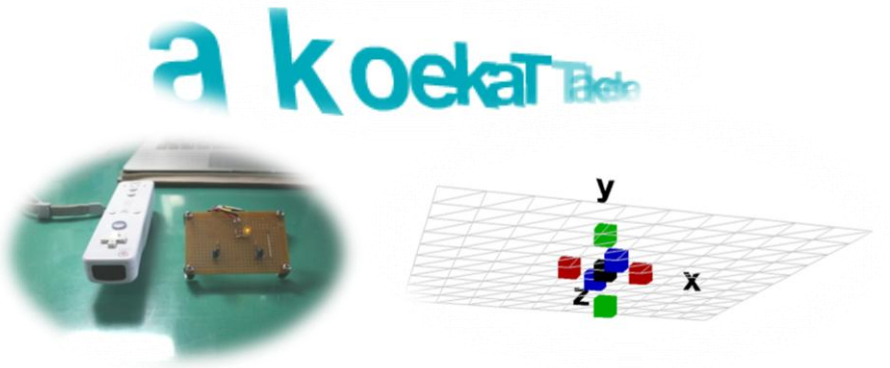
論文番号　２０１２－分２－３

平成２４年度全国情報技術教育研究会

第４１回全国大会（新潟大会）

ActionScript3.0による
プログラミング



期　日　平成２４年８月９日（木）～１０（金）
場　所　長岡市　シティホールプラザ「アオーレ長岡」

鹿児島県立武岡台高等学校
情報科学科　小 田　譲 二

1　はじめに

1.1　目的

ActionScript は Flash のアニメーション制御ツールとして生まれ，現在は ActionScript3.0 となり,Flash コンテンツやコンピュータなどのアプリケーション作成が可能になっている。

本校情報科学科は実習や課題研究のテーマとしてプログラミングを選択することも多い。ホームページでも広く活用されている Flash コンテンツをプログラミングだけで開発することで，プログラミング学習の幅を広げることを目的としている。

1.2　ActionSCript3.0 について

ActionScript とは，Flash アニメーションを制御するために用意されたスクリプト言語である。Flash の普及により機能の拡張がなされ，本格的なアプリケーションの開発が可能な ActionScript3.0（以下 AS3.0）となり，FalahPlayer9 以降で実行可能なスクリプト言語となった。ActionScript1.0 から 2.0 を経て，大きく見直され，パフォーマンス向上やオブジェクト指向の強化が図られている。Flash コンテンツだけでなく，AdobeAIR によって iOS や Android などのモバイル端末でのアプリケーションの開発も可能である。

1.3　開発環境

Flash CS，Flex Builder および FlashDevelop などがある。実習や課題研究などで使用することを考えると，オープンソースである FlashDevelop が最適であると考える。

1.4　他言語との比較

現在，AS3.0 と比較できる言語には C#，JAVA および HTML5 などがある。特に HTML5 においては，Flash にとって代わるものとしてインタラクティブな Web アプリケーションの開発が可能となってきている。

2　開発準備

2.1　FlashDevelop のライブラリ設定

今回使用したのは，「FlashDevelop-4.0.1-RTM」（http://www.flashdevelop.org/）である。フリーソフトであり，オープンソースの ActionScript 開発環境のひとつである。

ライブラリは，ActionScript3.0 のプログラミングを補助するコードである。例えば，三次元表現には高度な数学の知識が必要となるが，今回使用する Papervision3D ライブラリを導入すれば比較的簡単にその処理が可能である。必要に応じて AS3.0 のライブラリをそれぞれフォルダに保存する。ライブラリをそれぞれ準備できたら，「ツール」→「グローバルクラスパスの編集」をクリックして，図 2.1.1 のようにライブラリを入れてあるフォルダのパスを設定し，開発環境を再起動して完了となる。ここで注意すべきことは，実際に import 文を入力したときうまくパスが通っているかどうかである。図 2.1.1 のようにコード補完にて登録されたライブラリが出てくるので確認はとれるが，バージョンによってクラス構成やフォルダ構成が変化している場合があるので，必ず確認する。

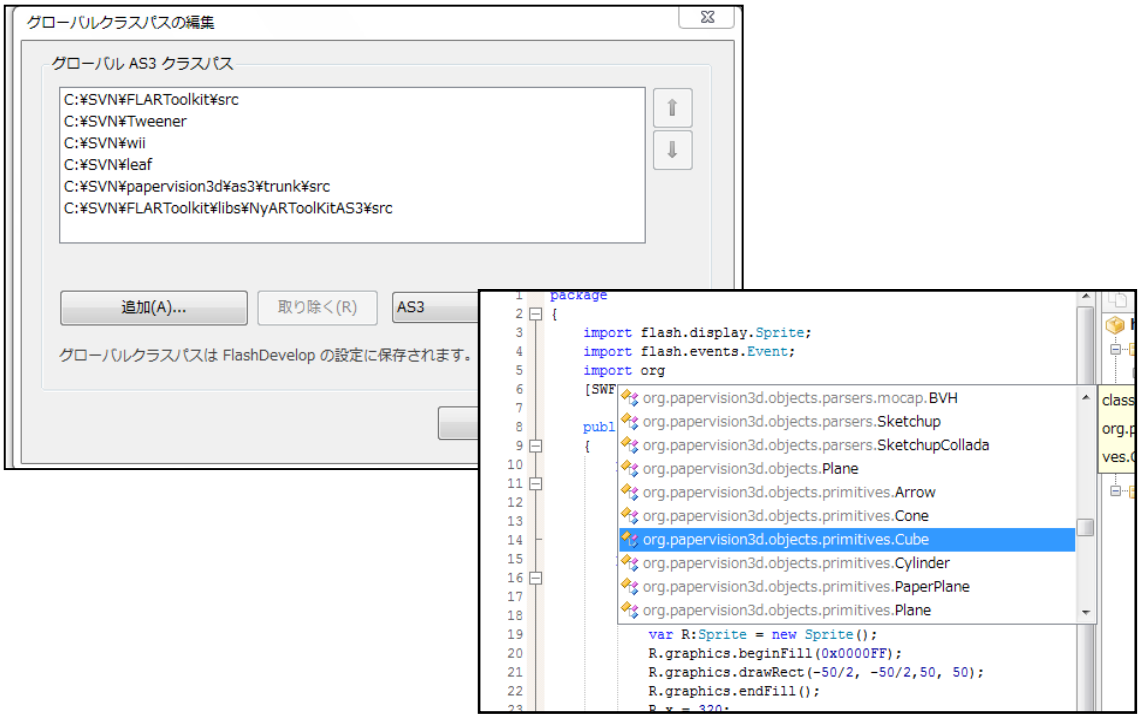


図 2.1.1　グローバルパスの設定とコード補完

2.2　テストプログラム

「新規プロジェクト」→「AS3 Project」と選択して，「プロジェクトフォルダーを選択」にチェックを入れて，「プロジェクト名」を入力して「OK」を押す。自動的に『Main.as』ファイルが作成され，必要なものが生成される。実際に，1を押して実行すると，ウィンドウが立ち上がる。次にテストプログラムとして「Test.as」で正方形を作成してみる。

