**ソースコード解説**

**はじめに**

　自分でも光学設計アプリケーションを作ってみたいと、このプログラムの内部構造に興味を持たれる方もいらっしゃると思います。そのような方に向け簡単な説明をと思いましたが、良い方法が思いつきませんでした。すべての処理について説明するのも量ばかり多く、あまり意味があるとは思えません。そこでデータ構造の説明といくつかの収差の計算手順などを追いながら、あまり煩雑にならないように説明をしてみました。

　素人が試行錯誤の末作ったものですので、ソフトウェア製作の常識からすると「やってはいけないことの例」満載だと思います。バカなことをやっていると笑っていただければ幸いです。自分ならもっとエレガントにやれると、モチベーションアップのきっかけにしていただければ何よりです。

　現在のPOPSのコードを修正してより高機能や使いやすさを目指すのは時間の無駄だと思います。まっさらの状態から書き起こす方が良いものが出来ると思います。コーディングの途中で迷ったりしたときに、この人はどのようにやっているのだろうかと参考にしていただくのが良いと思います。

　ソースコードを直接エディタで開くのではなく、Delphiの統合環境からプロジェクトを読み込んでいただき、『計算』などのボタンを押したときのイベントコードをたどっていく形でコードを追っていってください。

**1、プログラムのデータ構造**

このプログラムを構成する主なクラスは以下の通りです。実際のソースコードでは、使用したDelphiの慣例によりクラスネームの頭にＴがついています。

Ｃで書いたときに光学系の諸元を構造体というデータ構造にまとめました。オブジェクト指向でのプログラミングの必要から、構造体をそのままクラスに移行しました。

変数名やメソッド名は長くなるのをいとわず、出来るだけ省略しないで内容が分かるように付けています。メソッド内のローカル変数には、お決まりのループカウンターの　i,j,kなど以外は、aRayなどと不定冠詞を付けるようにしています。

またクラス名、変数名は英語名を付けていますが、英語が出来ないので辞書を引いて適当につけました。英語として正しいか、光学の用語として正しいかは全く分かりません。

主要なクラスは

OpticalSurface（光学面）

OpticalMedium（媒質）

OptSystem（光学系）

D3Ray（３次元光線）

ParAxialRay（近軸光線）

です。

クラスの構成としては、光学系(OpticalSystem)は光学面(OpticalSurface)と媒質(OpticalMedium)から構成され、光線(D3Ray,ParAxialRay)が光学系を通ることにより変化するという具合に考え、このようなクラス構成を作りました。このクラス設計が適切かどうかは自信がありません。

