

## Away3D Basics 6 – Bitmap materials

本　　ド　　キ　　ュ　　メ　　ン　　ト　　は　　、  
[http://www.flashmagazine.com/Tutorials/detail/away3d\\_basics\\_6\\_-\\_bitmap\\_materials/](http://www.flashmagazine.com/Tutorials/detail/away3d_basics_6_-_bitmap_materials/)で  
公開されている「Away3D Basics 6 – Bitmap 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](mailto:knagai@himco.jp)

(2009/11)

原文は、

[http://www.flashmagazine.com/Tutorials/detail/away3d\\_basics\\_6\\_-\\_bitmap\\_materials/](http://www.flashmagazine.com/Tutorials/detail/away3d_basics_6_-_bitmap_materials/)  
で読むことができます。



## Away3D の基本7: ビットマップマテリアル

前のチュートリアルで探ったカラーとワイヤーフレームのマテリアルの外見は

### 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。

ビットマップマテリアルで何が実現できるのかというアイデアを得るため、テクスチャをランダムに使用する下のムービーをクリックしてみてください。



ムービー: [Basic09\\_BitmapMaterialExplorer.as](http://Basic09_BitmapMaterialExplorer.as)

Basic09\_BitmapMaterialExplorer

```
package {  
  
    import away3d.cameras HoverCamera3D;  
    import away3d.containers.View3D;  
    import away3d.core.render.Renderer;  
    import away3d.core.utils.Cast;  
    import away3d.lights.DirectionLight3D;  
    import away3d.materials.PhongBitmapMaterial;  
    import away3d.materials.TransformBitmapMaterial;  
    import away3d.materials.utils.SimpleShadow;  
    import away3d.primitives.Plane;  
    import away3d.primitives.Sphere;  
  
    import flash.display.Bitmap;  
    import flash.display.Sprite;  
    import flash.events.Event;  
    import flash.events.MouseEvent;  
    //import flash.utils.getDefinitionByName;  
  
    [SWF(width="500", height="300",  
        frameRate="60", backgroundColor="#000000")]  
    public class Basic09_BitmapMaterialExplorer extends Sprite {  
  
        // テクスチャに必要なイメージを埋め込む
```

```
[Embed(source="spiralgraphics/p1.jpg")]
public var texture1:Class;
[Embed(source="spiralgraphics/p2.jpg")]
public var texture2:Class;
[Embed(source="spiralgraphics/p3.jpg")]
public var texture3:Class;
[Embed(source="spiralgraphics/p4.jpg")]
private var texture4:Class;
[Embed(source="spiralgraphics/p5.jpg")]
private var texture5:Class;
[Embed(source="spiralgraphics/p6.jpg")]
private var texture6:Class;
[Embed(source="spiralgraphics/p7.jpg")]
private var texture7:Class;
[Embed(source="spiralgraphics/p8.jpg")]
private var texture8:Class;
[Embed(source="spiralgraphics/p9.jpg")]
private var texture9:Class;
[Embed(source="spiralgraphics/p10.jpg")]
private var texture10:Class;
[Embed(source="spiralgraphics/p11.jpg")]
private var texture11:Class;
[Embed(source="spiralgraphics/p12.jpg")]
private var texture12:Class;
[Embed(source="spiralgraphics/p13.jpg")]
private var texture13:Class;
[Embed(source="spiralgraphics/p14.jpg")]
private var texture14:Class;
[Embed(source="spiralgraphics/p15.jpg")]
private var texture15:Class;

private var view:View3D;
private var plane:Plane;
private var sphere:Sphere;
private var material1:PhongBitmapMaterial;
private var material2:TransformBitmapMaterial;
```

```

private var cam:HoverCamera3D;

private var textureNum:uint = 15;

public function Basic09_BitmapMaterialExplorer() {
    cam = new HoverCamera3D();

    view = new View3D({x:250,y:150,camera:cam});
    view.renderer = Renderer.CORRECT_Z_ORDER;
    addChild(view);

    plane = new Plane(
    {segmentsW:6, segmentsH:6, y:-120, width:1100, height:1100});
    view.scene.addChild(plane);

    sphere = new Sphere(
        {radius:100,segmentsW:14,segmentsH:8});
    sphere.rotationX = 85;
    view.scene.addChild(sphere);

    cam.targetpanangle = cam.panangle = 0;
    cam.targettiltangle = cam.tiltangle = 35;
    cam.zoom = 10;
    cam.lookAt( sphere.position );
    cam.target.z += 10;
    this.addEventListener(MouseEvent.CLICK,changeMaterial);

    var light:DirectionalLight3D = new DirectionalLight3D();
    light.y = 500;
    light.x = -50;
    light.z = -50;
    light.specular = 0.6;
    view.scene.addChild(light);

    // 簡単な影をつける
    var simpleshadow:SimpleShadow =

