

## 例：Filter Workbench

Filter Workbench は、イメージやその他のビジュアルコンテンツにさまざまなフィルタを適用し、同じ効果を ActionScript で生成するために使用できる結果コードを表示するユーザインターフェイスを備えています。このアプリケーションは、さまざまなフィルタを試すためのツールを提供するだけでなく、次の操作も行います。

- さまざまなフィルタのインスタンスの作成
- 表示オブジェクトに対する複数のフィルタの適用

このサンプルのアプリケーションファイルを取得するには、[www.adobe.com/go/learn\\_programmingAS3samples\\_flash\\_jp](http://www.adobe.com/go/learn_programmingAS3samples_flash_jp) を参照してください。Filter Workbench アプリケーションファイルは Samples¥FilterWorkbench フォルダにあります。このアプリケーションは次のファイルで構成されています。

ファイル	説明
com¥example¥programmingas3¥filterWorkbench¥FilterWorkbenchController.as	フィルタを適用するコンテンツの切り替え、コンテンツへのフィルタの適用など、アプリケーションの主な機能を提供するクラス。
com¥example¥programmingas3¥filterWorkbench¥FilterFactory.as	各フィルタファクトリクラスによって実装される一般的なメソッドを定義するインターフェイス。このインターフェイスは、FilterWorkbenchController クラスが個々のフィルタファクトリクラスとやり取りするときに使用する一般的な機能を定義します。

ファイル	説明
com¥example¥programmingas3¥filterWorkbench¥フォルダ : BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	それぞれが IFilterFactory インターフェイスを実装するクラスのセット。これらの各クラスは、1種類のフィルタの値を作成し、設定するための機能を提供します。アプリケーションのフィルタプロパティパネルは、これらのファクトリクラスを使用して特定のフィルタのインスタンスを作成し、FilterWorkbenchController クラスがこのインスタンスを取得して、イメージコンテンツに適用します。
com¥example¥programmingas3¥filterWorkbench¥ColorStringFormatter.as	数値で表したカラー値を 16 進数の String 形式に変換するメソッドが含まれるユーティリティクラス。
com¥example¥programmingas3¥filterWorkbench¥GradientColor.as	値オブジェクトとして機能し、GradientBevelFilter と GradientGlowFilter の各カラーに関連付けられた 3 つの値 (カラー、アルファ値、比率) を 1 つのオブジェクトに結合するクラス。
<b>ユーザインターフェイス (Flash)</b>	
FilterWorkbench fla	アプリケーションのユーザインターフェイスを定義するメインファイル。
flashapp¥FilterWorkbench.as	メインアプリケーションのユーザインターフェイスの機能を提供するクラス。このクラスは、アプリケーション FLA ファイルのドキュメントクラスとして使用されます。
flashapp¥filterPanels フォルダ : BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	1 つのフィルタのオプションを設定するために使用する各パネルの機能を提供するクラスのセット。各クラスには、メインアプリケーション FLA ファイルのライブラリ内に、名前がクラスの名前に一致する関連付けられた MovieClip シンボルもあります (例えば、シンボル「BlurPanel」は BlurPanel.as で定義されたクラスにリンクされています)。ユーザインターフェイスを構成するコンポーネントは、これらのシンボル内部に配置され、名前が指定されます。
flashapp¥ImageContainer.as	画面上に読み込まれたイメージのコンテナとして機能する表示オブジェクト。
flashapp¥ButtonCellRenderer.as	DataGrid コンポーネントのセルにボタンコンポーネントを含めるために使用するカスタムセルレンダラー。

