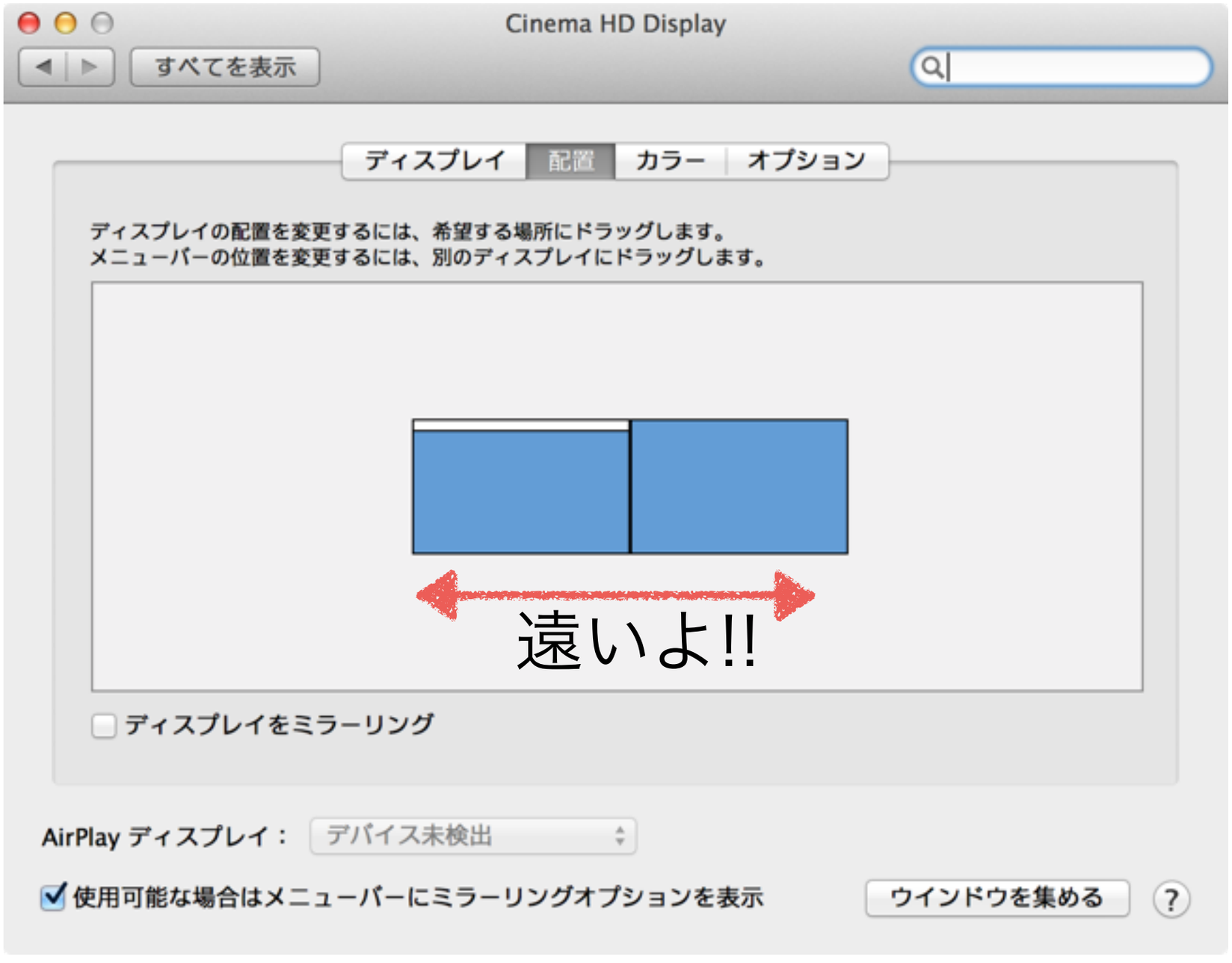
# インスペクタの
# 設計と実装

mindtools@mac.com

# なぜToolPaletteを複数必要なのか

# マルチDisplay環境での使い勝手の向上



# Tool Paletteのクラス構造

オリジナルは

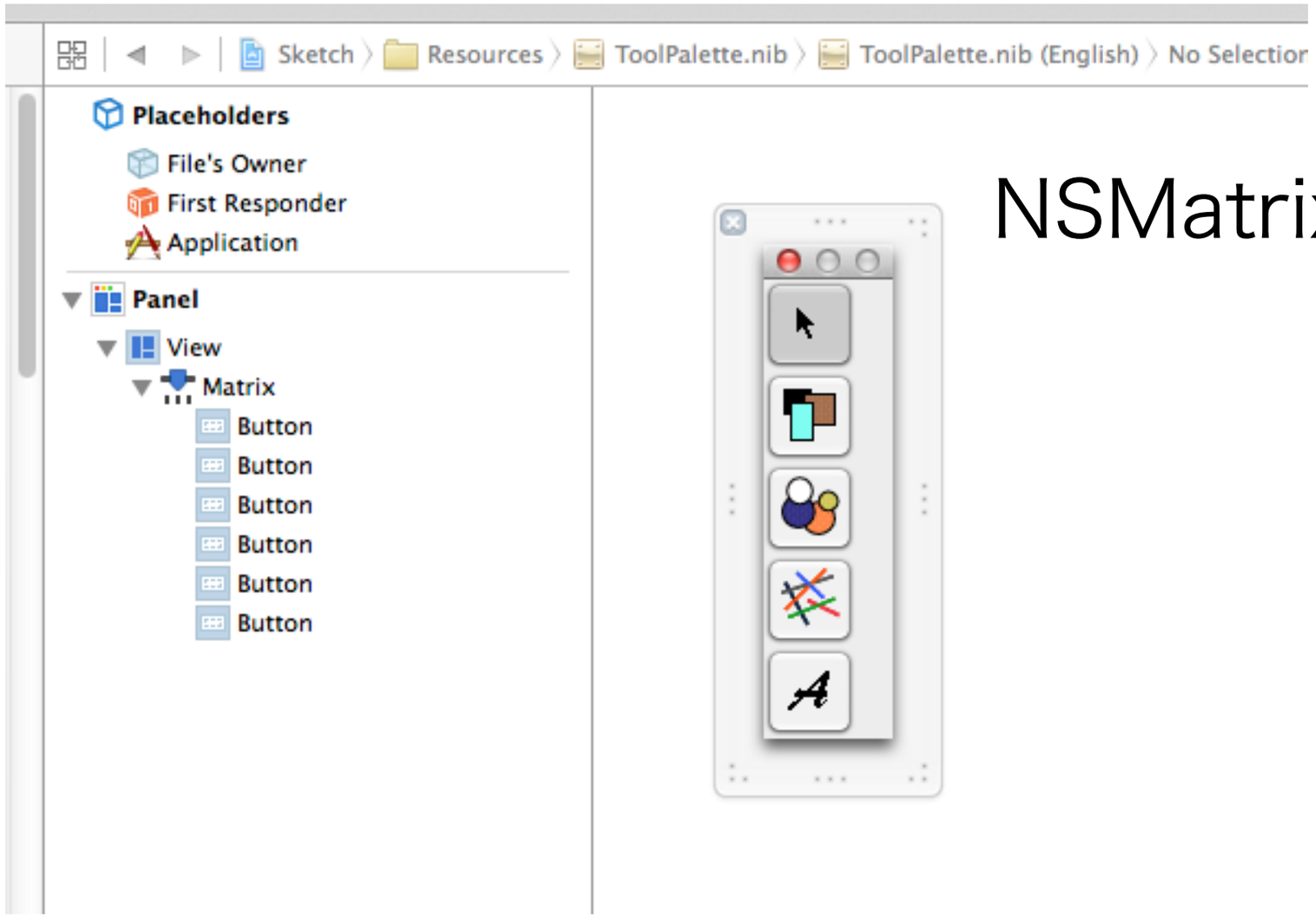
こんな感じ

# 2つの要素で構成

SKTToolPaletteController.m


