FLASH ActionScript3.0

１．目　　的　FLASH の ActionScript3.0 の操作方法を習得し、基礎的なプログラミングを行う。

２．予備知識　ActionScript(アクションスクリプト)とは､アドビシステムズ社の製品である Flash に使用されるプログラミング言語である。これを用いることにより、動画や音声のプレイヤーの作成など、コンテンツに複雑な処理や双方向性を持たせた Flash を作成することが可能である。Macromedia 社は 1996 年にリリースしたバージョン 1 から 2005 年のバージョン 8 まで、Flash の開発をしていましたが、Macromedia 社が 2005 年に Adobe 社に買収されてから､2007 年まで新しいバージョンの Flash はリリースされませんでした。ようやく 2007 年 4 月に、Adobe 社より Flash 9 に当たる、Adobe Flash CS3 Professional が発売され、ActionScript も 3.0 にバージョンアップされました。最新の開発環境では iphone などスマートフォンのプラットフォームにも対応し、開発の利便性からも注目されています。

３．実　　習　資料と説明を参考に以下のサンプルプログラムを作成し web ページから閲覧できるようにせよ。（２年次の FLASH 実習, JavaScript 実習のページを参考にする）

最後に「反省・感想」ページを作成する（４００文字以上）

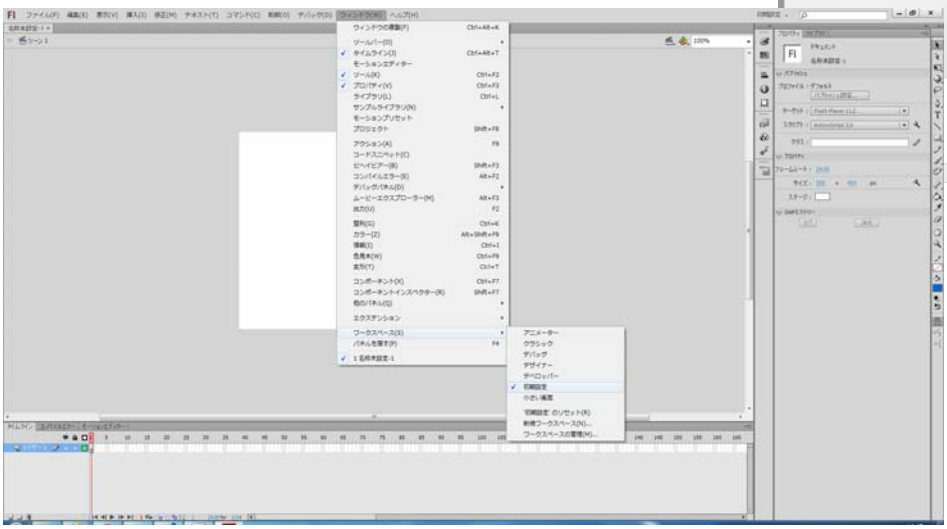
例題に使用する画像データ等は「jyouhou サーバー¥J23¥ActionScript 一斉」フォルダ内にありますが、自由な素材を使用してかまいません。