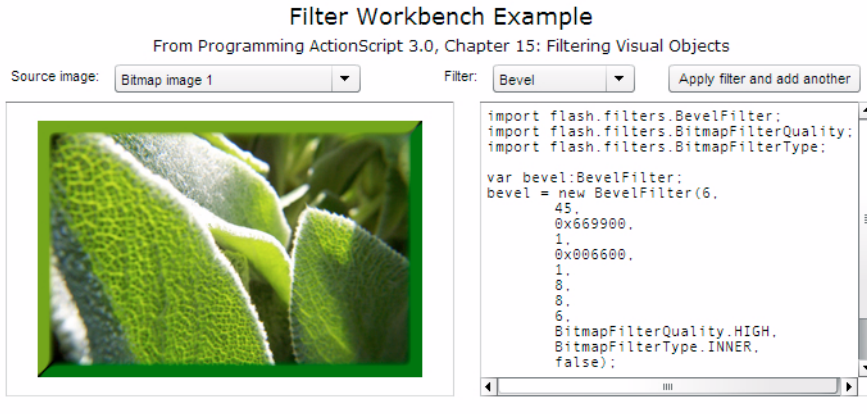
ファイル	説明
フィルタされたイメージコンテンツ	
com¥example¥programmingas3¥filterWorkbench¥ImageType.as	このクラスは、アプリケーションがフィルタを読み込んで適用できる1つのイメージファイルの種類とURLが含まれる値オブジェクトとして機能します。このクラスには、使用可能な実際のイメージファイルを表す定数のセットも含まれます。
images¥sampleAnimation.swf、 images¥sampleImage1.jpg、 images¥sampleImage2.jpg	アプリケーションでフィルタが適用されるイメージやその他のビジュアルコンテンツ。

## さまざまな ActionScript フィルタの使用

Filter Workbench アプリケーションは、さまざまなフィルタ効果を試し、その効果に関連する ActionScript コードを生成する際に役立つように設計されています。このアプリケーションを使用すると、ビットマップイメージや Flash アニメーションなどのビジュアルコンテンツが含まれる3つの異なるフィルタから選択し、選択したイメージに8つの異なる ActionScript フィルタを個別に、または他のフィルタと組み合わせて適用できます。このアプリケーションには次のフィルタが含まれます。

- ベベル (flash.filters.BevelFilter)
- ぼかし (flash.filters.BlurFilter)
- カラーマトリックス (flash.filters.ColorMatrixFilter)
- 畳み込み (flash.filters.ConvolutionFilter)
- ドロップシャドウ (flash.filters.DropShadowFilter)
- グロー (flash.filters.GlowFilter)
- グラデーションベベル (flash.filters.GradientBevelFilter)
- グラデーショングロー (flash.filters.GradientGlowFilter)

イメージと、そのイメージに適用するフィルタを選択すると、選択したフィルタに固有のプロパティを設定するためのコントロールが含まれるパネルが表示されます。例えば、次のイメージは、ベベルフィルタが選択されたアプリケーションを示しています。



### Filter Properties

Blur X: <input type="text" value="8"/>	Highlight color: <input type="color" value="#669900"/>	Angle: <input type="text" value="45"/>
Blur Y: <input type="text" value="8"/>	Highlight alpha: <input type="text" value="1"/>	Distance: <input type="text" value="6"/>
Strength: <input type="text" value="6"/>	Shadow color: <input type="color" value="#006600"/>	Knockout: <input type="checkbox"/>
Quality: <input type="text" value="High"/>	Shadow alpha: <input type="text" value="1"/>	Type: <input type="text" value="Inner"/>

フィルタプロパティを調整すると、プレビューがリアルタイムで更新されます。1つのフィルタをカスタマイズして「適用」ボタンをクリックし、別のフィルタをカスタマイズして「適用」ボタンをクリックすることにより、複数のフィルタを適用することもできます。

アプリケーションのフィルタパネルにはいくつかの機能と制限があります。

- カラーマトリックスフィルタには、明度、コントラスト、彩度、色相など、イメージの一般的なプロパティを直接操作するためのコントロールのセットが含まれます。さらに、カスタムカラーマトリックス値を指定できます。
- ActionScript を使用した場合にのみ利用できる畳み込みフィルタには、頻繁に使用される畳み込みマトリックス値が含まれます。カスタム値を指定することもできます。ただし、ConvolutionFilter クラスには任意のサイズのマトリックスを指定できますが、Filter Workbench アプリケーションは、最も一般的に使用されるフィルタサイズである固定の 3x3 マトリックスを使用します。

- 置き換えマップフィルタは ActionScript だけで利用可能であり、Filter Workbench アプリケーションでは使用できません。置き換えマップフィルタには、フィルタされたイメージコンテンツに加えてマップイメージも必要です。マップイメージはフィルタの結果を左右する主要な入力なので、マップイメージを読み込みまたは作成する機能がない場合は、置き換えマップフィルタを試す能力は大幅に制限されます。

