

## Blender to Flash workflow

本ドキュメントは、<http://ffdmag.com/magazine/848-how-to-become-a-flash-flex-developer> で公開されている「Blender to Flash workflow」の記事を、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://ffdmag.com/magazine/848-how-to-become-a-flash-flex-developer>

からのリンクで読むことができます。

Blender から Flash へのワークフロー

by Denis Ippel

Flash の 3D は、Flash Player 9 のリリース以来、すべてが新しい機能であり、その実行には相当な作業がともなうにも関わらず、大きな人気を博しています。3D Flash 体験を最適化する1つの方法に、3D モデルを直接 ActionScript 3.0 に書き出す方法があります。

●ここで学べる事柄

- ・Blender による簡単な 3D オブジェクトの作成方法
- ・Blender でのマテリアルとテクスチャの使用方法
- ・Flash 用 Sandy 3D エンジンの使用方法

●身につけていると役立つ事柄

- ・3D の基本的な概念
- ・Flash インターフェイスに慣れていること
- ・ActionScript 3.0 に慣れていること

難易度レベル: 中

XML ファイルのロードと解析は、Sandy や Away3D、Papervision3D などの 3D Flash エンジンに 3D オブジェクトをロードするとき、共通して取られる方法です。その XML ファイルの構造は [COLLADA](#) (COLLAborative Design Activity) 仕様で記述されます。多くの情報がこの XML ドキュメントから取得されるので、ファイルは肥大化する傾向にあります。この肥大化には、3D アプリケーションのパフォーマンスに影響を及ぼすいくつかのデメリットがあります。

- ・アプリケーションには巨大な XML ファイルをロードしなくてはならない
- ・XML DOM のすべてのノードをトラバース(移動)する必要がある
- ・3D オブジェクトの情報は圧縮を解除する必要がある
- ・頂点データは、3D エンジンが使用する形式に直す必要がある

3D モデルが多くの頂点と、テクスチャ座標に関する情報を持っている場合、スクリーンにオブジェクトを表示するには相応の時間がかかります。もしこれらすべての手順を飛ばし、3D モデルを ActionScript 3.0 クラスに書き出すことができたらどうでしょう？ 以下では、これが可能になる 3D コンテンツの作成について見ていきます。

Blender

Blender は、すべてのオペレーティングシステムで使用できる、フリーのオープンソースプログラムです。Blender は少量のインストールサイズしか必要としないにもかかわらず、驚くほど見事な 3D コンテンツの作成に役立つ広範囲のツールを提供します。Python プログラミング言語を使用すると、Blender の API とやりとりし、プラグインを作成することもできます。

Blender のユーザーインターフェイスは Flash や Photoshop のものと少し異なるので、最初は困惑するかもしれません。しかしその動作に慣れてしまえば、インターフェイスのよさが分かるようになります。以下はインターフェイスの使い方に関するごく簡単な解説です。

## Blender のユーザーインターフェイス

Blender を <http://www.blender.org/> からダウンロードしインストールして初めて起動すると、デフォルト画面が表示されます(図1 参照)。この画面は次の要素で構成されています。

### 訳者注:

本翻訳文で使用した Blender はバージョン 2.49b です。Windows Vista にインストールする場合には <http://blender.jp/modules/xfsection/article.php?articleid=277> が参考になります。なお本記事では Python コードを使用するので、Python がインストールされていない場合には、上記ページに書かれているようにシステムにインストールする必要があります。

