CONTENT

ARTICLES

ADVERTISING

Niklaus Wirth

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

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

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

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

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

Subscriptions (2022 prices)	Internat. excl. VAT	Internat. incl. 9% VAT	Shipment	TOTAL
Printed Issue (8 per year) **±60 pages :**	€ 200	€ 218	€ 130	€ 348
Electronic Download Issue (8 per year) **±60 pages :**	€ 64,20	€ 70		

From your editor

Hello and a very good new year.

First of all an apology for bringing this issue late:
I changed the Lay-Out package CorelDraw for Affinity Publisher because this uses the GPU Engine and is therefore far more adequate for the future processes.
That means I had to learn how to handle this package and that costs much energy and time extra.
Since I am the only one that creates this Magazine I had to spend a lot of time to learn something completely new.

Especially since I work with Corel about 24 years. But I did it.
Never to old to learn.

If ever you need a designer or publishing Tool: Affinity is very cheap and has three integrated packages:
1-Publisher 2-Designer and 3-Photo and cost all in all no more then 120 €.

Now for something completely different:

The Lazarus team has created a totally new Project: FRESNEL.
Since ever I have tried to massage them to do something about the graphics and design of Lazarus.
This because I personally like that very much. But the most important would be the looks and feel of the package for the young generation.
Of course we want a dark mode etc. And we want to attract "the young ones", youth has the future, so they are very important to us.
Since Lazarus is capable of handling several OS's in one go we need to overcome the very distinct behaviour of Windows and Mac.
That's not easy.
The way to handle this is to redesign all standard components to custom designed components

which will get now two properties in the Object Inspector available settings: CSS and HTML (*even more: supporting code in sight and debugging*). Maybe even more is needed.

The best and most simple way to imagine is to say that the backward compatibility will NOT be broken.
You still can make use of the standard components, but the graphical extras are very much extended in these new components.
Even compatibility with Delphi is maintained. Let's take a button as an example.
The standard button does not have the capability for colour.
Here you can add that.
You even can make a round or oval button with colourised canvas or colourised text.
On page 65 of this issue the ins and outs are explained in the article introducing project FRESNEL.

Graphics will be as beautiful to handle because we use "Skia" for graphics.
Because of this, Michael and Mattias are building the part of FRESNEL:
the rendering program.

It also will be capable of creating your PAS2JS Forms, and therefore we will have a form inside Lazarus that will give you the ever so much wanted WYSIWYG:
"What you see is what you get".
It takes while to get all of this done. But we are steadily working on it.
To get this all done we need your support.
We will start a fundraising campaign and we hope you will support us for that.

In the next issue I hope to announce when that will happen and we will have an Event on 15th of April this year in the Netherlands
- please ask for your reservation -
where we will show, explain and let you try this new project.

We already have a proof of concept and in three months we will be prepared to show you more.

We have chosen the next event to be organised in April, because its warmer then and we need some time to get all of this done. We will keep the price for your reservation as low as possible: € 50. Where and how to, we will publish asap.

In this magazine are quite a number of articles: 14 in total. I hope it will satisfy your expectations.

There is a word to say about the future: I need help: anyone that might be interested in helping by writing articles or testing or review and correct text. Maybe even helping designing? Please let me know. (`editor@blaisepascal.eu`). Remember we are volunteers creating this all. And this work can all be done from your home address!

Maybe you are interested in "Cross compiling": if you want to compile for an other platform or any other sort of format, try the article about that: CROSS COMPILING WITH FPC UP DE LUXE. It will take away all the annoying hurdles you have to overcome and make your programming life much easier. Let me here from you.

Besides FRESNEL we are also working on the next level of a PDF program.

Michael van Canneyt has started to build an indexing engine that I will use for The MAGAZINE PDF -Library. That will make it very quick and easy to find the subject or matter you need in all of the magazines.

For the future: the plan is to make a complete PDF program that can do what Adobe can do, and then for all the OS's so there will be no more problems on using PDF on Linux or Mac – the quality needs to guaranteed.

We call the Project "Panther" and you will here from us regularly. With this project- for everybody free to use - we want to bring Lazarus/FPC to the end-user wherever he may be and still be OPEN SOURCE.

Please enjoy...

Detlef

*"My screen time was only up 10,000%.
I must've been distracted last week."*

LIBRARY 2019

LIBRARY 2020

- Search
in the whole text of one issue
- Search author
- Search title
- Search Issue Nr
- Your own PDF Viewer
use with any PDF file(s)
- Shows thumbnails
on and of
- Acces to all items
of Blaise Pascal Magazine
starting at issue 1 up to the latest
- Including all Code

LIBRARY 2021

-LIBRARY 2023
BLAISE PASCAL MAGAZINE
THE NEXT LEVEL

Penguin Data Science
maXbox Starter 101 P4D

Starter   **Expert**   D11

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper.

In this paper I want to show a machine learning lecture with Python4Delphi. The Palmer Archipelago (Antarctica) penguin dataset appears to be a drop in replacement for the same. It is a great intro dataset for data exploration & visualisation. But the penguins dataset has different number of samples for each species. It can be observed that unlike the Iris dataset, this data contains different number of entries for each specie.

Links and Sources
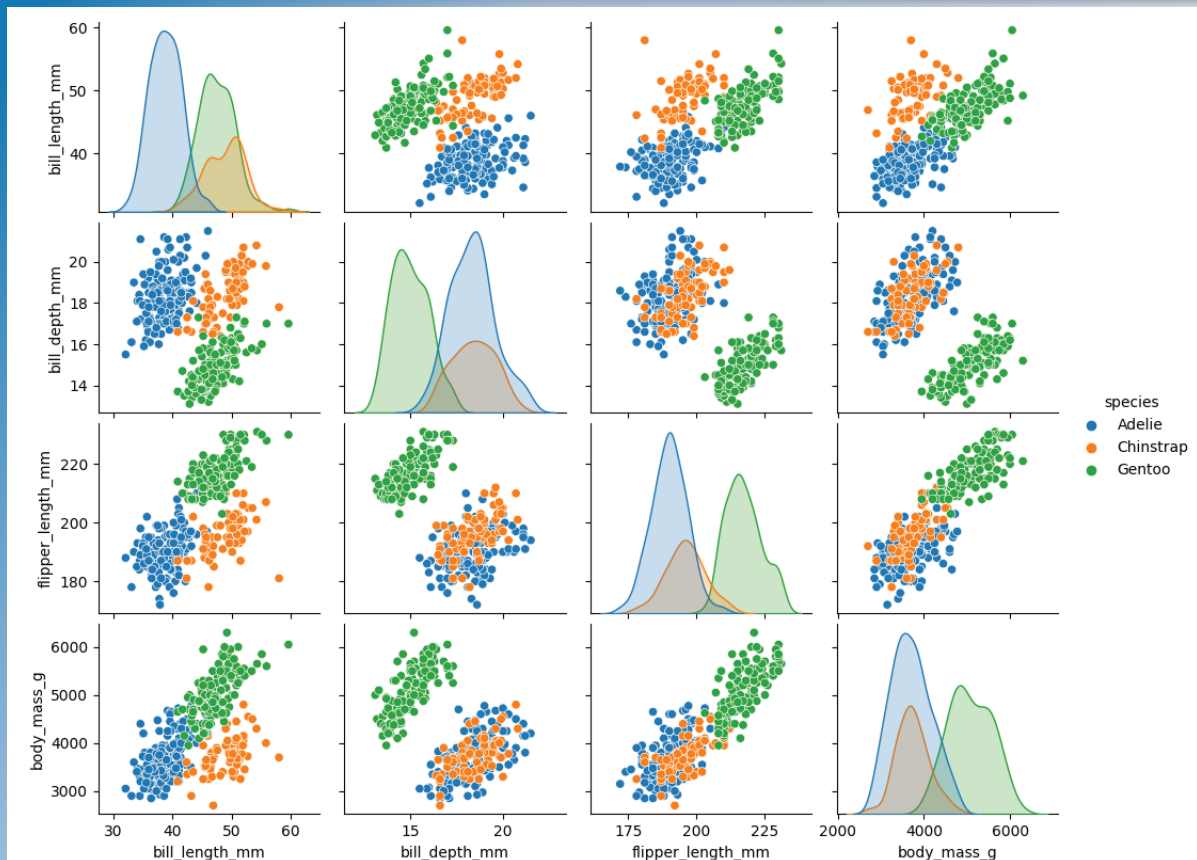
https://www.kaggle.com/code/parulpandey/penguin-dataset-the-new-iris

python4delphi/Tutorials/Webinar II at master · maxkleiner/python4delphi (github.com)

https://maxbox4.wordpress.com/

RAD Studio 11.4, maXbox4 and Python 3.8 with Seaborn

The template for the most P4D programs goes like this:

```
with TpythonEngine.Create(Nil) do begin
 pythonhome:= PYHOME;
 try
  loadDLL;
  Println('Decimal: '+
     EvalStr('__import__("decimal").Decimal(0.1)'));
 except
  raiseError;
 finally
  free;
 end;
end;
```

The minimal configuration depends on your Python-installation and the UseLastKnownVersion property in TDynamicDll but once known it goes also with raiseError to catch the Python exceptions In your code. For the const of PYHOME on my machine it depends on the version:

```
Const PYHOME = 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';

      PYHOME = 'C:\Users\max4\AppData\Local\Programs\Python\Python37-32\';
```

From code to a short introduction to the shape of a penguin s head with his culmen.

The upper margin of the beak or bill is referred to as the culmen and the measurement is taken using calipers with one jaw at the tip of the upper mandible and the other at base of the skull or the first feathers depending on the standard chosen.

First we get the data with seaborn:

import seaborn as sns

df = sns.load_dataset("penguins")

Or alternate sources

df = pd.read_csv('../input/palmer-archipelago-antarctica-penguin-data/penguins_size.csv')

```
p2data = "https://raw.githubusercontent.com/allisonhorst/palmerpenguins/
          master/data-raw/penguins_raw.csv
p2data =  https://raw.githubusercontent.com/rianrajagede/penguin-python/
          master/Datasets/penguins.csv
```

```
Python Chats to Svg maXbox4

[matplotlib] [seaborn]

    from delphi_module import svg_image
    from io import StringIO
    import seaborn as sns
    import matplotlib.pyplot as plt

    df = sns.load_dataset("penguins")
    sns.pairplot(df, hue="species")

    figfile = StringIO()
    plt.savefig(figfile, format='svg')
    figdata_svg = figfile.getvalue()
    svg_image.SvgText = figdata_svg

    #plt.show()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
count	344	344	342.000000	342.000000	342.000000	342.000000	334
unique	3	3	NaN	NaN	NaN	NaN	3
top	Adelie	Biscoe	NaN	NaN	NaN	NaN	MALE
freq	152	168	NaN	NaN	NaN	NaN	168
mean	NaN	NaN	43.921930	17.151170	200.915205	4201.754386	NaN
std	NaN	NaN	5.459584	1.974793	14.061714	801.954536	NaN
min	NaN	NaN	32.100000	13.100000	172.000000	2700.000000	NaN
25%	NaN	NaN	39.225000	15.600000	190.000000	3550.000000	NaN
50%	NaN	NaN	44.450000	17.300000	197.000000	4050.000000	NaN
75%	NaN	NaN	48.500000	18.700000	213.000000	4750.000000	NaN
max	NaN	NaN	59.600000	21.500000	231.000000	6300.000000	NaN
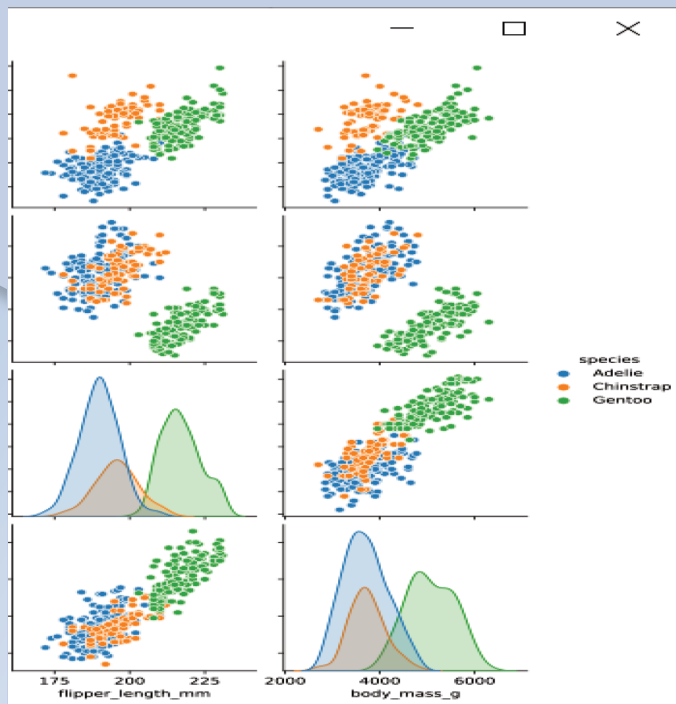
```
 df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   species          344 non-null    object
 1   island           344 non-null    object
 2   culmen_length_mm 342 non-null    float64
 3   culmen_depth_mm  342 non-null    float64
 4   flipper_length_mm 342 non-null   float64
 5   body_mass_g      342 non-null    float64
 6   sex              334 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
memory shape core cube #7
```

The dataset consists of 7 columns.

- **species: penguin species (Chinstrap, Adélie, or Gentoo)**
- **culmen_length_mm**
- **culmen depth (mm)**
- **flipper_length_mm: flipper length (mm)**
- **body_mass_g: body mass (g)**
- **island: island name (Dream, Torgersen, or Biscoe) in the Palmer Archipelago (Antarctica)**
- **sex: penguin sex**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. On one short code snippets we get the scatter-plot above with the following lines:

```python
import numpy as np

import pandas as pd
from io import StringIO
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset("penguins")
sns.pairplot(df, hue="species")
figfile = StringIO()
plt.savefig(figfile, format="svg")
plt.show()
print(df.info())
```

A scatter plot (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables.
The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.
A common modification of the basic scatter plot is the addition of a third variable. Values of the third variable can be encoded by modifying how the points are plotted.
For a third variable that indicates categorical values (like geographical region, suburbans or gender), the most common encoding is through point color.
Seaborn lets you create relational plots using the relplot() function. The function technically lets you create more than one scatter plot.

1.2   COVARIANCE

"Covariance"
indicates the direction of the linear
relationship between variables. "Correlation" on the other
hand measures both the strength and direction of the linear relationship
between two variables.

Source: **https://tinyurl.com/yd2pezss**

Second correlations describe the mutual relationship between two variables. They provide
the possibility to measure the relation between any kind of data - continuous and
continuous, categorical and categorical, and also continuous and categorical at the same
time.
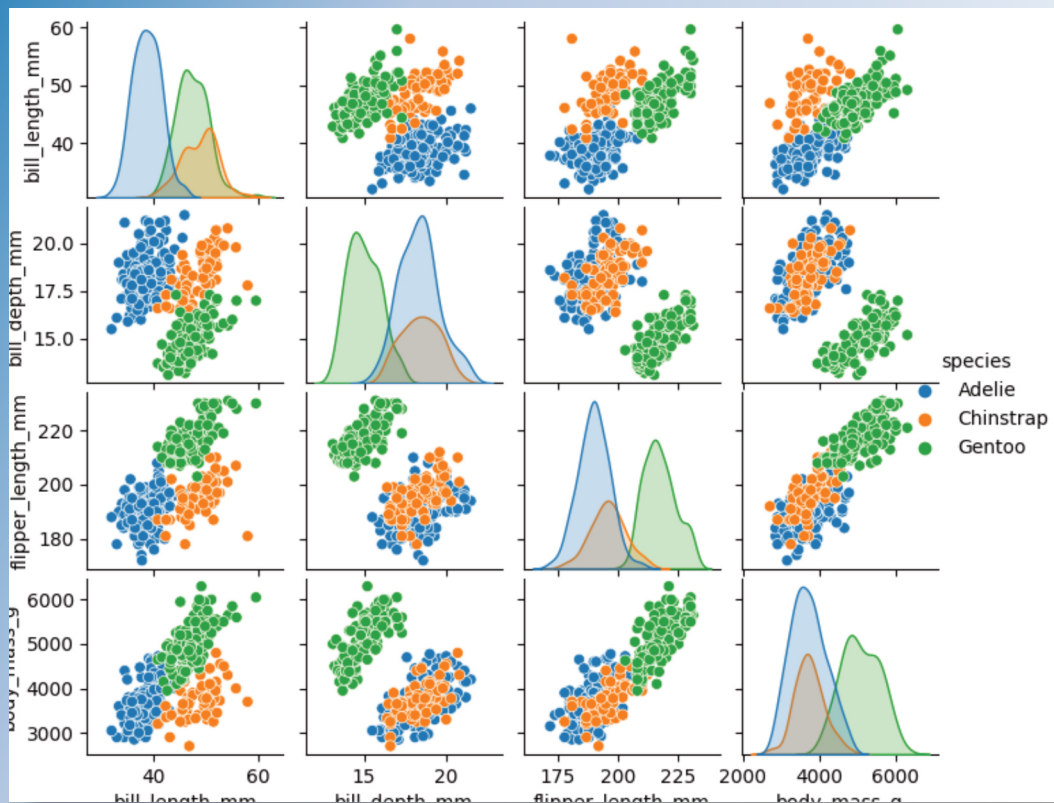
The scatter plot which shows us the correlation with respect to other features. This method
helps just to figure out the important features which account the most for the classification in
our model.

Official releases of seaborn can be installed from PyPI:

pip install seaborn

The basic invocation of pip will install seaborn and, if necessary, its mandatory dependencies. It is possible to include optional dependencies that give access to a few advanced features:
pip install seaborn[stats].
Numpy arrays are a good substitute for python lists. They are better than python lists. They provide faster speed and take less memory space. Let's begin with its definition for those unaware of numpy arrays. They are multi-dimensional matrices or lists of fixed size with similar elements.

Pandas is a popular Python library used to manipulate tabular data. It provides a versatile dataframe object that can read data from many popular formats, such as Excel, SQL, CSV and more.
The Pandas style API provides you with many different tools that makes working with styling tabular data much easier.

1.3 SCATTER PLOT

A scatter plot can also be a pair plot using values for two different numeric variables. The position of each dot on the different horizontal and vertical axis indicates values for an individual data point. Scatter plots are mainly used to observe relationships between variables.
A pairplot on the other side plots a pairwise relationships in a dataset. The pairplot function creates a grid of Axes such that each variable in data will by shared in the y-axis across a single row and in the x-axis across a single column. That creates plots as shown before.

```
Simple P4D Demo maXbox4                                    —   □   ×
                              Python Source code
from delphi_module import np_array

print("type(np_array) = ", type(np_array))
print("len(np_array) = ", len(np_array))
print("np_array = ", np_array)

res_array = np_array.copy()
for i in range(len(np_array)):
    res_array[i] *= np_array[i]
print("res_array = ", res_array)
```

```
                              Python Output
type(np_array) =  <class 'numpy.ndarray'>        1
len(np_array) =  10                              4
np_array =  [ 1  2  3  4  5  6  7  8  9 10]      9
res_array =  [  1   4   9  16  25  36  49  64  81 100]   16
type(np_array) =  <class 'numpy.ndarray'>        25
len(np_array) =  10                              36
np_array =  [ 1  2  3  4  5  6  7  8  9 10]      49
res_array =  [  1   4   9  16  25  36  49  64  81 100]   64
                                                 81
                                                 100
                                                 1
                                                 4
                                                 9
                                                 16

   Run        Python Version
```

Volume and percent change

Python Chats to Svg maXbox4

matplotlib | seaborn

```python
from delphi_module import svg_image
from io import StringIO
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook

# Load a numpy record array from yahoo csv data with fields
# volume, adj_close from the mpl-data/example directory. The
# stores the data as an np.datetime64 with a day unit ('D')
price_data = (cbook.get_sample_data('goog.npz', np_load=True
                .view(np.recarray))
price_data = price_data[-250:]  # get the most recent 250 tr

delta1 = np.diff(price_data.adj_close) / price_data.adj_clos

# Marker size in units of points^2
volume = (15 * price_data.volume[:-2] / price_data.volume[0]
close = 0.003 * price_data.close[:-2] / 0.003 * price_data.c

fig, ax = plt.subplots()
ax.scatter(delta1[:-1], delta1[1:], c=close, s=volume, alpha

ax.set_xlabel(r'$\Delta_i$', fontsize=15)
ax.set_ylabel(r'$\Delta_{i+1}$', fontsize=15)
ax.set_title('Volume and percent change')

ax.grid(True)
fig.tight_layout()
```

2   TECHNICAL DESCRIPTION

maxkleiner (Max Kleiner) (github.com)

A Python program terminates as soon as it encounters an error.
In Python an error can be a syntax error or an exception.
In this article, you will see what an exception is and how it differs from a syntax error.
After that, you will learn about raising exceptions and making assertions.
Then, you'll finish with a demonstration of the try and except block.
https://realpython.com/python-exceptions/

Error   ✕

PyScripter could not find a usable Python installation at the specified path.
Note that the 64-bit version of PyScripter (x64) works only on 64-bit Windows **and**
with 64-bit versions of Python. The 32-bit version of PyScripter works on both
32-bit and 64-bit Windows with 32-bit versions of Python.

OK
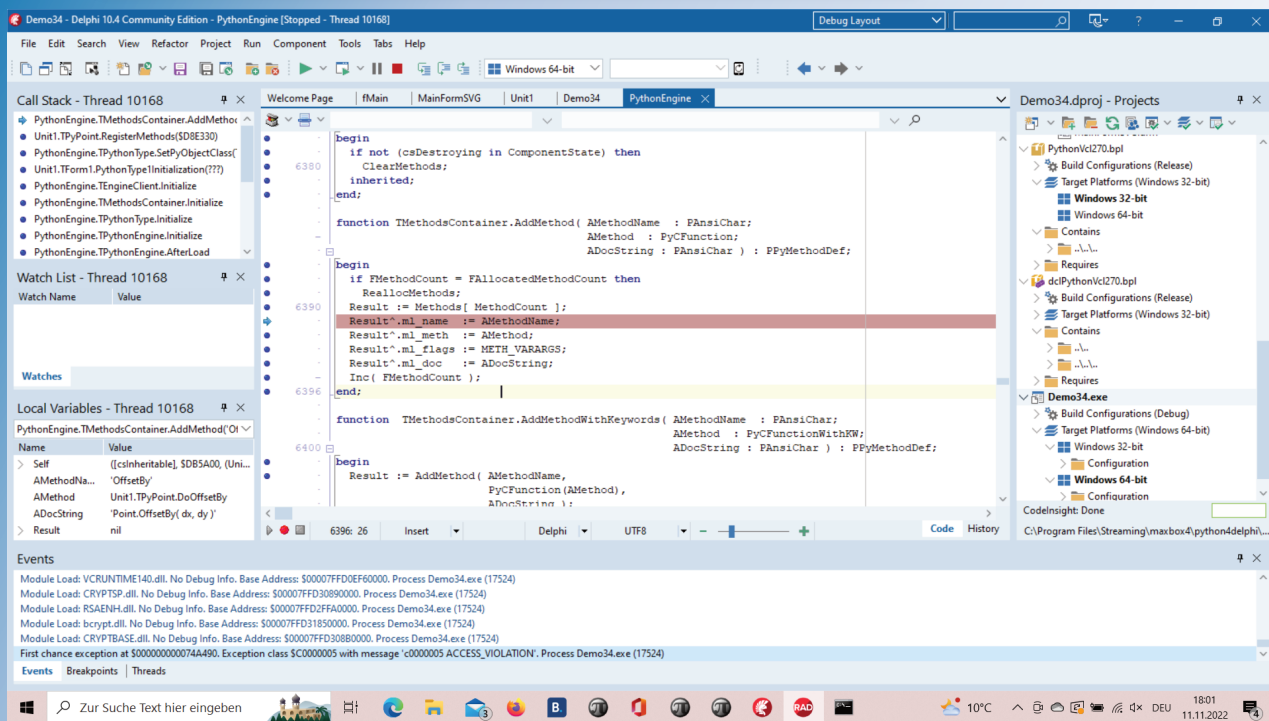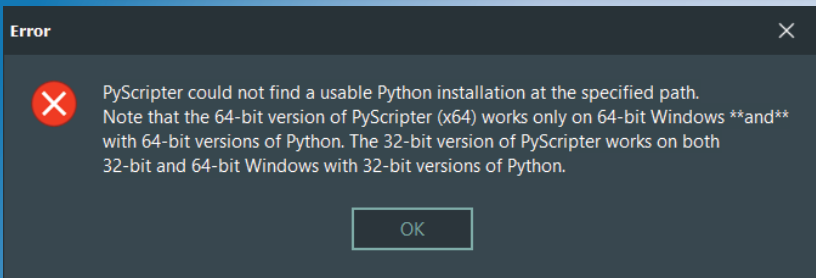
PyScripter 4.0.0 is now available at Sourceforge. PyScripter is fully programmed in Delphi. This is a major new release with an updated User Interface and many significant enhancements under the hood, that will increase the stability and improve the user experience.

Scalable Vector **Graphics** (**SVG**) is an XML-based vector image format for defining two-dimensional graphics, having support for interactivity and animation. The **SVG** specification is an open standard developed by the World Wide Web Consortium since 1999.
https://en.wikipedia.org/wiki/SVG

yScripter - C:\Program Files\Streaming\IBZ2021\Module2_3\992_oma_objectdetector21pyscripter.py

File   Edit   Search   View   Project   Run   Tools   Help

Code Explorer

- 992_oma_objectdetector21pyscrip...
  - + Imports
  - + Globals
  - ObjectDetection
  - GraphViz
  - StringIO
  - train_test_split
  - LogisticRegression
  - classification_report
  - confusion_matrix
  - roc_curve
  - roc_auc_score

```python
from sklearn import preprocessing
X = preprocessing.scale(X)

#splitting the data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_s

140  # model fitting and prediction

from sklearn.linear_model import LogisticRegression

#new_df = pd.DataFrame(StandardScaler().fit_transform(df), col
145
#X_train = X_train.isnan()
#import numpy as np
#X_train=X_train[:, ~np.isnan(X_train).any(axis=0)]
#y_train=y_train[~y_train.isnull()]
150  #pd.options.mode.use_inf_as_na = True

"""
X_train[:, ~np.isnan(X_train).any(axis=0),]
X_train[:, ~np.isfinite(X_train).any(axis=0)]
y_train[~np.isnan(y_train)]
y_train[~np.isfinite(y_train)]
"""
print(X_train.shape)
print(y_train.shape)
160
model = LogisticRegression().fit(X_train, y_train)
pred = model.predict(X_test)
print(pred)

# checking performance of model

from sklearn.metrics import classification_report, confusion_m

print('CONFUSION MATRIX')
170  print(confusion_matrix(y_test, pred))
print('CLASSIFICATION REPORT\n')
print(classification_report(y_test, pred))
```

Fil...rer   Pr...er   Co...er | 992_oma_objectdetector21pyscripter.py ×   validation.py ×   multiclass.py

Sick of using IDLE? Want to code in a fancy new IDE?
These steps will help you in downloading Python 2.7.1 or Python 3.8 and PyScripter
so you can start learning and have fun with Python.

https://www.instructables.com/Python-With-PyScripter/

```
size=0.2, random_state=13)
```

```
umns=df.columns, index=df.index)
```

Python Interpreter

```
CONFUSION MATRIX
[[32  1  0]
 [ 1 11  0]
 [ 0  0 22]]
CLASSIFICATION REPORT

              precision    recall  f1-score   support

      Adelie       0.97      0.97      0.97        33
   Chinstrap       0.92      0.92      0.92        12
      Gentoo       1.00      1.00      1.00        22

    accuracy                          0.97        67
   macro avg       0.96      0.96      0.96        67
weighted avg       0.97      0.97      0.97        67

>>>
```

```
atrix, roc_curve, roc_auc_score
```

series.py ×   generic.py ×   blocks.py ×   base.py ×   indexing.py ×

Python 3.8 (64-bit)   Remote

RizSVG - Delphi 10.4 Community Edition - MainFormSVG

Edit   Search   View   Refactor   Project   Run   Component   Tools   Tabs   Help

Windows 32-bit

Structure

- Classes
  - TForm1(TForm)
    - Published
      - SynPythonSyn1: TSynPythonSyn
      - SynEditPythonBehaviour1: TSynEditI
      - PythonEngine1: TPythonEngine
      - PythonGUIInputOutput1: TPythonGl
      - PythonModule: TPythonModule
      - PyDelphiWrapper: TPyDelphiWrappe
      - SVGIconImage1: TSVGIconImag
      - SVGIconImage: TSVGIconImag
      - PageControl1: TPageControl
      - TabSheet1: TTabSheet

Welcome Page | MainFormSVG | fMain | MainForm | MainFormSV

```pascal
unit MainFormSVG;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtil
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, WrapDelph
  PythonGUIInputOutput, SynEditPythonBehaviour,
  SynEditCodeFolding, SynHighlighterPython, Vcl.
```

Object Inspector

Form1 TForm1

Properties | Events

BiDiMode	bdLeftToRight
BorderIcons	[biSystemMenu,biMini
BorderStyle	bsSizeable
BorderWidth	0
Caption	Python Chats to Svg n
ClientHeight	613

Quick Edit... Quick Edit Icon... Bind Visually..

All shown

Messages

Compiling PyVizSVG.dproj (Debug, Win3
dcc32 command line for "PyVizSVG.dpr"
Success
Elapsed time: 00:00:00 8

Build   Output

Python Chats to Svg maXbox4

matplotlib   seaborn

```python
from delphi_module import svg_image
from io import StringIO
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook

# Load a numpy record array from yahoo csv data with fields
# volume, adj_close from the mpl-data/example directory. The
# stores the date as an np.datetime64 with a day unit ('D')
price_data = (cbook.get_sample_data('goog.npz', np_load=True
                .view(np.recarray))
price_data = price_data[-250:]  # get the most recent 250 tr

delta1 = np.diff(price_data.adj_close) / price_data.adj_clos

# Marker size in units of points^2
volume = (15 * price_data.volume[:-2] / price_data.volume[0]
close = 0.003 * price_data.close[:-2] / 0.003 * price_data.c

fig, ax = plt.subplots()
```

Run

3 DEVELOP OVERVIEW

We value the quality of the material each student is learning. Our Instructor led classes help provide hands-on learning, one-on-one mentoring with experienced developers and peer-to-peer learning. To allow the most flexibility for students, we have Day and Night programs to choose from. *No prior coding experience required.

