今回から新しい話(前回までの成果は引き続き使うが)

USB メモリの中身

20110906/
Fourier.tar.gz
RBCodeHighlighting_MoveEverythingInThisFolderToContributed.zip
SSK_20111005_051900.st
VisualWorks771ncWithJun790ForMac.zip
VisualWorks771ncWithJun790ForWin.zip

RBCodeHighlighting_MoveEverythingInThisFolderToContributed.zip の使い方が再度説明されたが、前回説明を書いたので省く

File Browser から SSK_20111005_051900.st を VisualWorks771ncWithJun790ForMac/ へコピー

SSK_20111005_051900.st を File in (容量が大きいので、今回は少し長めに時間がかかる)

File in すると Transcript にこのように表示される

```
Filing in from:
	/Users/fumiya/Desktop/20111005/VisualWorks771ncWithJun790ForMac/SSK_20111005_051900.st
```

SSK Package の中身

SSK-Fourier	FFT のコード
SSK-Pane	前回まで作っていたものと同じ
SSK-System	便利グッズとして登場 この中の ProgramManager Class saving save を使うと、 File out しなくても保存が出来るものがある この save は SSK Bundle の中に入っているものはすべからく保存する

 Bundle　Bundle の中には Bundle か Package が入っている

 Package　クラスやメソッドが入っている

名前空間を定義せずにやってきたが、これからはちゃんと名前を定義する

SSK, SSK-System, SSK の Definition

```
Smalltalk defineNameSpace: #SSK
	private: false
	imports: '
			private Smalltalk.*
			'
	category: 'SSK-System'
```