- ・メインアプリケーションメニュー
- ・デフォルトのキューブを置いた 3D ビュー
- ・3D ビューメニュー
- ・ボタンウィンドウ

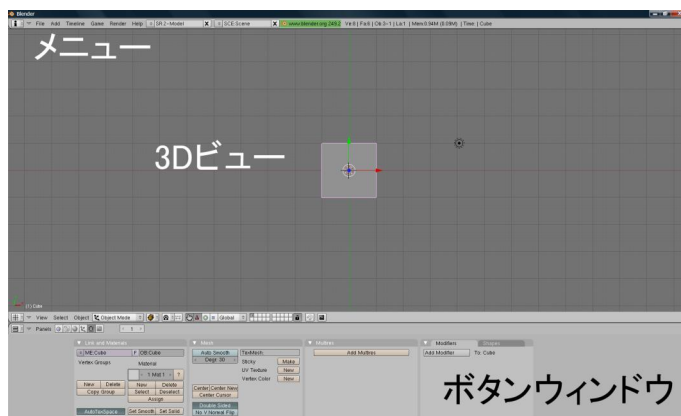


図1: Blender のユーザーインターフェイス

メインアプリケーションメニューはファイルを開き保存したり、シーンをレンダリングしたり、プログラムの設定を変更する場所です。3D ビューは 3D シーンを編集する場所で、いくつかの編集モードがあり、[Object]メニューの右にあるドロップダウンメニューをクリックすることでモードにアクセスできます(図2参照)。

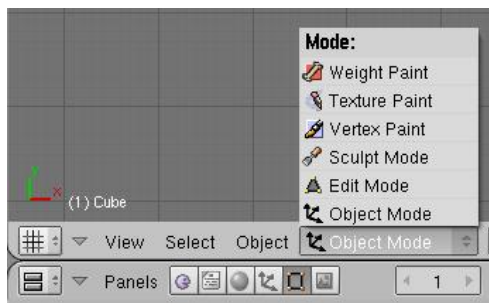


図2: [Mode]メニュー

本記事ではこれらのメニューの中で Edit Modeと Object Mode だけを使用します。Edit Mode はオブジェクトのジオメトリ(形状)に影響し、Object Mode はオブジェクト全体に影響するモードです。

[Mode]メニューの右にあるボタンからは、[Viewport Shading]メニューにアクセスできます(図3参照)。

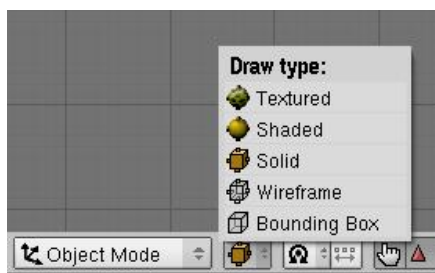


図3: [Viewport Shading]メニュー

ここでは、ビューポートに適用されるシェードタイプが選択できます。この後オブジェクトのマテリアルを作成するときには、Shadedと Textured を使用します。

一番左のボタンではウィンドウタイプが変更できます。このメニューはすべてのウィンドウで表示されます。これはつまり、希望するどの場所でも、すべてのウィンドウタイプが使用できるということです。たとえば 3D ビューは希望する数だけいくつでも作成できます。これを確認するため2つめの 3D ビューを作ってみましょう。現行の 3D ビューウィンドウは 3D ウィンドウとその下のウィンドウとの境界上で右クリックすることで、2つに分けることができます。

この境界上をマウスでなぞると、マウスカーソルがダブルアロー（2つの矢印が逆方向を向いている）に変わります。

その状態で右クリックするとコンテキストメニューが現れます（図4参照）。このメニューから[Split Area]を選択するとグレーの線が表示されます。



図4:ビューを分割する

その状態でマウスを動かし、3Dビューのどこかで左クリックします。この操作によって2つめの3Dウィンドウが作成できます。

2つめの3Dウィンドウは実際には必要ありませんが、後でImage Editorが必要になります。ビューはウィンドウタイプを選択するメニューで変更できます。これは一番左のドロップダウンメニューです（図5参照）。このメニューをクリックすると[UV/Image Editor]が選択できます。