以下諸クラスから生成されるインスタンスが持つ変数（プロパティ）の説明をします。

**1.1　OpticalSurfaceクラス**・・・ユニットOptSurfaceで定義

Curvature （曲率）

Radius （曲率半径）

Diameter （口径）

Power （パワー）：各波長をインデックスとする配列

PowerZ （水平垂直の面の曲率が違うときの水平方向のパワー）

ConicCoefficient （非球面の円錐係数）

TiltSheeta （光軸がチルトしている時のＸ軸とのなす角度）

TiltPhai （ＸＹ平面となす方位角）

BiasSheeta （面が偏芯している時のＸ軸とのなす角度）

BiasPhai （ＸＹ平面となす方位角）

TiltMatrix （光軸がチルトしている時の変換行列）：TMatrixで定義される

　　　　　　　　　　　　　　　　　　　　　　　　　　　行列

SiftVector （光軸がシフトしている時のベクトル）:TVectorで定義される

　　　　　　　　　　　　　　　　　　　　　　　　　　　　　ベクトル

BiasVector 　（面が偏芯している時のベクトル）

BiasSiftVector 　（面がシフトしている時のベクトル）

AsphericalTerm （非球面の項数）

AsphericalCoefficient　（非球面係数）

HighOrderTerm （任意時数曲面の項数）

HighOrderCoefficient （任意時数曲面の係数）

HighOrderAngle （任意時数曲面の傾角）

HighOrderSin （任意時数曲面のサイン）

HighOrderCos （任意時数曲面のコサイン）

HighOrderPowerY （任意時数曲面の冪数）

HighOrderPowerZ （任意時数曲面の冪数）

CylinderCz （トーリック面のＺ軸曲率）

CylinderAngle （トーリック面の傾き）

CylinderSin

CylinderCos

ApertureFigure （開口形）

ApertureBrockout （中心遮光径）

ApertureY （開口形高さ）

ApertureZ （開口形幅）

ApertureSiftY （開口形Ｙ軸シフト量）

ApertureSiftZ （開口形Ｚ軸シフト量）

ApertureSin

ApertureCos

FrenelZeroZorn （フレネル面のゼロゾーン径）

FrenelPich （フレネル面のピッチ数）

FrenelBack （フレネル面のセットバック量）

TiltSwitch （光軸チルト時のフラッグ）

SiftSwitch （光軸シフト時のフラッグ）

BiasSwitch （面偏芯時のフラッグ）

FrenelFlag （フレネル面時のフラッグ）

AspFlag （非球面時のフラッグ）

HighOrderSwitch

ConicSwitch

CylinderFlag

ApfigureSwitch

ReflectSwitch

ApproxChangeFlag

MeridionalLine（レンズ面を表示するときの３次元座標点）

SagittalLine （レンズ面を表示するときの３次元座標点）

OutLine （レンズ面外形を表示するときの３次元座標点）

MLineReal （レンズ面を表示するときの実座標３次元座標点）

SLineReal（レンズ面を表示するときの実座標３次元座標点）

OLineReal（レンズ面外形を表示するときの実座標３次元座標点）

Delphiではこれらの変数はprivateとして定義し、外部からはpropatyとしてアクセスする方法にしています。

インスタンス変数が多すぎますが、どのようにしたらよいのか分かりませんでした。

**1.2　OpticalMedium クラス**・・・ユニットOptMediumで定義

Distance 　　　 （面間隔）

ExchangeDistance （屈折率で割った換算面間隔）

RefractiveIndex （屈折率）

AbbeNumber　　　　　（アッベ数）

GlassName （ガラス名）

**1.3　OptSystem クラス**・・・ユニットOptSystemで定義

OpticalName （光学系の名称）

TotalSurfaceNumber （光学面数）

TotalSpectruNumber （追跡波長数）

OpticalSurface （光学面数分の配列にopticaisurfaceクラスから生成されたインスタンスが入る。静的に配列で確保しています。）

OpticalMedium （光学面数分の配列にopticalMediumクラスから生成されたインスタンスが入る。配列のインデックスはTspectruを定義して[nd]などとして使っている）

SpectruList 　　　 (追跡波長数に応じて生成されるスペクトルリスト)

FirstSpectru 　　　（追跡する最初の波長）

LastSpectru 　　　 （追跡する最後の波長）

SpectruString 　　　（追跡波長の文字列表示）

EffectiveAperture (有効径)

