:= FExceptionStack.Count -1 downto 0 do begin
    pp:= FExceptionStack[d];
    if Cardinal(FStack.Count) > pp.StackSize then begin
      for l:= Longint(FStack.count) -1 downto Longint(pp.StackSize) do
        FStack.Pop;
    end;
```

Then I can see the **ExceptionProc** as a first chance exception at $0000000100419E9E.

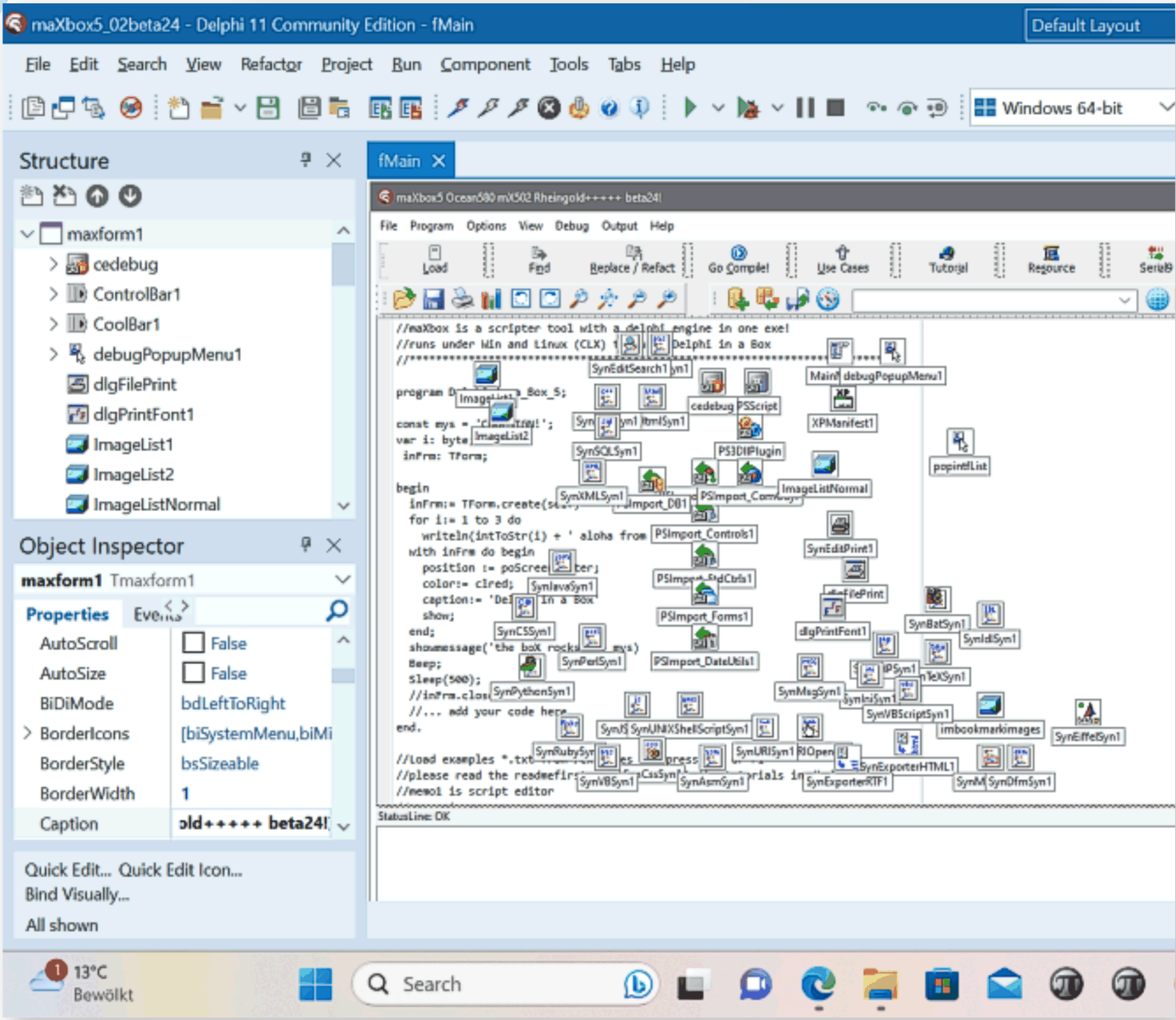
Exception class **EAccessViolation** with message
'Access violation at address 0000000100419E9E,
accessing address 00000009017241F8'.

```
Process TestApplication (5741)
Source Breakpoint at: C:\Program Files\Streaming\IBZ2021\Module2_
3\EKON26\maxbox4\pascalscript-master\pascalscript-master\Source\
uPSRuntime.pas line 2060. Process TestApplication (5741)

Upsruntime.TPSExec.Clear()(0x00000002017350d0)
Upsdebugger.TPSCustomDebugExec.Clear()(0x00000002017350d0)
Upscomponent.TPSScript.Compile()(0x0000000201734c20)
Fmain.TForm1.Compile1Click(System.TObject*)(0x00007ffeefbfe038)
Vcl.Menus.TMenuItem.Click()(0x0000000201734960)
Vcl.Menus.TMenu.DispatchCommand(unsigned short)(0x0000000201734340,2)
Vcl.Forms.TCustomForm.WMCommand(Winapi.Messages.
TWMCommand&)(0x0000000205039ff0,0x00007ffeefbfe878)
:000000010001132B System::Tobject::Dispatch(void*)
```

Another disturbing point by debugging was a redirect to debug.
Since I use `SynPdf.pas` I have to include `SynCommons.pas` in my project.
But it seems I got more than I wanted.
Sometimes when I debug in the IDE and try to dig into some routine, I get to the `Move procedure` from `SynCommons.pas` instead of the normal `system` call where I want to get!

The solution is easy: `FastCodem`, `Move` and `Fillchar` are included within `SynCommons.pas` optimized for speed and `SmartLinking` won't make it big in your `exe` and logging and all the other won't be part of it.
You can get rid of it, by commenting the corresponding lines in the initialization block of this unit.
Under new versions of the framework, you have a conditional setting to disable it.

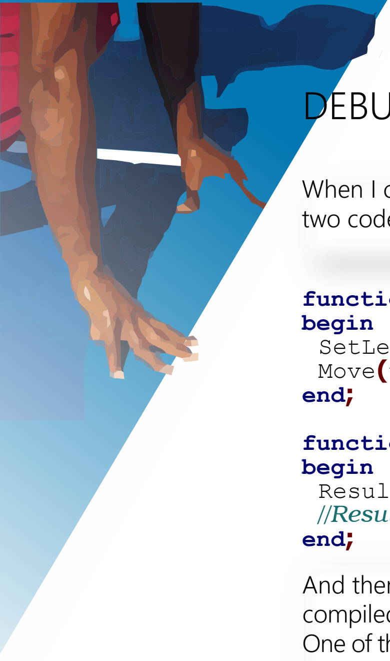
```
RedirectCode(GetAddressFromCall(@RecordCopyInvoke),@RecordCopy);
RedirectCode(GetAddressFromCall(@FillCharInvoke),@FillChar);
RedirectCode(GetAddressFromCall(@MoveInvoke),@Move);
```



```
//maXbox is a scripter tool with a delphi engine in one exe!
//runs under Win and Linux (CLX) t      Delphi in a Box
//****************************************************
program D          _Box_5;

const mys = 'C           !';
var i: byte
 inFrm: TForm;

begin
  inFrm:= TForm.create(s
  for i:= 1 to 3 do
    writeln(intToStr(i) + ' aloha from
  with inFrm do begin
    position := poScree     ter;
    color:= clred;
    caption:= 'Del      in a Box'
    show;
  end;
  showmessage('the box rocks       mys)
  Beep;
  Sleep(500);
  //inFrm.clos
  //... add your code here
end.

//Load examples *.txt           es    press
//please read the readmefir              rials in
//memo1 is script editor
```

StatusLine: OK

AutoScroll ☐ False
AutoSize ☐ False
BiDiMode bdLeftToRight
BorderIcons [biSystemMenu,biMi
BorderStyle bsSizeable
BorderWidth 1
Caption old+++++ beta24!



When I choose a second compile in a **CrossVCL** from the menu I always test also the debugger with two code functions as similarities but with different code solutions, for example **`GetBytes`**:

```
function GetBytes(value: string): TBytes;
begin
 SetLength(Result, SizeOf(value));
 Move(value, Result[0], SizeOf(value));
end;

function AnsiBytesOf(const S: string): Tbytes; //like GetBytes
begin
 Result := TEncoding.ANSI.GetBytes(S);
 //Result := GetBytes(S);
end;
```

And then maybe by intuition I made a build and the **AD** has gone away and I got my first screen, compiled and script executed:
One of the solved problems in the meantime is to catch an Access-violation instead of crashing the app. It's like the code in `uPSRuntime.pas` could not find the cause of halting or exiting like the following:

```
procedure TdynamicDll.Quit;
begin
   if not( csDesigning in ComponentState ) then begin
   {$IFDEF MSWINDOWS}
   MessageBox( GetActiveWindow, PChar(GetQuitMessage), 'Error',
                             MB_TASKMODAL or MB_ICONSTOP );
   ExitProcess( 1 );
   {$ELSE}
   WriteLn(ErrOutput, GetQuitMessage);
   Halt( 1 );
   {$ENDIF}
   end;
end;
```

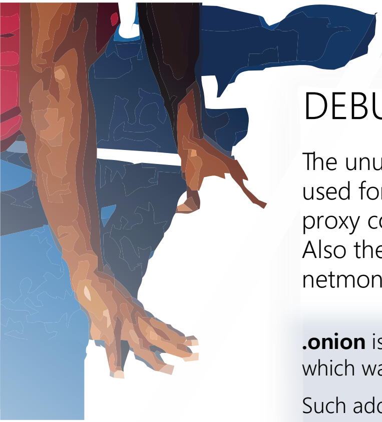
After debugging I realized that it is a `first chance exception` which works as long the debugger is running with break or continue but without debugger the app disappears without forwarding the **AV** on the output like **AV** at address xyz read of address 000.
You may know that the **`Application.OnException`** handler is only called for **unhandled** exceptions. An unhandled exception occurs when no **`try..except`** block has found the exception or where it has been caught and then re-raised!

Execution will continue in any outer except block. If there are none, or if any that may exist also re-raise the exception, then eventually the exception will reach the **`Application.OnException`** handler. In fact: an `Application.OnException` is your last chance to deal with an unhandled exception. It is not the first opportunity to respond to any exception.

Catching the exception and displaying the message is only helpful if you've already released the software to users and cannot reproduce the problem locally (*and even then, it's not as good as using a real exception library, like* **EurekaLog** *or* **MadExcept**). In this case, we already know which line caused the exception, which exception class was thrown, and what its message was. The suggested code does NOT add information to the investigation

So if you have a **network error** or **exception** in your web call from a **Rest-Client**, it could be wise to check the not-working-connection before you catch the possible exception, as seen in the next image.

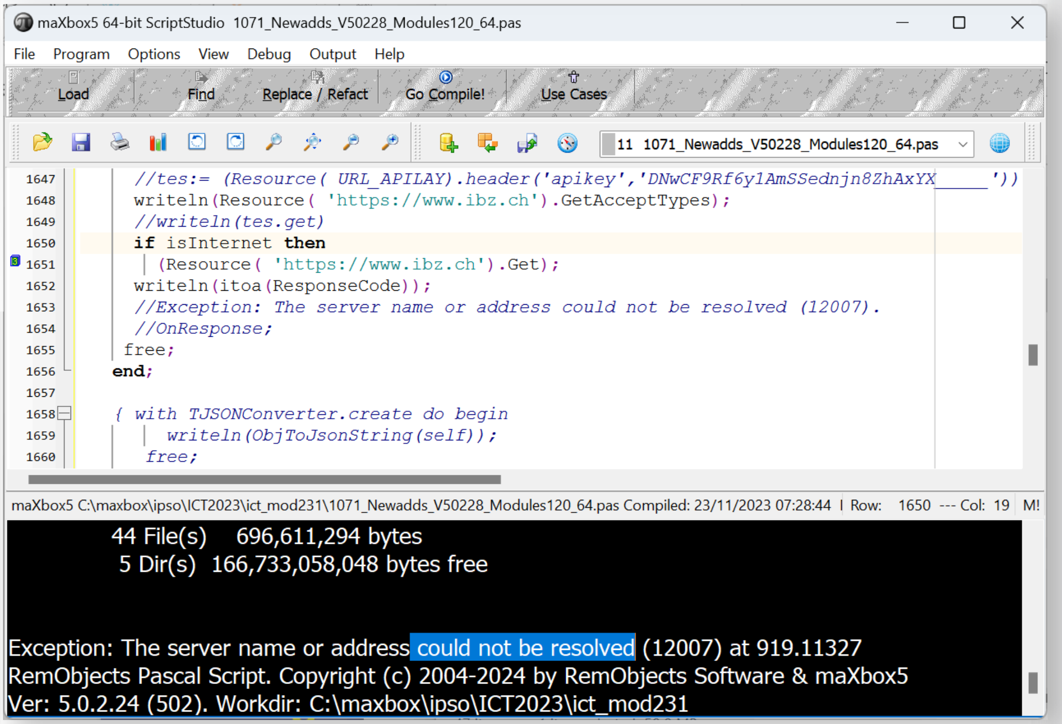
Hint to Error 12007: I am attempting to fetch the HTML source behind an onion site via the **WinINet API** but the **`InternetOpenUrl()`** returns the error code of **`12007`** which suggests there is an issue regarding the resolution of the web address.



The unusual part of this problem is that the error is not reproducible when a **non-onion site** * is used for the web link, which clearly means that there is no issue in the part concerning a possible proxy configurations.
Also the **GetLastError** function returns an error code like 12007. You can also check with netmon tool if an actual connection attempt has been made.

.onion is a special-use **top level domain name** designating an anonymous onion service, which was formerly known as a "hidden service", reachable via the Tor network.

Such addresses are not actual **DNS** names, and the **.onion TopLevelDomain** is not in the **Internet DN**S root
That explains the name:* **non onion site



maXbox5 64-bit ScriptStudio 1071_Newadds_V50228_Modules120_64.pas
File Program Options View Debug Output Help
Load Find Replace / Refact Go Compile! Use Cases
11 1071_Newadds_V50228_Modules120_64.pas

```
    //tes:= (Resource( URL_APILAY).header('apikey','DNwCF9Rf6y1AmSSednjn8ZhAxYX_____'))
    writeln(Resource( 'https://www.ibz.ch').GetAcceptTypes);
    //writeln(tes.get)
    if isInternet then
     | (Resource( 'https://www.ibz.ch').Get);
    writeln(itoa(ResponseCode));
    //Exception: The server name or address could not be resolved (12007).
    //OnResponse;
   free;
  end;

  { with TJSONConverter.create do begin
     | writeln(ObjToJsonString(self));
     free;
```

maXbox5 C:\maxbox\ipso\ICT2023\ict_mod231\1071_Newadds_V50228_Modules120_64.pas Compiled: 23/11/2023 07:28:44 | Row: 1650 --- Col: 19 M!

```
          44 File(s)    696,611,294 bytes
           5 Dir(s)  166,733,058,048 bytes free


Exception: The server name or address could not be resolved (12007) at 919.11327
RemObjects Pascal Script. Copyright (c) 2004-2024 by RemObjects Software & maXbox5
Ver: 5.0.2.24 (502). Workdir: C:\maxbox\ipso\ICT2023\ict_mod231
```

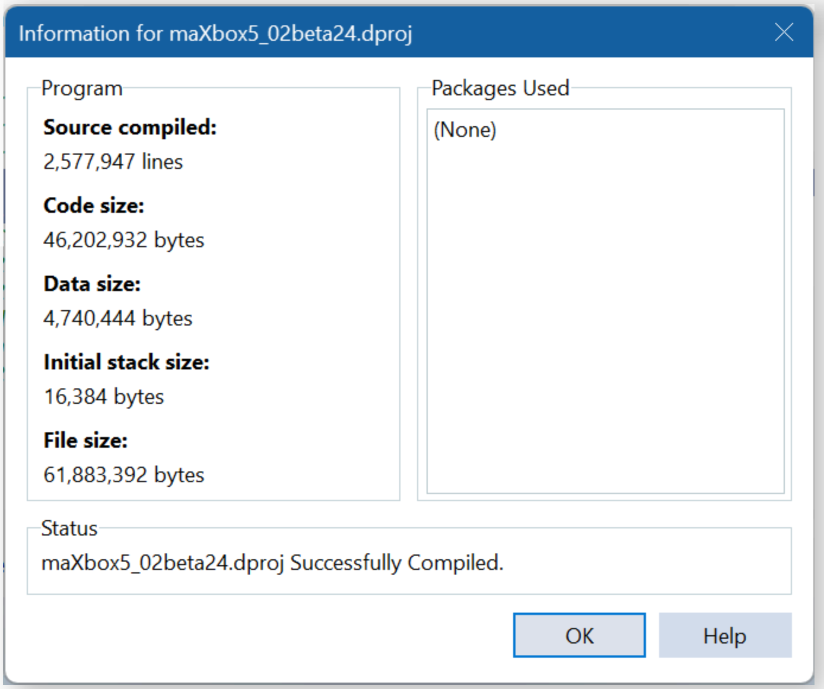
As we have seen you can tell the debugger to ignore certain kinds of exceptions.
Add an exception class to the list, and all exceptions of that type and of any descendant types will pass through your program without Delphi interfering.
You can use **Delphi's** "**advanced breakpoints**" to disable exception handling around a region of code. To begin, set a **breakpoint** on the line of code where you want the **IDE** to ignore exceptions or the call is an **API** from outside like the following User-Agent.

```
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36(KHTML, like Gecko) Chrome/88.0.4324.104 Safari/537.36'}
try:
    request_result = requests.get(url, headers=headers).json()
    print(request_result)
    print('[In English]:
         ' + request_result['alternative_translations'] [0]
         ['alternative'][0]  ['word_postproc'])
    print('[Language Dectected]: ' + request_result['src'])
except:
     pass
```

CONCLUSION:

Breakpoints do pause program-execution at a certain location or when a particular condition occurs. You can set **`source breakpoints`** and module load breakpoints in the **Code Editor** before and during a debugging session.

`Application.OnException` is your last chance to deal with an unhandled exception. Basically exceptions are thrown at the debugger first, then to the actual program where if it isn't handled it it will be repeated twice, offering a chance to do something with it in your IDE, before and after the application itself.



Information for maXbox5_02beta24.dproj

Program

Source compiled:
2,577,947 lines

Code size:
46,202,932 bytes

Data size:
4,740,444 bytes

Initial stack size:
16,384 bytes

File size:
61,883,392 bytes

Packages Used

(None)

Status

maXbox5_02beta24.dproj Successfully Compiled.

```
References:

Compiled Project:
https://github.com/maxkleiner/maXbox4/releases/download/
V4.2.4.80/maxbox5.zip

Docs and Tool: https://maxbox4.wordpress.com

maXbox herunterladen | heise Download
```

OVERVIEW OF PASCALSCRIPT FEATURE IN SYNCOVERY BY DMITRII V. KONNOV, TOBIAS GIESEN

ABSTRACT:

This article provides an overview of the new feature marked as **PascalScript**, implemented in the world's leading cloud file synchronization application called **Syncovery**

Keywords: Syncovery 10, cloud storage, Google Drive, Sharepoint, OneDrive, DropBox, PascalScript

.INTRODUCTION

Pascal Script is a scripting language **that builds upon the Pascal programming language**.
It **empowers automated runtime control** for scriptable applications and server software.
This scripting engine comprises a compiler and a byte code interpreter, offering a free and open-source solution.
Pascal Script is designed to be largely compatible with **Object Pascal**, making it partially interoperable with **Delphi, Free Pascal,** and **GNU Pascal**.

Originally created by **Carlo Kok** under the name **CajScript**, it was later re-branded as **Innerfuse Pascal Script** with version 2.23.
Subsequently, **RemObjects** took over the project, giving it the name **RemObjects Pascal Script**.
It was then made available as open-source software for integration into the **Delphi Integrated Development Environment (IDE).** As of version 2.07, Pascal Script was ported to Free Pascal.
Since 2017, Pascal Script has been integrated as a standard component within the Lazarus IDE, further expanding its reach and utility.

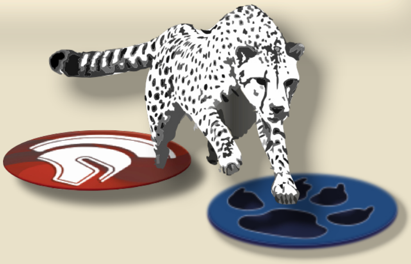
SYNCOVERY

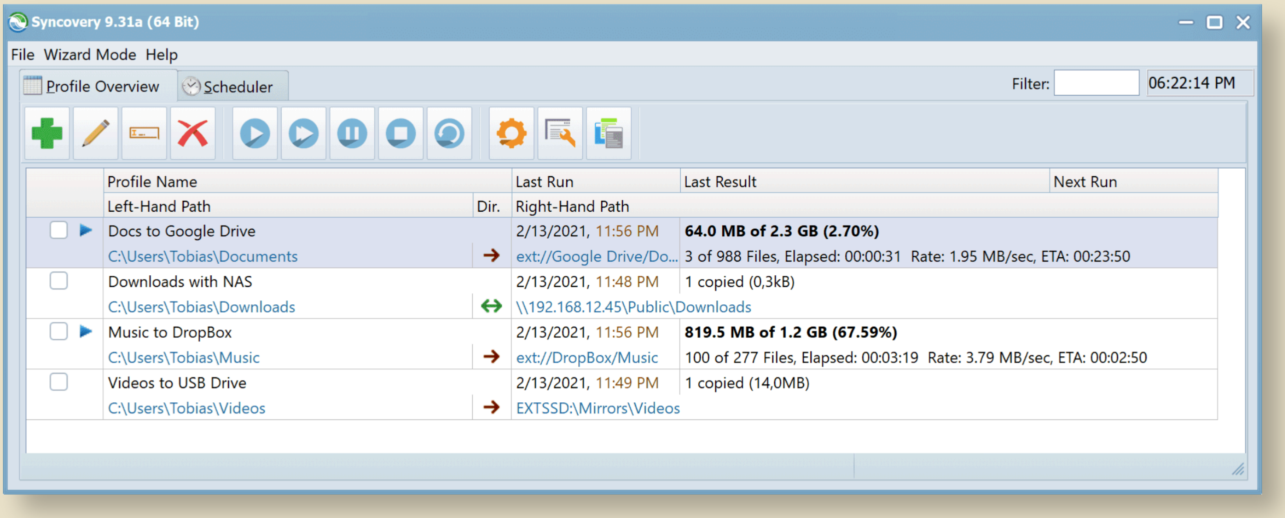
Among the software that widely uses **PascalScript**, **Syncovery** is a well-known backup and cloud file synchronization program.
Due to the growing number of **cloud storage** from providers of the **global cloud IT industry**, such as **Amazon S3, Google Drive, Microsoft Azure, OneDrive, SharePoint, DropBox, Box, and Backblaze B2**, the integration of cloud data storage is becoming one of the most popular tasks of a modern user.

Each cloud storage has its specifics, its user interaction interface, and its API.
From the user's point of view, the ideal is a file synchronization software that takes over the function of backing up files and, as a completely self-contained module, synchronizes local content with the cloud in the background.
In this case, the distinction between local disk and cloud is blurred.
Local smoothly flows into the Cloud and vice versa.

The user gets the opportunity not to worry about the safety of data, assigning all the routines to automated processes and devices that deal with it.

PascalScript is called in Syncovery to customize specific user requirements of file synchronization tasks.
The application provides a desktop GUI on Windows and macOS, whereas the Linux edition includes a web-based user interface that is accessed using a browser.
Syncovery is primarily based on the ease of understanding the concept of data synchronization. To do this, it was proposed to provide the user with an intuitive graphical interface in the form of Left - Right, as shown in Fig.1.





Profile Name / Left-Hand Path	Dir.	Last Run / Right-Hand Path	Last Result	Next Run
Docs to Google Drive C:\Users\Tobias\Documents	→	2/13/2021, 11:56 PM ext://Google Drive/Do...	**64.0 MB of 2.3 GB (2.70%)** 3 of 988 Files, Elapsed: 00:00:31 Rate: 1.95 MB/sec, ETA: 00:23:50	
Downloads with NAS C:\Users\Tobias\Downloads	↔	2/13/2021, 11:48 PM \\192.168.12.45\Public\Downloads	1 copied (0,3kB)	
Music to DropBox C:\Users\Tobias\Music	→	2/13/2021, 11:56 PM ext://DropBox/Music	**819.5 MB of 1.2 GB (67.59%)** 100 of 277 Files, Elapsed: 00:03:19 Rate: 3.79 MB/sec, ETA: 00:02:50	
Videos to USB Drive C:\Users\Tobias\Videos	→	2/13/2021, 11:49 PM EXTSSD:\Mirrors\Videos	1 copied (14,0MB)	

Figure 1.
Syncovery User Interface

The “Left path” and “Right path” - are the locations of the files, which can be either a local directory or a cloud directory. The user configures the synchronization points (*folders*) of his files, and the built-in Scheduler starts synchronization tasks.
Flexibility for synchronization tasks can be achieved not only through various settings in **Syncovery GUI** but also with **PascalScript**.

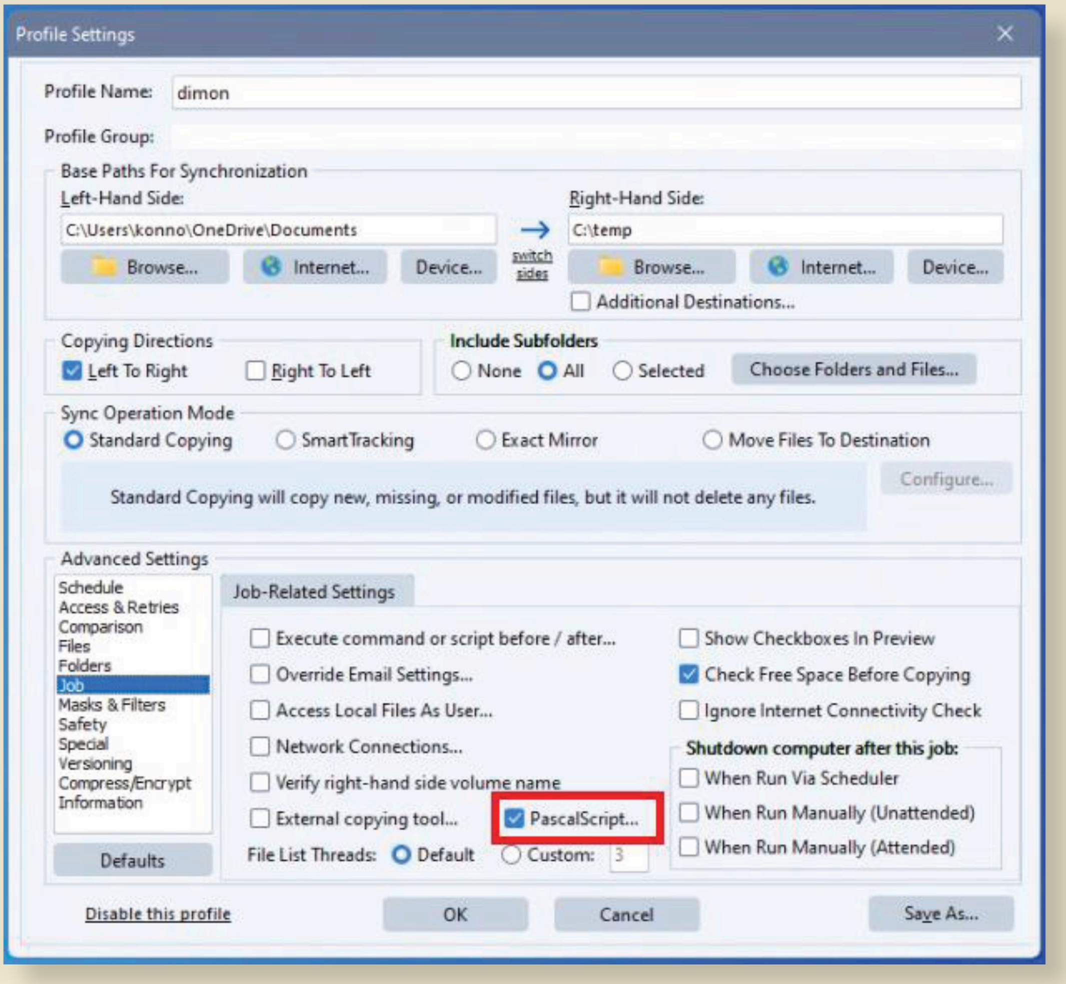
PASCALSCRIPT

Syncovery includes a PascalScript engine, allowing you to customize your profile’s behavior in many ways. **PascalScript** includes the following features:

- **`Variables, Constants`**
- **`Standard language constructs:`**
- **`Begin/End`**
- **`If/Then/Else`**
- **`For/To/Downto/Do`**
- **`Case x Of`**
- **`Repeat/Until`**
- **`While`**
- **`Uses`**
- **`Exit`**
- **`Continue`**
- **`Break`**
- **`Functions inside the script`**

There are many standard functions such as `Pos, Copy, Length, Ord, GetProfileProperty, SetProfileProperty, SaveProfileSettings, ConcatPath, ExtractFileName, ExtractFilePath, ExtractURLPartAfterServer, ExtractFileExt, ChangeFileExt, FileExists, FileExistsMatching, EntryExists, FileAge, FileCopy, FileDelete, FileRename,`
and many others you can get familiar with at
`https://www.syncovery.com/pascalscript/`

PascalScript was added in **Syncovery 8**, and many new hooks and functions have been added since the first release. Let us see how to start doing custom programming in PascalScript. Double-click on a profile to open the Profile Settings dialog (see Fig. 2).



Profile Settings

Profile Name: dimon

Profile Group:

Base Paths For Synchronization

Left-Hand Side: C:\Users\konno\OneDrive\Documents

Right-Hand Side: C:\temp

□ Additional Destinations...

Copying Directions: ☑ Left To Right □ Right To Left

Include Subfolders: ○ None ● All ○ Selected

Sync Operation Mode: ● Standard Copying ○ SmartTracking ○ Exact Mirror ○ Move Files To Destination

Standard Copying will copy new, missing, or modified files, but it will not delete any files.

Advanced Settings: Schedule, Access & Retries, Comparison, Files, Folders, Job, Masks & Filters, Safety, Special, Versioning, Compress/Encrypt, Information

Job-Related Settings

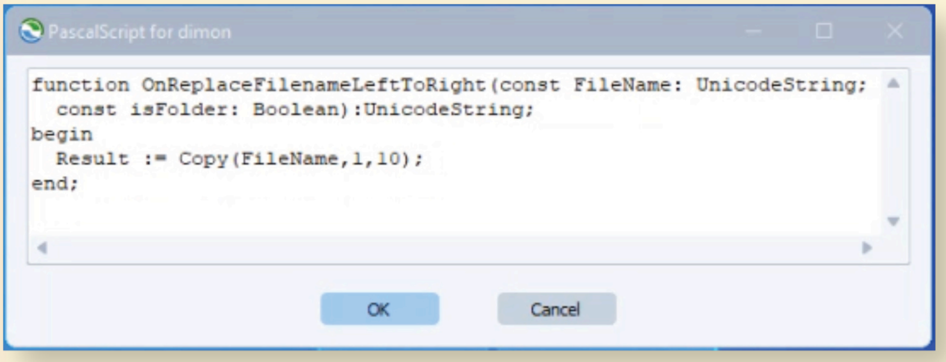
□ Execute command or script before / after...
□ Override Email Settings...
□ Access Local Files As User...
□ Network Connections...
□ Verify right-hand side volume name
□ External copying tool...
☑ PascalScript...
□ Show Checkboxes In Preview
☑ Check Free Space Before Copying
□ Ignore Internet Connectivity Check

Shutdown computer after this job:
□ When Run Via Scheduler
□ When Run Manually (Unattended)
□ When Run Manually (Attended)

File List Threads: ● Default ○ Custom: 3

Figure 2. Profile Settings dialog

Then selecting the 'Job' item in the **Advanced Settings** list and double-clicking on the **'PascalScript'** checkbox will bring the **PascalScript** editor. To change a particular behavior, you need to write a hook function and write some code.



PascalScript for dimon

```
function OnReplaceFilenameLeftToRight(const FileName: UnicodeString;
  const isFolder: Boolean):UnicodeString;
begin
  Result := Copy(FileName,1,10);
end;
```

Figure 3. PascalScript editor

This window allows you to edit a script associated with the **Syncovery's** profile.
Every function handler declared in the editor will be automatically hooked with **Syncovery's** event.
In the given script *(see Fig.3 Page 4 of this article*) we can see the function `OnReplaceFilenameLeftToRight` returning a new file name and we can shorten it for example if the FileName length is more than 10.
But this is just a simple example of **PascalScript** abilities and every customer's task may involve much more business logic.
In most cases, our tech support will write the code for you.
The power of **PascalScript** in **Syncovery** was regularly enhanced as new hooks and support functions were added to the language.
Some typical use cases are outlined here.

- **custom file renames:**

perhaps the most frequently used PascalScript hook in Syncovery is `OnReplaceFilenameLeftToRight.` It allows to specify or derive a new filename, either completely freely, or based on the original file name. Sample scripts are available to shorten long file names, or to convert disallowed characters.

- **custom filters:**

using the `OnIncludeItem` hook, a customer can use code to implement unusual filters. For example, a script can be used to enforce specific naming rules and discard any files that do not match. There are example scripts to exclude files without filename extension, as well as processing only subfolders that contain the file READY.to process.

- **sorting files into subfolders:**

another frequent use case for a PascalScript is the need to sort files into additional subfolders that do not exist on the source side. The `OnBeforeFileCopy` hook allows to change the destination path based on Pascal code. An example script from the Syncovery website will manipulate the destination paths for copying (left to right), by adding the subfolder 'archive' which is not present on the source side. Other customers need files to be sorted into subfolders based on the date, which is also possible.

- **custom schedules:**

the `OnGetNextRunTime` hook can be used to create custom rules for scheduling a job.
The best way to use it is to give the profile a regular, simple schedule, and use the hook to skip undesired run times.
An example from the **PascalScript** documentation is intended for a profile that is scheduled to run “every day at XX: YY". The hook ensures that it runs only on the *first weekday in a month*.

The documentation for PascalScript in Syncovery, along with many sample scripts, can be found on this page:
https://www.syncovery.com/pascalscript/

CONCLUSION

Soon **Syncovery** will be 20 years old. Over the years, **Syncovery** *2 has become an essential tool for system administrators.

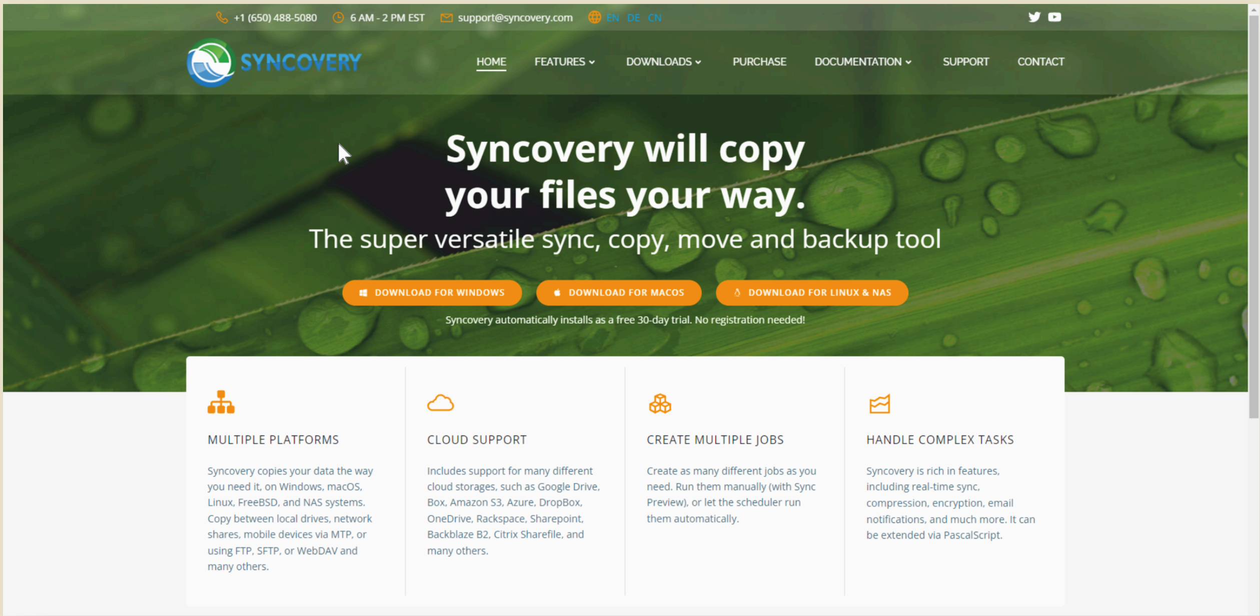
The first version of the application (*written in Delphi*) was released under the name **Super Flexible File Synchronizer in 2003**.
The project is a **German STEM Startup** that eventually grew into a working product. In 2012, the product was renamed Syncovery.

In 2021, at the **Delphi Showcase Challenge**, dedicated to the 26th anniversary of the Delphi programming language, the **Syncovery** tool was **awarded the fourth prize for excellent desktop synchronization** software that demonstrates the flexibility of the Delphi development environment *3. Although the **Delphi** compiler supports compilation for **Windows**, **Linux**, and **Mac**, it is **only used to compile Syncovery for Windows.**

Binaries for additional platforms are created using the open-source Free Pascal compiler.

Over the 20 years of its development, **Syncovery** has shown unmatched user loyalty and stable and sustainable product development. New features are constantly being added to the program following modern user requests. **We look forward to further cooperation with Syncovery users.**

REFERENCES
***1 RemObjects PascalScript**: `https://github.com/remobjects/pascalscript`
***2 Superflexible AG** `https://www.syncovery.com`
***3 Astounding Desktop Synchronization Software Displays Delphi Flexibility** // Embarcadero `https://blogs.embarcadero.com/astounding-desktop-synchronization-software-displays-delphi-flexibility/`



Syncovery will copy your files your way.

The super versatile sync, copy, move and backup tool

Syncovery automatically installs as a free 30-day trial. No registration needed!

MULTIPLE PLATFORMS

Syncovery copies your data the way you need it, on Windows, macOS, Linux, FreeBSD, and NAS systems. Copy between local drives, network shares, mobile devices via MTP, or using FTP, SFTP, or WebDAV and many others.

CLOUD SUPPORT

Includes support for many different cloud storages, such as Google Drive, Box, Amazon S3, Azure, DropBox, OneDrive, Rackspace, Sharepoint, Backblaze B2, Citrix Sharefile, and many others.

CREATE MULTIPLE JOBS

Create as many different jobs as you need. Run them manually (with Sync Preview), or let the scheduler run them automatically.

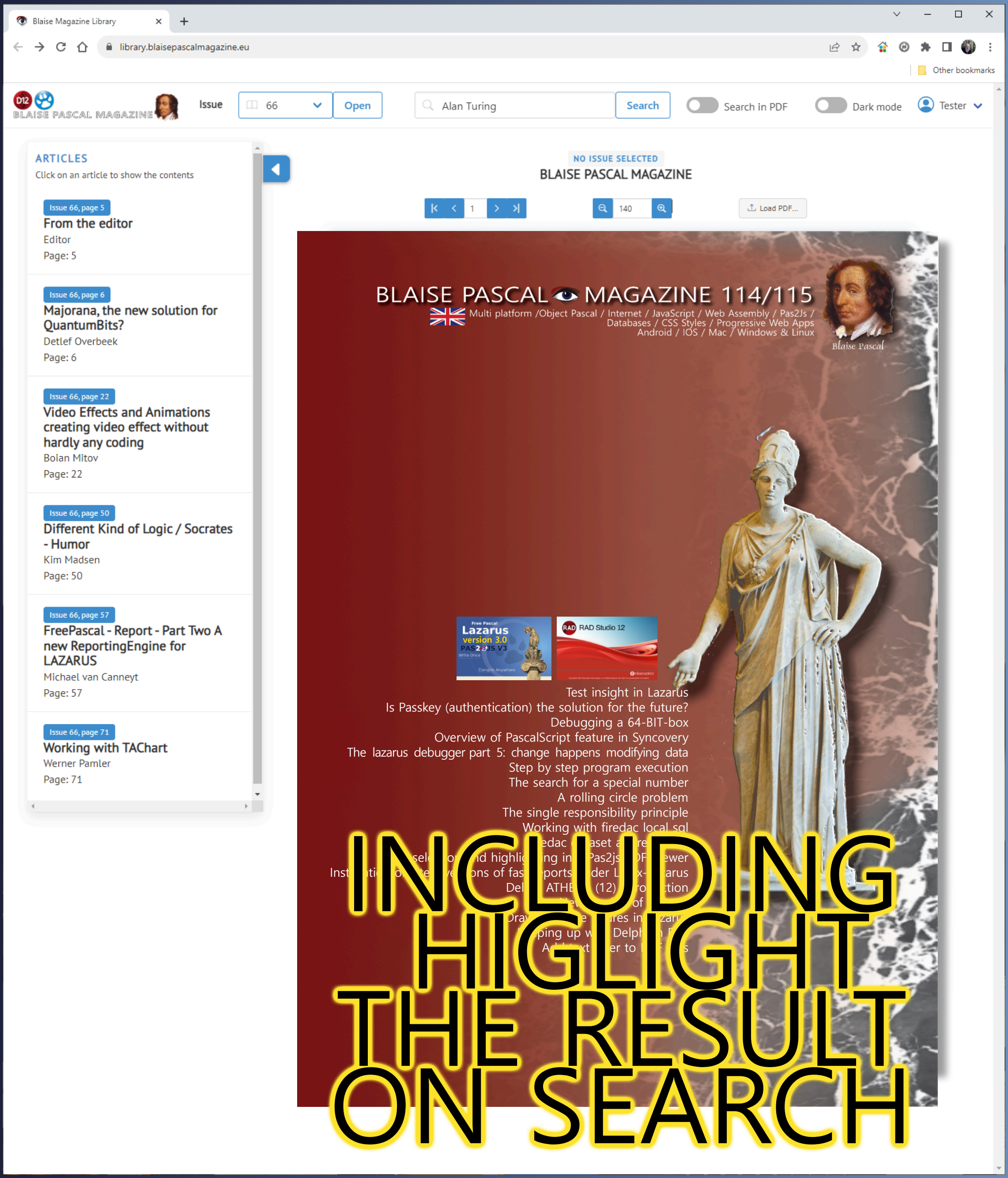
HANDLE COMPLEX TASKS

Syncovery is rich in features, including real-time sync, compression, encryption, email notifications, and much more. It can be extended via PascalScript.

THE SUBSCRIPTION FOR BLAISE PASCAL MAGAZINE

1. SUBSCRIPTION: PER YEAR
   issues starting at the latest issue available +1 year / code included + downloadable
   € 70,00 or without Vat 64,22.
   including the **FREE SEARCH ENGINE** internet library for all magazines

2 LIB-STICK USB-CARD all issues / code included, interface as the **SEARCH ENGINE**.
  € 100,00 **INCLUDING** 1 year subscription



ARTICLES

Click on an article to show the contents

Issue 66, page 5
From the editor
Editor
Page: 5

Issue 66, page 6
Majorana, the new solution for QuantumBits?
Detlef Overbeek
Page: 6

Issue 66, page 22
Video Effects and Animations creating video effect without hardly any coding
Boian Mitov
Page: 22

Issue 66, page 50
Different Kind of Logic / Socrates - Humor
Kim Madsen
Page: 50

Issue 66, page 57
FreePascal - Report - Part Two A new ReportingEngine for LAZARUS
Michael van Canneyt
Page: 57

Issue 66, page 71
Working with TAChart
Werner Pamler
Page: 71

BLAISE PASCAL MAGAZINE 114/115

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

Blaise Pascal

Free Pascal Lazarus version 3.0 PAS2JS V3 Write Once Compile Anywhere

RAD Studio 12

Test insight in Lazarus
Is Passkey (authentication) the solution for the future?
Debugging a 64-BIT-box
Overview of PascalScript feature in Syncovery
The lazarus debugger part 5: change happens modifying data
Step by step program execution
The search for a special number
A rolling circle problem
The single responsibility principle
Working with firedac local sql

INCLUDING HIGLIGHT THE RESULT ON SEARCH

USE WHERE EVER THE INTERNET IS

https://www.blaisepascalmagazine.eu/register/

THE LAZARUS DEBUGGER PART 5: CHANGE HAPPENS – MODIFYING DATA

BY MARTIN FRIEBE

Starter Expert

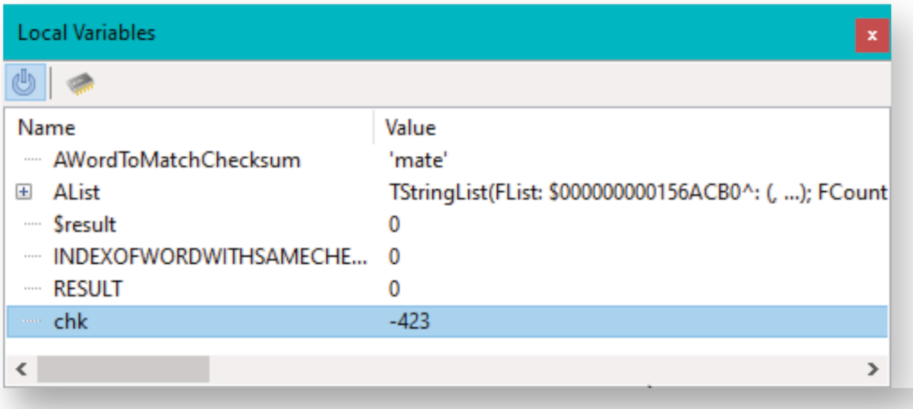
WHAT IS, AND WHAT SHOULD BE...

In all of the previous articles, whenever we had found the bug and fixed it,
we would recompile and restart the app to see that it worked.
In our sample apps that was no problem.
The app would run the code we fixed and we would be able see right away if the new code worked.

But what if the app would take a long time to reach the code we fixed?
What if we had to interact with it at length before we would know if the fix was any good.
What if we weren't even sure about the fix, but just wanted to test how it would work had it gotten a different value for one of the variables?
Then restarting, and waiting for the code to be executed again can be very bothersome.
Depending on what the issue is, we can keep the app running, yet test what would be if things were different.
This doesn't work for all sort of changes. It's limited to changes in the apps data.
We will explore this on an example.
Our example is quite simple. We will just pretend it would be a pain to re-run it.
It searches a list for a word with the same words as a given word.

