UPDATER KIT / SUMMIT 2009 / CHARTS & GRAPHS

SPECIAL MBS EDITION!

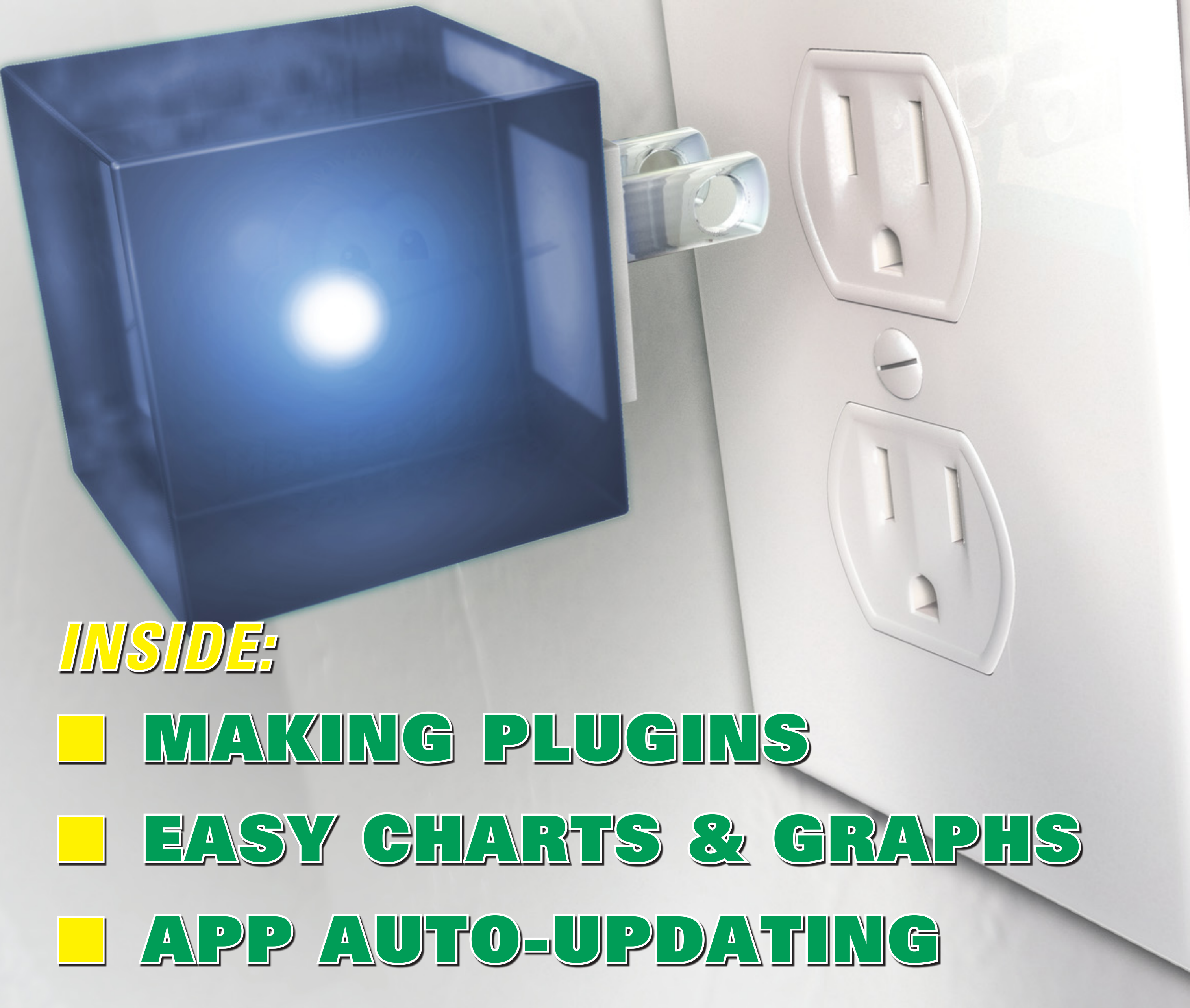
MonkeyBread Software

Published by DesignWrite

SUMMER 2010

REALbasic developer™

www.rbdeveloper.com



INSIDE:

- MAKING PLUGINS
- EASY CHARTS & GRAPHS
- APP AUTO-UPDATING

Free Movie Tutorials!

www.monkeybreadsoftware.de/realbasic/movies/

The MBS Plugin™

The Ultimate REALbasic plugin

*Adds Over **30,000 functions** to REALbasic!*

The Monkeybread Software REALbasic plug-in collection is a big toolbox to help you solving your problems. Thousands of new functions will help you developing better applications!

- Use the CUPS plugin to access the Mac OS X and Linux print system.
- Compress data using BZip2 and ZLib engines
- Copy files asynchronously or synchronously.
- Use the DotMacKit to access .mac/mobileme features.
- **New for Snow Leopard**: Get geo coordinates using CoreLocation / Use OpenCL / New CoreImage filters / Quicklook Preview Panel / New functions for PDFKit and QTKit classes
- Use Spotlight classes to search on a Mac
- Use OCR classes for optical text recognition
- Extract frames from a QuickTime movie
- Use GraphicsMagick to read/write/manipulate over 88 major formats.
- Create your own serial number system for your applications
- Access image metadata (EXIF, IPTC, and others) using CoreGraphics
- Powerful SQL plugin for accessing multiple SQL databases (Oracle, Microsoft SQL Server, DB2, Sybase, Informix, InterBase, SQLBase, MySQL, PostgreSQL, and ODBC, SQLite).
- The PictureMBS class handles pictures of any size using virtual memory.
- Connect to JDBC databases and use java classes using the Java plugin.
- Control USB HID devices on Mac and Windows
- Burn CD/DVDs in your application
- Record audio and video
- Create transparent windows
- Run Javascript in a HTMLviewer control
- Create professional charts using ChartDirector
- Change/fade the screen gamma
- Run PHP scripts in your applications.
- Perform window and screen transitions
- Read and write JPEG, TIFF, PNG, and GIF images.
- Use Dongles to protect your software (HASP, Unikey, and Rockey)
- Use the x-rite Eye-One Device for color measurement
- Handle ICC Profiles, XMP metadata, CMYK images in JPEG and TIFF.
- Cocoa classes: picture taker, color palette, spellchecking, and picture effects.
- Compile, decompile, and execute AppleScripts
- Download and Upload data using the CURL class over FTP, FTPS, HTTP, HTTPS, Gopher, Telnet, DICT, LDAP, and file protocol.
- Perform color management using ColorSync or LCMS.
- Use SpeechSynthesizer and SpeechRecognizer,
- Integrate your app with iCal, iChat, iPhoto, and Addressbook on Mac OS X

Supports Snow Leopard!

PARTS	$30 – $75	See the website for individual part pricing: https://www.monkeybreadsoftware.de/realbasic/buy/plugins-buy-nonEU-new-paypal-us.shtml
COMPLETE *	$294 / $147	Commercial License / Academic License
SQL	$148.50	Access SQL Databases: Oracle, Microsoft SQL Server, DB2, Sybase, Informix, InterBase/Firebird, SQLBase, MySQL, PostgreSQL and ODBC and SQLite
DynaPDF	$898.50 / $1498.50 (Lite/Pro) ($448.50 / 748.50)	Create/import/modify PDF files on Mac/Win/Linux. Includes embedding of fonts and text extraction. (Academic Lite/Pro License)
ChartDirector	$298.50 / $148.50	Professional charts for RB. Commercial License / Academic License

* Includes all plugins *except* independent plugins like DynaPDF, SQL, and ChartDirector.

FREE Trial Download and Full Documentation:

www.monkeybreadsoftware.de

SPECIAL MBS EDITION!

SUMMER**2010**

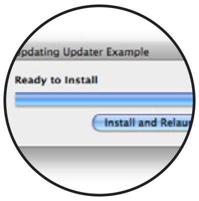
REALbasic developer

www.rbdeveloper.com

THE MAGAZINE FOR REALBASIC® USERS

REALbasic Developer is not affiliated with REAL Software, Inc.

FEATURES


Updater Kit
8

Charts
12 and 20

Plugins
25

RB Summit
31

REALbasic Developer (ISSN 1540-3122) is the ultimate source for tutorials and advanced techniques for programming with the REALbasic language. *REALbasic Developer* is published bi-monthly (six times per year) by DesignWrite, PO Box 872, Lafayette, OR 97127. **Subscription Rate**: Annual subscriptions are $49.99 (order at http://www.rbdeveloper.com/subscribe/). **Customer Service**: For changes of address, go to <http://www.rbdeveloper.com/addresschange.shtml>. For other customer service issues, see <http://www.rbdeveloper.com/support.shtml>. **Back Issues**: Previous issues may be ordered at <http://www.rbdeveloper.com/orders.shtml>. **Printed Issues**: Issues printed in full color may be ordered at <http://rbd.magcloud.com/>.

How to Download Source Code

Article resources available at:

http://www.rbdeveloper.com/browse/

The "browse" website gives every *RBD* article a *permanent web page* where you can find links to downloads, updates and corrections, and more. Note that downloads include a modification date. Every article in *REALbasic Developer* includes an "RBD number" at the beginning presented like this:

RBD#1234

To retrieve an article's resources, follow these steps:

Step 1: Go to the *RBD* website (**http://www.rbdeveloper.com/browse/**).

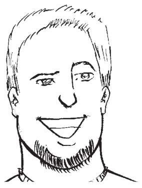
Step 2 (find article by number): In the **Get Article** search field type in the *RBD number* of the article you need.

Alternate Step 2a (find article by issue): Find the issue you'd like to Browse and click on the link for that issue's Table of Contents.

Alternate Step 2b (download all resources): On an issue's Table of Contents page, there's a link to a file which includes *all the downloads* for a particular issue in one large archive.

REALbasic developer
www.rbdeveloper.com

is published by DesignWrite,
P.O. Box 872, Lafayette, OR 97127-0872
and has no affiliation with REAL Software, Inc.



by Marc Zeedar
publisher@rbdeveloper.com

Thoughts from the Publisher

Special MBS Edition

AT A GLANCE

About the Author:
When RBD publisher Marc Zeedar was a kid he used to create magazines just for fun. Now he's doing it for a living!

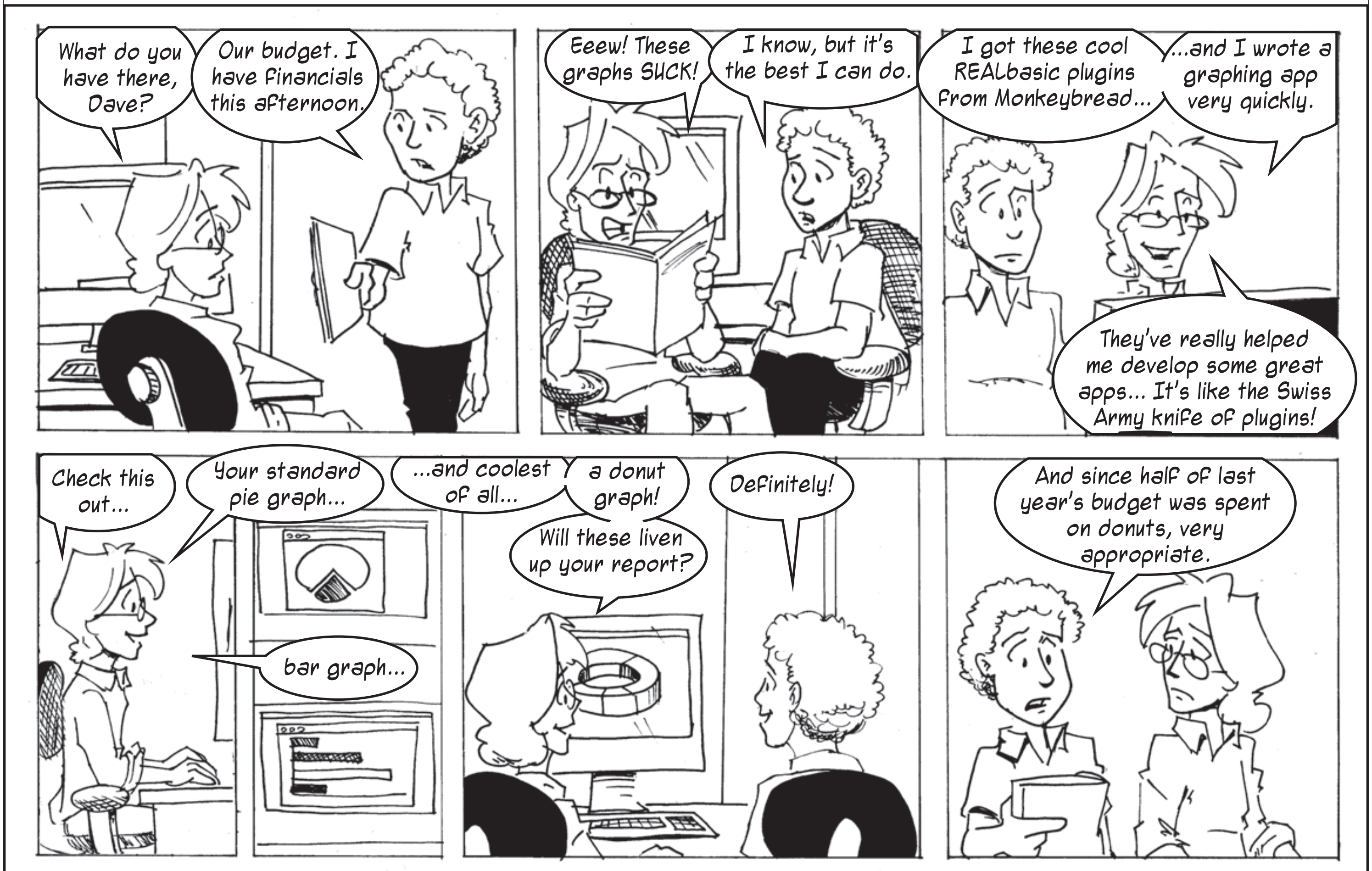
AS Christian Schmitz, author of the fantastic Monkeybread Software plugins for REALbasic, frequently writes articles for *RBD*, we've decided to collect some of his best articles for reprint in a special "MBS Edition" of the magazine for you to enjoy!

If you're unfamiliar with *REALbasic Developer* magazine, we're the premiere source of REALbasic articles, tutorials, and programming instruction. Each issue features over 50 pages of news, reviews, features, and regular columns. We've been publishing bimonthly since 2002 and have produced nearly 50 issues. It is my hope that after enjoying this sample edition you'll decide to subscribe to the full magazine. Note that all our previously published issues are available, either individually or in special bundles (printed or PDF). You can order subscriptions, books, CDs, and other items from our website:

http://www.rbdeveloper.com/orders.shtml

I hope you find this special edition enjoyable and enlightening!

HACKER by Dan Wilson <hacker@rbdeveloper.com>


What do you have there, Dave?
Our budget. I have financials this afternoon.
Eeew! These graphs SUCK!
I know, but it's the best I can do.
I got these cool REALbasic plugins from Monkeybread...
...and I wrote a graphing app very quickly.
They've really helped me develop some great apps... It's like the Swiss Army knife of plugins!
Check this out...
Your standard pie graph...
bar graph...
...and coolest of all...
a donut graph!
Will these liven up your report?
Definitely!
And since half of last year's budget was spent on donuts, very appropriate.

RBD#8401

MBS PLUGINS 10.2

The MBS plug-in comprises a collection of several plug-in parts which extend the REALbasic development environment with 1,200 classes featuring over 26,000 documented functions. This release adds two new plugin parts. One to integrate the mobileMe (.mac) features into your application and another one to control the CUPS printing engine on Mac OS X and Linux. This way you can print files on a printer directly. Also there are several new Cocoa classes including a better integration in the future Cocoa REALbasic target. For compressions there is the zlib compression engine, but now also the BZip2 engine. The plug-ins require REALbasic 2006r4 or newer. While all plug-in parts compile on Mac OS X (universal), Linux, and Windows, each function may depend on additional system requirements to work successfully. Plugin licenses are available for components or for the complete collection. If you buy a current license, you will get free updates for one year.

Product:
Monkeybread Software Plugin for REALBasic

Platforms:
OSX / WIN

Price:
200 Euros (complete set); 20 Euros and up (per component)

Website:
http://www.monkeybreadsoftware.de/realbasic/plugins.shtml

FORMATTED TEXT CONTROL

True North Software has announced the release of version 1.8 of the Formatted Text Control (FTC). The FTC is a canvas based control for REALbasic that implements word processor capabilities similar to Apple Pages or Microsoft Word. The FTC is useful for implementing reports and general word processing capabilities within your REALbasic applications. The FTC features the following capabilities: - Four display modes including page, normal, edit, and single - RTF support including picture support - A full range of paragraph and character styles - Picture support - Custom object support for adding application specific items - Built in undo management - Many customization options to adapt the FTC to your application - Highly optimized code written in REALbasic for maximum flexibility. Version 1.8.2 has 4 new features and enhancements including an RTF parser that is 30% faster! This version also address 14 bug fixes. For a more complete list of features and capabilities, see the "FTC User Guide" and "Version History" documents in the demo download.

Product:
Formatted Text Control 1.8.2

Platforms:
OSX / WIN

Price:
$300 (unencrypted modules, 33% off coupon available on developer's website)

Website:
http://www.truenorthsoftware.com/formattedtextcontrol

MBS RB UPDATER KIT

The MBS REALbasic Updater Kit helps you in adding an automatic update feature to your application. It features: * Crossplatform update engine for Mac OS X and Windows. * Full Source code access. * Using Sparkle on Mac OS X for updating and our own code for Windows. * Includes all code and help to setup Sparkle for Windows. * Includes script file for Inno Setup engine. (you can use others if you like) * Using digital signatures on Mac and MD5 Checksum on Windows to ensure download integrity. * Includes generator for XML file for your server and code to parse this file and find newer versions. * Better language handling with xml lang attributes. * Added italian translation in addition to German and English. * Added console version of Patch-MacApp. * Better handling of registry on Windows. The Updater Kit requires REALbasic 2008 or newer and a license of the MBS REALbasic Complete License.

Product:
MBS REALbasic Updater Kit 1.2

Platforms:
OSX / WIN

Price:
$99 USD or 79 Euro

Website:
http://www.monkeybreadsoftware.de/realbasic/UpdaterKit/

BASECONVERTER PLUGIN

BaseConverter Plugin for REALbasic converts a string containing an integer or non-integer expressed in one base to a string with an expression of the number in another base. The allowed bases range from 2 to 65536. The length of the integer or non-integer is limited only by available memory. Note that no multi-precision engine is involved. Operations are carried out using the string contents. The "digits" are appropriately chosen from the ordered sequence: 0123456789ABCD EFGHIJKLMNOPQRSTUVW XYZ<36><37>...<65534><6 5535> "Digits" in the alphabet can be entered in either lower case or upper case.

Product:
BaseConverter Plugin

Platforms:
All

Price:
Free

Website:
http://homepage.mac.com/delaneyrm/BaseConverterPlugin.html

HELPLOGIC 1.6.1

Electric Butterfly, Inc. proudly announces the immediate availability of HelpLogic 1.6.1, the award-winning help authoring solution for Mac OS X developers. Easily create help systems for your software and web sites from a single source! HelpLogic's innovative design removes the development headaches usually associated with help authoring, providing an ideal solution for Mac software developers, web site designers, and documentation writers. Save time with the integrated Workshop, Visual TOC Builder, HTML Editor, and Link Manager to quickly generate Web-based Help, Apple Help, Microsoft HTML Help, UniHelp, and PDF. This new 1.6.1 version adds improved Snow Leopard compatibility, new publishing enhancements, a few new features, and several minor bug fixes! This recommended update is a free upgrade for all existing HelpLogic customers. And if you're not yet a customer, download the free trial of HelpLogic and take it for a test drive!

