

チュートリアル_507 (1.0):

VirtualLab™ における構築ブロック、素子、 ディテクターのプログラマブル作成法

著者: Christian Hellmann, Michael Kuhn (LightTrans)

関連チュートリアル: [Tutorial 501](#)

必須ツールボックス: VirtualLab™ 5.5以降のStarter Toolbox



目次

- はじめに
 - プログラミングのゴール
 - プログラミング言語
 - 参照情報
- VirtualLabにおけるプログラマブルなオブジェクト
 - モジュール
 - Snippets
- プログラマブル素子のSnippets
 - コンセプト
 - プログラマブル素子のInputとoutput
 - 例

コンテンツ

- プログラミング環境
 - VirtualLabエディターの活用
 - Visual Studioプロジェクト
 - プロジェクトの設定方法
 - C++ DLLsの導入法
 - MATLAB® とVirtualLab
 - Snippets内のDLL
- プログラマブル・ディテクター
 - コンセプト
 - 出力タイプのまとめ

はじめに

VirtualLabでのプログラミングのゴール

- VirtualLabは市場に最初に紹介されたフィールドトレーシング・ソフトです
- 現代の光学シミュレーションに対する要求は、年々複雑になってきております
- LightTransでは、様々なシミュレーション技術を提供さしあげておりますがVirtualLab内でユーザー独自のアルゴリズムを採用する事が可能です。
- VirtualLabのプログラミング・インターフェースにより、構築ブロック、素子、ディテクターのカスタマイズを可能にします
- 上記により、光学系のシミュレーションに完全な自由度を与える事が可能です

プログラミング言語

- 本書はプログラミングの講習を行うものではありません。本書では操作しながら学ぶ、初期動作を説明するものです。プログラミングに関しては、ユーザー各位にてご対応お願いします。
- プログラミング言語: C#
 - C++のオブジェクト操作型言語です
- .NET Framework
 - Release 2002
 - ライブラリー化された機能、クラス、GUI

プログラミング言語

- C# – 参照資料
 - **Beginning C# 3.0: An Introduction to Object Oriented Programming**
(Jack Purdum, Wiley Publishing Inc., 2008)
 - **C# 3.0: A Beginner's Guide**
(Herbert Schildt, McGraw Hill, 2009)
 - **Professional C# 2008**
(Christian Nagel, Bill Evjen, Jay Glynn, Morgan Skinner, KarliWatson, Wiley Publishing Inc., 2008)
 - **C# 3.0 The Complete Reference**
(Herbert Schildt, McGraw Hill, 2009)

プログラミング言語

- .NET 機能
 - .NET データタイプ (double, int, etc)
 - e.g. Math.Sin(), GUI, etc.
 - [http://msdn.microsoft.com/en-us/library/67ef8sbd\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/67ef8sbd(VS.80).aspx)
- VirtualLab™ プログラミング
 - VirtualLabライブラリー(VirtualLabAPI)のクラスと機能にアクセス
 - VirtualLabのヘルプメニューにプログラミングのリファレンスを
用意しております

Some C# Syntax

- Comments:

```
/* comment (multiple lines) */
```

```
// comment (1 line)
```

- Variableタイプ

```
int i ; double x; float y;
```

- Math-function:

```
y = Math.Sin(x) ; y = Math.Abs (x) ;
```

- use “;” at end of a line

- to return a value use

```
return x;
```

VirtualLabにおけるプログラミングのレベル

VirtualLabAPI.dllにてプログラミング

- VirtualLabAPIのプログラミングは、LightTrans社にて行われ、よって権利もLightTransに帰属します
- 現在、約3000ファイルと、約800,000ラインのコードがあります
- 自由度は非常に高いものの、設計ルールは採用されます
- コードは、提出しかねますが、クラスと手法は、Snippetsとモジュールに使用可能です
- 多くの第三機関のDLLをVirtualLabAPIより活用する事ができます。これらのDLLに対する、開発ライセンスは、LightTrans社に帰属します

VirtualLabモジュール

- VirtualLabモジュールは、.NET 環境においてC# 及びVisual Basicにて構築する事が可能です
- モジュール内では、通常100～1000 ラインのコードとなります
- .NET と VirtualLabAPI.dll および第三機関からのクラス及び機能を用いる事が可能です
- 上記により、カスタムが必要な課題に対し、高い自由度を持って対応可能となります

VirtualLab Snippets

- VirtualLab では、Snippetsにより様々なシミュレーションの異なる部分に必要な機能を取り込む事が可能です：
 - ー ソースコード・ブロックにより、ある機能を有するボディーを示します
 - ー 通常10～100ラインのコードが必要となります
 - ー Snippet内ではクラスまたは機能を定義する事ができません
 - ー この機能は2013年に追加される予定です

VirtualLab Snippets

- 外部データをSnippetに追加することが可能です。これにより、外部のDLLから機能を追加する事が可能になります。採用可能なDLLは：
 - .NET DLL
 - C++ DLL
 - MATLAB® codeにより作成されたDLL。詳細は、Tutorial 501 “Using MATLAB® Functions from VirtualLab Snippets and Modules”をご参照下さい

外部DLLの活用と、ユーザーの権利

- 外部DLLを活用するには、以下のWindowsユーザー権利が必要となります：
 - － 拡張時には“DLLCache”フォルダーが形成され操作できます。
このフォルダーは VirtualLabがインストールされたディレクトリーに形成されます
 - － VirtualLabがシステムドライブにインストールされた場合（通常 C:）
アドミニストレータである必要があります。これは VirtualLabを
“Run as Administrator”として開く必要があります。
 - － VirtualLabを他のドライブにインストールした場合（C:以外）
アドミニストレータでなくても可能です

制限事項

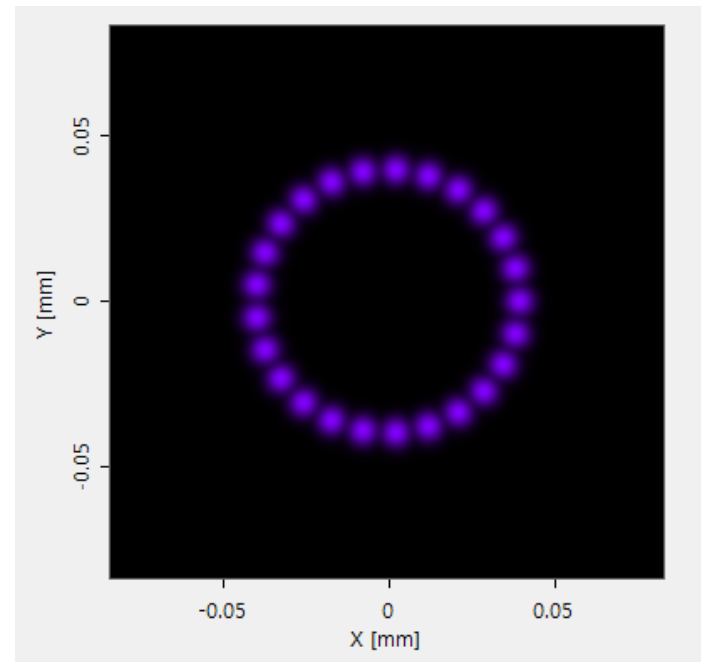
- VirtualLab Advanced
 - ー プログラミング機能に制限はありません
- VirtualLab (Standard)
 - ー プログラマブル素子とディテクターはRead Only対応。 ソースコードを編集する事ができません。
- VirtualLabTrial
 - ー モジュールの操作はできません
 - ー Snippetsによるプログラマブル・構築ブロックはRead Only対応