```

```

        new SimpleShadow(sphere,0x99333333,20,-95);
        simpleshadow.apply(view.scene);

        changeMaterial();
        this.addEventListener(Event.ENTER_FRAME,update);
    }

    private function update(e:Event):void    {
        cam.targetpanangle += .3;
        cam.hover();
        view.render();
    }

    // マウスのクリックで呼び出される
    private function changeMaterial(e:MouseEvent = null):void {
        // クラス名 (texture1 から texture15 まで)をランダムに選択
        var className1:String =
            "texture"+Math.floor( (Math.random()*textureNum)+1).toString();
        var className2:String =
            "texture"+Math.floor( (Math.random()*textureNum)+1).toString();

        // クラス名を使って
        // PhongBitmapMaterial と TransformBitmapMaterial を作成
        // 球用マテリアル
        material1 = new PhongBitmapMaterial(
            Cast.bitmap( new this[className1]() as Bitmap ) );
        // 平面用マテリアル
        material2 = new TransformBitmapMaterial(
            Cast.bitmap( new this[className2]() as Bitmap ),{repeat:true, scaleX:.75, scaleY:.75} );
        material2.smooth = true;
        sphere.material = material1;
        plane.material = material2;
        view.render();
    }
}
}
}

```

## ビットマップを使った基本的なテクスチャの貼り付け

一番簡単にスタートできるビットマップマテリアルはプレーンな `BitmapFileMaterial` です。これは URL からビットマップをロードし、ロードしたらその結果を表示します。しかし結果は自動的に表示されないため、`enterFrame` でビューのレンダリングを設定するか、イベントリスナーを設定する必要があります。まずは `enterFrame` の方法から見ていきましょう。

```
var earthMaterial:BitmapFileMaterial =
    new BitmapFileMaterial(
        "http://www.flashmagazine.com/articlefiles/away3d/resources/earthmap1k.jpg");
var sphere:Sphere = new Sphere({material:earthMaterial});
view.scene.addChild(sphere);
this.addEventListener(Event.ENTER_FRAME,update);
```

ここでは `BitmapFileMaterial` を作成し (`earthMaterial`)、これを `Sphere` のコンストラクタに渡して球 (`sphere`) を作成しています。マテリアルの設定はまた `sphere.material` プロパティを使っても行えます。それから球を 3D ステージに追加し、`enterFrame` ごとに `update()` メソッドを呼び出すリスナーの設定を行っています。

```
private function update(e:Event):void {
    view.render();
}
```

この完成ファイルはここ ([Basic09\\_BitmapFileMaterial.as](#)) からダウンロードできます。

### Basic09\_BitmapFileMaterial.as

```
package {

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

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

```
[SWF(width="500", height="200",
      frameRate="60", backgroundColor="#FFFFFF")]
public class Basic09_BitmapFileMaterial extends Sprite {

    private var view:View3D;

    public function Basic09_BitmapFileMaterial() {
        view = new View3D({x:250,y:100});
        addChild(view);

        var earthMaterial:BitmapFileMaterial =
new BitmapFileMaterial(
    "http://www.flashmagazine.com/articlefiles/away3d/resources/earthmap1k.jpg");
        var sphere:Sphere = new Sphere({material:earthMaterial});
        view.scene.addChild(sphere);
        this.addEventListener(Event.ENTER_FRAME,update);
    }

    private function update(e:Event):void {
        view.render();
    }
}
}
```

言うまでもなくこの方法の欠点は、マテリアルがロードされるまで何を表示されないことです。これは、LoadSuccess イベントを使って、ビットマップがロードされるタイミングを監視することで回避できます。まず明るい青のカラーマテリアルで球を作成します。

```
sphere = new Sphere({material:"lightblue"});
view.scene.addChild(sphere);
view.render();
```

次いで BitmapFileMaterial を作成し、LOAD\_SUCCESS イベントを監視します。

```
earthMaterial = new BitmapFileMaterial(
    "http://www.flashmagazine.com/articlefiles/away3d/resources/earthmap1k.jpg");
```

```
earthMaterial.addEventListener(LoaderEvent.LOAD_SUCCESS,setMaterial);
```

ビットマップがロードされると、Flash Player が setMaterial() を呼び出します。したがって setMaterial() を次のように定義します。

```
private function setMaterial(e:Event):void {  
    sphere.material = earthMaterial;  
    view.render();  
}
```

この完成ファイルはここ ([Basic09\\_BitmapFileMaterial2.as](#)) からダウンロードできます。この方法の利点は、毎フレーム view.render() メソッドを呼び出す必要がないことです。



Basic09\_BitmapFileMaterial2.as

```
package {  
  
    import away3d.containers.View3D;  
    import away3d.materials.BitmapFileMaterial;  
    import away3d.primitives.Sphere;  
  
    import flash.display.Sprite;  
    import flash.events.Event;  
  