Product:
HelpLogic 1.6.1

Platforms:
OSX

Price:
$99 (academic and multiple license volume discounts are also available)

Website:
http://www.ebutterfly.com/helplogic/

FULL KEYBOARD ACCESS

Provides full keyboard access on OS X. The problem: All/mostly controls in REALbasic have properties like AcceptFocus and TabStop, that could be set by the programmer. But, these properties should be set by the End User, from the OS System Preferences! (and *not* by the programmer). This module help you fix this problems, and the Full Keyboard Access on OS X would work as it is expected to do. In this version of Full Keyboard Access, we added an Canvas Mac OS X Help Button, that also works with Full Keyboard Access On!

Product:
Full Keyboard Access

Platforms:
OSX

Price:
Free

Website:
http://seeditmaxi.cachefly.net/rb/downloads/FullKeyboardAccess5.zip

ISCREENSAVER 4.0

Multimedia software producer Xochi Media Inc announces the release of iScreensaver Designer version 4.0, the world's only multi-platform screensaver editor. First released in 1998, iScreensaver Designer is a cross-platform screensaver editor for Macintosh and Windows. iScreensaver will convert QuickTime-compatible images, movies, and Adobe Flash (SWF) files into a standard screensaver with user-friendly installer. Version 4.0 provides support for the next generation of hardware and software: Mac OS X 10.6, Windows 7, and 64-bit CPUs with fast OpenGL graphics cards. All iScreensaver users are encouraged to upgrade.

Product:
iScreensaver 4.0

Platforms:
OSX / WIN

Price:
Free (trial), $250 (Pro), $500 (Ultimate)

Website:
http://iscreensaver.com/

RB3D GL V0.35

GL is a complete REALbasic-written replacement of RB3D, the (now depreciated) 3D API built into REALbasic. This new release primarily focuses on bug fixes. One of the fixes means that the trackball-like demo included with every version now actually works properly. New items for this release: - glCopyTexSubImage2D was incorrectly declared for RB2009r2 and lower (correctly declared for RB2009r3 and above, however). - MatrixMath.Mult4x4Matrix multiplication order switched. Previously multiplied B x A instead of the expected A x B. - MatrixMath Module removes matrix stuff from glManager Module. - glGroup3D.Opacity is now working. - Update to Quaternion.MultiplyBy. Older version may have been inaccurate. - OpenGL declares to built-in OpenGL Module now only for RB2009r3 and above. - Stability improvements to grabbing large screenshots. - Texturing mistakes cleaned up. Before, it was possible that a texture might be applied to the wrong object under certain circumstances. Note that more fixes may be still forthcoming in this area.

Product:
RB3D GL v0.35

Platforms:
OSX / WIN

Price:
Free

Website:
http://www.jcquan.com/REALbasic/index.html

NEW DYNAPDF PRICING

Monkeybread Software offers new REALbasic DynaPDF Plug-in Licenses. For adding PDF creation to your applications, you can now use the new DynaPDF Starter License for just 149 Euro or $199 USD.

Product:
Monkeybread Software DynaPDF Plugin for REALBasic

Platforms:
OSX / WIN

Price:
$199 / 149 Euro (Starter; new license), 449 Euro (Lite; was 599), 799 Euro (Pro; was 999), 1299 Euro (Enterprise; was 1499), 199 Euro (Lite Academic; was 299), 399 Euro (Pro Academic; was 499)

Website:
http://www.monkeybreadsoftware.de/realbasic/plugins.shtml

NEW RSREPORT DESIGNER

Roth Soft has written a whole set of Report-Classes (RSReport) in pure REALbasic. With RSReport you can create static or dynamic reports in an object-oriented manner, show them in a print-preview, of course print them - and even save them as .pdf. This on both OSX and Windows - without the need of any Plugin or PDF-Converter/Printer. It's all written in pure REALbasic. The following basic drawing elements are available: Elements: Text, Line, Oval, Rectangle, Checkbox, Image, RTF-Text, Triangle Charts: Line and Bar. With RSReport Designer, you can visually design the reports instead of creating reports only by code. Designed and coded reports can be combined, allowing very complex reports to be created. The Designer can be used and included in your software as: a Window, a standalone app - or embedded as a Container-Control.

Product:
RSReport Designer 2010.2.0

Platforms:
OSX / WIN

Price:
$86

Website:
http://www.rothsoft.ch/realbasic/rsreport/

RB DEV PACK 4.1

Radical Breeze has released the RB Dev Pack 4.1, a suite of components for REALbasic and REAL Studio that provides additional functionality for the development platform. All features of the RB Dev Pack (including RSS feed reading, graphical effects and user interface components) are supported on all platforms that REALbasic supports (Linux, Windows and MacOS X). Components in the RB Dev Pack include: - radRSS (RSS/ATOM feed reader) - radGraphics (Advanced image effects and processing) - radColorBar (Progress-bar style control with multiple appearances and customization options) - radThumbnailView (Image thumbnail browsing) - radLinkLabel (Clickable URL-like control) - radPicView (Zoomable, draggable, scroll-able picture viewer) - radSmoothResizeWindow (Smooth window resizing animations)

Product:
RB Dev Pack 4.1

Platforms:
All

Price:
$99 annual subscription

Website:
http://www.radicalbreeze.com

WSL-AMAZON 3

Electric Butterfly, Inc. proudly announces the free open source release of WSL-Amazon 3, the award-winning, cross-platform REALbasic class library for accessing the Amazon Product Advertising API. Version 3 has been updated to work with Amazon's new authentication signature requirement. This HTTPSocket subclass wraps the Amazon Product Advertising API into an easy-to-use library of more than 30 powerful methods and 150 properties with support for their US, UK, German, French, Canadian, and Japanese Amazon stores. Includes extensive search and information retrieval capabilities—all from within your REALbasic applications! This free open source class library also comes with the project source code for a fully-functional Amazon-powered application, which you can reuse in your own REALbasic projects!

Product:
WSL-Amazon 3

Platforms:
All

Price:
Free (open source)

Website:
http://www.ebutterfly.com/rb/wslamazon.php

by Christian Schmitz

Updater Kit

Automatic updates for your application

AT A GLANCE

RBD#8409

Target Reader:
Intermediate

Source Code:
No

RB Version Required:
2008+

Platform(s) Supported:
Mac OS X, Windows

Platform(s) Unsupported:
Linux

About the Author:
Christian Schmitz is the creator of the Monkeybread Software REALbasic Plugins (including the ChartDirector plugin).

WITH the Updater Kit from Monkeybread Software you can add automatically software updates to your REALbasic application on Mac and Windows. So this article describes what the kit contains and how to get started.

The Updater Kit is delivered as pure REALbasic code. So there is no limitation like encrypted classes, but also this means that there is no way to test it before you use it. If you like to see a demonstration, you can download the updater example from http://www.monkeybreadsoftware.de/realbasic/UpdaterKit/. This is the test project coming with the kit compiled for Mac and Windows. Once you run it, it will find the update and install it.

To actually use the updater kit, you need in the current version at least 7 plug-ins from Monkeybread Software: Cocoa, CocoaBase, MacOSX, Main, Network, Util and Win. While this list could be reduced, we want to keep all possibilities open to add new functionality and use other plugins, too. So for the current release licenses are only given to people who do have a license for the Complete plug-in set.

Parts of the Updater Kit

Once you bought a license, you receive an email with a download link. The archive you download contains the following items:

AppCast Creator

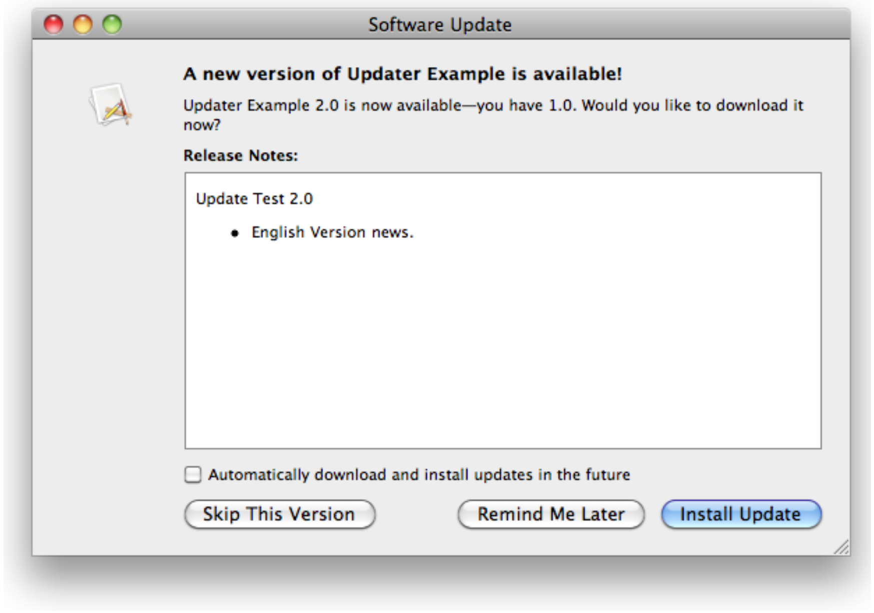
This is a little project to create appcast files for you. You may want to extend this project and add it to your release cycle. Some things like the digital signature and the MD5 checksum are automated here.

Drop file to see MD5 Checksum.rbp

A simply tool to show the MD5 Checksum of a given file. If you use the AppCast Creator, you won't need it, but if you do the appcast manually it is good to have an application to calculate the md5 checksum. Of course you can do the same in the terminal on Mac OS X with the md5 command. Simply type "md5", a space and the path to the file. For the path you can simply drop the file on the terminal window.

Manual

This PDF file is the manual for the Updater Kit.



A new version of Updater Example is available!

Updater Example 2.0 is now available—you have 1.0. Would you like to download it now?

Release Notes:

Update Test 2.0

- English Version news.

☐ Automatically download and install updates in the future

Figure 1: *English Dialog on Mac OS X*

New Version App.rbp

This is our test application project for the updating. We compiled this project and put it on the server as the update to download. So once you updated our test project, you get this application running.

on the server

This folder contains the files we have on our monkeybreadsoftware.de server in the /UpdaterTest folder.

We have here the UpdaterExampleMac.zip, which is the zip archive with the new version of the example application. The file UpdaterExampleWin.exe is an installer to install the new version of the example application for Windows. The appcast.xml is the xml file with the updates readable for the Update Kit. The two description files on the server are for the German and English description which is displayed in the updater window.

You can take a look:
http://www.monkeybreadsoftware.de/UpdaterTest/

PatchMacApp.rbp

This is an application to add the resources needed for the update engine to a Mac application. This includes the Sparkle Framework, the public key and the info.plist entries.

You will certainly want to customize this project for your needs. For example you can turn it into a console application and make it part of a build step in your projects. Or use it in the scripts you use to create the release package of your application. Like on Mac you may want to use a script to create a disc image and copy your application to the disc image. Here you can add the files, which you won't need for debugging.

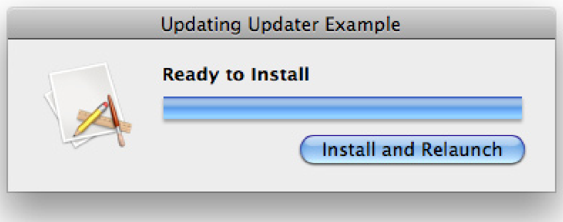


Figure 2: *English download Window on Mac OS X*

Sparkle

This folder contains files from the Sparkle updater engine. You should use the ruby scripts there to create your own pair of keys for the signing process.

To create a private key, you need to open the terminal application on Mac. Type "cd" for the change directory command. Drop your Sparkle folder on the terminal window and press return. Now you moved into the Sparkle folder. Here you can type "ruby ./generate_keys.rb" and press return. Two files will be generated: dsa_priv.pem and dsa_pub.pem. The dsa_priv.pem file contains your private key. Keep it secret. The other file dsa_pub.pem contains your private key which you bundle in your application.

To sign an update you run the other ruby script: sign_update.rb. So launch the terminal again and type "ruby". Now drop 3 files in the terminal window in the right order. First the sign_update.rb script file, second the application zip archive and third the dsa_priv.pem file with your private key. If you drop them right, the terminal command line is complete with all three paths and you can press return. A typical signature looks like "MC4CFQDPWH6GZllRS9Ic bbk+Q5XrddKphAIVAMWgylFuB11TKz DMm5mKoG6o4tWk". Basicly a Base 64 encoded string. Due the way signatures work the string is different on each time you calculate it.

You can find more information here:
http://sparkle.andymatuschak.org/

Updater Example.rbp

This is the example project to show you how this updates work. You can compile it. Once you run it, you can test the update engine. Before testing, you should of course run the patch application to add the resources necessary for Mac OS X.

UpdaterEngine

In this folder you find files to add to your application in order to use the update engine in your application. First there is the central module named UpdaterEngine with all the public methods. Next there is the UpdateHTTPSocket which is a HTTPSocket subclass and does the downloads. If you plan to use SSL on your website to secure your downloads, you can change this class to use SecureHTTPSocket. The UpdateWindow now is the only window our Updater Kit uses. It shows information about updates found.

Windows Installer

This folder contains files we use to create our installers. For our sample here we used Inno Setup Compiler 5.3.4, which is a free setup creation tool. You can find it here: http://www.jrsoftware.org/isinfo.php.

Which tool you use is your own decision. As long as it creates a self containing exe file for Windows, it will work.

Get the sample application running

For the sample we have already a folder on the monkeybreadsoftware.de web server with the appcast file, the archives and the description files.

The newer versions of our test project is created using the “New Version App.rbp” project. We created a Windows installer and uses the Finder compress feature on Mac to create a zip archive. Using the tool “Drop file to see MD5 Checksum.rbp” we got the MD5 checksum for the Windows file and using the ruby scripts in the Sparkle folder, we got a digital signature for the Mac zip archive. All those values with the correct links and size values are in the appcast.xml file.

To try out the example, you can compile the project “Updater Example.rbp”. After you compiled this project, you need to run the project “PatchMacApp.rbp” to install the required resources for the Sparkle engine. Once you launch the example application a second time (updates are never offered at first launch for a better use experience) you see the update window. On Mac it is the Sparkle engine and on Windows our own window.

Now you can run the update. See figures 1 to 3.

Create your own

To get this updater running in your application you need this steps:

First you add the items from the UpdaterEngine folder to your application. This includes currently a window, a module and a class.

In your app event you add a line of code after the registration of the MBS Plugins or any other initialization stuff:

```
UpdaterEngine.Init "http://www.monkeybreadsoftware.de/UpdaterTest/appcast.xml"
```

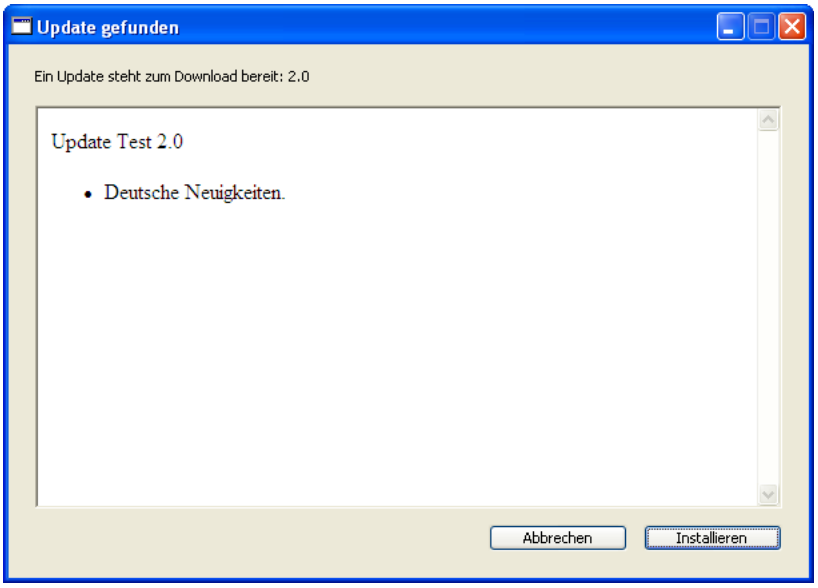
In this line you pass the URL for your appcast.xml file.

Next you may want to add some menu command or update button to your GUI and call there this line:

```
UpdaterEngine.CheckForUpdates
```

This is for manual updates showing the GUI.

Next you want to make sure that your application has a proper version number. The version number in app. ShortVersion must be parsable and


Update gefunden

Ein Update steht zum Download bereit: 2.0

Update Test 2.0

- Deutsche Neuigkeiten.

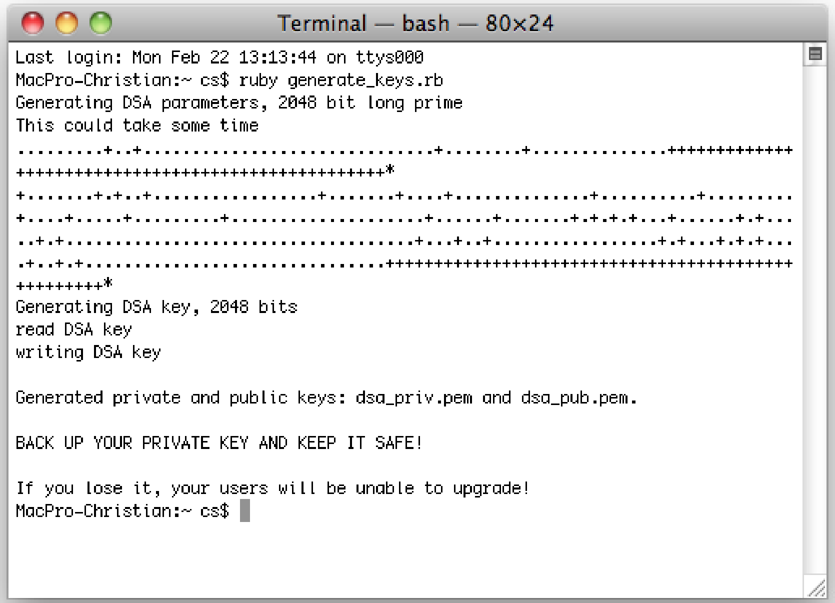
Figure 3: *German update dialog on Windows XP*

comparable to the newer versions you specify in your appcast.

For Sparkle Setup you can read the following webpage: http://sparkle.andymatuschak.org/documentation/pmwiki.php/Documentation/BasicSetup

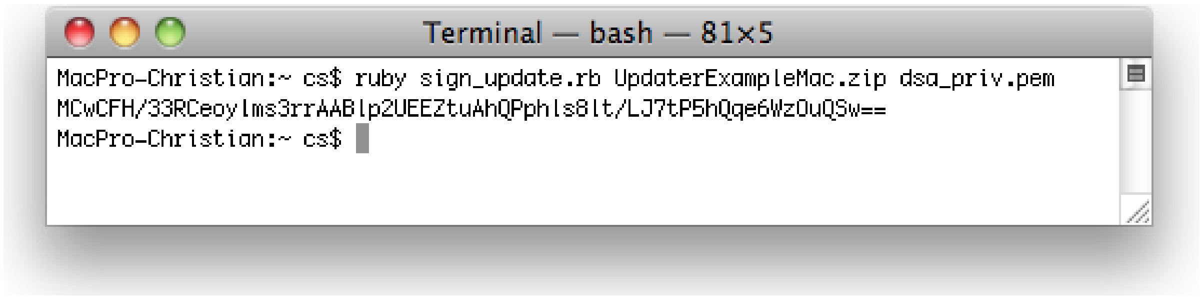
The important thing to do is to create a pair of keys to digitally sign your Mac updates. So launch the terminal, and type the command “ruby” followed with the path to the “generate_keys.rb” ruby script. This will generate you the files “dsa_priv.pem” and “dsa_pub.pem”.