VirtualLabのSnippetベース素子

プログラマブル (Mode Planar) 光源

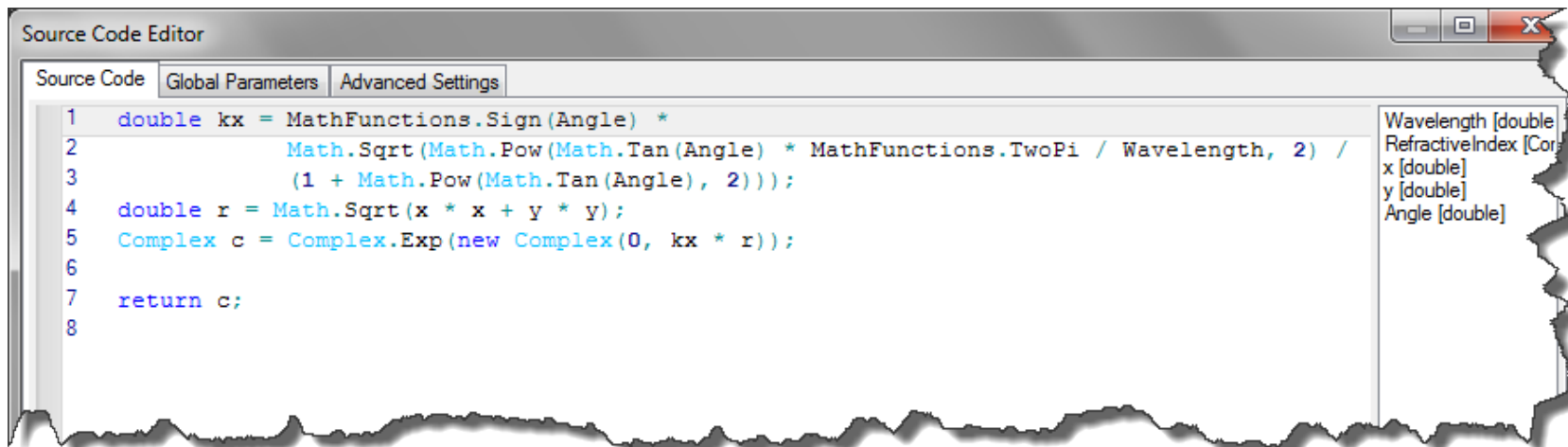
- 光源のプログラミング:
 - ラテラル配光分布
 - 波長分布 (spectrum generatorによる)
 - モード配置
 - モードのウェイト

光源カタログ:
サークル配置の
モード
($\lambda = 405 \text{ nm}$)



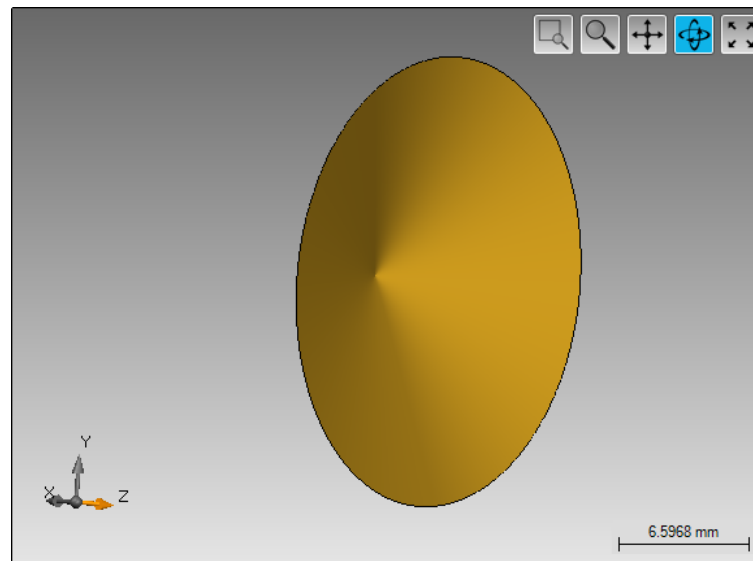
プログラマブル・トランスミッション

- Snippetを用いて、理想の現象を発生するトランスミッション(光学機能等価面)を定義する事が可能です
- ユーザーはトランスミッションを位置情報(x,y)として定義します



プログラマブル・インターフェース

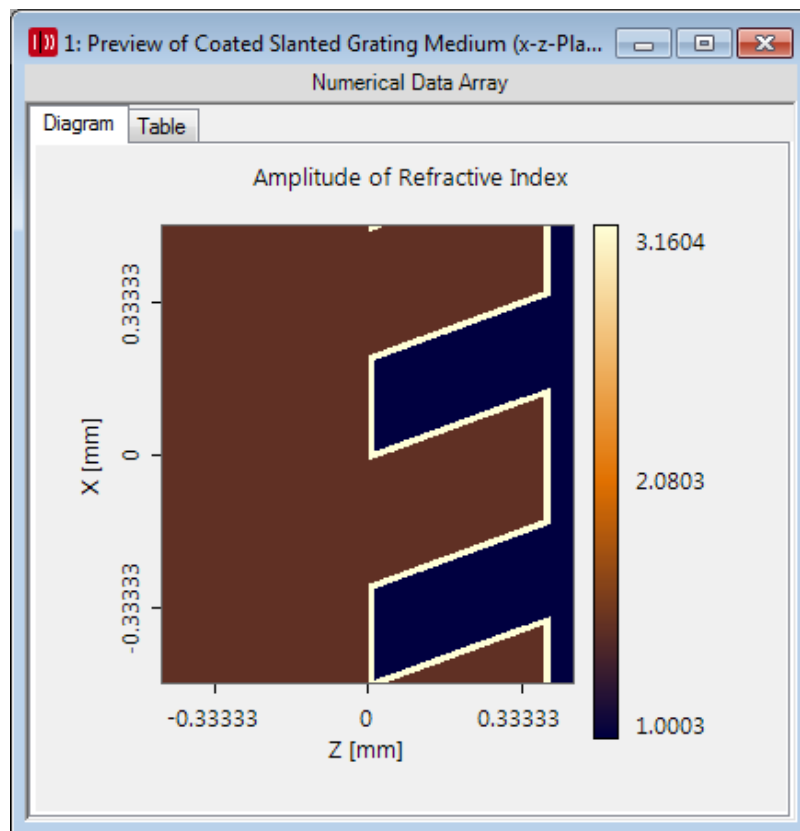
- ユーザーにより、光学インターフェース $h(x, y)$ の高さ情報を定義可能です
- さらに、部分的な派生体として $\frac{\partial h}{\partial x}(x, y)$ と $\frac{\partial h}{\partial y}(x, y)$ の定義が可能です



インターフェースカタログ：
アキシコン・インターフェース
20°

プログラマブル媒体

- プログラマブル媒体内にて、ユーザーは3D屈折率変調を定義する事が可能です



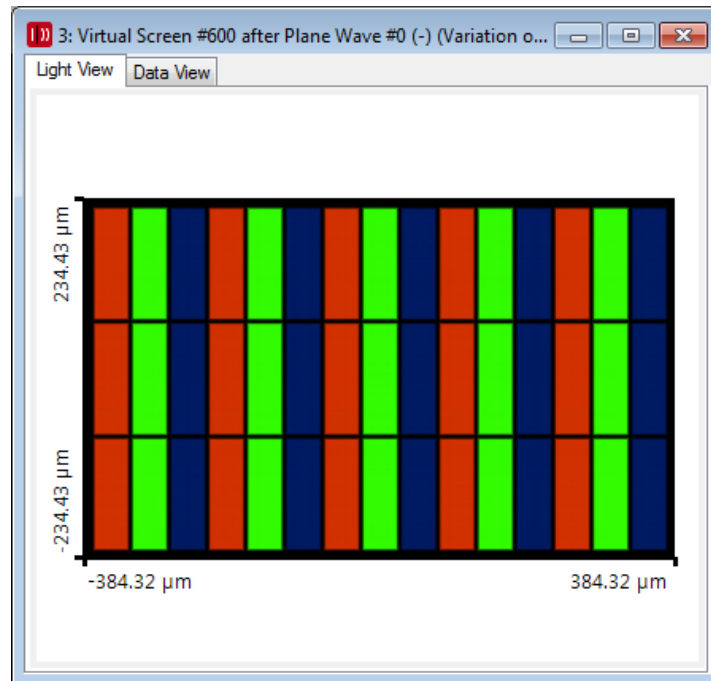
媒体カタログ：
コーティングされた斜形
グレーティング媒体

Parameter Run: プログラマブル・モード

- VirtualLabのParameter Runにより、ユーザー定義モードのパラメーターの可変が可能です(スキャニング、ランダム及びスタンダードモード)
- パラメーターを可変し評価する場合、ユーザー定義されたパラメーターをインポートし、Parameter Runにて評価する事が可能です

