

Away3D Basics 4 – Manipulating 3D objects

本　　ド　　キ　　ュ　　メ　　ン　　ト　　は　　、
[http://www.flashmagazine.com/Tutorials/detail/away3d_basics_4_-_manipulating_3d_o
bjects/](http://www.flashmagazine.com/Tutorials/detail/away3d_basics_4_-_manipulating_3d_objects/)で公開されている「Away3D Basics 4 – Manipulating 3D objects」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/>

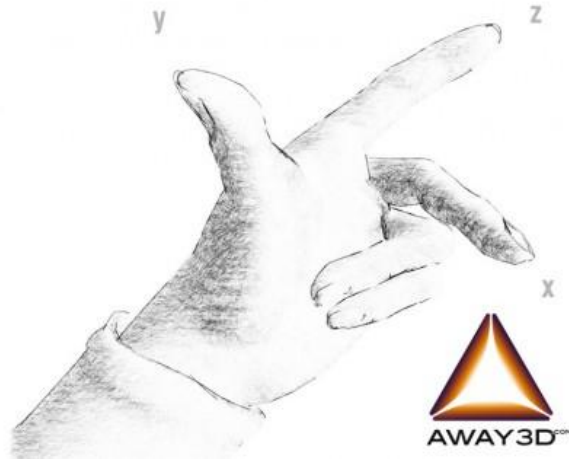
knagai@himco.jp

(2009/11)

原文は、

[http://www.flashmagazine.com/Tutorials/detail/away3d_basics_4_-_manipulating_3d_o
bjects/](http://www.flashmagazine.com/Tutorials/detail/away3d_basics_4_-_manipulating_3d_o
bjects/)

で読むことができます。



Away3D の基本4: 3D オブジェクトの操作

3D オブジェクトは、Flash の2次元オブジェクトと同じように、位置取りや回転、拡大縮小、グループ化などが行えます。本チュートリアルでは 3D 空間での操作方法と、みなさんのお好きなトゥイーンパッケージを使ったオブジェクトのトゥイーンを探っていきます。

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。

ではこれから行う実験の対象として、次のような基本的な Away3D クラス、Basic07_cube を作成しましょう。

Basic07_cube.as

```
package {  
  
    import away3d.containers.View3D;
```

```
import away3d.primitives.Cube;

import flash.display.Sprite;

[SWF(width="500", height="400",
     frameRate="60", backgroundColor="#FFFFFF")]
public class Basic07_cube extends Sprite {

    public function Basic07_cube() {
        // ビューポートの作成
        var view:View3D = new View3D({x:250,y:200});
        addChild(view);

        // キューブを作成し 3D シーンに配置
        var cube:Cube = new Cube();
        view.scene.addChild(cube);

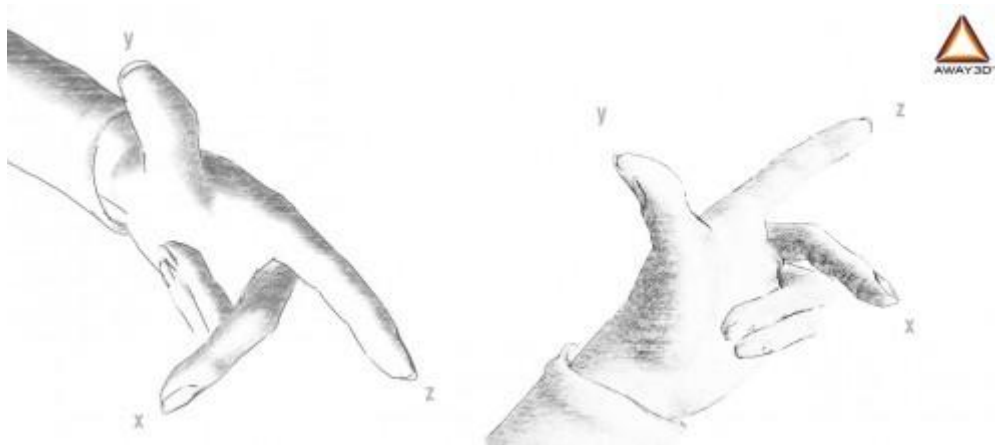
        // ビューをレンダリング
        view.render();
    }
}
}
```