For adding the keys, the plist URL and the Sparkle Framework to your own application, you should customize the “PatchMacApp.rbp” example project. Change the URL and the file paths so your own application is modified.



```
Last login: Mon Feb 22 13:13:44 on ttys000
MacPro-Christian:~ cs$ ruby generate_keys.rb
Generating DSA parameters, 2048 bit long prime
This could take some time
.........+..+..............................+........+..............+++++++++++++
++++++++++++++++++++++++++++++++++++++*
+.......+.+..+.................+.......+....+..............+..........+.........
+....+.....+.........+....................+......+.......+.+.+.+...+......+.+...
..+.+....................................+...+..+.................+.+...+.+.+...
.+..+.+...............................++++++++++++++++++++++++++++++++++++++++++
+++++++++*
Generating DSA key, 2048 bits
read DSA key
writing DSA key

Generated private and public keys: dsa_priv.pem and dsa_pub.pem.

BACK UP YOUR PRIVATE KEY AND KEEP IT SAFE!

If you lose it, your users will be unable to upgrade!
MacPro-Christian:~ cs$
```

Figure 4: *Creating keys in the terminal*



```
MacPro-Christian:~ cs$ ruby sign_update.rb UpdaterExampleMac.zip dsa_priv.pem
MCwCFH/33RCeoylms3rrAABlp2UEEZtuAhQPphls8lt/LJ7tP5hQqe6WzOuQSw==
MacPro-Christian:~ cs$
```

Figure 5: *Signing an update in the terminal*

Now you can patch your own application. After patching it should contain the Sparkle.framework in the MacOS folder, the dsa_pub.pem file in the Resources folder and the info.plist file should contain the SUFeedURL and SUPublicDSAKeyFile keys.

On Windows you create an installer for your application. For example you can use Inno Setup Compiler 5.3.4, which you can download here: http://www.jrsoftware.org/isinfo.php. The Updater Kit includes a sample setup file "Updater Example.iss" which you can modify for your installer. Basicly you copy the "Updater Example.iss" file and the application folder from your build folder to a Windows PC and let Inno Setup Compiler compress your application to a small exe file.

Next you want to create a folder on your webspace with all the files for the update engine. There you have a zip file with the Mac application and an exe file with the Windows application installer. Also you put there a description html file for each language and the appcast.xml file.

For your updates you need an appcast. We have an utility called AppCast Creator which helps you in creating this xml files. It can automatically calculate the MD5 Checksum and add the digital signature for your files. If you use our AppCast Creator project, you can skip the following instructions.

The example appcast.xml file looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:sparkle="http://www.andymatuschak.org/xml-namespaces/
sparkle">
    <channel>
        <title>Updates (English)</title>
      <link>http://www.monkeybreadsoftware.de/UpdaterTest/appcast.
xml</link>
            <description>Most recent changes with links to updates.</
description>
        <language>en</language>
        <item>
            <title>Version 2.0</title>
         <description>http://www.monkeybreadsoftware.de/UpdaterTest/
description.html</description>
            <pubDate>21 Feb 2010 20:00:00 +0000</pubDate>
      <enclosure url="http://www.monkeybreadsoftware.de/UpdaterTest/
UpdaterExampleMac.zip"
               sparkle:version="2.0"
               sparkle:dsaSignature="MCOCFQCRwRCxIp9N+X7h54AOxl
fSX/tcdwIUf2RBJl60guIir6xotfYYipysPno="
               length="2860194"
            urlWin32="http://www.monkeybreadsoftware.de/UpdaterTest/
UpdaterExampleWin.exe"
               lengthWin32="1145333"
               MD5Win32="193941CD70FD908CC9397D09597609E1"
               type="application/octet-stream" />
            </item>
        </channel>
    </rss>
```

As you see you have a channel for each language. Inside you have items, one for each version, but probably only one with the lastest version. In description you give the link to a webpage for the version details which is displayed in the updater window. Next you have the pubDate which is the date of the publication. In thenclosure URL finally is the actual update file. There is the URL for the Mac update zip file. Note that the name of the application in the zip file must be identical to the name of the old application. The version number of this update is given here and this is the version number we compare with the app.shortversion. Next you find the signature for the Mac update which you get with this terminal command:

```
ruby sign_update.rb UpdaterExampleMac.zip dsa_priv.pem
```

The output is the signature as Base64 encoded string which you copy to your appcast file. Note that you can get several signatures from this if you call it multiple times. All signatures are valid, but you need only one.

Next in the enclosure attributes is the urlWin32 which is the URL to download the Windows update. This must point to an executable file. Actually the name of the file in the URL does not matter as on disc it is written in an installer.exe file. Next value is the lengthWin32 field which ensures that the file is not too short. After that we have the MD5 checksum which is used to verify that the windows file contains the right content.

After you uploaded all files to your website, you can run your application in the older version and see whether it finds the update. For the upload we recommend that you upload the xml file as the last thing so no client tries to download the update while you still upload it.

Now you can test your application. For a simple test you can open your project and compile yourself a version with a little bit lower version number. So you see that the update is found.

Conclusion

We hope you have now an idea how the updater Kit works and how you use it. And you may see that it is not an easy task.

WEB RESOURCES:
http://www.monkeybreadsoftware.de/realbasic/UpdaterKit/

by Christian Schmitz

Easy Charts and Graphs (Part 1)

Using the ChartDirector Plugin

AT A GLANCE

RBD#7412

Target Reader:
Intermediate

Source Code:
Yes

RB Version Required:
2005+

Platform(s) Supported:
Mac OS X, Windows, Linux

Platform(s) Unsupported:
Mac Classic

About the Author:
Christian Schmitz is the creator of the Monkeybread Software REALbasic Plugins (including the ChartDirector plugin).

THE ChartDirector REALbasic plugin is a powerful chart component for creating professional looking charts. The list of charts is long and includes pie, bar, line, area, trending, box, scatter, bubbles, vectors, surface, pyramid, gantt and polor charts. Also there are meters and gauges. All those charts can be combined in a combo chart.

This article will show and explain small examples of creating charts and give you some code you can use as a starting point for your own projects. The 9.2 release of the MBS plugins comes with over 150 example projects for Chart Director. All those chart pictures of the examples are included, too, so you can just browse the pictures and find the example you need.

Test Driving ChartDirector

If you would like to try the ChartDirector plugin, you can download it at http://www.monkeybreadsoftware.de. Inside the download you find plugins files with file extension `rbx`. To install, simply copy the plugin files you need (in this case that's "MBS REALbasic ChartDirector Plugin.rbx") into REALbasic's plugins folder (located inside the folder where your copy of REALbasic is installed). Now (re)start REALbasic and if you type `dim c as CD` and the tab key in the code editor, you should see a long Autocomplete list of new classes beginning with "CD".

Without a license for the plugin, you run it in demo mode. There is a line drawn on every graph or chart saying that you have an unlicensed copy in use. And if you compile an application and run it, it will show a messagebox on every launch about the missing license. Once you order the license, this dialog will not show and the graphics have no warning. A license costs 199 Euro (around $270 USD) for commercial use and 99 Euro (around $135 USD) for academic use.

Using ChartDirector

We will begin by looking at pie charts. We'll follow that with pyramid charts and bar charts. Note that a bar chart is actually an XY chart with a bar layer. You can add several layers to each chart object, so we'll also show an example of a contour layer. Another example we will show is how to use line and spline layers together with the interline layer to color the space between lines and curves.

Another kind of graphic you can make with ChartDirector is a *meter*graph. I will show how to create a meter.

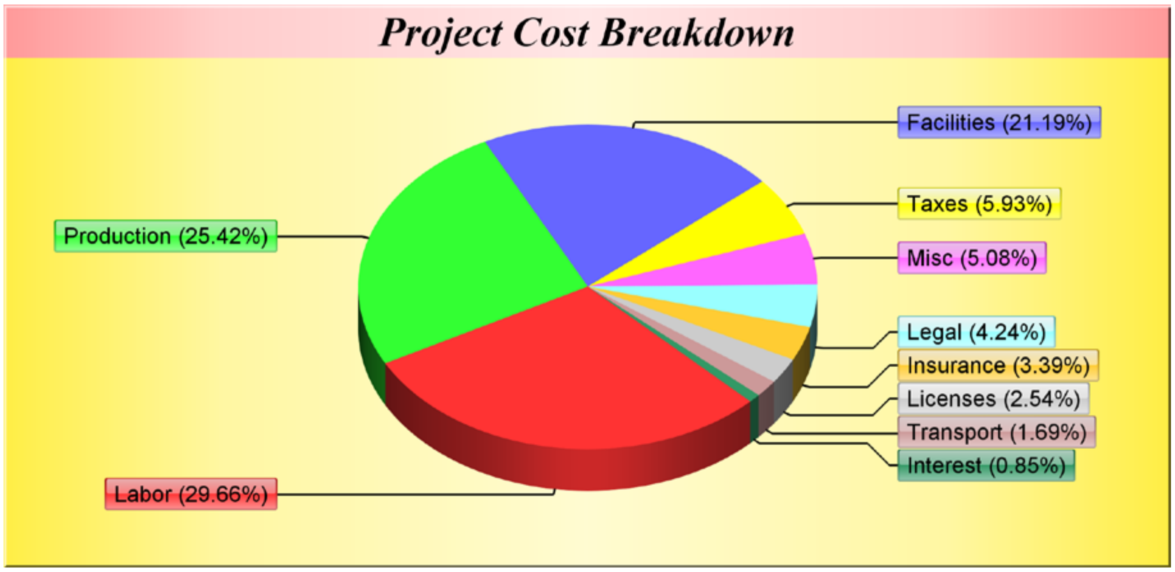


Figure 1: *A pie chart with 3D effects*

To display data with angles, you can use a polar chart. If your data can be viewed better in 3 dimensions, the surface chart may be right for you. Finally, everyone doing applications for finance reporting, should look on the finance chart at the end of this article.

Pie charts

ChartDirector can create wonderful pie and donut charts using the CDPieChartMBS class. First we need data to fill the chart. For this we create an array of doubles and fill it with values. In your application you can query data from a database or from a listbox, but in the end you need an array of doubles to send to the method:

```
dim data(-1) as double = array(35.0, 30, 25, 7, 6, 5, 4, 3, 2, 1)
```

Donut Chart

Figure 2: *This is a 3D donut with concave shadings*

Once we have the data, the next step is to get an array of strings for the labels. Labels are optional of course, but they make our chart look nicer. So we get an array of strings:

```
dim labels(-1) as string= array("Labor", "Production", "Facilities", "Taxes",
        "Misc", "Legal", "Insurance", "Licenses", "Transport", "Interest")
```

Now we create a PieChart object with a size of 560 x 270 pixels. For the background we use the goldColor function for a golden background. The fourth parameter is the edgeColor where we use -1 for the default color. With the fifth parameter we define a small raised effect.

```
dim c as new CDPieChartMBS(560, 270, CDPieChartMBS.goldColor, -1, 1)
```

With the addTitle method you can add a title to the chart. Like a lot of other methods this function returns a CDTextBoxMBS so you can specify more details on how the text is displayed. For the this example we use the Times font in bold italic at a size of 15 points. Once we've got the text box we set the background to a metal color in light red.

```
dim t as CDTextBoxMBS
t=c.addTitle("Project Cost Breakdown", "timesbi.ttf", 15)
t.setBackground(c.metalColor(&hff9999))
```

For a pie chart you need to decide whether you want a pie or a donut. So you can call setPieSize or call setDonutSize. Both need the position of the center at (280, 135) and the radius (110). For a donut we need the inner radius (55).

```
c.setPieSize(280, 135, 110)
// c.setDonutSize(280, 135, 110, 55)
```

You can set the chart to use a 3D effect. Here we use an effect with a height of 20 pixels. The set3D method takes more parameters, so take a look at the documentation. There you will find that you can optionally pass the angle for the 3D effect as well as the shadow mode to be used.

```
c.set3D(20)
```

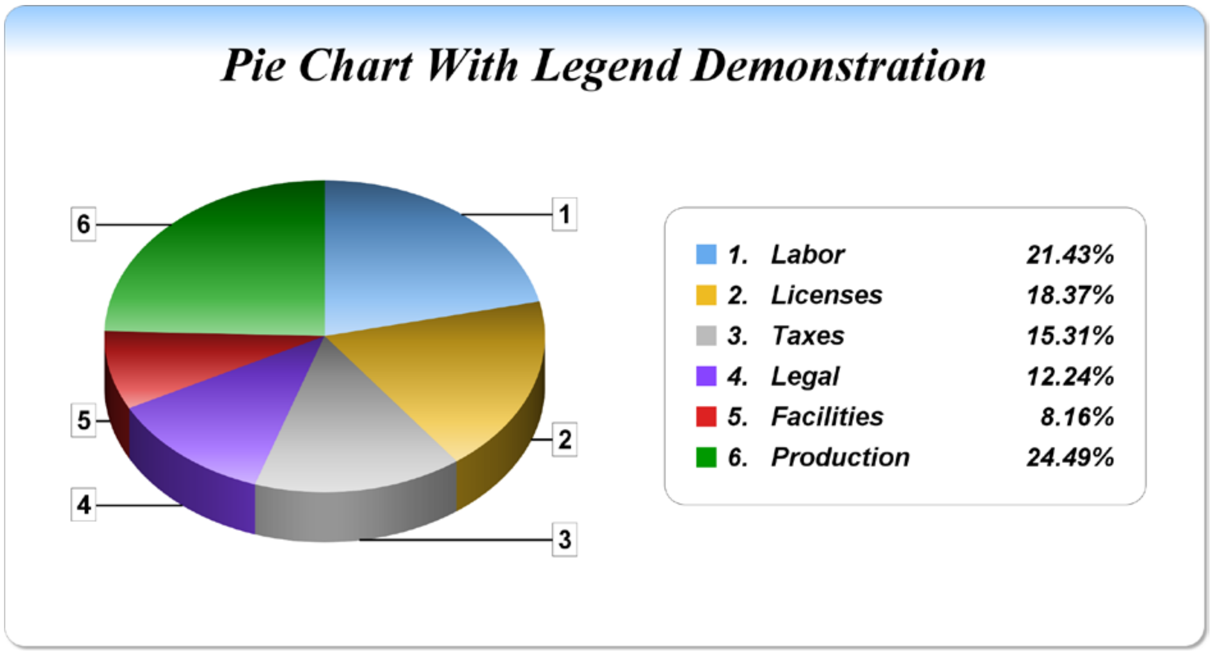


Figure 3: *A pie chart with legend.*

Another option we set in the label layout. For our pie chart this can be side or circle layout.

```
c.setLabelLayout(c.kSideLayout)
```

Now we define the style of the labels. As with the addtitle method we get a textbox back where we can change details. For setLabelStyle you can pass font name, font size, and font color, but we don't need that here and use the defaults. But here we do define the background color for the labels. The background color should be the same as the pie sector. For the edge color we pass kTransparent to hide the edges. Of course you can pass other colors and use REALbasic's RGB() function. For nicer labels we use the glassEffect as the raising effect for this labels. Finally, we want round corners for our labels with a five pixel radius.

```
t = c.setLabelStyle
t.setBackground(c.kSameAsMainColor, c.kTransparent, c.glassEffect())
t.setRoundedCorners(5)
```

Finally we define the border color for the sectors to be the same color as the sector fill color. And the line color for the line between sector and label should be black.

```
c.setLineColor(c.kSameAsMainColor, &h000000)
```

We set the start angle to 135 degrees to improve the layout as we have many small sectors at the end of the data array. Our array is sorted in descending order. With angle 135 we have the small sectors positioned near the horizontal axis, where the text labels have the least tendency to overlap. If the values are sorted in ascending order, you can use an angle of 45 degrees.

```
c.setStartAngle(135)
```

Now we pass the data and the label arrays to the pie chart.

```
c.setdata(data,labels)
```

Finally we render the chart to a picture to show it in the window backdrop. You can also save the chart to a picture file in various formats like PNG, JPEG, and TIFF. Version 5.0 of ChartDirector allows also to save in SVG, a vector format.

```
Backdrop=c.MakeChartPicture
```

(See Figures 1-3 for example pie charts.)

Pyramid charts

A pyramid chart shows a pyramid which is divided into sections. We start again with an array of double values.

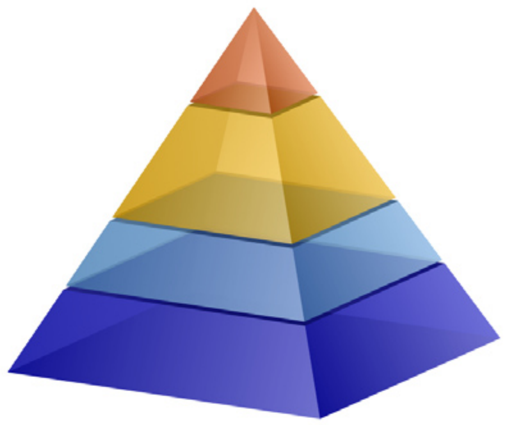


Figure 4: *A pyramid chart with 30 degree rotation and transparency*

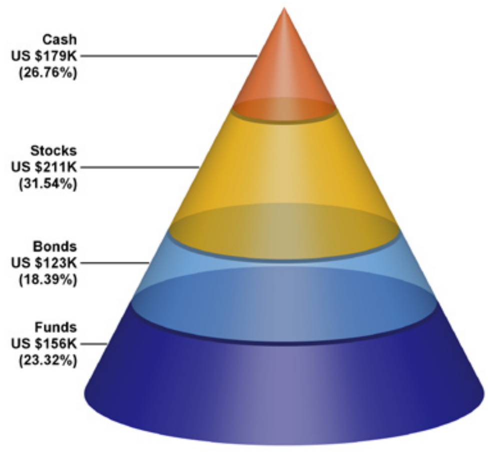


Figure 5: *A pyramid chart object can be configured to create a cone.*

```
dim data(-1) as double = array(156.0, 123, 211, 179)
```

For the colors we create an array and fill it with a color for each value. In ChartDirector colors are specified as integer values. Compared with REALbasic colors we can include an alpha value. So in the following color values we use &h40 for the alpha which makes the colors 25% transparent.

```
dim colors(-1) as integer
colors.Append &h400000cc
colors.Append &h4066aaee
colors.Append &h40ffbb00
colors.Append &h40ee6622
```

We create a new PyramidChart object with a size of 200 by 200 pixels. We use white (&hffffff) for the fill color and gray (&h888888) for the edgeColor.

```
dim c as new CDPyramidChartMBS(200, 200, &hffffff, &h888888)
```

As for the pie charts, we have to define the pyramid chart's position and size. We place it at the position (100,100) use a width of 60 pixels and a height of 120 pixels.

```
c.setPyramidSize(100, 100, 60, 120)
```

We add a title. This time we use call to ignore the function's result. In this case we use Arial Italic for the font, with 15 pixels as the size.

```
call c.addTitle("Rotation", "ariali.ttf", 15)
```

With setViewAngle you can set the elevation, the angle, and the twist. The default is 0 and we use 15 for the elevation and 30 for the angle.

```
c.setViewAngle(15, 30)
```

Now we pass the data to the pyramid chart and we set the colors for the data values to the color values we have in the colors array.

```
c.setData(data)
c.setColors(c.kDataColor, colors)
```

Between the layers of the pyramid we add a 1% gap.