Stop （絞り面となる光学面）

EntrancePupil （入射瞳位置）

ExitPupil （射出瞳位置）

ObjectPlane 　（物点面）：opticaisurfaceクラスから生成されたインスタンス

ObjectDistance 　（物点距離）

FocusPlaneDistance （焦点面距離）

FocusPlane 　（焦点面）：opticaisurfaceクラスから生成されたインスタンス

BackFocus 　　　（バックフォーカス）：追跡波長をインデックスとする配列

ObjectSideBackFocus （物側バックフォーカス）：追跡波長をインデックスとする配列

PrincipalPoint 　（主点）：追跡波長をインデックスとする配列

ObjectSidePrincipalPoint　　(物側主点)：追跡波長をインデックスとする配列

FocalLength 　　　　　　（物側主点）：追跡波長をインデックスとする配列）

BackFocusSagittaru　　　　　　（サジタル面バックフォーカス）

ObjectSideBackFocusSagittaru （サジタル面物側バックフォーカス）

PrincipalPointSagittaru 　　　（サジタル面主点）

ObjectSidePrincipalPointSagittaru （サジタル面物側主点）

FocalLengthSagittaru （サジタル面焦点距離）

AfocalSwitch 　（アフォーカル系フラグ）

SynmetrySwitch （対称形フラグ）

MxsinSwitch （臨界角フラグ）

RayCheckSwitch （光線チェックフラグ）

NaInputSwitch （開口入力フラグ）

FishEyeSwitch （魚眼フラグ）

**1.4　ParAxialRay クラス**・・・ユニットRayで定義

h （光線高）：光学面数と追跡波長数の2次元配列

a （換算傾角）：光学面数と追跡波長数の2次元配列

s （光軸交点距離）：光学面数と追跡波長数の2次元配列

u （傾角）：光学面数と追跡波長数の2次元配列

ObjectSideDistance （物点距離）

ReverseObjectDistance （逆追跡物点距離）

**1.5　D3Rayクラス**・・・ユニットRayで定義

ax, ay, az （松井吉哉「レンズ設計法」P42.図3.2　*Ｔｖ-1　）*

axi, ayi, azi （同図　*Ｔｖ-1　）*

dxi, dyi, dzi （同図　*Ｑ　）*

amx, amy, amz （同図　*Ｍ　）*

al, am, an （同図　*Ｅ　）*

dxia, dyia, dzia

xmd, dim

qntp, objd

pi, qi, deltaQi, gi

d12, pathLength

vv, vf, cosi, cosid

sini2, sinid2

cofP1, cofP2, cofP3

frnlbh

h, hxv, hyv, hzv （松井吉哉「レンズ設計法」P42.図3.2　*Ｔ　）*：光学面数個の配列

vdxi, vdyi, vdzi （同図　*Ｑｖ+1　）*：光学面数個の配列

sinmx

asSwitch

vyhat, vhyi, uyi

vzhat, uzi

vdfy, vdfz, vdfy2

vdfz2, vdfyz

rot, ros

a11, a21, a12, a22

b11, b21, b12, b22

ryi, rzi, ryzi

dq1yh, dq1zh

dq2yh, dq2zh

dr1yh, dr1zh

dr2yh, dr2zh

tas, sas

K3dastFlag （ 光線が 非点収差を計算されているかどうかの判定

１のとき初期値　追跡式を呼び出すたび＋１する）

IhightFlag （フレネル面で光線がハイゾーンか否かの判定）

IeabsFlag （非球面での逐次近似で収束したかの判定で使用）

Evold （非球面での逐次近似で球面と非球面との差）

　POPSを起動するとOpticalSystemという名前でOptSystemクラスからインスタンスが一つ作られます。バックアップ用などにOptSystemからインスタンスが作られますが、基本POPSプログラムの中には一つの光学系という構造です。

ただ焦点面は別の一つの光学系と考えてFinalOptという名前でOptSystemクラスから生成しています。OpticalSystemのFocusPlaneの面形状がコピーされて使われます。

光線（D3Ray、ParAxialRay）は随時生成がされます。

**2、メソッドの説明**

**2.2　近軸追跡**

近軸追跡メニューで開く、近軸値フォームの『近軸追跡』ボタンクリックを追っていくとたどることが出来ます。ボタンクリックイベントの中の、 ParAxialTrace（　）で近軸値計算をしています。

●ParAxialTrace（　）・・・OptSystemクラスにPublicで実装

ParAxialTraceの中で下記のメソッドを順次実行することで近軸追跡をしています。

CalculatePower・・・・・・・・面のパワーの計算。OptSystemクラスに実装

CalculateExchangeDistance・・・換算面間隔の計算。OptSystemクラスに実装

ParAxialTraceSynmetry・・・・対称形の場合の近軸追跡計算。OptSystemクラスに実装

ParAxialReverseTrace・・・・・・逆近軸追跡計算。OptSystemクラスに実装

CalculateEffectiveAperture・・・・有効径計算。OptSystemクラスに実装

CalculateEntrancePupil(aOptSys)・・入射瞳計算。OptSystemクラスに実装

CalculateExitPupil(aOptSys)・・・・射出瞳計算。OptSystemクラスに実装

　これらのメソッドでOpticalSystemのインスタンス変数となっている近軸値が書き換えられます。

光学系第一面が有効径より大きく設定されていることが良くあります。CalculateEffectiveApertureは第一面の半径を像高としてaStepごとに像高を減じていき、光線が通過してところで半分のステップで元に戻し、通過しなければまた半分進むという半歩法で有効径を求めています。

