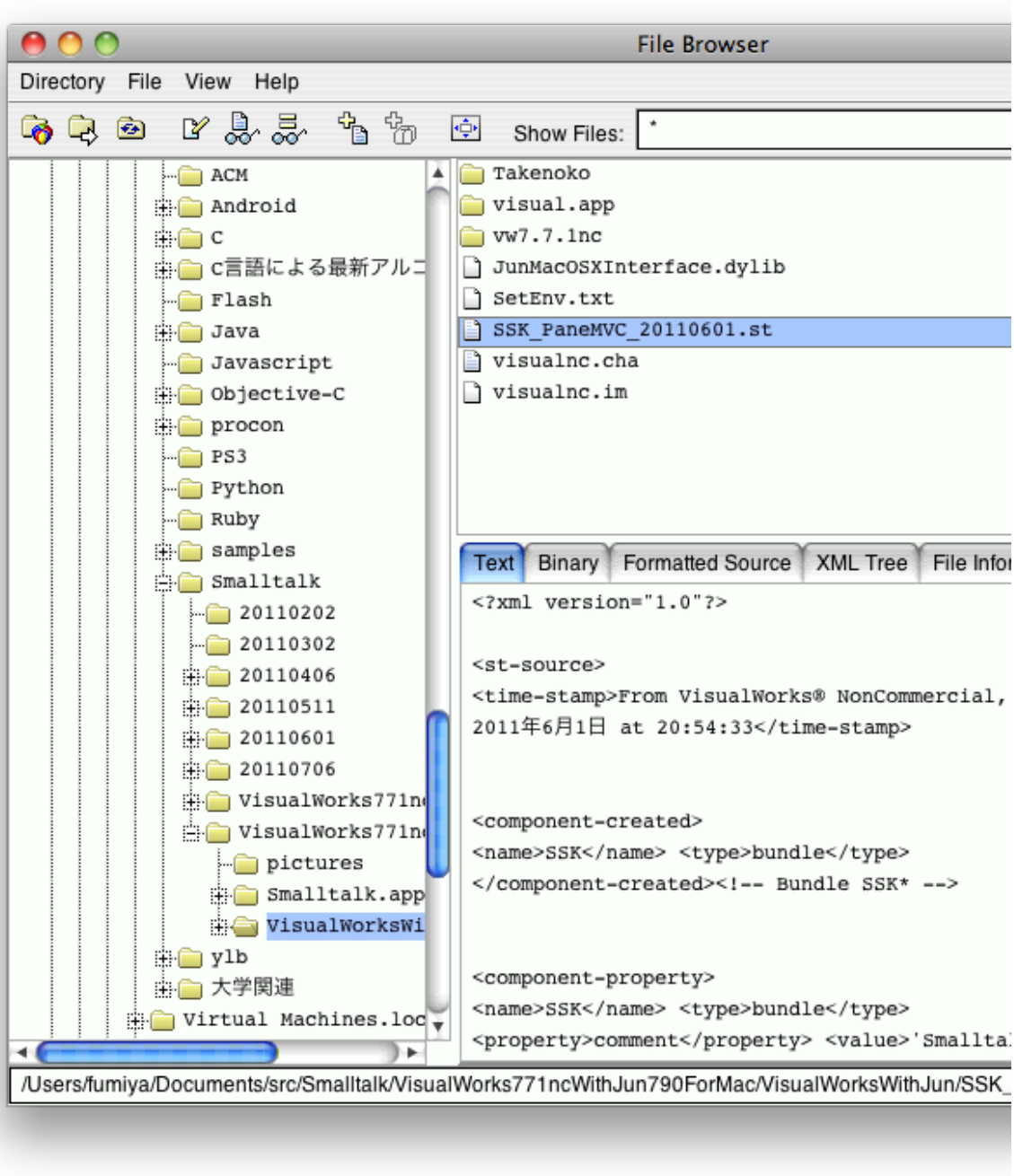
今回の USB メモリの中身は picture ディレクトリと .st が3つ

ディレクトリ構成は

pictures
Smalltalk.app
VisualWorksWithJun/
VisualWorksWithJun/SSK_PaneMVC_20110601.st

いつも通り File in



画像サイズそのままで描かれている状態が前回までの状態 (example4)
フックは出来ている

=====復習終わり

前回は Model を攻めたので、 View を攻めましょう

PaneView には Class メソッドは何もない
Instance メソッド
bounds accessing で、ウインドウサイズの変更をフック
ウインドウサイズを変更すると、 Transcript に色々表示される状態
displaying の displayOn: が表示
現在は、単にイメージを表示するだけ

displaying の displayOn を書き換え
前回までの状態

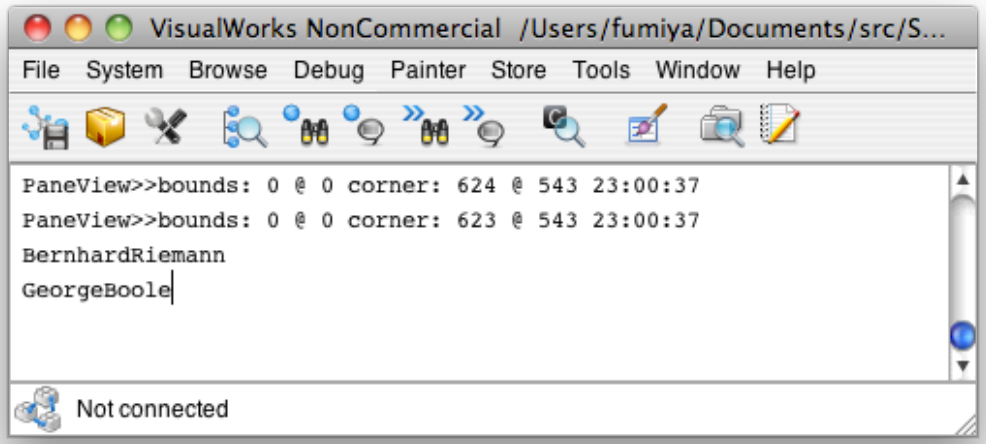
```
displayOn: graphicsContext
	"自分自身を表示する。"

	self model picture ifNotNil: [:anImage | anImage displayOn: graphicsContext]
```

```
displayOn: graphicsContext
	"自分自身を表示する。"

	self model picture
		ifNotNil:
			[:anImage |
			anImage displayOn: graphicsContext
				at: self bounds center - anImage bounds center]
```

