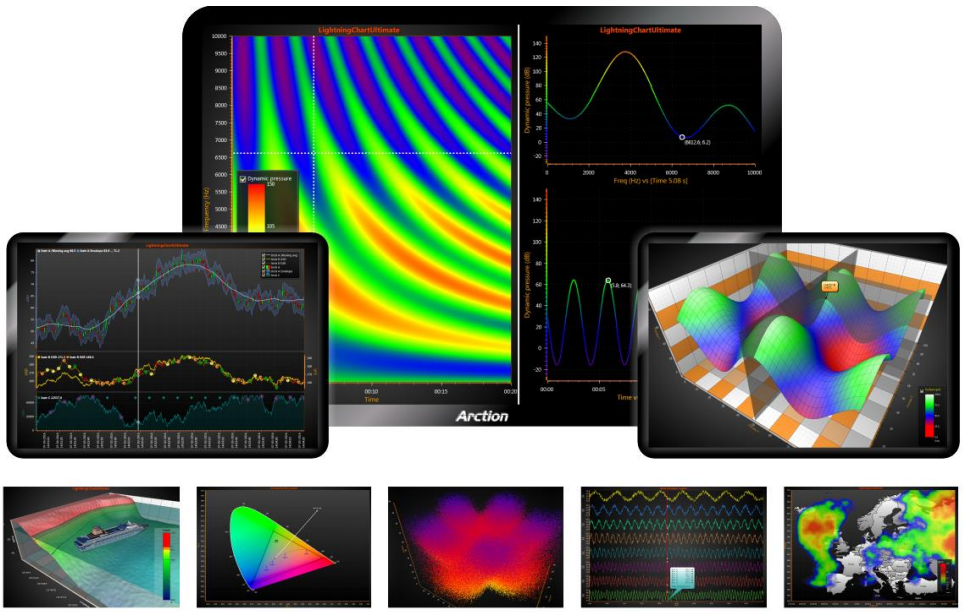


LightningChart®

사용 설명서

소개

본 문서는 ***Arction LightningChart® .NET*** 에 대한 간단한 사용 설명서이며, 필수 핵심 기능만 설명합니다. 여기서는 수백 개의 강의, 특성 또는 방법에 대해 설명하지 않습니다. 특정 기능에 대한 자세한 정보와 특정 기능에 대해 알아보시려면, 보다 포괄적인 영어 설명서를 확인하고 제공된 대화형 예제 데모 앱을 실행하여 여러 LightningChart 기능을 빠르게 미리 볼 수 있습니다. 포함된 데모 예제의 소스 코드는 코드에서 LightningChart 구성 요소를 사용하는 방법을 이해하는데 도움을 줍니다.

다른 언어로 된 사용 설명서는 LightningChart .NET 리소스 웹페이지에서 제공됩니다. (https://www.arction.com/lightningchart-net-resources/).

또한 문의 사항이 있으신 경우바로 지원팀에 문의해주십시오! (support@arction.com)

LightningChart .NET, v.10.1 적용

www.arction.com

www.lightningchart.com

목차

1. 개요

LightningChart® .NET SDK 는 WPF(*Windows Presentation Foundation*), UWP(*Universal Windows Platform*) 및 *Windows Forms* .NET 플랫폼용 데이터 시각화용 소프트웨어 구성 요소 및 도구 클래스로 구성된 Microsoft Visual Studio 의 추가 기능입니다.

Arction 구성 요소는 중요한 과학, 엔지니어링, 측정 및 거래 솔루션, 실행 성능 및 특정한 고급 기능을 위해 제공됩니다.

LightningChart 구성 요소는 느린 GDI/GDI + 또는 WPF 그래픽 API 대신 저수준 DirectX11 및 DirectX9 GPU 가속에 사용합니다. LightningChart 는 일부 가상 머신 플랫폼에서와 같이 GPU 에 액세스 할 수 없는 경우 DirectX11/DirectX10 WARP 소프트웨어 렌더링으로 대체됩니다.

1.1 차트 에디션

WPF 의 경우 LightningChart 구성 요소는 다양한 바인딩 수준 버전에서 사용할 수 있으므로 서로 다른 성능과 MVVM(모델 - 뷰- 모델 뷰) 바인딩 가능성 간의 균형을 맞출 수 있습니다. UWP 차트는 바인딩 가능한 WPF 버전을 기반으로 하며 유사한 성능, 바인딩 및 MVVM 능력을 제공합니다.

차트 에디션	속성 바인딩	시리즈 데이터 바인딩	데이터 포인트 당 바인딩	성능
WPF (바인딩 불가)	불가	불가	불가	훌륭함
WPF (바인딩 가능)	가능	가능	불가	매우 좋음
UWP (바인딩 가능)	가능	가능	불가	매우 좋음
WinForms	불가	불가	불가	최고

- WPF 의 최고의 성능과 다중 스레딩 이점을 위해서는 바인딩 불가 차트를 선택하십시오.
- WPF 바인딩 가능성과 성능 간의 적절한 균형을 유지하려면 바인딩 가능한 차트를 선택합니다. 바인딩 가능은 MVVM 디자인 패턴도 지원합니다.

바인딩 가능한 차트 API 는 LightningChart v.6 의 WPF 차트와 매우 유사하지만 컬렉션에서 생성된 개체도 포함하는 확장 속성 바인딩이 제공됩니다.

동일한 앱에서 다른 차트 에디션을 사용할 수 있으며, 성능이 중요한 작업에 바인딩 불가능한 차트를 사용하면서 바인딩 가능한 차트로 기본 차트를 만들고 다양한 속성을 바인딩할 수 있습니다. 바인딩 가능한

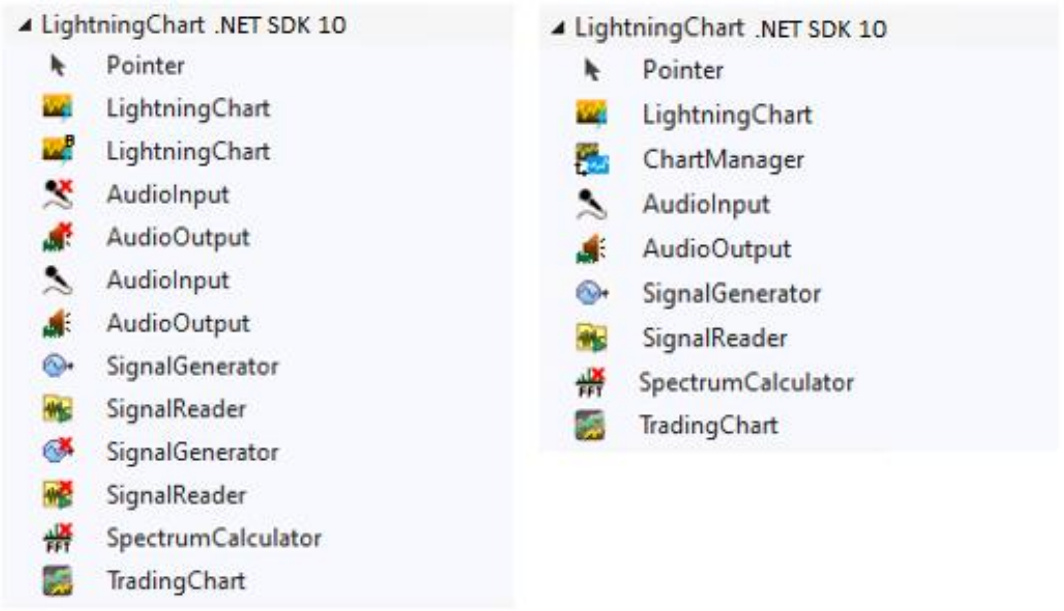
차트(예: ViewXY 축, 3D 조명)의 컬렉션 속성은 기본적으로 비어 있으며 XAML 편집기 전체를 지원합니다. 바인딩 불가 및 WinForms 컬렉션에는 기본 항목이 미리 준비되어 있습니다.

데이터 포인트 별 바인딩은 소스 코드에서만 사용할 수 있는, 완전 바인딩 가능한 WPF 에서만 지원됩니다. **소스 코드** 클라이언트는 여기에서 빌드할 수 있습니다. 8.5 는 데이터 포인트 바인딩 기능이 있는 Arction 에서 공식적으로 지원하는 마지막 버전입니다.

주의! 바인딩 불가 WPF 차트는 XAML 에서 구성할 수 없습니다. 코드 숨김에서 사용하십시오.

1.2 구성 요소

UI 가 없는 구성 요소는 **X** 로 표시됩니다.



왼쪽: WPF 도구 상자 구성 요소, 오른쪽: WinForms 도구 상자 구성 요소

차트 어셈블리

LightningChart: 차트 구성 요소입니다. 다양한 프레젠테이션에서 데이터를 시각화합니다.

아이콘 상단 모서리 B = ***바인딩 가능한 WPF 차트.***

UWP 차트: UWP 앱에서 사용 가능.

ChartManager: 여러 차트 구성 요소의 상호 운용 및 실시간 측정 메모리 관리를 제어합니다. 17 장 참조.

TradingCharts 어셈블리

TradingChart: 거래 및 금융 앱을 위한 차트 제어. 거래자 라이브러리는 LightningChart API 를 기반으로 구축되었습니다. 18 장 참조.

SignalTools 어셈블리

UI 가 없는 구성 요소는 **X** 로 표시됩니다.

AudioInput 사운드 장치에서 파형 오디오 스트림을 읽습니다. 라인 입력 또는 마이크 입력 커넥터는 사운드 장치에서 사용할 수 있는 일반적인 옵션입니다. 실시간 스트림은 다른 제어로 전달할 수 있습니다. 0 장 참조.

AudioOutput 예를 들어 스피커 또는 라인 출력으로 사운드 장치를 통해 실시간 데이터 스트림을 재생합니다. 오디오 스트림 일 필요는 없으며 모든 샘플링된 실시간 신호를 사용할 수 있습니다. 0 장 참조.

SignalGenerator 구성 가능한 여러 파형 구성 요소에서 신호를 생성합니다. 19 장 참조.

SignalReader PCM 형식의 WAV 와 같은 신호 파일에서 파형 데이터를 읽습니다. 0 장 참조.

SpectrumCalculator FFT(고속 푸리에 변환)를 사용하여 신호 데이터(시간 도메인)를 스펙트럼(주파수 도메인)으로 변환합니다. 역변환, 주파수 영역에서 시간 영역으로의 방법도 포함합니다. 0 장 참조.

1.3 명칭 공간

차트 에디션	어셈블리 명칭	명칭 공간 루트	XML 명칭 공간
WPF (바인딩 불가)	Arction.**Wpf.Charting**. LightningChart.dll	Arction.Wpf. Charting	xmlns:lcunb= "http://schemas.arction.com/ **charting**/ultimate/"
WPF (바인딩 가능)	Arction. **Wpf.ChartingMVVM**. LightningChart.dll	Arction.Wpf. ChartingMVVM	xmlns:lcusb= "http://schemas.arction.com/ ChartingMVVM/ultimate/"
UWP	Arction. **Uwp.ChartingMVVM**.	Arction.Uwp. ChartingMVVM	xmlns:lcu= "using:Arction.Uwp.ChartingMVVM"

	LightningChart.dll		
WinForms	Arction. **WinForms.Charting**. LightningChart.dll	`Arction.WinForms.Charting`	N/A

UWP 는 XML 에서 여러 명칭 공간을 사용합니다. 다음은 가장 일반적인 것입니다.

```
xmlns:lcu="using:Arction.Uwp.ChartingMVVM”
xmlns:viewxy="using:Arction.Uwp.ChartingMVVM.Views.ViewXY”
xmlns:axes="using:Arction.Uwp.ChartingMVVM.Axes”
xmlns:titles="using:Arction.Uwp.ChartingMVVM.Titles”
xmlns:seriesxy="using:Arction.Uwp.ChartingMVVM.SeriesXY”
```

ViewXY 이외의 보기를 사용하는 경우 각 보기 및 시리즈 명칭(View3D, ViewPolar 등)을 사용하십시오.

2. 설치

컴퓨터 구성이 요구 사항을 충족하는지 확인하십시오.

- DirectX 9.0c(셰이더 모델 3) 레벨 이상 그래픽 어댑터 또는 그래픽 하드웨어 없이 렌더링 하기 위한 DirectX11 호환 운영 체제. DirectX11 호환 그래픽 하드웨어 권장합니다.
- Windows Vista, 7, 8 또는 10, 32 비트 또는 64 비트, Windows Server 2008 R2 이상
- 개발 용 Visual Studio 2010-2019, 배포에는 필요하지 않습니다.
- .NET Framework v. 4.0 이상 설치

LightningChart .NET SDK v10.exe 를 마우스 오른쪽 클릭합니다. 설치 프로그램은 Visual Studio Toolbox 에 구성 요소를 설치합니다. 또한 도구 상자 제어와 관련된 도움말 파일을 설치합니다. 구성 요소 또는 도움말 설치가 실패하면 다음 섹션의 지침에 따라 수동으로 설치하십시오.

LightningChart 를 체험할 때 **SetupDownloader.exe** 가 가장 많이 사용됩니다. 이 방법을 사용하면, SDK 가 다운로드되고 설치됩니다. 즉, LightningChart .NET SDK v10.exe 를 명시적으로 실행할 필요가 없습니다.

2.1 Visual Studio Toolbox 에 수동으로 Arction 구성 요소 추가

WinForms

1. Visual Studio 를 엽니다. 새 **WinForms** 프로젝트를 생성합니다. 도구 상자를 마우스 오른쪽 버튼으로 클릭하고 **탭 추가**를 선택한 다음 "Arction"이라는 이름을 지정합니다.

2. Arction 탭을 마우스 오른쪽 버튼으로 클릭하고 **항목 선택...**을 선택합니다.
3. **도구 상자 항목 선택 창**에서 **.NET Framework** 구성 요소 선택 페이지를 클릭합니다. **찾아보기...**를 클릭하십시오.

구성 요소가 설치된 폴더 일반적으로 ***C:\Program Files (x86)\Arction\LightningChart .NET SDK v.10\LibNet4*** 에서 **Arction.WinForms.Charting.LightningChart.dll** 및 **Arction.WinForms.SignalProcessing.SignalTools.dll** 을 찾아 파일을 엽니다. 이제 도구 상자에서 구성 요소를 찾을 수 있습니다.

WPF

1. Visual Studio 를 엽니다. 새 **WPF** 프로젝트를 생성합니다. 도구 상자를 마우스 오른쪽 버튼으로 클릭하고 **탭 추가**를 선택한 다음 "Arction"이라는 이름을 지정합니다.
2. Arction 탭을 마우스 오른쪽 버튼으로 클릭하고 **항목 선택...**을 선택합니다.
3. **도구 상자 항목 선택 창**에서 **WPF 구성 요소** 선택 페이지. **찾아보기...**를 클릭하십시오.

구성 요소가 설치된 폴더 ***C:\Program Files (x86)\Arction\LightningChart .NET SDK v.10\LibNet4*** 에서 **Arction.Wpf.Charting.LightningChart.dll, Arction.Wpf.ChartingMVVM.LightningChart.dll** 및 **Arction.Wpf.SignalProcessing.SignalTools.dll** 을 찾아 파일을 엽니다. 이제 도구 상자에서 구성 요소를 찾을 수 있습니다.

2.2 Visual Studio 2010-2019 도움말 수동 설치

이 장에서는 LightningChart® .NET 도움말 콘텐츠를 수동으로 설치하는 방법에 대한 정보를 제공합니다. 이 정보는 Visual Studio 2010-2019 에 로컬 도움말 콘텐츠가 설치되어 있지 않은 경우 필요합니다. LightningChart® .NET 을 설치할 때 로컬 도움말 콘텐츠가 설치되어 있지 않으면 LightningChart® .NET 의 도움말이 설치되지 않습니다.
이 단계를 통해 Visual Studio 2010-2019 에서 LightningChart® .NET 의 도움말을 볼 수 있습니다. LightningChar 의 클래스, 속성 등에서 F1 을 누르거나 Microsoft 도움말 뷰어를 사용하여 도움말 내용을 찾아보십시오.

Visual Studio 2010

Visual Studio 2010 에 LightningChart® .NET 도움말 콘텐츠를 수동으로 설치하려면 다음 단계를 따르십시오.

1. Visual Studio 2010 을 엽니다.
2. ***도움말 -> 도움말 설정 관리***를 선택합니다.
3. 도움말 라이브러리 관리자에서 ***설정*** 링크를 클릭합니다.

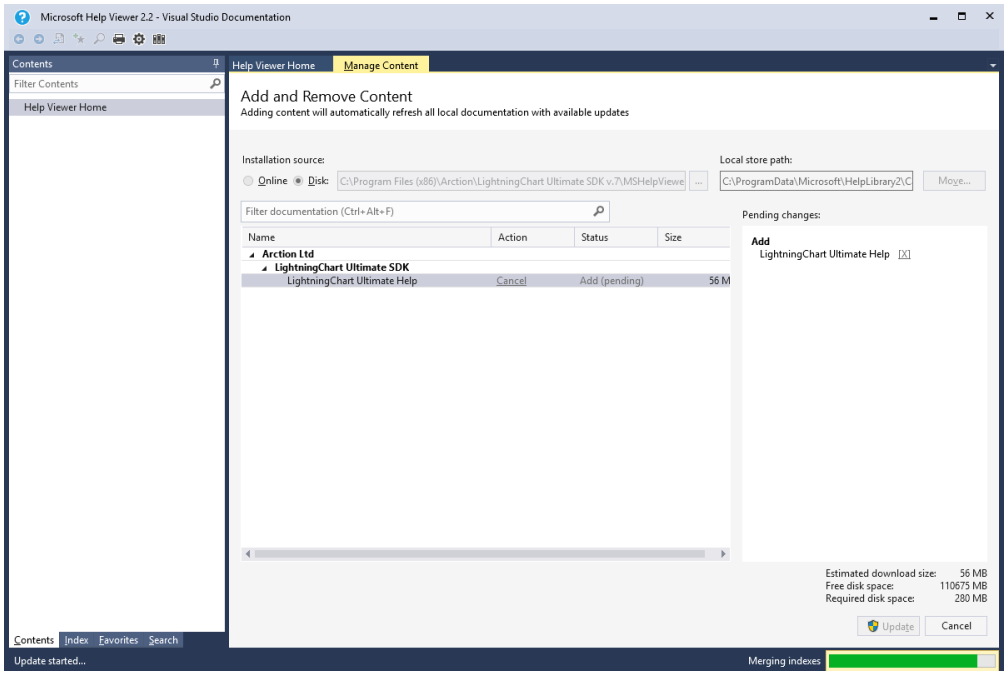
4. ***로컬 도움말 사용***이 선택되어 있는지 확인합니다.
5. ***로컬 도움말 사용***을 선택한 경우 ***취소***를 클릭하여 도움말 라이브러리 관리자로 돌아갑니다. 그렇지 않으면 ***확인***을 클릭합니다.
6. ***디스크 링크에서 콘텐츠 설치***를 클릭합니다.
7. ***찾아보기*** 버튼을 클릭하고 LightningChart® .NET 이 설치된 폴더로 이동합니다. 기본 경로는 ***C:\Program Files (x86)\Arction\LightningChart .NET SDK v.10\MSHelpViewer*** 입니다.
8. ***HelpContentSetup.msha*** 를 선택하고 ***열기*** 버튼을 클릭합니다.
9. ***다음*** 버튼을 클릭합니다.
10. LightningChart® .NET 도움말 옆에 링크 ***추가***가 있습니다. 링크를 클릭하고 ***상태*** 열 값이 ***업데이트 보류***로 변경되는지 확인합니다.
11. ***업데이트*** 버튼을 클릭합니다. 도움말 라이브러리 관리자가 계속 진행할지 묻는 메시지가 표시되면 ***예*** 버튼을 클릭하고 도움말 라이브러리 업데이트를 시작합니다.
12. 도움말 라이브러리가 업데이트된 후 ***종료*** 버튼을 클릭하여 도움말 라이브러리 관리자를 닫습니다.

Visual Studio 2012-2019

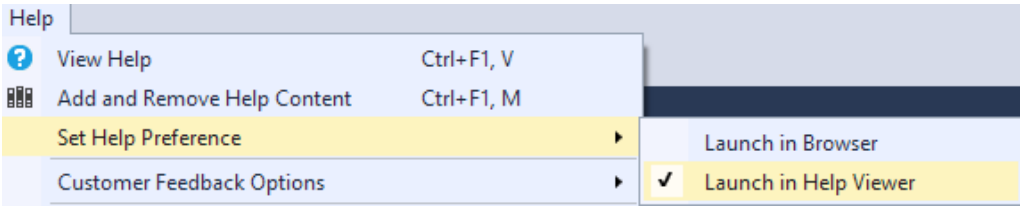
Visual Studio 2012-2019 에 LightningChart® .NET 도움말 콘텐츠를 수동으로 설치하려면 다음 단계를 따르십시오.

1. Visual Studio 2012, 2013, 2015, 2017 또는 2019 를 엽니다.
2. ***도움말-> 도움말 콘텐츠 추가 및 제거***를 선택합니다.
3. Microsoft 도움말 뷰어가 시작되면 ***콘텐츠 관리***를 선택합니다.
4. ***설치 소스***에서 ***디스크***를 선택합니다.
5. 세 개의 점이 있는 버튼을 클릭하여 파일을 찾습니다.
6. LightningChart® .NET 이 설치된 폴더로 이동합니다. 기본 경로는 ***C:\Program Files (x86)\Arction\LightningChart .NET SDK v.10\MSHelpViewer*** 입니다.
7. ***HelpContentSetup.msha*** 를 선택하고 ***열기*** 버튼을 클릭합니다.
8. LightningChart® .NET 도움말 옆에 링크 ***추가***가 있습니다. 이를 클릭하고 ***상태***열 값이 ***추가 보류***로 변경되는지 확인하십시오.

.



9. ***업데이트*** 버튼을 클릭합니다. 도움말 라이브러리 관리자가 계속할 것인지 물으면 ***예*** 버튼을 클릭하고 도움말 라이브러리 업데이트를 시작합니다.
10. 도움말 라이브러리가 업데이트된 후 Microsoft 도움말 뷰어를 닫을 수 있습니다.
11. Visual Studio 메뉴/도움말에서 ***도움말 기본 설정: 도움말 뷰어에서 시작***을 선택합니다.



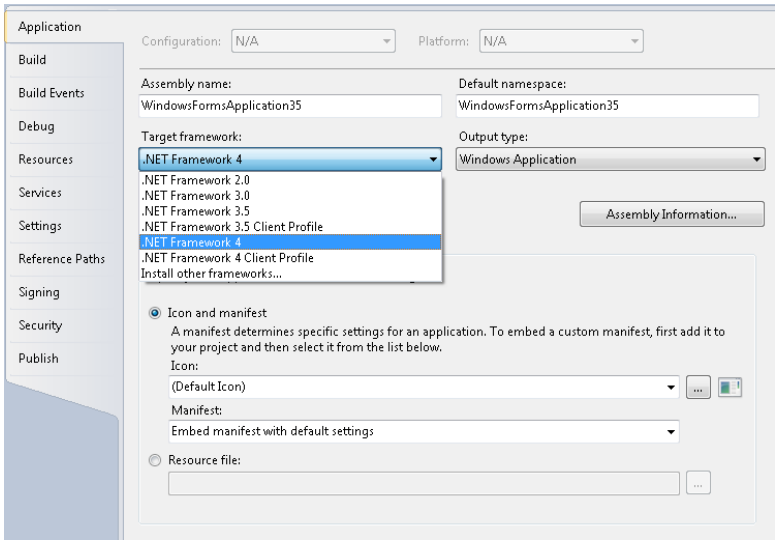
2.3 Visual Studio IntelliSense 의 코드 매개 변수 및 팁

LightningChart.dll 파일이 전역 어셈블리 캐시에서 참조되고 제어가 자동 도구 상자 설치 프로그램에 의해 설치되지 않은 경우 IntelliSense 는 LightningChart 관련 코드를 입력할 때 코드 힌트를 표시하지 않을 수 있습니다. 프로젝트의 참조 목록에서 LightningChart.dll 파일을 제거합니다. 그런 다음 설치 디렉터리 (일반적으로 ***C:\Program files (x86)\Arction\LightningChart .NET SDK v.10\LibNet4***)에서 검색하여 다시 추가합니다.

2.4 대상 프레임 워크 선택

C# 프로젝트에서는 **프로젝트 -> 속성 -> 앱 -> 대상 프레임 워크**에서 프레임 워크를 선택할 수 있습니다.

Visual Basic 프로젝트에서는 **프로젝트 -> 옵션 -> 컴파일 -> 고급 컴파일 옵션-> 대상 프레임 워크**에서 프레임 워크를 선택할 수 있습니다.



.NET Framework 4 Client Profile 또는 버전 .NET Framework 4 이상을 선택합니다. .NET Framework 4.5 이상 권장됩니다.

LightningChart® .NET SDK 제어는 올바른 .NET 프레임 워크를 선택한 경우에만 Visual Studio Toolbox 에 나타납니다.

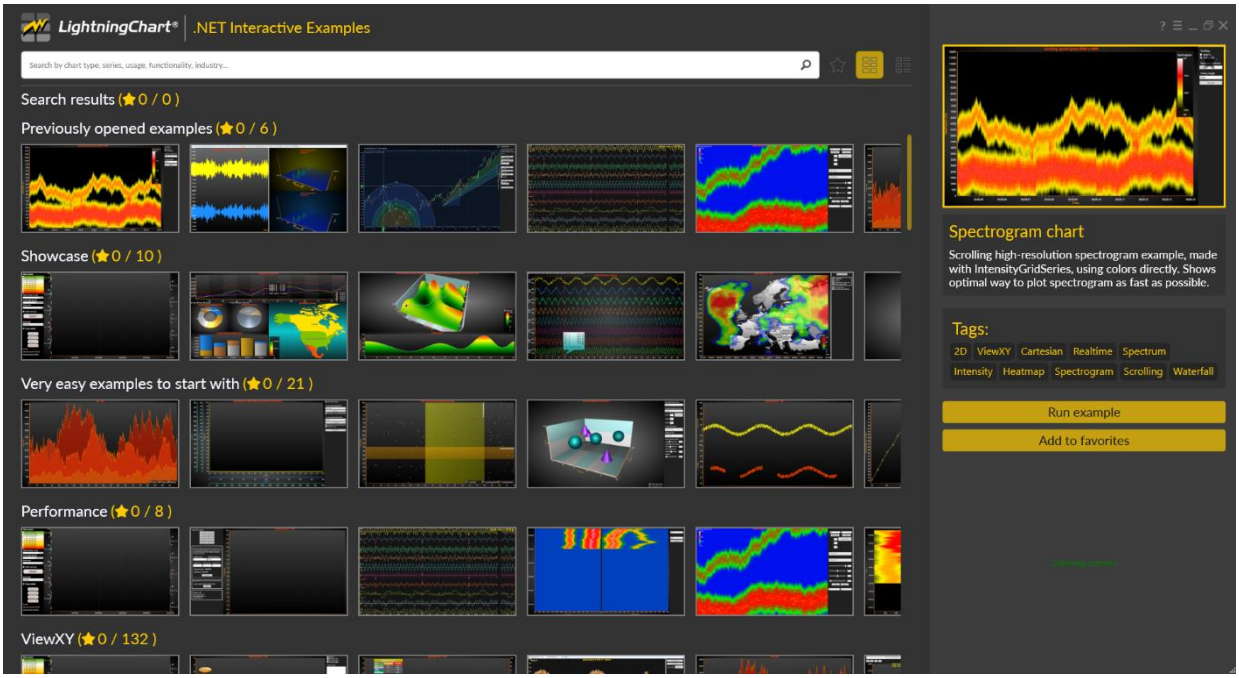
3. 개발자 센터

LightningChart .NET 버전 8.5 부터는 **LightningChart .NET SDK v10.exe** 설치를 실행할 때 Arction LightningChart .NET Dev Center 가 자동으로 설치됩니다. Dev Center 는 LightningChart® .NET 기능 및 리소스에 빠르게 액세스 할 수 있는 새로운 앱입니다. 다음 작업은 몇 번의 마우스 클릭으로 수행할 수 있습니다.

- ***대화형 예시*** 데모 앱을 엽니다
- 튜토리얼 및 사용 설명서와 같은 문서 리소스를 엽니다.
- 이메일을 통해 지원팀에 문의합니다.
- 기술 지원에 보낼 수 있는 앱 정보를 자동으로 수집합니다. 이는 종종 지원 팀이 문제를 더 빨리 해결하는데 도움이 됩니다.
- 제조 업체에 피드백을 보내는 빠른 링크를 클릭합니다.
- 라이선스 상태를 확인하고 라이선스 관리자를 열어 라이선스를 업데이트하거나 활성화합니다.
- 새 라이선스를 구매합니다.

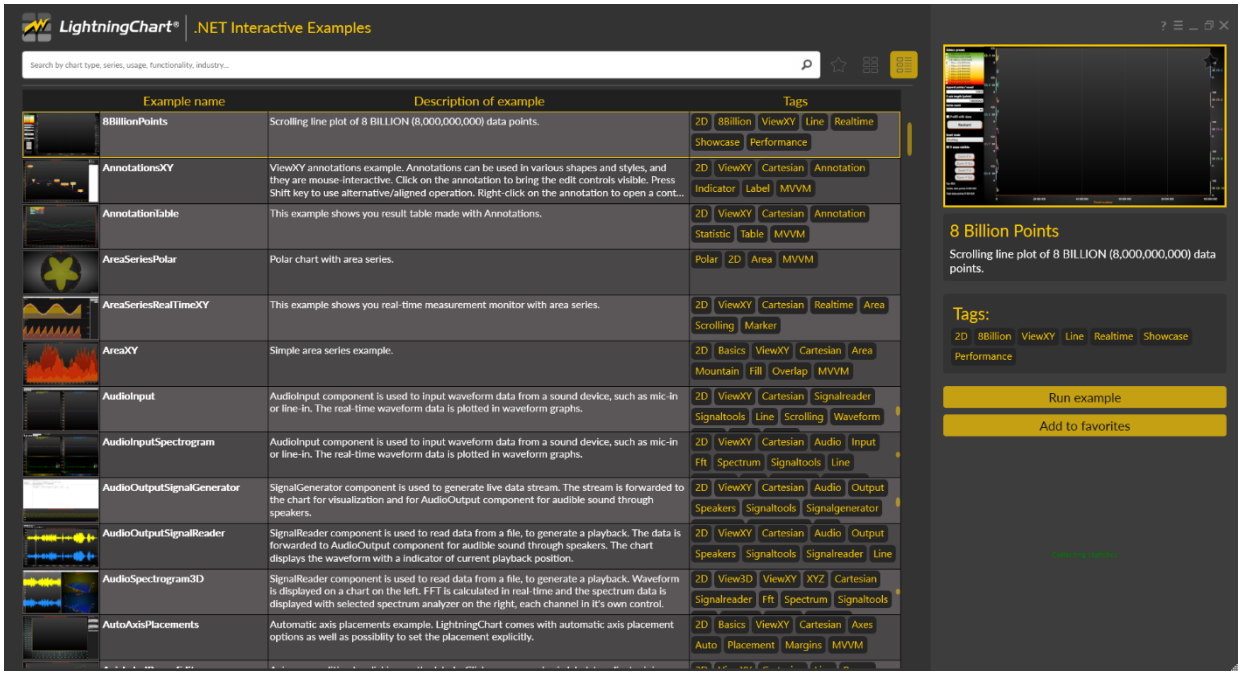
3.1 대화형 예시 열기

LightningChart ***대화형 예시***는 LightningChart 구성 요소 및 기능을 사용하는 방법을 배울 때 주요 정보 소스 중 하나입니다. ***대화형 예시***는개발자 센터에서 "대화형 예시 열기" 이미지를 클릭하거나 시작 메뉴의 바로 가기를 통해 실행할 수 있습니다. 대화형 예시에는 다양한 카테고리로 그룹화된 수많은 예시와 특정 예시를 검색할 수 있는 검색 창이 있습니다. 또한 개별 예시에는 차트 속성을 즉석에서 수정할 수 있는 ***속성*** 탭이 있습니다.

