基本的な運動

等速運動

```
addEventListener(Event.ENTER_FRAME,inloop);

function onloop (evt:Event):void{

    ball.x += 100;

    ball.y += 100;

}
```

等速運動はインスタンス変数（プロパティ）の x 座標と y 座標を定数で加算するだけです。これを ENTER_FRAME などで繰り返し処理します。

等加速度運動

```
addEventListener(Event.ENTER_FRAME,inloop);

function onloop (evt:Event):void{

    ball.x += a;

    ball.y += b;

    a += 10;

    b +=10;

}
```

等加速度運動は等速運動に少し手を加えるだけです。定数部分を少しづつ加算していきます。

角度

一般的に角度はラジアンで表されます。ラジアンは円周に沿った円の半径の長さを持つ円弧に対する角度です。

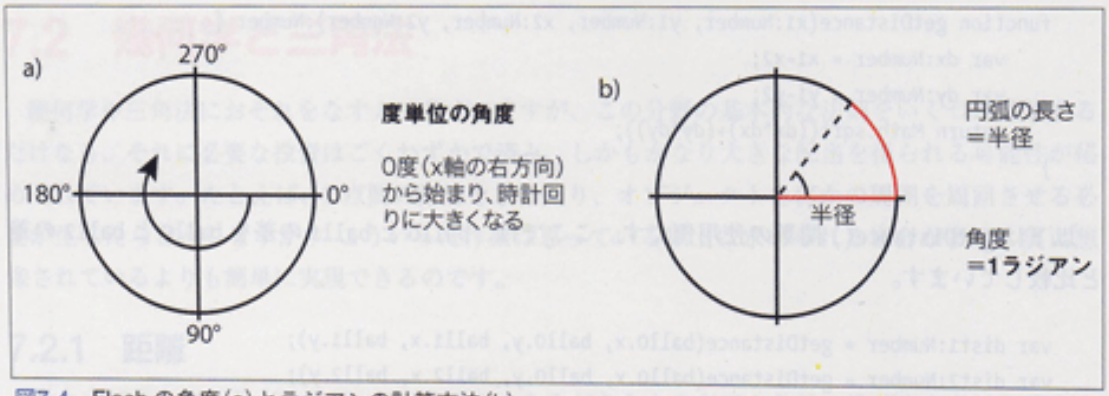

図7-4　Flash の角度(a)とラジアンの計算方法(b)

角度を度からラジアンに変更する場合の計算は以下の通り(公式として覚えましょう)

A度からラジアンに変換

A度＊(Math.PI / 180)

ラジアンから度に変換

ラジアン*(180 / Math.PI)

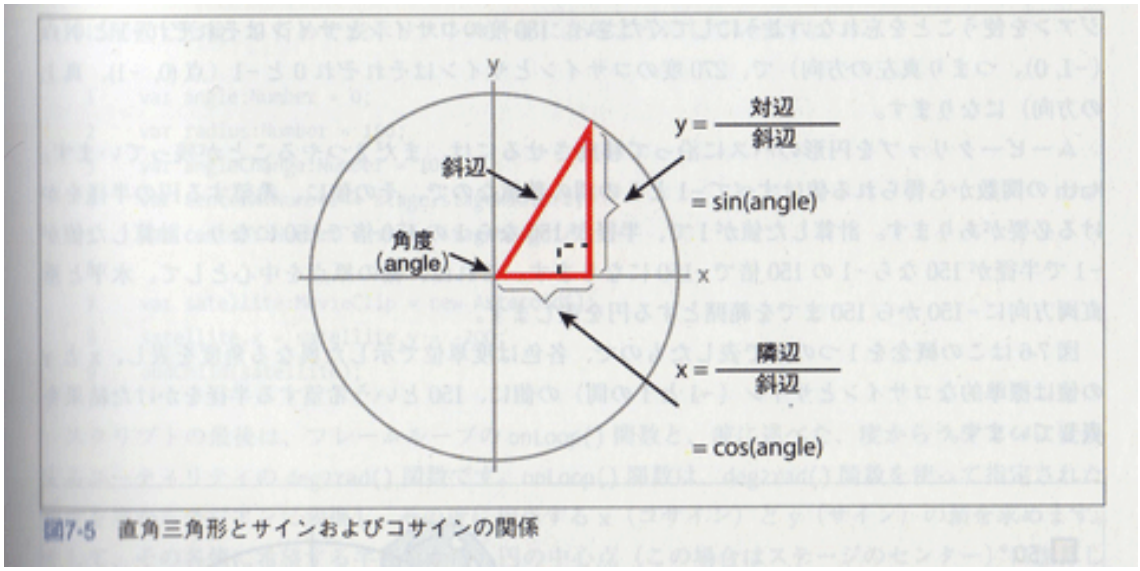

図7-5　直角三角形とサインおよびコサインの関係

y=対辺／斜辺＝sin(angle)

x=隣辺／斜辺＝cos(angle)

つまり角度（ラジアン）が決まればｘ座標とｙ座標が判ります。

cos(0)=1・・・・x=1

sin(0)=0・・・・y=0

角度０（計算はラジアンを使用）の時はｘ=1,y=0 座標つまり真右の方向です。

同様に考えると９０度（計算はラジアンを使用）のときは sin は１で cos は０です。従ってｘ=0,y=1 となり真下の方向です。１８０度（計算はラジアンを使用）のときは sin は０で cos は−１です。従ってｘ=−1,y=0 となり真左の方向、２７０度（計算はラジアンを使用）のときは sin は０で cos は−１です。従ってｘ=0,y=　−1 となり真上の方向になります。