    [SWF(width="500", height="200",  
        frameRate="60", backgroundColor="#FFFFFF")]  
    public class Basic09_BitmapFileMaterial2 extends Sprite {  
  
        private var view:View3D;
```

```

public function Basic09_BitmapFileMaterial2() {

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

    var earthMaterial:BitmapFileMaterial =
new BitmapFileMaterial(
    "http://www.flashmagazine.com/articlefiles/away3d/resources/earthmap1k.jpg",
    {checkPolicyFile:true});

    var sphere:Sphere = new Sphere({material:earthMaterial});
    view.scene.addChild(sphere);
    this.addEventListener(Event.ENTER_FRAME,update);
}

private function update(e:Event):void {
    view.render();
}
}
}

```

**訳者注:**

Flash のオーサリング環境では問題になりませんが、Flash Player 単独で SWF を動作させるとセキュリティエラーが発生するので、上記コードでは、BitmapFileMaterial コンストラクタの2つめの引数に {checkPolicyFile:true}を指定しています。また Flash では[パブリッシュ設定]の[ローカルでの再生に関するセキュリティ]で[ネットワークにのみアクセスする]を選択しておく必要があります。

マテリアルを構成するビットマップには、マテリアルの bitmap プロパティを使ってアクセスできます。

```
var myBitmap:BitmapData = earthMaterial.bitmap
```

これは、テクスチャを適用した後、ビットマップを操作したい場合に便利です。

またビットマップのマテリアルを埋め込みテクスチャから使用方法もあります。そのためには、お使いのエディタでビットマップを埋め込む方法を知っておく必要があります。提供しているコードサンプルは Flex ActionScript プロジェクトとして記述しているので、Flash CS 3 の場合には、Flash CS3 で Flex 特有

のコードをコメント(//)し、ビットマップファイルを埋め込む方法を述べている[このチュートリアル](#)に目を通しておいてください。Flex Builder か Flash CS3 を使っている場合には、コードサンプルに関するビットマップをダウンロードし、それを resources という名前のサブフォルダに置くだけです。

Flex Builder では、クラス変数を定義する場所にビットマップを埋め込みます。通常はクラスの冒頭です。

```
[Embed(source="resources/earthmap1k.jpg")]  
private var earthBitmap:Class;
```

Flash CS4 もこの Embed タグを読み取り使用できるので、処理は同じです。Flash CS3 では“[earthmap1k.jpg](#)”をライブラリに読み込み、ライブラリで読み込んだファイルを右クリックして、[ActionScript に書き出し]を選び、[クラス]に earthBitmap と入力します。すると、球を作成するとき初期化オブジェクトに埋め込みビットマップが渡せるようになり、Away3D が新しい BitmapMaterial を設定します。

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

Flex Builder と Flash CS4 用のサンプルファイルはこちらです (Basic09\_BitmapMaterial.as:リンクなし)。また CS3 用のファイルはこちらです ([Basic09 BitmapMaterial fla](#))。この FLA ファイルまた、タイムラインコードだけ (外部クラスファイルを使用せず) で基本的な 3D シーンを作成する方法の例にもなります。Away3D ソースを FLA と同じフォルダ内にコピーしておきます。

Basic09\_BitmapMaterial.fla のタイムラインのコード

```
import away3d.containers.View3D;  
import away3d.primitives.Sphere;  
  
import flash.display.Sprite;  
  
var view:View3D = new View3D({x:250,y:100});  
addChild(view);  
  
var sphere:Sphere = new Sphere({material:earthBitmap});  
view.scene.addChild(sphere);  
  
view.render();
```

初期化オブジェクトを使用しない場合には、BitmapMaterial 用の変数を宣言しますが、その前に Away3D のユーティリティクラス(Cast)をインポートしておく必要があります(型変換を行うため)。

```
import away3d.core.utils.Cast;
```

その後、インポートした Cast クラスを使って、次のようにビットマップをインスタンス化します。

```
var earthMaterial:BitmapMaterial =  
    new BitmapMaterial( Cast.bitmap(earthBitmap) );  
var sphere:Sphere = new Sphere();  
sphere.material = earthMaterial;  
view.scene.addChild(sphere);  
view.render();
```

このインポートの追加は少し面倒ですが、FlexとCS3で同じActionScriptコードが使えるようになります。ActionScript クラスはこちら( [Basic09 BitmapMaterial2.as](#) )で、FLA ファイルはこちら( [Basic09 BitmapMaterial2 fla](#) )です。

NOTE: Flex Builder をお使いの方は、次のようなもっと標準的な方法でこれを行えます。

```
var earthBitmapData:BitmapData = Bitmap( new earthBitmap() ).bitmapData;  
var earthMaterial:BitmapMaterial = new BitmapMaterial( earthBitmapData );
```

Away3D は Flash で処理できるすべてのイメージ形式をサポートし、特に PNG の透明な領域に留意しています。これは地球の上の空のアニメーションなど面白い効果の実現に使用できます(われわれの [Earth and Heaven チュートリアル](#) をチェックしてみてください)。

MovieClip をテクスチャとして使用する

MovieClip はテクスチャとして使用できます。次のサンプルでは“caustics fla”という名前の Away3D デモファイルを使用します。このファイルには“caustics”という名前のムービークリップシンボルが含まれています。これを球のテクスチャとして使用します。このサンプルでもオーサリングツールによって使用方法が異なります。Flex Builder と Flash CS4 で SWF からムービークリップを読み込むには次のようにします。