Parameter Run: プログラマブル・モード

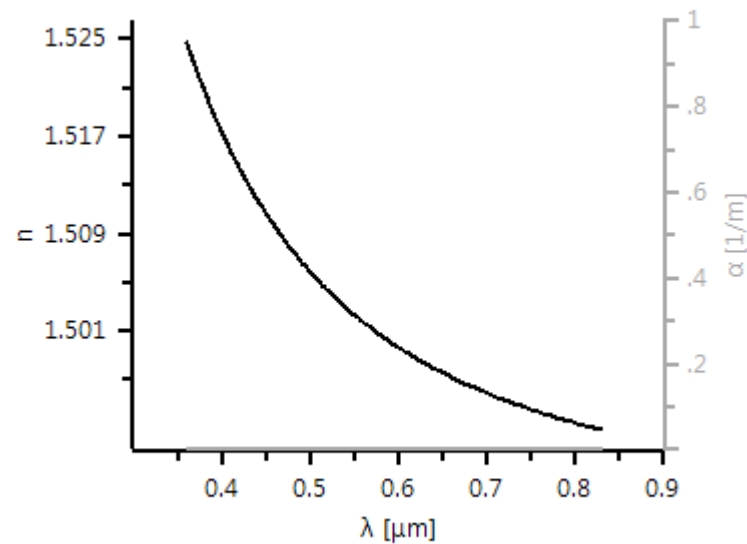
- 例:
 - ー アプリケーション_178.01: LCD光源のシミュレーション
 - ー 波長とウェイトのパラメーターは、相互に依存します。位置を追加条件として、可変する事が可能です。



アプリケーション_178.01の結果:
RGBピクセルをLCDのピクセル
として定義。このフィールドを
さらにシミュレーションに活用
します

プログラマブル素材

- Snippetにて、ディスページョンを定義する事が可能です



素材カタログ: アッベ数 V_d 素材

プログラマブル素子

- プログラマブル素子は、素子全体と伝播手法の定義を可能とします
- Snippetの開発者は、入射フィールド、構造パラメーター等から得られる出力として素子を定義します
- プログラマブル素子はエクステンションを含まないため、1面のみの効果をしめします
- 詳細は、VirtualLabのマニュアルと、プログラミング参照資料 (VirtualLabのヘルプ)

プログラマブル・ディテクター

- プログラマブル・ディテクターは、インプットされたフィールドに応じてディテクト結果を発生するものです
- ユーザーは、入射フィールドから下記の情報を抽出します：
 - 物理値（数値 + 物理プロパティ）
 - ハーモニックフィールド（複素振幅オブジェクト）
 - データアレー
- 物理値の出力は、後のパラメトリック最適化に活用可能です
- 詳細は、VirtualLabのマニュアル及びプログラミング参照資料 (VirtualLabのヘルプ)をご参照下さい

プログラマブル素子

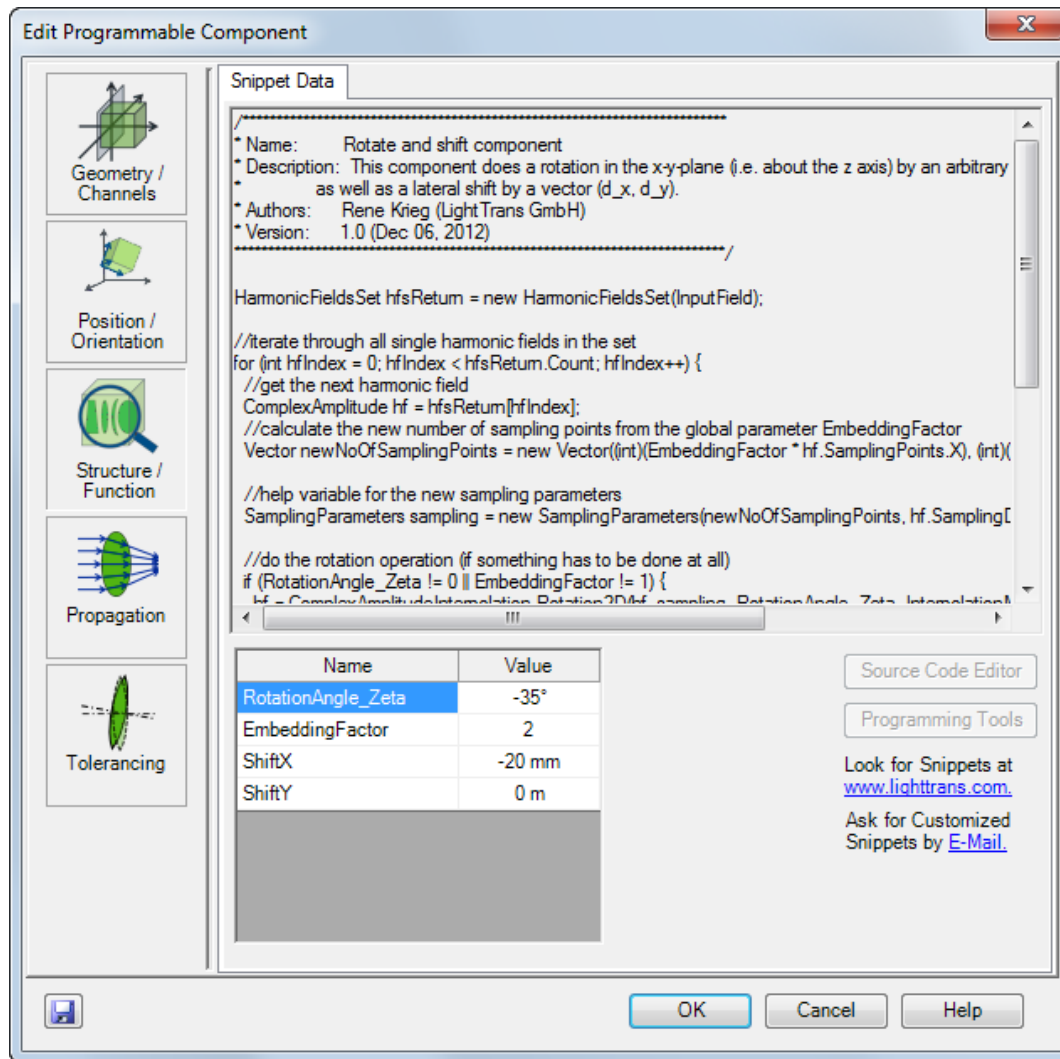
プログラマブル素子のコンセプト

- プログラマブル素子は、入射フィールドをSnippetにより定義された、任意の状態に変換するものです
- Light Path Diagramにおけるプログラマブル素子は、一面のみの位置情報を示します。従って、プログラマブル素子の出射フィールドは、光路上で入射フィールドと同じ位置となります。反射タイプの場合、単一インターフェースによる反射として、オリエンテーションが変化します。
- このコンセプトは2013年後半に拡張し、チルトと素子の拡張定義が可能となる予定です

