Welcome to

Cross-Platform Development Using Lazarus

Speaker: Fikret Hasovic

About me

Fikret Hasovic

USAID TARA Project in Bosnia & Herzegovina

Senior programmer/Analyst

Overview

- History
- What is Lazarus?
- GUI? Widget set?
- RAD like Delphi?
- Use existing Delphi code?
- Can I create commercial products with this?
- Where did the name come from?
- Database support
- Download and Install
- Active Lazarus Projects
- Lazarus-CCR Released Components

History

- Lazarus was started in February of 1999. It was primarily founded by three individuals:
  - Cliff Baeseman
  - Shane Miller
  - Michael A. Hess
- All three had attempted to get involved with the Megido project which dissolved. In frustration they started the Lazarus project. It has had a steady growth of supporters and developers during the following years. Of the three founders, only Michael A. Hess is still involved with the project.

  The next oldest member of the team is Marc Weustink. He got involved with the project in Aug. 1999. Following him is Mattias Gaertner who got involved in Sept. 2000.
- Both of them have been the major contributors to the core of what makes Lazarus tick.

What is Lazarus?

- **Lazarus** is a cross platform Visual IDE developed for and supported by Free Pascal. It aims to provide a Delphi Clone for Pascal and Object Pascal developers using the open source Free Pascal compiler.
- Lazarus is the class libraries for Free Pascal that emulate Delphi. Free Pascal is a GPL'ed compiler that runs on Linux, Win32, OS/2, 68K and more. Free Pascal is designed to be able to understand and compile Delphi syntax, which is of course OOP. Lazarus is the part of the missing puzzle that will allow you to develop Delphi like programs in all of the above platforms. Unlike Java which strives to be a write once run anywhere, Lazarus and Free Pascal strives for write once compile anywhere. Since the exact same compiler is available on all of the above platforms it means you don't need to do any recoding to produce identical products for different platforms.
  http://www.lazarus.freepascal.org/

GUI? Widget set?

- In Lazarus terminology, this part is called simply "the Interface". Actually it's more like one Interface per Widget toolkit.
- The current status of **Widget toolkit Interface** is roughly like this:
  - **win32 GDI** support (win32 native) is in mainstream use.
  - GTK+ 1.2.x is in mainstream use (Unix derivates including Mac OS X)
  - **GTK+ 2.x** is under heavy development, and is almost done. Mostly interesting because of better internationalisation support.
  - **Qt (3,4)** (C++) has headers translated, and have some interface modules.
  - for **wxWidgets** (C++) there is no header translation yet.
  - for **wx.net** (plain C wrapper for WxWindows), a header translation was announced but not publicly available yet. So also no interface yet.
  - for **Aqua** (Mac OS X native toolkit, Objective C) No header translation (via plain C interfaces) yet.
  - for **Carbon** (Mac OS X native toolkit, Objective C), Pascal header translation available and the interface is in development.

LCL & RTL

- The Lazarus GUI subsystem is called LCL (Lazarus Component Library), which is basically a set of visual and non visual component classes over a Widget toolkit dependant part. The LCL has been modelled after the Delphi VCL, but is not 100% compatible, and this is by design out of multi-platform considerations.
- RTL is quite well documented, much better than LCL

LCL & UNICODE

- Lazarus support of the Unicode standard needs further development, mostly in regard to the Windows platform
- The spirit of Lazarus is: "Write once, compile everywhere." This means that, ideally, an Unicode enabled application should have only one Unicode supporting source code version, without any conditional defines in respect to various target platforms.
- The "interface" part of the LCL should support Unicode for the target platforms which support it themselves, concealing at the same time all peculiarities from the application programmer.
- What concerns Lazarus, the internal string communication at the boundaries "Application code <--> LCL", as well as "LCL <--> Widgetsets" is based on the classical (byte oriented) strings. Logically, their contents should be encoded according to the UTF-8.

LCL & UNICODE

- Currently these are various groups of widgetsets, according to the encoding:
  - Interfaces that use ANSI encoding: win32 and gtk (1) interfaces.
  - Interfaces that use UTF-8 encoding: gtk (1), gtk2, qt, fpGUI
  - Interfaces that currently use ANSI encoding, but need migration to UTF-8: win32, wince

Notice that gtk 1 is on both ANSI and UTF-8 groups. That´s because the encoding is controled by an enviroment variable on Gtk 1.

•As Lazarus is today, existing software will work, if recompiled for win32, wince or gtk interfaces, but will face encoding issues compiling for other widgetset. And new software, using UTF-8 will work when recompiled for any of the widgetsets on the Unicode group.

- One very **important** note is that you must use the IDE compiled for the same group you are targeting. This is because the IDE uses the encoding of the widgetset it was compiled to, and not the one of the target widgetset to write LFM and LRS files.