```
[Embed(source="resources/caustics.swf", symbol="caustics")]  
private var causticsMovie:Class;
```

Flash CS3 ではライブラリで caustics シンボルの[シンボルプロパティ]ダイアログボックスで[ActionScript に書き出し]を選択し[クラス]に causticsMovie と入力します。これでこのムービークリップシンボルを ActionScript から causticsMovie という名前アクセスできるようになります。この後は Flex Builder/CS4と同じ方法で、テクスチャを作成し球に適用できます。

```
var materialMovie:MovieClip = new causticsMovie() as MovieClip;
var causticsMaterial:MovieMaterial = new MovieMaterial( materialMovie );
var sphere:Sphere = new Sphere({material:causticsMaterial});
view.scene.addChild(sphere);
```

ActionScript クラスはこちら ( [Basic09\\_MovieMaterial.as](#) ) で、FLA ファイルはこちらです ( [Basic09\\_MovieMaterial fla](#) )。

Basic09\_MovieMaterial.as

```
package {

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

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

    [SWF(width="500", height="200",
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic09_MovieMaterial extends Sprite {

        // SWF を埋め込み、クラスとして使用する
        [Embed(source="resources/caustics.swf"), symbol="caustics")]
        private var causticsMovie:Class;
        private var view:View3D;

        public function Basic09_MovieMaterial() {
            view = new View3D({x:250,y:100});
```

```

        addChild(view);

        // causticsMovie インスタンスの作成
        var materialMovie:MovieClip =
            new causticsMovie() as MovieClip;
        // そのインスタンスを使って MovieMaterial を作成
        var causticsMaterial:MovieMaterial =
            new MovieMaterial( materialMovie );
        // MovieMaterial を使って球を作成
        var sphere:Sphere = new Sphere(
                                {material:causticsMaterial});
        view.scene.addChild(sphere);

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

    private function update(e:Event):void {
        view.render();
    }
}
}
}

```

#### Basic09\_MovieMaterial fla

```

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

var view:View3D;

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

    // 1行分短くできる
    var earthMaterial:MovieMaterial = new MovieMaterial(
                                                new causticsMovie() as MovieClip );
}

```

```
var sphere:Sphere=new Sphere({material:earthMaterial});
view.scene.addChild(sphere);

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

function update(e:Event):void {
    view.render();
}

init();
```

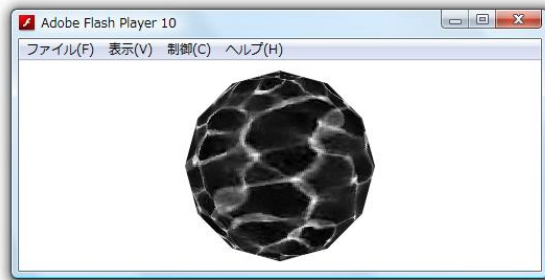
ムービークリップがインタラクティブ性を持っている場合には、3D オブジェクト上でもそれを利用できます。必要なことはそのマテリアルをインタラクティブに設定するだけです。

```
earthMaterial.interactive = true;
```

たとえば、ユーザーが球をクリックするとそのムービークリップの再生と停止を行いたい場合には、sphereClick()メソッドは次のようになります。

```
private function sphereClick(e:MouseEvent3D):void {
    if(isMaterialPlaying){
        materialMovie.stop();
        isMaterialPlaying = false;
    } else {
        materialMovie.play();
        isMaterialPlaying = true;
    }
}
```

ここまでをまとめた ActionScript クラスはこちら([Basic09 MovieMaterial2.as](#))です。下はその実行画面です。



Basic09\_MovieMaterial2.as

```
package {

    import away3d.containers.View3D;
    import away3d.events.MouseEvent3D;
    import away3d.materials.MovieMaterial;
    import away3d.primitives.Sphere;

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

    [SWF(width="500", height="200",
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic09_MovieMaterial2 extends Sprite {

        [Embed(source="resources/caustics.swf", symbol="caustics")]
        private var causticsMovie:Class;
        private var view:View3D;
        private var materialMovie:MovieClip;
        private var isMaterialPlaying:Boolean = true;

        public function Basic09_MovieMaterial2() {
            view = new View3D({x:250,y:100});
            addChild(view);

            materialMovie = new causticsMovie() as MovieClip;
            var causticsMaterial:MovieMaterial =
                new MovieMaterial( materialMovie );
```



```
view.scene.addChild(sphere);
```

ColorMaterial と同様、PhongBitmapMaterial でも、シーンに光源を追加する必要があります。追加しないと、マテリアルは通常の BitmapMaterial としてレンダリングされます。

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

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

ライトの ambient、diffuse、specular の各プロパティがテクスチャに与える影響は、[前の PhongColorMaterial のチュートリアル](#)を参照してください。またムービークリップをテクスチャとして使用できる Phong マテリアルもあります。この PhongMovieMaterial は前のサンプルと動作はまったく同じで、ただ光源を追加する必要があります。

