Flashup#5

「簡単なゲームを作ってみよう。if文, for文, 配列をマスターしよう」

2010.10.23(土)
特別協賛:株式会社かっぺ様

orange-suzuki.com 鈴木克史

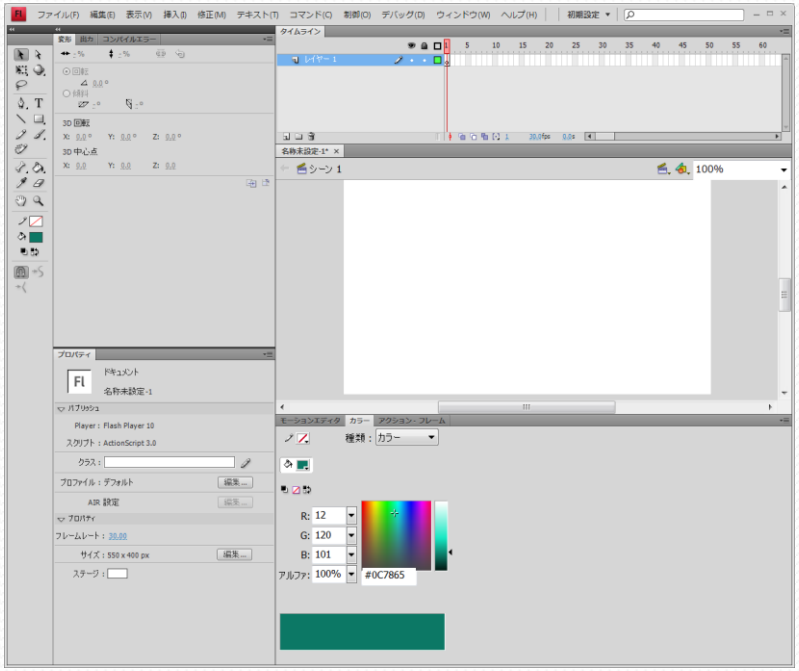
本日のメニュー

1. Flash コンテンツの作り方
2. ActionScript とは？
3. 簡単なゲームを作ってみよう

Flash コンテンツの作り方

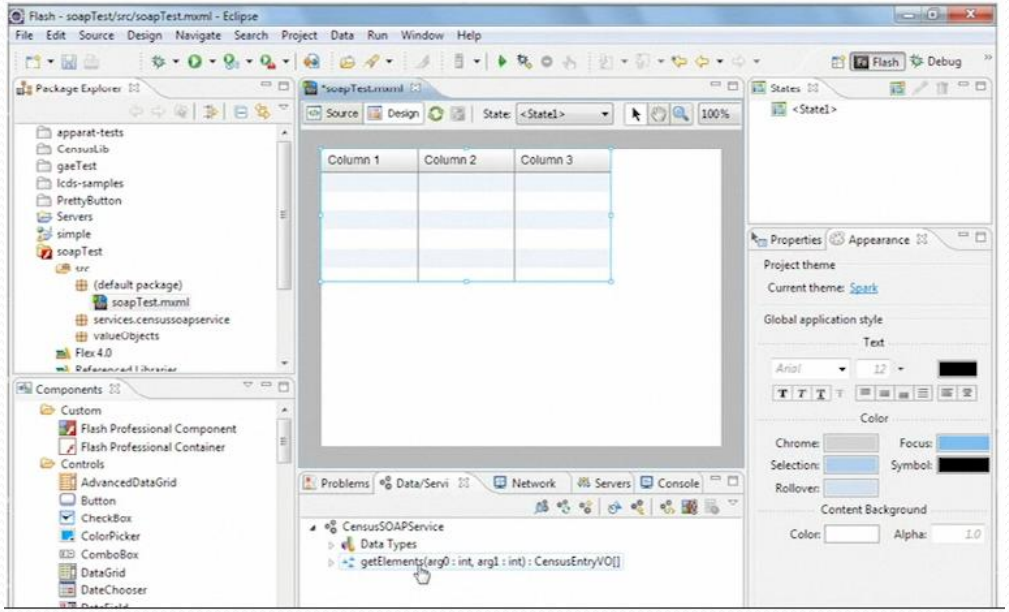
その1: Flash CSから作る

AdobeのFlash CS製品を用いて作成。
タイムラインを使ってのアニメーション作成、
見た目を確認しながら製作できる。スクリプトでの作成も可能。



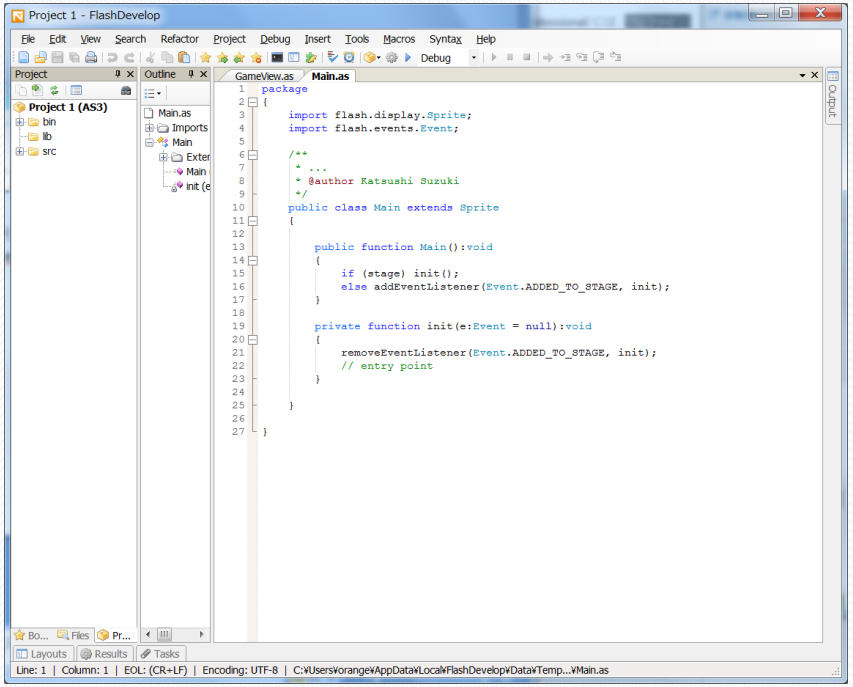
その2: Flash Builderから作る

AdobeのFlash Builder製品を用いて作成。
データ中心型の開発、チームでの大規模開発に適したツール。
デバッグ、パフォーマンスチューニングに優れている。



その3: 無料のコンパイラで作る

Adobeから無料で提供されているFlexSDKを用いて作成。
FlashDevelopなどのフリーソフトと併せて使うと便利。



ActionScript とは？

Flashのコンテンツの中で使われるスクリプト言語のこと。
ActionScriptを使うことで、動画や音楽を再生させたり、
グラフィックを制御したり、サーバーと通信させたりと、
様々なことを実現できます。

※参考資料
ActionScript 3.0の概要 | デベロッパーセンター
http://www.adobe.com/jp/devnet/actionscript/articles/actionscript3_overview.html

ActionScript 3.0書き方教室 | デベロッパーセンター
http://www.adobe.com/jp/devnet/flash/articles/writing_actionscript.html

