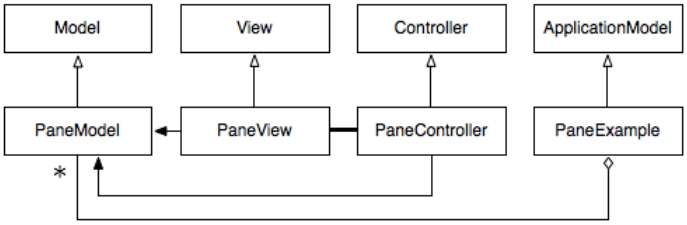
■ 2011/5/11

酒匂寛さん来場。青木さんとは SRA でもコロラドでも一緒だったそうで。多く訳書があると。
amazon.co.jp で引いたら 27件ありました。

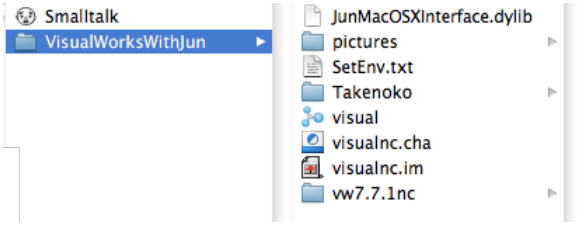
今回は日本語版の配布も行われた。Windows 版のみ。どうやら Shift JIS のエンコーダが入ったもの、の模様。

□ つづき

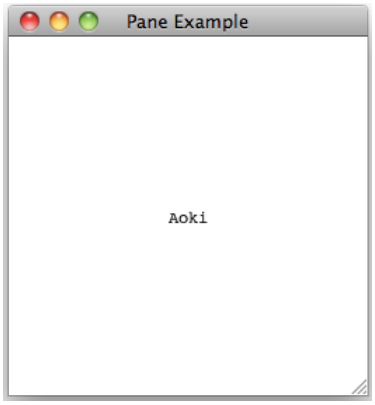
ここから始める。



復習を兼ねてやるか。
配布した pictures フォルダを VisualWorks フォルダに入れておく。



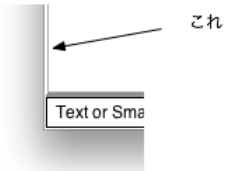
ファイルブラウザで配布された SSK_PaneMVC_20110406.st ファイルを file in して動作確認。
前回は以下のウィンドウを出して中央に文字を書くところまで。



さて続き。今回の目標はこれに枠を付ける。

□ 枠を付ける

前回出した画面はウィンドウにそのまま View を貼り付けていたので、見た目上裸というか何も無い。
が、普通に Smalltalk のウィンドウなどを見ているとそれなりに枠が付いていたりする。



コードで言うとこのあたり。

example1
"例題プログラム：窓MVCを作り、ウィンドウに乗せて開く。"
"PaneExample example1."

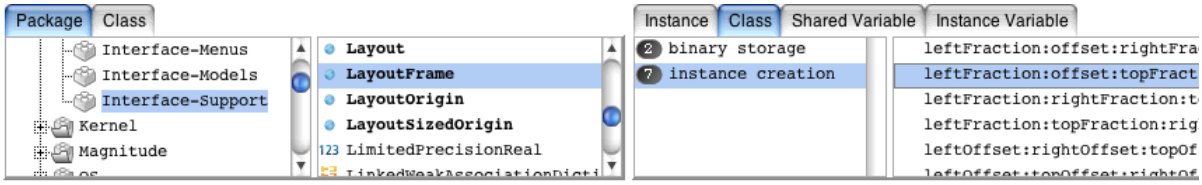
```
| aModel aView aWindow |
aModel := PaneModel new.
aView := PaneView model: aModel.   <<< ※1 Model を View につなぐ
aWindow := ScheduledWindow new.   <<< ※2 Window を用意する
aWindow label: 'Pane Example'.
aWindow component: aView.   <<< ここで直接 View を Window に乗せている。
aWindow openWithExtent: 256 @ 256.
^aWindow
```

ウィンドウに直接 view を載せているのでどうにも飾り気のないものになっている。
これを（他の物を間に挟んで）階層的に積むことによって他のウィンドウのように枠線を入れることを試みる。
具体的には上記コードの※1と※2の間で挟み込むものを入れる。

```
aView := PaneView model: aModel.
aWrapper := BorderedWrapper on: aView in: (aLayout) border: aBorder. <<< 中間的役割として Wrapper なるものを利用
aWindow := ScheduledWindow new.
aWindow component: aWrapper. <<< それを使って Window に乗せる（これは後に CompositePart なるものを使うことになる）
```