Cross compiling for Win32 under Linux

- Since 0.9.10 there is an rpm 'fpc-crosswin32', that installs the needed binutils (e.g. cross assembler, cross linker), the fpc .ppu files cross compiled for win32 and modifies /etc/fpc.cfg.
- It does not contain the cross compiled LCL .ppu files. You need to compile them yourself, after installing fpc-crosswin32.
- Why cross compiling: FreePascal is a compiler and basically converts source into binaries (machine language). These binaries also contains information, how the operating system starts the executable. Therefore these binaries are platform specific. FreePascal itself does not need much setup. It can create binaries for many platforms. But the compiler is only one part. There is also the assembler and the linker. And these tools are not able to create cross code. That's why we have to create special linker 'ld' and assembler 'as' for every target platform. These are the binutils. After creating the cross tools, all the fpc pascal units will be cross compiled.

RAD like Delphi?

- It sure is. Is it totally completed? No not yet. The forms design portion is still in need of a great deal of development. The over all IDE is complete and can be used for most programming needs. Several aspects of the project are still in need of help.

Lazarus & FPC vs Delphi

- Advantages of Lazarus
  - The first and the biggest advantage point of Lazarus over Delphi is his **multiplatform** capability. I think FPC is the second (native) compiler after gcc regarding number of platforms supported.
  - Lazarus handles include files (.inc) very well, as if they're integrated parts of the main source file (.pas).
  - Almost every visual component has public **Canvas** property. This is really fun because we could add extra drawing to any visual components without being bothered to derive it into a new component.
- Disadvantages of Lazarus
  - Since **FPC** has no intention to support .**Net** then obviously Lazarus won't run upon .Net either, not even on Mono.
  - Lacks of (complete and stable) widgetsets (user interface) library binding. Until this version, only ··win32 and gtk1 widgetsets that are considerably complete and working on Lazarus. Though some other widgetsets are also available (qt4, gtk2, winCE, carbon, etc) but not as complete and stable as win32 and gtk1 widgetsets. gtk2 is almost complete...