ActionScript の文法

変数の宣言

var 変数名:変数の型;

数値や、文字列、配列、オブジェクトなどの値を保持する入れ物。入れたり、出したり、くっつけたり、消したり、様々なことに使う。使うには一度 var を使って宣言する必要がある。

var book:String;
(bookという名前のString型の変数を宣言)

var num:Number;
(numという名前のNumber型の変数を宣言)

var book:String = “教科書”;
(bookという名前のString型の変数を宣言し、”教科書”という文字列を代入)

var num:Number = 100;
(bookという名前のString型の変数を宣言し、100を代入)

var _deck:Array = [];
(_deckという名前のArray型の変数を宣言し、空の配列を作成)

例題：

```
var familyName:String = “Tanaka”;
trace(familyName);

var firstName:String = “Makiko”;
trace(firstName + “ “ + familyName);

var money1:Number = 100;
trace(money1);

var money2:Number = money1 + 200;
trace( money2 );
```

関数の宣言

ある機能をコードにしたもの。コード内では、例えば変数同士を足したり、
グラフィックを移動させたり、if文を使って分岐させたりすることができる。

```
function 関数名(引数:引数の型):戻り値の型
{
        実行するコード
}
```

例題：

```
function sum( a:Number , b:Number ):Number
{
    return a + b;
}
var result:Number = sum( 10, 20 );
trace( result );

var result2:Number = sum( 10, -20 );
trace( result2 );
```

