Verwendung der Komponente Chart2D in C#

Autor: Lars Seckler, Diplomarbeit FH- Lübeck Juni 2011

Aktualisiert von Gunnar Weinschenker, 3.7.2016

Motivation für diese Komponente

Seit DotNet 4.0 gibt es die sehr umfangreiche absolut mächtige Komponente Chart zur grafischen Darstellung von Daten. Die Schnittstelle ist daher sehr komplex und man benötigt viel Einarbeitungszeit zum Programmieren der Funktionen. Will man nur Messdaten als Kurven grafisch darstellen, wie es in ingenieurtechnischen Programmen oft der fall ist, dann bietet sich diese Komponente an, die quasi einen „Wrapper“ um Chart darstellt. Das Interface ist auf wenige Funktionen eingeschränkt, um den Programmieraufwand deutlich zu reduzieren.

Außerdem bietet diese Komponente eine Toolbar an, mit der die wesentlichen Grundfunktionen wie Drucken, Beschriften, Scalieren, Zoomen und vieles mehr direkt für den Enduser angeboten werden. Man braucht sich also praktisch nicht mehr darum kümmern. In meinen Toolprogrammen RegC# und Windfc# wird genau diese Chart2D auch deswegen genutzt.

Für einen Programmierer, der unter Visual Studio eine Anwendung erstellt, ist es vorteilhaft, den gesamten Funktionsumfang einer Komponente zu kennen, um diese korrekt nutzen zu können. In diesem Kapitel wird beschrieben, wie die Komponente Chart2D in Visual Studio 2012/3/5 eingebunden werden kann und welche Eigenschaften, Methoden sowie Ereignisse für den Programmierer zur Verfügung stehen.

Einbindung von Chart2D in Visual Studio 2015

Die Komponente Chart2D steht als DLL-Datei (*Dynamic Link Library*) zur Verfügung und muss vor der Implementierung in ein Projekt unter Visual Studio 2012/3/5/6 eingebunden werden. Entweder wird die Komponente Chart2D direkt über die Toolbox als Element hinzugefügt. Eine einfachere Möglichkeit ist folgende: Öffnen Sie mit dem Windows- Explorer das Verzeichnis, in dem die DLL abgespeichert ist. Dann ziehen Sie mit gedrückter Maustaste (Drag & Drop) diese DLL in das Toolbox (Werkzeugkasten) – Fenster von VS. Dann ist dort die Komponente Chart2D verfügbar.

Nach erfolgreicher Einbindung erscheint Chart2D als eigenes Steuerelement in der Toolbox (Bild 3.1).

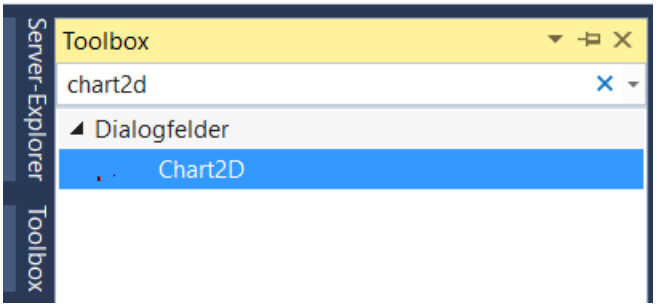

Bild 0.1: Visual Studio 2015 Toolbox

1.1 Verfügbare Eigenschaften, Methoden und Ereignisse

Die Komponente Chart2D ist ein `UserControl`, welches sich von der Klasse `Control` ableitet. Sie benutzt die Standard DotNet- Komponente Chart. Daher bietet es von sich aus bereits eine Vielzahl unterschiedlicher Eigenschaften, Methoden und Ereignisse[1]. Diese Standardelemente, wie zum Beispiel die Eigenschaften `Dock` oder `Visible`, werden in diesem Abschnitt jedoch nicht näher erläutert. Es wird gezielt auf die neu hinzugefügten eingegangen.

1.1.1 Eigenschaften

Chart2D bietet viele Eigenschaften, um die Darstellung der Graphen und die Benutzeroberfläche schon während der Programmierung komfortabel anpassen zu können. Nachfolgend sind die spezifischen Eigenschaften der Komponente in Kategorien, wie sie auch in Visual Studio angezeigt werden, aufgeführt.

Achseneinstellung:
In dieser Kategorie sind alle Eigenschaften zusammengefasst, die für die Einteilung der Achsen zuständig sind. Dazu zählen auch die minimalen und maximalen Werte der jeweiligen Achse. Außerdem besteht die Möglichkeit, das Koordinatenraster zu aktivieren beziehungsweise zu deaktivieren. Die Tabelle 3.1 zeigt eine Auflistung der Namen der Eigenschaften sowie deren Datentypen.