```
program project1;

uses Classes;

function Checksum(s: string): integer;
var
  i: Integer;
begin
  Result := 0;
  for i := 1 to Length(s) do
    Result := Result + ord(s[i]);
end;

function IndexOfWordWithSameChecksum(AWordToMatchChecksum: String;
                                     AList: TStringList): Integer;
var
  chk: Integer;
begin
  if AList = nil then
    exit(-1);
  chk := -Checksum(AWordToMatchChecksum);
  Result := AList.Count - 1;
  while Result >= 0 do begin
    if Checksum(AList[Result]) = chk then
      exit;
    dec(Result);
  end;
end;

var
  Words: TStringList;
  i: Integer;
begin
  Words := TStringList.Create;
  Words.Add('words');
  Words.Add('tame');
  Words.Add('town');

  i := IndexOfWordWithSameChecksum('mate', Words);
  writeln(i);
end.
```

As the checksum is the sum of ordinal values of each letter, words with the same letters in any order are words with the same checksum. In the example we expect "**mate**" to be matched by "**tame**".
So it should print "**1**".
Yet it prints "**-1**", indicating it hasn't found anything.
As usual, we start debugging and `break` at line 18, the first line in "`IndexOfWordWithSameChecksum`".
There we can use the local window, to have a look at the variables involved.
Next, we step over the assignment of the checksum to "**chk**":

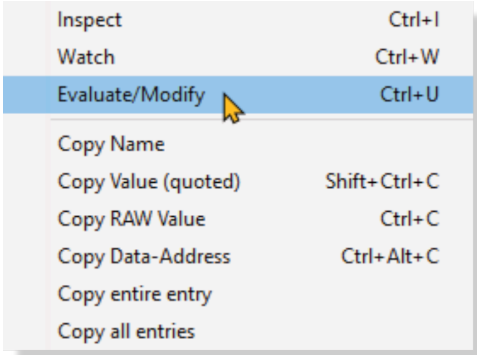


Local Variables

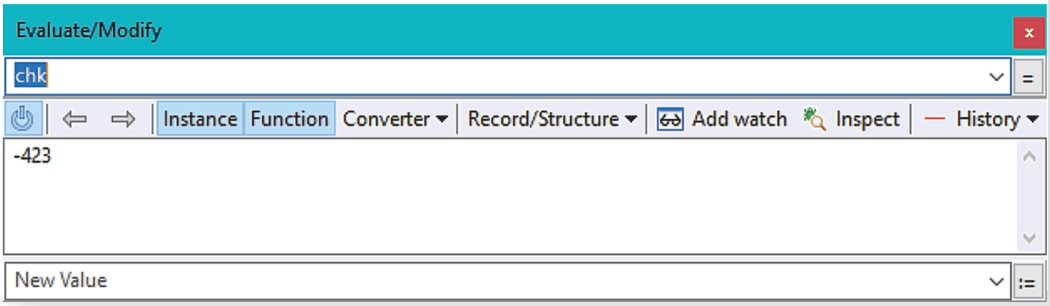
Name	Value
AWordToMatchChecksum	'mate'
⊞ AList	TStringList(FList: $000000000156ACB0^: (, ...); FCount
$result	0
INDEXOFWORDWITHSAMECHE...	0
RESULT	0
chk	-423

And we note that "**chk**" has a value of **-423**.
But a negative value is not what we expect, we summed up the ordinal values of all chars in "**mate**" and they are all positive integers. On closer inspection of the source code we notice that a typo has made it into our code. There is a unary minus in front of the call to "**Checksum**". The actual checksum in "**chk**" should be **423**.

Now, we could fix that by changing the code and restarting the app. Or we could change the value in the running application, and continue the current run with the correct value.
From the popup menu of the locals window we choose "**Evaluate/Modify**" on the "**chk**" variable:



This will open the corresponding window.

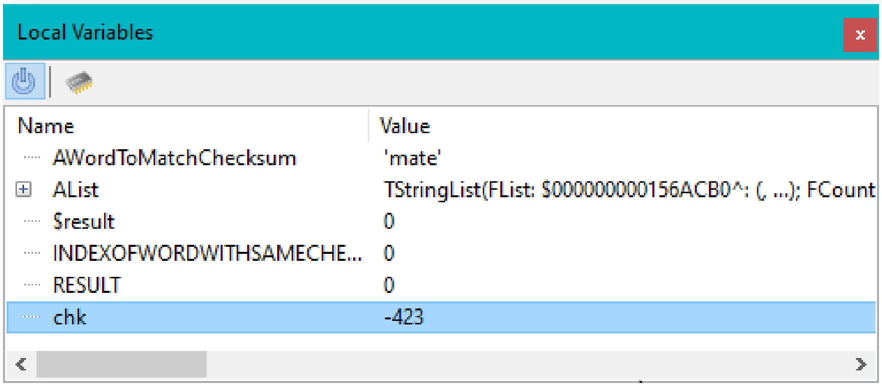


chk

-423

New Value

This window offers yet another way to evaluate and view values. Yet, for now we are going to focus on the "**Modify**" feature. The variable we want to change is already filled in and its value shown.
At the bottom is an edit to enter a "**New Value**" and a button "`:=`" to apply it.
We enter the positive **423** into the edit and press the button. Immediately, we can see the value being updated. Both the "**Evaluate/Modify**" and the locals window now show **423** for the variable "**chk**".
With that done, we continue our app to the end of "`IndexOfWordWithSameChecksum`" where we check the value of "**Result**" in the locals window.



Local Variables

Name	Value
AWordToMatchChecksum	'mate'
⊞ AList	TStringList(FList: $0000000000156ACB0^: (...); FCount
$result	0
INDEXOFWORDWITHSAMECHE...	0
RESULT	0
chk	-423

And there we can confirm that this time the `Result` has the correct value of **"1"**, which will also be printed once we let the app run to its end.
Changing the value without having to restart the application did save us some time.
However, once the error has been confirmed and we know how to amend the code,
we will eventually still have to recompile.

Otherwise, the value will go wrong each time the function is entered, and we would have to modify the value over and over again.
We might set a breakpoint, and do that a couple of times, but eventually manually changing the variable would loose us the advantage over recompiling and restarting.
On the other hand, if the change had not fixed the problem, we might have had another go to further investigate when the function would have been called again.
In the end, each case is different and the benefits need to be checked for each debug session.

WHEN CHANGE IS NOT AN OPTION

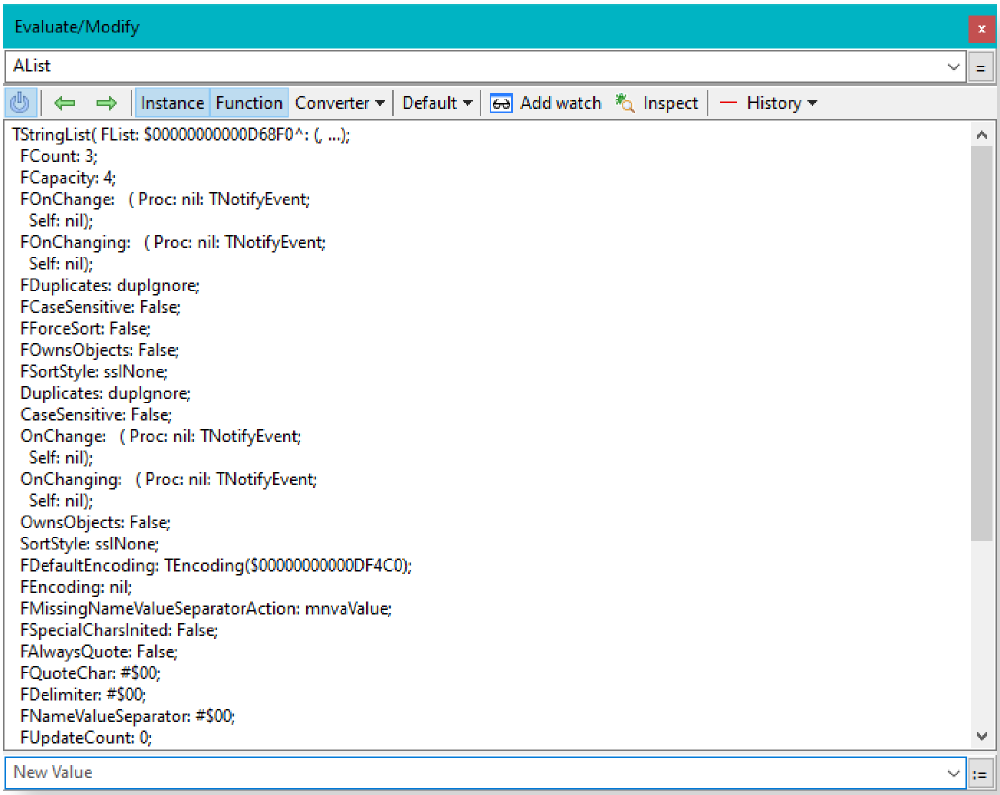
The "**Modify**" feature has some limitations. It currently only works for ordinal types (`numbers, enum, boolean,` ...). For structures and arrays, individual fields can be changed.
Unfortunately, strings can not yet be modified.

THE "EVALUATION" PART

The full name of the Window is "**Evaluate/Modify**". We have looked at the "**Modify**" part, and we will now explore the "**Evaluate**" part.

In short, the Evaluate window acts like a window for a single watch, showing the result in the same manner as the watches window does in the "**detail pane**".
The main advantage is that all the settings from the "**watches properties**" dialog are directly accessible, so it is much easier to change them or the expression to evaluate.
And using nearly the entire window to show the result, it is much easier to read it
than in the "detail pane". This is especially useful for structures, but can also help with **memory dumps**, `arrays` or `long strings`.

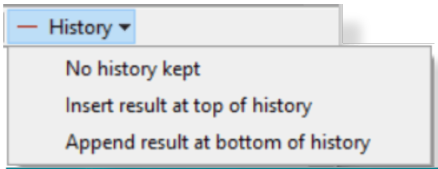
Evaluating “**AList**” will show the following.



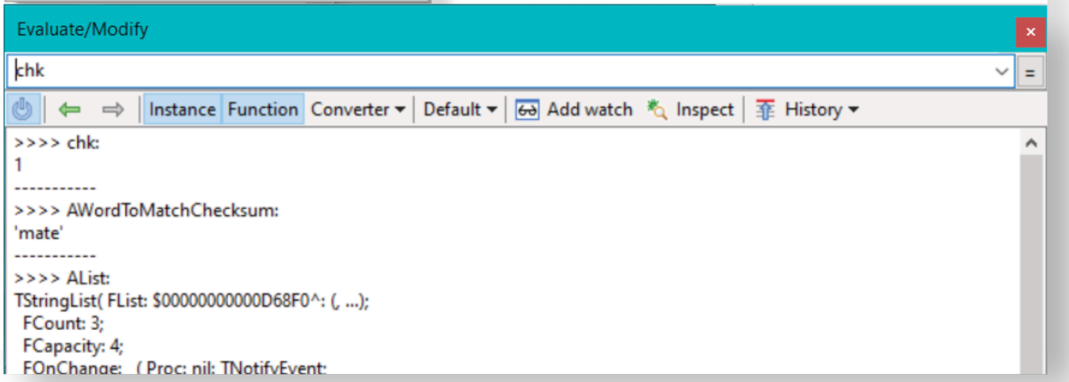
```
TStringList( FList: $00000000000D68F0^: ( ...);
  FCount: 3;
  FCapacity: 4;
  FOnChange:   ( Proc: nil: TNotifyEvent;
    Self: nil);
  FOnChanging:   ( Proc: nil: TNotifyEvent;
    Self: nil);
  FDuplicates: dupIgnore;
  FCaseSensitive: False;
  FForceSort: False;
  FOwnsObjects: False;
  FSortStyle: sslNone;
  Duplicates: dupIgnore;
  CaseSensitive: False;
  OnChange:   ( Proc: nil: TNotifyEvent;
    Self: nil);
  OnChanging:   ( Proc: nil: TNotifyEvent;
    Self: nil);
  OwnsObjects: False;
  SortStyle: sslNone;
  FDefaultEncoding: TEncoding($00000000000DF4C0);
  FEncoding: nil;
  FMissingNameValueSeparatorAction: mnvaValue;
  FSpecialCharsInited: False;
  FAlwaysQuote: False;
  FQuoteChar: #$00;
  FDelimiter: #$00;
  FNameValueSeparator: #$00;
  FUpdateCount: 0;
```

All fields are easily visible, and even nested values like in events (e.g. “**OnChange**”) are inlined. The toolbar provides the same properties as the watches properties dialog which was explained in the last article.

While you can **only see one variable at a time**, you can go back and forth between the values you have evaluated using the **green arrow buttons**. You can also select previous values from the drop-down. And if you need, you can keep previous values visible, and have new results inserted on top or at the bottom of them. The “**History**” tool-button provides the following options:


History
No history kept
Insert result at top of history
Append result at bottom of history

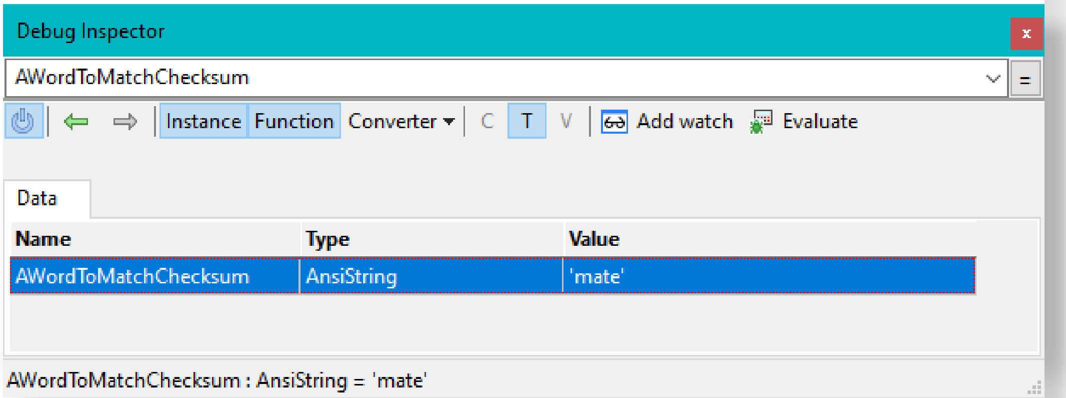
Giving the following view:



```
>>>> chk:
1
-----------
>>>> AWordToMatchChecksum:
'mate'
-----------
>>>> AList:
TStringList( FList: $00000000000D68F0^: ( ...);
  FCount: 3;
  FCapacity: 4;
  FOnChange:   ( Proc: nil: TNotifyEvent;
```

THE DEBUG INSPECTOR

While the topic of this article is the **Evaluate/Modify** Window, we will extend it a bit to look at one other window for viewing data from the application: The **"Debug Inspector"**.
Similar to the **Evaluate** window, it allows to view one watch at a time.
For simple values it does not offer much more than any of the other ways to view watches, except that it includes the type-name of the data.



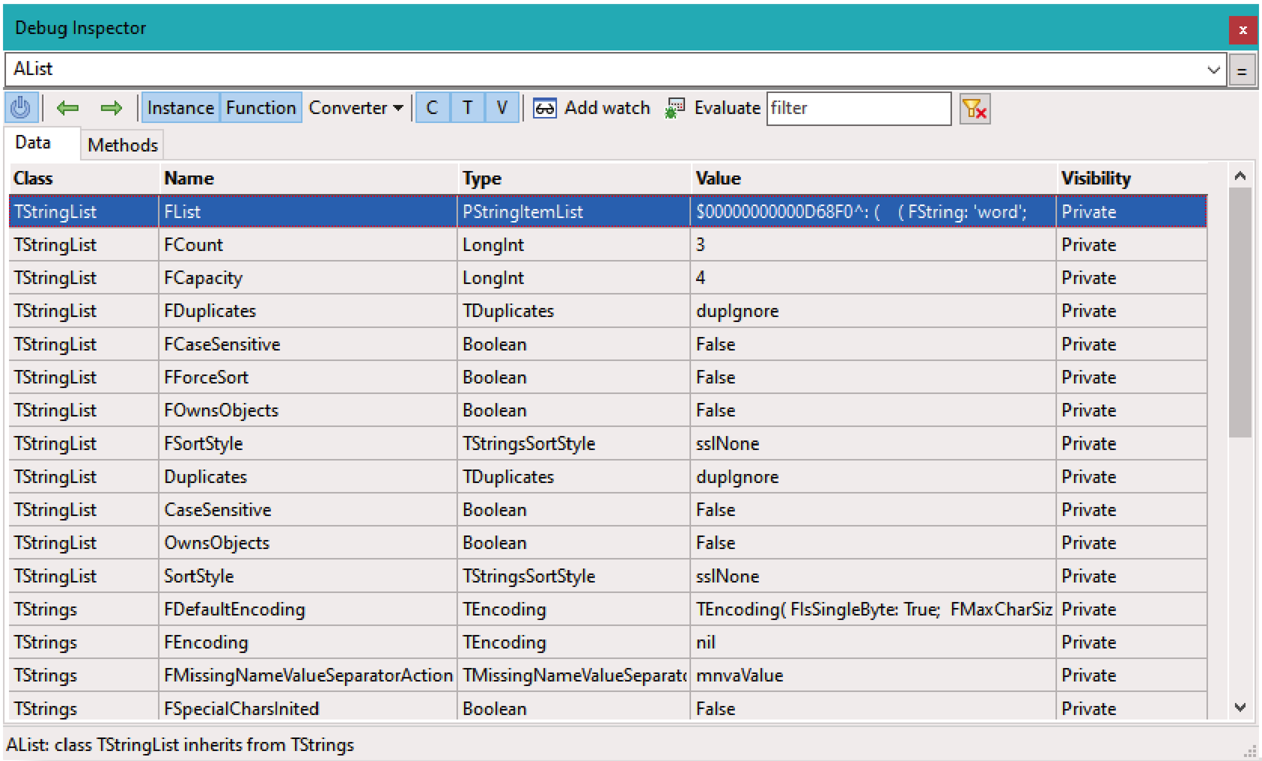
Debug Inspector

AWordToMatchChecksum

Name	Type	Value
AWordToMatchChecksum	AnsiString	'mate'

AWordToMatchChecksum : AnsiString = 'mate'

It plays its full strength for structured types and arrays. It lists each field or entry as a separate row.



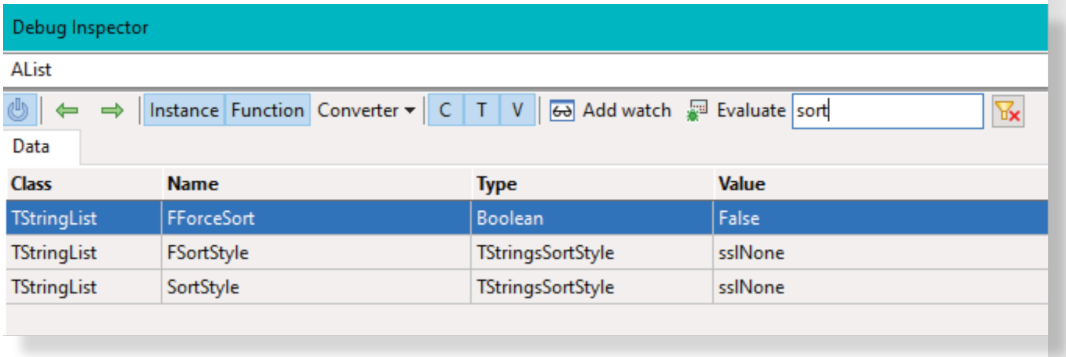
Debug Inspector

AList

Class	Name	Type	Value	Visibility
TStringList	FList	PStringItemList	$00000000000D68F0^: ((FString: 'word';	Private
TStringList	FCount	LongInt	3	Private
TStringList	FCapacity	LongInt	4	Private
TStringList	FDuplicates	TDuplicates	dupIgnore	Private
TStringList	FCaseSensitive	Boolean	False	Private
TStringList	FForceSort	Boolean	False	Private
TStringList	FOwnsObjects	Boolean	False	Private
TStringList	FSortStyle	TStringsSortStyle	sslNone	Private
TStringList	Duplicates	TDuplicates	dupIgnore	Private
TStringList	CaseSensitive	Boolean	False	Private
TStringList	OwnsObjects	Boolean	False	Private
TStringList	SortStyle	TStringsSortStyle	sslNone	Private
TStrings	FDefaultEncoding	TEncoding	TEncoding(FIsSingleByte: True; FMaxCharSiz	Private
TStrings	FEncoding	TEncoding	nil	Private
TStrings	FMissingNameValueSeparatorAction	TMissingNameValueSeparat	mnvaValue	Private
TStrings	FSpecialCharsInited	Boolean	False	Private

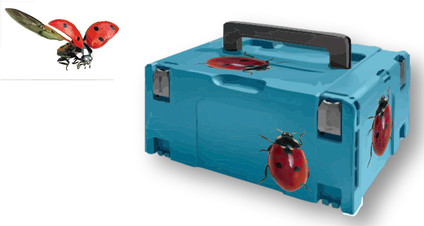
AList: class TStringList inherits from TStrings

Rows can be double-clicked to inspect the value of a field or entry for `arrays`. This also works for dereferencing `pointer` values. Together with the **forward/backward navigation** the data can be easily explored, and any inspected **value** that is of more permanent interest can be added as **watch** at the click of a button.
On top of that the inspector offers a filter to search the data.
This filter is only available for structured values and arrays.
Any value entered will be matched against all visible columns.
So one can search for fields containing some text in the data.
But one can also search for a field by name.
If in "**AList**" we want to find sorting related fields we can enter "**sort**" in the filter:



Class	Name	Type	Value
TStringList	FForceSort	Boolean	False
TStringList	FSortStyle	TStringsSortStyle	sslNone
TStringList	SortStyle	TStringsSortStyle	sslNone

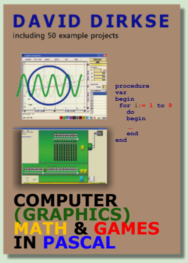
If it is the "**FSortStyle**" that we are after, we can then press `cursor-down` to navigate to it, and **Ctrl-Enter** to select it and change the inspected expression to `Alist.FSortStyle`.
This allows for very quick navigation even in objects with lots of fields.
In `arrays` this can also match the `index`, however only within the list of `indexes` on the current page (*arrays are paginated like in the watches window*)



Blaise Pascal

STEP BY STEP PROGRAM EXECUTION

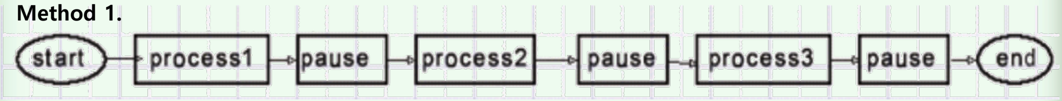
BY DAVID DIRKSE



ABSTRACT

For educational software it is explorative to show the individual steps toward a solution. This article describes two ways to accomplish this. In both cases a process is made of process1, process2 and process3 which may be executed step by step.

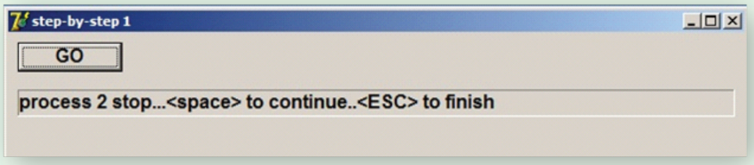
Method 1.



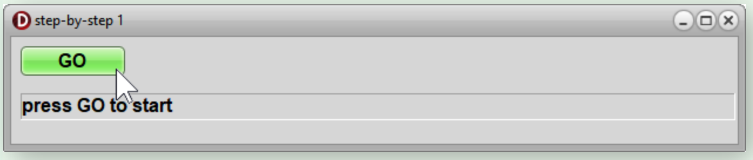
The pause procedure sets a `stopflag (Boolean)` and calls application.processmessages until the stopflag clears. This clearing is done by pressing the space bar.

When step by step operation is no longer wanted, pressing the escape key finishes the complete process without pauses. This is controlled by the `abortflag (Boolean)`. The pause procedure checks the abort flag and if set, exits immediately. Pressing the escape key sets the abort flag and clears the stopflag.

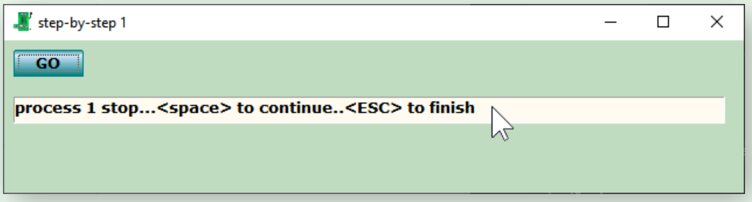
```
const  msg1 = 'press GO to start';
       msg2 = '<space> to continue..<ESC> to finish';
var stopflag, abortflag : boolean;

procedure pause;
begin
  if abortFlag = false then
  begin
    stopFlag := true;
    form1.msglabel.Caption := form1.msglabel.caption + '...' + msg2;
    while stopFlag do application.processmessages;
  end;
end;

procedure process1;  //similar for process2,3
begin
  form1.msglabel.caption := 'process 1 stop';
end;

procedure TForm1.startBtnClick(Sender: TObject);
begin
  activecontrol := nil;
  abortFlag := false;
  process1;
  pause;
  process2;
  pause;
  process3;
  pause;
  msglabel.Caption := 'finished';
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;Shift: TShiftState);
begin
  case key of
    VK_SPACE   : stopflag := false;
    VK_ESCAPE  :  begin
                   abortflag := true;
                   stopflag := false;
                  end;
  end;//case
end;
```



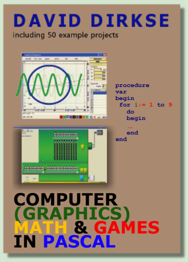
process 2 stop...<space> to continue..<ESC> to finish



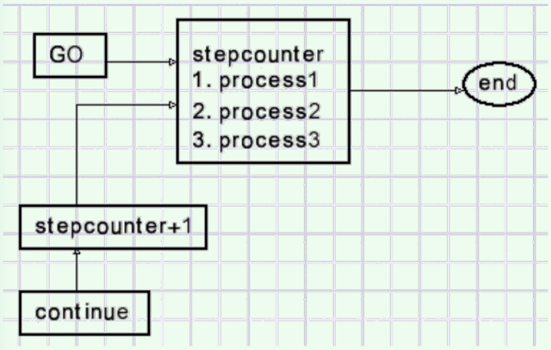
press GO to start



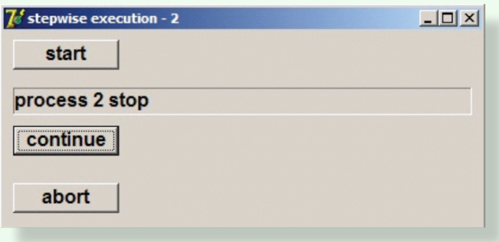
process 1 stop...<space> to continue..<ESC> to finish



Method 2.



This method does not use the pause procedure but buttons to step through the process.
The step counter (byte) selects the processes.
Pressing GO sets the step counter to zero.
`Nextproc` procedure increments the `stepcounter` to 1 and calls process1.



process 2 stop

```
var stepcount : byte;

procedure process1,2,3.....same as above;

procedure nextproc;
begin
  case stepcount of
    0 : begin
          stepcount := 1;
          nextproc;
        end;
    1 : process1;
    2 : process2;
    3 : process3;
    4 : begin
          form1.msglabel.Caption := 'finished';
          stepcount := 0;
        end;
   end;//case
end;
```

Starting the process......

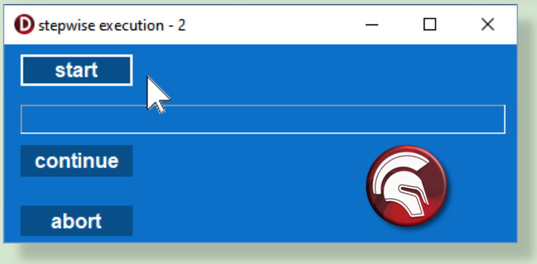
```
procedure TForm1.startBtnClick(Sender: TObject);
begin
  stepcount := 0;
  nextproc;
end;
```

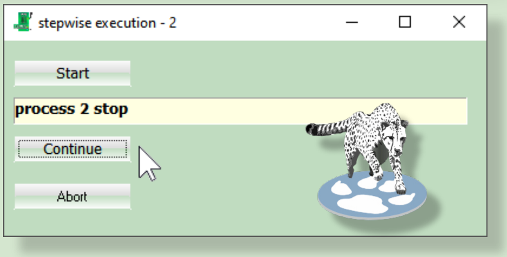
next process step....

```
procedure TForm1.continueBtnClick(Sender: TObject);
begin
  if stepcount > 0 then
    begin
      inc(stepcount);
      nextproc;
    end;
end;
```

finish process without pause.......

```
procedure TForm1.abortBtnClick(Sender: TObject);
begin
  while (stepcount > 0) and (stepcount < 4) do
    begin
      inc(stepcount);
      nextproc;
    end;
end;
```





process 2 stop

This concludes the step-by-step program execution description.

THE SEARCH FOR A SPECIAL NUMBER

Lazarus 1-3

STARTER EXPERT

ABSTRACT

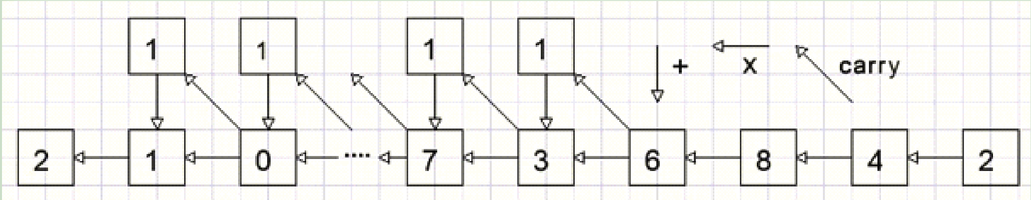
A well known math puzzle is this:
A number has 2 as the lowest digit.
If this digit is moved to the left of the number the effect is multiplication by 2.
Similar:
A number has 2 as the leftmost digit.
If this digit is moved right of the number the effect is division by 2.
The **funny thing about this puzzle** is that is requires only primary school calculation, however even math teachers have problems solving it.
After some tries by hand the question arises if such a number exists anyhow.
Also: are there digits other then 2 possible and: are there other factors possible than 2?
This article describes a Delphi program that searches for number having these properties.

THE ALGORITHM.

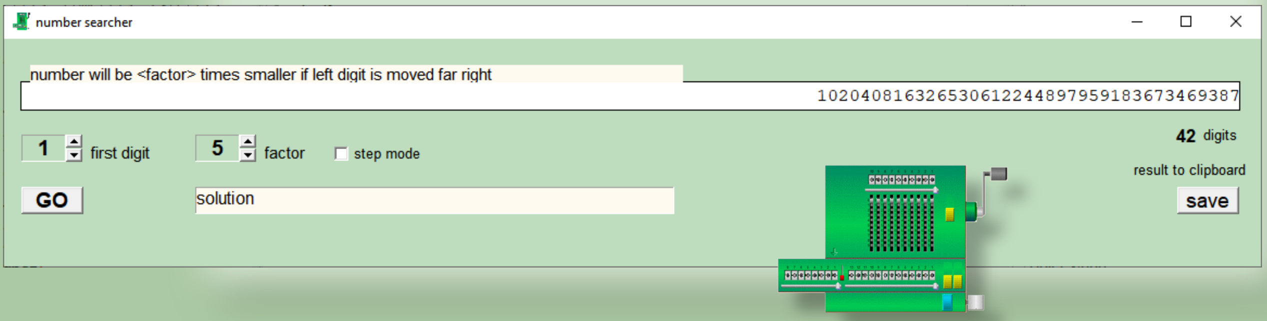
For explanation we use digit 2 as a start and also a multiplication factor of 2.
Note: a carry is written as (..)

```
To start                                        2
2 x 2 = 4                                      42
2 x 4 = 8                                     842
2 x 8 = 16                               (1)6842
2 x 6 = 12 + 1 = 13                     (1)36842
2 x 3 = 6 + 1 = 7                         736842
Finally                                105…736842
2 x 1 = 2                             2105…736842
```

Now the first and the last digit are equal (and no carry exists)
Discard the lowest "2" digit and we have found a number that divides by 2 if the first digit is moved right of the number.



THE PROGRAM.



number will be <factor> times smaller if left digit is moved far right

102040816326530612244897959183673469387

1 first digit

5 factor

☐ step mode

42 digits

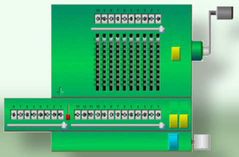
solution

Picture above shows the program at work.
UpDown controls, associated with statictext components, select the first digit and the multiplication factor.

The **GO button** starts the search.
The **SAVE button** copies the result tot he clipboard, which allows for pasting in text editors.

If step mode is checked, the program stops after each iteration, showing intermediate results.
The number is stored in `byte array A[0…120]`
Integer N is the index of `array A[ ].`
Initialization:.
`N := 0;`
`A[0] := start digit.`
Step 1:
`A[N]` is multiplied by 2, the result placed in `A[N+1]` (lower digit) and `A[N+2] (carry).`
Step 2:
`N := N + 1;`
Steps 1 and 2 are repeated until `N = 120` (no solution) or `A[N] = A[0]` and `A[N+1] = 0` (no carry).
This code does the job:

```
Const maxN = 120; //maximal number of digits
.....
procedure TForm1.GObtnClick(Sender: TObject);

var carry,i,f,h,N : byte;
   hit : boolean;
begin
 activecontrol := nil;
 for i := 0 to maxN do A[i] := 0;
 displayA(0);
 A[0] := N0upDown.Position; //starting digit
 f := factorUpdown.Position; //factor
//--
 N := 0; //array A index
 repeat
  inc(N);
  A[N] := A[N-1]*f + A[N];
  h := 0;
  while A[N+h] >= 10  do   //handle carries
   begin
    carry := A[N+h] div 10;
    A[N+h] := A[N+h] mod 10;
    inc(h);
    A[N+h] := A[n+h] + carry;
   end;
  hit := (A[N] = A[0]) and (h=0);
until (N = maxN) or  hit;
//--
 if hit then begin
                  msgText.Caption := 'solution';
                 displayA(N);
              end
  else begin
      msgText.Caption := 'no solution';
      label4.Caption := '';
     end;
end;
```



Step mode
If stepMode is selected , after each iteration then variable stopflag is set true followed by :
`while stopFlag do application.ProcessMessages;`

Pressing **<space bar>** resets the **stopflag** so the process continues.
This needs the **`keyPreview`** property set `true` in the `form1` **object inspector**.

The search procedure started with
`Activecontrol := nil.`
This prevents the <space> character being send to the **GO button** which has the focus after pressing.

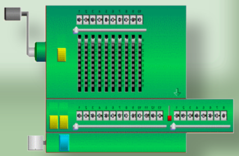
Display of array A[ ]

I use a paintbox for display.
Reason is that this allows for the display of digits in different colors.
Carries are displayed in red.
Courier new font is used, having characters with a fixed width.
`A[1]` is placed right, character position x is decremented to display the next characters left.
`procedure displayA(n : byte);` does the job.
n is the number of digits.
Any non zero character from a higher index of n from `A[n]` is painted in red.

Saving the result to the clipboard.
Clipbrd unit is added to the uses clause.
With the result saved in `string s`, this statement does the job:
`clipboard.As Text := s;`
But first `A[ ]` must be copied to `s`.
The highest digit is transferred first.
At counter `p = 3` “ . ” is placed for clarity, separating triple digits.

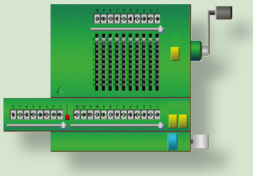
```
 var s : string;
   i,n,p : byte;
begin
 s := '';
 n := maxN;
 while (A[n] = 0) and (n > 0) do dec(n);//find first non zero digit
p := 0;
 for  i := n downto 1 do
  begin
    if p = 3 then begin
                            s := s + ‘.’;
                            p := 0;
                         end;
   s := s + char(A[i] + ord('0'));
   inc(p);
 end;
end;
```

Please refer to the source code for more details.
An astonishing property of the answer (for factor 11) is this:
Split the number in halves and add then.
The result shows only “9” digits: 99......99.



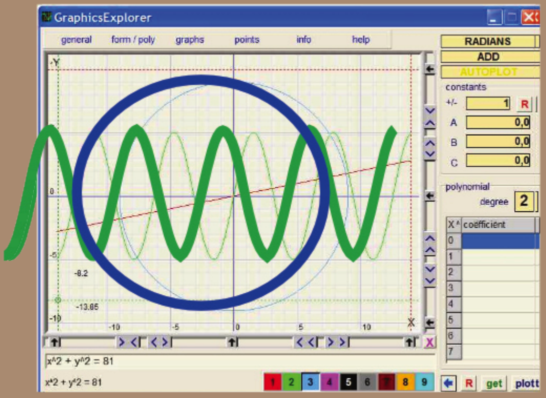
ADVERTISEMENT

David Dirkse's website: `davdata.nl/math`



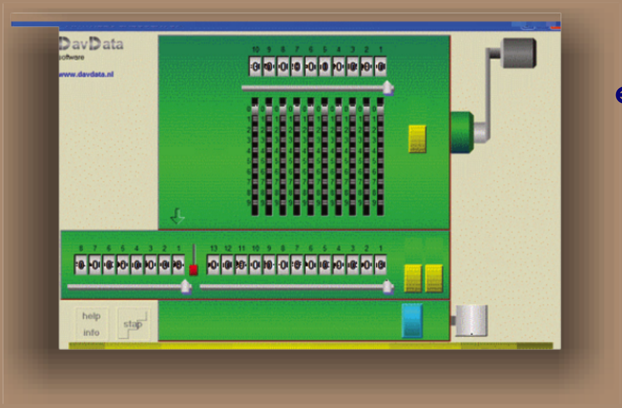
DAVID DIRKSE

including 50 example projects



x^2 + y^2 = 81

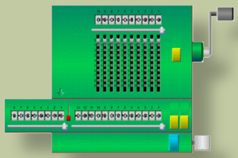
```
procedure
var
begin
 for i:= 1 to 9
   do
   begin

   …
   end
end
```



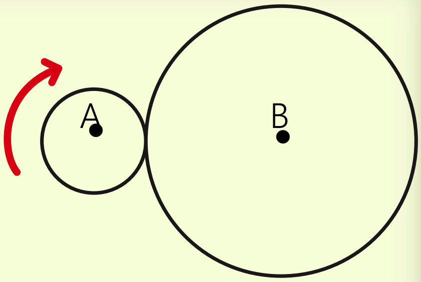
COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

https://www.blaisepascalmagazine.eu/product-category/books/

A ROLLING CIRCLE PROBLEM



In the figure above, the radius of circle A is 1/3 of the radius of circle B.
Starting from the position shown in the figure, circle A rolls around the circle B.
At the end of how many revolutions of circle A will the center of circle first reach its starting point circle?

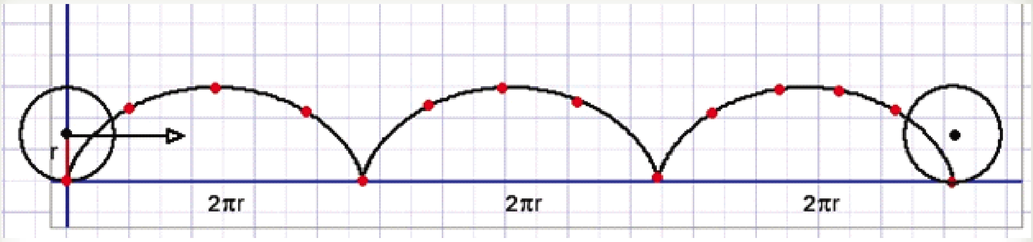


(A) $\frac{3}{2}$ (B) 3 (C) 6 (D) $\frac{9}{2}$ (E) 9

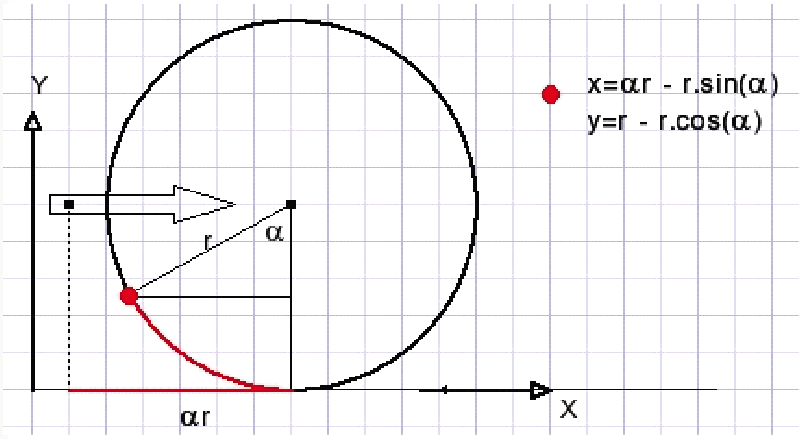
I suggest, before reading further, that the reader tries to answer this question for himself.

THEORY

We may stretch the perimeter of circle B and roll circle A over this straight line.



We notice three revolutions.

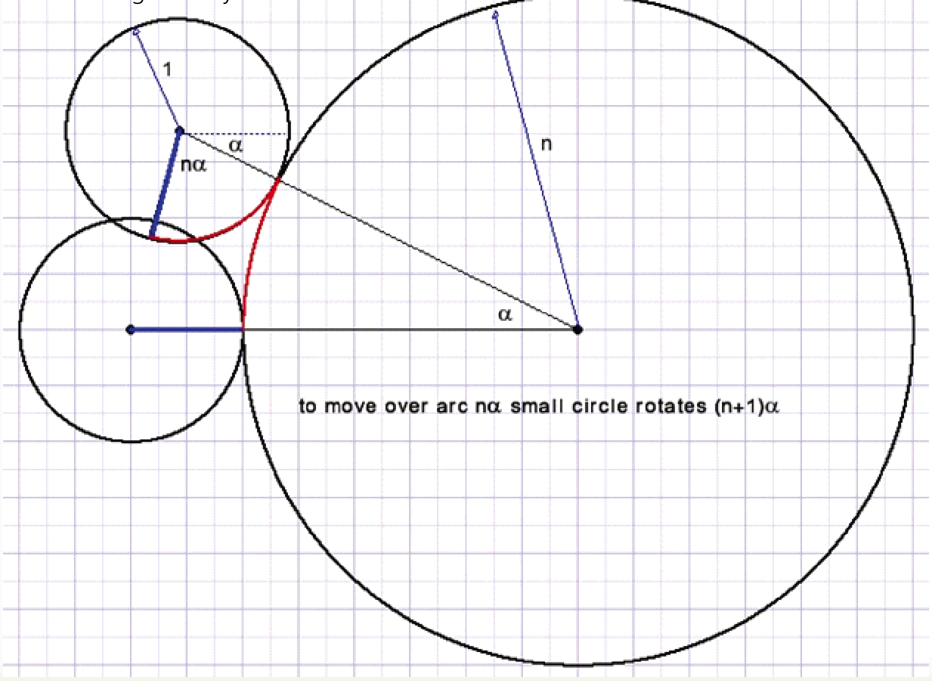


$$x = \alpha r - r.\sin(\alpha)$$
$$y = r - r.\cos(\alpha)$$

For those interested in the math:

Movement of the red dot is given by a parametric function :
x and y are both expressed as a function of rotation angle α.
However, is this still true when circle A rolls over circle B?
Surprising answer: it is not.
See the geometry below:

See the geometry below:



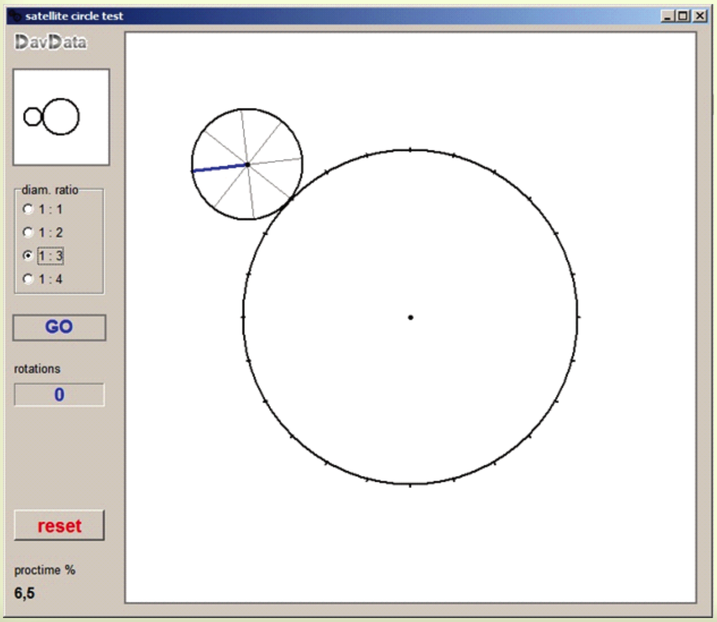
Circle A needs four revolutions to roll completely over circle B, not three as in the case of a straight line.

To illustrate this phenomenon I have written a small Delphi program.

The program adds some options

- Outer- or inner circle A
- Diameter ratios 1:1 , 1:2 , 1:3 , 1 : 4

The theory for the case where A is the inner circle is left to the reader.

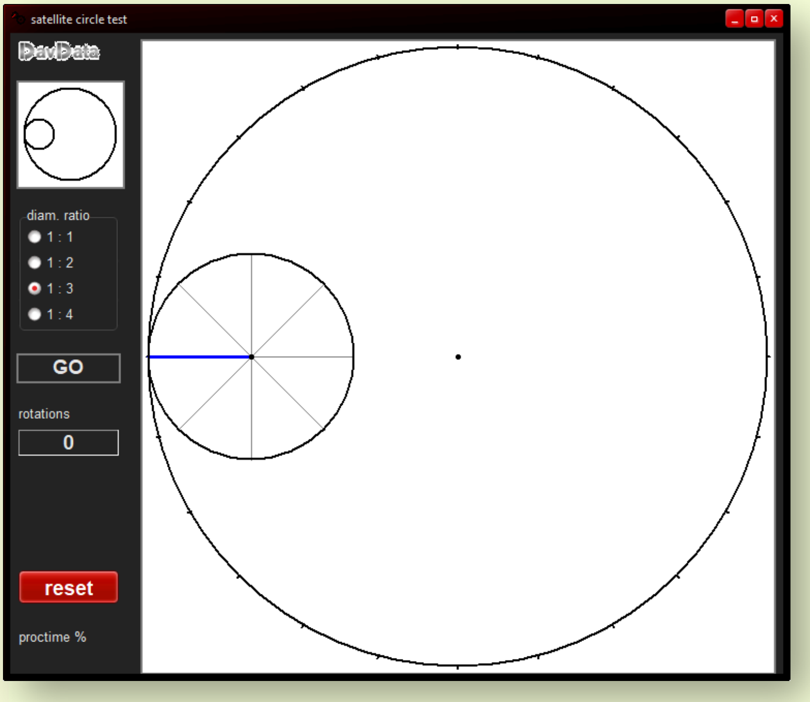


diam. ratio
○ 1 : 1
○ 1 : 2
● 1 : 3
○ 1 : 4

rotations 0

proctime % 6,5

Example in D11 and D12


satellite circle test

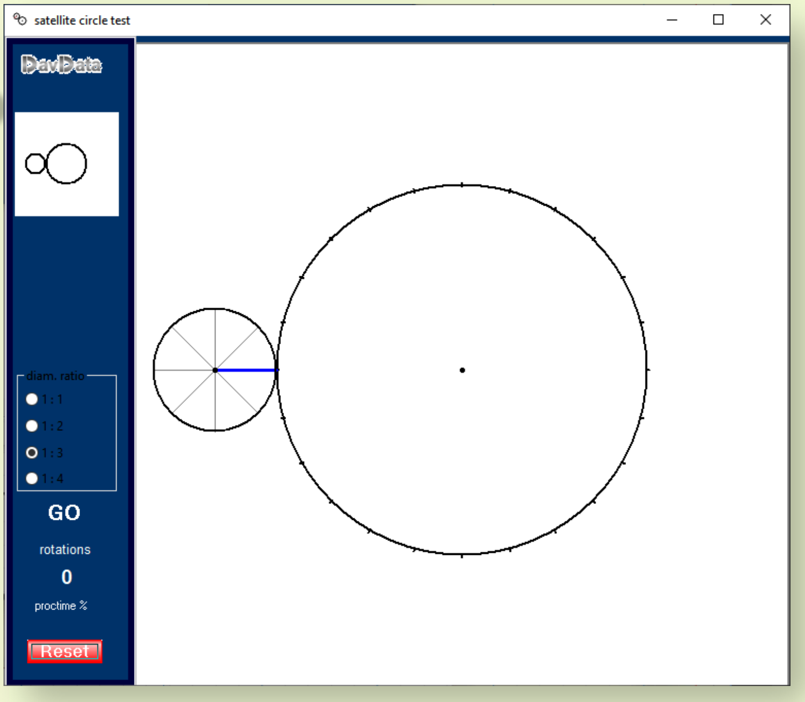
diam. ratio

○ 1 : 1
○ 1 : 2
● 1 : 3
○ 1 : 4

rotations 0

proctime %

Example in Lazarus

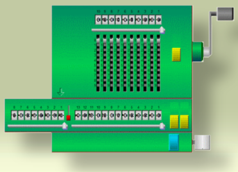

satellite circle test

diam. ratio

○ 1 : 1
○ 1 : 2
● 1 : 3
○ 1 : 4

rotations 0

proctime %



Select the diameter ratio.
Click on the left top image to select A as outer or inner circle.
Press GO to roll circle A over circle B.

PROGRAM DESCRIPTION

The drawing is made in bitmap map.
Map is copied to `paintbox1` on the main form to become visible.
The picture is repainted **100 times per second.**
Left bottom label shows the percentage of processor time needed for an update.
See 2nd picture before.
Angle α starts from 0 and is incremented in steps of 0.005 radians.
A complete revolution takes 2π/0.005 = 1250 steps.
At 100 increments per second revolution time is 12.5 seconds.

Painting a complete new image takes:

- Erasing the bitmap
- Painting circle B with appropriate marks to match the spokes of circle A.
- Painting circle A and it's spokes.
- Copy map to `paintbox` by `form1.paintbox1.canvas.draw(0,0,map)`

CONSTANTS AND VARIABLES

```
Type Tmode = (modeIn, modeOut);
   Tradius = record
       rad1 : word;  //radius of circle B
       rad2 : word;  //radius of circle A
       xamp  : byte; //rotation angle magnifier (=4 if ratio is 1:3)
   end;

Const  pi2 = 2*pi;
       pi04 = pi/4;
       radOut : array[0..3] of Tradius =     //outer circle mode radius
            ((rad1:100 ;rad2:100 ;xamp:2),   //1:1
             (rad1:150 ;rad2: 75 ;xamp:3 ),  //1:2
             (rad1:180 ;rad2: 60 ;xamp:4 ),  //1:3
             (rad1:200 ;rad2: 50 ;xamp:5));  //1:4
       radIn  : array[0..3] of Tradius =     //inner circle mode radius
            ((rad1:300 ;rad2:300 ;xamp:0),
             (rad1:300 ;rad2:150 ;xamp:1),
             (rad1:300 ;rad2:100 ;xamp:2),
             (rad1:300 ;rad2: 75 ;xamp:3));

var map : Tbitmap;
cx,cy : word;                 //center of circle B, measured in pixels
sx,sy : word;                 //center of circle A (satellite)
   radius1,radius2 : word;    //radius of circles B and A, preset from constants array
   angle : double;            //rotation angle of circle A center around B center
   amp             : byte;    //rotating angle magnifier, preset from constants array
   GOflag          : boolean; //control rolling
   mode            : Tmode = modeOut;   //out- or inside circle A
   ModeEntered     : boolean; //for mode button enter / leave control
   frotation       : double;  //circle A rotation in floating point format
   rotations       : byte;    //number of rotations of circle A
```

continuation of the code is on the next page

Continuation of the code

Procedure GOproc controls the rolling.