```
var causticsMaterial:PhongMovieMaterial = new PhongMovieMaterial(
    new causticsMovie() as MovieClip );
var sphere:Sphere = new Sphere({material:causticsMaterial});
```

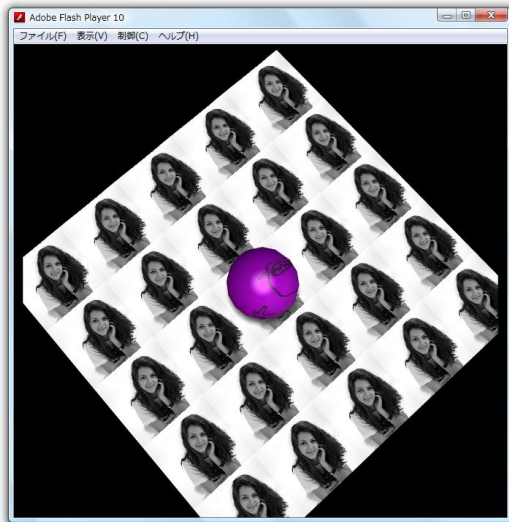
### ビットマップのタイル表示

ビットマップタイルは、切れ目なしに繰り返し表示できるイメージのことです。たとえば繰り返し可能な小さな草のイメージを使用すると、アプリケーションに巨大なイメージを追加しなくても、芝全体を表すことができます。イメージをタイル表示するには、TransformBitmapMaterial という特殊なマテリアルを使用します。次のコードでは、TransformBitmapMaterial を作成し、オブジェクトのサーフェース全体で5回繰り返し設定を行っています。

```
var transformedMaterial:TransformBitmapMaterial =
    new TransformBitmapMaterial( image.bitmapData );
transformedMaterial.repeat = true;
transformedMaterial.scaleX = 0.2;
transformedMaterial.scaleY = 0.2;
```

またマテリアルをオフセットしたり回転させることもできます。

```
transformedMaterial.offsetX = 25;  
transformedMaterial.offsetY = 25;  
transformedMaterial.rotation = 5;
```



ムービー: [Basic09\\_BitmapMaterialExplorer.as](http://Basic09_BitmapMaterialExplorer.as)

このマテリアルは Phong シェーディングと直接組み合わせることはできませんが、高度な CompositeMaterial と併用する場合には、似たような効果可以实现できます。CompositeMaterial はこの後簡単に述べています。

Basic09\_BitmapMaterialExplorer2.as(このコードには訳者が意図的に変えた部分があります)

```
package {  
  
    import away3d.cameras HoverCamera3D;  
    import away3d.containers.View3D;  
    import away3d.core.render.Renderer;  
    import away3d.core.utils.Cast;  
    import away3d.lights.DirectionLight3D;  
    import away3d.materials.PhongBitmapMaterial;  
    import away3d.materials.TransformBitmapMaterial;  
    import away3d.materials.utils.SimpleShadow;  
    import away3d.primitives.Plane;  
    import away3d.primitives.Sphere;
```

```
import flash.display.Bitmap;
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;

[SWF(width="800", height="800",
      frameRate="60", backgroundColor="#000000")]
public class Basic09_BitmapMaterialExplorer2 extends Sprite {
    // テクスチャに必要なイメージを埋め込む
    [Embed(source="spiralgraphics/p1.jpg")]
    public var texture1:Class;
    [Embed(source="spiralgraphics/p2.jpg")]
    public var texture2:Class;
    [Embed(source="spiralgraphics/p3.jpg")]
    public var texture3:Class;
    [Embed(source="spiralgraphics/p4.jpg")]
    private var texture4:Class;
    [Embed(source="spiralgraphics/p5.jpg")]
    private var texture5:Class;
    [Embed(source="spiralgraphics/p6.jpg")]
    private var texture6:Class;
    [Embed(source="spiralgraphics/p7.jpg")]
    private var texture7:Class;
    [Embed(source="spiralgraphics/p8.jpg")]
    private var texture8:Class;
    [Embed(source="spiralgraphics/p9.jpg")]
    private var texture9:Class;
    [Embed(source="spiralgraphics/p10.jpg")]
    private var texture10:Class;
    [Embed(source="spiralgraphics/p11.jpg")]
    private var texture11:Class;
    [Embed(source="spiralgraphics/p12.jpg")]
    private var texture12:Class;
    [Embed(source="spiralgraphics/p13.jpg")]
    private var texture13:Class;
```