Name der Eigenschaft	Datentyp	Standardwert
XGap	double	1
XGrid	bool	False
XMax	double	10
XMin	double	0
YGap	double	1
YGrid	bool	False
YMax	double	10
YMin	double	0
Y2Min	Double	0
Y2Max	Double	0

Tabelle 0.1: Achseneinteilung

Die beiden „Minor Grid“, also die feine Griddarstellung lässt sich in dieser Version nicht ein- und ausschalten, das geht nur über die Toolbuttons.

Beschriftung:
Die beiden Eigenschaften dieser Kategorie ermöglichen es, der Darstellungsfläche von Chart2D eine Unter- und Überschrift hinzuzufügen (Tabelle 3.2).

Name der Eigenschaft	Datentyp	Standardwert
BottomTitle	string	-
TopTitle	string	-

Tabelle 0.2: Beschriftung

Links- und Rechtsbeschriftung sind hier nicht vorgesehen, das geht nur manuell über die T- Taste in der Buttonleiste

Buttonleiste:

Über diese Eigenschaften können einzelne Buttons gezielt ausgeblendet werden. Zudem ist es möglich, mit der Eigenschaft „ButtonPanel“ die gesamte Buttonleiste der Komponente ein- und auszuschalten. In Tabelle 3.3 sind alle Buttons aufgelistet.

Name der Eigenschaft	Datentyp	Standardwert
AllClearButton	bool	True
AutoScaleButton	bool	True
BackColorButton	bool	True
ButtonPanel	bool	True
ColorBlackButton	bool	True
EditorButton	bool	True
InfoButton	bool	True
LastClearButton	bool	True
LegendButton	bool	True
LoadButton	bool	True
PenScaleButton	bool	True
PrintButton	bool	True
SaveButton	bool	True
ScalingButton	bool	True
TitleButton	bool	True
XGridButton	bool	True
YGridButton	bool	True
ZoomButton	bool	True
ButtonLinLog	bool	True

Tabelle 0.3: Buttonleiste

Kurvenfarben:

Die Farben der einzelnen Graphen können unter Zuhilfenahme dieser Eigenschaften definiert werden. Die Tabelle 3.4 fasst die zehn Eigenschaften zusammen.

Name der Eigenschaft	Datentyp	Standardwert
CurveColor0	color	Blue
CurveColor1	color	Red
CurveColor2	color	Green
CurveColor3	color	Cyan
CurveColor4	color	Orange
CurveColor5	color	LightGreen
CurveColor6	color	Purple
CurveColor7	color	Pink
CurveColor8	color	Brown
CurveColor9	color	Yellow

Tabelle 0.4: Kurvenfarben

Kurvenpunkte:

Diese Eigenschaft liefert ein Array, in welchem die Anzahl der Koordinatenpaare der einzelnen Graphen enthalten ist (Tabelle 3.5). Im Gegensatz zu den vorherigen Eigenschaften kann auf diese

nur lesend zugegriffen werden. Eine Änderung des Wertes im Array über die Eigenschaft hat keinen Einfluss auf den Graphen in Chart2D.

Name der Eigenschaft	Datentyp	Standardwert
CurvePoints	int[]	0

Tabelle 0.5: Kurvenpunkte

Kurvenzähler:
Die Anzahl der in Chart2D existierenden Graphen kann über diese schreibgeschützte Eigenschaft ausgelesen werden (Tabelle 3.6). Der Zähler beginnt von null und endet bei neun. Ist kein Graph vorhanden, wird "-1" angezeigt.

Name der Eigenschaft	Datentyp	Standardwert
Counter	int	-1

Tabelle 0.6: Kurvenzähler

Spracheinstellung:
Die beschreibenden Texte der Benutzeroberfläche von Chart2D können entweder auf Deutsch oder auf Englisch angezeigt werden. Mit dieser Eigenschaft kann eine der beiden Sprachen eingestellt werden. Die voreingestellte Sprache der Komponente ist Englisch (Tabelle 3.7).

Name der Eigenschaft	Datentyp	Standardwert
Germanlanguage	bool	False

Tabelle 0.7: Spracheinstellung

1.1.2 Methoden

Die zur Verfügung stehenden Methoden von Chart2D erlauben einen Datenaustausch zwischen der eigenen Anwendung und der Komponente. Im Anschluss werden die einzelnen Methoden sowie deren Funktionalität erläutert.