ムービー: [Basic07_cube.as](#)

本チュートリアルではこのファイルを開始点とし、これを使ってさまざまなテストを行います。

自分の手を使おう！

みなさんの周囲はすべて 3D なので、3D 空間で物を動かすのは簡単そうに思えます。実際それほど難しくはありませんが、手の助けを借りると非常に役に立ちます。オブジェクトを移動または回転させる方法を理解するには、手の指をそれぞれ下図のような方向に向けます。



親指は y 軸、人差し指は z 軸、中指は x 軸と考えてください。いたって単純ですが、この何でもない工夫が複雑な 3D オブジェクトの動きの理解に役立つのです。

NOTE: Away3D (やそのほかの 3D エンジン) では [Flash が採用している](#) 逆の座標システムは使用されず、Away3D では数学や幾何学で使われる直交座標システムが使用されます。このシステムでは x 値は正数が原点右方向に延び、負数が左方向に伸びます。y 値は正数が原点上方向に延び、負数が下方向に伸びます。したがって Flash での位置取りで学んだことはここでは忘れる必要があります。

3D 空間での移動

Away3D のすべてのオブジェクトは、2D のオブジェクトと同じように x と y プロパティの設定によって移動できますが、3D なので z 位置も指定できます。

```
cube.x = 100;  
cube.y = -100;  
cube.z = 100;
```

この位置には正の値でも負の値でも設定できます。その結果スクリーンでの位置は 3D のワールドセンターとビューが基準になります。ワールドセンターはビューポート (View3D) の作成時に設定しますが、通常は初期ビューの真ん中に配置します。

また位置の設定には、moveTo() や position プロパティが便利です。

```
// Number3D は 3D 空間の点  
cube.position = new Number3D(100,-100,300);
```

```
cube.moveTo(100,-100,300);
```

この2行の結果は同じです。異なるのは position プロパティには新しい Number3D を与える必要があり、moveTo()メソッドには通常の数値を指定することです。

オブジェクトをシーン内の別のオブジェクトの方向や位置に向かせたい場合には、lookAt()メソッドを使用します。

```
cube.lookAt( new Number3D(0,0,0));
```

これらの方法を組み合わせると、オブジェクトに別のオブジェクトを“追跡”させたいときに非常に便利です。別のオブジェクトの position を調べると Number3D が返されるので、これを lookAt()と併用します。

```
cube.lookAt( sphere.position );
```

訳者注:

Sphere クラスの position プロパティは Number3D クラスのプロパティで、その 3D オブジェクトの位置を親 ObjectContainer3D のローカル座標を基準に定義します。

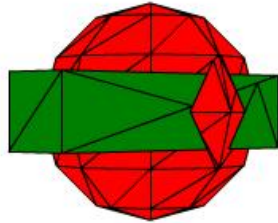
Away3D ではまた、オブジェクトを移動させるための特別なメソッドも用意されています。

```
cube.moveForward(X);  
cube.moveBackward(X);  
cube.moveUp(X);  
cube.moveDown(X);  
cube.moveLeft(X);  
cube.moveRight(X);
```

これらはその名前通りの働きをし、オブジェクトを X 単位分だけ移動させます。これらのメソッドには、単に x、y、z プロパティを設定するだけではない大きなメリットがあります。それは、オブジェクトはその親のオブジェクトの軸ではなく、自分自身の軸に沿って移動するというメリットです。サンプルを見てみましょう。

cube.zを使用中:
cubeはその親オブジェクト(ビュー)を基準に移動

移動方法の切り替え



ムービー: [Basic07_moveForward.as](#)