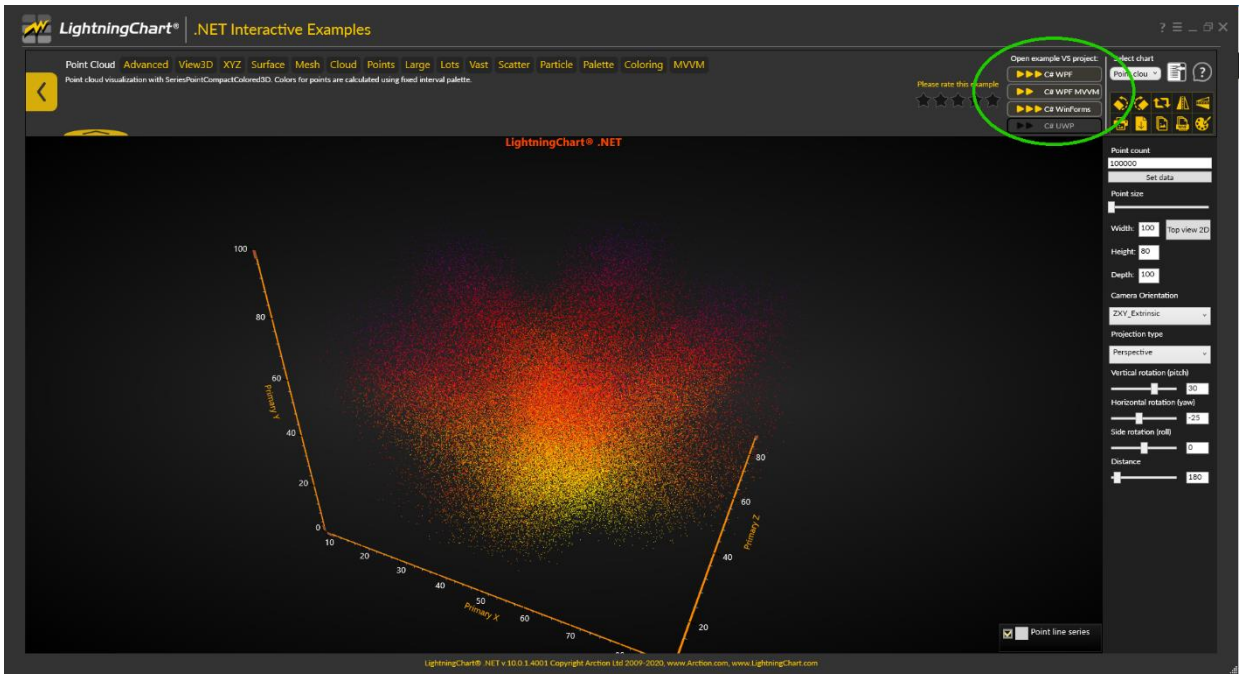


대화형 예시의 타일보기를 사용하면 범주별로 예시를 찾아볼 수 있습니다.

LightningChart .NET SDK 를 설치하면 모든 데모 예시의 소스 코드가 컴퓨터에 자동으로 추가됩니다. 대화형 예시에서 "VS project 예시 열기" 섹션을 클릭하면 Visual Studio 에서 현재 예시를 독립 실행형 프로젝트로 열고 수정할 수 있습니다.



목록보기는 분류 없이 모든 예시를 표시합니다.



예시가 열렸습니다. 강조 표시된 영역의 버튼을 통해 독립형 프로젝트로 추출할 수 있습니다.

4. 라이선스 관리

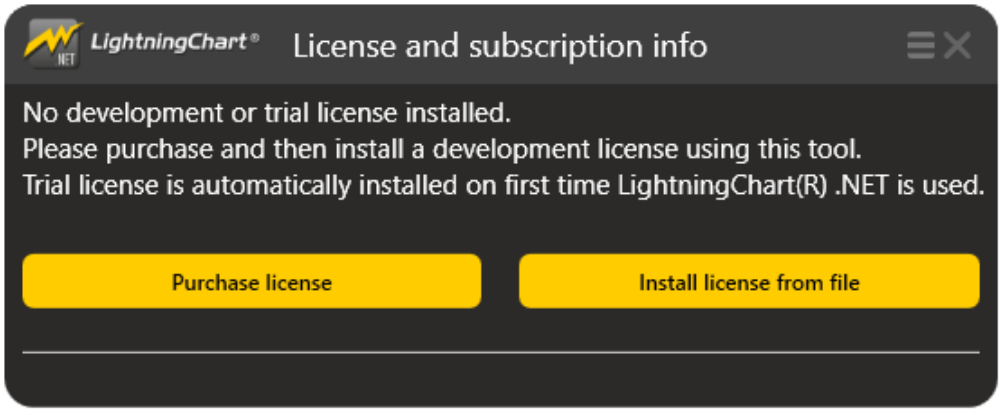
4.1 라이선스 추가

개발자 센터 또는 Windows 시작 메뉴: Programs / Arction / LightningChart .NET SDK / **License Manager** 에서 라이선스 관리자 앱을 실행하여 라이선스를 관리합니다.

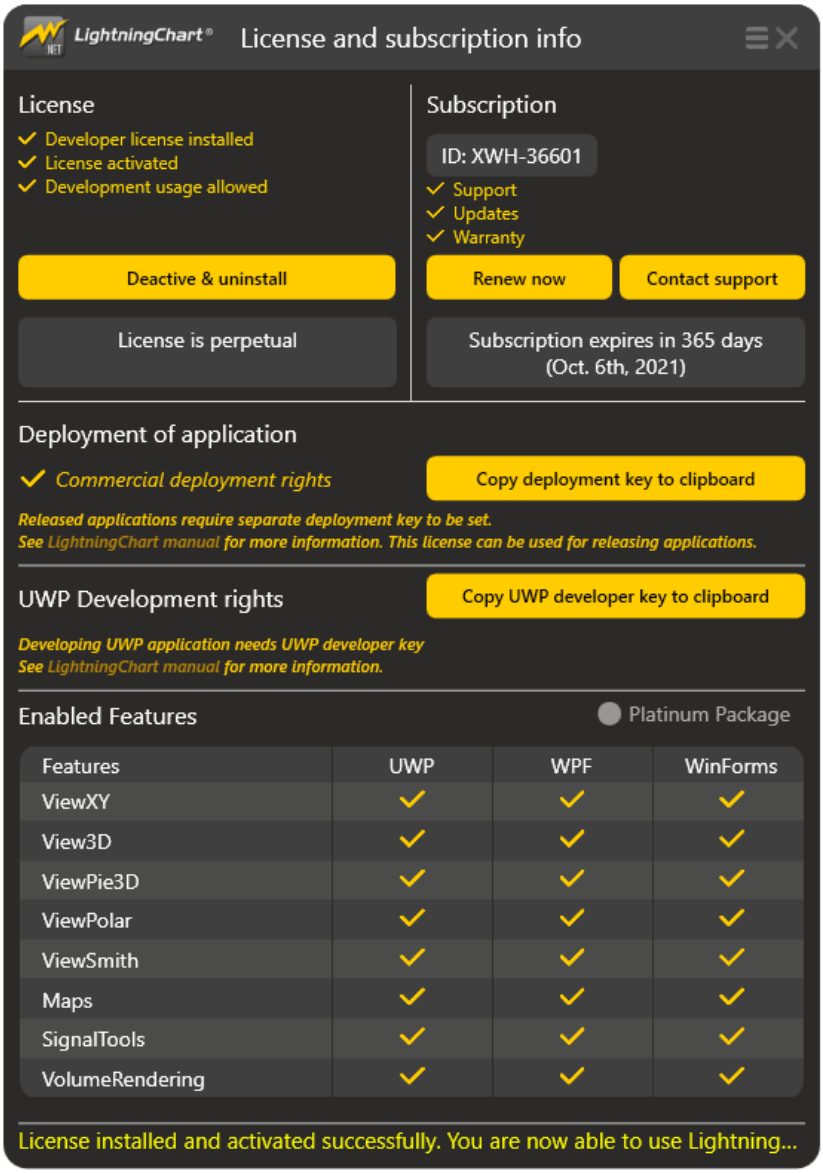
Arction 구성 요소는 라이선스라이선스 키 보호 시스템을 사용합니다. 구성 요소는 유효한 라이선스라이선스로만 사용할 수 있으며, 라이선스라이선스에는 다음 정보가 포함됩니다.

- ViewXY, View3D, ViewPie3D, Maps, ViewPolar, ViewSmith, 볼륨 렌더링, SignalTools 와 같은 활성화 된 기능
- WPF / WinForms / UWP / 모든 플랫폼
- 라이선스를 활성화할 수 있는 사용자 수 (표준으로 1 명)
- 구독 만료 날짜 (업데이트 및 지원 종료 날짜)
- 기술 지원 포괄성
- 개발자 별 라이선스 또는 유동 라이선스
- 학생 라이선스

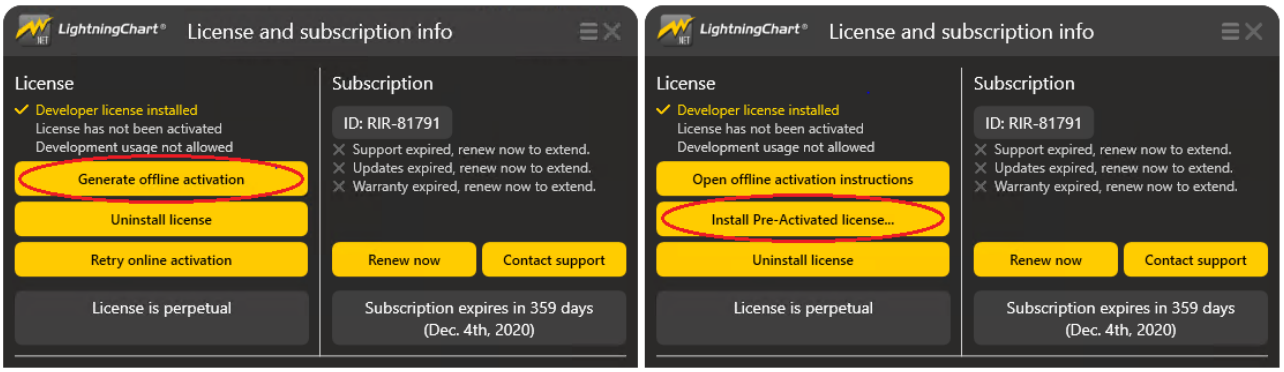
처음으로 도구 상자에서 앱으로 Arction 구성 요소를 끌기 하면 라이선스라이선스 관리자 창에 라이선스라이선스 키가 표시됩니다. **파일에서 라이선스 설치** ... 를 클릭하고 **.alf** 파일을 찾아서 받은 라이선스라이선스 파일에서 라이선스라이선스 키를 추가하십시오.



개발자별 라이선스는 라이선스 파일을 추가한 후 인터넷을 통해 Arction 라이선스 서버에 자동으로 활성화됩니다.



예를 들어 인터넷 연결을 사용할 수 없거나 연결 속도가 너무 느려서 온라인 활성화가 불가능한 경우 이메일을 통해 라이선스를 활성화할 수 있습니다. **오프라인 활성화 요청** 버튼은 해당 온라인 작업이 몇 번 실패한 후에 사용할 수 있으며, 오프라인으로 라이선스를 비활성화하는 방법도 비슷합니다.



오프라인 버튼을 클릭하면 화면에 지침이 표시됩니다. 이에 따라 Arction 라이선스 팀 (licensing@arction.com)으로 이메일을 보내십시오. Arction 은 오프라인으로 라이선스를 설치하는 방법을 제공하며, 영업일 기준 2 일 이내에 답변을 받으실 수 있습니다.

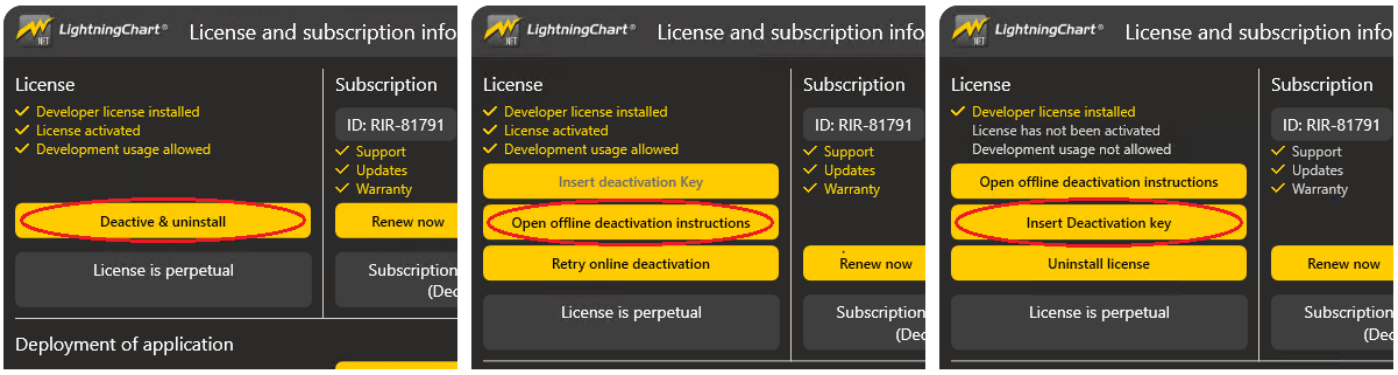
주의! 전화를 통한 활성화 / 비활성화는 키 코드에 수천 개의 문자가 포함되어 있으므로 사용할 수 없습니다.

주의! LightningChart v.8.0 부터는 LIC 형식 라이선스 키가 지원되지 않으며, ALF 라이선스가 필요합니다. ALF 라이선스를 받지 못한 경우 Arction 에 문의하십시오.

4.2 라이선스 제거

비활성화 및 제거 버튼을 사용하여 시스템에서 라이선스를 제거 할 수 있습니다. 자동 비활성화를 위해서는 온라인 연결이 필요하며, 인터넷 연결을 사용할 수 없는 경우 이전 장에서 설명한 대로 이메일을 통해 비활성화할 수 있습니다.

라이선스가 비활성화된 후 다른 컴퓨터에 설치할 수 있습니다.

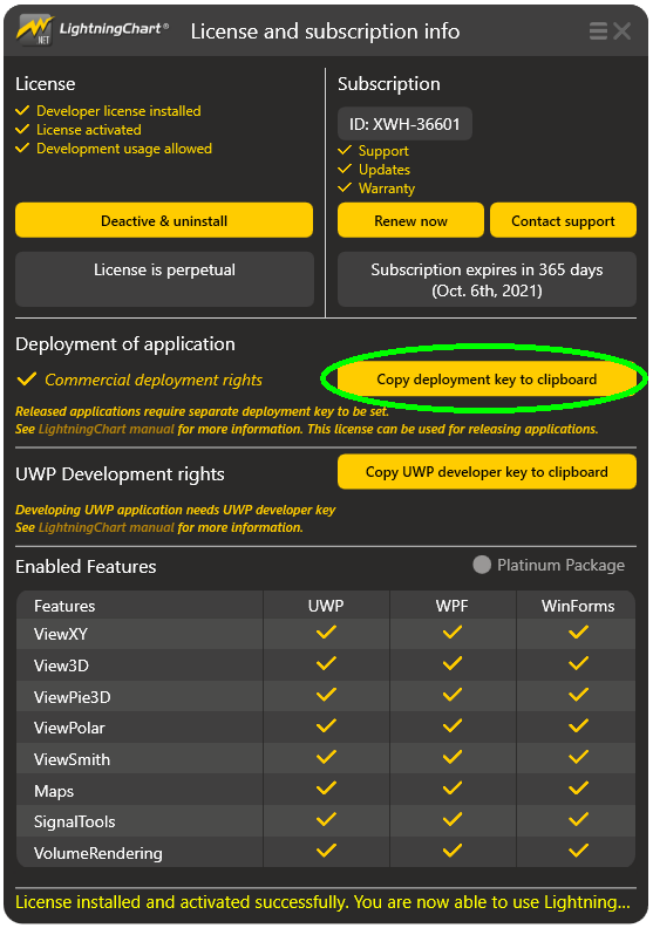


4.3 라이선스 업데이트

라이선스를 처음 설치한 후에도 계속 업데이트할 수 있습니다. 가령, 구독 기간이 연장되거나 더 나은 버전으로 업그레이드되거나 소스 코드를 구입한 경우 등. **참고로**, 라이선스는 사용자 컴퓨터에서 자동으로 업데이트되지 **않습니다**. 따라서 각 사용자는 개발자 컴퓨터의 라이선스가 최신 상태인지 확인하기 위한 조치를 취해야 하며, 이를 위해 먼저 이전 라이선스를 비활성화하고 제거해야 합니다. (이 작업을 수행하는 방법은 이전 장 "라이선스 제거" 참조) 그 후 Arction 의 고객 포털에서 새 라이선스 키(.alf 파일)를 얻습니다. 그런 다음 4.1 장 "라이선스 추가"의 지침에 따라 설치합니다.

4.4 배포 키 추출

소프트웨어가 배포된 컴퓨터에서 LightningChart 앱을 실행하려면 코드에 배포 키를 적용해야 합니다. ***배포 키를 클립 보드에 복사***버튼을 눌러 라이선스 키에서 배포 키를 추출할 수 있습니다.



4.5 앱에 배포 키 적용

코드에서는 원하는 구성 요소에 대해 정적 ***SetDeploymentKey*** 방법을 사용하며, 사용하지 않는 구성 요소에 대한 키를 설정할 필요가 없습니다. (즉, 바인딩할 수 없는 앱에서 바인딩 가능한 차트에 대한 키 설정) 구성 요소를 사용하기 전 ***SetDeploymentKey*** 방법을 다른 곳에서 호출하십시오. 이를 호출하는 가장 좋은 위치는 차트를 사용하는 클래스의 정적 생성자 또는 앱의 기본 클래스입니다.

배포에 대한 자세한 지침은 0 장을 참조하십시오.

WinForms

다음은 모든 WinForms 앱에 대해 기본적으로 생성되는 Program 클래스의 정적 생성자 방법에서 키를 적용하는 방법의 예입니다.

```
namespace WindowsFormsApplication1
{
  static class Program
  {

      static Program()
      {
         //Set Deployment Key for Arction components
         string deploymentKey = "VMalgCAAO6kO1RgiNIBJABVcG.R..Kikfd...";

         Arction.WinForms.Charting.LightningChart.SetDeploymentKey(deploymentKey);
         Arction.WinForms.SignalProcessing.SignalGenerator.SetDeploymentKey(deploymentKey);
         Arction.WinForms.SignalProcessing.AudioInput.SetDeploymentKey(deploymentKey);
         Arction.WinForms.SignalProcessing.AudioOutput.SetDeploymentKey(deploymentKey);
         Arction.WinForms.SignalProcessing.SpectrumCalculator.SetDeploymentKey(deploymentKey);
         Arction.WinForms.SignalProcessing.SignalReader.SetDeploymentKey(deploymentKey);
      }

      // Rest of the class ...
   }
}
```

WPF

다음은 *App* 클래스의 정적 생성자에서 App.xaml.cs 의 시작 부분에 키를 적용하는 방법의 예입니다.

```
using Arction.Wpf.SignalProcessing;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        static App()
        {
```

```
            // Set Deployment Key for Arction components
            string deploymentKey = "- DEPLOYMENT KEY FROM LICENSE MANAGER
            GOES HERE-";

            // Setting Deployment Key for bindable chart
            Arction.Wpf.ChartingMVVM.LightningChart
            .SetDeploymentKey(deploymentKey);

            // Setting Deployment Key for non-bindable chart
            Arction.Wpf.Charting.LightningChart
            .SetDeploymentKey(deploymentKey);

            // Setting of deployment key to other Arction components
            SignalGenerator.SetDeploymentKey(deploymentKey);
            AudioInput.SetDeploymentKey(deploymentKey);
            AudioOutput.SetDeploymentKey(deploymentKey);
            SpectrumCalculator.SetDeploymentKey(deploymentKey);
            SignalReader.SetDeploymentKey(deploymentKey);

        }
    }
```

UWP 앱에서는 개발자 키 또는 배포 키를 사용할 수 있지만 둘 다 사용할 수는 없습니다. 앱을 개발 및 디버깅할 때 개발자 키를 사용하고, 배포할 때 배포 키를 사용합니다.

주의! 앱에서 배포 키를 설정하지 않으면 LightningChart 앱이 대상 컴퓨터에서 30 일 무료체험 모드로 들어갑니다. (개발 라이선스 키가 설치되지 않은 컴퓨터에 적용됨)

4.6 개발 컴퓨터에서 배포 키로 실행

개발 라이선스가 설치된 컴퓨터에서 ***SetDeploymentKey*** 로 배포 키가 적용된 앱을 실행할 때 라이브러리는 **개발 라이선스 키의 우선순위**를 지정합니다. 배포 키에 로컬로 설치된 라이선스(예: 실버 팩)보다 더 높은 수준의 기능(예: 골드 팩)이 포함되어 있으면 사용자 또는 디버깅 혼동이 발생할 수 있으므로 개발자는이 제한 사항을 인지하고 있어야 합니다.

Arction 은 모든 라이선스가 전체 팀 내에서 동일한 유형일 것을 권장합니다.

4.7 디버거로 실행

배포 키가 올바르게 설정된 상태에서 디버거가 연결된 Visual Studio 에서 프로젝트를 실행하고 시스템에서 개발 라이선스 키를 찾을 수 없는 경우 차트는 느린 렌더링 모드로 전환되는데, 최대 FPS 는 ~ 1 이며 차트는 차트 위에 메시지 텍스트를 표시합니다.

개발자 라이선스 키 없이 LightningChart 를 사용한 직접 개발 및 디버깅은 LightningChart EULA 에 의해 금지되어 있습니다.

4.8 체험판 사용 기간

체험판은 30 일 동안 사용할 수 있습니다. 그 후에 제품을 계속 사용하려면 라이선스를 구입해야 합니다. 체험판 라이선스로 빌드된 모든 프로젝트는 적절한 라이선스로 업데이트한 후에도 작동하지만, 평가판 라이선스로 빌드된 차트 앱을 실행할 때 평가판 nag 메시지가 표시됩니다.

4.9 유동 라이선스

유동 라이선스는 수많은 컴퓨터에 설치할 수 있으며, 동시 개발자 수는 Arction 에서 구성합니다. 구매 한 동시 사용자 수만 LightningChart 로 동시에 개발할 수 있으며, 개발자가 LightningChart 개발을 완료한 후 다른 개발자가 사용을 시작할 수 있을 때까지 약 10-15 분의 시간제한이 있습니다.

배포 키는 개발자 별 라이선스와 유사하게 설정해야 합니다. 유동 라이선스는 기본적으로 Arction Licensing Server 에 의해 제어되며, 개발 중에는 지속적인 인터넷 연결이 필요합니다.

고객 측 유동 라이선스 관리자를 사용할 수 있습니다. 개발 컴퓨터는 LAN 을 통해 고객의 조직에서 실행되는 서비스에 연결되며, Arction 또는 다른 당사자와의 온라인 커뮤니케이션은 이루어지지 않습니다. 라이선스를 통해 Arction 은 관리자 서비스 및 유동 라이선스 설치에 대한 별도의 지침을 제공합니다.

5. LightningChart 어셈블리 배포 / 배급

5.1 참조 어셈블리

실행 폴더 옆, 전역 어셈블리 캐시 또는 .NET 어셈블리 확인 시스템에서 찾을 수 있는 다른 폴더와 함께 Arction .dll 파일을 전달합니다. LightningChart 는 ***ClickOnce*** 배포도 지원합니다.

WinForms:

- Arction.WinForms.Charting.LightningChart.dll
- Arction.Licensing.dll
- Arction.DirectX.dll
- Arction.RenderingDefinitions.dll
- Arction.RenderingEngine.dll
- Arction.RenderingEngine9.dll
- Arction.RenderingEngine11.dll
- Arction.DirectXInit.dll
- Arction.DirectXFiles.dll

SignalTools 를 사용하는 경우

- Arction.WinForms.SignalProcessing.SignalTools.dll
- Arction.MathCore.dll

WPF:

- Arction.Wpf.Charting.LightningChart.dll **(바인딩 불가 WPF 차트)**
- Arction.Wpf.ChartingMVVM.LightningChart.dll **(바인딩 가능 WPF 차트)**
- Arction.Licensing.dll
- Arction.DirectX.dll
- Arction.RenderingDefinitions.dll
- Arction.RenderingEngine.dll
- Arction.RenderingEngine9.dll
- Arction.RenderingEngine11.dll
- Arction.DirectXInit.dll
- Arction.DirectXFiles.dll

SignalTools 를 사용하는 경우

- Arction.Wpf.SignalProcessing.SignalTools.dll
- Arction.MathCore.dll

UWP:

- Arction.Uwp.ChartingMVVM.LightningChart.dll
- Arction.Uwp.RenderingDefinitions.dll
- Arction.Uwp.RenderingEngine.dll
- Arction.Uwp.RenderingEngineBase.dll
- Arction.Uwp.Licensing.dll
- SharpDX.D3DCompiler.dll
- SharpDX.Direct2D1.dll
- SharpDX.Direct3D11.dll
- SharpDX.dll
- SharpDX.DXGI.dll
- SharpDX.Mathematics.dll
- UwpAttributes.dll

모든 구성 요소에 대해 정적 ***SetDeploymentKey*** 방법을 사용해야 합니다. 그렇지 않으면 차트가 체험판 모드로 들어가 30 일 동안만 작동하며 체험판 메시지가 표시됩니다. *DeploymentKey 생성 및 자세한 라이선스 키 관리는* ***Error! Reference source not found.****. 장을 참조하십시오.*

5.2 기타 사항

LightningChart 라이선스 키가 .NET 디스어셈블러 도구에 표시되지 않도록 앱 코드를 **숨겨야 합니다**. 라이선스 키가 유출되면 라이선스 종료, 법적 조치 및 손해 배상 청구로 이어질 수 있습니다.

LightningChart 소스 코드 구독자는 LightningChart 라이브러리의 소스 코드에 액세스 할 수 있습니다. Arction 의 지적 재산권 및 코드 유출을 방지하려면 LightningChart 소스 코드에서 어셈블리 빌드를 **숨겨야 합니다**. **숨겨지지** 않은 LightningChart 라이브러리를 배포하는 것은 EULA 를 위반하는 것이며 라이선스 종료, 법적 조치 및 손해 배상 청구로 이어질 수 있습니다.

다음 XML 파일의 배포는 금지되어 있습니다.

- Arction.WinForms.Charting.LightningChart.xml
- Arction.Wpf.BindableCharting.LightningChart.xml
- Arction.Wpf.Charting.LightningChart.xml
- Arction.Wpf.ChartingMVVM.LightningChart.xml
- Arction.Wpf.SignalProcessing.SignalTools.xml
- Arction.WinForms.SignalProcessing.SignalTools.xml

Arction 에서 제공하는 파일은 앱 개발을 돕기 위한 것이며, 주로 코드 매개 변수 및 속성 팁을 표시하는 데 사용됩니다. 소스 코드에서 LightningChart 어셈블리를 다시 빌드할 때 위에서 언급 한 XML 파일이 배포되지

않았는지 확인하십시오. .NET 디스어셈블러 및 리버스 엔지니어링 앱에 대한 정보를 공개하므로 **배포하는 것은 엄격히 금지되어 있습니다**.

6. LightningChart 구성 요소

6.1 LightningChart® .NET 라이브러리 사용

LightningChart® .NET 구성 요소를 사용하려면 Arction .dll 파일을 참조에 추가해야 하며, 이를 설치 폴더에서 찾을 수 있습니다. 앱을 개발할 때 다음 어셈블리가 필요합니다.

Winforms:	Arction.WinForms.Charting.LightningChart.dll.
WPF 바인딩 불가:	Arction.Wpf.Charting.LightningChart.dll Arction.DirectX.dll Arction.RenderingDefinitions.dll
WPF 바인딩 가능:	Arction.Wpf.ChartingMVVM.LightningChart.dll Arction.DirectX.dll Arction.RenderingDefinitions.dll
UWP 차트:	Arction.Uwp.ChartingMVVM.LightningChart.dll Arction.Uwp.RenderingDefinitions.dll Arction.Uwp.RenderingEngineBase.dll
SignalTools 를 사용할 경우:	Arction.WinForms.SignalProcessing.SignalTools.dll or Arction.Wpf.SignalProcessing.SignalTools.dll or Arction.Uwp.SignalProcessing.SignalTools.dll

위의 참조가 추가되면 프로젝트 빌드가 필요한 모든 어셈블리를 출력 폴더에 자동으로 복사합니다. 28 장에서는 LightningChart 앱을 배포할 때 필요한 어셈블리를 보여줍니다.

Arction.DirectXFiles.dll 은 초기화 시간을 증가시킬 수 있는 대용량 파일이므로 자동으로 참조로 포함되지 않으며, 시스템에 이미 올바른 DirectX 어셈블리가 없는 경우에만 필요합니다. Arction.DirectXInit.dll 루틴은 기존 dll 을 확인하고 필요할 때 로드합니다. 한 번 로드되면 나중에 LightningChart 에서 액세스 할 수 있는 Windows 임시 폴더에 DirectX-dll 을 기록하여 초기화를 빠르게 합니다.

Arction.DirectXFiles.dll 을 참조로 포함하지 않고 대신 exe 옆에 복사하는 것이 좋습니다.

6.2 코드에 차드 생성

LightningChart 구성 요소는 도구 상자에서 드래그하거나 코드에서 완전히 생성하여 추가할 수 있으며, 코드에서 차트 개체를 만들면 버전 업데이트가 더 쉬워진다는 장점이 있습니다. 또한 일부 (비) 직렬화 관련 문제를 피할 수 있습니다.

