

Away3D Basics 6 – The Color Materials

本　　ド　　キ　　ュ　　メ　　ン　　ト　　は　　、
http://www.flashmagazine.com/Tutorials/detail/away3d_basics_6_-_materials_and_light_part_1/で公開されている「Away3D Basics 6 – The Color Materials」を、ヒム・カンパニー 永井勝則が自主的に日本語に訳したものです。

<http://www.himco.jp/>

knagai@himco.jp

(2009/11)

原文は、

http://www.flashmagazine.com/Tutorials/detail/away3d_basics_6_-_materials_and_light_part_1/

で読むことができます。



Away3D の基本6: カラーマテリアル

われわれはすでにこれまでのチュートリアルの中でマテリアルを多少は扱ってきました。本チュートリアルでは基本的なカラーマテリアルと Phong シェーダ、ライトの使用方法を探っていきます。

カラーやワイヤーフレームマテリアルを追加すると、3次元的な実感が増します。本チュートリアルでは、Away3D のカラーマテリアルの基本的な扱い方を見ていきます。

必要とされる予備知識

本チュートリアルはわれわれの[Away3D チュートリアル](#)上に構築されています。Flash 3D が初めての方は本チュートリアル以前のチュートリアルに一通り目を通されたほうがよいでしょう。各サンプルには完全なソースファイルをつけています。付随する ActionScript ファイルへのリンクをクリックするとそのサンプルのクラスファイルがダウンロードできます。また多くのサンプルでは Cover.as ファイルが必要です。これは多くの Flash ファイルを一度に表示しようとする際に発生する可能性のあるマシンのフリーズを避けるために使用します (Flash 3D は多くのマシンリソースを消費するので、一度に多くの 3D を再生するとマシンが固まる恐れがあります)。サンプルのクラスファイルの使用方法がよく分からないという方は、[このチュートリアル](#)にまず目を通してください。

カラーマテリアル

Away3D のすべての 3D オブジェクトはデフォルトで、黒いワイヤーフレームを持つ、ランダムに選択されるカラーマテリアルで構成される標準的なマテリアルを備えています。これまでのほとんどのチュートリアルで使用してきたのはこのマテリアルです。したがってチュートリアルを再ロードするとそのたびに、3D モデルは前と異なるカラーで表示されます。このマテリアルは WireColorMaterial と呼ばれます。

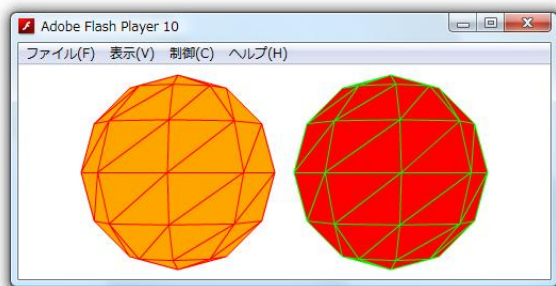
Away3D のすべてのオブジェクトと同様、これもコンパクトな方法と冗長な方法で記述できます。コンパクトな方法では、オブジェクトの作成時、初期化オブジェクトにカラーを渡します。

```
var sphere:Sphere = new Sphere({material:"orange#red"});
view.scene.addChild(sphere);
```

これは実にまとまった書き方ですが、プログラマーの中には、Away3D を新たに始めようとする人々にはコードが読みにくくなるという理由で、この“魔法のように自動的な”方法を嫌う人もいます。冗長なシンタックスを使用するには、次のようにカラーマテリアルを作成して設定し、それを球に追加します。

```
var wireColorMaterial:WireColorMaterial = new WireColorMaterial();
wireColorMaterial.color = 0xFFA500;
wireColorMaterial.wirecolor = 0xFF0000;
var sphere:Sphere = new Sphere();
sphere.material = wireColorMaterial;
view.scene.addChild(sphere);
```

この書き方ではコードがかなり長くなりますが、初期化オブジェクトを知らない場合でも容易に読むことができます。自分に合ったコーディングスタイルで記述してください。



ムービー: [Basic09_WireColorMaterial.as](#)