ToolPalette.nib

# ToolPalette.nib




NSMatrixでRadioButton type

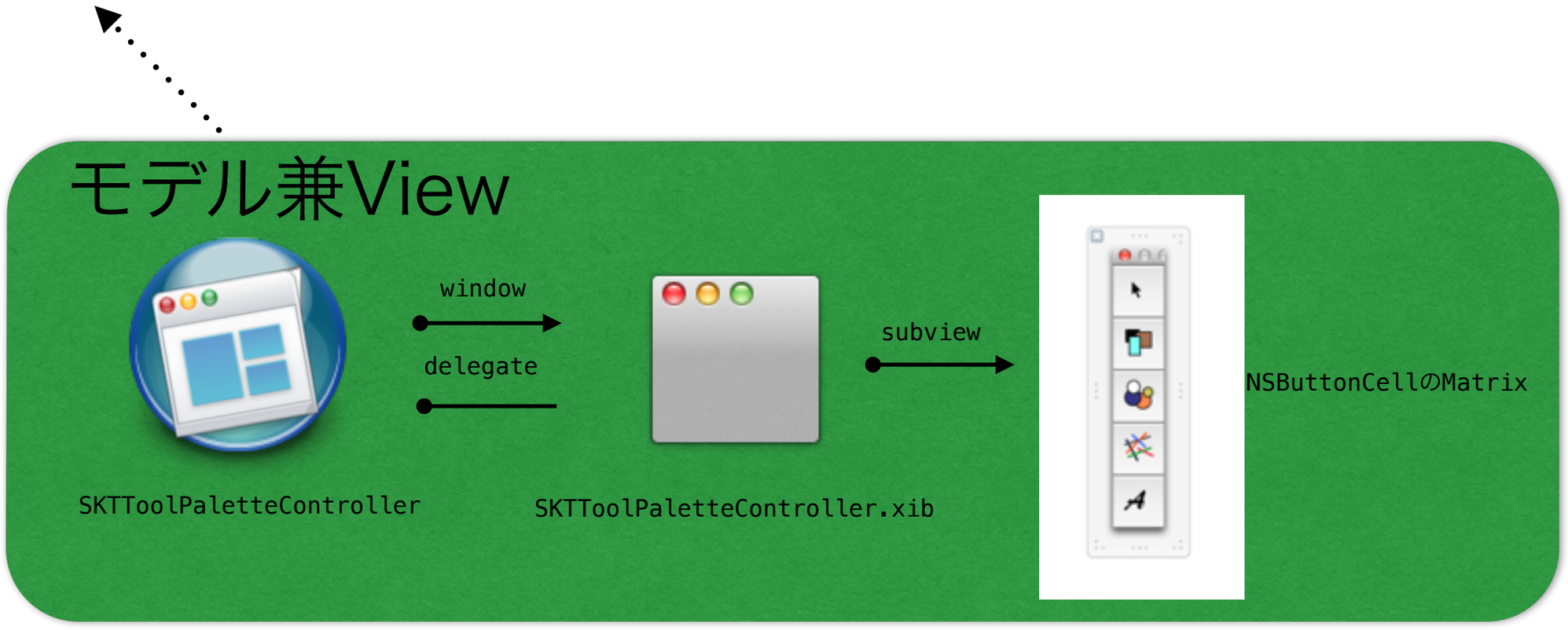
OwnerはSKToolPaletteControllerクラス

# SKTToolPaletteController

- シングルトン化
- ツールが変更された事をNotification
- どのツールを選んでいるかを保持してるのはButton cellのMatrix(!!)

