今年の秋号も超面白い記事が満載だよっ!!

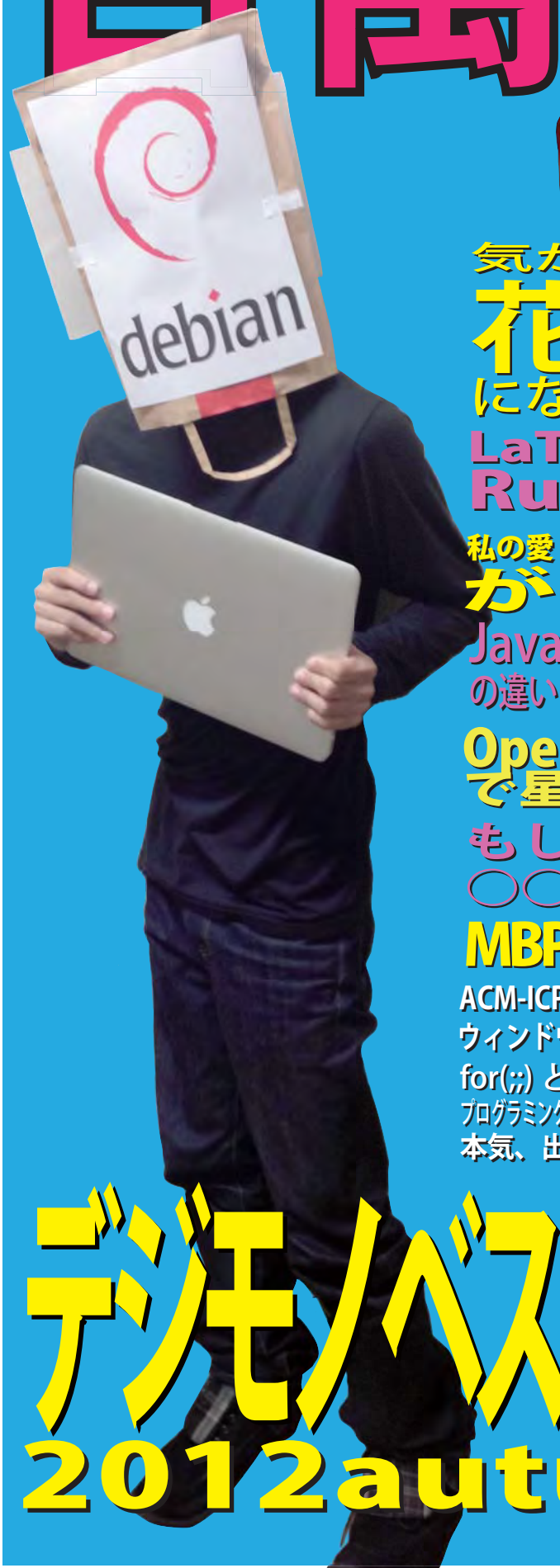
平成 24 年 11 月 23 日発行
（毎年春・秋発行） 通巻 38 号

# 季刊 百萬石

No.038

2012 秋号 0円

サーバー管理から花火の打ち上げまで、なんでもやっちゃう電気通信大学MMA公式の活動情報誌！




## 最強爆速IRCクライアント tenet

## 痛 グラス 痛 ガラス制作

気が付いたら花火師になっていた件

LaTeXにRubyを埋め込む

私の愛したらーめんまぜそば

がっつん

JavaScriptとJavaの違いのわかる人になろう！

OpenGL(OpenTK)で星を書いたよ

もし最近の森が○○になったら

MBP 環境構築記

ACM-ICPC 2012 国内予選の解説

ウィンドウマネージャを作ってみた

for(;;) と while(1) の熱き戦い

プログラミングを人に聞く前にすべきソリューション集

本気、出してみませんか

## デジモノベストバイ 2012autumn

# 部長挨拶

百萬石をお手に取って頂きありがとうございます。MMA(Microcomputer Making Association) は計算機にまつわる様々なことに取り組んでいるサークルです。コンピュータの好きな人たちが集まって各人好き勝手に活動しているという性格が強いサークルですが、人が集まっているからこそできる活動も多数行っており、この百萬石の発行もその 1 つです。今回も多くの記事が集まり、ボリュームのある冊子に仕上がりました。楽しんでいただければ幸いです。

