# PDLS
# Photoshopプラグイン

Jun Hirabayashi jun@hirax.net

http://hirax.net/pukiwiki/pukiwiki.php?PDLS

# 内容

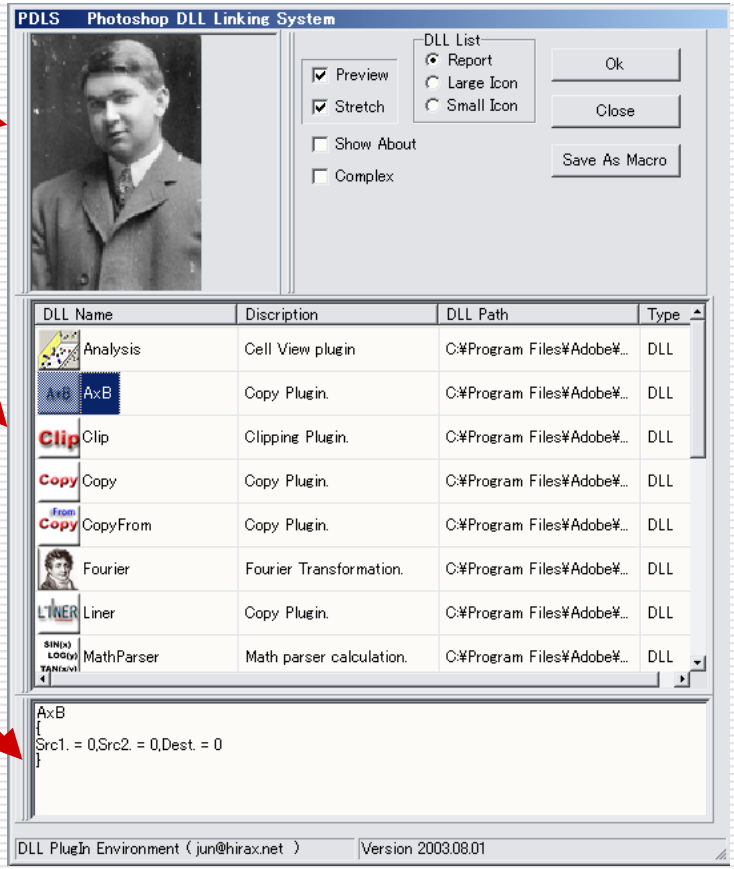
- PDLSって何？
- PDLSで動くプラグイン
- PDLSのマクロ
- PDLSのインストール
- PDLSのプラグインを開発する

# PDLSプラグインの特徴

- Photoshopプラグインとして使うアナタに
  - 16bitやCMYKモード等のプラグイン開発を楽にできる
  - 画像演算や種々の表示（鳥瞰図・セル表示）が便利
- 二次元複数データ処理をしたいアナタに
  - Mathematicaは処理速度が遅い！
  - Excelは横のサイズの上限が256。問題外。
  - Photoshopで浮動小数点を扱う
- マクロを使うことができる
  - GUI作業を自動化可能、マクロでプラグイン開発可能

