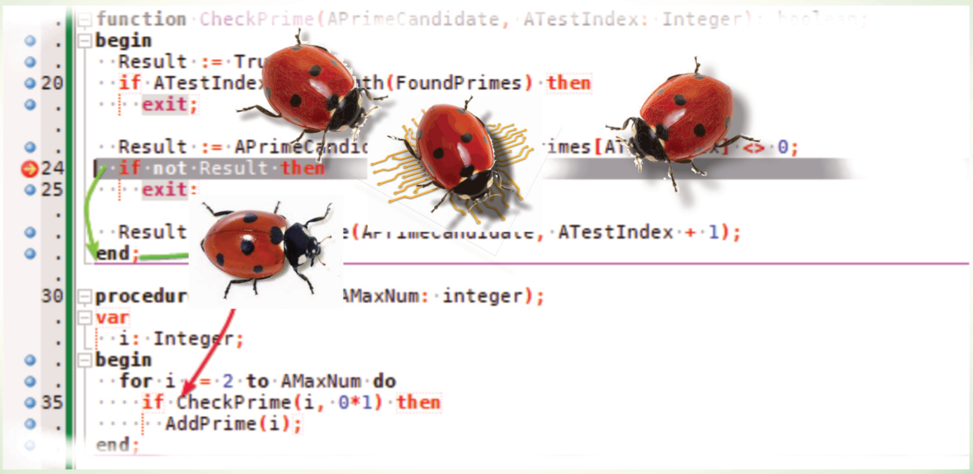
BLAISE PASCAL MAGAZINE 110 / 111

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



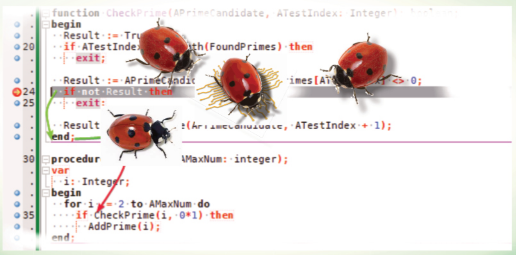
```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := Tru
  if ATestInde          th(FoundPrimes) then
    exit;

  Result := APrimeCandic            imes[AT        ] <> 0;
  if not Result then
    exit;

  Result                  (APrimeCandidate, ATestIndex + 1);
end;

procedur              AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0*1) then
      AddPrime(i);
end;
```

BLAISE PASCAL MAGAZINE 110 / 111

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

Blaise Pascal

CONTENT



```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := Tr[illegible]
  if ATestInde[illegible]th(FoundPrimes) then
    exit;

  Result := APrimeCandi[illegible]imes[AT[illegible]] <> 0;
  if not Result then
    exit;

  Result [illegible](APrimeCandidate, ATestIndex + 1);
end;

procedur[illegible]AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0*1) then
      AddPrime(i);
end;
```

ADVERTISING

Niklaus Wirth

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

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

CONTRIBUTORS

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

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

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

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,
by creating this new issue of our Magazine
I had the summer very much in my mind.
I suppose you do have the same feeling of need form warmth and sunshine.
My wishes seem to be fulfilled by now: It is warm an beautiful.

I had so much to tell you that I now had to make a double issue out of it. There are even more extra's but they are for the next issue 112.

We already had talked about the AI (*Artificial Intelligence*) and image classifiers.
Now some more disturbing news came up:
I write about that in the article **AI enabled brain scanner.**
We need to think what's more to come because this is a bit spooky. Let's discus where this has to end.

I am personally convinced that what ever comes to your mind we can make it,
how strange it ever may be:
read the article.

To create something that I never would have thought of: **Lazarus for Visual Studio**.
It is for good reasons.
Lazarus is the most advanced environment or is coming to be. Anything you want you can do with Lazarus. Even you yourself because the sources are open and the industry wants this.
All the important Programming Languages have a direct connection for Visual Studio,
so Lazarus needed that as well.

Of course there is still a lot to do - but we are getting there. My future goal is to make Pascal available for the kids and youngsters.
I have some special ideas about that.

I think the team and I should - with playing and fooling around - make the environment so friendly for them that they will learn without noticing by solving problems they themselves are creating…
AI will play a big role in this because it already has shown that it can be very useful.
… I would not be able to translate an issue into other languages without it, because loosing so much time.

Martin Friebe has created and written the debugger story and this is a very good way to learn what it is capable of and doing that.
He extended even more and better functions for it. Lazarus has now the debugger we needed so much:
simple in use but very versatile.
- If you have request for it let me know…

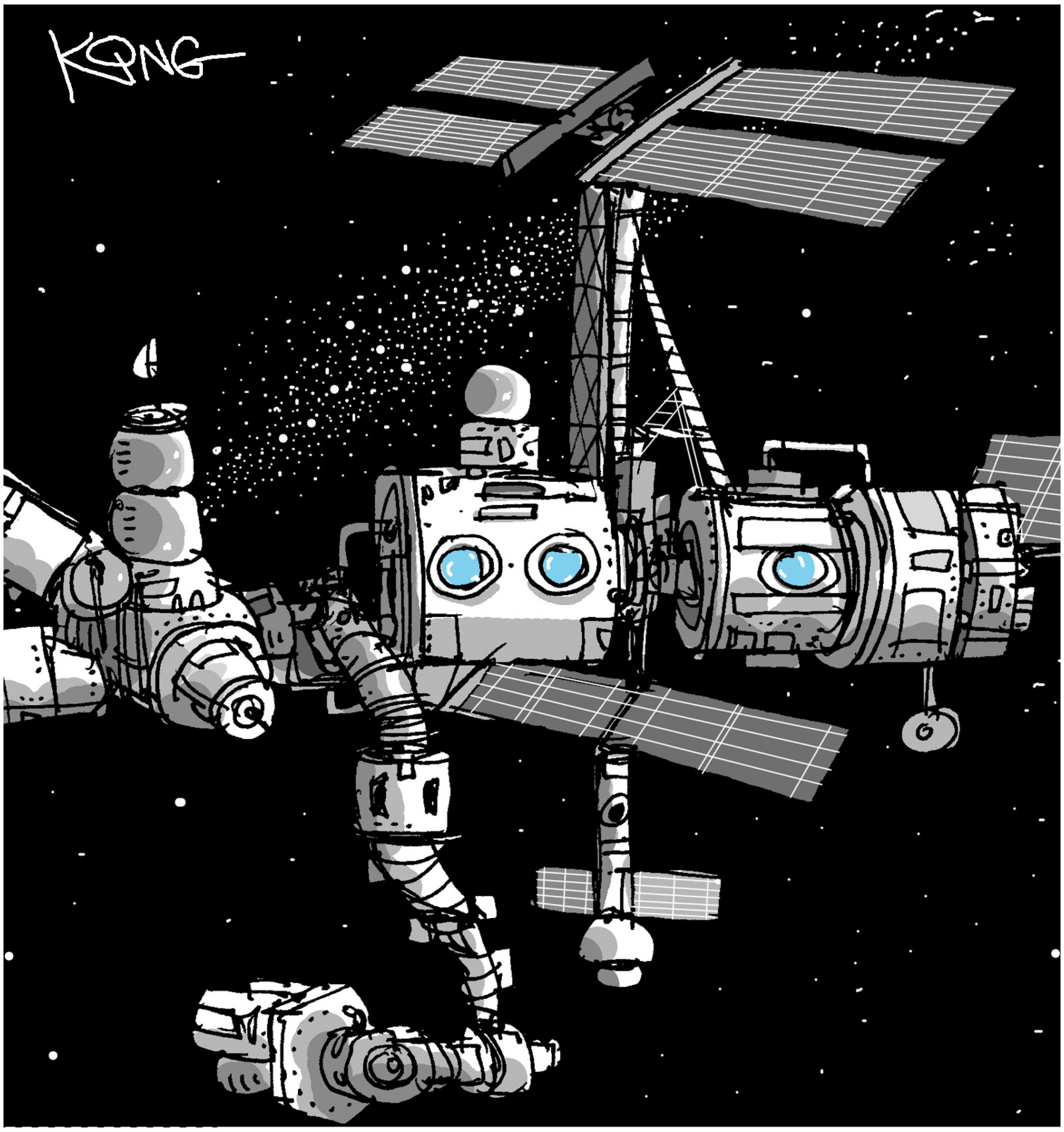
I had asked Michael van Canneyt to write the new **PDF Kit for Blaise Pascal Magazine**,
so that we can search in all pages and issues for a special text.
That is an enormous task but he managed to do so.
It is now available and we can give you even better service than ever. We will create some extra examples...

Jim McKeeth left **Embarcadero** to do something very new to him: Working on the web3 version…(*I already wrote about about webs 3 in the last issue*)
I spoke to his successor **Ian Barker**, and we discussed some new actions for Delphi for the future.
One of them is that we will get the news from **Delphi** early enough to publish so that it will be **REAL** news.
I thank him for that in advance.
So for the future we will be able to surprise you even more about our favourite language Pascal and try to find young people to learn about and with it.

Yours

Detlef

From our technical advisor Jerry King



"Daddy's at work, sweetheart. Have mommy look up the Wi-Fi password."

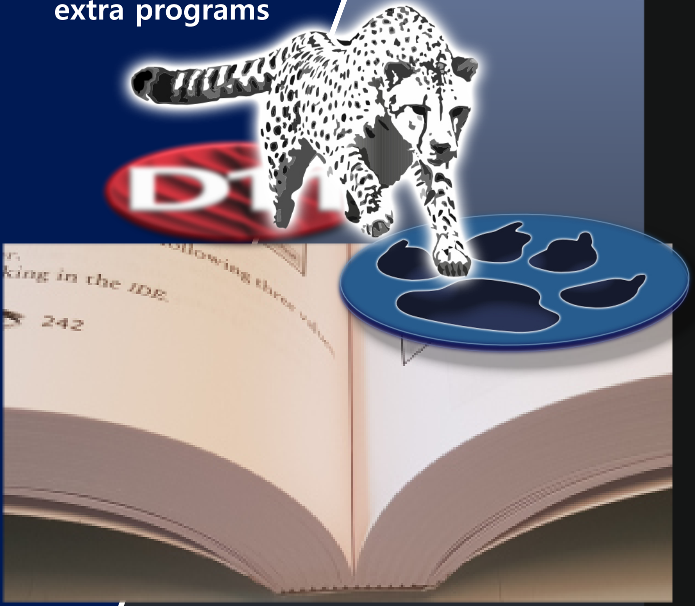
ADVERTISEMENT

LAZARUS HANDBOOK
POCKET PACKAGE (2BOOKS)

Price: € 35,00
Excluding VAT and Shipping

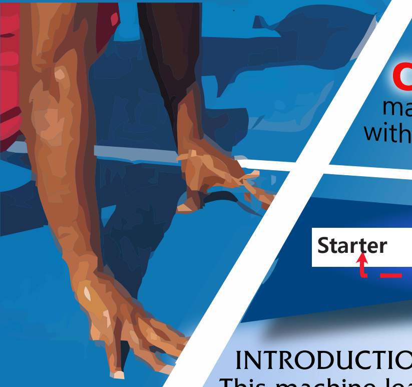
**934 PAGES
PDF & INDEX
INCLUDED**

- English
- Printed black & white
- 2 Volumes
- PDF included
- 934 Pages
- Weight: 2kg
- Extra protected
- Including
  40 Example
  projects and
  extra programs



BlaisePascalMagazine
PDF viewer included

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



CLASSIFY CIFAR10

maXbox Starter 105 – Image Classifier with loading and testing a pre-trained model.

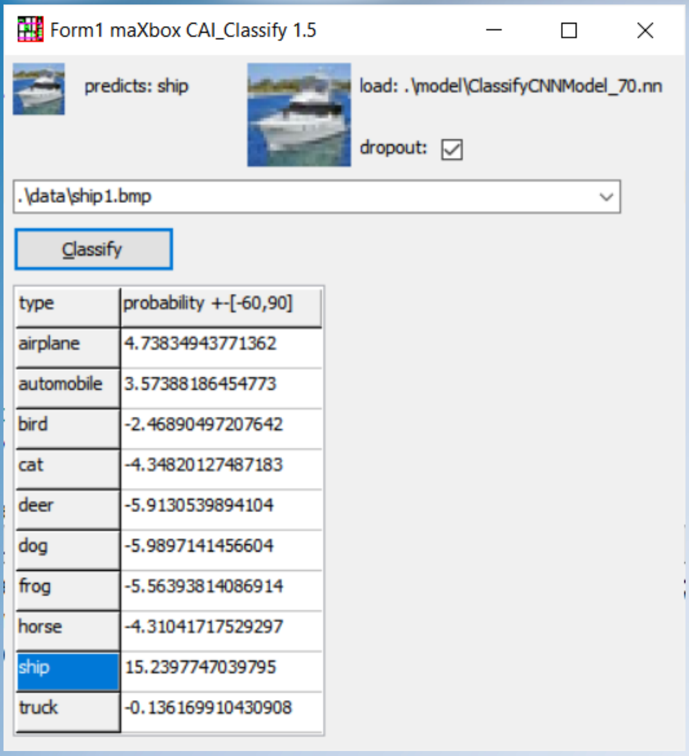
Starter Expert

INTRODUCTION

This machine learning tutor explains a classifier based on the so called CIFAR-10 Image Classifier with a pre-trained model. The pre-trained model is a file: ClassifyCNNModel_70.nn

As the name implies, it is a CNN-model. A Convolutional Neural Network (CNN*) is a type of deep learning algorithm that is particularly for image recognition and object-detection tasks. It is made up of multiple layers, including convolution layers, pooling layers, and fully connected layers.
* *CNNs also known as Shift Invariant or Space Invariant Artificial NN.*

Now let's have a look at the app/script below with individual images from Cifar test data. For this, we wrote two useful functions. The first one returns the label associated with predictions made by the model. The second one accepts one image as an argument. Then it will show the image, the prediction the model made and the actual class the image belongs to. Also other probabilities are shown in the multi-classification grid:



Form1 maXbox CAI_Classify 1.5

predicts: ship

load: .\model\ClassifyCNNModel_70.nn

dropout: ✓

.\data\ship1.bmp

Classify

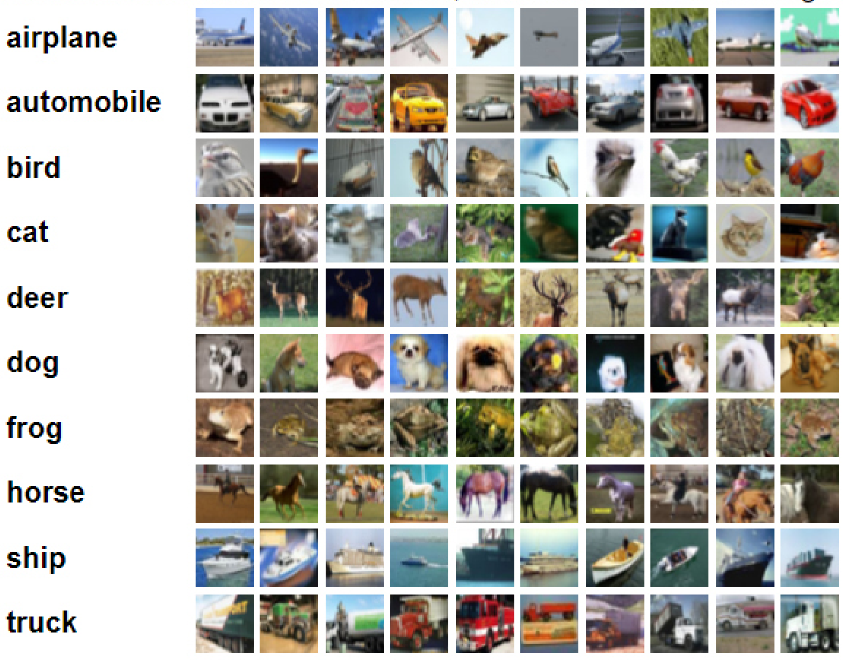
type	probability +-[-60,90]
airplane	4.73834943771362
automobile	3.57388186454773
bird	-2.46890497207642
cat	-4.34820127487183
deer	-5.9130539894104
dog	-5.9897141456604
frog	-5.56393814086914
horse	-4.31041717529297
ship	15.2397747039795
truck	-0.136169910430908

This app allows you to classify pictures from an airplane to a truck or a train. And you see similarities for example a ship (15.2) has some elements of an airplane or automobile (4.7, 3.5) in his feature map.Specifically, models are comprised of small linear filters and the result of applying filters called activation maps, or more generally, feature maps. Looking at the following dataset, it will extract features in a constant dot product, even though images has shadows or positioned with various angle. It is important to note that filters acts as feature detectors from the original input image, in our case 32*32 bitmaps.

```
Const PICPATH   = '.\data\';
    TRAINPATH = '.\model\ClassifyCNNModel_70.nn';
```

The proper way to use a CNN doesn't exists. The advice for ugly score is to use a smaller learning rate or larger batch size for the weights that are being fine-tuned and a higher one for the randomly initialized weights (*e.g. the ones in the softmax classifier*) TNNetSoftMax. Pre-trained weights (*in ClassifyCNNModel_70.nn*) are already good, they need to be fine-tuned, not distorted.

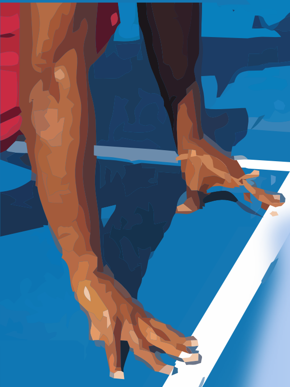
Here are the classes in the dataset, as well as 10 random images from each:



The learning rate is the crucial hyper-parameter used during the training of deep convolution neural networks (DCNN) to improve model accuracy;
By following these ways you can make a CNN model that has a validation set accuracy of more than 95 % but the question is how specific or relevant is this validation.
In our example, values smaller than 0.7 mean false while values bigger than 0.7 mean true. This is called monopolar encoding. CAI also supports bipolar encoding (-1, +1). Let's have a look directly into the source code for the labels and the classify method:

```
var cs10Labels: array[0..9] of string;

procedure setClassifierLabels;
begin
  cs10Labels[0]:= 'airplane';
  cs10Labels[1]:= 'automobile';
  cs10Labels[2]:= 'bird';
  cs10Labels[3]:= 'cat';
  cs10Labels[4]:= 'deer';
  cs10Labels[5]:= 'dog';
  cs10Labels[6]:= 'frog';
  cs10Labels[7]:= 'horse';
  cs10Labels[8]:= 'ship';
  cs10Labels[9]:= 'truck';
end;
```

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images.
We now build the Convolution neural network by using 1 Convolution layer, 4 Relu-activation function, dropout- and pooling-layer, 1 fully Connected layer and a SoftMax activation function. Below is the list of all layers which we also define the optimizer and a loss function for the optimizer:



```
Debug TNNet.Struct.LoadFromString ST:
-1)TNNetInput:32;32;3;0;0;0;0;0
0)TNNetConvolutionLinear:64;5;2;1;1;0;0;0
1)TNNetMaxPool:4;4;0;0;0;0;0;0
2)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
3)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
4)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
5)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
6)TNNetDropout:2;0;0;0;0;0;0;0
7)TNNetMaxPool:2;2;0;0;0;0;0;0
8)TNNetFullConnectLinear:10;1;1;0;0;0;0;0
9)TNNetSoftMax:0;0;0;0;0;0;0;0
```

The main procedure to classify incoming images loads the model, decides dropout or not (later more) and creates input- and output-volumes with the shape of 32;32;3 or a 32x32x3 volume:

```
Begin
NN:= THistoricalNets.create; //TNNet.Create();
NN.LoadFromFile(TRAINPATH);
label2.caption:= 'load: '+TRAINPATH;

if chkboxdrop.checked then
 NN.EnableDropouts(true) else
   NN.EnableDropouts(false);
pInput:= TNNetVolume.Create0(32, 32, 3, 1);
pOutPut:= TNNetVolume.Create0(10, 1, 1, 1);

LoadPictureIntoVolume(image1.picture, pinput);
pInput.RgbImgToNeuronalInput(csEncodeRGB);
NN.Compute65(pInput,0);
NN.GetOutput(pOutPut);
writeln('result get class type: '+itoa(pOutPut.GetClass()));
```

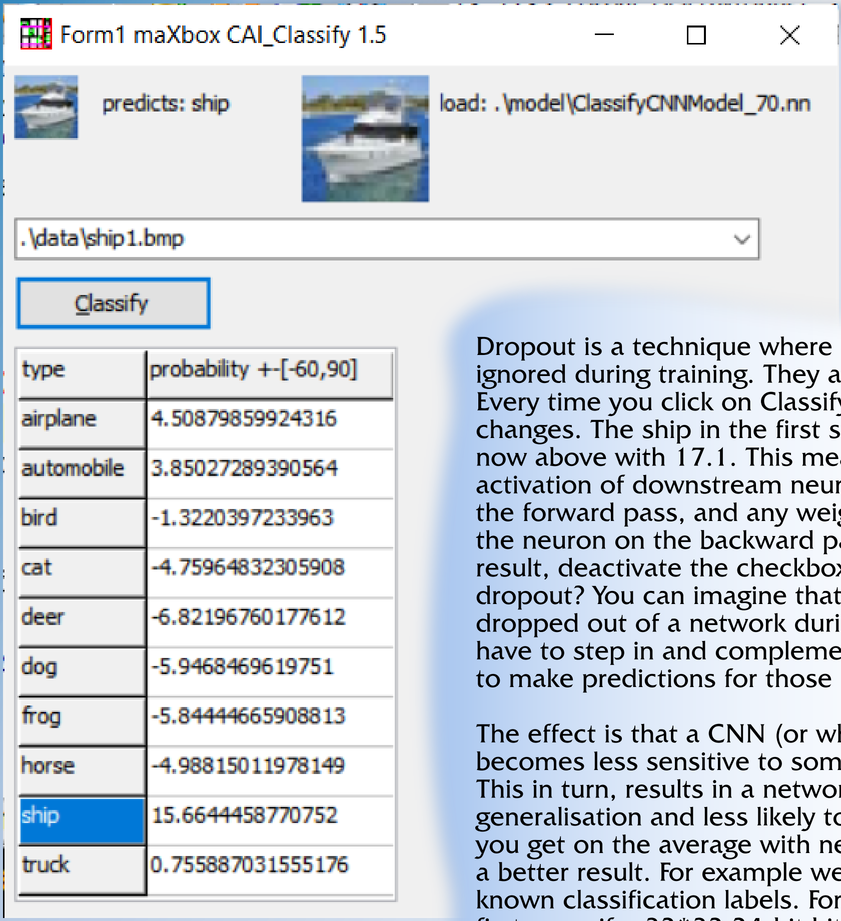
Then, we need to add `RgbImgToNeuronalInput` and with the use of SoftMax, we can print output class probabilities to show in the `Stringgrid`.
The *.nn file in TRAINPATH serves as a pre-trained file (`FAvgWeight`) to classify/predict images we trained on. Also the CIFAR-10 classification examples with `experiments/testcnnalgo/testcnnalgo.lpr` and a number of CIFAR-10 classification examples are available on /experiments.
Imagine the accuracy goes up and the loss-function (error-rate) goes down. The loss function is the bread & butter of modern machine learning; it takes your algorithm from theoretical to practical and transforms matrix multiplication into deep learning.

DROP OUT EXPERIMENTS

It's usually very hard to understand neuron by neuron how a neural network dedicated to image classification internally works.
In this technique, an arbitrary neuron is required to activate and then the same back-propagation method used for learning is applied to an input image producing an image that this neuron expects to see.
Adding neurons and neuronal layers is often a possible way to improve artificial neural networks when you have enough hardware and computing time. In the case that you can't afford time, parameters and hardware, you'll look for efficiency with Separable Convolutions (SCNN).
But there's another for me interesting point, the dropout regularisation. The dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. In our model you can see layer 6 as the dropout:

```
 6) TNNetDropout:2;0;0;0;0;0;0;0
```

We can apply a Dropout layer to the input vector, in which case it nullifies some of its features; but we can also apply it to a hidden layer, in which case it nullifies some hidden neurons.



Form1 maXbox CAI_Classify 1.5

predicts: ship

load: .\model\ClassifyCNNModel_70.nn

.\data\ship1.bmp

type	probability +-[-60,90]
airplane	4.50879859924316
automobile	3.85027289390564
bird	-1.3220397233963
cat	-4.75964832305908
deer	-6.82196760177612
dog	-5.9468469619751
frog	-5.84444665908813
horse	-4.98815011978149
ship	15.6644458770752
truck	0.755887031555176

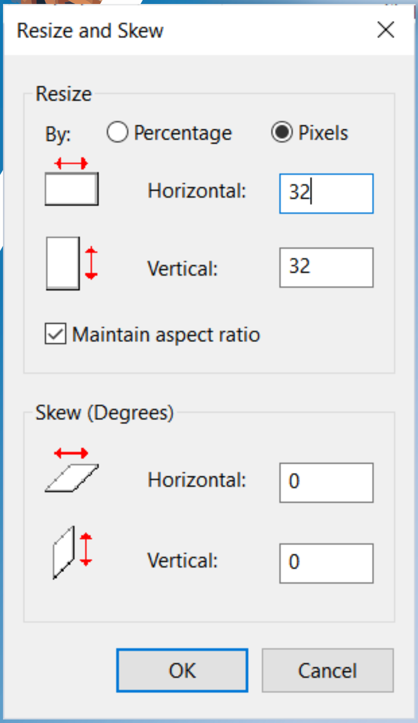
Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped out" randomly. Every time you click on Classify you get another result in small changes. The ship in the first screen is classified with 15.2 now above with 17.1. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass, and any weight updates are not applied to the neuron on the backward pass. If you want a comparable result, deactivate the checkbox. So what's the advantage of dropout? You can imagine that if neurons are randomly dropped out of a network during training, other neurons will have to step in and complement the representation required to make predictions for those missing neurons.

The effect is that a CNN (or whatever deep learning nn) becomes less sensitive to some specific weights of neurons. This in turn, results in a network capable of better generalisation and less likely to specialise training data, means you get on the average with new or in training unseen pictures a better result. For example we take a new picture out of the known classification labels. For this we convert the picture at first as a cifar 32*32 24-bit bitmap:

CLASSIFY CIFAR10

maXbox Starter 105 – Image Classifier with loading and testing a pre-trained model.

Then we load the picture as *.bmp (*just drop in the ./data directory*) and try to classify an unknown class with unseen training, but - and that's sort of surprising - we get a result:



Resize and Skew

Resize

By: ( ) Percentage (●) Pixels

Horizontal: 32

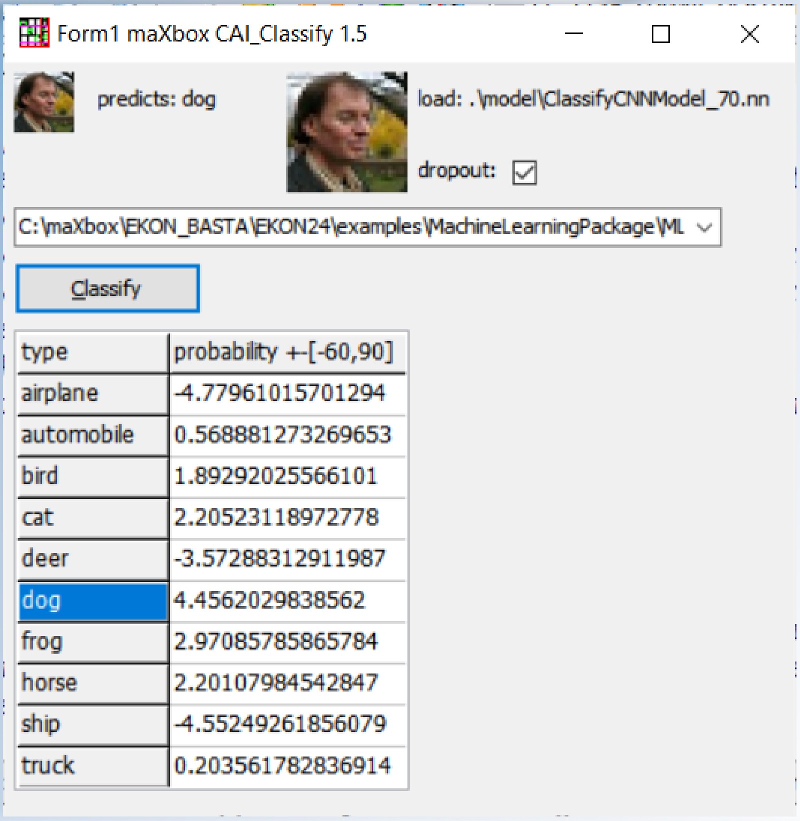
Vertical: 32

☑ Maintain aspect ratio

Skew (Degrees)

Horizontal: 0

Vertical: 0



Form1 maXbox CAI_Classify 1.5

predicts: dog

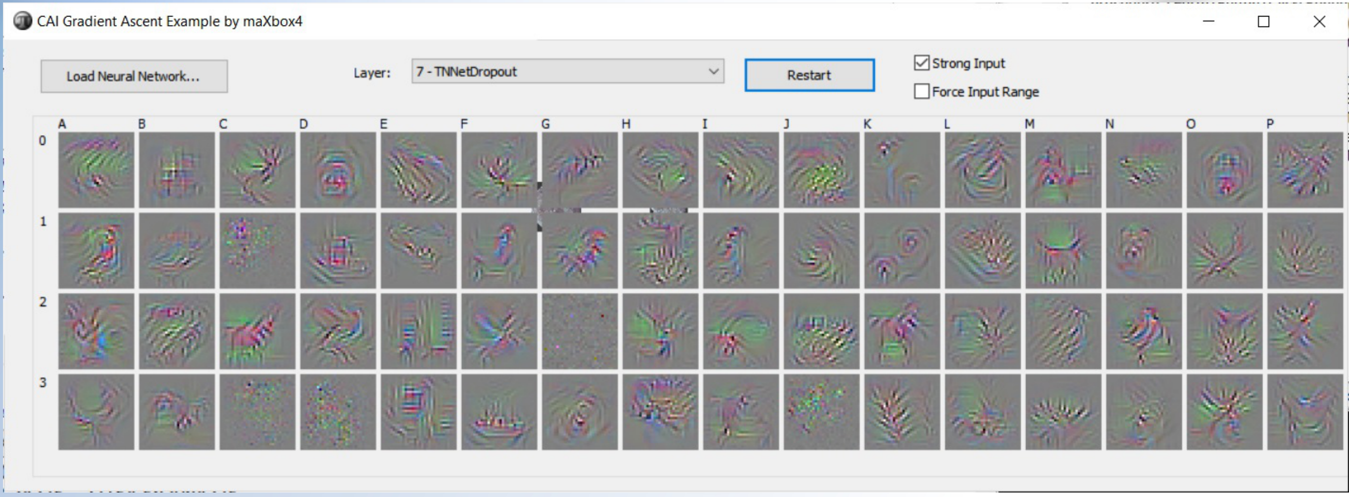
load: .\model\ClassifyCNNModel_70.nn

dropout: ☑

C:\maXbox\EKON_BASTA\EKON24\examples\MachineLearningPackage\ML

type	probability +-[-60,90]
airplane	-4.77961015701294
automobile	0.568881273269653
bird	1.89292025566101
cat	2.20523118972778
deer	-3.57288312911987
dog	4.4562029838562
frog	2.97085785865784
horse	2.20107984542847
ship	-4.55249261856079
truck	0.203561782836914

So the result is devastating and amazing too, somewhat between dog and horse is the kind of bionics! But this can be a baseline for similarities in a recommender system or you can classify the age or sex of a person, enable also in a gender gap research. In our model, a new dropout layer between `TNNetConvolutionReLU` (*activation layer*) and the hidden layer `TNNetMaxPool` was added.
You can also make them visible, but its more art than science or more science than fiction:



CAI Gradient Ascent Example by maXbox4

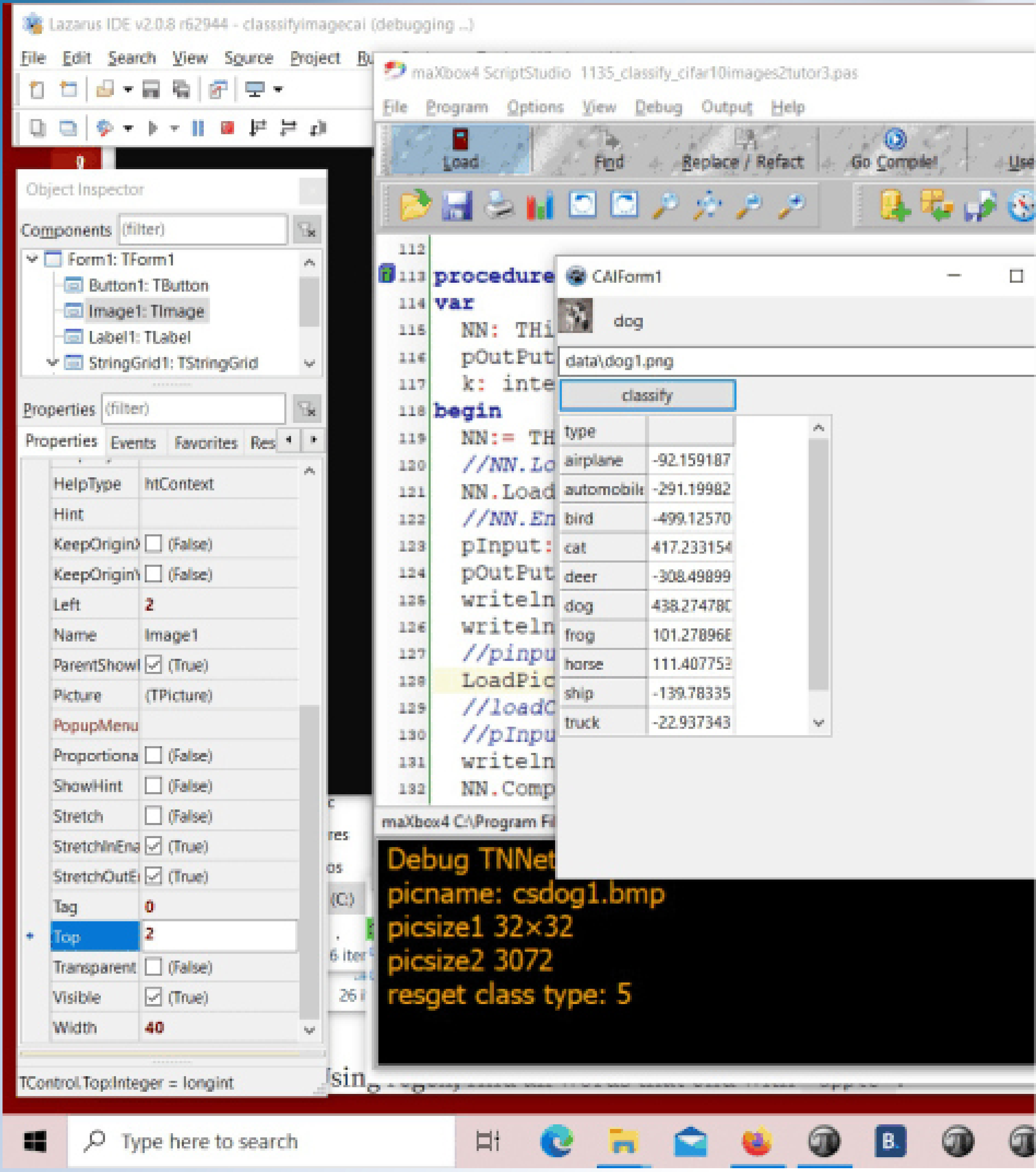
Layer: 7 - TNNetDropout

☑ Strong Input

☐ Force Input Range

This visual technique above used to help with the understanding about what individual neurons represent is called Gradient Ascent. You can find more about gradient ascent at `http://yosinski.com/deepvis.`

In the archive MachineLearningPackage.zip you find the script, model and data you need, which works with Lazarus, Jupyter and maXbox too:


Lazarus IDE v2.0.8 r62944 - classsifyimagecai (debugging ...)
File Edit Search View Source Project Ru
Object Inspector
Components (filter)
Form1: TForm1
Button1: TButton
Image1: TImage
Label1: TLabel
StringGrid1: TStringGrid
Properties (filter)
Properties Events Favorites Res
maXbox4 ScriptStudio 1135_classify_cifar10images2tutor3.pas
File Program Options View Debug Output Help
Load Find Replace / Refact Go Compile! Use
CAIForm1
dog
data\dog1.png
classify
maXbox4 C:\Program Fi
Type here to search

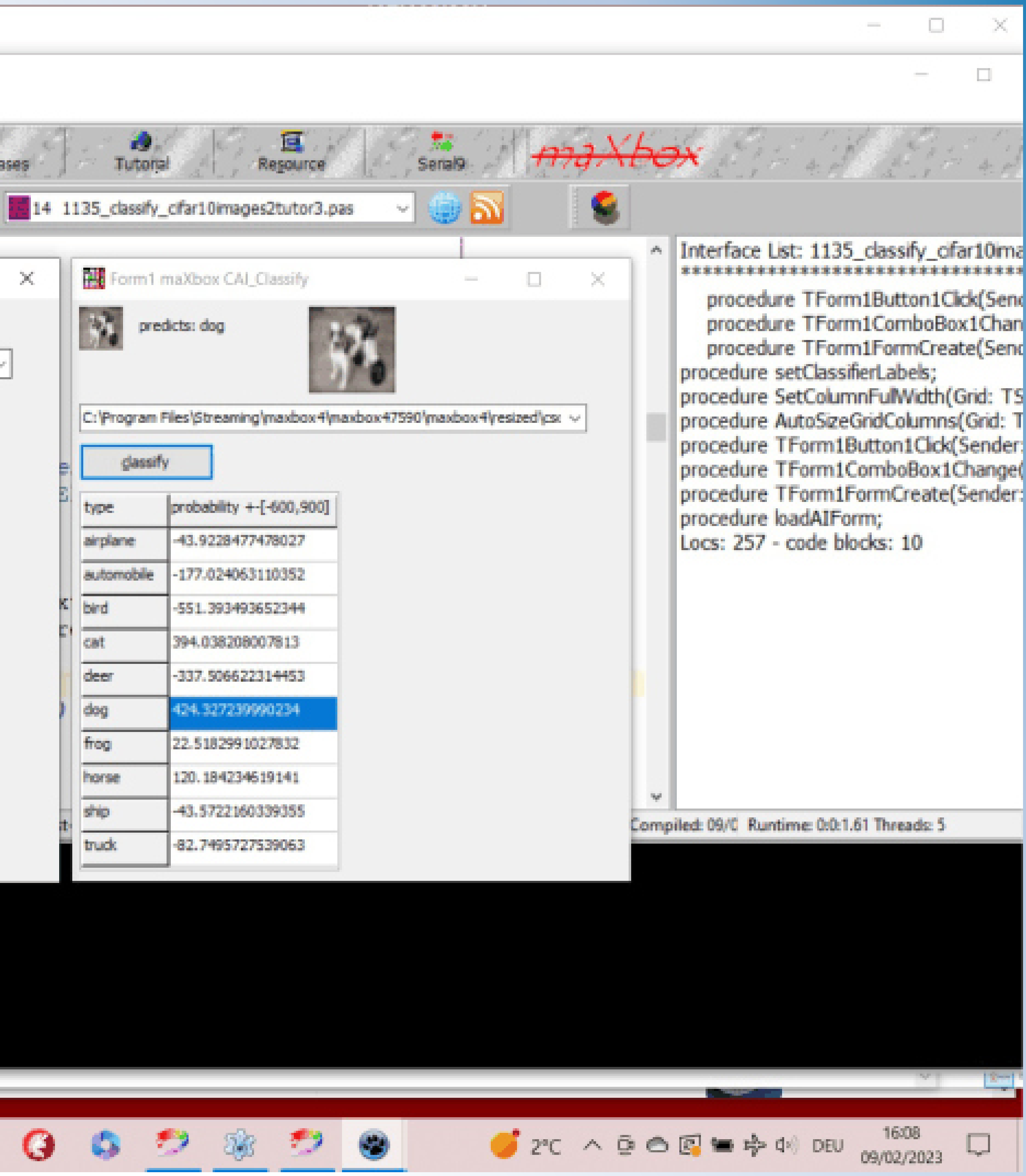
HelpType	htContext
Hint	
KeepOriginX	☐ (False)
KeepOriginY	☐ (False)
Left	2
Name	Image1
ParentShowI	☑ (True)
Picture	(TPicture)
PopupMenu	
Proportiona	☐ (False)
ShowHint	☐ (False)
Stretch	☐ (False)
StretchInEna	☑ (True)
StretchOutE	☑ (True)
Tag	0
Top	2
Transparent	☐ (False)
Visible	☑ (True)
Width	40

TControl.Top:Integer = longint

```
procedure
var
  NN: THi
  pOutPut
  k: inte
begin
  NN:= TH
  //NN.Lo
  NN.Load
  //NN.En
  pInput:
  pOutPut
  writeln
  writeln
  //pinpu
  LoadPic
  //loadC
  //pInpu
  writeln
  NN.Comp
```

type	
airplane	-92.159187
automobil	-291.19982
bird	-499.12570
cat	417.233154
deer	-308.49899
dog	438.27478C
frog	101.27896E
horse	111.407753
ship	-139.78335
truck	-22.937343

```
Debug TNNet
picname: csdog1.bmp
picsize1 32x32
picsize2 3072
resget class type: 5
```