```
procedure GOproc;

var t1, t2 : Int64;

begin
  repeat
    t1 := getCPUticks;   //start CPU time
    paintmap;            //repaint image and copy to paintbox1
    t2 := getCPUticks;   //stop CPU time
    form1.proctimelabel.caption := formatfloat('0.#',proctime(t2-t1)*0.01);
                         //% of 10msec.
  repeat
    application.processmessages;   //GOflag may be cleared by GObutton release
    t2 := getCPUticks;
    until proctime(t2-t1) > 10000;  //wait for 10 milliseconds since t1
    if angle >= pi2 then angle := angle - pi2;
    angle := angle + 0.005;
    incRotation(0.005);             //rotation increment for small circle A
    GOflag := GOflag and (angle <= pi2);
    until GOflag = false;
end;
```

NOTE: GObutton actually is a `TLabel`. It looks like a button by drawing a rectangle around it on the form canvas. `OnEnter`, `OnLeave`, `OnMouseDown` and `OnmouseUp` events change the button edges.
I do not list here the procedures that paint the circles. Please refer to the source code for details.

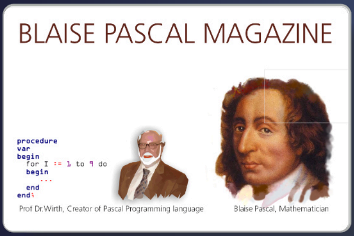
CONCLUSION

The problem answers (*see first picture*) are incomplete.
Answer (F) -none of the above- or (F) 4 is missing.

This once was question 17 of a SAT (Scholastic Aptitude Test) for high school students.
Only a few students noted the error.
I doubt that the teachers who formulated this question were aware of their mistake.

LIB-STICK ON USB CREDIT CARD BLAISE PASCAL MAGAZINE

LIB-STICK USB-CARD: ALL ISSUES / CODE INCLUDED. SAME INTERFACE AS THE INTERNET LIBRARY CHRISTMAS OFFER € 89,00



BLAISE PASCAL MAGAZINE

```
procedure
var
begin
  for I := 1 to 9 do
  begin
    ...
  end
end;
```

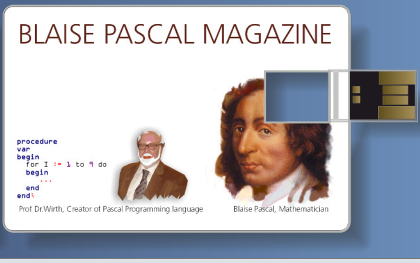
Prof Dr.Wirth, Creator of Pascal Programming language

Blaise Pascal, Mathematician

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands

Prof Dr.Wirth, Creator of Pascal Programming language

editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu

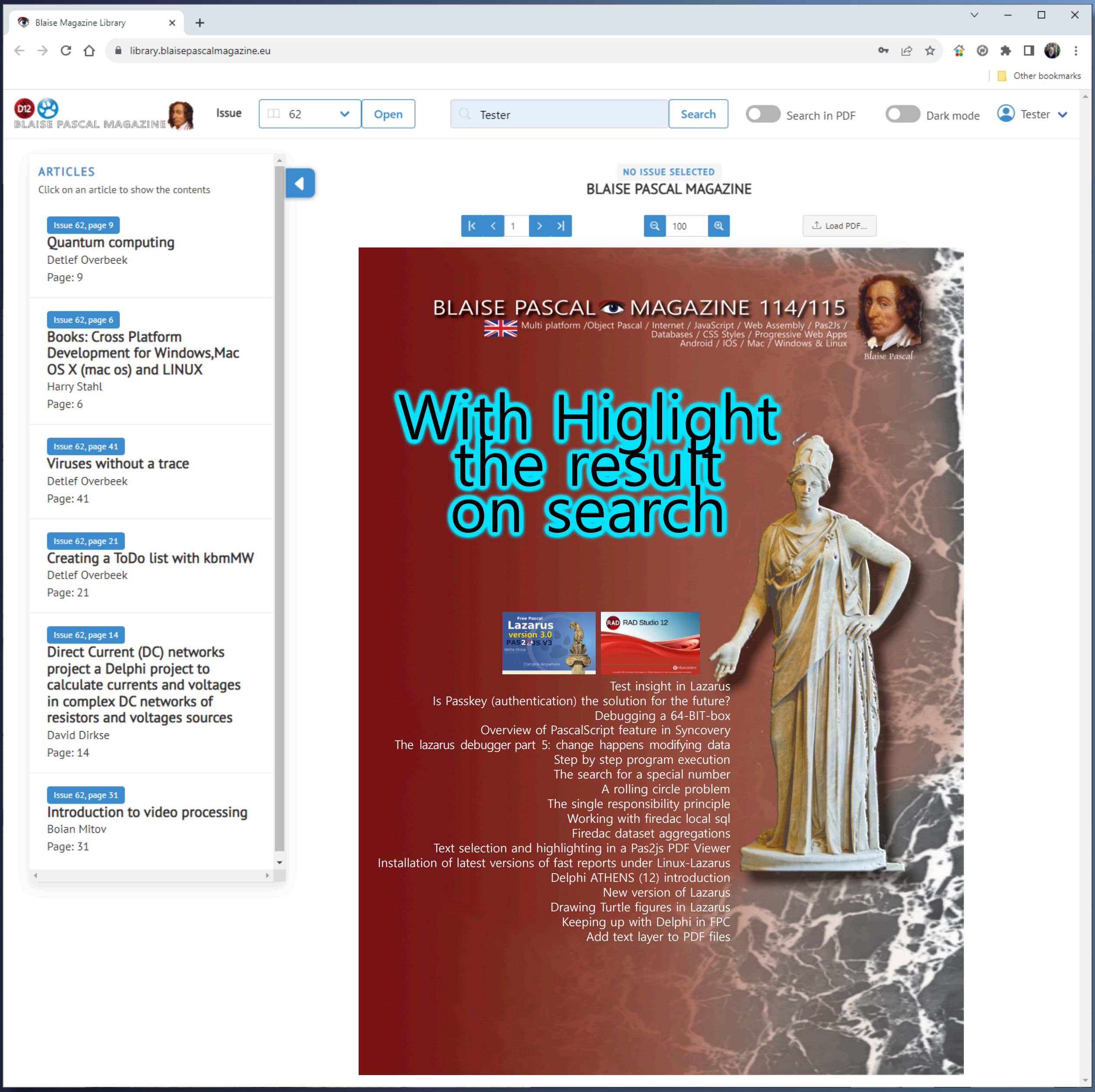


BLAISE PASCAL MAGAZINE

```
procedure
var
begin
  for I := 1 to 9 do
  begin
    ...
  end
end;
```

Prof Dr.Wirth, Creator of Pascal Programming language

Blaise Pascal, Mathematician



Blaise Magazine Library

library.blaisepascalmagazine.eu

Other bookmarks

BLAISE PASCAL MAGAZINE

Issue: 62 Open

Tester Search

Search in PDF

Dark mode

Tester

ARTICLES

Click on an article to show the contents

Issue 62, page 9
Quantum computing
Detlef Overbeek
Page: 9

Issue 62, page 6
Books: Cross Platform Development for Windows,Mac OS X (mac os) and LINUX
Harry Stahl
Page: 6

Issue 62, page 41
Viruses without a trace
Detlef Overbeek
Page: 41

Issue 62, page 21
Creating a ToDo list with kbmMW
Detlef Overbeek
Page: 21

Issue 62, page 14
Direct Current (DC) networks project a Delphi project to calculate currents and voltages in complex DC networks of resistors and voltages sources
David Dirkse
Page: 14

Issue 62, page 31
Introduction to video processing
Boian Mitov
Page: 31

NO ISSUE SELECTED

BLAISE PASCAL MAGAZINE

1 100 Load PDF...

BLAISE PASCAL MAGAZINE 114/115
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux
Blaise Pascal
With Higlight the result on search
Lazarus version 3.0
PAS2JS V3
RAD Studio 12

Test insight in Lazarus
Is Passkey (authentication) the solution for the future?
Debugging a 64-BIT-box
Overview of PascalScript feature in Syncovery
The lazarus debugger part 5: change happens modifying data
Step by step program execution
The search for a special number
A rolling circle problem
The single responsibility principle
Working with firedac local sql
Firedac dataset aggregations
Text selection and highlighting in a Pas2js PDF Viewer
Installation of latest versions of fast reports under Linux-Lazarus
Delphi ATHENS (12) introduction
New version of Lazarus
Drawing Turtle figures in Lazarus
Keeping up with Delphi in FPC
Add text layer to PDF files

SOLID PRINCIPLES IN DELPHI GDK software
❶ – THE SINGLE RESPONSIBILITY PRINCIPLE

BY MARCO GEUZE

Beginner Expert

ABSTRACT

Today, we'll dive into the S of the Solid principles: The Single Responsibility Principle.
But before we start, just a quick refresher on Solid.

SOLID is an acronym for a set of five software development principles, which if followed, are intended to help developers create flexible and clean code. The five principles are:

❶ **The Single Responsibility Principle**
Classes should have a single responsibility and thus only a single reason to change.

❷ **The Open/Closed Principle**
Classes and other entities should be open for extension but closed for modification.

❸ **The Liskov Substitution Principle**
Objects should be replaceable by their subtypes.

❹ **The Interface Segregation Principle**
Clients should not be forced to depend upon interfaces that they do not use.

❺ **The Dependency Inversion Principle**
Depend on abstractions rather than concretions.

So, the single responsibility principle.
As the name suggests, each class in a program must have a single responsibility for only one part of the program's functionality.
But that seems easier than it is.
What exactly is a one part of a program, and how to know when to separate functionality?
It's too simple to say that a class should only do one thing.

Robert C. Martin expresses the principle as '**Gather together the things that change for the same reasons. Separate those things that change for different reasons**', and more recently '**This principle is about people**'. That should point us into the right direction.

When you write a software module, you want to make sure that when changes are requested, those changes can only originate from a single person, or a single tightly coupled group of people representing a single narrowly defined business function.
This means that a software module or class should have **one responsibility for that particular group of people**.

It is easier to explain this by means of an example. Let's take a look at this following class:

```
type
  TShip = class
  private
    FPosition: TPoint;
    FHeading: Integer;
    FSpeed: Integer;
    FCargoLoad: string;
  public
    procedure SetHeading(NewHeading: Integer);
    procedure SetSpeed(NewSpeed: Integer);
    function GetCoordinate: TPoint;
    procedure PlotCourse;
    procedure LoadCargo(NewCargo: string);
    procedure PrintCargo;
    procedure ReportPosition;
    procedure CalculateProfit;
  end;
```

Continuation 1
Solid Principles

This is a class that is doing a couple of things, all about managing a ship's position and course, its load and some reporting things. As this is a brief example to show how to think about the Single Responsibility Principle, don't pay too much attention to the details of the code itself; it is all about the large overview of the structure of this particular class.

I think we all know these kind of 'God' classes. Usually packed with lots of functionality and code, and managing one particular part or module of your program. The question is, if we need to make some changes to this class, how can we refactor this class to make sure we gather together the things that change for the same reason, and separate those things that change for different reasons.

Let's stop for a moment and think about responsibilities of this code regarding to the (group of) people. We can define a couple of specific people that will have some responsibility regarding the ship's management, direction and heading, and reporting tools. So let's say in this case we define a skipper, a navigator, a cargo load master and a financial manager. If you think of these different roles, it's suddenly very easy to separate this class into different modules, with just one responsibility to that particular role. We should have a class for setting the heading and power (skipper), one for managing position and plotting the course of the ship (navigator), one for managing the load of this ship (cargo load master), and one for all our (financial) reporting (financial manager).

Our new classes might now look like this:

```
type
 TShipLocation = class
 private
  FPosition: TPoint;
 public
  function GetCoordinate: TPoint;
  procedure PlotCourse;
  procedure ReportPosition;
 end;

 TShipMovement = class
 private
  FHeading: Integer;
  FSpeed: Integer;
 public
  procedure SetHeading(NewHeading: Integer);
  procedure SetSpeed(NewSpeed: Integer);
 end;

 TCargo = class
 private
  FCargoLoad: string;
 public
  procedure LoadCargo(NewCargo: string);
  procedure PrintCargo;
 end;

 TShipReport = class
 public
  procedure CalculateProfit;
 end;

 TShip = class
 private
 public
  // reference to subclasses
 end;
```

Continuation 2
Solid Principles

Would you have done the same if you didn't think of the people behind the class's responsibilities? Maybe, but I can imagine the ShipLocation and ShipMovement class could have ended up in the same class.

So, what happens now if we got a feature request from the skipper to add a bow thruster to make it easier to steer the ship in small canals?
Just make this change in the ShipMovement class, without affecting any of the other classes.
And if we want to implement a new cargo load system? Just alter the Cargo class, again without touching any of the other classes.

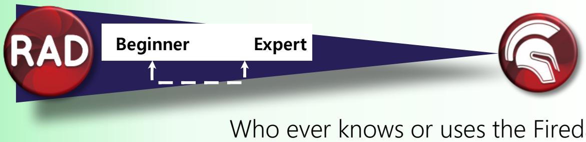
I hope by now you see why the single responsibility principle is really about people, or actors, and the responsibility of the functionality of modules or classes of your program in relation to these people. And of course, you can apply this on different levels of your program, from modules to classes to specific functions.

If you always have this principle in mind when refactoring or designing a module or class, I'm sure your code will be better maintainable and is easy to change.

WORKING WITH FIREDAC LOCAL SQL

ARTICLE 2 PAGE 1 / 3

BY KEES DE KRAKER GDK software



Who ever knows or uses the Firedac component LocalSQL? Or the BatchMove? For most developers, these components are relatively unknown, even though they are very useful. In two blogs, I want to explain the added value of these components and show how they work. This blog is about the LocalSQL component.

THE ADDED VALUE

Using the LocalSQL dataset, you can combine data from different datasets in a Firedac query. The component ensures that datasets become available as if they were database tables. Datasets can therefore be queried with an SQL query from a TFDQuery component.

This is very useful if you have data from different sources. For example from multiple databases. Or partly in memory and partly from the database. Or from different dataset components, for example ADO and Firedac. With LocalSQL, you can bring this data together very easily with a simple SQL query. It allows you to show the combined data very simply in a grid or create combined statistics.

Below I work out an example, combining data from a CSV with data from the database. In the database, I have stored customers with an id and name. In the CSV, only the id of the customer is used. When displaying the contents of the CSV, I want to show the customer's name directly. This can be done with LocalSQL without too much effort.

THE BASIC DESIGN

Below you can see the setup of my application. On the left is the connection to the database containing the table Customer. That is read through the QueryCustomers. On the right you see the CsvDataset, a Firedac MemTable. In this, the CsvMove (TFDBatchMove) loads the data from the CSV.

In the middle, you see the LocalSQL component, which I have named CombinedLocalSQL. Underneath, this component uses SQLite and therefore requires a SQLite connection. You don't need to configure that connection any further, just indicate that the driver is SQLite. In this case, this is the LocalSQLConnection. The CombinedDataset component is a TFDQuery containing the SQL query that collects the data from the datasets.

The grid first shows two columns with the customer data from the database. The three columns after that come from the CSV.

BY KEES DE KRAKER

Continuation 1
Local Sql

Who ever knows or uses the Firedac component LocalSQL? Or the BatchMove? For most developers, these components are relatively unknown, even though they are very useful. In two blogs, I want to explain the added value of these components and show how they work. This blog is about the LocalSQL component.

THE ADDED VALUE

Using the LocalSQL dataset, you can combine data from different datasets in a Firedac query. The component ensures that datasets become available as if they were database tables. Datasets can therefore be queried with an SQL query from a TFDQuery component.

This is very useful if you have data from different sources. For example from multiple databases. Or partly in memory and partly from the database. Or from different dataset components, for example ADO and Firedac. With LocalSQL, you can bring this data together very easily with a simple SQL query. It allows you to show the combined data very simply in a grid or create combined statistics.

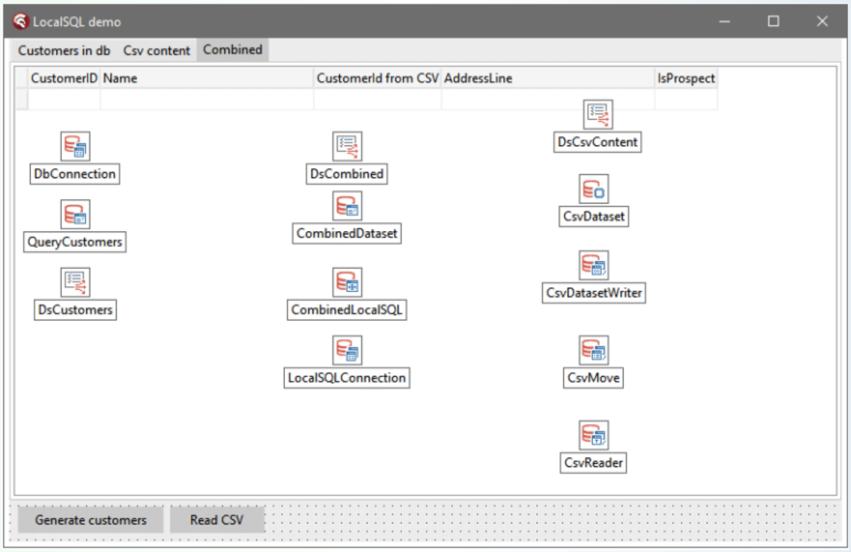
Below I work out an example, combining data from a CSV with data from the database. In the database, I have stored customers with an id and name. In the CSV, only the id of the customer is used. When displaying the contents of the CSV, I want to show the customer's name directly. This can be done with LocalSQL without too much effort.

THE BASIC DESIGN

Below you can see the setup of my application. On the left is the connection to the database containing the table Customer. That is read through the QueryCustomers. On the right you see the CsvDataset, a Firedac MemTable. In this, the CsvMove (TFDBatchMove) loads the data from the CSV.

In the middle, you see the LocalSQL component, which I have named CombinedLocalSQL. Underneath, this component uses SQLite and therefore requires a SQLite connection. You don't need to configure that connection any further, just indicate that the driver is SQLite. In this case, this is the LocalSQLConnection. The CombinedDataset component is a TFDQuery containing the SQL query that collects the data from the datasets.

The grid first shows two columns with the customer data from the database. The three columns after that come from the CSV.

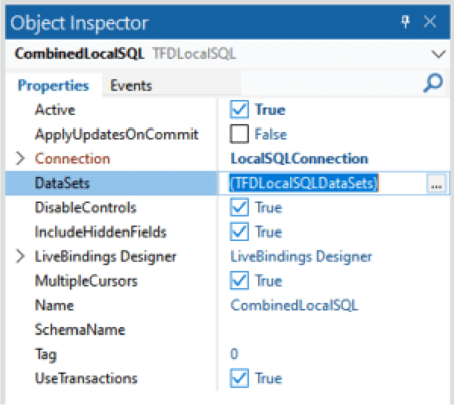


Continuation 2
Local Sql

THE CONFIGURATION

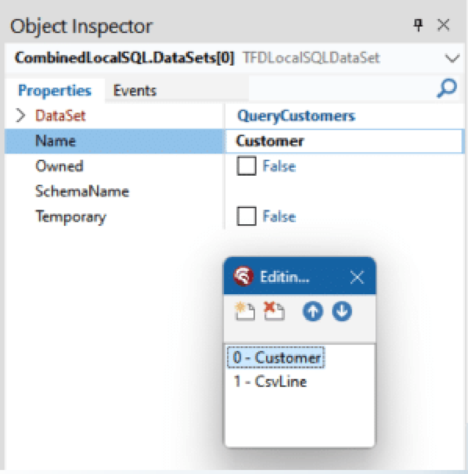
The LocalSQL component, as mentioned before, makes datasets available as if they were database tables. To make that possible, we specify which datasets we want to use and what “table name” they will be given. In the properties of the LocalSQL component, you will find the “DataSets” option for this purpose (*see images below*). Add the datasets and give them the name you want to use as the table name in the query.

By setting the LocalSQL component to Active, you can then run the query live after this and easily add the fields, for example. Be careful with that, because once you run the query, the underlying datasets are also set to Active.



Object Inspector
CombinedLocalSQL TFDLocalSQL
Properties | Events

Property	Value
Active	☑ True
ApplyUpdatesOnCommit	□ False
> Connection	LocalSQLConnection
DataSets	(TFDLocalSQLDataSets)
DisableControls	☑ True
IncludeHiddenFields	☑ True
> LiveBindings Designer	LiveBindings Designer
MultipleCursors	☑ True
Name	CombinedLocalSQL
SchemaName	
Tag	0
UseTransactions	☑ True



Object Inspector
CombinedLocalSQL.DataSets[0] TFDLocalSQLDataSet
Properties | Events

Property	Value
> DataSet	QueryCustomers
Name	Customer
Owned	□ False
SchemaName	
Temporary	□ False

Editin...
0 - Customer
1 - CsvLine

The combined query
We use a Firedac query to apply the LocalSQL. The connection of this query is the same as the LocalSQL connection. And that’s all. We can now start creating the query as if we had a table Customer and a table CsvLine.

```
SELECT *
FROM Customer c
JOIN CsvLine csv ON (csv.CustomerId = c.CustomerId)
```

And of course you can extend and use this with all the possibilities that are in (SQLite) SQL. Think of the WHERE clause, but also aggregation functions such as SUM, COUNT, etc. The Firedac query then works as usual.

IN SUMMARY

The LocalSQL component opens a diversity of possibilities to combine, filter and aggravate data. It is widely applicable because it supports all forms of datasets. It can make complicated datasets with lookup fields a lot simpler. And it works quickly and easily. Highly recommended to apply in your projects.

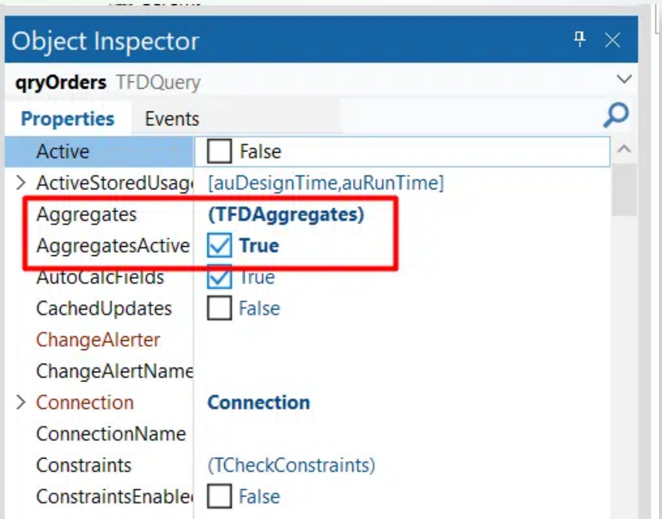
FIREDAC DATASET AGGREGATIONS

BY KEES DE KRAKER

RAD Beginner Expert

Did you know that in a **Firedac** query or table, you can work with runtime **aggregations** at the dataset level? Displaying a total amount of all orders loaded, or a total of all orders yet to be shipped can be done very easily without the need for a separate query.

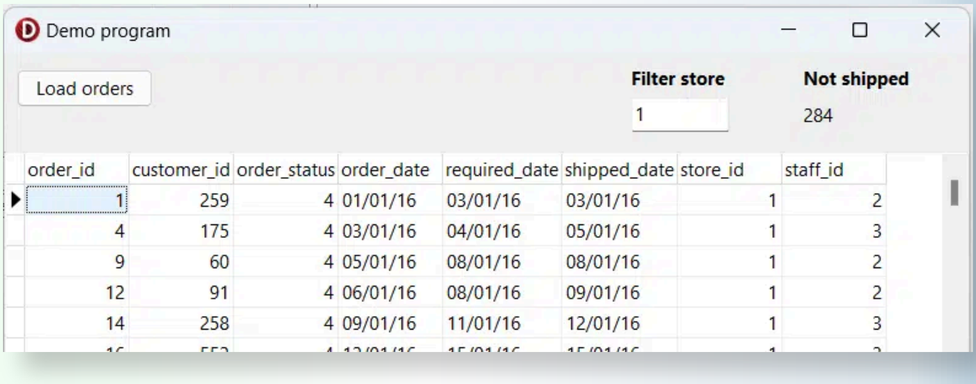
Maybe you were already familiar with an aggregated field, but that is limited to row level. However, with the property `Aggregrates` of a `TFDDataSet`, you can also totalise at dataset level.


Object Inspector

qryOrders TFDQuery

Active	□ False
> ActiveStoredUsag	[auDesignTime,auRunTime]
Aggregates	**(TFDAggregates)**
AggregatesActive	☑ **True**
AutoCalcFields	☑ True
CachedUpdates	□ False
ChangeAlerter	
ChangeAlertName	
> Connection	**Connection**
ConnectionName	
Constraints	(TCheckConstraints)
ConstraintsEnable	□ False

As an example, I have a demo program in which I retrieve all orders from a database into a table. I can filter these by store id. At the top right, you can see the number of orders yet to be sent. This number is calculated by an aggregated field of the query.


Demo program

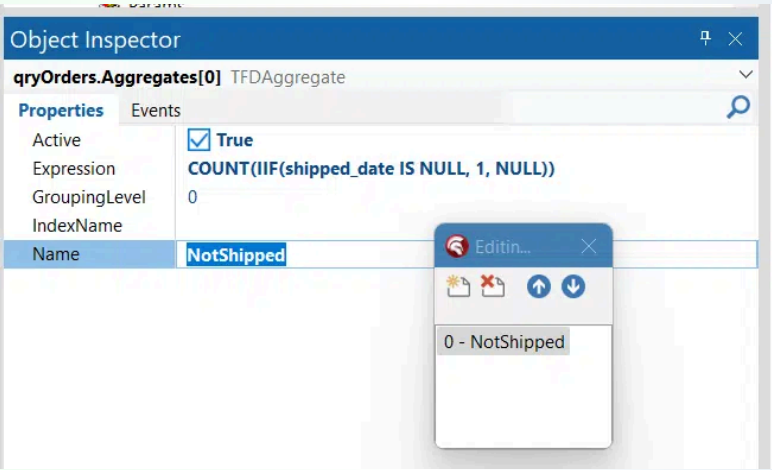
Filter store: 1

Not shipped: 284

order_id	customer_id	order_status	order_date	required_date	shipped_date	store_id	staff_id
1	259	4	01/01/16	03/01/16	03/01/16	1	2
4	175	4	03/01/16	04/01/16	05/01/16	1	3
9	60	4	05/01/16	08/01/16	08/01/16	1	2
12	91	4	06/01/16	08/01/16	09/01/16	1	2
14	258	4	09/01/16	11/01/16	12/01/16	1	3

To start with, I define an aggregation field in the aggregation property of the query. The most important field here is Expression. This is where we define the aggregation we want to perform. Possibly with a condition in it. The standard Firedac expression syntax is used for this.

Continuation 1
Dataset Aggregations



qryOrders.Aggregates[0] TFDAggregate

Active	☑ True
Expression	COUNT(IIF(shipped_date IS NULL, 1, NULL))
GroupingLevel	0
IndexName	
Name	NotShipped

0 - NotShipped

In the above example, I count the number of records (COUNT) of the field Shipped_Date with the condition that this field is counted only if the value is empty. For this, the function IIF can be used. Other simple examples you can use here are:

SUM(order_amount)

MIN(order_date)

As you can see in the properties, you can set a field to active. If you don't, it will obviously not be calculated. A name can also be useful, to be able to find the aggregation in the code.

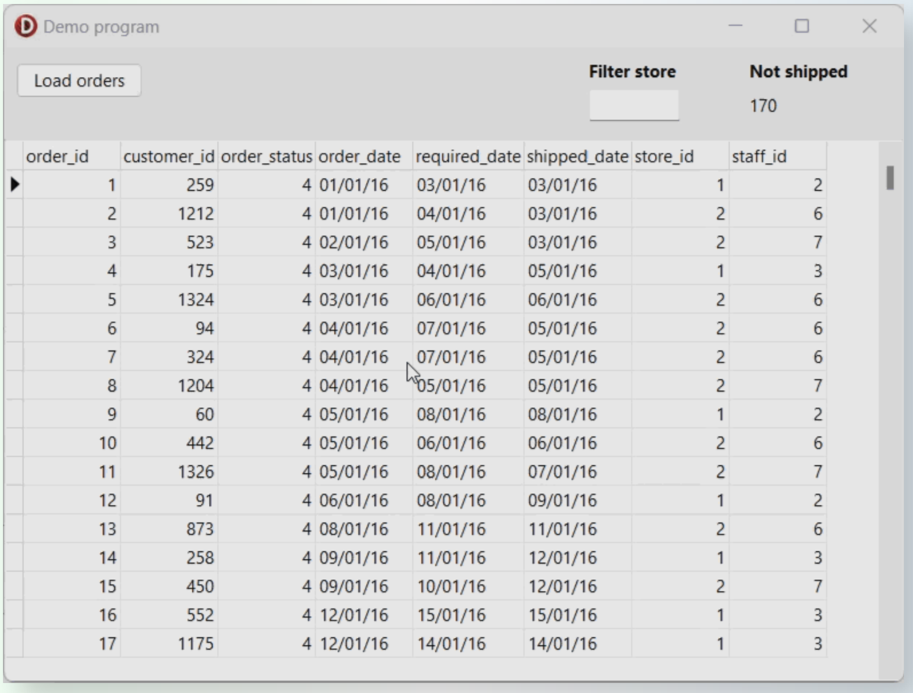
Unfortunately, you cannot link an aggregation as a database field to a data-aware control. So rendering has to be handled in code. I created a separate procedure for this, so I can easily call it when a refresh is needed. I call this procedure in the AfterOpen event of the query and when filtering on a store.

```
procedure TfrmDemoApp.qryOrdersAfterOpen(DataSet: TDataSet);
begin
  ShowNotShippedAggregation;
end;

procedure TfrmDemoApp.ShowNotShippedAggregation;
begin
  var NotShipped := qryOrders.Aggregates.Items[0].Value;

  if NotShipped = Null then
     lblNotShipped.Caption := '0'
  else
    lblNotShipped.Caption := NotShipped;
end;
```

Continuation 2
Dataset Aggregations

On the query, remember to set the property `AggegratesActive` to `True`, otherwise the aggregations will not be calculated. Another point to bear in mind: the aggregates operate on the data retrieved in the dataset. By default, **Firedac** retrieves 50 records. If you want a total which is based on all records go to the FetchOptions in the query properties and set the Mode to fmAll.

See below the result in the demo application.



Filter store

Not shipped
170

order_id	customer_id	order_status	order_date	required_date	shipped_date	store_id	staff_id
1	259	4	01/01/16	03/01/16	03/01/16	1	2
2	1212	4	01/01/16	04/01/16	03/01/16	2	6
3	523	4	02/01/16	05/01/16	03/01/16	2	7
4	175	4	03/01/16	04/01/16	05/01/16	1	3
5	1324	4	03/01/16	06/01/16	06/01/16	2	6
6	94	4	04/01/16	07/01/16	05/01/16	2	6
7	324	4	04/01/16	07/01/16	05/01/16	2	6
8	1204	4	04/01/16	05/01/16	05/01/16	2	7
9	60	4	05/01/16	08/01/16	08/01/16	1	2
10	442	4	05/01/16	06/01/16	06/01/16	2	6
11	1326	4	05/01/16	08/01/16	07/01/16	2	7
12	91	4	06/01/16	08/01/16	09/01/16	1	2
13	873	4	08/01/16	11/01/16	11/01/16	2	6
14	258	4	09/01/16	11/01/16	12/01/16	1	3
15	450	4	09/01/16	10/01/16	12/01/16	2	7
16	552	4	12/01/16	15/01/16	15/01/16	1	3
17	1175	4	12/01/16	14/01/16	14/01/16	1	3

Codolex
Simplify development

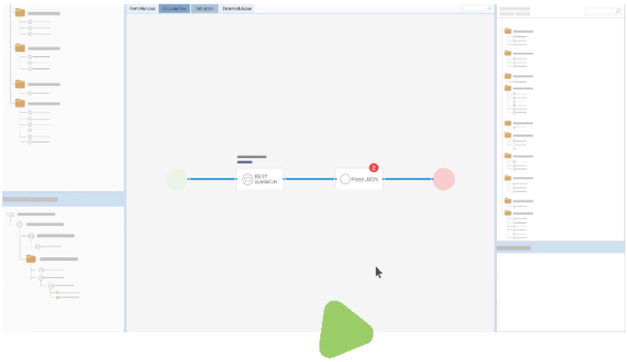
embarcadero®
Official Technology Partner

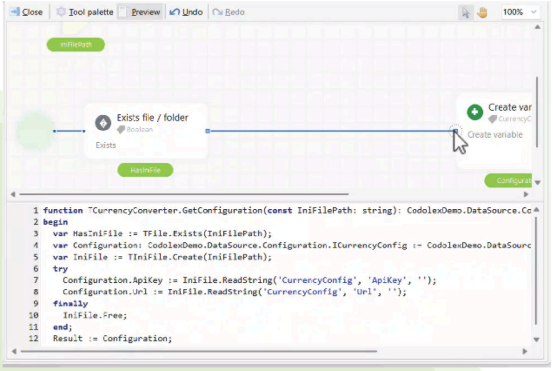
Revolutionary.

Codolex is a low code solution specifically created for Delphi allowing you to develop rapidly whilst remaining in control of the source code.

Visual development

Develop your code through visual design. Understand logic faster through visual representation and adjust easily by modifying the flow. A picture says more than a thousand words!





```
function TCurrencyConverter.GetConfiguration(const IniFilePath: string): CodolexDemo.DataSource.Co
begin
  var HasIniFile := TFile.Exists(IniFilePath);
  var Configuration: CodolexDemo.DataSource.Configuration.ICurrencyConfig := CodolexDemo.DataSourc
  var IniFile := TIniFile.Create(IniFilePath);
  try
    Configuration.ApiKey := IniFile.ReadString('CurrencyConfig', 'ApiKey', '');
    Configuration.Url := IniFile.ReadString('CurrencyConfig', 'Url', '');
  finally
    IniFile.Free;
  end;
  Result := Configuration;
```

Code generation

Codolex generates code that is immediately usable in your projects. Using the preview function in the modeller you can immediately see what code is generated for the flow. There is no vendor lock-in because everything is generated into code.

Codolex provides everything to simplify your development!

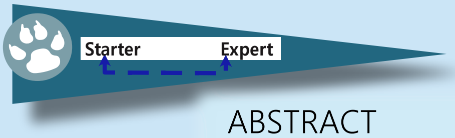
SCAN ME

www.codolex.com

Try it out

TEXT SELECTION AND HIGHLIGHTING IN A PAS2JS PDF VIEWER

BY MICHAEL VAN CANNEYT



ABSTRACT

In previous articles we introduced a way to show a PDF in the browser, and to search in a PDF. In this article we add a missing feature: highlighting search results in the text and text selection.

1 INTRODUCTION

In a series of articles on **PAS2JS** we demonstrated how to show a **PDF** in the browser, and how to search for a text in the shown `PDF`.
This can be done relatively easy using **PDF.js,** the **Javascript PDF** displaying library by **Mozilla**.

Later on we added the capability to search in a series of PDFs using a server-side mechanism and an indexer written in **Free Pascal.**
The result is the basis for the **Blaise Pascal Magazine Library**, a **search-able library of articles** published in Blaise Pascal Magazine: a program that works both offline and online.

These demo programs had 2 drawbacks:
The first drawback was that the search mechanism was **limited to finding the text** and **displaying the correct page** in the PDF.

The second drawback was that it was impossible to select (*and possibly copy*) text in the PDF.
Luckily, the **PDF.js API** contains some calls that solve both these problems.
In this article we show how to use one of these calls, thereby **solving both problems at once**.

2 UPGRADING PDF.JS

Before diving in, a small detour is needed:
In between the publication of the various programs that show the workings of the **PDF.js** library, the library has changed in a way that necessitates changes in the Pascal code and the HTML.

These changes are not extensive, but are necessary or your program will stop working - if it has not already stopped working, which is likely if you used the online distribution of **PDF.js** through some **CDN (Content Delivery Network).**

Previously, the **PDF.js library** exposed a global variable **`pdfjsLib`** which was defined in the programs as:

```
var
pdfjsLib : TPDFJSStatic; external name 'pdfjsLib';
```

The **PDF.js library** is now created as a **Javascript** module, similar to a dynamically loadable library.
The library filename extension was also changed to `.mjs` to reflect this.
As a result, the above variable is **no longer defined if you load it as a regular script,** because the browser will refuse to load the script.

The solution is to load the library as a module:
This simply means that you must add a `type` attribute with the value `module` to the `script tag` that includes PDF.js, for example as follows:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/pdf.js/4.0.269/pdf.mjs"
type="module">
</script>
```

Then the script will be correctly loaded and the global variable will be defined.
If you are using a private copy of the legacy build of **PDF.js**, then there is no need to change anything.

3 A SELECTION AND HIGHLIGHTING MECHANISM

The reason why the first versions of the PDF viewer did not allow highlighting and selection is because basically, the **PDF is rendered as a bitmap** on a HTML canvas, including the text.
It should be clear that the drawn text bitmap cannot be selected (*unless using OCR-Optical Character Recognition*), and that some other mechanism is needed.

Luckily, **PDF.js** offers a solution: it contains a mechanism to render the text elements of a PDF file as HTML, overlaid on the canvas.
This **HTML** is styled so it is completely transparent, and the user does not see it.
But it is actual HTML that is part of the HTML page's **DOM**, and can be manipulated with the usual **DOM and CSS techniques.**

Additionally, when selecting something in the browser, the 'hidden' HTML is taken into account: the **'selected' pseudo-element** will also work, and can be styled:
by **changing the background color** of the 'selected' pseudo-element, the selected text can be made visible.

This means that if we use the **PDF.js API** for rendering text, we have our text selection mechanism.

Since the **rendered HTML contains the actual text** (*but simply rendered invisibly*), it means that when we searched for a text and a page containing a match of the text is displayed,
we can traverse the created HTML and highlight matches in the text.

In the below, we'll show how to do this.

4 ALLOWING SELECTION: RENDERING PAGE TEXT AS HTML.

The **PDF.js API** has 2 calls that create or update the **HTML DOM** with the text of the **PDF**. Both are implemented using a background task, and they both return an instance of this background task:

```
function renderTextLayer(params : TPDFJSRenderTextLayerParameters):TPDFTextLayerRenderTask;
function updateTextLayer(params : TPDFJSUpdateTextLayerParameters):TPDFTextLayerRenderTask;
```

The call that interests us is the `RenderTextLayer` task.
It accepts the following parameter object:

```
TPDFJSRenderTextLayerParameters = class
  textContentSourceStream  : TJSReadableStream; external name 'textContentSource';
  textContentSourceItems   : TTextContent; external name 'textContentSource';
  container              : TJSHTMLElement;
  viewport               : TPDFPageViewport;
  isOffscreenCanvasSupported : Boolean;
  textDivs               : TJSHTMLElementArray;
  textDivProperties      : TJSMap;
  textContentItemsStr    : TStringDynArray;
end;
```

The first 5 fields in this class are input:

`textContentSourceStream` The text contents of the PDF as a `stream`.
`textContentSourceItems` The text contents of the PDF as returned by the `getTextContent` call of the `TPDFPageProxy` object.
`container` The HTML element in which to render the text.
`viewport` The viewport using which the PDF was rendered.
`isOffscreenCanvasSupported` Set to True if the PDF layer can use an offscreen canvas. (*needed to determine text widths*)

Continuation 4

The last 3 fields of the class are output: they will be filled after the `renderTextLayer` call did its work. Initially they should be set to empty arrays.

`textDivs`	An **array with the generated HTML** elements.
`textDivProperties`	An **array with the properties needed to generate the HTML** elements.
`textContentItemsStr`	The raw texts of the generated HTML elements.

We'll demonstrate this call in the **PDF search example** we presented earlier. In that example, the user could enter an **URL** or **select a PDF file** which would then be shown, and which could be searched. As a reminder - *figure 1 on article page 3* - shows what the application looked like. We'll reuse and expand that example, the source code will of course again be available for you.