# PDLSのメイン･ダイアログ

- プレビュー画面
- プラグイン一覧
  - クリックだけで起動･処理
- マクロ出力ウィンドウ
  - 処理内容は自動でマクロに

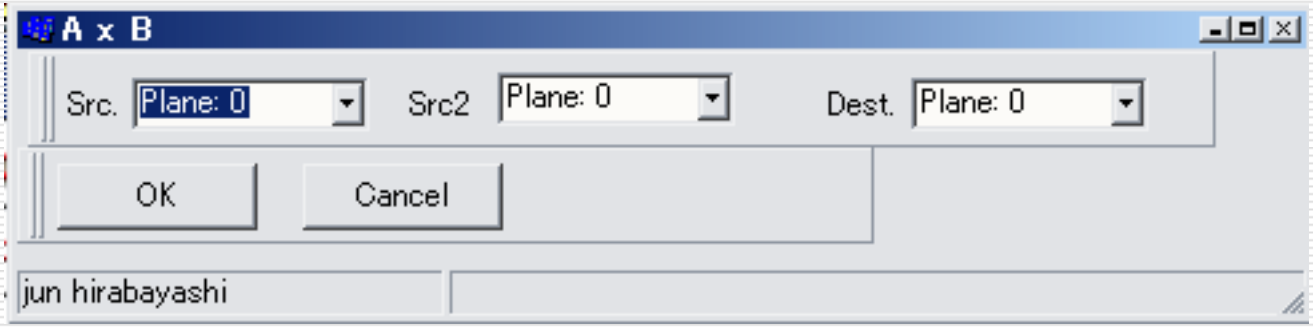


# 標準添付のプラグイン一覧

- AxB
- Clip
- Copy
- Liner
- MathParser
- Multiply
- Normalize
- Fourier
- Subtraction
- Surfaceplot
- Table

