センサを利用したFlashゲームの製作手引書

課題１．Ｆｌａｓｈの起動と保存

課題２．画面の設定

課題３．ムービークリップの移動とイベント処理

課題４．テキストフィールドの配置及び表示設定

課題５．スプライトの表示設定及びｔｒａｃｅ文活用

課題６．マウスによるムービークリップ制御

課題７．Ｗｉｉリモコンの学習及び設定

課題８．無線機器Ｂｌｕｅｔｏｏｔｈの学習及び設定

課題９．ＷｉｉＦｌａｓｈの学習及び設定

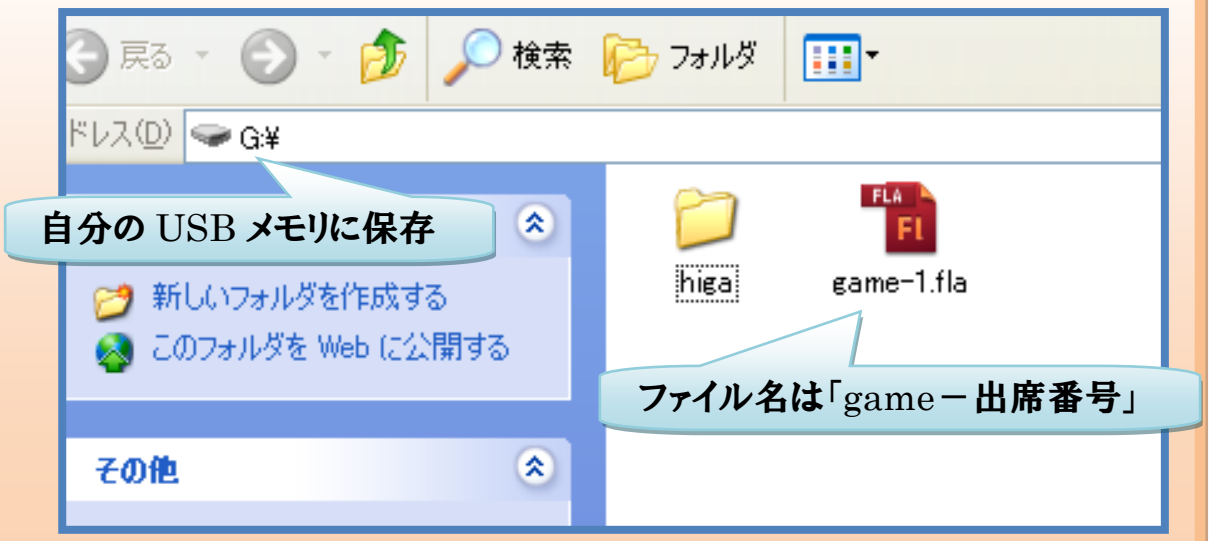
課題１０．赤外線センサバーの製作

課題１１．赤外線センサによるムービークリップ制御

課題１２．加速度センサによるムービークリップの制御

課題１３．ActionScript ライブラリの活用

課題1．Flashを起動させ、新規ファイルを自分のUSBメモリに保存しましょう。



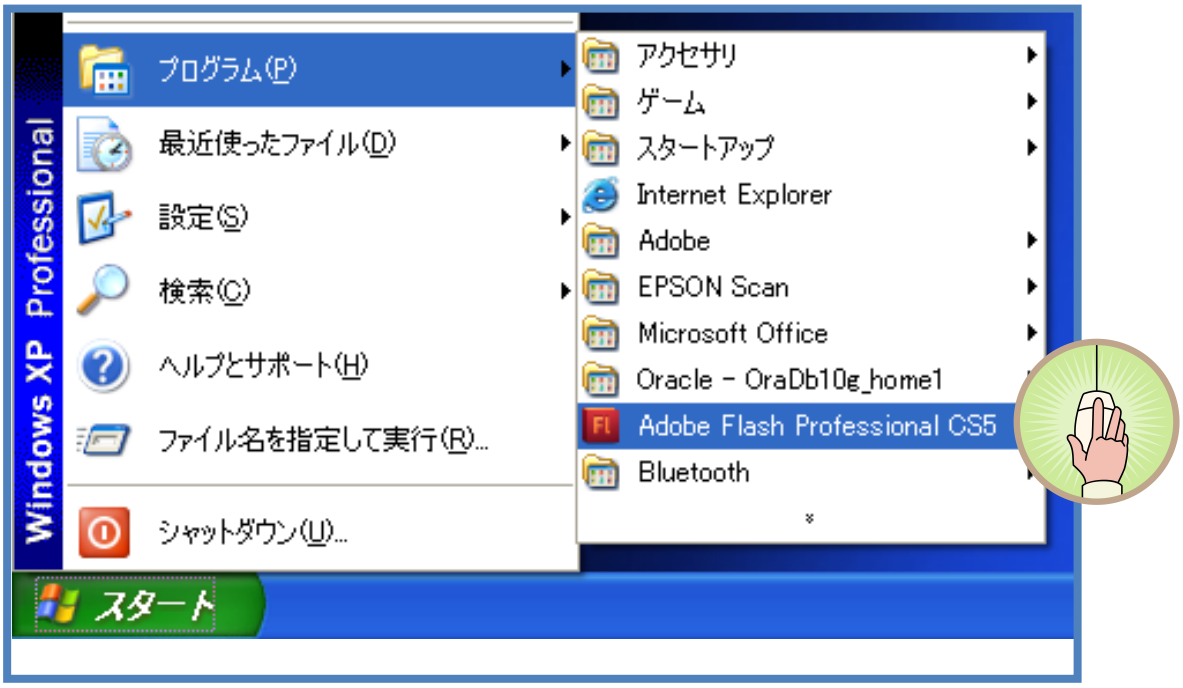
本課題の目標.

1．Flash **が起動できる。**

2．**ファイルの保存ができる。**

今回使用するソフトは「Adobe Flash Professional CS5」になります。
これまで使用してきた「Flash8」とは記述の仕方などが変わっており，注意しながら学習を進めていきましょう。

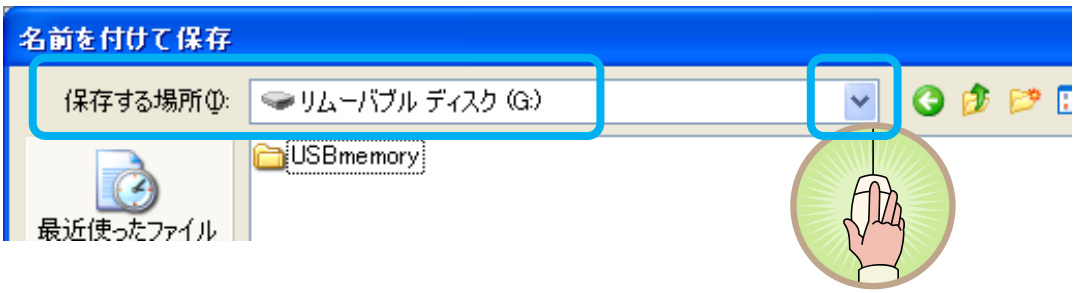
STEP① Adobe Flash Professional CS5 **を起動します。**



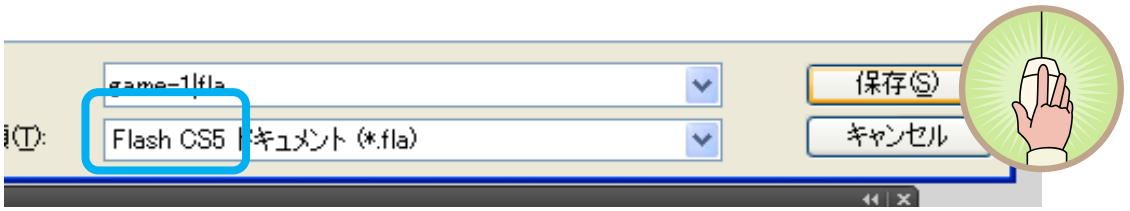
STEP② **新規作成**—Actionscript3.0 **を選択します。**

STEP③ **新規ファイルに名前をつけます。**

STEP④ **保存する場所を「リムーバブル　ディスク」にします。**

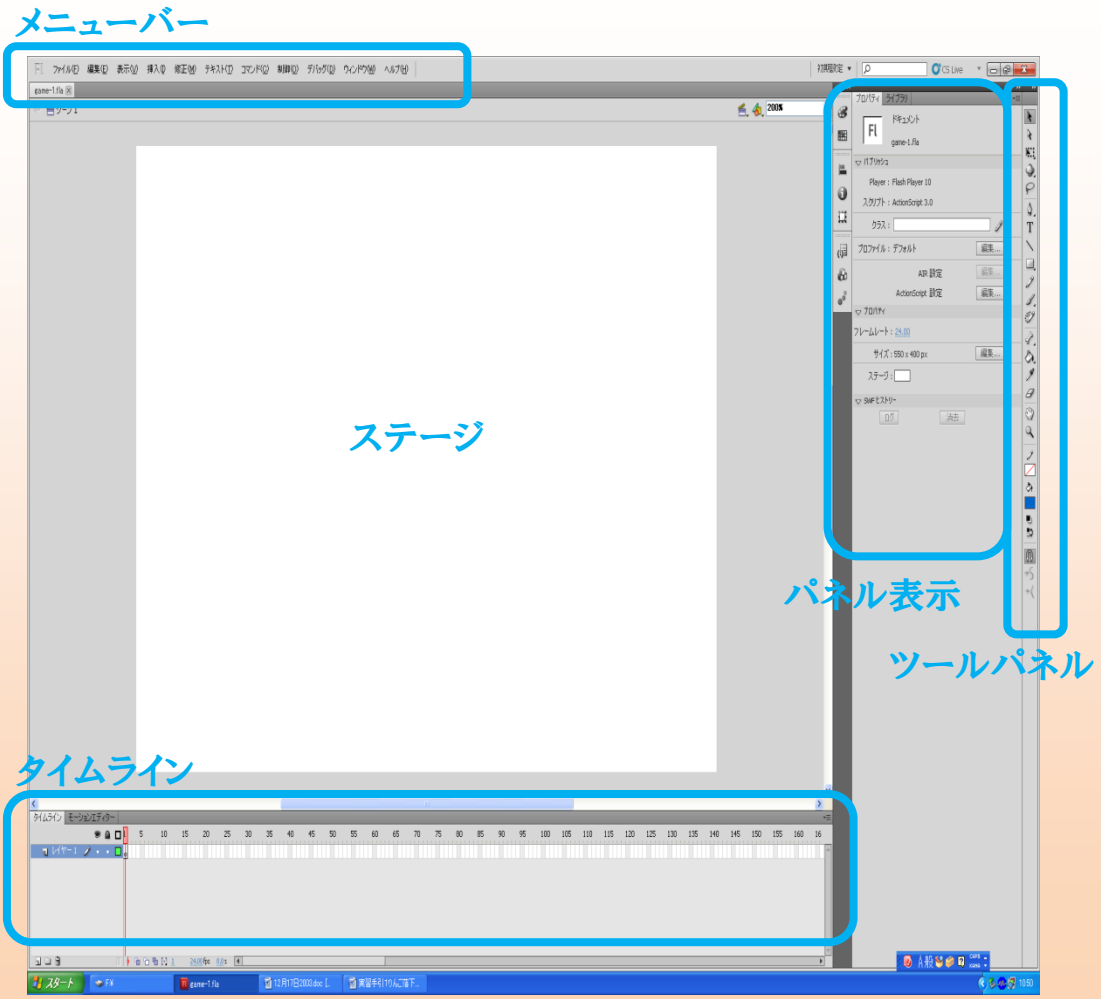


ファイル名を「game-〇〇（**出席番号**）.fla**」と書いて、保存します。**



＊＊　課題１の完成です。これで USB **メモリに新規ファイル「**game-1.fla**」ができました。次は作業画面の設定をします。＊＊**

課題2．画面の設定をしましょう。



本時の目標.

1．作業画面の設定ができる。

2．各名称とその役割が理解できる。

STEP①　**作業画面の主な各名称は次のとおりです。**

メニューバー：さまざまな操作を行うコマンドが用途別に分類されています。

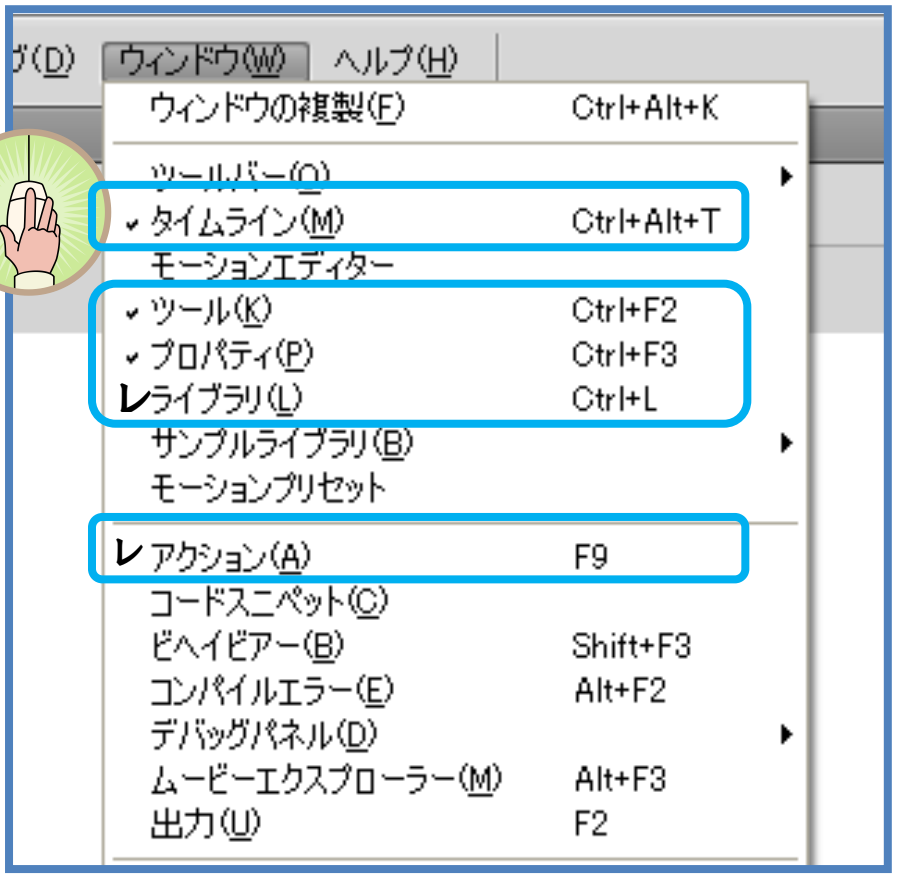
ステージ：ムービーとして表示する絵を描きます。

タイムライン：ムービーの進行管理を行います。

パネル表示：プロパティパネルやライブラリパネルなどを表示します。

ツールパネル：絵に関する編集ツールがあります。

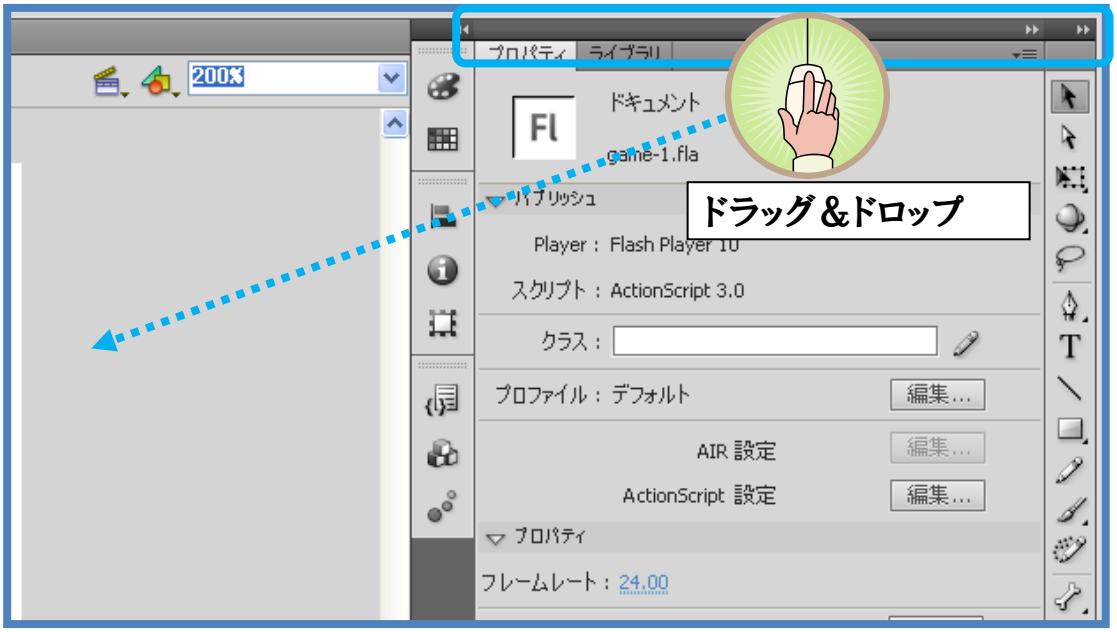
STEP②　**作業しやすいように必要なウィンドウパネルにチェックマークを付けて，開いておきます。**



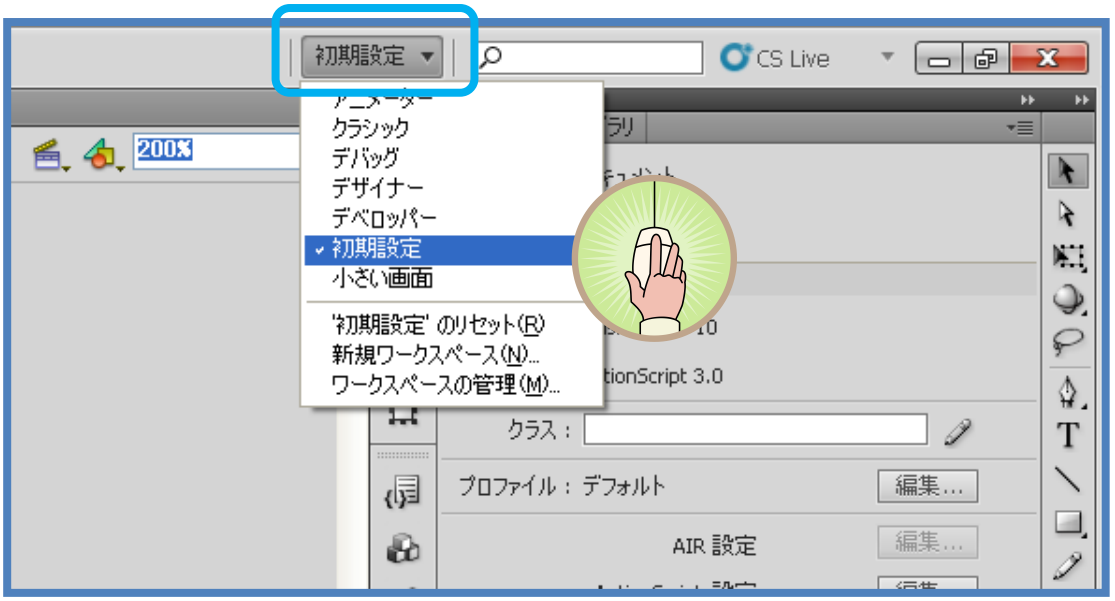
また、パネルを隠すとき及び非表示するときは、パネル右上のボタンを選択します。

STEP③ **パネルの位置を調整します。**

パネルのタブをドラッグ＆ドロップして位置へ移動します。



STEP④ **パネルの配置を初期状態に戻すには、画面の右上にある初期設定を選択します。**
本学習において，パネルの配置は「初期設定」の状態で作業を進めていきます。