整列させるには、 at: の部分をいじればいいはずである

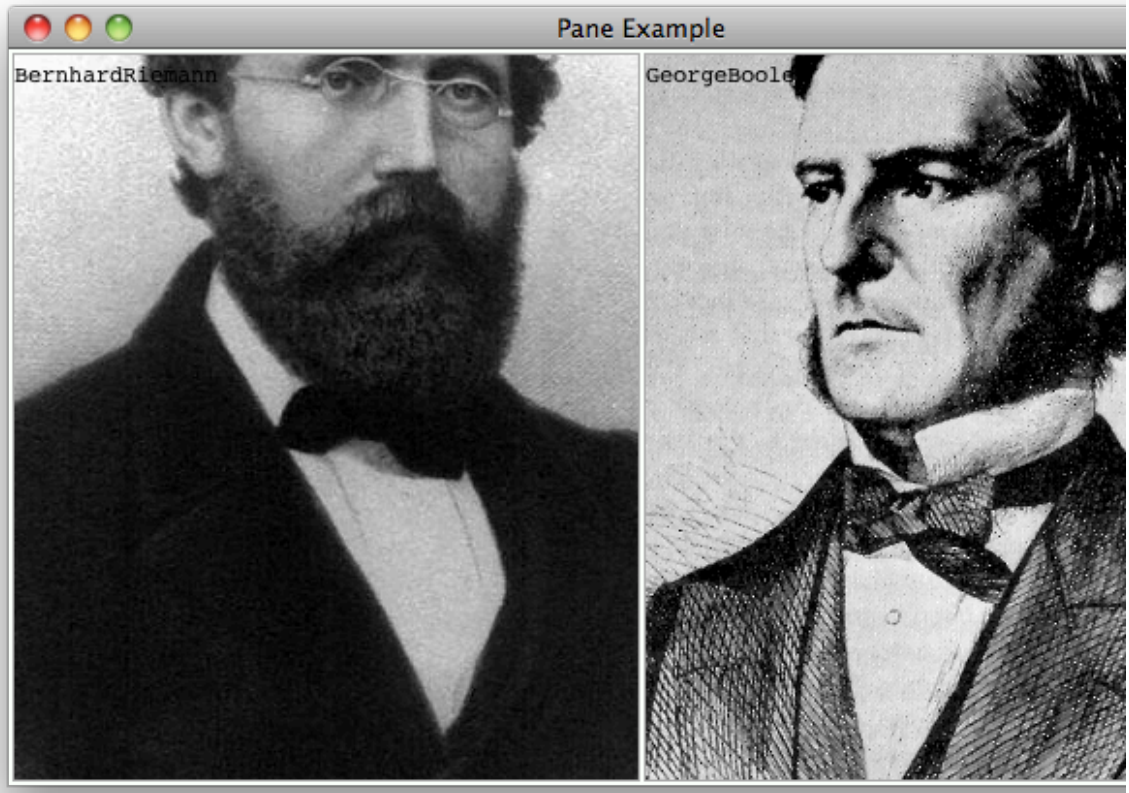
以前は、ファイル名からラベリングしていたので、ラベリングのためにデータを持ってくる
とりあえず、 Transcript に書いてみる

```
displayOn: graphicsContext
	"自分自身を表示する。"

	self model picture
		ifNotNil:
			[:anImage |
			anImage displayOn: graphicsContext
				at: self bounds center - anImage bounds center].
	self model label
		ifNotNil:
			[:aString |
			Transcript
				cr;
				show: aString]
```



ちゃんと名前が取れているので、左上に名前が表示されるように

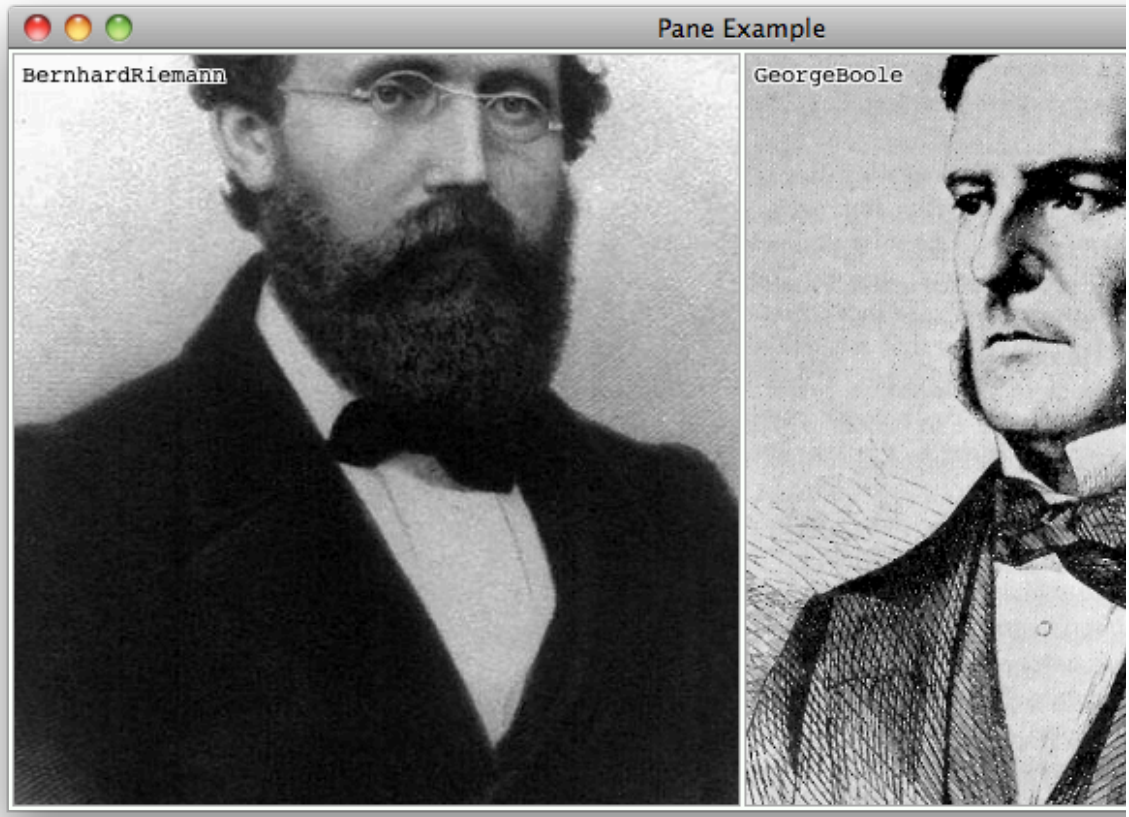
```
displayOn: graphicsContext
	"自分自身を表示する。"

	| aComposedText |
	self model picture
		ifNotNil:
			[:anImage |
			anImage displayOn: graphicsContext
				at: self bounds center - anImage bounds center].
	self model label
		ifNotNil:
			[:aString |
			aComposedText := aString asComposedText.
			aComposedText displayOn: graphicsContext at: 0 @ 0]
```



上記のように、下の画像と文字が重なると読めないので、先に白で塗りつぶして、その後文字を書くようにする

```
displayOn: graphicsContext
	"自分自身を表示する。"

	| aComposedText aPoint |
	self model picture
		ifNotNil:
			[:anImage |
			anImage displayOn: graphicsContext at: self bounds center - anImage bounds center].
	self model label
		ifNotNil:
			[:aString |
			aComposedText := aString asComposedText.
			aPoint := 4 @ 0. "<-- 文字を少し右に寄せる"
			graphicsContext paint: ColorValue white.
```

```
                (-1 to: 1)
                        do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
"<-- 本来描くべき文字のフチの部分を(x座標y座標共に)白く塗る"
                graphicsContext paint: ColorValue black.
                aComposedText displayOn: graphicsContext at: aPoint]
```



後ろの画像が黒くても、ちゃんと文字が読めるように

anImage を縮小してみましょう
とりあえず、 Jun を使わずに 1/2 にする
難しいことを考えてキャッシュをするということはしないで、毎回書き直し

```
displayOn: graphicsContext
        "自分自身を表示する。"

        | aComposedText aPoint |
        self model picture
                ifNotNil:
                        [:anImage |
                        (anImage shrunkenBy: 2 @ 2) displayOn: graphicsContext "<-- 試しに、 0.5 @ 0.5 などと入力すると、2倍になる"
                                at: self bounds center - anImage bounds center].
        self model label
                ifNotNil:
                        [:aString |
                        aComposedText := aString asComposedText.
                        aPoint := 4 @ 0.
                        graphicsContext paint: ColorValue white.
                        (-1 to: 1)
                                do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
                        graphicsContext paint: ColorValue black.
```