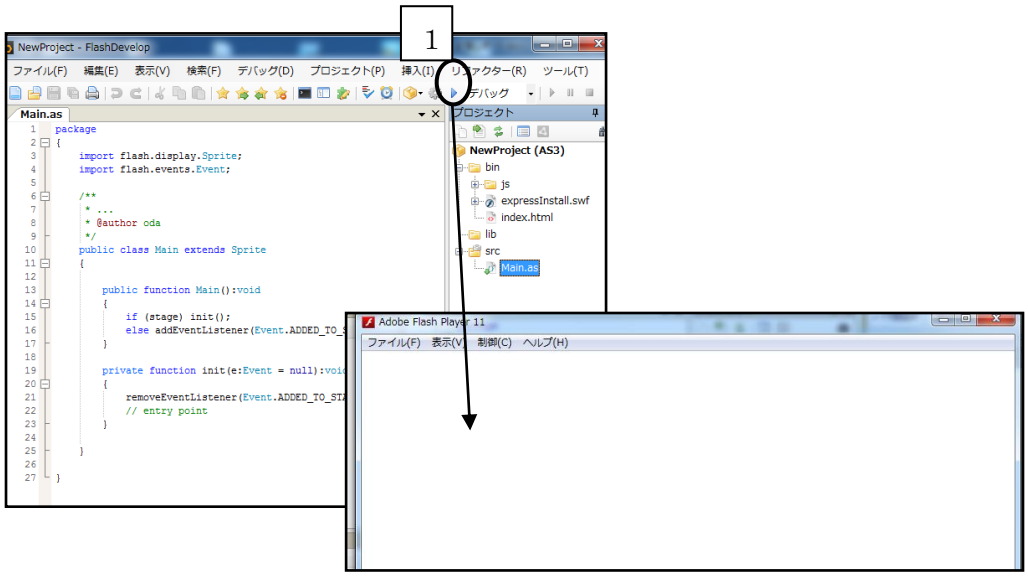


図 2.2.1　FlashDevelop の開発画面

```
package
{
    import flash.display.Sprite;
    import flash.events.Event;
    [SWF(width = "640", height = "480", backgroundColor = "#FFFFFF", frameRate = "30")]
    public class Test extends Sprite
    {
          public function Test():void
          {
            if (stage) init();
              else addEventListener(Event.ADDED_TO_STAGE, init);
          }
          private function init(e:Event = null):void
          {
              removeEventListener(Event.ADDED_TO_STAGE, init);
              // entry point
              var hako:Sprite = new Sprite();
              hako.graphics.beginFill(0x0000FF);
              hako.graphics.drawRect(-25, -25,50, 50);
              hako.graphics.endFill();
              hako.x = 320;
              hako.y = 240;
              addChild(hako);
          }
    }
}
```

図 2.2.2　Test.as

点線部分には，Sprite クラスからインスタンスを作成し graphics プロパティにて正方形を作成している。画面上に表示をするには addChild にてそのオブジェクトを指定する必要がある。図 2.2.3 には「Test.as」の実行結果と基本座標を示す。

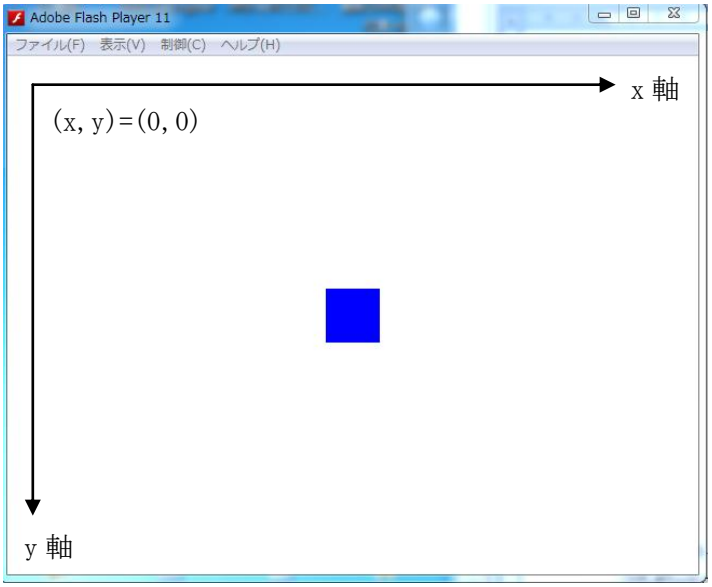


図 2.2.3　Test.as の実行結果

次の図 2.2.4 のプログラムは，上記のオブジェクトをキーボードの矢印で動かせるようにしてみたものである。キーボードからの入力を受け付けるには，イベント処理を使う。

```
stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDownHandler);
```

```
private function onKeyDownHandler(e:KeyboardEvent):void
{
          switch (e.keyCode)
          {
                    case Keyboard.LEFT:
                              hako.x -= 5;
                              break;
                    case Keyboard.RIGHT:
                              hako.x += 5;
                              break;
                    case Keyboard.UP:
                              hako.y-= 5;
                              break;
                    case Keyboard.DOWN:
                              hako.y +=5;
                              break;
                    default:
                              break;
          }
}
```

図 2.2.4　キーボードイベント

3　ライブラリ

3.1　Papervision3D

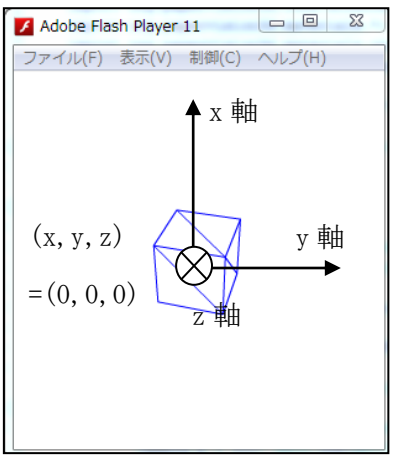
3.1.1　Papervison3D とは

ライブラリを用いることによって三次元表現の手間が省くことができる。三次元表現のライブラリはいくつか存在するが一番資料が豊富で，なじみやすいのは「Papervision3D」（以下 PV3D）である。現在は「PV3D」は開発が進んでいるのか不明であるが，別の「Away3D」というライブラリは「PV3D」から派生したもので，今後ライブラリを変えたとしても十分対応できると考える。また，「PV3D」と別のライブラリの連携についてもいろいろできることは多く，三次元表現の幅を広げている。いくつかサンプルプログラムを示し，他のライブラリとの連携も試してみる。

3.1.2　画面表示の概念

```
var cube:Cube;
var material:WireframeMaterial = new WireframeMaterial(0x0000FF);
var materials:MaterialsList = new MaterialsList( { all:material } );
cube = new Cube(materials, 200, 200, 200, 1, 1, 1);
scene.addChild(cube);
```

```
var material:ColorMaterial = new ColorMaterial(0x0000FF);
```



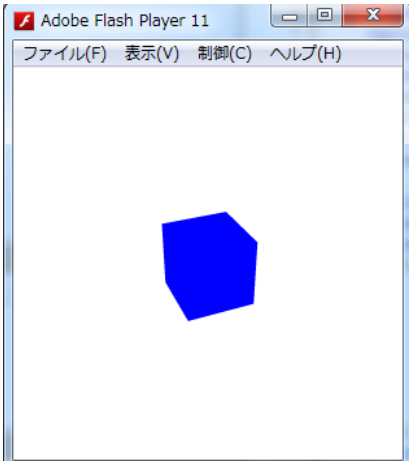


図 3.1.1　直方体の表示（上プログラム：左図はワイヤーフレーム
下プログラム（変更部分のみ）：右図はカラーマテリアル）

図 3.1.1 は PV3D で作った直方体である。プログラムの仕組みは，図 3.1.2 に示している。三次元空間に配置されているオブジェクトを，カメラを通して Flash ステージにビューポートを通して表示することになる。そのため座標軸が Flash とは違うことになる。その座標軸は図 3.1.1 のようになる。座標軸の中心は，画面の左上ではなく画面の中心になる。また，画面から奥に向かって z 軸が伸びていることになる。

