

processingで遊ぼう ;)

[Programs](#)

概要

Processingを使ったサンプルゲームプログラムや、その開発雑記です。

目次

- [概要](#)
- [目次](#)
- [サンプル](#)

- [Processingで作る、100行プログラムのアドベンチャーゲーム](#)
- [Processingで作る、300行の3Dスペースシューティング](#)
- [続報](#)
- [OpenProcessing](#)
- [ProcessingでMovie作成](#)
- [Processingで3D描画を絵画調に](#)
- [ProcessingでTweetカードゲーム](#)
- [窓シミュレーター- パソコンのモニターをバーチャルな窓に](#)
- [3/29 追記](#)
- [3/30 追記](#)
- [窓シミュレーターの認識を向上&高速化](#)
- [Processingでドリブルアクションゲーム](#)
- [Processingで貼り絵シェーダー](#)
- [Processingで簡単Androidプログラミング](#)
- [Kinectで魔法の剣や弓を手にしたら？ - Kinect Magic Knight](#)
- [9/5 追記](#)
- [KinectMagicKnight - 魔法の剣と弓矢で遊ぶ300行プログラム](#)
- [KinectとPCの接続方法について](#)
- [Processingで3Dモデル表示&環境マッピング](#)
- [ProcessingのMQOLoaderをライブラリ化](#)
- [Processingでスクリーン座標を3D座標に変換](#)
- [Processingでレイアウトエディタを作ろう](#)
- [Processingで3D物理シミュレーション](#)
- [Processingでプルプル立体GIFアニメ](#)
- [ゲームの中に入って遊ぼう - Kinect Sky Labyrinth](#)
- [アイデアがどんどん湧いてくる!? Processingで作るスマートフォン向けWebアプリ](#)
- [いっきに作ってしまおう。ProcessingでAndroidアプリを作成&公開する方法](#)
- [Processing.jsの画像の透過処理を高速化](#)
- [ゲーム作りで冒険！「遊んで作るスマホゲームプログラミング」](#)
- [イチから簡単なアクションゲームを作る15分ぐらいの動画](#)
- [お風呂でスクリーンに浸かる！ フロジェクションマッピングに挑戦](#)
- [続報](#)
- [テルマエと風呂ジェクションマッピング](#)
- [続報 2014-7-28](#)
- [Unityで簡単プログラミング、Processing気分を味わう](#)
- [風呂の水面+浴槽にも投影する、風呂ジェクションマッピング2](#)
- [BATHROOM SYMPHONY](#)

[はじめての方へ](#)

[Top](#)

[Game / Programs](#)

[Stories](#)

[Plays](#)

[Links](#)

[Store](#)

[BBS](#)

[About Me](#)

[Diary Logs](#)

[Tweet Logs](#)

[\(English\)](#)



Photo from [WritingTumblr](#)

search

407586



[Android](#) [C++](#) [Flash](#) [HTML5](#) [Kinect](#)

[LifeHack](#) [OpenFrameworks](#)

[Processing](#) [UI](#) [Unity](#) [WritingCafe](#)

[iPhone](#) [processing](#) [おいしい](#) [アイ](#)

[デア](#) [アプリ](#) [ガジェット](#) [ゲーム](#) [ツ](#)

[ール](#) [引越し](#) [映画](#) [家庭](#) [家庭菜園](#) [花](#)

[火](#) [画像処理](#) [開発](#) [嬉しい](#) [興味深い](#)

[講義](#) [講座](#) [手帳](#) [素材](#) [東京](#) [本](#) [名言](#)

[面白い](#) [遊び](#) [旅行](#)

[Twilog](#)



Syndicate this site

[RSS](#) [1.0](#)

- [DARTROOMSTIMULATION](#)
- [360行でイチから作るRPG - Processingで簡単RPG開発](#)
- [370行で作るRPG - 画像対応版](#)



We're Sorry - Service has been deprecated

Amazon Ecommerce Web Service 3.0 has been deprecated after many years of useful service on March 31st 2008. Please upgrade to the Amazon Associates Web Service 4.0 as detailed in the [migration guide](#) . Please visit [Amazon Associates Web Service Developer Forum](#) for more information. If you came to this page from an RSS feed, visit [Amazon's Product RSS Feeds](#) page for an upgrade.

サンプル

Processingで作る、100行プログラムのアドベンチャーゲーム

ひさびさの更新でWikiの文法を忘れてる今日この頃ですが、アート系のプログラミングで使われることが多い[Processing](#)で、アドベンチャーゲーム風のプログラムを書いてみました。



[AdventureGameサンプル](#)

ソースコードはコメント込みで約100行です。シナリオのテキストを読み込んで解釈して実行しています。フラグや条件分岐、サウンド再生などは省いているので、そういった機能を足してみるのもいいですね。

Processingは3Dでもカメラ入力でもサウンドでもお手軽な関数が揃っていて、シンプルなのがいいです。Javaですが、単純な命令で動かすので、N88-BASICを使ってた頃を思い出します。

気になる方は[公式サイト](#)からダウンロードしてみましょう。

今回はJavaアプレットとして書き出しましたが、実行ファイル（win、osx、linux）も作れます。移植もされているので、iPhoneやAndroid、Flashなどでも（移植されている部分については）同じコードを動かすことができます。Add-onを使えばOpenGLやBox2Dなども使えます。[Processing.js](#)を使えばHTML5も先取り（？）できます。

ゲームのプロトをだーっと作って試すのにも悪くないんじゃないかなあ。

ちょっと残念なのはProcessingのエディタにコード補完やデバッグ機能がないので、そういうのがほしければEclipseでJavaっぽい書き方も若干必要になることです。

速度が必要な場合は似た思想を持つC++ライブラリの[OpenFrameworks](#)もいいですね。

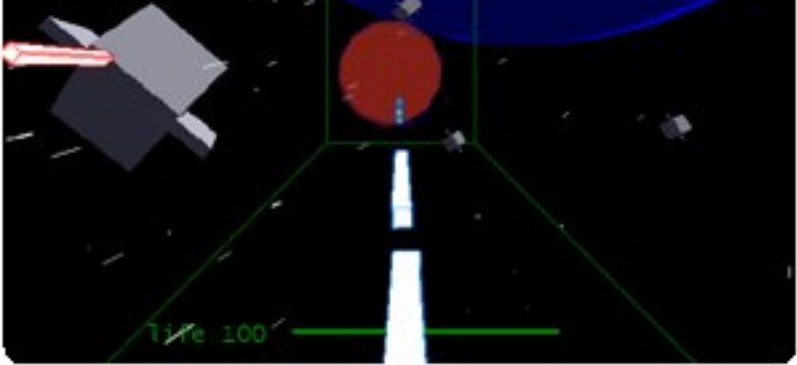
Tag : [Processing](#), [ゲーム](#), [開発](#)

[Trackback\(0\)](#)

Processingで作る、300行の3Dスペースシューティング

[前回](#)が2Dだったので、今回は3Dのゲームを作ってみました。





SpaceShootingサンプル

マウスカーソルを動かして向きを変えて、左クリックで弾を発射です。右クリックかスペースキーで加速、離すと減速です。10体の敵をすべて倒せばクリアです。

何秒で敵を殲滅できるでしょうか。

プログラムするときに困ったのは、pushMatrixやpopMatrixはあるのに、Matrixを取ってくる関数やMatrix型がリファレンスに無かったこと。Matrixなしで3Dゲームは辛い……。いや、普通あるだろうーとWebでいろいろ検索してみたら[getMatrix](#)とか、[PMatrix](#)とかありまして、おかげで無事プログラムできました。

今回はClassも使ってしまいました。ProcessingのIDE（=PDE）は、クラスを別ファイル（別タブ）にして扱ったりもできるのですが、あえて1ファイルにしてあります。

宇宙モノはモーションいらないし、背景の星も点や球でよいので楽ですね。

■ 続報

3/5にこのゲームの画面をMovieで出力する解説を追加しました。

Tag : [Processing](#), [ゲーム](#), [開発](#)

Trackback(0)

■ OpenProcessing

[OpenProcessing](#)というサイトでProcessingのコードをシェアできるようなので、[登録してみました](#)。いろいろ眺めるのも楽しいです。

登録のやり方は[OpenProcessingでスケッチを共有する](#)が分かりやすかったです。

Tag : [Processing](#), [開発](#)

Trackback(0)

■ ProcessingでMovie作成

Processingで処理した画像をムービーとして出力するテスト。



 Ads by Google

[▶ Processing pdf](#)

[▶ Web processing](#)

[▶ Game processing](#)



少し前に作った3D Shootingを使ってやってみました。

Processingには便利なライブラリがあるので、こういったことも簡単にできます。
付け加えるコードは7行程度です。

ファイルの冒頭にまず、これ。

```
import processing.video.*;
MovieMaker mm;
```

次にsetup関数内のsizeを指定した後に、

```
mm = new MovieMaker(this, width, height, "sample.mov", 60, MovieMaker.MOTION_JPEG_A, MovieMaker.LOSSLESS);
```

あとはdraw関数の最後に

```
mm.addFrame();
```

で、キー（今回はMovieの頭をとって、Mキーにしました）を押したら出力を終了するようにしたら完成です。

```
void keyPressed() {
  if (key == 'm' || key == 'M') mm.finish();
}
```

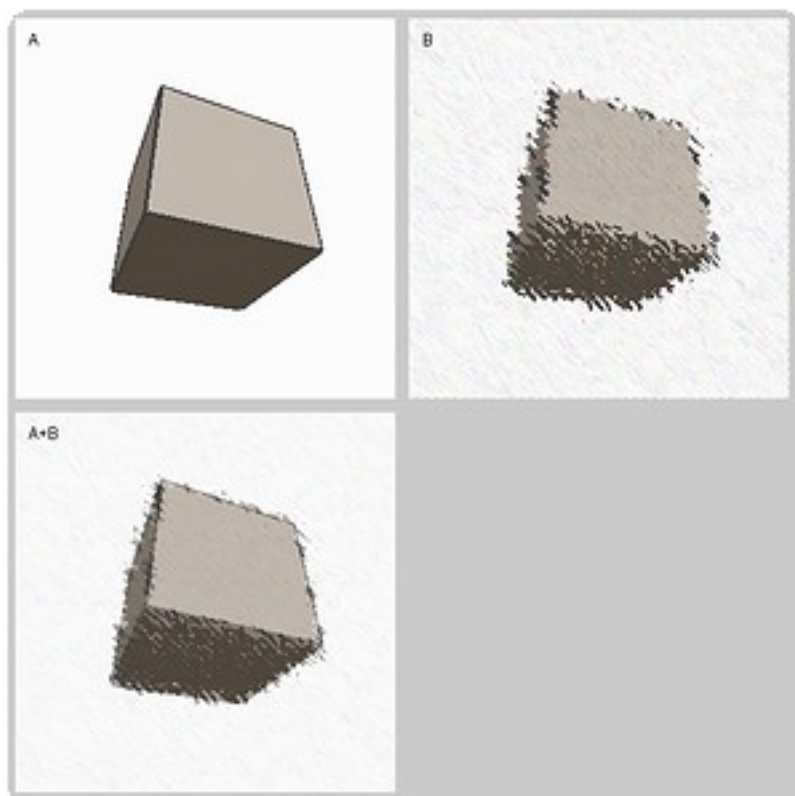
出力したファイルをYouTubeに投稿したら冒頭のものできました。お手軽ですね。

Tag : [Processing](#), [開発](#)

[Trackback\(0\)](#)

■ Processingで3D描画を絵画調に

今回は3Dの箱を絵画調（？）に描画してみました。



Pictorial3DViewサンプル

左上のAがオリジナルの3D画像、右上のBが絵画調に変換した画像、
左下がその2つを合成した画像です。

左上から右下へとスライドした画像です。

処理は簡単です。ピクセルを見てランダムに斜めの線を引いてるだけ。
そのわりにソースコードが冗長で、約60行ぐらいです。

でもこういうことも簡単にできるのがProcessingのいいところです。

Tag : [Processing](#), [画像処理](#), [開発](#)

[Trackback\(0\)](#)

■ ProcessingでTweetカードゲーム

今回はProcessingでTwitterのつぶやきをカードゲームのカードふうにしてPDF出力してみました。



TweetCardsサンプル（Web上で実行はしません）

Twitterの最新PublicTimelineからつぶやきをひっぱってきて、つぶやきごとにカードを作成しています。

できあがったPDFのサンプルは[こちら](#)。
両面印刷するか厚紙に貼り付けるかして、ハサミで切り取ればカードにできます。





ユーザーアイコンの左右にあるのは「フォローしている数」、「フォローされている数」です。これらを使ってゲームルールを作るのもよし、2部印刷して神経衰弱するのもよし、単純にポスターとして飾るのもよし。

ソースは約150行。Twitterへのアクセスには [tweet4j](#) というライブラリを使用しました。

日本語フォントのPDFへの出力がうまくいかなかったので、PGraphicsPDFとは別にPGraphicsを作成し、そちらにカードを1枚描画してからimage命令でPDFの方に描画する、といった小細工をしています。

あとカードの裏の模様は、画像を使えばいいものを、ムダにプログラムで図形描画して生成しています。

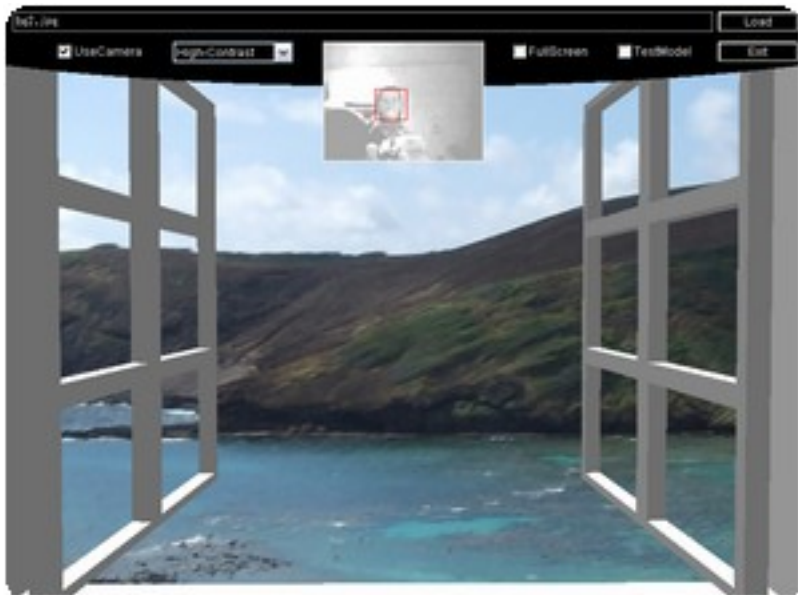
このソースをもとに、Friendsメインに改造するのもよいですね。

Tag : [Processing](#), [開発](#), [ゲーム](#)

[Trackback\(0\)](#)

■ 窓シミュレーター- パソコンのモニターをバーチャルな窓に

今回はProcessingでバーチャルな窓を作ってみました。
どこでもドアみたいな窓です。



[WindowSimulator](#) (Web上で実行はしません)