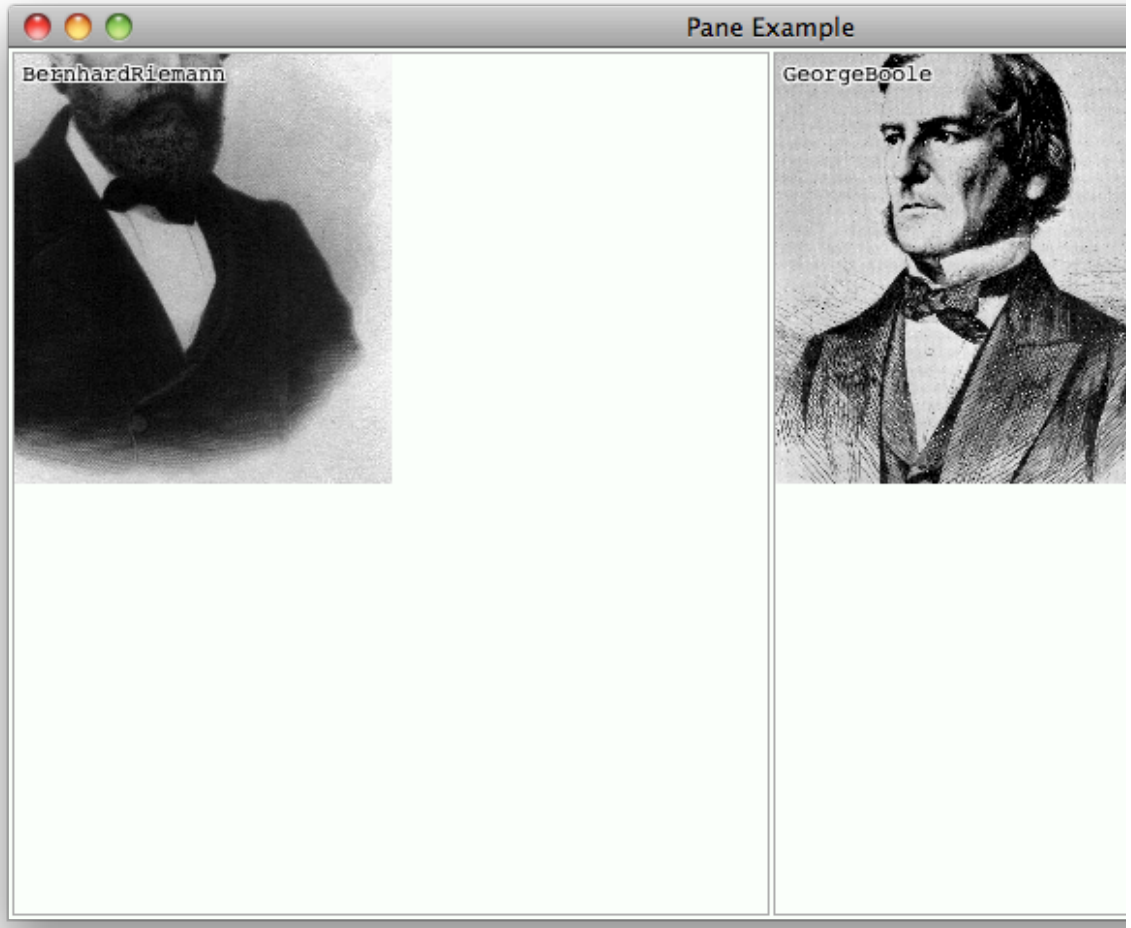
```
                aComposedText displayOn: graphicsContext at: aPoint]
```

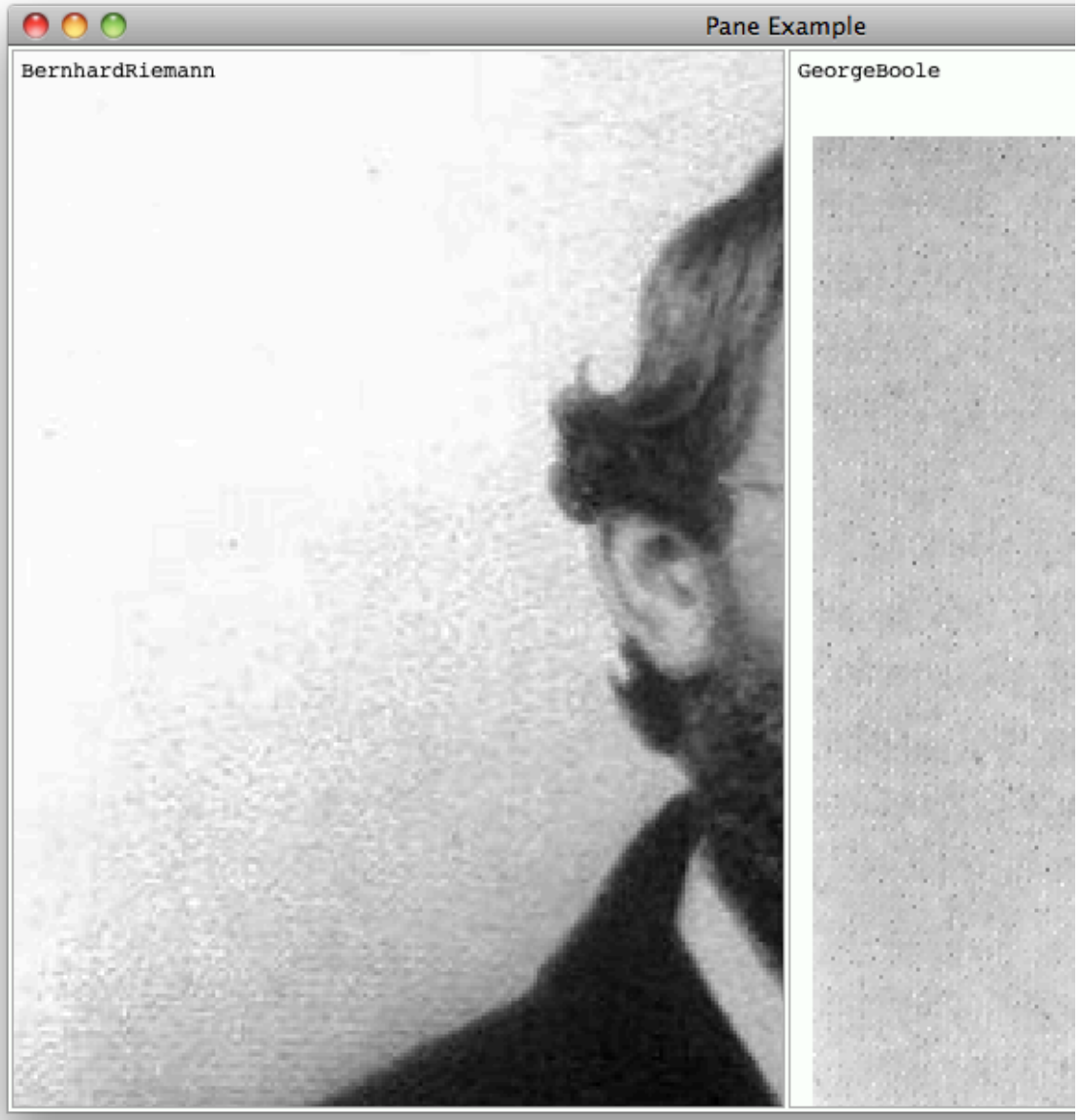
こちらの場合、この様に書くと、 2倍になるのだが、 1/2倍にすることが出来ない！

```
displayOn: graphicsContext
    "自分自身を表示する。"

    | aComposedText aPoint |
    self model picture
        ifNotNil:
            [:anImage |
            (anImage magnifiedBy: 2 @ 2) displayOn: graphicsContext
                at: self bounds center - anImage bounds center].
    self model label
        ifNotNil:
            [:aString |
            aComposedText := aString asComposedText.
            aPoint := 4 @ 0.
            graphicsContext paint: ColorValue white.
            (-1 to: 1)
                do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
            graphicsContext paint: ColorValue black.
            aComposedText displayOn: graphicsContext at: aPoint]
```

