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

Test Insight in Lazarus
Ist Passkey (Authentifizierung) die Lösung für die Zukunft?
Fehlersuche in einer 64-BIT-Box
Überblick über das PascalScript-Feature in Syncovery
Der Lazarus Debugger Teil 5: Man kann Daten ändern
Programmausführung Schritt für Schritt
Die Suche nach einer speziellen Zahl
Ein Problem mit einem rollenden Kreis
Das Prinzip der einzigen Verantwortung
Arbeiten mit firedac local SQL
Firedac-Datensatz-Aggregationen
Textauswahl und Hervorhebung in einem Pas2js-PDF-Viewer
Installation der neuesten Version von FastReport für Lazarus unter LINUX
Delphi ATHENS (12) Einführung
Neue Version von Lazarus
Zeichnen von Turtle-Grafiken in Lazarus
Mit Delphi mithalten im FPC
Hinzufügen einer Textebene zu PDF-Dateien

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

INHALT

ARTIKEL

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

Chefredakteur

Detlef D. Overbeek, Niederlande Tel.: Mobil: +31 (0)6 21.23.62.68

Nachrichten und Pressemitteilungen nur per E-Mail an editor@blaisepascal.eu

Abonnemente können online unter **`https://www.blaisepascalmagazine.eu/`** oder per schriftlicher Bestellung abgeschlossen werden oder indem Sie eine E-Mail an **`office @ blaisepascal. eu`** senden. Das Abonnement kann zu einem beliebigen Zeitpunkt beginnen. Es werden alle 8 Ausgaben, nachfolgend dem Anfangsdatum bis zum vorab angekündigten Beendigungsdatum für Sie bereitgestellt.
Das Abonnement hat eine Laufzeit von **365 Tagen**. Abonnemente werden nicht ohne Vorankündigung verlängert.
Der Zahlungseingang wird per E-Mail verschickt. Sie können das Abonnement bezahlen, indem Sie die Zahlung an folgende Adresse senden:
ABN AMRO Bank Konto Nr. 44 19 60 863 oder per **Kreditkarte** oder **Paypal Name**: **Stiftung Pro Pascal** (Stichting Programeertaal Pascal)
`IBAN: NL82 ABNA 0441960863 BIC ABNANL2A` Umsatzsteuer-Nr.: **NL814254147B01** (Stichting Programmeertaal Pascal)
Abonnementsabteilung Edelstenenbaan 21 / 3402 XA IJsselstein, Niederlande **Mobil: + 31 (0) 6 21.23.62.68** office@blaisepascal.eu

Markenzeichen Alle verwendeten Markenzeichen sind Eigentum der jeweiligen Inhaber.
Vorbehalt: Obwohl wir uns bemühen sicherzustellen, dass die in der Zeitschrift veröffentlichten Informationen korrekt sind, können wir keine Verantwortung für Fehler oder Auslassungen übernehmen.
Wenn Sie etwas bemerken, das möglicherweise nicht korrekt ist, wenden Sie sich bitte an den Herausgeber, und wir werden gegebenenfalls eine Korrektur veröffentlichen.

Mitglied der **Königlich Niederländischen Bibliothek** KB

Mitglied und Spender von WIKIPEDIA

Abonnemente (2024 Preise)	**Internat. excl. VAT**	**Internat. incl. 9% VAT**
Elektronische Download Issue (8 per year) **±60 pages :**	**€ 64,20**	**€ 70**

Von der Redaktion

Grüße, geschätzte Leser,
Ich wünsche Ihnen allen ein frohes neues Jahr.
Trotz der gewaltigen Herausforderung bleibt die Erreichung des Weltfriedens ein übergeordnetes Ziel.

Man sollte alle Feindseligkeiten beenden. Außerdem schaffen diese antiquierten Individuen, die ständig nach Überlegenheit und Grandiosität streben, unweigerlich Probleme. Aber die Hoffnung ist immer vorhanden.
Optimismus für Verbesserungen und die Zukunft.

Der Frühling steht vor der Tür.

Ich hatte vor, diese Ausgabe noch vor den Weihnachtsferien in Angriff zu nehmen, aber das umfangreiche Arbeitspensum, das für die Fertigstellung dieser Aufgabe erforderlich war - insgesamt 154 Seiten -, machte es notwendig, sie über mehrere Tage auszudehnen. Erst jetzt habe ich es geschafft, es zu Ende zu bringen und zu übersetzen. Es gab zahlreiche Vorkommnisse, die es erforderlich machten, dass ich unermüdlich, fast rund um die Uhr, an der Fertigstellung arbeitete.

Der Grund für diese Entscheidung ist, dass wir ein zusätzliches Ziel zu erreichen hatten: Unser Magazin soll in einer anderen Sprache erscheinen, nämlich in brasilianischem Portugiesisch.
Nach Abschluss dieser Aufgabe werde ich mit der Entwicklung von zwei weiteren Versionen fortfahren:
Die deutsche Ausgabe und die brasilianische Ausgabe.
Dieser Erfolg wurde durch Geuze (GDK Software) ermöglicht,
die aufgrund ihrer Niederlassung in Brasilien, wo sie Muttersprachler beschäftigen, bei der Übersetzung ins Portugiesische geholfen haben.

Grüße an unsere Leser in Brasilien.
Diese Nachricht ist von größter Bedeutung.

Zwei Artikel in dieser Ausgabe markieren die Aufnahme der aktuellen Version von Delphi (12) Athen und natürlich, die neueste Version von Lazarus 3.0.
Die Veröffentlichung von PAS2JS Version 3 ist für diese Woche geplant.
Die Pascal-Familie wächst, und ich habe vor, weitere Sachen zu unternehmen.
Wir werden mit Hilfe der Universität zu Köln im Oktober 2024 eine bedeutende Pascal-Konferenz in Köln gestalten.
Die Konferenz wird an zwei Tagen stattfinden, und zwar am Donnerstag, den 10ten. und Freitag, den 11ten Oktober.
Die Veranstaltung wird sowohl aus Präsentationen als auch aus Workshops bestehen.

Im März werde ich eine weitere Pascal-Café-Veranstaltung in Holland ausrichten.
Ich bitte Sie um Ihre Anwesenheit aus allen Nationen,
so wie es im letzten Jahr der Fall war.
Die Erfahrung war großartig.

Im der nächsten Ausgabe, genauer gesagt in Ausgabe 116, werde ich Ihnen weitere Erläuterungen zu dieser Angelegenheit geben und unsere geplante Vorgehensweise darlegen.
Zum jetzigen Zeitpunkt wünsche ich Ihnen allen ein ruhiges und angenehmes Jahr 2024.

Ihr Redakteur:

Detlef

Von unserem technischen Berater, Jerry King

Meine Damen und Herren, da wir in Clouds voller Daten fliegen, werden wir einige Turbulenzen erleben. Bitte kehren Sie zu Ihren Sitzen zurück und legen Sie Ihre Sicherheitsgurte an.

TEST INSIGHT IN LAZARUS

VON MICHAEL VAN CANNEYT

ABSTRACT

Die FPCUnit Unterstützung in Lazarus hat ein Upgrade erhalten:
Das **FPCUnit Unit Testing Framework** kann jetzt direkt mit der Lazarus IDE kommunizieren, Dadurch wird es noch einfacher, Fehler in Ihrem Code zu beheben.

❶ EINFÜHRUNG

Free Pascal verfügt seit fast 20 Jahren über ein **Unit Testing Framework: FPCUnit** wird für das Testen vieler der in **Free Pascal** enthaltenen Pakete verwendet. Es ist in etwa kompatibel zu **DUnit**, dem **Delphi-Unit-Testframework: Vanilla-Testcode**, der für **DUnit** geschrieben wurde, lässt sich mit **Free Pascal** kompilieren.
Die **Lazarus IDE** verfügt über einige Assistenten, um einen **FPCUnit-Testfall** zu erstellen und ein **FPC-Unit-Testprogramm** zu starten. Es können 2 Arten von Testprogrammen erstellt werden:
Ein **Konsolen-Testprogramm** und ein **GUI-Testprogramm.**
Ersteres zeigt die Ergebnisse aller Tests auf der Konsole an.
Letzteres verwendet ein GUI-Fenster mit einer Baumansicht, um alle Tests und die Ergebnisse der Tests anzuzeigen.
Beide Programme haben den gleichen Nachteil: **Sie interagieren nicht mit der IDE.**
Es wäre viel schöner, wenn Sie **einfach auf den Namen des Tests klicken** könnten und zur Implementierung dieses Tests in der IDE gelangen würden. Oder, wenn es einen Stack-Trace auf die Fehlerstelle gibt und **Sie gelangen zu der Stelle, wo der Fehler** auftritt.
Nun, jetzt können Sie das: In der Stammversion von Lazarus wurde das **LazFPCUnit Package** um **'Test Insight'** erweitert.
Die Idee für diese Funktion wurde von dem **ähnlich benannten Delphi-Plugin von Stefan Glienke** übernommen.

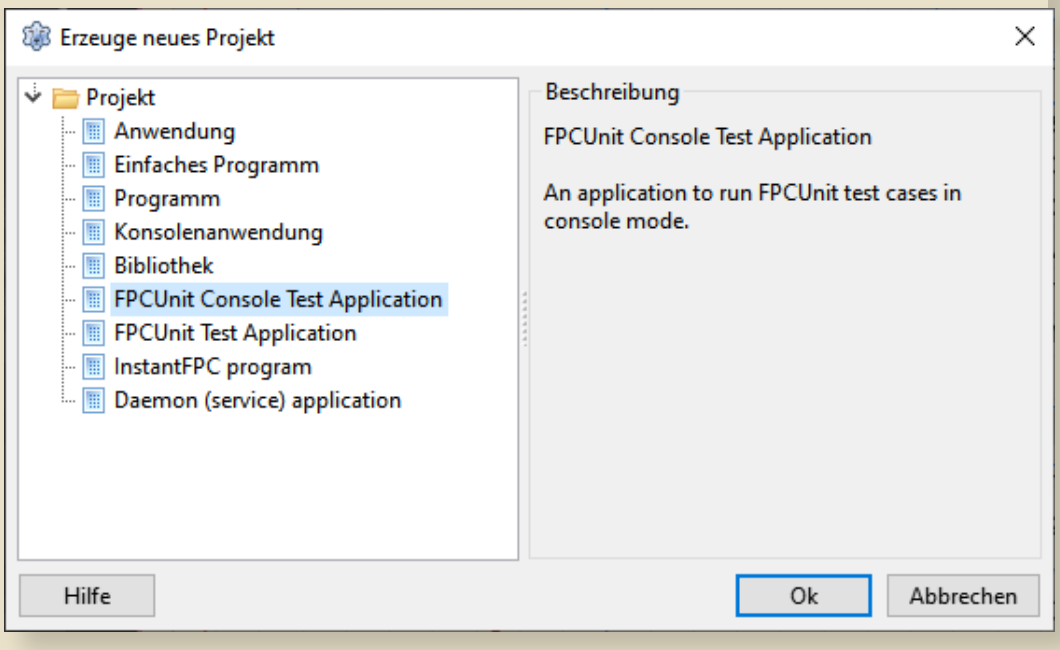


ABBILDUNG 1: ERSTELLEN EINES FPCUNIT KONSOLENPROGRAMMS

❷ ARCHITEKTUR

Das **FPCUnit Unit Test System** funktioniert wie alle anderen **Testsysteme**.
Es gibt eine **Testregistrierung** und es gibt **Test-Listener.** Wenn die Tests aus der Registrierung ausgeführt werden, werden der Fortschritt und die Ergebnisse von den registrierten **Listenern** gemeldet. Die Konsolen-**Listener** schreiben einfach in einem von mehreren Formaten auf die Konsole (*Sie können Ihre eigenen Formate registrieren*), während der **UI-Listener Knote**n zu einer Baumansicht hinzufügt.
Der **testinsight-Listener** sendet die Ergebnisse mit HTTP-Anfragen an einen Server. Der Standort dieses Servers kann in den Quellen oder in einer kleinen Konfigurationsdatei angegeben werden.

Die **TestInsight Unterstützung in der Lazarus IDE** startet einen kleinen HTTP-Server, der auf die HTTP-Anfragen mit den Testergebnissen wartet. Der HTTP-Server wird gestartet, wenn Sie das Fenster '**TestInsight**' über das Menü '**Ansicht - TestInsight**' in der **Lazarus IDE** öffnen.
Wenn das Testprogramm existiert, wird es mit einer speziellen Option gestartet, um eine Liste von Tests zu erhalten.
Wenn Sie einen Testlauf starten, werden die Tests ausgeführt und das Fenster zeigt die Testergebnisse an. Wenn Sie auf einen Test doppelklicken, werden die Code-Tools der IDE verwendet, um zu der richtigen Methode im Testprojekt zu springen. Wenn das Testprogramm den HTTP-Server von testinsight nicht erreichen kann, wird es wieder als normales **FPCUnit-Testkonsolenprogramm** ausgeführt. Sie könnten dasselbe mit einem **UI-Programm** tun, wenn Sie dies wünschen, aber diese Option wurde für das **FPCUnit UI-Testprogramm** nicht aktiviert.

❸ ANWENDUNG

Damit dies funktioniert, benötigen Sie die neuesten IDE-Quellen und müssen natürlich das **lazfpcunit-Paket** installieren. Es ist Teil der **Standardliste** der Pakete in der IDE.
Der Menüpunkt '**Neues Projekt' → 'FPCUnit Konsolentestanwendung**' (*Abbildung 1 auf Seite 5*) verfügt nun über einen Projektassistenten mit folgenden Optionen bietet:

- **Alle Tests standardmäßig ausführen**
  Das **Standard-FPCUnit-Konsolenprogramm** zeigt eine **Hilfe** an, wenn es ohne Befehlszeilenoptionen ausgeführt wird. Wenn diese Option aktiviert ist, führt das Programm die Tests auch aus, wenn keine Befehlszeilenoptionen angegeben sind.
- **Standard-Ausgabeformat**
  Mit diesem Kontrollkästchen können Sie das Standardausgabeformat für die Konsolenausgabe festlegen. Sie können zwischen XML, einfachem Text (*mit oder ohne Zeitangaben*) oder **LaTeX** wählen.
- **Testinsight verwenden,** um Ergebnisse an die IDE zu übermitteln.
  Wenn diese Option aktiviert ist, werden die Testergebnisse mit testinsight an die IDE gesendet.

ERSTEN TESTFALL ERSTELLEN
Wenn diese Option aktiviert ist, startet die IDE sofort den Assistenten '**FPCUnit Testfall erstellen**' wenn das Projekt erstellt wird. Der Optionsdialog ist in *Abbildung 2 auf Seite 7* dargestellt.
Sobald Ihr Projekt mit der Option '**Testinsight verwenden**' erstellt wurde, sollten Sie es sofort speichern und kompilieren. Dadurch kann die testinsight-Unterstützung der **IDE** eine Liste der Tests abrufen.
Sobald das Projekt kompiliert ist, können Sie das Testinsight-Fenster öffnen. Das Fenster sieht genauso aus wie das Fenster des grafischen **FPCUnit-Testprogramms** - nicht überraschend, da es von diesem Fenster kopiert wurde. Es wurden jedoch einige Elemente hinzugefügt, und es verhält sich anders als das Originalfenster.

Wenn das Fenster geöffnet wird, versucht es, das aktuelle Projekt mit den entsprechenden Optionen auszuführen, um eine Liste der Tests zu erhalten.

Das Ergebnis sieht wie in *Abbildung 3 auf Artikelseite 3* aus, wenn alles gut gegangen ist.
Mit dem Button '**Aktualisieren**' (links) können Sie jederzeit die Liste der Tests aktualisieren.
Die Liste der Tests wird in jedem Fall aktualisiert, wenn das Testprogramm tatsächlich ausgeführt wird.
Wenn das aktive Projekt gewechselt wird, wird auch die Liste der Tests automatisch aktualisiert:
Der Pfad der aktuellen ausführbaren **FPC-Unit-Testdatei** wird immer in der Statusleiste am unteren Rand des Fensters angezeigt.

3 VERWENDUNG (FORTSETZUNG)

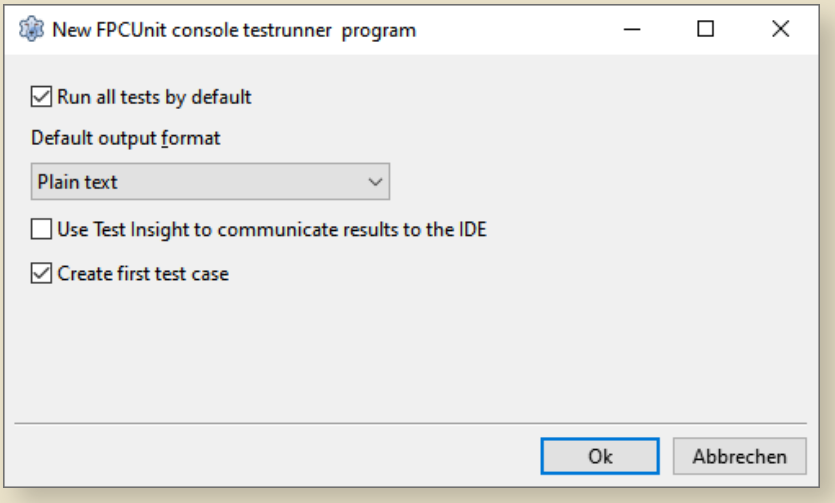


Abbildung. 2: Die Optionen des FPCUnit-Konsolenprogramms

Wenn Sie auf den Button 'Alle Tests ausführen' (den grünen Doppelpfeil) klicken, wird das Testprogramm ausgeführt. Wenn die Testergebnisse da sind, ändert die farbige Statusanzeige vor jedem Test ihre Farbe entsprechend dem Ergebnis des Tests.
Liegt ein Fehler vor, werden weitere Informationen in Knoten unterhalb des Testknotens angezeigt, wie in Abbildung 4 auf Seite 8 dargestellt. Wenn Sie auf den Test oder einen Fehlerknoten doppelklicken, wird versucht, den Code des Tests zu finden und die Quelldatei zu öffnen. In den meisten Fällen funktioniert dies ohne Probleme.
In solchen Fällen kann der Mechanismus jedoch fehlschlagen:

❶ Das derzeit aktive Projekt in der IDE ist kein Unit-Test-Projekt.
❷ Die Quellen des Tests sind nicht Teil des Projekts.
❸ Wenn Sie einige fortgeschrittene Funktionen des Test-Frameworks verwenden (*dynamisches Erstellen von Tests*).
TestInsight nimmt an, dass die Testnamen Methodennamen einer Klasse sind.

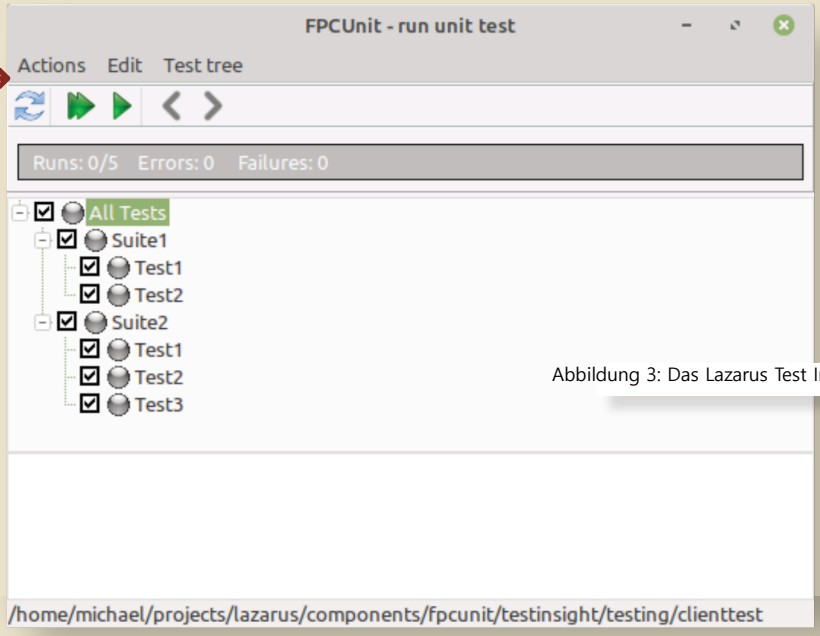


Abbildung 3: Das Lazarus Test Insight Fenster

EIN EXISTIERENDES FCPUNIT TEXT PROGRAMM KONVERTIEREN

Wenn man den **'New program**' Wizard von **Testinsight** benutzt, wird ein existierendes Programm nicht für die Nutzung von Testinsight konvertiert. In dem Falle muss man einige Änderungen im Main project file vornehmen.

Das typische FPCUnit test Konsolenprogramm hat den folgenden Main project file Quellcode:

```
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.
```

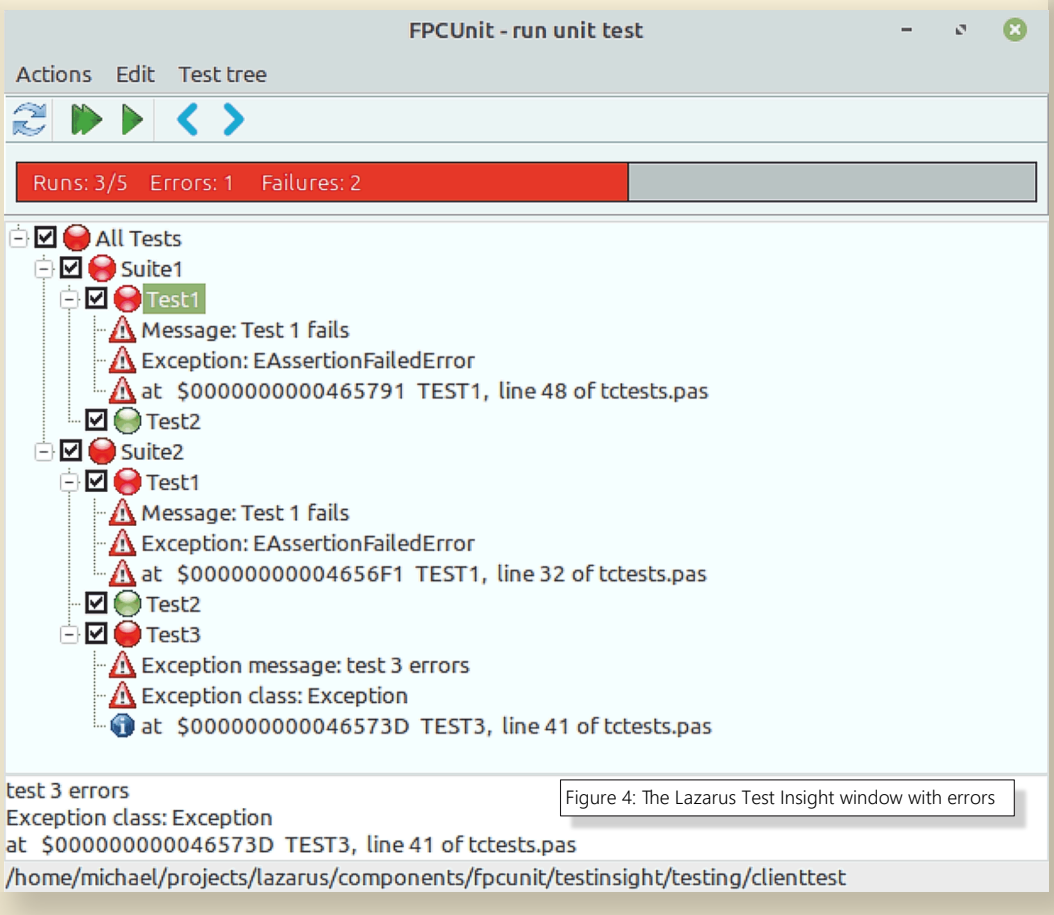


Figure 4: The Lazarus Test Insight window with errors

Bei diesem Projektcode müssen zwei Dinge getan werden:

❶ Die Unit `fpcunittestinsight` muss zur uses-Klausel hinzugefügt werden.
❷ Die Funktion `IsTestinsightListening` muss aufgerufen werden.
Wenn sie `True` zurückgibt, muss die Routine `RunRegisteredTests` aufgerufen werden.

Beide sind in der Unit `fpcunittestinsight` implementiert.
Wenn IsTestinsightListening False zurückgibt, muss der ursprüngliche Code ausgeführt werden.

Die beiden zu verwendenden Funktionen werden wie folgt deklariert:

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

Das Argument config ist der Name einer `.INI-Datei` mit den Einstellungen für den Testlauf.
Standardmäßig ist dies die Datei `TestInsightSettings.ini`, die von der **IDE** verwendet wird.
Die ausgewählten Tests werden in diese Datei geschrieben sowie der **Port**, an dem die **IDE** lauscht (*die baseURL*).
Das Argument `BaseURL` ist die **URL**, auf der der testinsight-Server lauscht.
Standardmäßig ist dies **`http://localhost:8081/tests`**,
aber dies wird von der IDE in der Konfigurationsdatei festgelegt.
Sie können diese Voreinstellungen ändern, um beispielsweise das Testprogramm aus der Ferne auszuführen,
aber dennoch die Ergebnisse lokal auf Ihrem Entwicklungs-PC erhalten.
Mit diesen Änderungen wird der neue Projektcode wie folgt aussehen:

```
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.
```

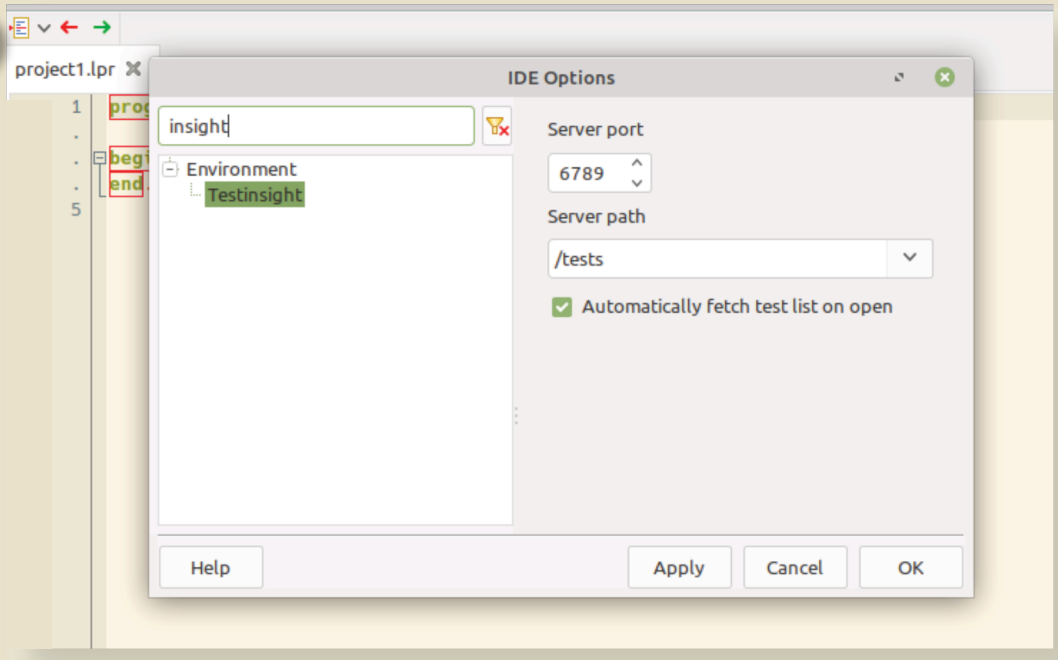


Abbildung 5: Das Lazarus Test Insight Konfigurationsfenster

KONFIGURATION

In der IDE können Sie den Port und den Basisstandort, an dem der Server lauschen soll, festlegen. Im Menü **'Extras - Optionen'** unter TestInsight können die folgenden Einstellungen konfiguriert werden:

Basispfad: Dies ist der Pfad auf dem HTTP-Server, an den die Anfragen gesendet werden müssen. Die Standardeinstellung ist `'/tests'`.

Server-Port: Dies ist der Port auf dem HTTP-Server, an den die Anfragen gesendet werden müssen. Die Standardeinstellung ist 6789

Tests automatisch abrufen: Wenn das Fenster 'Test Insight' geöffnet wird oder sich das aktuelle Projekt ändert, wird die Liste der Tests automatisch abgerufen.

Wenn die Funktion deaktiviert ist, können Sie den Button Aktualisieren verwenden, um die Liste der Tests abzurufen.

6 SCHLUSSFOLGERUNG

Die 'Test Insight'-Funktionalität erleichtert die test-getriebene Entwicklung, da Sie aus der in die IDE eingebetteten Anzeige der Testergebnisse sofort zur Implementierung eines Tests oder einer Fehlerstelle springen können.

Die aktuelle Version von 'Test insight' ist nur eine erste Version:

Einige Erweiterungen sind geplant, wie z.B. die Filterung von Tests und die automatische Auswahl von Tests.

Die PAS2JS-Unterstützung von testinsight ist ebenfalls in Arbeit.

EXECUTING PROGRAMS ON THE SERVER IN PAS2JS

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 simple tasks such as executing an SQL statement on a database and returning the result, or more complicated and time-consuming tasks such as making a backup of the database, indexing PDF files, compiling a software project and running a test suite, or even installing software on the server. Usually, these remote programs should also be executed by the user.

To keep programs scalable, these tasks should be short-lived. A return time of 1 second for a HTTP request is already too long, so executing a time consuming task and waiting for the return of a single HTTP request is not a good idea:
the HTTP server is occupied with the request, the browser 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.

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.

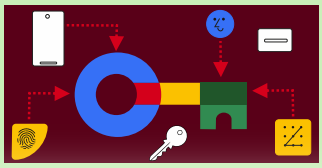
These components are ignorant of the communication mechanism between browser and server, this means that they do not implement the actual RPC calls used to start the process: There are many possible mechanisms, and some may be more suitable for your purpose than others.

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.

OHNE PASSWORT LOGIN MIT PASSKEYS



ABSTRACT
In diesem Artikel möchte ich die Aspekte der Erstellung und Verwendung von Passkeys erläutern. Es sieht ziemlich kompliziert aus, aber wenn Sie die Hauptziele verfolgen, ist es ein sehr interessanter und nützlicher Weg, um einen Dienst zu erhalten, der sowohl Sicherheit als auch Komfort bietet.

WAS SIND PASSKEYS?

Ein Passkey ist eine Art digitaler Berechtigungsnachweis, der mit einer Anwendung oder Website und einem Benutzerkonto verbunden ist.
Ein Berechtigungsnachweis ist ein Dokument, das eine Qualifikation, Kompetenz oder Befugnis beschreibt, die einer Person von einem Dritten mit entsprechender Befugnis ausgestellt wurde.
Mit Passkeys können sich Benutzer anmelden, ohne dass weitere Authentifizierungsfaktoren oder die Eingabe eines Benutzernamens und eines Passworts erforderlich sind.
Passwörter und andere antiquierte Authentifizierungsmethoden sollen durch diese Technologie ersetzt werden.

❶ VORTEILE

Mit Passkeys können Sie sich schnell und sicher bei Webdiensten anmelden:

- ohne ein Passwort,
- ohne spezielle Hardware
- ohne das Risiko von Phishing.

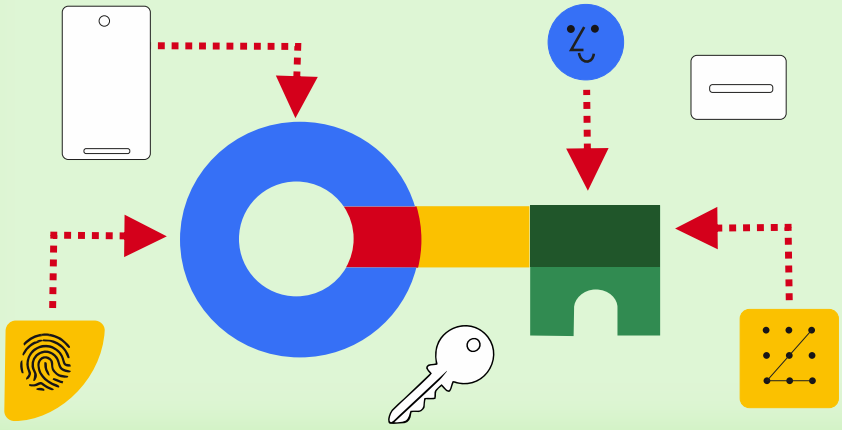
Ich werde versuchen, Ihnen zu erklären, wie dieser Passwortnachfolger funktioniert und wo Sie Passkeys mit Ihrem PC (Desktop, um genau zu sein), Smartphone und Tablet verwenden können.

NACHTEILE, DIE ZU VORTEILEN FÜHREN

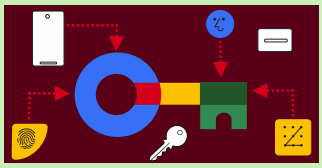
Sowohl Entwickler als auch Benutzer mögen Passwörter nicht:

- sie bieten ein schlechtes Benutzererlebnis,
- sie führen zu höherem Aufwand bei der Konvertierung,
- sie schaffen Sicherheitshaftung für Benutzer und Entwickler.

Der Password Manager (von Google) in Android und Chrome reduziert den Aufwand und die Fehleranfälligkeit durch automatisches Ausfüllen.
Er ist gut für Entwickler, die nach weiteren Verbesserungen bei der Konvertierung und Sicherheit suchen, Passkeys und Förderierte Identität (zusammengefasste Identität) sind die besten modernen Ansätze für Entwickler.
Ein Passkey kann die Anforderungen der Multi-Faktor-Authentifizierung in einem einzigen Schritt erfüllen und ersetzt sowohl ein Passwort als auch ein OTP - One Time Password - (z.B. 6-stelliger SMS-Code), um einen robusten Schutz gegen Phishing-Angriffe zu bieten, und vermeidet den UX* - Probleme von SMS (Short Message/Messaging Service) oder App-basierten One Time Passwords.
*Die User eXperience (UX) ist das Gefühl, das ein Benutzer bei der Interaktion mit einem Produkt, System oder Service hat.



OHNE PASSWORT LOGIN MIT PASSKEYS



VORTEILE
Fortsetzung

Da Passkeys standardisiert sind, ermöglicht eine einzige Implementierung ein passwortloses Erlebnis auf allen Geräten eines Benutzers, auf verschiedenen Browsern und Betriebssystemen. Genau in der Mitte, wo sich diese Komponenten treffen, finden Sie den Schlüssel. Diese Überschneidung ist die Art und Weise, wie Ihre Website aufgebaut, strukturiert und gestaltet sein sollte, um ein perfektes Benutzererlebnis zu schaffen.

Passkeys sind sicherer:

- Entwickler speichern nur einen öffentlichen Schlüssel auf dem Server statt eines Passworts, was bedeutet, dass es für einen böswilligen Akteur viel weniger attraktiv ist, sich in Server einzuhacken und dass im Falle eines Einbruchs viel weniger Aufräumarbeiten erforderlich sind.
- Passkeys schützen Benutzer vor Phishing-Angriffen.
  Passkeys funktionieren nur auf den bei ihnen registrierten Websites und Anwendungen;
- Ein Benutzer kann nicht dazu verleitet werden, sich auf einer betrügerischen Website zu authentifizieren, da der Browser oder das Betriebssystem die Verifizierung übernimmt.
- Passkeys reduzieren die Kosten für den Versand von SMS und sind damit ein sicheres und kostengünstiges Mittel zur Zwei-Faktoren-Authentifizierung.
- Viele komplizierte Prozesse werden im Hintergrund ausgeführt, um die Sicherheit zu gewährleisten. Aber diese Aktivitäten bleiben für den Einzelnen unbemerkt.

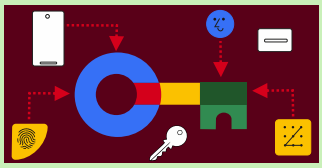
Für Sie ist der Zugriff auf Ihr Konto mit einem Passkey so einfach wie das Antippen des Fingerabdrucksensors Ihres Smartphones.
Denken Sie daran, dass ein kreativer Krimineller leicht einen Stempel herstellen kann - das ist nur ein Beispiel. Der erhöhte Komfort könnte Personen, die normalerweise nicht auf Sicherheit bedacht sind, davon überzeugen, die neue Anmeldetechnik zu verwenden.

❷ KRIMINELLE
Jeden Tag übernehmen Cyberkriminelle Millionen von Konten, meist per Passwort.
Analysen von geleakten Passwörtern zeigen, dass viele Benutzer dazu neigen, einfache Passwörter zu wählen und diese obendrein für mehrere Webdienste verwenden.
Die schützende, aber umständliche Zwei-Faktor-Authentifizierung wird aus Bequemlichkeit oft ignoriert. Das kann man niemandem verübeln.
Mit der Zeit vermehren sich die Konten für Websites und Apps, weil man sich heutzutage fast überall registrieren muss. Wenn Sie sich konsequent an Best Practices halten, um sich zu schützen, haben Sie eine Menge Arbeit vor sich. Für technisch versierte Menschen ist das kein Problem, aber versuchen Sie einmal, es älteren Menschen zu erklären.

Die effektive Verwaltung von Passwörtern kann eine Herausforderung sein, denn das Konzept ist älter als das Internet, liegt nahe bei den Dinosauriern und sogar weit vor der Erfindung des ersten Computers. Sein ursprünglicher Zweck bestand nicht darin, Hacker abzuschrecken oder Phishing-Versuche zu verhindern.
Daher ist der Einsatz von Maßnahmen wie der Zwei-Faktor-Authentifizierung unerlässlich, um das veraltete Verfahren auch in der heutigen Zeit weiter zu verwenden.

PHISHING
Phishing bezeichnet die betrügerische Praxis, zu versuchen, an sensible Informationen wie Passwörter oder Kreditkartendaten zu gelangen, indem man sich als vertrauenswürdige Instanz ausgibt. Die Electronic Frontier Foundation (EFF) hat einige Kommentare dazu abgegeben
https://www.eff.org/deeplinks/2023/10/what-passkey

OHNE PASSWORT LOGIN MIT PASSKEYS

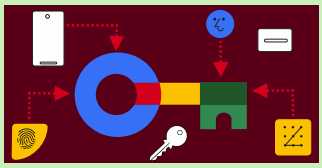


PHISHING - FORTSETZUNG
Wenn Ihnen eine Person eine URL zu einer Anmeldeseite auf einem Domänennamen gibt, der dem Original sehr ähnlich ist, könnten Sie getäuscht werden.
Passkeys enthalten Informationen über den spezifischen Domainnamen, für den sie erstellt wurden.
Ihr Webbrowser wird dann NICHT die Domain aufrufen, da er mühelos eine genaue Übereinstimmung überprüfen kann.
Um Ihre Sicherheit zu gewährleisten, erlaubt Ihr Browser NICHT die Übermittlung eines Passworts an den betrügerischen Domainnamen.
Solange Sie jedoch neben Ihrem Passkey ein gespeichertes Passwort haben, könnte eine betrügerische Website Sie täuschen, indem sie behauptet, Ihr Passkey funktioniere nicht und Sie auffordert, stattdessen das Passwort einzugeben.
In diesem Fall führt die Eingabe des Passworts zur erfolgreichen Ausführung des Phishing-Angriffs.
Phishing ist nach wie vor eine ernstzunehmende Bedrohung, aber Personen, die gewöhnlich einen Passkey für den Zugriff auf eine bestimmte Website verwenden, werden eher misstrauisch, wenn sie aufgefordert werden, stattdessen ein Passwort einzugeben. Dies bietet ein gewisses Maß an Sicherheit, wenn auch nicht absolut.
BEISPIEL PAYPAL
Die Authentifizierung umfasst von Natur aus die Domäne der Website.
Dies erhöht die Widerstandsfähigkeit von Passkeys gegen Phishing-Versuche.
Wenn Sie über eine Phishing-E-Mail auf PayPal.com zugreifen, können Sie den zuvor für PayPal.com erstellten Passkey nicht mehr verwenden.
Alternativ können Sie nur einen neuen Passkey für die alternative Domäne erstellen.
Dies ist für Phisher völlig aussichtslos.
Zweifelsohne sollten Sie der Phishing-Website dennoch nicht Ihre sensiblen Daten anvertrauen.
Sehen Sie sich das Angebot von PayPal auf der nächsten Seite an.

EIN STAR IST GEBOREN
Mit der Einführung von Passkeys bricht ein neues Zeitalter an.
Ohne die Notwendigkeit von Passwörtern und ohne die potenziellen Gefahren, die mit Phishing-Angriffen verbunden sind.
Die nötige Technologie ist bereits in Ihrem Smartphone, Tablet und Computer vorhanden.
Zahlreiche Webdienste sind bereits für diese neue Entwicklung gerüstet.

Das Logo für Passkeys

OHNE PASSWORT LOGIN MIT 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:

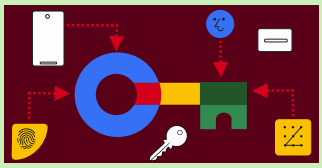
- It's a physical device so it can't be rerouted to another device
- If you lose it, the key has no identifiable info about who it belongs to
- Enter it into your USB slot or hold it near your device and that's it

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.

Improve Your 2-Step Verification

OHNE PASSWORT LOGIN MIT PASSKEYS



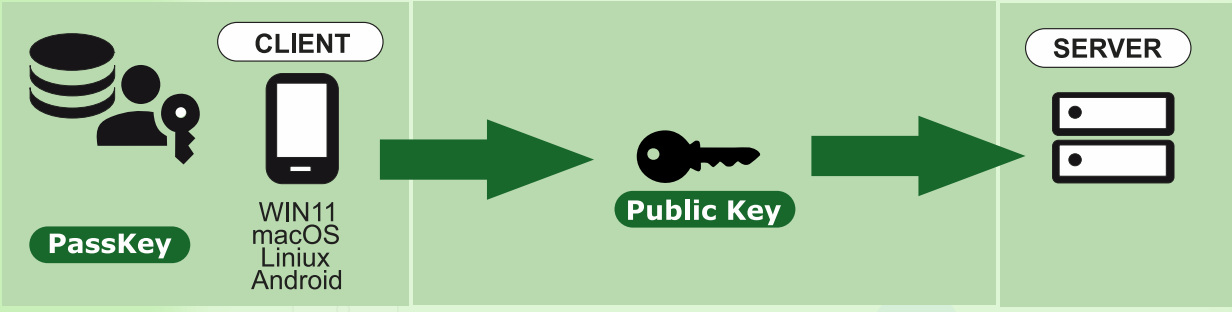
❷ WIE FUNKTIONIERT DAS?
PASSKEY-AUTHENTIFIZIERUNG
Im Gegensatz zu Passwörtern gibt es bei der Passkey-Authentifizierung kein gemeinsames Geheimnis, das sowohl Ihnen als auch dem Dienst bekannt ist und von einem Angreifer abgefangen werden kann. Stattdessen haben Sie einen privaten Schlüssel und einen öffentlichen Schlüssel für jedes Ihrer Konten. Nur Sie besitzen den wichtigen privaten Schlüssel.

*1. Wenn Sie einen Passkey für einen Webdienst erstellen, erhält der Dienst nur den öffentlichen Schlüssel und verknüpft ihn mit Ihrem Konto. Mit dem öffentlichen Schlüssel kann der Dienst nun überprüfen, ob Sie den privaten Schlüssel besitzen, ohne dass der Dienst den privaten Schlüssel jemals zu Gesicht bekommt.

"Sowohl Ihnen als auch dem Dienst ist kein gemeinsames Geheimnis bekannt, das ein Angreifer abfangen könnte".

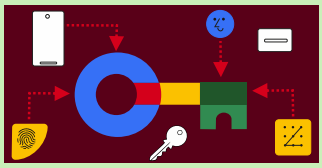
Konkret funktioniert dies wie folgt:
Wenn Sie sich mit einem Passkey anmelden,
liefert der Webdienst Zufallsdaten, die so genannte Challenge, die Ihr Gerät mit Ihrem privaten Schlüssel signiert.

Der Webdienst überprüft die digitale Signatur und kann mit Sicherheit feststellen, dass sie von Ihrem privaten Schlüssel stammt. Diese Methode wird PUBLIC KEY CRYPTOGRAPHY genannt. Sie ist seit Jahrzehnten erprobt und wird z.B. auch für HTTPS- und E-Mail-Verschlüsselung verwendet.

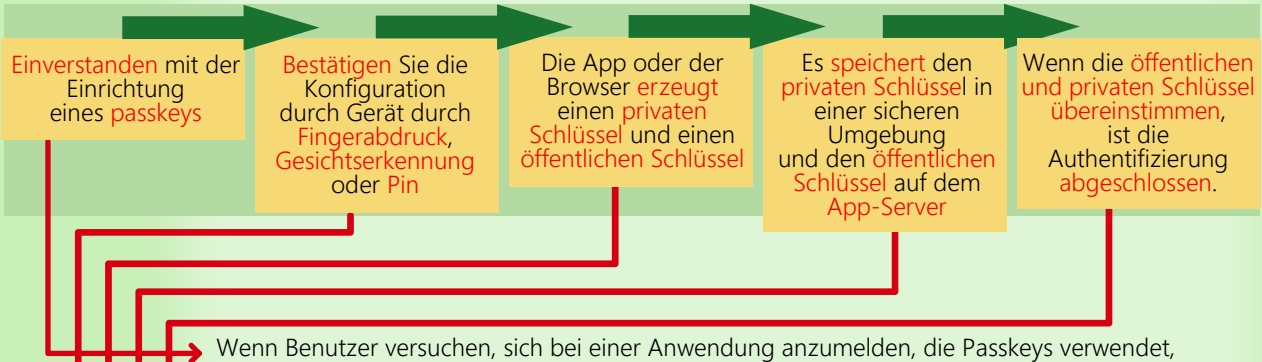


Die EFF Electronic Frontier Foundation hat dazu einige Anmerkungen:
Ein Passkey besteht aus ca. 100-1400 Bytes zufälliger Daten, die auf Ihrem Gerät (z.B. Ihrem Telefon, Laptop oder Sicherheitsschlüssel) für die Anmeldung bei einer bestimmten Website generiert werden.
Sobald der Passkey generiert ist, registriert Ihr Browser ihn bei der Website und er wird an einem sicheren Ort gespeichert (z.B. in Ihrem Passwortmanager).
Ich persönlich würde dies lieber NICHT zulassen und einen anderen Weg wählen, um meinen Schlüssel zu speichern.
Denken Sie daran, dass dies der Teil ist, der privat ist. Siehe Erklärung 1*
(Der Passwort-Manager kann ihn nicht auf andere Geräte kopieren).
Von da an können Sie diesen Schlüssel verwenden, um sich bei dieser Website anzumelden, ohne ein Passwort einzugeben. Wenn Sie die Anmeldeseite einer Website aufrufen, haben Sie die Möglichkeit, sich "mit einem Passkey anzumelden".
Wenn Sie diese Option wählen, erhalten Sie eine Bestätigungsaufforderung von Ihrem Passwortmanager und werden nach der Bestätigung eingeloggt.
Damit dies alles funktioniert, müssen die Website, Ihr Browser, Ihr Passwort-Manager und in der Regel auch Ihr Betriebssystem Passkeys unterstützen.

OHNE PASSWORT LOGIN MIT PASSKEYS



PASSKEY INITIATION AND FLOW



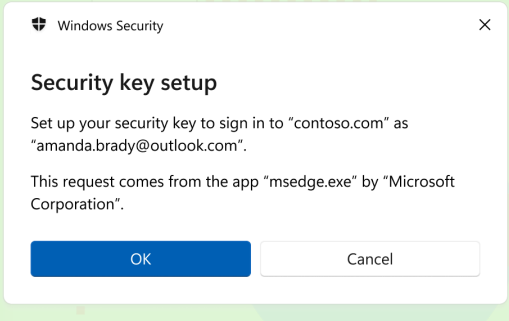
Wenn Benutzer versuchen, sich bei einer Anwendung anzumelden, die Passkeys verwendet, erscheint in der Regel ein Dialogfeld, in dem sie aufgefordert werden, einen Passkey (in der Regel ein Fingerabdruck, eine Gesichtserkennung oder ein Muster) für zukünftige Authentifizierungszwecke zu speichern.

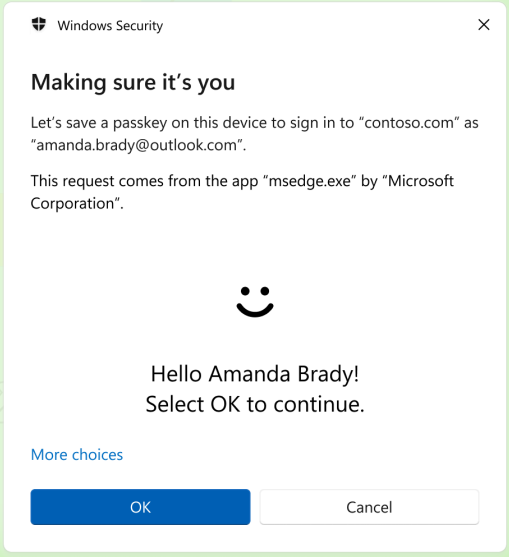
Der Benutzer muss dies bestätigen und fortfahren.
Wenn der Benutzer zustimmt, die Passkey-basierte Authentifizierung zu verwenden, erzeugt die Anwendung ein kryptografisches Schlüsselpaar, das aus einem öffentlichen und einem privaten Schlüssel besteht. Diese Methode wird lokal auf dem Gerät des Benutzers durchgeführt, z. B. auf dem Computer, dem Tablet, oder Smartphone.

Während des Vorgangs wird der private Schlüssel sicher in einer Brieftasche gespeichert, während der öffentliche Schlüssel zur Speicherung an den Anwendungsserver gesendet wird.

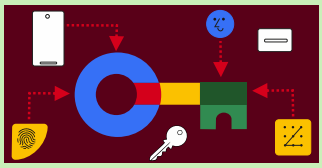
Beim nächsten Anmeldeversuch des Benutzers sendet der Anwendungsserver verschlüsselte Daten, eine Aufgabe (Challenge) an das Gerät des Benutzers und verwendet dabei den öffentlichen Schlüssel des Geräts. Um die Identität des Benutzers zu überprüfen, der sich anzumelden versucht, löst das Gerät die Aufgabe und sendet eine mit dem privaten Schlüssel signierte Antwort zurück an den Server.

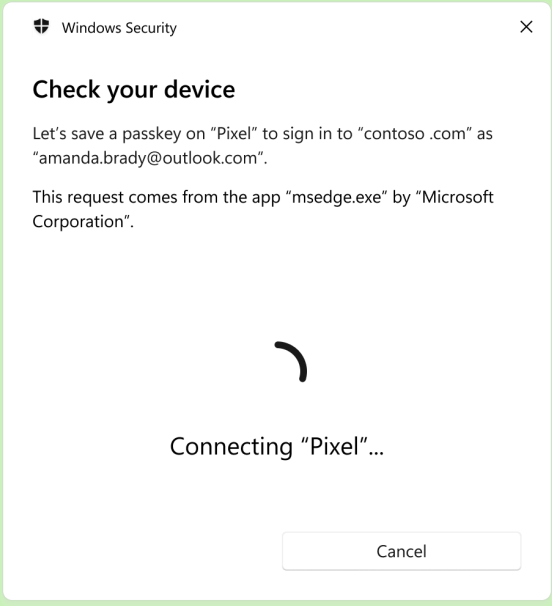
Nachdem die Überprüfung des öffentlichen und privaten Schlüssels abgeschlossen ist, kann der Benutzer erfolgreich auf die Anwendung zugreifen.

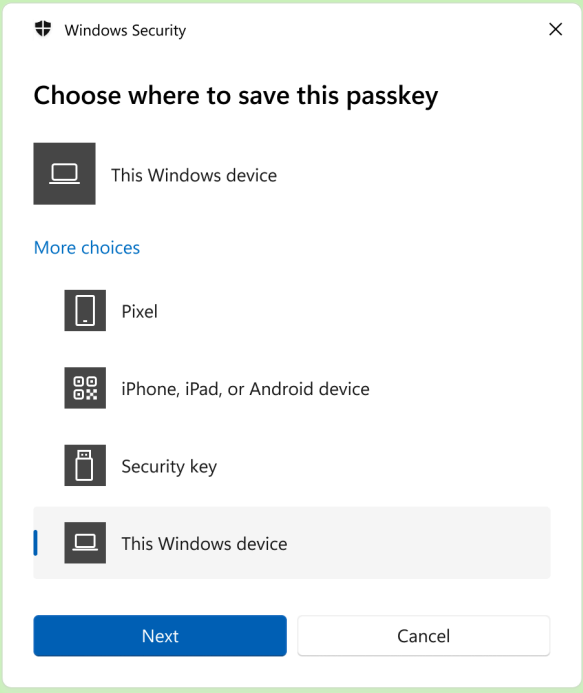


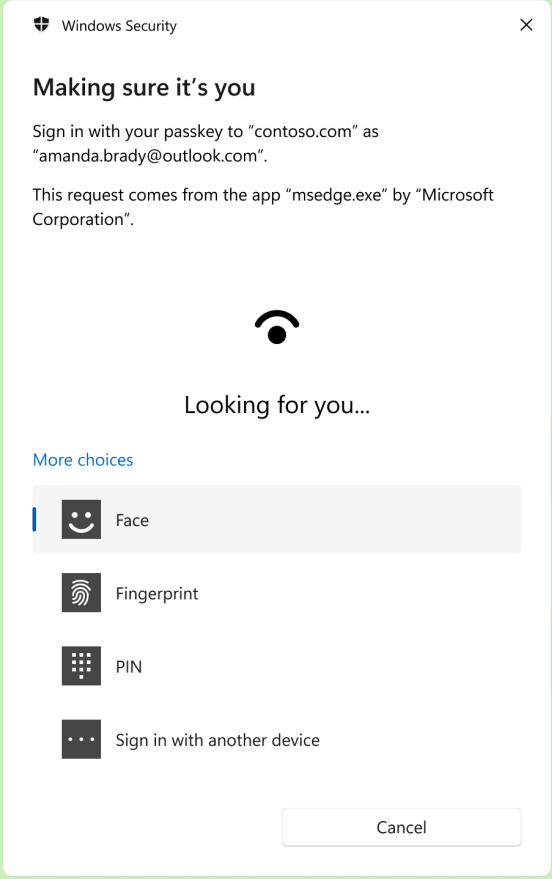


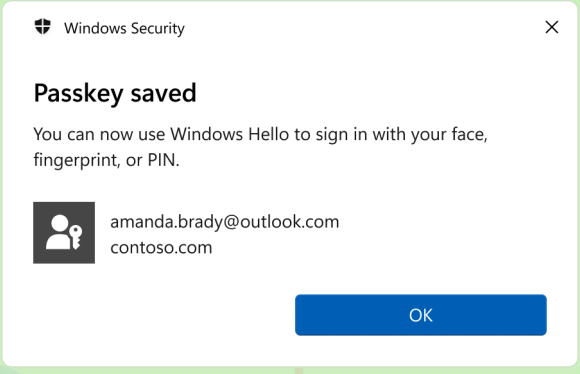
OHNE PASSWORT LOGIN MIT PASSKEYS

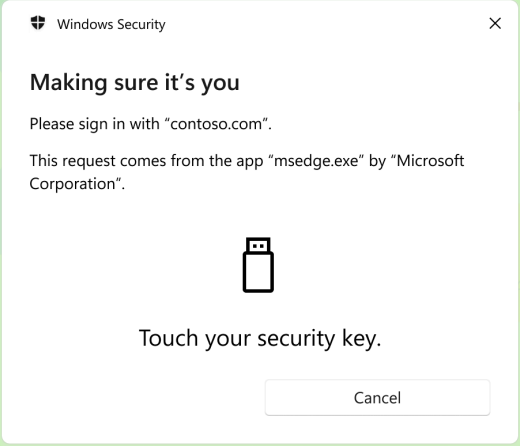




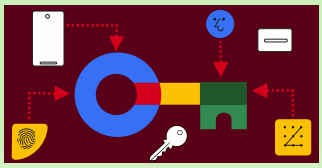




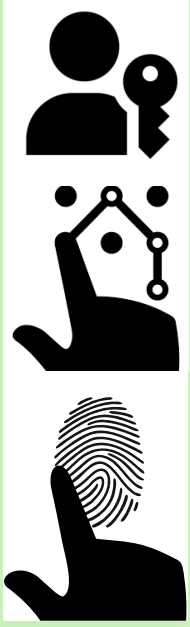




OHNE PASSWORT LOGIN MIT PASSKEYS



• Sie können mehrere Passkeys erstellen:
Jeder Passkey schaltet ein einzelnes Konto auf einer einzigen Website frei. Für mehrere Konten auf einer einzigen Website können Sie mehrere Passkey für diese Website haben.
Wenn Sie z.B. ein Konto für soziale Medien für den privaten Gebrauch und eines für den geschäftlichen Gebrauch haben, würden Sie für jedes Konto einen anderen Passkey haben.
• In der Regel haben Sie sowohl ein Passwort als auch einen Passkey für Ihr Konto und können sich mit einem von beiden anmelden. Die Anmeldung mit einem Passkey ist in der Regel schneller, da Ihr Passwort-Manager Ihnen anbietet, dies mit einem einzigen Klick zu tun, anstatt der vielen Klicks, die die Anmeldung mit einem Passwort normalerweise erfordert. Außerdem können Sie bei der Anmeldung mit einem Passkey in der Regel auf die herkömmliche Zwei-Faktor-Authentifizierung verzichten (SMS, Authentifizierungs-App oder Sicherheitsschlüssel).

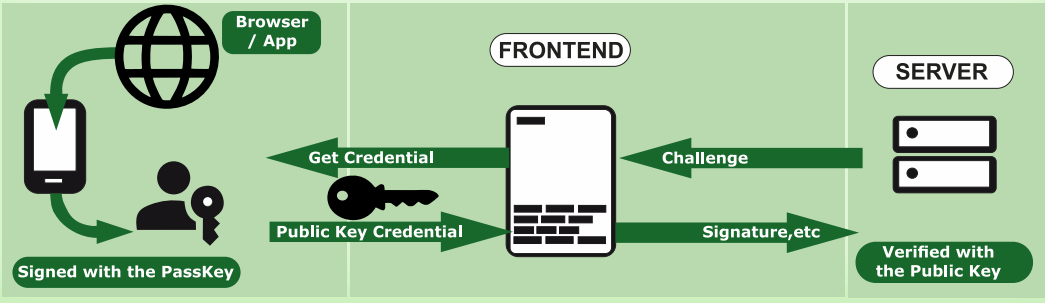


Mit Passkeys wird ein zweiter Faktor eingebaut.
Jedes Mal, wenn Sie sich mit dem Passkey anmelden, kann Ihr Browser oder Betriebssystem Sie auffordern Ihre PIN zum Entsperren des Geräts erneut einzugeben.
Wenn Sie zum Entsperren Ihres Geräts einen Fingerabdruck oder eine Gesichtserkennung verwenden, kann Ihr Browser Sie stattdessen auffordern, Ihren Fingerabdruck erneut einzugeben oder Ihr Gesicht zu zeigen, um zu bestätigen, dass Sie es wirklich sind, der sich anmelden möchte.
Ich persönlich mag die Gesichtserkennung nicht. Es ist heutzutage sehr einfach, ein Bild als Gesichtsmaske zu erstellen.
Damit haben Sie zwei Faktoren zur Authentifizierung:
Das Gerät, das Ihren Passkey speichert, ist etwas, das Sie haben, und es wird von etwas begleitet, das Sie kennen (die PIN) oder etwas, das Sie haben (ein Fingerabdruck oder ein Gesicht)..

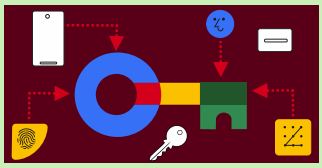
Anstelle eines Passworts erhalten Sie einen eindeutigen Schlüssel für jedes einzelne Konto.

- Der Schlüssel wird auf Ihrem mobilen Gerät, PC oder Tablet gespeichert.
- Um sich anzumelden, wählen Sie auf der entsprechenden Seite einfach die Option zur Anmeldung mit dem Passkey.
- Überprüfen Sie die Verwendung des Passkeys mit einer prägnanten PIN. Dies ist für alle Passkeys gleich.