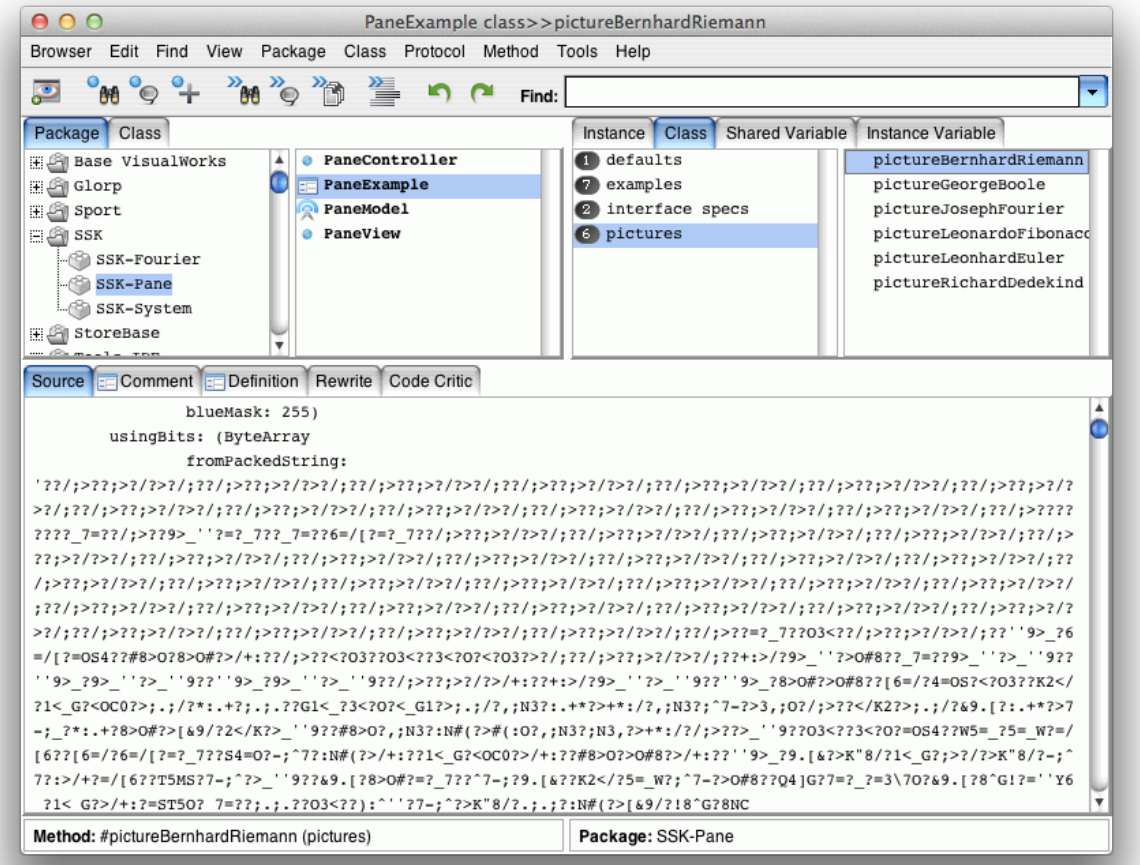
こんな感じ。

念のため Pane が以前通りちゃんと動くのか確認

以前は picture ディレクトリに画像が保存されていたはずだが、無い！

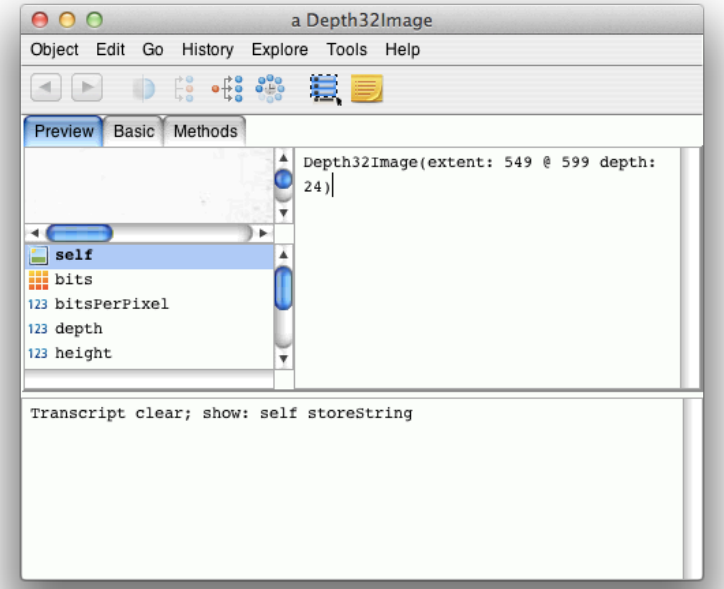
で、何処に行ったかというと、

PaneExample, Class, picture に文字列として保存されている！



コレの作り方は
pictureBerhardRiemann のコメントを Inspect it してみる

```
pictureBernhardRiemann
	"SSK.PaneExample pictureBernhardRiemann."

	^Image
		extent: 549 @ 599
		depth: 24
		bitsPerPixel: 32
		palette: (Graphics.FixedPalette
				redShift: 16
				redMask: 255
				greenShift: 8
				greenMask: 255
				blueShift: 0
				blueMask: 255)
		usingBits: (ByteArray
				fromPackedString:
"以下略 (画像データを文字列化したものなので)"
```



Transcript clear; show: self storeString
これを Do it で、 Transcript に 画像データが文字列として保存される

Workspace にて
123 storeString '123'

#(10 20 30) storeString '#(10 20 30)'

#(10 20 30) asOrderedCollection storeString '((Core.OrderedCollection new) add: 10; add: 20; add: 30; yourself)'

#(10 20 30) asOrderedCollection printString 'OrderedCollection (10 20 30)'

storeString は結局のところデータを(シリアライズして)文字列に置き換えてくれるものなのである！

ちなみに、先ほどの Transcript に表示された文字列を Inspect it してみると、やはり、画像が生成される。

((#(10 20 30) asOrderedCollection)
	add: #(100 200 300);
	yourself) storeString
このように、OrderedCollection の中に OrderedCollection が入っているようなものでも
'((Core.OrderedCollection new) add: 10; add: 20; add: 30; add: #(100 200 300); yourself)'
ちゃんと変換される

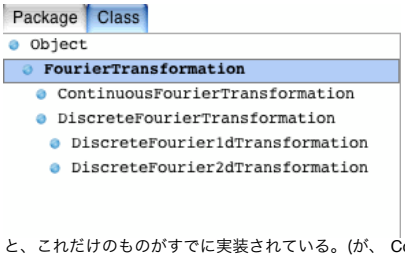
ちなみに、 CodeHighlighting はとても長いソースコードはあきらめる。

```
pictureBernhardRiemann
    "SSK.PaneExample pictureBernhardRiemann."

    ^Image
        extent: 549 @ 599
        depth: 24
        bitsPerPixel: 32
        palette: (Graphics.FixedPalette
                redShift: 16
                redMask: 255
                greenShift: 8
                greenMask: 255
                blueShift: 0
                blueMask: 255)
        usingBits: (ByteArray
                fromPackedString:
'??/;>??;>?/?>?/;??/;>??;>?/?>?/;??/;>??;>?/?>?/;??/;>??;>?/?>
```