課題１　ActionScript3.0 の操作方法（ムービークリップの移動）

Adobe Master Collection から

Adobe Flash Professional CS6　を起動します。

「新規作成」から「ActionScript3.0」を選択



各種パネルの配置は

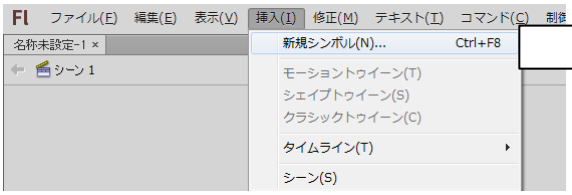
「ウィンドウ」

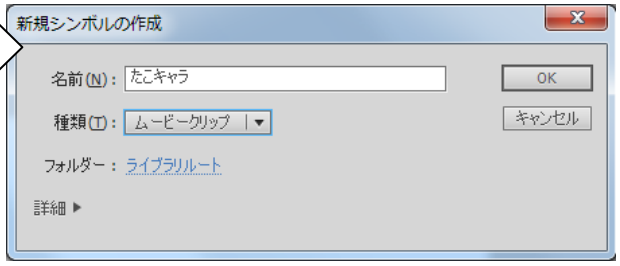
↓

「ワークスペース」

から選択できます。

① ムービークリップの作成





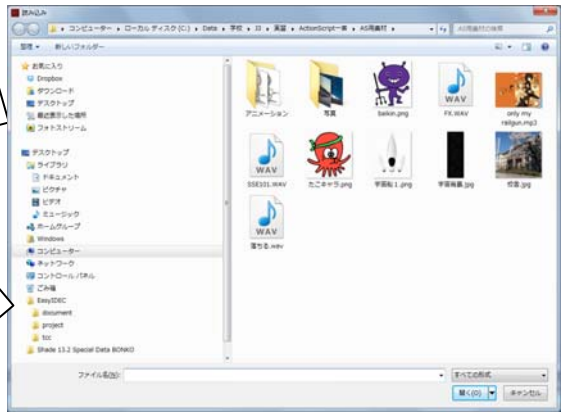
「挿入」→「新規シンボル」

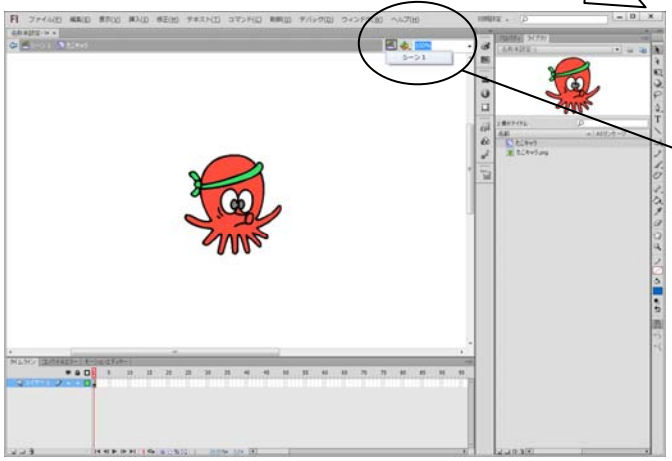
シンボルの名前を「たこキャラ」とする。

シンボル「たこキャラ」の編集画面となります。

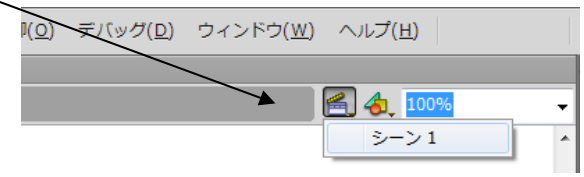
「ファイル」→「読み込み」

→「ステージに読み込み」



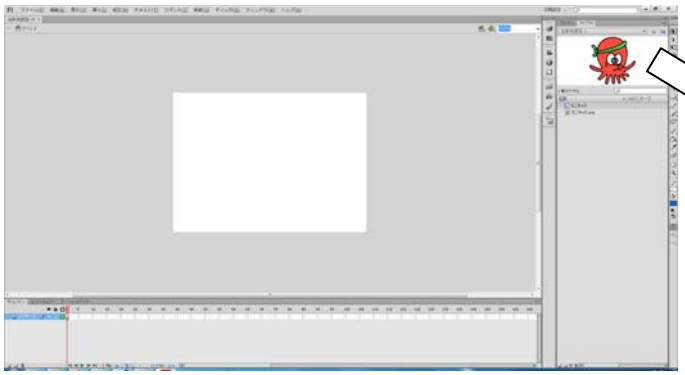


読み込んだ画像の中心を、ステージの原点付近に合わせておきます。



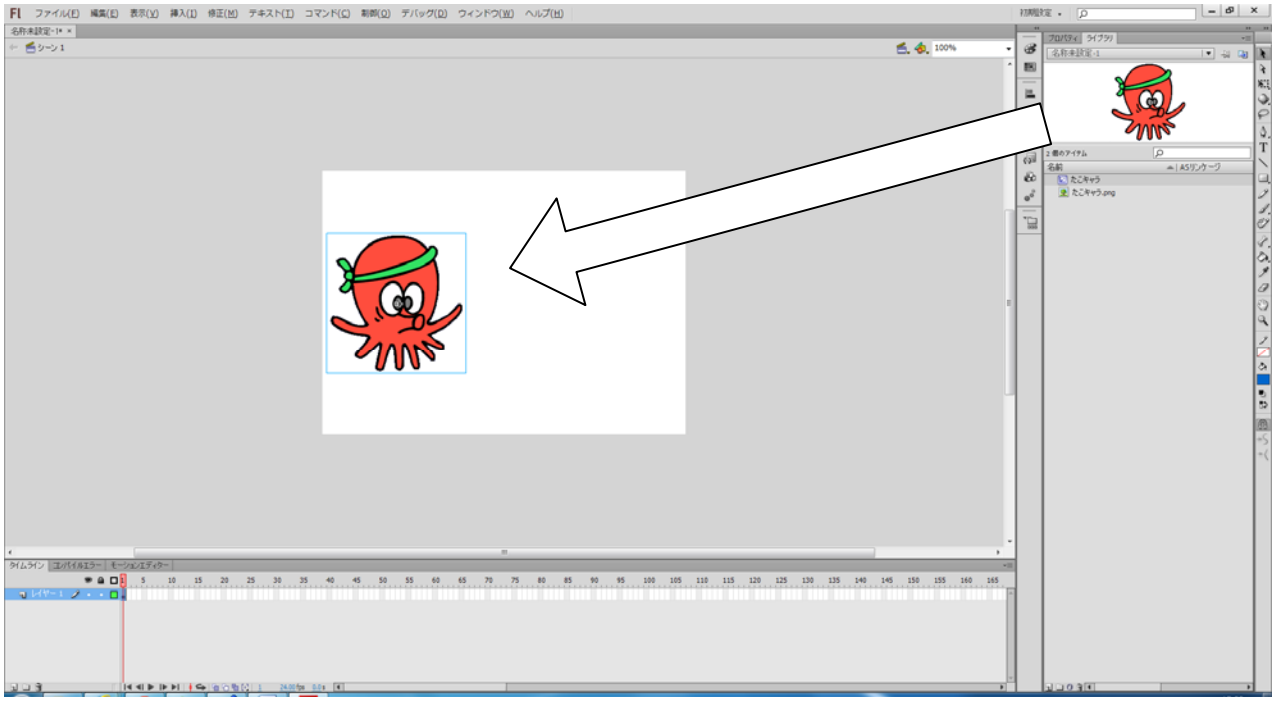
「シーン１」に戻り

ライブラリから「たこキャラ」を確認しておきます。

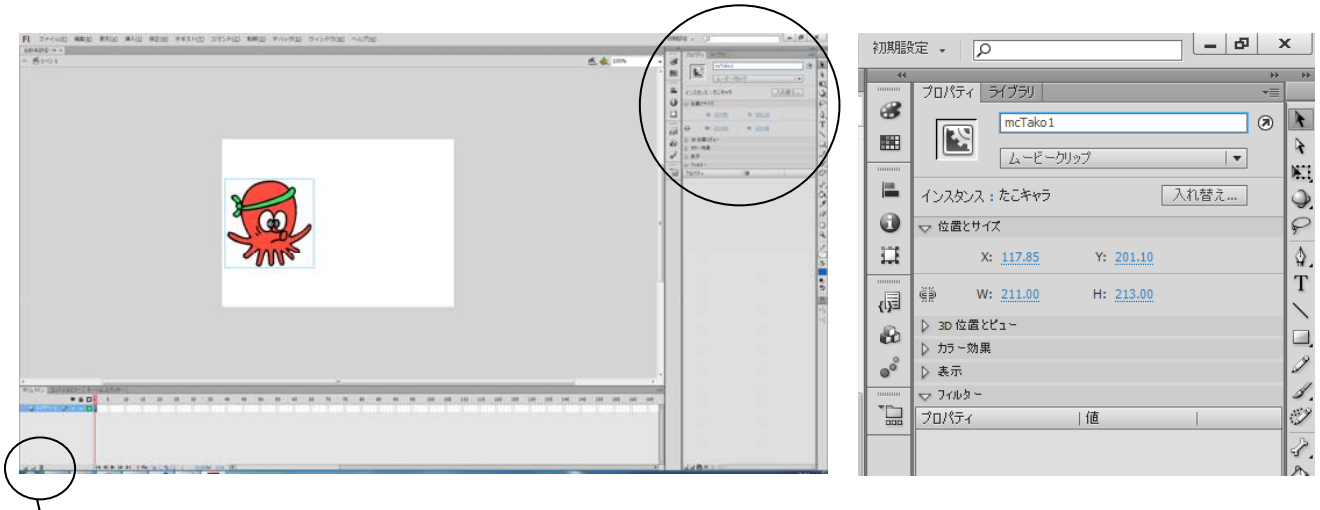


② インスタンスの配置

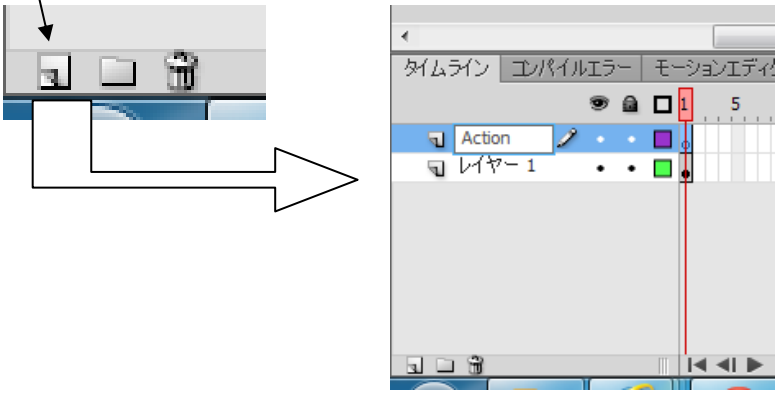
ライブラリから「たこキャラ」をドラッグして、シーン１のステージに配置します。



「プロパティ」タブを選択し「インスタンス名」を「mcTako1」と設定します。（英大小文字区別）



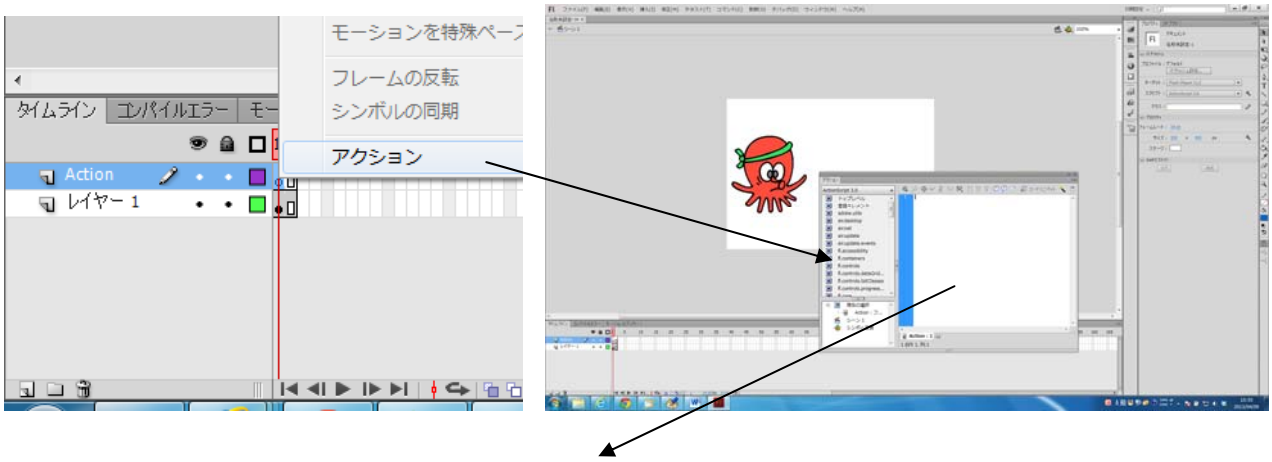
ActionScript3.0 からは全てのソースコードをタイムラインに記述することになりました。

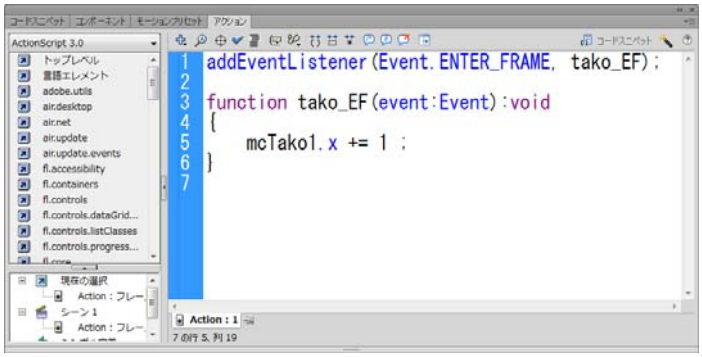


ActionScript を記述するためのレイヤーを新たに作成し名前を「Action」とします。

③ ActionScript3.0 ソースコードの入力

「Action」レイヤーの１フレーム目で右クリック、「アクション」を選択しアクションパネルを表示





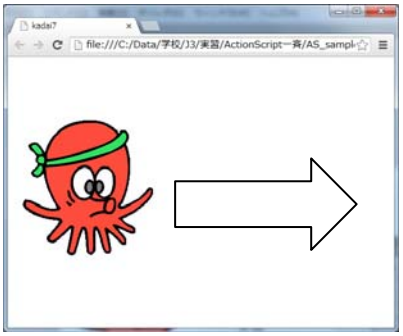
ソースコードを入力します。

英大文字と小文字は完全に区別されます

自身のネットフォルダへ保存し，パブリッシュ（ＨＴＭＬ）します。

○増減値を変化させて実行してみよう。

○ｗｅｂページから見れるように設定する



◆イベントの処理

イベントは、「マウスをクリックする」、「キーボードが押された」といったユーザからの入力や Flash 実行時に生じる処理の一部のことを指します。イベントの発生によって何かが起こるようなプログラムをイベントドリブン型プログラムといいます。イベントドリブン型プログラムでは、イベントを監視して、イベントが発生した時に動作するプログラムを記述します。この、イベントの監視から動作までを記述した部分をイベントハンドラと呼びます。つまりイベント処理とは**「オブジェクト」**に**「イベント」**が発生した際に**「イベントハンドラ」**を実行させることだと言えます。

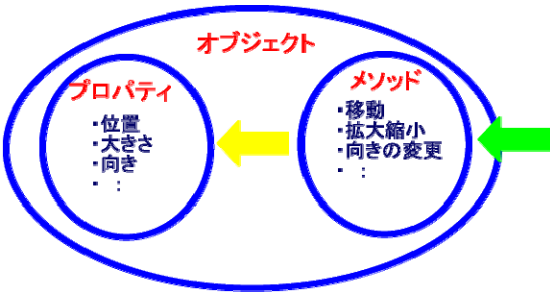
```
addEventListener(Event.ENTER_FRAME, tako_EF);
function tako_EF(event:Event):void
{
    mcTako1.x += 1 ;
}
```

イベントハンドラ（関数）

Event.ENTER_FRAME イベントは、タイムラインが次のフレームに進む速度で自動的に発生するイベント（２４ｆｐｓであれば　1/24 秒ごとに発生する）

◆オブジェクトのプロパティとメソッド

Action Script の特徴としては、オブジェクト指向とイベントドリブン型であることが挙げられます。Action Script では Flash のさまざまな要素をオブジェクトとして扱います。ムービークリップもムービークリップオブジェクトという一つのオブジェクトです。オブジェクトは、以下の２つのものを持ちます。



○プロパティ

オブジェクトの持つ位置や角度など性質を表わします。直接アクセスすることも可能ですが、オブジェクト指向の理念ではメソッドを通してアクセスを行います。

```
mcTako.x += 1 ;
```

↑ インスタンス名　↑ プロパティ

○メソッド

オブジェクトに定義された命令（関数）です。外部からこれにアクセスしてオブジェクトの制御を行います。

mcインスタンスのおもなプロパティ	
種　　類	プロパティ
Ｘ座標	x
Ｙ座標	y
高さ	height
幅	width
横方向の拡大率（%）	scaleX
縦方向の拡大率（%）	scaleY
回転角度	rotation
透明度（0.0～1.0）	alpha
表示／非表示	visible

課題２　ムービークリップを斜めに移動

課題３　ムービークリップを拡大縮小

課題４　ムービークリップを回転

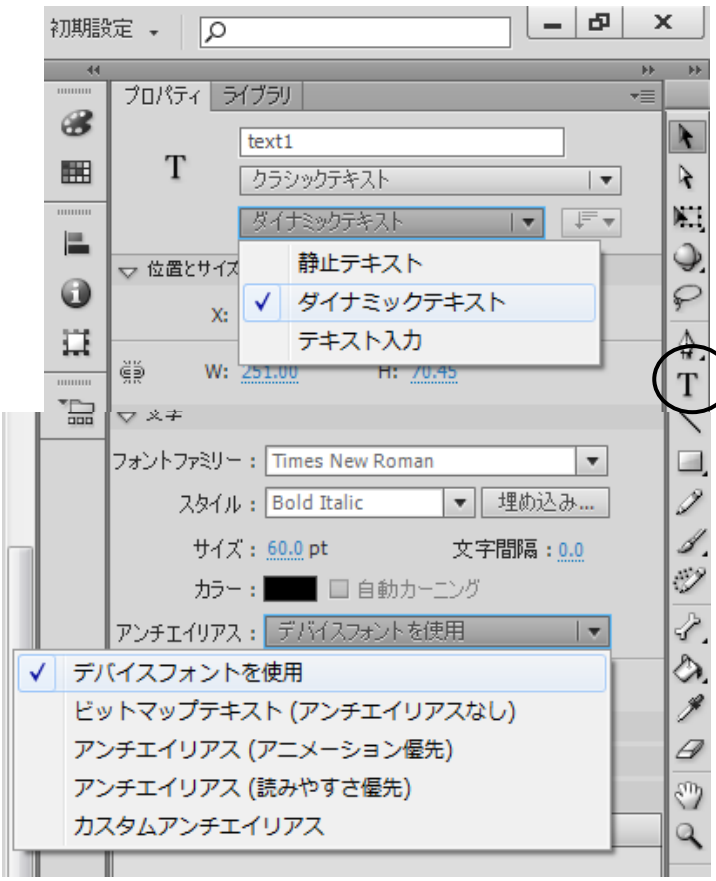
課題５　ムービークリップの透明度を変化

課題６　ムービークリップの位置をランダムで移動

Math.random()　関数で 0.0～1.0　の乱数を発生させます。

```
mcTako1.x = Math.random() * [　　] ;
mcTako1.y = Math.random() * [　　] ;
```