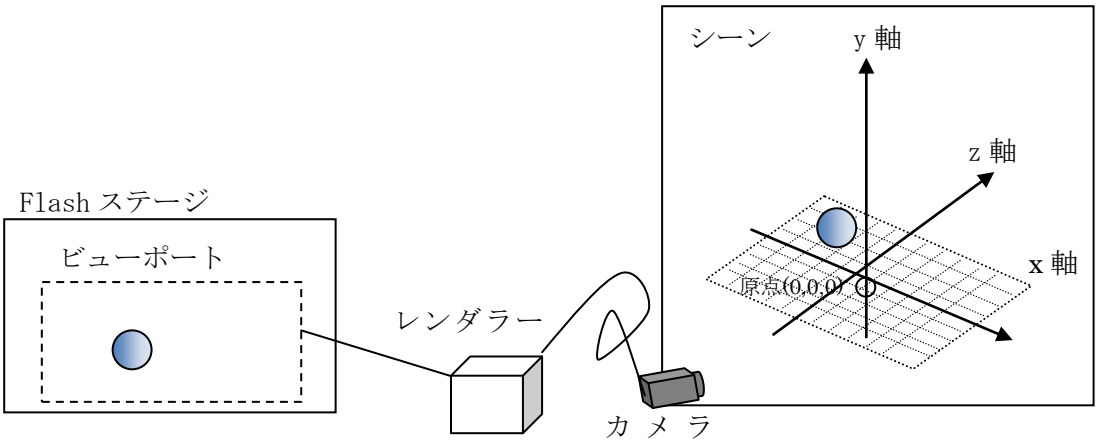
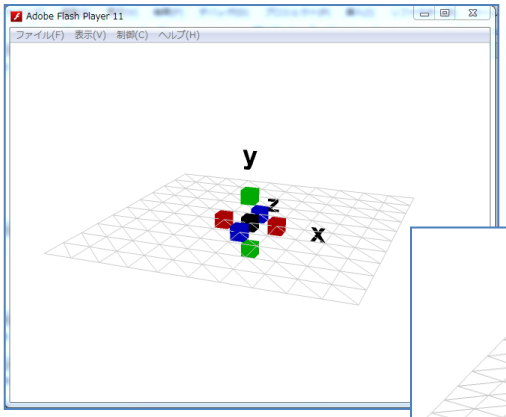


図 3.1.2　Papervision3D の表示概念

2.2 のテストプログラムで作った正方形（図 2.2.2）との違いを明確にするため，図 3.1.3 に示すように，直方体 7 個配置し，回転させてみるプログラムを作成した。




```
var word0:Text3D = new Text3D("z", font, mat);
var word1:Text3D = new Text3D("x", font, mat);
var word2:Text3D = new Text3D("y", font, mat);
word0.z = 50;
word1.x = 50;
word2.y = 50;
```

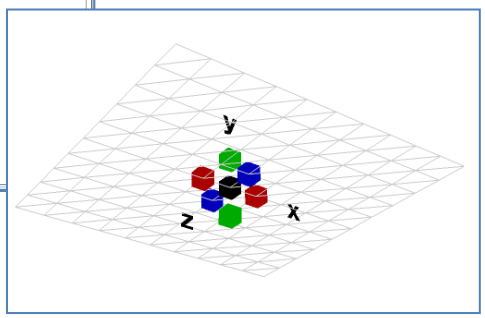


図 3.1.3　座標の確認プログラム

三次元表現を行うにあたってプログラムを作成するためには，PV3D では準備が必要である。上記で示したプログラムは BasicView クラスを継承したものと，Sprite クラスを継承したものでそれぞれ作成している。BasicView クラスは PV3D の初期化設定を省くことができ，すぐに三次元表現を始めることができる。Sprite クラスを利用する場合は，ステージ，三次元空間およびカメラの初期化などを自分で行わなければならない。

```
public function Main() {
                        init();
}
private function init():void
{
        setStage();
        setScene();
        setCamera();
        addEventListener(Event.ENTER_FRAME,onEnterFrameHandler);
}
private function setStage():void
{
        stage.quality = StageQuality.MEDIUM;
        stage.scaleMode = StageScaleMode.NO_SCALE;
        stage.align = StageAlign.TOP_LEFT;
        stageW=stage.stageWidth;
```

図 3.1.4　Sprite クラスによる初期化設定の一部

3.1.3　その他

図 3.1.5 は，直方体を 6 個横に並べたものを回転させるプログラムである。図 3.1.1 のプログラムを応用しているが，大きな違いはレンダラーである。図 3.1.1 は BasicView と呼ばれる簡略化したレンダラーによる描画である。図 3.1.5 は見て分かるように鏡面反射するようにした ReflectionView というものを利用している。また，図 3.1.6 は生徒が作った三次元テキストアニメーションの様子である。生徒には，テキストの作成と動かす基本のみを教えたが，自分なりに工夫して面白い動きを作り出している。

図 3.1.5　直方体の回転と鏡面反射

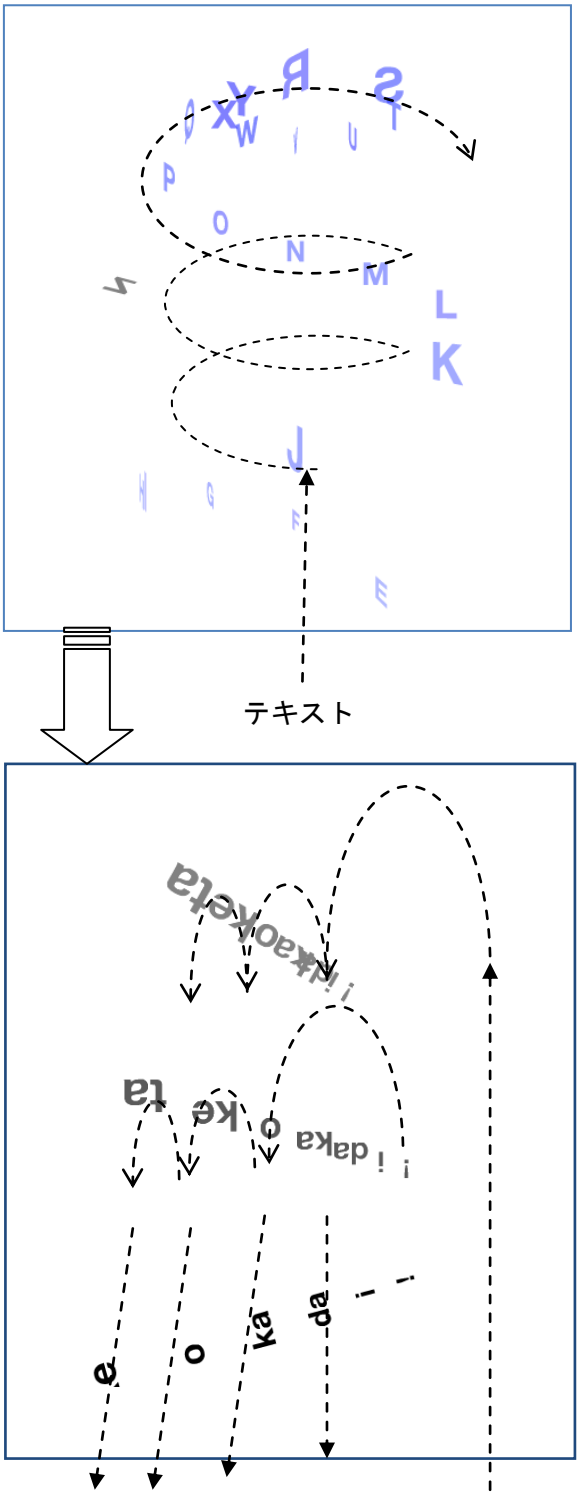


図 3.1.6　三次元テキストの回転とアニメーション（生徒作品）

3.2　ページめくり表現

PV3D では基本的な三次元物体（プリミティブオブジェクト）がいくつか用意されている。代表的なものは，Plane（平面）・Cube（直方体）・Sphere（球体）である。しかし，実際にはさまざまな動きや形状に対応する必要もある。3.1 では上記の Cube を使ってプログラムを組んだが，ここではメッシュ制御を使った形状変化として，本のページがめくれる表現ができないか探して試してみた。現在はめくれるところ（図 3.2.1 参照）までできているので，途中経過のみ紹介する。「ページめくり」の表現には，いくつかクラスやライブラリがあり公開されている。今回はその中で，「CurlingPage」を使うことにした。これは，Lee Felarca 氏のサイトにある「CurlingPlane.as」というクラスである。残念ながら，このクラスは PV3D（1.5）のときのクラスであり，現在の PV3D（2.x）では使用できない。そのため，クラス自体を現在のバージョンに対応させる必要があった。メッシュ制御におけるクラスは PV3D（1.5）では「Mesh3D」，「Face3D」を使ったが，PV3D（2.0）では「TriangleMesh3D」，「Triangle3D」にそれぞれ変更されている。また，それに伴いコンストラクタ引数も違ったので，変更していった。主な変更点のみ図 3.2.2 に示している。

