

## Away3D Basics 5 – Primitives(Part 3)

本　　ド　　キ　　ュ　　メ　　ン　　ト　　は　　、  
[http://www.flashmagazine.com/Tutorials/detail/away3d\\_basics\\_5\\_-\\_primitives\\_part\\_3/](http://www.flashmagazine.com/Tutorials/detail/away3d_basics_5_-_primitives_part_3/)で  
公開されている「Away3D Basics 5 – Primitives(Part 3)」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。

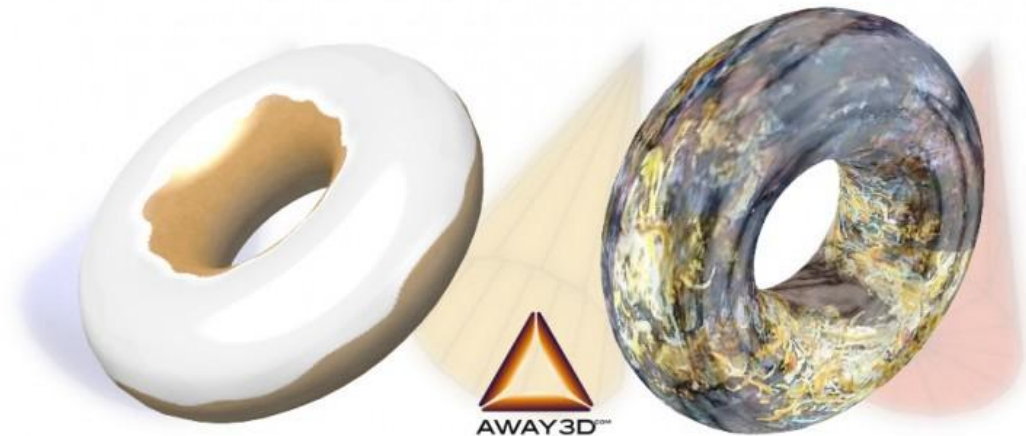
<http://www.himco.jp/>

[knagai@himco.jp](mailto:knagai@himco.jp)

(2009/11)

原文は、

[http://www.flashmagazine.com/Tutorials/detail/away3d\\_basics\\_5\\_-\\_primitives\\_part\\_3/](http://www.flashmagazine.com/Tutorials/detail/away3d_basics_5_-_primitives_part_3/)  
で読むことができます。



## Away3D の基本5: プリミティブ (パート3)

本チュートリアルは、Away3D プリミティブを取り上げた3つめにして最後のチュートリアルです。ここではほかのプリミティブと趣が異なり、あまり使用することもないと思われるプリミティブを扱います。Cone や Torus、SeaTurtle はなくても特段支障のないプリミティブですが、GridPlane と LineSegment は非常に有用なデバッグ用ツールです。

本チュートリアルで取り上げるプリミティブの中には、多くの 3D エンジンやモデリングパッケージに備わっているものもありますが、使用範囲が非常に限られているので、前の2つのチュートリアルのような“リッチな”サンプルは紹介しません。とはいえ使用方法を示すソースコードは提供します。

### 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。

本チュートリアルのサンプルの中にはテクスチャを使うものがあります。テクスチャとマテリアルについては別のチュートリアルでもっと詳しく見ていきますが、Flash CS3 や CS4 でソースファイルを作成したい場合には、

[このチュートリアル](#)を先に読んでください。

## 円錐 (Cone)

円錐はアイスクリームやトラフィックコーン(道路に置く赤い円錐)やパーティハットなどに使用できますが、使用範囲は限られています。円錐 (Cone)には radius、segmentsH、segmentsW などのプロパティがあります。また高さ (height) や口を開くかどうか (openEnded、つまり底面のあるなし) も指定でき、その内部を見ることもできます。



ムービー: [Basic08\\_cone.as](#)

このサンプルでは、円錐の作成時に指定できるいくつかの組み合わせを示しています。ソースコードを見て、そのプロパティが結果にどのように影響しているかを確認してみてください。