Einlesen der Koordinatenpunkte:
Mit der Methode `setPoint()` wird ein Koordinatenpunkt, bestehend aus X- und Y-Wert, an einen ausgewählten Graphen übergeben. Es können maximal zehn Graphen erstellt werden, wobei eine Nummerierung beginnend von null eingehalten werden muss. Wird die Methode ein weiteres Mal aufgerufen, so wird der nächste Koordinatenpunkt hinter dem vorherigen des ausgewählten Graphen geschrieben. Folglich bedarf es wiederholter Aufrufe der Methode zur Übergabe aller Koordinatenpunkte eines Graphen an Chart2D. Die Tabelle 3.8 beinhaltet den Namen der Methode, die zu übergebenden Parameter sowie den Rückgabewert.

Name der Methode	Parameter	Rückgabewert
setPoint	int nr, double x, double y	void

Tabelle 0.8: Einlesen von Koordinatenpunkten

Starten des Zeichenvorgangs:
Nachdem alle Koordinatenpunkte einer gewünschten Anzahl von Graphen eingelesen worden sind, muss die Methode `startDrawing()` aufgerufen werden. Diese zeichnet alle bis dahin

übergebenen Graphen auf die Darstellungsfläche von Chart2D. Außerdem werden die Koordinatenpunkte dem Dateneditor hinzugefügt.

Name der Methode	Parameter	Rückgabewert
startDrawing	-	void

Tabelle 0.9: Starten des Zeichenvorgang

Autoskalieren

Mit der Methode autoscaling() werden die x- und y- Achse so gut es eben möglich ist mit einer automatischen Skalierung mit möglichst gutem Grid- Raster versehen. Die automatische Skalierung der Original- Microsoft- Chart ist einfach grauenhaft. Diese sollte deutlich besser sein. Man kann aber jederzeit seine eigene Skalierung fest programmieren (siehe oben in Achseinstellungen) oder der Enduser kann manuell nachskalieren mit der entsprechenden Taste in der Toolbar.

Werden beide y- Achsen (also auch Y2) benutzt, so wirkt diese Methode nur auf die linke y- Achse. Mit der Methode `autoscaleY2()` kann die Y2- Achse rechts autoskaliert werden. Nur leider passen dann die Gridlinien oft nicht gut zusammen. Dieses kann man versuchen zu verbessern mit der Methode `Fix_YwithY2()`.

Übrigens kann man mit der Methode `setY2(int nr, bool wert)` jeder Kurve die Y2- Achse zuweisen, nr ist die Kurvennummer 0-9 und wert bestimmt die Aktivierung der Y2- Achse.

Echtzeitzeichnen:

Die Methode `realTimeChart()` vereint die beiden Methoden `setPoint()` und `startDrawing()`, sodass ein Koordinatenpunkt direkt beim Aufruf der Methode auf die Darstellungsfläche von Chart2D gezeichnet wird (Tabelle 3.10).

Name der Methode	Parameter	Rückgabewert
realTimeChart	int nr, double x, double y	void

Tabelle 0.10: Echtzeitzeichnen

Auslesen der Koordinatenpunkte:

Chart2D beinhaltet zwei Methoden, die es ermöglichen, Koordinatenpunkte vorhandener Graphen auszulesen (Tabelle 3.11). Um alle Koordinatenpunkte eines bestimmten Graphen zu erhalten wird die Methode `getPoints()` verwendet. Diese liefert ein zweidimensionales Array mit allen X- und Y-Werten, die jeweils in einer Spalte stehen, zurück. Die zweite Methode `getPoint()` liefert hingegen an einer gewünschten Position eines Graphen den Koordinatenpunkt zurück. Der Rückgabewert ist hierbei ein einfaches Array, in welchem sich der X- und Y-Wert des ausgewählten Punktes hintereinander befinden.

Name der Methode	Parameter	Rückgabewert
getPoints	int nr	double[,]
getPoint	int nr, int position	double[]

Tabelle 0.11: Auslesen von Koordinatenpunkten

Löschen der Kurven:

Auch hier bietet Chart2D zwei unterschiedliche Methoden an. Einerseits können mit der Methode `clearAll()` alle existierenden Graphen gelöscht werden. Andererseits wird mit der Methode `clearLast()` der zuletzt hinzugefügte Graph wieder entfernt (Tabelle 3.12). Zudem werden bei

diesen Methoden auch alle betreffenden Einträge in der Legende und im Dateneditor gelöscht sowie alle Namen und Farben der Graphen zurückgesetzt.

Name der Methode	Parameter	Rückgabewert
clearAll	-	void
clearLast	-	void

Tabelle 0.12: Löschen von Kurven

Kurvennamen:

Die jeweiligen Namen der Graphen können direkt an das öffentliche String-Array `CurveNames` übergeben werden, welches für jeden der zehn Graphen einen Namen speichern kann. Mit der Methode `setPoints()` werden diese Namen an die Benutzeroberfläche übergeben. Besteht jedoch der Wunsch, die Namen nachträglich zu ändern, ist es erforderlich, den neue Name in das String-Array zu schreiben und anschließend die Methode `updateNames()` aufzurufen (Tabelle 3.13).