Der Vorgang kann durch die Verwendung Ihres Fingerabdrucks oder Gesichtsscans beschleunigt werden:
Danach werden Sie umgehend eingeloggt.
Seien Sie vorsichtig mit diesen biologischen Merkmalen, da sie in der heutigen Zeit überzeugend gefälscht werden können.
Die von Ihnen angegebene PIN oder biometrischen Daten werden ausschließlich vor Ort verwendet, um auf den Passkey zuzugreifen, und werden niemals an Dritte weitergegeben.
Der Passkey wird automatisch durch einen zweiten Faktor gesichert, wobei der erste Faktor der Passkey selbst ist.
Und das alles ohne die Notwendigkeit einer separaten und lästigen Zwei-Faktor-Authentifizierung.
Ein Passkey ist im Wesentlichen ein kryptografisches Schlüsselpaar, das automatisch von einem Sicherheitschip auf Ihrem Computer oder Smartphone erzeugt wird.
Die Anforderung, für jede Registrierung einen neuen Passkey zu generieren, um die Anforderungen der Website zu erfüllen, wird hoffentlich in naher Zukunft überflüssig werden.



OHNE PASSWORT LOGIN MIT PASSKEYS



SPEICHERUNG UND SICHERUNG
Ein Passkey, der nur auf einem Computer oder Telefon gespeichert ist, ist nicht sehr nützlich.
Kann man sich von einem anderen Gerät aus anmelden?
Was ist, wenn Ihr Gerät gestohlen wird, verloren geht oder auseinander fällt?

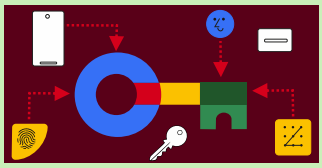
Für die Speicherung und Sicherung gibt es eine Reihe von Antworten.
Aber jede ist eine andere Geschichte
Das ist ein Grund dafür, dass Passkeys als eine besondere Option für die zu lösenden Umstände angesehen werden, aber ich denke, sie sind in ihrer Strategie widersprüchlich...

❶ Sie könnten die Passkeys im Passwort-Manager speichern, der sie verschlüsselt und in der Cloud sichert. Das macht Sie aber angreifbar, wenn man den Speicher knacken kann. Die Cloud hilft zwar, alle Ihre Geräte zu synchronisieren, aber ich habe keinen Glauben und kein Vertrauen in die "Cloud". Es ist eigentlich ein Marketingname für etwas Uraltes: Den Computer über eine Internetverbindung oder ein Kabel mit einem Speicher zu verbinden, der aber nicht in Ihren Händen liegt. Dies ist auch gegen die Regel, niemandem den Schlüssel zu überlassen.

❷ Es gibt immer noch die altmodische Möglichkeit, den Schlüssel unter anderen Gegenständen zu verstecken und sicher in Ihrem Büro aufzubewahren. Aber der Schlüssel kann natürlich gefunden und dann gestohlen werden. Auch müssen Sie den Schlüssel manuell auf das Gerät übertragen, wenn er dort gebraucht wird

❸ Es gibt auch eine physische Lösung: einen USB-Stick Passkeys werden auf einem physischen Sicherheitsschlüssel erstellt und gespeichert, den Sie über USB3 einstecken.
Um sich auf einem anderen Gerät anzumelden, stecken Sie den Sicherheitsschlüssel ein, wenn Sie dazu aufgefordert werden.
Dies kann sogar mit Ihrem Handy geschehen.
Auf diese Weise erstellte Passkeys können nicht kopiert werden.
Nur kürzlich erstellte Sicherheitsschlüssel unterstützen dies.

❹ Passkeys werden auf einem hochsicheren Chip in Ihrem Computer oder Telefon erstellt und gespeichert (z.B. ein TPM oder Secure Enclave, das in den meisten Geräten der letzten Jahre vorhanden ist).
Ein TPM (Trusted Platform Module) wird verwendet, um die Sicherheit Ihres PCs zu verbessern.
Es wird von Diensten wie BitLocker-Laufwerksverschlüsselung, Windows Hello und anderen verwendet, um kryptographische Schlüssel sicher zu erstellen und zu speichern und um zu bestätigen, dass das Betriebssystem und die Firmware auf Ihrem Gerät
und die Firmware auf Ihrem Gerät so sind, wie sie sein sollen, und nicht manipuliert wurden.
Sichere Enklave
Moderne Computergeräte verfügen oft über einen speziellen Hardware-Chip, auf dem der wichtige Informationen wie Verschlüsselungsschlüssel und Hashes speichert.
Bei PCs handelt es sich dabei um ein Trusted Platform Module (TPM), während er bei mobilen Geräten wie Android und iPhones wird es Secure Enclave genannt...



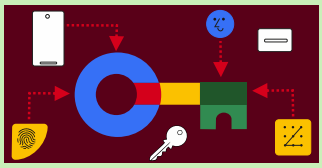
Wie Punkt ❸ können diese Passkeys nicht kopiert werden.
Antwort 3 und 4 sind weniger bequem (und Antwort 3 kostet etwas Geld: den Kauf eines Sicherheitsschlüssels).
Aber sie bieten ein höheres Maß an Sicherheit gegen den Diebstahl Ihrer Geräte.
Bei Antwort 1 könnte jemand, der Ihren Computer stiehlt, in der Lage sein, die Passkeys zu kopieren, wenn Ihr Passwortmanager nicht gesperrt ist.

Außerdem lösen die Lösungen 3 und 4 nicht wirklich das Problem des "verlorenen Geräts".
Wenn Sie eine dieser Lösungen verwenden, sollten Sie mehrere Passkeys auf verschiedenen Geräten als Backup speichern.
Alternativ können Sie sich auch auf die Wiederherstellung Ihres Kontos per E-Mail verlassen.

IST ES RATSAM, PASSKEYS ZU VERWENDEN?
Wie bei anderen Fragen zu Sicherheit und Datenschutz hängt die Antwort von verschiedenen Faktoren ab.
Im Allgemeinen sind Passwörter jedoch für die meisten Menschen ratsam.
Wenn Sie derzeit einen Passwort-Manager verwenden, lange und eindeutige Passwörter für jede Website erstellen und konsequent die Autofill-Funktion für die Anmeldung nutzen (im Gegensatz zum manuellen Kopieren und Einfügen von Passwörtern), bieten Passkeys ein etwas höheres Maß an Schutz bei gleichzeitig deutlich mehr Komfort.

Wenn Sie derzeit keine Passwortverwaltung verwenden, wird die Einführung von Passkeys Ihre Sicherheitsmaßnahmen erheblich verbessern (und auch die Einführung einer Passwortverwaltung erforderlich machen).
Wenn Sie auf Websites eine Zwei-Faktor-Authentifizierung (2FA) verwenden, bieten Passkeys mehr Komfort und potenziell mehr Sicherheit.
Die 2FA-Lösungen per SMS oder Authentifizierungs-App sind anfällig für Phishing-Angriffe, da betrügerische Websites den Einmalcode von den Benutzern anfordern und ihn anschließend zusammen mit dem gefälschten Passwort an die legitime Website übermitteln können.
Passkeys bieten im Vergleich zu 2FA-Lösungen per SMS oder Authentifizierungs-App mehr Sicherheit, da sie immun gegen Phishing-Angriffe sind.
Im Gegensatz zu SMS oder Authentifizierungs-Apps sind Passkeys nicht anfällig für die Täuschung durch betrügerische Websites, da der Browser jeden Passkey genau mit der entsprechenden Website verknüpfen kann.
Die 2FA mit Sicherheitsschlüssel ist nicht anfällig für Phishing-Angriffe, so dass der Wechsel von der 2FA mit Sicherheitsschlüssel zu einem Passkey in erster Linie Bequemlichkeit bietet, da ein zusätzlicher Anmeldeschritt und die Notwendigkeit, sich ein zusätzliches Passwort zu merken, wegfallen.
Wenn Sie Ihre Passkeys auf einem Sicherheitsschlüssel speichern, der mit einer PIN oder einer biometrischen Authentifizierung abgesichert ist, können Sie vergleichbare Ergebnisse erzielen wie mit der Zwei-Faktor-Authentifizierung (2FA) mit Sicherheitsschlüssel.
Die Speicherung von Passkeys in einem Passwort-Manager verringert die Sicherheit bis zu einem gewissen Grad, da unbefugte Personen, die sich Zugang zum Passwort-Manager verschaffen und die Passkeys verwenden können, ohne in den Besitz des Sicherheitsschlüssels zu gelangen.

OHNE PASSWORT LOGIN MIT PASSKEYS



Noch im Jahr 2023 ist der Grad der Unterstützung für **Passkeys** sehr uneinheitlich, insbesondere wenn es um die Synchronisierung geht.
Adam Langley (*Senior Staff Engineer bei Google*) gibt Beispiele für die Einschränkungen der verschiedenen Passwortverwaltungssysteme.
Er erklärt, dass **Windows Hello** überhaupt nicht synchronisiert, **Google Password Manager** nur zwischen Android-Geräten synchronisiert und **iCloud Keychain** ausschließlich auf **Apple**-Geräten funktioniert. Selbst nach der Lösung dieser Probleme wird die Synchronisierung von Daten zwischen verschiedenen Ökosystemen (**wie iOS und Windows**) weiterhin eine große Herausforderung darstellen.
Passwort-Organisatoren von Drittanbietern wie **1Password, Bitwarden und Dashlane** bieten Passkey-Funktionen und können Daten über mehrere Plattformen hinweg synchronisieren.
Es ist jedoch zu beachten, dass diese Dienste derzeit nicht alle Plattformen unterstützen.
Zum Beispiel bietet 1Password im Oktober 2023 **keine vollständige Unterstützung für Passkeys** auf Android.

Wenn Sie mit **Passkeys** mit einem **Einwegkonto** experimentieren möchten,
können Sie eines auf passkeys.io oder webauthn.io erstellen.
Wenn Sie gerne an der Spitze des technologischen Fortschritts stehen, möchte ich Sie ermutigen, Passkeys auszuprobieren. Während der Einführungsphase werden Sie vielleicht auf Hindernisse stoßen und gezwungen sein, auf die altbewährte und umstrittene Methode der Verwendung eines Passworts zurückzugreifen.

Benutzer können ein Konto auswählen, mit dem sie sich anmelden möchten.
Das Eingeben des Benutzernamens ist nicht erforderlich.

- Benutzer können sich mit der **Bildschirmsperre** des Geräts authentifizieren,
  z. B. mit einem Fingerabdrucksensor, einer Gesichtserkennung oder einer PIN.
- **Sobald ein Passkey erstellt und registriert ist,** kann der Benutzer nahtlos zu einem neuen Gerät wechseln und es sofort verwenden, ohne sich erneut anmelden zu müssen (*im Gegensatz zur herkömmlichen biometrischen Authentifizierung, die auf jedem Gerät eingerichtet werden muss*).

Wenn sich ein Benutzer bei einem Dienst anmelden möchte, der Passkeys verwendet,
hilft ihm sein Browser oder Betriebssystem bei der Auswahl und Verwendung des richtigen Passkeys.
Das funktioniert ähnlich wie bei gespeicherten Passwörtern.

Um sicherzugehen, dass nur der **rechtmäßige Besitzer** einen Passkey verwenden kann, wird das System ihn auffordern, sein Gerät zu entsperren.
Dies kann mit einem **biometrischen Sensor**, wie bereits erläutert, einer **PIN** oder einem **Muster** erfolgen.
Sie können entweder eine **nicht biologische Basis** oder **eine PIN** oder **ein Muster** wählen

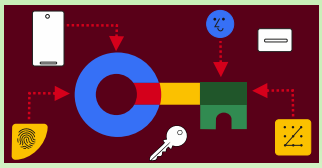
Um einen Passkey für eine Website oder Anwendung zu erstellen, muss sich ein Benutzer zunächst bei dieser Website oder Anwendung registrieren.

❶ Rufen Sie die Website/Anwendung auf
und melden Sie sich mit der vorhandenen Anmeldemethode an.
❷ Klicken Sie auf die Schaltfläche Einen Passkey erstellen.
❸ Überprüfen Sie die mit dem neuen Passkey gespeicherten Informationen.
❹ Verwenden Sie die Entsperrung des Gerätebildschirms, um den Passkey zu erstellen.

Eigentlich geschieht dies alles, wenn Sie ein Mobiltelefon verwenden.

Für WINDOWS gibt es einen anderen Ansatz...

OHNE PASSWORT LOGIN MIT PASSKEYS



PASSKEY-UNTERSTÜTZUNG AUF ANDROID UND CHROME

Passkeys können zwischen Geräten desselben Ökosystems synchronisiert werden.
Auf **Android** erstellte **Passkeys** werden zum Beispiel im Google Password Manager gespeichert.

HINWEIS: Ab Android 14 können Benutzer ihre Passwörter auch in Anwendungen von Drittanbietern zur Verwaltung von Anmeldeinformationen speichern.
Passkeys sind eine aufstrebende Technologie und die unterstützten Umgebungen entwickeln sich noch weiter.
Ab August 2023 speichert Chrome auf macOS und Windows Passkeys nur auf dem lokalen Gerät.

GOOGLE PASSWORT-MANAGER

Google Password Manager speichert, bedient und synchronisiert Passkeys auf Android und Chrome. Die Passwörter von Google Password Manager sind für alle Android-Apps, einschließlich Chrome und andere Browser, verfügbar.

Wenn der Nutzer einen **Passkey** auf einem Android-Gerät erstellt, wird er gespeichert und mit seinen anderen Android-Geräten synchronisiert.
Die Geheimnisse des Passkeys werden Ende-zu-Ende verschlüsselt.

Auf diese Weise stehen die Passwörter dem Nutzer auf allen Android-Geräten zur Verfügung, die Google Password Manager verwenden und mit demselben Google-Konto angemeldet sind.

Google Password Manager auf Chrome hilft bei der Erstellung und Anmeldung mit Passkeys. Je nach Desktop-Betriebssystem (**z.B. Chrome OS, iOS, macOS, Windows)** wird den Nutzern ein QR-Code angezeigt, mit dem sie einen auf ihrem mobilen Gerät gespeicherten Passkey sicher verwenden können, oder es wird eine Benachrichtigung angezeigt, die den Nutzer auffordert, sein Telefon zu entsperren, um den entsprechenden **Passkey** zu verwenden.

UNTERSTÜTZUNG DES PASSKEYS VON CHROME AUF VERSCHIEDENEN BETRIEBSSYSTEMEN

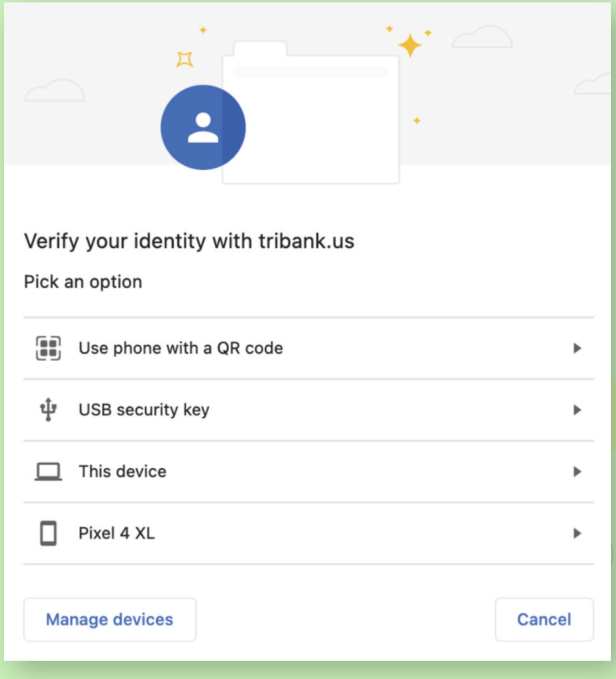
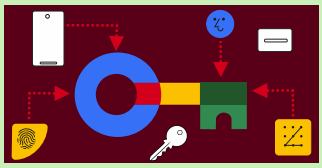


Abbildung 0: Authenticator-Auswahl

OHNE PASSWORT LOGIN MIT PASSKEYS



Chrome unterstützt auf allen Desktop-Plattformen die Verwendung von Passkeys von mobilen Geräten aus. Um einen Passkey von Ihrem **Android**- oder **iOS**-Gerät zu verwenden, wählen Sie die entsprechende Option, wenn Sie danach gefragt werden. *Siehe Abbildung 0: auf Seite 24 Artikelseite 12*. Um mehr darüber zu erfahren, wie Sie sich mit einem Telefon anmelden können, lesen Sie Anmeldung mit einem Telefon.

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

In den folgenden Abschnitten wird das Verhalten von Chrome auf verschiedenen Betriebssystemen beschrieben.

ANDROID

Chrome auf Android OS 9 oder höher unterstützt **Passkeys**. Die in **Chrome auf Android** erzeugten **Passkeys** werden im **Google Password Manager** gespeichert. Diese **Passkeys** sind auf allen anderen Android-Geräten verfügbar, solange der Google Password Manager verfügbar ist und das Google-Konto desselben Benutzers angemeldet ist.

WINDOWS

Chrome unter Windows speichert **Passkeys in Windows Hello,** das sie ab Oktober 2023 nicht mehr mit anderen Geräten synchronisiert. Wenn sich ein Nutzer zum ersten Mal auf einer Website mit **Chrome unter Windows** anmeldet, sollte er mit einem anderen Gerät, das bereits einen **Passkey** hat, einen **QR-Code** scannen. Danach kann er auf dem lokalen Windows-Gerät einen **Passkey** erstellen, den er dort künftig verwenden kann.

MACOS

Chrome auf macOS 13.5 und höher kann **iCloud Keychain** zum Speichern von **Passkey** verwenden. Die Passwörter im iCloud-Schlüsselbund werden über die Apple-Geräte des Benutzers hinweg synchronisiert und können von anderen Browsern und Apps verwendet werden.
Chrome auf macOS kann Passwörter auch in einem lokalen Profil speichern, d.h. sie werden nicht mit anderen Geräten synchronisiert. Die Speicherung von Passkeys in einem lokalen Profil ist in früheren Versionen von macOS verfügbar.

IOS / IPADOS

Chrome auf iOS 16 und iPadOS 16 verwendet iCloud Keychain zum Speichern von Passkeys. Die Passwörter im iCloud-Schlüsselbund werden über die Apple-Geräte des Benutzers hinweg synchronisiert und können von anderen Browsern und Apps verwendet werden.

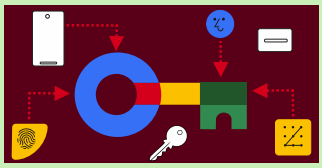
LINUX

Chrome unter Linux unterstützt keine Passkeys mit einem integrierten Plattform-Authentifikator.
Linux-Benutzer können **Passkeys** von einem anderen Gerät wie einem **Android-Telefon** oder einem **iPhone** verwenden, indem sie einen **QR-Code** scannen

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

OHNE PASSWORT LOGIN MIT PASSKEYS



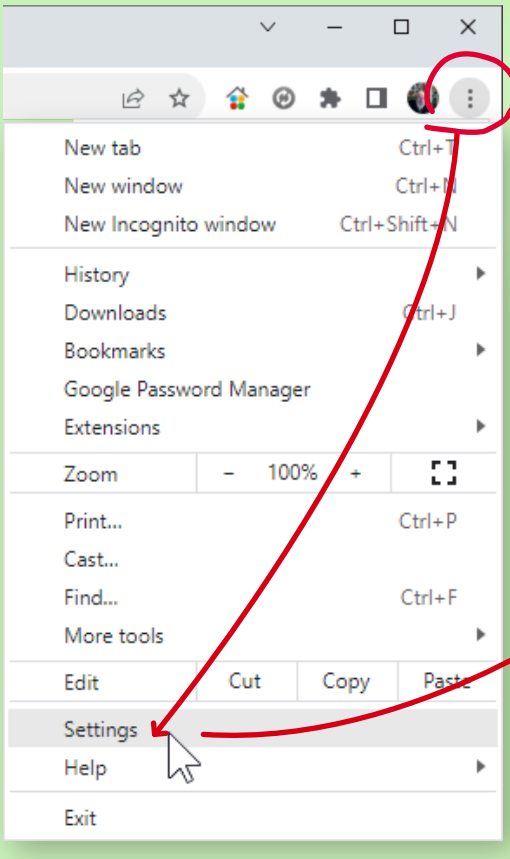


Abbildung 1 Button und Einstellungen

Um die richtigen Einstellungen zu finden, gehen Sie folgendermaßen vor:

- öffnen Sie Ihren Browser → oben rechts sehen Sie eine Button mit drei Punkten. Wenn Sie darauf klicken, klappt das Menu auf. Siehe das linke Bild.
- Sie sehen Einstellungen.
- Klicken Sie darauf und die nächste Seite wird angezeigt:
- Hier ist der Menüpunkt Sicherheit. Wählen Sie Sicherheit: Es öffnet sich ein neues kleines Fenster. Hier müssen Sie auf Ihr Konto verwalten klicken

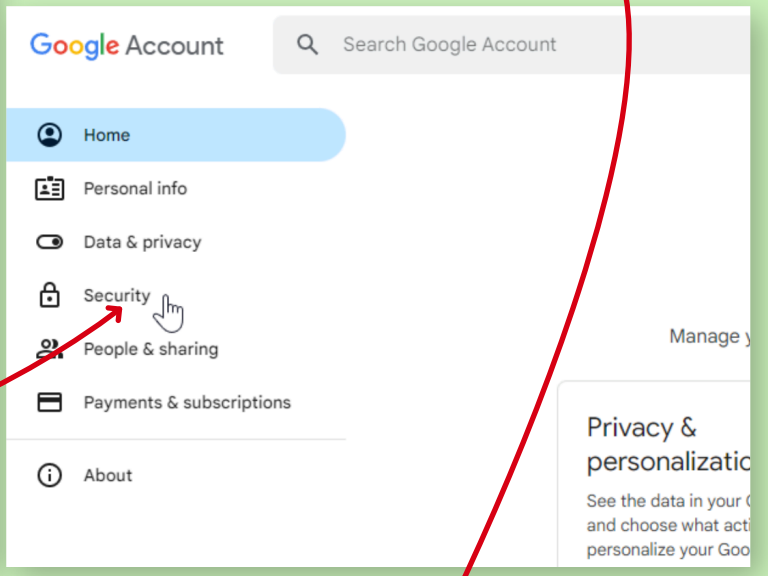


Abbildung 2 Sicherheit muss gewählt werden

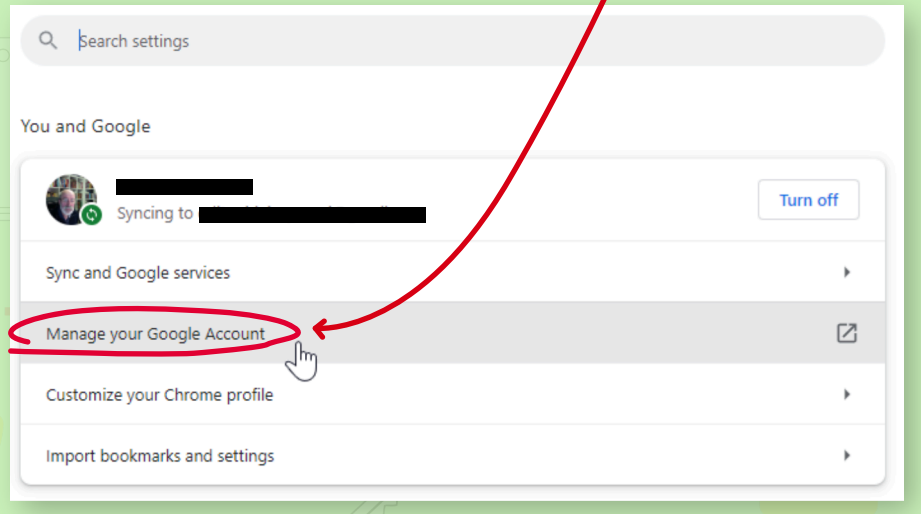
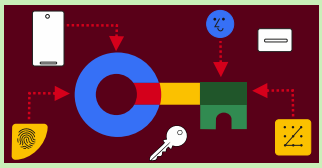


Abbildung 3 Sie müssen Ihr Konto verwalten, bevor Sie Einstellungen für den Passkey vornehmen können

OHNE PASSWORT LOGIN MIT PASSKEYS



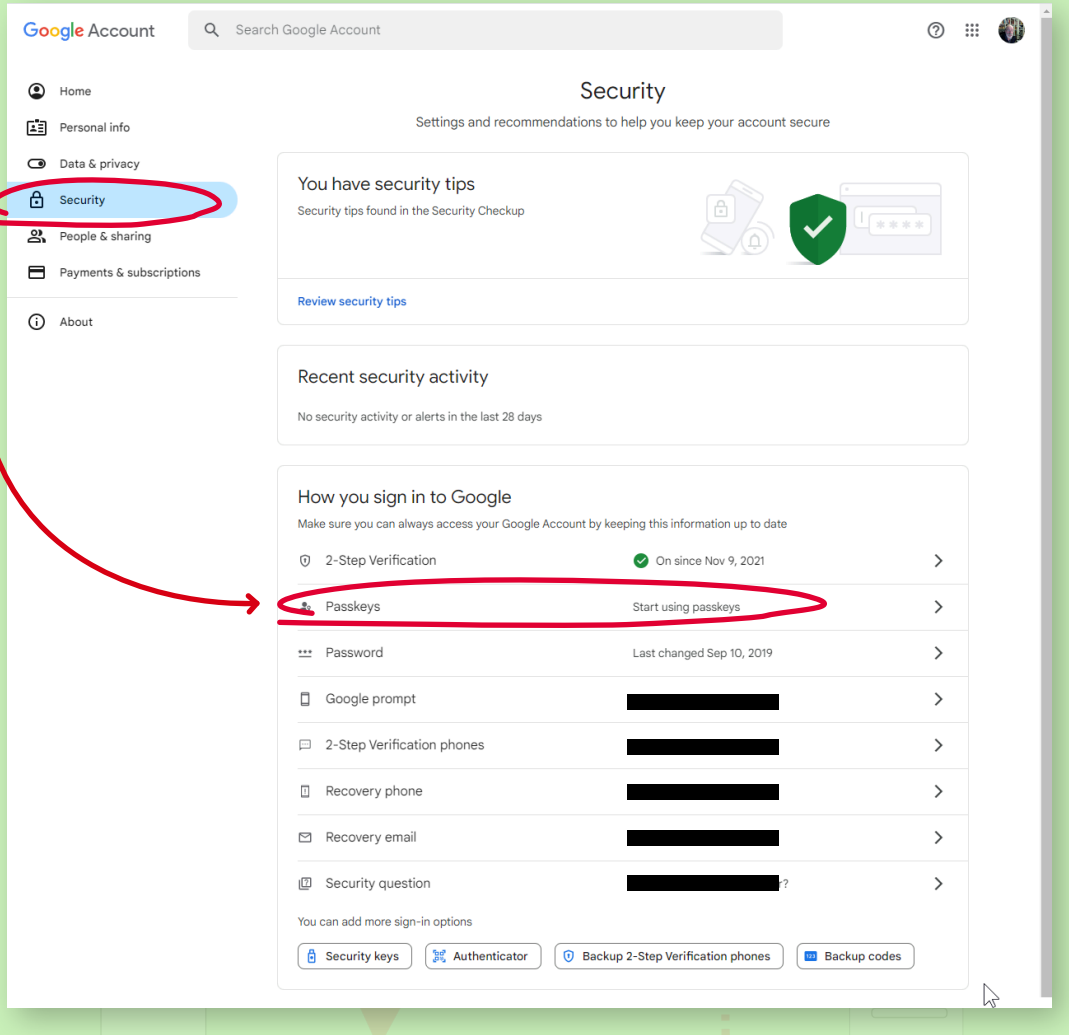
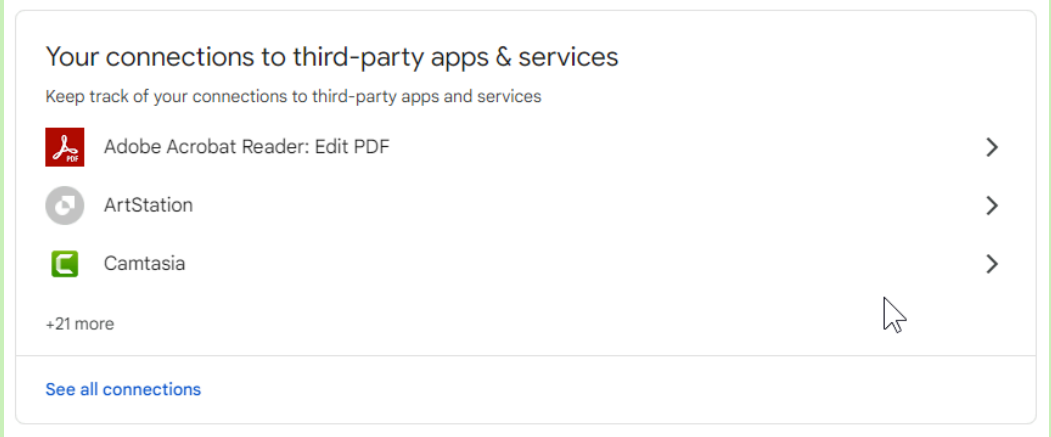
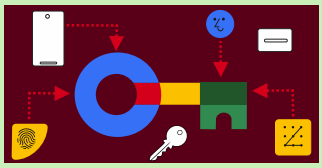
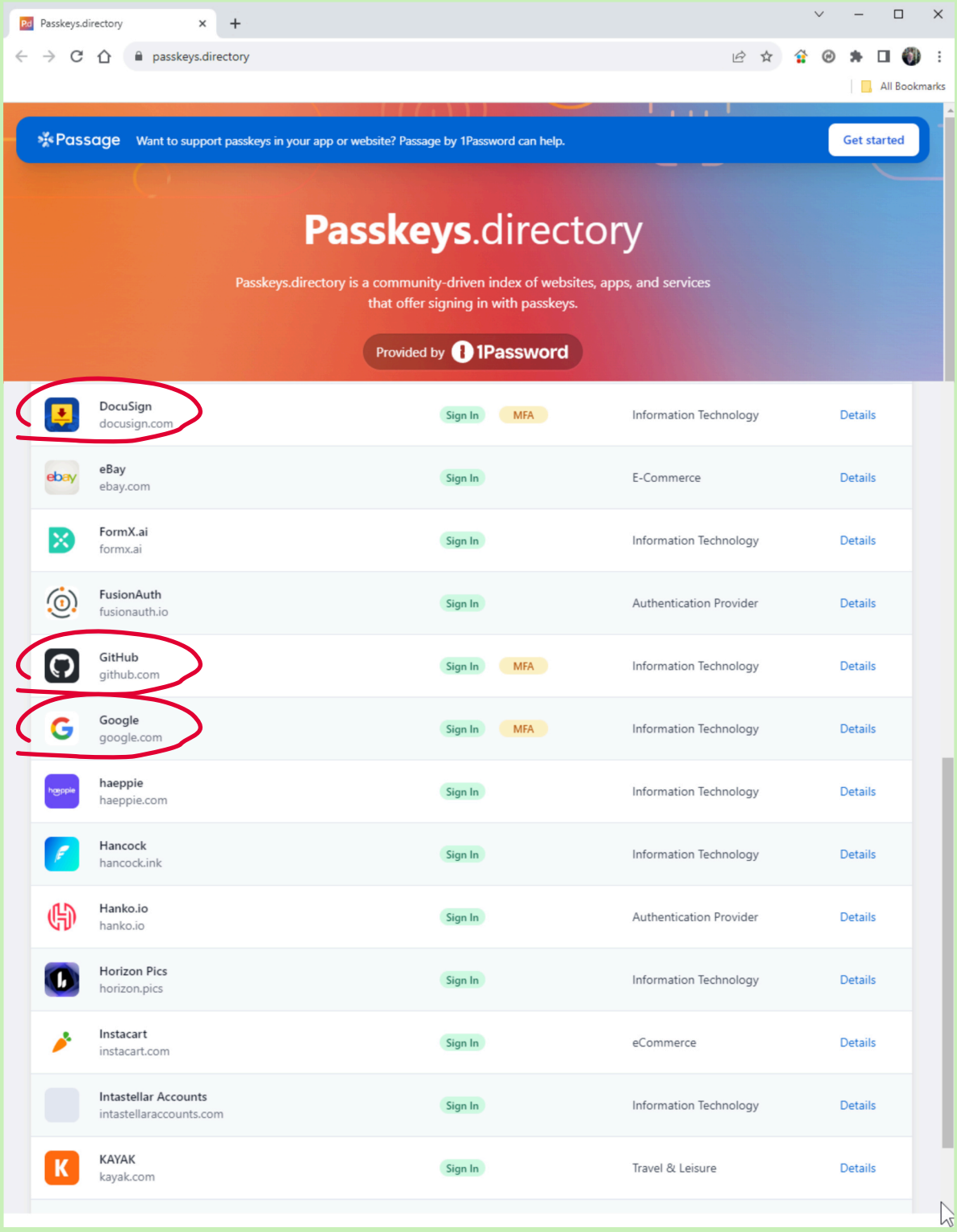


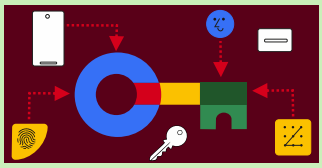
Abbildung 4 Wenn Sie den Menüpunkt Sicherheit wählen, wird eine Liste von Elementen angezeigt: Wählen Sie Passkeys



OHNE PASSWORT LOGIN MIT PASSKEYS







HÄUFIG GESTELLTE FRAGEN (FAQs)

- **Funktionieren Passkeys auch auf Geräten, für die keine Bildschirmsperre eingerichtet ist?**
  Es hängt von der Implementierung des Passwortmanagers ab, ob ein Anbieter von Anmeldeinformationen die Erstellung eines Passkeys und die Authentifizierung ohne die Abfrage des Wissensfaktors des Benutzers zulässt.
  Anbieter können Benutzer auffordern, eine PIN oder eine biometrische Bildschirm sperre einzurichten, bevor sie einen Passkey erstellen.

- **Wie können Passkeys, die auf einer Plattform (z.B. Android) registriert sind, zur Anmeldung auf anderen Plattformen (wie Web oder iOS) genutzt werden?**
  Ein auf **Android** registrierter passkey kann z.B. zur Anmeldung auf anderen Plattformen verwendet werden, indem Sie das Android-Telefon mit einem anderen Gerät verbinden.

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

Um eine Verbindung zwischen den beiden Geräten herzustellen, müssen die Benutzer die Website, auf dem Gerät aufrufen, auf dem noch kein Passkey registriert ist,
einen QR-Code scannen und dann die Anmeldung auf dem Gerät bestätigen, auf dem sie den Passkey erstellt haben (in diesem Fall das Android-Gerät).

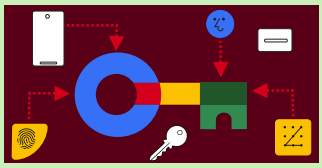
DER HAUPTSCHLÜSSEL VERLÄSST NIEMALS DAS ANDROID-GERÄT,
Daher schlagen Apps in der Regel vor, einen neuen P**asskey** auf dem anderen Gerät zu erstellen, um die Anmeldung beim nächsten Mal zu erleichtern. Dieser Ablauf funktioniert in ähnlicher Weise auch für andere Plattformen.

- **Kann ich synchronisierte Passkeys von einem Plattformanbieter zu einem anderen verschieben?**
  Die **Passkeys** werden bei dem von der Plattform definierten Anbieter von Anmeldeinformationen gespeichert.
  Bei einigen Plattformen, wie z.B. Android, können Benutzer den Anbieter ihrer Wahl aussuchen (*ein System oder ein Passwortmanager eines Drittanbieters*), beginnend mit **Android 14**,
  der möglicherweise in der Lage ist, Passkeys über verschiedene Plattformen hinweg zu synchronisieren.
  Unterstützung für die direkte Übertragung von Kennwörtern von einem Plattformanbieter zu einem anderen ist zur Zeit nicht verfügbar

- **Kann ein Benutzer seine Passwörter mit anderen Android-Geräten als Google synchronisieren?**
  Passwörter werden nur innerhalb des Ökosystems des Geräts synchronisiert (*d.h. von Android zu Android mit Google Password Manager*), aber nicht innerhalb des gesamten Ökosystems.
  Android öffnet die Plattform (**beginnend mit Android 14**), damit die Benutzer auswählen können, welchen sie verwenden möchten (*z. B. einen Passwortmanager eines Drittanbieters*).
  Das wird Anwendungsfälle wie die Synchronisierung von Passkeys zwischen verschiedenen Ökosystemen ermöglichen (*je nachdem, wie offen andere Plattformen sind*).

- **Was sollten Entwickler bei Geräten und Plattformen tun, die Passkeys nicht unterstützen**?
  Entwicklern wird empfohlen, die bestehenden Anmeldeoptionen in ihrer App vorerst beizubehalten damit sie auch weiterhin für Geräte und Oberflächen zur Verfügung stehen, die keine Passkeys unterstützen.

OHNE PASSWORT LOGIN MIT PASSKEYS



- Kann eine **RelyingParty** ein Konto angeben, mit dem sich der Benutzer anmelden soll?
  RelyingParties (*Anwendungen von Drittanbietern*) können die '`allowCredentials`' mit einer Liste von Credentials IDs auffüllen die von ihrem App-Backend gesendet werden und angeben welche **Passkeys** zur Authentifizierung des Benutzers verwendet werden sollen.
  HINWEIS: Der erste Schritt zur Aktivierung der Passkey-Unterstützung für Ihre Android-App ist die Verknüpfung Ihrer App und der Website.
  Hosten Sie dazu eine **Digital Asset Links JSON-Datei** auf Ihrer Website und fügen Sie einen Link zur **Digital Asset Link-Datei** in das Manifest Ihrer App ein.
  Damit zeigen Sie, dass Sie sowohl Eigentümer der Website als auch der App sind.
  Weitere Informationen finden Sie in der Dokumentation zu **Digital Asset Links**.

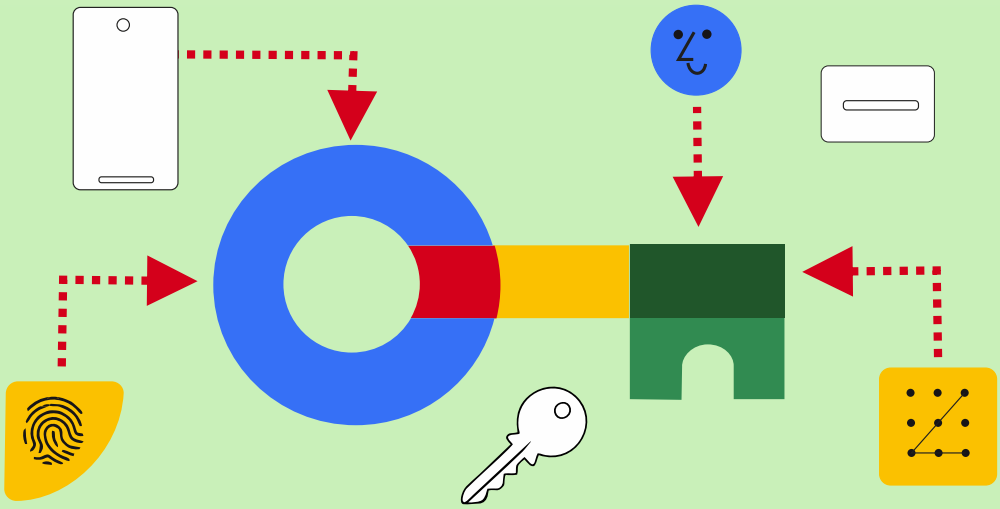
- Für Passkeys, die in **Chrome** auf anderen Plattformen erstellt wurden:
  Wenn der Passkey in **Chrome** auf anderen Plattformen (*Mac*, *iOS*, *Windows*) erstellt wurde, dann geht das nicht.
  Weitere Informationen finden Sie in den unterstützten Umgebungen.
  In der Zwischenzeit können sich die Benutzer mit dem Telefon anmelden, auf dem sie den Hauptschlüssel erstellt haben.

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

PROBIEREN SIE ES SELBST AUS

Sie können **passkeys** in dieser Demo ausprobieren:

`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

FRÜHJAHRSANGEBOT 2024

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

Erweiterte Debugsuche unter der Haube mit RAD Studio 11.3 Alexandria. von Max Kleiner

Starter Expert D11

Wie Sie sicher wissen, können Sie den Debugger anweisen, bestimmte Arten von Exceptions zu ignorieren. Abbildung 1 auf dieser Seite zeigt die Optionen für Exceptions vom Delphi Code und native OS-Exceptions. Fügen Sie der Liste der Delphi Exceptions eine Ausnahmeklasse hinzu, und alle Exceptions dieses Typs und aller davon abgeleiteten Typen werden in Ihrem Programm berücksichtigt, ohne dass Delphi eingreift.

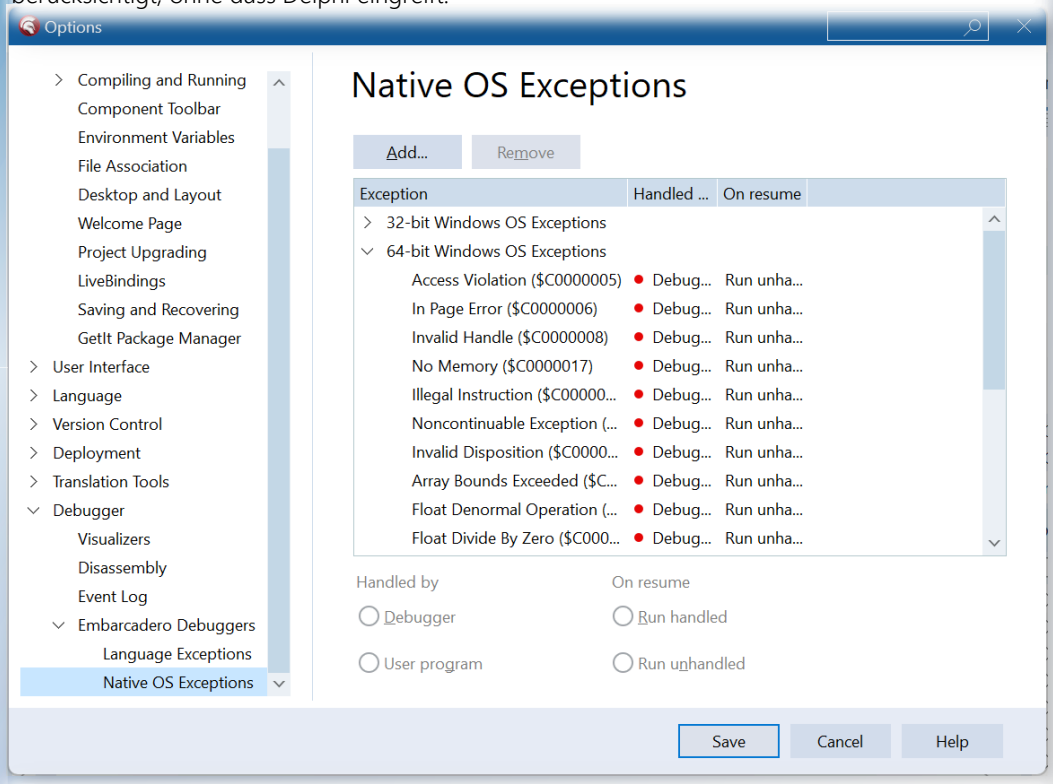


Abbildung 1

In den Standardeinstellungen benachrichtigt Sie die Delphi IDE, wenn in Ihrem Programm eine Exception auftritt, wie in Abbildung 2 auf dieser Seite.
Wichtig ist, dass zu diesem Zeitpunkt noch kein Code Ihres Programms zur Behandlung von Exceptions ausgeführt wurde.

Das ist alles Delphi intern. Sein besonderer Status als Debugger erlaubt es ihm, jede Exception in Ihrem Programm zuerst zu melden, noch bevor Ihr Programm davon erfährt.
Es ist also nicht so einfach, Ihre 64-Bit-Anwendung in der IDE zu öffnen, die 64-Bit-Debugger-Optionen hinzuzufügen und zu aktivieren und Ihre Anwendung als 64-Bit-Windows-Anwendung mit dem richtigen Debugging zu kompilieren.

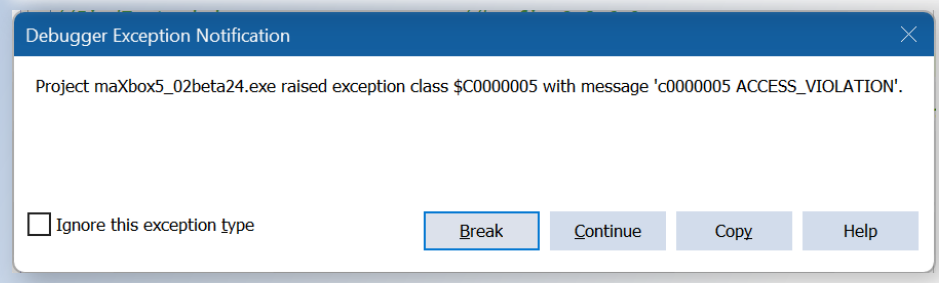


Abbildung 2

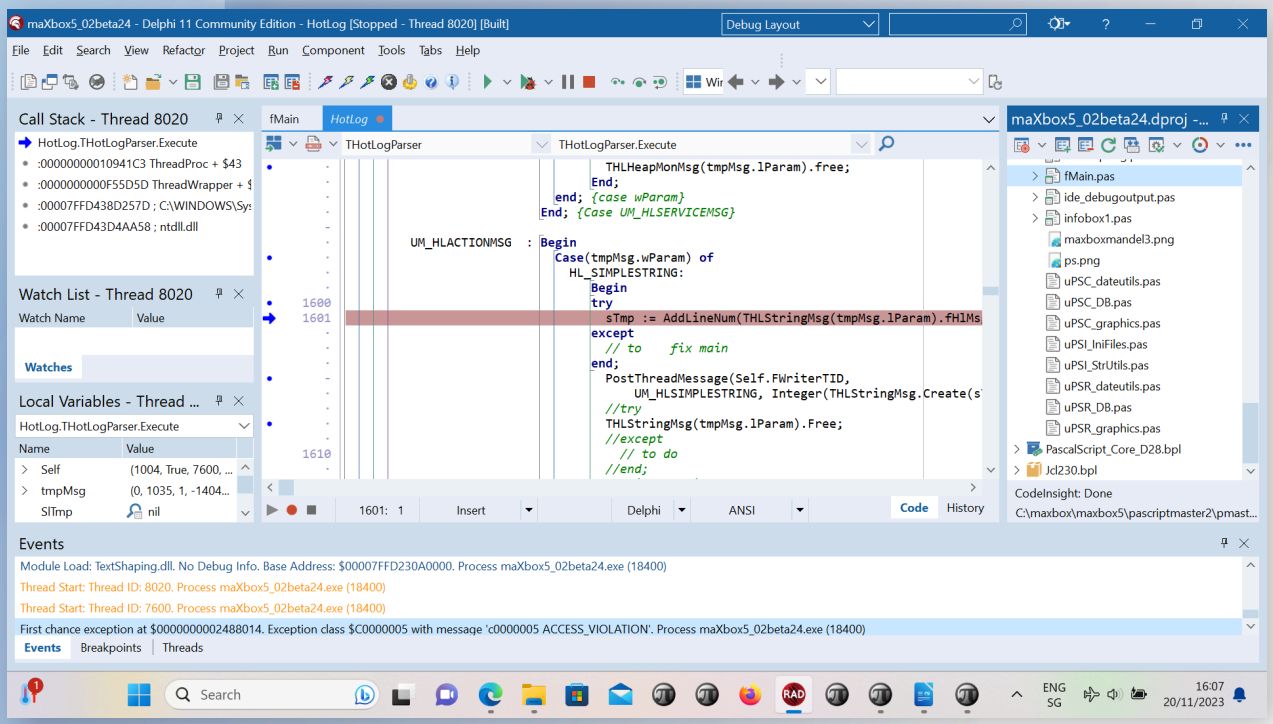
Abbildung 3

Sie können die Optionseinstellungen in jeder Konfiguration ändern, auch in der Basis.
Sie können die Debug- und Release-Konfigurationen löschen, aber Sie können die Basiskonfiguration nicht löschen oder verschieben.
Wenn Sie den Quellcode von **maXbox4** durchforsten, scheint es unmöglich zu sein, über 3354 Einheiten auf anständige und korrekte Weise auf **maXbox5** alias die 64-Bit-Version zu migrieren.
Einige Leute haben sich darüber beschwert, dass dies Probleme verursacht, ohne jedoch ein richtiges Beispiel zu nennen. Außerdem hat Embarcadero kürzlich die Empfehlung zur Verwendung von **`FreeAndNil`** in sein Handbuch aufgenommen (*endlich*!).
HINWEIS: Kompilieren Sie die Anwendung immer im Release- und Debug-Modus.
Stellen Sie sicher, dass die Projektoptionen für den Debug-Modus richtig eingestellt sind.
Die DEFAULT-Einstellungen für den Debug-Modus sind NICHT korrekt/vollständig - zumindest nicht in Delphi XE7 und Tokyo.
Früher wurden die korrekten Optionen für den Debug-Modus gesetzt, falls es welche gab.
Aktivieren Sie also und studieren Sie Themen wie:

- "Stack frames"
- "Map-Datei-Erzeugung (detailliert)"
- "Bereichsprüfung",
- "Symbolreferenz-Informationen"
- "Debug-Informationen"
- "Überlaufprüfung"
- "Zusicherungen (Assertions)"
- "DCU Debugging"

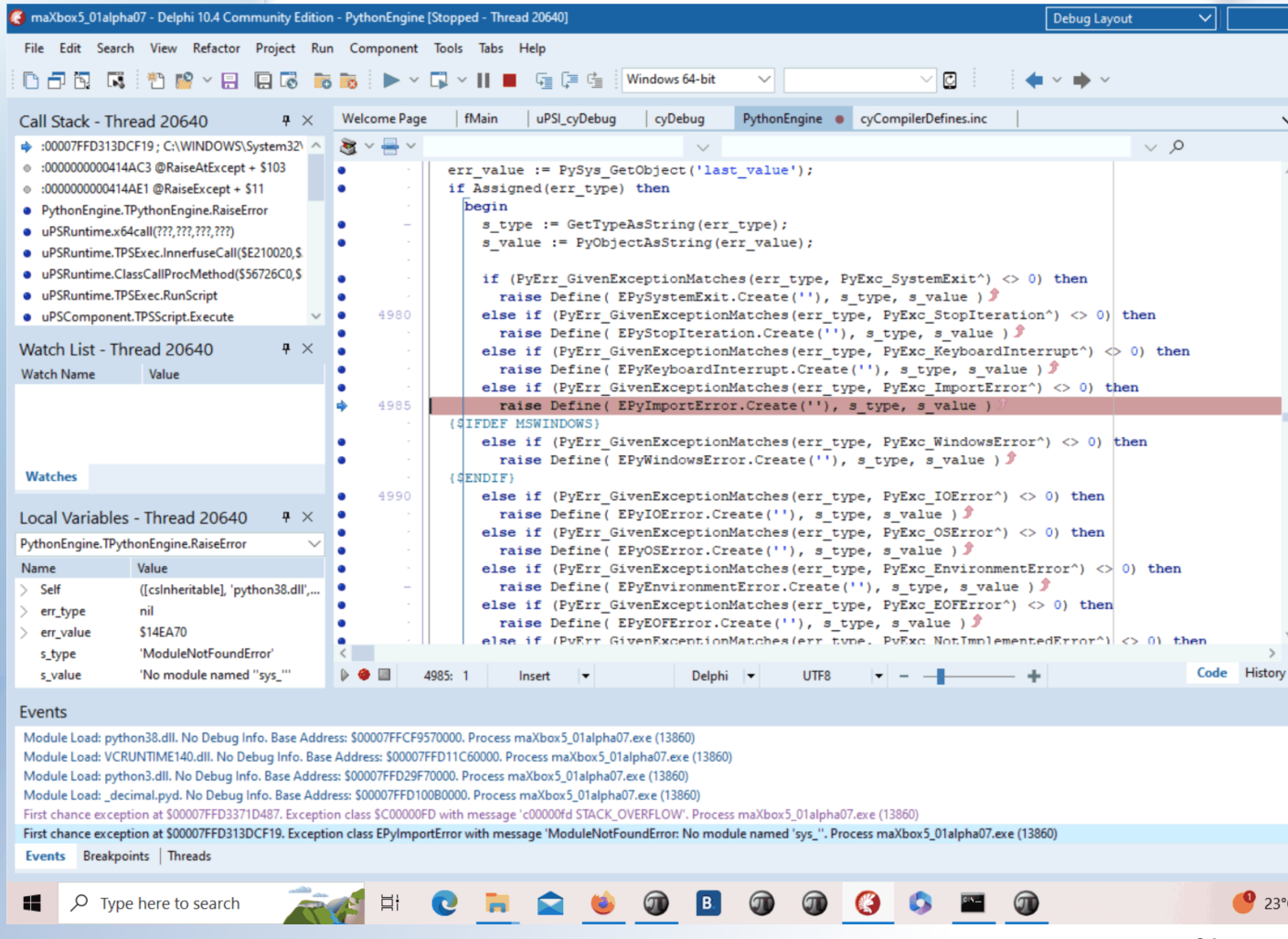
Meistens erhalten Sie auf Anhieb einige Exceptions. Nachdem Sie die Debugger-Ausnahmebenachrichtigung in der Abbildung unterbrochen haben, erhalten Sie genau die Zeile, um den Fehler zu beheben (*es war ein Nil-Zeiger*):

Figure 4



Sie können die "erweiterten Breakpoints" von Delphi verwenden, um die Behandlung von Exceptions in einem bestimmten Bereich des Codes zu deaktivieren. Zunächst setzen Sie einen BreakPoint in der Codezeile, in der die IDE Exceptions ignorieren soll. Klicken Sie mit der rechten Maustaste auf den BreakPoint-Punkt am linken Rand und öffnen Sie den BreakPoint-Eigenschaften-Dialog.
Im erweiterten Bereich finden Sie einige Kontrollkästchen. Deaktivieren Sie das Kästchen "Unterbrechen", um zu verhindern, dass der Debugger Ihr Programm in dieser Zeile unterbricht, und aktivieren Sie das Kästchen "Spätere Exceptions ignorieren". Setzen Sie anschließend einen weiteren BreakPoint an der Stelle, an der der Debugger die Behandlung von Exceptions wieder aufnehmen soll. Ändern Sie seine Eigenschaften, um nachfolgende Exceptions zu behandeln.
Was sind nachfolgende Exceptions:
Sie behandeln alle nachfolgenden Exceptions, die vom aktuellen Prozess während der aktuellen Debug-Sitzung ausgelöst werden (der Debugger hält bei Exceptions an, basierend auf den aktuellen Ausnahmeeinstellungen in:
Extras → Optionen → Debugger-Optionen → Embarcadero Debugger → DelphiExceptions).
Diese Option hält bei allen Exceptions an. Verwenden Sie sie, um das normale Ausnahmeverhalten wieder einzuschalten, nachdem ein anderer BreakPoint das normale Verhalten mit der Option Nachfolgende Exceptions ignorieren deaktiviert hat.
Sie ist in einem Codeblock mit der Option Ignoriere nachfolgende Exceptions sinnvoll.
Sie ignoriert alle nachfolgenden Exceptions, die vom aktuellen Prozess während der aktuellen Debug-Sitzung ausgelöst werden (der Debugger hält bei keiner Ausnahme an). Normalerweise wissen Sie, dass die Ausführung angehalten wird: die traditionelle und standardmäßige Aktion eines Breakpoints).
Verwenden Sie die Option Nachfolgende Exceptions ignorieren mit Nachfolgende Exceptions behandeln als Paar. Sie können bestimmte Codeblöcke mit dem Paar Ignorieren/Behandeln umgeben, um alle Exceptions zu überspringen, die in diesem Codeblock auftreten, wie in der folgenden Abbildung 5 auf dieser Seite.

Figure 5



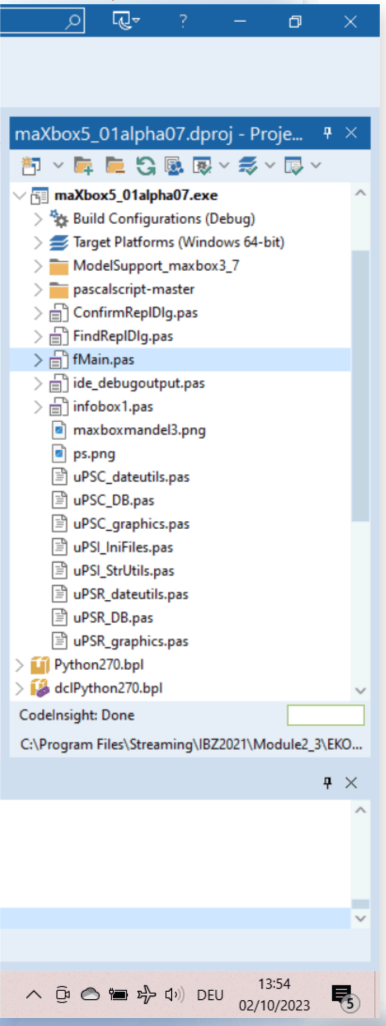
DEN DEBUGGER VORBEREITEN

Wenn in **Delphi 11** - und wahrscheinlich auch in den meisten anderen Versionen - eine Exception aus der Execute-Methode ausgelöst wird, ohne behandelt zu werden, wird sie von der Funktion, die Execute aufgerufen hat, abgefangen und in der Eigenschaft **`FatalException`** des Threads gespeichert. (*Siehe in* `Classes.pas, ThreadProc`.)
Mit dieser Exception wird nichts weiter unternommen, bis der Thread freigegeben wird, woraufhin auch die Exception freigegeben wird.

```
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))
```

Im Gegensatz zur Windows API TerminateThread MSDN, die die sofortige Beendigung des Threads erzwingt, fordert die Terminate-Methode lediglich die Beendigung des Threads an.
Dadurch kann der Thread alle Aufräum- und Abschlussarbeiten durchführen, bevor er beendet wird.

Abbildung5 Fortsetzung



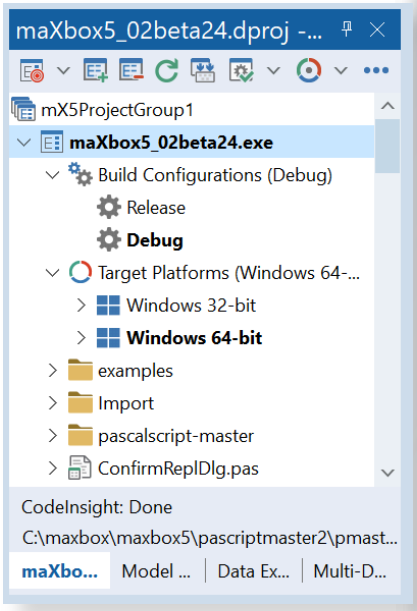
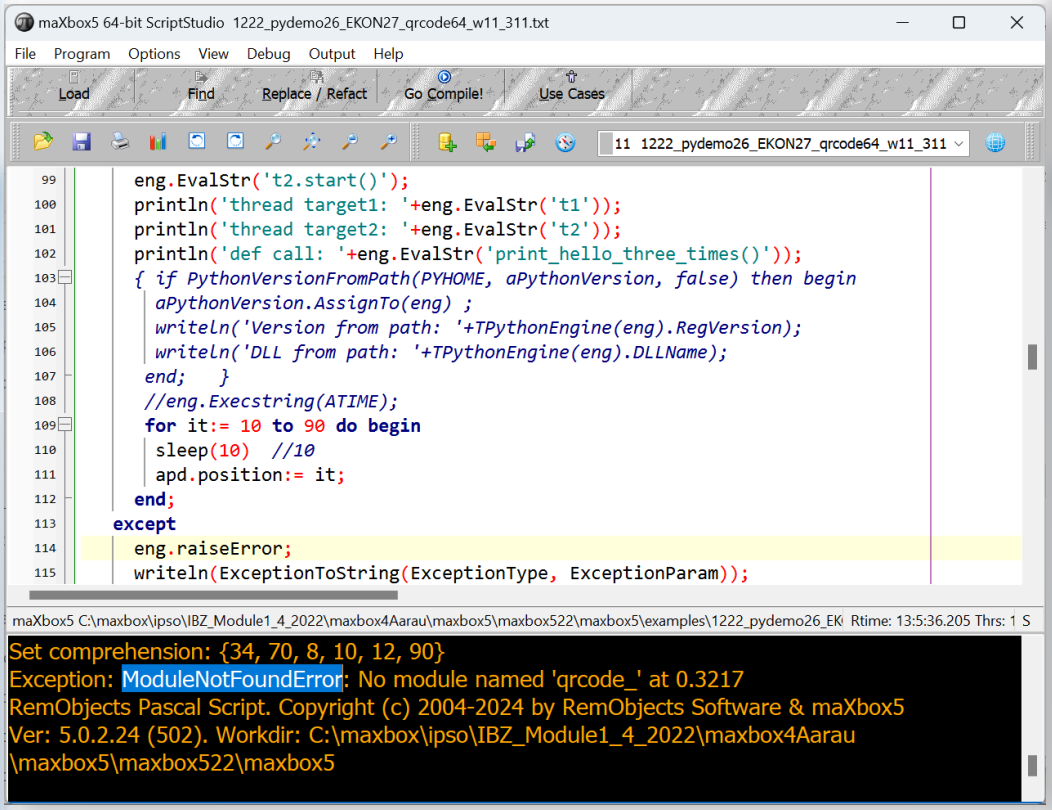


Abbildung 6

Um die **Exceptions** abzufangen, die innerhalb Ihrer **Thread-Funktion** auftreten, fügen Sie der Implementierung der **Execute-Methode** einen `try...except`-Block hinzu!
So habe ich das Projekt und das Debugging im Debug-Modus vorbereitet:
Abbildung 6 auf dieser Seite

Das Seltsame ist, dass ich meinen Pascal-Aufruf in ein try except verpackt habe, verpackt habe, das Handler für `AccessViolationException`, `COMException` und alles andere hat, aber wenn **Delphi 10.4 oder Studio 11.3** die `AccessViolationException` abfängt, bricht der Debugger beim Methodenaufruf (`doc.OCR`) ab.