Basic09_WireColorMaterial.as (訳者注: 以下のコードは、ダウンロードファイルと少し変えています)

```
package {

    import away3d.containers.View3D;
    import away3d.materials.WireColorMaterial;
    import away3d.primitives.Sphere;
```

```

import flash.display.Sprite;

[SWF(width="500", height="200",
      frameRate="60", backgroundColor="#FFFFFF")]

public class Basic09_WireColorMaterial extends Sprite {

    public function Basic09_WireColorMaterial() {

        var view:View3D = new View3D({x:250,y:100});
        addChild(view);

        // 初期化オブジェクトを使ったコンパクトな方法
        // マテリアルのカラーはオレンジ、ボーダーワイヤーのカラーは赤
        var sphere1:Sphere = new Sphere(
            {material:"orange#red",x:110});
        view.scene.addChild(sphere1);

        // 冗長なシンタックスを使う方法
        var colorMaterial:WireColorMaterial =
            new WireColorMaterial();
        // マテリアルのカラーは赤
        colorMaterial.color = 0xFF0000;
        // ボーダーワイヤーのカラーは緑
        colorMaterial.wirecolor = 0x00FF00;
        var sphere2:Sphere = new Sphere();
        sphere2.x = -110;
        sphere2.material = colorMaterial;
        view.scene.addChild(sphere2);

        // ビューをレンダリング
        view.render();

    }

}

```

また WireColorMaterial では、ワイヤーの幅の設定もサポートされています。

```
wireColorMaterial.width = 5;
```

これによって、すべてのワイヤーの幅にこの値が設定されます。

純粋なカラーマテリアル(つまりワイヤーがない)を指定することもできます。それにはマテリアルを指定するときに、#文字を使わずマテリアルのカラーのみを指定します。

```
var sphere:Sphere = new Sphere({material:"orange"});
```

冗長な方法では、ColorMaterial を次のように使用します。

```
var colorMaterial:ColorMaterial = new ColorMaterial();  
colorMaterial.color = 0xFFA500;  
var sphere:Sphere = new Sphere();  
sphere.material = colorMaterial;
```



ムービー: [Basic09_ColorMaterial.as](#)

Basic09_ColorMaterial.as

```
package {  
  
    import away3d.containers.View3D;  
    import away3d.materials.ColorMaterial;  
    import away3d.primitives.Sphere;  
  
    import flash.display.Sprite;
```

```

[SWF(width="500", height="200",
      frameRate="60", backgroundColor="#FFFFFF")]
public class Basic09_ColorMaterial extends Sprite {

    public function Basic09_ColorMaterial() {
        var view:View3D = new View3D({x:250,y:100});
        addChild(view);

        // コンパクトな方法
        // マテリアルのカラーはオレンジ
        var sphere1:Sphere = new Sphere({material:"orange",x:110});
        view.scene.addChild(sphere1);

        // ColorMaterial を使った冗長な方法
        var colorMaterial:ColorMaterial = new ColorMaterial();
        // マテリアルのカラーは赤
        colorMaterial.color = 0xFF0000;
        var sphere2:Sphere = new Sphere();
        sphere2.x = -110;
        sphere2.material = colorMaterial;
        view.scene.addChild(sphere2);

        view.render();
    }
}

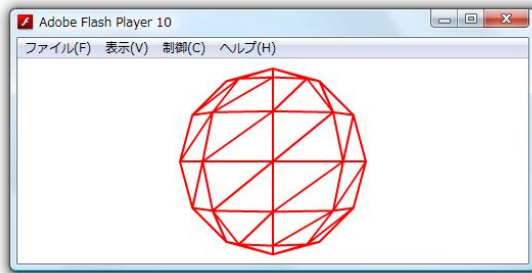
```

WireframeMaterial を使うと、これとは逆にワイヤーだけを表示することができます。

```

var colorMaterial:WireframeMaterial = new WireframeMaterial();
colorMaterial.color = 0xFFA500;
colorMaterial.width = 2;
var sphere:Sphere = new Sphere();
sphere.material = colorMaterial;
view.scene.addChild(sphere);

```



ムービー: [Basic09_WireframeMaterial.as](#)

このマテリアルには、これまで見てきたコンパクトなシンタックスでは記述できません。

Basic09_WireframeMaterial

```
package {

    import away3d.containers.View3D;
    import away3d.materials.WireframeMaterial;
    import away3d.primitives.Sphere;

    import flash.display.Sprite;

    [SWF(width="500", height="200", frameRate="60", backgroundColor="#ffffff")]
    public class Basic09_WireframeMaterial extends Sprite{

        public function Basic09_WireframeMaterial() {
            var view:View3D = new View3D({x:250,y:100});
            addChild(view);

            // WireframeMaterial を使用した方法
            var colorMaterial:WireframeMaterial =
                new WireframeMaterial();
            // ワイヤーのカラーは赤
            colorMaterial.color = 0xff0000
            // ワイヤーの幅は 2
            colorMaterial.width = 2;
        }
    }
}
```

```
var sphere:Sphere = new Sphere();
sphere.material = colorMaterial;
view.scene.addChild(sphere);

view.render();
}
}
}
```