# AxB

- チャンネル間で掛け算を行う
- 行列の掛け算

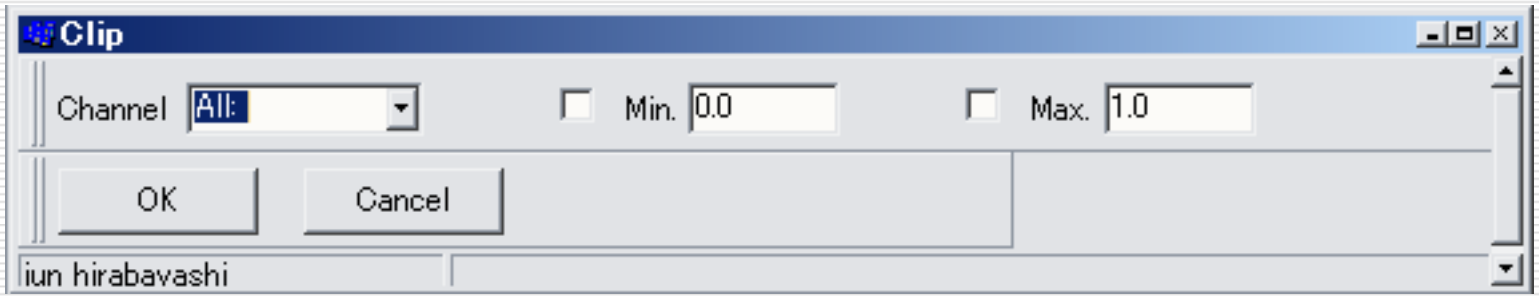



- マクロ記述

```
AxB
{
Src1. = チャンネル1,Src2. = チャンネル2,Dest. = 答を代入するチャンネル
}
```

# Clip

## □ 値を最小値･最大値でクリッピングする

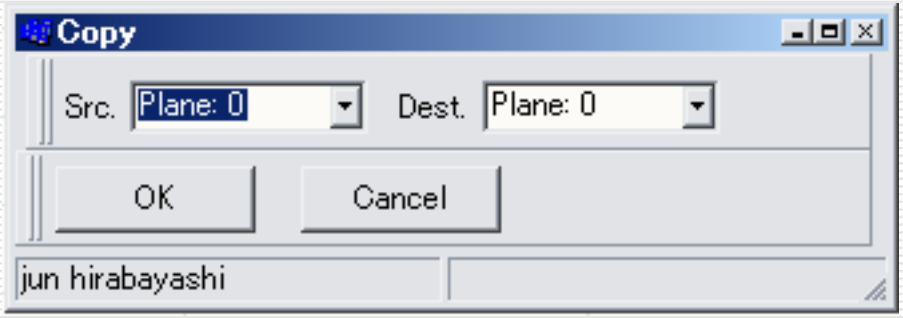



### □マクロ記述

```
Clip
{
Min Clip = 最小値でクリップするか,Max Clip = 最大値でクリップするか,Min.=
最小値,Max.=最大値,Channel=処理チャンネル
}
```

# Copy

- **データをチャンネル間でコピーする**