# ModelとViewが同一

SKTSelectedToolDidChangeNotificationで
Toolの変更を知らせる



# 改造するとこんな感じ

# 改造の方針

- どのツールを選択しているかのデータをモデルクラスに持たせる。

- パレットは複数表示可能にする。
    - createPalette: メソッドの作成

- viewの表示はbindingを使って、モデルクラスの値と同期する。

- モデルクラスからツールの変更をNotificationする

# 3つの要素で構成

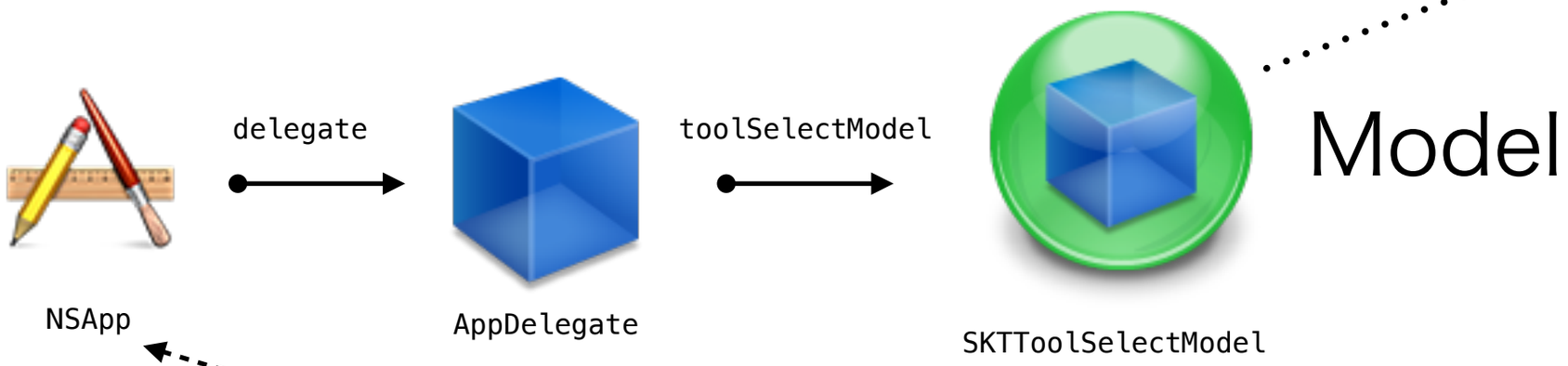
SKTToolSelectModel

SKTToolPaletteController


ToolPalette.nib

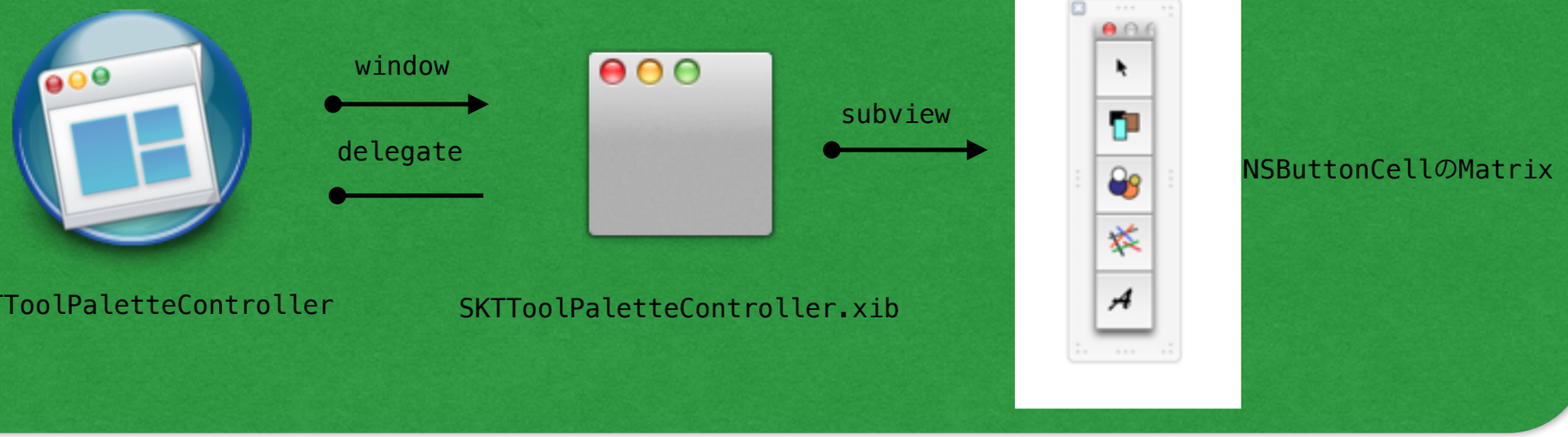
# 改良はModelとViewを分ける

Notificationで
Toolの変更を知らせる




bindTo:
self.delegate.toolSelectModel.currentToolNumber

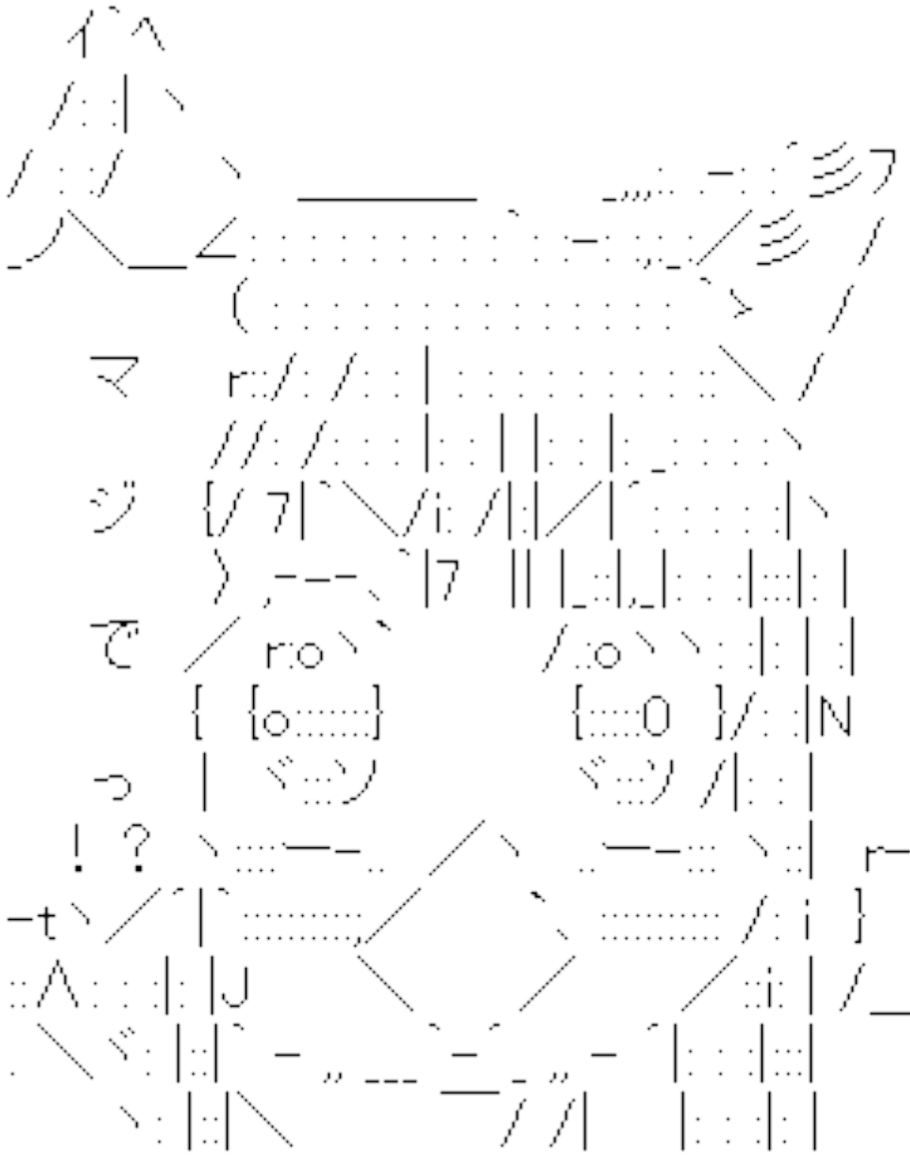
## ViewとViewController



# コードの見所

勝手に消えるのを防ぐ

# ARC環境で、NSPanelをalloc&initで作って放置したら Autoreleseされて次々と消えていった。!!



SKTToolPaletteController

# 勝手にリリースされないように(1)

```
static NSMutableArray* sWindowArray;

+ (void)addToolPaletteWindowController:(SKTToolPaletteController *)windowController
{
    if( sWindowArray == nil )
    {
        sWindowArray = [[NSMutableArray alloc] init];
    }

    [sWindowArray addObject:windowController];
}

+ (void)removeToolPaletteWindowController:(SKTToolPaletteController *)windowController
{
    [sWindowArray removeObject:windowController];
}
```

