

USB メモリの中身は

ColorCMY.pdf
ColorCMY.png
ColorCMY.st
ColorHSB.pdf
ColorHSB.png
ColorHSB.st
ColorRGB.pdf
ColorRGB.png
ColorRGB.st
PreLoad.st

PreLoad.st の中身を確認しておく

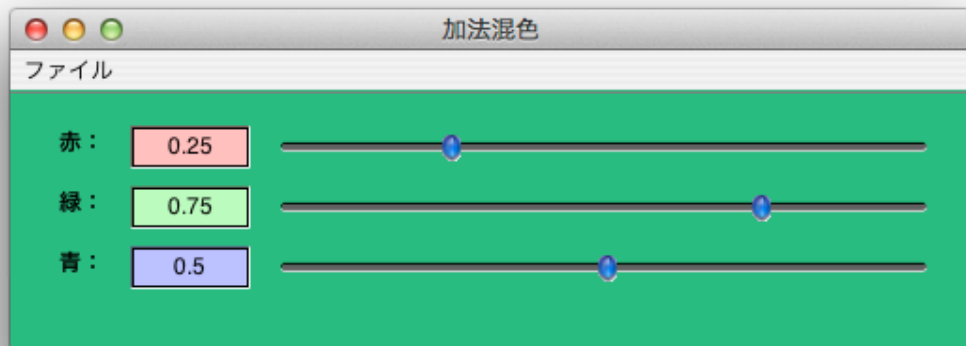
この中身は前回、手で作ったパッケージを自動で作ってくれるもの

ということで、PreLoad.st を File in すると KSU の中に KSU-Template が自動で作られる

この状態で、ColorRGB.st を File in する。

実際に作っていく物は

ColorRGB.png



これを作って、ColorCMY の方は ColorRGB と似ているのでハッカソン風に自分で直してみる。

ColorRGB, Class の Definition

```
Smalltalk.KSU defineClass: #ColorRGB
  superclass: #{UI.ApplicationModel}
  indexedType: #none
  private: false
  instanceVariableNames: 'redGauge greenGauge blueGauge redField
greenField blueField '
```

```
classInstanceVariableNames: "  
imports: "  
category: 'KSU-Template'
```

それぞれ、スライダーと Field に対応している

redGauge

```
redGauge  
  ifNil:  
    [redGauge := 0.5 asValue.  
     redGauge compute: [:aValue | self updateColorRed: aValue]].  
^redGauge
```

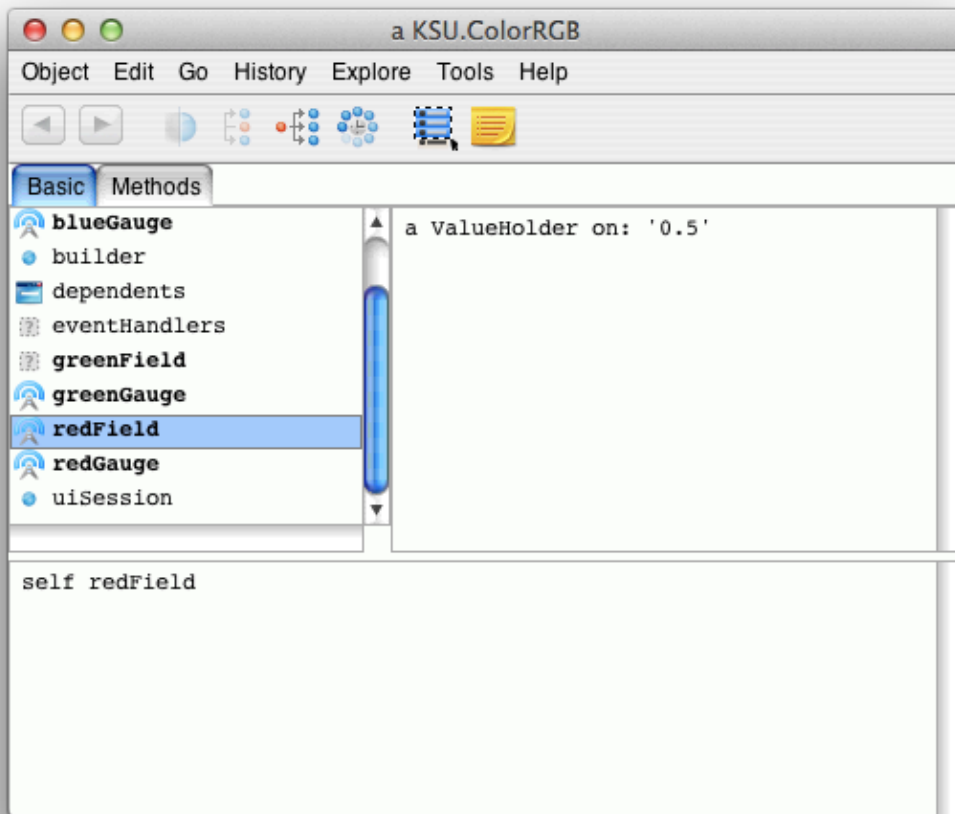
redField

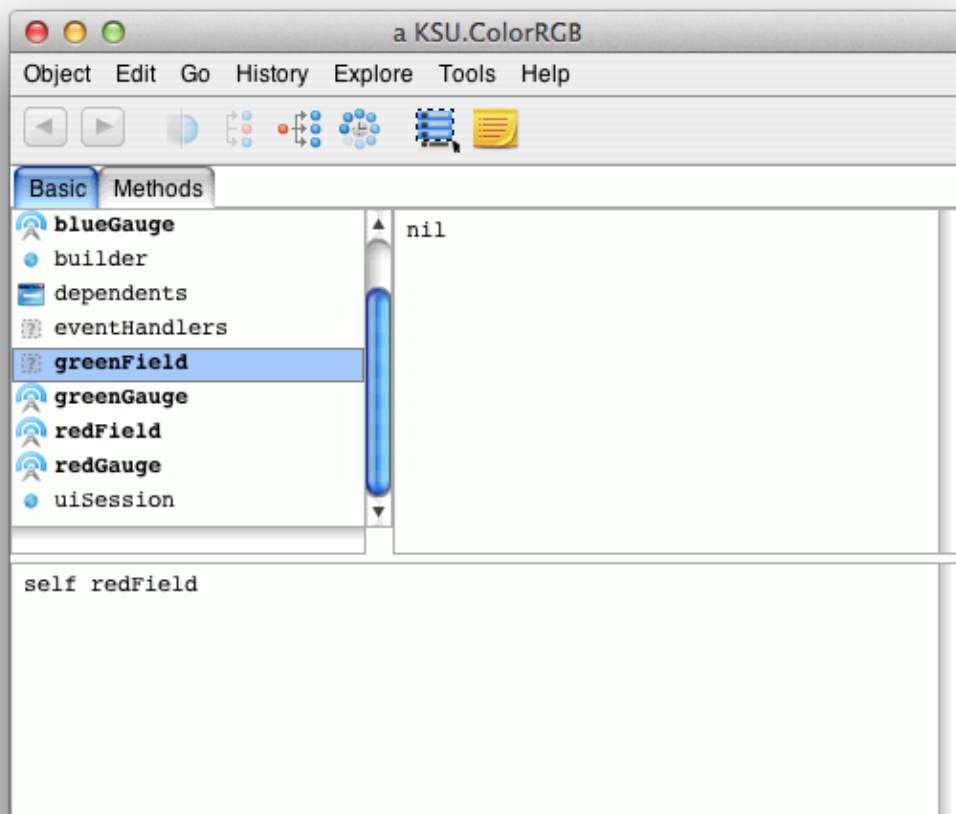
```
redField ifNil: [redField := (self valueString: self redGauge value) asValue].  
^redField
```