Fourier.tar.gz の中身は京都産業大学の来年度のプロジェクト演習という授業で制作するプログラムの要求仕様などが存在するものである

Fourier/Requirement/index.html を見てみると、これから作るものがよくわかる
(暇なときに読んでおいてくださいとのこと)

SSK-Fourier, FourierTransformation



と、これだけのものがすでに実装されている。(が、 ContinuousFourierTransformation は特許の関係で、実装されて良い無い)

SSK-Fourier, DiscreateFourier1dTransformation, Class, data, dataSampleWave に、先ほどみた波形のサンプルデータが存在する
SSK-Fourier, DiscreateFourier1dTransformation, Class, examples, example1 は、サンプルの波形を csv ファイルとして、出力してくれるものである
実行してみると、 VisualWorksWithJun/Fourier1d.csv として、出力される
このファイルを Excel でグラフとして適切に処理すれば、波形として正しく出るが、自分の Pane で、ちゃんと表示してみましょう

example1 を元に作業をしていく
example2 に名前を変更し、コメント部分も 2 に変え、anArray 以下を削除
sourceSize, inverseData は proceed で、 anArray は remove it

この状態になる

```
example2
	"SSK.DiscreteFourier1dTransformation example2."

	| sourceData sourceSize aTransformation realPart imaginaryPart inverseData |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	inverseData := aTransformation inverseRealPart
```

```
example2
	"SSK.DiscreteFourier1dTransformation example2."

	| sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	aTransformation inverseRealPart.
	aPixmap := Pixmap extent: sourceSize @ 300.
	anImage := aPixmap asImage.
	aModel := SSK.PaneModel picture: anImage.
	aModel open
```

(結局、 inverseData は remove it した)
Do it すると

画面が開くだけ

```
example2
	"SSK.DiscreteFourier1dTransformation example2."
```

```
	| sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel aGraphicsContext |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	aTransformation inverseRealPart.
	aPixmap := Pixmap extent: sourceSize @ 300. "紙を持ってきて"
	aGraphicsContext := aPixmap graphicsContext.	"筆を持ってくる"
	aGraphicsContext paint: ColorValue white.	"白色に設定して" <-- Mac の場合、背景色は default で白だが、 Windows や Linux では色が異なるので、ちゃんと書いておく
	aGraphicsContext displayRectangle: aPixmap bounds.	"塗りつぶす"
	anImage := aPixmap asImage.
	aModel := SSK.PaneModel picture: anImage.
	aModel open


example2
	"SSK.DiscreteFourier1dTransformation example2."

	| sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel aGraphicsContext halfHeight |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	aTransformation inverseRealPart.
	aPixmap := Pixmap extent: sourceSize @ 300. "紙を持ってきて"
	aGraphicsContext := aPixmap graphicsContext.	"筆を持ってくる"
	aGraphicsContext paint: ColorValue white.	"白色に設定して"
	aGraphicsContext displayRectangle: aPixmap bounds.	"塗りつぶす"
	halfHeight := aPixmap height // 2.	"半分の高さを出しておく"
	aGraphicsContext paint: ColorValue black.
	aGraphicsContext displayLineFrom: 0 @ halfHeight
		to: aPixmap width @ halfHeight.	"軸を描く"
	anImage := aPixmap asImage.
	aModel := SSK.PaneModel picture: anImage.
	aModel open
```

と、このように、軸が出てくるようになった

これに先ほど csv ファイルとして、出力したものを貼り付けていけば良いのだが、そのまま貼り付けると、軸の周りに固まったようなものになってしまうので、そこを修正する

