Practical .NET Chart Development and Applications

Book · March 2016

CITATIONS
0

READS
514

1 author:

Ji-Hai Xu
drxudotnet.com
42 PUBLICATIONS **644** CITATIONS

SEE PROFILE

Practical .NET Chart Development and Applications

Advanced Chart Programming
for Real-World .NET 4.5 Applications
Using C#, WPF, and MVVM

Practical .NET Chart Development and Applications

Advanced Chart Programming
for Real-World .NET 4.5 Applications
Using C#, WPF, and MVVM

Jack Xu, PhD

UniCAD Publishing

Practical .NET Chart Development and Applications

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1UC

Editor: Anna Hsu

Contact:
jxu@dxudotnet.com
Visit us at our website: www.drxudotnet.com

Published by UniCAD Publishing.
New York, USA
ISBN-13: 978-0-9793725-4-4
ISBN-10: 0-9793725-4-2

Publisher's Cataloging-in-Publication Data

Xu, Jack
Practical .NET Chart Development and Applications – Advanced Chart Programming for Real-World .NET 4.5 Applications Using C#, WPF, and MVVM/ Jack Xu
– 1st ed.
p.cm.
ISBN 978-0-9793725-4-4

1. C# Programming. 2. Charts and Graphics. 3. XAML. 4. Windows Presentation Foundation. 5. Data Binding. 6. .NET Applications. 7. Model-View-ViewModel. 8. MVVM. 9. SQL Database. 10. ADO.NET. 11. Entity Framework. 12. LINQ
I. Title. II. Title. III Title: Practical .NET Chart Development and Applications

For my wonderful family

Contents

Introduction

Overview

Welcome to *Practical .NET Chart Development and Applications*. This book will provide all the tools you need to develop professional chart applications and reusable chart control packages using C#, the Windows Presentation Foundation (WPF), and the Model-View-View Model (MVVM) pattern based on the .NET 4.5 Framework. I hope this book will be useful for .NET programmers of all skill levels.

We have all heard the saying "A picture's worth a thousand words", and charts are some of the most informative pictures there are. Charts play an important role in every Windows application. They make data easier to understand, add interest to reports, and have wide applications in our daily life. The scientific, engineering, mathematics, and financial communities always have a need to present data and results graphically. Microsoft's .NET platform with C# and WPF is one of the few and best development tools available for providing the capability both to generate data as a simulation engine and to display it in a variety of graphical representations based on the .NET graphics capacity.

As a C# programmer, you are probably already familiar with Windows Forms, the mature and full-featured development tool built on top of the .NET Framework that uses the Windows Application Programming Interface (API) to create the visual appearance of standard user interface elements. It provides all kinds of tools for laying out windows, menus, dialogs, and controls. You can also develop graphics applications based on Windows Forms using the Graphics Device Interface (GDI+). However, creating a feature-rich graphics application using Windows Forms can be a difficult and tedious task. For example, Windows Forms provides no tools for creating three-dimensional (3D) graphics applications. Even a 3D point, the simplest of 3D graphics objects, must be defined first in a suitable 3D coordinate system before it can be used as a 3D graphics object.

WPF changes the landscape of graphics programming completely. At first, you might think that WPF simply provides another way to create windows, menus, dialogs, and controls. However, WPF has much more to offer than any other Windows programming framework. It integrates three basic Windows elements – text, controls, and graphics – into a single programming model and puts these three elements into the same element tree in the same manner.

Without WPF, developing a chart and graphics application would involve a number of different technologies, ranging from GDI/GDI+ for 2D graphics to Direct3D or OpenGL for 3D graphics. WPF, on the contrary, is designed as a single model for graphics application development, providing seamless

integration between such services within an application. Similar constructs can be used for creating animation, data binding, and 3D models.

To take further advantage of new, powerful graphics hardware technologies, WPF implements a vector-based graphics model. This allows for graphics to be scaled based on screen-specific resolution without the loss of image quality, something impossible with fixed-size raster graphics. In addition, WPF leverages Direct3D for vector-based rendering and makes use of the graphics processing unit (GPU) on any video card that implements DirectX in hardware.

With WPF, graphics elements can easily be integrated into any part of your user interface. For example, WPF provides 2D shape elements that can be involved in the user interface (UI) tree like other elements can. You are free to mix these shapes with any other kind of element, such as a button. The WPF 3D model is based on Direct3D technology and allows you to create a custom 3D shape library that can be reused in your projects. The main benefits that WPF offers in 3D are its ease of use and its ability to integrate 3D content anywhere in a WPF application.