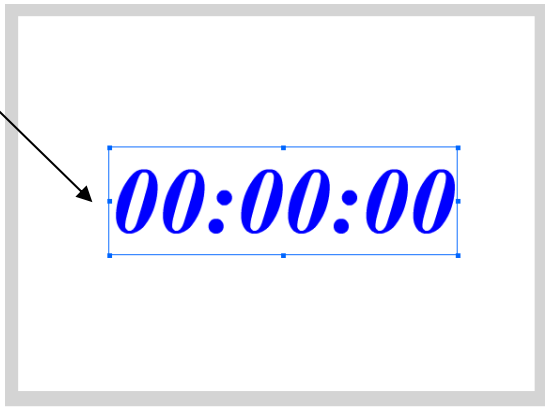
課題７　デジタル時計を作る

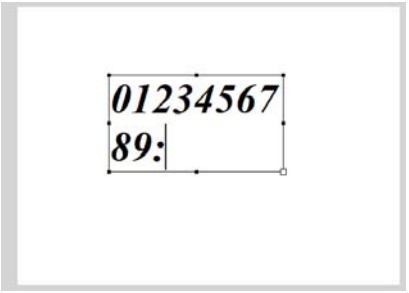


① テキストを配置し、インスタンス名を「text1」
　種類を「ダイナミックテキスト」に設定

②アンチエイリアス設定で「デバイスイフォントを使用」　をチェックする

③「００：００：００」と文字を入力し、大きさ、位置などを整える。

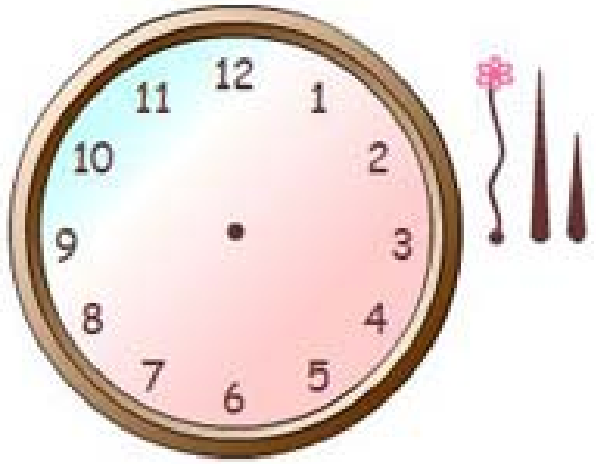




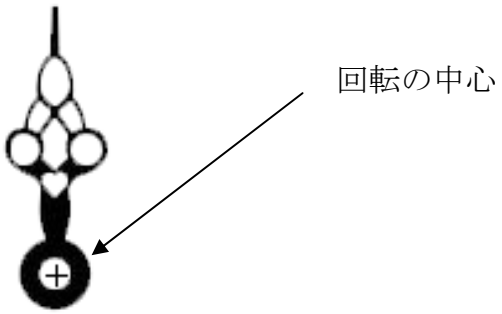
好みのフォントを使用したい場合は、右のように使用する文字をすべて埋め込み、好みのフォントスタイルを適用する。

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

function tokei_EF(event:Event):void
{
        var date_obj = new Date();                  // ローカル時間を取得

        var hour:int = date_obj.getHours();         // 時間を取得
        var min:int   = date_obj.getMinutes();      // 分　を取得
        var sec:int   = date_obj.getSeconds();      // 秒　を取得

        text1.text = hour + ":" + min   + ":" + sec;      //表示
}
```

課題８　アナログ時計を作る



①時計の　文字盤　、　長針　、　短針　、秒針のデータを用意する（描く）
　このとき、画像の「回転中心」に注意する



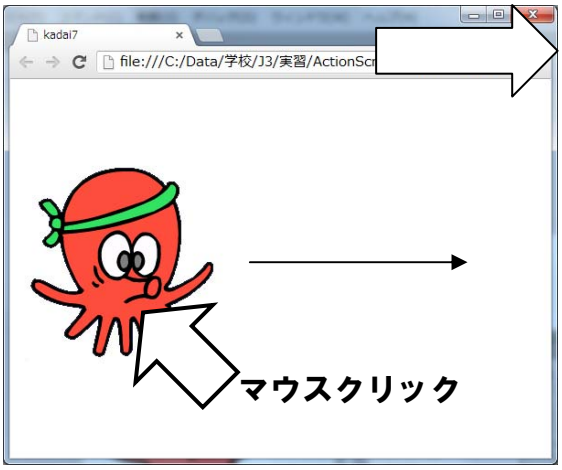
① それぞれをムービークリップとして作成し、インスタンス名を設定する。
　長針　cyousin
　短針　tansin
　秒針　byousin

② 時間、分、秒の値から回転角度を計算し設定する。

```
this.addEventListener(Event.ENTER_FRAME,tokei_EF);

function tokei_EF(event:Event){
        var date_obj = new Date();

        var hour = date_obj.getHours();
        var min = date_obj.getMinutes();
        var sec = date_obj.getSeconds();

        text1.text = hour +":" + min + ":" + sec ;

        tansin.rotation =
        cyousin.rotation =
        byousin.rotation = sec *(360/60) ;
}
```

課題９　mc がクリックされたら移動する（マウスイベント）

イベントは、「マウスをクリックする」、「キーボードが押された」といったユーザからの入力やFlash 実行時に生じる処理の一部のことを指します。イベントの発生によって何かが起こるようなプログラムをイベントドリブン型プログラムといいます。イベントドリブン型プログラムでは、イベントを監視して、イベントが発生した時に動作するプログラムを記述します。この、イベントの監視から動作までを記述した部分をイベントハンドラと呼びます。つまりイベント処理とは**「オブジェクト」**に**「イベント」**が発生した際に**「イベントハンドラ」**を実行させることだと言えます。



タイムラインの１フレーム目に記述するコード　（課題⑥のコードは削除する）

```
mcTako1.addEventListener(MouseEvent.CLICK,  tako_click);
```

オブジェクトを　・・・・・・　クリックしたら　・・・・　この関数を実行
（イベント）　（イベントハンドラ）

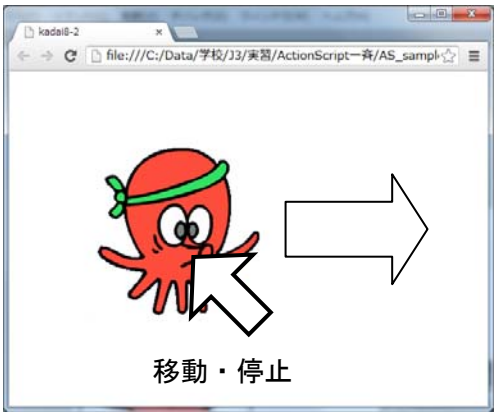
```
function tako_click(event:MouseEvent):void
{
      mcTako1.x += 10 ;
}
```

おもなイベント

マウスボタンのクリック	MouseEvent.CLICK
〃　　のダウン	MouseEvent.MOUSE_DOWN
〃　　のアップ	MouseEvent.MOUSE_UP
領域内でマウスが動いた	MouseEvent.MOUSE_MOVE
キーボードが押された	KeyboardEvent.KEY_DOWN
次のフレームへ進んだ	Event.ENTER_FRAME

◆ムービークリップのクリックで移動・停止を繰り返す

移動中のムービークリップのインスタンスをクリックすると停止、再びクリックすると移動の動作を繰り返す。
移動量を制御する変数を設定し対応します。



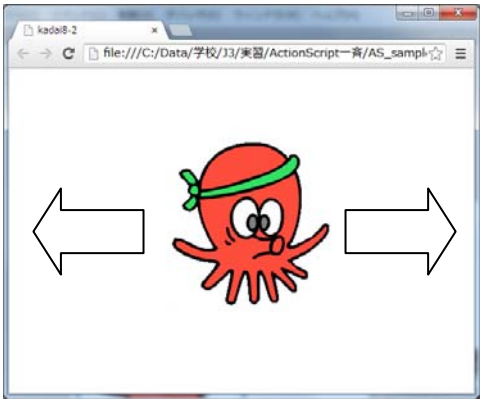
Event.ENTER_FRAME イベントは、タイムラインが次のフレームに進む速度で自動的に発生するイベント
（２４ｆｐｓであれば　1/24 秒ごとに発生する）

ActionScript をメインタイムラインの１フレームに記述

```
var mv:int= 2 ;          // 移動量の変数宣言

addEventListener(Event.ENTER_FRAME, tako_EF);

function tako_EF(event:Event):void
{
    mcTako1.x += mv ;
}

mcTako1.addEventListener(MouseEvent.CLICK, tako_click);

function tako_click(event:MouseEvent):void
{
      if( mv == 2 )              // 停止・移動の切替（移動量）
            mv = 0 ;
      else
            mv = 2 ;
}
```

課題１０　mc のクリックで移動方向を反転

左から右へ移動中のムービークリップをクリックしたら、移動方向を右から左へ反転させる。これを繰り返すようにプログラミングせよ。



課題１１　ムービーの階層化（ｍｃ自身に動きを入れる）

課題８をベースに作成していきます。

① ステージに配置されているインスタンス「mcTako1」を消去します。

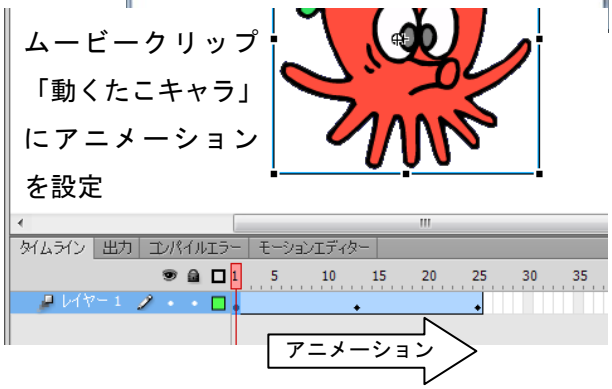
② メニューの「挿入」→「新規シンボル」と選択し、名前を「動くたこキャラ」と設定します。

③ 「動くたこキャラ」の編集画面でライブラリからｍｃ「たこキャラ」をドラッグして配置します。

④ 配置したインスタンスに「mcTako1」と名前を付けます。

⑤ インスタンスにモーショントゥイーンを設定し、その場で回転、拡大縮小するようなアニメーションをつけます。

⑥ シーン１に戻ってパブリッシュで確認します。

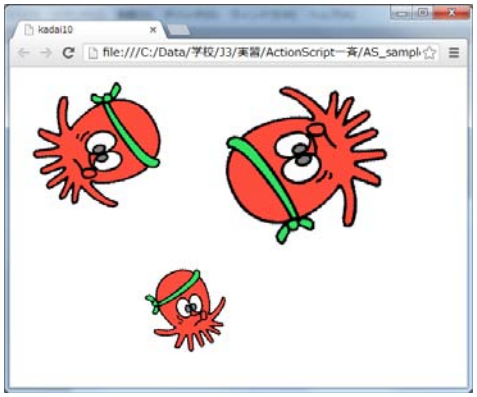


課題１２　ムービーの階層化（ActionScript を内包したインスタンス）

課題８をベースに作成していきます。

① シーン１に記述してあるスクリプトを次のように変更（太字部分）し、動作を確認します。