プログラマブル素子のコンセプト

- 素子内部の仕様は、光学インターフェース、光学素材、媒体、反射データアレーなどのリストからプログラマブル素子が定義されます
- これらの光学系ブロックはSnippet内に採用可能で、入射フィールドを受けた出力フィールド情報を計算します
- プログラマブル素子をParameter Run内で用いた場合、Snippetのパラメーターと光学インターフェース、素材、及び媒体を可変する事が可能となります

プログラマブル素子の編集ダイアログ



ソースコード・エディターのダイアログ

The screenshot shows the 'Source Code Editor' dialog box with the 'Global Parameters' tab selected. The dialog has three tabs: 'Source Code', 'Global Parameters', and 'Advanced Settings'. The 'Global Parameters' tab contains a table with the following data:

Variable Name	Value	Quantity	Minimum	Maximum
RotationAngle_Zeta	-35°	Angle (Deg)	0°	360°
EmbeddingFactor	2	No Unit	1	1000
ShiftX	-20 mm	Length	-1 km	1 km
ShiftY	0 m	Length	-1 km	1 km

Below the table are buttons for 'Add', 'Edit', and 'Delete'. The dialog also includes sections for 'Global Materials', 'Global Media', and 'Global Interfaces', each with a table for 'Index', 'Variable Name', and 'Material/Medium/Interface'. At the bottom, there is a 'Reference Field (2D Data Array)' section with 'Set', 'Show', and 'Remove' buttons, and a 'Check Consistency' button. The bottom right corner has 'Ok', 'Cancel', and 'Help' buttons.

- グローバル・パラメーター:

- 数値
- 素材
- 媒体
- インターフェース
- データアレー

Input/Output: ハーモニックフィールドセット

- プログラマブル素子内で入射/出射に用いられる光束フィールドはハーモニックフィールド・セットです
- ハーモニックフィールドセットには、下記の屈折率にアクセス可能な複数のハーモニックフィールドが含まれます
 - ハーモニックフィールドセット[int i] は屈折率 i のハーモニックフィールドをセットする事が可能です
- ハーモニックフィールドセットの作成:
 - Copy-Constructor: `new HarmonicFieldsSet(hfsOld).`
 - Empty Constructor: `new HarmonicFieldsSet().`

Input/Output: ハーモニックフィールドセット

- さらに、ハーモニックフィールドセット(HFS) はオブジェクトを可変する手法をサポートしております：
 - Add(複素振幅 ca) (HFSに複素振幅を追加)
 - Remove(inti) (屈折率 i の複素振幅の削除)
 - Insert(複素振幅 ca , inti) (屈折率 i に複素振幅を挿入)

Input/Output: ハーモニックフィールドセット

- ハーモニックフィールドセット全メンバーの一般的なループ:

```
HarmonicFieldsSet hfsReturn = new HarmonicFieldsSet();

for(int runMember = 0; runMember < InputField.Count; runMember++){
    ComplexAmplitude caCurrent = (ComplexAmplitude)InputField[runMember].Clone();

    /*****
     * Do something the ComplexAmplitude *
     *****/

    hfsReturn.Add(caCurrent);
}

return hfsReturn;
```

Input/Output: 複素振幅クラス

- VirtualLabAPI.dll 内の複素振幅クラスはハーモニックフィールドを表現します
- 複素振幅オブジェクトに含まれる重要なプロパティは:
 - `ca.Wavelength` (caの真空波長)
 - `ca.IsLocallyPolarized` (局所的偏光のフラッグ)
 - `ca.Field` (全面的偏光のフィールド値のマトリクス)
 - `ca.FieldX`, `ca.FieldY` (全面的偏光のフィールド値のX,Y値)

Input/Output: 複素振幅クラス

- `ca.JonesVector` (`ca`のJonesベクトル。`ca`が全面的偏光の場合のみ)
- `ca.EmbeddingMedium` (`ca`が定義された媒体)
- `ca.LFO_CoordinateSystem` (`ca`の位置と配置)
- さらに、複素振幅クラスは、屈折率をサポートします。 `ca[x,y,false]` をピクセル配置(x, y)におけるデータマトリクス E_x に簡単にアクセス可能です。
- 複素振幅クラスは、ハーモニックフィールドの可変は評価方法多数をサポートしております

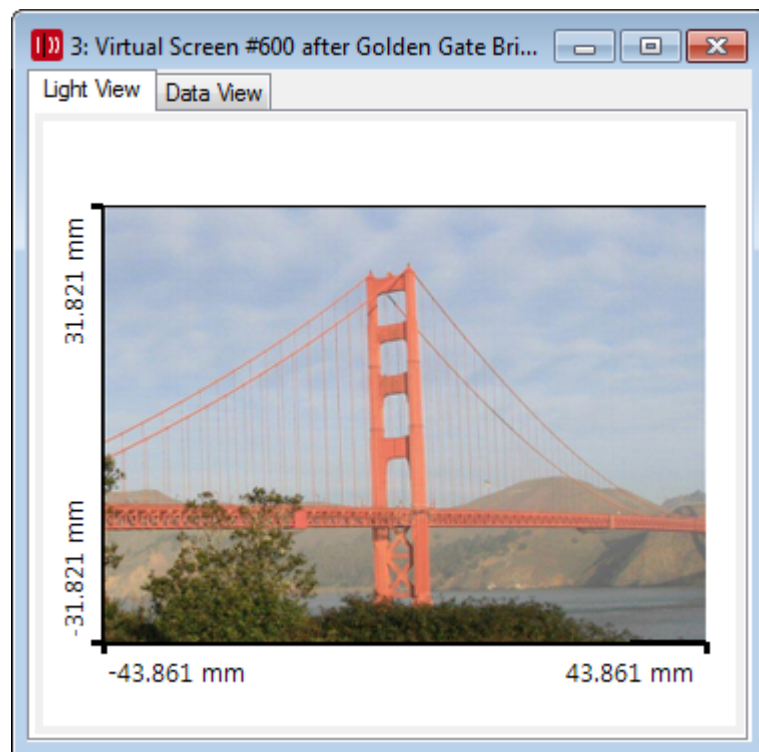
プログラマブル素子：回転とシフト

```
8
9  HarmonicFieldsSet hfsReturn = new HarmonicFieldsSet(InputField);
10
11  //iterate through all single harmonic fields in the set
12  for (int hfIndex = 0; hfIndex < hfsReturn.Count; hfIndex++) {
13      //get the next harmonic field
14      ComplexAmplitude hf = hfsReturn[hfIndex];
15      //calculate the new number of sampling points from the global parameter EmbeddingFactor
16      Vector newNoOfSamplingPoints = new Vector((int)(EmbeddingFactor * hf.SamplingPoints.X), (int)(EmbeddingFactor * hf.SamplingPoints.Y));
17
18      //help variable for the new sampling parameters
19      SamplingParameters sampling = new SamplingParameters(newNoOfSamplingPoints, hf.SamplingDistance);
20
21      //do the rotation operation (if something has to be done at all)
22      if (RotationAngle_Zeta != 0 || EmbeddingFactor != 1) {
23          hf = ComplexAmplitudeInterpolation.Rotation2D(hf, sampling, RotationAngle_Zeta, InterpolationMethod.Cubic6P);
24      }
25
26      if (ShiftX != 0 || ShiftY != 0) {
27          //determine new zero point that considers the shift
28          VectorD newZero = new VectorD(-ShiftX, -ShiftY);
29
30          //do the shift operation via Interpolation method
31          hf = ComplexAmplitudeInterpolation.Interpolation(hf, sampling, newZero, InterpolationMethod.Cubic6P);
32      }
33
34      //set the rotated and shifted harmonic field back to the set
35      hfsReturn[hfIndex] = hf;
36  }
37
38
39  return hfsReturn;
```