Another powerful feature of WPF is its data binding, which provides a simple and consistent way for applications to present and interact with data. WPF data binding is the process that establishes a connection between the application UI and business logic. If the data has the correct settings and provides the proper notifications, then when the data changes its value, the elements that are bound to the data reflect the changes automatically. WPF also provides two-way data binding: namely, if an outer representation of the data in an element changes, then the underlying data can be automatically updated to reflect the change. The data binding functionality in WPF has several advantages over traditional models, including a broad range of properties that inherently support data binding, flexible UI representation of data, and clean separation of business logic from UI.

The Model-View-View Model (MVVM) pattern is the most used architecture for WPF applications. MVVM introduces three layers of separation of application code: *Model*, *View*, and *ViewModel*. *View* holds the actual UI; *ViewModel* holds the collection of properties, commands, and property changed notifications; while *Model* holds business data, business logic, and business rules. You will gain several advantages using the MVVM pattern, including: 1) proper separation of the view and the data. The data is not stored in the view and the view is just for presenting the data; 2) clean, testable, and manageable code; and 3) no code-behind so that the presentation layer and the logic are loosely coupled.

As you may have already noticed, a plethora of WPF programming books are currently available in bookstores. The vast majority of these books are general-purpose user guides and tutorials that explain the basics of WPF and how to use it to implement simple WPF applications. Users who want to take full advantage of WPF graphics and other advanced features, however, require a book that provides an in-depth introduction specifically to WPF chart development and applications based on WPF's good practice of data binding and MVVM.

This book is written with the intention of providing a complete and comprehensive explanation of .NET chart programming, and it pays special attention to creating various charts and reusable chart control packages that can be used directly in real-world .NET applications. Much of this book contains original work based on my own programming experience when I was developing commercial Computer Aided Design (CAD) packages and chart applications for quantitative analysis in the financial field. Without C#, WPF, and the .NET framework, developing advanced charts is a difficult and time-consuming task. To add even simple charts or graphs to your applications, you often have to waste effort creating a chart program, or buy commercial graphics and chart add-on packages.

Using third-party graphics and chart add-on products in your applications has several drawbacks, however:

- It is not cost effective – it might cost hundreds or thousands of dollars for a sophisticated graphics and chart package.
- Compatibility is an issue – these third-party graphics and chart add-on tools are usually provided as DLL or COM components, which often leads to unexpected interface exceptions and unstable operations.
- There is little flexibility – from users' point of view, these packages appear to be black boxes because the source code is usually not provided, making it hard for users to add or modify any functionalities. You might find that these third-party products lack the special features you want in your applications, even while they often provide an excess of extraneous functionalities you will never use.
- The coding is inefficient – these third-party add-on tools are often very large packages that contain far more functionalities than you will ever need in your applications. Even a simple program can end up with a huge final release due to the use of third party add-ons. This is very inefficient for both coding management and distribution.
- License royalty is another issue – some third-party add-ons require not only the developing license, but also the distributed license royalty, resulting in an unnecessary increase in development cost.
- Finally, maintenance is a problem – in most cases, third-party tools use a different programming language than the one you use in developing your applications, so you have to maintain the codes in an unmanaged manner.

Practical .NET Chart Development and Applications provides everything you need to create your own advanced chart applications and reusable chart control packages using C# and WPF based on the MVVM pattern. It shows you how to use C# and WPF to create a variety of chart applications that range from simple two-dimensional (2D) *X-Y* plots to complicated 3D surfaces and combination charts. I will try my best to introduce you to C# and WPF chart programming in a simple way – simple enough to be easily followed by a .NET developer who has basic prior experience in developing .NET applications. From this book, you can learn how to create a full range of 2D and 3D chart applications and how to use custom chart controls to create impressive charts without having to buy expensive third-party add-on products.

What this Book Includes

This book and its sample code listings, which are available for download at my website at www.drxudotnet.com, provide you with:

- A complete, in-depth instruction on practical .NET chart programming with C#, WPF, and MVVM. After reading this book and running the example programs, you will be able to add various sophisticated charts to your .NET applications.
- Ready-to-run example programs that allow you to explore the charting techniques described in the book. You can use these examples to understand how the chart algorithms work. You can modify the code examples or add new features to them to form the basis of your own projects. Some of the example code listings provided in this book are already sophisticated chart packages that you can use directly in your own real-world .NET applications.
- Many classes in the sample code listings that you will find useful in your .NET chart development. These classes include matrix manipulation, coordinate transformation, color maps, chart controls,

and the other useful utility classes. You can extract these classes and plug them into your own applications.