```
var rt:int= 5 ;
addEventListener(Event.ENTER_FRAME, tako_EF);
function tako_EF(event:Event):void
{
    mcTako1.rotation += rt ;
}
mcTako1.addEventListener(MouseEvent.CLICK, tako_click);
function tako_click(event:MouseEvent):void
{
        rt *= -1 ;      //クリックで回転を反転
}
```



② ステージに配置されているインスタンス「mcTako1」消去します。

③ メニューの「挿入」→「新規シンボル」と選択し、名前を「ＡＳ付たこキャラ」とします。

④ 「ＡＳ付たこキャラ」の編集画面でライブラリからｍｃ「たこキャラ」をドラッグして配置します。

⑤ 配置したインスタンスに「mcTako1」と名前を付けます。

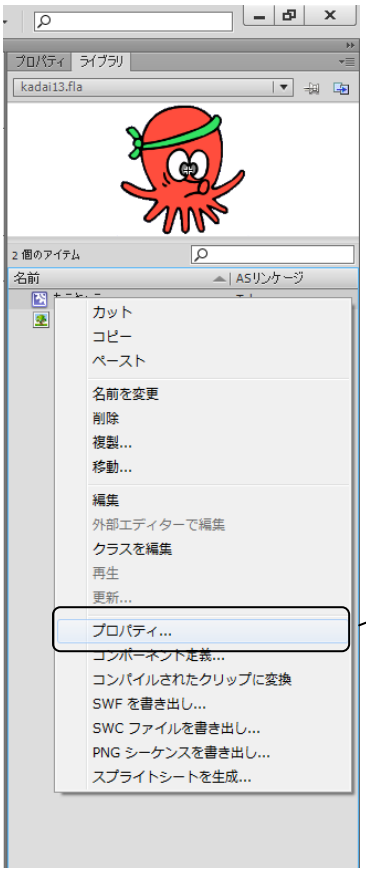
⑥ シーン１のタイムライン「フレーム１」からスクリプトをカットし、「ＡＳ付たこキャラ」のタイムライン「フレーム１」へペーストします。

⑦ シーン１のステージへ「ＡＳ付たこキャラ」をいくつか配置します。（大きさ変更可）

⑧ パブリッシュして動作を確認します。（インスタンスごとにスクリプトが動作）

◆ムービークリップのインスタンスを生成（コピーして増やす）

ライブラリに登録したムービークリップシンボルのインスタンスをスクリプトで追加するには、シンボルに「埋め込みアセットクラス」の設定を行う必要があります。

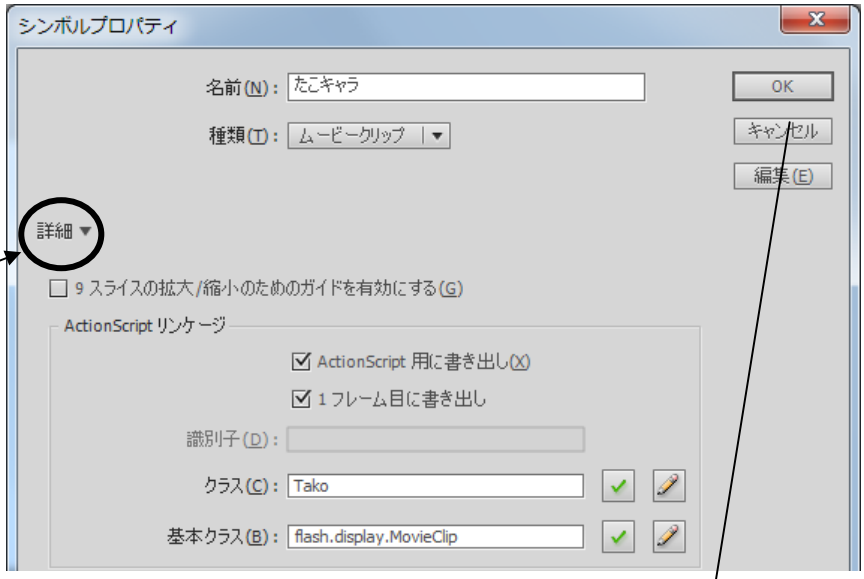


① 課題１と同様に操作してムービークリップ「たこキャラ」を作ります。

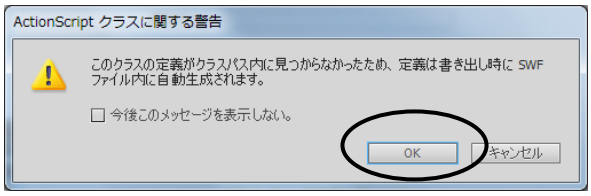
② ライブラリから「たこキャラ」を右クリック「プロパティ」を選択

③ 「詳細」をクリックして詳細画面を開きます。

④ 「ActionScript 用に書き出し」にチェックを入れ
「クラス」に半角英数で「**Tako**」と入力し「ＯＫ」とします。



警告が表示されますが
「ＯＫ」をクリックします。

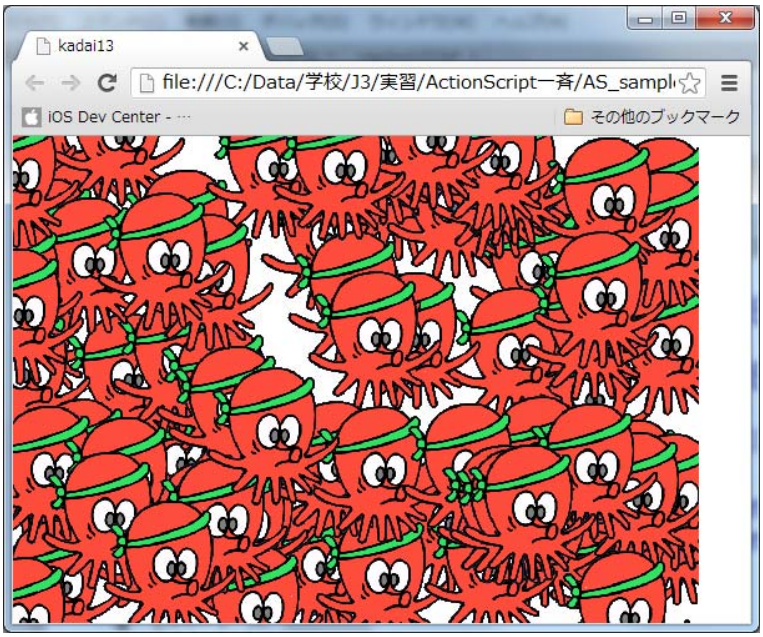


タイムラインにスクリプトを記述します。

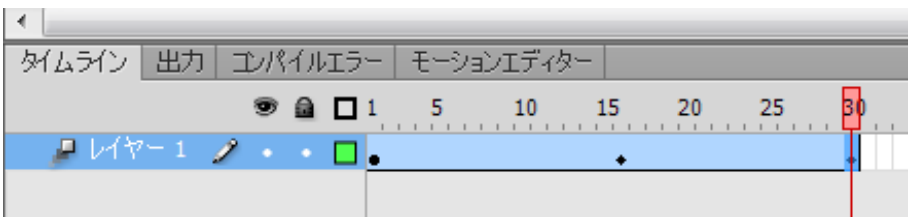
```
addEventListener(Event.ENTER_FRAME, tako_EF);

function tako_EF(event:Event):void
{
      var new_Tako:MovieClip = new Tako();        //インスタンス生成(コピー)
        addChild(new_Tako);                       //  ステージに表示
        new_Tako.x =Math.random()*550;            //  位置をランダムに
        new_Tako.y =Math.random()*400;
}
```

インスタンスが１秒ごとに増えるようにスクリプトを追加します。

```
var old_time = getTimer();              //現在時刻をミリ秒単位で取得

addEventListener(Event.ENTER_FRAME, tako_EF);

function tako_EF(event:Event):void
{
        if( getTimer() - old_time > 1000 ){                    // 差が1秒になったら…
                var new_Tako:MovieClip = new Tako();
                addChild(new_Tako);
                new_Tako.x =Math.random()*550;
                new_Tako.y =Math.random()*400;
                old_time = getTimer();
        }
```



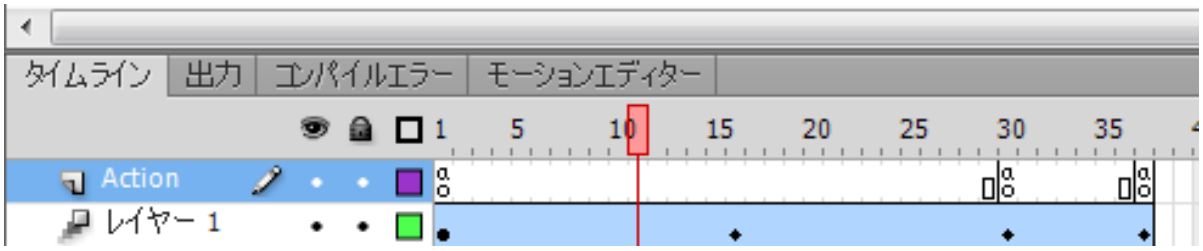
課題１３　**「課題１１」のように「たこキャラ」ムービークリップ自身にモーショントゥーイーンを使用して動きをつけ、これを１秒ごとに増殖させるムービーを作成しなさい。**



ムービクリップ「たこキャラ」にモーショントゥーイーンを設定する。

課題１４　簡単なゲームを作成（クリックしたらインスタンスを破棄）

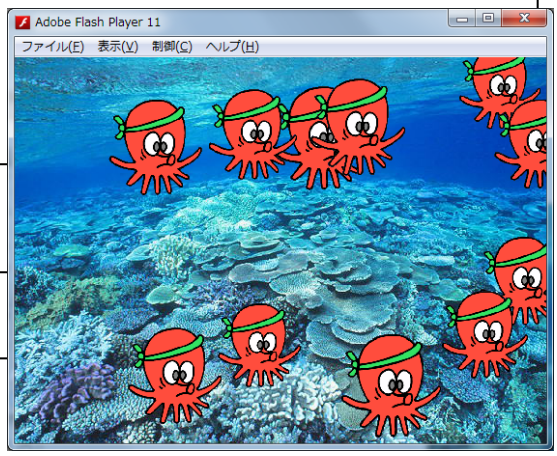
ムービークリップ「たこキャラ」のタイムラインとスクリプト



通常のアニメーション　　**消える時のアニメーション**

１フレーム目