2012 年 11 月
部長　clear

表紙の人: デビアンマン

デビアンマンは、Linux ディストリビューションの一つである「Debian」の普及のため、日々活動を行っている。フリーソフトウェア原理主義の第一人者として有名。今年の春には、都内某大学 (※私立ではない) にある得体の知れない謎サークル MMA の新歓のため来日。この調布祭にも友人のハスケルマンと共に駆けつけてくれた。

# 目次

# デジモノベストバイ 2012autumn

alstamber

**はじめに**

１年生がガチな Tech 系記事を書きまくっているので、老害であるところの私はちょっと Tech 系記事とは趣向を変えた記事を書いてみようと思った。そこで今年の百萬石では週刊ア○キーっぽく僕が今年購入したデジモノの中でもこれはおすすめというものをピックアップしてみようと思う。

## 1　私とデジモノの付き合い方

ベストバイを紹介する前に、まず私とデジモノの付き合い方について軽く紹介したいと思う。

まずここで言う「デジモノ」とはデジタル家電や PC、タブレット、スマートフォンや携帯電話、ゲーム機といった物を指す。私が今年買い集めたデジモノはこの中でも PC、タブレット、スマートフォンといったものに偏っているので、この記事でもそれらのデジモノが中心となることを予めご了承願いたい。

わたしのここ最近のデジモノとの付き合い方を端的に言えば、「とりあえず気になれば買う」というスタンスである。もちろんそんなことをなんの考えもなく続けていけば容易に破産に追い込まれてしまうのは目に見えている。そこで同時並行して「よほど気に入ったデジモノ以外は手放す」というスタンスもとっている。手放す方法はいろいろあるが、手軽なものとしては人との物々交換や中古ショップへの放出、オークションへの出品などがある。こうしたスタンスをとっているので、たとえデジモノを買っても購入した金額をそのまま未来永劫負担しているわけではない、というわけである。

中古ショップへの放出も以前に比べ手軽になったし、買取価格も最近は企業努力のかいあってか非常に良い値段を提示してくれる。オークションも中古ショップへ持って行くよりは面倒があるが、やはり手軽に他人と中古品をやり取りできる手段である。こうした手段がいくらでも充実してきているおかげで、デジモノをとりあえず買ったあと、暫く使用して吟味した上で厳選するという贅沢な付き合い方が可能になったといえる。

もちろん購入時価格を抑えることも考える必要がある。スマートフォンなどでは携帯電話会社各社が新規契約増加のための施策を打っていて、安価に端末を購入できるようなケースがしばしばある。こうした手段の活用が「とりあえず試す」私のやり方を支えてくれているわけである。

## 2　これが私のベストバイだ

### 2.1　MacBook Air

今年買ったデジモノの中でも文句なしに最高だったと断言できるのが、MacBook Air である。ごくごく最近に購入したのでそれだけ印象が強いこともあるだろうが、それを差し引いてもかなり良い買い物だったと断言して良いと思う。

#### 2.1.1　購入の経緯

以前から私は 2010 年春大学に入学した時に購入した MacBook Pro を使っていた。生活の中で毎日使う相棒のような存在であったのだが、いかんせん重さが 2kg ほどあって大学に持って行く際には若干苦労していた。また約 2 年半の使用によってキーボードはキーががたがたになり、裏についているゴム足は全部とれてしまい、HDD からは不穏な音がし始めた。また Core2 Duo マシンだったこともあり、日頃の作業でもたつきを感じることが増えていた。

折しも今年の春に購入した新しい iPad[*1]が使い道を失って放置状態になっていたため、iPad を売却した上で、iPad のような軽さを持ちかつ MacBook Pro のようにひと通りの作業をこなせるマシンを購入すれば幸せになるのではないかと考えた。その条件を丁度満たす存在が MacBook Air だったというわけである。