これらのマテリアルはテストには便利ですが、このままではあまり使用されず、ライトやシェードを追加することで、驚くような効果を発揮します。

カラーマテリアルだけを使ってオブジェクトを設定しても、そのオブジェクトがどのように回転するのかは実際には分かりづらいですが、アウトライン (outline) を追加すると使いやすくなります。アウトラインはすべてのマテリアルに追加できます。そのためには WireframeMaterial を次のように使用します。

```
var outlineMaterial:WireframeMaterial = new WireframeMaterial(0x000000);
outlineMaterial.width = 3;
sphere.outline = outlineMaterial;
```

これによって球の外側のエッジが3ピクセルの幅のアウトラインで示されます。これを海がめに使用すると次のようになります。



ムービー: [Basic09_ColorMaterial2.as](#)

これは正確には“カートゥーンシェーダ”ではありませんが(線がちかちかしている点に注意してください)、技術的な説明図が欲しい場合には便利です。この方法によって 3D モデルが 2D のイラストのように見えます。

Basic09_ColorMaterial2

```
package {

    import away3d.containers.View3D;
    import away3d.materials.WireframeMaterial;
    import away3d.primitives.SeaTurtle;

    import flash.display.Sprite;
    import flash.events.Event;

    [SWF(width="500", height="200",
         frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic09_ColorMaterial2 extends Sprite {

        private var view:View3D;
        private var turtle:SeaTurtle;

        public function Basic09_ColorMaterial2() {

            view = new View3D({x:250,y:100});
            addChild(view);

            //コンパクトな方法
            turtle = new SeaTurtle({material:"orange"});
            turtle.scale(0.3);
            // WireframeMaterial を黒で作成
            var outlineMaterial:WireframeMaterial =
                new WireframeMaterial(0x000000);
            outlineMaterial.width = 3;
            turtle.outline = outlineMaterial;
            view.scene.addChild(turtle);

            this.addEventListener(Event.ENTER_FRAME,refresh);
        }

        private function refresh(e:Event):void {
```

```
        turtle.rotationX += .45;
        turtle.rotationY += 1;
        view.render();
    }
}
}
```

ライトと反射するカラーマテリアル

シェーダは、3D オブジェクトのマテリアルをレンダリングするとき、ライトを計算に入れます。ライトを必要とするマテリアルを使用しているとき、光源の追加を忘れると、モデルはレンダリングされません。Away3D では3種類の光源がありますが、複数のマテリアルで動作するには DirectionalLight3D (太陽光のように遠くにあって減衰しない光、平行光源)だけです。

```
var light:DirectionalLight3D = new DirectionalLight3D();
view.scene.addChild(light);
```

光源を追加すると、オブジェクトが見えるようになりますが、それは表示したい結果ではありません。なぜなら、新しいライトはワールドの中心 ((0, 0, 0)の位置)に作成されるからです。良好な結果を得るには、ライトをカメラに向かって移動させ、位置を調整する必要があります。

```
light.y = 500;
light.x = -300;
light.z = -200;
```

これは適当に指定しただけの値ですが、反射はこの位置と、平行光源の3つの主要設定 (ambient、diffuse、specular)にもとづいて変わります。ambient、diffuse、specular はどれも 0 から 1 の間の値を取ります。以下はそれぞれのデフォルト値です。

```
// 反射光の強さの係数を定義
light.specular = 1;
//拡散光の強さの係数を定義
light.diffuse = 0.5;
//環境光の強さの係数を定義
light.ambient = 0.5;
```

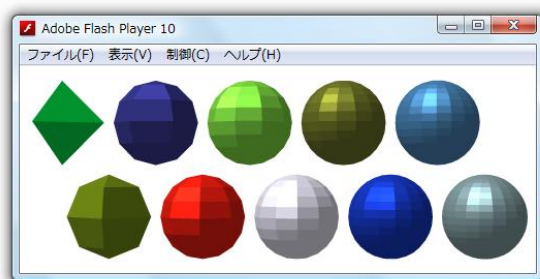
NOTE: ライトには brightness プロパティもあります。これは上記値の乗数で、ライトが初めてレンダリングされたら、もう設定できません。したがって、ライトの作成時に必ず設定するようにします。