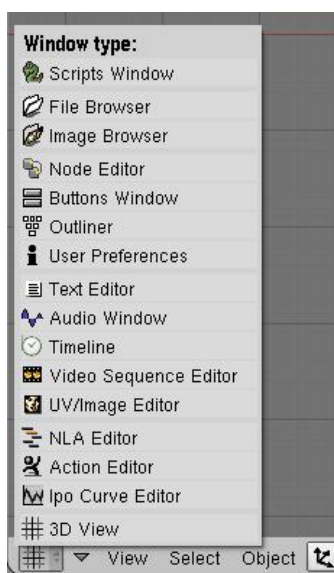


図5:ウィンドウタイプを選択するメニュー

3Dウィンドウの下にあるウィンドウはボタンウィンドウと呼ばれます。このウィンドウには、オブジェクトの構成（コンフィグレーション）やマテリアルのオプションやプロパティの設定、レンダリングオプションの設定を行うパネルが含まれています。この後われわれは Shading パネルと Scene パネルを使用します。

## テクスチャの作成

本記事では Blender によるモデリングの詳細は解説しません。これに関してはすぐれた書籍が発行されています(最後の“ここからどこへ行く?”節参照)。われわれはデフォルトの3Dシーンにすでに存在しているキューブ(立方体)のテクスチャを作成します。作成を開始するにはまず、[Shading]コンテキストをアクティブにし、そのサブコンテキストの[Texture buttons]をアクティブにします(図6参照)。

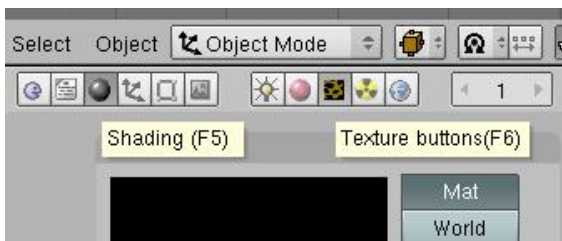


図6: [Shading]をクリックしてから[Texture buttons]をクリックする

モデルに適用するテクスチャを確認するには、Viewport Shading タイプを[Shaded]に設定します(図3参照)。

これでテクスチャを指定しその結果を確認する設定ができました。[Texture buttons]パネルの[Texture Type]ドロップダウンメニューから[Marble]テクスチャを選択し(図7参照)、[Preview]の[Mat]ボタンをクリックすると、3D ウィンドウ内のオブジェクトが変化します。

しかしテクスチャは、[Texture buttons]パネルのプレビュー表示ほど詳細に表示されません。テクスチャのレンダリングにはパワーが必要なので、テクスチャを変更するたびにレンダリングするのはよい考えではありません。設定したテクスチャがキューブ上でどのように見えるかを確認するには、F12 キーを押します。これによって現在のフレームのレンダリングが引き起こされます。するとカラーがかなり奇妙でテクスチャのプレビューと一致していないことに気づかれるでしょう。レンダリングでは白黒ではなくマゼンタと白で表示されています。このカラーは[Material buttons]パネル([Texture buttons]コンテキストの左)で変更できます(図8参照)。  
[Material buttons]パネルの一番右にある[Map To]タブをクリックします(図9参照)。マゼンタはここで好きなカラーに変更できます。

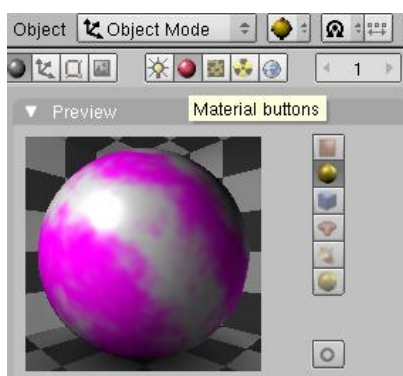


図8: [Material buttons]パネル



図9: [Map To]タブ

マゼンタ色の矩形をクリックし、新しいカラー値(たとえば#990000)を入力し、F12 キーを押してレンダリングします。今度は赤い大理石模様のキューブが表示されます。