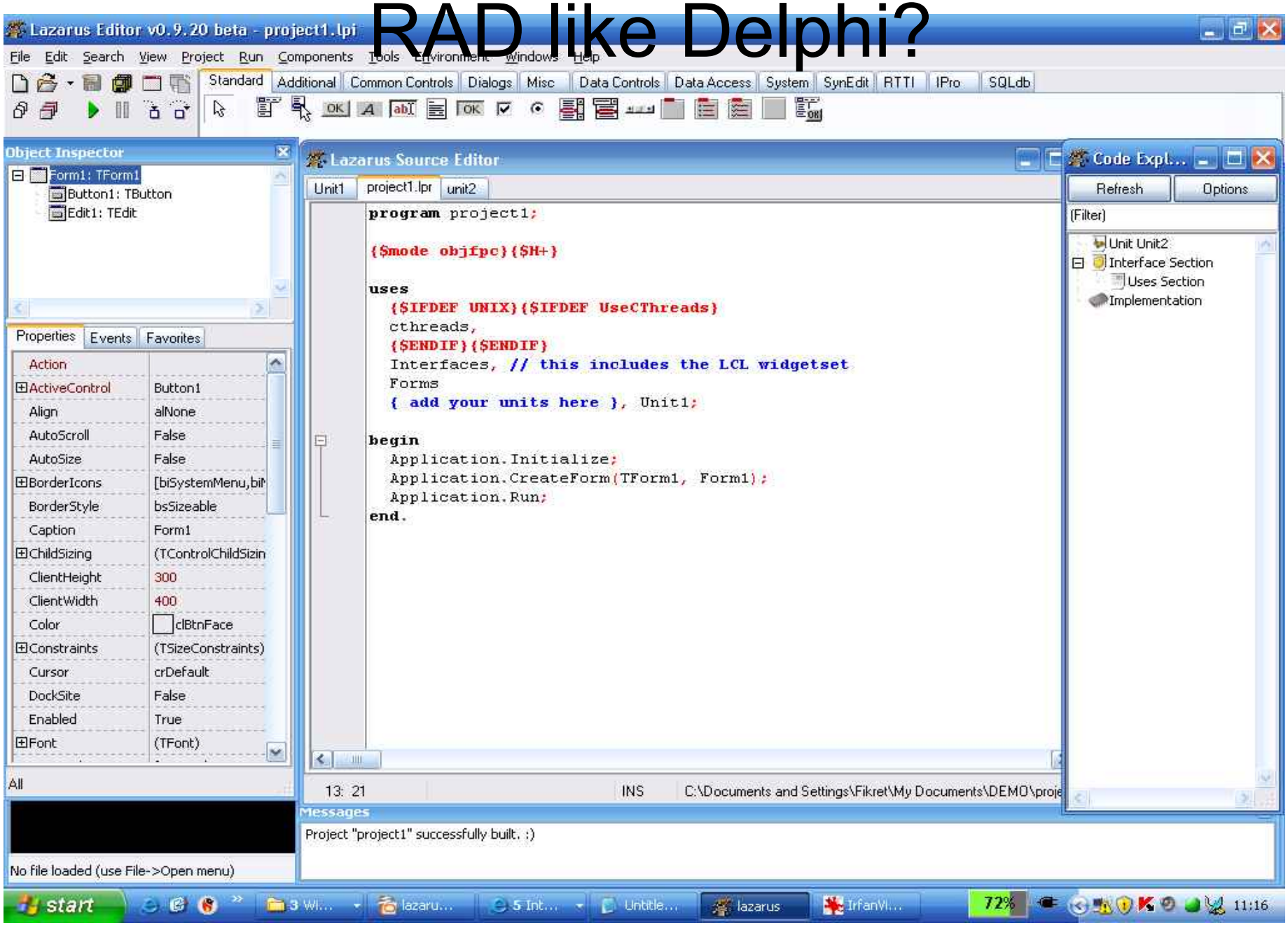
Lazarus & FPC vs Delphi

- Advantages of Delphi
  - If you're a .Net fan and happy to bound yourself to M$ but don't want to learn other (new) languages, then Delphi is all you need. It is still the best object pascal IDE on .Net world, IMO.
  - Delphi IDE -especially the source editor- is marvelous! This is the thing that keep me using Delphi. I usually write code in Delphi and compile it using Lazarus.
- Disadvantages of Delphi
  - Obviously Delphi is getting more and more tied to M$ Windows environment (and .Net oriented) which is very regretted by many of users because Delphi is the biggest supporter of object pascal language. We need **Kylix** back!!!