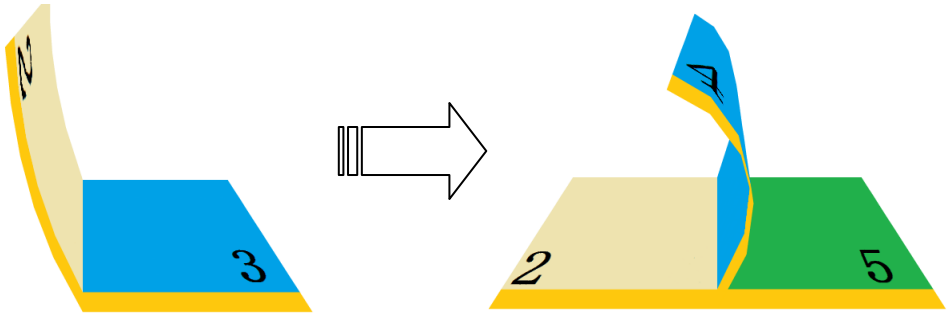


図 3.2.1　ページめくりの動作

```
public class CurlingPlane extends Mesh3D{
```

⇩

```
public class CurlingPlane extends TriangleMesh3D{
```

```
faces.push( new Face3D( [ a, b, c ], null, [ uvA, uvB, uvC ] ) );
```

⇩

```
faces.push(new Triangle3D(this,[a, b, c], null, [uvA, uvB, uvC]));
```

図 3.2.2　「CurlingPlane.as」の主な変更点

3.3　AR（拡張現実）

AR はスマートフォンの普及やゲーム機に導入されるなど目にする機会が増えてきた。最近では Google が AR を使ったメガネを発表したり，カーナビで現実の世界に直接ナビ画面を投影したりするなど注目を集めている。AR 制作のソフトウェアはたくさんあるが，Flash でも AR 用のライブラリが以前からあるので，活用してみた。今回は，ひとつのマーカーを使ったものと，複数のマーカーを使ったプログラムを作成した。また，実際にマーカーに穴が開くような表現を行うプログラムも試してみた。

3.3.1　FLARToolkit ライブラリ

「FLARToolkit」は C 言語用の「ARToolkit」やそれを Java 用に移植した「NyARToolkit」からさらに ActionScript3.0 用に移植した AR 用のライブラリである。マーカーの上に三次元物体を載せることができる。また，PV3D も活用できるので PV3D の知識さえあれば，いろいろな応用プログラムも可能である。

3.3.2　マーカー作成

FLARToolkit で使えるマーカーは正方形で，色の濃い部分の中に明るい部分が中心にあるようなもの（図 3.3.1）である。

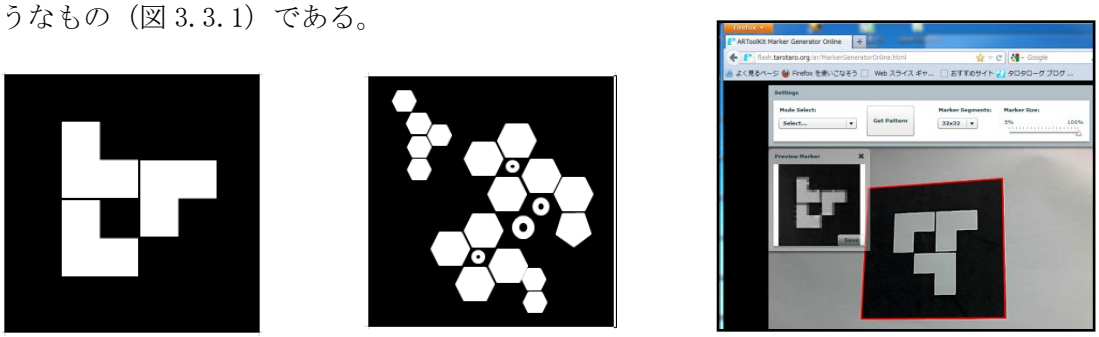


図 3.3.1　自作マーカーと取り込む様子

今回，マーカー作成には Microsoft Office Excel2007 を用いた。図 3.3.2 に簡単な Excel の説明を載せている。これで作ったマーカーを印刷して，「ARToolkit Marker Generator Online」（http://flash.tarotaro.org/ar/MarkerGeneratorOnline.html）ツールを使ってマーカーをデータ化する。Excel ではアドインで，印刷時に長さが 8cm×8cm になるようにしてある。

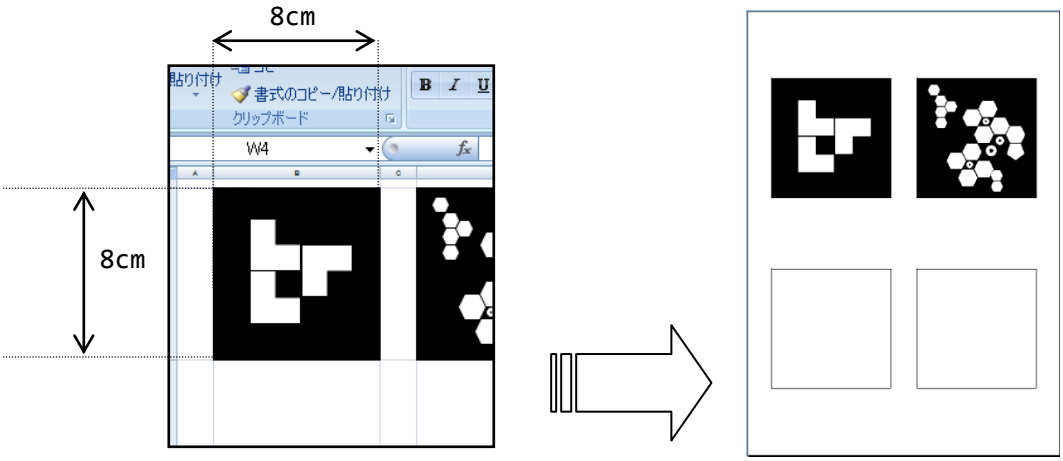
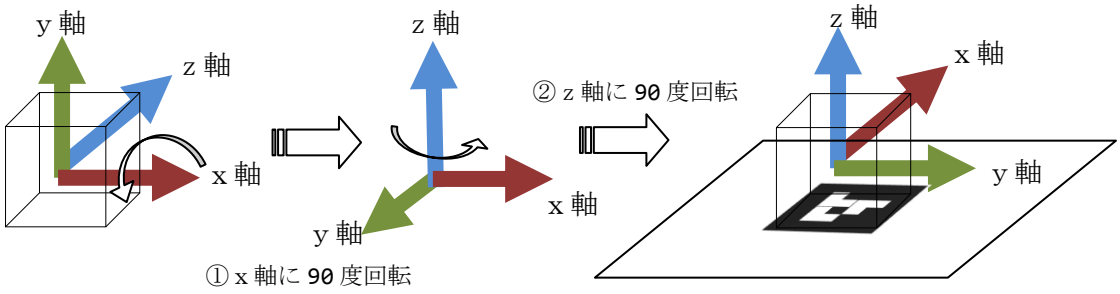


図 3.3.2　Excel によるマーカー作成

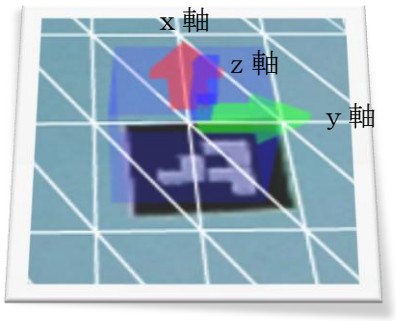
3.3.3　作成したプログラムについて

(1)　オブジェクトの表示と回転

図 3.3.3 に示したものは，PV3D 上の座標系(a)と AR 上の座標系(b)である。PV3D のオブジェクトは AR で表示する際は,図 3.3.3 のように x 軸，z 軸それぞれ 90 度回転させる必要がある。実際にテストプログラムで座標系を再現してみたものが，図 3.3.3(c)である。