課題3. りんごをステージ上部のランダムな位置から落下させましょう。

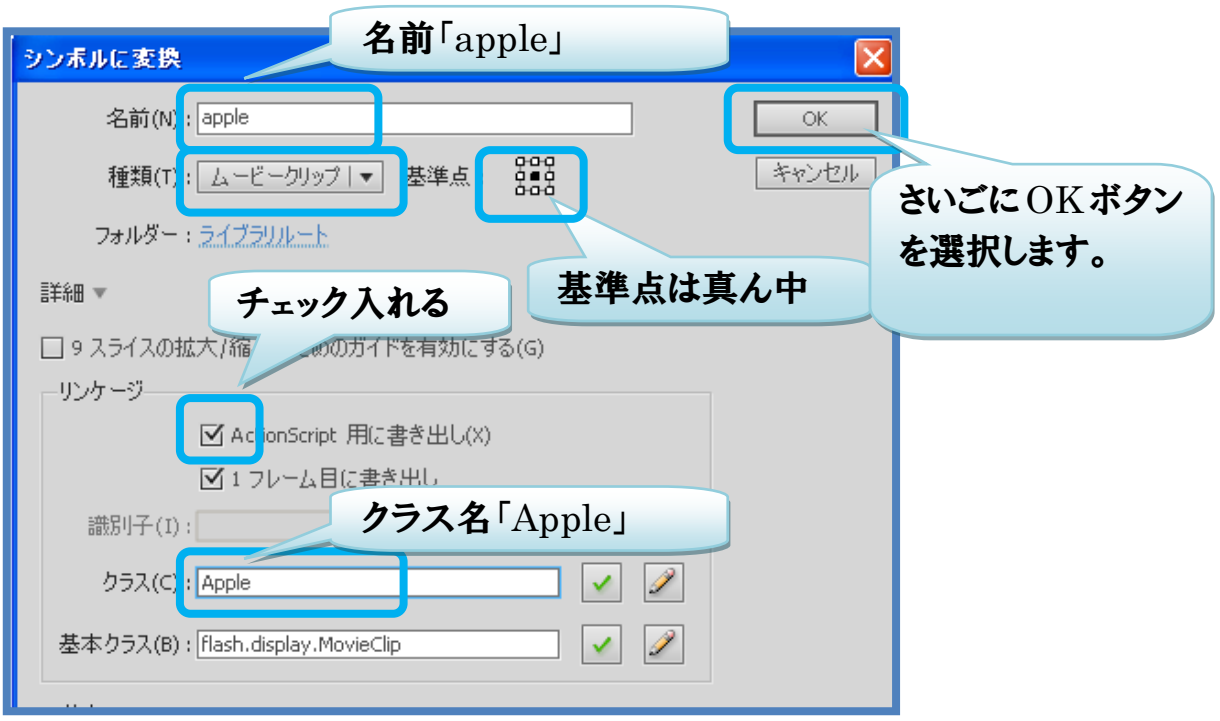
本時の目標

1. ムービークリップをライブラリに登録できる。

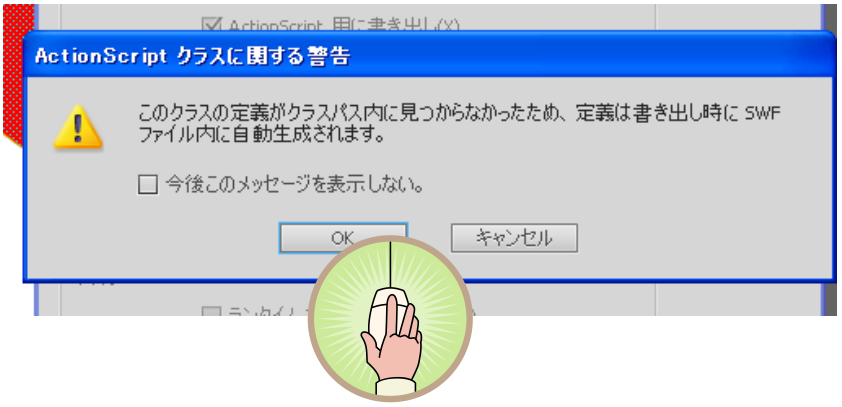
2. ムービークリップの移動を Actionscript によりプログラムできる。

STEP① **りんごのイラストを描き、シンボルに変換します。**

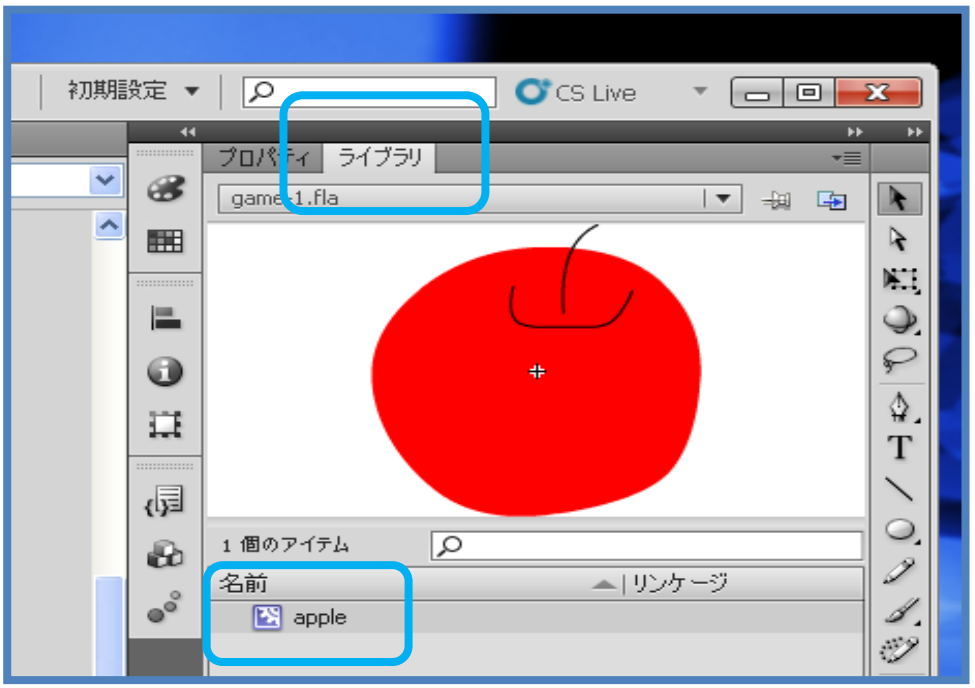
フリーハンドでりんごを描き、メニューバー「修正」の「シンボルに変換」を選択します。下記部分を選択します。



続いて，クラスパスに関する警告が表示されますので OK **ボタンを選択します。**

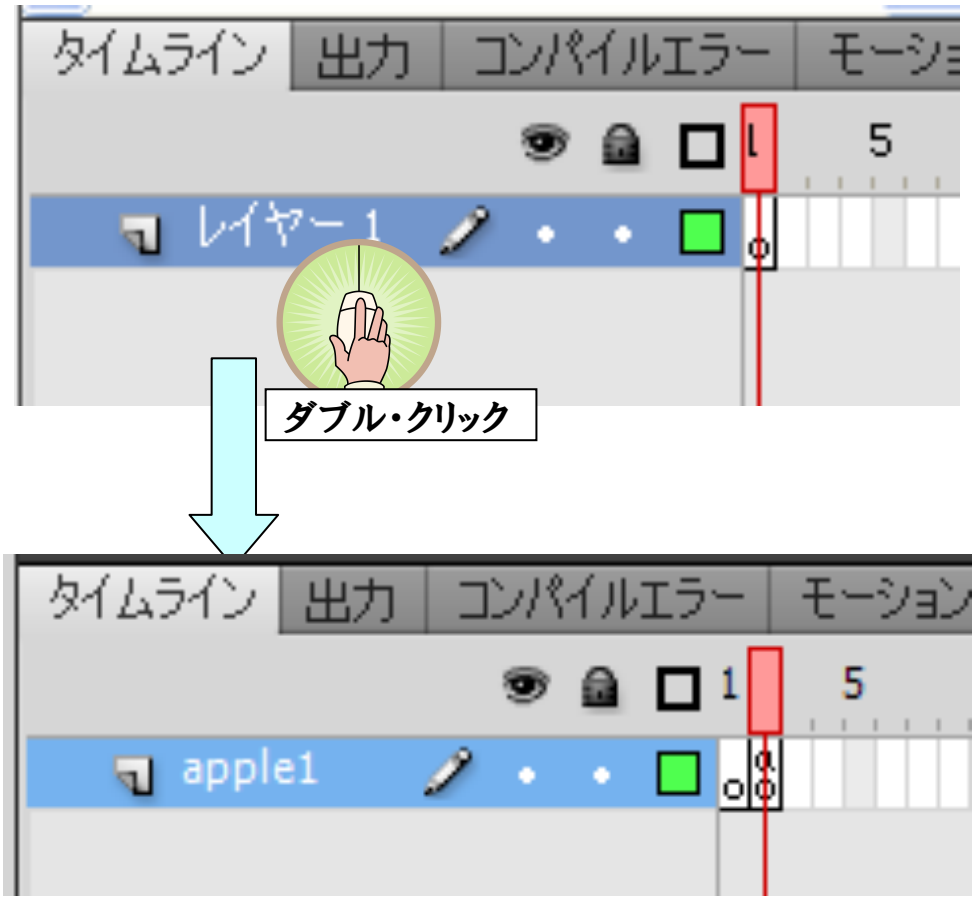


これでライブラリに、ムービークリップ「apple**」が登録されました。**



次に，ムービークリップが下方向に移動するように，Actionscript **の記述を行います。**

STEP② **タイムラインパネルのレイヤー1をレイヤー名「apple1」にします。**
レイヤー1上でマウスをダブルクリックして名称を「apple1」とします。



STEP③ **第2フレームにおいて、アクションパネルを開き、りんごが落下する記述を行います。**

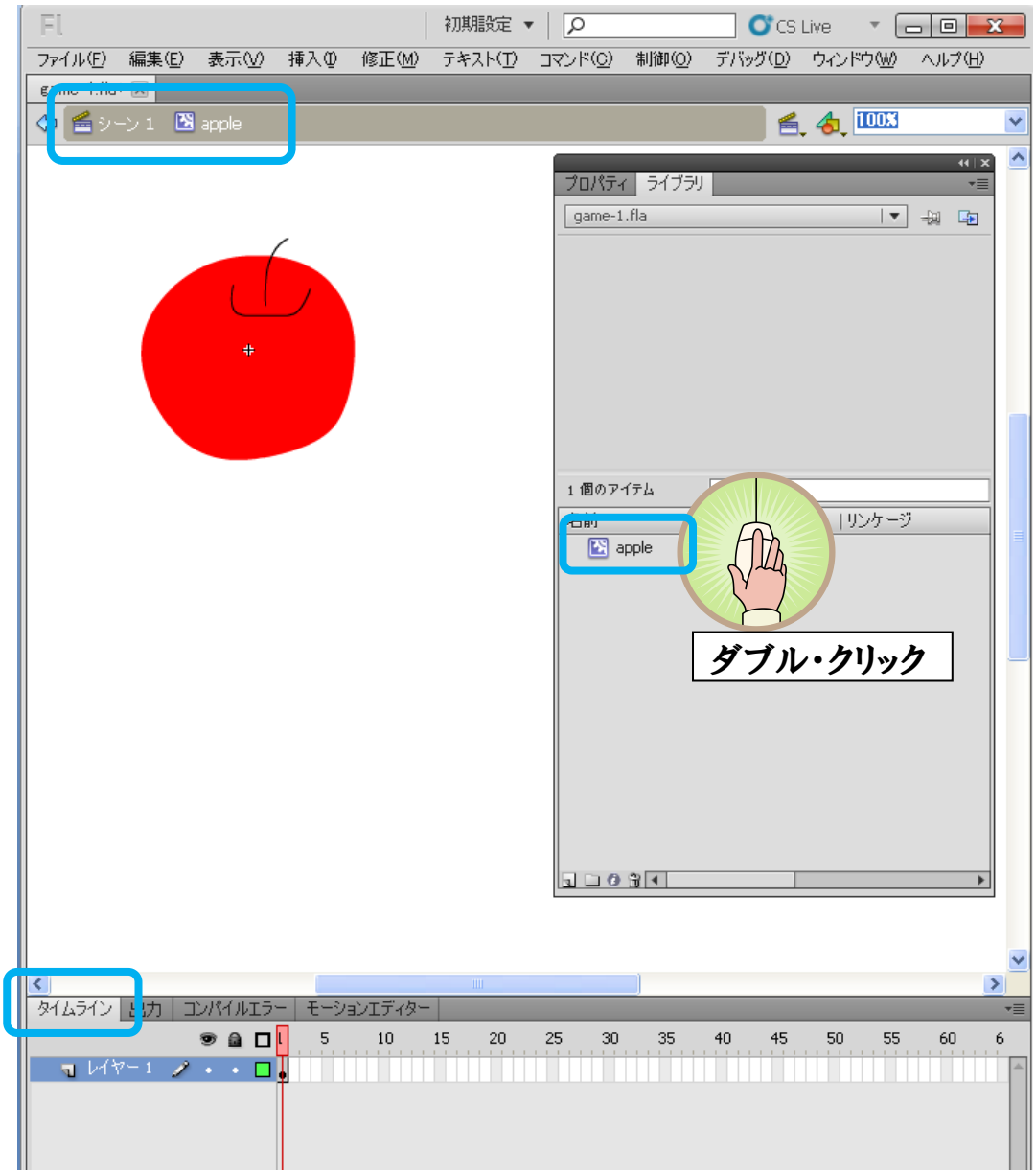
```
1  var apple:Apple=new Apple();
2  var a:int=Math.floor(Math.random()*(stage.width-50-50+1))+50;
3  apple.x=a;
4  apple.y=-100;
5  addChild(apple);
6
7
```

1行目： 変数 apple **の宣言。**

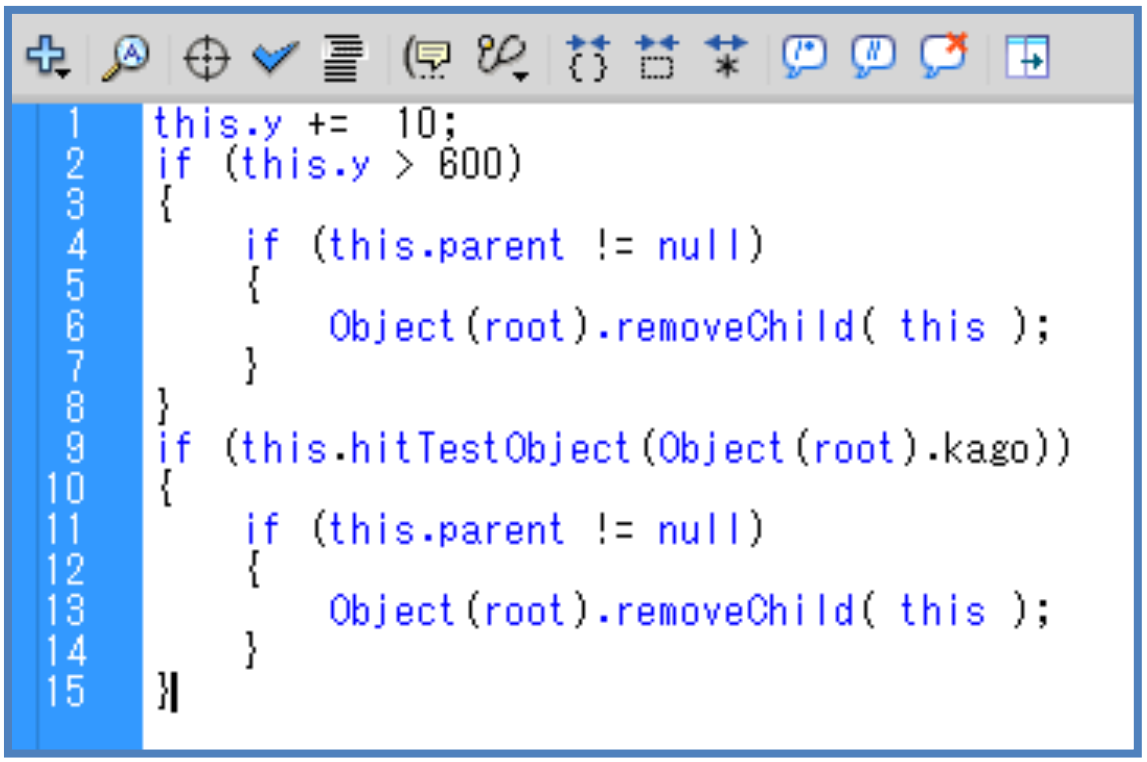
2行目：　ランダム関数により、apple のx座標 a を求める。
3行目：　apple 発生時のx座標に変数 a の値に出現。
4行目：　apple 発生時のy座標は-100 に出現。
5行目：　画面に表示。

これでりんごがステージに表示される状態になりました。
STEP④以降ではりんごが落下する（下向きに移動する）プログラム、かごに入ったら消えるプログラム、ステージ下を通過したら消えるプログラムを加えていきます。

STEP④　シンボル「apple」内のタイムラインを表示させます。
ライブラリパネルの「apple」のアイコンをダブルクリックすると、「apple」のタイムラインが表示されます。



STEP⑤ タイムラインの第1フレームを選択し、アクションパネルを開きます。りんごのy座標が10ずつ下移動するプログラム、ステージ外に出ると削除されるプログラムを記述します。

