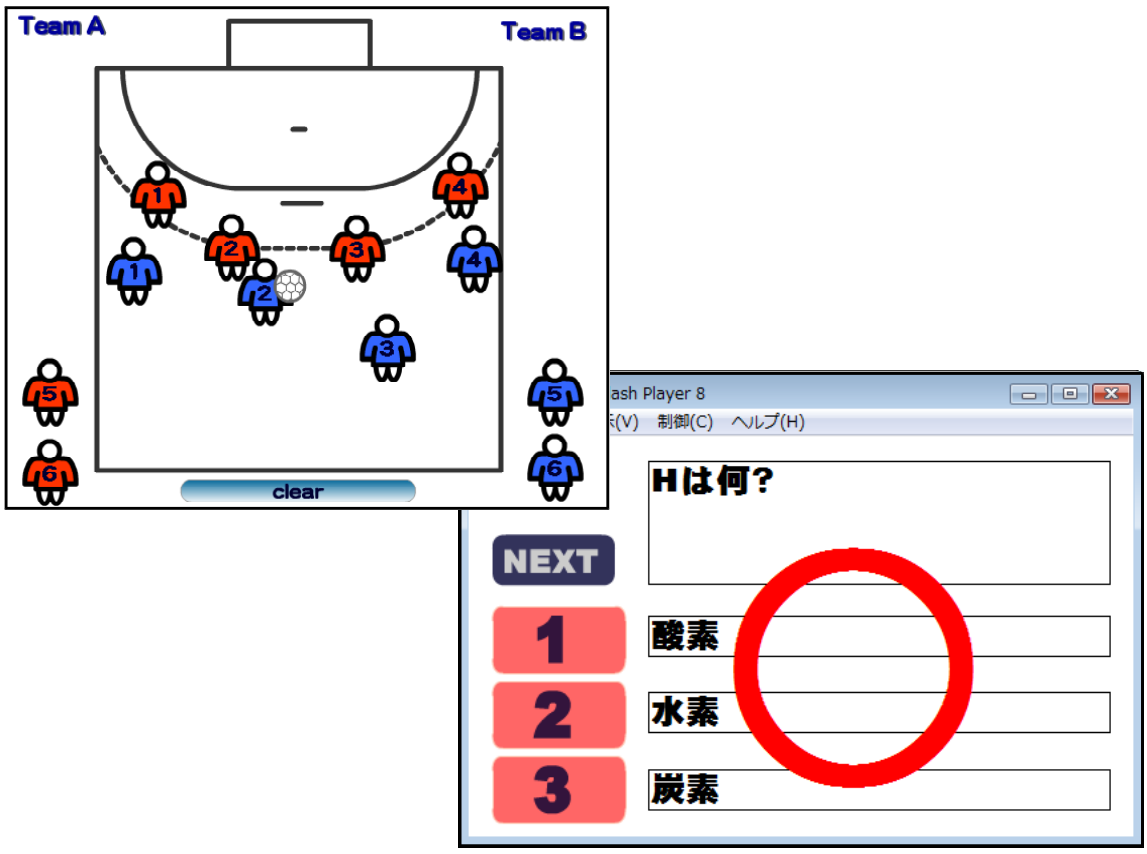
平成20年度

Flash を活用した教材作成研修講座

応用(ActionScript)編



岩手県立総合教育センター
http://www1.iwate-ed.jp/

－－－　目　　次　－－－

第1章 ActionScriptとは?

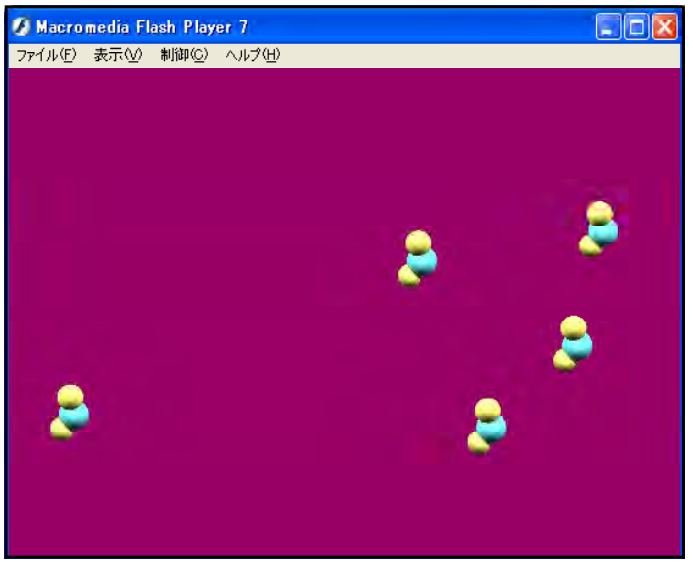
１　ActionScript

ActionScript とは、**Flash** に搭載されたオブジェクト指向のスクリプトです。**Flash** で作成したムービーに対して、命令を記述して実行することにより、ムービーを制御し、今まで以上にインタラクティブなコンテンツとして表現することができます。

２　ActionScriptが得意なこと

ActionScript を使用すると、通常の **Flash** ムービーに次のような機能や処理を追加することができます。

■ムービークリップの位置・サイズ・色などを動的にコントロール
■点数計算や移動場所の座標計算
■タイムラインのコントロール
■オプションボタンやリストボックスなどの使用
■マウスの位置に合わせたムービークリップ制御
■ムービークリップのドラッグ＆ドロップ
■ムービークリップの入れ替え
■キーボードのキー判断
■サウンドコントロール
■文字列表示
■外部テキストの読み込み
■ **CGI** 利用
■クラスやオブジェクトの自作



左図のように、特定の粒子が画面内をランダムに動き回る場合を考えてみましょう。壁にぶつかって反射し、粒子どうしでぶつかって反射するような動きは、その瞬間での位置について衝突判定を判別する必要があります。

このようなコンテンツを作成する場合には、**ActionScript** が得意とするところです。

３　ActionScriptの実行環境

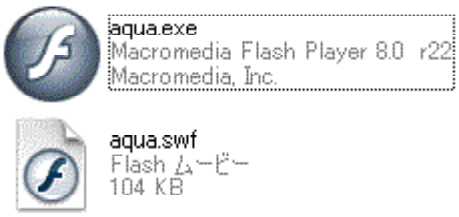
Flash ムービーの配布形式には、大きく分けて２つのパターンがあります。**ActionScript** を使用している場合も、同じように配布を行うことができます。

●ブラウザ上で再生

ブラウザ上で再生できる「*.swf」形式で配布する方法です。多くのブラウザは Flash ムービーを再生するためのプラグインや ActiveX コントロールを標準で備えているため、どのようなブラウザで見ても、Windows、Macintosh、そして UNIX でさえも同じ Flash ムービー＋ ActionScript で同じようにコンテンツを表示することができます。

●プロジェクタで再生

「プロジェクタ」と呼ばれる実行形式のファイルで再生する方法です。プロジェクタは Flash の開発環境があればすぐに作成することができます。ブラウザで再生する場合には、ブラウザがなかったり、最新のプラグインや ActiveX コントロールがなかったりする場合には再生されませんが、実行形式のプロジェクタならば単体のファイルのみでムービーを再生することができます。



４　ActionScriptの安全性

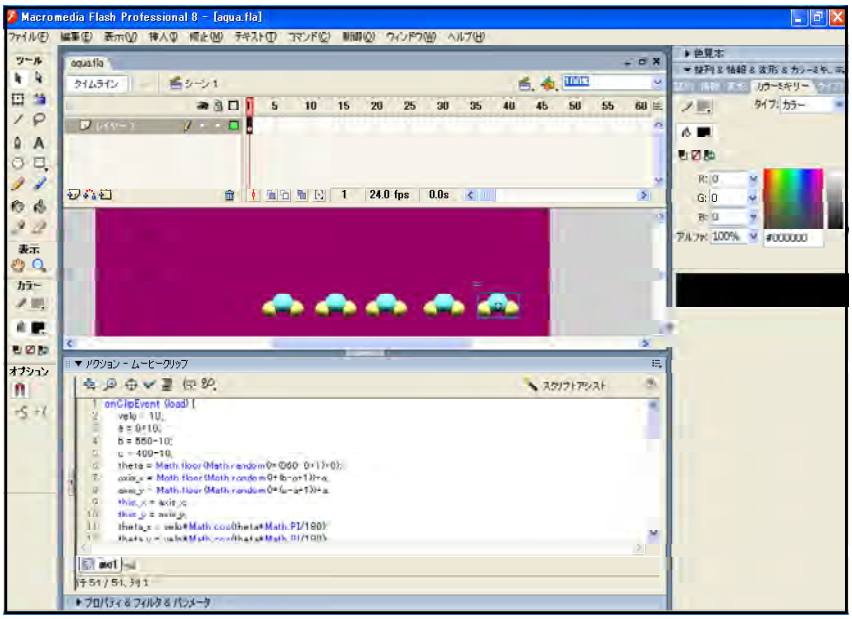
Web に公開する場合に編集するファイル（*.fla）を Flash ムービー（*swf）に変換しますが、このとき ActionScript も書いたスクリプトそのままではなく、FlashPlayer 用のコードに変換されます。swf 形式はオープンフォーマットで、そのファイルの仕様は公開されています。そのため、コードから ActionScript へ戻す解析ツールもあるので、パスワードや重要な情報は、ActionScript 内に残さないようにしましょう。また、複雑な処理も容易に書けるようになりましたが、あまりに重い処理をさせると FlashPlayer 側で ActionScript の実行を停止して、PC 全体に影響を及ぼさないようになっています。

５　ActionScriptの作成方法

ActionScript を記述するには、まずムービーやムービークリップ、サウンドを用意しておいて、それに対して動きを付けていきます。

具体的に ActionScript を記述する作業は、「アクションパネル」を使用します。作成したムービーのタイムラインやムービークリップなどを選択し、メニューから［ウィンドウ］→［アクション］とたどるか、［F2］キーを押すとした図のようなアクションパネルが表示されます。

このアクションパネルを使用してスクリプトを記述していきます。アクションパネルは、選択した場所（フレーム、インスタンス、ボタンなど）に対してスクリプトを記述し、すでにスクリプトが記述されている場合には、そのスクリプトを表示します。

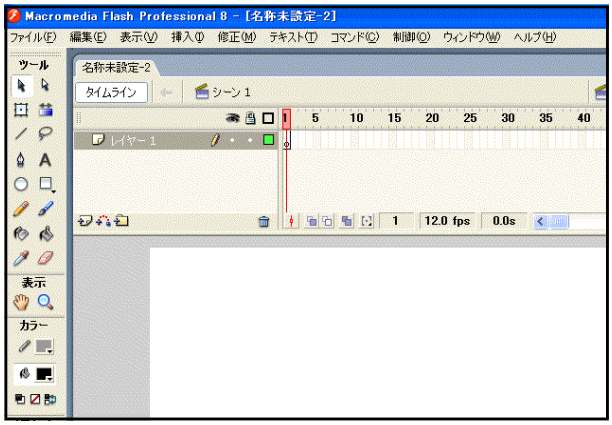


１　ドラッグ＆ドロップできる ムービーの作成

(1) Flash の起動

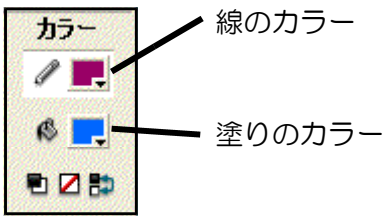
(a) ［スタート］から［すべてのプログラム］－［Macromedia］－［Macromedia Flash 8］をクリックします。

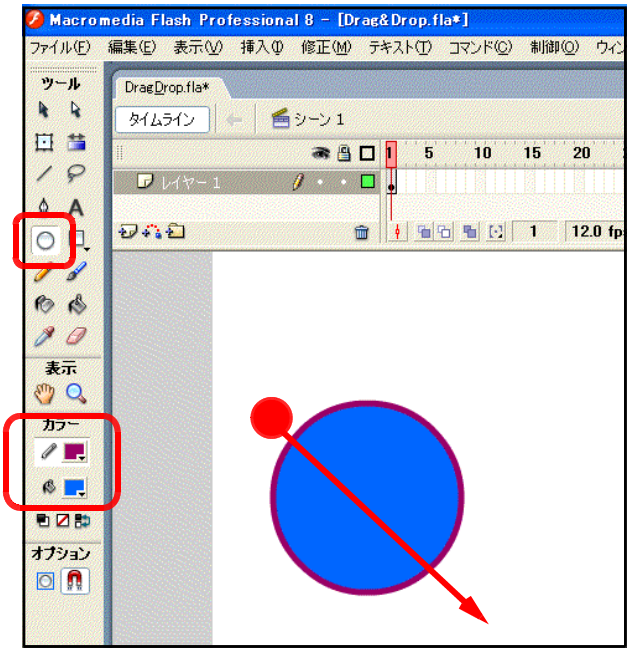
(b) Flash のメニューバーから[ファイル]－［新規］もしくは、［Flash ドキュメント］をクリックします。



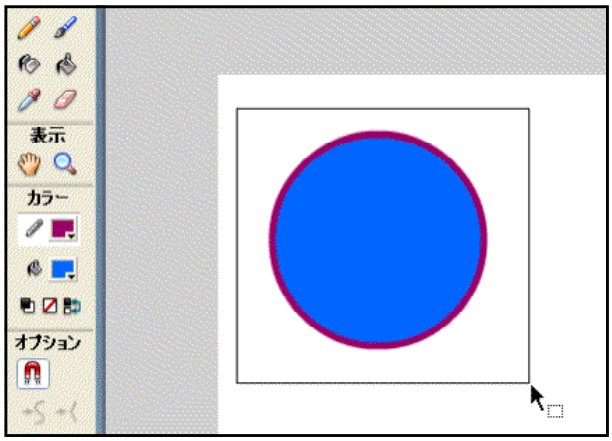
(2) ムービークリップの作成

(a) ツールパネルの［楕円ツール］  をクリックして、キャンパス上に円を描きます。このとき、事前に［線のカラー］と［塗りのカラー］を設定しておきます。



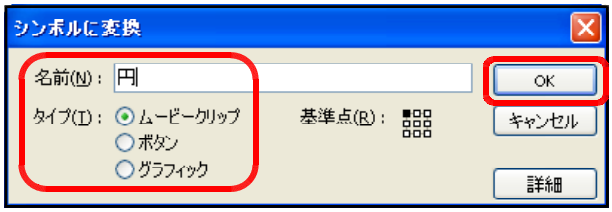


(b) ツールパネルから［選択ツール］をクリックして、円全体を範囲指定します。

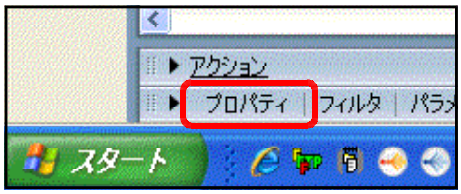


(c) メニューから［修正］－［シンボルに変換］をクリックします。

(d) ［シンボルに変換］ダイアログが表示されますので、［名前：］を『円』、［タイプ：］は『ムービークリップ』にチェックして、［**OK**］ボタンをクリックします。



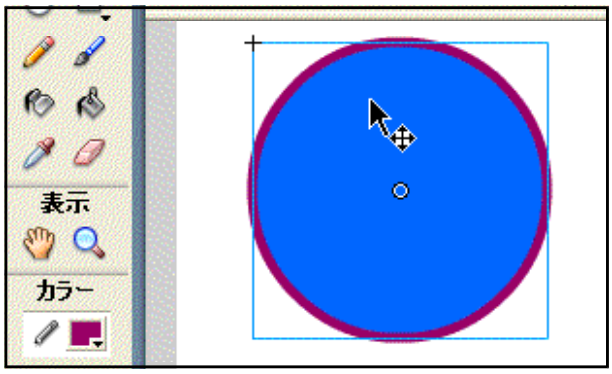
(e) 画面下の［プロパティ］をクリックし、プロパティパネルを表示させます。



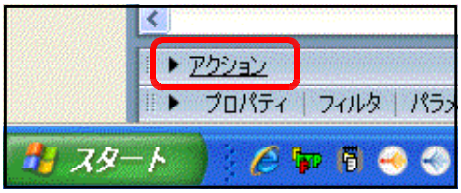
(f) ムービークリップに『円』と名前を入力します。これで描画された円は、シンボルとして変換され、『円』という名前を付けました。