Tutorial
Resource
Serial9
maXbox
14 1135_classify_cifar10images2tutor3.pas
Form1 maXbox CAI_Classify
predicts: dog
C:\Program Files\Streaming\maxbox4\maxbox47590\maxbox4\resized\csx
classify

type	probability +-[-600,900]
airplane	-43.9228477478027
automobile	-177.024063110352
bird	-551.393493652344
cat	394.038208007813
deer	-337.506622314453
dog	424.327239990234
frog	22.5182991027832
horse	120.184234619141
ship	-43.5722160339355
truck	-82.7495727539063

```
Interface List: 1135_classify_cifar10ima
************************************
   procedure TForm1Button1Click(Send
   procedure TForm1ComboBox1Chan
   procedure TForm1FormCreate(Send
procedure setClassifierLabels;
procedure SetColumnFullWidth(Grid: TS
procedure AutoSizeGridColumns(Grid: T
procedure TForm1Button1Click(Sender:
procedure TForm1ComboBox1Change(
procedure TForm1FormCreate(Sender:
procedure loadAIForm;
Locs: 257 - code blocks: 10
```

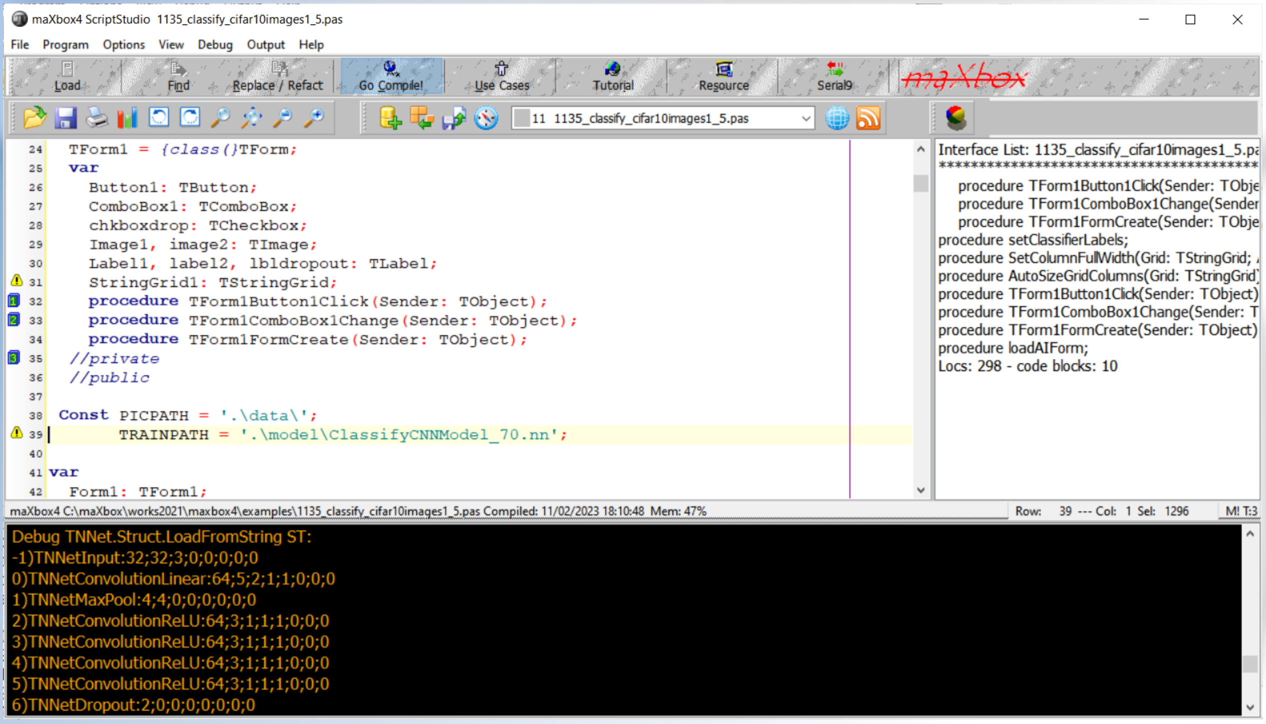
```
Compiled: 09/0  Runtime: 0:0:1.61 Threads: 5
```

The `FormCreate()` can also be triggered with this few lines of code:

```pascal
procedure TForm1FormCreate(Sender: TObject);
var k,t: integer;
 items: TStringList;
begin
 items:= TStringList.create;
 for k:= 0 to 9 do
  StringGrid1.Cells[0, k+1]:= cs10Labels[k];

 //FindAllFiles(ComboBox1.Items, 'csdata');
 FindFiles(exepath+'data', '*.bmp',items);
 writeln(items.text);
 for t:= 1 to items.count-1 do
   ComboBox1.Items.add(items[t]);
 if ComboBox1.Items.Count > 0 then begin
  ComboBox1.text:= ComboBox1.Items[0];
  if FileExists(ComboBox1.text) then begin
   Image1.Picture.LoadFromFile(ComboBox1.text);
   Image2.Picture.LoadFromFile(ComboBox1.text);
   label1.Caption:= extractfilename(ComboBox1.text);
  end;
 end;
end;
```

FindFiles(exepath+'data','*.bmp',items) is an adoption from Lazarus. In the case that an input image isn't 32x32, you can resize (via copying):

`TVolume.CopyResizing(Original: TVolume; NewSizeX,NewSizeY: integer);`
And Given that you have a trained NN, you could call this:
TNeuralImageFit.ClassifyImage(pNN: TNNet; pImgInput,pOutput:TNNetVolume);


maXbox4 ScriptStudio 1135_classify_cifar10images1_5.pas
File Program Options View Debug Output Help
Load Find Replace / Refact Go Compile! Use Cases Tutorial Resource Serial9
11 1135_classify_cifar10images1_5.pas

```pascal
  TForm1 = {class()}TForm;
  var
    Button1: TButton;
    ComboBox1: TComboBox;
    chkboxdrop: TCheckbox;
    Image1, image2: TImage;
    Label1, label2, lbldropout: TLabel;
    StringGrid1: TStringGrid;
    procedure TForm1Button1Click(Sender: TObject);
    procedure TForm1ComboBox1Change(Sender: TObject);
    procedure TForm1FormCreate(Sender: TObject);
  //private
  //public

 Const PICPATH = '.\data\';
       TRAINPATH = '.\model\ClassifyCNNModel_70.nn';

var
  Form1: TForm1;
```

```
Interface List: 1135_classify_cifar10images1_5.pa
*************************************************
   procedure TForm1Button1Click(Sender: TObje
   procedure TForm1ComboBox1Change(Sender
   procedure TForm1FormCreate(Sender: TObje
procedure setClassifierLabels;
procedure SetColumnFullWidth(Grid: TStringGrid;
procedure AutoSizeGridColumns(Grid: TStringGrid
procedure TForm1Button1Click(Sender: TObject)
procedure TForm1ComboBox1Change(Sender: T
procedure TForm1FormCreate(Sender: TObject)
procedure loadAIForm;
Locs: 298 - code blocks: 10
```

```
maXbox4 C:\maXbox\works2021\maxbox4\examples\1135_classify_cifar10images1_5.pas Compiled: 11/02/2023 18:10:48 Mem: 47%    Row: 39 --- Col: 1 Sel: 1296    M! T:3
```

```
Debug TNNet.Struct.LoadFromString ST:
-1)TNNetInput:32;32;3;0;0;0;0;0
0)TNNetConvolutionLinear:64;5;2;1;1;0;0;0
1)TNNetMaxPool:4;4;0;0;0;0;0;0
2)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
3)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
4)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
5)TNNetConvolutionReLU:64;3;1;1;1;0;0;0
6)TNNetDropout:2;0;0;0;0;0;0;0
```

Conclusion:
The neural-API or CAI API (*Conscious Artificial Intelligence*) is some-thing like TensorFlow for Pascal and is platform-independent open source library for artificial intelligence or machine learning in the field of speech recognition, image classification, OpenCL, big data, data science and computer vision2.

To be able to run this example, you'll need to load an already trained neural network file and then select the image you intend to classify.
CAI stores both architecture and weights into the same `*.nn` file!
Dropout is a simple and powerful regularization technique for neural networks and deep learning models.

Loss functions are different based on a problem statement to which deep learning is being applied. The cost function is another term used inter-changeably for the loss function, but it holds a more different meaning. A loss function is for a single training example, while a cost function is an average loss over the complete train dataset.

`https://github.com/joaopauloschuler/neural-api`

`https://sourceforge.net/projects/cai/files/`

`https://github.com/maxkleiner/neural-api`

REFERENCE:

As a Jupyter Notebook:
`https://github.com/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb`

and the same in colab.research:

`https://colab.research.google.com/github/maxkleiner/maXbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb`

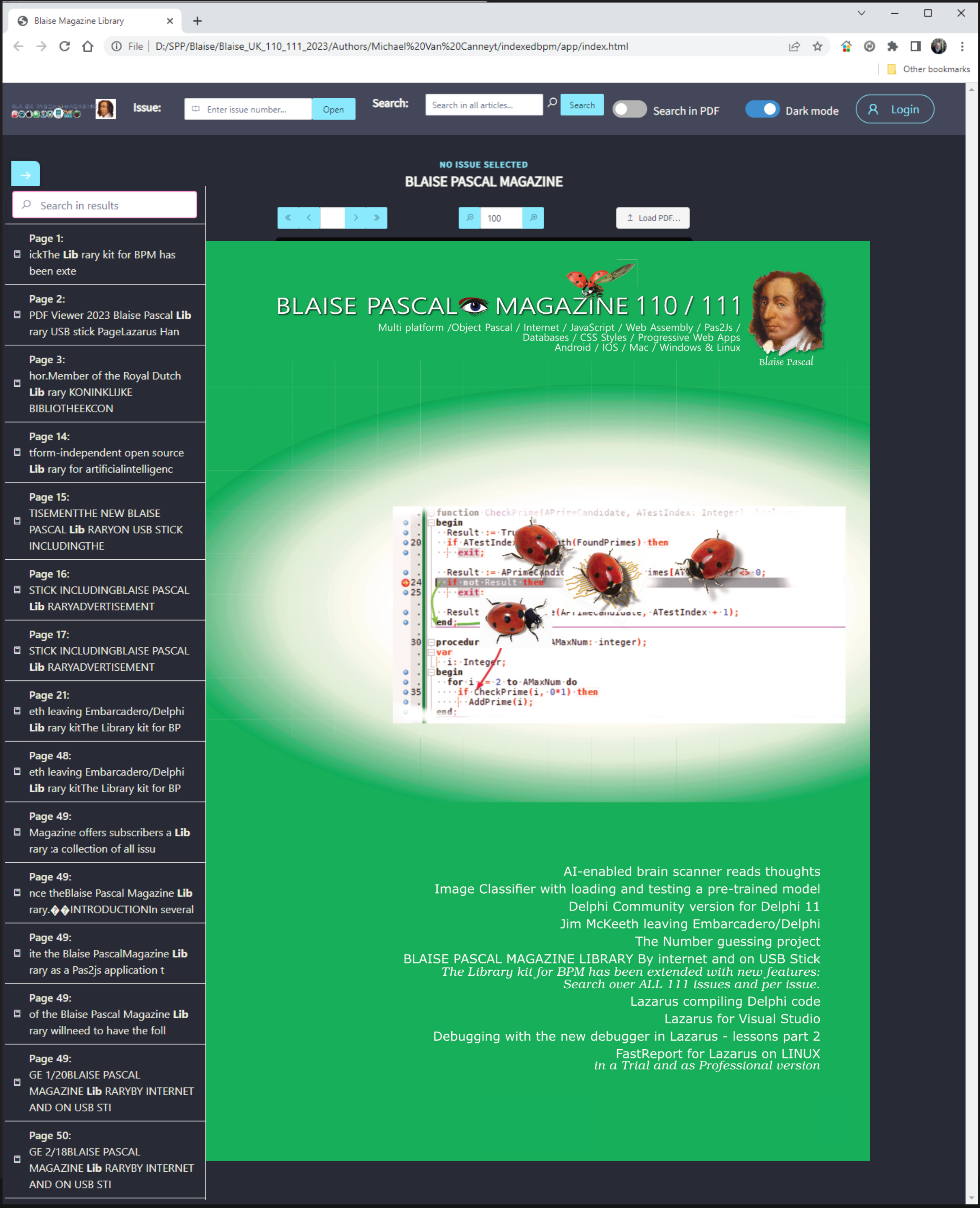
The whole package with app, script, tutorial, data and model:
`https://github.com/maxkleiner/neural-api/blob/master/examples/SimpleImageClassifier/MachineLearningPackage.zip`

Doc and Tool: **`https://maxbox4.wordpress.com`**

`Script Ref: 1135_classify_cifar10images1_5.pas`

THE NEW FUTURE BLAISE PASCAL LIBRARY 2023

ON USB STICK INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK



NO ISSUE SELECTED

BLAISE PASCAL MAGAZINE

Page 1:
ickThe **Lib** rary kit for BPM has been exte

Page 2:
PDF Viewer 2023 Blaise Pascal **Lib** rary USB stick PageLazarus Han

Page 3:
hor.Member of the Royal Dutch **Lib** rary KONINKLIJKE BIBLIOTHEEKCON

Page 14:
tform-independent open source **Lib** rary for artificialintelligenc

Page 15:
TISEMENTTHE NEW BLAISE PASCAL **Lib** RARYON USB STICK INCLUDINGTHE

Page 16:
STICK INCLUDINGBLAISE PASCAL **Lib** RARYADVERTISEMENT

Page 17:
STICK INCLUDINGBLAISE PASCAL **Lib** RARYADVERTISEMENT

Page 21:
eth leaving Embarcadero/Delphi **Lib** rary kitThe Library kit for BP

Page 48:
eth leaving Embarcadero/Delphi **Lib** rary kitThe Library kit for BP

Page 49:
Magazine offers subscribers a **Lib** rary :a collection of all issu

Page 49:
nce theBlaise Pascal Magazine **Lib** rary.��INTRODUCTIONIn several

Page 49:
ite the Blaise PascalMagazine **Lib** rary as a Pas2js application t

Page 49:
of the Blaise Pascal Magazine **Lib** rary willneed to have the foll

Page 49:
GE 1/20BLAISE PASCAL MAGAZINE **Lib** RARYBY INTERNET AND ON USB STI

Page 50:
GE 2/18BLAISE PASCAL MAGAZINE **Lib** RARYBY INTERNET AND ON USB STI

BLAISE PASCAL MAGAZINE 110 / 111

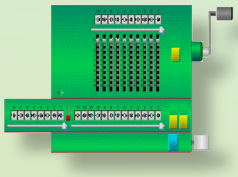
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

Blaise Pascal

```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := Tru
  if ATestInde            th(FoundPrimes) then
    exit;

  Result := APrimeCandic                imes[AT              <> 0;
  if not Result then
    exit;

  Result              e(APrimeCandidate, ATestIndex + 1);
end;

procedur              AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0*1) then
      AddPrime(i);
end;
```

AI-enabled brain scanner reads thoughts
Image Classifier with loading and testing a pre-trained model
Delphi Community version for Delphi 11
Jim McKeeth leaving Embarcadero/Delphi
The Number guessing project
BLAISE PASCAL MAGAZINE LIBRARY By internet and on USB Stick
The Library kit for BPM has been extended with new features:
Search over ALL 111 issues and per issue.
Lazarus compiling Delphi code
Lazarus for Visual Studio
Debugging with the new debugger in Lazarus - lessons part 2
FastReport for Lazarus on LINUX
in a Trial and as Professional version

GUESS:
NUMBER GUESSING PROJECT.

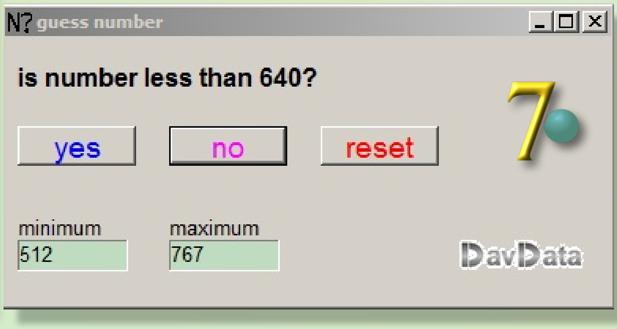


Starter Expert

INTRODUCTION

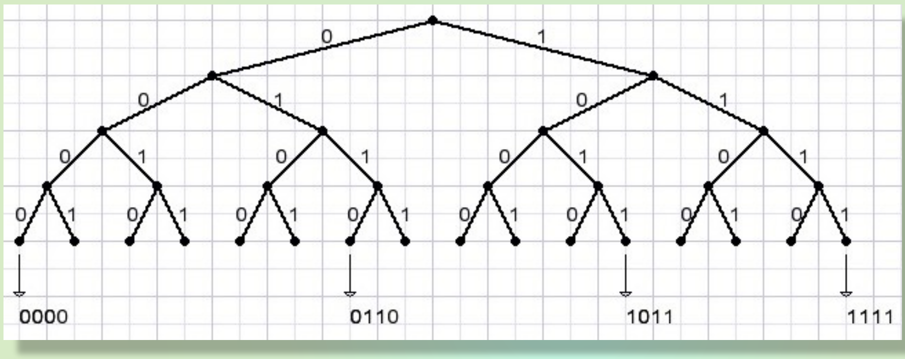
Computers are information processing machines.
Receiving information is the answer of a question.
So, the smallest amount of information is the answer Yes or No.
Calling "No" = 0 and "Yes"=1 we have one binary digit, called BIT.
A bit is the unity of information.

Receiving n bits of information, one per splitting road, we are able to find the way to 2n different destinations.


N? guess number

is number less than 640?

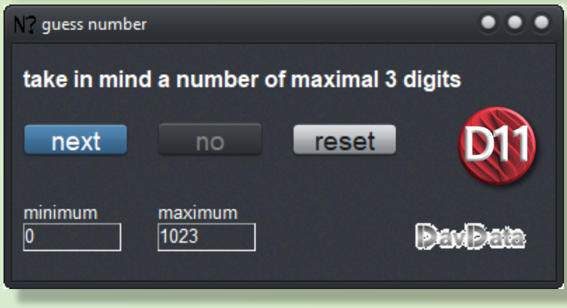
minimum 512
maximum 767



Each bit of information reduces the number of destinations by half.
In general, if the choice is of N1 destinations before and N2 choices after receiving information the number of bits of information I = log2(N1/N2)
From the top of the tree we may reach 16 destinations.
At the bottom this number is 1.
So, we received log2(16/1) = 4 bits information.

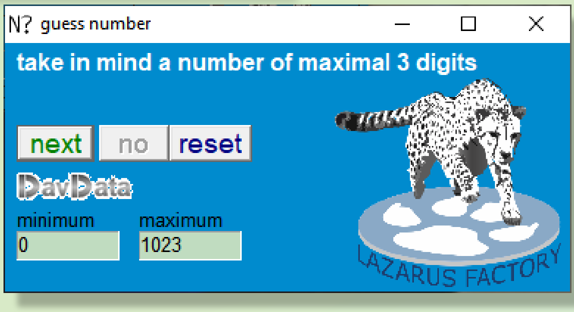
THE PROGRAM

A small program was written to show how information reduces the choices.
The user is requested to take in mind a number less than 1000 (actually less than 1024).
The computer then asks questions to find the number.
At each Yes/No answer of the user the range of numbers is halved.
It takes 10 questions because 210 = 1024.


N? guess number

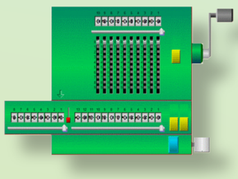
take in mind a number of maximal 3 digits

minimum 0
maximum 1023


N? guess number

take in mind a number of maximal 3 digits

minimum 0
maximum 1023



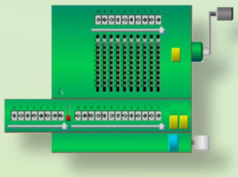
ALGORITHM
At the start the smallest possible number is 0, the largest number is 1023.
base=0, range=1024-0=1024.
At each answer, the range halves.
The question is N ≥ base + range/2.
If true, base = base + range/2.
This repeats until range=1.
Then, the base is the number.
More vivid play
Asking the same type of question is boring.
So the questions are randomly chosen from
N ≥ range + base/2 and
N < range + base/2

Program description
Decisions are taken in procedure gamecontrol
The game is controlled by messages to gamecontrol.

```
 type TGameStatus = (gsStart,gsQuestion,gsEnd);
   TcontrolMessage =(cmStart,cmYes,cmNo,cmNextQuestion,cmEnd);

 var gameState : TGamestatus;
   base,range : word;
   QGR : boolean;
.....
.....

procedure gamecontrol(cm : TControlmessage);
```

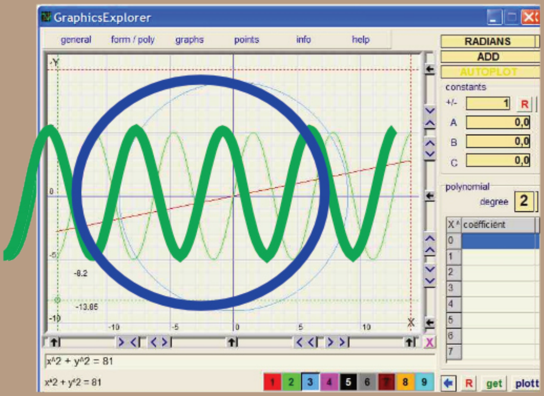
gamecontrol calls procedures procStart, procQuestion(var Q: boolean);
for action.
Q returns the type of question that was asked.
cmYes and cmNo are messages from "Yes" and "No" button clicks.

For details, please refer to the source code.
Delphi 7/11 and Lazarus code is available

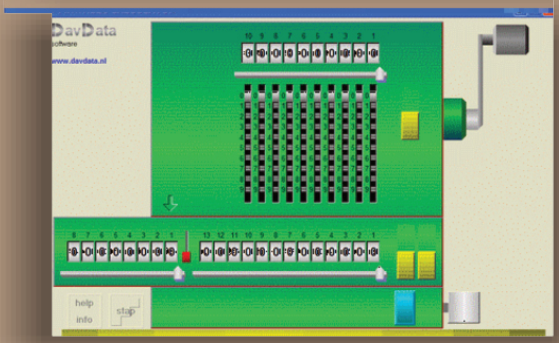


DAVID DIRKSE

including 50 example projects



x^2 + y^2 = 81



COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

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

ADVERTISEMENT

LAZARUS HANDBOOK (PDF)
+SUBSCRIPTION 1 YEAR

- **Lazarus Handbook**
- Printed in black and white
- PDF Index for keywords
- Almost 1000 Pages
- Including 40 Examples
- **Blaise Pascal Magazine**
- English and German
- Free Lazarus PDF Kit Indexer
- 8 Issues per year
- minimal 60 pages
- Including example projects and code

SPECIAL OFFER € 75

+

BLAISE PASCAL MAGAZINE 110
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux
Blaise Pascal

Image Classifier with loading and testing a pre-trained model
Delphi Community version for Delphi 11
Jim McKeeth leaving Embarcadero/Delphi
Library kit
The Library kit for BPM has been extended with new features: Search over ALL 109 issues and per issue.
Lazarus compiling Delphi code
Lazarus for Visual Studio
Debugging with the new debugger in Lazarus - lessons part 2

THE LAZARUS DEBUGGER

Starter Expert

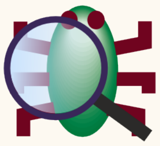
PART 2: NEXT STEPS - STEPPING

FOLLOWING THE CODE

In the previous article we have used breakpoints to pause at specific lines. That gives a rather limited view of just a few snapshots. The debugger offers functionality like running a single line and automatically pausing at the next line without requiring a breakpoint there. This is called stepping. In this article we will explore different methods of stepping.

If we run the example code:

```pascal
program primes;

const
  MAX_NUM = 100;

var FoundPrimes: Array of integer;

procedure AddPrime(APrimeNum: Integer);
var
  l: SizeInt;
begin
  l := Length(FoundPrimes);
  SetLength(FoundPrimes, l+1);
  FoundPrimes[l] := APrimeNum;
end;

function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := True;
  if ATestIndex >= Length(FoundPrimes) then
    exit;

  Result := APrimeCandidate mod FoundPrimes[ATestIndex] <> 0;
  if not Result then
    exit;

  Result := CheckPrime(APrimeCandidate, ATestIndex + 1);
end;

procedure FindPrimes(AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 1) then
      AddPrime(i);
end;

procedure PrintPrimes;
var
  i: Integer;
begin
  for i := 0 to Length(FoundPrimes) - 1 do
    WriteLn(FoundPrimes[i]);
end;

begin
  FindPrimes(MAX_NUM);
  PrintPrimes;
  readln;
end.
```



We would expect the output to begin with:

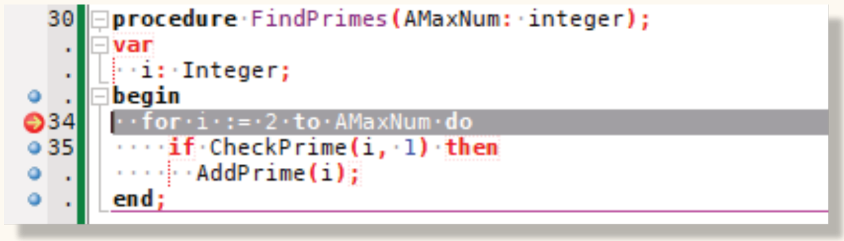
```
2
3
5
7
11
13
```

But instead it starts with a series including the number "4"

```
2
3
4
5
7
11
13
```

STEP OVER

In the last article we used a breakpoint and looked at the data to predict what the code would do.To start the debug session we will set a breakpoint at line 34 and run with F9. (*Figure 1*)

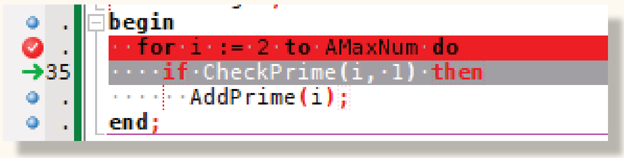


```
30 procedure FindPrimes(AMaxNum: integer);
 . var
 .   i: Integer;
 . begin
34   for i := 2 to AMaxNum do
35     if CheckPrime(i, 1) then
 .       AddPrime(i);
 . end;
```

Figure 1

Once the debugger reached the line the app is paused. Instead of running the program to hit the breakpoint again, we will execute it line by line.

If we press F8 the debugger will "step over". It will execute the current line and pause on the next line. The arrow that had been shown in the red dot of the breakpoint, will now be in front of the next line. (*Figure 2*).



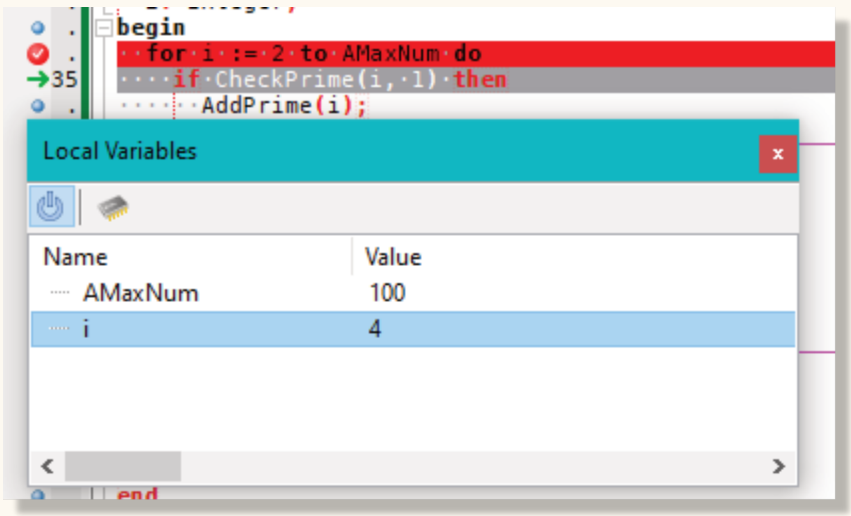
```
 . begin
 .   for i := 2 to AMaxNum do
35     if CheckPrime(i, 1) then
 .       AddPrime(i);
 . end;
```

Figure 2

As the debugger has now executed the "`for i := 2 to AMaxNum`" the variable "`i`" is now initialized, and we can watch its value in the Locals window. (Ctrl-Alt-L).
Line 35 marked as current by the arrow has not yet been executed. We will run that line by pressing F8 again. The debugger will execute the entire line, that is it will run the call to "`CheckPrime`" as a single step. It will not stop inside of "`CheckPrime`".
As "`i`" has the value 2 and that is a prime the next line should be the conditional "then" statement. We can see that this is indeed the case, as the green arrow is now in front of line 36. The next "step over" (F8) will execute "`AddPrime`" and bring us back to the start of the loop. "i" is still 2, it will increase to 3 when we execute the "`for i := " line`.

STEP INTO

We can step through the loop again (using F8), and when we enter the 3rd iteration of the loop "`i`" will become 4. (*Figure 3*)

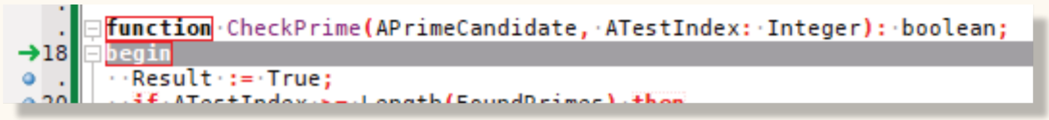


```
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 1) then
      AddPrime(i);
end
```

Name	Value
AMaxNum	100
i	4

Figure 3

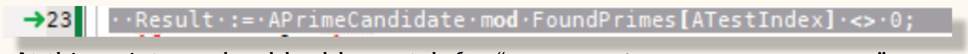
As we have seen in the output, the number 4 will be incorrectly added to the list of primes. This means we need to check what happens in "`CheckPrime`".
Instead of setting a breakpoint (*as we did in the last article*) we will use "step into" (F7) to enter the function.



```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := True;
```

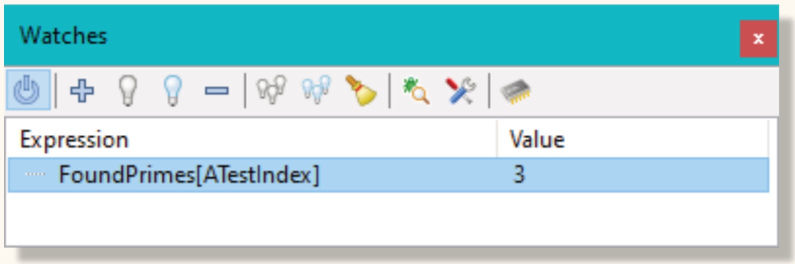
Figure 4

The debugger will step to the first line in "`CheckPrime`" (*Figure 4*).
From there we can step again line by line, using "step over" (F8).
The function "`CheckPrime`" will test recursively if any of the already found primes is a natural divisor of the current `APrimeCanditate`.
If the candidate is not divisible by any of those primes, then it is itself a prime number.
If we use "step over" (F8) we will go over the lines that check if "`ATestIndex`" is within the length of the `FoundPrimes` array.
We will reach the line



```
  Result := APrimeCandidate mod FoundPrimes[ATestIndex] <> 0;
```

At this point we should add a watch for "`FoundPrimes[ATestIndex]`", so we can see what the code is testing. (*Figure 5*)



Expression	Value
FoundPrimes[ATestIndex]	3

Figure 5

Stepping over line 23 the value for "`Result`" will be calculated, and shown in the "locals window". 3 is not a natural divider of 4.
And Result is true since no natural divider has been found. The code so far supposes that 4 may be a prime.

We use further "`step over`" to go over the "`if not result`" to the code line with the recursive call of "`CheckPrime`". Here we use "step into" (F7) to enter the nested function call and see how it continues to check if 4 is prime or not.

The debugger takes us again to line 18 the "`begin`" of "`CheckPrime`". We use "step over" to follow the code execution. This time stepping over the "`if ATestIndex >= Length(FoundPrimes)`" will take us to the "`end`" line of the function. This means the "`then exit`" was executed. But due to code optimizations by the compiler, the "`exit`" line was skipped.

***NOTE** that, if we had compiled the project with* "`optimization level 0 - no optimizations`" *instead of the default "level 1" then the stepping would have included the line 21 "exit".*

So the code returns that all primes had been checked as divisor, and none had divided the current candidate "4". But we only saw it being tested with 3 as divisor. Yet we know that 2 had been found as a prime too.
So we need to inspect our code, why it skipped 2.
In the loop in "`FindPrimes`" we call "`if CheckPrime(i, 1) then`".
The 1 as 2nd param should start the prime checking with the first found prime. However we use the param "`ATestIndex`" as index to the array "`FoundPrimes`" and dynamic array indexes are zero based. So that line should be

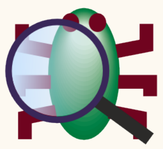
```
 35.  if CheckPrime(i, 0) then
```

If we recompile the project and run it (*after removing the breakpoint*), we will get the correct output. So we have successfully debugged the project using "step over" and "step into".

STEP OUT

We are going to look at a few other methods of stepping.
We wont be finding any more bugs, but we will use the same example code (*with the above fix applied*)
We used "step in" to enter a subroutine. Now lets do the reverse. Lets set a breakpoint on line 24 "`if not Result then`" (after "`Result := APrimeCandidate mod` ...") and run the project.

When we pause at that line we can check locals of the function.
If we hit "run" (F9) several times we will see the locals "`ATestIndex`"=1 and "`APrimeCanditate`"=9.
Now that we have those particular values, we wish to know what the caller of the function will do with the result.
It is possible to hit F8 several times, until we get to the "`end;`" line, and then F8 would take us to the code in the caller. However, we can take an easier route. We can press Shift-F8 ("step out").
This will do the same, it will run the rest of the current function (*including any code called by the current function*), and then stop as soon as execution returns to the caller.

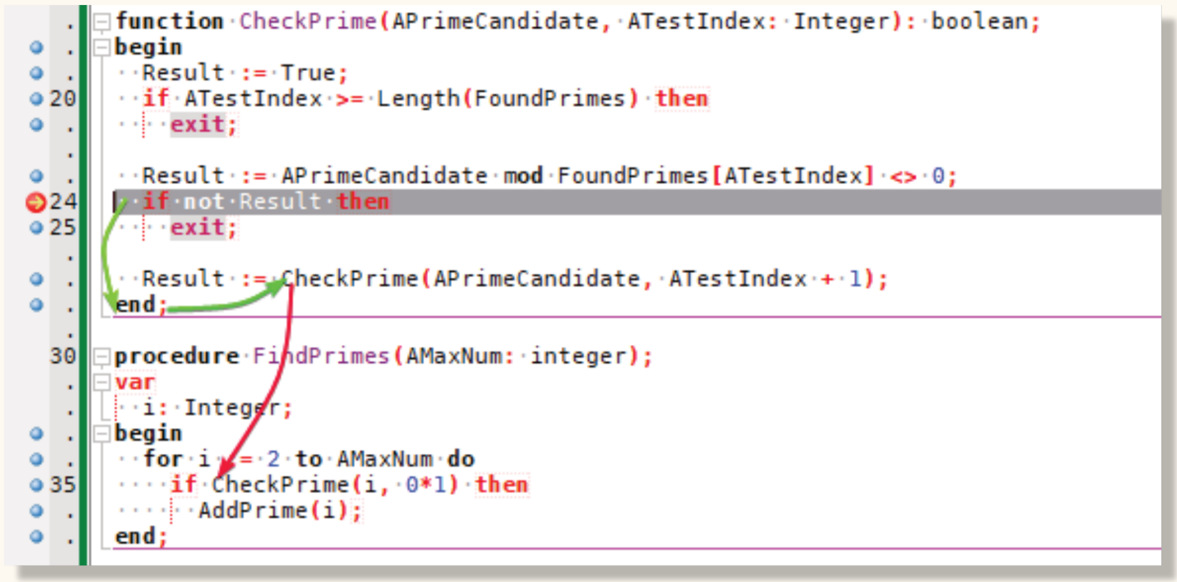


Once we invoked "step out" the debugger will continue to line 27. However, this is line 27 in the calling code. See the green arrows in *Figure 6*, the debugger has run over the nested call to "`CheckPrime`" on line 27, reached the "`end`" on line 28, returned and paused on line 27, which had called the function.

We can also see that we are in the calling code, as the value of "`ATestIndex`" changed from 1 to 0.

Note: depending on the calling code, "step out" may return to the line that contains the call, in this case line 27 that has "CheckPrime(APrimeCandidate, ATestIndex + 1)". Or it may return to the line after the call, which would the "end" at line 28. Both is a valid result.

As we have returned to line 27, that line has not yet been fully executed. The "`result`" variable of the caller may not yet have been set. But as soon as we step to the next line (*using F8*) it will be set, to what the function returned.



```
  . function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
  . begin
  .   Result := True;
 20   if ATestIndex >= Length(FoundPrimes) then
  .     exit;
  .
  .   Result := APrimeCandidate mod FoundPrimes[ATestIndex] <> 0;
 24   if not Result then
 25     exit;
  .
  .   Result := CheckPrime(APrimeCandidate, ATestIndex + 1);
  . end;
  .
 30 procedure FindPrimes(AMaxNum: integer);
  . var
  .   i: Integer;
  . begin
  .   for i := 2 to AMaxNum do
 35     if CheckPrime(i, 0*1) then
  .       AddPrime(i);
  . end;
```

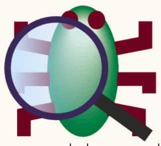
Figure 6

We can now continue to debug in the code of the caller. We can also "step out" again and this would take us to line 35 in "`FindPrimes`". Indicated by the red line in *Figure 6*.

BREAKPOINTS VERSUS STEPPING

There are some differences between running to a breakpoint, and stepping to the next line, into or out of a function.
When you have recursive code, a breakpoint will always pause your code if you get to that line.

```
procedure foo;
begin
  if recurse_done then exit;
  foo();
  writeln;
end;
```



If you are on the line 4 "`foo()`" and you want to go to line 5 "`writeln`" then you can use "step over". This will execute the entire nested (*and sub-nested*) invocation(s) of "`foo`" and pause at line 5 in the current call of "`foo`".
But if you set a breakpoint on line 5 "`writeln`", and run (F9) to the breakpoint, then you will pause inside the (sub)nested call of "`foo`". It is the same line of code, but at a very different context.
The same happens for "step out".
If you are inside a nested call of "`foo`" and at line 4, then step out will run any further nested calls of `foo`, it will run to the end of `foo` at the current level, and then return to the caller. Setting a break point at line 5 (*the line in the caller, at which you would need to pause after returning from the current invocation of "foo"*) will not allow you to do this. It will hit the breakpoint inside the nested call to foo, or if that exits to do "`recurse_done"=true`, then it will just hit the breakpoint on the next line, before ever reaching the "`end`" statement and returning from there.

BREAKPOINTS STOP STEPPING

In all the examples we have either used run (F9) to breakpoint or stepping. It should be mentioned that breakpoints also take precedence over stepping.
If you "step over" a procedure and there is a breakpoint inside that procedure, then the breakpoint will be hit. This means that the step will not run to the line that was next when you invoked the step.
If this happens you can use a series of "step out" to get back to the code were you started.
Similarly "step out" can be interrupted by a breakpoint.
Either if the breakpoint is in any sub-routine that is called while leaving the current routine, or if the breakpoint is in the current routine before the "`end`" is reached.

STEPPING LINES, NOT STATEMENTS

As a language Pascal is statement based. But in the debugger the execution works by `line(s)`.
This can manifest itself in different scenarios. You may have a line with several statements on it.

```
1.    a := 1; b := 2; c := a+b;
```

In this case stepping will execute all 3 statements at once.

You can also have a single statement, that wraps over more that one line;

```
1.                          writeln(
2.                            random(1),
3.                            random(2),
4.                            random(3)
5.                          );
```

In this case, each line can be executed separately. Your project will first run the code for each of the 3 parameters (*calls to "random"*), and once it has the values it will invoke "writeln" with the values. Because FPC evaluates function arguments from right to left (*backwards*), if you step the above code, the first line to execute is "`random(3)`", then "`random(2)`", `random(1)` and finally "writeln".

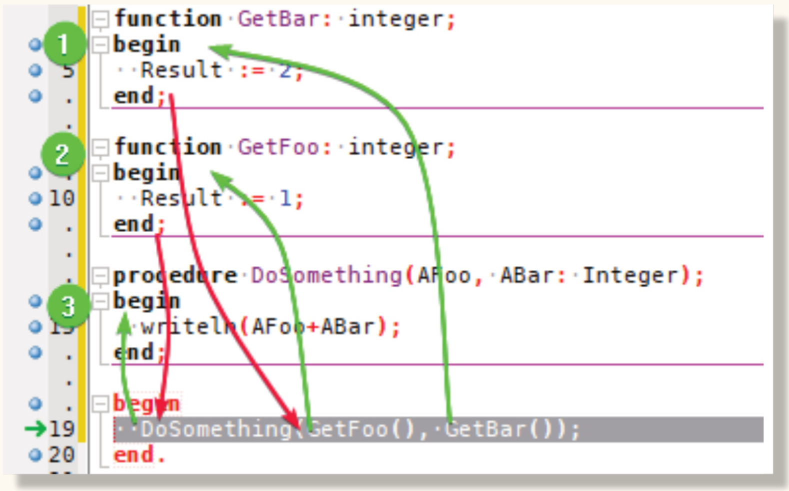
If those were functions in your own code, and they had debug info, you could on each line decide to use “step in” (F7) to go into the functions.