## Basic08\_cone

```
package {  
  
    import away3d.containers.View3D;  
    import away3d.core.light.DirectionLight;  
    import away3d.lights.DirectionLight3D;  
    import away3d.materials.PhongColorMaterial;  
    import away3d.primitives.Cone;  
  
    import flash.display.Sprite;  
    import flash.events.Event;  
    [SWF(width="500", height="220",  
        frameRate="60", backgroundColor="#FFFFFF")]  
  
    public class Basic08_cone extends Sprite {
```

```

private var view:View3D;
private var cone1:Cone;
private var cone2:Cone;
private var cone3:Cone;
private var cover:Cover;
private var light:DirectionalLight3D;

public function Basic08_cone() {

    view = new View3D({x:250,y:110});
    addChild(view);

    // 3つのコーンを異なるオプションで作成
    // 赤地に黒線、segmentsH は 3
    cone1 = new Cone(
{material:"red#black",segmentsH:3,segmentsW:15,height:100,radius:50,x:-170});
    view.scene.addChild(cone1);
    // オレンジ地に黒線、segmentsH1、底面なし、両面あり
    cone2 = new Cone(
{material:"orange#black",segmentsH:1,segmentsW:15,
height:100,radius:50,x:0,openEnded:true,bothsides:true});
    view.scene.addChild(cone2);

    var mat:PhongColorMaterial =
        new PhongColorMaterial(0x7777ff);
    // Phong マテリアルあり、segmentsH は 1
    cone3 = new Cone(
{material:mat,segmentsH:1,segmentsW:15,height:100,radius:50,x:170});
    view.scene.addChild(cone3);

    // Phong マテリアルが見えるようにライトを追加
    light = new DirectionalLight3D({x:100,y:300,z:500});
    light.color = 0xffffffff;
    light.specular = 1;
    light.diffuse = .75;

```

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

        view.render();
        cover = new Cover(this,500,220);
        addChild(cover);

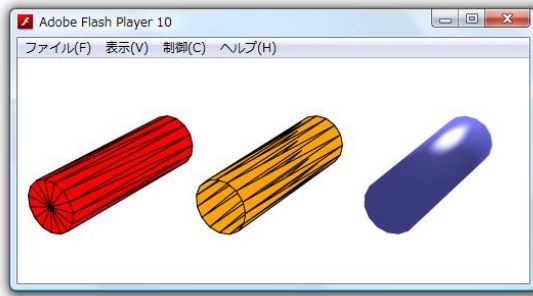
        this.addEventListener(Event.ENTER_FRAME,render);
    }

    private function render(e:Event):void {
        if(!cover.visible) {
            // コーンを回転
            cone1.rotationX += 1;
            cone1.rotationY += .5;
            cone2.rotationX += 1;
            cone2.rotationY += .5;
            cone3.rotationX += 1;
            cone3.rotationY += .5;

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

## 円柱 (Cylinder)

円柱のプロパティは、次のサンプルからも分かるように、ほとんどが円錐と同じです。



ムービー: [Basic08\\_cylinder.as](#)

円柱は円錐よりも有用かもしれませんが、ジオメトリの観点からいうと“高くつく”プリミティブです。Windows XP のスクリーンセーバーで、画面上にランダムにリアルタイムで描かれるパイプを思い出してください(訳者注;わたしには何のことか分かりません)。これは簡単そうに見えますが、リアルタイム 3D エンジンには相当にハードな作業なのです。これまでのサンプルで見てきたように、回転して見えるすべてのものには多数の三角形が必要で、円柱もその例外ではありません。こういったことをふまえ家庭用ゲーム機を考えると、ゲームに出てくるモデルのサーフェイスに丸みがつき始めたのは、ここ2、3年のことです。