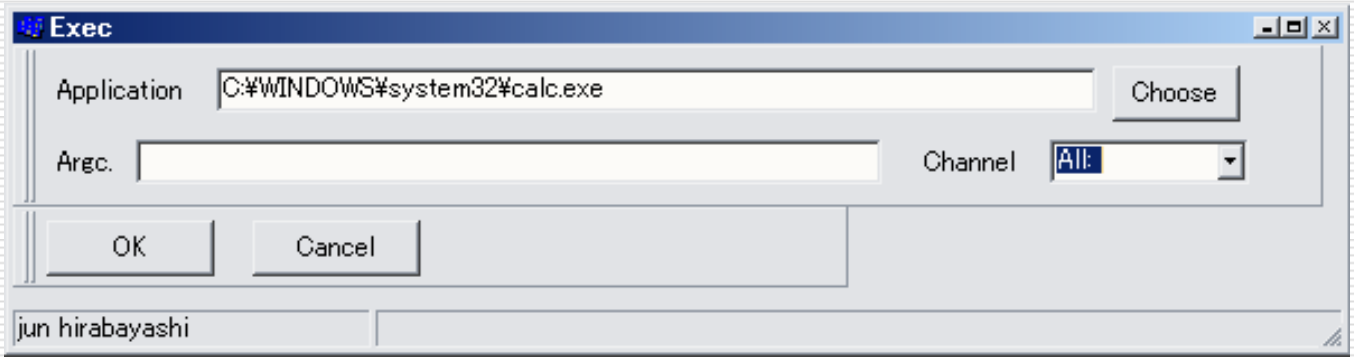



- **マクロ記述**

```
Copy
{
Src. = コピー元チャネル,Dest. = コピー先チャンネル
}
```

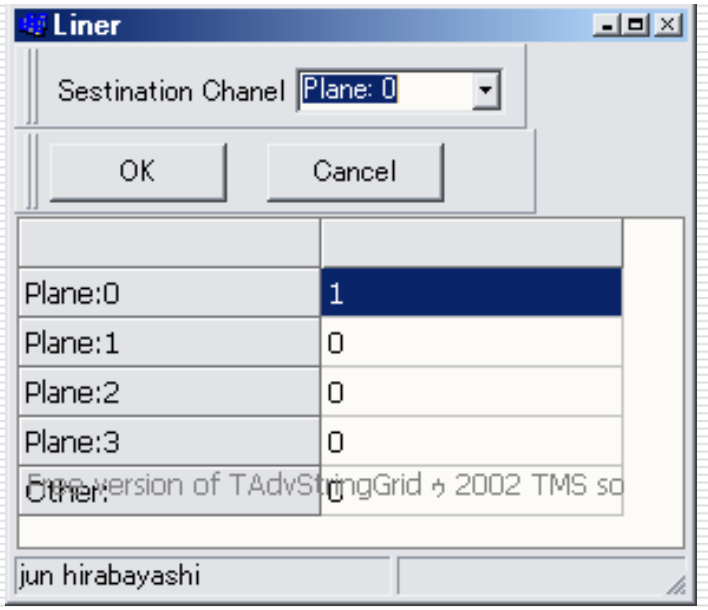
# EXEC

- 引数をsystem()でアプリケーション起動
- 起動したアプリケーションの出力がマクロに戻る



# Liner

- チャンネル間で線形演算をする

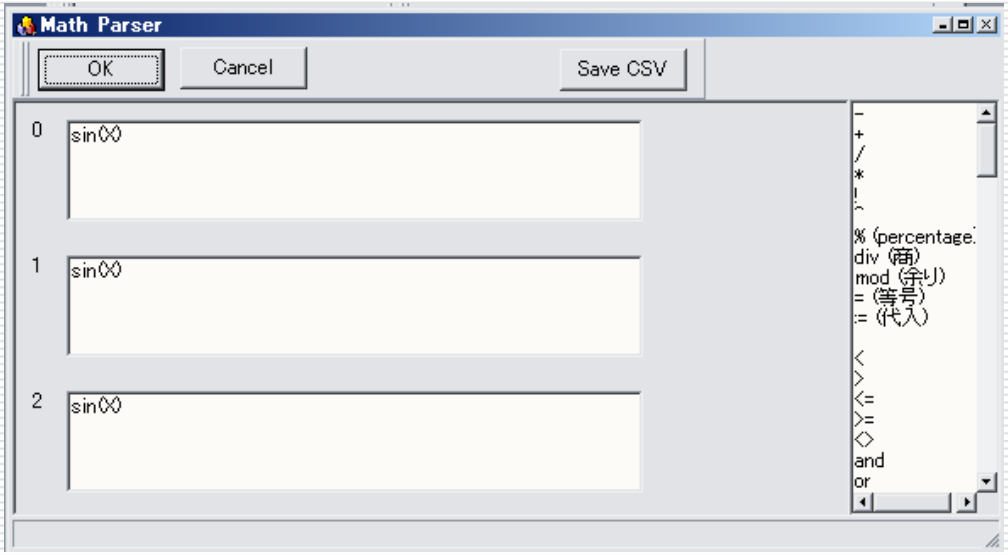



- マクロ記述

```
Liner
{
Dest. = 0,Chanel0=1,Chanel1=0,Chanel2=0,Chanel3=0,Other=0
}
```