Is This Book for You?

You do not have to be an experienced .NET developer or an expert to use this book. I designed this book to be useful to people of all levels of .NET programming experience. In fact, I believe that if you have some prior experience with the programming language C#, Windows Forms, and the .NET framework, you will be able to sit down in front of your computer, start up Microsoft Visual Studio Community 2013 and .NET 4.5, follow the examples provided in this book, and quickly become proficient in .NET chart programming. For those of you who are already experienced .NET developers, I believe this book has much to offer as well. A great deal of the information about chart programming in this book is not available in other .NET tutorial and reference books. In addition, you can use most of the example programs directly in your own real-world application development. This book will provide you with a level of detail, explanation, instruction, and sample program code that will enable you to do just about anything related to .NET chart development and applications.

Perhaps you are a scientist, an engineer, a mathematician, a quant developer in finance, a student, or a teacher rather than a professional programmer; nevertheless, this book is still a good bet for you. In fact, my own background is in theoretical physics, a field involving extensive numerical calculations as well as graphical representations of calculated data. I devoted my effort to this field for many years, all the way from undergraduate to PhD. My first computer experience was with FORTRAN. Later on, I had programming experience with Basic, C, C++, and MATLAB. I still remember how hard it was in the early days to represent computational results graphically. I often spent hours creating a publication-quality chart by hand, using a ruler, graph paper, and rub-off lettering. A year later, our group bought a graphics and chart package; however, I still needed to prepare my data in a proper format in order to process it with this package. During that time, I started paying attention to various development tools that I could use to create integrated applications. I tried to find an ideal development tool that would allow me not only to generate data easily (computation capability) but also to represent data graphically (graphics and chart power). The C# and Microsoft Visual Studio .NET development environment made it possible to develop such integrated applications. Ever since Microsoft .NET 1.0 came out, I have been in love with the C# language, and I have used it successfully to create powerful graphics and chart applications, including commercial CAD packages and powerful 2D and 3D chart applications for quantitative analysis when I worked on Wall Street.

.NET developers and technical professionals can use the majority of the example programs in this book routinely. Throughout the book, I will emphasize the usefulness of chart programming to real-world applications. If you closely follow the instructions presented in this book, you will easily be able to develop various practical .NET chart applications, from 2D charts to a sophisticated 3D surface chart library. At the same time, I won't spend too much time discussing programming style, execution speed, and code optimization, because a plethora of books out there already deal with these topics. Most of the example programs you will find in this book omit error handlings. This makes the code easier to understand by focusing only on the key concepts and practical applications.

What Do You Need to Use This Book?

You will need no special equipment to make the best use of this book and understand the algorithms. To run and modify the sample programs, you will need a computer capable of running either Windows 7,

8, or 10. The software installed on your computer should include Visual Studio 2013 (Community version is fine), .NET 4.5 standard edition or higher, and SQL Server Express 2012 or higher. If you have Visual Studio 2012, .NET 4.0, and SQL Server Express 2008 or older versions, you can also run most of the sample code with few modifications. Please remember, however, that this book is intended for Visual Studio 2013, .NET 4.5, and SQL Server Express 2014, and that all of the example programs were created and tested on this platform, so it is best to run the sample code on the same platform.

How the Book Is Organized

This book is organized into nine chapters, each of which covers a different topic about .NET chart programming and applications. The following summaries of each chapter should give you an overview of the book's content:

Chapter 1, *Overview of C# and WPF Programming*

This chapter introduces the basics of WPF and reviews some of the general aspects of WPF programming, including XAML files used to define user interfaces.

Chapter 2, *Introduction to MVVM*

This chapter introduces processes for implementing the MVVM pattern in WPF applications, including data binding with property changed notifications, command binding, lambda expressions, and observable collections. It also reviews some free open-source MVVM toolkits, and explains why we will use Caliburn.Micro in this book.

Chapter 3, *Databases and the ADO.NET Entity Framework*

This chapter introduces the SQL Server Data Tool (SSDT) and LocalDB, two built-in features shipped as part of the core product of Visual Studio 2013, and shows how to create simple databases and interact with data. It also reviews the ADO.NET Entity Framework, which allows developers without extensive knowledge of SQL to interrogate the database, create complex queries, and generate classes with the help of a user-friendly interface.

Chapter 4, *2D Line Charts*

This chapter contains instructions on how to create elementary 2D *X-Y* line charts. It introduces basic chart elements, including the chart area, axes, title, labels, ticks, gridlines, symbols and legend. These basic chart elements are common in the other types of charts as well.