Basic08\_cylinder.as

```
package {

    import away3d.containers.View3D;
    import away3d.lights.DirectionLight3D;
    import away3d.materials.PhongColorMaterial;
    import away3d.primitives.Cylinder;

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

    [SWF(width="500", height="220",
        frameRate="60", backgroundColor="#FFFFFF")]

    public class Basic08_cylinder extends Sprite {

        private var view:View3D;
        private var cylinder1:Cylinder;
        private var cylinder2:Cylinder;
        private var cylinder3:Cylinder;
```

```

private var cover:Cover;
private var light:DirectionalLight3D;

public function Basic08_cylinder() {

    view = new View3D({x:250,y:110});
    addChild(view);

    // 3つのシリンダーを異なるオプションで作成
    // 赤地に黒線、segmentsH は 1
    cylinder1 = new Cylinder(
{material:"red#black",segmentsH:1,segmentsW:15,height:70,radius:30,x:-170});
    view.scene.addChild(cylinder1);
    // オレンジ地に黒線、segmentsH1、底面なし、両面あり
    cylinder2 = new Cylinder(
{material:"orange#black",segmentsH:1,segmentsW:15,
    height:70,radius:30,x:0,openEnded:true,bothsides:true});
    view.scene.addChild(cylinder2);

    var mat:PhongColorMaterial =
        new PhongColorMaterial(0x7777ff);
    // Phong マテリアルあり、segmentsH は 1
    cylinder3 = new Cylinder(
{material:mat,segmentsH:1,segmentsW:15,height:70,radius:30,x:170});
    view.scene.addChild(cylinder3);

    // Phong マテリアルが見えるようにライトを追加
    light = new DirectionalLight3D({x:100,y:300,z:500});
    light.color = 0xffffffff;
    light.specular = 1;
    light.diffuse = .75;
    view.scene.addChild(light);

    view.render();
    cover = new Cover(this,500,220);
    addChild(cover);
}

```

```

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

    private function render(e:Event):void {
        if(!cover.visible) {
            //シリンダーを回転
            cylinder1.rotationX += 1;
            cylinder1.rotationY += .5;
            cylinder2.rotationX += 1;
            cylinder2.rotationY += .5;
            cylinder3.rotationX += 1;
            cylinder3.rotationY += .5;

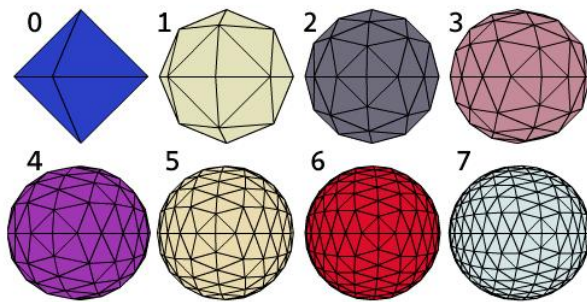
            // ビューをレンダリング
            view.render();
        }
    }
}

```

### GeodesicSphere(ジオデシック・スフィア)

ジオデシック・スフィア(三角形でできた球体)はジオデシック・ドーム(三角形の部材を組み合わせた半球形の構造)とも呼ばれます。これは建築構造に用いられてきた形状で([Spaceship Earth at Epcot](#)を参照)、従来の球プリミティブと比べ、ユニークなプロパティを持っています。Away3D のジオデシック・スフィアは2つのピラミッドを合わせたような基本形状から始まります。その後この形状は、基本形状の各三角形を三角形に分割し、球に近づけていくことで、精密になっていきます。

三角形を分割する量が多ければ多いほど、形状は丸くなり球に近づいていきます。下のイメージでは、各丸の左に fractures 設定(球の割れ目数を定義)を示しているので、この設定が球の丸さに影響する様子が分かります。



ムービー: [Basic08\\_geodesicsphere\\_refine.as](#)