example1 を Inspect It すると



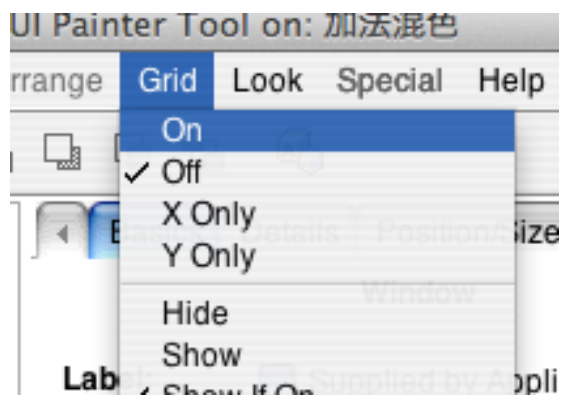
こんなグレーの画面が開く (RGB がそれぞれ 0.5)





Window spec から Edit

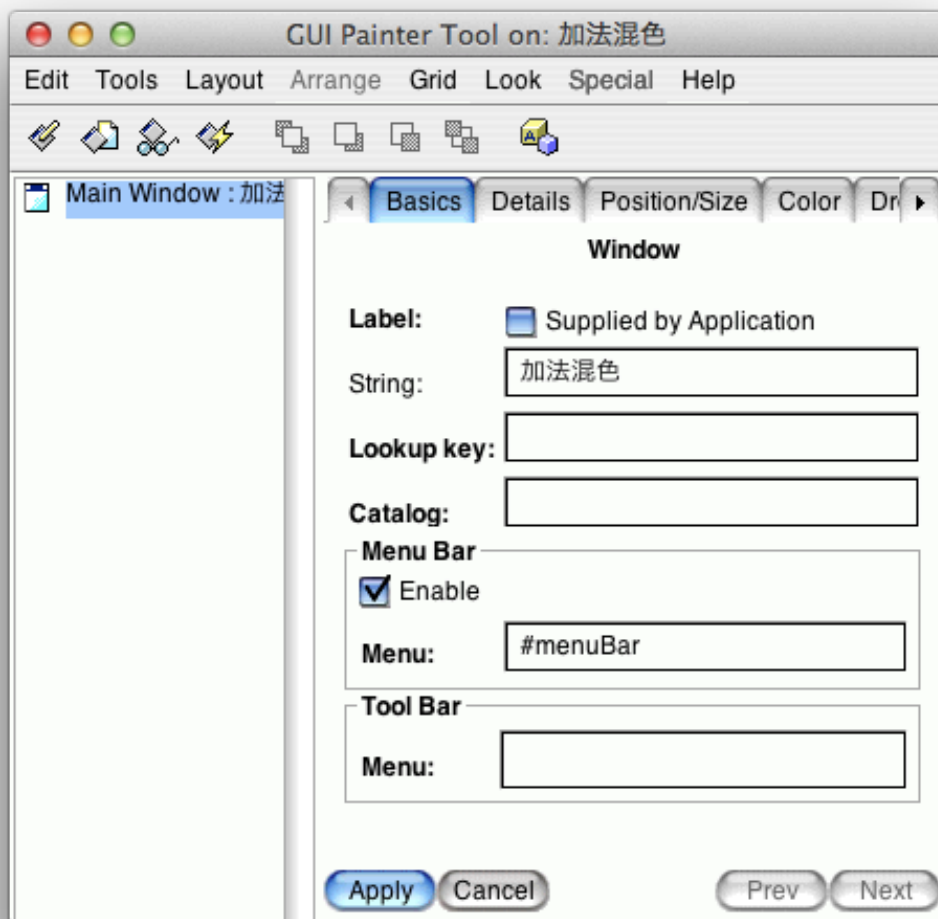
いろいろ置くので Grid を表示しておく




こうなる