[*1] 新しかった iPad とか古くて新しい iPad とか新しくない iPad とか言うな

### 2.1.2　スペック

MacBook Air には 11inch モデルと 13inch モデルが存在するが、今回購入したのは 13inch モデルである。13inch モデルの中でも最も下位のスペックであり、俗にネットでは梅モデルと言われているものである。

- CPU……Core i5 1.8GHz Dual-Core(Ivy Bridge) Turbo-boost 時最大 2.8GHz
- Memory……DDR3 1600MHz 4GB
- Storage……SSD 128GB(東芝製)
- Graphics……Intel HD Graphics 4000
- Battery……公称 7 時間

### 2.1.3　レビュー

まず手にとって感じるのが、意味がわからないぐらい軽く、意味がわからないぐらい薄いということだろう。重さは約 1.3kg、厚さは最も厚いところでも 1.7cm である。片手で持てるぐらいに軽いし、本当に封筒に入ってしまうぐらいに薄い。鞄の中にも大きさとしてはちょっとたくさん書類の入っているクリアファイルぐらいの感覚で突っ込んでしまえる。この手軽さは私がラップトップに今まで求めてきたものそのものだった。今までのラップトップは厚さや重さといった面でそこまで手軽に鞄に突っ込めるものではなかった。しかし MacBook Air や Ultrabook の登場によってこういった課題が一挙に解決したと言ってしまっていいだろう。私は以前から Mac を使っていたので MacBook Air を選択したが、Mac を使っていないという人には各種 Ultrabook もそういった意味でよい選択になるだろうと思う。

薄くて軽いとなると、まともに使えるラップトップなのかという疑問が当然ながらに生じる。しかしそこは全く心配するに足りない。CPU は若干クロックが低めであるものの Ivy Bridge の Core i5 であるし、メモリも 4GB 搭載している。動画編集などヘビーな作業をするのであればこのスペックは不足だろうが、逆に言えばそうしたヘビーな作業をしない限りこのスペックで十分である。少々メモリが心もとないと思うのであれば、8,000 円ほど追加することでメモリを倍の 8GB にすることもできる。8GB もあれば通常用途には十分である。実際私はメモリを増やさず 4GB のままで使用しているが、多少スワップする程度でほとんど動作への影響はない。

ストレージが完全に SSD 化されているのもポイントである。これが MacBook Air の快適さを後ろから支えている。ストレージが SSD になったことでランダムアクセス性能が飛躍的に向上している。これが OS の起動やアプリの起動の高速化に寄与している。私が今使っている MacBook Air の場合、OS の起動には 15 秒ほどしかかからない。Evernote などのやや重いアプリもほとんど一瞬で起動する。さらに Chrome も以前より高速に動作するようになった。回線環境は変わっていないにもかかわらずである。キャッシュからの読み出し性能が SSD 化によって向上したためだろう。

図 1 は SSD の速度を実際に測定したものである。書き込み速度が若干遅い印象だが、読み込み速度は 400MB/s を超えており Intel 335Series などの測定結果にも迫ろうという結果を出している。


図 1　SSD 速度

バッテリーも公称 7 時間であるが、軽い作業をしている程度であれば本当に公称時間ほどに持つ。さすがにヘビーなネットワーク通信 (巨大なファイルのやり取り) などを行うと電池は減っていくが、そのようなことをあまり行わない私の場合電池に問

題を感じることはない。MacBook Pro の時は AC アダプタを外に持ち出すことが多かったが、MacBook Air にしてからは AC アダプタが必要になる頻度は大幅に減少した。AC アダプタそのものもそれほどかさばるものではないので、本当にヘビーな作業をする人も持ち出す上でそれほど苦にはならないと思われる。

発熱問題もほとんど存在しない。私があまり CPU に負荷をかける作業をしないことも影響しているだろうが、ファンが回転する音を購入以来まだ一度も聞いていない。ネットのレビューによれば HD 動画を 6 個同時再生したあたりからファンが漸く回り出すようなので、相当に負荷を与えないとファンが回転することは無さそうである。ファンが回転しないということは、発熱が大きくないということも意味している。Ivy Bridge 移行によって発熱が抑えられているのは確実だろう。パームレストが熱くなりにくいため、ストレスも溜まりにくい。