Wenn ich einen Schritt weiter gehe, fährt er mit der nächsten Zeile fort, anstatt in den catch- oder except-Block zu gehen.
Also habe ich beschlossen, die Ausnahme im Skript in einer Laufzeitroutine abzufangen, um das meiste auf meiner Konsole von einer dynamischen Debugbox maXbox zu erhalten:



Außerdem habe ich die Variable `JITEnable` aktiviert, die steuert, wann der **Just-in-Time-Debugger** aufgerufen wird.
Für die Umstrukturierung der Quellen habe ich die neueste Revision mit den Patches aus Heft #202 (Commit 86a057c), aber ich kann die Dateien zunächst nicht kompilieren (`Core_D27`) die Teil des `PascalScript_Core_D27.dpk` für diese Plattform sind, weder für **Linux64, Win64** noch **MacOS64.**
Hier ist zunächst eine Quelltextausgabe, die die interne Ausnahmebehandlung für den 10.4 dccosx64 oder dcc64 Compiler zeigt (*ähnliche Ergebnisse gibt es für 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;
```

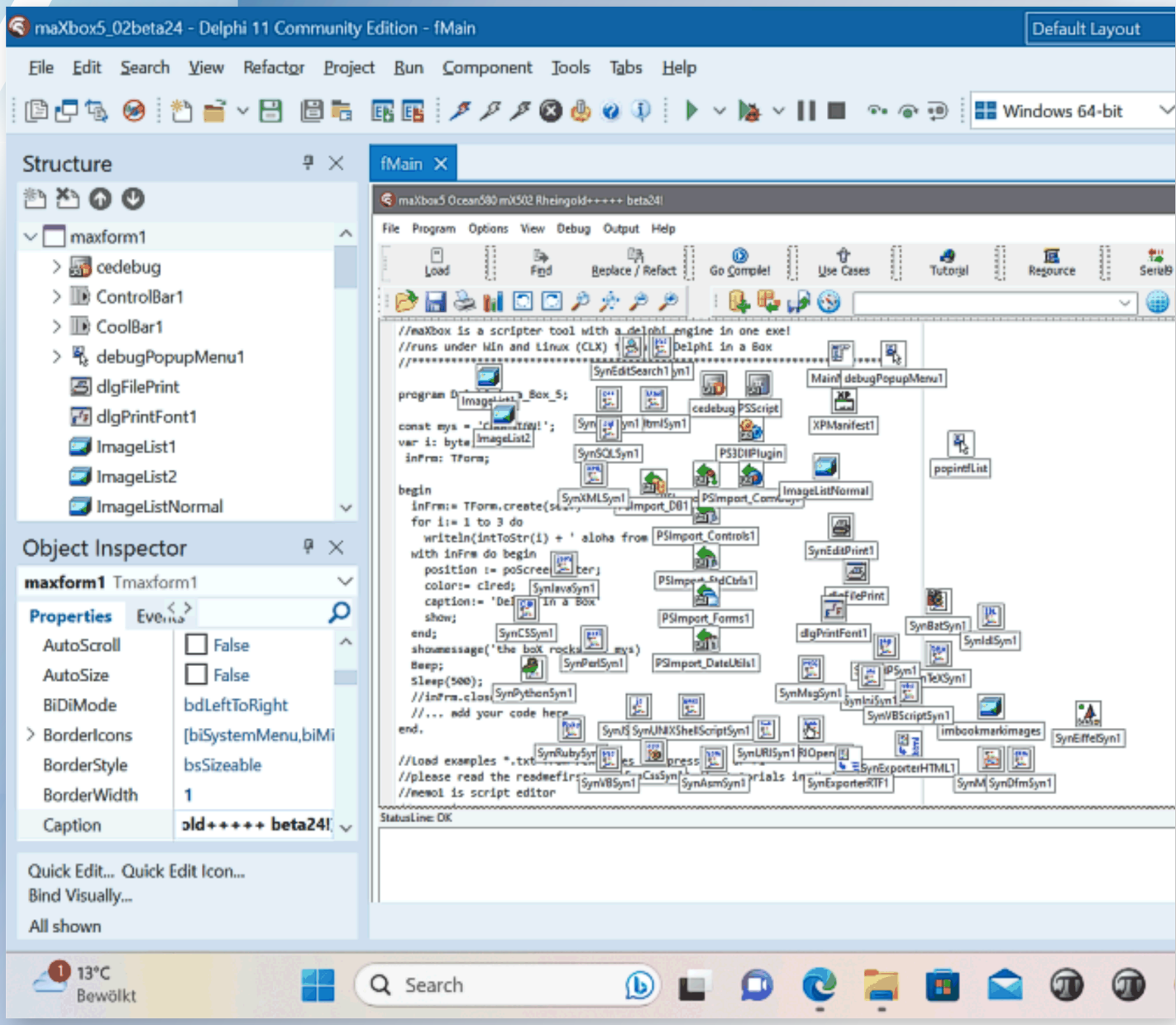
Dann sehe ich die ExceptionProc als eine First-Chance-Exception auf Adresse $0000000100419E9E..

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*)
```

Ein weiterer störender Punkt beim Debuggen war eine Umleitung zum Debuggen.
Da ich **SynPdf.pas** verwende, muss ich SynCommons.pas in mein Projekt aufnehmen.
Aber es scheint, dass ich mehr bekommen habe, als ich wollte. Wenn ich in der IDE debugge und versuche, in eine Routine zu springen, lande ich manchmal bei der Move-Prozedur von SynCommons.pas statt zum normalen Systemaufruf, zu dem ich gelangen möchte!
Die Lösung ist einfach: **FastCodem, Move und Fillchar** sind in der **SynCommons.pas** enthalten, die für Geschwindigkeit optimiert ist, welche durch **SmartLinking** die Exe nicht größer macht. Die Protokollierung und all die anderen Sachen sind nicht Teil davon.
Sie können sie entfernen, indem Sie die entsprechenden Zeilen im Initialisierungsblock dieser Einheit auskommentieren. In den neuen Versionen des Frameworks gibt es eine bedingte Einstellung, um es zu deaktivieren.

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



Wenn ich eine zweite Kompilierung in einer **CrossVCL** aus dem Menü wähle, teste ich immer auch den Debugger mit zwei Code-Funktionen als Ähnlichkeiten, aber mit unterschiedlichen Code-Lösungen, zum Beispiel **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;
```

Und dann habe ich vielleicht aus Intuition einen Build gemacht. Das AD ist verschwunden und ich habe meinen ersten Bildschirm, kompiliert und das Skript ausgeführt:
Eines der in der Zwischenzeit gelösten Probleme besteht darin, eine Access-Violation abzufangen, anstatt die App zum Absturz zu bringen. Es ist, als ob der Code in `uPSRuntime.pas` die Ursache für das Anhalten oder Beenden nicht finden konnte, wie im Folgenden beschrieben

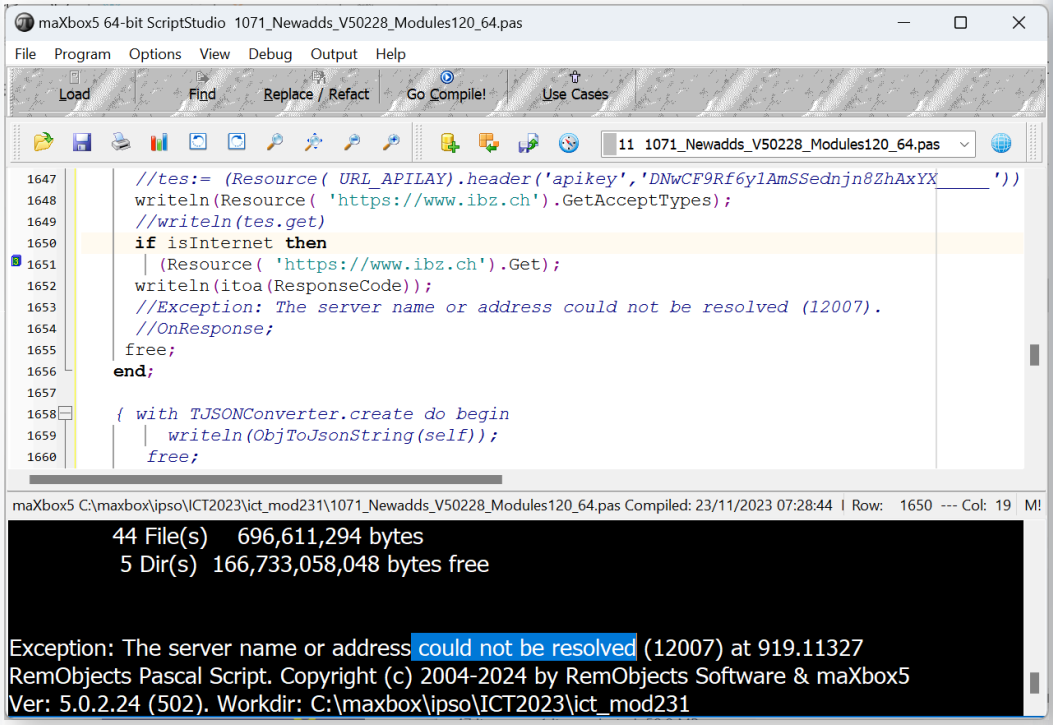
```
procedure TdynamicDll.Quit;
begin
   if not( csDesigning in ComponentState ) then begin
   {$IFDEF MSWINDOWS}
   MessageBox( GetActiveWindow, PChar(GetQuitMessage), 'Error',
                              MB_TASKMODAL or MB_ICONSTOP );
   ExitProcess( 1 );
   {$ELSE}
   WriteLn(ErrOutput, GetQuitMessage);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

Nach der Fehlersuche habe ich festgestellt, dass es sich um eine **First Chance Exception** handelt, die funktioniert, solange der Debugger mit break oder continue läuft, aber ohne Debugger verschwindet die App, ohne die **AV** auf der Ausgabe weiterzuleiten als etwas wie "**AV at address xyz read of address 000**".
Sie wissen vielleicht, dass der **Application.OnException-Handler** nur bei unbehandelten Exceptions aufgerufen wird. Eine unbehandelte Exception tritt auf, wenn kein **try..except-Block** die Exception gefunden hat oder wenn sie abgefangen und dann erneut ausgelöst wurde!
Die Ausführung wird in jedem äußeren Ausnahmeblock fortgesetzt. Wenn es keinen gibt oder wenn ein eventuell vorhandener Block die Exception erneut auslöst, wird die Exception schließlich den **Application.OnException-Handler** erreichen.
Tatsächlich ist `Application.OnException` Ihre letzte Chance, mit einer unbehandelten Exception umzugehen. Es ist nicht die erste Gelegenheit, auf eine Exception zu reagieren.
Das Abfangen der Exception und die Anzeige der Meldung ist nur dann hilfreich, wenn Sie die Software bereits für die Benutzer freigegeben haben und das Problem nicht lokal reproduzieren können (*und selbst dann ist es nicht so gut wie die Verwendung einer echten* **Exception-Bibliothek, wie EurekaLog oder MadExcept**).
In diesem Fall wissen wir bereits, welche Zeile die Exception verursacht hat, welche Exception-Klasse ausgelöst wurde und welche Meldung sie enthielt.
Der vorgeschlagene Code liefert **KEINE** zusätzlichen Informationen für die Untersuchung.
Wenn Sie also einen Netzwerkfehler oder eine Ausnahme in Ihrem Webaufruf von einem Rest-Client haben, könnte es sinnvoll sein, die nicht funktionierende Verbindung zu überprüfen, bevor Sie die mögliche Exception abfangen, wie im nächsten Bild zu sehen.
Hinweis zu Fehler 12007: Ich versuche, die HTML-Quelle hinter einer Onion-Site über die WinINet API abzurufen, aber die Funktion InternetOpenUrl() gibt den Fehlercode 12007 zurück, was darauf schließen lässt, dass es ein Problem bezüglich der Auflösung der Webadresse gibt.

Das Ungewöhnliche an diesem Problem ist, dass der Fehler nicht reproduzierbar ist, wenn für den Weblink eine Nicht-Onion-Site * verwendet wird, was eindeutig bedeutet, dass es kein Problem in dem Teil gibt, der eine mögliche Proxy-Konfiguration betrifft.
Auch die Funktion GetLastError gibt einen Fehlercode wie 12007 zurück. Sie können auch mit dem Tool netmon überprüfen, ob tatsächlich ein Verbindungsversuch stattgefunden hat.

.onion ist ein spezieller Top-Level-Domainname, der einen anonymen Onion-Dienst bezeich
der früher als "versteckter Dienst" bezeichnet wurde und über das Tor-Netzwerk erreichbar ist
Solche Adressen sind keine eigentlichen DNS-Namen, und die .onion TopLevelDomain befindet
sich nicht im Internet-DNS-Root

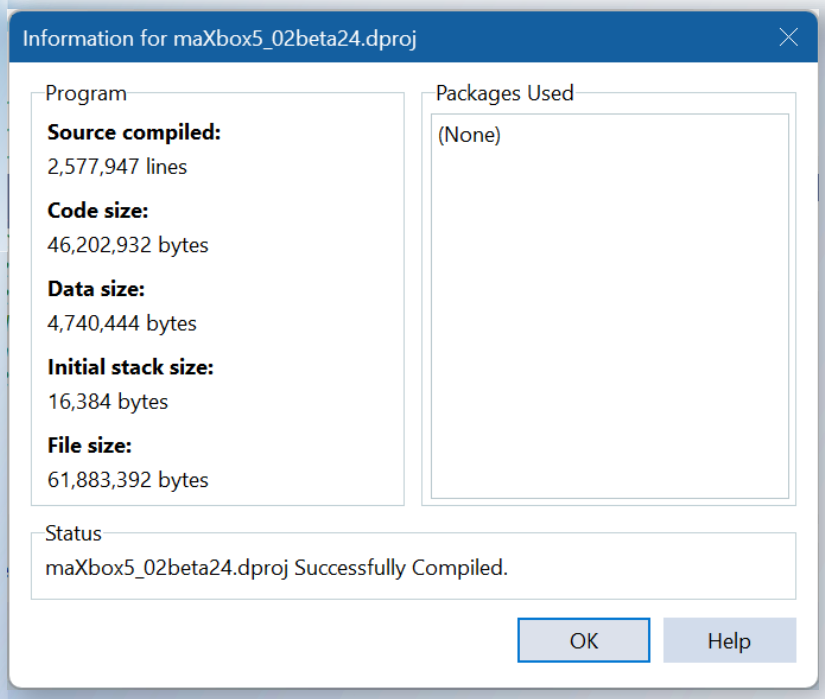
***Das erklärt den Namen: non onion site**



Wie wir gesehen haben, können Sie den Debugger anweisen, bestimmte Arten von Exceptions zu ignorieren.
Fügen Sie der Liste eine Ausnahmeklasse hinzu, und alle Exceptions dieses Typs und aller davon abhängigen Typen werden Ihr Programm durchlaufen, ohne dass Delphi eingreift.
Sie können die "erweiterten Breakpoints" von Delphi verwenden, um die Exception-Behandlung in einem bestimmten Bereich des Codes zu deaktivieren. Setzen Sie zunächst einen BreakPoint in der Codezeile, in der die IDE Exceptions ignorieren soll oder in der der Aufruf einer API von außen erfolgt, wie im folgenden 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
```

SCHLUSSFOLGERUNG:
Breakpoints unterbrechen die Programmausführung an einer bestimmten Stelle oder wenn eine bestimmte Bedingung eintritt. Sie können vor und während einer **Debugging-Sitzung** im Code-Editor Breakpoints für den Quellcode und das Laden von Modulen setzen.
`Application.OnException` ist Ihre letzte Chance, mit einer unbehandelten Exception umzugehen. Grundsätzlich werden Exceptions zuerst an den Debugger geworfen, dann an das eigentliche Programm, wo sie, wenn sie nicht behandelt werden, zweimal wiederholt werden, um Ihnen die Möglichkeit zu geben, in Ihrer IDE etwas damit zu tun,
nämlich vor und nach der eigentlichen Anwendung.



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

ÜBERBLICK ÜBER DIE PASCALSCRIPT-FUNKTION IN SYNCOVERY

VON DMITRII V. KONNOV, TOBIAS GIESEN

ABSTRACT:
Dieser Artikel gibt einen Überblick über die neue Funktion, die als **PascalScript** bezeichnet wird und in der weltweit führenden Cloud-Dateisynchronisationsanwendung **Syncovery** implementiert ist.
Schlüsselwörter:
Syncovery 10, Cloud-Speicher, Google Drive, Sharepoint, OneDrive, DropBox, PascalScript

EINFÜHRUNG

PascalScript ist eine Skriptsprache, die auf der Programmiersprache **Pascal** aufbaut.
Sie ermöglicht eine automatische Laufzeitkontrolle für skriptfähige Anwendungen und Serversoftware.
Diese Skript-Engine besteht aus einem Compiler und einem Bytecode-Interpreter und bietet eine freie und quelloffene Lösung.

PascalScript wurde so entwickelt, dass es weitgehend mit **Object Pascal** kompatibel ist, was es teilweise mit **Delphi, Free Pascal** und **GNU Pascal** kompatibel macht.
Ursprünglich von **Carlo Kok** unter dem Namen **CajScript** entwickelt, wurde es später mit Version 2.23 in **Innerfuse Pascal Script** umbenannt.

Später übernahm **RemObjects** das Projekt und gab ihm den Namen **RemObjects Pascal Script.**
Es wurde dann als Open-Source-Software zur Integration in die integrierte Entwicklungsumgebung (*IDE*) von Delphi zur Verfügung gestellt. **Ab der Version 2.07 wurde Pascal Script auf Free Pascal portiert.**
Seit 2017 ist **PascalScript** als **Standardkomponente in die Lazarus IDE integriert**, was seine Reichweite und seinen Nutzen weiter erhöht.

SYNCOVERY

Zu den Programmen, die **PascalScript** in großem Umfang nutzen, gehört **Syncovery**, ein bekanntes Programm für Backups und Cloud-Dateisynchronisation.
Aufgrund der wachsenden Anzahl von Cloud-Speichern von Anbietern der globalen Cloud-IT-Branche, wie **Amazon S3, Google Drive, Microsoft Azure, OneDrive, SharePoint, DropBox, Box** und **Backblaze B2,** wird die Integration von Cloud-Datenspeichern zu einer der beliebtesten Aufgaben eines modernen Benutzers.

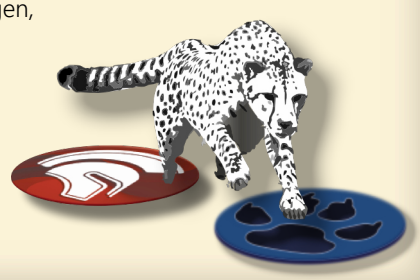
Jeder Cloud-Speicher hat seine Eigenheiten, seine Benutzerschnittstelle und seine **API**.
Aus der Sicht des Benutzers ist das Ideal eine Dateisynchronisationssoftware, die die Funktion des Backups von Dateien übernimmt und als völlig eigenständiges Modul im Hintergrund lokale Inhalte mit der Cloud synchronisiert.

In diesem Fall verschwimmt die Unterscheidung zwischen lokaler Festplatte und Cloud.
Das Lokale fließt nahtlos in die Cloud und umgekehrt.

Der Benutzer hat die Möglichkeit, sich nicht um die Sicherheit der Daten zu kümmern, sondern alle Routinen an automatisierte Prozesse und Geräte zu übertragen, die mit den Daten umgehen.
PascalScript wird in **Syncovery** aufgerufen, um spezifische Benutzeranforderungen von Dateisynchronisationsaufgaben anzupassen.

Die Anwendung bietet einen Desktop-GUI unter **Windows** und **macOS**, während die **Linux-Edition** eine **webbasierte Benutzeroberfläche** enthält, auf die über einen Browser zugegriffen wird.

Syncovery basiert in erster Linie auf dem einfachen Verständnis des Konzepts der Datensynchronisierung. Zu diesem Zweck wurde vorgeschlagen, dem Benutzer eine intuitive grafische Oberfläche in Form einer Links-Rechts-Kombination zur Verfügung zu stellen, wie in Abbildung 1 dargestellt.
Siehe nächste Seite.



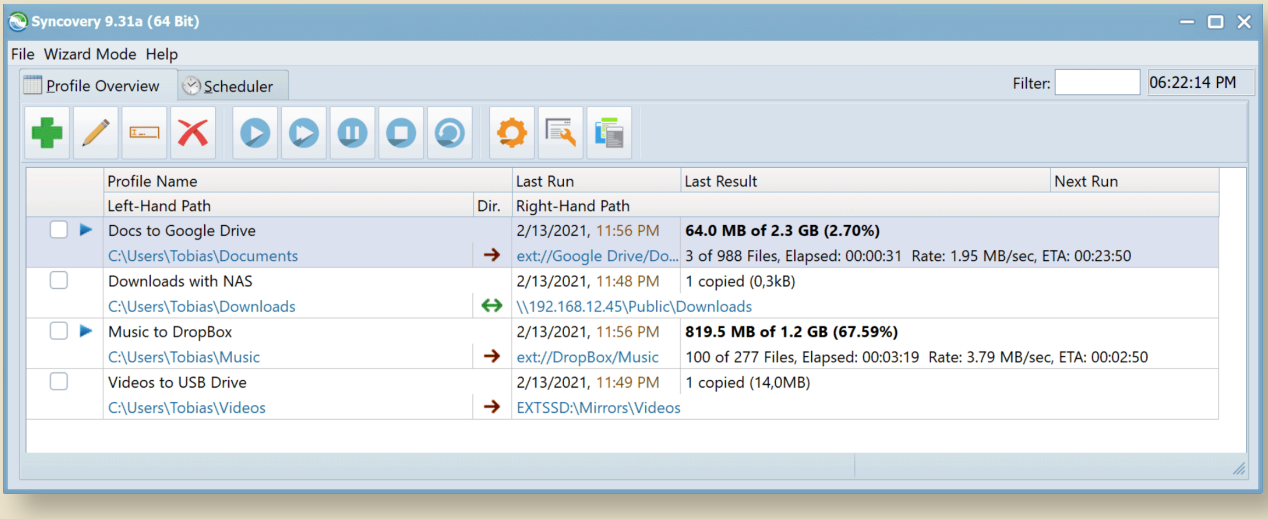


Abbildung 1. Syncovery Benutzeroberfläche

Der "Linke Pfad" und der "Rechte Pfad" - sind die Speicherorte der Dateien, die entweder ein lokales Verzeichnis oder ein Cloud-Verzeichnis sein können. Der Benutzer konfiguriert die Synchronisierungspunkte (*Ordner*) seiner Dateien, und der integrierte Scheduler startet die Synchronisierungsaufgaben.
Die Flexibilität bei den Synchronisierungsaufgaben kann nicht nur durch verschiedene Einstellungen in der **Syncovery GUI**, sondern auch mit **PascalScript** erreicht werden.

PASCALSCRIPT
Syncovery enthält eine **PascalScript**-Engine, mit der Sie das Verhalten Ihres Profils in vielerlei Hinsicht anpassen können. **PascalScript** umfasst die folgenden Funktionen:

- `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 innerhalb des Skripts**

Es gibt viele Standardfunktionen wie `Pos, Copy, Length, Ord, GetProfileProperty, SetProfileProperty, SaveProfileSettings, ConcatPath, ExtractFileName, ExtractFilePath, ExtractURLPartAfterServer, ExtractFileExt, ChangeFileExt, FileExists, FileExistsMatching, EntryExists, FileAge, FileCopy, FileDelete, FileRename,`
und viele andere, mit denen Sie sich vertraut machen können unter
`https://www.syncovery.com/pascalscript/`

PascalScript wurde in **Syncovery 8** hinzugefügt, und seit der ersten Version sind viele neue Hooks und Funktionen hinzugekommen. Sehen wir uns an, wie Sie mit **PascalScript** eigene Programmierungen vornehmen können. Doppelklicken Sie auf ein Profil, um den Dialog Profileinstellungen zu öffnen (*siehe Abbildung. 2*).

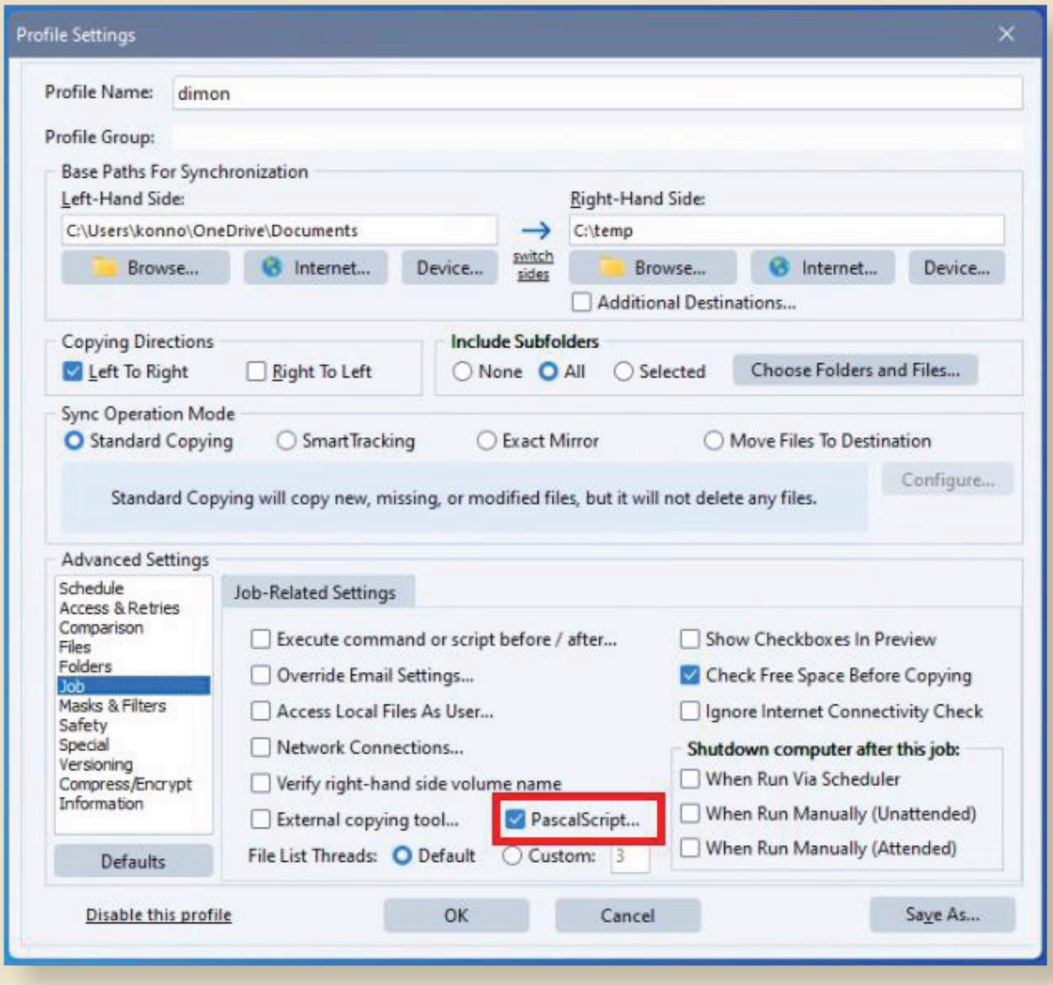


Abbildung 2. Dialogfeld Profileinstellungen

Wählen Sie dann in der Liste **Advanced Settings** den Eintrag 'Job' und doppelklicken Sie auf das Kontrollkästchen '**PascalScript**', um den PascalScript-Editor aufzurufen. Um ein bestimmtes Verhalten zu ändern, müssen Sie eine Hook-Funktion schreiben und etwas Code eingeben.

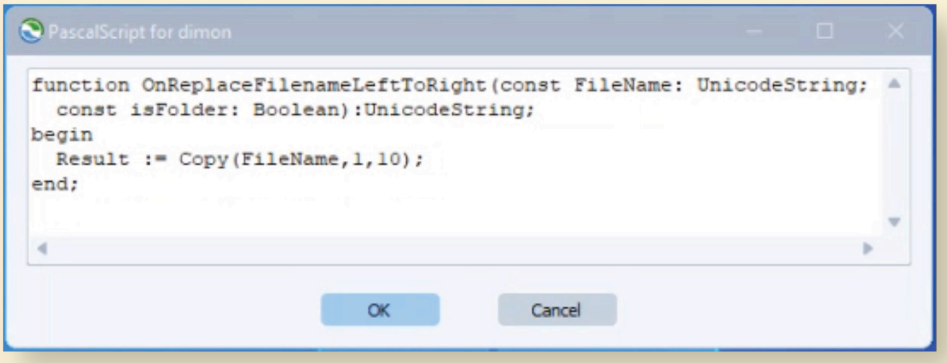


Abbildung 3. PascalScript-Editor

In diesem Fenster können Sie ein Skript bearbeiten, das mit dem Profil der Syncovery verknüpft ist. Jeder Funktionshandler, der im Editor deklariert wird, wird automatisch mit dem Ereignis von Syncovery verknüpft. In dem vorliegenden Skript (siehe Abb.3 Seite 4 dieses Artikels) sehen wir die Funktion OnReplaceFilenameLeftToRight, die einen neuen Dateinamen zurückgibt. Wir können zum Beispiel den Dateinamen kürzen, wenn die Länge des Dateinamens mehr als 10 beträgt. .

Dies ist jedoch nur ein einfaches Beispiel für die Fähigkeiten von **PascalScript.** Die Aufgabe eines jeden Kunden kann viel mehr Geschäftslogik beinhalten.

In den meisten Fällen wird unser technischer Support den Code für Sie schreiben.
Die Leistungsfähigkeit von **PascalScript** in **Syncovery** wurde regelmäßig erweitert, indem der Sprache neue **Hooks** und Supportfunktionen hinzugefügt wurden.
Einige typische Anwendungsfälle werden hier beschrieben.

- **Benutzerdefinierte Dateiumbenennungen:**

Der vielleicht am häufigsten verwendete **PascalScript-Hook** in **Syncovery** ist `OnReplaceFilenameLeftToRight`. Er ermöglicht die Angabe oder Ableitung eines neuen Dateinamens, entweder völlig frei oder basierend auf dem ursprünglichen Dateinamen.
Es sind Beispielskripte verfügbar, um lange Dateinamen zu kürzen oder nicht erlaubte Zeichen zu konvertieren.

- **Benutzerdefinierte Filter**:

Mit dem **OnIncludeItem-Haken** kann ein Kunde Code verwenden, um ungewöhnliche Filter zu implementieren. Beispielsweise kann ein **Skript** verwendet werden, um bestimmte Benennungsregeln durchzusetzen und alle Dateien zu verwerfen, die nicht übereinstimmen.
Es gibt Beispielskripte zum Ausschluss von Dateien ohne **Dateinamenserweiterung** sowie zur Verarbeitung nur von Unterordnern, die die Datei READY.to process enthalten.

- **Sortieren von Dateien in Unterordnern:**

Ein weiterer häufiger Anwendungsfall für ein **PascalScript** ist die Notwendigkeit, Dateien in zusätzliche Unterordner zu sortieren, die auf der Quellseite nicht vorhanden sind.
Mit dem **OnBeforeFileCopy-Hook** können Sie den Zielpfad auf der Grundlage von **Pascal-Code** ändern. Ein Beispielskript von der Syncovery Website manipuliert die Zielpfade für das Kopieren (*von links nach rechts*), indem es den Unterordner 'archive' hinzufügt, der auf der Quellseite nicht vorhanden ist. Andere Kunden benötigen eine Sortierung der Dateien in Unterordner auf der Grundlage des Datums, was ebenfalls möglich ist.

- **Benutzerdefinierte Zeitpläne:**

Der **OnGetNextRunTime-Haken** kann verwendet werden, um benutzerdefinierte Regeln für die Zeitplanung eines Auftrags zu erstellen.
Am besten verwenden Sie ihn, wenn Sie dem Profil einen regelmäßigen, einfachen Zeitplan geben und den Hook verwenden, um unerwünschte Laufzeiten zu überspringen.
Ein Beispiel aus der PascalScript Dokumentation ist für ein Profil gedacht, das "jeden Tag um XX: YY" ausgeführt werden soll. Der Haken sorgt dafür, dass es nur am ersten Wochentag eines Monats ausgeführt wird.

Die Dokumentation für **PascalScript** in Syncovery sowie viele Beispielskripte finden Sie auf dieser Seite:
`https://www.syncovery.com/pascalscript/`

SCHLUSSFOLGERUNG

Bald wird Syncovery 20 Jahre alt. Im Laufe der Jahre hat sich Syncovery *2 zu einem unverzichtbaren Werkzeug für Systemadministratoren entwickelt.
Die erste Version der Anwendung (geschrieben in Delphi) wurde unter dem Namen Super Flexible File Synchronizer im Jahr 2003 veröffentlicht.
Das Projekt ist ein deutsches STEM-Startup, das sich schließlich zu einem funktionierenden Produkt entwickelte. Im Jahr 2012 wurde das Produkt in Syncovery umbenannt.
Im Jahr 2021, bei der Delphi Showcase Challenge, die dem 26. Jahrestag der Programmiersprache Delphi gewidmet war, wurde das Syncovery-Tool mit dem vierten Preis für eine hervorragende Desktop-Synchronisationssoftware ausgezeichnet, die die Flexibilität der Delphi-Entwicklungsumgebung demonstriert *3. Obwohl der Delphi-Compiler die Kompilierung für Windows, Linux und Mac unterstützt, wird er nur zur Kompilierung von Syncovery für Windows verwendet.
Binärdateien für weitere Plattformen werden mit dem Open-Source-Compiler Free Pascal erstellt.
In den 20 Jahren seiner Entwicklung hat Syncovery eine unübertroffene Benutzertreue und eine stabile und nachhaltige Produktentwicklung gezeigt. Das Programm wird ständig um neue Funktionen erweitert, die den Wünschen der Benutzer entsprechen. Wir freuen uns auf die weitere Zusammenarbeit mit den Benutzern von Syncovery.

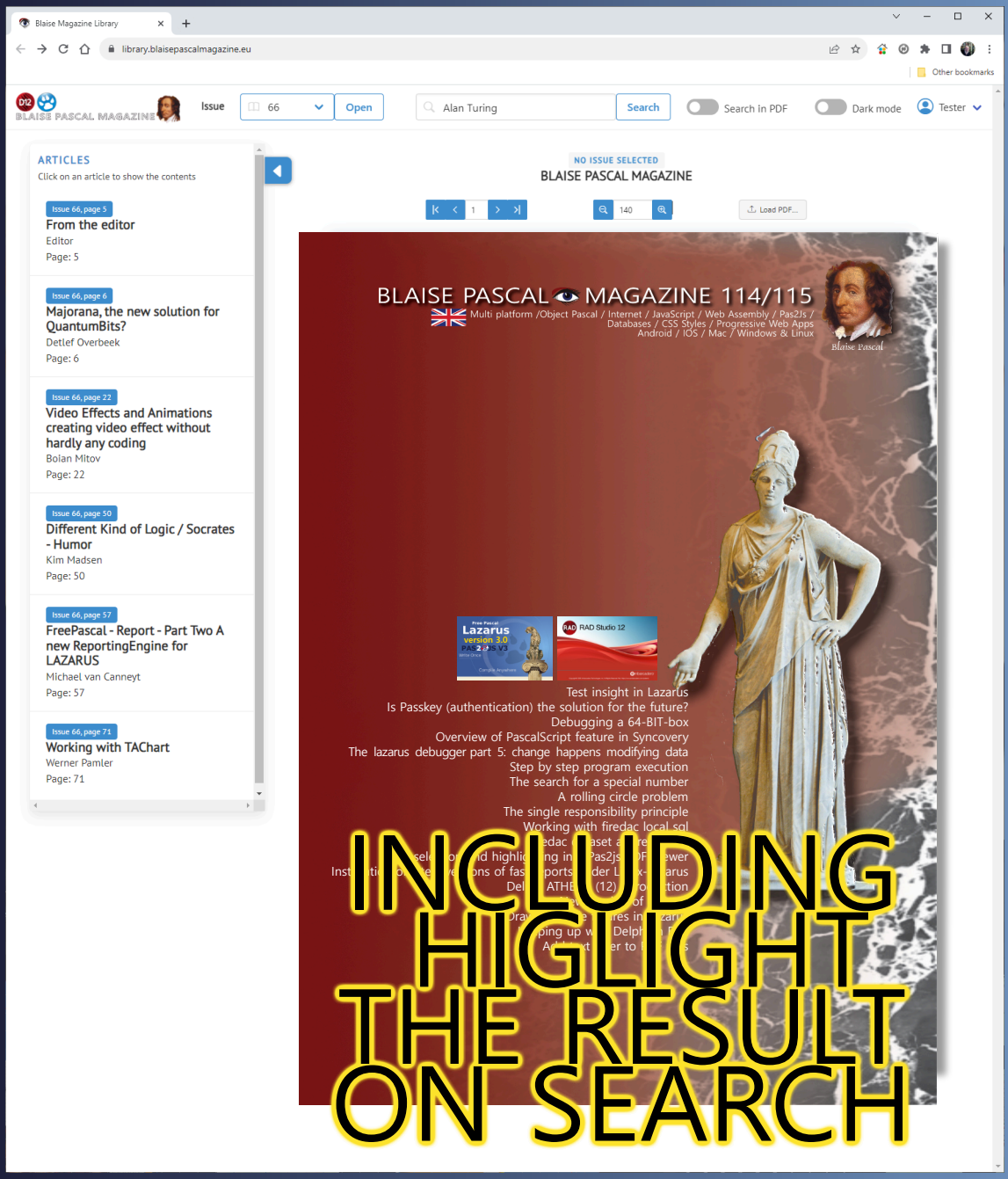
REFERENZEN

***1 RemObjects PascalScript: https://github.com/remobjects/pascalscript**
***2 Superflexible AG https://www.syncovery.com**
***3 Erstaunliche Desktop-Synchronisations-Software zeigt Delphi-Flexibilität** // Embarcadero
`https://blogs.embarcadero.com/astounding-desktop-synchronization-software-displays-delphi-flexibility/`

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

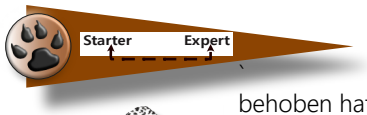


USE WHERE EVER THE INTERNET IS

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

DER LAZARUS DEBUGGER
TEIL 5: ÄNDERUNGEN PASSIEREN
DATEN ÄNDERN

VON MARTIN FRIEBE

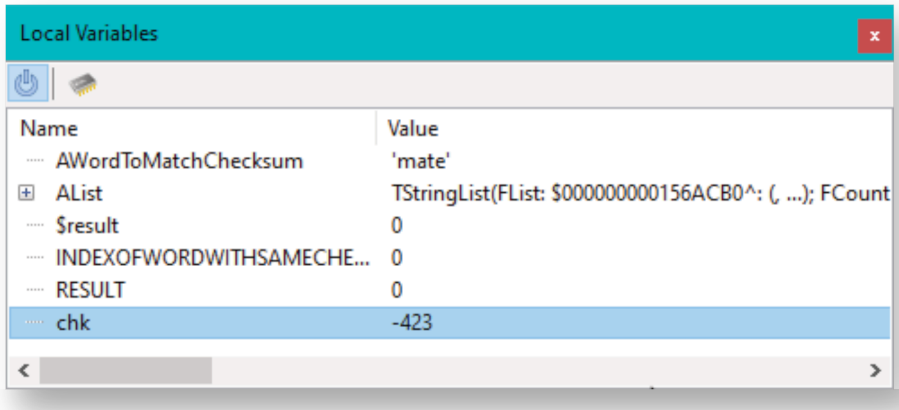


WAS IST UND WIE ES SEIN SOLLTE...

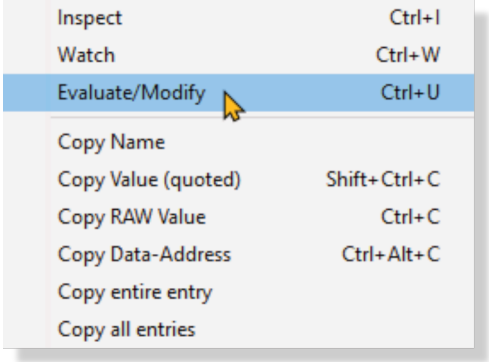
In allen vorherigen Artikeln haben wir immer, wenn wir den Fehler gefunden und behoben hatten, die Anwendung neu kompiliert und neu gestartet, um zu sehen, ob sie funktioniert. In unseren Beispielanwendungen war das kein Problem. Die App führte den von uns korrigierten Code aus und wir konnten sofort sehen, ob der neue Code funktionierte. Was aber, wenn die App lange brauchte, um den von uns korrigierten Code zu erreichen?
Was wäre, wenn wir lange mit der App interagieren müssten, bevor wir wüssten, ob die Korrektur etwas taugt?
Was wäre, wenn wir uns nicht einmal sicher wären, ob die Korrektur funktioniert, sondern nur testen wollten, wie sie funktionieren würde, wenn sie einen anderen Wert für eine der Variablen erhalten hätte?
Dann kann es sehr lästig sein, das Programm neu zu starten und darauf zu warten, dass der Code erneut ausgeführt wird. Je nachdem, um welches Problem es sich handelt, können wir die Anwendung weiterlaufen lassen und trotzdem testen, wie es wäre, wenn die Dinge anders wären. Dies funktioniert nicht für alle Arten von Änderungen. Es ist auf Änderungen an den Daten der App beschränkt. Wir werden dies anhand eines Beispiels untersuchen. Unser Beispiel ist recht einfach. Wir werden einfach so tun, als ob es mühsam wäre, es erneut auszuführen.
Es sucht in einer Liste nach einem Wort, das die gleichen Buchstaben enthält wie ein bestimmtes Wort.

```
1. program project1;
2.
3. uses Classes;
4.
5. function Checksum(s: string): integer;
6. var
7.   i: Integer;
8. begin
9.    Result := 0;
10.    for i := 1 to Length(s) do
11.     Result := Result + ord(s[i]);
12. end;
13.
14. function IndexOfWordWithSameChecksum(AWordToMatchChecksum: String;
                                          AList: TStringList): Integer;
15. var
16.   chk: Integer;
17. begin
18.   if AList = nil then
19.     exit(-1);
20.   chk := -Checksum(AWordToMatchChecksum);
21.   Result := AList.Count - 1;
22.   while Result >= 0 do begin
23.     if Checksum(AList[Result]) = chk then
24.       exit;
25.     dec(Result);
26.   end;
27. end;
28.
29. var
30.    Words: TStringList;
31.    i: Integer;
32. begin
33.    Words := TStringList.Create;
34.    Words.Add('words');
35.    Words.Add('tame');
36.    Words.Add('town');
37.
38.    i := IndexOfWordWithSameChecksum('mate', Words);
39.    writeln(i);
40. end.
```

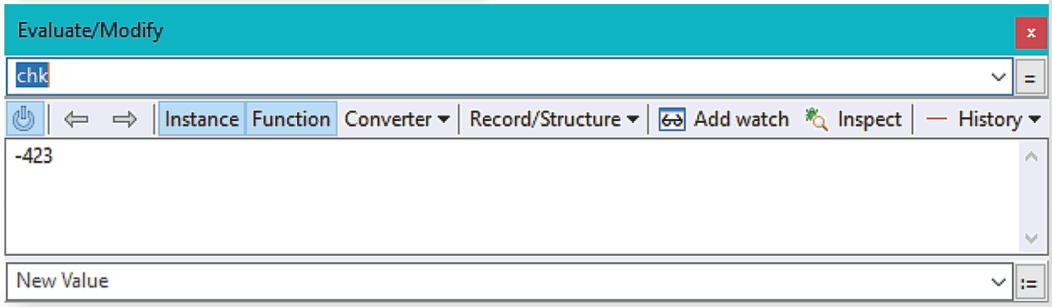
Da die Prüfsumme die Summe der Ordinalwerte der einzelnen Buchstaben ist, sind Wörter mit gleichen Buchstaben in beliebiger Reihenfolge Wörter mit der gleichen Prüfsumme. In unserem Beispiel erwarten wir, dass "mate" mit "tame" übereinstimmt.
Es sollte also "1" ausgegeben werden.
Er gibt jedoch "-1" aus, was bedeutet, dass er nichts gefunden hat. Wie üblich beginnen wir mit der Fehlersuche und brechen in Zeile 18 ab, der ersten Zeile in "IndexOfWordWithSameChecksum". Dort können wir das lokale Fenster verwenden, um einen Blick auf die beteiligten Variablen zu werfen.
Als nächstes gehen wir die Zuweisung der Prüfsumme an "chk" durch:



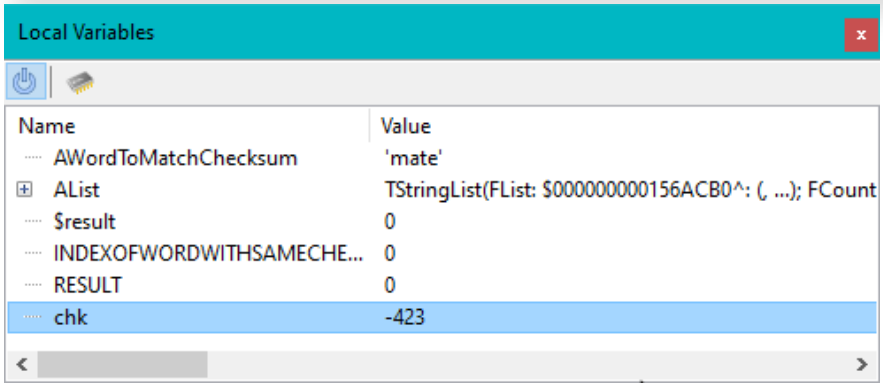
Und wir stellen fest, dass "chk" einen Wert von -423 hat.
Aber ein negativer Wert ist nicht das, was wir erwarten. Wir haben die Ordnungswerte aller Zeichen in "mate" summiert und sie sind alle positive Ganzzahlen. Bei näherer Betrachtung des Quellcodes stellen wir fest, dass sich ein Tippfehler in unseren Code eingeschlichen hat. Es gibt ein überflüssiges Minus vor dem Aufruf von "Checksum". Die tatsächliche Prüfsumme in "chk" sollte 423 lauten. Nun könnten wir das beheben, indem wir den Code ändern und die Anwendung neu starten. Oder wir könnten den Wert in der laufenden Anwendung ändern und die aktuelle Ausführung mit dem korrekten Wert fortsetzen. Aus dem Popup-Menü des Fensters locals wählen wir "Auswerten/Ändern" für die Variable "chk":



Dadurch wird das entsprechende Fenster geöffnet.

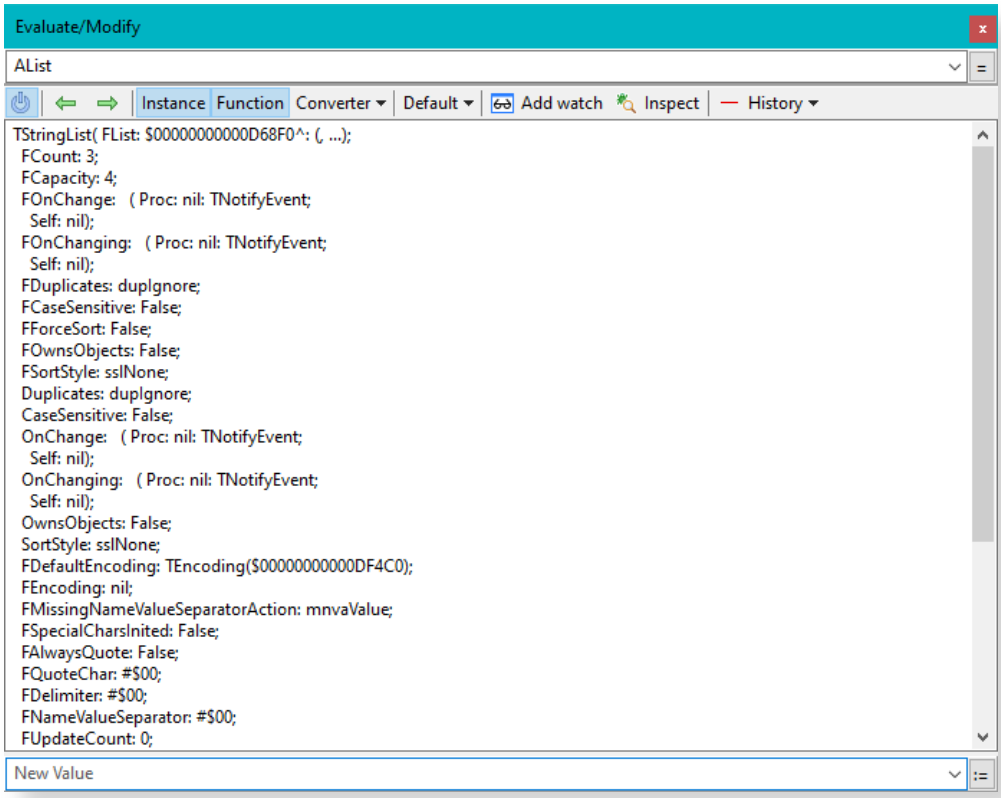


Dieses Fenster bietet noch eine weitere Möglichkeit, Werte zu prüfen und anzuzeigen. Doch zunächst wollen wir uns auf die Funktion "Ändern" konzentrieren. Die Variable, die wir ändern möchten, ist bereits ausgefüllt und ihr Wert wird angezeigt.
Unten befindet sich ein Feld zur Eingabe eines "Neuen Wertes" und ein Button ":=", um ihn zu übernehmen.
Wir geben den positiven Wert 423 in das Eingabefeld ein und drücken den Button. Sofort können wir sehen, dass der Wert aktualisiert wird. Sowohl das Fenster "Auswerten/Ändern" als auch das Fenster "Einheimische" zeigen jetzt 423 für die Variable "chk" an.
Damit fahren wir mit unserer Anwendung bis zum Ende von "IndexOfWordWithSameChecksum" fort, wo wir den Wert von "Result" im Fenster "Locals" überprüfen.

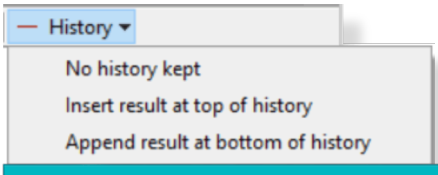


Und dort können wir bestätigen, dass das Ergebnis dieses Mal den korrekten Wert "1" hat, der auch ausgegeben wird, sobald wir die App zu Ende laufen lassen.
Das Ändern des Wertes, ohne die Anwendung neu starten zu müssen, hat uns etwas Zeit erspart. Sobald der Fehler jedoch bestätigt ist und wir wissen, wie wir den Code ändern müssen, müssen wir trotzdem neu kompilieren.
Andernfalls wird der Wert bei jeder Eingabe der Funktion falsch sein und wir müssten den Wert immer wieder ändern. Wir könnten einen BreakPoint setzen und das ein paar Mal tun, aber schließlich würden wir durch die manuelle Änderung der Variable den Vorteil gegenüber der Neukompilierung und dem Neustart verlieren.
Andererseits hätten wir, wenn die Änderung das Problem nicht behoben hätte, bei einem erneuten Aufruf der Funktion einen weiteren Versuch gehabt, das Problem zu untersuchen.
Letztendlich ist jeder Fall anders und die Vorteile müssen bei jeder Debug-Sitzung geprüft werden.

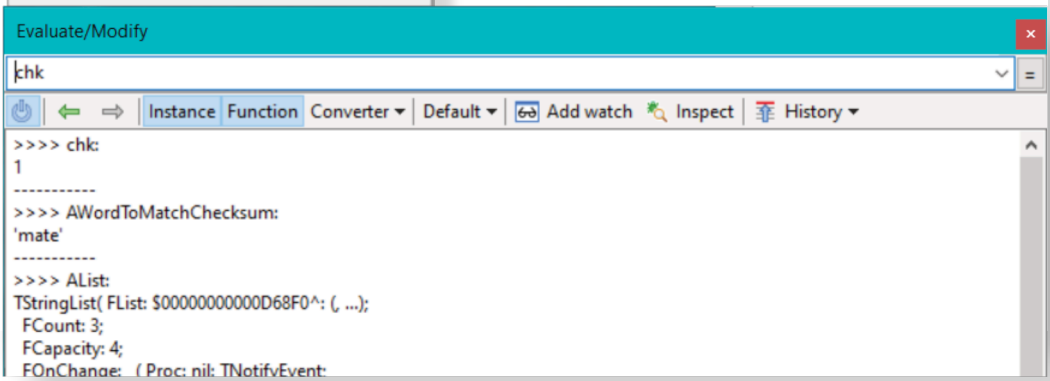
DER TEIL "AUSWERTUNG"
Der vollständige Name des Fensters lautet "Auswerten/Ändern". Wir haben uns den "Ändern"-Teil angesehen und werden nun den "Auswerten"-Teil untersuchen.
Kurz gesagt, das Fenster "Auswerten" funktioniert wie ein Fenster für eine einzelne Überwachung und zeigt das Ergebnis auf dieselbe Weise an wie das Fenster "Überwachung" im "Detailbereich".
Der Hauptvorteil besteht darin, dass alle Einstellungen aus dem Dialogfeld "Eigenschaften der Überwachungen" direkt zugänglich sind, so dass es viel einfacher ist, sie oder den zu bewertenden Ausdruck zu ändern.
Und da fast das gesamte Fenster für die Anzeige des Ergebnisses verwendet wird, ist es viel leichter zu lesen als im "Detailfenster". Dies ist besonders nützlich für Strukturen, kann aber auch bei memory dumps, arrays or long strings helfen.



Wenn Sie "AList" auswerten, sehen Sie Folgendes.
Alle Felder sind gut sichtbar, und selbst verschachtelte Werte wie in Ereignissen (*z.B.* `"OnChange"`) sind unterstrichen. Die Symbolleiste bietet dieselben Eigenschaften wie der Dialog "Eigenschaften von Überwachungen.", der im letzten Artikel erläutert wurde. Sie können zwar immer nur eine Variable auf einmal sehen, aber mit den grünen Pfeil-Buttons können Sie zwischen den ausgewerteten Werten hin- und hergehen. Sie können auch frühere Werte aus der Dropdown-Liste auswählen. Und wenn Sie möchten, können Sie frühere Werte sichtbar lassen und neue Ergebnisse über oder unter ihnen einfügen. Der Button "Historie" bietet Ihnen die folgenden Optionen: Geben Sie die folgende Ansicht an:

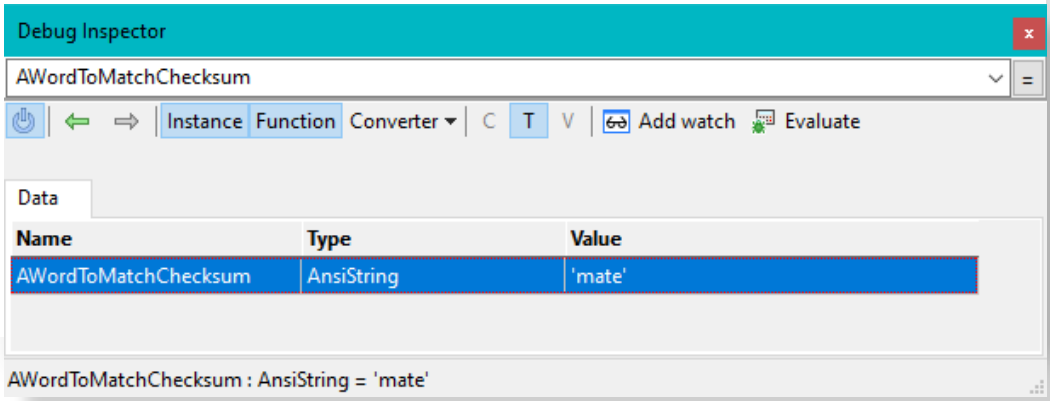


Sie geben die folgende Stellungnahme ab:

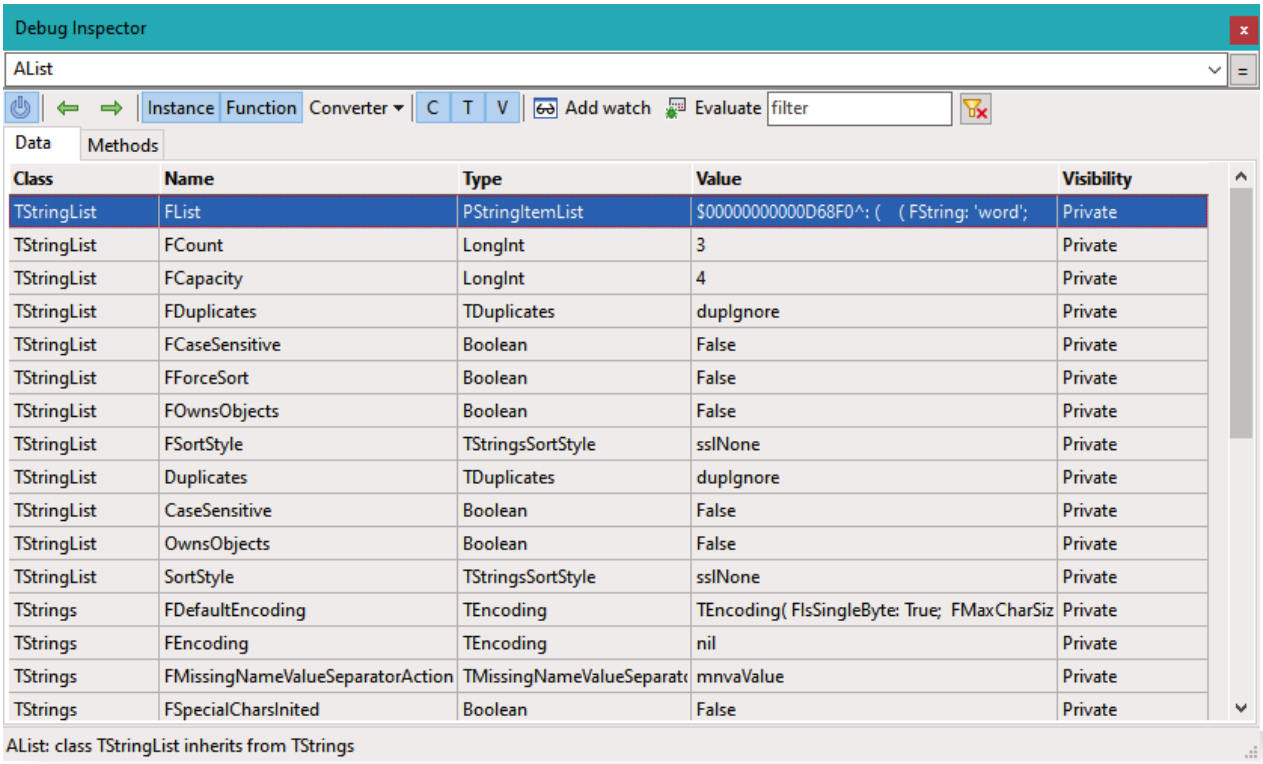


DER DEBUG-INSPEKTOR

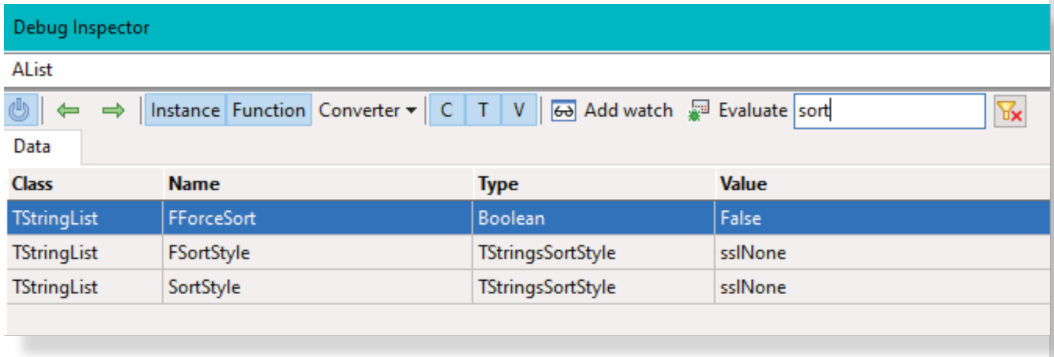
Das Thema dieses Artikels ist zwar das Fenster Auswerten/Ändern, aber wir werden es ein wenig erweitern, um ein weiteres Fenster zur Anzeige von Daten aus der Anwendung zu betrachten: Der "Debug-Inspektor". Ähnlich wie das Auswerten-Fenster ermöglicht es die Betrachtung jeweils einer Überwachung. Für einfache Werte bietet es nicht viel mehr als die anderen Möglichkeiten zur Anzeige von Überwachung, außer dass es den Typ-Namen der Daten mit einbezieht.



Bei strukturierten Typen und Arrays spielt es seine ganze Stärke aus.
Es listet jedes Feld oder jeden Eintrag als separate Zeile auf..

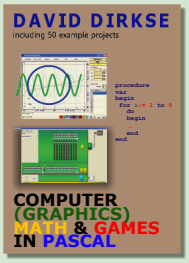


Sie können auf Zeilen doppelklicken, um den Wert eines Feldes oder eines Eintrags für Arrays zu prüfen. Dies funktioniert auch bei der Dereferenzierung von Zeigerwerten. Zusammen mit der Vorwärts-/Rückwärtsnavigation lassen sich die Daten leicht erkunden, und jeder inspizierte Wert, der von dauerhaftem Interesse ist, kann mit einem Klick auf einen Button als Überwachung hinzugefügt werden.
Darüber hinaus bietet der Inspektor einen Filter zum Durchsuchen der Daten.
Dieser Filter ist nur für strukturierte Werte und Arrays verfügbar.
Jeder eingegebene Wert wird mit allen sichtbaren Spalten abgeglichen.
Sie können also nach Feldern suchen, die einen Text in den Daten enthalten.
Sie können aber auch nach einem Feld anhand seines Namens suchen.
Wenn wir in "AList" sortierbezogene Felder finden möchten, können wir "sort" in den Filter eingeben:



Wenn es der "`FSortStyle`" ist, den wir suchen, können wir dann mit Cursor-nach-unten zu ihm navigieren, und Strg-Enter drücken, um es auszuwählen und den geprüften Ausdruck in `Alist.FSortStyle` zu ändern. Dies ermöglicht eine sehr schnelle Navigation auch in Objekten mit vielen Feldern. In Arrays kann dies auch mit dem Index übereinstimmen, allerdings nur innerhalb der Liste der Indizes auf der aktuellen Seite (*Arrays werden wie im Überwachungsfenster aufgelistet*)

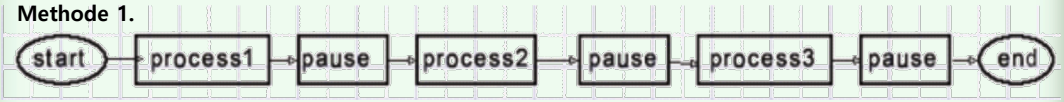
VON DAVID DIRKSE



ABSTRACT

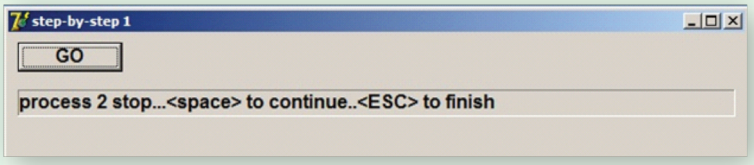
Bei Lernsoftware ist es aufschlussreich, die einzelnen Schritte zu einer Lösung zu zeigen. Dieser Artikel beschreibt zwei Möglichkeiten, dies zu erreichen. In beiden Fällen wird ein Prozess aus process1, process2 und process3 gebildet, der Schritt für Schritt ausgeführt werden kann.

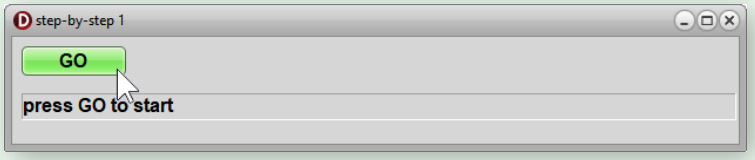
Methode 1.

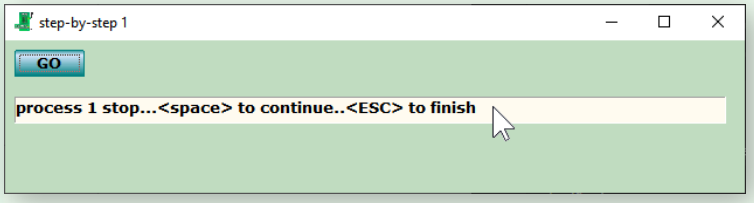


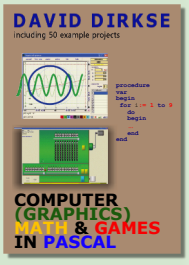
Die Pause-Prozedur setzt ein Stopflag (`Boolean`) und ruft `Application.ProcessMessages` auf, bis das Stopflag gelöscht wird. Diese Löschung erfolgt durch Drücken der Leertaste.
Wenn der Schritt-für-Schritt-Betrieb nicht mehr erwünscht ist, können Sie durch Drücken der Escape-Taste den gesamten Prozess ohne Pausen beenden. Dies wird durch das Abbruchflag (Boolean) gesteuert.
Die Pausenprozedur prüft das Abbruchflag und verlässt den Vorgang sofort, wenn es gesetzt ist. Das Drücken der Escape-Taste setzt das Abbruchflag und löscht das 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;
```

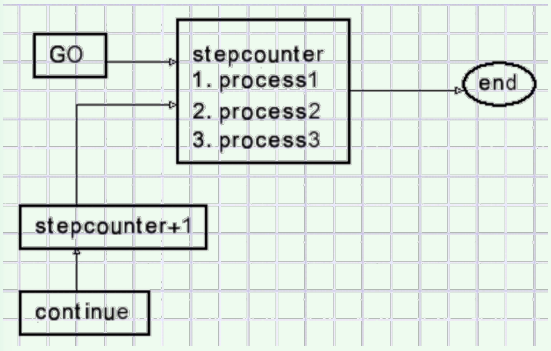




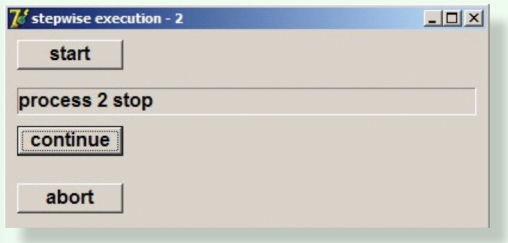




Methode 2.