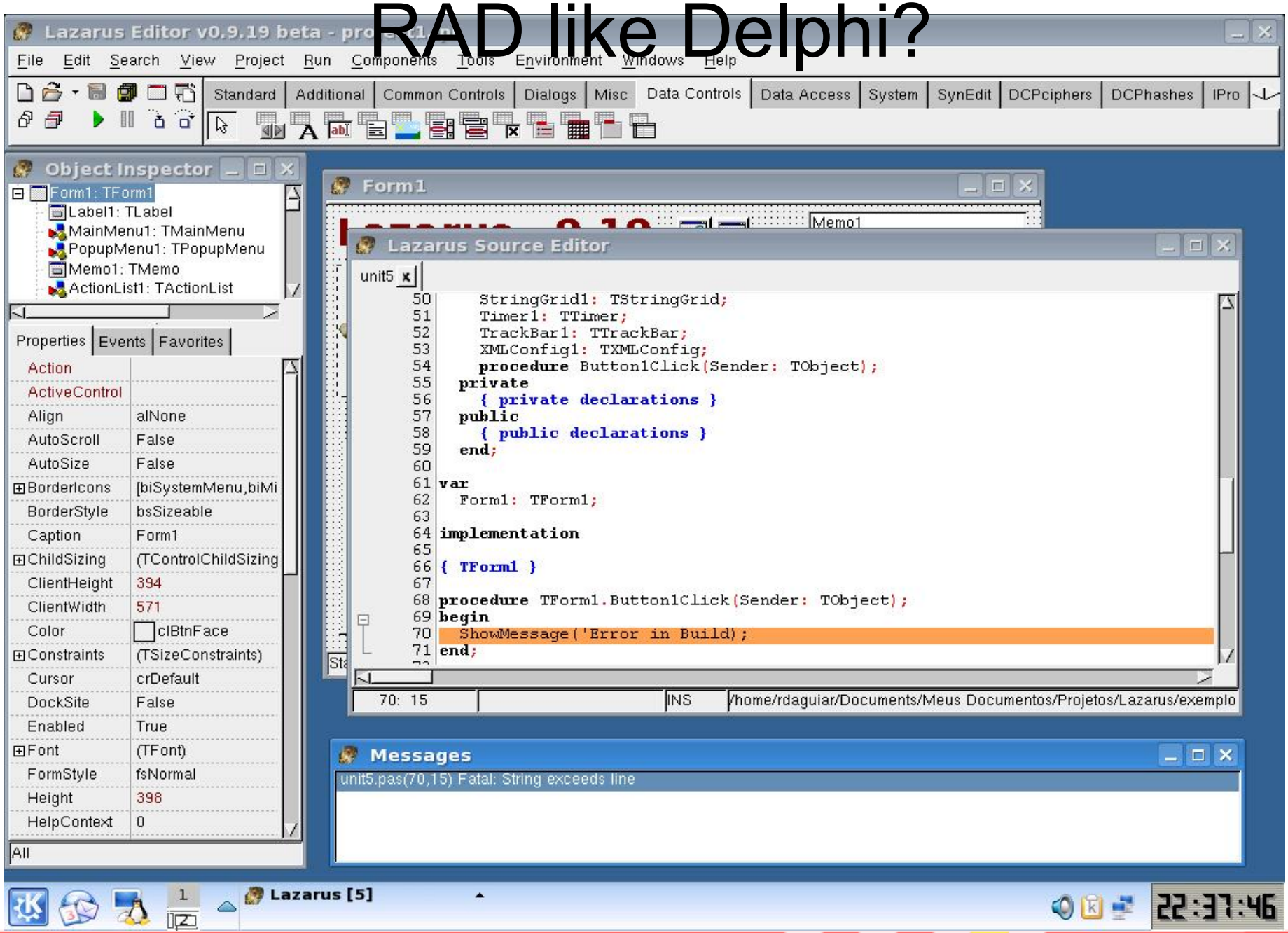
Lazarus & FPC vs Delphi

- Advantages of working with both
  - The compatibility between the two is “one direction”, I mean while Lazarus always try to be compatible with Delphi but on the other side Delphi (seems to) ignore Lazarus existence. Though not fully 100% compatible (but not far from that), we still could share common codes (and resources) between them. We hope that things will change with DevCo!
  - As a result from compatibility point above, the lack of third-party component on Lazarus automatically got covered by Delphi resources (and community). In fact, a Lazarus user usually also a Delphi user, I’m one of the examples.
- Conclusions
- Basically, Delphi and FPC/Lazarus is complement to each other. So, I’m not quite agree if one say both are competing to each other, or even enemy. Saying one is superior than another (in all aspects) is not quite right either. Both Delphi and FPC/Lazarus is object pascal IDE and compiler, by using them both, I got good things from each.

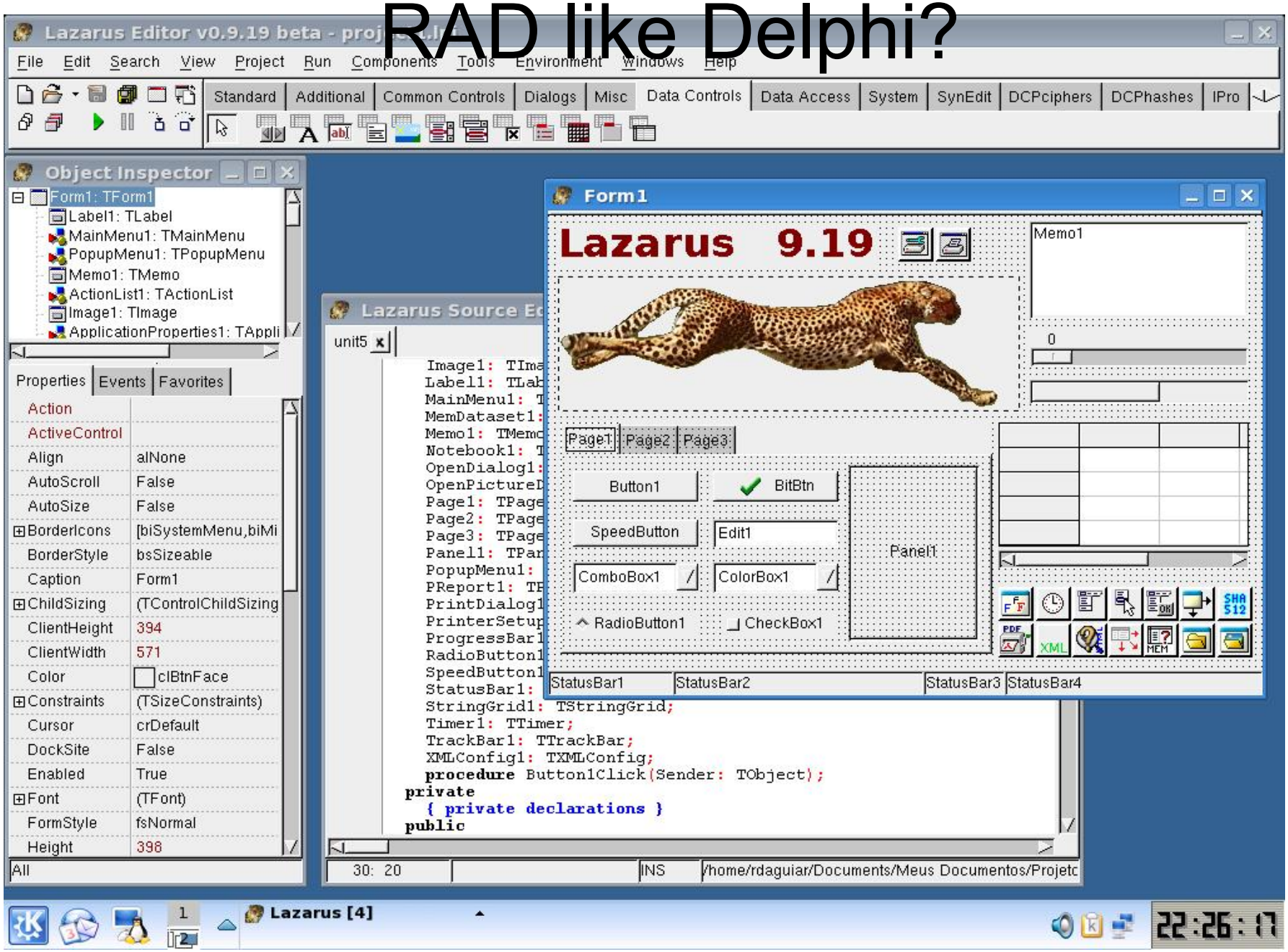
RAD like Delphi?