その他の近軸計算は、一般の光学書の計算式そのままをプログラムしています。

**2.3　収差曲線追跡**

収差曲線フォームの計算ボタンクリックで追っていきます。ボタンクリックイベントの中に書かれている、 CalculateSphericalAberrationで計算をしています。

●CalculateSphericalAberration・・CurveBrowserユニットに実装

有効径（aDr）、収差曲線の入射高などを設定し３次元光線（a3Dray）を生成してOpticalSystem.RayTraceInAspherical(a3DRay, n)で光線追跡をします。追跡終了後の３次元光線（a3Dray）のベクトルから球面収差（SA[n,k]）を計算しています。

　SA[]やOSC[]などの収差量は光学系（OpticalSystem）インスタンスに保持させるのではなく、収差曲線フォームに属する変数として定義しています。これはスポットダイヤグラム計算フォームでも同様にしています。

　非点収差の計算では主光線を決定する必要があります。CaluculateAS・・CurveBrowserユニットに実装　の中のOpticalSystem.CalculatePrincipalRay(a3DRay,anAngle[k])で主光線を探索しています。メインフォームの絞り面の欄で指定された光学面で、高さが0となる光線を、最上部光線と最下部光線を決定したのち、半歩法で追い込んで計算しています。

**2.4　スポットダイヤグラム**

スポットダイヤグラムフォームの計算ボタンクリックで追っていきます。ボタンクリックイベントの中に書かれている、CulculateSpot で計算をしています。

●CulculateSpot・・SpotBrowserユニットに実装

３次元光線生成、入射角度設定、有効径（aDr）設定をしてスポット計算の角度ごとにループを回します。その際光線の初期値の方向余弦（aRay.dxi、aRay.dyi、aRay.dzi）をループの中で設定します。入射角０度、中間角、最大角で上下左右あるので８方向、合計９方向の角度を計算します。

角度ループの内で追跡波長のループを回します。d線から追跡波長数で指定されている波長までです。

波長ループの内でスポットダイヤグラムの光線本数分ループを回します。ループを回すごとに光線の初期値の入射瞳上の位置座標（aRay.aｙ、aRay.az）をそのつど設定しOpticalSystem.RayTraceInAspherical(aRay, n)で光線追跡をします。

角度、波長、光線数の３重ループとなります。

　スポット数ぶんの光線の（D3Ray）インスタンスを生成し初期化などして追跡した方がスッキリしますが、メモリーをたくさん使うのでやめたという経緯があります。昔メモリーが少ないころにコーディングしたときの名残です。

焦点面での光線座標（スポット図）は、焦点面を光学面数一面の光学系として、光学系最終面通過座礁と方向余弦を入射光線の初期値としてself.finalPointCulculateで計算しています。ディフォーカスの時はこの最終面との交点計算のみでおこなっています。

CenterBrockSwitchで、光学面の中央遮光があるときは追跡を中止するようしているのですが、入射瞳中央を通る１本のみはRayCheckSwitchを操作して追跡するというトリッキーなことをしています。スポット図はこの光線の焦点面交点を中心にしているため、どうしても必要となるからです。

**2.5　コアとなるメソッド**

収差曲線やスポット計算で使われているOpticalSystem.RayTraceInAspherical(aRay, n)は色々な所で使われます。このメソッドの中で使われるPrivateメソッドのAsphericalTrace(var Ray:TD3Ray; color:TSpectru)がPOPSのコアとなる３次元光線追跡のメソッドです。３次元光線と追跡波長を引数として各光学面の通過点を計算します。