다음은 코드 숨김(.xaml.cs -file)에서 WPF 바인딩을 지원하지 않는 차트를 생성하는 한 가지 방법을 보여줍니다.

```
using Arction.Wpf.Charting;

namespace ExampleProject
{
    public partial class ExampleApp : Page
    {
        private LightningChart _chart = null;

        public ExampleApp()
        {
            InitializeComponent();

            CreateChart();
        }

        private void CreateChart()
        {
            _chart = new LightningChart();

            // Chart control into the parent container.
            (Content as Grid).Children.Add(_chart);

            // Disable rendering until the whole chart is set up correctly.
            _chart.BeginUpdate();

            // Configure chart here.

            // Allow rendering the chart.
            _chart.EndUpdate();
        }
    }
}
```

6.3 도구 상자에서 차트 추가

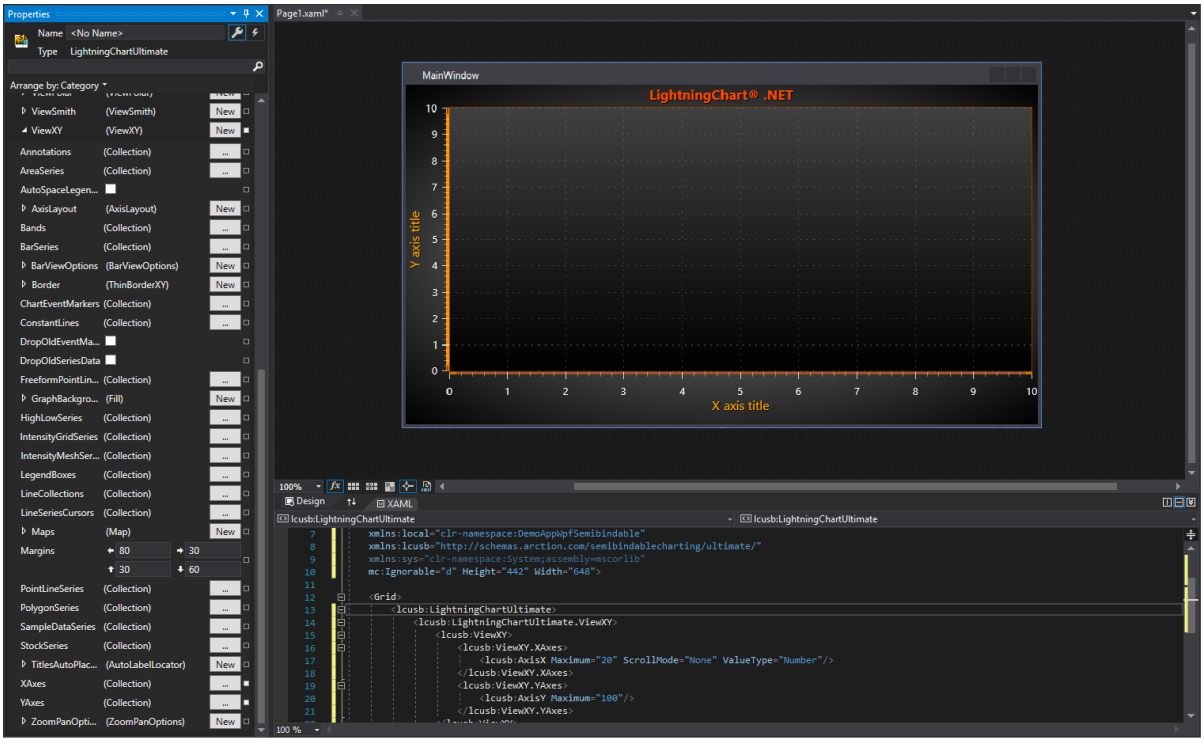
도구 상자의 ***LightningChart*** 제어를 폼이나 창에 추가합니다. 차트가 양식에 나타나고 해당 속성이 ***속성 창***에 표시되며, 바인딩 가능한 WPF 를 사용하는 경우 XAML 편집기는 차트 기본 속성에 대한 내용과 수정 사항을 표시합니다.

차트의 속성은 자유롭게 수정할 수 있으며, 또한 새 시리즈 및 기타 개체를 컬렉션에 삽입할 수 있습니다. 시리즈 데이터 포인트는 코드로 제공되어야 합니다. 차트 기본 레벨의 이벤트 핸들러는 속성 그리드로 할당할 수 있으며, 컬렉션에 추가된 개체의 경우 코드에서 이벤트 처리기를 할당해야 합니다.

버전 업데이트와 관련된 모범 사례

차트 속성 데이터는 Visual Studio 프로젝트의 ***.resx*** 파일로 직렬화됩니다. LightningChart API 는 ***.resx*** 파일에 존재하는 새 버전에 대해 호환되지 않는 직렬화로 이어질 수 있는 버전 업데이트로 약간 변경되는 경향이 있습니다.

더 쉬운 버전 업데이트를 위해 차트 개체를 만들고 모든 시리즈, 이벤트 핸들러 등을 코드에 추가하는 것이 좋습니다. 그러면 프로젝트가 올바르게 로드되고 컴파일 시간에 가능한 오류가 표시되므로 .resx 파일을 수정하는 것보다 쉽게 수정할 수 있습니다. .resx 파일을 사용하면 일부 속성 정의가 손실될 수 있지만 코드에서는 항상 지정됩니다.



6.4 UWP 프로젝트 생성

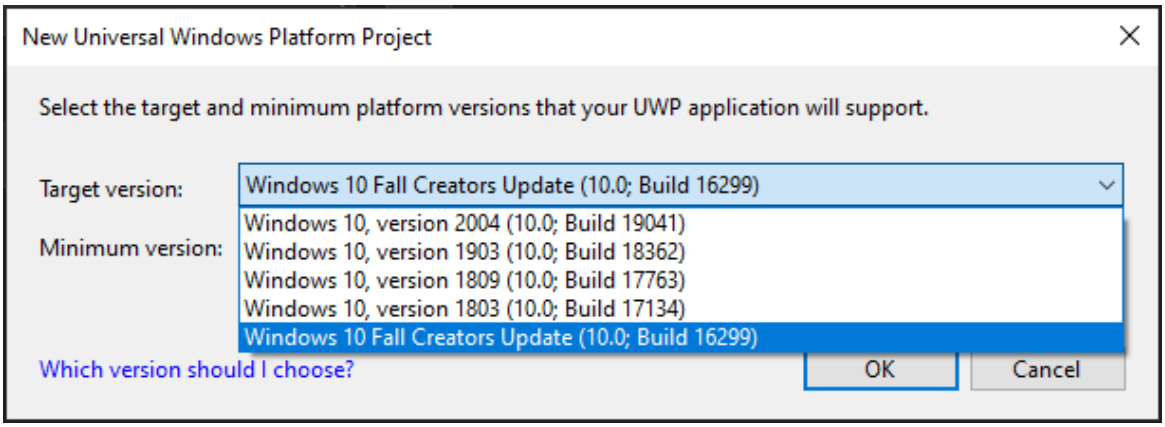
UWP 에서 LightningChart 를 사용하면 바인딩 및 MVVM 기능이 비슷하므로 바인딩 가능한 WPF 차트와 비슷하게 작동합니다. 바인딩 가능한 WPF 차트와 마찬가지로 UWP 차트의 컬렉션 속성(예 : ViewXY 축, 3D 조명)은 기본적으로 비어 있습니다. LightningChart 를 사용하여 UWP 앱을 개발하려면 Windows 10 및 Visual Studio 2017 또는 2019 가 필요합니다. UniversalWindowsPlatform 개발 워크로드는 다음을 포함하여 Visual Studio 에도 설치되어야 합니다.

- Microsoft.NETCore.UniversalWindowsPlatform: Microsoft.NETCore.UniversalWindowsPlatform : 6.2.8 이상 (Nuget 패키지).
- Microsoft.Toolkit.Uwp: v 4.0.0 이상, 6.0.0 이상 추천. 최신 툴킷은 이전 대상 버전과 호환되지 않을 수 있습니다.

UWP 앱 생성

LightningChart 를 활용하여 UWP 앱을 생성하려면 다음 단계를 따르십시오.

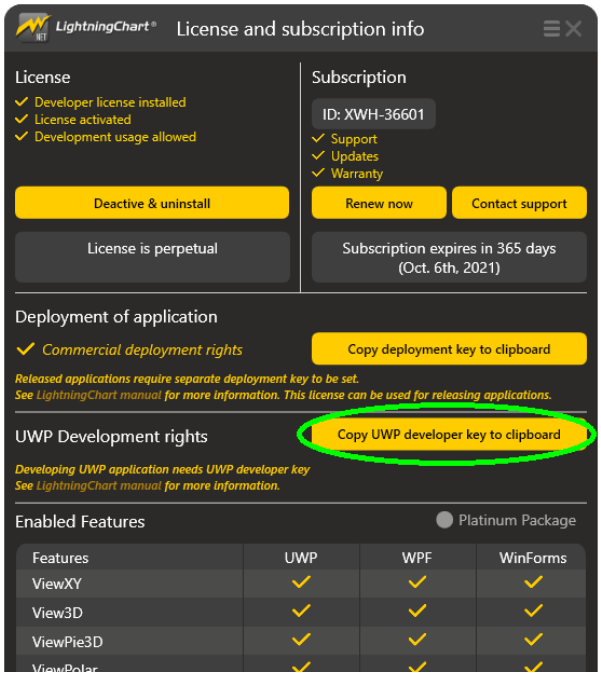
1. Visual Studio 로 새 프로젝트를 만듭니다. ***빈 앱(.UniversalWindows)***을 선택합니다.
2. 프로젝트에 이름과 파일 위치를 지정하십시오.
3. 프로젝트의 ***대상*** 및 ***최소 버전***을 설정합니다. 사용 가능한 버전은 컴퓨터에 설치된 SDK 에 따라 다릅니다. 자세한 내용은 Microsoft 문서 (https://docs.microsoft.com/en-us/windows/uwp/updates-and-versions/choose-a-uwp-version)를 참조하십시오. 버전 16299 이상을 권장하며, 나중에 프로젝트 -> 속성을 통해 변경할 수 있습니다.



4. 참조에 LightningChart 어셈블리를 추가합니다: ***Arction.Uwp.ChartingMVVM.dll***, ***Arction.Uwp.RenderingDefinitions.dll*** 및 ***Arction.Uwp.RenderingEngineBase.dll***. 동일한 UWP

어셈블리가 x86, x64, Arm 및 Arm64 플랫폼에서 작동합니다. 프로젝트를 마우스 오른쪽 버튼으로 클릭하고 속성 -> 빌드 -> 플랫폼 대상을 선택하여 대상 플랫폼을 변경할 수 있습니다.

5. 프로젝트에 **Microsoft.Toolkit.Uwp** NuGet 패키지를 설치합니다. 버전 6.0 이상 권장.
6. UWP 를 사용하려면 앱에서 개발자 키를 설정해야 합니다. "UWP 개발자 키를 클립 보드에 복사"버튼을 통해 LicenseManager 에서 키를 추출한 다음 LightningChart.SetUwpDeveloperKey () 메서드를 사용하여 App.Xaml.cs 파일에 설정합니다.




```
using Arction.Uwp.ChartingMVVM;

namespace ExampleProject
{
    /// <summary>
    /// Provides application-specific behavior to supplement the default Application class.
    /// </summary>
    1 reference
    sealed partial class App : Application
    {
        /// <summary>
        /// Initializes the singleton application object.  This is the first line of authored code
        /// executed, and as such is the logical equivalent of main() or WinMain().
        /// </summary>
        0 references
        public App()
        {
            LightningChart.SetUwpDeveloperKey("Copy developer key here");

            this.InitializeComponent();
            this.Suspending += OnSuspending;
        }
```

7. 이제 코드에서 또는 도구 상자에서 끌어서 LightningChart 구성 요소를 만들 수 있습니다.

8. 앱을 빌드, 배포 및 실행합니다. 앱이 실행되지 않는 경우(예: "Windows Store 앱 활성화..."오류로 인해) 대상 및 최소 버전을 변경하는 것이 도움이 됩니다.
9. UWP 앱을 다른 컴퓨터에 배포할 때 배포 키를 적용해야 합니다(4 장 4 절 참조). 배포 키는 개발 키와 함께 사용할 수 없습니다. 따라서 배포하기 전에 개발자 키 설정을 제거하십시오.

6.5 Windows 유형, WPF 및 UWP 32 의 차이점

Windows Forms 와 WPF 간의 속성 트리와 개체 모델은 차트 범주와 관련하여 거의 동일합니다. 주요 차이점은 다음과 같습니다.

	Windows 유형	WPF	UWP
렌더링 옵션 속성	RenderOptions	ChartRenderOptions	ChartRenderOptions
배경 채우기 속성	Background	ChartBackground	ChartBackground
글꼴	System.Drawing.Font	Arction.WPF.LightningChart.WPFFont	Arction.Uwp.ChartingMVVM.UwpFont
색깔	System.Drawing.Color	System.Windows.Media.Color	Windows.UI.Color

6.6 LightningChart 보기

LightningChart 에는 다음과 같은 기본 보기가 있습니다.

- ViewXY (0 장 참고)

- View3D (0 장 참고)

- ViewPie3D (0 장 참고)

- ViewPolar (0 장 참고)

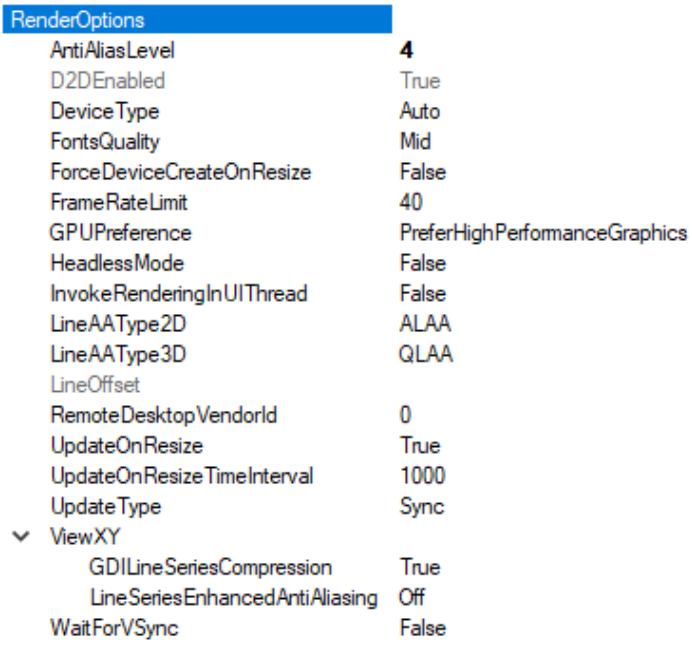
- ViewSmith (0 장 참고)

눈에 보이는 보기는 ActiveView 속성을 설정하여 변경할 수 있습니다. 기본 보기는 ViewXY 입니다.

```
// Set 3D as the visible view
chart.ActiveView = ActiveView.View3D;
```

6.7 외관/성능 설정 구성

ChartRenderOptions (*WinForms* ***의 RenderOptions)***에는 모양 및 성능을 구성하기 위한 속성이 포함되어 있습니다.



장치 유형

```
// Changing the rendering device in code
chart.ChartRenderOptions.DeviceType = RendererDeviceType.Auto;
```

Auto 는 AutoPreferD11 옵션의 별칭이며 기본 설정입니다.

AutoPreferD9 는 DirectX9 하드웨어 렌더링을 선호하며 가용성에 따라 HW9-> HW11-> SW11-> SW9 순서로 장치를 자동으로 선택합니다. 하드웨어를 사용할 수 없을 때 WARP (SW11) 소프트웨어 렌더링으로 대체됩니다.

AutoPreferD11 은 DirectX11 하드웨어 렌더링을 선호하며 가용성에 따라 HW11-> HW9-> SW11-> SW9 순서로 장치를 자동으로 선택합니다. 하드웨어를 사용할 수 없을 때 WARP (SW11) 소프트웨어 렌더링으로 대체되며, **일반적인 고성능 및 최상의 외관 설정으로 사용합니다**. 외관은 DirectX9 렌더러보다 낮습니다.

HardwareOnlyD9 는 하드웨어 9 렌더링만 사용합니다.

HardwareOnlyD11 은 하드웨어 11 렌더링만 사용합니다.

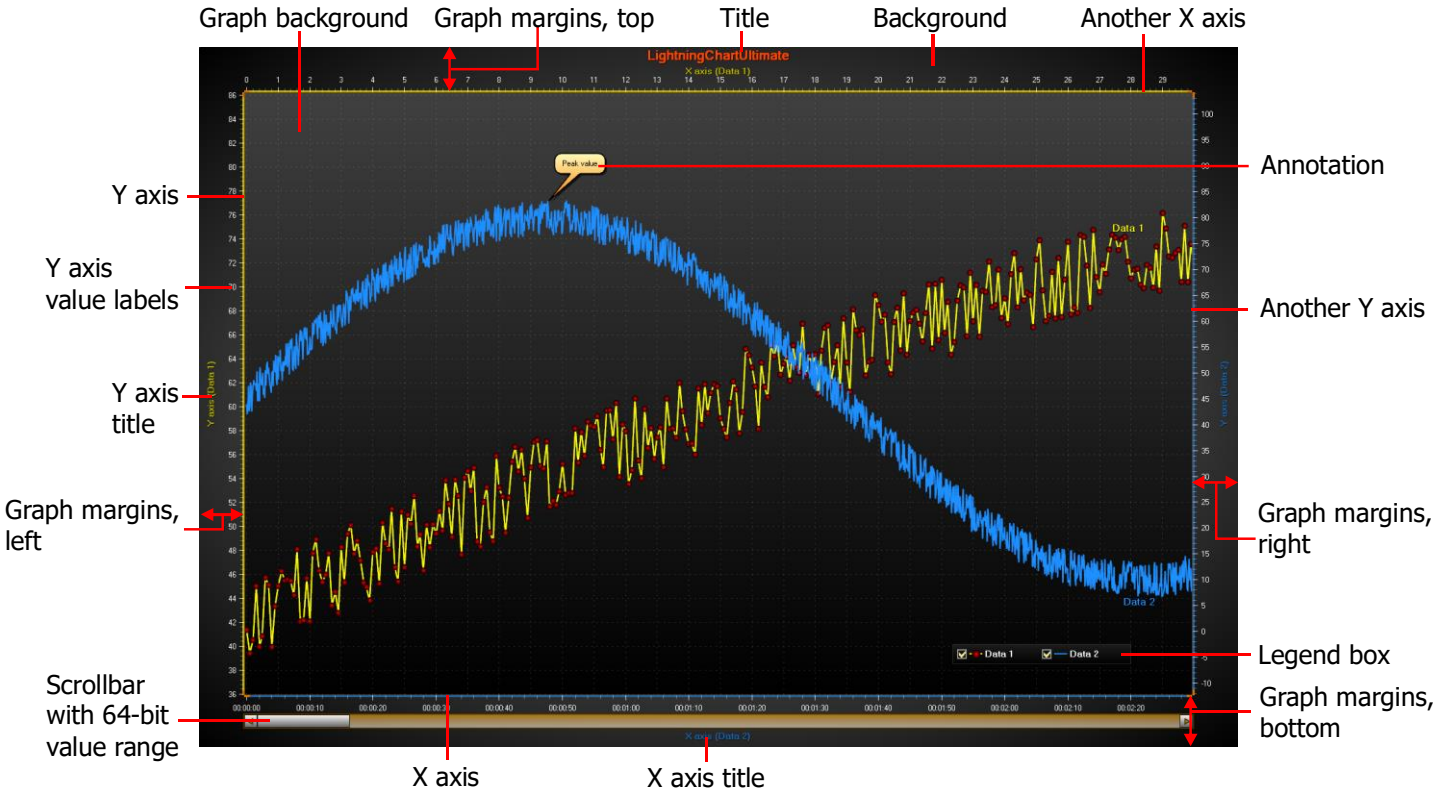
SoftwareOnlyD11 은 DirectX11 WARP 를 사용합니다. DirectX9 참조 래스터 라이저와 비교할 때 매우 빠르지만 하드웨어 옵션보다 느립니다.)

SoftwareOnlyD9 는 DirectX9 참조 레스터라이저를 사용합니다. (매우 느림)

차트가 숨겨져 있거나 백그라운드에서 비활성화된 경우 **없음**, **장치 유형을없음**으로 설정하면 그래픽 리소스가 다른 차트에 해제됩니다.

7. ViewXY

ViewXY 를 사용하면 포인트 라인 시리즈, 영역 시리즈, 하이 로우 시리즈, 히트 맵, 막대 시리즈, 밴드, 라인 시리즈 커서 및 지리적 맵과 같은 다양한 2 차원 시리즈를 데카르트, XY 그래프 형식으로 표시할 수 있습니다. 계열은 X 및 Y 축에 바인딩되며 할당된 축의 값 범위를 사용합니다.



7.1 축 레이아웃

축 배치, 자동 여백 등을 조정하는 일반 속성은 ***ViewXY.AxisLayout*** 속성 및 하위 속성에서 찾을 수 있습니다.

XAxisAutoPlacement 는 X 축이 수직으로 배치되는 방식을 제어합니다.

```
chart.ViewXY.AxisLayout.XAxisAutoPlacement = XAxisAutoPlacement.AllTop;
```

각각 ***YAxisAutoPlacement*** 는 Y 축이 수평으로 배치되는 방식을 제어합니다.

정의된 Y 축이 여러 개인 경우 ***레이어, 스택 및 세그먼트***의 세 가지 방법으로 수직으로 정렬할 수 있습니다. ***ViewXY.AxisLayout.YAxesLayout*** 속성으로 선택할 수 있습니다.

레이어

레이어 보기에서 모든 Y 축은 그래프 상단에서 시작하여 그래프 하단까지 늘어납니다. 여기에 바인딩된 축과 계열은 동일한 수직 공간을 공유합니다.

스택

스택 보기에서 각 Y 축은 고유 한 수직 공간을 갖습니다. 모든 Y 축의 높이는 동일합니다.

세그먼트

분할 보기에서 세로 공간은 ***세그먼트*** 간에 나뉩니다. 각 세그먼트에는 여러 개의 Y 축이 포함될 수 있습니다. 각 세그먼트의 관계형 높이는 ***높이*** 속성을 통해 설정할 수 있으며 세그먼트 내의 모든 Y 축은 세그먼트의 높이를 가져옵니다.

AxisLayout.Segments 컬렉션에서 ***세그먼트***를 만들어야 합니다. 먼저 추가된 세그먼트가 차트 하단에 배치됩니다. ***yAxis.SegmentIndex*** 속성을 설정하여 Y 축에 세그먼트를 할당할 수 있습니다. ***SegmentIndex*** 는 ***AxisLayout.Segments*** 컬렉션의 인덱스입니다.

7.1.1 범위 설정

최소 및 ***최대*** 속성에 값을 지정하여 축의 값 범위를 설정합니다. 두 값을 동시에 설정하려면 ***SetRange(...)*** 방법을 사용합니다. ***AllowScrolling*** 이 활성화된 경우 마우스로 축을 드래그하여 값 범위를 직접 스크롤할 수 있습니다. ***AllowScaling*** 속성이 활성화된 경우 배율 펜촉 영역(축 끝)을 위 또는 아래로 드래그하여 ***최소*** 또는 ***최대*** 값을 수정할 수 있습니다.

7.1.2 분할 및 그리드

MajorDiv 및 ***MinorDiv*** 속성에 의해 제어되는 Division 은 차트에서 메이저 및 마이너 눈금의 양을 결정합니다. 예를 들어, 5 개의 주요 분할을 설정하면 눈금과 주요 그리드 선으로 구분된 동일한 크기의 5 개의 공간에서 Y 축이 분할됩니다. 기본적으로 메이저 눈금은 활성화되고 마이너 눈금은 비활성화됩니다.

AutoDivSpacing -속성을 사용하면 주요 분할을 자동으로 계산할 수 있습니다. 기본적으로 활성화되어 있습니다. 간격은 값 레이블의 글꼴 크기와 ***AutoDivSeparationPercent*** 속성을 기반으로 계산되어 최대한 사용자 친화적입니다. 마이너 디비전은 ***MinorDivCount*** 속성 값에 따라 메이저 디비전간에 계산됩니다.

AutoDivSpacing 이 비활성화된 경우 ***MajorDiv*** 및 ***MajorDivCount*** -속성을 사용하여 나누기 간격을 수동으로 제어할 수 있습니다. ***MajorDiv*** 는 크기별로 간격을 제어하는 반면 ***MajorDivCount*** 는 나누기 수로 간격을 제어합니다. ***KeepDivCountOnRangeChange*** 속성은 ***MajorDiv*** 설정에 관계없이 축 범위가 변경될 때마다 동일하게 분할 수를 유지하는데 사용할 수 있습니다.

주요 분할 눈금 스타일은 ***MajorDivTickStyle*** 속성에서 설정할 수 있습니다. ***MajorDivTickStyle.Alignment*** 속성을 사용하여 눈금 및 레이블 방향을 편집합니다. 값 레이블은 주요 분할 눈금 옆에 그려집니다. 각각 ***MinorDivTickStyle*** 속성을 사용하여 부 나누기 속성을 수정할 수 있습니다.

분할 눈금의 수직 위치에 수평 그리드 선이 그려집니다. 메이저 분할 눈금을 위한 매이저 그리드, 마이너 디비전 눈금을 위한 마이너 그리드. ***MajorGrid*** 및 ***MinorGrid*** 속성을 사용하여 그리드의 모양을 편집할 수 있습니다.

7.1.3 사용자 지정 눈금

축 눈금 위치 및 레이블 텍스트는 사용자 지정 눈금을 사용하여 수동으로 설정할 수 있습니다. ***CustomTicksEnabled*** 를 true 로 설정하고 ***CustomTicks*** 목록 속성으로 눈금의 위치를 정의합니다. 사용자 지정 눈금은 눈금, 격자 또는 둘 모두로 구성될 수 있습니다. ***Style*** 을 사용하여 각각 Tick, Grid 또는 TickAndGrid 중에서 선택합니다. 눈금 또는 격자의 색상은 ***Color*** 속성을 통해 수정할 수 있습니다. ***Length*** 속성에서 눈금 길이를 설정합니다. 그리드 선 패턴은 축의 ***MajorGrid.Pattern*** 및 ***PatternScale*** 속성 설정을 따릅니다.

CustomAxisTick 에는 해당 위치 및 해당 레이블 텍스트를 정의하는 ***AxisValue*** 및 ***LabelText*** 속성이 있습니다. 사용자 지정 눈금을 사용할 때 ***AutoFormatLabels*** 를 비활성화하여 사용자 지정 레이블 텍스트를 표시합니다. 또한 코드에서 새 사용자 지정 눈금을 설정한 후에 ***InvalidateCustomTicks()***를 호출해야 합니다.

7.1.4 로그 축

ScaleType 을 ***Logarithmic*** 으로 설정하여 로그 표현을 사용하며, ***LogBase*** 속성을 사용하여 로그 기준 값을 설정합니다. 차트는 0... 1 사이의 로그 값을 표시할 수도 있습니다. ***LogZeroClamp*** 를 사용하여 축의 최소값을 설정합니다. 로그 축의 일반적인 최소값을 사용하려면 1 을 설정하고, 0 미만의 값을 사용하려면 사용된 데이터에 적합한 1.0E-20 과 같은 적절한 작은 양의 값을 설정하십시오. 10 기본 로그 또는 자연로그와 같은 눈금 레이블에 특수 형식을 사용하려면 ***LogLabelsType*** 을 사용하십시오.

7.1.5 축 값과 화면 좌표 간 변환

축에는 축 값(데이터 포인트 값)을 화면 좌표로, 화면 좌표를 축 값으로 변환하는 방법이 있습니다. ***ValueToCoord*** 방법을 사용하여 축 값을 화면 좌표로 변환하고 ***CoordToValue*** 를 사용하여 화면 좌표를 축 값으로 변환합니다. 픽셀이 선호되는 경우 장치 독립 픽셀(DIP)이 아니라 ***UseDIP = False*** 로 설정하십시오.

```
float screenCoordinate = _chart.ViewXY.XAxes[0].ValueToCoord(axisValue);
```

ValueToCoord 및 CoordToValue 방법은 차트가 최종 크기를 얻은 후에 사용할 수 있습니다. 예를 들어 ***chart.AfterRendering*** 이벤트를 구독하여 차트가 완전히 렌더링 되었는지 확인합니다.

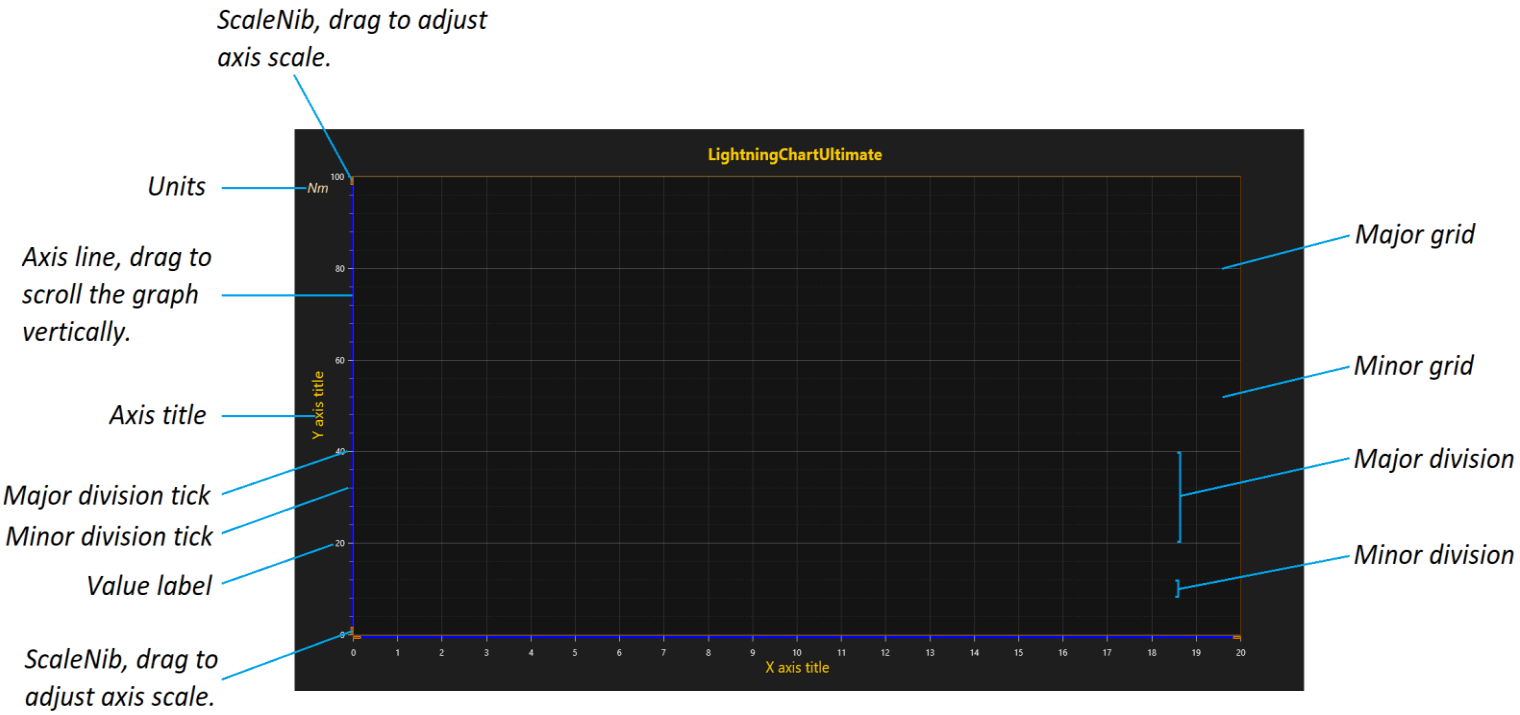
여러 값 또는 좌표를 한 번에 변환하려면 ***ValuesToCoords*** 및 ***CoordsToValues*** 방법을 사용합니다. 축 값을 이중 배열(X 축 ***CoordToValue*** 의 정수 배열)로 취하고 / 반환하고 화면 좌표를 부동 배열로 반환합니다.