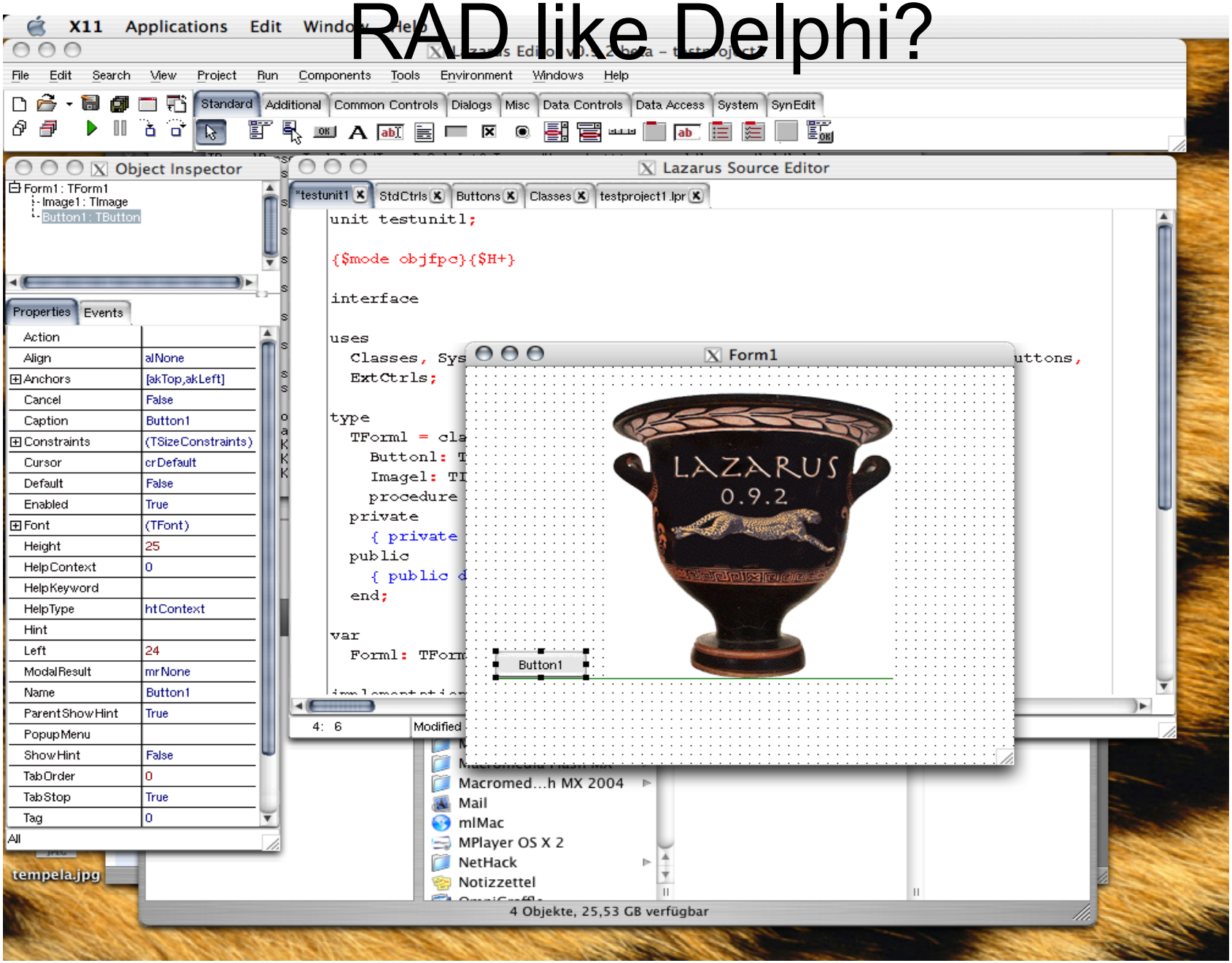
RAD like Delphi?