カメラで顔の位置を認識して、その位置から見えるであろう風景や窓をモニターに描画する、という仕組みで本物の窓っぽさを出しています（技術的な新しさはないです）。ただ、そんなに動作速度が速くないので、ゆっくり動くのがコツです。あとカメラの性能や室内の明るさや背景にある物なども影響します。

カメラがない場合はマウスカーソルの位置を顔の位置ということにして動作します。起動直後はカメラの認識がOFFになっているので、カメラを使う場合は、マウスカーソルを画面上部にもっていきと出てくるUseCameraチェックボックスをONにしましょう。

背景はローカルファイル、ネット上のURLを指定することで、自由に変えられます。

今はそれどころではないと思いますが、[今回の震災で変わる前の街の様子](#)をこの窓を通して見ることもできます。過去の大切な思い出にいつでもつながる窓として、ほんのひととき、なんらかの形でお役に立てたらいいなと思います。

震災とは別の話ですが、自分の場合は、上京している間に故郷の海岸が整備され、

小さい頃に遊んでいた景色が様変わりしてしまいました。そのことが寂しくて、

またそこにつながったらいいな、という気持ちもありました。
切なくもあります、それが確かにあったという事実は変わらないですから。

ソースは約330行で、カメラキャプチャに JMyron、顔位置の認識に OpenCV、ボタンなどのユーザーインターフェイスに SpringGUI を使用しました。

プログラムを作る上でのポイントは、窓枠と背景でカメラの視線のずれ具合を変えている点です。これをしないと窓枠が非現実的にモニターからはみ出したり、背景がほとんど動かなかったりするので。

しかし、プログラムは2日ぐらいで出来たのに、公開できるようにいろいろ整えるのに時間がかかるなあ。

3/29 追記

カメラを使おうとすると、OpenCVとJMyronのDLLが不足していて動かなかったのでDLLをZIPに含めるようにしました。動作確認はWindowsXPでのみ行っています。

3/30 追記

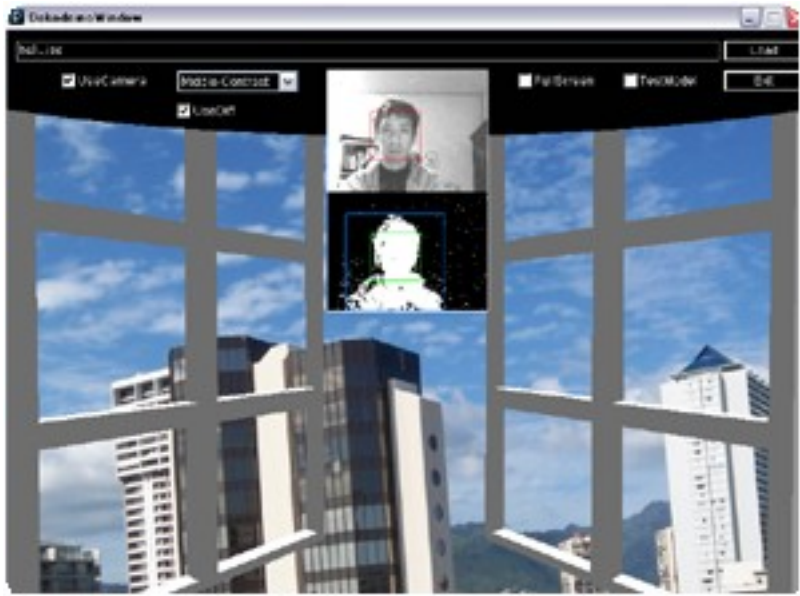
背景差分機能の追加と高速化を行いました。詳しくは こちら。

Tag : Processing, UI, 旅行

Trackback(0)

窓シミュレーターの認識を向上&高速化

前回作成したWindowSimulatorをバージョンアップしました。
やったのは背景差分機能の追加と高速化です。



WindowSimulator (Web上で実行はしません)

身体を素早く動かしているときなど、顔の位置を認識できなかったときに、背景と人物の差分から顔の位置を推測（頭は普通上の方にあるよね、と決めうち）するようにします。

UseDiffのチェックボックスをONにして3秒後にカメラから無人の背景を一時記憶するので、いったんカメラに映らない場所に身を隠してください。
（カメラに明るさを自動補正する機能がある場合、うまく背景差分がとれないことがあります）

上にあるグレーのカメラ画像で顔の前に出る赤枠が顔認識位置です。その下にある白黒カメラ画像の上に出る青枠が背景差分の塊認識で、緑枠がそれをもとにした最終的な顔位置です。

その他、以下のような各種高速化も行っています。

- 描画をP3DからOpenGLに変更
- Webカメラのキャプチャサイズを半分に
- Webカメラの更新間隔を8フレームに1回に

さらなる高速化などはOpenFrameworksとか使わないと難しいかな、という印象です。

ソースは400行ちょっとになっちゃいました。

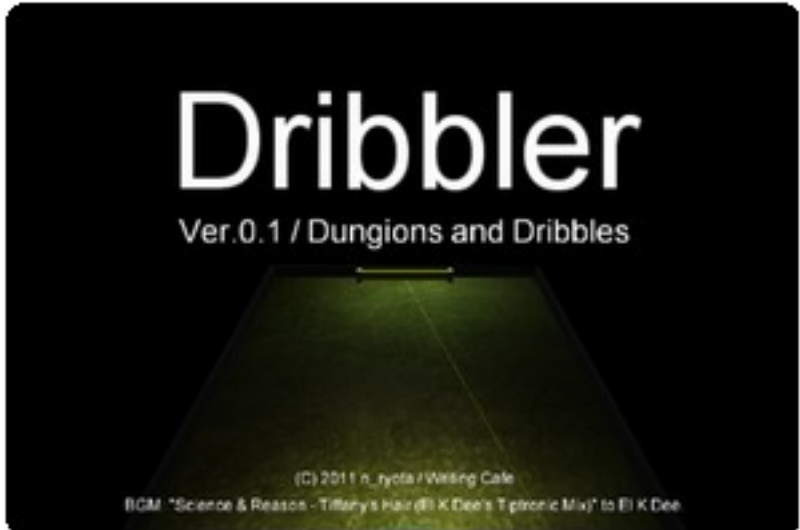
Tag : [Processing](#), [UI](#)

[Trackback\(0\)](#)

Processingでドリブルアクションゲーム

フットサルというミニサッカーのようなスポーツがあって、たまに友人たちと遊んだりします。普通はドリブルよりも、ボールをパスをうまくつなげる方が断然早いし、それはそれで楽しいのですが.....でもやっぱり、ボールをたくさん触りたいじゃないですか。

そこでコレ。「ひとりでドリブルしてシュートしたい」という欲求を受け止めてくれるゲームを作ってみました。



[Dribbler\(Web上で実行はしません\)](#)

ソースは、今回あまりにも雑なので載せてません。

全体的にいろいろ未完成のプロトタイプバージョンですが、enjoy it!

Tag : [Processing](#), [ゲーム](#)

[Trackback\(0\)](#)

Processingで貼り絵シェーダー

おむすびをもらったお礼.....ではないですが、
今回は画像を貼り絵（ちぎり絵）風にしてリアルタイム描画してみました。



HarieViewサンプル

上がオリジナルの画像、下が貼り絵風に変換した画像です。

最初は折り紙を破いた写真を撮影してそれを散りばめようと思ったのですが、面倒だったので、ひとまず矩形描画だけでやってみました。

Pictorial3DViewのときと同じく、処理は簡単です。ベース画像のピクセルを見て、新しい画像の位置にランダムに回転させた矩形を描いているだけ。ポイントは若干スキマが空くようにするのと、たまに白い縁取りをつけること。

ソースコードは、約60行ぐらいです。

ちなみに描画をカクカクさせているのはわざとです。あまりに高速に切り替わると貼り絵なのか判別できないので。

リンク先のアプレットをクリックすると元画像が切り替わります。

同じ理屈で応用もいろいろ。矩形や紙の代わりに石の写真を使えば石シェーダー、水滴を使えば水滴シェーダー、豆腐を使えば豆腐シェーダー、果物を使えば果物シェーダー、人を使えば人文字シェーダー。.....あとはご想像にお任せします。

Tag : Processing, 画像処理, 開発

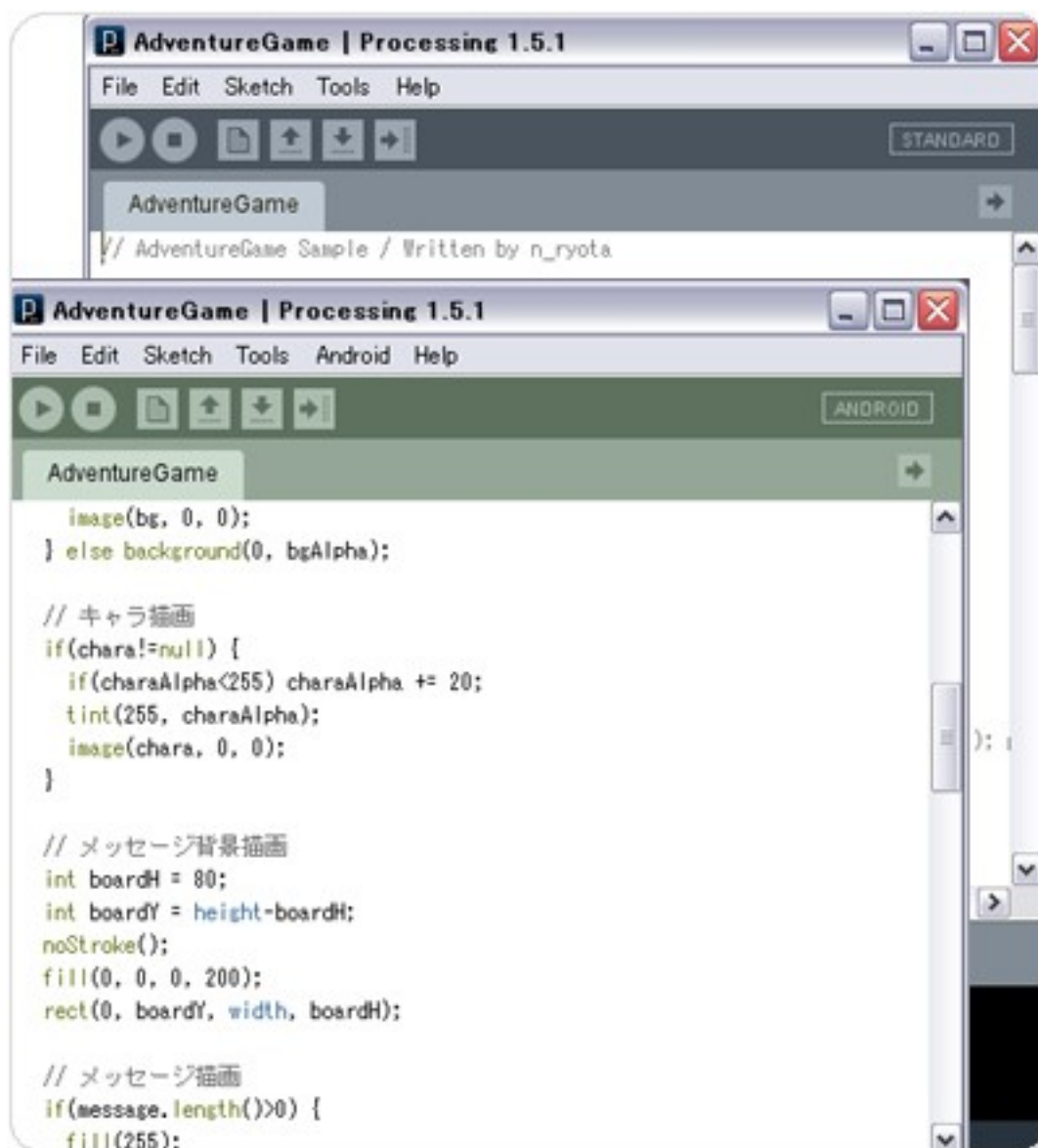
Trackback(0)

■ **Processingで簡単Androidプログラミング**

最近Androidの話題をよく見かけてまして、そういえばProcessing 1.5からAndroid書き出し機能が追加されてたなと思いーProcessingプログラムをAndroidで実行してみました。

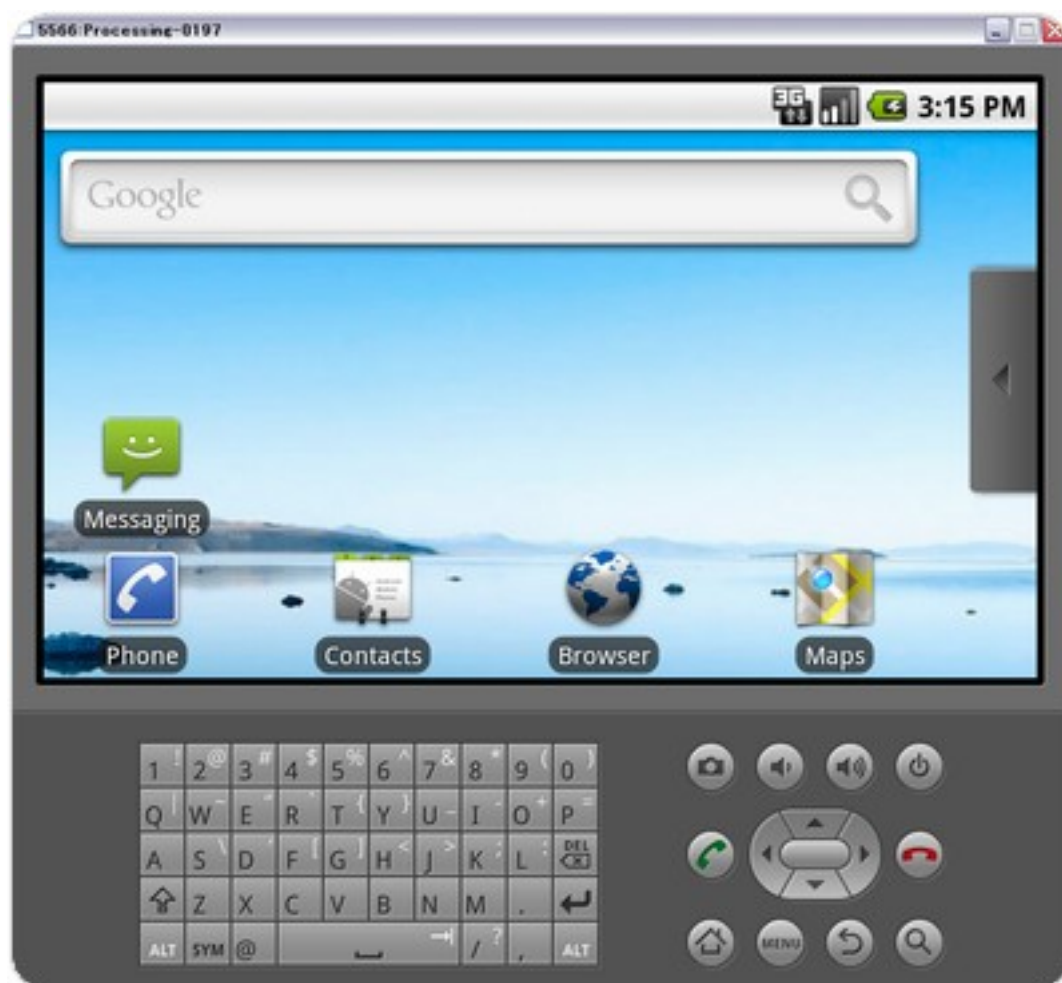
元となるプログラムはProcessingで作る、100行プログラムのアドベンチャーゲームのソースです。