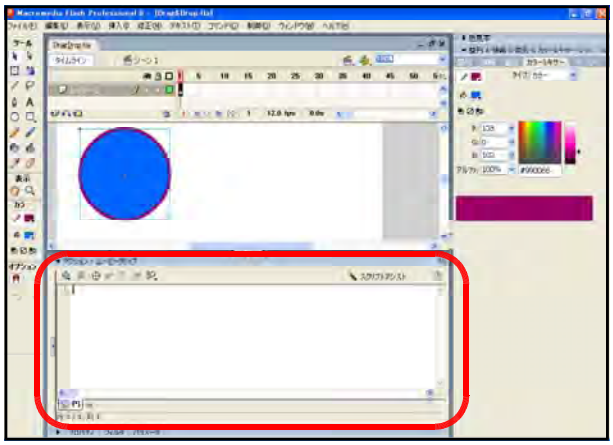
(3) ActionScript の設定

(a) ツールパネルから［選択ツール］をクリックし、円のムービークリップをクリックして選択します。



(b) 画面下の［アクション］をクリックして、アクションパネルを表示させます。



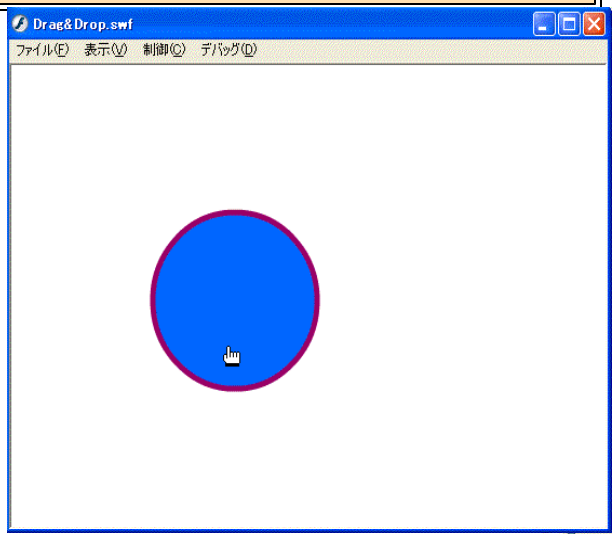


(c) 以下のスクリプトをアクションパネルに記述します。

```
on (press) {
    this.startDrag();
}
```

※ マウスのボタンが押されたら（press）、このインスタンスは、ドラッグ開始（startDrag）です。

（参考）ここまでの設定で、どのような動作をするか確認してみましょう。
［Ctrl］キーと［Enter］キーを同時に押して、ムービーのプレビューを表示させてみます。
一度クリックすると、円のムービークリップがついて回る（もう離れません）ことが確認できます。

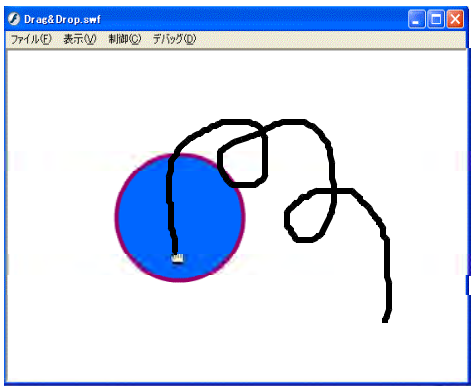


(d) 次にマウスボタンから手が離れたら、その場所にムービークリップが止まるためのスクリプトを入力します。先のスクリプトに続いて、以下のスクリプトを記述します。

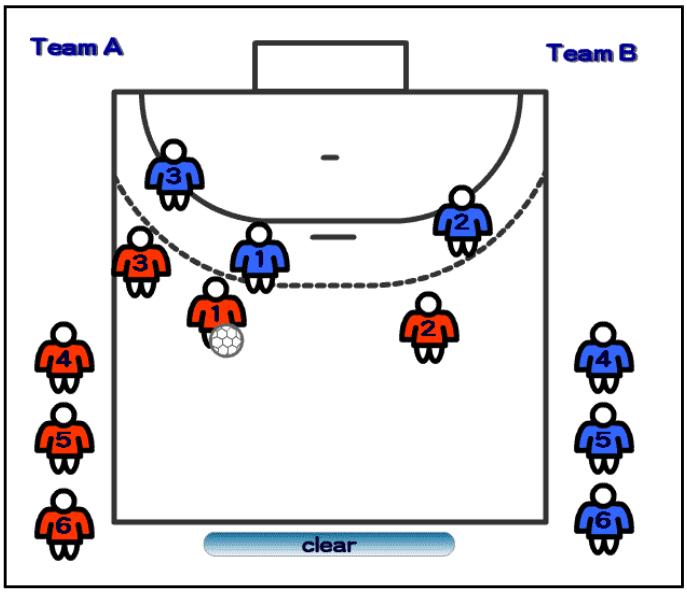
```
on (release,releaseOutside) {
    this.stopDrag();
}
```

※ マウスのボタンが戻ったら（release）、このインスタンスは、ドラッグ終了（stopDrag）です。

これで画面上の円に対して、ドラッグ＆ドロップができるようになりました。
自由に移動してみてください。



２　応用：作戦盤の作成

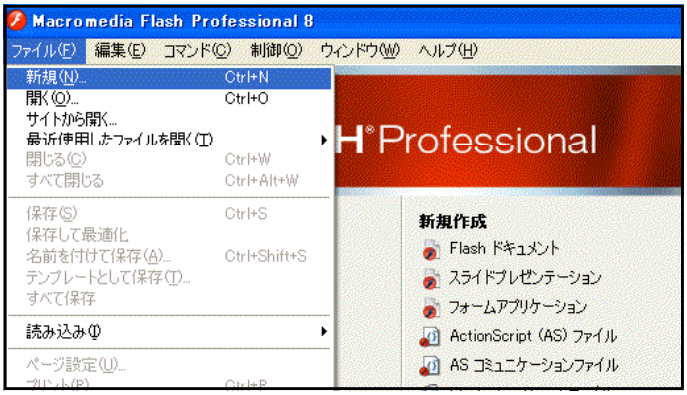


それでは、先に作成した「ドラッグ＆ドロップできるムービー」を応用して、作戦盤を作成してみましょう。

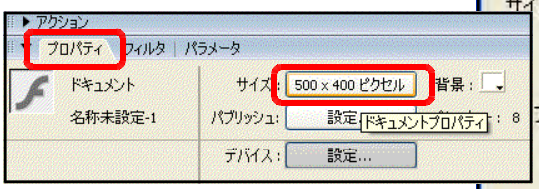
手順は簡単です。背景にその競技のコードを配置して、さらに２チーム分の選手のアイコンを配置して、ドラッグ＆ドロップができる **ActionScript** を適用させれば、作戦盤が完成です。

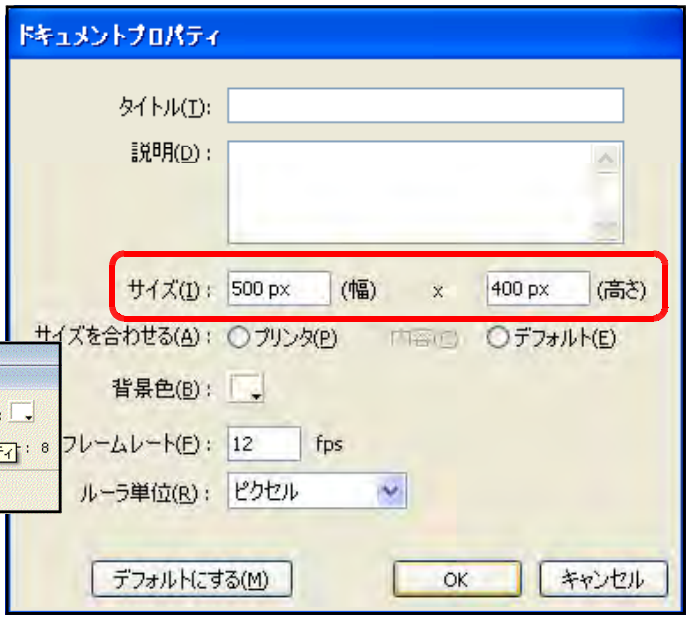
(1) 背景の作成

(a) **Flash** を起動して、［ファイル］－［新規］をクリックして、新しいドキュメントを準備します。



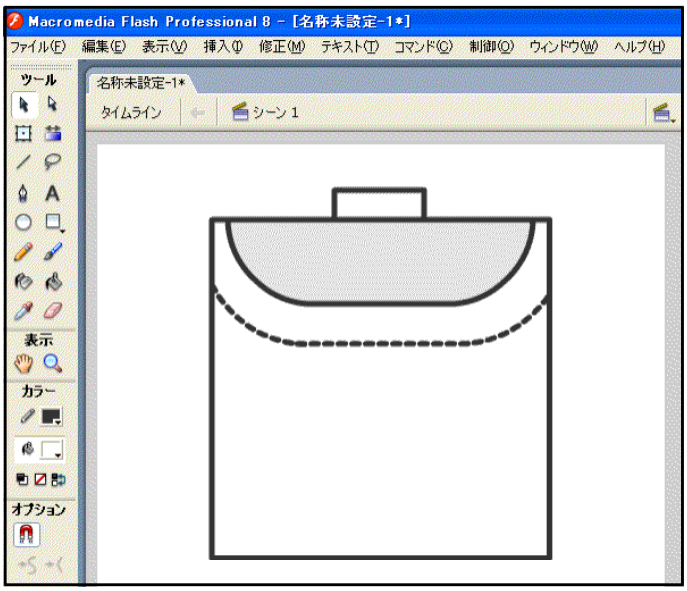
(b) ドキュメント上を一度クリックします。画面下の［プロパティ］をクリックし、サイズボタンをクリックします。［ドキュメントプロパティ］のダイアログが表示されますので、サイズを［**500px**］×［**400px**］に設定して、［**OK**］ボタンをクリックします。





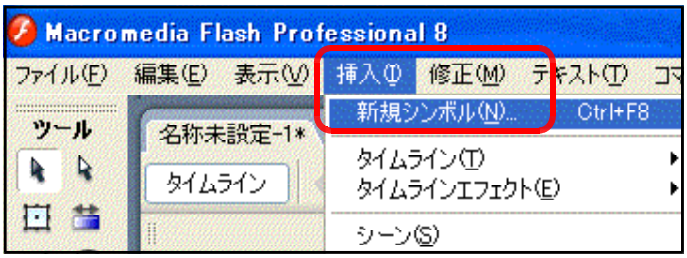
(c) ［タイムライン］をクリックして、レイヤー名を［背景］に変更します。

(d) 矩形ツールや楕円ツールを用いて競技のコード（反面）を作成します。右図はハンドボールのコート半面です。

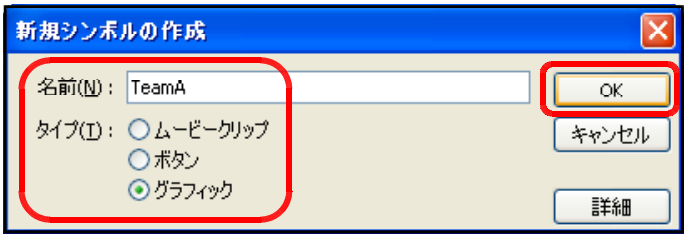


(2) 選手アイコンの作成

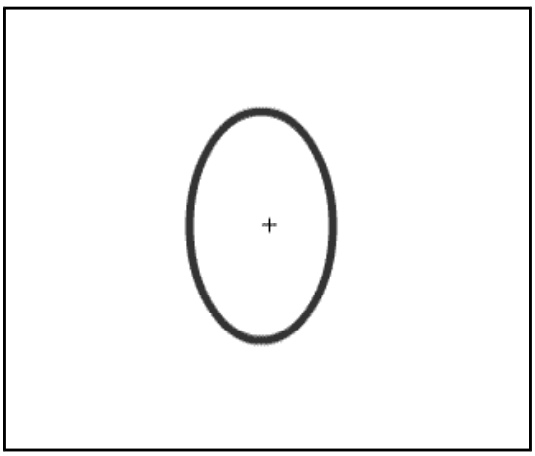
(a) メニューから［挿入］－［新規シンボル］をクリックします。



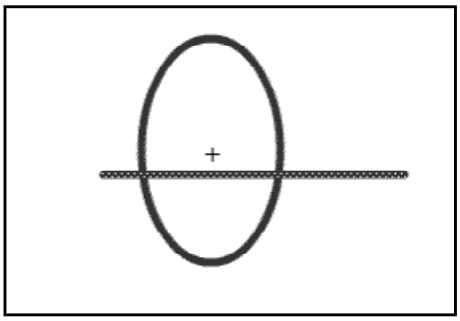
(b) ［新規シンボルの作成］ダイアログが表示されますので、タイプは［グラフィック］を選択し、名前を［**TeamA**］と入力して、［**OK**］ボタンをクリックします。



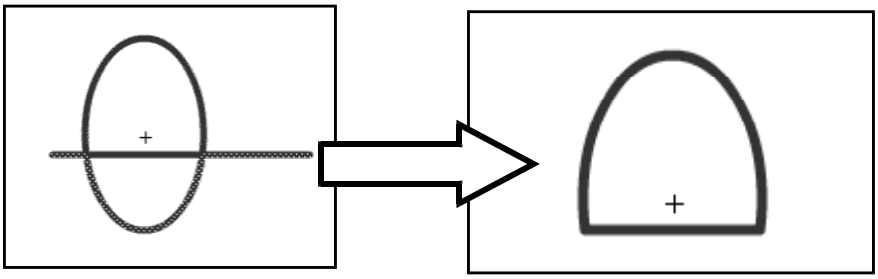
(c) 楕円ツールを用いて、任意のサイズで縦長の楕円を描画します。



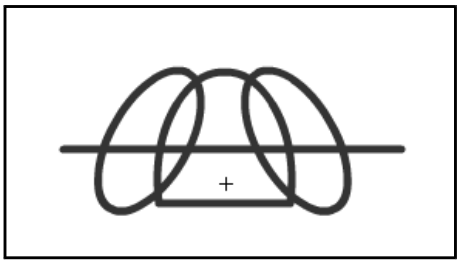
(**d**) 線ツール を用いて、楕円を横切るように線を描画します。



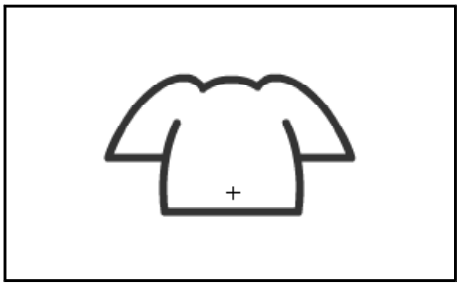
(**e**) 不必要な線を［**Shift**］キーを押しながら、選択して、［**Del**］キーをクリックして、削除します。



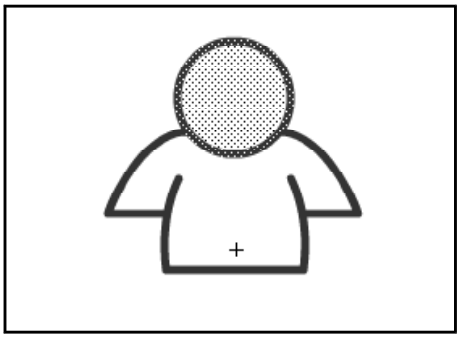
(**f**) さらに袖（そで）を作成するために、楕円と直線を組み合わせて描画します。



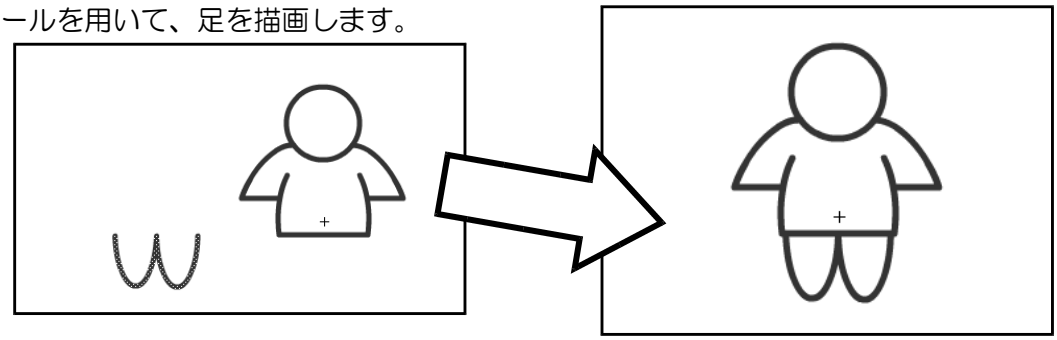
(**g**) 不必要な部分を削除して、シャツを完成させます。



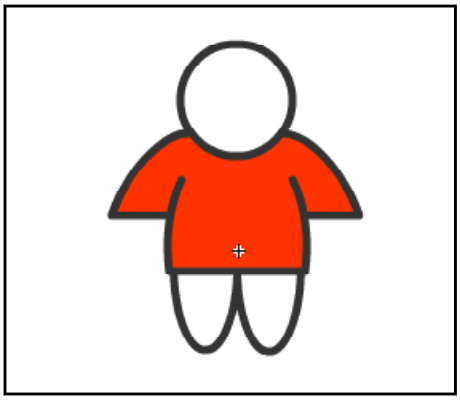
(**h**) 楕円ツールを用いて、顔を描画します。



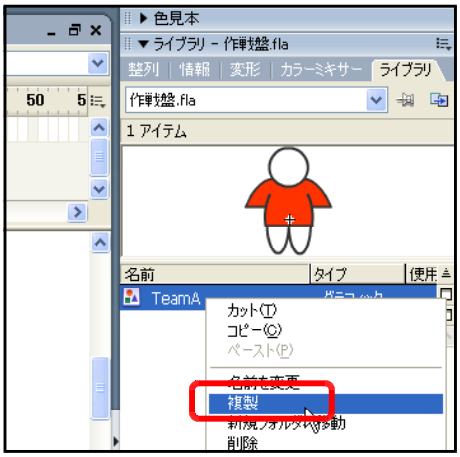
(i) 楕円ツールを用いて、足を描画します。



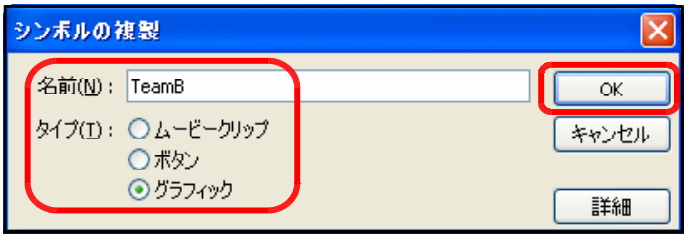
(j) バケツツールを選択して、任意の色で着色します。これで、［**TeamA**］というグラフィックシンボルが作成されました。