```
this.addEventListener(MouseEvent.CLICK, fl_ClickToPosition);
function fl_ClickToPosition(event:MouseEvent):void
{
        gotoAndPlay(31);
}
```



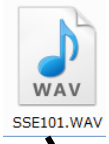
３０フレーム目

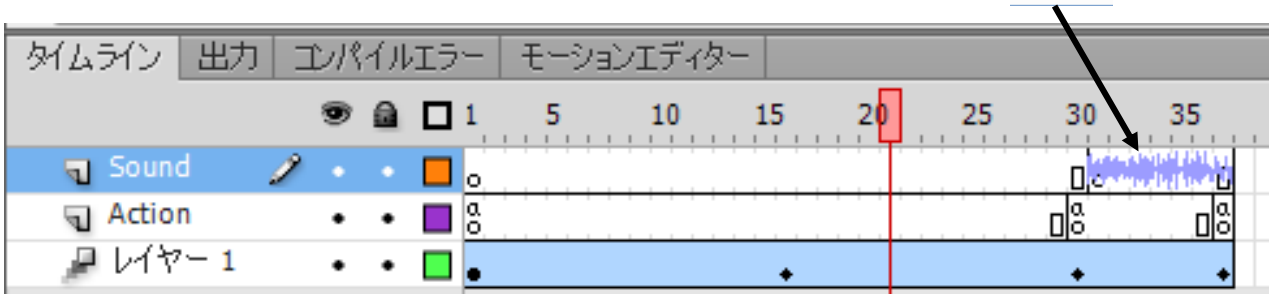
```
gotoAndPlay( 1 );
```

３７フレーム目（最終フレーム）

```
Object(parent).removeChild(this);     //自分自身の消去
```

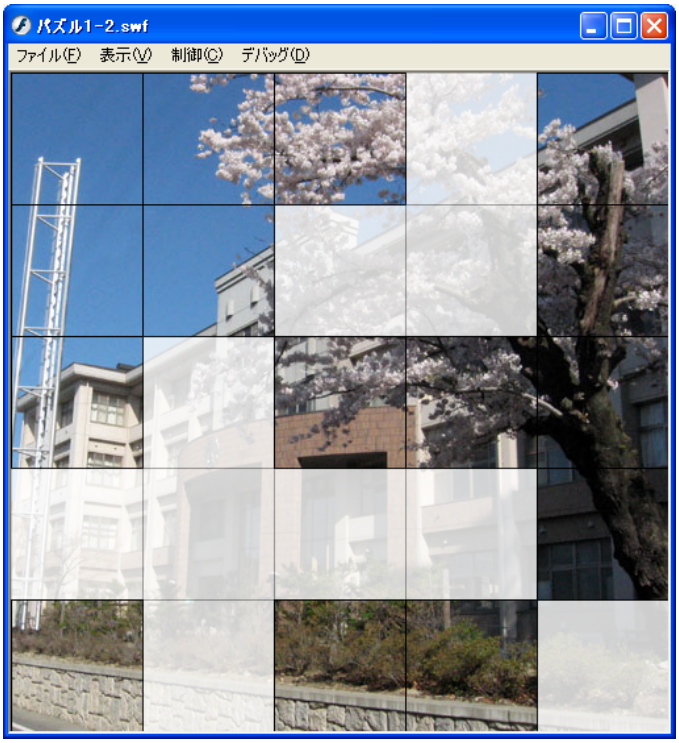
「たこキャラ」のタイムラインにサウンド用のレイヤーを追加し
クリップ消滅時などの効果音を入れて、よりゲームらしくしてみよう





「シーン１」に背景レイヤーを追加し、海中の写真なども設定してみよう。

課題１５　ライツアウト（パズルゲーム）

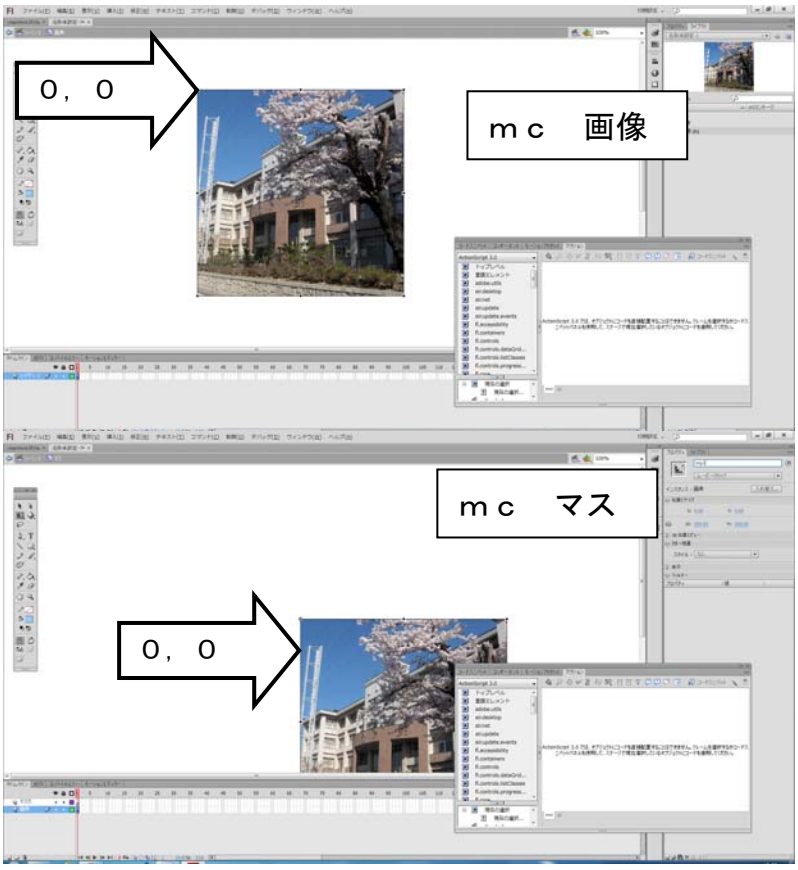


ライツアウトは５×５に区切られたマスをクリックするごとにそのマスの上下左右マスの表示・非表示を反転させ、最終的に元の画像を表示させるパズルです。今回はマスの表示・非表示ではなくマス（ムービークリップ）の透明度を変化させます。

基本設定
・ステージサイズ　５００×５００ピクセル
・１マスの大きさ　１００×１００ピクセル

元画像として５００×５００ピクセルのデータを用意しておきます。

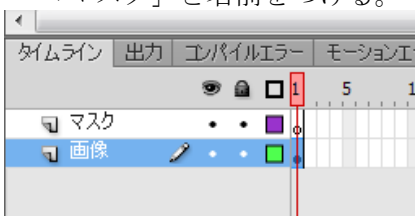
◆マスの設定、配置（マスク）

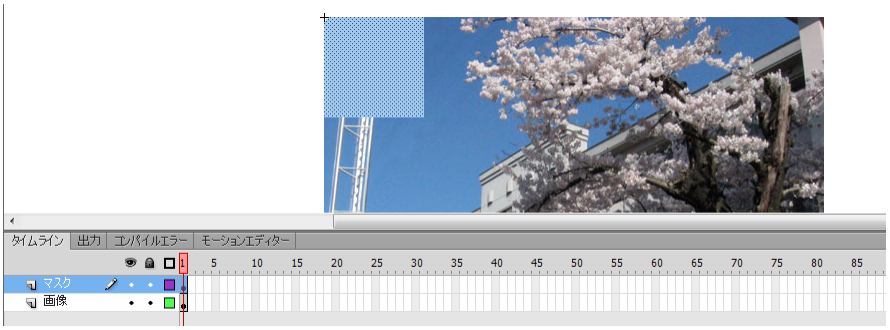


①新規シンボル（ムービクリップ）を作成し「マス」と名前を付ける。

③ 新規シンボル（ムービークリップ）を作成し「画像」と名前を付ける。

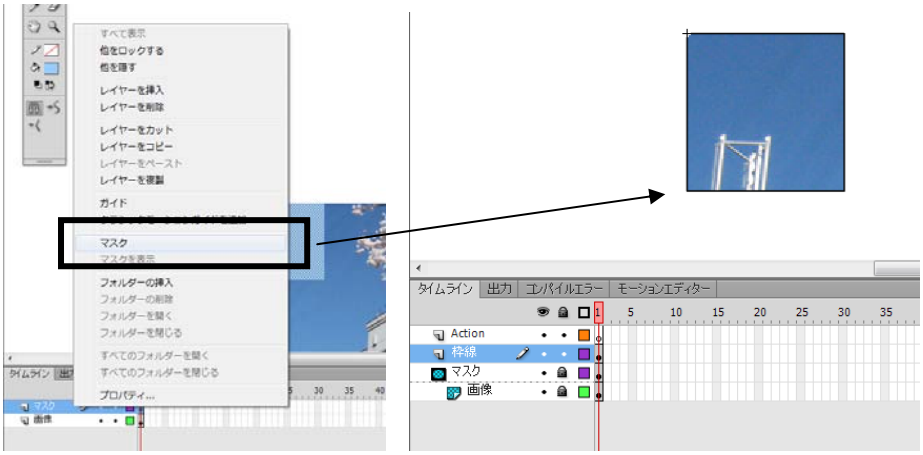
④ 「画像」にイメージデータを配置し、左上位置を（０，０）に設定する。
⑤ ｍｃ「マス」へ、ｍｃ「画像」を配置し、左上位置を（０，０）に設定する。レイヤ名を「画像」に変更。
⑥ インスタンス名を「ｉｍｇ１」に
⑦ タイムラインへレイヤを追加し「マスク」と名前をつける。





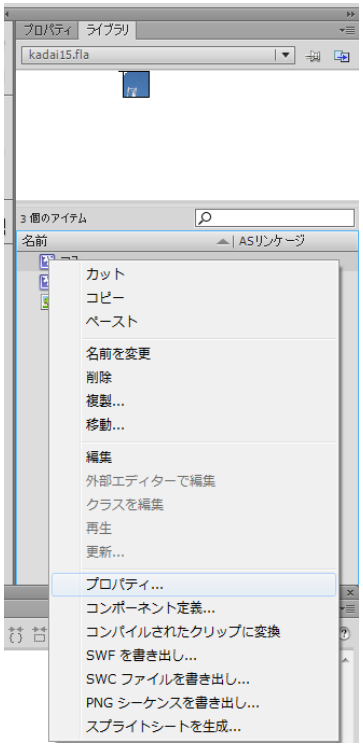
⑧ 「マスク」レイヤに画像の左上（０，０）を基準に100px × 100px の正方形を描く

⑨ 「マスク」レイヤで右クリックしメニューから「マスク」を選択



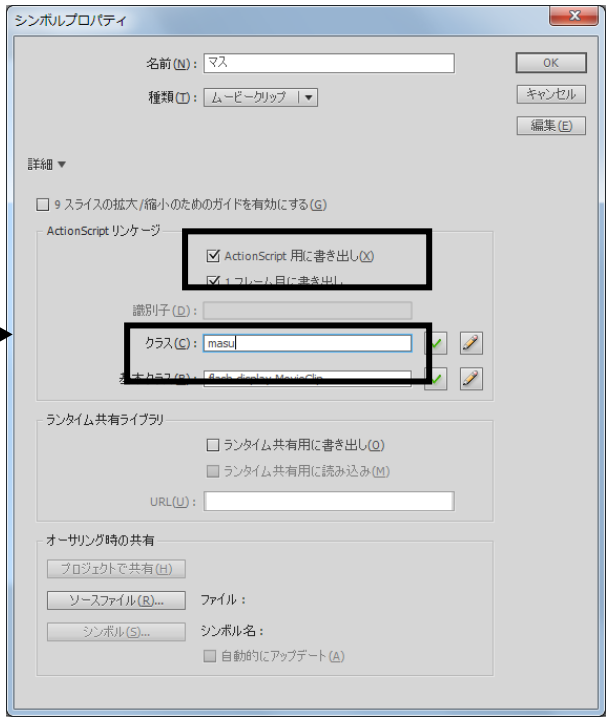
⑩ 「枠線」レイヤを追加し、マスの外枠に線を描く

⑪ スクリプト記述用のレイヤー「Action」も追加しておく。



⑫ シーン１に戻り、ライブラリパネルのｍｃ「マス」を右クリックしてプロパティを選択。インスタンスのコピーをするための設定をします。

⑬ 「詳細」をクリックして「ActionScript 用に書き出し」にチェック
クラス名に「masu」と入力し「ＯＫ」をクリックます。



⑭ 「警告」が表示されますが「ＯＫ」をクリックします。

◆マスのコピー　→　並べて配置

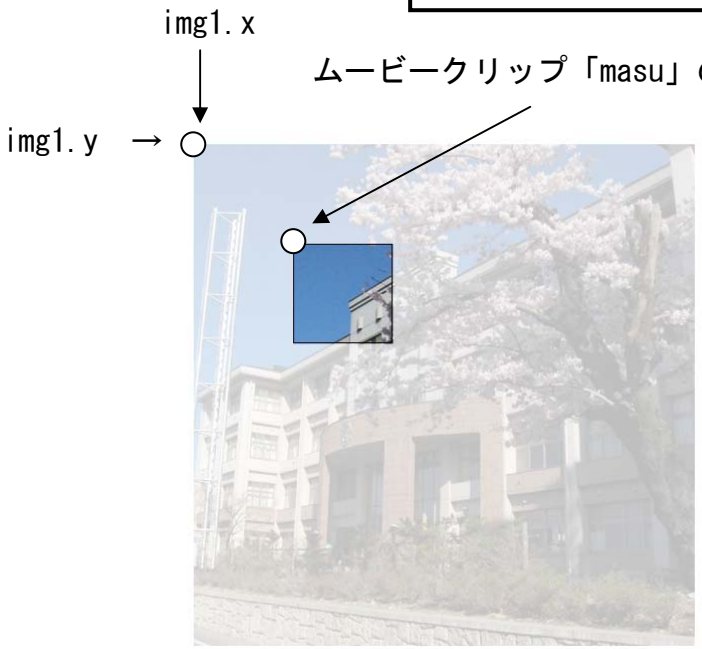
```
var kx,ky,no;