のようにセットするとして、この aLayout と aBorder を作っていくことにする。
この Wrapper は、指定した View を、指定したレイアウトと指定した枠線（ボーダー）で囲んでくれるもの。

aBorder 部分は SimpleBorder new でやる。お仕着せのものの一つ。（他にもあると）
aLayout の部分はオリジナル（LayoutFrame クラスのメソッド）を見て取ってくる事にする。書くとちょっと大変。



ここ。長いメソッド名だ。このソースの先頭部分（引数一覧）は以下の通り。
leftFraction: left offset: leftOffset topFraction: top offset: topOffset
rightFraction: right offset: rightOffset bottomFraction: bottom offset: bottomOffset

これを元の
aWrapper := BorderedWrapper on: aView in: (aLayout) border: aBorder.
の aLayout 部分にペーストして format すると以下の通り。（長い！）

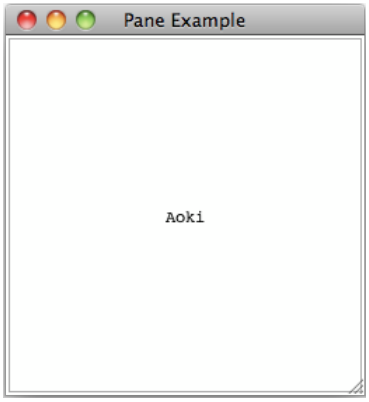
```
aWrapper := BorderedWrapper
                on: aView
                in: (LayoutFrame
                        leftFraction: left
                        offset: leftOffset
                        topFraction: top
                        offset: topOffset
                        rightFraction: right
                        offset: rightOffset
                        bottomFraction: bottom
                        offset: bottomOffset)
                border: SimpleBorder new.
```

この引数部分に値を直接書き込んでテストランしよう。具体的には以下の通り。
CompositePart なるものが登場しているのに注意。

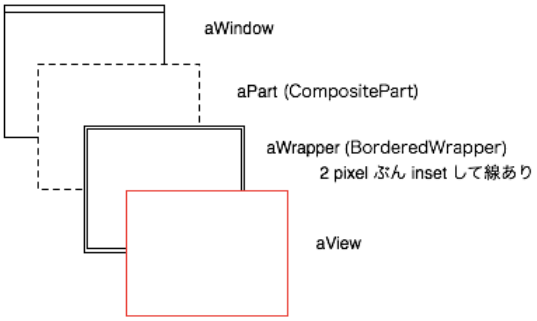
```
aView := PaneView model: aModel.
aWrapper := BorderedWrapper
                on: aView
                in: (LayoutFrame
                        leftFraction: 0
                        offset: 2
                        topFraction: 0
                        offset: 2
                        rightFraction: 1
                        offset: -2
                        bottomFraction: 1
                        offset: -2)
                border: SimpleBorder new.
aPart := CompositePart new.   <<< このタイミングで登場。
aPart addWrapper: aWrapper.
aWindow := ScheduledWindow new.
aWindow label: 'Pane Example'.
```

```
aWindow component: aPart.
aWindow openWithExtent: 256 @ 256.
^aWindow
```