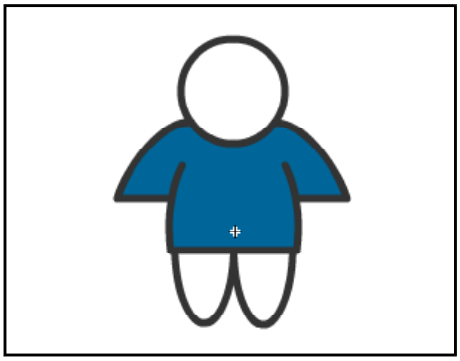
(k) 画面右のライブラリパネル［**TeamA**］上で右クリックから、［複製］をクリックします。



(l) 名前を［**TeamB**］に変更して、［**OK**］ボタンをクリックします。

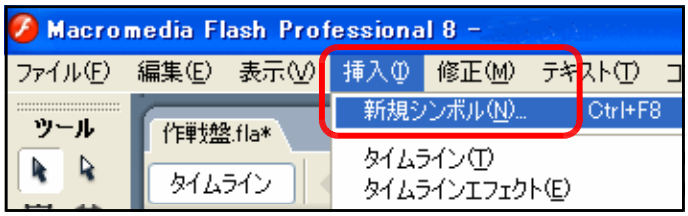


(m) ライブラリの［**TeamB**］をダブルクリックして選択します。バケツツールをクリックして、任意の色で着色します。これで **TeamB** のグラフィックシンボルが完成しました。

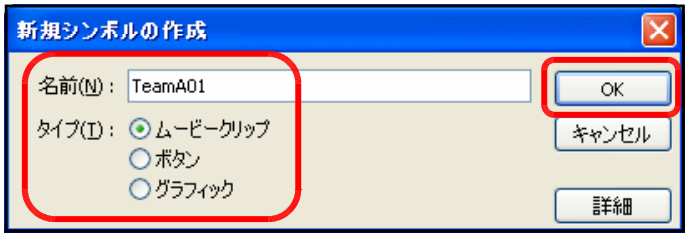


(3) 選手アイコンの作成２

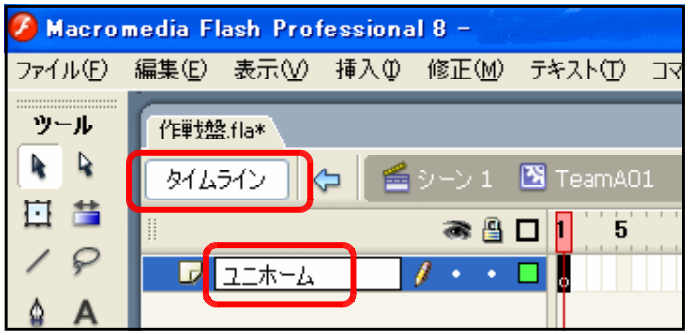
(a) メニューから［挿入］－［新規シンボル］をクリックします。



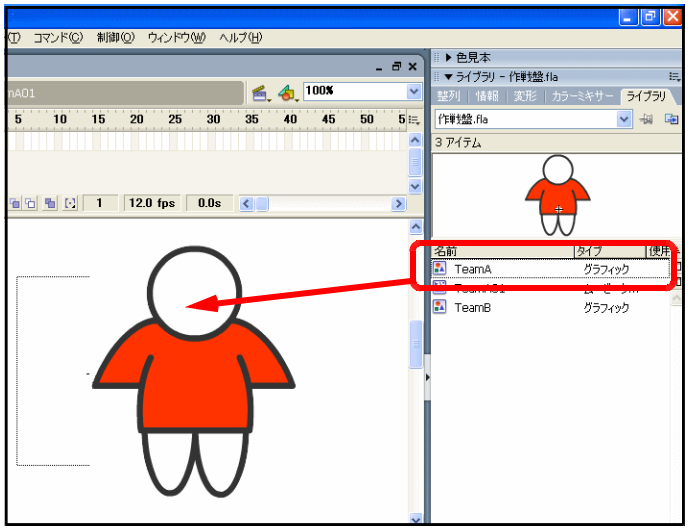
(b) タイプを［ムービークリップ］、名前を［**TeamA01**］に変更して、［**OK**］ボタンをクリックします。



(c) ［タイムライン］をクリックして、レイヤー名を［ユニホーム］に変更します。

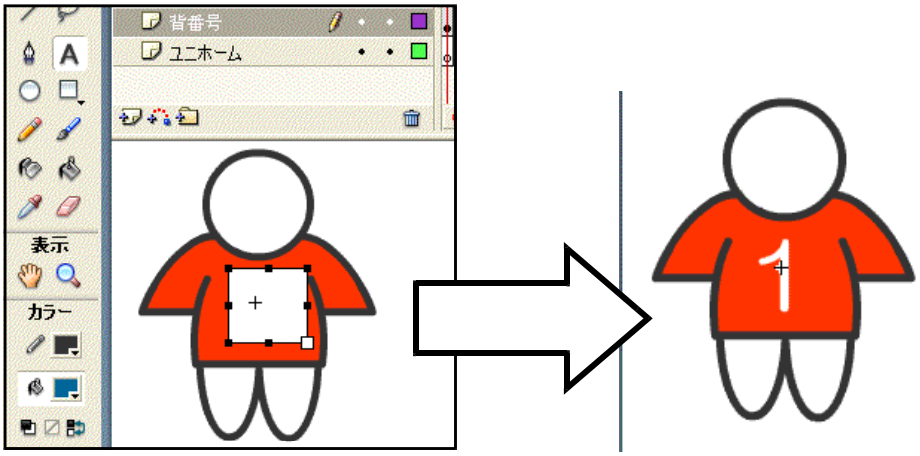


(d) ライブラリパネル内の［**TeamA**］グラフィックシンボルをステージ上にドラッグして配置させます。

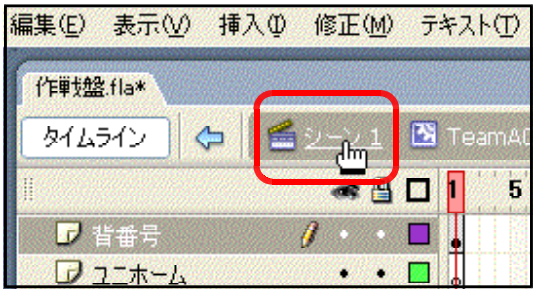


(e) ［レイヤーの追加］ボタンをクリックします。レイヤー名を［背番号］に変更します。

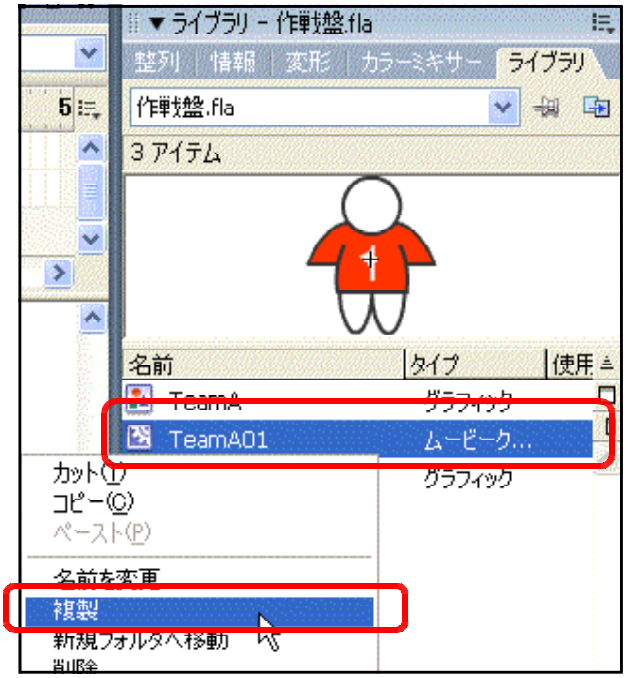
(f)　［テキストツール］をクリックして、［背番号］レイヤー上にテキストボックスを作成します。



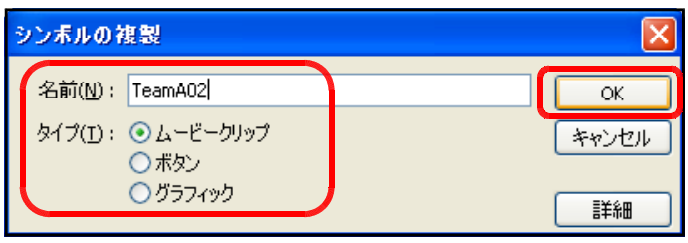
(g)　［シーン１］をクリックして、ドキュメントに戻ります。ライブラリには［**TeamA01**］ムービークリップシンボルが追加されていることを確認します。



(h)　ライブラリパネル内の［**TeamA01**］をクリックから、右クリックー［複製］をクリックします。

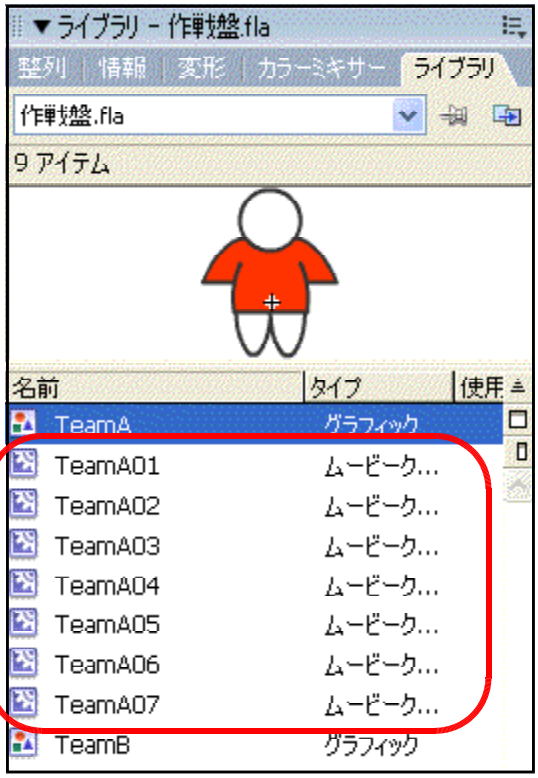


(i)　名前を［**TeamA02**］に変更して、［**OK**］ボタンをクリックします。

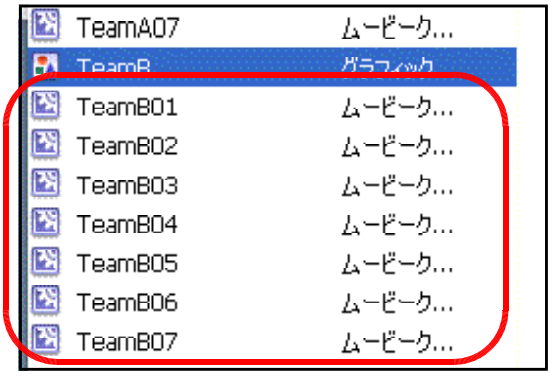


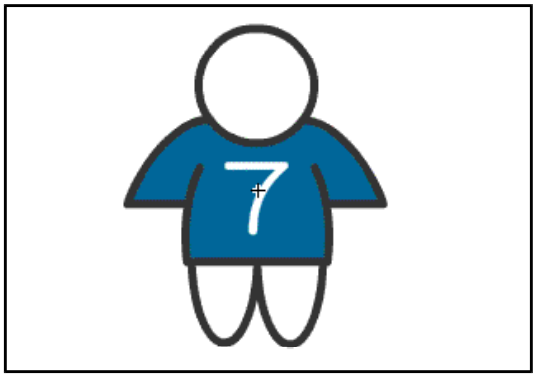
(j) テキストツールを選択して、背番号１の番号を［2］に変更します。

(k) 同様に上記の(h)～(j)を繰り返して、背番号３～７まで合計７個の選手アイコンを作成します。ただし、ムービークリップ名は、［TeamA01］～［TeamA07］とする。



(l) チームＢの選手アイコンの作成を上記操作(a)～(k)を繰り返して作成します。ただし、ムービークリップ名を［TeamB01］～［TeamB07］とします。

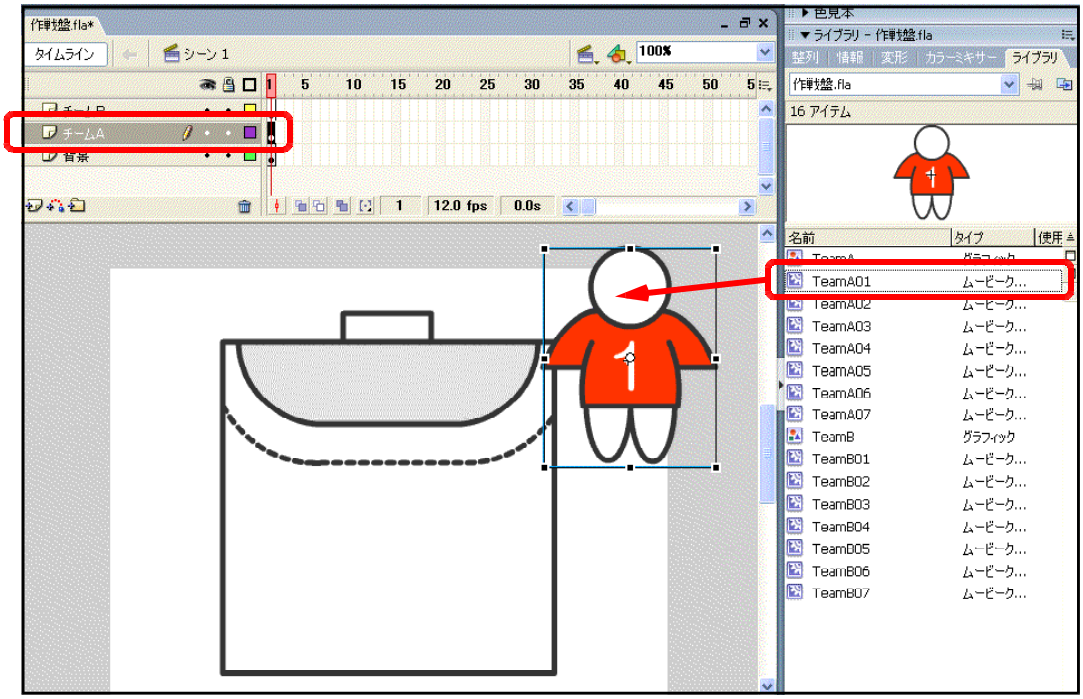




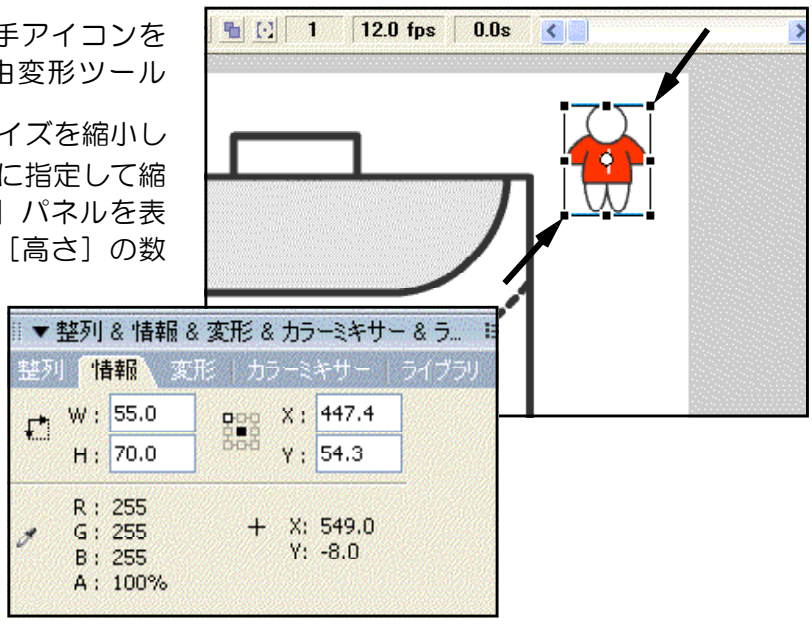
⑷ 選手アイコンの配置

⒜ ［レイヤーの追加］ボタンをクリックして、［背景］レイヤー上に［チームA］と［チームB］レイヤーを配置します。

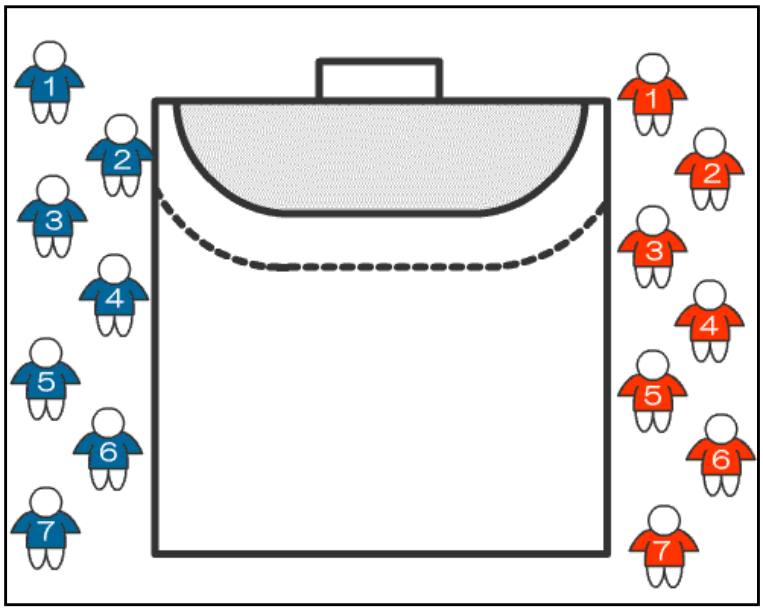
⒝ ［チームA］レイヤーをクリックして、ライブラリパネル内から［**TeamA01**］ムービークリップシンボルをステージ上にドラッグします。