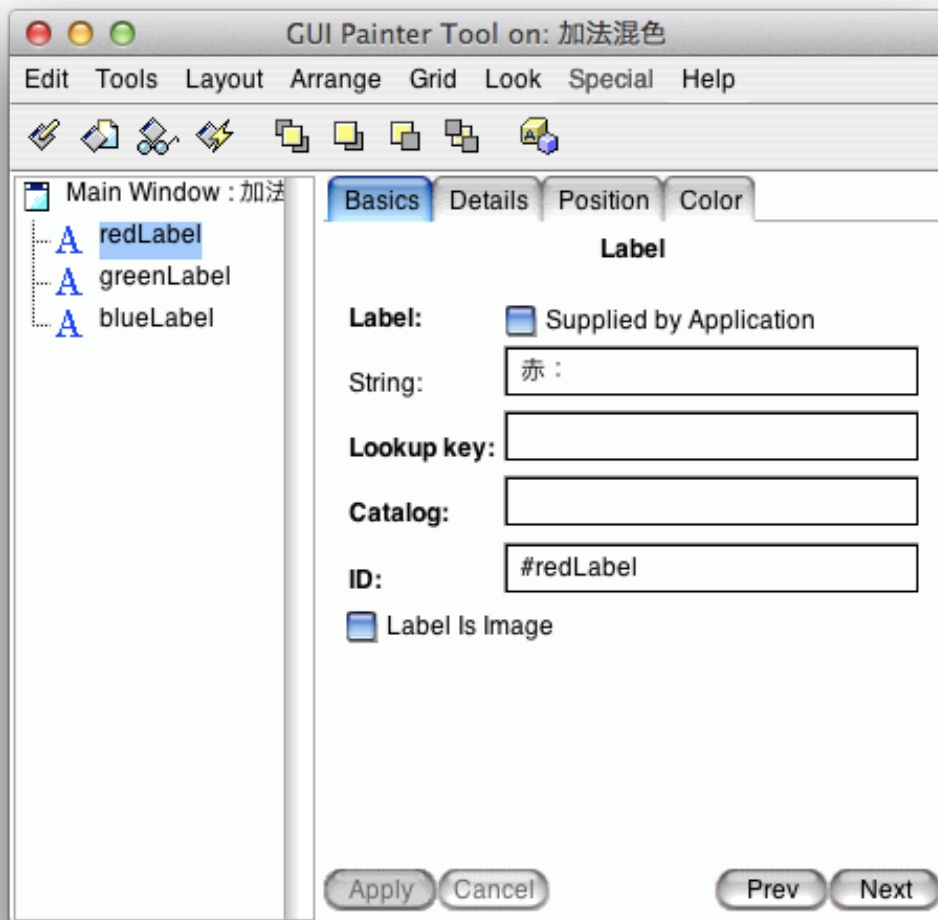
Menu Bar を Enable にする



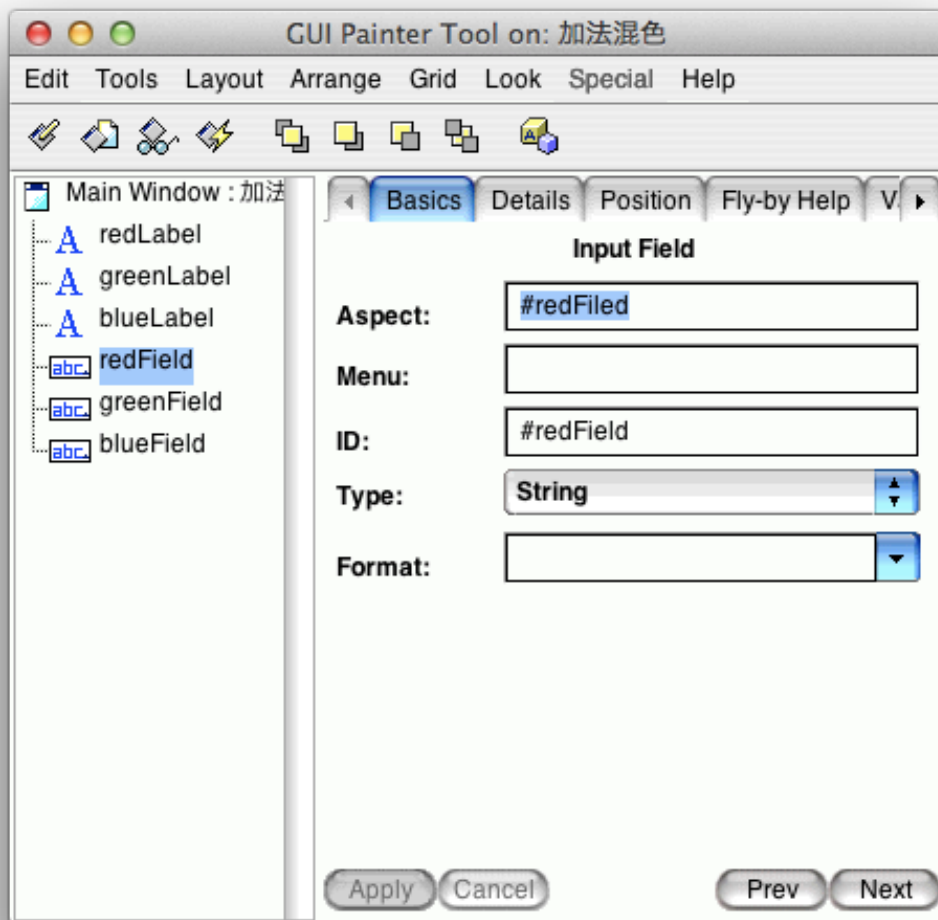
ラベル  を適当に 3 つ置いて



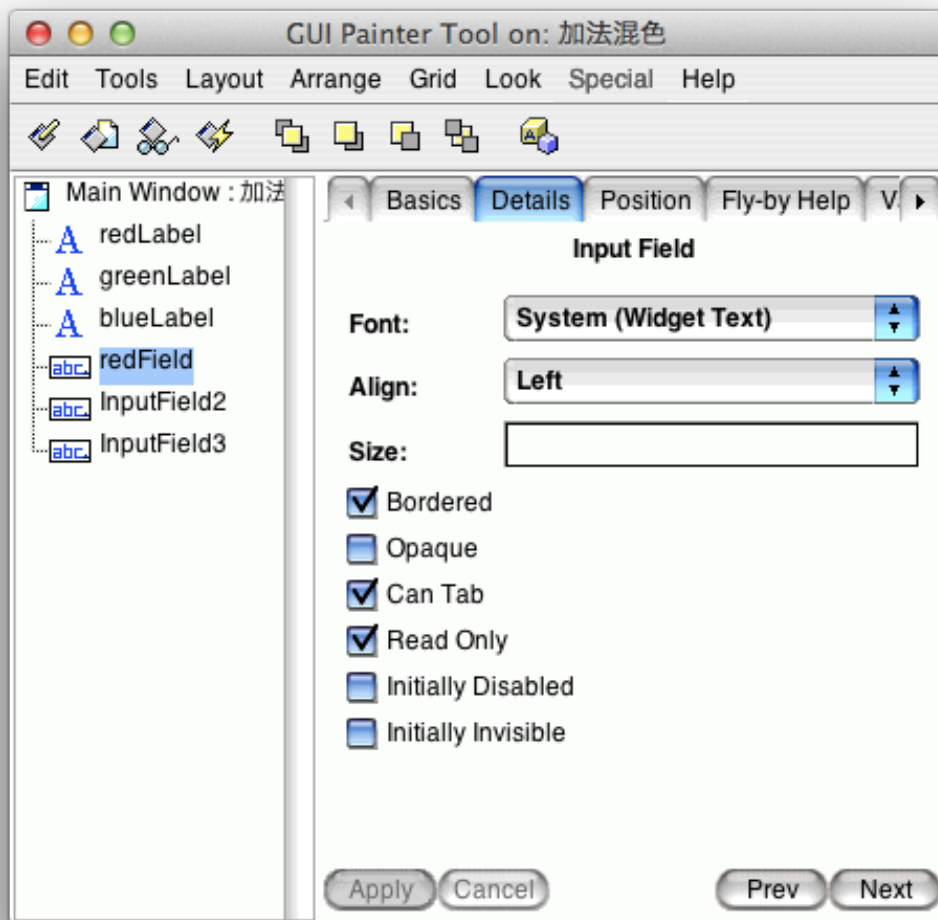
それぞれ、String: に赤:、緑:、青: ID に redLabel, greenLabel, blueLabel とつける