●AsphericalTrace（var Ray:TD3Ray; color:TSpectru）・・OptSystemユニットに実装。

self.SurfaceSpace(Ray, k) ・・・入射瞳と第一面との距離の処理

self.ComputeVectorM(Ray ,k)・・・光線への垂直ベクトル計算

self.AxisTiltSift(Ray ,k)・・・光軸がシフトチルトしている時の処理

self.AxisBiasStart(Ray,k)・・・面を傾けているときの 処理

self.FrlCylAspTransact(Ray, k, color)・・面がフレネル面、円筒面、非球面の時の処理

self.NormalizeVectorE(Ray, k)・・・面法線によるノーマライズ

self.IntersectionStock(Ray, k)・・・交点のストック

self.Raycheck(Ray, k)・・・光線が光学面内にあるかチェック

self.OpticalPathLength(Ray, k, color)・・・光路長ストック

self.ComputeSinCos(Ray, k, color)・・・全反射チェック

self.ComputeVectorQ(Ray, k)・・・面通過後ベクトル計算

self.RefractionStock(Ray, k)・・・面通過後ベクトル保存

self.OpticalPathLength(Ray, k, color)・・・光路長保存

self.Astigmatizum(Ray, k)・・・アスを計算しているときの処理

self.FrenelEnd(Ray,k) ・・・フレネル面の終わりの処理

self.AxisBiasEnd(Ray, k) ・・・面が傾いているときの終わりの処理

self.AstigmatizumLast(Ray)・・・アスを計算しているときの終わりの処理

以上の様なメソッドで非球面、非共軸の追跡が行われます。面形状や偏心のフラッグでメソッド実行の判定を行っています。

各メソッドのコメントに、対応する草川先生の本のFORTRUNプログラムの行番号を{180-185}のように振ってあります。全体のフローは異なっていますが、草川本の本文の式解説と合わせて読めば参考になると思います。

FrlCylAspTransact(Ray, k, color)の中では、フラグ分岐でフレネル面、円筒面、非球面の光線通過点計算メソッドを呼び出しています。メソッドの入れ子が2～3段ほど深くなっています。各面形状での交点を求める式は一番複雑に感じるところです。

**2.6　ワイヤーフレーム表示**

構成図のワイヤーフレーム表示フォームの再表示ボタンクリックイベントでDrawが呼び出されます。

●Draw・・・D3VewBrowserに実装。

Drawの中の以下のメソッドで計算をしています。

OpticalSystem.CaluculateWorldCoordinatesInReal・・光学系ワイヤーフレームの実数値をワールド座標で計算。

OpticalSystem.ConvertWorldCoordinatesRealToInt(magnification)

・・ワイヤーフレームを整数値に変換。

OpticalSystem.RotationInWorldCoordinates( (degreeToRadian(Rx)),(DegreeToRadian(Ry)), (DegreeToRadian(Rz)) ) ・・ワイヤーフレームの回転を計算。

OpticalSystem.TransferVewCoordinates・・ワイヤーフレームをビュー座標に変換。

OpticalSystem.TransferInVewCoordinates(0,0,OriginDistance) ・・ワイヤーフレームをビュー座標に変換。

ここまでのメソッドはOptSystemに実装。

以下のメソッドはD3Vewに実装しています。

self.VewPointRotation((degreeToRadian(0)),(DegreeToRadian(0)), (DegreeToRadian(0)) )・・ビュー座標の回転の処理

self.CenterProjection・・中心射影座標計算。

光学面の形状はOpticalSurfaceのインスタンス変数の

MeridionalLine（レンズ面を表示するときの３次元座標点）

SagittalLine （レンズ面を表示するときの３次元座標点）

OutLine （レンズ面外形を表示するときの３次元座標点）

MLineReal （レンズ面を表示するときの実座標３次元座標点）

SLineReal（レンズ面を表示するときの実座標３次元座標点）

OLineReal（レンズ面外形を表示するときの実座標３次元座標点）

に保持されます。最初のOpticalSystem.CaluculateWorldCoordinatesInRealの中でのメソッドOpticalSurface[k].CulculateSurfacePointInRealで各面の形状データが実数値で計算されます。これは３次元グラフィックでいうボディー座標に当たり、各レンズ面の頂点を原点とした座標値です。このままでは全部のレンズ面が重なってしまっています。これをレンズ第一面の頂点としたワールド座標系に再配列します。再配列したのち光学面のシフト、チルト、光学軸のシフト、回転を計算しています。