Bei dieser Methode wird nicht die Pause-Prozedur verwendet, sondern Buttons, um schrittweise durch den Prozess zu gehen.
Der Schrittzähler (Byte) wählt die Prozesse aus.
Wenn Sie GO drücken, wird der Schrittzähler auf Null gesetzt.
Die Prozedur Nextproc erhöht den Schrittzähler auf 1 und ruft process1 auf.



```
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;
```

Starten des Prozesses.....

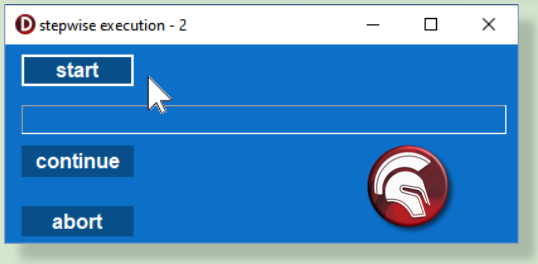
```
procedure TForm1.startBtnClick(Sender: TObject);
begin
  stepcount := 0;
  nextproc;
end;
```

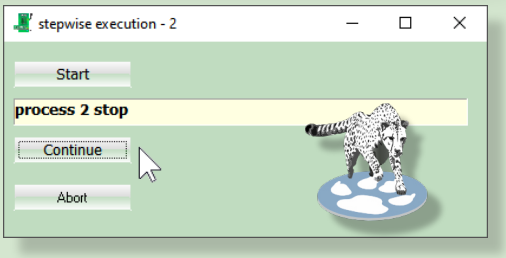
nächster Prozessschritt....

```
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;
```



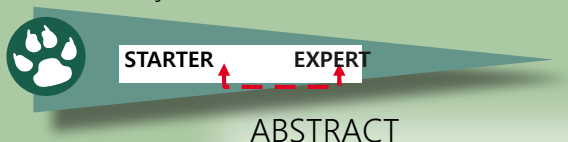


Damit ist die Beschreibung der schrittweisen Programmausführung abgeschlossen.

DIE SUCHE NACH EINER BESTIMMTEN ZAHL

Das Projekt wurde schon veröffentlicht, aber dieses mal als Lazarus-Projekt.

Lazarus 1-3



ABSTRACT

Ein bekanntes mathematisches Rätsel ist dieses:
Eine Zahl hat eine 2 als niedrigste Ziffer.
Wenn diese Ziffer nach links verschoben wird, ergibt sich eine Multiplikation mit 2.

Oder so:
Eine Zahl hat eine 2 als äußerste linke Ziffer.
Wenn diese Ziffer nach rechts verschoben wird, ergibt sich die Division durch 2.
Das Lustige an diesem Rätsel ist, dass es nur Grundschulrechnungen erfordert,
aber selbst Mathelehrer haben Probleme, es zu lösen.

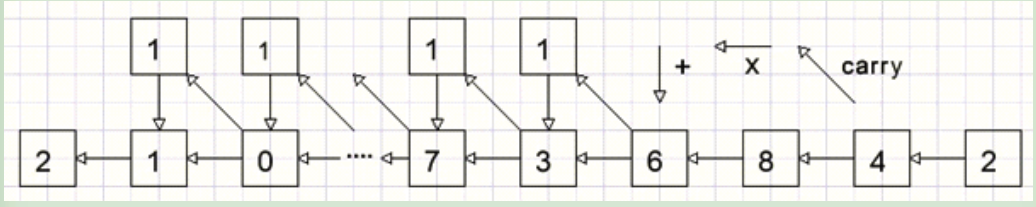
Nach einigen Versuchen mit der Hand stellt sich die Frage, ob eine solche Zahl überhaupt existiert.
Außerdem: Sind auch andere Ziffern als 2 möglich Und: sind andere Faktoren als 2 möglich?
Dieser Artikel beschreibt ein Delphi-Programm, das nach Zahlen mit diesen Eigenschaften sucht.

DER ALGORITHMUS.

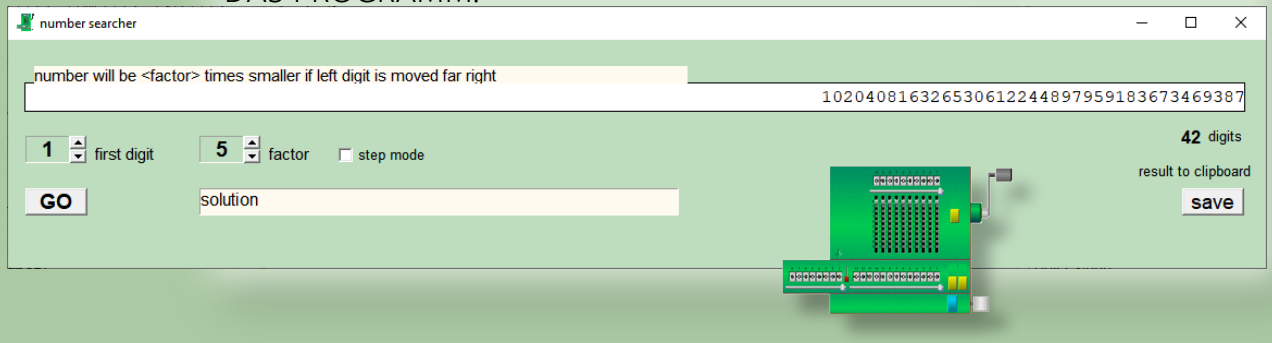
Zur Erklärung verwenden wir die Ziffer 2 als Startwert und einen Multiplikationsfaktor von 2.
HINWEIS: Ein Übertrag wird als (..) geschrieben.

```
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
```

Jetzt sind die erste und die letzte Ziffer gleich (*und es gibt keinen Übertrag*)
Lassen Sie die unterste "2"-Ziffer weg und wir haben eine Zahl gefunden,
die durch 2 teilbar ist, wenn die erste Ziffer nach rechts verschoben wird.



DAS PROGRAMM.



DIE SUCHE NACH EINER BESTIMMTEN ZAHL

Das Projekt wurde schon veröffentlicht, aber dieses mal als Lazarus-Projekt.

Die Abbildung auf der vorhergehenden Seite zeigt das Programm bei der Arbeit.
Mit den **UpDown**-Steuerelementen, die mit Static Text-Komponenten verbunden sind, wählen Sie die erste Ziffer und den Multiplikationsfaktor aus.
Der **GO Button** startet die Suche.
Der **SAVE Button** kopiert das Ergebnis in die Zwischenablage, so dass es in Texteditoren eingefügt werden kann. Wenn der **Schrittmodus** aktiviert ist, hält das Programm nach jeder Iteration an und zeigt Zwischenergebnisse an.
Die Zahl wird im `Byte-Array A[0...120]` gespeichert.
Die ganze Zahl N ist der Index des Arrays `A[ ]`.

Initialisierung:.

```
N := 0;
A[0] := Startziffer.
```

Schritt 1:

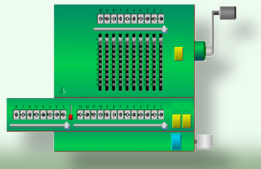
`A[N]` wird mit 2 multipliziert, das Ergebnis wird in `A[N+1]` (untere Ziffer) und `A[N+2]` (Übertrag) abgelegt.

Schritt 2:

```
N := N + 1;
```

Die Schritte 1 und 2 werden wiederholt, bis `N = 120` (keine Lösung) oder `A[N] = A[0]` und `A[N+1] = 0` (kein Übertrag).

```
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;
```



Schritt-Modus
Wenn `stepMode` ausgewählt ist, wird nach jeder Iteration die Variable `stopflag` auf `true` gesetzt.
gefolgt von :
`while stopFlag do Application.ProcessMessages;`
Durch Drücken der **<Leertaste>** wird das **Stopflag** zurückgesetzt, so dass der Prozess fortgesetzt wird.
Dazu muss die Eigenschaft `keyPreview` im form1 **Objektinspektor** auf true gesetzt werden.
Der Suchvorgang begann mit `Activecontrol := nil.`
Dadurch wird verhindert, dass das Zeichen `<space>` an den **GO Button** gesendet wird,
der nach dem Drücken den Fokus hat.

Anzeige des `Arrays A[ ]`
Ich verwende eine **Paintbox** für die Anzeige.
Der Grund dafür ist, dass dies die Anzeige von Ziffern in verschiedenen Farben ermöglicht.
Überträge werden in **Rot** angezeigt. Es wird die Schriftart **`Courier new`** verwendet,
deren Zeichen eine feste Breite haben.

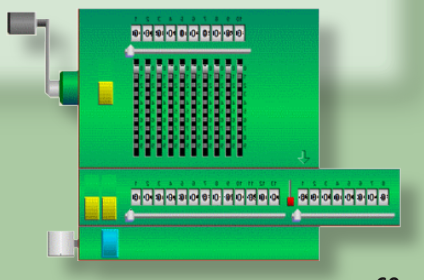
`A[1]` wird rechts platziert, die **Zeichenposition x** wird dekrementiert,
um die nächsten Zeichen links anzuzeigen.
`procedure displayA(n : byte);` erledigt die Aufgabe.
`n` ist die Anzahl der Ziffern.
Jedes Zeichen, das von einem höheren Index als `n` von `A[n]` stammt und nicht `Null` ist,
wird in **Rot** dargestellt.

Das Ergebnis wird in der Zwischenablage gespeichert.
Die `Unit Clipbrd` muss dazu der **uses-Klausel** hinzugefügt werden.
Da das Ergebnis in der Zeichenkette s gespeichert ist, erledigt diese Anweisung die Aufgabe:
`clipboard.As Text := s;`

Aber zuerst muss `A[ ]` nach `s` kopiert werden.
Die höchste Ziffer wird zuerst übertragen.
Beim Zähler `p = 3` wird `" . "` als Tausendertrenner eingefügt.

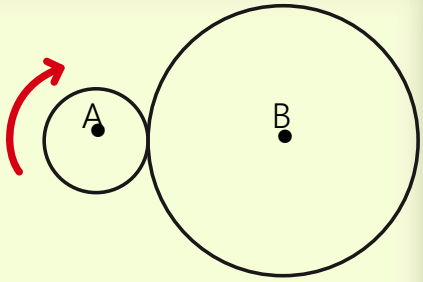
```
 var s : string;
   i,n,p : byte;
begin
 s := '';
 n := maxN;
 while (A[n] = 0) and (n > 0) do dec(n);//find first non zero digit
p := 0;
 for  i := n downto 1 do
  begin
    if p = 3 then begin
                    s := s + '.';
                    p := 0;
                  end;
   s := s + char(A[i]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

Weitere Einzelheiten entnehmen Sie bitte dem **Quellcode**.
Eine erstaunliche Eigenschaft der Antwort *(für den Faktor 11)* ist diese:
Teilen Sie die Zahl in zwei Hälften und addieren Sie dann.
Das Ergebnis zeigt nur **"9"** Ziffern: `99.....99`.



STARTER EXPERT 7 D11

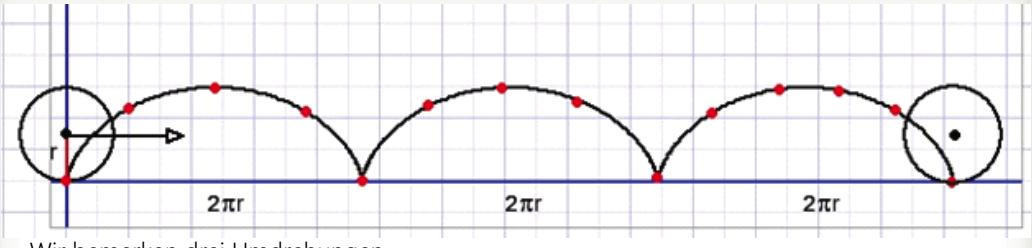
In der obigen Abbildung ist der Radius von Kreis A 1/3 des Radius von Kreis B.
Ausgehend von der in der Abbildung gezeigten Position rollt der Kreis A um den Kreis B.
Nach wie vielen Umdrehungen des Kreises A wird der Mittelpunkt des Kreises B seinen Ausgangspunkt erreichen?



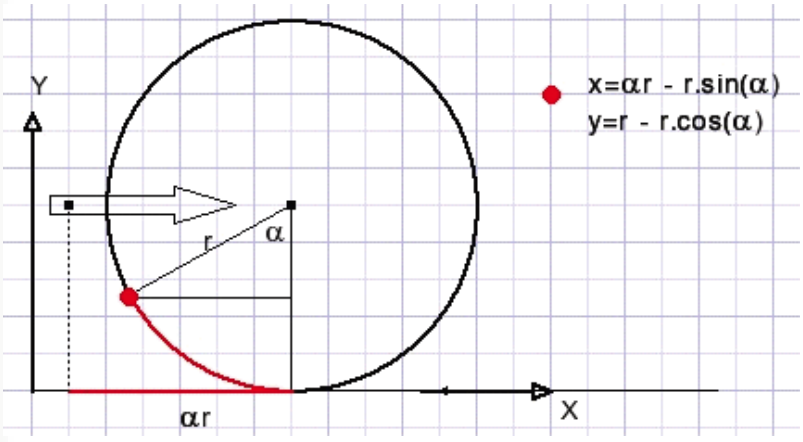
(A) $\frac{3}{2}$ (B) 3 (C) 6 (D) $\frac{9}{2}$ (E) 9

Ich schlage vor, dass der Leser, bevor er weiter liest, versucht, diese Frage selbst zu beantworten.

THEORIE
Wir können den Umfang des Kreises B dehnen und den Kreis A über diese gerade Linie rollen.

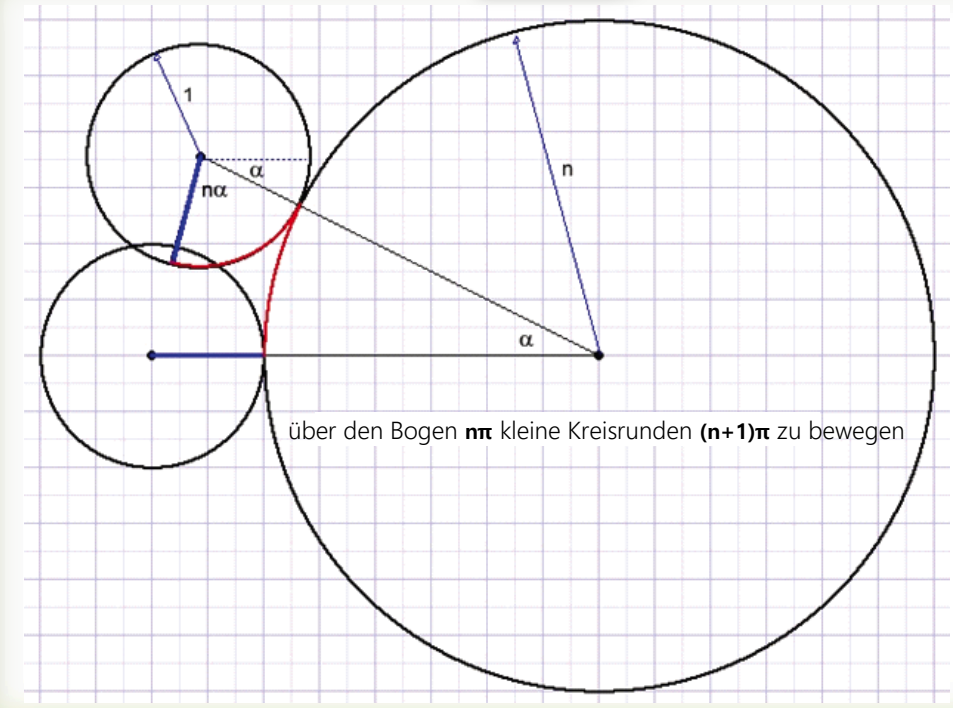


Wir bemerken drei Umdrehungen.



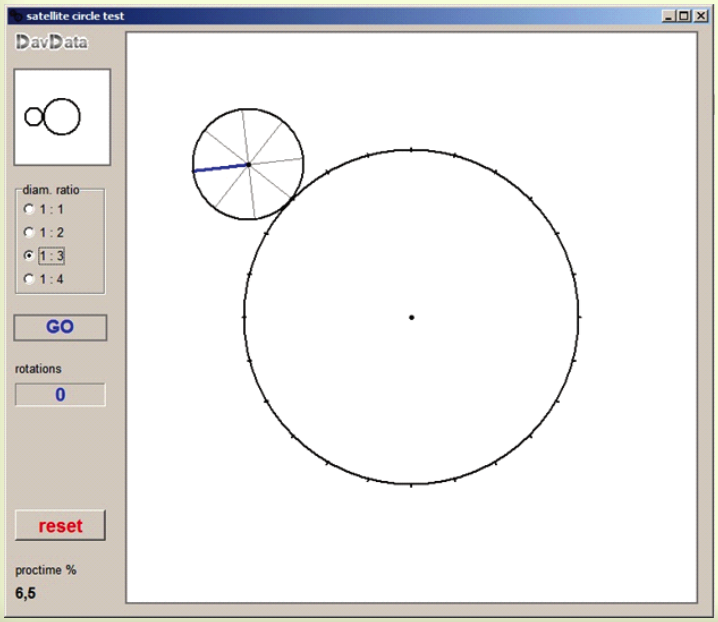
Für diejenigen, die sich für die Mathematik interessieren:

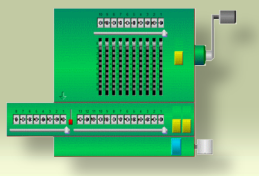
Wir können den Umfang des Kreises B dehnen und den Kreis A über diese gerade Linie rollen.
Wir stellen drei Umdrehungen fest. Für diejenigen, die an der Mathematik interessiert sind:
Die Bewegung des roten Punktes ist durch eine parametrische Funktion gegeben:
x und y werden beide als eine Funktion des Drehwinkels α ausgedrückt.
Aber gilt das auch noch, wenn Kreis A über Kreis B rollt?
Die überraschende Antwort: Nein, tut es nicht. *Sehen Sie sich die Geometrie auf der nächsten Seite an:*



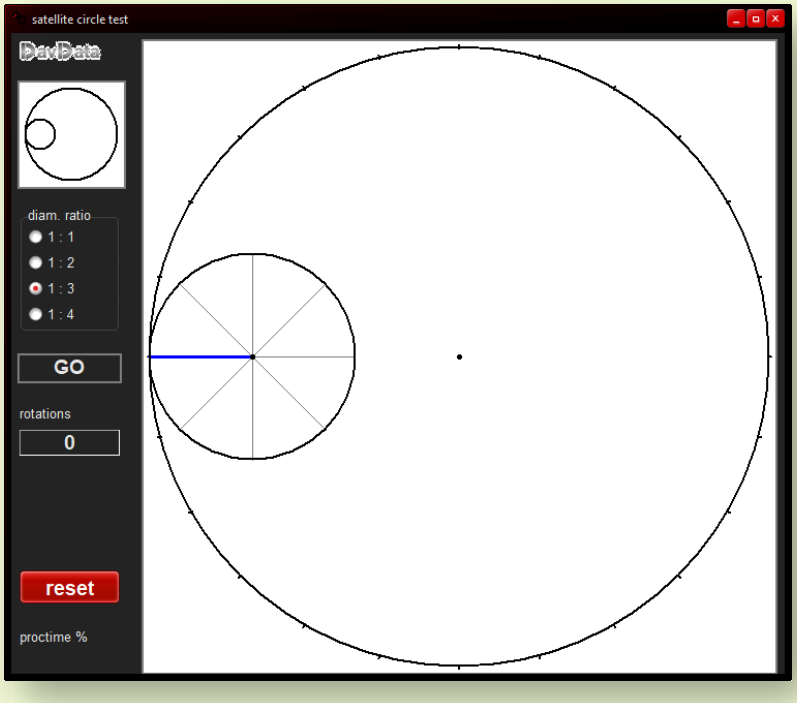
Kreis A braucht vier Umdrehungen, um vollständig über Kreis B zu rollen,
nicht drei wie im Fall einer geraden Linie.
Um dieses Phänomen zu veranschaulichen, habe ich ein kleines Delphi-Programm geschrieben.
Das Programm fügt einige Optionen hinzu
- Äußerer- oder innerer Kreis A
- Durchmesser-Verhältnisse 1:1 , 1:2 , 1:3 , 1:4

Die Theorie für den Fall, dass A der innere Kreis ist, überlasse ich dem Leser.

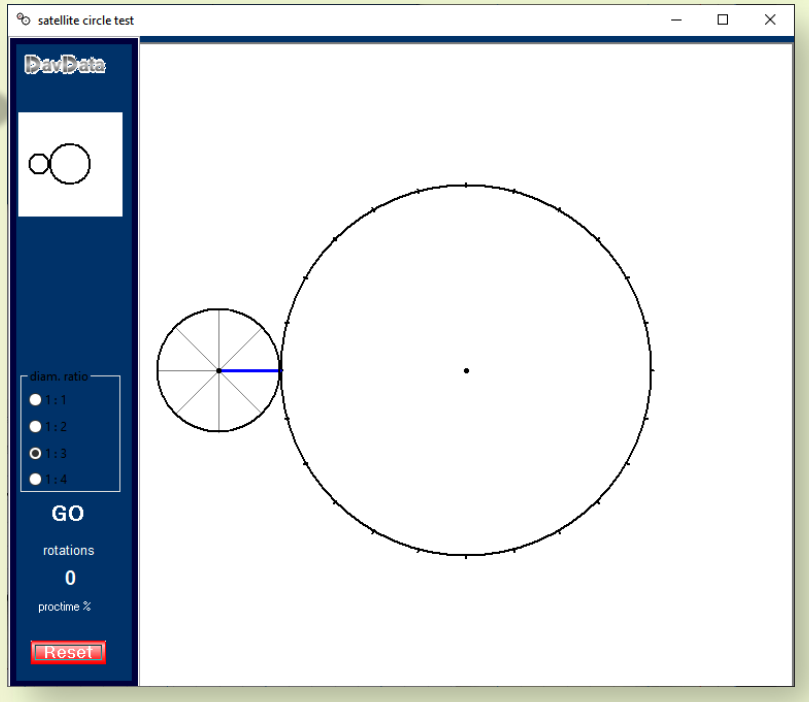


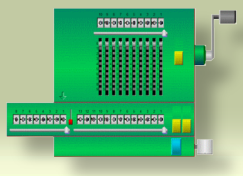


Vorbild in D11 and D12



Vorbild in Lazarus





Wählen Sie das Durchmesserverhältnis.
Klicken Sie auf das Bild links oben, um A als äußeren oder inneren Kreis auszuwählen.
Drücken Sie GO, um Kreis A über Kreis B zu rollen.

PROGRAMMBESCHREIBUNG
Die Zeichnung wird in einer Bitmap-Map erstellt.
Die Karte wird in paintbox1 auf dem Hauptformular kopiert, um sichtbar zu werden.
Das Bild wird 100 Mal pro Sekunde neu gezeichnet.

Das Label links unten zeigt an, wie viel Prozent der Prozessorzeit für eine Aktualisierung benötigt wird.
Siehe 2tes Bild zuvor. Der Winkel α beginnt bei 0 und wird in Schritten von 0,005 Radiant erhöht.
Eine vollständige Umdrehung dauert 2π/0,005 = 1250 Schritte.
Bei 100 Inkrementen pro Sekunde beträgt die Umdrehungszeit 12,5 Sekunden.

Das Malen eines kompletten neuen Bildes benötigt:

- Löschen der Bitmap
- Malen des Kreises B mit entsprechenden Markierungen, die den Speichen des Kreises A entsprechen.
- Malen von Kreis A und seine Speichen.
- Kopieren der Karte in den Malkasten durch `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
```

Die Fortsetzung des Codes finden Sie auf der nächsten Seite

Fortführung des Codes

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;
```

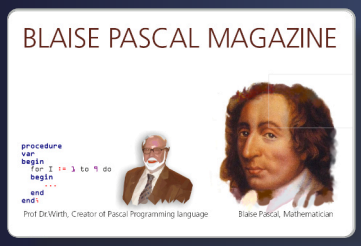
HINWEIS: **GObutton** ist eigentlich ein `TLabel`. Es sieht wie ein Button aus, indem es auf dem Formular-Canvas ein Rechteck um sich herum zeichnet. Die events `OnEnter, OnLeave, OnMouseDown` und `OnmouseUp` verändern die Ränder des Buttons.
Ich führe hier nicht die Prozeduren auf, mit denen die Kreise gezeichnet werden.
Einzelheiten entnehmen Sie bitte dem Quellcode.

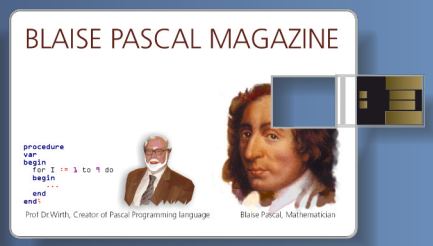
SCHLUSSFOLGERUNG

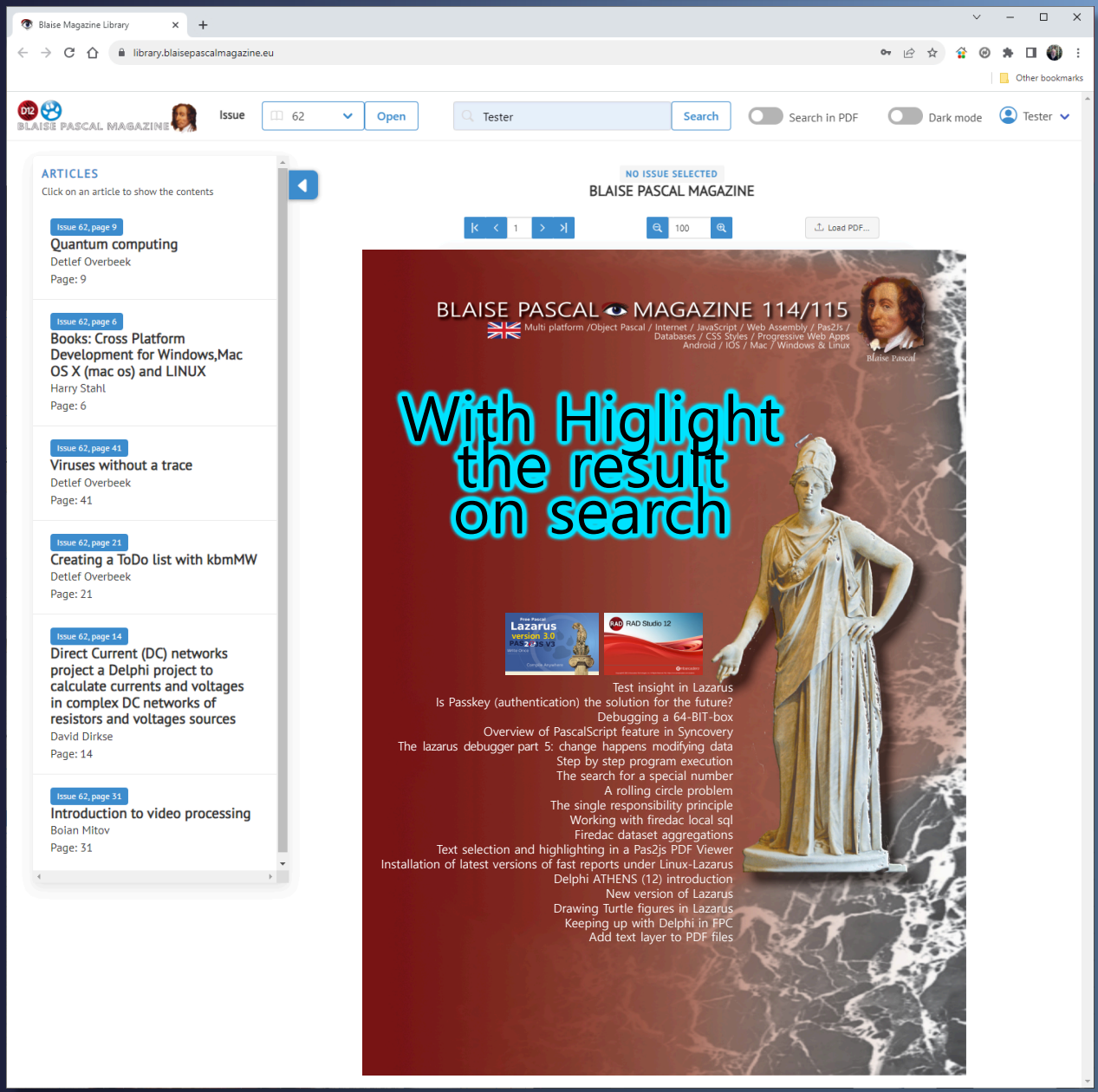
Die Antworten auf das Problem (*siehe erstes Bild*) sind unvollständig.
Es fehlt die Antwort (F) -keiner der oben genannten- oder (F) 4.
Dies war einmal Frage 17 eines SAT (*Scholastic Aptitude Test*) für Oberschüler.
Nur wenige Schüler haben den Fehler bemerkt.
Ich bezweifle, dass die Lehrer, die diese Frage formuliert haben,
sich ihres Fehlers bewusst waren.

LIB-STICK ON USB CREDIT CARD BLAISE PASCAL MAGAZINE

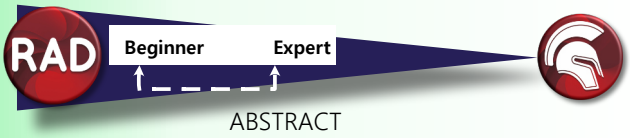
LIB-STICK USB-CARD: ALL ISSUES / CODE INCLUDED. SAME INTERFACE AS THE INTERNET LIBRARY CHRISTMAS OFFER € 89,00







BY MARCO GEUZE



ABSTRACT
Heute werden wir uns mit dem S der Solid-Prinzipien beschäftigen: Das Prinzip der einzigen Verantwortung (Single Responsibility Principle).

SOLID ist ein Akronym für eine Reihe von fünf Softwareentwicklungsprinzipien, die, wenn sie befolgt werden, den Entwicklern helfen sollen, flexiblen und sauberen Code zu erstellen. Die fünf Prinzipien sind:

❶ **Das Prinzip der einzigen Verantwortung**
Klassen sollten nur eine einzige Verantwortung haben und somit auch nur einen einzigen Grund für eine Änderung.

❷ **Das Offen/Geschlossen-Prinzip (Open/Closed Principle)**
Klassen und andere Einheiten sollten offen für Erweiterungen, aber geschlossen für Änderungen sein.

❸ **Das Liskov-Substitutions-Prinzip**
Objekte sollten durch ihre Subtypen ersetzbar sein.

❹ **Das Prinzip der Schnittstellentrennung (Interface Segregation Principle)**
Kunden sollten nicht gezwungen werden, von Schnittstellen abhängig zu sein, die sie nicht verwenden.

❺ **Das Prinzip der Inversion von Abhängigkeiten** (Dependency Inversion Principle)
Verlassen Sie sich lieber auf Abstraktionen als auf Konkretionen.

Also das Prinzip der einzelnen Verantwortung.
Wie der Name schon sagt, darf jede Klasse in einem Programm nur eine einzige Verantwortung für einen Teil der Program-Funktionalität haben. Aber das scheint einfacher zu sein, als es ist.
Was genau ist ein Teil eines Programms, und woher weiß man, wann man die Funktionalität trennen muss? Es ist zu einfach zu sagen, dass eine Klasse nur eine Sache tun sollte.

Robert C. Martin drückt das Prinzip folgendermaßen aus:
"Bringen Sie die Dinge zusammen, die sich aus den gleichen Gründen verändern. Trennen Sie die Dinge, die sich aus unterschiedlichen Gründen verändern", und in jüngerer Zeit: "Bei diesem Prinzip geht es um Menschen". Das sollte uns in die richtige Richtung weisen. Wenn Sie ein Softwaremodul schreiben, möchten Sie sicherstellen, dass bei Änderungen nur von einer einzigen Person oder einer einzelnen, eng miteinander verbundenen Gruppe von Personen stammen können, die eine einzige, eng definierte Geschäftsfunktion vertreten.
Das bedeutet, dass ein Softwaremodul oder eine Klasse **eine einzige Verantwortung für diese bestimmte Gruppe von Personen haben sollte**.

Es ist einfacher, dies anhand eines Beispiels zu erklären. Werfen Sie einen Blick auf die folgende `Klasse:`

```
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;
```

Fortsetzung 1
Solide Prinzipien

Solide Prinzipien

Dies ist eine Klasse, in der es um die Verwaltung der Position und des Kurses eines Schiffes, seiner Ladung und einiger Berichte geht. Da es sich hier um ein kurzes Beispiel handelt, das zeigen soll, wie man über das Prinzip der einzigen Verantwortung nachdenkt, sollten Sie den Details des Codes selbst nicht allzu viel Aufmerksamkeit schenken; es geht hier nur um den großen Überblick über die Struktur dieser speziellen Klasse.

Ich denke, wir alle kennen diese Art von 'Gott'-Klassen. Normalerweise sind sie vollgepackt mit vielen Funktionen und Code und verwalten einen bestimmten Teil oder ein Modul Ihres Programms. Die Frage ist, wie können wir diese Klasse refaktorisieren wenn wir einige Änderungen an dieser Klasse vornehmen müssen, um sicherzustellen, dass wir die Dinge, die sich aus demselben Grund ändern, zusammenfassen und die Dinge, die sich aus unterschiedlichen Gründen ändern, trennen.

Lassen Sie uns einen Moment innehalten und über die Verantwortlichkeiten dieses Codes in Bezug auf die (Gruppe von) Personen nachdenken. Wir können eine Reihe von Personen definieren, die für das Management, die Richtung und den Kurs des Schiffes sowie für die Berichterstattungsinstrumente verantwortlich sind. Nehmen wir also an, dass wir in diesem Fall einen Kapitän, einen Navigator, einen Lademeister und einen Finanzmanager definieren. Wenn Sie an diese verschiedenen Rollen denken, ist es plötzlich sehr einfach, diese Klasse in verschiedene Module aufzuteilen, mit nur einer Verantwortung für diese spezielle Rolle. Wir sollten eine Klasse haben, die den Kurs und die Leistung festlegt (Kapitän), eine, die die Position verwaltet und den Kurs des Schiffes festlegt (Navigator), eine, die die Ladung des Schiffes verwaltet (Lademeister) und eine für die gesamte (*finanzielle*) Berichterstattung (*Finanzmanager*).

Unsere neuen Klassen könnten nun wie folgt aussehen:

```
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;
```

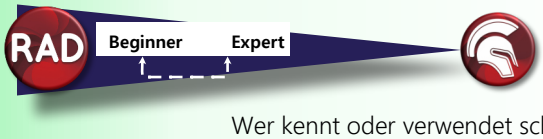
Fortsetzung 2
Solid Prinzipien

Hätten Sie dasselbe getan, wenn Sie nicht an die Menschen hinter den Verantwortlichkeiten der Klassen gedacht hätten? Vielleicht, aber ich kann mir vorstellen, dass die Klassen `ShipLocation` und `ShipMovement` in derselben Klasse hätten landen können.
Was passiert nun, wenn der Kapitän ein Bugstrahlruder anfordert, um das Schiff in kleinen Kanälen leichter steuern zu können? Nehmen Sie diese Änderung einfach in der Klasse `ShipMovement` vor, ohne dass sich dies auf eine der anderen Klassen auswirkt. Und wenn wir ein neues Frachtladesystem einführen wollen? Ändern Sie einfach die Klasse Cargo, wiederum ohne eine der anderen Klassen zu berühren.

Ich hoffe, Sie sehen jetzt, warum es beim Prinzip der einzigen Verantwortung eigentlich um Menschen oder Akteure geht, und die Verantwortung für die Funktionalität von Modulen oder Klassen Ihres Programms im Verhältnis zu diesen Personen. Und natürlich können Sie dies auf verschiedenen Ebenen Ihres Programms anwenden, von Modulen über Klassen bis hin zu bestimmten Funktionen.
Wenn Sie dieses Prinzip beim **Refactoring** oder beim Entwurf eines Moduls oder einer Klasse immer im Hinterkopf haben, bin ich sicher, dass Ihr Code besser wartbar ist und sich leicht ändern lässt.

ARBEITEN MIT FIREDAC LOCAL SQL

VON KEES DE KRAKER
GDK software



Wer kennt oder verwendet schon die Firedac-Komponente **LocalSQL**?
Oder den **BatchMove**?
Für die meisten Entwickler sind diese Komponenten relativ unbekannt,
auch wenn sie sehr nützlich sind.
In zwei Artikeln möchte ich den Mehrwert dieser Komponenten erläutern und zeigen, wie sie funktionieren. In diesem Artikel geht es um die Komponente **LocalSQL**.

DER MEHRWERT

Mit dem **LocalSQL**-Datensatz können Sie Daten aus verschiedenen Datensätzen in einer Firedac-Abfrage kombinieren.
Die Komponente sorgt dafür, dass Datasets so verfügbar werden, als wären sie Datenbanktabellen. Datasets können daher mit einer SQL-Abfrage aus einer `TFDQuery` Komponente abgefragt werden.

Dies ist sehr nützlich, wenn Sie Daten aus verschiedenen Quellen haben.
Zum Beispiel aus mehreren Datenbanken.
Oder teilweise im Speicher und teilweise in der Datenbank.
Oder aus verschiedenen Dataset-Komponenten, z.B. **ADO** und **Firedac**.
Mit **LocalSQL** können Sie diese Daten ganz einfach mit einer einfachen SQL-Abfrage zusammenführen. Damit können Sie die kombinierten Daten ganz einfach in einem Raster anzeigen oder kombinierte Statistiken erstellen.

Nachfolgend habe ich ein Beispiel ausgearbeitet, bei dem ich Daten aus einer **CSV**-Datei mit Daten aus der Datenbank kombiniert habe.
In der Datenbank habe ich Kunden mit einer ID und einem Namen gespeichert.
In der **CSV** wird nur die ID des Kunden verwendet.
Wenn ich den Inhalt der **CSV**-Datei anzeige, möchte ich den Namen des Kunden direkt anzeigen.
Das ist mit **LocalSQL** ohne allzu großen Aufwand möglich.

Fortsetzung 1 Local Sql

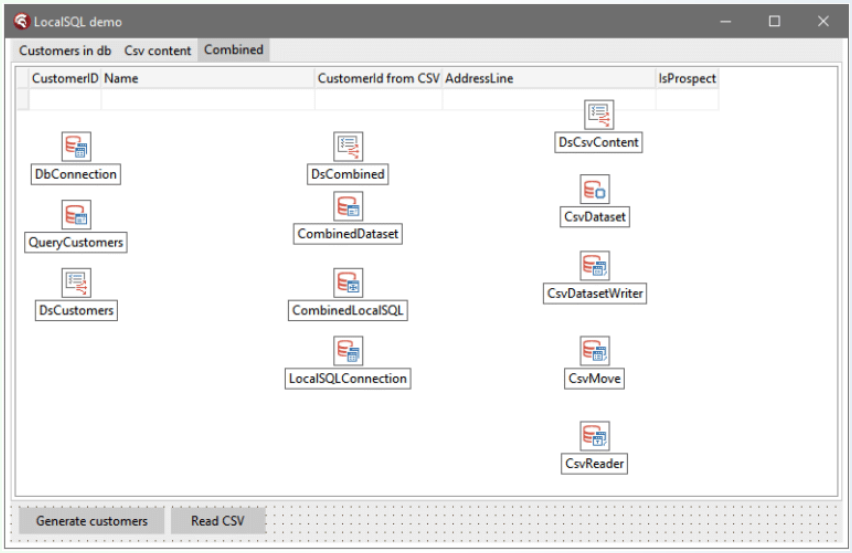
DER GRUNDLEGENDE ENTWURF
Auf der nächsten Seite sehen Sie das Setup meiner Anwendung.
Auf der linken Seite ist die Verbindung zur Datenbank mit der Tabelle **Customer**.
Diese wird über die **QueryCustomers** gelesen.

Auf der rechten Seite sehen Sie das **CsvDataset**, eine **Firedac MemTable**.
Darin lädt der **CsvMove** (`TFDBatchMove`) die Daten aus der **CSV**-Datei.

In der Mitte sehen Sie die Komponente LocalSQL, die ich **CombinedLocalSQL** genannt habe.
Darunter verwendet diese Komponente **SQLite** und benötigt daher eine **SQLite**-Verbindung.
Sie müssen diese Verbindung nicht weiter konfigurieren, sondern nur angeben, dass der Treiber **SQLite** ist.

In diesem Fall ist dies die **LocalSQLConnection**.
Die Komponente **CombinedDataset** ist eine `TFDQuery`, die die **SQL**-Abfrage enthält,
mit der die Daten aus den Datasets gesammelt werden.

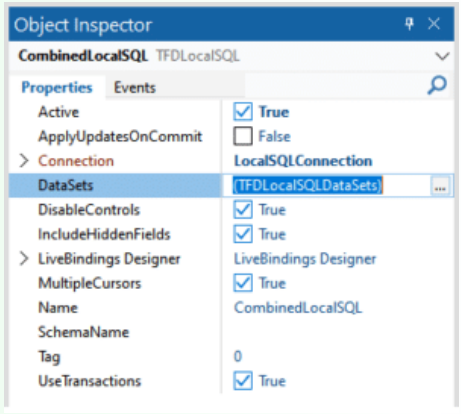
Die Tabelle zeigt zunächst zwei Spalten mit den Kundendaten aus der Datenbank an.
Die drei folgenden Spalten stammen aus der **CSV**-Datei.

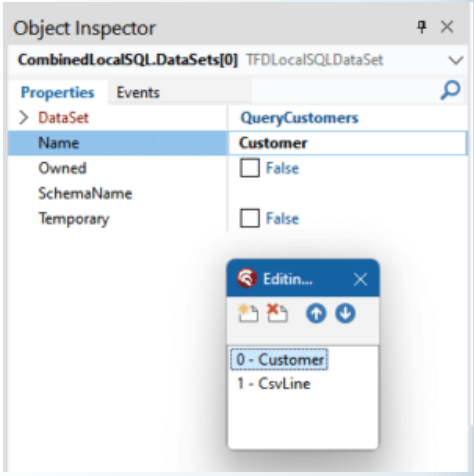


Fortsetzung 2
Local Sql

DIE KONFIGURATION

Die Komponente **LocalSQL** stellt, wie bereits erwähnt, Datensätze so zur Verfügung, als wären sie Datenbanktabellen. Um dies zu ermöglichen, legen wir fest, welche **Datasets** wir verwenden möchten und welchen "**Tabellennamen**" sie erhalten sollen. In den Eigenschaften der Komponente **LocalSQL** finden Sie zu diesem Zweck die Option ‚**DataSets'** (*siehe Bilder unten*).
Fügen Sie die Datasets hinzu und geben Sie ihnen den Namen, den Sie als Tabellennamen in der Abfrage verwenden möchten.
Wenn Sie die Komponente **LocalSQL** auf Aktiv setzen, können Sie die Abfrage anschließend live ausführen und z.B. die Felder einfach hinzufügen. Seien Sie dabei vorsichtig, denn sobald Sie die Abfrage ausführen, werden auch die zugrunde liegenden Datasets auf Aktiv gesetzt.





DIE KOMBINIERTE ABFRAGE

Wir verwenden eine **Firedac-Abfrage**, um die LocalSQL anzuwenden. Die Verbindung dieser Abfrage ist die gleiche wie die **LocalSQL-Verbindung**. Und das ist alles. Wir können jetzt mit der Erstellung der Abfrage beginnen, als ob wir eine Tabelle **Customer** und eine Tabelle **CsvLine** hätten.

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

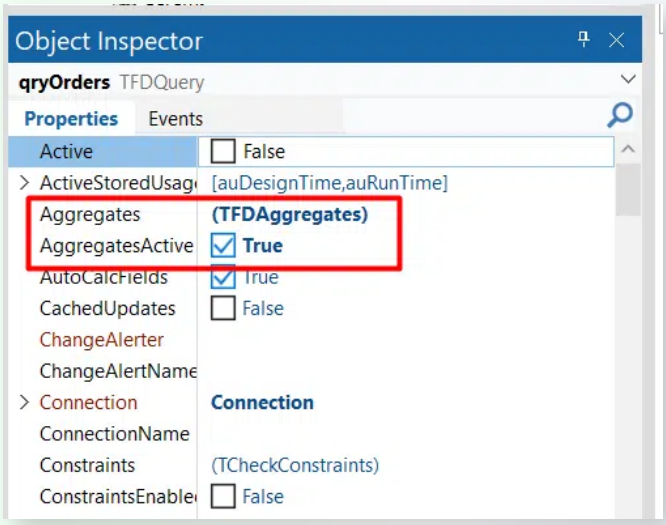
Und natürlich können Sie diese Abfrage mit allen Möglichkeiten, die (**SQLite**) **SQL** bietet, erweitern und verwenden. Denken Sie an die **WHERE**-Klausel, aber auch an **Aggregationsfunktionen** wie `SUM`, `COUNT`, usw. Die Firedac-Abfrage funktioniert dann wie gewohnt.

ZUSAMMENFASSUNG
Die Komponente LocalSQL eröffnet eine Vielzahl von Möglichkeiten Daten zu kombinieren, zu filtern und zu aggregieren. Sie ist breit einsetzbar, da sie alle Formen von Datensätzen unterstützt. Sie kann komplizierte Datensätze mit lookup fields erheblich vereinfachen. Und es funktioniert schnell und einfach. Sehr empfehlenswert für den Einsatz in Ihren Projekten.

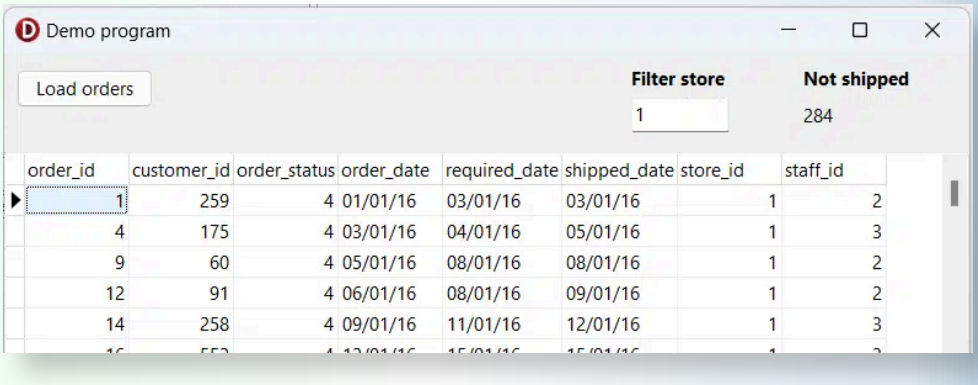
VON KEES DE KRAKER

Wussten Sie schon, dass Sie in einer Firedac-Abfrage oder -Tabelle mit **runtime Aggregationen** auf Datensatzebene arbeiten können? Die Anzeige einer Gesamtsumme aller geladenen Bestellungen oder einer Gesamtsumme aller noch zu versendenden Bestellungen kann sehr einfach erfolgen, ohne dass eine separate Abfrage erforderlich ist.

Vielleicht waren Sie bereits mit einem aggregierten Feld vertraut, aber das ist auf Zeilenebene beschränkt. Mit der Eigenschaft `Aggegrates` eines `TFDDataSet` können Sie jedoch auch auf Datensatzebene aggregieren.



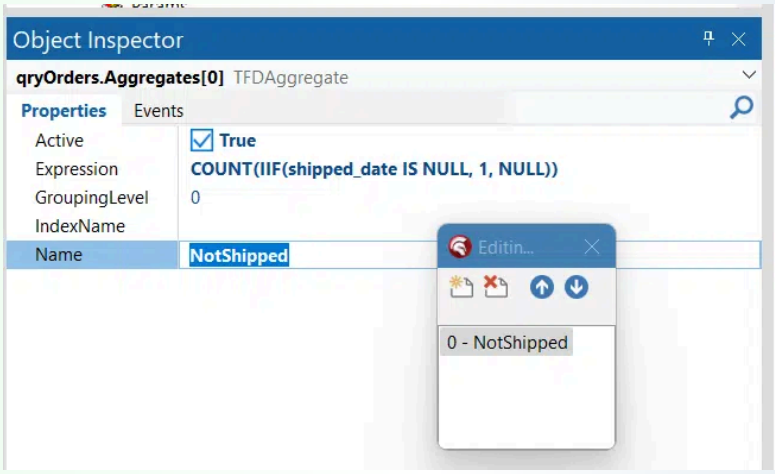
Als Beispiel habe ich ein **Demoprogramm**, in dem ich alle Bestellungen aus einer Datenbank in eine Tabelle abrufe. Ich kann diese nach der Filialnummer filtern. Oben rechts sehen Sie die Anzahl der Bestellungen, die noch versendet werden müssen. Diese Zahl wird durch ein **aggregiertes** Feld der Abfrage berechnet.



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

Zunächst definiere ich ein Aggregationsfeld in der Eigenschaft **Aggregation** der Abfrage. Das wichtigste Feld hier ist **Expression**. Hier definieren wir die Aggregation, die wir durchführen möchten. Möglicherweise mit einer Bedingung darin. Hierfür wird die standardmäßige **Firedac** Ausdruckssyntax verwendet.

Fortsetzung 1
Dataset Aggregations



Im obigen Beispiel zähle ich die Anzahl der Datensätze (**COUNT**) des Feldes **Shipped_Date** mit der Bedingung, dass dieses Feld nur gezählt wird, wenn der Wert leer ist. Hierfür kann die Funktion IIF verwendet werden. Andere einfache Beispiele, die Sie hier verwenden können, sind:

```
SUM(order_amount)
```

```
MIN(order_date)
```

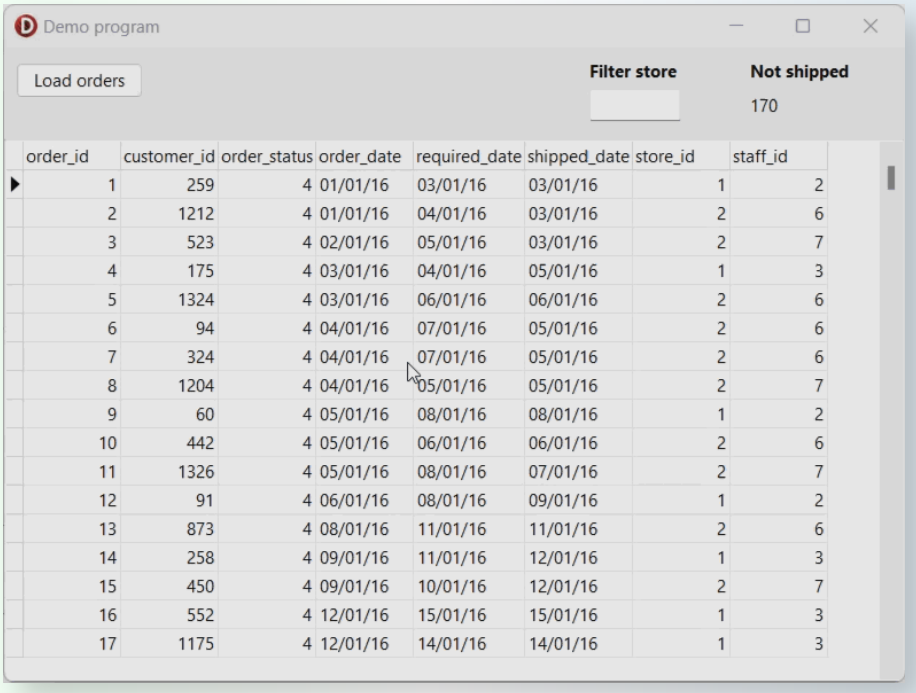
Wie Sie in den Eigenschaften sehen können, können Sie ein Feld auf aktiv setzen. Wenn Sie das nicht tun, wird es natürlich nicht berechnet.
Auch ein Name kann nützlich sein, damit Sie die Aggregation im Code finden können.

Leider können Sie eine **Aggregation** nicht als Datenbankfeld mit einem datengesteuerten Button verknüpfen. Das Rendering muss also im Code behandelt werden.
Ich habe dafür eine eigene Procedure erstellt, die ich einfach aufrufen kann, wenn eine Aktualisierung erforderlich ist. Ich rufe diese Procedure im `AfterOpen`-Ereignis der Abfrage und beim Filtern auf einem Speicher auf.