クラスの宣言

クラスとは、ある機能や性質を持った集まり。
例えば、Cardクラスでは、カードの機能と性質を持つ。など。

```
package パッケージ名
{
        import 継承したクラス;
        import コードで使用するクラス;

        public class クラス名 extends 継承クラス
        {
                //変数
                public var 変数名:型;

                //コンストラクタ
                public function クラス名()
                {
                }

                public function 関数名():戻り値
                {
                }
        }
}
```

```
package
{
    import flash.display.MovieClip;

    public class Card extends MovieClip
    {
        public var id:uint;

        public function Card()
        {
        }

        public function flip():void
        {
        }
    }
}
```

(MovieClipクラスを継承した、Cardという名前のクラス。変数 id を持ち、関数 flip を持つ。)

・コンストラクタとは、クラスを生成したときに、実行される関数のこと。
・継承とは、そのクラスを機能を持ち、さらに機能を拡張したい時に使う。

クラスの使い方

var 変数名:クラスの型 = new クラス名();
(インスタンス化)

インスタンス.関数();
(インスタンスの利用)

クラス名.関数();
静的関数(staticな関数の利用)

例題:

var mySprite:Sprite = new Sprite();
(Spriteクラスのインスタンスが作成され、mySpriteという変数に格納する。)

mySprite.x = 100;
(mySpriteのx座標を100とする)

mySprite.addChild(title);
(mySpriteにtitleという表示オブジェクトを追加する)

trace(Math.random());
(Mathクラスのスタティックな関数、randomを実行する)

画像の生成について

Flash CSを使う場合

```
package
{
    import flash.display.Bitmap;
    import flash.display.MovieClip;

    public class Test extends MovieClip
    {

            public function Test():void
            {
                    //画像の生成
                    var title:Bitmap = new title1_png();
                    title.x = 258;
                    title.y = 88;
                    addChild(title);
            }

    }

}
```

タイムラインを使う場合

```
//画像の生成
var title:Bitmap = new title1_png();
title.x = 258;
title.y = 88;
addChild(title);
```

画像の生成について

FlashDevelopを使う場合

```
package
{
    import flash.display.Bitmap;
    import flash.display.MovieClip;

    public class Test extends MovieClip
    {
            [Embed(source = "asset/title.png")]
            private static var title1_png:Class;

            public function Test():void
            {
                    //画像の生成
                    var title:Bitmap = new title1_png();
                    title.x = 258;
                    title.y = 88;
                    addChild(title);
            }
    }
}
```

配列について

今回使用する配列の機能

```
var items:Array = [];
```
(配列の初期化)

```
items.push( item1 );
```
(item1を格納する)

```
var items:Array = [ item1, item2, item3 ];
```
(配列にitem1, item2, item3 を順番に格納する)

```
var item:Bitmap = items[ 0 ];
```
(配列の0番目の要素を参照する)

```
var len:int = items.length;
```
(配列の長さを参照する)

例題：

```
var numSet:Array = [];
numSet.push(100);
numSet.push(200);
numSet.push(300);

trace( numSet[0] );
trace( numSet[1] );
trace( numSet[2] );
trace( numSet.length );

var numSet2:Array = [1,2,3,4,5];
trace( numSet2 );
```

if文について

今回使用するif文の機能

```
if ( 条件式 ){
    //実行する内容
}
```

例：

```
if ( state == "END"){
    viewStart();
}
```

(state変数が、”END”という値の時、関数viewStartを実行する)

```
if ( yen < 10000 ){
    goHome();
}
```

(yenという変数が、10000未満の時、関数goHomeを実行する)

例題：

```
var point:Number = 80;

var exam:String;
if( point > 80 ){
    exam = “pass”;
}
else {
    exam = “failure”;
}

trace( exam );
```

for文について

今回使用するfor文の機能

```
for( 初期化式; 条件式; 再初期化式)
{
    //実行する内容
}
```

例題：

```
var sum:uint = 0;
for (var i:int = 0; i < 10; i++)
{
    sum += i;
    trace( sum );
}
trace( sum );
```

ランダムな値の作り方

今回使用するMathクラスの機能

Math.random();
(0～1未満の中からランダムな値を返す)

Math.floor(value);
(パラメータ val で指定された値を切り捨てた値を返します。)