for( ky=0; ky<5; ky++ ){
    for( kx=0; kx<5; kx++ ){
      var new_masu:MovieClip = new masu();
      addChild(new_masu);
      new_masu.name = "masu" + ( ky*5 + kx );
      new_masu.x = kx*100;
      new_masu.y = ky*100;
    }
}
```

シーン１・タイムラインの１フレームへスクリプトを記述する。
ムービークリップをコピーし、名前を付けて(masu0 ～ masu24)縦横５×５に配置する。
このとき各マスには同じ画像が表示されている。

マスクされている画像「img1」をずらすことにより１枚の画像として見えるように並べる。

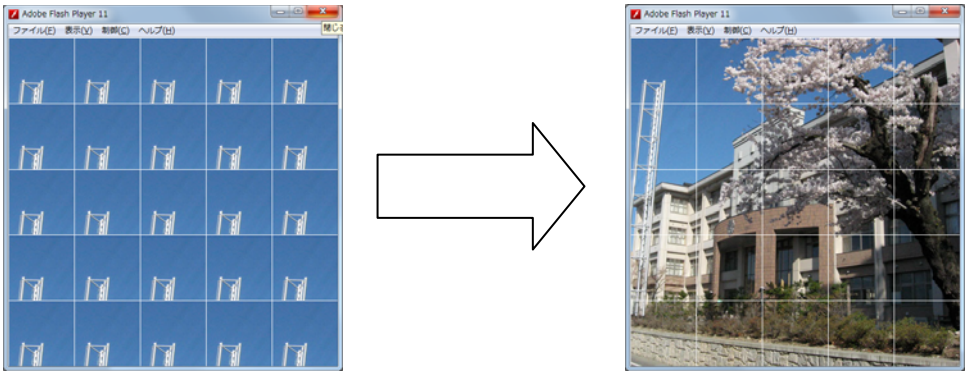
画像をずらす処理を追加

```
new_masu.img1.x =
new_masu.img1.y =
```



ムービークリップ「マス」の「マスク」の「窓」から下の画像を見ている状態。

下の画像（インスタンス名　ｉｍｇ１）の座標位置をずらすことにより、窓から見える画像が変わる。
この例では　　img1.x = -100
　　img1.y = -100



◆クリックしたマスの透明度を変える

次のスクリプトを追加します。

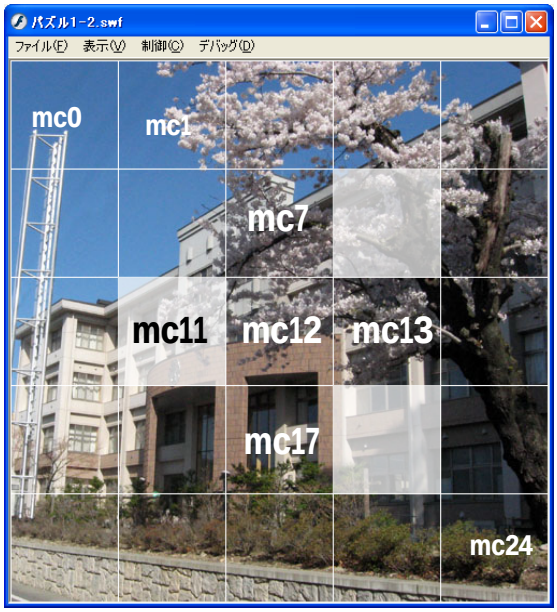
① 各マスのクリックイベントをイベントリスナーへ登録

② マウスクリック時のイベントハンドラ（関数）を記述

```
var kx,ky,no;

for(ky=0;ky<5;ky++){
    for(kx=0;kx<5;kx++){
        var new_masu:MovieClip = new masu();
        addChild(new_masu);
        new_masu.name = "masu" + (ky*5+kx);
        new_masu.x = kx*100;
        new_masu.y = ky*100;
        new_masu.img1.x = -kx*100;
        new_masu.img1.y = -ky*100;
        new_masu.addEventListener(MouseEvent.CLICK, masuClick);
    }
}
function masuClick(event:MouseEvent):void      イベントの発生したマス
{
        var mcTarget:MovieClip = MovieClip(event.currentTarget);
        mcTarget.alpha = 0.2;        //クリックされた mc の透明度を２０％に設定
}
```

クリックするたびに反転（透明度１００％ ←→２０％）させる

```
・・・・
・・・
function masuClick(event:MouseEvent):void
{
        var mcTarget:MovieClip = MovieClip(event.currentTarget);
        if(   mcTarget.alpha == 1.0 )
              mcTarget.alpha = 0.2;
        else
              [                                    ]
}
```



◆クリックしたマスの上下左右も反転させる

ステージには図のようにムービークリップのインスタンスが配置されています。たとえば mc12 がクリックされた時、その上下左右にもあるインスタンス(mc7,mc11,mc13,mc17)も反転させなければなりません。自分自身がイベントを受け取って透明度を変化させるのは簡単ですが、他のインスタンスも操作するのは工夫が必要です。

① イベント（マウスプレス）の発生したインスタンス名（番号）を取得
② 上下左右のインスタンス名（番号）を計算
③ 合計５マスの透明度を変更

の手順に従わなければなりません。

① 例）イベント発生インスタンス mcTarget
インスタンス名の取得（プロパティ） mcTarget.name
番号文字列の切出し（４文字目～終り） mcTarget.name.substring(4)
番号文字列を整数値に変換 parseInt(mcTarget.name.substring(4))

↓

後のプログラムのわかりやすさも考えて、マスを反転させる処理は別関数とし、イベントの発生したインスタンスの番号を渡して処理するように変更する。

```
function masuClick(event:MouseEvent):void
{
        var mcTarget:MovieClip = MovieClip(event.currentTarget);
        var no = parseInt(mcTarget.name.substring(4));
        masu_no( no );
}

function masu_no( no ):void
{
        rev( no );
}

function rev( no ):void                インスタンス名 を インスタンス自身に変換
{
        var mc_name = "masu"+no;
        var mcTarget:MovieClip = MovieClip(getChildByName(mc_name));
        if(     mcTarget.alpha == 1.0)
                mcTarget.alpha = 0.2;
        else
                mcTarget.alpha = 1.0;
}
```

上下左右のマスも反転させる

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

基本的に

上は　－５

下は　＋５

左は　－１

右は　＋１

すればよいが

上下左右へはみ出す場合を考える。