```

[Embed(source="spiralgraphics/p14.jpg")]
private var texture14:Class;
[Embed(source="spiralgraphics/p15.jpg")]
private var texture15:Class;

private var view:View3D;
private var plane:Plane;
private var sphere:Sphere;
private var material1:PhongBitmapMaterial;
private var material2:TransformBitmapMaterial;
private var cam:HoverCamera3D;

public function Basic09_BitmapMaterialExplorer2() {
    cam = new HoverCamera3D();

    view = new View3D({x:400,y:400,camera:cam});
    view.renderer = Renderer.CORRECT_Z_ORDER;
    addChild(view);

    plane = new Plane(
    {segmentsW:6, segmentsH:6, y:-120, width:1100, height:1100});
    view.scene.addChild(plane);

    sphere = new Sphere(
        {radius:100,segmentsW:14,segmentsH:8});
    sphere.rotationX = 85;
    view.scene.addChild(sphere);

    // 俯瞰気味のカメラ
    cam.targetpanangle = cam.panangle = 0;
    cam.targettiltangle = cam.tiltangle = 70;
    cam.zoom = 10;
    cam.lookAt( sphere.position );
    cam.target.z += 10;

    this.addEventListener(MouseEvent.CLICK,changeMaterial);

```

```

        var light:DirectionalLight3D = new DirectionalLight3D();
        light.y = 500;
        light.x = -50;
        light.z = -50;
        light.specular = 0.6;
        view.scene.addChild(light);

        var simpleshadow:SimpleShadow =
            new SimpleShadow(sphere,0x99333333,20,-95);
        simpleshadow.apply(view.scene);

        changeMaterial();
        this.addEventListener(Event.ENTER_FRAME,update);
    }

    private function update(e:Event):void {
        cam.targetpanangle += .3;
        cam.hover();
        view.render();
    }

    private function changeMaterial(e:MouseEvent = null):void {
        var className1:String =
            "texture"+Math.floor( (Math.random()*15)+1).toString();
        var className2:String =
            "texture"+Math.floor( (Math.random()*15)+1).toString();

        material1 = new PhongBitmapMaterial(
            Cast.bitmap( new this[className1]() as Bitmap ) );
        // 繰り返しを可能にし、1/5 に縮小
        material2 = new TransformBitmapMaterial(
            Cast.bitmap( new this[className2]() as Bitmap ),
            {repeat:true, scaleX:.2, scaleY:.2}
        );
        material2.smooth = true;
    }

```

```
        sphere.material = material1;
        plane.material = material2;
        view.render();
    }
}
```

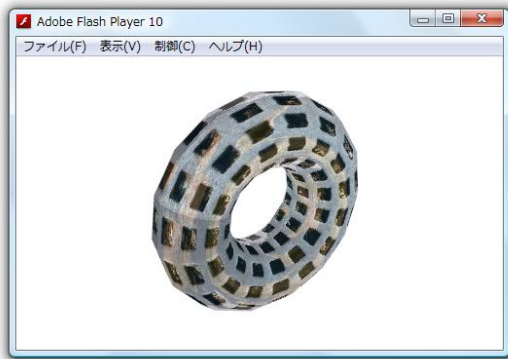
### 反射の追加—環境マテリアル

本チュートリアル最後のマテリアルのタイプは環境マテリアルです。環境マテリアルは、サーフェスがライトやオブジェクトの周囲を反映する錯覚を生み出します。3D オブジェクトの周りに実際にあるものを映すことは多量の計算が必要になりますが、これは少し“だまし”を加えることで解決できます。反射に見せかけるイメージを使用することで、適度なスピードの確保が実現できます。反射マテリアルは、ビットマップマテリアルに比べ、相応の代償がともなうことを知っておく必要があります。反射の追加はわずかの CPU 負荷で済みますが、反射平面を使った屈折には多大な負荷がともないます。これについてはこの後取り上げています。

Away3D では、環境反射を提供するマテリアルは `EnviroColorMaterial` と `EnviroBitmapMaterial` の 2 つが提供されています。想像されているように `EnviroColorMaterial` はカラーを使用し、`EnviroBitmapMaterial` はテクスチャにビットマップを使用します。これらは 2 つめのパラメータとして、反射に使用するビットマップを取ります。

```
var reflectiveMaterial:EnviroBitmapMaterial =
    new EnviroBitmapMaterial(
        Cast.bitmap( textureBitmap ),
        Cast.bitmap( reflectionBitmap )
    );
reflectiveMaterial.reflectiveness = 0.3;
```

`reflectiveness` プロパティは、メインのマテリアルを通して、反射がどれだけ輝くかという量を調整します。これは 0(反射なし)から 1(反射のみ)の間で調整できます。反射について知っておくべきことの 1 つは、実際に反射を目にするには、オブジェクトかカメラが移動するときだけということです。さらに反射は平面でなく、局面のサーフェスで最良に機能するということも知っておく必要があります。次のサンプルをクリック&ドラッグで試してみてください。



ムービー: [Basic09\\_EnviroBitmapMaterial.as](#)

ムービーをスタートさせると、円環が回転を始めます。円環をクリックすると、その回転が止まりドラッグできません。注意してみると、実際の反射が作用しているわけではなく、地球のビットマップが映し出されていることが分かります。しかしリアルタイム 3D ではこれで十分です。

Basic09\_EnviroBitmapMaterial.as