たとえば宇宙船が自由に移動できるスペースシューティングゲームを作成するようときには、x、y、z の代わりにこの方法を使用すると、正確な結果がかなり簡単に得られるようになります。

```
package {

    import away3d.containers.View3D;
    import away3d.core.render.Renderer;
    import away3d.primitives.Cube;
    import away3d.primitives.Sphere;
    import away3d.test.Button;

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

    [SWF(width="500", height="400",
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic07_moveForward extends Sprite {

        private var view:View3D;
        private var cube:Cube;
        private var toggle:Boolean = false;
        private var bt:Button;
```

```
private var cubeSpeed:Number = 5;
private var label:TextField;

public function Basic07_moveForward() {
    // 2つの移動方法を切り替えるボタンを追加
    bt = new Button("移動方法の切り替え",140);
    bt.x = 10;
    bt.y = 60;
    bt.addEventListener(MouseEvent.CLICK,resetCube);
    this.addChild(bt);

    // テキストフィールドを追加
    label = new TextField();
    label.autoSize = TextFieldAutoSize.LEFT;
    label.multiline = true;
    label.wordWrap = true;
    label.x = 10;
    label.y = 10;
    label.width = 480;
    label.defaultTextFormat = new TextFormat(
        "Arial", 14, 0x000000);
    this.addChild(label);

    // ビューポートを作成
    view = new View3D(
        {x:250,y:200,
         renderer:Renderer.INTERSECTING_OBJECTS});
    addChild(view);

    // 球を作成に 3D シーンに配置
    var sphere:Sphere = new Sphere(
        {material:"red#black",radius:75});
    view.scene.addChild(sphere);

    // 直方体を作成
    cube = new Cube(
```

```

        {material:"green#black",depth:200,width:50,height:50});
// 違いが分かるように回転させる
cube.rotationY = 45;
// 直方体の位置をリセット
resetCube();
// 直方体を 3D シーンに追加
view.scene.addChild(cube);

// ビューをレンダリング
this.addEventListener(Event.ENTER_FRAME,update);
}

// コンストラクタから 1 回と、ボタンクリックから呼び出される
private function resetCube(e:MouseEvent = null):void {
    // 位置をリセット
    cube.x = 0;
    cube.y = 0;
    cube.z = 0;

    // 移動方法を切り替える
    if(toggle){
        toggle = false;
        label.text =
"cube.moveForward()を使用中: ¥ncube はそれ自体の rotation にしたがって前方移動";
    } else {
        toggle = true;
        label.text =
"cube.z を使用中: ¥ncube はその親オブジェクト(ビュー)を基準に移動";
    }
}

// 毎フレーム実行
private function update(e:Event):void {
    // cube を境界内に維持
    // (飛び出る前に z 方向を反転)
    if(cube.z+cubeSpeed > 200 ||

```

```

        cube.z+cubeSpeed < -200){ cubeSpeed = cubeSpeed*(-1); }

    // toggle 値によって移動方法を変える
    if(toggle){
        cube.z += cubeSpeed;
    } else {
        cube.moveForward(cubeSpeed);
    }

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

```

訳者注:

View3D のコンストラクタで使用している `Renderer.INTERSECTING_OBJECTS` は、Render クラスのプロパティで、三角形を分割して、オブジェクトが交差しているシーンを正確にレンダリングする設定を行います。

回転させる

オブジェクトは、どの軸上でもその回転のプロパティを使って回転させることができます。

```

cube.rotationX = 45;
cube.rotationY = -10;
cube.rotationZ = 200;

```

冒頭の手を使ったトリックを覚えていますか？ 左手を自分の前に置きその指の形を作ると、z 軸を回る回転の理解に役立ちます。

また3つの回転値は便利な `rotateTo()` メソッドを使うと一度に設定できます。

```

cube.rotateTo(45,45,0)

```

拡大縮小する

またオブジェクトはどの軸についても拡大縮小ができます。

```
cube.scaleX = 2;  
cube.scaleY = .5;  
cube.scaleZ = 1;
```

拡大縮小の設定では、1 がそのオブジェクトの元のサイズになります。拡大縮小を 2 に設定することは2倍に拡大することで、0.5に設定することは半分に縮小することです。単にオブジェクト全体を拡大縮小するときには、scale()メソッドが使用できます。

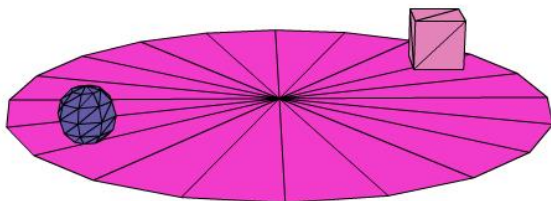
```
cube.scale(2);
```

これによって cube は2倍に拡大されます。

オブジェクトをグループ化する

複数のオブジェクトを同時に移動させた場合があります。これは、オブジェクトをグループに追加することで簡単に行えます。Away3d でのオブジェクトのグループは”ObjectContainer3D”(シーン内のほかの 3D オブジェクト用の 3D オブジェクトコンテナノード)と呼ばれ、オブジェクトをシーン追加するのと同じシンタックス(書き方)で追加します。