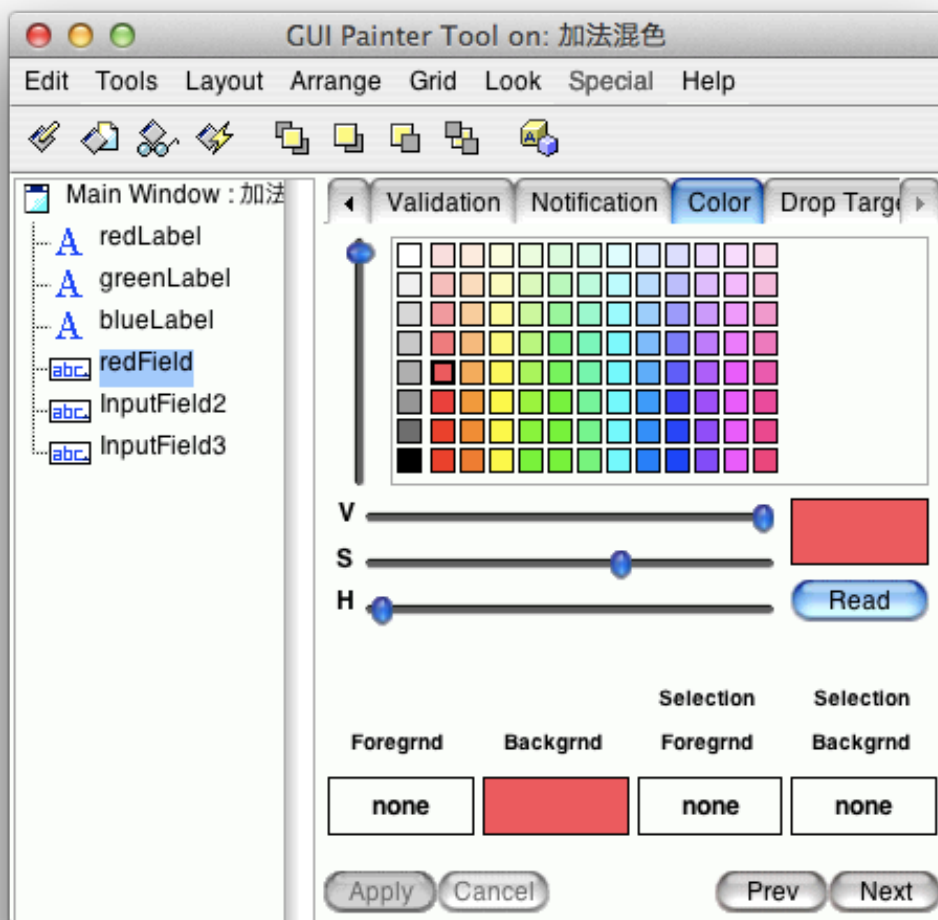
Input Filed  を同様に3つ並べて Aspect と ID を redFiled にする



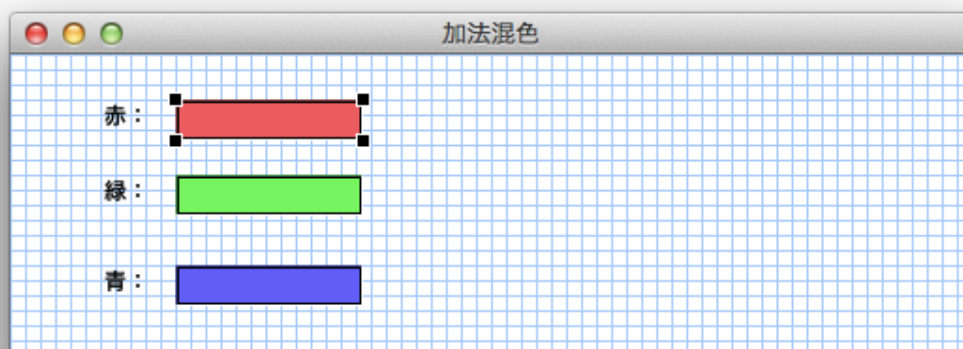
Details タブの Read Only にチェックを入れておく (変更されないようにする)