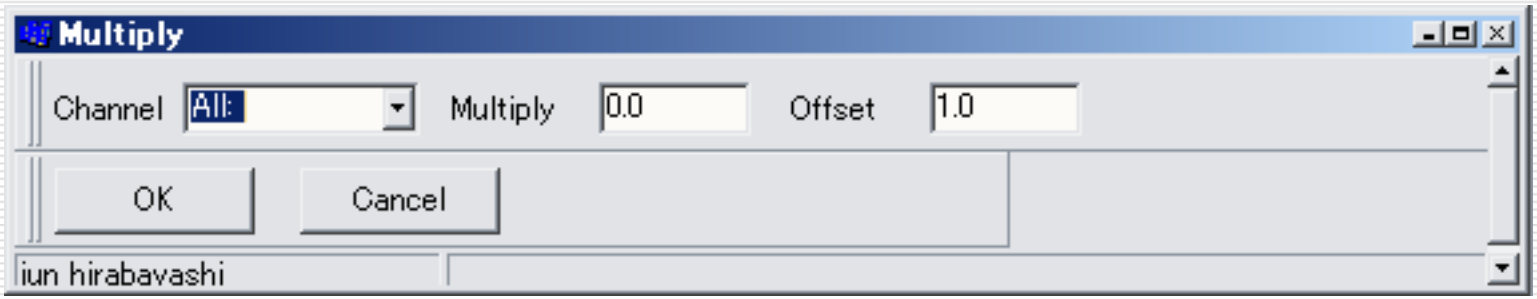
# MathParesr

- **数式を使ったフィルタ処理ができる**
- FilterFactory**では**RGB，8bit**のみ**
- **現在はマクロ非対応**



# Multiply

## 画像値に一定値を掛ける

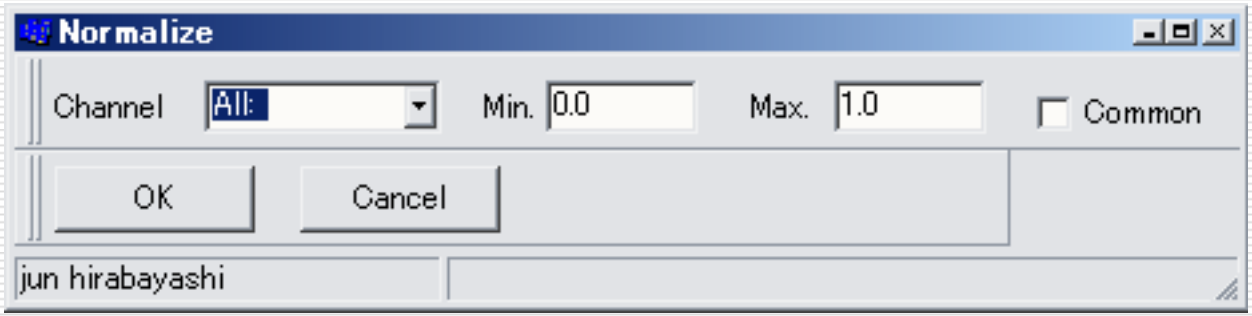



### マクロ記述

```
Multiply
{
Multiply=掛ける値,Offset=オフセット値,Channel=処理チャンネル番号
}
```

# Normalize

## □ 最大値･最小値に応じて値を変える

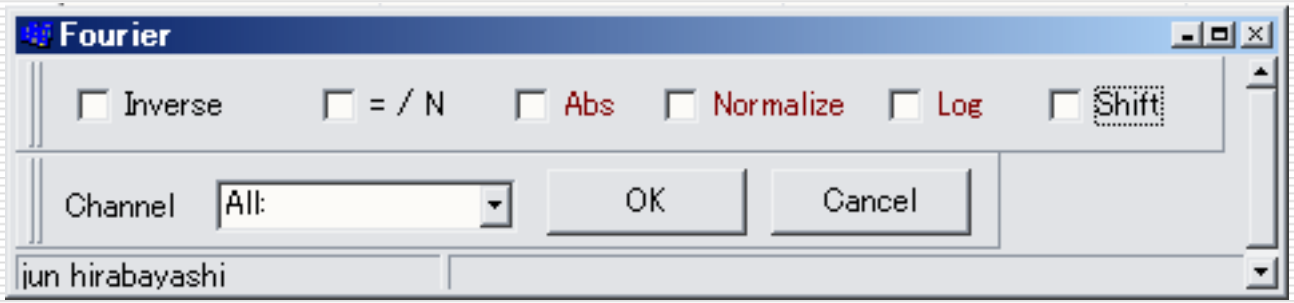



### □マクロ記述

```
Normalize
{
Min.=0,Max.=1,Common=0,Channel=-1Offset=0Scale=0
}
```

# Fourier

- フーリエ変換を行う
- 複素数モード時はFourier逆変換も自由自在

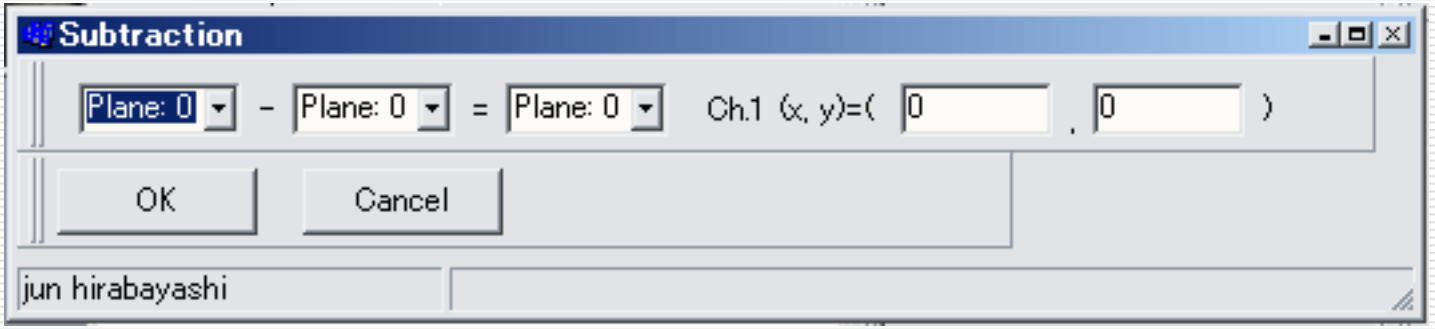



- マクロ記述

```
Fourier
{
IsInverse = 0,IsDivN = 0,IsAbs =0, IsNormalize=0, IsLog=0, IsShift=0,
Channel=-1
}
```