Basic08\_geodesicsphere\_refine.as (上記リンクコードに、4つの球を追加しています)

```
package {  
  
    import away3d.cameras HoverCamera3D;  
    import away3d.containers View3D;  
    import away3d.primitives GeodesicSphere;  
  
    import flash.display Sprite;  
    import flash.events Event;  
  
    [SWF(width="500", height="400",  
        frameRate="50", backgroundColor="#FFFFFF")]  
    public class Basic08_geodesicsphere_refine extends Sprite {  
  
        private var cam:HoverCamera3D;  
        private var View:View3D;  
        private var cover:Cover;  
  
        private var geoSphere1:GeodesicSphere;  
        private var geoSphere2:GeodesicSphere;  
        private var geoSphere3:GeodesicSphere;  
        private var geoSphere4:GeodesicSphere;  
        private var geoSphere5:GeodesicSphere;  
        private var geoSphere6:GeodesicSphere;  
        private var geoSphere7:GeodesicSphere;  
        private var geoSphere8:GeodesicSphere;
```

```
public function Basic08_geodesicsphere_refine() {

    cam = new HoverCamera3D();
    cam.z = -1000;
    cam.panangle = 0;
    cam.tiltangle = 0;
    cam.targetpanangle = 0;
    cam.targettiltangle = 0;
    cam.mintiltangle = -90;
    cam.zoom = 5;

    View = new View3D({camera:cam,x:250,y:100});
    addChild(View);

    // 8個の GeodesicSphere を、異なる fractures で作成
    geoSphere1 = new GeodesicSphere({radius:90,x:340});
    geoSphere1.fractures = 0;
    View.scene.addChild(geoSphere1);

    geoSphere2 = new GeodesicSphere({radius:90,x:130});
    geoSphere2.fractures = 1;
    View.scene.addChild(geoSphere2);

    geoSphere3 = new GeodesicSphere({radius:90,x:-80});
    geoSphere3.fractures = 2;
    View.scene.addChild(geoSphere3);

    geoSphere4 = new GeodesicSphere({radius:90,x:-290});
    geoSphere4.fractures = 3;
    View.scene.addChild(geoSphere4);

    geoSphere5 = new GeodesicSphere({radius:90,x:340, y:-250});
    geoSphere5.fractures = 4;
    View.scene.addChild(geoSphere5);

    geoSphere6 = new GeodesicSphere({radius:90,x:130, y:-250});
```

```

        geoSphere6.fractures = 5;
        View.scene.addChild(geoSphere6);

        geoSphere7 = new GeodesicSphere({radius:90,x:-80, y:-250});
        geoSphere7.fractures = 6;
        View.scene.addChild(geoSphere7);

        geoSphere8 = new GeodesicSphere({radius:90,x:-290, y:-250});
        geoSphere8.fractures = 7;
        View.scene.addChild(geoSphere8);

        cam.hover();
        View.render();

        cover = new Cover(this,500,400);
        addChild(cover);

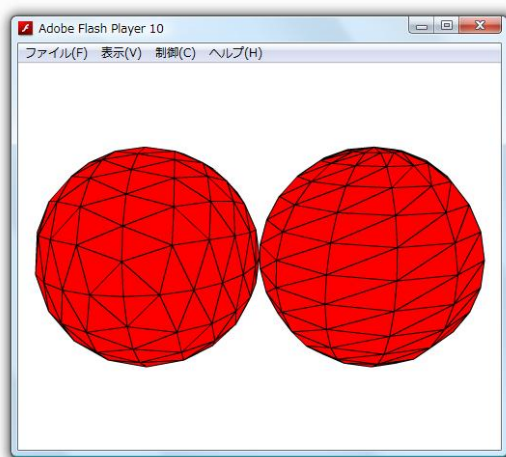
        this.addEventListener(
            Event.ENTER_FRAME,onEnterFrame);
    }

    private function onEnterFrame(e:Event):void {
        if(!cover.visible) {
            // 各球を回転
            geoSphere1.rotationX += 1;
            geoSphere2.rotationX += 1;
            geoSphere3.rotationX += 1;
            geoSphere4.rotationX += 1;
            geoSphere5.rotationX += 1;
            geoSphere6.rotationX += 1;
            geoSphere7.rotationX += 1;
            geoSphere8.rotationX += 1;
            // ビューをレンダリング
            cam.hover();
            View.render();
        }
    }