(a)　PV3D の座標系　　(b)　AR の座標系



(c)　AR のテストプログラム

図 3.3.3　AR マーカーと座標軸の関係

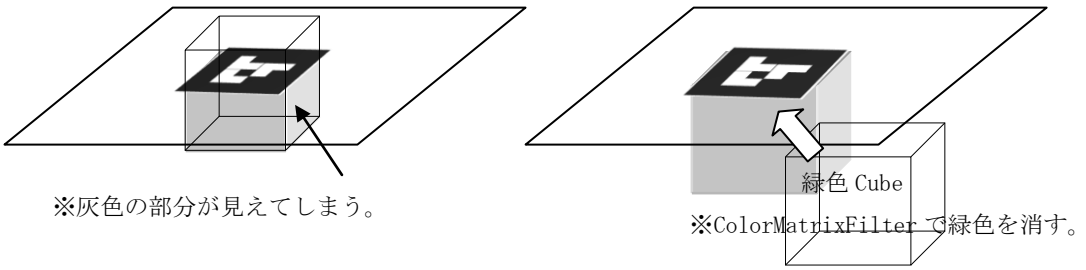
3.3.4　応用プログラム

オブジェクトを表示するだけのプログラムから，マーカー部分に穴をあけるプログラムを試してみる。「HAPPY NEW YEAR '09」（http://09.aid-dcc.com/）のサイトでは年賀状のマーカーが下がっていき，物が飛び出してくる。こうしたプログラムはどのように行われているのか，調べて試してみた。

マーカー部分に穴をあけるためには，２つの工程が必要である。一つ目は，穴をどのように表現するかである。また，穴をあけたときにオブジェクトが透けて見えてしまうことを防ぐことである。

(1)穴の表現

「Saqoosha」サイト（http://saqoo.sh/a/1676）にどのように表現しているかが公開されていた。それを参考に穴をマーカー上にあけてみる。当初，平面を 4 枚用意し穴を表現してみた。これは失敗に終わった。AR でマーカー上に平面を 4 枚並べると図 3.3.5 のようになり，マーカーを動かすと明らかにマーカーの上に載っているようにしか見えない。
そこで，直方体である Cube オブジェクトを作って，上面に穴をあけたものを図 3.3.5(b)のように配置すればよい。さらにこの状態では，まだ灰色部分が見えてしまうのでこの部分を見えなくすれば完成である。消す方法は，動画の合成に使うクロマキのようなものである。ColorMatrixFilter クラスを使えば，特定の色を消すことができる。ColorMatrixFilter は 4 行×5 列の行列で表し，赤・青・緑・アルファ・加算といったものからできている。今回は，緑色を消すようにしている。図 3.3.5(b)のように灰色部分に同じサイズの緑色の Cube をかぶせて,ColorMatrixFilter を適用すると完成である。図 3.3.6 にプログラムのポイントを示している。



(a)　平面 4 枚で表現　　(b)　直方体で表現

図 3.3.5　穴の表現

内側のみ表示する Cube の作成

```
Var hole00:Cube=new Cube(mat1,83,83,40,10,10,10,Cube.ALL-Cube.TOP-Cube.BOTTOM, Cube.TOP)) ;
```

外側を緑色表示する Cube の作成

```
Var hole01:Cube=new Cube(mat1,83,83,40,10,10,10,0, Cube.TOP)) ;
```

緑色を消す ColorMatrixFilter

```
viewport.filters = [
    new ColorMatrixFilter([
          1, 0, 0, 0, 0,
          0, 1, 0, 0, 0,
          0, 0,1, 0, 0,
          1, -1, 1, 1, 0
    ])
];
```

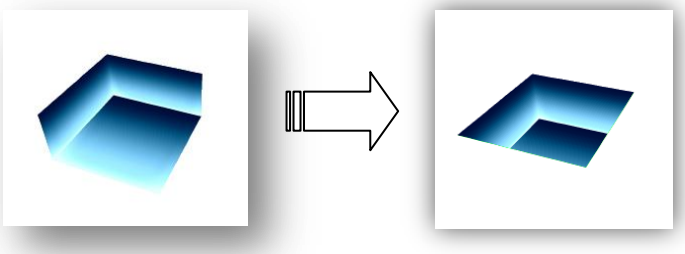
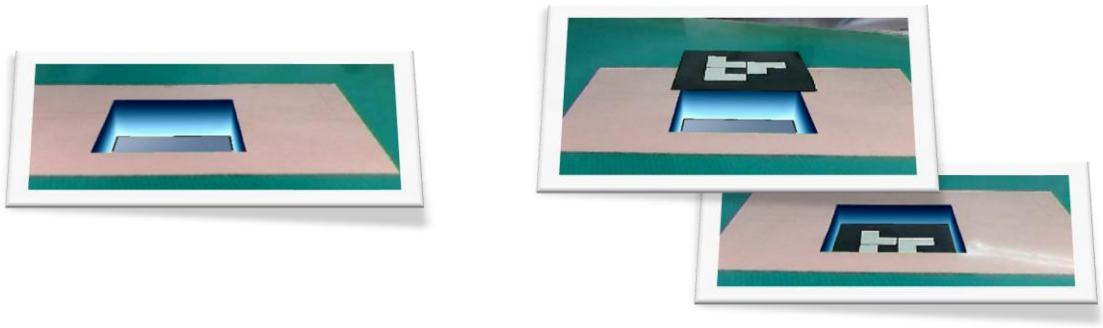


図 3.3.6　穴をあけるためのプログラム



(a)　穴をあけるだけ　　(b)マーカーを切り出す

図 3.3.7　実行結果

さらに，表面のマーカーがその穴に沈むようにしてみる。上記のプログラムで穴をあけることができるが，マーカー自体が沈んだり，動いたりすることで穴がそこに存在している感覚が強くなる（図 3.3.7(b)）。マーカーを動かすためには，ページめくりで使用した Triangle3D クラスを用いて，UV 座標を利用する。このプログラムの詳細は，「gihyo.jp（技術評論社）」（http://gihyo.jp/design/feature/01/flartoolkit/0004）のサイトにて説明がなされている。仕組みとしては，カメラでマーカーを認識する際にキャプチャしている画像をマテリアルとして平面を作成する。キャプチャするところは，ダミーで動かす平面と同じ平面を用意することで行っている。そして，ダミーの面から取得した面を動かす面へと変換させている。

3.4 WiiFlash

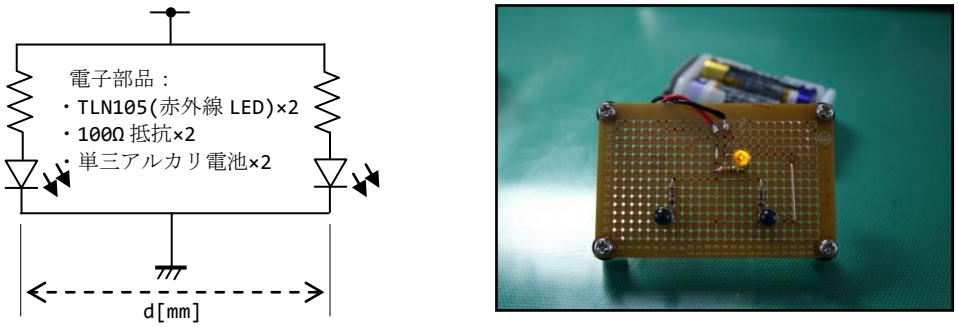
Wii リモコンは任天堂の家庭用ゲーム機 Wii でコントローラとして使われているものである。Bluetooth による接続ができるのでパソコンでも使用できる。WiiFlash ライブラリを用いれば，簡単にパソコンで入力装置として利用することができる。Johnny Chung Lee 氏が Wii リモコンを受信部として使う逆転の発想で作った「WiiDesktopVR」は YouTube で公開され話題になっている。このプログラムは C#，DirectX で作っているが，このようなものを PV3D と WiiFlash で作ってみた。