背景色は Color タブで、色を選択して Backgrnd

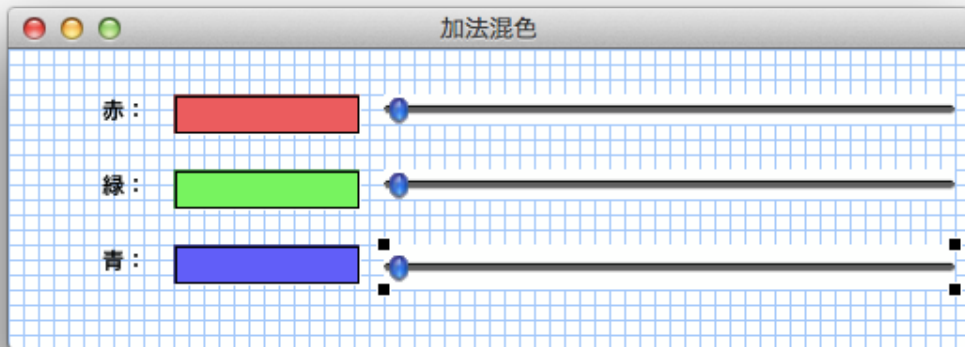


この時点ではこんな感じ

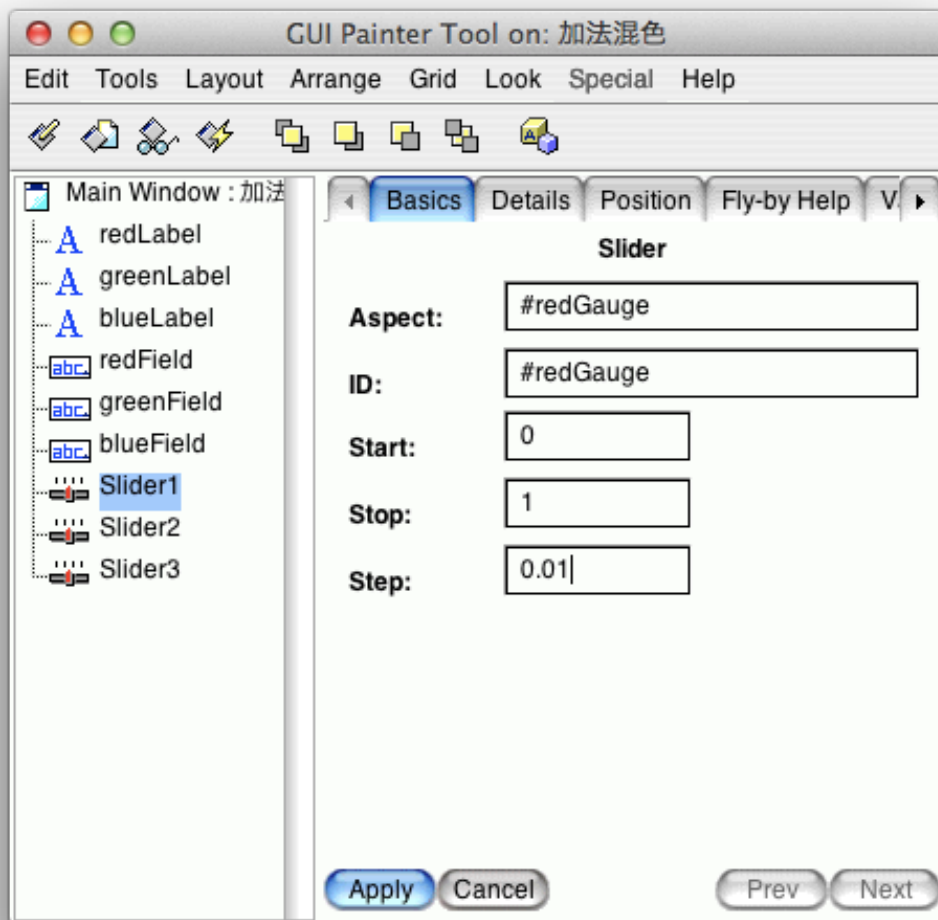



次は、Slider  を3つ置く

ちゃんと、長さも適当にしておく

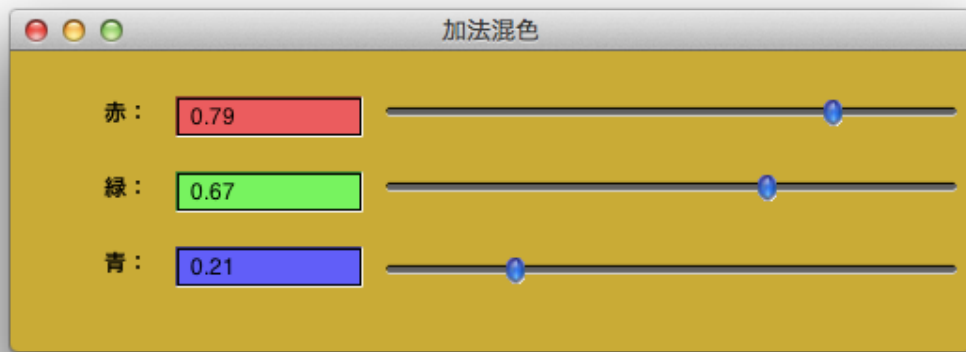


Aspect と ID は同じような感じで redGauge とする
Start, Stop はそれぞれスライダーの上限値と下限値のこと
Step は刻み幅



ここまで来たら Install 

example1 を Do it するとこんな感じで動く



中身を見ていく

初めて開かれたときにウィンドウを初期化する(グレーにする)

postOpenWith: aBuilder

```
super postOpenWith: aBuilder.  
self updateColor
```

実際にバックグラウンドを塗る

updateColor

```
self builder  
  ifNotNil:  
    [:aBuilder |  
      aBuilder window  
        ifNotNil:  
          [:aWindow |  
            aWindow  
              background: self color;  
              display]]
```

これで丸めてしまう

valueString: aValue

```
^(aValue roundTo: 0.01) printString
```

スライダーをいじると呼ばれて、中身を書き換える

updateColorRed: aValue

```
InputState default altDown  
  ifTrue:  
    [self greenGauge value = aValue ifFalse: [self greenGauge value:  
aValue].  
    self blueGauge value = aValue ifFalse: [self blueGauge value:
```

```
aValue]].  
self redField value: (self valueString: aValue).  
self updateColor
```