例：
var r:Number = Math.random();
(0～1の間でランダムな値を変数rに代入)

var r:uint = Math.floor(Math.random() * 5);
(0～4の整数の中からランダムな値を取得し変数rに代入)

例題：

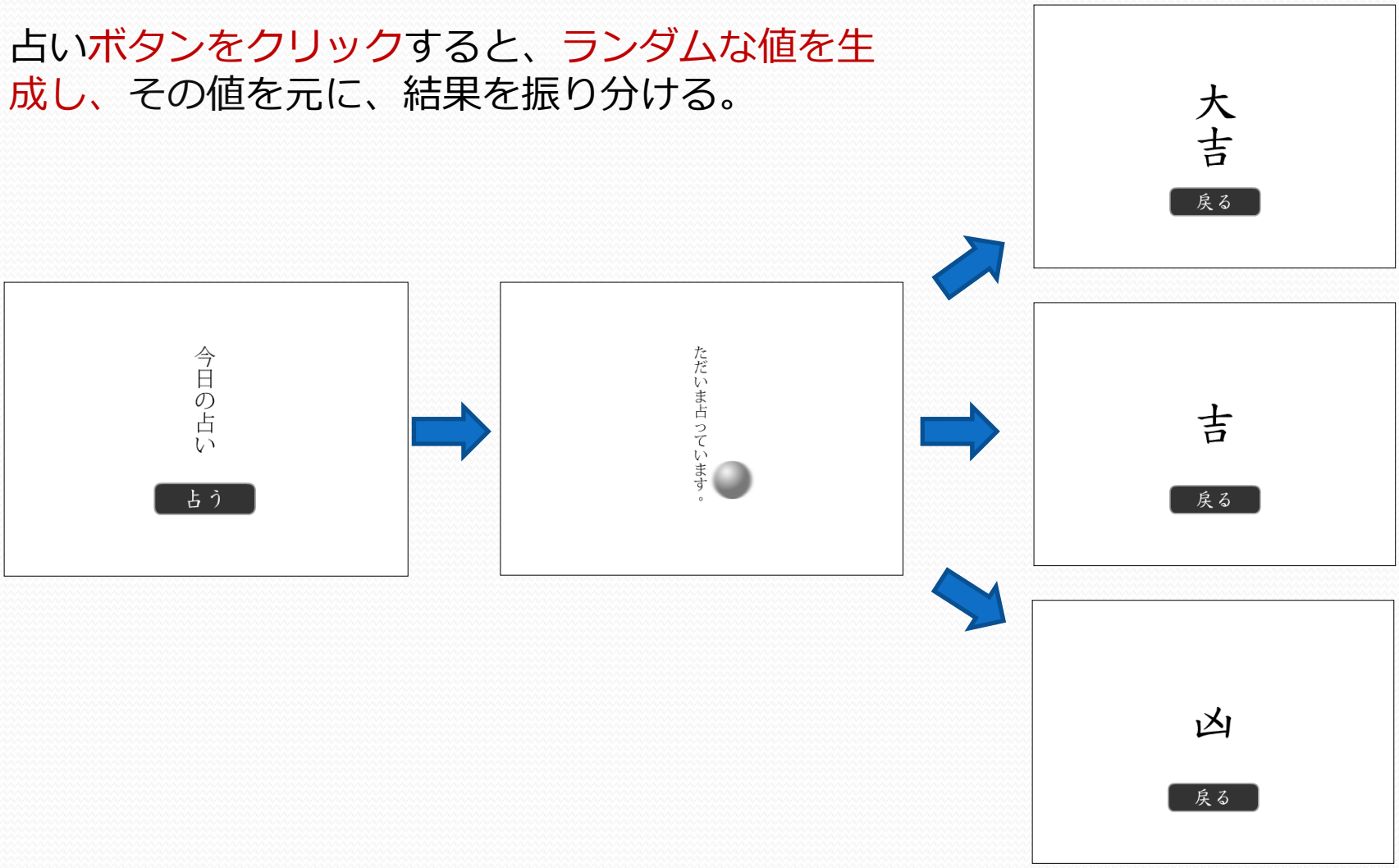
```
var r:Number = Math.random();
trace( r );
trace("------");

for (var i:int = 0; i < 10; i++)
 {
     var r2:uint =
     Math.floor(Math.random() * 5);
     trace( i+"回目: "+r2 );
}
```