SKTToolPaletteController

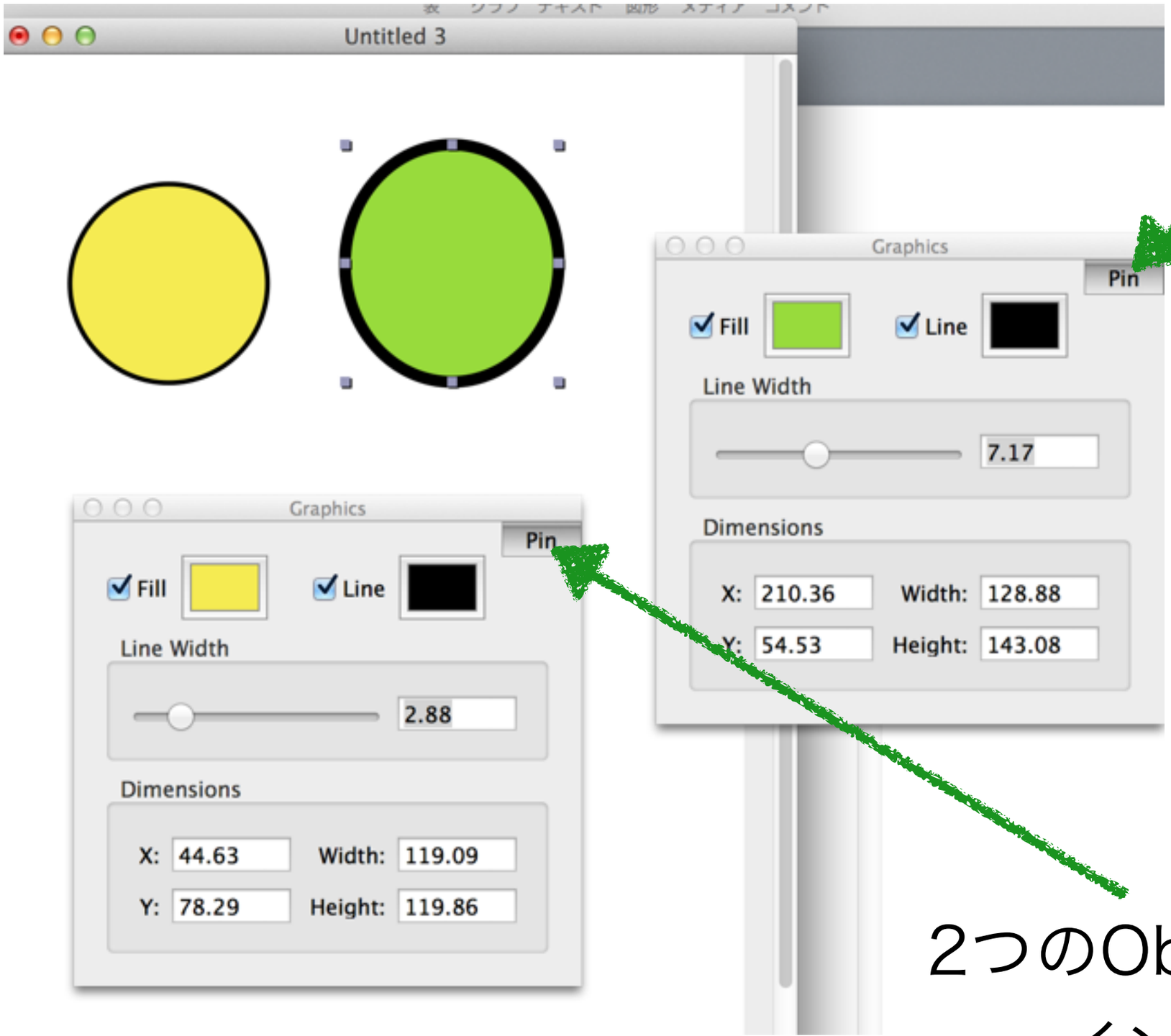
# 勝手にリリースされないように(2)

```
- (id)init
{
    self = [self initWithWindowNibName:@"SKTToolPaletteController"];
    if (self)
    {
        [SKTToolPaletteController addToolPaletteWindowController:self];
    }

    return self;
}

- (void)windowWillClose:(NSNotification *)notification;
{
    [SKTToolPaletteController removeToolPaletteWindowController:self];
}
```