```
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;
```

Fortsetzung 2
Dataset Aggregations

Denken Sie daran, in der Abfrage die Eigenschaft `AggegratesActive` auf `True` zu setzen, da sonst die **Aggregationen** nicht berechnet werden.
Ein weiterer Punkt, den Sie beachten sollten:
Die **Aggregate** arbeiten mit den im Dataset abgerufenen Daten.
Standardmäßig ruft Firedac 50 Datensätze ab.
Wenn Sie eine Gesamtsumme wünschen, die auf allen Datensätzen basiert,
gehen Sie zu den **FetchOptions** in den **Abfrageeigenschaften** und setzen Sie den Modus auf `fmAll`.



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

GDK software

embarcadero®
Official Technology Partner

We are GDK.

Do you have a Delphi challenge and are you looking for expertise or capacity? Our experts and developers are ready to realise your ambitions and goals.

30 **Delphi developers**
Work with our Delphi Experts

99+ **Delphi conversions**
Smart upgrade to the latest Delphi

5 **Embarcadero MVP's**
Authority within the Delphi community

4 **Offices worldwide**
The Netherlands, UK, Brazil and USA

GDK IN A NUTSHELL

About GDK.

We share a passion for software development and love to keep up with the latest technologies. We have a strong team of specialists with a lot of knowledge and expertise in Delphi.

ACHIEVING YOUR AMBITIONS

Work together.

Looking for a partner to maintain and extend your software? Or upgrade your software to the newest Delphi version? We are ready to help you!

Contact us

www.gdksoftware.com

SCAN ME

Codolex
Simplify development

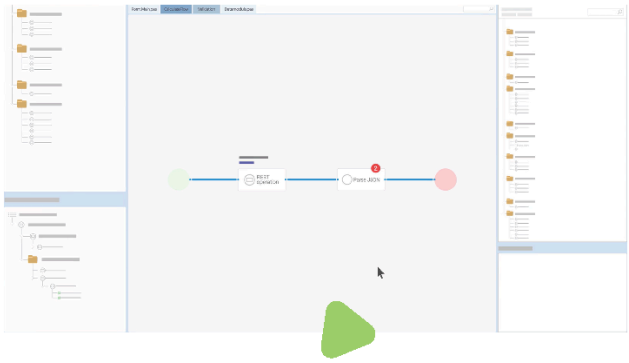
embarcadero®
Official Technology Partner

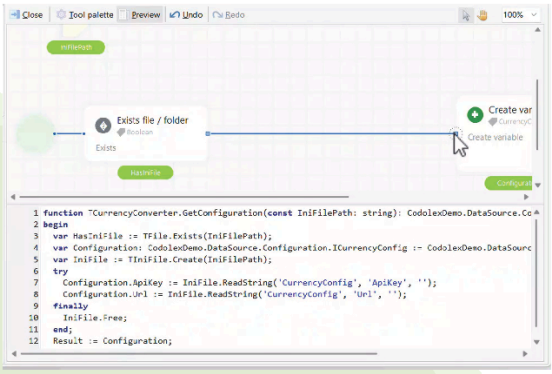
Revolutionary.

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!





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!

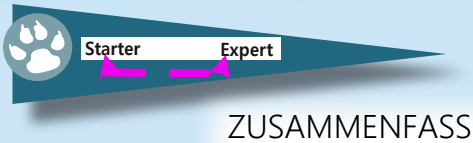
SCAN ME

www.codolex.com

Try it out

TEXTAUSWAHL UND -HERVORHEBUNG (HIGHLIGHTING) IN EINEM PAS2JS PDF VIEWER

VON MICHAEL VAN CANNEYT



ZUSAMMENFASSUNG

Im letzten Artikel haben wir eine Methode eingeführt, um PDF-Dateien im Browser anzuzeigen und im PDF zu suchen. In diesem Artikel führen wir eine fehlendes Feature ein: Hervorhebung von Suchergebnissen im Text und Textauswahl.

❶ EINLEITUNG

In einer Reihe von Artikeln über PAS2JS haben wir gezeigt, wie man eine PDF-Datei im Browser anzeigen kann und wie man nach einem Text in dem angezeigten PDF sucht. Dies kann relativ einfach mit PDF.js, der Javascript-Bibliothek zur Anzeige von PDFs von Mozilla, durchgeführt werden.
Später fügten wir eine in Free Pascal geschriebene Funktion hinzu, in einer Reihe von PDFs zu suchen, indem wir einen serverseitigen Mechanismus sowie einen Indexer benutzen.
Das Ergebnis ist die Grundlage für die Blaise Pascal Magazine Library, eine durchsuchbare Bibliothek der im Blaise Pascal Magazine veröffentlichten Artikel. Dies ist ein Programm, das sowohl offline als auch online funktioniert.
Diese Demoprogramme hatten 2 Nachteile:
Der erste Nachteil war, dass der Suchmechanismus darauf beschränkt war, den Text zu finden und die richtige Seite in der PDF-Datei anzuzeigen.
Der zweite Nachteil war, dass es nicht möglich war, Text in der PDF-Datei auszuwählen (zum Beispiel zum Kopieren).
Glücklicherweise enthält die PDF.js-API einige Aufrufe, die diese beiden Probleme lösen.
In diesem Artikel zeigen wir, wie man einen dieser Aufrufe verwendet und damit beide Probleme auf einmal löst.

❷ PDF.JS AKTUALISIEREN

Bevor wir loslegen ist ein kleiner Umweg nötig:
In der Zeit zwischen der Veröffentlichung der verschiedenen Programme, die die Funktionsweise der PDF.js-Bibliothek zeigen, hat sich die Bibliothek in einer Weise verändert, die Änderungen im Pascal-Code und im HTML-Code erforderlich macht. Diese Änderungen sind nicht umfangreich, aber sie sind notwendig, weil Ihr Programm dann nicht mehr funktioniert, wenn es nicht sowieso schon aufgehört hat zu funktionieren, was wahrscheinlich ist, wenn Sie die Online-Distribution von PDF.js über ein CDN (Content Delivery Network) genutzt haben. Zuvor stellte die PDF.js-Bibliothek eine globale Variable pdfjsLib zur Verfügung, die in den Programmen definiert waren als:

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

Die PDF.js-Bibliothek wird nun als Javascript-Modul erstellt, ähnlich wie eine dynamisch ladbare Bibliothek. Die Dateinamenserweiterung der Bibliothek wurde in .mjs geändert, um dies widerzuspiegeln.
Dies hat zur Folge, dass die obige Variable nicht mehr definiert ist, wenn Sie es als normales Skript laden, da der Browser das Laden des Skripts ablehnt.
Die Lösung besteht darin, die Bibliothek als Modul zu laden. Das bedeutet einfach, dass Sie dem Skript-Tag, das PDF.js enthält, ein type-Attribut mit dem Wert module hinzufügen müssen, also so:

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

Dann wird das Skript korrekt geladen und die globale Variable wird definiert.
Wenn Sie jedoch eine private Kopie des Legacy-Builds von PDF.js verwenden, müssen Sie nichts ändern.

③ EIN AUSWAHL- UND HERVORHEBUNGSMECHANISMUS

Der Grund, warum die ersten Versionen des PDF-Viewers keine Hervorhebung (HIGHLIGHTING) und Auswahl zuließen, liegt darin, dass die PDF-Datei einschließlich des Textes grundsätzlich als Bitmap auf der HTML-Seite gerendert wird.

Es sollte klar sein, dass als Bild gezeichneter Text nicht ausgewählt werden kann, es sei denn, man verwendet **OCR (Optical Character Recognition).** Deshalb ist ein anderer Mechanismus erforderlich.
Glücklicherweise bietet **PDF**.js eine Lösung: Es enthält einen Mechanismus, um die Textelemente einer **PDF-Datei als HTML** zu rendern und auf der HTML-Seite zu überlagern.

Dieses HTML ist so angelegt, dass es völlig **transparent** ist und der **Benutzer es nicht sehen kann**. Es handelt sich jedoch um echtes **HTML**, das Teil des **DOM** der **HTML-Seite** ist und mit den üblichen **DOM**- und **CSS-Techniken** manipuliert werden kann.
Außerdem wird das "versteckte" HTML berücksichtigt, wenn im Browser etwas ausgewählt wird: Das **Pseudo-Element "selected"** funktioniert ebenfalls und „**gestyled**“, also mit Attributen versehen werden:

Durch **Änderung der Hintergrundfarbe des Pseudoelements "selected"** kann der ausgewählte Text sichtbar gemacht werden. Das heißt, wenn wir die **PDF.js-API zum Rendern von Text** verwenden, haben wir unseren **Textauswahlmechanismus**.
Da das gerenderte HTML den eigentlichen Text enthält (*aber eben unsichtbar gerendert*), bedeutet dies, dass die Seite angezeigt werden kann, die eine Übereinstimmung mit dem Text enthält. Wir können wir das erstellte **HTML** durchlaufen und Übereinstimmungen im Text hervorheben. Im Folgenden wird gezeigt, wie man das macht.

④ AUSWAHL ERMÖGLICHEN: SEITENTEXT ALS HTML WIEDERGEBEN.

Die PDF.js-API hat 2 Aufrufe, die das HTML-DOM mit dem Text erstellen oder aktualisieren der PDF-Datei. Beide werden mit Hilfe einer Hintergrundaufgabe implementiert und beide geben eine Instanz dieser Hintergrundaufgabe zurück:

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

Der Aufruf, der uns interessiert, ist die Aufgabe RenderTextLayer.
Sie nimmt das folgende Parameterobjekt an:

```
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;
```

Die ersten 5 Felder in dieser Klasse sind Eingabefelder:

textContentSourceStream	Der Textinhalt der PDF-Datei als Stream.
textContentSourceItems	Der Textinhalt der PDF-Datei, wie er mit dem `getTextContent`-Aufruf des `TPDFPageProxy`-Objekts zurückgegeben wird.
container	Das HTML-Element, in dem der Text dargestellt werden soll.
viewport	Das Ansichtsfenster, mit dem die PDF-Datei gerendert wurde.
isOffscreenCanvasSupported	Auf True gesetzt, wenn die PDF-Ebene einen Offscreen-Canvas verwenden kann. *(wird benötigt, um die Textbreiten zu bestimmen)*

Fortsetzung 4

Die letzten drei Felder der Klasse werden gefüllt und ausgegeben, nachdem der Aufruf von `renderTextLayer` seine Arbeit getan hat. Zu Beginn sollten sie auf leere **Arrays** gesetzt werden.ty arrays.

`textDivs`	Ein **Array** mit den generieten **HTML** Elementen..
`textDivProperties`	Ein **Array** mit den Eigenschaften, die zur Generierung des **HTML-Elements** benötigt werden.
`textContentItemsStr`	The raw texts of the generated **HTML** elements.

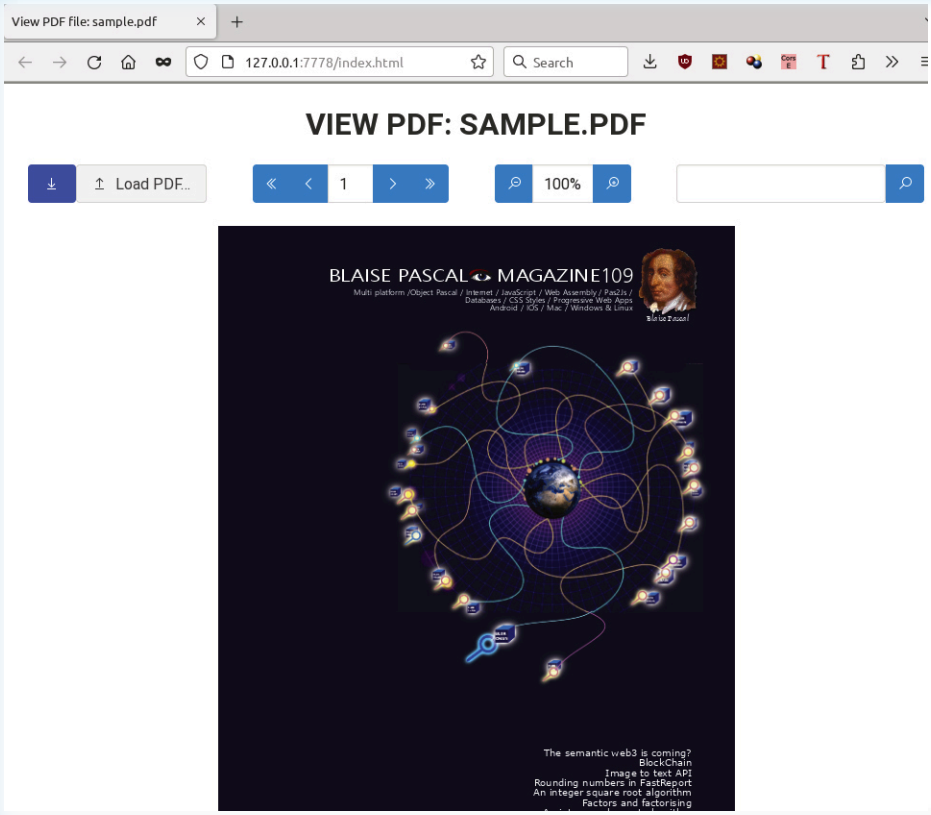
Wir werden diesen Aufruf anhand des Beispiels für die **PDF-Suche** das wir bereits vorgestellt haben, demonstrieren. In diesem Beispiel konnte der Benutzer eine URL eingeben oder eine **PDF-Datei** auswählen, die dann angezeigt und durchsucht werden konnte. Zur Erinnerung - *Abbildung 1 auf Artikelseite 3* – zeigt wie die Anwendung aussah. Wir werden dieses Beispiel wiederverwenden und erweitern. Der Quellcode steht Ihnen natürlich wieder zur Verfügung. Um den **`renderTextLaye`**-Aufruf nutzen zu können, müssen wir dem **HTML** für unseren **PDF-Viewer** zusätzliche Elemente hinzufügen. Bisher hatten wir folgenden **HTML-Code**, in dem die **PDF-Datei** angezeigt wurde:

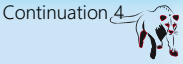
```
<div class="is-flex is-justify-content-center">
  <canvas id="PDFCanvas" height="737" width="538"></canvas>
</div>
```

Wir benötigen nun ein zusätzliches Element, das zum Überlagern des Textes verwendet wird (*mit der ID pdfTextLayer*), und ein weiteres Element, das einen zusätzlichen Stilparameter enthält (*mit der 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>
```

Abbildung 1:
Der PDF-Viewer in Aktion





Die beiden IDs werden in unserem Programm mit zwei Variablen verbunden:

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

Das Rendern einer Seite des PDF-Dokuments wurde in der RenderPage-Methode unseres Programms durchgeführt:
In dieser Methode wird `GetPage` aufgerufen, um ein Proxy-Objekt für eine PDF-Seite abzurufen. Die Render-Methode wird verwendet, um die Seite tatsächlich im Canvas zu rendern:
In Computernetzwerken ist ein Proxyserver eine Serveranwendung, die als Vermittler zwischen einem Client, der eine Ressource anfordert, und dem Server, der diese Ressource bereitstellt, fungiert.

```
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;
```

Wie Sie im Code der `havePage`-Routine sehen können, sind die Parameter für die PDF-Seiten-Renderaufgabe die gleichen wie die für den `RenderTextLayer`-Aufruf. Es ist daher sinnvoll, sie in einen Record aufzunehmen, der in unserer Anwendungsklasse deklariert wird: (*Siehe nächste Seite*)

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;
```

Wir können nun die havePage-Prozedur so umschreiben, dass sie den neu eingeführten Record verwendet, um die zum Rendern der Seite benötigten Informationen zu speichern:

```
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;
```

Wenn das Rendering abgeschlossen ist, wird die Prozedur `RenderOK` aufgerufen. Dort können wir nun den notwendigen Code zum Rendern des Textes hinzufügen. Wir beginnen mit dem Abrufen des eigentlichen Textes mit den Aufruf von `getTextContent`. Der aufmerksame Leser wird feststellen, dass der Aufruf `getTextContent` derselbe Aufruf ist, mit dem auch der **PDF-Text** für die Suche in der **PDF-Datei** abgerufen wurde. Der `renderOK`-Aufruf wird dann zu:

```
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;
```

Fortsetzung

Das pdfViewer-Element erhält eine **Style**-Eigenschaft, die eine **CSS**-Variable für den Skalierungsfaktor setzt. Diese Variable wird von dem **CSS** benötigt, welches von **PDF.js** generiert wird.
Der Wert der Variable wird auf die aktuelle Skalierung gesetzt, die zum Zeichnen der PDF-Datei verwendet wird. Wird sie vergessen (*oder auf einen falschen Wert gesetzt*), so wird der HTML-Code nicht korrekt über dem PDF-Bild positioniert. Wenn der Text abgerufen wird, wird der RenderText-Callback, den wir dem von getTextContent zurückgegebenen **Javascript-Promise-Objekt** übergeben haben, aufgerufen. Der Callback bereitet einfach die Argumente für den **RenderTextLayer-API-Aufruf** vor, unter Verwendung der `FCurrentPageInfo` und das Ergebnisses dieses Versprechens ist:

```
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;
```

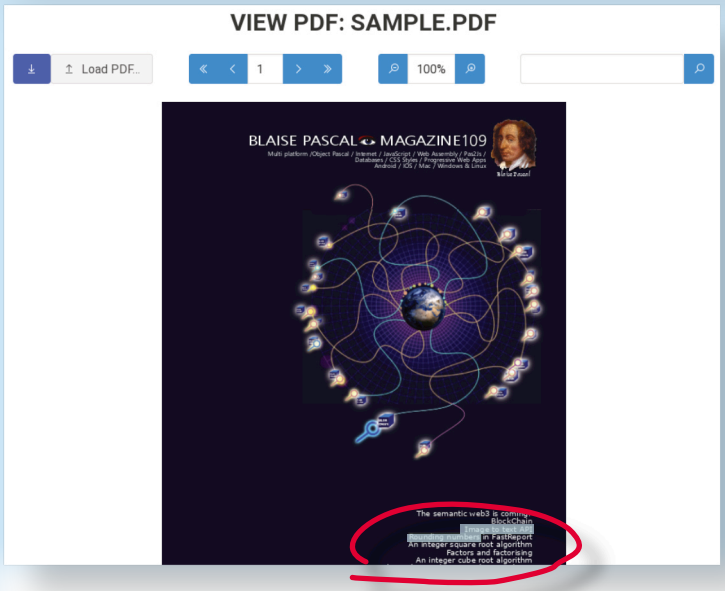


Abbildung 2: Auswahl im PDF-Viewer in action

Mit dem obigen Code ist die Möglichkeit, Text zu markieren und zu kopieren, fast gegeben. Das erforderliche HTML wird erzeugt und an der richtigen Stelle in der HTML-Seite positioniert.
Was noch fehlt, ist etwas CSS, das für das generierte HTML benötigt wird. Der interessierte Leser findet es in der Datei viewer.css, die wir mit einem einfachen Link-HTML-Element in die HTML-Seite einbinden:

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

Das Ergebnis ist in Abbildung 2 auf Seite 8 zu sehen: Ein Teil des Textes (über mehrere Zeilen) wird in der Übersicht der Artikel ausgewählt. Wird die Skalierung des PDFs geändert, wird das PDF neu gerendert, und der Text wird ebenfalls neu gerendert. Da die CSS-Variable scale vor dem Rendering gesetzt wird, sind die beiden immer synchronisiert.

❺ HERVORHEBUNG VON SUCHERGEBNISSEN

Was noch fehlt, ist die Hervorhebung von Suchergebnissen, wenn eine Seite als Ergebnis eines Suchvorgangs gerendert wurde. Das Hinzufügen dieser Funktionalität ist nicht so schwierig: Wir können es im **TextDone-Callback** des **renderTextLayer**-Aufrufs machen. Wenn wir uns das von **PDF.js** generierte HTML ansehen, sehen wir, dass es sich einfach um eine flache Reihe von span-HTML-Elementen handelt, die etwas positioniert wurden. Die **span-Elemente** enthalten nur Text. Um den Text hervorzuheben, können wir einfach eine Schleife über die span-Elemente ziehen und prüfen, ob das **span-Element** den Text enthält. Im Grunde ist dies die gleiche Schleife wie im Suchalgorithmus, mit dem Unterschied, dass wir jetzt über das generierte **HTML** laufen, anstatt über die Strukturen, die vom **getTextContent**-Aufruf zurückgegeben werden.
Der folgende Code prüft, ob eine Suche durchgeführt wurde.
Wenn nicht, wird er sofort beendet.
Wenn eine Suche durchgeführt wurde, bereitet er den für die Suche verwendeten regulären Ausdruck vor und durchläuft eine Schleife über alle span-Tags, wobei **Highlight** für jedes span-Element aufgerufen wird.

```
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 der **highlight-Routine** ändern wir das innere HTML des span-Elements. Wenn einer oder mehrere Suchtreffer gefunden wurden, dann wird der/die Treffer in ein neues span-Element gepackt, dem wir die **CSS**-Klasse '**highlight**' geben. Wenn zum Beispiel der Suchtext '**integer**' ist, dann konvertieren wir das folgende span-Element:

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

in

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

Die ‚**highlight**' wird vom CSS zum transparenten Einfärben des Texthintergrunds genutzt und der text erscheint als hervorgehoben.

Das kann folgendermaßen gecoded werden:
Es handelt sich dabei um eine einfache Schleife unter Verwendung des regulären Ausdrucks. Solange der reguläre Ausdruck eine Übereinstimmung findet, wird der Text zwischen der Übereinstimmung und dem Ende der vorherigen Übereinstimmung unverändert zum span-Element übernommen. Dann wird der übereinstimmende Text, eingebettet in ein neues span-Element, hinzugefügt.
Wenn es keine weiteren Übereinstimmungen gibt, fügt die Routine den restlichen Text an.

```
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;
```

Damit ist die Hervorhebung abgeschlossen. Das Ergebnis ist in Abbildung 3 auf Seite 11 zu sehen.

❻ SCHLUSSFOLGERUNG

Das Hinzufügen einer Textebene zum **PDF.js**-Viewer ermöglicht die Implementierung von Auswahl- und Kopier- und Einfügeoperationen ohne weiteres Zutun.
Wie in diesem Artikel gezeigt, lässt sich damit auch die **Hervorhebung von Suchbegriffen** auf einer Seite recht einfach realisieren.
Der Mechanismus ist nicht perfekt, da der PDF-Text manchmal in verschiedene PDF-Elemente aufgeteilt wird *(z. B. bei Formatierungsänderungen)*:
Der **HTML**-Text wird folglich über **HTML-Tags** aufgeteilt.
Für gängige Situationen ist der Mechanismus aber durchaus geeignet.

Beispiel: Siehe nächste Seite.

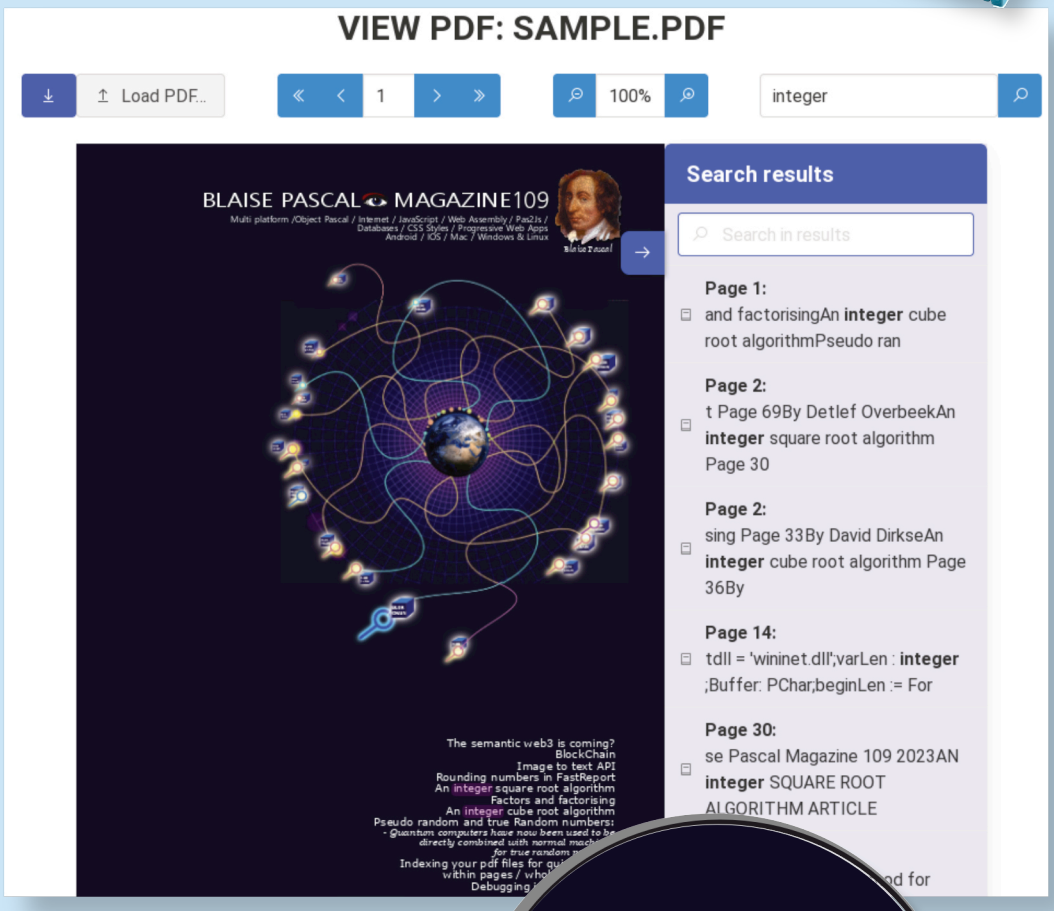
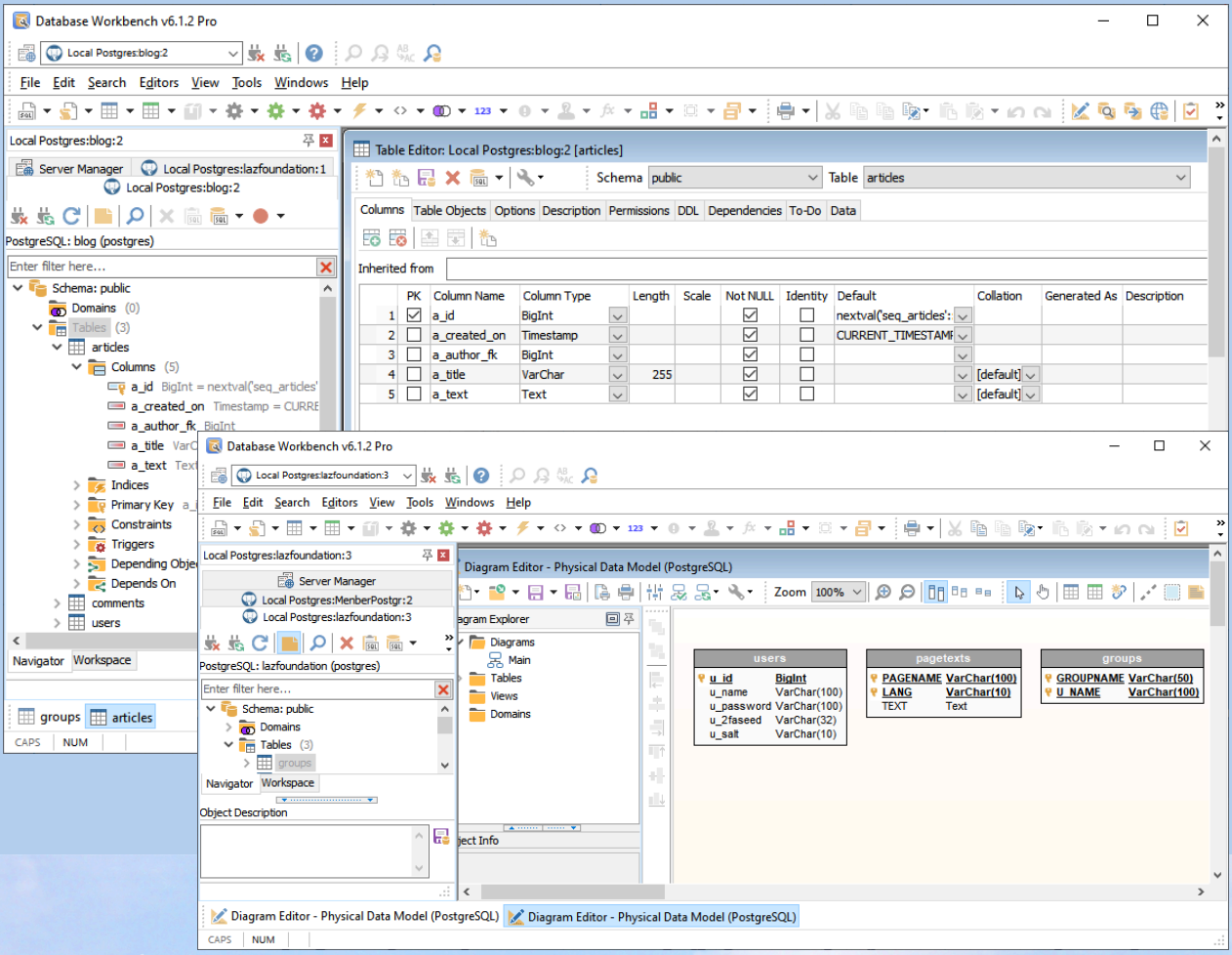


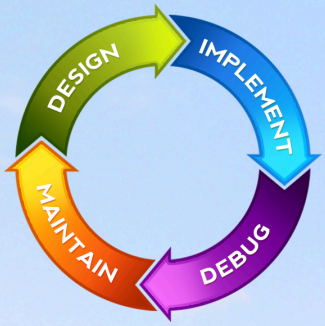
Abbildung 3: Hervorhebungen im PDF Viewer in Aktion



Introducing

Database Workbench 6

database development environment



Consistent user interface, modern code editors, Unicode enabled, HighDPI aware, ER designer, reverse engineering, meta data browsing, visual object editors, meta data migration, meta data compare, stored routine debugging, SQL plan visualizer, test data generator, meta data printing, data import and export, data pump, Grant Manager, DBA tasks, code snippets, SQL Insight, built in VCS, report editor, database meta data search, numerous productivity tools and much more...

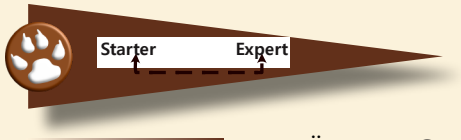
for SQL Server, Oracle, MySQL, MariaDB, Firebird, InterBase, NexusDB and PostgreSQL

Upscene *Database tools for developers*

www.upscene.com

INSTALLATION DER NEUESTEN VERSION VON FASTREPORT FÜR LAZARUS UNTER LINUX.

BY ALKEXANDER REDKOW



EINFÜHRUNG

In der **Ausgabe 110-111 des "Blaise Pascal Magazine"** gab es eine Anzeige über ein Update der Fast Report Software und einen Artikel **"FASTREPORT FOR LAZARUS - LINUX"** von **Sergey Plastun**.
Ich bin ein begeisterter Benutzer dieser Software seit Version "0", habe alle Versionen durchlaufen und verwalte fast tausend Berichtsformulare. Vor kurzem habe ich eine interessante Erfahrung mit der letzten **FR-Version FastreportVCL Pro 2023.2** gemacht. In der Werbung hieß es:
Ein Installationssystem mit Online-Autorisierung
- installieren und aktualisieren Sie alle Ihre Produkte auf einmal.

INSTALLIEREN

Nun, das klingt gut.
Aber bei der Arbeit habe ich keine Verbindung zum Internet (und ich glaube, ich bin nicht allein).
Eine Online-Installation ist also per se nichts für mich.

Trotzdem habe ich es zu Hause ausprobiert.
Ich habe die Datei `"setup.exe"` heruntergeladen und sie gestartet.
Nach der **"Online-Autorisierung"** sah ich ein leeres Formular und wusste nicht,
was ich als nächstes tun sollte.
Also wandte ich mich an den technischen Support (*der übrigens großartig ist und mit dem man immer gerne zu tun hat - prompt, genau und sehr menschlich*) und bekam eine Antwort:
Um fortzufahren, muss ich **Delphi** oder **Lazarus** installiert haben.
Aber ich habe **Lazarus** - als portable Installation durch das Dienstprogramm **fpcupdeluxe**.
Das ist für mich ein **Muss**, denn wir sind auf dem Weg von **Windows** zu **Linux** und
die ausführbaren Dateien für jede Anwendung, die ich verwalte, müssen von 3 Arten sein:
Win32, Win64 und **Linux.**

Also wieder zum technischen Support.
Auf ihren Rat hin installierte ich **Lazarus** von einer Distribution (*eine gute Gelegenheit, um einen Blick auf die neue Version 3.0RC1 zu werfen*) und die Einrichtung ging voran (*dauerte fast 2 Stunden, vielleicht war meine Internetverbindung schlecht*). **Redaktionsbemerkung: Bei uns war das ungefähr 10 Minuten.**
Am Ende hatte ich eine Reihe von Ordnern im Ordner Components von Lazarus:

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

Der Rest war ein Kinderspiel, genau wie im oben genannten Artikel beschrieben.
Der Ordner **"Win64"** enthält alle **`.lpk`**-Dateien, die wir benötigen.
Ich habe diese Ordner in die **fpcupdeluxe**-Installation kopiert, die Pakete kompiliert und installiert - und es hat funktioniert!

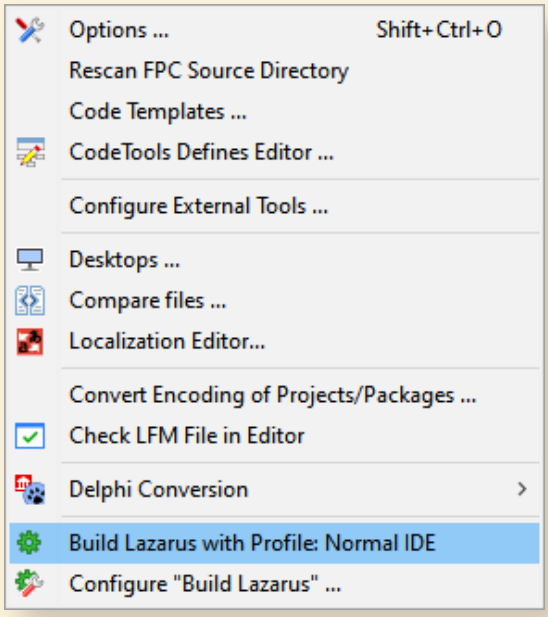
Aber für Windows hat es funktioniert.
Und was ist mit **Linux**?
Ich habe den technischen Support gefragt, und sie sagten, sie würden ein `deb- (und rpm-)` Paket für Linux liefern. Sie haben ihr Wort gehalten (**Version 2023.3**).
Jetzt wird es zusammen mit dem Online-Installationsprogramm ausgeliefert.
Vielleicht ist es wie bei früheren Versionen (**ohne Online-Installation**) wirklich nicht notwendig, wenn Sie die Installationspakete bereits von einer Windows-Installation haben.
Kopieren Sie sie einfach in **Linux** mit der **Lazarus-Installation** und führen Sie die Kompilier-/Installationsroutine aus. Aber jetzt habe ich dieses Paket, also habe ich es heruntergeladen und ausprobiert. Alles ging gut und schnell.
Ich installierte das **Paket fast_report-professional-2023.3.0.deb** (*wir verwenden die Debian-Variante von Linux*) mit apt und erhielt diese Ordner:

```
/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
```

Wie wir sehen, befinden sich alle `.lpk`-Dateien, die wir zum Kompilieren und Installieren benötigen, im Ordner `/usr/share/FastReport-Professional/Lpks`.

Lazarus unter meinem nicht privilegierten Konto weigerte sich, die Pakete zu kompilieren, weil ich keine Schreibrechte im Installationsordner hatte.
Ich konnte den Installationsordner in den `Ordner Lazarus/Components` kopieren, wie ich es immer getan habe, und dann die Eigentümerschaft und die Berechtigungen korrigieren.
Aber ich habe einige **Lazarus**-Installationen zu Testzwecken.
Also habe ich **Lazarus** stattdessen unter **sudo** gestartet.
Nach dem Kompilieren und Installieren der **FR-Pakete** habe ich **Lazarus** unter meinem nicht privilegierten Konto gestartet und alles lief gut.
Was den Artikel betrifft, habe ich einige Anmerkungen.

- Es wird suggeriert, dass die **Professional Edition** über **Client/Server-Komponenten** verfügt". Das stimmt nicht, nur **Enterprise** und **Ultimate** haben Client/Server-Komponenten.
- Klicken Sie auf "**Use**"- das reicht nicht, Sie müssen auch auf "**Install**" klicken.
- Die Installation von `frxrichlazarus.lpk` erfordert die vorherige Installation des Pakets `richmemo`.
  Nach der Installation jedes Packages wird **Lazarus** neu gestartet.
  Lazarus muss nicht nur neu starten, es startet auch neu, nachdem es sich neu aufgebaut hat, und das dauert eine ganze Weile.
  Nach der Installation jedes Packages (*eigentlich nach dem Markieren jedes Packages zur Installation*) fragt **Lazarus**: Jetzt neu aufbauen oder nicht?
  Wenn Sie mit "Nicht" antworten, kann der Neuaufbau nur einmal nach der Installation des letzten Packages durchgeführt werden.



Und eine letzte Bemerkung zum "Online Installer".
Ich denke, es wäre praktisch, eine Option zu haben, mit der Sie eine nicht im System registrierte Lazarus-Installation manuell hinzufügen können, z.B. **fpcupdeluxe** oder **Code Typhon Studio.**

Und jetzt lassen Sie uns etwas Spaß machen.

Sie kaufen es, Sie installieren es und es funktioniert

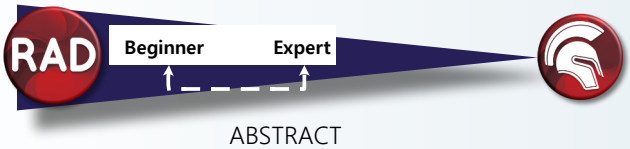
Eines Tages war ich bei der Präsentation einer neuen Version von **FR**.
Am Ende gab es eine Art Auktion. Die Teilnehmer sollten **FR** loben, der **FR**-Direktor bewertete die Äußerung und gab eine Belohnung (oder auch nicht).
Zu den Belohnungen gehörten Kugelschreiber, Flash-Sticks und T-Shirts, alle mit **FR**-Logo.
Die T-Shirts galten als Spitzenpreis und wurden nur hin und wieder vergeben.
Als eine Art Flaute eintrat, sagte ich faul (ohne aufzustehen):
"Ich weiß nicht, warum jemand **FastReport** loben sollte. So ein langweiliger Soft...".
Der Direktor wurde fast apoplektisch und krächzte: "Aber warum?!".
Ich erklärte es ihm: "Wenn Sie fast jede andere Software nehmen und versuchen, sie zum Laufen zu bringen - ist das ein Kampf, harte Arbeit.
Und wenn Sie nach schlaflosen Nächten, literweise Kaffee, Zigarettenschachteln und endlosen Gesprächen eine Software zum Laufen bringen, dann ist das ein Sieg, ein Glücksgefühl!
Und jetzt **FR**.
Sie kaufen es, Sie installieren es und es funktioniert.
Sehr langweilig.".

Der Direktor wäre fast wieder apoplektisch geworden, aber er fasste sich ein Herz und sagte:
"Ein T-Shirt für den Benutzer!".
Jetzt ist mir dieses T-Shirt irgendwie zu klein geworden, aber ich kann es immer noch zeigen.

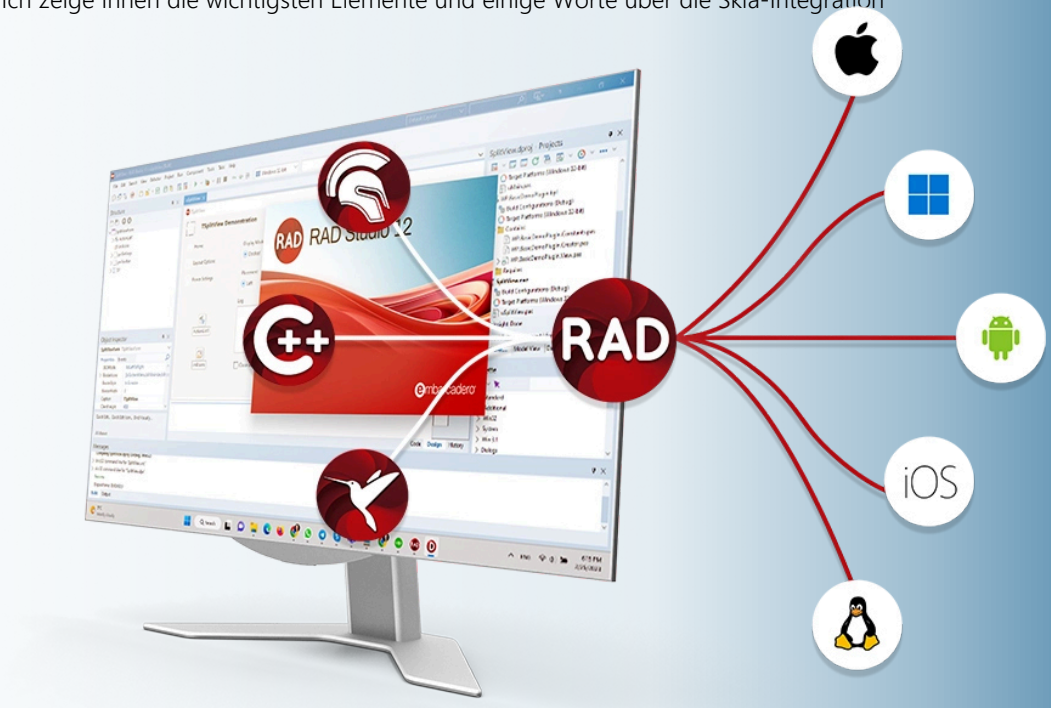
WAS IST NEU IN RAD STUDIO?

DELPHI 12 /. RAD STUDIO 12

Von Detlef Overbeek



ABSTRACT
Dieser Artikel soll Ihnen einen Einblick in die neuesten Funktionen von Delphi geben. Ich zeige Ihnen die wichtigsten Elemente und einige Worte über die Skia-Integration



Plattformversionen, die Rad Studio 12 unterstützt.
Es bietet offizielle Unterstützung für **iOS 17** (nur für **Delphi**), **Android 14** und **macOS Sonoma.** und unterstützt auch **Ubuntu 22 LTS** und **Windows Server 2022**.

Mehrzeilige String-Literale für Delphi-Quellcode.
Mehrzeilige String-Literale ermöglichen eine einfachere Einbettung von mehrzeiligem **SQL-, HTML-, JSON- und XML-Text** in den Quellcode einer Anwendung.
Delphi's Object Pascal erlaubt `String-Literale` und `statische Strings`, die in den Code eingebettet sind. In der Vergangenheit waren diese auf "kurze Strings" mit einer Obergrenze von 255 Zeichen beschränkt. Mit Delphi 12 ändert sich das. `Lange String-Literale`**: Delphi 12 unterstützt jetzt 4K Zeichen pro Zeile**.
Mehrzeilige Zeichenketten: **Delphi 12** führt mehrzeilige Strings ein, die von einem dreifachen Anführungszeichen (''') umgeben sind. Diese Syntax macht es wirklich einfach, mehrzeilige Strings zu verwenden, ohne das '+'-Zeichen verwenden zu müssen.

```
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.
```

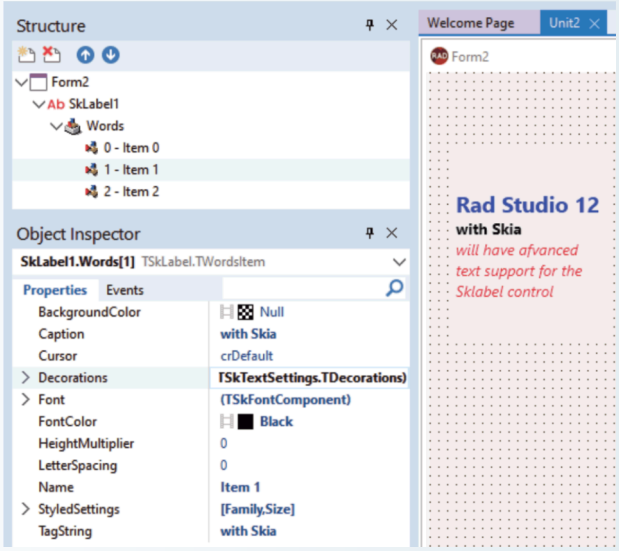
SKIA-Unterstützung für UI-Design in FireMonkey
Es verbessert die Leistung und Qualität beim Rendern von Grafiken und UI-Steuerelementen auf allen Zielplattformen. **FireMonkey** hat schon immer Stile für das UI-Rendering verwendet.
Diese Stile bestimmen das Aussehen und die Funktionalität von UI-Elementen auf verschiedenen Plattformen, einschließlich **DirectX** und **Metal**.
Skia selbst ist für seine Fähigkeiten in 2D-Grafikanwendungen bekannt.
Die von **Google** entwickelte Bibliothek verbessert die Leistung erheblich. Die Benutzer der **Skia4Delphi-Bibliothek** dürften mit ihren Funktionen vertraut sein.

RAD Studio bildet eine Schnittstelle zu Skia, indem es das **Open-Source-Projekt Skia4Delphi** nutzt, enthält aber zusätzliche Funktionen, die in den Open-Source-Projekten nicht zu finden sind, wie den Vulkan-Treiber. Es bietet eine umfassende **2D-API zum Rendern von Bildern für mobile**, Server- und Desktop-Modelle. Es ist mit allen **RAD Studio-Frameworks** (**Console, FMX und VCL**) und Plattformen kompatibel. Es bietet gängige 2D-APIs, indem es die Komplexität der Implementierung von dahinter liegenden **Low-Level-Bibliotheken wie OpenGL, Vulkan*** (*siehe nächste Seite*), **DirectX** oder **Metal** abstrahiert und Optimierungen und neue Funktionen implementiert. In der Skia-Dokumentation finden Sie alle wichtigen Funktionen und erfahren, wie Sie die Skia-Integration aktivieren können

Skia-basierte UI-Steuerelemente
Die Integration der **Skia**-Bibliothek bietet auch einige neue spezifische native Steuerelemente und Komponenten. Diese Steuerelemente sind nur verfügbar, wenn **Skia** aktiviert ist und auf der Zielplattform eingesetzt wird.

`TSkAnimatedImage:`
das Steuerelement, das animierte Bilder lädt und rendert, einschließlich Vektoranimationen.

`TSkLabel:`
implementiert neue Funktionen, die entweder von TLabel nicht unterstützt wurden oder schwierig zu implementieren waren, wie z.B.:
Schriftfamilien; (*Font-Fallback-Liste wie in CSS*)
Schriftstärke und Neigung;
Unterstützung für mehrere Stile im Text;
Unterstützung für BiDi (Rechts-nach-Links);
Unterstützung für die horizontale Ausrichtung;
Unterstützt Hintergrundfarbe für Teile des Textes;
Option für automatische Größe;
Erweiterte Dekorationen (Unterstreichen, Überstreichen, gestrichelte Linie und mehr).



SKIA

* Vulkan-Backend

Skia mit RAD Studio bietet Vulkan-Backend-Unterstützung für Android und optional auch für Windows.

Skia Canvas mit RAD Studio nutzt automatisch Vulkan auf Android, wenn es unterstützt wird, was zu einer verbesserten grafischen Leistung und Energieeffizienz im Vergleich zu OpenGLES führt.

Um Vulkan unter Windows zu aktivieren, wenn es unterstützt wird, setzen Sie den booleschen Wert FMX.Types.GlobalUseVulkan auf True und den booleschen Wert FMX.Skia.GlobalUseSkiaRasterWhenAvailable auf False im Initialisierungsabschnitt.

```
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;
  ...
```

HINWEIS: Die Präferenz für das Vulkan-Backend auf Android kann mit demselben Booleschen Wert GlobalUseVulkan deaktiviert werden, der auch für die Aktivierung unter Windows verwendet wird. Um sie zu deaktivieren, sollte sie jedoch auf False gesetzt werden.

TSkPaintBox:
Steuerelement zum Malen mit Skia APIs direkt auf dem Bildschirm mit dem OnDraw-Ereignis.
TSaAnimatedPaintBox:
ermöglicht das Einstellen der Dauer einer Animation und das Zeichnen des Verlaufs dieser Animation mit Skia-APIs über das OnAnimationDraw-Ereignis.
TSkSVG: Steuerelement zur Anzeige von 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

Die folgenden Skia-Demos sind in der Installation enthalten.
Skia Demo FMX - RAD Studio
https://docwiki.embarcadero.com/RADStudio/Athens/en/Skia_Demo_FMX
Standort
Sie finden das Skia Demo FMX Beispielprojekt unter:
Start > Programme > Embarcadero RAD Studio 12 > Samples und navigieren Sie zu:

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

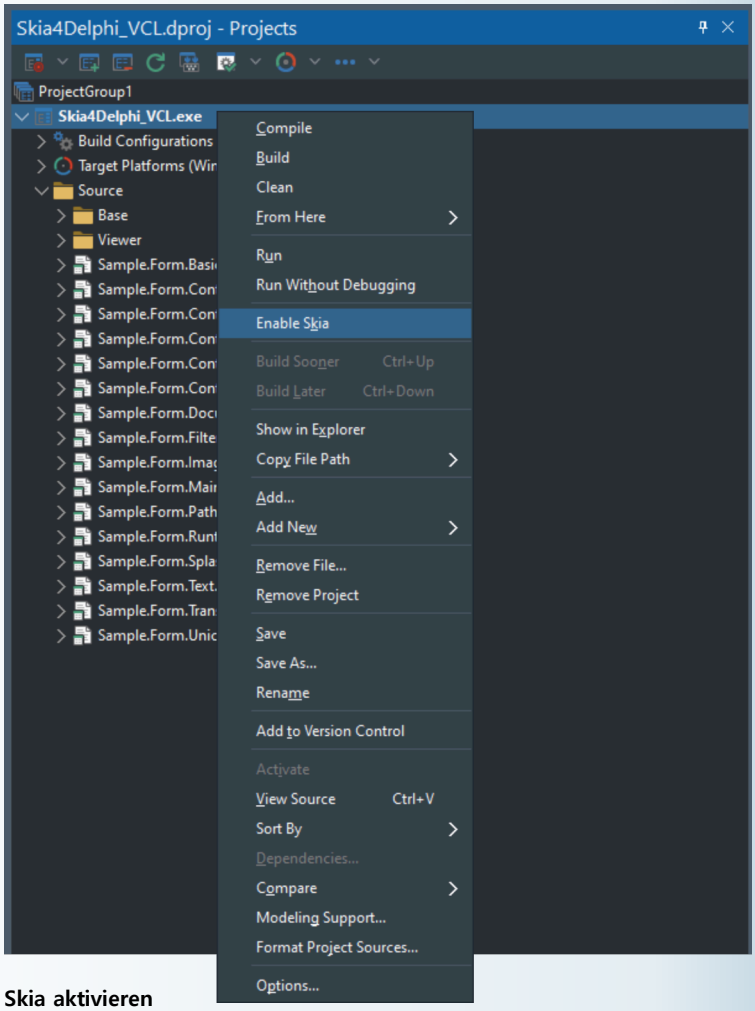
Skia4Delphi - RAD Studio

- Nach oben zu den Software-Add-Ins von Drittanbietern: (Siehe den Standardpfad unten)

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

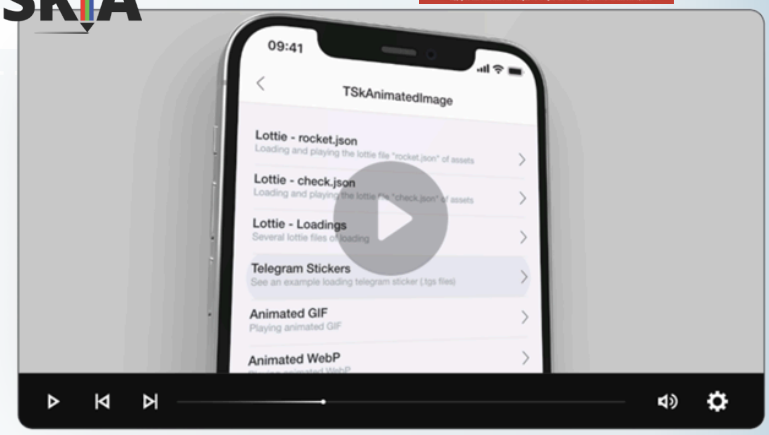
Obwohl die Bibliothek mit allen Frameworks kompatibel ist, sind einige Funktionen einzigartig:

Feature set	Console	VCL	FMX	Description
Skia API	✓	✓	✓	Zugriff auf die reine Skia-Bibliothek über eine einzige Unit: System.Skia.pas (oder System.Skia.hpp)
Controls		✓	✓	TSkAnimatedImage, TSkLabel, TSkPaintBox und TSkSvg
App render			✓	Ersetzung der FMX-Grafik-Engine durch Skia

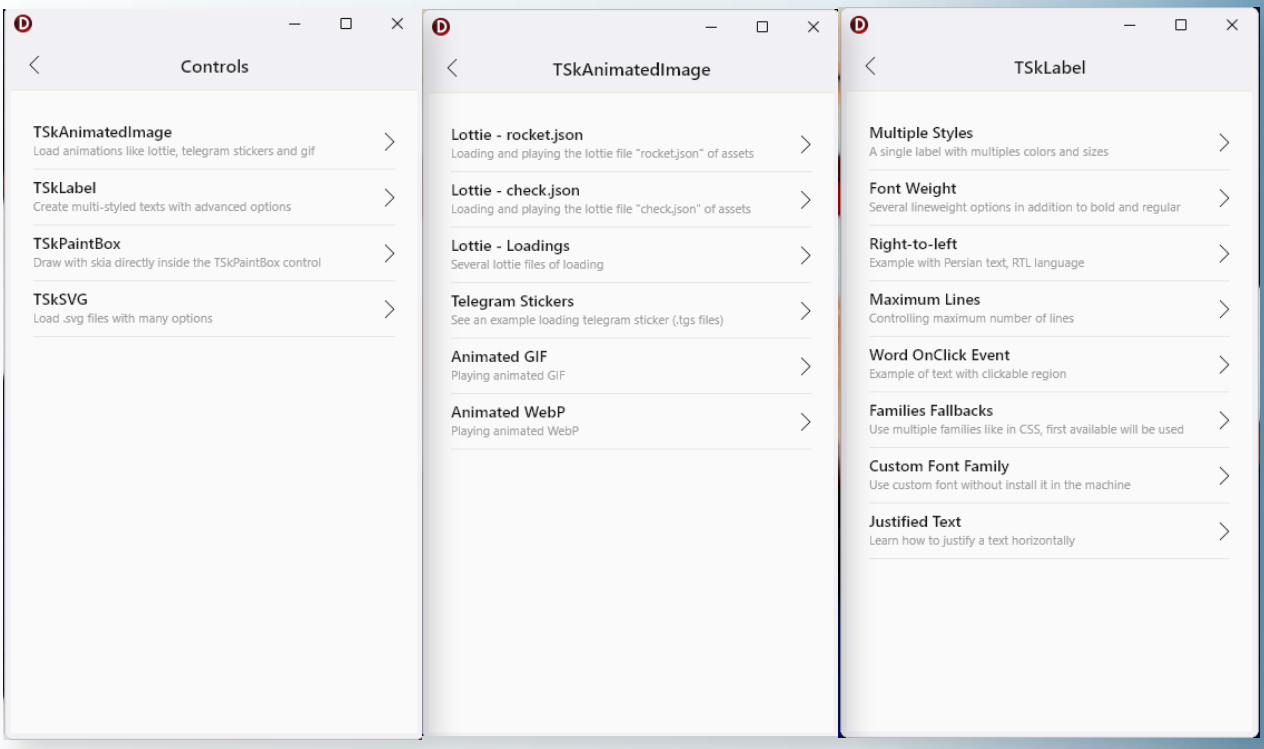


Skia aktivieren

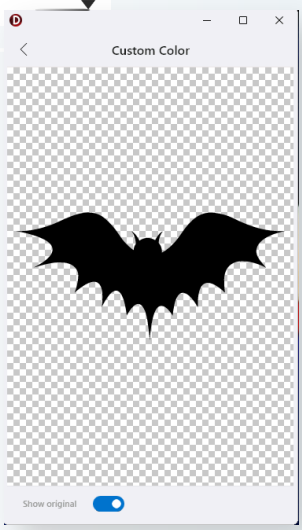
Bevor Sie eine Skia-Funktion oder -Unit verwenden können, müssen Sie Ihr Projekt für die Verwendung von Skia aktivieren. Öffnen Sie dazu Ihr Projekt in RAD Studio und klicken Sie mit der rechten Maustaste auf Ihr Projekt und dann auf Skia aktivieren:
Diese Aktivierung wird nur für Anwendungen vorgenommen. Pakete, die Skia verwenden, müssen diese Aktivierung nicht vornehmen.
Dieser Schritt konfiguriert automatisch die Verknüpfungen und Bereitstellungen der Bibliotheks-Binärdateien für das Projekt. Wenn Sie Skia nicht aktivieren und seine Units verwenden, wird Ihre Anwendung beim Start Probleme haben.