Note that between the execution of the “random” statements, the code may step to the “writeln” line, as it stores the result for the later invocation.

STEP OUT – AND IN AGAIN

“Step out”, that may mean neither by line, nor by statement. Earlier in this article it was mentioned that “step out” could either return to the line that contained the actual calling statement, or it could return to the next line after that statement.
In generally it returns to the middle of the line, right to the position after the call. The remainder of the line can then for example be the assignment of the returned value to a variable. Or it can be further statements, or further calls.

```
1.    DoSomething(GetFoo(), GetBar());
```



```
function GetBar: integer;
begin
  Result := 2;
end;

function GetFoo: integer;
begin
  Result := 1;
end;

procedure DoSomething(AFoo, ABar: Integer);
begin
  writeln(AFoo+ABar);
end;

begin
  DoSomething(GetFoo(), GetBar());
end.
```

Figure 7

If you “step in” on the above line you will enter “GetBar” (remember parameters are evaluated right to left). Now if you “step out” from “GetBar” then you may still want to enter “GetFoo” or “DoSomeThing” or both. If “step out” would go to the next line after the statement, then both of those functions would have run already.
But because “step out” (red arrows) stops in the middle of the line, once you stepped out of “GetBar” you can use “Step in” again and enter “GetFoo” and after that “DoSomething”.
If you just want to enter “GetFoo” then you must step into “GetBar”. But you can then immediately “step out” and “step into” “GetFoo”.

Note that for this to work, you must use “step out”. If you enter “GetBar” and single step to its “end” statement, and then use “step over” to return from it, then the debugger will execute the rest of the calling line, and only stop at the next line after it.

If you do stop through the entire code of “GetBar” and are at its “end” statement, then you can use “step into” (F7) to step into the next routine, which will be “GetFoo”. In that case “Step into” will return to the middle of the calling line, and it will immediately step into the next routine. (On some debugger settings, it may only step out and require a 2nd step into)

"RUN TO CURSOR" OR "STEP OVER TO CURSOR"

Sometimes it may be necessary to step several lines at once. Maybe to leave a loop, that has too many iterations left to run and could not be single stepped with F8. Setting a breakpoint may be possible, but only if it is not recursive.
For this you can use "step over to cursor". Position the text-cursor on the line at which you like to pause. Then press F4. The code will run until it either hits that line, or the current function returns to it caller. The latter is a safety net, if the line was for some reason not reached (*e.g. if it was in a conditional block*). "Step over to cursor" will only pause on that line, if it is reached in the current invocation level.
It will ignore it, if it happens in a recursion.
As "step over to cursor" only pauses the application with the current invocation level, it can not be used to run to any line outside the current function.
In order to do this you can use "run to cursor" from the menu. This is the same as setting a breakpoint, run, and remove the breakpoint. "Run to cursor" will also pause if the line is hit inside recursive calls.

***Note** 2: In Lazarus before versions 2.4 the FpDebug based debugger has an issue, which can in some cases cause "Step out" to pause to early. Depending on the debugger config you use (e.g. Fp-Lldb on Mac) those two commands may not be available.*

SUMMARY

In this article we have explored 3 methods of stepping through the code.

- Step over

Run the code in the current line, and pause at the next line.

- F8
- Menu: Run → Step over
- Toolbutton:

- Step into

If the current code contains a function call, step to the first line of the called routine.
Also act as shortcut for "Step out" + "Step in", if at the "`end`" line of a routine, and another routine is called from the same line than this routine was called from.

- F7
- Menu: Run → Step into
- Toolbutton:

- Step out

Execute the rest of the current routine and pause at the calling line.
Pauses in the middle of the calling line, so further routine calls on that line can be "stepped into".

- Shift-F8
- Menu: Run → Step out
- Toolbutton:

- Step over to cursor

Execute code until it reaches the line where the text-cursor is.
Only reacts in the current function and ignores hits during recursion.
Only available in Lazarus since version 2.2.

- F4
- Menu: Run → Step over to cursor

- Run to cursor

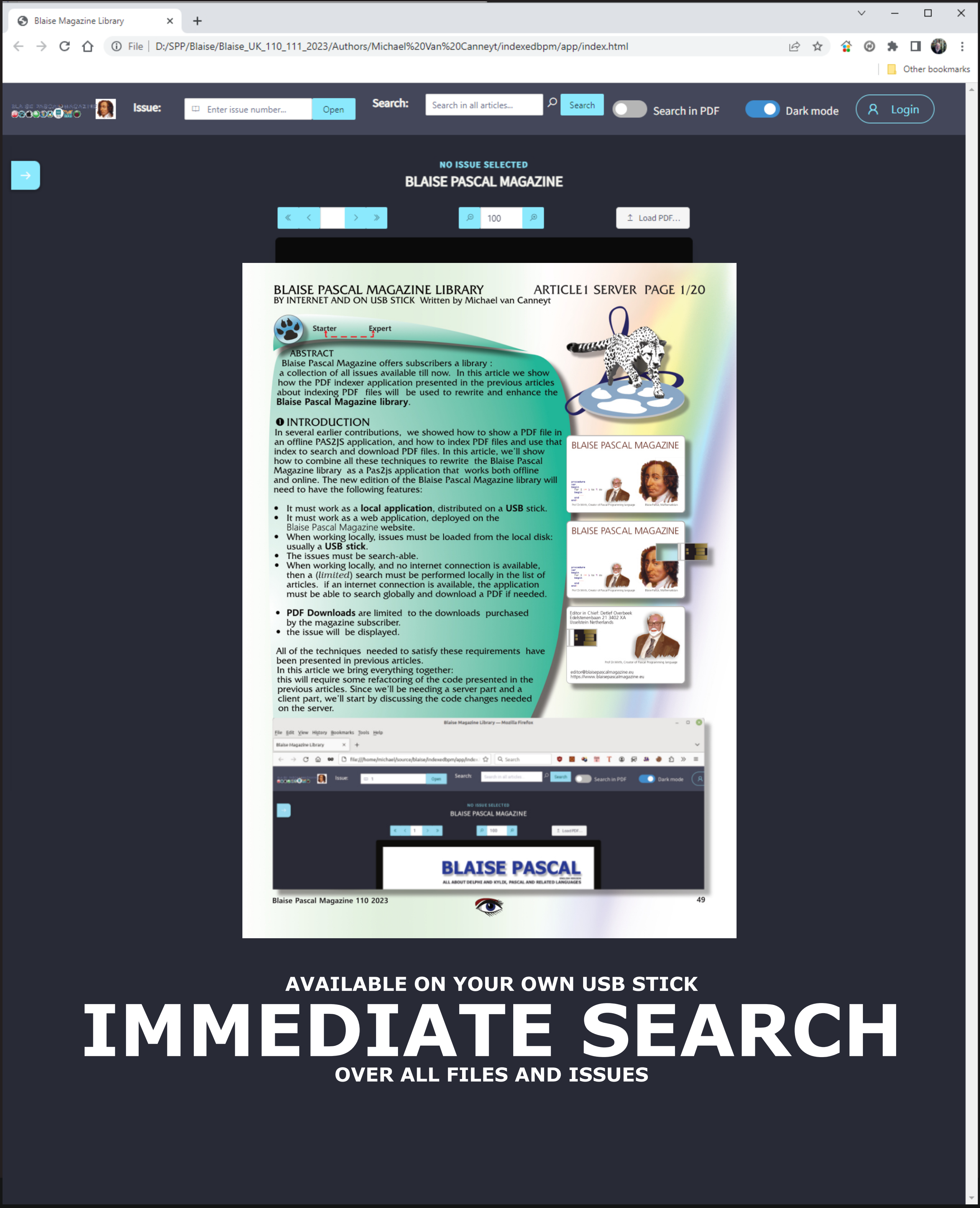
Execute code until it reaches the line where the text-cursor is.
Acts like running to a breakpoint at that line.

- Menu: Run → Run to cursor

The images for tool-buttons have changed between Lazarus versions.
They may differ in your version.

THE NEW FUTURE BLAISE PASCAL LIBRARY 2023

ON USB STICK INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK



BLAISE PASCAL MAGAZINE LIBRARY ARTICLE1 SERVER PAGE 1/20
BY INTERNET AND ON USB STICK Written by Michael van Canneyt

ABSTRACT
Blaise Pascal Magazine offers subscribers a library : a collection of all issues available till now. In this article we show how the PDF indexer application presented in the previous articles about indexing PDF files will be used to rewrite and enhance the **Blaise Pascal Magazine library**.

❶ INTRODUCTION
In several earlier contributions, we showed how to show a PDF file in an offline PAS2JS application, and how to index PDF files and use that index to search and download PDF files. In this article, we'll show how to combine all these techniques to rewrite the Blaise Pascal Magazine library as a Pas2js application that works both offline and online. The new edition of the Blaise Pascal Magazine library will need to have the following features:

- It must work as a **local application**, distributed on a **USB** stick.
- It must work as a web application, deployed on the Blaise Pascal Magazine website.
- When working locally, issues must be loaded from the local disk: usually a **USB stick**.
- The issues must be search-able.
- When working locally, and no internet connection is available, then a (*limited*) search must be performed locally in the list of articles. if an internet connection is available, the application must be able to search globally and download a PDF if needed.

- **PDF Downloads** are limited to the downloads purchased by the magazine subscriber.
- the issue will be displayed.

All of the techniques needed to satisfy these requirements have been presented in previous articles.
In this article we bring everything together:
this will require some refactoring of the code presented in the previous articles. Since we'll be needing a server part and a client part, we'll start by discussing the code changes needed on the server.

BLAISE PASCAL MAGAZINE
BLAISE PASCAL MAGAZINE
Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands
editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu
Blaise Magazine Library — Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///home/michael/source/blaise/indexedbpm/app/index
Issue:
Open
Search:
Search in PDF
Dark mode
NO ISSUE SELECTED
BLAISE PASCAL MAGAZINE
BLAISE PASCAL

Blaise Pascal Magazine 110 2023 49

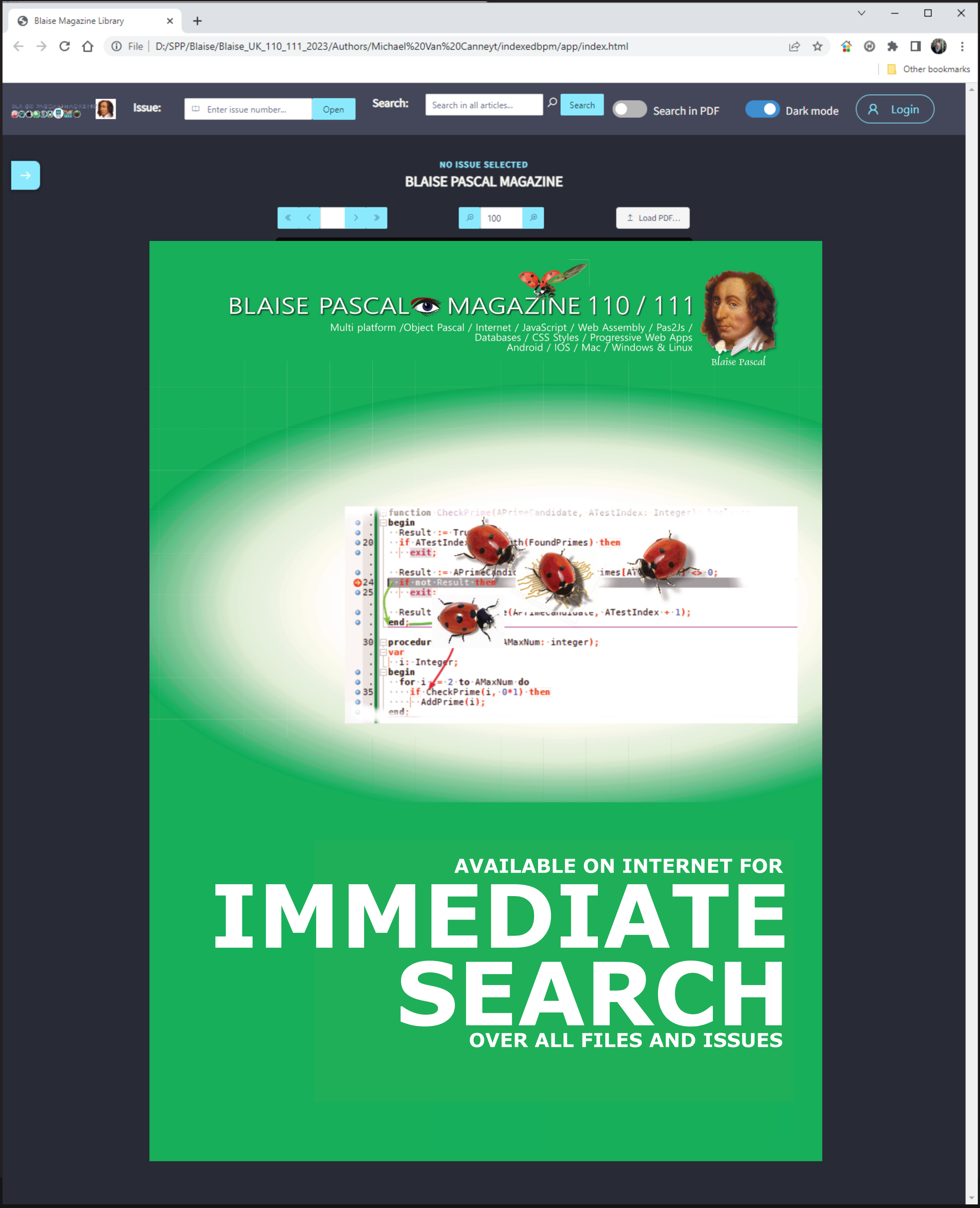
AVAILABLE ON YOUR OWN USB STICK

IMMEDIATE SEARCH

OVER ALL FILES AND ISSUES

THE NEW FUTURE BLAISE PASCAL LIBRARY 2023

ON INTERNET INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK


Blaise Magazine Library
File | D:/SPP/Blaise/Blaise_UK_110_111_2023/Authors/Michael%20Van%20Canneyt/indexedbpm/app/index.html
Other bookmarks
Issue:
Enter issue number...
Open
Search:
Search in all articles...
Search
Search in PDF
Dark mode
Login
NO ISSUE SELECTED
BLAISE PASCAL MAGAZINE
100
Load PDF...
BLAISE PASCAL MAGAZINE 110 / 111
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux
Blaise Pascal
AVAILABLE ON INTERNET FOR
IMMEDIATE
SEARCH
OVER ALL FILES AND ISSUES

```
function CheckPrime(APrimeCandidate, ATestIndex: Integer):
begin
  Result := Tru
  if ATestInde                 th(FoundPrimes) then
    exit;

  Result := APrimeCandic              imes[AT           <> 0;
  if not Result then
    exit;

  Result                    (APrimeCandidate, ATestIndex + 1);
end;

procedur                    AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0*1) then
      AddPrime(i);
end;
```

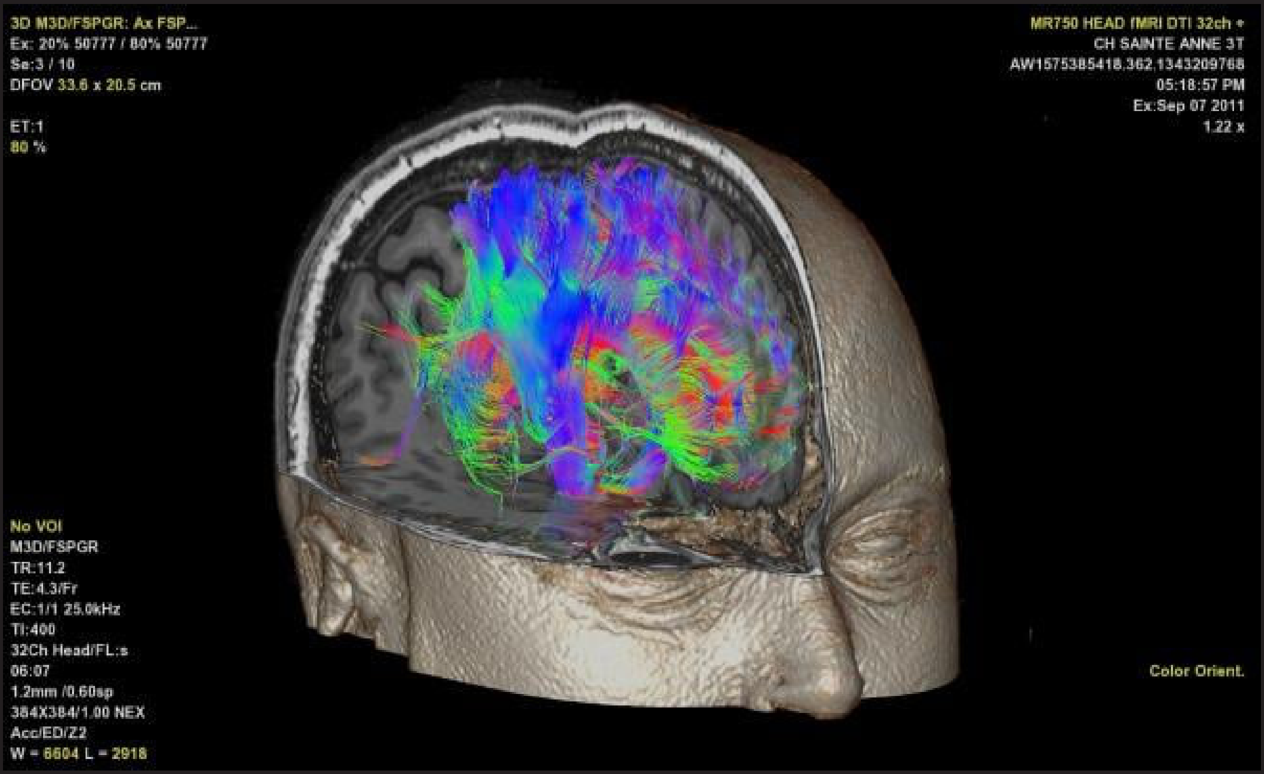


Figure 1 from https://www.itnonline.com/content/machine-learning-uncovers-new-insights-human-brain-through-fmri

A brain scanner can now, at least occasionally, can decode the background voice in your head.
The first non-invasive technique for analysing the content of imagined speech has been developed by researchers. It can offer a potential means of communication for those who are unable to speak.
But how close is the currently available technology, which is only slightly accurate, to realizing true mind-reading?

The brains of no one should be decoded without their consent.

How can policymakers guarantee that these advancements (in the future) aren't misapplied?
The majority of thought-to-speech systems in use today make use of brain implants to track motor cortex activity in users and anticipate the words that their lips are attempting to utter. Alexander Huth and Jerry Tang, computer scientists at the University of Texas at Austin, collaborated with other researchers to combine artificial intelligence (AI) algorithms with functional magnetic resonance imaging (fMRI), a non-invasive method of measuring brain activity, to understand the true meaning behind the thought.
(Functional magnetic resonance imaging or functional MRI measures brain activity by detecting changes associated with blood flow. This technique relies on the fact that cerebral blood flow and neuronal activation are coupled. When an area of the brain is in use, blood flow to that region also increases.)

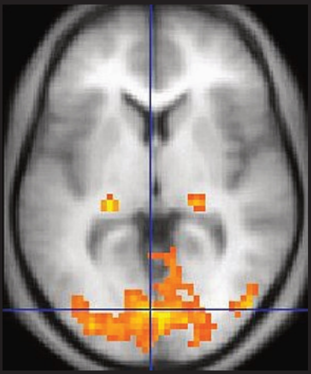


Figure 2 (from Wikipedia)
A fMRI image with yellow areas showing increased activity compared with a control condition.
Purpose Measures brain activity detecting changes due to blood flow.

These algorithms, also known as **LARGE LANGUAGE MODELS (LLM's)**, are what power ChatGPT and are trained to anticipate the next word in a passage of text.
In a study described in *J. Tang et al. Nature Neuroscience 26, 858-866; 2023*, the researchers had three volunteers lie in a fMRI-scanner and record their brain activity while they were listening to podcasts.
The researchers created an encoded map of how each individual's brain responds to various words and phrases by combining this knowledge with the LLM's capacity to comprehend how words relate to one another.

The participants then either listened to a story, pictured telling one, or watched a silent movie while the researchers recorded the fMRI activity. The researchers then attempted to decode this new brain activity using a combination of the patterns they had previously encoded for each individual and algorithms that figure out how a sentence is likely to be constructed based on other words in it.

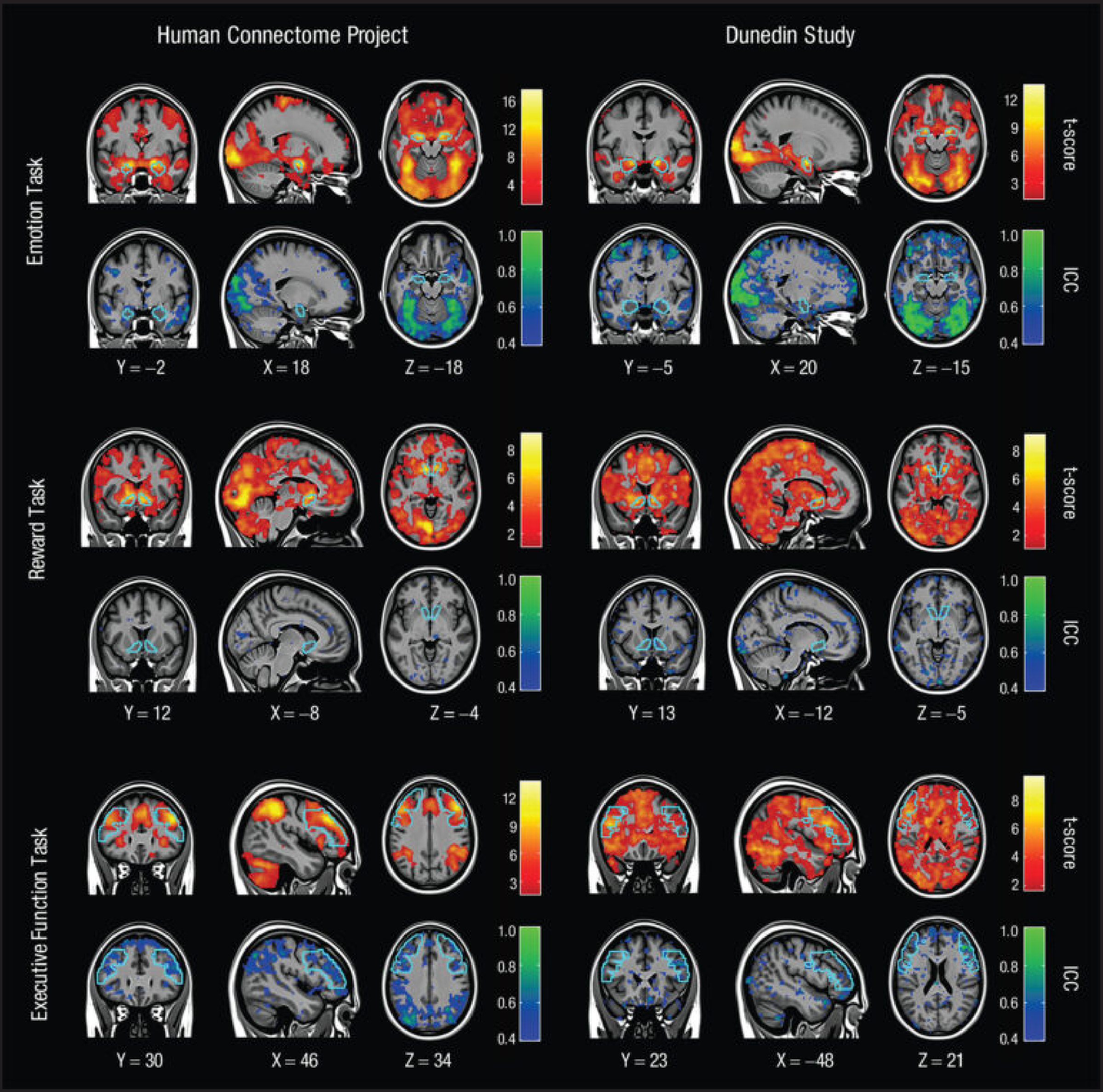


Figure 3 Brain scans showing MRI mapping for 3 tasks across 2 different days. Warm colors show how the results hold up in groups. Cool colors show how results are less reliable person to person. (Annchen Knodt/ Duke University)

Additionally, it performed a decent job of accurately explaining what viewers were seeing in the movies. But many of the sentences it came up with were wrong. The researchers also discovered that the technology was simple to swindle. The decoder was unable to discern the words participants were hearing when they imagined a different story while listening to a recorded one. Additionally, the encoded map varied between people, making it impossible for the researchers to develop a universal decoder.

WAKE UP CALL

On whether the most recent development poses a threat to mental privacy, NEUROETHICISTS disagree. BIOETHICIST **Gabriel Lázaro-Muoz** from **Harvard Medical School in Boston, Massachusetts**, said:
"I'm not calling for panic, but the development of sophisticated, non-invasive technologies like this one seems to be closer on the horizon than we expected. think it's a major wake-up call for the public and policymakers,

Adina Roskies, a science philosopher at **Dartmouth University in Hanover, New Hampshire**, disagrees, claiming that the technology is currently too incorrect and difficult to use to be a threat.
First of all, because fMRI machines are not portable, it is challenging to scan someone's brain without their consent. She also questions whether training a decoder for an individual would be worthwhile in terms of both time and money if the goal were something other than regaining communication skills.

It's not the time (yet) to start worrying. There are numerous additional ways that the government has a variety of additional ways to learn what we are thinking. It is encouraging to **Greta Tuckute**, a cognitive NEUROSCIENTIST at the **Massachusetts Institute of Technology in Cambridge**, since people can readily fool the decoding system by thinking of other things and that it cannot be applied across individuals. It's a fantastic illustration of the degree of agency we actually possess, she says. Do so with caution.

Roskies warns that issues can develop if attorneys or judges utilise the decoder without being aware of its technical limits. For instance, the sentence "I just jumped out [of the car]" was translated into "I had to push her out of the car" in the current study. The differences are so glaring that they could significantly alter the outcome of a legal case.

Tang stated at a press conference that **"the polygraph is not accurate but has had negative consequences."** The brains of no one should be decoded without their consent. He and Huth urged authorities to aggressively address the legalities of using mind-reading technologies.

According to **Lázaro-Muoz**, this regulation might be modelled after a US statute that prohibits insurers and employers from utilising genetic information for discriminatory purposes. He is especially concerned about the effects of the decoder on persons who may have intrusive, unwelcome ideas about harming others even though they would never take such action.

A NEUROSCIENTIST at the US NATIONAL INSTITUTE OF MENTAL HEALTH in Bethesda, Maryland, named **Francisco Pereira** thinks it's unclear how precise decoders will get or whether they'll ever become universal rather than person-specific. Though the decoder might ultimately improve at foretelling the following word in a series, it might have trouble deciphering metaphors or sarcasm.
Putting words together and understanding how the brain encodes the relationships between them are two very different things, according to Pereira."

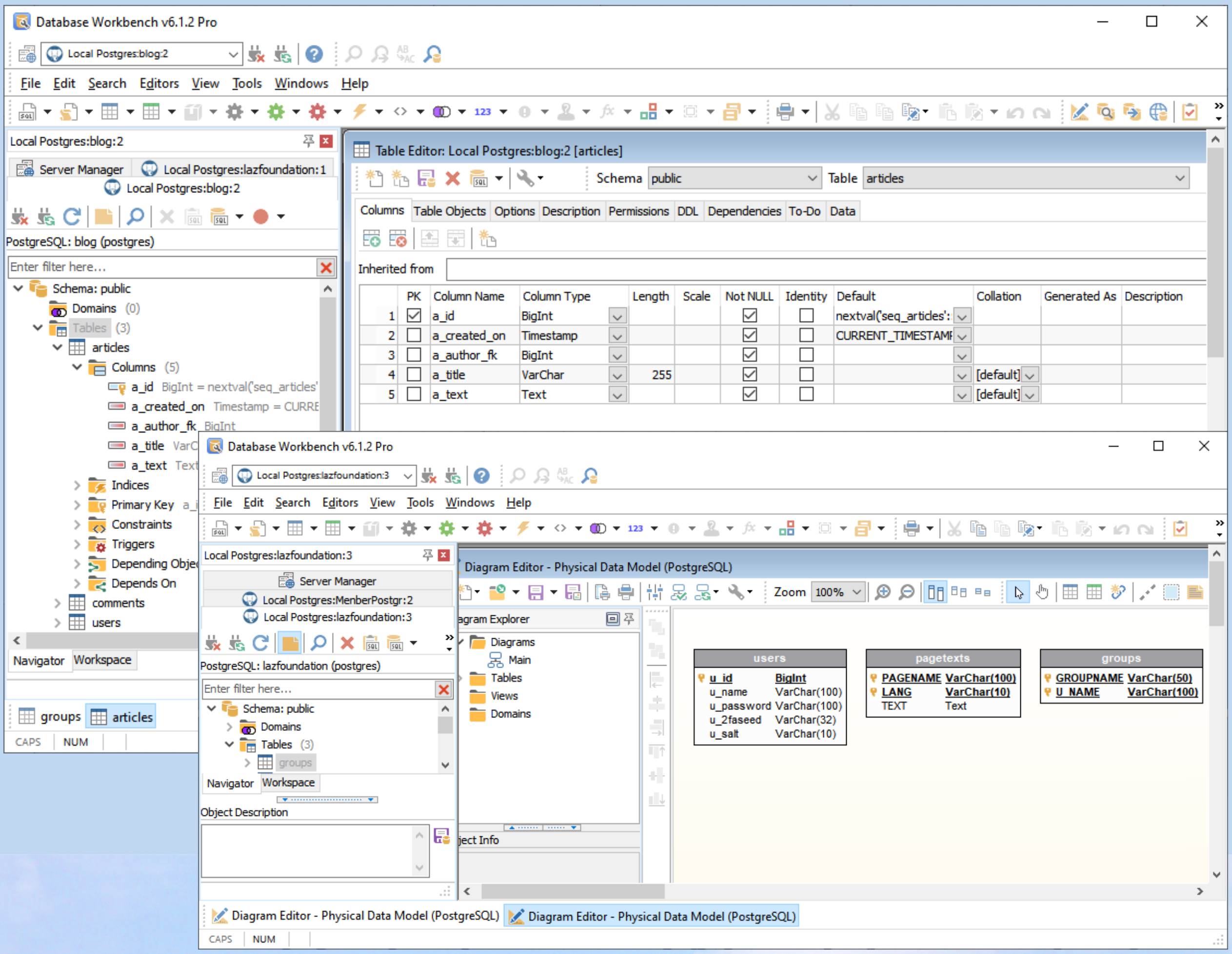


Table Editor: Local Postgres:blog:2 [articles]

Schema public Table articles

	PK	Column Name	Column Type	Length	Scale	Not NULL	Identity	Default	Collation	Generated As	Description
1	✓	a_id	BigInt			✓	☐	nextval('seq_articles':			
2	☐	a_created_on	Timestamp			✓	☐	CURRENT_TIMESTAMP			
3	☐	a_author_fk	BigInt			✓	☐				
4	☐	a_title	VarChar	255		✓	☐		[default]		
5	☐	a_text	Text			✓	☐		[default]		

Diagram Editor - Physical Data Model (PostgreSQL)

users	
u_id	BigInt
u_name	VarChar(100)
u_password	VarChar(100)
u_2faseed	VarChar(32)
u_salt	VarChar(10)

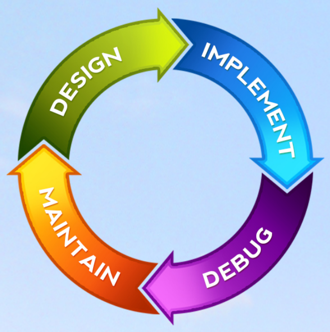
pagetexts	
PAGENAME	VarChar(100)
LANG	VarChar(10)
TEXT	Text

groups	
GROUPNAME	VarChar(50)
U_NAME	VarChar(100)

Introducing

Database Workbench 6

database development environment



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

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

Upscene *Database tools for developers*

www.upscene.com

USING THE DELPHI COMPILER IN THE LAZARUS IDE

By Michaël Van Canneyt

ABSTRACT
The Lazarus IDE has some advantages over the Delphi IDE. For one thing, it works cross-platform: you can use Lazarus on your Mac or on Linux.
So can you use it to work on your Delphi project and compile the result with the Delphi compiler? In this article, we show how.

❶ INTRODUCTION

The Delphi IDE is suitable for windows only.
Lazarus can be used on many platforms. So what if you wish to work on your delphi code on your mac, but still compile with Delphi?

Theoretically, you can: the Delphi command-line compiler can easily be executed in wine, a platform that provides a windows-compatible API on UNIX systems, which can run windows binaries directly on LINUX (*and on MAC*).
Although it should be noted that not all versions of wine are suitable to run the Delphi command-line compiler (*let alone the full IDE*).

Even on windows, there can be reasons to prefer using the Lazarus IDE over the Delphi IDE: The Lazarus IDE is faster, and has superior code tools compared to the Delphi IDE: the code completion is far better.
You could edit in lazarus, and execute the delphi command-line compiler in a terminal or console window.

But Lazarus can do better. While it is of course primarily designed for pascal code, it can in fact also be used to edit and compile C code or JAVASCRIPT - or indeed any compiler.

Lazarus offers an API to help you call a compiler and analyse the output of that compiler. For example, there is a package that can analyse the output of the GCC (GNU C Compiler) compiler and jump to the right place in the code on GCC error messages.

❷ THE DELPHI TOOL

Delphi comes with a command-line compiler.
Can this API not be used to execute the Delphi compiler ?
The answer is: yes, of course !

Since some time, there is a package that does exactly that:
the Delphitool package.

It is in the Lazarus source tree, and will be shipped with the next major release. It allows you to call the Delphi compiler,
and will analyze the output messages of the Delphi compiler,
and display them in the messages window
in a manner that allows you to click on the message
and the IDE will jump to the relevant source location.

The package goes further than that:
optionally, it will take the compiler command-line options used for the FPC compiler and convert them to equivalent Delphi command-line options: paths for units, generation of debug information or optimizations are just some of the options that are converted to delphi options.

The options are written to a configuration file, and you can use this configuration file in your command-line to invoke the Delphi compiler.

Since the only practical way to execute the Delphi command-line compiler on Linux or mac is to use Wine, the Lazarus IDE will offer you to convert the filenames in output messages from Windows notation to Unix notation: it will map drive letters to directories on your Linux or mac machine, and changes backslashes to slashes. It will analyze the wine drive letter mapping to be able to correctly map the paths.

Installing this package is done in the usual manner:

The package is in Lazarus' source tree, in the development branch of the lazarus git repository. It is located in the `components/compilers/delphi` directory, and the package file is `lazdelphi.lpk`.

If you don't have a git version of Lazarus on your system, you can also simply download the files for this package from Gitlab and install the package in an older IDE:
the API's used by this package exist for quite some time.

Open the package file lazdelphi.lpk, and install it with the '**Use - install**' context menu. After rebuilding the IDE and restarting the IDE you should have new pages in the IDE tools - options dialog and in the project options dialog.

3 IDE CONFIGURATION

Before you can use the Delphi compiler, you must configure the Lazarus IDE. In the **Tools** - **Options** dialog, there is a new frame '**Delphi compiler**', *see figure 1 on page 3 of this article*. On this page, the following settings are available:

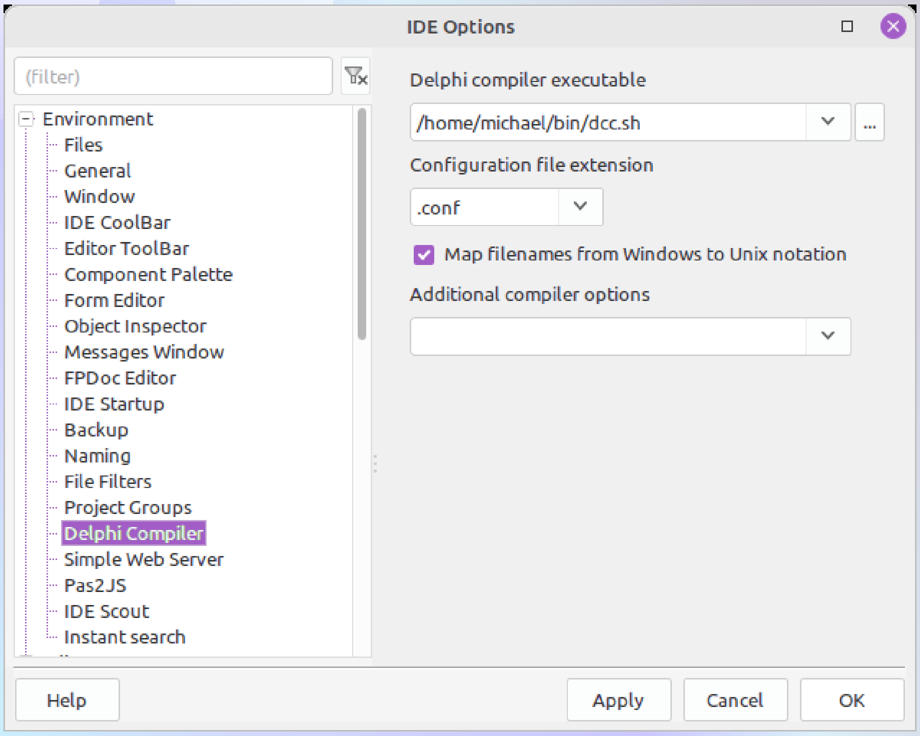
Delphi compiler executable
This is the full path of the delphi command-line compiler.

Delphi configuration file extension
When the IDE generates a configuration file for the Delphi commmand-line compiler, it will use the same name as the Lazarus project file, but with the extension indicated here.

Map filenames from Windows to Unix notation
This option is only available on Linux or mac. When checked, the IDE will convert all filenames in the output of the Delphi compiler to UNIX notation, and will replace drive letters assigned by Wine with the correct directory on your UNIX system.

Additional compiler options
these options will be passed on the command-line to the compiler in addition to the configuration file, for every project you wish to compile.

On Linux and MacOS, the compiler can be called through wine. Distributed with the delphi tool is a script dcc.sh that handles this for you:
it will transform paths, and will make the generated binary executable *(something which the delphi compiler does not do)*. You can point the IDE to this script instead of wine and the actual DCC binary.
The script can also be used on the command-line, it is not specific to Lazarus.



Delphi compiler executable: /home/michael/bin/dcc.sh

Configuration file extension: .conf

☑ Map filenames from Windows to Unix notation

Additional compiler options:

Figure 1: The delphi tool configuration page

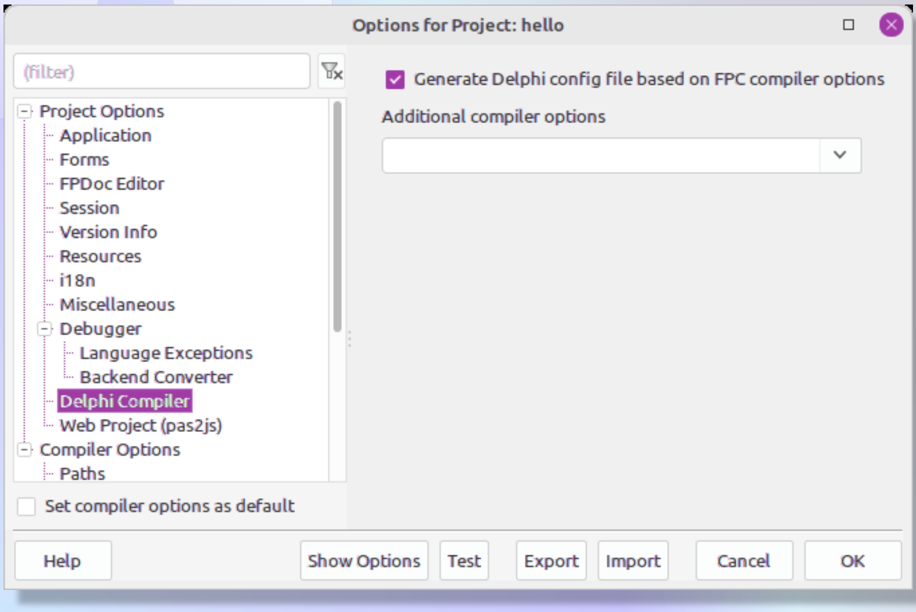
4 PROJECT CONFIGURATION

In order to use the Delphi compiler for a project, you need to configure the project for this. This is of course done in the project options dialog. There are 2 steps to configure. The first is general and can be configured in the frame '**Delphi compiler**', *see figure 2 on page 4 of this article* - options can be set here:

Generate Delphi config file based on FPC compiler options when checked, the IDE will - on every compile - generate a configuration file with the current options from the FPC compiler configuration.

Additional compiler options
these options will be passed on the command-line to the compiler in addition to the configuration file, only for this project.
The second configuration is to adjust the compiler call. This is done in the 'Compiler commands' frame, the last frame under the compiler options. See figure 3 on page 4 of this article things must be done:

1. In the 'Execute before' box, enter the delphi compile command you wish to execute when giving the "compile" or "build" commands and check the 'compile' 'build' and 'run' check boxes. More about the command below.
2. In the 'Execute before' box , check the 'Delphi compiler' in the list of Parsers. This tells the IDE that it should parse and analyse the output of the compile command with the 'Delphi compiler' parser tool. This tool has been registered by the 'lazdelphi' package.
3. In the 'Compiler' box, uncheck the 'compile' 'build' and 'run' checkboxes.