簡単なゲームを作ってみよう

おみくじゲームの作成

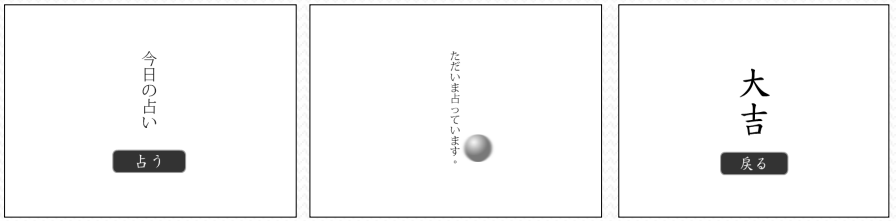
ゲームの内容

占いボタンをクリックすると、ランダムな値を生成し、その値を元に、結果を振り分ける。



画面の作成

元となる画面を作成し、その画面にタイトルの画像を追加していく。




```
//スタート画面の作成
var _startBase:Sprite = new Sprite();

//タイトル画像の作成
var title:Bitmap = new title1_png();

//タイトル画像をスタート画面に追加
_startBase.addChild(title);

//表示リストに追加
addChild(_startBase);
```

ボタンの作成

ボタン用のクラス画面を作成し、クリックイベントを定義する。
さらにクリック時のイベントハンドラを定義する。


```
//ボタン用のクラスを作成
var start_btn:StartBtn = new StartBtn();

//ハンドカーソルの表示を有効にする
start_btn.buttonMode = true;

//クリックイベントを受け取る
start_btn.addEventListener(MouseEvent.CLICK, clickHandler);

//クリックしたときのイベントハンドラ(クリック時に実行される関数)
function clickHandler(e:MouseEvent):void{
    //クリックした時のコード
}
```

結果を配列に格納

元となる画面を作成し、その画面に結果の画像を追加していく。
さらに配列に画像を配列に格納する。


```
//結果画面の作成
var _resultBase:Sprite = new Sprite();

//結果用の画像の作成
var result0:Bitmap = new result0_png();
var result1:Bitmap = new result1_png();
var result2:Bitmap = new result2_png();

//結果を配列に格納する
_resultItems = [result0, result1, result2];

//表示リストに追加
_resultBase.addChild(result0);
_resultBase.addChild(result1);
_resultBase.addChild(result2);
addChild(_resultBase);
```

ランダムな値の生成と結果の分岐

結果が格納された配列から、1つの結果をランダムで選び出し、その結果を表示させる。
(その他は非表示とする。)


```
//全結果を非表示にする
for (var i:int = 0; i < _resultItems.length; i++) {
    _resultItems[i].visible = false;
}

//ランダムな値を生成
var r:uint = Math.floor(Math.random() * 5);

//目的の結果のみ表示させる
_resultItems[r].visible = true;
```

画面の切り替え

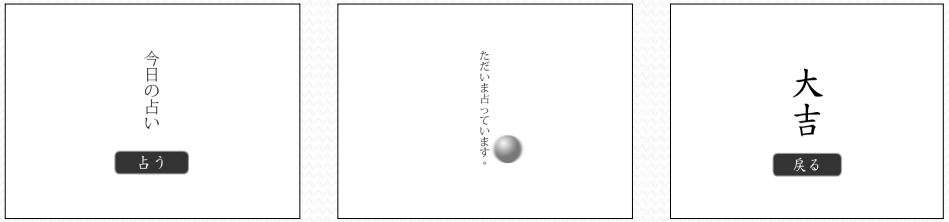
現在の状態を元に画面を切り替える。
状態により表示を変える関数を作成する。

```
/**
 * 状態により表示を変える関数
 */
function changeView():void
{
	//一旦非表示にする
	_startBase.visible = false;
	_processBase.visible = false;
	_resultBase.visible = false;

	//目的の画面を表示する
	if (_state == "START")
	{
			_startBase.visible = true;
	}
	else if (_state == "PROCESS")
	{
			_processBase.visible = true;
	}
	else if (_state == "END")
	{
			_ resultBase.visible = true;
	}
}
```