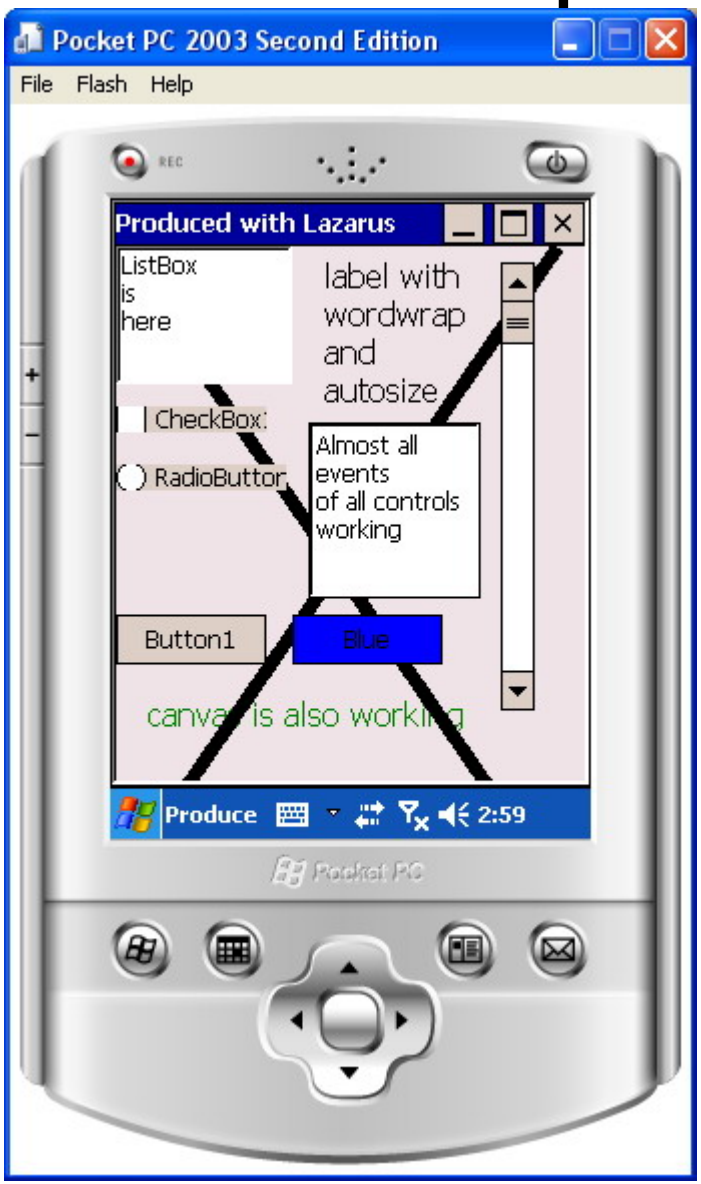
RAD like Delphi?



RAD like Delphi?



RAD like Delphi?



Versions released in last 12 months

- **Lazarus 0.9.12** released **2006-02-08** based on **fpc 2.0.2** and the binary packages now contain many standard packages:RunTimeTypeInfoControls, Printer4Lazarus, CGILaz, CGILazIDE,MemDSLaz, SDFLaz, TurboPowerIPro, JPEGForLazarus,FPCUnitTestRunner, FPCUnitIDE, ProjTemplates

  Changes: carbon interface: implemented basic key handling, less dependencies,better integration in IDE, improved mouse handling, a tool to create normal MacOSX application resource files, and more.

  gtk2 intf can now optionally use the file chooses,compile with HasGTK2_6

  removed htmllite package. Use turbopower ipro instead.

Versions released in last 12 months

- **Lazarus 0.9.16** released **2006-05-28** based on **fpc 2.0.2**

Changes: IDE Online Help for IDE windows
Help for LCL applications
The default is to use F1 as help key.
qt interface: many improvements
XML streaming: There are now functions to save/load components to/from xml documents.
New IDE menu item: designer -> right click -> popup menu -> Save form as xml

windows installer scripts: ldw and asw are not needed anymore
added scripts for making macosx snapshots
added gtk TMemo font and color implementation
added {$inline on} so that building lazarus does not depend on -Si in fpc.cfg
win32 interface: use ownerdrawn menu items, so images can be shown better
and several hundred bug fixes and minor changes.