Options for Project: hello

(filter)

- Project Options
  - Application
  - Forms
  - FPDoc Editor
  - Session
  - Version Info
  - Resources
  - i18n
  - Miscellaneous
  - Debugger
    - Language Exceptions
    - Backend Converter
  - Delphi Compiler
  - Web Project (pas2js)
- Compiler Options
  - Paths

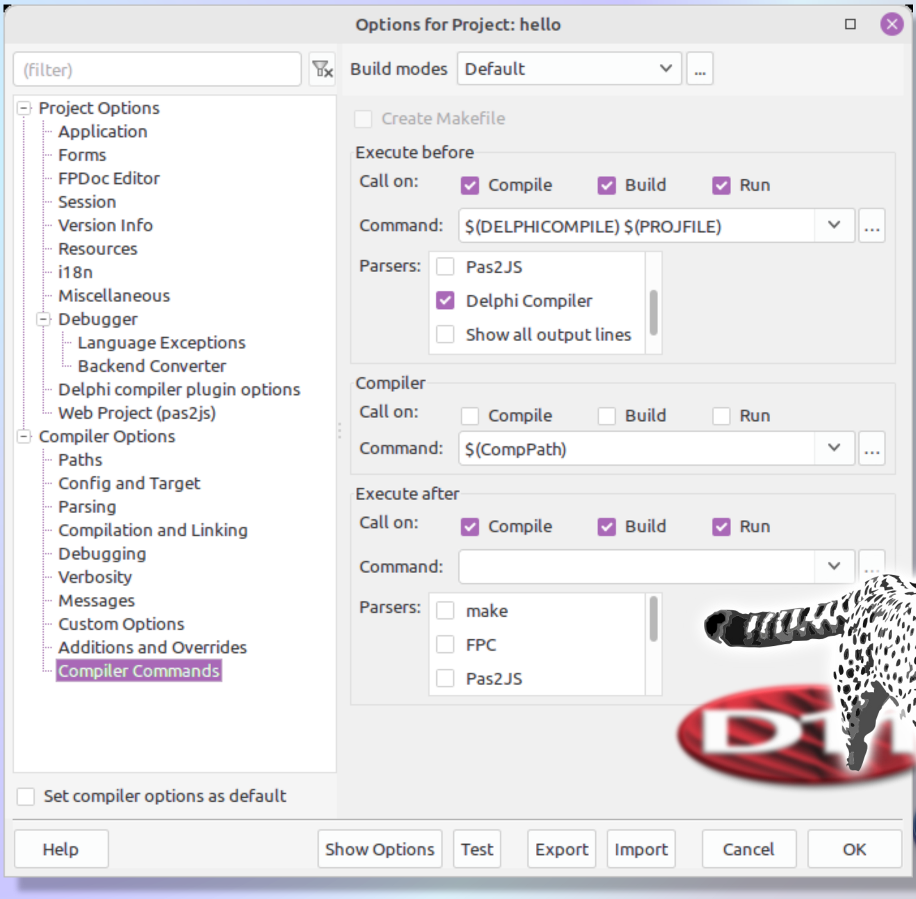
☐ Set compiler options as default

☑ Generate Delphi config file based on FPC compiler options

Additional compiler options

Help | Show Options | Test | Export | Import | Cancel | OK

Figure 2: The project delphi configuration page



Options for Project: hello

(filter)

- Project Options
  - Application
  - Forms
  - FPDoc Editor
  - Session
  - Version Info
  - Resources
  - i18n
  - Miscellaneous
  - Debugger
    - Language Exceptions
    - Backend Converter
  - Delphi compiler plugin options
  - Web Project (pas2js)
- Compiler Options
  - Paths
  - Config and Target
  - Parsing
  - Compilation and Linking
  - Debugging
  - Verbosity
  - Messages
  - Custom Options
  - Additions and Overrides
  - Compiler Commands

☐ Set compiler options as default

Build modes: Default

☐ Create Makefile

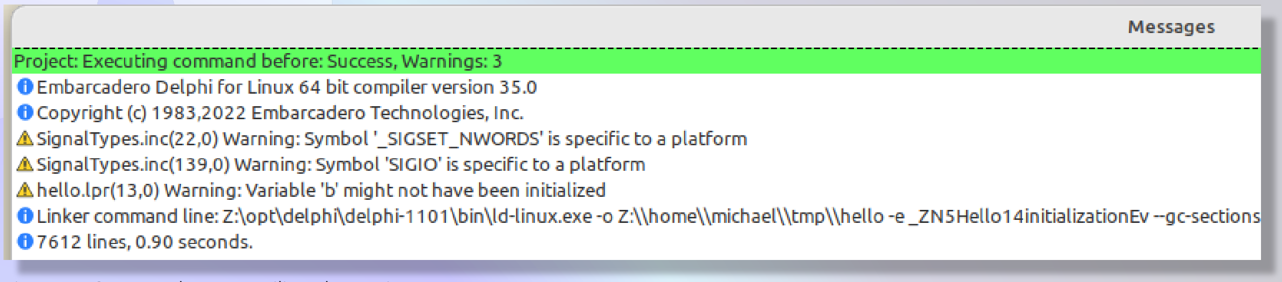
Execute before
Call on: ☑ Compile ☑ Build ☑ Run
Command: $(DELPHICOMPILE) $(PROJFILE)
Parsers: ☐ Pas2JS ☑ Delphi Compiler ☐ Show all output lines

Compiler
Call on: ☐ Compile ☐ Build ☐ Run
Command: $(CompPath)

Execute after
Call on: ☑ Compile ☑ Build ☑ Run
Command:
Parsers: ☐ make ☐ FPC ☐ Pas2JS

Help | Show Options | Test | Export | Import | Cancel | OK

Figure 3: The project compiler commands page



```
Project: Executing command before: Success, Warnings: 3
Embarcadero Delphi for Linux 64 bit compiler version 35.0
Copyright (c) 1983,2022 Embarcadero Technologies, Inc.
SignalTypes.inc(22,0) Warning: Symbol '_SIGSET_NWORDS' is specific to a platform
SignalTypes.inc(139,0) Warning: Symbol 'SIGIO' is specific to a platform
hello.lpr(13,0) Warning: Variable 'b' might not have been initialized
Linker command line: Z:\opt\delphi\delphi-1101\bin\ld-linux.exe -o Z:\\home\\michael\\tmp\\hello -e _ZN5Hello14initializationEv --gc-sections
7612 lines, 0.90 seconds.
```

Figure 4: Output when compiling the project

After this, when compiling or building the project in the IDE, the 'execute before' command-line will be executed and the FPC compiler will not be called.
NOTE that you can perfectly have 2 build modes: one to compile your project with Delphi, one to compile your project with FPC.
So how to specify the delphi compile command in the 'Execute before' edit box ?
You can specify the compile command completely yourself, for example:

```
c:\delphi\bin\dcc32.exe -V c:\projects\myproject.dpr
```

Or you can use macros. The lazdelphi package defines several macros, which you can use to compose your command as you see fit:

DCC
This macro is expanded to the Delphi compiler binary path as set in the IDE options.

DCCCONFIG
This macro is expanded to the delphi compiler configuration file generated by the IDE for your project. The filename is preceded by an @sign if it is non-empty. (*the @sign is how the config file is specified on delphi compiler command-line*)

DCCARGS
This expands to the concatenation of the 'additional compiler options' specified in the global and project-specific settings.

DELPHICOMPILE
this macro is in fact equivalent to the three above macros combined:
$(DCC) $(DCCARGS) $(DCCCONFIG)
it exists for convenience.

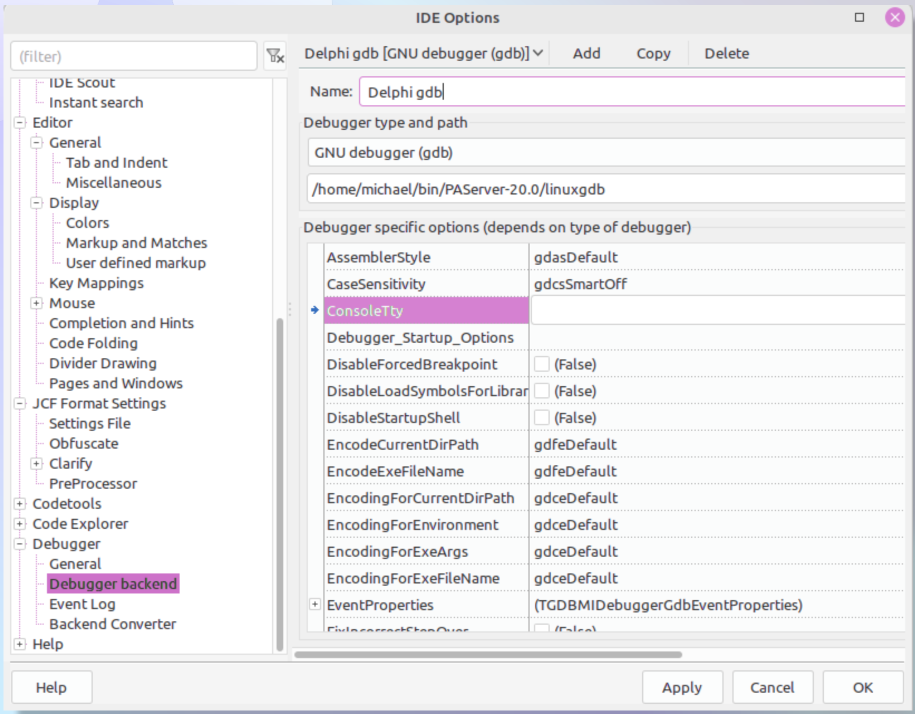
So the simplest compile command is:

```
$(DELPHICOMPILE) $(PROJFILE)
```

Exactly as shown in *figure 3 on page 4* of this article.
Once this is all done, you can compile your project with Delphi, and the result can look like what you see in *figure 4 on page 5*. Clicking on the warning will take you to the correct location in the project.

5 DEBUGGING

The lazarus IDE of course offers debugging, and the debugger has lots of advanced features
(*see the recent contributions by Martin Friebe in Blaise pascal magazine*).


IDE Options

(Filter)

- IDE Scout
- Instant search
- Editor
  - General
    - Tab and Indent
    - Miscellaneous
  - Display
    - Colors
    - Markup and Matches
    - User defined markup
  - Key Mappings
  - Mouse
  - Completion and Hints
  - Code Folding
  - Divider Drawing
  - Pages and Windows
- JCF Format Settings
  - Settings File
  - Obfuscate
  - Clarify
  - PreProcessor
- Codetools
- Code Explorer
- Debugger
  - General
  - Debugger backend
  - Event Log
  - Backend Converter
- Help

Delphi gdb [GNU debugger (gdb)] | Add | Copy | Delete

Name: Delphi gdb

Debugger type and path

GNU debugger (gdb)

/home/michael/bin/PAServer-20.0/linuxgdb

Debugger specific options (depends on type of debugger)

AssemblerStyle	gdasDefault
CaseSensitivity	gdcsSmartOff
ConsoleTty	
Debugger_Startup_Options	
DisableForcedBreakpoint	(False)
DisableLoadSymbolsForLibrar	(False)
DisableStartupShell	(False)
EncodeCurrentDirPath	gdfeDefault
EncodeExeFileName	gdfeDefault
EncodingForCurrentDirPath	gdceDefault
EncodingForEnvironment	gdceDefault
EncodingForExeArgs	gdceDefault
EncodingForExeFileName	gdceDefault
EventProperties	(TGDBMIDebuggerGdbEventProperties)

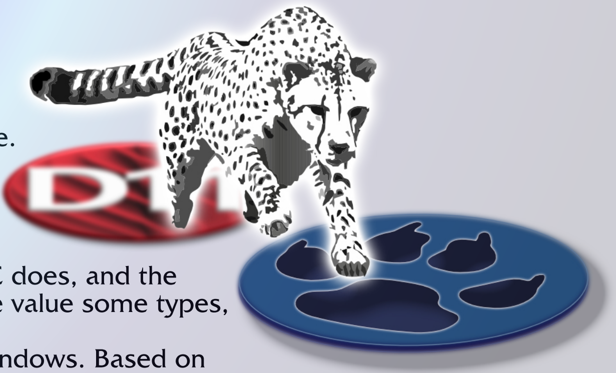
Help | Apply | Cancel | OK

Figure 5: Defining the delphi debugger

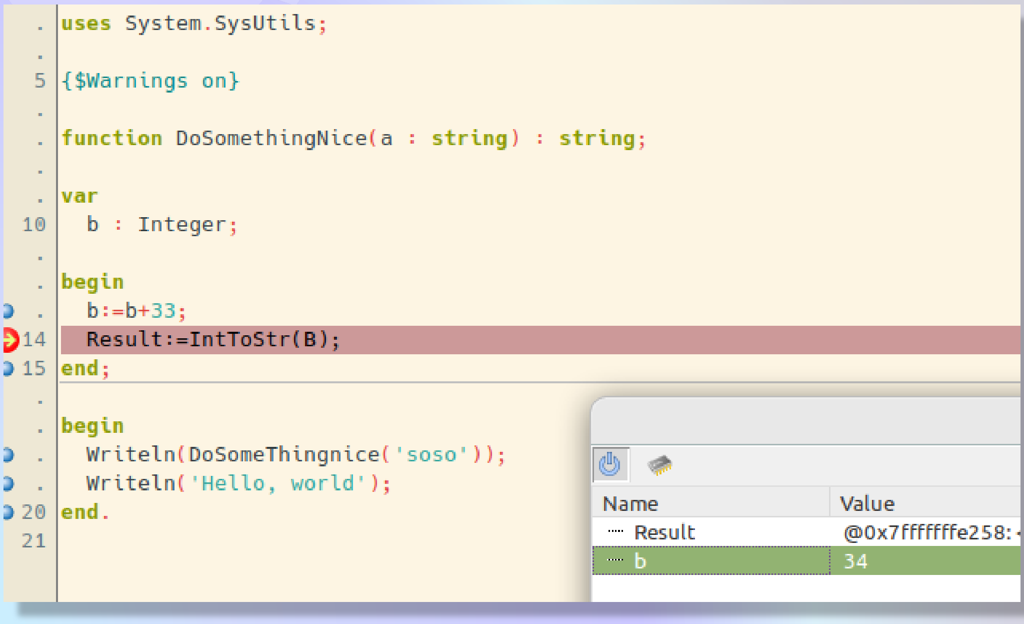
Thus the question whether the compiled executable can be debugged in the lazarus IDE is of course relevant.
The answer to this question is '**Yes, but...**':
On Linux it is definitely possible to debug the delphi-generated executable. In order to do this, GDB (*the GNU debugger*) must be used.
To get the best results, a specially modified version of GDB is needed. Delphi ships this gdb version as part of it platform assistant '**PAServer**' for Linux.
Fortunately, Lazarus can use this GDB executable. You can use set it up in the Lazarus IDE in the tools - Options dialog under '**Debugger - debug backend**'. The IDE can work with multiple debug backends, so what we must do here is define a new debugger backend. To do so, the following actions must be performed:

1. click '**Add**' at the top of the dialog.
2. Enter a new name in the '**Name**' edit.
3. For debugger type, select '**GNU debugger** *(gdb)*'.
4. Select the '**linuxgdb**' binary that is part of PAServer.

The result will look like *figure 5 on page 6* of this article.
Once this is done, you can run your application in the debugger: you can set breakpoints, you can watch variables, as shown in *figure 6 on page 7*
The debug experience is not yet flawless:
Delphi encodes certain Pascal types differently than FPC does, and the Lazarus IDE is (*not yet*) aware of this info. So to see the value some types, some typecasts may be necessary.
The author has not tested debugging on MacOS and Windows. Based on available knowledge about the tools on Windows and MacOS, it is most probable that: → see next page



- On MacOS Lazarus can be used for debugging with lldb (*the debugger used on Mac*) using a similar technique as on Linux.
- On Windows the expectation is that debugging will not be possible, since Delphi uses a proprietary format for debug info on Windows.



```
uses System.SysUtils;

{$Warnings on}

function DoSomethingNice(a : string) : string;

var
  b : Integer;

begin
  b:=b+33;
  Result:=IntToStr(B);
end;

begin
  Writeln(DoSomeThingnice('soso'));
  Writeln('Hello, world');
end.
```

Name	Value
Result	@0x7fffffffe258:
b	34

Figure 6: Debugging a Delphi-generated executable

6 CONCLUSION

The delphi tool can be used to compile your free pascal or delphi code with the Delphi compiler, right from within the Lazarus IDE.
The tool works as it is now, but some improvements are still planned: the quick fixes that exist for the FPC compiler can also be implemented for the delphi compiler. Additional ideas for improvement are of course always welcomed by the lazarus maintainers.

A word of warning:
While coding is not a problem, one should take care when editing visual forms: When you add a component (*say, a* `TEdit`) to a form, there are properties in the LCL that do not exist in Delphi.

These properties will be written to the `.dfm` file by the Lazarus IDE. When you run the program and the form is created at runtime, there will be a streaming error: the components that are actually instantiated will be the Delphi components, which may be missing some properties.

Inversely, when loading a Delphi form in the Lazarus IDE, there may be VCL properties (*or even complete components*) in the form that have no counterpart in the LCL and the Lazarus IDE will show an error, asking you what should be done with these properties.

International Pascal Conference

VNiVERSiDAD Ð SALAMANCA

From 3 to 7 July 2023, the International Pascal Congress (IPC) will take place in the city of Salamanca, Spain.
This event seeks to bring together diverse actors of the software development sector and provide various activities for attendees to update and increase their knowledge on technologies and tools related to the Pascal family languages.
The IPC will offer plenary conferences on different aspects of software development, courses to learn new technologies (neural networks, cloud computing, video games, etc.),
technological and scientific presentations, and round tables on open projects.
In addition, the Niklaus Wirth Award for the Most Valuable Contributor will be conferred. Don't miss one of the most important events ever held on Pascal in one of the most beautiful cities in Spain:
Salamanca `https://pascalcongress.com.`

International Pascal Conference

VNiVERSiDAD Đ SALAMANCA

International Pascal Conference
VNiVERSiDAD D SALAMANCA

The IPC activities will be held in the Hospedería Fonseca Auditorium, the Faculty of Sciences Auditorium and in classrooms in the Faculty of Sciences building.

International Pascal Conference
VNiVERSiDAD Ð SALAMANCA

July 3-7, 2023, Salamanca (Spain)
The International Pascal Congress (IPC) is an international forum dedicated to bringing together the different players in the software industry whose technologies are based on the Pascal family of programming languages. The IPC 2023 will be held from 3 to 7 July 2023, hosted by the University of Salamanca in Spain, and it will be a stunning and forward-looking reflection on the Pascal family of languages in the software industry. The official language of the IPC 2023 will be English.

International Pascal Conference
VNiVERSiDAD D SALAMANCA

REGISTRATION
https://www.pascalcongress.com/registration.html

Location
https://www.pascalcongress.com/location.html

Keynote speakers
https://www.pascalcongress.com/keynote_speakers.

Johannes W. Dietrich Marco Cantù Michalis Kamburelis

Primož Gabrijelčič Bruno Fierens Daniele Teti

International Pascal Conference

VNiVERSiDAD D SALAMANCA

Niklaus Wirth Award

The International Pascal Congress (IPC) considers it essential to implement measures that promote the development of Pascal and technologies based on this language within the computer science field. Among these actions is the creation of awards to stimulate and strengthen the Pascal community. The IPC will grant one award in honour of Niklaus Wirth, a pioneer in the creation of programming languages and the designer of Pascal. The Niklaus Wirth Award for the Most Valuable Contributor to Pascal is intended to recognise individuals who have made decisive contributions to improving, developing, and strengthening the community of those who use Pascal programming languages in any variant to develop software.

NWA for the Most Valuable Contributor
This award is intended to recognise individuals who have made a significant contribution to improving, developing and strengthening the community of software developers using Pascal in any of its variants for software creation.

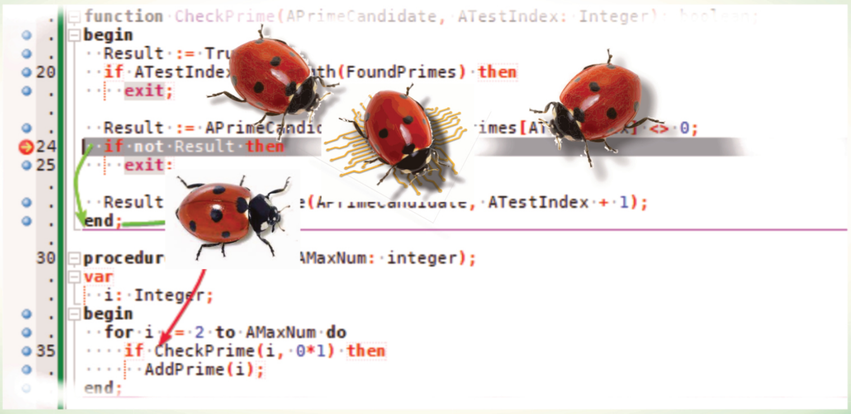
If the candidate wins, he/she agrees to give a plenary lecture in person after receiving the prize on a topic of his/her choice related to Pascal programming languages or technologies associated with the Pascal family.

The award will be presented by the Deputy Rector for Research and Transfer of the University of Salamanca.

Nomination period: MAY/06/2023 – JUNE/05/2023
Award announcement: May/04/2023
Ceremony: JULY/03/2023

BLAISE PASCAL MAGAZINE 110 / 111

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



```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := Tru
  if ATestInde           th(FoundPrimes) then
    exit;

  Result := APrimeCandic            imes[AT         ] <> 0;
  if not Result then
    exit;

  Result                 e(APrimeCandidate, ATestIndex + 1);
end;

procedur                AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0*1) then
      AddPrime(i);
end;
```

AI-enabled brain scanner reads thoughts

Image Classifier with loading and testing a pre-trained model

Delphi Community version for Delphi 11

Jim McKeeth leaving Embarcadero/Delphi

The Number guessing project

BLAISE PASCAL MAGAZINE LIBRARY By internet and on USB Stick
The Library kit for BPM has been extended with new features: Search over ALL 111 issues and per issue.

Lazarus compiling Delphi code

Lazarus for Visual Studio

Debugging with the new debugger in Lazarus - lessons part 2

FastReport for Lazarus on LINUX
in a Trial and as Professional version

SUBSCRIPTION FOR 2 YEAR
BLAISE PASCAL MAGAZINE
ONLY €120 ex Vat

BLAISE PASCAL MAGAZINE LIBRARY

BY INTERNET AND ON USB STICK Written by Michael van Canneyt

Starter Expert

ABSTRACT

Blaise Pascal Magazine offers subscribers a library : a collection of all issues available till now. In this article we show how the PDF indexer application presented in the previous articles about indexing PDF files will be used to rewrite and enhance the **Blaise Pascal Magazine library**.

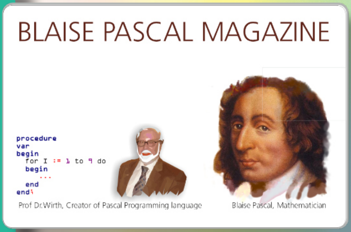
❶ INTRODUCTION

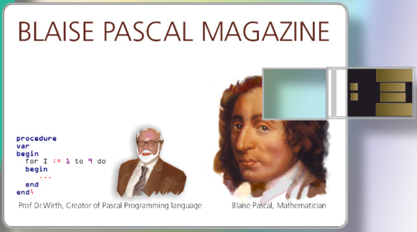
In several earlier contributions, we showed how to show a PDF file in an offline PAS2JS application, and how to index PDF files and use that index to search and download PDF files. In this article, we'll show how to combine all these techniques to rewrite the Blaise Pascal Magazine library as a Pas2js application that works both offline and online. The new edition of the Blaise Pascal Magazine library will need to have the following features:

- It must work as a **local application**, distributed on a **USB** stick.
- It must work as a web application, deployed on the Blaise Pascal Magazine **website**.
- When working locally, issues must be loaded from the local disk: usually a **USB stick**.
- The issues must be search-able.
- When working locally, and no internet connection is available, then a (*limited*) search must be performed locally in the list of articles. if an internet connection is available, the application must be able to search globally and download a PDF if needed.

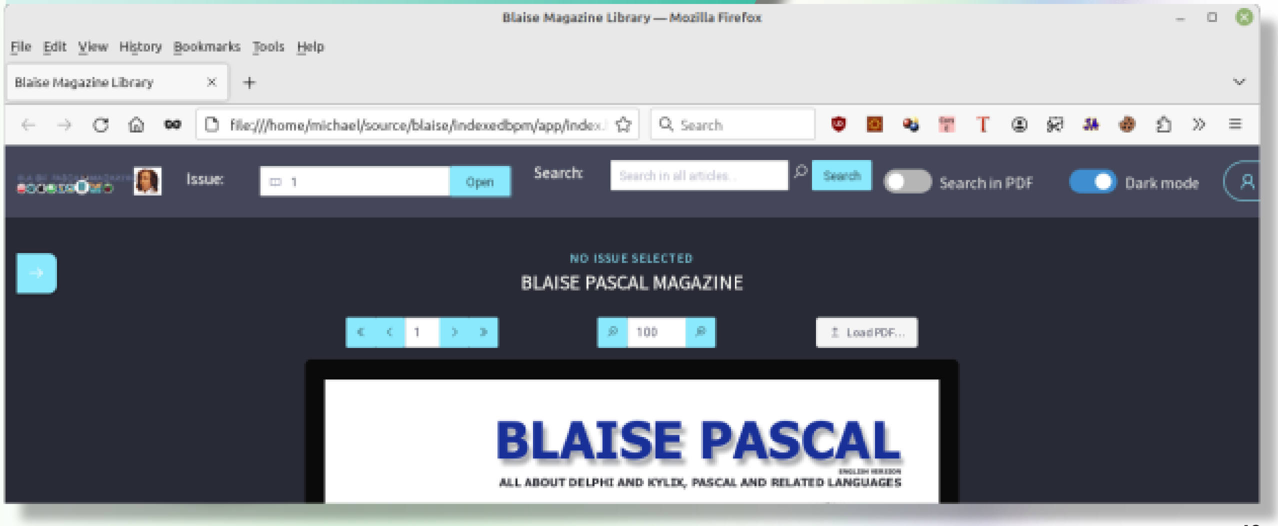
- **PDF Downloads** are limited to the downloads purchased by the magazine subscriber.
- the issue will be displayed.

All of the techniques needed to satisfy these requirements have been presented in previous articles.
In this article we bring everything together:
this will require some refactoring of the code presented in the previous articles. Since we'll be needing a server part and a client part, we'll start by discussing the code changes needed on the server.


BLAISE PASCAL MAGAZINE
procedure
var
begin
for I := 1 to 4 do
begin
...
end;
end;
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician


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

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands
Prof Dr.Wirth, Creator of Pascal Programming language
editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu


Blaise Magazine Library — Mozilla Firefox
File Edit View History Bookmarks Tools Help
Blaise Magazine Library
file:///home/michael/source/blaise/indexedbpm/app/index.
Search
Issue: 1 Open
Search: Search in all articles.. Search
Search in PDF
Dark mode
NO ISSUE SELECTED
BLAISE PASCAL MAGAZINE
1
100
Load PDF...
BLAISE PASCAL
ALL ABOUT DELPHI AND KYLIX, PASCAL AND RELATED LANGUAGES

❷ ADDING SECURITY TO THE SERVER

In previous articles about indexing PDF files, we implemented a simple mechanism to download issues from the server in addition to the search mechanism and word list mechanism.
Any issues could be downloaded from the server.
This needs to be modified so the user can only download what he has subscribed for.
This means we need to add a login mechanism:
the server needs to know who is attempting to download an issue.
We'll also need a mechanism to determine what issues a user is allowed to download. In order to implement this, we need to extend the database with this information.
First, we'll need a list of users: a table with at least a username and a password. The SQL to create such a table (*let's name it users*)can for example look like this in **Postgres**:

```
create   sequence seq_users;

  create   table users   (
  u_id  bigint not  null  default nextval('seq_users'),
  u_firstname varchar(50) NOT  NULL,
  u_lastname  varchar(50) NOT  NULL,
  u_login varchar(127) not  null,
  u_password varchar(127) not  null,
  constraint pk_users  primary  key (u_id)
);

create   unique  index  udx_users  on users  (u_login);
```

The fieldnames speak for themselves, and the index is there to make sure every login is unique.
Similarly, we'll need a table with all available issues. We could take the articles table presented in the previous articles, but every issue occurs more than once in that table, so it is difficult to create a foreign key on this table for referential integrity.
Instead, we create a new table, aptly named issues:

```
create   sequence seq_issues;

create table issues (
  i_id bigint not  null  default nextval('seq_issues'),
  i_issue varchar(10) NOT  NULL,
  i_filename varchar(127) NOT  NULL,
  constraint pk_issues  primary  key (i_id)
);
```

The `I_issue` field is not a number to accommodate for double issues, so we can support a notation as `81 82` for the combined issues `81` and `82`. To know which issues a given user can access, we need a third table (*named userissues*):

```
create sequence seq_userissues;
create table  userissues (
  ui_id bigint not  null  default nextval('seq_userissues'),
  ui_user_fk bigint  NOT  NULL,
 ui_issue_fk bigint  NOT  NULL,
 constraint pk_userissues primary key (ui_id)
);
```

For each user and each issue that user can access, a record is inserted in this table, with a link to the users table and issues table.
For referential integrity, we enforce a foreign key to these tables;

```
alter table  userissues add constraint fk_userissues_users
  foreign key (ui_user_fk)  references users(u_id) on delete  cascade;
alter table  userissues add constraint fk_userissues_issues
  foreign key (ui_issue_fk)  references issues(i_id) on delete  cascade;
```

Additionally, we make sure there is only one record for each user - issue combination.

```
create   unique  index  udx_userissue   on userissues (ui_user_fk,ui_issue_fk);
```

Armed with these tables, we can now implement some security mechanisms. We will not describe how data gets into these tables: we'll assume the tables have been filled through some external mechanism, through some link with the subscription system.
The first thing is to implement a kind of login routine:
we implement a simple HTTP endpoint which receives a JSON with a username and password (*this is not a JSON-RPC mechanism*):

```
 {
   "username":  "michael",
   "password": "verysecret"

}
```

The server will respond with a token (a simple GUID) :

```
{
   "token" : "{F5A07A5B-5184-4256-8EE6-20E2DE987AF5}",
   "expires" : "2023-06-03T17:12:09.489Z"
}
```

This token can be passed to the server when a PDF is downloaded: the server will then verify this token and allow the download if it is valid.
The tokens are also stored in the database, in a table called tokens:

```
CREATE TABLE tokens
(
  tk_id bigint NOT  NULL  DEFAULT  nextval('seqTokens'::regclass),
  tk_token character varying(38)  NOT NULL
    DEFAULT (upper((('{'::text || uuid_generate_v4() ||
       '}'::text)))::character varying(38)
  tk_user_fk bigint  NOT NULL,
  tk_expires_on timestamp  without time  zone NOT NULL
    DEFAULT  (now()  + '00:30:00'::interval),
  CONSTRAINT  pktokens  PRIMARY  KEY  (tk_id)
);

create   index  idx_token_expires on tokens(tk_expires_on);
create   unique  index  udx_tokens  on tokens(tk_token);
```

The `tk_expires_on` is by default filled with a timestamp 30 minutes from the time of insert of the record. In effect, the token will be valid for 30 minutes.
The `uuid_generate_v4` function is part of a Postgres extension and generates a GUID, which will serve as our unique session token.
The extension needs to be activated with the following SQL statement:

```
CREATE  EXTENSION  IF  NOT EXISTS   "uuid-ossp";
```

To implement this scheme, we create a new class `TSessionManager`:

```
TSessionManager = Class(TComponent)
Public
  constructor Create(aOwner : TComponent); override;
  destructor destroy; override;
  // Public API
  Procedure  GetToken(aRequest : TRequest;  aResponse : TResponse);
  Function CheckToken(const aToken : string) : int64;
  function CheckFileAllowed(aUserID : int64; const  aFileName : string) : Boolean;
  Property DB  : TSQLConnection Read FDB  Write  FDB;
end;
```

The GetToken call is registered as the handler for the **`/token`** login during application startup:

```
aSession:=TSessionmanager.Create(Nil);
aSession.DB:=aSearch.Connection;
HTTPRouter.RegisterRoute('/token',@aSession.GetToken,False);
```

The `GetToken` call is registered as the handler for the /token login during application startup:

```
aSession:=TSessionmanager.Create(Nil); aSession.DB:=aSearch.Connection;
HTTPRouter.RegisterRoute('/token',@aSession.GetToken,False);
```

The DB property is the SQL connection used by the search class:
it was set up in code of the previous articles and is simply reused here.

The `GetToken` call is pretty straightforward.
It starts with checking for a `CORS` request, in a similar manner to the way it was done in the PDF file download in the previous article.
This is needed, because when the client application is started from a local disk, the origin differs from the server.

When the `CORS` request indicates that all is well, the code starts by decoding the request payload as a JSON structure.

If something goes wrong then the `ReportInvalidParam` routine is used to report a `HTTP  400` return code. If the JSON is decoded correctly, the username and password are extracted.
Again a check is done to see if values have been passed for both username and password.
If not a `HTTP 400 error` is again reports:

```
procedure TSessionManager.GetToken(aRequest: TRequest; aResponse: TResponse);
Var Req, Resp : TJSONData;
     Obj : TJSONObject absolute Req;
     token,UserName, UserPW : String;
     aExpiresOn : TDatetime;
begin
   if FCors.HandleRequest(aRequest,aResponse,[hcDetect,hcSend]) then exit;
   Req:=Nil;
   Resp:=Nil;
     try Req:=GetJSON(aRequest.Content);
       if not (Req is TJSONObject) then
         ReportInvalidParam(aResponse)
       else
       begin
         userName:=Obj.Get('username','');
         userPW:=Obj.Get('password','');
         if (UserName='')  or (userPW='') then
           ReportInvalidParam(aResponse)
         else
           begin token:=ValidateUser(UserName,UserPW,aExpiresOn);
             if Token='' then
             ReportForbidden(aResponse)
             else
               begin
                 Resp:=TJSONObject.Create([
                         'token',token,
                         'expires',DateToISO8601(aExpiresOn)
                 ]);
                 SendJSON(aResponse,200,'OK',Resp);
               end;
           end;
         end;
     except
       on E : Exception do
         ReportException(aResponse,E);
     end;
     Resp.Free;
     Req.Free;
end;
```

The username and password are validated using the `ValidateUser` call: it will return a token if the **username/password** pair was valid.
If the token is empty, it means the combination was not valid and an error is reported. Lastly, if we received a token and expiration date, we send both back to the client in a JSON structure using `SendJSON`:

```
procedure TSessionManager.SendJSON(aResponse : TResponse;aCode : Integer; aText  : String; aJS

begin
   if aResponse.ContentSent then
      exit;
   aResponse.Code:=aCode;
   aResponse.CodeText:=aText;
   aResponse.ContentType:='application/json';
   aResponse.Content:=aJSON.FormatJSON();
   aResponse.SendContent;
end;
```

The same `SendJSON` method is used in e.g. the `ReportInvalidParam` method:

```
procedure TSessionManager.ReportInvalidparam(aResponse: TResponse); Var J : TJSONObject;
begin
   J:=TJSONObject.Create(['message','need username/password']);
   try
      SendJSON(aResponse,400,'INVALID PARAM',J);
   finally
      J.Free;
   end;
end;
```

The `ValidateUser` method is again very straightforward: the only noteworthy thing is that the password is stored encrypted in the database. We use the native **Postgres** database cryptographic mechanisms for this: the crypt function encrypts a value using a salt, and the same function can be used to insert the data.
The crypto functionality must be enabled with the following SQL statement:

```
CREATE EXTENSION pgcrypto;
```

Using this function, the SQL statement to verify a user password is as follows:

```
SELECT
  U_password=crypt(:password,U_password) as PasswordOK, *
from
  Users where
    (U_login=:login);
```

If the user login is not found, the query will return no records.
If the user is found, then there will be a single record (*because the login is unique*), and the `PasswordOK` field will be `True` if the password passed in parameter:password matches the one stored in the database, and it will be `False` if not.
Using this SQL statement we can easily create the `ValidateUser` call.
It starts out by creating a database transaction: every operation is done in its own transaction.
After the transaction is created, it is used to create a `TSQLQuery` dataset and run the SQL statement (*The* `CreateTransaction` *and* `CreateQuery`) functions are trivial and will not be presented here:

```
function TSessionManager.ValidateUser(const aUser, aPassword: String;
out aExpires: TDateTime): String;
Const
  SQLSelectUser =
  'SELECT U_password=crypt(:password,U_password) as PasswordOK,*' +
  'from Users '+
  ' where (U_login=:login)';

Var
  Tr  : TSQLTransaction;
  Qry : TSQLQuery; Res,OK : Boolean;
  aID : Int64;

begin
  OK:=False;
  Result:=''; qry:=Nil;
  Tr:=CreateTransaction;
  try Qry:=CreateQuery(SQLSelectUser,['LOGIN',aUser,'PASSWORD',aPassword],Tr);
    Qry.Open;
    // If we have a user and the password matches
    Res:=(Not Qry.IsEmpty) and (Qry.FieldByName('PasswordOK').AsBoolean);
    aID:=Qry.FieldByName('u_id').AsLargeInt;
    if Res then
      // We get a token
      Result:=CreateToken(aID,aExpires,Qry.SQLTransaction);
    Tr.Commit;
    OK:=True;
  finally
    if not OK then
      Tr.Rollback;
      ReleaseQuery(Qry,Tr);
  end;
end;
```

If the user is verified, then the user ID and transaction are passed to `CreateToken`, which will create a new token.
NOTE that the token is created in the same transaction:
The `CreateToken` function is again quite simple, it makes use of the fact that the 'default' values in the table column definitions create usable values, and simply returns the values created by **Postgres**.

```
function TSessionManager.CreateToken(aUser: Int64; out Expires: TDateTime;
    aTransaction : TSQLTransaction): String;

Const
SQLInsert  =
  '  insert into tokens  (tk_user_fk) values  (:USER)' +
  '  returning tk_token, tk_expires_on;';

Var
  Qry : TSQLQuery; OK  : Boolean;

begin OK:=False;
  Qry:=CreateQuery(SQLInsert,['USER',aUser],aTransaction);
  try
    Qry.Open;
    if Qry.IsEmpty  then
      DatabaseError(SErrFailedToCreateToken, self);
      Result:=Qry.FieldByName('tk_token').AsString;
      Expires:=Qry.FieldByName('tk_expires_on').AsDateTime; OK:=True;
  finally
    if Not OK  then
      Qry.SQLTransaction.RollBack; ReleaseQuery(Qry);
  end;
end
```

With these routines we have created a HTTP endpoint that can be used in the application to ask for a token.