reciprocal をつければ、直感的な値に出来る

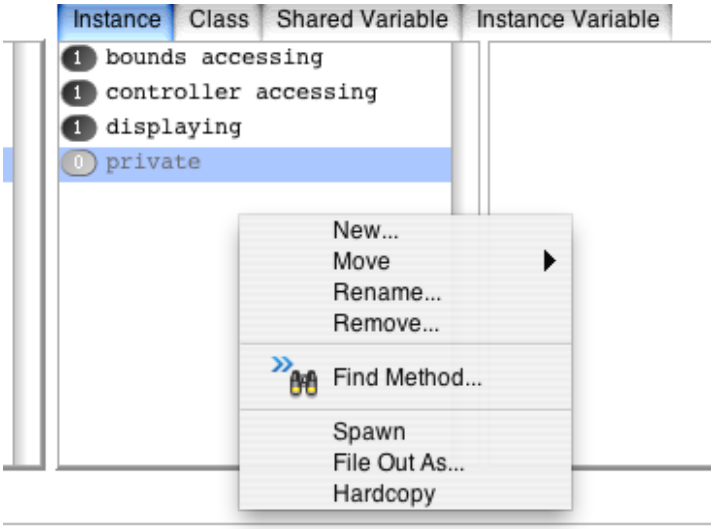
```
displayOn: graphicsContext
    "自分自身を表示する。"

    | aComposedText aPoint |
    self model picture
        ifNotNil:
            [:anImage |
            (anImage shrunkenBy: (2 @ 2) reciprocal) displayOn: graphicsContext "<-- reciprocal とは逆数の
こと"
                at: self bounds center - anImage bounds center].
    self model label
        ifNotNil:
            [:aString |
            aComposedText := aString asComposedText.
            aPoint := 4 @ 0.
            graphicsContext paint: ColorValue white.
            (-1 to: 1)
                do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
            graphicsContext paint: ColorValue black.
            aComposedText displayOn: graphicsContext at: aPoint]
```


Pane Example
BernhardRiemann
GeorgeBoole



Jun を用いて、同様のこと出来るし速いが、 QuickTime が必要
ちなみに、アスペクト比を異なったものにすることも出来る

private カテゴリを追加



拡大縮小の比率を算出する物を追加

private に scallingFactor を追加
アスペクト比を小さい方に合わせて縮小
青字部分を抜くと、アスペクト比が保たれなくなり、画面に張り付いて動くようになる
scallingFactor

```
| viewBounds imageBounds xScallingFactor yScallingFactor |
viewBounds := self bounds.
imageBounds := self model picture bounds.
```

```
        xScallingFactor := viewBounds width / imageBounds width.
        yScallingFactor := viewBounds height / imageBounds height.
        xScallingFactor := xScallingFactor min: yScallingFactor.
        yScallingFactor := xScallingFactor.
        ^xScallingFactor @ yScallingFactor
```

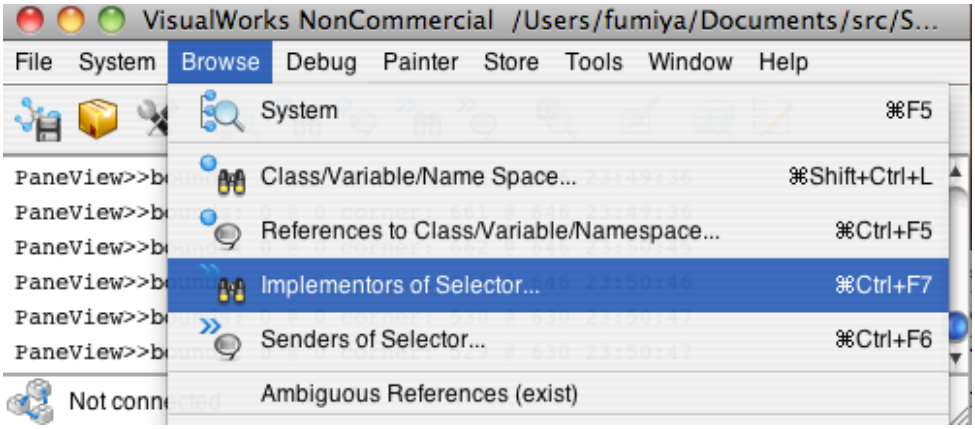
displaying の displayOn も修正

```
displayOn: graphicsContext
        "自分自身を表示する。"

        | aComposedText aPoint |
        self model picture
                ifNotNil:
                        [:anImage |
                        (anImage shrunkenBy: self scallingFactor reciprocal) displayOn: graphicsContext
                                at: 0 @ 0]. "<-- この部分があるので、左上整列になっているのである"
        self model label
                ifNotNil:
                        [:aString |
                        aComposedText := aString asComposedText.
                        aPoint := 4 @ 0.
                        graphicsContext paint: ColorValue white.
                        (-1 to: 1)
                                do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
                        graphicsContext paint: ColorValue black.
                        aComposedText displayOn: graphicsContext at: aPoint]
```