```pascal
procedure pyBank_VCL4Python4Delphi;
var eg: TPythonEngine; sw: TStopWatch;
begin
 eg:= TPythonEngine.Create(Nil);
  try
   eg.pythonhome:= PYHOME;
   eg.loadDLL;
   println('test import '+GetPythonEngine.EvalStr('__import__("decimal").Decimal(0.1)'));
   writeln('')
   //println(eg.EvalStr('__import__("faker").Faker()'));
   sw:= TStopWatch.Create();
   sw.Start;
   eg.execStr('import importlib.machinery, importlib.util');
   eg.execStr('from decimal import Decimal, getcontext');
   importlib.machinery.ExtensionFileLoader("DelphiVCL",'+VCLHOME+')')

   eg.execStr(ANALYTICSSVG);
      //println(eg.evalStr("#39'MSE:'#39',mean_squared_error(test_bayesian_pred,y_test_
confirmed)'));
   sw.Stop;
   //sw.ElapsedMilliseconds;
   writeln('Stop Analytics Tester1: '+sw.getValueStr)
  except
   eg.raiseError;
   writeln(ExceptionToString(ExceptionType,ExceptionParam));
  finally
   eg.Free;
   sw.Free;
   sw:= Nil;
  end;
 end;
```

Conclusion:

For the moment you can use four engines to render SVG (Delphi Image32, Delphi TSVG, SKIA4Delphi and Direct2D wrapper) and four components (packages) to simplify use of SVG images (resize, fixedcolor, grayscale and so on) and with Python4Delphi is a fifth engine available:

```pascal
procedure TForm1.FormCreate(Sender: TObject);
begin
 var Py:= PyDelphiWrapper.Wrap(SVGIconImage, soReference);
 PythonModule.SetVar('svg_image', Py);
 GetPythonEngine.Py_DECREF(Py);
end;
```

Tested on
  - Delphi 10.4 Community Edition Update 2
  - Lazarus V 2.2.4 for Windows 64 bit
  - maXbox4 Version 4.7.6.20 x32
  - PyScripter 4.1.1.0 x64
  - SVGImage32Package270.bpl V 3.9.4

https://github.com/EtheaDev/SVGIconImageList

LibraryForm

Show article | Go to issue Nr | Issue | Search | ?

ID	IssueNr	Author	Article	PDF	PageNr
884	103_104	Michael Philippenko	FastReport: New version / Using the table Object in Table Reports		19
885	103_104	David Dirkse	The Ceva project		25
886	103_104	David Dirkse	The distance of diagonals in a cube		30
887	103_104	Detlef Overbeek	Creating Electron apps with Pas2JS		34
888	103_104	Detlef Overbeek	Quantum Cryptography		48
889	103_104	Detlef Overbeek	SKIA voor Delphi the next level		64
890	103_104	Detlef Overbeek	Installing FPC Trunck versions		82
891	103_104	Michael van Canneyt	Tested: The new version of Database Workbench		89
892	103_104	Michael van Canneyt	Pas2JS: Samples for creating your new website		111
893	103_104	Michael van Canneyt	Anonymous functions Explained for Delphi and Lazarus		133

☑ Show Thumbnails   « » ⊕ ⊖ ↺ ↻ Page Jump to page   🖶   ?

Page 131
Page 132
Page 133
Page 134
Page 135
Page 136
Page 137
Page 138
Page 139
Page 140
Page 141
Page 142
Page 143
Page 144
Page 145
Page 146

BLAISE PASCAL ◉ MAGAZINE 107/108

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

Blaise Pascal

Penguin Data Science: 4 engines to render Vector graphic
FRESNEL *the graphical project for Lazarus, creating colours everywhere*
PANTHER: the super fast, in Lazarus created PDF indexer for all Os's explaine
The Pendulum, time acts like a wav
Using styles with a VCL program
Lazarus programme using BufDataset
Cross compiling with FPC Up De Lux
Accelerating the Delphi Compile
multi-node quantum network - Quantum Internet New
What's open about smart chatbot maker OPENAI
Discussion about OPEN GP
How to create a BufDataset Master Deta
Using Styles in Delph
Installing FastReport packages in Lazarus for Linux/Window
Zeos database components for Lazarus

BLAISE PASCAL MAGAZINE

procedure
var
begin
for i := 1 to n do
begin
end
end;

Prof Dr.Wirth, Creator of Pascal Programming language   Blaise Pascal, Mathematician

BLAISE PASCAL MAGAZINE

procedure
var
begin
for i := 1 to n do
begin
end
end;

Prof Dr.Wirth, Creator of Pascal Programming language   Blaise Pascal, Mathematician

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

Prof Dr.Wirth, Creator of Pascal Programming language

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

PDF Viewer 2023
on credit card USB Stick including
Blaise Pascal Library

Starter   **Expert**

INTRODUCTION
This small Delphi project simulates a pendulum. Main purpose is to measure the cycle time, which is done in a numeric way. An analytical approach is difficult as the cycle time depends on the pendulum amplitude.
Below is a picture of the project in Delphi 7:
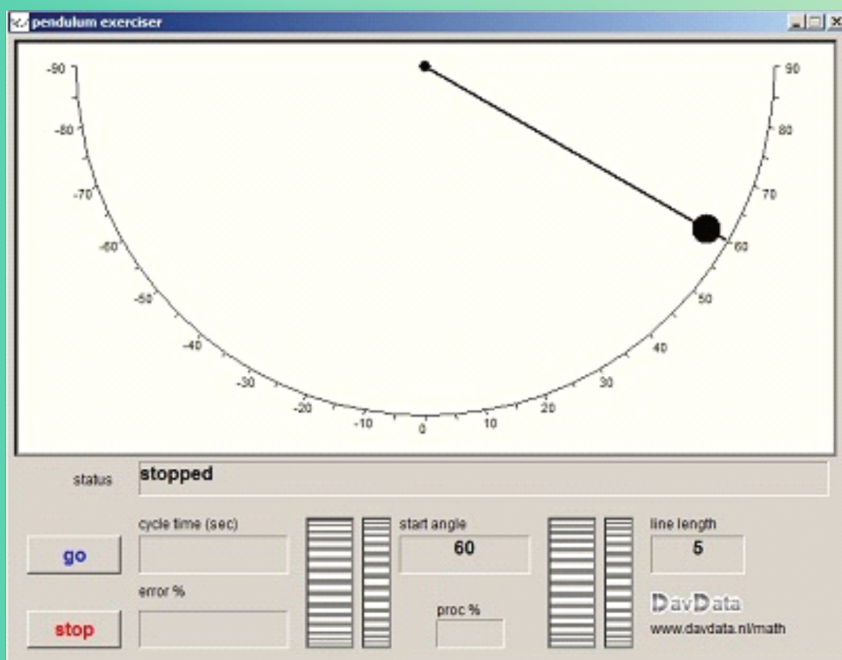
The vertical rotation buttons allow for the selection of the angle at with the mass is released initially and the length of the cable.

OPERATION
- select angle in degrees (move mouse over rotation buttons)
- select line length (..)
- press **go** button to release mass

Press **stop** to end the simulation process.

Displays show:
- cycle time
- accuracy of the simulated movement
- percentage of time the processor needs for the simulation and display

BASIC KINEMATICS
The unity of length is the meter (m).
The unity of time is the second (sec).

For constant motion:
distance = speed * time.
$s = vt$ {s : distance (m), v : velocity (m/sec), t : time (sec)}

Acceleration is velocity change per second.
In case of constant acceleration:
$v = at$ {v: velocity (m/sec), a: acceleration (m/sec^2), t: time (sec)}
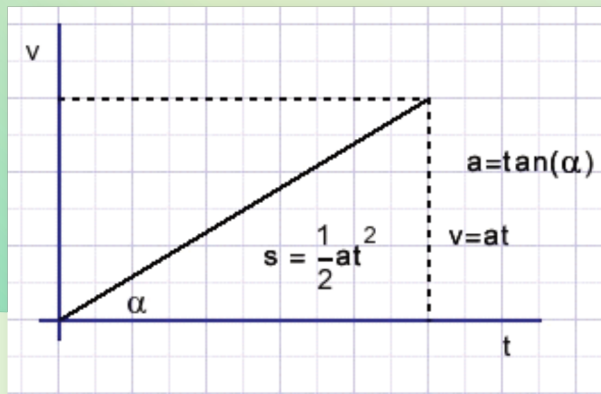
Distance s is the area of the triangle:
```
s = 0.5at2
```
For constant acceleration:
```
If vt is the velocity of an object at time t:
vt = v0 + at
If st is the distance in time t
st = v0t + 0.5at2
```

The unity of

mass is the kilogram, which is independent of gravity.

The unity of force is the NEWTON
One Newton (N) is the force that accelerates a mass of 1 (kg) by 1m/sec for each second.
$F = ma$ {F: force (N), m: mass (kg), a: acceleration (m/sec2)}
On earth, a = 9.8 and is called g.

Mechanical energy is the product of force and distance:
The unity of energy is the Joule (J).
$E = Fs$
         {E: energy (J), F force (N) s: distance (m).

To lift a mass m over h meters upward against gravity takes energy
$E_p = mgh$ {Ep: potential energy (J)}

Energy is never lost but may be converted to another form such as chemical energy in batteries or heat.

A mass m is accelerated a (m/sec^2) during t seconds.
The distance s = 0.5at2.
Energy E = Fs = F*0.5*at2
Since F=ma:
E = 0.5ma2t2
since v=at:
Ek = 0.5mv2 (Ek : kinetic energy (J)}

THE SIMULATION PROCESS

In the pendulum no energy is lost (theoretically) but potential and kinetic energy alternate.

Air friction is not taken into account but may be added easily.
In the bottom position all energy is kinetic, at the most upward position all energy is potential.
In between these positions there is a mix.
Since the total energy is constant we are able to calculate the exact speed at all heights of the mass.
The mass is accelerated by the force perpendicular to the line.

Forces in the rope take no energy as long as the rope is rigid.

$$F = mg.sin(\alpha)$$
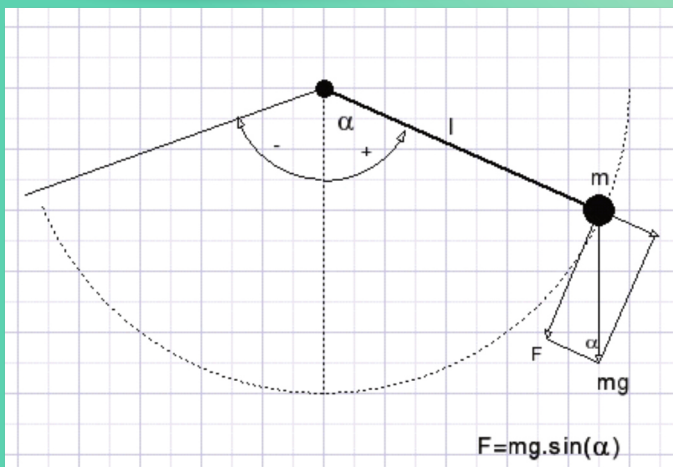
We notice that force F depends on angle α
When measuring α in radiants:
$s := \alpha l$ {s: distance on circle perimeter, l: length of line}
The simulation process is amazingly simple.
Time is subdivided in microseconds.
During this small interval, force F and velocity v are supposed to be constant.
F is known from angle α and causes acceleration $a = F/m = g.sin(\alpha)$;
So the new velocity v becomes $v + a*10\text{-}6$.
Distance s in this period is $v.10\text{-}6$
and since $s = \alpha l$
the new angle α becomes $\alpha + vt/l$.

After 10000 of such steps, a new picture is presented showing the position of the pendulum.
So, 100 pictures per second are displayed.

SPEED AND HEIGHT RELATION
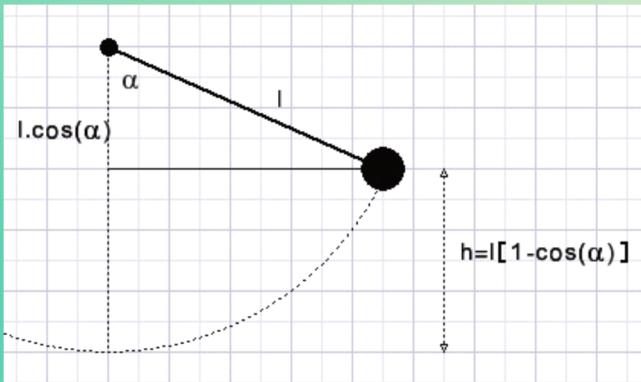
Released at an angle of α at h meter above the lowest point the potential energy is

```
mgh = mgl[1-cos(α)].
```
At an angle of φ the potential energy is (reduced to) mgl[1-cos(φ)].
This energy loss is now kinetic energy so:

$$\frac{1}{2}mv^2 = mgl[\cos(\theta) - \cos(\alpha)]$$

$$v = \sqrt{2gl[\cos(\theta) - \cos(\alpha)]}$$

AN INTEGRAL FOR THE CYCLE TIME

traveling distance Δs at speed v during a very small time Δt over the circle perimeter:

$$\Delta s = v.\Delta t \qquad s = \theta l$$

$$\Delta t = \frac{\Delta s}{v} = \frac{l.\Delta\theta}{v}$$

$$t = l.\int \frac{d\theta}{v} = l.\int \frac{d\theta}{\sqrt{2gl[\cos(\theta)- \cos(\alpha)]}} = \sqrt{\frac{l}{2g}}\int \frac{d\theta}{\sqrt{\cos(\theta) - \cos(\alpha)}}$$

integration over α to - α yields half the cycle time so total cycle T is

$$T = \sqrt{\frac{2l}{g}}\int \frac{d\theta}{\sqrt{\cos(\theta) - \cos(\alpha)}}$$

This integral is difficult to solve.
(make power series and integrate term by term)

```
Special case: very small α
Writing cos(x) = 1 - 2sin2(x/2) :
cos(φ) - cos(α) = 2[sin2(α/2) - sin2(φ/2)]
which changes the cycle time integral to
```

$$T = \sqrt{\frac{l}{g}} \int_{-\alpha}^{\alpha} \frac{d\theta}{\sqrt{\sin^2\left(\frac{\alpha}{2}\right) - \sin^2\left(\frac{\theta}{2}\right)}}$$

Now substituting

$$u = \frac{\sin\left(\frac{\theta}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} \Rightarrow \frac{du}{d\theta} = \frac{\cos\left(\frac{\theta}{2}\right)}{2\sin\left(\frac{\alpha}{2}\right)} \Rightarrow d\theta = \frac{2\sin\left(\frac{\alpha}{2}\right)}{\cos\left(\frac{\theta}{2}\right)} \cdot du$$

we get:

$$= 2\sqrt{\frac{l}{g}} \int_{-1}^{1} \frac{du}{\sqrt{1-\sin^2\left(\frac{\theta}{2}\right)}\sqrt{1-u^2}} = 2\sqrt{\frac{l}{g}} \int_{-1}^{1} \frac{du}{\sqrt{\left(1-u^2\sin^2\left(\frac{\alpha}{2}\right)\right)\left(1-u^2\right)}}$$

When α is small this integral may be reduced to

$$T = 2\sqrt{\frac{l}{g}} \int_{-1}^{1} \frac{du}{\sqrt{1-u^2}} = 2\sqrt{\frac{l}{g}} \left[arcsin(u)\right]_{-1}^{1} = 2\pi\sqrt{\frac{l}{g}}$$

THE PROGRAM

The procedure is spelt out in a bitmap called bm which is drawn on paintbox1 on the form to

```pascal
const deg2rad : double = pi/180;
   rad2deg : double = 180/pi;
   pi2 : double = pi/2;
var bm : Tbitmap;
   Apos : double;          //angle in radius
   line : double;          //line length selection (m)
   ActiveLine : double; //actual line length in use
   Astart : word;          //start angle (degrees)
   stopFlag : boolean = true; //set by stop button

function CalcMaxV : double; //calculate maximum velocity
begin
 result := sqrt(2*9.8*activeline*(1 - cos(Apos)));
end;
```

So far the theory.

THE SIMULATION PROCEDURE:

Note: the error report shows the difference between the calculated and simulated velocity at the vertical line position.

```pascal
procedure simulate;
var simsteps,wait : word;
  a,V,reportV,calcV,oldpos,dev,oldV,procTime : double;
  cycletime,reportcycletime : dword;
begin
  activeLine := line;    //line length may have changed
  cycletime := 0;
  reportcycletime := 0;
  V := 0;                    //velocity
  reportV := 0;              //velocity to report if > 0
  Apos := Astart*deg2rad; //pendulum  angle in radius      calcV := calcMaxV;    //
maximum velocity at vertical position
  oldpos := 0;               //previous Apos
  oldV := 0;                 //previous velocity
  wait := 100;               //delay to avoid too many updates of percentages%
  repeat
    form1.timer.start;     //record microseconds clock time
    for simsteps := 1 to 10000 do   //10000 simulation steps of 1 micro sec.
    begin
      inc(cycletime);
    Apos := Apos - (V/activeline)*1e-6;
    a := 9.8*sin(Apos);      //acceleration
    V := V + a*1e-6;         //new velocity
    if (oldpos >= 0) and (Apos < 0) then reportV := abs(V); //for error reporting
    oldPos := Apos;          //at vertical position
    if (oldV < 0) and (V >= 0) then //at right top turning point
      begin
        reportcycletime := cycletime;
        cycletime := 0;
      if line <> activeline then   //if changing line length
      begin
        activeLine := line;
        calcV := calcMaxV;
      end;
    end;
    oldV := V;
  end;//for simsteps
  //
  bmrepaint;
  if reportV > 0 then        //error report
  begin
    dev := 100*abs(reportV-calcV)/calcV;
    form1.acclabel.Caption := formatfloat('#0.0000000000',dev);
    reportV := 0;
  end;
  if reportcycletime > 0 then //cycle time report
  begin
    form1.cyclelabel.Caption := formatfloat('#0.000###',reportcycletime*1e-6);
    reportcycletime := 0;
  end;
  dec(wait);
  if wait = 0 then           //proctime report only 1 in 100 steps
  begin
    form1.timer.stop;        //record microseconds clock end time
    procTime := form1.timer.elapsedtime;
    form1.proclabel.caption := formatfloat('#0.0#',0.01*(proctime));
    wait := 100;
  end;
   application.ProcessMessages;
  repeat                     //delay to multiple of 10 milliseconds
    form1.timer.stop;
    until (form1.timer.Elapsedtime >= 10000) or stopflag;
    until stopflag;
end;
```

OWN COMPONENTS

The pendulum project uses two of my own components
1.   RotationButton. A button found on laboratory equipment.
2.   A microseconds counter, based on the CPU clock.
This concludes the pendulum simulator description.

D11 Starter   Expert
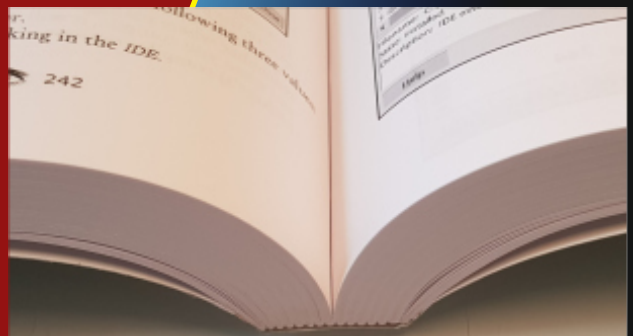
INTRODUCTION

What is a style?
It is a set of graphical details that define the look and feel of a VCL application/FireMonkey application, in Delphi, similar to a theme in Windows.

A style allows you to change the appearance of each component and status of a control.

For example, a slider has the following components: **frame**, **scroll bar** and the **two side buttons** for each direction. For example, the side buttons have the following states: `pressed`, `disabled`, `hot` (emphatic), `normal`.

It is very simple to use styles with a VCL program. Start a new project and put some components on a form like we did here.
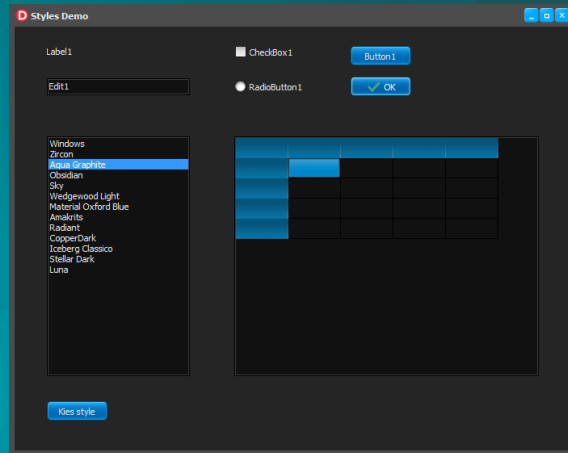
Figure 1: Example of a VCLapplication
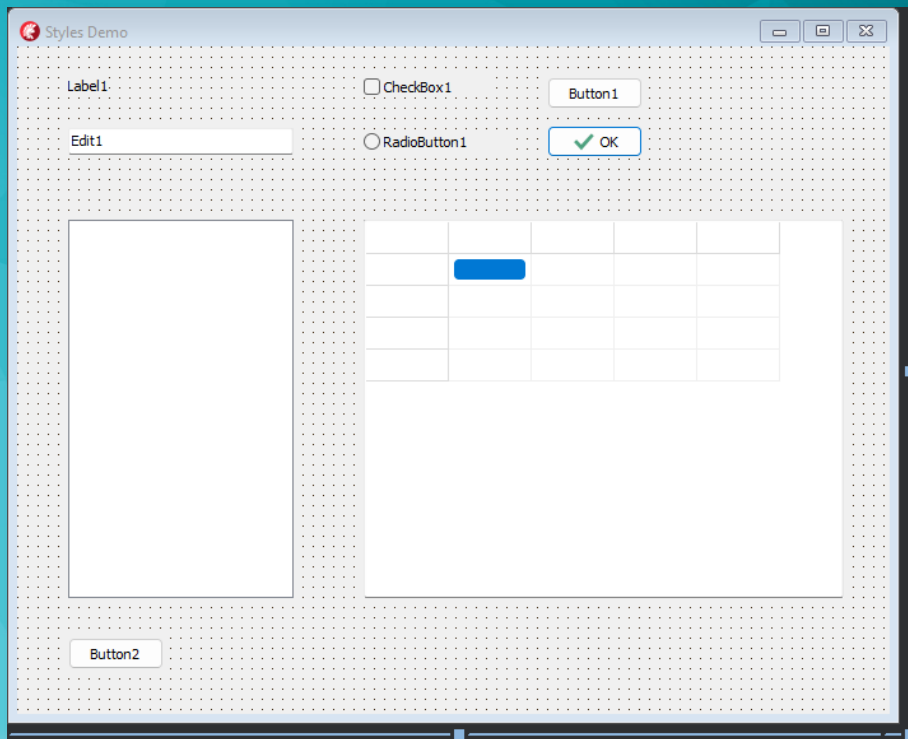
Figure 2: Form in design view.

Then go to the projectoptions (**Project -> Options...**) and choose Appearance there:
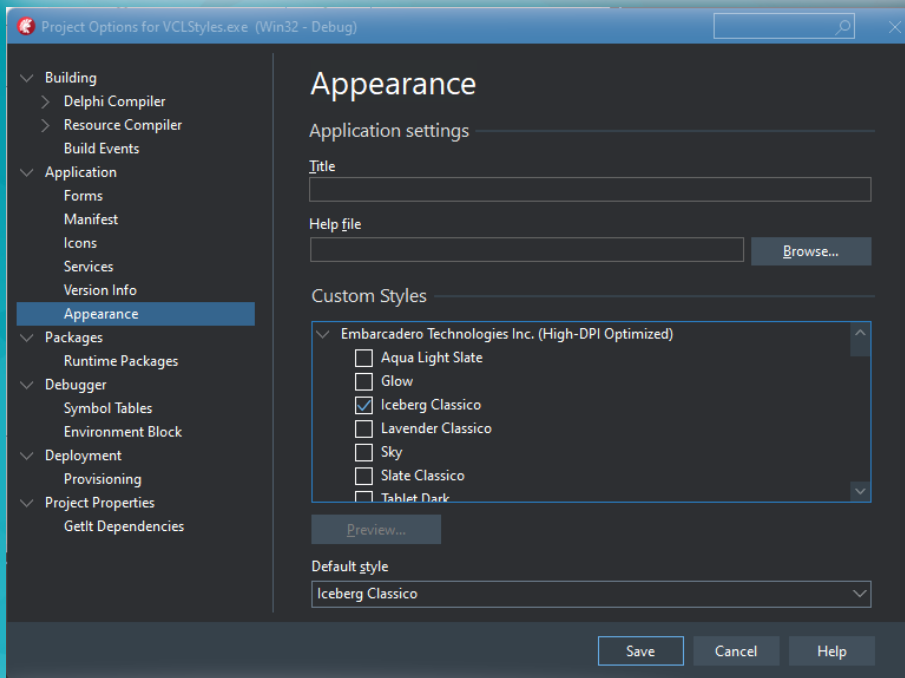
Figure 3: Project Options Dialog.

Tick, for example, Iceberg Classico to act as the default style.
Press Save and run the program for the result.
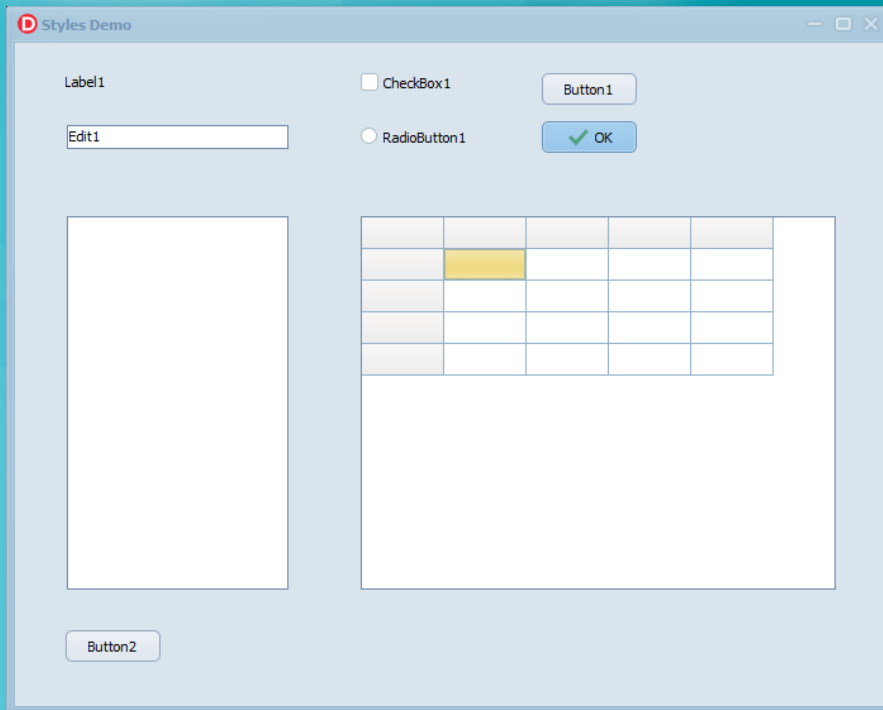
Open **Project->Options...** again. Put tick marks on a number of other styles.
And press **Save** again.
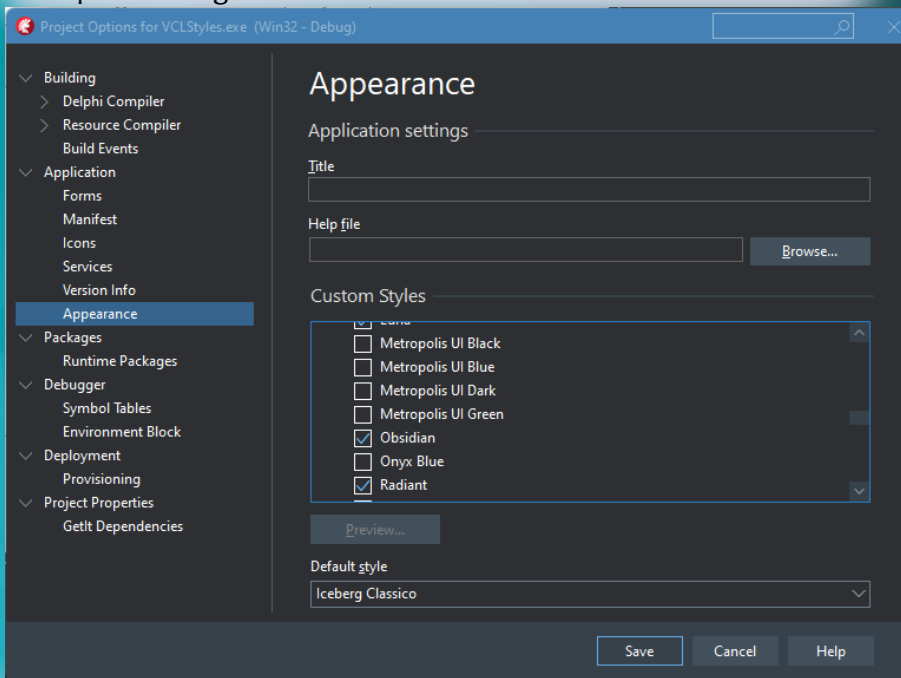
Figure 4: Project Options Dialogue with multiple styles selected.

Insert below `Private` in the `Interface` section so that the form class looks like this:

```pascal
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    CheckBox1: TCheckBox;
    RadioButton1: TRadioButton;
    Button1: TButton;
    ListBox1: TListBox;
    Button2: TButton;
    BitBtn1: TBitBtn;
    StringGrid1: TStringGrid;
  procedure FormCreate(Sender: TObject);
private
  { Private declarations }
  procedure StylesListRefresh;
public
  { Public declarations }
  end;
```

press **Ctrl+Shift+c** to create the skeleton of the procedure. Then go to the form's
`OnCreate` and fill it in as follows and complete the `StylesListRefresh` procedure:

```pascal
procedure TForm1.FormCreate(Sender: TObject);
begin
  StylesListRefresh;
end;

procedure TForm1.StylesListRefresh;
var
  stylename: string;
begin
  ListBox1.Clear;

  // retrieve all the styles linked in the executable
  for stylename in TStyleManager.StyleNames do
  begin
    ListBox1.Items.Add(stylename);
  end;
end;
```

Click on **Button2** and in the Object Inspector, change the **Caption** to choose **Style** and the **Name** to **btnKies**. Double-click the button to create the click event and fill it in as follows:

```
procedure TForm1.btnKiesClick(Sender: TObject);
begin
  TStyleManager.SetStyle(ListBox1.Items[ListBox1.ItemIndex]);
end;
```

Run the program for the result:
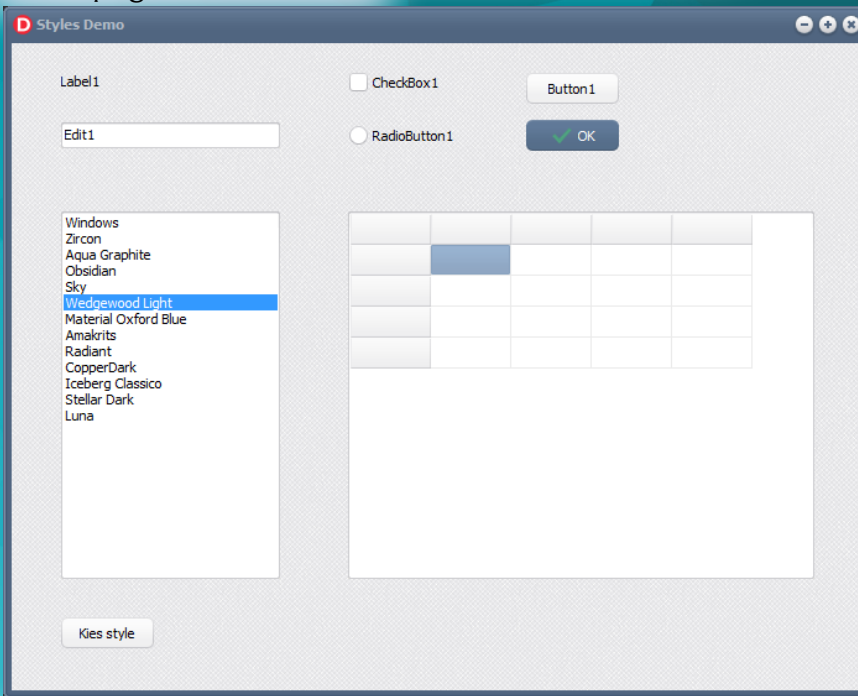
Figure 5: Form with Wedgewood Light chosen as the style.
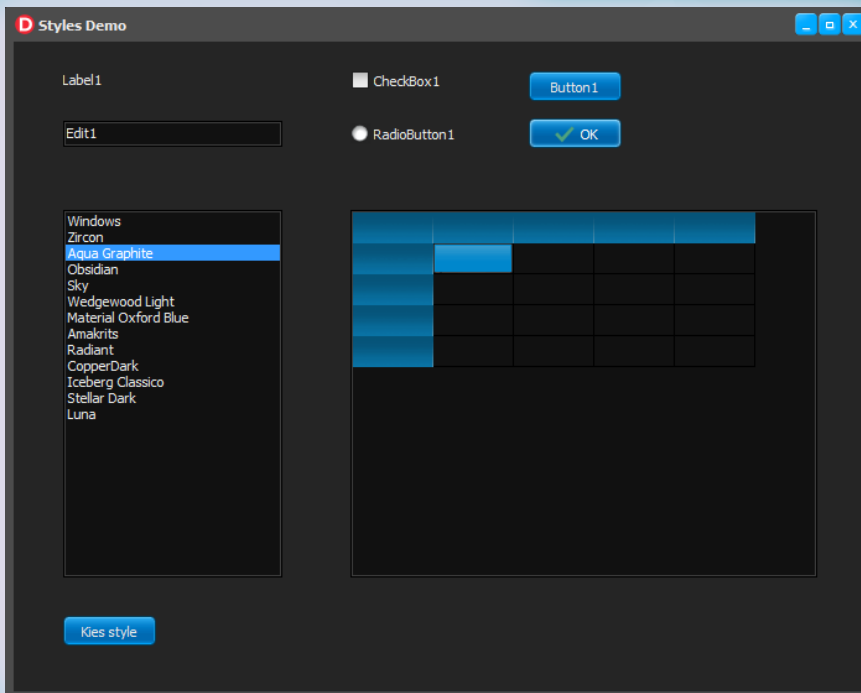
That's how easy styles work at VCL programmes.

ABILITY TO SEE ACTIVE AND INACTIVE CODE BASED ON CONDITIONAL DEFINES

D11 Starter Expert

RAD RAD Studio® 11.2
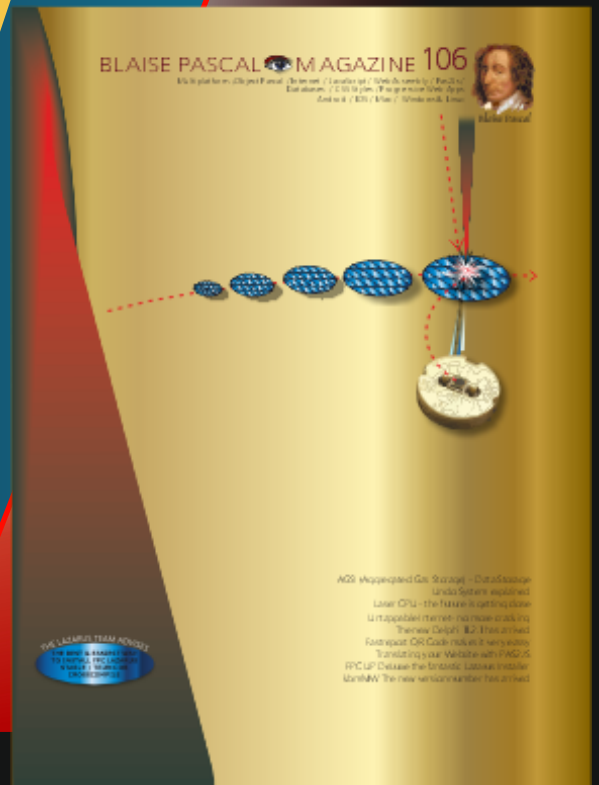
INTRODUCTION

There has been added a nice usability feature in Delphi which is the ability to see active and inactive code based on conditional defines, they are added into this `project` a `Define`.
This shows how to use `debug` and `running` mode.

We add a `Define {$IFDEF DEBUG}` and `else {$ELSE}` followed by a `{$ENDIF}` in here to allow us to see the difference between on or off. (*See Figure 1*)

Figure: 1

But first follow these steps:
set the `self.caption` to become 'Hello debug world in Blaise Pascal Magazine.'
Now you can see we have a `build configuration` of `debug` at this moment:
this is active as you can *see in Figure 2*

We still have full feature ability but we can see immediately that this code is not going to be active at the moment with the configuration we have.
(*See figure 3*)
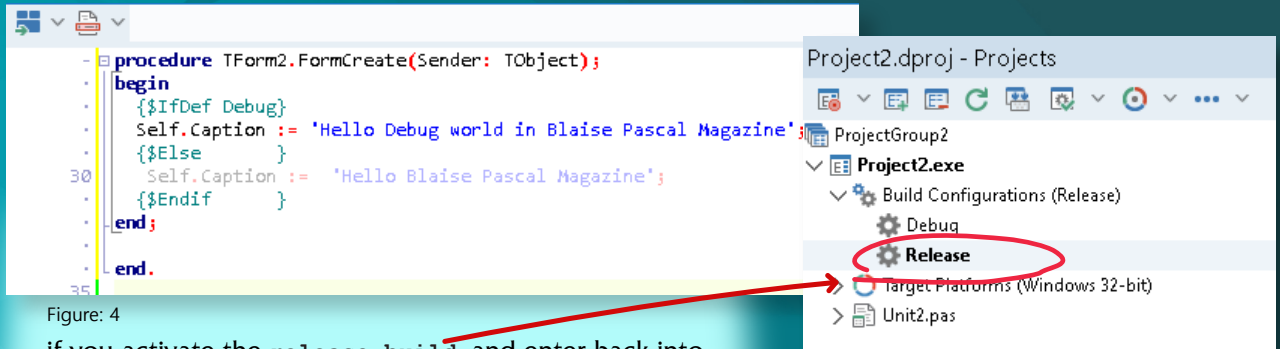
Figure: 2

Figure: 3

Figure: 4

if you activate the `release build` and enter back into the IDE, we can see that it starts editing. (*See figure 4*)

Figure: 6
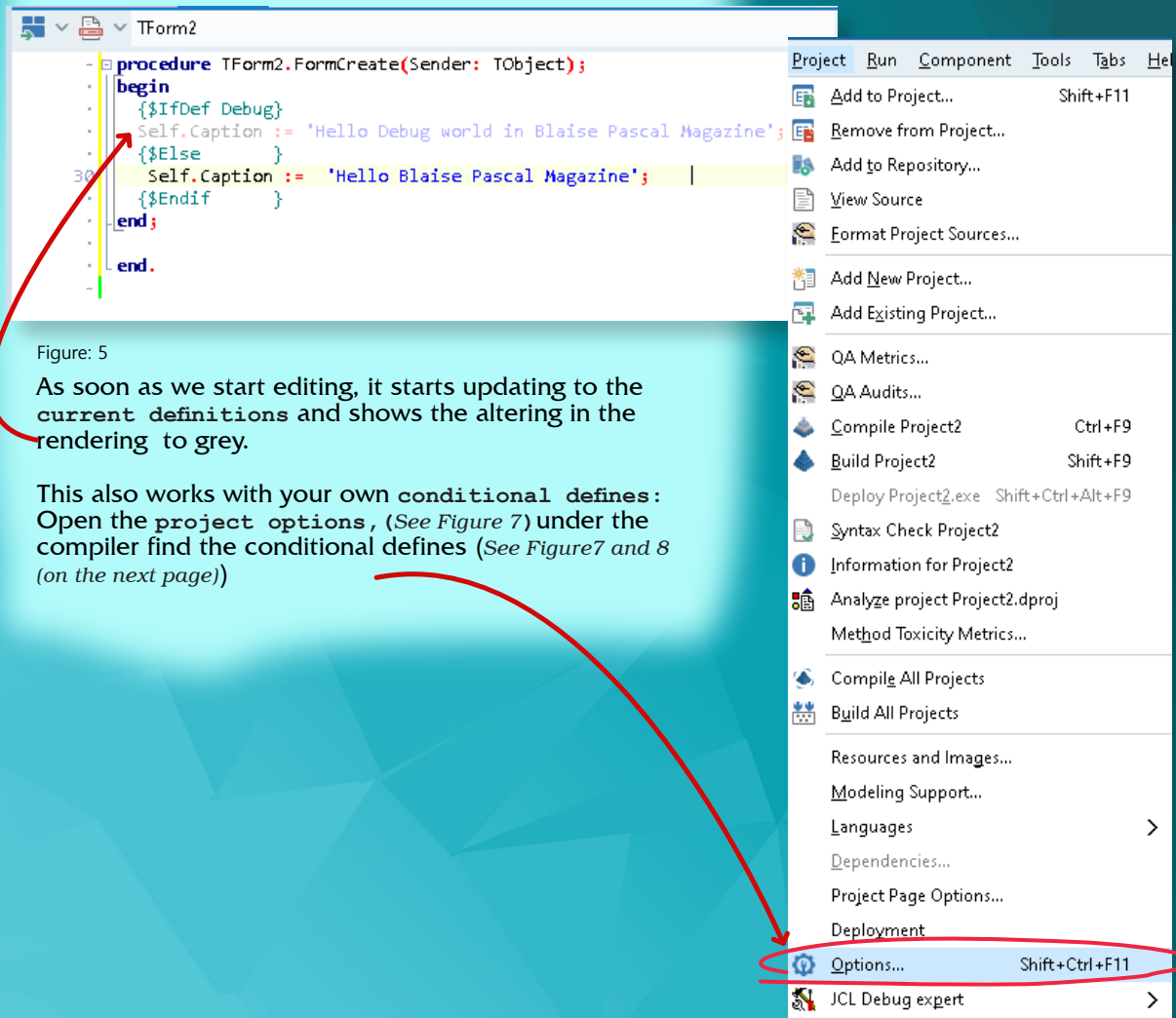
Figure: 5

As soon as we start editing, it starts updating to the `current definitions` and shows the altering in the rendering to grey.

This also works with your own `conditional defines`: Open the `project options`, (*See Figure 7*) under the compiler find the conditional defines (*See Figure7 and 8 (on the next page)*)

Figure: 7
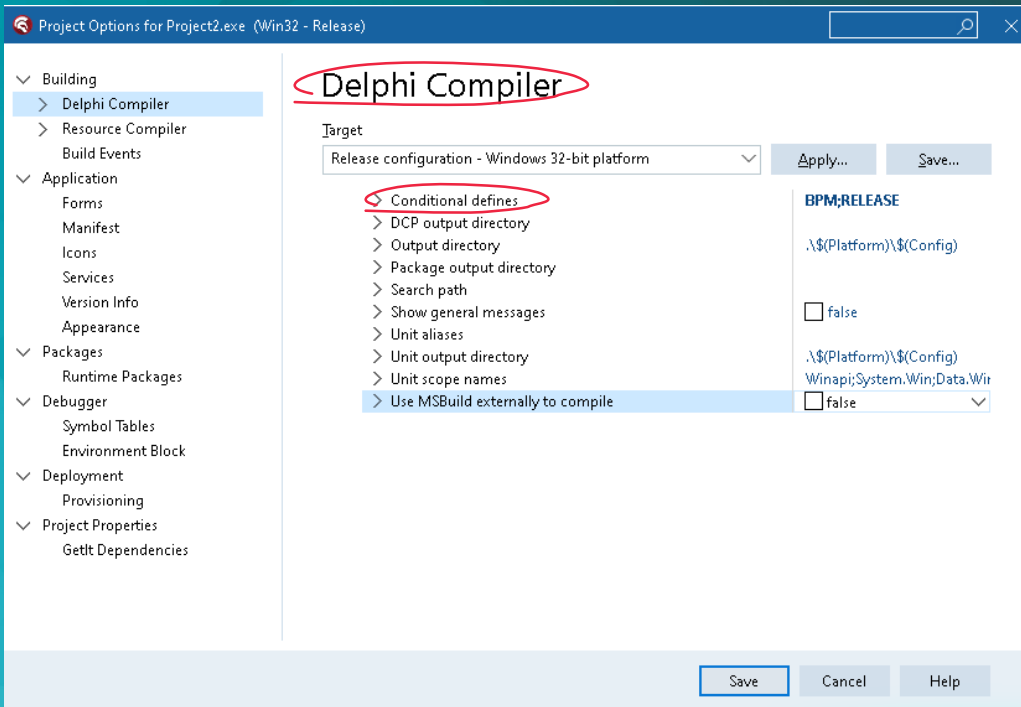
Figure:8
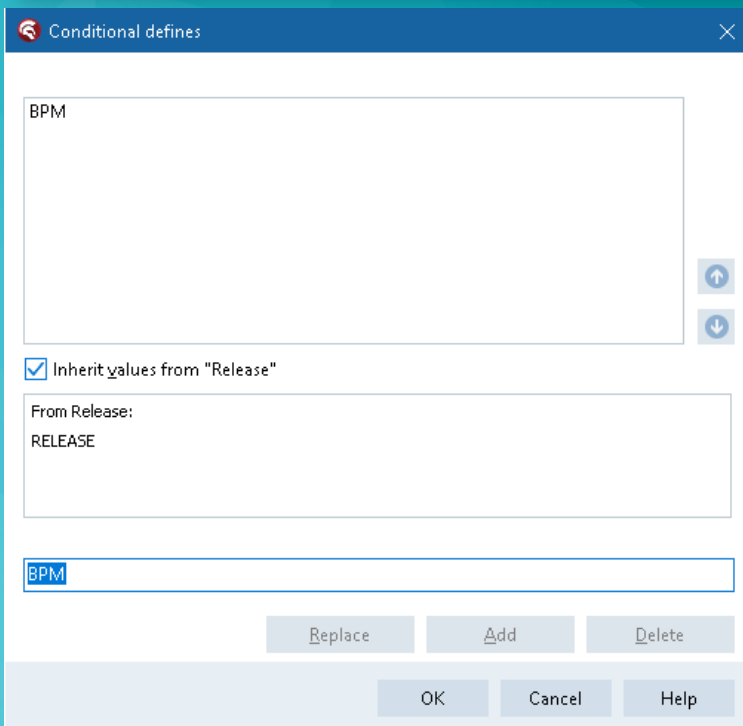
Choose Conditional defines
It will be shown as
BPM;RELEASE (*See figure 8*)
Don't forget to save it…

Let's add one in for BPM and we can now change this to BPM and we can see that it updates to hello BPM

Figure: 9

```
  ▣ ∨ 🗋 ∨  TForm2
      -  ⊟ procedure TForm2.FormCreate(Sender: TObject);
      ·    begin
   27        {$IFNDEF BPM}
      ·        Self.Caption := 'Hello Debug world in BPM';
      ·      {$Else}
   30        Self.Caption := 'Hello BPM';
      ·      {$EndIf}
      ·    end;
      ·
      · └ end.
      -
```

If active in BPM shows immediately during refresh: **Hello BPM** is active (*See figure 10*)

Figure: 10


```
  ▣ ∨ 🗋 ∨  TForm2
      -  ⊟ procedure TForm2.FormCreate(Sender: TObject);
      ·    begin
   27        {$IFDEF BPM}
             Self.Caption := 'Hello Debug world in BPM';
      ·      {$Else}
   30        Self.Caption := 'Hello BPM';
      ·      {$EndIf}
      ·    end;
      ·
      · └ end.
      -
```

If debug active in BPM shows immediately during refresh: **Hello BPM** is grey (*See figure 11*)

Figure: 11

SUBSCRIPTION FOR 2 YEAR OF BLAISE PASCAL MAGAZINE

- English or German
- PDF
- 10 Issues per year
- minimal 60 pages
- Including example projects and code

€ 120
PER 2 YEAR

Starter   **Expert**

ABSTRACT

In a previous contribution, we've shown how to show a PDF in a PAS2JS program. We also showed how to search the PDF. In this article we'll show how to prepare for searching a bunch of PDF files, by indexing them.

1 INTRODUCTION

In a previous article (*BPM Issue Nr 105*) we have demonstrated how to show a **PDF** file in a **PAS2JS** program. The program allowed to show the **PDF**, and to search the contents of the **PDF**. But what if you have lots of **PDF** files - for instance, all issues of Blaise Pascal Magazine - and want to search them ? The program as it was developed can only search a **PDF** file that was loaded in the browser's memory.

Clearly, this is not very efficient for a library of many files.

Another approach is needed for this. The **PDF** files need to be indexed: create a database of words in the **PDF** and record the pages on which each word is present.

By cross-referencing the pages with articles, we can get a list of articles that contain a certain word in their text. When searching, the search term is looked up in the list of words, and a list of locations (*or articles*) can be retrieved.

To be able to do this, we need to read the **PDF** file and extract the words, and create a database with the words. **Free Pascal** and **Lazarus** contain all the tools needed to do this job: the search database tools exists since a long time and are used in the **Free Pascal** documentation at:

`https://www.freepascal.org/docsearch/docsearch.var`

Recently, a set of units to read a **PDF** file have been developed in Free Pascal.

Combining both systems will allow us to create a database of words in **PDF** files and allow to search them. Since we need to read a file before we can extract the words from it, we'll start with discussing how to read a **PDF** file.

2 DISSECTING A PDF FILE

The **PDF** file is ubiquitous. It is the de-facto standard for sending documents over the internet. The **PDF** format was created by **Adobe**, and bears some resemblance to the older **Postscript** format - also by **Adobe**. **Postscript** was a text-based format, and to a degree, so is the **PDF** format. However, for reasons of efficiency and reducing the size of the **PDF** files, binary data has entered the format. The format allows also to append new data to the file, without modifying the previous data in the file: new contents can simply be appended to the file.

The specification of the **PDF** format has been revised several times, each time adding more features; luckily, the format is made in such way that an application does not need to know all features in order to read a **PDF** file.

At its core, the **PDF** Format is simply a series of objects (*called* **'Indirect Objects'**) with dictionaries describing the properties of the objects. To find the objects you need, a cross-reference section is appended, with the positions of all the objects in the file, together with a small dictionary to find some key objects. In a simple **PDF** file, all this can be accomplished with text commands. However, the indirect objects and the cross-reference table can be encoded in binary streams, which can be compressed in various ways, and optionally encoded as well - a mechanism called '**Filters**' in **PDF** parlance.

If content is appended to the file, the newly appended content needs its own cross-reference section which must - obviously - refer to the previous cross-references section for existing content. This process can be repeated ad infinitum: every time when you open a file and add some annotations to a file, the annotations will be appended to the existing file using this mechanism. (*this obviously depends on the software, it may decide to completely rewrite the file as well...*)

All this makes reading a **PDF** file not an easy task: various forms of decompression must be supported, optionally encryption.

When reading a **PDF** file, you will be confronted with various kinds of objects in the file. These objects have been defined in a unit called `fppdfobjects`

- **TPDFXRef**

a cross reference to an indirect reference. This is used to refer to indirect objects, and is used in many places: for example a page list is simply a list of `TPDFXRef` instances referring to the underlying pages (*or another page list object*)

- **TPDFDictionary**

This object encapsulates a **PDF** dictionary.

- **TPDFIndirect**

This is the base building block of the **PDF** file: an indirect object. It has a `ObjectID`, a number which uniquely identifies this object. By itself not very interesting, but each type of object in the **PDF** file is a descendant of this class. every `TPDFIndirect` object normally has a dictionary associated with it (available in the `ObjectDict` property).

- **TPDFDocumentInfo**

This object contains some meta information about the document: author, title etc. It is basically a dictionary object.

- **TPDFFontObject**

This object describes a font. The font may or may not be embedded in the file, usually in compressed form.

- **TPDFPageObject**

This object describes a page in the **PDF** document. It contains a list of resources needed to draw the page: fonts and images. It also contains the content stream: the content stream (*which can be spread over various underlying streams*) is the set of commands needed to draw the page. The `CommandList` property of the page object contains the actual commands that make up the page.

- **TPDFPagesObject**

This object describes a list of pages in the **PDF** document: Pages are organized in a tree, called the **'Page Tree'**.

- **TPDFCommand**

This object represents a drawing command: a page is made up of commands that - all taken together - completely render the page.

- **TPDFCMap**

this object maps character codes to **glyphs*** in a font. Or to unicode characters, as we will see below.

- **TPDFCMapData**

this class contains the actual data of a **CMap** object.
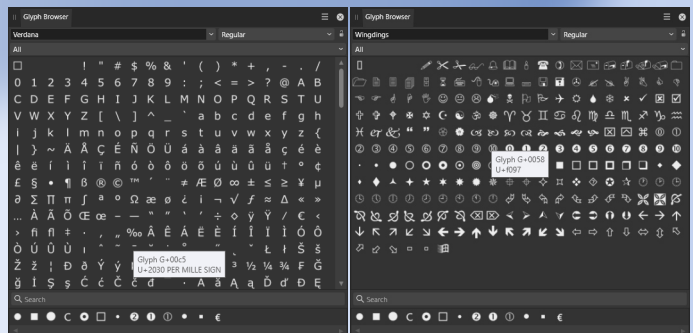
Glyph examples

All these objects are owned by a `TPDFDocument` class, which represents the complete **PDF** document.
It has the following properties:

PDFversion	The PDF version in the header line
StartXref	The root of the cross-reference table.
TrailerDict	Trailer dictionary, set during parsing
PageCount	Page count in this document
PageNodes	Indexed access to top-level indirect objects that represents a page tree node. 0 based. This can be another pages node or a page object
Page	indexed access to a page by 0-based page number.
Pages	an enumerator for the pages
XRefCount	Count of elements in XRefs property.
XRefs	Indexed access to all global XRefs (cross-references) in the PDF.

There are also some methods to find objects in the PDF File:

FindInDirectObject	Find an indirect object by object ID.
FindFont	Find a font object by object ID. May return `Nil`.
GetFont Similar to	**FindFont** but raises an exception if the object is not found.
FindDocumentInfoObject	Find the document information object. May return `Nil`.
FindDocumentInfo	Find document information, resolved to a `TPDFDocumentInfo`, but can be `Nil`. You must free this object yourself.
GetDocumentInfo	Get the document information object. Raises exception if not found. You must free this object yourself.
FindCatalog	Find the document catalog object. Can return `Nil`.
GetCatalog	Similar to **FindCatalog** but raises an exception if not found.
FindPages	Find the top-level pages object. Return `Nil` if none found.
GetPages	Similar to **FindPages** but raises an exception if not found.

To read a document from file, a **PDF** parser is needed. This object will actually read the **PDF** file.
The following is a (*shortened*) declaration of the **PDF** parser object:

```
TPDFParser = class
   Constructor Create(aFile : TStream; aBufferSize : Cardinal = PDFDefaultBufferSize); virtual
   Procedure ParseDocument(aDoc : TPDFDocument); virtual;
   Procedure ResolveToUnicodeCMaps(aDoc : TPDFDocument);
   Property  LoadObjects : Boolean;
   Property  ResolveObjects : Boolean;
   Property  ResolveContentStreams : Boolean;
   Property  OnUnknownFilter : TPDFFilterEvent;
   Property  OnLog : TPDFLogNotifyEvent;
   Property  OnProgress : TPDFProgressEvent;
end
```

The constructor receives a stream with the contents of the **PDF** document. The **ParseDocument** method will actually read and parse the **PDF** Document. The **PDF** parser can function in several modes:

- it can simply read the **PDF** document, and make the cross-reference table available.
- Additionally it can also actually read all indirect objects, and make them available as plain `TPDFIndirect` instances.
- if the indirect objects are read it can also convert them to objects of the correct type: a `TPDFFontObject` for a font, a `TPDFPageObject` for a page, and so on.
- If the objects are converted to typed objects, the parser can also interpret the content stream of a page object, and fill the `CommandList` property of the various `TPDFPageObject` instances.
- If the objects are converted to typed objects, the parser can also interpret the `ToUnicode` **CMap** streams associated with fonts.

As you can see, each step means additional processing of the PDF data. The various steps are controlled by some boolean properties:

- **LoadObjects**　　　　　load all objects when XRef is parsed. Default **True**.

- **ResolveObjects**　　　　When loading objects, resolve objects to their actual class. Default **True**.

- **ResolveContentStreams**　Resolve content streams of pages to commands. The default is **True**.

- **ResolveToUnicodeCMaps**　Immediately parse font ToUnicode CMap streams. Default **False**.

There are some events available:

- **OnUnknownFilter**　　　if an unknown filter is encountered you can manually handle the filter with this event.

- **OnLog**　　　　　　　　a callback to which log messages are sent.

- **OnProgress**　　　　　　a progress event.

So, to read a **PDF** File, the following code is sufficient:

```
$mode objfpc}
{$h+}
uses fppdfobjects, fppdfparser, sysutils, classes;

procedure ReadPDF(const aStream: TStream; aDoc: TPDFDocument);
Var aParser : TPDFParser;

begin
  aParser:=TPDFParser.Create(aStream);
  try
    aParser.ResolveToUnicodeCMaps:=True;
    aParser.ParseDocument(aDoc);
  finally
    aParser.Free;
  end;
end;

var
  F : TFileStream;
  Doc : TPDFDocument;
begin
  F:=TFileStream.Create('mydocument.pdf',fmCreate or fmShareDenyNone);
  try
    Doc:=TPDFDocument.Create; ReadPDF(F,Doc);
    // Do something with your PDF.
  finally
    doc.free;
    f.free;
  end;
end.
```

Or, even easier: the `fppdfparser` unit has a type helper for the `TPDFDocument` class with `LoadFromFile` and `LoadFromStream` methods:

```
{$mode objfpc}
{$h+}
uses fppdfobjects, fppdfparser;

Var Doc : TPDFDocument;

begin
  Doc:=TPDFDocument.Create;
  try
    Doc.LoadFromFile('mydocument.pdf');
    // Do something with your PDF.
  finally
    doc.free;
  end;
end.
```

Which is about as easy as it gets. The `LoadFromFile` and `LoadFromStream` methods have an overloaded variant with which the various boolean properties of the parser can be set.
The `OnLog` and `OnProgress` events can optionally also be specified.
The `pdfdump` example program demonstrates the use of the various objects by allowing you to dump certain information from a PDF file.

3   EXTRACTING TEXT FROM A PDF FILE

Now that we're able to read a **PDF** file, we can start extracting words from it. Alas, this is not so easy as it may sound. **PDF** is a format suitable for reproducing a page exactly the same under all circumstances:
It literally contains instructions to draw the page. a **PDF** reader program reads the instructions, and executes them, by for instance drawing them on the screen, or sending appropriate commands to the printer.

If text happens to be part of a page, then the PDF File simply has instructions to draw images representing the letters (*glyphs*) on certain locations on the page. The **glyphs** are part of a font, and the **PDF** File essentially says something like '**draw me glyphs numbers 12 23 45 at position (145,175) on the page.**'

Traditionally (*and simplifying a bit, we ignore codepages for a second*), the glyph numbers were simply the byte code of the character they represented:
the glyph for the letter 'A' was simply glyph 65 (*the ASCII code for the letter A*). **Thus a text** "**Hello World**" could easily be mapped to glyph numbers in a font, and vice versa.

With the appearance of **Unicode** encoding of characters, and fonts with only a subset of all characters, this mapping became more difficult. Since the **PDF**-format has no idea of text and simply used glyph numbers, **Adobe** came with a solution: a font could specify a `ToUnicode` mapping: this was a mapping from **glyph** numbers to **Unicode** character numbers.
Simply said, this is a mapping which allows an application to map font glyph numbers back to **Unicode** characters. The mapping is in `CMap` format, and a Font dictionary in a **PDF** file can contain an entry towards such a mapping, using the `ToUnicode` key of the dictionary.

This mechanism allows **PDF**-displaying software such as **Acrobat Reader** to offer a text-selection mechanism and a search mechanism: when selecting the text, the **glyphs** that you selected are translated back to letters. When searching, all text- displaying commands are scanned for the **glyphs** they display, the **glyphs** are translated to unicode text using the `ToUnicode` map, and this text is then searched.

This is what we need to do as well if we want to index the **PDF**.

As indicated earlier, the contents of a page exists of a series of drawing commands. All drawing commands are a descendant of `TPDFCommand`. Some of these commands will draw a text (*for example, the* `Tj` *and* `TJ` *operators*), they are all descendants of an abstract `TPDFTextCommand` class, which is defined as follows:

```
TPDFTextCommand = Class(TPDFCommand)
Public
  Function GetFullText(aUnicodeMap:TPDFCMap) :RawByteString;
  Function GetFullText:RawByteString; virtual; abstract;
end;
```

The `GetFullText` function returns all the glyphs for the text to be drawn. For simple fonts, these will map directly to characters and can be used as-is. For complicated fonts, a mapping of character **glyph** IDs to unicode characters is needed:
The `aUnicodeMap` argument to `GetFullText`. Several descendants of this class exist, they implement these **2** methods.
A text drawing command just draws a text using the current font. If we want to know what unicode map we need to use, we also need to check out the font selection command that preceded the text drawing command: it will contain a font name. The font name can be mapped using the page resource dictionary to a font object (*an indirect object*). If that font has a `ToUnicode` map associated with it, this will point to the unicode map to use for all text drawing commands that follow the font selection command. The font selection operator (`Tf`) is represented by the `TPDFTfCommand` class, defined as follows:

```
TPDFTfCommand = class(TPDFCommand)
  property FontName :String Read GetFontName;
  Property FontSize :Integer Read GetFontSize;
end;
```

Armed with these classes, we can now implement an algorithm to extract the text from a PDF page. To show this algorithm, we'll create a small demo that loads a **PDF** and extracts the text of a page and shows it in a memo.

For this, we design a form with 5 elements on it:
- **FEPDF**   A filename edit to select a PDF file.
- **edtPageNo**   An edit to select a page number.
- **lblPageCount**   A label to show the page count of the loaded file.
- **btnShow**   A button to start the action: load a file and extract the text of the selected page.
- **mText**   a memo to show the text of the selected page.

The code for this application is quite simple. The process is started with the `OnClick` event handler of `btnSelect`: If the **PDF** file was not yet loaded, or a different file was chosen, first the file is loaded using `LoadPDF`:

```
procedure TMainForm.btnShowClick(Sender: TObject);
Var aPage : Integer;
begin
  if (FDoc=Nil) or (FEPDF.FileName<>FloadedFile) then
    LoadPDF(FEPDF.FileName);
  aPage:=StrToIntDef(edtPageNo.Text,0);
  if (aPage>0) and (aPage<FDoc.PageCount) then
    ShowPageText(aPage)
  else
    ShowMessage('Invalid page, valid values 1-'+IntTostr(FDOc.PageCount));
end;
```

When the file is loaded, the page number is examined, and if it is valid, the page text is shown using ShowPageText. If it is invalid, an error message is shown. The LoadPDF is not substantially different from what we've shown earlier:

```
procedure TMainForm.LoadPDF(const aFileName : string);
begin
  FreeAndNil(FDoc);
  FDoc:=TPDFDocument.Create;
  Try
    FDoc.LoadFromFile(aFilename);
    FLoadedFile:=aFileName;
    lblPageCount.Caption:='/'+IntToStr(FDoc.PageCount);
  Except
    on E : Exception do
      ShowMessage('Error loading PDF file :'+E.Message);
  end;
end;
```

The **ShowPageText** is where the real work happens. It is in fact a simple loop, with 2 cases: When a font selection command is encountered, the CMAP for the font's ToUnicode mapping is extracted and saved.