❸ SECURING THE DOWNLOAD

When the user wants to downloads a PDF file, the token must be supplied so the server can verify who is making the download, and whether the user is allowed to download the requested PDF file. The token can be specified in one of 2 ways:

- As a URL query parameter, called 'token':

```
http://localhost:3010/pdf/BlaisePascalMagazine_61_UK.
pdf?token=%7BF5A07A5B-5184-4256-8EE
```

- As a HTTP header, called 'X-Access-Token':

```
X-Access-Token:   {F5A07A5B-5184-4256-8EE6-20E2DE987AF5}
```

This means we must adopt the download module so it first checks the token, and then checks if the user that owns the token can download the file. The change is trivial:

```
procedure TCorsFileModule.HandleRequest(ARequest: TRequest;
  AResponse: TResponse);
begin
  Cors.Enabled:=true;
  if Cors.HandleRequest(aRequest,aResponse) then exit;
  if not CheckToken(aRequest,aResponse) then exit;
  inherited HandleRequest(ARequest, AResponse);
end;
```

The CheckToken function does the actual work. It uses the CheckToken function from the TSessionManager class to verify the token. If the token is OK, the user ID is returned, if the token is not OK, -1 is returned. The returned user ID is then used to check if the user is allowed to download the requested PDF (the GetRequestFileName function is a method of the file download datamodule that comes with FPC):

```
function TCorsFileModule.CheckToken(ARequest: TRequest; AResponse: TResponse): Boolean;
var
   aToken,aFileName : String;
   aID : int64;
begin
   // Check URL parameter and HTTP header for token.
   aToken:=aRequest.QueryFields.Values['token'];
   if aToken='' then
      aToken:=aRequest.CustomHeaders.Values['x-access-token']; Result:=(aToken<>'');
   if Result then
     begin
     // Check token in database
      aID:=aSession.CheckToken(aToken); Result:=aID<>-1;
     if Result then
       begin
         // Get requested filename.
         aFileName:=ExtractFileName(GetRequestFileName(aRequest));
         // Check if user is allowed to download this file.
         Result:=aSession.CheckFileAllowed(aID,aFileName)
       end;
     end;
   if not Result then
     begin
       aResponse.Code:=403;
       aResponse.CodeText:='FORBIDDEN';
       aResponse.SendContent;
     end;
end;
```

Note that if the token is invalid, or the user is not allowed to download the PDF, a 403 FORBIDDEN HTTP return code is sent to the browser. The CheckToken function of the session manager does the actual check of the token. It is again quite simple:

```
function TSessionManager.CheckToken(const aToken: string): int64;
const
   SQLSelect = 'select tk_user_fk,tk_expires_on from tokens where (tk_token=:token)';
var
   Tr : TSQLTransaction;
   Qry : TSQLQuery;
   OK : Boolean;
begin
   OK:=False;
   Qry:=nil; Result:=-1;
   TR:=CreateTransaction;

   try Qry:=CreateQuery(SQLSelect,['token',aToken],Tr); Qry.Open;
     if Not Qry.IsEmpty and (Qry.FieldByName('tk_expires_on').asDateTime>Now) then
       Result:=Qry.FieldByName('tk_user_fk').asLargeInt;
     if Result<>-1 then
       UpdateToken(aToken,Tr);
   finally
     if not OK then TR.Rollback; ReleaseQuery(Qry,Tr);
   end;
end;
```

If the token has not expired, it is extended with 30 minutes using the `UpdateToken`:

```
function TSessionManager.UpdateToken(const aToken: String;
    aTrans: TSQLTransaction): TDateTime;

Const
   SQLUpdate  =
     'update tokens  set  ' +
     ' tk_expires_on = clock_timestamp() + interval ''30 minutes'' '+
     ' where (tk_token=:TOKEN)  returning tk_vervalt_op;';

Var
   Qry : TSQLQuery;

begin
   Qry:=CreateQuery(SQLUpdate,['TOKEN',aToken],aTrans);
   try
     Qry.Open;
     if not  Qry.IsEmpty  then
       Result:=Qry.Fields[0].AsDateTime
     else Result:=0;
   finally
     ReleaseQuery(Qry);
   end;
end;
```

This is to avoid that the user needs to login every 30 minutes, but after more than 30 minutes of inactivity, the token does expire.
The above mechanism is a simple one, in practice, more advanced strategies can be used.
Lastly, the `CheckFileAllowed` call is used to check whether the user is entitled to download the requested PDF file.
This is done using the userissues table:
if a record is present for the requested issue and user, the user is allowed to download the pdf. The check is then very simple:

```
function TSessionManager.CheckFileAllowed(aUserID: int64;
const  aFileName:  string): Boolean;

Const
    SQLSelect =
      'select '+ '     ui_id '+ 'from '+ '    userissues '+
     ' inner join issues  on (i_id=ui_issue_fk) '+ 'where  '+
     ' (i_filename=:filename) '+'and (ui_user_fk=:uid)';

var
   Tr  : TSQLTransaction;
   Qry : TSQLQuery;

begin
   Tr:=CreateTransaction;
   try
     Qry:=CreateQuery(SQLSelect,[
          'uid',aUserID,
          'filename',lowercase(aFileName)],Tr);
     Qry.Open;
     Result:=Not Qry.IsEmpty;
   finally
     ReleaseQuery(Qry,Tr);
   end;
end;
```

❹ ALLOWING TO DOWNLOAD AN ISSUE BY NUMBER

One of the requirements was that the user can download and view an issue by entering the number of the issue. To map this to a PDF filename, a routine is needed that checks the issues table and returns the corresponding filename. To implement this, when the application needs to show an issue, we'll let it download a PDF with a special URL:

```
http://localhost:3010/issue/45
```

In the above URL, the number 45 must be replaced with the actual issue. To code this on the server, we must implement a handler for the above URL. We register it with the HTTP router as follows:

```
HTTPRouter.RegisterRoute('/issue/:Issue',@IssueToPDF);
```

The IssueToPDF is a simple routine. The search mechanism has the list of articles in an issue in an array in memory. This list can be accessed to retrieve the filename of the issue.
When a filename is found, instead of sending the file, a redirect response is sent to the browser with the 'normal' pdf download location: The redirect response means a 307 HTTP return code is sent, and the location of the PDF is returned in the Location HTTP header.
The server response will be something like this:

```
HTTP/1.1307 Temporary Redirect
Location:  /pdf/BlaisePascalMagazine_45_46_UK.pdf
```

Upon receiving the 307 return code, the browser will immediately do a second request to the new location. To the user, this is transparent.
To code this is quite simple:

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

var Cors : TCORSSupport;
    PDF  : String;

begin
   Cors:=TCORSSupport.Create;
   try
     Cors.Enabled:=true;
     if Cors.HandleRequest(aRequest,aResponse) then exit;
   finally
     Cors.Free;
   end;
   PDF:=aSearch.IssueToPDF(aRequest.RouteParams['Issue']);
   if (PDF<>'') then
     aResponse.SendRedirect('/pdf/'+PDF)
   else
     begin aResponse.Code:=404;
       aResponse.CodeText:='Not Found';
     end;
   aResponse.SendContent;
end;
```

With this, we have completed the extension of the server. We can now turn to the changes in the client (PAS2JS) application.

server server

❺ SEARCHING IN THE CLIENT - second part

In the previous articles where searching through a PDF in a browser application was handled, 3 mechanisms have been treated, where a search was performed in 3 different locations:

❶ In the displayed PDF, using the mechanisms provided by the `pdf.js` package in the browser.
❷ In a list of articles, using an in-memory copy of the list of articles.
❸ In a database built with a PDF indexer.

If our application should be able to work online and offline, we must consider if each of the mechanisms is usable:

Searching in the displayed PDF is of course always possible.
When offline, the search in the database is unavailable, and the best we can do is replace it transparently by a search in the list of articles.

In order to do so, we need to adapt the search mechanism:
in our last iteration of the PDF application, the search algorithms (*PDF and database*) were handled by the *TPDFSearchControl*.
We now need to add the search in the list of articles
(*as demonstrated in the first article about showing a PDF*)
to this class.

To keep the code simple, we'll split out the search mechanisms in separate classes.
The `TPDFSearchControl` will then, depending on the user setting and the on- line/offline status of the browser, select a search mechanism.

The 3 search classes are responsible for searching and displaying the results below a given HTML tag.
When the user selects a result, a special event is triggered with enough information to show the selected result.

The `TPDFSearchControl` will then do what is necessary to display the PDF and jump to the page containing the result.
The search class is also responsible for returning a list of words for auto-completion in the search box:
the mechanism that was built to search the indexed database has an implementation.

For the list of articles, a list of words can be built on the fly, and a list of words in the current PDF can also be constructed.

Since the three mechanisms need to perform the same functions, we define the following interface to encapsulate the requirements:

```
TPageInfo  = record
  Issue, Title,  FileName : String; Page: Integer;
  useIssue  : Boolean;
end;

TShowPDFPageEvent = procedure(aPage  : TPageInfo)  of  object;
TWordListCallBack = reference to  procedure(List : TStrings);

{ ISearchEngine  }

ISearchEngine  = Interface
  // Property getters  &  setters
  function GetOnShowResultPanel: TNotifyEvent;
  procedure SetOnShowResultPanel(AValue: TNotifyEvent);
  procedure  SetResultsElement(aValue : TJSHTMLElement);
  Function GetResultsElement  : TJSHTMLElement;
  procedure  SetShowPageEvent(aValue : TShowPDFPageEvent);
  function GetShowPageEvent  : TShowPDFPageEvent;

  // Actual   interface
  procedure  Search(const aTerm : string; const  aIssue  : String);
  procedure  GetWordList(aTerm  : string; aOnResults : TWordListCallBack);

  // Easy access using  properties
  property ResultsElement  : TJSHTMLElement
    Read GetResultsElement  Write  SetResultsElement;
  Property ShowPDFPageEvent  : TShowPDFPageEvent
    Read GetShowPageEvent  Write  SetShowPageEvent
  Property OnShowResultPanel : TNotifyEvent
    Read GetOnShowResultPanel Write  SetOnShowResultP
end;
```

The `Search` call will display the list of found occurrences of the search term below `ResultsElement`. The `OnShowResultPanel` event can be used to notify the caller that there were results, and that the result element needs to be shown (*the result panel is by default closed, it needs to be opened when results are available*).

When the user clicks a result, the `ShowPDFPageEvent` event is triggered with a `TPageInfo` record: this record contains enough information to download a PDF if needed, and jump to the correct page.
The `GetWordList` method is also clear: it needs to show a list of words. When a word list is available the `aOnResults` callback is called, passing it the list of words.

A callback must be used, since the search can be asynchronous: consulting the database on the server is an asynchonous call.

In the previous iteration of the PDF viewing and indexing application, the `TPDFSearchControl` contained 2 search mechanisms.
We'll factor these out into their own classes, so we'll have 4 classes that work together to implement the search functionality.
The first 3 classes are just a refactoring of the existing classes

TPDFSearchControl
This will just handle the search mechanism's UI:
it man- aged the edit and search buttons, shows the word list for completion and shows or hides the results panel. The actual search is handled by the other 3 components. When a PDF must be shown, an event handler is called.

TServerSearch
Implements the above interface using the server search mechanism discussed in the previous article.

TPDFSearch
Implements the above interface for searching in the displayed PDF. It uses the PDF search mechanism discussed in the first article in this series.

TArticleSearcher
Implements the above interface using a search mechanism in a list of articles which is included in the application when it is loaded from disk.

When the TPDFSearchControl class is created, it creates instances of the 3 search mechanisms:

```
constructor TPDFSearchControl.Create(aOwner:  TComponent);
begin
  Inherited;
  FLocalsearch:=TArticleSearcher.Create(Self);
  FServerSearch:=TServerSearch.Create(Self);
  FSearch:=TPDFSearch.Create(Self);
end;
```

And it initializes them in its BindElements method:

```
procedure  TPDFSearchControl.BindElements;

begin
   // ... other   code...
   PrepareEngines(True);
end;

procedure  TPDFSearchControl.PrepareEngines(Full : boolean);
begin
  PrepareEngine(FLocalsearch as ISearchEngine,Full);
  PrepareEngine(FServerSearch  as ISearchEngine,Full);
  PrepareEngine(FSearch  as ISearchEngine,Full);
end;

procedure  TPDFSearchControl.PrepareEngine(aEngine : ISearchEngine;
             Full  : Boolean);
begin
  aEngine.ShowPDFPageEvent:=FOnShowPDFPage;
  if Full  then
     begin
       aEngine.OnShowResultPanel:=@HandleShowResultPanel;
       aEngine.ResultsElement:=pnlResults;
     end;
end;
```

The PrepareEngine is called for all 3 search engines: it will initialize the relevant properties so the classes can do their work.
The FOnShowPDFPage is an event handler that is set by the main application class: the main application class is responsible for loading a PDF file.
The 'click' event handler of the search button in TPDFSearchControl now becomes quite simple:

```
procedure  TPDFSearchControl.onSearch(aEvent: TJSEvent);
var
  aterm  : string;
begin
  aterm:=SearchTerm;
  if Length(aTerm)<=1 then exit;
  CurrentSearchEngine.Search(aTerm,FIssueFilter);
end;
```

The CurrentSearchEngine property returns an ISearchEngine interface. The getter of this property decides which search engine to return based on the PDFSearch property (*basically the value of the* '**Search PDF**' *checkbox*) and a property OffLine:

```
function TPDFSearchControl.GetSearchEngine: ISearchEngine;
begin
  if PDFSearch  then
    Result:=FSearch  as ISearchEngine
  else
    if OffLine  then
      Result:=FLocalsearch as ISearchEngine
    else
      Result:=FServerSearch as ISearchEngine;
end
```

The OffLine property is determined by the online or offline status of the navigator. It is determined during startup of the application, and is maintained during the lifetime of the application. We'll show how to do this later on.
The application has a feature where the edit box shows a list of words for completion, based on the returns from the server. This mechanism needs to be reworked so the list of words is fetched from the current search engine. This mechanism was implemented in a timer event, which now becomes quite short:

```
function TPDFSearchControl.DoCompleteWord(Event: TEventListenerEvent): boolean;

  procedure  DoServerSearchWord;
  begin
    if Length(edtSearch.Value)>1 then
      CurrentSearchEngine.GetWordList(edtSearch.Value,@DoShowWordList);
     end;

begin
  Result:=False;
  if FSearchTimerID<>0 then
     window.clearTimeout(FSearchTimerID);
  FSearchTimerID:=window.SetTimeout(@DoServerSearchWord,200);
end;
```

The GetWordList will call DoShowWordList as soon as the list of words has been retrieved. The DoShowWordList routine which will actually show the list of words now simply needs to iterate over all words in the list:

```
procedure  TPDFSearchControl.DoShowWordList(List : TStrings);
Var
S  : String;
P  : TJSHTMLELement;
A  : TJSHTMLAnchorElement;

begin mnuAutoComplete.style.setProperty('display','none');
  mnuAutoComplete.InnerHTML:='<div  class="dropdown-content"></div>';
  P:=TJSHTMLELement(mnuAutoComplete.firstElementChild);
  For  S  in List do
    begin
      a:=TJSHTMLAnchorElement(Document.createElement('a'));
      a.href:='#';
      a.classlist.Add('dropdown-item');
      a.innerText:=s; a.dataset['value']:=s;
      a.addEventListener('click',@DoWordSelected); P.appendChild(a);
    end;
  mnuAutoComplete.style.setProperty('display','block');
end;
```

The implementation of the algorithm to retrieve a word list was moved to the TServerSearch class. It remains almost the same as it was, with exception that it fills a TStringlist.
The ISearchEngine interface contains some boilerplate code to define 3 properties (*they must be defined through* getters *and* setters).
In order to reduce the code needed in the 3 search classes to implement this interface, we'll descend all 3 classes from a common ancestor: TSearchBase. Here is the definition:

```
TSearchBase = class(TComponent)
Private
  FShowPDFPageEvent  : TShowPDFPageEvent;
  FOnShowResultPanel : TNotifyEvent;
  FResultsElement: TJSHTMLElement;
Protected
  // Property getters  &  setters
  Function GetOnShowResultPanel: TNotifyEvent;
  Procedure SetOnShowResultPanel(AValue: TNotifyEvent);
  Procedure  SetResultsElement(aValue : TJSHTMLElement);
  Function GetResultsElement  : TJSHTMLElement;
  Procedure  SetShowPageEvent(aValue : TShowPDFPageEvent);
  Function GetShowPageEvent  : TShowPDFPageEvent;
  // Easy access for  descendents
  procedure  ShowPDFPage(aInfo : TPageInfo);
  // Show  results  panel.
  procedure  ShowResultsPanel;
  // Clear  results  panel.
   procedure  ClearResultPanel;
  // Append a HTML  node to  the  results  panel.
  procedure  AppendToResults(aElement : TJSHTMLElement);
Public
  // Easy access using  properties
  Property ResultsElement  : TJSHTMLElement  Read GetResultsElement
                                            Write SetResultsElement;
  Property ShowPDFPageEvent  : TShowPDFPageEvent  Read GetShowPageEvent
                                                 Write SetShowPageEvent
  Property OnShowResultPanel : TNotifyEvent Read  GetOnShowResultPanel
                                            Write   SetOnShowResultP
end;
```

The various Get* and Set* methods do nothing but setting and setting the values of the private fields for the properties. The easy access methods are there to call the event handlers, if they have been set. This avoids that descendents must all implement a check. The `TServerSearch` class is reworked as a descendent of this class, but contains no new code compared to the previous iteration of our application, so we won't repeat it here. The same is true for the `TPDFSearch` class: nothing changes for this class, except that the signature of the method changes somewhat.

6 WORKING OFFLINE

When working offline, we cannot contact the server and perform a search on a database. We also cannot distribute and access the database from within the browser.
What we can do is offer limited search: the list of articles and issues is distributed with the offline version of the application. Basically, we create a javascript file in containing a 'database' of articles. The database is then simply a Javascript array of records, which looks like this (*formatting added for display purposes*):

```
var  BPMArticles  = [
{
"i" : "1", "p"  : 6,
"a"  : "Representing graphics for  math functions",
"u"  : "Peter Bijlsma",
"c" : ""
},
{
"i" : "1", "p"  : 8,
"a"  : "Client Dataset  Toolkit", "u"  : "Detlef  Overbeek",
"c" : ""
},
// ....
]
```

This javascript file is included in the html page using a script element:

```
<script src="js/articles.js"></script>
```

Accessing this array from a Pascal program is easy: One record in this array can be declared in Pascal as an external class:

```
Type
  TArticle = Class external name 'Object' (TJSObject)
  Issue  : String; external name 'i';
  Page : Integer; external name 'p';
  Title : String; external name 'a';
  Author  : String; external name 'u';
  Code : string;  external name 'c';
end;
  TArticleArray = Array of TArticle;
```

Note the use of 'external name' to map the human-readable fields (Issue, Page etc.) to the actual member names used in Javascript. The array itself is then defined as follows:

```
var
    BPMArticles : TArticleArray; external name 'BPMArticles';
```

As you can see, the variable is declared external: this means it is actually defined outside the pascal code.
Armed with this, we can now set about creating a class that implements a local search mechanism and a mechanism to get a word list: the `TArticleSearcher` class in the `articlesearch` unit.
This class is defined as a descendent of `TSearchBase`, and has 2 main methods. The first is to get a list of words:

```
procedure TArticleSearcher.GetWordList(aTerm: string; aOnResults: TWordListCallBack);
var
   L : TStringList; aArticle : TArticle; S  : String;
   R  : TJSRegexp;
begin
   if not assigned(aOnResults) then exit;
   aTerm:=UpperCase(aTerm);
   L:=TStringList.Create;
   try
     L.Sorted:=True;
     L.Duplicates:=dupIgnore;
     R:=TJSRegexp.New('\b(?:\w|-)+\b','g');
     For aArticle in BPMArticles do
       for S  in TJSString(aArticle.Title).match(R) do
         if pos(aTerm,Uppercase(S))>0 then L.Add(s);
         aOnResults(L);
   finally
     L.Free;
   end;
end;
```

This is a very simple loop over the array of article records: For every article, the Title field is split into words using the Javascript 'match' method of the String type: If the word contains the search term (*we check this case-insensitively*), we add it to the list: the list ignores duplicates, so we get each word only once. At the end we call the callback.
The search mechanism works in a completely similar way. It clears any previous results, loops over the article list, and if an article matches the search term, it is included in the result.

```
procedure TArticleSearcher.Search(const aTerm: string; const aIssue: String);
Var
   aIdx : integer;
   aArticle : TArticle;
V,IssueFilter : string;
I : integer;

begin
   if Not Assigned(ResultsElement) then exit;
   IssueFilter:=”;
   For I:=1 to Length(aIssue) do
     if Pos(aIssue[I],’0123456789_’)>0 then
       IssueFilter:=IssueFilter+V[i];
   ClearResultPanel;
   For aIdx:=0  to Length(BPMArticles)-1 do
     begin
       aArticle:=BPMArticles[aIdx];
       if aArticle.IsMatch(aTerm,IssueFilter) then
         ResultsElement.AppendChild(CreateArticleRow(aIdx,aArticle));
     end;
   ShowResultsPanel;
end;
```

At the end, the results panel is shown.
The `IsMatch` procedure which is used to determine if an article is matched, is a helper method for `TArticle`:

```
TArticleHelper = class  helper  for TArticle
   Function IsMatch (aTerm : String; aIssue  : string) : Boolean;
end;

function TArticleHelper.IsMatch(aTerm: String; aIssue: string): Boolean;
   begin aTerm:=UpperCase(aTerm);
   Result:=(aTerm=”) or ((Pos(aTerm,UpperCase(Author))>0)
                        or (Pos(aTerm,UpperCase(Title))>0));
   if Result and (aIssue<>”) then
     Result:=(aIssue=Issue);
end;
```

This must be implemented as a helper method, since the `TArticle` class is defined as an external class, and therefore its definition cannot contain pascal methods.
The `CreateArticleRow` method uses a string constant `DefaultPanel` with a HTML template to construct the actual HTML using a simple search and replace mechanism:

```
function TArticleSearcher.CreateArticleRow(aIdx: Integer; aArticle:  TArticle):
   TJSHTMLElement;
Var
   Panel  : String;

begin
   Result:=TJSHTMLElement(Document.createElement(’div’));

   Panel:=StringReplace(DeeaultPanel,’{{issue}}’,aArticle.Issue,[rfReplaceAll]);
   Panel:=StringReplace(Panel,’{{page}}’,IntToStr(aArticle.Page),[rfReplaceAll]);
   Panel:=StringReplace(Panel,’{{title}}’,aArticle.Title,[rfReplaceAll]);
   Panel:=StringReplace(Panel,’{{author}}’,aArticle.Author,[rfReplaceAll]);

   Result.dataset[’issue’]:=aArticle.Issue; Result.dataset[’page’]:=IntToStr(aArticle.Page);
   Result.dataset[’articleid’]:=intToStr(aIdx); Result.dataset[’title’]:=aArticle.Title;
   Result.AddEventListener(’click’,@OnArticleClick); Result.innerHTML:=Panel;
end;
```

Finally, the OnClick handler for the result element collects some data to set up a TPageInfo record, which is then used to display the correct PDF page:

```
procedure TArticleSearcher.OnArticleClick(aEvent : TJSEvent);

var
  aPage : TPageInfo;

begin
  aPage:=Default(TPageInfo);
  With TJSHTMLElement(aEvent.currentTargetElement) do
    begin
      aPage.Issue:=dataset['issue'];
      aPage.Page:=StrToIntDef(dataset['page'],-1);
      aPage.Title:=dataset['title']; aPage.useIssue:=True;
    end;
  ShowPDFPage(aPage);
end;
```

The ShowPDFPage uses the ShowPDFPageEvent event handler to actually show the PDF on the correct page. And with this, our offline search mechanism is ready.

7 DETECTING OFFLINE STATUS AND SHOWING A PDF

The offline search mechanism has to be activated when the navigator has no access to internet: We implemented the OffLine property for this in the TPDFSearchControl. But this property has not yet been set to a correct value. Luckily, the browser has a property that indicates whether it is currently online or offline: The window.Navigator.onLine property indicates whether the browser is currently online or offline. What is more, the Window class implements 2 events
'online' and 'offline' that are triggered when the browser goes online or offline, respectively. So we can use AddEventListener to install an event handler and react to changes in online or offline status.
This is done in the DetectOffLine routine in the application class. It does 2 things: it detects whether the application was started by double clicking the index.html file in the file explorer or whether it was started from a website. The result is stored in the IsLocal property, and the online status is stored in the IsOffLine property:

```
procedure TBPMLibraryApplication.DetectOffline;

   Procedure updateOnlineStatus(event : TJSEvent);
   begin
     IsOffLine:=not window.Navigator.onLine;
   end;

begin
   IsLocal:=Copy(window.location.protocol,1,4)='file';
   IsOffLine:=not window.Navigator.onLine;
   window.addEventListener('online',@updateOnlineStatus);
   window.addEventListener('offline',@updateOnlineStatus);
end;
```

NOTE that the online and offline eventhandler is used to update the IsOffLine property.
The IsOffLine property of the application object has a setter, and is used to propagate the value to the searchcontrol:

```
procedure TBPMLibraryApplication.SetIsOffLine(AValue: Boolean);
begin
   if FIsOffLine=AValue then Exit;
   FIsOffLine:=AValue;
   FSearchPane.OffLine:=FisOffline;
end;
```

Thus, the search mechanism will know whether to search locally or remote.
As we've seen, the search mechanism only has an event which it must use to open a PDF and display a certain page.
The event is set in the main application to the following event handler:

```
procedure TBPMLibraryApplication.HandleShowPDFPage(aPage: TPageInfo);

   Function IsSameAsLastPDF : Boolean;
   begin
     if aPage.useIssue then
       Result:=(FLastIssue<>”) and (FLastIssue=aPage.Issue)
     else
       Result:=(FLastPDF<>”) and (FLastPDF=aPage.FileName)
   end;

begin
   // local search or PDF already loaded
   if IsSameAsLastPDF then
     FViewer.ShowPage(aPage.Page)
     else if Not HaveLocalFile(aPage) then
       LoadRemotePDF(aPage)
     else
       LoadLocalPDF(aPage);
end;
```

If the PDF is already loaded (*checked in* `IsSameAsLastPDF`), then the viewer panel is simply instructed to show the new page. If the PDF is not yet loaded, then it must be loaded before the correct page can be shown.
If the page was loaded from local disk (as is the case on the USB-stick version of the Blaise Pascal magazine), then it is loaded from disk using `LoadLocalPDF`, else it is loaded using `LoadRemotePDF`.
The `HaveLocalFile` function uses the IsLocal property that was initialized by the application to decide whether a file can be loaded from disk or not:
If `IsLocal` is false, we know the page is loaded from a website, and the PDF files will not be available locally. But if `IsLocal` is `true`, it still can be that the PDF is not available locally:
When an online search was performed, the search result could have returned a PDF that is not available locally. To cater for that case we must check the list of available articles to see if the requested issue is present, and this is done by checking the issue number in the list of articles:

```
function TBPMLibraryApplication.HaveLocalFile(aPage : TPageInfo) : Boolean;

Var
   aArticle : TArticle;

begin
   Result:=IsLocal;
   if Result then
     begin
     // Determine if we have the PDF locally
       Result:=False;
       for aArticle in BPMArticles do
         if (aPage.Issue=aArticle.Issue) then
           Exit(True);
     end;
end
```

When the PDF is available locally, we load it using the trick shown in the first article in this series: a script tag is inserted which defines contents of the PDF as a Javascript variable (`pdfData`):

```
var
  pdfData : String; external name 'pdfData';

procedure TBPMLibraryApplication.LoadLocalPDF(aPage: TPageInfo);

  Procedure DoShowPage;
  begin
     FViewer.ShowPage(aPage.Page);
  end;

  function DoLoaded(Event : TEventListenerEvent) : Boolean;
  var
     Src : TPDFSource;

  begin
     Src:=TPDFSource.new;
     Src.Data:=pdfData;
     Fviewer.StartPDFRender(Src,@DoShowPage);
  end;

Var
  Script : TJSHTMLScriptElement;
  FN : String;

begin
  Script:=TJSHTMLScriptElement(document.CreateElement('script'));
  FN:=aPage.FileName;
  if Pos('file://',FN)=1 then
     Delete(FN,1,7);
  else if FN='' then
     FN:='issues/issue'+aPage.Issue+'.js';
  Script.Src:=FN;
  Script.Onload:=@DoLoaded;
end;
```

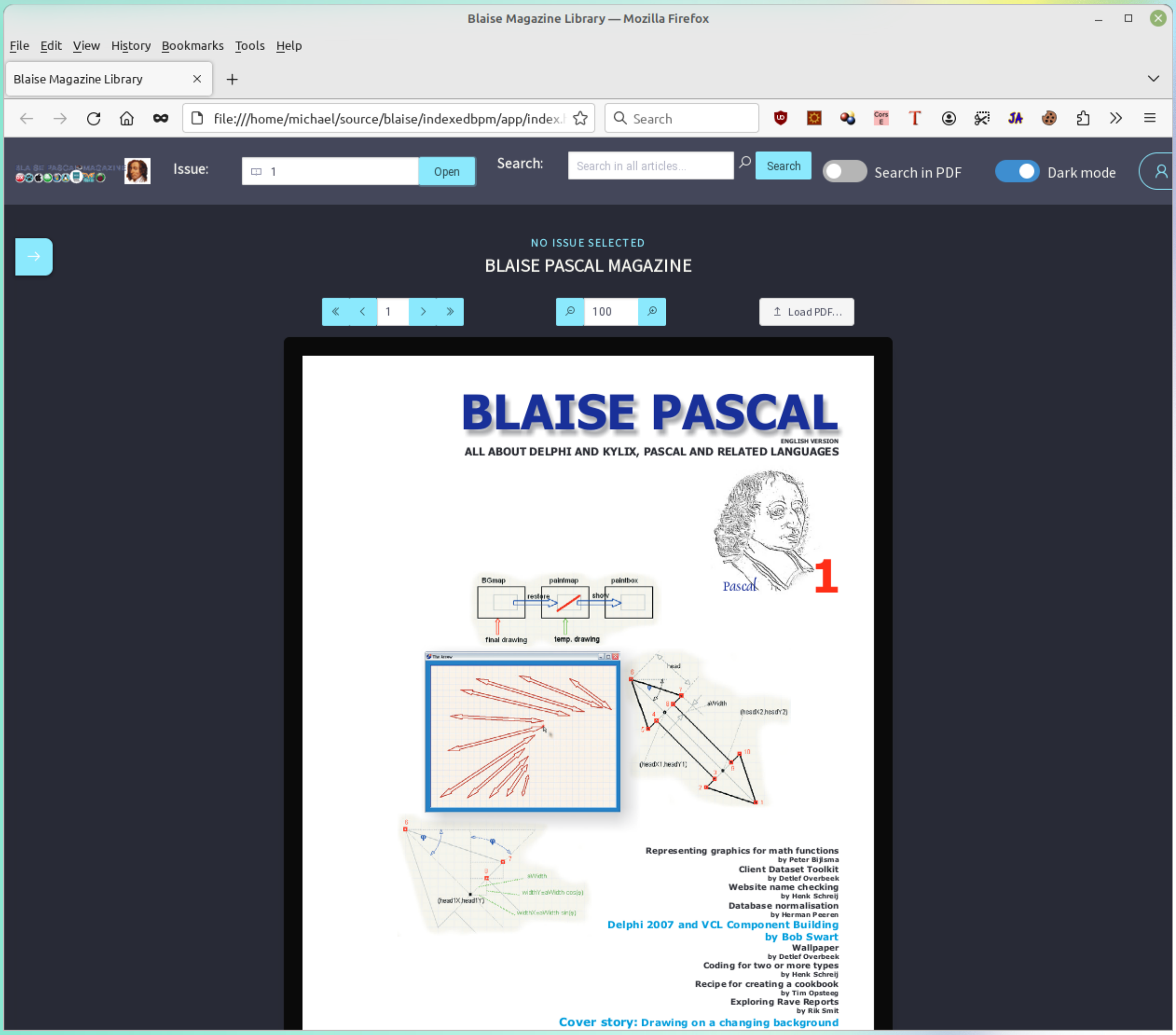
Finally, the `LoadRemotePDF` routine loads a PDF from the server. It needs to distinguish between a call where only the issue number is given (*the user requested simply to see an issue*) or when the PDF filename is known. In the former case it uses the `/issue/` URL which we showed in the beginning of the article, in the latter case the `/pdf/` URL is used:

```
procedure TBPMLibraryApplication.LoadRemotePDF(aPage: TPageInfo);

   Procedure DoShowPage;
   begin
     FViewer.ShowPage(aPage.Page);
   end;

var
   Src : TPDFSource;