で、accept して（ aPart など一時変数の追加あり）実行すると以下のようにボーダー付きで 2 pixel inset して枠が出た。よしよし。

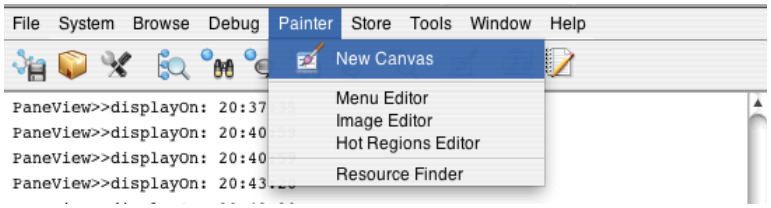


試みに Fraction と Offset の値をいじってどこに反映されるか見ておくと次にやることが分かりやすい。
構造的には以下のようになっている。

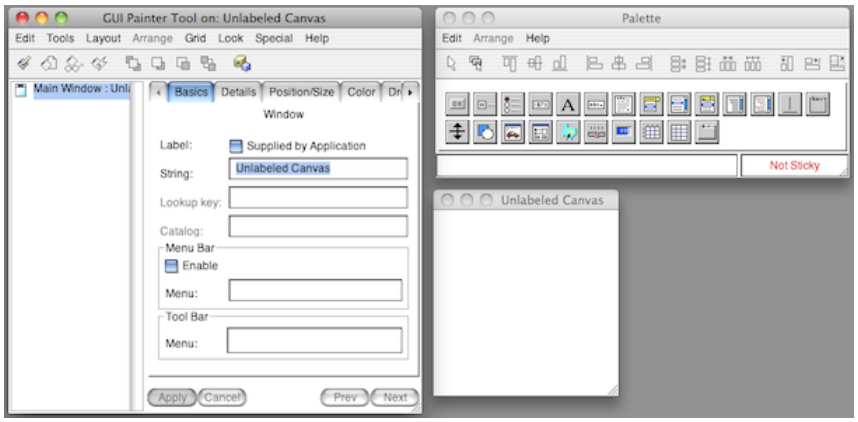


□ インタラクティブにツールを使って作る

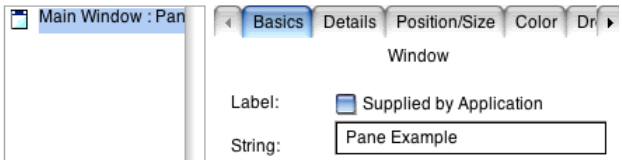
先の作業を UI 作成ツールを使ってやり直す。



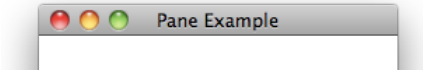
Painter メニューの New Canvas を実行。こんなのが出る。（正式名称は GUI Painter なのか？？）



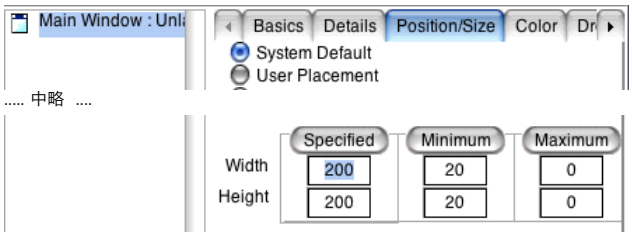
この左側ウィンドウでまず名前を Pane Example などとしておく。



すると右上ウィンドウ（サンプル表示、というか、これが実物）の名前が Pane Example になる。



まずこのウィンドウサイズを変更する。Position/Size タブの Width, Height を変更。

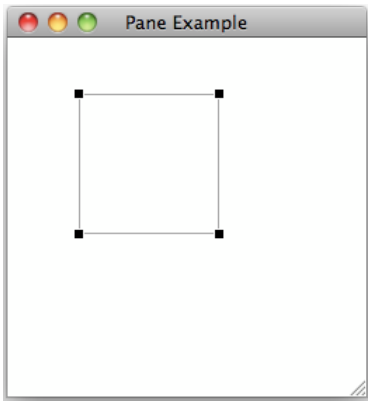


..... 中略 ....

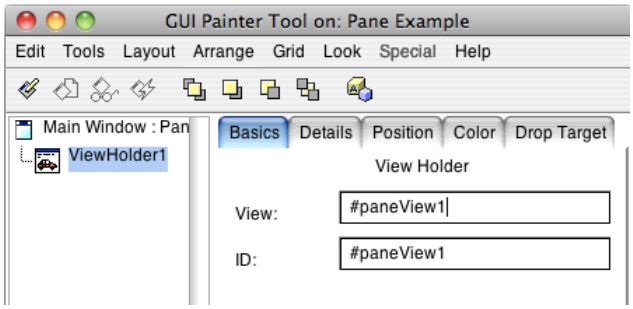
デフォルトは 200x200 になっているのでこれを 250x250 にする。Apply すると実物ウィンドウが実際にサイズ変更されることを確認。

次に枠付きのラッパーを設置する。右側のパレットウィンドウ（以下）から、自動車アイコン（これ ViewHolder という名前）をクリックする。

これが実は先の wrapper に相当する。これを選んでから、Pane Example ウィンドウ上でクリックすれば設置することができる。位置はいい加減。



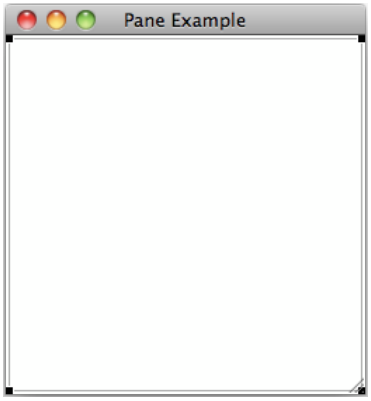
これで GUI Painter には ViewHolder が一つ出来ているはず。（下図）
これの名称を #paneView1 としよう。



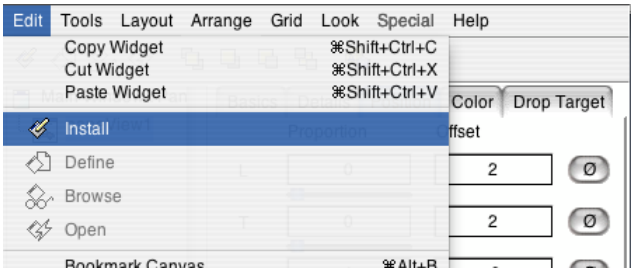
（安田はここの二つの名前のそれぞれの結びつきが良く分からなかった。二つの #paneView1 には意味の違いがありそうだが、、）