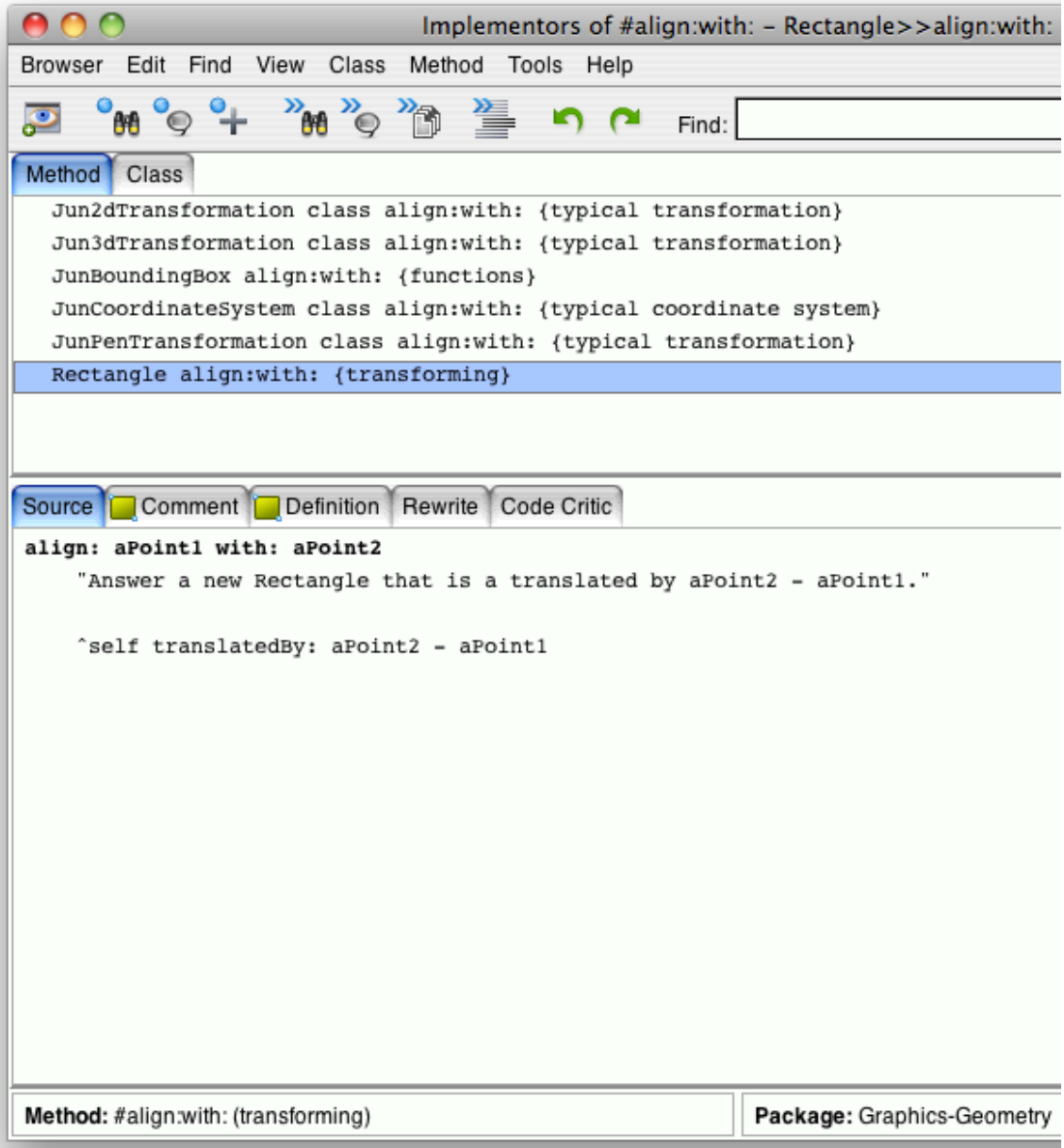
少し修正

```
displayOn: graphicsContext
        "自分自身を表示する。"

        | aComposedText aPoint |
        self model picture
                ifNotNil:
                        [:anImage |
                        | scaledImage |
                        scaledImage := anImage shrunkenBy: self scallingFactor reciprocal.
                        scaledImage displayOn: graphicsContext at: 0 @ 0].
        self model label
                ifNotNil:
                        [:aString |
                        aComposedText := aString asComposedText.
                        aPoint := 4 @ 0.
                        graphicsContext paint: ColorValue white.
                        (-1 to: 1)
                                do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
                        graphicsContext paint: ColorValue black.
                        aComposedText displayOn: graphicsContext at: aPoint]
```

今は左上揃えで固定的に書いているけれども、起点を変更できるようにする
とりあえず、どのようなメッセージを使うかだけ確認してみる




Select
Browse implementors of selector:
align:
Known Selectors:
align:on:
align:with:
OK
Cancel



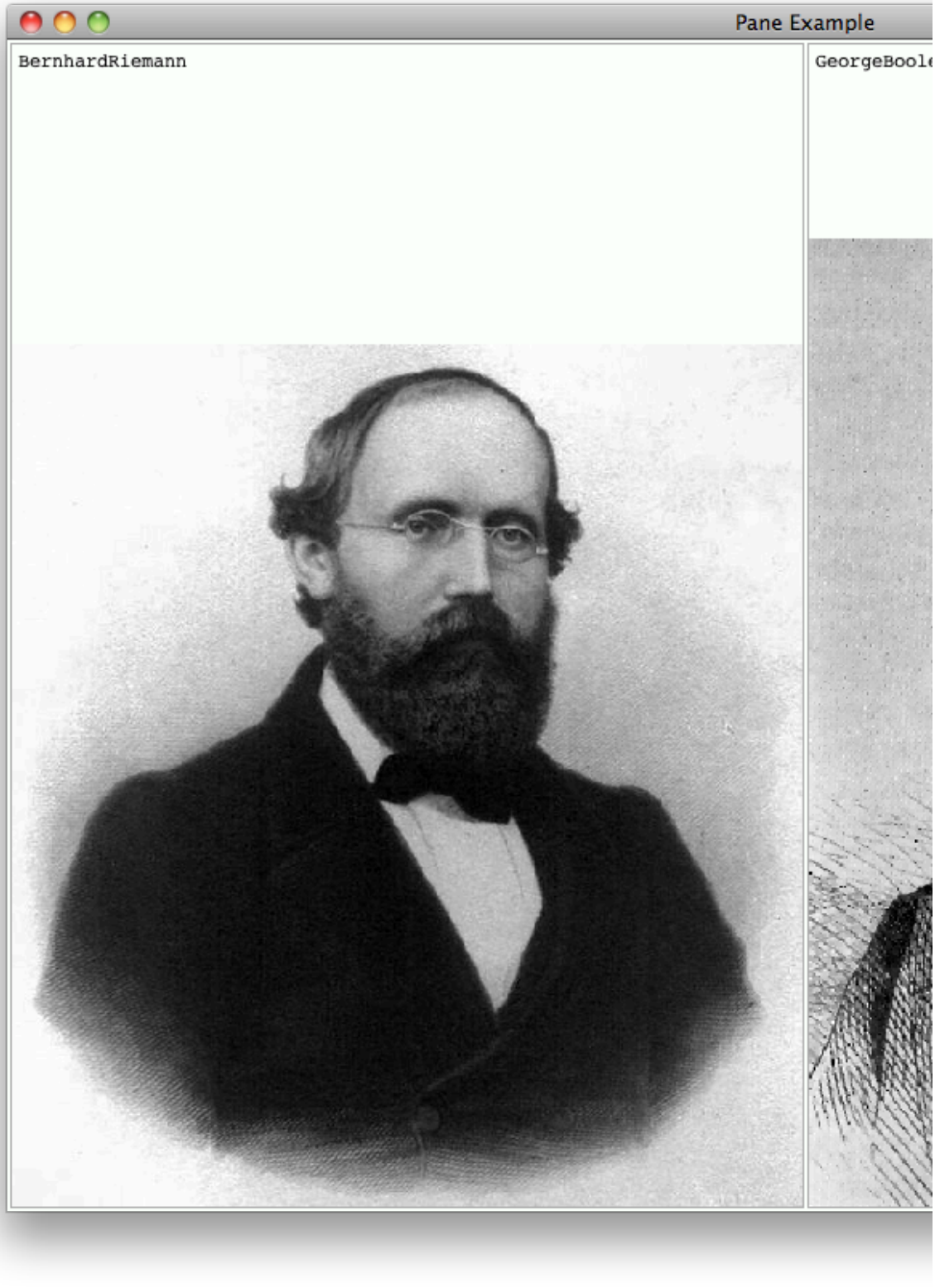
これを使おう

試しに右下揃えで描く

```
displayOn: graphicsContext
	"自分自身を表示する。"

	| aComposedText aPoint aRectangle |
	self model picture
		ifNotNil:
			[:anImage |
			| scaledImage |
			scaledImage := anImage shrunkenBy: self scallingFactor reciprocal.
			aRectangle := scaledImage bounds.
			aRectangle := aRectangle align: aRectangle bottomRight with: self bounds bottomRight. "<-- 右下の座標を手に入れる"
			scaledImage displayOn: graphicsContext at: aRectangle origin]. "<-- origin とは、左上のこと"
```

```
self model label
    ifNotNil:
        [:aString |
        aComposedText := aString asComposedText.
        aPoint := 4 @ 0.
        graphicsContext paint: ColorValue white.
        (-1 to: 1)
            do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
        graphicsContext paint: ColorValue black.
        aComposedText displayOn: graphicsContext at: aPoint]
```