まず、右上にある「STANDARD」をクリックして「ANDROID」に変更します。



と、その前にAndroidSDKを入れないとですね。 [Processing for Android](#)などの記事を参考にAndroid SDKをインストールして、エミュレーターを起動します。

Program Filesなどの下に入れるとパスのスペースのせいでヴァーチャルデバイスの起動に失敗したため、ドライブ直下にAndroidフォルダを作ると上手くいきました。



CTRL+F11で横向きにしてあります。

あとはいつもどおりProcessingの実行ボタン（またはCTRL+R）を押すだけ。





エミュレーターは遅いですが、今回は特にソースの変更もせずに、一応動きました。
全部上手くいくとは限りませんが、お手軽ですね。

Tag : [Processing](#), [ゲーム](#), [開発](#), [Android](#)

[Trackback\(0\)](#)

Kinectで魔法の剣や弓を手にしたら？ - Kinect Magic Knight

[Kinect](#) は、なんて楽しいおもちゃなんだろう。

というわけで、Kinectを使って魔法の剣や弓を召還.....もとい、合成表示するプログラムを作ってみました。

プレイ動画はこちら。

ところどころ、昔の白黒映画みたいな動きになってるのはご愛嬌。

この動画に撮る処理もプログラムで書いているため、フレームレートが実際の半分ぐらいになってます（実際はもうちょっと滑らかです）。

プログラムには、[Processing](#)と[SimpleOpenNI](#)を使用しています。
サンプルも充実しているので2日ぐらいで作れました。

と言っても、いろいろ適当すぎる内容だからというの也有ります。見ての通り、剣、盾、弓、矢はすべてBoxやSphereだし。でも、作っていて楽しかったです。

剣モードと弓モードの切り替えや、弓の操作が若干ジェスチャ入力っぽく思ふかもしれませんが、ここもシンプルに手首の現在位置を見るだけで処理しています。

あとはターゲットを作ったり、視点を変えたりすれば、このままゲームも作れそうですね。

そのうちアプリもアップする予定ですが、今日はこのへんで。
その際はKinect用ドライバのインストールから記事を書きます。

9/5 追記

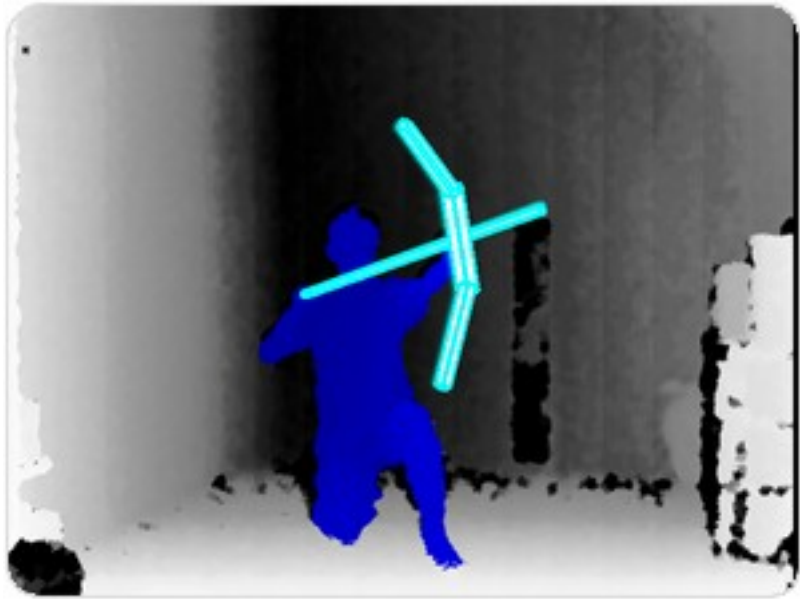
ドライバのインストール方法や、ソースコード、実行ファイルを用意しました。
[こちらから](#)どうぞ。

Tag : [Processing](#), [Kinect](#), [開発](#)

[Trackback\(0\)](#)

KinectMagicKnight - 魔法の剣と弓矢で遊ぶ300行プログラム

[前回](#)は動画だけでしたが、
今回はKinect Magic Knightのアプリとソースコードを公開します。



[KinectMagicKnightサンプル](#)

3種類の画面モード（通常カメラ、深度カメラ、3Dドットカメラ）があります。
このスクリーンショットは深度カメラのもの。

ソースコードも適当に書いてるときは600行ぐらいあったのですが、
300行程度にスリム化してみました。

プログラムはProcessingで書いてます。
剣と弓の切り替えの認識は単に手や頭の位置を見ているだけだったり、わりとシンプルな仕組みでこう
いったことができることが分かります。
（盾は球をscaleで変形させたり変なことしてますが）

このアプリで遊んだり、ソースを参考に色々KinectHackしてもらえると嬉しいです。

KinectとPCの接続方法について

KinectMagicKnightやOpenNIを利用したKinectHackを試すには、少し手間ですが、あらかじめ、OpenNI
関連のドライバをインストールして、
Xbox360の周辺機器 "Kinect" をPCにUSBケーブルなどで繋いでおく必要があります。



インストールは[OpenNI: WindowsでKinectを使う](#)のページが大変参考になりましたが、現在のバージョンでは不要な手順もあるので、以下に必要なものを書きます。

1. OpenNIをインストール

[OpenNIのページ](#)にある、

[OpenNI Unstable Build for Windows x86 \(32-bit\) v1.3.2.3 Development Edition](#)

などをダウンロードして、インストールしてください。

2. SensorKinectをインストール

[SensorKinectのページ](#)にある、[Downloads -> Download.zip](#)

などをダウンロードして、解凍してください。

KinectをUSBケーブルでWindows PCに接続すると、

ドライバのインストール画面がでるので、ドライバの検索場所に

avin2-SensorKinect-12ad25d/Platform/Win32/Driver

など、先ほど解凍したフォルダ内の Platform/Win32/Driver を指定してインストールを行います。

※Kinect単体で購入した場合は問題ありませんが、Xbox360本体のKinect同梱版には、PCにつなぐためのケーブル（電源/USBが二股になったもの）が付いていません。

これは、[Xbox カスタマーサポートに電話](#)して3,830円（送料込み）で注文できます。

その後、先ほど解凍したフォルダ内に

avin2-SensorKinect-12ad25d/Bin/SensorKinect-Win-OpenSource32-5.0.3.3.msi

といったインストーラーがあるので、こちらも忘れずにインストールしてください。

3.NITEのインストール

[OpenNIのNITEのページ](#)にある、

[PrimeSense NITE Unstable Build for Windows x86 \(32-bit\) v1.4.1.2 Development Edition](#)

などをダウンロードして、インストールしてください。

4.再起動

以上でOpenNI関連のインストールは完了です。

念のためPCを再起動しておきましょう。

Tag : [Processing](#), [Kinect](#), [開発](#)

[Trackback\(0\)](#)

■ Processingで3Dモデル表示&環境マッピング

Processingで3Dモデルを表示するライブラリは[いくつもあります](#)。

でも微妙に思い通りにいかなかったり、P3DではなくOpenGLレンダラー用だったりしたため、今回は[MQOモデル](#)を表示するクラスをベタに自作してみました。

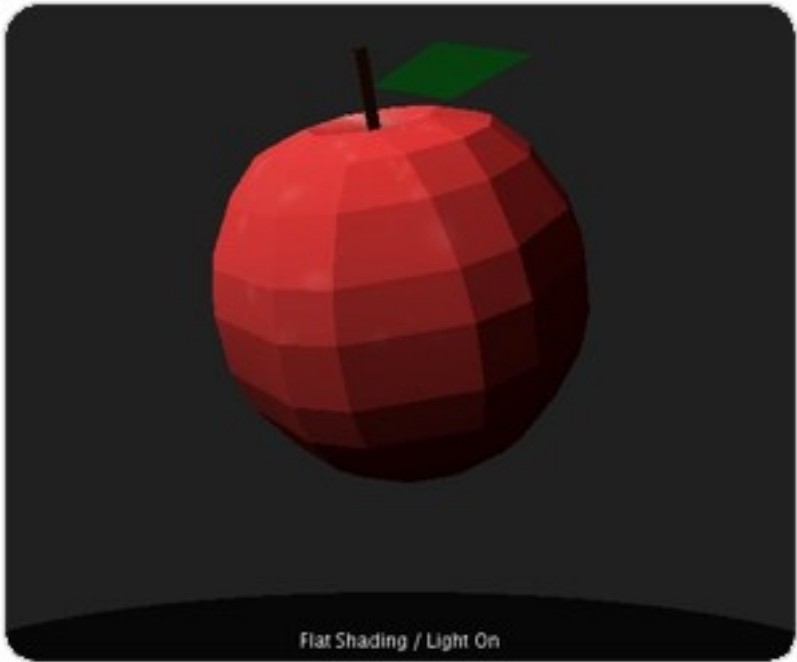




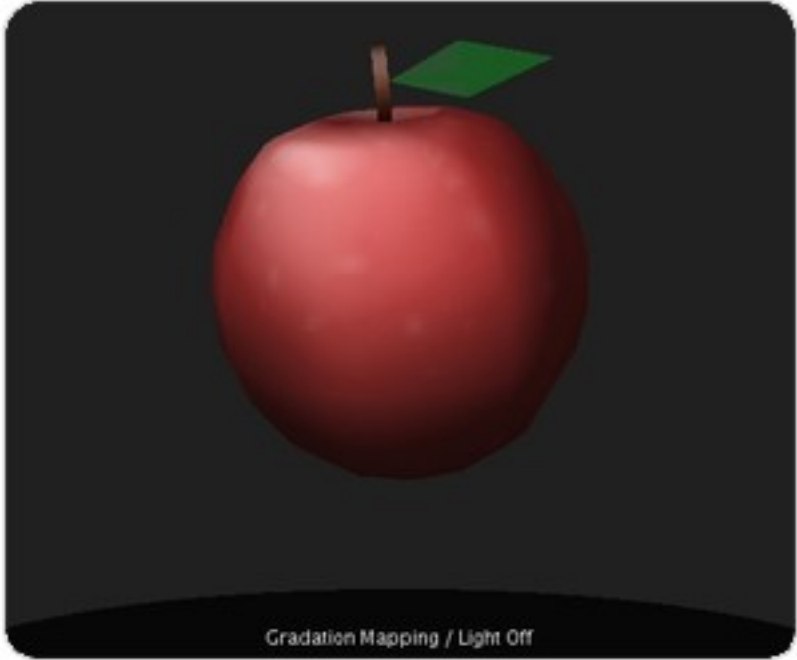
Model3DViewサンプルのページへ

モデルロードとワイヤーフレーム、フラットシェーディング、グローシェーディングなどだけだと地味だったので、ついでに環境マッピングに対応しています。

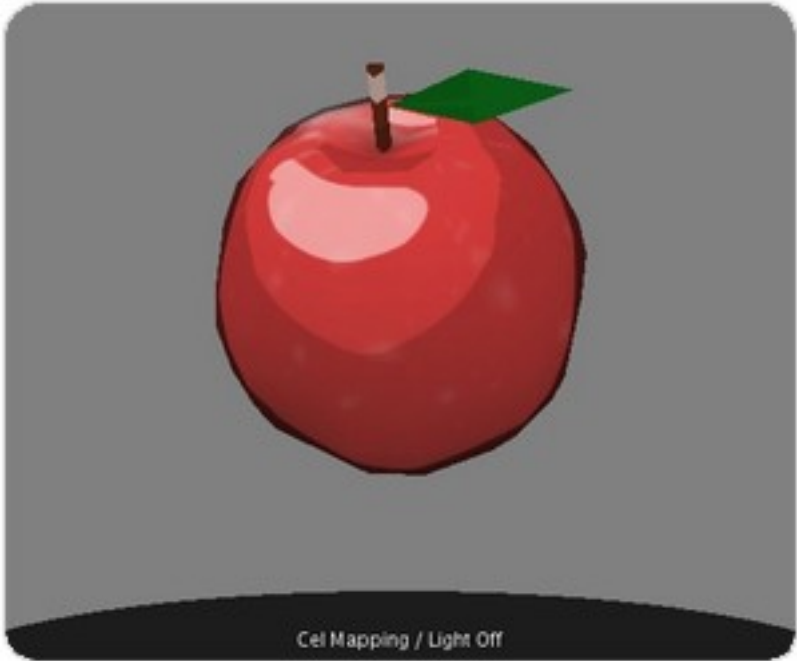
普通のフラットシェーディングだと以下のような感じ。



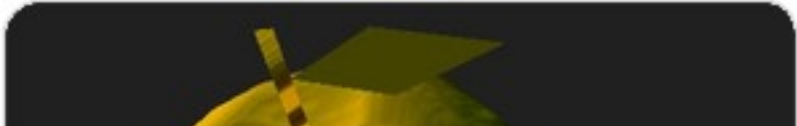
実は最初に出した以下の画像は、ライトオフで描画したモデルの上にαチャンネルありのスフィア環境マップ（反射マップ）として光と影を描き込んでいます。



その画像の階調を少なくすると……



こんなふうセルシェーディング（トゥーンシェーディング）風になります。





金色の環境マップを適用すれば金属風に（メッキっぽいですね）。
反射ではなく、屈折マッピングもできます。



これもフェイクですが、ガラスっぽくなりました。

なお、MQOモデル表示といえば[Processing](#)上でメタセコイア形式データを表示するライブラリもあるので、OpenGLレンダラーで高速に表示したい方は、そちらもオススメです。

Tag : [Processing](#), [ツール](#), [開発](#)

[Trackback\(0\)](#)

■ ProcessingのMQOLoaderをライブラリ化

[前回](#)作成したMQOモデルファイルの読み込みクラスを、ライブラリ化しました。

使い方は、プログラムの冒頭あたりで

```
import eylн.mqolаoder.*;
```

とライブラリを import した後、

```
MQOModel model = new MQOModel(this, "ファイル名");
```

のようにモデルをロードして^[1]

- ^[1]ファイル名なしのコンストラクタで初期化したあと model.load("ファイル名") でもOK

```
model.draw();
```

と描画するだけです。

使ってみたい方は、こちらの[MQOLoader.zip](#)をダウンロード後、Processingの作業フォルダ（Sketchbookフォルダ）直下のlibrariesフォルダに展開（解凍）してご使用ください。ライセンスはMITです。

はMIT Lisenceです。

しかし、Javaは不慣れ（というかProcessing以外で触れたことがほとんどない）でライブラリ化に結構手間取りました。公開されている [processing-library-template](#) やフォーラム [Problems using the library template](#) のおかげでなんとか。（でもライブラリテンプレート自体にバグがあったとは...）

改良Verなどできたら、ぜひご連絡を：)

Tag : [Processing](#), [ツール](#), [開発](#)

[Trackback\(0\)](#)