## フィルタインスタンスの作成

Filter Workbench アプリケーションには、フィルタを作成するために各パネルによって使用されるクラスのセットが、使用可能なフィルタごとに1つずつ含まれます。フィルタを選択すると、そのフィルタパネルに関連付けられた ActionScript コードが、適切なフィルタファクトリクラスのインスタンスを作成します（これらのクラスは、その目的が他のオブジェクトのインスタンスを作成することであり、実際の工場が個別の製品を製造することと類似しているため、「ファクトリクラス」と呼ばれます）。

パネルのプロパティ値を変更すると、パネルのコードがファクトリクラスの適切なメソッドを呼び出します。各ファクトリクラスには、パネルが適切なフィルタインスタンスの作成に使用する固有のメソッドが含まれます。例えば、ぼかしフィルタを選択すると BlurFactory インスタンスが作成されます。BlurFactory クラスには、希望の BlurFilter インスタンスを作成するために合わせて使用される3つのパラメータ blurX、blurY、quality を受け入れる modifyFilter() メソッドが含まれます。

```
private var _filter:BlurFilter;

public function modifyFilter(blurX:Number = 4, blurY:Number = 4,
    quality:int = 1):void
{
    _filter = new BlurFilter(blurX, blurY, quality);
    dispatchEvent(new Event(Event.CHANGE));
}
```

これに対して、畳み込みフィルタを選択した場合は、そのフィルタによって柔軟性が大幅に向上し、その結果、制御するプロパティが増えます。ConvolutionFactory クラスでは、フィルタパネルで別の値を選択すると、次のコードが呼び出されます。

```
private var _filter:ConvolutionFilter;

public function modifyFilter(matrixX:Number = 0,
    matrixY:Number = 0,
    matrix:Array = null,
    divisor:Number = 1.0,
    bias:Number = 0.0,
    preserveAlpha:Boolean = true,
    clamp:Boolean = true,
    color:uint = 0,
    alpha:Number = 0.0):void
{
```

```

        _filter = new ConvolutionFilter(matrixX, matrixY, matrix, divisor, bias,
        preserveAlpha, clamp, color, alpha);
        dispatchEvent(new Event(Event.CHANGE));
    }
}

```

いずれの場合でも、フィルタ値が変化すると、ファクトリオブジェクトが `Event.CHANGE` イベントを送出し、フィルタの値が変更されたことをリスナーに通知します。フィルタされたコンテンツに実際にフィルタを適用する `FilterWorkbenchController` クラスは、そのイベントを待機して、フィルタの新しいコピーを取得し、フィルタされたコンテンツにそのコピーを再度適用する必要がある時期を確認します。

`FilterWorkbenchController` クラスは、各フィルタファクトリクラスに固有の詳細情報を認識する必要はありません。フィルタが変更され、フィルタのコピーにアクセスできるようになったことだけを認識する必要があります。これをサポートするために、アプリケーションの `FilterWorkbenchController` インスタンスがそのジョブを実行できるように、フィルタファクトリクラスが提供する必要がある動作を定義するインターフェイス、`IFilterFactory` がアプリケーションに含まれます。`IFilterFactory` は、`FilterWorkbenchController` クラスで使用される `getFilter()` メソッドを定義します。

```
function getFilter():BitmapFilter;
```

`getFilter()` インターフェイスメソッドの定義では、特定の種類のフィルタではなく、`BitmapFilter` インスタンスを返すことを指定しています。`BitmapFilter` クラスは、特定の種類のフィルタを定義するものではなく、すべてのフィルタクラスを構築する際の基礎となるクラスです。各フィルタファクトリクラスは、`getFilter()` メソッドの特定の実装を定義し、このメソッドで、構築したフィルタオブジェクトへの参照を返します。例えば、`ConvolutionFactory` クラスの簡略化したソースコードを次に示します。

```

public class ConvolutionFactory extends EventDispatcher implements
    IFilterFactory
{
    // ----- プライベート変数 -----
    private var _filter:ConvolutionFilter;
    ...
    // ----- IFilterFactory 実装 -----
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}