7.2 Y 축

Y 축은 무제한으로 정의할 수 있습니다. Y 축 컬렉션 속성을 사용하여 ***Y 축을*** 추가합니다.

```
// Adding Y-axes to the chart
AxisY axisY = new AxisY(_chart.ViewXY);
axisY.Title.Text = "Y-axis";
chart.ViewXY.YAxes.Add(axisY);
```

7.3 X axis



X 축 분할 및 그리드 설정은 Y 축 설정과 동일합니다. 따라서 이전 장에서 설명한 모든 속성과 기능을 X 축에도 적용할 수 있습니다. 그러나 X 축에는 Y 축에는 없는 여러 실시간 스크롤 관련 속성이 있습니다.

7.3.1 실시간 모니터링 스크롤

실시간 모니터링 솔루션을 만들 때 x 축을 스크롤하여 현재 모니터링 위치를 올바르게 표시해야 합니다. 이는 일반적으로 최신 신호 지점의 타임 스탬프입니다. 새 신호 포인트가 시리즈로 설정된 후 최신 타임 스탬프를 ScrollPosition 속성으로 설정합니다.

LightningChart 에는 ***ScrollMode*** 속성을 사용하여 선택한 여러 스크롤 모드가 있습니다.

None

기본 옵션입니다. ***ScrollPosition*** 을 ***None*** 으로 설정하면 스크롤이 적용되지 않습니다. 실시간 모니터링을 사용하지 않을 때 사용하는 경우가 많습니다.

Stepping

수집된 데이터가 x 축 끝에 도달하면 모든 시리즈 데이터가 있는 축이 스테핑 간격 만큼 왼쪽으로 이동합니다. 이 시프트는 x 축 끝에 도달할 때마다 실행되며, ***SteppingInterval*** 속성은 값 범위로 정의됩니다.

Scrolling

x 축은 스크롤링 간격에 도달할 때까지 고정된 상태로 유지되며 이후 모든 시리즈의 x 축은 계속 왼쪽으로 이동합니다. 스크롤 위치가 x 축 끝에 도달했을 때 스크롤링이 적용되는 경우 ***ScrollingGap*** 을 0 으로 설정하며, ***ScrollingGap*** 속성은 그래프 너비의 백분율로 정의됩니다.

LightningChart 는 ***series.AddPoints(), AddValues()*** 또는 ***AddSamples()*** 방법을 사용할 때 실시간 신호의 증분 렌더링 데이터 구성을 지원합니다. 즉, 렌더링 데이터는 데이터의 새 부분에서만 계산되고 기존 렌더링 데이터와 결합됩니다.

증분 렌더링 데이터 생성을 사용하려면 다음과 같이 새 포인트를 추가합니다.

```
chart.BeginUpdate();
series.AddPoints(array,false);
xAxis.ScrollPosition = latestXValue;
chart.EndUpdate();
```

시리즈의 ***InvalidateData()*** 호출로 언제든지 렌더링 데이터를 완전히 새로 고칠 수 있습니다.

Sweeping

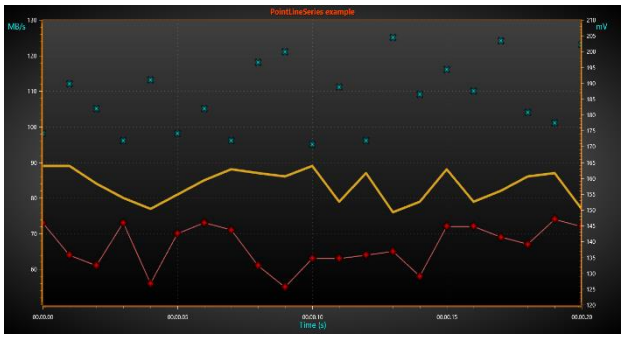
스위핑 모드는 아마도 가장 사용자 친화적인 실시간 모니터링 보기를 제공합니다. 스위핑은 두 개의 x 축을 사용합니다. 첫 번째 축이 완전히 수집된 후 스윕 간격이 나타납니다. 그런 다음 두 번째 x 축이 첫 번째 x 축 위로 스윕 되며, 두 x 축 모두 자체값 레이블을 표시합니다. ***SweepingGap*** 속성은 그래프 너비의 백분율로 정의됩니다.

Triggering

X 축 위치는 트리거 레벨을 초과하거나 아래로 떨어지는 시리즈 값에 의해 결정됩니다. ***Triggeing*** 속성을 사용하여 트리거링 옵션을 설정하며, ***Triggering.TriggeringActive*** 속성을 활성화하여 트리거링을 활성화할 수 있습니다.

하나의 시리즈를 트리거링 시리즈로 설정해야 합니다. 허용되는 트리거링 시리즈 유형은 ***PointLineSeries*** 및 ***SampleDataSeries*** 입니다. ***Triggering.TriggerLevel*** 을 사용하여 트리거링 Y 레벨을 설정하며, ***Triggering.TriggeringXPosition*** 을 사용하여 레벨 트리거 포인트가 그래프 너비의 백분율로 수평으로 그려지는 위치를 정렬합니다.

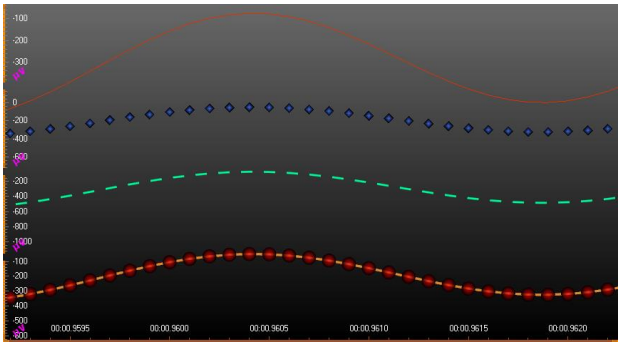
7.4 PointLineSeries



PointLineSeries 는 단순한 선, 점 (분산) 또는 둘 다를 점 선으로 표시할 수 있습니다. 시리즈의 X 값은 점진적 순서여야 합니다. ***PointLineSeries*** 목록에 ***PointLineSeries*** 개체를 추가하여 차트에 계열을 추가합니다.

LineStyle 속성으로 선 스타일을 정의합니다. 선이 표시되지 않아야 하는 경우 ***LineVisible = false*** 를 설정하며, 포인트를 표시하려면 ***PointsVisible = true*** 를 설정합니다. ***PointStyle*** 속성을 설정하여 포인트 스타일을 변경하고, ***PointStyle.Shape*** 를 ***사용하여 원, 삼각형, 십자형 및 비트 맵***과 같은 미리 정의된 여러 스타일에서 모양을 선택합니다. ***PointLineSeries*** 는 개별 포인트 컬러링도 지원합니다.

7.5 SampleDataSeries

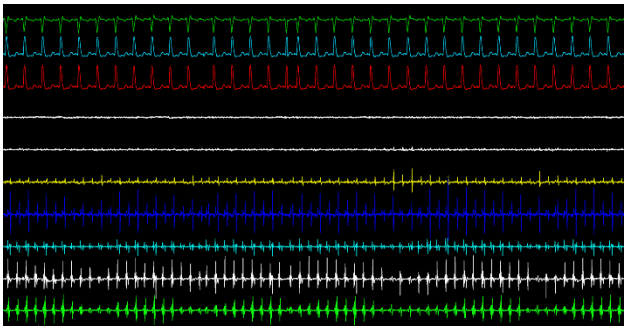


SampleDataSeries 는 실시간 DSP 앱에서 자주 사용되는 샘플링된 신호 데이터(이산 신호 데이터)를 표시하는 데 사용되며, 점진적 고정 간격 데이터를 사용합니다. 시각적으로 ***PointLineSeries*** 와 유사하므로 모든 선 및 점 서식 옵션이 적용됩니다. ***SampleDataSeries*** 에는 고정된 샘플 간격이 있으므로 포인트 X 값을 저장하기 위해 메모리를 예약할 필요가 없습니다.

SampleDataSeries 개체를 ***SampleDataSeries*** 목록에 추가하여 차트에 계열을 추가합니다. ***SampleDataSeries*** 는 단 정밀도 및 배정 밀도 샘플 Y 값을 지원하며, 메모리 예약을 가능한 한 낮게 유지하려면 단 정밀도 값을 사용하는 것이 좋습니다. ***SampleFormat*** 속성을 사용하여 샘플 형식을 선택하고, 시리즈 ***SamplingFrequency*** (1/샘플 간격)를 사용하여 고정 샘플 간격을 설정합니다. 샘플이 시작되는 X 값(타임스탬프)을 설정하려면 ***FirstSampleTimeStamp*** 속성을 설정합니다.

샘플은 코드에 추가되어야 합니다. ***AddSamples*** 방법을 사용하여 기존 샘플 끝에 샘플을 추가합니다.

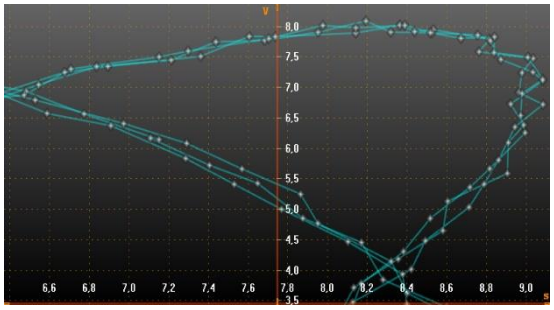
7.6 SampleDataBlockSeries



SampleDataBlockSeries 는 실시간 앱에 완전히 최적화된 ***SampleDataSeries*** 버전입니다. 최소한의 CPU 및 메모리 소비로 최상의 성능을 제공하므로 매우 많은 수의 데이터 포인트를 동시에 렌더링 할 수 있습니다. 시리즈의 이름에서 알 수 있듯이 데이터는 내부적으로 블록으로 관리되고 차례로 개별적으로 메모리로 관리되므로, 따라서 매우 큰 연속 선형 메모리가 필요하지 않습니다. ***SampleDataBlockSeries*** 는 ECG / EKG, EEG, 산업용 모니터링 앱, 원격 측정 및 파형 진동 모니터링과 같은 실시간 의료 모니터링 앱을 위한 최적의 시리즈 유형입니다.

SampleDataBlockSeries 는 ***SampleDataSeries*** 와 거의 유사하게 작동합니다. 마찬가지로 추가된 데이터가 점진적 순서로 고정된 데이터 간격을 가져야 합니다. ***SamplingFrequency***(1/샘플 간격)를 사용하여 고정된 샘플 간격을 설정할 수 있습니다. 샘플이 시작되는 X 값(타임스탬프)을 설정하려면 ***FirstSampleTimeStamp*** 속성을 설정하지만, 시각적으로 ***SampleDataBlockSeries*** 는 다른 라인 시리즈에 비해 서식 옵션이 적습니다. ***색상***및 ***너비*** 속성을 사용하여 선의 색상과 너비를 각각 변경할 수 있습니다. 또한 ***SampleDataBlockSeries*** 는 개별 점이 아닌 선만 표시합니다.

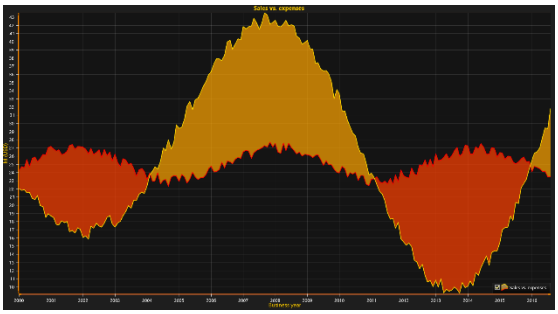
7.7 FreeformPointLineSeries



FreeformPointLineSeries 는 단순한 선, 점(분산) 또는 둘 다를 점 선으로 표시할 수 있으며, ***FreeformPointLineSeries*** 를 사용하면 이전 점에서 임의의 방향으로 선 점을 그릴 수 있습니다. 즉, 포인트는 점진적 순서일 필요는 없지만 데이터 간격은 무엇이든 될 수 있습니다. 따라서 ***FreeformPointLineSeries*** 는 다른 라인 시리즈에 비해 렌더링

하기가 약간 더 어렵습니다. ***PointLineSeries*** 의 모든 선 및 점 서식 옵션이 적용되며, ***FreeformPointLineSeries*** 개체를 ***FreeformPointLineSeries*** 목록에 추가하여 차트에 계열을 추가합니다.

7.8 High-lowSeries

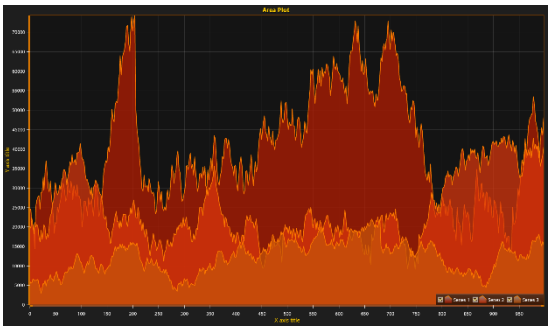


High-low 시리즈는 높은 값과 낮은 값 사이의 채워진 영역으로 데이터를 표시하며, ***HighLowSeries*** 개체를 ***HighLowSeries*** 목록에 추가하여 차트에 계열을 추가합니다.

채우기는 ***Fill*** 속성 및 하위 속성으로 설정할 수 있습니다. 데이터의 높은 값이 Low 값보다 작으면 해당 부분에 ***ReverseFill*** 이 적용되며, 선 및 점 스타일의 경우 높은 선과 낮은 선에 대한 별도의 속성이 있습니다. (예: ***LineStyleHigh*** 및 ***LineStyleLow***)

UseLimits 를 활성화하면 시리즈는 초과 한계 초과 및 사망 한계 미만의 다른 단색을 표시합니다. 그 후 일반 채우기 및 ***ReverseFill*** 은 제한 사이의 범위에만 적용되며, ***UsePalette*** 를 활성화하면 채우기가 ***ValueRangePalette*** 단계를 사용합니다. 균일 한 색상과 그라데이션 색상이 모두 지원됩니다. 데이터 값은 코드에 추가되어야 합니다. 데이터는 X 값 (**Points[i+1].X ≥ Points[i].X**)에 따라 점진적인 순서로 제공되어야 합니다. ***AddValues(HighLowSeriesPoint[], bool invalidate)*** 방법을 사용하여 기존 값 배열의 끝에 데이터 값을 추가합니다.

7.9 AreaSeries



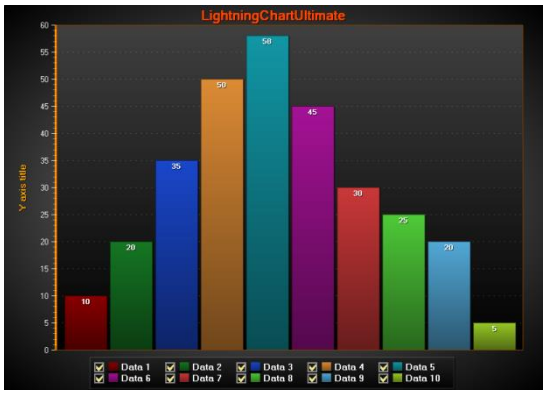
AreaSeries는 기본 레벨과 값 사이의 채워진 영역으로 데이터를 표시합니다. Area series는 ***HighLowSeries***와 매우 비슷하지만 더 간단한데, ***AreaSeries*** 목록에 ***AreaSeries*** 개체를 추가하여 차트에 계열을 추가합니다.

BaseValue 속성으로 기본 수준을 설정하고, ***Fill*** 속성으로 선호하는 채우기 스타일을 설정합니다. ***LineStyle*** 속성으로 선 스타일을 설정하고 ***PointStyle*** 속성으로 포인트 스타일을 각각 설정할 수 있습니다. 초과 및 기한 제한은 ***HighLowSeries***에서와 같이 사용할 수 있습니다.

데이터 값은 코드에 추가되어야 하며, 데이터는 X 값, ***Points [i + 1] .X ≥ Points [i] .X***에 따라 오름차순으로 제공되어야 합니다. ***AddValues (AreaSeriesPoint [], bool invalidate)*** 방법을 사용하여 기존 값 배열의 끝에

데이터 값을 추가합니다. 이전 데이터를 덮어쓰면서 전체 시리즈 데이터를 한 번에 설정하려면 새 데이터 배열을 직접 할당하십시오.

7.10 BarSeries



BarSeries 는 데이터를 수평 또는 수직 막대로 표시할 수 있습니다. 막대 계열의 값을 저장하려면 ***Values*** 배열 속성을 사용합니다. ***AddValue(...)*** 방법을 값을 추가합니다. ***SetValue(...)*** 방법을 사용하여 주어진 값 인덱스로 기존 값을 업데이트합니다. 값은 다음 필드가 있는 ***BarSeriesValue*** 유형입니다.

- ***값*** 바 길이
- ***위치*** 막대의 X 축 위치 (수직 막대) 또는 Y 축 위치 (수평 막대)
- ***텍스트*** 막대에 표시되는 텍스트

막대가 표시되는 방식을 제어하려면 차트의 ***BarViewOptions*** 속성을 사용합니다. ***BarViewOptions.Orientation*** 은 ***가로*** 및 ***세로*** 막대 방향 중에서 선택하며, ***BarViewOptions.Grouping*** 을 사용하면 값 인덱스, 너비 맞춤을 사용하는 인덱스 또는 위치 값 별로 막대를 그룹화할 수 있습니다. 다른 막대 시리즈의 값을 시각적으로 함께 가져오며, 그룹화를 원하지 않는 경우 ***BarViewOptions.Grouping.ByLocation*** 을 사용하고 모든 ***BarSeriesValue*** 개체에 대해 다른 위치 필드를 설정합니다.

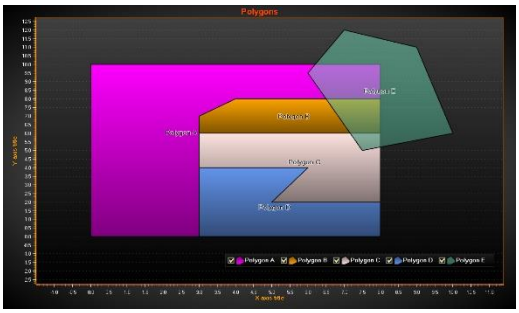
7.11 StockSeries

Stock 시리즈를 사용하면 캔들 스틱 또는 주식 막대 형식으로 증권 거래소 데이터를 시각화할 수 있습니다. ***StockSeries*** 목록 속성에 ***StockSeries*** 개체를 추가하여 동일한 차트에 여러 StockSeries 를 추가할 수 있으며, Style 속성으로 스타일을 선택합니다. 옵션은 ***Bars***, ***CandleStick*** 및 ***OptimizedCandleStick*** 입니다.

ColorStickDown, ***ColorStickUp***, ***FillDown*** 및 ***FillUp*** 속성으로 채색 및 채우기 옵션을 설정하고, ***StickWidth*** 속성을 사용하여 스틱 너비를 픽셀 단위로 조정합니다. ***ItemWidth*** 속성을 사용하여 총 데이터 항목 너비를 조정합니다. ***StockSeries*** 는 ***Behind = True*** 로 설정하여 라인 시리즈 이전에 렌더링 하도록 설정할 수 있습니다.

StockSeries 에 데이터를 설정하려면 데이터 배열을 만들고 배열 항목을 설정합니다. 각 항목에는 ***날짜***, ***시가***, ***종가***, ***낮음***, ***높음***, ***거래***(선택적), ***거래량***(선택적) 항목이 있으며, 데이터를 날짜 값에 따라 오름차순으로(가장 오래된 날짜부터) 유지합니다.

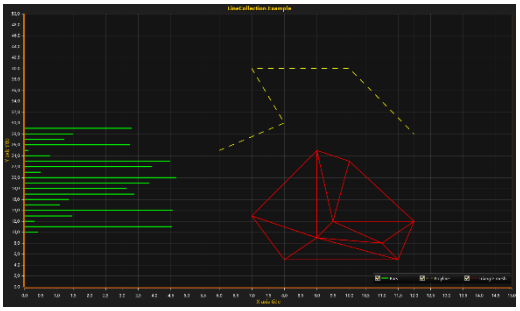
7.12 PolygonSeries



PolygonSeries 는 주어진 테두리 경로에 따라 채우기와 테두리를 렌더링 합니다.

Fill 속성에서 채우기 기본 설정을 지정합니다. ***PolygonSeries*** 의 ***Border*** 속성을 사용하여 테두리선 스타일을 설정하고, ***Points*** 속성에서 경로 지점을 설정합니다. ***PolygonSeries*** 에는 자동 경로 닫기 기능이 있습니다. 마지막 포인트가 첫 번째 포인트에 연결되어 있지 않으면 차트가 자동으로 연결합니다.

7.13 LineCollections



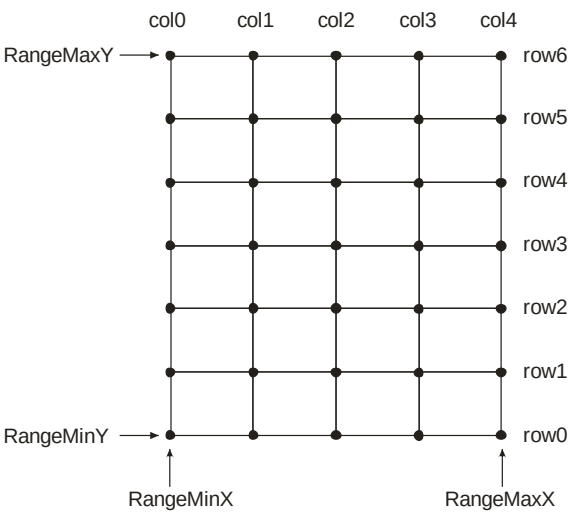
LineCollection 은 선분의 모음입니다. 각 선분은 A 지점에서 B 지점으로 가는 선이므로 4 개의 필드(시작 지점 X 및 Y, 끝 지점 X 및 Y)를 포함합니다. 하나의 ***LineCollection*** 에는 수천 개의 선 세그먼트가 포함될 수 있으며, ***LineCollection*** 은 ***PointLineSeries***, ***FreeformPointLineSeries*** 또는 ***SampleDataSeries*** 와 달리 수천 개의 개별 선 세그먼트를 렌더링 하는 데 매우 효율적입니다. ***PointLineSeries***, ***FreeformPointLineSeries*** 또는 ***SampleDataSeries*** 는 수백만 포인트의 연속 폴리 라인을 렌더링 하는 데 더 효율적입니다.

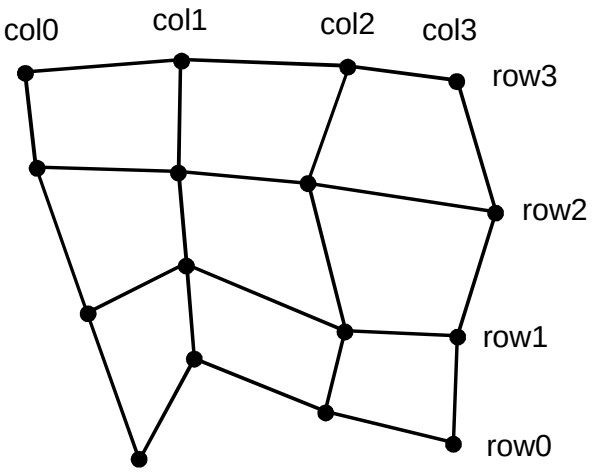
ViewXY.LineCollections 목록 속성에 ***LineCollection*** 개체를 추가하고, ***LineStyle*** 속성을 사용하여 선 색상, 스타일 및 너비를 제어하며, ***Lines*** 속성에서 선분을 설정합니다. 다음과 같이 ***Lines*** 속성에 ***SegmentLines*** 배열을 추가합니다.

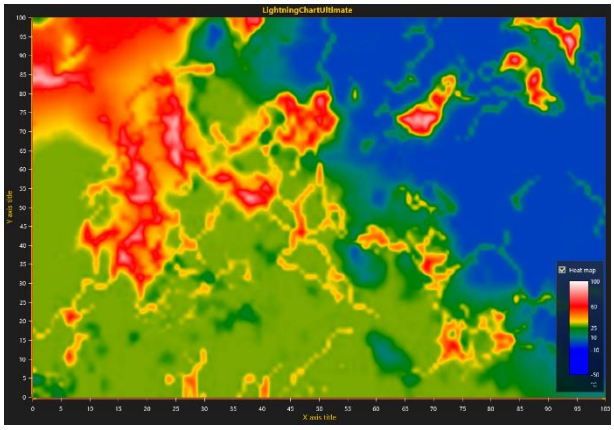
```
lineCollection.Lines = new SegmentLine[] {
                new SegmentLine(6,25,8,30),
                new SegmentLine(10,40,12,28) };
```

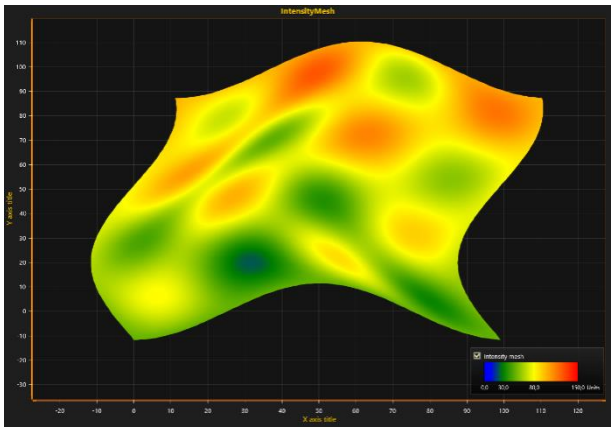
7.14 IntensityGrid- 및 IntensityMeshSeries

IntensityGrid- 및 ***IntensityMeshSeries***를 사용하면 할당된 값 범위 팔레트로 색상이 지정된 M x N 배열의 노드를 시각화할 수 있으며, 노드 사이의 색상이 보간 됩니다. ***IntensityGridSeries***는 X 및 Y 차원의 균일 한 간격의 직사각형 계열이며 ***IntensityMeshSeries*** 계열 노드는 X-Y 공간에 임의로 배치될 수 있습니다. 두 시리즈 모두 등고선, 등고선 레이블 및 와이어 프레임을 렌더링 할 수 있습니다.









왼쪽:IntensityGridSeries, 오른쪽: IntensityMeshSeries.

Fill 속성을 사용하여 채우기 스타일을 선택합니다. ***Paletted*** fill 을 사용할 때 ***ValueRangePalette*** 속성을 사용하여 값 색상 지정을 위한 색상 단계를 정의하고, 팔레트는 ***MinValue***, ***유형*** 및 ***단계*** 특성으로 정의됩니다. 유형에는 ***균일*** 및 ***그라데이션***의 두 가지 선택 사항이 있습니다. ***ValueRangePalette*** 는 ***Fill***, ***Wireframe*** 및 ***Contour*** 선에 사용할 수 있으며, 윤곽 팔레트에 대한 여러 단계를 정의합니다. 각 단계에는 높이 값과 해당 색상이 있습니다.

참고! 20 단계가 사전 컴파일되고 빠르게 로드됩니다. 단계 수가 많을수록 차트를 초기화할 때 몇 초 지연될 수 있습니다.

데이터는 ***Data*** 속성에 2 차원 배열로 저장되며, 각 배열 항목은 ***IntensityPoint*** 유형입니다. ***IntensityPoint*** 구조체의 ***Value*** 필드에 각 노드의 데이터 값을 저장하여 ***ValueRangePalette*** 에서 사용해야 하는 색상을 알려줍니다. ***IntensityMesh*** 의 지오메트리 또는 ***IntensityGrid*** 시리즈의 ***SizeX*** 또는 ***SizeY*** 가 데이터가 빠르게 변경되는 동안 변경되지 않는 경우 ***SetValuesData*** 및 ***SetColorsData*** 메서드를 사용하여 그리드를 업데이트하는 것이 가장 유리하고, ***InvalidateData()*** 방법을 호출하여 그리드/메시를 새로 고칩니다.

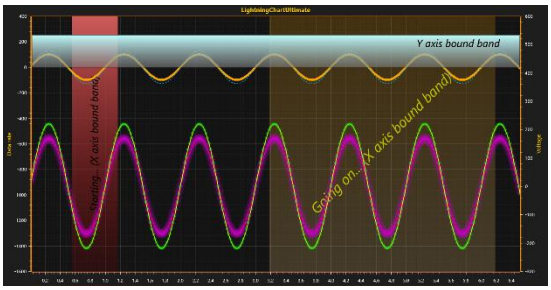
Intensity grid 데이터 설정

- ***RangeMinX*** 및 ***RangeMaxX*** 속성을 사용하여 할당된 X 축의 X 범위를 설정합니다.
- ***RangeMinY*** 및 ***RangeMaxY*** 속성을 사용하여 할당된 Y 축의 Y 범위를 설정합니다.
- ***SizeX*** 및 ***SizeY*** 속성을 설정하여 그리드 크기를 열과 행으로 지정합니다.
- 각 노드에 대한 ***값*** 설정:

Intensity mesh 데이터 설정

- ***SizeX*** 및 ***SizeY*** 속성을 설정하여 메쉬 크기를 열과 행으로 지정합니다.
- 모든 노드에 대해 X, Y 및 값을 설정합니다.
- 형상이 변경되지 않으면 ***최적화***를 ***DynamicValuesData*** 로 설정하십시오.
- 형상이 변경되지 않으면 모든 노드에 대한 값만 설정하여 데이터를 업데이트하십시오. 형상이 변경되면 모든 노드의 X, Y 및 값을 업데이트합니다.

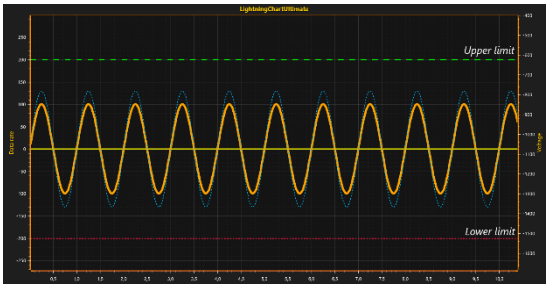
7.15 Bands



Bands 는 여백에서 다른 부분으로 이어지는 수직 또는 수평 영역입니다. ***바인딩*** 속성을 사용하여 밴드를 Y 축 또는 X 축에 바인딩할 수 있으며, Bands 가 Y 축에 바인딩된 경우 ***AssignYAxisIndex*** 속성도 설정해야 합니다.

Band 가장자리는 바인딩된 축의 값인 ***ValueBegin*** 및 ***ValueEnd*** 속성에 의해 설정되고, 마우스로 밴드를 다른 위치로 드래그할 수 있습니다. Bands 를 가장자리에서 드래그하여 크기를 조정하면 드래그 한 가장자리 값, ***ValueBegin*** 또는 ***ValueEnd*** 가 업데이트됩니다.