なお、ステージ上に選手アイコンを重ねて大きい場合は、自由変形ツール  を選択して、適宜、サイズを縮小します。また、サイズを正確に指定して縮小したい場合には、［情報］パネルを表示させて、**W**［幅］と **H**［高さ］の数値で指定します。

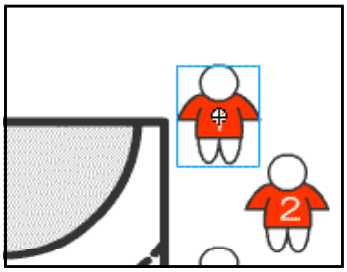


(c) (b)の操作をくり返して、ステージ上に各チームのアイコンを整列させます。

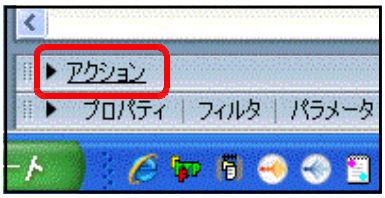


(5) ActionScript の設定

(a) ステージ上のチームＡの背番号１の選手アイコンをクリックして、選択します。

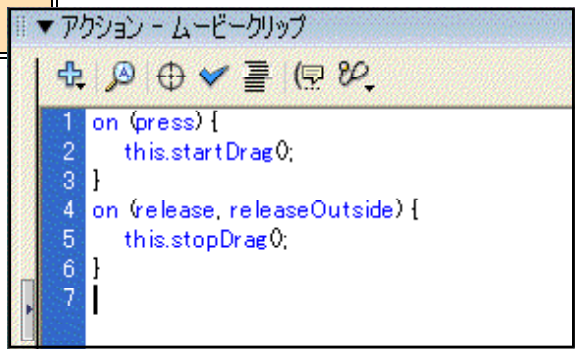


(b) ［アクション］をクリックして、アクションパネルを表示させます。

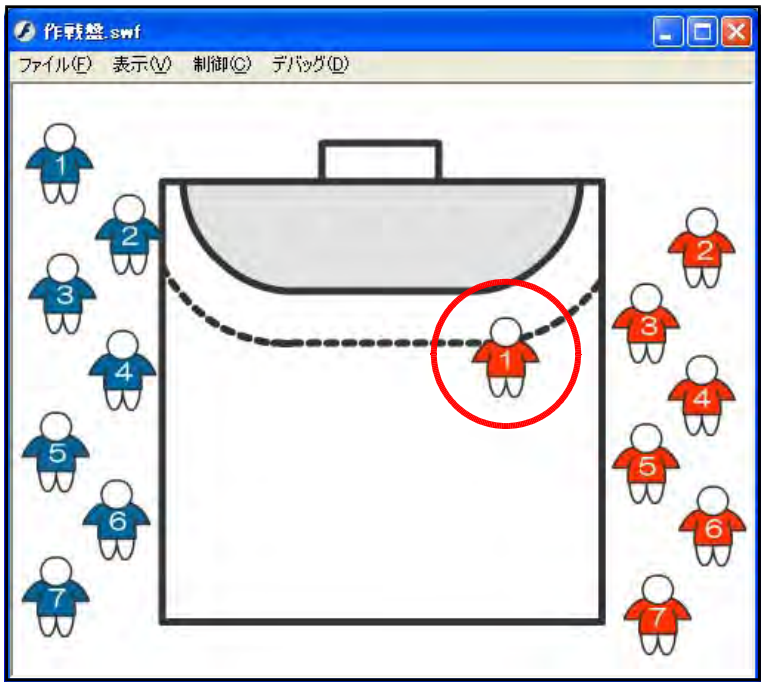


(c) アクションパネル内に以下のスクリプトを入力します。

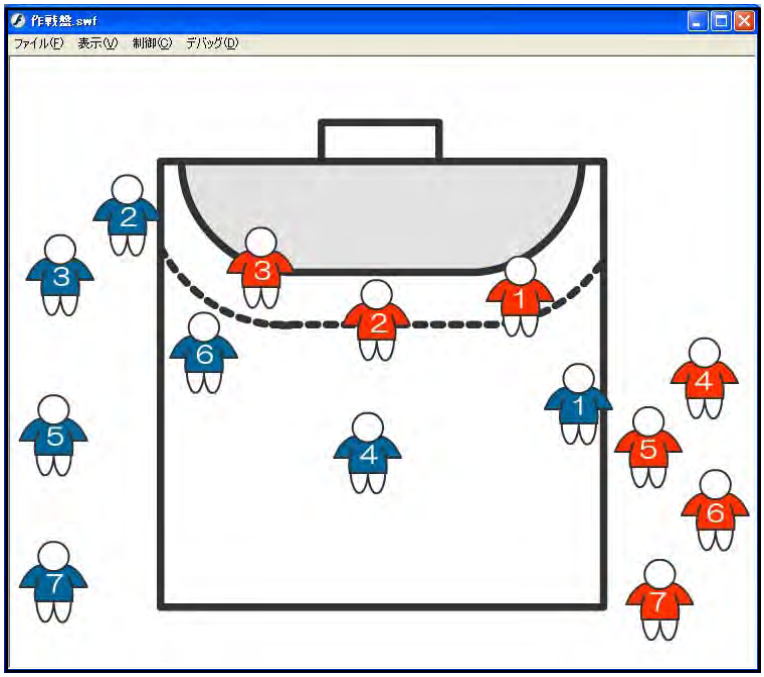
```
on (press) {
    this.startDrag();
}
on (release,releaseOutside) {
    this.stopDrag();
}
```



(d) ［Ctrl］キーと［Enter］キーを同時に押して、ムービーのプレビューを作成します。チームＡの１番の選手アイコンをドラッグして移動させてみます。



(e) 他の選手アイコンについても、(a)～(c)の操作をくり返して、ActionScript を設定します。これで作戦盤は完成です。各選手をドラッグして、フォーメーションの確認やディフェンス形態の確認ができます。



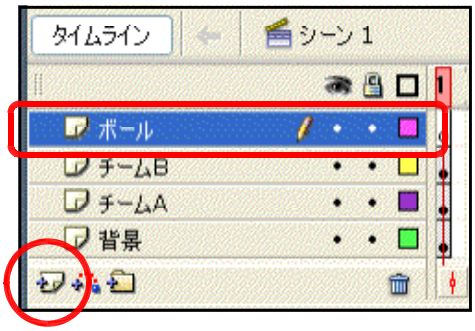
〈memo〉

作成した作戦盤を使ってみると、さまざま改善点が思い浮かびませんか？最低限の動きに加えて、バージョンアップしてみましょう。

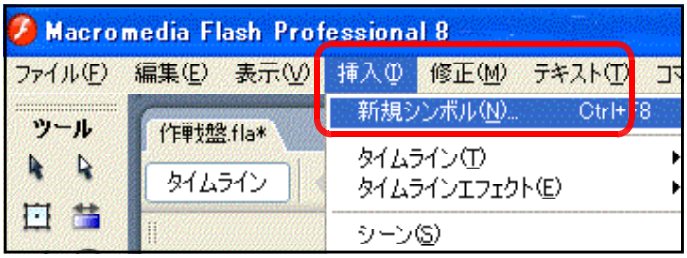
【追加機能１】　～　ボールがあればいい！！

もちろんドラッグ＆ドロップができるように、ボールのアイコンも同様に追加してみましょう。

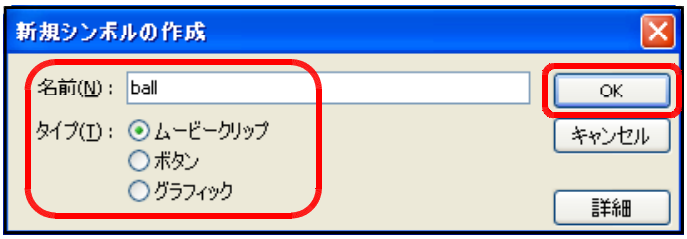
(a)　［レイヤーの追加］ボタンをクリックして、［チームB］レイヤー上に［ボール］レイヤーを作成します。



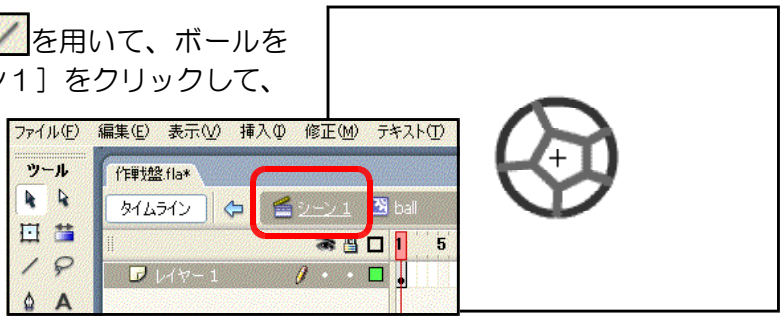
(b)　メニューから［挿入］－［新規シンボル］をクリックします。



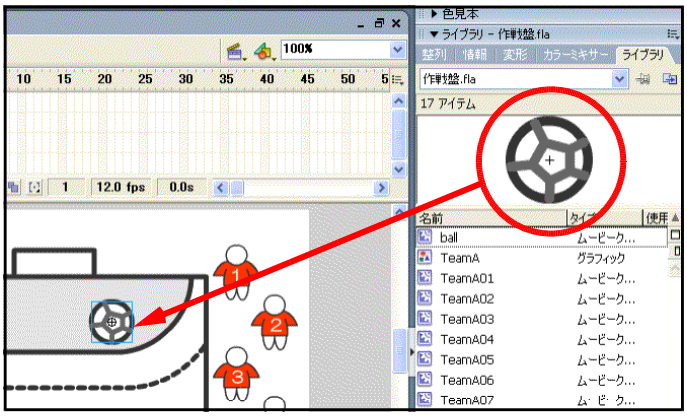
(c)　タイプを［ムービークリップ］、名前を［**ball**］として、［**OK**］ボタンをクリックします。



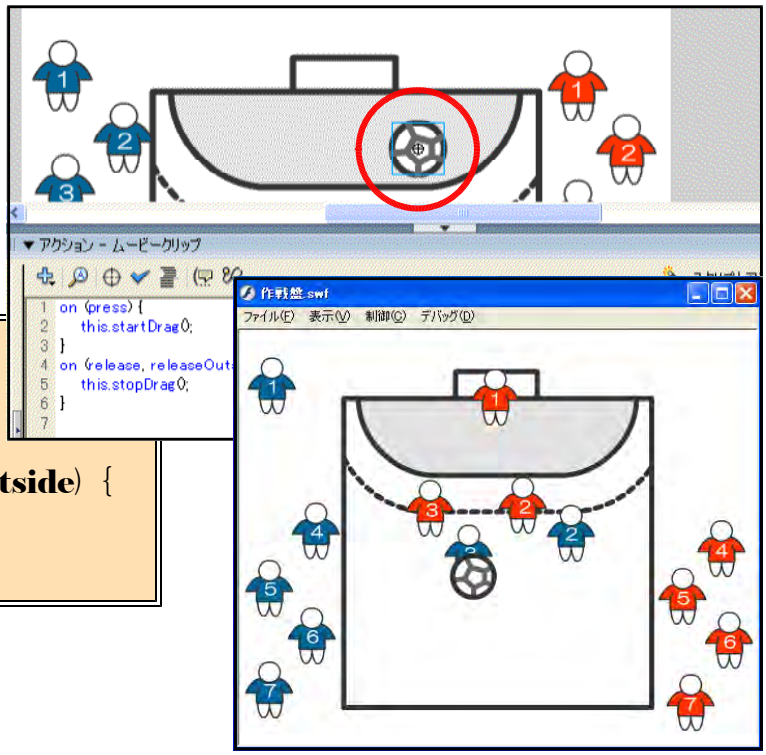
(d)　楕円ツール◯と線ツール／を用いて、ボールを描画します。その後、［シーン１］をクリックして、ドキュメントに戻ります。



(e)　ライブラリから［**ball**］ムービークリップをドラッグして、ステージ上に配置します。サイズが大きい時には、変形ツールをクリックして、適宜、サイズを変更します。



(f) ボールをクリックして、アクションパネルに以下の **ActionScript** を入力します。これで、ボールの動きも加えて、作戦盤を利用することができます。



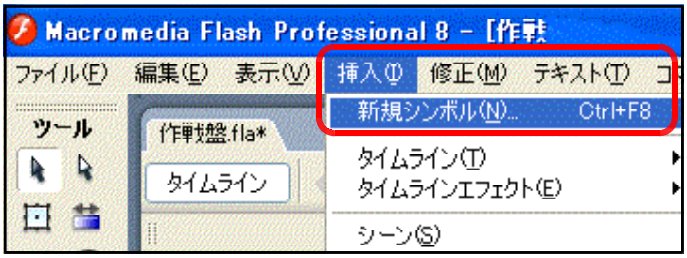

```
on (press) {
    this.startDrag();
}
on (release,releaseOutside) {
    this.stopDrag();
}
```

【追加機能２】　～　選手アイコンがワンクリックで整列できればいい！！

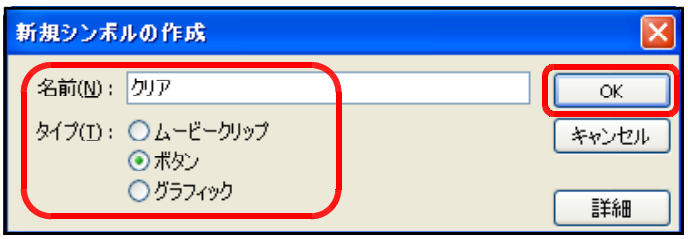
作戦盤を用いてフォーメーションの確認後、アイコンを整列させるためのボタンを追加して見ましょう。

(a) ［レイヤーの追加］ボタンをクリックして、［ボール］レイヤー上に［ボタン］レイヤーを作成します。

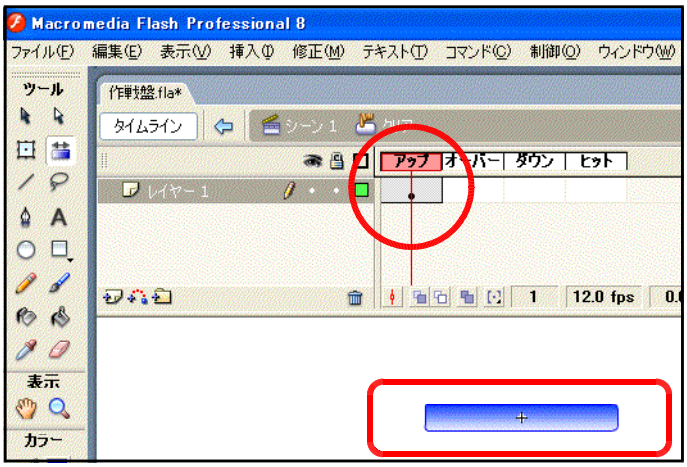
(b) メニューから［挿入］－［新規シンボル］をクリックします。



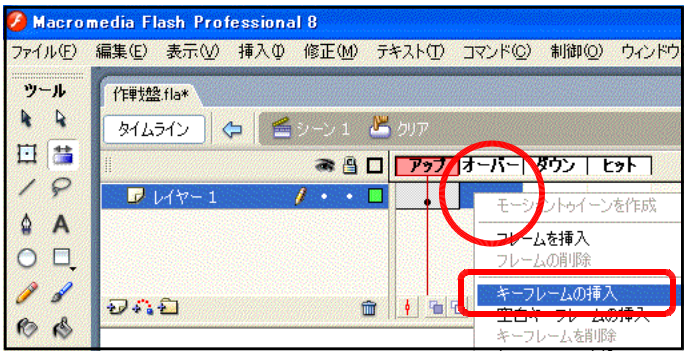
(c) タイプを［ボタン］、名前を［クリア］として、［**OK**］ボタンをクリックします。