オブジェクトのグループを作成するには、まず新しい ObjectContainer3D を作成し、それにシーンを追加します。次のサンプルでは、RegularPolygon インスタンスの“皿”(disc)と立方体(cube)、球(sphere)をグループ(group)に追加し、毎フレーム 0.5 度回転させています。また cube だけ、つねにビューワ(見る人)の方をつねに向くように、逆回転させています。



ムービー: [Basic07_group.as](#)

このコンテナは視覚表現を持っていません(表示されない)が、ほかの 3D オブジェクトと同じように移動させ

ることができます。

Basic07_group.as

```
package {

    import away3d.containers.ObjectContainer3D;
    import away3d.containers.View3D;
    import away3d.core.math.Number3D;
    import away3d.primitives.Cube;
    import away3d.primitives.RegularPolygon;
    import away3d.primitives.Sphere;

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

    [SWF(width="500", height="400",
        frameRate="60", backgroundColor="#FFFFFF")]
    public class Basic07_group extends Sprite {

        private var view:View3D;
        private var label:TextField;

        private var group:ObjectContainer3D;
        private var disc:RegularPolygon;
        private var cube:Cube;
        private var sphere:Sphere;

        public function Basic07_group() {
            // ビューポートの作成
            view = new View3D({x:250,y:200});
            addChild(view);
            // カメラの移動
            view.camera.y = 300;
            // カメラがシーンの中心を向くように設定
            // シーンを少し俯瞰するようになる
        }
    }
}
```

```

view.camera.lookAt( new Number3D(0,0,0));

// グループを作成し、シーンに追加
group = new ObjectContainer3D();
view.scene.addChild(group);

// オブジェクトをグループに割り当てる
disc = new RegularPolygon(
    {radius:250,y:-25,sides:20,pushback:true});
group.addChild(disc);
cube = new Cube({depth:50,width:50,height:50,z:200});
group.addChild(cube);
sphere = new Sphere({radius:25,z:-200});
group.addChild(sphere);

// ビューをレンダリング
this.addEventListener(Event.ENTER_FRAME,update);
}

private function update(e:Event):void    {
    // グループを回転させる
    group.rotationY += .5;

    // cube は逆回転させる。これによって cube はつねにビューワの方を向く
    cube.rotationY -= .5;

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

```

3D オブジェクトをトゥイーンさせる

3D オブジェクトのトゥイーンは、Flash でのほかのオブジェクトのトゥイーンとまったく変わりません。好きなトゥイーンの方法をインストールし、希望するようにそのプロパティをトゥイーンさせてみてください。実に簡単で

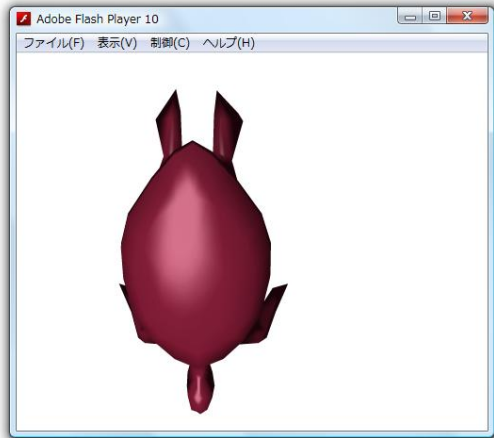
す。

Flash にはネイティブなトゥイーンが備わっていますが、実際には遅いのでその使用は推奨されません。われわれは以下のトゥイーンソリューションをおすすめします。

- [Tweener](#)
- [TweenLite](#)
- [TweenFilterLite](#)
- [TweenMax](#)
- [ZigoEngine \(Fuse\)](#)
- [Twease](#)
- HydroTween
- [gTween](#)
- [TweeGo](#)
- [Go3D](#)
- [GoASAP](#)