これで、RGB は終わり



CMY をやる

ColorCMY.st を File in

こんなものを作っていく

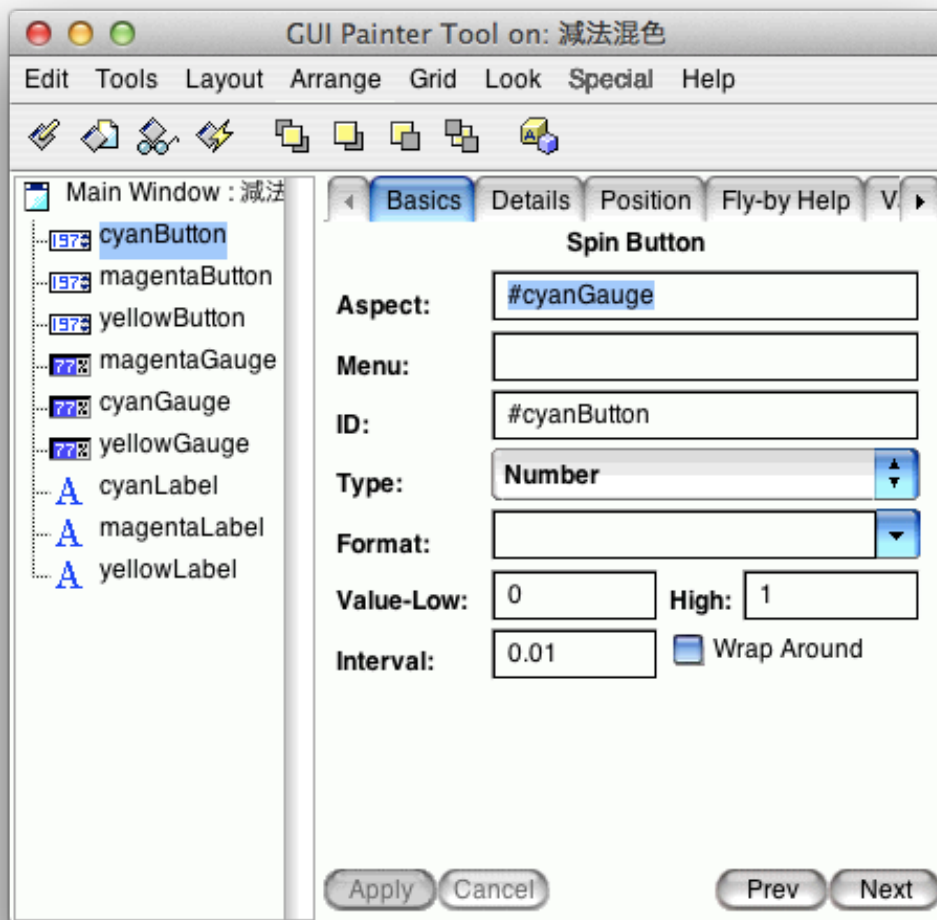


Window Spec を開く

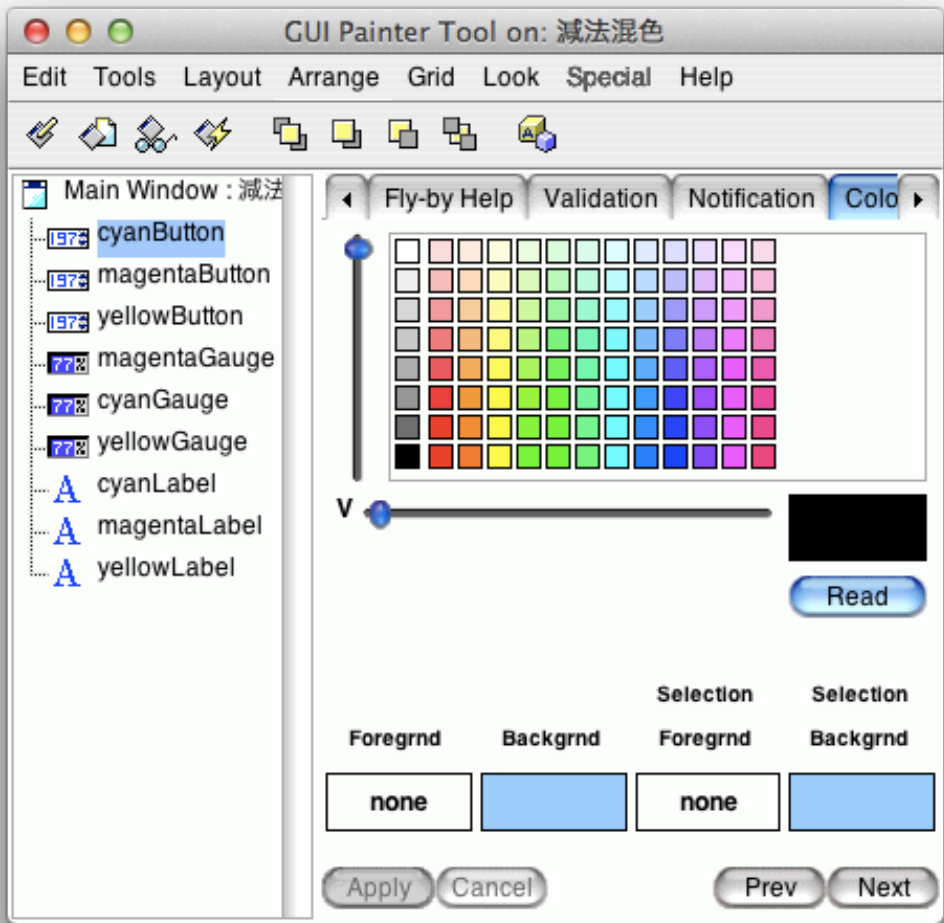
Spin Button  と Percent Done Bar  を使う

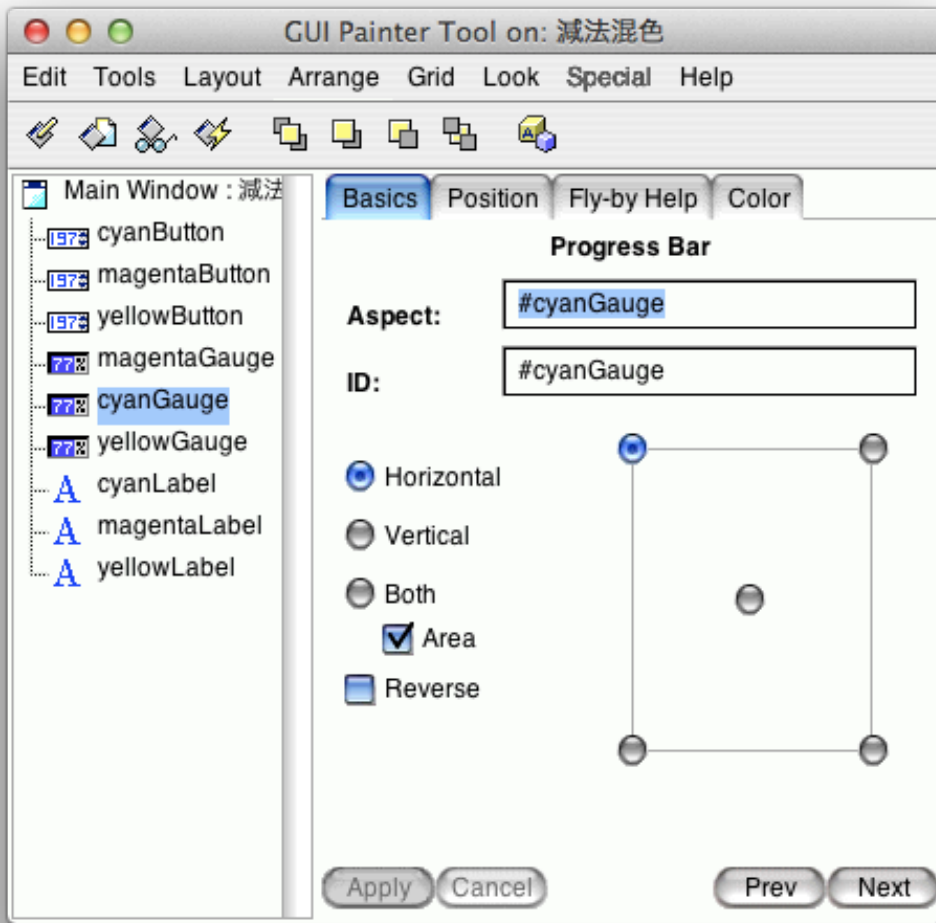


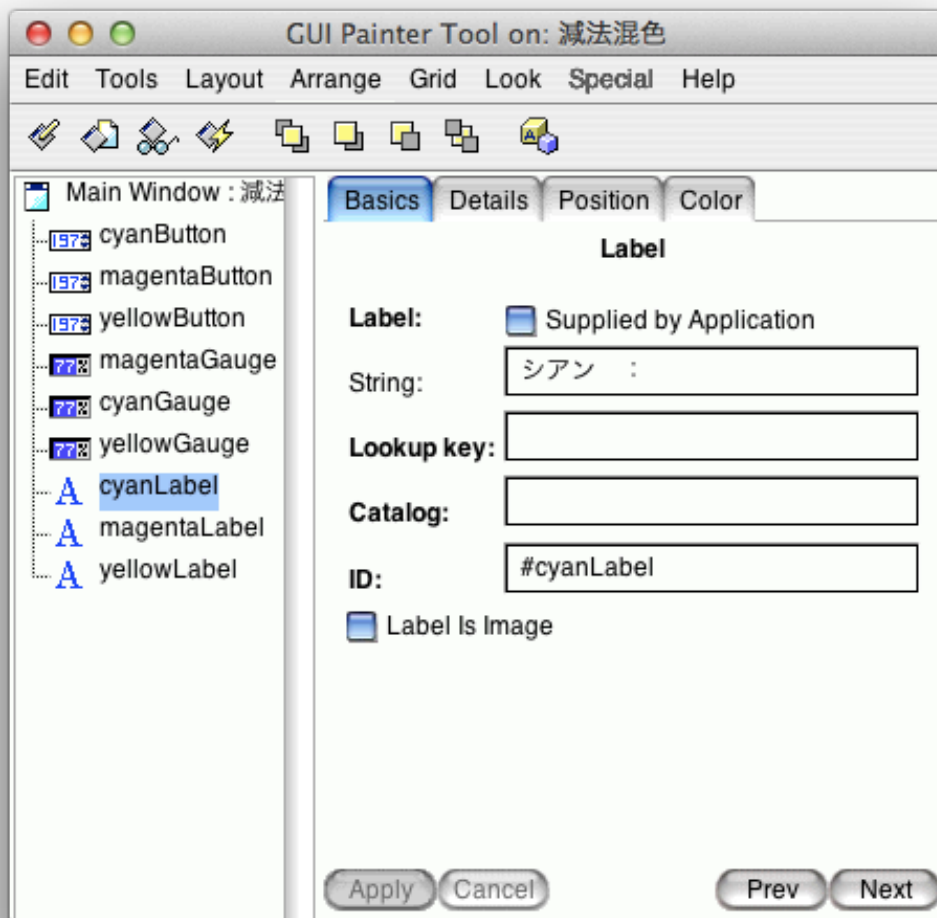
今回はバーとボタンが同じ値を見ている
RGB と違って Type を Number にしている