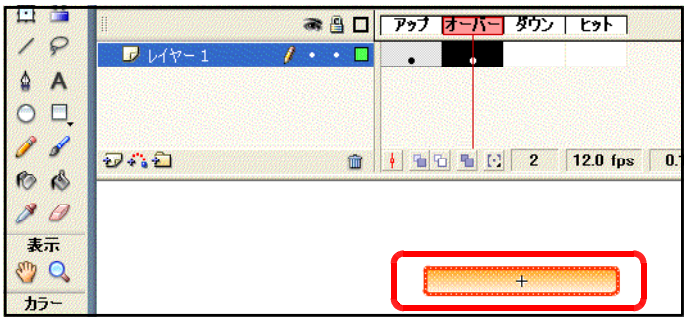
(d)　［レイヤー１］の［アップ］をクリックして、矩形ツールを用いて、ボタンを作成します。適宜、線のカラーや塗りのカラーを変更します。



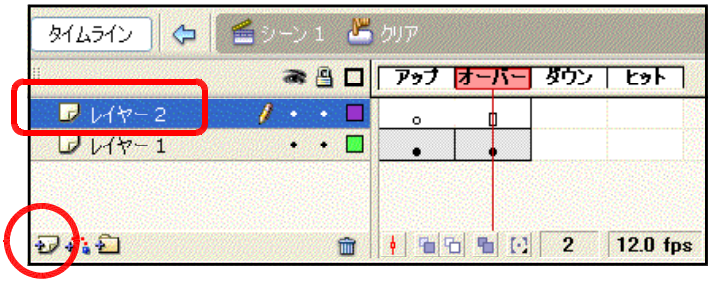
(e)　［レイヤー１］の［オーバー］上で右クリックから、［キーフレームの挿入］をクリックします。



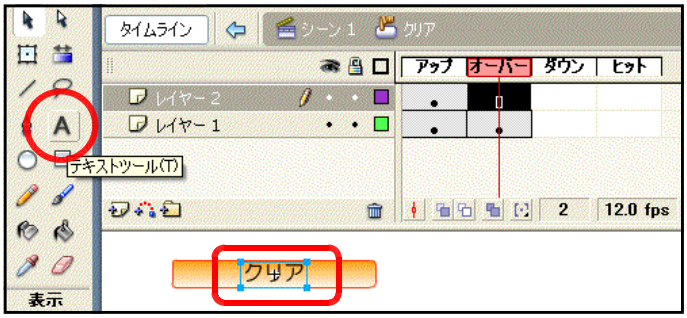
(f)　適宜、線のカラーや塗りのカラーを変更します。これで、通常時のボタンの配色とマウスのカーソルが重なった時で色が変化するようになります。



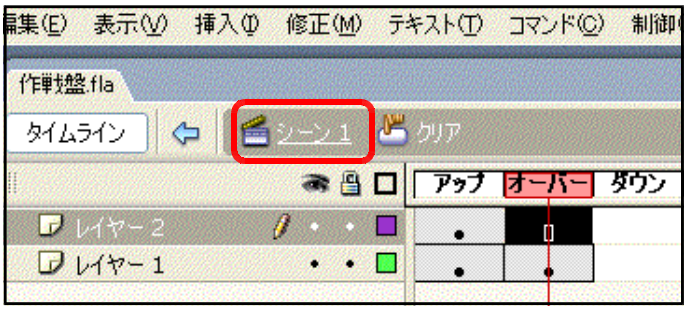
(g)　［レイヤーの追加］ボタンをクリックして、［レイヤー１］上に［レイヤー２］を作成します。



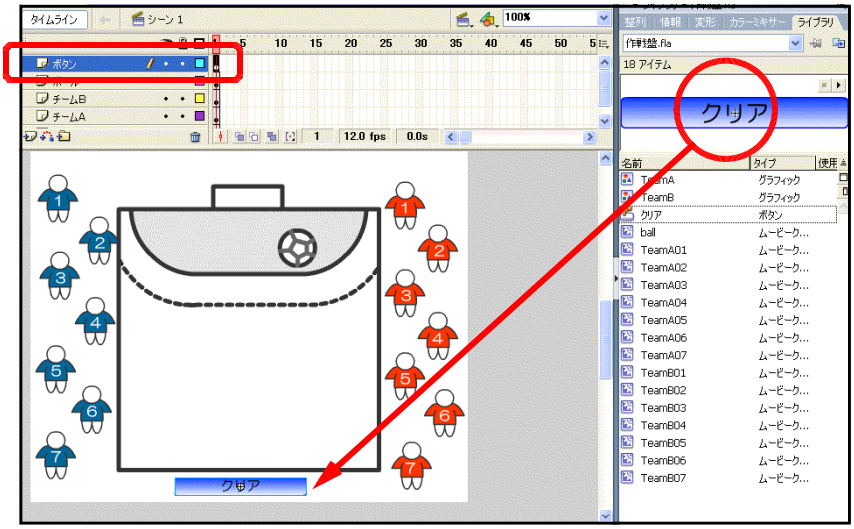
(h)　［テキストツール］を用いて、ボタン上に［クリア］の文字を入力します。



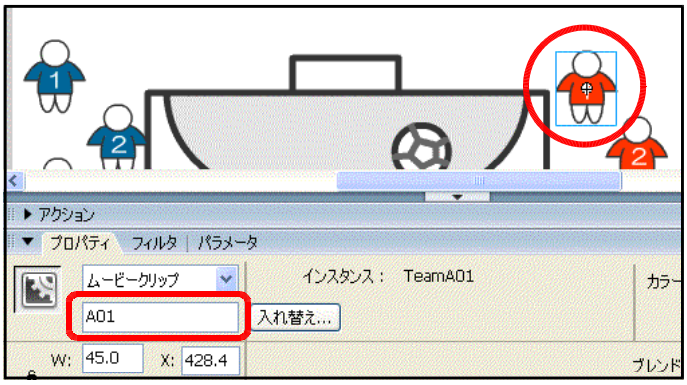
(i)　［シーン１］をクリックして、ドキュメントに戻ります。



(j)　［ボタン］レイヤーをクリックして、ライブラリパネルから［ボタン］シンボルをステージ上にドラッグして配置させます。

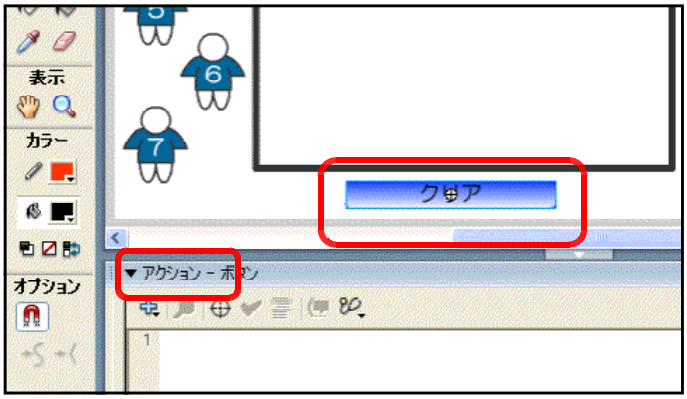


(k)　チームＡの背番号１の選手アイコンをクリックして選択します。プロパティインスペクタで名前を［**A01**］と入力します。これで、チームＡの背番号１のアイコンの名前を **A01** と付けたことになります。



(l)　同様にチームＡの背番号２を[**A02**]、・・・背番号７を［**A07**］、チームＢの背番号１を［**B01**］、・・・・、背番号７を［**B07**］、ボールを［**ball**］というように、それぞれ名前を付けます。

(m)　ステージ上のボタンをクリックして選択し、［アクション］をクリックして、アクションパネルを表示させます。

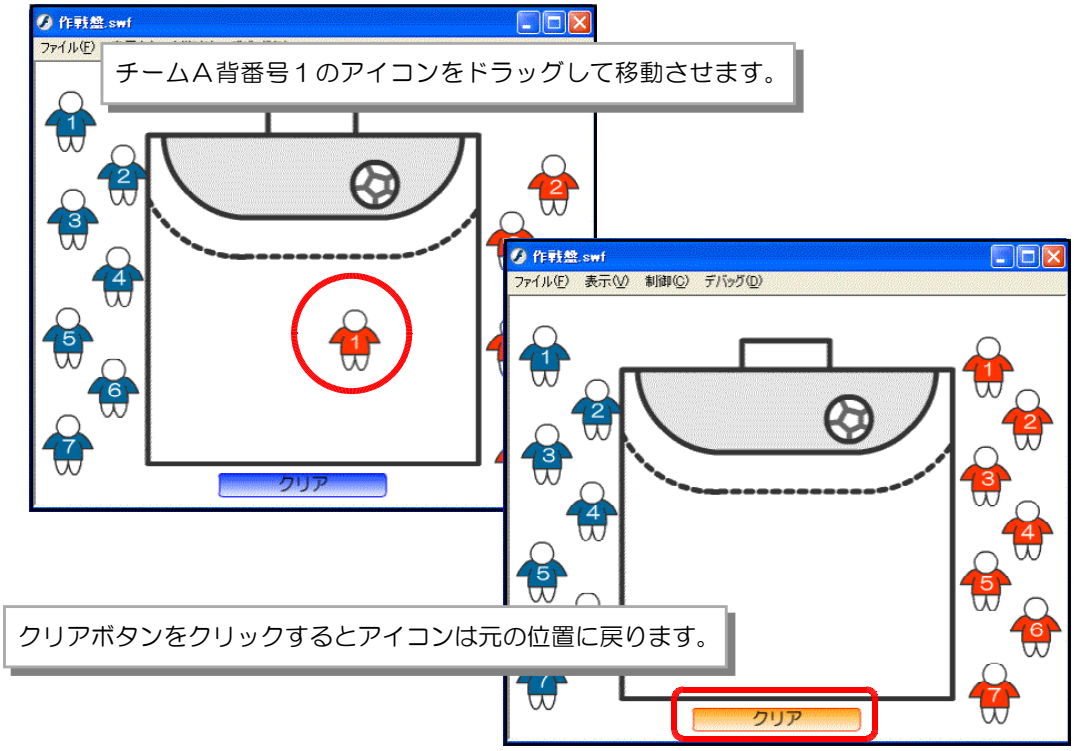


(n) 以下のスクリプトを入力します。

```
on (press) {
   _root.A01._x = 430;
   _root.A02._y = 65;
}
```

※ ボタンが押されらならば、
インスタンス a01 の X 座標は 630 へ、
Y 座標は 65 へ
移動させる。

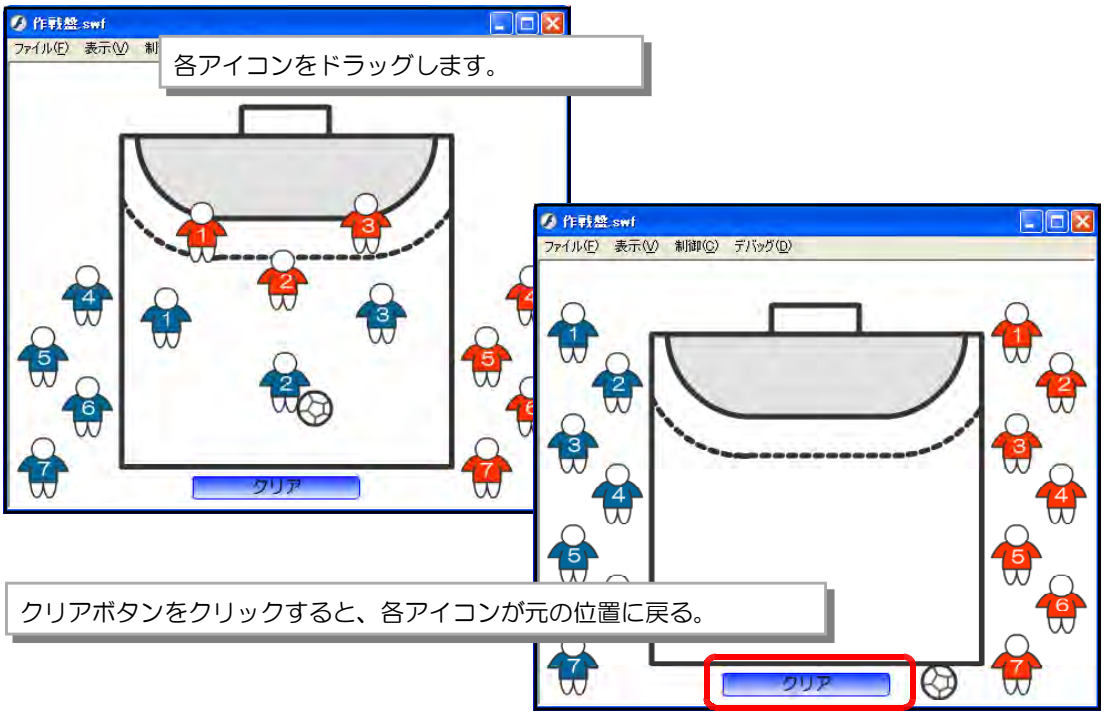
(o) プレビューで動作を確認してみましょう。［Ctrl］キーと［Enter］キーを同時に押して、プレビューさせます。



(p) 他のアイコンについても、元の位置に戻るためのスクリプトを以下のようにボタンへ追加します。

```
on (press) {
   _root.A01._x = 430;
   _root.A01._y = 65;
   _root.A02._x = 470;
   _root.A02._y = 110;
   _root.A03._x = 430;
   _root.A03._y = 165;
   _root.A04._x = 470;
   _root.A04._y = 210;
   _root.A05._x = 430;
   _root.A05._y = 265;
   _root.A06._x = 470;
   _root.A06._y = 310;
   _root.A07._x = 430;
   _root.A07._y = 365;
   _root.B01._x = 30;
   _root.B01._y = 65;
   _root.B02._x = 70;
   _root.B02._y = 110;
   _root.B03._x = 30;
   _root.B03._y = 165;
   _root.B04._x = 70;
   _root.B04._y = 210;
   _root.B05._x = 30;
   _root.B05._y = 265;
   _root.B06._x = 70;
   _root.B06._y = 310;
   _root.B07._x = 30;
   _root.B07._y = 365;
   _root.ball._x = 360;
   _root.ball._y = 380;
}
```

(q) 以上でスクリプトの設定が終わりましたので、動作確認をしてみます。［Ctrl］キーと［Enter］キーを同時に押して、プレビューさせます。

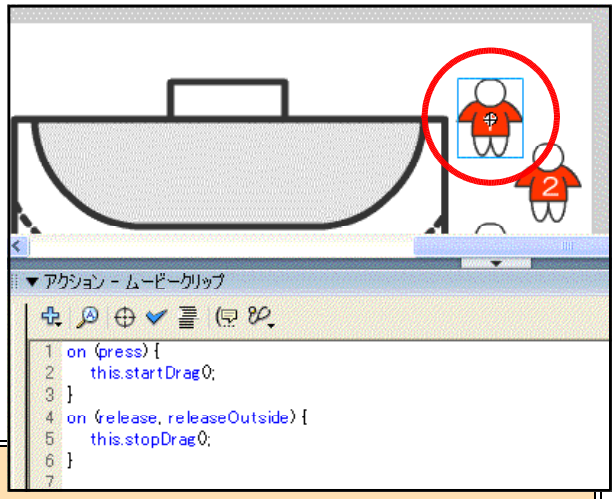


【追加機能３】　～　アイコンの重ね順を変えれれば！！

基本的に各インスタンスは後から作成したものが前面に配置されます。ここでは意図的に重ね順を変えてみましょう。具体的にはクリックしたものが最前面に配置されるようにしてみます。

(a) チームＡの背番号１のアイコンをクリックして選択します。

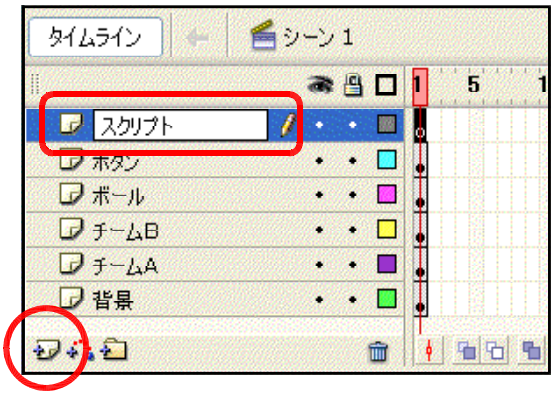
(b) すでにスクリプトが設定されていますが、on(press) {　～　}内に以下のスクリプトを入力します。




```
on (press) {
    this.swapDepths(_root.depth++);  ←――― 追加部分
    this.startDrag();
}
```