In order to use the `renderTextLaye` call, we need to **add additional elements** to the **HTML** for our PDF viewer. Where previously we had the following **HTML** in which the PDF was shown:

```
<div class="is-flex is-justify-content-center">
  <canvas id="PDFCanvas" height="737" width="538"></canvas>
</div>
```

We now need an extra element that will be used to overlay the text (*with ID* `pdfTextLayer`), and another one to carry an extra style parameter (*with id* `pdfViewer`):

```
<div class="is-flex is-justify-content-center">
  <div id="pdfViewer" style="position: relative">
    <div class="canvaswrapper" >
      <canvas id="PDFCanvas" height="737" width="538"></canvas>
    <div>
    <div id="pdfTextlayer" class="textLayer">
    </div>
  </div>
</div>
```

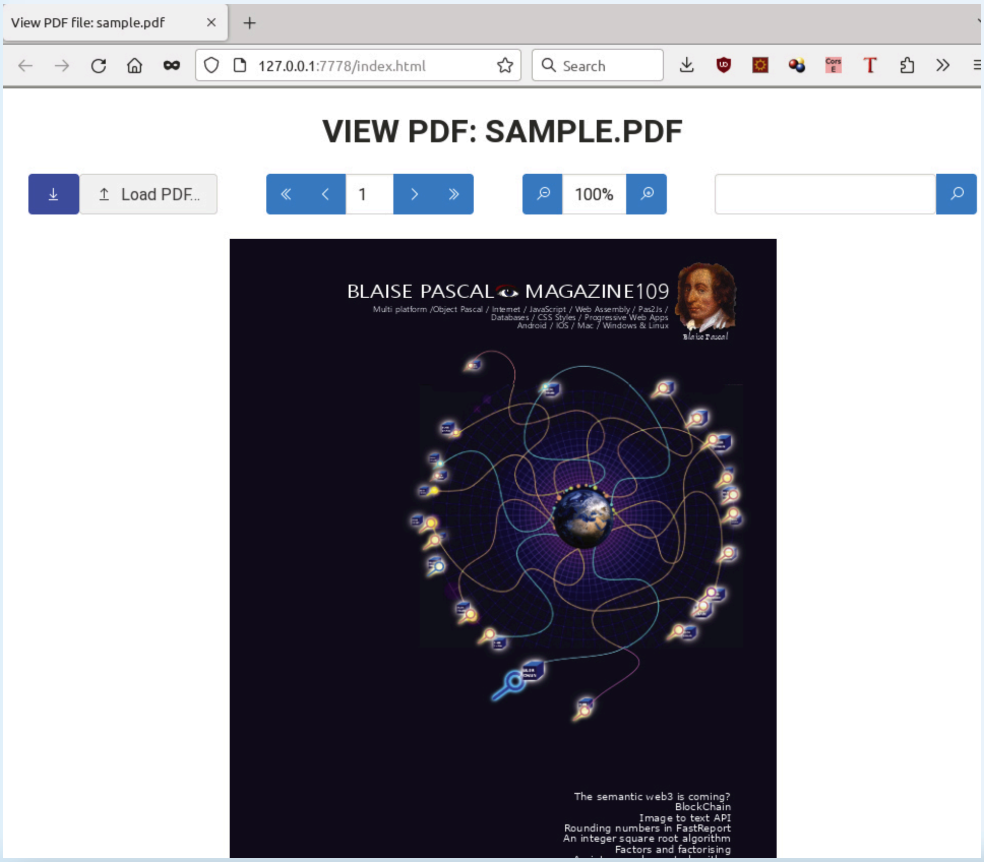

View PDF file: sample.pdf
127.0.0.1:7778/index.html
Search
VIEW PDF: SAMPLE.PDF
Load PDF...
1
100%
BLAISE PASCAL MAGAZINE109
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux
Blaise Pascal
The semantic web3 is coming?
BlockChain
Image to text API
Rounding numbers in FastReport
An integer square root algorithm
Factors and factorising

Figure 1:
The PDF viewer in action

Continuation 4

The two IDs will be bound to two variables in our program:

```
TMyApplication = class(TBrowserApplication)
  divpdfViewer,
  FTextlayer,
  lblFileLocation,
  lblZoom : TJSHTMLElement;
   // ...
end;
```

The **rendering of a page** of the PDF document was done in the `RenderPage` method of our program: in this method, the `GetPage` call is used to retrieve a **proxy object** for a PDF page, and the render method is used to actual render the page in the canvas:
In computer networking, a proxy server is a server application that acts as an intermediary between a client requesting a resource and the server providing that resource.

```
procedure TMyApplication.renderPage(aNum: Integer);

function renderOK(aValue : JSValue) : JSValue;

Var
   N : Integer;
begin
   FPageRendering:=false;
   if (FPageNumPending <> -1) then
      begin
        N:=FPageNumPending;
        FpageNumPending:=-1;
        renderPage(N);
      end;
   Result:=True;
end;

function havePage (aValue : JSValue) : JSValue;

var
   page : TPDFPageProxy absolute aValue;
   viewport : TPDFPageViewport;
   renderContext: TPDFRenderParams;
   renderTask : TPDFRenderTask;
   viewportParams : TViewportParameters;

begin
   viewportParams:=TViewportParameters.new;
   viewportParams.scale:=FScale;
   viewport:=page.getViewport(viewportParams);
   Fcanvas.height := viewport.height;
   Fcanvas.width := viewport.width;
   renderContext:=TPDFRenderParams.New;
   renderContext.canvasContext:=Fctx;
   renderContext.viewport:=viewport;
   renderTask:=page.render(renderContext);
   renderTask.promise.&then(@renderOK);
   Result:=True;
end;

begin
   FpageRendering:=True;
   pdfDoc.getPage(aNum).&then(@HavePage);
   edtPageNo.Value:=IntToStr(anum);
end;
```

As you can see in the code of the havePage routine, the parameters for the PDF page render task are the same as the ones for the RenderTextLayer call. It makes therefore sense to put them in a record which is declared in our application class: (*See next page*)

Continuation 4

```
 TCurrentPageInfo = record
page : TPDFPageProxy;
viewport : TPDFPageViewport;
renderContext: TPDFRenderParams;
renderTask : TPDFRenderTask;
viewportParams : TViewportParameters;
end;
TMyApplication = class(TBrowserApplication)
divpdfViewer,
FTextlayer,
lblFileLocation,
lblZoom : TJSHTMLElement;
FCurrentPageInfo : TCurrentPageInfo;
// ...
end;
```

We now can rewrite the havePage procedure so it uses the newly introduces record to store the information needed to render the page:

```
function havePage (aValue : JSValue) : JSValue;
var
page : TPDFPageProxy absolute aValue;
begin
FCurrentPageInfo.Page:=page;
FCurrentPageInfo.viewportParams:=TViewportParameters.new;
FCurrentPageInfo.viewportParams.scale:=FScale;
FCurrentPageInfo.viewport:=page.getViewport(FCurrentPageInfo.
viewportParams);
Fcanvas.height := FCurrentPageInfo.viewport.height;
Fcanvas.width := FCurrentPageInfo.viewport.width;
FCurrentPageInfo.renderContext:=TPDFRenderParams.New;
FCurrentPageInfo.renderContext.canvasContext:=Fctx;
FCurrentPageInfo.renderContext.viewport:=FCurrentPageInfo.viewport;
 FCurrentPageInfo.renderTask:=page.render(FCurrentPageInfo.
renderContext);
FCurrentPageInfo.renderTask.promise.&then(@renderOK);
Result:=True;
end;
```

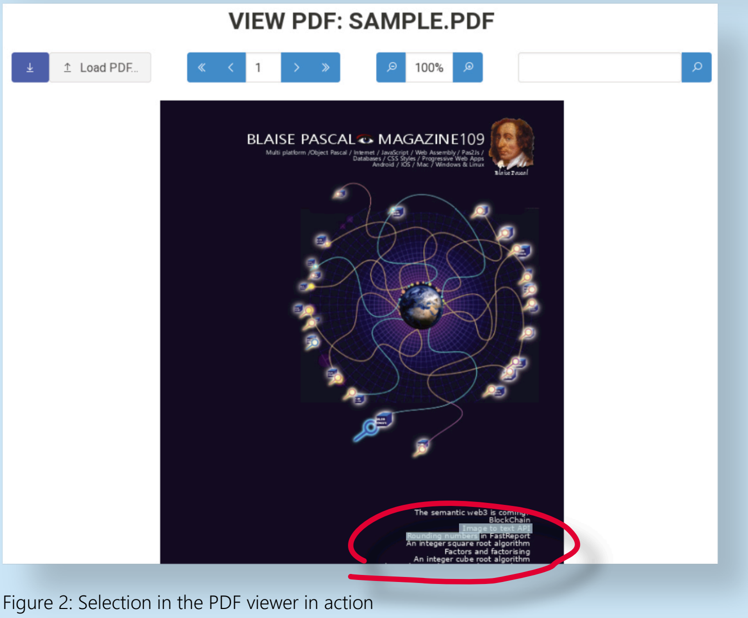
When the rendering is done, the `RenderOK` procedure is called, and there we can now add the necessary code to render the text. We start by getting the actual text using the `getTextContent` call. The observing reader will note that the `getTextContent` call is the same call that was used to retrieve the PDF text to search in the PDF.
The `renderOK` call then becomes:

```
function renderOK(aValue : JSValue) : JSValue;
Var
N : Integer;
begin
FPageRendering:=false;
if (FPageNumPending <> -1) then
begin
N:=FPageNumPending;
FpageNumPending:=-1;
renderPage(N);
end;
Result:=True;
FCurrentPageInfo.page.getTextContent().&Then(@RenderText);
divpdfViewer.style.setProperty('--scale-factor',FloatToStr(FScale))
end;
```

Continuation 4

The pdfViewer element gets a style property that sets a scale-factor CSS variable: this variable is needed by the CSS that is generated by **PDF.js**.
The value of the variable is set to the current scale used to draw the PDF.
Forgetting to set it (*or setting it to a wrong value*) will result in HTML that is not **properly positioned** over the PDF image.
When the text is retrieved, the `RenderText` callback which we supplied to the Javascript **Promise** object returned by `getTextContent`, will be called.
The callback simply prepares the arguments for the **`RenderTextLayer`** API call, using the `FCurrentPageInfo` and the result of the promise:

```
Function TMyApplication.RenderText(aValue: JSValue) : JSValue;
Var
  aContent : TTextContent absolute aValue;
  aTextRender : TPDFJSRenderTextLayerParameters;
begin
  FTextlayer.InnerHTML  :=”;
  aTextRender           :=TPDFJSRenderTextLayerParameters.New;
  aTextRender.container :=FTextlayer;
  aTextRender.isOffscreenCanvasSupported :=true;
  aTextRender.textContentSourceItems    :=aContent;
  aTextRender.viewPort  :=FCurrentPageInfo.viewport;
  pdfjsLib.renderTextLayer(aTextRender).promise.&then(@TextDone);
end;
```


VIEW PDF: SAMPLE.PDF
Load PDF...
«
‹
1
›
»
100%
BLAISE PASCAL MAGAZINE109
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux
Blaise Pascal

The semantic web3 is coming!
BlockChain
Image to text API
Rounding numbers in FastReport
An integer square root algorithm
Factors and factorising
An integer cube root algorithm

Figure 2: Selection in the PDF viewer in action

With the above code in place, the capability to select and copy text is almost there: the **necessary HTML is generated and positioned** on the correct place in the HTML page.
What is missing is some **CSS** that is required by the generated HTML.
The interested reader can find it in the viewer.css file, which we include in the HTML page with a simple link HTML element:

```
<link rel="stylesheet" href="viewer.css">
```

The result of all this can be seen in figure 2 on page 8: some text (*over multiple lines*) is selected in the overview of articles.
If the **scale of the PDF is changed**, then he PDF is **re-rendered,** and the text is re rendered as well: Since the scale **CSS** variable is set before rendering, the two will always be in sync.

❺ HIGHLIGHTING SEARCH MATCHES

What is still missing is the **highlighting of search results** once a **page has been rendered** as a result of a search operation. Adding this functionality is not so difficult:

In the `TextDone` callback to the `renderTextLayer` call, we have the opportunity to do so.
Looking at the generated HTML by PDF.js, we can see that it is simply a flat series of **span HTML elements** with some positioning applied:
the span elements only contain text.
To highlight the text, we can **simply loop over the span elements** and check if the span contains the text.
Basically, this is the same loop as done in the search algorithm, the difference is that now we loop over the generated HTML instead of the structures returned by the `getTextContent` call.
The following code checks whether a search was performed. If not, it exits at once.
If a search was performed, **it prepares the regular expression used for the search**,
and loops over all span tags, calling **`Highlight`** for every span element:

```
Function TMyApplication.TextDone(aValue: JSValue) : JSValue;
var
  El : TJSELement;
  aRegex : TJSRegExp;
  aReg,aTerm : String;
begin
  aTerm:=edtSearch.Value;
  if aTerm=” then exit;
  aReg:=Format(’%s’,[aTerm]);
  aRegEx:=TJSRegExp.New(aReg,’gi’);
  El:=FTextlayer.firstElementChild;
  While Assigned(El) do
    begin
      if (el is TJSHTMLElement) and SameText(El.tagName,’span’) then
        HighLight(TJSHTMLElement(El),aRegex);
      El:=El.nextElementSibling;
    end;
end;
```

In the **`highlight`** routine, we modify the inner HTML of the span element.
If a match (*or multiple matches*) is found, then we surround the match with a new span element which we give a **'highlight' CSS class:**
For example, if the search text is '`integer`' then the following span element:

```
<span>An integer square root algorithm</span>
```

becomes

```
<span>An <span class="highlight">integer</var> square root algorithm</span>
```

The '`highlight`' class is picked up by the **CSS to color the text background** in a transparent way, and the text in the canvas below will appear as highlighted.

This mechanism can be coded as follows:

It is in fact a **simple loop using the regular expression**:
as long as the regular expression finds a match, the text between the match and the end of the previous match is added to the span element as-is, and then the matched text is added embedded in a new span element.
If there are no more matches, the routine appends the remaining text.

```
Procedure TMyApplication.HighLight(El : TJSHTMLElement; aRegex : TJSRegexp);
Var
  S,aText, aLeft : String;
  Matches : TStringDynArray;
  aLast : Integer;
  aSpan : TJSHTMLElement;

begin
  aText:=El.innerText;
  Matches:=aRegex.exec(aText);
  aLast:=1;
  // We exit at once if there is nothing to do.
  if not Assigned(Matches) then
    exit;
  // Clear the HTML
  EL.InnerHTML:=’’;

  While Assigned(Matches) do
    begin
      // Add preceding text
      S:=Copy(aText,aLast,aRegex.lastIndex-Length(Matches[0]));
      // Create span.
      El.AppendChild(document.createTextNode(S));
      aSpan:=TJSHTMLElement(document.createElement(’span’));
      aSpan.InnerText:=Matches[0];
      aSpan.className:=’highlight’;
      El.AppendChild(aSpan);
      // Search again.
      aLast:=aRegex.LastIndex+1;
      Matches:=aRegex.exec(El.innerText);
    end;
    // Append last text.
    if aLast<length(aText) then
      begin
        S:=Copy(aText,aLast,Length(aText)-aRegex.lastIndex);
        El.AppendChild(document.createTextNode(S));
      end;
end;
```

With this, the highlighting is complete. The result can be seen in figure 3 on page 11.

❻ CONCLUSION

Adding a text layer to the **PDF.js** viewer allows to implement selection and copy and paste operations out of the box.

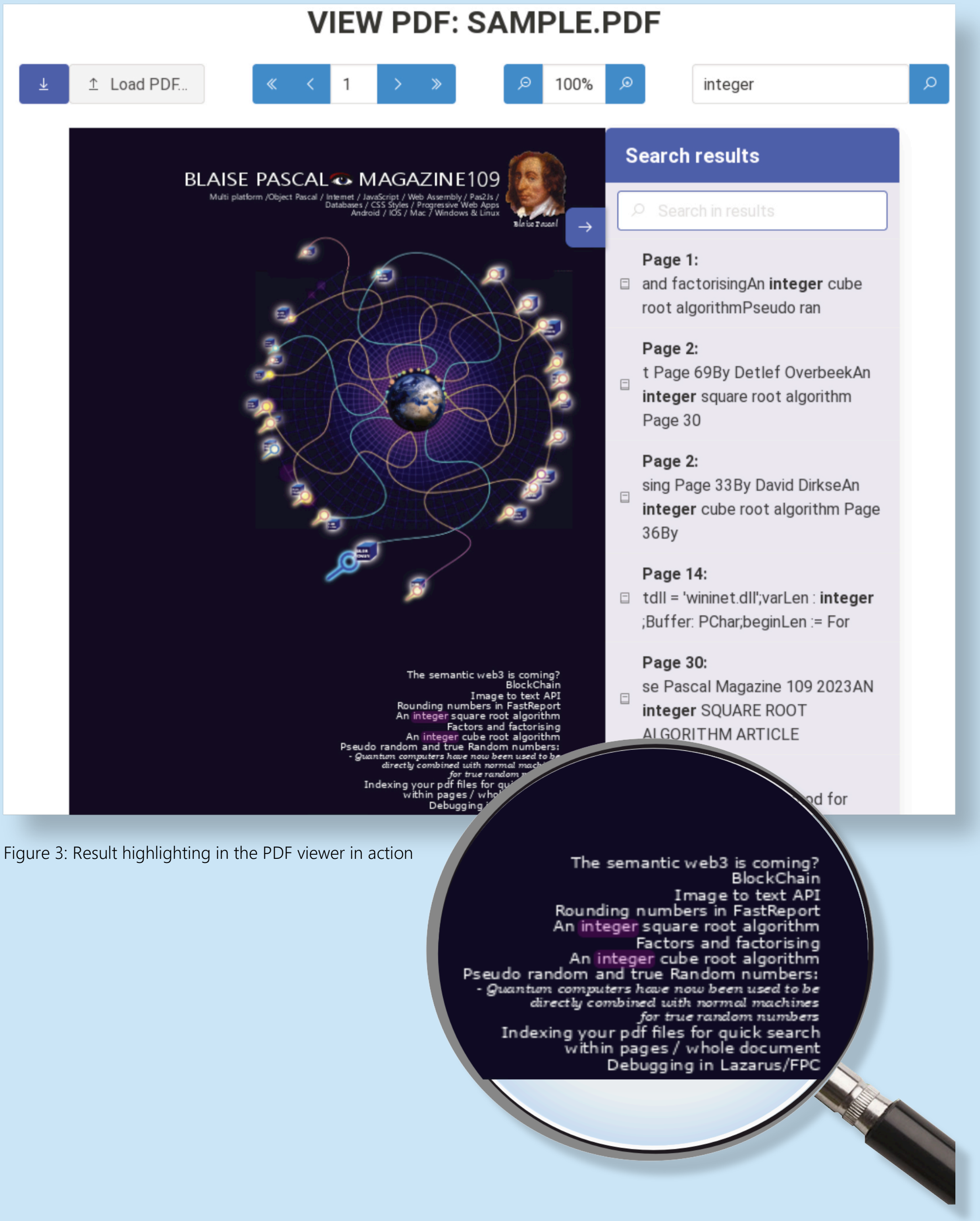
As shown in this article, it can also be used to implement in a rather straightforward manner the **highlighting of search terms** on a page.

The mechanism is not perfect, as the PDF text will sometimes be split into different PDF elements (*for example when formatting changes*):

The **HTML** will consequently be split over **HTML tags**.

For common situations the mechanism is perfectly suitable.

Example: see next page



VIEW PDF: SAMPLE.PDF

BLAISE PASCAL MAGAZINE 109

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

The semantic web3 is coming?
BlockChain
Image to text API
Rounding numbers in FastReport
An integer square root algorithm
Factors and factorising
An integer cube root algorithm
Pseudo random and true Random numbers:
- *Quantum computers have now been used to be directly combined with normal machines for true random numbers*
Indexing your pdf files for quick search within pages / whole document
Debugging in Lazarus/FPC

Search results

Page 1:
and factorisingAn **integer** cube root algorithmPseudo ran

Page 2:
t Page 69By Detlef OverbeekAn **integer** square root algorithm Page 30

Page 2:
sing Page 33By David DirkseAn **integer** cube root algorithm Page 36By

Page 14:
tdll = 'wininet.dll';varLen : **integer** ;Buffer: PChar;beginLen := For

Page 30:
se Pascal Magazine 109 2023AN **integer** SQUARE ROOT ALGORITHM ARTICLE

d for

Figure 3: Result highlighting in the PDF viewer in action

INSTALLATION OF LATEST VERSIONS OF FAST REPORTS UNDER LINUX - LAZARUS.

BY ALKEXANDER REDKOW

Starter Expert

INTRODUCTION

In Issue 110-111 of "Blaise Pascal Magazine" there was an ad about an update of **Fast Report** software and an article **"FASTREPORT FOR LAZARUS - LINUX"** by Sergey Plastun.

I am avid user of this soft since version "0",
went through all versions, maintain near a thousand of report forms.
Quite recently I had some interesting experience with FR's last iteration **FastreportVCL Pro 2023.2.**

The Advertisement said:
"One installation system with online authorization
- install and update all your products at once".

INSTALLING

Well, that sounds good.
But at work I have no connection to Internet (*and I think I am not alone*).
So an online installation per se is not for me.
Nevertheless I gave it a try at home.
I downloaded file "setup.exe" and started it.
After "**online authorization**" I see an empty form and didn't know what to do next.
So I went to tech support (*by the way, it's great and always pleasure to deal with - prompt, exact and very human*) and had an answer:
in order to proceed I have to have **Delphi** or **Lazarus** installed.

But I have **Lazarus** - as portable installation by **fpcupdeluxe** utility.
It's a must for me, because we are on a way from **Windows** to **Linux** and executable's for every application I am maintaining have to be of 3 kinds:
Win32, **Win64** and **Linux**.
So to tech support again.
On their advice I nevertheless installed Lazarus from a distribution (*handy opportunity to have a look at new 3.0RC1 release*) and setup went forward (*took near 2 hours, maybe my Internet connection was low*).
In the end I had a bunch of folders in the **Components** folder of **Lazarus**:

```
2023.3
    Sources
        LibLazarus 3.0RC1 (elazarus)
           FPC
             Win64
           Sources
             FastCore
             FastGraphics
             Localization
             FastReport
             FastScript
```

The rest was a piece of cake, just as advised in above mentioned article.
"Win64" folder contains all `.lpk` files we need.
I copied this folders to the **`fpcupdeluxe`** installation, compiled and installed packages
- and it worked!

But it worked for **Windows**.
And what about **Linux**?
I asked tech support about it and they said that they would ship a deb (*and rpm*) package for **Linux**. And they were true to their word (*version 2023.3*).
Now it is shipped along with the online installer.

May be as in previous versions (*without online installation*) it really it is not necessary if you already have the installation packages from a **Windows** installation.

Just copy them into **Linux** with **Lazarus** installation and execute the compile/install routine.
But now I have this package, so I downloaded and tried it.
All went well and fast.
I installed fast_report-professional-2023.3.0.deb (*we use the debian kind of Linux*) package with apt and got this folders:

```
/usr/share/FastReport-Professional
    Core
    Demos
    FastReport
    FastScript
    Graphics
    Localization
    Lpks
        Libs
        Res
        fr_lazarus.lpk
        frxchartlazarus.lpk
        frxe_lazarus.lpk
        frxlazdbf.lpk
        frxlazsqlite.lpk
        frxPDFlazarus.lpk
        frxrichlazarus.lpk
        fs_ibx.lpk
        fs_lazarus.lpk
```

As we see all `.lpk` files we need to compile and install are in the `/usr/share/FastReport-Professional/Lpks` folder.

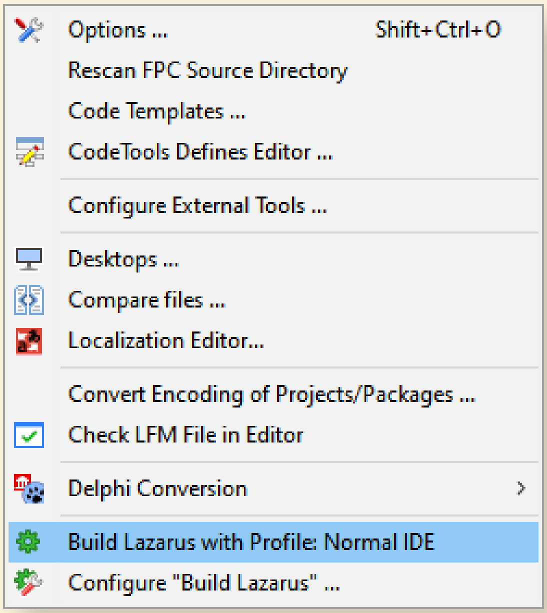
Lazarus under my non-privileged account refused to compile packages because of lack of write privileges in installation folder.
I could copy the installation folder to `Lazarus/Components` folder as I always did and then correct the ownership and privileges.

But I have a few of **Lazarus** installations for testing purposes.
So I instead started **Lazarus** under sudo.
After compiling and installing **FR** packages I started **Lazarus** under my my non-privileged account and all went well.

As for the article, I have some comments.

- The **Professional edition** has **client/server components**" is being suggested
  That's not true, only **Enterprise** and **Ultimate** have **client/server components.**

- Click "**Use**"" - is not enough, you have to click "**Install**" too.
- Installing `frxrichlazarus.lpk` requires previous installing of richmemo package.
  After installing each package, **Lazarus** will reboot - **Lazarus** has to not just reboot,
  it reboots after rebuilding itself and it takes quite a time.
  So after installing each package (*really, after marking each package for installation*)
  Lazarus demands: rebuild **Now or Not?**
  If yow answer "Not", rebuilding could only be done once after the last package was done


Options ... Shift+Ctrl+O
Rescan FPC Source Directory
Code Templates ...
CodeTools Defines Editor ...
Configure External Tools ...
Desktops ...
Compare files ...
Localization Editor...
Convert Encoding of Projects/Packages ...
Check LFM File in Editor
Delphi Conversion
Build Lazarus with Profile: Normal IDE
Configure "Build Lazarus" ...

And a last remark about the "**Online Installer**".
I think it would come in handy to have an option to manually add a **Lazarus** installation not registered in the system for such cases as **fpcupdeluxe** or **Code Typhon Studio**.

And now lets have some fun.

You buy it, you install it and it works

Some day I attended presentation of a new version of FR.
In the end was a kind of auction.
Attendees were to praise FR, FR director judged the utterance and gave (or not) reward.
Rewards included pens, flash sticks and T-shirts, all of cause with FR logo.
T-shirts were considered top prise and were given only once in a while.
When there was a kind of lull I said lazily (*not bothering to stand*):
"**I don't know why anyone should praise FastReport. Such a dull soft...**".

The Director went nearly apoplectic and croaked: "But why?!".
I explained: "If you take nearly any other software and try to make it work
- it's a fight, hard work.
And when after sleepless nights, liters of coffee, packs of cigarettes, endless chats you make a software work - it's victory, a bliss!

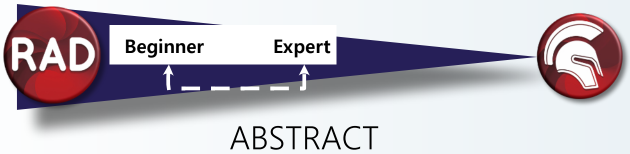
And now **FR**.
You buy it, you install it and it works.
Very dull.".

The Director nearly went apoplectic again, but found some strength and uttered:
"**A T-shirt for the user!**".
Now I somehow have overgrown this T-shirt but I still can show it.

WHAT IS NEW IN RAD STUDIO?

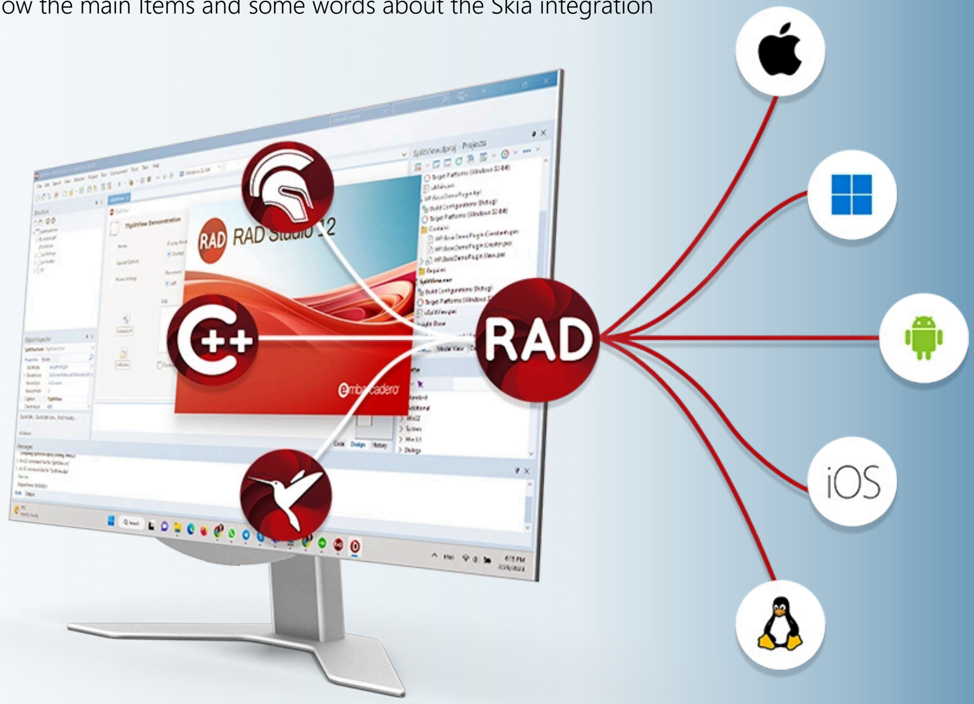
DELPHI 12 /. RAD STUDIO 12 HAS BEEN RELEASED

By Detlef Overbeek



ABSTRACT

This article is meant to give you an insight of the newest features in Delphi.
I will show the main Items and some words about the Skia integration



Platform Versions that Rad studio 12supports.
It offers official support for iOS 17 (*for Delphi only*), Android 14, and macOS Sonoma.
and also supports Ubuntu 22 LTS and Windows Server 2022.

Multiline String Literals
for Delphi Source Code. Multiline string literals enable easier embedding of SQL, HTML, JSON, XML multi-line text within an application source code.
Delphi’s Object Pascal permits string literals and static strings embedded in code.
Historically, these were limited to “short strings” with a 255-character cap.
With Delphi 12, that changes. Long String Literals: Delphi 12 now supports 4K characters per line.
Multiline Strings: Delphi 12 introduces multiline strings, enclosed by a triple quote (“’). This syntax makes it really easy to use multiline strings, without having to use the ‘+’ sign.

```
Begin
  Var MultilineString :=
  '''
     Long String Literals: Delphi 12 now supports 4K characters per line.
     Multiline Strings: Delphi 12 introduces multiline strings, enclosed by a triple quote (‘’’).
     This syntax makes it really easy to use multiline strings, without having to use the ‘+’ sign.
  '''
end.
```

SKIA Support for UI Design in FireMonkey

It improves performance and quality in rendering graphics and UI controls across all target platforms.
FireMonkey always used styles for UI rendering.
These styles determine the appearance and functionality of UI elements across various platforms, including DirectX and Metal.
Skia itself is recognized for its capability in 2D graphics applications.
The library, developed by Google, improves largely performance. The users of the **Skia4Delphi** library might be familiar with its features.

RAD Studio is interfacing with Skia by leveraging the **Skia4Delphi open-source project** but includes additional capabilities not found in the open-source projects, like the Vulkan driver.
It provides a comprehensive 2D API to render images across mobile, server, and desktop models.
It is compatible with all RAD Studio frameworks (Console, FMX, and VCL) and platforms.

It provides common **2D APIs** by abstracting complexities in implementing low-level libraries used behind, such as OpenGL, **Vulkan***(*See next page*), DirectX, or Metal, implementing optimizations and new features. See the Skia documentation for all the main features and how to enable Skia integration.

Skia-based UI Controls

The Skia library integration also offers some new specific native controls and components.
These controls are available only if Skia is enabled and deployed to the target platform.

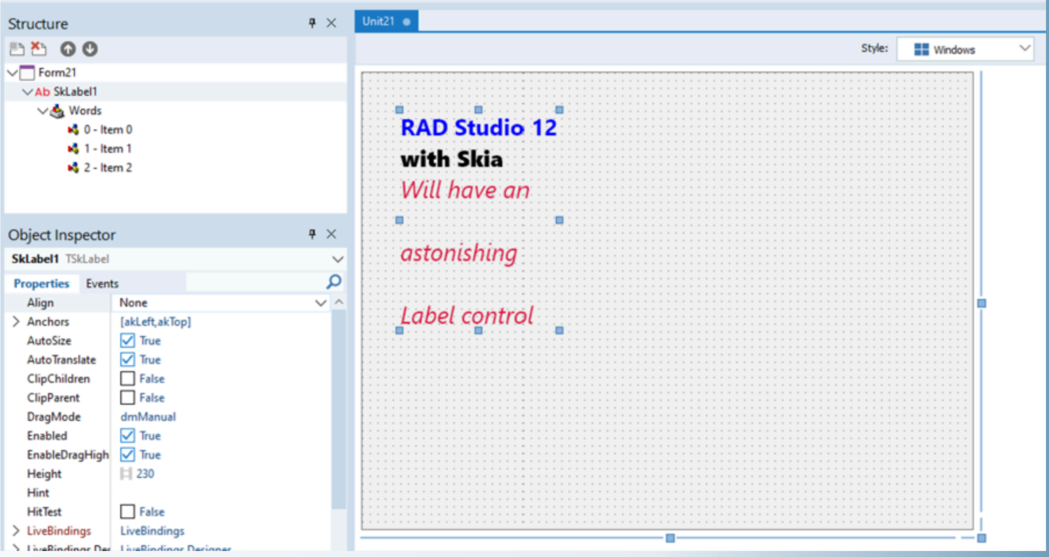
`TSkAnimatedImage:`

the control that loads and renders animated images, including **vector animations**.

`TSkLabel:`

implements new features that were either not supported by `TLabel` or were difficult to implement, such as:

Font families; (font fallback list like in CSS)
Font weight and slant;
Support for multiple styles in text;
Support for BiDi (Right-to-Left);
Support justify horizontal alignment;
Supports background color on parts of the text;
Auto size option;
Advanced decorations (underline wavy, overline, dashed line, and more).



Form21
- SkLabel1
  - Words
    - 0 - Item 0
    - 1 - Item 1
    - 2 - Item 2

SkLabel1 TSkLabel

Align	None
Anchors	[akLeft,akTop]
AutoSize	☑ True
AutoTranslate	☑ True
ClipChildren	□ False
ClipParent	□ False
DragMode	dmManual
Enabled	☑ True
EnableDragHigh	☑ True
Height	230
Hint	
HitTest	□ False
LiveBindings	LiveBindings
LiveBindings Des	LiveBindings Designer

RAD Studio 12
with Skia
Will have an

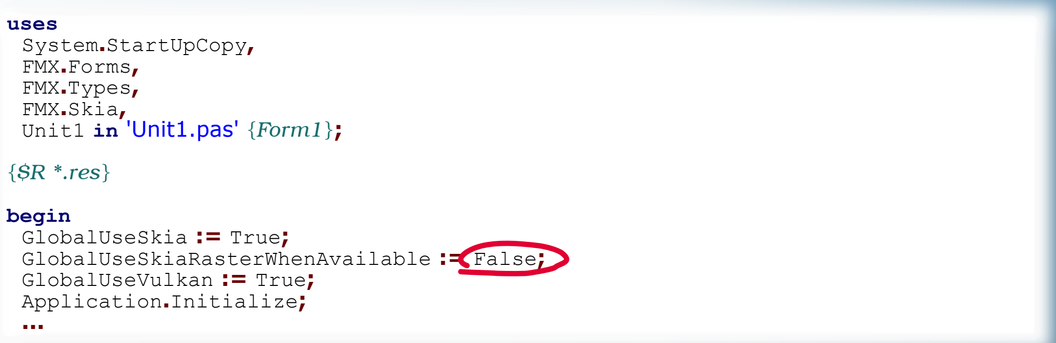
astonishing

Label control

SKIA

*** Vulkan Backend**

The Skia with RAD Studio offers **Vulkan backend support** for Android and, optionally, for Windows. **Skia Canvas** with RAD Studio automatically utilizes Vulkan on Android if supported, resulting in enhanced graphical performance and energy efficiency compared to OpenGLES.
To enable Vulkan on Windows when supported, set the boolean `FMX.Types.GlobalUseVulkan` to `True` and the boolean `FMX.Skia.GlobalUseSkiaRasterWhenAvailable` to `False` in the initialization section.



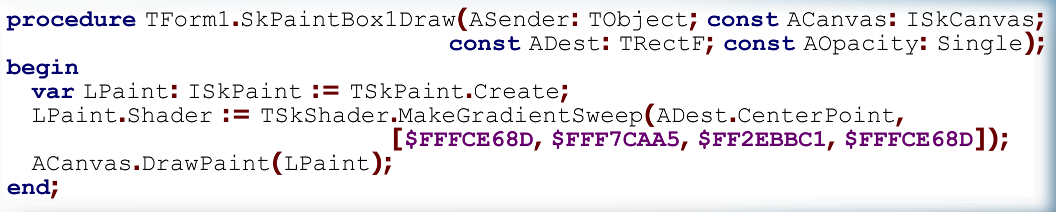
```
uses
  System.StartUpCopy,
  FMX.Forms,
  FMX.Types,
  FMX.Skia,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  GlobalUseSkia := True;
  GlobalUseSkiaRasterWhenAvailable := False;
  GlobalUseVulkan := True;
  Application.Initialize;
  ...
```

NOTE: The preference for the Vulkan backend on Android can be disabled using the same boolean GlobalUseVulkan employed for enabling it on Windows, but it should be set to `False` to deactivate.

`TSkPaintBox:`
control for painting with Skia APIs directly on the screen with the `OnDraw` event.

`TSaAnimatedPaintBox`:
allows setting the duration of an animation and drawing the progress of this animation using Skia APIs through the `OnAnimationDraw` event.

`TSkSVG:` control to display SVG.



```
procedure TForm1.SkPaintBox1Draw(ASender: TObject; const ACanvas: ISkCanvas;
                                 const ADest: TRectF; const AOpacity: Single);
begin
  var LPaint: ISkPaint := TSkPaint.Create;
  LPaint.Shader := TSkShader.MakeGradientSweep(ADest.CenterPoint,
                              [$FFFCE68D, $FFF7CAA5, $FF2EBBC1, $FFFCE68D]);
  ACanvas.DrawPaint(LPaint);
end;
```

SKIA DEMOS

The following Skia demos are included with the installation.

Skia Demo FMX - RAD Studio
`https://docwiki.embarcadero.com/RADStudio/Athens/en/Skia_Demo_FMX`
Location
You can find the Skia Demo FMX sample project at:
Start > Programs > Embarcadero RAD Studio 12 > Samples and navigate to:

- `Object Pascal\Multi-Device Samples\Skia4Delphi`
- `CPP\Multi-Device Samples\Skia4Delphi`

SKIA

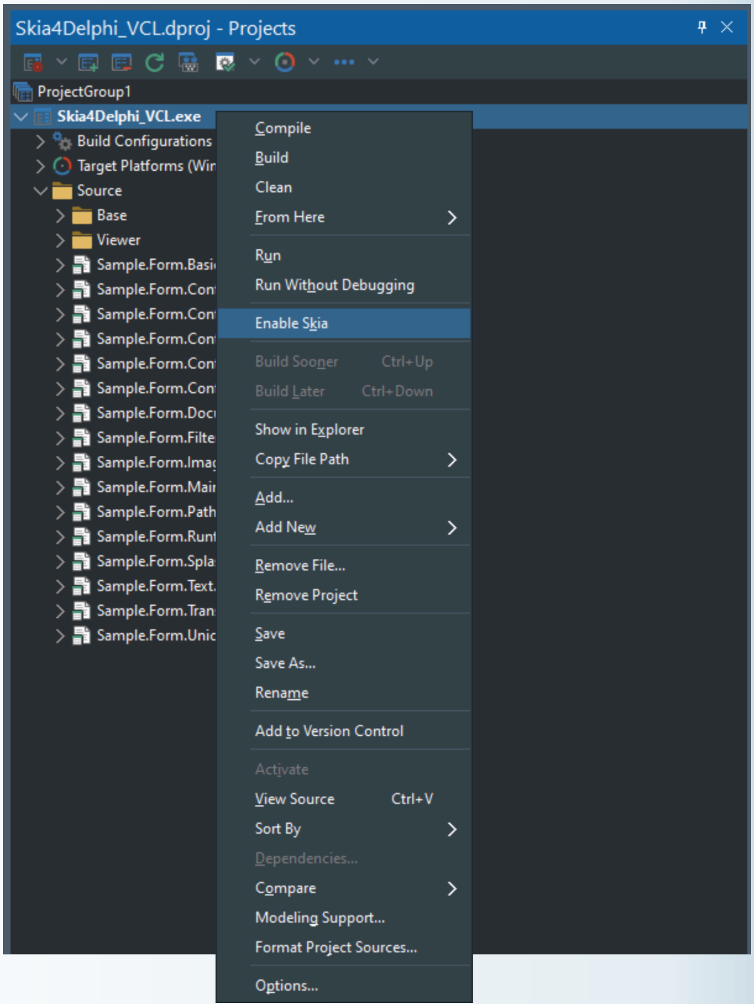
Skia4Delphi - RAD Studio

→ Go Up to Third Party Software Add-Ins: (See the standard path below)

```
C:\Users\Public\Documents\Embarcadero\Studio\23.0\Samples\Object Pascal\VCL\Skia4Delphi
```

Although the library is compatible with all frameworks, some features are unique:

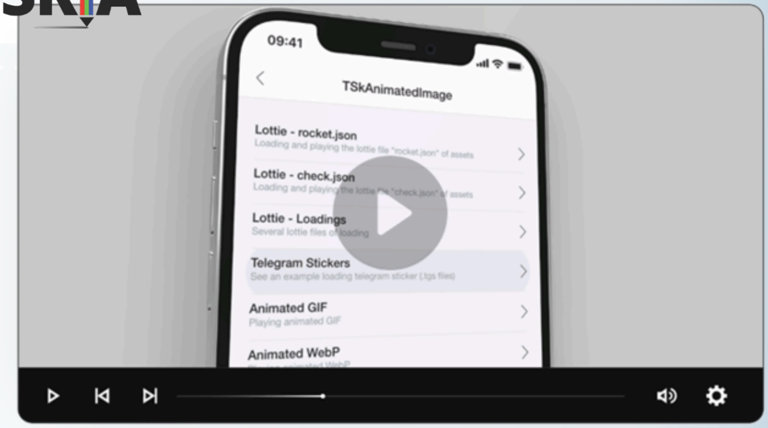
Feature set	**Console**	**VCL**	**FMX**	**Description**
Skia API	✓	✓	✓	Access to the pure Skia library, through a single unit: System.Skia.pas (*or System.Skia.hpp*)
Controls		✓	✓	TSkAnimatedImage, TSkLabel, TSkPaintBox and TSkSvg
App render			✓	Replacement of FMX graphics engine by Skia



Enabling Skia

Before using any Skia feature or unit, **you must enable your project to use Skia.** To do this, open your project in RAD Studio and right-click on your project and click Enable Skia:

This enabling is only done for applications. Packages using Skia do not have to make this enabling. This step automatically configures the links and deployments of the library binaries to the project. By not enabling Skia and using its units, your application will have issues at startup.



09:41

TSkAnimatedImage

Lottie - rocket.json
Loading and playing the lottie file "rocket.json" of assets

Lottie - check.json
Loading and playing the lottie file "check.json" of assets

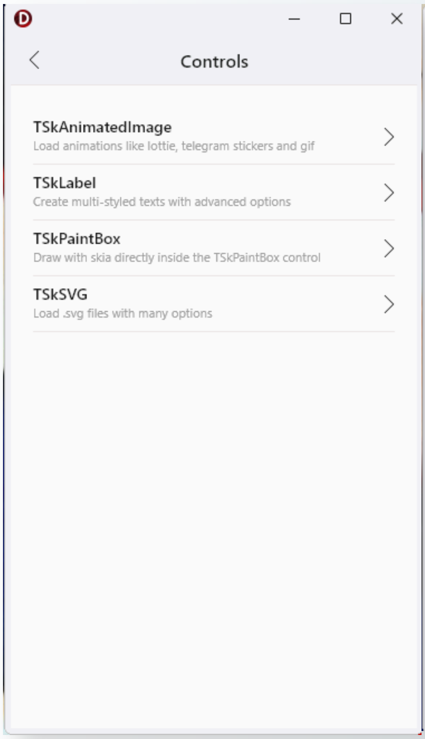
Lottie - Loadings
Several lottie files of loading

Telegram Stickers
See an example loading telegram sticker (.tgs files)

Animated GIF
Playing animated GIF

Animated WebP

The standard application to show the examples



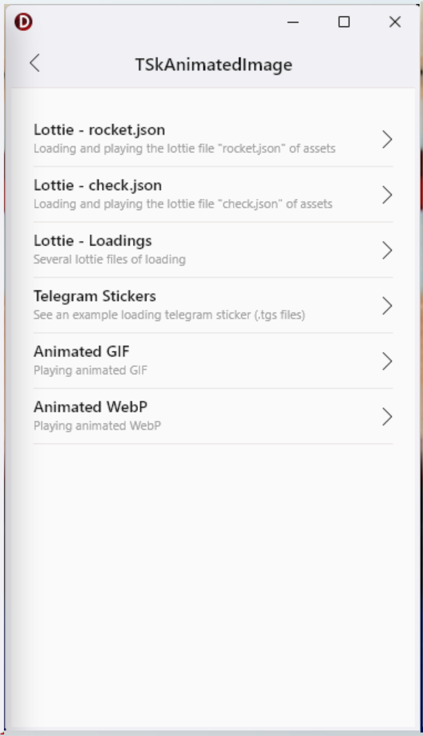
Controls

TSkAnimatedImage
Load animations like lottie, telegram stickers and gif

TSkLabel
Create multi-styled texts with advanced options

TSkPaintBox
Draw with skia directly inside the TSkPaintBox control

TSkSVG
Load .svg files with many options



TSkAnimatedImage

Lottie - rocket.json
Loading and playing the lottie file "rocket.json" of assets

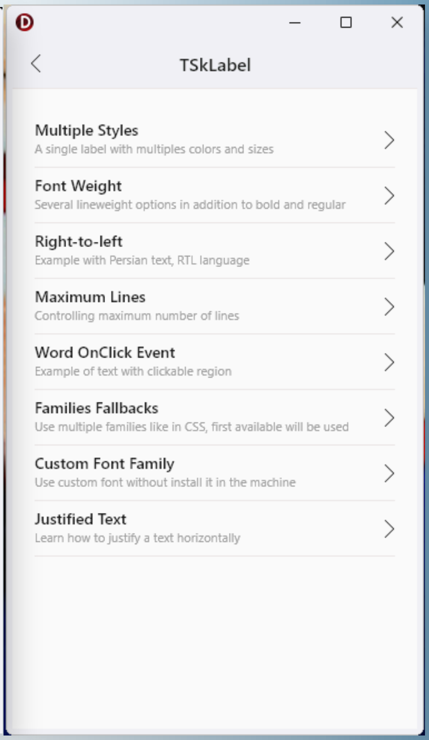
Lottie - check.json
Loading and playing the lottie file "check.json" of assets

Lottie - Loadings
Several lottie files of loading

Telegram Stickers
See an example loading telegram sticker (.tgs files)

Animated GIF
Playing animated GIF

Animated WebP
Playing animated WebP



TSkLabel

Multiple Styles
A single label with multiples colors and sizes

Font Weight
Several lineweight options in addition to bold and regular

Right-to-left
Example with Persian text, RTL language

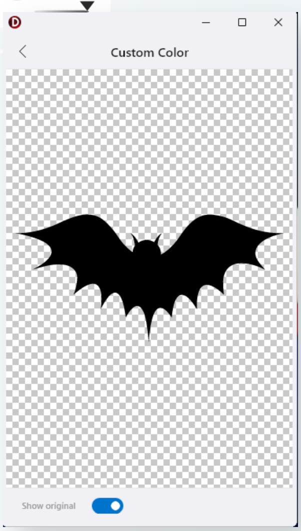
Maximum Lines
Controlling maximum number of lines

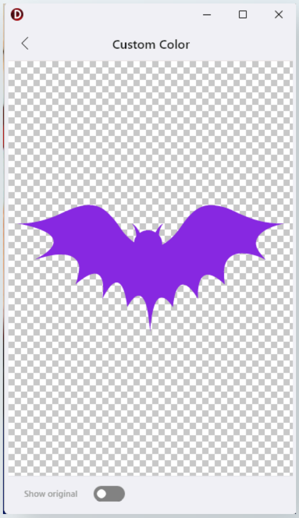
Word OnClick Event
Example of text with clickable region

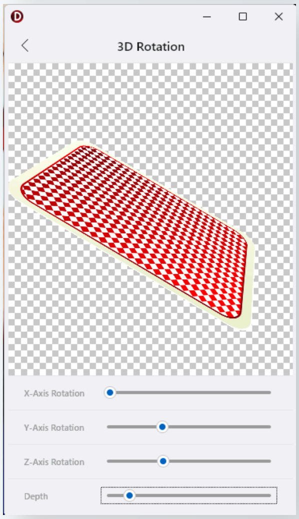
Families Fallbacks
Use multiple families like in CSS, first available will be used

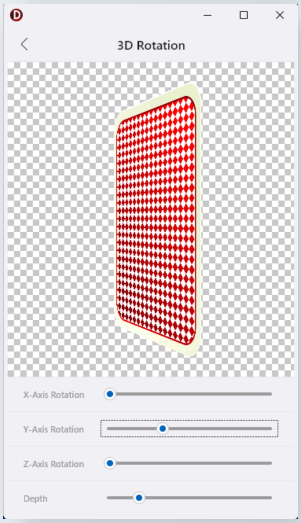
Custom Font Family
Use custom font without install it in the machine

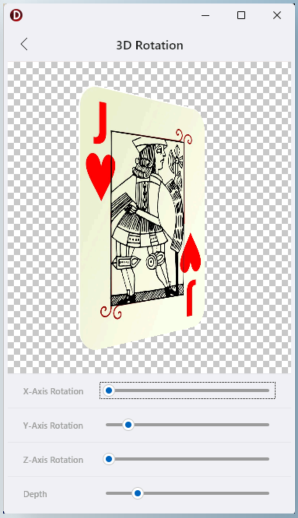
Justified Text
Learn how to justify a text horizontally

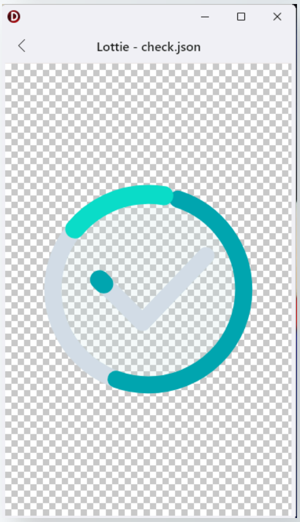


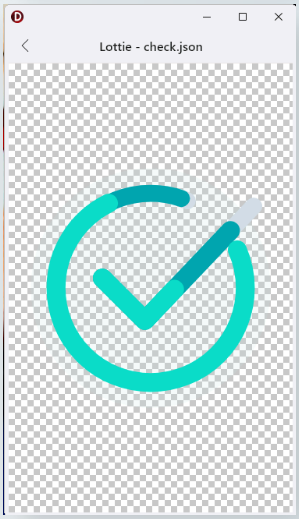




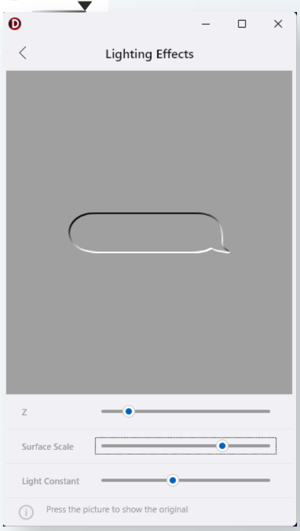








You can imagine any additional enhancement of your user interface for your app.
In the next issue I will create some extra examples to let you have some free code example to use.