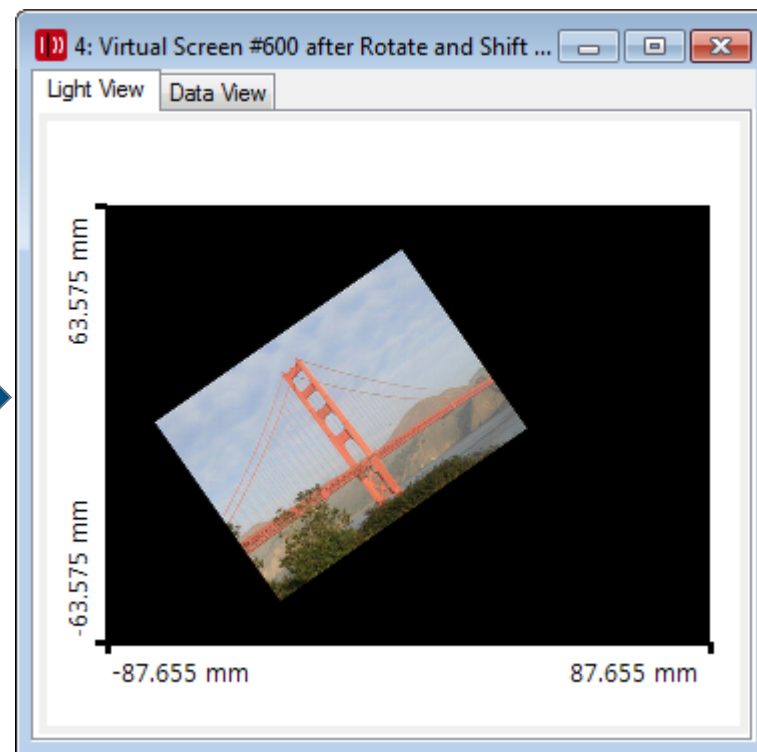
素子カタログ:
素子の回転とシフト

回転とシフト: 結果

Input



Output




Snippetsとモジュールの開発

VirtualLabソースコード・エディター vs Visual Studio

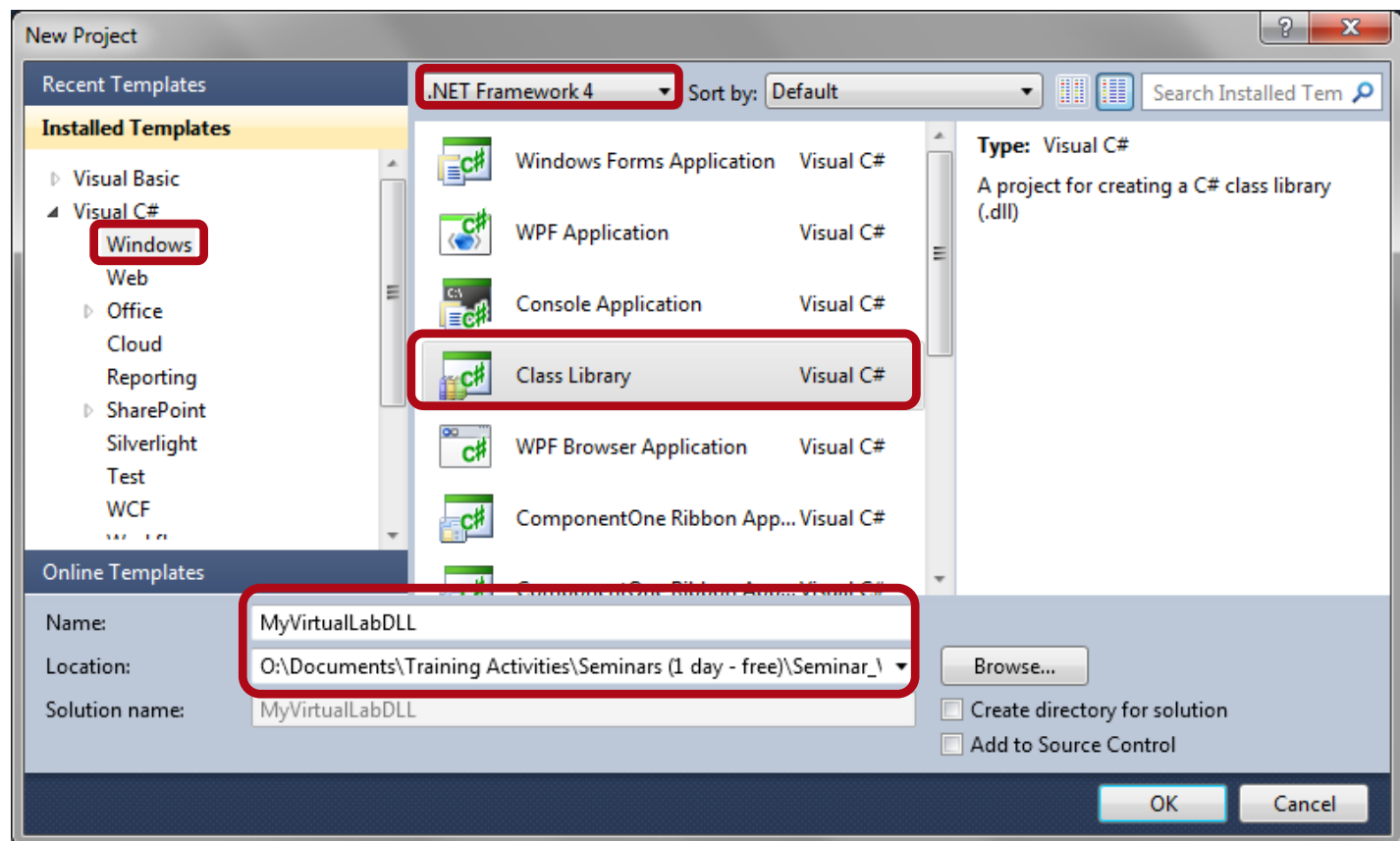
- Snippetの開発は、VirtualLabのソースコードエディターを用いて行うことができます
- ソースコードエディター：
 - Syntax highlighting.
 - (Un)Collapsing (regions).
 - Comment/Uncomment 連続コードライン
- Snippetの開発は、Microsoft Visual Studioでも可能で、syntax completionも可能となります