```
c.setLayerGap(0.01)
```

As before we render the chart to show it in a window:

```
Backdrop=c.makeChartPicture
```

See Figures 4 and 5 for examples.

Bar charts

The next chart we'll do is a bar chart. Basically the XY Chart class can do any chart which has some x/y coordinate system (like line charts, bubble charts, and area charts).

```
dim data(-1) as double = array(450.0, 560, 630, 800, 1100, 1350, 1600,
        1950, 2300, 2700)
```

Now we create an array holding the labels for those data values.

```
dim labels(-1) as string =array( "1996", "1997", "1998", "1999", "2000",
        "2001", "2002", "2003", "2004", "2005")
```

Next we create a new XY Chart object with a size of 600 x 360 pixels:

```
dim c as new CDXYChartMBS(600, 360)
```

We add a title to the chart using Times Bold Italic font with a size of 18 pixels. As before we ignore the result (by using call).

```
call c.addTitle("Annual Revenue for Star Tech", "timesbi.ttf", 18)
```

For an XY chart you need to define the plot area. We set the position to (60, 40) and the size to 500 x 280 pixels. This leaves room for title and axis labels. We use the linearGradientColor function to create a color gradient from light blue (&heeeeff) to deep blue (&h0000cc) for the background. For the border and grid lines we use white (&hffffff).

```
call c.setPlotArea(60, 40, 500, 280, c.linearGradientColor(60, 40, 60, 280,
        &heeeeff, &h0000cc), -1, &hffffff, &hffffff)
```

Now we add a multi-color bar chart layer using the supplied data. To create a bar layer, we pass in data and colors. In this case we pass an empty array of colors, so ChartDirector will use the default color values. For the borders of these bars, we define a transparent color, so the border is not visible. But the raised effect is set using the softLighting method which creates a soft lighting effect for us with a light direction from the left.

```
dim b as CDBarLayerMBS
dim colors(-1) as integer
b=c.addBarLayer(data,colors)
b.setBorderColor(c.kTransparent, c.softLighting(c.kLeft))
```

Using the xAxis and yAxis methods you can access the axes of the chart. An XY chart has by default an x and a y axis. You can have, on the right side, a second Y axis. (You can, of course, add more axes if you need. The multiaxes example project shows you how to create a chart with four y axes.) In our case here we set the labels based on the labels string array:

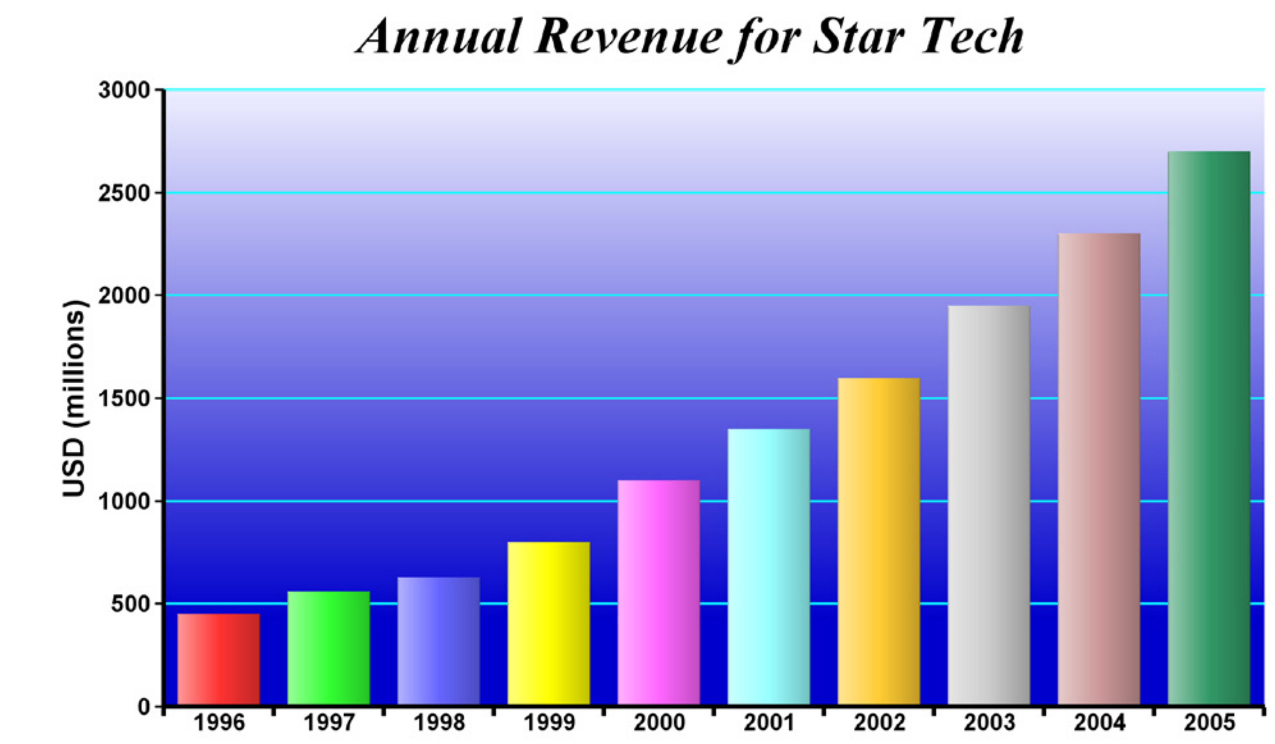


Figure 6: *An XY chart with bars.*

```
call c.xAxis.setLabels(labels)
```

On an axis you normally have ticks. By default the ticks a centered on the values, but here the values are shown as bars. So we set the tick offset to 0.5 so the ticks are between the bars.

```
c.xAxis.setTickOffset(0.5)
```

Here we add a title to the y axis:

```
call c.yAxis.setTitle("USD (millions)", "arialbd.ttf", 10)
```

For each axis we define the label style to be Arial Bold at 8 points:

```
call c.xAxis.setLabelStyle("arialbd.ttf", 8)
call c.yAxis.setLabelStyle("arialbd.ttf", 8)
```

The tick lines on the axes need a line width, so we set it to two pixels for both axes.

```
c.xAxis.setWidth(2)
c.yAxis.setWidth(2)
```

Finally, we render the chart.

```
Backdrop=c.makeChartPicture
```

Figure 6 shows this chart.

Contour layer in XY Chart

New in ChartDirector 5 is the contour layer, which we will use in this chart. First we need data to fill this chart. We define the x and y values using arrays to cover the x and y axes.

```
dim dataX(-1) as double = array(-10.0, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4,
        5, 6, 7, 8, 9, 10)
dim dataY(-1) as double = array(-10.0, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4,
        5, 6, 7, 8, 9, 10)
```

For the grid points we need z values. In this example, we will compute the values using the formula z = x * sin(y) + y * sin(x).

```
dim dataZ(21*21-1) as double
for yIndex as integer = 0 to UBound(datay)
  dim y as double = dataY(yIndex)
  for xIndex as integer = 0 to UBound(datay)
    dim x as double = dataX(xIndex)
    dataZ(yIndex * 21 + xIndex) = x * sin(y) + y * sin(x)
  next
next
```

Now we create a XY chart object with size of 600 x 500 pixels and we add a title to the chart.

```
dim c as new CDXYChartMBS(600, 500)
call c.addTitle("z = x * sin(y) + y * sin(x)     ", "arialbi.ttf", 15)
```

Next we add the plot area at (75, 40) with the size of 400 x 400 pixels. The background should be semi-transparent black (&h80000000) with dotted lines for both horizontal and vertical grid lines.

```
call c.setPlotArea(75, 40, 400, 400, -1, -1, -1, c.dashLineColor(&h80000000,
        CDXYChartMBS.kDotLine), -1)
```

The x-axis and y-axis should show a title with Arial font in bold italic. And the labels of this axes should use the Arial Bold font.

```
call c.xAxis.setTitle("X-Axis Title Place Holder", "arialbi.ttf", 12)
```

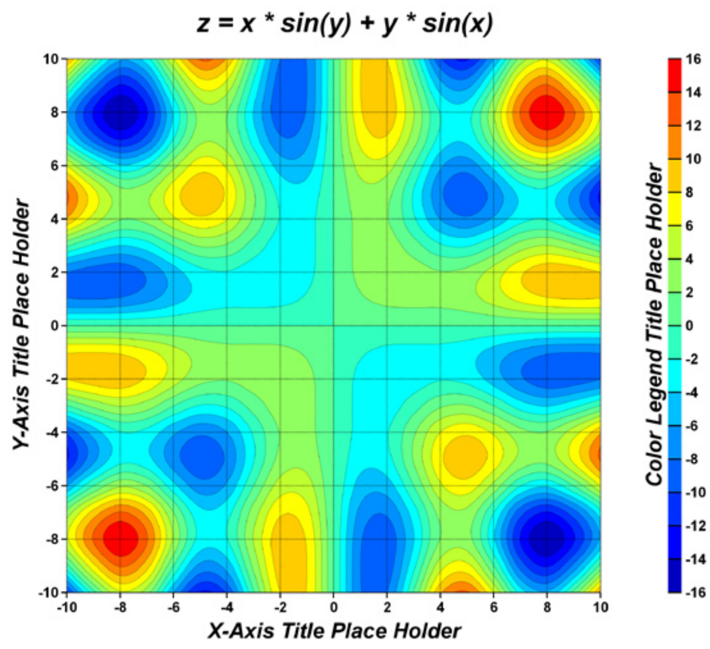


Figure 7: *A chart with a contour layer.*

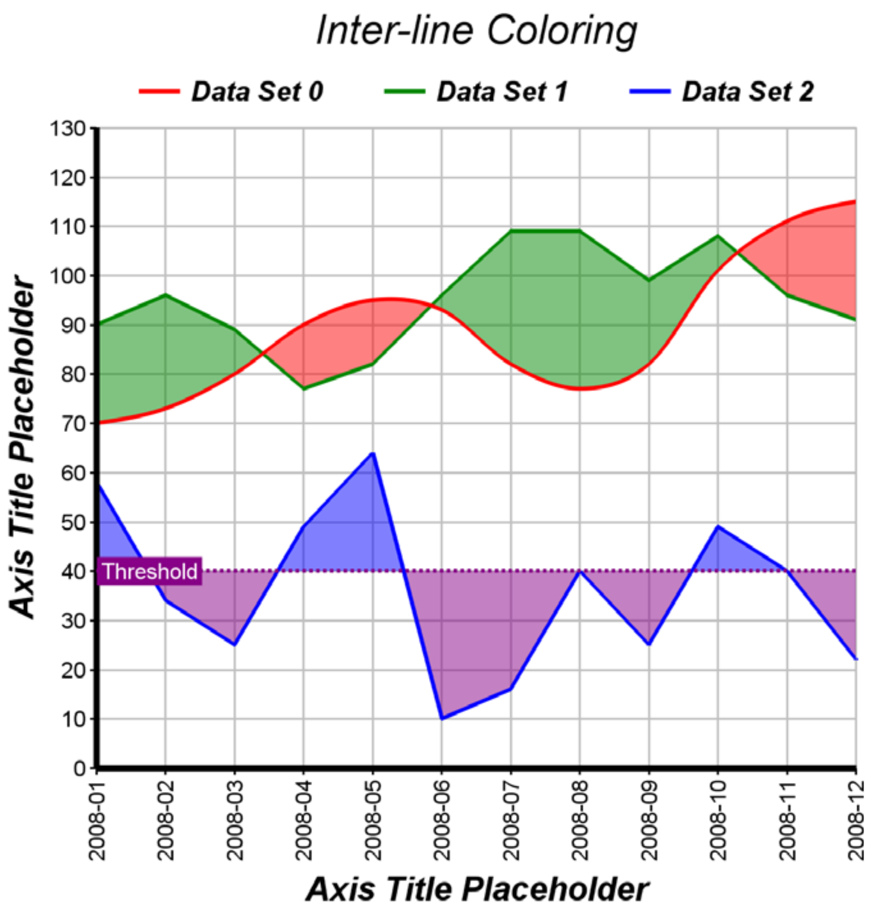


Figure 8: *A chart with a contour layer.*

```
call c.yAxis.setTitle("Y-Axis Title Place Holder",
"arialbi.ttf", 12)
call c.xAxis.setLabelStyle("arialbd.ttf")
call c.yAxis.setLabelStyle("arialbd.ttf")
```

We use automatic scaling of the axes and give 40 pixels as hint for the tick density. You can have an axis with linear, log, and/or automatic scaling.

```
c.yAxis.setTickDensity(40)
c.xAxis.setTickDensity(40)
```

Now we create the contour layer with our x, y, and z data arrays. The dataz array is filled with two-dimension data, but the array must have one dimension as the plugin can not handle two dimension arrays.

```
dim layer as CDContourLayerMBS = c.addConto
urLayer(dataX,dataY,dataZ)
```

The grid of the plot area is normally in the background, but you can move it to the front of any given layer. We move the grid lines in front of the contour layer.

```
c.getPlotArea.moveGridBefore(layer)
```

We create a color axis (for the legend) in which the top left corner is anchored at (505, 40). The length is set to 400 pixels and the labels should show on the right side.

```
dim cAxis as CDColorAxisMBS = layer.
setColorAxis(505, 40, CDXYChartMBS.kTopLeft,
400, CDXYChartMBS.kRight)
```

We add a title to the new axis:

```
call cAxis.setTitle("Color Legend Title Place
Holder", "arialbi.ttf", 12)
```

On the end we define the label style:

```
call cAxis.setLabelStyle("arialbd.ttf")
```

Finally we can display the chart in the window.

```
Backdrop=c.makeChartPicture
```

See Figure 7 for a picture of this chart.

Fun with line layers

Now let's create a more complex chart with three line layers showing how to fill the space between line layers. We'll also add a threshold to a line layer to color the line.

First we need some data to fill the layers. And for the x axis we need labels.

```
dim data0(-1) as double = array(70.0, 73, 80, 90,
95, 93, 82, 77, 82, 101, 111, 115)
```

```
dim data1(-1) as double = array(90.0, 96, 89, 77, 82, 96, 109, 109, 99, 108,
        96, 91)
dim data2(-1) as double = array(58.0, 34, 25, 49, 64, 10, 16, 40, 25, 49, 40,
        22)
dim labels(-1) as string = array("2008-01", "2008-02", "2008-03", "2008-04",
        "2008-05", "2008-06", "2008-07", "2008-08", "2008-09", "2008-10",
        "2008-11", "2008-12")
```

We create an XY chart object with a size of 450 by 450 pixels and we add a title with the Arial Italic font and a size of 15.

```
dim c as new CDXYChartMBS(450, 450)
dim title as CDTextBoxMBS = c.addTitle("Inter-line Coloring", "ariali.ttf", 15)
```

Next we add a legend box where the top-center is anchored to the horizontal center of the chart, just under the title. We use horizontal layout and transparent background and border. Legends can have different styles — we use the line style here.

```
dim legendBox as CDLegendBoxMBS = c.addLegend(c.getWidth / 2, title.
        getHeight, false, "arialbi.ttf", 10)
legendBox.setBackground(CDXYChartMBS.kTransparent, CDXYChartMBS.
        kTransparent)
legendBox.setAlignment(CDXYChartMBS.kTopCenter)
legendBox.setLineStyleKey
```

Tentatively we set the plotarea to (70, 65) and both width and height to 100 pixels less than the chart's size. We use light gray (&hc0c0c0) for the border, the horizontal, and the vertical grid lines.

```
dim plotArea as CDPlotAreaMBS = c.setPlotArea(70, 65, c.getWidth - 100,
        c.getHeight - 110, -1, -1, &hc0c0c0, &hc0c0c0, -1)
```

We add a title for the x and y axes. Next we increase the line width of the axes to three pixels.

```
call c.yAxis.setTitle("Axis Title Placeholder", "arialbi.ttf", 12)
call c.xAxis.setTitle("Axis Title Placeholder", "arialbi.ttf", 12)
c.xAxis.setWidth(3)
c.yAxis.setWidth(3)
```

Now we take the array of label strings and install them to the x axis.

```
call c.xAxis.setLabels(labels)
```

For the label style of the x axis we use 90 degrees to rotate the text. And for the font we use again Arial at 8 points with the default text color. By the way, these examples all use Arial and Times fonts because they are available across platforms. You can, of course, use any of the fonts you have on the computer you use.

```
call c.xAxis.setLabelStyle("Arial", 8, CDXYChartMBS.kTextColor, 90)
```

Now we add a spline layer to the chart and pass in the first data array. This line should have a red color. Also we pass in a title for this line. Later the legend box will use this title. This is the first line for the interline layer we will create later.

```
dim layer0 as CDSplineLayerMBS = c.addSplineLayer(data0, &hff0000,
        "Data Set 0")
layer0.setLineWidth(2)
```

Now we add a normal line to the chart. We use the second data array with a dark green color and the title "Data Set 1". A normal line has edges while the spline has curves so it may look nicer. Both lines get the line width of two pixels.

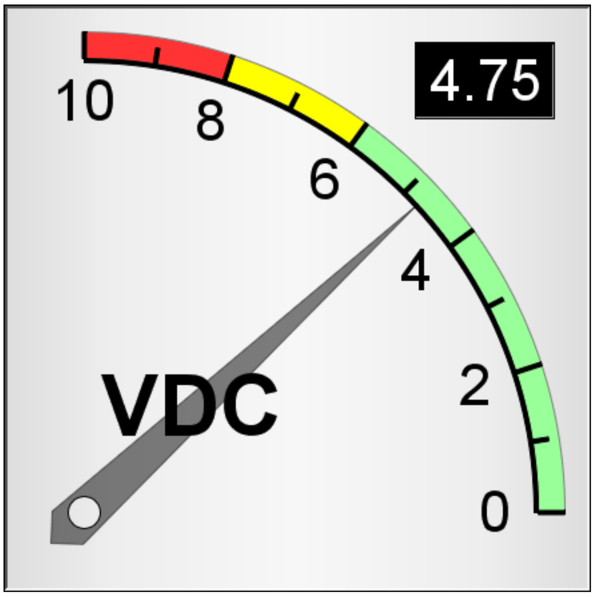


Figure 9: *An angular meter.*

```
dim layer1 as CDLineLayerMBS = c.addLineLayer(data1, &h008800, "Data
        Set 1")
layer1.setLineWidth(2)
```

We add an interline layer to color the region between the above spline curve and normal line. We use the semi-transparent red color (&h80ff000000) if the spline curve is higher than the normal line. Otherwise we use the semi-transparent green color (&h80008800). Note that on the color values the 80 on the front makes the color 50% transparent. The following hex digits are the same as in the &c constants in REALbasic. First two digits are red, second two digits green, and last two digits blue.

```
call c.addInterLineLayer(layer0.getLine, layer1.getLine, &h80ff0000,
        &h80008800)
```

Now we add a third line to the chart with normal style (no curves). This line should be blue and will use the third data set. Again we set a title for the legend and a line width of two.

```
dim layer2 as CDLineLayerMBS = c.addLineLayer(data2, &h0000ff, "Data
        Set 2")
layer2.setLineWidth(2)
```

To add a threshold to the chart, we add a horizontal mark line to the chart at y = 40 on the y axis. We label it "Threshold" and use the default line color first. But in the next lines we change that with more options. As the other lines we use a line width of two.

```
dim mark as CDMarkMBS = c.yAxis.addMark(40, -1, "Threshold")
mark.setLineWidth(2)
```

After we created the mark, we customize it. The mark line should be a purple (&h880088) dash line. For dash lines you can specify the dash widths, but we just take the defaults. We use white (&hffffff) for the mark label. We move the mark label to the left side of the mark and set the background to purple (&h880088).