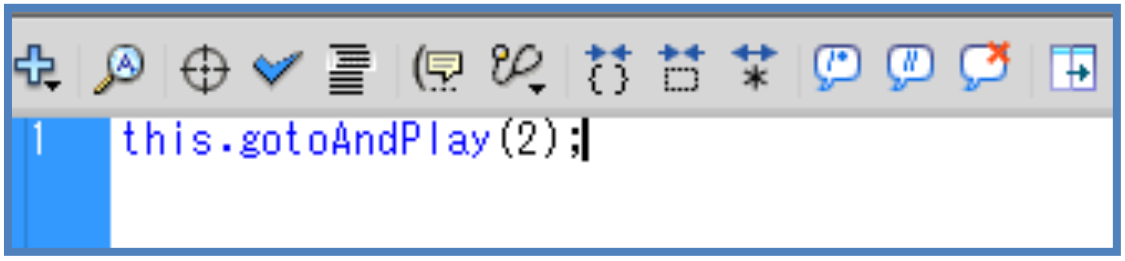


```
1  this.y +=  10;
2  if (this.y > 600)
3  {
4      if (this.parent != null)
5      {
6          Object(root).removeChild( this );
7      }
8  }
9  if (this.hitTestObject(Object(root).kago))
10 {
11     if (this.parent != null)
12     {
13         Object(root).removeChild( this );
14     }
15 }
```

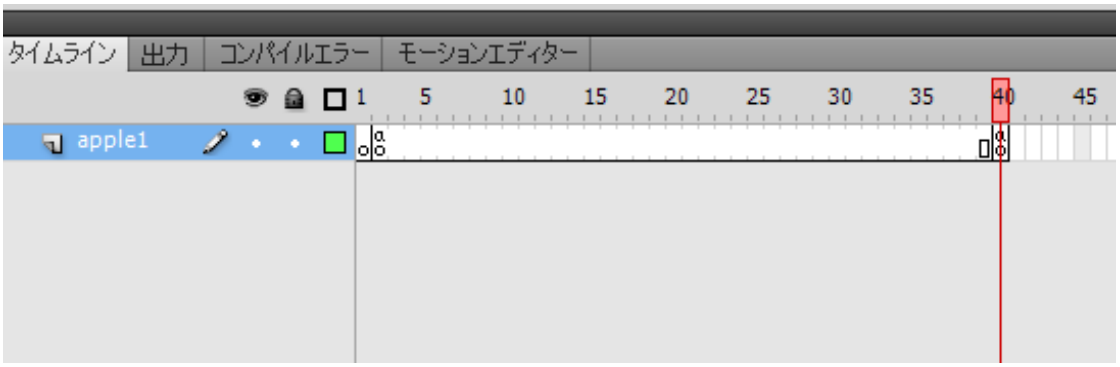
1行目： apple のy座標の移動値。
2行目～8行目： if 関数を用いている。y座標の値が600を超すと削除される。
9行目～15行目：kagoと接触したら削除される。

STEP⑥ 次に第39フレームを選択し、右クリックのメニューから「フレームを挿入」を選択します。

STEP⑦ 第40フレームを選択し、右クリックのメニューから「キーフレームの挿入」を選択します。アクションパネルを開き、第1フレームに戻り、りんご落下を繰り返す命令文を記述します。



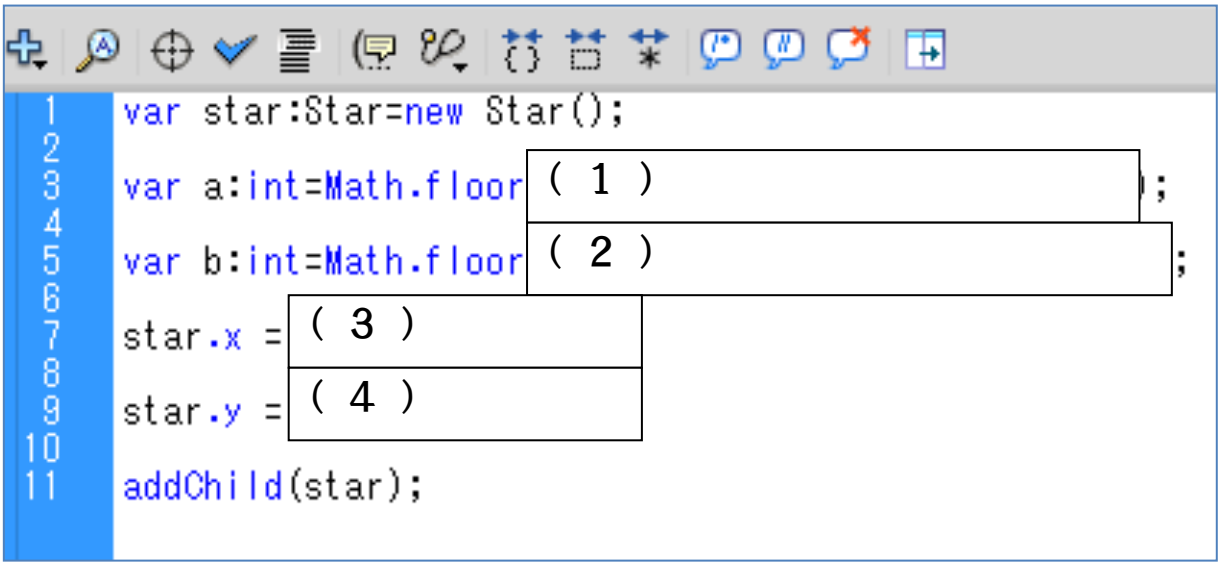
1 行目： 第2フレームに移動し、実行されます。



＊＊これでりんごの動きが設定されました。＊＊

ワークシート『ムービークリップの出現』

問1．ムービークリップのランダムな出現を次の条件で表示させてみよう。
ムービークリップは「star」とし使用する。
star **出現のx座標:ステージ上のランダムな位置**
star **出現のy座標:ステージ上のランダムな位置**



答えができたら、Actionscript画面に記述して、実行してみましょう。

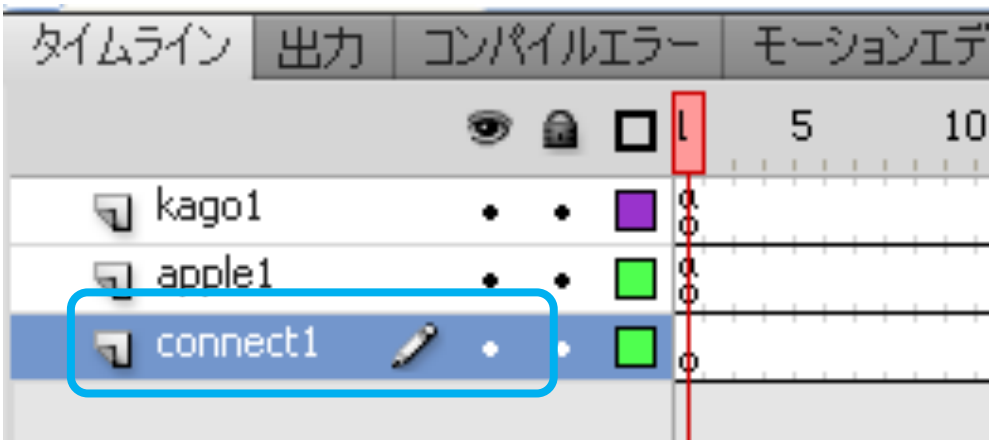
課題4. Wii リモコンの接続状態をテキストフィールドに表示させましょう。

本時の目標

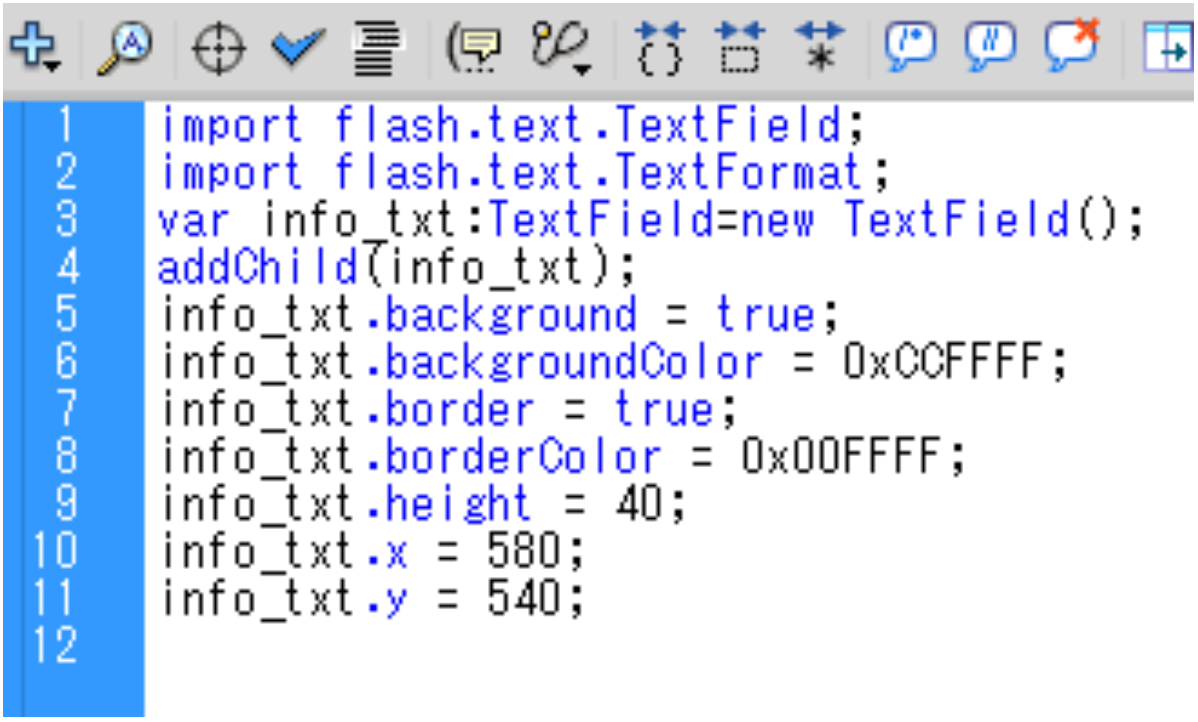
1. テキストフィールドの表示を Actionscript でプログラムできる。

右下に Wii リモコンの接続状態をテキストフィールドに表示し、またテキストフィールドの背景色設定、境界線色設定、フィールドの高さ設定、配置される座標設定を行います。

STEP① レイヤーの挿入を行い名前を付けます。レイヤの挿入を行い、 レイヤー1にカーソルを合わせ、ダブルクリックして名前を「connect1」とします。



STEP② **第1フレームにカーソルを合わせ、アクションパネルを開きます。**
アクションパネルには次のように記述します。



```
1  import flash.text.TextField;
2  import flash.text.TextFormat;
3  var info_txt:TextField=new TextField();
4  addChild(info_txt);
5  info_txt.background = true;
6  info_txt.backgroundColor = 0xCCFFFF;
7  info_txt.border = true;
8  info_txt.borderColor = 0x00FFFF;
9  info_txt.height = 40;
10 info_txt.x = 580;
11 info_txt.y = 540;
12
```

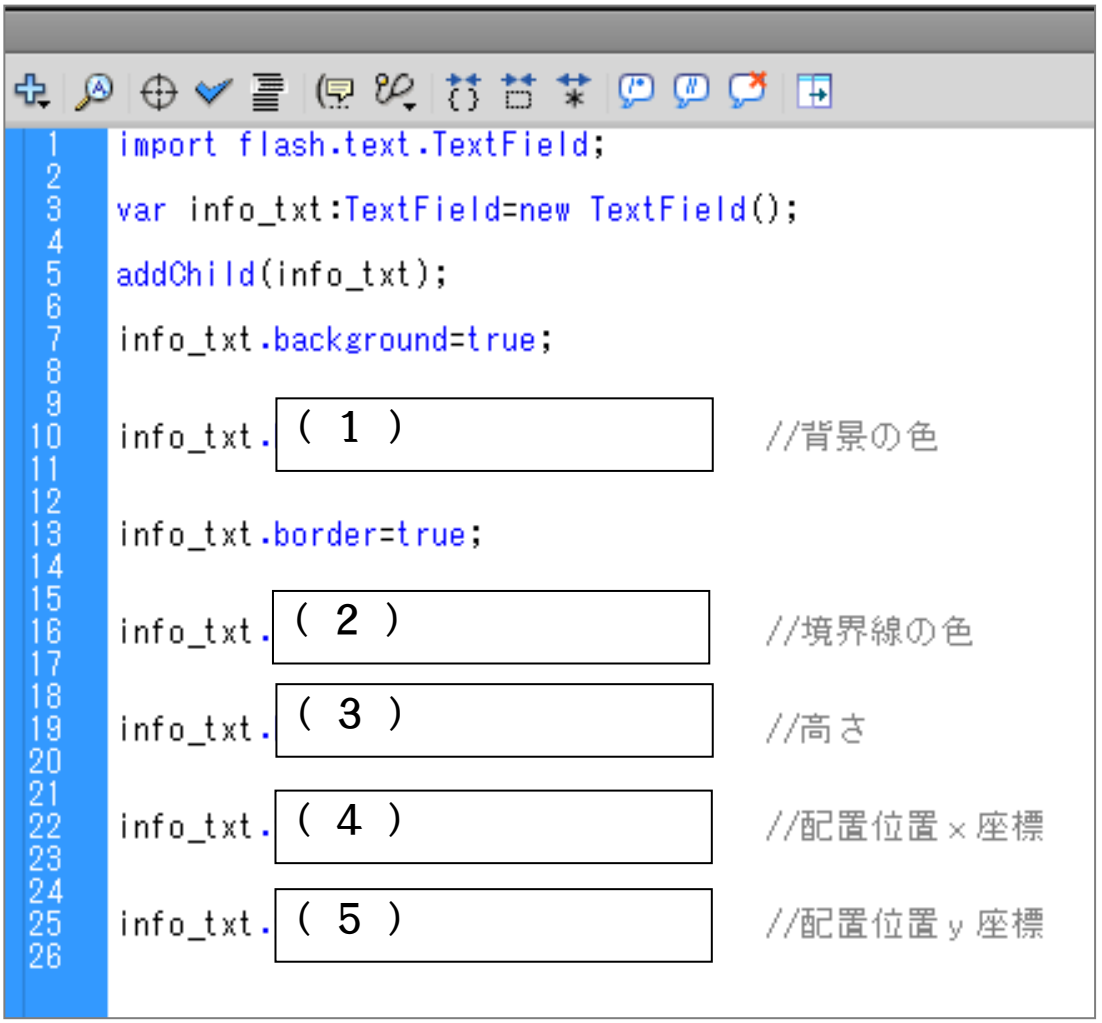
プログラム解説

1行目：自動挿入
3行目：変数「info_txt」の宣言
4行目：画面に表示。
5行目：背景の表示。
6行目：背景の色。
7行目：境界線の表示。
8行目：境界線の色設定。
9行目：テキストフィールドの高さ設定。（幅は設定なし、文字数により変化）
10行目：テキストフィールドの配置x座標。（ステージ幅が700なので580に設定）
11行目：テキストフィールドの配置y座標。（ステージ幅が580なので540に設定）

ワークシート『テキストフィールドの表示』

問1．テキストフィールドの表示を次の条件で表示させてみよう。

背景の色：表示、緑色（0x33FF00）
境界線の色：表示、紺色（0x000033）
高さ：20
配置位置：x座標500
配置位置：y座標100



```
1  import flash.text.TextField;
2
3  var info_txt:TextField=new TextField();
4
5  addChild(info_txt);
6
7  info_txt.background=true;
8
9
10 info_txt.（ 1 ）          //背景の色
11
12
13 info_txt.border=true;
14
15
16 info_txt.（ 2 ）          //境界線の色
17
18
19 info_txt.（ 3 ）          //高さ
20
21
22 info_txt.（ 4 ）          //配置位置ｘ座標
23
24
25 info_txt.（ 5 ）          //配置位置ｙ座標
26
```

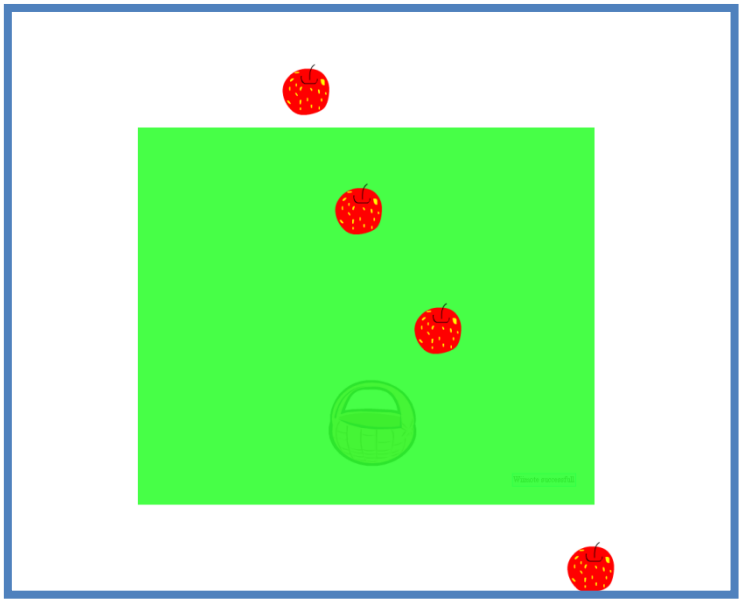
答えができたら、Actionscript画面に記述して、実行してみましょう。
また別紙『Webセーフカラー216色』表より、自分の使用したい色を選択し、同様に記述及び実行してみましょう。