次にこの ViewHolder の position について設定する。

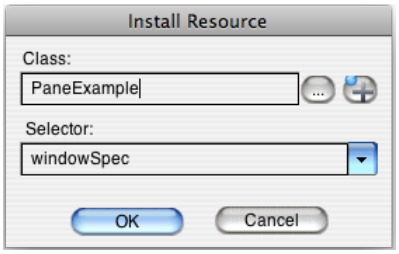
これでApply すると、実物ウィンドウの表示がこうなる。



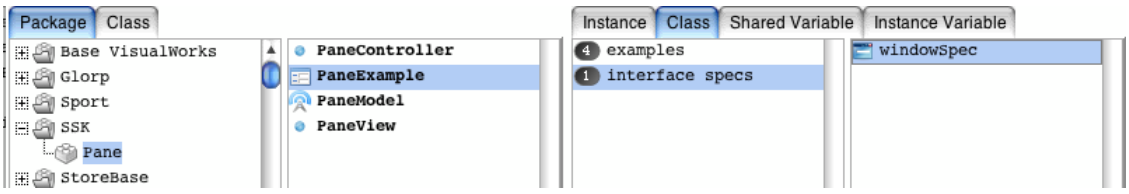
つまり 2 ドット inset された位置に枠付きで View が配置された。この状態で作ったパーツを Install する。



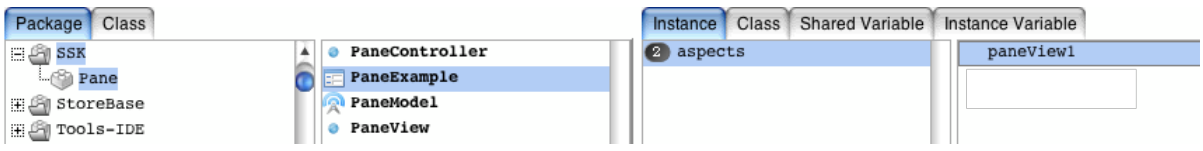
この時インストール先を尋ねられるので、PaneExample を指定してやる。セレクタ名を書き換えたいが、まあおいておこう。



これでInstall。PaneExample をもう一度クリックすると（画面再描画させると）増えているのが確認できる。



さて instance 側に移って、メッセージを一つ（名前を paneView1 とする）を作る。カテゴリ（プロトコル？）名は aspects がいい。



paneView1 メッセージの内容は以下のとおり。

paneView1
```
^PaneView new
```

さてこれで UI 設計ツールを使ってウィンドウを開けるためのメソッドを完成させる。名前は example22 とした。

example22
```
"例題プログラム：窓MVCを作り、ウィンドウに乗せて開く。"
"PaneExample example22."
| anExample |
anExample := PaneExample new.
anExample open
```

（むむ。これ、PaneView との結びつきはどこだ？open すると、windowSpec に書いてあった paneView1 （メソッド）を呼び、そこに書いてある PaneView new が呼ばれ、そこを通して Aoki とか書かれてあった PaneView が作られて、上に乗っけられる、のか。）

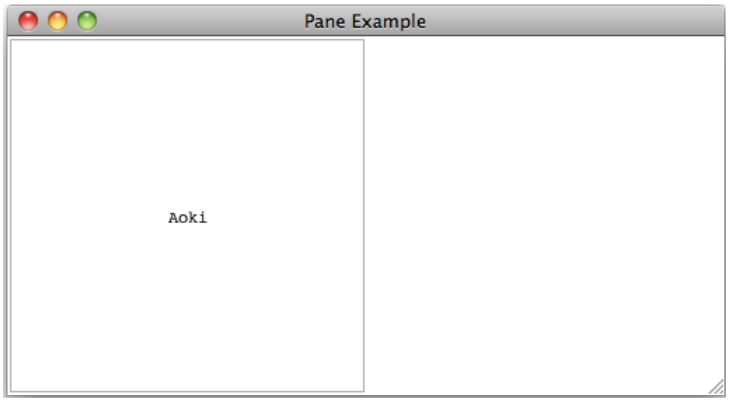
これで実行すれば正しく枠付きのウィンドウが出る事が確認できる。

□ さて複数の Pane を View に載せよう

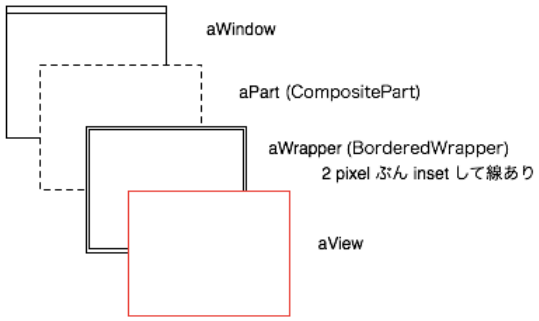
example21 （手動記述）版を 31 に変名して、ここからスタートする。
まず横に二つ並べようと思うので、まず全体のサイズを 256 x 256 から 512 x 256 にする。

```
aWindow openWithExtent: 512 @ 256.
```