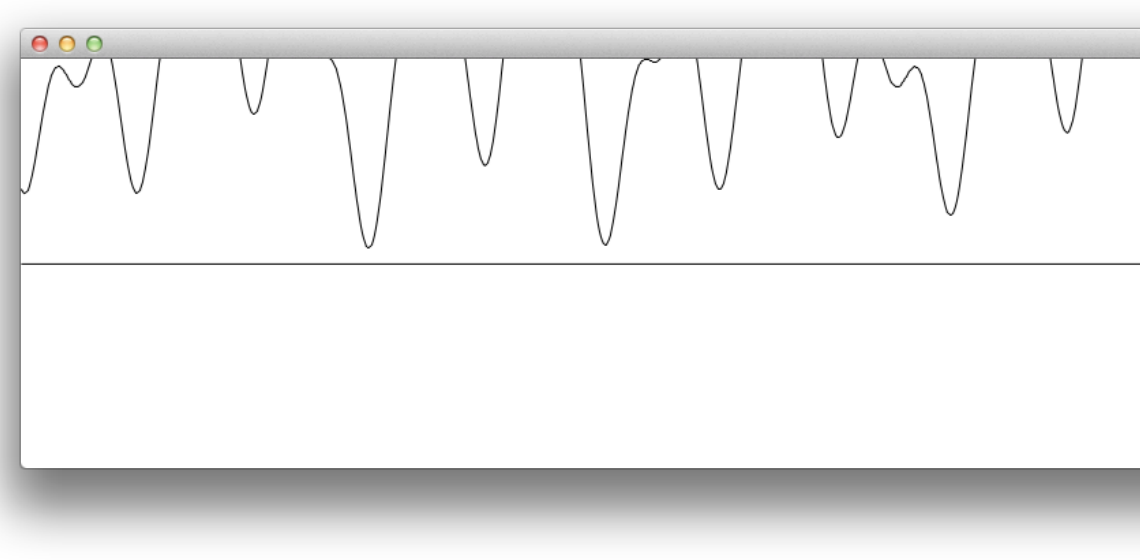
```
example2
	"SSK.DiscreteFourier1dTransformation example2."

	| sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel aGraphicsContext halfHeight maximumValue |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	aTransformation inverseRealPart.
	aPixmap := Pixmap extent: sourceSize @ 300. "紙を持ってきて"
	aGraphicsContext := aPixmap graphicsContext.	"筆を持ってくる"
```

```
        aGraphicsContext paint: ColorValue white.      "白色に設定して"
        aGraphicsContext displayRectangle: aPixmap bounds.   "塗りつぶす"
        halfHeight := aPixmap height // 2.     "半分の高さを出しておく"
        aGraphicsContext paint: ColorValue black.
        aGraphicsContext displayLineFrom: 0 @ halfHeight
                to: aPixmap width @ halfHeight.       "軸を描く"
        maximumValue := sourceData inject: 0.0d
                        into: [:maximum :value | maximum max: value abs].
        "一番大きな値に高さを調節するために、一番大きいものを取り出す"
        Transcript
                cr;
                show: maximumValue printString.    "確認のために表示"
        anImage := aPixmap asImage.
        aModel := SSK.PaneModel picture: anImage.
        aModel open
```

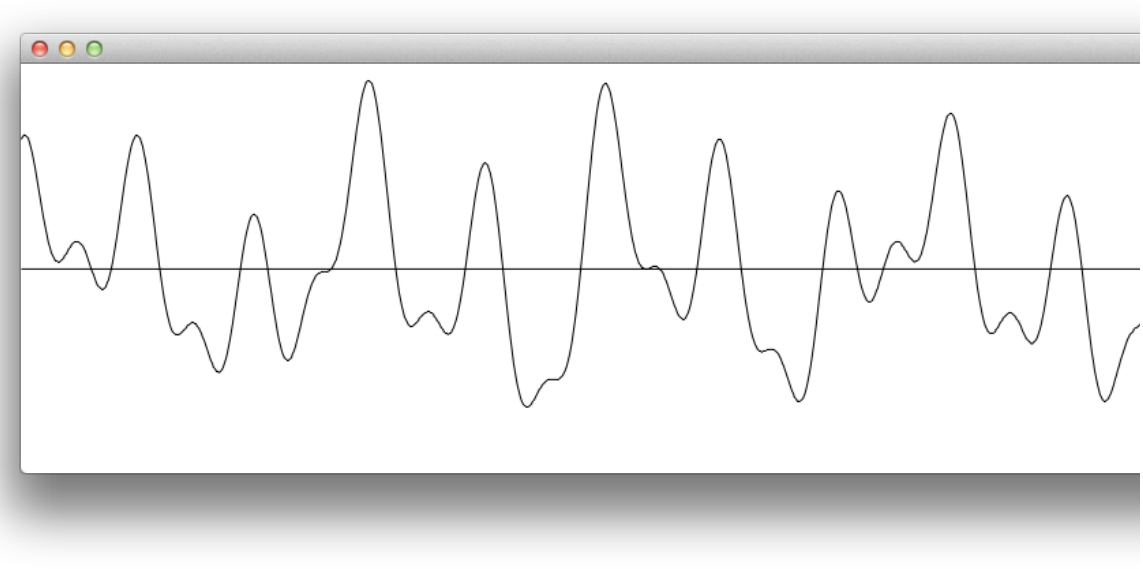
14.190187223735d が一番大きい値であると、求まっていることが確認できた