begin
   Src:=TPDFSource.new;
   if aPage.useIssue then
     Src.url:=ServerURL+’issue/’+aPage.Issue
   else
     Src.url:=ServerURL+’pdf/’+aPage.FileName+’?token=’+ encodeURIComponent(FLogin.Token);
     FLastPDF:=aPage.FileName;
     FViewer.StartPDFRender(Src,@DoShowPage);
end;
```

And with this routine, the application is ready to go. The application with an issue loaded locally is shown in *figure 1 on page 21 on the next page*.

8 CONCLUSION

In this article, we've shown how to make a real-world application using PAS2JS that can work both online and offline and adapts itself: we refactored techniques introduced in the previous articles in this series. The application can still be improved: for instance the online/offline status can be made visual - e.g. changing a background color.
The application times out after 30 minutes, but no warning is given: this can also be improved. As with all software, the work is never finished...

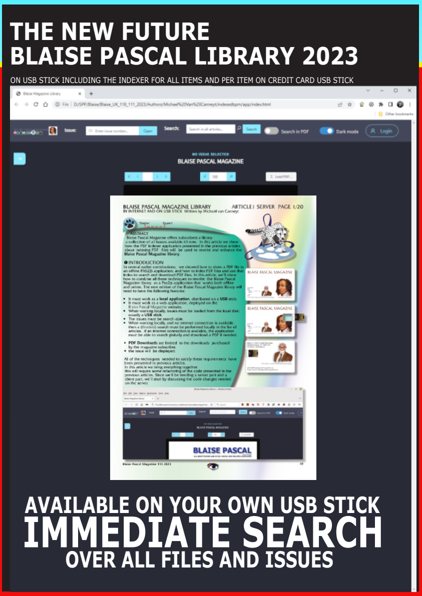


ADVERTISEMENT

1

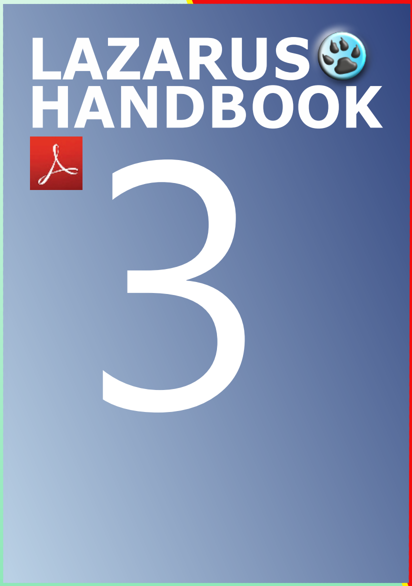
BLAISE PASCAL MAGAZINE 110 / 111
Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

AI-enabled brain scanner reads thoughts
Image Classifier with loading and testing a pre-trained model
Delphi Community version for Delphi 11
Jim McKeeth leaving Embarcadero/Delphi
The Number guessing project
BLAISE PASCAL MAGAZINE LIBRARY By internet and on USB Stick
The Library kit for BPM has been extended with new features: Search over ALL 111 issues and per issue.
Lazarus compiling Delphi code
Lazarus for Visual Studio
Debugging with the new debugger in Lazarus - lessons part 2
FastReport for Lazarus on LINUX
in a Trial and as Professional version

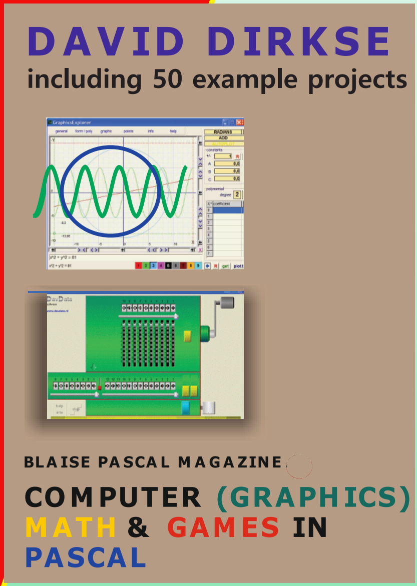

THE NEW FUTURE
BLAISE PASCAL LIBRARY 2023
ON USB STICK INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK

AVAILABLE ON YOUR OWN USB STICK
IMMEDIATE SEARCH
OVER ALL FILES AND ISSUES

2


LAZARUS HANDBOOK
3

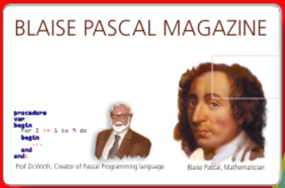
4
LEARN TO PROGRAM USING LAZARUS
HOWARD PAGE-CLARK


DAVID DIRKSE
including 50 example projects

BLAISE PASCAL MAGAZINE
COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

INCLUDING
THE NEW LAZARUS
PDF KIT INDEXER

1. One year Subscription
2. The newest LIB Stick
   - All issues 1-111
   - On Credit Card
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


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

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

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands
Prof Dr.Wirth, Creator of Pascal Programming language
editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu

SUPER
PACK
2023

PRICE € 160
NORMAL PRICE € 275

D11 Starter Expert

ABSTRACT

DropMaster was created by Ray Konopka is a solution for drag and drop support with applications for Microsoft Windows. This is a very useful and often necessary tool to make the user interface interactive and logic for the End-user. It should be in any application because even internally in an application it can be very convenient or necessary.

INTRODUCTION

This is a collection of components for bringing drag-and-drop capabilities across applications to Delphi programs running under Microsoft Windows. Drag and drop of text-based data, images, and customized formats is supported.
More than 40 sample apps are included with it, representing the findings of in-depth research into the drag-and-drop functionality of numerous well-known commercial applications. This is quite helpful, particularly for beginners.

The most recent Delphi 11 Alexandria 11.3 as well as likely future releases are supported by the Current Release 2.5.2, which started with RAD Studio (Delphi) 2009.

It's a great value. Particularly when you are aware of the tactic being used: extremely complicated: at $99, you'll quickly receive value for your money.

INSTALLING

As a favor to potential clients interested in inspecting the product's components, this component suite provides a trial edition.
The components in this trial edition are the same as the released versions, as is typical in most trial versions of VCL-based components, with the exception that they can only be used concurrently with Delphi.
This means that an application using any of the trial edition's components **can only be launched from Delphi**.

Make sure you have administrator privileges, and it is preferable to place the projects on a different disk than C:. The installation process is incredibly straightforward; you do not need to follow the usual steps.
You can start using DropMaster in RAD Studio, Delphi once the installation program is finished. The "DropMaster" page will appear on the component palette after restarting RAD Studio. *See figure 1:*

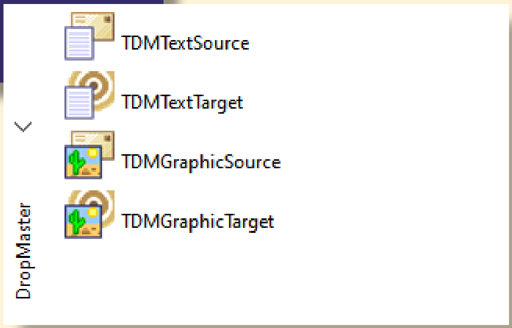
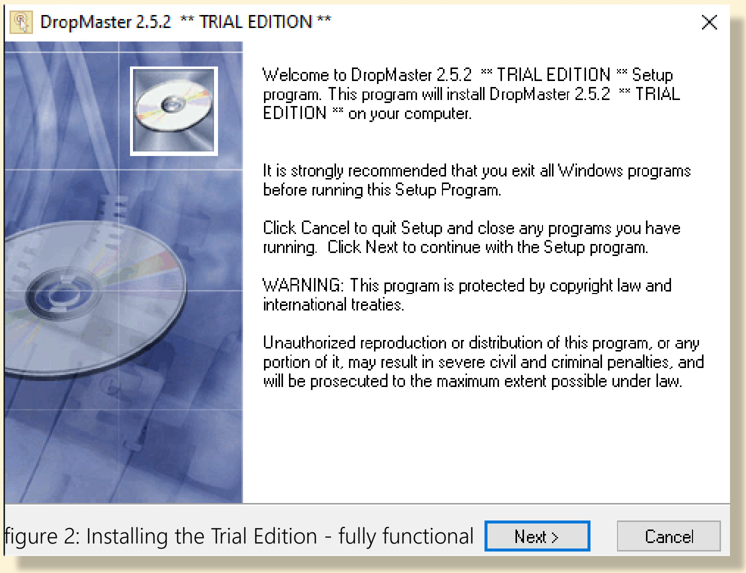


figure 1: Overview of the components group.



Welcome to DropMaster 2.5.2 ** TRIAL EDITION ** Setup program. This program will install DropMaster 2.5.2 ** TRIAL EDITION ** on your computer.

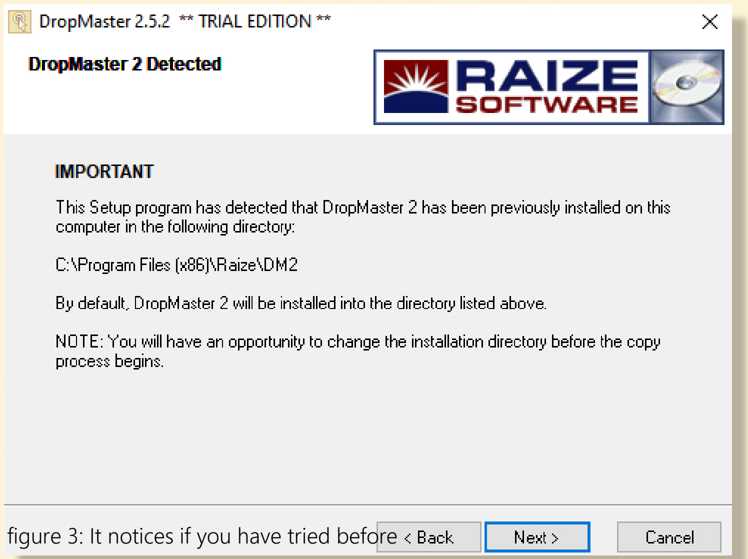
It is strongly recommended that you exit all Windows programs before running this Setup Program.

Click Cancel to quit Setup and close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

figure 2: Installing the Trial Edition - fully functional



DropMaster 2 Detected

IMPORTANT

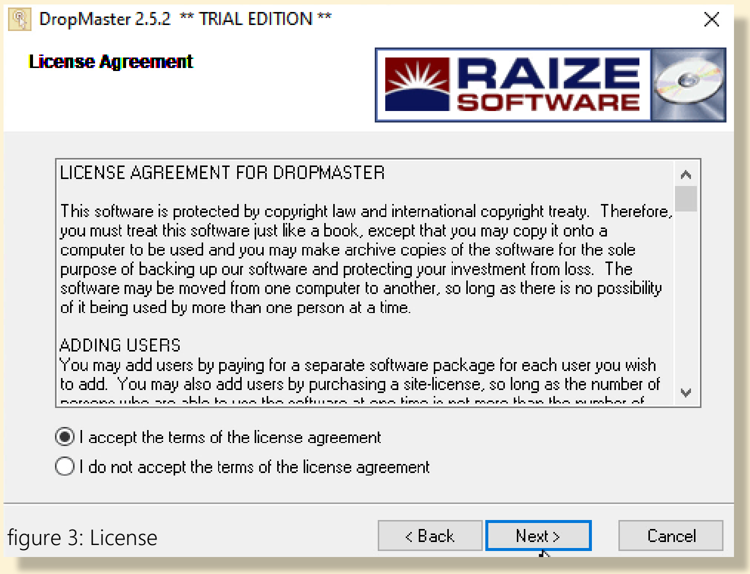
This Setup program has detected that DropMaster 2 has been previously installed on this computer in the following directory:

C:\Program Files (x86)\Raize\DM2

By default, DropMaster 2 will be installed into the directory listed above.

NOTE: You will have an opportunity to change the installation directory before the copy process begins.

figure 3: It notices if you have tried before



License Agreement

LICENSE AGREEMENT FOR DROPMASTER

This software is protected by copyright law and international copyright treaty. Therefore, you must treat this software just like a book, except that you may copy it onto a computer to be used and you may make archive copies of the software for the sole purpose of backing up our software and protecting your investment from loss. The software may be moved from one computer to another, so long as there is no possibility of it being used by more than one person at a time.

ADDING USERS
You may add users by paying for a separate software package for each user you wish to add. You may also add users by purchasing a site-license, so long as the number of

[✓ I accept the terms of the license agreement]
I do not accept the terms of the license agreement

figure 3: License

As is typically the case with Rays products, it is incredibly effective and helpful. It makes sense given that he wrote the book on creating components. He is the real inventor in addition to being incredibly creative.

The pictures in this post demonstrate the procedure and will persuade you of how easy it is to use.

UNINSTALLING

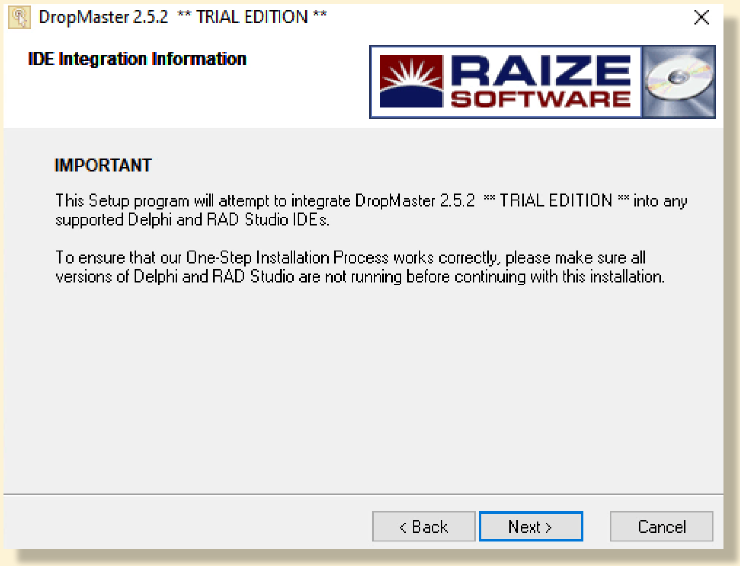
Removing the components from the RAD Studio component palette:
close all files and projects, and select **Component|Install Packages**... to display the Packages page in the Project Options dialog.
Select the "**DropMaster 2.x**" package from the **Design Packages** list and click the **Remove** button.
A message box will be displayed to confirm your request-press OK.

Next, depending on the IDE (*Integrated Development Environment*), you may be asked if a runtime package should be removed from the **Runtime Packages list.**
If so, click OK to remove the runtime package.Close the **Project Options** dialog box by clicking the OK button.
Repeat the above steps for each IDE that is using DropMaster.

And to make it very complete:
Removing all of the component files from your hard disk
At this point, all IDE's are no longer using DropMaster.

To remove the DropMaster files from your hard disk the **Add/Remove Programs** icon from the Control Panel.

Next, select the "DropMaster 2.x" entry from the list of installed programs, and then click the Remove button. (*versions of Delphi*).


DropMaster 2.5.2 ** TRIAL EDITION **

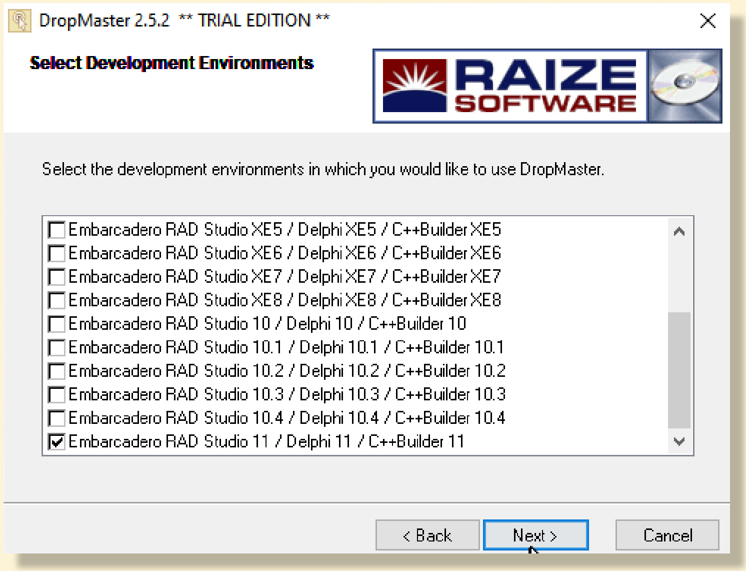
IDE Integration Information

IMPORTANT

This Setup program will attempt to integrate DropMaster 2.5.2 ** TRIAL EDITION ** into any supported Delphi and RAD Studio IDEs.

To ensure that our One-Step Installation Process works correctly, please make sure all versions of Delphi and RAD Studio are not running before continuing with this installation.

figure 5: Do NOT run any RAD STUDIO or Delphi during the install process

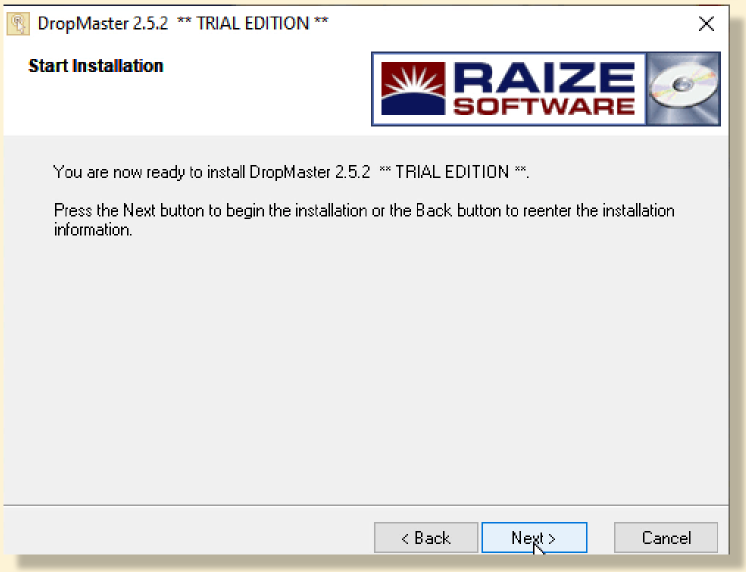

DropMaster 2.5.2 ** TRIAL EDITION **

Select Development Environments

Select the development environments in which you would like to use DropMaster.

- [ ] Embarcadero RAD Studio XE5 / Delphi XE5 / C++Builder XE5
- [ ] Embarcadero RAD Studio XE6 / Delphi XE6 / C++Builder XE6
- [ ] Embarcadero RAD Studio XE7 / Delphi XE7 / C++Builder XE7
- [ ] Embarcadero RAD Studio XE8 / Delphi XE8 / C++Builder XE8
- [ ] Embarcadero RAD Studio 10 / Delphi 10 / C++Builder 10
- [ ] Embarcadero RAD Studio 10.1 / Delphi 10.1 / C++Builder 10.1
- [ ] Embarcadero RAD Studio 10.2 / Delphi 10.2 / C++Builder 10.2
- [ ] Embarcadero RAD Studio 10.3 / Delphi 10.3 / C++Builder 10.3
- [ ] Embarcadero RAD Studio 10.4 / Delphi 10.4 / C++Builder 10.4
- [✓] Embarcadero RAD Studio 11 / Delphi 11 / C++Builder 11

figure 6: The Menu

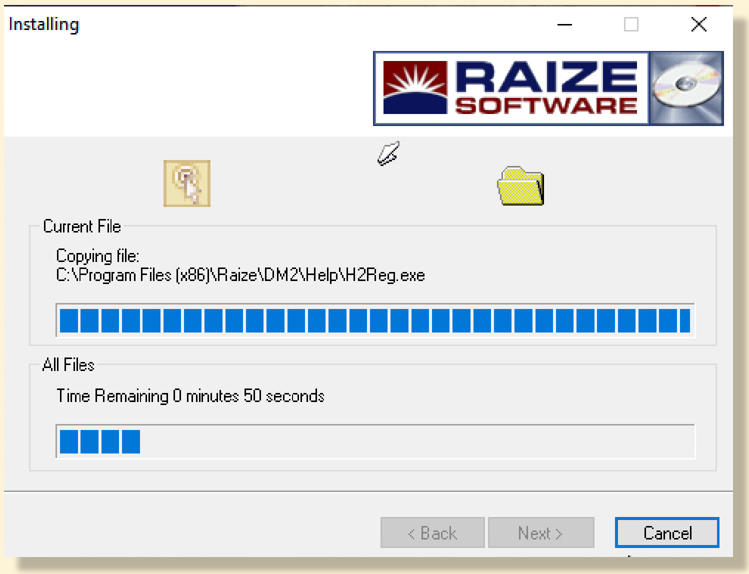

DropMaster 2.5.2 ** TRIAL EDITION **

Start Installation

You are now ready to install DropMaster 2.5.2 ** TRIAL EDITION **.

Press the Next button to begin the installation or the Back button to reenter the installation information.

figure 7: Last warning before installing



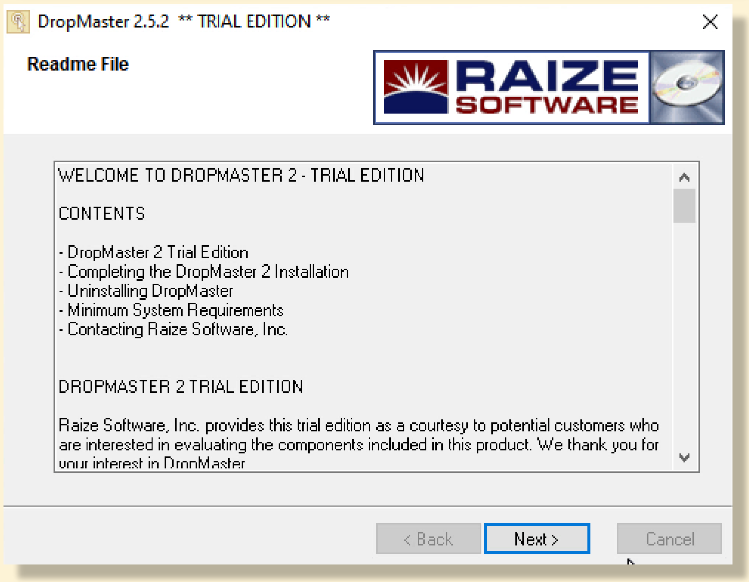
Current File

Copying file:
C:\Program Files (x86)\Raize\DM2\Help\H2Reg.exe

All Files

Time Remaining 0 minutes 50 seconds

figure 8: The progress of the installation



WELCOME TO DROPMASTER 2 - TRIAL EDITION

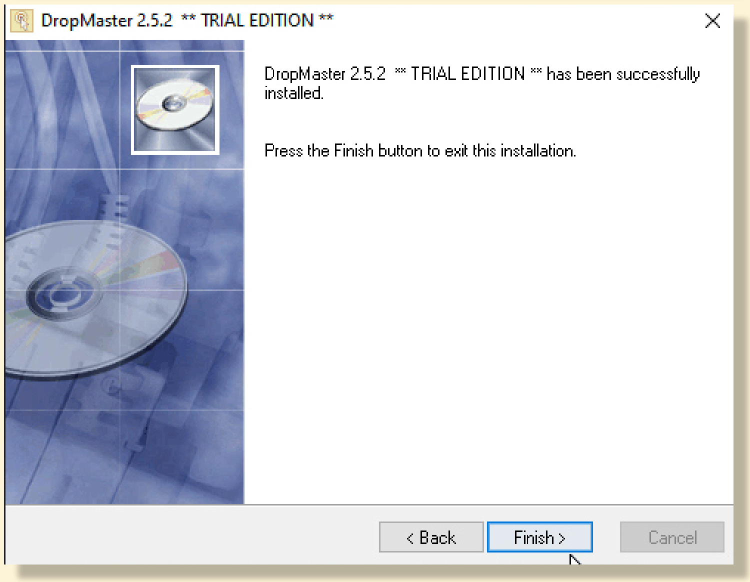
CONTENTS

- DropMaster 2 Trial Edition
- Completing the DropMaster 2 Installation
- Uninstalling DropMaster
- Minimum System Requirements
- Contacting Raize Software, Inc.

DROPMASTER 2 TRIAL EDITION

Raize Software, Inc. provides this trial edition as a courtesy to potential customers who are interested in evaluating the components included in this product. We thank you for your interest in DropMaster

figure 9: Overview of the contents



DropMaster 2.5.2 ** TRIAL EDITION ** has been successfully installed.

Press the Finish button to exit this installation.

figure 10: Ready for use

THE COMPONENTS

TDMTextTarget

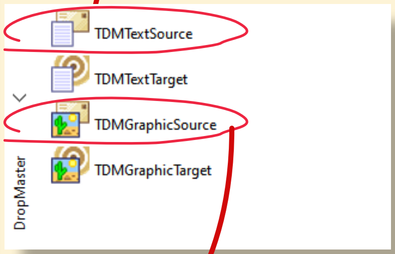
The main component to accept data dragged into your program from another is called `TDMTextTarget`.
To respond to dropped data, just assign the `AcceptorControl` property and create an `OnDrop` event handler.
Text, RTF, HTML, lists of files, and URLs can all be accepted by `TDMTextTarget`.
In addition to accepting text, this component can also accept other arbitrary forms.

All dropped formats are accessible through the `OnDrop` event.
I will show some example projects on the next pages.
Of Course you can alter these examples to your needs.

But make sure you safe the project in a different directory: NOT C: -
Windows protects that disc and will not allow you to do things normally.
You will get bad results!

TDMTextSource

Data that needs to be dragged to another application is served up via the `TDMTextSource` component. To use, simply provide a control the `DonorComponent` property, call the `Execute` method when the control's `OnMouseDown` event has detected the drag.
To serve up any customized material, use the `Text` attribute.
The **simultaneous serving of several formats** is another function of this component.



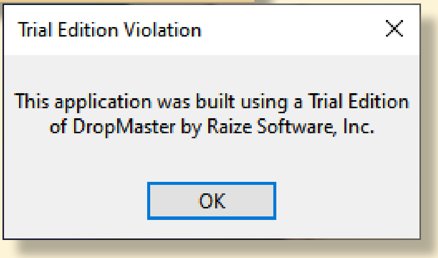
TDMGraphicTarget

The `TDMGraphicTarget` component is utilized to accept photos as well as images that are dragged from another application.
The developer simply needs to provide a **placeholder control** for the graphic data that is received because the component interacts with numerous formats for convenience, including DIBs, bitmap handles, and metafiles.

TDMGraphicSource

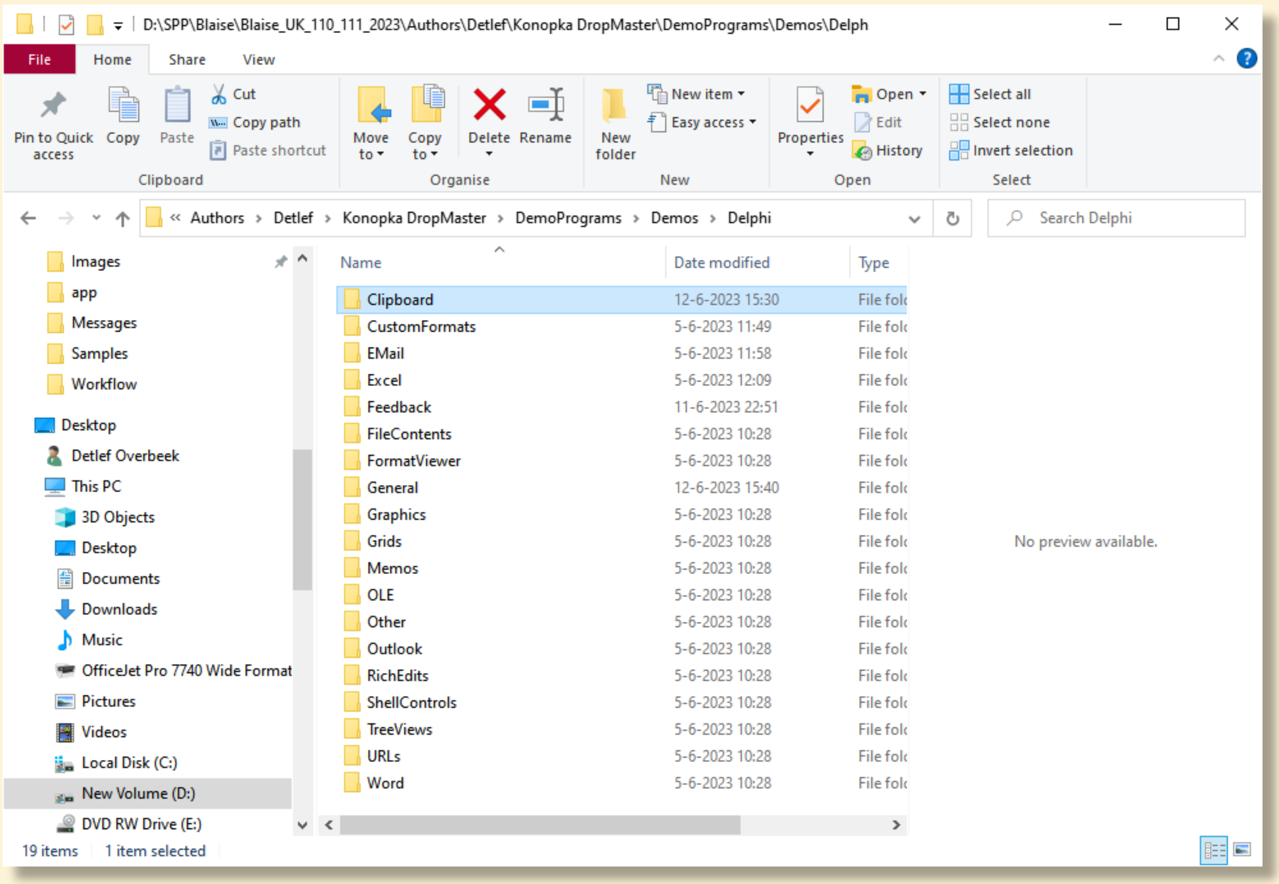
Drag and drop commonly uses graphic images as a data type.
The `TDMGraphicSource` is comparable to the `TDMTextSource`, with the exception that it has `DonorImage` and `Picture` attributes instead of the `TDMTextSource's DonorComponent` and `Text` properties.
It is simple to serve up a picture in drag data by assigning it to a property, recognizing the drag, and then invoking `Execute`.

Run the projects from inside Delphi in debug mode, or else



Trial Edition Violation

This application was built using a Trial Edition of DropMaster by Raize Software, Inc.



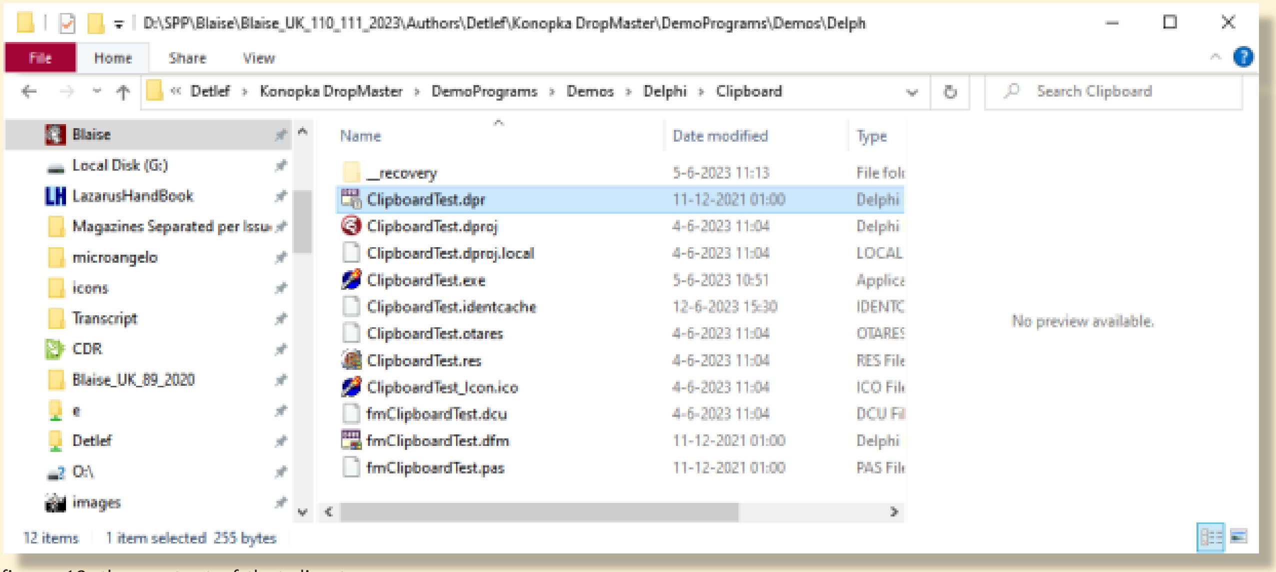
D:\SPP\Blaise\Blaise_UK_110_111_2023\Authors\Detlef\Konopka DropMaster\DemoPrograms\Demos\Delph

« Authors › Detlef › Konopka DropMaster › DemoPrograms › Demos › Delphi

Name	Date modified	Type
Clipboard	12-6-2023 15:30	File folc
CustomFormats	5-6-2023 11:49	File folc
EMail	5-6-2023 11:58	File folc
Excel	5-6-2023 12:09	File folc
Feedback	11-6-2023 22:51	File folc
FileContents	5-6-2023 10:28	File folc
FormatViewer	5-6-2023 10:28	File folc
General	12-6-2023 15:40	File folc
Graphics	5-6-2023 10:28	File folc
Grids	5-6-2023 10:28	File folc
Memos	5-6-2023 10:28	File folc
OLE	5-6-2023 10:28	File folc
Other	5-6-2023 10:28	File folc
Outlook	5-6-2023 10:28	File folc
RichEdits	5-6-2023 10:28	File folc
ShellControls	5-6-2023 10:28	File folc
TreeViews	5-6-2023 10:28	File folc
URLs	5-6-2023 10:28	File folc
Word	5-6-2023 10:28	File folc

19 items 1 item selected

figure 11: Overview of available projects



D:\SPP\Blaise\Blaise_UK_110_111_2023\Authors\Detlef\Konopka DropMaster\DemoPrograms\Demos\Delph

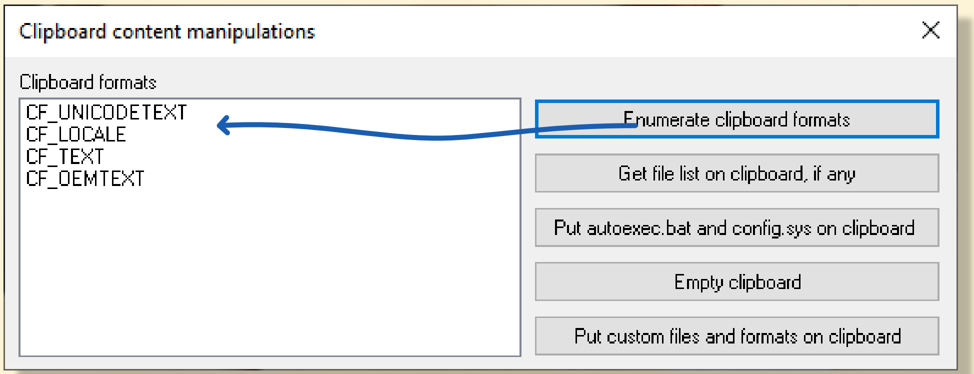
« Detlef › Konopka DropMaster › DemoPrograms › Demos › Delphi › Clipboard

Name	Date modified	Type
__recovery	5-6-2023 11:13	File fol
ClipboardTest.dpr	11-12-2021 01:00	Delphi
ClipboardTest.dproj	4-6-2023 11:04	Delphi
ClipboardTest.dproj.local	4-6-2023 11:04	LOCAL
ClipboardTest.exe	5-6-2023 10:51	Applica
ClipboardTest.identcache	12-6-2023 15:30	IDENTC
ClipboardTest.otares	4-6-2023 11:04	OTARES
ClipboardTest.res	4-6-2023 11:04	RES File
ClipboardTest_Icon.ico	4-6-2023 11:04	ICO File
fmClipboardTest.dcu	4-6-2023 11:04	DCU Fi
fmClipboardTest.dfm	11-12-2021 01:00	Delphi
fmClipboardTest.pas	11-12-2021 01:00	PAS File

12 items 1 item selected 255 bytes

figure 12: the content of that directory.

Here you can see the content of the directory of the CLIPBOARD PROJECT.
you would only buy this component suite for this app it is already worth it. You could create a small app where you can see how you can make a list of products and names which can enter it into the subject of your email automatically

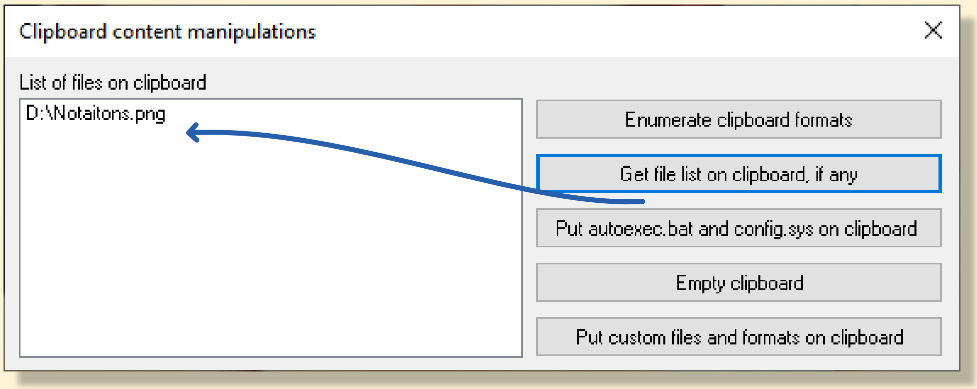

Clipboard content manipulations

Clipboard formats

```
CF_UNICODETEXT
CF_LOCALE
CF_TEXT
CF_OEMTEXT
```

Enumerate clipboard formats
Get file list on clipboard, if any
Put autoexec.bat and config.sys on clipboard
Empty clipboard
Put custom files and formats on clipboard

figure 13: Listing the possible sorts of text

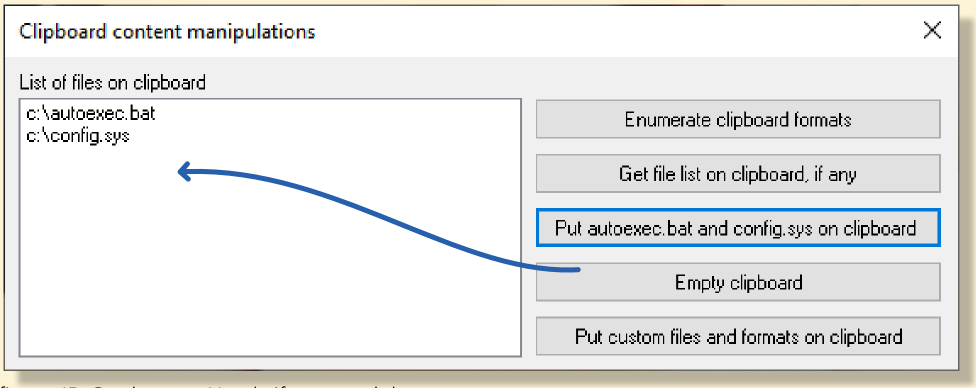

Clipboard content manipulations

List of files on clipboard

```
D:\Notaitons.png
```

Enumerate clipboard formats
Get file list on clipboard, if any
Put autoexec.bat and config.sys on clipboard
Empty clipboard
Put custom files and formats on clipboard

figure 14: Found a PNG file In memory


Clipboard content manipulations

List of files on clipboard

```
c:\autoexec.bat
c:\config.sys
```

Enumerate clipboard formats
Get file list on clipboard, if any
Put autoexec.bat and config.sys on clipboard
Empty clipboard
Put custom files and formats on clipboard

figure 15: Can be very Handy if you need them

Project Overview Form

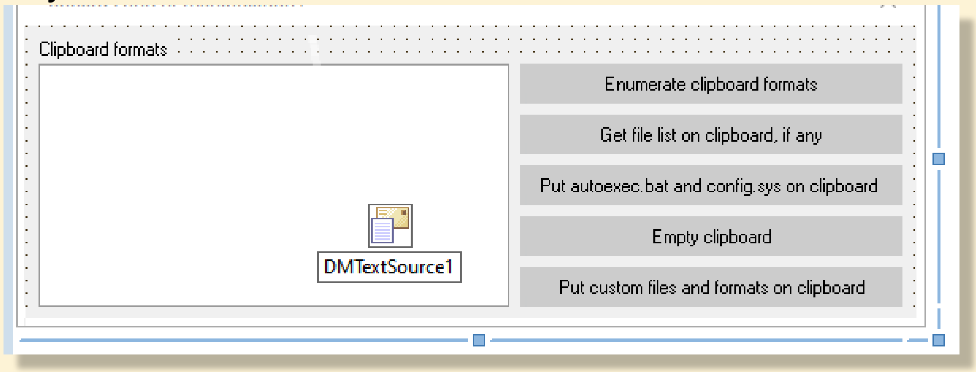
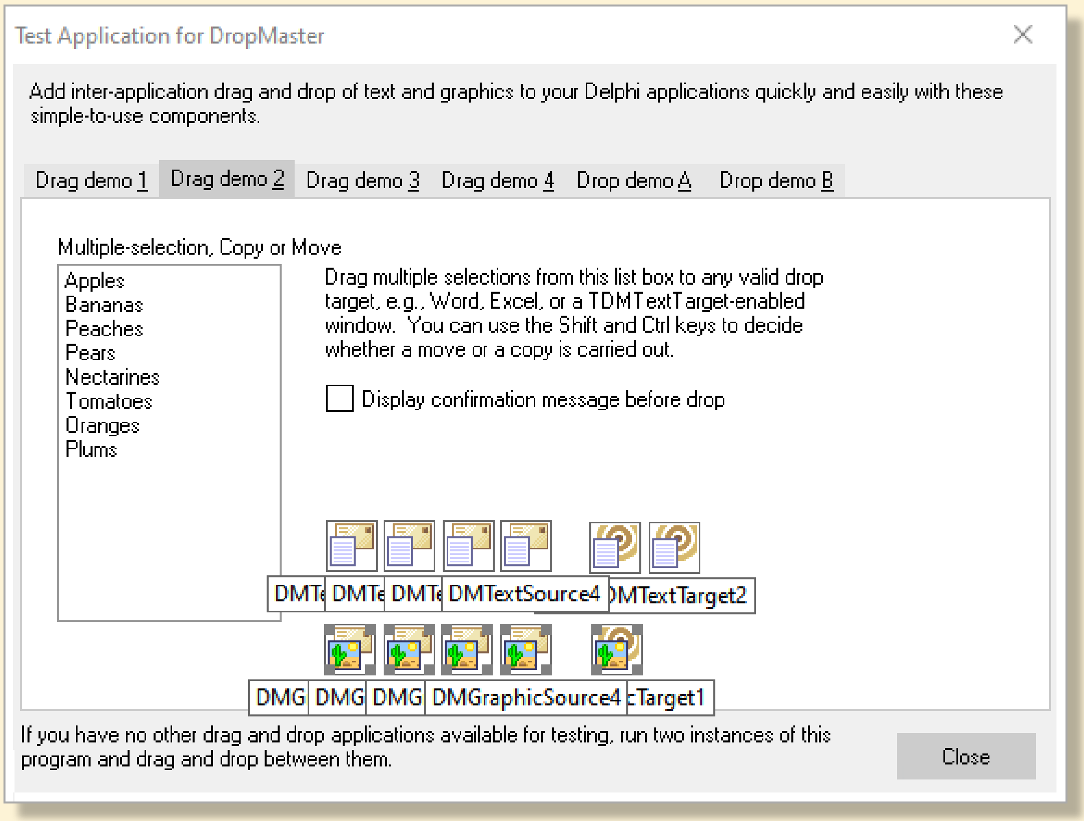

Clipboard formats
DMTextSource1
Enumerate clipboard formats
Get file list on clipboard, if any
Put autoexec.bat and config.sys on clipboard
Empty clipboard
Put custom files and formats on clipboard

figure 16: Project overview Clipboard Form inside Delphi

```
 unit fmClipboardTest;

{ Example application for DropMaster.

  Demonstrates using the DMUtil clipboard format helper functions to manipulate
  actual clipboard data rather than drag-and-drop data.  In fact, this example
  has nothing to do with drag-and-drop, and contains no DropMaster components!

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, DMComps;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    btnEnumClipboard: TButton;
    btnCFHDROP: TButton;
    btnPutClipboard: TButton;
    btnEmptyClipboard: TButton;
    Label1: TLabel;
    btnCustomFiles: TButton;
    DMTextSource1: TDMTextSource;
    procedure btnEnumClipboardClick(Sender: TObject);
    procedure btnCFHDROPClick(Sender: TObject);
    procedure btnPutClipboardClick(Sender: TObject);
    procedure btnEmptyClipboardClick(Sender: TObject);
    procedure btnCustomFilesClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses
  ClipBrd, DMUtil, ActiveX;

{$R *.DFM}

procedure TForm1.btnEnumClipboardClick(Sender: TObject);
// Fill the list box with the name of every format currently available
// on the clipboard
var
  aFmt: DWORD;
  j: Integer;
begin
  ListBox1.Clear;

  for j := 1 to Clipboard.FormatCount do    // Iterate
  begin
    aFmt := Clipboard.Formats[j-1];
    ListBox1.Items.Add(ClipboardFormatDisplayname(aFmt));
  end;    // for

  Label1.Caption := 'Clipboard formats';
end;
```

```
procedure TForm1.btnCFHDROPClick(Sender: TObject);
// Get the list of files, if any, available on the clipboard
// You'd use this, e.g., to do a paste operation for files.
var
 aSL: TStringList;
 S: AnsiString;
begin
 ClipBoard.Open;
 try
  s := GetHandleDataToString(Clipboard.GetAsHandle(CF_HDROP));
  aSL := FileListFromHDROP(s);
  // Now we have the file list.  We could paste these, or make shortcuts.
  // Here we just put the names into a listbox.
  try
   ListBox1.Clear;
   ListBox1.Items.AddStrings(aSL);
  finally
   aSL.Free;
  end;
 finally
  Clipboard.Close;
 end;

 Label1.Caption := 'List of files on clipboard';
end;

procedure TForm1.btnPutClipboardClick(Sender: TObject);
// Put a list of files on the clipboard for pasting or pasting as shortcut
var
 aSL: TStringList;
 s, t: AnsIString;
begin
 aSL := TStringList.create;
 try
  // Put two file names in the list
  aSL.Add('c:\autoexec.bat');
  asl.add('c:\config.sys');
  // Generate both CF_HDROP and Shell IDList Array formats
  s := HDropFromFileList('', aSL);
  t := ShellIDListFromFileList('', aSL);
 finally
  aSL.Free;
 end;

 // Put the formats on the clipboard
 Clipboard.Open;
 try
  // CF_DROP, so we can paste files
  Clipboard.SetAsHandle(CF_HDROP,
   SetHandleDataFromString(s));
  // Shell IDList Array, so we can paste shortcuts also
  Clipboard.SetAsHandle(ClipboardFormatFromString('Shell IDList Array'),
   SetHandleDataFromString(t));
 finally
  Clipboard.Close;
 end;
end;

procedure TForm1.btnEmptyClipboardClick(Sender: TObject);
// Clear the clipboard
begin
 ClipBoard.Clear;

 ListBox1.Clear;
 Label1.Caption := 'Clipboard formats';
end;
```

```
 procedure TForm1.btnCustomFilesClick(Sender: TObject);
// Put some custom stuff on the clipboard (to paste files that don't really exist)
// Works just like the FileContentsTest* demos
var
  sizes: array of integer;
  aSL: TStringList;
  bSL: TStringList;
  i: Integer;
  j: integer;
  aDataObject: IDataObject;
begin
  aSL := TStringList.Create;
  bSL := TStringList.Create;

  try
   // File names
   aSL.Add('first file.txt');
   aSL.Add('second file.txt');

   // File contents
   bSL.Add('contents of first file');
   bSL.Add('contents of second file');

   // Get sizes of contents
   SetLength(Sizes, bSL.Count);
   for I := 0 to bSL.Count - 1 do     // Iterate
   begin
     Sizes[i] := length(bSL[i]);
   end;     // for

   // Set up the data object.  Use the internal support in TDMTextSource to do this.
   With DMTextSource1, CustomFormatData do
   begin
     Clear;  // Empty the data formats

     // Pasting nonexistent files needs FileGroupDescriptor and FileContents formats.
     Add-format(DMFileGroupDescriptorFormatName,
     FileGroupDescriptorFromFileListEx('', aSL, sizes));

     // FileContents is an indexed format
     for j := 0 to bSL.Count-1 do
     begin
       AddFormatEx('FileContents', bSL[j], TYMED_HGLOBAL, j);
       // Make sure we don't get a trailing null in the content items
       // Items[Count-1] is the TCustomFormatData we just added.
       Items[Count-1].AllowTrailingNull := false;
     end;

   end;

    // Make a data object and put it on the clipboard.
    aDataObject := TTextDataObject.CreateExWithFormats(DMTextSource1.CustomFormatData,
                                                                     DMTextSource1);
    OleSetClipboard(aDataObject);

  finally
    // Clean up
    aSL.Free;
    bSL.Free;
  end;
end;

end.
```


Test Application for DropMaster

Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag demo 1 | Drag demo 2 | Drag demo 3 | Drag demo 4 | Drop demo A | Drop demo B

Multiple-selection, Copy or Move

Apples
Bananas
Peaches
Pears
Nectarines
Tomatoes
Oranges
Plums

Drag multiple selections from this list box to any valid drop target, e.g., Word, Excel, or a TDMTextTarget-enabled window. You can use the Shift and Ctrl keys to decide whether a move or a copy is carried out.

☐ Display confirmation message before drop

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

figure 17: Project overview TEST APPLICATION of various examples. Form inside Delphi

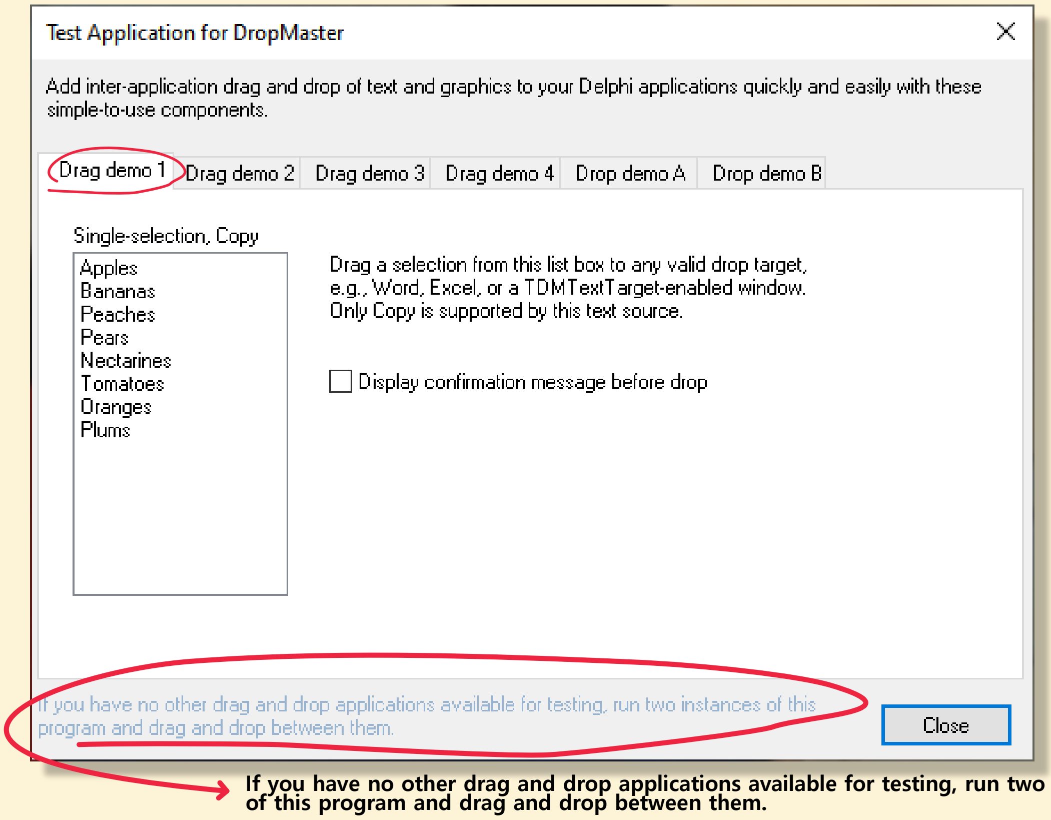
Demonstrates various techniques in the use of `TDMTextSource`, `TDMTextTarget`, `TDMGraphicSource`, `TDMGraphicTarget`.

The main point here: see how little code is needed.
Most of the code here was written by Delphi itself!
And the portion that was manually coded is mainly for user-interface purposes.