まとめ

必要な画面：

・スタート画面
・占い中画面
・結果画面



準備すること：

・画面の作成
・ボタンの作成
・結果を配列に格納

必要な機能：

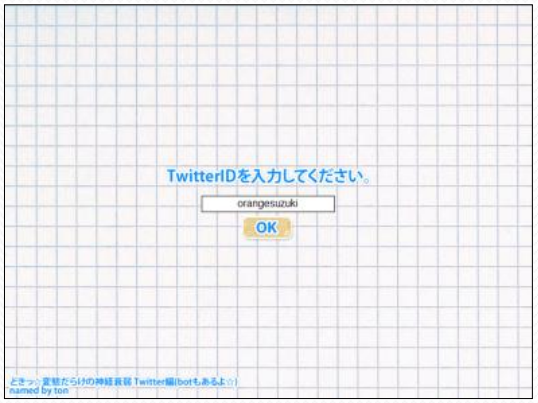
・画面の切り替え
・ランダムな値の生成と結果の分岐

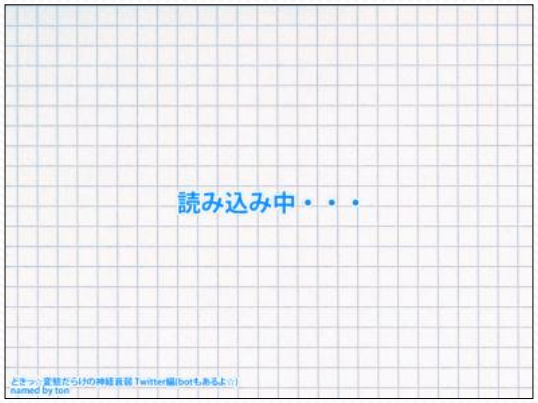
神経衰弱ゲームの作成

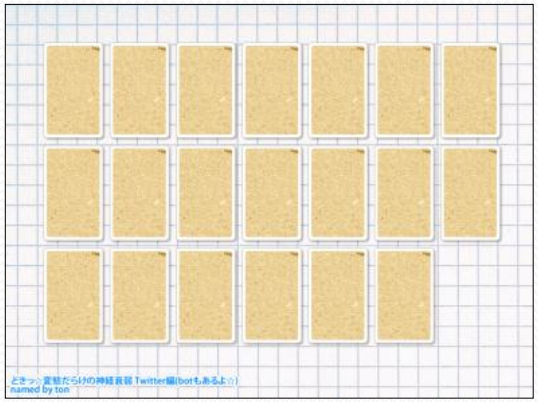
ゲームの概要 1

必要な画面：

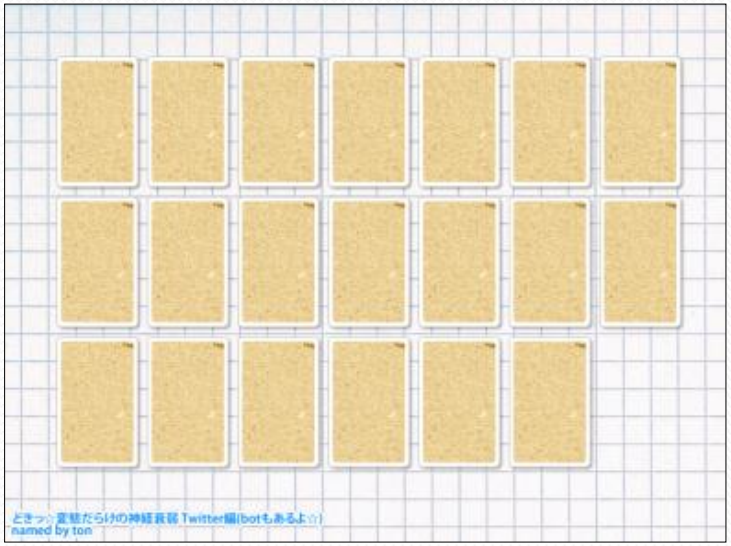
- スタート画面
- 読み込み中画面
- カード選択画面
- 演出画面×2







ゲームの概要 2



準備すること：

・画面の作成
・演出の作成
・Twitter画像の読み込み(API利用)
・ペアのカードを作成する

全体の機能：

・画面の切り替え
・正解を判別する関数(カードごとにIDをもたせ判別)
・全問正解かどうかを判別する関数

カードの機能(Cardクラスを作成)：

・カードをひっくり返す
・カードをクリックできるようにする
・カードをクリック不可とする
・アイコン画像を保持する
・名前を保持する
・正解に判別するIDを保持する