Visual Studioのダウンロードとインストール

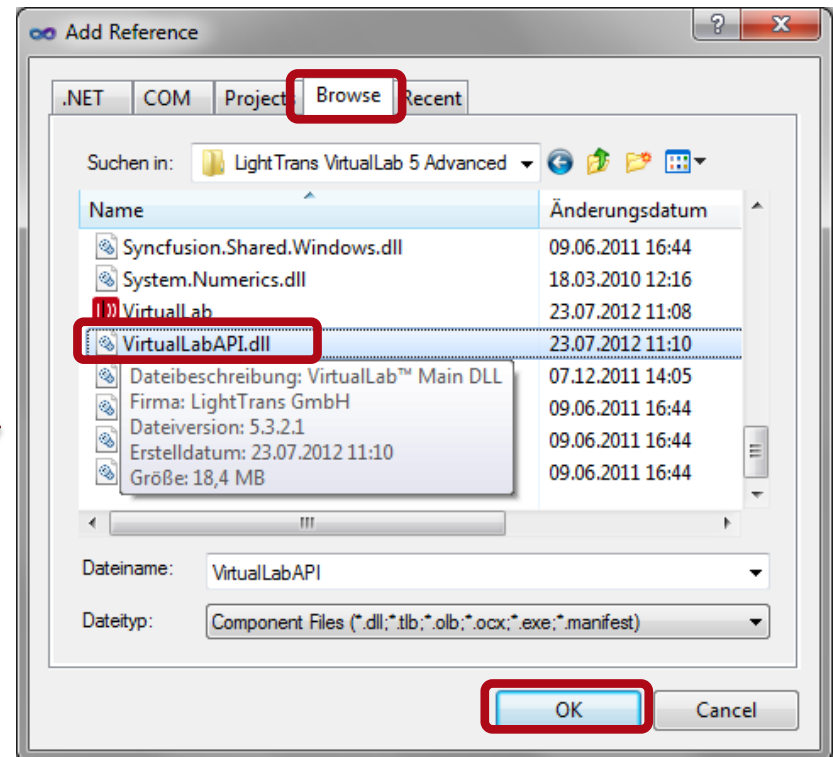
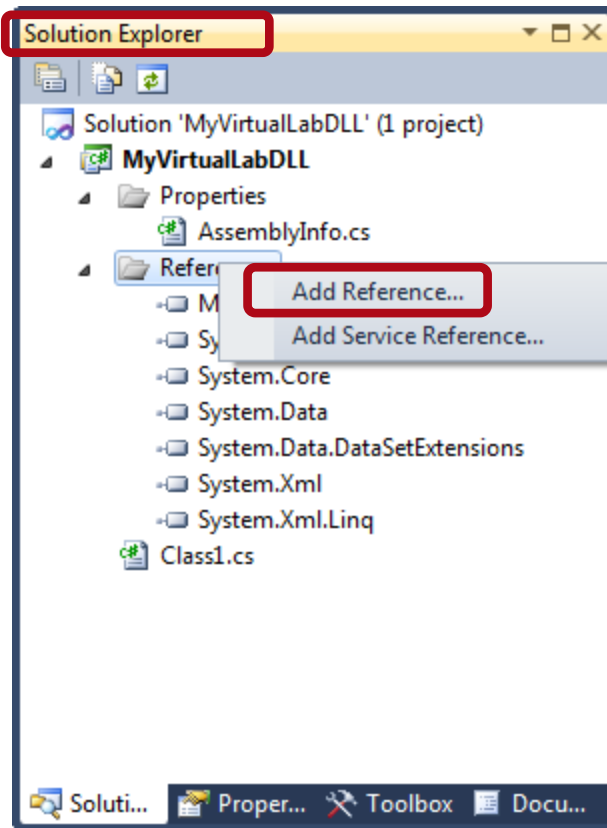
- Visual Studio Express バージョンは無償でダウンロード可能です：
(<http://www.microsoft.com/germany/express/download/default.aspx>)
 - Professional とExpressバージョンの違いは：
 - SQL Serverのサポート無し
 - Team Foundation Managerの必要無し
 - F#言語によるアプリケーションの開発
-  Visual Studio 2010 Express はVirtualLabのSnippet開発に有効です

Visual Studio Projectの設定

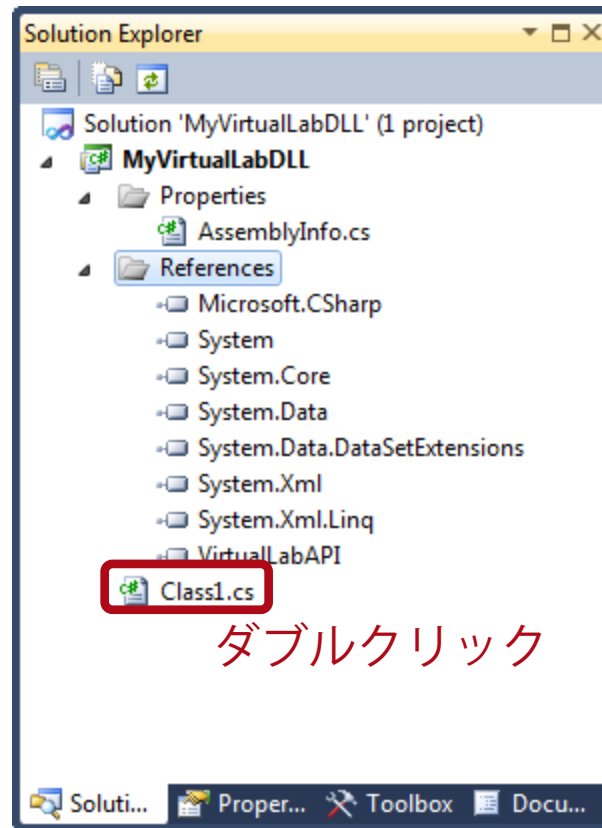
- Visual Studio 2010を開き、New Projectを選択します