```
interface

uses
 Windows, Types, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, ComCtrls, Buttons, Grids, ExtCtrls, DMComps;

 ...

 var
 Form1: TForm1;

implementation

{$R *.dfm}

uses
 TypInfo;
 procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
// Detect the start of a drag for ListBox1, and begin the drag operation.
begin
  if DragDetect(ListBox1.Handle, POINT(X,Y)) then
    DMTextSource1.Execute;
end;

procedure TForm1.ListBox2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
// Detect the start of a drag for ListBox2, and begin the drag operation.
begin
  if DragDetect(ListBox2.Handle, POINT(X,Y)) then
    DMTextSource2.Execute;
end;

procedure TForm1.DMTextTarget1Drop(Sender: TObject; Acceptor: TWinControl;
  const dropText: String; X, Y: integer);
// Handle drops on memo1.  Notify the user whether files or just non-file text
// has been dropped.

begin
  label24.caption := Format('%s: (%d,%d)', ['Drop', X, Y]);

  if ((Sender as TDMTextTarget).DroppedTextFormat = dtfFiles) then
    label24.caption := label24.caption + ' [files]'
  else
    label24.caption := label24.caption + ' [text]';

  // If this is a URL, the actual URL address is in dropText
  // (also in droppedLines[0] and Text).  The title is in URLTitle.
  if ((Sender as TDMTextTarget).DroppedTextFormat = dtfURL) then
    label24.caption := label24.caption
      + Format(' [URL, title=%s]', [(Sender as TDMTextTarget).URLTitle]);

  // The only REQUIRED part of this handler.  Do something with the text
  // that was just dropped.  If this is missing, the drop won't do anything.
  // You have to decide what you want to do with the text you get!
  memo1.lines.add(dropText);
  // The following line is equivalent to the preceding line.
  // memo1.lines.add((Sender as TDMTextTarget).Text);

end;

procedure TForm1.btnCloseClick(Sender: TObject);
// Close down
begin
  Application.Terminate;
end;

procedure TForm1.DMTextSource2BeforeDrop(Sender: TObject;
  Donor: TComponent; var dropText: String; var cancelDrop: Boolean);
// Called before the drop of text from DMTextSource2.  Allow the user to cancel.
// You can modify dropText here if you want.  This handler is shared by DMTextSource1
// and DMTextSource2.  The particular one in question is identified by looking at Sender.
var
  aString: string;
  showConfirmation: boolean;
begin
  showConfirmation := ((Sender = DMTextSource1) and (CheckBox1.Checked))
    or ((Sender = DMTextSource2) and (CheckBox2.Checked));
  if showConfirmation then
  begin
    if (deMove = (Sender as TDMTextSource).ReturnedEffect) then
      aString := 'Do you really want to move the string "%s"?'
    else
      aString := 'Do you really want to copy the string "%s"?';

    aString := Format(aString, [dropText]);

    cancelDrop := (MessageDlg(aString, mtConfirmation, [mbYes, mbNo], 0) = mrNo);
  end;
end;
```

```
 procedure TForm1.DMTextSource2AfterDrop(Sender: TObject; Donor: TComponent;
  droppedOK: Boolean);
var
  j: integer;
begin
  // A drop has been done from DMTextSource2.  If the drop was a move, we
  // have to remove the text from ListBox2.  You have to make sure that
  // droppedOK is true;  if you end the drag while the "no drag" cursor is
  // showing, this event will still fire, but with droppedOK = false.
  //
  // In this handler, the expressions (Sender as TDMTextSource).DonorComponent,
  // Donor, and ListBox2 all refer to the same thing.

  if droppedOK then
  begin
    if (deMove = (Sender as TDMTextSource).ReturnedEffect) then
    begin
      // It's a move.  Delete from the top down to avoid messing
      // up the Selected[] property.
      for j := ListBox2.items.count downTo 1 do    // Iterate
      begin
        if ListBox2.Selected[j-1] then
          ListBox2.items.delete(j-1);
      end;    // for
    end;

  end;
end;

procedure TForm1.DMTextTarget2Drop(Sender: TObject; Acceptor: TWinControl;
  const dropText: String; X, Y: integer);
// Handle drops on ListBox3
//var
//  j: integer;
///  aSL: TStringList;
begin
  // Bring the form to the top, so that the message dialog won't accidentally
  // be hidden.
  SetForegroundWindow(Handle);

  if CheckBox3.Checked and
    (MessageDlg('Do you want to clear the list before dropping?', mtConfirmation,
      [mbYes, mbNo], 0) = mrYes) then
    ListBox3.Items.Clear;

  // Show a sign that files were dropped (rather than text)
  Label7.visible := ((Sender as TDMTextTarget).DroppedTextFormat = dtfFiles);

  // Handle the drop.  Show the coordinates, just for fun.
  ListBox3.Items.Add(Format('     Drop at client pos (%d,%d):', [X, Y]));
  ListBox3.Items.AddStrings((Sender as TDMTextTarget).DroppedLines);

  // The following is equivalent to the preceding line, i.e., DroppedLines
  // and dropText contain the same information.
  //aSL := TStringList.create;
  //try
  //  // Get the dropped text into dropText
  //  aSL.text := dropText;
  //  for j := 1 to aSL.count do    // Iterate
  //  begin
  //    ListBox3.Items.Add(aSL[j-1]);
  //  end;    // for
  //finally
  //  aSL.free;
  //end;

end;
```

**For continuation
please download the project:
https://raize.com/**


Test Application for DropMaster

Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag demo 1 | Drag demo 2 | Drag demo 3 | Drag demo 4 | Drop demo A | Drop demo B

Single-selection, Copy

Apples
Bananas
Peaches
Pears
Nectarines
Tomatoes
Oranges
Plums

Drag a selection from this list box to any valid drop target, e.g., Word, Excel, or a TDMTextTarget-enabled window. Only Copy is supported by this text source.

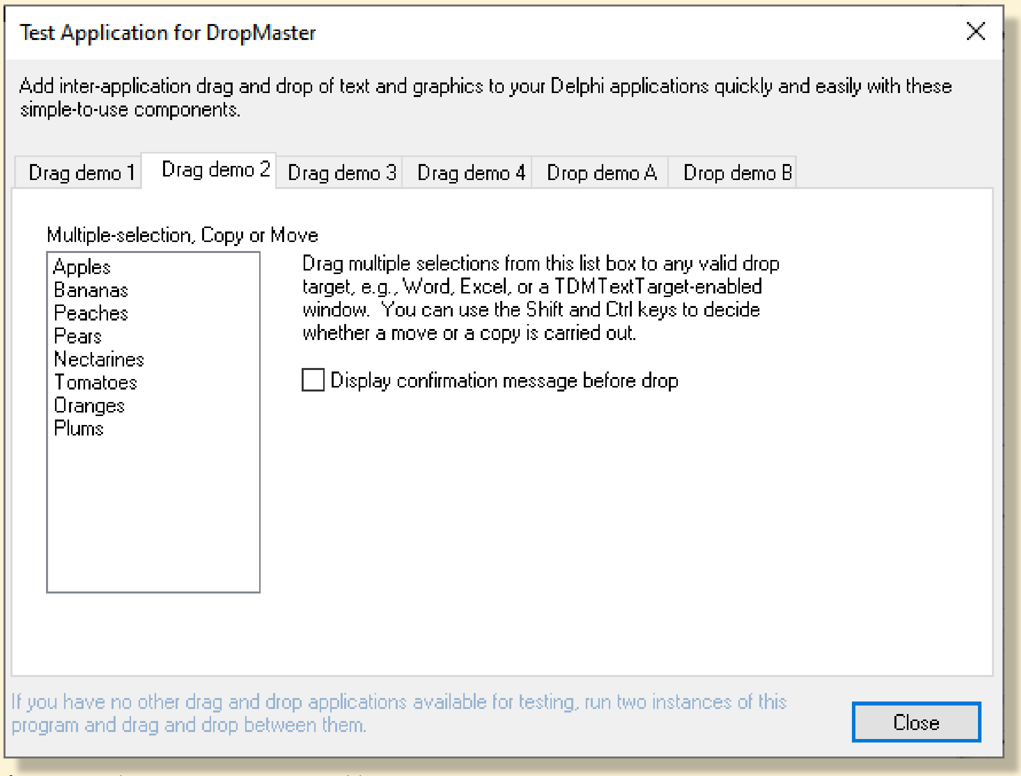
☐ Display confirmation message before drop

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

Close

figure 18: Tab 1

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.


Test Application for DropMaster

Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag demo 1 | Drag demo 2 | Drag demo 3 | Drag demo 4 | Drop demo A | Drop demo B

Multiple-selection, Copy or Move

Apples
Bananas
Peaches
Pears
Nectarines
Tomatoes
Oranges
Plums

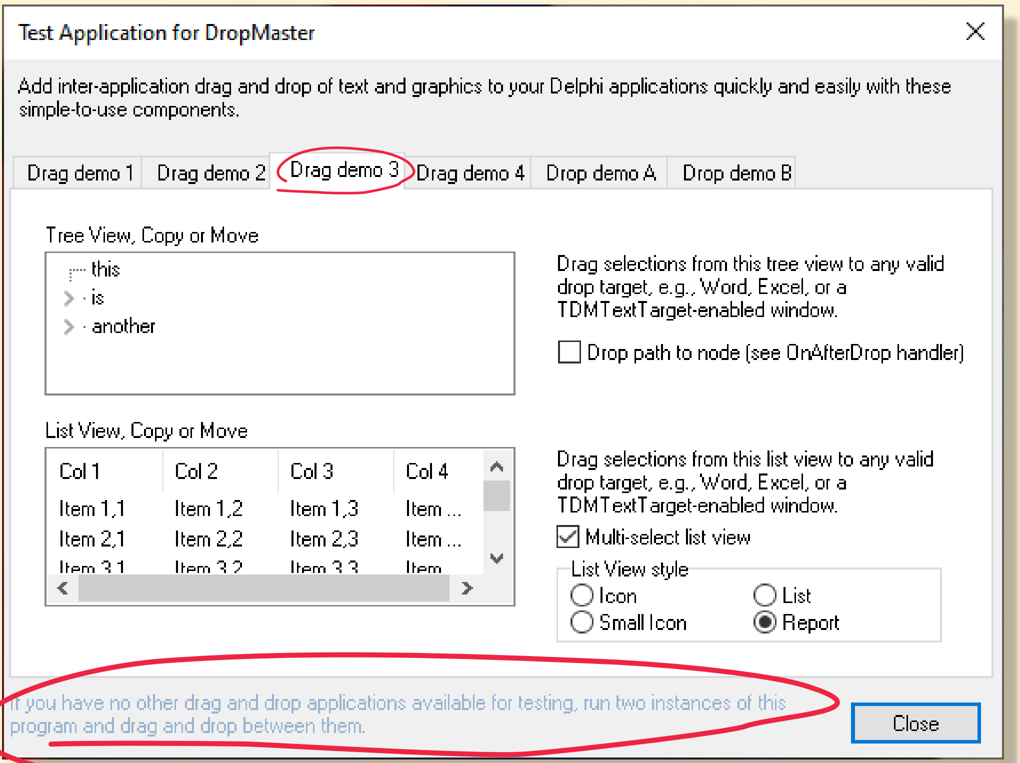
Drag multiple selections from this list box to any valid drop target, e.g., Word, Excel, or a TDMTextTarget-enabled window. You can use the Shift and Ctrl keys to decide whether a move or a copy is carried out.

☐ Display confirmation message before drop

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

Close

figure 19: Tab2 Interactions are possible



Test Application for DropMaster

Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag selections from this tree view to any valid drop target, e.g., Word, Excel, or a TDMTextTarget-enabled window.

☐ Drop path to node (see OnAfterDrop handler)

Col 1	Col 2	Col 3	Col 4
Item 1,1	Item 1,2	Item 1,3	Item ...
Item 2,1	Item 2,2	Item 2,3	Item ...
Item 3,1	Item 3,2	Item 3,3	Item

Drag selections from this list view to any valid drop target, e.g., Word, Excel, or a TDMTextTarget-enabled window.

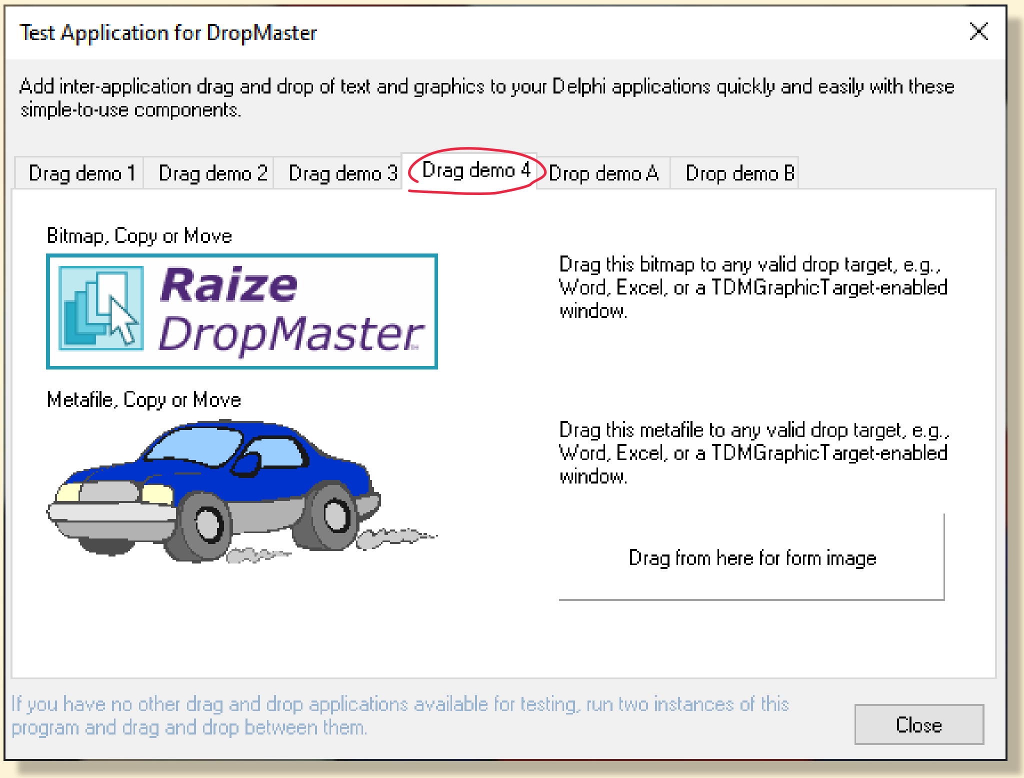
☑ Multi-select list view

List View style: ○ Icon ○ List ○ Small Icon ● Report

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

figure 20: Tab 3

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.



Test Application for DropMaster

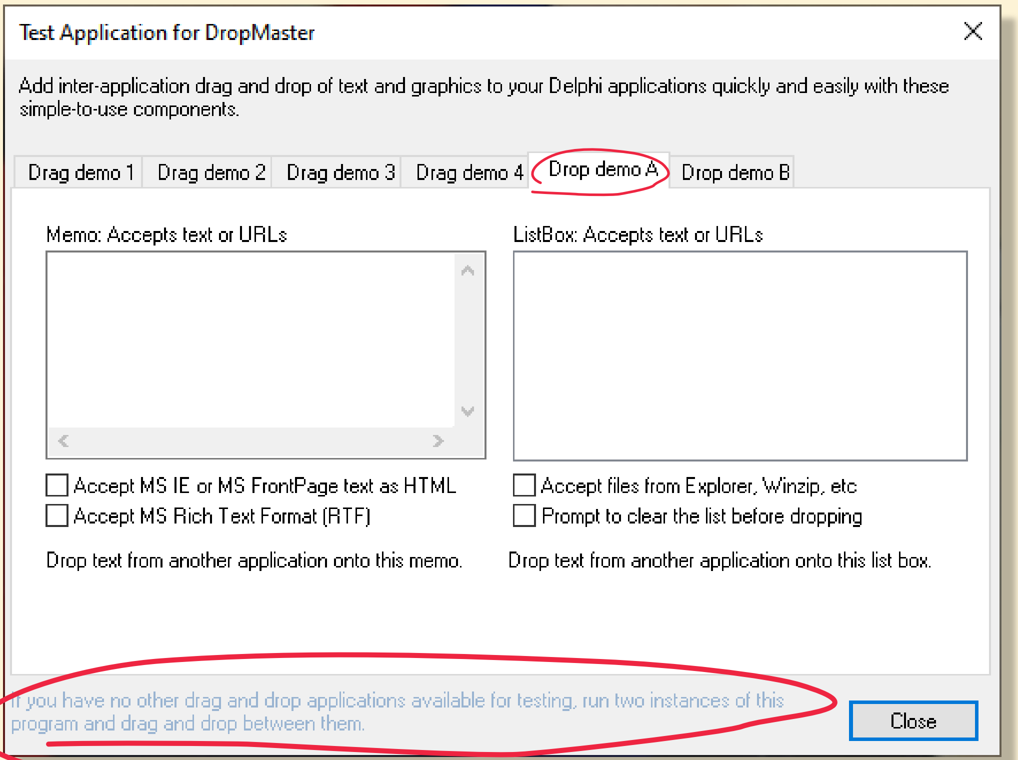
Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag this bitmap to any valid drop target, e.g., Word, Excel, or a TDMGraphicTarget-enabled window.

Drag this metafile to any valid drop target, e.g., Word, Excel, or a TDMGraphicTarget-enabled window.

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

figure 21: Tab 4


Test Application for DropMaster

Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag demo 1 | Drag demo 2 | Drag demo 3 | Drag demo 4 | Drop demo A | Drop demo B

Memo: Accepts text or URLs

- [ ] Accept MS IE or MS FrontPage text as HTML
- [ ] Accept MS Rich Text Format (RTF)

Drop text from another application onto this memo.

ListBox: Accepts text or URLs

- [ ] Accept files from Explorer, Winzip, etc
- [ ] Prompt to clear the list before dropping

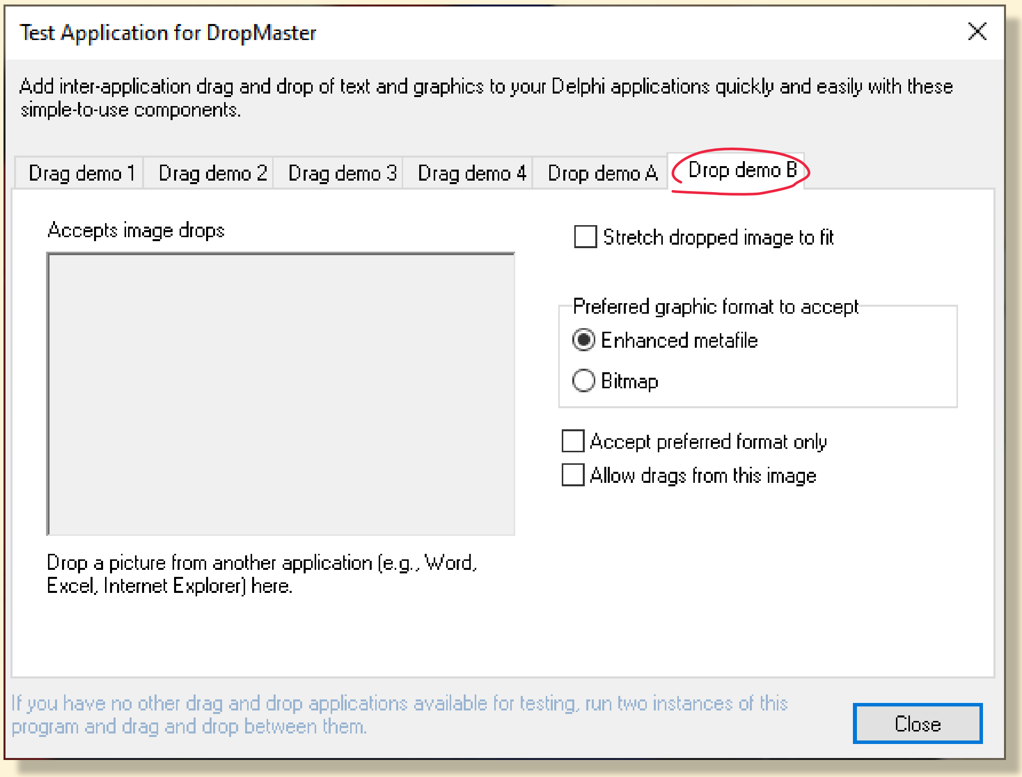
Drop text from another application onto this list box.

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

Close

figure 22: Tab A

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.


Test Application for DropMaster

Add inter-application drag and drop of text and graphics to your Delphi applications quickly and easily with these simple-to-use components.

Drag demo 1 | Drag demo 2 | Drag demo 3 | Drag demo 4 | Drop demo A | Drop demo B

Accepts image drops

Drop a picture from another application (e.g., Word, Excel, Internet Explorer) here.

- [ ] Stretch dropped image to fit

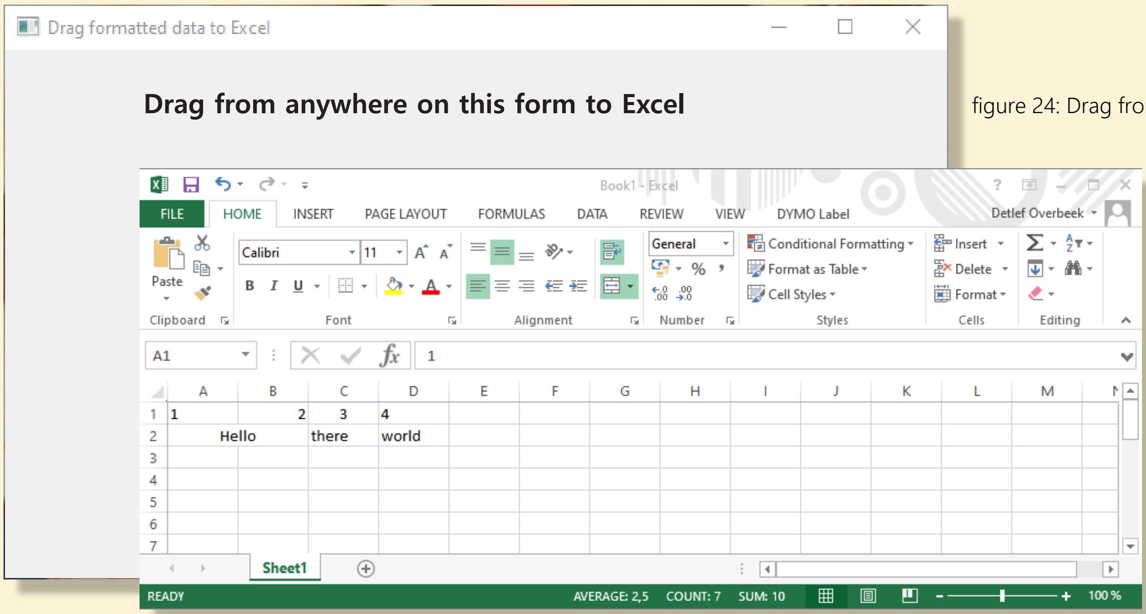
Preferred graphic format to accept
- ● Enhanced metafile
- ○ Bitmap

- [ ] Accept preferred format only
- [ ] Allow drags from this image

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

Close

figure 23: Tab B

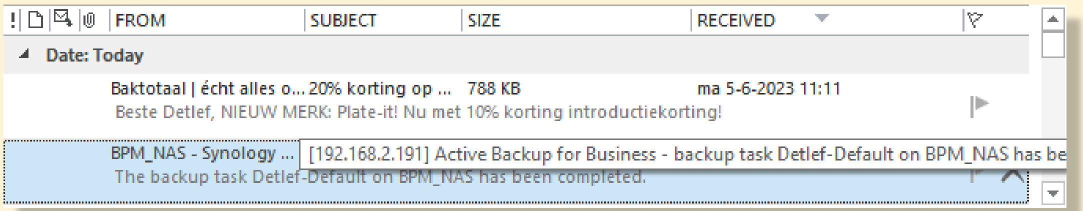


	A	B	C	D
1	1	2	3	4
2		Hello	there	world

AVERAGE: 2,5 COUNT: 7 SUM: 10

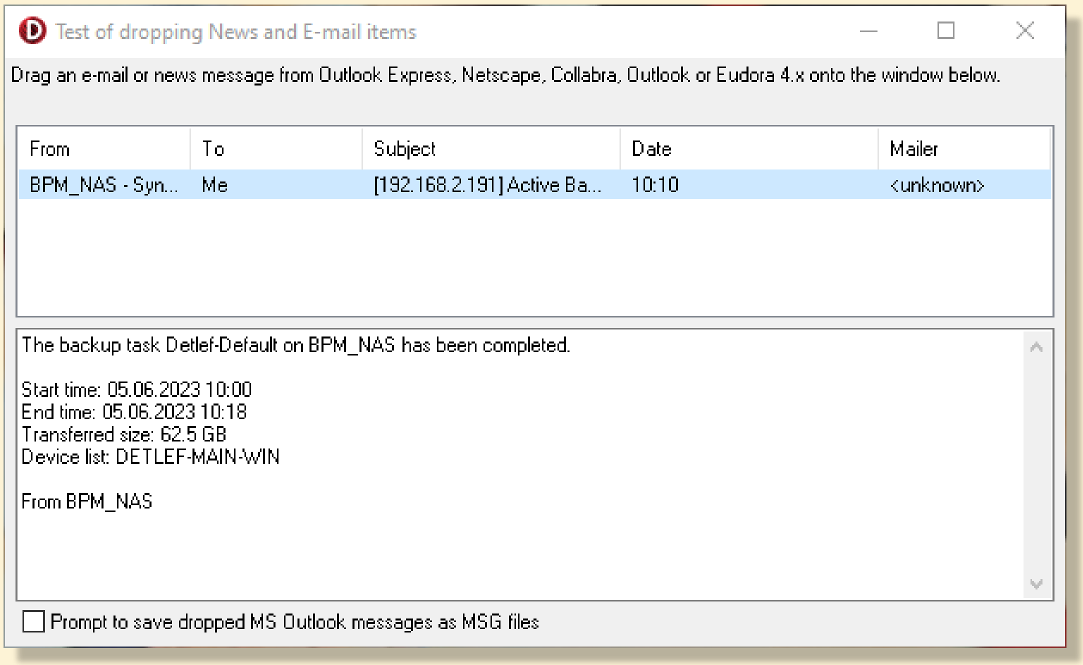
figure 24: Drag from anywhere

figure 25: Inserted into Te sheet



FROM	SUBJECT	SIZE	RECEIVED
Date: Today			
Baktotaal | écht alles o...	20% korting op ...	788 KB	ma 5-6-2023 11:11
Beste Detlef, NIEUW MERK: Plate-it! Nu met 10% korting introductiekorting!			
BPM_NAS - Synology ...	[192.168.2.191] Active Backup for Business - backup task Detlef-Default on BPM_NAS has be		
The backup task Detlef-Default on BPM_NAS has been completed.			

figure 26: Copied from an Email



Drag an e-mail or news message from Outlook Express, Netscape, Collabra, Outlook or Eudora 4.x onto the window below.

From	To	Subject	Date	Mailer
BPM_NAS - Syn...	Me	[192.168.2.191] Active Ba...	10:10	<unknown>

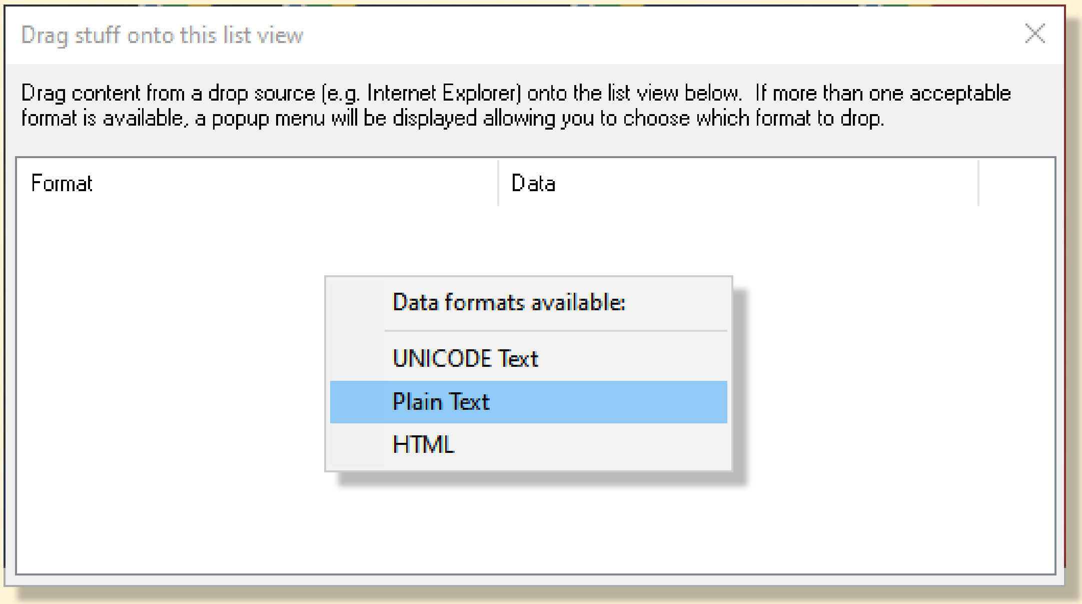
The backup task Detlef-Default on BPM_NAS has been completed.

Start time: 05.06.2023 10:00
End time: 05.06.2023 10:18
Transferred size: 62.5 GB
Device list: DETLEF-MAIN-WIN

From BPM_NAS

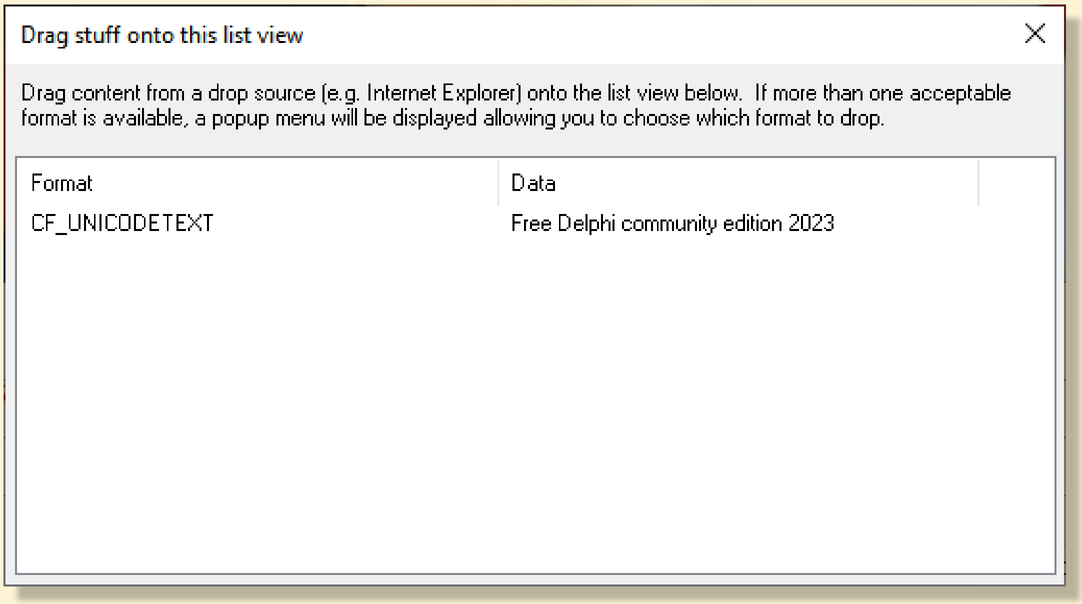
☐ Prompt to save dropped MS Outlook messages as MSG files

figure 27: Into text fields


Drag stuff onto this list view

Drag content from a drop source (e.g. Internet Explorer) onto the list view below. If more than one acceptable format is available, a popup menu will be displayed allowing you to choose which format to drop.

figure 27: Into text fields


Drag stuff onto this list view

Drag content from a drop source (e.g. Internet Explorer) onto the list view below. If more than one acceptable format is available, a popup menu will be displayed allowing you to choose which format to drop.

Format	Data
CF_UNICODETEXT	Free Delphi community edition 2023

CONCLUSION:

All in all: this is a very interesting component group which provides especially for the starter a lot of ease to do things he probably would never had dared to.
Ray is always very original and creative in making things easy and better.

ONLY AT BARNSTEN ENDING 30 JUNE

Quotation Request Form - Barnsten.com

https://www.barnsten.com

LAZARUS TO THE AID OF VISUAL STUDIO CODE

By Michael van Canneyt

Starter Expert

ABSTRACT
Lazarus has excellent code tools. VS Code has a framework for adding support for new languages. In this article we show how first-class Pascal support can be implemented in **Visual Studio** code `using` the codetools of the Lazarus IDE.

1 INTRODUCTION

It is no secret that the code tools of the **Lazarus IDE** are excellent, and even surpass the ones in the Delphi IDE. So for coding in Object Pascal, the Lazarus IDE is an excellent choice.

But sometimes you need to code more than just Pascal.
You may wish to edit **Markdown, HTML, CSS, C** or create Makefiles or shell scripts. This can also be done in the Lazarus editor, but then the support the editor offers you on top of basic editing is very limited: in the best case, you have syntax highlighting.
If you want more than that, you need to open another editor to do the editing.

You may wish to edit Markdown, HTML, CSS, C or create Make files or shell scripts.

Many modern editors offer support for many languages: not only syntax highlighting, but also more advanced features one expects in an editor:
code completion, identifier completion (Intellisense) finding references to a symbol, refactorings such as renaming a symbol and so on. One such editor is Visual Studio Code (*an evolution of the now defunct Atom editor*):
`https://code.visualstudio.com/`

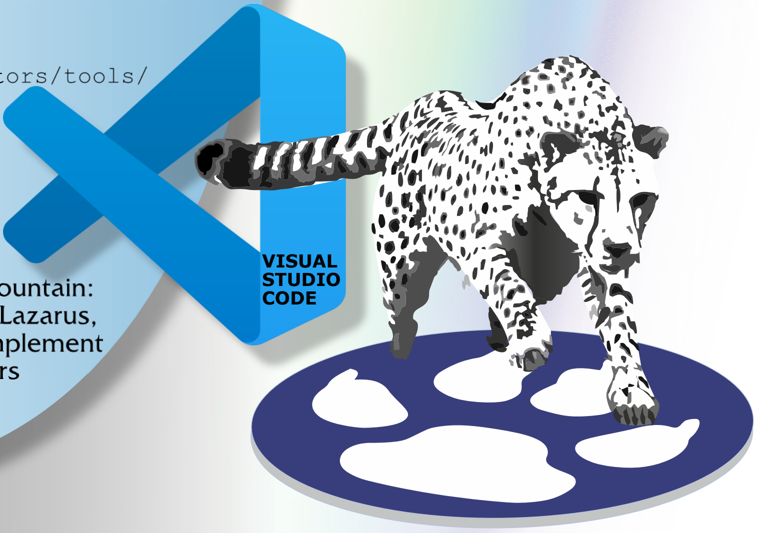
It has become very popular, and has a staggering amount of extensions - including several for Pascal.
The Visual Studio Code editor is managed by Microsoft, and Microsoft has introduced a standard for extending its editor with support for new Languages:
The **Language Server Protocol**
`https://microsoft.github.io/language-server-protocol/`

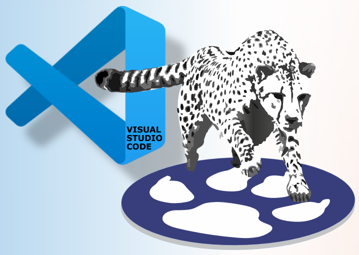
This standard has been adopted by several other editors, including **Emacs, Vim, Delphi, Sublime Text, IntelliJ** and the **KDE editor suite (Kate & KDevelop).**

A more complete list is available here:
`https://microsoft.github.io/language-server-protocol/implementors/tools/`

Unfortunately, the Lazarus IDE is not in this list, as it does not yet support the **LSP protocol**:
it would enable to use any language in the Lazarus IDE.

If the mountain will not come to Mohamed, Mohamed must go to the mountain: Pending support for the LSP protocol in Lazarus, the lazarus code tools can be used to implement the LSP protocol and extend other editors first-class Pascal support.





Several **LSP** implementations using the Lazarus codetools are available on **Github**,but in this article we'll concentrate on one:
`https://github.com/genericptr/pascal-language-server`

2 THE LSP PROTOCOL

The Language Server Protocol is based on a JSON-RPC communication mechanism.
The editor starts a program that acts as a Language Server, and sends JSON-RPC
messages to the process over standard input. It reads the results and possible
commands from the LSP server from standard output.
This exchange looks as follows. The editor sends a request:

```
{
  "jsonrpc" : "2.0",
  "method" : "textDocument/didOpen",
  "params" : {
     "textDocument" : {
       "uri" : "file:///home/michael/source/testio/testio.lpr",
       "languageId" : "pascal",
       "version" : 1,
       "text" : "program testio;\n\n ... end.\n"
     }
  }
}
```

In this case the server replies with a command:

```
{
  "jsonrpc" : "2.0",
  "method" : "textDocument/publishDiagnostics",
  "params" : {
     "diagnostics" : [],
     "uri" : "file:///home/michael/source/testio/testio.lpr"
  }
}
```

The full list of commands a server can implement is documented in the LSP protocol (*see the URL above*). When the editor starts the server, a handshake is performed:

the `initialize` command.

In the `initialize` command *(the first command the editor sends to the server)*, the client *(the editor)* indicates the capabilities it has, and the language server replies with the capabilities it has:
this is normally a list of '**Providers**' of certain functionalities.

This handshake is important, because not all servers support all commands, and not all clients support all commands.

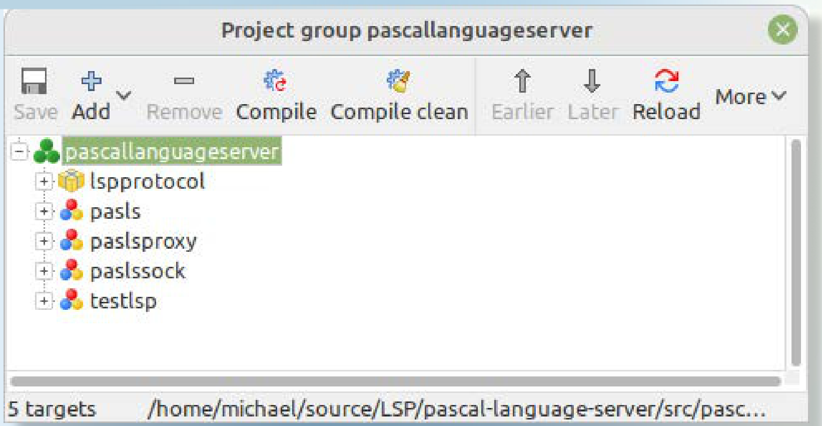
A server can also specify custom commands: these are commands that are not part of the LSP Protocol, but for which the LSP protocol has a special command in place, aptly named '**executeCommand**'.
The Pascal Language Server mentioned above implements various of the functionalities expected by the language server protocol:

textDocument/declaration	Goto declaration
textDocument/implementation	Goto implementation
textDocument/references	find references to a symbol
textDocument/signatureHelp	Show function or method signature (parameters).
textDocument/documentSymbol	List of symbols in a project.
textDocument/documentHighlight	Similar to textDocument/references find references to a symbol.
textDocument/completion	Identifer completion
textDocument/hover	Smart hints about your code
window/showMessage	allow the LSP server to show a message in the editor.
workspace/symbol	List all symbols matching a piece of text.
workspace/executeCommand	Execute a custom command.
diagnostics	Allows the server to send a list of diagnostics to the client: these can be warnings, errors etc. They are displayed in the editor (*under 'Problems' in VS Code*)

Additionally, the Pascal Language Server implements some custom commands:

pasls.completeCode	Code completion: will complete the current class, define a variable etc. The equivalent of code completion in the IDE.
pasls.formatCode	calls the Jedi code formatter on the pascal file.
pasls.invertAssignment	a refactoring which inverts the assignment statements in the selection.
pasls.removeEmptyMethods	a refactoring which removes all empty methods from the current file.

More custom methods are being added to the server: theoretically, all code tools offered by Lazarus can be implemented.
What does the language server not do ? It does not compile the pascal code, it also does not offer functionality to edit form files. There are also several commands in the LSP that it does not implement. It also does not do syntax highlighting. *(The 'pascal magic' extension in the VS extension marketplace does this for you).*



```
pascallanguageserver
  lspprotocol
  pasls
  paslsproxy
  paslssock
  testlsp
5 targets    /home/michael/source/LSP/pascal-language-server/src/pasc...
```

Figure 1: The language server project group

3 USING THE LSP SERVER

To use the LSP server in VS Code, 2 things are needed:

❶ Compile the LSP server.
❷ Install an extension in VS Code that registers the LSP server, and the extra commands made available by it.

To compile the LSP server, you can clone the official repository from the URL above.

When you do so, below the Src directory, you'll have a project group file `pascallanguageserver.lpg` with 4 projects and a lazarus package (*see figure 1 on page 3 of this article*):

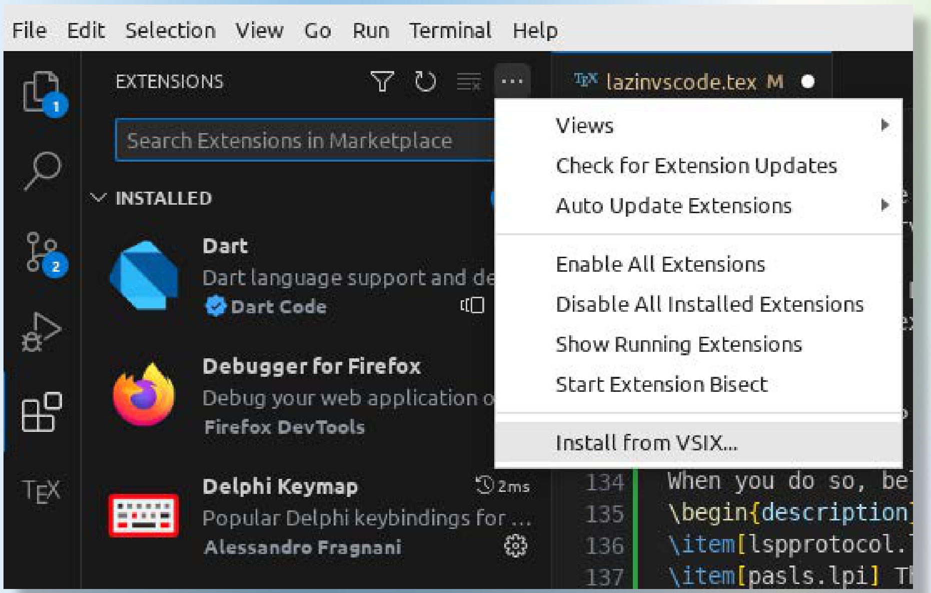
`lspprotocol.lpk`
A package with the units that make up the language server protocol.
It depends on the Lazarus codetools package.

`pasls.lpi`
The actual LSP server program. This is the program that you must compile and use. In order to compile it, you must first open and compile the `lspprotocol` package.

`paslsproxy.lpi`
A proxy program that implements the JSON-RPC protocol on standard input/output and forwards the messages on a TCP/IP socket using a special high-speed message scheme.

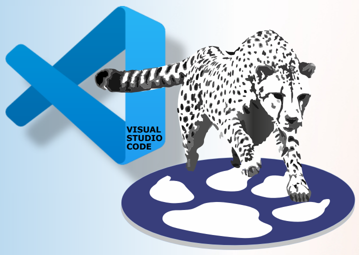
`paslssock.lpi`
A language server program that listens on a TCP/IP socket and implements the JSON-RPC protocol using the same message scheme as paslsproxy.
`testlsp.lpi` A minimal unit test program.

The `paslsproxy` and `paslssock` programs are only used for debugging the language process server: because the editor starts the LSP process, it is difficult to debug startup and message flow. By running `paslssock` in the Lazarus debugger and letting the editor start the `paslsproxy` program, you can debug the language server process. But for regular use, you only need the `pasls` program, and this is the one you should compile. It will compile on all platforms that Lazarus supports.