```
・・・
・・・・
function masu_no( no ):void
{
        rev( no );
        if( no-5 >=  0 )
        if( no+5 <= 24 ) rev( no+5 );
        if( no%5 !=  0 )
        if( no%5 !=  4 ) rev( no+1 );
}
・・・・
・・
```

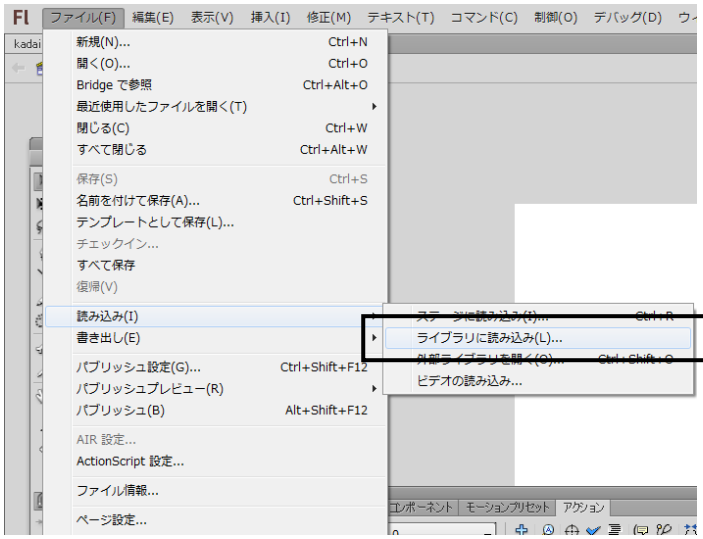
スタート時に数か所を反転させる（パズル出題）

マスを並べて表示した後に追加

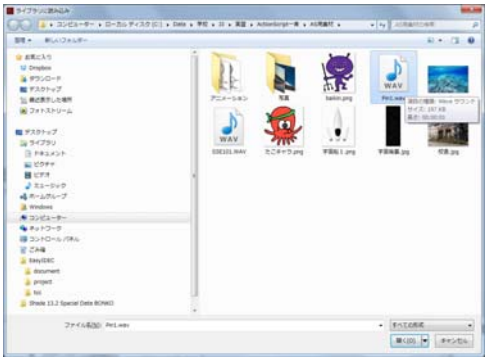
```
・・・
・・・・                    どこか３か所をクリック
for(var i=0;i<3;i++){
        no = int( Math.random()*25 );
        masu_no( no );
}
・・・・
・・
```

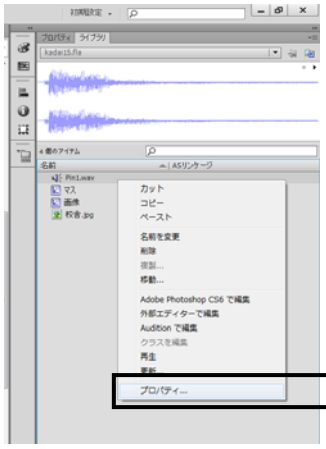
ActionScript でサウンド出力（ライブラリに登録済みのデータ）

マスをクリックしたら、ライブラリに登録してあるサウンドデータを再生する。

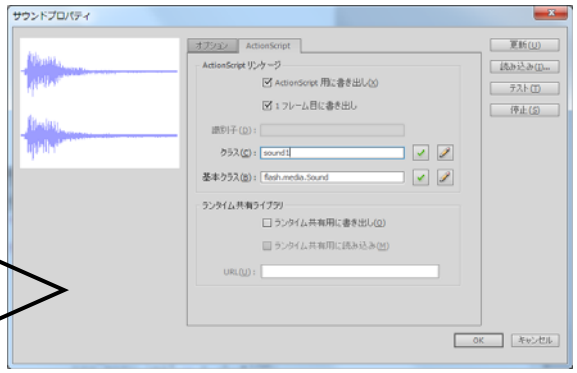


① 「ファイル」→「読み込み」
「ライブラリに読み込み」と選択し、
サウンドファイルを読み込む





② 右上のライブラリパネルで、読み込んだサウンドファイルを
右クリックして「プロパティ」を選択



③ 「Actionscript」タブ

④ 「Actionscript 用に書き出し」をチェック

⑤ クラス名を付ける
例： sound1

⑥ 「OK」をクリック
警告にも「OK」

スクリプトの記述

```
var click_sound:Sound = new sound1();        サウンドインスタンスの生成
・・・
・・・
click_sound.play();                  鳴らしたい場所で play()メソッドを実行
・・・・
・・
```

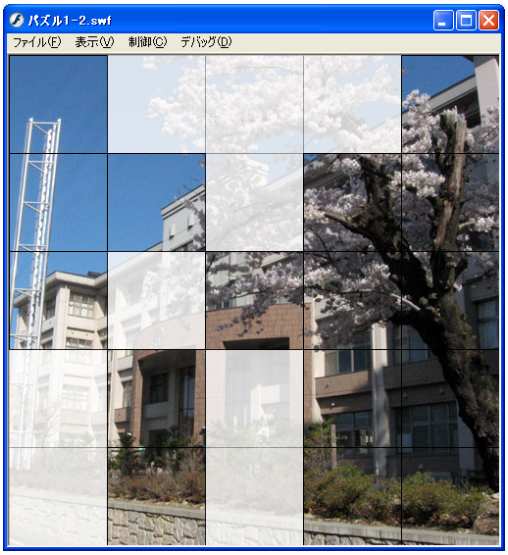
サウンド関連のメソッド

Sound．stop（）	サウンドの停止
Sound．play（0，10）	繰返し再生（ループ）　１０は繰返し回数

課題１５　効果音を入れて、ライツアウトを完成させよ。

課題１５　ライツアウトとりあえずの完成版（メインタイムラインの１フレームへ記述）

```
var kx,ky,no;
var click_sound:Sound = new sound1();

for(ky=0;ky<5;ky++){
        for(kx=0;kx<5;kx++){
                var new_masu:MovieClip = new masu();
                addChild(new_masu);
                new_masu.name = "masu" + (ky*5+kx);
                new_masu.x = kx*100;
                new_masu.y = ky*100;
                new_masu.img1.x = -kx*100;
                new_masu.img1.y = -ky*100;
                new_masu.addEventListener(MouseEvent.CLICK, masuClick);
        }
}

for(var i=0;i<3;i++){
        no = int( Math.random()*25 );
        masu_no( no );
}

function masuClick(event:MouseEvent):void
{
        var mcTarget:MovieClip = MovieClip(event.currentTarget);
        var no = parseInt(mcTarget.name.substring(4));
        click_sound.play();
        masu_no( no );
}

function masu_no( no ):void
{
        rev( no );
        if( no-5 >=  0 ) rev( no-5 );
        if( no+5 <= 24 ) rev( no+5 );
        if( no%5 !=  0 ) rev( no-1 );
        if( no%5 !=  4 ) rev( no+1 );
}