7.16 상수 라인

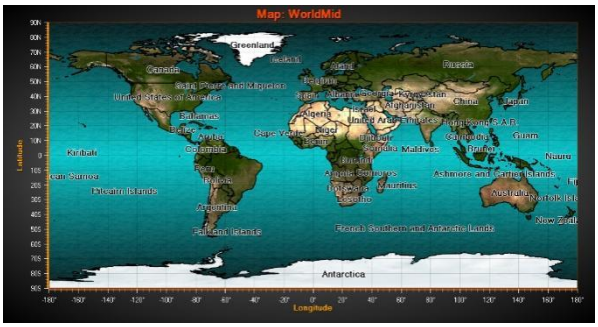


Bands 와 마찬가지로 상수 라인은 시리즈로 간주할 수 있습니다. 상수 라인은 Y 축에 바인딩되며 그래프 왼쪽 가장자리에서 오른쪽 가장자리까지 하나의 수평선을 나타냅니다. ***Value*** 속성을 통해 레벨을 설정하고, 일정한 선은 마우스로 드래그하여 수직으로 이동할 수 있습니다. **Behind** 속성을 true 로 설정하면 상수 라인이 선과 막대 계열 뒤에 그려지고, 그렇지 않으면 그 앞에 그려집니다.

7.17 Maps

Maps 속성 및 하위 속성을 사용하여 지리적 지도를 표시합니다. LightningChart 맵은 ***Vector Maps*** 와 ***Tile Maps*** 의 두 가지 범주로 나뉩니다. 지도는 정방형 투영적으로 표시되며, 이 투영을 사용하면 LightningChart 의 시리즈 유형과 거의 모두 X 및 Y 축에 바인딩된 기타 오브젝트를 맵과 동시에 사용할 수 있습니다.

7.17.1 Vector maps



지리 벡터 데이터는 확장자가 ***.md*** 인 LightningChart 맵 파일에 저장됩니다. LightningChart 는 맵 파일 세트와 함께 제공됩니다.

X 축은 경도에 사용되며 Y 축은 위도에 사용됩니다. 지도 좌표는 십진수도이며, 위도 원점은 적도, 경도 원점은 영국 그리니치입니다.

디렉터리 이름을 맵 파일이 있는 ***Path*** 속성으로 설정합니다. LightningChart 와 함께 제공되는 맵의 경우 활성 맵을 ***Type*** 속성으로 선택할 수 있습니다. 자체 지도 파일을 사용하려면 ***FileName*** 속성을 설정합니다. 맵이 필요하지 않으면 ***Type*** 을 ***Off*** 로 설정하십시오.

각 맵 파일에는 여러 레이어가 포함될 수 있습니다. 예를 들어, 육지, 호수, 강, 도로 및 도시에 대한 레이어입니다. 레이어 및 해당 데이터는 ***Layers*** 배열 속성에서 액세스 할 수 있습니다. 각 레이어에는 특정 유형이 있으며, 레이어 모양 옵션은 해당 옵션 속성으로 변경할 수 있습니다. 예를 들어, 토지 지역의 모양을 수정하려면 ***LandOptions*** 를 사용하십시오.

Mouse interactivity

지도 영역 및 개체와의 모든 종류의 상호 운용을 위해 ***AllowUserInteraction*** 을 활성화합니다. 지역 (토지, 호수) 및 벡터 레이어(강, 도로)는 마우스로 가리킬 수 있습니다. 마우스가 개체 위에 있으면 ***Highlight*** 속성에 따라 강조 표시되며, ***Highlight*** 를 ***None*** 으로 설정하면 객체가 강조 표시되지 않지만 여전히 클릭하여 ***Maps.ButtonDownOnMapItem*** 이벤트를 호출하는 데 사용할 수 있습니다.

지도 개체에는 인구 또는 기타 통계 데이터와 같은 관련 데이터가 포함될 수 있습니다. ***UserInteractiveDeviceOverOnMapItem***/***UserInteractiveDeviceOverOffMapItem*** /***ButtonDownOnMapItem*** 이벤트 처리기를 사용하여 데이터에 액세스 합니다. 지도 항목의 데이터는 키와 값의 사전을 제공하는 GetInfo 메서드를 사용하여 검색할 수 있습니다.

ESRI 모양 파일 데이터에서 지도 가져오기

가져오기 기능은 .shp 파일에서 LightningChart 맵 파일(.md)을 만듭니다. ESRI shapefile(*.shp)은 벡터 및 다각형 데이터를 지원하는 널리 사용되는지도 파일 형식입니다.

맵 마법사를 사용하여 shapefile 데이터를 LightningChart(LC) 맵 데이터 형식으로 변환할 수 있습니다. LC 형식은 계층화를 지원하므로 여러 shapefile 을 단일 파일로 병합할 수 있으며, 최대 런타임 성능을 위해 맵 파일 구조 및 개체가 사전 처리됩니다.

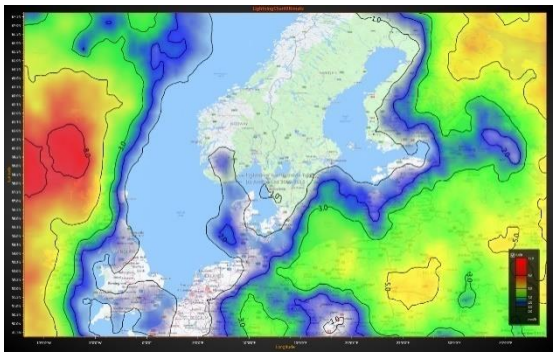
팁: LightningChart® .NET 의 데모 앱에는 맵 가져 오기의 예가 있습니다. 거기에서 가져오기 마법사를 실행하여 가져오기를 통해 사용자 정의 LC 맵 파일을 만듭니다.

변환은 최소 세 단계로 수행됩니다.

1. **Error! Reference source not found.**g 에서 파일을 선택하고 파일을 기반으로 레이어를 설정합니다.
2. 파일 텍스트 인코딩을 결정합니다.
3. 결과 맵 파일에 포함된 항목 선택합니다.

각 소스 shp 파일에 대해 2 단계와 3 단계가 반복됩니다. Shapefile 은 어떤 인코딩을 사용하는지 알려주지 않으므로 사용자가 선택해야 합니다.

7.17.2 Tile maps



LightningChart 는 현재 Here 맵을 지원 합니다. (스트리트 맵 및 위성 이미지, 개발자 또는 최종 사용자가 Here 서버를 사용하려면 Here 와 자체 계약을 체결해야 합니다)

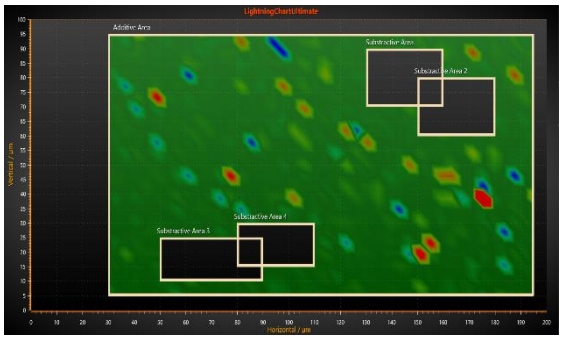
ViewXY.Maps.TileLayers 컬렉션에 ***TileLayer*** 개체를 추가합니다. ***AlphaLevel*** 속성으로 여러 레이어를 삽입하고 반투명하게 만들 수 있으며, ***TileLayer*** 개체는 ***TileLayers*** 컬렉션에 나타나는 순서대로 렌더링 되며 첫 번째 레이어는 배경에 있습니다. ***AboveVectorMap = False*** 로 설정하면 벡터 맵이 정의된 경우 레이어가 먼저 렌더링 됩니다. (6.26 참조) 기본적으로 ***TileLayer*** 는 벡터 맵 이후에 렌더링 됩니다.

TileLayer 는 http 프로토콜을 통해 온라인 서비스 제공 업체로부터 작은 이미지로 정보를 가져와 차트 영역에 표시합니다. 지도보기를 확대 / 축소하거나 이동하면 이미지가 새로 고쳐집니다.

차트는 타일을 캐시 폴더에 저장하므로 동일한 영역에서 자주 이동하거나 확대/축소할 때 로드 시간이 크게 단축됩니다. 차트에 타일을 표시해야 하는 경우 먼저 캐시 폴더에서 찾을 수 있는지 확인하고 그렇지 않은 경우 웹 서비스에서 검색합니다. 기본적으로 캐시 폴더는 ***c:\Users\[Current user]\AppData\Local\Temp*** 입니다.

ViewXY.Maps.TileCacheFolder 에서 캐시 폴더를 설정합니다. ***ViewXY.Maps.ClearTileCacheFolder()*** 방법을 호출하여 캐시 폴더를 지웁니다.

7.18 StencilAreas

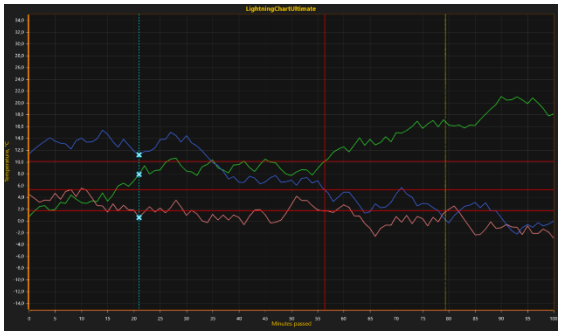


IntensityGridSeries, ***IntensityMeshSeries*** 및 ***Maps*** 에는 그려진 데이터의 영역 안팎을 마스킹할 수 있는 ***StencilArea*** 기능이 있습니다. ***StencilArea*** 는 새 ***StencilArea*** 객체를 만든 다음 ***AddPolygon()***을 통해 PointDouble2D - array 로 정의하거나 ***AddMapLayerIndex()***를 통해 맵 레이어로 정의한 다음 마지막으로 마스크 해야 하는 시리즈에 추가하여 적용할 수 있으며, 여러 ***StencilArea*** 를 설정할 수 있습니다. 두 개 이상의 영역이 겹치는 경우 영역이 결합됩니다.

- ***AdditiveAreas*** 는 포지티브 스텐실 마스크를 만듭니다. 영역 내부의 데이터만 그려지고 외부는 잘립니다.
- ***SubtractiveAreas*** 는 음의 스텐실 마스크를 만듭니다. 영역 내부의 데이터는 잘리고 외부는 그려집니다. ***SubtractiveAreas*** 는 클리핑이 적용되지 않고 ***AdditiveAreas*** 와 함께 작동하도록 설계되었습니다.

StencilArea 객체가 목록(***AdditiveAreas*** 또는 ***SubtractiveAreas***)에 추가될 때마다 해당 시리즈에 대해 ***InvalidateStencil()*** 또는 ***InvalidateData()***를 호출해야 합니다. 스텐실을 정의하는 점 배열은 시계 방향으로 설정하는 것이 좋습니다.

7.19 LineSeriesCursors



LineSeriesCursors 는 X 좌표로 값을 추적하여 라인 시리즈 데이터를 시각적으로 분석할 수 있습니다. 시리즈 값은 ***ITrackable*** 인터페이스를 구현하는 시리즈(***SampleDataSeries***, ***PointLineSeries***, ***AreaSeries***, ***HighLowSeries***)로만 확인할 수 있으며, 다른 계열 유형의 경우 Y 좌표는 커서에 의해 자동으로 추적되지 않습니다.

LineSeriesCursor 개체를 ***LineSeriesCursors*** 컬렉션에 추가하고, ***SnapToPoints*** 를 활성화하여 커서를 한 지점에서 지점으로 이동합니다. ***Style*** 속성으로 커서 추적 스타일을 설정합니다. ***Style*** 이 ***PointTracking*** 으로 설정되어 있으면 비트 맵 이미지를 비롯한 모든 추적 지점 스타일을 사용할 수 있습니다. ***HairCrossTracking*** 스타일을 사용하는 경우 라인 시리즈 포인트 Y 값에 수평선이 그려지며, 커서 위치에 동일한 시리즈의 여러 포인트가 부딪히면 최소 및 최대 포인트의 중간에 선이 그려집니다.

LineSeriesCursor 위치에서 데이터 값 풀기

ITrackable 인터페이스를 구현하는 시리즈는 X 화면 좌표 또는 X 축 값으로 해결할 수 있는데, 정확한 방법 인 ***SolveYValueAtXValue*** 는 데이터 포인트를 반복하고 가장 가까운 데이터 포인트 일치를 찾습니다. 거친 방법 ***SolveYCoordAtXCoord*** 는 일치하는 Y 화면 좌표를 해결하기 위해 시리즈의 캐시 된 렌더링 데이터를 사용합니다. ***AnnotationXY*** 객체는 종종 커서 옆에 값을 표시하는 데 사용됩니다.

7.20 EventMarkers

EventMarkers 를 사용하면 실시간 모니터링 중에 특별한 일이 발생했거나 특수 주석으로 데이터 조각을 표시하려는 경우 관심 지점을 표시할 수 있습니다. ***Symbol*** 속성으로 마커 심볼을 정의하고 ***Label*** 속성으로 텍스트 레이블을 정의하고, ***VerticalPosition*** 속성으로 수직 위치를 설정하고 필요한 경우 ***Offset*** 을 사용하여 개체 속성을 이동합니다. 모든 마커는 X 축에서 마커의 위치를 설정하는 ***XValue*** 로 할당되어야 합니다. 마커의 모양을 선택하려면 ***Symbol.Shape*** 를 설정하십시오.

차트 이벤트 마커

시리즈 이벤트 마커와 달리 ***ChartEventMarkers*** 는 특정 시리즈에 연결되지 않으며, 마커는 마우스로 다른 위치로 드래그할 수 있습니다.

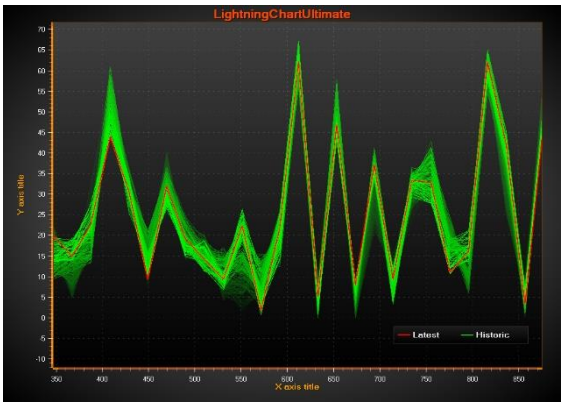
차트 마커의 위치는 ***VerticalPosition***, ***XValue*** 및 ***Offset*** 속성을 통해 설정할 수 있습니다. 또한 ***BindToXAxis*** 를 true 로 설정하면 마커가 특정 X 축에 바인딩됩니다. 실제로 이렇게 하면 마커가 현재 X 값을 유지하고 예를 들어 X 축이 이동될 때 이동하고, ***BindToXAxis*** 가 비활성화되면 축이 이동하는 방법에 관계없이 마커가 동일한 차트 위치에 유지됩니다.

라인 시리즈 이벤트 마커

라인 시리즈에는 ***SeriesEventMarkers*** 컬렉션 속성이 있습니다. 시리즈 별 이벤트 마커를 할당하는 데 사용할 수 있고, 시리즈 이벤트 마커는 시리즈 값에 연결된 이벤트 마커를 유지하면서 마우스로 다른 위치로 드래그할 수 있습니다. 이를 활성화하려면 마커의 ***VerticalPosition*** 을 ***TrackSeries*** 로 설정해야 합니다. ***ITrackable*** 인터페이스를 구현하는 시리즈에 사용할 수 있습니다.

HorizontalPosition 을 ***SnapToPoints*** 로 설정하면 마커가 가장 가까운 데이터 포인트의 위치에 수평으로 정렬되고, ***HorizontalPosition = AtXValue*** 를 사용하면 모든 x 값에 마커를 배치할 수 있습니다. 각각 ***VerticalPosition = AtYValue*** 를 사용하면 마커를 Y 레벨로 수직으로 설정할 수 있습니다.

7.21 영구 시리즈 렌더링 레이어 (PersistentSeriesRenderingLayer)



PersistentSeriesRenderingLayer 는 반복적인 선 및 점 데이터 또는 동일한 X 및 Y 범위에 반복적으로 그려지는 선/점/고저/영역 채우기 데이터의 매우 빠른 렌더링에 사용할 수 있습니다. ***PersistentSeriesRenderingLayer*** 는 렌더링 데이터를 점진적으로 추가할 수 있는 일종의 비트 맵입니다. 명령으로 지울 때까지 그래픽을 유지합니다. 이렇게 하면 각 업데이트 라운드에서 하나의 시리즈만 레이어에서 렌더링 한 다음 화면에서 레이어를 렌더링 하면 됩니다. CPU 로드 또는 메모리 사용량이 증가하지 않습니다. 기존 데이터를 점차적으로 희미하게 해야 하는 경우 비트 맵 픽셀의 알파를 곱하여 수행할 수 있습니다.

필요한 만큼 ***PersistentSeriesRenderingLayer*** 개체를 만들 수 있으며 모든 업데이트 라운드에서 시리즈 수를 렌더링 할 수 있습니다. 또한, ***PointLineSeries***, ***SampleDataSeries***, ***FreeformPointLineSeries***, ***HighLowSeries*** 또는 ***AreaSeries*** 객체를 레이어로 렌더링 할 수 있습니다.

레이어 구성

PersistentSeriesRenderingLayer 개체는 코드에서 만들어야 합니다.

```
using Arction.[edition].Charting.Views.ViewXY;

PersistentSeriesRenderingLayer layer = new PersistentSeriesRenderingLayer
(m_chart.ViewXY, m_chart.ViewXY.XAxes[0]);
```

렌더링 된 시리즈와 함께 사용되는 동일한 XAxis 객체를 제공합니다. 그런 다음 ***layer.RenderSeries()***를 사용하여 레이어에 하나 또는 여러 시리즈를 렌더링 합니다.

MultiplyAlpha (value)를 사용하면 레이어를 더 투명하게 만들거나 불투명하게 만들 수 있으며, 곱하면 레이어의 모든 픽셀에 개별적으로 효과가 있습니다. 1 미만의 값을 제공하면 투명도가 증가합니다. (레이어가 감쇠함) 각각 값> 1 을 제공하면 불투명도가 증가합니다. (레이어가 더 잘 보이게 함)

layer.Clear()는 레이어를 지우고 ARGB = (0,255,255,255)로 색상을 초기화합니다. ***layer.Clear(Color color)***는 주어진 색상으로 레이어를 지웁니다. 대부분의 경우 배경에 사용된 것과 동일한 색상을 설정하는 것이 가장 유용하지만 A = 0 으로 설정합니다. 레이어를 삭제하고 차트와 함께 렌더링 되지 않도록 하려면 ***layer.Dispose()***를 호출합니다.

알아야 할 몇 가지 레이어 제한 사항

특수 렌더링 기술로 인한 다음 제한 사항을 염두에 두십시오.

- X 축 ***ScrollMode*** 는 ***None*** 으로 설정해야 합니다. 이 방법에서는 X 축의 실시간 스크롤이 불가능합니다.

- 확대/축소, 이동, 축 조정 및 차트 크기 조정은 이미지가 축 범위와 동기화되지 않게 합니다. 영구 유동을 사용하거나 레이어를 지우고 새 레이어 렌더링(축 범위 변경 및 크기 조정을 위한 이벤트 핸들러가 있음)을 위해 일시적으로 이전 라인 시리즈를 다시 생성하도록 만들어진 앱 로직을 사용할 때는 이러한 기능을 비활성화해야 합니다.

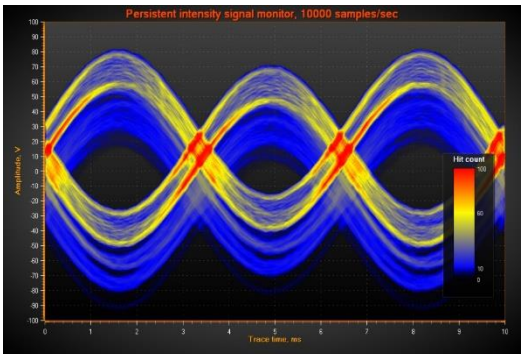
- 차트 크기 조정은 레이어를 지우고 Windows 데스크톱 잠금 상태에서 다시 시작합니다.

- 레이어에서만 렌더링 되는 시리즈에서는 마우스 상호 작용이 지원되지 않습니다.

- EMF/WMF/SVG 내보내기, 벡터 형식으로 클립 보드에 복사, 벡터 형식으로 인쇄는 레이어를 지원하지 않으며, 레스터 형식만 지원됩니다.

7.22 영구 시리즈 렌더링 강도 레이어 (PersistentSeriesRenderingIntensityLayer)

PersistentSeriesRenderingIntensityLayer 를 사용하면 추적을 레이어로 수집하고 픽셀 당 히트 수에 따라 색상을 지정할 수 있습니다. 색상은 값 범위 팔레트를 사용하여 만들고 추적은 ***PersistentSeriesRenderingLayer*** 에서와 동일한 시리즈 유형으로 사용할 수 있습니다. 두 번째 렌더링 호출로 픽셀 위치에서 트레이스를 다시 렌더링 하면 강도가 증가하여 값 범위 팔레트의 값이 증가합니다.



레이어 구성

PersistentSeriesRenderingIntensityLayer 개체는 코드에서 만들어야 합니다.

```
PersistentSeriesRenderingIntensityLayer layer = new
PersistentSeriesRenderingIntensityLayerngLayer(m_chart.ViewXY,
m_chart.ViewXY.XAxes[0]);
```

레이어의 ***ValueRangePalette*** 속성에서 팔레트 유형과 단계를 정의합니다. ***ValueRangePalette.Type = Gradient*** 는 그래디언트 색상을 만들고 ***ValueRangePalette.Type = Uniform*** 은 레이어를 개별 색상 단계로 렌더링 한 후, ***layer.RenderSeries()***를 사용하여 레이어에 하나 또는 여러 시리즈를 렌더링 합니다.

layer.Clear()는 레이어를 지우고 카운터를 재설정합니다. 레이어를 삭제하고 차트와 함께 렌더링 되지 않도록 하려면 ***layer.Dispose()***를 호출합니다.

새로운 흔적의 강도 효과 및 오래된 흔적의 감쇠 조정

NewTraceIntensity 속성을 사용하여 RenderSeries 호출로 렌더링 된 새 트레이스의 강도 효과를 제어합니다. 일반적인 값은 1... 100 이며, 트레이스를 채우도록 색상 범위를 설정하는 속도에 따라 다릅니다.

HistoryIntensityFactor 를 사용하여 이전 트레이스의 감쇠 속도를 조정합니다. 일반적인 값의 범위는 0.5 – 0.99 입니다.

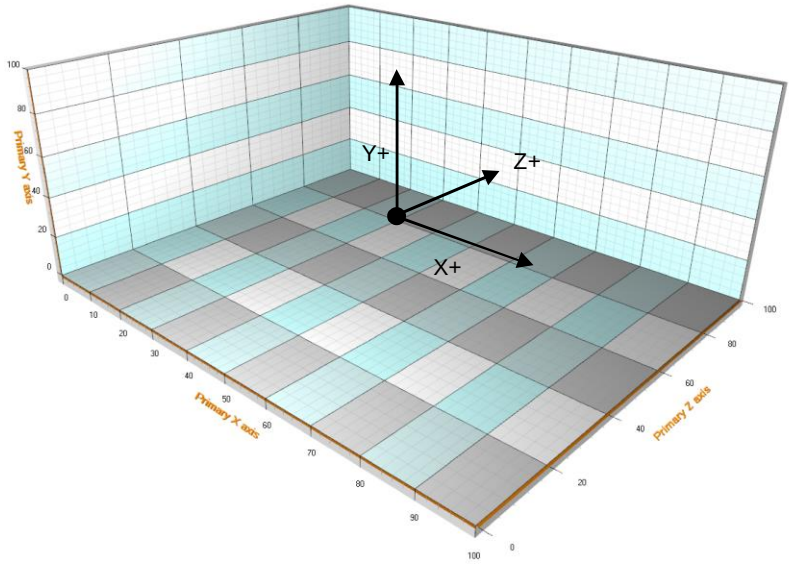
데이터가 레이어로 업데이트되면 ***NewTraceIntensity*** 가 새 추적에 사용됩니다. 이전 추적 데이터는 동시에 ***HistoryIntensityFactor*** 와 함께 감쇠됩니다. ***layer.RenderSeries(List<PointLineSeriesBase> seriesList)***는 모든 시리즈 객체 이후에 오래된 트레이스를 감소시킵니다.

8. View3D

View3D 를 사용하면 3D 공간에서 데이터를 시각화할 수 있습니다. 3D 모델은 다양한 방법으로 확대, 회전 및 조명할 수 있으며, 여러 시리즈 유형을 동일한 3D 보기에 배치하여 결합된 시각화를 만들 수 있습니다.

3D 모델은 3D 세계의 중심에 구성되고, 치수 크기는 3D 공간에서 모델 상자의 크기를 정의합니다. 벽과 축 크기는 이 치수 상자로 정의됩니다. ***Dimensions*** 속성을 사용하여 각 차원 크기를 설정합니다.

카메라 회전이 정의되지 않은 경우 양의 X 방향은 오른쪽, 양의 Y 치수는 위쪽, 양의 Z 방향은 화면 안쪽입니다.



일부 3D 개체는 축 값이 아닌 "표준 좌표"를 사용합니다. 예를 들어, 조명은 축 범위와 독립적으로 배치되며, 원점[0,0,0]은 모델의 중앙에 있습니다. 실제 3D 모형 공간의 범위는 [-Dimensions.X/2-Dimensions.X/2], [-Dimensions.Y/2-DimensionsY/2] 및 [-Dimensions.Z/2-Dimensions.Z/2 입니다.]

8.1 벽

벽(***WallOnFront***, ***WallOnBack*** 등)은 축 그리드와 그리드 스트립을 표시하고 축의 기반을 제공하는 데 사용됩니다. 기본적으로 하단, 왼쪽, 오른쪽, 후면 및 전면 벽이 표시되며, ***AutoHide*** 속성이 true 로 설정됩니다. 보기를 회전할 때 방해하는 벽이 일시적으로 숨겨져 차트 내용 보기를 가리지 않습니다. 벽을 강제로 표시하려면 ***Visible = true*** 및 ***AutoHide = false*** 로 설정합니다.

XGridAxis, YGridAxis, ZGridAxis, GridStripColorX, GridStripColorY, GridStripColorZ 및 ***GridStrips*** 속성을 사용하여 그리드를 적용할 축을 선택하고 그리드 스트립의 색상을 수정합니다. 사용 가능한 속성은 벽 방향에 따라 다르고, ***FullTransparent*** 속성은 솔리드 벽을 숨기면서 그리드 만 표시할 수 있습니다. ***FullTransparent*** 가 활성화된 경우에도 그리드는 여전히 벽의 ***Visible*** 및 ***AutoHide*** 특성을 따릅니다.

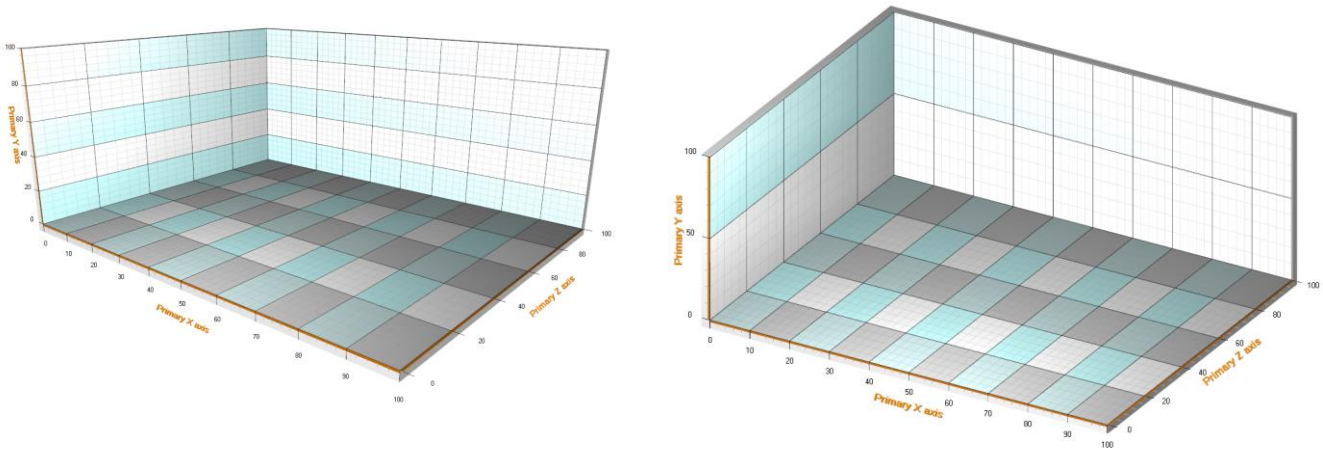
벽 대신 단순화된 3D 상자 프레젠테이션을 사용할 수 있습니다. 모든 벽에 대해 Visible = false 를 설정한 다음 ***FrameBox.Style = All***Edges 를 설정하고, ***FrameBox.LineColor*** 로 색상 또는 프레임을 설정합니다.

8.2 카메라

카메라 유형, 위치, 거리 및 대상이 함께 3D 시점을 결정합니다. ***RotationX, RotationY, RotationZ 및 ViewDistance*** 를 사용하여 3D 모델 공간에서 카메라 위치를 설정하고, ***Target*** 속성을 설정하여 카메라를 원하는 방향으로 지정합니다.

Projection 속성으로 투영 유형을 선택합니다.

- ***원근,*** 현실적인 투영을 보여줍니다.
- ***직교 투영***, 과학 및 엔지니어링 앱에 사용되는 유형입니다. 이는 ***OrthographicLegacy*** 보다 권장됩니다.
- ***직교 레거시,*** 축 값이 아닌 3D 표준 좌표로 정의된 경우 3D 개체의 크기를 유지합니다. 이는 직교에 비해 확대/축소 후 렌더링 속도가 느립니다.

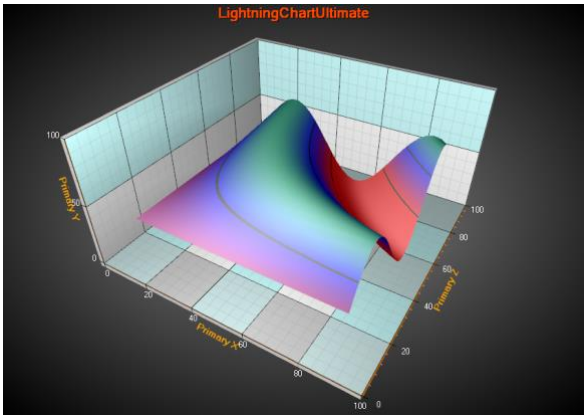


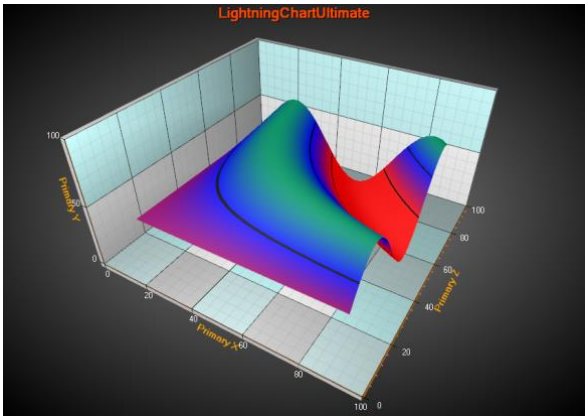
3D 공간에서 원근 및 직교 카메라 보기.