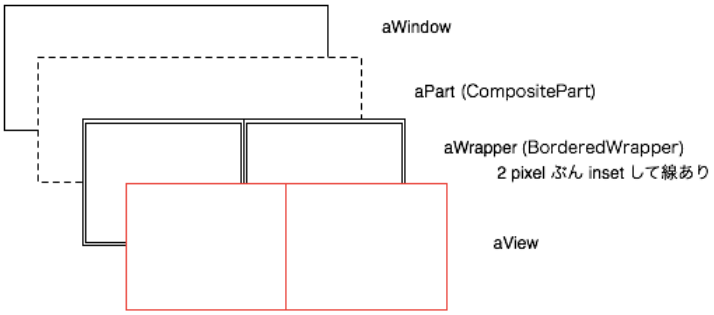
で、もう一つ、画面左側だけに片寄せて出そう。
```
                in: (LayoutFrame
                        leftFraction: 0
                        offset: 2
                        topFraction: 0
                        offset: 2
                        rightFraction: 0.5 <<< これこれ
                        offset: -1 <<<< ここも
                        bottomFraction: 1
                        offset: -2)
```
実行すると



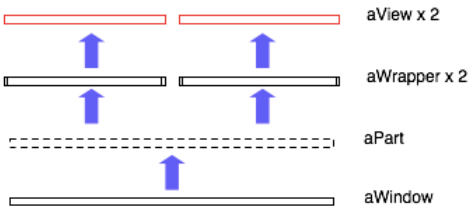
次、これを 6 枚出すために、、、6 つこれをハードコードするのは嫌なのでプログラムで合成する。（ための準備をする。まずは二枚でね。）
さて構造復習。



これを以下のような構造にする。



階層構造としては以下のようになっている。



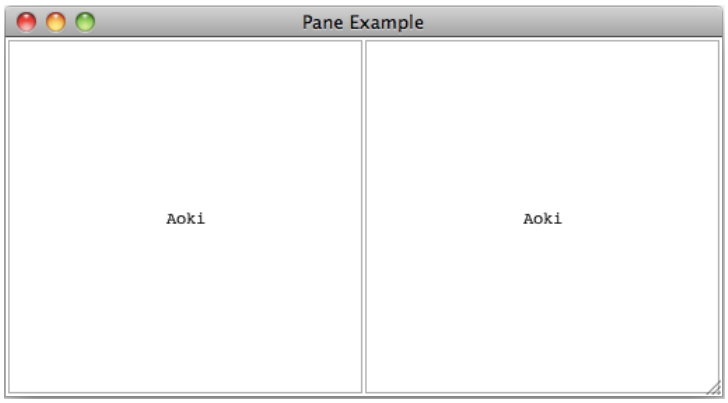
コードの方針としては、
・配列を用意して、
・そこに Layout 情報をあらかじめ揃えておき、
・配列を順繰りに回りながら Layout の要素を取り出し、
・それに合わせて（先に一つだけ用意しておいた）aPart に（View とセットで）乗せていき、
・最後に aPart を aWindow に乗せる
といったもの。

実際に出来上がったコード。（example31）

```
| aModel aView aWindow aWrapper aPart aCollection |
aPart := CompositePart new.    <<< あらかじめ aPart を一つ用意しておく
aCollection := OrderedCollection new.  <<< 配列を用意
aCollection add: (LayoutFrame  <<< 配列に第一要素を入れてみる（ハードコードだけどさ！）
            leftFraction: 0
            offset: 2
            topFraction: 0
            offset: 2
            rightFraction: 0.5
            offset: -1
            bottomFraction: 1
            offset: -2).
aCollection add: (LayoutFrame  <<< 配列に第二要素を入れてみる（ハードコードだけどさ！）
            leftFraction: 0.5
            offset: 1
            topFraction: 0
            offset: 2
            rightFraction: 1
            offset: -2
            bottomFraction: 1
            offset: -2).
aCollection do:   <<< 配列に合わせてループする
        [:aLayoutFrame |    <<< レイアウト要素を取りだし、
        aModel := PaneModel new.    <<< Model を独立に作り、
        aView := PaneView model: aModel.  <<< その（Model を抱えた）View を（やはり）独立に作り、
        aWrapper := BorderedWrapper  <<< 枠付きラッパーに View を（レイアウト指定して）乗せる
                on: aView
                in: aLayoutFrame
                border: SimpleBorder new.
        aPart addWrapper: aWrapper].  <<< 作ったラッパーを aPart に乗せる（※）
aWindow := ScheduledWindow new.
aWindow label: 'Pane Example'.
aWindow component: aPart.  <<< 作った aPart が window に乗った
aWindow openWithExtent: 512 @ 256.
^aWindow
```