3.4.1 必要なもの

・Wii リモコン（旧式）

・自作赤外線センサ（図 3.4.1 参照）

・WiiFlashServer

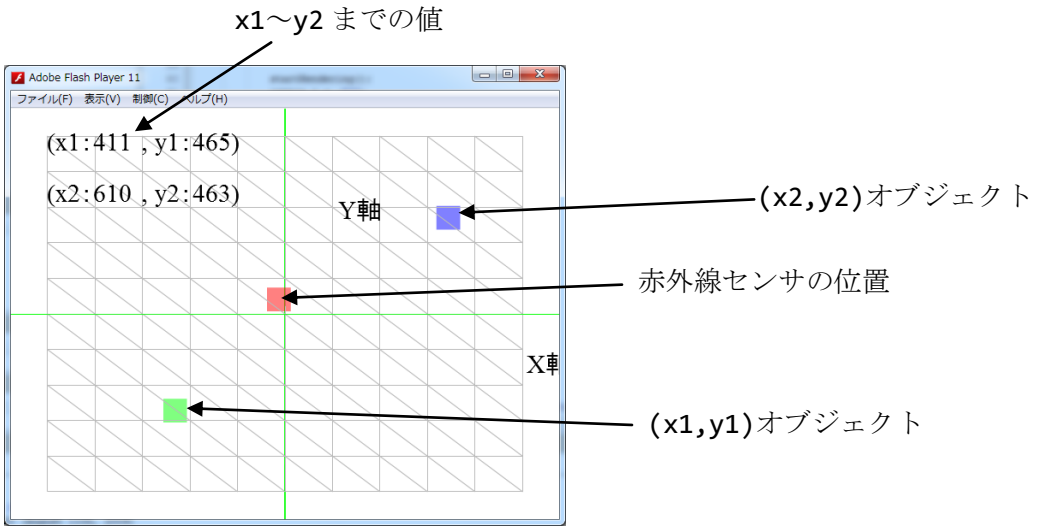


(a)回路図　　(b)実装図

図 3.4.1　自作赤外線センサ

3.4.2 計測プログラム

Wii リモコンを使用するためには，どのように赤外線センサを受け取っているのか知る必要がある。書籍やインターネットなどでも情報があり，仕組みは理解できるが，実際に計測用のプログラムを作り，計測してみた。どのように中心をとらえているのかだいたい理解したうえでプログラムを作成したので，どこに赤外線センサがあるのかを表示するオブジェクトを用意できた。大切なのはどのような値が表示されるのかを計測することである。図 3.4.2 が計測用プログラムの画面である。生徒にも計測用プログラムのひな形である三次元座標を作らせてみた。Wii リモコンの値の代わりにマウスの座標値を使って，計測用プログラムである。図 3.4.3 のようなもので，マウスの位置によって座標が動く。



※(x1，y1)オブジェクトは反転して位置を表示している。

図 3.4.2　計測用プログラム

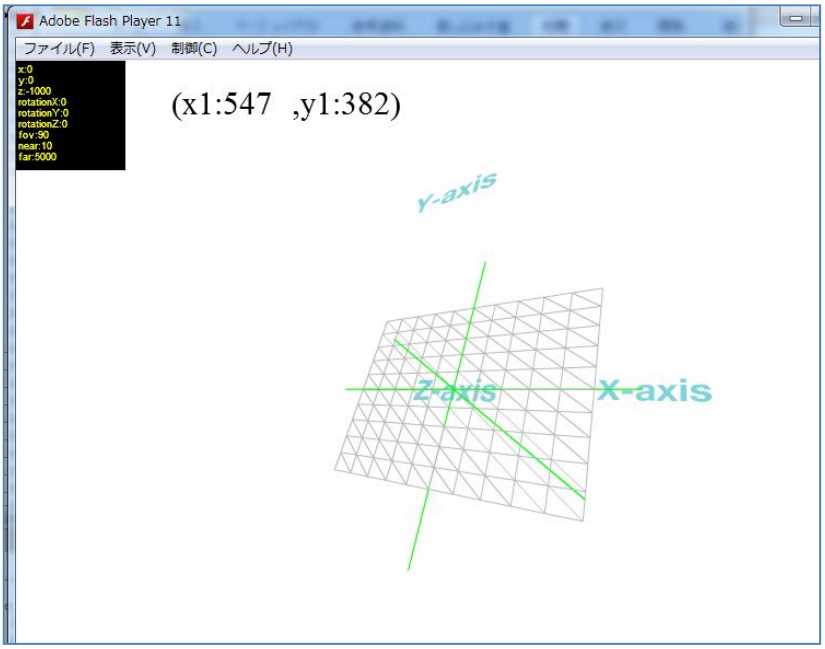


図 3.4.3　計測用プログラム生徒作品

現在，計測を継続中であり生徒は課題研究の一環として，いろいろな条件で計測をしている。図 3.4.4 は実際の計測の様子である。外からの赤外線も取り込むことがあるのでできるだけ，カーテンがある部屋で行っている。