課題5. 背景のムービークリップ(Sprite)を表示させましょう。

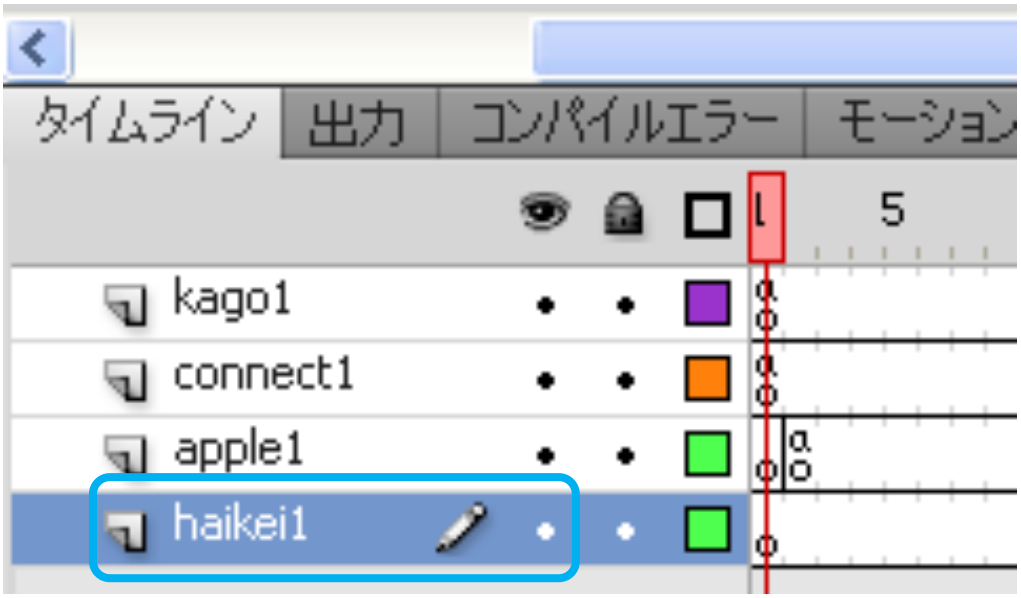
本時の目標

1. タイムラインのないムービークリップ(Sprite)をActionscriptで描くことができる。
2. オブジェクトの表示する順番を変更できる。
3. ステートメントtraceを使い、正しく動いているか確認できる。

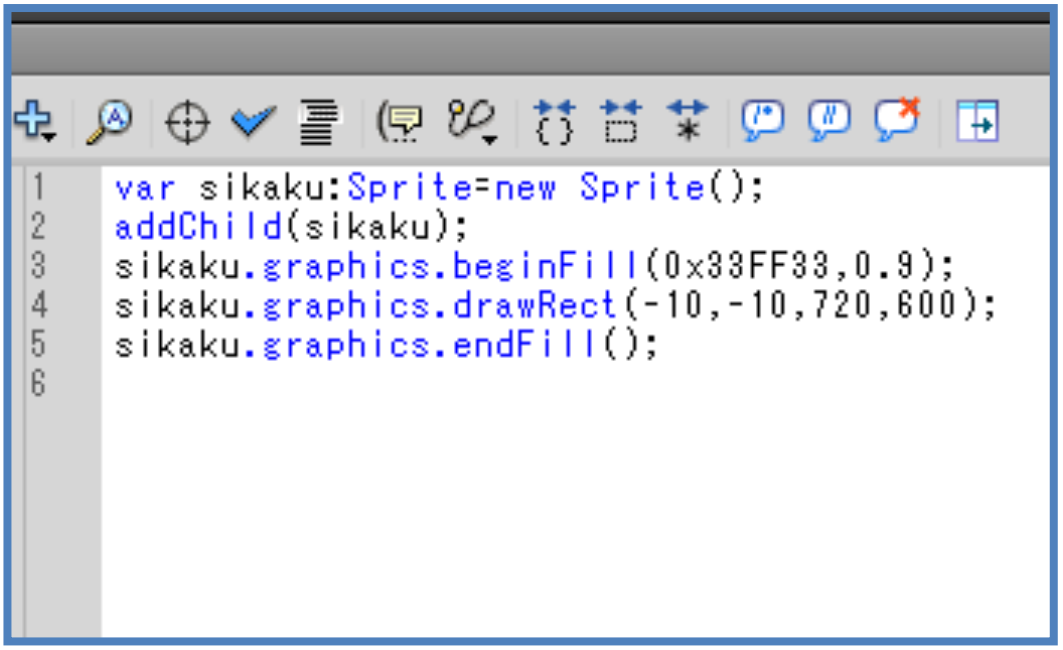
タイムラインのないムービークリップを Sprite(スプライト)といいます。ステージに四角形だけを描く場合、タイムラインが必要ないので、処理速度を速めるためにも Sprite にて表示させます。
また、ステージ全体に Sprite が描かれるため、下図のようにかごが後面にいってしまいます。重なるオブジェクトの順番を変更する必要があります。



STEP① **横700px×縦580pxのステージ全体に緑色の四角形を描きます。新しくレイヤーの挿入を行い名前を付けます。レイヤの挿入を行い、 レイヤー1にカーソルを合わせ、ダブルクリックして名前を「haikei1」とします。**



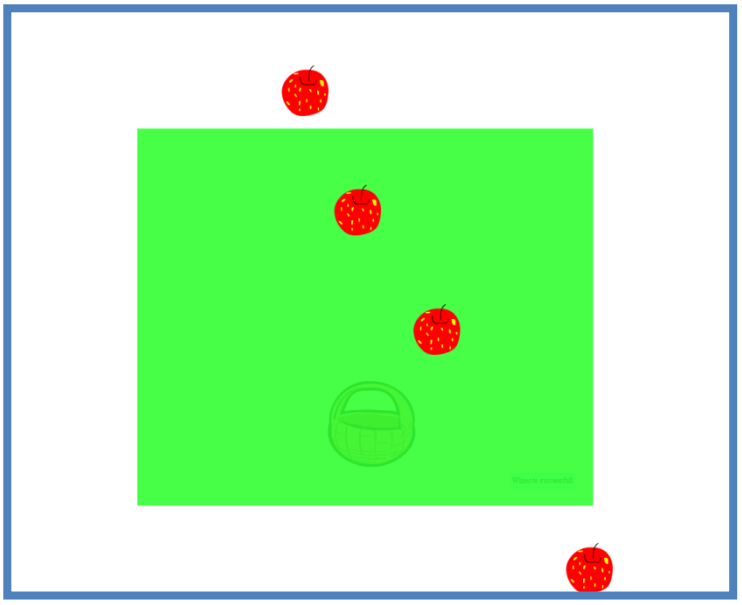
STEP② **第1フレームにカーソルを合わせ、アクションパネルを開きます。アクションパネルには次のように記述します。**



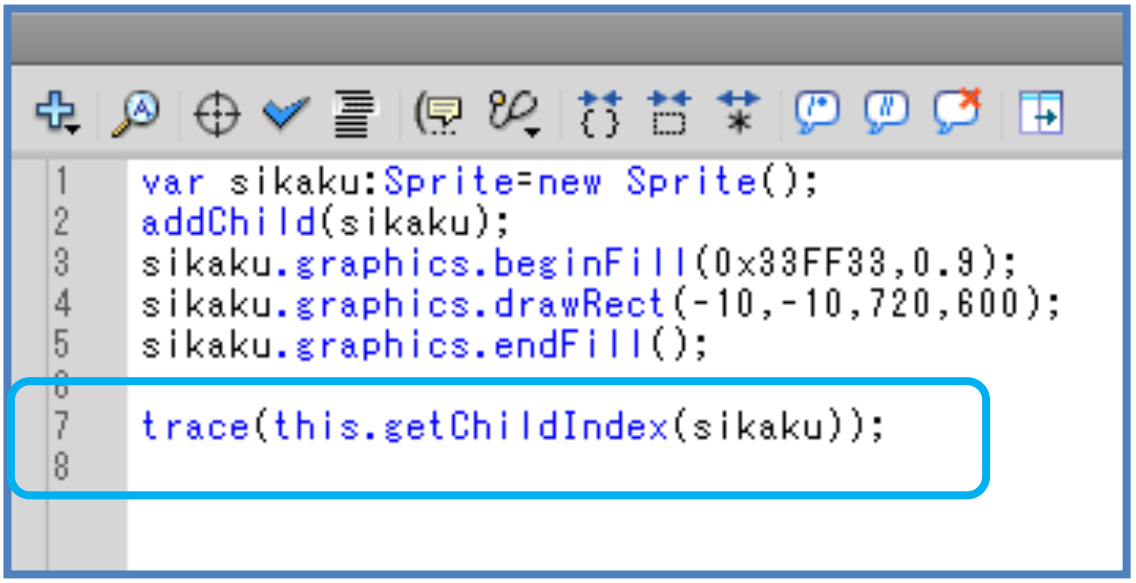
```
1  var sikaku:Sprite=new Sprite();
2  addChild(sikaku);
3  sikaku.graphics.beginFill(0x33FF33,0.9);
4  sikaku.graphics.drawRect(-10,-10,720,600);
5  sikaku.graphics.endFill();
6
```

1行目：変数「sikaku」の宣言、タイムラインのないムービークリップのため、Sprite で表示します。
2行目：「sikaku」を画面に表示します。
3行目：「sikaku」の色は緑（0x33FF33）で、透明度は0．9。（透明度は0～1．0までで、0が透明）
4行目：四角形は「drawRect」でx座標－10、y座標－10、幅720、高さ600。
他にも円、だ円、曲線を描くことができます。円は「drawCircle」、だ円は「drawEllipse」、曲線は「curveTo」になります。
5行目：図形の塗りつぶしをやめる。

＊＊ムービープレビューを実行して、動作確認を行います。すると、下図のようにスプライト「sikaku」とムービークリップ「kago」とムービークリップ「apple」の3つのオブジェクトが重なっており、ステートメントtrace（）を使い、スプライト「sikaku」の重なり順を調べます。

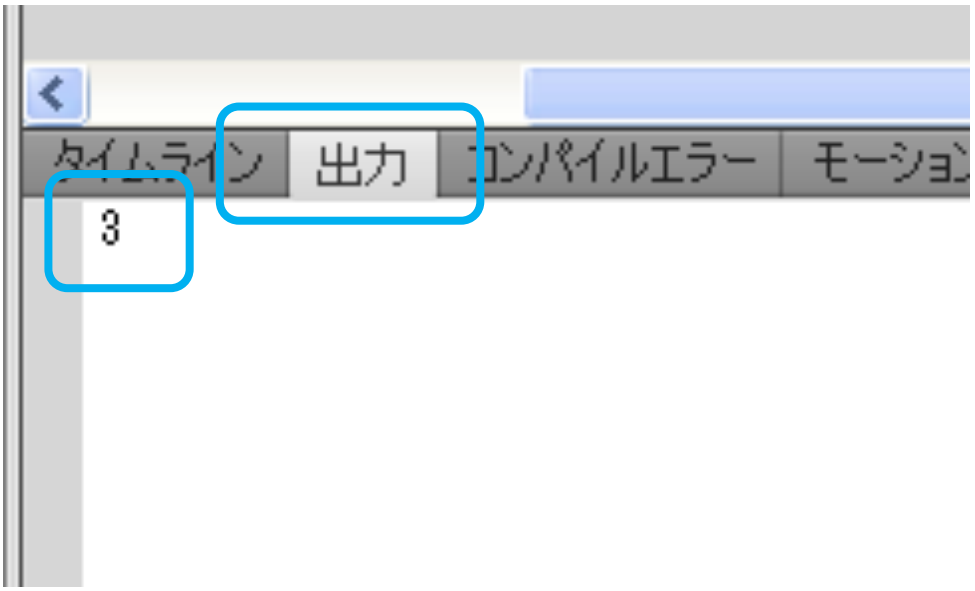


STEP③　STEP②に続けて記述します。



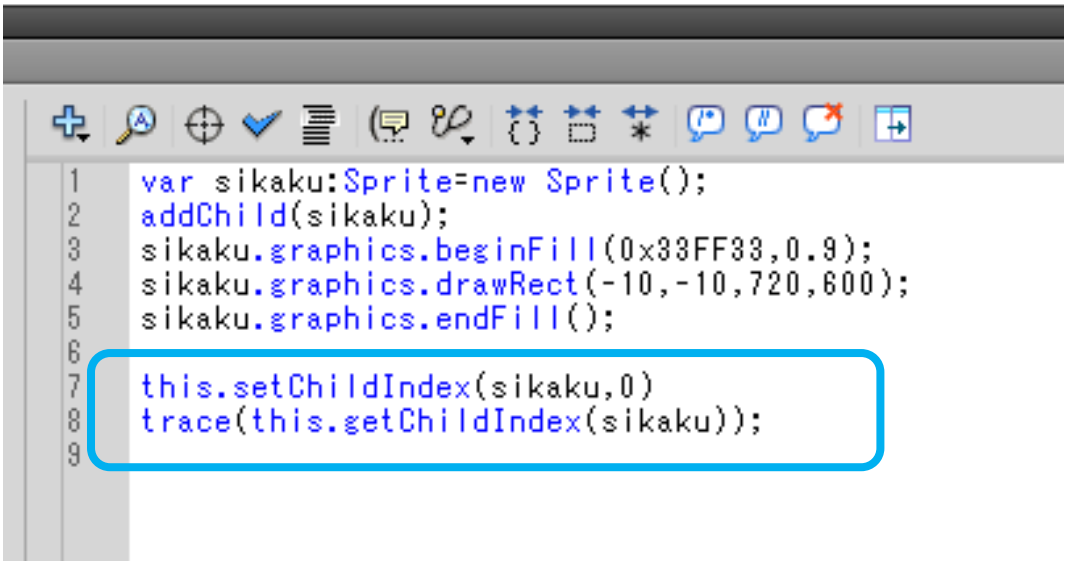
```
1  var sikaku:Sprite=new Sprite();
2  addChild(sikaku);
3  sikaku.graphics.beginFill(0x33FF33,0.9);
4  sikaku.graphics.drawRect(-10,-10,720,600);
5  sikaku.graphics.endFill();
6
7  trace(this.getChildIndex(sikaku));
8
```

記述後、ムービープレビューを実行します。



タイムラインパネルの出力表示に「3」と表示されます。奥から3番目にSprite「sikaku」があることが分かります。これを一番奥となる0番目に変更する必要があります。

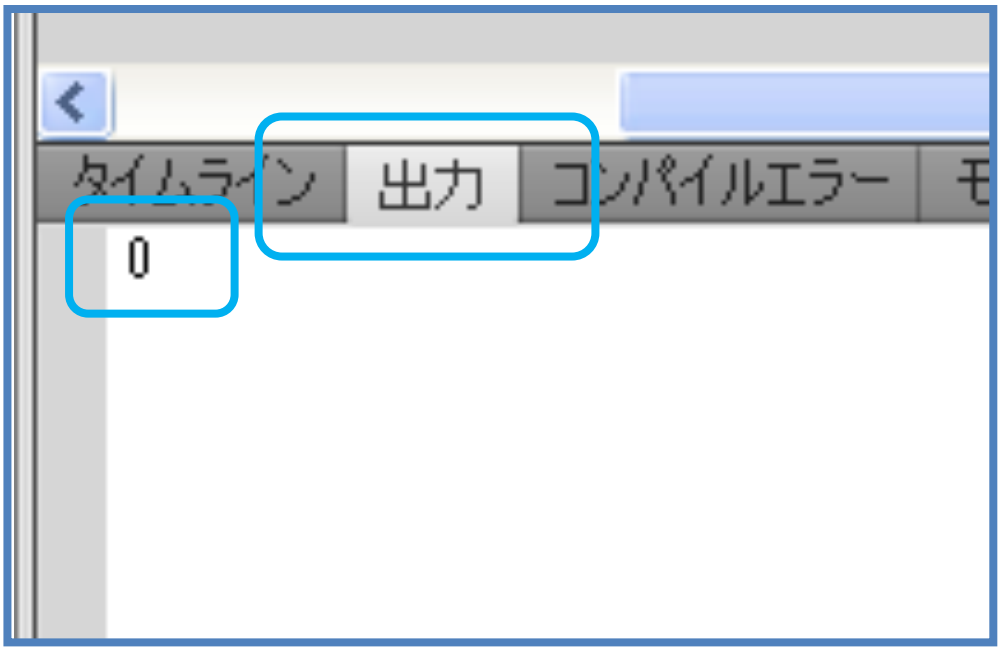
STEP④　Sprite「sikaku」の表示を3番目から0番目に変更します。



```
1  var sikaku:Sprite=new Sprite();
2  addChild(sikaku);
3  sikaku.graphics.beginFill(0x33FF33,0.9);
4  sikaku.graphics.drawRect(-10,-10,720,600);
5  sikaku.graphics.endFill();
6
7  this.setChildIndex(sikaku,0)
8  trace(this.getChildIndex(sikaku));
9
```