これ以外のコードは簡単なので解説無し。
質問あればドーゾ

# なぜInspectorPanelが複数必要なのか

Pinを押すと固定




2つのObjectを同時に
インスペクト

# InspectorPanelの
# クラス構造

オリジナルは

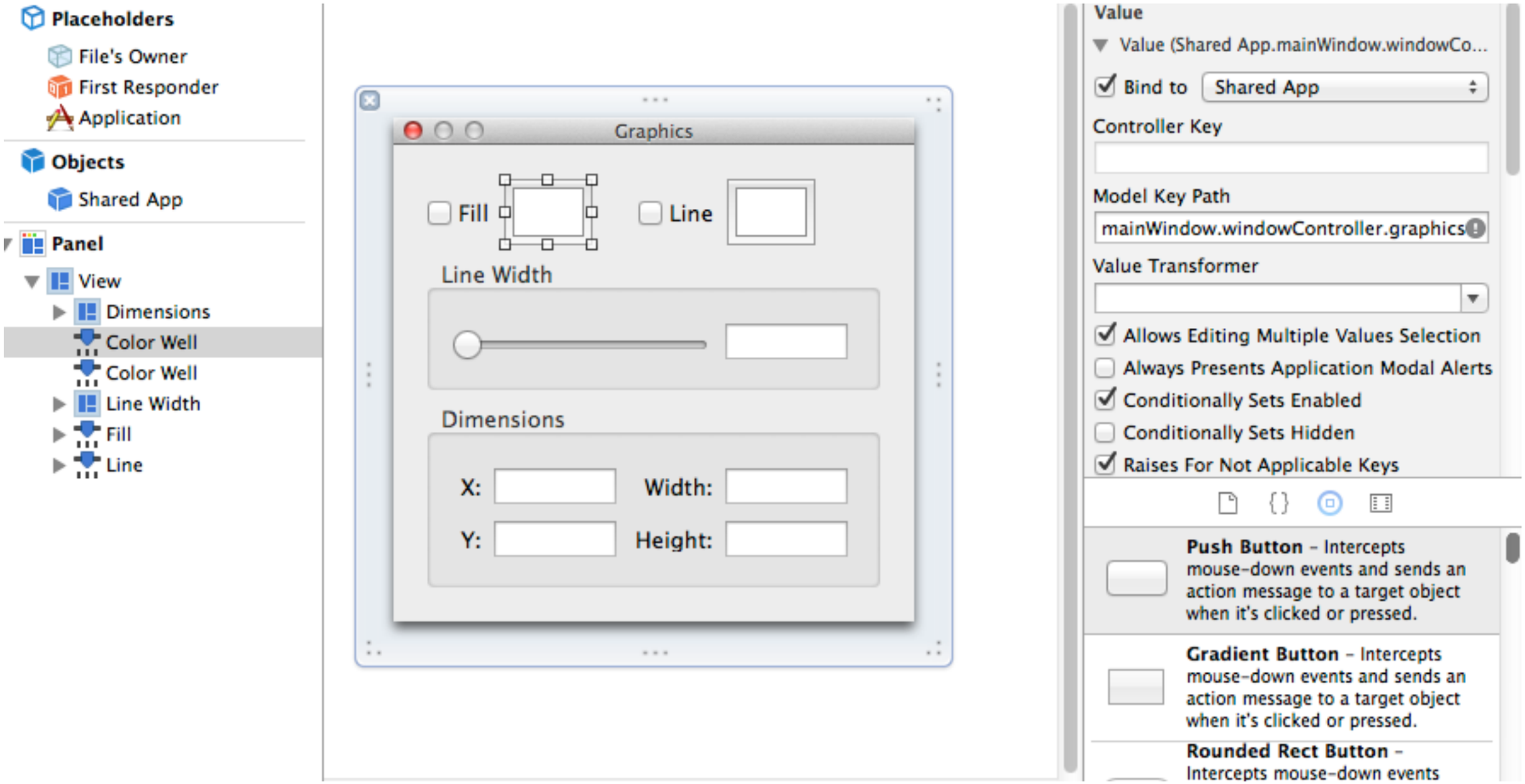
こんな感じ

# 1つの要素で構成

WindowController無し


Inspector.nib

# Inspector.nib




各種ボタン等は、
mainWindow.windowController.**graphicsController**.selection.XXXXXにバインディング

**graphicsControllerはNSArrayController**

# 大回りしてBinding


**NSApp.mainWindow.windowController.graphicsController.selection.XXXXXにバインディング**


Inspector.nib

NSApp.mainWindowはfirstResponderのように随時変わる値。
これによって”今選択中のWindowのオブジェクト”へbindingできている。
問題は、同時に他のObjectの値をインスペクトできない。