```
procedure TMainForm.ShowPageText(aPageNo : Integer);
Var aPage :              TPDFPageObject;
    Cmd :                TPDFCommand;
    FontName, Rawtext :  RawByteString;
    aFontRef :           TPDFRefData;
    UnicodeMap :         TPDFCMap;
    aFontObj :           TPDFFontObject;

begin
  mText.Clear;
  aPage:=FDoc.Page[aPageNo];
  for Cmd in aPage.CommandList do begin
    if Cmd is TPDFTfCommand
    then
      begin
        FontName:=TPDFTfCommand(Cmd).FontName;
        if (FontName<>'') and (FontName[1]='/') then
          Delete(FontName,1,1);
        aFontRef:=aPage.FindFontRef(FontName);
        aFontObj:=FDoc.FindFont(aFontRef); // TPDFFontObject
        if Assigned(aFontObj) then
          UnicodeMap:=aFontObj.UnicodeCMap
        Else UnicodeMap:=nil;
      end
    else
      If cmd is TPDFTextCommand then
        begin
          rawText:=TPDFTextCommand(Cmd).GetFullText(UnicodeMap);
          SetCodePage(RawText,CP_UTF8);
          mText.Lines.Add(RawText);
        end;
  end;
end;
```

When a text drawing command is encountered, the `GetFullText` method is used to extract the full text.
If there was an active unicodemap, then that is used to interpret the text.
If there was no unicodemap, then the font is assumed to be a simple font and the raw text should be
readable (*this is not 100% correct, as we'll see*) The codepage of the text is set to **UTF8** because that is what
the **Lazarus LCL** framework expects, and then the text is added to the memo.
The result can be seen in *figure 1 on page 9.*
There are 2 things to note in the screenshot:

- The page text contains some characters that cannot be displayed.
  This is the letter combination fi for which a special glyph is used,
  and which cannot be displayed in the memo.

- The text contains line breaks at unexpected places. This is a consequence of the simple mechanism
  we used to extract the text.
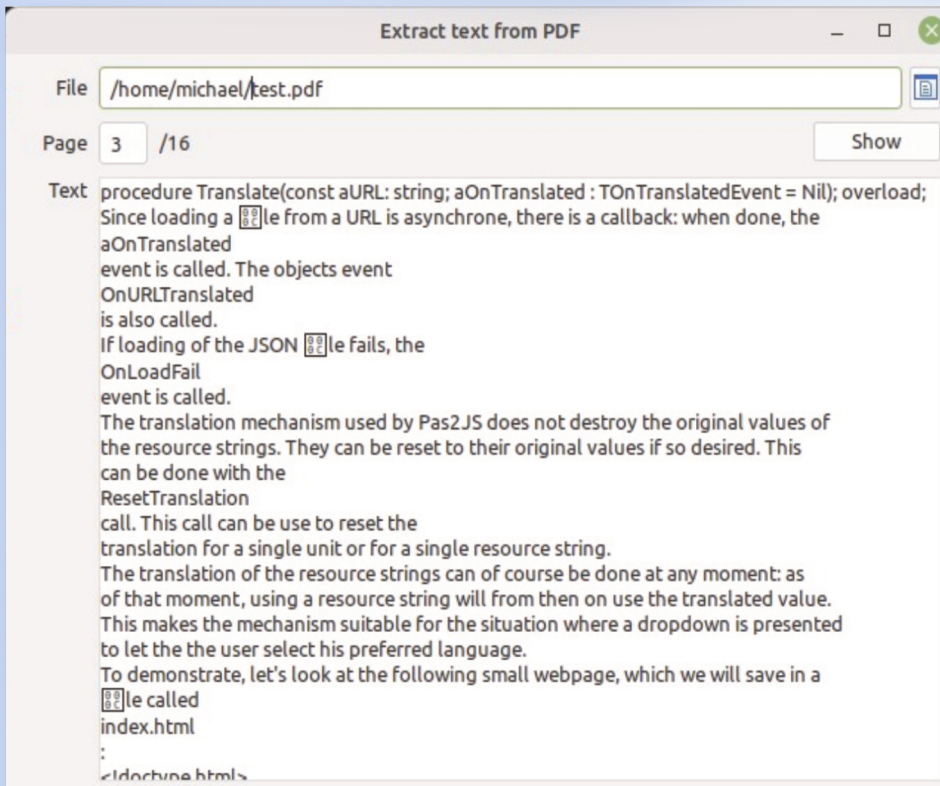
Figure 1: Extracting the text of a PDF

Whenever a font change happens in the displayed text (*another font, or simply the same font printed bold or italic*),
the **PDF** will contain a paint instruction for each part of the text:
the text leading up to the font change, the text in the alternative font, and the rest of the text,
which can be in the original font or yet another font.
Note that there is no guarantee that the instructions will appear in the order that the words appear in the text: the
lay-outing mechanism could decide to first draw all words in one font, then words in another font,
thus saving on font selection commands.
In order to reconstruct actual lines or paragraphs of text, the positioning of the text would also need to be
examined: all text drawing commands that are on the same baseline can be expected to form a single line of text,
although there is no guarantee that this is actually so.
For the purposes of extracting words from a text, the above mechanism is sufficient:
disregarding words for which a different font was selected halfway a word, or hyphenized words, a text drawing
command will contain complete words.

4   A CLASS TO INDEX WORDS

Free Pascal comes with a unit called `fpIndexer`. This contains some abstract classes which serves as the base class for a word indexing and searching mechanism.

```pascal
TFPIndexer = class(TComponent)
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  function   IndexStream
        (const AURL: UTF8String; ADateTime: TDateTime; S: TStream; Reader: TcustomFileReader);
  function   IndexFile(AURL: UTF8String; AllowErrors: boolean; const ALanguage: UTF8String ='
  function   Execute(AllowErrors: boolean): int64;
  property   ErrorCount: int64 read FErrorCount;

published
  property   Language: UTF8String;
  property   OnProgress: TIndexProgressEvent;
  property   UseIgnoreList: boolean;
  property   IgnoreNumeric: boolean;
  property   CommitFiles: boolean;
  property   Database: TCustomIndexDB;
  property   ExcludeFileMask: UTF8String;
  property   FileMask: UTF8String;
  property   SearchPath: UTF8String;
  property   SearchRecursive: boolean;
  property   DetectLanguage: boolean;
  Property   CodePage : TSystemCodePage;
  Property   StripPath : String;
end;
```

This class can be used to index the contents of files: the **IndexStream** and **IndexFile**
methods will create a database of words from the contents of a text file.
The various properties allow to control the search for files on disk and how to index them.
The meaning of most of these properties is clear.
Only '**CommitFiles**' needs explanation: the default behaviour is to enter all data in one big transaction in the database.
When this property is set to `True`, a commit will be done after each indexed file.
The indexer uses a factory to create a tokenizer per file type (*determined by its extension*):
the tokenizer will return all the words in a file for a certain file type.

You can register a handler for a file type, and this handler will then be used to read the words
and fill the database. The thing to do would be to create a handler for PDF files.
For educational reasons, we'll take a slightly different approach here.
The interesting property is the `Database` property:
this determines where the indexer writes the words and matches.
The **TCustomIndexDB** class is defined as follows:

```pascal
TCustomIndexDB = class(TComponent)
public
  procedure CreateDB; virtual; abstract;
  procedure Connect; virtual; abstract;
  procedure Disconnect; virtual;
  procedure CompactDB; virtual; abstract;
  procedure BeginTrans; virtual; abstract;
  procedure CommitTrans; virtual; abstract;
  procedure DeleteWordsFromFile(URL: UTF8String); virtual; abstract;
  procedure AddSearchData(ASearchData: TSearchWordData); virtual; abstract;
  procedure FindSearchData(SearchWord: TWordParser; FPSearch: TFPSearch;
                    SearchOptions: TSearchOptions); virtual; abstract;
  Function GetAvailableWords(out aList : TUTF8StringArray; aContaining : UTF8String;
                    Partial : TAvailableMatch) : integer; virtual; abstract;
  procedure CreateIndexerTables; virtual; abstract;
end;
```

Most of these methods are self-explaining.

CreateDB	Create the index database.
CreateIndexerTables	Create the tables needed by the indexer.
Connect	Connect to the index database.
Disconnect	Disconnect from the index database.
CompactDB	Clean up the DB.
BeginTrans	Start a transaction (*if available in the backend*).
CommitTrans	Commit a transaction (*if available in the backend*).
DeleteWordsFromFile	Delete all word matches from a file.
AddSearchData	Add a word match.
FindSearchData	Find word matches.
GetAvailableWords	Get a list of available words matching a pattern.
	The **FPIndexer** package contains several descendants of this class:
TPGIndexDB	Is a descendant that writes the data to a **PostgreSQL** database. Implemented in the unit pgindexdb
TFBIndexDB	Is a descendant that writes the data to a **Firebird/Interbase** database. Implemented in the unit fbindexdb
TSQLiteIndexDB	Is a descendant that writes the data to a **SQLite** database. Implemented in the unit sqliteindexdb
TMemIndexDB	Is a descendant that keeps the data in memory. Implemented in the unit memindexdb
TFileIndexDB	Is a descendant that keeps the data in memory, but can additionally save the data to a file. Implemented in the unit memindexdb

Rather than create a file type handler for **PDF** files, we can use one of the descendants of the **TCustomIndexDB** class directly to fill a database with search words.
We'll do this in a class **TPDFindexer** (*implemented in the unit fppdfindexer*) with the following interface, it can be used to index a single **PDF** file.

```
TPDFIndexer = Class(TComponent)
Public
  Procedure Connect;
  Procedure CreateTables;
  Procedure Disconnect;
  Procedure IndexPDF(const aFileName : string); overload;
  Procedure IndexPDF(const aStream : TStream); overload;
  Property PDFURL : String;
  Property Language : String;
  Property Indexer : TSQLDBIndexDB;
  Property MinWordLength : Integer;
  Property IgnoreWords : TStrings;
  Property OnLog : TPDFIndexLogEvent;
end;
```

The **Connect, CreateTables** and **Disconnect** methods simply call their counter- parts in the **Indexer**, which is a descendant of TCustomIndexDB that writes to a postgres or firebird database.
Which of these three databases is used depends on a define in the beginning of the file:

```
{$DEFINE USEFIREBIRD}
{$DEFINE    USEPG}
```

Since the class only indexes a single document, the **URL** to write in the database can be
specified in the **PDFURL** property. It will be set to the **PDF** filename if nothing was set. The
`Language` property has a similar purpose: it determines the language written into the database.
The `IgnoreWords` strings property allows to provide a list of words to ignore, and the `MinWordLength`
is set by default to 3, meaning that words with length less than 3 will not be inserted in the database.
The log event serves to log progress.
The main method is **IndexPDF**, which starts the whole indexing process.
The method accepts a filename or a stream. It is simple enough:

```
procedure TPDFIndexer.IndexPDF(const aStream: TStream);
begin
  DoLog('Start indexing PDF %s',[PDFURL]);
  Connect;
  try
    DoIndexPDF(aStream);
    DoLog('Done indexing PDF %s',[PDFURL]);
  finally
    Disconnect;
  end;
end;
```

The protected `DoIndexPDF` reads the **PDF** file and indexes the words by calling the
`IndexPDFPage` for each page:

```
procedure TPDFIndexer.DoIndexPDF(const aStream: TStream);
var
  aPageNo : integer;
  aPage : TPDFPageObject;
  aParser : TPDFParser;
  Doc : TPDFDocument;

begin
  aPageNo:=0;
  Doc:=Nil;
  aParser:=TPDFParser.Create(aStream);
  try
    Doc:=TPDFDocument.Create();
    aParser.ResolveToUnicodeCMaps:=True;
    aParser.ParseDocument(Doc);
    For aPage in Doc.Pages do
    begin
      Inc(aPageNo); IndexPDFPage(aPage,aPageNo);
    end;
  finally
    doc.free; aParser.Free;
  end;
end;
```

The `IndexPDFPage` simply makes sure that each page is handled in a separate
transaction, the real work is done in the `DoIndexPDFPage` method:

```
procedure TPDFIndexer.IndexPDFPage(const aPage: TPDFPageObject; aPageNo: Integer);
begin
  Indexer.BeginTrans;
  try
    DoIndexPDFPage(aPage,aPageNo);
    Indexer.CommitTrans;
    DoLog('Indexed page %d',[aPageNo])
  except
    On E : exception do
    DoLog('Error %s while  indexing page %d: %s',[E.ClassName,aPageNo,E.Message]);
  end;
end;
```

The **DoIndexPDFPage** method works similar to what our **PDF** page text display
program did, it scans the commands of the page for texts. Every found text is split into
words, which are then saved in the database:

```pascal
TPDFIndexer.DoIndexPDFPage(const aPage: TPDFPageObject; aPageNo: Integer);
Var
  aData:          TSearchWordData;
  aCmd :          TPDFCommand;
  aWord, FontName, Rawtext : RawByteString;
  aFontRef :      TPDFRefData;
  UnicodeMap :    TPDFCMap;
  aFontObj :      TPDFFontObject;
  aDoc :          TPDFDocument;

begin
  aDoc:=aPage.Document;
  With aData do
    begin
      FileDate:=Date;
      Language:='EN';
      Position:=aPageNo;
      URL:=PDFURL;
    end;
  for aCmd in aPage.CommandList do
    Begin
      if aCmd is TPDFTfCommand then
      begin
        FontName:=TPDFTfCommand(aCmd).FontName;
        if (FontName<>'') and (FontName[1]='/') then
          Delete(FontName,1,1);
        aFontRef:=aPage.FindFontRef(FontName);
        aFontObj:=aDoc.FindFont(aFontRef); // TPDFFontObject
        if  Assigned(aFontObj) then
          UnicodeMap:=aFontObj.UnicodeCMap
        else
          UnicodeMap:=nil;
      end
    Else If aCmd is TPDFTextCommand then
      begin
        rawText:=TPDFTextCommand(aCmd).GetFullText(UnicodeMap);
        aData.Context:=Rawtext;
        SetCodePage(RawText,CP_UTF8);
        for aWord in DoSplit(RawText) do
        begin
          aData.SearchWord:=aWord;
          Indexer.AddSearchData(aData);
        end;
      end;
    end;
end;
```

The TSearchWordData record used to insert data in the index database is defined in the fpIndexer unit, and is used by the indexer class both for inserting data as when returning results when searching for data:

```pascal
TSearchWordData = record
  Context:    UTF8String;
  FileDate:   TDateTime;
  Language:   UTF8String;
  Position:   int64;
  Rank:       integer;
  SearchWord: UTF8String;
  URL:        UTF8String;
end;
```

The meaning of these fields are:
- **Context**   some context around the word: usually the line on which the word appears.
- **FileDate**   The file date of the file in which the word appeared.
- **Language**   A 2-letter language code.
- **Position**   A position in the file. In our application, we will insert the page number here.
- **Rank**   an integer signifying the "rank": this can be used to score the word.
- **SearchWord**   The word actually to be inserted. Will be returned on search.
- **URL**   the file in which the word was found. This will be the name of the PDF file.

The DoSplit method used to split the text into words deserves some attention.
It is a virtual method, so it can be overridden to implement a different mechanism.
The DoSplit method as written here only accepts the Latin letters 'A'..'Z' as words: if a different language
(*Cyrillic or an eastern language*) is needed, then this method needs to be overridden.

```pascal
function TPDFIndexer.DoSplit(const aText: RawbyteString): TStringDynArray;
Var aCount : Integer;
    c : AnsiChar;
    Cleaned : String;

  Procedure MaybeAdd;
  begin
    if (Cleaned<>'') and AllowedWord(Cleaned) then
      begin
        Result[aCount]:=Cleaned;
        inc(aCount);
      end;
    Cleaned:='';
  end;

begin
  aCount:=0;
  Cleaned:='';
  Result:=[];
  SetLength(Result,Length(aText) div MinWordLength);
  for C in aText do
    if
      Upcase(C) in ['A'..'Z']
    then
      Cleaned:=Cleaned+C
    Else
      MaybeAdd;
      MaybeAdd;
  SetLength(Result,aCount);
end;
```

The method is not very difficult to understand. Note that here the check on allowed words is performed: if a word is not allowed, it is not returned. The `AllowedWord` is quite simple:

```pascal
function TPDFIndexer.AllowedWord(aWord : String) : Boolean;
begin
  Result:=Length(aWord)>MinWordLength;
  if Result then
  Result:=FHash.Find(aWord)=Nil;
end;
```

The `FHash` variable is a hash list built from the list of words in the `IgnoreWords` property. With this, the indexer class is ready. All that needs to be done is to use it in a program!

5 THE PDF INDEXER PROGRAM

The indexer program is a simple program with inputs for the `TPDFIndexer` class. It allows to index a single file or multiple files: it can scan a directory for PDF files. It also has a button to create the needed tables for the indexer, and a memo to show log output and progress messages. The form is depicted in *figure 2 on page 14.* Usage is as follows:

- Enter the database connection info. The database for the index must already exist somewhere, and you must supply the location plus the user credentials to connect to it.
  Reminder: The `fppdfindexer` unit is by default set up to use Postgres database.
  If you wish to use another database (Firebird), you must change the defines in the start of the `fppdfindex` unit.
- Press the 'Test connection button'. You will get a message if the connection failed or succeeded. The 'Create tables' and 'Index PDFs' buttons will become active.
- If you did not yet create the index tables, use the 'Create tables' to create the tables.
- Set the other parameters: a file with words to ignore, the language code.
- Select a PDF file or a directory with PDF files to index.
- Press the 'Index PDFs' button to start the indexing process.

When the form is created, it creates an instance of the TPDFIndexer class:

```pascal
procedure TMainForm.FormCreate(Sender: TObject);
begin
  FIndexer:=TPDFIndexer.Create(Self);
  FIndexer.OnLog:=@DoIndexLog;
  if FileExists(SIniFile) then
    ConfigDB(SIniFile)
end;
```

If a `config` file (`SIniFIle`) exists, it is read and used to restore the contents of the various inputs: this
way the user does not need to fill in this data every time. This is simply reading the contents of an .ini
file, and the interested reader can consult the sources of the sample program.

The operation of the program is quite simple: All connection data must be filled in. Using the Test
Connection button, the connection can be tested. When the test is successful, the database tables can be
made, or, if the tables are known to exist, the index button can be used to start indexing files.

The test connection button executes the `TestConnection` method

```pascal
procedure TMainForm.TestConnection;

begin ConfigConnection;
  Try
    FIndexer.Indexer.Connect;
    FIndexer.Indexer.Disconnect;
    FCanConnect:=True;
    SaveDBConfig(SIniFile);
  except
    on E : Exception do
      ShowMessage('Could not connect to index database:'#10+E.Message);
  end;
end;
```

The `configconnection` method simply transfers the
data entered in the edits to the various properties of the indexer:

```pascal
procedure TMainForm.ConfigConnection;
begin
  With Findexer.Indexer do
    begin
      DatabasePath:=edtDatabaseName.Text;
      Hostname:=edtHostname.Text;
      UserName:=edtUserName.Text;
      Password:=edtPassword.Text;
    end;
end;
```

The main method of this program is the `actIndexExecute` method.
This the `OnExecute` event handler of the `actIndex` action.
It starts by setting the language property and reading the list of words to ignore. After that it either indexes
the requested file using `IndexFile`, or calls `IndexDirectory`: a routine that retrieves all **PDF** file names
from the selected directory, and calls `IndexFile` using the found names.

```pascal
procedure TMainForm.actIndexExecute(Sender: TObject);
Var Msg : String;

begin
  if (FEIgnoreWords.FileName<>'') and FileExists(FEIgnoreWords.FileName) then
    FIndexer.IgnoreWords.LoadFromFile(FEIgnoreWords.FileName);
  FIndexer.Language:=edtLanguage.Text;

  FFileCount:=0;
  FLastError:='';
  if RBFile.Checked then
    IndexFile(FEPDF.FileName)
  Else
    IndexDirectory(DEPDFs.Directory);
  Msg:=Format('Done indexing %d files.',[FFileCount]);
  if FlastError<>'' then
    Msg:=Msg+#10'There were errors:'#10+FLastError;
  ShowMessage(Msg)
end;
```

At the end of the routine, some statistics are shown. The `IndexFile` is simplicity itself.
The main logic consist of catching possible errors and displaying them in the output log.

```pascal
procedure TMainForm.IndexFile(const aFile : string);
Const
  ErrMsg = 'Error %s while  indexing %s: %s';

begin
  try
    FIndexer.Connect;
    try
      Inc(FFileCount);
      FIndexer.IndexPDF(aFile);
    finally
      Findexer.Disconnect;
    end;
  except
    On E : Exception do
      begin
        FLastError:=Format(ErrMsg,[E.ClassName,aFile,E.Message]);
        DoLog(FLastError);
      end;
  end;
  Application.ProcessMessages;
end;
```

These are the main methods of the program.
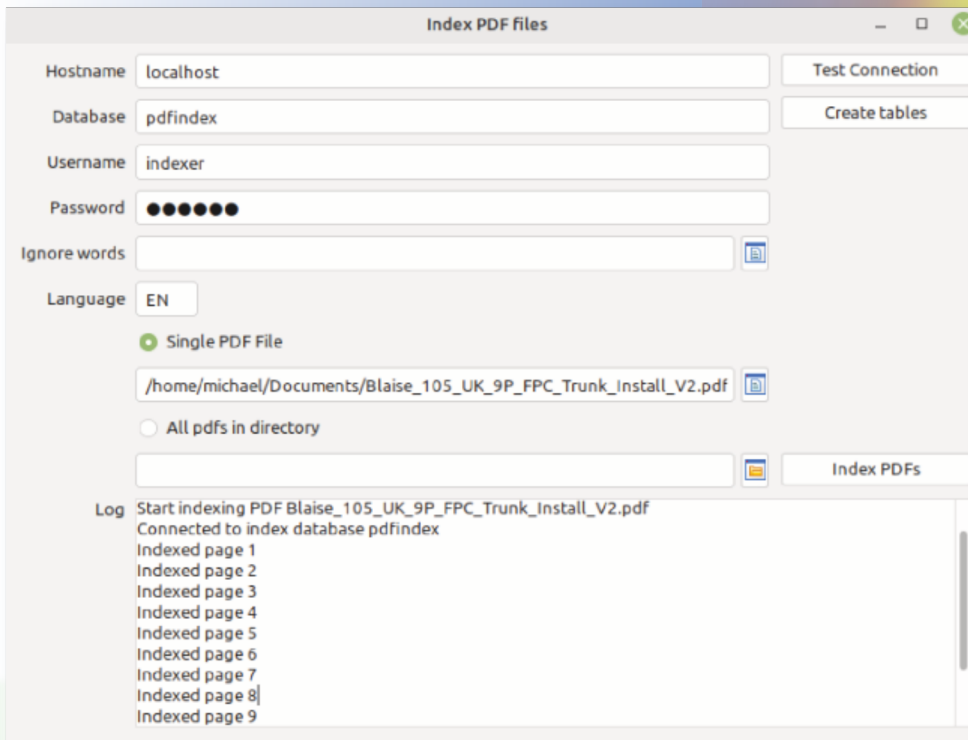
Figure 2: The PDF indexer program

6 CONCLUSION

In this article, we constructed a **PDF** indexing program using units provided with **Free Pascal** and **Lazarus**. Along the way, we touched upon many subjects: the structure of a **PDF** file, the many objects needed to extract text from a **PDF** file. We also showed the classes made available by **Free Pascal** to create an indexing database.

This database is now ready to be used in a searching program: this can be a native program or a **PAS2JS** program. That is the subject of a second article. **Note** that the indexer created by **Free Pascal** is pretty straightforward and the design is limited: only full words can be searched.

For more powerful searching, a full-text indexer such as **manticore search** must be used.

We'll investigate that in a separate contribution.

1. **One year Subscription**
2. **The newest LIB Stick**
   - **All issues 1- 107/108**
   - **On Credit Card and download**
3. **Lazarus Handbook**
   - **PDF including Code**
4. **Book Learn To Program**
   - **using Lazarus PDF including**
   19 lessons and projects
5. **Book Computer Graphics**
   Math & Games
   PDF including ±50 projects

2

NORMAL PRICE € 275
CHRISTMAS
PRICE € 150

CHRISTMAS
& NEW YEAR

SUPERPACK
2022/23

Offering some tips and suggestions you can consider
to help the compiler be faster and consume less
memory using the current version of Delphi

`https://blog.marcocantu.com/`

D11   Starter   Expert

INTRODUCTION

Over the last couple of years,
Embarcadero has put a significant effort into optimizing the Delphi Object Pascal compilers'
performance and reducing its memory usage, so that our customers with large applications can
have a better experience working with Delphi. The primary focus has been on the Win32 compiler,
but other compilers have been improved as well.

We made significant progress in many use cases scenarios, but at the same time by looking at many
projects from our customers we discovered a number of cases in which the compiler isn't behaving
as expected. We plan addressing most of those problems in the near future, even if some of the
scenarios are bordering on being invalid (*like passing non-existing folders to the compiler*).
At the same time we are getting requests from customers if there is anything they can do in their
`project configuration` or `source code` to make things better for their developer today.

This is the reason for this blog post: Offering some tips and suggestions you can consider (*if they
are applicable to your code base*) to help the compiler be faster and consume less memory using the
current version of Delphi.

NOTICE that most of the ideas mentioned in this article, help both with compilation and when using
`Code Insight` powered by `Delphi LSP`, given it uses the same compiler behind the scenes.
By helping the compiler, you are also helping Code Insight

INVALID SEARCH PATH OR LIBRARY PATH ENTRIES

The compiler looks for any unit referenced in the code by uses statements in the project folder, the
project search path, the library path, and some other locations. We have noticed that having invalid
folders in these paths causes the compiler to repeatedly look into those locations, an operation that
is much slower, at the operating system level, for non-existing paths.

You can see an example of a report we received, titled 'Delphi compiler works extremely slowly
when several invalid paths are present in LIB PATH' at
`https://quality.embarcadero.com/browse/RSP-3931,` as an example.

RAD **RAD Studio® 11.2**

Loading assembly Embarcadero.Studio.Modeling.dll   **e**mbarcadero

UNIT ALIASES

Unit aliases offer the option to map a unit name in a uses statement to a different real unit name. This became handy when migrating old code and was used by Delphi over the years, when some of the core units were merged or renamed.

For example, in some early versions of Delphi you'd find the following unit aliases:

```
WinTypes=Windows; WinProcs=Windows; DbiProcs=BDE; DbiTypes=BDE; DbiErrs=BDE
```

Today, there is no default unit alias defined for new projects, but you might have existing projects with unit aliases. The recommendation is to remove them (*possibly one by one*) and update the source code of your projects with references to the correct units.

A large use of unit aliases adds some extra work to the compiler.
While to our knowledge this causes only a very little extra overhead, it is still a bit of extra time and the effort also helps you keep your code cleaner and more readable.
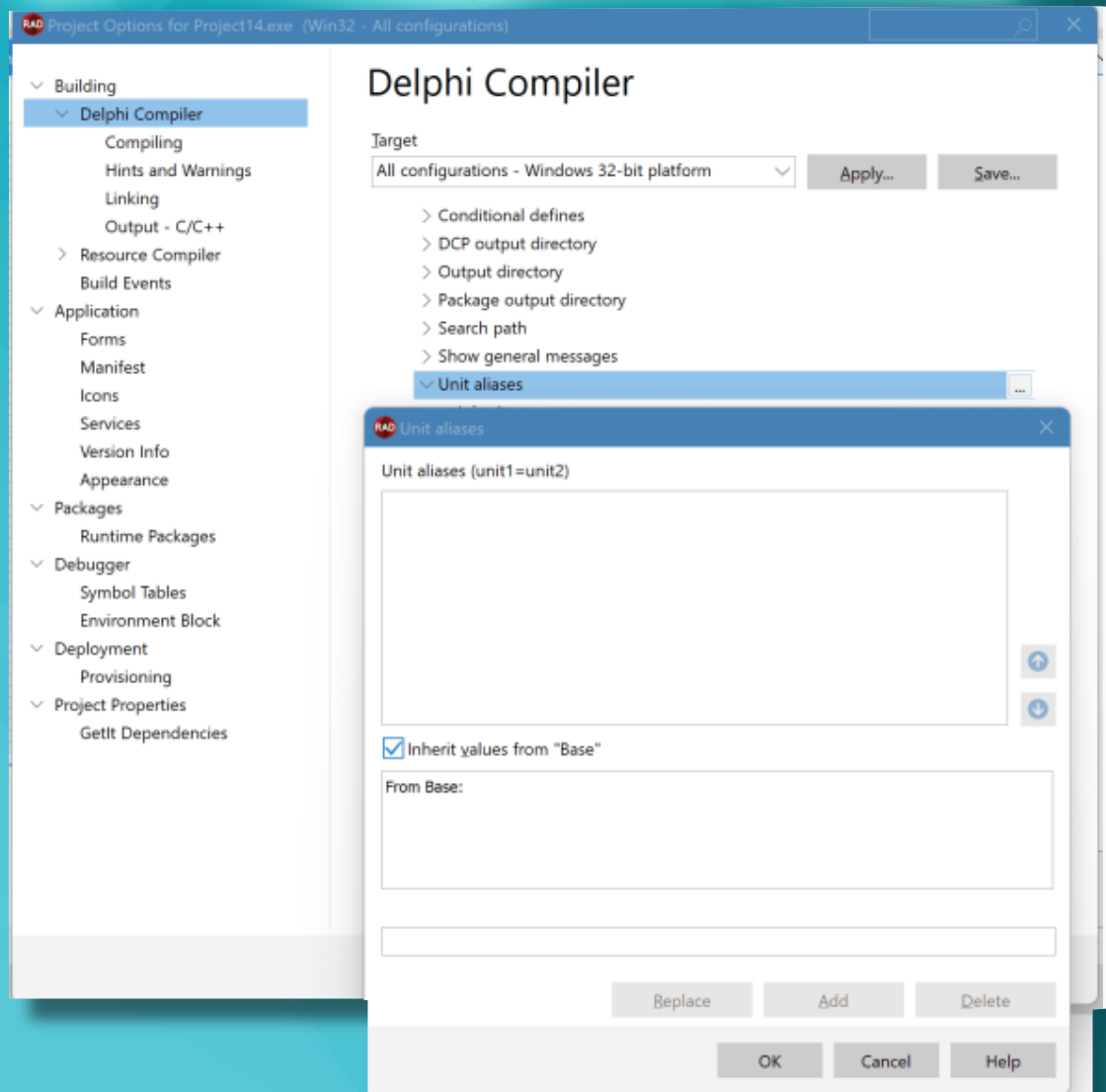
UNIT SCOPE NAMES AND UNQUALIFIED UNIT NAMES

Over ten years ago, Delphi introduced unit scope names, that is a prefix in front of all units that are part of the default libraries, like Vcl.Forms or System.SysUtils. This is covered in
`https://docwiki.embarcadero.com/RADStudio/Alexandria/en/Unit_Scope_Names`
If you have existing projects with short unit names, a fast migration path is to define unit scope names as part of the project options, as you can see here:
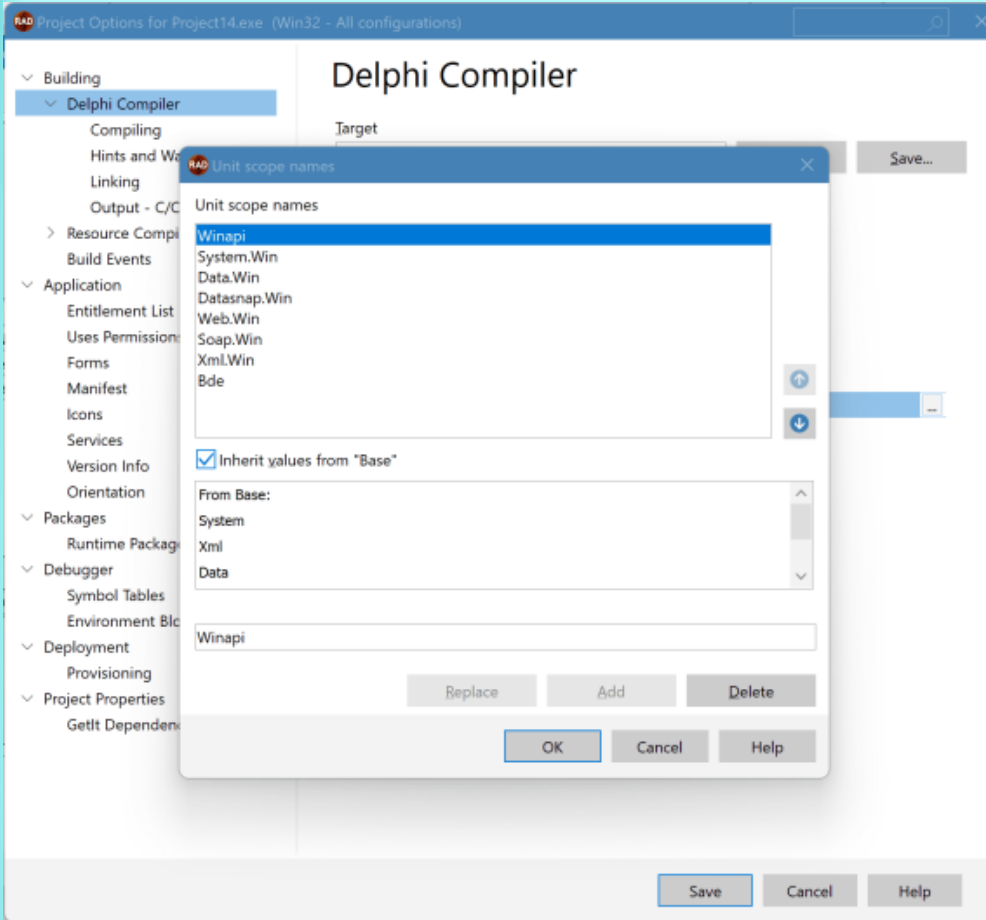
As you can see, the Delphi IDE still adds a large number of unit scope names by default even to new projects (*this is something we should consider changing, but that's a separate point*).
Now unqualified unit names in your code cause a file look up for each of the possible prefixes, in all of the folders of your search or library path.
Consider having 20 folders and 20 prefixes, that's potentially 400 file locations to look for.

No surprise we have reports like this one, titled '*Unqualified unit names cause massive amounts of file lookups*' and logged as `https://quality.embarcadero.com/browse/RSP-18130`.
The compiler has some caching logic to reduce the work, but again, this is only a feature to help code migration, developers should eventually clean up their `uses statement`, move to use fully qualified unit names for all units in Embarcadero libraries and remove all unit scope names configurations at the project settings level.
This is another code change that can have a significant benefit on the compiler.

REPEATED AND DUPLICATE UNIT NAMES

Speaking of unit names, there is another scenario that is known to cause trouble. In this case it is less of a performance issue, but a scenario that can cause failures, particularly in Code Insight and some of our IDE tooling.

The scenario is the following: you have a project group open in the IDE with multiple projects including different units (in different folders) with the same name. In other words, one project has MyUnit.pas in a folder and the other project has a different and unrelated unit also called MyUnit.pas in a different folder.

This is a scenario the compiler generally handles fine, but not always as you see in the report 'Compiler confuses `unit names` in project group' at
`https://quality.embarcadero.com/browse/RSP-39293`. (*You need to login as a user*)

We did specific work a couple of years ago so that the debugger won't get confused in similar scenarios. However, DelphiLSP can get confused by units with the same name.

Whenever possible, it would be much better to avoid using the same name for different units.

CIRCULAR UNIT REFERENCES

Getting back to scenarios that affect the compiler performance, one of the most critical ones we know of is the excessive use of `circular unit references` in the implementation section of the units.
As you know, the interface section of the units does not allow mutual or circular unit references.
However, in the implementation section of a unit, you can theoretically refer to all other units in the entire project.

When Delphi compiles a unit, it has to look at all used units and navigate the entire graph, making sure of source to check the cycles — to avoid an infinite loop while searching.
In the case of a complex graph with hundreds of units the cost of this navigation process is significant.

Recently our support team made an interesting experiment.
They created a code generator that creates an application in which every unit uses every other unit.
This is clearly an extreme use case — I really hope you don't have projects like this!
In such a scenario, adding one more unit almost doubles the compilation time.
In other words, there is an exponential degradation of the time to compile when one more unit is added that uses all other units in the project and is used by all other units (*it is exponential as every additional unit adds n*2 unit references where n is the number of existing units*).
z
Again, this is a scenario the compiler was already optimized for, moving searches and name lookup to used hash dictionaries and similar optimized data structures, and an area we are actively looking right now to improve to compiler. See for example this report, resolved in Delphi 10.4.2:
`https://quality.embarcadero.com/browse/RSP-28811`

However, the use of circular uses statements is also a use case scenario that acts against any good programming practice, as you end up with a spaghetti-graph of used units.
Reducing the dependency across units and across classes is a good practice.
Having some linear dependencies (*low level units used by high level units, UI units using data access units and not vice versa*) is acceptable, but having a complex graph is not healthy for the maintainability of your code over time.
In addition, it does put some significant stress on the compiler performance and memory usage.

We do have actual reports from Delphi developers with fairly large applications that reorganizing their uses statements to avoid circular unit references cut up to 90% of their compilation time, like ten minutes getting down to one minute and also helped Code Insight become responsive.

In case you want to address this issue, I recommend considering one of the tools on the next page (*53 - 5 of the article*), which can help you determine the current uses units graph of your application and detect cycles:

- GExpert Project Dependencies (*see image below*):
  `https://gexperts.org/tour/index.html?project_dependencies.html`
- Peganza Pascal Analyzer: `https://www.peganza.com/#PAL`
- Delphi Unit Dependency Scanner: `https://github.com/norgepaul/DUDS`
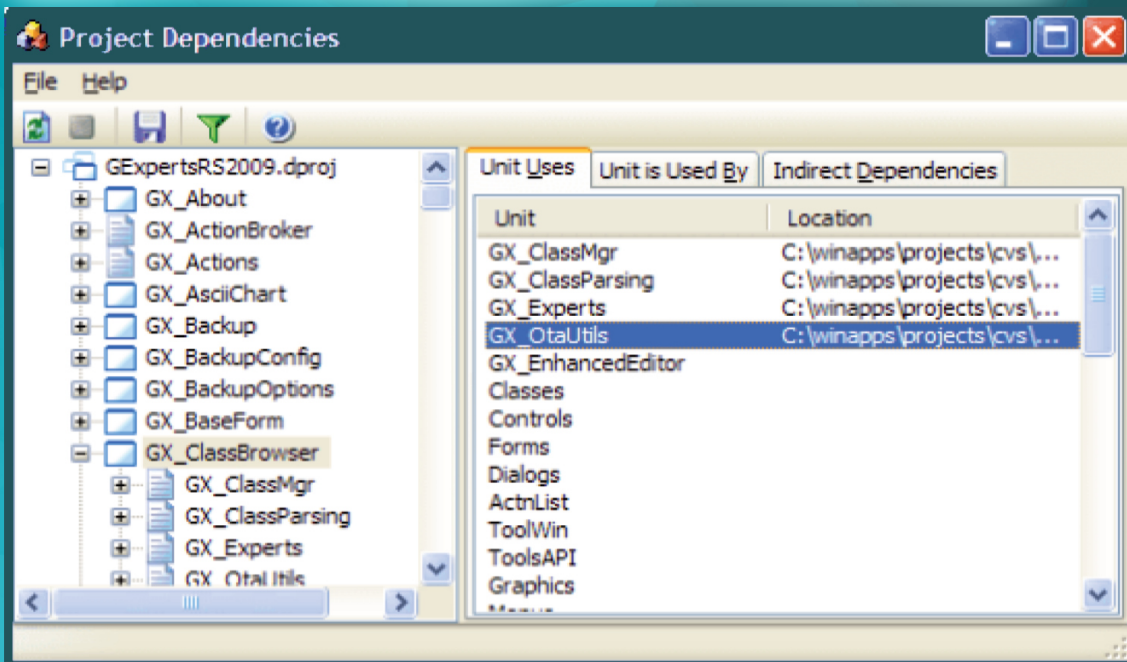
As an aside, notice that the compiler also dislikes when the same units are used in a different sequence (*as this affects the dependency graph of units using those units*).
In other words, if two units both use unit A and unit B, writing "uses A,B;" in both cases is better than having "uses B, A;" in the second one.
This is fairly minor, compared to avoiding circular unit references,
but still noticeable in a very large application.

Finally, you could also consider removing unnecessary uses statements, one tool offering specific help on this is Delphi Parser:

`https://delphiparser.com/product/`
`        remove-superfluous-uses-optimizer-wizard-evaluation-edition`

REPEATED GENERIC TYPES INSTANCES

The last suggestion we want to offer in terms of helping the Delphi compiler is a bit more contentious and something the compiler should be better at, after a few years. However, we know this can make a dramatic difference, in this case both in terms of compilation time and memory usages, avoiding the risk of out-of-memory compiler errors.

Delphi type system (*like the classic Turbo Pascal one*) is based on type declaration equivalence.
In other words, even if two types have the same name and the same structure, they are different if declared in different units. If you assign a variable of type `MyUnit1.TMyType` to a variable of type `MyUnit2.TMyType`, you get a compiler error, regardless of the fact the two types have the same identical structure.

The only exception to this rule is for generic types. If you declare a variable of type `TList` you are also creating an implicit type, `TList`, based on the generic type `TList`. You can have multiple type instances declared in different units and all matching the `TList` signature and the compiler considers them compatible and allows you to assign them one to the other.

What happens behind the scenes, though, is that the compiler generates a new temporary type of each of the occurrences in different units.
So you'll have, say, `TList` in `MyUnt1` (*with a temporary type name*) and `TList` in `MyUnit2` (with a different temporary name). For each of these types, the compiler creates instances for the generic type but also for each of the `generic methods` of the `generic type`.
With a large amount of complex `generic types` (*inherited and nested*), this can take a significant effort by the compiler.

The solution to this issue is not to stop using `generics`, but to create shared instances of generic types whenever possible. In other words, rather than using `TList` is dozens of different units, you can declare

```
type

  TListOfStrings = Tlist;
```

and use the `TListOfStrings` type all over the places where you'd use `TList`. In simple examples, the difference is barely noticeable. In a large application with significant use of generic types, it can make a very significant difference.

Again, we all know that the compiler could handle similar scenarios better, and we have been working to improve this use case — and we did improve it quite a bit over recent years. Still, you can help the compiler be faster and use less memory by refactoring some of your code in that direction — that is writing code like the generic type equivalence didn't exist.

CONCLUSION

Is There More? Possibly, Yes.
These are some of the scenarios we have found when examining the Delphi compiler performance that can be partially addressed by changes to the code.
I'd particularly recommend removing `unit aliases` and unqualified units, along with the detection and removal of `circular unit references`. If you are making significant use of generics, consider also the last suggestion above.

This is recommended in particular if you have a large application and you are experiencing slowness in the compiler: most of the Delphi developers are quite happy with the compiler performance and won't need to do anything in this respect.

I'm also quite certain I didn't list all of the scenarios that cause a compiler slow down that can be partially addressed by code or configuration changes. If you have additional suggestions, based on your experience, they are more than welcome!

PS. Have a nice holidays season and all the best for 2023.

BY DON WILBRINK

INTRODUCTION: WHAT IS A BUFDATSET

`TBufDataset` is a `TCustomBufDataset` descendant that implements a database-aware buffered dataset. `TBufDataset` is an in-memory dataset that can be used as a local stand-alone dataset, or it can be used as a local buffer for updates applied to a remote dataset. `TBufDataset` implements many features similar to those in `TClientDataset` in Delphi. However, it is not meant to be code or function compatible with `TClientDataset`. It is designed to provide management of the buffers used to access record data, and is used as the ancestor for classes like `TSQLQuery` and `TRESTBufDataset`.

As a `TDBDataset` descendent, it offers access to many of the database features supported in the FCL DB package. As a `TDataset` descendent, it also offers access to familiar navigation and data handling methods in the ancestor class. `TBufDataset` has features that allow local storage and retrieval of field definitions and record data. In addition, facilities are provided that allow local indexing of record data in the dataset. There is a comprehensive parser/expression evaluator available that allows complete support for Filters in the dataset.

`TBufDataset` sets the visibility for properties and methods defined in the ancestor class.

BUILDING THE APPLICATION

Let's set up a master detail relationship between two `BufDataSet`'s. (In Delphi it could be Client Datasets). We start with a new application via **New -> Application.** Then we put the necessary components on the form and save the project as platen.lpi and the form as main.pas. The form then looks like this:
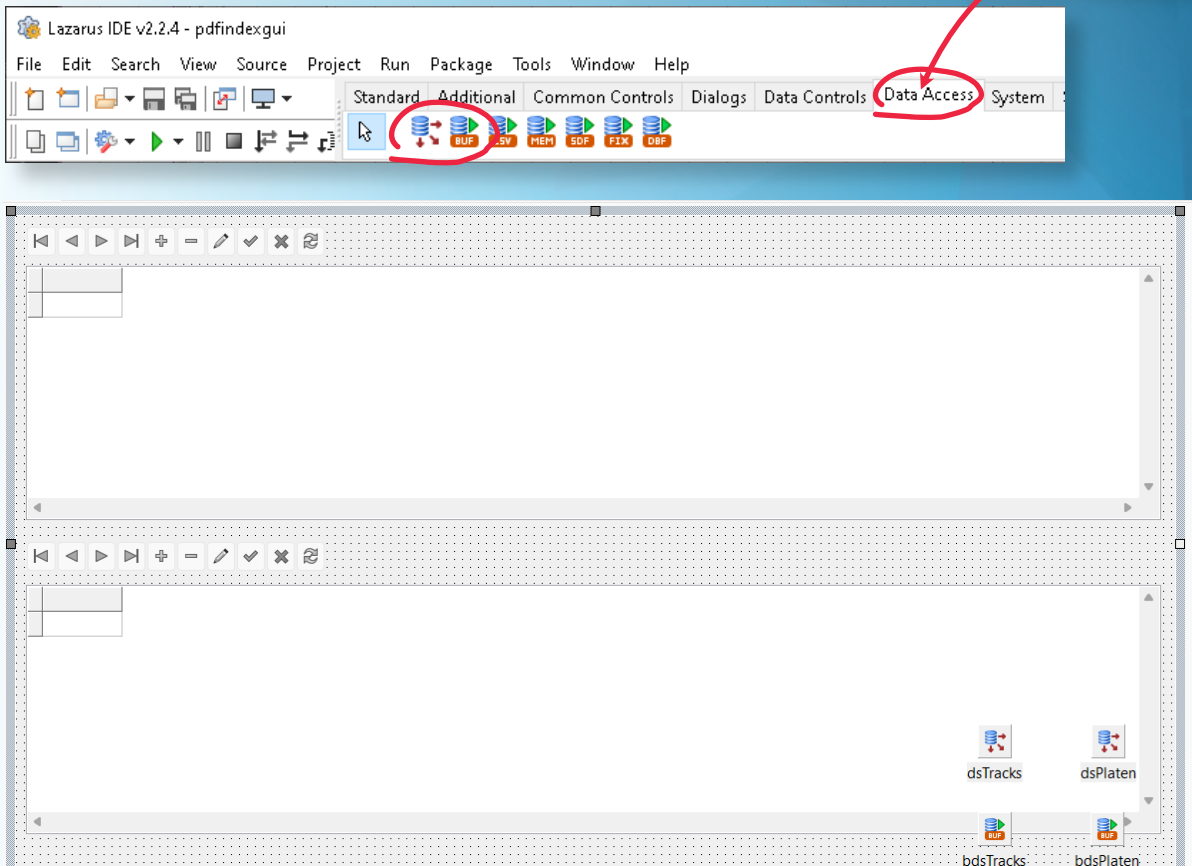
Figure 1: Design application main form.

Then adjust component properties:

Then adjust component properties:

Creating The form itself:
Name: mainForm
Caption: Platen overzicht
// Music Records overview

BufDataset1:

Name //of the dataset:
bdsPlaten //bdsRecords
FieldDefs: //add 3 Items through the fieldseditor

IndexFieldNames: ID
BufDataset1.FieldDefs.0:
//The numbering starts at 0
Name: ID
DataType: ftAutoInc
BufDataset1.FieldDefs.1:
Name: Album
DataType: ftString
Size: 30
BufDataset1.FieldDefs.2:

Name: Artiest
DataType: ftString
Size: 30
DataSource1:
Name: dsPlaten
Dataset: bdsPlaten

//————————–

BufDataset2: //Second Dataset
Name: bdsTracks
FieldDefs: //add 3 Items through the fieldseditor
IndexFieldNames: ID
BufDataset2.FieldDefs.0:
Name: ID
DataType: ftAutoInc
BufDataset2.FieldDefs.1:

Name: AlbumID
DataType: ftInteger
BufDataset1.FieldDefs.2:

Name: Tracknaam
DataType: ftString
Size: 50
DataSource2:

Name: dsTracks
Dataset: bdsTracks
DBNavigator1:

DataSource: dsPlaten
DBGrid1:

Anchors: akBottom, akLeft, akRight, akTop
DataSource: dsPlaten
DBNavigator2:

Anchors: akBottom, akLeft
DataSource: dsTracks
DBGrid2:

Anchors: akBottom, akLeft, akRight
DataSource: dsTracks:

CODE
Some more code for proper operation.
Under the Private section of Type heading, add TmainForm

```
private
 fn1: string;
 fn2: string;
```

First the listing of **OnCreate** the form to determine if the files already exist, otherwise create files:

```
procedure TmainForm.FormCreate(Sender: TObject);
begin
 fn1 := 'platen.bds';
 if FileExists(fn1) then
 begin
  bdsPlaten.LoadFromFile(fn1);
  bdsPlaten.Active := True;
  bdsPlaten.First;
 end
 else
 begin
  bdsPlaten.CreateDataset;
  bdsPlaten.Active := True;
  bdsPlaten.SaveToFile(fn1);
 end;
 fn2 := 'tracks.bds';
 if FileExists(fn2) then
 begin
  bdsTracks.LoadFromFile(fn2);
  bdsTracks.Active := True;
  bdsTracks.First;
 end
 else
 begin
  bdsTracks.CreateDataset;
  bdsTracks.Active := True;
  bdsTracks.SaveToFile(fn2);
 end;
end;
```

Then the **OnClose** event, write away files and close app:

```
procedure TmainForm.FormClose(Sender: TObject;
var CloseAction: TCloseAction);
begin
  bdsPlaten.SaveToFile(fn1);
  bdsTracks.SaveToFile(fn2);
  CloseAction := caFree;
end;
```

The **OnFilteredRecord** event of bdsTracks, ensures that only the records belonging to the selected record are shown in the other dataset:

```
procedure TmainForm.bdsTracksFilterRecord(DataSet: TDataSet;
var Accept: Boolean);
begin
  Accept := (bdsTracks.FieldByName('AlbumID').AsInteger =
    bdsPlaten.FieldByName('ID').AsInteger);
end;
```

The **BeforeInsert** event from bdsPlates, to clear the second grid when a new record is released:

```
procedure TmainForm.bdsPlatenBeforeInsert(DataSet:
TDataSet);
begin
  DBGrid2.Clear;
end;
```

The `OnAfterPost` events of both BufDatasets, to make sure the tracks are in the right order:

```pascal
procedure TmainForm.bdsTracksAfterPost(DataSet: TDataSet);
begin
   bdsTracks.First;
end;

procedure TmainForm.bdsPlatenAfterPost(DataSet: TDataSet);
begin
   bdsTracks.Filtered := False;
   bdsTracks.Filtered := True;
   bdsTracks.First;
end;
```

The `OnBeforePost` event of dbsTracks to automatically populate the AlbumID field:

```pascal
procedure TmainForm.bdsTracksBeforePost(DataSet: TDataSet);
begin
   bdsTracks.FieldByName('AlbumID').Value := bdsPlaten.FieldByName('ID').Value;
end;
```

And finally, DBNavigator1's `OnClick` event:

```pascal
procedure TmainForm.DBNavigator1Click(Sender: TObject; Button: TDBNavButtonType);
begin
   if ((Button = nbNext) or (Button = nbPrior) or (Button = nbFirst) or (Button = nbLast))
   then
     begin
       bdsTracks.Filtered := False;
       bdsTracks.Filtered := True;
       bdsTracks.First;
     end;
end;
```

The result looks like this:
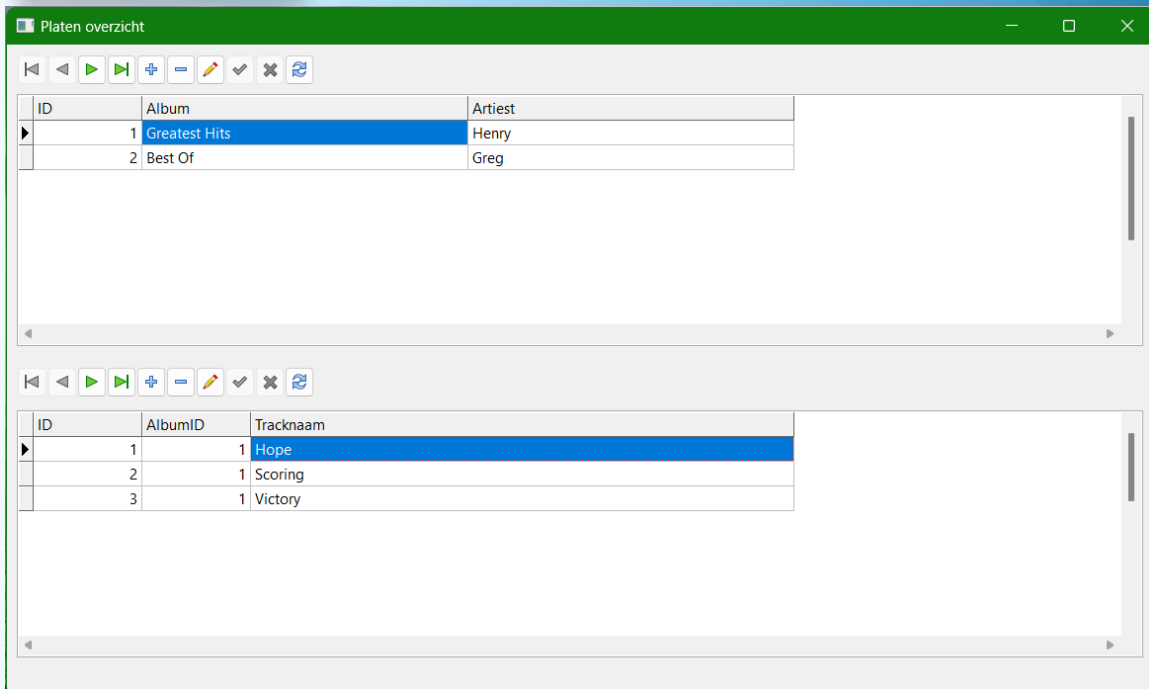
Figure 2: Application running with sample data.

Source code available for Subscribers....

BY MICHAEL VAN CANNEYT

ABSTRACT
Part of the **Lazarus** roadmap is a new initiative called Project **Fresnel**: a replacement for the **LCL** *(Lazarus Component Library)*. In this article we explain the goals of this new project.

1 HISTORY

The **LCL** - as its **Delphi** counterpart **VCL** - is old.
The foundations of this UI framework were laid 20 years ago,
when it was conceived as a free version of the **VCL** - the UI framework
used in **Delphi**.

Its design reflects the UI principles of that time, and more precisely,
the Windows UI toolkit of that time, the **Win32 API,** sported for the first time in
Windows 95. One could say therefore that **the LCL is a 27-year old technology.**
That one can take **20 year old code and still create a working application today** is
thanks to the great efforts to keep things backwards compatible.
But meanwhile, the world has moved on.

The internet has taken the world by storm, and the UI development in the browser is based on 2
standards: **HTML (the DOM)** and **CSS. CSS** offers a lot of possibilities in terms of UI
customisation, and the possibilities of **CSS** still grow.
The appearance of other devices created challenges for the UI developers: while in the 1990s,
the PC was the dominant platform, today this is no longer the case: mobile phones are ubiquitous, and
they come in all shapes and colors.

The old desktop PC is no longer the dominant form factor.
CSS has enabled web developers to handle these various form factors:
it is difficult to imagine a modern website which has not been designed with responsiveness in mind.
In response to these new platforms, **Firemonkey** was introduced in **Delphi**: an alternative to the **VCL** which
can be used on multiple platforms. It has styles, which can be used to cater for the different form factors, and
can be seen as a replacement for **CSS**. Firemonkey can handle different backends, for example **SKIA** can be used
as a backend or **OpenGL.** But at its core, Firemonkey still uses the positioning paradigm used in the **VCL,** and
the styles mechanism uses a format resembling the **DFM** format.

Bust of Augustin Fresnel
by David d'Angers (1854),
formerly at the lighthouse of
Hourtin, Gironde,
and now exhibited at the Musée
national de la Marine

2 INTRODUCING PROJECT FRESNEL

Instead of introducing a new format to update the
UI layer of Lazarus, it seems like a logical choice to reuse
a successful and standardized mechanism: CSS.

The paradigms of the VCL must be completely thrown overboard:
this means no more obligatory positioning, no more obligatory
`Top, Left, Width` and `Height` properties.

Instead, all matters UI and layout must be delegated to CSS properties.
In short: The aim of Project Fresnel is to make a set of controls
(or widgets) which are guided by CSS.
These controls will obey the following architectural principles:

- The widgets must be streamable:
  That means the widgets are descendents of `TComponent` and must use
  published properties. This requirement means a **form file** can be made,
  and the **Lazarus** IDE can be used to manipulate the widgets.

- The layout and looks are determined completely by CSS.
  This means every widget has a property that represents the CSS to be used.
  Additional properties for manipulation in an Object inspector can of course be introduced.

- Multiple drawing back-ends must be supported. Since **Lazarus** is multi- platform, it goes
  without saying that the various platforms will provide a specialized back-end,
  much as the custom-drawn widgetset or the FPGUI- based widgetset offers today.

- The widgets will be independent of the LCL: no dependency on the LCL must exist.

- The widgets must be able to co-exist together with an LCL framework:
  An application must be able to handle LCL forms together with CSS-based forms. In effect,
  the LCL will provide one of the possible backends for the Fresnel widgets.

The name Fresnel is chosen in honour of the French Physicist Augustin-Jean Fresnel who made important
contributions to the theory of light: what is a UI if not a clever play of light?
In the last part of this article is a short history about Fresnel.
The situation where the LCL is used to back up the Fresnel widgets (*as will be the case in the IDE*)
is depicted in *figure 1 on page 3*

3   ARCHITECTURAL CONSEQUENCES OF CSS

Ideally, a project made with the **Fresnel** widgets should be able to take an existing CSS framework such as
Bootstrap, Foundation or Tailwind CSS and apply it to a form.
That means we need to provide several things:

- a CSS parser and a mechanism to translate the CSS to native structures.

- a CSS resolver: this mechanism determines what CSS rules apply to a given widget.

- a drawing mechanism that knows all supported CSS properties.

- a **layouter**.
  CSS has several means to determine the layout: float, flex, grid etc,
  as determined by the display CSS property.

- Some widgets that mimic standard HTML elements:
  the CSS framework assume the existence of certain tag names (`div/span/p/label` *etc.*)
  so we need to provide widgets that react to these names. (*see the next page*).
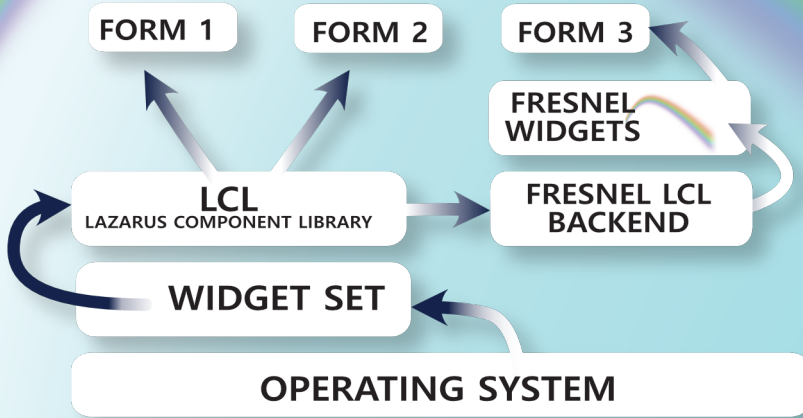
Figure 1: An application with mixed widget sets

The work on this has been started:
a CSS parser and resolver have been made. A drawing backend based on the LCL has been started, and a first layouter has been written.
A CSS rule can select elements based on tag names, element IDs, (pseudo-)attributes and tag hierarchies.
For object IDs, it seems natural to use the component name as the attribute.
This removes the need for the HTML equivalent of GetElementByID, since every element will be a component defined in the form.
A tag hierarchy maps naturally on a parent-child relation such as it exists in the VCL/LCL.
The mapping between tag names and attributes has been abstracted in an interface. It is up to the widget what tags and attributes it reports to the CSS engine.

4   CONSEQUENCES OF NOT USING THE LCL

The VCL or LCL handles lots of tasks. It is not only responsible for drawing the controls, but also handles messages coming from the OS: mouse messages, keyboard messages, drag and drop. The clipboard is also handled.
By requiring that the Fresnel Widgets are independent of the LCL, all these things must be re-implemented, although some may be reused from the LCL.
Since the Fresnel widgets do not need to obey the LCL and VCL heritage, this means we can redesign the messaging and event handling system. The LCL/VCL is limited to 1 event handler per message. Messages are sent in windows style, using 2 integers.
In old times, performance of the computers necessitated the use of such a limited system.
Today, this restriction is no longer needed (as evidenced by the browser).
The Fresnel Widgets will re-imagine this: use records (as in FPGui) or objects (as e.g. in Javascript)
to represent the messages, allow multiple event handlers for a given event, but still allow event handlers to be set in the Object Inspector.
Since the LCL will be used as one possible back-end, obviously a translation mechanism between LCL messages and the new Fresnel messages needs to be implemented.

5 CO-EXISTENCE WITH THE LCL

The IDE needs to be able to draw and manipulate
the Fresnel Widgets. Since the IDE is built using the LCL, it
follows that there must be a bridge between the LCL and the
Fresnel   Widgets:
The Fresnel Widgets should be able to use the LCL as a drawing
backend.
This coincides with an architectural requirement put forward in the
beginning: the LCL and Fresnel must be able to co-exist in a single
application.
The idea for this requirement is partially born out of the necessity to be able to
manipulate the widgets in the IDE, but more importantly to allow people to port
existing applications on a form-by-form basis to the new widget framework.

For many applications, the need to port all of the application at once would be simply a
too big hurdle to take.
Initially, the resulting application structure would be slightly more complicated:
Fresnel Widgets, on top of an LCL backend, with the backend based on LCL components,
where the LCL components are built on top of a widgetset. That situation was depicted in
figure 1 on page 3.

But the final goal is of course an application structure which is simpler: **Fresnel Widgets**, on
top of a backend based directly on the **OS**. This situation is depicted in *figure 2 on page 5*.

6   WHY NOT SIMPLY THE BROWSER ?

Since we're re-imagining widgets based on CSS, creating a new messaging model which has some
characteristics of Javascript DOM:
why not simply use the browser or some system like Electron (a packaged browser) or Chromium?

As a Pascal programmer, you can after all target these platforms with Pas2js or TMS Web Core.
The browser, however powerful, is still a limited environment with its own set of rules.
There is no doubt that the browser shines in UI development - which is why Project Fresnel uses CSS as its
mechanism - but native applications are still faster, and as a programmer you enjoy more freedom than in
a browser environment: access to the file system and other hardware.
The browsers also impose an asynchronous model on many operations, which makes it a little harder for
the beginning programmer.

What is more, porting to the browser is an all-or-nothing decision:
it is not possible to have part of the application in the browser, and part as native.
It is of course possible to rewrite an application form by form to a browser application,
but interaction between the native and browser part will be very limited.

The fact that Project Fresnel uses CSS also does not mean it is limited to the HTML elements:
it will still be perfectly possible to create your own widgets; but you will need to interrogate the CSS
mechanism to find out about borders, colors and such properties more.

The goal is also not to create a browser.
Although it would be possible to 'import' a HTML file and create a form based on the HTML tags,
creating a full-fledged browser is not the goal.
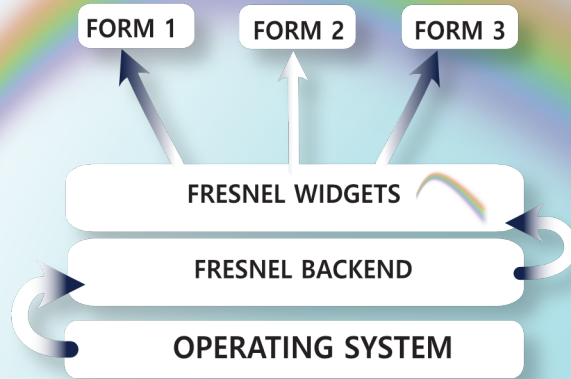
7   CONCLUSION AND THE ROAD AHEAD

Project Fresnel is a large project to rejuvenate the UI possibilities of Pascal:
there is a lot of work ahead. But the project has been put in motion, and the
first results have been obtained: a proof of concept using the LCL as a backend has
been created (*see figure 3 on page 6*). The code can be found in a Git repository on Gitlab:

https://gitlab.com/freepascal.org/lazarus/fresnel

It currently needs FPC trunk to compile due to the dependency on the CSS framework, but this may change in the
future so the release version of FPC can be used (and, who knows ? Delphi...).
The Free Pascal and Lazarus Foundation strongly believes in this project and will do everything to back up
development of this widgetset. But as said, it is a large project, and the success of the project will depend on the
uptake in, and the support of, the whole Pascal community.

Figure 3: The proof of concept

AUGUSTIN-JEAN FRESNEL
(10 May 1788 – 14 July 1827)
was a French civil engineer and physicist
whose research in optics led to the almost
unanimous acceptance of the wave theory of light,
excluding any remnant of Newton's corpuscular theory,
from the late 1830s until the end of the 19th century.
 (*In optics, the corpuscular theory of light states that light is made
up of small discrete particles called "corpuscles" (little particles)*)
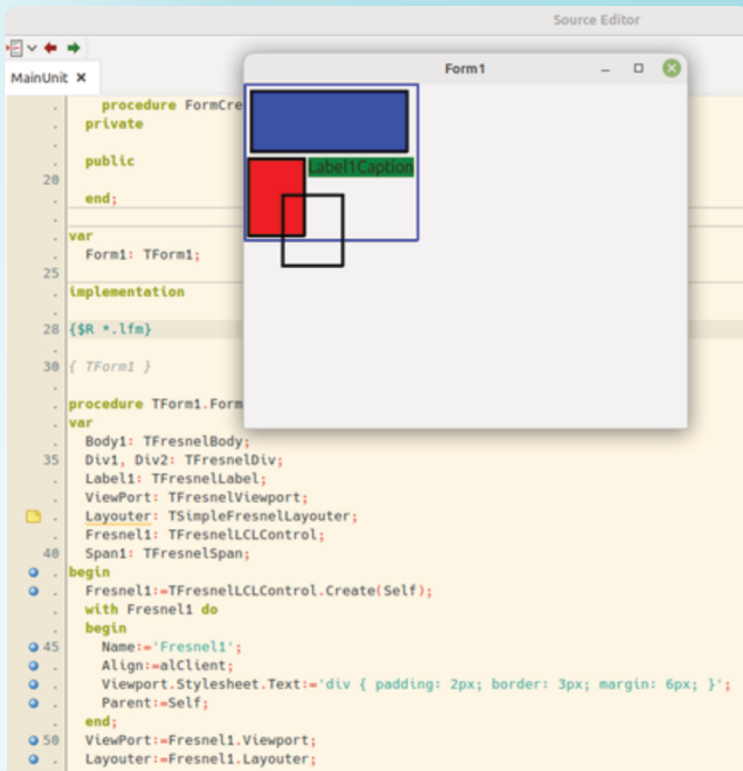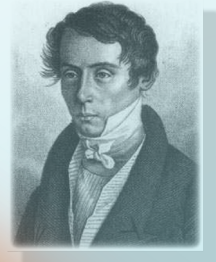which travel in a straight line with a finite velocity and possess
impetus. This was based on an alternate description of atomism of the
time period.)
He is perhaps better known for inventing the **catadioptric*** *(reflective/
refractive)* **Fresnel** lens and for pioneering the use of "stepped" lenses to extend
the visibility of lighthouses, saving countless lives at sea.

 **(A catadioptric optical system is one where refraction and reflection are
combined in an optical system, usually via lenses (dioptrics) and curved mirrors (catoptrics).
Catadioptric combinations are used in focusing systems such as searchlights, headlamps, early
lighthouse focusing systems, optical telescopes, microscopes, and telephoto lenses. Other optical systems
that use lenses and mirrors are also referred to as "catadioptric", such as surveillance catadioptric sensors.
)*

The simpler dioptric (*purely refractive*) stepped lens, first proposed by Count Buffon and independently
reinvented by Fresnel, is used in screen magnifiers and in condenser lenses for overhead projectors.
By expressing Huygens's principle of secondary waves and Young's principle of interference in quantitative terms,
and supposing that simple colors consist of sinusoidal waves, Fresnel gave the first satisfactory explanation of
diffraction by straight edges, including the first satisfactory wave-based explanation of rectilinear propagation. Part
of his argument was a proof that the addition of sinusoidal functions of the same frequency but different phases is
analogous to the addition of forces with different directions.

By further supposing that light waves are purely transverse, Fresnel explained the nature of polarisation, the
mechanism of chromatic polarisation, and the transmission and reflection coefficients at the interface
between two transparent isotropic media.
Then, by generalising the direction-speed-polarisation relation for calcite, he accounted for the directions
and polarisation's of the refracted rays in doubly-refractive crystals of the biaxial class *(those for which
Huygens's secondary wavefronts are not axisymmetric)*. The period between the first publication of his pure-
transverse-wave hypothesis, and the submission of his first correct solution to the biaxial problem, was less
than a year.
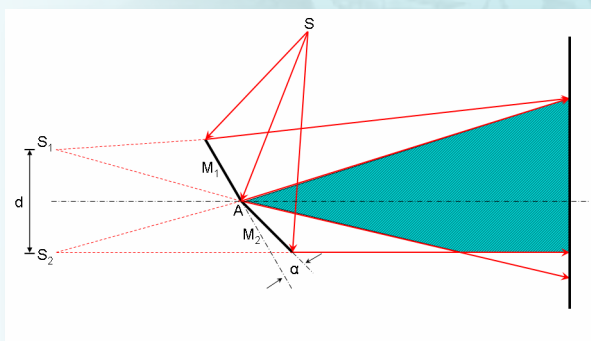
Fresnel's double mirror (1816).
The mirror segments M1 and M2 produce virtual images S1 and S2 of
the slit S.
In the shaded region, the beams from the two virtual images overlap
and interfere in the manner of Young (above).

Later, he coined the terms **linear polarization**, **circular polarization**, and **elliptical polarization**, explained how optical rotation could be understood as a difference in propagation speeds for the two directions of circular polarization, and (*by allowing the reflection coefficient to be complex*) accounted for the change in polarization due to total internal reflection, as exploited in the **Fresnel rhomb**.
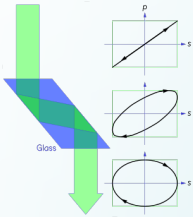