7行目：「sikaku」の重なりを一番後面、0番目にする
8行目：traceで出力パネルに何番目に表示されているかを調べる。

記述後、ムービープレビューを実行すると、出力パネルに0番目と表示されます。



＊＊このように確認したい変数の値をtrace文で確認することができます。＊＊

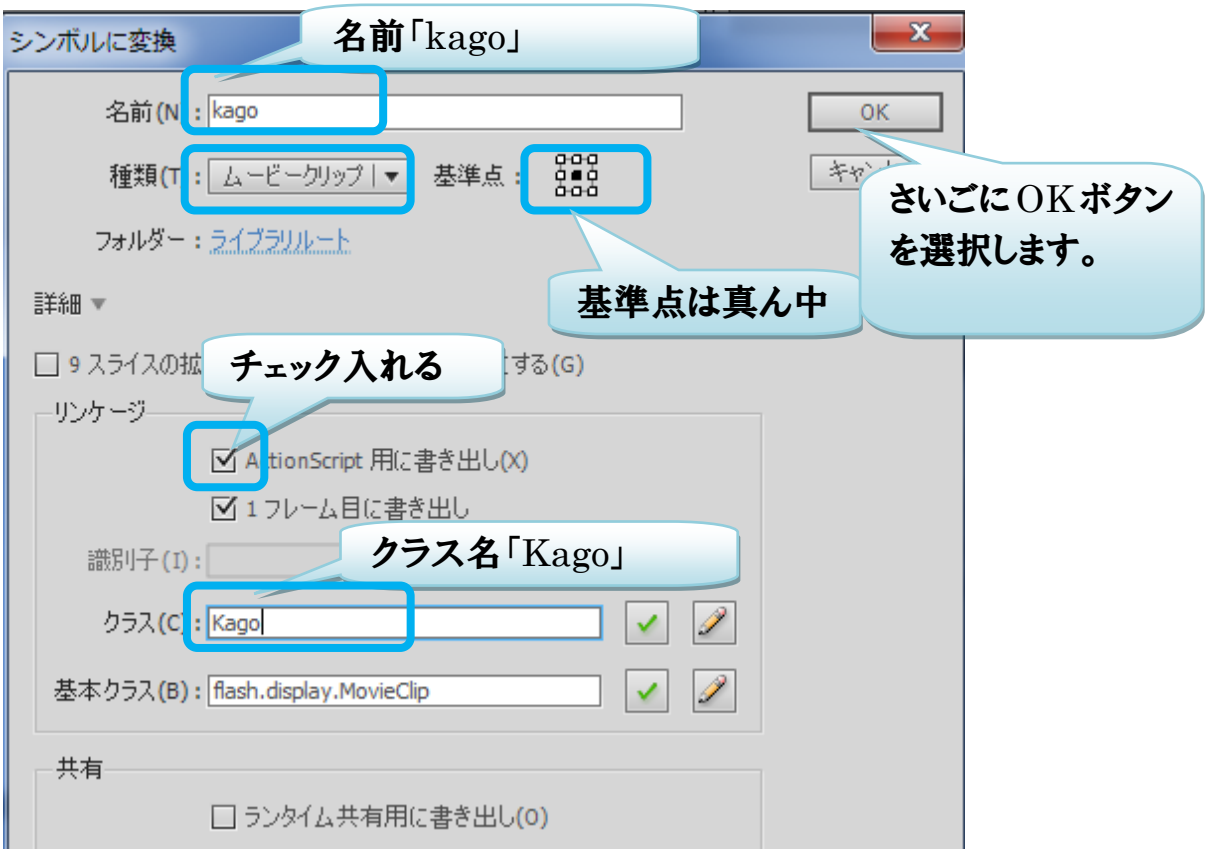
課題6. りんごを拾うかごを作成しましょう。

本時の目標

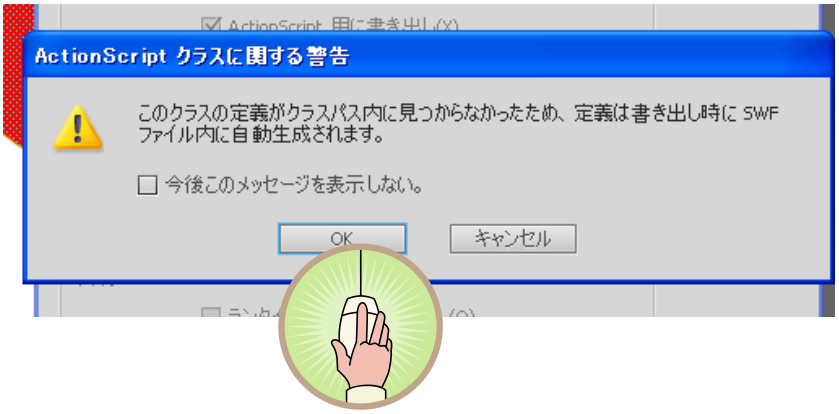
1．ムービークリップをマウスで操作できるように

Actionscript でプログラムできる

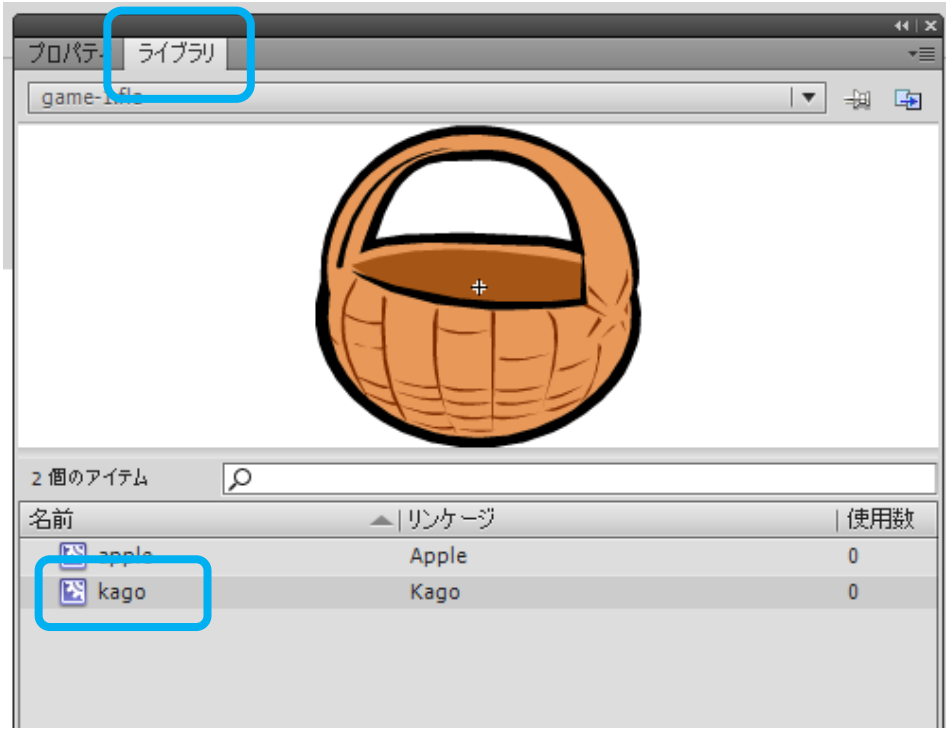
STEP① **かごのイラストを描き、シンボルに変換します。下記部分を選択します。**



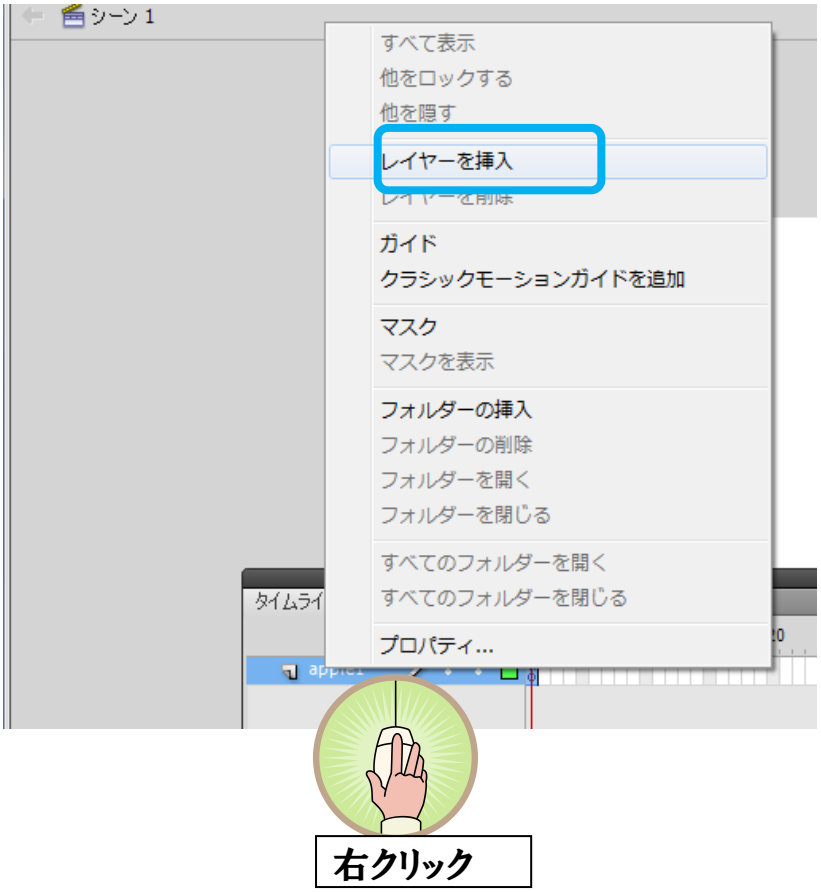
クラスパスに関する警告が表示されますので OK ボタンを選択します。



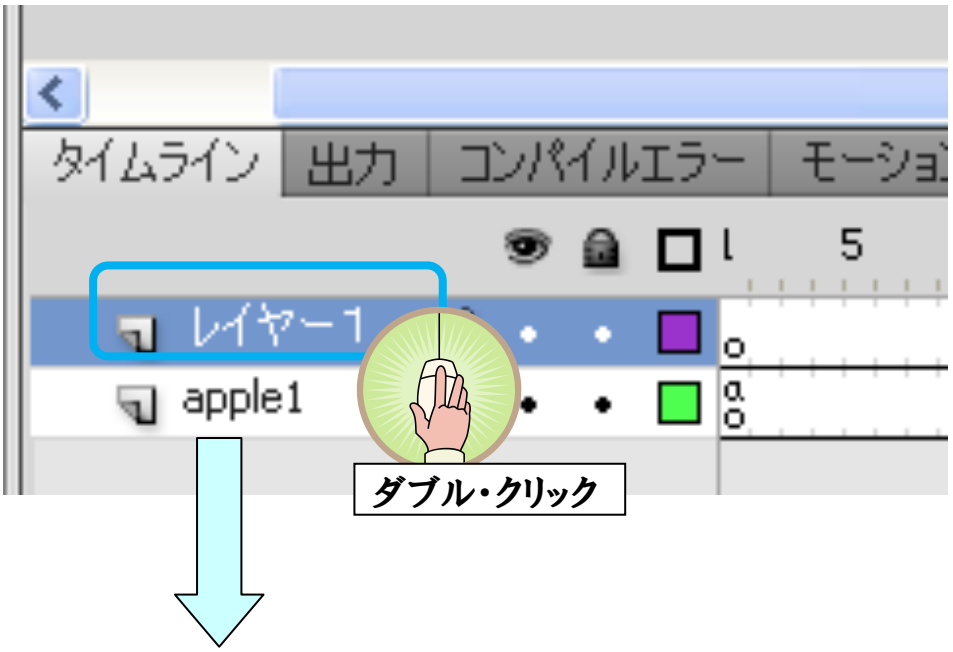
これでライブラリに、ムービークリップ「kago」が登録されました。

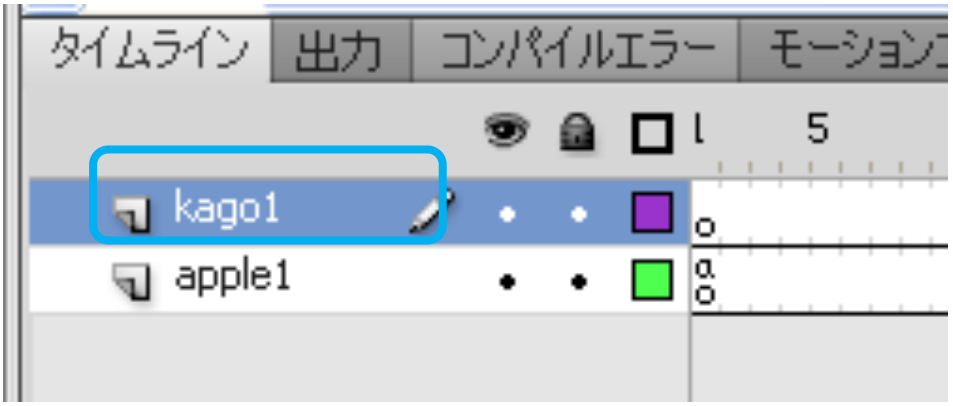


STEP② **タイムラインパネルに新しくレイヤーを挿入します。**
レイヤー名「apple1」上にカーソルを持ってきて、右クリックのメニューから「レイヤーを挿入」を選択します。

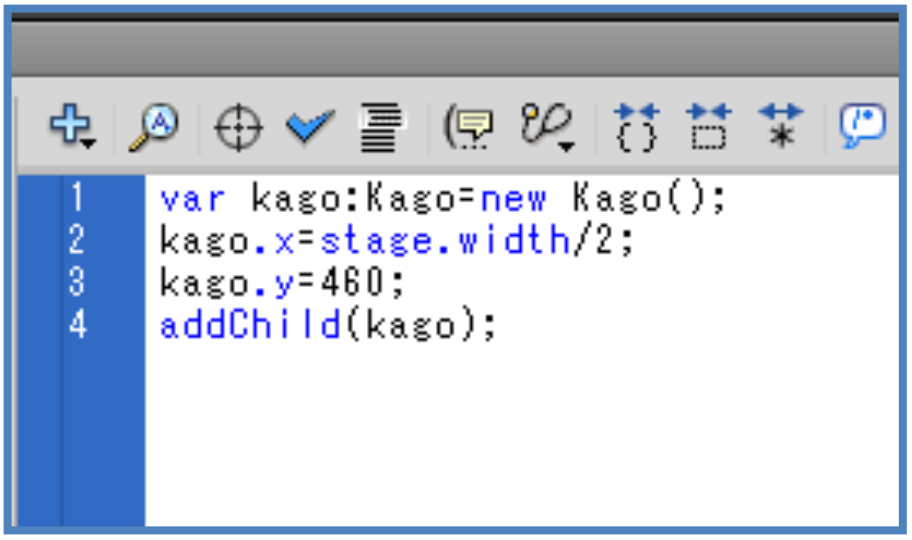


STEP③ **新しく挿入したレイヤー１をレイヤー名「kago1」にします。**
レイヤー１上でマウスをダブルクリックして名称を「kago1」とします。





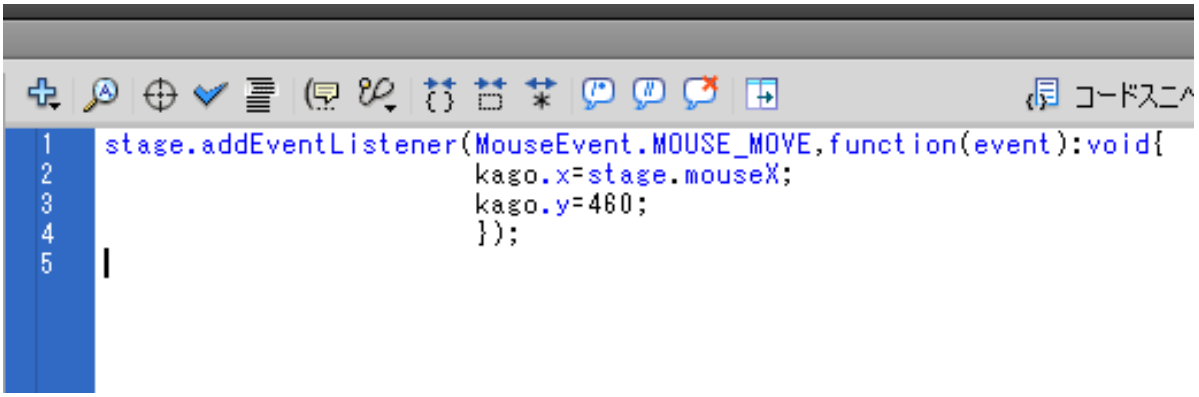
STEP④ **レイヤ「kago1」の第１フレームにおいて、アクションパネルを開き、スタート時にかごを画面下部中央に配置する記述を行います。**



1行目：　変数 kago の宣言。
2行目：　開始前のかごのx座標はステージ中央。
3行目：　開始前のかごのy座標は460。
4行目：　画面に表示。

＊＊これでかごがステージに表示される状態になりました。＊＊
STEP⑤では、かごがカーソルの移動により操作するプログラムを加えていきます。

STEP⑤　第2フレームにおいて、アクションパネルを開き、かごをマウスによって操作させるプログラムを記述していきます。



1行目：　マウスイベントの宣言。
2行目：　マウスカーソル位置をかごのx座標に反映。
3行目：　かごのy座標は460で固定。

＊＊これでかごをマウス操作によって移動することができます。＊＊
今回はマウスでの操作になりますが、Wiiリモコンの傾きによる3軸加速度センサを用いて操作することも可能です。まず周辺機器の設定を行っていきます。

課題7．Wiiリモコンについて学習しましょう。

本時の目標

1．Wiiリモコンの機能が理解できる。

STEP①　WiiRemoteの外装

Nintendo社WiiRemote
ウィーリモートまたはウィーリモコンとよばれる。
赤外線CMOSセンサまたは3軸加速度センサなどが搭載されているゲーム機リモコンである。

STEP②　WiiRemoteの仕様

サイズ	縦 148mm、横 36.2mm、厚さ 30.8mm
通信機能	Bluetoothによる無線接続、最大接続4台
プレイ可能距離	モニタから5m
ポインター	画面の指し示すポインティング機能
モーションセンサ	傾きや動きの変化を3軸で検出
ボタン	デジタル11入力
振動機能	バイブレーター1個
スピーカー	モノラルスピーカー1個
プレイヤーインジケーター	青色LED4個
拡張ユニット接続可能	ヌンチャクなど

ゲーム機リモコンであるが、安価で多機能なため学習教材として使用していきます。

課題8. 無線機器Bluetoothについて学習および設定しましょう。

本時の目標

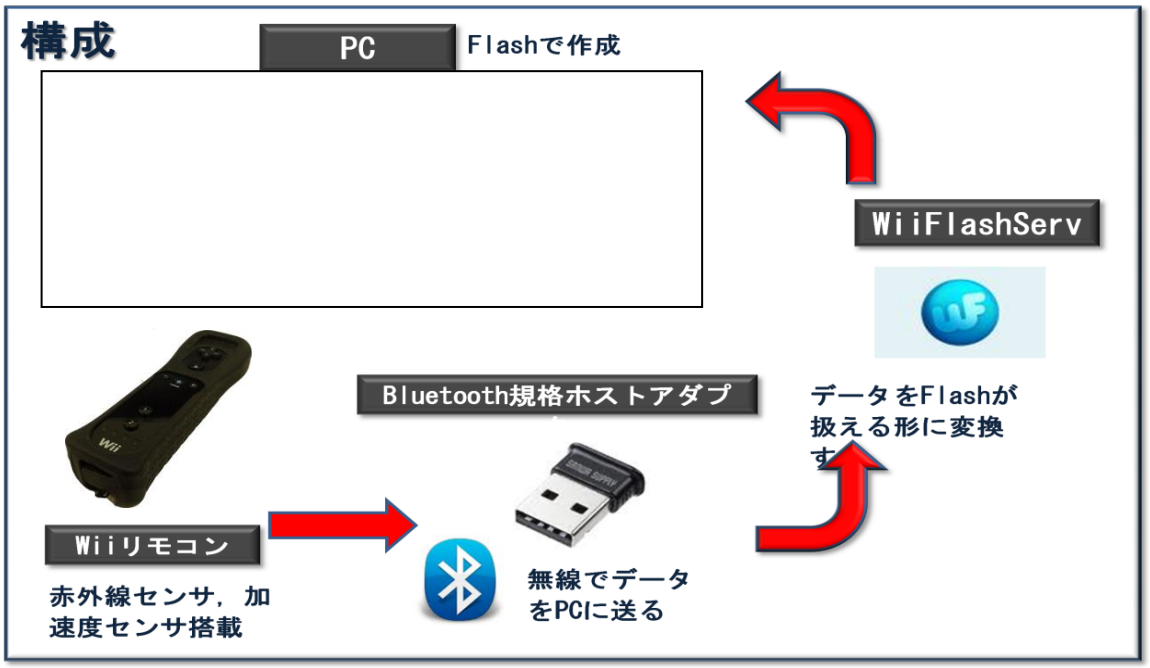
1. 無線機器Bluetoothのドライバをインストールできる

2. 無線機器 Bluetooth を起動させ，Wii リモコンをパソコンに接続できる。

無線通信規格 Bluetooth の設定

Bluetooth
2. 4GHzの電波を使用したワイヤレス通信システムです。
パソコン・携帯電話・プリンタ・PC 用のキーボードなどの製品間のワイヤレス通信を約10m程度の範囲で行う共通規格です。