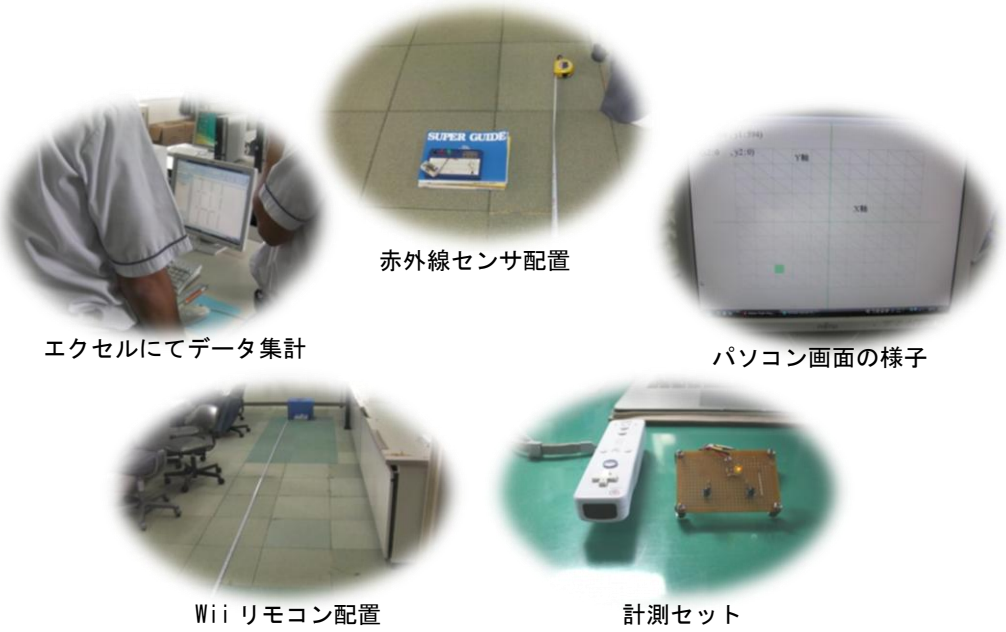


図 3.4.4　計測風景

事前に計測して実験した値やテストプログラムによる内容を示しておく。計測配置図は下図 3.4.5 のようにする。

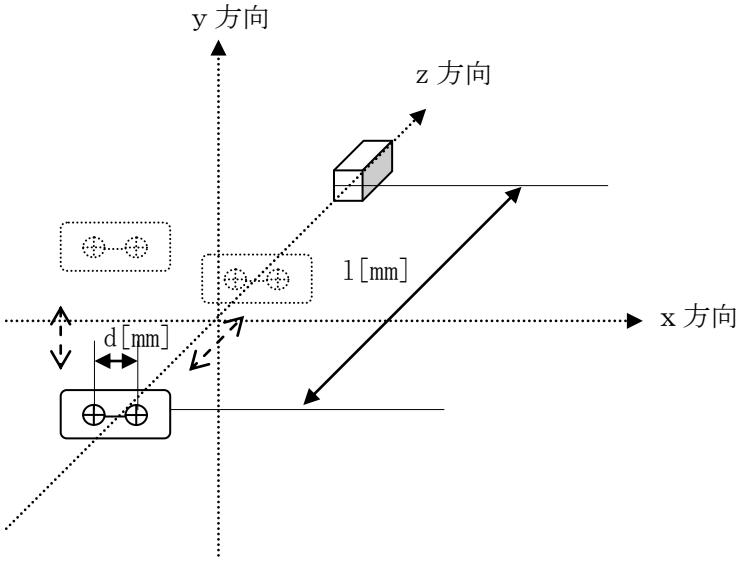
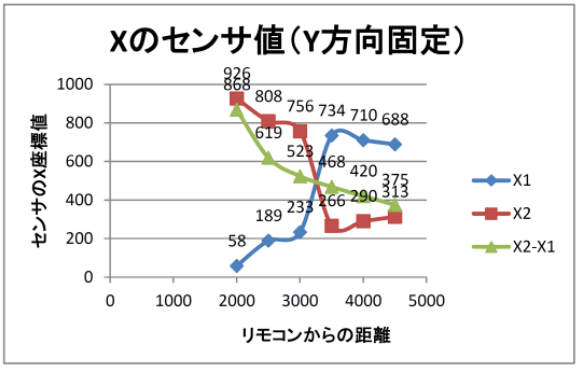


図 3.4.5　計測配置図

赤外線センサの 1 点目，2 点目をそれぞれ（x1, y1），（ｘ2，ｙ2）で 0～1000 の値をとるようにプログラムしてある。

センサーの間隔：1300mm

リモコンからの距離[mm]	X1	X2	X2-X1
1500			
2000	58	926	868
2500	189	808	619
3000	233	756	523
3500	734	266	468
4000	710	290	420
4500	688	313	375



リモコンからの距離[mm]	Y1	Y2	Y2-Y1
1500			
2000	504	500	4
2500	529	480	49
3000	520	486	34
3500	490	521	31
4000	480	518	38
4500	482	523	41

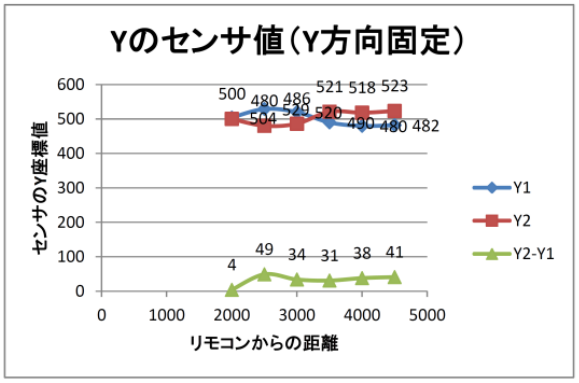
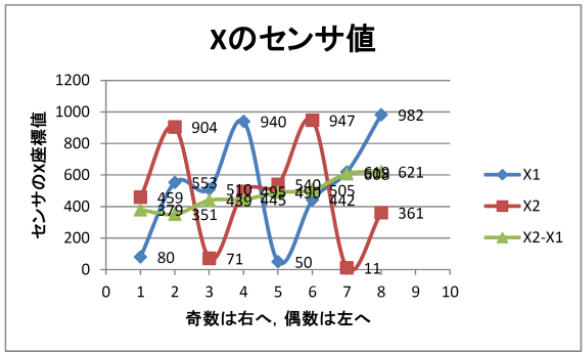


図 3.4.6　ｙ方向を固定したときの，ｚ方向による値

図 3.4.6 から，右と左の赤外線センサの差がｚ方向の位置を表しているようだ。ｙ方向は手によって固定していることと，計測データが少ないことから，データにばらつきがあるが，データ数を増やすことと，しっかり赤外線センサを固定して値をとることでばらつきは減ると考える。上記の計測用センサは赤外線センサの 2 つの位置を 13cm にしてある。生徒用は 3cm で計測させている。センサのそれぞれの位置の値の差をとって，現在の位置を読み取っているようだ。近いほど差は，1000 に近づいている。なお，センサ幅を 13cm にしている場合は，15cm 以下に近づけると計測不能になった。このことから，センサ幅によって計測できる位置が決まってくる。

横軸	リモコンから左右のデータ	X1	X2	X2-X1
1	45cmでの右に1000mm	80	459	379
2	45cmでの左に1000mm	553	904	351
3	40cmでの右に1000mm	510	71	439
4	40cmでの左に1000mm	940	495	445
5	35cmでの右に1000mm	50	540	490
6	35cmでの左に1000mm	442	947	505
7	30cmでの右に1000mm	619	11	608
8	30cmでの左に1000mm	982	361	621



横軸	リモコンから左右のデータ	Y1	Y2	Y2-Y1
1	45cmでの右に1000mm	542	520	22
2	45cmでの左に1000mm	534	510	24
3	40cmでの右に1000mm	470	530	60
4	40cmでの左に1000mm	474	535	61
5	35cmでの右に1000mm	569	450	119
6	35cmでの左に1000mm	570	427	143
7	30cmでの右に1000mm	460	570	110
8	30cmでの左に1000mm	474	553	79

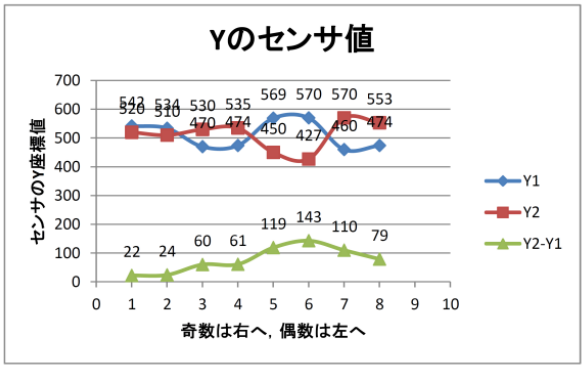


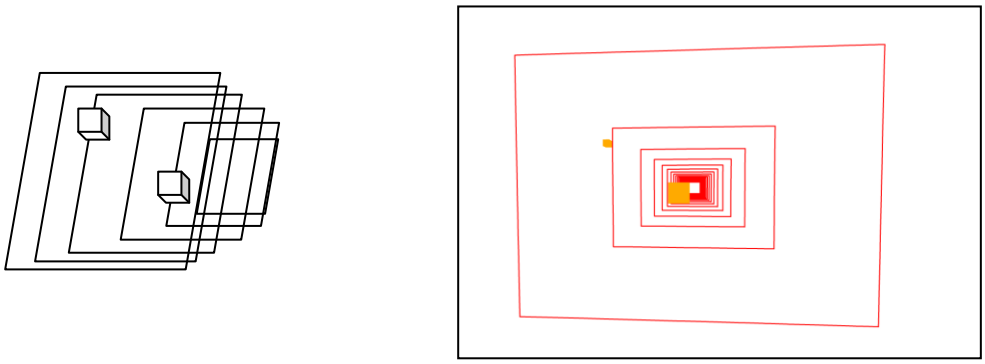
図 3.4.7　センサの値とグラフ（y 方向固定）

図 3.4.7 からは，z 方向の変化における x 方向の値変化を記録したものである。奇数番号が右へ 1000㎜ 動かしたときで，偶数番号が左へ 1000㎜ 動かしたときの値である。z 方向のときと同じように差をとってみた。

今回は，y 方向の変化は計測していないが生徒の課題研究では行う予定である。こうした計測を通じて，どのようにセンサを配置すればよいのか，距離はどのくらいが適切なのかなどを決めていけばよい。