つまり、この処理は、ウインドウの右下の位置を取得して、画像の幅分の Rectangle を取っている



perform を使って書き換え
青色の部分を同様に変更しなければならないので、変更箇所を一箇所にする

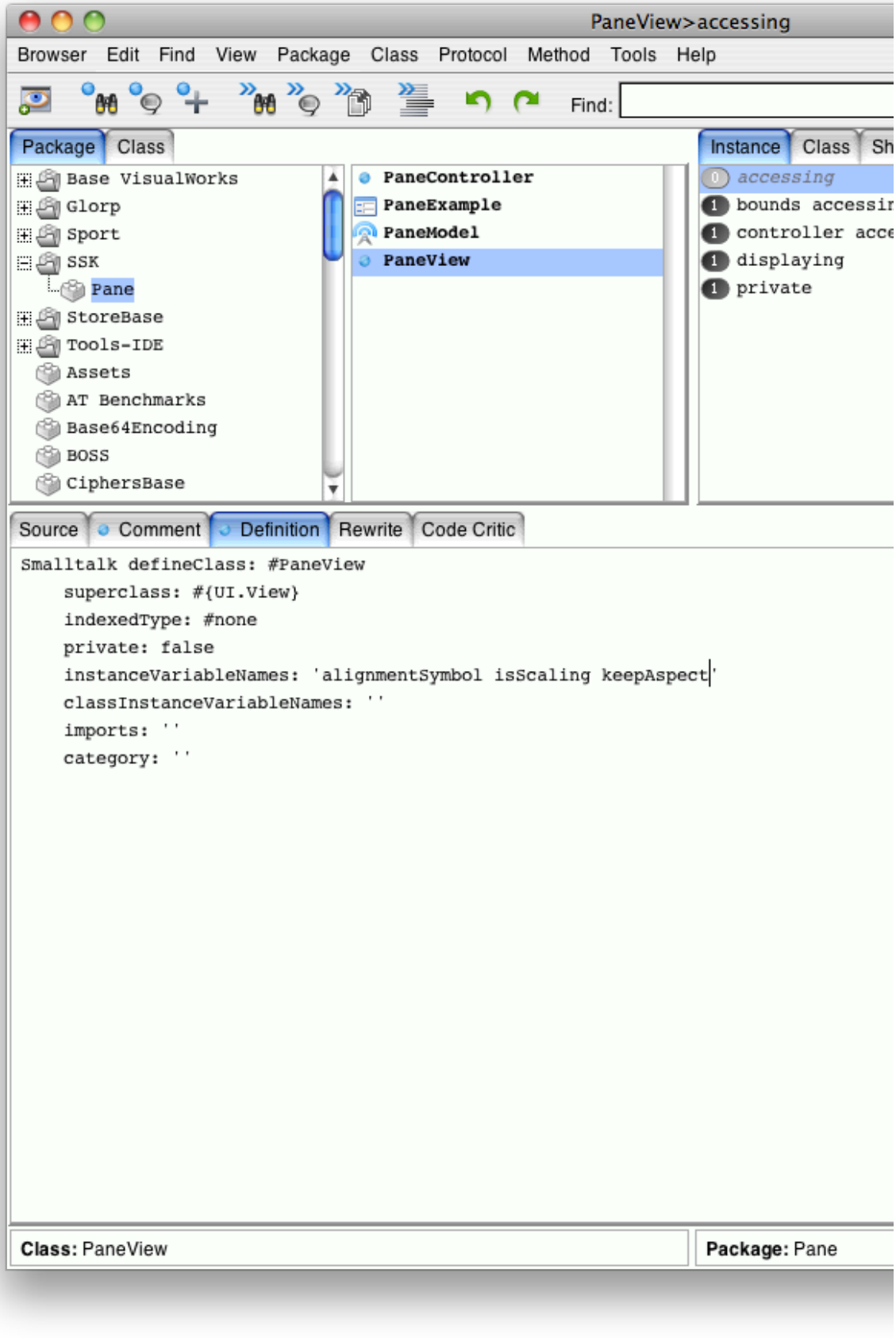
```
displayOn: graphicsContext
	"自分自身を表示する。"

	| aComposedText aPoint aRectangle messageSelector |
	self model picture
```

```
        ifNotNil:
            [:anImage |
            | scaledImage |
            messageSelector := #bottomRight. "<-- #center と書けば、中央揃えになる"
            scaledImage := anImage shrunkenBy: self scallingFactor reciprocal.
            aRectangle := scaledImage bounds.
            aRectangle := aRectangle align: (aRectangle perform: messageSelector)
                                with: (self bounds perform: messageSelector).
            scaledImage displayOn: graphicsContext at: aRectangle origin].
    self model label
        ifNotNil:
            [:aString |
            aComposedText := aString asComposedText.
            aPoint := 4 @ 0.
            graphicsContext paint: ColorValue white.
            (-1 to: 1)
                do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
            graphicsContext paint: ColorValue black.
            aComposedText displayOn: graphicsContext at: aPoint]
```

今はハードコーディングしているが、 messageSelector やアスペクト維持の scallingFactor などなど後々変更したい部分がある

それを実現するために、 accessing を追加

Definition を設定 (Source タブではなく、 Definition タブを選ぶように)



accessing はゲッターセッター群のことである

accessing に alignmentSymbol を追加

alignmentSymbol

```
	alignmentSymbol ifNil: [alignmentSymbol := #topLeft].
	^alignmentSymbol
```

alignmentSymbol: aSymbol

```
	(#(#topLeft #topCenter #topRight #leftCenter #center #rightCenter #bottomLeft #bottomCenter
#bottomRight #origin) "<-- これらしか受け付けないというアサーション(表明)"
		includes: aSymbol) ifFalse: [^nil].
	alignmentSymbol := aSymbol
```

isScaling

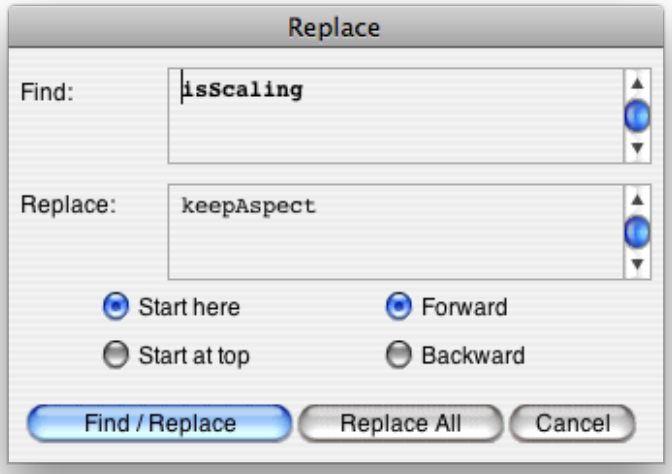
```
	isScaling ifNil: [isScaling := true].
	^isScaling
```

isScaling: aBoolean

```
	isScaling := aBoolean = true "<-- Boolean しか入らないようにする"
```

この様にして、 Boolean 鹿はいらないようにするのはソフトウェア工学的には正しいお作法かどうかはちょっと悩みどころ

isScaling を開いた状態で isScaling を keepAspect と書き換えた後
ctrl + r で Replace 画面が表示される
(勝手に補完される)