今回は Wii リモコンからのデータをコンピュータにワイヤレス通信することに使用します。

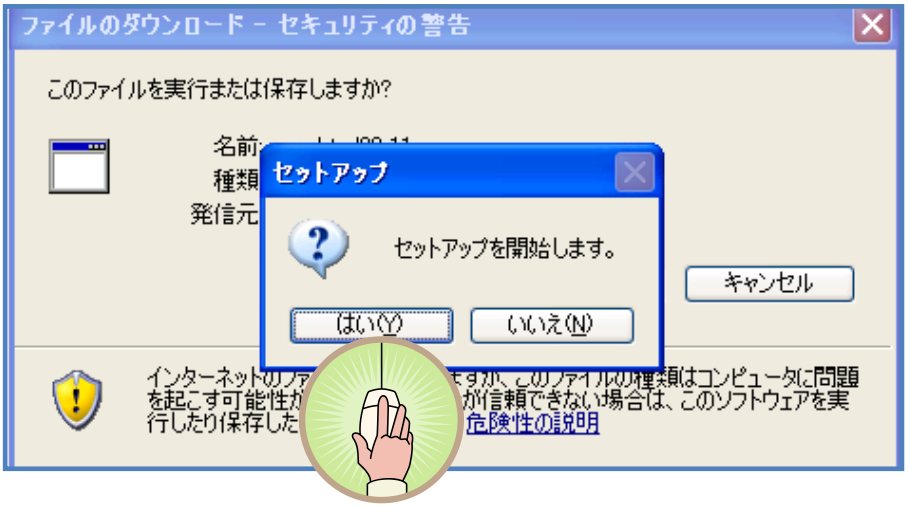


STEP① SANWASUPPLY 社 BluetoothUSB アダプタ「MM―BTUD23」を使用します。

SANWASUPPLY 社ホームページ（http://www.sanwa.co.jp/support/download/）よりソフト（ドライバ）をダウンロードします。

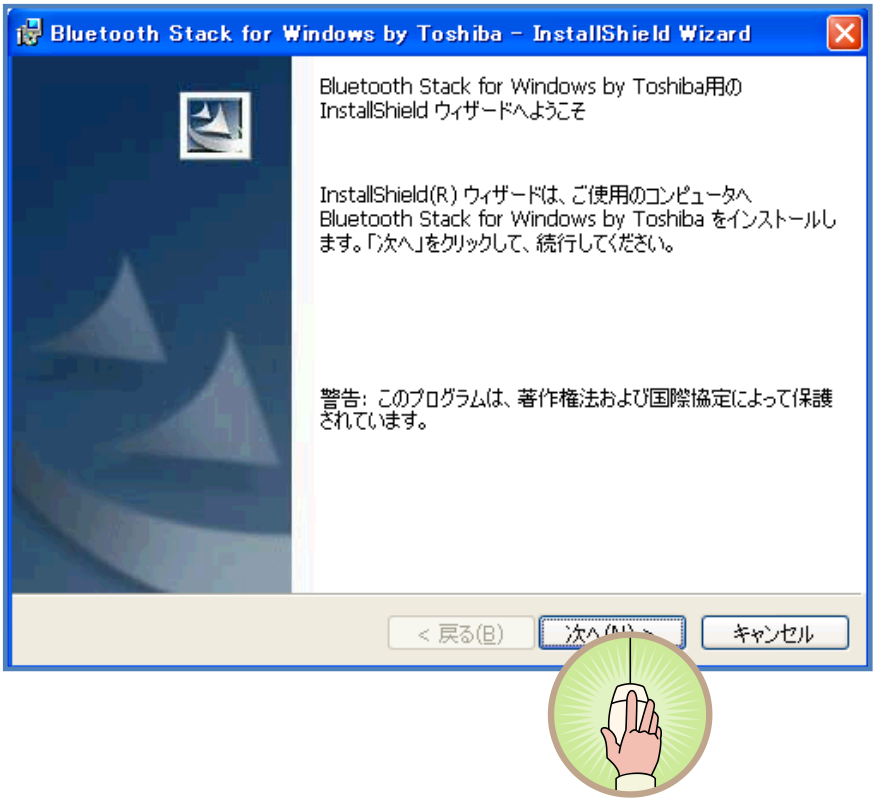
STEP② ファイル「mmbtud22_11.exe」のダウンロードを実行します。

STEP③セットアップを開始します。

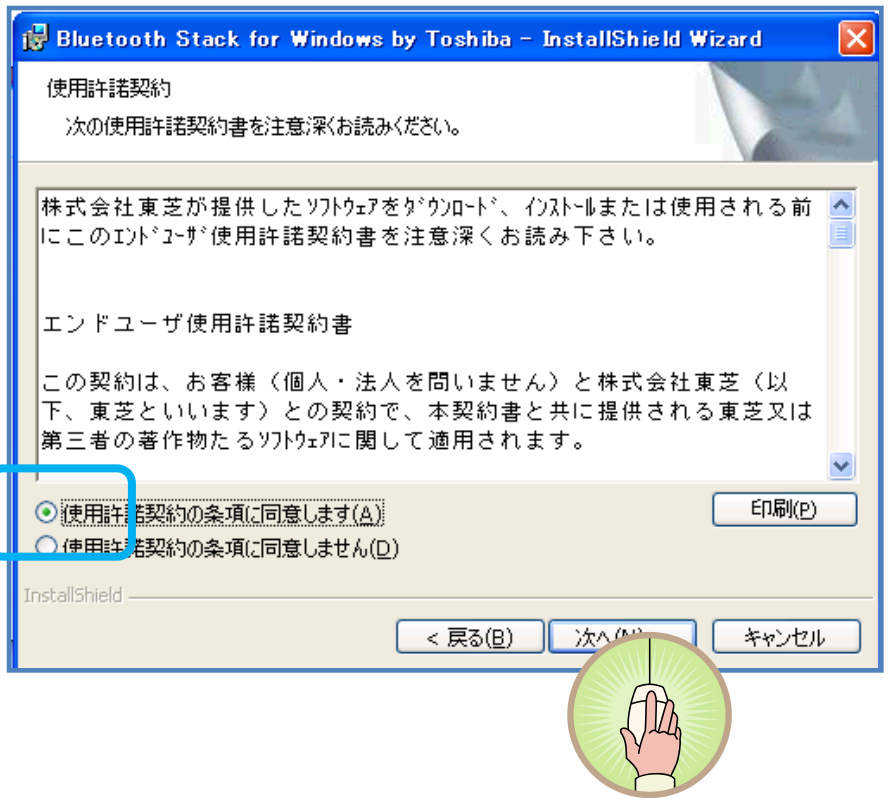


STEP④ 「ドライバをインストール」を選択します。

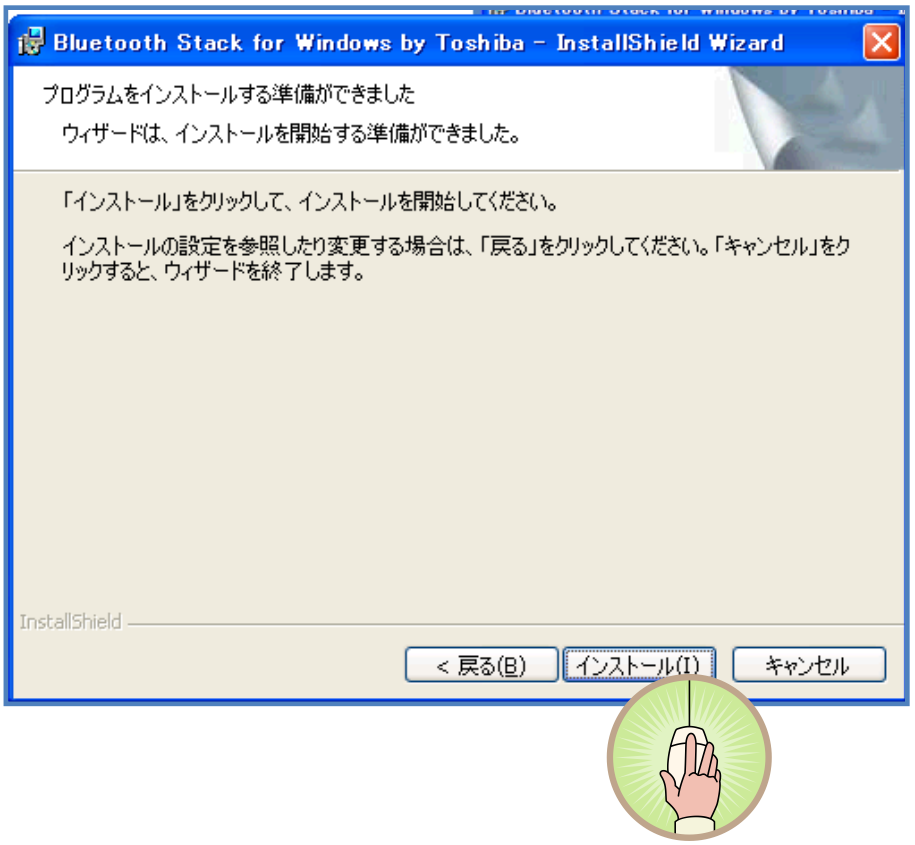
STEP⑤ InstallShield ウィザード画面となります。
「次へ」を選択します。



STEP⑥使用許諾契約の条項を確認し,同意出来れば「使用許諾契約の条項に同意します」を選び,「次へ」を選択します。

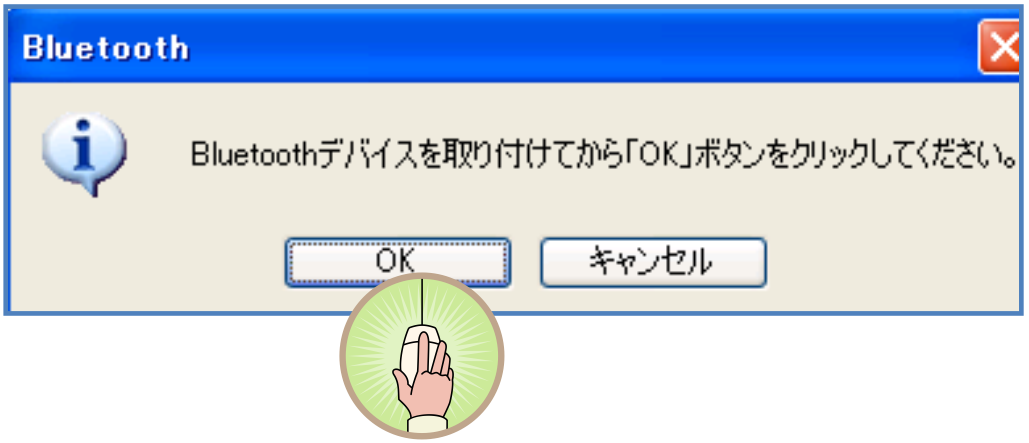


STEP⑦ 「インストール」を選択します。

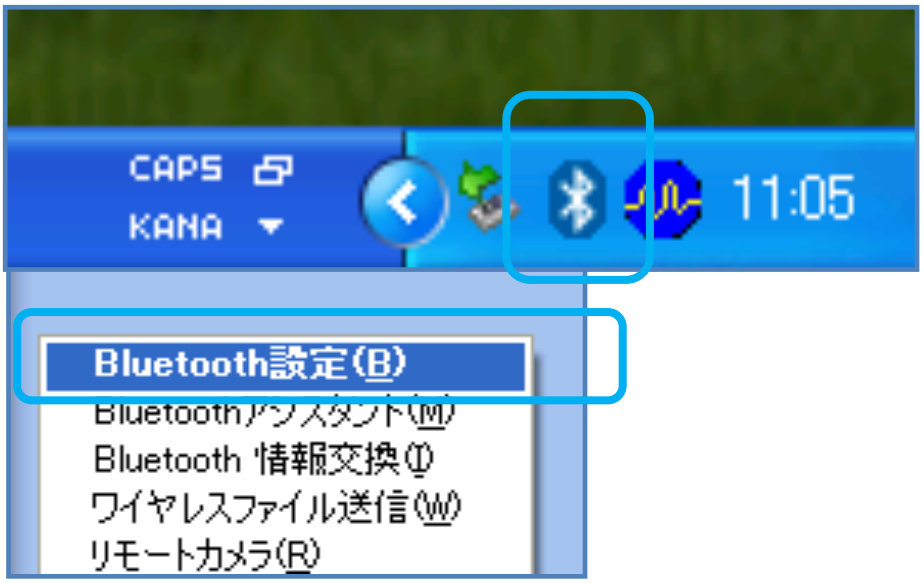


STEP⑧ Bluetooth アダプタである「MM―BTUD２３」をコンピュータのUSB 端子に接続し，「OK」ボタンをクリックします。

Bluetooth 用ドライバがインストールされます。これで Bluetooth アダプタ（MM-BTUD２３）のドライバがインストールされました。



STEP⑨　続いて Bluetooth 機器「WiiRemote」を認識させていきます。
Bluetooth アダプタ「MM-BTUD23」を接続し，アイコンを右クリックし，メニューより「Bluetooth 設定」を選択します。



STEP⑩　Bluetooth 機器「WiiRemote」を認識させるため，
「新しい接続」を選択します。

STEP⑪ 「新しい接続の追加ウィザード」の「エクスプレスモードおすすめ」を選択します。

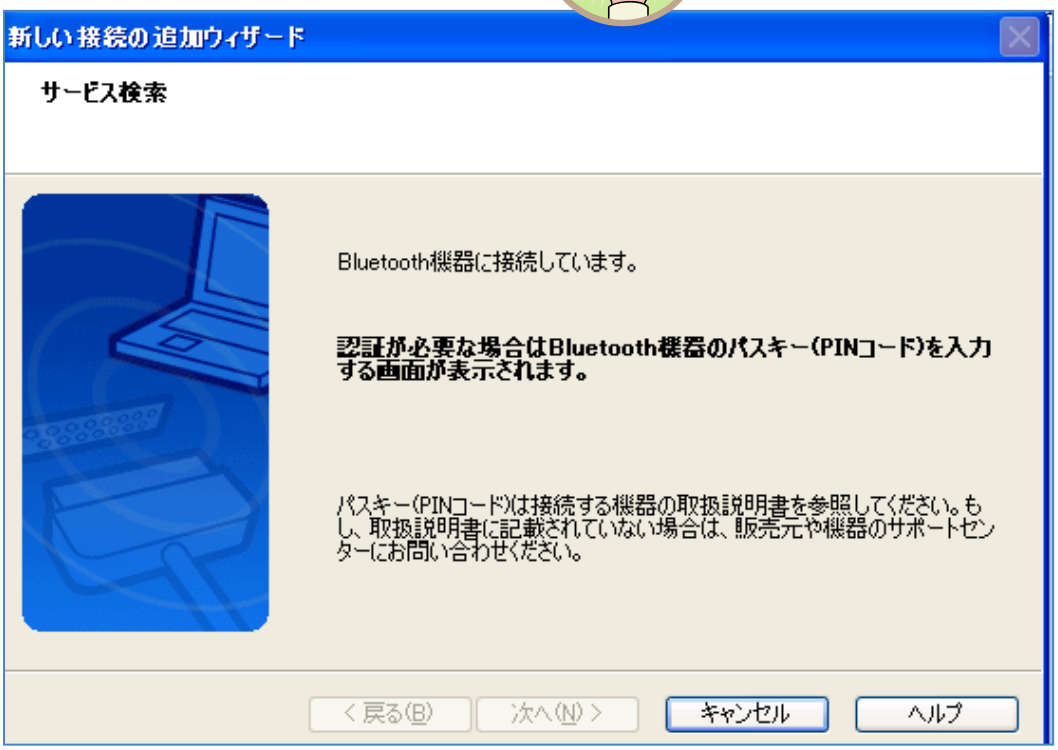
この際，Bluetooth 機器（Wiiremote）の電源を ON にし，探索可能状態にしておきます。

STEP⑫ Bluetooth 機器（Wiiremote）の検索が始まるので，Wiiremote のボタン1とボタン2を押し，Bluetooth 機器（Wiiremote）を認識させます。

ボタン1とボタン2を押す。

STEP⑬ Wiiremote は「Nintendo RVL-CNT-01」と認識されるので，選択し「次へ」を選択します。

次に Wiiremote に接続します。「サービス検索」中に Wiiremote のボタン1とボタン2を押します。

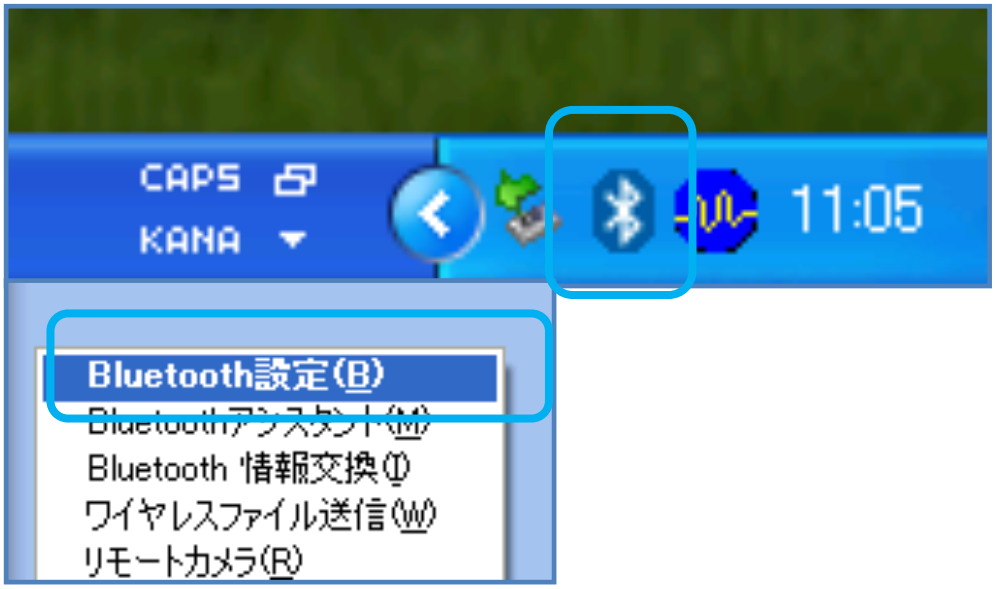


STEP⑭ これで Wiiremote **が「**Nintendo　RVL-CNT-01**」というデバイス名で接続できました。**

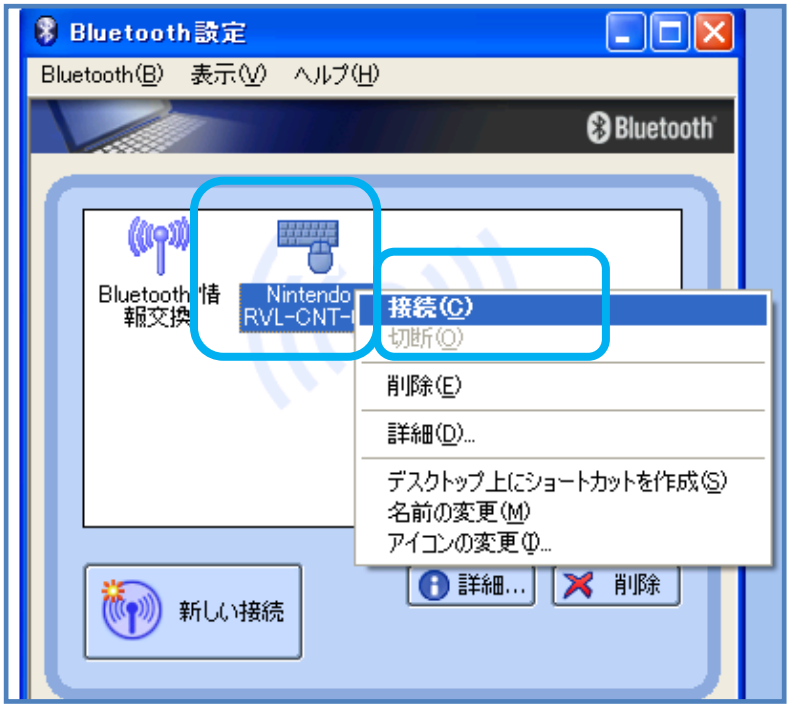
＊接続されている状態

＊接続されていない状態

STEP⑮　2回目からの接続方法は右下にある Bluetooth アイコン　　を右クリックし，メニューから「Bluetooth 設定（R）」を選択します。



STEP⑯　「Bluetooth 設定」から「Nintendo　RVL-CNT-01」を選択し，Wiiremote のボタン1とボタン2を押しながら，「接続（C）」を選択します。右図が接続された状態です。