```

```
    }  
  }  
}
```

前のイメージから分かるように、これは、通常の Sphere プリミティブを使って球を作成する方法とはまったく異なります。GeodesicSphere オブジェクトの三角形はどれもほぼ同じサイズなので、同じ三角形の数を使った場合、GeodesicSphereの方がSphereよりも丸くなります。下のサンプルにマウスオーバーさせて、GeodesicSphereとSphereを比較してみてください。



ムービー: [Basic08\\_geodesicsphere.as](http://Basic08_geodesicsphere.as)

またジオデシック・スフィアは[このコードのように](#)、ほかのプリミティブにもとづいても作成できます。

Basic08\_geodesicsphere.as

```
package {  
  
    import away3d.cameras HoverCamera3D;  
    import away3d.containers View3D;  
    import away3d.primitives GeodesicSphere;  
    import away3d.primitives Sphere;  
  
    import flash.display Sprite;  
    import flash.events Event;  
    import flash.events KeyboardEvent;  
    import flash.ui Keyboard;
```

```
[SWF(width="500", height="400",
      frameRate="50", backgroundColor="#FFFFFF")]
public class Basic08_geodesicsphere extends Sprite {

    private var cam:HoverCamera3D;
    private var View:View3D;
    private var cover:Cover;
    private var lastKey:uint;
    private var keyIsDown:Boolean = false;
    private var sphere:Sphere;
    private var geoSphere:GeodesicSphere;

    public function Basic08_geodesicsphere() {

        cam = new HoverCamera3D();
        cam.z = -1000;
        cam.panangle = 0;
        cam.tiltangle = 0;
        cam.targetpanangle = 0;
        cam.targettiltangle = 0;
        cam.mintiltangle = -90;
        cam.zoom = 5;

        View = new View3D({camera:cam,x:250,y:200});
        addChild(View);

        // GeodesicSphere を作成
        geoSphere = new GeodesicSphere(
            {radius:200,material:"red#black",x:200});
        geoSphere.fractures = 5;
        //フェースの数は 288
        trace(geoSphere.faces .length)
        View.scene.addChild(geoSphere);

        // Sphere を作成
        sphere = new Sphere(
```

```

        {radius:200,material:"red#black",x:-200,segmentsW:12,segmentsH:13});
        //フェースの数は 288
        trace(sphere.faces .length)
        View.scene.addChild(sphere);

        cam.hover();
        View.render();

        cover = new Cover(this,500,400,
"Roll over and Click to activate. Use Space key to show both and arrow keys to toggle.");
        addChild(cover);

        this.stage.addEventListener(
            KeyboardEvent.KEY_DOWN,onKeyDown);
        this.stage.addEventListener(
            KeyboardEvent.KEY_UP,onKeyUp);
        this.addEventListener(
            Event.ENTER_FRAME,onEnterFrame);
    }

    private function onEnterFrame(e:Event):void {
        if(!cover.visible) {
            if(keyIsDown){
                // 左矢印キーで GeodesicSphere を表示
                // 右矢印キーで Sphere を表示
                // スペースキーで両方を表示
                switch(lastKey){
                    case Keyboard.LEFT:
                        showLeft(); break;
                    case Keyboard.RIGHT:
                        showRight(); break;
                    case Keyboard.SPACE:
                        showBoth(); break;
                }
            }
        }
        // 両方を回転
    }

```

```

        geoSphere.rotationX += 1;
        sphere.rotationX += 1;
        // ビューをレンダリング
        cam.hover();
        View.render();
    }
}

private function showBoth():void {
    geoSphere.visible = true;
    sphere.visible = true;
}

private function showLeft():void {
    geoSphere.visible = true;
    sphere.visible = false;
}

private function showRight():void {
    geoSphere.visible = false;
    sphere.visible = true;
}

private function onKeyDown(e:KeyboardEvent):void {
    lastKey = e.keyCode;
    keyIsDown = true;
}

private function onKeyUp(e:KeyboardEvent):void {
    keyIsDown = false;
}
}
}