# 改造するとこんな感じ

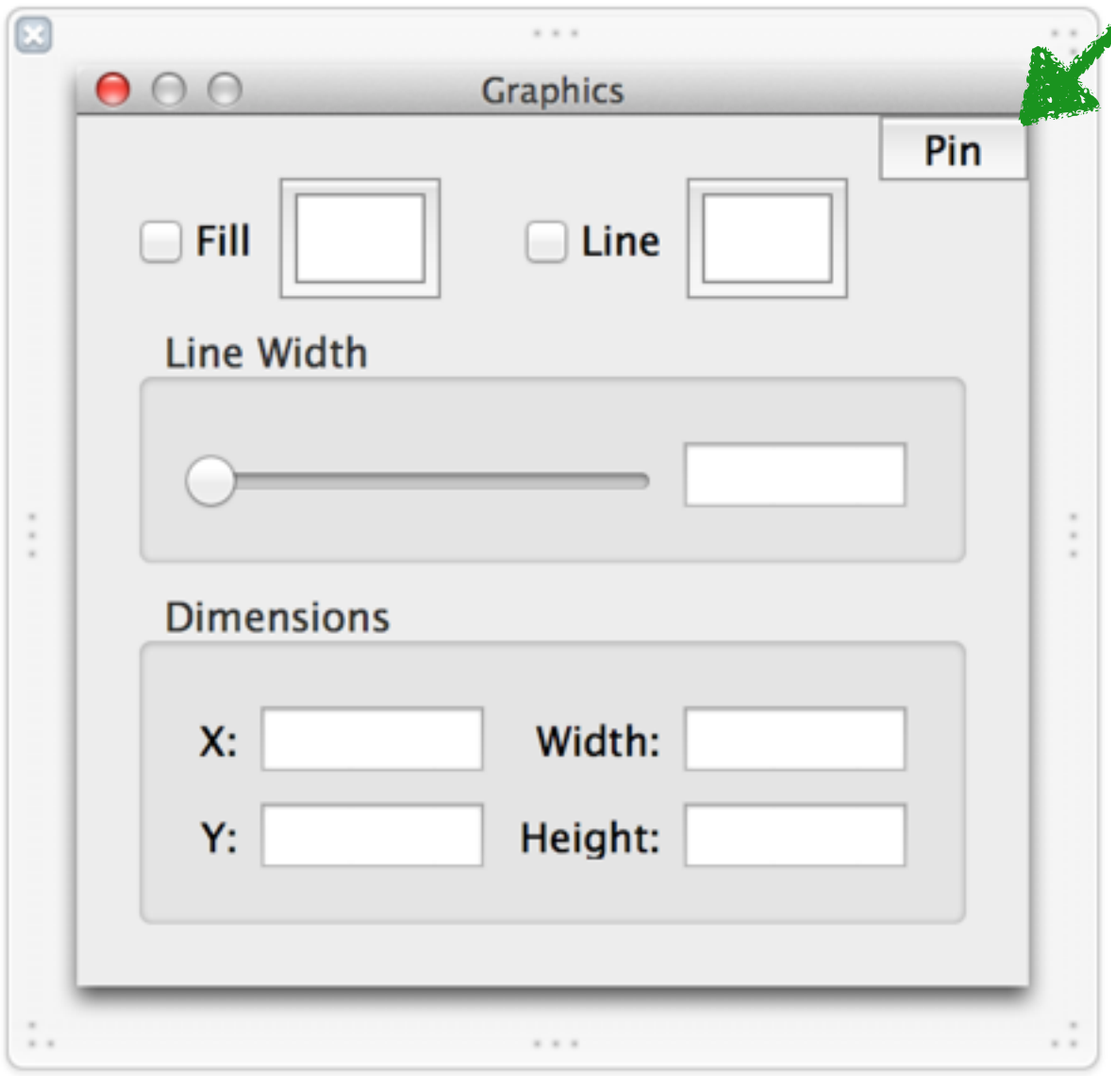
# 改造の方針

- Windowに1個ボタンを追加

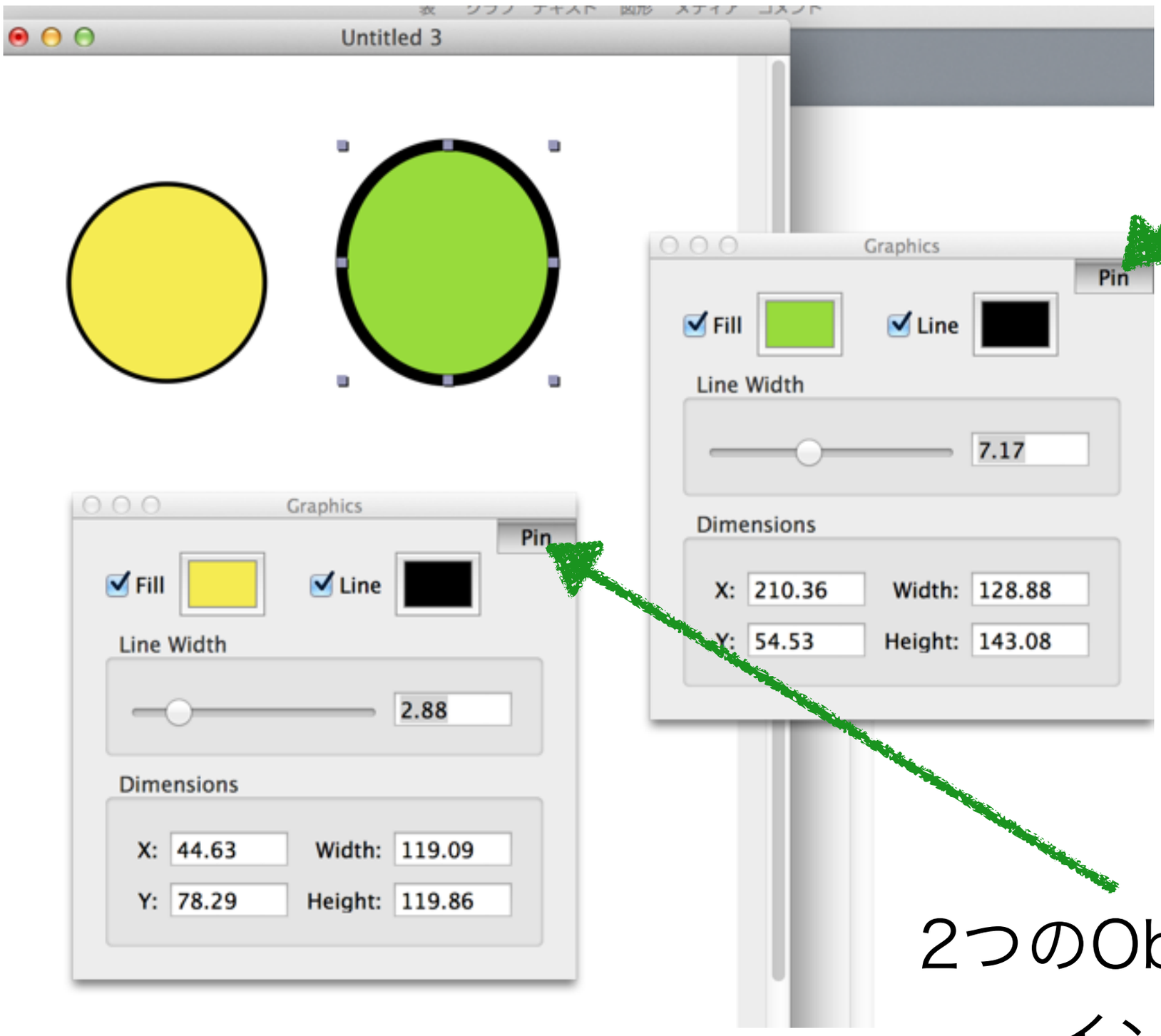
- 特定のオブジェクトをインスペクトしているモードと、カレントウインドウのオブジェクトをインスペクトしているモードの2つのモードを作る