＊接続されている状態

課題9. WiiFlashについて学習および設定しましょう。

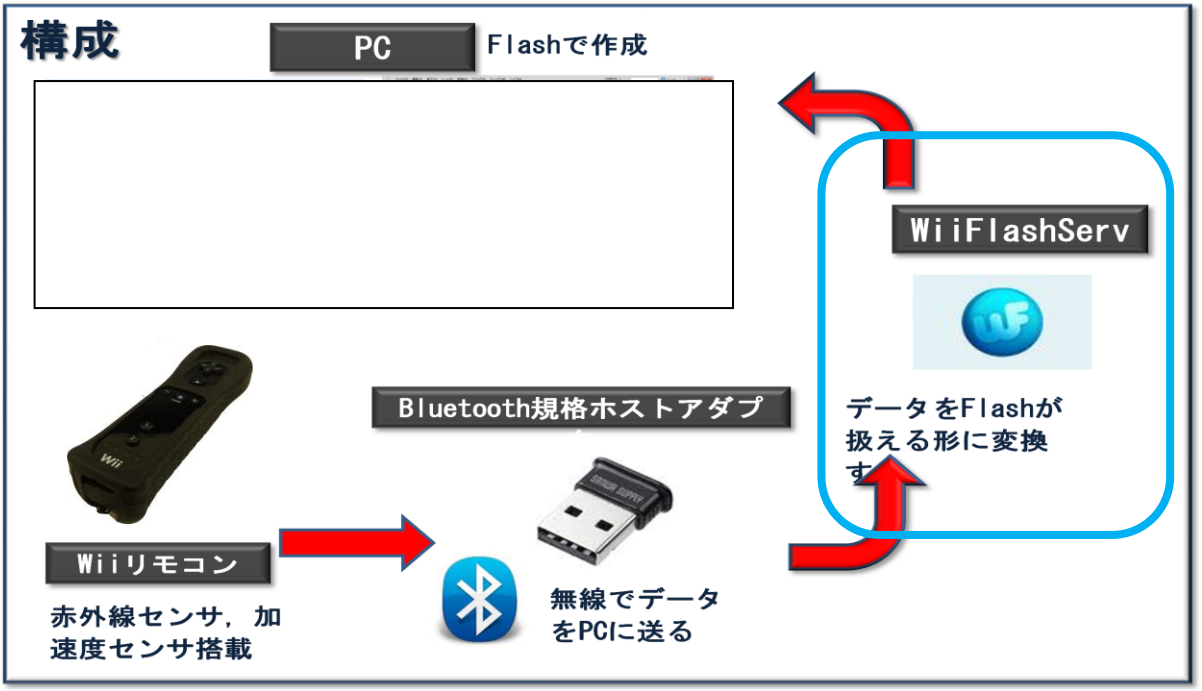
本時の目標

1. WiiFlashServer をインストールできる。

WiiFlash とは Actionscript を用いて WiiRemote と Flash アプリケーションをつなぐためのライブラリです。(http://wiiflash.bytearray.org/)
「WiiFlashServer」と「WiiFlashActionScriptAPI」に分けて提供されています。

＊WiiFlashServer
バイナリソケットサーバーで WiiRemote からの情報を Flash が扱える形に変換して通信します。

＊WiiFlashActionScriptAPI
WiiFlashServer を使って WiiRemote と通信するための API です。



STEP① WiiFlash のダウンロード

WiiFlash サイト(http://wiiflash.bytearray.org/)
を開き、Download を選択します。
「WiiFlash0[1].4.5.zip」をダウンロードします。

STEP② 「WiiFlash0[1].4.5.zip」をデスクトップに保存し，圧縮フォルダを「すべて展開」します。

STEP③ ダウンロードした WiiFlash のアーカイブを展開すると，次のようなフォルダが現れます。それぞれの概要については下表に示す内容になります。

フォルダ名	概要
Server/WiiFlash Server	WiiFlash Serve 本体
Core/api	WiiFlash API
Examples	デモファイル(.fla)
Documentation	ドキュメント

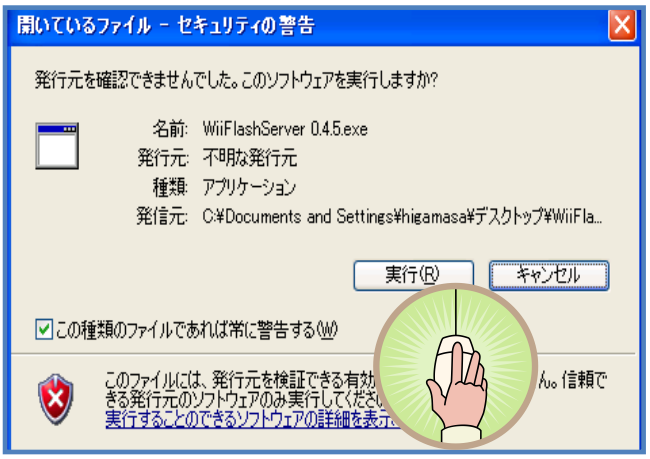
Coreフォルダ－api フォルダ－source-classes フォルダ内の「org」フォルダ内に今回使用する Actionscript ファイルが一括して入ってあります。ライブラリを用いる場合は「org」フォルダと同じ場所に Flash ファイルを置くようにします。

STEP④

フォルダ「Servers」を開き，実行ファイル「WiiFlashServer0.4.5.exe」を選択します。

STEP⑤　Bluetooth **設定の** WiiRemote(NitendoRVL-CNT-01)**が認識されている状態で，実行します。**

＊Bluetoothが接続されている状態



STEP⑥ WiiFlashServer に WiiRemote が正しく接続された場合はこのように青く表示します。

また接続ができていない場合は赤く表示され，手順を再度おこなう必要があります。

これで，設定は完了となり、WiiRemote と Flash アプリケーションをつなげることができます。

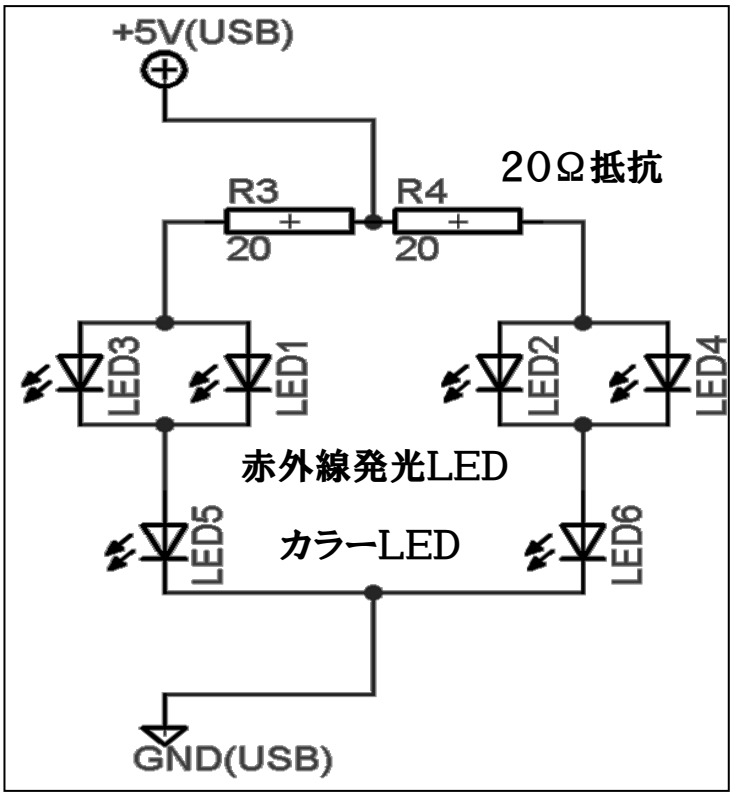
課題10. 赤外線センサバーをつくろう。

本時の目標

1. LEDを用いた電子工作により赤外線センサバーをつくる。

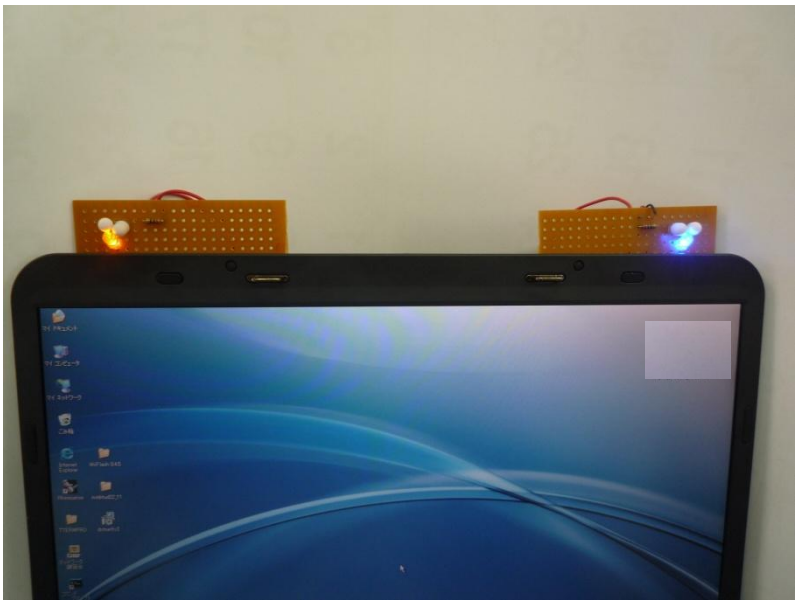
STEP① 準備するものはこのようになります。
20Ωカラー抵抗 ×2
赤外線発光LED×4
（今回使用したのは OptoSupplyLimited 社 PSIR5113A）
点灯確認用LED×2
LED拡散キャップ×4
携帯電話充電用USBケーブル×1
ユニバーサル基盤×1
はんだごて一式
はんだ

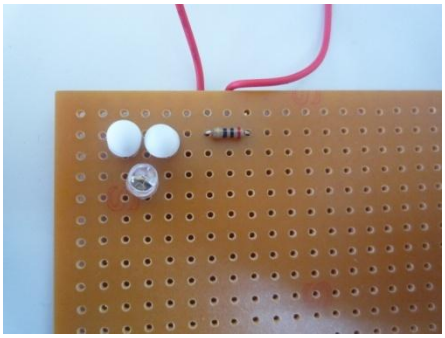
STEP② 製作回路図はこのようになります。

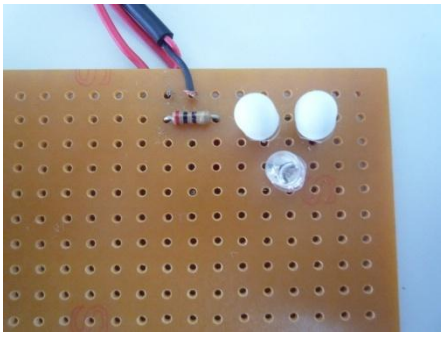


STEP③　電源は平型のUSBプラグは金属端子面を下にして，１番左が電源となるＶＣＣ(＋５Ｖ)で，１番右がGNDです。
USBのケーブルを必要な長さで切断して，テスターで確認しながら，回路につなぎます。
LED，抵抗はユニバーサル基板に配置して，それぞれ接続します。

STEP④　完成図はこのようになります。







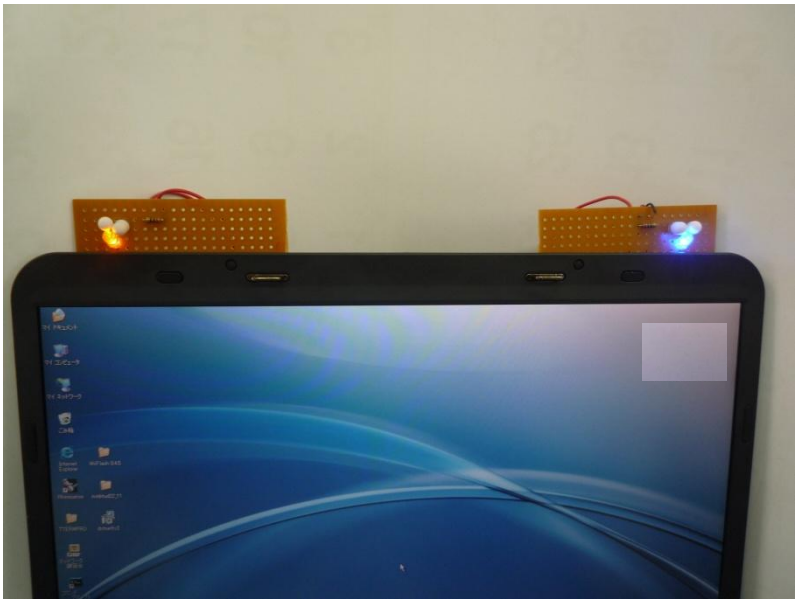
課題11. Wii リモコンの赤外線センサを用いて、マウスを操作しよう。

本時の目標

1. Wii リモコンの赤外線センサによってマウスを操作できる。

Wii リモコンの赤外線センサを用いて、すなわち Wii リモコンの傾きによってポインタ（マウス）を左右に操作させてみましょう。Wii リモコンを Flash において使用するには前記にあった WiiFlash の Actionscript ライブラリを用います。