しいて後悔している部分を挙げれば、ストレージとメモリの容量だろう。4GB でも特段問題なく動いているが、今後新しい OS がリリースされるに従って 4GB では不足していくような気がしてならない。もちろんその時には古いものを売却して新しいものを買えば良い話なのだが……。ストレージも今のところ 128GB で足りてはいるが、今後音楽データや動画データなどを詰め込んでいった時にちょっと心もとないかなという気はしている。予算を優先して、このような部分をケチってしまったことについては後悔しているといえるだろう。

またディスプレイについても少々気になる点がある。MacBook Pro with Retina Display がリリースされて久しいが、やはりそれに比べるとかなり見劣りする。ラップトップの液晶としては綺麗な部類ではあるが、精細度ではもちろん Retina Display に勝てないし、発色も比較すると良くないように思う。Retina Display モデルは IPS 液晶なので比較するのも酷と言えるわけだが。

このような欠点を差し引いても、非常に満足度が高く、Windows や Linux 系 OS にこだわらないのであれば、軽量ラップトップとして MacBook Air は最高の製品だと私は思う。

## 2.2 iPhone5

### 2.2.1 購入の経緯

今年の 9 月に iPhone5 が発表されたが、当初は購入の予定はなかった。Softbank の回線は持っていたが、docomo のスマートフォンをメインに使っていたので必要性を感じなかったためである。しかしいざ発売されると LTE が想像以上に良いスピードを出しているという報告がネットに多数。Xi の速度低下に我慢できなくなっていたことと、もうすぐ以前から持っていた Softbank の回線が前回の端末購入から 2 年となり毎月の割引が消滅し割高になることを踏まえて、急遽 Apple Store に向かったのであった。

### 2.2.2 スペック

- ホワイト&シルバー
- 16GB
- Softbank 版
- 高さ 124mm 幅 59mm 厚さ 8mm
- 重量 112g
- 1,136 x 640 4 インチディスプレイ
- 連続待受時間 225 時間

### 2.2.3 レビュー

まず iPhone5 の最大の利点であり、飛び抜けた部分はやはり LTE に対応していることである。高々新しい通信方式に対応しただけと思うかもしれないが、この僅かな違いが大きな操作性の違いを生んでいるのである。快適な操作性に貢献しているのは LTE に移行したことによって ping の時間が大幅に短縮されたことである。LTE 網につながっている状態だと平均で 70ms ほどで ping が返ってくる。調子が良ければ 50ms を切ってくる。3G 網だと 100ms は確実にかかる。この差がそのまま Web ブラウジングの際の読み込みがスタートするまでの時間に影響している。いくら読み込みそのものが速くても読み込みが始まるまでに時間がかかっていれば結局快適なブラウジングには繋がらない。LTE の利点は通信速度で語られることが多いが、こうした伝送

遅延の少なさも大きな利点である。

もちろん速度も十二分である。調布駅北側入口付近で測定すると下り 30Mbps 程度/上り 10Mbps 程度出ることもザラである。自宅 (調布市内) でも下り 10Mbps/上り 5Mbps ほどが平均で出る。発売当初は都心でも繋がらない場所がそれなりにあったが、11 月上旬現在では山手線の駅でもほぼ入るようになり、都心でも LTE の電波を掴まないことが確実に減っている。このまま行けば近いうちに都内では実用的に使えるレベルになるだろうと思われる。

賛否両論の縦長になった画面であるが、私は概ね好意的に受け入れている。私はもともと iPhone が画面を大きくすることは Apple の戦略的にないだろうと思っていたので、今回の画面サイズ変更は非常に驚きであった。Android の大画面スマートフォンに慣れた私にとって、iPhone の画面は小さすぎるなという印象であったが、それが今回の変更によって改善されただろう。