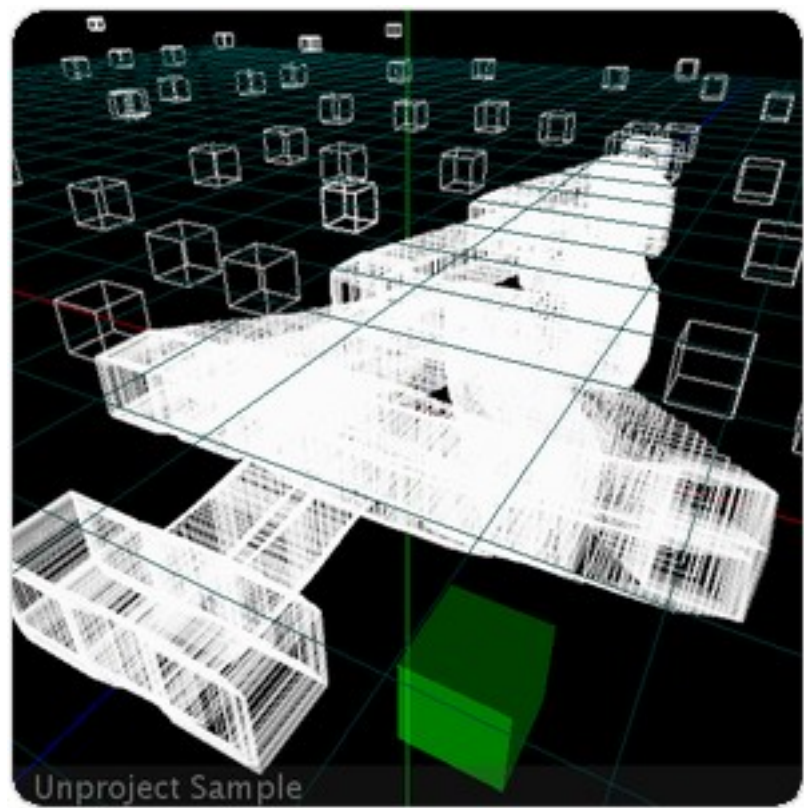
■ **Processingでスクリーン座標を3D座標に変換**

今、巷では[技術系Advent Calendar](#)というのが流行っているらしく、WritingCafeでもひとつ参加してみることにしました。

まあ、12月頭からクリスマスまでの間、各言語やテーマに沿った記事を1日1つ、いろんな人が書いていくお祭りみたいなもの(?)です。

そんなわけで[Processing Advent Calendar 2011](#)の12/2担当記事はこちら。

「**Processingでスクリーン座標を3D座標に変換する方法**」です。



[Unprojectサンプルのページへ](#)

Processingで3D空間の座標（モデル座標、オブジェクト座標）をスクリーン上の2D空間に変換するには、
screenX(x, y, z)、screenY(x, y, z)といった関数を使います。

では、スクリーン座標を3Dのモデル座標に変換するには、どうしたらいいでしょう。

そんなときにあるのがmodelX(x, y)、modelY(x, y)といった関数ーだといいのですが、そうではなく。modelX、modelYは単にローカルなモデル座標をワールド座標に変換してくれるだけです。なので、あいかわらず3D空間の値しか取れないのです。

OpenGLの関数、gluUnprojectを使えばスクリーンの2D空間の座標を3Dの座標に変換できますが、それだけのためにOpenGL関数やOpenGLレンダラーを使うのはどうも.....。

というわけで、今回はこれをProcessingの関数のみで計算してみます。

2D→3D変換の前に、まず3D→2D変換のための行列を作りましょう。

```
// モデル座標をスクリーン座標に変換する行列を取得
PMatrix3D getModelToScreenMatrix() {
```



```

PMatrix3D projection = new PMatrix3D();
matrixMode(PROJECTION); getMatrix(projection);

PMatrix3D modelview = new PMatrix3D();
matrixMode(MODELVIEW); getMatrix(modelview);

PMatrix3D viewport = new PMatrix3D();
viewport.m03 = viewport.m00 = width * 0.5f;
viewport.m13 = viewport.m11 = height * 0.5f;

PMatrix3D m = new PMatrix3D(modelview);
m.preApply(projection);
m.preApply(viewport);
return m;
}

```

モデルビュー行列やプロジェクション行列の取得方法は、[リファレンス](#)には載っていませんが、`matrixMode`と`getMatrix`を組み合わせると取得可能です。^[2]
ちゃんとviewport行列も作っているところがポイント。

- ^[2]PGraphics3Dのメンバ変数を `PGraphics3D p3d = (PGraphics3D)g`; 経由で、`p3d.modvw`とか、`p3d.projection`とか、`p3d.camera`といった行列を直接参照する方法もあります。

3D→2D変換の行列ができれば、目的の8割は達成したといえます。

```

// モデル座標をスクリーン座標に変換
PVector projectScreen(PVector v) {
    PMatrix3D m = getModelToScreenMatrix();
    return matrixCMult(m, v);
}

```

`projectScreen`関数内では、`getModelToScreenMatrix`で取得した3D→2D変換行列を
`matrixCMult`関数を使ってモデル座標のベクトルを掛けあわせた後、`w`で割っています。

```

// 行列で変換したベクトル(wで割ったもの)を返す
PVector matrixCMult(PMatrix3D m, PVector v) {
    PVector ov = new PVector();
    ov.x = m.m00*v.x + m.m01*v.y + m.m02*v.z + m.m03;
    ov.y = m.m10*v.x + m.m11*v.y + m.m12*v.z + m.m13;
    ov.z = m.m20*v.x + m.m21*v.y + m.m22*v.z + m.m23;
    float w = m.m30*v.x + m.m31*v.y + m.m32*v.z + m.m33;
    if(w!=0) ov.mult(1.0f / w);
    return ov;
}

```

中身はともかく、**`projectScreen`関数は、以下のように呼び出せばOK。**

```

PVector modelPos = new PVector(10, 20, 30);
PVector screenPos = projectScreen(modelPos);

```

`screenPos.x`、`screenPos.y` に それぞれ `screenX` や `screenY` 関数で計算したときと
ほぼ同じ値が入り、スクリーンに投影できていることが分かります。

これをふまえて、2D→3D変換の関数を作ります。

```

// スクリーン座標をモデル座標に変換
PVector unprojectScreen(PVector v) {
    PMatrix3D m = getModelToScreenMatrix();
    m.invert();
}

```

```
m.invert();  
return matrixCMult(m, v);  
}
```

違うのは関数名の他に 1 行だけ。

```
m.invert();
```

と、変換行列を逆行列にしているだけです。

unprojectScreen関数を使うときは、以下のように呼び出せばOK。

```
PVector mouse3D = unprojectScreen(new PVector(mouseX, mouseY, 0));
```

さらにunprojectScreenでZ値を1にした座標やカメラ位置とmouse3Dの差分ベクトルを使ったりすればマウス位置から3D空間に飛ぶ直線が得られます。

詳しいところはサンプルをいじって確認してみましょう。

本当は別の少しくレイジーなネタを書くつもりだったのですが、
そのためにこの変換処理の説明が必要だったので、こちらを記事にしちゃいました。

Tag : [Processing](#), [開発](#)

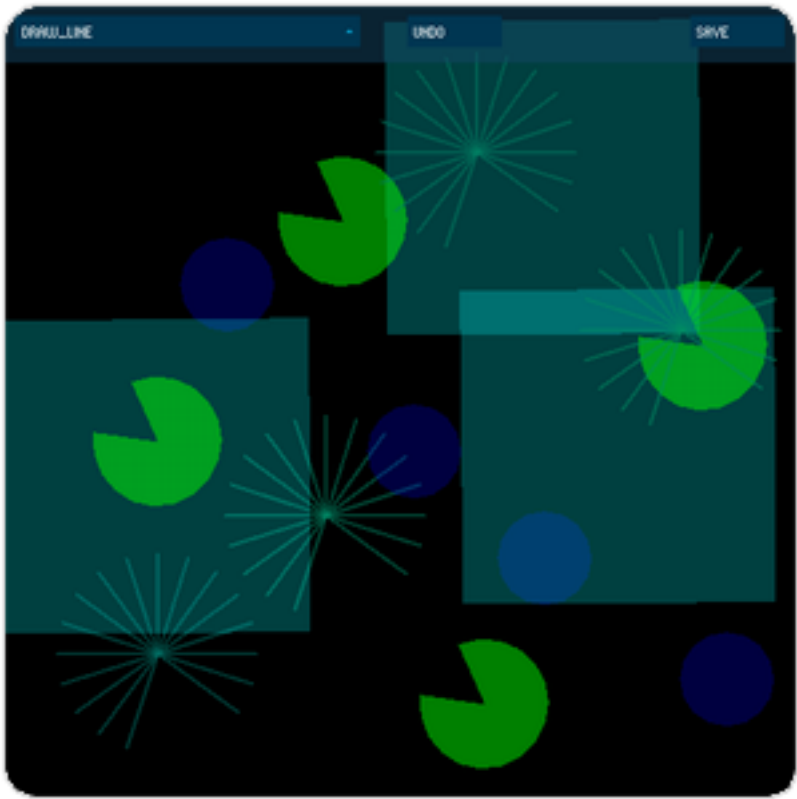
[Trackback\(0\)](#)

■ Processingでレイアウトエディタを作ろう

[前回](#)に続いて、[Processing Advent Calendar 2011](#)の12/15の記事を書きました。

今回は、レイアウトエディタもどきを作りたいと思います。

でも、いったい何をレイアウト（配置）するのか？
ひとことで言うなら、「**描画関数をレイアウト**」します。



[LayoutEditorサンプルのページへ](#)

ソースファイルは2つ。
描画関数を抽出したり、描画リストを管理する LayoutManager.pde と、
マウス位置にあわせて描画関数を描画リストに登録する LayoutEditor.pde です。

まず、setupの中で LayoutManager というクラスを初期化しています。

```
// 初期化
```

```

void setup() {
    size(512, 512);

    // 描画関数をレイアウトマネージャに登録
    layout = new LayoutManager(this, "draw_");

    setupGUI();
}

```

このLayoutManager クラスは、引数として渡された "draw_" という文字列で始まる関数を this (PApplet) の中から見つけてきて記録しています。

```

// 特定の名前で始まる描画関数を抽出
LayoutManager(Object classObj, String methodPrefix) {
    this.classObj = classObj;
    this.methodPrefix = methodPrefix;
    this.methods = new HashMap<String, Method>();
    this.items = new ArrayList<Item>();

    // 引数で渡されたクラス内で特定名で直接実装された関数を記録
    Method decMethods[] = classObj.getClass().getDeclaredMethods();
    for(Method m: decMethods) {
        if(m.getName().indexOf(methodPrefix)==0) {
            println("LayoutManager: add method : " + m.getName());
            methods.put(m.getName(), m);
        }
    }
}

```

関数を名前で探すためには、**クラス情報にアクセス**する必要があります。
 そのために[Java のリフレクション](#)という機能を使ってます。

getDeclaredMethods() は、直接宣言している全てのメソッドを返す関数です。つまり、親クラスのメンバは無視してるわけですね。まあリフレクションの乱用は禁物だと思いますが、なかなか面白いです。^[3]

記録した描画関数のテーブル（HashMap）は、LayoutEditorの描画関数選択や描画リスト追加で使っています。

なお、リフレクションを使うには、最初に以下のような import をしておく必要があります。

```
import java.lang.reflect.*;
```

- ^[3]ちなみに、名前で抽出というのはどうもなあ、と思ったときには、関数の前に「@～」と独自の印を付けられるアノテーションという機能を組み合わせるとよいでしょう。ただ、Processingの一見グローバルな位置にあるコードは、実際はProcessingのアプリアス内部のコードになります。独自アノテーションを定義するには、ライブラリ化やEclipse利用など、工夫が必要かもしれません。

さて、ここまでで描画関数全種類の記録はできました。あとは現在選択している描画関数を、クリックした位置にあわせて LayoutManager の描画リストへ追加するだけです。

```
layout.add(currentMethod, mouseX, mouseY);
```

この currentMethod は関数の名前を持つString型の変数です。

今回は [controlP5](#) というライブラリを使って、ドロップダウンリストやボタンを作っています。[GUI](#)については、サンプルコード、LayoutEditor.pde の setupGUI や controlEvent 関数を見てみてください。

リフレクションで取得した関数の実行方法は、LayoutManager の draw 関数内にあります。

```
// 登録した描画コマンドを発行
void draw() {
  for(Item it: items) {
    pushMatrix();
    applyMatrix(it.matrix);      // 位置などを設定
    try {
      it.method.invoke(classObj); // 登録してある描画関数の呼び出し
    } catch (Exception e) {
      throw new RuntimeException(e);
    }
    popMatrix();
  }
}
```

it.method.invoke(classObj); が関数を実行している箇所です。
try～catchの例外処理で挟まないとエラーになるのでご注意ください。

今回はよりクレイジーな雰囲気重視して（？）、リフレクション機能を使って生の関数を登録するようにはしていますが、引数やメンバの値をパラメータ設定して渡したり、クラスを使ったりするのもよいと思います。

たとえば、GUI部品レイアウトエディターとか作れそうです。

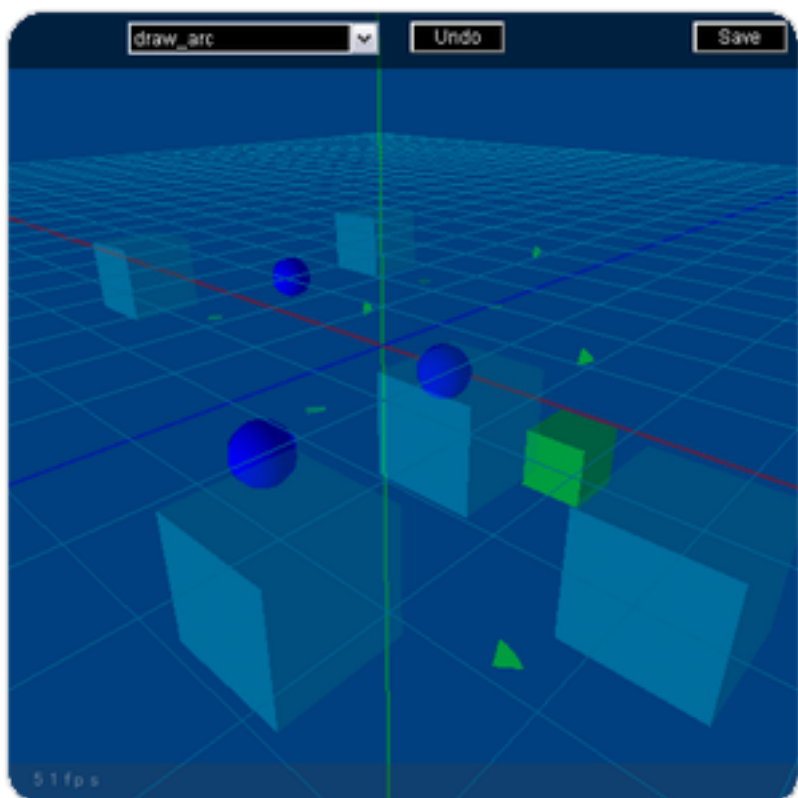
描画位置については、今回は位置だけなので applyMatrix ではなく translate で十分なのですが、回転やスケーリングもすぐに対応可能なように Matrix にしてあります。