8.3 조명 및 재질

조명은 3D 모형 공간 어디에서나 자유롭게 배치할 수 있습니다. ***Lights*** 컬렉션 속성에 여러 조명을 추가할 수 있으며, 조명 유형에는 ***Directional*** 과 ***PointOfLight*** 의 두 가지가 있습니다.

모든 3D 개체에는 ***Material*** 속성이 있습니다. 재질은 조명에 반응하는 방법을 알려줍니다. 재질의 ***DiffuseColor*** 는 라이트의 ***DiffuseColor*** 와 반응하고, 재질의 ***SpecularColor*** 는 라이트의 ***SpecularColor*** 와 반응합니다. 확산 색상은 무광택 기본 색상으로 이해될 수 있는 반면, 반사 색상은 조명 표면에서 반사되는 색상입니다. 높은 ***SpecularPower*** 를 사용하면 물체가 금속처럼 보이며, Surface 시리즈에는 ***ColorSaturation*** 속성도 있으며 유효 범위는 0... 100 % 입니다. 높은 값은 표면 채우기 색상을 높이고 음영 효과를 줄입니다.

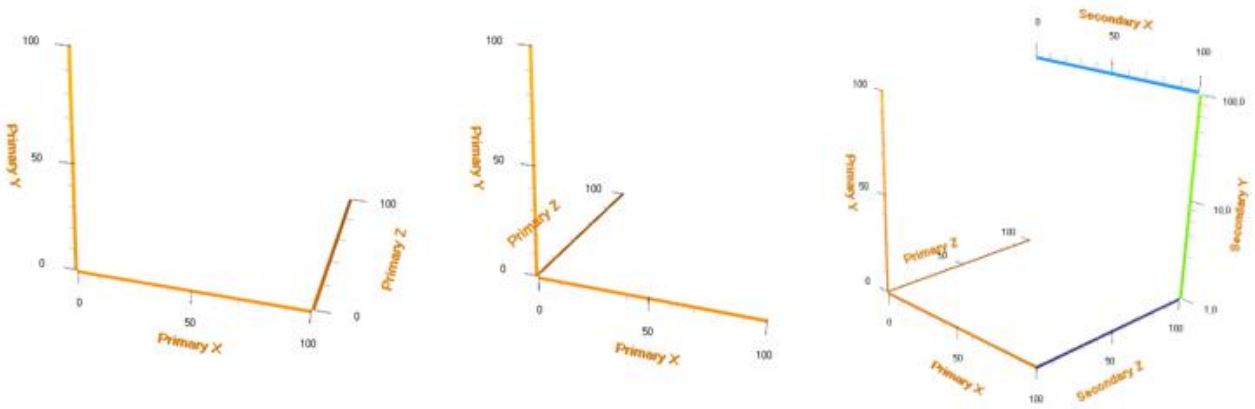




8.4 축

각 차원에 대해 기본 및 보조 축이 있습니다. 예를 들어 X 차원의 경우 ***XAxisPrimary3D*** 및 ***XAxisSecondary3D*** 입니다. 일반적으로 3D 축은 ViewXY 의 축과 매우 유사하게 작동합니다.

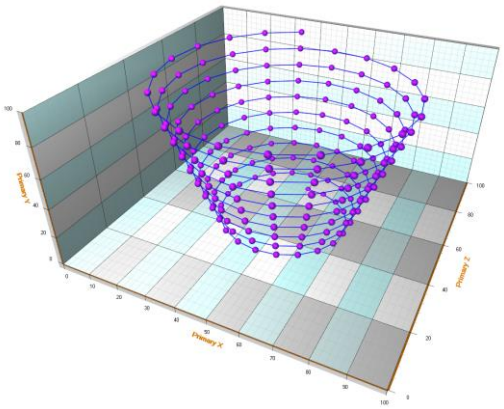
축은 3D 모델 상자 모서리에 배치할 수 있습니다. 축의 ***Location*** 속성을 사용하여 위치를 조정합니다.



8.5 3D 시리즈, 일반

View3D 시리즈를 사용하면 다양한 방식과 형식으로 데이터를 시각화할 수 있습니다. 모든 시리즈는 축 값 범위에 바인딩됩니다. 각 차원에 대해 계열을 선택하여 기본 또는 보조 축에 바인딩할 수 있으며, ***XAxisBinding, YAxisBinding*** 및 ***ZAxisBinding*** 속성을 사용하여 이를 제어합니다.

8.6 PointLineSeries3D



PointLineSeries3D 는 3D 공간에서 포인트와 라인을 표현할 수 있습니다. 포인트의 경우 사용할 수 있는 기본 3D 모양이 많이 있으며, ***LineVisible*** 속성이 ***true*** 로 설정된 경우 점은 선으로 연결됩니다. 포인트는 실제 3D 포인트 또는 2D 모양으로 표시될 수 있으며, 선은 음영 처리된 3D 선 또는 1 픽셀 너비의 머리카락 선으로 렌더링할 수 있습니다. 시리즈에 많은 데이터가 있는 경우 성능 문제를 방지하려면 ***LineOptimization = Hairline*** 을 설정하는 것이 좋습니다.

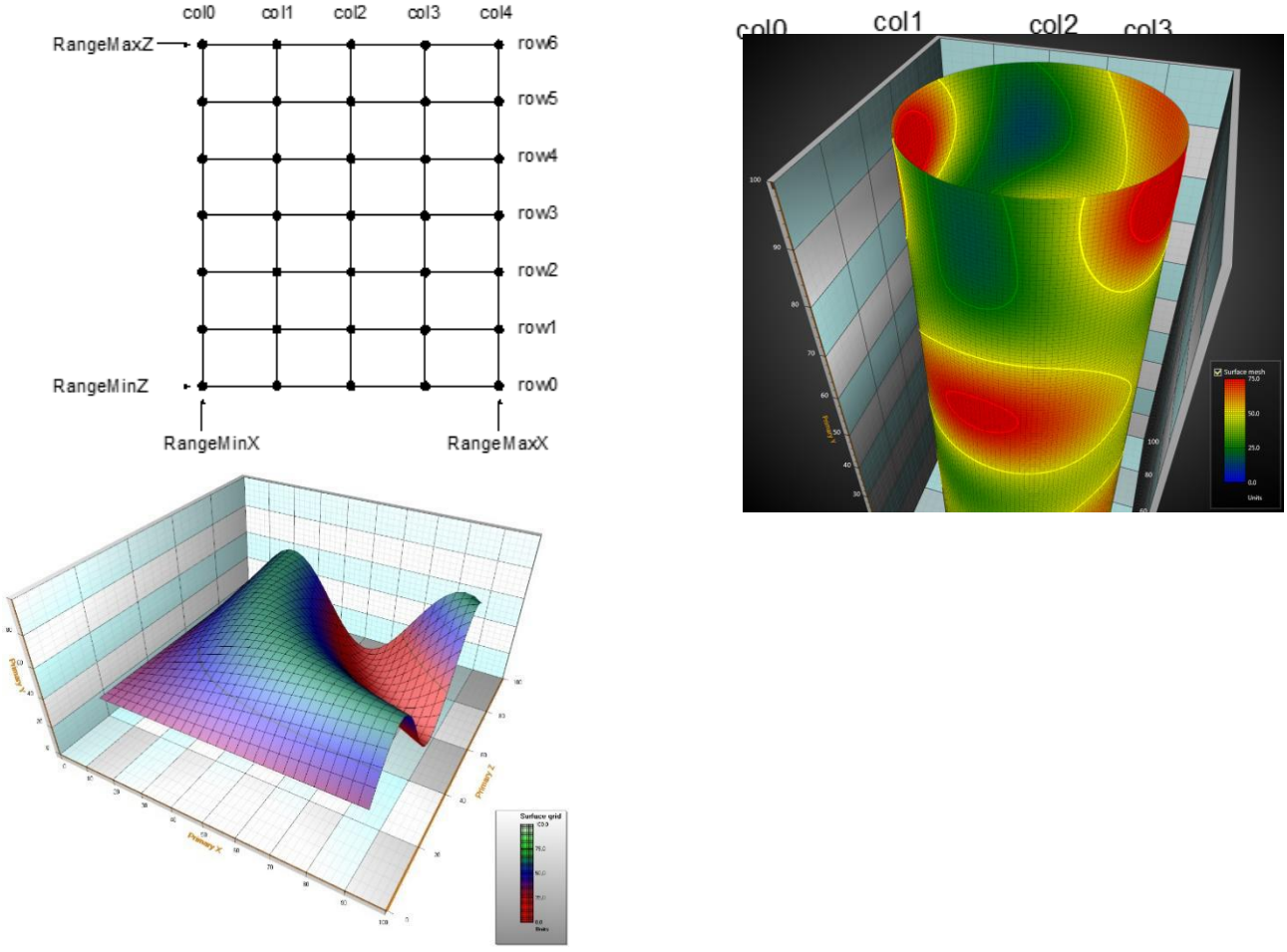
PointLineSeries3D 는 Points(SeriesPoint3D 배열), PointsCompact(SeriesPointCompact3D 배열) 및 PointsCompactColored(SeriesPointCompactColored3D 배열)의 세 가지 다른 포인트 형식을 지원합니다. ***PointsCompact*** 및 ***PointsCompactColored*** 구조는 매우 메모리 효율적이므로 간단한 포인트 스타일로 최대 1억 개의 데이터 포인트 시각화가 가능하고, ***PointsType*** 속성을 통해 포인트 형식을 설정합니다.

시리즈 포인트는 코드에 추가되어야 합니다. ***AddPoints(...)*** 방법을 사용하여 기존 점 끝에 점을 추가합니다. 이전 포인트를 덮어쓰면서 전체 시리즈 데이터를 한 번에 설정하려면 새 포인트 배열을 직접 할당하십시오.

IndividualPointColors = True 로 설정하면 ***Material.DiffuseColor*** 대신 포인트의 색상 필드가 적용됩니다. 주어진 데이터 포인트 색상으로 선의 색상을 지정하려면 ***MultiColorLine = True*** 를 설정하십시오. 차트는 인접한 점 사이의 색상 그라디언트를 보간하며, ***IndividualPointSizes = True*** 를 설정하면 포인트의 ***sizeFactor*** 필드가 적용됩니다. 이 계수는 ***PointStyle.Size*** 에 정의된 크기를 곱합니다.

8.7 SurfaceGrid- 및 SurfaceMeshSeries3D

SurfaceGridSeries3D 및 ***SurfaceMeshSeries3D***를 사용하면 데이터를 3D 표면으로 시각화할 수 있습니다. ***SurfaceGridSeries3D***에서 노드는 X 및 Z 차원에서도 동일한 간격으로 배치되는 반면 ***SurfaceMeshSeries3D***에서는 표면 노드를 3D 공간에 자유롭게 배치하여 표면을 거의 모든 모양으로 뒤틀 수 있습니다. 두 시리즈 모두 등고선 및 와이어 프레임 렌더링 가능합니다.



왼쪽: IntensityGridSeries, 오른쪽: IntensityMeshSeries.

Fill 속성을 사용하여 표면의 채우기 스타일을 선택합니다. ***ContourPalette*** 속성을 사용하면 높이 채색을 위한 색상 단계를 정의할 수 있으며, ***ContourPalette*** 는 ***와이어 프레임 메시*** 및 ***윤곽선***에도 사용할 수 있습니다.

윤곽 팔레트에 대해 무제한 단계 수를 정의할 수 있습니다. 각 단계에는 높이 값과 해당 색상이 있으며, 팔레트에는 ***MinValue***, ***유형 및 단계*** 특성이 포함됩니다. ***유형***에는 ***균일 및 그라데이션***의 두 가지 선택 사항이 있습니다.

표면 그리드 데이터 설정

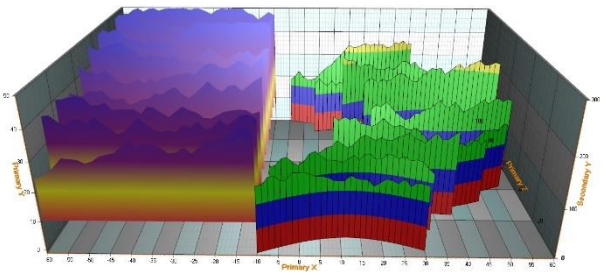
- ***RangeMinX*** 및 ***RangeMaxX*** 속성을 사용하여 할당된 X 축의 X 범위를 설정합니다.
- ***RangeMinZ*** 및 ***RangeMaxZ*** 속성을 사용하여 할당된 Z 축의 Z 범위를 설정합니다.
- ***SizeX*** 및 ***SizeZ*** 속성을 설정하여 그리드 크기를 열과 행으로 지정합니다.
- 모든 노드에 대해 Y 값을 설정합니다.

표면 메시 데이터 설정

- ***SizeX*** 및 ***SizeZ*** 속성을 설정하여 격자 크기를 열과 행으로 지정합니다.
- 모든 노드에 대해 X, Y 및 Z 값을 설정합니다.

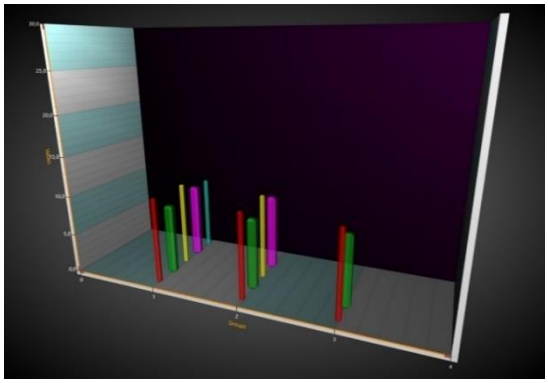
Invalidate Data() 방법을 호출하여 그리드/메시를 새로 고칩니다.

8.8 WaterfallSeries3D



WaterfallSeries3D 를 사용하면 데이터가 영역 스트립으로 시각화됩니다. ***SurfaceGridSeries3D*** 와 같이 영역을 채우고, 와이어 프레임으로, 윤곽선을 그릴 수 있으며, Y 차원에서 영역은 ***BaseLevel*** 속성 값에서 시작됩니다. 노드 데이터는 ***SurfaceMeshSeries3D*** 와 같이 설정할 수 있습니다. ***WaterfallSeries3D*** 는 특히 전통적인 3D 스펙트럼을 표현하는 데 유용합니다.

8.9 BarSeries3D



BarSeries3D 는 막대 데이터를 3D 로 시각화할 수 있습니다. Bar 시리즈 데이터는 ***x, y, z*** 및 ***텍스트*** 필드를 포함하는 ***BarSeriesValue3D*** 구조로 추가할 수 있으며, ***BarSeries3D*** 에는 막대 모양을 제어하기 위한 ***Shape*** 속성이 있습니다. 또한 일부 셰이프에서는 ***CornerPercentage*** 를 사용하여 모서리 라운딩을 변경하고 ***DetailLevel*** 을 사용하여 시각적 품질을 변경할 수 있습니다.

Bar 시리즈는 View3D 의 ***BarViewOptions*** 속성에서 사용할 수 있는 많은 옵션으로 그룹화할 수 있습니다. ***BarViewOptions.ViewGrouping*** 은 3D 보기에서 막대가 그룹화되는 방식을 제어합니다.

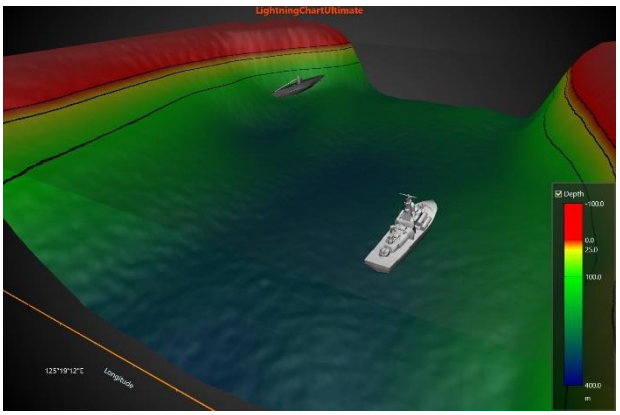
8.10 MeshModels

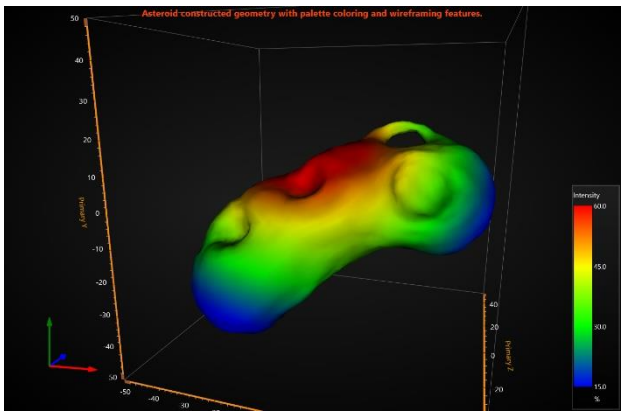
MeshModel 을 사용하면 외부 3D 모델 편집기의 3D 모델을 LightningChart View3D 에 삽입할 수 있습니다. 모델은 3D 모델링 앱 및 게임 엔진의 일반 형식인 OBJ 형식으로 가져올 수 있으며, ***MeshModel*** 생성은 .obj

파일의 정점 색상을 지원합니다. 정점 위치는 x, y 및 z (XYZRGBA) 뒤의 빨간색, 녹색, 파란색 및 알파 값을 지원합니다.

- 파일에서 모델을 로드하려면 경로와 파일 이름을 ***ModelFileName*** 속성에 설정하거나 ***LoadFromFile*** 방법을 사용합니다. 파일에서 모델을 로드할 때 동일한 경로에 있는 경우 텍스처 채우기도 로드되고 MTL 파일 및 이미지 파일에 액세스 할 수 있습니다.
- 스트림에서 모델을 로드하려면 ***LoadFromStream*** 방법을 사용합니다. 스트림 읽기 방법은 지역과 재질만 읽지만 텍스처는 읽지 않습니다.
- 리소스에서 모델을 로드하려면 ***LoadFromResource*** 방법을 사용합니다.

MeshModel 객체 ***위치***는 할당된 X, Y 및 Z 축을 따릅니다. ***Rotation*** 속성을 편집하여 모델을 회전할 수 있으며, ***크기***는 원래 모델 크기에 대한 요소 모음이며 축 범위 또는 3D 세계 치수를 따르지 않는 ***크기*** 속성으로 정의할 수 있습니다.





기본적으로 모델은 OBJ 모델의 색상으로 렌더링 됩니다. 채우기를 표시하려면 ***채우기***를 활성화합니다. 와이어 프레임을 표시하려면 ***WireFrame*** 을 활성화하고 ***WireFrameLineColor*** 에서 선호하는 선 색상을 설정합니다.

모델의 정점에 사용자 지정 색상을 적용하려면 모델이 생성된 후 ***UpdateFillColors(int [] colors)*** 방법을 사용하고 이를 확인하기 위해 ***GeometryConstructed*** 이벤트를 사용합니다. ***UpdateFillColors*** 를 주기적으로 호출하여 실시간 색상 업데이트를 적용할 수도 있습니다. 이 방법에는 꼭지점 위치(***X.Length***) 길이가 동일하며, 각 정점에 대해 하나의 색상,.ARGB 색상 배열이 필요합니다.

일부 모델은 역 와인딩 순서로 제작되므로 컬링으로 인해 보이지 않습니다. 모델이 제대로 나타나지 않을 경우 ***Cull*** 설정을 ***Clockwise***, ***CounterClockwise***, ***None*** 으로 변경하십시오.

정점에서 프로그래밍 방식으로 MeshModel 생성

MeshModel 지오메트리는 프로그래밍 방식으로 생성될 수도 있습니다. 회전, 크기 조정 및 위치 지정 속성 등은 이벤트뿐만 아니라 로드된 객체에 대해 작동하는 것과 유사한 방식으로 정점에서 프로그래밍 방식으로 생성된 ***MeshModel*** 에도 적용됩니다.

다음 ***Create*** 방법을 사용할 수 있습니다.

- *Create(positions, colors, indices)*
- *Create(positions, colors, normals, indices)*
- *Create(positions, textureCoordinates, bitmap, textureWrapMode, indices)*
- *Create(positions, normals, textureCoordinates, bitmap, textureWrapMode, indices)*

인덱스 배열(***indices***) 매개 변수는 선택 사항입니다. 제공된 경우 주어진 배열에서 사용할 정점, 색상, 조명 법선 및 텍스처 좌표를 정의하며, 인덱스를 사용하면 여러 삼각형 간에 동일한 정점이 공유될 때 리소스가 절약됩니다.

8.11 VolumeModels

VolumeModels 는 직접 볼륨 렌더링을 통해 볼륨 데이터 시각화를 위한 도구입니다. ***VolumeModel*** 은 내부 볼륨 데이터를 가져와서 시각화합니다. LightningChart 의 볼륨 렌더링 엔진은 ***Volume Ray Casting*** 을 기반으로 합니다.

이미지는 데이터 세트 내부를 이동하는 광선의 트랙을 따라 볼륨 데이터 샘플링을 통해 알고리즘에 의해 생성됩니다. ***Volume Ray Casting*** 에 대한 하드웨어 가속을 간단하게 실현하려면 볼륨 객체에 대한 경계를 생성해야 합니다. 일반적으로 큐브로 표시되는데, 인공물이 없는 높은 렌더링 품질과 상호 교환 가능한 광선 기능 사용이 이 기술의 주요 장점입니다.

RayFunction 은 매우 높은 수준의 유연성을 제공하는 알고리즘의 핵심입니다. 이 기술은 데이터가 샘플링되고 결합되는 방식을 지정하기 때문에 무척 강력하며, 이를 특징 추출에 매우 유용한 도구입니다.

주의! VolumeModel 은 DirectX 11 렌더러를 사용하는 경우에만 사용할 수 있습니다.

데이터 로딩

로드 함수와 생성자를 사용하면 데이터를 조각 모음(Data 속성과 유사) 또는 조각이 있는, 폴더 경로가 있는 문자열로 제공할 수 있습니다. 데이터는 도구로 생성된 텍스처 맵으로 제공될 수도 있으며, 텍스처 맵은 슬라이스로 구성되어 있지만, 보충에는 그림의 슬라이스 수에 대한 추가 정보가 필요합니다.

ChartTools.CreateMap 함수를 통해 텍스처 맵을 생 할 수 있습니다. 텍스처 맵의 직접 입력은 매우 큰 데이터 세트에 대한 앱의 시작 속도를 높이는 데 사용됩니다. ***LoadFromSlices()***와 같은 로드 함수 중 하나를 통해 ***VolumeModel*** 에 데이터를 제공할 수 있습니다.

속성

VolumeModel 에는 LightningChart 에 있는 3D 개체의 일반적인 속성(예: ***Visible, Rotation, Size, Position, AllowUserInteraction 및 HighLight***)이 포함됩니다. 또한 개체에는 볼륨 렌더링 엔진이 처리하는 방법을 정의하는 특정 속성이 있습니다.

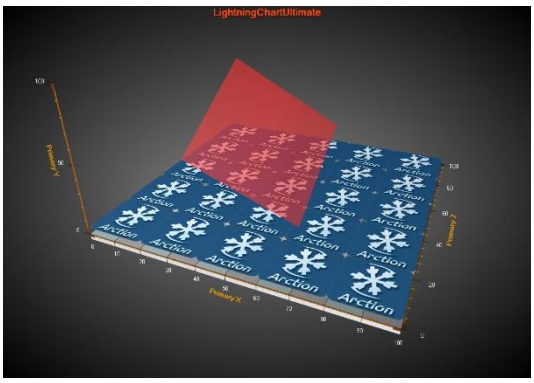
RayFunction 속성을 사용하면 LightningChart 볼륨 렌더링 엔진에서 사용할 수 있는 세 가지 복셀 샘플링 및 구성 방법 중 하나를 선택할 수 있습니다.

- ***RayFunction.Accumulation*** 은 가능한 한 많은 데이터를 수집하고 결합합니다. 이 기술로 생성된 시각화는 반투명 젤처럼 보입니다. 아래 그림은 의료 데이터 세트를 시각화하는 ***RayFunction.Accumulation*** 애플리케이션의 예를 보여줍니다.
- ***RayFunction.MaximalIntensity*** 는 광선에 의해 샘플링된 가장 밝은 값만 고려합니다. 시각적으로 X-ray 이미지와 매우 유사한 결과를 제공하며, 개체의 내부 구조에 대한 추가 정보를 얻을 수 있습니다. 아래는 골격 시각화 및 초음파 간섭 시뮬레이션을 위한 ***RayFunction.MaximalIntensity*** 앱입니다.
- ***RayFunction.Isosurface*** 는 다각형 모델 렌더링 처럼 보이는 방식으로 모델 표면을 그립니다. 결과는 ***간접 볼륨 렌더링***으로 생성된 결과와 매우 유사합니다. 그림은 인간 두개골 CT 시각화 및 물 흐름 시뮬레이션을 위한 ***RayFunction.Isosurface*** 애플리케이션의 예를 보여줍니다.

Volume Rendering Engine 은 ***VolumeModel*** 에 속성으로 ***Threshold*** 범위를 적용할 수 있습니다. 모든 색상 채널에 대해 별도의 경계가 있습니다. 복셀은 해당 색상 값이 모든 채널에서 높은 경계보다 낮고 낮은 경계보다 높은 경우에만 시각화되며, 허용되는 영역은 보이지 않습니다. 이 속성은 마우스 적중 테스트에서 고려되지 않습니다.

SliceRange 속성은 ***VolumeModel*** 의 일부를 잘라낼 수 있습니다. 개체의 내부 구조를 탐색하는 데 매우 유용한 도구이고, ***SliceRange*** 는 ***Min*** 과 ***Max*** 라는 두 개의 경계를 포함하며 둘 다 세 개의 포인팅 부동 값으로 표시됩니다.

8.12 Rectangle3D 개체

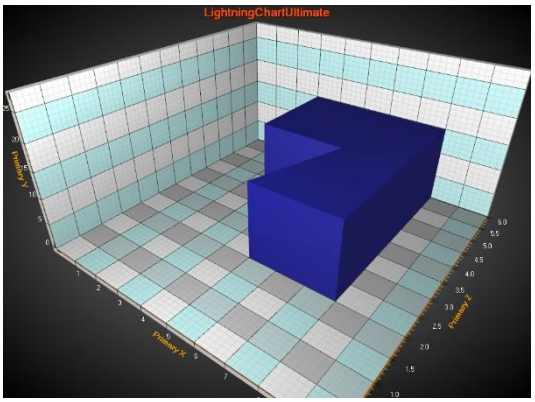


Rectangle3D 를 사용하면 모든 위치에서 모든 크기, 모든 각도로 회전된 사각형을 표시할 수 있습니다. ***View3D.Rectangles*** 목록에 추가할 수 있으며, 직사각형은 ***View3D.Dimensions*** 에 따라 크기를 정의하여 평면 역할을 할 수도 있습니다.

3D 표준 치수(X, Y 또는 Z 축 값이 아님)의 ***크기***를 ***너비*** 및 ***높이***로 설정합니다. X, Y 및 Z 축 값에 정의된 ***Center*** 속성을 통해 중심점을 설정하고, ***Rotation*** 속성은 회전 각도를 지정합니다.

채우기 설정은 ***채우기*** 속성을 통해 수정할 수 있습니다. 단색 및 비트 맵 채우기를 사용할 수 있습니다. 비트 맵 채우기를 사용하려면 ***Image*** 에서 비트 맵을 설정하고 ***UseImage*** 를 활성화하고, ***Fill.Layout = Stretch*** 로 설정하면 비트 맵이 늘어나 사각형을 채웁니다. ***Fill.Layout = Tile*** 을 설정하면 동일한 비트 맵이 바둑판 식으로 배열되어 사각형을 채웁니다. 타일 수는 ***Fill.TileCountWidth*** 및 ***Fill.TileCountHeight*** 속성을 통해 변경할 수 있습니다.

8.13 Polygon3D objects



Polygon3D 객체를 사용하면 주어진 Y 범위로 늘어난 2D 다각형을 표시할 수 있습니다. ***View3D.Polygons*** 목록에 추가할 수 있습니다.

X 및 Z 축 값에 다각형 경로를 정의하고 ***Points*** 배열에 저장합니다. ***YMin*** 및 ***YMax*** 값으로 Y 범위를 설정하고, ***Material.Diffuse*** 는 사각형의 기본 색상을 제어합니다. ***Rotation.X, Rotation.Y 및 Rotation.Z*** 를 사용하여 다각형을 다른 각도로 회전합니다.

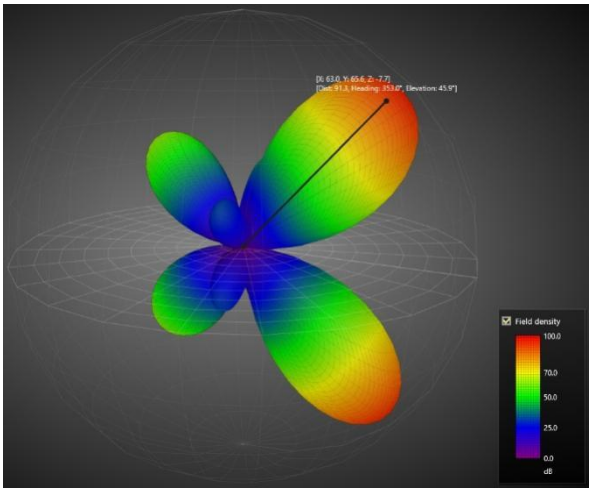
8.14 좌표계 변환 체계

다음 좌표계 변환 체계는 View3D 사용을 보완하는 ***CoordinateConverters*** 명칭 공간에서 사용할 수 있습니다.

- Cartesian 3D <-> Spherical 3D
- Cartesian 3D <-> Cylindrical 3D

SphericalCartesian3D

SphericalCartesian3D 변환기 클래스는 구형 및 3D 데카르트 좌표 간을 변환합니다.



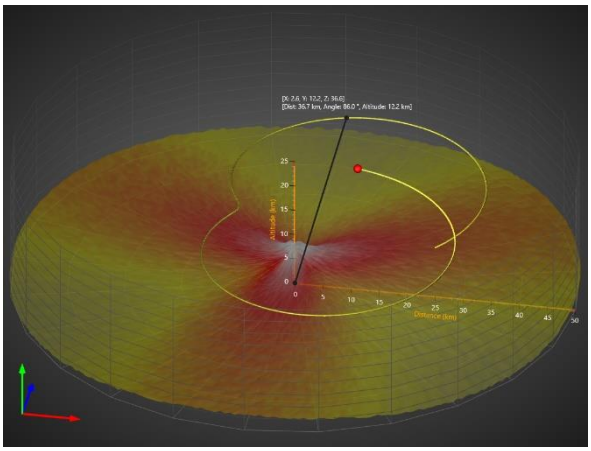
구형 데이터 포인트는 다음 필드를 포함하는 SphericalPoint 객체에 의해 정의됩니다.