# Subtraction

## チャンネルの適当な位置の値どうしで減算をする

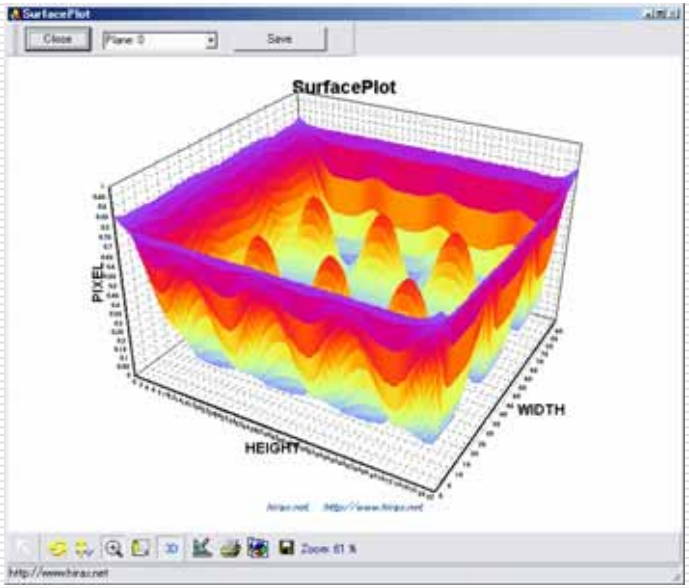



### マクロ記述

```
Subtraction
{
Channel1=0,Channel2=0,Channel3=0,Xoffset=0,Yoffset=0
}
```

# Surfaceplot

- 鳥瞰図表示
- グルグル回して眺める
- 描いたグラフはコピー
- グラフをファイル保存も可能

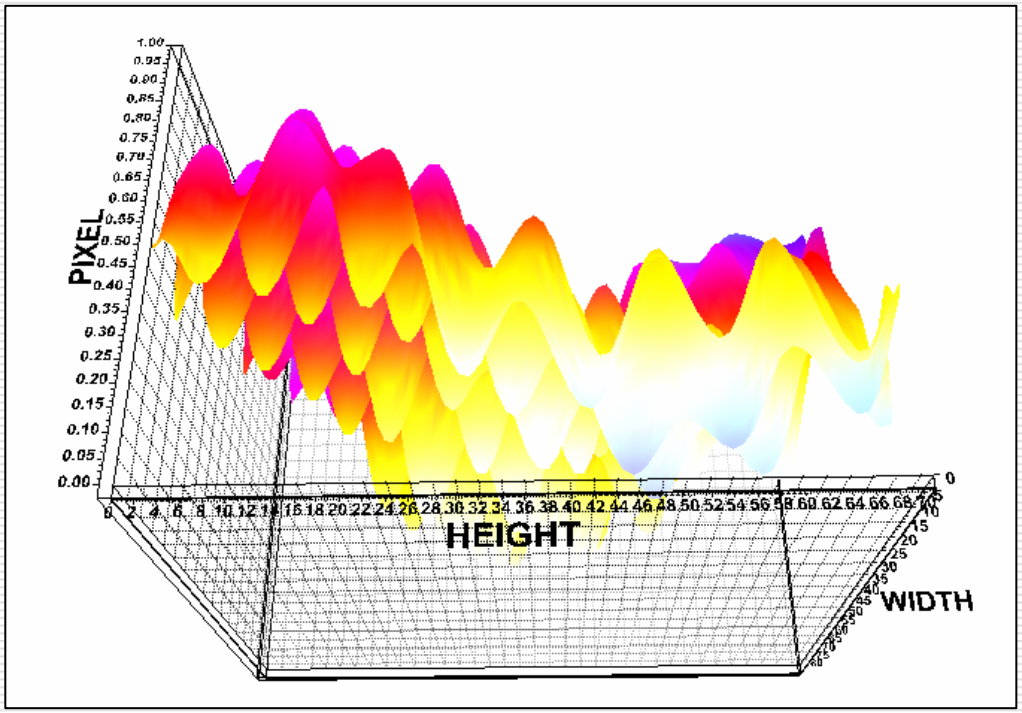



- マクロ記述

```
SurfacePlot
{
Channel=処理チャンネル,Save=保存するか,SaveFileName=保存ファイルパス
}
```