※ちょっと注意すべきはレイアウト情報は aWrapper の時点で指定されている。
つまり aPart は（上の構造図とはちょっと違って）単にオブジェクトを束ねているだけで、特別「この上に乗せる」といったことをしているわけではない模様。（あるいは「この上に乗せる」といったことを意識している癖に、「ここに乗せる」ということは全く意識していない。）

出来上がりはこんな感じ。



さてこれが動いた後で、上の「二つ丸ごとハードコード」するのがちょっと嫌なので、これをパラメタ列を解釈するような形態に変える。
アイディアとしては単純で、以下の赤文字のところに注目されたい。

example31

```
"PaneExample example31"
| aModel aView aWindow aWrapper aPart aCollection |
aPart := CompositePart new.
aCollection := OrderedCollection new.
#(#(0 2 0 2 0.5 -1 1 -2) #(0.5 1 0 2 1 -2 1 -2)) do:    <<< ※1
        [:anArray |
        aCollection add: (LayoutFrame
                leftFraction: (anArray at: 1)
                offset: (anArray at: 2)
                topFraction: (anArray at: 3)
                offset: (anArray at: 4)
                rightFraction: (anArray at: 5)
                offset: (anArray at: 6)
                bottomFraction: (anArray at: 7)
                offset: (anArray at: 8))].
aCollection do:                                    <<< ※2
        [:aLayoutFrame |
        aModel := PaneModel new.
        aView := PaneView model: aModel.
        aWrapper := BorderedWrapper
                on: aView
                in: aLayoutFrame
                border: SimpleBorder new.
        aPart addWrapper: aWrapper].
aWindow := ScheduledWindow new.
aWindow label: 'Pane Example'.
aWindow component: aPart.
aWindow openWithExtent: 512 @ 256.
^aWindow
```

ところがよくよく見てみると、※1 と※2 のループは全く同じだけ回る。ということはこのループは単一化すればよい。
以下のように直した。これで example31 は完成。

```
| aModel aView aWindow aWrapper aPart aLayoutFrame |
aPart := CompositePart new.
#(#(0 2 0 2 0.5 -1 1 -2) #(0.5 1 0 2 1 -2 1 -2)) do:
        [:anArray |
        aLayoutFrame := LayoutFrame
                leftFraction: (anArray at: 1)
                offset: (anArray at: 2)
                topFraction: (anArray at: 3)
                offset: (anArray at: 4)
                rightFraction: (anArray at: 5)
                offset: (anArray at: 6)
                bottomFraction: (anArray at: 7)
                offset: (anArray at: 8).
        aModel := PaneModel new.
        aView := PaneView model: aModel.
        aWrapper := BorderedWrapper
                on: aView
                in: aLayoutFrame
```

```
                                    border: SimpleBorder new.
                aPart addWrapper: aWrapper].
    aWindow := ScheduledWindow new.
    aWindow label: 'Pane Example'.
    aWindow component: aPart.
    aWindow openWithExtent: 512 @ 256.
    ^aWindow
```

□ 次、example32。同じ事をツールで作った UI でやる。

example32

```
    "例題プログラム：窓MVCを作り、ウィンドウに乗せて開く。"
    "PaneExample example32."
    | anExample |
    anExample := PaneExample new.
    anExample openInterface: #windowSpec2
```

というように先に windowSpec2 で開く、ということだけ作っておく。あとは　windowSpec2 を作ればとりあえず良し。
（openInterface って以前出たことあったっけ？）

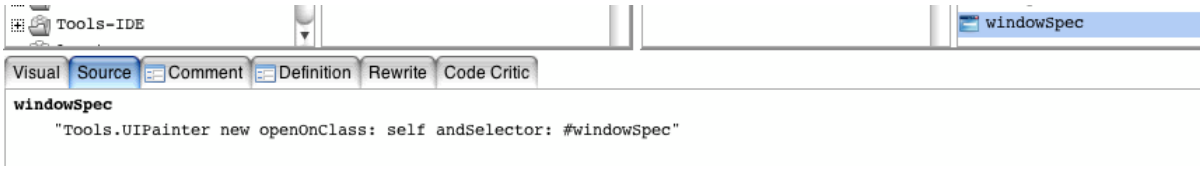
先に paneView1 メッセージを直しておこう。これが本来の姿らしく。（いまいちぴんと来ない）

paneView1

```
    | aModel aView |
    aModel := PaneModel new.
    aView := PaneView model: aModel.
    ^aView
```