以上でオブジェクトにテクスチャが設定できたので、次はファイルに書き出します。

#### テクスチャの書き出し

このテクスチャを 3D エンジンで使用するにはその前に、イメージを新たに作成する必要があります。それが終わったら UV 座標をアンラップ(展開)します。UV 座標はエンジンに対し、イメージの各部をどこに配置するかを伝える数値です。では前のテクスチャの新しいイメージを作成しましょう。ウィンドウを[UV/Image Editor]ウィンドウに変え(図5参照)、[Image]メニューから[New...]を選択します(図 10 参照)。



図 10: 新しいイメージの作成

表示されるダイアログボックスで、[Width]に 1024(デフォルト)、[Height]に 768 を入力して[OK]をクリックします(数値はテキストフィールドをダブルクリックして入力します)。

つづいて UV 座標をアンラップします。ウィンドウを[3D View]に変更し、[Edit Mode]に変えます(図2参照)。[Mode]メニューの左に[Mesh]メニューがあるので、これをクリックし[UV Unwrap]を選択します(図 11 参照)。すると[UV Calculation]メニューが開きます。

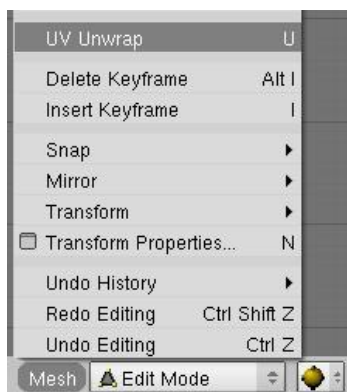


図 11: [UV Unwrap]を選択

このメニューからは一番下の[Unwrap (smart projections)]を選択します。すると Projection と UV Layout に関する指定を行うダイアログボックスが開きます。われわれは単純なプリミティブをアンラップするので、デフォルト値のままでも問題ありません。そのまま変更せずに[OK]をクリックします。するとその直後 UV/Image Editor ウィンドウが更新されます。UV/Image Editor ウィンドウを見ると、アンラップされた UV 座標が線でつながれ、イメージの上に乗っているのが分かります。

テクスチャをイメージに焼き込むときがやってきました。UV/Image Editor ウィンドウで新たに作成したイメージに戻す必要があります。UV メニューの右にある2つの矢印ボタンをクリックし、Untitled というイメージを選択します(図 12 参照)。次いで Button ウィンドウで[Scene]ボタンをクリックします(図 13 参照)。

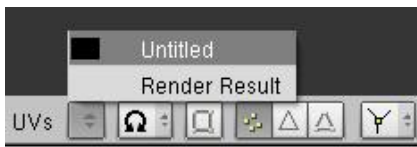


図 12: 新たに作成したイメージに変更する



図 13: [Scene]コンテキストに移動

Scene Button ビューで、左から3つめのパネルにある[Bake]タブをクリックします(図 14 参照)。パネルの右側には縦に並んでいるボタンがあります。その1つめの[Full Render]ボタンがデフォルトでは選ばれています。この選択のまま進めると、シェーダや輝度、反射などがすべてイメージに焼き込まれます。またこのテクスチャのシャドウはかなり暗いので、シャドウも焼き込みたくありません。同じボタングループの中に [Textures]ボタンがあります。これを選択し、パネルに左側にある大きな[BAKE]ボタンをクリックします。

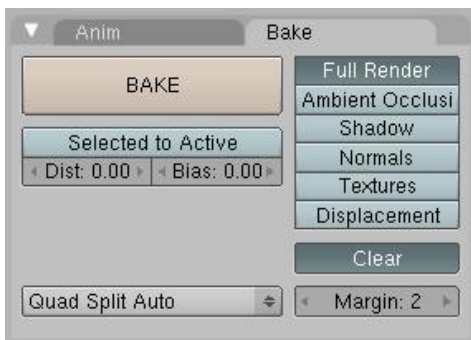


図 14: テクスチャを焼く

結果は UV/Image Editor ウィンドウに表示されます。ファイルは UV/Image Editor ウィンドウの [Image]メニューから[Save as...]を選択します(図 15 参照)。ファイルは cube-texture.png という名前で Flash ファイルを保存するフォルダに保存します。ファイルタイプ(ここでは PNG)はこのダイアログボックス下部で選択する必要があることに注意してください。