- 特定のオブジェクトをインスペクトしている時はWindowControllerにそのオブジェクトへの参照を持つ。

このPinボタンがスゴイ



Pinを押すと固定




2つのObjectを同時に
インスペクト

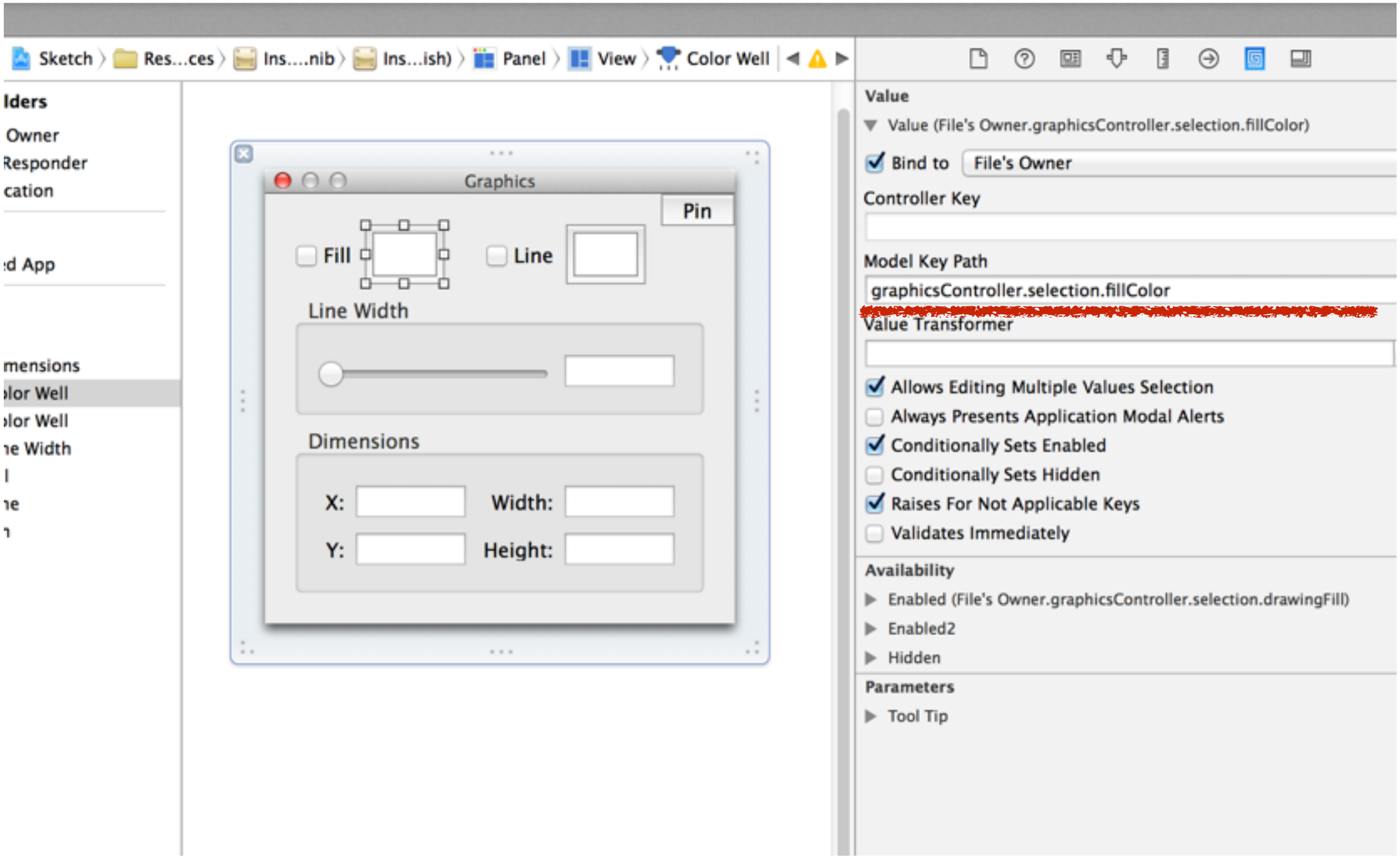
スゴイだろ？

# 2つの要素で構成

SKTInspectorPaletteController


Inspector.nib

# Inspector.nib

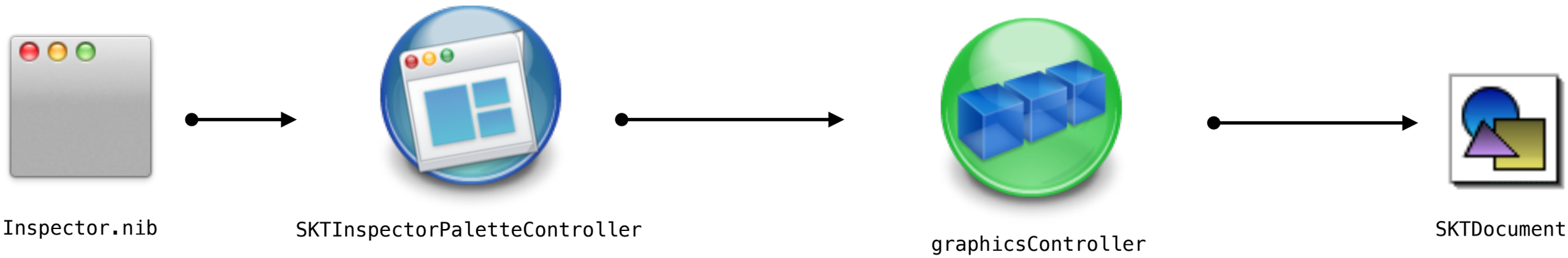



各種ボタン等は、

file's Owner.**graphicsController**.selection.XXXXXにバインディング

**graphicsControllerはNSArrayController**

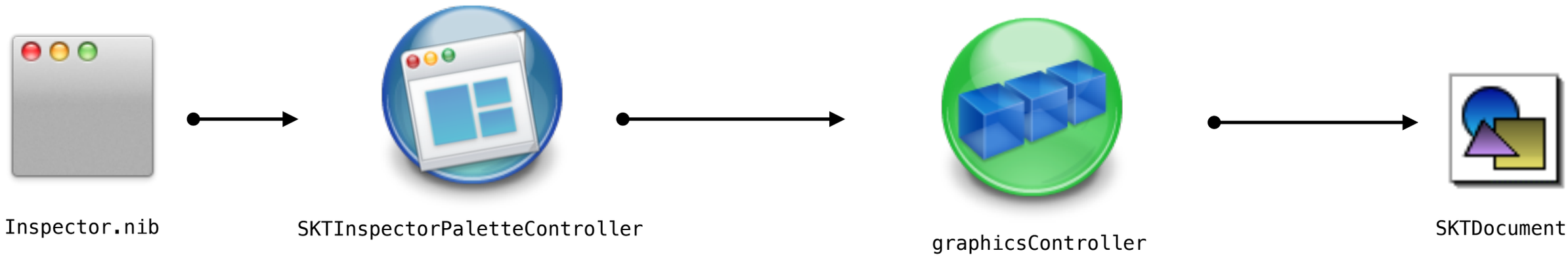
# Pinされていない時のクラス関係図




**File’s Owner.graphicsController.selection.XXXXXにバインディング**

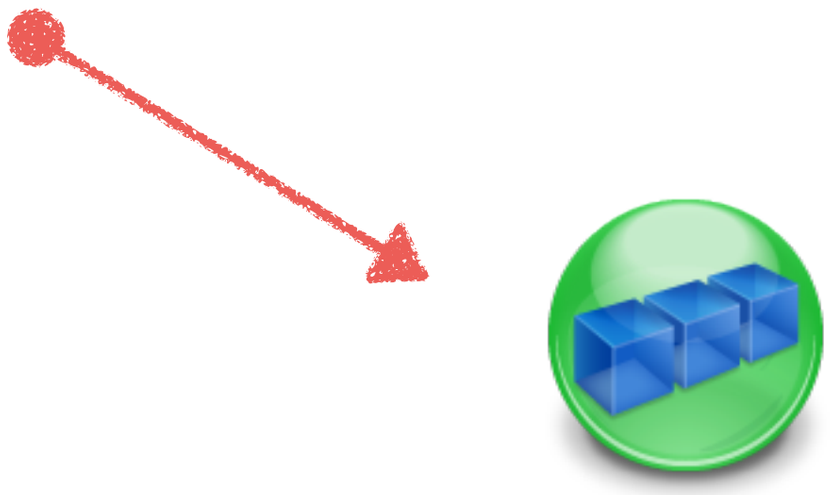
# Pinされている時のクラス関係図

**bindingは変わらない!!**




**File’s Owner.graphicsController.selection.XXXXXにバインディング**

# これをKVOイベントによって随時変更する



graphicsController

# Pinボタンを押された時に呼ばれるメソッド

```
- (IBAction) pinnedAction:(id)sender
{
    if( self.isPinned )
    {
        [self stopFloatingType];
        [self startFixedType];
    }
    else
    {
        [self stopFixedType];
        [self startFloatingType];
    }
}
```