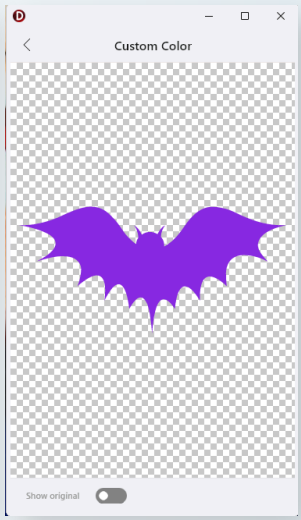


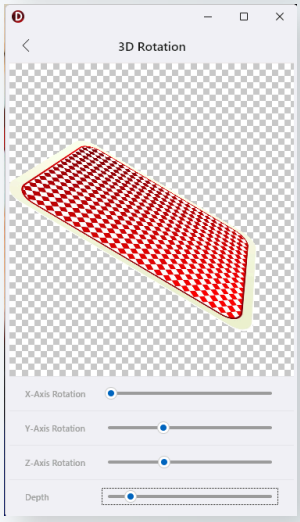
Die Standardanwendung, um die Beispiele zu zeigen

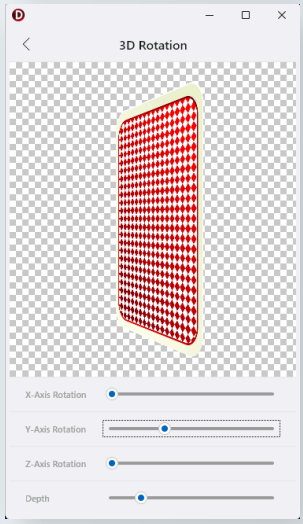


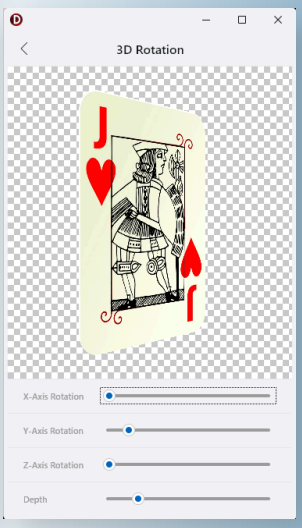
SKIA

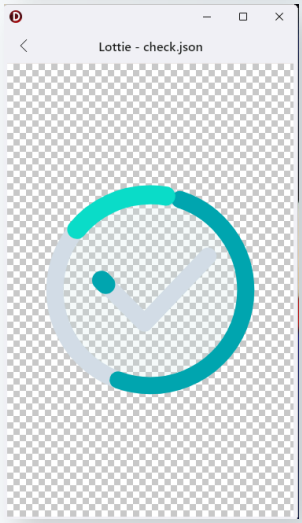




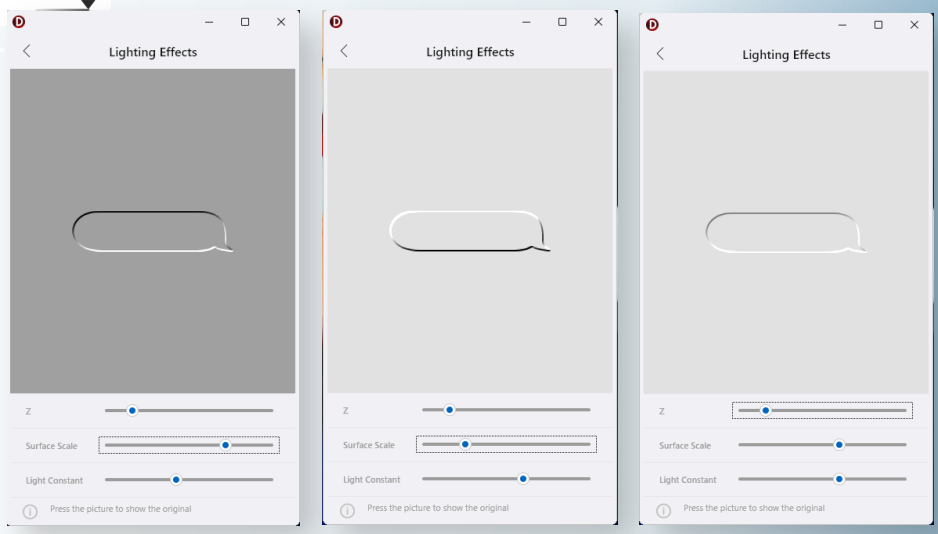


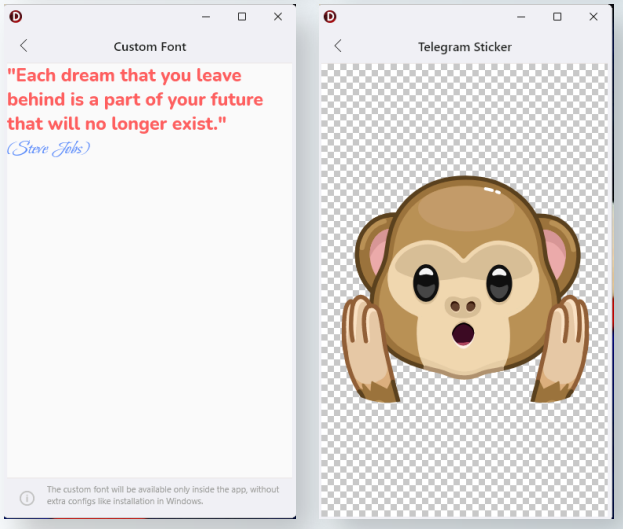


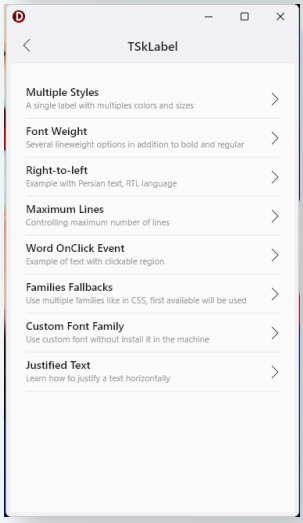




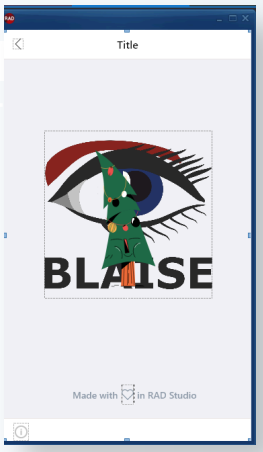
In der nächsten Ausgabe werde ich einige zusätzliche Beispiele erstellen, um Ihnen ein kostenloses Codebeispiel zur Verfügung zu stellen.


Lighting Effects
Z
Surface Scale
Light Constant
Press the picture to show the original

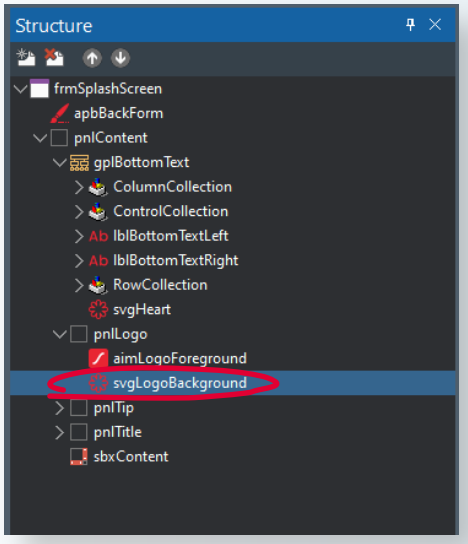

Custom Font
"Each dream that you leave behind is a part of your future that will no longer exist."
(Steve Jobs)
The custom font will be available only inside the app, without extra configs like installation in Windows.
Telegram Sticker

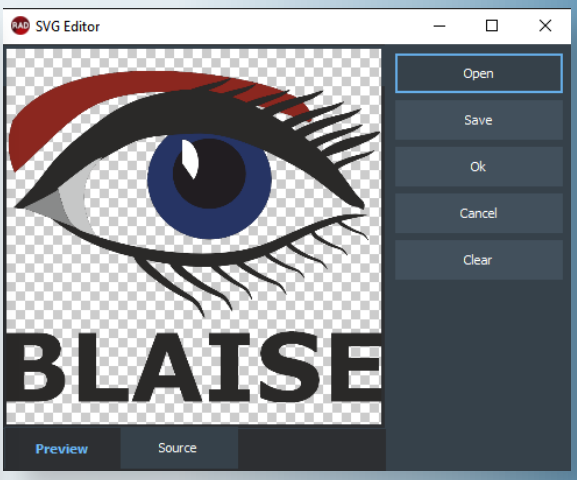

TSkLabel
Multiple Styles
A single label with multiples colors and sizes
Font Weight
Several lineweight options in addition to bold and regular
Right-to-left
Example with Persian text, RTL language
Maximum Lines
Controlling maximum number of lines
Word OnClick Event
Example of text with clickable region
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

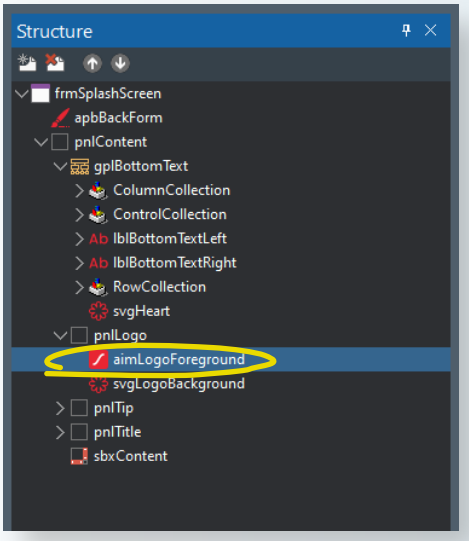


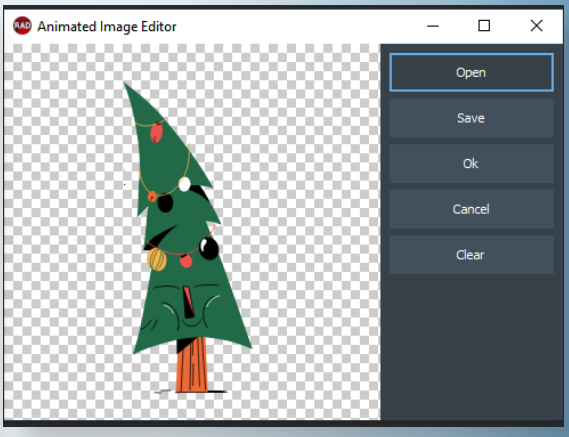
Um Ihnen einen schnellen Einblick in das Projekt zu geben, kann es interessant sein, ein paar Änderungen vorzunehmen und diese zu verwenden.
Ohne auch nur ein Stück Code zu schreiben, habe ich meinen eigenen Startbildschirm erstellt und einen Weihnachtsgruß erstellt. Eigentlich besteht dieser Splash aus zwei Hauptelementen: einem Hintergrund und einem sich bewegenden Vordergrund. Um die .svg zu erstellen, musste ich einige Tricks anwenden: Ich habe aus dem Vektordesign des Auges eine Bitmap gemacht und sie dann wieder in eine .svg umgewandelt. Andernfalls wäre die SVG nicht mit all den Ebenen und Farben umgewandelt. Außerdem habe ich die Bewegung im Vordergrund mit einem bestehenden Lottie-Beispiel geändert.
Ich habe das Beispiel des Codes auf die nächsten beiden Seiten gestellt, damit Sie einen Eindruck von der Schwierigkeit, dies zu handhaben, bekommen.





Background SVG





Sie können Ihr Beispiel von Ihrer persönlichen Download-Adresse herunterladen.

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;
```

SKIA

```
 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.
```


SKIA

RAD Studio 12

Wie setzen wir Skia ein?

"**Skia4Delphi** ist eine plattformübergreifende 2D-Grafik-API für Delphi und C++ Builder, die auf der **Skia Graphics Library** von **Google** basiert. Bietet gemeinsame **2D-APIs** durch Abstraktion der Komplexität bei der Implementierung der dahinter stehenden **Low-Level-Bibliotheken wie OpenG L, Vulkan, DirectX und Metal und anderen.**

SKIA

RAD Studio 12

Hauptmerkmale von Skia (1/2)

Funktionen	Details
2D-Zeichnung	Formen, Pfade und Texte
SVG	Rendering und -Erstellung
Bilddekodierer	`BMP`, `GIF`, `ICO`, `JPG`, `PNG`, `WBMP`, `WEBP` und `Rohbilder`
Bildkodierer	`PNG`, `JPG` und `WEBP`
Animation Player	**Lottie, Telegram Sticker**, animierte **GIFs** und animiertes **WEBP**
Anti-Aliasing	Hohe Zeichenqualität, keine gezackten Kanten
Schriftart	Schriftstärke, Familien, Tailbacks und benutzerdefinierte Schriftart (ohne Installation), Ligatur

RAD Studio 12

Hauptmerkmale von Skia (2/2)

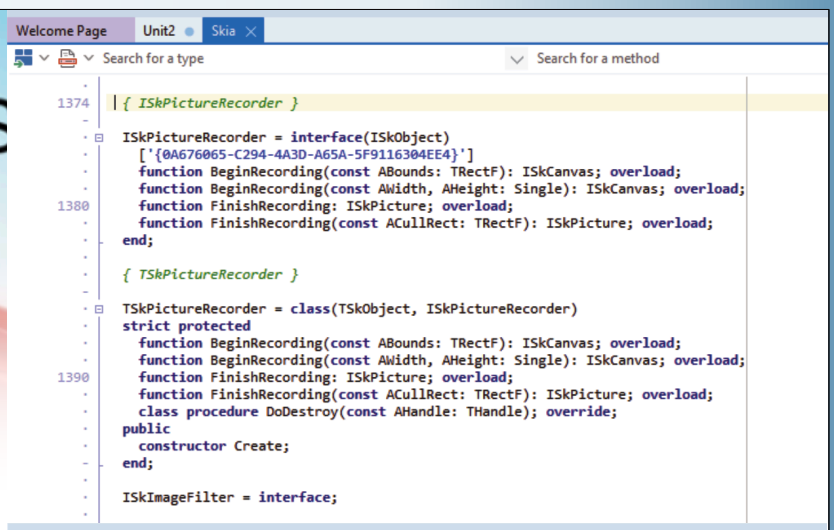
Funktionen	Details
Text	Mehrere Stile, maximale Zeilenanzahl, Zeilenabstand, Blocksatz, Textumriss, Farbverlauf und Verzierungen
Rechts-nach-links	Sprachen Rendering von Texten in Persisch, Arabisch, Hebräisch usw.
PDF	Generierung von vektorisiertem PDF
Unicode	Graphem-Parser
Filter	Farb-, Masken- und Bildfilter
Clippings	Unterstützung für viele erweiterte Beschneidungsoperationen wie Pfade und Shader
Gradients (Farbverläufe)	Lineare, radiale, gebogene und konische Farbverläufe
Shader	Erstellung von Shadern zur Ausführung bestimmter Zeichnungen direkt auf dem Grafikprozessor, über eine einzige Shader-Sprache (SkSL)

Skia in RAD Studio

Unterschiedliche Nutzungsebenen für Delphi und C++Builder

1. SkiaAPI

Zugriff auf die reine Skia-Bibliothek, über eine einzige Einheit:
Skia.pas oder Skia.hpp



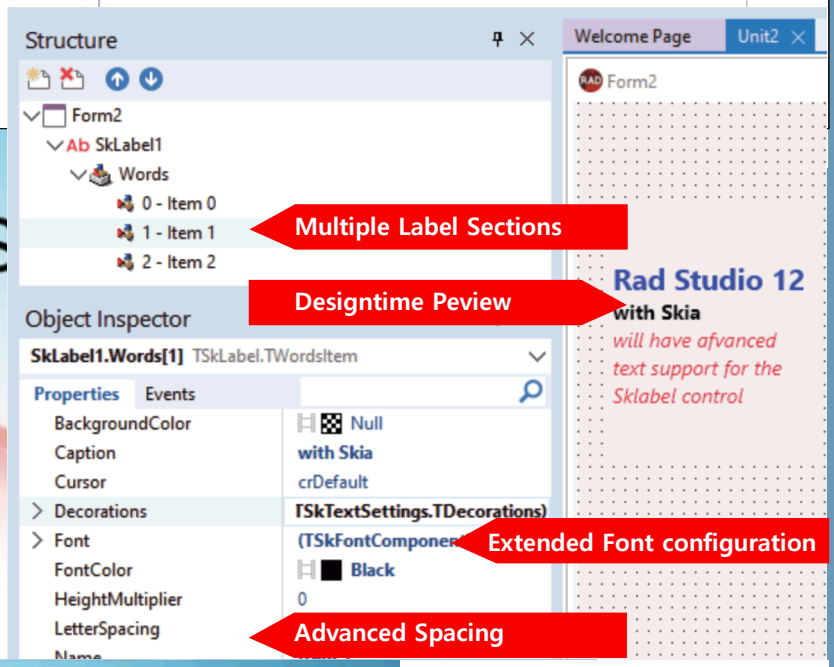
Skia in RAD Studio

Unterschiedliche Nutzungsebenen für Delphi und C++Builder

2. UI-Steuerelemente

(sowohl für FMX als auch für VCL)

- TSkAnimatedImage
- TSklabel
- TSkPaintBox
- TSkSvg

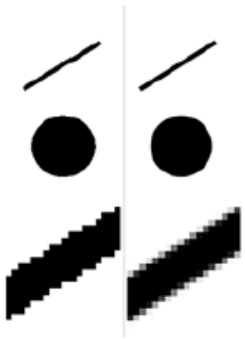


Skia in RAD Studio

Unterschiedliche Nutzungsebenen für Delphi und C++Builder

3. Rendering von Anwendungen und Stilen

Ersetzen der FMX Grafik-Engine durch Skia. Bessere Leistung, verbessertes Antialiasing
FMX.Skia.GlobalUseSkia := True
Verbessertes Zeichnen mit Antialiasing in Skia (rechts)



Verbessertes Zeichnen mit Antialiasing

RAD Studio 12

Skia in RAD Studio

Unterschiedliche Nutzungsebenen für Delphi und C++Builder

4. Codecs für Bildsteuerungen

Neue Bildcodecs, die Skia unterstützt, sind in den Frameworks registriert. FMX- oder VCL-Bildsteuerungen können neue Bildformate wie WebP direkt laden und speichern.

RAD Studio 12

Neue UI-Steuerungen (FMX/VCL) basierend auf Skia

- **TSkAnimatedImage**
  - Unterstützt Lottie-Datei, Telegram-Sticker, Animiertes GIF, Animiertes WebP
- **TSklabel**
  - Schriftstärke, Schriftneigung, mehrere Stile im Text, BiDi (Rechts-nach-Links)
  - Horizontale Ausrichtung ausrichten, und vieles mehr
- **TSkPaintBox & TSkAnimatedPaintBox**
  - Malen mit Skia APIs direkt auf dem Bildschirm mit dem On Draw Ereignis
- **TSkSVG**
  - SVG einfach anzeigen

RAD Studio 12

Skia and FireMonkey

Warum Skia?

- Skia ist eine Multiplattform 20 Grafikbibliothek
- Skia bietet hohe Leistung und hohe Qualität beim Rendering
- Skia unterstützt fortgeschrittene Grafikoperationen auf allen Plattformen
- Skia bietet Unterstützung für viele Bild- und Videoformate
- Skia wird die neue Grundlage für FireMonkey Rendering

SKIA

RAD Studio 12

Was gibt es Neues in FireMonkey?
Wirklich eine Menge!

- Unterstützung der Skia Graphic Library Bibliothek
  - Für alle Zielplattformen, sowohl Delphi als auch C++Builder
- Fertigstellung der Neugestaltung von TEdit/TMemo mit Stil
- Unterstützung für Android API 33 (siehe später)
- Neuer Icon- und Splash-Screen-Designer (früher behandelt)
- Geteilte Ansichten für mobile Plattformen
- Plus kleinere Funktionen und Qualität

RAD Studio 12

FireMonkey Skia Unterstützung für alle Plattformen

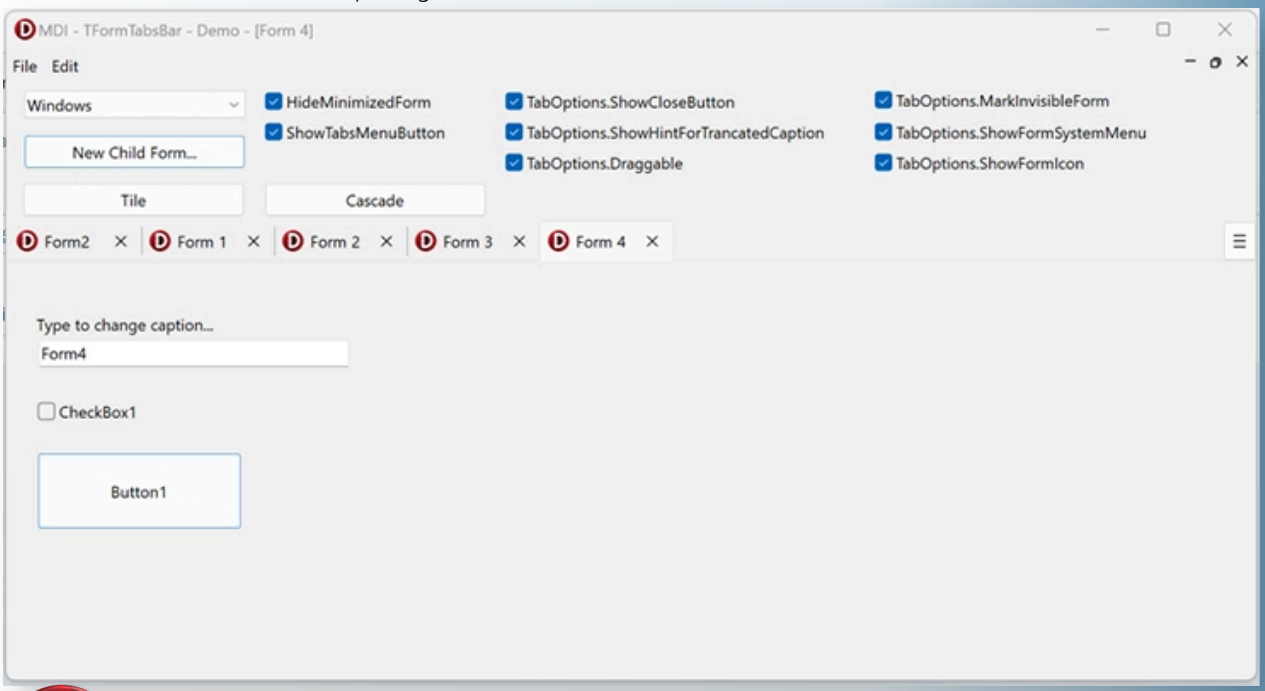
Sowohl für C++Builder als auch für Delphi,
FireMonkey, aber auch VCL
Nutzung des Open-Source-Projekts Skia4Delphi
Einschließlich zusätzlicher Funktionen, die im Open-Source-Projekt nicht enthalten sind:

- Vulkan-Unterstützung
  - Verbesserte grafische Leistung und Energieeffizienz auf Android im Vergleich zu OpenGLES
- Skia Shading Language (SKSL) für Effekte und Filter
- WebP Encoder
- Native Drucker für Windows und PDF-Druck für alle Plattformen

Wird auch für den neuen Icon- und Splash-Assistenten verwendet

Verbessertes mobiles Design mit FireMonkey-Verbesserungen
Verbesserungen bei der Unterstützung von Android-Plattformen, geteilter Bildschirm für iOS und Android, Assistent für das vollständige Einstellen von Symbolen und Startbildschirmen, Unterstützung für Android API Level 33

Modernisierte VCL mit überarbeiteter MDI und Registerkarten-UI für VCL
Verbesserte Anwendungsmodernisierung mit Unterstützung für HighDPI und neue VCL-Designer, die von Konopka Signature VCL Controls stammen

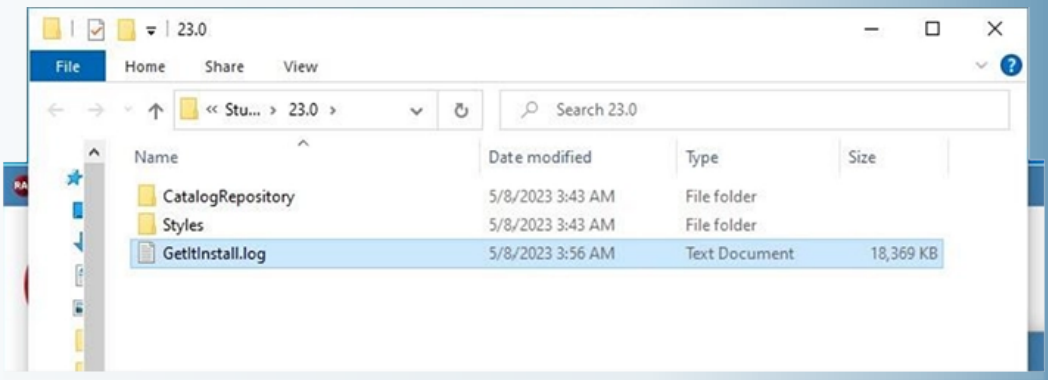


Mehr Windows-APIs bereit zur Verwendung in Object Pascal
Umfassende Sammlung aller Windows-API-Header, die nach Object Pascal konvertiert wurden, um es Delphi-Entwicklern zu erleichtern, jede Windows-Plattform-API aufzurufen


RAD Studio 12
File Edit Search View Refactor Project Run Component Tools Tabs Help
Welcome Page
Structure Object Inspector
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\Project5Splash\
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.
Edit Layout

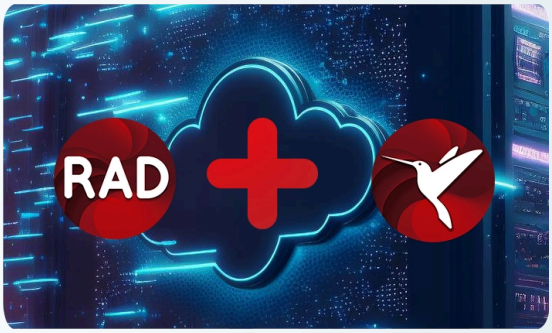

Startup Layout
Extend the IDE
Manage Features
GetIt Package Manager
Skia4Delphi_VCL.dproj
C:\Users\Public\Documents\Embarcadero\Studio\23.0\Samples\Object Pascal\VCL\Skia4Delphi\
Project1.dproj
D:\SPP\Blaise\Blaise_UK_114_115_2023\Authors\David Dirkse\stepbystep-2\
PGGLedenAdmin.dproj
F:\Nieuwe Ledenadmin\
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...
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
Palette Projects Model View Data Explorer Multi-Device Preview

Mehr Windows-APIs bereit zur Verwendung in Object Pascal
Umfassende Sammlung aller **Windows-API-Header**, die nach Object Pascal konvertiert wurden, um es Delphi-Entwicklern zu erleichtern, jede Windows-Plattform-API aufzurufen.



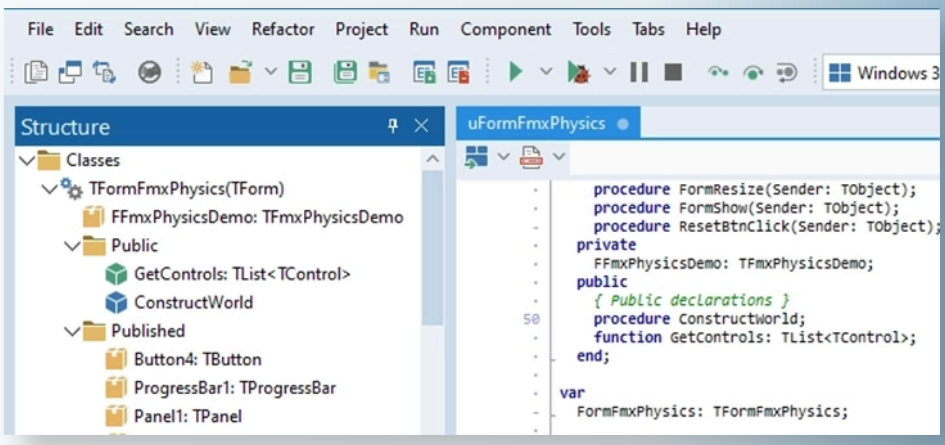
QBE Support in FireDAC,
Neuer **JSON Wizard für Delphi, Query-by-Example in FireDAC** verfügbar. **JSON** data mapping Wizard zum Generieren von Klassen, die der JSON-Datenstruktur entsprechen, zum Zuordnen von Daten zu Objekten wie XML und zum Ausgeben in eine neue Datei

Nutzen Sie Biometric Authentication!
RAD Studio 12 bietet eine neue Komponente zur mobilen biometrischen Authentifizierung für **FireMonkey-Mobilanwendungen.**

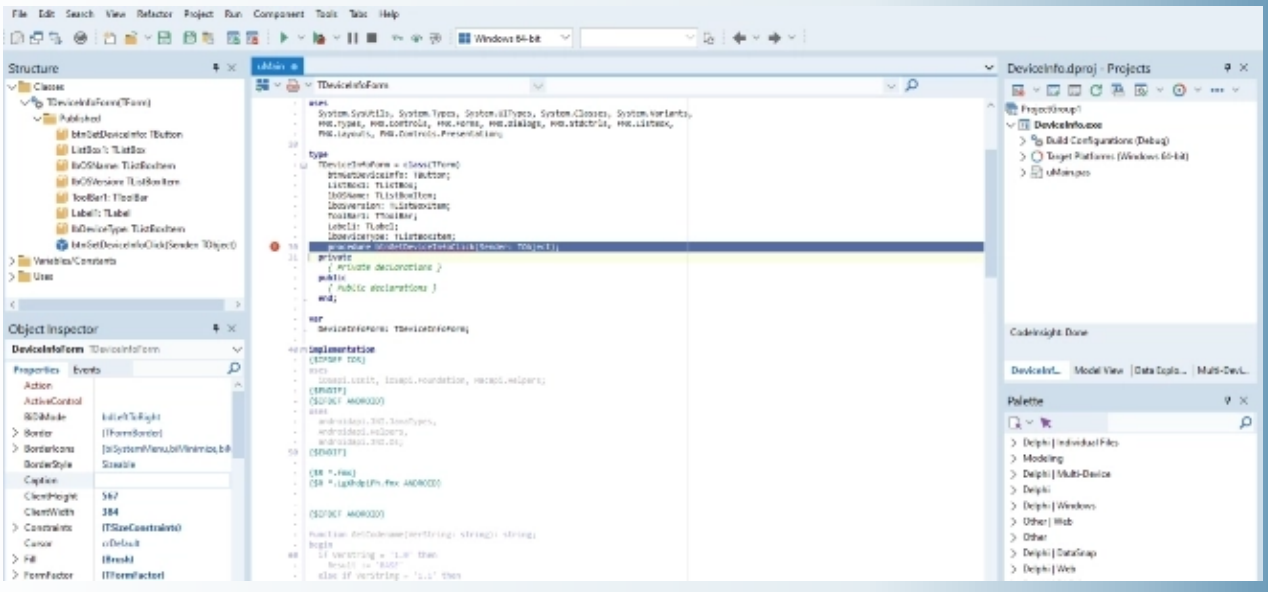


Stellen sie Embedded InterBase Dev Edition bereit!
RAD Studio 12 wird mit der kürzlich veröffentlichten InterBase 2020 Update 5 Developer Edition und der IBLite/ToGo-Edition ausgeliefert.

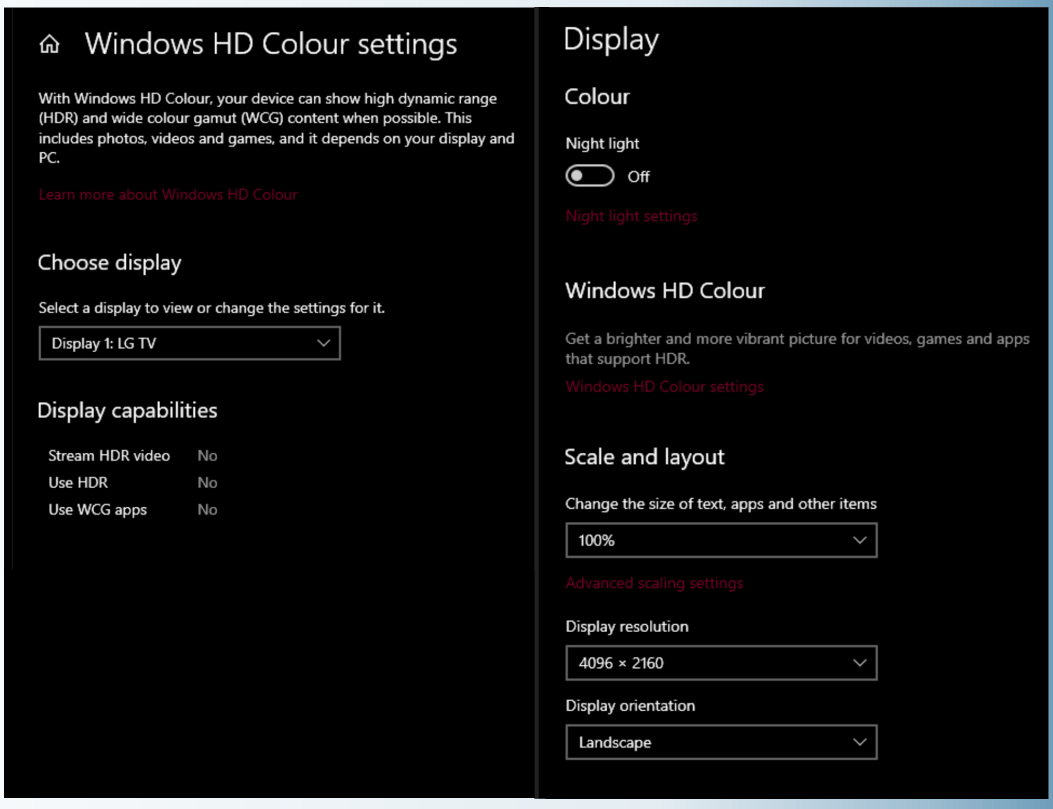
Verbesserte Application Security Through SQL Restrictions
Höhere Anwendungssicherheit durch **Einschränkungen für SQL-Befehle,** Sperren für mehrere Befehle und SQL-Änderungen.



Unterstützung für Smart IDs in RAD Server
Leistungsfähigere und flexibler gehostete **REST-APIs mit neuen Smart-IDs** (Sqids).
Bessere Leistung, verbessertes Paging von Daten, bessere Sitzungsauthentifizierung.



Verwenden Sie RAD Studio auf 4k+ Bildschirmen!
RAD Studio 12 bietet High-DPI-Unterstützung für die IDE, so dass Entwickler auf größeren, hochauflösenden Bildschirmen arbeiten können. Die volle Unterstützung für die neuesten hochauflösenden 4k+ Monitore verbessert die tägliche Arbeit von Entwicklern durch sauberere, schärfere Schriftarten und Symbole sowie hochauflösende Unterstützung in allen IDE-Fenstern, einschließlich der VCL- und FMX-Formulardesigner und des Code-Editors



RAD Studio 12

VCL: Fonts und Screens

Überarbeitete Unterstützung für VCL TFont-Skalierung unabhängig von der DPI-Skalierung

- **`TFont.Size property`** passt sich jetzt an verschiedene DPIs an
- **`TFont class`** neue Eigenschaften und Methoden:
  - **`property IsDPIRelated: Boolean`** *// ermöglicht die Verwendung der Eigenschaft TFont.PixelsPertnch*
  - **`property IsScreenFont: Boolean`** *// für globale Schriftarten in der Klasse TScreen*
  - **`procedure ChangeScale`** *// zum Initialisieren und Skalieren aller DP-bezogenen Schriften aufgerufen*
  - **`procedure ScalefForDPI`** *// im Code verwendet, um eine Schrift an jede DPI anzupassen*

Options

IDE
Default Folders
Compiling and Running
Build Log
Component Toolbar
Environment Variables
File Association
Desktop and Layout
Welcome Page
Project Upgrading
LiveBindings
Saving and Recovering
GetIt Package Manager
User Interface
Object Inspector
Palette
Colors
Difference Viewer
Merge Viewer
Reopen Menu
IDE Style
Structure
Form Designer
FireUI Live Preview
Device Manager
High DPI
Editor
Line Endings
Search
Tabs
Status Bar
Language
Color
Structural Highlighting
Display
Key Mappings
Language
Toxicity Metrics
Delphi
C++
HTML
Formatter
Version Control
Git
Mercurial
Subversion
Deployment
Connection Profile Manager
Provisioning
SDK Manager
Modeling
Debugger
Visualizers
Disassembly
Event Log
Embarcadero Debuggers
Language Exceptions
Native OS Exceptions

High DPI

VCL Designer High DPI mode *
Automatic (Screen PPI)

Custom Design PPI
96

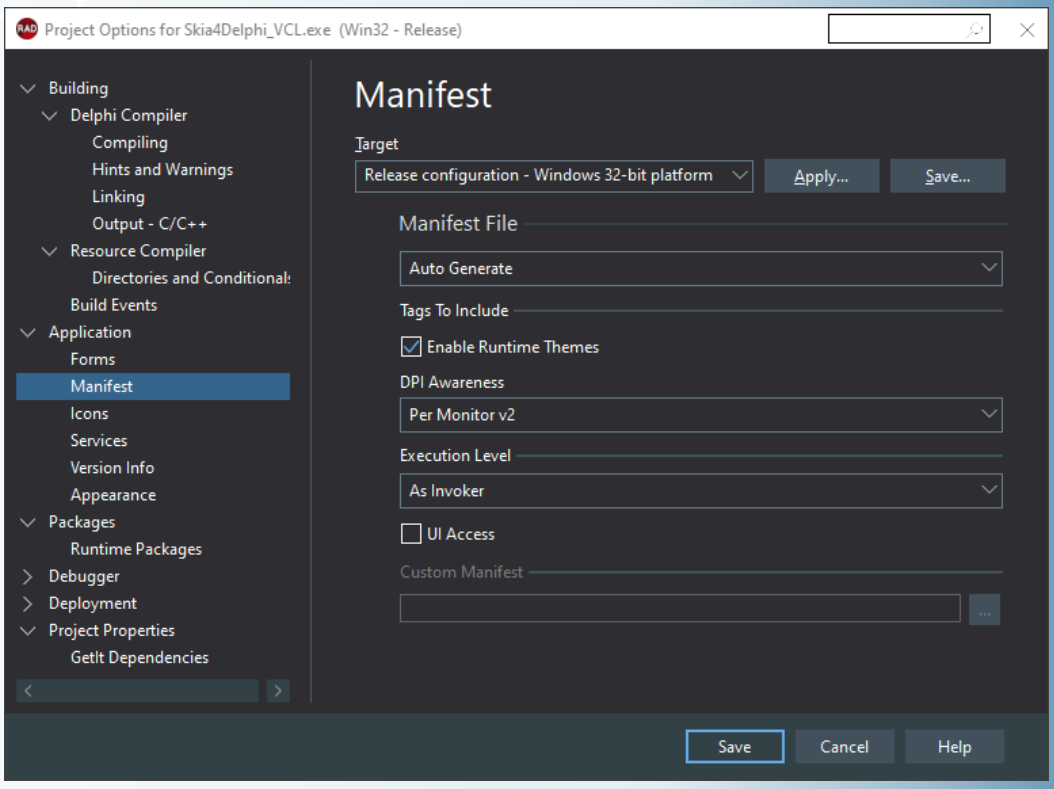
Scale grid size / snap tolerance to design PPI *

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

Save
Cancel
Help

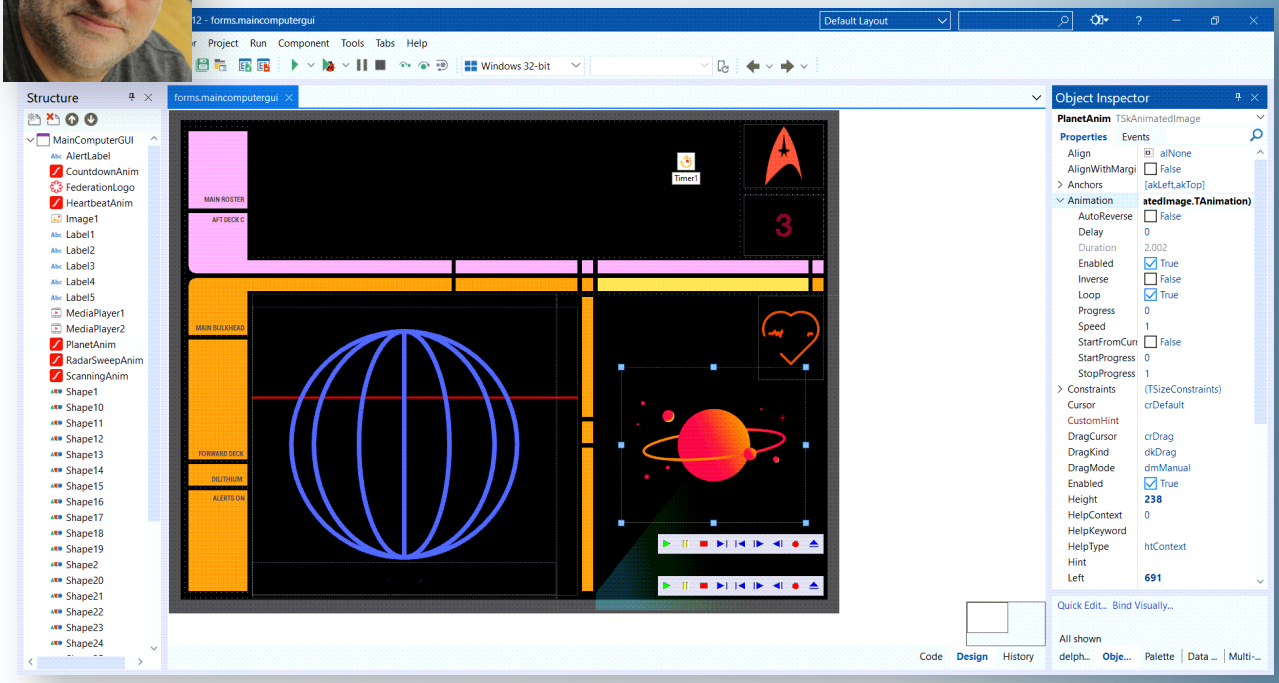


Ziel Windows 11
Offizielle Unterstützung für die Bereitstellung von **Windows 11** mit integrierter **MSIX-Generierung**.
Webbrowser-Komponente für **Windows**, mit Unterstützung für **IE ActiveX** und das neue Microsoft WebView 2-Steuerelement (**Chromium-basiertes Edge**).
Verbesserter VCL-Form-Designer zur visuellen Erstellung nativer Windows-Anwendungen mit **Live-Snap-to**-Hinweisen und Layout-Richtlinien.
Verbessertes Delphi und C++ RTL für 32-bit Windows und 64-bit Windows.

Verwenden Sie VCL Styles zur Entwurfszeit!
VCL Styles bietet jetzt Unterstützung für die Entwurfszeit:
Entwerfen Sie elegante Benutzeroberflächen noch schneller, indem Sie bereits zur Entwurfszeit sehen, wie Ihre gestylten Formulare und Steuerelemente bei der Ausführung aussehen werden. Wenn Sie zur Entwurfszeit sehen, wie sich die Stile zur Laufzeit auf die Benutzeroberfläche auswirken werden, verbessert dies den Entwurfs- und Testprozess für moderne Benutzeroberflächen. Die schnellere Erstellung besserer Benutzeroberflächen ist besonders nützlich, wenn Sie mit Stilen pro Steuerelement arbeiten.

Bereitstellen auf dem M-Series Apple Silicon!
Kompilieren Sie für macOS (M-Serie Apple Silicon) und verwenden Sie das neue Universalpaket für die Einreichung im AppStore. Sie können jetzt sowohl für bestehende Intel- als auch für neue macOS-Prozessoren der M-Serie (Apple Silicon) kompilieren. Die Kompilierung für die neuesten Prozessorversionen ermöglicht die schnellste Leistung auf allen Plattformen und unterstützt die universelle Paketierung für den macOS App Store.

Arbeiten Sie aus der Ferne zusammen!
Verbesserte Remote-Desktop-Unterstützung für VCL und IDE, die Entwicklern hilft, die vom Büro aus arbeiten. Verbessertes Debugging für entfernte und lokale 64-Bit-Windows-Anwendungen und macOS 64-Bit-Anwendungen (Intel und ARM). Die verbesserte Remote-Desktop-Unterstützung steigert die Effizienz Ihres Teams und verbessert Ihr Geschäftsergebnis.



Den Code für dieses Projekt finden Sie unter
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 mal 12 neue Funktionen in Delphi 12

RAD Studio 12 enthält einige großartige Verbesserungen für C++Builder und das Einführungs-Webinar und andere Online-Inhalte heben dies hervor.
Aber auch für **Delphi-Entwickler** ist es ein fantastisches Release. Ich habe 12 Listen mit jeweils 12 Verbesserungen für **Delphi 12** zusammengestellt.
Es handelt sich also nicht um eine Liste mit 12 Verbesserungen für **Delphi 12.**
Es ist eine **Liste mit 12x12=144 Verbesserungen**, plus ein halbes Dutzend für die native Windows-Version, so dass sich die **Gesamtzahl auf satte 150 beläuft** -- ohne die bestehenden Verbesserungen für C++Builder, da ich hier die **Delphi**-Seite betonen möchte (*aber die meisten der unten aufgeführten Funktionen sind tatsächlich für beide Sprachen*).
`https://blogs.embarcadero.com/3-x-12-vcl-enhancements-in-delphi-12/.`
Im ersten Blogbeitrag ging es um 3 x 12 VCL-Verbesserungen in Delphi 12, wie Sie unter
`https://blogs.embarcadero.com/3-x-12-firemonkey-and-android-enhancements-in-delphi-12/`
Der zweite Blogbeitrag konzentrierte sich auf FireMonkey und die Unterstützung der Android-Plattform, wie Sie hier sehen können.
Der dritte Teil enthielt drei Listen, die sich auf drei verwandte Bereiche konzentrierten**, Delphi Runtime Library (RTL)**, Datenbankzugriff und Internetzugang, wie Sie sehen können unter
`https://blogs.embarcadero.com/3-x-12-rtl-data-and-internet-enhancements-in-delphi-12/`
Dieser letzte Blog-Beitrag der Serie enthält jeweils 12 Einträge für die IDE,
das Installationsprogramm und die Sprache Delphi.
Dazu kommen 6 Einträge zur Windows-API-Integration, womit sich die Gesamtzahl auf 150 erhöht.

In VCL: MDI und Forms-Verwaltung

- MDI überarbeitet für HighDPI und VCL Styles Unterstützung
- Child-Formulare können jetzt einen neuen modernen, flachen Rahmen haben (die neue Eigenschaft ist `TStyleManager.ChangeChildFormSystemBorder`)
- MDI Child deaktivierte Rahmensymbole werden nicht gezeichnet
- Die Parent-Eigenschaft funktioniert für die Verschachtelung eines beliebigen Formulars in einem anderen Formular, mit vollständiger Rahmenverwaltung
- Erhebliche MDI-Bereinigung und Verbesserungen
- Das brandneue `TFormsBar` Steuerelement
- Automatisches Ausblenden von minimierten Unterfenstern
- Die Schnittstelle `IFormVisualManager`
- Die Eigenschaft VisualManager der Klasse `TCustomForm`
- Der aktualisierte MDI-Assistent
- Eine neue ShowInTaskbar-Eigenschaft für `TForm`
- Ein neuer `CreateScaledNew` Konstruktor in der `TCustomForm` Klasse

VCL-Steuerelemente

- Unterstützung der Kachelansicht für `TListView` (einschließlich der neuen Eigenschaften `TileOptions` und TileColumns) `TGroupCollection` hat jetzt zwei Items-Eigenschaften
- Neuer ToolButton-Stil: `tbsWholeDropDown`
- Das `TNumberbox` Steuerelement hat einen zusätzlichen "nbmInt64" Modus, der 64-bit Zahlen als Eingabe akzeptiert
- ActivityIndicator hat Unterstützung für benutzerdefinierte Farben (Eigenschaft IndicatorCustomColor), neue vordefinierte Icons RotatingLines und Refresh und weitere Erweiterungen
- `TControlList` verfügt über die neuen Eigenschaften `SelectedItemsCount` und `SmoothMouseWheelScrolling`
- `TControlList` bietet Unterstützung für weitere Elementtypen wie `TControlListCheckBox` und `TControlListRadioButton`
- TWICImage und TImageCollection haben jetzt eine Dormant()-Methode, um die GDI-Nutzung zu reduzieren
- Die neue `TSkLabel` Skia-basierte Komponente für VCL
- Die neue `TSkPaintBox` Skia-basierte Komponente für die VCL
- Die neue `TSkAnimatedPaintBox` Skia-basierte Komponente für die VCL
- Die neue `TSkSvg` Skia-basierte Komponente für VCL

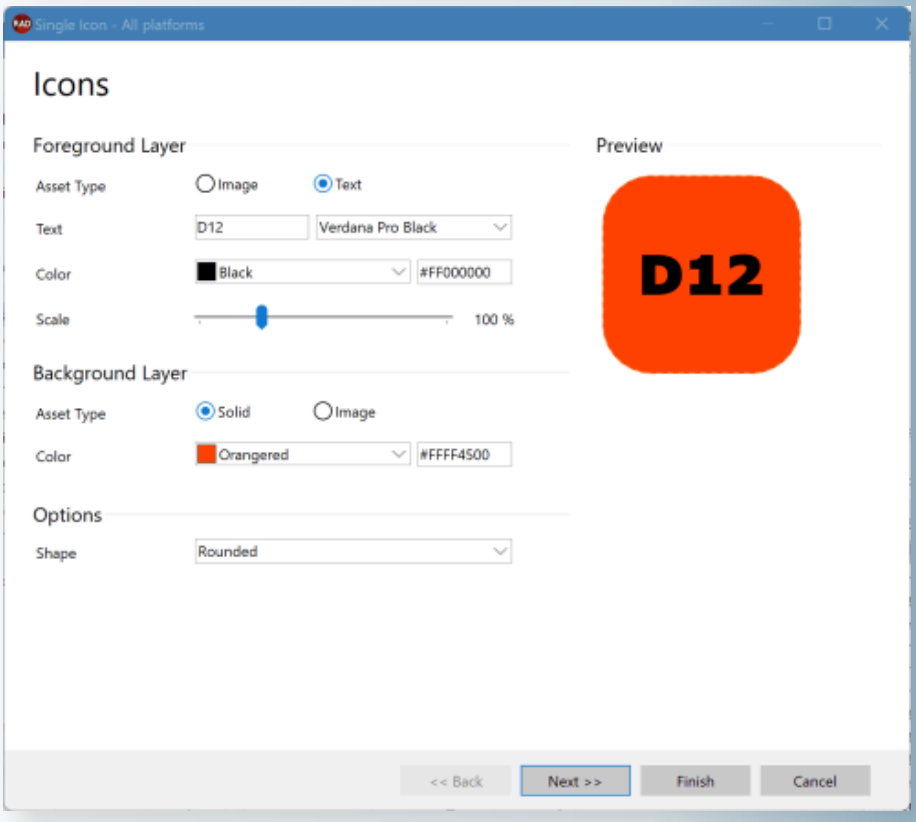
Bitmaps (Windows GDI)

Ein Bitmap ist ein grafisches Objekt, das zum Erstellen, Bearbeiten (Skalieren, Scrollen, Drehen und Malen) und Speichern von Bildern als Dateien auf einer Festplatte verwendet wird. Dieser Überblick beschreibt die Bitmap-Klassen und Bitmap-Operationen.

In der IDE

1. Der Dialog Suchen in Dateien **(Find in Files)** hat jetzt eine neue Option " **Subdirectory exclude mask** ".
2. Die **Syntaxhighlighting** wurde zu den Hinweisen von **Error Insight**, der Navigationssymbolleiste und dem Call Stack hinzugefügt
3. Die Syntax der Strukturansicht hebt Methoden und Typen hervor und hat **Syntaxhighlighting** zu den **Error Insight**-Meldungen hinzugefügt
4. Die Seite **Options - IDE - Saving und Recovering** enthält eine neue Checkbox zum Speichern des Editorstatus
5. Das **Markdown**-Fenster ändert jetzt seine Farbe, wenn das IDE-Thema geändert wird
6. Die Willkommensseite unterstützt jetzt das smooth mouse wheel scrolling
7. "Die Anzahl der GDI-Bitmaps* ist in der IDE geringer, da die Bilder inaktiv gemacht werden wenn sie für einige Zeit nicht benutzt werden."
8. "Die Delphi LSP-basierte Code-Vervollständigung kann nun Code-Templates und Sprach-Schlüsselwörter (Language keywords) anzeigen."
9. Die Code-Vervollständigung fügt jetzt Array-Klammern [ ] für Array-Typen hinzu
10. Alle Plattformen Einzel-Symbol-Assistent
11. PAServer-Meldungen, einschließlich Hinweisen, werden im IDE-Nachrichtenfenster angezeigt
12. Neue ToolsAPI, IOTARawEditReader Schnittstelle

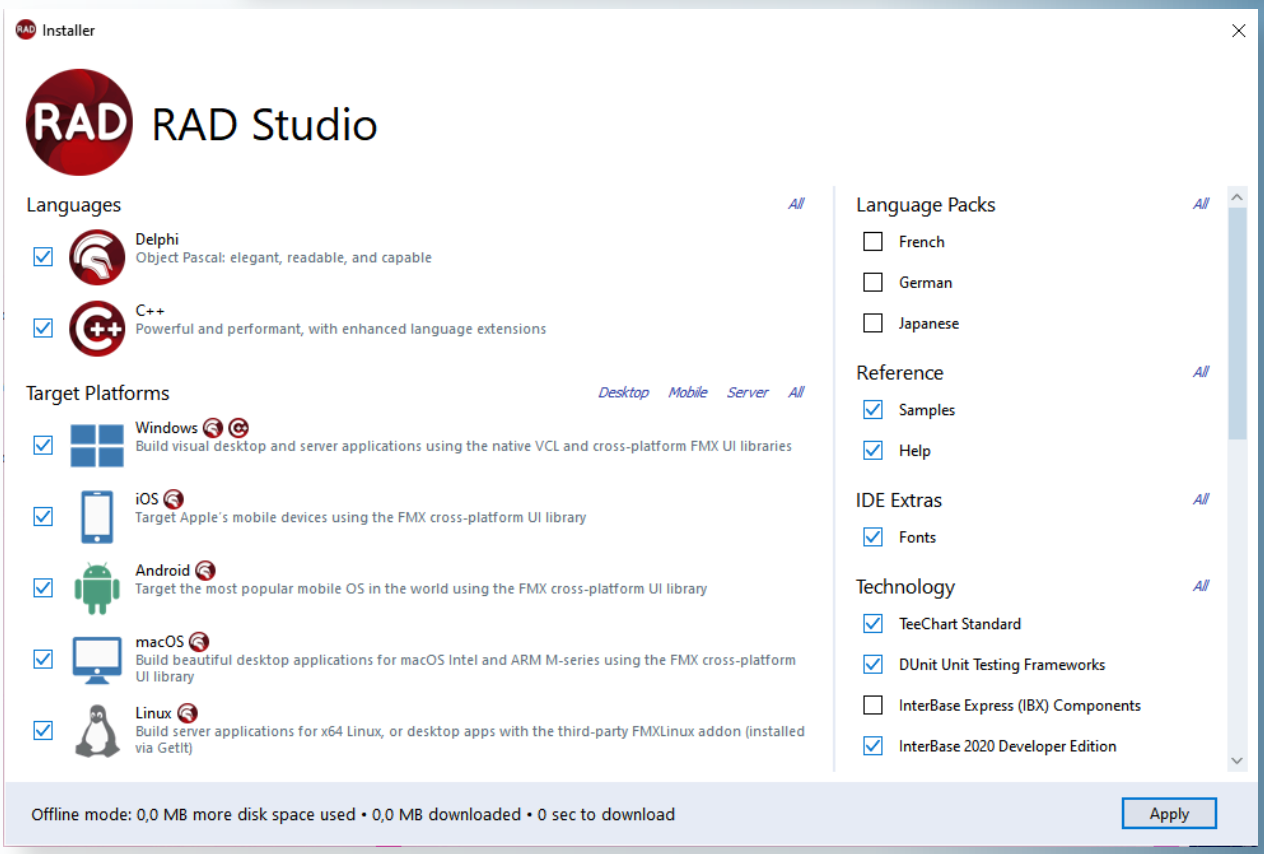
Unten: Der Assistent für alle Plattformen mit einem Symbol



In GetIt und dem Installationsprogramm

- Der Platform Manager wurde in Feature Manager umbenannt.
- Der Feature Manager hat eine völlig neue Benutzeroberfläche, die mit regulären VCL-Steuerelementen und Stilen aufgebaut ist
- Der Feature Manager bietet alle Optionen auf einer einzigen Seite an
- Der Feature Manager trennt die Sprachen und die Plattformen, die Sie installieren möchten
- Der Feature Manager enthält voreingestellte Konfigurationen für gängige Szenarien wie Desktop oder Mobile
- Der Feature Manager hat eine neue Schaltfläche "Fehler", die Installationsfehler direkt anzeigt und hat einen direkten Link zur Fehlerprotokolldatei
- Das GetItCmd Kommandozeilentool protokolliert jetzt in der Datei GetItInstall.log
- Der GetIt-Paketmanager hat die Möglichkeit, mehrere lokale GetIt-Pakete mit einem einzigen Vorgang zu laden
- Die integrierte Version von DUnitX wurde aktualisiert
- Die integrierte Version von Indy wurde aktualisiert
- Die seit langem veralteten VCL-Übersetzungstools wurden aus der Kernproduktinstallation entfernt
- Die relativ alte Modellierungsunterstützung wurde zu einer optionalen Funktion im Feature Manager

Below: The new Feature Manager window



Und schließlich, in der Sprache Delphi

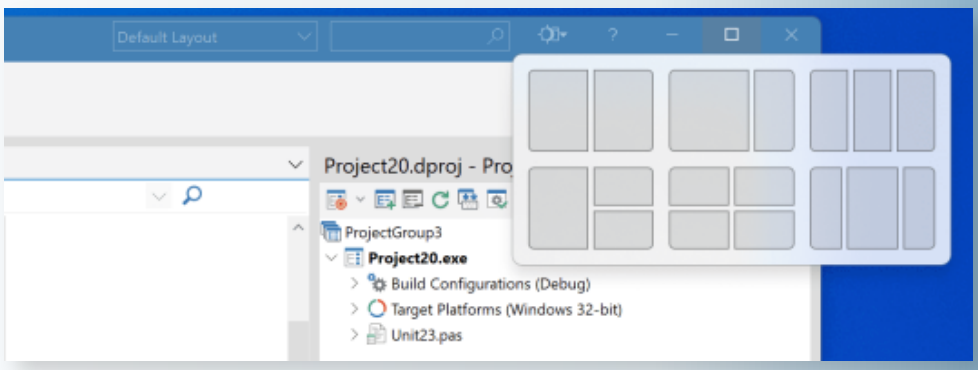
❶ Langer **literal String** mit mehr als 255 Zeichen
❷ **Mehrzeilige** String-Literale
❸ **TEXTBLOCK-Direktive** zur Angabe des Zeilenumbruchformats von mehrzeiligen Strings
❹ **NativeInt**, ein Typ, der je nach Plattform auf **Integer** oder **Int64** abgebildet wird, ist jetzt ein weak Type-Alias
❺ Verbesserte Warnungen in generic classes
❻ Neues **LLVM-Symbol**, das in allen LLVM-basierten Delphi-Compilern definiert ist
❼ Option zum Exportieren des Graph der verwendeten Unit in eine **GraphViz-Datei** (-graphviz)
❽ Möglichkeit, eine Familie von Units aus der GraphViz-Datei **auszuschließen** (-graphviz-exclude)
❾ Unterstützung für **NaN**-Vergleiche (keine Zahl), wie von **IEEE** gefordert
❿ Optimierter generated Code für Div-Operationen, wenn der Divisor eine Konstante ist
⓫ **Zwei neue Funktionen für die System-Unit,** `GetCompilerVersion` und `GetRTLVersion`
⓬ **FloatingPoint-Exceptions** sind jetzt **standardmäßig** auf allen Plattformen **deaktiviert**

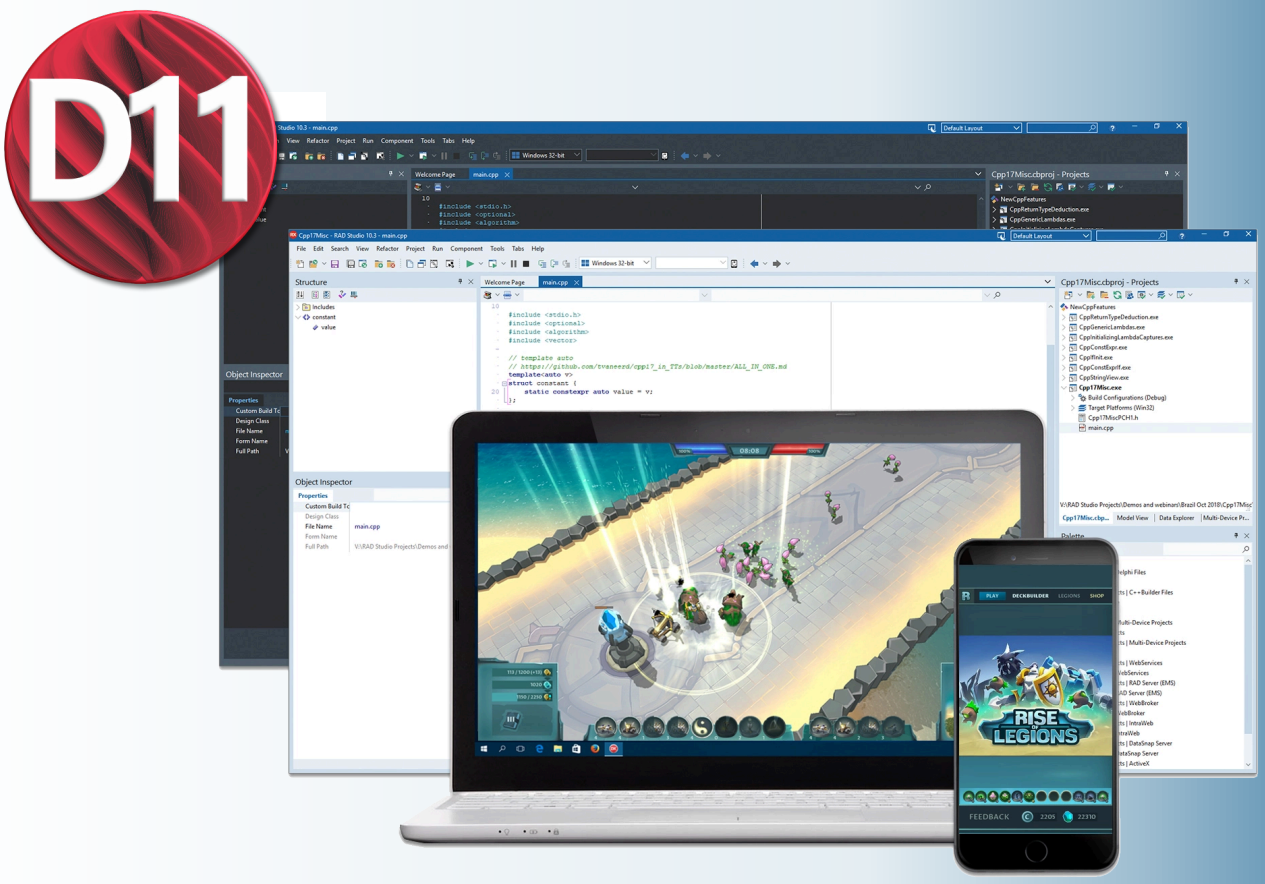
Mehrzeilige Delphi-Stringliterals in der IDE - ein Beispiel finden Sie auf Seite 1 dieses Artikels.

Bonusabschnitt: In Windows-Plattform

- **Neue WinAPI-Definition** basierend auf **WinMD Microsoft-Metadaten** (*311 Header-Dateien mit 41 MB Delphi-Code*)
- **WinRT** APIs und **WebView 2** Control API auf die neueste Version aktualisiert
- Edge-Browser: UserAgent ist in `ICoreWebView2Settings` verfügbar; **ICoreWebView2Profile2** enthält die Methoden `ClearBrowsingData`, `ClearBrowsingDataAll`, und `ClearBrowsingDataInTimeRange`; `TEdgeBrowser` `OnDownloadStarting` Ereignis, `NavigateWithWebResourceRequest` Methode, `Print` und `ShowPrintUI` Methoden
- **Neue Windows 11 Stile**
- `TForm` `Class` `EnableImmersiveDarkMode` Methode und `RoundedCorners` Eigenschaft `für` **Windows 11**
- Unterstützung der **Titelleiste für Windows 11 Snap-Layouts** (*siehe Bild unten*)

Snap-Layout für die Titelleiste funktioniert auch in der IDE





FREE COMMUNITY EDITION

Erhalten Sie **Community Edition Kostenlos / Kostenlose Delphi IDE mit vollem Funktionsumfang** für die Erstellung nativer, plattformübergreifender Anwendungen

Kann ich die Delphi CE erhalten?
Wenn Sie eine Einzelperson sind, können Sie **Delphi CE (C**ommunity **E**dition**)** verwenden, um Apps für den Eigengebrauch und Apps zu erstellen, die Sie verkaufen können, bis Ihr Umsatz 5.000 US-Dollar pro Jahr erreicht.

Wenn Sie ein kleines Unternehmen oder eine Organisation mit einem **Jahresumsatz** von bis zu 5.000 US-Dollar sind, können Sie Delphi CE ebenfalls verwenden.
Sobald der Gesamtumsatz Ihres Unternehmens 5.000 US-Dollar erreicht oder Ihr Team auf mehr als fünf Entwickler angewachsen ist, können Sie auf eine uneingeschränkte kommerzielle Lizenz mit der Professional Edition umsteigen.
Delphi CE eignet sich auch perfekt für Startups in der Frühphase, die ihre Produktvision noch vor der Beschaffung von Kapital in die Tat umsetzen wollen! Entwickeln Sie Ihre professionelle Anwendung mit der Community Edition in der Gewissheit, dass Sie die Lernkurve überspringen können, die Ihre Konkurrenz bei der Entwicklung für mehrere Plattformen durchlaufen muss.
Weitere Einzelheiten finden Sie in den FAQs zur Community Edition **`.https://www.embarcadero.com/products/delphi/faq`**

ÄNDERUNGEN IM LCL INTERFACE

COCOA

- **IME (**Editor für Eingabemethoden, *erlaubt die Eingabe von Zeichen die nicht auf der Computertastatur verfügbar sind***)** unterstützt komplett solche Zeichen wie: **`Chinesische/japanische/koreanische Zeichen, DeadKeys, Emoji & Symbole`**.
- Volle Unterstützung für Multibildschirm-Systeme,
- Volle Unterstützung für Docking, auch bei der IDE (*Integrated Development Environment*).
- Kursor komplett überarbeitet und deutlich verbessert, kompatibel mit **MacOS Ventura**..
- **`TPageControl`** deutlich verbessert.
- Viele Kontrollelemente überarbeitet (`TComboxBox/TListBox/TDateTimePicker/TStaticText/TSpeedButton/TPanel etc.`)
- Viele Speicherlecks beseitigt.

Qt

- `TCheckBox.Alignment` und `TradioButton.Alignment` implementiert.
- `TCustomComboBox.AdjustDropDown` und `TcustomComboBox.ItemWidth` implementiert.

Qt5

- **Qt5** benutzt native Event Loops auf allen Platformen. C bindings aktualisiert. Die minimale C bindings Version für L**azarus** 3.0 ist 1.2.15.
- Beachten Sie, dass die meisten **Linux** Distributionen bis zu ihrer nächsten Version nach der regulären Freigabe von **Lazarus 3.0** noch nicht über eine geeignete **libqt5pas library** verfügen. Bauen Sie **Lazarus** von ihren eigenen Quellverzeichnis oder laden sie die aktuellsten hier herunter: **`https://github.com/davidbannon/libqt5pas/releases/latest`**
- `TCheckBox.Alignment` und `TRadioButton.Alignment` implementiert.
- `TCustomComboBox.AdjustDropDown` und `TcustomComboBox.ItemWidth` implementiert.

Qt6

- Qt6 Widgetset implementiert. C bindings basieren auf Qt6 6.2.0 LTS. Die minimale C bindings Version für L**azarus** 3.0 ist 6.2.7.
- Beachten Sie, dass die meisten **Linux** Distributionen bis zu ihrer nächsten Version nach der regulären **Freigabe von Lazarus 3.0** noch nicht über eine geeignete **libqt6pas library** verfügen. Bauen Sie **Lazarus** von ihren eigenen Quellverzeichnis oder laden sie die aktuellsten hier herunter: **`https://github.com/davidbannon/libqt6pas/releases/latest`**
- `TCheckBox.Alignment und TRadioButton.Alignment` implementiert.
- `TCustomComboBox.AdjustDropDown` und `TcustomComboBox.ItemWidth` implementiert.

Gtk3

- **Gtk3 Pascal Bindings komplett überarbeitet**.
- Verbesserung der Stabilität.
- Erfordert jetzt **GTK** >= 3.24.24 und Glib2.0 >= 2.66

ÄNDERUNGEN IN DER LCL

TCustomImageList

`TCustomImageList` erweitert:

- Eine Protected `MarkAsChanged` Methode, welche `FChanged` auf `true` setzt, eingeführt (Erlaubt *custom triggers für den* `OnChange` *Event*).
- Virtual protected `DoAfterUpdateStarted` und `DoBeforeUpdateEnded` Methoden zugefügt. Diese werden im ersten `BeginUpdate` und im letzten `EndUpdate` aufgerufen.

TTaskDialog

- Altes Verhalten in Win32:
  Für `FooterIcon = tdiNone` und `MainIcon = tdiNone`. wurde ein Platzhalter-Icon benutzt.
- Neues Verhalten in **Win32:**
  Es gibt kein Icon mehr für `FooterIcon = tdiNone` und `MainIcon = tdiNone`.
- **Grund**: Fehler beim Zeichnen beseitigt.
  Der Text wurde verschoben, um mehr Inhalt und eine bessere Ausrichtung zu ermöglichen. *Siehe Issue #39172*

TSpeedButton

- **Altes Verhalten:** Mehrzeilige Captions konnten nur im Code eingegeben werden und waren immer links ausgerichtet.
- **Neues Verhalten:** Im Object Inspector können jetzt mehrzeilige Captions eintragen werden. Die neue Eigenschaft `Alignment` stellt ein, ob die Caption links, rechts oder zentriert ausgerichtet ist. **Default**: Zentriert wie in **Delphi**.
- **Grund:** Bessere Nutzbarkeit.

TLabel.Transparent, .Color und .ParentColor geändert:

- Altes Verhalten: Die Eigenschaft `Transparent` war an `Color=clNone` gebunden.
- **Neues Verhalten:** Transparent ist nun eine separate Eigenschaft.
- **Grund: Delphi** Kompatibilität und Problembehebung bei `ParentColor`.
- **Abhilfe:** Wenn sie die `Color` Eigenschaft setzen, wird `Transparent` nicht mehr automatisch von `True` nach `False` umgeschaltet. Sie müssen das selber machen. Dies steht im Einklang mit **Delphi** und löst auch die Probleme mit `Color/ParentColor` Änderungen.

TPanel.VerticalAlignment

- **Altes Verhalten:** Die **Panel** caption war immer vertikal zentriert.
- **Neues Verhalten:** Die neue Eigenschaft `VerticalAlignment` (`taAlignTop`, `taAlignBottom`, `taVerticalCenter`) erlaubt, die `Caption` auch oben oder unten im Panel zu platzieren.
- **Grund: Delphi** Kompatibilität und bessere Nutzbarkeit.

TCalendar

- Eigenschaften `MinDate` und `MaxDate` implementiert. Diese Limitierungen sind gültig, wenn `MaxDate > MinDate` ist. Leider unterstützt das **GTK2/3 widgetsets** dies nicht. Es wird also die Eingabe eines Datums außerhalb des `MinDate/MaxDate` Bereichs immer noch möglich sein.

TCheckbox, TRadioButton

- Unterschiedliche Berechnung der `checkbox/radiobutton` Größe, um richtig auf das Win-10 "Ease of access" Feature zu reagieren. Siehe issue #39398
- **Konsequenz:** Im Falle von *auto-sized* werden `lfm` Dateien im Vergleich zu früheren Versionen unterschiedliche Größen für diese Steuerelemente enthalten.

LCL-ÄNDERUNGEN WEITERFÜHRUNG

Grids

- Sie können nun die Zelleditor-Eigenschaften `ParentColor` und `ParentFont` durch Einfügen von `goEditorParentColor` sowie `goEditorParentFont` in Grid-Eigenschaft Options2 setzen.

`TShellTreeView`

- Die neue Eigenschaft `ExpandCollapseMode` Optionen definiert, ob ein kollabierender Knoten seine Kindknoten löschen soll.
- Implementiert eine benutzerdefinierte Sortierung der treeview Items durch das Setzen von `FileSortType` nach `fstCustom` und liefert eine benutzerdefinierte Vergleichsfunktion im Event `OnSortCompare`.

`TShellListView`

`TShellListView` ist jetzt eine Unterklasse von `TListItem` damit man Dateiinformationen darin speichern kann.
Wenn Sie `OnCreateItemClass` benutzen, um eine eigene abgeleitete Klasse von `TListItem` zu erzeugen, kann es ratsam sein, Ihre eigene Klasse auf `TShellListItem` anstatt auf `TlistItem` aufzubauen. Auf diese Art werden Sie auch Zugriff auf die `TShellListItem's FileInfo` Eigenschaft haben.

`TTreeView`

`ShowSeparators` als published Eigenschaft wie `ShowLines` und `ShowRoot` eingefügt.
`TTrayIcon` nur für gtk2

ÄNDERUNGEN BEI DER IDE

Zeichentabelle

- Zeichengröße veränderbar, um Lesbarkeit zu verbessern.
- Die Zeichentabelle wurde von der IDE abgetrennt und und in ein separates Package verschoben. Das designtime Package ist standardmäßig installiert, damit hier kein Unterschied zur IDE ist. Jetzt können Benutzer in ihren eigenen Anwendungen darauf zugreifen, nachdem sie das Laufzeitpaket "`charactermappkg.lpk`" zu den Projektanforderungen hinzugefügt haben.

DEBUGGER

Projektoptionen

- **"Run(F9)"** kann sowohl "**Debug**" als auch "**Lauf ohne Debug**" bedeuten. In den neu eingeführten **Modes Debug/Release** wird der Release Mode nicht mehr standardmäßig den Debugger aufrufen. **Existierende Debug/Release Modes** müssen editiert werden, um dies zu erlauben.
- **Projektabhängige "Debugger backend" Einstellungen:** Zusätzlich zur Auswahl eines spezifischen Backends von den globalen IDE-Einstellungen kann ein Backend spezifisch für ein Projekt konfiguriert werden (*z. B. spezielle gdb-server Einstellungen*).

IDE Dialogs

- Verbessertes Fenster für Überwachungen (`watches`)
  - Auf- und Zuklappen für `classes, records,` usw.
  - Auf- und Zuklappen mit Seitennavigation für `Arrays`.
  - Drag und Drop zum Neuordnen der Überwachungen
  - Drag und Drop, um neue "Top-Level"-Überwachungen aus verschachtelten Einträgen zu erstellen (*in auf- oder zugeklappten Listen*)
  - Adressspalte für Typen mit internem Pointer (`classes,long-string dyn-array,(real)pointer`)
- Verbessertes Locals Fenster
  - Auf- und Zuklappen für classes, records, usw.
  - Auf- und Zuklappen mit Seitennavigation für Arrays.
  - Adressspalte für Typen mit internem Pointer (`classes,long-string,dyn-array,(real)pointer`)
    - Power Button

IDE CHANGES CONTINUATION

Improved Inspect window

Verbessertes Inspect Fenster

- Gelöst: Daten aktualisieren, wenn sich der Kontext ändert.
- Optionen für Funktionsaufrufe / und "Converter" (*Siehe* `FpDebug: SysVarToLstr`) zugefügt.
- Suchfilter für Suche in Text oder Werten eingeführt.
- `ctrl-Up/Down/Page-Up/Page-Down` eingefügt, um in einem Grid zu navigieren.
- `Alt-Left/Right` für History. und `ctrl-Enter` zur Selektion eingeführt.

Verbessertes Evaluate/Modify Fenster

- Neues Layout.
- DisplayFormat zugefügt.
- Optionen für Funktionsaufrufe / und "Converter" (*Siehe* `FpDebug: SysVarToLstr`) zugefügt.

Verbessertes Assembler Fenster

- Navigationshistorie (*vorwärts/rückwärts*).
- Für `FpDebug`: Annotationen für jump/call Targets und Freigabe für ctrl-click, um Targetadressen zu disassemblieren.

FpDebug / LazDebuggerFp

- Verbessertes "function calling" bei Evaluierung von Überwachungen. Siehe `FpDebug-Watches-FunctionEval`
- %RAX greift auf CPU Register in Überwachungsausdrücken zu. (nur vollständige Register, *nicht für AH oder AL oder EAX bei 64bit*)
- Integrierte Funktionen: FpDebug-Watches-Intrinsic-Functions
- Integrierte/erweiterte Operatoren: `MyArray[1..3] array` Abschnitt mit Operator-Mapping. (*Siehe: https://wiki.lazarus.freepascal.org/FpDebug-Watches-Intrinsic-Functions#Intrinsic_Operators*)
- Optionen, um "`variant`" zu erkennen und Aufruf von "`SysVarToLStr`" in der Zielanwendung.
- Einzelne Threads anhalten/fortsetzen (*Dies muss erfolgen, während die Anwendung pausiert. Es wird dann für den nächsten Schritt/Lauf angewendet*)
- Teilweise Verbesserungen beim Debuggen in DLLs: **`https://wiki.freepascal.org/Debugger_Status#Other`**
- Der Disassembler kommentiert `call/jmp/jne/...` mit Informationen über die Zieladresse (*Funktionsname, Datei, Zeile*)
- F7/F8 Schritt Hinein/Weiter kann jetzt benutzt werden, um den Debugger zu starten und zur ersten Zeile das Hauptprogramms zu springen `begin/end`.

LazDebuggerFpLldb (default on MacOS)

- Speicher Limits (und `String,Pchar,Array`) zur Debugger Konfiguration zugefügt. Siehe **`https://wiki.freepascal.org/LazDebuggerFp`** `(MaxMemReadSize, MaxStringLen, MaxArrayLen, MaxNullStringSearchLen)` Begrenzung der Standardergebnisse für Überwachungen (`watches)/locals/stack-params,...` soll langsames Abarbeiten verhindern. Arrays können im Überwachungsfeanster mit der neuen "Seitenansicht für Arrays" (Erweitern mit `[+]`) angesehen werden.

Fließkomma Properties im Object Inspector

- Der **Object Inspector verbietet es nun ausdrücklich, den Wert einer Fließkomma Property auf** +/-Inf oder **NaN** (*Not a Number*).
- **Grund:** +/-Inf und **NaN** are valid values for a floating point property, sind zwar gültige Werte für Fließkomma Properties, können aber nicht gestreamt werden (also das Form kann sie nicht laden). Sie auf **NaN** zu setzen, verursacht Chaos in der IDE.
- Abhilfe: Setze den Wert während der Laufzeit (im Code).

IDEEN-ÄNDERUNGEN FORTSETZUNG

Unitnamen im lfm lesen

- Der FPC 3.3.1 Komponenten-Editor unterstützt optional das Schreiben von Typen mit ihren Unitnamen als unitname/type. Lazarus kann dies nun lesen.

Zweideutige Classtypes

- Sie können nun zwei Komponentenklassen mit demselben Namen registrieren, z. B. fresnel.TButton und StdCtrls.TButton. Sie können sogar beide in die dasselbe Formular einfügen.

Editor

- Hervorhebungen für PasDoc.

IDE Optionen

In Feld Werkzeuge -> Eistellungen -> Umgebung -> Fenster sind diese drei Settings standardmäßig ON:

- IDE-Titel beginnt mit Projektname,
- IDE-Titel beinhaltet Projektverzeichnis,
- IDE-Titel beinhaltet Erstellmodus.

Dies war eine Forderung vieler Nutzer. Wenn nämlich die Titelleiste der IDE keine Informationen über das aktive Projekt enthält, muss der Benutzer den **Projektinspektor** oder die **Projekteinstellungen** öffnen, um dies zu sehen, was wirklich unpraktisch ist.

Lazarus Beispiele

- Es gibt einen neuen Ansatz für Beispiele. Ein Beispiel wird in einen neuen Arbeitsbereich kopiert, wodurch das Linux-Read-Only-Problem vermieden wird. Und es gibt ein möglicherweise einfacher zu nutzendes Suchmodell.

ÄNDERUNGEN IM LCL INTERFACE

Komponenten

TAChart

- `TLegendClickTools` kann nun Klicks auf eine ganze Reihe von Legendeneinträgen erkennen und sendet die angeklickten Einträge an das neue Event `OnSeriesClick`.
- Es gibt ein neues `TDatapointMarksClickTool` welches aktiv wird, wenn der Nutzer auf die Bemessungsmarken einer Serie klickt..
- Es gibt eine neue Eigenschaft `TickWidth` für die Diagrammachsen.
- Die neu eingeführte Eigenschaft `FullWidth` ermöglicht, den Hintergrund des Diagrammtitels oder der Fußzeile übe die ganze Breite des Diagramms auszudehnen.
- Eigenschaft `RandomColors` für `TrandomChartSource` eingefügt.
- Weitere neue Eigenschaften `YIndexWhiskerMin`, `YIndexBarMin`, `YIndexCenter`, `YIndexBarMax`,`YindexWhiskerMax` und `YDataLayout` in `TBoxAndWhiskerSeries` für flexiblere Zuordnung der Y-Werte zu Box- und Whisker-Plots eingeführt.
- Eine neue Option `aipInteger` im Set `TAxisIntervalParamOptions` setzt Achsen-Labels nur auf Integer-Werte und unterdrückt damit unerwünschte Zwischenlabels in Balkendiagrammen. Dies hilft dabei, Beschriftungen in logarithmischen Diagrammen in den Potenzen der logarithmischen Basis (normalerweise 10) zu erzwingen.
- Ein ganz neuer Event `OnAddStyleToLegend` für `TChartStyles` hat u.a. einen `boolean` Parameter `AddToLegend`, mit dem man bestimmen kann, ob die Ebene einer Serie, die diesen Stil benutzt, in der Legende angezeigt wird.

TDateTimePicker

- Neue Eigenschaften `MonthDisplay` und `CustomMonthNames` sind dafür gedacht, die Eigenschaft `MonthNames` zu ersetzen, welche veraltet ist.
- Die neue Eigenschaft `DecimalSeparator` erlaubt, benutzerspezifische Werte anstatt von hart codierten Semikolons zu nutzen.

IDE INTERFACE CHANGES CONTINUATION

TDBDateTimePicker

- Eigenschaft Options ist jetzt public.
- Eigenschaft `DecimalSeparator` public gemacht.
- Die fehlende Eigenschaft `Alignment` public gemacht.
  Dies ist nun konsistent mit `TDateTimePicker`.

T(Float)SpinEditEx

- Es gibt eine neue Eigenschaft **Orientation**, die erlaubt, die Spin-Buttons horizontal anzulegen.

TCheckListBox

- Eigenschaften `HeaderColor` und `HeaderBackgroundColor` zugefügt. Diese werden in Listitems genutzt, wo die Header Eigenschaft aktiviert ist. Implementiert für das Win32 Widgetset.

PAS2JS

- lazbuild kann nun **PAS2JS-Projekte** kompilieren indem die Umgebungsvariable PAS2JS mit dem Pfad der ausführbaren Datei pas2js übergeben wird.
- Projektgruppen mit pas2js-Projekten können nun kompiliert werden, ohne sie zu öffnen.
- Neuer Projekttyp **PROGRESSIVE WEB APPLICATION**
- Neuer Projecttyp **ELECTRON WEB APPLICATION**
- `pas2jsdsgn` benutzt jetzt das `SimpleWebServerGUI` Package.
  Dies ersetzt den eigenen **http server controlle**r.
- **F9, Run** baut und startet einen **HTTP server** und einen B**rowser.**

Lazarus Icon Collection

- Keine Komponente, aber die **Lazarus** Installation enthält jetzt ein Verzeichnis mit universell einsetzbaren Symbolen zur Verwendung in Symbolleisten, Menüs, Schaltflächen usw. beliebiger GUI-Anwendungen (*Ordner images/general_purpose*).
  Die Bilder kommen in verschiedenen Größen und sind daher kompatibel mit der skalierten Bilderliste von Lazarus v2.0+. Autor: Roland Hahn (**`https://www.rhsoft.de/`**).
  Lizenz: Creative Commons CC0 (keine Einschränkungen in der Nutzung).

gir2pascal

Die Quellcodes von `gir2pascal` (ein Tool, um **GObject Introspection** *in* **Pascal** *Dateien zu übersetzen*) sind im **Lazarus source tree** (`tools/gir2pascal directory`) enthalten und werden dort gepflegt.

lazdelphi

Ein IDE Addon, das einen Parser für die Delphi Compiler Fehler und Hinweise hinzufügt. Sie können dcc32.exe als externes Tool ausführen oder den Befehl before in den Compiler-Optionen und den Delphi Compiler Parser für die Ausgabe verwenden, so dass die Fehler/Hinweise im Messages-Fenster den Quelltext öffnen können.
Siehe **Lazarus Delphi Compiler Tool.**

Jedi Code Format

Einige Verbesserungen bei Codeformatierung eingeführt..
Das **`jcf`** Kommandozeilen-Tool ist jetzt eine Textanwendung und erfordert nicht mehr **Xwindow,** um auf **LINUX** zu laufen.

DockedFormEditor

Falls Sie noch den angedockten Formular-Editor von Sparta benutzen, verwenden Sie stattdessen das Paket **dockedformeditor**.

ÄNDERUNGEN, DIE DIE KOMPATIBILITÄT BETREFFEN

IDE Einstellungen

Start > Startparameter

Das Working-Dir und die Launch-/Host-App können nun relativ zum Project-Dir angegeben werden. Dadurch und aufgrund von Inkonsistenzen zwischen "Run in debugger" und "Run without debug" haben sich einige Details der Auflösung dieser Felder geändert.
In manchen Fällen müssen Sie daher Ihre Einstellungen anpassen.

Das Arbeitsverzeichnis (working-dir) wird nun wie folgt festgelegt:

❶ BuildMode.RunParams "working directory" kann vom Nutzer eingestellt werden (*Neu: Das kann jetzt auch relativ zum Projektpfad sein. Um das Verzeichnis zu erhalten, in dem sich die ausführbare Datei befindet (exe), kann man folgendes angeben:* `"$Path($(OutputFile))"`)
❷ Das Projektverzeichnis, wenn es nicht virtuell ist.
❸ Das Verzeichnis von der Host-App (*RunParams*) oder wenn (und nur wenn) die Host-App leer ist, das Verzeichnis der Project.exe (*Wenn die Host-App in %Path steht, dann gibt es da kein „working directory“*).

Die ersten beiden Punkte sind wie zuvor.
Der dritte Punkt war vorher nur für "debugging", aber "run without debug" nutzte „Launch-App", "Host-App", `Project.exe`

Änderung

Der Pfad der "Launch App" wird nicht mehr berücksichtigt.
Die Launch-/Host-App Position wird nun wie folgt festgelegt:
❶ Eine App mit absolutem Pfad wird wie angegeben verwendet.
❷ Ein relativer Pfad (oder gar kein Pfad) wird als relativ zum Projektverzeichnis aufgelöst.
❸ Eine Anwendung ohne jeglichen Pfad (wenn sie in Schritt 2 nicht gefunden wurde) wird in der %PATH Umgebungsvariable gesucht.

Änderung

Die Suche in %PATH wurde nur von "run without debug" durchgeführt, aber nicht von „debug“. Jetzt wird sie von beiden durchgeführt.

Änderung

Die Prüfung auf eine exe relativ zum Projektverzeichnis wurde hinzugefügt.

LCL Inkompatibilität

TLabel: autosized und right-aligned

Altes Verhalten:

Autosized Label mit `Alignment=taRightJustify` aber `Anchors=[akLeft,...]` dehnt sich nach links aus.

Neues Verhalten: Das Label dehnt sich jetzt nach rechts aus.

Grund:

Es war nicht möglich, das Verhalten auch für versteckte Etiketten zu implementieren, ohne erhebliche Erweiterungen in der LCL zu implementieren. Die LCL hat eine andere und allgemeinere Funktion der Anker an den Bedienelementen, die den gleichen Effekt hat (siehe Abhilfe unten), so dass es nicht nötig und gewünscht ist diese Funktion zu verdoppeln und den LCL-Code komplexer und fehleranfälliger zu machen.

Abhilfe: Benutze **LCL anchoring** zu einem sekundären Steuerelement**.**

Verankere die rechten Seite eines Labels an einem anderen Steuerelement.
Das autosized Label dehnt sich dann nach links aus, wird sich aber nicht nach rechts bewegen, wenn das übergeordnete Element die Größe verändert, wie bei einem simplen `akRight` Anker ohne Referenzelement ist.

ÄNDERUNGEN, DIE DIE KOMPATIBILITÄT BETREFFEN

TDateEdit/TTimeEdit

Der Wert von `NullDate` wurde geändert.

Grund:

Es war unmöglich, das Datum, das `NullDate` (standardmäßig *30. Dez. 1899)* im Kontrollelement auszuwählen.

Lösung (1):

Wenn der Code vom `NullDate` abhängig ist (derzeit 0.0), muss der Code angepasst werden.

Lösung (2):

Wenn der Code `NullDate` für ein `TTimeEdit` benutzt, muss der neue Wert `NullTime` stattdessen benutzt werden.

Anmerkung: `NullDate` ist eigentlich eine beschreibbare Konstante. Dies wurde aus Gründen der Kompatibilität beibehalten. Es ist jedoch keine gute Idee, den Wert auf ein tatsächliches Datum zu ändern, das innerhalb des Bereichs des Steuerelements liegt.

Cocoa

Einige globale Konfigurationsvariablen wurden von `CocoaInt` zu `CocoaConfig`. verschoben. Wie zum Beispiel `CocoaBasePPI, CocoaIconUse, CocoaToggleBezel, CocoaToggleType` etc.

GTK3

GTK3 wird von früheren **Linuxes** wie **Ubuntu 20.04** nicht mehr unterstützt.
Es wird GTK >= 3.24.24 und Glib2.0 >= 2.66 benötigt.

LAZUTILS

Masks unit

Die Masks Unit ist komplett neu geschrieben.

Gründe:

Geschwindigkeit: Die alte Matches() Methode hatte eine O(n^2) or eben O(n^3) Charakteristik.

Verbesserte Kontrolle darüber, wie die Maske interpretiert wird.

Neue Typen (*für Parameter*) und ein spezielle `TMaskWindows` Klasse wurde eingeführt. `TMask.MatchesWindowsMask` und die alten `TMaskOptions` Typen sind veraltet und werden mit der nächsten Version entfernt.

Ranges und Sets

Die alte masks Implementation unterstützte **sets** aber keine **ranges**.
Die neue Implementation unterstützt beides `sets ([abc])` und `ranges ([a-c])`.
Als Konsequenz wird ein `'-'` in so einem Konstrukt nun als Teil der range Definition interpretiert und nicht als Zeichen '-'.

Grund: ranges sind standardmäßig eine gute Sache. Sie fehlten einfach in der alten Implementierung. Wir entschlossen uns diesen kleinen Preis zu zahlen.

Abhilfe: Entweder das Zeichen '-' mit EscapeChar (*standardmäßig* `'\'`) kennzeichnen oder ausschließen mit `mocRange` vom `TMaskOpcodes` Parameter.

Konstruktoren schlagen bei einer ungültigen Maske nicht mehr fehl

- Bei der Übergabe einer ungültigen Maske an die alten `T(Windows)Mask(List)` -Konstruktoren wurde eine Ausnahme ausgelöst.
- Die neuen Konstruktoren lösen in diesem Fall keine Ausnahme aus. Stattdessen wird eine Ausnahme ausgelöst wenn Matches() aufgerufen wird.
- **Grund:** Unschön, wenn ein Konstruktor fehlschlägt.

VERÄNDERUNGEN DIE SICH AUF AUSWIRKEN KOMPATIBILITÄT FORTSETZUNG 2

Translations Unit

`GetLanguageID` Funktion zugefügt. Sie gibt einen Record mit Sprachkodierung (*in ISO 639-1 oder ISO 639-2*) und Ländercode (*in ISO 3166*) für die derzeitige System-Lokalisierung.

Funktion `GetLanguageIDFromLocaleName` eingeführt.

Es parst den UNIX-Locale-Namen und gibt einen Datensatz mit Sprachcodierung und Ländercode zurück. Sie ist nützlich, um Sprachidentifikatoren zu analysieren, die z. B. über Kommandozeilenparameter übergeben werden.

Die Implementierung basiert auf der Prozedur `GetLanguageIDs` aus der Unit `GetText`, ist aber so umgeschrieben, dass sie die folgenden Eigenschaften hat:

- Sprach- und Ländercodes werden unter Windows immer im ISO-Format zurückgegeben.
- UNIX locale identifier wird korrekt geparst und Sprach-/Ländercodes werden korrekt extrahiert.
- Man kann nie davon ausgehen, dass der Sprachcode immer aus zwei Buchstaben besteht (ISO 639-1), er kann auch eine größere Länge haben (z. B. drei Buchstaben, wie in ISO 639-2).
- Die Locale-ID wird in einem Record Typ zurückgegeben. Dies wird es ermöglichen, in Zukunft weitere Felder in rückwärtskompatibler Weise zurückzugeben. Derzeit enthält er Sprachcode, Ländercode und Sprach-ID (Kombination aus *language code und country code*).

Diese Funktionen werden nun in der gesamten **Lazarus** Codebasis verwendet. Dies verbessert die automatische Spracherkennung und das Laden der korrekten Übersetzungen durch **Lazarus** erheblich:

- Dreistellige Sprachcodes (*ISO 639-2*) werden nicht mehr auf zwei Zeichen verkürzt. So werden die Übersetzungen für diese Sprachen korrekt geladen, wenn sie verfügbar sind.
- Bisher wurden unter Windows einige Sprach- und Ländercodes im Nicht-ISO-Format bezogen, was das korrekte Laden einiger Übersetzungen verhinderte, z. B. Chinesisch (zh_CN).
- Unter Unix werden Übersetzungen mit Ländercodes, wie brasilianisches Portugiesisch (pt_BR) oder Chinesisch (zh_CN) jetzt korrekt geladen.
- MacOS wird nun wie jedes andere Unix behandelt.

Dies entfernt die Abhängigkeit von der Sprachliste im Lazarus Bundle (die manuell gepflegt werden musste) und behebt somit das Laden von **tschechischen, ungarischen, brasilianischen, portugiesischen und ukrainischen** Übersetzungen.

LazUTF8 unit

- **LazGetLanguageIDs** (*rGibt Kombination von language und country codes zurück*) und `LazGetShortLanguageID` (gibt nur language code zurück) sind veraltete Prozeduren.
- **Grund:** Diese Funktionalität gehört zur Translations Unit (*Aufrufe dieser Prozeduren sind fast immer gefolgt von Aufrufen der Prozeduren aus der Translations Unit*) und diese Prozeduren sind jetzt Thin Wrappers von der `GetLanguageID` Funktion aus der `Translations Unit`.
- **Abhilfe:** Nutze die `GetLanguageID` Funktion aus der `Translations Unit`.

VERÄNDERUNGEN DIE SICH AUF AUSWIRKEN KOMPATIBILITÄT FORTSETZUNG 3

LazUTF8Classes und LazUTF8SysUtils Units
Alles aus diesen Units ist schon lange Zeit veraltet und wurde **jetzt entfernt.**
`LazUTF8SysUtils` wurde früher nach **`LazSysUtils`** umbenannt. Diese veraltete Version wurde für eine Übergangszeit belassen.

`Class` **`TStringListUTF8`** kann mit **`TStringList`** ersetzt werden.
`Class` **`TMemoryStreamUTF8`** kann mit **`TMemoryStream`** ersetzt werden.
`Global` procedure **`LoadStringsFromFileUTF8`** kann mit **`TStrings.LoadFromFile`** ersetzt werden.
`Global` procedure **`SaveStringsToFileUTF8`** kann mit **`TStrings.SaveToFile`** ersetzt werden.
Funktionen **`NowUTC`** und **`GetTickCount64`** sind nun in **`LazSysUtils`**.

INKOMPATIBILITÄT BEI KOMPONENTEN
LazControls

Abgeleitete Klassen von TSpinEditExBase
Alle abgeleiteten Klassen von `TSpinEditExBase` müssen eine `SameValue` Methode implementiert haben.
Diese Methode ist als eine `abstract` Methode in `TSpinEditExBase` definiert.
Grund: Alle abgeleiteten Klassen benutzen **Math.SameValue.** Das ist falsch beim Vergleichen von `integer types` (*auch wenn es sicher ist, wenn man relativ kleine Werte vergleicht*).
Abhilfe: Leider müssen Sie ihren Code anpassen.

TFloatSpinEditEx
Die Eigenschaft `NumbersOnly` ist nicht länger published.
Grund: Die Eigenschaft macht für dieses Steuerelement keinen Sinn und verwirrt die Benutzer nur.
Abhilfe: Wenn Sie wirklich `NumbersOnly` auf `True` brauchen, müssen Sie es im Code setzen.

FpVectorial
Das **Size element** im `FPVectorial` `TvFont` Record ist jetzt ein Floating point Wert `(type double)`.
Grund: Vermeidet Rundungsfehler, da die Zeichenkoordinaten bereits als `double` vorhanden sind.
Abhilfe: Es ist selten, dass diese Änderung Auswirkungen auf den Benutzercode hat.
Nur wenn `font size` in einer Variablen gespeichert wird, muss diese als `double` anstatt als `integer` definiert werden.

TurboPower_ipro
Type declarations und die HTML Nodes wurden von der `Unit IpHtml` in eigene Units `IpHtmlTypes, IpHtmlClasses und IpHtmlNodes` verschoben.
Dies kann das Kompilieren bestehender Projekte verhindern.
Grund: M**aintainability** der extrem langen Unit IpHtml verbessern
Abhilfe: Einfügen von `IpHtmlTypes, IpHtmlClasses` und/oder `IpHtmlNodes` in die `uses` Klausel der Projekt Units, wenn ein "identifier not found" Fehler betreffs dieses Packages vom Compiler geworfen wird.

TURTLE-GRAFIKEN IN LAZARUS ZEICHNEN

PROF DR HANS ZANTEMA
TECHNISCHE UNIVERSITEIT EINDHOVEN / RADBOUD UNIVERSITEIT NIJMEGEN

ZUSAMMENFASSUNG
In diesem Artikel geben wir Beispiele für erstaunliche Turtle-Grafiken, die mit Lazarus nur ein paar Zeilen Code brauchen.

DEFINITION VON TURTLE-GRAFIKEN

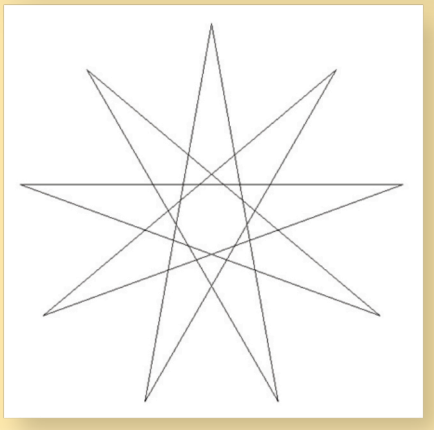
Für eine (*endliche oder unendliche*) Folge von Nullen und Einsen, und zwei Winkeln h0 und h1, ist die entsprechende Turtle-Grafik definiert als der Weg, dem ein Roboter folgt, der die folgenden Befehle nacheinander für alle Elemente der Folge ausführt:

- Wenn das Symbol 0 ist, dreht die Laufrichtung des Roboters den Winkel h0
- Wenn das Symbol 1 ist, dreht die Laufrichtung des Roboters um den Winkel h1
- In beiden Fällen geht der Roboter einen Schritt vorwärts, der eine gewisse Einheitslänge umfasst.

Traditionell wird ein Roboter, der diese grundlegenden Befehle zum Drehen und Vorwärtsgehen hat, eine Schildkröte (**Turtle**) genannt. Dies geht auf die LOGO-Programmierung in den 1960er Jahren zurück.

EIN ERSTES BEISPIEL

Als erstes Beispiel betrachten wir die unendliche Folge 00000..., die nur aus Nullen besteht, und wählen h0 = 160°.
Da die Folge keine Einsen enthält, spielt der Wert von h1 keine Rolle.
Die Turtle-Grafik besteht aus unendlich vielen Segmenten von einheitlicher Länge, die jeweils nach einer Drehung von 160° entstehen, also jedes Mal einen spitzen Winkel von 20° bilden.
Nachdem die imaginäre Schildkröte dies neun Mal getan hat, hat sie sich insgesamt 9 x 160° gedreht, was ein Vielfaches von 360° ist. Damit befindet sich die Schildkröte wieder an ihrem ursprünglichen Startpunkt. In allen weiteren Schritten werden nur die Segmente überschrieben, die zuvor gezeichnet wurden. Für diese unendliche Folge ist die resultierende Turtle-Grafik also endlich und sieht wie folgt aus:



TURTLE-GRAFIKEN IN LAZARUS ZEICHNEN

Um so eine Turtle-Grafik mit Lazarus zu zeichnen, berechnen wir für jeden Schritt die neue Position (x, y) und benutzen den lineto Befehl, um die Linien zu der neuen Position zu zeichnen.
Neben den Variablen x und y benötigen wir eine Variable h zur Darstellung der Richtung der Schildkröte, eine Variable u für die Längeneinheit und eine Variable n für die Anzahl der Schritte.
Die Variablen x und y müssen real-Variablen sein und sollten auf ganze Zahlen gerundet werden, um die den Befehl lineto Befehl anzuwenden zu können.

Für dieses Beispiel kann das nach der Initialisierung aller Variablen in einer Canvas-Umgebung folgendermaßen aussehen:

```
 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;
```

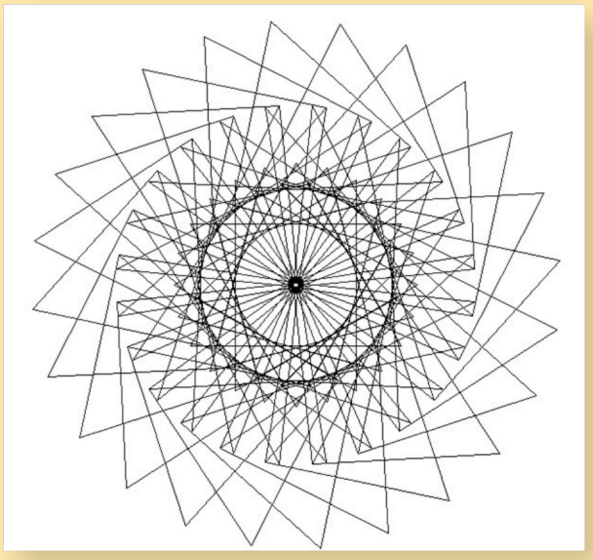
Für dieses Beispiel muss für n eine beliebige Zahl gewählt werden, die mindestens 9 ist.
Für Folgen, die sowohl 0 als auch 1 enthalten, wird der Anfangsteil der Folge der Länge n in einem Array a gespeichert, und in der obigen for-Schleife wird die Einstellung der Variablen h ersetzt durch

```
if a[i] = 0 then h := h + h0 else h := h + h1;
```

PERIODISCHE FOLGEN
Eine Folge heißt periodisch, wenn sie aus unendlich vielen Kopien der gleichen endlichen Folge besteht. Ein Anfangsteil einer periodischen Folge erhält man zum Beispiel durch:

```
    for i := 1 to n do
        if (i mod 9) in [0,2,3,4] then a[i] := 0 else a[i] := 1;
```

Wenn man h0 = 120 und h1 = -147 setzt, dann ergibt das folgende Turtle-Grafik:



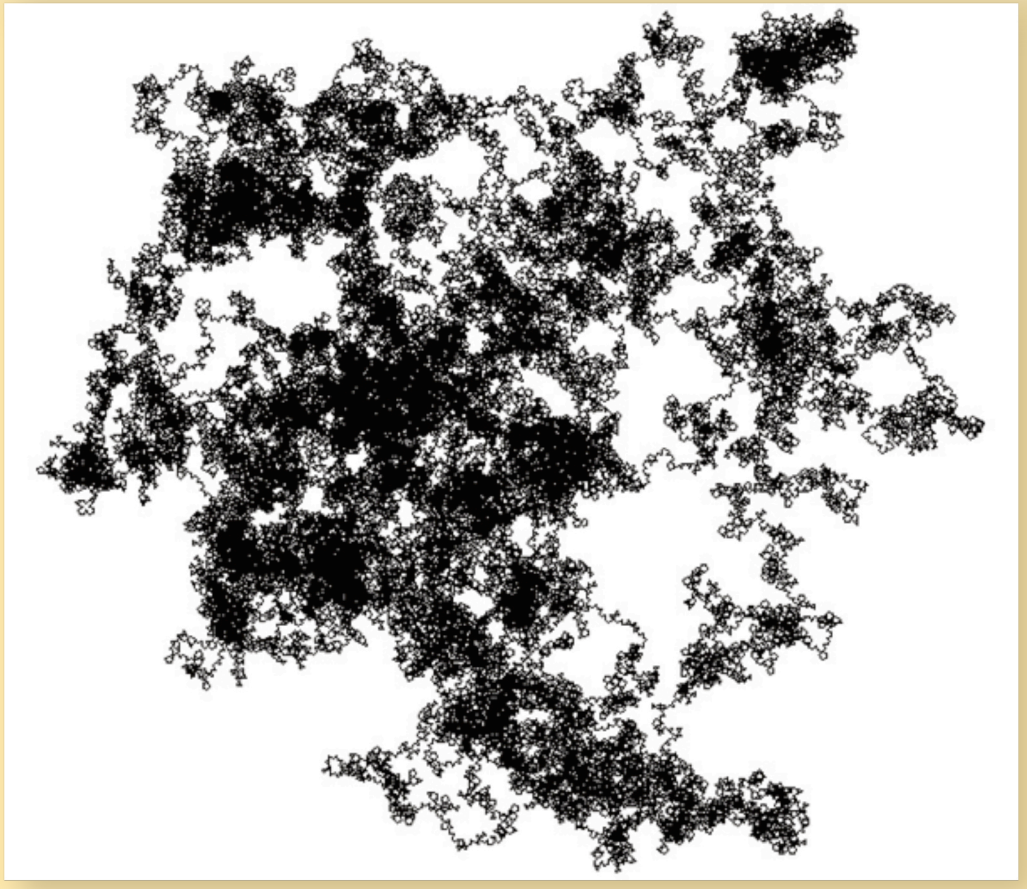
Um wieder am Ausgangspunkt und Startwinkel zu sein, sollte die Turtle-Grafik der kopierten endlichen Folge 100011110 9 mal gezeichnet werden. Das heißt, n muss mindestens 216 sein

NOCH MEHR KOMPLIZIERTE FOLGEN

Periodische Folgen sind sehr strukturiert. Folgen ohne Struktur erhält man durch zufällige Auswahl von a[i], zum Beispiel durch:

```
for i := 1 to n do a[i] := random(2);
```

Wie erwartet, zeigt die resultierende Schildkrötenfigur keine Struktur und sieht typischerweise wie folgt aus:



DIE MORSE-FOLGE

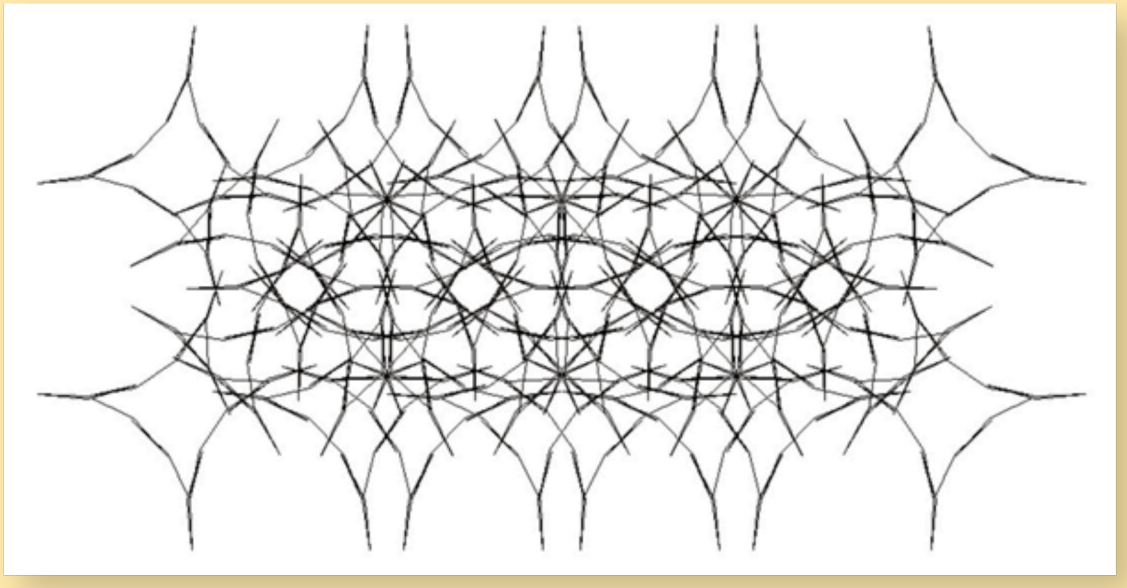
Interessantere Schildkrötenfiguren erhält man aus morphischen Folgen, die durch die Eigenschaft definiert sind, dass sie sich aus selbst ergeben, wenn man einen bestimmten Morphismus f anwendet, d. h. jede 0 auf eine endliche Folge f(0) und jede 1 auf eine endliche Folge f(1) abbildet. Zum Beispiel ist die Morse-Folge

0110100110010110.....

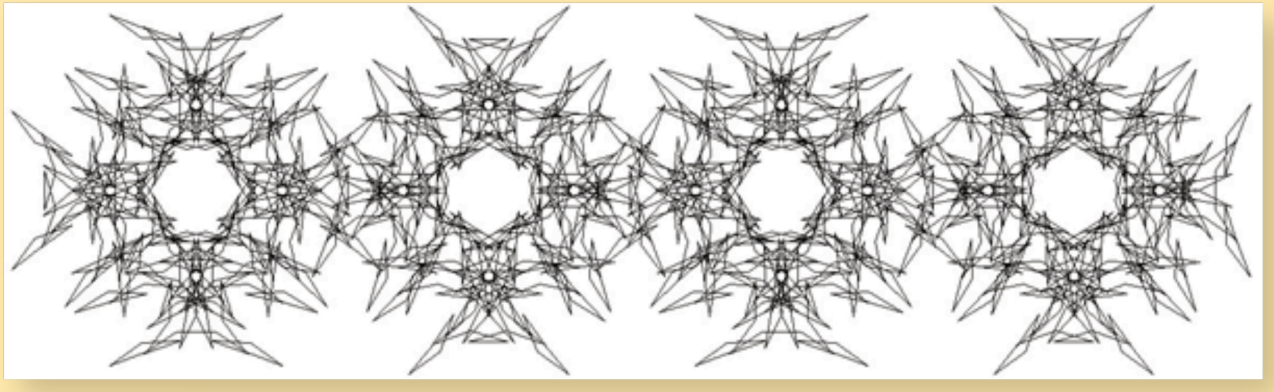
definiert als die morphische Folge, die mit 0 beginnt, und sich auf sich selbst abbildet, indem jede 0 durch f(0) = 01 und jede 1 durch f(1) = 10 ersetzt wird.

Man kann argumentieren (für die Theorie verweisen wir auf [1,2]), dass die Turtle-Grafik der Morse-Folge endlich ist, wenn 2kh0 und 2kh1 für irgendein k Vielfache von 360 Grad sind.

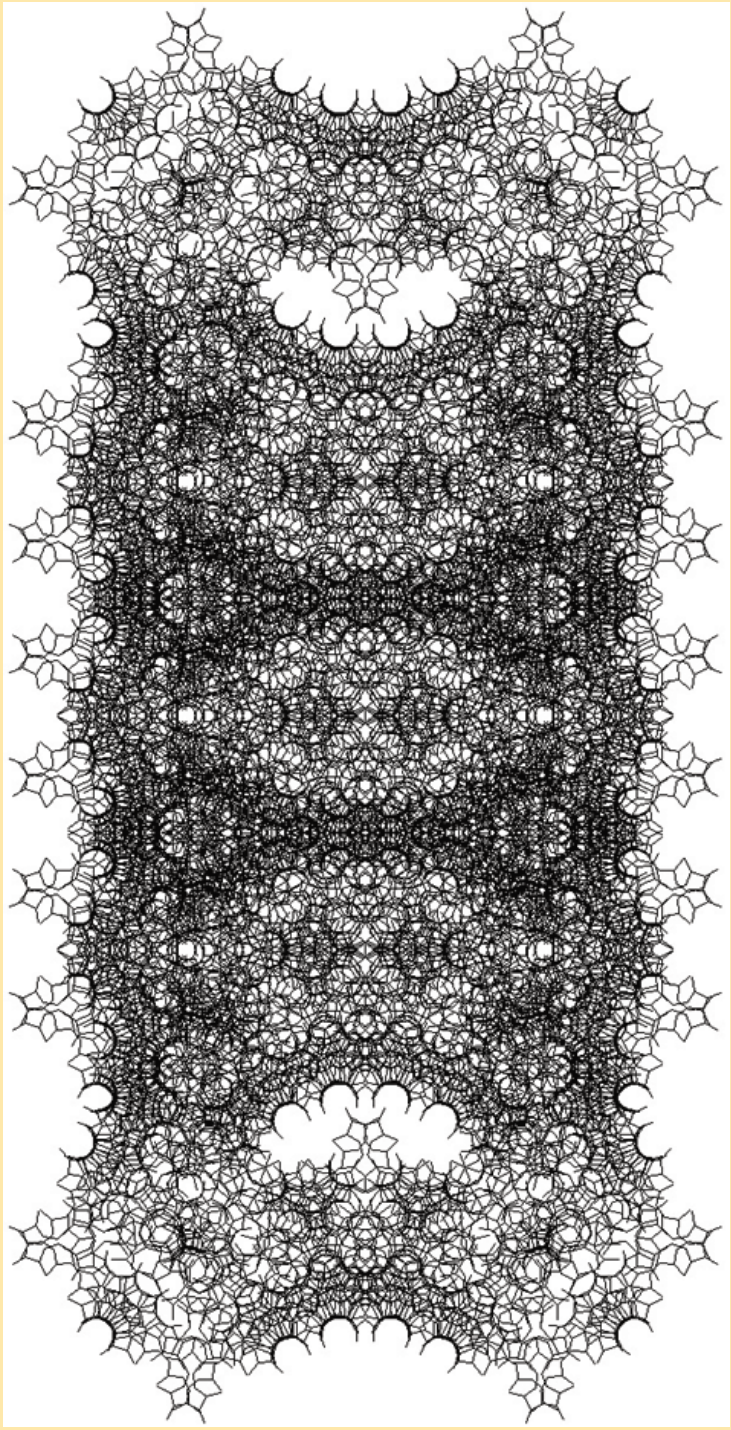
Es stellt sich heraus, dass sie oft schöne Bilder ergeben. Wählt man zum Beispiel h0 = 22,5 = 360/16 und h1 = 177,1875 = 63 x 360/128 ergibt die folgende Schildkrötenfigur:



Wählt man andere Winkel, die diese Bedingung ebenfalls erfüllen, ergeben sich folgende Turtle-Grafiken:



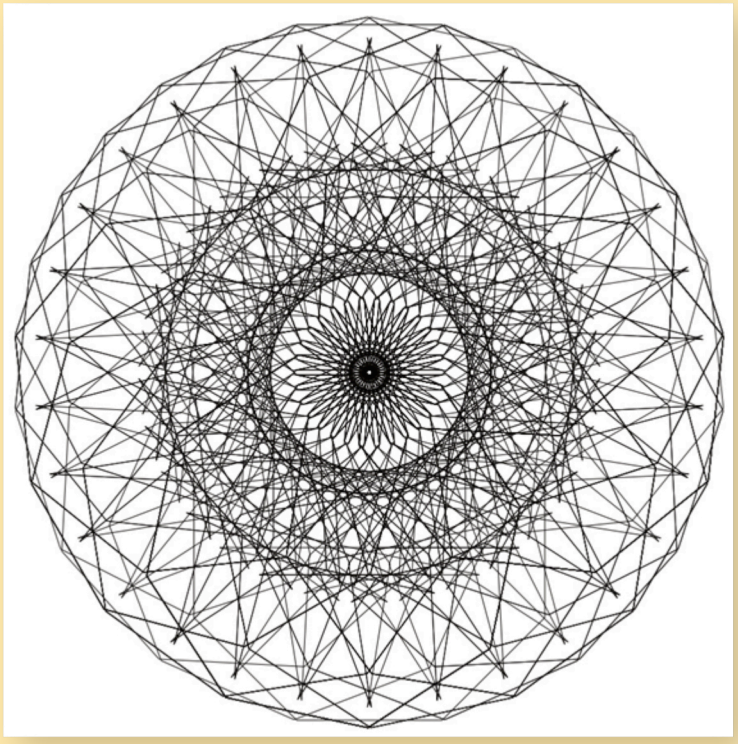
und → *(siehe nächste Seite)*

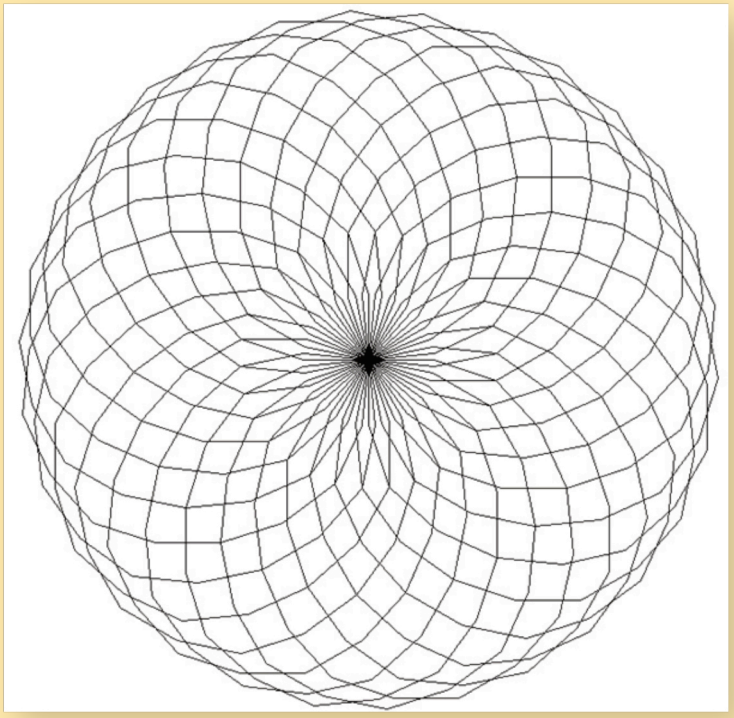


Für alle diese Turtle-Grafiken der Morse-Folge ist die Gesamtzahl der gezeichneten Segmente eine Potenz von 2, für diese drei Beispiele jeweils 210 = 1024, 211 = 2048 und 216 = 65536. Für alle diese Beispiele sollte die Zahl n im Programm mindestens die passende Zahl sein. Das Erzeugen des Anfangsteils einer beliebigen morphischen Reihe kann mit nur wenigen Codezeilen erledigt werden. Für die obigen Beispiele könnte der vollständige Code zum Zeichnen der Turtle-Grafik so aussehen:

```
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;
```

Alle drei genannten Beispiele (und viele weitere) werden durch unterschiedliche Initialisierungen der Variablen n, h0, h1, h, x, y und u erreicht.
Im Wesentlichen ergibt sich das Bild durch die Wahl von n, h0 und h1. Die Anfangswerte von h, x, y und u werden typischerweise gewählt, nachdem man mit diesen Werten herumgespielt hat, bis das resultierende Bild gut auf den Bildschirm passt.





ANDERE MORPHISCHE FOLGEN

Durch die Wahl anderer endlicher Sequenzen f(0) und f(1), wobei f(0) bei 0 beginnen sollte und aus mindestens zwei Symbolen besteht, erhalten wir weitere morphische Folgen, die zu vielen weiteren spannenden Turtle-Grafiken führen.

Einige von ihnen sind endlich, genau wie die Beispiele, die wir für die Morse-Folge gegeben haben.

Für die zugrunde liegende Theorie verweisen wir wiederum auf [1,2]. Als erstes Beispiel betrachten wir f, definiert durch f(0) = 010 und f(1) = 11, was die morphische Folge

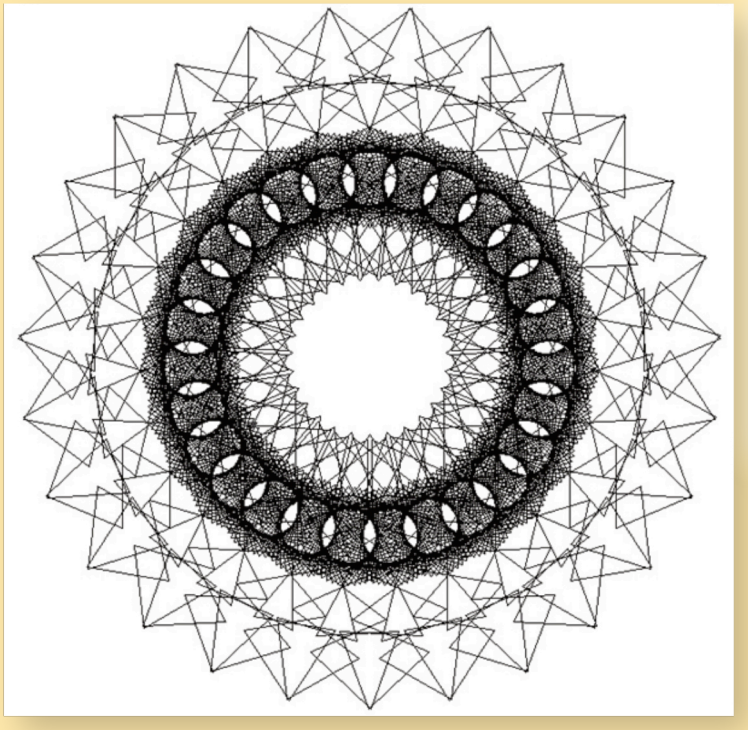
0101101011101011010....

ergibt, die die einzige Folge ist, die bei 0 beginnt und auf sich selbst abbildet, wenn gleichzeitig jede 0 durch f(0) = 010 und jede 1 durch f(1) = 11 ersetzt wird.

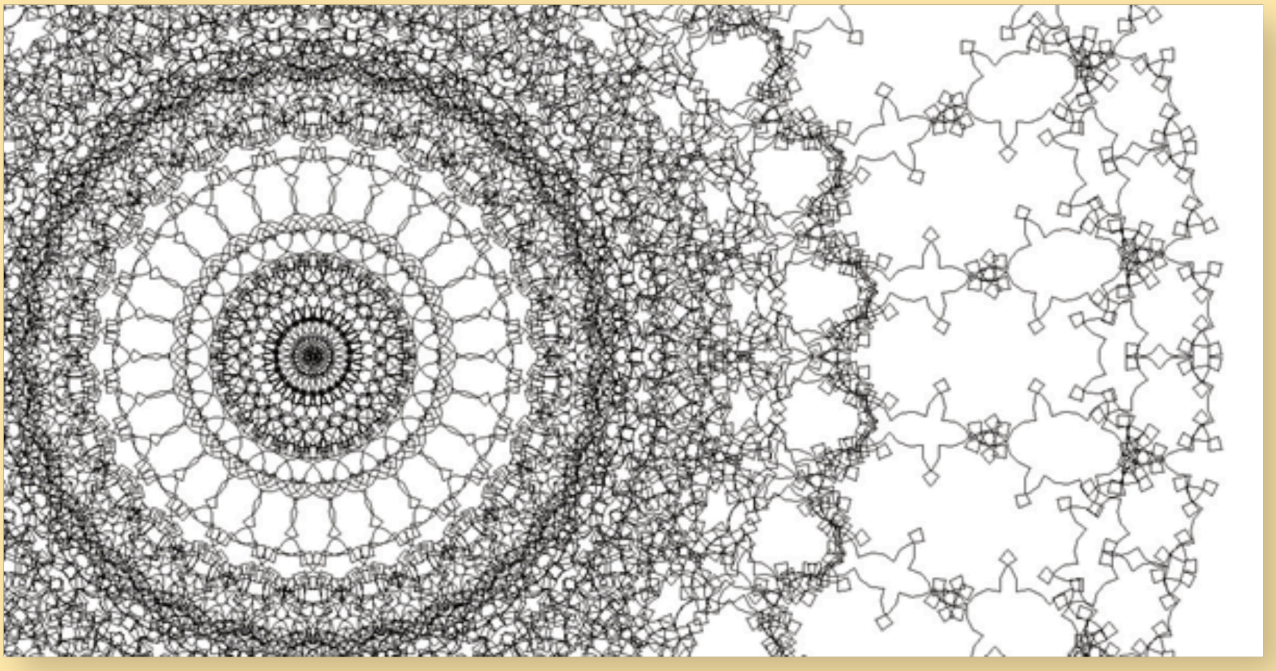
Wählt man h0 = 168° und h1 = -45° und n hinreichend groß, so erhält man die links oben auf dieser Seite dargestellte Turtle-Grafik. Genauer gesagt, besteht diese Turtle-Grafik aus 840 Segmenten.

Wenn man n = 840 wählt, werden nicht alle Segmente gezeichnet, da dann bereits einige Segmente doppelt gezeichnet sind, aber wenn man n = 1300 wählt, erhält man das volle Bild: Dann überzeichnet die Fortsetzung nur die vorhandenen Segmente.

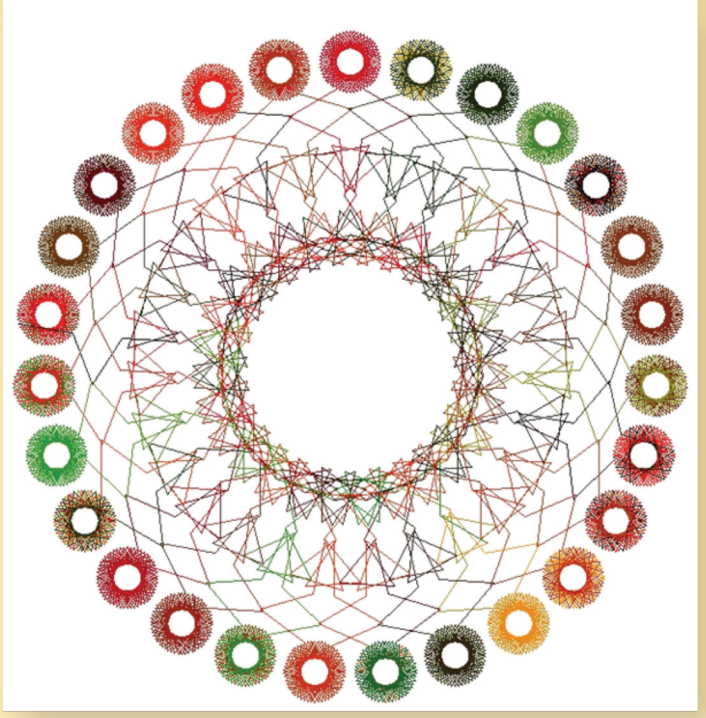
Andere Werte für f(0), f(1), h0 und h1 ergeben die links unten und oben auf der nächsten Seite gezeigten Turtle-Grafiken.



Alle diese Beispiele sind nur kleine Modifikationen des Programms, das wir für die Morse-Folge gegeben haben: Nur die erste Schleife wird modifiziert, um die gewünschte morphische Folge zu erzeugen.
Außerdem müssen die Winkel h0 und h1 so gewählt werden, dass die Voraussetzungen für die Endlichkeitstheoreme aus [1], [2] erfüllt sind und die Anfangswerte von h, x, y und u müssen so gewählt werden, dass die resultierende Figur gut auf den Bildschirm passt.
Wenn man u absichtlich zu groß wählt, zoomt man in die resultierende Figur hinein, wie im unten gezeigten Beispiel.



Ein **Lazarus-Programm** mit einer Benutzeroberfläche zur Eingabe aller Parameter zur Erstellung der von uns angegebenen Turtle-Grafiken, einschließlich aller Quellen und einiger Beispiele, ist verfügbar unter:
`https://github.com/hzantema/turtle-graphics`
Bis jetzt wurde die Stiftfarbe nicht angegeben, wodurch sie standardmäßig schwarz ist. Indem man die Stiftfarbe in der zweiten Schleife mit zunehmendem i verändert, erhält man Bilder wie die folgenden:



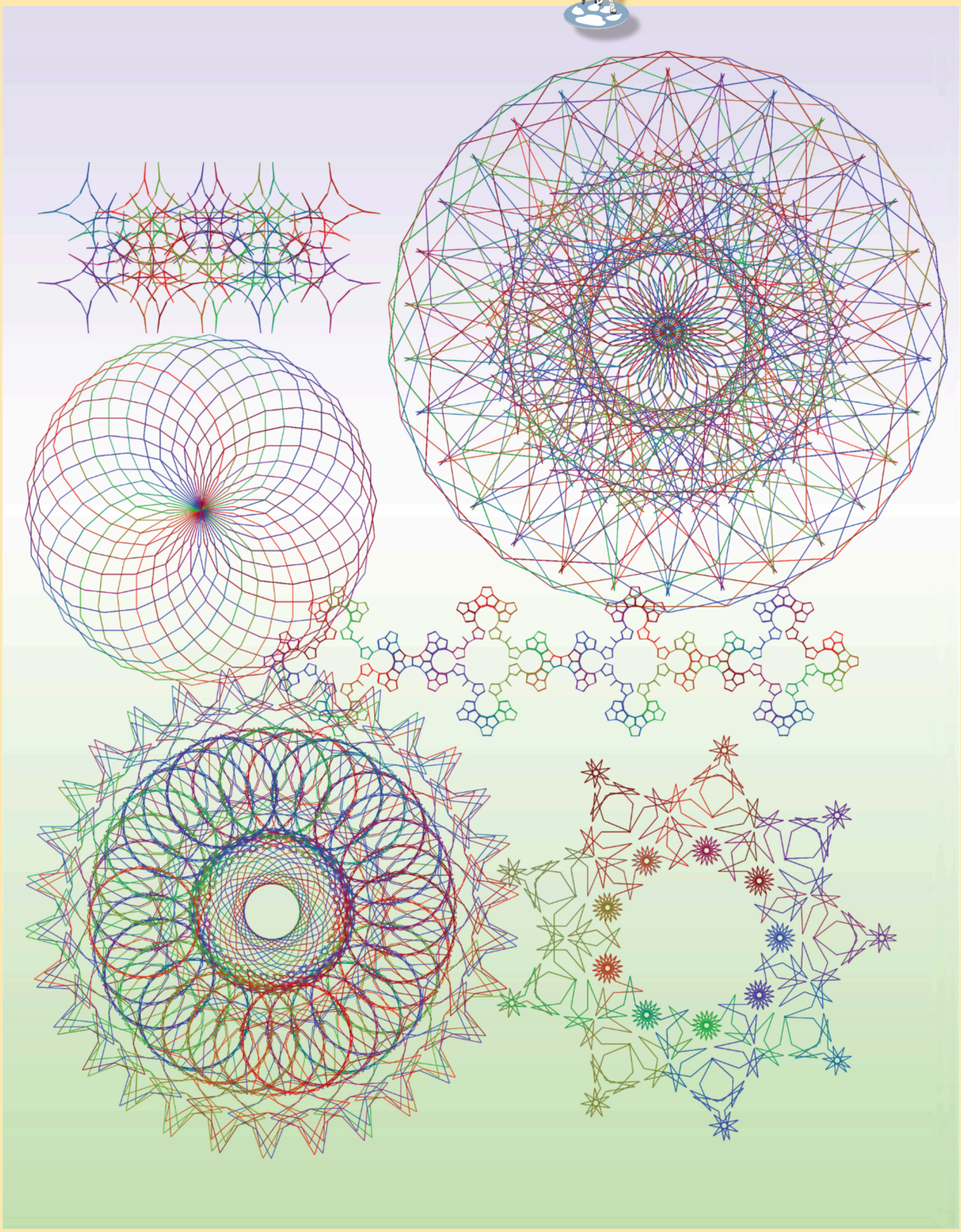
In all diesen Beispielen wird jedes Segment auf das Canvas mit dem Befehl
`lineto(round(x),round(y));` gezeichnet.
Für die reine Darstellung des Bildes auf dem Bildschirm funktioniert dies gut, bietet aber aufgrund der Rundung keine hohe Auflösung. Um in hoher Auflösung auszugeben oder die Möglichkeit zum Zoomen zu bieten, sollte man von diesem Bitmap-Ansatz zu einem vektorbasierten Ansatz wechseln, indem man einfach die Rundungsbefehle entfernt und den Befehl lineto durch den entsprechenden Befehl in einer vektorbasierten Einstellung ersetzt. Dann könnte eine Zusammenstellung von Beispielen, wie sie in diesem Beitrag vorgestellt wurde, folgendermaßen aussehen:
(siehe Abbildung auf Seite 9 des Artikels)

SCHLUSSBEMERKUNG

In diesem Beitrag haben wir gezeigt, wie man mit wenig Code mehrere bemerkenswerte Figuren durch ein frei verfügbares **Lazarus**-Programm zeichnen kann.

LITERATURHINWEISE

[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, und pictures. To appear in 2024 at CRC Press, Taylor und Francis group, around 250 pages. Translated from Dutch version Spelen met oneindigheid: verrassende figuren en patronen, Noordboek 2023



NACHRUF VON PROF. WIRTH

Der Computerpionier Niklaus Wirth ist gestorben Niklaus Wirth, eine herausragende Persönlichkeit auf dem Gebiet der Informatik, ist verstorben.

Niklaus Wirth, eine prominente Persönlichkeit auf dem Gebiet der Informatik, ist am 1. Januar 2024 im Alter von fast 90 Jahren gestorben. Der geschätzte Professor für Informatik an der ETH erlangte weltweite Anerkennung für seine Pionierarbeit bei der Entwicklung der Programmiersprache Pascal im Jahr 1970. Im Jahr 1984 wurde er als einziger deutschsprachiger Informatiker mit dem Turing Award geehrt, der oft als Äquivalent zum Nobelpreis im Bereich der Informatik angesehen wird. Er war ein Empfänger des Turing-Preises, ein Wegbereiter auf dem Gebiet der Informatik und der Schöpfer sehr einflussreicher Programmiersprachen: Niklaus Wirth hat auf dem Gebiet der Informatik bedeutende Beiträge geleistet und beachtliche Erfolge erzielt. Seine bemerkenswerteste Leistung ist die Entwicklung der Programmiersprache Pascal. Sein Einfluss geht jedoch über Pascal hinaus. Niklaus Wirths Bemühungen und sein Engagement haben einen bedeutenden und wesentlichen Beitrag zum weltweiten Fortschritt der Computerwissenschaft geleistet. Bis zum heutigen Tag haben seine Bemühungen einen tiefgreifenden Einfluss auf die Informatik und Generationen von Programmierern. Die Verwandten von Niklaus Wirth haben berichtet, dass er am 1. Januar 2024 in aller Ruhe verstorben ist.

Niklaus Wirth war maßgeblich an der Entstehung der Informatik in der Schweiz beteiligt. Er führte erfolgreich die Fortschritte der Informatik aus den Vereinigten Staaten, der führenden Nation in der Computerentwicklung in dieser Zeit, in die Schweiz ein. Dies trug dazu bei, die Informatik in der Schweiz als eigenständigen Forschungs- und Berufszweig zu etablieren. ETH-Präsident Joël Mesot erinnert sich an Niklaus Wirth als eine bedeutende Persönlichkeit, die nicht nur bahnbrechende Beiträge zur Entwicklung von Programmiersprachen leistete, sondern auch eine Schlüsselrolle bei der Etablierung der Informatik in der Schweiz spielte. Niklaus Wirth war 31 Jahre lang, von 1968 bis 1999, als Professor an der ETH Zürich tätig. Dank seiner und seiner Kollegen unermüdlichen Entschlossenheit erhielt die ETH Zürich 1981 ein eigenständiges Departement für Informatik und einen entsprechenden Studiengang.

Großes Interesse an der Technik von klein auf

Niklaus Wirth, der am 15. Februar 1934 in Winterthur geboren wurde, zeigte schon in jungen Jahren ein ausgeprägtes Interesse an der Technik. In seiner Kindheit beschäftigte er sich aktiv mit dem Bau von Flugzeugen und konstruierte erfolgreich seine ersten Radios und Verstärker. Von seinem Eifer getrieben, schrieb er sich als Student an der ETH Zürich ein. Er absolvierte ein Studium der Elektrotechnik und schloss dieses erfolgreich ab. 1960 machte Wirth seinen Master-Abschluss an der Universität von Laval in Kanada. Seine ersten Erfahrungen mit Computern, Programmiersprachen und Compilern machte er während seiner Zeit an der University of California in Berkeley. Dort wagte er den Einstieg in die Softwarebranche und promovierte 1963 in Berkeley unter der Leitung von Harry Huskey. Seine Forschung konzentrierte sich auf die Erweiterung der Programmiersprache Algol 60.

Nach seinen Stationen als Assistenzprofessor an der Stanford University und der Universität Zürich kam er 1968 als Professor für Informatik an die ETH Zürich zurück. Er lehrte und forschte in diesem Bereich bis 1999. In den Jahren 1976-1977 und 1984-1985 widmete er seine Zeit der Forschung am Palo Alto Research Centre (PARC) von Xerox.

Während seiner 31-jährigen Tätigkeit an der ETH Zürich leistete Niklaus Wirth Pionierarbeit bei der Entwicklung mehrerer neuer Programmiersprachen, darunter Euler, PL360, Algol W, Pascal, Modula, Modula 2, Oberon und LoLa. Darüber hinaus konstruierte er die ersten Personal Computer (PCs) in der Schweiz und bildete die erste Gruppe von Schweizer Informatikern aus. Darüber hinaus hat er zahlreiche kanonische Publikationen verfasst, die übersetzt und weltweit verbreitet wurden. Er wurde mit zahlreichen Auszeichnungen geehrt, darunter der angesehene ACM Turing Award, den er 1984 als bisher einziger deutschsprachiger Informatiker erhielt. Im Jahr 1988 wurde er mit dem IEEE Computer Pioneer Award geehrt. Nach Niklaus Wirth ist das so genannte Wirthsche Gesetz benannt, das besagt, dass die Geschwindigkeit, mit der sich die Softwareleistung verschlechtert, größer ist als die Geschwindigkeit, mit der sich die Hardwareleistung verbessert.

Pascal - die Suche nach einer einflussreichen und unkomplizierten Programmiersprache

Im Jahr 1984 gab es wichtige Ereignisse im Bereich der Informatik und für Niklaus Wirth. Apple brachte den Macintosh PC auf den Markt, IBM stellte den IBM Personal Computer/AT vor und Niklaus Wirth wurde mit dem Turing Award geehrt, der prestigeträchtigsten Auszeichnung in der Informatik, vergleichbar mit einem Nobelpreis in den Naturwissenschaften oder der Fields-Medaille in der Mathematik. Wirth wurde für seine Beiträge zur Entwicklung mehrerer Programmiersprachen geehrt, darunter Euler, Algol W, Modula und Pascal.

Pascal ist der bekannteste Beitrag von Niklaus Wirth auf dem Gebiet der Programmiersprachen. Der Hauptvorteil ist seine Geradlinigkeit und Raffinesse. Pascal basiert auf den expliziten Prinzipien der strukturellen Programmierung, wie sie von dem Informatiker Edsger W. Dijkstra formuliert wurden. Sie basiert außerdem auf einer mathematischen Grundlage, die von dem Informatiker Tony Hoare definiert wurde, und enthält Niklaus Wirths Implementierung der Algol W-Konzepte. Diese Sprache integriert effektiv solide Programmierprinzipien mit strukturierter Programmierung und Datenorganisation. Folglich gewann sie schnell an Popularität als Bildungssprache. Pascal war die Programmiersprache, die viele Generationen von Studenten, auch an der ETH Zürich, für ihre ersten Programmiererfahrungen an den Universitäten weltweit nutzten.

Niklaus Wirth war immer bestrebt, neue Herausforderungen anzunehmen und gab sich nie mit seinen bisherigen Erfolgen zufrieden. Pascal ist zwar weithin als seine berühmteste Errungenschaft bekannt, aber seine Beiträge gehen darüber hinaus. Er hat auch die Nachfolgesprache Modula-2, das Oberon-System und die "Lilith"-Workstation entwickelt, die als Vorläufer der späteren Personal Computer diente. Wirth widmete sein ganzes Leben der kontinuierlichen Weiterentwicklung und Verbesserung seiner Programmiersprachen. Der Übergang von Euler zu Oberon markierte die Entwicklung einer Sprache, die die Objektorientierung und die Typenhierarchie einbezog. Oberon zielte darauf ab, gleichzeitig maximale Stärke und Einfachheit zu erreichen. Niklaus Wirth war bestrebt, eine Erfindung zu schaffen, die sich an die breite Bevölkerung wendet, und zwar nach dem Prinzip, kostengünstig und verständlich zu sein.

Oberon war mehr als nur eine Sprache. Das Ergebnis war ein ganzes System und schließlich wurde das Buch "Projekt Oberon" veröffentlicht, in dem die Software, die Sprache und die Hardware auf rund 500 Seiten beschrieben werden - der ganze Stolz seiner Arbeit: "Ich verfolgte das lebenslange Ziel, eine Sprache zu entwickeln, die so mächtig wie möglich, aber so einfach wie möglich ist. "Oberon stellt die letzte Stufe in dieser Entwicklungsreihe dar", so Niklaus Wirth.

Lilith - und ihr Engagement für die Informatik in der Schweiz

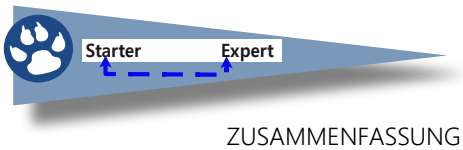
Die Schweiz nimmt derzeit weltweit eine bedeutende Stellung im Bereich der Informatik ein und hat zahlreiche wesentliche Beiträge sowohl zu den theoretischen Grundlagen als auch zur praktischen Umsetzung der Disziplin geleistet. Bis in die 1970er Jahre hinein vollzog sich ein Wandel der Verhältnisse: Obwohl die Vereinigten Staaten bereits die ersten Workstations entwickelt hatten und die Informatik weithin studiert wurde, war die Schweiz sowohl in der Ausbildung als auch in der praktischen Umsetzung im Rückstand. Ein Beispiel dafür ist die Lilith von Wirth, die erst nach einigen Jahren die Aufmerksamkeit der Industrie erregte.

Lilith war eine frühe Computer-Workstation, die über einen hochauflösenden Bildschirm und eine Maus verfügte und damit ein Vorläufer der modernen Personal Computer war. Niklaus Wirth entwickelte sie 1980 an der ETH als Grundlage für mehrere Forschungssoftwareprojekte. Ab 1982 bemühten sich Akademiker der ETH, das System zu Geld zu machen, aber ihre Versuche waren nicht erfolgreich. Der PC wurde in den Vereinigten Staaten industriell entwickelt. Nichtsdestotrotz hatte Lilith einen tiefgreifenden Einfluss auf eine ganze Kohorte von Informatikern. Niklaus Wirth entwickelte 1986 das Computersystem Ceres, das auf seine Arbeit an Lilith folgte. Dieses System umfasste das Betriebssystem Oberon und die Programmiersprache Oberon. Die Ceres-Computer wurden bis etwa 2003 für die Ausbildung von Informatikstudenten an der ETH Zürich eingesetzt.

Auch die Entwicklung der Informatik an der ETH und in der Schweiz verlief nicht linear: Niklaus Wirth und seine Kollegen mussten zunächst einige Hindernisse überwinden. In den frühen 1970er Jahren unternahmen sie den Versuch, die Informatik als eigenständige akademische Disziplin zu etablieren. Allerdings erwiesen sich sowohl die ersten als auch die nachfolgenden Bemühungen als erfolglos. Als Reaktion auf den offensichtlichen Mangel an Informatikern in der Schweiz gründete die ETH Zürich 1981 schließlich die Informatik als Departement und Studiengang. Das Engagement von Niklaus Wirth und seinen Kollegen legte den Grundstein für die Entwicklung der Informatik in der Schweiz.

MIT DELPHI MITHALTEN IM FPC

VON MICHAEL VAN CANNEYT



ZUSAMMENFASSUNG

Delphi Kompatibilität war immer wichtig für **Free Pascal**.
Allerdings fehlte diese Kompatibilität in einigen Bereichen.
In letzter Zeit wurde viel Arbeit geleistet, um die Delphi-Kompatibilität von FPC zu verbessern.

① EINLEITUNG

Das Free Pascal Compiler Team hat die Delphi Kompatibilität immer als ein wichtiges Merkmal des Free Pascal Compilers betrachtet.
Diese Kompatibilität gilt in erster Linie für die Sprache **Pascal**.
Wenn **Delphi** sie kompilieren kann, dann **sollte es auch Free Pascal können**.
Lange Zeit war Delphi 7 die Version, zu der Free Pascal am meisten kompatibel war.
Aber natürlich hat sich Delphi weiterentwickelt:
Moderne Sprach-Features wie **Generics**, **anonyme Funktionen**, **Attribute** und **erweitertes RTTI** wurden hinzugefügt.
Diese Features wurden in FPC entwickelt - manche schon seit längerer Zeit, manche erst kürzlich, aber viele dieser Features sind noch nicht in einer offiziell freigegebene Version von Free Pascal enthalten.
Um sie zu nutzen, müssen Sie die **Entwicklungsversion (Trunk)** von **FPC** verwenden.
Die Delphi-Kompatibilität gilt auch für die grundlegenden Units, die mit Free Pascal ausgeliefert werden.
Die grundlegenden **RTL-Units von Delphi** finden sich auch in **Free Pascal** wieder.
Natürlich entwickeln sich Delphi und Free Pascal im Laufe der Zeit weiter.
Einige der Unterschiede in den **RTL-Units** machen es aber schwierig, **Code sowohl in Free Pascal als auch in Delphi lauffähig zu halten:**

- **Einige Basis-Units sind nicht vorhanden.**
  Die Anzahl der Units in der **Delphi-RTL** hat sich erheblich erweitert.
  Grundlegende Funktionalität wird in **`System.IOUtils`**, **`System.JSON`**, **`System.Threading`** usw. angeboten. **Diese Units wurden kürzlich zu FPC hinzugefügt.**
- Delphi ist dazu übergegangen, **durch Punkte getrennte Namen für Units zu verwenden:** die Präfixe `System`, `Data`, `Vcl`, `FMX` werden seit vielen Jahren in der gesamten Delphi-Codebasis verwendet.
- Delphi hat seit vielen Jahren den ‚**string**' **Alias-Typ** geändert, so dass er ein Alias für einen **UnicodeString** ist, ein **String-Typ**, bei dem **jedes Element des Strings** (ein Zeichen) ein **UTF-16-Zeichen ist** - er verwendet 2 Bytes.

Natürlich muss das Free Pascal Team immer mit Delphi mithalten
- die Anforderung der Kompatibilität geht nur in eine Richtung.

Dies hat das FPC-Team vor ein Dilemma gestellt:
Zum einen möchte FPC mit sich selbst **abwärtskompatibel** bleiben.
Etwas, das als ebenso wichtig (*wenn nicht sogar wichtiger*) erachtet wird als die **Delphi-Kompatibilität.**
Die Umstellung auf den String-Typ und die Verwendung von Namespaces in den Unit-Namen **bricht eindeutig die Abwärtskompatibilität**.

WIE LÖST MAN DAS?

② DIE LÖSUNG

Nach vielen Jahren wurde nun **eine Lösung für dieses Dilemma** gefunden.
Die Idee ist einfach: **Man benutzt dieselbe Codebasis, um den gesamten Code von FPC RTL und Packages zweimal neu zu kompilieren.**

- Eine Kompilierung erfolgt mit Einstellungen, die mit FPC selbst rückwärts kompatibel sind: **keine gepunkteten Namen, String ist der 1-Byte-String**.
- Ein zweiter Kompiliervorgang wird mit Einstellungen durchgeführt, die gepunktete Namen erzeugen, und bei denen string der 2-Byte-String ist.

Daraus ergeben sich **2 Sätze von Units**, die nicht gemischt werden können.
Der Benutzer muss wählen welchen Satz von Einheiten er verwenden möchte:
* Der **Free Pascal abwärtskompatible** Satz von Units,
* Der **"Jüngstes Delphi"-kompatible** Satz von Units.

Der letztere synchronisiert die ‚`unicode rtl`'.

Es gibt noch eine dritte Möglichkeit:
Alles mit persönlichen Einstellungen kompilieren.
Dazu später mehr. Um diese Lösung zu implementieren, waren einige Arbeiten am Compiler erforderlich:

- Der Compiler musste seine Auffassung darüber ändern, was der **Grundtyp Char ist**.
  Bis vor kurzem war dieser als 1-Byte-Zeichen hartkodiert und **AnsiChar** war ein **Alias** für **Char**.
  Jetzt ist **AnsiChar** der Basistyp und **Char** ist ein in der **Systemeinheit definierter Alias**.
  Damit kann das Schlüsselwort **String** entweder ein **AnsiString** oder ein **UnicodeString** sein und Char entspricht immer dem Typ eines einzelnen Zeichens in einem String.
- Ein **Compiler 'target'** wurde bis vor kurzem als eine Kombination aus der Ziel-CPU und dem Zielbetriebssystem definiert.
  Das bedeutet, dass zum Beispiel **i386-win32** und **x86 64-win64** 2 Windows-Plattformen sind: Eine 32-Bit und eine 64-Bit.
  Linux kennt noch mehr Plattformen, so wie **Darwin** (macos) bereits 3 Plattformen hat.
  Diese Definition von Plattform musste erweitert werden und der Begriff '**SubTarget**' wurde im Compiler eingeführt.

Beide Änderungen haben maßgeblich dazu beigetragen, die beiden Sätze von Units für den Endbenutzer zu erstellen.

③ DAS SUBTARGET

Der Begriff des Compiler-Targets wurde erweitert. Er muss eine CPU, ein Betriebssystem und optional auch eine beliebige Menge von Einstellungen kapseln.
Für die Delphi-kompatiblen Unicode-RTL wäre dieser "beliebige Satz von Einstellungen":

- nutze **UTF16** strings,
- nutze **dotted** names.

Das Definieren eines **SubTargets** bedeutet einfach, dass wir diesem **Satz von Einstellungen** einen Namen geben. Das SubTarget wird dem Compiler immer mit einem Befehlszeilenschalter angegeben: **`-t`**.
Wenn Sie also eine Datei für das Unterziel "**`unicodertl`**" kompilieren möchten, würden Sie Folgendes angeben

```
fpc -Mdelphi -tunicodetrl myproject.pas
```

Der Name des Unterziels wird dann auf verschiedene Weise verwendet:
Die erste Art und Weise, wie der SubTarget-Name verwendet wird, ist im Compiler.
Wenn der Compiler für ein bestimmtes Ziel kompiliert, definiert er ein **Makro fpctarget**.
Der Wert dieses Makros ist die **Kombination aus Ziel-CPU und Betriebssystem**,
getrennt durch einen Bindestrich.

Fortsetzung Kapitel 3

Dieses Makro wird in der Konfigurationsdatei des Compilers verwendet:

```
-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
```

Wenn also die **Compiler-Version 3.1.1** für **i386-linux** kompiliert,
"sieht" der Compiler die folgende Konfiguration:

```
-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
```

Nun wird das optionale '**SubTarget**' eingeführt, und eine der Auswirkungen seiner Verwendung ist, dass es den Namen des Unterziels zum Makro fpctarget hinzufügt.
Das heißt, wenn der **Compiler Version 3.1.1** für **i386-linux** und SubTarget '**unicodertl**' kompiliert, wird die Konfiguration so:

```
-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
```

Die Suchpfade des Compilers sind je nach SubTarget plötzlich **unterschiedlich**.
Wie bereits gesagt, ist das **SubTarget** ein Name für einen Satz von Einstellungen.

Dies wird durch den zweiten Effekt der Verwendung von SubTarget konkretisiert und erklärt auch, warum es auf der Kommandozeile gesetzt werden muss:
Bei der Suche nach der Compiler-Konfiguration sucht der Compiler auch nach einer Konfigurationsdatei, die das SubTarget in ihrem Namen enthält.
Beim Kompilieren mit SubTarget unicodertl
`fpc -Delphi -tunicodetrl myproject.pas`
wird der Compiler an der selben Stelle, wo er nach fpc.cfg sucht auch nach der Datei **`fpc-unicodertl.cfg`** suchen.
Auf Unix-ähnlichen Plattformen wird auch nach der üblichen "versteckten" Datei im Home-Verzeichnis des Benutzers gesucht:
Diese zweite Konfigurationsdatei enthält eine Reihe von Einstellungen, die das SubTarget definieren.

Bei dieser zusätzlichen Konfigurationsdatei gibt es zwei Dinge zu beachten:

- **Sie muss existieren.**
  Wenn sie nicht vorhanden ist, zeigt der Compiler einen Fehler an (*sie kann aber auch leer sein*).
- **Sie wird immer geladen**,
  auch wenn die Option, die Standardkonfigurationsdateien nicht zu laden **`(-n)`** angegeben ist.

ALSO WIE WIRD DAS BENUTZT UM EINE DELPHI KOMPATIBLE RTL ZU BAUEN?

Eine Konfiguationsdatei mit dem Namen `fpc-unicodertl.cfg` wurde
mit folgendem Inhalt angelegt:

```
-dUNICODERTL
-Municodestrings
```

Diese Konfigurationsdatei wird dann benutzt, um die **RTL** und die Packages zu compilieren.
Es definiert **`UNICODERTL`** und es spezifiziert den **Unicodestrings Modeswitch**,
der den Compiler anweist, dass String als **UnicodeString** interpretiert werden muss.
Natürlich enthalten viele Units Low-Level-Code, bei dem die Größe des Char-Typs sehr wichtig ist.
Die Quellen wurden, wo nötig, geändert, so dass der Code kompiliert und mit beiden Definitionen der Typen String und Char korrekt funktioniert.

Fortsetzung Kapitel 3

Nicht nur die Quellen wurden geändert, sondern auch der Kompilier- und Installationsprozess wurde leicht modifiziert. Wir haben bereits gesehen, dass bei der Angabe von -tunicodertl auf dem Zielsystem i386-linux der Compiler nach kompilierten Units in den folgenden Verzeichnissen sucht:

```
-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
```

Das bedeutet, dass die Units bei der Installation der kompilierten Units auch dort installiert werden müssen.
Die **Makefiles** für die **RTL** und die Packages wurden entsprechend angepasst.
Es gibt jetzt eine Variable **SUB_TARGET**, die die **Makefiles** so konfiguriert, dass sie den Compiler mit dem angegebenen **SubTarget** aufruft. **Bei der Installation von Units wird der SubTarget-Name an den Namen des Verzeichnisses angehängt**, in das die Units installiert werden.

④ NAMESPACES IN DATEINAMEN BENUTZEN

Um Units **mit** Namespaces und die gleichen Units **ohne** Namespaces zu erzeugen,
wird ein **kleiner Trick** angewendet. Es sollte klar sein, dass das **Free Pascal Team** nicht zwei Sätze von Units pflegen kann.

Die jetzige Lösung verwendet einen Satz von Dateien, benutzt aber einen **Include-Mechanismus**, um die zweite (**namespaced**) Datei zu erzeugen.
Hier als Beispiel die Datei für die **'System.Math' Unit:**

```
unit System.Math;

{$DEFINE FPC_DOTTEDUNITS}
{$i math.pp}
```

Die DATEI **math.pp** ist geändert worden

```
{$IFNDEF FPC_DOTTEDUNITS}
unit Math;
{$ENDIF FPC_DOTTEDUNITS}

interface
uses

{$IFDEF FPC_DOTTEDUNITS}
  System.SysUtils;
{$ELSE FPC_DOTTEDUNITS}
  sysutils;
{$ENDIF FPC_DOTTEDUNITS}
```

BEACHTE, DASS DIE USES KLAUSEL GEÄNDERT WURDE.
Natürlich sind weitere Änderungen erforderlich:

Wenn beispielsweise ein voll qualifizierter Bezeichner verwendet wird (*d. h. ein Bezeichner, der den Namen der Einheit enthält*), musste der Name korrigiert werden.
Dieses System ermöglicht es dem FPC-Team, eine RTL mit namenstragenden Dateinamen und eine RTL mit rückwärtskompatiblen Dateinamen zu kompilieren.

Die obige Vorgehensweise wurde für alle von FPC gelieferten Einheiten durchgeführt.

- Für jede Unit wurde eine Version **mit Namespaces** erstellt.
- Für Units, die in **Delphi** existieren, ist der **namengebende Dateiname** identisch mit dem **Namen** in **Delphi**.
- Für Units, die **keine Entsprechung in Delphi** haben, wurde die Gelegenheit genutzt, die Dateinamen konsistenter zu gestalten.

Alle makefiles und **fpmake-Programm**e in der Free Pascal-Distribution wurden so angepasst, dass sie die Units mit Namensraum kompilieren, wenn die Option **FPC_DOTTEDUNITS=1** in der make-Kommandozeile angegeben wird.

Fortsetzung Kapitel 4

Bis jetzt war es die Regel des Free Pascal Teams, die Unit-Dateinamen auf Dateisystemen, die **Groß- und Kleinschreibung** unterscheiden, **klein zu schreiben.**
Das bedeutete, dass man in der uses Klausel schlampig sein konnte.
Die **Groß-/Kleinschreibung** des Unit-Namens **spielte keine Rolle**, da der Compiler nicht nur nach einer Datei mit der gleichen Groß-/Kleinschreibung wie in der uses Klausel suchte, **sondern auch nach einer klein geschriebenen Version der Datei.**

Diese Praxis wurde für Units mit Namespaces aufgegeben.
Auf Dateisystemen, die Groß- und Kleinschreibung unterscheiden,
muss der Unit-Name nun die korrekte Groß- und Kleinschreibung aufweisen.

WIE FINDE ICH DIE NAMESPACED VERSION EINES UNIT NAMENS?

Es gibt eine Datei, die für jede Unit **ohne Namespaces** den Namen der Unit **mit Namespaces** angibt.

Diese Datei kann verwendet werden, um die Namen der Einheiten zu ändern.
Sie brauchen dies nicht manuell zu tun, **es gibt ein Tool, das dies für Sie erledigt.**
Mehr über dieses Tool später.
Dem aufmerksamen Leser wird aufgefallen sein, dass die Erzeugung von **Units mit Namespaces** nicht mit dem das neu eingeführte Konzept der **SubTargets** geht. In der Tat sind die beiden Funktionen unvereinbar.
Der Grund dafür ist, dass es kein **"spezielles" SubTarget** gibt. Sie können so viele SubTargets erstellen, wie Sie wollen und für jedes SubTarget können Sie eine Version der RTL mit Namespaces oder eine rückwärts-kompatible Version erstellen. Das bedeutet, dass es möglich ist, eine RTL ohne Namespaces, aber mit String gleich UnicodeString oder eine RTL mit Namespaces und String=AnsiString zu erzeugen. Die Wahl des FPC-Teams wird durch die Anforderung bestimmt, eine FPC-kompatible RTL und eine Delphi-kompatible RTL zu haben.

⑤ ERSTELLEN DER DELPHI_KOMPATIBLEN RTL

Mit der nächsten **Hauptversion von Free Pascal** wird das FPC-Team die zwei oben erwähnten RTLs erstellen. Aber Sie können diese erhöhte **Delphi-Kompatibilität** schon heute genießen, indem Sie den Compiler und die Delphi-kompatible RTL selbst kompilieren.

WIE GEHT DAS?
Erstens muss man **git installieren** und die **Compilersources** downloaden.
Dieser Vorgehensweise wurde in **früheren Artikeln** (**97, Seite 34 | 98, Seite50 | 99/100 Seite 36**) bereits ausführlich behandelt. Angenommen **git** ist installiert und im **PATH**, dann kann man in einem **Terminalfenster** (*auf der Kommandozeile*) die FPC Sources klonen:

```
git clone https://gitlab.com/freepascal.org/fpc/source.git fpc
```

Danach können Sie das **neueste FPC kompilieren und installieren**:

```
cd fpc
make clean all PP=/path/to/FPC/3.2.2
make install PP=/path/to/FPC/3.2.2
```

Die **PP-Variable** muss auf das **fpc-Binary** der **FPC-Version 3.2.2** verweisen (*dies ist eine Voraussetzung*).
Wo dieses fpc-Binary installiert ist, hängt von Ihrem System ab. Obiges kompiliert und installiert die Version **3.3.1 von FPC**. **Sie können dafür auch FPCUpdeluxe verwenden,** Installations video auf unserer website: **`https://www.blaisepascalmagazine.eu/fpcupdeluxe-install-lazarus-stable-trunk-cross-compiling-raspberrypi-versions/`** oder **`https://youtu.be/Eacb7PvoV8A`**
Sobald dies geschehen ist, sollten Sie sich notieren, wo die neue **Version von FPC 3.3.1** installiert wurde.
Der nächste Schritt ist die Erstellung der Konfigurationsdatei **`fpc-unicodertl.cfg`** an einem der Orten, an denen der Compiler nach der Konfigurationsdatei sucht:

```
-dUNICODERTL
-Municodestrings
```

Am einfachsten ist es, nach der bestehenden **`fpc.cfg`** zu suchen und die obige Datei direkt dort zu erstellen.

Fortsetzung Kapitel 5

Dann werden im gleichen **Terminal**, in dem die vorherigen Befehle eingegeben wurden, die folgenden make-Befehle das **Unicode-RTL** erzeugen und installieren. Diese Befehle müssen im **selben Verzeichnis ausgeführt** werden wie die vorherigen 'make'-Befehle.

```
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
```

Sobald dies geschehen ist, können Sie aktuellen Delphi-Code mit dem neuen Befehlszeilenschalter **'-tunicodertl'** kompilieren.

❻ CODE KONVERTIEREN UM NAMESPACED UNITS ZU NUTZEN

WAS IST, WENN SIE IHREN CODE AKTUALISIEREN UND DIE DELPHI-KOMPATIBLE RTL VERWENDEN MÖCHTEN?

Wie auf *Seite 5 dieses Artikels* dargestellt, muss die `uses clause` eines Projekts geändert werden. Wenn Sie viele Units haben, bedeutet dies eine Menge Arbeit. Glücklicherweise gibt es Tools zur Verfügung zur **Erstellung von FPC-Units mit doppeltem Verwendungszweck.**

Im Verzeichnisbaum der FPC Sources, im **Verzeichnis utils/dotutils**, gibt es ein Programm namens `prefixunits`. Sie können es auf der Kommandozeile oder mit Lazarus kompilieren und es benutzen, um die Units für das neue Verfahren zu konvertieren.

Das folgende Kommando konvertiert die Unit **`myunit.pas`** zu **`company.myunit.pas`** mit dem gleichen Verfahren wie oben beschrieben:

`prefixunits -b -k known.txt c:\Temp\myunit.pas -n company`

Eine neue Datei **`varcompany.myunit.pas`** wird erzeugt, welche **`myunit.pas`** enthält.
Die uses Klausel in **`myunit.pas`** wird neu erstellt und enthält nun die bedingten Definitionen wie oben beschrieben.
Der **-b Schalter** erzeugt ein Backup, **-k** muss genutzt werden, um die Datei zu bestimmen, welche die alten Unit Namen den neuen Namen zuordnet. Diese Datei befindet sich im dotutils directory, nahe bei den Quelltexten vom prefixunits Tool.
Sie können sich auch dafür entscheiden, einfach nur die neuen Unit Namen zu verwenden.
Dann geben Sie die **'-r' Option** (for 'replace') an:

`prefixunits -r -b -k known.txt c:\Temp\myunit.pas -n company`

In diesem Fall wird keine neue Datei erstellt und die uses Klausel in **`myunit.pas`** wird ersetzt durch eine Unit Klausel ersetzt, die nur die **dotted Unit Namen** verwendet.
Dieses Werkzeug ist etwas rudimentär, aber es tut was es tun soll. Zweifellos wird die Lazarus IDE mit der Zeit um ein nettes GUI Werkzeug erweitert werden das die gleiche Aufgabe für alle Dateien in ihrem Projekt durchführt.

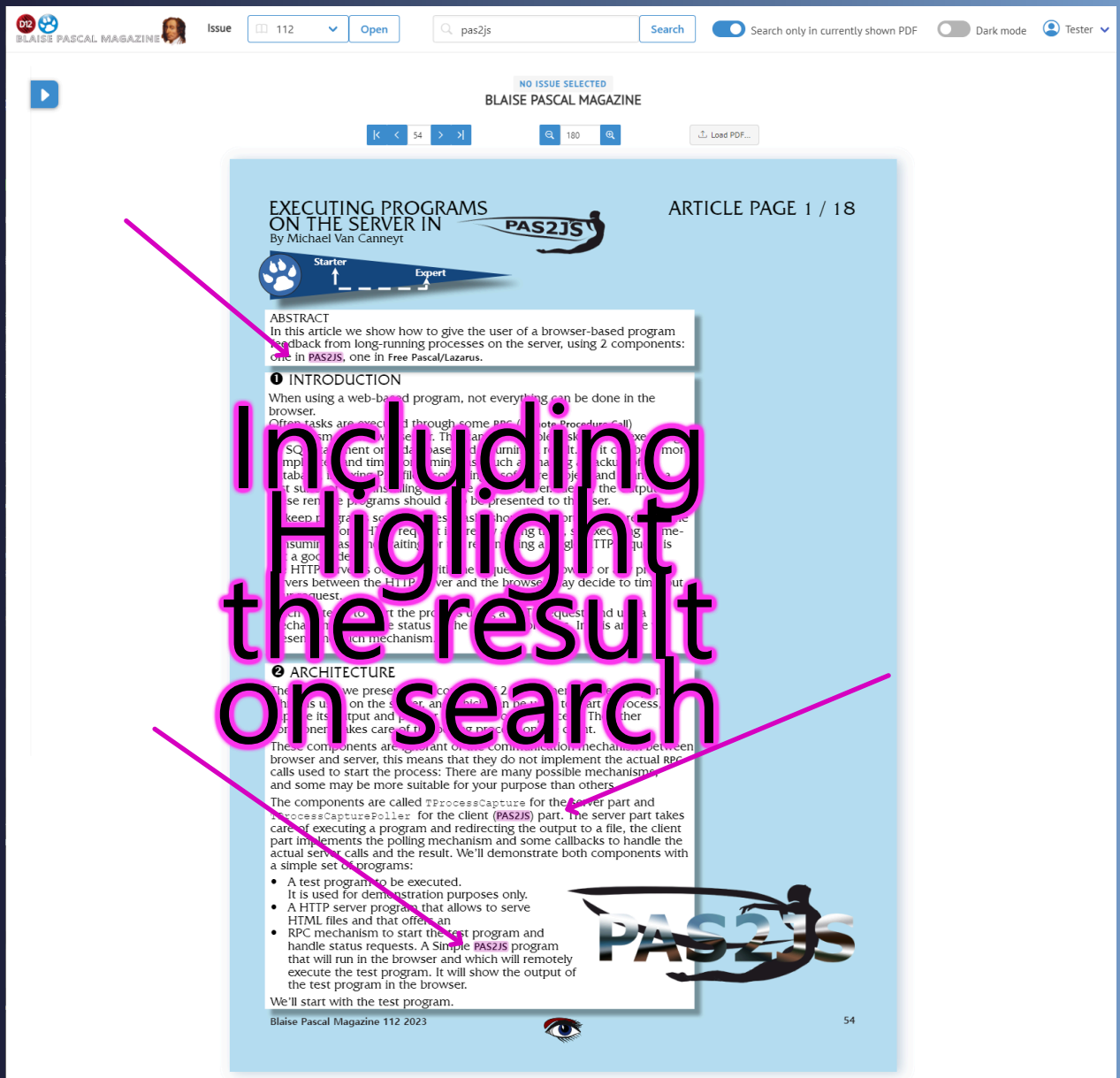
❼ WAS IST MIT PAS2JS?

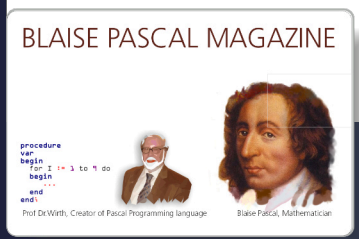
Für **PAS2JS** wird die gleiche namespacing Operation ausgeführt.
Die **Option -t** wurde bei den Transpiler Kommandozeilenoptionen zugefügt, so dass die Verfahrensweise bei beiden Compilern die gleiche ist. Um die Delphi-Kompatibilität zu verbessern, wurde zusätzlich zur bereits vorhandenen Unterstützung für mehrzeilige Strings in **PAS2JS** die Unterstützung für den **Delphi 12 Multiline** String hinzugefügt.

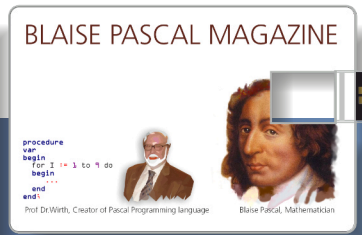
8 SCHLUSSBEMERKUNG

Es wurde viel Arbeit investiert, um **FPC Delphi-kompatibler** zu machen: Neue Sprachfunktionen, dotted Unit-Namen. Die Arbeit wurde größtenteils von einer Firma gesponsert, die ihr **Delphi**-Programm mit **Free Pascal** kompilieren möchte, ohne die Quellen des Programms ändern zu müssen. Als Ergebnis hat die **Delphi**-Kompatibilität von **Free Pascal** einen Schub bekommen und alle Benutzer von **Free Pascal** können nun eine verbesserte **Delphi** Kompatibilität genießen. Im Gegensatz zu den verschiedenen Übergängen in **Delphi**, die die Rückwärtskompatibilität brechen könnten, hat das **FPC-Team** entschieden, dass **FPC** mit sich selbst rückwärtskompatibel bleibt. Als Ergebnis können die Benutzer nun wählen, ob sie diese neueren Einheiten verwenden oder nicht. Sie haben immer die Wahl.

LIB-STICK ON USB CREDIT CARD BLAISE PASCAL MAGAZINE







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

UNTER LINUX EINE TEXTEBENE ZU PDF-DATEIEN HINZUFÜGEN

VON HELMUT ELSNER

EINLEITUNG

Für die meisten Drucker gibt es unter **LINUX** keine vom Hersteller gelieferte Tools, um beim Einscannen von Dokumenten durchsuchbare PDF-Dateien zu erhalten.
Die Texte werden immer nur als Bilder in den PDFs eingebettet.
Das macht es schwierig, später etwas Wichtiges wiederzufinden.
Natürlich gibt es auch unter **LINUX** entsprechende Tools, aber diese sind oft nur an der Kommandozeile zu bedienen. "**PDFsandwich**" ist eines davon und ein schönes dazu.
Für dieses Tool wollen wir eine **GUI (Graphical User Interface)** bauen, um mit ein paar Klicks einer oder mehreren PDF-Dateien eine Textebene hinzuzufügen.

VORBEREITUNGEN

Zuerst müssen die benötigten Programme installieren:

1. **ImageMagick** (ist bei den meisten **LINUX**-Distributionen bereits installiert)
2. **pdfsandwich**
3. **tesseract-ocr**

```
sudo apt install pdfsandwich tesseract-ocr-de -y
```

Dann noch ImageMagick das Recht geben, PDF-Dateien zu editieren:

```
sudo nano /etc/ImageMagick-6/policy.xml
```

In der Zeile fast ganz unten

```
<policy domain="coder" rights="none" pattern="PDF" />
```

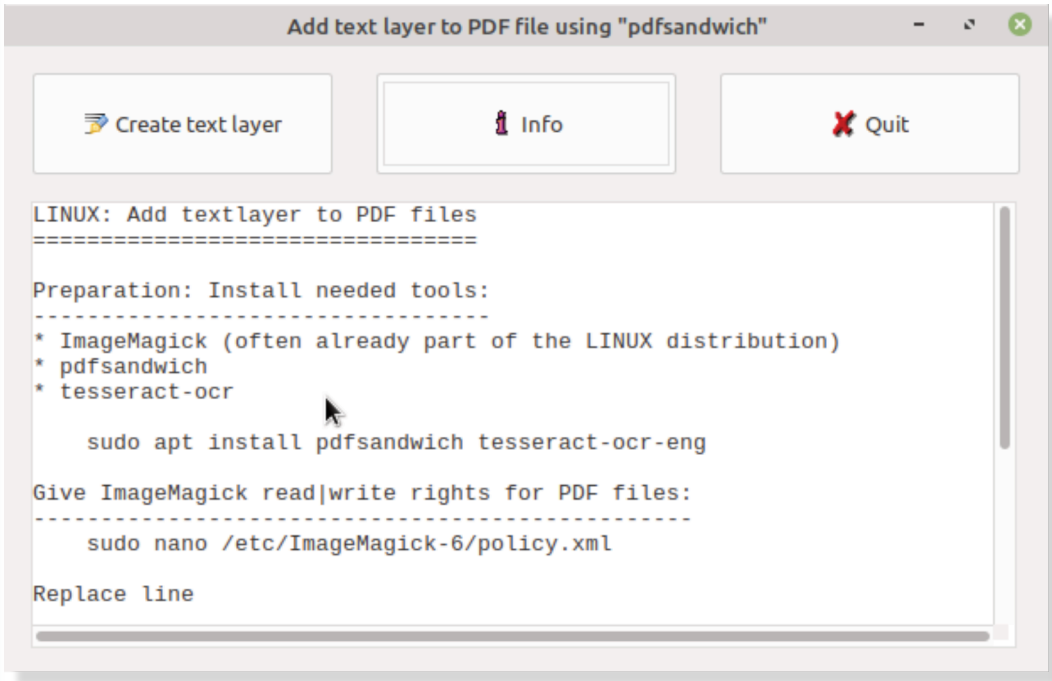
mit `<policy domain="coder" rights="read|write" pattern="PDF" />` überschreiben.

Speichern mit **Strg+O** und Beenden mit **Strg+X.**

Das Programm "**pdftext**" irgendwo im Homeverzeichnis ablegen (*z.B. unter /tools*).
Eventuell zum Starten aus den Startmenü eine `.desktop` Datei dazu anlegen.

PDFTXT USAGE

Die **GUI** ist sehr einfach und versteht sich eigentlich von selbst:



Mit "Textebene anlegen" wird man aufgefordert, PD… zum Bearbeiten auszuwählen. Man kann mehrere PDF-Dateien auswählen und alle mit einem R…sch bearbeiten lassen.
Es geht allerdings auch, PDF-Dateien zum Bearbeiten aus dem Dateimanager per Drag & Drop auf das Programmfenster zu ziehen.
Das Ergebnis der Bearbeitung wird im gleichen Verzeichnis wie die Ursprungsdatei in eine Datei mit dem Zusatz "_ocr" gespeichert.

AUFBAU DES PROGRAMMS

EINFACHSTE HILFEFUNKTION:
Eigentlich bräuchten wir nur einen Button zum Ausführen der Aktion, aber etwas mehr Information ist immer hilfreich. Ich habe mich entschlossen, als Hilfe nur eine Textdatei in einem TMemo anzuzeigen. Natürlich muss geprüft werden, ob sich die Textdatei mit den Installationshinweisen dort befindet, wo wir sie erwarten - im Programmverzeichnis (*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;
```

BEARBEITEN MEHRERER DATEIEN:
Zum Bearbeiten einer oder mehrerer PDF-Dateien gibt es den **Button "Textebene anlegen"**.
Um mehr als eine Datei auswählen zu können, müssen wir bei `TOpenDialog` die Option `ofAllowMultiSelect` setzen. Dann kann es losgehen, die Dateiliste abzuarbeiten.

```
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;
```

EIN TERMINALKOMMANDO MIT TPROCESS AUFRUFEN:
Wir rufen für jede Datei die eigentliche Bearbeitungsroutine auf. Wir prüfen anhand der **Fileextension**, ob die Dateien PDF-Dateien sind.
Die Prozedur erzeugt einen Prozess (`pdftxt: TProcess;`) um das gewünschte Terminalkommando, nämlich "`pdfsandwich`" auszuführen, die Ausgabe des Kommandos einzufangen und ins `TMemo` als Protokoll zu schreiben. Dazu wird dem Kommando noch eine Reihe von Parametern mitgegeben, darunter unser Dateiname der PDF-Datei, die bearbeitet werden soll (`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:
Um den Vorteil einer GUI richtig zu nutzen sollte auch **Drag & Drop** möglich sein. Dazu muss zuerst für das Formular die Option **AllowDropFiles** auf true gesetzt werden. dann können wir eine beliebige Menge PDF-Dateien zur Bearbeitung auf das Programmfenster ziehen.

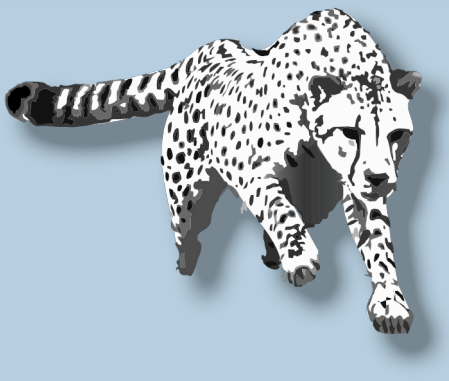
```
procedure TForm1.FormDropFiles(Sender: TObject; const FileNames:array of string);
var
  i: integer;
begin
  Screen.Cursor:=crHourGlass;
  btnClose.Enabled:=false;
  Memo1.Lines.Clear;
  Application.BringToFront; {Set focus to this program}
  try
    for i:=0 to high(FileNames) do
      PDFAddText(FileNames[i]);
  finally
    btnClose.Enabled:=true;
    Screen.Cursor:=crDefault;
  end;
end;
```

BENUTZERFREUNDLICHKEIT

Natürlich gibt auch eine kleine Hilfe in der Eigenschaft **Hint** zu jedem Bedienelement.
Um die Schriftgröße im `TMemo` mit **Mausrad + Strg** zu verändern, bauen wir noch eine Funktion ein, die auf die Events **`OnMouseWheelDown`** und **`OnMouseWheelup`** reagiert.

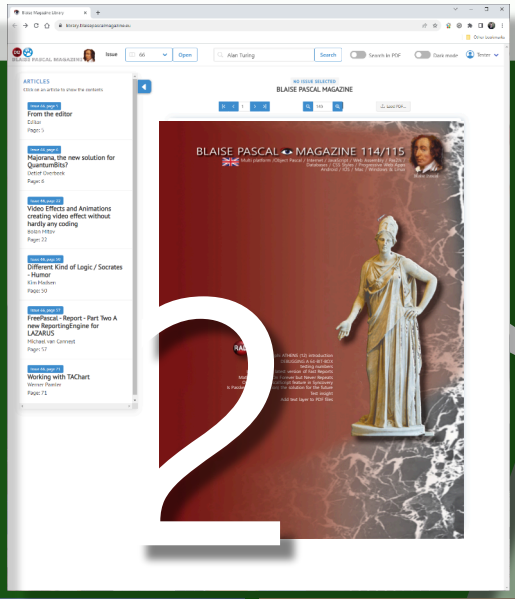
```
procedure TForm1.Memo1MouseWheelDown(Sender: TObject; Shift: TShiftState;
                                     MousePos: TPoint; var Handled: Boolean);
begin
  if ssCtrl in Shift then
    Memo1.Font.Size:=Memo1.Font.Size-1;
end;

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;
```

Das war dann schon alles.
Nach diesem Prinzip können wir für alle möglichen komplizierten Terminalkommandos, die wir uns nicht merken können, ein kleines Tool mit **GUI** bauen.
Das Projekt findet ihr hier: `https://github.com/h-elsner/PDFtext`



ADVERTISEMENT

1



2

3

LAZARUS HANDBOOK

4

POCKET
Edition
+shipment

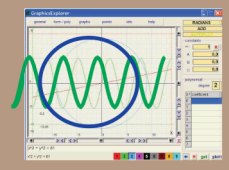
LAZARUS HANDBOOK PDF

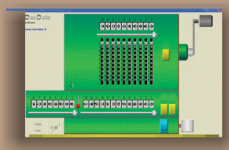
5

6

LEARN TO PROGRAM USING LAZARUS
HOWARD PAGE-CLARK

DAVID DIRKSE
including 50 example projects





7

BLAISE PASCAL MAGAZINE

COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

3

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

SPRING 2024
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 4 DEVELOPERS

Donate for Ukraine and get a free license at:
https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

COMPONENTS 4 DEVELOPERS

Donate for Ukraine and get a free license at:
https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

kbmMW Professional and Enterprise
NEW EDITION V. 5.23
kbmMemTable
NEW EDITION V. 7.99.00
Standard and Professional Edition

5.23.00 is a release with containing new stuff, refinements and bugfixes, **openSSL v3 support,** WebSocket support, further improvements to SmartBind, new high performance hashing algorithms, improved RemoteDesktop sample and much more.
This release requires the use of **kbmMemTable** v. 7.98.00 or newer.

- RAD Alexandria supported
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OS X client and server support
- Native high performance 100% developer defined application server
- Full support for centralised and distributed load balancing and fail-over
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multi thread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronounceable password generators.
- High performance LZ4 and J peg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, J SON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

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

- Easily supports large datasets with millions of records
- Easy data streaming support
- Optional to use native SQL engine
- Supports nested transactions and undo
- Native and fast build in M/D, aggregation/grouping range selection features
- Advanced indexing features for extreme performance
- New: full Web-socket support.
  The next release of kbmMW Enterprise Edition will include several new things and improvements.
  One of them is full Web-socket support.
- New I18N context sensitive internationalisation framework to make your applications multilingual.
- New ORM LINQ support for Delete and Update.
- Comments support in YAML.
- New StreamSec TLS v4 support (by StreamSec)
- Many other feature improvements and fixes.

Please visit http://www.components4developers.com
for more information about kbmMW

- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.

COMPONENTS 4 DEVELOPERS