Name der Methode	Parameter	Rückgabewert
updateNames	-	void

Tabelle 0.13: Kurvennamen

Oszillograph- Funktion

Mit der Methode void `scopeChart(nr1,y)` lässt sich diese Komponente wie ein Digitalscope benutzen. Die Zeitachse wird automatisch durch die Uhrzeit definiert. Stellt man den Parameter XMax z.B. auf 1, so hat man eine Zeitbasis von 1 sec. Werden längere Vorgänge aufgezeichnet, so beginnt das Zeichnen wie bei einem Scope wieder neu von links. Die alte Kurve wird dabei nicht gelöscht, sondern beim Erreichen der jeweiligen Zeit der neuen Kurve erst gelöscht.

Hier sind aber nur maximal 5 Traces / Kurven möglich, da für jedes darzustellende Signal zwei gleichfarbige Kurven verbraucht werden. Also die Nummer der Kurve in Parameter nr1 darf nur zwischen 0 und 4 liegen.

Bevor man diese Methode benutzen kann, muss sie initialisiert werden mit der Methode `initializeScopeMode()`.

Beispielcode:

Konstruktor der Form:

```
public Form1()
 {
     InitializeComponent();
     chart2D1.XMax = 1;
     chart2D1.XMin = 0;
     chart2D1.YMax = 1.5;
     chart2D1.YMin = -1.5;
     chart2D1.initializeScopeMode();
 }
```

Mit Standard Timer auf 10 msec Intervall gestellt:

```
int i = 0;
Random rand = new Random();

private void timer1_Tick(object sender, EventArgs e)
{
```

```
            int noise = rand.Next(1000);
            chart2D1.scopeChart(1, Math.Sin(i * Math.PI * 4.0 / maxPoints ) + noise/3000.0);
            chart2D1.scopeChart(0, Math.Sin(i * Math.PI * 8.0 / maxPoints) + noise / 3000.0);
            i++;
        }
```

Wird der Timer aktiviert, so sieht man zwei verrauschte Sin- förmige Signale mit zwei unterschiedlichen Frequenzen.

Floating Point Convertierung

Auch in dieser Komponente können die Fleißkommazahlen sowohl mit dem Punkt als auch dem Komma als Dezimaltrenner eingegeben werden. Dazu wird die öffentliche Methode `ConvertToDouble(string text,ref double nb)` benutzt, die damit auch an anderen Stellen gerne benutzt werden kann. Damit kann man sich unabhängig machen von der Windows- Einstellung. Tausender- Trenner sind auch möglich, sofern sie mit einem Dezimaltrenner zusammen benutzt werden. Ist nur ein Tausendertrenner vorhanden, wird er als Dezimaltrenner interpretiert. In text steht der zu konvertierende String, in nb wird – falls erfolgreich konvertiert werden konnte - die Zahl zurückgegeben. Bei einem Konvertierfehler wird eine Fehlermeldung ausgegeben und die Zahl nb nicht verändert. Diese muss vorher einmal beschrieben und damit instanziiert sein (sonst geht ja bekanntlich die ref- Geschichte nicht).

1.1.3 Ereignisse

Wie zuvor erwähnt, besitzt ein `UserControl` nicht nur von `Control` geerbte Eigenschaften und Methoden, sondern auch Ereignisse. Diese legen fest, welche Reaktion auf einen bestimmten Auslöser folgt. Neben den bereits vorhandenen Ereignissen, wie zum Beispiel `Click` oder `Load`, die an dieser Stelle nicht weiter erläutert werden, wurde der Komponente Chart2D ein neues hinzugefügt. Hierbei handelt es sich um das Ereignis `CoordClick`, welches die Koordinaten X und Y der Darstellungsfläche bei einem Klick auf diese liefert. Ausgelöst wird dieses Ereignis nur, wenn auch die Koordinatenanzeige im Kontextmenü von Chart2D aktiviert ist. Der nachfolgende Quellcodeausschnitt zeigt beispielhaft das Auslesen der X- und Y-Koordinate in einer Anwendung mit eingefügter Chart2D-Komponente.

```
private void chart2D1_CoordClick(object sender,
ChartTool.CoordClickEventArgs e)
        {
            //Übergabe der Koordinaten
            double koordinateX = e.x;
            double koordinateY = e.y;
            //...
        }
```

Das Ereignis `CoordClick` ist als Standard definiert, was bedeutet, dass das Ereignis bei einem Doppelklick auf das Steuerelement im Entwurfsfenster von Visual Studio dem Quellcode hinzugefügt wird.