```
When you do so, be
\begin{description
\item[lspprotocol.
\item[pasls.lpi] Th
```

Figure 2: The VSIX install menu in VS Code



You can compile it with the released version of Free Pascal (3.2.2) or with the development version from git. If you do the latter, you'll have better syntax checking due to improvements in the PAS2JS parser (*which is used to do syntax checking*).

To create the pasls binary, open the project in the lazarus IDE and hit the compile key combination or use the Run-Compile menu item in lazarus. You can also try to compile the pasls binary without Lazarus installed. You can checkout the lazarus sources from the git repository at

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

If you do this, you have to specify the paths to the lazarus codetools package and all other packages on which the latter depends:

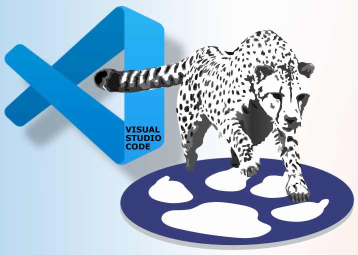
- **codetools** (components/codetools)
- **jcfbase** (components/jcf2) - the jedi code formatter.
- **lazutils** (components/lazutils)

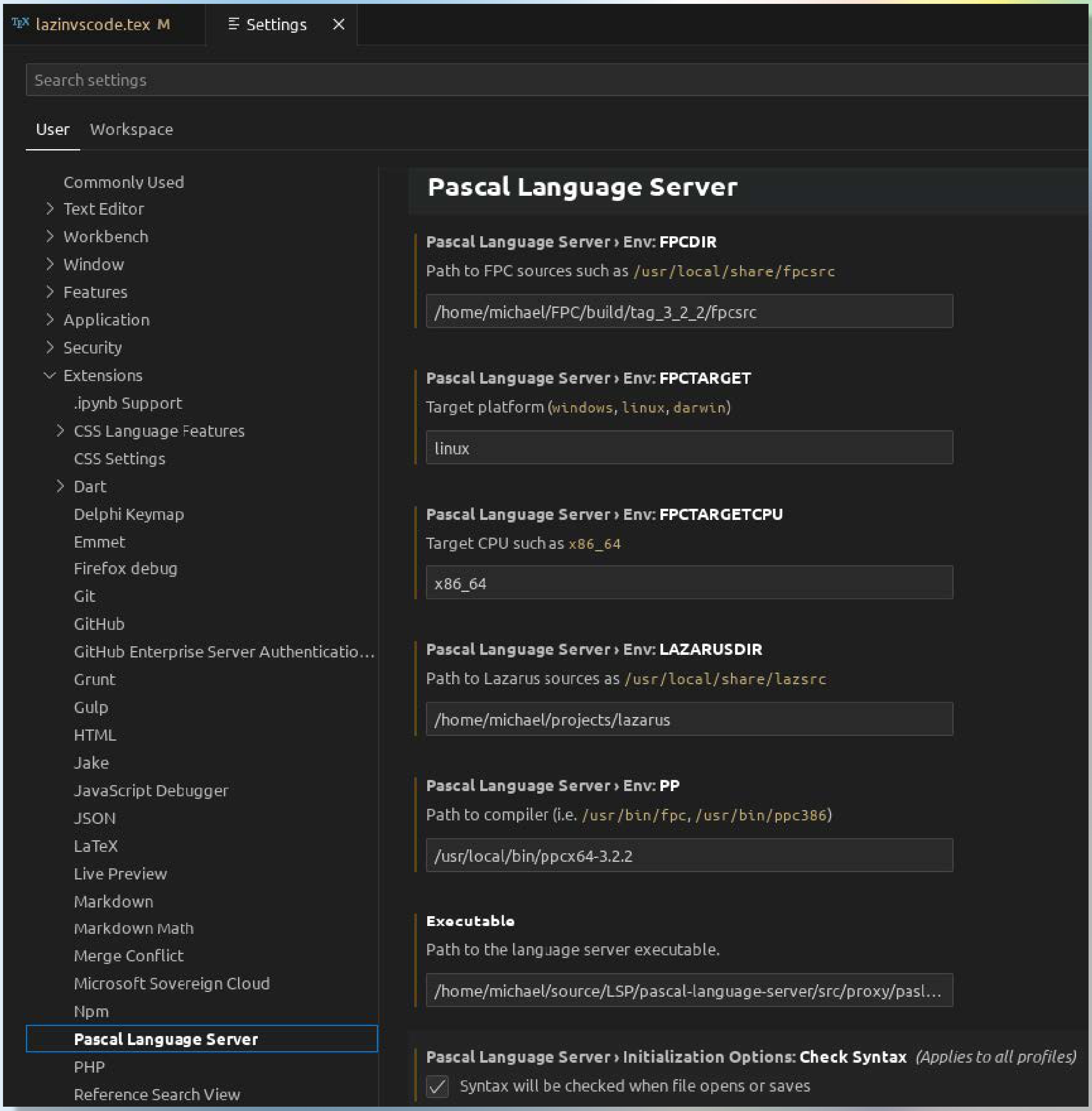
Once you have a pasls binary, it can be used in an editor. For VS Code, there is an extension package available on github:
`https://github.com/genericptr/pasls-vscode`

You can package and install the extension in VS Code yourself, but a `.vsix` package file is available.

This is an extension package for **Visual Studio Code**, which can be installed using a menu item: in the extensions tab on the left of the IDE, the menu at the top contains an item **'Install from VSIX'** (*see figure 2 on page 4 of this article*).

You can use that to select and install the `.vsix` file. Once installed, there are settings in the **VS Code IDE** that must be configured (*see figure 3 on page 5*). Important settings that are needed to make the codetools work correctly are the following: (*Proceed to Page 7 of this article*)





Pascal Language Server

Pascal Language Server › Env: FPCDIR
Path to FPC sources such as `/usr/local/share/fpcsrc`
/home/michael/FPC/build/tag_3_2_2/fpcsrc

Pascal Language Server › Env: FPCTARGET
Target platform (`windows`, `linux`, `darwin`)
linux

Pascal Language Server › Env: FPCTARGETCPU
Target CPU such as `x86_64`
x86_64

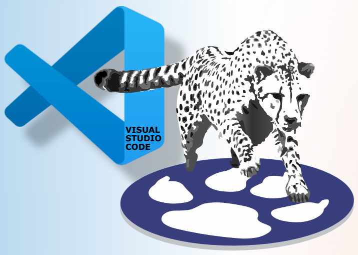
Pascal Language Server › Env: LAZARUSDIR
Path to Lazarus sources as `/usr/local/share/lazsrc`
/home/michael/projects/lazarus

Pascal Language Server › Env: PP
Path to compiler (i.e. `/usr/bin/fpc`, `/usr/bin/ppc386`)
/usr/local/bin/ppcx64-3.2.2

Executable
Path to the language server executable.
/home/michael/source/LSP/pascal-language-server/src/proxy/pasl...

Pascal Language Server › Initialization Options: Check Syntax *(Applies to all profiles)*
✓ Syntax will be checked when file opens or saves

Figure 3: The Pascal Language server settings



Env:FPCDIR
The directory where the FreePascal sources are located.

Env:FPCTARGET
The target operating system.

Env:FPCTARGETCPU
The target CPU.

Env:LAZARUSDIR
The directory where the Lazarus sources are located
(*only needed if you work on programs that use some lazarus packages*)

Env:PP
The path to the Free Pascal compiler binary: the codetools use this to get some compiler information.

Executable

The path to the pasls binary that you compiled.

format Config
The path to the configuration file for the code formatter. This file can be copied from the settings file of a Lazarus installation *(the file is called* `jcfsettings.cfg`*).*
Furthermore, there are options that control the behavior of the language server itself, they are part of the initialization options:

Check syntax
When checked, the language server checks the syntax of the current file whenever you save it.

Document symbols
When checked, querying for document symbols is possible.

FPC options
these are options that you would normally specify when compiling your project: the codetools analyse these options to determine defines etc.

Include work space folders as include paths
When checked, all sub directories of the VS Code work space directory will be used as include paths (*the* `-Fu` *command-line option of the compiler*).

Include work space folders as unit paths
When checked, all sub directories of the VS Code work space directory will be used as include paths (*the* `-Fu` *command-line option of the compiler*).

Insert completion procedure brackets
when checked and you complete a procedure call, the procedure call will have () brackets appended (*even if no parameters are expected*).

Insert completions as snippets
when checked and you complete a procedure call, the procedure call is inserted as a snippet:
it will have a cursor placeholder for the parameter (*i.e.* `'($0)'`).

Maximum completions
the maximum amount of possible completions to be shown.

Minimalistic completions
Provide minimal completion information : only the symbol name is shown, not what kind of symbol it is.

Overload policy
determines how overloads are handled in the symbol list.

A numerical value with the following meanings:

❶ - Duplicate function names appear in the list

❷ - Ignore overloads, only the first is used.

❸ - Add a suffix which denotes the overload count

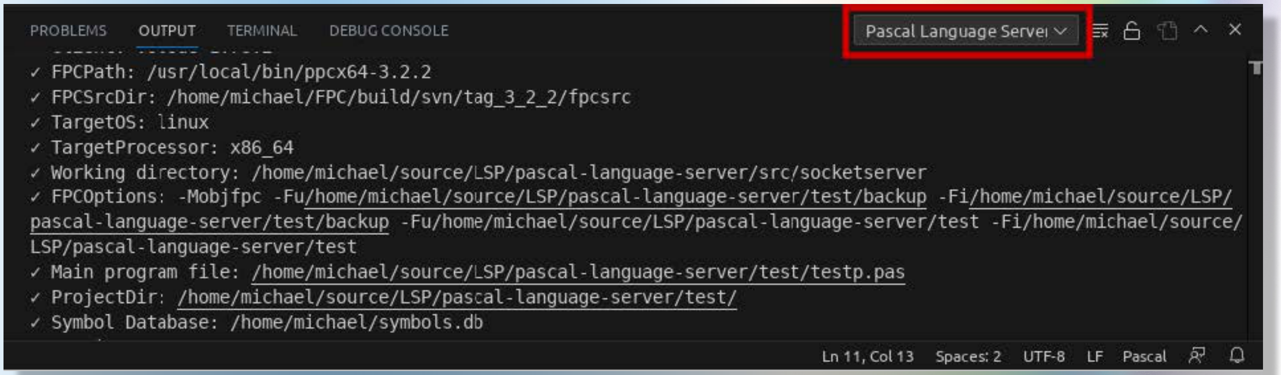
Program The main program file:
This is used by the codetools to determine what units are part of the project, and to find references.

Publish diagnostics
When the codetools return an error in the language server, this error is reported as diagnostics.

Show syntax errors
In case of syntax errors during a syntax error check, they are shown as small windows in the editor.

Symbol database
a sqlite database to use for symbols: this database will be created and filled with symbols. This is then used as a cache: instead of parsing the files, the contents of the cache is shown instead.

Once you're done with the settings, you're all good to go. When you open a pascal project in VS Code, you can see that the pascal language server is correctly started in the output window (*Figure 4 on page 8 of this article*), when you select the 'Pascal Language server' tool (*see image, the red rectangle at the top right*).
If the pascal language server was initialized correctly, you can start enjoying enhanced coding editing such as in Figure 5 on page 9, in VS Code.



```
✓ FPCPath: /usr/local/bin/ppcx64-3.2.2
✓ FPCSrcDir: /home/michael/FPC/build/svn/tag_3_2_2/fpcsrc
✓ TargetOS: linux
✓ TargetProcessor: x86_64
✓ Working directory: /home/michael/source/LSP/pascal-language-server/src/socketserver
✓ FPCOptions: -Mobjfpc -Fu/home/michael/source/LSP/pascal-language-server/test/backup -Fi/home/michael/source/LSP/
pascal-language-server/test/backup -Fu/home/michael/source/LSP/pascal-language-server/test -Fi/home/michael/source/
LSP/pascal-language-server/test
✓ Main program file: /home/michael/source/LSP/pascal-language-server/test/testp.pas
✓ ProjectDir: /home/michael/source/LSP/pascal-language-server/test/
✓ Symbol Database: /home/michael/symbols.db
```

Figure 4: The Pascal Language server output on startup

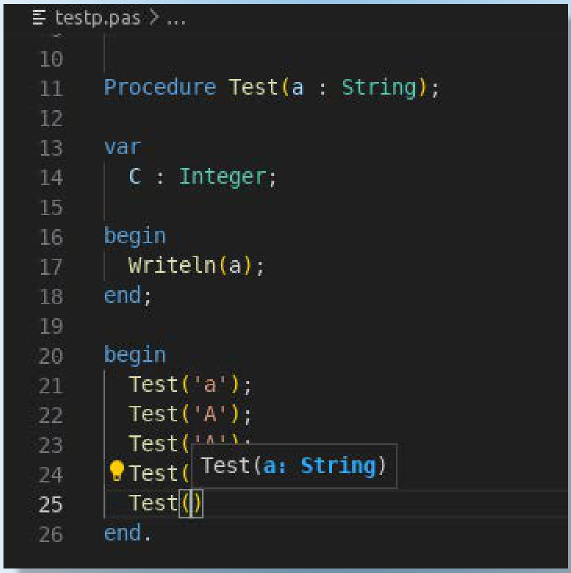
4 EXECUTING CUSTOM COMMANDS

To execute the code formatter, you invoke the usual code formatting request in VS Code: the extension redirects this request to the language server. The standard key combination for this is **ctrl-shift-i**.

The code formatter uses a configuration file, you can set the location of the configuration file in the settings. The configuration file is an XML file.

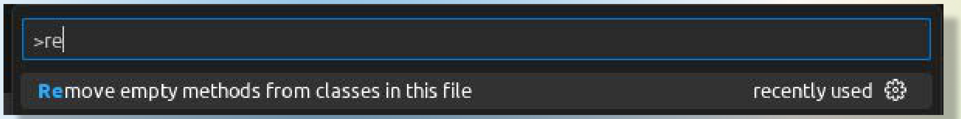
A sample file has been included in the github repository of the pascal language server. Most settings are self-explanatory, so editing the XML is possible, but at this time, the configuration file is most easily edited in the Lazarus '**Tools - Options**' dialog.

The code completion feature of Lazarus is mapped to the standard key combination of Lazarus : **ctrl-shift-c**. But you can simply type '**code completion**' in the command palette and activate it like that.
To execute the other commands (*remove empty methods or code formatter*), you invoke the command palette (**ctrl-shift-;**) and type the description of the command.
For instance '**remove**' will result in a list as shown in *figure 6 on page 9 of this article* in a later version of the language server, these will be added to the refactoring menu.



```
testp.pas > ...

Procedure Test(a : String);

var
  C : Integer;

begin
  Writeln(a);
end;

begin
  Test('a');
  Test('A');
  Test('A').
  Test( Test(a: String)
  Test()
end.
```

Figure 5: A helpful hint about the function you're about to call.



```
>re
Remove empty methods from classes in this file          recently used
```

Figure 6: Invoking the 'remove empty methods' command.

5 CONCLUSION

The Pascal Language server allows you to enjoy part of the tools that the Lazarus IDE offers you right in VS Code. The Pascal language server is not limited to use in VS Code. It can be used with all editors that support the LSP process: one of the authors uses Sublime Text to develop Pascal. The pascal language server is under active development, so more goodies from the Lazarus code tools will be made available in the near future.

Delphi Alexandria Community version for Delphi 11

Suited for individual developers or early-stage startups with limited revenue

Features Full-Featured Free Delphi IDE for Creating Native Cross-Platform Apps

`https://www.embarcadero.com/products/delphi/starter/free-download`

Delphi Community Edition (CE) is a full featured IDE for building iOS, Android, Windows and macOS apps from a single Delphi codebase (limited commercial use license).
Delphi CE is shared free of charge with our community of freelance developers, startups, students and non-profits.
Delphi CE includes a code editor, powerful debugging tools, built-in access to popular local databases with live data at design time, Bluetooth capabilities, and a visual UI designer with support for pixel perfect, platform-specific styling.

EXPLANATION:
Can you get the edition without problems?
If you're an individual, you may use Delphi CE to create apps for your own use and apps that you can sell until your revenue reaches US$5,000 per year.

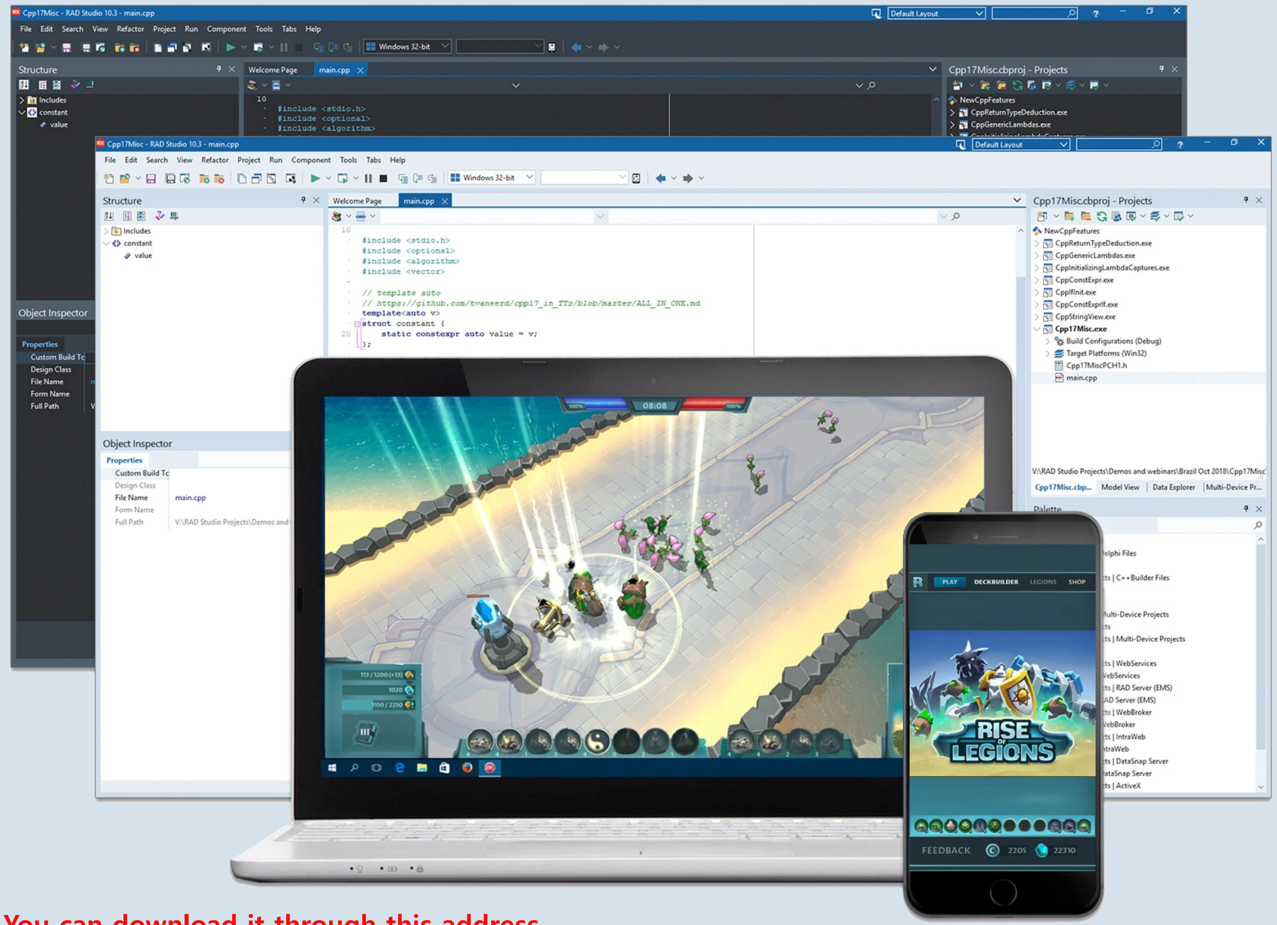
If you're a small company or organization with up to US$5,000 per year in revenue, you can also use the Delphi CE. Once your company's total revenue reaches US$5,000, or your team expands to more than five developers, you can move up to an unrestricted commercial license with Professional edition.

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

See the Community Edition FAQs for additional details.

Delphi is available in Community, Academic, Professional, Enterprise, and Architect editions.
For details on the differences between the editions, see the Product Editions page and Feature Matrix.

Move up to the Professional edition or above to get additional features including components and drivers for database connectivity, a full commercial development license, and much more.


Cpp17Misc - RAD Studio 10.3 - main.cpp
File Edit Search View Refactor Project Run Component Tools Tabs Help
Windows 32-bit
Structure
Welcome Page
main.cpp
Cpp17Misc.cbproj - Projects
Object Inspector
RISE OF LEGIONS

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

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

You can download it through this address
`https://www.embarcadero.com/products/delphi/starter/free-download`

Delphi Alexandria Community version for Delphi 11

Suited for individual developers or early-stage startups with limited revenue

Features

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

Build native Windows Applications with High-Performance UI Framework and Components (VCL) Visual Component Library (VCL) is a visual component-based object-oriented framework for developing UIs for Windows applications. It delivers a number of visual and non-visual components to achieve optimal performance and native platform user experience on the Windows OS.	✓
Build Mobile First, Cross-Platform Apps with Native Experience UI Framework (FMX) and Components (iOS, Android, macOS, Windows) FireMonkey (FMX) is a visual component framework that uses smart styles and platform services to design the UI once and adapt it to each platform, so you can to target multiple platforms, including both application logic and UI, with the same code.	✓
Run-Time Library source code Includes source code for the VCL, FMX and most other libraries, to learn from or extend with your own code	**Limited Use**
Full Commercial use license Consult the EULA for the full license terms for each edition.	**Limited Commercial Use**
Connect to local databases and build data-aware applications with support for multiple data sources with FireDAC FireDAC local/embedded connectivity to certain local databases, including Microsoft Access database, SQLite database, InterBase ToGo / IBLite, InterBase on localhost, MySQL Embedded, MySQL Server on localhost, Advantage Database local engine, PostgreSQL on localhost, Firebird Embedded, and Firebird on localhost.	✓
InterBase Embedded Database InterBase is an award winning, high-performance SQL Database with multiple advanced features, including enterprise security, change views, alerts, generators, and more. There are 2 embedded versions, IBLite and IBToGo, which adds encryption support and extra features.	✓ **IBLite Mobile Deployment**

Upgrading and Maintaining Delphi Legacy Projects
Copy link
Delphi Legacy Projects
Strategies and Survival Guide
William Meyer
D11
Upgradin
&
Maintainin
https://youtu.be/GYUyWaMw3ws
YouTube

Below is an online youtube video address
that helps with the installation of Delphi Win 11 CE:
:
`https://www.youtube.com/watch?v=kjP680wlj-M`

Jim McKeeth has left Embarcadero

Jim McKeeth left and his successor Ian Barker Right

If you have watched any of **Embarcadero's** online content, attended a **RAD Studio** webinar, or been to one of the in-person events you will most likely know the wonderful **Jim McKeeth**.
Jim has been the **Chief Developer Advocate** and **Engineer** for Embarcadero since July 13th, 2013, just short of ten years.
Today, however, the big news is that Jim has left Embarcadero and is moving on to a new role as a developer advocate at **EOS Network Foundation.**

Of course, we're devastated that Jim's particular brand of jovial code geekery will no longer be at the helm of the Developer Relations program, but we're also thrilled for him to be moving on to new horizons and will get to stretch that burgeoning tech brain of his with the delights of such things as block-chain.

Along with that news comes, of course, an announcement that **Ian Barker** has taken over as Embarcadero Developer Advocate. He'll be dealing with most of the things Jim did, those that are public, along with those which he did so capably behind the scenes too, of which there are many.
Eli Mapstead will be expanding his role too and taking over some of the Python projects that Jim oversaw and championed. Yes, Jim can legitimately make the claim "it took two people to replace me".

Ian Barker is going to do a more comprehensive retrospective blog post of some of the many great things Jim has been responsible for as well as highlight a few of his crazy professor moments.

Jim was loved by many people because of his humour and energy.

ADVERTISEMENT

Update for FastReport

Creating a single ecosystem of report generators for Delphi in the FastReport 2023.2 release:

One installation system with online authorization — install and update all your products at once.

Common release system — major releases are published as a general.

Shared library for all products — fixes and new functionality can be available in multiple products at once.

Convenient to work with all products of the Delphi family already from the current release!

Also:

Improved compatibility with the current version of the IDE, HiDPI support for FastCube, and improved work with styles.

Upgrade to version 2023.2 or Download a free demo here:
www.fast-report.com

Starter Expert

Installing FastReport in Lazarus for Linux

There is a new **FastReport** Edition for Lazarus-Linux
in a **Trial** version as well **Professional**.

The **Trial** has a page limit with a note in the corner that the edition is a Trial.
This version misses the Rich View,
but the Professional edition has it and client/server components.
This article is about how to install **FastReport** in Lazarus for Linux

Each item in this list consists of 4 files
(3 installer packages and a text file).

- Lazarus(project) installation package;
- `fpc-src` installation package;
- `fpc(laz)` installation package;
- `README.txt`.

It is important to install them in the right order.
First `fpc(laz)`, then `fpc-src`
and finally Lazarus(project).

Let's fix the **problem with fonts** in advance.
All operating systems have default fonts.
For example, the Arial font is the default font
in both Windows and Ubuntu.
But in fact, default Arial n **Ubuntu** is not the
same as default Arial in Windows, so text reports created in
Windows Lazarus will look terrible in **Linux** Lazarus (*and vice versa*).To avoid this, we will immediately install fonts in Linux
as in Windows.

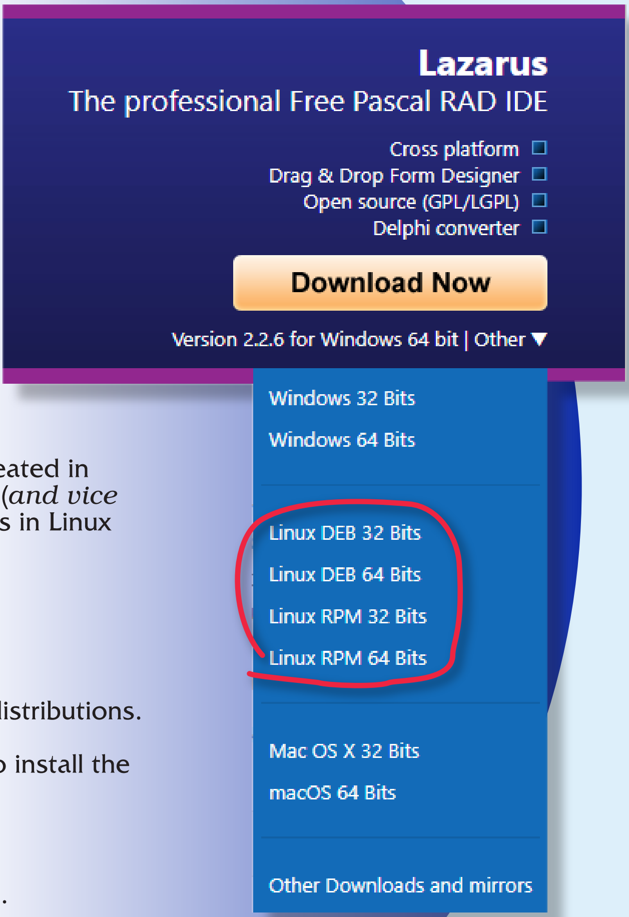
For Ubuntu, you can use the following command:

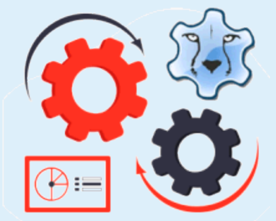
```
sudo apt-get install msttcorefonts
```

But the command may be different for other Linux distributions.

Further, for proper functioning of SqLite, you need to install the following packages: `sqlite3, libsqlite3-dev.`
For more details go to:
`https://wiki.freepascal.org/SQLite`

We launch Lazarus, it will prompt you to configure it.
Click "OK" to accept the default settings.

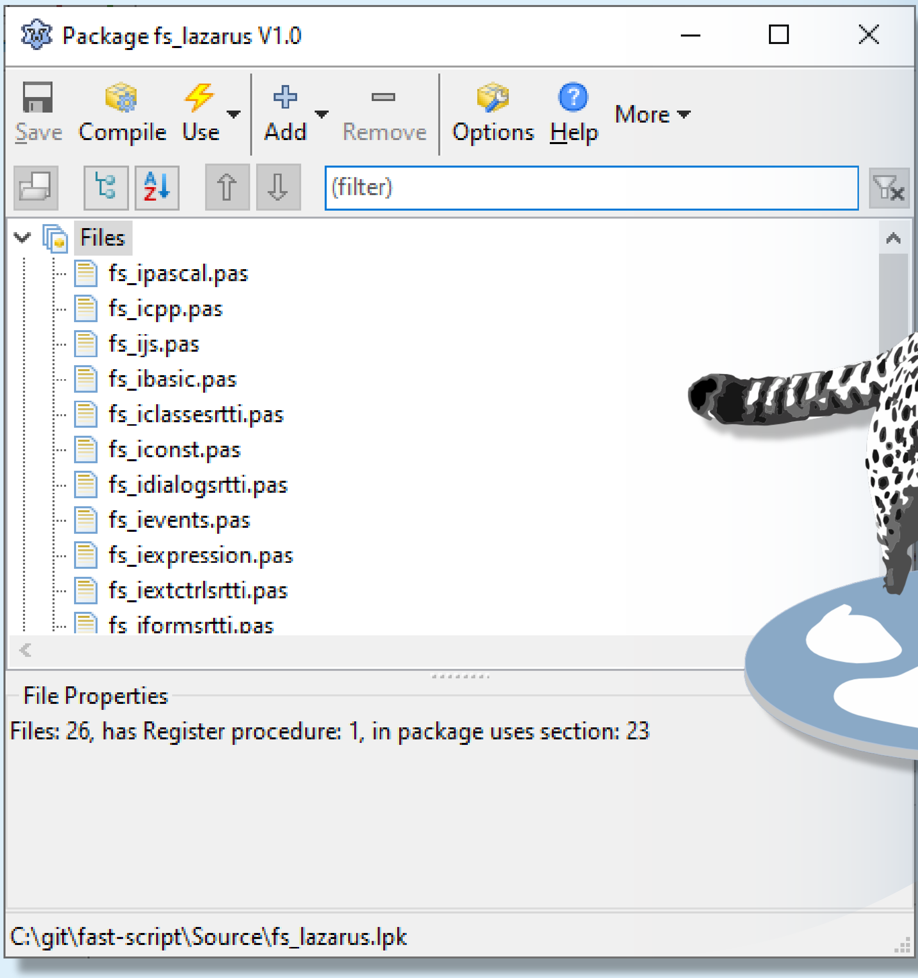




INSTALLING FASTREPORT PACKAGES IN LAZARUS FOR LINUX/WINDOWS

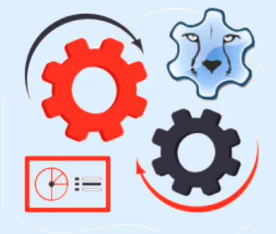
So, we have already installed Lazarus, now let's move on to installing the FastReport VCL report generator packages in Lazarus.

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:



Files: 26, has Register procedure: 1, in package uses section: 23

C:\git\fast-script\Source\fs_lazarus.lpk



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;

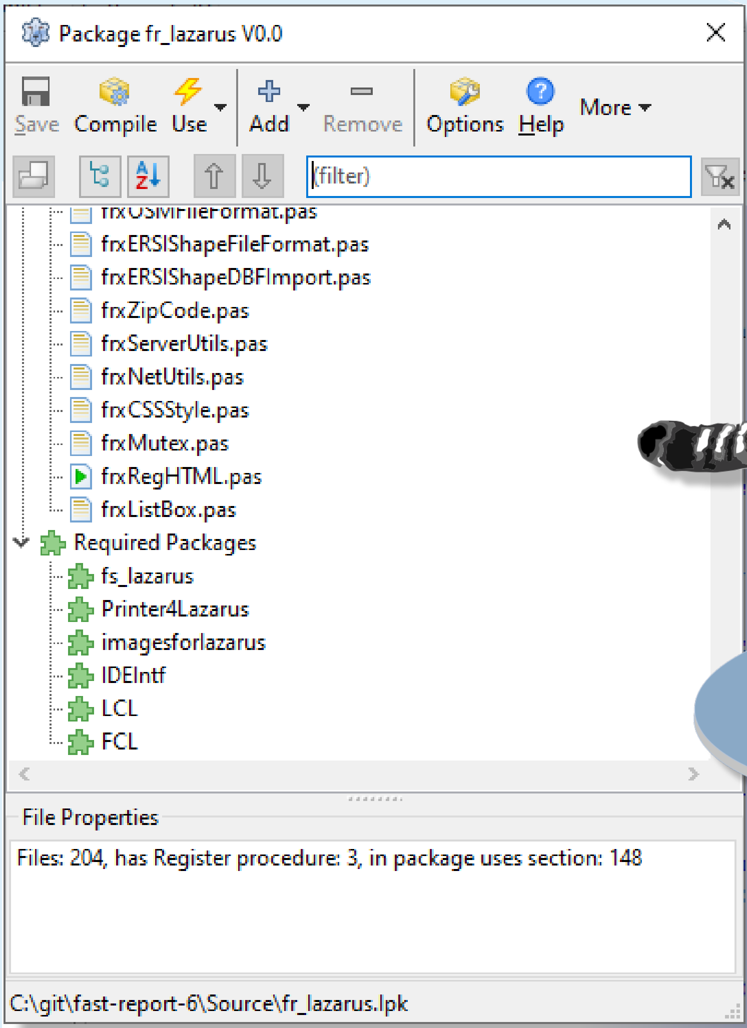
❸ In any order:
`fast-report\Source\ExportPack\frxeN_lazarus.lpk`
package with exports;
`fast-report\Source\\frxchartlazarus.lpk`
package for charts (*diagrams*);
`fast-report\Source\lazdbf\frxlazdbf.lpk`
a package for working with a BDF format database;
`fast-report\Source\sqlite\frxlazsqlite.lpk`
a 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 2.2.6 and this error has maybe 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.



frxOSMFileFormat.pas
frxERSIShapeFileFormat.pas
frxERSIShapeDBFImport.pas
frxZipCode.pas
frxServerUtils.pas
frxNetUtils.pas
frxCSSStyle.pas
frxMutex.pas
frxRegHTML.pas
frxListBox.pas
Required Packages
fs_lazarus
Printer4Lazarus
imagesforlazarus
IDEIntf
LCL
FCL

File Properties

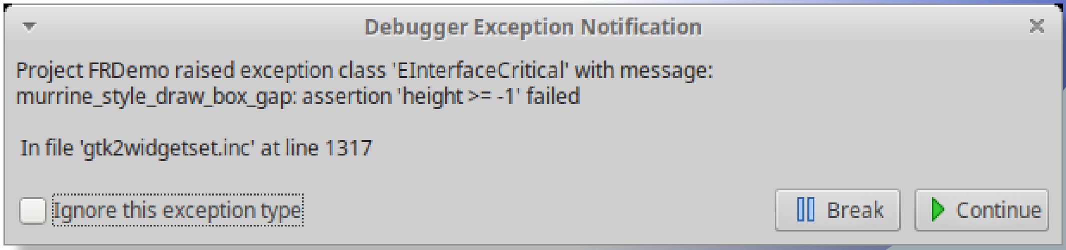
Files: 204, has Register procedure: 3, in package uses section: 148

C:\git\fast-report-6\Source\fr_lazarus.lpk

To do this, double-click on it and recompile, and then reinstall.

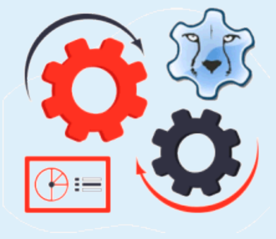
After successful installation of all the packages, click **Project -> Open Project** and open the `fast-report\LDemo\FRDemo.lpi` project and try to run it, then click the Design button.

If you get this negative height error on **Linux**:



Project FRDemo raised exception class 'EInterfaceCritical' with message:
murrine_style_draw_box_gap: assertion 'height >= -1' failed

In file 'gtk2widgetset.inc' at line 1317



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 the uses section.

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

If you are from Ukrainian origin you can get a free Subscription for Blaise Pascal Magazine, we will also give you a free pdf version of the Lazarus Handbook. You need to send us your Ukrainian Name and Ukrainian email address (*that still works for you*), so that it proofs you are real Ukrainian. please send it to editor@blaisepascal.eu and you will receive your book and subscription

BLAISE PASCAL MAGAZINE

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

Blaise Pascal

COMPONENTS DEVELOPERS 4

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

COMPONENTS DEVELOPERS 4

COMPONENTS DEVELOPERS 4

June 21 2023

Kim Madsen will do a live presentation for a

DAPUG erfa meeting

in Fredericia Denmark

DAPUG is the Danish Delphi user group which started out in 1989, in the days where Turbo Pascal, Turbo C and Paradox were hugely popular products from Borland.

So at the time, the acronym DAPUG meant “DAnish Paradox User Group”, but today their focus is primarily on Delphi and Pascal and the acronym changed meaning to “Database Application Programmers Users Group”.

Through the times, they have hosted prominent meetings with many of the known faces from Borland and the Delphi world.

Kim will do a 2 – 2.5 hour presentation of some of kbmMW, including

- **kbmMW? What is it?**
- **ORM**
- **SmartServices (REST and more)**
- **SmartBinding**

Kim will during the presentation get to slightly touch logging, transport layers, object notation frameworks, configuration, authorization and more.

As far as known the session is open for any to join (*physical attention only*), but if you are not a DAPUG member you will need to pay a fee for lunch.

Find contact information here:

https://www.dapug.dk/p/test-af-pages.html

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

COMPONENTS DEVELOPERS 4

D11

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

NEW RELEASE

kbmMW Professional and Enterprise Edition v. 5.22.00
kbmMemTable v. 7.98.00 Standard and Professional Edition

5.22.00 is a release with containing new stuff, refinements and bugfixes.,**OpenSSL v3 support,** WebSocket support, further improvements to SmartBind, new high performance hashing algorithms, improved RemoteDesktop sample and much more.
This release requires the use of **kbmMemTable** v. 7.97.00 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

NEW RELEASE

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

COMPONENTS 4 DEVELOPERS

D11

ADVERTISEMENT