### 3D オブジェクトの書き出し

エクスポーターは Python スクリプトで、Blender の scripts フォルダに追加する必要があります。スクリプトはわたしの[ブログ](#)からダウンロードできます(AS3Export.zip)。Blender プロジェクトを保存し(メインアプリ

ケーションメニューの[File]-[Save as...]を選択します)、プログラムを終了させます。

AS3Export.zip を解凍すると AS3Export.py と AS3Export フォルダが現れるので、この2つのそのまま次のフォルダにコピーします。

Windows Vista : C:\Users< ユーザー名 >\AppData\Roaming\Blender Foundation\Blender\blender\scripts

Mac : /Applications/blender/blender.app/Contents/MacOS/.blender/scripts

訳者注:

.blender というフォルダが表示されない場合には、Windows Vista では[フォルダオプション]の[表示]で[すべてのフォルダとファイルを表示する]を有効にします。Mac ではメニューの[移動]-[フォルダへ移動]を選択して、上記パスを入力して移動します。

スクリプトを保存したら再度 Blender を起動し、作業していたファイルを開きます。実はファイルを書き出す前に重要な作業が残されています。ポリゴンはすべて、Flash 3D エンジンが理解できる形式に変換する必要があります。これは四角形を三角形に変換するということです。この操作は Edit Mode (図3参照)での[Ctrl] + [T]で行えます。

これでエクスポートの準備が整いました。キューブを右クリックで選択します。メインアプリケーションメニューの[File]をクリックし[Export]まで下ると、scripts フォルダにあるエクスポートの全スクリプトが表示されます。インストールしたエクスポーターの名前は ActionScript 3.0 Class(.as)です(図 16 参照)。

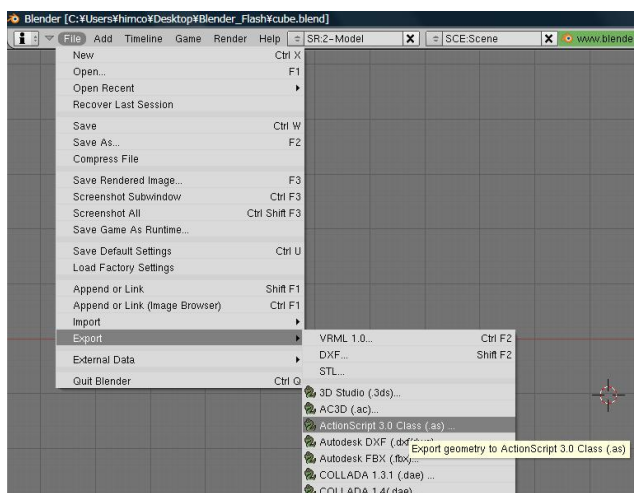


図 16: ActionScript 3.0 Exporter を選択する

ActionScript 3.0 Class(.as)を選択すると[ActionScript 3.0 Exporter]ダイアログウィンドウが開きます。ここではわれわれの 3D オブジェクトクラスの詳細を入力します(図 16-2 参照)。

- Package name: 空欄のまま
- 3D Engine: Sandy
- File location: テクスチャのイメージのある同じ場所

訳者注:

原文には Class name に BlenderCube を指定するとありますが、翻訳時にダウンロードしたエクスポーターでは Class name の項目は存在しないようです。上記の通り指定して書き出すと、Cube.as ファイルが生成されました。



図 16-2: ActionScript 3.0 Exporter

[Export]ボタンをクリックすると、ActionScript クラスが書き出され保存されます。ついに Flash に切り替えるときがやってきました。

### Sabdy 3D エンジンの設定

Sandy 3D エンジンのクラスは ZIP ファイルでまとめられたものが、<http://www.flashsandy.org/download> からダウンロードできます。本記事で使用するバージョンは 3.1 です(本翻訳では Sandy3-1-1\_src\_rev1008.zip をダウンロードしました)。