Selection Backgrnd(o が無いけど仕様と思うしかない...) も色を付けておかないと
チカチカする





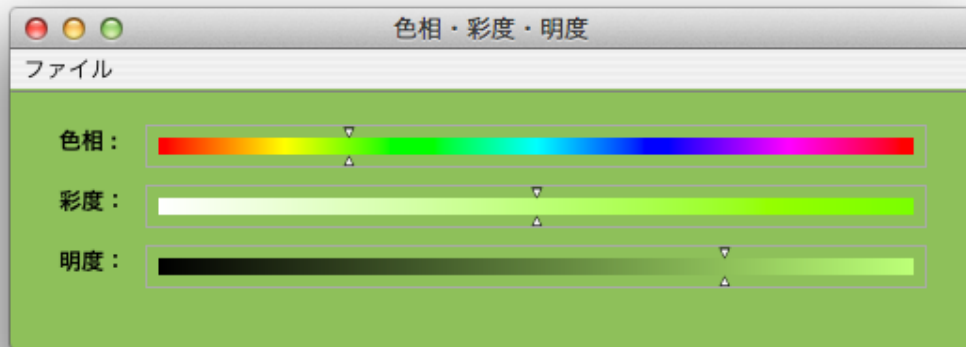


cyan うんたら以外も cyan 部分を magenta や yellow と変更する。

こんな感じで動くようになった



最後に HSB



こんな部品はないので自前でつくる
HSB は Model だけじゃなくて Controller と View がある

Definition をみると Gauge が3つだけ

```
Smalltalk.KSU defineClass: #ColorHSB
  superclass: #{UI.ApplicationModel}
  indexedType: #none
  private: false
  instanceVariableNames: 'hueGauge saturationGauge brightnessGauge '
  classInstanceVariableNames: ''
  imports: ''
  category: 'KSU-Template'
```