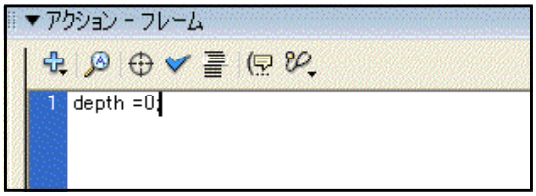
(c) 同様に他のアイコンについても、上記のスクリプトを追加します。

(d)　［レイヤーの追加］ボタンをクリックして、［ボタン］レイヤーの上に［スクリプト］レイヤーを追加します。

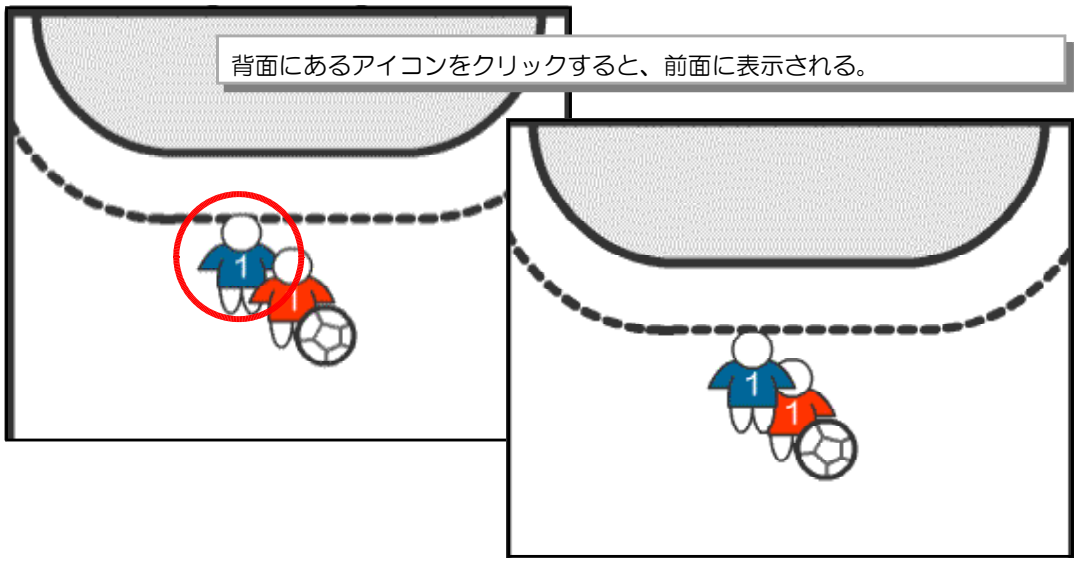


(e)　［アクション］パネルを表示させ、以下のスクリプトを入力します。

```
depth = 0;
```



(f)　以上で設定終了です。プレビューで動作確認をしてみましょう。［Ctrl］キーと［Enter］キーを同時に押します。

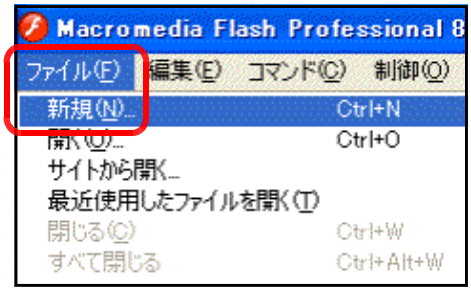


〈memo〉

1　外部ファイルからデータを読み込むムービーの作成

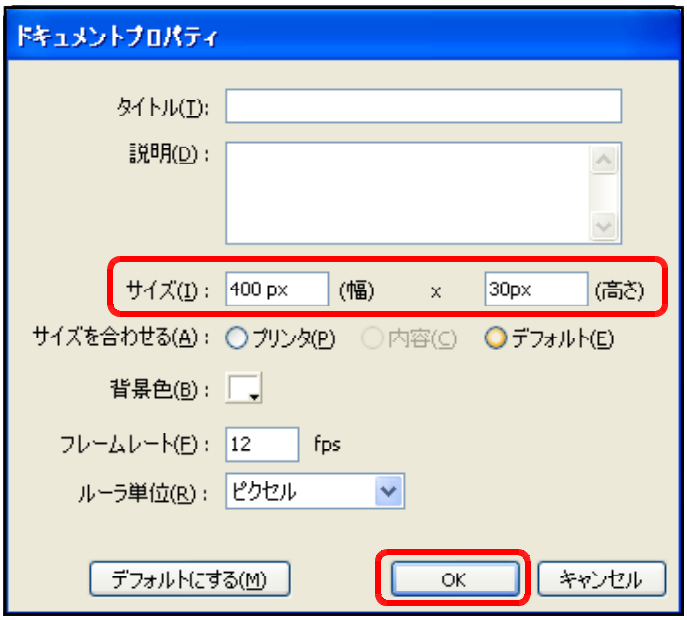
(1) テキストフィールドの作成

(a) **Flash8** を起動し、メニューから［ファイル］－［新規］をクリックして、新規ドキュメントを作成します。

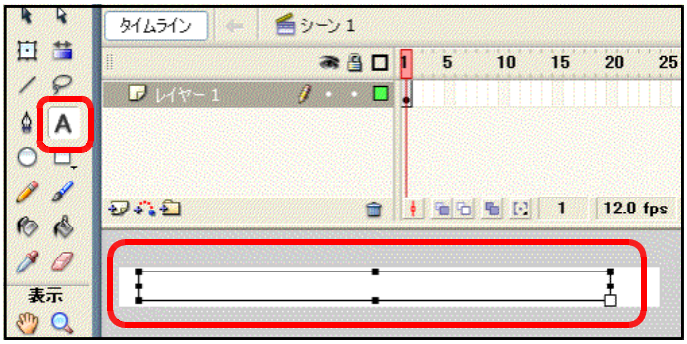


(b) ［プロパティ］をクリックして、プロパティパネルを表示させます。ドキュメントのサイズを変更するために［**550** × **400** ピクセル］ボタンをクリックします。

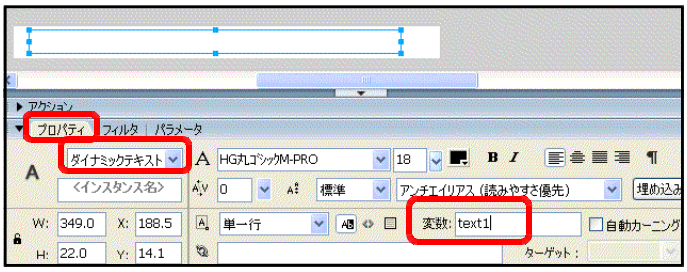
(c) ［ドキュメントプロパティ］ダイアログが表示されますので、サイズを［**400px**］（幅）×［**30px**］(高さ)に変更して［**OK**］ボタンをクリックします。



(d) ［テキストツール］ A をクリックして、ドキュメント上にドラッグして、任意のサイズでテキストフィールドを作成します。



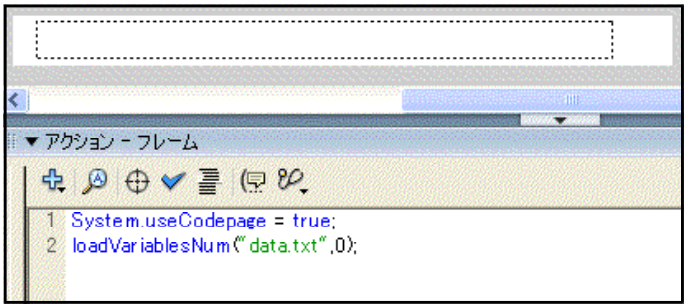
(c) ［プロパティ］をクリックしてプロパティパネルを表示させます。［ダイナミックテキスト］を選択して、変数を［**text1**］と入力して設定します。



(2) ActionScript の設定

(a) ドキュメントをクリックしてから［アクション］をクリックし、アクションパネルを表示させます。

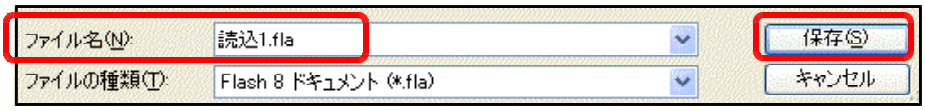
(b) 外部ファイルを読み込むためのスクリプトを入力します。



System.useCodepage = true;	文字コードを **Shift-JIS** にする
loadVariablesNum("data.txt",0);	**data.txt** から変数を読み込む

(c) メニューから［ファイル］－［名前を付けて保存］をクリックします。

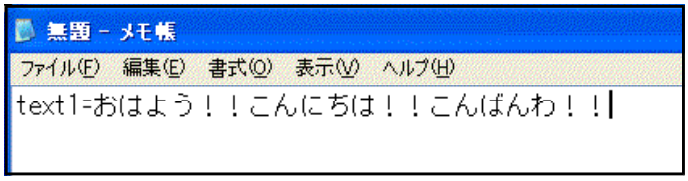
(d) ファイル名を［読込１**.fla**］と入力し、保存するディレクトリを指定して、［保存］ボタンをクリックします。



(3) 外部ファイルの作成

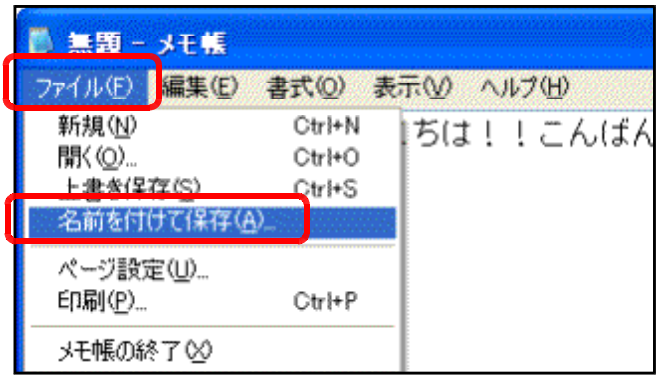
(a) デスクトップの［スタート］から［すべてのプログラム］－［アクセサリ］－［メモ帳］をクリックします。

(b) メモ帳に以下の文字列を入力します。



text1=おはよう！！こんにちは！こんばんわ！！　　（※）「**text1**=」は半角入力

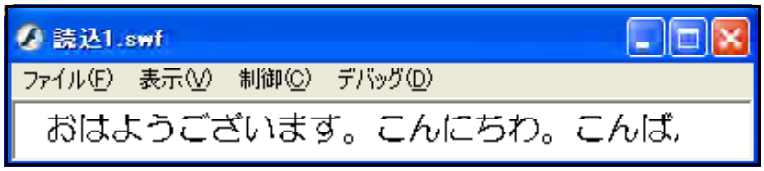
(c) メニューから［ファイル］－［名前を付けて保存］をクリックします。



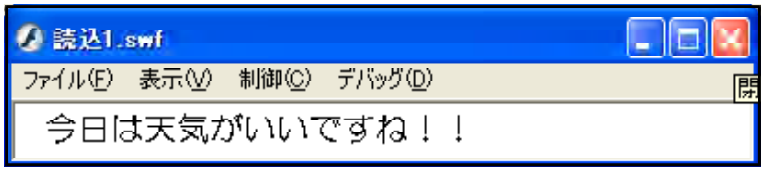
(d) ファイル名を［**data.txt**］として、保存ディレクトリを設定したら、［保存］ボタンをクリックして、データファイルを保存します。ただし、この **data.txt** ファイルは、先に作成した［読込１**.fla**］と同じ階層のディレクトリに保存してください。

(4) 動作確認

(a) ［読込１**.fla**］の編集画面から、［**Ctrl**］キーと［**Enter**］キーを同時に押して、プレビュー画面を表示させます。文字列が表示されるかを確認します。



テキストファイル（**data.txt**）の内容を変更して、再度プレビューすると反映されます。



２　応用：「簡易問題集」の作成

外部のテキストファイルに問題や解答を作成しておき、**swf** ファイルで読み込むことにより問題集を作成する。

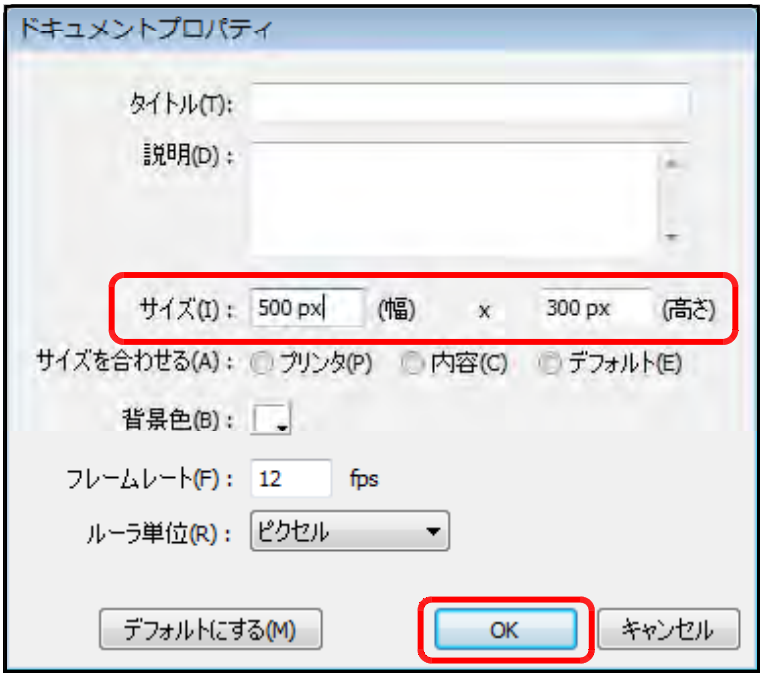
(1) 新規ドキュメントの作成

(a) ステージ中央の［**Flash** ドキュメント］をクリックして、新規ドキュメントを作成します。

(b) ［プロパティ］をクリックして、［プロパティ］パネルを表示させます。

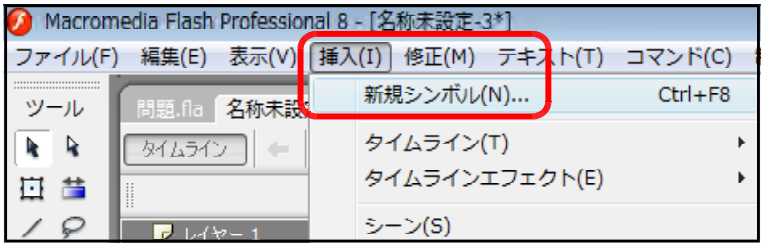
(c) サイズの［**550px** × **400px**］ボタンをクリックします。

(d) サイズを[**500px**](幅)、［**300px**］（高さ）に設定して、［**OK**］ボタンをクリックし、ドキュメントサイズを変更します。

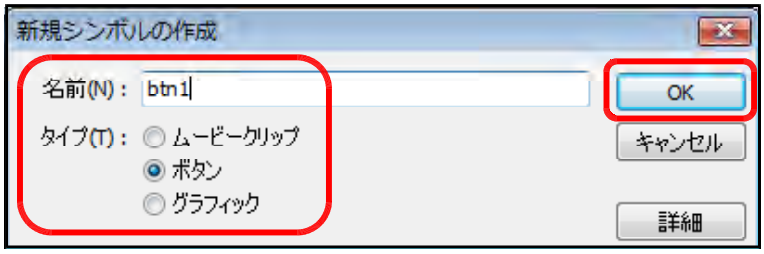


(2) ボタンシンボルの作成

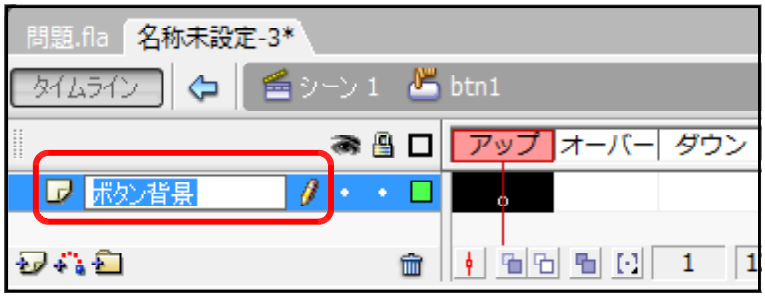
(a) メニューから［挿入］ー［新規シンボル］をクリックします。



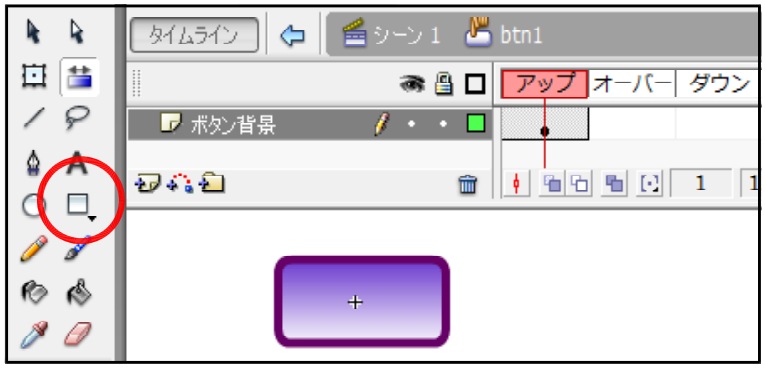
(b) タイプを［ボタン］、名前を［**btn1**］として、［**OK**］ボタンをクリックします。



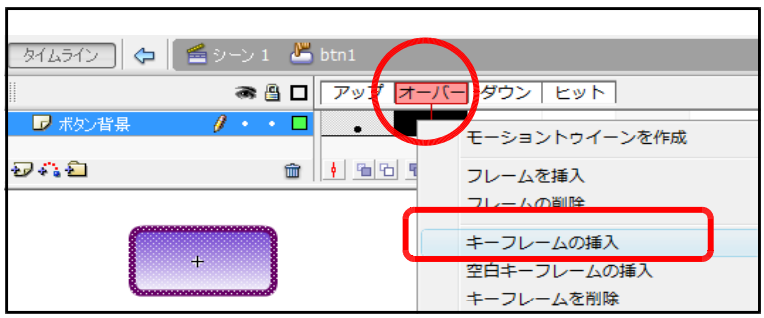
(c) レイヤー名［レイヤー１］をダブルクリックして、名称を［ボタン背景］に変更します。