```
package {

    import away3d.cameras HoverCamera3D;
    import away3d.containers View3D;
    import away3d.core.utils Cast;
    import away3d.materials EnviroBitmapMaterial;
    import away3d.primitives Sphere;
    import away3d.primitives Torus;

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

    [SWF(width="500", height="300", )
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic09_EnviroBitmapMaterial extends Sprite {
        // 反射用ビットマップ
        [Embed(source="resources/earthmap1k.jpg")]
        private var reflectionBitmap:Class;
        // テクスチャ用ビットマップ
```

```

[Embed(source="resources/metal.jpg")]
private var textureBitmap:Class;

private var view:View3D;
private var torus:Torus;
private var sphere:Sphere;

private var cam:HoverCamera3D;
private var move:Boolean = false;
private var lastPanAngle:Number;
private var lastTiltAngle:Number;
private var lastMouseX:Number;
private var lastMouseY:Number;

public function Basic09_EnviroBitmapMaterial() {

    cam = new HoverCamera3D();
    cam.targetpanangle = cam.targettiltangle =
        cam.tiltangle = cam.panangle = 0;
    cam.mintiltangle = -90;
    view = new View3D({x:250,y:150,camera:cam});
    addChild(view);

    // 環境ビットマップマテリアルの作成
    var reflectiveMaterial:EnviroBitmapMaterial =
        new EnviroBitmapMaterial(
            Cast.bitmap( textureBitmap ),
            Cast.bitmap( reflectionBitmap )
        );
    reflectiveMaterial.reflectiveness = 0.3;

    // 円環を作成し、マテリアルを適用
    torus = new Torus(
        {segmentsR:15,segmentsT:12,material:reflectiveMaterial});
    torus.tube = 35;
    torus.radius = 75;

```

```

        view.scene.addChild(torus);

        this.addEventListener(Event.ENTER_FRAME,update);
        stage.addEventListener(
            MouseEvent.MOUSE_DOWN, MouseDown);
        stage.addEventListener(
            MouseEvent.MOUSE_UP, MouseUp);
    }

    private function update(e:Event):void    {
        // ドラッグ中でない場合には、円環自体を回転
        if(!move){
            torus.rotationX += .5;
            torus.rotationY += .8;
        }

        var cameraSpeed:Number = 0.3;
        if (move) {
            cam.targetpanangle =
                cameraSpeed*(stage.mouseX - lastMouseX) + lastPanAngle;
            cam.targettiltangle =
                cameraSpeed*(stage.mouseY - lastMouseY) + lastTiltAngle;
        }
        cam.hover();
        view.render();
    }

    private function MouseDown(event:MouseEvent):void {
        lastPanAngle = cam.targetpanangle;
        lastTiltAngle = cam.targettiltangle;
        lastMouseX = stage.mouseX;
        lastMouseY = stage.mouseY;
        move = true;
    }

    private function MouseUp(event:MouseEvent):void {

```

```
        move = false;
    }
}
}
```

最良の結果を得るには

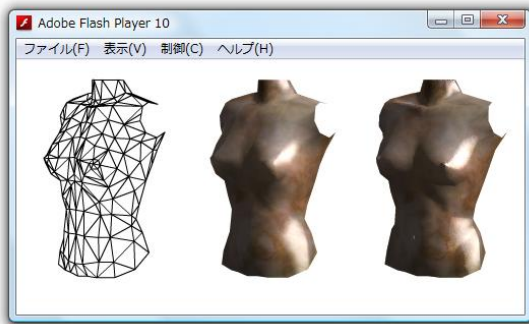
Flash 3D はまだ初期段階にあります。パフォーマンスと品質の点でいうと、8年前のコンピュータゲームくらいの段階です。この事実にもとづくと、Flash 3D を Xbox360 や PS3、Wii などのゲームと比べるとはかさか公正さに欠けます。しかし使用するのはエンドユーザーであるので、われわれにできるのはここまで述べてきたマテリアルの限界を克服して、Flash 3D の外観をよいものにするということです。そのためのトリックがいくつかあり、そのポイントは多くの場合マテリアルにあります。

まず必要なことは、マテリアルに使用する適切なテクスチャを見つけることです。グーグルで“3d texture maps”を検索すると、ビットマップテクスチャを提供する無数のサイトが見つかります。通常はそこで気に入ったものを購入することになります。高品質でしかもフリーのテクスチャを探し当てるには時間がかかるので、多くの場合は買った方が安く上がります。多くのサイトではさまざまな目的で作成済みのテクスチャを提供していて、ブロックや木、水、金属といった範疇に分けられています。

またモデル用のテクスチャを生成するソフトウェアもあります。マウスのクリックでマテリアルが変わった最初のサンプルでは、[Genetica](#)という PC ソフトウェアで作成した高品質なテクスチャを使用しています。これを使用すると“タイル化”が可能な高品質のテクスチャが無数に作成できます。

適切なテクスチャが入手できたら、次はシーンに奥行きを加えます。これは次のように考えてください。たとえば外のシーンに出ると(FPS ゲーム(“一人称視点シューティングゲーム”)のように)、太陽の移動は通常認識されないので、影はつねに同じ場所にできます。これによってテクスチャに前もって影を“焼きこんでおく”ことが可能になります。これでうまくいく場合には、3D 環境の見栄えは非常によくなります。テクスチャを焼きこむ処理は 3D エンジン自体で行われないので、“焼きこみ”をサポートするモデリングソフトウェア(3ds max、Maya、Blender やそのほか)でこれを行う必要があります。

Away3D にはまた、複数のマテリアルと組み合わせることのできるパワフルな Dot3 と CompositeMaterial という2つが提供されています。これらには多くの可能性があるので、また別のチュートリアルで触れたいと思っています。その簡単な予習として、単純な胴体モデルに Dot3 マテリアルでできることを下のサンプルで見てください。



ムービー: [Pedestal.as](#)

法線マップを持つ Dot3 ビットマップマテリアルを追加することで、この単純なモデルが何千ものポリゴンで作られているように見えます。Away3D ではまた、こういった法線マップを生成するクラスがあり、サーフェースのバンプマッピングもサポートされています。

Pedestal.as