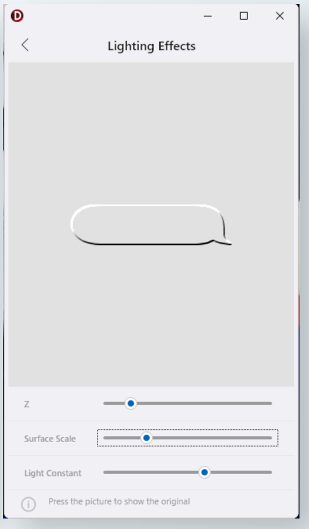

Lighting Effects

Z

Surface Scale

Light Constant

Press the picture to show the original

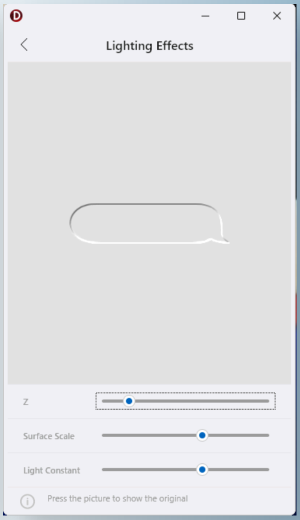

Lighting Effects

Z

Surface Scale

Light Constant

Press the picture to show the original

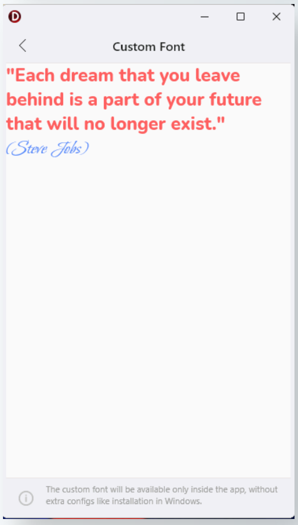

Lighting Effects

Z

Surface Scale

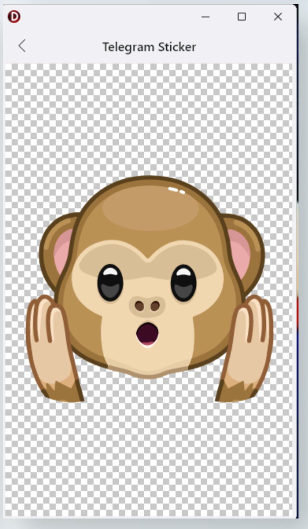
Light Constant

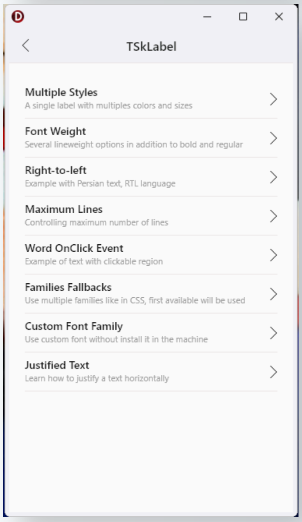
Press the picture to show the original


Custom Font

"Each dream that you leave behind is a part of your future that will no longer exist."
(Steve Jobs)

The custom font will be available only inside the app, without extra configs like installation in Windows.


Telegram Sticker


TSkLabel

Multiple Styles
A single label with multiples colors and sizes

Font Weight
Several lineweight options in addition to bold and regular

Right-to-left
Example with Persian text, RTL language

Maximum Lines
Controlling maximum number of lines

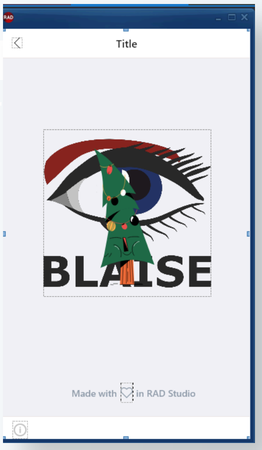
Word OnClick Event
Example of text with clickable region

Families Fallbacks
Use multiple families like in CSS, first available will be used

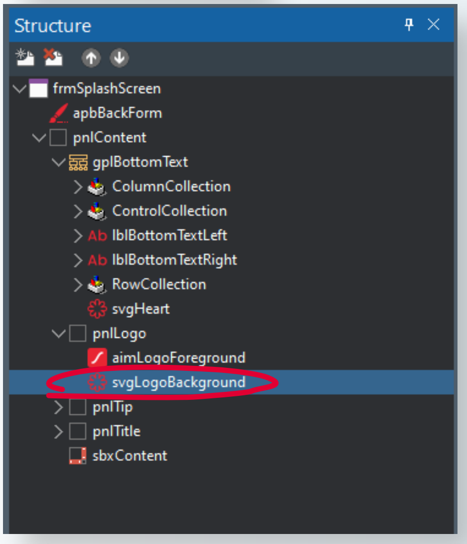
Custom Font Family
Use custom font without install it in the machine

Justified Text
Learn how to justify a text horizontally

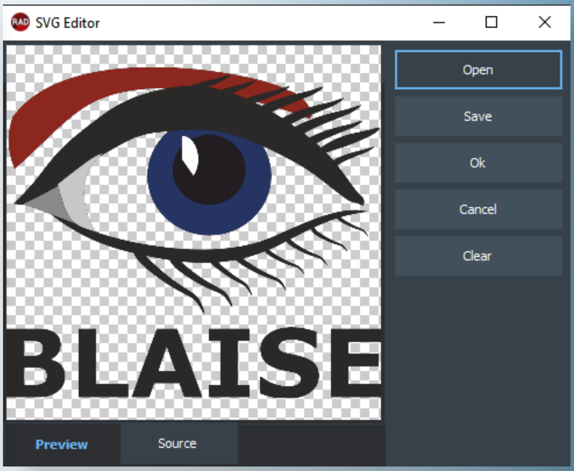
SKIA



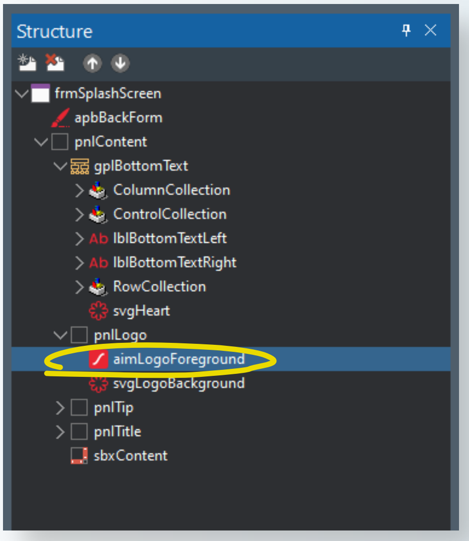
To give you a quick insight of what is going on inside the project it can be interesting to make a few changes and use them.
Without writing one piece of code I made my own splash screen and created a Christmas greeting.
Actually this splash consists of two main items:
a background and a moving foreground.
To make the **.svg** I had to do some trick.:
I made a bitmap out of the vector design of the eye and then transformed it into an **.svg** again. Otherwise the SVG would not be transformed with all the layers and coloring. I also changed the foreground movement with an existing **lottie** example.
I have put the example of the code on the *next two pages* so that you get an impression of the difficulty to handle this is shown.


Structure

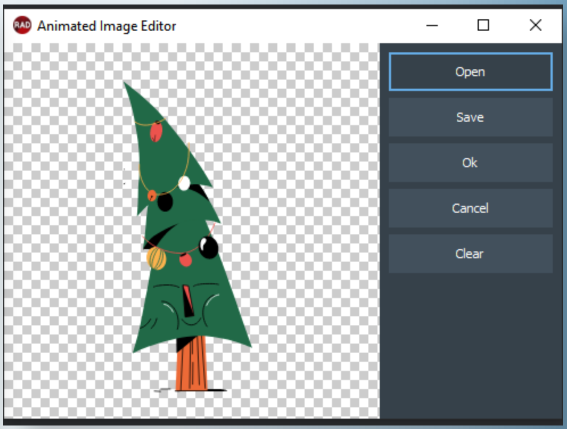
- frmSplashScreen
  - apbBackForm
  - pnlContent
    - gplBottomText
      - ColumnCollection
      - ControlCollection
      - lblBottomTextLeft
      - lblBottomTextRight
      - RowCollection
      - svgHeart
    - pnlLogo
      - aimLogoForeground
      - svgLogoBackground
    - pnlTip
    - pnlTitle
    - sbxContent


SVG Editor

Background SVG

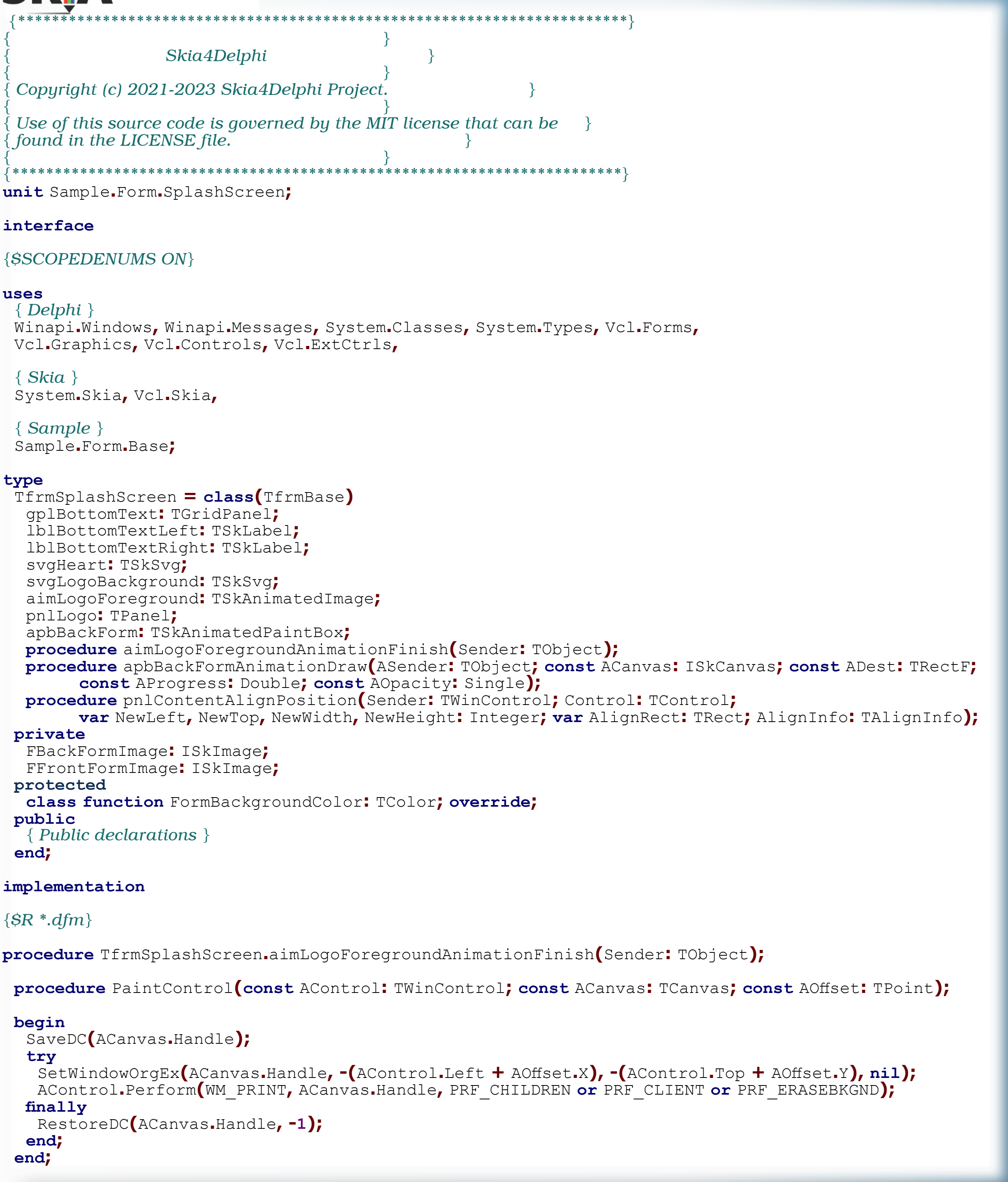

Structure

- frmSplashScreen
  - apbBackForm
  - pnlContent
    - gplBottomText
      - ColumnCollection
      - ControlCollection
      - lblBottomTextLeft
      - lblBottomTextRight
      - RowCollection
      - svgHeart
    - pnlLogo
      - aimLogoForeground
      - svgLogoBackground
    - pnlTip
    - pnlTitle
    - sbxContent


Animated Image Editor

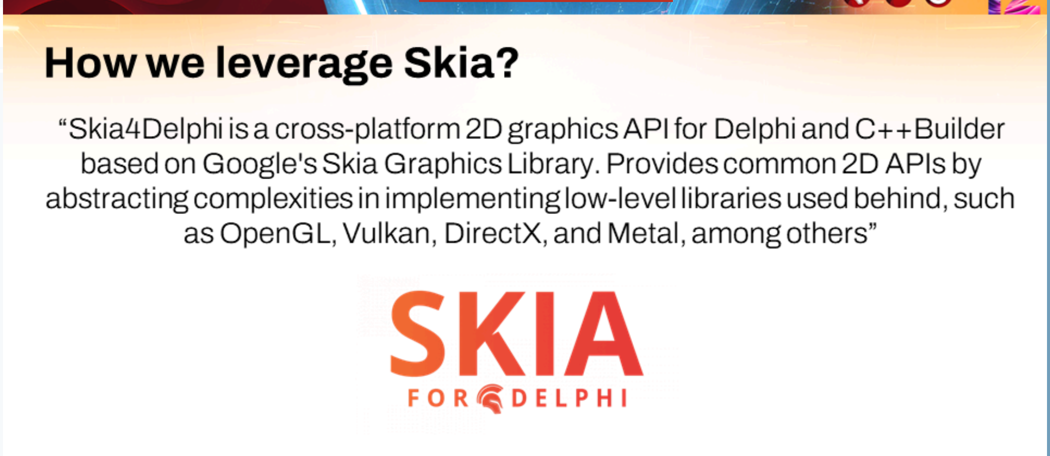
You can download your example from your personal downloads address.

SKIA



```
{************************************************************************}
{                                                                        }
{                           Skia4Delphi                                  }
{                                                                        }
{ Copyright (c) 2021-2023 Skia4Delphi Project.                           }
{                                                                        }
{ Use of this source code is governed by the MIT license that can be     }
{ found in the LICENSE file.                                             }
{                                                                        }
{************************************************************************}
unit Sample.Form.SplashScreen;

interface

{$SCOPEDENUMS ON}

uses
  { Delphi }
  Winapi.Windows, Winapi.Messages, System.Classes, System.Types, Vcl.Forms,
  Vcl.Graphics, Vcl.Controls, Vcl.ExtCtrls,

  { Skia }
  System.Skia, Vcl.Skia,

  { Sample }
  Sample.Form.Base;

type
  TfrmSplashScreen = class(TfrmBase)
    gplBottomText: TGridPanel;
    lblBottomTextLeft: TSkLabel;
    lblBottomTextRight: TSkLabel;
    svgHeart: TSkSvg;
    svgLogoBackground: TSkSvg;
    aimLogoForeground: TSkAnimatedImage;
    pnlLogo: TPanel;
    apbBackForm: TSkAnimatedPaintBox;
    procedure aimLogoForegroundAnimationFinish(Sender: TObject);
    procedure apbBackFormAnimationDraw(ASender: TObject; const ACanvas: ISkCanvas; const ADest: TRectF;
         const AProgress: Double; const AOpacity: Single);
    procedure pnlContentAlignPosition(Sender: TWinControl; Control: TControl;
         var NewLeft, NewTop, NewWidth, NewHeight: Integer; var AlignRect: TRect; AlignInfo: TAlignInfo);
  private
    FBackFormImage: ISkImage;
    FFrontFormImage: ISkImage;
  protected
    class function FormBackgroundColor: TColor; override;
  public
    { Public declarations }
  end;

implementation

{$R *.dfm}

procedure TfrmSplashScreen.aimLogoForegroundAnimationFinish(Sender: TObject);

  procedure PaintControl(const AControl: TWinControl; const ACanvas: TCanvas; const AOffset: TPoint);

  begin
    SaveDC(ACanvas.Handle);
    try
     SetWindowOrgEx(ACanvas.Handle, -(AControl.Left + AOffset.X), -(AControl.Top + AOffset.Y), nil);
     AControl.Perform(WM_PRINT, ACanvas.Handle, PRF_CHILDREN or PRF_CLIENT or PRF_ERASEBKGND);
    finally
     RestoreDC(ACanvas.Handle, -1);
    end;
  end;
```

```
 function MakeScreenshot(const AForm: TfrmBase): ISkImage;
 var
  LBitmap: TBitmap;
 begin
  LBitmap := TBitmap.Create;
  try
   LBitmap.SetSize(AForm.pnlContent.Width, AForm.pnlContent.Height);
   LBitmap.Canvas.Lock;
   try
    PaintControl(AForm.pnlContent, LBitmap.Canvas, Point(0, 0));
   finally
    LBitmap.Canvas.Unlock;
   end;
   Result := LBitmap.ToSkImage;
  finally
   LBitmap.Free;
  end;
 end;

begin
 FFrontFormImage := MakeScreenshot(Self);
 FBackFormImage := MakeScreenshot(Application.MainForm as TfrmBase);
 BeginUpdate;
 try
  apbBackForm.BoundsRect := Rect(0, 0, pnlContent.Width, pnlContent.Height);
  apbBackForm.Align := alClient;
  apbBackForm.Parent := pnlContent.Parent;
  pnlContent.Visible := False;
  apbBackForm.Visible := True;
 finally
  EndUpdate;
 end;
end;

procedure TfrmSplashScreen.apbBackFormAnimationDraw(ASender: TObject;
 const ACanvas: ISkCanvas; const ADest: TRectF; const AProgress: Double;
 const AOpacity: Single);
begin
 ACanvas.Save;
 try
  ACanvas.Scale(1 / TSkAnimatedPaintBox(ASender).ScaleFactor, 1 / TSkAnimatedPaintBox(ASender).
             ScaleFactor);
  ACanvas.DrawImage(FBackFormImage, 0, 0);
  ACanvas.SaveLayerAlpha(Round(255 * (1 - AProgress)));
  try
   ACanvas.DrawImage(FFrontFormImage, 0, 0);
  finally
   ACanvas.Restore;
  end;
 finally
  ACanvas.Restore;
 end;
end;

class function TfrmSplashScreen.FormBackgroundColor: TColor;
begin
 Result := FormBorderColor;
end;

procedure TfrmSplashScreen.pnlContentAlignPosition(Sender: TWinControl;
 Control: TControl; var NewLeft, NewTop, NewWidth, NewHeight: Integer;
 var AlignRect: TRect; AlignInfo: TAlignInfo);
begin
 inherited;
 if Control = pnlLogo then
 begin
  NewLeft := AlignRect.Left + ((AlignRect.Width - Control.Width) div 2);
  NewTop := AlignRect.Top + ((AlignRect.Height - Control.Height) div 2);
 end;
end;

end.
```

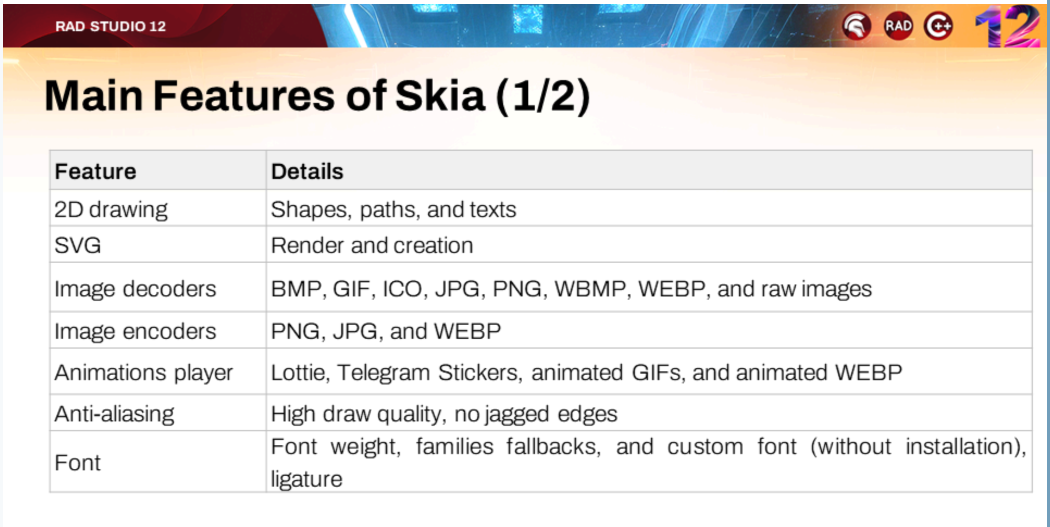


RAD STUDIO 12

How we leverage Skia?

"Skia4Delphi is a cross-platform 2D graphics API for Delphi and C++Builder based on Google's Skia Graphics Library. Provides common 2D APIs by abstracting complexities in implementing low-level libraries used behind, such as OpenGL, Vulkan, DirectX, and Metal, among others"

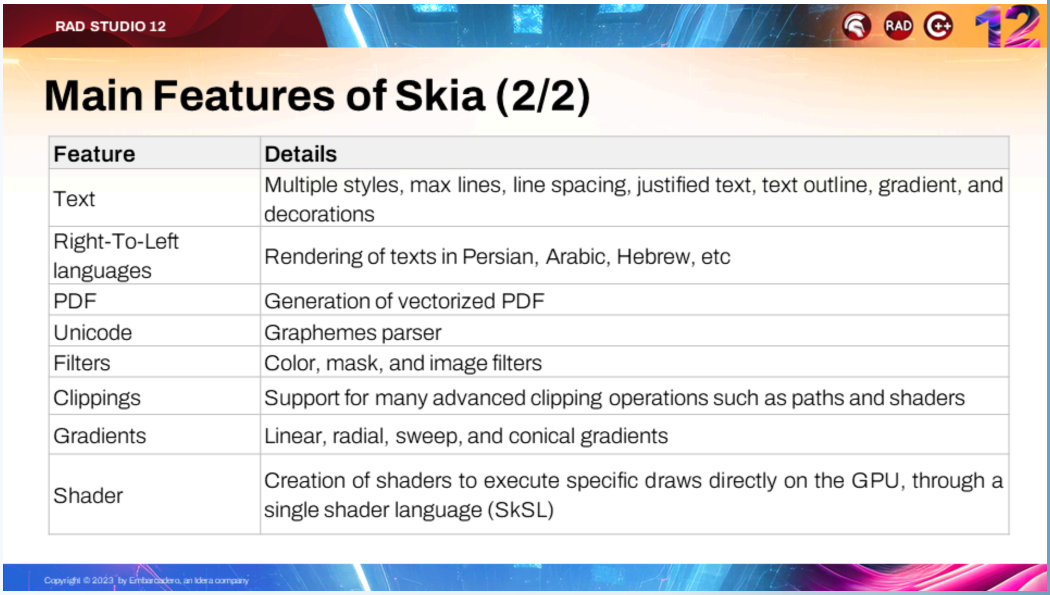
SKIA
FOR DELPHI



RAD STUDIO 12

Main Features of Skia (1/2)

Feature	Details
2D drawing	Shapes, paths, and texts
SVG	Render and creation
Image decoders	BMP, GIF, ICO, JPG, PNG, WBMP, WEBP, and raw images
Image encoders	PNG, JPG, and WEBP
Animations player	Lottie, Telegram Stickers, animated GIFs, and animated WEBP
Anti-aliasing	High draw quality, no jagged edges
Font	Font weight, families fallbacks, and custom font (without installation), ligature



RAD STUDIO 12

Main Features of Skia (2/2)

Feature	Details
Text	Multiple styles, max lines, line spacing, justified text, text outline, gradient, and decorations
Right-To-Left languages	Rendering of texts in Persian, Arabic, Hebrew, etc
PDF	Generation of vectorized PDF
Unicode	Graphemes parser
Filters	Color, mask, and image filters
Clippings	Support for many advanced clipping operations such as paths and shaders
Gradients	Linear, radial, sweep, and conical gradients
Shader	Creation of shaders to execute specific draws directly on the GPU, through a single shader language (SkSL)

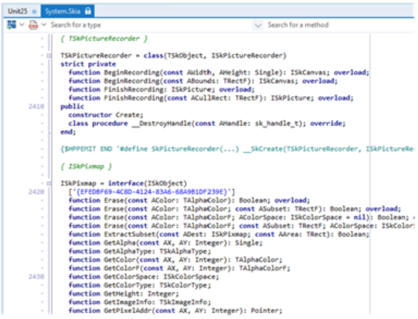
RAD STUDIO 12

Skia in RAD Studio

Different levels of usage for both Delphi and C++Builder

1. Skia API

Access to the pure Skia library, through a single unit: Skia.pas or Skia.hpp



```
{ TSkPictureRecorder }

TSkPictureRecorder = class(TSkObject, ISkPictureRecorder)
strict private
  function BeginRecording(const AWidth, AHeight: Single): ISkCanvas; overload;
  function BeginRecording(const ABounds: TRectF): ISkCanvas; overload;
  function FinishRecording: ISkPicture; overload;
  function FinishRecording(const ACullRect: TRectF): ISkPicture; overload;
public
  constructor Create;
  class procedure __DestroyHandle(const AHandle: sk_handle_t); override;
end;

{$HPPEMIT END '#define SkPictureRecorder(...) __SkCreate(TSkPictureRecorder, ISkPictureRe

{ ISkPixmap }

ISkPixmap = interface(ISkObject)
  ['{EFED8F69-4C8D-4124-83A6-68A9B1DF239E}']
  function Erase(const AColor: TAlphaColor): Boolean; overload;
  function Erase(const AColor: TAlphaColor; const ASubset: TRectF): Boolean; overload;
  function Erase(const AColor: TAlphaColorF; AColorSpace: ISkColorSpace = nil): Boolean; 
  function Erase(const AColor: TAlphaColorF; const ASubset: TRectF; AColorSpace: ISkColor
  function ExtractSubset(const ADest: ISkPixmap; const AArea: TRect): Boolean;
  function GetAlpha(const AX, AY: Integer): Single;
  function GetAlphaType: TSkAlphaType;
  function GetColor(const AX, AY: Integer): TAlphaColor;
  function GetColorF(const AX, AY: Integer): TAlphaColorF;
  function GetColorSpace: ISkColorSpace;
  function GetColorType: TSkColorType;
  function GetHeight: Integer;
  function GetImageInfo: TSkImageInfo;
  function GetPixelAddr(const AX, AY: Integer): Pointer;
```

RAD STUDIO 12

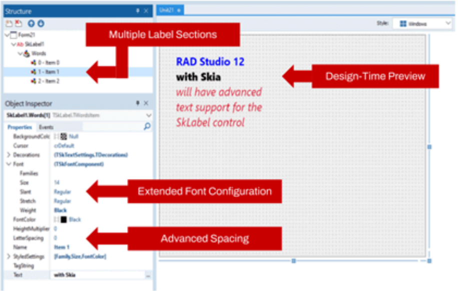
Skia in RAD Studio

Different levels of usage for both Delphi and C++Builder

2. UI Controls

(for both FMX and VCL)

- TSkAnimatedImage
- TSkLabel
- TSkPaintBox
- TSkSvg



Multiple Label Sections

Design-Time Preview

Extended Font Configuration

Advanced Spacing

RAD Studio 12
with Skia
will have advanced text support for the SkLabel control

BackgroundColor Null
Cursor crDefault
Decorations (TSkTextSettings.TDecorations)
Font (TSkFontComponent)
Families
Size 14
Slant Regular
Stretch Regular
Weight Black
FontColor Black
HeightMultiplier 0
LetterSpacing 0
Name Item 1
StyledSettings [Family,Size,FontColor]
TagString
Text with Skia

RAD STUDIO 12

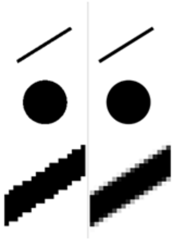
Skia in RAD Studio

Different levels of usage for both Delphi and C++Builder

3. App and Styles Rendering

Replacement of FMX graphic engine with Skia. Better performance, improved anti-aliasing

FMX.Skia.GlobalUseSkia := True



Improved drawing with anti-aliasing in Skia (on the right)

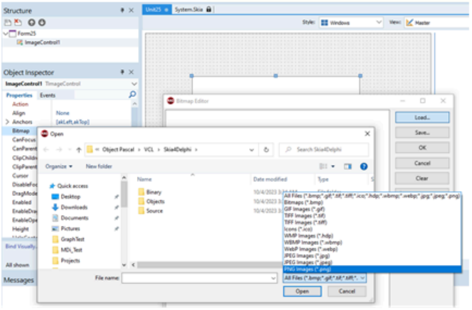
RAD STUDIO 12

Skia in RAD Studio

Different levels of usage for both Delphi and C++Builder

4. Codecs for image controls

New image codecs Skia supports are registered in the frameworks, FMX or VCL image controls can directly load and save new image formats like WebP



RAD STUDIO 12

New UI Controls (FMX/VCL) Based on Skia

- TSkAnimatedImage
  - Supports Lottie file, Telegram Sticker, Animated GIF, Animated WebP
- TSkLabel
  - Font weight, Font slant, multiple styles in text, BiDi (Right-to-Left)
  - Justify horizontal alignment, and much more
- TSkPaintBox & TSkAnimatedPaintBox
  - Paint with Skia APIs directly on the screen with the OnDraw event
- TSkSVG
  - Display SVG easily

RAD STUDIO 12

Skia and FireMonkey

Why Skia?

Skia is a multi-platform 2D graphics library

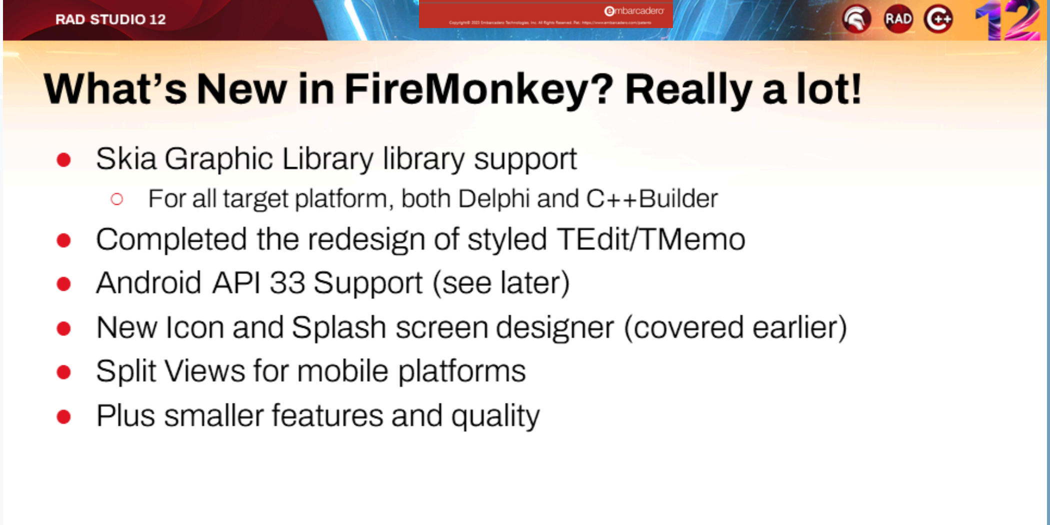
Skia offers high performance and high quality rendering

Skia supports advanced graphic operations across all platforms

Skia offers support for many image and video formats

Skia is becoming the new foundation for FireMonkey rendering

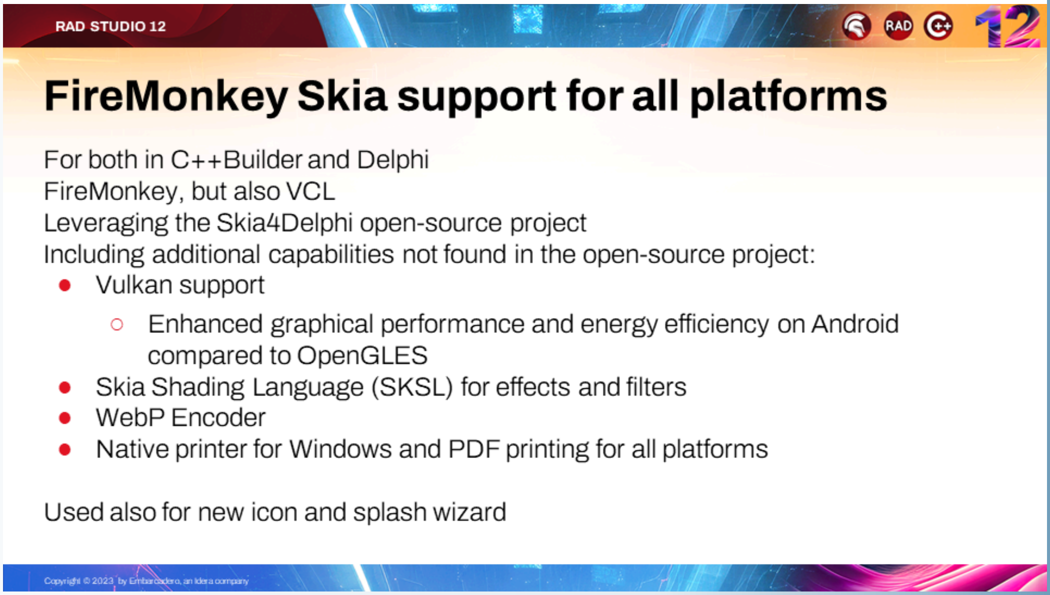
Skia logo source: https://en.wikipedia.org/wiki/Skia_Graphics_Engine



RAD STUDIO 12

What's New in FireMonkey? Really a lot!

- Skia Graphic Library library support
  - For all target platform, both Delphi and C++Builder
- Completed the redesign of styled TEdit/TMemo
- Android API 33 Support (see later)
- New Icon and Splash screen designer (covered earlier)
- Split Views for mobile platforms
- Plus smaller features and quality



RAD STUDIO 12

FireMonkey Skia support for all platforms

For both in C++Builder and Delphi
FireMonkey, but also VCL
Leveraging the Skia4Delphi open-source project
Including additional capabilities not found in the open-source project:

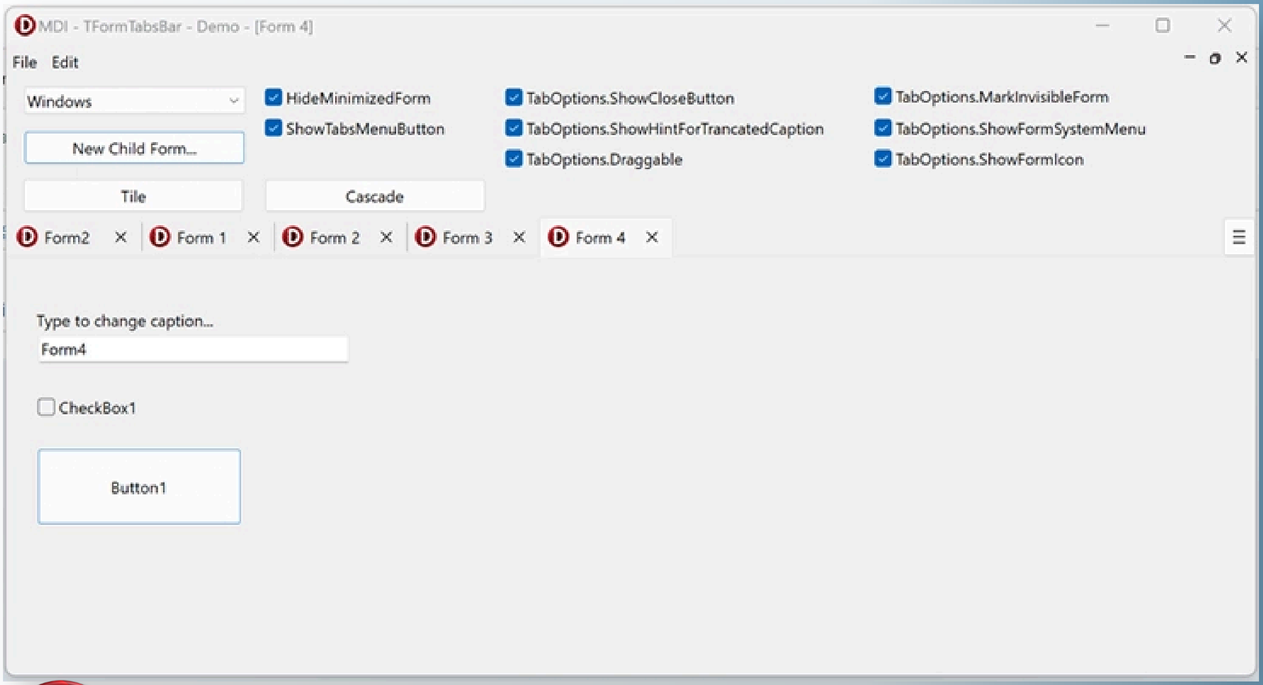
- Vulkan support
  - Enhanced graphical performance and energy efficiency on Android compared to OpenGLES
- Skia Shading Language (SKSL) for effects and filters
- WebP Encoder
- Native printer for Windows and PDF printing for all platforms

Used also for new icon and splash wizard

Copyright © 2023 by Embarcadero, an Idera company

Improved Mobile Design with FireMonkey Enhancements
Improvements to Android platform support, split-screen iOS and Android panes, full-set icon and splash screen wizard, support for Android API level 33

Modernized VCL with Reworked MDI and Tabbed UI for VCL
Improved application modernization with support for HighDPI and new VCL designers originating from Konopka Signature VCL Controls

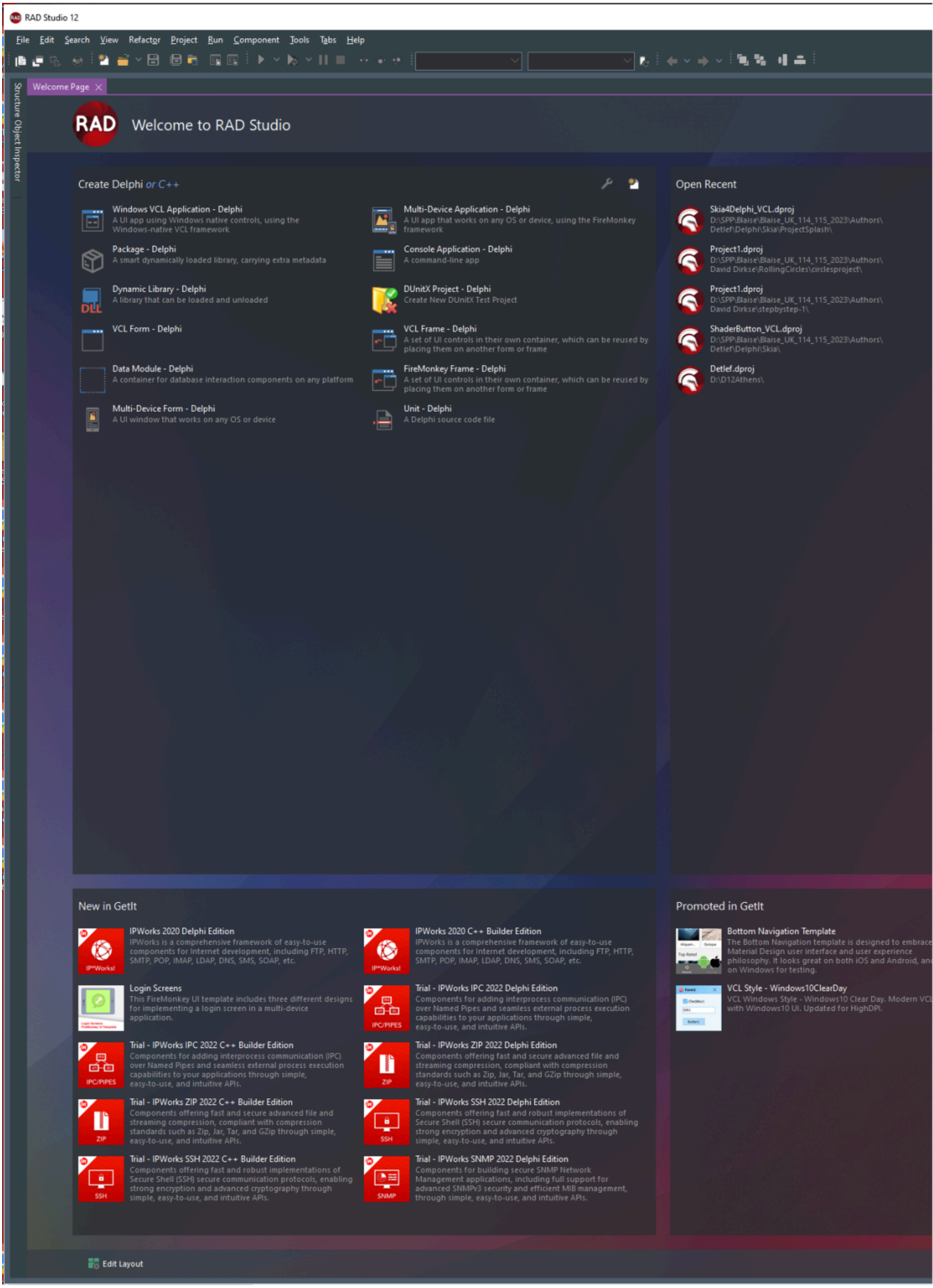


☑ HideMinimizedForm
☑ ShowTabsMenuButton
☑ TabOptions.ShowCloseButton
☑ TabOptions.ShowHintForTrancatedCaption
☑ TabOptions.Draggable
☑ TabOptions.MarkInvisibleForm
☑ TabOptions.ShowFormSystemMenu
☑ TabOptions.ShowFormIcon

Type to change caption...
Form4

□ CheckBox1

More Windows APIs Ready to Use in Object Pascal
Comprehensive set of all Windows API headers converted to Object Pascal, to make it easier for Delphi developers to call any Windows platform API



Welcome to RAD Studio

Create Delphi *or C++*

Windows VCL Application - Delphi
A UI app using Windows native controls, using the Windows-native VCL framework

Multi-Device Application - Delphi
A UI app that works on any OS or device, using the FireMonkey framework

Package - Delphi
A smart dynamically loaded library, carrying extra metadata

Console Application - Delphi
A command-line app

Dynamic Library - Delphi
A library that can be loaded and unloaded

DUnitX Project - Delphi
Create New DUnitX Test Project

VCL Form - Delphi

VCL Frame - Delphi
A set of UI controls in their own container, which can be reused by placing them on another form or frame

Data Module - Delphi
A container for database interaction components on any platform

FireMonkey Frame - Delphi
A set of UI controls in their own container, which can be reused by placing them on another form or frame

Multi-Device Form - Delphi
A UI window that works on any OS or device

Unit - Delphi
A Delphi source code file

Open Recent

Skia4Delphi_VCL.dproj
D:\SPP\Blaise\Blaise_UK_114_115_2023\Authors\Detlef\Delphi\Skia\ProjectSplash\

Project1.dproj
D:\SPP\Blaise\Blaise_UK_114_115_2023\Authors\David Dirkse\RollingCircles\circlesproject\

Project1.dproj
D:\SPP\Blaise\Blaise_UK_114_115_2023\Authors\David Dirkse\stepbystep-1\

ShaderButton_VCL.dproj
D:\SPP\Blaise\Blaise_UK_114_115_2023\Authors\Detlef\Delphi\Skia\

Detlef.dproj
D:\D12Athens\

New in GetIt

IPWorks 2020 Delphi Edition
IPWorks is a comprehensive framework of easy-to-use components for Internet development, including FTP, HTTP, SMTP, POP, IMAP, LDAP, DNS, SMS, SOAP, etc.

IPWorks 2020 C++ Builder Edition
IPWorks is a comprehensive framework of easy-to-use components for Internet development, including FTP, HTTP, SMTP, POP, IMAP, LDAP, DNS, SMS, SOAP, etc.

Login Screens
This FireMonkey UI template includes three different designs for implementing a login screen in a multi-device application.

Trial - IPWorks IPC 2022 Delphi Edition
Components for adding interprocess communication (IPC) over Named Pipes and seamless external process execution capabilities to your applications through simple, easy-to-use, and intuitive APIs.

Trial - IPWorks IPC 2022 C++ Builder Edition
Components for adding interprocess communication (IPC) over Named Pipes and seamless external process execution capabilities to your applications through simple, easy-to-use, and intuitive APIs.

Trial - IPWorks ZIP 2022 Delphi Edition
Components offering fast and secure advanced file and streaming compression, compliant with compression standards such as Zip, Jar, Tar, and GZip through simple, easy-to-use, and intuitive APIs.

Trial - IPWorks ZIP 2022 C++ Builder Edition
Components offering fast and secure advanced file and streaming compression, compliant with compression standards such as Zip, Jar, Tar, and GZip through simple, easy-to-use, and intuitive APIs.

Trial - IPWorks SSH 2022 Delphi Edition
Components offering fast and robust implementations of Secure Shell (SSH) secure communication protocols, enabling strong encryption and advanced cryptography through simple, easy-to-use, and intuitive APIs.

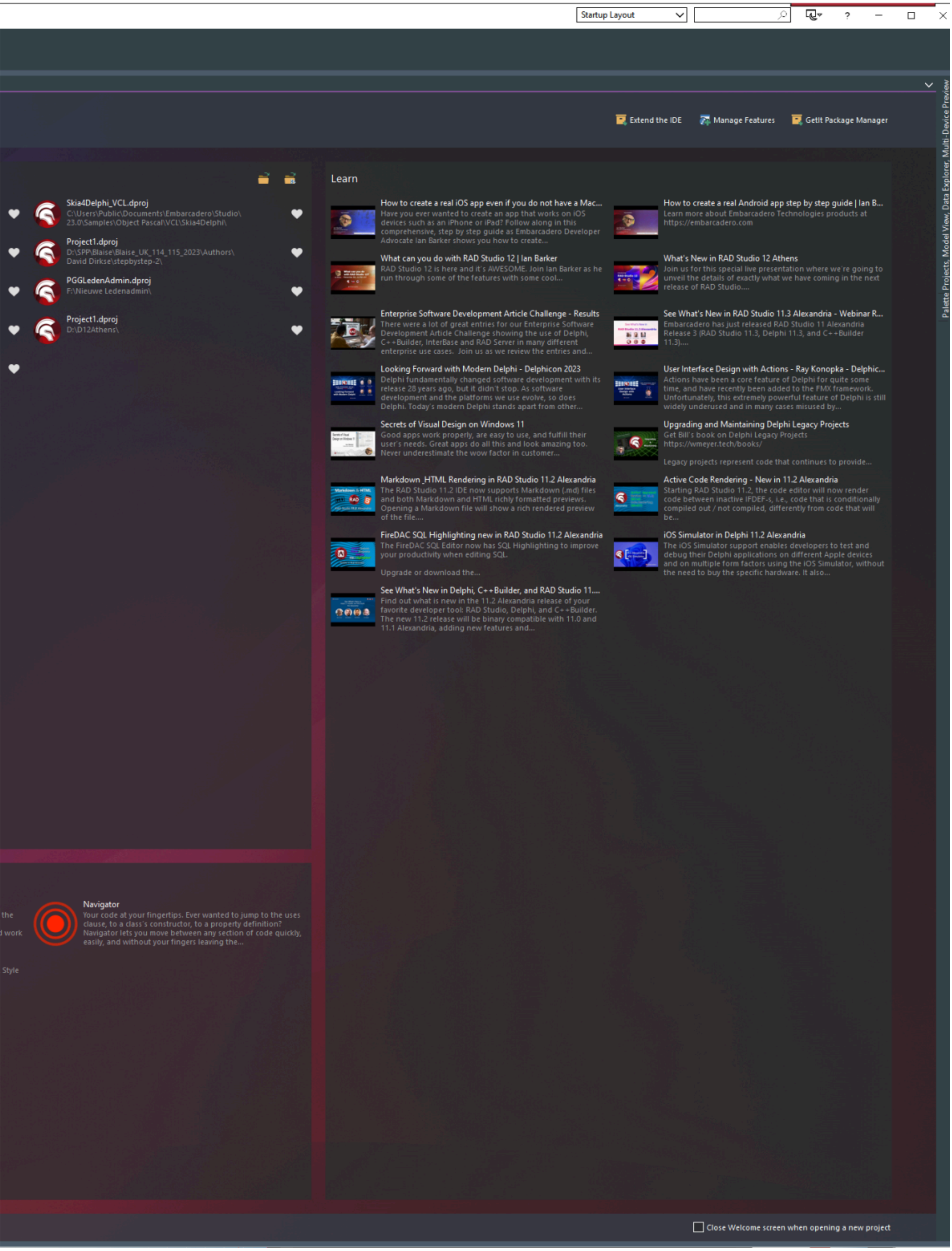
Trial - IPWorks SSH 2022 C++ Builder Edition
Components offering fast and robust implementations of Secure Shell (SSH) secure communication protocols, enabling strong encryption and advanced cryptography through simple, easy-to-use, and intuitive APIs.