そうそう、書き忘れていましたが、アプレット上では「Save」ボタンが使用できません。
ローカルで「Save」すると、LayoutEditor で追加した描画リストをProcessingのコードとして出力できます。

Processingはソースコードを書いて簡単に絵的な表現ができるのが楽しい言語ですが、逆に絵的な入力からソースコードを出すこともできる、と。

最後に**オマケとして3Dバージョンも作ってみました**ので、参考までにどうぞ。

こちらは controlP5 のP3Dでのテキスト描画に問題があったため、
かわりに SpringGUI という GUI ライブラリを使っていますが、やっていることは同じです。



[LayoutEditor3Dサンプルのページへ](#)

Utility.pde は[前回](#)の Unproject.pde 内から必要な関数を持ってきただけです。
LayoutEditor3D.pde は LayoutEditor.pde と Unproject.pde をあわせた感じで、
2Dを3D対応にして、カメラ回転や移動ができるようにしました。
LayoutManager.pde の中身は同じです。

えーと、今回はこんなところです。

昨今はソースコードと素材データだけでモノを作る作り方だけではなく、エディターと連携した作り方も増えてきています（といいつつ、実際は昔からそうすけど）。そういった**ツール作りの参考**になればと思います。

工夫すればUnityのような「動かしながらの調整」ができるリアルタイムエディタも作れるかもしれませんね。

本当は3Dの方を先に作ったのですが、余計なものが多いので2Dバージョンを作ったという。

Tag : [Processing](#), [開発](#)

[Trackback\(0\)](#)

■ Processingで3D物理シミュレーション

新年あけましておめでとうございます！
今年もマイペースに頑張りますので、どうぞよろしくお願いいたします。

そうそう、今年も[ミストの福袋を買いましたよ](#)。

じゃなくって、今年はプログラム記事もあります。

今回は[JBullet](#)（3D物理シミュレーションライブラリ）をProcessingから簡単に呼び出すPBulletというクラスを作ってみました。



[Physics3Dサンプルのページへ](#)

ライブラリ化はまだしてませんが、ひとまずサンプルをどうぞ。
新年ということで紅白カラーで縁起の良さをアピールしています。

使い方は簡単。setupなどで以下のようにクラスを生成します。

```
PBullet bullet = new PBullet(this);
```

オブジェクトの追加はこんな感じ。

```
// 球を追加(位置、半径、質量を渡している)

bullet.add( new RigidSphere(new PVector(0, 0, 0), 20, 2.0f) );
```

あとはdraw内で

```
bullet.simulation(frameRate);  
bullet.draw();
```

とすれば物理シミュレーションを行ったオブジェクトを描画できます。

このクラスを使うときは、Processingのスケッチフォルダの `libraries` に `jbullet` フォルダを作って、さらにその中に `library` フォルダを作り、[JBulletのサイトのjbullet-20101010.zip](#)にある `jbullet.jar` と `vecmath.jar` を入れておきましょう。（[ここ](#)に固めたものも作っておきました）

ついでに、MQOファイルをロードするクラスを若干拡張してMQOファイルから頂点データをもってきて剛体にするようなこともしてあります。

こんな感じで `RigidBody` クラスを派生したクラスで独自描画もできますし、`PBullet`の`draw`を使わずに、各オブジェクトの`applyTrans`を自分で呼んで独自描画をしたりもできます。

最初はAB法（ベルレ物理というらしい）を使った自前の物理処理を書いたのですが、
Bullet使えるならそっちの方が「使えるもの」になりそうだったのでこのとおり公開してみました。

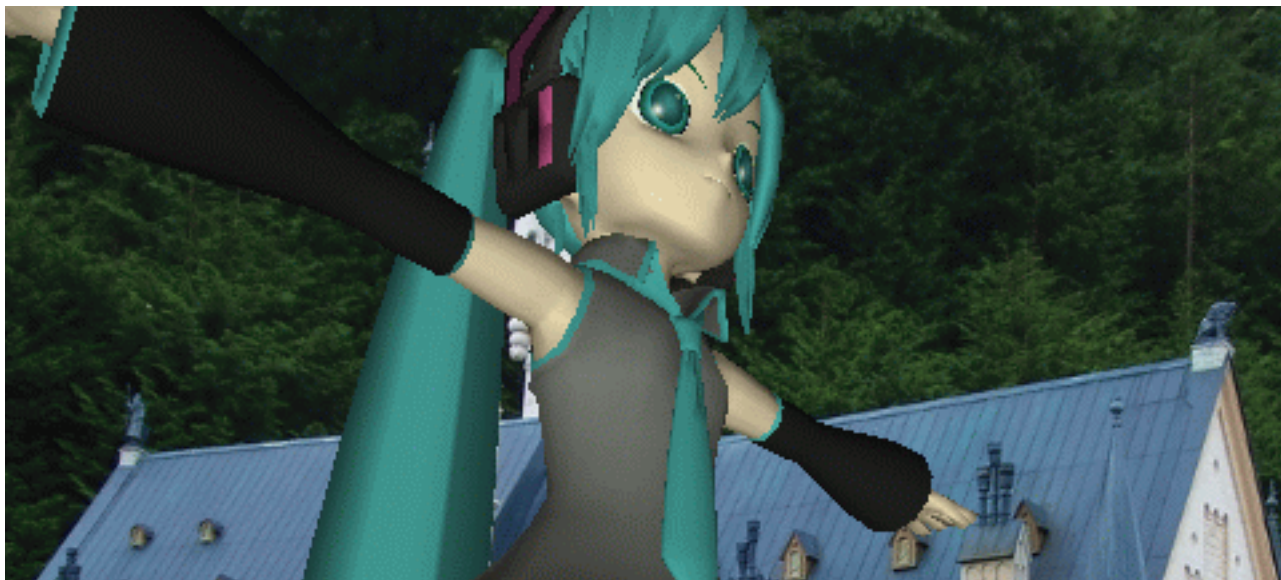
enjoy, it. それでは、また。

Tag : [Processing](#), [開発](#), [WritingCafe](#)

[Trackback\(0\)](#)

■ Processingでプルプル立体GIFアニメ

なんかすぐできるんじゃ？ と思って
Processingを使って立体感を強調（？）したGIFアニメ画像を作ってみました。





カメラの位置、被写体の位置、画像の細かさ、注視点で
立体感の出やすさがずいぶん違うようで、
試してみるといろいろと分かりますね。

とはいえこれ級の立体感には遠く及ばないですが。

この状態で自由に視点動かしたら立体感のあるゲーム画面になるのか？
というとそうでもなく、落ち着かないゲーム画面という印象：P

手前のmikuのモデルはTripshotsのspecialstuffsからお借りしました。
(ありがとうございます)

ちなみに後ろのお城は東武ワールドスクエアで撮影した、
ノイシュヴァンシュタイン城（ミニチュア）の写真です。

注視点はキャラモデルのそばに、
BGは遠めに配置するのがコツ。
(BGはP3Dモードだと歪むのでポリゴンを割ってあります)

東武ワールドスクエア、若干さびれてたけど、また行きたいなあ。

Tag : Processing

Trackback(0)

■ ゲームの中に入って遊ぼう - Kinect Sky Labyrinth

8月にQUMI Q2という小型プロジェクターを買ったので、このプロジェクターとKinectを使って遊べる
ゲームを作ってみました。

床に広げたただのシートが、突然広々とした空が広がる世界になり、自分がゲームの中に入って遊んで
いるような感じになります。

映像、手持ちのiPhone5で撮ったので見づらいですね。暇があったら撮り直したい。

最初に位置あわせのためのキャリブレーションを行っています。固定設置だったらいいけど、毎回動かしてるので。でもこんなやり方でいいのかな。

ゲームのルールとしては、ブロックからブロックに飛び移って遊び、たくさんブロックを踏めるほどよい、というもの。

映像は手抜きで立方体しか使ってないです。シーツのアイロンがけの方が面倒で.....。

でも、Kinectで遊ぶのは本当、楽しい。以前[Kinect Magic Knight](#)を作ったときもそう思ったけど。先週、体調悪いのに楽しすぎて深夜作業してしまったのはアレですが。

プログラムはProcessing + SimpleOpenNIを使って書いています。工学ナビの[Processingで手抜きプロジェクト](#)も参考にさせていただきました！

微妙すぎて分からないと思いますが、プレイヤーの位置にあわせて、見える景色に微妙に角度をつけたりしているので、歩くだけでも多少奥行き感があります。これは以前作った[Window Simulator](#)と同じ手法ですね。

ただ歩くスペースがない。もう少し広い部屋でやってみたいものです。

Tag : [Processing](#), [Kinect](#), [開発](#), [ゲーム](#)

[Trackback\(0\)](#)

■ アイデアがどんどん湧いてくる!? Processingで作るスマートフォン向けWebアプリ

今年も[Processing Advent Calender 2012](#)に参加しました。本日12/6のテーマは適当に「スマートフォン」とだけしていたのですが.....ちょうどいいネタがない！

～ここから回想～

思い起こせば、小学生の頃のこと。「しんはつめいじどうぼちぼちき」という絵本を読みました。記憶の中での絵本の内容は、「すごい機械を発明したので、その機械にさせたいことを募集したところ、大量の手紙が届き、じどうぼちぼち機は大量の手紙を自動処理する機械になった」というもの。

それが現在郵便局で使われている機械だと思うと非常に感慨深いですね。

ーではなくて、ネタに困った結果、前々から作りたかった**ネタを作るためのWebアプリ**を作ってみました。

iPhone / Androidなどのスマートフォン向けです。が、iPhone5、iPhone3GSでしかテストしてません。Androidはエミュレーターで少しだけ。



上記URLにスマートフォンからアクセスするとWebアプリを確認できます。



Shuffleボタンで3つのキーワードすべてをシャッフル。

各キーワードは上下にドラッグして入れ替えたり、左右に動かして個別に新しいキーワードと入れ替えたりもできます。

アイデアの生み出し方について考えた古典「アイデアの作り方」で、ジェームス・W・ヤングは次のように語っています。

アイデアとは既存の要素の新しい組み合わせ以外の何ものでもない。

ー ジェームス・W・ヤング著、今井茂雄訳 『アイデアの作り方』 阪急コミュニケーションズ、1988年、28頁



そう、新しい組み合わせこそが、新しい発見であり、アイデアなんです。

さあ、これで湯水のごとくアイデアが湧き出てきますね！

新しいアプリ、新しい料理、新しい政党、クリスマスプレゼントでもなんでもござれ！！

……と言い切るにはナンセンスなものですが、作りながらひとりで笑えたのでこれはこれでよし。

ソースコードはこちら [IdeaMan.pde](#)。200行ちょっとです。 [index.html](#) もリンク先を保存するか、アクセスしてソースコードを表示すれば確認できます。

※UTF8なので文字化けする方はダウンロードして対応エディタでご確認ください。

構成を単純にするためキーワードのテキストデータもソースに入れてます。

データはまだ100個+αぐらいしか入れてないので増やすと面白いですよ。

さて、ではこのアプリのようなスマートフォン向けのWebアプリ作り方をピンポイントで説明しましょう。

1. Processing 2.0をダウンロード

いまはProcessing 2.0 beta 6を[ダウンロード](#)できますね。

2. JavaScriptモードでプログラムを作成

Processingを起動したあと右上にあるモードのボタンでJavaをJavaScriptに変更します。

iPhoneとAndroidでどうも画面拡大率が同じように書けなかったので、以下のように分岐して画面サイズを決めました。

```
void setup()
{
    size(640, 1136); // PCでのテスト用。iPhone 5相当

    if( navigator.userAgent.indexOf("iPhone")>0 ||
        navigator.userAgent.indexOf("iPod")>0 ||
        navigator.userAgent.indexOf("iPad")>0 )
    {
        size(screen.width * 2, screen.height * 2);
    }
    else if( navigator.userAgent.indexOf("Android")>0 )
    {
        size(screen.width, screen.height);
    }
    ...
}
```

PC用のものが余計ですが、PC用のsizeを最初に書いておかないとprocessing(PDE)上でのテストがうまくいかないため、こんなありさまに。

でも、あとは普通にProcessingでプログラミングするだけ！

3. 実行するとできるweb-exportフォルダのindex.htmlにスマートフォン向けのmetaタグを追加

幅とか拡大率とか、拡大縮小禁止とかを指定しました。

Android用にtarget-densitydpi=device-dpiと指定しています。

```
<meta name="viewport" content="width=device-width, target-densitydpi=device-dpi, initial-scale=0.5, minimum-scale=0.5, maximum-scale=0.5, user-scalable=no" />

<meta name="apple-mobile-web-app-capable" content="yes" />
```

iPhoneのHome画面にブックマークを登録するときのWebアプリ用のアイコンを指定する場合は以下のような感じで。

```
<link rel="apple-touch-icon" href="icon.png" />""
```

あと余計なcssなども取り除いています。

それから、web-exportフォルダのままだと上書きされるので、書き換えたindex.htmlは別のところに退避しておくとういでしょう。

4. ブラウザのURLバーを見えなくする&スクロールを禁止に

そうするためには強引にウィンドウをスクロールさせるらしいので以下のようにしてみました。

```
<body onLoad="window.scrollTo(0, 1); setTimeout(scrollTo, 500, 0, 1);"
ontouchmove="event.preventDefault()"">
```

でもダブルタップするとスクロールしちゃいますが。どうにかならないのかな？

5. Webサイトにアップロード

自分のWebサイトがないときには、[Dropbox](#)のpublicフォルダに置いて、index.htmlのURLリンクを公開（長いときは短縮URLにして）という技もあります。

以上で完了です。

調整したいところはまだ色々あるのですが、ひとまずお試しください。

もしmetaタグやスクロール、canvas側のサイズなど、iPhone/Android両対応でうまいやり方をご存知の方がいましたら、ぜひ [twitter @n_ryota](#)、[BBS](#)などにコメントください。

12/25 追記

このWebアプリをもとに[ProcessingでAndroidアプリを作成しました](#)

Tag : [Processing](#), [HTML5](#), [iPhone](#), [Android](#), [アプリ](#), [アイデア](#), [面白い](#)

[Trackback\(0\)](#)

■ いっきに作ってしまおう。ProcessingでAndroidアプリを作成&公開する方法

今月は5年ぶりにPCを新調してMac Book Airを買い、[Parallels](#)でWindows8環境を作ったりして、いっきに散財してしまい、しばらくは節電ならぬ節ガジェットだなあと諦めかけていたのですが.....クリスマスプレゼントとして[Nexus7](#)を貰いました。これはー嬉しい。