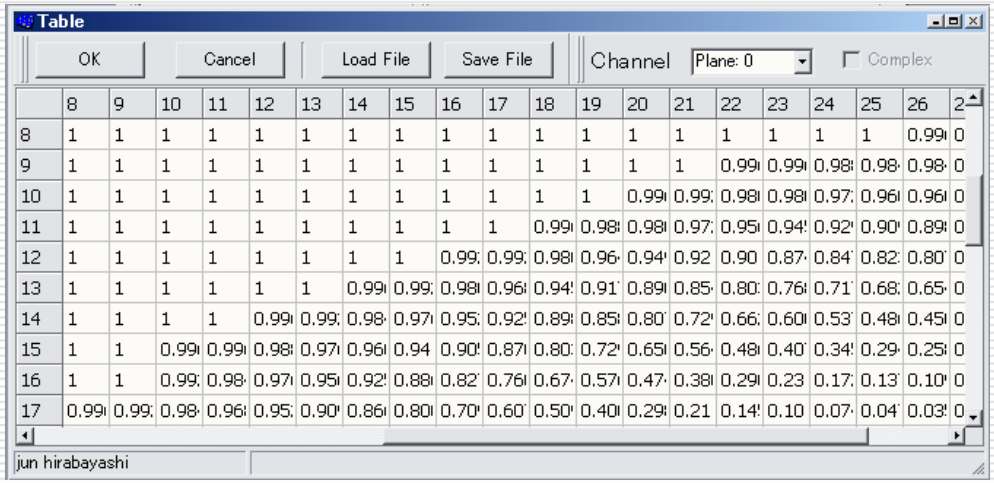
# SurfacePLot2

- **レンダリングエンジン→OpenGL**



# Table

- 表計算ソフトのようなセル表示･編集ができる
- CSV形式でデータを保存や読み込みも可能
- 複素数対応
- コピー･ペースト自由




- マクロ記述

```
Table
{
Channel=処理チャンネル,Save=CSV保存するか,SaveFileName=保存ファイル,Load=読み込むか,LoadFileName=読み込みファイル
}
```

# PDLSプラグインのマクロ機能

- マクロを使った自動作業や機能拡張ができる
  - GUIで作業した内容がマクロで出力される
  - 出力された内容を保存→マクロプラグインのできあがり
  - マクロ言語を覚える必要はない
  - マクロで新たなプラグインを作ることもできる
  - マクロ･プラグインをGUIから使うこともできる
- Photoshopの「自動バッチ処理」
  - 最後にPDLSを用いて行った一連の作業が自動実行

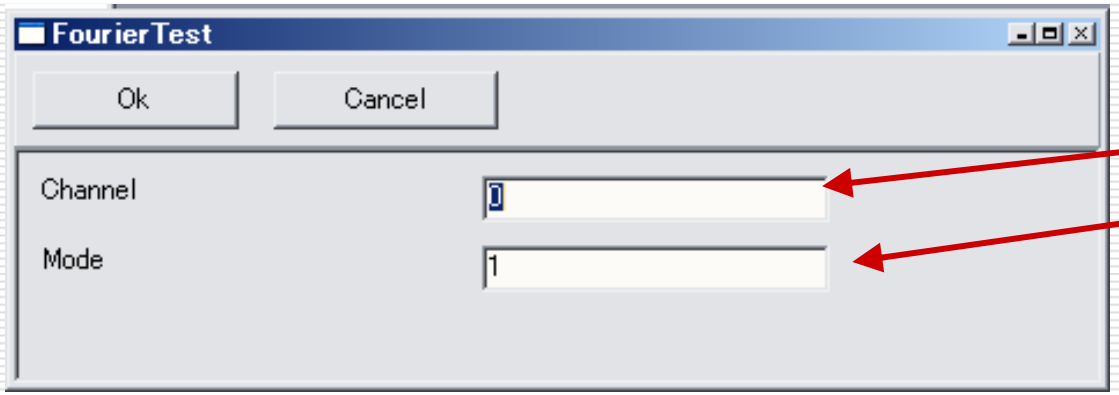
# 変数を定義できるマクロ･プラグイン

- Def関数を冒頭で使い、変数を宣言できる
  - 例えば、Fourier関数を少し変えたマクロ例

```
Def
{
Channel=0,Mode=1
}
Fourier
{
Mode,1,0,0,0,1,Channnel
}
```