Trial - IPWorks SNMP 2022 Delphi Edition
Components for building secure SNMP Network Management applications, including full support for advanced SNMPv3 security and efficient MIB management, through simple, easy-to-use, and intuitive APIs.

Promoted in GetIt

Bottom Navigation Template
The Bottom Navigation template is designed to embrace Material Design user interface and user experience philosophy. It looks great on both iOS and Android, and on Windows for testing.

VCL Style - Windows10ClearDay
VCL Windows Style - Windows10 Clear Day. Modern VCL with Windows10 UI. Updated for HighDPI.


Startup Layout
Palette Projects, Multi-Device Preview, Model View, Data Explorer
Extend the IDE
Manage Features
GetIt Package Manager

Skia4Delphi_VCL.dproj
C:\Users\Public\Documents\Embarcadero\Studio\23.0\Samples\Object Pascal\VCL\Skia4Delphi\

Project1.dproj
D:\SPP\Blaise\Blaise_UK_114_115_2023\Authors\David Dirkse\stepbystep-2\

PGGLedenAdmin.dproj
F:\Nieuwe Ledenadmin\

Project1.dproj
D:\D12Athens\

Learn

How to create a real iOS app even if you do not have a Mac...
Have you ever wanted to create an app that works on iOS devices such as an iPhone or iPad? Follow along in this comprehensive, step by step guide as Embarcadero Developer Advocate Ian Barker shows you how to create...

How to create a real Android app step by step guide | Ian B...
Learn more about Embarcadero Technologies products at https://embarcadero.com

What can you do with RAD Studio 12 | Ian Barker
RAD Studio 12 is here and it's AWESOME. Join Ian Barker as he run through some of the features with some cool...

What's New in RAD Studio 12 Athens
Join us for this special live presentation where we're going to unveil the details of exactly what we have coming in the next release of RAD Studio....

Enterprise Software Development Article Challenge - Results
There were a lot of great entries for our Enterprise Software Development Article Challenge showing the use of Delphi, C++Builder, InterBase and RAD Server in many different enterprise use cases. Join us as we review the entries and...

See What's New in RAD Studio 11.3 Alexandria - Webinar R...
Embarcadero has just released RAD Studio 11 Alexandria Release 3 (RAD Studio 11.3, Delphi 11.3, and C++Builder 11.3)....

Looking Forward with Modern Delphi - Delphicon 2023
Delphi fundamentally changed software development with its release 28 years ago, but it didn't stop. As software development and the platforms we use evolve, so does Delphi. Today's modern Delphi stands apart from other...

User Interface Design with Actions - Ray Konopka - Delphic...
Actions have been a core feature of Delphi for quite some time, and have recently been added to the FMX framework. Unfortunately, this extremely powerful feature of Delphi is still widely underused and in many cases misused by...

Secrets of Visual Design on Windows 11
Good apps work properly, are easy to use, and fulfill their user's needs. Great apps do all this and look amazing too. Never underestimate the wow factor in customer...

Upgrading and Maintaining Delphi Legacy Projects
Get Bill's book on Delphi Legacy Projects https://wmeyer.tech/books/

Legacy projects represent code that continues to provide...

Markdown_HTML Rendering in RAD Studio 11.2 Alexandria
The RAD Studio 11.2 IDE now supports Markdown (.md) files and both Markdown and HTML richly formatted previews. Opening a Markdown file will show a rich rendered preview of the file....

Active Code Rendering - New in 11.2 Alexandria
Starting RAD Studio 11.2, the code editor will now render code between inactive IFDEF-s, i.e., code that is conditionally compiled out / not compiled, differently from code that will be...

FireDAC SQL Highlighting new in RAD Studio 11.2 Alexandria
The FireDAC SQL Editor now has SQL Highlighting to improve your productivity when editing SQL.

Upgrade or download the....

iOS Simulator in Delphi 11.2 Alexandria
The iOS Simulator support enables developers to test and debug their Delphi applications on different Apple devices and on multiple form factors using the iOS Simulator, without the need to buy the specific hardware. It also...

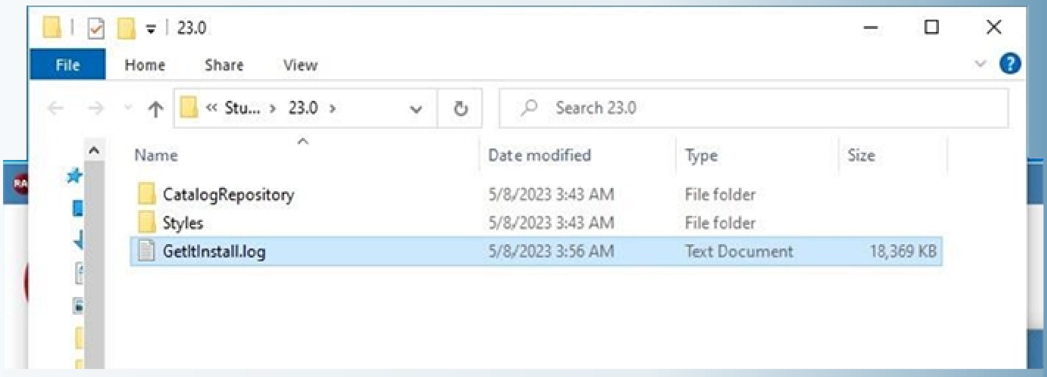
See What's New in Delphi, C++Builder, and RAD Studio 11....
Find out what is new in the 11.2 Alexandria release of your favorite developer tool: RAD Studio, Delphi, and C++Builder. The new 11.2 release will be binary compatible with 11.0 and 11.1 Alexandria, adding new features and...

the
d work
Style

Navigator
Your code at your fingertips. Ever wanted to jump to the uses clause, to a class's constructor, to a property definition? Navigator lets you move between any section of code quickly, easily, and without your fingers leaving the...

☐ Close Welcome screen when opening a new project

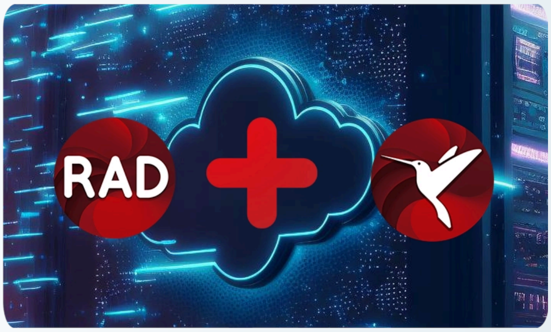
More Windows APIs Ready to Use in Object Pascal
Comprehensive set of all Windows API headers converted to Object Pascal, to make it easier for Delphi developers to call any Windows platform API



Name	Date modified	Type	Size
CatalogRepository	5/8/2023 3:43 AM	File folder	
Styles	5/8/2023 3:43 AM	File folder	
GetItInstall.log	5/8/2023 3:56 AM	Text Document	18,369 KB

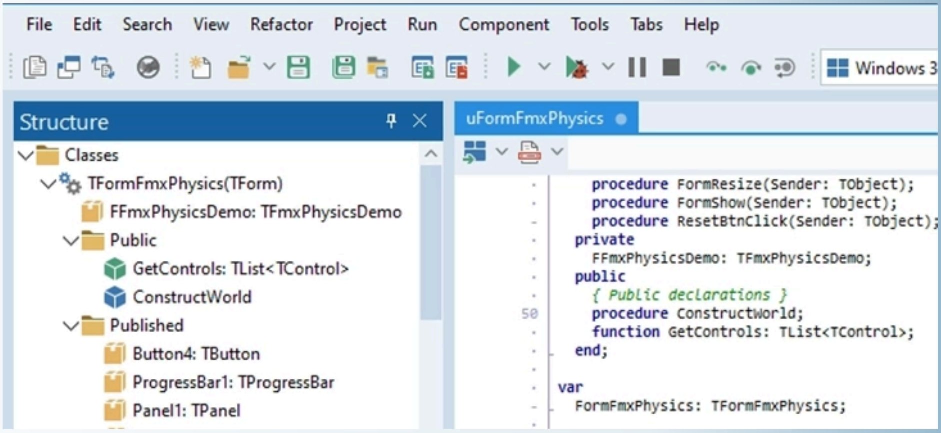
QBE Support in FireDAC,
New JSON Wizard for Delphi
Query-by-Example available in FireDAC. JSON data mapping wizard to generate classes matching JSON data structure, map data to objects like XML and stream out to new file

Use Biometric Authentication!
RAD Studio 12 offers a new Mobile Biometric Authentication component for FireMonkey mobile applications



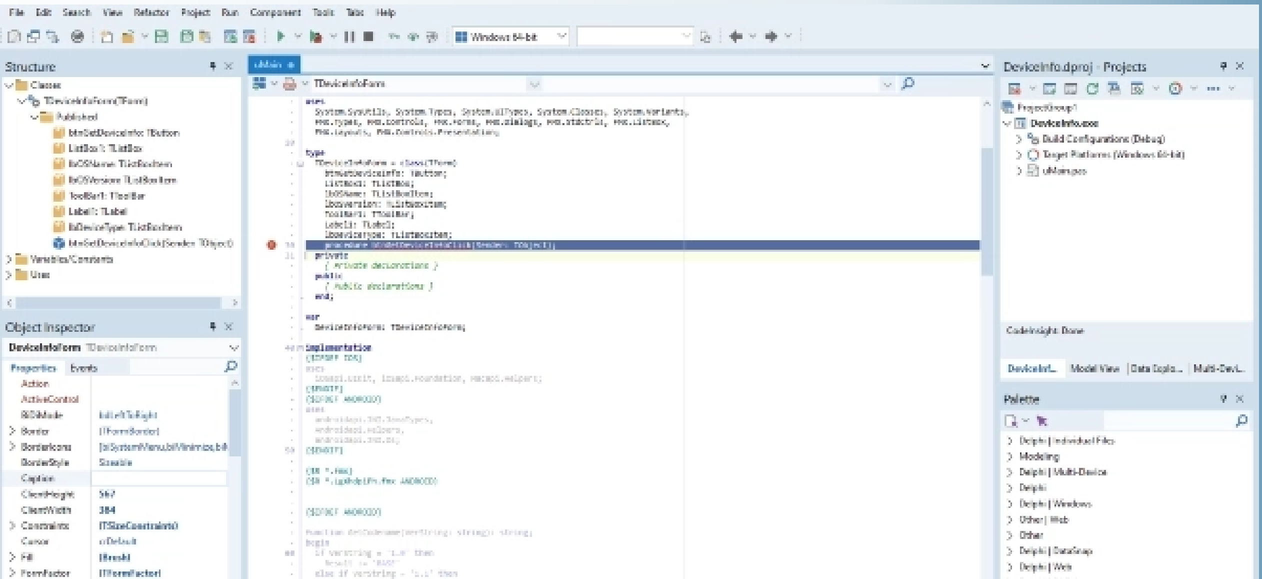
Deploy Embedded InterBase Dev Edition!
RAD Studio 12 ships with the recently released InterBase 2020 Update 5 Developer edition and IBLite/ToGo edition

Improved Application Security Through SQL Restrictions
Deeper application security through restrictions on SQL commands, blocks on multiple commands and SQL changes


File Edit Search View Refactor Project Run Component Tools Tabs Help
Windows 3
Structure
Classes
TFormFmxPhysics(TForm)
FFmxPhysicsDemo: TFmxPhysicsDemo
Public
GetControls: TList<TControl>
ConstructWorld
Published
Button4: TButton
ProgressBar1: TProgressBar
Panel1: TPanel
uFormFmxPhysics

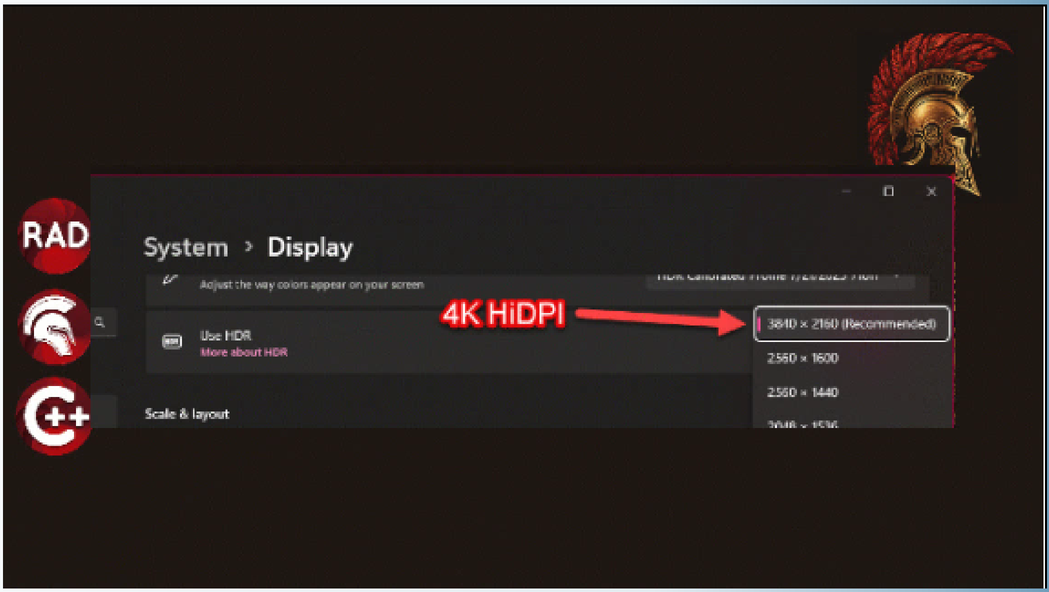
```
    procedure FormResize(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure ResetBtnClick(Sender: TObject);
  private
    FFmxPhysicsDemo: TFmxPhysicsDemo;
  public
    { Public declarations }
    procedure ConstructWorld;
    function GetControls: TList<TControl>;
  end;

var
  FormFmxPhysics: TFormFmxPhysics;
```

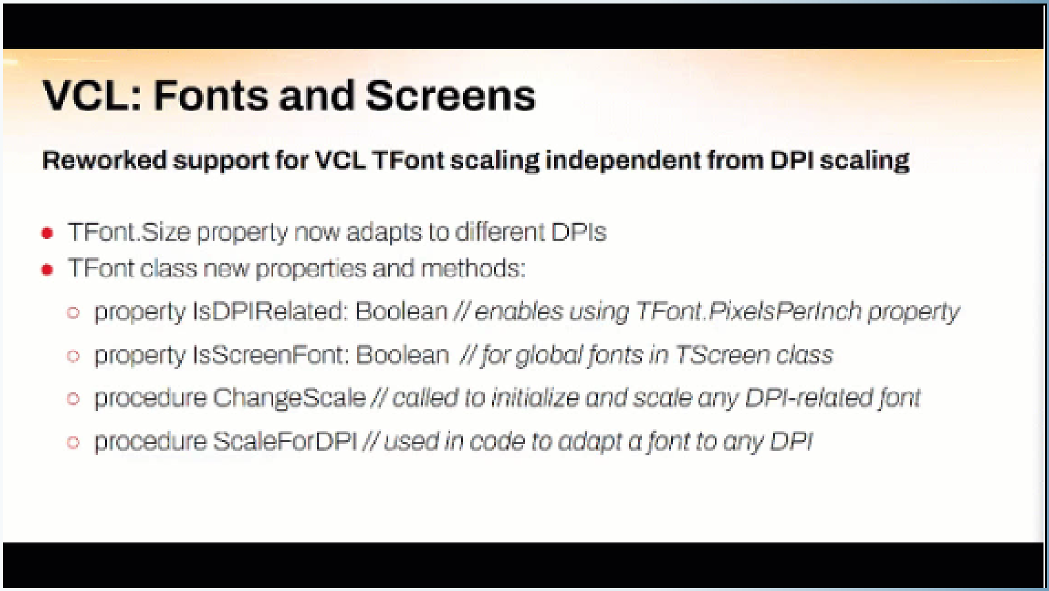
Support for Smart IDs in RAD Server
More powerful and flexible hosted REST APIs with new smart IDs (Sqids). Better performance, data paging improvements, better session authentication.



```
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Dialogs, FMX.StdCtrls, FMX.ListBox,
  FMX.Layouts, FMX.Controls.Presentation;

type
  TDeviceInfoForm = class(TForm)
    btnGetDeviceInfo: TButton;
    ListBox1: TListBox;
    lbOSName: TListBoxItem;
    lbOSVersion: TListBoxItem;
    ToolBar1: TToolBar;
    Label1: TLabel;
    lbDeviceType: TListBoxItem;
    procedure btnGetDeviceInfoClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  DeviceInfoForm: TDeviceInfoForm;

implementation
{$IFDEF IOS}
uses
  iOSapi.UIKit, iOSapi.Foundation, Macapi.Helpers;
{$ENDIF}
{$IFDEF ANDROID}
uses
  Androidapi.JNI.JavaTypes,
  Androidapi.Helpers,
  Androidapi.JNI.Os;
{$ENDIF}

{$R *.fmx}
{$R *.LgXhdpiPh.fmx ANDROID}


{$IFDEF ANDROID}

function GetCodename(VerString: string): string;
begin
  if VerString = '1.0' then
    Result := 'BASE'
  else if VerString = '1.1' then
```

Use RAD Studio on 4k+ Screens!
RAD Studio 12 offers high-DPI support to the IDE, enabling developers to work on larger, high-resolution screens. Full support for the latest 4k+ high-resolution monitors improves daily developer activities with cleaner, sharper fonts and icons, and high-resolution support throughout the IDE windows, including in the VCL and FMX form designers and code editor

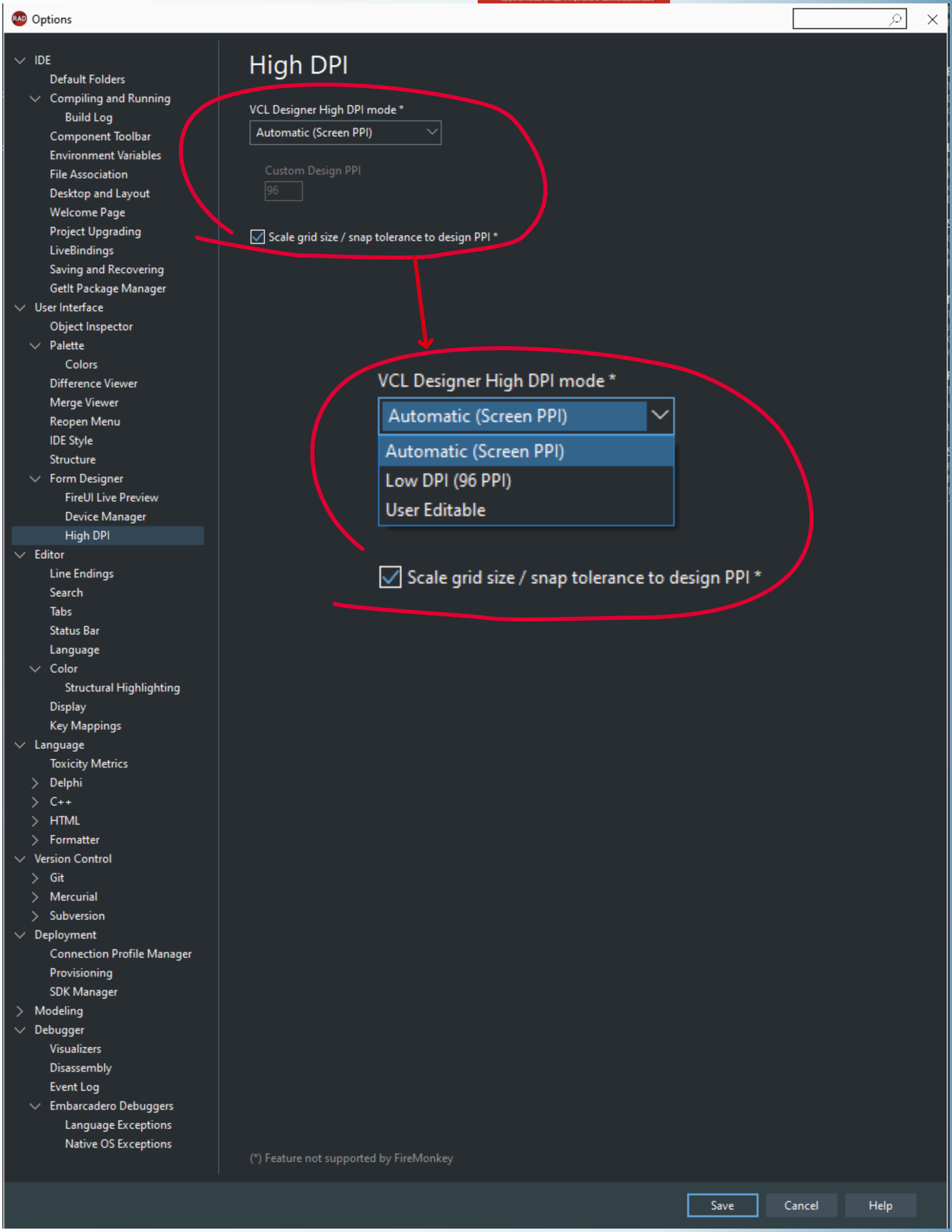




VCL: Fonts and Screens

Reworked support for VCL TFont scaling independent from DPI scaling

- TFont.Size property now adapts to different DPIs
- TFont class new properties and methods:
  - property IsDPIRelated: Boolean *// enables using TFont.PixelsPerInch property*
  - property IsScreenFont: Boolean *// for global fonts in TScreen class*
  - procedure ChangeScale *// called to initialize and scale any DPI-related font*
  - procedure ScaleForDPI *// used in code to adapt a font to any DPI*



High DPI

VCL Designer High DPI mode *: Automatic (Screen PPI)

Custom Design PPI: 96

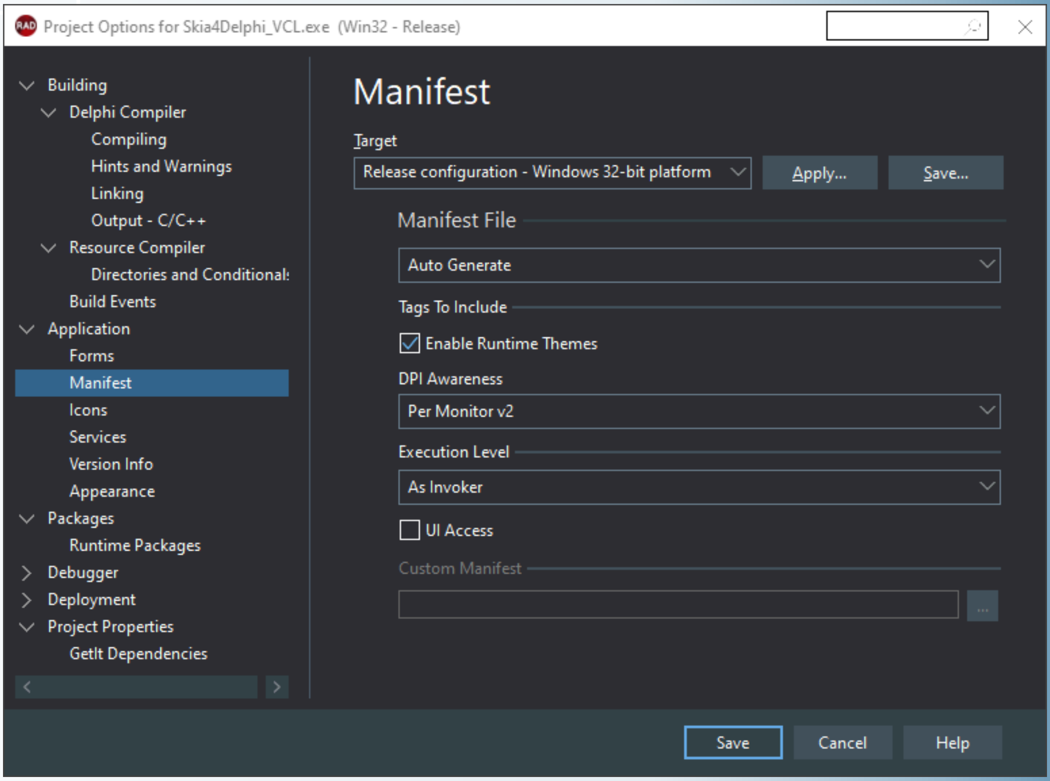
☑ Scale grid size / snap tolerance to design PPI *

VCL Designer High DPI mode *: Automatic (Screen PPI)

- Automatic (Screen PPI)
- Low DPI (96 PPI)
- User Editable

☑ Scale grid size / snap tolerance to design PPI *

(*) Feature not supported by FireMonkey



Project Options for Skia4Delphi_VCL.exe (Win32 - Release)

- Building
  - Delphi Compiler
    - Compiling
    - Hints and Warnings
    - Linking
    - Output - C/C++
  - Resource Compiler
    - Directories and Conditional:
  - Build Events
- Application
  - Forms
  - Manifest
  - Icons
  - Services
  - Version Info
  - Appearance
- Packages
  - Runtime Packages
- Debugger
- Deployment
- Project Properties
  - GetIt Dependencies

Manifest

Target: Release configuration - Windows 32-bit platform

Manifest File: Auto Generate

Tags To Include

☑ Enable Runtime Themes

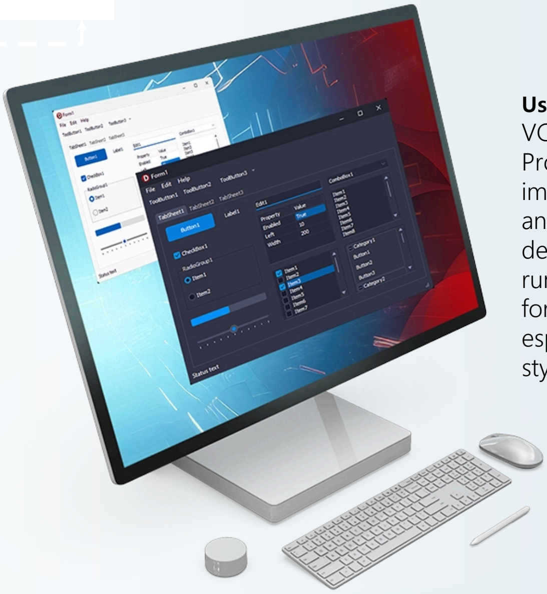
DPI Awareness: Per Monitor v2

Execution Level: As Invoker

□ UI Access

Custom Manifest

Target Windows 11
Official support for Windows 11 provisioning with integrated MSIX generation. Web Browser component for Windows, with support for both the IE ActiveX and the new Microsoft WebView 2 control (Chromium-based Edge). Enhanced VCL Form Designer to visually build native Windows applications, with live snap-to hints and layout guidelines. Enhanced Delphi and C++ RTL for 32-bit Windows and 64-bit Windows.



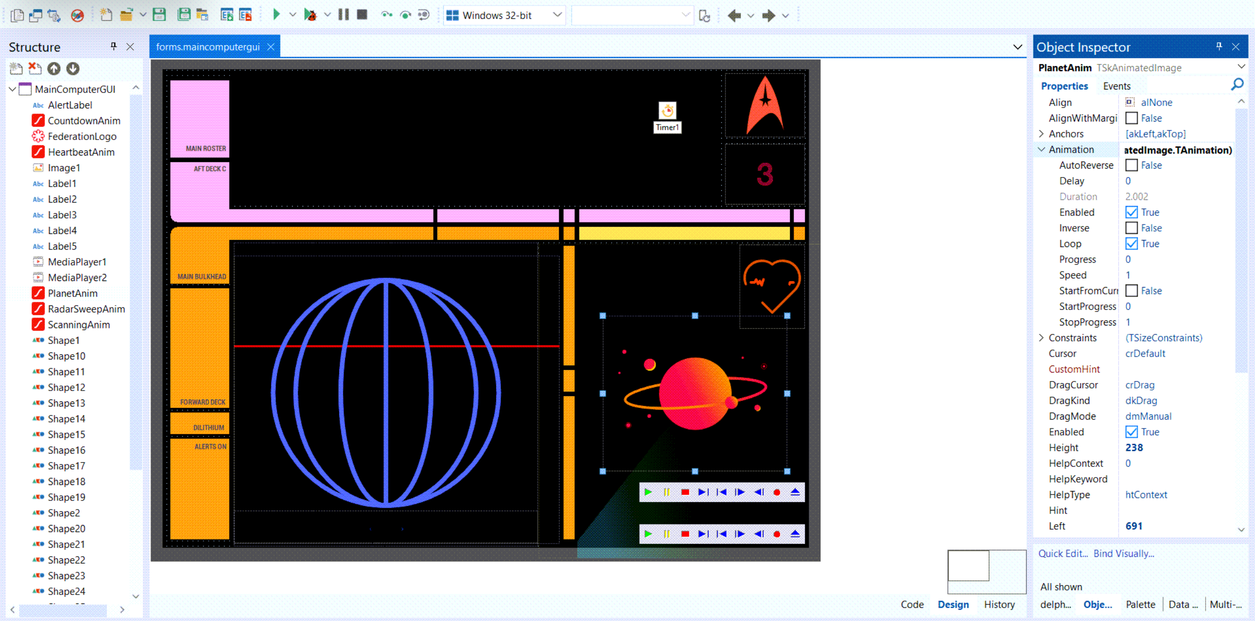
Use VCL Styles at Design Time!
VCL Styles now provides design-time support: Prototype stylish UIs even faster by seeing immediately at design-time how your styled forms and controls will look when running. Viewing at design time how styles will impact the UI at runtime improves the design and testing process for modern UIs. Creating better UIs faster is especially useful when working with per-control styles.

Deploy on M-Series Apple Silicon!
Compile for macOS (M-series Apple Silicon) and use the new universal package for AppStore submission. You can now compile for both existing Intel and new M-series macOS processors (Apple Silicon). Compiling for the newest processor versions enables the fastest performance across all platforms, and supports universal packaging for the macOS app store.

Collaborate Remotely!
Improved Remote Desktop Support for VCL and IDE, helping developers working remotely from the office. Enhanced debugging for remote and local 64-bit Windows applications and macOS 64-bit applications (Intel and ARM). Enhanced remote desktop support boosts your team's efficiency and improves your bottom line.

A 'STARSHIP COMPUTER CONSOLE FROM THE FUTURE' - EXAMPLE OF USING SKIA4DELPHI

BY IAN BARKER



Structure:

```
MainComputerGUI
  AlertLabel
  CountdownAnim
  FederationLogo
  HeartbeatAnim
  Image1
  Label1
  Label2
  Label3
  Label4
  Label5
  MediaPlayer1
  MediaPlayer2
  PlanetAnim
  RadarSweepAnim
  ScanningAnim
  Shape1
  Shape10
  Shape11
  Shape12
  Shape13
  Shape14
  Shape15
  Shape16
  Shape17
  Shape18
  Shape19
  Shape2
  Shape20
  Shape21
  Shape22
  Shape23
  Shape24
```

Object Inspector — PlanetAnim TSkAnimatedImage — Properties / Events

Align	alNone
AlignWithMargi	□ False
Anchors	[akLeft,akTop]
Animation	atedImage.TAnimation)
AutoReverse	□ False
Delay	0
Duration	2.002
Enabled	☑ True
Inverse	□ False
Loop	☑ True
Progress	0
Speed	1
StartFromCurı	□ False
StartProgress	0
StopProgress	1
Constraints	(TSizeConstraints)
Cursor	crDefault
CustomHint	
DragCursor	crDrag
DragKind	dkDrag
DragMode	dmManual
Enabled	☑ True
Height	238
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
Left	691

The code for this project can be found at
https://github.com/checkdigits/spacecomputer

```
function TMainComputerGUI.RandomKlingonString(const DesiredLength: integer): string;
var  Klingon:   char;
begin
 Result := '';
 for var count := 1 to DesiredLength do
 begin
  if Random(5) mod 5 = 0 then
   Result := Concat(Result,' ')
  else
   begin
    Klingon := Char($F8D0 + Random(25));
    Result  := Result + Klingon;
   end;
 end;
end;

function TMainComputerGUI.RandomLineOfNumbers: string;
var
 iWidth:  integer;
 iSpaces: integer;
 iIndex:  integer;

const
 cSpaces: array[0..5] of integer = (4, 6, 4, 8, 4, 4);

begin
 Result := '';
 for var iLoop := 0 to 10 do
 begin
  iSpaces := cSpaces[iLoop mod 6];
  Result  := Result + Format('%.4f%s', [Random, StringOfChar(' ', iSpaces)]);
 end;
end;
```

12 times 12 new features in Delphi 12

RAD Studio 12 includes some great improvements for C++Builder and the launch webinar and other online content highlights it. However, it is also a fantastic release for Delphi developers. I compiled 12 lists with 12 improvements each for Delphi 12. So this is not a list of 12 improvements for Delphi 12. It's a list of 12x12=144 improvements, plus half a dozen for native Windows bringing the total to a whopping 150 -- excluding all the existing improvements for C++Builder, as here I want to underline the Delphi side (but most of the features below are in fact for both languages).

The first blog post had 3 x 12 VCL Enhancements in Delphi 12, as you can read at **https://blogs.embarcadero.com/3-x-12-vcl-enhancements-in-delphi-12/.**
The second blog post focused on FireMonkey and the Android platform support, as you can see at **https://blogs.embarcadero.com/3-x-12-firemonkey-and-android-enhancements-in-delphi-12/**
The third installment had three lists focused on three related areas, Delphi Runtime Library (RTL) , database access, and Internet access, as you can see at **https://blogs.embarcadero.com/3-x-12-rtl-data-and-internet-enhancements-in-delphi-12/**
This last blog post of the series has 12 entries each for the IDE, the installer, and the Delphi language. Plus a bonus 6 on Windows API integration, bringing the overall total to 150.

In the IDE

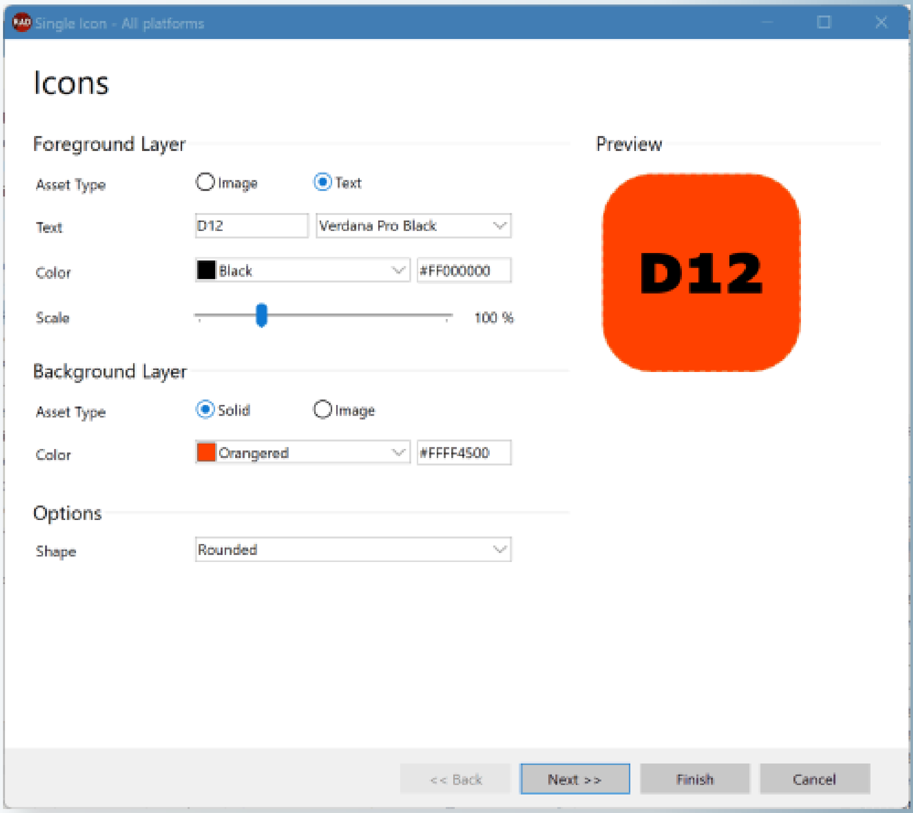
❶ The Find in Files dialog now has a new "Subdirectory exclude mask" option
❷ Syntax highlighting has been added to Error Insight hints, Navigation toolbar, Call Stack
❸ The Structure view syntax highlights methods and types and has syntax highlighting added to the Error Insight messages
❹ The Options - IDE - Saving and Recovering page has a new checkbox to save the editor state
❺ The Markdown window now changes colors when the IDE theme is changed
❻ The Welcome Page now supports smooth mouse wheel scrolling
❼ GDI bitmap* counts is lower through the IDE as images are made dormant if not used for some time
❽ Code templates and language keywords can now be displayed in Delphi LSP-based code completion
❾ Code completion now adds array braces [ ] for array types
❿ All Platforms Single Icon Wizard
⓫ PAServer messages, including hints, will be shown in the IDE Messages pane
⓬ New ToolsAPI, IOTARawEditReader interface

Bitmaps (Windows GDI)
A bitmap is a graphical object used to create, manipulate (scale, scroll, rotate, and paint), and store images as files on a disk. This overview describes the bitmap classes and bitmap operations.

In the IDE

❶ The Find in Files dialog now has a new "Subdirectory exclude mask" option
❷ Syntax highlighting has been added to Error Insight hints, Navigation toolbar, Call Stack
❸ The Structure view syntax highlights methods and types and has syntax highlighting added to the Error Insight messages
❹ The Options - IDE - Saving and Recovering page has a new checkbox to save the editor state
❺ The Markdown window now changes colors when the IDE theme is changed
❻ The Welcome Page now supports smooth mouse wheel scrolling
❼ GDI bitmap counts is lower through the IDE as images are made dormant if not used for some time
❽ Code templates and language keywords can now be displayed in Delphi LSP-based code completion
❾ Code completion now adds array braces [ ] for array types
❿ All Platforms Single Icon Wizard
⓫ PAServer messages, including hints, will be shown in the IDE Messages pane
⓬ New ToolsAPI, IOTARawEditReader interface

Below: The All Platforms Single Icon Wizard



Single Icon - All platforms

Icons

Foreground Layer

Asset Type ○ Image ● Text

Text D12 Verdana Pro Black

Color Black #FF000000

Scale 100 %

Background Layer

Asset Type ● Solid ○ Image

Color Orangered #FFFF4500

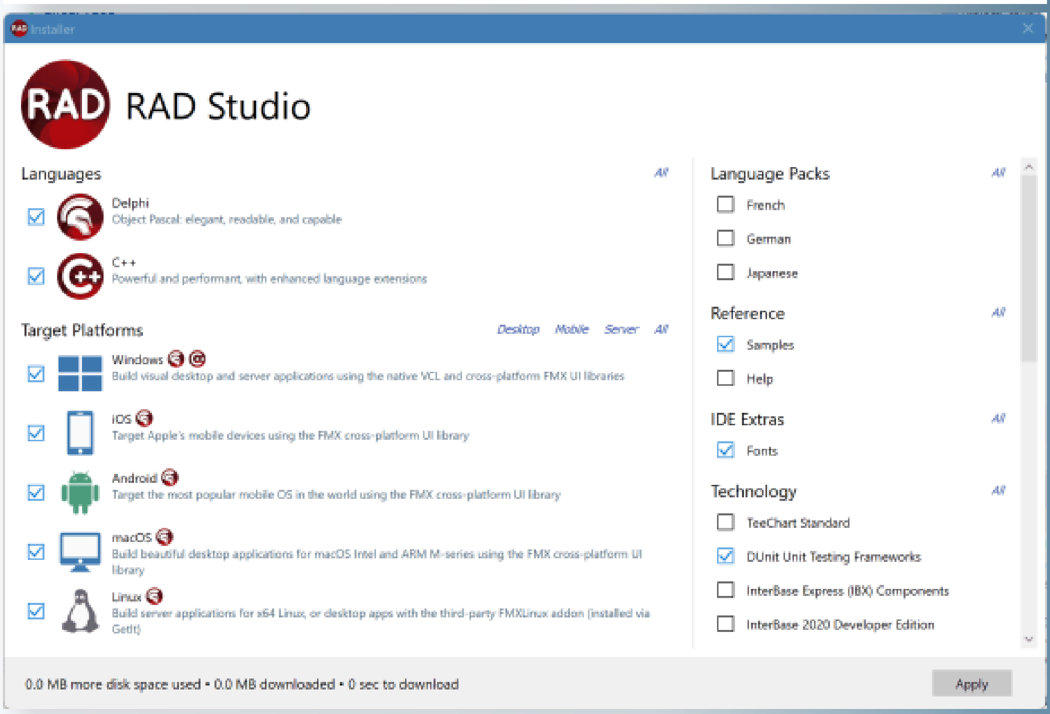
Options

Shape Rounded

In GetIt and the Installer

The Platform Manager has been renamed to Feature Manager
The Feature Manager has a completely new UI, build with regular VCL controls and styles
The Feature Manager offers all options in a single page
The Feature Manager separates the languages and the platforms you want to install
The Feature Manager includes preset configurations for common scenarios like Desktop or Mobile
The Feature Manager has a new “Errors” button that shows installation errors directly and has a direct link to the error log file
The GetItCmd command line tool is now logging to the GetItInstall.log file
The GetIt package manager has the option to load multiple local GetIt packages with a single operation
The integrated version of DUnitX has been updated
The integrated version of Indy has been updated
The long deprecated VCL translations tools has been removed from the core product installation
The fairly old Modeling support has become an optional feature in the Feature Manager

Below: The new Feature Manager window



Languages — All

☑ Delphi — Object Pascal: elegant, readable, and capable

☑ C++ — Powerful and performant, with enhanced language extensions

Target Platforms — Desktop Mobile Server All

☑ Windows — Build visual desktop and server applications using the native VCL and cross-platform FMX UI libraries

☑ iOS — Target Apple's mobile devices using the FMX cross-platform UI library

☑ Android — Target the most popular mobile OS in the world using the FMX cross-platform UI library

☑ macOS — Build beautiful desktop applications for macOS Intel and ARM M-series using the FMX cross-platform UI library

☑ Linux — Build server applications for x64 Linux, or desktop apps with the third-party FMXLinux addon (installed via GetIt)

Language Packs — All
☐ French
☐ German
☐ Japanese

Reference — All
☑ Samples
☐ Help

IDE Extras — All
☑ Fonts

Technology — All
☐ TeeChart Standard
☑ DUnit Unit Testing Frameworks
☐ InterBase Express (IBX) Components
☐ InterBase 2020 Developer Edition

0.0 MB more disk space used • 0.0 MB downloaded • 0 sec to download

And, Finally, in the Delphi language

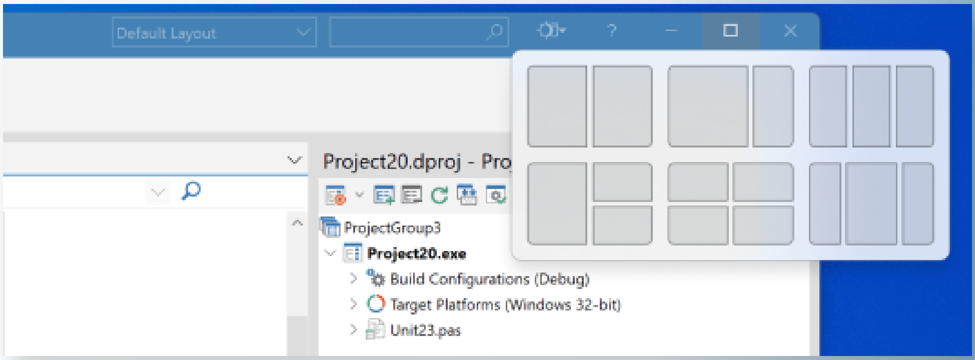
❶ Long literal string With more than 255 characters
❷ Multiline string literals
❸ TEXTBLOCK directive to specify the line breaks format of multi line strings
❹ NativeInt, a type mapped to Integer or Int64 depending on the platform, is now a weak type alias
❺ Improved warnings in generic classes
❻ New LLVM symbol defined in all LLVM-based Delphi compilers
❼ Option to export the units uses graph in a GraphViz file (–graphviz)
❽ Ability to exclude family of units form the GraphViz file (–graphviz-exclude)
❾ Support for NaN (not a number) comparisons as required by IEEE
❿ Optimized generated code for div operations when the divisor is a constant
⓫ Two new functions to the System unit, GetCompilerVersion and GetRTLVersion
⓬ Floating-Point Exceptions are now disabled by default on all Platform

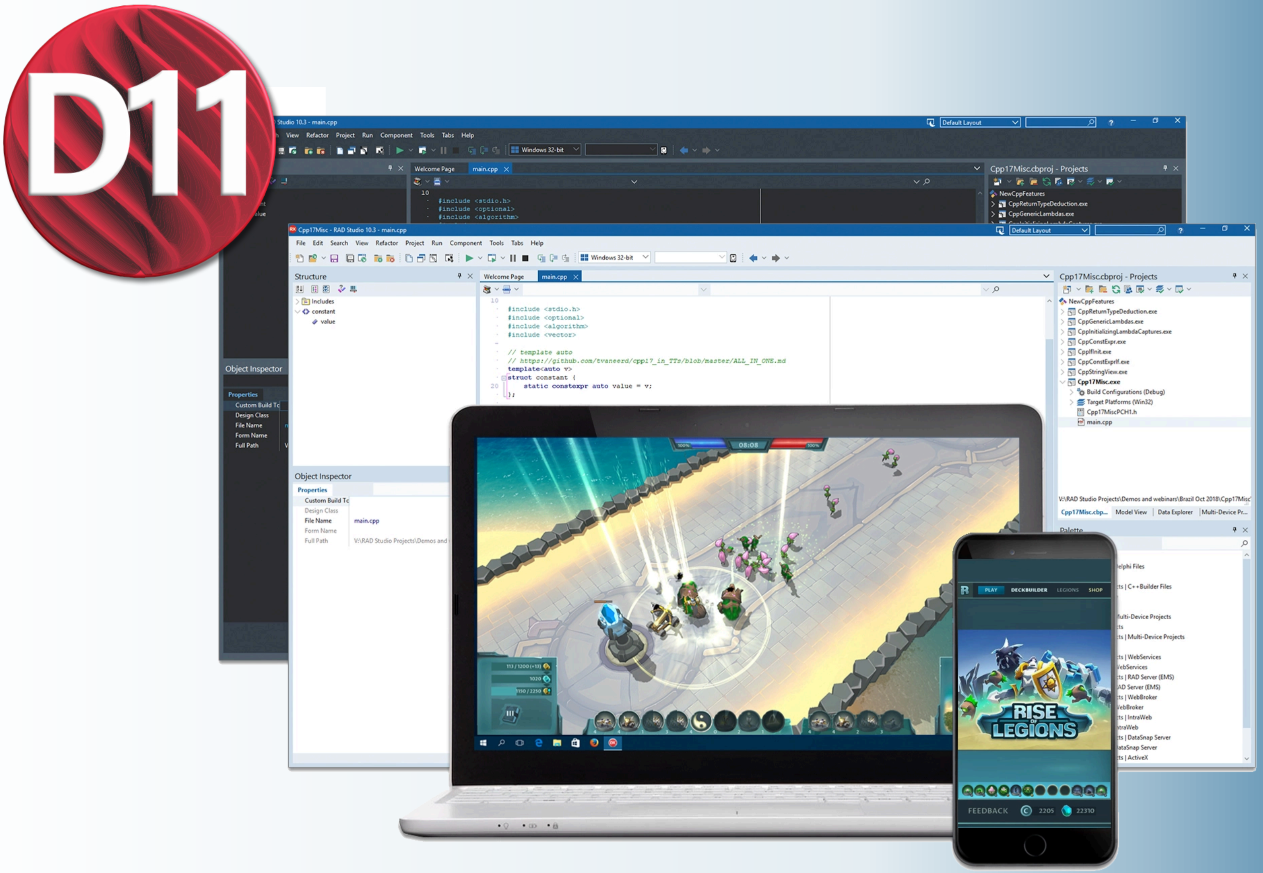
Multiline Delphi string literals in the IDE example is seen on page 1 of this article.