その後整数値に変換してフォームのスライダで指定された回転の処理をし、ビュー座標を計算します。後はDelphiのPaintBoxというグラフィックパネルにラインを引いていきます。

　光線のラインはD3Rayのインスタンスを生成しOpticalSystemに通すと―おなじみのOpticalSystem.RayTraceInAspherical(aRay, n)メソッド―、D3Rayのインスタンス変数に各光学面通過座標が記録されますから、面形状のデータと全く同様の手順で変換すれば光線が描けます。

　２次元の構成図もほぼ同様の手順です。実数値でデータを計算して整数値に直しています。ヴュー投影する必要はありませんからそのままDelphiのペイントボックスにラインを引いています。

　3次元表示は以下の書籍を参考にしました。

『入門グラフィックス』佐藤義雄著　株式会社アスキー

『応用グラフィックス』佐藤義雄著　株式会社アスキー

『応用グラフィックス』太田昌孝　竹内あきら　大口孝之　株式会社アスキー

古書でしか手に入らないと思いますが、本当の初歩から応用まで実に丁寧に書かれている本です。

**2.7　MTF**

MTFはスポットダイヤグラムの計算と一緒に計算しています。像高は0から最大像高まで9段階、追跡波長数のMTF用スポットを計算した後、オプトロニクス社「光学入門」の178ページ5.21の公式で幾何光学的MTFを計算しています。

●CulculateMTF・・SpotBrowserユニットに実装。

**2.8　収差の自動補正**

　添付文章の「変化表の最小二乗法での最適化について」で説明している原理でプログラムを組んでいます。

　データセットのボタンで現在のデータを一時退避させた後計算を始めます。

●DeltaCurvetureSetting・・SpotCorrectionユニットに実装

　曲率な微小変化させたときのスポット、各収差の微小変化量を計算して、連立方程式の微係数を設定しています。

●LSQ(Sender) ・・SpotCorrectionユニットに実装

　連立方程式を最小二乗法で解き、新しいデータを生成します。

●NewDataSettingCurveture・・SpotCorrectionユニットに実装

　新しいデータに書き換えて収差図を書き換えます。

**3、データの保存と読み込み**

光学系データの保存はデータ表示グリッドをＣＶＳファイルなどで保存するのが一般的かと思いますが、生成したオブジェクトをシリアライズしてディスクに保存、読み込んでオブジェクトとして動的に生成という形にしたいと思いました。しかしその様な形式にするのは大変難しく、プログラミングスキルがないため断念しました。

　そこでOpticalSurface、OpticalMediumクラスにそれぞれのプロパティをディスクに書き出すメソッドOutputToFile、OutputToTextFileメソッドを作りました。

　OpticalSystemクラスにもOutputToFileメソッドを作り、すべてのプロパティをディスクに書きだし、OpticalSystemがプロティとして保持しているOpticalSurface、OpticalMediumも書きだすという構造にしました。ディスクからの読み込みに関しても全く同様な形をとりました。

　POPSを形にすることが出来たSmalltark上のプログラミングでは簡単にオブジェクトの生成、保存が出来たのでそれに近い形にしたいと思ったからです。このようにしたからと言って特に利点があったわけではありませんが、データの表示グリッドの形式を変えた場合でも、保存メソッドを書き換えたりする必要がなかったことくらいでしょうか。

////////////////////////////////////////////////

まだ続きます・・・

////////////////////////////////////////////////

参考文献

『レンズ設計法』松居吉哉　共立出版

『レンズ光学』草川徹　東海大学出版