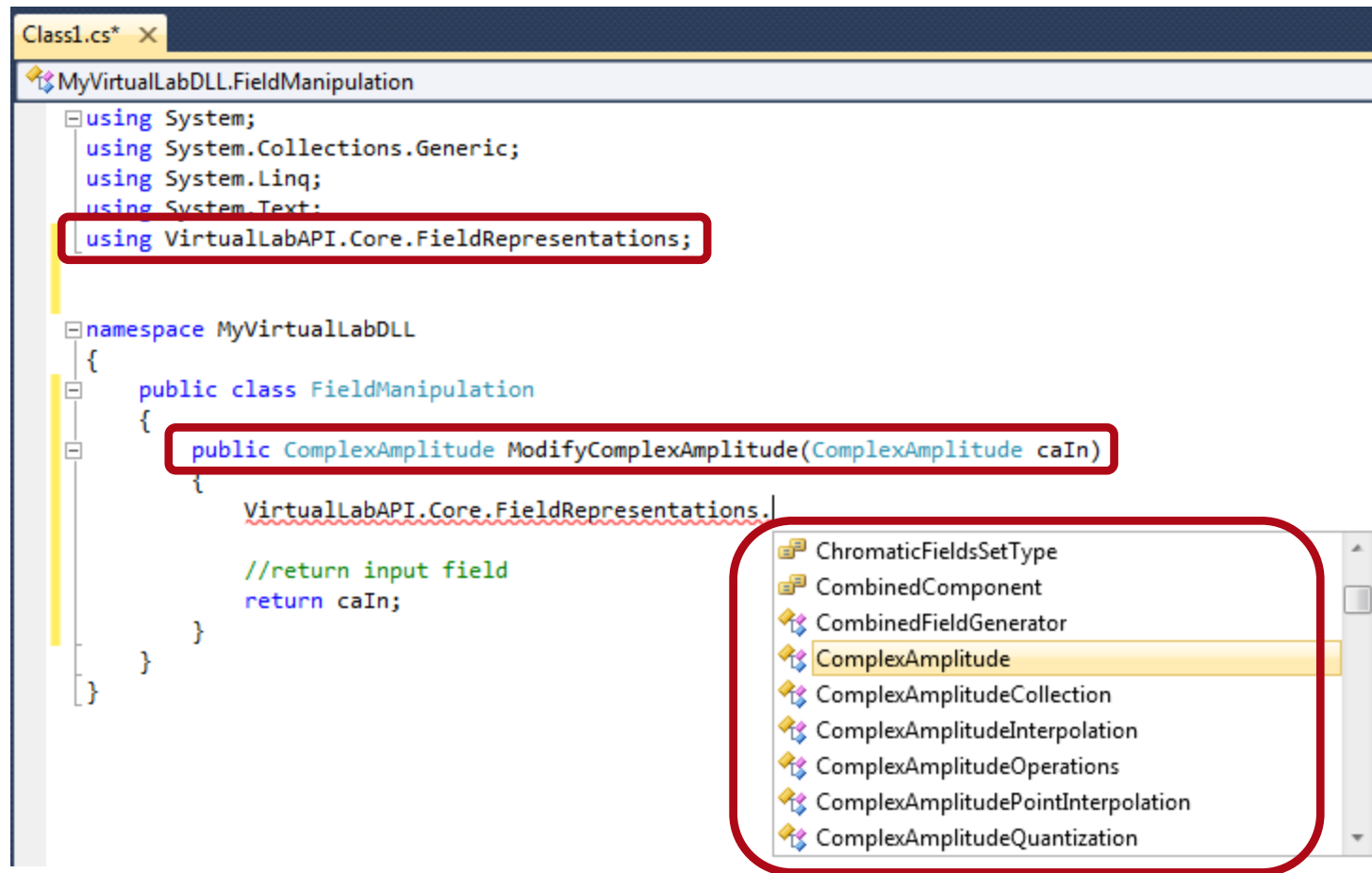
Visual Studio Projectの設定



Visual Studio Projectの設定



例: Syntax Completion

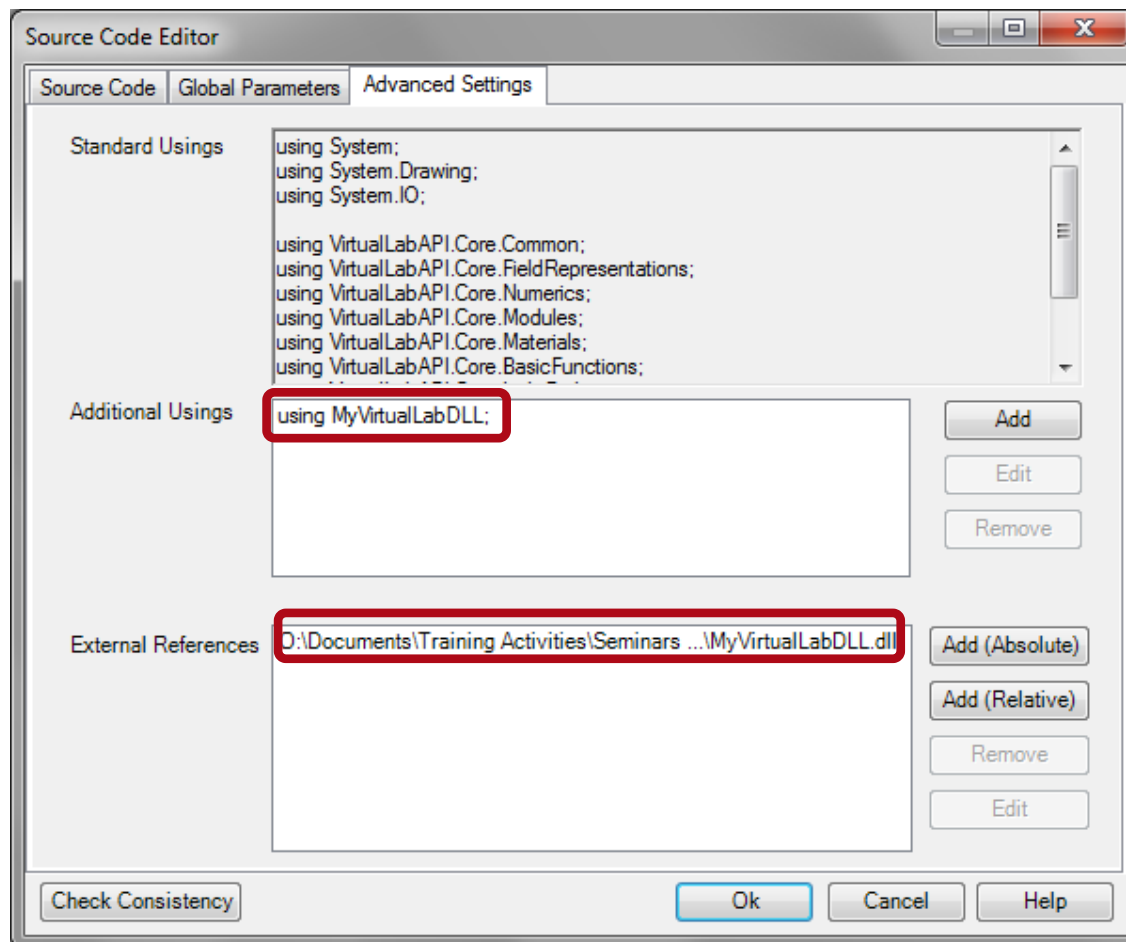


Snippets と DLL

- syntax completionを活用する事が出来る事から、Visual Studio projectを用いる事により、Snippetとモジュール開発をシンプルに行う事が可能です
- ここまで、全コードはSnippet内で書かれております
 - ー Visual Studioにて書かれたソースコードは、Snippetにコピーする事が可能です
- Snippetから外部DLLを用いて、機能呼び込む事が可能です
 - ー Visual StudioにてDLLを書く事が可能です。 機能とクラス定義をより複雑なアプリケーションに活用する事が可能です。
 - ー DLLを外部リファレンスに追加する事が可能です

Advanced 設定

- SnippetのAdvanced 設定:



例: DLLの活用

- DLL 内のコード:

```
namespace MyVirtualLabDLL
{
    public class FieldManipulation
    {
        /// <summary> ...
        public ComplexAmplitude MultitplyComplexAmplitudeWithFactor(ComplexAmplitude caIn,
                                                                    double factor)
        {
            ComplexAmplitude caOut = (ComplexAmplitude)caIn.Clone();
            //run through all sampling points in x and y direction
            for (int runSamplingY = 0; runSamplingY < caOut.SamplingParameters.SamplingPoints.Y; runSamplingY++){
                for (int runSamplingX = 0; runSamplingX < caOut.SamplingParameters.SamplingPoints.X; runSamplingX++){
                    //check whether locally or globally polarization
                    if (caOut.IsGloballyPolarized){
                        //use Field property to multiply factor on data
                        caOut.Field[runSamplingX, runSamplingY] *= factor;
                    }
                    else{
                        //use FieldX and FieldY property to multiply factor on data
                        caOut.FieldX[runSamplingX, runSamplingY] *= factor;
                        caOut.FieldY[runSamplingX, runSamplingY] *= factor;
                    }
                }
            }

            //return input field
            return caOut;
        }
    }
}
```

例: DLLの活用

- Snippet内のコード:

```
1 //create output HarmonicFieldsSet
2 HarmonicFieldsSet hfsReturn = new HarmonicFieldsSet();
3
4 //run through all members of the harmonic fields set
5 for(int runMember = 0; runMember < InputField.Count; runMember++){
6     //generate instance of the class which is defined within DLL
7     FieldManipulation manipulation = new FieldManipulation();
8     //call method of the class that multiplies the factor on the input field
9     ComplexAmplitude caMulitply = manipulation.MulitplyComplexAmplitudeWithFactor(InputField[runMember],
10                                                                                      FactorToMultiply);
11
12     //add the manipulated ComplexAmplitude to the output HarmonicFieldsSet
13     hfsReturn.Add(caMulitply);
14 }
15 //return the output harmonic fields set
16 return hfsReturn;
```

C++ DLLの活用

- C++ - DLLを活用する事が可能です

- ー C++ DLL内の“extern”ステートメントを活用します:

```
#ifdef __cplusplus
extern "C"
#endif
HEADER_USER_DLL void SetFrequency(double frequency);
```

- ー C# DLLを書いて、“DllImport”ステートメントを用いて、C++機能を活用します

```
[DllImport(@"User.dll")]
public static extern void SetFrequency(double frequency);
```

- ー インポートされた機能の活用例:

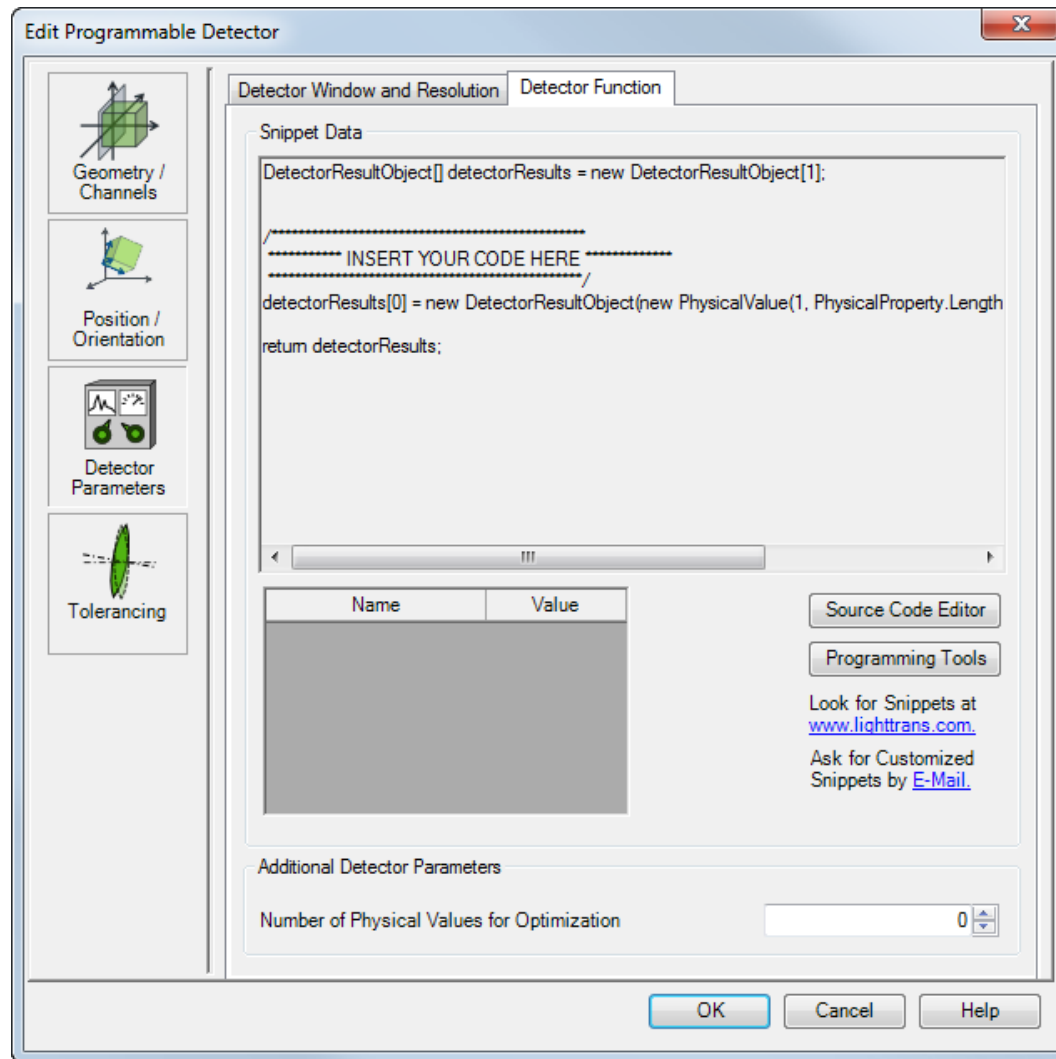
```
UserDLL.SetFrequency(2.0 * Math.PI / wavelength);
```

プログラマブル・ディテクター

プログラマブル・ディテクターのコンセプト

- Virtual Labのプログラマブル・ディテクターはユーザーが光学系解析に用いたいメリットファンクションを活用する事を可能とします
- プログラマブル・ディテクターは、入射フィールドを評価し、任意の数のディテクターを作成するSnippetにより定義されます
(タイプ: ハーモニックフィールドセット)
- さらに、ユーザーはパラメトリック最適化に用いる数値の定義が可能です。
これはディテクターの編集ダイアログにより設定可能です。

プログラマブル・ディテクターの編集ダイアログ



ディテクターの結果

- プログラマブル・ディテクターは、Snippetにより、入射フィールドに応じて DetectorResultObject のデータアレーを算出させます

- 例:

```
//declare array of DetectorResultObject of dimension 1
DetectorResultObject[] detectorResults = new DetectorResultObject[1];
//specify first detector result to be a PhysicalValue with the detectorname "My Detector"
detectorResults[0] = new DetectorResultObject(new PhysicalValue(1, PhysicalProperty.Length, "My Detector Result"),
                                              "My Detector");
```

- DetectorResultObjects の一般的なデータタイプ:
 - List<PhysicalValue>
 - DataArray1D
 - DataArray2D

出力: List<Physical Value>

- 数値の出力には、List<PhysicalValues>を用います
- PhysicalValueObjectには3つの異なる情報が含まれます(数値、物理的プロパティ、コメント)
- PhysicalValueClassは、VirtualLabAPI.Core.Numericsに含まれます
- 例:

```
//generate new list of physical values
List<PhysicalValue> listResults = new List<PhysicalValue>();
//add a new physical value [1m, Comment: Length) to the list
listResults.Add(new PhysicalValue(1, PhysicalProperty.Length, "Distance"));
```

出力: DataArray1D

- `dataArray1D`を発生するには、いくつかの作成法があります
- `completion`をサポートするVisual Studio
- `Namespace; VirtualLabAPI.Core.Numerics`
- 例:

[illegible]

出力: DataArray2D

- DataArray2Dを発生するにはいくつかの作成法があります
- completionを含むVisual Studio
- Namespace: VirtualLabAPI.Core.Numerics
- 例:

```
//generate data field of dimension 20x20
ComplexField cfData = new ComplexField(new Vector(20,20));
//fill with 2 (only for demonstration purpose)
cfData.Fill(2);
//generate new data array
DataArray2D dArray2D = new DataArray2D(new ComplexFieldArray(cfData), //the data within the dataarray
    new PhysicalProperty[] { PhysicalProperty.AngleDeg }, //physical property of data
    new string[] { "My Angles" }, //comment of data
    0.2, //sampling distance x-coordinate
    -10, //start coordinate x-coordinate
    PhysicalProperty.Length, //physical property of x-coordinate
    "X", //comment of x-coordinate
    1, //sampling distance y-coordinate
    0, //start coordinate y-coordinate
    PhysicalProperty.Percentage, //physical property of y-coordinate
    "Power"); //comment of y-coordinate
```

Data Arrayについて

- DataArray1DとDataArray2Dは複合データタイプで、VirtualLabの標準の出力形体です
- Data Arrayのサンプリングは、等間隔あるいは非等間隔でも構いません
- 等間隔あるいは非等間隔のデータアレーに関わらず、適正な作成法を用いる必要があります

例

- プログラマブル・ディテクターの例
 - Snippet 028: Diffractive Optics Merit Functions for Harmonic Fields Set Detector
 - Snippet 027: Coherence Detector

まとめ

- VirtualLabはユーザー定義したシミュレーション手法を用いて、フィールドトレーシング法をさらに拡充する事が可能です
- チュートリアル、マニュアル、プログラミング参照資料などプログラミングをサポートする資料を用意しております
- 更に、有償で下記のサポートをご希望の場合、LightTrans社及び代理店にご一報ください：
 - － LightTransによるプログラミング講習
 - － ユーザーが作成したソリューション(Snippet, モジュールなど)のサポート
 - － LightTrans社によるカスタムソリューション開発