Versions released in last 12 months

- **Lazarus 0.9.18** released **2006-09-29** based on **fpc 2.0.4**

Changes: lazbuild: There is now a command line tool to compile projects and packages called 'lazbuild'.
important for win32 users: win32 installer: all fpc binaries and sources are now in the fpc subdirectory.
win32 installer: localization of context menus
debian packages for lazarus and fpc-crosswin32 (for example ubuntu)
added scripts to create slackware package for lazarus
'make install' now supports INSTALL_PREFIX
improved rpm package for SuSE
carbon interface improvements
gtk interface improvements
gtk2 interface improvements
win32 interface improvements
qt interface improvements
updated translations
and many other improvements, bug fixes and minor changes.

Versions released in last 12 months

- **Lazarus 0.9.20** released **2006-11-08** based on **fpc 2.0.4**

Changes: New widgetset: fpgui
updated translations
IDE: renaming a component now automatically renames methods with default names, and inherited components
LCL and IDE improvements
sqldb package: added TOracleConnection to component palette
win32 installer: fixed including the binutils from fpcbuild
creating package makefiles: the win64 OS uses the win32 widget set
debugger improvements
fpcunit: new console runner uses new fpcunitconsolerunner package
qt intf: Patch from zeljko. Implements TQtAbstractSlider, TqtScrollBar, TQtTrackBar, TQtPen (CreatePenIndirect) and TQtRegion (CreateRectRgn).
Unified qt4 header files. Updated bindings to 1.21 added support for menus, font, progressbar and statusbar Combobox
And two hundred fixes and minor improvements.

Version Numbering

- The most important thing to know is that if the last number of the version is an even value, it's a stable/published release. For example version 0.9.16 is released, and will never change, ever.
- But the developers are working in an ongoing version, which changes every day. Those versions have odd last numbers. Thus after the moment that 0.9.16 was released, the developers were working on version 0.9.17. This version is maintained using SVN, every patch gets a 'revision'-number.

Road To 1.0

- When new bugs are entered, developers try to give them a target in which version the bug will be fixed. If a bug is set to *post 1.0*, that means the developers think this bug is not important enough to block a 1.0 release. In order to have a 1.0 sooner rather than later, developers will leave those bugs for later. Of course you can make sure these *post 1.0* issues are fixed in the 1.0 release by providing patches for these issues.
- Some criteria are:
  - Only **gtk1** and **win32** widget sets are stable in 1.0. So bugs for other widget set (gtk2, carbon) are set to post 1.0.
  - Until the 1.0 there will be a feature freeze. New features and components generally get a post 1.0 target. Bugs affecting stability have a higher priority than bugs fixing the implementation of a property.
  - Some components are not stable enough and should be disabled for 1.0. If they are disabled, then fixing them before 1.0 will not be necessary.

Use existing Delphi code?

- Some of it, yes. If the code is standard Delphi pascal and it uses the standard components found in Delphi then the answer is yes. If it uses some specific database, OCX, or DCU then the answer would be no. These items are specific to Windows and would only work on and within Windows. However, if you are only looking to create a Windows product using Free Pascal and Lazarus then the answer would be yes. This hasn't been added to the LCL yet but it should be possible in the future.

Can I create commercial products with Lazarus & FPC?

- Yes. The code for the Free Pascal compiler is licensed under the GPL. This means that it is open source, free, whatever name you want to stick to it. You can modify the code if you wish but you MUST distribute those changes or make them available to others if they wish to use it.

- The FCL (Free Pascal Component Libraries) and the LCL (which will eventually become part of the FCL) are licensed under a modified LGPL. In a nut shell this means that you can write your own proprietary software that just links to these libraries. You can sell your application without the need to supply or make available your code. However, as with the compiler if you make modifications to the FCL or LCL you must make those changes available to the general public and the world.

I give up, where did the name come from?

- One of the original projects that made an attempt to build a Delphi clone was Megido. However this effort died. Lazarus as you know was the biblical figure that was raised from the dead by Christ. Soooooo. The project is named Lazarus as it was started/raised from the death of Megido.

Database support

- Lazarus supports several databases out-of-the-box, however the developer must install the adequate packages for each one. The programmer can access the database throught code or by dropping components on a form.
- The **following databases** are supported out-of-the-box:
  - **PostgreSQL** requires the PSQL package
  - **dBase** and **FoxPro** can be supported without the need to a external server or library throught the TDbf component
  - **MySQL** works
  - **SQLite** needs only a single external library and the TSqliteDataset component
  - **MSSQL** is working with Zeoslib
  - **InterBase** / **Firebird** also work with the latest ZeosLib and UIB2