その記念というか、無駄な勢いというか、[前回のWebアプリ](#)をほぼそのままにAndroidアプリとしてリリース！ 今回はその手順をご紹介します。

[アイデアがどんどん湧いてくる！？ Idea Man \(Google Play - Androidアプリ：無料\)](#)



processing.jsとWebViewを使ったアプリではなく、Javaコードのアプリとして作成しています。

テストした開発環境はWindows 8+Nexus7(Android)です。

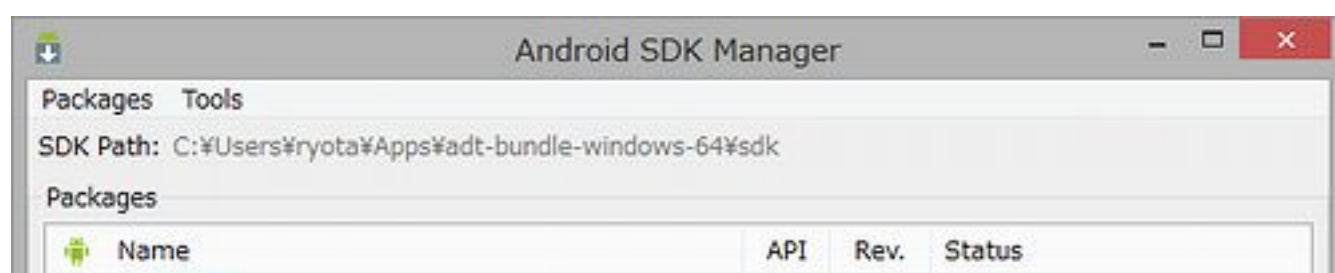
1. Javaモードで動作チェック

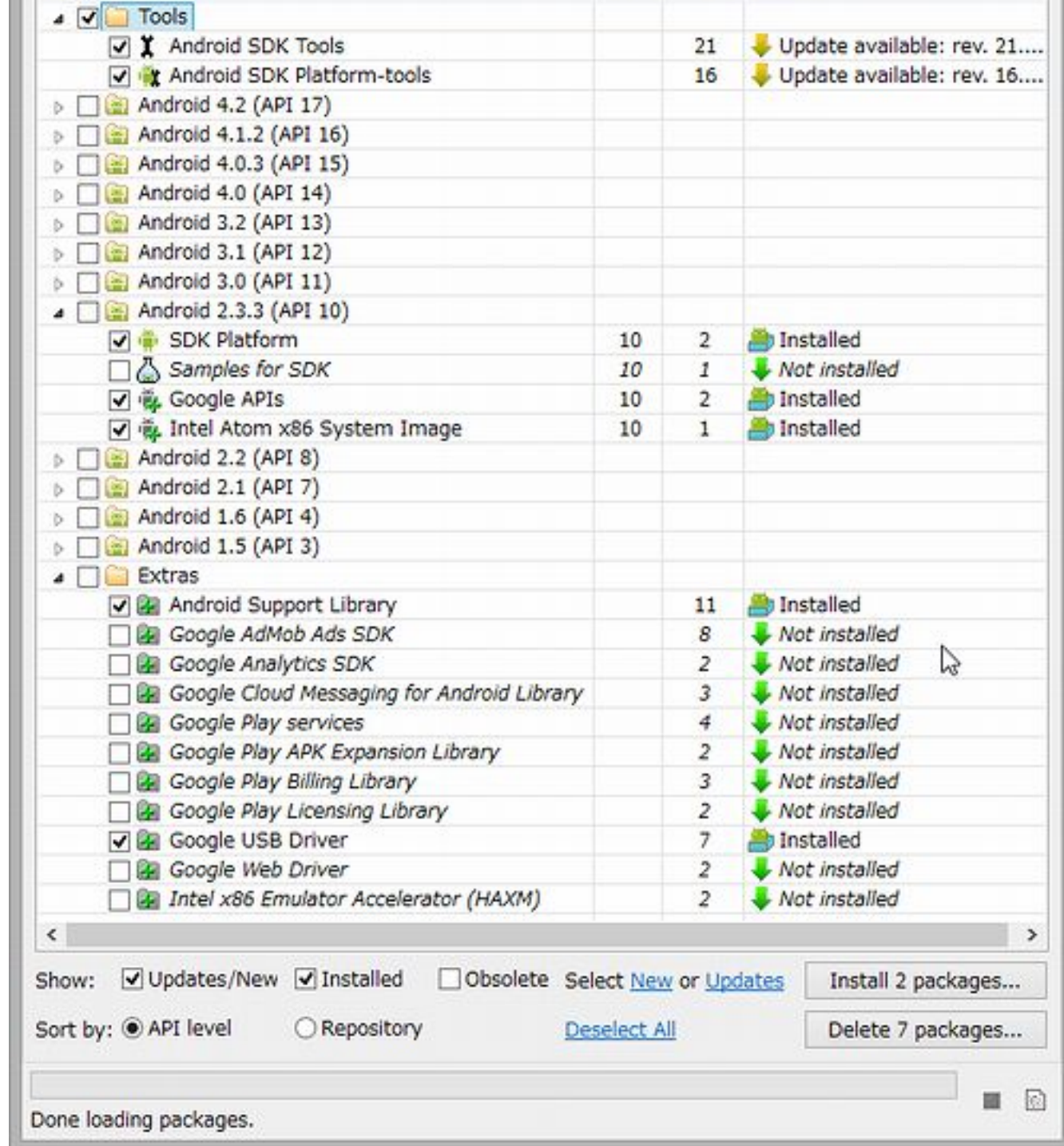
processingのスケッチ（コード）を普通にJava(Standard)モードで動作確認します。
問題なければ次へ。

2. Android SDKをダウンロードしてProcessing - Androidモードの環境設定

Androidモードを使う場合、Android用のSDK（ADK）が必要なので
[Googleのサイトからダウンロード](#)してProcessingにパスを設定します。

以下のようなパッケージをインストールしておきましょう。





※バージョン違いなどでうまくいかない場合は、古いAPIや最新のAPIのパッケージのインストールもお試ください。

もしAndroid SDKのパスの指定を間違えたときは、

ProcessingのFile->Preferenceで開くダイアログの下部に書いてある preferens.txt の

```
android.sdk.path=xxx
```

を編集してProcessing(PDE)を再起動しましょう。

3. Androidモードで実機転送

エミュレータを使用する場合はProcessingのメニューのAndroid->AVDManagerで設定や起動ができます。が、Androidの開発でエミュレータを使うのはオススメしませんので説明は割愛。

以下は実機、つまり実際のAndroid端末を使う場合の手順です。

まず、Android端末内の設定画面で、開発者向けオプションを表示します。

Android OS 4.2以降を使っている場合など、開発者向けオプションがない場合は[こちらのサイト](#)などを参考に設定->タブレット情報->ビルド番号を7回タップすると、開発者向けオプションが表示されるようになります。

設定->開発者向けオプションの「USBデバッグ」（USB接続時はデバッグモードにする）をONにしてください。

また、OSのバージョンによっては、

設定->アプリケーションの「提供元不明のアプリ」（サードパーティアプリケーションのインストールを許可する）をONにする必要があるかもしれません。（Nexus7 OS 4.2.1ではセキュリティに「提供元不明のアプリ」がありましたが、特にONにする必要はありませんでした）

あとはAndroid端末をUSBケーブルでパソコンにつないで、実行.....なのですが、

Android端末開発用のUSBドライバーをインストールする必要があります。

各メーカーのサイトからダウンロードしてインストールするか、
「2」でPCにインストールしたパッケージ内の、

```
sdk\extras\google\usb-driver
```

にある該当ドライバ(android-winusb.inf)をインストールしてください。
手動の場合はこのファイルを選択して右クリックメニューで「インストール」。

Nexus7でしか試してないので他の端末で正しいかどうかはよく知りません。

デバイスマネージャで見たときにその端末の「！」マークが無くなっていればOKです。

あとはProcessingをAndroidモードにしたまま、通常通り再生ボタン（Ctrl+R）でスケッチを実行すればOK。無事起動したかな？

ちなみにCtrl+Shift+Rでエミュレータでの実行ができます。

実行したときに画面サイズが変だなと思ったら、

```
void setup(){
    size(displayWidth, displayHeight); // Android用
```

とすると端末の画面いっぱいに表示できます。

ただ動かすだけならここで完了ですが、
アプリをGoogle Playなどのストアで公開したい場合は以下の手順に進んでください。

4. ビルドの準備（antのダウンロードとパスの設定）

ビルド用に[Antをダウンロード](#)して、フォルダに解凍。
[.zip archive: apache-ant-1.8.4-bin.zip](#)とか書いてるヤツです。

環境変数のPATHにantのbinフォルダをセミコロン(;)で区切って追加します。

```
例)
antを置いたフォルダ\apache-ant-1.8.4\bin
```

後述するkeytoolとjarsigner用にjdkのbinも、PATHに追加します。

```
例)
processingを置いたフォルダ\processing-2.0b7\java\bin
```

同じく後述するzipalign用にAndroidSDKのtoolsも、PATHに追加します。

```
例)
androidSDKを置いたフォルダ\adt-bundle-windows-64\sdk\tools
```

以上で環境変数への設定は完了です。

5. apkファイルをリリースビルド

あとは[Processing Forumのやり取り](#)を見てやってみたところできたという手順です。

なおそのうちProcessingでExport Signed Packageがサポートされるようになれば
このような面倒な手順は省略できるようになると思います。

まず、Processing(PDE)のメニューのFile->Export Android Projectを選択すると
スケッチのあるフォルダにandroidというフォルダが作られてプロジェクトが出力されます。

コマンドプロンプトを起動して、さきほど作ったandroidフォルダに移動し

```
ant release
```

を実行すればビルド完了です。

「スケッチ名-release-unsigned.apk」という名前のファイルができていると思います。
読んで字のごとく、このapkはまだ未認証のapkなのでストアには公開できません。

ちなみにスケッチ直下（androidフォルダの一階層上）に

- icon-48.png
- icon-72.png

といった名前でそれぞれのサイズのアイコンを用意してExportすると、アプリのアイコンを変更できます。
アイコンやストア用のスクリーンショットなどのガイドラインは [こちら](#)。

6. keytoolで認証キーを作成

スケッチフォルダ直下で以下のコマンドを実行してください。

```
keytool -genkey -v -keystore スケッチ名-release-key.keystore -alias エイリアス名 -keyalg  
RSA -keysize 2048 -validity 10000
```

スケッチ名はスケッチフォルダ名そのもの（大文字小文字も同じで）、エイリアス名はあなたの名前とかドメインとかでしょうか？

実行するとパスワードとか所在地とか色々聞かれるので答えていきましょう。

スケッチ名-release-key.keystoreというファイルができればOKです。

7. jarsignerとzipalignで認証済みapkファイルを作成

スケッチフォルダ直下で以下のコマンドを実行してください。

```
jarsigner -verbose -keystore スケッチ名-release-key.keystore android\bin\スケッチ名-  
release-unsigned.apk エイリアス名
```

問題なければ、さらに次のコマンドを実行して、
アライメントをそろえて認証済みapkファイルを作成してください。

```
zipalign -v 4 android\bin\スケッチ名-release-unsigned.apk android\bin\スケッチ名apk
```

最後のファイル名は好きな名前でOKです。

これでストアに公開可能なapkファイルができました！

8. Google Playにデベロッパー（開発者）登録

[Google Playのデベロッパー登録の回答ページ](#)にある通り、
[Google Play Developer Console](#)にアクセスしましょう。

メールアドレスや電話番号などのプロフィールを記載して、
初回費用として\$25払えば完了です。

9. アプリ(apk)やスクリーンショットをアップロードして公開

アプリの管理ページでアプリ名を入力したあと、apkファイルやスクリーンショットをアップロードしたり、アプリの説明を書いたりしましょう。



必須のものをすべてを用意して保存すれば、数時間後にはGoogle Playに公開されます。

おつかれさまでした。

以上、全部まとめると長いですが、一度やってしまえばあとは簡単ですね。
コマンドはバッチファイルにしてもいいですし。

なお、アプリ内で使用する関数によっては Processing(PDE)のAndroid->Sketch Permissionsで AndroidManifest.xmlの設定をする必要があるかもしれません。

その他ProcessingでAndroidを使う際の設定については[Processing for Android Wiki](#)や[Arduino Wikiのこちらのページ\(日本語訳\)](#)などが参考になります。

IdeaManアプリの中身の方は、新しいデータを入れないと見飽きてしまって面白くないし、暇があればもうちょっとまともに作りたいところです。

Tag : [Processing](#), [Android](#), [開発](#), [ツール](#), [アイデア](#)

[Trackback\(0\)](#)

■ Processing.jsの画像の透過処理を高速化

いつもTwitterのつぶやきを拝見させていただいている [@clockmaker](#)さんがHTML5のグラフィック系 JavaScriptライブラリのパフォーマンス検証をされていて、興味深いです。

[HTML5開発者必見、最速のJavaScriptライブラリはどれだ!? パフォーマンスの徹底検証](#)

しかし、Processing.js (ProcessingのJavaScript版) がやけに遅いのが気になりました。tint()を使わなければ速いようです。

たぶん、ライブラリ側で正直な実装をしているせいなんだろうな...と思いながら、processing.jsのソース見てみるとピクセル単位で色を合成していることが分かりました。tint()は画像を好きな色にフィルタリングできるので、先にそういう画像を作ってからdrawImageしている、と。HTML5のCanvasには globalAlphaなるものがあるようなので、透明度だけを変えているときにはこれを使うようにハックしてみるとー

- [tint高速化版](#)

- [通常版](#)

6fps→24fpsと4倍ぐらい高速になりました。

ライブラリのソースは[ここに置いてあります](#)ので、ご自由にお使いください。

あとProcessingも初期化時に指定すればWebGL使えるはずですが、パフォーマンスは謎です。

Tag : [Processing](#), [HTML5](#)

[Trackback\(0\)](#)

ゲーム作りで冒険！「遊んで作るスマホゲームプログラミング」

長ーい執筆期間を経て、ようやく本が完成しました。

「**遊んで作るスマホゲームプログラミング for Android**」というタイトルの本で、ゲーム作りを楽しく学べる冒険の手引きです。「for Android」とあるとおり、Androidネイティブアプリを作ります……なのですが、iOSでも動くHTML5ゲーム、PCで動くゲームやツール、さらにはKinectまで扱うという、脱線っぷり。

本日、秀和システムから発売です。



[Amazonの書籍ページを見る](#)

~~ただオンライン書店のステータスが「取扱いしていません」に変わってるところも多いようなので、再入荷に少し時間がかかるのかも。~~ また注文できるようになったようです。あと、もし奇遇にも実際の書店に並んでたら、パラパラめくっていただけると嬉しいです。

開発環境には[Processing](#)を使っていて、ソースコードも短めなので、気楽にゲーム作りをしたり、改造したりできます。巻末にはプログラミング入門的なオマケ付き。

書籍とサンプルゲームの紹介動画はこちら！

これまでこのホームページでもProcessingのプログラムを公開してきましたが、実はこの本のためでもありました。そうーそんなに前からこの本を書いていたのです。でも、ホームページで公開したおかげで、アドバイスをいただいたり、Processing界隈の知り合いができたり、理解も進んで、読者の方に伝える内容をよりよくできたのではないかと思います。

ここで公開していないプログラムも本書には収録しています。公開しているものも、バグを修正したり、機能を発展させたり、Processing 2.0やAndroid、HTML5対応したりといろいろ変わっています。そ

して、ホームページにはあまりなかった解説的なものも、本書にはあります。

プログラミング以外に、ゲームのアイデアの発想、企画作り、データ制作、ストア申請、プロモーションなど、ゲーム作り全体を「**楽しく作る**」をコンセプトに書きました。