```

RoundedCube(角が丸い直方体)

わたしの知る限り、このプリミティブは Away3D だけのもので、その単純性において球に似ています。その外見をよくするにはかなりの数の三角形が必要ですが、その結果は強い印象を与えます。



このイメージをクリックして、Fabrice Closier のブログに進み、このプリミティブを操作してみてください。Fabrice は Away3D チームのコアコーダーの1人です。

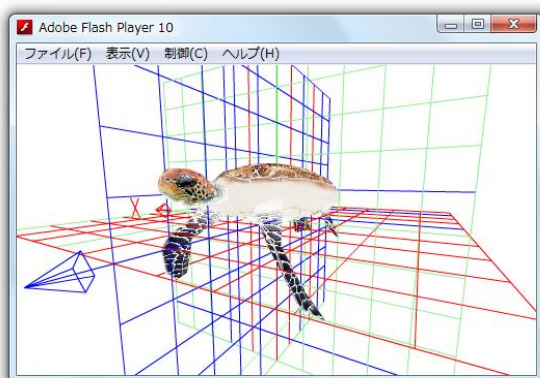
このプリミティブはあるユーザーの求めに応じて最近追加されました。もしみなさんが Away3D に含まれていないプリミティブが必要な場合には、Away3D のフォーラムにその要求を投稿してみましょう。それが有用だと判断されたときには、チームメンバーの誰かが、この RoundedCube のように作成するかもしれません。そのプリミティブを自分で作成できるだけのスキルのある方は、もちろん Away3D エンジンにそのコードを提供することも歓迎されます。

### SeaTurtle(海がめ)

SeaTurtle は、ロードしたモデルにもとづきコンパクトな ActionScript 3.0 クラスを生成する AS3 エクスポーターが初めて出力したクラスです。これまで何度も更新されてきたので今では非常にパワフルになっています。コンパクトな 3D プロジェクトが必要な場合には、AS3 エクスポーターによってファイルサイズやロード時間が改善できないか、テストしてみるべきです。SeaTurtle は実際にはプリミティブではありませんが、Away3D の初期には楽しみをあたえるマスコットのようなものでした。形状が複雑でたくさんのフェースを持っているので、照明のテストにも役立ちます。海がめのテクスチャは/techdemos/images/フォルダにあり、これは Away3D サブバージョン・レポジトリからダウンロードできます。SeaTurtle は次の GridPlane サンプルで使用します。

### GridPlane(3D のグリッド・プリミティブ)

このプリミティブはシーンを設定するときに大きな助けになります。これは単なる矩形のグリッドですが、ほとんどの 3D プログラムに含まれているのには理由があります。下のモデルを見てください。



ムービー: [Basic08\\_gridplane.as](#)、イメージ: [seaturtle.jpg](#)

このようにトライデントとグリッドを追加すると、容易に海がめのセンターが分かり、グリッドの目盛りが分かると、モデルを視覚的に分析することができます(このサンプルでは、海がめの全長はおおよそ8目盛り分です)。

Torus(円環)

円環は多くの 3D ツールに備わっている 3D 形状で、[ドーナツ](#)や救命具に使用できます。しかし丸く三角形の数が多いので、シェーダのテストにも有効です。回転する Torus に、反射や照明をシミュレーションする環境シェーダを適用すると、見栄えが一変します。



ムービー: [Basic08\\_torus.as](#) イメージ: [doughnut.jpg](#) [mandelbrot.jpg](#) [marble.jpg](#)

この Torus のサンプルではよく使用される segmentsW と segmentsH ではなく、segmentsR と segmentsT を使用している点に注意してください。

```
// 外側と内側の円をいくつのセグメントで構成するか
torus1.segmentsR = 20;
// チューブ(管)をいくつのセグメントで構成するか
torus1.segmentsT = 10;
```