```
mark.setMarkColor(c.dashLineColor(&h880088), &hffffff)
mark.setAlignment(CDXYChartMBS.kLeft)
```

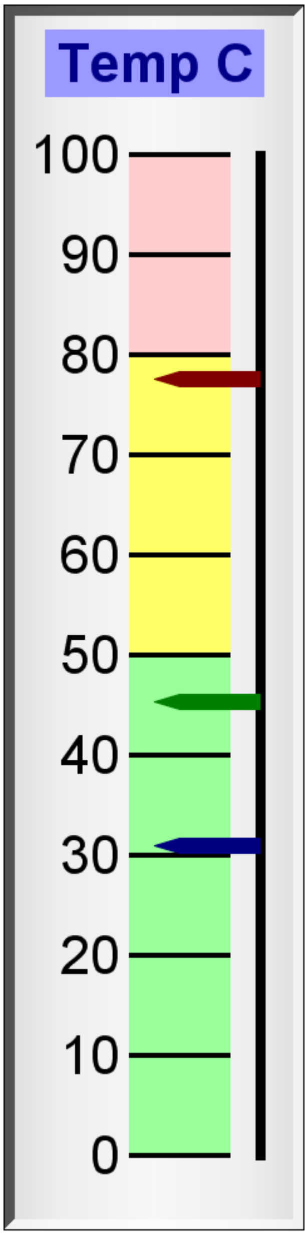


Figure 10: *This is an linear meter with three pointers.*

```
mark.setBackground(&h880088)
```

Now we add a second interline layer to color the region between the above normal line and mark line. We use the semi-transparent blue (&h800000ff) if the normal line is higher than the mark line, otherwise we use semi-transparent purple (&h80880088).

```
call c.addInterLineLayer(layer2.getLine, mark.
        getLine, &h800000ff, &h80880088)
```

Before we can define the final size of the plot area, we need to know the height of the legend. So we layout the legend:

```
call c.layoutLegend
```

After we have laid out the legend, we adjust the plot area size, such that the bounding box (inclusive of axes) is 10 pixels from the left edge, just under the legend box, 25 pixels from the right edge, and 10 pixels from the bottom edge. We have to keep the margins on the left and bottom to leave room for the axes titles.

```
c.packPlotArea(10, legendBox.getTopY +
        legendBox.getHeight, c.getWidth - 25, c.
        getHeight - 10)
```

After we determine the exact plot area position, we can adjust the legend box and the title positions so that they are centered relative to the plot area. By default they are centered relative to the chart.

```
legendBox.setPos(plotArea.getLeftX + (plotArea.
        getWidth - legendBox.getWidth) / 2,
        legendBox.getTopY)
title.setPos(plotArea.getLeftX + (plotArea.
        getWidth - title.getWidth) / 2, title.
        getTopY)
```

Finally we can render the picture to show it in a window.

```
Backdrop=c.makeChartPicture
```

Figure 8 shows what this chart looks like.

Meters and gauges

ChartDirector can render meters and gauges. Meters can be angular or linear. In the next lines we show you how to use the CDAngularMeterMBS class. In the example the value is a constant, but you can easily replace that with a value taken from a slider to make it dynamically change. We will start our example with a value of 4.75.

```
const value = 4.75
```

Now we create the AugularMeter object with a size of 110 by 110 pixels. We color the background with silver and add a black two pixels 3D depressed border.

```
dim m as new CDAngularMeterMBS(110, 110,
        CDBaseChartMBS.silverColor,
        &h000000, -2)
```

Next we define the size of the meter. The meter is centered at bottom left corner (15, 95) with a radius of 85 pixels. And for the range, we specify 90 to zero degrees.

```
m.setMeter(15, 95, 85, 90, 0)
```

We add a label to the meter centered at (35, 75).

```
call m.addText(35, 75, "VDC", "arialbd.ttf", 12,
        CDBaseChartMBS.kTextColor,
        CDBaseChartMBS.kCenter)
```

We add a text box to show the value at top right corner (103, 7). On the CDTextBoxMBS object returned we change the background options:

```
m.addText(103, 7, m.formatValue(value, "2"),
        "arial.ttf", 8, &hffffff,CDBaseChartMBS.
        kTopRight).setBackground(0, 0, -1)
```

The meter scale should be from zero to ten, with a major tick every two units, and minor tick every one unit:

```
m.setScale(0, 10, 2, 1)
```

We add three zones to the meter. From zero to six the meter should be green (&h99ff99), from six to eight it should be yellow (&hffff00) and from 8 to 10 it should be red (&hff3333).

```
m.addZone(0, 6, &h99ff99, &h808080)
m.addZone(6, 8, &hffff00, &h808080)
m.addZone(8, 10, &hff3333, &h808080)
```

On the end we add a semi-transparent black (&h80000000) pointer at the specified value.

```
call m.addPointer(value, &h80000000)
```

Finally we can render this to the window backdrop.

```
w.Backdrop=m.makeChartPicture
```

Figures 9 and 10 show sample meters.

Conclusion

I hope you have enjoyed all these charting examples and start adding charts to your applications. Next issue I'll finish with a few more examples.

Good looking charts and graphs can be a positive selling point of your application.

WEB RESOURCES:

Chart Director Plugin (free demo):

http://www.monkeybreadsoftware.de/realbasic/plugin-chartdirector.shtml

http://www.advsofteng.com/

by Christian Schmitz

Easy Charts and Graphs Part 2

Using the ChartDirector Plugin

AT A GLANCE

RBD#7512

Target Reader:
Intermediate

Source Code:
Yes

RB Version Required:
2005+

Platform(s) Supported:
Mac OS X, Windows, Linux

Platform(s) Unsupported:
Mac Classic

About the Author:
Christian Schmitz is the creator of the Monkeybread Software REALbasic Plugins (including the ChartDirector plugin).

THE ChartDirector REALbasic plugin is a powerful chart component for creating professional looking charts. In Part 1 we began with some simple examples of using the plugin, so in this part we'll finish with a few more.

Polar charts

You can create a lot of different polar charts with ChartDirector. Here I will show how to create a polar chart with bubbles. Basically we draw lines on the polar chart, but these are invisible lines where we only show the data points with different sizes. First we start with data for the values, the angles, and the size of each data point. As we want to show bubbles in two colors, we use two layers with each having its own fill color.

```
dim data0(-1) as double = array(6, 12.5, 18.2, 15.0)
dim angles0(-1) as double = array(45, 96, 169, 258.0)
dim size0(-1) as double = array(41, 105, 12, 20.0)
dim data1(-1) as double = array(18, 16, 11, 14.0)
dim angles1(-1) as double = array(30, 210, 240, 310.0)
dim size1(-1) as double = array(30, 45, 12, 90.0)
```

We create a polar chart with a size of 460 x 460 pixels.

```
dim c as new CDPolarChartMBS(460, 460)
```

We add a title to the chart at the top left corner using the Arial font in bold and italic style. Also we have some inline format commands to underline the text.

```
call c.addTitle(CDPolarChartMBS.kTopLeft, "<*underline=2*>EM Field Strength", "arialbi.ttf", 15)
```

Now we center the plot area with 230 by 240 pixels and a radius of 180 pixels.

```
c.setPlotArea(230, 240, 180)
```

For the background we use alternative light gray and dark gray circular background color.

```
c.setPlotAreaBg(&hdddddd, &heeeeee)
```

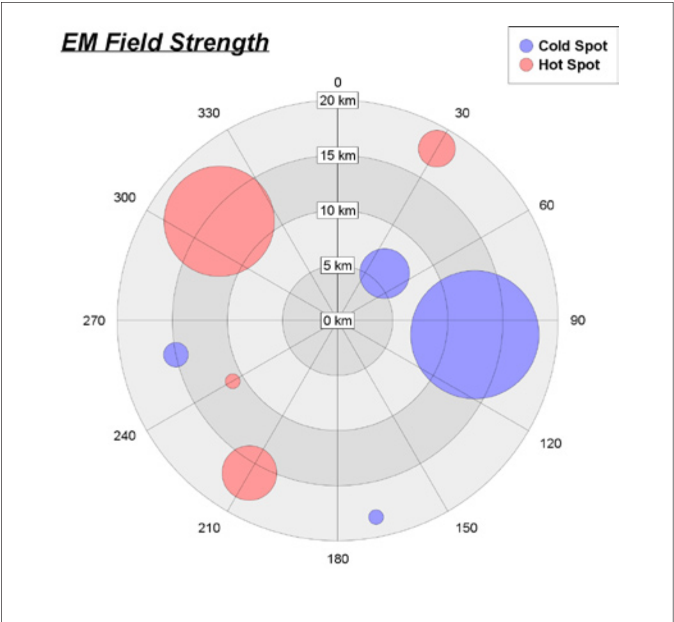


Figure 1: *A polar chart showing bubbles.*

You can decide between using circular grids or polygon grids, and whether the grid lines are on top of the polar plot area or are at the back. We decide to use the circular grid (false for the first parameter). For the second parameter we take the default which is to have the grid on the top.

```
c.setGridStyle(false)
```

Next we add a legend box at the top right corner of the chart. As you see we use the CDLegendBoxMBS object returned by addLegend directly to call a method on it to set the alignment.

```
c.addLegend(459, 0, true, "arialbd.ttf", 9).setAlignment(CDPolarChartMBS.
          kTopRight)
```

For our angular axis we ask for scaling linear between 0 and 360, with a spoke every 30 units.

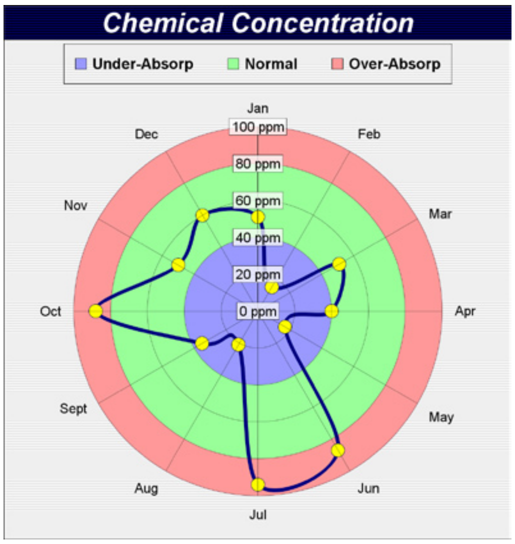


Figure 2: *Another polar chart showing areas and a spline layer with circles on the data points.*

```
c.angularAxis.setLinearScale(0, 360, 30)
```

You can format labels. In this case, we ask to add kilometer as the unit behind the label value.

```
c.radialAxis.setLabelFormat("{value} km")
```

Now we add a blue (&h9999ff) line layer to the chart using the values in the data0 array and the angle0 array. The string "Cold Spot" is the label for the legend. As you see you need to call setAngles on the polar line layer to specify the angles as the addLineLayer method does not take that values.

```
dim layer0 as CDPolarLineLayerMBS
layer0 = c.addLineLayer(data0, &h9999ff, "Cold Spot")
layer0.setAngles(angles0)
```

As we want to show only the bubbles, we disable the line by setting its width to zero.

```
layer0.setLineWidth(0)
```

For drawing, we use circular data point symbols. We pass 11 for the default size, but that size won't come to effect as the next line overwrites this. We use the size0 array to create the bubble chart effect.

```
layer0.setDataSymbol(CDPolarChartMBS.kCircleSymbol, 11)
layer0.setSymbolScale(size0)
```

Now we do the same again with a new polar line layer. This layer should draw in red (&hff9999) and the data comes from the data1 and angles1 arrays.

```
dim layer1 as CDPolarLineLayerMBS
layer1 = c.addLineLayer(data1,&hff9999, "Hot Spot")
layer1.setAngles(angles1)
```

We disable the line drawing and set the symbols like in the first polar line.

```
layer1.setLineWidth(0)
layer1.setDataSymbol(CDPolarChartMBS.kCircleSymbol, 11)
layer1.setSymbolScale(size1)
```

On the end we can show the chart.

```
Backdrop=c.makeChartPicture
```

See Figures 1 and 2 for these polar charts.

Surface charts

A surface chart is a nice way to display data in 3D. So we start with data arrays. First we create an array for the x axis where we add time values. The chartTime shared method in the CDBaseChartMBS returns you a time value for a date or optionally for a date and time.

```
dim dataX(-1) as double
dataX.Append CDBaseChartMBS.chartTime(2008, 9, 1)
dataX.Append CDBaseChartMBS.chartTime(2008, 9, 2)
dataX.Append CDBaseChartMBS.chartTime(2008, 9, 3)
dataX.Append CDBaseChartMBS.chartTime(2008, 9, 4)
dataX.Append CDBaseChartMBS.chartTime(2008, 9, 5)
```

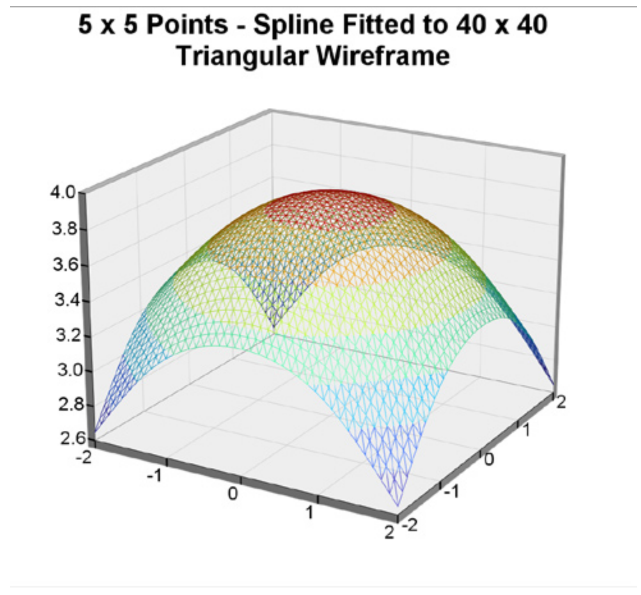


Figure 3: *A surface chart with wire frame.*

```
dataX.Append CDBaseChartMBS.chartTime(2008, 9, 6)
```

For the y axis we use three labels.

```
dim dataY(-1) as string = array("Low", "Medium", "High")
```

For the actual chart data we use three data series which we will join later.

```
dim lowData(-1) as double = array(24.0, 38, 33, 25, 28, 36)
dim mediumData(-1) as double = array(49.0, 42, 34, 47, 53, 50)
dim highData(-1) as double = array(44.0, 51, 38, 33, 47, 42)
```

Now we can create the surface chart object with a size of 760 x 500 pixels. Than we add a title.

```
dim c as new CDSurfaceChartMBS(760, 500)
call c.addTitle("Surface Chart Axis Types", "Arial", 18)
```

We set the center of the plot region at (385, 240), set the width to 480 pixels, the depth to 240 pixels, and the height to 240 pixels.

```
c.setPlotRegion(385, 240, 480, 240, 240)
```

For a nicer view we set the elevation and rotation angles to 30 and -10 degrees. As a third parameter you could set the twist to some other value than the default value zero.

```
c.setViewAngle(30, -10)
```

Now it is time to join the data. As the plugin needs a one dimensional array, we have to create one data array and fill it with the values from the three data sets.

```
dim dataz(-1) as double
dim count as integer
count=UBound(lowData)
for i as integer=0 to count
```

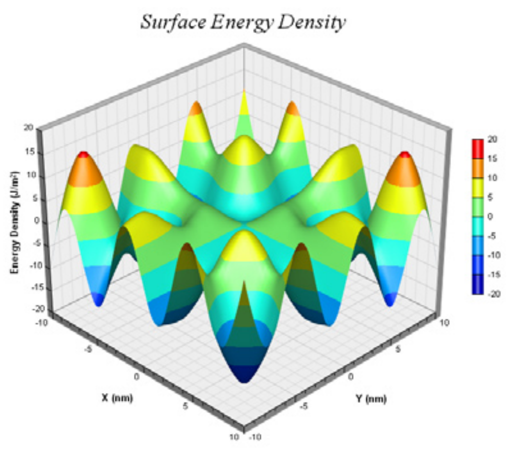


Figure 4: *A surface chart.*

```
 dataz.Append lowData(i)
next
count=UBound(mediumData)
for i as integer=0 to count
 dataz.Append mediumData(i)
next
count=UBound(highData)
for i as integer=0 to count
 dataz.Append highData(i)
next
```

We set the data used to plot the chart. As the y-data are text strings (enumerated), we will use an empty array for the y-coordinates. For the z data series, they are just the concatenation of the individual data series.

```
dim data2(-1) as double
c.setData(dataX,data2,dataz)
```

Now we apply our labels array to the y axis

```
call c.yAxis.setLabels(dataY)
```

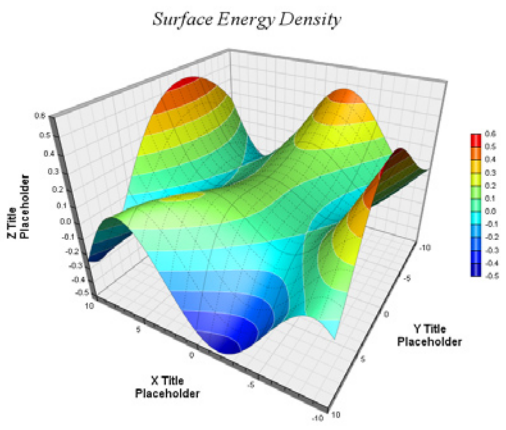


Figure 5: *A surface chart.*

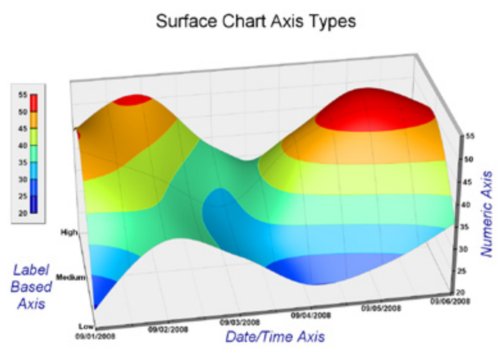


Figure 6: *A surface chart showing bubbles.*

The x-axis tick density should be 75 pixels. The ChartDirector auto-scaling feature will use this value as a guideline when putting ticks on the x-axis.

```
c.xAxis.setTickDensity(75)
```

We set the spline interpolate data to a 80 x 40 grid for a smooth surface:

```
c.setInterpolation(80, 40)
```

The surface has grid lines which we draw with a semi-transparent black color (&hcc000000).

```
c.setSurfaceAxisGrid(&hcc000000)
```

We set the contour lines to the same color as the fill color at the contour level.

```
call c.setContourColor(c.kSameAsMainColor)
```

For the legend we add a color axis. We anchored the top right corner at (95, 100) and we set the length to 160 pixels and the labels on the left side.

```
dim cAxis as CDColorAxisMBS = c.setColorAxis(95, 100, c.kTopRight, 160, c.kLeft)
```

The axis can have a bounding box. We use one with light gray (&heeeeee) background and grey (&h888888) border.

```
cAxis.setBoundingBox(&heeeeee, &h888888)
```

We set label style to Arial bold for all axes.

```
call c.xAxis.setLabelStyle("arialbd.ttf")
call c.yAxis.setLabelStyle("arialbd.ttf")
call c.zAxis.setLabelStyle("arialbd.ttf")
call c.colorAxis.setLabelStyle("arialbd.ttf")
```

We set the x, y and z axis titles using deep blue (&h000088) and use a Arial Italic font. As you see labels can have multiple lines which are divided by `chr(10)` (which you can access in `EndOfLine.unix`).