詳しくは特設サイトの[EYLN -ゲーム作りの冒険-](#)をご覧ください。ページの雰囲気が分かる写真や、動作環境、小話などもこちら。

この本を読んで誰かがわくわくしてくれたり、何か少しでも役に立てていたらいいなと思います。

本を書く間、「時間の使い方」「ノマド」「楽しいゲーム作り」「原稿を書くツール」「出版までの流れ」など、色々と気付いたこともあって、そういうことも共有すると楽しそう＆何かの役に立てそうなので、またちょこちょこそのあたりの話をするかも。

Tag : [processing](#), [ゲーム](#), [開発](#), [本](#)

[Trackback\(0\)](#)

■ イチから簡単なアクションゲームを作る15分ぐらいの動画

夏休みの特別企画！

イチから横スクロールアクションゲームを作る、15分ぐらいの解説動画です。

プログラミングして完成させるソースコードは100行。

でも全部手入力で途中間違えて修正したりもしてるのでそこそこ時間がかかりました。
倍速再生なので実際は30分ぐらいです。

ただ、ムービー作成の方が時間がかかった…。実際は夏休みでもなく非常に忙しかったので、深夜に何日かけて準備しました。忙しいときこそ、仕事と家庭と趣味を。

ソースコードや実際にWebブラウザ上で動作するデモは[こちらのサイト](#)からどうぞ。

[Trackback\(0\)](#)

■ お風呂でスクリーンに浸かる！ プロジェクションマッピングに挑戦

今年も[Processing Advent Calender 2013](#)に参加しました。今回はごく単純なプロジェクションマッピングの話題です。やり方は[工学ナビ：Processingで手抜きプロジェクションマッピング](#)を見ていただければOK！

プロジェクションマッピングの歴史については[いまさらプロジェクションマッピングを追う！](#)の記事が分かりやすいですので、ぜひご覧ください。

――以上――

というわけにはいかないので続きます。

でも、これから紹介するものはそんな大層なものでもなく、Processingのわずかなプログラムとプロジェクター、そして風呂があればできる単純なものです。

風呂……そうです。風呂の水面に映像を投影する、俗にいう風呂ジェクションマッピングというやつですね。

プロジェクターはなんの湿気対策もなく素っ裸で配置するので、スリル満点というか、壊れそうなのが懸念でしたが、扉のそばだったのでそれほど湿気はありませんでした（短時間では）。



むしろ高い位置から投影するために椅子の上に三脚を乗せたりしてたのが危なかったです。以前、Kinectをカーテンレールから落として、こんな有り様にしてしまった戦いの記憶が蘇りました。





※今回、Kinectは使用しません。KinectをProcessingから使うサンプルゲームは [遊んで作るスマホゲームプログラミング](#)などで解説してます（for Androidなのに）ので、興味がありましたらぜひ。

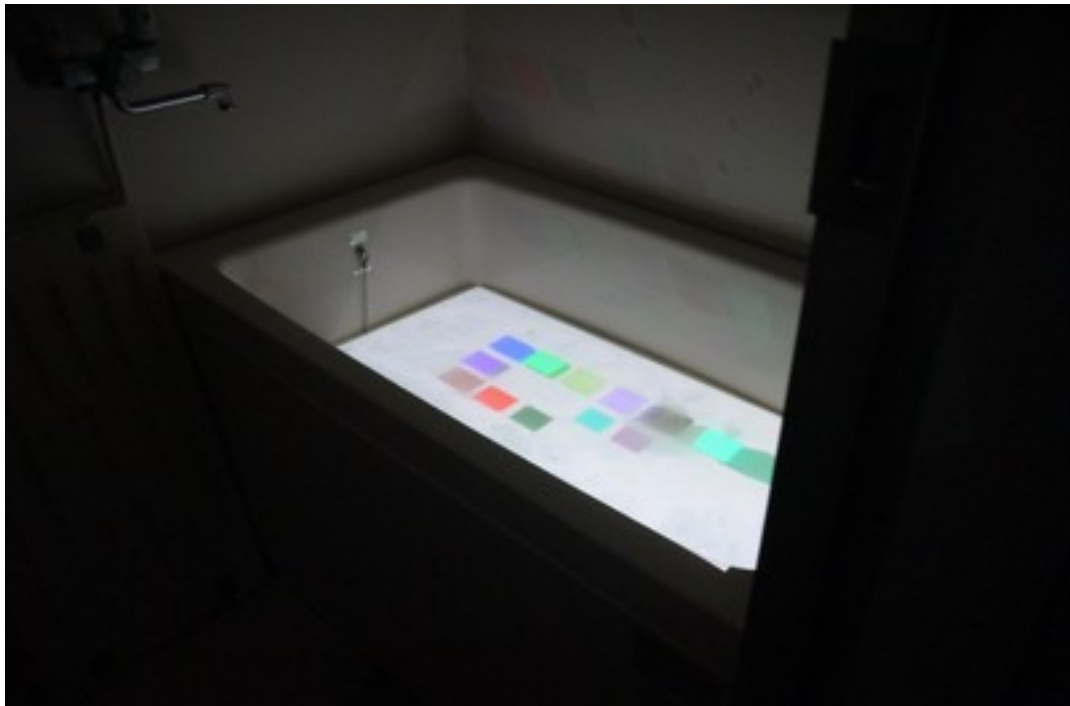
お風呂の水が透明な場合は、バスタブの底面や側面に映ります。

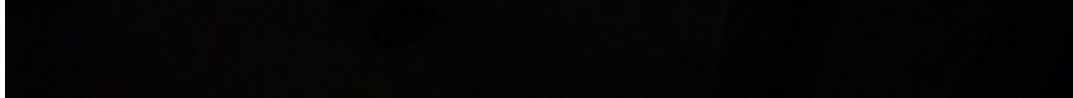
水面に映したい場合は、しずかちゃんの懂れる牛乳風呂にするとか、[薬用ソフレ](#)などの入浴剤をたっぷり入れるなどして、なるべく白い面を作っておくと綺麗に表示されます。



スクリーンの角を右ドラッグ（Macの場合はCTRLを押しながらドラッグ）すると、スクリーンの形状を変形して、お風呂の水面の形にあわせられます。最初にプロジェクターでお風呂の水面全体が映るように三脚その他を駆使して頑張りましょう。

本来は真上から真下に向けて投影するのがベストですが、そうするのは設置が難しく、湯気、落下、感電、その他悲劇が待っているかもしれませんのでご注意を。





真っ暗にしなくても映ってるのは見えましたが、分かりやすいようにしてテスト。
水面が光ってること自体が新鮮です。

マウสดラッグでカラフルな球を描けます（座標は変形前の座標）。

（撮影時に使った前バージョンのプログラムはカラフルな箱でした）

なんてことはない、単に水面の色が変わったり、絵や模様が出たりするだけなのですが、
実際にお風呂で浸かってみると、なんか楽しいです。

スペースキーでモード切り替え。

「なにもしない→タイル状に白と赤の箱を表示→ランダムに箱を表示→ランダムに球を表示」
を繰り返します。

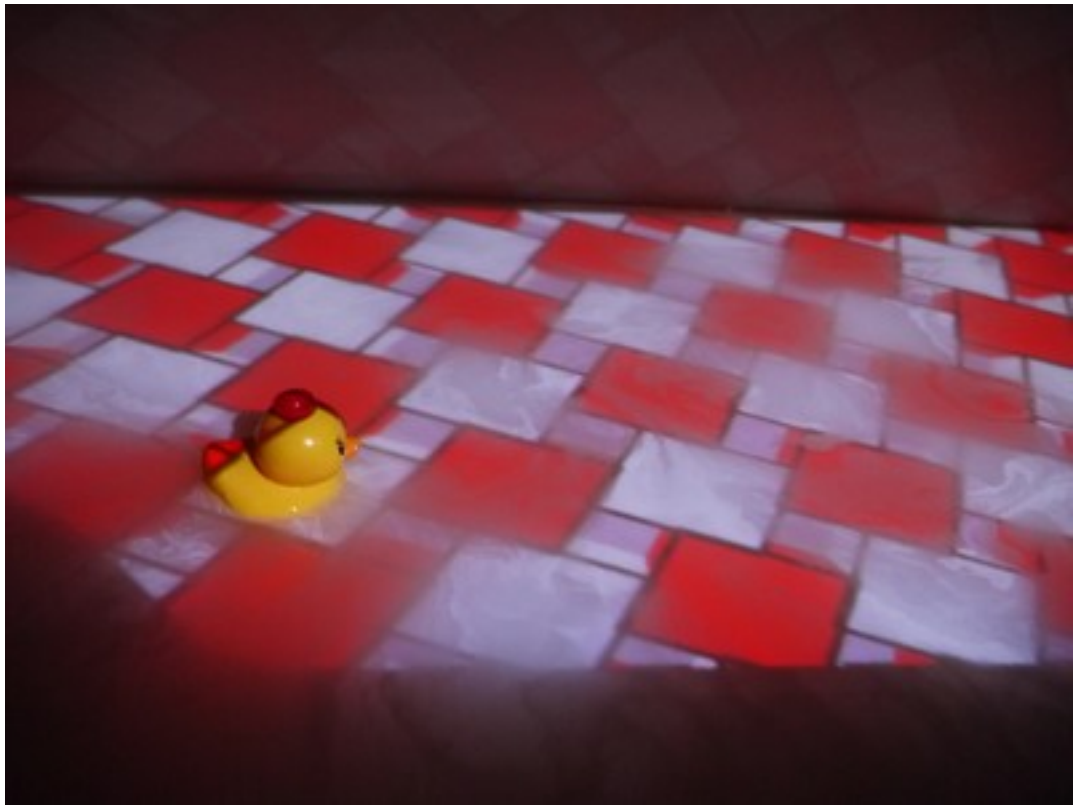


水面に絵が出るなんとも言えない違和感があり、自分がその中にいて色を触るとぐにゃぐにゃ変形したり、不思議な感覚です。

こういう銭湯があってもいいんじゃないかと思ったりしました。

※レーザーショーのある温泉施設とかはあります。

あと同時に窓シミュレーターのように壁にもバーチャルな窓を作ったり、地平線テープみたいな謎の空間にするのも面白そうです。



プログラム冒頭のIS_DEVをtrueにすると（デフォルトでtrue）、風呂の背景画像の上にスクリーンを変形して投影しているような、開発用のテストモードになります。

これを使うと、プロジェクターが無くてもプロジェクションしている気分に。



Bキーで背景画像切り替え。dataフォルダに入っているscreenBG0～6の画像を順番に切り替えます。

プログラム冒頭のIS_FULLSCREENをtrueにすると（デフォルトはfalse）、フルスクリーンでプログラムを起動します。
プログラムはESCキーまたはQキーで終了させられます。



Processingで書いたプログラムは[こちら](#)。250行ほどです。

最後に、動かしている様子を動画で簡単にご紹介。

クリスマスツリーの飾り付けのようにしたり、キラキラ輝く南の海のようにしたり、
青の洞窟のようにしたり、札幌風呂にしたり、
溶岩風呂や、クリームシチュー、大理石に入浴したり、
ミクでも、目玉おやじでも、スライムでも好きなキャラと入浴したり
サメが追ってくるようにしたり、
人や血が湧き出てくるようなホラーテイストにしたり、
いろいろ応用できると思いますので、気が向いた方はぜひ。

enjoy it!

続報

2014年4月25日

フジテレビ・スーパーニュースでバージョンアップ版をお披露目しました。

2014年7月28日

風呂の水面+浴槽にも投影する、風呂ジェクションマッピング2を自宅で撮影して、
動画を公開しました。

Tag : Processing, 開発, ツール, 遊び

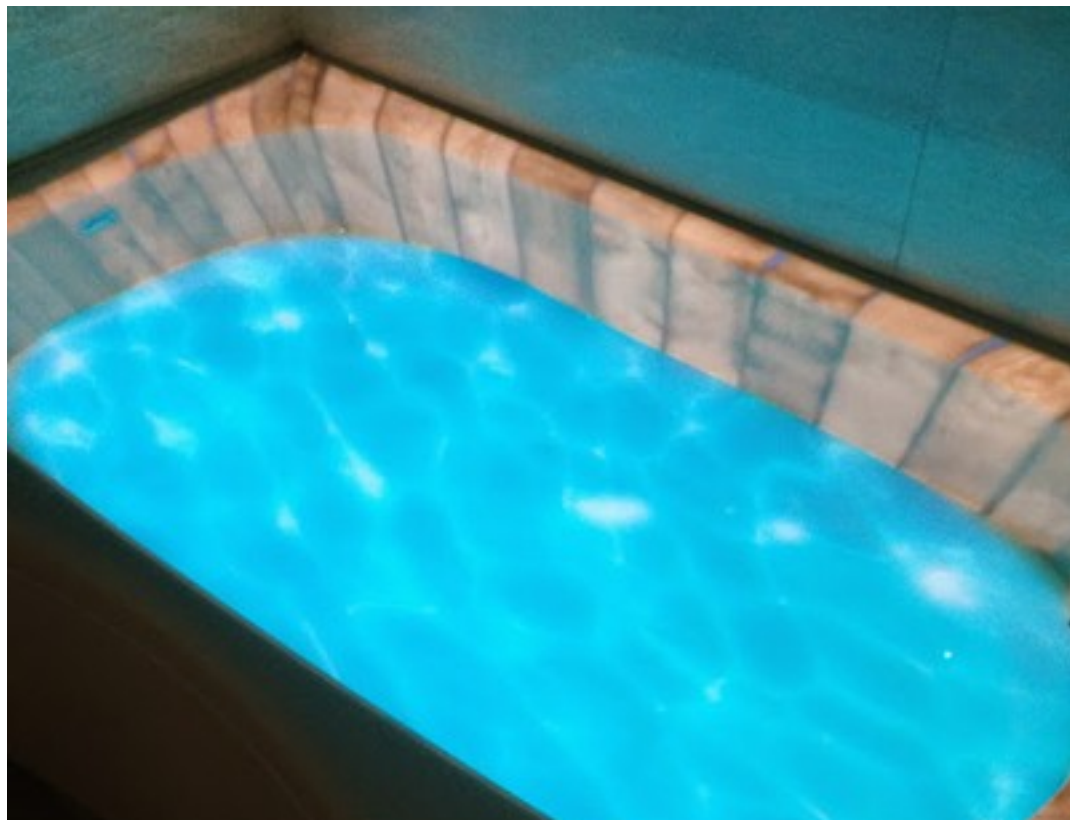
Trackback(0)

テルマエと風呂ジェクションマッピング

テルマエ・ロマエII公開を記念して、テレビでお風呂グッズを紹介するコーナーがあったのですが、なんと、そこに風呂ジェクションマッピングが。

少し前に体験取材したいというメールをいただいていたのに、メールにテレビ局の名前がある時点で本当の依頼なわけがないと思い込み、スパムメール扱いでスルーしていたという...（汗）。

でも放送予定の3日前に気づいて、打ち合わせて、夜に動作チェック&プログラムのバージョンアップ。





以前のバージョンだと水面だけのプロジェクションだったのですが、今回は浴槽の側面にもマッピングできるようにしました。そのうえで、頂点情報もセーブ・ロードできるようにして、現場でのトライ＆エラーが早く終わられるように準備。