ZIP ファイルを解凍し、現れた sandy フォルダをテクスチャイメージとエクスポーターが生成した ActionScript クラスのあるフォルダにコピーします。

Flash で新規 FLA ファイルを作成し同じフォルダに保存します。新規 ActionScript クラスを Main.as という名前で作成しこれも同じフォルダに保存します。このファイルがメインアプリケーションクラスになります。

FLA ファイルのプロパティパネルで[クラス]に Main を指定します(図 17 参照)。



図 17: ドキュメントクラスの設定

ではコードを組み合わせていきましょう。最初に必要なのは 3D シーンの設定です。これは数行のコードで済みます。Sandy 3D エンジンがみなさんに代わって多くの事柄を行ってくれるのです。Camera3D クラスのインスタンスの作成から始めます。このクラスのコンストラクタには幅と高さの2つのパラメータが要ります。ここではステージの幅と高さ(550 と 400)を使用することになります。次いで 3D 空間におけるカメラの位置を決める x、y、z 座標を指定する必要があります。カメラを作成しその位置を決めたら、Scene3D オブジェクトが作成できます(リスト1参照)。

#### リスト1: 3D シーンの設定

```
private function setup3DScene():void {  
    var camera:Camera3D=new Camera3D(550,400);  
    camera.x=0;  
    camera.y=0;  
    camera.z=150;  
    camera.lookAt( 0, 0, 0 );  
  
    scene = new Scene3D( "scene", this, camera, new Group( "rootGroup" ) );  
}
```

Cube クラスのインスタンスを作成する前に、そのテクスチャをロードする必要があります。これは通常の Loader と、Event.COMPLETE イベント用と IOErrorEvent.IO\_ERROR イベント用の 2 つのイベントリスナーを使って行います。テクスチャのロードが完了したら、その LoaderInfo オブジェクトからビットマップを取得します(リスト2参照)。

#### リスト2: テクスチャイメージのロード

```

private function loadTexture():void {
    var loader : Loader = new Loader();
    loader.contentLoaderInfo.addEventListener(
        Event.COMPLETE, textureCompleteHandler );
    loader.contentLoaderInfo.addEventListener(
        IOErrorEvent.IO_ERROR, textureErrorHandler );
    loader.load( new URLRequest( "cube-texture.png" ) );
}

private function textureCompleteHandler(event : Event ):void {
    var loaderInfo:LoaderInfo=event.currentTarget as LoaderInfo;
    cubeTexture=Bitmap(loaderInfo.content);
    createCube();
}

private function textureErrorHandler( event :IOErrorEvent ):void {
    trace( "テクスチャがロードできませんでした。");
}

```

これでキューブを作成する準備ができました。Cube クラスのコンストラクタは引数を渡さずにインスタンス化できますが、われわれは Blender 内でキューブを拡大しなかったため、scaleX、scaleY、scaleZ プロパティを設定して拡大します（拡大しないと非常に小さいキューブが作成されます）。

テクスチャを適用するにはまず、BitmapMaterial オブジェクトを作成し、ロードしたビットマップの bitmapData オブジェクトをそのコンストラクタに渡します。この BitmapMaterial オブジェクトを使用するとキューブの Appearance インスタンスが作成できます。

### リスト3: キューブの作成

```

private function createCube():void {
    blenderCube = new Cube();
    blenderCube.scaleX=30;
    blenderCube.scaleY=30;
    blenderCube.scaleZ=30;
    var material:BitmapMaterial=new BitmapMaterial(cubeTexture.bitmapData);
    blenderCube.appearance=new Appearance(material);
    scene.root.addChild( blenderCube );
}

```

```
addEventListener( Event.ENTER_FRAME, enterFrameHandler );  
}
```

ここまででオブジェクトを作成し、テクスチャを割り当て、3D シーンに追加できました。しかしまだスクリーンには何も表示されません。最後の手順はレンダーループを作成することです。ここではただキューブの回転をつづけるだけにし、最後に Scene3D オブジェクトの render()メソッドを呼び出します(リスト4参照)。