Basic08\_torus.as

```
package {

    import away3d.containers.View3D;
    import away3d.lights.DirectionLight3D;
    //import away3d.materials.Dot3BitmapMaterial;
    import away3d.materials.EnviroBitmapMaterial;
    import away3d.materials.PhongBitmapMaterial;
    import away3d.primitives.Torus;
    import away3d.core.utils.Cast;

    //import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Sprite;
    import flash.events.Event;

    [SWF(width="500", height="300",
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic08_torus extends Sprite {

        private var view:View3D;
        private var torus1:Torus;
        private var torus2:Torus;
        private var cover:Cover;
        private var light:DirectionLight3D;

        [Embed(source="resources/mandelbrot.jpg")]
        private var enviroTexture:Class;
        [Embed(source="resources/doughnut.jpg")]
        private var doughTexture:Class;
        [Embed(source="resources/marble.jpg")]
```

```
private var grassTexture:Class;

public function Basic08_torus() {

    view = new View3D({x:250,y:150});
    addChild(view);

    // Torus を作成し、派手なマテリアルを適用
    var enviro:BitmapData = Cast.bitmap(enviroTexture);
    var grass:BitmapData = Cast.bitmap(grassTexture);
    var mat:EnviroBitmapMaterial =
        new EnviroBitmapMaterial(enviro,grass);
    mat.reflectiveness = 0.3;
    torus1 = new Torus({material:mat,x:-140,radius:80});
    torus1.segmentsR = 20;
    torus1.segmentsT = 10;
    view.scene.addChild(torus1);

    // Torus をもう1つ作成し、おいしそうなテクスチャを適用
    var doughTexture:BitmapData = Cast.bitmap(doughTexture);
    var mat2:PhongBitmapMaterial =
        new PhongBitmapMaterial(doughTexture);
    //mat2.specular = 0.5;
    torus2 =
        new Torus({material:mat2,radius:80,x:140,rotationX:90});
    torus2.segmentsR = 20;
    torus2.segmentsT = 10;
    view.scene.addChild(torus2);

    // Phong マテリアルが見えるように、照明を追加
    light = new DirectionalLight3D({x:100,y:300,z:-500});
    light.color = 0xffffffff;
    light.specular = 1;
    light.diffuse = .75;
    view.scene.addChild(light);
```

```
view.render();
cover = new Cover(this,500,300);
addChild(cover);

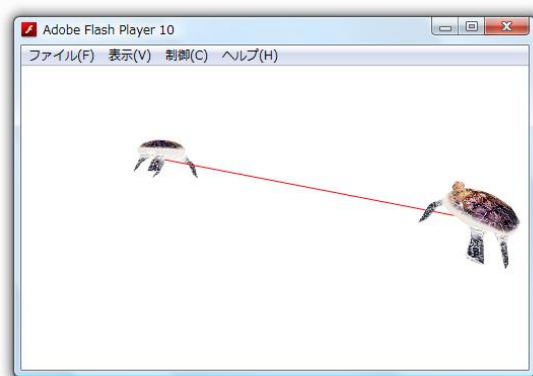
this.addEventListener(Event.ENTER_FRAME,render);
}

private function render(e:Event):void {
    if(!cover.visible) {
        // Torus を回転
        torus1.rotationX += 1;
        torus1.rotationY += .5;
        torus2.rotationX -= 1;
        torus2.rotationY -= .5;

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

## LineSegment

これは線のプリミティブで、次のサンプルのように2点間を線で描画するときに使用できます。次のサンプルではマウスをムービーの上に移動させると、赤い線でつながれた海がめがランダムに回転して移動します。



ムービー: [Basic08\\_linesegment.as](#) image: [seaturtle.jpg](#)

訳者注:

このサンプルではこれまで使ってきた Away3D では、赤い線が移動しませんでした。が、`away3d¥branches¥3.0.0¥src` に変えると移動するようになりました。