後で要るので paneView2 も（同じ内容で）作っておこう。

さて windowSpec2 を作る、、んだけど面倒くさいので windowSpec1 のソースを開いて、windowSpec1 の名前を 2 に変えちゃえ。

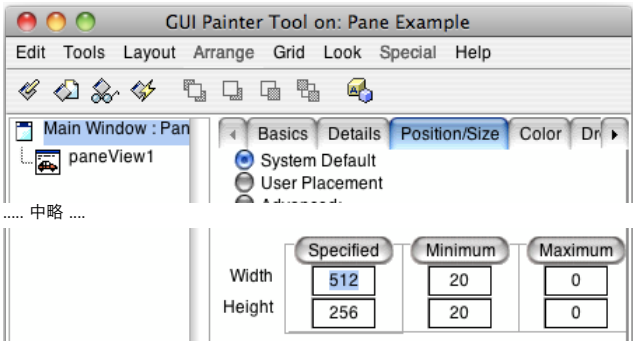


これを以下のように。

windowSpec2

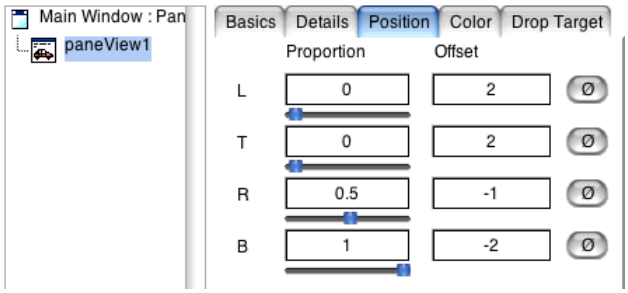
```
    "Tools.UIPainter new openOnClass: self andSelector: #windowSpec2"
```

ところでこの（上の）コメント部分を実行するとエディタが開いたりする。
今度は（Main Window の）幅を 512 にする。

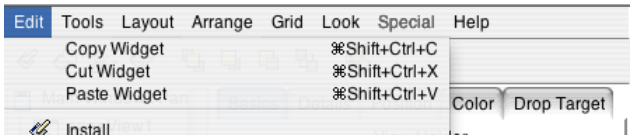


..... 中略 ....

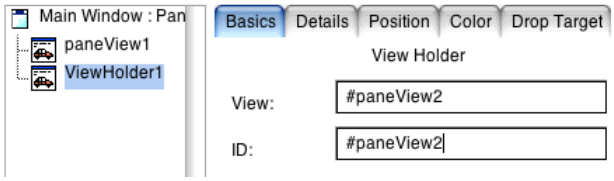
次に paneView1 の Position （Fraction に相当する）などを適切な値（左側に寄るように）修正。



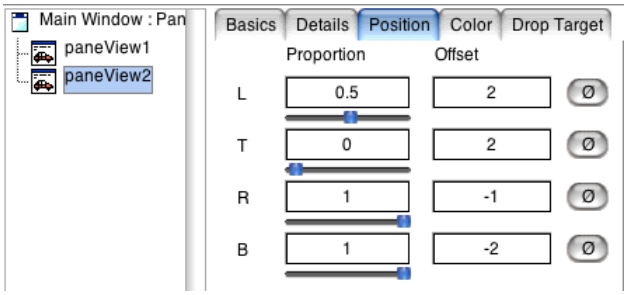
次、paneView1 をコピー＆ペーストで paneView2 を作る。Edit メニューに Copy / Paste がある。



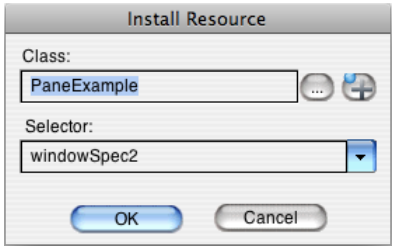
ペーストすると、viewHolder1 とかなんとかいう名前で新規作成されるので名前を paneView2 に変更。