ShadingColorMaterial

この材料はシーン内のライトの設定に効果的です。すべてのフェースが等しい明るさなので、ライトがモデルにどのように影響するかを調べることができます。カラーは、初期化オブジェクトかこの材料の color プロパティを使って設定します。

```
var mat:ShadingColorMaterial = new ShadingColorMaterial(0xff00ff);
```

下のサンプルでは球をクリックすると、そのカラーがランダムに変化します。



ムービー: [Basic09_ShadingColorMaterial.as](#)

Basic09_ShadingColorMaterial.as

```
package {  
  
    import away3d.containers.View3D;  
    import away3d.events.MouseEvent3D;  
    import away3d.lights.DirectionLight3D;  
    import away3d.materials.ShadingColorMaterial;  
    import away3d.primitives.Sphere;  
  
    import flash.display.Sprite;  
    import flash.events.Event;  
    import flash.events.MouseEvent;  
  
    [SWF(width="500",height="200", backgroundColor="#ffffff")]
```

```

public class Basic09_ShadingColorMaterial extends Sprite {

    private var view:View3D;
    private var sp:Sphere;
    private var mat:ShadingColorMaterial;
    private var changed:Boolean = true;

    public function Basic09_ShadingColorMaterial() {

        view = new View3D();
        view.x = 250;
        view.y = 100;
        addChild(view);

        var l:DirectionalLight3D = new DirectionalLight3D();
        view.scene.addChild(l);
        l.y = 500;
        l.x = -300;
        l.z = -200;

        // 球を 10 個、順に複雑性を増すように作成
        for(var i:uint = 1; i<11; i++){
            sp = new Sphere(
                {radius:45,segmentsW:i*2,segmentsH:i*2});
            // 数学的な手法を使った配置
            sp.x = (i*50)-280;
            i%2 == 0 ? sp.y = -50 : sp.y = 50;
            sp.material = new ShadingColorMaterial(0xffffffff);
            view.scene.addChild(sp);
            sp.addEventListener(
                MouseEvent3D.MOUSE_DOWN,setColor);
        }
        this.addEventListener(Event.ENTER_FRAME,update);
    }

    private function update(e:Event):void {

```



```
import away3d.containers.View3D;
import away3d.lights.DirectionLight3D;
import away3d.materials.PhongColorMaterial;
import away3d.primitives.Sphere;

import flash.display.Sprite;

[SWF(width="500", height="200",
    frameRate="60", backgroundColor="#FFFFFF")]

public class Basic09_PhongColorMaterial extends Sprite {
    public function Basic09_PhongColorMaterial() {

        var view:View3D = new View3D({x:250,y:100});
        addChild(view);

        // PhongColorMaterial を作成し、球のマテリアルに設定
        var colorMaterial:PhongColorMaterial =
            new PhongColorMaterial(0xFFA500);
        var sphere:Sphere = new Sphere();
        sphere.material = colorMaterial;
        view.scene.addChild(sphere);

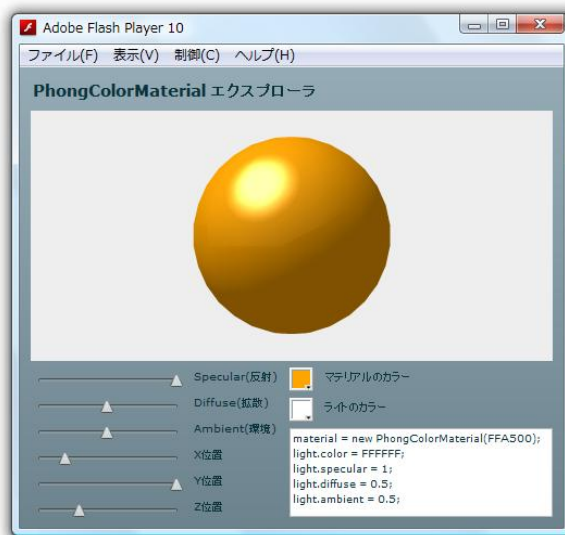
        // 光源が必要
        var light:DirectionLight3D = new DirectionLight3D();
        view.scene.addChild(light);

        // 反射を見るために、ライトをデフォルトの(0, 0, 0)から動かす
        light.y = 500;
        light.x = -300;
        light.z = -200;

        // ビューをレンダリング
        view.render();
    }
}
```

```
}  
}
```