example2

```
        "SSK.DiscreteFourier1dTransformation example2."

        | sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel aGraphicsContext halfHeight maximumValue normalizedData
aPolyline |
        sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
        sourceSize := sourceData size.
        aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
        realPart := aTransformation realPart.
        imaginaryPart := aTransformation imaginaryPart.
        aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
                        with: imaginaryPart.
        aTransformation inverseRealPart.
        aPixmap := Pixmap extent: sourceSize @ 300."紙を持ってきて"
        aGraphicsContext := aPixmap graphicsContext.      "筆を持ってくる"
        aGraphicsContext paint: ColorValue white.      "白色に設定して"
        aGraphicsContext displayRectangle: aPixmap bounds.   "塗りつぶす"
        halfHeight := aPixmap height // 2.     "半分の高さを出しておく"
        aGraphicsContext paint: ColorValue black.
        aGraphicsContext displayLineFrom: 0 @ halfHeight
                to: aPixmap width @ halfHeight.       "軸を描く"
        maximumValue := sourceData inject: 0.0d
                        into: [:maximum :value | maximum max: value abs]. "一番大きな値に高さを調節するために、一番大きいものを取り出す"
        normalizedData := sourceData collect: [:value | value / maximumValue]. "元々のデータを正規化したデータ -1 ~ 1"
        aPolyline := OrderedCollection new: sourceSize.
        normalizedData with: (0 to: sourceSize - 1)
                do: [:value :index | aPolyline add: index @ (value * halfHeight)]. "高さを調節"
        aGraphicsContext displayPolyline: aPolyline.  "描く"
        anImage := aPixmap asImage.
        aModel := SSK.PaneModel picture: anImage.
        aModel open
```