あとは半径を積算すれば円運動させることができます！！

```
var angle:Number = 0;

var radius:Number = 150;

var radian:Nunber = degrad(angle);

var centerX:Number = stage.stageWidth/2;

var centerY:Number = stage.stageHeight/2;

ball.x=centerX+radius*Math.cos(radian);

ball.y=centerY+radius*Math.sin(radian);

angle += 10;

angle %= 360;



function degrad(deg:Number):Number{

        return deg*(Math.PI/180);

}
```

重力

重力のシミュレーションはｙ方向の加速度に少し手を加えるだけです。

ボールを空中に投げ上げた効果は以下のようにします。

ｙ速度のみに１を加えて（x 速度は変えません）ｙの初期値を負の値にするだけです。

```
addEventListener(Event.ENTER_FRAME,onloop);

function onloop (evt:Event):void{

  var b =-10

  ball.x += 10;

  ball.y += b;

  b +=1;

}
```

ゼノンのパラドックスを利用した摩擦の表現

分割のパラドックス

矢が的に向かって飛んでいます。矢は今いる位置と的のちょうど半分の位置を通らねばなりません。

その後その位置と的のちょうど半分の位置を通らねばなりません。

またその後その位置と的のちょうど半分の位置を通らねばなりません。

こうしていくと、矢は無限の課題をクリアーしなくてはならなくなります。

具体的には以下のようになります。

ボールの元の位置 orig から行き先の位置 dest の差を分割回数で割る式になります。

(dest-orig)/coeff

```
addEventListener(Event.ENTER_FRAME,onLoop);

function onLoop(evt:Event):void{

ball.x += vel(ball.x,mouseX,8);

ball.y += vel(ball.y,mouseY,8);

}

function vel(orig:Number,dest:Number,coeff:Number):Number{

return (dest-orig)/coeff;

}
```

フックの法則でバネの動きを表現

弾性はフックの法則を使って簡単に計算できます。

フックの法則　　F=−kx

F は結果として生じる力、−k はバネの強さを表す定数、 x は変形するバネの長さです。

```
var xVel:Number = 0;

var yVel:Number = 0;

addEventListener(Event.ENTER_FRAME, onLoop);

function onLoop(evt:Event):void {

xVel = velElastic(ball_mc.x, mouseX, .14, .85, xVel);

yVel = velElastic(ball_mc.y, mouseY, .14, .85, yVel);

ball_mc.x += xVel;

ball_mc.y += yVel;

}



function velElastic(orig:Number, dest:Number, springConst:Number,
damp:Number, elas:Number):Number {

elas += -springConst * (orig - dest);

return elas *= damp;

}
```

ピタゴラスの定理

２点の距離を算出する方法

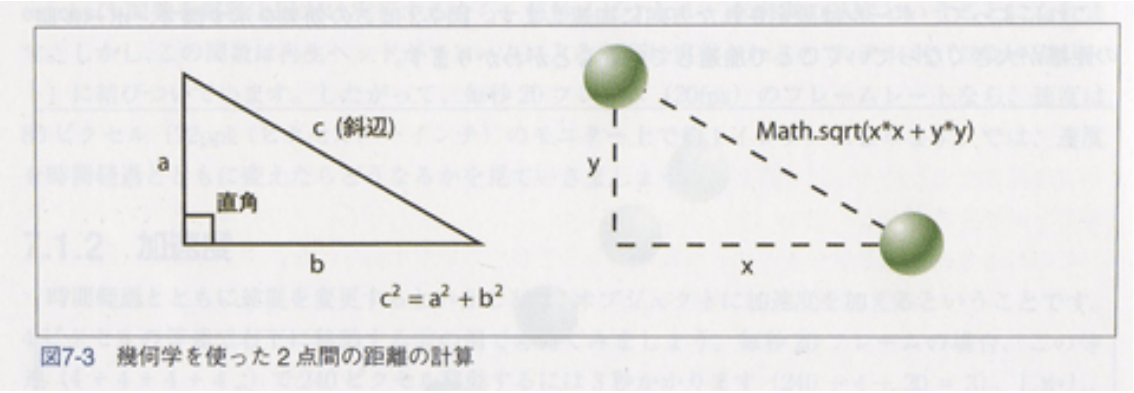

図7-3　幾何学を使った２点間の距離の計算

２点の距離を算出するにはピタゴラスの定理を使用します。

ピタゴラスの定理は上図のようなものです。

$$a^2 + b^2 = c^2$$

AS3 で記述する際には以下のようなメソッドにするとよいでしょう。

```
function getDistansce(x1:Number, y1:Number,x2:Number,y2:Number):Number{

var dx:Number = x1-x2;

var dy:Number = y1-y2;

return Math.sqrt((dx*dx)+(dy*dy));

}
```

２点間の距離が判れば衝突判定ができます。シューティングゲームなどの衝突判定はこれを利用します。簡単な歯衝突判定を行って使い方を覚えましょう。