Tween エンジンによってどれだけスピードの違いがあるか、[このスピードテスト](#)で実際に調べてみてください。エンジンごとに機能が異なるので、必要なコードの量もさまざまです。シンタックスも同様に異なるので、実際にプロジェクトに追加する前に、行いたい事柄が選択したエンジンでかなえられるのか確認する必要があります。

すでに述べたように 3D のトゥイーンは 2D のトゥイーンと同じです。トゥイーンさせるプロパティを選択し、それを設定してトゥイーンをスタートさせるだけです。次のサンプルでは、Away3D に備わっている SeaTurtle プリミティブを使っています。この海がめをクリックすると、ランダムな位置と回転にトゥイーンします。



ムービー: [Basic07_tweening.as](#)

このサンプルは Greensock の [TweenLite](#) を使って作成しているので、コード自体を使ってテストする場合には、これをプロジェクトフォルダ内に置く必要があります。

訳者注:

具体的には、上記リンクから `greensock-tweening-platform-as3.zip` をダウンロードし解凍します。中に `com` フォルダがあるので、これを FLA ファイルのあるフォルダにコピーします。

Basic07_tweening.as

```
package {

    import away3d.cameras HoverCamera3D;
    import away3d.containers.View3D;
    import away3d.events.MouseEvent3D;
    import away3d.lights.DirectionLight3D;
    import away3d.materials.PhongColorMaterial;
    import away3d.primitives.SeaTurtle;
    import away3d.primitives.Sphere;

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

    //TweenLite のインポート
    import com.greensock.TweenLite;
```

```
[SWF(width="500", height="400",
    frameRate="60", backgroundColor="#FFFFFF")]
public class Basic07_tweening extends Sprite {

    private var view:View3D;
    private var cam:HoverCamera3D;
    private var plaything:SeaTurtle;
    private var cover:Cover;
    private var mat:PhongColorMaterial;
    private var light:DirectionalLight3D;

    public function Basic07_tweening() {
        // カメラの作成
        cam = new HoverCamera3D();
        // カメラは必ずデフォルトの(0, 0, 0)座標から移動させる
        cam.z = -1000;
        cam.panangle = cam.targetpanangle = 0;
        cam.tiltangle = cam.targettiltangle = 0;
        cam.mintiltangle = -90;
        cam.zoom = 5;
        cam.focus = 200;

        // ビューポートの作成
        view = new View3D({camera:cam, x:250,y:200});
        addChild(view);

        // ライトの設定
        light = new DirectionalLight3D();
        light.ambient = .2;
        light.diffuse = .8;
        light.specular = 1;

        light.x = 100;
        light.y = -200;
        light.z = 400;
        light.brightness = 0.5;
    }
}
```

```

view.scene.addChild(light);

// 海がめを作成して追加
mat = new PhongColorMaterial(0xff3366);
plaything = new SeaTurtle(
    {material:mat,segmentsH:15,
     segmentsW:20,height:100,radius:50,x:-80});
plaything.scale(0.5);
view.scene.addChild(plaything);

plaything.addEventListener(
    MouseEvent3D.MOUSE_DOWN,doClick);

cam.hover();
view.render();
cover = new Cover(this,500,400);
addChild(cover);

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

private function doClick(e:MouseEvent3D):void {
    // クリックした位置に球体を作る
    // var sp:Sphere = new Sphere(
        {material:mat,segmentsW:5, segmentsH:5,
         radius:10, x:e.sceneX, y:e.sceneY, z:e.sceneZ});
    // view.scene.addChild(sp);
    var randomX:Number = Math.random()*200-100;
    var randomY:Number = Math.random()*200-100;
    var randomZ:Number = Math.random()*1200-600;
    var randomRot:Number = Math.random()*365;
    var randomCamRot:Number = Math.random()*365;
    // TweenLite の実行
    TweenLite.to(plaything, 1,
        {x:randomX,y:randomY,z:randomZ,rotationX:randomRot});
}

```

```
private function render(e:Event):void    {
    if(!cover.visible) {
        cam.hover();
        view.render();
    }
}
}
```