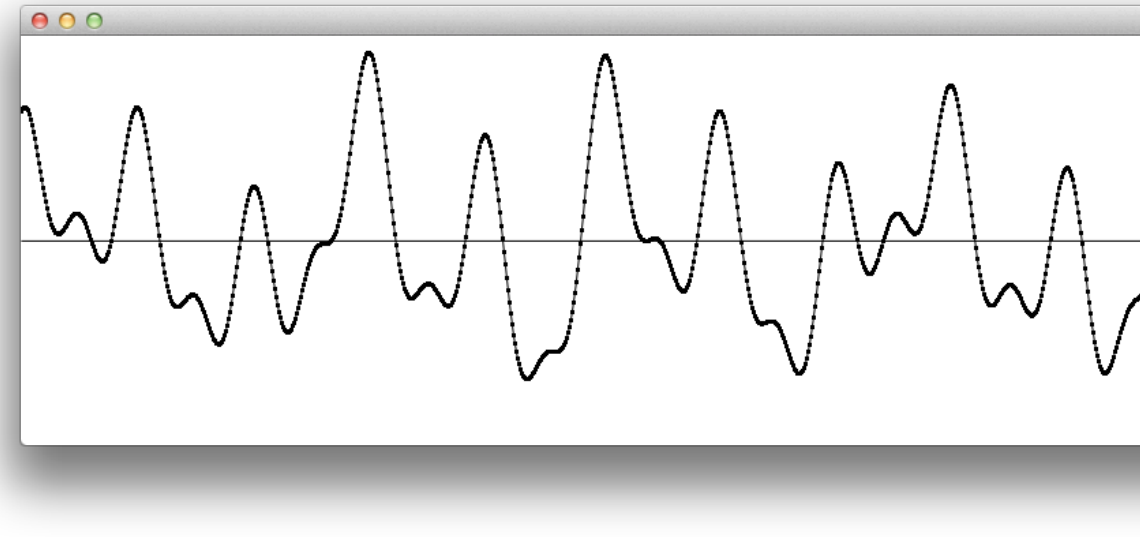


高さ調節の部分がおかしいので、



```
example2
	"SSK.DiscreteFourier1dTransformation example2."

	| sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel aGraphicsContext halfHeight maximumValue normalizedData
aPolyline |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	aTransformation inverseRealPart.
	aPixmap := Pixmap extent: sourceSize @ 300. "紙を持ってきて"
	aGraphicsContext := aPixmap graphicsContext.	"筆を持ってくる"
	aGraphicsContext paint: ColorValue white.	"白色に設定して"
	aGraphicsContext displayRectangle: aPixmap bounds.	"塗りつぶす"
	halfHeight := aPixmap height // 2.	"半分の高さを出しておく"
	aGraphicsContext paint: ColorValue black.
	aGraphicsContext displayLineFrom: 0 @ halfHeight
		to: aPixmap width @ halfHeight.	"軸を描く"
	maximumValue := sourceData inject: 0.0d
				into: [:maximum :value | maximum max: value abs]. "一番大きな値に高さを調節するために、一番大きいものを取り出す"
	normalizedData := sourceData collect: [:value | value / maximumValue]. "元々のデータを正規化したデータ -1 ~ 1"
	aPolyline := OrderedCollection new: sourceSize.
	normalizedData with: (0 to: sourceSize - 1)
		do: [:value :index | aPolyline add: index @ (value negated * halfHeight + halfHeight)]. "高さを調節"
	aGraphicsContext displayPolyline: aPolyline.	"描く"
	anImage := aPixmap asImage.
	aModel := SSK.PaneModel picture: anImage.
	aModel open

maximumValue := sourceData inject: 0.0d into: [:maximum :value | maximum max: value abs].
この inject: into: が正規化するときに良く用いられる

example2
	"SSK.DiscreteFourier1dTransformation example2."

	| sourceData sourceSize aTransformation realPart imaginaryPart anImage aPixmap aModel aGraphicsContext halfHeight maximumValue normalizedData
aPolyline |
	sourceData := SSK.DiscreteFourier1dTransformation dataSampleWave.
	sourceSize := sourceData size.
	aTransformation := SSK.DiscreteFourier1dTransformation new: sourceData.
	realPart := aTransformation realPart.
	imaginaryPart := aTransformation imaginaryPart.
	aTransformation := SSK.DiscreteFourier1dTransformation new: realPart
				with: imaginaryPart.
	aTransformation inverseRealPart.
	aPixmap := Pixmap extent: sourceSize @ 300. "紙を持ってきて"
	aGraphicsContext := aPixmap graphicsContext.	"筆を持ってくる"
	aGraphicsContext paint: ColorValue white.	"白色に設定して"
	aGraphicsContext displayRectangle: aPixmap bounds.	"塗りつぶす"
	halfHeight := aPixmap height // 2.	"半分の高さを出しておく"
	aGraphicsContext paint: ColorValue black.
	aGraphicsContext displayLineFrom: 0 @ halfHeight
		to: aPixmap width @ halfHeight.	"軸を描く"
	maximumValue := sourceData inject: 0.0d
				into: [:maximum :value | maximum max: value abs]. "一番大きな値に高さを調節するために、一番大きいものを取り出す"
	normalizedData := sourceData collect: [:value | value / maximumValue]. "元々のデータを正規化したデータ -1 ~ 1"
	aPolyline := OrderedCollection new: sourceSize.
	normalizedData with: (0 to: sourceSize - 1)
```

```
        do: [:value :index | aPolyline add: index @ (value negated * halfHeight + halfHeight)]. "高さを調節"
    aGraphicsContext displayPolyline: aPolyline.  "描く"
    aPolyline do:
                [:aPoint |
                | aRectangle |
                aRectangle := (aPoint extent: 1 @ 1) expandedBy: 1.
                aGraphicsContext displayRectangle: aRectangle]. "座標データのある部分に dot を打つ"
    anImage := aPixmap asImage.
    aModel := SSK.PaneModel picture: anImage.
    aModel open
```



dot が打たれるようになった

最後に保存
SSK-System, ProgramManager, Class, saving, save を使って！