Bonus Section: In Windows Platform

❶ New WinAPI definition based on WinMD Microsoft metadata
(*311 header files with 41 MB of Delphi code*)
❷ WinRT APIs and WebView 2 control API refreshed to the most recent version
❸ Edge browser: UserAgent is available in ICoreWebView2Settings; ICoreWebView2Profile2 contains methods `ClearBrowsingData, ClearBrowsingDataAll,` and `ClearBrowsingDataInTimeRange; TEdgeBrowser OnDownloadStarting` event, `NavigateWithWebResourceRequest` method, `Print` and `ShowPrintUI` methods
❹ New Windows 11 Styles
❺ `TForm` class `EnableImmersiveDarkMode` method and `RoundedCorners` property for Windows 11
❻ Title Bar support for Windows 11 snap layouts (see image below)

Snap Layout for Title-bar work also for the IDE





```
#include <stdio.h>
#include <optional>
#include <algorithm>
#include <vector>

// template auto
// https://github.com/tvaneerd/cpp17_in_TTs/blob/master/ALL_IN_ONE.md
template<auto v>
struct constant {
    static constexpr auto value = v;
};
```

FREE COMMUNITY EDITION

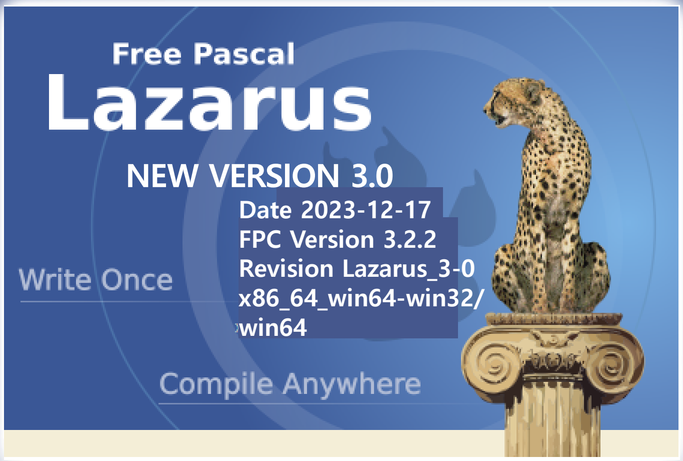
Get Community Edition Free
Full-Featured Free Delphi IDE for Creating Native Cross-Platform Apps

Can I Get The Delphi CE?
If you're an individual, you may use **Delphi CE** to create apps for your own use and apps that you can sell until your revenue reaches US$5,000 per year.

If you're a small company or organization with up to US$5,000 per year in revenue, you can also

use the Delphi CE. Once your company's total revenue reaches US$5,000, or your team expands to more than five developers, you can move up to an unrestricted commercial license with Professional edition.

Delphi CE is also perfect for early stage startups who are bootstrapping their product vision before securing capital! Develop your professional app with the **Community Edition** knowing that you can skip the learning curve your competition faces when building for multiple platforms.
See the Community Edition FAQs for additional details.



NEW VERSION 3.0
Date 2023-12-17
FPC Version 3.2.2
Revision Lazarus_3-0
x86_64_win64-win32/
win64

LCL INTERFACES CHANGES

COCOA

- **IME (***Input method editor, allowing the entry of characters not physically present on a computer keyboard***)** fully supported, such as **`Chinese/Japanese/Korean, DeadKeys, Emoji & Symbols`**.
- Multi Displays/Monitors fully supported.
- Docking fully supported, including the IDE (*Integrated Development Environment*).
- Cursor completely refactored and significantly improved, compatible with macOS Ventura.
- TPageControl significantly improved.
- Many controls improved (`TComboxBox/TListBox/TDateTimePicker/TStaticText/TSpeedButton/TPanel etc.`)
- Many memory leaks fixed.

Qt

- Implemented `TCheckBox.Alignment` and `TRadioButton.Alignment`.
- `TCustomComboBox.AdjustDropDown` and `TCustomComboBox.ItemWidth`.

Qt5

- Qt5 uses native event loop on all platforms. C bindings are updated.
  Minimum C bindings version for **lazarus** 3.0 is 1.2.15.
- Note that most **Linux** Distributions will not have an appropriate **libqt5pas library** until their next release after the formal release of **Lazarus 3.0.**
  Build your own from the **Lazarus** source tree or download from
  `https://github.com/davidbannon/libqt5pas/releases/latest`
- Implemented `TCheckBox.Alignment` and `TRadioButton.Alignment.`
- Implemented `TCustomComboBox.AdjustDropDown` and `TCustomComboBox.ItemWidth`.

Qt6

- Qt6 widgetset implemented. C bindings are based on Qt6 6.2.0 LTS.
  Minimum C bindings version for lazarus 3.0 is 6.2.7.
- Note that most **Linux Distributions** will not have an appropriate libqt6pas library until their next release after the formal release of Lazarus 3.0.
  Build your own from the **Lazarus** source tree or download from
  `https://github.com/davidbannon/libqt6pas/releases/latest`
- Implemented `TCheckBox.Alignment and TRadioButton.Alignment.`
- Implemented `TCustomComboBox.AdjustDropDown` and `TCustomComboBox.ItemWidth`.

Gtk3

- **Gtk3 pascal bindings are completely reworked**.
- A number of stability improvements.
- Now requires GTK >= 3.24.24 and Glib2.0 >= 2.66

LCL CHANGES

TCustomImageList

`TCustomImageList` is made more extensible:

❶ Protected `MarkAsChanged` method is added, which sets `FChanged` to `true` (*this allows to make custom triggers for* `OnChange` *event*).

❷ Virtual protected `DoAfterUpdateStarted` and `DoBeforeUpdateEnded` methods are added. They are called in first BeginUpdate and last EndUpdate respectively.

TTaskDialog

- Old behaviour Win32:
  A placeholder icon was used for `FooterIcon = tdiNone` and `MainIcon = tdiNone.`
- **New behaviour Win32:**
  No icon is used for `FooterIcon = tdiNone` and `MainIcon = tdiNone.`
- **Reason**: Removing drawing glitch**.**
  The text move over to allow more content and better alignment. See Issue #39172

TSpeedButton

- Old behaviour: Multi-line captions could only be entered by code.
  And multi-line captions were always left-aligned.
- **New behaviour:** Multi-line captions can also be entered in the object inspector.
  New property Alignment to specify whether the caption should be left-/right-aligned or centered. Default: centered, like in **Delphi**.
- **Reason:** Better usability.

TLabel.Transparent, .Color and .ParentColor changes

- Old behaviour: `Transparent` property was bound to `Color=clNone`
- **New behaviour:** Transparent is a standalone property
- **Reason**: Delphi compatibility and to fix ParentColor issues.
- **Remedy**: If you are setting the `Color` property, `Transparent` is not automatically switched from `True` to `False` now, you have to do it yourself.
  This is in compliance with **Delphi** and also solves problems with `Color/ParentColor` changes.

TPanel.VerticalAlignment

- Old behaviour: The panel caption was always centered vertically.
- **New behaviour:** The new property `VerticalAlignment` (`taAlignTop, taAlignBottom, taVerticalCenter`) allows to place the caption also at the top or bottom of the panel interior.
- **Reason**: **Delphi** compatibility and better usability

TCalendar

- Properties `MinDate` and `MaxDate` are implemented. These limits are only imposed if `MaxDate > MinDate`. **Unfortunately GTK2/3 widgetsets do not support this**, so selecting a date outside the `MinDate/MaxDate` range will still be possible there.

TCheckbox, TRadioButton

- Different calculation of `checkbox/radiobutton` size in order to correctly take care of Win-10 "ease of access" feature. See issue #39398
- **Consequence:** `lfm` files will contain different sizes of these controls *(if auto-sized)* compared with earlier versions.

LCL CHANGES CONTINUATION

Grids

- You can now set the cell editors properties `ParentColor` and `ParentFont` by including `goEditorParentColor` resp. `goEditorParentFont` in the grid's Options2 property

TShellTreeView

- **New property** `ExpandCollapseMode` defining options whether a collapsing node should clear its child nodes.
- Implements custom sorting of the treeview items by setting `FileSortType` to `fstCustom` and providing a custom compare function in the event `OnSortCompare`.

TShellListView

`TShellListView` now subclasses `TListItem`, so it can store file info in it.
If you use `OnCreateItemClass` to create your own descendant of `TListItem`, it may be advisable to base your own class on `TShellListItem` instead of on `TListItem`.
This way you will also have access to the `TShellListItem's FileInfo` property.

TTreeView

Adds `ShowSeparators` as a published property like `ShowLines` and ShowRoot.
`TTrayIcon` gtk2 Only

IDE CHANGES

Character map

- Resizable characters to improve readability.
- The character map was split off from the IDE and moved to separate packages.
  The designtime package is installed by default so that there is no difference in the IDE.
  Now users can access it in their own applications after adding the runtime package "charactermappkg.lpk" to the project requirements.

DEBUGGER

Project Options

- "Run(F9)" can either be "Debug" or "Run without debug".
  In newly created Debug/Release modes, the Release mode will no longer invoke the debugger by default.
  Existing Debug/Release modes must be edited, to enable this.
- Per project "Debugger backend" settings. In addition to choosing a specific backend from the global IDE settings, a backend can be configured just for the project (i.e. special gdb-server settings)

IDE Dialogs

- Improved Watches window
  - Expand/Unfold for classes, records, etc
  - Expand/Unfold with paged browser for arrays
  - Drag and Drop to reorder watches
  - Drag and Drop to create new "top level" watches from nested entries (in expand/unfolded lists)
  - Address column for types with internal pointer (classes, long-string, dyn-array, (real) pointer)
- Improved Locals window
  - Expand/Unfold for classes, records, etc
  - Expand/Unfold with paged browser for arrays
  - Address column for types with internal pointer (classes, long-string, dyn-array, (real) pointer)
  - Power button

Free Pascal
Lazarus
Version: 3.0
Date: 2023-12-17
FPC Version: 3.2.2
Revision: lazarus_3_0
x86_64-win64-win32/win64

IDE CHANGES CONTINUATION

Improved Inspect window

- Fixed: Updating value when context changes
- Added options for Function calling / and "Converter" (*See* `FpDebug: SysVarToLstr`)
- Added filter to search for text in name or value.
- Added `ctrl-Up/Down/Page-Up/Page-Down` to navigate the grid.
- `Alt-Left/Right` for history. And `ctrl-Enter` to select.

Improved Evaluate/Modify window

- New Layout
- Added DisplayFormat
- Added options for Function calling / and "Converter" (*See* `FpDebug: SysVarToLstr`)

Improved Assembler window

- Added history navigation (*forward/backward*)
- For `FpDebug`: added annotations to jump/call targets, and allow to ctrl-click to disassemble target address

FpDebug / LazDebuggerFp

- Improved "function calling" in watch eval. See `FpDebug-Watches-FunctionEval`
- %RAX Accessing cpu registers in watch expression (*only full registers, not yet AH or AL or EAX on 64bit*)
- Intrinsic functions: FpDebug-Watches-Intrinsic-Functions
- Intrinsic/extended operators: `MyArray[1..3] array` slice with operator mapping.
- FpDebug-Watches-Intrinsic-Functions#Intrinsic_Operators
- Option to detect "`variant`" and call "`SysVarToLStr`" in the target app.
- Suspend/Resume individual threads (*must be done while app is paused, and will be applied for subsequent step/run*)
- Partial improvements to debug in DLL: `https://wiki.freepascal.org/Debugger_Status#Other`
- Disassembler now annotates lines for `call/jmp/jne/...` with info on the target address (*function name, file, line*)
- F7/F8 Step-Into/Over can now be used to start the debugger and run to the first line of the main program `begin/end`.

LazDebuggerFpLldb (default on MacOS)

- Added Mem-Limits (*and* `String,Pchar,Array`) to the debugger config
  See `https://wiki.freepascal.org/LazDebuggerFp`
  `(MaxMemReadSize, MaxStringLen, MaxArrayLen, MaxNullStringSearchLen)`
  Limiting the default results for `watches/locals/stack-params,...`
  can prevent slow evaluation.
  Arrays can then be browsed in the watches window, using the new "paged browser for arrays" (*expand via [+]*)

Floating point properties in the Object Inspector

- The **Object Inspector** now explicitely **disallows to set a floating point property's value** to +/-Inf or **NaN** (*Not a Number*).
- **Reason**: whilst +/-Inf and **NaN** are valid values for a floating point property, they cannot be streamed (*so, the form could not be loaded*) and setting it to **NaN** caused havoc in the IDE.
- Remedy: set the value at runtime (in code)

IDE CHANGES CONTINUATION

Reading unit names in lfm

- FPC 3.3.1 component writer supports optionally writing types with their unit names as unitname/type. Lazarus can now read this.

Ambiguous Classtypes

- You can now register two component classes with the same name, e.g. fresnel.TButton and StdCtrls.TButton. You can even put both on the same form.

Editor

- Highlight for PasDoc

IDE Options

In Tools -> Options -> Environment -> Window page these three settings are now ON by default :

- IDE title starts with project name
- IDE title shows project directory
- IDE title shows selected build mode

It has been requested by many users.

If the IDE's title bar has no info about the active project, a user must open **Project Inspector** or **Project Options** to see it, which is inconvenient.

Lazarus Examples

- A new approach to Examples that copies an Example to a fresh working area (avoiding the Linux Read Only Problem) and, possibly an easier to use search model.

IDE INTERFACE CHANGES

COMPONENTS

TAChart

- The `TLegendClickTools` now is able to detect clicks on series legend items and reports the clicked series in the new `OnSeriesClick` event.
- New `TDatapointMarksClickTool` which becomes active when the user clicked on the marks of a series.
- New property `TickWidth` for the chart axes.
- New property `FullWidth` for the chart title and footer to run their background across the entire chart width.
- New property `RandomColors` for `TRandomChartSource`.
- New properties `YIndexWhiskerMin`, `YIndexBarMin, YIndexCenter, YIndexBarMax, YIndexWhiskerMax, and YDataLayout` in `TBoxAndWhiskerSeries` for more flexible assignment of y values to the parts of the box/whisher shape.
- New option `aipInteger` in the set `TAxisIntervalParamOptions` which sets axis `labels` only at `integer` values and thus supresses the unwanted intermediate labels in bar charts and helps to enforce labels in logarithmic plots at the powers of the logarithmic base (*usually 10*).
- New event `OnAddStyleToLegend` for `TChartStyles`. It has a `boolean` parameter `AddToLegend` with which you can determine whether the series level using this style is displayed in the legend.

TDateTimePicker

- New properties `MonthDisplay` and `CustomMonthNames`.
  They are meant to replace the `MonthNames` property, which has been deprecated.
- New property `DecimalSeparator`.
  Allows a user-specified value to be used instead of a hard-coded Colon character.

IDE INTERFACE CHANGES CONTINUATION

TDBDateTimePicker

- Adds Options as a published property.
- Publishes the `DecimalSeparator` property.
- Publishes the missing `Alignment` property. Consistent with `TDateTimePicker`,

T(Float)SpinEditEx

- **New property Orientation** which allows to arrange the spin buttons horizontally.

TCheckListBox

- Adds HeaderColor and HeaderBackgroundColor properties. Used on list items where the
- Header property is enabled. Implemented for the Win32 widget set.

PAS2JS

- lazbuild now can compile **PAS2JS** projects by passing the environment variable **PAS2JS** with the path of the pas2js executable.
- Project groups with pas2js projects now can compile without being opened.
- **New project type PROGRESSIVE WEB APPLICATION**
- **New project type ELECTRON WEB APPLICATION**
- `pas2jsdsgn` now uses the `SimpleWebServerGUI` package, replacing its own **http server controlle**r.
- **F9, Run** now **builds, starts a HTTP server** and a **browser**

Lazarus Icon Collection

- Not a component, but the **Lazarus** installation now contains a folder with general-purpose icons for usage in toolbars, menus, buttons etc. of any GUI applications (*folder images/general_purpose*).
  The images come in various sizes and thus are compatible with the scaled image list of **Lazarus v2.0+**. Author: **Roland Hahn** (`https://www.rhsoft.de/`).
  License: Creative Commons CC0 (no restrictions in usage).

gir2pascal

Sources of `gir2pascal` (*a tool to convert* **GObject Introspection** *descriptions to* **Pascal** *files*) are now included in **Lazarus source tree** (`tools/gir2pascal directory`) and maintained there.

lazdelphi

An IDE addon adding a parser for the **Delphi** compiler errors and hints.
You can run `dcc32.exe` as external tool or execute before command in the compiler options and use the **Delphi Compiler parser** for the output, so that the errors/hints in the Messages window can open the source. See **Lazarus Delphi Compiler Tool**

Jedi Code Format

Some improvements in code formatting.
The **`jcf`** command line tool is now a text-mode application and no longer requires **XWindow** on **LINUX** to run.

DockedFormEditor

In case you are still using the sparta docked form editor, use the **dockedformeditor** package instead.

CHANGES AFFECTING COMPATIBILITY

IDE SETTINGS

Run > Run Parameters

The Working-Dir and the Launch-/Host-App can now be specified relative to the Project-Dir. As a result of this, and due to inconsistencies between "Run in debugger" and "Run without debug" some details of how those fields are resolved changed.

In some cases you may therefore have to adjust your settings.

The Working-dir is now determined as follows

❶ BuildMode.RunParams "working directory" set by user (*New, this can now be relative to project dir / to get the "output dir" containing the exe:* `"$Path($(OutputFile))"`)
❷ Project Dir, if not virtual
❸ Directory from Host-App (*RunParams*),
or if (*and only if*) Host-App is empty from Project.exe
(*If host app is in %PATH, then there is no "working directory"*)
The first 2 steps are the same as before the change.
The 3rd step was previously only used for "debugging",
but "run without debug" did use:
"Launch-App", "Host-App", `Project.exe`

Change

The path of the "Launch App" is no longer considered
The Launch-/Host-App location are now determined as follows
❶ An app with absolute path is used as given
❷ A relative path (including no path at all) is resolved as relative to the Project-dir.
❸ An app without any path at all (*if not found in step 2*) is searched in the `%PATH` environment.

Change

Search in `%PATH` was only done by "run without debug",
but not by debug. It is now done by both.

Change

Checking for an exe relative to the project-dir was added.

LCL incompatibility

TLabel: **autosized and right-aligned**

Old behaviour:

Autosized label with `Alignment=taRightJustify` but `Anchors=[akLeft,...]` grew to left.

New behaviour: The label grows to right now.

Reason: It wasn't possible to implement the behavior also for hidden labels without significant extensions in the **LCL**. The **LCL** has a different and more generic feature of control-based anchoring that delivers the same effect (*see Remedy down*), so it is not needed and wanted to double this feature and make the **LCL** code more complex and prone to bugs.

Remedy: Use the **LCL anchoring to a secondary control.**

Anchor the right side of the label to another control. Then the autosized label will grow to the left but won't move to the right when the parent is resized like it is done with a simple `akRight` anchor without a reference control.

CHANGES AFFECTING COMPATIBILITY CONTINUATION

TDateEdit/TTimeEdit

The value of `NullDate` has changed.

Reason:

It was impossible to actually select the date corresponding to `NullDate` (*30 dec 1899 by default)* in the control.

Remedy (1):

if your code depended on `NullDate` actually being 0.0, you have to adjust your code.

Remedy (2):

if your code used `NullDate` for a `TTimeEdit`, change that to the new constant `NullTime` instead.

NOTE: `NullDate` is actually a writeable constant. This was kept for compatibility reasons. It is however a bad idea to change it's value to anything that is an actual date that is within the range of the control.

Cocoa

Some global configuration variables were moved from `CocoaInt` to `CocoaConfig`. such as `CocoaBasePPI, CocoaIconUse, CocoaToggleBezel, CocoaToggleType` etc.

GTK3

GTK3 is no longer supported on earlier **Linuxes**, such as **Ubuntu 20.04**.
It requires GTK >= 3.24.24 and Glib2.0 >= 2.66

LAZUTILS

Masks unit

The masks unit has been completely rewritten.

Reasons:

speed: the old Matches() method had O(n^2) or even O(n^3) characteristics.
improved control over how the mask is interpreted.

New types (*for parameters*) and a dedicated `TMaskWindows` class have been added. `TMask.MatchesWindowsMask` and the old `TMaskOptions` type have been deprecated and will be removed in the next release.

Ranges and Sets

The old masks implementation supported sets, but not ranges.
The new implementation supports both `sets ([abc])` and `ranges ([a-c]).`
As a consequence a `'-'` inside such a construct is now interpreted as part of the range definition, not as a literal '-'.

Reason: ranges are a good thing to have by default (*the old implementation simply lacked this*). We decided it's a small price to pay.

Remedy: either escape the '-' with EscapeChar (*which defaults to* `'\'`) or exclude `mocRange` from the `TMaskOpcodes` parameter.

Constructors do not fail anymore on an invalid mask

- When providing an invalid mask to the old T(Windows)Mask(List) constructors an exception was raised.
- The new constructors do not raise an exception in this case. Instead an exception is raised in when Matches() is called.
- **Reason**: it's not very nice to have a constructor fail.

CHANGES AFFECTING COMPATIBILITY CONTINUATION 2

Translations unit

Added `GetLanguageID` function.
It returns a record with language code (*in ISO 639-1 or ISO 639-2*) and country code (*in ISO 3166*) for current system locale.

Added `GetLanguageIDFromLocaleName` function.

It parses **Unix** locale name and returns a record with language code and country code.
It is useful to parse language identifiers passed e. g. via command-line parameters.

Implementation is based on `GetLanguageIDs` procedure from `GetText` unit, but is rewritten to have the following properties:

- Language and country codes are always returned in **ISO formats** on **Windows**.
- **Unix** locale identifier is properly parsed and language/country codes are properly extracted.
- Don't assume that language code is always two-letter (*ISO 639-1*), it can have bigger length (*e. g. three letters, like in ISO 639-2*).
- Locale ID is returned in a record type. This will allow to return additional fields in backwards-compatible manner in future. Currently it contains language code, country code and language ID (*combination of language code and country code*).

These functions are used now throughout the **Lazarus** codebase. This greatly improves automatic language detection and loading of correct translations by **Lazarus**:

- Three-letter (*ISO 639-2*) language identifiers are no more truncated to two letters. Thus, translations for such languages will be correctly loaded when available.
- Previously on Windows some language and country codes were obtained in non-ISO format, which prevented correct loading of some translations, e. g. **Chinese (zh_CN)**.
- On **Unix** translations with country codes, like **Brazilian Portuguese (pt_BR)** or **Chinese (zh_CN)** are correctly loaded now.
- **macOS** is now handled as any other **Unix**. This removes dependency on language list in **Lazarus** bundle (*which had to be maintained manually*) and thus fixes loading of **Czech, Hungarian, Brazilian Portuguese, Ukrainian t**ranslations.

LazUTF8 unit

- Deprecated **LazGetLanguageIDs** (*returns combination of language and country codes*) and `LazGetShortLanguageID` (returns only language code) procedures.
- **Reason**: this functionality belongs to Translations unit (*calls of these procedures are almost always followed by calls to procedures from Translations unit*), and these procedures are now thin wrappers of `GetLanguageID` function from `Translations unit`.
- **Remedy**: use `GetLanguageID` function from `Translations unit`.

CHANGES AFFECTING COMPATIBILITY CONTINUATION 3

LazUTF8Classes and LazUTF8SysUtils units

Everything in these units was deprecated for a long time and **now they were removed.**
LazUTF8SysUtils was earlier renamed to **LazSysUtils** and this deprecated version was left for a transit period.

Class **TStringListUTF8** can be replaced with **TStringList**.
Class **TMemoryStreamUTF8** can be replaced with **TMemoryStream**.
Global procedure **LoadStringsFromFileUTF8** can be replaced with **TStrings.LoadFromFile**.
Global procedure **SaveStringsToFileUTF8** can be replaced with **TStrings.SaveToFile**.
Functions **NowUTC** and **GetTickCount64** can be found in **LazSysUtils**.

COMPONENTS INCOMPATIBILITY

LazControls

TSpinEditExBase derived classes

All derived classes form TSpinEditExBase must implement a SameValue method.
This method is defined as an abstract method in TSpinEditExBase.
Reason: All derived classes used **Math.SameValue.** This is wrong for comparing integer types (*even if is is safe when comparing relative small values*).
Remedy: unfortunately you'll have to adjust your code.

TFloatSpinEditEx

The property NumbersOnly is no longer published.
Reason: the property makes no sense for this control and only confuses users.
Remedy: if you really need NumbersOnly to be True, you must set it in code.

FpVectorial

The **Size element** in the FPVectorial TvFont record is a floating point value now (type double).
Reason: Avoid rounding errors because the drawing coordinates are double already.
Remedy: There is rarely a chance that this change will have an effect on user code.
Only when the font size is stored in a variable it must be declared as double rather than as integer.

TurboPower_ipro

Type declarations and the html nodes were moved from unit IpHtml to separate units, IpHtmlTypes, IpHtmlClasses and IpHtmlNodes.
This may break compilation of existing projects.
Reason: Improve **maintainability** of the extremely long unit IpHtml
Remedy: Add IpHtmlTypes, IpHtmlClasses and/or IpHtmlNodes to the uses clause of the project unit(s) when an "identifier not found" error referring to this package is reported by the compiler.

DRAWING TURTLE FIGURES IN LAZARUS

PROF DR HANS ZANTEMA
TECHNISCHE UNIVERSITEIT EINDHOVEN / RADBOUD UNIVERSITEIT NIJMEGEN

Starter Expert

ABSTRACT

In this article we give examples of amazing turtle figures created by Lazarus, only using a few lines of code.

DEFINITION OF TURTLE FIGURE

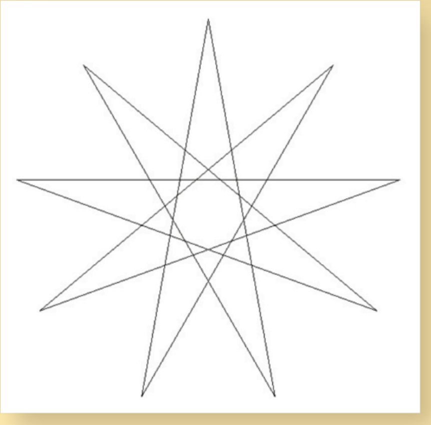
For a (*finite or infinite*) sequence of zeros and ones, and two angles `h0` and `h1`, the corresponding turtle figure is defined to be the path followed by a robot that executes the following commands one by one, for all elements of the sequence:

- If the symbol is 0, the walking direction of the robot turns the angle `h0`
- If the symbol is 1, the walking direction of the robot turns the angle `h1`
- In both cases the robot walks a step forward, of some unit length

Traditionally, a robot having these basic commands of turning and walking forward, is called a turtle. This goes back to LOGO programming in the 1960s.

A FIRST EXAMPLE

As a first example, consider the infinite sequence `00000`... only consisting of zeros, and choose `h0 = 160°`.
As the sequence does not contain ones, the value of h1 does not play a role.
The turtle figure consists of infinitely many segments of unit length, each obtained after turning `160°`, so every time making an acute angle of 20°.
After doing this 9 times, in total the turtle has turned `9 x 160°`, being a multiple of `360°`, and the turtle is back at its original starting point.
In all next steps only segments are overwritten that were drawn before.
So for this infinite sequence the resulting turtle figure is finite and looks as follows:



DRAWING TURTLE FIGURES IN LAZARUS

For drawing such a turtle figure in Lazarus for every step we compute the new location (x,y) of the turtle, and us the `lineto` command to draw the line to this new location.
Apart from the variable x and y we need a variable h to represent the actual direction of the turtle, a variable u to represent the unit length and a variable n to represent the number of steps to be done. The variables `x` and `y` have to be real, and should be rounded to integers in order to apply the `lineto` command.

For the example just given after initialization of all these variable in a canvas environment this may read as follows:

```
 moveto(round(x),round(y));
for i := 1 to n do
        begin
        h := h + 160;
        x := x + u * cos(h);
        y := y + u * sin(h);
        lineto(round(x),round(y));
        end;
```

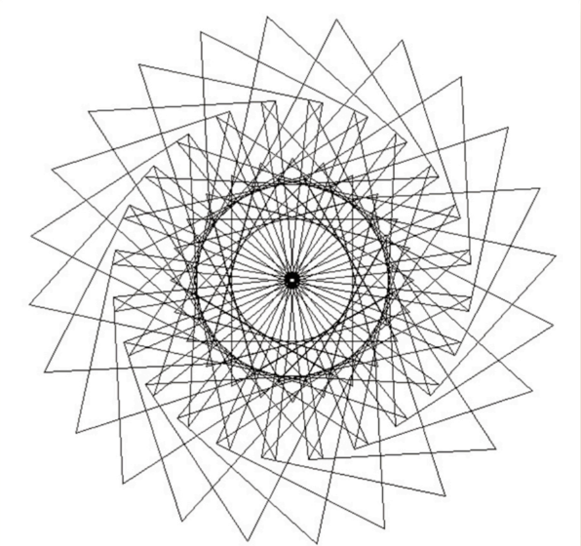
For this example n has to be chosen any number being at least 9.
For sequences containing both 0 and 1, the initial part of the sequence of length n is stored in an array a, and in the above for loop the adjustment of the variable h is replaced by

```
if a[i] = 0 then h := h + h0 else h := h + h1;
```

PERIODIC SEQUENCES
A sequence is called periodic if it is consists of infinitely many copies of the same finite sequence. For instance, an initial part of a periodic sequence is obtained by

```
    for i := 1 to n do
        if (i mod 9) in [0,2,3,4] then a[i] := 0 else a[i] := 1;
```

and by choosing `h0 = 120 and h1 = -147` this yields the following turtle figure:



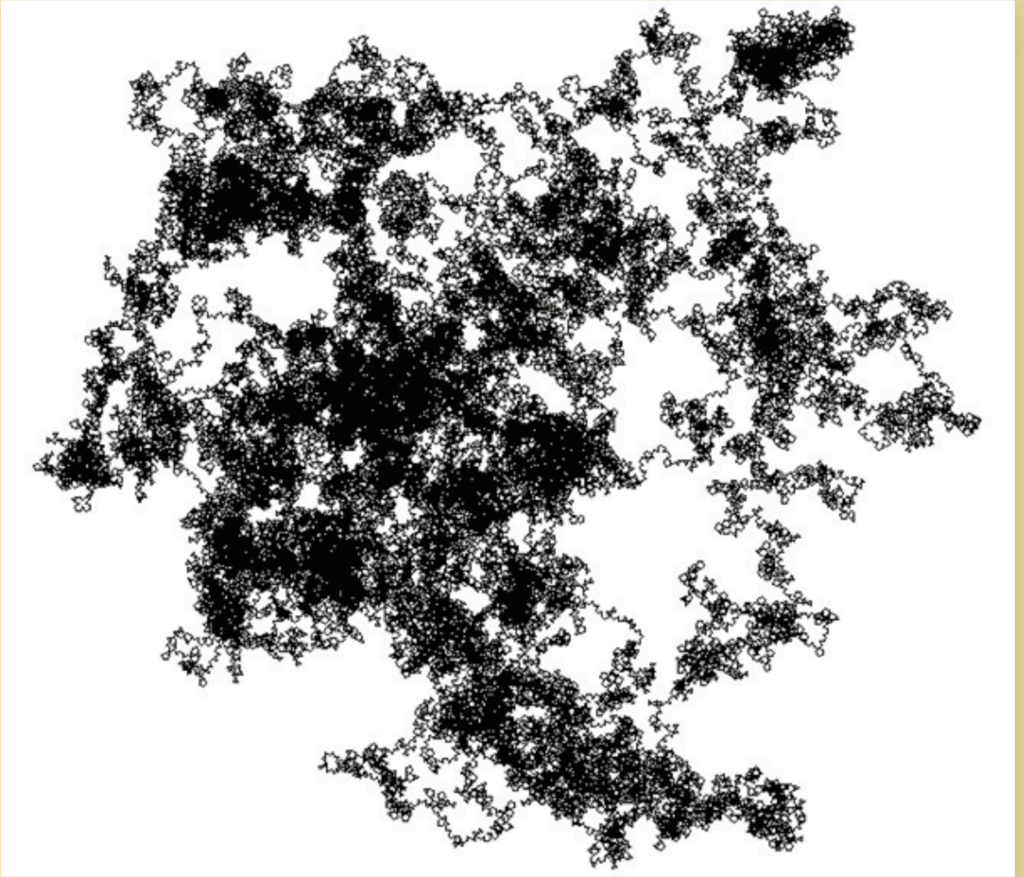
Here the turtle figure of the copied finite sequence `100011110` should be drawn 9 times in order to be back at the initial point and angle, so `n` should be chosen at least 216.

MORE COMPLICATED SEQUENCES

Periodic sequences are very structured. Sequences without any structure are obtained by randomly choosing a[i], for instance by

```
for i := 1 to n do a[i] := random(2);
```

As expected, then the resulting turtle figure does not show any structure, and typically look as follows:

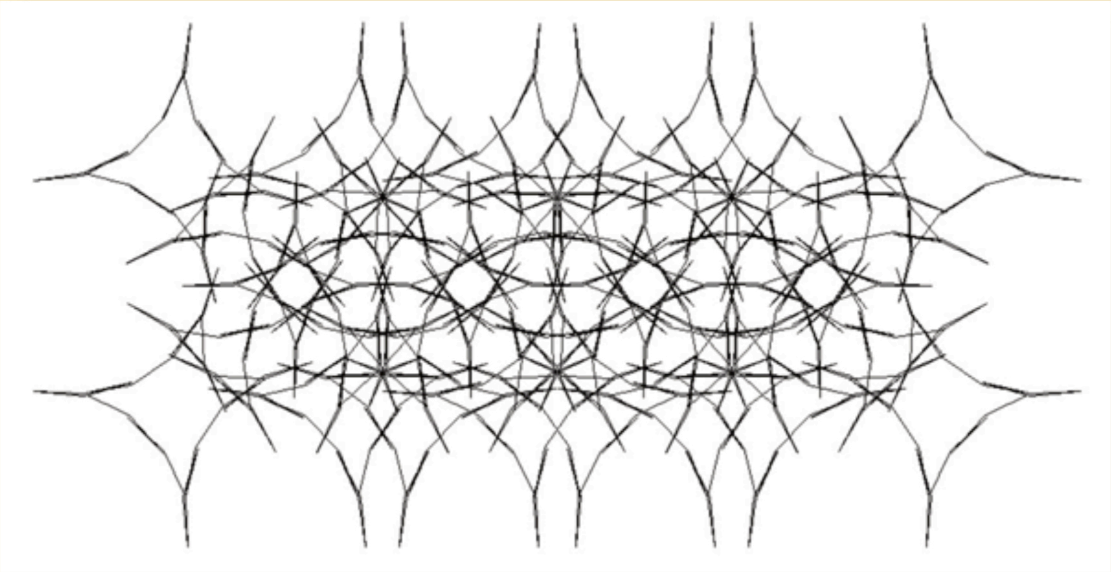


THE THUE-MORSE SEQUENCE

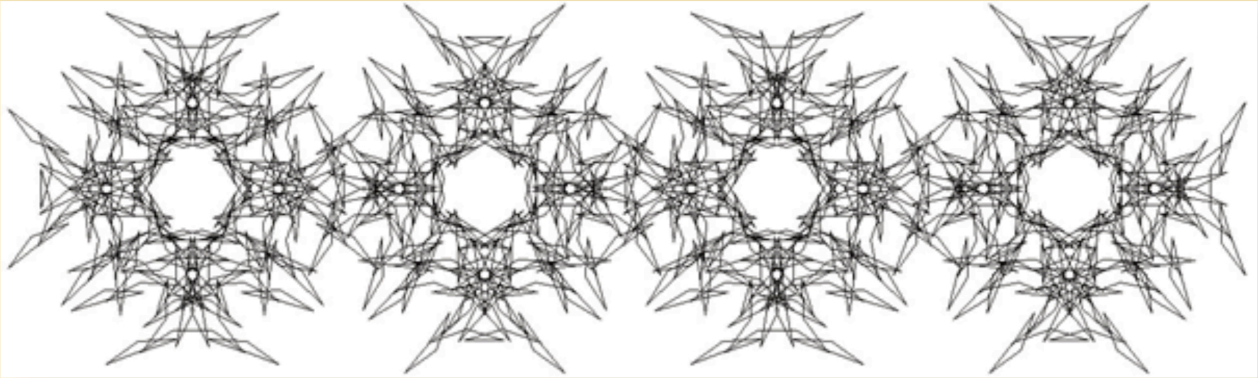
More interesting turtle figures are obtained from morphic sequences, defined by the property that they yield itself by applying a particular morphism f, that is, mapping every 0 to a finite sequence f(0) and every 1 to a finite sequence f(1). For instance, the Thue-Morse sequence

```
0110100110010110…..
```

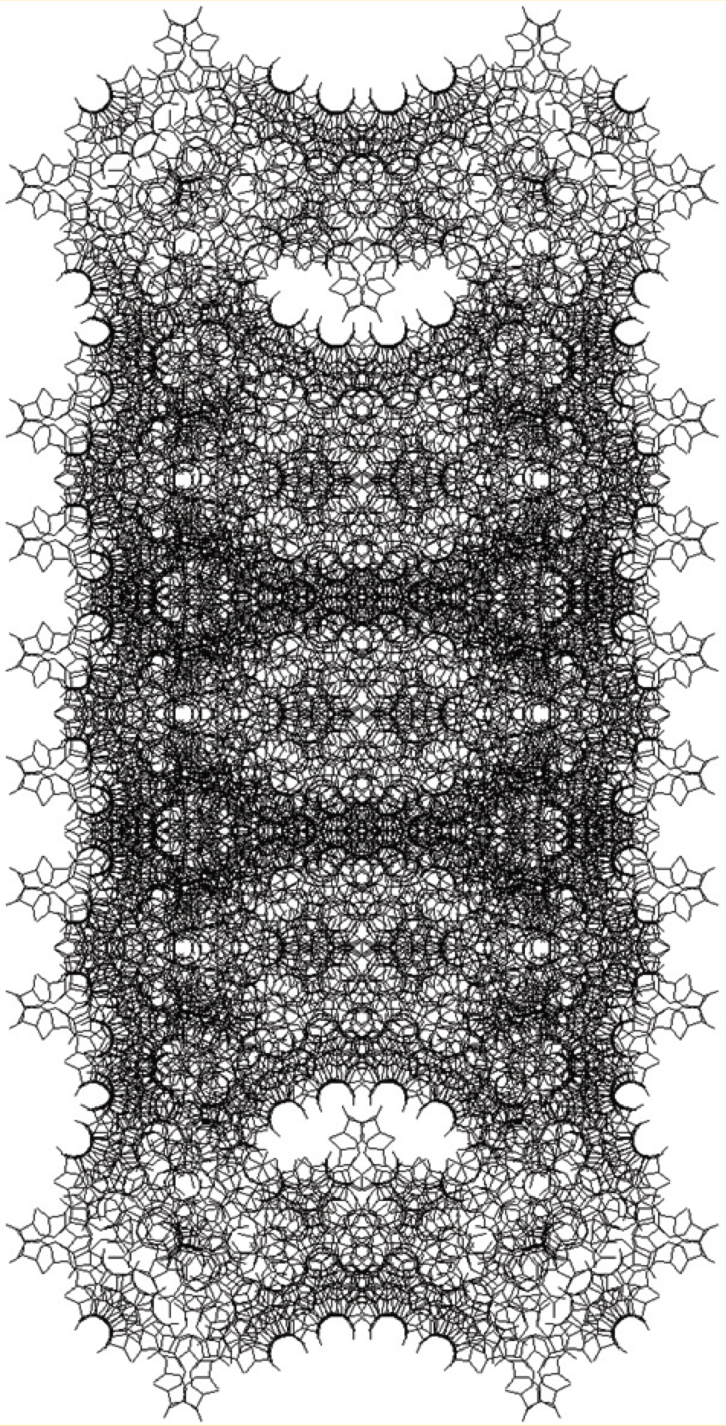
is defined to be the morphic sequence starting in 0, and maps to itself
by replacing every 0 by f(0) = 01 and every 1 by f(1) = 10.
It may be argued (for the theory we refer to [1,2]) that if 2^kh0 and 2^kh1 are multiples of 360 degrees, for some k, then the turtle figure of the Thue-Morse sequence will be finite.
It turns out that they often yield nice pictures. For instance, choosing h0 = 22.5 = 360/16
and h1 = 177.1875 = 63 x 360/128 yield the following turtle figure:



Choosing other angles also satisfying this condition yield the following turtle figures:



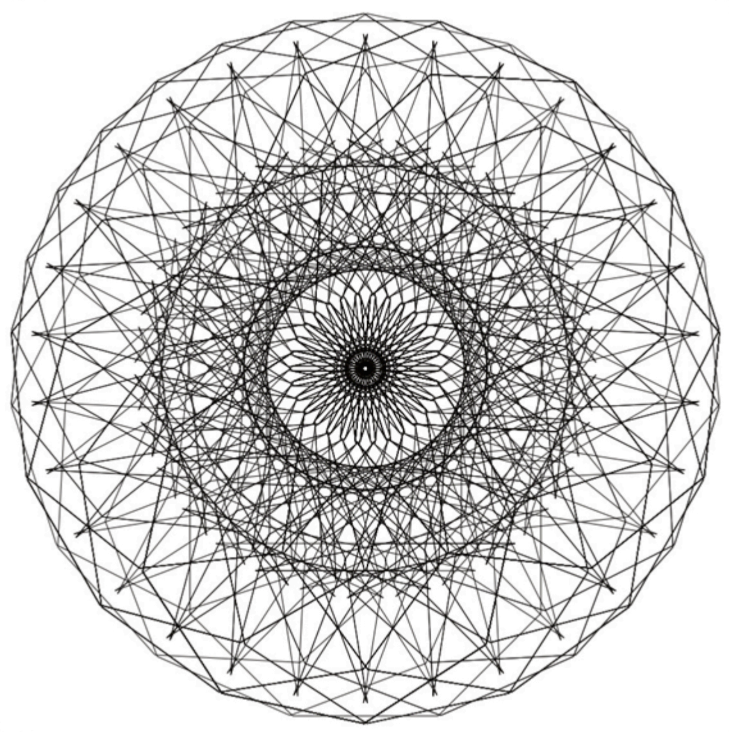
and → *(see next page)*



For all of these turtle figures of the Thue-Morse sequence the total number of drawn segments is a power of 2, for these three examples respectively 2^{10} = 1024, 2^{11} = 2048 and 2^{16} = 65536. For all these examples the number n in the program should be at least the corresponding number. Generating the initial part of any morphic sequence can be done by only a few lines of code. For instance, for the above examples the full code to draw the turtle curve may read as follows:

```
a[1] := 0;
a[2] := 1;
i    := 2;
j    := 1;
while i < n do
  begin
    i := i+1; j := j+1;
    if a[j] = 0
    then
      begin
        a[i] := 0;
        i    :=i+1;
        a[i] := 1;
      end
    else
      begin
        a[i] := 1;
        i    :=i+1;
        a[i] := 0;
      end;
  end;
for i := 1 to n do
  begin
    if a[i]  := 0
    then h   := h+h0
    else h   := h+h1;
    x := x + u * cos(h);
    y := y + u * sin(h);
    lineto(round(x),round(y));
  end;
```

All three examples we gave (and many more) are obtained by different initializations of the variables n, h0, h1, h, x, y and u.
Essentially the picture is obtained by the choice of n, h0 and h1; the initial values of h, x, y and u are typically chosen after playing around with these values until the resulting picture nicely fits on the screen.



OTHER MORPHIC SEQUENCES

By choosing other finite sequences f(0) and f(1), in which f(0) should start in 0 and consists of at least two symbols, we obtain other morphic sequences, giving rise to many more exciting turtle figures.

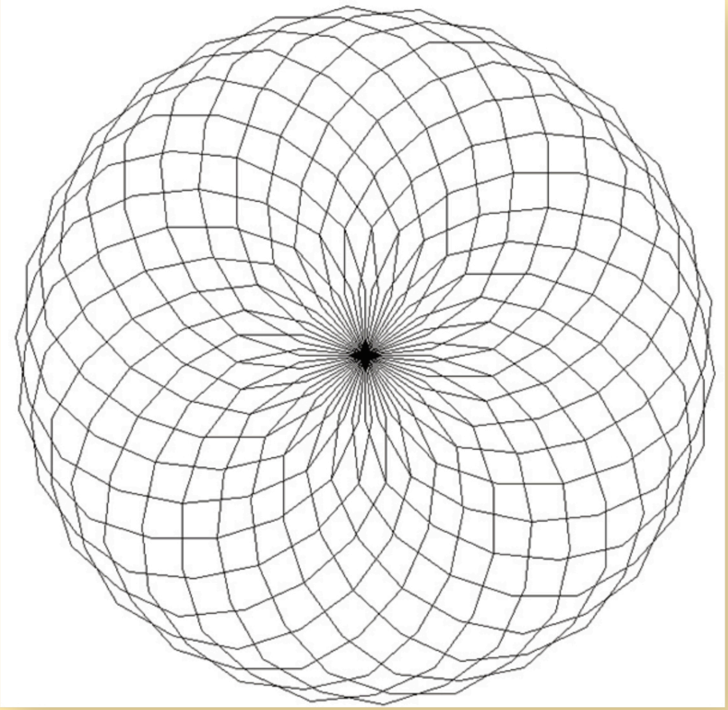
Some of them are finite, just like the examples we gave for the Thue-Morse sequence.