```
package {

    import away3d.containers.View3D;
    import away3d.core.base.Mesh;
    import away3d.core.utils.Cast;
    import away3d.lights.DirectionLight3D;
    // Md2 ファイル形式用のファイルローダ(非アニメーション版).
    import away3d.loaders.Md2still;
    import away3d.materials.Dot3BitmapMaterial;
    import away3d.materials.EnviroBitmapMaterial;
    import away3d.materials.IMaterial;
    import away3d.materials.PhongBitmapMaterial;
    import away3d.materials.WhiteShadingBitmapMaterial;

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

    [SWF(width="500", height="250",
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Pedestal extends Sprite {
```

```

private var view:View3D;
private var torso1:Mesh;
private var torso2:Mesh;
private var torso3:Mesh;

// 胴体用の大理石テクスチャ
[Embed(source="resources/torso_marble2.jpg")]
public static var TorsoImage:Class;

// 胴体用の法線マップ用テクスチャ
[Embed(source="resources/torso_normal_400.jpg")]
public static var TorsoNormal:Class;

// 胴体のメッシュ
[Embed(source="resources/torsov2.MD2",
        mimeType="application/octet-stream")]
public static var TorsoMD2:Class;

public function Pedestal()    {
    // シーンの設定
    view = new View3D();
    this.addChild(view);
    var light:DirectionalLight3D =
        new DirectionalLight3D({color:0xFFFFFFFF,
                                ambient:0.25, diffuse:0.75, specular:0.9});
    light.x = 40000;
    light.z = -40000;
    light.y = 40000;
    view.scene.addChild( light );

    // マテリアルの設定
    // マテリアルのテクスチャに大理石テクスチャを、
    // DOT3 マップに法線マップ用テクスチャを使用
    var torsoNormalMaterial:Dot3BitmapMaterial =
        new Dot3BitmapMaterial(
            Cast.bitmap(TorsoImage),

```

```

        Cast.bitmap(TorsoNormal)
    );
    // マテリアルのテクスチャに大理石テクスチャを使用
    var torsoPhongMaterial:PhongBitmapMaterial =
        new PhongBitmapMaterial(
            Cast.bitmap(TorsoImage), {specular:0.5}
        );

    // 胴体の設定
    // 右の胴体には DOT3 マップ
    torso1 = addTorso(450,-125, torsoNormalMaterial);
    // 真ん中の胴体には Phong マテリアル
    torso2 = addTorso(270,-125, torsoPhongMaterial);
    // 左の胴体にはマテリアルなし
    torso3 = addTorso(90,-125);

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

private function update(e:Event):void    {
    // 3つの胴体を回転
    torso1.rotationY += 3;
    //torso2.rotationY += 3;
    //torso3.rotationY += 3;
    view.render();
}

// 胴体を設定して指定された位置に追加
private function addTorso(
    x:Number,y:Number,m:IMaterial = null
):Mesh {

    // md2 ファイルの生の xml データから 3D メッシュオブジェクトを作成
    var torso:Mesh = Md2still.parse(
TorsoMD2,
{ownCanvas:true, material:"white#black", name:"torso", back:"red#blue"}

```

```
);
torso.movePivot(
    (torso.minX+torso.maxX)/2,
    (torso.minY+torso.maxY)/2,
    (torso.minZ+torso.maxZ)/2
);
torso.x = x;
torso.y = y;
if(m!=null){ torso.material = m; }
torso.scale(.04);
view.scene.addChild( torso );
return torso;
}
}
}
```

訳者注:  
このクラスに必要な JPG や MD2 ファイルは、Away3D のクラスファイル内の branches¥clipping¥demos¥src¥assets に含まれています。また <http://away3d.googlecode.com/svn/branches/clipping/demos/src/assets/> からダウンロードすることもできます。