3.4.3　応用プログラム

Wii リモコンを受信部にして，自作した赤外線メガネをかけて画面上の三次元空間が動くものを作成した。メガネをかけて，画面をのぞくとその方向から空間を見ることができる。テストプログラムは，図 3.4.8 のように，正方形を z 方向に並べて，その中にいくつか直方体を配置してみた。



(a)オブジェクト配置　　(b)実際の画面

図 3.4.8　テストプログラム

赤外線センサの動きに応じて，画面が動くことにより空間の中を覗くようにしている。Wii リモコンの位置や傾きなど工夫してできるだけスムーズに動くようにした。

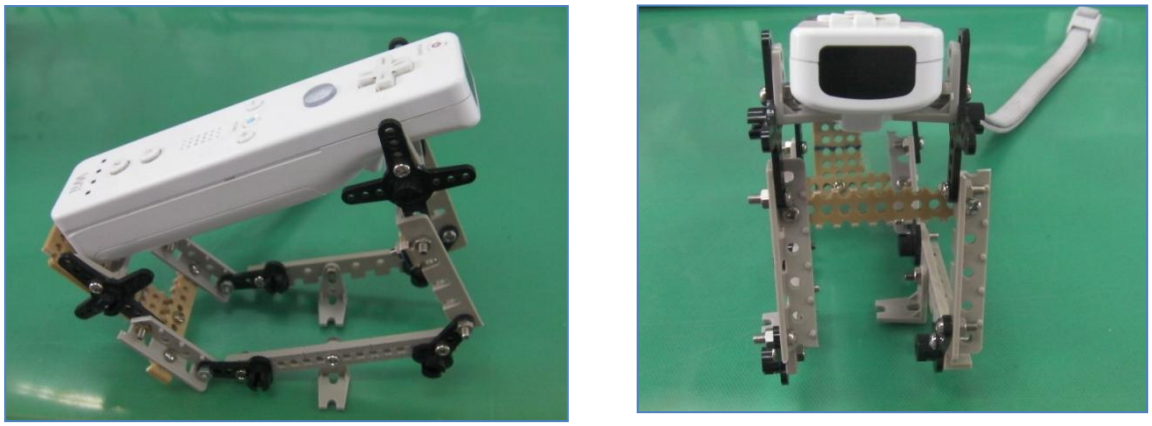
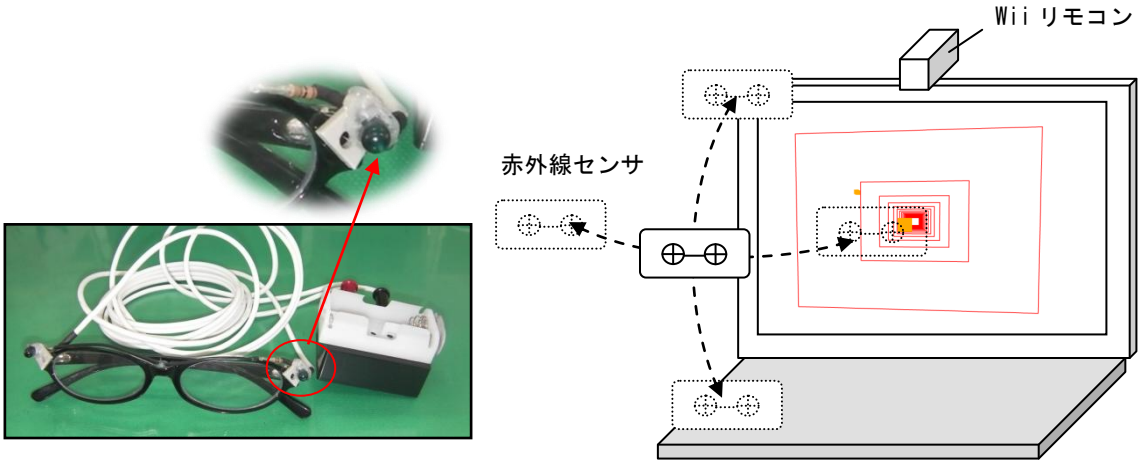
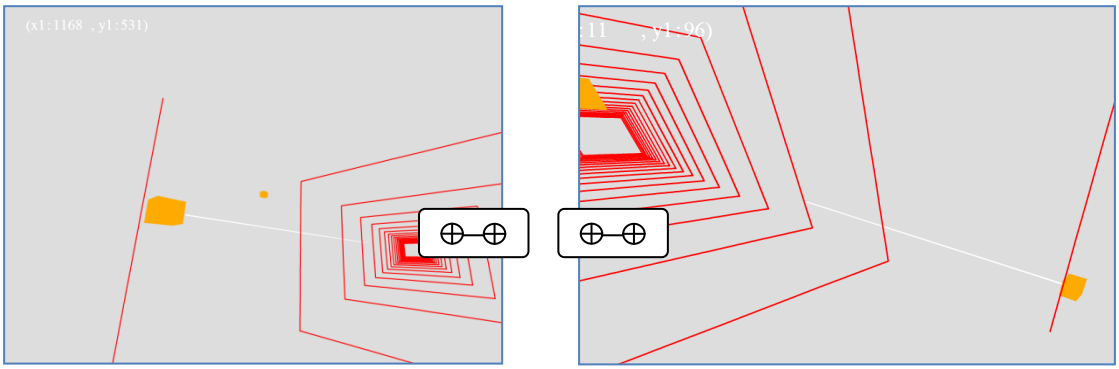


図 3.4.9　Wii リモコン設置

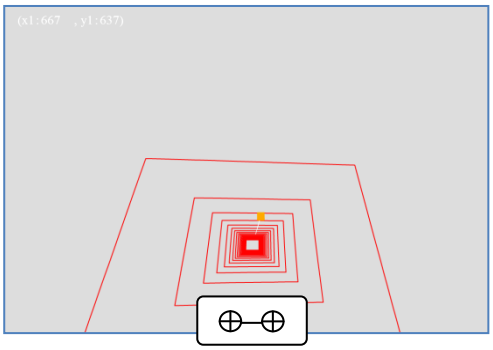


(a) 赤外線センサ　　(b) 赤外線センサの動き

図 3.4.10　赤外線センサと画面の関係



(a) 右から見る　　(b) 左から見る



(c) 下から見る

図 3.4.11　赤外線センサの位置による見え方

4　最後に

ActionScript3.0 によるプログラミングで作成できる Flash コンテンツは，アイディア次第でいろいろなことができることが分かる。ライブラリを活用したプログラムを試せるだけ試してみたが，他にも多くのライブラリが存在している。こうしたプログラムの大半は書籍やインターネットで容易に情報収集できる。

今回示したプログラムをするにあたって，ライブラリを利用することのメリットは，プログラムの入力が非常に少なくて済むということである。事前にサンプルプログラムを作り生徒に示すことで興味を持って取り組める。今後はさらにオブジェクト指向についても理解を深め，生徒へ還元していきたい。

iOS や Andoroid などのモバイル端末では Flash をサポートしないものも増えてきたが，ActionScript3.0 による AdobiAIR 技術で同じようなことができる。日々の情報技術の進歩と学校現場の情報技術を学ぶ環境のずれをできるだけなくせるように,工夫していきたい。

参考文献等

・zero point nine.com:
http://www.zeropointnine.com/blog/3d-page-curl-effect-using-papervision3d/
http://www.zeropointnine.com/blog/3d-page-curl-effect-updated/
http://blog.r3c7.net/?p=105

・gihyo.jp（技術評論社）:
http://gihyo.jp/design/feature/01/flartoolkit/0004

・Flash3D コンテンツ制作のための Papervision3D 入門（エクスナレッツ：池田泰延 著）

・Papervision3D ではじめる Flash3D アニメーション（技術評論社：ハヤシ カオル 著）

・ActionScript3.0 ライブラリ入門
（翔泳社：新藤愛大，池田泰延，浦野大輔，加茂雄亮
河村晃匡，小林陽介，高輪知明，タロタローグ，召田敬，森山篤 著）

・WiiRemote プログラミング
（オーム社：白井暁彦，小坂崇之，くるくる研究室，木村秀敬 共著）