STEP① 10章で製作した赤外線センサバーをパソコンの上部にセッティングします。5V 電源はパソコンの USB 端子から供給します。

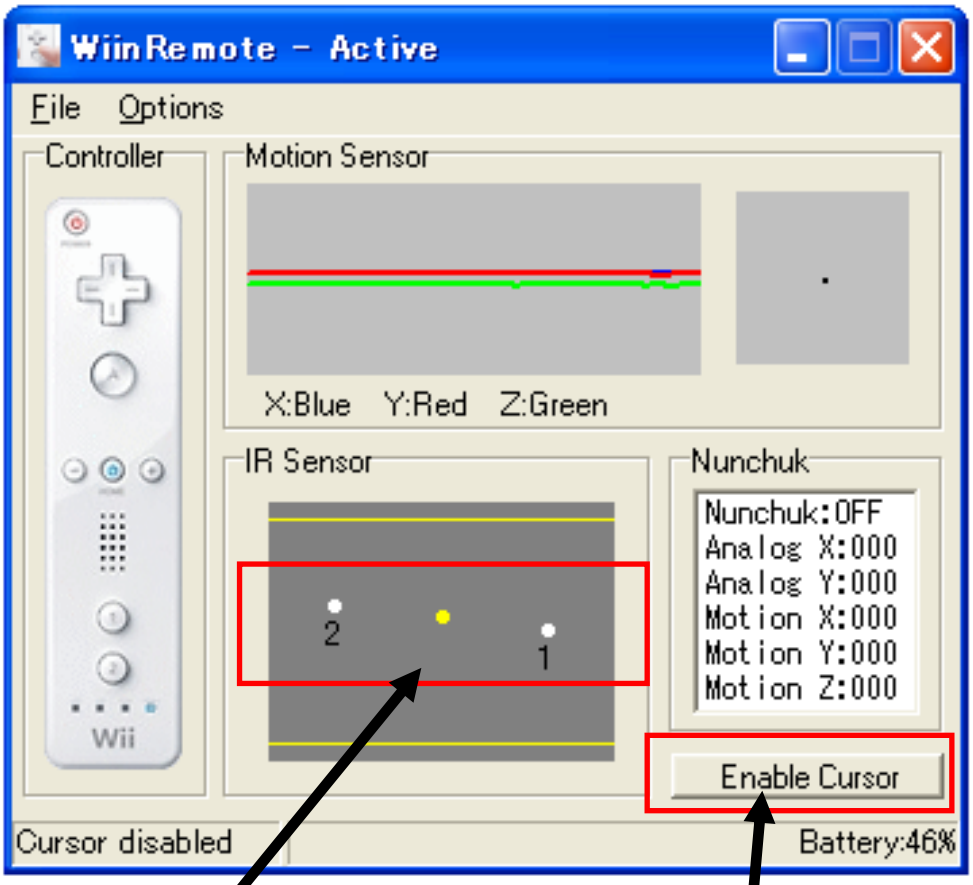


STEP② Bluetooth設定を行います。

8 章Buletoothの設定を参考にしてWiiリモコンとの接続を行います。左図が接続されている状態になります。

STEP③フリーソフト「WiinRemote」を実行します。

「WiinRemote. exe」ファイルを実行します。実行画面においてIRセンサーが認識されているのがわかります。



白い丸が赤外線発光となる。黄色い丸がカーソルの位置となる。

Wiiリモコンの背面にあるBボタンを押すと、Enable Cursorが押され、センサーがカーソルと連動します。

STEP④ これで、Wiiリモコンの赤外線センサによるカーソルの操作ができます。あとは6章ムービークリップのマウス制御を用いていきます。

課題１２．Wii リモコンの加速度センサを用いて、かごを操作しよう。

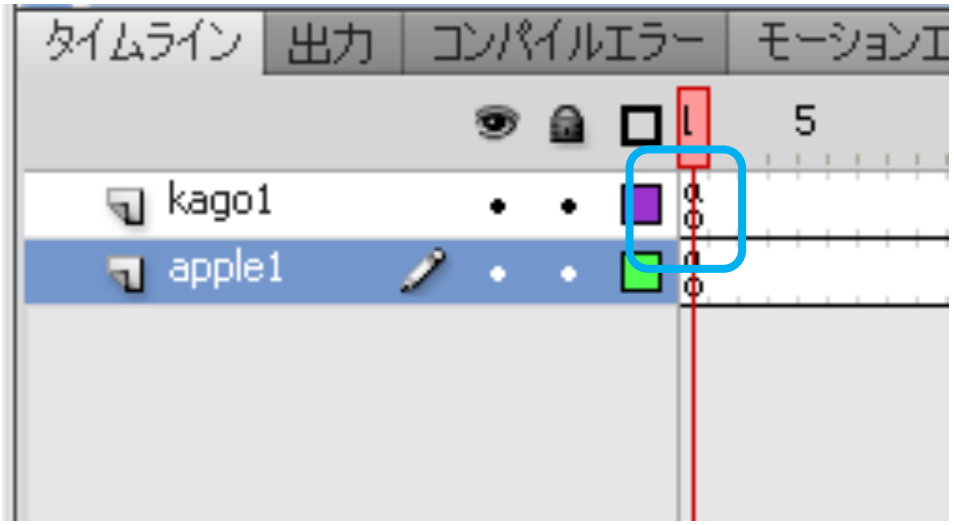
本時の目標

１．ムービークリップを Wii リモコンで操作できるように Actionscript **でプログラムできる。**

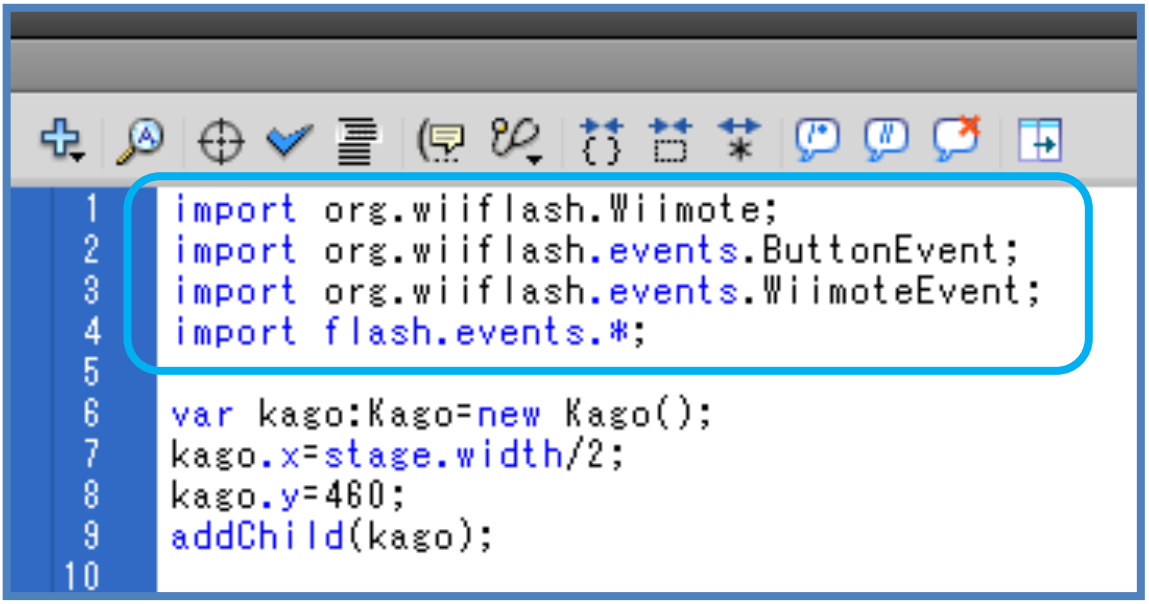
Wii リモコンの加速度センサを用いて、すなわち Wii リモコンの傾きによってかごを左右に操作させてみましょう。Wii リモコンを Flash **において使用するには前記にあった** WiiFlash **の** Actionscript **ライブラリを用います。**

STEP① WiiFlash **ライブラリのあるフォルダ「**org**」を本ファイル「**game-1.fla**」と同じフォルダ内に置きます。フォルダ「**org**」は先生から配布してもらって下さい。**

STEP② **課題4のSTEP⑤と同様に，レイヤー「kago1」の第1フレームにカーソルをあてます。**



STEP③ **レイヤ「kago1」の第1フレームのアクションパネルを開き，使用するActionscriptライブラリを** import **していきます。次のように記述します。**



```
1  import org.wiiflash.Wiimote;
2  import org.wiiflash.events.ButtonEvent;
3  import org.wiiflash.events.WiimoteEvent;
4  import flash.events.*;
5
6  var kago:Kago=new Kago();
7  kago.x=stage.width/2;
8  kago.y=460;
9  addChild(kago);
10
```

1行目から4行目までを挿入記述します。これはフォルダ「org」内にあるAction script文を取り込み、Wiiリモコンが Flash **で使用できるようになります。**
次ページは「Wiimote. as**」ファイルの一部です。「Wiimote」ファイル、「Button Event」ファイル、「WiimoteEvent」ファイルの内容は別紙参考資料（A4プリント）にて確認して下さい。**

「Wiimote.as」ファイルの一部

```
21  */
22  package org.wiiflash
23  {
24      import flash.events.Event;
25      import flash.events.EventDispatcher;
26      import flash.events.IEventDispatcher;
27      import flash.events.IOErrorEvent;
28      import flash.geom.Point;
29      import flash.utils.ByteArray;
30
31      import org.wiiflash.events.*;
32
33      /**
34       * Dispatched when a Wiimote object has successfully connected to the WiiFlash server.
35       *
36       * @eventType flash.events.Event.CONNECT
37       */
38      [Event(name='connect', type='flash.events.Event')]
39
40      /**
41       * Dispatched when a Wiimote object could not establish a connection.
42       *
43       * @eventType flash.events.IOErrorEvent.IO_ERROR
44       */
45      [Event(name='ioError', type='flash.events.IOErrorEvent')]
46
47      /**
48       * Dispatched when Wiimote data has been updated.
49       *
50       * @eventType org.wiiflash.events.WiimoteEvent.UPDATE
51       */
52      [Event(name='update', type='org.wiiflash.events.WiimoteEvent')]
53
54      /**
55       * Dispatched when Nunchuk has been connected to Wiimote.
56       *
57       * @eventType org.wiiflash.events.WiimoteEvent.NUNCHUK_CONNECT
58       */
59      [Event(name='nunchukConnect', type='org.wiiflash.events.WiimoteEvent')]
60
61      /**
62       * Dispatched when Nunchuk has been disconnected from Wiimote.
63       *
64       * @eventType org.wiiflash.events.WiimoteEvent.NUNCHUK_DISCONNECT
65       */
66      [Event(name='nunchukDisconnect', type='org.wiiflash.events.WiimoteEvent')]
67
68      /**
69       * Dispatched when the Classic Controller has been connected to Wiimote.
70       *
```

STEP④ **次に** Wii **リモコンの接続プログラムを記述します。**STEP③**の続きから記述します。**

```
var kago:Kago=new Kago();
kago.x=stage.width/2;
kago.y=460;
addChild(kago);

// create a new Wiimote
var myWiimote:Wiimote = new Wiimote();

// connect wiimote to WiiFlash Server
myWiimote.connect ();
var mySynchronize:Synchronize = new Synchronize();
addChild( mySynchronize );
stage.addEventListener ( MouseEvent.CLICK, onClick );

// makes the wiimote rumble
function onClick ( pEvt:MouseEvent ):void
{   myWiimote.rumbleTimeout = 1000;
}
mySynchronize.x = (stage.stageWidth - mySynchronize.width) / 2;
mySynchronize.y = (stage.stageHeight - mySynchronize.height) / 2

// register Wiimote button events
myWiimote.addEventListener( Event.CONNECT, onWiimoteConnect );
myWiimote.addEventListener( IOErrorEvent.IO_ERROR, onWiimoteConnectError );
myWiimote.addEventListener( Event.CLOSE, onCloseConnection );
myWiimote.addEventListener( WiimoteEvent.UPDATE, onUpdated );

function onCloseConnection ( pEvent:Event ):void
{   mySynchronize.info_text = "Connection closed, WiiFlash Server has been closed";
    addChild( mySynchronize );
}

function onWiimoteConnectError ( pEvent:IOErrorEvent ):void
{   mySynchronize.info_txt.text = "Couldn't connect, make sure WiiFlash Server is running !";
}

function onWiimoteConnect ( pEvent:Event ):void
{   removeChild ( mySynchronize );
    info_txt.text = "Wiimote successfully connected";
}
```

STEP③の
続きから記述

Wii **リモコンが接続された場合、接続エラーが出た場合にはテキストフィールドに緑文字部分が表示され、接続状態を確認できます。**
今回利用した Actionscript **は** Actionscript **ファイル「**Wiimote.as**」、「**ButtonEvent.as**」「**WiimoteEvent.as**」とフォルダ「**events**」になります。これらを** import **することにより、**6 **行目以降の命令文が使用できることとなります。**

STEP⑤ Wii リモコンに関する記述とかごを操作する記述を行います。STEP④に続けて，47行目から50行目までを記述します。

```
42  function onWiimoteConnect ( pEvent:Event ):void
43  {   removeChild ( mySynchronize );
44      info_txt.text = "Wiimote successfully connected";
45  }
46
47  function onUpdated ( pEvt:WiimoteEvent ):void
48  {   if(pEvt.target.roll >0.1){kago.x +=8;}
49      else if(pEvt.target.roll <-0.1){kago.x -=8;}
50  }
51
52
```

STEP④の続きから記述

47行目：WiimoteEvent
48行目～50行目：Wiiリモコンの右傾きroll値が0.1以上でかごを右に移動、左傾きroll値が0.1以上でかごを左に移動させる。

課題13. ActionScriptライブラリの活用

本時の目標

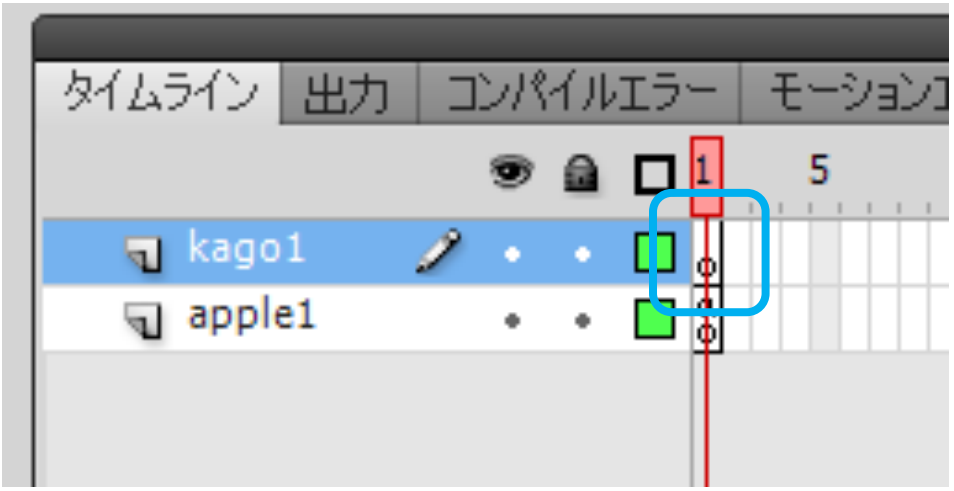
1. ActionScriptライブラリを活用できる。

Actionscript ライブラリについて説明します。ライブラリを利用するメリットとしては

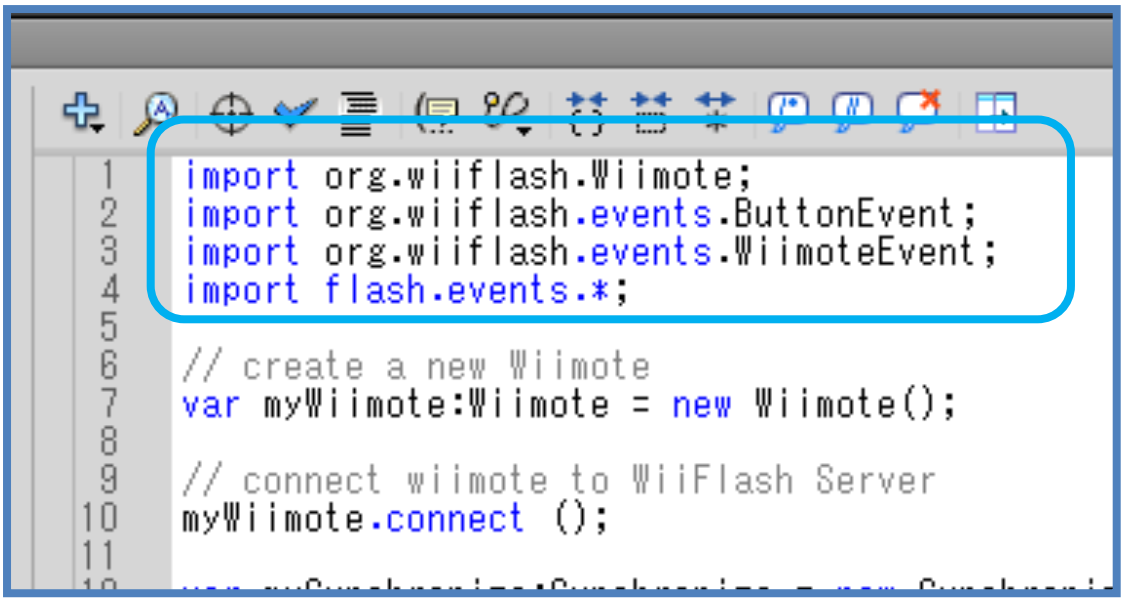
1. 共有できて、みんなで活用できる。
   課題研究においてはグループにおいて別々の作業を行い、あとでライブラリによりグループ化することができます。
2. 難しい内容も取り入れることができる。
   今回の Wii リモコンの操作なども一からプログラムを作成すると大変な作業、高度な作業となるがライブラリ利用することにより、センサの組み込まれた Wii リモコンを学習しつつ、Flash ゲームの入力装置として使用できた。

これにより学習内容が深まり、応用を広げることができます。

STEP① 課題4のSTEP⑤と同様に，レイヤー「kago1」の第2フレームにカーソルをあてます。



STEP③　アクションパネルを開き，使用するActionscriptライブラリをimportしていきます。次のように記述します。



1行目から4行目までを記述します。

STEP④　今回利用したActionscriptはActionscriptファイル「Wiimote.as」、「ButtonEvent.as」「WiimoteEvent.as」とフォルダ「events」になります。これらをimportすることにより、6行目以降の命令文が使用できることとなります。

「Wiimote. as」ファイル一部です。

```
21  */
22  package org.wiiflash
23  {
24      import flash.events.Event;
25      import flash.events.EventDispatcher;
26      import flash.events.IEventDispatcher;
27      import flash.events.IOErrorEvent;
28      import flash.geom.Point;
29      import flash.utils.ByteArray;
30
31      import org.wiiflash.events.*;
32
33      /**
34       * Dispatched when a Wiimote object has successfully connected to the WiiFlash server.
35       *
36       * @eventType flash.events.Event.CONNECT
37       */
38      [Event(name='connect', type='flash.events.Event')]
39
40      /**
41       * Dispatched when a Wiimote object could not establish a connection.
42       *
43       * @eventType flash.events.IOErrorEvent.IO_ERROR
44       */
45      [Event(name='ioError', type='flash.events.IOErrorEvent')]
46
47      /**
48       * Dispatched when Wiimote data has been updated.
49       *
50       * @eventType org.wiiflash.events.WiimoteEvent.UPDATE
51       */
52      [Event(name='update', type='org.wiiflash.events.WiimoteEvent')]
53
54      /**
55       * Dispatched when Nunchuk has been connected to Wiimote.
56       *
57       * @eventType org.wiiflash.events.WiimoteEvent.NUNCHUK_CONNECT
58       */
59      [Event(name='nunchukConnect', type='org.wiiflash.events.WiimoteEvent')]
60
61      /**
62       * Dispatched when Nunchuk has been disconnected from Wiimote.
63       *
64       * @eventType org.wiiflash.events.WiimoteEvent.NUNCHUK_DISCONNECT
65       */
66      [Event(name='nunchukDisconnect', type='org.wiiflash.events.WiimoteEvent')]
67
68      /**
69       * Dispatched when the Classic Controller has been connected to Wiimote.
70       *
```

STEP⑤　Wii **リモコンに関する記述とかごを操作する記述を行います。**STEP③**に続けて，6行目から14行目までを記述します。**

```
 6  var myWiimote:Wiimote = new Wiimote();
 7  myWiimote.connect ();
 8  var mySynchronize:Synchronize = new Synchronize();
 9  addChild( mySynchronize );
10  function onUpdated ( pEvt:WiimoteEvent ):void
11  {
12      if(pEvt.target.roll >0.1){kago.x +=1;}
13      else if(pEvt.target.roll <-0.1){kago.x -=1;}
14  }
15
16
```

6行目：変数 myWiimote **の宣言**
7行目：Wii **リモコンとの接続**
8行目：変数 mySynchronize **の宣言（**WiiFlash **使用に必要な宣言）**
9行目：画面に表示。
10行目：WiimoteEvent
11行目～14行目：Wii**リモコンの右傾き** roll **値が** 0.1 **以上でかごを右に移動、左傾き** roll **値が** 0.1 **以上でかごを左に移動させる。**

STEP⑥ Actionscript **ライブラリについて説明します。ライブラリを利用するメリットとしては**

1. **共有できて、みんなで活用できる。**
   課題研究においてはグループにおいて別々の作業を行い、あとでライブラリによりグループ化することができます。
2. **難しい内容も取り入れることができる。**
   今回の Wii **リモコンの操作なども一からプログラムを作成すると大変な作業、高度な作業となるがライブラリ利用することにより、センサの組み込まれた** Wii **リモコンを学習しつつ、**Flash **ゲームの入力装置として使用できた。これにより学習内容が深まり、応用を広げることができます。**