ライトはそれぞれ、それ自体のカラーを持つこともできます。ライトのカラーはデフォルトで純粋な白に設定されます。ライトのカラーとマテリアルのカラーはミックスされるので、たとえば青の球に黄のライトを追加すると、結果として球は緑に見えます。下の Flex アプリケーションでは、ライトやマテリアルの設定を変えてその結果を確認することができます。またその組み合わせをコピーし、みなさんのプロジェクトにペーストして利用できます。



コード: [DirectionalLight3DTest.mxml](#)、[Viewport.mxml](#)

訳者注:

上記のMXMLファイルは訳者の環境では動作しなかったため、以下のViewport.mxmlでは太字部分に修正しています。

Viewport.mxml

```
<?xml version="1.0" encoding="utf-8"?>  
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="400" height="300"  
creationComplete="init()">  
    <mx:Script>  
        <![CDATA[  
                import away3d.core.clip.RectangleClipping;  
                import away3d.containers.View3D;  
                import away3d.primitives.Sphere;
```

```

import mx.core.UIComponent;

private var UIRef:UIComponent;

[Bindable]
public var view:View3D;

public function init():void
{
    // Flex 用の土台
    UIRef = new UIComponent();
    addChild(UIRef);

    // Canvas から値を取得
    var w:Number = myCanvas.width/2;
    var h:Number = myCanvas.height/2;

    // ビューポートを作成
    view = new View3D({x:w,y:h});
    view.clipping = new RectangleClipping(
        {minX:-w,minY:-h,maxX:w,maxY:h});
    UIRef.addChild(view);
}

]]>
</mx:Script>
<mx:Canvas id="myCanvas" width="100%" height="100%"
background-color="0xeeeeee">
</mx:Canvas>
</mx:Canvas>

```

DirectionalLight3DTest.mxml

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" xmlns:ns1="*" width="500"
height="425" applicationComplete="init0">
    <mx:Script>
        <![CDATA[

```

```
import away3d.materials.PhongColorMaterial;
import away3d.lights.DirectionallLight3D;
import away3d.primitives.Plane;
import mx.core.UIComponent;
import away3d.materials.ColorMaterial;
import away3d.materials.MovieMaterial;
import away3d.primitives.Sphere;

[Bindable]

public var sphere:Sphere;
private var light:DirectionallLight3D;
private var materialType:Number;

// 変化を表示する 3D ステージは、
// Viewport コンポーネントの Away3D として参照する

private function init():void
{

    // 球を作成し、3D ステージ(に追加
    sphere = new Sphere({segmentsW:15,segmentsH:15});
    Away3D.view.scene.addChild(sphere);

    // マテリアルを作成し球に設定して描画
    setMaterial();

    // 表示にはライトが必要
    light = new DirectionallLight3D();
    Away3D.view.scene.addChild(light);

    // ライトを移動
    light.y = 500;
    light.x = -300;
    light.z = -200;

    // ライトのデフォルトのプロパティ値を使って、Flex のスライダーを更新
```

```

        specularSlider.value = light.specular;
        diffuseSlider.value = light.diffuse;
        ambientSlider.value = light.ambient;
        xSlider.value = light.x;
        ySlider.value = light.y;
        zSlider.value = light.z;

        setResult();

        // ビューをレンダリング
        Away3D.view.render();
    }

    private function setMaterialType():void
    {
        trace("setMaterialType "+colorPicker.selectedColor);
        var colorMaterial:PhongColorMaterial =
            new PhongColorMaterial(colorPicker.selectedColor);
        sphere.material = colorMaterial;
        setResult();
        Away3D.view.render();
    }

    private function setMaterial():void
    {
        trace("setMaterial "+colorPicker.selectedColor);
        var colorMaterial:PhongColorMaterial =
            new PhongColorMaterial(colorPicker.selectedColor);
        sphere.material = colorMaterial;
        setResult();
        Away3D.view.render();
    }

    private function setLight():void
    {
        trace("setLight "+lightPicker.selectedColor);
        light.color = int(lightPicker.value);
        setResult();
        Away3D.view.forceUpdate = true;
    }

```

```

        Away3D.view.render();
    }
    private function refresh():void
    {
        light.ambient = ambientSlider.value;
        light.specular = specularSlider.value;
        light.diffuse = diffuseSlider.value;

        light.x = xSlider.value;
        light.y = ySlider.value;
        light.z = zSlider.value;

        setResult();