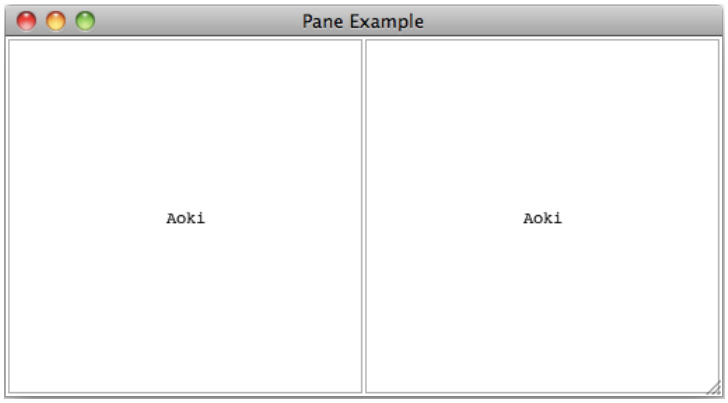
今度は右に寄るように position などを修正。



出来上がったら再びインストール。



これで実行すると、ちゃんと以下のように二つ並んで出る。



□ レイアウトをいじる

少し時間が残ったので、UI ツールで作ったオブジェクトに働きかけるケースを試す。
具体的にはウィンドウのサイズを動的に変更する。

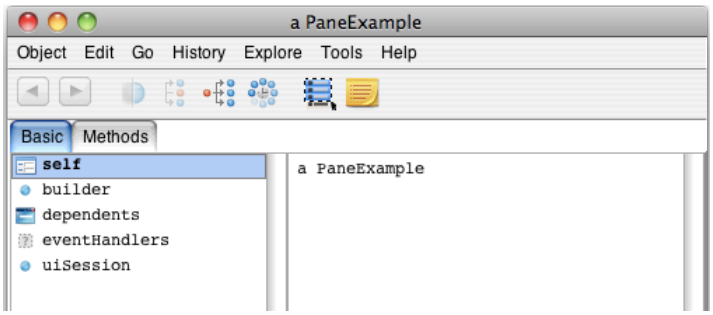
アプローチとしてはいろいろあるだろうが、常套手段としては作られたオブジェクトをそのまま Inspector で中身を直接見ることが行われる。
そのために example33 メッセージで生成されたを返すようにする。

example33

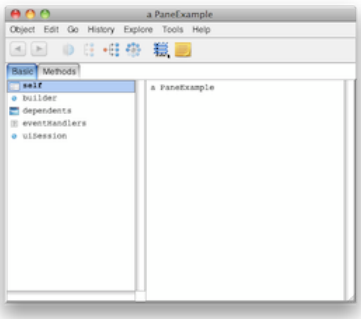
```
"PaneExample example33."
| anExample |
anExample := PaneExample new.
anExample openInterface: #windowSpec2.
```

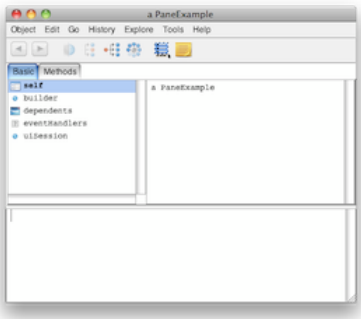
^anExample <<< 値を返すように修正

これを Inspect It して結果をつかまえる。



ここで  で Evaluation pane （ウィンドウ下半分）を出す。





この下半分の領域にメッセージ式を書いてテストできる。self builder window と書いて Print It すると以下のようになる。

self builder window a ScheduledWindow 4063968

builder てのはツールが用意している、裏でいろいろ spec を読んでオブジェクトを作ってくれる奴。
なので、そいつに言ってウィンドウを得ている。
上のように正しく Window オブジェクトが得られていることが分かったので、今度はそのサイズを得る。self builder window displayBox で Print It。

self builder window displayBox 36 @ 263 corner: 548 @ 519

なおこのときウィンドウを手でサイズ変更して Print It すると値が変わる。（あたりまえ）
では、と、ちょっと複雑になるが、この Rectangle を取り出して、幾らか伸ばして、また再設定すればウィンドウサイズが変わるだろう、と。
今度は以下のように書いて実行する。赤文字が Rectangle のサイズを上げているところ。

```
| aWindow aRectangle |
aWindow := self builder window.
aRectangle := aWindow displayBox.
aRectangle := aRectangle expandedBy:(0 @ 0 corner: 50 @ 50).
aWindow displayBox: aRectangle.   <<< 再設定
```

これで 50 x 50 伸びるようになるだろう。insetBy とかもあるんだよ、とのこと。

さて Inspector の中ではそれで良いが、これをコードの中に埋めてみる。青文字が上の Inspector 上のコードから取ってきたもの。

example33

```
    "PaneExample example33."
    | anExample aWindow aRectangle |
    anExample := PaneExample new.
    anExample openInterface: #windowSpec2.
    3 seconds wait. <<< 3 秒待ってみる（伸びたことが分かりやすいように）
    aWindow := anExample builder window.  <<< 元は self builder だったが、ここでは anExample だよね
    aRectangle := aWindow displayBox.
    aRectangle := aRectangle expandedBy: (0 @ 0 corner: 50 @ 50).
    aWindow displayBox: aRectangle.
    ^anExample
```

これで実行すれば、まず 512 x 256 で表示し、3 秒待つと伸びる、というようになる。