Installation

- *Installation under Windows*
- *Installation under Linux*
- *Other OS's*

Active Lazarus Projects

- **The Synapse Project**
  provides a serial port and synchronous TCP/IP Library
- **The Icebox**
- **TruckBites**
  Business management software for independant trucking companies and owner/operators (for the USA.)
- **Project Theseus**
  rapid deployment and distribution system for Linux called Epik-Builder
- **SilentCoder's site**
  DireqCafe A complete and full full featured internet cafe solution for LTSP
- **GTK-Fireadmin**
  GTK based Firebird Administration tool
- **GTK2forpascal**
  bindings for pascal to use the gtk2 libraries

Active Lazarus Projects

- **Cactus**
  Cactus is an audio player that comes with a database to organize your mp3 file collection.
- **OggBase**
  program for managing your Ogg-Vorbis files in a Database.
- **The Light Pascal Toolkit (LPTK)**
- **CUPS for lazarus**
  bindings for pascal to use the CUPS
- **LazReport** **Report generator based on FreeReport 2.32.**
- **Indy for lazarus** ***http://indy4lazarus.sourceforge.net/***

Active Lazarus Projects

- **Lazarus *SQL Explorer***
  General Database IDE for all databases supported by SQLDB suite.
- ***Seksi Commander***
  GPL Filemanager for Linux. Integrated bin, text, hex viewer and editor based on SynEdit.
- ***Synaser***
  Library for serial communication (Linux, Windows) from author of Synapse.
- ***Dedalu***
  A collection of small and simple projects.They are games, editors, utilities, etc.
- ***Audio Component Suite***
  A collection of components to develop applications for audio playing/recording/processing.
- ***GLScene*** *A complete 3D graphics library using OpenGL for rendering.*
- ***ISA Digital Osciloscope***

Lazarus-CCR Released Components

- PowerPDF
- ColorPalette
- DCPcrypt
- Fshcomp
- HistoryFiles
- httpd
- LazRGBGraphics
- LazRichView
- libview
- lNet
- MDButtonsBar

Lazarus-CCR Released Components

- MultiLogViewer
- OnGuard
- PascalMagick
- PlotPanel
- RingChart and AnalogWatch
- rxfpc
- SMNetGradient
- Spook's Panel Components
- TAChart
- Web Service Toolkit
- ZipFile

Lazarus-CCR Released Components

- ACS
- GLScene
- EpikTimer
- FpSystools
- FreePascalArchivePackage
- TPSQL
- Sockets
- VirtualTreeview
- Zlibar
- ColorBox

Case Study

- *"My day job is Director of IT for a document management company. We scan around 60,000 images of paper medical records on a daily basis. We use a mixture of Linux and Windows servers and workstations, and my network pushes around 30 GB of data around per work shift. We're a small business (14 employees), and providing solutions for our customers is our priority, not just selling a service or project. Read on for how we use Lazarus in our day to day operations.*

  I'm a strong advocate of Linux, so I've attempted to switch as many machines in the office to Linux as possible. This has left me with a mixture of Windows and Linux machines, because the main application we use to produce our product is Win32 only.

Case Study

- *Lazarus and FreePascal have allowed me to write very quick and easy (RAD) utilities that allow us to convert the data and images over to the appropriate format, after being exported from our own server. Since I have some machines in Linux and some in Windows, in most cases I'm able to compile the same conversion utility for either environment, depending upon what server is available at the time. The ease of the Lazarus IDE and RAD design allow us to take a basic framework application that I've developed as a series of objects and modify it for use for new formats as they arise, in a very short period of time.*
- *We also aquired some tape backups from old AIX servers that have MS Word documents embedded within a proprietary database. Windows won't even recognize the tapes. Using a combination of Lazarus, OpenOffice and the standard Linux tape utilities, I'm able to extract the indexes and embedded documents from those tapes convert them to TIFF images using OpenOffice and burn them in a format ready for import in a new system, completely unattended.*

Case Study

- *Lazarus is the glue that helps us provide that "value add" to our customers that many companies lack. Its RAD design, similarity with previous Pascal IDE's that we were already familiar with and cross-platform nature have allowed us to provide services we never would have been able to before.*
-
- *Tony Maro*
- *Dir. Information Technology*
- *HCR Imaging, Inc.*
-
- *http://www.hcrimaging.com"*

Game development

- *Skinhat has made a new install that combines Lazarus, GLScene, OpenBSP and Quark. The purpose of the install is to allow people to quickly and easily develop a 3D game using Lazarus. GLScene is a powerful 3D library and can load and display Quake like maps. A map can be created and edited with Quark. The install is centered around a working demonstration Lazarus application that allows a person to walk through the streets of a town built with Quark. So it is easy to get something up and running straight away.*

  For more information refer to:

  www.skinhat.com/3dpack

Thank you!

End of session

Cross-Platform Development Using Lazarus

Speaker: Fikret Hasovic

fikret.hasovic@gmail.com