高度な移動

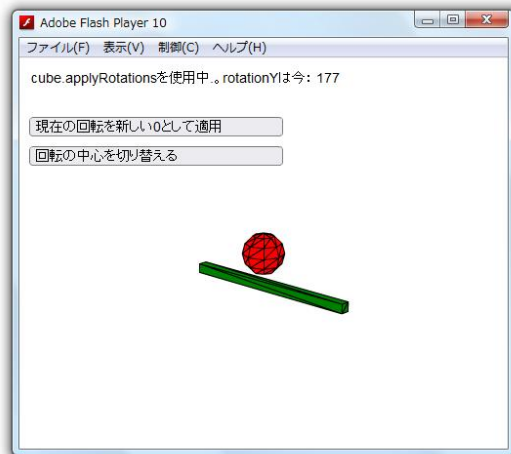
時には初期の回転値をデフォルトから変えたい場合があります。これは、3D アプリケーションから、間違った方向に回転しているモデルをロードしたときに起こることがあります。そのためには希望する回転のプロパティを新しい 0 回転に設定し、それをオブジェクトに適用します。

```
cube.rotationY = 45;
//ローカルの回転を、メッシュの外見を変更せずに、ジオメトリに適用
cube.applyRotation();
```

またオブジェクトの中心を変えたい場合もあります。これによってオブジェクトは別の点を中心に回転するようになります。Flash の 2D オブジェクトでは、これを“基準点”を変更することで行います。3D では基準点のことを“回転の中心”(pivot point)といいます。この基準点を移すには、次のように新しい値を設定します。

```
//オブジェクトが回転する、ローカルの中心点を移動
cube.movePivot(0, 0, -120);
```

次のサンプルは、これらの方法を使って直方体の回転を変更する実験です。



ムービー: [Basic07_apply.as](#)

movePivot()メソッドはモデル内のジオメトリには影響しません。モデルのメッシュを変更する変換を適用したい場合には、applyPosition()メソッドを使用します。

```
// 任意の位置を、メッシュの外見を変更せずに、ジオメトリに適用
cube.applyPosition(0, 0, -120);
```

applyPosition()はモデリングや修正目的、またたとえばオブジェクトを複製し、movePivot()を使って異なるオフセットで示したい場合などに非常に便利です。

Basic07_apply.as

```
package {

    import away3d.containers.View3D;
    import away3d.core.math.Number3D;
    import away3d.core.render.Renderer;
    import away3d.primitives.Cube;
    import away3d.primitives.Sphere;
    import away3d.test.Button;

    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.text.*;
```

```
[SWF(width="500", height="400",
      frameRate="60", backgroundColor="#FFFFFF")]
public class Basic07_apply extends Sprite {

    private var view:View3D;
    private var cube:Cube;
    private var toggle:uint = 0;
    private var cubeSpeed:Number = 5;
    private var label:TextField;

    public function Basic07_apply() {
        // 回転のリセット用ボタンを追加
        // (オブジェクトの値ではなく、メッシュを回転させる)
        var bt1:Button = new Button(
            "現在の回転を新しい 0 として適用",260,20);
        bt1.x = 10;
        bt1.y = 60;
        bt1.addEventListener(MouseEvent.CLICK,applyRotations);
        this.addChild(bt1);

        // 2つの移動方法を切り替えるボタンを追加
        var bt2:Button = new Button("回転の中心を切り替える",260,20);
        bt2.x = 10;
        bt2.y = 90;
        bt2.addEventListener(MouseEvent.CLICK,applyPivot);
        this.addChild(bt2);

        // テキストフィールドを追加
        label = new TextField();
        label.autoSize = TextFieldAutoSize.LEFT;
        label.multiline = true;
        label.wordWrap = true;
        label.x = 10;
        label.y = 10;
        label.width = 480;
        label.defaultTextFormat = new TextFormat(
```

```

        "Arial", 14, 0x000000);
this.addChild(label);

        // ビューポートの作成
        view = new View3D(
        {x:250,y:200,renderer:Renderer.INTERSECTING_OBJECTS});
        addChild(view);
        // カメラを移動
        view.camera.y = 300;
        // カメラはシーンの中心を見る
        view.camera.lookAt( new Number3D(0,0,0));

        // 直方体の作成
        cube = new Cube(
            {material:"green#black",depth:200,width:10,height:10});
        view.scene.addChild(cube);

        // 球の作成
        var sphere:Sphere = new Sphere(
            {material:"red#black",radius:25});
        view.scene.addChild(sphere);

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

    // 現在の回転を新しい0として適用
    private function applyRotations(e:MouseEvent):void {
        // 45 を新しい0として適用
        cube.rotationY = 45;
        cube.applyRotations();
        toggle = 1;
    }

    // メッシュに新しい位置を適用する
    private function applyPivot(e:MouseEvent):void {
        if(toggle == 2){