- ***Distance***: 원점으로부터의 거리입니다. (0,0,0)
- ***ElevationAngle***: 고도 각도. XZ 평면에서 측정한 고도 또는 고지라고 합니다. 고도 각도는 90 도이며, 경사각입니다. XZ 평면은 참조 평면입니다.
- ***HeadingAngle*** : 헤딩 각도. 방위각 및 절대 방위라고도 합니다.

SphericalPoint 를 데카르트 좌표로 변환하려면 ***SphericalCartesian3D.ToCartesian ()*** 방법을 사용하십시오. 데카르트 점을 구형 점으로 변환하려면 ***SphericalCartesian3D.ToSpherical ()*** 방법을 사용하십시오.

CylindricalCartesian3D

원통형 좌표와 3D 데카르트 좌표 간을 변환하는 변환기 클래스입니다.



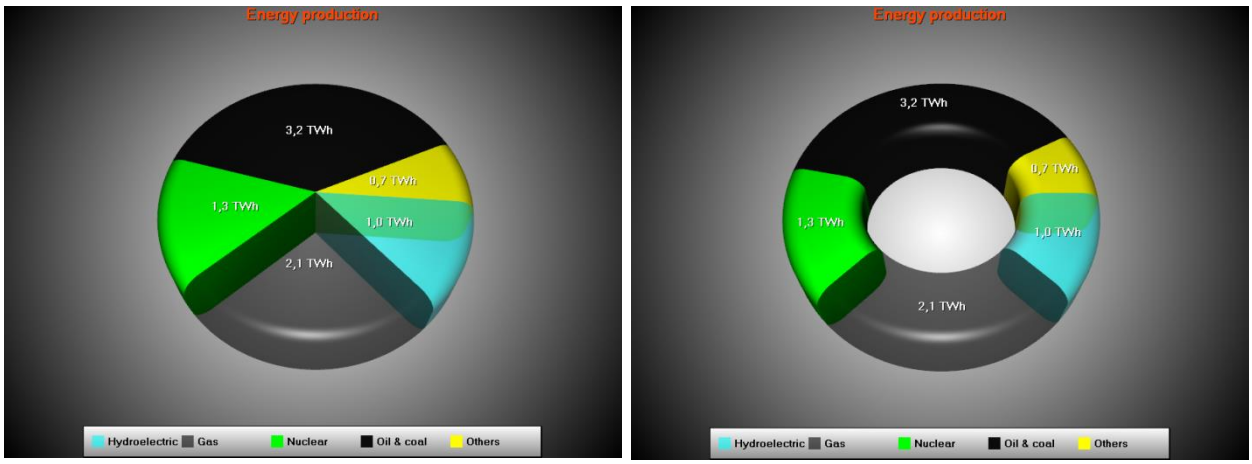
원통형 점은 다음 필드를 포함하는 ***CylindicalPoint*** 개체에 의해 정의됩니다.

- ***Distance***: XZ 평면을 따른 거리입니다.
- ***Y***: Y 값입니다.
- ***Angle***: 방위각, 방위각 및 절대 방위라고도 합니다.

CylindricalPoint 를 데카르트 좌표로 변환하려면 ***CylindricalCartesian3D.ToCartesian()*** 방법을 사용하십시오. 데카르트 점을 원통형 점으로 변환하려면 ***CylindricalCartesian3D.ToCylindrical ()*** 방법을 사용하십시오.

9. ViewPie3D

ViewPie3D 는 데이터를 3D 로 원형 및 도넛형 차트로 표시합니다.



Style 속성을 사용하여 차트 유형 (***파이*** 또는 ***도넛***)을 선택합니다. View3D 와 유사하게 ***ZoomPanOptions*** 속성 트리를 사용하여 확대/축소, 이동 및 회전을 제어합니다. (7.18 장 참조)

카메라 속성은 시점을 제어합니다. (7.4 장 참조) ***LightingScheme*** 속성으로 미리 정의된 조명 설정을 선택할 수 있습니다. ***재질*** 속성 및 하위 속성을 사용하여 일반적인 3D 표면 모양과 광택을 조정합니다.

DonutInnerPercents 를 사용하여 도넛 내부 반경을 설정하고 ***Rounding*** 을 사용하여 가장자리 라운딩 반경을 조정하고 ***StartAngle*** 을 사용하여 원형을 회전하고 ***두께***를 사용하여 원형 두께를 조정합니다. ***ExplodePercents*** 는 슬라이스의 ***분해가 참으로*** 설정된 경우 분해된 원형 슬라이스가 얼마나 멀리 떨어져 있는지 조정합니다.

TitlesStyle 은 ***제목***, ***값*** 또는 ***백분율*** 중 하나의 파이 조각 텍스트를 설정합니다. 예를 들어 ***TitlesNumberFormat*** 을 "0.0 TWh"로 편집하여 끝에 단위를 포함합니다.

PieSlice

원형 차트 데이터는 ***Values*** 컬렉션에 저장됩니다. 목록의 각 항목은 ***PieSlice*** 유형입니다. ***Value*** 속성에서 데이터 값을 편집하며, 제목 문자열을 ***Title.Text*** 속성으로 설정합니다. ***TitleAlignment = Outside*** 를 정의하면 제목이 원형 외부에 그려집니다.

코드에 파이 슬라이스 설정은 다음과 같습니다.

```
PieSlice slice1 = new PieSlice("Hydroelectric",
Color.FromArgb(150, Color.Aqua), 1.0, chart.ViewPie3D, true);
```

10. ViewPolar

ViewPolar 를 사용하면 극좌표 형식으로 데이터를 시각화할 수 있습니다. 데이터 포인트 위치는 각도 값과 진폭에 의해 결정되며, (ViewXY 에서 각도를 X 로, 진폭을 Y 로 비교) 극좌표 보기에는 확대/축소 및 이동 기능도 있습니다.

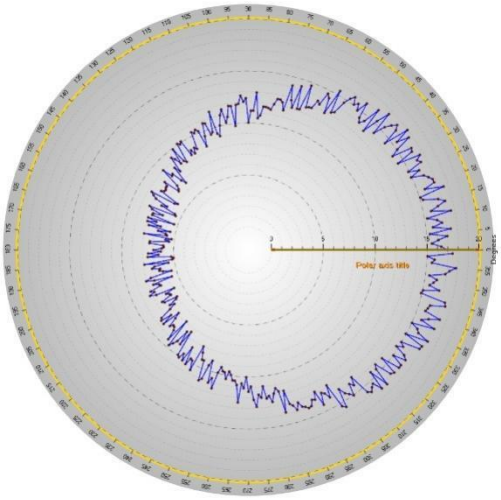
10.1 축

축 목록 속성을 통해 극축을 정의할 수 있습니다. 동일한 차트에서 여러 축을 사용할 수 있으며, 계열의 ***AssignPolarAxisIndex*** 속성을 설정하여 이러한 축에 계열을 할당할 수 있습니다. 축은 각도 스케일과 진폭 스케일을 모두 나타냅니다. 그렇지 않으면 극축은 ViewXY 축과 매우 유사합니다.

AngleOrigin 을 사용하여 각도 스케일의 회전 각도를 설정합니다. ***AmplitudeAxisAngle*** 을 사용하여 진폭 축 위치를 회전하고, 진폭 스케일 각도는 절대 각도 (***AmpitudeAxisAngleType = Absolute***) 또는 상대 (***AmpitudeAxisAngleType = Relative***)로 각도 눈금의 각도에 설정할 수 있습니다.

MajorDivCount 로 진폭 분할 수를 설정하고 ***MajorDiv*** 속성으로 분할 크기를 설정합니다. 진폭 스케일은 그에 따라 조정됩니다. (***MaxAmplitude*** 업데이트) ***MinorDivCount*** 로 진폭 마이너 디비전 카운트를 설정하고, 기본적으로 차트는 가능한 한 거의 많은 각도 분할을 포함합니다. 각도 분할을 제어하려면 ***AngularAxisAutoDivSpacing*** 을 ***False*** 로 설정한 후, 차트는 ***AngularAxisMajorDivCount*** 분할 수를 시도합니다. 차트 공간이 너무 작아 모든 분할 및 레이블을 렌더링 할 수 없는 경우 적합할 수 있는 더 낮은 분할 수를 사용합니다.

10.2 PointLineSeriesPolar



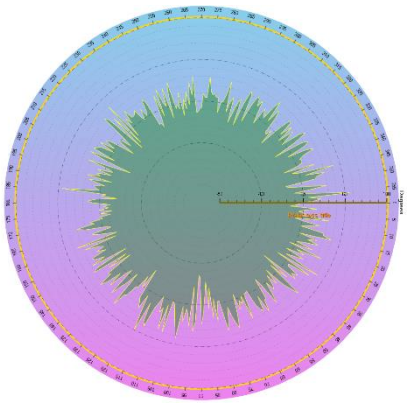
ViewPolar 의 ***PointLineSeriesPolar*** 는 선, 점 그룹 또는 점선을 그리는 데 사용할 수 있습니다. ***LineStyle*** 및 ***PointStyle*** 속성에서 많은 선 및 점 스타일을 사용할 수 있습니다.

라인 컬러링은 팔레트를 지원합니다. ***ColorStyle*** 속성을 사용하여 팔레트 색상이 적용되는 방식을 선택할 수 있습니다.

- ***LineStyle***: 팔레트 채우기가 없습니다. LineStyle.Color 속성에 설정된 색상이 적용됩니다.
- ***PalettedByAngle***: 데이터 포인트 ***각도*** 필드가 색상을 결정합니다.
- ***PalettedByAmplitude*** : 데이터 포인트 ***진폭*** 필드는 색상을 결정합니다.
- ***PalettedByValue*** : 데이터 포인트 ***값*** 필드는 색상을 결정합니다.

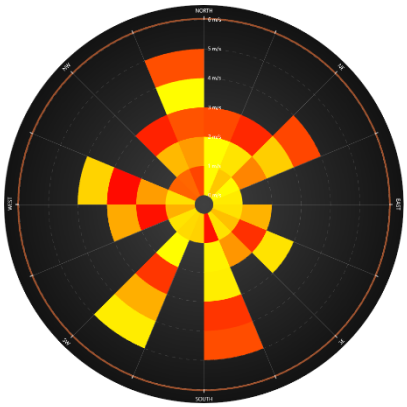
ValueRangePalette 속성을 사용하여 색상 및 값 단계를 정의합니다. ViewXY 및 View3D 시리즈와 유사하게 작동합니다.

10.3 AreaSeries



AreaSeries 를 사용하면 채워진 영역 스타일로 데이터를 시각화할 수 있습니다. 선 내부/아래 영역이 컬러로 표시된다는 점을 제외하면 PointLineSeries 와 유사하게 작동합니다. 가장자리의 선 스타일은 ***LineStyle*** 속성으로 편집할 수 있으며, ***FillColor*** 속성으로 채우기를 변경할 수 있습니다.

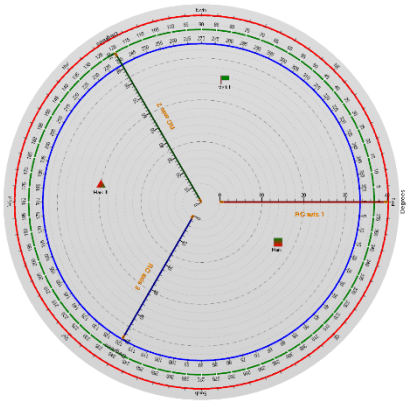
10.4 섹터



일부 각도 또는 진폭 범위를 나타내도록 ***섹터***를 정의할 수 있습니다. ***MinAmplitude*** 및 ***MaxAmplitude*** 속성으로 진폭 범위를 정의합니다.

BeginAngle 및 ***EndAngle*** 을 사용하여 각도 범위를 정의합니다. 마우스로 드래그하여 섹터를 이동합니다.

10.5 마커



마커는 특정 위치에서 특정 데이터 값을 표시하는 데 사용할 수 있습니다. ***AssignPolarAxisIndex*** 를 설정하여 기본 축으로 마커를 할당하고, ***Amplitude*** 및 ***AngleValue*** 속성을 정의하여 배치합니다. 선호하는 모양을 갖도록 ***Symbol*** 을 편집하고 ***Label*** 속성으로 마커 텍스트를 정의합니다.

마커는 마우스로 드래그하여 이동할 수 있고, ***SnapToClosestPoint*** 를 ***선택됨*** 또는 ***모두***로 설정하여 끌 때 가장 가까운 데이터 포인트 스냅을 활성화합니다. ***선택된*** 트랙은 이 마커가 ***SetSnapSeries()*** 방법으로 스냅 하도록 설정된 시리즈만 추적합니다. ***All*** 은 모든 시리즈를 추적합니다.

11. ViewSmith

Smith 차트는 일반적으로 임피던스 측정 및 임피던스 매칭 앱의 전자 제품에 사용됩니다. Smith 차트는 데이터를 실수 및 허수 값**(R + jX)**으로 표시합니다.

조건

임피던스 = **Z** = **R** + j**X**

R = 유도 저항, 실수 값

X = 유도 저항, 허수 값

X > 0: 용량
X < 0: 유도

데이터 위치는 원형 Real 및 Imaginary 로그-로그 스케일의 각도에 의해 2D 플롯에서 결정됩니다.

11.1 축

Smith 차트에는 확장 속성 트리 ***축***을 통해 구성할 수 있는 실제 축이 하나만 있습니다. 대부분의 속성은 ViewPolar의 좌표축 및 ViewXY의 좌표축과 동일하며, ViewSmith 조정과 관련된 고급 속성도 있습니다.

GridDivCount는 실제 축의 원형 그리드 선과 가상 스케일의 로그 그리드 선의 양을 정의합니다.

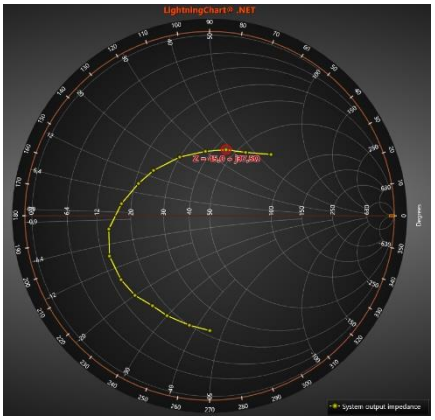
GridImg 및 ***GridReal*** 속성은 ***실수*** 또는 ***허수*** 배율로 그리드 선을 사용자 지정하는 역할을 합니다. 또한 ***Visible*** 속성을 사용하여 그리드를 숨길 수 있으므로 사용자는 그리드 중 하나를 숨기고 다른 작업을 계속할 수 있습니다.

RealAxisLineVisible 속성은 축 선을 숨깁니다.

ShowAbsoluteValues 속성은 스케일 (절대 또는 정규화)에 있는 값을 정의합니다.

ClipGridInsideGraph는 눈금 선을 차트 원 외부에 표시합니다.

11.2 PointLineSeries



ViewSmith의 ***PointLineSeries***는 ViewPolar에서와 같이 선, 점 그룹 또는 점선을 그리는 데 사용할 수 있습니다. ***LineStyle*** 및 ***PointStyle*** 속성에서 많은 선 및 점 스타일을 사용할 수 있습니다.

아래 코드는 Smith 차트 컬렉션에 한 세트의 데이터 포인트를 추가합니다.

```
SmithSeriesPoint[] m_aPoints;
PointLineSeriesSmith Series = new PointLineSeriesSmith(m_chart.ViewSmith, axis);
//Create data for series
m_iCount = 5000;
m_aPoints = new SmithSeriesPoint[m_iCount];
for (int i = 0; i < m_iCount; i++)
{
          // Sine from left to right
          m_aPoints[i].RealValue = i * (MaxReal / m_iCount);
          m_aPoints[i].ImgValue = Math.Sin(0.01 * i)/Math.PI * MaxReal;
}
Series.Points = m_aPoints;
//Add series to chart
m_chart.ViewSmith.PointLineSeries.Add(Series);
```

11.3 마커

마커는 특정 위치에서 특정 데이터 값을 표시하는 데 사용할 수 있습니다. 마커는 마우스로 드래그하여 이동할 수 있으며, 이 속성은 ViewPolar의 마커와 동일한 정의를 갖습니다 (10.8 장 참조).

ImgValue 및 ***RealValue*** 속성을 정의하여 배치합니다. 선호하는 모양을 가지도록 ***기호***를 편집하고 ***Label*** 속성으로 텍스트를 정의합니다.

12. 범례 상자

ViewXY 는 동일한 그래프에서 여러 범례 상자를 지원합니다. ***ViewXY.LegendBoxes*** 컬렉션에 이러한 범례 상자를 삽입하고, 다른 뷰(3D, Polar, Smith, Pie)의 범례 상자는 ViewXY 의 범례 상자와 거의 비슷합니다. 그러나 그래프 당 하나의 범례 상자만 허용되며, 또한 축을 세그먼트로 나눌 수 없기 때문에 세그먼트 기반 속성이 존재하지 않습니다.

범례 상자에서 확인란을 표시하거나 숨기려면 ***ShowCheckBoxes*** 속성을 사용합니다. ***CheckBoxColor*** 및 ***CheckMarkColor*** 는 확인란의 모양을 변경하는 데 사용할 수 있으며 ***CheckBoxSize*** 는 상자의 크기를 픽셀 단위로 제어합니다.

```
_chart.ViewXY.LegendBoxes[0].ShowCheckboxes = true;
_chart.ViewXY.LegendBoxes[0].CheckBoxColor = Colors.Green;
_chart.ViewXY.LegendBoxes[0].CheckMarkColor = Colors.Blue;
_chart.ViewXY.LegendBoxes[0].CheckBoxSize = 15;
```

아이콘을 숨기려면 ***ShowIcons = False*** 를 설정하십시오. 특정 시리즈가 범례 상자에 나열되지 않아야 하는 경우 해당 시리즈에 대해 ***series.ShowInLegendBox = False*** 를 설정하고, ***IntensityGrid*** 또는 ***-Mesh*** 의 팔레트 배율을 숨기려면 ***IntensityScales.Visible = False*** 를 설정합니다. 크기를 조정하려면 ***ScaleSizeDim1*** 및 ***ScaleSizeDim2*** 속성을 설정하고, 눈금의 경계와 제목의 위치도 수정할 수 있습니다. View3D 에는 ViewXY 의 ***IntensityScales*** 대신 ***SurfaceScales*** 속성이 있습니다.

위치 제어

범례 상자는 자동 또는 수동으로 배치할 수 있습니다. 자동 배치를 통해 그래프 세그먼트의 왼쪽/위쪽/오른쪽/아래쪽 또는 여백에 정렬할 수 있으며, 위치 속성으로 위치를 제어합니다. 위치 옵션은 ***TopCenter, TopLeft, TopRight, LeftCenter, RightCenter, BottomLeft, BottomCenter, BottomRight, Manual*** 입니다.

뷰가 여러 세그먼트로 분할된 경우 해당 뷰가 속한 세그먼트에 따라 범례 상자를 정렬할 수 있습니다. (이를 제어하려면 ***SegmentIndex*** 사용) 세그먼트 기반 제어에는 ***SegmentTopLeft, SegmentTopCenter, SegmentTopRight, SegmentBottomLeft, SegmentBottomCenter, SegmentBottomRight, SegmentLeftMarginCenter, SegmentRightMarginCenter*** 옵션이 있습니다.

Offset 속성은 ***Position*** 속성에 의해 결정된 위치에서 주어진 양만큼 위치를 이동합니다.

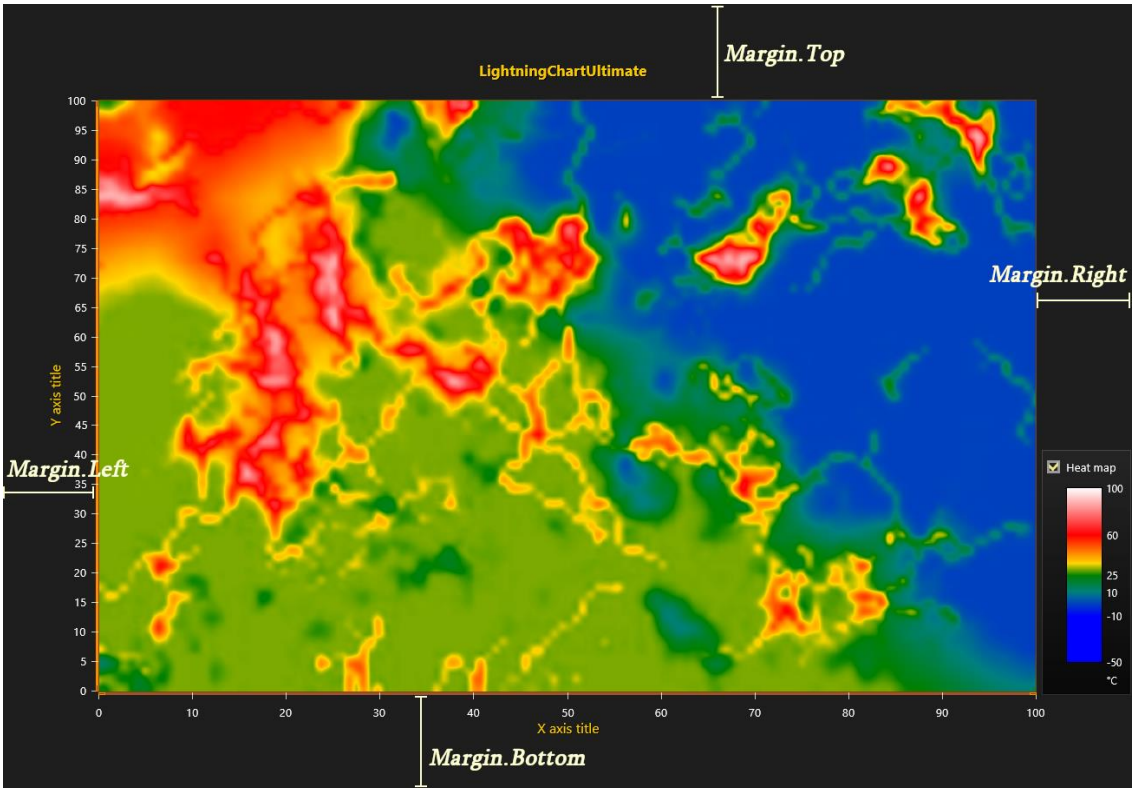
```
// Setting legend box position, offset shifts from RightCenter position
chart.ViewXY.LegendBoxes[0].Position = LegendBoxPositionXY.RightCenter;
chart.ViewXY.LegendBoxes[0].Offset = new PointIntXY(-15, -70);
```

수동 위치 지정은 범례 상자의 왼쪽 위 모서리에서 뷰의 왼쪽 위 모서리까지의 오프셋을 계산합니다. 이는 그래프 영역의 상단에서 계산되는 ***TopLeft*** 옵션과 다르며, 범례 상자를 이동하거나 크기를 조정할 때 해당 ***위치***는 ***수동***으로 설정되고 ***오프셋*** 속성은 새 위치를 반영하도록 업데이트됩니다.

위치를 '***수동***' 이외의 옵션으로 다시 설정할 때까지 자동 범례 상자 정렬이 비활성화됩니다. ***위치*** 옵션 간에 전환할 때 오프셋이 업데이트되지 않기 때문에 범례 상자가 때때로 사라지는 것처럼 보일 수 있습니다. (보기 외부에 있음) ***오프셋***을 다시 0, 0 으로 설정하여 이 문제를 해결할 수 있습니다.

13. 여백

여백은 그래프 영역 주변의 빈 공간이며, 보기의 내용은 여백 외부에서 자동으로 잘립니다. 차트 제목, 주석, 범례 상자를 제외한 모든 내용은 화면 좌표에 정의되어 있으므로 여백에도 자유롭게 배치할 수 있습니다. 1 픽셀 너비의 테두리 직사각형인 ***Border*** 를 그려 여백이 있는 위치를 표시할 수 있습니다. 사각형의 색상은 ***Border.Color*** 를 통해 변경할 수 있으며, 기본적으로 테두리는 ViewXY 에서만 볼 수 있습니다.



AutoAdjustMargins 가 활성화되면 모든 축과 차트 제목에 충분한 공간이 있도록 그래프 크기가 조정됩니다. ***비활성화***되면 ***여백*** 속성이 적용되어 여백을 수동으로 설정할 수 있습니다.

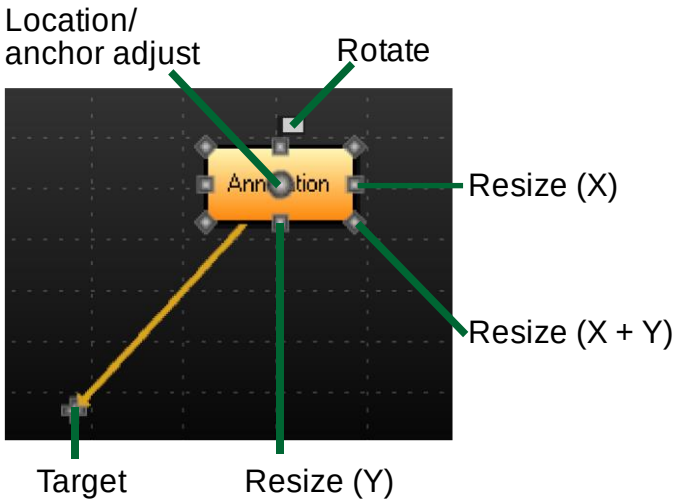
런타임 동안 자동 및 수동 여백에 모두 적용되는 뷰의 ***GetMarginsRect*** 메서드를 호출하여 여백 사각형(픽셀)을 검색할 수 있으며, 화면 좌표 기반 계산 또는 개체 배치를 수행해야 할 때 유용합니다.

MarginsChanged 이벤트는 여백 사각형이 크기 조정 등으로 변경된 경우 트리거 되도록 설정할 수 있습니다.

14. 주석

주석을 사용하면 차트 영역의 모든 위치에 마우스 대화형 텍스트 레이블 또는 그래픽을 표시할 수 있습니다. 주석은 마우스로 이동, 크기 조정, 회전, 대상 및 위치 변경 등을 수행할 수 있으며, 또는 코드로 제어할 수 있습니다. 주석은 다양한 스타일과 모양으로 렌더링 할 수 있으므로 사용자 정의 그래픽을 화면에 렌더링해야 하는 경우에도 유용합니다. ***ViewXY.Annotations*** 컬렉션에서 ***AnnotationXY*** 개체를 만듭니다.

주석 위로 마우스를 이동하면 마우스 상호 작용 편집 상태가 되어 주석을 재배치하고 크기를 조정하고 회전하고 화살표가 가리키는 위치를 결정할 수 있습니다.



14.1.1 대상 및 위치 제어

대상은 화살표의 끝점, 화살표 또는 설명 선 끝이 가리키는 지점이며, 목표는 축 값 또는 화면 좌표에서 설정할 수 있습니다. ***TargetCoordinateSystem*** 을 사용하여 ***AxisValues*** 또는 ***ScreenCoordinates*** 중에서 선택하고, ***AxisValues*** 를 선택하면 ***TargetAxisValues*** 속성이 화살표 선이 가리키는 위치(화살표 선의 끝)를 설정합니다. 그리고 ***TargetScreenCoords*** 를 사용하여 화면 좌표로 설정할 수 있습니다.

위치는 화살표의 시작점이며, 화면 좌표, 축 값 또는 ***Target*** 에서의 상대 오프셋으로 설정할 수 있습니다. ***LocationCoordinateSystem*** 을 사용하여 선택하고 ***LocationScreenCoords***, ***LocationAxisValues*** 또는 ***LocationRelativeOffset*** 을 사용하여 선택한 방법으로 위치를 제어합니다. ***위치***는 텍스트 영역 회전의 중심점이기도 합니다.

Anchor 속성은 텍스트 영역이 ***위치***에 배치되는 방식을 제어합니다. ***Anchor.X*** = 0.5 및 ***Anchor.Y*** = 0.5 로 설정하면 화살표의 시작이 중간에 있으며, ***Anchor.X*** 0.1 및 ***Anchor.Y*** = 0.25 를 설정하면 다음 그림과 같이 화살표 시작이 왼쪽 상단 모서리 근처에 있습니다.

14.1.2 마우스를 사용하여 이동, 회전 및 크기 조정

대상에서 드래그하여 화살표 끝을 이동합니다. 새 ***위치***를 설정하려면 텍스트 영역에서 드래그하십시오. 라운드 위치/앵커 노드에서 드래그하여 ***앵커*** 및 ***위치*** 속성을 동시에 조정하여 텍스트 상자를 같은 위치에 유지할 수 있습니다.

Shift 키를 누른 상태에서 X 또는 Y 크기 조정 노드에서 드래그하면 대칭 작업이 사용되며 양쪽이 동시에 조정됩니다. 모서리 크기 조정 노드 (X + Y)에서 드래그하는 동안 ***Shift*** 키를 누른 상태에서 크기를 조정하면 종횡비가 유지되며, 회전 작업에서 ***Shift*** 키는 가장 가까운 15 도의 배수로 회전 각도를 스냅 합니다.

14.1.3 모양 조정

Style 속성을 설정하여 주석 모양을 선택합니다. 옵션은 ***Rectangle, RectangleArrow, RoundedRectangle, RoundedRectangleArrow, Arrow, Callout, RoundedCallout, Ellipse, EllipseArrow, Triangle 및 TriangleArrow*** 입니다.

화살표가 있는 스타일에서 ***ArrowLineStyle, ArrowStyleBegin 및 ArrowStyleEnd*** 를 사용하여 화살표 디자인을 제어하며, 화살표 끝 스타일에는 ***없음, 정사각형, 화살표, 원형 및 캘리퍼스*** 옵션이 있습니다.

채우기를 사용하여 주석 채우기를 수정합니다. 편집 상태 마우스-대화 형 노드의 모양은 ***NibStyle*** 에서 변경할 수 있으며, ***TextStyle*** 은 텍스트 영역 내의 글꼴 설정 및 텍스트 정렬을 제어합니다. ***BorderLineStyle*** 및 ***CornerRoundRadius*** 는 테두리 선 모양을 제어합니다.

크기 조정속성은 주석 텍스트 상자의 크기를 조정하는 방법을 제어합니다.

- ***Automatic*** 은 내용에 따라 크기를 조정하고 테두리에 ***AutoSizePadding*** 공간을 둡니다.

- ***AxisValuesBoundaries*** 를 사용하면 축 값으로 주석의 크기를 설정할 수 있습니다. ***AxisValuesBoundaries.XMin, XMax, YMin 및 YMax*** 를 사용하여 정의하십시오.
- ***ScreenCoordinates*** 는 화면 좌표로 설정을 활성화합니다. ***SizeScreenCoords.Height*** 및 ***Width*** 를 사용합니다.

Annotation3D 컬렉션을 사용하면 3D 장면에 주석을 추가할 수 있습니다. 일반적으로 X, Y 및 Z 치수를 사용하는 ***대상*** 및 ***위치*** 속성을 제외하고는 ViewXY 의 ***주석***과 유사합니다. 3D 에서 마우스로 ***대상***을 이동할 수 있습니다. 이동을 돕기 위해 주석은 마우스가 ***대상*** 노드 위에 있을 때 십자선을 표시합니다. ***ShowTargetCrosshair*** 속성을 ***Auto/On/Off*** 로 설정하고 ***TargetCrosshairLineStyle*** 에서 선 스타일을 조정합니다.