Again for the underlying theory we refer to [1,2]. As a first example we consider f defined by f(0) = 010 and f(1) = 11, yielding the morphic sequence

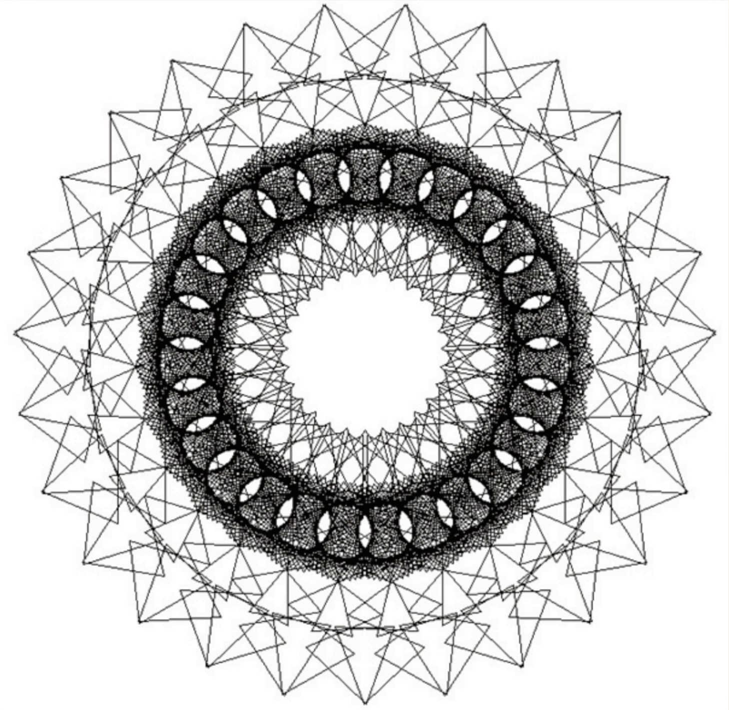
```
    010110101111010110 10....
```

being the unique sequence starting in 0 and mapping to itself when simultaneously every 0 is replaced by f(0) = 010 and every 1 by f(1) = 11. Choosing h0 = 168° and h1 = -45° and n sufficiently large yields the turtle figure shown on the left top of this page. More precisely, this turtle figure consists of 840 segments.

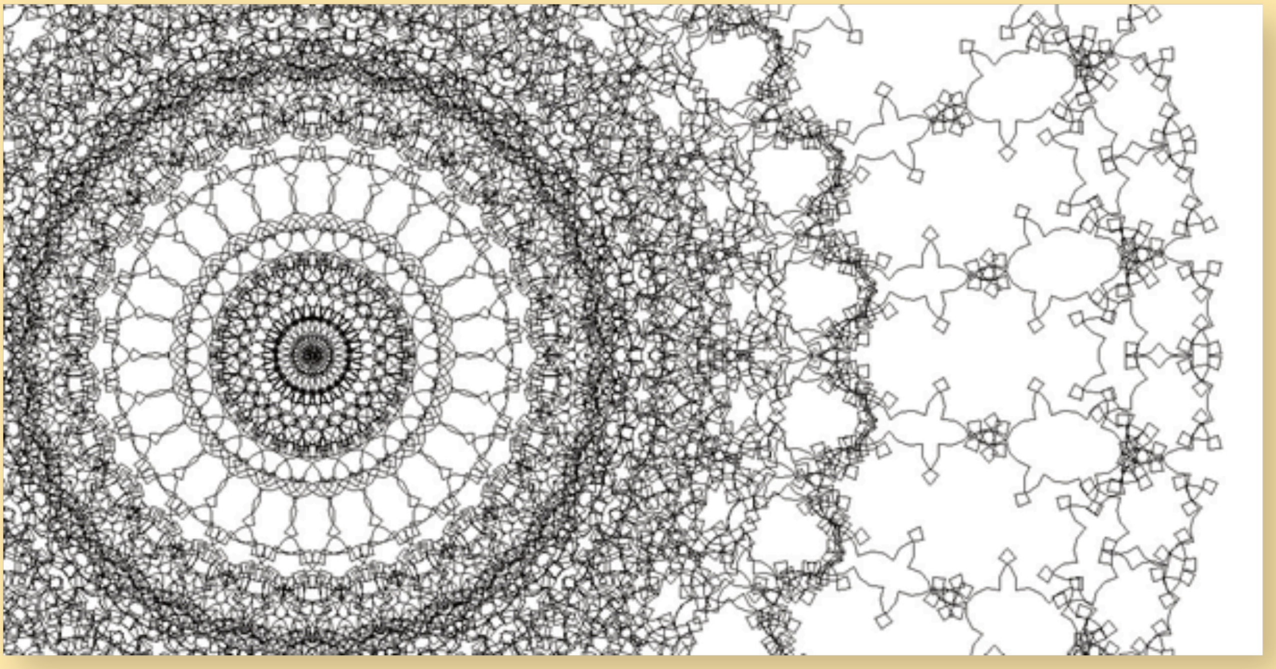
When choosing n = 840 not all segments will be drawn since then already some segments are doubly drawn, but choosing n = 1300 will yield the full picture: then continuation will only overdraw existing segments.



Other choices of f(0), f(1), h0 and h1 yield the turtle figures shown left below, and on top of the next page.

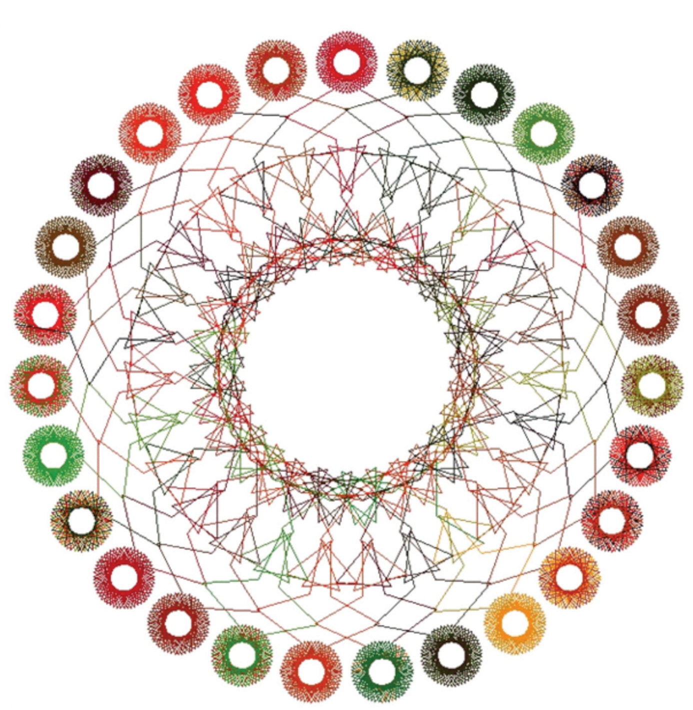


For all these examples the program is just a minor modification of the program we gave for the Thue-Morse sequence: only the first loop is modified to generate the desired morphic sequence.
Further, the angles h0 and h1 have to be chosen such that the requirements for theorems concluding finiteness from [1], [2] are met,
and the initial values of h, x, y and u have to be chosen such that the resulting figure nicely fits on the screen.
By choosing u too large on purpose, the resulting figure zooms in, like in the example shown below.



A Lazarus program with a user interface to enter all parameter to create all turtle figures we gave, including all sources and several examples, is available at
`https://github.com/hzantema/turtle-graphics`

Until now the pen color was not specified, by which it is the default being black. By modifying the pen color in the second loop for increasing i, pictures like the following may be obtained:



In all these examples every segment is drawn on the canvas by the command

```
   lineto(round(x),round(y));
```

For only showing the picture on the screen this works well, but it does not provide high resolution due to the rounding.
For printing in high resolution or offering the facility to zoom in, one should switch from this bitmap approach to a vector based approach, simply by removing the round commands and replacing the lineto command to the corresponding command in a vector based setting.
Then a compilation of examples like presented in this paper may look as follows:
(*see the figure on article page 9*)

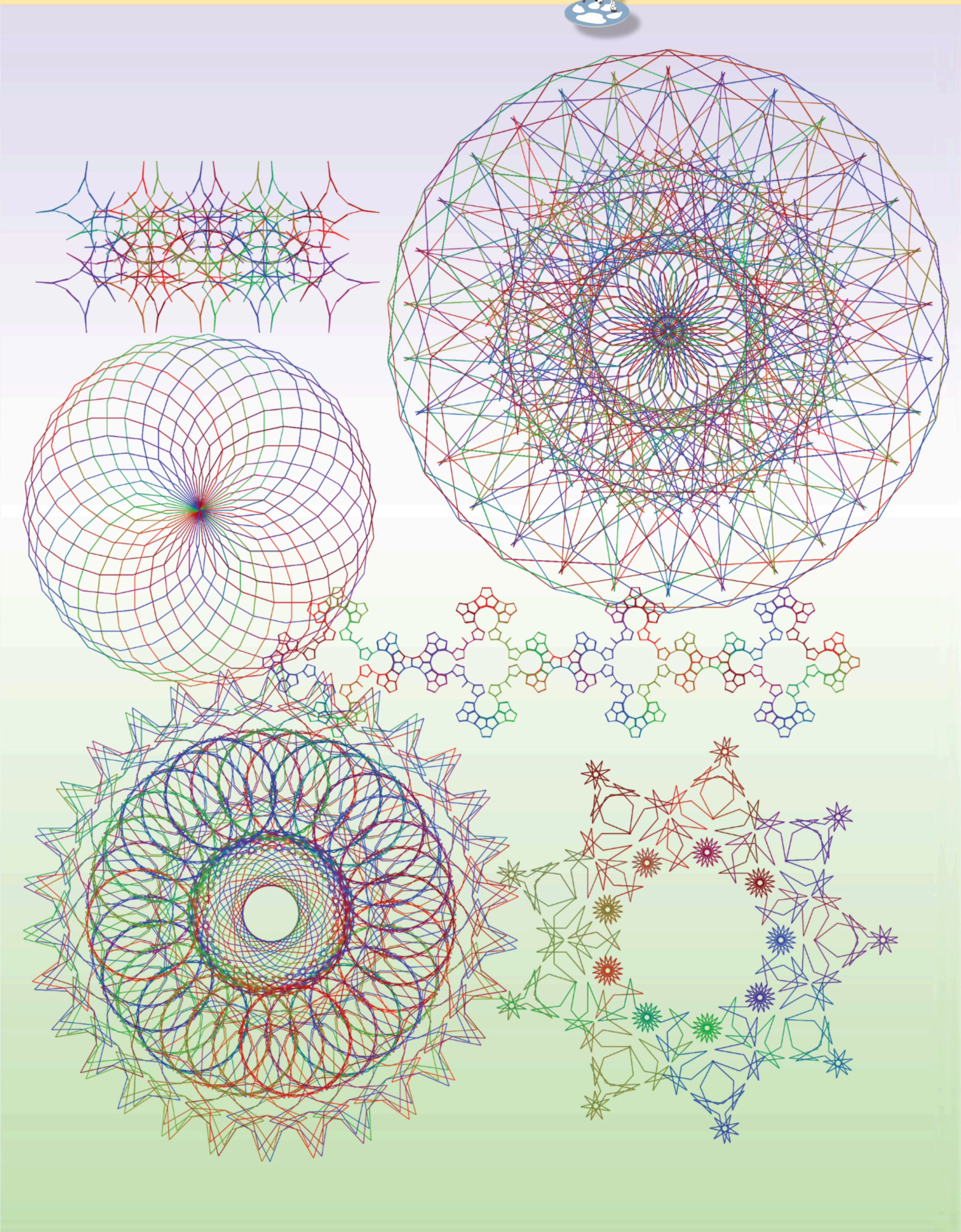
CONCLUSION

In this paper we showed how several remarkable figures may be drawn by a freely available Lazarus program in which the underlying code is of a very limited size.

REFERENCES

[1] H.Zantema, Turtle graphics of morphic sequences,
Fractals, 2016, volume 24, number 1, preliminary version available at
`https://www.win.tue.nl/~hzantema/turtle.pdf`

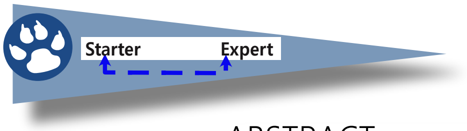
[2] H. Zantema, Playing with infinity: turtles, patterns, and pictures. To appear in 2024 at CRC Press, Taylor and Francis group, around 250 pages. Translated from Dutch version Spelen met oneindigheid: verrassende figuren en patronen, Noordboek 2023



Blaise Pascal

KEEPING UP WITH DELPHI IN FPC

BY MICHAEL VAN CANNEYT



ABSTRACT

Delphi compatibility has always been important for Free Pascal.
However - in certain respects - this compatibility was lacking.
Recently, lots of work has been done to **improve the Delphi-compatibility of FPC**.

1 INTRODUCTION

The **Free Pascal** compiler team has always regarded **Delphi** compatibility as an important feature of the **Free Pascal** compiler. This compatibility applies first and foremost to the **Pascal** language: if **Delphi** can compile it, then so should **Free Pascal**.

For a long time, **Delphi 7** was the version to which **Free Pascal** was most compatible.
But obviously, **Delphi** evolved:
Modern Language features were added such as generics, anonymous functions, attributes and extended RTTI.

These features have been developed in **FPC** - some since a longer time, some recently, but many of these features have not yet been incorporated in an officially released version of **Free Pascal.**
To use them, you must **use the development version of FPC**.

The **Delphi** compatibility also applies to the basic units that are distributed with **Free Pascal:**
The basic RTL units of **Delphi** can also be found in **Free Pascal**.

Delphi and **Free Pascal** of course evolve over time, and some of the differences in the RTL units make it difficult to keep code working in both **Free Pascal** and **Delphi**:

- Some basic units are not present.
  The number of units in the **Delphi RTL** has expanded considerably.
  Basic functionality is offered in `System.IOUtils`,`System.JSON`, `System.Threading` etc.
  These units have been added to FPC recently.
- Delphi switched to using dotted (namespaced) units:
  the prefixes `System`, `Data`, `Vcl`, `FMX` are used throughout the `Delphi` codebase since many years.
- **Delphi** has since many years changed the '`String`' alias type so it is an alias for a `UnicodeString`: a string type in which each element of the `string` (a character) is a UTF-16 character - it uses 2 bytes.

Needless to say, the **Free Pascal** team always needs to play catch-up with **Delphi** - the requirement of compatibility is a one-way street.

This has posed a dilemma for the FPC team:
In the first place, **FPC wishes to remain backwards compatible** with itself.
Something which is considered equally (*if not more*) important than *Delphi* compatibility.
Clearly, switching the string type and starting to use namespaces in the unit names renders the code backwards incompatible.

HOW TO SOLVE THIS ?

2 THE SOLUTION

After many years, a solution for this dilemma has been implemented.
The idea is simple: use the same codebase to recompile all **FPC RTL** and **Packages** code twice.

- **One compile** is done with settings which are **backwards compatible with FPC** itself: no dotted names, string is the 1-byte string.
- A second compile is done with settings that will generated dotted names, and in which string is the 2-byte string.

This results in 2 sets of units, which cannot be mixed. The user needs to choose which set of units he wishes to use:

- The **Free Pascal backwards-compatible set of units**,
- The **"Recent Delphi"-compatible set of units.**

The latter has been dubbed the '`unicode rtl`'.

There is a third option, which is to compile everything with a personal set of settings: more about this later.

To put this solution in effect, some compiler work was needed:

- The compiler had to change its view on what constitutes the basic `Char` type – till recently this was hardcoded as a 1-byte character and `AnsiChar` was an alias for `Char`.

  Now, `AnsiChar` is the a basic type, and Char is an alias defined in the system unit. This allows to let the keyword String be either an `AnsiString` or a `UnicodeString` and `Char` will always match the type of a single character in a `String`.
- A compiler 'target' was till recently defined as a combination of the **target CPU** and **the target OS**.

  This means that for example the i386-win32 and x86 64-win64 are 2 windows platforms: one 32-bit, one 64 bit.
  Linux knows even more platforms, just as Darwin (macos) has already 3 platforms.

  This definition of platform needed to be extended, and the notion of 'SubTarget' was introduced in the compiler.

Both changes have been instrumental in making it possible to create the 2 sets of units for the end-user.

3 THE SUBTARGET

The notion of a compiler target was extended:
it needs to encapsulate a CPU, an OS and optionally also an arbitrary set of settings.
For the **Delphi**-compatible unicode rtl this "arbitrary set of settings" would be:

- use **UTF16** strings.
- use **dotted** names.

Defining a subtarget simply means we give a name to this set of settings.
The subtarget is always specified to the compiler with a command-line switch: **`-t`**.

So if you wished to compile a file for subtarget '**unicodertl**', you would specify:

```
fpc -Mdelphi -tunicodertl myproject.pas
```

The subtarget name is then used in various ways:
The first way the subtarget name is used, is in the compiler.
When the compiler is compiling for a certain target, it defines a macro `fpctarget.`

The value of this macro is the combination of target CPU and OS, separated by a dash.

Continuation chapter 3

This macro is used in the configuration file of the compiler:

```
-Fu/usr/local/lib/fpc/$fpcversion/units/$fpctarget
-Fu/usr/local/lib/fpc/$fpcversion/units/$fpctarget/*
-Fu/usr/local/lib/fpc/$fpcversion/units/$fpctarget/rtl
```

So when compiler version 3.1.1 is compiling for **i386-linux**,
the compiler 'sees' the following configuration:

```
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linux
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linuxt/*
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linux/rtl
```

Now, the optional 'Subtarget' is introduced, and one of the effects of using it is that it adds the name of the subtarget to the fpctarget macro.
That means that when compiler version 3.1.1 is compiling for **i386-linux** and subtarget 'unicodertl', the configuration becomes

```
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linux-unicodertl
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linuxt-unicodertl/*
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linux-unicodertl/rtl
```

The search paths of the compiler are suddenly different depending on the subtarget.

As said before, the subtarget is a name of a set of settings.
This is made concrete by the second effect of using subtarget,
and also explains why it needs to be set on the command-line:
When looking for the compiler configuration, the compiler will also look for a configuration file which has the subtarget included in its name.

When compiling with subtarget `unicodertl`

```
fpc -Delphi -tunicodetrl myproject.pas
```

The compiler will - in the same places where it looks for `fpc.cfg` - also look for a file called

```
fpc-unicodertl.cfg
```

On Unix-like platforms, it will also look for the usual "hidden" file in the user's home dir:

```
.fpc-unicodertl.cfg
```

This second configuration file contains the set of settings that defines the subtarget.

There are 2 things to note about this extra configuration file:

❶ It must exist. If it does not exist, the compiler will display an error (*however, it can be empty*).

❷ It is always loaded even if the option to not load the default configuration files (**`-n`**) is specified.

So, how is this used to create a **Delphi** compatible rtl ?
A configuration file with the name `fpc-unicodertl.cfg` is created with the following contents:

```
-dUNICODERTL
-Municodestrings
```

This configuration file is then used to compile the **RTL** and packages.
It defines **UNICODERTL** and it specifies the unicodestrings modeswitch, which instructs the compiler that String must be interpreted as `UnicodeString`.

Of course, many units contain some low-level code where the size of the `Char` type is very important.
The sources have been changed where needed so that the code compiles and functions correctly with both definitions of the String and `Char` types.

Continuation chapter 3

Not only the sources were changed, but also the compilation and installation process was slightly modified:
We saw earlier that when specifying `-tunicodertl` on target **i386-linux,** the compiler will look for compiled units in the following directories:

```
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linux-unicodertl
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linuxt-unicodertl/*
-Fu/usr/local/lib/fpc/3.3.1/units/i386-linux-unicodertl/rtl
```

That means that the units must also be installed there when installing the compiled units.
The makefiles for the **RTL** and Packages have been adapted to cater for this.
There is now a variable `SUB_TARGET` which configures the `Makefiles` to call the compiler with the specified subtarget. When installing units, the subtarget name will be appended to the directory name where the units are installed.

4 USING NAMESPACES IN FILENAMES

To create units with namespaces in it and the same units but without namespaces, a small trick is used.
It should be clear that the **Free Pascal team** cannot maintain 2 sets of units.

The solution put in place uses 1 set of files, but uses an include mechanism to create the second (*namespaced*) file.
As an example, here is the file for the '`System.Math`' unit:

```
 unit System.Math;

{$DEFINE FPC_DOTTEDUNITS}
{$i math.pp}
```

The `math.pp` file is changed

```
{$IFNDEF FPC_DOTTEDUNITS}
unit Math;
{$ENDIF FPC_DOTTEDUNITS}

interface
uses

{$IFDEF FPC_DOTTEDUNITS}
  System.SysUtils;
{$ELSE FPC_DOTTEDUNITS}
  sysutils;
{$ENDIF FPC_DOTTEDUNITS}
```

NOTE THAT THE USES CLAUSE IS DIFFERENT.

More changes are of course needed:
For example, whenever a fully qualified identifier is used (*i.e. an identifier which includes the unit name*) the name had to be corrected.

This system allows the **FPC** team to compile an **RTL** with namespaced filenames, and a **RTL** with backwards-compatible filenames.

The above exercise has been done for all units distributed by **FPC**:

For every unit,a namespaced version has been put in place. For units that exist in **Delphi**, the **namespaced filename is identical** to the name in **Delphi**.
For units that do not have an equivalent in **Delphi**, the opportunity has been taken to make the filenames more consistent.

All makefiles and fpmake programs in the **Free Pascal** distribution have been adapted so they will compile the namespaced units when the `FPC_DOTTEDUNITS=1` option is given on the make command-line.

Continuation chapter 4

Till now the policy of the **Free Pascal team** has been to **lowercase** the unit filenames on **case sensitive filesystems**. It meant that you can be sloppy in the uses clause:
the case of the unit name did not matter, since the compiler looked for a lowercased version of a file in addition to looking for a file with the same case as used in the uses clause.
This practice has been abandoned for namespaced units: The **unit name must now have the correct case on case-sensitive filesystems**.

HOW TO FIND THE NAMESPACED VERSION OF A UNIT NAME?
There is a file that specifies for each non-namespaced unit the namespaced name of the unit.
This file can be used to change the unit names.
You don't need to do this manually, there is a tool that will do this for you.
More about this tool later.

The astute reader will have noticed that generating namespaced units is not done using the newly introduced concept of subtargets. Indeed, the 2 features are orthogonal.
The reason is that there is no 'special' subtarget. You **create as many subtargets as you want, and for each subtarget you can create a namespaced version of the RTL or a backwards-compatible version.**

This means it would be possible to generate a **RTL** without namespaced units but with String equal to `UnicodeString`, or a namespaced **RTL** with `String=AnsiString`.

The choice of the **FPC** team is determined by the requirement of having a **FPC** compatible **RTL** and a **Delphi-compatible RTL.**

5 CREATING THE DELPHI COMPATIBLE RTL

When the next major release of **Free Pascal** is released, the **FPC team** will create the 2 **RTLs** mentioned above.
But you can already enjoy this **increased Delphi compatibility** today by compiling the compiler and **Delphi-compatible RTL** yourself.

HOW TO GO ABOUT IT?
The first thing to do is to install **git** and download the compiler sources.
This mechanism has been treated in depth in previous articles.

Assuming **git** is installed and in your PATH, in a terminal window (*on the command line*) you can clone the **FPC** sources:

```
git clone https://gitlab.com/freepascal.org/fpc/source.git fpc
```

Once done, you can compile and install the latest FPC:

```
cd fpc
make clean all PP=/path/to/FPC/3.2.2
make install PP=/path/to/FPC/3.2.2
```

The **PP** variable must point to the **fpc binary** of **FPC** version 3.2.2 (*this is a requirement*).
Where this **fpc binary** is installed, depends on your system.
The above will compile and install version 3.3.1 of **FPC**.
You can also use FPCUpdeluxe for this.

Once this is done, **you should note** where the new version of **FPC 3.3.1** was installed.
The next step is to create the configuration file `fpc-unicodertl.cfg` in one of the locations where the compiler looks for the configuration file:

```
-dUNICODERTL
-Municodestrings
```

The easiest strategy is to look for the existing `fpc.cfg` and to create the above file right next to it.

Continuation chapter 5

Then, in the same terminal where the previous commands were typed, the following make commands will create and install the unicode rtl.
These commands must be executed in the same directory as the previous 'make' commands.

```
make -C rtl clean all PP=/path/to/FPC/3.3.1 SUB_TARGET=unicodertl FPC_DOTTEDUNITS=1
make -C rtl install PP=/path/to/FPC/3.3.1 SUB_TARGET=unicodertl FPC_DOTTEDUNITS=1

make -C packages clean all PP=/path/to/FPC/3.3.1 SUB_TARGET=unicodertl FPC_DOTTEDUNITS=1
make -C packages install PP=/path/to/FPC/3.3.1 SUB_TARGET=unicodertl FPC_DOTTEDUNITS=1
```

Once this is done, you can compile recent Delphi code using the new '`-tunicodertl`' command-line switch.

❻ CONVERTING CODE TO USE NAMESPACED UNITS

WHAT IF YOU WISH TO UPDATE YOUR CODE AND MAKE USE OF THE DELPHI-COMPATIBLE RTL?

As shown on *page 5 of this article*, the **uses clause** of a project must be changed.
If you have a lot of units, this means a lot of work.
Luckily, the tools that were used to make the dual-use **FPC** units are available to you.
In the **FPC source tree,** in directory `utils/dotutils` there is a program called **prefixunits**.
You can compile it on the commandline or using **Lazarus**, and use it to convert units to the newly used mechanism.
The following **command-line** will convert unit `myunit.pas` to `company.myunit.pas` using the same mechanism as shown above:

```
prefixunits -b -k known.txt c:\Temp\myunit.pas -n company
```

A new file `varcompany.myunit.pas` will be written which includes `myunit.pas`.
The uses clause in `myunit.pas` will then be rewritten using the conditional define as shown above.
The **`-b`** option tells the program to make a backup, the **`-k`** option must be used to point to the file which **maps the old unit names** to the **new names.**
This file is present in the `dotutils directory`, next to the sources of the prefixunits tool.

You can also decide to simply use only the new unit names.
Then you specify the '**`-r`**' flag (for 'replace'):

```
prefixunits -r -b -k known.txt c:\Temp\myunit.pas -n company
```

In that case, no new file is created, and the uses clause in `myunit.pas` will be replaced with a unit clause that uses **only the dotted unit names**.

This **tool is somewhat rudimentary** but does what it is designed to do, and no doubt in time the **Lazarus IDE** will be extended with a nice **GUI tool** which performs the same task for all files in your project.

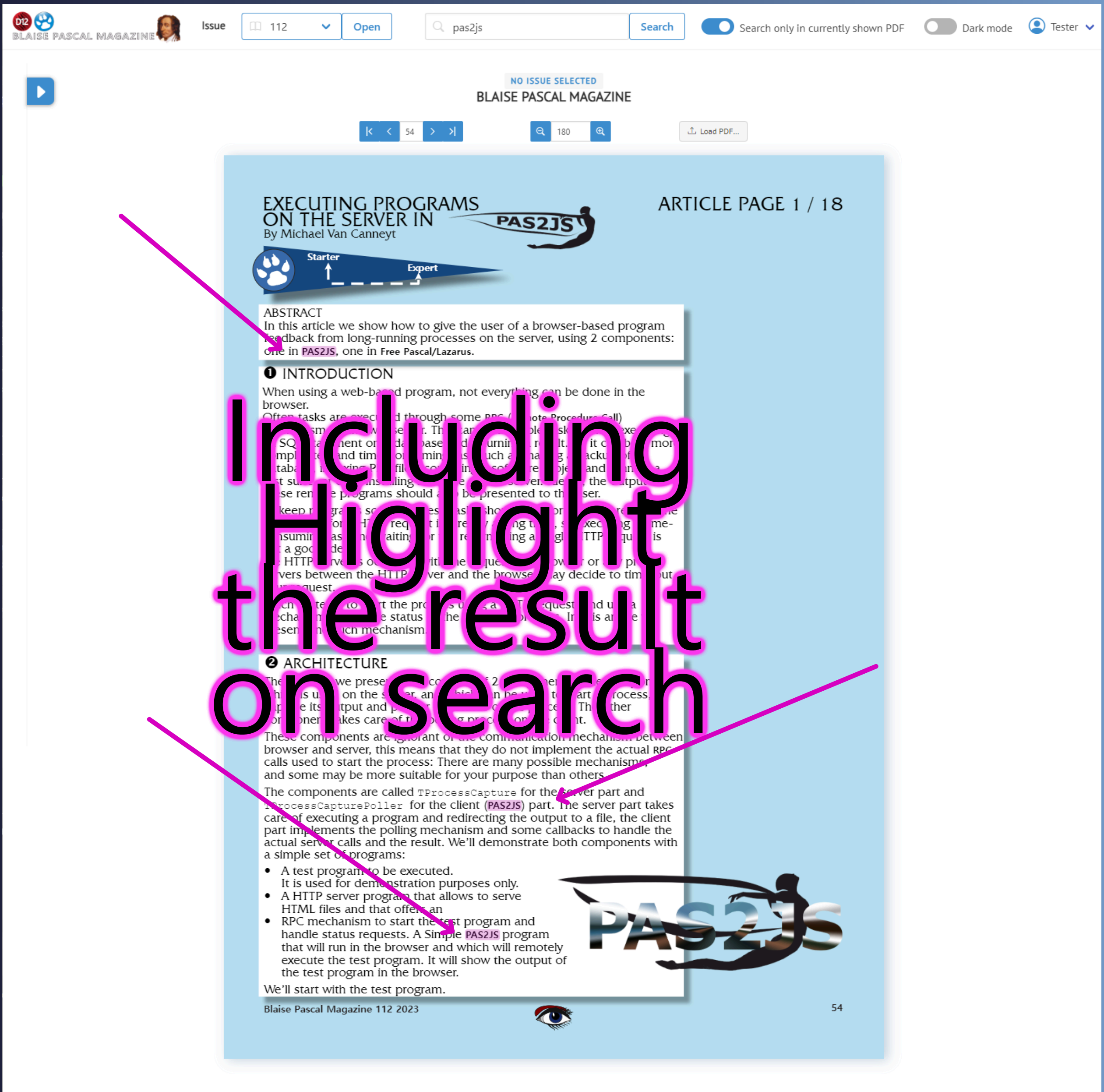
❼ WHAT ABOUT PAS2JS ?

For **PAS2JS**, the **same namespacing operation** has been performed.
The **`-t`** option has also been added to the transpiler command-line options, so the working of the two compilers is the same.
To improve the **Delphi** compatibility, support for the **Delphi 12 multiline string has been added** in addition to the **already existing multiline string** support in PAS2JS

8 CONCLUSION

A lot of work has been put into making **FPC** more delphi compatible: new language features, dotted unit names.
The work was largely sponsored by a company that wishes to compile its **Delphi** program with **Free Pascal** without having to change the sources of the program. As a result, the **Delphi** compatibility of **Free Pascal** has received a boost and all users of **Free Pascal** can now enjoy an improved **Delphi** compatibility. Unlike the various transitions done in **Delphi** which could break backwards compatibility, the **FPC** team decides that **FPC REMAINS BACKWARDS COMPATIBLE WITH ITSELF**.
As a result, users can now choose whether they use these more recent units or not: the choice is always theirs.

LIB-STICK ON USB CREDIT CARD
BLAISE PASCAL MAGAZINE



EXECUTING PROGRAMS ON THE SERVER IN PAS2JS
By Michael Van Canneyt

ARTICLE PAGE 1 / 18

ABSTRACT
In this article we show how to give the user of a browser-based program feedback from long-running processes on the server, using 2 components: one in PAS2JS, one in Free Pascal/Lazarus.

❶ INTRODUCTION

When using a web-based program, not everything can be done in the browser.
[illegible]

❷ ARCHITECTURE

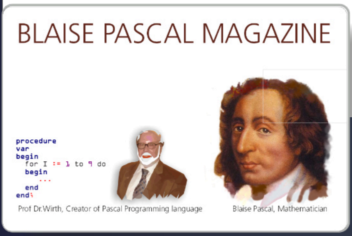
[illegible] browser and server, this means that they do not implement the actual RPC calls used to start the process: There are many possible mechanisms, and some may be more suitable for your purpose than others.

The components are called `TProcessCapture` for the server part and `TProcessCapturePoller` for the client (PAS2JS) part. The server part takes care of executing a program and redirecting the output to a file, the client part implements the polling mechanism and some callbacks to handle the actual server calls and the result. We'll demonstrate both components with a simple set of programs:

- A test program to be executed. It is used for demonstration purposes only.
- A HTTP server program that allows to serve HTML files and that offers an
- RPC mechanism to start the test program and handle status requests. A Simple PAS2JS program that will run in the browser and which will remotely execute the test program. It will show the output of the test program in the browser.

We'll start with the test program.

Blaise Pascal Magazine 112 2023 54



BLAISE PASCAL MAGAZINE

```
procedure
var
begin
  for I := 1 to ¶ do
  begin
    ...
  end
end;
```

Prof Dr.Wirth, Creator of Pascal Programming language

Blaise Pascal, Mathematician

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands

Prof Dr.Wirth, Creator of Pascal Programming language

editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu

BLAISE PASCAL MAGAZINE

```
procedure
var
begin
  for I := 1 to ¶ do
  begin
    ...
  end
end;
```

Prof Dr.Wirth, Creator of Pascal Programming language

Blaise Pascal, Mathematician

LIB-STICK USB-CARD: ALL ISSUES / CODE INCLUDED. SAME INTERFACE AS THE INTERNET LIBRARY € 89 CHRISTMAS OFFER

ADDING A TEXT LAYER TO PDF FILES UNDER LINUX BY HELMUT ELSNER

ABSTRACT

For most printers under **LINUX**, there are no tools supplied by the manufacturer to obtain searchable PDF files when scanning documents. The texts are only embedded as images in the PDFs. This makes it difficult to find something in your scans.
Of course, there are also corresponding tools under LINUX, but these can often only be operated from the command line. "**PDFsandwich**" is one of them and a nice one at that.
We want to build a **GUI** for this tool to add a text layer to one or more PDF files with just a few clicks.

PREPARATION

Install needed tools:

❶ **ImageMagick** (often already part of the **LINUX** distribution)
❷ **pdfsandwich**
❸ **tesseract-ocr**

```
sudo apt install pdfsandwich tesseract-ocr-de -y
```

Give ImageMagick read|write rights for PDF files:

```
sudo nano /etc/ImageMagick-6/policy.xml
```

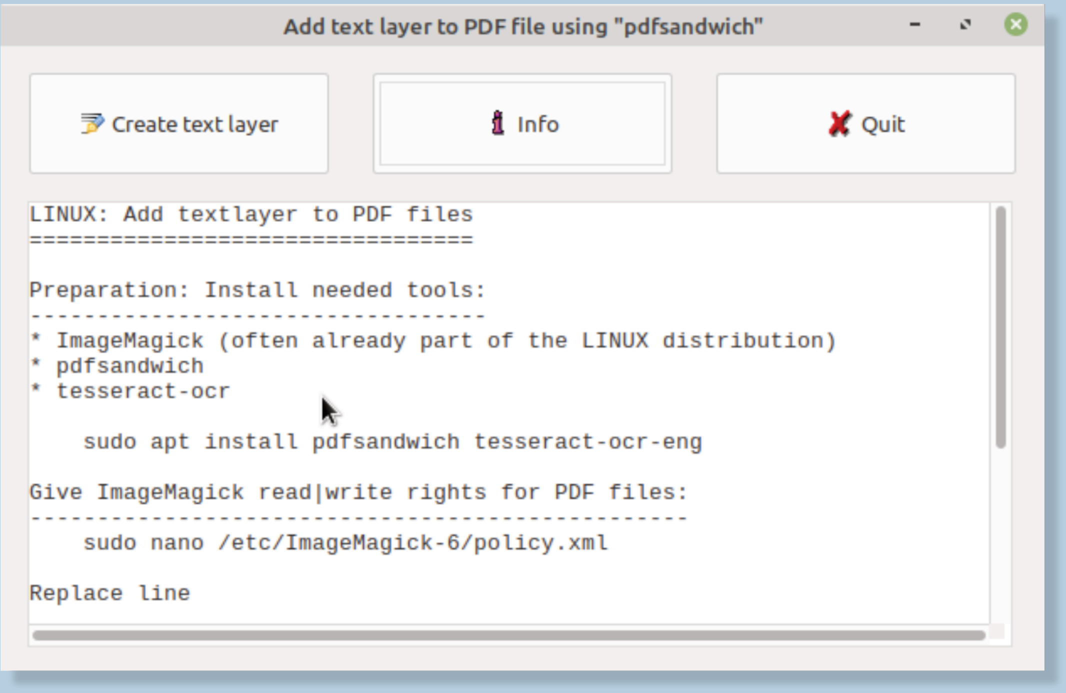
Replace line `<policy domain="coder" rights="none" pattern="PDF" />`
with `<policy domain="coder" rights="read|write" pattern="PDF" />`

save with **Ctrl+O close** with **Ctrl+X**.

Unzip the downloaded archive and put it somewhere in your home directory, for example in a directory **`/tools`** if you want create a .desktop file for it.

PDFTXT USAGE

This **GUI** is simple and (*hopefully*) self-explanatory:



```
LINUX: Add textlayer to PDF files
=================================

Preparation: Install needed tools:
----------------------------------
* ImageMagick (often already part of the LINUX distribution)
* pdfsandwich
* tesseract-ocr

    sudo apt install pdfsandwich tesseract-ocr-eng

Give ImageMagick read|write rights for PDF files:
-------------------------------------------------
    sudo nano /etc/ImageMagick-6/policy.xml

Replace line
```

With "Create text layer" you are prompted to select PDF files for processing. You can select several PDF files and process them all in one go.
It's also possible to drag and drop PDF files from the file manager onto the program window and process them all.
The result of processing is saved in the same directory as the original file in a file indicated with "_ocr" in the file name.

STRUCTURE OF THE PROGRAM

Simple help function:
Actually, we would only need a button to execute the action, but a little more information is always helpful. I have decided to display only a text file in a TMemo as help. Of course, we have to check whether the text file with the installation instructions is located where we expect it to be - in the program directory (Application.Location).

```
procedure TForm1.btnInfoClick(Sender: TObject); {Button Info}
begin
  if FileExists(Application.Location+infofile) then
   begin
     Memo1.Lines.LoadFromFile(Application.Location+infofile);
   end else
   begin
     Memo1.Lines.Add(infofile+errInfo);
     Memo1.Lines.Add(hntInfofile);
   end;
end;
```

PROCESS A COUPLE OF PDF FILES:

To process one or more PDF files, there is the "Create text layer" button. To be able to select more than one file, we must set the `ofAllowMultiSelect` option in `TOpenDialog`. Then we can start working through the file list.

```
procedure TForm1.btnAddTxtClick(Sender: TObject);
var
  i: Integer;
begin
  OpenDialog1.Title:=capOpenPDF; {Option ofAllowMultiSelect must be set}
  if OpenDialog1.Execute then
    begin
      Screen.Cursor:=crHourGlass;
      btnClose.Enabled:=false;
      Memo1.Lines.Clear; {Empty log output}
    try
      for i:=0 to OpenDialog1.Files.Count-1 do
        PDFAddText(Opendialog1.Files[i]);
    finally
      btnClose.Enabled:=true;
      Screen.Cursor:=crDefault;
    end;
  end;
end;
```

CALL A TERMINAL COMMAND WITH TPROCESS:

We call the processing routine for each file.
We use the file extension to check whether the files are PDF files.
This procedure creates a process (`pdftxt: TProcess;`) to execute the desired terminal command, namely "**pdfsandwich**", capture the output of the command and write it to `TMemo` as a log.
The command is also given a number of parameters, including the file name of the PDF file to be processed `(pdftxt.Parameters.Add(fn);)`.

```
procedure TForm1.PDFAddText(fn: string); {Process one PDF file}
var
  pdftxt: TProcess;
  outlist: TStringList;
  i: integer;
begin
  if Lowercase(ExtractFileExt(fn))=pdfext then
    begin
      Memo1.Lines.Add(fn);
      Memo1.Lines.Add('');
      outlist:=TStringList.Create;
      pdftxt:=TProcess.Create(nil);
      Application.ProcessMessages;
    try
      pdftxt.Executable:=app;
      pdftxt.Parameters.Add(paralang);
      pdftxt.Parameters.Add(Sprache);
      pdftxt.Parameters.Add(fn);
      pdftxt.Options:=pdftxt.Options+[poWaitOnExit, poUsePipes];
      pdftxt.Execute;
      outlist.LoadFromStream(pdftxt.Output); {Get commandline output}
    for i:=0 to outlist.Count-1 do {Append to log}
      Memo1.Lines.Add(outlist[i]);
      Memo1.Lines.Add(separ);
      Memo1.Lines.Add('');
    finally
      pdftxt.Free;
      outlist.Free;
    end;
    end else begin
      Memo1.Lines.Add(ExtractFileName(fn)+errPDF);
    end;
end;
```

DRAG & DROP:
To take full advantage of a **GUI**, **drag & drop** should also be possible.
To do this, the `AllowDropFiles` option must be set to `true` for the form.
Then we can drag any number of **PDF files** onto the program window for processing.

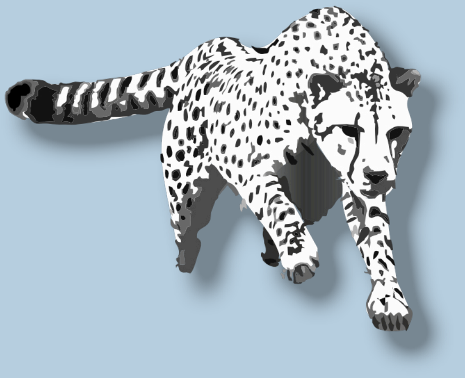
```
procedure TForm1.FormDropFiles(Sender: TObject; const FileNames:array of string);
var
  i: integer;
begin
  Screen.Cursor:=crHourGlass;
  btnClose.Enabled:=false;
  Memo1.Lines.Clear;
  Application.BringToFront; {Set focus to this program}
  try
    for i:=0 to high(FileNames) do
      PDFAddText(FileNames[i]);
  finally
    btnClose.Enabled:=true;
    Screen.Cursor:=crDefault;
  end;
end;
```

USABILITY

Of course, there is also a little help in the **Hint property** for each control element.
To change font size in `TMemo` with **mousewheel + Ctrl** we add a procedure to react to the events `OnMouseWheelDown` and `OnMouseWheelup`.

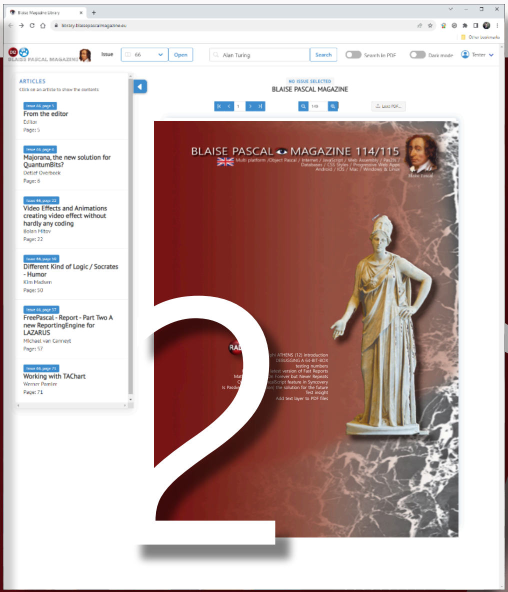
```
procedure TForm1.Memo1MouseWheelDown(Sender: TObject; Shift: TShiftState;
                                     MousePos: TPoint; var Handled: Boolean);
begin
  if ssCtrl in Shift then
    Memo1.Font.Size:=Memo1.Font.Size-1;
end;

procedure TForm1.Memo1MouseWheelUp(Sender: TObject; Shift: TShiftState;
                                     MousePos: TPoint; var Handled: Boolean);
begin
  if ssCtrl in Shift then
    Memo1.Font.Size:=Memo1.Font.Size+1;
end;
```

That's it. Using this principle, we can build a small tool with a **GUI** for all kinds of complicated terminal commands that we can't remember.
The project is here: `https://github.com/h-elsner/PDFtext`



ADVERTISEMENT

1
BLAISE PASCAL MAGAZINE 114/115
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux
Blaise Pascal



ARTICLES
Click on an article to show the contents

Issue 66, page 5
From the editor
Editor
Page: 5

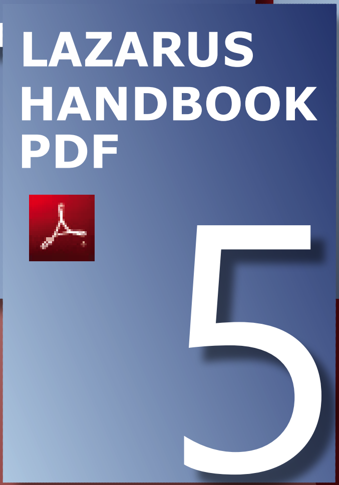
Issue 66, page 6
Majorana, the new solution for QuantumBits?
Detlef Overbeek
Page: 6

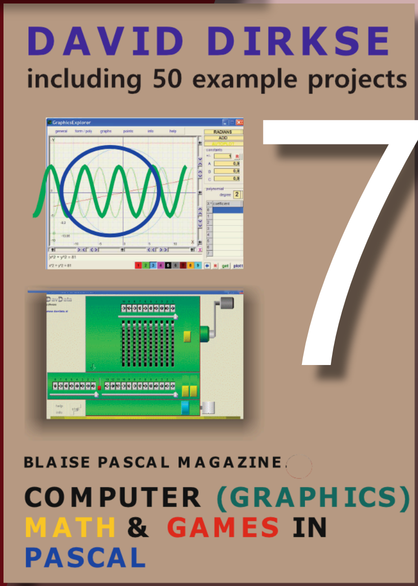
Issue 66, page 22
Video Effects and Animations creating video effect without hardly any coding
Bolan Mitov
Page: 22

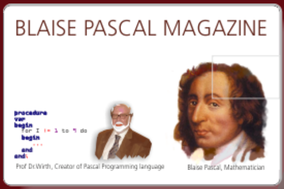
Issue 66, page 50
Different Kind of Logic / Socrates - Humor
Kim Madsen
Page: 50

Issue 66, page 57
FreePascal - Report - Part Two A new ReportingEngine for LAZARUS
Michael van Canneyt
Page: 57

Issue 66, page 71
Working with TAChart
Werner Pamler
Page: 71






BLAISE PASCAL MAGAZINE
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

BLAISE PASCAL MAGAZINE
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands

Prof Dr.Wirth, Creator of Pascal Programming language

editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu

1. One year Subscription
2. Internet Viewing of the Magazine
3. The newest LIB Stick
   - All issues 1-111
   - On Credit Card
4. Lazarus Handbook Pocket
5. LH PDF including Code
6. Book Learn To Program
   - using Lazarus PDF including
   19 lessons and projects
7. Book Computer Graphics
   Math & Games
   - PDF including ±50 projects

3

CHRISTMAS 2023
SUPER
PACK
7 ITEMS

PRICE € 120
NORMAL PRICE € 275

Donate for Ukraine and get a free license at:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

If you are from Ukrainian origin you can get a free Subscription for Blaise Pascal Magazine, we will also give you a free pdf version of the Lazarus Handbook. You need to send us your Ukrainian Name and Ukrainian email address (*that still works for you*), so that it proofs you are real Ukrainian. please send it to editor@blaisepascal.eu and you will receive your book and subscription

BLAISE PASCAL MAGAZINE

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Blaise Pascal

COMPONENTS DEVELOPERS 4

Donate for Ukraine and get a free license at:
https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

COMPONENTS DEVELOPERS 4

D11