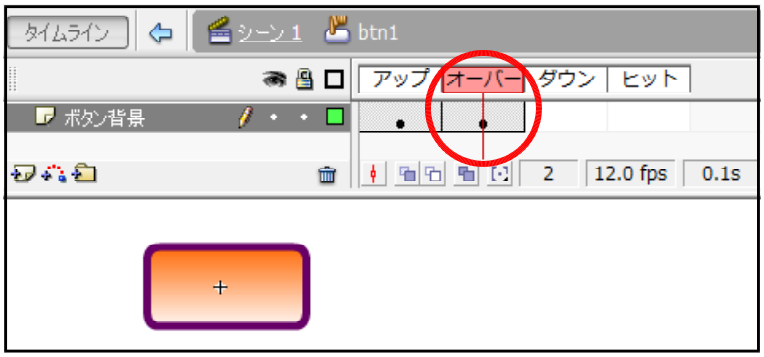
(d) 矩形ツール  を用いて、長方形をステージ上に描画します。適宜、線のカラーと塗りのカラーを設定します。なお、長方形のサイズは、［**100px**］（幅）×［**50px**］（高さ）とします。



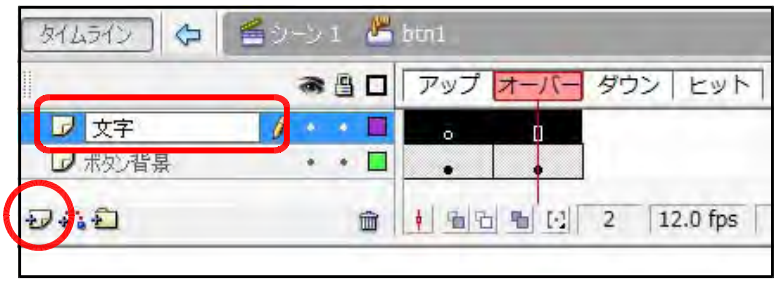
(e) ［ボタン背景］レイヤーの［オーバー］上で右クリックから、［キーフレームの挿入］をクリックします。



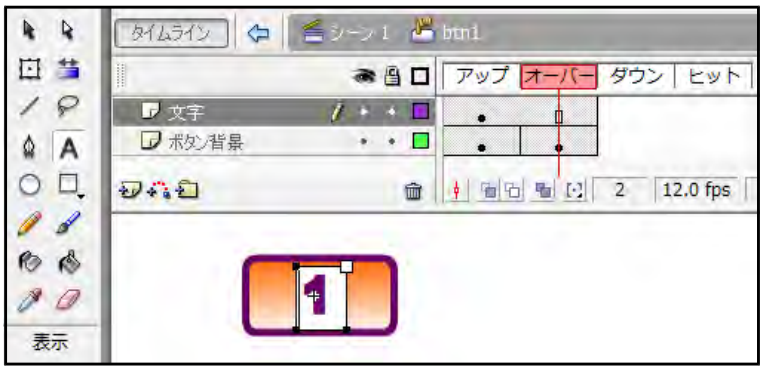
(f)　［ボタン背景］レイヤーの［オーバー］上でクリックし、適宜、線のカラーや塗りのカラーを変更します。このように設定をすることで、このボタン上にマウスのカーソルが重なった時に、色が変化するアクションが可能になります。



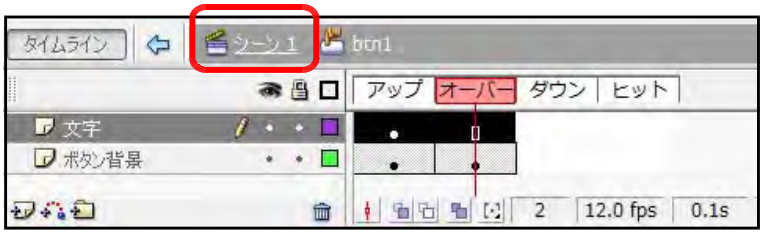
(g)　［レイヤーの追加］ボタン をクリックして、［文字］レイヤーを追加します。



(h)　テキストツールを用いて、［静止テキスト］として先に描画した長方形の上に、テキストボックスを配置して、［１］を入力します。



(i)　［シーン１］をクリックして、ドキュメントに戻ります。これで、ボタンのシンボルが完成しました。

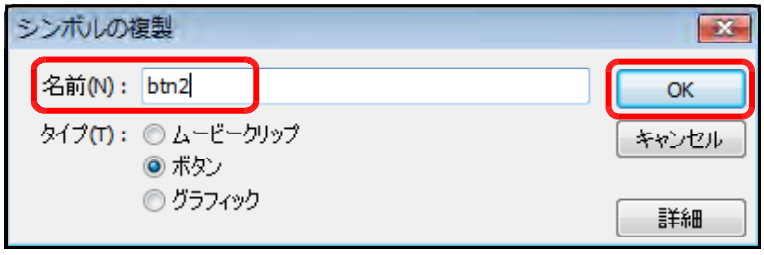


(3)　ボタンシンボルの複製

(a)　画面右下の［ライブラリ］パネルに、先に作成した［**btn1**］が登録されていることを確認します。もし、［ライブラリ］パネルが表示されていない場合には、メニューから［ウインドウ］－［ライブラリ］をクリックします。

(b)　［ライブラリ］パネル内の［**btn1**］シンボル上で右クリックから、［複製］をクリックします。

(c)　名前を［**btn2**］に変更して、［**OK**］ボタンをクリックします。



(d)　［ライブラリ］パネルに［**btn2**］シンボルが複製されました。［**btn2**］をダブルクリックします。

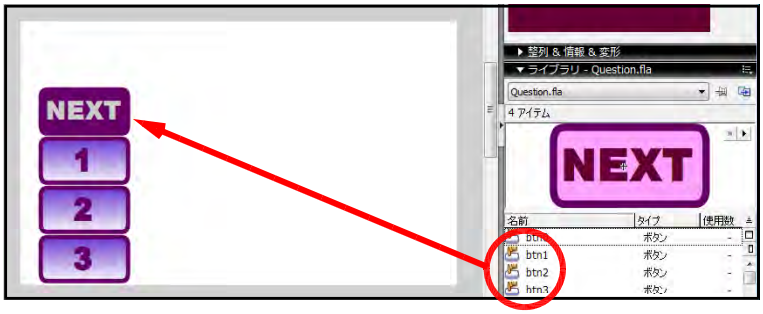
(e)　ステージ上の編集が［**btn2**］が対象となっていることを確認して、テキストツールで文字を「**2**」に変更します。

(f)　［シーン１］をクリックして、［**btn2**］シンボルの編集を終了します。(a)～(e)の操作を繰り返して、同様に［**btn3**］、［**btn0**］シンボルを作成します。

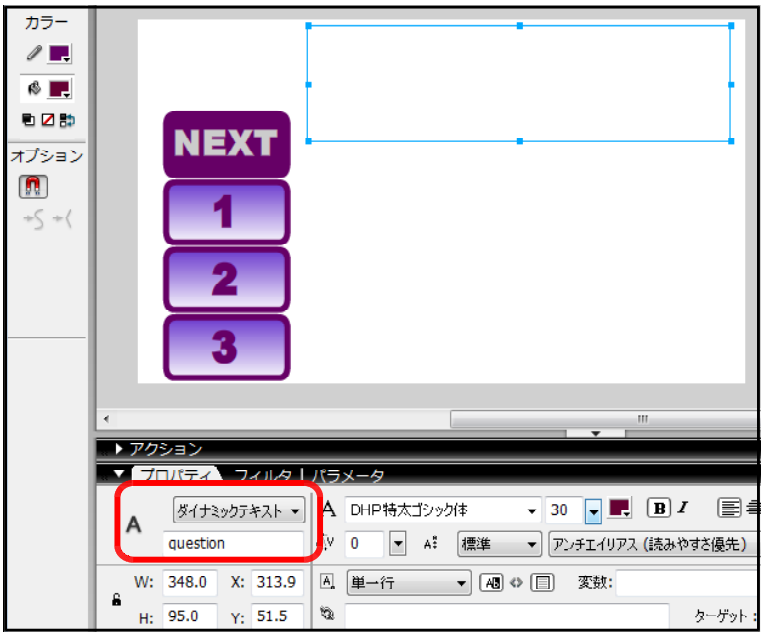
btn0　**btn1**　**btn2**　**btn3**

(4) ステージ上へのテキストインスタンスの配置

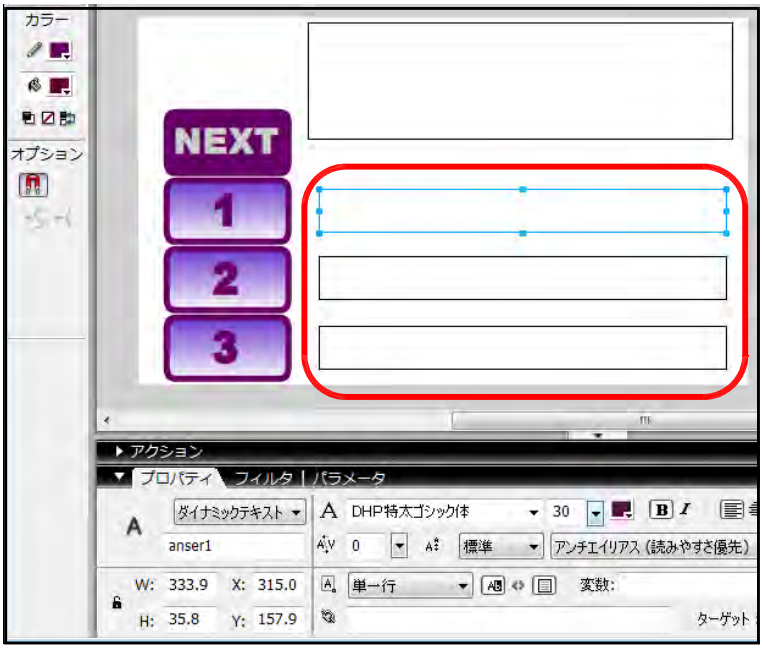
(a) ［ライブラリ］パネルからステージ上に［**btn0**］～［**btn3**］シンボルをドラッグして配置する。



(b) 次に、問題文の表示欄を設定します。テキストツールを用いて、ステージ上にテキストボックスをドラッグして作成します。［プロパティ］パネルで、種類を［ダイナミックテキスト］に変更して、インスタンス名を［**question**］と入力します。

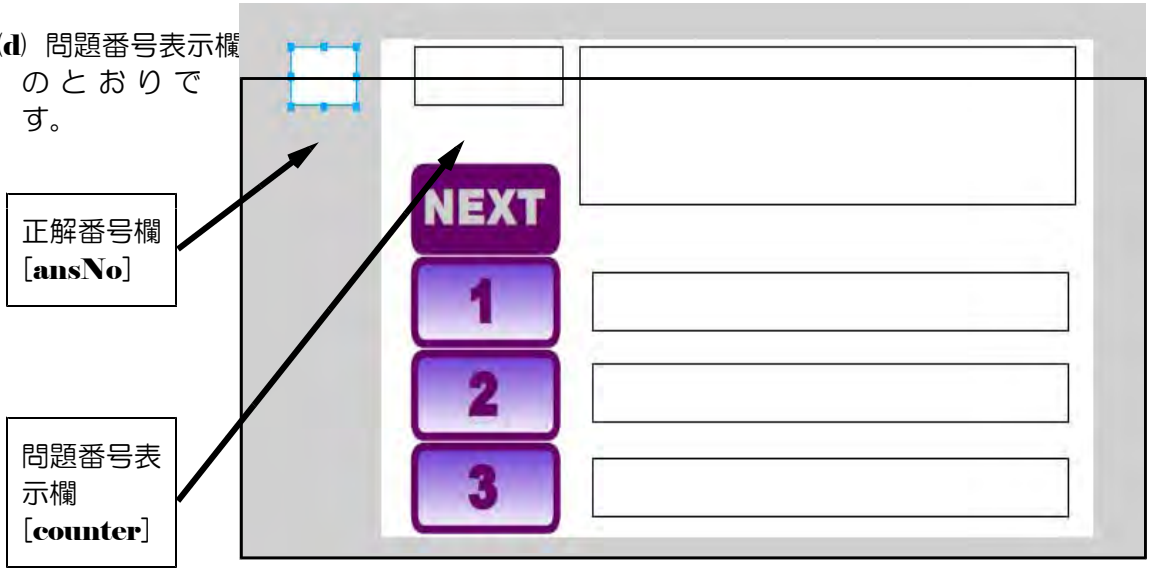


(c) 同様に、選択肢欄を以下のように設定します。



［インスタンス名］
- **anser1**
- **anser2**
- **anser3**

(d) 問題番号表示欄のとおりです。

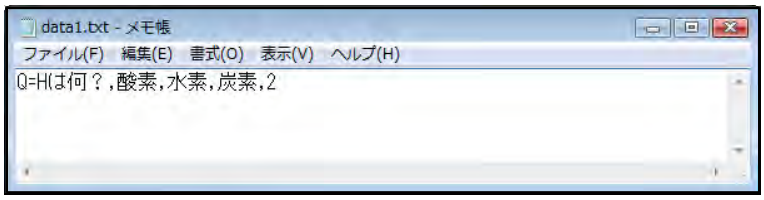


なお、正解番号欄は、ドキュメントキャンパス外に配置し、表示しないようにします。

(5) 問題用外部テキストファイルの作成

(a) メモ帳を開きます。

(b) 以下の文字列を入力します。



Q=H は何？,酸素,水素,炭素,**2**

上記の意味は、[**Q=**]：問題、「**,**（半角カンマ）」で区切って選択肢、最後の数字が正解番号というものです。

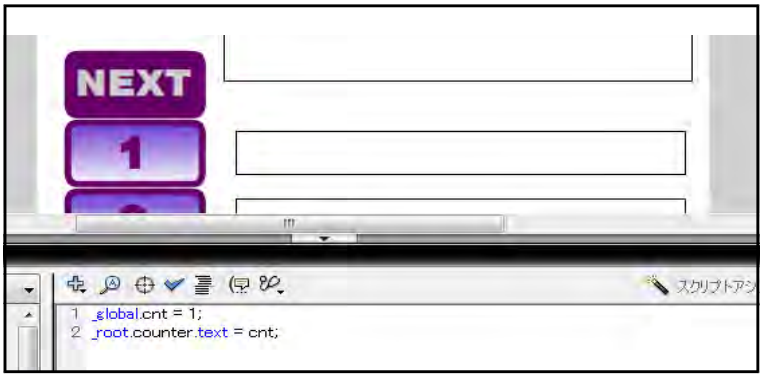
(c) **Flash** ドキュメントと同じフォルダ内に、[**data1.txt**] として保存します。

(d) 同様に、[**data2.txt**] ～ [**data5.txt**] としてテキストファイルを作成します。

【サンプル】
[**data1.txt**] 内容　**Q=H** は何？,酸素,水素,炭素,**2**
[**data2.txt**] 内容　**Q=Fe** は何？,ニッケル,銅,鉄,**3**
[**data3.txt**] 内容　**Q=S** は何？,塩素,硫黄,リン,**2**
[**data4.txt**] 内容　**Q=Ag** は何？,金,銀,銅,**2**
[**data5.txt**] 内容　**Q=Zn** は何？,鉛,銅,亜鉛,**3**