これを用いて、 isScaling から keepAspect を生成
keepAspect

```
	keepAspect ifNil: [keepAspect := true].
	^keepAspect
```

keepAspect: aBoolean

```
	keepAspect := aBoolean = true
```

アスペクトを維持するかどうか任意に変更できるようにしておく
scallingFactor

```
	| viewBounds imageBounds xScallingFactor yScallingFactor |
	viewBounds := self bounds.
	imageBounds := self model picture bounds.
```

```
	xScallingFactor := viewBounds width / imageBounds width.
	yScallingFactor := viewBounds height / imageBounds height.
	self keepAspect
		ifTrue:
			[xScallingFactor := xScallingFactor min: yScallingFactor.
			yScallingFactor := xScallingFactor].
	^xScallingFactor @ yScallingFactor
```

こちらは、何処に揃えるか、拡大するか否かを変更できるようにしておく

```
displayOn: graphicsContext
	"自分自身を表示する。"

	| aComposedText aPoint aRectangle messageSelector |
	self model picture
		ifNotNil:
			[:anImage |
			| scaledImage |
			messageSelector := self alignmentSymbol.
			scaledImage := self isScaling
					ifTrue: [anImage shrunkenBy: self scallingFactor reciprocal]
					ifFalse: [anImage yourself].
			aRectangle := scaledImage bounds.
			aRectangle := aRectangle align: (aRectangle perform: messageSelector)
					with: (self bounds perform: messageSelector).
			scaledImage displayOn: graphicsContext at: aRectangle origin].
	self model label
		ifNotNil:
			[:aString |
			aComposedText := aString asComposedText.
			aPoint := 4 @ 0.
			graphicsContext paint: ColorValue white.
			(-1 to: 1)
				do: [:y | (-1 to: 1) do: [:x | aComposedText displayOn: graphicsContext at: x @ y + aPoint]].
			graphicsContext paint: ColorValue black.
			aComposedText displayOn: graphicsContext at: aPoint]
```

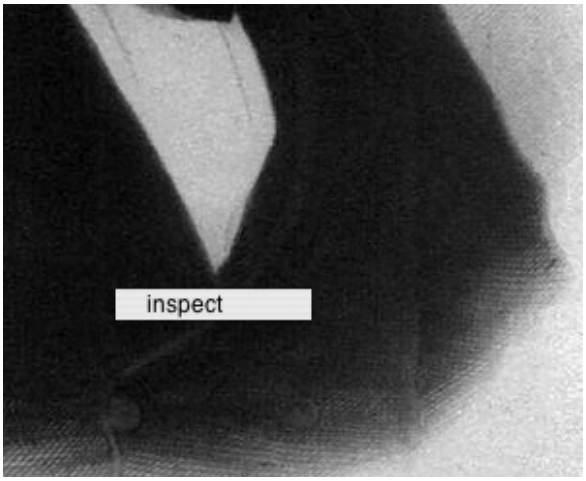
PaneController を変更して、実際にアスペクト維持などを任意に変更できるようになっているのか確認してみる

events の yellowButtonPress を変更

とりえあず、動くかどうかを確認

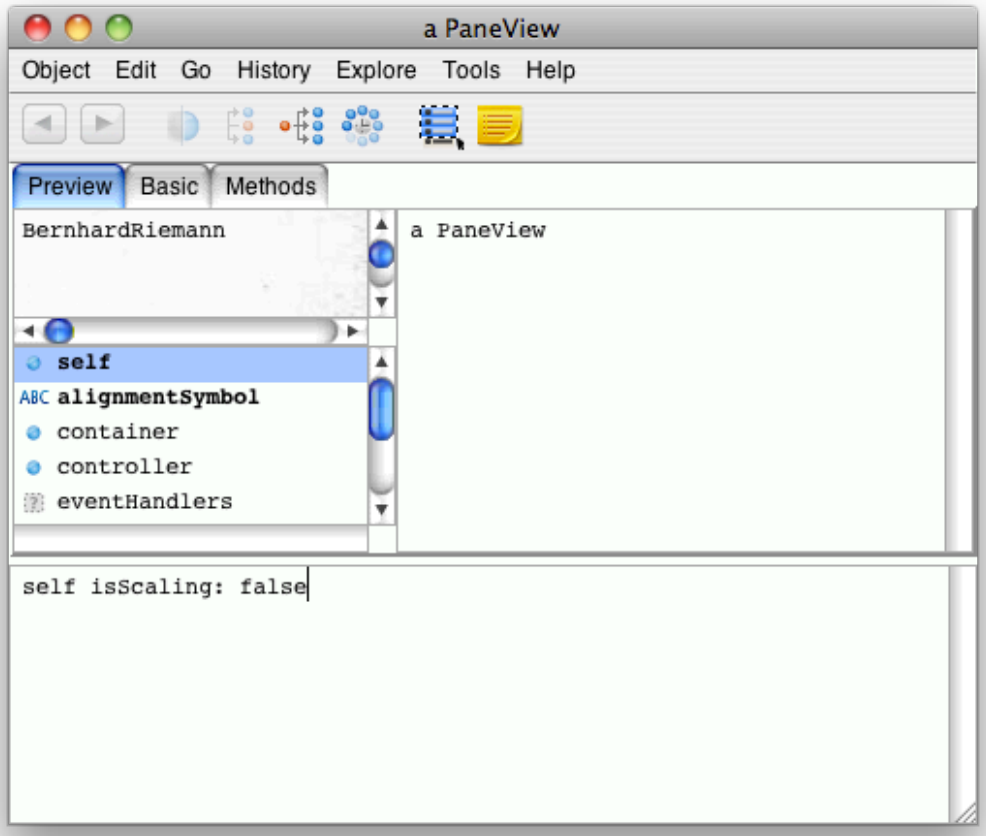
```
yellowButtonPressedEvent: event
	"マウスの右ボタンが押されたときの処理をする。"

	| aMenu aSymbol |
	aMenu := Menu labelArray: #('inspect') values: #(#inspect).
	aSymbol := aMenu startUp.
	(aSymbol isKindOf: Symbol)
		ifTrue:
			[Transcript
				cr;
				show: thisContext printString;
				space;
				show: Time now printString].
	^nil
```