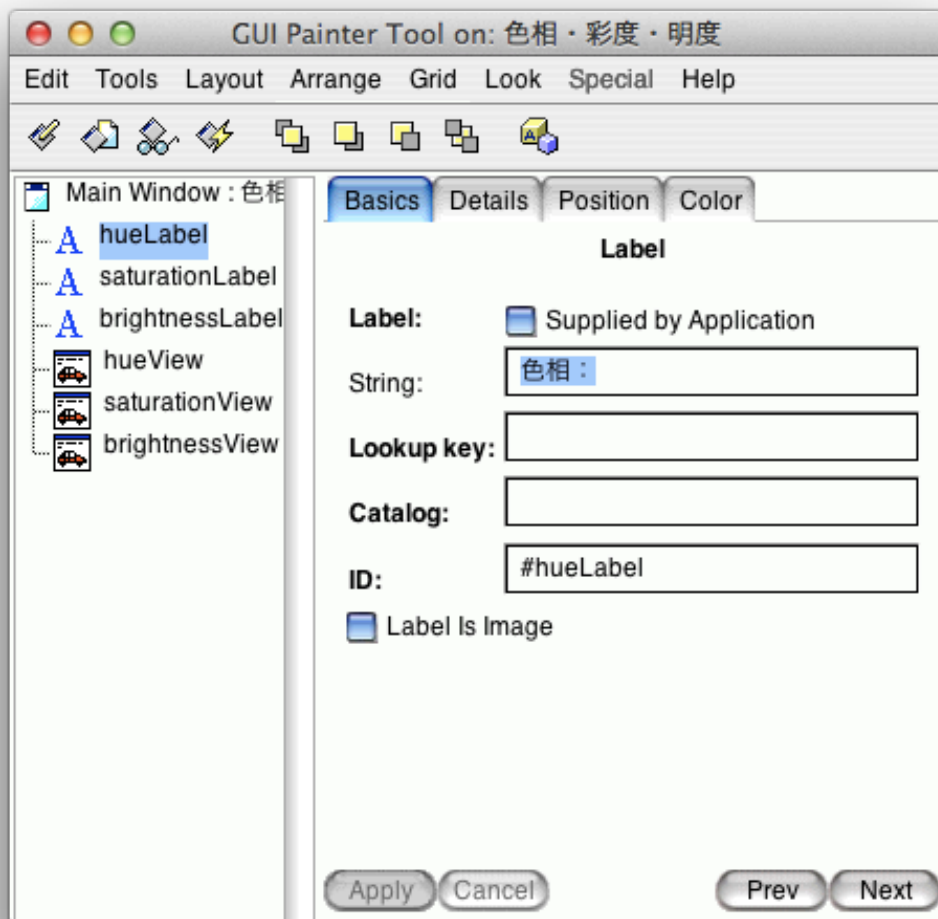
interface opening を見ると、それぞれの View がある

WindowSpec を Edit

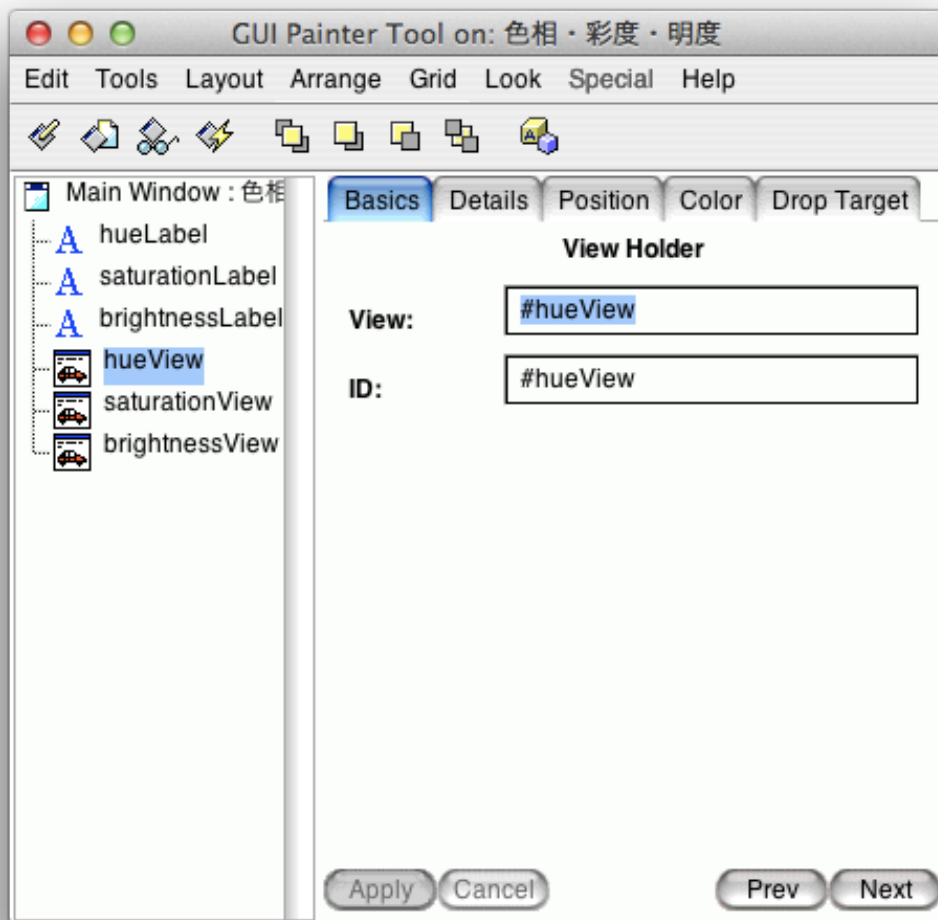
View Holder  を使う
この様に設置して



それぞれのラベルを設定して

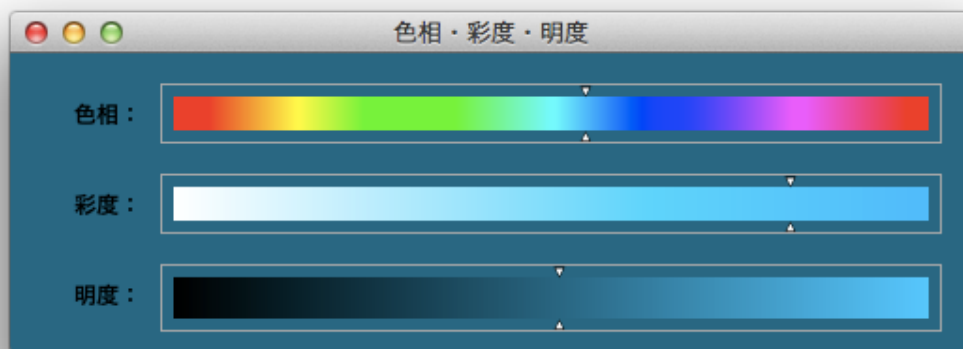


View Holder も設定する



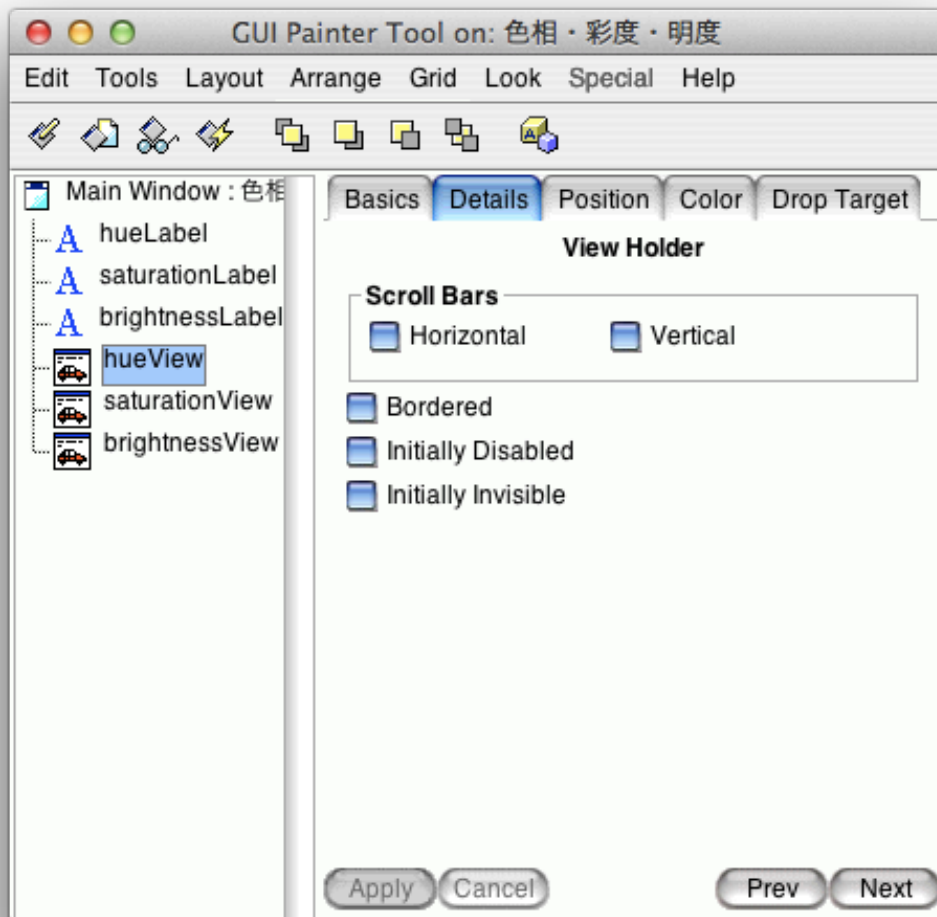
で、最後に Install

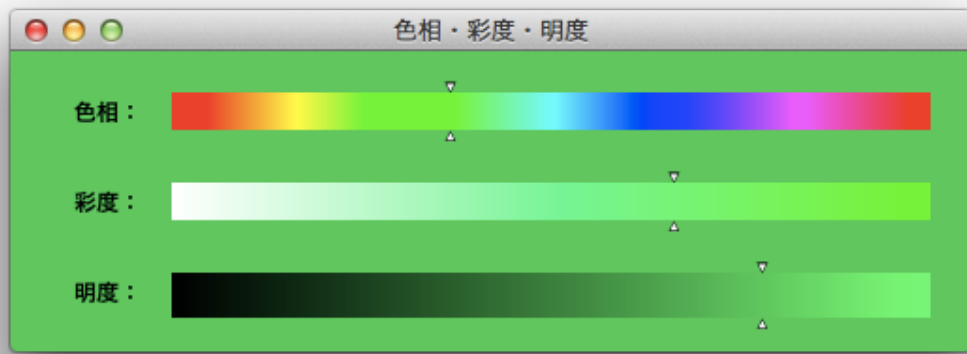
この状態で、example1 を実行すると



この様に動く

で、変な線が出ていて見栄えが良くないので、
Bordered のチェックボックスを外すと消える





これにておしまい。