ViewPolar 및 ViewSmith 에서 ***주석***은 Polar(각도 및 진폭) 또는 Smith(실제 및 가상) 축 값으로 정의되는 ***대상*** 및 ***위치***를 제외하고는 ViewXY 의 ***주석***과 유사합니다. 축 값에 의한 ***크기 조정***은 적합하지 않으므로 크기 조정 속성에는 ***Automatic*** 및 ***ScreenCoordinates*** 값만 있습니다.

15. 내보내기 및 인쇄

이미지 내보내기

SaveToFile() 방법을 사용하여 차트를 .PNG, .BMP 및 .JPG 파일로 내보낼 수 있습니다. ***SaveToFile(...)*** 방법을 사용하면 해상도 감소 및 다듬기/앤티 앨리어싱 옵션을 사용하여 이미지 파일을 내보낼 수 있습니다. 스트림으로 내보내려면 ***SaveToStream()*** 방법을 사용하십시오.

ViewXY, ViewPolar 및 ViewSmith 는 .WMF, .EMF 및 .SVG 형식으로 내보낼 수도 있지만, View3D 및 ViewPie3D 는 현재 이를 지원하지 않습니다. 선택한 벡터 파일 형식에 ***SaveToFile*** 또는 ***SaveToStream*** 메서드를 사용합니다.

주의! 벡터 출력이 단순화되고 복잡한 포인트 스타일과 같은 모든 세부 사항이 일반 색상과 단순한 모양으로 표시될 수 있습니다. 벡터 출력에는 일부 비트 맵 요소가 포함될 수도 있습니다.

클립보드에 복사

CopyToClipboard(...)를 호출하여 차트를 클립 보드에 복사할 수 있습니다. ViewXY, ViewPolar 및 ViewSmith 는 ***CopyToClipboardAsEmf()*** 방법을 사용하여 벡터 형식으로 복사할 수 있습니다.

바이트 배열로 캡처

차트에는 ***CaptureToByteArray*** 방법이 있어 빠른 원시 이미지 데이터를 외부 구성 요소에 복사하거나 데이터를 추가로 처리합니다.

연속 프레임 쓰기용 출력 열

chart.OutputStream 속성을 사용하여 차트가 렌더링 된 프레임을 쓸 열을 설정합니다. 이 속성은 특히 ***헤드리스 모드***(24 장 참조)에서 차트에서 연속 프레임을 캡처하는 가장 빠른 방법입니다.

열은 원시 바이트 스트림이며 각 픽셀은 채널당 1 바이트 4 바이트로 설명됩니다. 채널의 순서는 렌더러 및 해당 설정에 따라 다르며 생성된 이미지 크기는 픽셀 단위의 차트 크기여야 합니다. ***GetLastOutputStreamFormat*** 및 ***GetLastOutputStreamSize*** 방법을 사용하여 마지막으로 작성된 이미지의 형식 및 출력 크기를 찾습니다.

주의! LightningChart 의 다른 속성과 달리 set stream 은 차트의 dispose 에 배치되지 않습니다.

인쇄

PrintPreview() 방법을 호출하여 인쇄 미리 보기 대화 상자를 열거나 ***Print()***를 호출하여 기본 설정으로 직접 인쇄합니다. 수동 설정으로 인쇄하려면 ***Print(...)***를 호출합니다. 인쇄 ViewXY, ViewPolar 및 ViewSmith 는 벡터 인쇄도 지원하며, Print(...) 방법에 매개 변수에 ***래스터*** 또는 ***벡터*** 형식을 제공합니다.

16. 차트 업데이트

모든 속성 또는 계열 데이터 값 변경으로 인해 LightningChart 제어는 다시 그려집니다. 다시 그릴 때마다 CPU 및 디스플레이 어댑터 오버 헤드가 발생하는데, 둘 이상의 속성이 프로그래밍 방식으로 동시에 변경되는 경우 ***BeginUpdate ()*** 및 ***EndUpdate()*** 방법 호출간에 일괄적으로 속성을 변경해야합니다. ***BeginUpdate()***는 ***EndUpdate()***가 호출될 때까지 제어 그리기를 중지하며, 보류 중인 ***BeginUpdate()*** 호출에 대한 내부 카운터가 있으며 동일한 양의 ***EndUpdate()*** 호출에 도달하면 ***EndUpdate()***가 제어를 다시 그립니다. 다음 예시는 컴퓨터에 최소한의 부하로 차트를 업데이트하는 방법을 보여줍니다.

```
chart.BeginUpdate(); //Disable redraws

//Add data to series
chart.ViewXY.SampleDataSeries[0].AddSamples(multiChannelSampleStream[0],
    false);
chart.ViewXY.SampleDataSeries[1].AddSamples(multiChannelSampleStream[1],
    false);
chart.ViewXY.SampleDataSeries[2].AddSamples(multiChannelSampleStream[2],
    false);

//Update point counter bar
chart.ViewXY.BarSeries[0].SetValue(0,1,(double)totalPointsCollected,"",
    false);

// Point counter label
chart.Title.Text = totalPointsCollected.ToString();

// Set monitoring scroll position to latest x
newestX = firstSampleTimeStamp + (double)(pointsLen - 1) / genSampFreq;
chart.ViewXY.XAxes[0].ScrollPosition = newestX;

chart.EndUpdate(); // Enable redraws and redraw
```

시리즈 데이터 업데이트는 데이터 저장 방법에 따라 다릅니다. 배열 시리즈의 경우 배열 내용을 업데이트한 후 ***InvalidateData()*** 방법을 호출해야 합니다. 그렇지 않으면 UI 에 변경 사항에 대한 알림이 표시되지 않습니다.

WPF 바인딩 가능 차트의 데이터 바인딩에 유용한 ObservableCollection 은 자동 알림으로 인해 모든 지점 또는 지점의 필드로 자체 업데이트되므로, 따라서 ***InvalidateData()***가 필요하지 않습니다. 그러나 ObservableCollections 는 배열 또는 목록에 비해 성능이 약간 저하되며, 이는 특히 많은 양의 데이터 포인트가 있는 경우에 두드러집니다.

17. LightningChart 알림, 오류 및 예외 처리

버전 8.4 부터 LightningChart 는 ***ChartMessage*** 이벤트를 통해 차트에서 사용자에게 메시지를 보냅니다. 메시지에는 차트 성능, 잘못된 사용, 경고 또는 오류에 대한 알림이 포함되며, 메시지를 수신할 ***chart.ChartMessage*** 이벤트에 대한 처리기를 정의하십시오. 이벤트에는 메시지 정보를 보유하는 ***ChartMessageInfo*** 구조체가 포함됩니다.

버전 8.4 이전의 차트는 ***ChartMessage*** 보다 적은 정보를 포함하는 ***ChartError*** 이벤트(현재는 사용되지 않음으로 표시됨)를 통해 메시지를 보냈지만, 사용자는 ***ChartMessages*** 대신 ***ChartError*** 이벤트를 수신하고 동일한 기본 정보를 얻을 수 있지만 대신 ***ChartMessage*** 를 사용하는 것이 좋습니다.

ChartMessageInfo 의 ***MessageSeverity*** 속성은 메시지의 심각도를 나타냅니다. 심각도에 따라 메시지를 필터링할 수 있습니다. 메시지의 가능한 심각도 수준은 다음과 같습니다.

- ***디버그*** – 일반적으로 사용자에게 흥미롭지 않으며 조치가 필요하지 않은 디버그 정보입니다.
- ***정보*** – 차트 성능에 영향을 주지 않아야 하는 잘못된 등록 정보 설정 사용과 같은 잘못된 차트 사용이 발생했습니다. 일반적으로 사용자 작업은 필요하지 않습니다.
- ***경고*** – 삭제된 개체를 사용하는 등 일부 잘못된 차트 사용이 발생하여 성능 손실과 같은 사소한 문제가 차트에 발생할 수 있습니다. 사용자 조치가 필요할 수 있습니다.
- ***복구 가능한 오류*** – 차트가 복구되어야 하는 오류가 발생했습니다. 사용자는 ***ChartMessage*** 이벤트를 수신해야 합니다. 그렇지 않으면 이 심각도의 메시지가 예외로 처리됩니다.
- ***복구 불가능한 오류*** – 차트를 복구할 수 없는 오류가 발생했습니다. 들어오는 예외를 나타낼 수 있으며, 사용자는 ***ChartMessage*** 이벤트를 수신해야 합니다. 그렇지 않으면이 심각도의 메시지가 예외로 처리됩니다.
- ***심각*** – 차트에서 항상 예외로 처리되는 심각한 오류가 발생했습니다.

MessageType 속성은 메시지의 기본 유형을 설명하는 반면 ***Details*** 속성에는 보다 구체적인 정보가 있으며, 가능한 모든 메시지 유형은 LightningChart 명칭 공간에 있는 ***MessageType*** 열거 형에서 찾을 수 있습니다.

원하지 않는 메시지는 ***chart.Options.ChartMessageMinimumLevel*** 속성 값을 변경하여 필터링할 수 있습니다. 이 속성은 설정된 최소 수준 이상의 메시지만 이벤트 시스템을 통해 전송되도록 허용하며, 기본적으로 ***MessageSeverity.Warning*** 으로 설정됩니다.

예외는 ***ChartMessage*** 이벤트와 유사한 예외에 대한 자세한 정보와 함께 ***ExceptionInfo*** 구조체를 포함하는 ***ChartException*** 개체로 바뀌게 되며, 경우에 따라 차트에서 렌더링 엔진 예외와 같은 다른 유형의 예외를 바꿀 수 있습니다. 사용자가 심각도 수준이 ***MessageSeverity.Warning*** 이상인 모든 메시지에 대해 차트에서 예외를 발생시키려면 ***chart.Options.ThrowChartExceptions*** 속성을 ***true*** 로 설정해야 합니다. (기본값은 ***false*** 임)

차트의 오류 및 목록에 없는 메시지의 가능한 예외에 대한 알림을 받으려면 항상 ***ChartMessage*** 이벤트를 구독하는 것이 좋습니다. 차트에 문제가 있고 이에 대한 지원이 필요한 경우 앱에 작동 중인 메시지/예외 처리기가 있는지 확인하고 ***ChartMessage*** 를 기록한 다음 지원 요청에 포함시키십시오.

18. LightningChart® 트레이더

거래자 라이브러리(Arction.Wpf.TradingCharts.dll / Arction.WinForms.TradingCharts.dll)는 거래 및 금융 앱을 쉽게 만들 수 있는 제어, 도구 및 방법으로 구성됩니다. 트레이더 라이브러리는 강력하고 빠른 LightningChart API 를 기반으로 구축되었으며, ***TradingChart*** 는 현재 주요 제어입니다. 복잡한 엔지니어링 앱에서 발생하는 API 오버 헤드 없이 거래 앱을 구축하기 위한 속성 및 방법의 간결한 인터페이스가 함께 제공되고, 현재 버전에는 WPF 및 WinForms 양식 차트가 포함되어 있으며 UWP 거래 차트는 나중에 사용할 수 있습니다.

18.1 TradingChart 생성

거래 차트를 사용하려면 해당 어셈블리를 프로젝트에 추가해야 합니다. 프로젝트의 참조에 Arction.Wpf.TradingCharts.dll(또는 WinForms 버전)을 추가한 후 거래 차트 객체를 생성할 수 있습니다.

```
using Arction.CustomControls.Trader.Wpf;

// Creating a TradingChart component
TradingChart _chart = new TradingChart();

// Adding chart into the parent container, in this case a grid.
(Content as Grid).Children.Add(_chart);
```

Visual Studio 의 도구 상자에서 끌어서 거래 차트 구성 요소를 만드는 것도 가능합니다.

18.2 TradingChart 배포

소프트웨어가 배포된 컴퓨터에서 거래 차트 앱을 실행하려면 배포 키를 코드에 적용해야 합니다. 이는 일반 LightningChart 와 유사하게 수행되며, (4.4 참조) 내부 차트 구성 요소에 대한 참조를 프로젝트에 추가해야 합니다. (Arction.Wpf.Chart.LightningChart 또는 Arction.WinForms.Chart.LightningChart)

18.3 내부 LightningChart 제어 사용

거래 차트는 일반 LightningChart 제어 위에 구축됩니다. ***GetInternalChart()*** -방법을 사용하면 거래 차트의 내부 LightningChart 제어와 모든 속성에 직접 액세스 할 수 있습니다. 따라서 두 차트의 기능을 조합하는 것이 가능합니다. 내부 차트에 액세스 하려면 각 어셈블리에 대한 참조가 프로젝트에 포함되어야 합니다. (예: Arction.Wpf.Charting.LightningChart)

18.4 UI 구성 요소

검색 표시 줄을 사용하면 기호 또는 회사 이름을 기반으로 공급자(AlphaVantage.co)의 거래 데이터를 검색할 수 있으며, ***ShowSearchBar*** 속성을 비활성화하여 숨길 수 있습니다.

도구 메뉴는 차트의 오른쪽 상단에 있습니다. 이 메뉴에는 사용 가능한 모든 그리기 도구가 포함되어 있으며 차트의 색상 테마를 변경할 수도 있습니다.

차트의 시간 범위는 차트의 오른쪽 하단 모서리에 있는 버튼을 통해 수정할 수 있습니다. ***ShowTimeRangeSelection*** -속성은 가시성을 제어합니다.

Volume 및 ***RelativeStrengthIndex*** 와 같은 일부 기술 지표는 차트 아래 별도의 세그먼트에 그려집니다. 이 경우 세그먼트 스플리터라는 수평선이 세그먼트 사이에 자동으로 그려지며, 이 선을 마우스로 드래그하면 세그먼트의 높이를 수정할 수 있습니다.

18.5 거래 데이터 추가

데이터 공급자

TradingChart 에는 기호 또는 보안 이름을 기반으로 증권을 검색할 수 있는 내장 데이터 공급자와 검색 표시줄이 있습니다. 텍스트 상자에 검색 문자열을 입력한 다음 "검색" 버튼 또는 Enter 키를 눌러 증권을 검색할 수 있으며, 검색 결과가 검색 표시 줄 아래 목록에 표시됩니다. 결과를 클릭하면 공급자로부터 해당 거래 데이터 세트를 로드하고 차트에 추가합니다.

검색 창을 사용하지 않고도 데이터를 가져올 수 있습니다. ***OpenSymbol()*** - 방법은 차트의 ***Symbol*** 속성에 따라 데이터를 얻기 위해 코드 숨김에서 사용할 수 있습니다. ***OpenSymbol()***은 얻은 데이터를 차트에 자동으로 추가합니다. 또한 통화와 같은 ***Symbol*** 을 기반으로 정보를 가져와서 차트 제목에 표시합니다.

TradingChart 는 달리 설정하지 않는 한 사전 정의된 데이터 제공자를 사용합니다. 그러나 이러한 데이터 공급자는 모든 사용자가 사용하고 허용되는 데이터 요청 수에 제한이 있는 경우가 많기 때문에 테스트 및 학습 목적으로만 사용하는 것이 추천되므로, 따라서 최종 제품에 자체 ApiKey 또는 데이터 공급자를 사용하는 것이 좋습니다. 미리 정의된 데이터 공급자를 사용하지 않아야 하는 경우 ***DataProvider*** -속성을 ***UserDefined*** 로 설정해야 합니다.

사용자가 미리 정의된 데이터 공급자 중 일부를 사용하고 싶지만 자체 API 키를 사용하려는 경우 ***SetRestApiKey()*** -방법을 사용해야 합니다. 거래 데이터를 요청할 때 TradingChart 가 주어진 키를 사용하도록 합니다.

파일에서

TradingChart 는 OHLC 데이터 배열을 반환하는 ***GetOhlcDataFromFile()*** 방법을 통해 .csv 파일에서 직접 거래 데이터를 읽을 수 있습니다. ***GetOhlcDataFromFile()*** -방법은 파일의 데이터가 DateTime, Open, Close, High, Low, Volume(선택 사항), OpenInterest(선택 사항)의 열 순서로 있다고 가정합니다. DateTime 필드에는 시, 분, 초가 포함되지 않아도 됩니다. ***SetData()*** -방법은 로드된 거래 데이터를 차트에 추가할 수 있습니다. 파일에서

데이터를 로드하면 데이터 세트의 첫 번째 및 마지막 DateTime 값을 기반으로 차트의 시간 범위가 자동으로 조정됩니다.

OhlcData 배열은 파일에서 데이터를 읽어 채울 필요가 없습니다. 데이터를 어디서 어떻게 얻을 수 있는지 결정한 다음 필요한 논리를 구현하는 것은 전적으로 사용자의 몫입니다. 따라서 데이터가 OHLC 형식으로 표시될 수 있는 한 ***SetData()***를 사용하여 외부 서버에서 데이터를 읽을 수도 있습니다.

18.6 기술 지표

TradingChart 에는 로드된 거래 데이터를 기반으로 자동 계산되는 몇 가지 내장 기술 지표가 있습니다. 차트에 기술 지표를 추가하려면 먼저 지표를 만들고 구성한 다음 모든 기술 지표를 저장하는 데 사용되는 ***지표*** 목록에 추가합니다.

```
RelativeStrengthIndex rsi = new RelativeStrengthIndex()
{
    PeriodCount = 14,
    HighColor = Colors.Lime,
};
_chart.Indicators.Add(rsi);
```

기술 지표를 제거하려면 TradingChart 의 ***지표*** 목록에서 제거하면 됩니다. 표시기는 목록에서 제거된 후 자동으로 삭제됩니다.

```
_chart.Indicators.Remove(indicator);
```

TradingChart 에는 다음과 같은 기본 제공 기술 지표가 있습니다.

- 상대 강도 지수 (RSI)
- 화폐 흐름 지수 (MFI)
- 확률 적 발진기 (SO)
- 이동 평균 수렴 발산 (MACD)
- 음량
- 미결제 약정 (OI)
- 단순 이동 평균 (SMA)
- 지수 이동 평균 (EMA)
- 가중 이동 평균 (WMA)
- 볼린저 밴드

18.7 그리기 도구

그리기 도구는 트레이딩 차트에서 자유롭게 그릴 수 있는 시각적 도구입니다. ***FreehandAnnotation*** 을 제외한 모든 그리기 도구는 두 개의 제어점을 기반으로 합니다. 첫 번째 점은 그리기 도구 그리기가 시작될 때 설정됩니다. 두 번째는 마우스 왼쪽 클릭으로 그리기가 끝날 때 설정됩니다. 이러한 제어점은 설정된 후 마우스로 이동할 수도 있으며, 그리기 도구는 그리기 및 제어점 이동 중에 실시간으로 그려지고 업데이트됩니다.

그리기 도구는 기본 제공 도구 메뉴를 통해 추가하거나 ***StartDrawing()*** 방법을 호출하여 그릴 수 있습니다. 메서드가 호출된 후 차트를 왼쪽 클릭하면 바로 그리기가 시작되고 첫 번째 제어점이 설정됩니다. 차트를 다시 왼쪽 클릭하면 그리기가 중지됩니다.

```
// Start drawing a yellow Trend line.
TrendLine trendLine = new TrendLine();
trendLine.LineColor = Colors.Yellow;
_chart.DrawingTools.Add(trendLine);
trendLine.StartDrawing();
```

모든 그리기 도구는 마우스 오버를 통해 제어점 중 하나를 강조 표시하고 Delete 키를 눌러 제거할 수 있습니다. 코드에서 그리기 도구를 제거하는 것은 지표와 유사하게 수행됩니다.

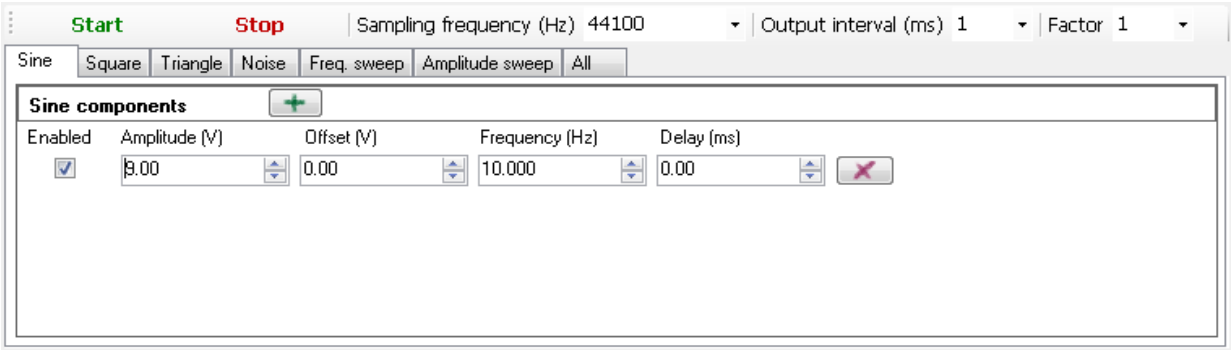
트레이딩 차트에는 다음과 같은 내장 그리기 도구가 있습니다.

- 추세선
- 피보나치 팬
- 피보나치 되돌림
- 피보나치 아크
- 선형 회귀
- 자유형 주석

19. SignalTools

SignalGenerator

SignalGenerator 구성 요소를 사용하여 실시간 신호를 생성할 수 있습니다. 신호는 서로 다른 파형의 합으로 생성되며, 여러 ***SignalGenerator*** 구성 요소를 마스터-슬레이브 관계로 연결하여 동기화된 다중 채널 출력을 생성할 수 있습니다. ***SignalGenerator*** 는 LightningChart 로 신호 모니터링 또는 데이터 수집 소프트웨어를 개발할 때 매우 유용합니다.



사용 가능한 파형은 다음 범주로 나뉩니다.

- 사인파
- 사각 파형
- 삼각형 파형
- 노이즈 파형-무작위로 생성된 신호
- 주파수 스위프-주파수 1 에서 주파수 2 로의 사인 스윕
- 진폭 스위프-진폭 1 에서 진폭 2 로의 사인 스윕

SignalReader 구성 요소

SignalReader 구성 요소를 사용하면 신호 소스 파일에서 데이터를 읽고 선택한 속도로 재생할 수 있습니다. SignalReader 출력 데이터 열 형식은 SignalGenerator 와 유사합니다. 파일 이름과 함께 제공된 OpenFile(...) 방법을 호출합니다. 파일 이름은 지원되는 형식(wav 및 sid)의 확장자를 가져야 합니다. 그런 다음 Start() 방법을 호출합니다.

AudioInput 구성 요소

AudioInput 구성 요소를 통해 사용자는 Windows 의 녹음 장치에서 System.Double 값으로 신호를 캡처할 수 있습니다. 이러한 값은 LightningChart 에서 렌더링 되고 AudioOutput 구성 요소로 전송되고 파일 등에 저장될 수 있습니다.

소스 코드에서 수동으로 또는 Visual Studio 의 도구 상자에서 양식, 창, 사용자 컨트롤 등으로 끌어서 놓는 방식으로 새 ***AudioInput*** 인스턴스를 만듭니다. ***AudioInput*** 인스턴스가 대신 명시 적 라이선스 키를 사용하도록 ***LicenseKey*** 속성을 설정하는 것이 좋으며, Windows 레지스트리에서 하나를 찾으려고 합니다. 체험판 / 라이선스를 사용하는 경우 ***LicenseKey*** 속성을 기본값으로 둘 수 있습니다.

AudioOutput 구성 요소

AudioOutput 구성 요소를 사용하면 System.Double 신호 데이터를 오디오 열로 변환한 다음 스피커를 통해 재생하거나 사운드 장치의 라인 출력 커넥터로 보낼 수 있습니다.

SpectrumCalculator 구성 요소

SpectrumCalculator 구성 요소는 시간 도메인과 주파수 도메인 간의 변환을 허용합니다. 다음과 같은 공개 방법을 사용할 수 있습니다.

- ***CalculateForward*** *(double [] samples, out double [] fftData)* - FFT 를 사용하여 시간 영역 신호 데이터를 주파수 영역으로 변환합니다. 출력 fftData 에는 음수 값도 포함됩니다. 입력 및 출력 데이터 배열은 길이가 같아야 합니다. 길이는 데이터의 분해능으로, 출력 값 사이에 동일한 주파수 간격으로 0Hz 에서 샘플링 주파수 / 2 로 확산됩니다.
- ***CalculateForward*** *(float [] samples, out float [] fftData)* - 이전 방법과 유사하지만 단일 정확도 부동 소수점 값에 사용됩니다.
- ***CalculateBackward*** *(double [] fftData, out double [] samples)* - 주파수 영역 데이터를 시간 영역으로 변환합니다. FFT 데이터에서 신호 샘플을 만들며, 샘플 수는 입력 fftData 길이와 같습니다.
- ***CalculateBackward*** *(float [] fftData, out float [] samples)* - 이전 방법과 유사하지만 단일 정확도 부동 소수점 값에 사용됩니다.
- ***PowerSpectrum*** *(double [] samples, out double [] fftData)* - 신호 데이터의 전력 스펙트럼을 계산합니다. ***CalculateForward*** 와 동일하지만 절대 출력 값이 있습니다.
- ***PowerSpectrum*** *(float [] samples, out float [] fftData)* - 이전 방법과 유사하지만 단일 정확도 부동 소수점 값에 사용됩니다.
- ***PowerSpectrumOverlapped*** *(double [] samples, int fftWindowLength, double overlapPercent, out double [] fftData, out int processingSampleCount)* - 소스 신호 샘플 데이터 내부의 계산 창을 겹침 비율로

이동하여 전력 스펙트럼을 계산합니다. 신호 데이터는 주어진 FFT 창 길이보다 길어야 하며, 출력 FFT 데이터는 fftWindowLength 의 길이이고 소스 데이터의 길이와 반드시 동일하지는 않습니다. 출력 데이터에는 절대 값이 있습니다.

20. 헤드리스 모드

헤드리스 모드는 GUI(그래픽 사용자 인터페이스)에 액세스 하지 않고 장치에서 작업하는 소프트웨어 기능입니다. "헤드리스"라는 용어는 소프트웨어가 주변 장치(예: 디스플레이, 키보드, 마우스)의 존재 또는 이에 대한 액세스를 필요로 하지 않는 경우에도 사용됩니다. 주변 장치가 없다고해서 초기화 또는 실행 프로세스가 실패하지는 않으나, 이 경우 소프트웨어는 입력을 수신하고 다른 통신 인터페이스(예: 네트워크 또는 직렬 포트를 통해)를 통해 출력을 제공할 수 있습니다.

LightningChart SDK 는 WPF 용 서비스, 콘솔 애플리케이션 및 클라이언트 앱을 포함하는 예시 Visual Studio 솔루션(***DemoService.sln***)과 함께 제공됩니다. DemoService.sln 은 ***C:\ProgramData\Arction\LightningChart .NET SDK v.10\DemoService*** 폴더에서 찾을 수 있습니다.

기본적으로 Windows 서비스는 시스템 사용자 계정의 보안 콘텍스트에서 실행됩니다. ***체험판 및 개발 라이선스의 설치가 불가능하며***, 이러한 이유로 서비스 앱은 유효한 ***배포 키***를 포함하거나 활성 라이선스(평가판/개발)가 있는 일반 사용자의 자격 증명으로 실행되어야 합니다.

20.1.1 헤드리스 렌더링

헤드리스 구성을 사용하면 헤드리스 / 서버 환경에서 LightningChart 를 실행할 수 있습니다. 예상 시나리오에는 UI(사용자 인터페이스)가 없는 소프트웨어 앱의 배경 렌더링 및 차트 콘텐츠에서 비트 맵 이미지 생성이 포함됩니다. 그런 다음 이미지를 추가 렌더링을 위해 헤드 풀 시스템으로 전달할 수 있습니다.

헤드리스모드는 ***chart.ChartRenderOptions***(WPF 의 경우) 또는 ***chart.RenderOptions***(WinForms 의 경우)를 통해 ***HeadlessMode*** 플래그를 활성화하여 사용할 수 있습니다. LightningChart 는 Windows 서비스 유형 앱에서 자동으로 사용을 감지하므로 모드를 지정할 필요가 없습니다.

UI 및 시각적 부모가 누락된 초기화 된 LightningChart 인스턴스는 렌더링 요청 또는 렌더링 엔진 초기화 신호를 수신하지 않습니다. 따라서 사용자는 다음 작업 및 구성을 차트에 적용해야합니다.

- ***chart.Width*** 및 ***chart.Height*** 속성을 사용하여 크기를 정의합니다.
- ***chart.InitializeRenderingDevice(true)***를 호출하여 렌더링 엔진을 초기화합니다.(WPF 에만 해당)
- 이미지 내보내기 로직을 구현하려면 ***chart.AfterRendering*** 이벤트를 구독하십시오.

차트는 여전히 속성 변경에 반응합니다. 새 프레임의 렌더링은 필요한 경우 연속적인 ***BeginUpdate()*** 및 ***EndUpdate()*** 호출로 질문할 수 있습니다.

렌더링 된 프레임은 ***OutputStream*** 속성, ***SaveToStream*** 방법, ***CopyToClipboard*** 방법, ***CaptureToByArray*** 방법 또는 ***SaveToFile*** 방법과 같은 다양한 방법으로 내보낼 수 있습니다. 일반적으로 비트 맵 열이 선호되며, 또한 ViewXY 차트는 ***SaveToStream*** 및 ***SaveToFile*** 방법의 헤드리스 모드에서 EMF, WMF, SVG 를 지원합니다.

21. 크레딧

Intel Math Kernel Library\

LightningChart® .NET SDK 는 일부 부분(예: 고속 푸리에 변환 방법)에서 Intel Math Kernel Library 를 사용합니다. Arction 어셈블리에는 이 라이브러리에서 빌드된 일부 네이티브 DLL 파일이 포함되어 있습니다. Arction Ltd.는 Intel Math Kernel Library 를 사용할 수 있는 라이선스를 받았습니다.

오픈 소스 프로젝트

다음 오픈 소스 프로젝트 및 자료 제공 업체에 감사를 표합니다.

.NET 용 DirectX 라이브러리

LightningChart 는 Arction 에서 만든 확장과 함께 SharpDX 파생 DirectX .NET DLL 을 사용합니다, http://www.sharpdx.org/

맵 소스

LightningChart® .NET 맵은 다음과 같이 맵 공급자에서 가져 왔습니다.

세계, 북미, 유럽:	Natural Earth, http://www.naturalearthdata.com/
호주:	Australian Bureau of Statistics, http://www.abs.gov.au/
미국 도로:	National Atlas of the United States, http://www.nationalatlas.gov

확장 가능한 벡터 그래픽 출력

LightningChart SVG 내보내기는 RiskCare Ltd 의 SvgNet 프로젝트 코드를 부분적으로 사용합니다.

다항 회귀

다항 회귀 계산 코드는 부분적으로 Math.Net 라이브러리를 기반으로 합니다, http://www.mathdotnet.com/

수정된 소스 코드 부분은 Arction Support 에서 요청 시 무료로 제공됩니다. (support@arction.com)

오픈 소스 프로젝트의 저작권 고지 사항은 LightningChart SDK 설치 폴더의 **LightningChart .NET Readme.txt** 를 참조하십시오.