# Pinボタンを押された時に呼ばれるメソッド

```
    // mainWindowに追随するのを開始
[self startFloatingType];

    // mainWindowに追随するのを停止
[self stopFloatingType];

    // 特定Objectへの追跡を開始
[self startFixedType];

    // 特定Objectへの追跡を停止
[self stopFixedType];
```

# コードの見所

どーやって追跡するの？

SKTInspectorPaletteController

# MainWindowを追跡を開始

```
// 変動インスペクタ型 (floating type)
- (void) startFloatingType
{
   // 初期値としてメインウインドウをインスペクト対象にする
    self.graphicsController = [[NSApp mainWindow]
                                    valueForKeyPath:@"windowController.graphicsController"];

    // メインウインドウが変われば選択箇所も変わる。インスペクトする場所も変わるように変更追跡を開始する。
    [NSApp addObserver:self
            forKeyPath:@"mainWindow"
              selector:@selector(updateFloatingType)
              userInfo:nil
               options:0];
}
```

SKTInspectorPaletteController

# MainWindowを追跡をやめる

```
- (void) stopFloatingType
{
    // メインウインドウの変更追跡を停止する
    [NSApp removeObserver:self
                  keyPath:@"mainWindow"
                 selector:@selector(updateFloatingType)];

    self.graphicsDocument   = nil;
    self.graphicsController = nil;
}
```

SKTInspectorPaletteController

# MainWindowに変更があれば更新

```
- (void) updateFloatingType
{
    // 追跡対象に変更が有ったので、インスペクトする場所を再設定する
    self.graphicsController = [[NSApp mainWindow]
            valueForKeyPath:@"windowController.graphicsController"];
}
```

# 固定インスペクタ型 (fixed type)はどーする？

SKTToolPaletteController

```
- (void) startFixedType
{
    .
    .
    .

    // 対象とするgraphics(NSArray)のコピーを保存(スナップショット)
    self.selectedGraphics = [[[NSApp mainWindow]
                                valueForKeyPath:@"windowController.graphicsController.selectedObjects"] copy];

    // コントローラを作成して、上記のスナップショットを割り当てる
     self.graphicsController = [[NSArrayController alloc ] initWithContent:self.selectedGraphics];

    // 当然selectionはオブジェクト全部
     [self.graphicsController setSelectedObjects:self.selectedGraphics];


   // 画像の増減の検知を開始(undo/redo対策)
     [self.graphicsDocument addObserver:self
                             forKeyPath:@"graphics"
                               selector:@selector(updateFixedTypeForGraphics)
                               userInfo:nil
                                options:0];
    .
    .
    .

}
```

SKTToolPaletteController

# 固定インスペクタ型 (fixed type)はどーする？

## インスペクトするObject群を保持する

```
- (void) startFixedType
{
    .
    .
    .

    // 対象とするgraphics(NSArray)のコピーを保存(スナップショット)
    self.selectedGraphics = [[[NSApp mainWindow]
                                valueForKeyPath:@"windowController.graphicsController.selectedObjects"] copy];

    // コントローラを作成して、上記のスナップショットを割り当てる
     self.graphicsController = [[NSArrayController alloc ] initWithContent:self.selectedGraphics];

    // 当然selectionはオブジェクト全部
     [self.graphicsController setSelectedObjects:self.selectedGraphics];


   // 画像の増減の検知を開始(undo/redo対策)
     [self.graphicsDocument addObserver:self
                             forKeyPath:@"graphics"
                               selector:@selector(updateFixedTypeForGraphics)
                               userInfo:nil
                                options:0];
    .
    .
    .

}
```

SKTToolPaletteController

# 固定インスペクタ型 (fixed type)はどーする？

## Documentの削除を検知する

```
- (void) startFixedType
{
    .
    .
     // Document削除の検知を開始
     [[NSNotificationCenter defaultCenter]
              addObserver:self
                 selector:@selector(updateFixedTypeForDocumentWillDealloc)
                     name:SKTDocumentWillDeallocNotification
                   object:self.graphicsDocument];
    .
    .

}
```

対象のオブジェクトがなくなったら、

元のmainWIndowに追随するようにする

SKTToolPaletteController

# 固定インスペクタ型 (fixed type)はどーする？

# Undo/Redo対策(汎用性有るのか？)

```
// 追跡中のgraphicsDocumentのgraphicsが変更された(増減が有った)
- (void) updateFixedTypeForGraphics
{
    // Undo/Redo対策に、ドキュメントが直接保持しているグラフィックオブジェクトだけを対象にする。
    // 何も考えないと、undo/redoスタックに入ったオブジェクトもインスペクタから編集できてしまう。
    // 保管されたスナップショット   :  self.selectedGraphics
    // ドキュメントの中のgraphics : self.graphicsDocument.graphics
    // 上記2つの&をとって、その値をコントローラーに設定する。
    // しかし、配列の順列を順列を維持するために以下のようになる。

    NSMutableSet* theSelectedGraphicsSet = [NSMutableSet setWithArray:self.selectedGraphics];
    NSSet*        theDocumentGraphicsSet = [NSSet setWithArray:
                                                [self.graphicsDocument valueForKey:@"graphics"]];

    [theSelectedGraphicsSet minusSet:theDocumentGraphicsSet];

    NSMutableArray* theSelectedGraphics = [NSMutableArray arrayWithArray:self.selectedGraphics];
    [theSelectedGraphics removeObjectsInArray:[theSelectedGraphicsSet allObjects]];

    self.graphicsController = [[NSArrayController alloc ] initWithContent:theSelectedGraphics];
    [self.graphicsController setSelectedObjects:theSelectedGraphics];
}
```

# KVOを使い易くした
# クラス

# MAKVONotificationCenter

# MAKVONotificationCenter

```
// NSAppのmainWindowに変更があったら
// selfのupdateFloatingTypeメソッドを呼んでね。

    [NSApp addObserver:self
            forKeyPath:@"mainWindow"
              selector:@selector(updateFloatingType)
              userInfo:nil
               options:0];
```

```
// NSAppのmainWindowに変更があったら
// selfのupdateFloatingTypeメソッドを呼んでと頼んだけど
// あれはもうやめて。

    [NSApp removeObserver:self
                  keyPath:@"mainWindow"
                 selector:@selector(updateFloatingType)];
```