# 変数使用マクロのGUI使用もできる

- 変数定義したマクロプラグインのGUIモード利用
  - ダイアログが自動生成→使用時に変数値設定も自由




```
Def
{
Channel=0,Mode=1
}
Fourier
{
Mode,1,0,0,0,1,Cha
nnnel
}
```

# 新たな機能追加の例

- Subtraction(減算)を使い加算機能を追加した例
- ex. “Add.mcr”

```
Def
{
Src1=0,Src2=1,Dest=2
}
Multiply
{
Multiply=-1,Offset=0,Channel=Src2
}
Subtraction
{
Channel1=Src1,Channel2=Src2,Channnel3=Dest,Xoffset=0,Yoffset=0
}
```

# Def定義の中で変数や数式処理ができる

- Def定義中での変数や数式処理が可能
  - 相対的な処理が簡単にできる
- “_”で始まる変数はGUIダイアログでは非表示
- “$”で始まる変数は文字列→数式処理はしない

```
Def
{
Src1=0,_Src2=Src1+1,_Dest=Src1+2
}
Multiply
{
Multiply=-1,Offset=0,Channel=_Src2
}
Subtraction
{
Channel1=Src1,Channel2=_Src2,Channnel3=_Dest,Xoffset=0,Yoffset=0
}
```

# PDLSのインストール

- Photoshop.exeがあるディレクトリ
  - DllPluginsフォルダ
  - historyフォルダ
  - PDLS.dll
- プラグイン-フィルタディレクトリ
  - Dllmulti.8bf

# プラグイン開発

- プラグイン＝ダイナミックライブラリ
- Windows, Unix(コンソール。プログラム)共通コード
  - Windows＝*.dll, Unix＝*.so
- 開発環境例
  - g++
  - Visual C++
  - Borland C++, C++ Builder
- Photoshopを忘れたプログラミングはお気楽

# 処理関数＝DoFilter()

- DoFilter(**変数群**)
  - float *fpData, **画像値へのポインタ** Plane**順**
  - int iWidth, **画像の幅**
  - int iHeight, **画像の高さ**
  - int ColorMode, Plane**数**
  - char *cParameters, **パラメータ受け渡し文字列**
    - 4096**文字まで**
  - int iComplex, **複素数モード時**＝1, **実数モード時**＝0

# その他の関数

- bool DoAbout(void)
  - **アバウト画面表示時の処理を行う**
- bool Discription(char *cDiscription)
  - **プラグインを説明する文字列を返す**
- bool IsOK(int ColorMode)
  - **現在のカラーモードに対応しているかを返す**

# プラグインのアイコン

- プラグインの一覧表示に使われるアイコン
- DllPluginsディレクトリに置く
  - ***.dllというプラグインの場合
  - ***.bmp, ***s.bmp
- ***.bmp
  - 64 x 64 ピクセルのBitmap
- ***s.bmp
  - 16 x 16 ピクセルのBitmap

# データの扱い

- 浮動小数点　float 4 Byte
  - 精度は十分
- 複素数扱い　*2＝8Byte (実数モードでは *1)
  - 周波数空間でもフィルタリングが簡単
- 整数処理をしたい場合には32bit, or 64bit処理
  - 整数処理でも精度は十分

# コンパイル

- Borland C++ Buidler, Visual C++
  - **サンプルプロジェクトでメイク一発**
- Borland C++
  - bcc32 -WD -ps hoge.cpp
- g++ on Linux
  - g++ -shared hoge.cpp -o hoge.so