#### リスト 4: レンダーループ

```
protected function enterFrameHandler( event : Event ):void {  
    blenderCube.rotateZ++;  
    blenderCube.rotateY++;  
    scene.render();  
}
```

ここまでのコードをすべてまとめるとリスト5のようになります。最終的な出力は図 18 のようになります。

#### リスト5: 最終的な Main クラス

```
package {  
  
    import flash.display.Bitmap;  
    import flash.display.Loader;  
    import flash.display.LoaderInfo;  
    import flash.display.Sprite;  
    import flash.events.Event;  
    import flash.events.IOErrorEvent;  
    import flash.net.URLRequest;  
  
    import sandy.core.Scene3D;  
    import sandy.core.scenegraph.Camera3D;  
    import sandy.core.scenegraph.Group;  
    import sandy.materials.Appearance;  
    import sandy.materials.BitmapMaterial;  
  
    public class Main extends Sprite {  
  
        private var scene:Scene3D;
```

```

private var blenderCube:Cube;
private var cubeTexture:Bitmap;

public function Main() {
    setup3DScene();
    loadTexture();
}

private function setup3DScene():void {
    var camera:Camera3D=new Camera3D(550,400);
    camera.x=0;
    camera.y=0;
    camera.z=150;
    camera.lookAt( 0, 0, 0 );

    scene = new Scene3D(
        "scene", this, camera, new Group( "rootGroup" ) );
}

private function loadTexture():void {
    var loader : Loader = new Loader();
    loader.contentLoaderInfo.addEventListener(
        Event.COMPLETE, textureCompleteHandler );
    loader.contentLoaderInfo.addEventListener(
        IOErrorEvent.IO_ERROR, textureErrorHandler );
    loader.load( new URLRequest( "cube-texture.png" ) );
}

private function textureCompleteHandler(event : Event):void {
    var loaderInfo:LoaderInfo=
        event.currentTarget as LoaderInfo;
    cubeTexture=Bitmap(loaderInfo.content);
    createCube();
}

private function textureErrorHandler( event :IOErrorEvent ):void {

```

```
        trace( "テクスチャがロードできませんでした。");
    }

    private function createCube():void {
        blenderCube = new Cube();
        blenderCube.scaleX=30;
        blenderCube.scaleY=30;
        blenderCube.scaleZ=30;
        var material:BitmapMaterial=
            new BitmapMaterial(cubeTexture.bitmapData);
        blenderCube.appearance=new Appearance(material);
        scene.root.addChild( blenderCube );
        addEventListener(
            Event.ENTER_FRAME, enterFrameHandler );
    }

    protected function enterFrameHandler( event : Event ):void {
        blenderCube.rotateZ++;
        blenderCube.rotateY++;
        scene.render();
    }
}
}
```

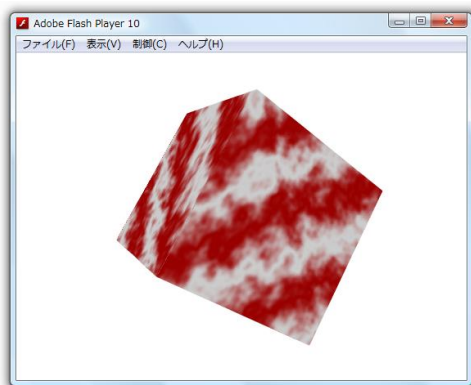


図 18: 最終結果

ここからどこへ行く？

本記事は Blender と 3D Flash の世界への単なる入り口にすぎません。Blender をもっと学びたいと思われるみなさんに、わたしは書籍「[Blender for Dummies](#)」と「[The Essential Blender](#)」を強くおすすめします。

Blender ではどのような驚くべき作品が作れるのかを実感するには、Blender Foundation 制作による2つのオープンムービー、「[Elephant's Dream](#)」と「[Big Buck Bunny](#)」の視聴をおすすめします。