function rev( no ):void
{
        var mc_name = "masu"+no;
        var mcTarget:MovieClip = MovieClip(getChildByName(mc_name));
        if(     mcTarget.alpha == 1.0)
                mcTarget.alpha = 0.2;
        else
                mcTarget.alpha = 1.0;
}
```



Flash ムービーを  ipad iphone Android アプリ として生成する

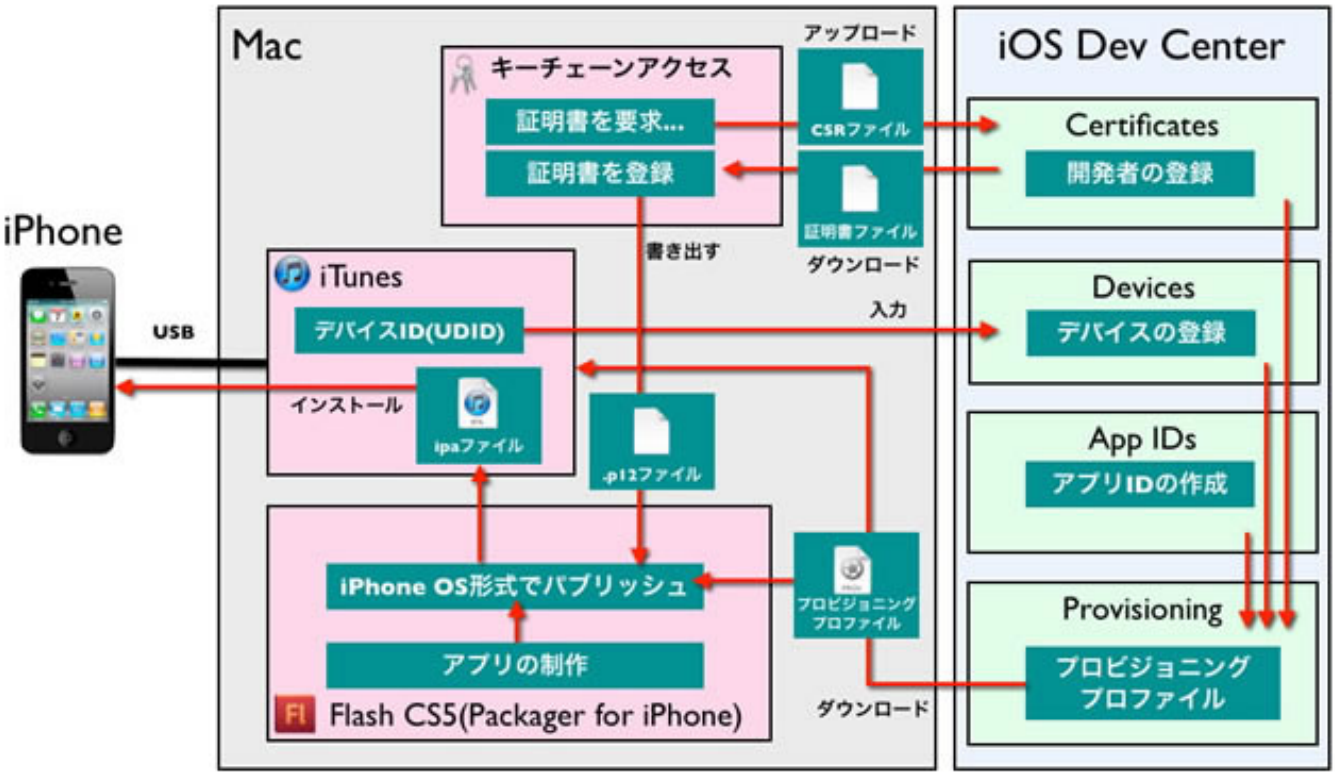
Apple 社は誰でも iphone/ipad のアプリを開発し AppStore で配布できるよう開発キット（iOS SDK）と開発ツール(Xcode)を無償で公開・配布しています。この公式な方法の他にも iphone の開発ツールはいくつか存在しますが、開発環境はＭａｃに限定されており、Windows ベースでは使用することができません。しかし、Flash は CS5.5 からパッケージされた AIR for iOS という機能を使って Flash で作成したアプリケーションを iPad/iPhone で利用できる形式に変換することが可能になりました。(Android 形式も可能)

今回作成した Flash ムービーを iPad 用アプリに書き出して、実機で実行させてみましょう！

iPad/iPhone アプリ開発の流れ

① Apple 開発ライセンスの取得（開発者の登録）
② 開発証明書の発行
③ テスト機の登録（デバイスの登録）
④ アプリケーションＩＤの作成
⑤ プロビジョニングファイルの作成
⑥ Flash によるアプリ開発、ipa ファイルの生成
⑦ iTunes で実機に転送
⑧ 実機での実行

今回はこの部分から行う



課題１６　課題８「アナログ時計」を iPad で実行する

① 課題８を開き、課題１６として名前を付けて保存する。

本来ならｉＰａｄやｉＰｈｏｎｅの画面サイズに合わせてムービーを作り直すべきだが、今回はこのサイズのままで出力してみる。（もちろん変更してもよい）

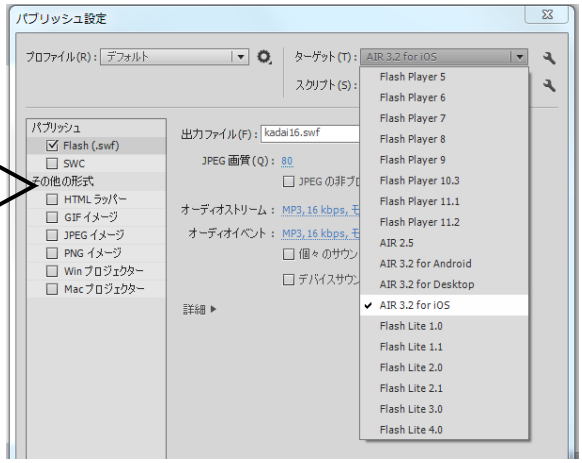
（参考）各端末の画面サイズ

iPhone 3Gs	480px × 640px	
iPhone 4/4s	640px × 960px	
iPhone 5	640px × 1136px	
iPad（第 1、第 2 世代)	768px × 1024px	
iPad（第 3、第 4 世代）	1536px × 2048px	今回の実習で使用
iPad mini	768px × 1024px	

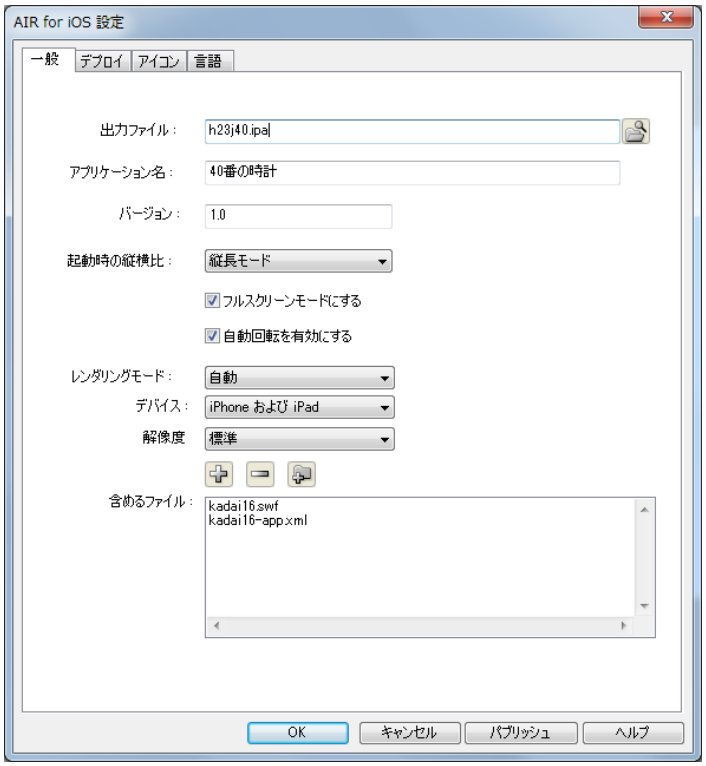
② アプリのアイコンを photoshop で作成し（114px×114px　png 形式）自身のフォルダへ保存する。

③ 「ファイル」→「パブリッシュ設定」

④ 「ターゲット」から「AIR3.2 for iOS」を選択し「ＯＫ」



⑤ すると「ファイル」メニューに「AIR3.2 for iOS 設定」が表示されるようになります。
これを選択します。



40 番の人を例に説明します。

⑥　「一般」タブ

「出力ファイル」

h23j40.ipa

「アプリケーション名」

40 番の時計

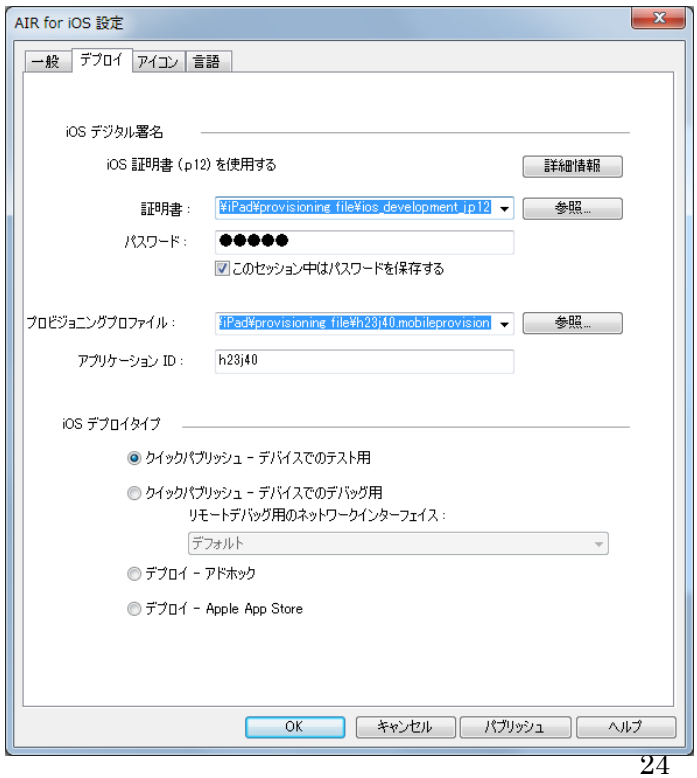
⑦　「デプロイ」タブ

「証明書」　共通サーバー　……¥iPad¥provisioning file¥ios_development_j.p12

「パスワード」→「okako」　「このセッション…」にチェックを入れる

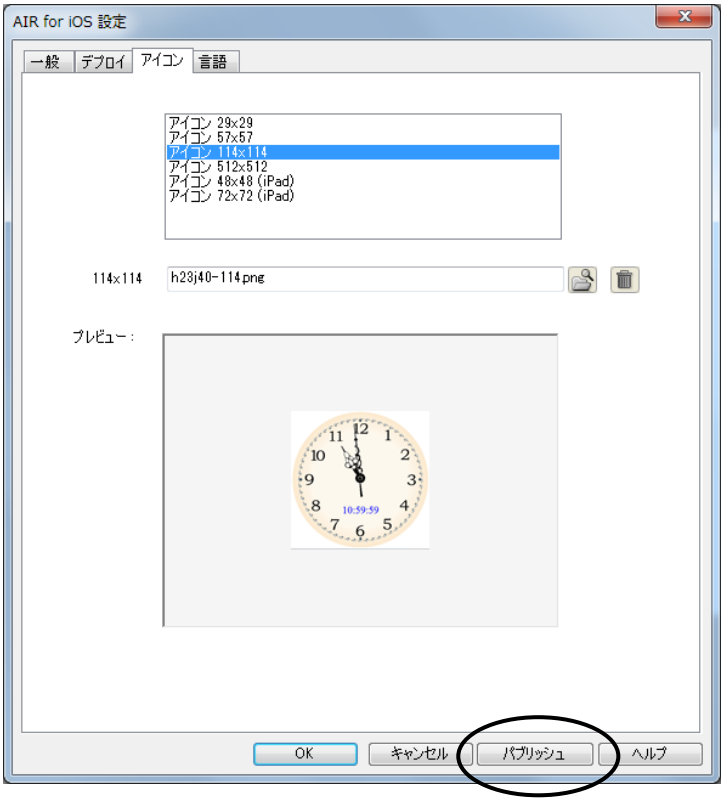
「ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞﾌｧｲﾙ」　共通サーバー　……¥iPad¥provisioning file¥h23j40.mobileprovision

「ｱﾌﾟﾘｹｰｼｮﾝ ID」　「h23j40」　←　**自分のユーザーＩＤ**　↑



ｉＯＳデプロイタイプ

「クイックパブリッシュ」にチェック



⑧　「アイコン」タブ

「アイコン 114×114」を選択し、別に作成済みの「．png」ファイルを読み込む。

⑨　以上の設定が済んだら「パブリッシュ」ボタンをクリックします。

３、４分でｉアプリ形式のファイル「h23j40．ipa」が出力されます。

ｉＰａｄ用アイコンファイルの作成（ｐｎｇ形式）

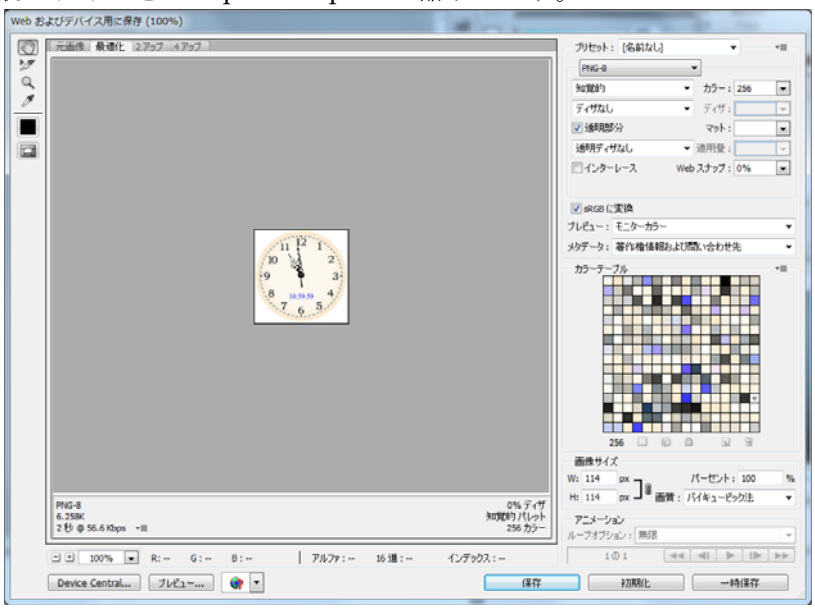
ｉアプリには iPad、iPhone、パソコンなどのためにサイズの異なる数種類のアイコンを指定する必要があります。今回はｉＰａｄ用の「114px×114px」の大きさのものを用意します。

①　」Photoshop で正方形の画像を編集後、
「イメージ」→「画像解像度」から横縦のサイズを 144px×144px　に縮小します。

②　「ファイル」→
「Web およびデバイス用に保存」

③　「png8」か「png24」を選択し
「保存」ボタンをクリック

④　名前を付けて、自身のフォルダへ保存します。



実機（ｉＰａｄ）への転送・実行

① 生成された「○○○．ｉｐａ」を共通サーバー ……¥ipad 転送用 フォルダへコピーします。

② ｉｐａｄが接続されたパソコンへ移動し、ｉＴｕｎｅｓを起動し操作します。

③ すでにｉＰａｄ内に存在する同じファイルを転送したい場合は、ｉＰａｄ内とｉＴｕｎｅｓ内の該当ファイルを削除しておきます。

④ 右上の「iPad」ボタンをクリック

⑤ ｉＰａｄの２ページ目を選択

⑥ 画面左側の該当ファイルをｉＰａｄの２ページ目へドラッグします。

⑦ 右下の「適用」ボタンをクリック

⑧ 転送が行われ実行可能になります。

課題１７　課題 14「簡単なゲーム」か課題 15「ライツアウト」をｉＰａｄ用に修正し実機へ転送して実行せよ。

※今回の例題で使用したマウスイベントは「クリック」のみである。これはｉＰａｄ用に書き出した場合、自動的に「タップ」の動作に置き換えられる。ｉＰａｄ特有の「スワイプ」や「ピンチアウト」といった動作をさせるためには、それ用のスクリプトを別に記述する必要がある。

※Ａｎｄｒｏｉｄ端末用のアプリへ書き出すのも同じような方法だが、作成には「Adobe AIR」のランタイムが必要になるため、あらかじめ端末へインストールしておく必要があります。

「スタート」ボタンをタップしたら出題する・・・のようにｉＰａｄ用にスクリプトを追加、修正してみよう