        Away3D.view.forceUpdate = true;
        Away3D.view.render();
    }
    private function setResult():void
    {
        var rez:String = "";
        rez += "material =
new PhongColorMaterial("+makeHex(colorPicker.selectedColor)+");\n";
        rez += "light.color = "+makeHex(lightPicker.selectedColor)+";\n";
        rez += "light.specular = "+specularSlider.value+";\n";
        rez += "light.diffuse = "+diffuseSlider.value+";\n";
        rez += "light.ambient = "+ambientSlider.value+";";
        result.text = rez;
    }
    public function makeHex(n:uint):String
    {
        var str:String = n.toString(16).toUpperCase();
        while(str.length < 6)
        {
            str = "0" + str;
        }
        return str;
    }

```

```

    }

    ]]>
</mx:Script>

<mx:VBox top="10" bottom="10" left="10" right="10" width="100%" height="100%">
    <mx:Label text="PhongColorMaterial エクスプローラ" fontWeight="bold" fontSize="15"/>
    <ns1:Viewport x="10" y="10" id="Away3D" height="231" width="480">
    </ns1:Viewport>
    <mx:Grid width="100%">
        <mx:GridRow width="100%" height="100%">
            <mx:GridItem width="230" height="100%">
                <mx:VBox width="100%" height="100%">
                    <mx:HBox width="100%">
                        <mx:HSlider minimum="0"
maximum="1" liveDragging="true" id="specularSlider" change="refresh0" width="140"/>
                        <mx:Label text="Specular(反射)"/>
                    </mx:HBox>
                    <mx:HBox width="100%">
                        <mx:HSlider minimum="0"
maximum="1" liveDragging="true" id="diffuseSlider" change="refresh0" width="140"/>
                        <mx:Label text="Diffuse(拡散)"/>
                    </mx:HBox>
                    <mx:HBox width="100%">
                        <mx:HSlider minimum="0"
maximum="1" liveDragging="true" id="ambientSlider" change="refresh0" width="140"/>
                        <mx:Label text="Ambient(環境)"/>
                    </mx:HBox>
                    <mx:HBox width="100%">
                        <mx:HSlider minimum="-500"
maximum="500" liveDragging="true" id="xSlider" change="refresh0" width="140"/>
                        <mx:Label text="X 位置"/>
                    </mx:HBox>
                    <mx:HBox width="100%">
                        <mx:HSlider minimum="-500"
maximum="500" liveDragging="true" id="ySlider" change="refresh0" width="140"/>
                        <mx:Label text="Y 位置"/>
                    </mx:HBox>
                </mx:VBox>
            </mx:GridItem>
        </mx:GridRow>
    </mx:Grid>
</mx:VBox>

```

```

        </mx:HBox>
        <mx:HBox width="100%">
            <mx:HSlider minimum="-500"
maximum="500" liveDragging="true" id="zSlider" change="refresh0" width="140"/>
            <mx:Label text="Z 位置"/>
        </mx:HBox>
    </mx:VBox>
</mx:GridItem>
<mx:GridItem width="100%" height="100%">
    <mx:VBox width="100%" height="100%">
        <mx:HBox width="100%">
            <mx:ColorPicker
selectedColor="0xFFFA500" id="colorPicker" change="setMaterial0"/>
            <mx:Label text="マテリアルのカラー"/>
        </mx:HBox>
        <mx:HBox width="100%">
            <mx:ColorPicker
selectedColor="0xFFFFFFFF" id="lightPicker" change="setLight0"/>
            <mx:Label text="ライトのカラー"/>
        </mx:HBox>
        <mx:TextArea width="100%" height="100%"
id="result" fontSize="9"/>
    </mx:VBox>
</mx:GridItem>
</mx:GridRow>
</mx:Grid>
</mx:VBox>
<mx:TextArea x="270" y="246" width="220" height="167" id="code" visible="false"/>
</mx:Application>

```

訳者注:

MXML ファイルは Flex Builder を持っていないでも SWF にコンパイルできます。Flex SDK に備わっている mxmmlc コマンドをコマンドラインから使用します。また Away3D を使用するので、-source-path を使用して使用する Away3D クラスへのソースパスを指定します。たとえば次のようになります。

```
> mxmmlc -source-path+=C:\Users\himco\away3d\trunk\fp10\Away3D\src
```

DirectionalLight3DTest.xml