```
call c.xAxis.setTitle("Date/Time Axis", "ariali.ttf", 15, &h000088)
call c.yAxis.setTitle("Label"+EndOfLine.unix+"Based"+EndOfLine.unix+"Axis",
        "ariali.ttf", 15, &h000088)
call c.zAxis.setTitle("Numeric Axis", "ariali.ttf", 15, &h000088)
```

As before we render the graphics to a picture which we display in a window.

```
Backdrop=c.makechartPicture
```

Figures 3 through 7 show several variations of surface charts.

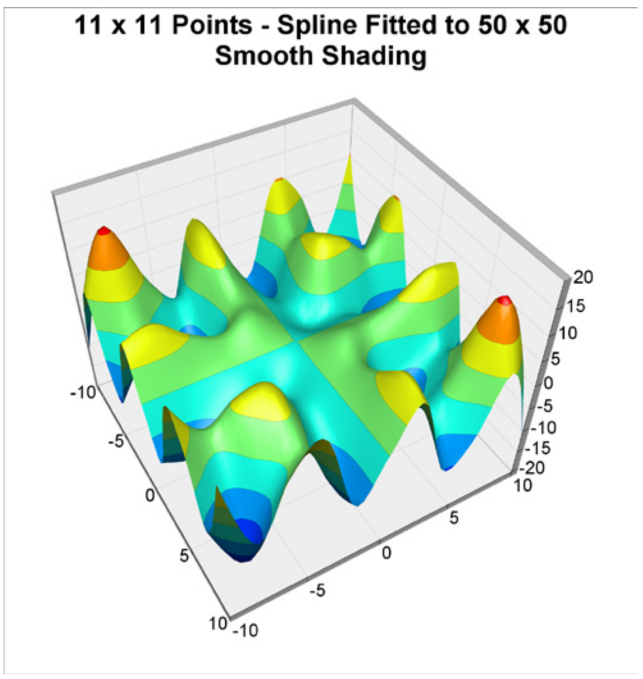


Figure 7: *A surface chart with shading.*

Financial charts

There is a finance chart class which is built upon all the other classes. Thus everything in this class can be done with your own code and that this makes this class extremely customizable.

For this demo we will create example data using the `CDRanTableMBS` class. We will create data for 100 days. To compute moving averages starting from the first day, we need to get extra data points before the first day.

```
const noOfDays = 100
const extraDays = 30
```

In this exammple, we use a random number generator to simulate the data. We set up the random table to create six columns with each `noOfDays + extraDays` rows, using nine as the seed.

```
dim rantable as new CDRanTableMBS(9, 6, noOfDays + extraDays)
```

We set the first column to be the timestamp, starting from Sep 4, 2002, with each row representing one day, and counting weekdays only (skipping Saturday and Sunday). There are 86,400 seconds one day so that's our increment between rows.

```
rantable.setDateCol(0, CDFinanceChartMBS.chartTime(2002, 9, 4), 86400, true)
```

Next we set the second, third, forth, and fifth columns to be high, low, open, and close data. The open value starts from 100, and the daily change is random from negative five to five.

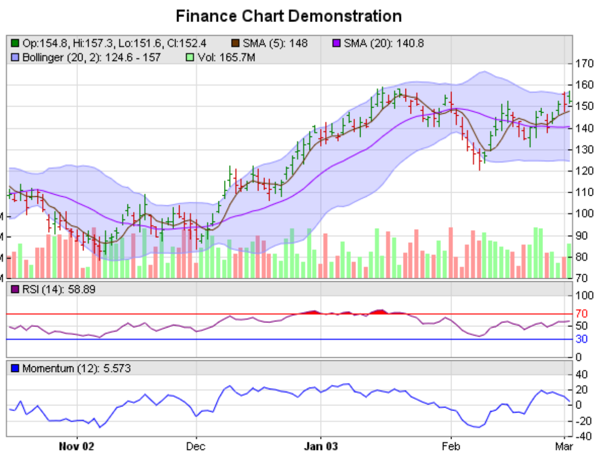


Figure 8: *One of the finance charts you can create.*

```
rantable.setHLOCCols(1, 100, -5, 5)
```

Set the sixth column as the vol data from five to 25 million. As you see this can be the chart needed for a stock exchange application.

```
rantable.setCol(5, 50000000, 250000000)
```

After our random data is created, we copy the data from the CDRanTableMBS object to REALbasic arrays.

```
dim timeStamps(-1) as double = rantable.getCol(0).Values
dim highData(-1) as double = rantable.getCol(1).Values
dim lowData(-1) as double = rantable.getCol(2).Values
dim openData(-1) as double = rantable.getCol(3).Values
dim closeData(-1) as double = rantable.getCol(4).Values
dim volData(-1) as double = rantable.getCol(5).Values
```

Now it is time to create a FinanceChart object with a width of 640 pixels. The height of the chart will be determinated later based on what parts we add. We start by adding a title.

```
dim c as new CDFinanceChartMBS(640)
call c.addTitle("Finance Chart Demonstration")
```

We set the data for this finance chart. This data is used in the methods below.

```
call c.setData(timeStamps, highData, lowData, openData, closeData, volData,
        extraDays)
```

We add the main chart with 240 pixels in height.

```
call c.addMainChart(240)
```

Next we add a five-period simple moving average to the main chart. For this we use a brown color (&h663300).

```
call c.addSimpleMovingAvg(5, &h663300)
```

Now we add a 20-period simple moving average to the main chart, using purple color (&h9900ff).

```
call c.addSimpleMovingAvg(20, &h9900ff)
```

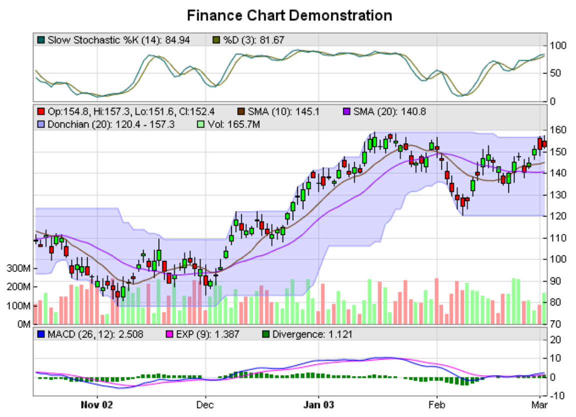


Figure 9: *Another finance chart with different options.*

We add an HLOC (High, Low, Open, and Close) symbols to the main chart, using green/red for up/down days.

```
call c.addHLOC(&h008000, &hcc0000)
```

For our chart, we add a 20-day bollinger band to the main chart, using light blue (&h9999ff) as the border and semi-transparent blue (&hC06666ff) as the fill color.

```
call c.addBollingerBand(20, 2, &h9999ff, &hc06666ff)
```

We add a 75 pixels volume bars sub-chart to the bottom of the main chart, using green/red/grey for up/down/flat days.

```
call c.addVolBars(75, &h99ff99, &hff9999, &h808080)
```

Next we append a 14-day RSI indicator chart (75 pixels high) after the main chart. The main RSI line should be purple (&h800080). We set the threshold region to +/- 20 (that is RSI = 50 +/- 25 pixels). The upper/lower threshold regions will be filled with red (&hff0000) and blue (&h0000ff).

```
call c.addRSI(75, 14, &h800080, 20, &hff0000, &h0000ff)
```

On the end, we append a 12-day momentum indicator chart (75 pixels high) using blue (&h0000ff) color.

```
call c.addMomentum(75, 12, &h0000ff)
```

Finally, we can show this chart.

```
Backdrop=c.makeChartPicture
```

Figures 8 and 9 show examples of the financial charts.

Conclusion

I hope you have enjoyed all these charting examples and start adding charts to your applications. Good looking charts and graphs can be a positive selling point of your application.

WEB RESOURCES:

http://www.monkeybreadsoftware.de/realbasic/plugin-chartdirector.shtml

http://www.advsofteng.com/

by Christian Schmitz

REALbasic Plugins

Getting started with the Plugin SDK

AT A GLANCE

RBD#8109

Target Reader:
Intermediate

Source Code:
No

RB Version Required:
2005+

Platform(s) Supported:
Mac OS X, Windows, Linux

Platform(s) Unsupported:
Mac Classic

About the Author:
Christian Schmitz is the creator of the Monkeybread Software REALbasic Plugins.

GETTING started with the REALBasic plugin SDK can be difficult. The following article shows you how to get started on Mac OS X, Windows, and Linux. It is based on the presentation from the REALbasic Summit 2009 in Boulder, Colorado.

About plugins

If you compare REALbasic with a car, you can see plug-ins as replacement, extension, or tuning parts. So while REALbasic comes with a lot of built-in stuff, you always find things which are missing. Or you have a function in REALbasic which doesn't work for you and you need a replacement function. Or you need a tuning part to perform a task faster. For example, a picture effect which performs better in optimized C code than in REALbasic code. And in that case you can take a plugin off the shelf or make your own.

There are a few reasons you may have to write a plugin. For example you can embed an existing static library or existing code in C/C++ or Objective-C into a plug-in. Technically you can even embed code from other platforms into your plugin. For example you can compile and link into your plugin code written in Fortran. Another reason to write a plugin is to call operating system functions which are difficult to call from REALbasic. Some calls to COM interfaces or C++ classes are much easier from a plug-in. A third reason may be that you want to use symmetric multi-threading. This is easier to setup and run in a plug-in than in REALbasic code with declares or existing plug-ins.

Requirements

To compile a plugin you need a C compiler. We'll later check the different compilers you find on different platforms. Technically you could use any other compiler which creates a shared library with a specific entry point, but a C/C++ compiler is the easiest as the SDK has C/C++ code. Once you download the SDK and get started you need a lot of patience. The plug-in documentation is limited and for a few functions you only find documentation in an email in the plugin mailing list.

Mac OS X

For Mac OS X you can use different compilers. GCC and LLVM come for free with Xcode on Mac OS X. For a better optimization you can of course pick other compilers like the Intel compilers. If you like you can use the Xcode IDE from Apple.

In the long run, and if you have several plug-ins, you will certainly switch to some scripts running in the terminal. Whatever you use doesn't matter, but you need to compile a dylib shared library in bundle style. You compile a 32 bit universal binary library for the Mac OS X Universal target in REALbasic and if you support older REALbasic versions a second PPC-only library for the Mac OS X target.

Until version 2009r4REALbasic was based on the Carbon application framework, but future versions will use the Cocoa framework. So you will want to compile all GUI code twice. Once for Carbon and one for Cocoa. Non-GUI code should still be linked twice, one time for Cocoa without the Carbon framework.

Windows

On Windows you can go with the free GCC compiler. You may want to use the Microsoft Visual C compilers. Of course it does not really matter which C compiler as long as it creates a native Windows DLL. Even Codewarrior 8 can do this. And it has the great advantage that it runs on Mac. So Mac users can use it to compile code on Mac for both Mac and Windows. But whatever DLL you compile, make sure it is native code and not some byte code from the .net world.

Linux

For Linux you can use the compilers coming with your favorite Linux distribution. If you don't have one preinstalled, it is easy to install the packages for gcc or llvm. Those are two open source compilers you can use for free. For your distribution, make sure you install the 32 bit version so you have it easier and all libraries already installed in 32 bit. Because REALbasic expects a 32 bit ELF shared library compiler for x86 CPUs.

One important thing on Linux is to keep your plugin linking to older library versions. Each Linux distribution has different versions of the libraries installed. And your existing user base may use older distributions with even older libraries. So you may want to install packages for older C/C++ runtime libraries, and you may prefer to link to libstdc++.so.5 instead of libstdc++.so.6 or libstdc++.so.7. Each newer version introduces new functions which may not be available in older versions. So if you depend on the older versions, your plugin will work with older and newer versions.

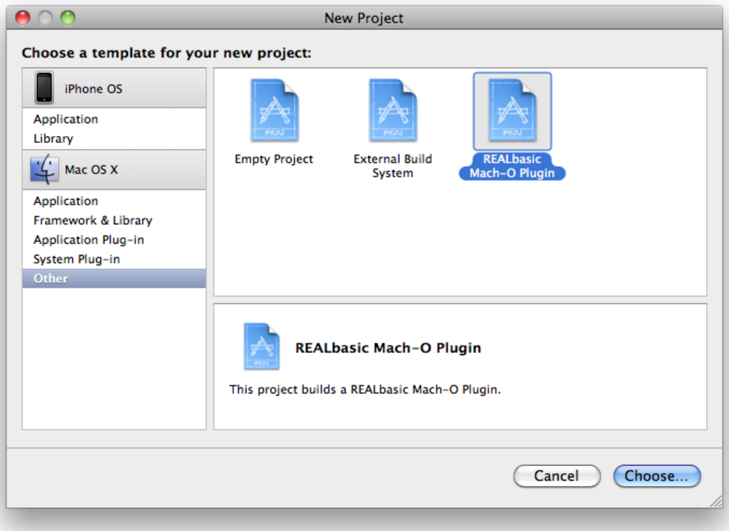


Figure 1: *The new project dialog in Xcode with the REALbasic template*

REALbasic versions

What REALbasic version do you want to support at a minimum? Depending on which version you choose as minimum you are limited to what you can use. REALbasic 2005r4 introduces console flags. So you can mark which functions, classes, and modules are supported for console applications. If you don't set the console flag the given item will not be visible in developing a console application. And console safe declarations are not allowed to use the picture, the graphics, or the window class. But those console flags will be no problem for older versions of REALbasic as they are ignored there.

For REALbasic 2009r3 and newer you need to provide a universal Mac OS X library. For older REALbasic versions you need to provide a PPC library, too. With REALbasic 2006r4 new 64 bit data types are added. If you want to use these new data types any older REALbasic versions will report an error

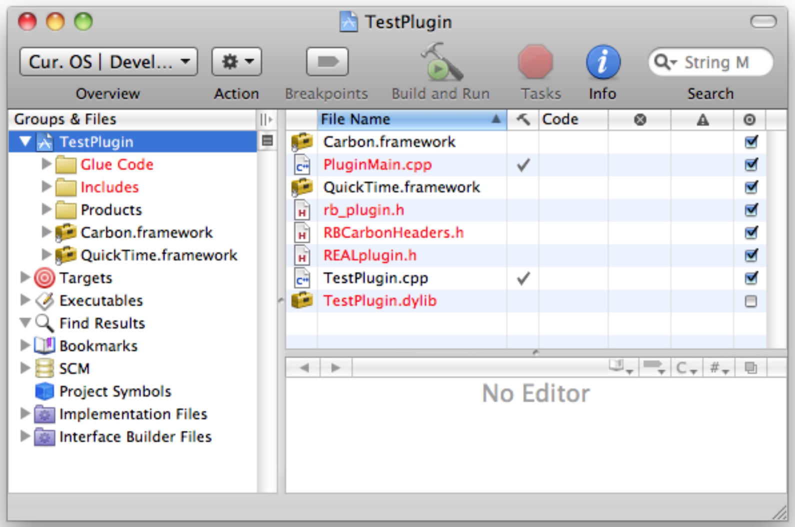
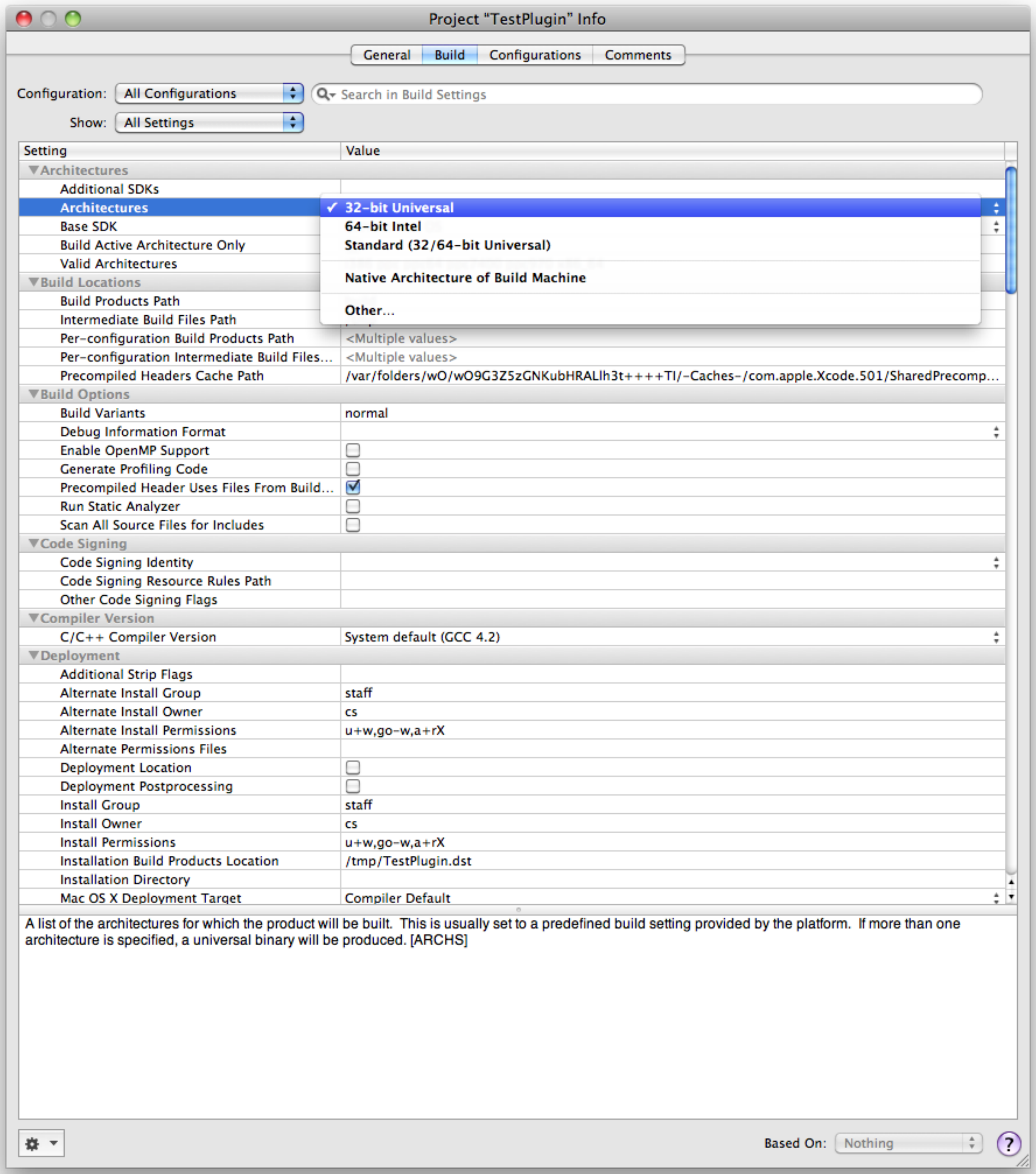


Figure 2: *The new project in Xcode*



Configuration: All Configurations

Show: All Settings

Setting	Value
▼Architectures	
Additional SDKs	
Architectures	✓ 32-bit Universal
Base SDK	
Build Active Architecture Only	
Valid Architectures	
▼Build Locations	
Build Products Path	
Intermediate Build Files Path	
Per-configuration Build Products Path	<Multiple values>
Per-configuration Intermediate Build Files...	<Multiple values>
Precompiled Headers Cache Path	/var/folders/wO/wO9G3Z5zGNKubHRALlh3t+++TI/-Caches-/com.apple.Xcode.501/SharedPrecomp...
▼Build Options	
Build Variants	normal
Debug Information Format	
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Uses Files From Build...	☑
Run Static Analyzer	☐
Scan All Source Files for Includes	☐
▼Code Signing	
Code Signing Identity	
Code Signing Resource Rules Path	
Other Code Signing Flags	
▼Compiler Version	
C/C++ Compiler Version	System default (GCC 4.2)
▼Deployment	
Additional Strip Flags	
Alternate Install Group	staff
Alternate Install Owner	cs
Alternate Install Permissions	u+w,go-w,a+rX
Alternate Permissions Files	
Deployment Location	☐
Deployment Postprocessing	☐
Install Group	staff
Install Owner	cs
Install Permissions	u+w,go-w,a+rX
Installation Build Products Location	/tmp/TestPlugin.dst
Installation Directory	
Mac OS X Deployment Target	Compiler Default

A list of the architectures for which the product will be built. This is usually set to a predefined build setting provided by the platform. If more than one architecture is specified, a universal binary will be produced. [ARCHS]

Based On: Nothing

Figure 3: *We change the architecture setting to Universal binary*

on compilation. It simply does not know what an "SInt32" or an "UInt64" data type is. With REALbasic 2008r1 you have better array access functions in the plugin SDK. You need this to create and fill an array in the plugin and return it to REALbasic code. And starting with REALbasic 2008r3 the SDK comes with everything you need to make plugins for the new Cocoa target.

Plugin file format

Plugin archives in REALbasic are virtual volumes. You can create and open virtual volumes in REALbasic using folderitem.CreateVirtualVolume and folderitem.OpenAsVirtualVolume. This virtual volume contains for each platform a shared library in a special folder hierarchy. There is a folder for build resources containing the shared libraries and there are folders for IDE resources which contain the help files or the control palette pictures.

You don't need to build this folder hierarchy yourself. You can use the Plugin Converter project that comes with the plugin SDK to create the plugin archive for your libraries. Over the long run you will want to write your own tool so you can automate the build process.

If you like you can add pictures for the control palette. For a better look on different operating systems you can even have different pictures there. One for Windows 2000, another picture for Windows XP, and one for Windows Vista. You can also add html pages for the online help. But using a lot of html pages slows REALbasic down as the html files are copied to a temporary folder.

Developing a plugin

Now we want to show you how to start with the plugin SDK to get a plugin compiled. Together we will create a little plugin with one simple function and compile it for all three platforms. At the end we merge those libraries to one plugin archive.

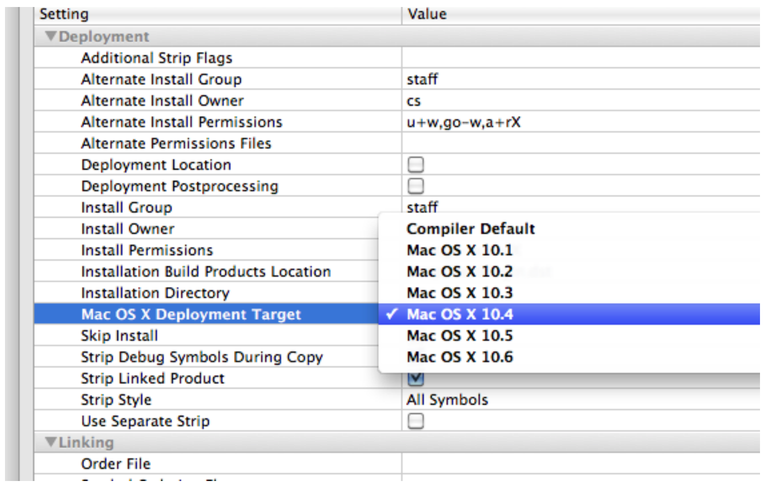
To get started go to the REALbasic website and download the REALbasic package. Inside the extras folder you should find the Plugin SDK files.

Mac OS X

On Mac OS X we use Xcode for this example. Inside the examples that come with the plugin SDK, you find a folder named "Xcode Mach-O Template". Inside is the "REALbasic Mach-O Plugin" folder which you want to copy to the other Xcode templates. Depending on your Xcode version this location is different, but for Snow Leopard you can use the templates folder in "/Developer/Library/Xcode/Project Templates".

Now launch Xcode and create a new project. Select the "Xcode Mach-O Template" template and save your new project (see figure 1). Once you have the project open, you see a lot of red entries (see figure 2). To fix this, find your plugin project folder in the Finder and copy the folders "Glue Code" and "Includes" from the plugin SDK. Once you are back in Xcode the red entries are gone and Xcode finds all of the files.

Now you want to edit the project settings and opt to compile a universal binary. For that you select in Architecture the entry "32-bit Universal". Also you want to check the deployment target setting. Scroll down and look for the



Setting	Value
▼Deployment	
Additional Strip Flags	
Alternate Install Group	staff
Alternate Install Owner	cs
Alternate Install Permissions	u+w,go-w,a+rX
Alternate Permissions Files	
Deployment Location	☐
Deployment Postprocessing	☐
Install Group	staff
Install Owner	
Install Permissions	
Installation Build Products Location	
Installation Directory	
Mac OS X Deployment Target	✓ Mac OS X 10.4
Skip Install	
Strip Debug Symbols During Copy	
Strip Linked Product	✓
Strip Style	All Symbols
Use Separate Strip	☐
▼Linking	
Order File	

Figure 4: *We change the deployment target setting to Mac OS X 10.4*

"Mac OS X deployment target" which is by default the "Compiler default". Change it to "Mac OS X 10.4" to compile your plugin for all Mac OS X versions starting with Mac OS X 10.4. (see figures 3 and 4)

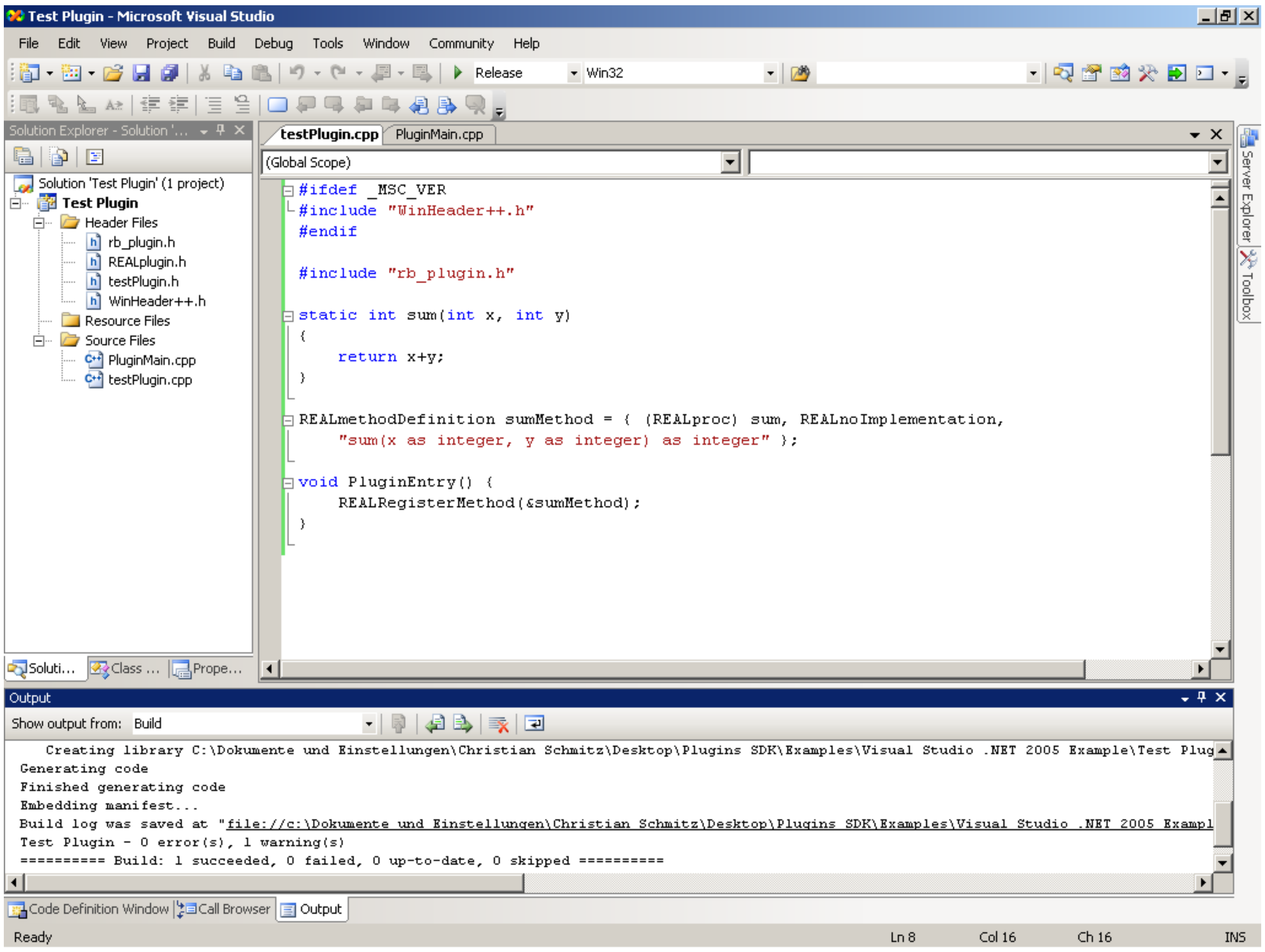
Now you can build your project once. Hit Command-B or select Build from the Build menu. It takes some time for Xcode to build the project, but once it finishes you should see an error message or a "Succeeded" message on the bottom right of the window. Now replace your plugin cpp file with this code. By default it is very short with just an empty PluginEntry function and one include line. Take this code:

```
#ifdef _MSC_VER
#include "WinHeader++.h"
#endif

#include "rb_plugin.h"

static int sum(int x, int y)
{
 return x+y;
}

REALmethodDefinition sumMethod = {
        (REALproc) sum,
        REALnoImplementation, "sum(x as
        integer, y as integer) as integer" };

void PluginEntry()
{
 REALRegisterMethod(&sumMethod);
}
```

Let's discuss the code above. The first lines are for Windows only. So we use conditional compiling to only compile them where the _MSC_VER constant is defined. Actually this is the version number of the Microsoft C compiler which exists only if the code is compiled using the Microsoft C compiler. Mac and Linux targets will ignore it. As we want to use the same code for all platforms, we have it already here. Next is the include for the REALbasic plugin functions. In the rb_plugin.h file and the realplugin.h file it references you find all the classes, constants, and functions in the REALBasic plugin API.

Now in the next 4 lines you find the definition of the C function called sum. It basically adds one integer to another integer and returns that value. Very simple. The key thing is the next line. There we define a REALmethodDefinition structure which is the definition for a global function. This definition tells the REALbasic compiler which C function it needs to call for the given REALbasic function. And for REALbasic we provide here the function pointer to the sum function and the REALbasic declaration. The compiler parses this declaration in order to know how to call our function in C. Although both are named sum here, only the name in the REALbasic method matters. The C name is just decoration, but we keep it in sync because this name can appear in a crash report.

In the PluginEntry function, you can do all the initialization of your plugin. Be aware that this function is called in the IDE so the compiler can learn the plugin function. So you should check for your initialization code to make it very efficient and not mess up the IDE. Better to make initialization methods or put your initialization in a class constructor. But the important part of the initialization is what we do here. We register the global



```
#ifdef _MSC_VER
#include "WinHeader++.h"
#endif

#include "rb_plugin.h"

static int sum(int x, int y)
{
    return x+y;
}

REALmethodDefinition sumMethod = { (REALproc) sum, REALnoImplementation,
    "sum(x as integer, y as integer) as integer" };

void PluginEntry() {
    REALRegisterMethod(&sumMethod);
}
```

```
   Creating library C:\Dokumente und Einstellungen\Christian Schmitz\Desktop\Plugins SDK\Examples\Visual Studio .NET 2005 Example\Test Plug
 Generating code
 Finished generating code
 Embedding manifest...
 Build log was saved at "file://c:\Dokumente und Einstellungen\Christian Schmitz\Desktop\Plugins SDK\Examples\Visual Studio .NET 2005 Exampl
 Test Plugin - 0 error(s), 1 warning(s)
 ========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Figure 5: *In Visual C on Windows with the test plugin project*

method to REALbasic by calling the REALRegisterMethod function. Actually you can see a deprecation warning here. For your plugin you might want to go and make a new module or a class and add a shared method. But for some things you want to have a short name and still use a global method.

If you want to test it, copy the compiled dylib to your plugins folder. REALbasic loads plugin libraries for the current platform directly from the plugins folder. But for bundling plugin libraries for several platform, you need to create the rbx archive. Run REALbasic with the dylib in your plugins folder and you can use the sum function in your code:

```
MsgBox str(sum(5,4))
```

This messagebox shows 9 if the plugin is found and compiled correctly. That is your plugin for the Mac. Next we look at the Windows target.

Windows

For Windows, you start by copying the plugin SDK to a Windows (virtual) machine. For this example I prefer to use Visual C 2005 on Windows 2000 inside VMWare Fusion. Visual C 2005 is a good choice because the plugin SDK comes with an example for that Visual C version.

So copy your plugin SDK folder to a Windows machine. Locate the “Visual Studio .NET 2005 Example” folder in the examples. Open the Test Plugin project in Visual C by double clicking the “Test Plugin.vcproj” file. We copied the whole plugin SDK file so Visual C automatically finds the Includes and GlueCode files automatically.

Now change TestPlugin.cpp. You should simply copy the code we have above on the Mac to Windows with the clipboard. Choose Build Solution from the Build menu and after a couple of seconds, your project will be compiled (see figure 5). You find several files in the Release folder inside the project folder, but the one you need is the “Test Plugin.dll” file.

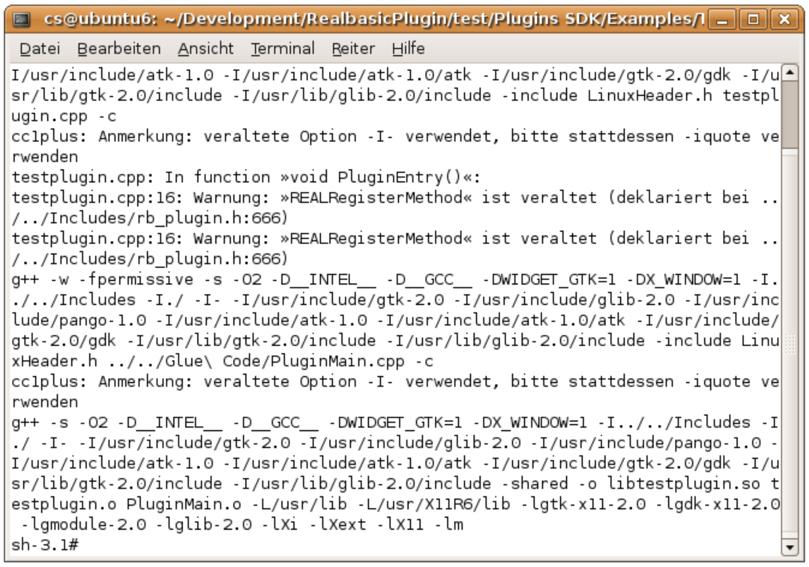
With that DLL we have the Windows part compiled. If you like you can drop this DLL file into the plugins folder on Windows and launch REALbasic to test it.

Linux

For Linux you copy the plugin SDK on your Linux machine. Also you copy the TestPlugin from Mac or Windows to the Linux machine. Now in the plugin SDK you find the example “Box Control” which has a makefile for you. Copy this makefile to your PluginTest folder. A makefile is a little script file which is run with the make tool. It runs compiler and linker commands if needed so you don’t need to recompile everything every time. You need to modify it:

First replace all instances of “boxcontrol” with “testplugin”. On the top of the makefile below the comment lines you find a few constants defined. The PLUGINSDKDIR needs to point to your plugin SDK folder. You should change it to "../.." for the case where you have your testplugin inside the examples folder. For other folders you need to have the path different of course. Further down you find the definition of the PREFIXHEADER constant. Change it to "LinuxHeader.h" so it finds the Linux header file correctly.

Now you can go to the terminal. Use cd and ls commands to go inside the plugin folder. Now run the makefile with the make command. You type "make all" to call the make utility and let it start on the all: line in the makefile. From there it will see that it needs to compile both testplugin.o and PluginMain.o before it compiles the libtestplugin.so library (see figure 6). This library file is needed for the next step. Of course you can test your plugin by copying it into the plugins folder and running REALbasic for Linux.


cs@ubuntu6: ~/Development/RealbasicPlugin/test/Plugins SDK/Examples/1
Datei Bearbeiten Ansicht Terminal Reiter Hilfe

```
I/usr/include/atk-1.0 -I/usr/include/atk-1.0/atk -I/usr/include/gtk-2.0/gdk -I/u
sr/lib/gtk-2.0/include -I/usr/lib/glib-2.0/include -include LinuxHeader.h testpl
ugin.cpp -c
cc1plus: Anmerkung: veraltete Option -I- verwendet, bitte stattdessen -iquote ve
rwenden
testplugin.cpp: In function »void PluginEntry()«:
testplugin.cpp:16: Warnung: »REALRegisterMethod« ist veraltet (deklariert bei ..
/../Includes/rb_plugin.h:666)
testplugin.cpp:16: Warnung: »REALRegisterMethod« ist veraltet (deklariert bei ..
/../Includes/rb_plugin.h:666)
g++ -w -fpermissive -s -O2 -D__INTEL__ -D__GCC__ -DWIDGET_GTK=1 -DX_WINDOW=1 -I.
./../Includes -I./ -I- -I/usr/include/gtk-2.0 -I/usr/include/glib-2.0 -I/usr/inc
lude/pango-1.0 -I/usr/include/atk-1.0 -I/usr/include/atk-1.0/atk -I/usr/include/
gtk-2.0/gdk -I/usr/lib/gtk-2.0/include -I/usr/lib/glib-2.0/include -include Linu
xHeader.h ../../Glue\ Code/PluginMain.cpp -c
cc1plus: Anmerkung: veraltete Option -I- verwendet, bitte stattdessen -iquote ve
rwenden
g++ -s -O2 -D__INTEL__ -D__GCC__ -DWIDGET_GTK=1 -DX_WINDOW=1 -I../../Includes -I
./ -I- -I/usr/include/gtk-2.0 -I/usr/include/glib-2.0 -I/usr/include/pango-1.0 -
I/usr/include/atk-1.0 -I/usr/include/atk-1.0/atk -I/usr/include/gtk-2.0/gdk -I/u
sr/lib/gtk-2.0/include -I/usr/lib/glib-2.0/include -shared -o libtestplugin.so t
estplugin.o PluginMain.o -L/usr/lib -L/usr/X11R6/lib -lgtk-x11-2.0 -lgdk-x11-2.0
 -lgmodule-2.0 -lglib-2.0 -lXi -lXext -lX11 -lm
sh-3.1#
```

Figure 6: *The typical output of make in Linux*

Merging parts

Back on your favorite platform you now have all three libraries in one folder. Inside the plugin SDK folder you find the "Plugin Converter" folder. There you find the "Plugin Converter.rbp" project which you compile for your platform. Now you have the plugin converter command line application. Open the terminal on Mac OS X/Linux or the command shell on Windows. There you can use cd to go to the folder with your libraries. Now run the PluginConverter tool with the following options:

```
PluginConverter -n test.rbx -w32 "Test Plugin.dll" -car TestPlugin.dylib -lnx
        libtestplugin.so
```

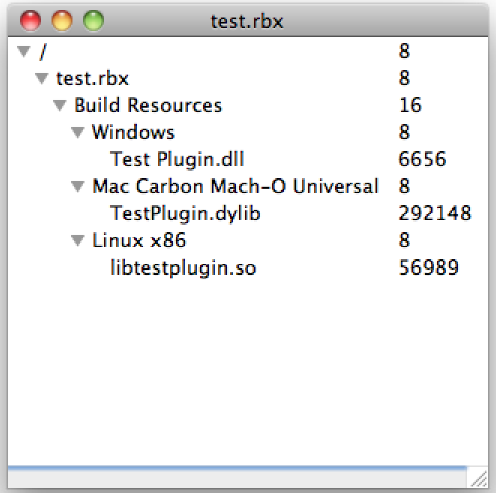
The -n switch tells the converter to create a new plugin. With the other three options you can specify the parts to embed. We specify the Windows part, the Carbon partn and the Linux part. In the future you may want to add another dylib for the Cocoa target.