```

```

        toggle = 0;
        // 回転の中心をリセット
        cube.movePivot(0,0,0);
    } else {
        toggle = 2;
        // 回転の中心を設定
        cube.movePivot(0,0,-120);
        cube.roll(15);
    }
}
private function update(e:Event):void {

    cube.rotationY += 1;
    if(toggle == 1){
        label.text = "cube.applyRotations 使用中。
            rotationY は今: "+Math.floor(cube.rotationY);
    } else if (toggle == 2){
        label.text = "回転の中心を(0, 0, -120)に変更。
            cube.z は今: "+Math.floor(cube.x);
    } else {
        label.text = "標準の回転の中心(0, 0, 0)使用中。
            cube.z は今: "+Math.floor(cube.x);
    }

    view.render();
}
}
}

```

ロール、ピッチ、ヨー

cube.moveForward()や関連するそのほかのメソッドと同じように、3つの回転メソッドがあります。

```

// 3D オブジェクトを、そのローカルの z 軸で回転させる
cube.roll(15);

```

```
// 3D オブジェクトを、そのローカルの x 軸で回転させる
cube.pitch(5);
// 3D オブジェクトを、そのローカルの y 軸で回転させる
cube.yaw(5);
```

これらは回転のプロパティと同じことを行いますが、それ自身のローカルの座標システムにもとづいています。このコードを前のサンプルにコピーして、その動作を観察してみてください。

訳者注:

ロール、ピッチ、ヨーのサンプルとして以下のクラスを作成しました。

```
package {

    import away3d.containers.View3D;
    import away3d.core.math.Number3D;
    import away3d.primitives.Cube;
    import away3d.primitives.Trident;
    import away3d.test.Button;

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

    [SWF(width="500", height="400",
         frameRate="60", backgroundColor="#FFFFFF")]
    public class RollPitchYaw extends Sprite {

        private var view:View3D;
        private var cube:Cube;

        public function RollPitchYaw() {

            var rollBtn:Button = new Button("ロール",80,20);
            rollBtn.x = 10;
            rollBtn.y = 10;
            rollBtn.addEventListener(MouseEvent.CLICK,setRoll);
            this.addChild(rollBtn);
```

```
var pitchBtn:Button = new Button("ピッチ",80,20);
pitchBtn.x = 10;
pitchBtn.y = 40;
pitchBtn.addEventListener(MouseEvent.CLICK,setPitch);
this.addChild(pitchBtn);

var yawBtn:Button = new Button("ヨー",80,20);
yawBtn.x = 10;
yawBtn.y = 70;
yawBtn.addEventListener(MouseEvent.CLICK,setYaw);
this.addChild(yawBtn);

var resetBtn:Button = new Button("リセット",80,20);
resetBtn.x = 10;
resetBtn.y = 100;
resetBtn.addEventListener(MouseEvent.CLICK,reset);
this.addChild(resetBtn);

view = new View3D({x:250,y:200});
addChild(view);

view.camera.y = 300;
view.camera.lookAt( new Number3D(0,0,0));

cube = new Cube(
    {material:"green#black",depth:80,width:50,height:30});
view.scene.addChild(cube);

var axis:Trident = new Trident(180);
view.scene.addChild(axis);

view.render();

}
```

```
private function setRoll(evt:MouseEvent):void {
    cube.roll(15)
    view.render();
}

private function setPitch(evt:MouseEvent):void {
    cube.pitch(5)
    view.render();
}

private function setYaw(evt:MouseEvent):void {
    cube.yaw(5)
    view.render();
}

private function reset(evt:MouseEvent):void {
    // cube.rotationX = cube.rotationY = cube.rotationZ = 0;
    cube.rotateTo(0, 0, 0);
    view.render();
}
}
```