⑹ 外部ファイルの読み込み設定

⑻ ステージ上のキャンパスをクリックして、［アクション］パネル内に以下の２行のスクリプトを入力します。



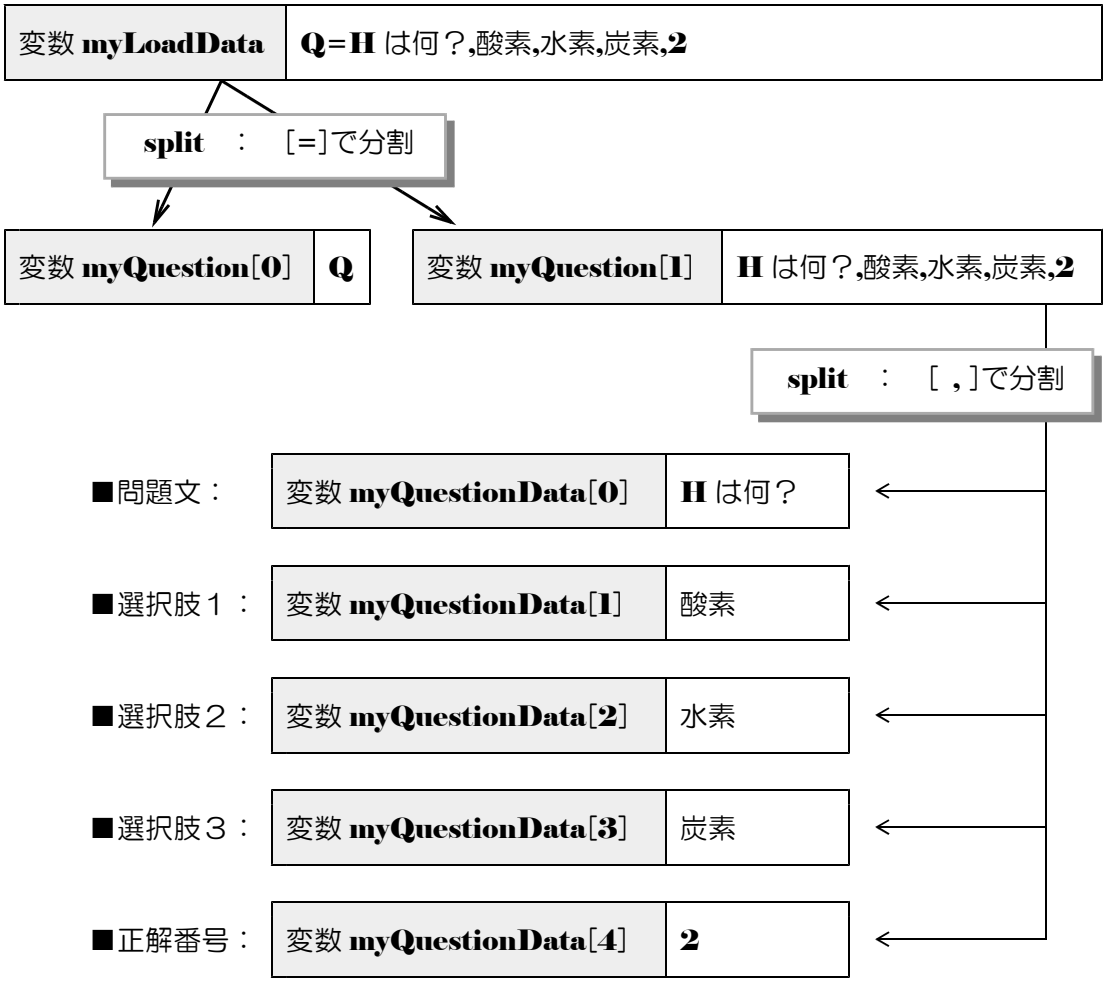
_global.cnt = 1;	変数 **cnt** をグローバル変数として、**1** を代入
_root.counter.text = cnt;	**counter** インスタンスに変数 **cnt** の値を代入

⑼ 次に、インスタンス[**btn0**]をクリックして、［アクション］パネル内に以下のスクリプトを入力します。

```
on (release) {                                          ボタンが押されたら
   System.useCodepage = true;                           文字コードの設定
   var myLoadData = new LoadVars();                     変数の定義
   strFileName = "data"+cnt+".txt";                     読み込みファイル名の作成
   myLoadData.load(strFileName);                        ファイルから読み込み
   myLoadData.onData = function(myString) {             読み込みが終了したら
      myQuestion = myString.split("=");                 [=]で文字列分割
      myQuestionData = myQuestion[1].split(",");        [,]で文字列分割
      _root.question.text = myQuestionData[0];          問題文を表示
      _root.anser1.text = myQuestionData[1];            選択肢１を表示
      _root.anser2.text = myQuestionData[2];            選択肢２を表示
      _root.anser3.text = myQuestionData[3];            選択肢３を表示
      _root.ansNo.text = myQuestionData[4];             正解番号を表示
      cnt = cnt+1;                                      カウンターを＋１
      _root.counter.text = cnt-1;                       問題番号を表示
      if (cnt>5) {                                      問題番号５を越えたら
         cnt = 1;                                       問題１に戻る
      }
   };
}
```

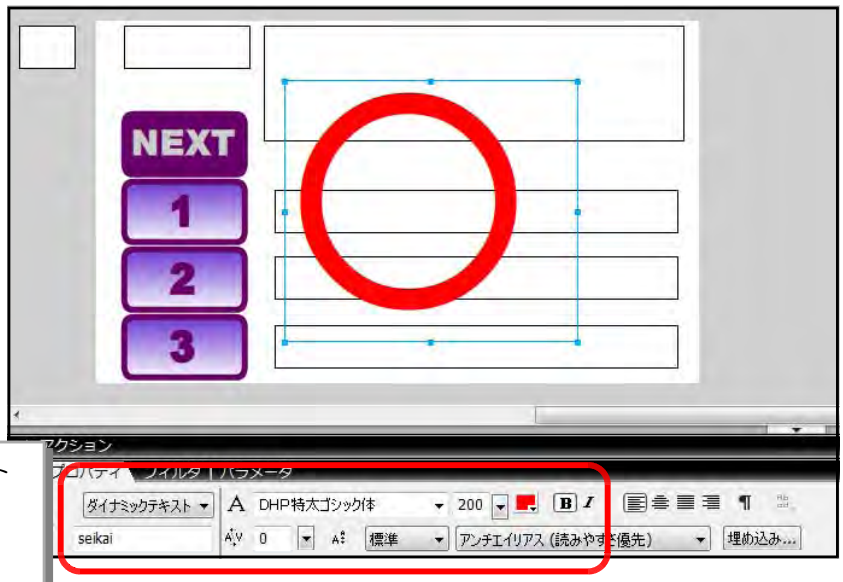
⑽ ［**ctrl**］＋［**enter**］キーを同時に押して、プレビューしてみましょう。［**NEXT**］ボタンをクリックするごとに、問題と選択肢が変更になることを確認しましょう。

（注）文字列を **split** 関数を用いて分割する方法



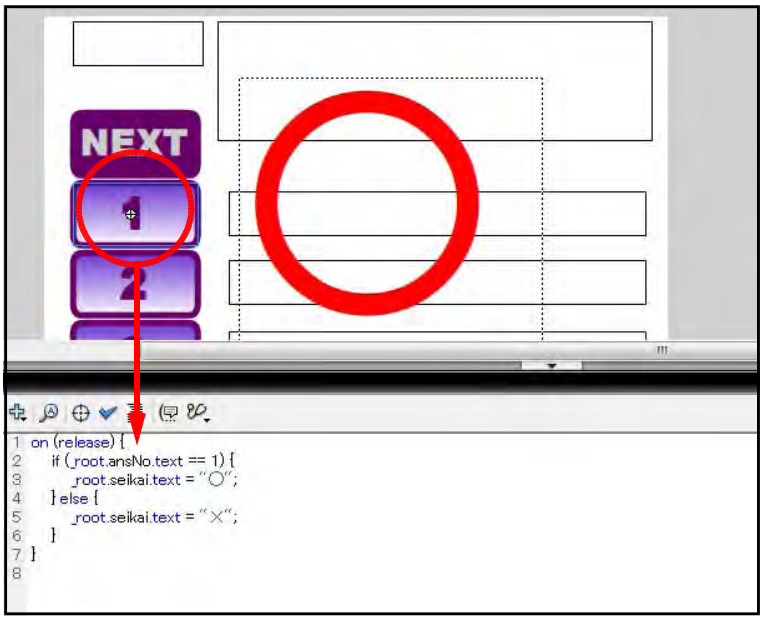
⑺ 正誤○×の表示設定

⑻ キャンパス上にテキストツールを用いて、文字列○を挿入します。［プロパティ］パネルで以下の設定を変更します。



■種類：ダイナミックテキスト
■インスタンス名：**seikai**
■文字サイズ：**200**
■文字色：赤

(b) インスタンス[btn1]をクリックして、[アクション]パネル内に以下のスクリプトを入力します。




```
on (release) {                          ボタンが押されたら
  if (_root.ansNo.text == 1) {          正解番号１が 1 の時
    _root.seikai.text = "○";            インスタンス seikai に○を表示
  } else {                              そうでない時
    _root.seikai.text = "×";            インスタンス seikai に×を表示
  }                                     する
}
```

(c) 同様に、[btn2]・[btn3]についても以下のスクリプトをそれぞれ入力します。

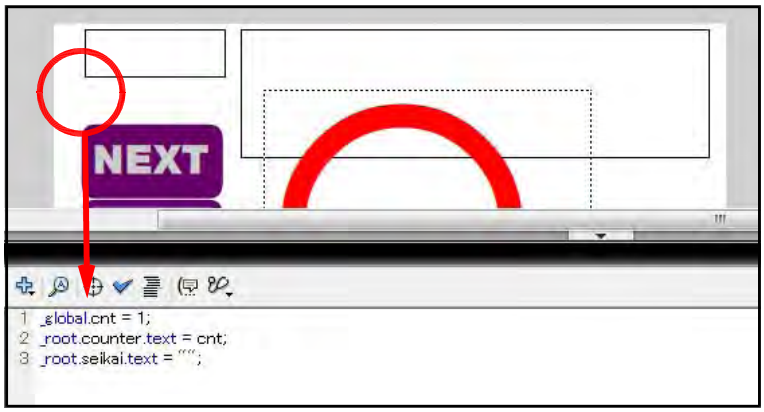
●[btn2]

```
on (release) {                          ボタンが押されたら
  if (_root.ansNo.text == 2) {          正解番号１が 2 の時
    _root.seikai.text = "○";            インスタンス seikai に○を表示
  } else {                              そうでない時
    _root.seikai.text = "×";            インスタンス seikai に×を表示
  }                                     する
}
```

●[btn3]

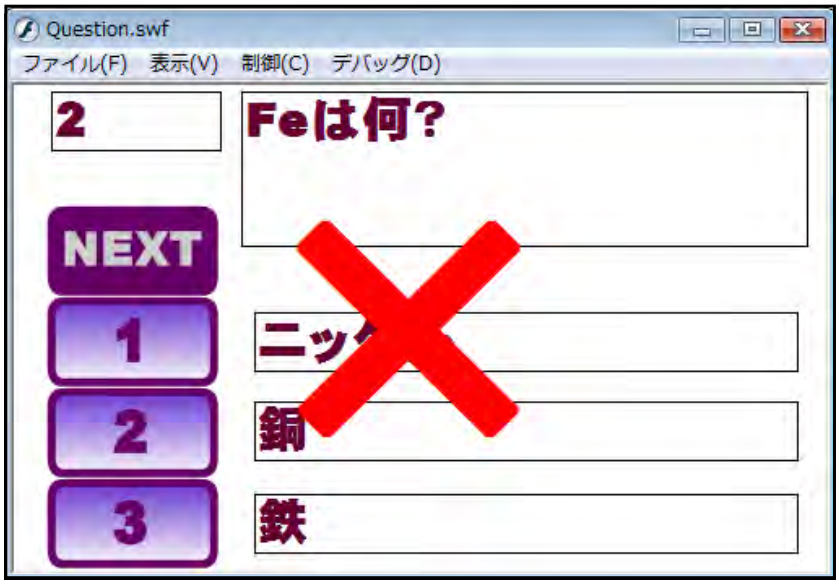
```
on (release) {                          ボタンが押されたら
  if (_root.ansNo.text == 3) {          正解番号１が 3 の時
    _root.seikai.text = "○";            インスタンス seikai に○を表示
  } else {                              そうでない時
    _root.seikai.text = "×";            インスタンス seikai に×を表示
  }                                     する
}
```

(d) ステージ上のキャンパスをクリックして、以下のスクリプトを［アクション］パネル内に入力します。



_root.seikai.text = "";	最初は、インスタンス **seikai** には文字を表示しない

(e) 以上で、設定は完了です。［**ctrl**］＋［**enter**］キーを同時に押して、プレビューしてみましょう。回答番号をクリックすることで、○と×の表示がされることを確認してみましょう。



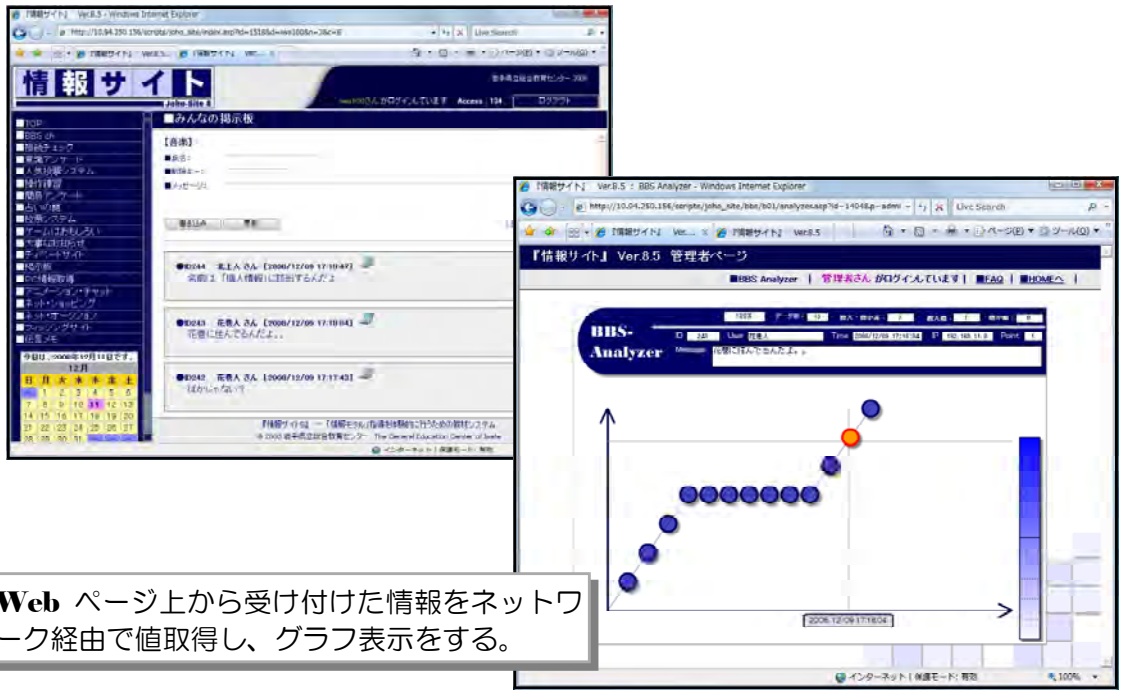
基本的な構成が完成しましたので、テキストファイル（**data1.txt** ～ **data5.txt**）の内容を変更することで問題が変更されます。後は、自由にカスタマイズしてみましょう。

〈**memo**〉

【応用編】

■ CGI と連携したグラフ表示

・・・ネットワーク上のデータを取得し、その値をもとにグラフ表示を行う。随時更新することでリアルタイムにアンケート結果の変異が確認できる。



Web ページ上から受け付けた情報をネットワーク経由で値取得し、グラフ表示をする。

■ CGI と連携したチャット

・・・インターネット上でよくみかける「チャット」のインターフェースを Flash で作成することにより、動きのあるチャットが再現できます。

ログインする際に自分のアイコンを選択し、チャット内で利用できる。発言したコメントが吹き出しとして表示される。

参考資料 ActionScript

on (press) { 〜 }	■ボタンまたはムービークリップのインスタンスに対して、マウスボタンをクリックしたら「〜」でしてされたイベントを実行する
例） **on (press) {** **x =1;** **}**	このムービークリップ上で、マウスボタンが押されたなら、 変数 **x** に **1** を代入しなさい。

on (release) { 〜 }	■ボタンまたはムービークリップのインスタンスに対して、マウスボタンが戻ったら「〜」で指定されたイベントを実行する
例） **on (release) {** **x = 1;** **}**	このムービークリップ上で、マウスボタンが戻ったら、 変数 **x** に **1** を代入しなさい。

on (release,releaseOutside) { 〜 }	■ボタンまはたムービークリップのインスタンスに対して、マウスボタンが戻るか、ドキュメント外へカーソルが移動したときに「〜」で指定されたイベントを実行する
例） **on (release,releaseOutside) {** **y = 2;** **}**	このムービークリップ上で、マウスボタンが戻るか、ドキュメント外にマウスカーソルが移動したら、変数 **y** に **2** を代入しなさい。

startDrag()	■ムービークリップのインスタンに対してドラッグを開始する
例）**this.startDrag()**	このムービークリップのドラッグを開始据する

stopDrag()	■ムービークリップのインスタンスに対してドラッグを終了する
例）**this.stopDrag()**	このムービークリップのドラッグを終了する。

_x	■ムービークリップのＸ座標を取得する
例）**this._x = 600**	このムービークリップの **x** 座標を **600** に設定する

_y　　■ムービークリップの y 座標を取得する
例　**this._y = 400**　　このムービークリップの y 座標を 400 に設定する

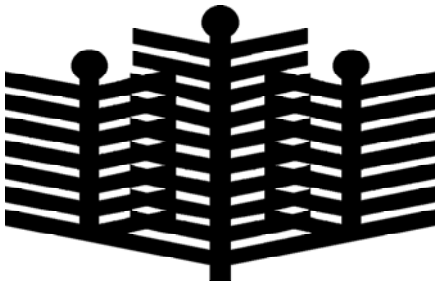
swapDepths()　　■ムービークリップの深度を変更する

useCodepage　　■文字コードを変更する
例）**System.useCodepage = true**　　このプロパティを **true** に設定すると、**Flash layer** を実行するオペレーティングシステムの通常のコードページを使用して外部テキスファイルが解釈します。

loadVariablesNum()　　■外部ファイルの値を読み込み、指定した変数に代入する
例）**loadVariablesNum("data.txt")**　　**data.txt** ファイルからデータを読み込み、指定した変数に代入する

getURL()　　■指定した **URL** からページを読み込む
例）**getURL("http://www.yahoo.co.jp/")**　　**Yahoo!**ページを表示させる

〈**memo**〉



岩手県立総合教育センター
情報教育担当
平成21年１月８日発行