実際に右クリックをして inspect をすると、 inspector が開くようにする



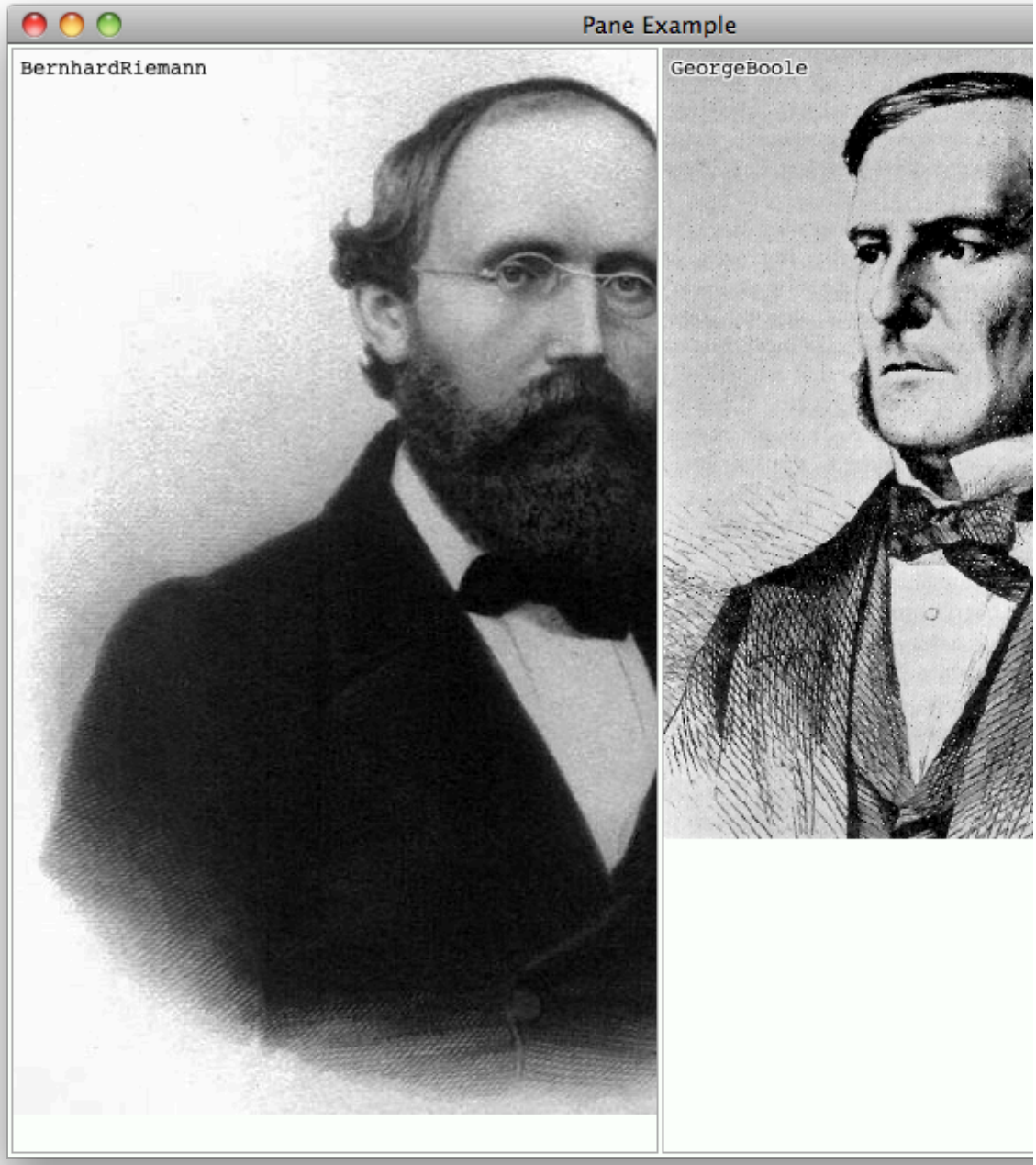

```
yellowButtonPressedEvent: event
	"マウスの右ボタンが押されたときの処理をする。"

	| aMenu aSymbol |
	aMenu := Menu labelArray: #('inspect') values: #(#inspect).
	aSymbol := aMenu startUp.
	(aSymbol isKindOf: Symbol) ifTrue: [self view perform: aSymbol].
	^nil
```

例えば、状態を変更してみる(右上のワークスペースを開いて)

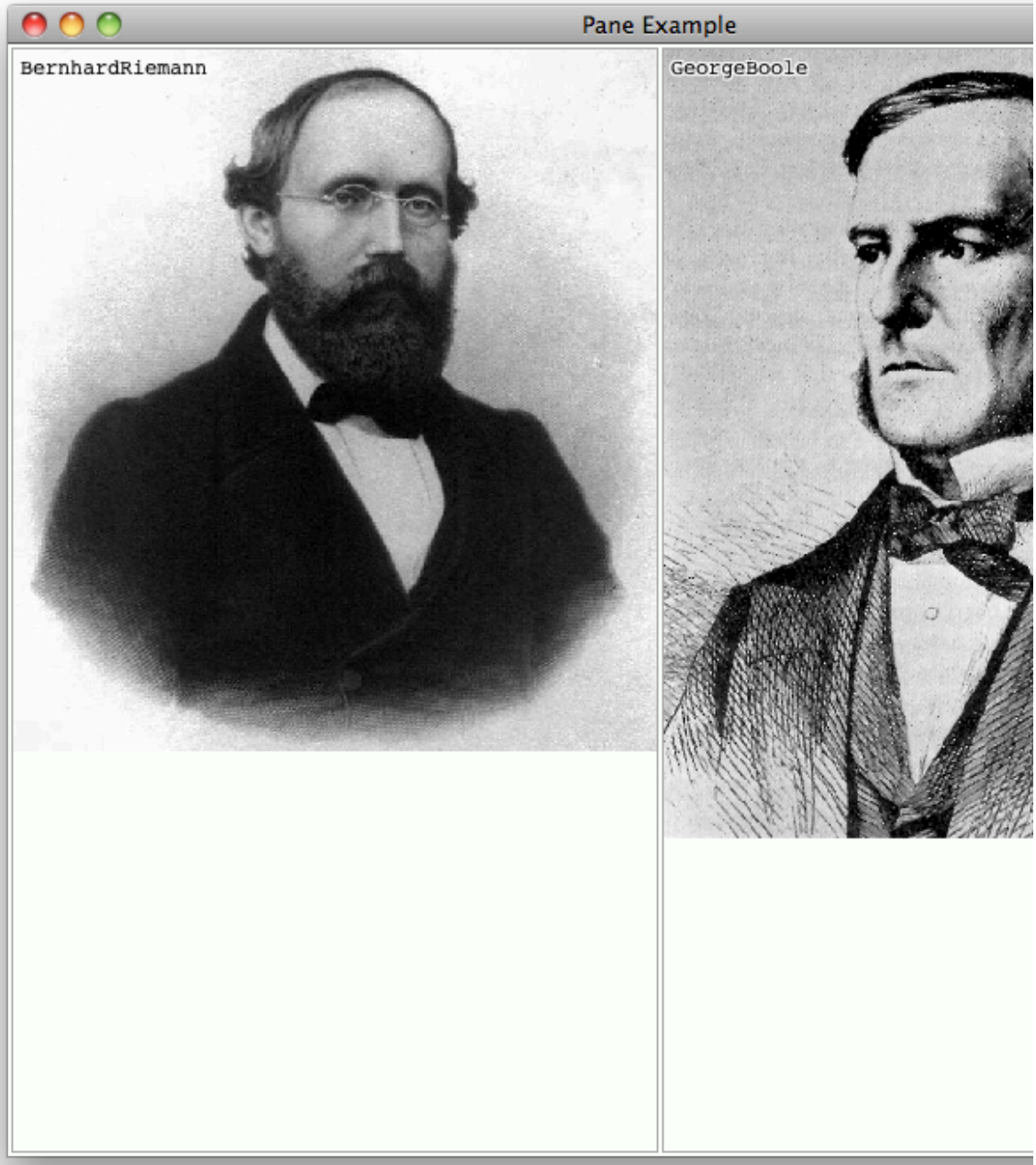



```
self isScaling: false.
self model changed.
```

と入力して、Do it して画面サイズを変更してみる



true にすると



この様になるので、この機能を追加していこう…というのが次回以降の話(たぶん)

質問
accessing に initialize を用意して最初に初期化しちゃダメ…？
昔は、メモリをけちるために、ゲッターが初めて呼ばれる時に初期値を設定していたが、最近はメモリが潤沢にあるので、最初に初期化した方がよいかな。
ifNil: はインライン展開されているので、実はオーバーヘッドは小さい。
使い分けは、時と場合による