```

`ConvolutionFactory` クラスによる `getFilter()` メソッドの実装では `ConvolutionFilter` インスタンスが返されますが、`getFilter()` を呼び出すどのオブジェクトも、必ずしもそれを認識していません。`ConvolutionFactory` が従う `getFilter()` メソッドの定義に応じて、いずれかの `ActionScript` フィルタクラスのインスタンスである任意の `BitmapFilter` インスタンスを返す必要があります。

## 表示オブジェクトに対するフィルタの適用

前の節で説明したように、Filter Workbench アプリケーションは FilterWorkbenchController クラスのインスタンス（以降は「コントローラインスタンス」と呼ぶ）を使用します。このインスタンスが、選択したビジュアルオブジェクトに対するフィルタの適用という実際のタスクを実行します。コントローラインスタンスがフィルタを適用するには、その前に、フィルタの適用先のイメージまたはビジュアルコンテンツを認識しておく必要があります。イメージを選択すると、アプリケーションは FilterWorkbenchController クラスの setFilterTarget() メソッドを呼び出し、ImageType クラスで定義された定数の1つで渡します。

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE,
        targetLoadComplete);
    ...
}
```

その情報を使用して、コントローラインスタンスは指定のファイルを読み込み、その後に \_currentTarget という名前のインスタンス変数に保存します。

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

フィルタを選択すると、アプリケーションはコントローラインスタンスの setFilter() メソッドを呼び出し、関連するフィルタファクトリオブジェクトに対する参照をコントローラに渡し、\_filterFactory という名前のインスタンス変数に保存します。

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

前述のように、コントローラインスタンスは、指定されたフィルタファクトリインスタンスの固有のデータ型を認識せず、オブジェクトが `IFilterFactory` インスタンスを実装することだけを認識します。つまり、`getFilter()` メソッドを持っており、フィルタが変更されたときに `change(Event.CHANGE)` イベントを送出するということです。

フィルタのプロパティをそのフィルタのパネルで変更すると、コントローラインスタンスが、フィルタファクトリの `change` イベントを通じてそのフィルタが変更されたことを検出し、コントローラインスタンスの `filterChange()` メソッドを呼び出します。そのメソッドは、次に `applyTemporaryFilter()` メソッドを呼び出します。

```
private function filterChange(event:Event):void
{
    applyTemporaryFilter();
}

private function applyTemporaryFilter():void
{
    var currentFilter:BitmapFilter = _filterFactory.getFilter();

    // 現在のフィルタをセットに一時的に追加する
    _currentFilters.push(currentFilter);

    // フィルタターゲットのフィルタセットを更新する
    _currentTarget.filters = _currentFilters;

    // 現在のフィルタをセットから削除する
    // (ターゲットがフィルタ配列のコピーを内部で使用するので、
    // フィルタはフィルタターゲットからは削除されない)
    _currentFilters.pop();
}
```

表示オブジェクトに対するフィルタの適用は `applyTemporaryFilter()` メソッド内部で行われます。最初に、コントローラがフィルタファクトリの `getFilter()` メソッドを呼び出して、フィルタオブジェクトに対する参照を取得します。

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

コントローラインスタンスには `_currentFilters` という名前の `Array` インスタンスがあり、ここには、表示オブジェクトに適用されたすべてのフィルタが保存されています。次に、新たに更新されたフィルタを配列に追加します。

```
_currentFilters.push(currentFilter);
```

次に、このコードがフィルタの配列を表示オブジェクトの `filters` プロパティに割り当て、このプロパティが実際にフィルタをイメージに適用します。

```
_currentTarget.filters = _currentFilters;
```

最後に、この最後に追加したフィルタはこの時点でも「稼働中」のフィルタなので、表示オブジェクトに永続的に適用してはなりません。したがって、`_currentFilters` 配列から削除されます。

```
_currentFilters.pop();
```

このフィルタを配列から削除しても、フィルタされた表示オブジェクトには影響はありません。これは、表示オブジェクトが、`filters` プロパティに適用されたときにフィルタ配列のコピーを作成し、元の配列ではなく、その内部配列を使用するからです。したがって、フィルタの配列に対して行われたすべての変更は、配列が表示オブジェクトの `filters` プロパティに再び割り当てられるまでは、表示オブジェクトに影響を与えません。