Chapter 5, *Specialized 2D Charts*

This chapter covers the specialized 2D charts often found in commercial chart packages and spreadsheet applications. These specialized charts include bar charts, stair-step charts, stem charts, charts with error bars, pie charts, area charts, and polar charts.

Chapter 6, *Stock Charts*

This chapter shows how to create a variety of stock charts in WPF, including the interface that interacts with market data stored in the database, the standard Hi-Lo-Open-Close stock charts, candlestick stock charts, volume charts, moving averages, and linear analysis. In addition, it also discusses how to retrieve data from stock charts.

Chapter 7, *2D Chart Controls*

This chapter shows how to convert 2D chart applications into a custom user control and how to reuse such a control in WPF applications based on data binding and the MVVM pattern

Chapter 8, *3D Charts*

This chapter begins with a description of 3D matrix transformation and the coordinate system used in 3D charts, and shows how to create 3D coordinate axes, tick marks, axis labels, and gridlines using the azimuth-elevation view. It then explains techniques for creating various 3D charts.

Chapter 9, *3D Chart Controls*

This chapter converts the 3D chart application developed in Chapter 8 into a custom user control that includes three modules: 3D line charts, 3D surface-like charts, and 3D specialized charts. It also shows you how to reuse this 3D chart control in your .NET applications based on WPF's advanced features such as data binding and the MVVM pattern.

Changes in this Book

I have received plenty of feedback from readers since I first published my books *Practical C# Charts and Graphics* (2007), *Practical WPF Graphics Programming* (2007), and *Practical WPF Charts and Graphics* (2009). Many of you asked for an updated edition. I realize that .NET technology has advanced and changed a lot in the past few years, and I now need to incorporate these new developments in the .NET Framework into my book. In this new book, I rewrite most of the example programs to reflect both the advancement of .NET and the new programming experience I have gained as a quant developer/analyst in the last few years. The key new features in this book include

- *Data binding*: In Windows Forms applications, data binding is mainly used for populating elements on your application with information. The beauty of data binding is that you can populate the interface while writing little to no code. With data binding in WPF you can take data from almost any property of any object and bind it to almost any other dependency property of another object. In this book, I will try to use data binding whenever possible to implement code examples.
- *Databases and the ADO.NET Entity Framework*. For the past several years, I have worked with a financial firm as a quant analyst/developer on Wall Street. The most important thing I deal with every day is market data. Most .NET applications in different fields also need to interact with data stored in databases. Therefore, this book includes a chapter that deals with databases and the ADO.NET entity Framework. It shows you how to create a simple database and how to use the entity data model to access the database data.
- *The MVVM pattern*: In traditional UI development, you create a view using a window or user control and then write all logical code in the code-behind. This approach creates a strong dependency between UI and data logic, which is hard to maintain and test. In MVVM, however, the glue code is the view model. If property values in the view model change, those new values automatically propagate to the view via data binding and notification. In this book, I will introduce the MVVM pattern and try to use the view model for data binding. In some examples, I may not use the full version of MVVM and instead write some code-behind code for dynamically creating WPF elements for simplicity's sake.
- *Powerful Chart Controls*: In this new book, I convert 2D line charts, stock charts, and 3D charts into powerful chart controls that you can easily reuse in your own .NET applications. In particular, these chart controls are MVVM compatible and allow you to develop .NET applications with 2D and 3D charts based on the MVVM pattern.

- *Financial Market*: This new book incorporates more topics and examples in the financial market, based mainly on my own working experience, including interaction with market data, moving average calculation, linear regression, principal component analysis (PCA) for pair trading, retrieving market data from stock charts, and implementing reusable stock chart controls.
- *Real-time Charts*: Many fields require real-time chart capability. For example, if you design a stock trading system, you need to develop a real-time data feeder and a chart control to display the real-time stock market data on your screen. This new book provides some examples that show how to create such real-time chart applications using our reusable chart controls.

Using Code Examples

You may use the code in this book in your own applications and documentation. You do not need to contact the author or the publisher for permission unless you are reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing the example code listings does require permission. Incorporating a significant amount of example code from this book into your applications and documentation also requires permission. Integrating the example code from this book into commercial products is not allowed without written permission of the author.

Customer Support

I am always interested in hearing from readers, and I would enjoy hearing your thoughts about this book. You can send me comments by e-mail to `jxu@DrXuDotNet.com`. I also provide updates, bug fixes, and ongoing support via my website:

```
www.DrXuDotNet.com
```

You can also obtain the complete source code for all of the examples in this book from the website.