あと、浴槽の角が丸いのに合わせて平面の形状も丸みをもたせるようにマスクを指定可能にしておいたのは、よかったです。

撮影は放送の前日、会社に行く前の午前中に貸しスタジオで収録を行うというスケジュールでした。

普通プロジェクションマッピングをするときには、前もって物体の形状を詳細に計測したあとに行うと思うのですが、当日ぶっつけ本番のプロジェクションだったので、「なんかこの浴槽、かなり丸いぞ」とその場でマスクを調整（rect関数の丸みの数値を調整）、Processingのおかげで数分もあれば近い感じにできました。

そんなこんなで、昨日フジテレビのスーパーニュースにて無事放送されました。



番組上では最初「いったい何が始まるのでしょうか」とまさかのクイズ形式でスタート、Processingの画面やプロジェクター、MacBookAirの画面を写して尋ねるという無茶ぶり。

「えっ、お風呂がいろいろ変わるってこと？ なるほどこれはいいですね、毎日違うタイプのお風呂に入れるんでしょ」と、あのテルマエのルシウス(阿部寛)さんがコメントしてくださるとは...なんとありがたいことでしょう。

プログラムが手抜きで自動処理があまり多くなかったため、現場では風呂VJみたいな状態。

しかし古代ローマで再現するのはどうしたらいいんだろう。



さらに「なんだっけいま流行りの...もう、お子様がお風呂に飽きないですね、ずーっと入っちゃう」とマミ(上戸彩)さん。収録してよかったです

ちなみに、このお風呂にはアナウンサーの榎並大二郎さんが入ってます。

実際のところ技術的にも発想的にも特別なものではないですが、そんなことは気にせず、思いついたことを（駄洒落でも）形にしておくと、たまにこんなことがあったりするので、面白いですね。

新しいバージョンの動画も暇ができれば、作成したいところです。

■ 続報 2014-7-28

ニコニコ動画に、新しいバージョン（4月当時のものと同じプログラム）で撮影した映像を投稿しました。興味のある方はぜひ。

Tag : [Processing](#), [ガジェット](#), [面白い](#), [遊び](#), [嬉しい](#)

[Trackback\(0\)](#)

■ Unityで簡単プログラミング、Processing気分を味わう

こちらで書くのをすっかり忘れてましたが、UnityでProcessingっぽく書けるライブラリ（アセット）を作りました。

オリジナルのProcessingにはないprefab()とかloadScene()みたいな関数も追加してあって、UnityChanを使った簡単なアクションのサンプルも入っています。

ただし、現在移植してあるProcessingの関数は一部で、最適化などはあまりされていません。移植といってもProcessingの元のコードを見たりはしないので、仕様が違うものもあるかも。

UnityのLowLevelAPIで描くのではなく、Hierarchyに普通にGameObjectを生成して使う構成なのが特徴です。ワイヤースケッチだけはGLを使っちゃったけど。

[GitHubのuProcessingリポジトリ](#)に一式あげてあるので興味のある方はどうぞ！

矩形と丸を描くサンプルコードはこんな感じ。

```
using UnityEngine;
using System.Collections;

public class Primitives : PGraphics {

    protected override void setup() {
        size(512, 512, P3D);

        stroke(0, 128, 64);
        strokeWeight(20);
        fill(0, 255, 128);
        rect(20, 20, 300, 400);

        noStroke();
        fill(255);
        ellipse(350, 350, 300, 300);

        noLoop();
    }

    protected override void draw() {
    }

    protected override void onKeyTyped() {
        if(key == ESC) { loadScene("Menu"); }
    }
}
```

Unityはプログラミングしなくてもいろいろできて便利なのですが、いざプログラミングするときに、すぐ図形が出るようなものがあってもよいのではと思って作ってみました。

Unityの基本思想と異なる仕組みのものを無理やりいれているので、作った後、もうちょっとこうしておけば、ああしておけばというのはあるのですが.....まあ、とりあえず。

ソースは全部公開しているので、ProcessingユーザーがUnityの関数をざっと知る場合や、その逆でも参考になるかも？

Unity WebPlayerによるデモはdev.EYLN.comのサイトに置いてあります。

[Trackback\(0\)](#)

風呂の水面+浴槽にも投影する、風呂ジェクションマッピング2

ニコニコ動画に、風呂ジェクションマッピングの[新バージョン動画を投稿](#)しました。
興味のある方はぜひ。



[2014年4月25日 フジテレビ・スーパーニュースでバージョンアップ版をお披露目](#) したときのものと同じ

プログラムで、自宅のお風呂で撮影。夏場に撮影はきつかったものの、子供は大喜びでした。

BATHROOM SYMPHONY

そうそう、直接の関係はないのですが、[お風呂をもっと楽しく！プロジェクト - BATHROOM SYMPHONY](#)というお風呂関係のプロジェクションデモ（広告）があります。

これに「風呂ジェクション」というキーワードつながりで、制作中に連絡があり、ちょっとだけコメントしたりしました（ほぼなにもしてないのにスペシャルサンクスに名前を載せていただいてたり）。
歌声と浴室のプロジェクションマッピングが融合されていて、とても美しいです。
こちらも撮影は大変だったみたいですよー。

[Trackback\(0\)](#)

[Trackback\(0\)](#)

■ 360行でイチから作るRPG - Processingで簡単RPG開発

いやぁ、今年ももう12月ですか。早いなぁ。
そして12月といえば[Processing Advent Calender](#)ということで、
ProcessingでRPGを作ってみました。

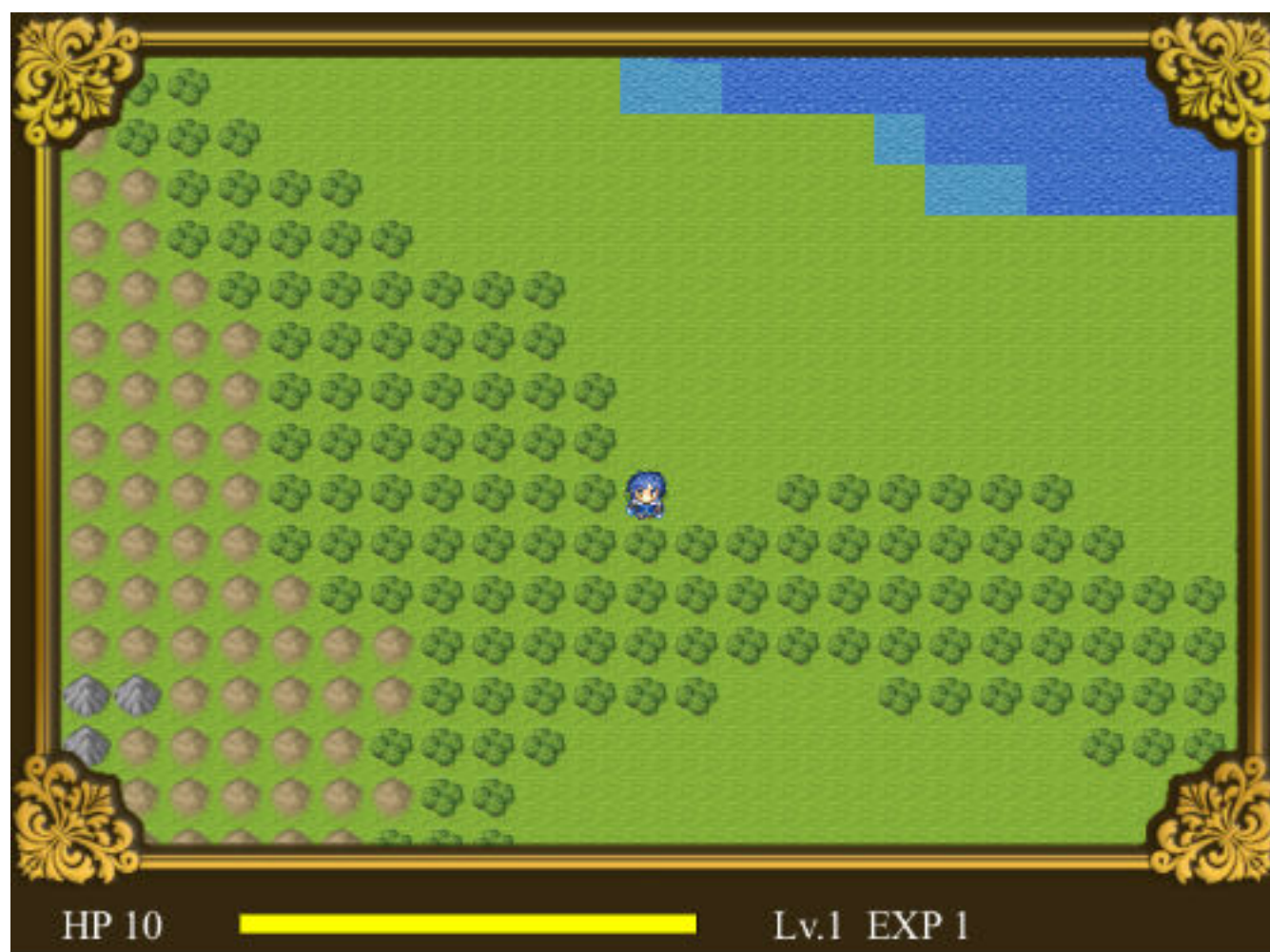
1ファイルのソースのみのチープなものですが、作っていると楽しいです。
[ソースコードや解説、ブラウザ上で実際に動くデモはこちらから](#)どうぞ

Tag : [Processing](#), [ゲーム](#), [開発](#)

[Trackback\(0\)](#)

■ 370行で作るRPG - 画像対応版

[先日](#)の1ソースファイルのRPG、画像を使うバージョンもさりげなく追加しました。



モンスターは10体登録してあります。





ソースコードや解説、ブラウザ上で実際に動くデモはこちらからどうぞ

Tag : [Processing](#), [ゲーム](#), [開発](#)

[Trackback\(0\)](#)

ツイート

19

B! Bookmark

29

m チェック


クリップ

G+1


0

@n_ryotaさんをフォロー

お知らせ



[【動画】モニター募集！1日2粒で年間トマト約800個分](#)
トマトに含まれる話題の健康成分「リコピン」が今なら初回購入50%OFF！(カゴメ)







[乾燥による小ジワを目立たなくする。「粧」のチカラ！ロート製薬の「粧肌くりーむ」](#)
輝く粧のように、透明感のある肌へ。うるおい実感！500円モニター募集中 (ロート製薬)

Sponsored

関連する記事

表示されるまで数分～最大で数日ほどかかります。しばらくお待ちください。

関連するみんなの記事

-  [Processing](#)
-  [Processing](#)
-  [Processing](#) 恐るべし
-  [\[K!\]【Arduin】Processingのリストとは？【Processing】](#)

Twitter(最新 5)



ちょすけ@相互フォロー100% @vip_tyosuke 07月05日
“Processingで遊ぼう - Writing Cafe” <http://t.co/whDMA36MAt> #processing #プログラミング



@robo8080 07月14日
ここの「Processingで作る、300行の3Dスペースシューティング」をProcessing.jsに変換してkonashi.jsで試したが、さすがに無理だった。 : Processingで遊ぼう - Writing Cafe <http://t.co/s21H6Fj57a>



@kaiyan 03月01日
@Cii_you 参考 <http://t.co/huUVbC2R>



@n_ryota 01月18日
2位は「Processingで遊ぼう」 <http://t.co/EQDCGC78>



@haihai_babuuu 11月28日
ここ面白い <http://t.co/V5oOdLOu>

はてなブックマーク(総数 0)

Facebook

POWERED BY

ツイート 19

B! Bookmark 29

m チェック

クリップ

@n_ryotaさんをフォロー

お知らせ

関連する記事

表示されるまで数分～最大で数日ほどかかります。しばらくお待ちください。

関連するみんなの記事

- [Processing](#)
- [Processing](#)
- [Processing 恐るべし](#)
- [\[K!\] 【Arduin】 Processingのリストとは？ 【Processing】](#)

Processing 関数 位置 座標 カメラ

Twitter(最新 5)



ちょすけ@相互フォロー100% @vip_tyosuke 07月05日
“Processingで遊ぼう - Writing Cafe” <http://t.co/whDMA36MAt> #processing #プログラミング



@robo8080 07月14日
ここの「Processingで作る、300行の3Dスペースシューティング」をProcessing.jsに変換してkonashi.jsで試したが、さすがに無理だった。 : Processingで遊ぼう - Writing Cafe <http://t.co/s21H6Fj57a>



@kaiyan 03月01日
@Cii_you 参考 <http://t.co/huUVbC2R>

@Cii_you 参考 <http://t.co/huUVbC2R> ↩ ↻ ☆



@n_ryota 01月18日
2位は「Processingで遊ぼう」 <http://t.co/EQDCGC78> ↩ ↻ ☆



@haihai_babuuu 11月28日
ここ面白い <http://t.co/V5oOdLOu> ↩ ↻ ☆

はてなブックマーク(総数 0)

Facebook

POWERED BY

ツイート 19

チェック

クリップ

G+1 0

@n_ryotaさんをフォロー

お知らせ



【動画】モニター募集！1日2粒で年間トマト約800個分
トマトに含まれる話題の健康成分「リコピン」が今なら初回購入50%OFF
！（カゴメ）



乾燥による小ジワを目立たなくする。「粧」のチカラ！ロート製薬の「粧肌くりーむ」
輝く粧のように、透明感のある肌へ。うるおい実感！500円モニター募集中 (ロート製薬)

Sponsored

関連する記事

表示されるまで数分～最大で数日ほどかかります。しばらくお待ちください。

関連するみんなの記事

- [Processing](#)
- [Processing](#)
- [Processing](#) 恐るべし
- [\[K!\] 【Arduin】 Processingのリストとは？ 【Processing】](#)

Processing 関数 位置 座標 カメラ

Twitter(最新 5)



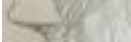
ちょすけ@相互フォロー100% @vip_tyosuke 07月05日
“Processingで遊ぼう - Writing Cafe” <http://t.co/whDMA36MAt> #processing #プログラミング ↩ ↻ ☆



@robo8080 07月14日
この「Processingで作る、300行の3Dスペースシューティング」をProcessing.jsに変換して
konashi.jsで試したが、さすがに無理だった。 ： Processingで遊ぼう - Writing Cafe
<http://t.co/s21H6Fj57a> ↩ ↻ ☆



@kajivan 03月01日
@Cii_you 参考 <http://t.co/huUVbC2R> ↩ ↻ ☆



@n_ryota 01月18日

2位は「Processingで遊ぼう」 <http://t.co/EQDCGC78> ↩ ↲ ☆



@haihai_babuuu 11月28日

ここ面白い <http://t.co/V5oOdLOu> ↩ ↲ ☆

はてなブックマーク(総数 0)

Facebook

いいね！

シェア

10人がいいね！しています。

POWERED BY [zenback](#)

[トップ](#) [差分](#) [一覧](#) [ソース](#) [検索](#) [ヘルプ](#) [RSS](#) [RSS10](#) [RSS20](#) [ATOM](#) [ログイン](#)

Copyright (C) 2002-2012 [n_ryota](#)

Powered by [FreeStyleWiki](#)