Some more words

In the long run you will want to have all platforms share the same files. A common way is to host them on one platform and use Samba file sharing (SMB) to let all platforms access all files. Or to use subversion (SVN) to synchronize between platforms. After you setup your file sharing you need to setup both projects and the makefile. The paths must be correct and the build folder paths need to be changed too. Also you may want to write a script to create the final rbx file automatically.

On MonkeybreadSoftware we currently have 350 plugin projects which assemble to 43 rbx plugin files. So you see you can have several plugins in one rbx file. But with this number you need to automate things. So we have several scripts which compile all these plugins. One for Windows, one for Mac OS X, and one for Linux. And a final one to create the rbx files. Also we use self-written tools to write the scripts above, create the rbx files, and extract the declarations for the documentation.

If you have questions about developing plugins, please ask your questions in the plugins section of the REALbasic forums.



```
▼ /                                 8
  ▼ test.rbx                        8
    ▼ Build Resources               16
      ▼ Windows                     8
          Test Plugin.dll           6656
      ▼ Mac Carbon Mach-O Universal 8
          TestPlugin.dylib          292148
      ▼ Linux x86                   8
          libtestplugin.so          56989
```

Figure 7: *Looking into the final plugin file*

But please read the documentation coming with the SDK first, check the other plugins and search on the web for other people having the same problems first.

WEB RESOURCES:

http://www.realsoftware.com/download/

http://www.monkeybreadsoftware.de/realbasic/movies/CreatePluginMac.shtml

http://forums.realsoftware.com/viewforum.php?f=5

by Geoffrey A. Rabe

RB Summit 2009

Visiting the Boulder, Colorado Conference

AT A GLANCE

RBD#8107

Target Reader:
Beginner

About the Author:
Geoffrey began programming on a Radio Shack Color Computer, back in the early-to-mid 80s. When he could finally afford one, he bought a PC clone and VB3-6, but eventually wised up and moved to the Mac platform and REALbasic 2009r3. He's much happier now! (All photos courtesy of Geoffrey Rabe.)

INSPIRING Apps, an independent Colorado firm that specializes in writing custom software using REALbasic, and the Association of REALbasic Professionals (ARBP), jointly sponsored a three-day Summit at Inspiring Apps's offices in Boulder (see Figure 1), September 24-26, the first such REALbasic conference held in Colorado.

The Summit began Thursday afternoon with a "meet-and-greet" during which our hosts, Joe Strout and Brad Weber of Inspiring Apps, and ARBP President, Bob Keeney, outlined the sessions to be held the next two days. Following that, we broke for dinner at several nearby restaurants on Pearl Street, a pedestrian mall that runs through downtown Boulder (see Figure 9).

On Saturday morning, Bob Keeney (who also writes the *BKeeney Briefs* column for this magazine) had the first session, "Reporting Tools, Options, and Techniques" (see Figure 2). He mentioned that, at the last REALWorld Conference, there was only one such application to use for reporting, On-Target Reports, and that a good reporting tool was at the top of the list of feature requests received by REAL Software's Feedback tool. But one year later, there are several, in various stages of development, usability and support (see his column in this issue for more on that topic). He also discussed how making a print-out was one such way of "reporting" that many REALbasic developers use, and the various tools and controls for making print-outs that come with REALbasic.

During the next session, we engaged in a live iChat with Geoff Perlman, CEO of REAL Software (see Figure 3). Many of his comments regarded the then-upcoming Release 4 of RB 2009. The transition of the underlying framework for Mac OS X from Carbon to Cocoa was mentioned, but was not going to be ready for R4, nor was an interesting-sounding new control, the Passcode Field, which would allow for built-in password-checking functionality, though they both would be in a future release. Related to Bob's session, Geoff also showed a demo using the reporting tool now built-in to REALbasic, and as noted earlier, one of the most requested features. He also mentioned moving away from Type and Creator codes, long used by Macs to identify which programs could open which files, beginning with Tiger, to Uniform Type Identifiers, a much more useful way to do this. (A very informative article on Uniform Type Identifiers can be found at http://forums.appleinsider.com/showthread.php?s=&threadid=103219 if you're interested.)

The next several sessions and demos were all centered around databases, an area I'm no expert in, though the comments and discussion of the other participants left me wanting to know more. First up was Ryan Vail, of Inspiring Apps (see Figure 4). His

Figure 1: *Taken Thursday, during the "meet-and-greet." Brad Weber noticed the rainbow out of the window of the Inspiring Apps office and pointed it out to us, commenting on the frequently-changing nature of the weather here on the Front Range.*

session, "SQLite Power Tools," provided an introduction to the uses of SQLite, with many examples tied in with the REALSQLdatabase, documentation for which can be found within the Language Reference. Some pros mentioned in association with this are that it's free (i.e., it comes with RB), it's cross-platform, and while not for use directly in the IDE, there is a Firefox plug-in available for development in the browser environment; a con that was mentioned was its user-interface. In addition, he went over the command-line client of SQLite3, which comes with the development tools, and provided a demo. He also showed demos for encrypting, backing-up, and attaching databases (e.g., shipping a program to a client with some default information set on a USB drive). Both attaching and detaching a database sounded relatively easy.

However, importing an SQLite database to a MySQL database did not sound nearly as simple. The reference Ryan gave for this can be found at http://stackoverflow.com/questions/18671/ if you'd like more details. However, I discovered another reference online (http://sqlite.phxsoftware.com/forums/p/941/4725.

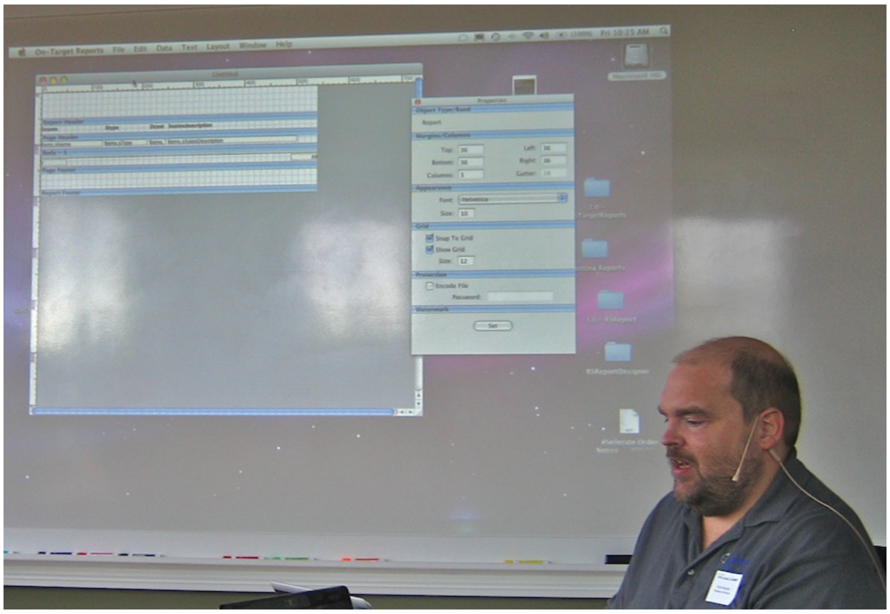


Figure 2: *This are from the first session, on Friday ("Reporting Tools, Options, and Techniques") given by Bob Keeney, Pres. of the ARBP, co-sponsors of the Summit, and frequent contributor to RB Developer.*

Figure 3: *This is from the second session ("Cocoa and Reporting Features in REALbasic") which was given by Geoff Perlman, head of REAL Software, via iChat, where he also discussed some of the changes/features of RB2009r4, which is now available. (That's Joe Strout in the blue shirt in the foreground.)*

aspx) which was easier for me to understand given my lack of experience with databases.

Ryan's session was followed by a demo, given by Brad Weber, of how even large databases can be imported, encrypted/decrypted, and zipped, which led to a large discussion on the topic, which — as mentioned above — was beyond my experience. But on the other hand, Seth Verrinder, of BKeeney Software, Inc., led the next session, "OO Database Framework and Introspection," which was easier for me to follow.

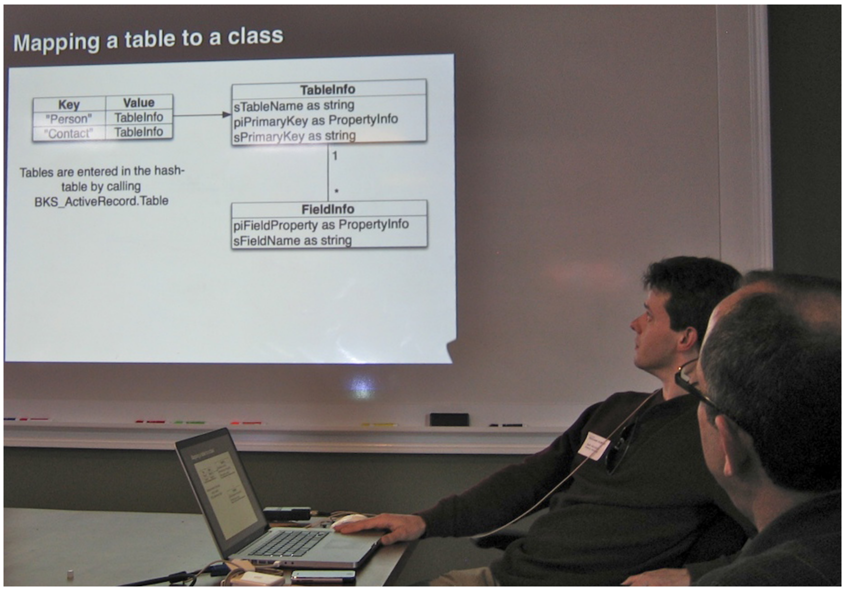

Mapping a table to a class
Key
Value
"Person"
TableInfo
"Contact"
TableInfo
Tables are entered in the hash-table by calling BKS_ActiveRecord.Table
TableInfo
sTableName as string
piPrimaryKey as PropertyInfo
sPrimaryKey as string
1
*
FieldInfo
piFieldProperty as PropertyInfo
sFieldName as string

Figure 4: *This is Ryan Vail of Inspiring Apps, delivering his session on "SQLite Power Tools."*

I want to make a comment here which relates to the Summit specifically and to the REALbasic community in general. As mentioned several times, I don't have nearly as much experience using RB as the other participants, and in fact I'm more of a hobbyist, while the others were all professional programmers. However, I found the group to be very helpful and patient with my questions and comments. In fact, for a project I've been "working" on (more planning than actual programming) I needed a custom control which I've tried designing myself with rather unsatisfactory results. However, as a result of my bringing this up with the group, the following day Bob Keeney had an excellent suggestion on a different approach I could take designing the control, and both Christian Schmitz and Joe Strout gave me a control they had whipped up in minutes. So what I'm getting at is that the RB community, from my experience and what I've seen online and in the *REALbasic Developer* magazine, is *very* open and helpful to both new and experienced programmers, and that these smaller events are well worth attending even if you're not a full-time developer.

(In a similar vein, I found the following quote from a review of REALbasic 5 on MacWorld's site: *Software-Development Application Remains Top in Its Field* "When you commit to REALbasic, you become part of an enormous base of smart, dedicated, and friendly users who provide a variety of great resources, from books to message boards."

Now, back to the Summit....)

Speaking of Joe, who is a frequent contributor to the RB community in many areas, he had the next session, "Profiling and Performance Tweaking." Simply speaking, this topic relates to getting the most out of your code and the best performance of your application. One of his first points was on program optimization — "don't do it yet" — wait until after you've profiled your code. How to do this? You can do it manually, using the Pause button at certain points in your application's run, to check properties, variables, etc. You can also measure the time certain routines take by including profiling code before and after the run, using microseconds.

In addition, Joe included some performance pitfalls to watch out for: with databases, minimize the number of queues and cache small look-up tables; avoid unnecessary double-buffering (and be careful — OSX uses double-buffering with all drawing routines); when repeated operations are involved, `DIM` local variables, as they are always fast; and avoid creating

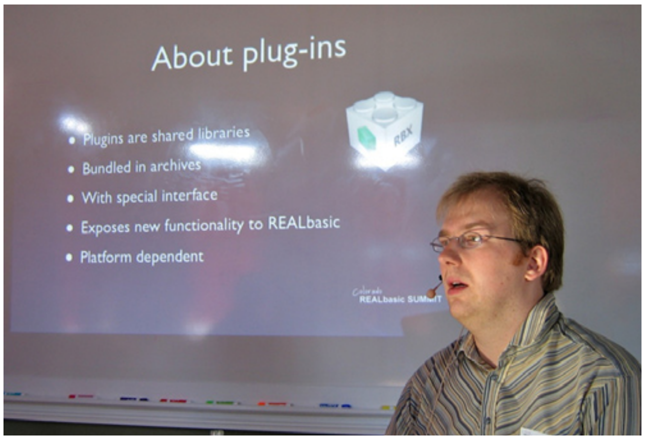


Figure 5: *This is Christian Schmitz of Monkeybread Software talking about "Building REALbasic Plugins," the first session on Saturday.*

and destroying objects and strings when possible. And he also mentioned some handy habits: retain objects you'll need again, creating and exporting them; use `Split` and `Join`, which are almost always faster than `CountFields` and `NthField`; flipping loops around can sometimes add speed; and use a `Dictionary` for random look-ups.

The last session on Friday was also one of the most lively ones: "Version Control, and Why it Rocks," by Aaron Gerber and Mathias Gran. You could tell they were really into their subject from the start as they provided lots of fun comments and examples. Unfortunately, their's was one of the longer sessions and is beyond the scope of this article. However, they did provide some URLs to resources online: you can get the Mac Subversion Client (http://versionsapp.com/#), the documentation (http://svnbook.red-bean.com/), and the main site for Subversion (http://subversion.tigris.org/).

After Friday's last session, we were in for quite a treat. Included with the Summit's ticket price was dinner at Salt, a fairly new restaurant downstairs from and next door to IA's offices. Our group took over most of the basement seating area, and we could order basically whatever we wanted to eat and drink. However, this was more than just a nice dinner, but a chance to socialize and network with the other members of the Summit, and as such was very worthwhile. It also showed what great hosts we had!

The first session on Saturday was "Building REALbasic Plugins," by Christian Schmitz, well-known for his MonkeyBread Software collection of plugins for REALbasic (see Figure 5). However, again, the summation of this presentation is beyond the scope of this article (he's written a full article for this magazine based on his presentation). But Christian is very accessible and helpful, via his website (http://www.monkeybreadsoftware.de/) and mailing list (https://www.monkeybreadsoftware.de/realbasic/mailinglist.shtml).

The last formal session was "Usability and Design Techniques," by Jay Crain, Inspiring Apps Designer (see Figure 6). He packed quite a lot into his hour, all

Introducing Usability
Web vs. desktop
Shared principles
Re-newed focus on usability has grown with the web.
User-Centered Design (UCD)
Human factors and ergonomics have been around since the 40s
UCD is a Scandinavian principle that has been evolving since the 70s
REALbasic SUMMIT

Figure 6: *This is Jay Crain of Inspiring Apps talking about "Usability ¥ Design Techniques," the second session on Saturday.*

of it great, but I'll try to summarize briefly. His introduction to usability began with a comparison of desktop versus web design, their shared principles, and he mentioned a renewed focus on usability having grown with the web. A major point he began with is user-centered design, putting the user at the center of your design strategy. Some principles he shared are as follows: designs should be transparent; aesthetics are important, but not everything; and the success of a design can be measured by how infrequently the user has to stop and figure out how the program works. He also emphasized consistency: within the user expectations; within an app or web site; and between related apps and web sites. Inconsistencies can also be a strength within an app,

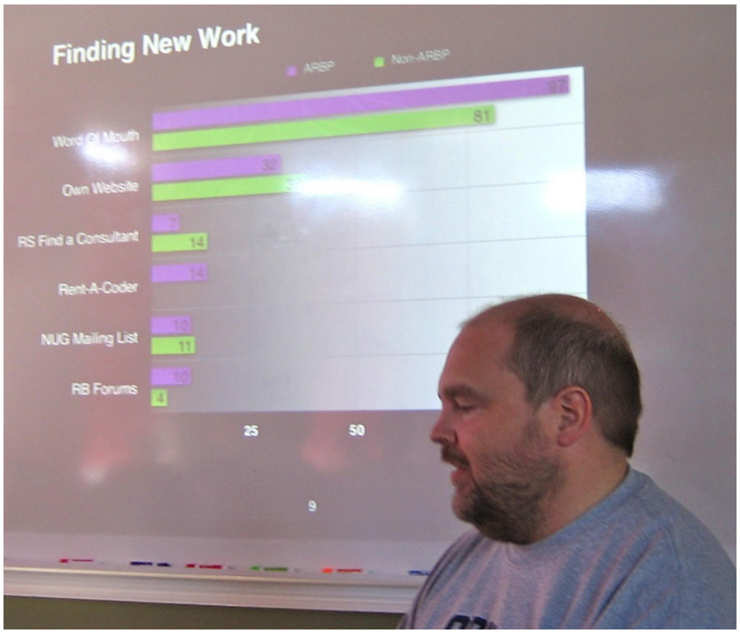


Figure 7: *Bob Keeney, discussing the state of REALbasic programmers, and showing the difference between ARBP members and non-members.*

Figure 8: *Various participants of the Summit. That's Brad Weber on the laptop, leading the "Facilitated RB/RS/Summit Feedback Session" at the end of the Summit, on Saturday afternoon.*

as cues to the user that some things are meant to act or to be operated differently. In addition, he says to always allow Undo, to give the user an 'out,' and that feedback is key, to let the user know they are in the right place or doing something the right way. All in all, many useful suggestions.

After Jay's session and a lunch break, we all were involved in a feedback session, led by Brad Weber (see Figure 8). This was not only to be about the Summit, though of course that was key to the people from Inspiring Apps and Bob Keeney (for the ARBP), but we also discussed in general our thoughts on the RB community as a whole. Norman Palardy, unofficial REAL Software representative and frequent pres-

Figure 9: *After the "meet-and-greet" on Thursday afternoon opening of the Summit, people split into groups for dinner. This is at Old Chicago Pizza. Clockwise around the table, from the left: Monika and Christian Schmitz, Rick Praetzel, Rick Roberts, Fred Roller, and the author, Geoffrey Rabe.*

Figure 10: *Another shot of the participants of the Summit, also showing Ryan Vail (in front with the laptop) and Seth Verrinder, operating the video camera to tape the conference.*

ence on the RB Forums, answered some questions and comments, and the session was very positive for all involved.

As I mentioned, I found the Summit and my interaction with professional members of the RB community to be a very good one, and I look forward to future conferences, here in Boulder, or abroad, if I can make it. And I encourage any RB programmer out there who has a chance to attend such a conference, whether it's REAL World in Austin, or one of the smaller ones that are beginning to take place around the country, to do so. I think you'll find it very worthwhile (and a lot of fun!).

New to *REALbasic Developer*? You need *all* the issues!

Order back issues in Print or on CD!

CDs Include

Issues in high-res PDF format, searchable database, project files, REALbasic University archive, and more!

Order today—this offer is for a limited time!

www.rbdeveloper.com/cd/