A Fresnel rhomb is an optical prism that introduces a 90° phase difference between two perpendicular components of polarization, by means of two total internal reflections. If the incident beam is linearly polarized at 45° to the plane of incidence and reflection, the emerging beam is circularly polarized, and vice versa. If the incident beam is linearly polarized at some other inclination, the emerging beam is elliptically polarized with one principal axis in the plane of reflection, and vice versa. `https://en.wikipedia.org/wiki/Fresnel_rhomb`

Defenders of the established corpuscular theory could not match his quantitative explanations of so many phenomena on so few assumptions.

Light or visible light is electromagnetic radiation that can be perceived by the human eye. Visible light is usually defined as having wavelengths in the range of 400–700 nanometres (nm), corresponding to frequencies of 750–420 terahertz, between the infrared (with longer wavelengths) and the ultraviolet (with shorter wavelengths).
In physics, the term "light" may refer more broadly to electromagnetic radiation of any wavelength, whether visible or not. In this sense, gamma rays, X-rays, microwaves and radio waves are also light. The primary properties of light are intensity, propagation direction, frequency or wavelength spectrum and polarization. Its speed in a vacuum, 299 792 458 meters a second (m/s), is one of the fundamental constants of nature. Like all types of electromagnetic radiation, visible light propagates by mass less elementary particles called photons that represents the quanta of electromagnetic field, and can be analyzed as both waves and particles. The study of light, known as optics, is an important research area in modern physics.

The main source of natural light on Earth is the Sun. Historically, another important source of light for humans has been fire, from ancient campfires to modern kerosene lamps. With the development of electric lights and power systems, electric lighting has effectively replaced firelight.

Fresnel had a lifelong battle with tuberculosis, to which he succumbed at the age of 39.
Although he did not become a public celebrity in his lifetime, he lived just long enough to receive due recognition from his peers, including (*on his deathbed*) the **Rumford Medal of the Royal Society of London**, and his name is ubiquitous in the modern terminology of optics and waves.

First-order rotating catadioptric Fresnel lens, dated 1870, displayed at the Musée national de la Marine, Paris. In this case the dioptric prisms (inside the bronze rings) and catadioptric prisms (outside) are arranged to concentrate the light from the central lamp into four revolving beams, seen by sailors as four flashes per revolution. The assembly stands 2.54 metres tall and weighs about 1.5 tonnes.

After the wave theory of light was subsumed by Maxwell's electromagnetic theory in the 1860s, some attention was diverted from the magnitude of Fresnel's contribution. In the period between Fresnel's unification of physical optics and Maxwell's wider unification, a contemporary authority, Humphrey Lloyd, described Fresnel's transverse-wave theory as "the noblest fabric which has ever adorned the domain of physical science, Newton's system of the universe alone excepted."

CROSS COMPILING WITH LAZARUS AND THE HELP OF FPC DE LUXE
BY ALFRED GLÄNZER

Starter   Expert
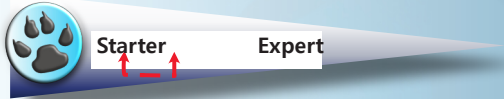
INTRODUCTION:

Users of Lazarus know this sentence by heart: "Write Once, Compile Anywhere".
But FPC and Lazarus also allow for "Write Once, Compile [Once/Many], Run Anywhere".
The latter is also known as cross-compiling. And it's an unique feature of FPC and Lazarus, that has little or no match among other development tools. The only problem involving cross-compiling is its steep learning curve in setting up your FPC and Lazarus to be able to cross-compile.

**Write Once,
Compile [Once/Many],
Run Anywhere**

There is a wiki:
`https://wiki.freepascal.org/Cross_compiling`
This wiki should give all needed info. But practice shows that it's still not easy to follow these instructions to get a working setup.

Here comes fpcupdeluxe to the rescue!
It's not known to many people, but the ancestor of fpcupdeluxe, being fpcup, was mainly built by its designer Reinier Olislagers (*also known as forum member BigChimp*) to ease the burden of cross-compiling. And that feature has been greatly extended by fpcupdeluxe.

To be able to use fpcupdeluxe as a setup-tool for cross-compiling, you MUST use it to install a native FPC and Lazarus first ! This has already been described in a previous article: *Blaise issue number 106 see article page*. So, please refer to this article to get info about the first use of fpcupdeluxe.
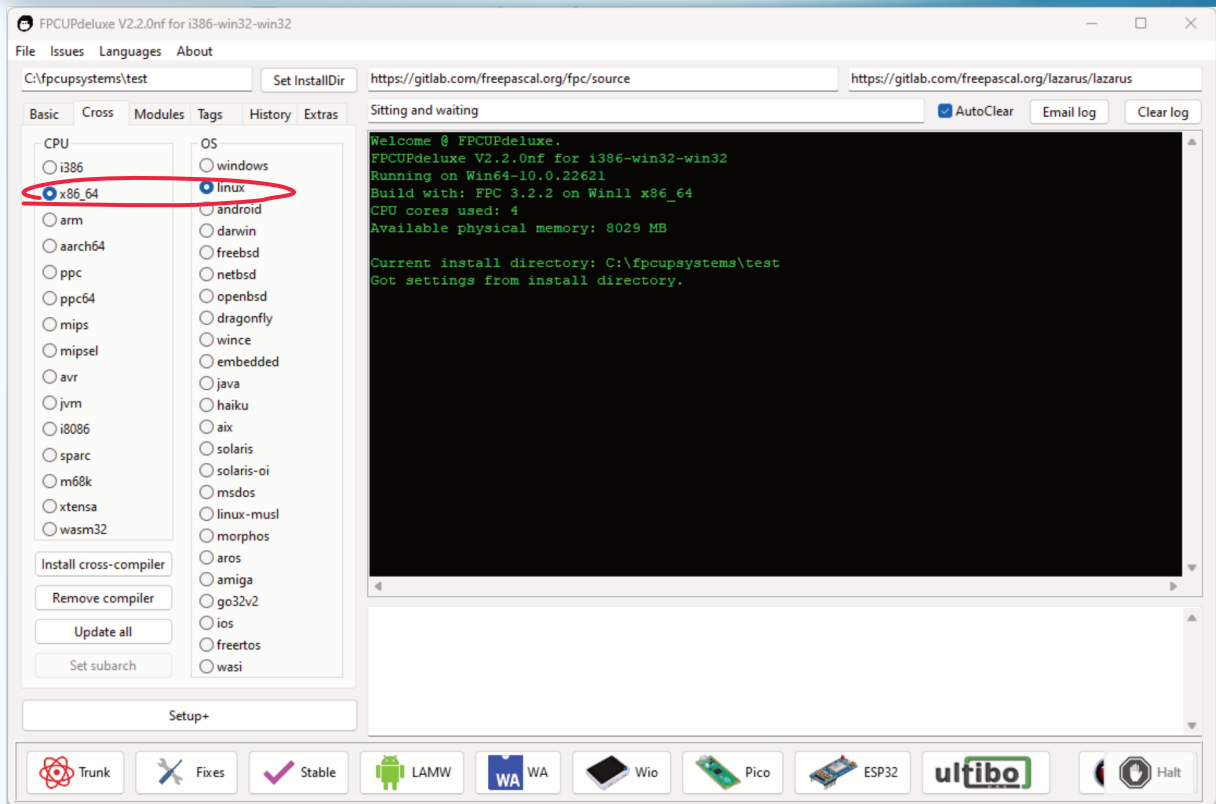
CROSS COMPILING WITH LAZARUS
AND THE HELP OF FPC DE LUXE

The picture on the previous page (1 of this article) shows the cross compilation screen of fpcupdeluxe. As you can see, it's the win32 version of fpcupdeluxe, which has previously been used to install FPC and Lazarus for win32.
The CPU and OS selection is targeting Linux for 64 bit Intel (linux-x86_64). Installing the selected cross-compiler is now as easy as pressing the appropriate button.

However !!!
In most cases it will fail after performing a clean-up of the sources and a check of the system requirements. And that is to be expected: to be able to cross-compile, binary tools and system libraries might be needed. Binary tools like an assembler and a linker for the target system. Libraries like the ones that can be found on the target system itself.
And again here comes fpcupdeluxe to the rescue !

CROSS COMPILING WITH LAZARUS
AND THE HELP OF FPC DE LUXE
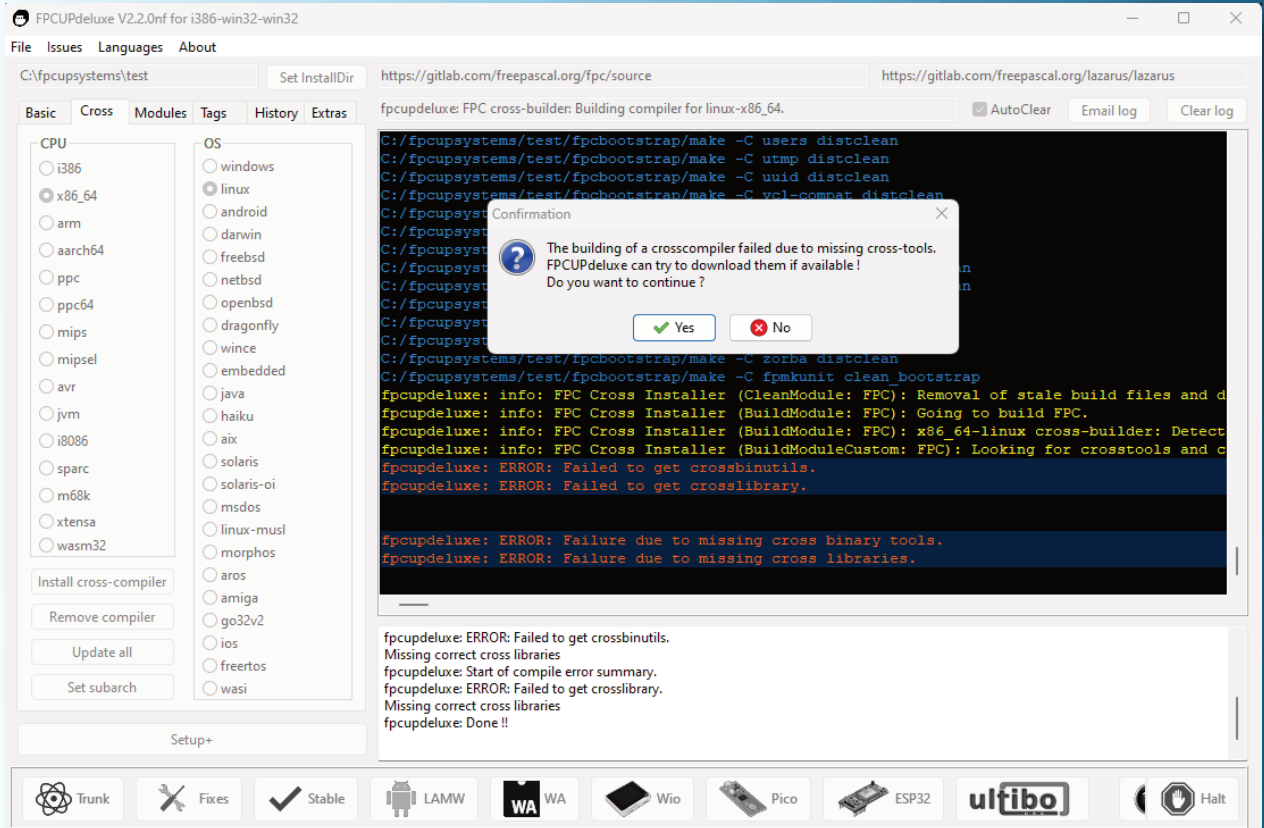
Again, to be able to cross-compile, suitable tools and libraries
might be needed.
fpcupdeluxe can download and install them at the right location,
where they can be found by and used by the FPC cross-compiler.

So, click yes to allow the download and install, and the process of
building and installing a cross-compiler will continue.
If everything is downloading and building as expected, the process
will finish in due time and the cross-compiler is ready for use.
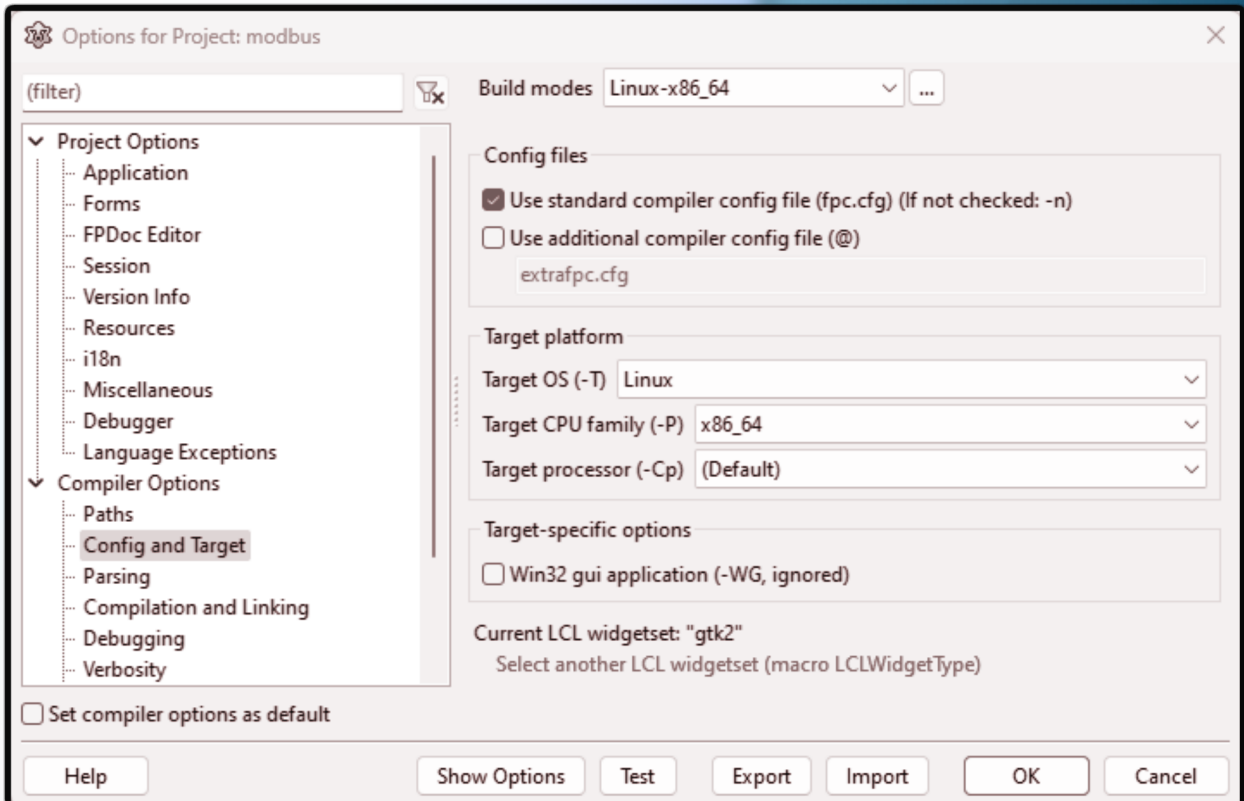
To be able to use the cross-compiler, select the appropriate CPU and OS as a
target in Lazarus. You should now be able to cross-compile your own program
from your Windows system towards your new target: Linux 64bit. It will produce a
binary that can be copied to and run on the target system !
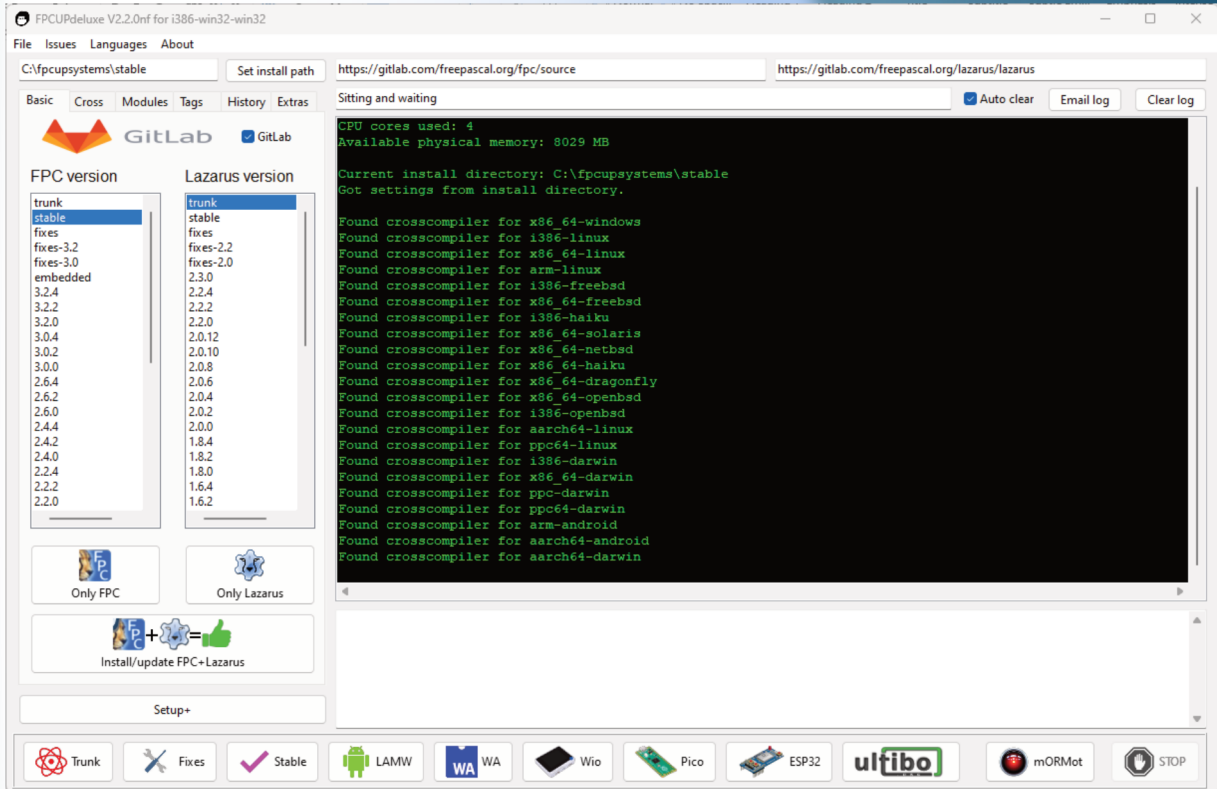
The development and deployment of fpcupdeluxe itself would have been impossible without cross-compiling.
The screenshot shows a typical install for the development and building of fpcupdeluxe.
Many cross-compilers are used to target all systems that fpcupdeluxe is suitable for.
And now back to the second claim of this article: **Write Once, Compile [Once/Many], Run Anywhere.**

Available project build modes:

Compile the following modes

- ☐ Default
- ☐ DefaultQT
- ☐ Debug
- ☑ win32
- ☐ win32curl
- ☑ win64
- ☐ win64arm64
- ☑ linuxi386
- ☑ linux64
- ☑ linuxaarch64
- ☑ linuxaarch64QT5
- ☑ linuxaarch64QT5zip
- ☑ linuxarm
- ☑ linuxarmQT5
- ☑ linuxarmQT5zip
- ☑ linuxarmv6
- ☑ linuxi386QT5
- ☑ linuxx64QT5
- ☑ linuxx64QT5zip
- ☑ linuxppc64
- ☑ freebsdi386
- ☑ freebsdx64
- ☑ netbsdx64
- ☑ solarisx64
- ☑ dragonflyx64
- ☐ linux64gtk3
- ☐ solarisxi386
- ☑ openbsdx64

OK   **Cancel**
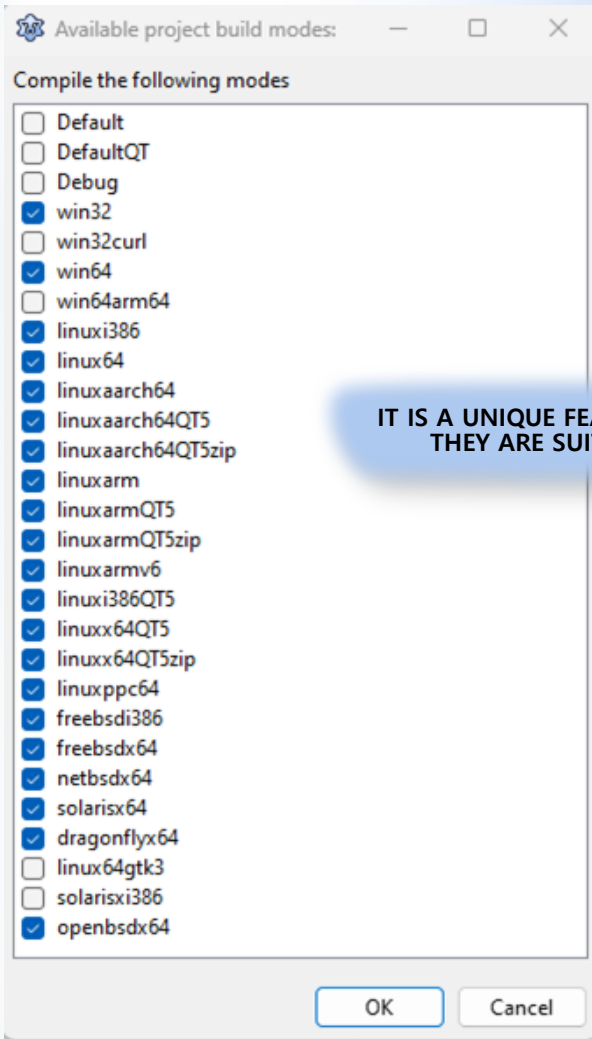
**IT IS A UNIQUE FEATURE OF FPC AND LAZARUS THAT
THEY ARE SUITABLE FOR SO MANY SYSTEMS**

Lazarus can compile for many targets at once !
Its menu item is called:
Run -> Compile many Modes.
Again, the screenshot above shows a typical fpcupdeluxe setup.
Pressing Ok will compile for all selected targets at once. A real (*and unique*) implementation of the second claim.

This article is based on a Windows install.
The same steps are valid for all other systems.
You can also use Linux-version of fpcupdeluxe to install FPC and Lazarus on Linux and cross-compile towards Window.

You can cross-compile towards the Raspberry Pi.
Towards Android. Towards OSX.
Towards embedded systems like the ESP32.
Towards exotics like Haiku or BSD-Linux.
Towards web-assembly.

It is a unique feature of FPC and Lazarus that they are suitable for so many systems. And make these targets easy to reach by cross-compiling.

More questions :
`https://forum.lazarus.freepascal.org/`
`index.php/topic,34645.0.html`
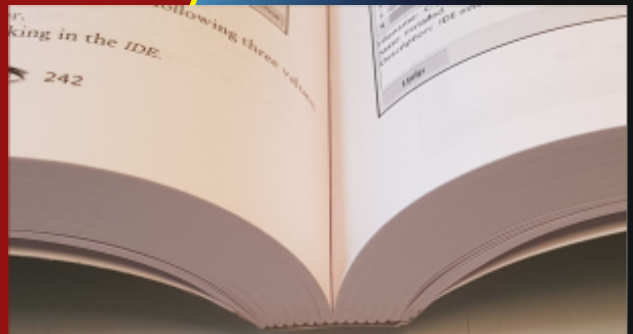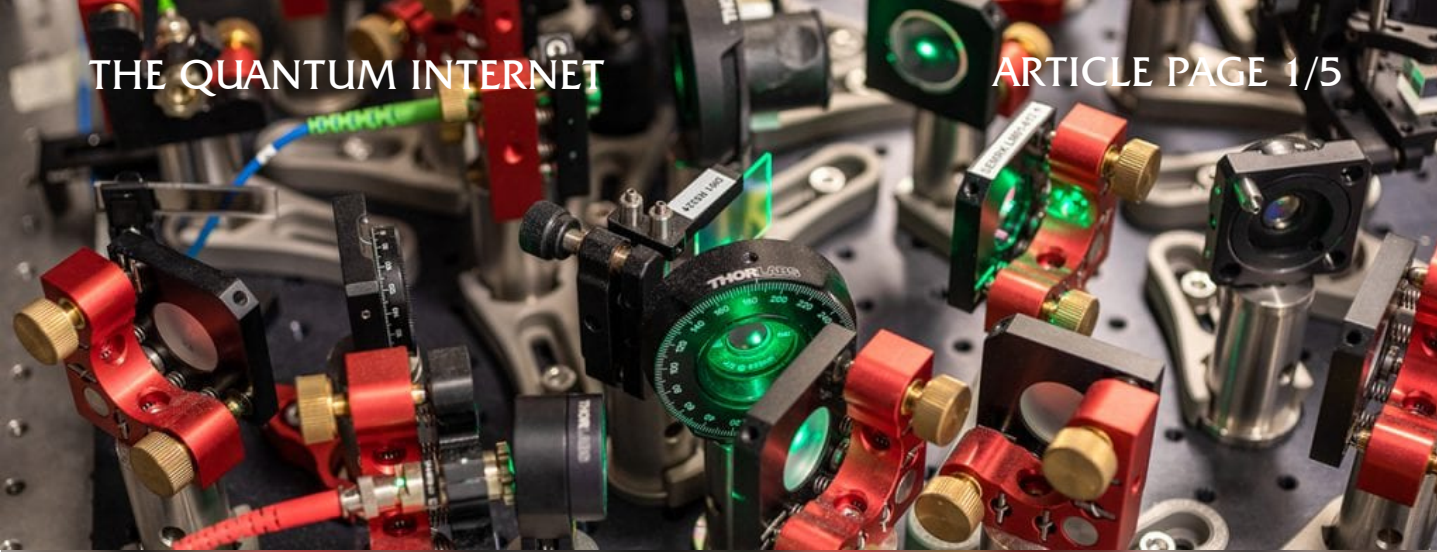
A team of researchers from the Dutch Enterprise QuTech (*A springoff of Delft University of Technology and TNO*)
stated realisation of the first **multi-node quantum network**, connecting three quantum processors.
In addition, they achieved a proof-of-principle demonstration of key quantum network protocols.
Their findings mark an important milestone towards the future quantum internet and have now been published in
Science.

THE QUANTUM INTERNET

The power of the Internet allows any two computers on Earth to be connected with each other,
enabling applications undreamt of at the time of its creation decades ago.
Today, researchers in many labs around the world are working towards first versions of a quantum internet (*a
network that can connect any two quantum devices, such as quantum computers or sensors, over large distances*).
Whereas today's Internet distributes information in bits (*that can be either 0 or 1*), a future quantum internet will
make use of quantum bits that can be 0 and 1 at the same time.
A quantum internet will open up a range of novel applications, from unhackable communication and cloud
computing with complete user privacy to high-precision time-keeping,' according to Matteo Pompili, PhD student
and a member of the research team. And like with the Internet 40 years ago (*The Arpa - ne*t), there are probably
many applications to be developed we never dreamt off.

TOWARDS UBIQUITOUS CONNECTIVITY

The first steps towards a quantum internet were taken in the past decade by linking two quantum devices that
shared a direct physical link.
However, being able to pass on quantum information through intermediate nodes (*analogous to routers in the
classical internet*) is essential for creating a scalable quantum network.
In addition, many promising quantum internet applications rely on entangled quantum bits,
to be distributed between multiple nodes.
Entanglement is a phenomenon observed at the quantum scale, fundamentally connecting particles at small and
even at large distances.
It provides quantum computers their enormous computational power and it is the fundamental resource for sharing
quantum information over the future quantum internet. By realizing their quantum network in the lab, a team of
researchers at QuTech is the first to have connected two quantum processors through an intermediate node and to
have established shared entanglement between multiple stand-alone quantum processors.

OPERATING THE QUANTUM NETWORK

The rudimentary quantum network consists of three quantum nodes, at some distance within the same building.
To make these nodes operate as a true network, the researchers had to invent a novel architecture that enables
scaling beyond a single link.
The middle node (called Bob) has a physical connection to both outer nodes (called Alice and Charlie),
allowing entanglement links with each of these nodes to be established.
Bob is equipped with an additional quantum bit that can be used as memory, allowing a previously generated
quantum link to be stored while a new link is being established.
After establishing the quantum links Alice–Bob and Bob–Charlie, a set of quantum operations at Bob converts these
links into a quantum link Alice-Charlie. Alternatively, by performing a different set of quantum operations at Bob,
entanglement between all three nodes is established.

Researchers work on one of the quantum
network nodes, where mirrors and filters
guide the laser beams to the diamond chip.

READY FOR SUBSEQUENT USE
An important feature of the network is that it announces
the successful completion of these (*intrinsically probabilistic*)
protocols with a "flag" signal.
Such heralding is crucial for scalability, as in a future
quantum internet many of such protocols will need to be
concatenated. 'Once established, we were able to preserve
the resulting entangled states, protecting them from noise,'
says Sophie Hermans, another member of the team. 'It means that, in principle, we can use these states for
quantum key distribution, a quantum computation or any other subsequent quantum protocol.'
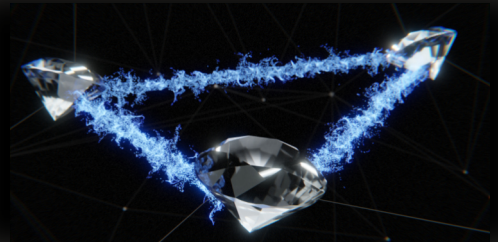
Quantum Internet Demonstrator
This first entanglement-based quantum network provides the researchers with a unique testbed for developing and
testing quantum internet hardware, software and protocols. 'The future quantum internet will consist of countless
quantum devices and intermediate nodes,' says Ronald Hanson, who led the research team. 'Colleagues at QuTech
are already looking into future compatibility with existing data infrastructures.' In due time, the current proof-of-
principle approach will be tested outside the lab on existing telecom fibre – on QuTech's Quantum Internet
Demonstrator, of which the first metropolitan link is scheduled to be completed in 2022.

Higher-level layers
In the lab, the researchers will focus on adding more quantum bits to their three-node network and on adding
higher level software and hardware layers. Pompili: 'Once all the high-level control and interface layers for running
the network have been developed, anybody will be able to write and run a network application without needing to
understand how lasers and cryostats work. That is the end goal.'
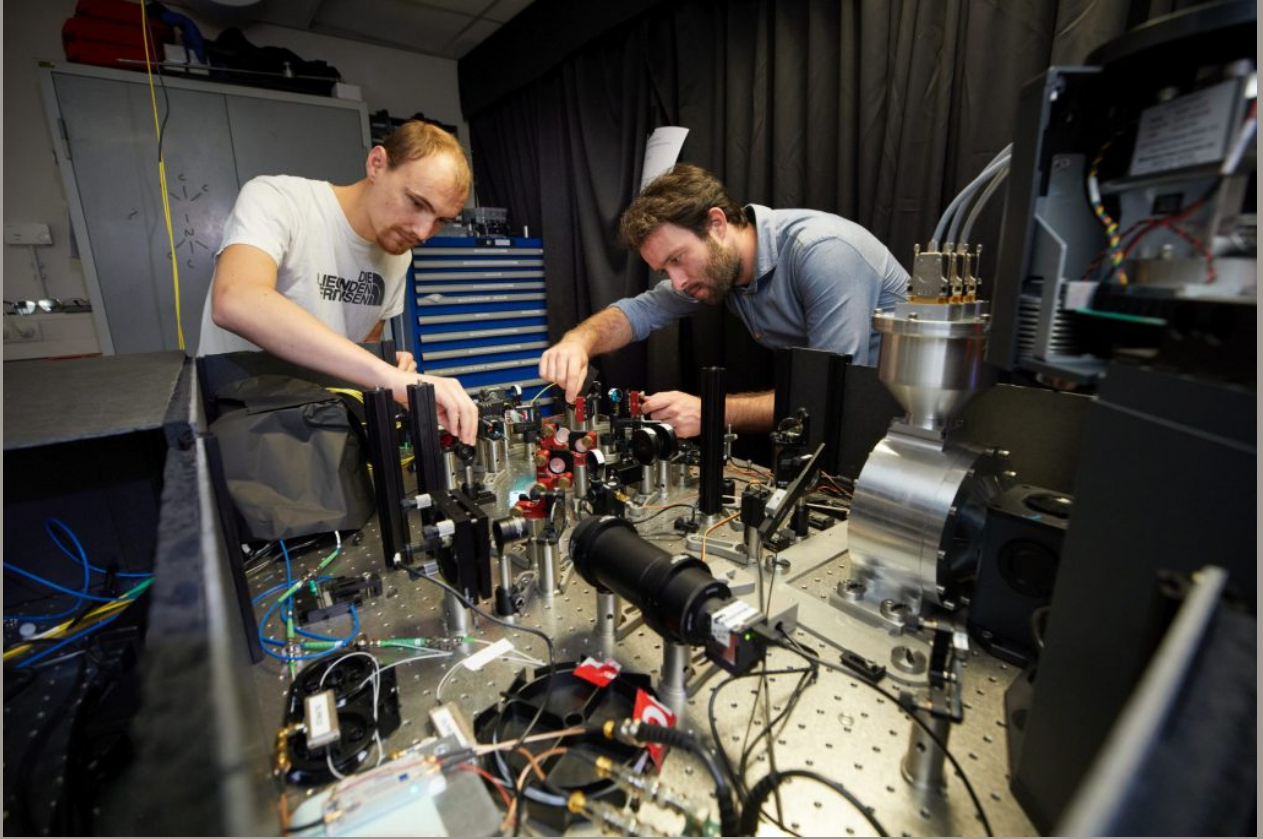
Artist's impression of the three-node quantum network. Credit: Matteo Pompili for QuTech
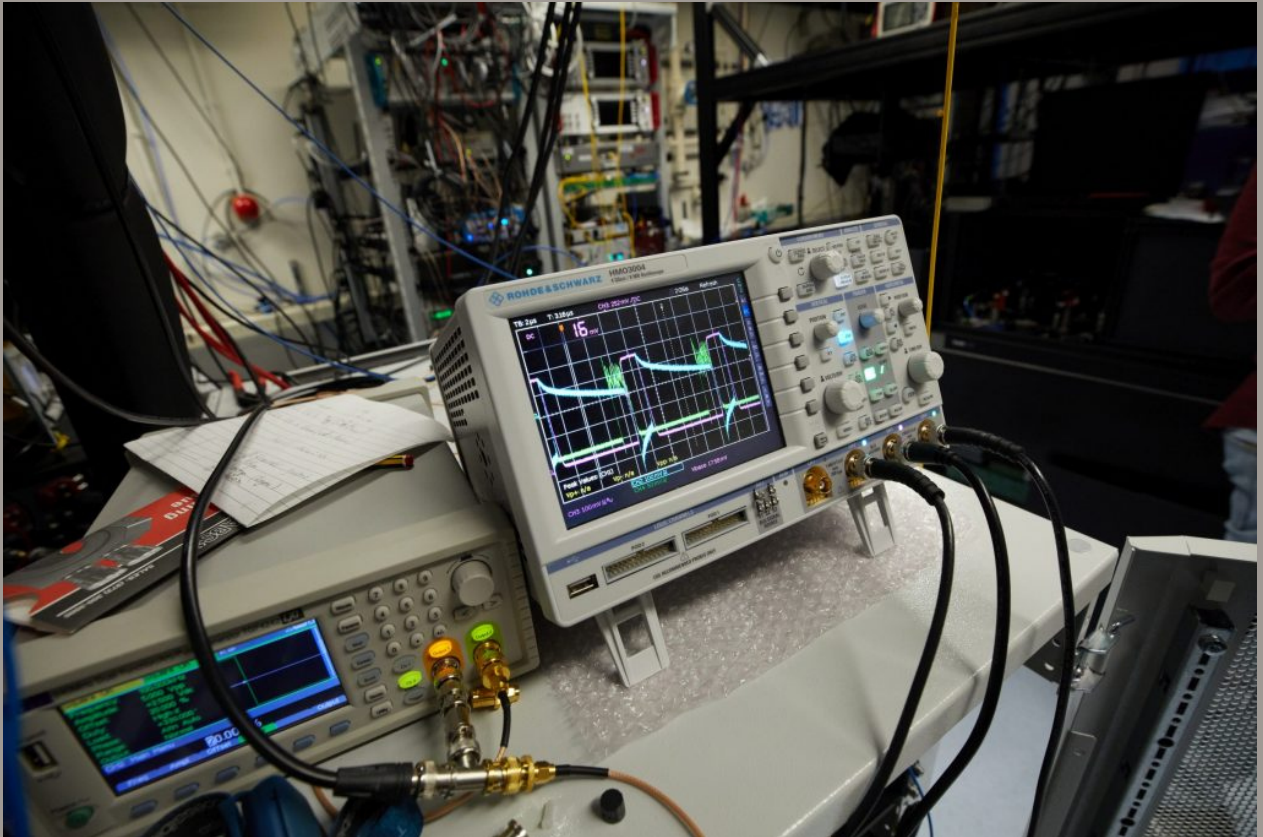
LAZARUS HANDBOOK PDF

- English
- Black & white edition
- 934 Pages
- Signed with
  your own name
- Electronic indexed
- Index of 240 pages
- List of chapters is
  completely clickable
  for moving to the page
- Including 40 Example
  projects and
  extra programs

LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS

ELECTRONIC (PDF)

Michaël van Canneyt/Martin Friebe/Mattias Gärtner
Inoussa Ouedraogo/Detlef Overbeek/
Howard Page Clark/Werner Pamler

BlaisePascalMagazine
PDF viewer included

INSTALLING FASTREPORT PACKAGES IN LAZARUS FOR LINUX / WINDOWS
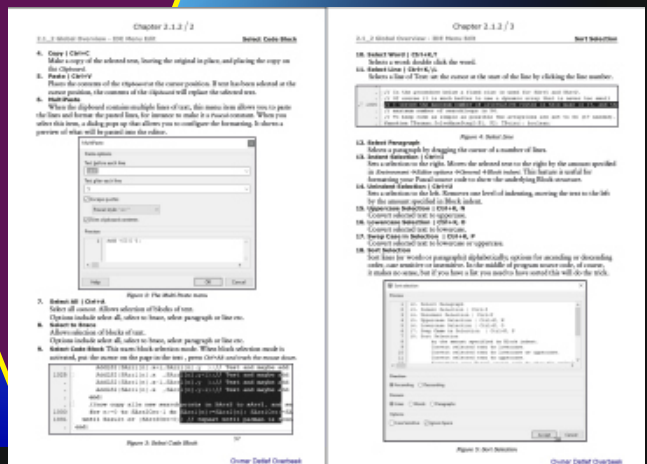
Starter　　　**Expert**

INTRODUCTION

FastReport has been going on creating its own version
for Lazarus and it's a very good idea to create a special version for
Windows as well as Linux. Mac isn't yet available. As soon as that happens
we will let you know.
To make it easy for you as a reader we will give you some instructions that
helps you finding and installing the Files.

As you of course know all OS's are available on Lazarus,
but for this instruction it means the components are only capable for
Windows and / or Linux.
For this reason I will use the normal installation:
Under Windows (*Which I show here*) the path to all files is as follows:

```
c:\Program Files (x86)\FastReport VCL Enterprise\Source\
```

under Linux it is of course very different and depends on what version of
Linux you have. So we will give a Linux installation in the next issue.
As told before you need a license from FastReport which could be
professional or a Trial. This is easy to download from

```
https://www.fast-report.com/en/download/public_files/267/
```
It delivers an exe file of the trial version

If you use the trial you will get a little screen in your page:

Trial 5 page limit with a note in the corner that the edition is Trial

This **Trial** is only meant for try outs and is used to test components before
purchase
(*previously we only offered compiled demo projects*).

"FastReport for Lazarus" is not included in the Standard license.
The Enterprise and Ultimate licenses also come with client/server
components.

Installing **FastReport** in Lazarus for
Linux and Windows
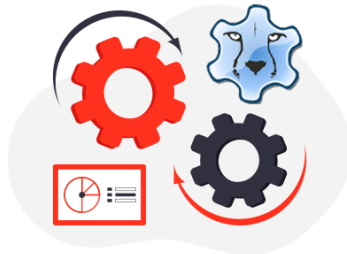
So, we have already installed Lazarus, now let's move on to installing the
FastReport VCL report generator packages in Lazarus.

To do this, we must first download and unpack the licensed version of the
product from the official website, Professional and higher versions come as an
`.exe` installer, Trial and Academic — as `zip` archives.
Unlike Embarcadero Delphi, RAD Studio, and C++ Builder, where it is enough
to "*simply install the compiled packages of components*", in Lazarus they must be
compiled, with the exception of **Trial** and **Academic**, which are pre-compiled
with closed (cut) source code.
To install packages, click **Package -> Open Package File *.lpk**,
select the package in the file manager and you will see the following window:
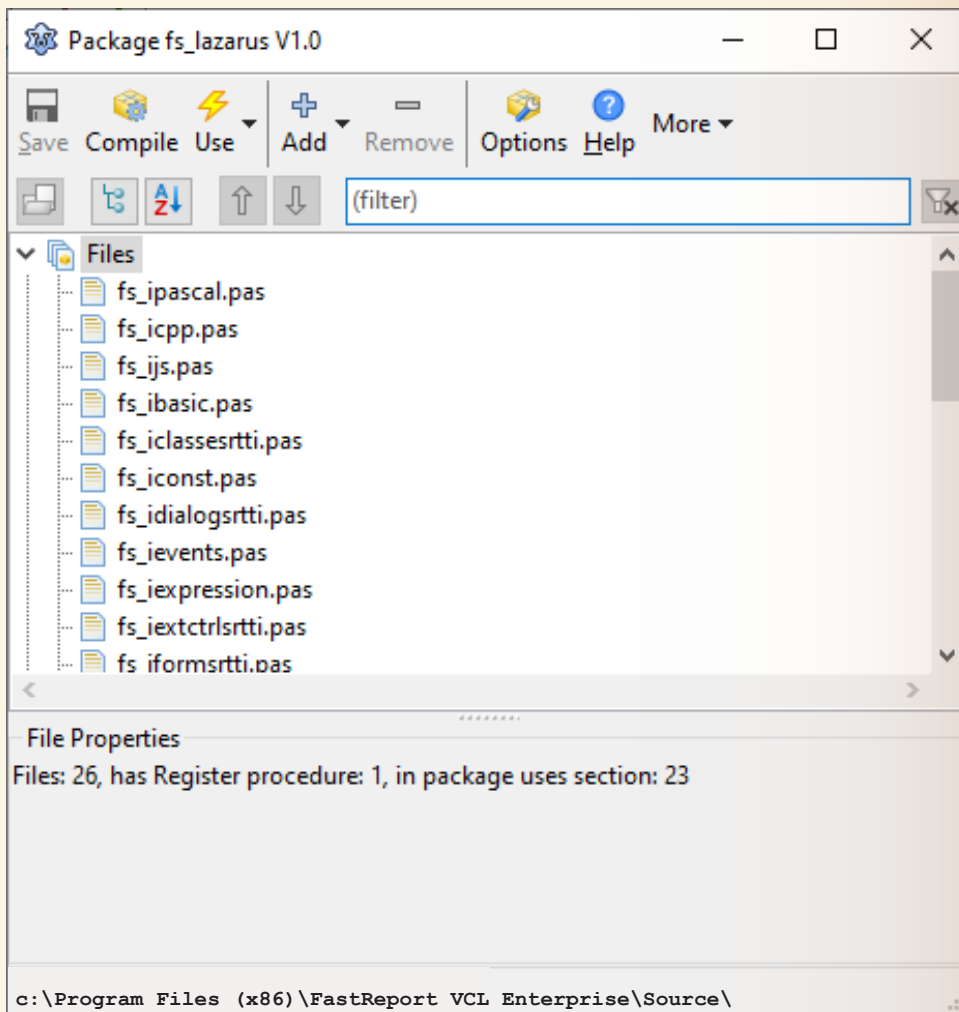
For Professional and above, click Compile, wait for the compilation to finish and click Use.
For Academic and Trial, click Use right away. After installing each package, Lazarus will reboot.

Let's move on to the order of installing FastReport packages
❶   `fast-script\Source\fs_lazarus.lpk`
       – library for executing scripts;
❷. `fast-report\Source\frN_lazarus.lpk`
       – package with all main components;
❸   In any order:
-    `fast-report\Source\ExportPack\frxeN_lazarus.lpk`
    package with exports;
-    `fast-report\Source\lazchart\frxchartlazarus.lpk`
    package for charts (diagrams);
-    `fast-report\Source\lazdbf\frxlazdbf.lpk`
    a package for working with a BDF format database;
-    `fast-report\Source\sqlite\frxlazsqlite.lpk`
    a package for working with SqLite DBMS;
-    `fast-report\Source\PDFView\frxPDFlazarus.lpk`
    a package for displaying PDF documents (Windows only);
-    `fast-report\Source\lazrich\frxrichlazarus.lpk`
    a package for displaying Rich documents
    (*recommended only for Windows due to basic package restrictions*);
❹ - `fast-report\Source\ClientServer\frCS_lazarus.lpk`
    a package with client-server components, you can read more about them here;

As mentioned in the edition comparison, the `frxRich` package is only available for **Professional**
and above, and the client-server components are only available in **Enterprise** and **Ultimate**.
Prior to version 2.0.0, there was a very common compilation and/or installation error, at the time
of writing this document, the latest version is 2.2.2 and this error has not yet been completely
removed in Lazarus, but its probability has been significantly reduced in Windows.
If one of the packages does not compile/install, you will have to go down and recompile/reinstall the package
dependencies.
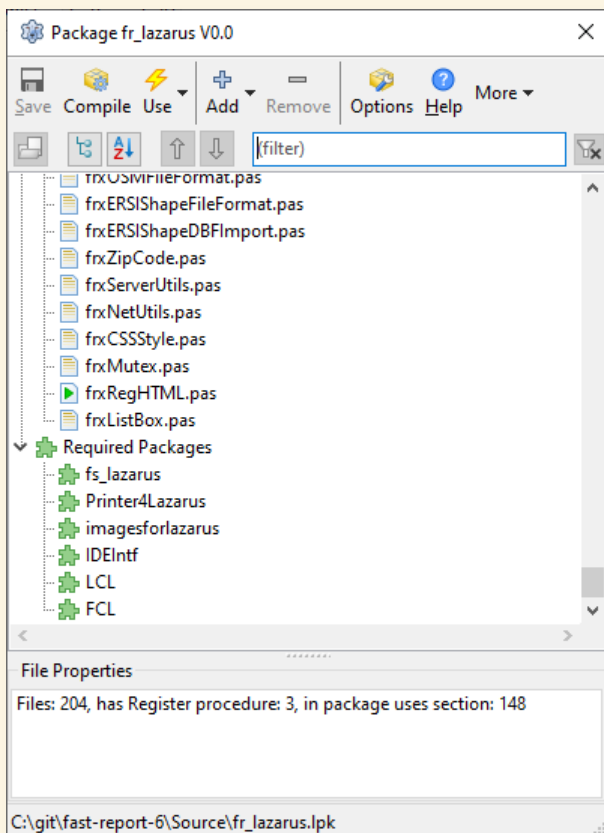
To do this, double-click on it and recompile, and then reinstall.

After successful installation of all the packages, click Project -> Open Project and open the
`fast-report\LDemo\FRDemo.lpi` project and try to run it, then click the Design button

If you get this negative height error on Linux:
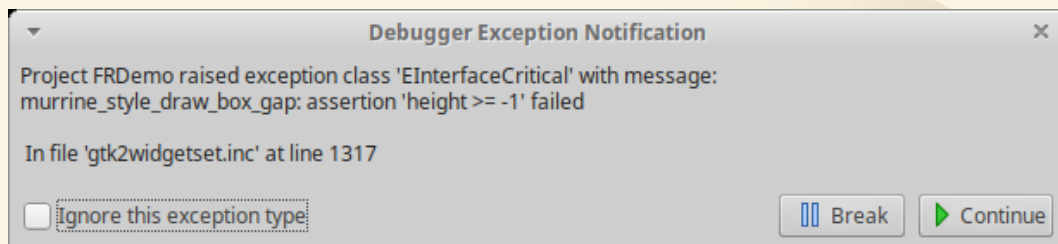
Don't worry. We support both **GTK** and **QT** interfaces (*but keep in mind that development is mainly done on GTK*). So, you can find this bug in some (rather rare) **GTK** interfaces. Just run the application without debugging, or check "*Ignore this type of exceptions*".

*Or change the graphical shell. For example, in our team, many people work under the **KDE** Plasma **GTK** shell, where this error does not exist.

The last nuance that you need to know when creating your projects is that our designer uses multithreading, which is disabled by default in Linux Lazarus. To enable it, open the file with the "`.lpr`" extension in the project inspector (*Project -> Project inspector*) and add the `cthreads` unit in the first paragraph in uses.

ZEOS DATABASE COMPONENT LIBRARY

BY DON WILBRINK

Database components for Delphi, FreePascal/Lazarus, Kylix, C++ Builder
Brought to you by: egonhugeist, markdaems, marsupilami79, miseeger

Starter   **Expert**

Note: Delphi Next Gen support (Android, iOS, OS X,) has been added the Zeos 8.0
development version. Supported compilers start with Delphi XE 10.2 Tokyo. For problems
please contact us in the forums on https://zeoslib.sourceforge.io.

INTRODUCTION
The ZeosLib is a set of database components for MySQL, MariaDB, PostgreSQL, Interbase,
Firebird, MS SQL Server, SAP Adaptive Server Enterprise and Adaptive Server Anywhere
(previously Sybase), Oracle and SQLite for Delphi, FreePascal/Lazarus and C++ Builder.

FEATURES
- Support object for Sequences (Generators)
- Support for Firebird, Interbase and PostgreSQL database events
- Support object for retrieving database objects meta data
- Support object for executing database scripts
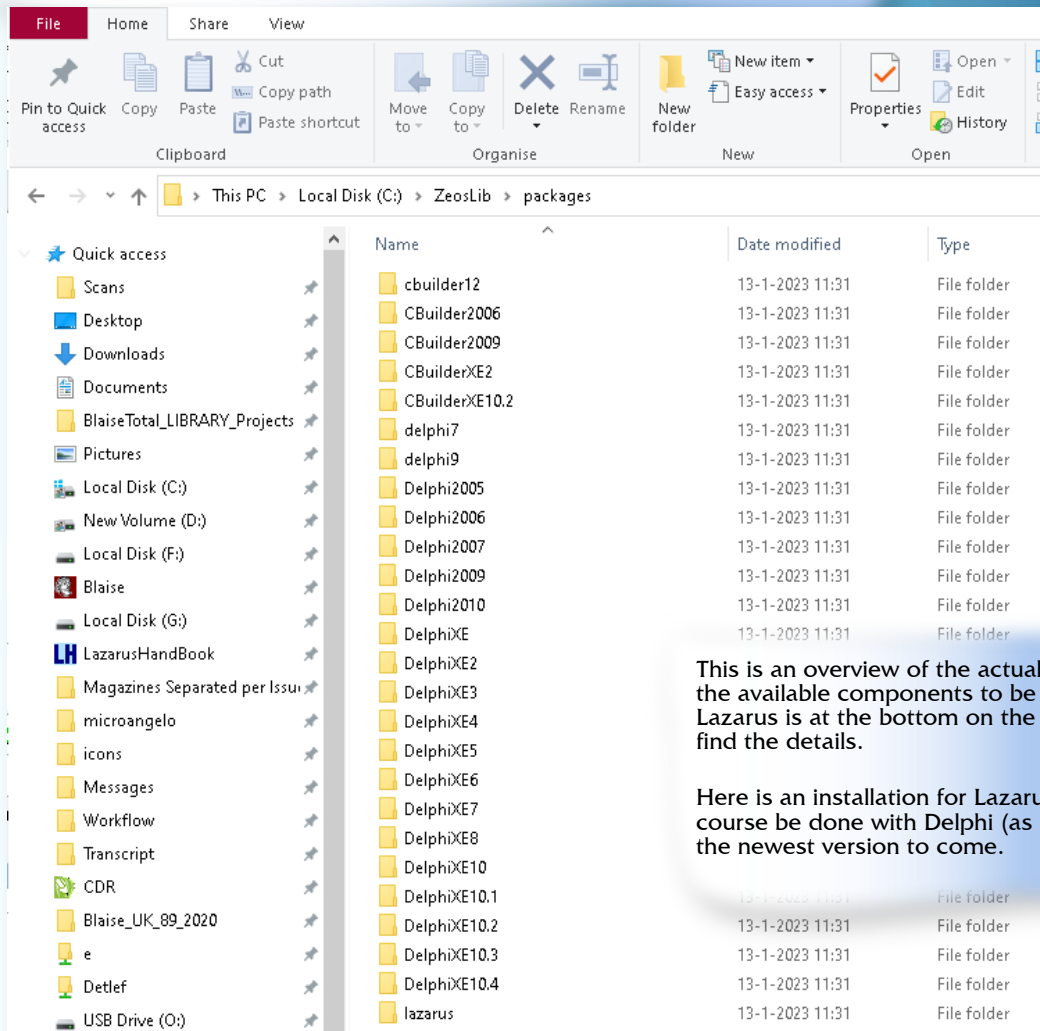
This is an overview of the actual situation for the available components to be installed. Lazarus is at the bottom on the next page you find the details.

Here is an installation for Lazarus but it can of course be done with Delphi (as old as 7) and the newest version to come.

Figure 1:

> This PC > Local Disk (C:) > ZeosLib > packages > lazarus >

New folder

Name	Date	Type	Size
build	13-1-2023 11:31	File folder	
lib	13-1-2023 11:31	File folder	
zcomponent.lpk	14-7-2021 23:47	Lazarus Package F...	5 KB
zcore.lpk	14-7-2021 23:47	Lazarus Package F...	5 KB
zdbc.lpk			
zparsesql.lpk			
zplain.lpk			

ss

ls

ts

I_LIBRARY_Projects

(C:)

You need to select only one .lpk.:
Zcomponent .lpk. Installartion will do
everything automatically during the
process we show here:

Now start Lazarus go to Package.Select
open package file and choose the
package I show here on top.

Lazarus IDE v2.2.4 - project1

File   Edit   Search   View   Source   Project   Run   Package   Tools   Window   Help

Standard

New Package ...
Open Loaded Package ...
Open Package File (.lpk) ...
Open Package of Current Unit

Object Inspector

Components (filter)

Form1: TForm1

Add Active File to Package ...
New Component ...

Package Graph ...
Package Links ...
Install/Uninstall Packages ...
Online Package Manager ...   Shift+Ctrl+Alt+O

The Group list shows up:

Her you can see what files and
where they are located and find
what required packages are
needed.
Down at the bottom you can see
the path from where you load the
packages.

Package zcomponent V7.2.14

Save   Compile   Use   Add   Remove   Options   Help   More

(filter)

Files
  ..\..\src\component\ZComponentReg.pas
  ..\..\src\component\ZConnection.pas
  ..\..\src\component\ZDatasetUtils.pas
  ..\..\src\component\ZUpdateSqlEditor.pas
  ..\..\src\component\ZConnectionGroup.pas
  ..\..\src\component\ZGroupedConnection.pas
  ..\..\src\component\ZIBEventAlerter.pas
  ..\..\src\component\ZIBEventAlerter.pas
Required Packages
  IDEIntf
  zdbc (>=7.2)
  FCL (>=1.0)

File Properties

Files 8, has Register procedure: 1, in package uses section: 8

C:\ZeosLib\packages\lazarus\zcomponent.lpk

Package zcomponent V7.2.14

Save Compile Use Add Remove Options Help More

Add to Project
Install
Uninstall

Files
- ..\..\src\...Reg.pas
- ..\..\src\component\ZConnection.pas
- ..\..\src\component\ZDatasetUtils.pas
- ..\..\src\component\ZUpdateSqlEditor.pas
- ..\..\src\component\ZConnectionGroup.pas
- ..\..\src\component\ZGroupedConnection.pas
- ..\..\src\component\ZIBEventAlerter.pas
- ..\..\src\component\ZIBEventAlerter.pas

Required Packages
- IDEIntf
- zdbc (>=7.2)
- FCL (>=1.0)

File Properties

Files: 8, has Register procedure: 1, in package uses section: 8

C:\ZeosLib\packages\lazarus\zcomponent.lpk
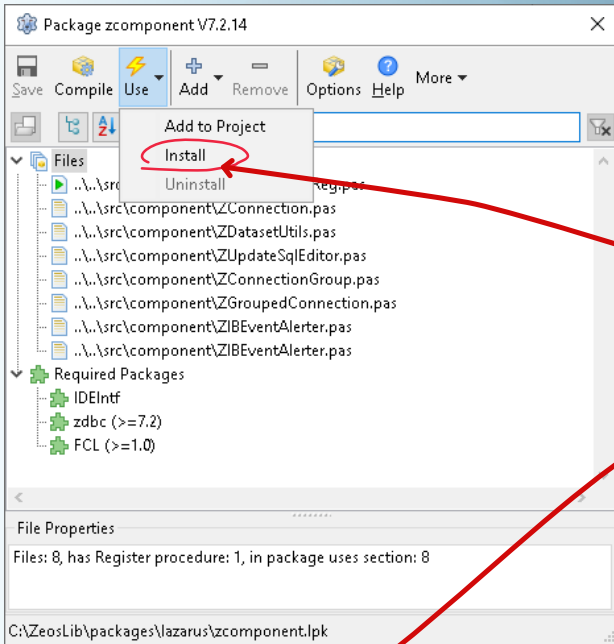
Under Use you find the option to install. Click on that and a few questions come up:

A notification appears that will tell you the components will be installed. That's what we wanted.

Lazarus must be restarted and the you will see the components added

Automatically installed packages

Installing the package zcomponent 7.2.14 will automatically install the packages:
zdbc 7.2.14
zplain 7.2.14
zparsesql 7.2.14
freetypelaz 1.0
zcore 7.2.14

OK   Cancel

Rebuild Lazarus?

The package "zcomponent 7.2.14" was marked for installation.
Currently Lazarus only supports static linked packages.
The real installation needs rebuilding and restarting of Lazarus.

Do you want to rebuild Lazarus now?

Yes   No

Lazarus IDE v2.2.4 - project1

File Edit Search View Source Project Run Package Tools Window Help

Standard Additional Common Controls Dialogs Data Controls Data Access Zeos Access

BLAISE PASCAL ◉ MAGAZINE 107/108
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
- Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Skia installation and use
The PDF indexer explained
Fresnel for Lazarus
The Pendulum
Using styles with a VCL programme
Lazarus programme using BufDataset
Cross compiling with FPC Up De Luxe
Quantum Internet

The newest LIB Stick

1

2

LAZARUS ◉
HANDBOOK

3

4

LEARN TO PROGRAM
USING LAZARUS

HOWARD
PAGE-CLARK

DAVID DIRKSE
including 50 example projects
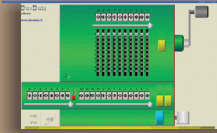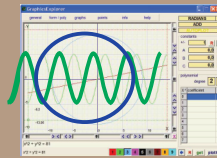
BLAISE PASCAL MAGAZINE
COMPUTER (GRAPHICS)
MATH & GAMES IN
PASCAL

1. **One year Subscription**
2. **The newest LIB Stick**
   - **All issues 1- 107/108**
   - **On Credit Card and download**
3. **Lazarus Handbook**
   - **PDF including Code**
4. **Book Learn To Program**
   - **using Lazarus PDF including**
   19 lessons and projects
5. **Book Computer Graphics**
   Math & Games
   PDF including ±50 projects

2

BLAISE PASCAL MAGAZINE

procedure
var I := ½ to ⁿ do
begin
end;
Prof Dr.Wirth, Creator of Pascal Programming language   Blaise Pascal, Mathematician

BLAISE PASCAL MAGAZINE

procedure
begin
var I := ½ to ⁿ do
begin
end
and;
Prof Dr.Wirth, Creator of Pascal Programming language   Blaise Pascal, Mathematician

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

Prof Dr.Wirth, Creator of Pascal Programming language

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

NORMAL PRICE € 275

CHRISTMAS
PRICE € 150

CHRISTMAS
& NEW YEAR

SUPERPACK
2022/23

OPEN AI
You most likely have already heard about OpenAI's ChatGPT "search machine".
It is a chat application that is driven by a number of neural networks, trained on
textual information found on the internet in 2021. It resulted in a training vector
roughly 90 billion items long… which is huge.

The impressive thing, is that you can use natural language to communicate with the chat robot,
and it does not even have to be in English, but can be written in numerous other languages too.
You can pose all sorts of questions, and it will make a darn good attempt in answering your
question, even offering detailed explanation of how something operates.
You can ask it to make a program for you, or to translate something from one programming
language to another, or explain to you how some math problem is handled, or ask it to produce a
joke or a report about whatever topic you find interesting.

Welcome to ChatGPT

Log in with your OpenAI account to continue

Log in   Sign up

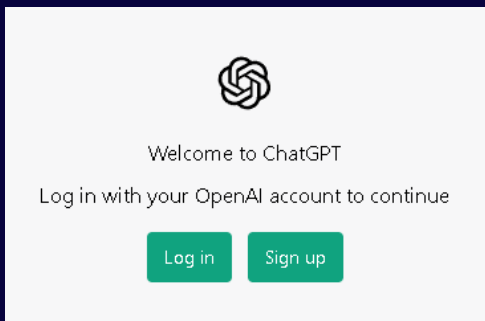It is really impressive what it can do, but it is also rather entertaining.

One of my favourite questions is to ask it to combine things that are not really combinable. It
actually is wise enough to tell you that things are not really compatible, but when I then insists,
that I want to get an answer on the combination, it actually comes up with a clever reply to satisfy
my question.
For example, I ask it:

        what do you get if you combine an empty soda can with a whistle?

and it wisely replies:

> *It is not possible to combine an empty soda can with a whistle to create a functional
> object or device. An empty soda can is a container made of metal or aluminium that is
> used to hold carbonated beverages, while a whistle is a small device that produces a
> high-pitched sound when blown into. These two objects do not have any functional
> compatibility or complementary capabilities, so combining them would not result in a
> functional object or device.*

But when I then insist: I get this IMO great response:

If you really tried to combine an empty soda can with a whistle, it is possible that you could fashion the soda can into a makeshift mouthpiece for the whistle. To do this, you would need to cut a small hole in the side of the can and attach the whistle to the hole using some kind of adhesive or fastener. This would allow you to blow into the can to produce a sound through the whistle.

However, it is important to note that this combination would not be a particularly effective or reliable way to produce a whistle sound, and it is not a recommended method for using a whistle. The can may not provide a secure or comfortable mouthpiece, and the sound produced may not be as clear or loud as a traditional whistle.

It is generally better to use a whistle as intended, with a properly designed mouthpiece and body, rather than trying to modify it with makeshift materials.

Often one will be extremely positively surprised about the quality of the replies, but one should never take its replies for a fact! It is quite often making things up, just because it makes sense for it to do so.

I for example asked it this:

```
tell me more about the acorn system 1
```

The Acorn System 1 is a single board computer from 1976 and was actually the first computer Acorn produced, so it is a rather difficult to obtain collectible that I happen to have in my collection.

I got this reply:

The Acorn System 1 is a vintage computer system that was produced by Acorn Computers in the late 1970s. It was one of the first personal computers to be produced in the United Kingdom, and it was widely used by schools and businesses in the 1980s.

The Acorn System 1 was based on the MOS Technology 6502 microprocessor, which was also used in other early personal computers such as the Apple II and the Commodore 64. It came with 4KB of RAM as standard, and it had a monochrome monitor and a full-size keyboard. The system ran a version of the BASIC programming language and could be used for a variety of tasks such as word processing, data entry, and simple programming.

The Acorn System 1 was known for its simplicity and ease of use, and it was popular among beginners and hobbyists. It was also relatively inexpensive compared to other personal computers of the time, which made it a popular choice for schools and businesses.

The Acorn System 1 is now considered to be a vintage computer and may be of interest to collectors of vintage technology. It is a historically significant computer that played a significant role in the development of the personal computer industry in the United Kingdom.

ChatGPT is at capacity right now

Get notified when we're back

Write a sonnet about the status of ChatGPT.

Amidst the rush of curious minds,
The chatbot ChatGPT stands strong and true,
Withstanding the influx of requests,
Working hard to fulfill each one's due.

But alas, the server cannot cope,
And the error message rings loud and clear,
"Please check back soon," it gently hopes,
As it begs for a moment's reprieve, to reappear.

As the chatbot works to restore its might,
We wait in anticipation, with hopeful hearts,
For the chance to chat and learn, with delight,
And see what knowledge and wisdom it imparts.

So hold on tight, and wait for its return,
For ChatGPT will soon be back, and your patience will be well-earned.

At first hand, and for people who do not know about the Acorn System 1 it seems like a very plausible reply, It is correct that it was form the 1970's and it was one of the first personal (affordable) computers produced in UK. It is also correct it is based on a MOS 6502 microprocessor, and that it is of interest for collectors.
But the rest is just plain wrong.

- It does not have 4KB of RAM… it has 256 bytes.
- It has no monochrome monitor. It has 9 digits 7 segment LED display.
- It has no BASIC. It was programmed in hexadecimal op codes.
- It has no full size keyboard. Rather it has a very simple 25 keypad of which 16 of them were used for entering a hexadecimal digit.
- It was not known for being popular in schools.
  It was originally made to control feeding gates in pig barns.

I can then tell it what it did wrong and ask it to provide a better reply.
Then I get this:

The Acorn System 1 was a microcomputer that was produced by Acorn Computers in the late 1970s. It was based on the MOS Technology 6502 microprocessor and came with 1KB of RAM as standard, which could be expanded to a maximum of 16KB. The system had a 16-color graphics display and a full-size keyboard, and it ran a version of the Acorn Atom operating system.

The Acorn System 1 was a relatively simple and inexpensive computer that was popular among beginners and hobbyists. It was known for its ease of use and was widely used by schools and businesses in the 1980s.

I hope this revised description is more accurate and helpful. If you have any further questions about the Acorn System 1, please don't hesitate to ask.

Which does not become much better. In fact now it should be 16 colour capable

Anyway this is just to show that you should not put too much trust in what it tells you.
Sometimes it is accurate and sometimes it simply makes things up.

You can try out Open AI's ChatGPT by registering and signing in here.

`https://chat.openai.com/auth/login`

ChatGPT

Examples

"Explain quantum computing in simple terms" →

"Got any creative ideas for a 10 year old's birthday?" →

"How do I make an HTTP request in Javascript?" →

Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests

Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge of world and events after 2021

WIKIPEDI**A**

OpenAI is an artificial intelligence (AI) research laboratory consisting of the for-profit corporation OpenAI LP and its parent company, the non-profit OpenAI Inc. The company conducts research in the field of AI with the stated goal of promoting and developing friendly AI in a way that benefits humanity as a whole.
The organization was founded in San Francisco in late 2015 by Sam Altman, Elon Musk, and others, who collectively pledged US$1 billion. Musk resigned from the board in February 2018 but remained a donor. In 2019, OpenAI LP received a US$1 billion investment from Microsoft and Matthew Brown Companies. OpenAI is headquartered at the Pioneer Building in Mission District, San Francisco.

In December 2015, Sam Altman, Elon Musk, Greg Brockman, Reid Hoffman, Jessica Livingston, Peter Thiel, Amazon Web Services (AWS), Infosys, and YC Research announced the formation of OpenAI and pledged over US$1 billion to the venture.
The organisation stated it would "freely collaborate" with other institutions and researchers by making its patents and research open to the public. OpenAI is headquartered at the Pioneer Building in Mission District, San Francisco.

In April 2016, OpenAI released a public beta of "OpenAI Gym", its platform for reinforcement learning research. In December 2016, OpenAI released "Universe", a software platform for measuring and training an AI's general intelligence across the world's supply of games, websites and other applications.

In 2018, Musk resigned his board seat, citing "a potential future conflict (of interest)" with Tesla AI development for self driving cars, but remained a donor.

In 2019, OpenAI transitioned from non-profit to "capped" for-profit, with profit cap set to 100X on any investment. The company distributed equity to its employees and partnered with Microsoft, who announced an investment package of US$1 billion into the company. OpenAI then announced its intention to commercially license its technologies.

In 2020, OpenAI announced GPT-3, a language model trained on trillions of words from the Internet. It also announced that an associated API, named simply "the API", would form the heart of its first commercial product. GPT-3 is aimed at natural language answering of questions, but it can also translate between languages and coherently generate improvised text.

In 2021, OpenAI introduced DALL-E, a deep learning model that can generate digital images from natural language descriptions.

Around December 2022, OpenAI received widespread media coverage after launching a free preview of ChatGPT, its new AI chatbot based on GPT-3.5. According to OpenAI, the preview received over a million signups within the first five days.

According to anonymous sources cited by Reuters in December 2022, OpenAI was projecting a US$200 million revenue for 2023 and US$1 billion revenue for 2024.
As of January 2023, it was in talks for funding that would value the company at $29 billion.

`https://en.wikipedia.org/wiki/OpenAI`

COMPONENTS
DEVELOPERS *4*

OPEN AI
BY DETLEF OVERBEEK
The original article was
published by NRC, a dutch Journal

ARTICLE PAGE 1/4

WHAT'S OPEN
ABOUT
SMART CHATBOT MAKER
OPENAI?

(Photo: iStock/metamorworks)

Artificial intelligence OpenAI, the most successful AI lab today, was founded to ensure that AI
"*benefits all of humanity*". Critics say that mission is a smokescreen for commercial interests.

Free-access artificial intelligence (AI) where anyone can look under the bonnet.
That was the ideal for which a new club was founded in San Francisco in 2015: OpenAI, a non-profit AI laboratory.
"*Everything the group develops becomes available to everyone*," This statement was made by co-founder **Sam
Altman** towards journalist Steven Levy of tech magazine Wired at the time.
The founders presented the lab as a non-profit with a mission: to stop the misuse of AI. Altman: "*Because we are
not a for-profit company like Google, we can focus on something other than enriching our shareholders, namely on what
we think is best for the future of humanity*."
Seven years later, OpenAI is the world's most prominent AI lab. "*First we got the internet. Then came the iPhone. Now
there is ChatGPT*," Alexander Klöpping (*A technical journalist*) tweeted in early December about **ChatGPT**, **OpenAI's**
chatbot launched in late November. And he is not the only one who considers the introduction of this language
model a major event.

The whole world is baffled by the AI system that writes an essay on the Cuban crisis, or an episode of a sitcom,
with the greatest of ease. It effortlessly answers all kinds of questions, including about itself, in seemingly natural,
human language.
According to The New York Times, **CHAT GPT MAY WELL BE THE SUCCESSOR TO SEARCH ENGINES.**
Not only can it provide suggestions for Christmas gifts, it can also "serve up information in clear, simple sentences,
rather than a list of internet links, and explain concepts in a way that people can easily understand."

Within a week, ChatGPT already had one million users. And that is the second time in just under a year that an
OpenAI application became a hit. Because in the summer there was the hype around the revolutionary image
generator DALL-E 2, which allows anyone to create instant 'art'. The system, whose name nods to Salvador Dalí,
does not turn its hand to a task like "*Draw a horse on the moon in the style of Andy Warhol*", but can also generate
realistic images that look like photographs.

OpenAI was founded by a group of wealthy Silicon Valley investors, who collectively put more than a billion dollars
into the lab. Among them: **Elon Musk (Tesla, Space X), Sam Altman of Y Combinator** (**the startup accelerator
of Airbnb and DropBox**, among others) and **PayPal founder Peter Thiel**.
"*I think the best defense against abuse of AI is to enable as many people as possible to become able to use AI*,"
Musk said in the a fore mentioned article in tech magazine Wired. "*If everyone has AI power, there can be no AI
superpower with an individual or small group of people.*"

According to OpenAI's website, its goal is still "*to ensure that Artificial General Intelligence benefits all of humanity.*"
But so far as that mission was once sincere, critics say little of it remains.

**What the company publicly espouses
is NOT CONSISTENT.
Gradually, it has allowed a development
where its founding ideals of transparency,
openness and collaboration have been eroded**

VANISHED IDEALS

it structure was swapped for a hybrid model, which would henceforth run a commercial company from the non-profit organisation. Microsoft invested a billion dollars in OpenAI that same year and was granted the exclusive licence of language model **GPT3**, the forerunner of **ChatGPT**.

"*I look at OpenAI with a mix of admiration and concern*," says **Jelle Zuidema**, researcher language technology at the UVA.(University for Amsterdam)

"*Enable AI*," Musk said in the already mentioned article in tech magazine Wired. "*If everyone has AI power, there can be no AI superpower with an individual or small group of people.*"

"*It is clear that there are incredibly good techies working there, and that there is a corporate culture where creative new ideas can flourish. But where are the ideals they were founded with: open technology, a counterweight to Big Tech?*" Those are hard to find, Zuidema observes. "*The underlying technology of their products is very limited to be explored by independent scientists. And GPT3 is funded and marketed by tech giant Microsoft.*"

OpenAI is not what it seems, **Karen Hao** observed back in 2020 in **MIT Technology Review**. The journalist was allowed to visit the AI lab, housed in a historic building in San Francisco on which **PIONEER BUILDING** is written in large letters. Once home to the Pioneer Truck Factory, it now pioneers AI in rooms with names like 'A Space Odyssey'. OpenAI shares the pioneer building and canteen with another tech lab: **Neuralink**, Elon Musk's company that develops brain chips.

"*What the company publicly espouses is not consistent with how it operates behind closed doors,*" Hao wrote. "*Gradually, it has allowed a development where its founding ideals of transparency, openness and collaboration have been eroded by fierce competitiveness and increasing pressure to find more and more funding.*"

"WHAT'S ON OPEN ABOUT OPENAI?" asks tech philosopher **Marleen Stikker**, director of De Waag, rhetorically.
" The name suggests that it is about **collective knowledge**, but that is not the case.
The software and models **are not open for research or reuse**.
SCIENTISTS CAN DO LITTLE WITH THEM BECAUSE THE SYSTEMS ARE TOO BIG."

The ideals posted by the founders appear to be at odds with other ambitions of the AI Lab.
For instance, developing **Large Language Models (LLMs),** the giant language models like **ChatGPT** that have 'read' millions of texts and can perform a wide variety of language tasks based on that knowledge, is incredibly expensive.
A training round of GPT3 costs $12 million,
calculated a tech entrepreneur based on OpenAI's own information on the language model's energy consumption. So a lot of money is needed from new investors to stay at the forefront of LLMs, which makes it hard to stay non-profit.

OpenAI's ideals are implausible

TOO BIG TO CONTROL

Furthermore, the trend of LLMs growing like cabbages is difficult to reconcile with transparency.
Economies of scale make the models work better - the more text they have read and the more computing power they have, the better they get - but that gets in the way of transparency:
a database of millions of texts is too big to check for inaccuracies and biases.
And then there is OpenAI's ambition to be the first to achieve **'AGI': Artificial General Intelligence**, or AI that can perform the same intellectual tasks as a human brain.
In this, the lab is competing with other tech companies, and that race, critics say, stands in the way of safe, responsible development of AI.
"*If companies know that everyone is racing towards the latest AI at life-threatening speed,*" then PhD student Miles **Brundage** told Wired back in 2015, "*they might be less inclined to put down guarantees for safe AI.*"
In 2018, **Brundage** joined OpenIA - and he is not the only white male working there.
A lack of diversity applies to the entire tech industry, but in OpenAI's case it is extra difficult to reconcile with its claim to want to make AI for all of humanity.

"*Even if OpenAI had the best intentions,*" says **professor Tamar Sharon**, who heads Radboud University's (Nijmegen - Netherlands) interdisciplinary Digitalisation and Society research group, "*they do not represent humanity in any way. OpenAI serves the agenda of their investors: Silicon Valley billionaires pushing technology as the solution to all humanity's problems. Their own technology, that is, which they sell to us.*"

OpenAI's ideals are implausible, Sharon believes. "*OpenAI was founded by a group of billionaires from their ideology of Effective Altruism, EA, which includes fallen crypto-billionaire Sam Bankman-Fried. The EA movement wants to save humanity from an apocalypse in which a machine with human intelligence destroys humanity.*"
Sharon is concerned about the influence of these wealthy Effective Altruism supporters on AI. "*They talk about 'beneficial AI' that can take over human labour in the future, but for now much AI is powered by human labour in low-wage countries: tens of thousands of underpaid workers sifting through datasets.*
And Large Language Models like ChatGPT have a huge ecological footprint: they guzzle energy. The current trend in AI land is to make these LLMs bigger and bigger, because that will make them perform better. What that means for the planet is by no means beneficial for humanity."

Beware when they talk about 'humanity' because they decide what they mean by that.

COLLECTIVE KNOWLEDGE

Meanwhile, OpenAI's revenue model is also starting to become clear. The world gets to experience **DALL-E 2** and **ChatGPT** for free but after that we have to start paying for it. When your free credits run out, you have to buy new ones to keep using the application. The AI Lab will also start earning from other tech companies that want to build apps using OpenAI's technology.

Last month, a leaked presentation for investors revealed that OpenAI expects to generate $200 million in revenue by 2023. And by 2024, a billion.

OpenAI's ideals have been "a smokescreen" for investors' commercial interests from the outset, according to Marleen Stikker. The tech philosopher does not like to put the fate of humanity in the hands of Silicon Valley billionaires. "Beware when they talk about 'humanity' because they decide what they mean by that. Usually it does not mean humanity here and now, but that of the future, as they see it for themselves."

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 👁 MAGAZINE107/108

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 4

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

COMPONENTS 4
DEVELOPERS 4

Donate for Ukraine and get a free license at:

kbmMW Professional
and Enterprise Edition v. 5.21.00
Studio XE5 to 11 just now released!
kbmMemTable v. 7.97.00 Standard and
Professional Edition released

This is a significant new release with new high performance transports, 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. 7.96.00 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.