順当な進化を遂げた製品と言えるが、気になる点も残ってはいる。話題にもなった地図アプリはその一例といえよう。位置がずれていることがとにかく話題となっているが、首都圏の観点から見ると鉄道の扱いが適当なのがかなり痛い。まず相当に拡大しないと鉄道の路線が地図上で確認できない。駅舎も地図上に現れないのでどこに駅があるのか非常にわかりにくい。首都圏は鉄道を中心とした街の構成となっているため、この仕様が地図の閲覧性を著しく低下させている。

また今回大きく変更されたデザインも気になる。私はホワイト＆シルバーを購入したが、裏面が図 2 のようにツートーンになっている。全面アルミのほうがかっこいいような気がするのは私だけだろうか。このデザインにはあまり納得ができていない。また側面のエッジに面取り加工がされているが、この面取り加工の強度があまり高くなく、使っているうちに欠けやすい加工になっている。こうした部分にもう少し気を遣ってほしいというところが正直なところである。

幾つかの不満は残るが LTE の速度と広くなった画面は使っているうちにじわじわと良さがわかる利点であると思う。この利点を薄く軽い筐体に詰め込んだという点で評価できる端末だろうと思う。


図 2　iPhone5 の裏面

## 2.3　GALAXY S III

### 2.3.1　購入の経緯

ハイスペックな docomo 向けスマートフォンが欲しいと感じていたところ、GALAXY S III が発売から 1ヶ月ほど経過して値段がこなれてきた。大画面なスマートフォンに対する興味が高まっていたこともあり、購入を決断。

### 2.3.2　スペック

- docomo 版 (SIM フリーでない)
- 高さ 137mm 幅 71mm 厚さ 9mm
- 重量 139g

- 1280x720 4.8 インチディスプレイ
- 連続待受時間 3G：約 400 時間／ LTE：約 270 時間

### 2.3.3 レビュー

GALAXY S III の魅力は画面にあるといえる。4.8 インチという大画面なディスプレイは大きすぎて持ちにくいという意見もあるだろうが、やはり大きな画面でブラウジングができるというのは利点であろうと思う。Twitter もやはり大画面のほうが快適である。Web ブラウジングなど電話としてではなくインターネットから情報を取り出す端末として割り切れるのであれば快適だと断言していいだろう。

ディスプレイは液晶ではなく有機 EL である。最初は有機 EL 特有のビビッドな発色に慣れないかもしれないが、この発色を許容出来るのであれば GALAXY S III のディスプレイは非常に高品質であるといえる。液晶でないため視野角も広く保たれており、精細度も高い。

電池持ちも悪くない。適度に Web をぶらついたり Twitter したりといった使い方であれば一日電池が切れることはない。もちろんテザリングを使えばもりもり電池は減る。4 時間持てば良い方ではないだろうか。

処理スペックそのものも十二分である。国際版とは異なりデュアルコアチップとなってしまったが、代わりにメモリが国際版の倍となっている。この判断は最終的には良い方向に転んだのではないかと私は思っている。クァッドコアを使いこなせる環境はまだまだ少ないが、メモリが潤沢にあることはバックグラウンドでたくさんプロセスが走っていても処理に影響しにくいということであり、パワーユースにじわじわながらも良い影響を与えていると思う。購入以来様々なアプリをこの端末の上で動作させているが、もっさりしていると感じたことは今のところない。私がゲームなどのアプリをあまり触らないということもあるだろうが、ちょっとヘビーな使い方であってもちゃんとついてきてくれるのがこの端末だと思う。

肝心の通信は少々微妙である。ドコモが提供する LTE サービスである Xi はユーザー数が今年に入って急増したことによって回線が慢性的に混雑している状態である。結果として LTE の本来の利点である低遅延・高速度がほとんどこの端末では生きてこない状態になっている。この端末の購入から 2ヶ月後に前述の iPhone5 を購入したわけだが、あまりの体感速度の違いに愕然としたほどである。整備が更に進めば改善が進むものと期待されるが、今後の課題といえよう。またこのあたりの問題についてはドコモがこの秋から 800MHz および 1.5GHz で LTE サービスを開始したため改善への期待は高まってきているとは言える。ただし GALAXY S III は 2.1GHz 帯にしか対応していないため、恩恵は Xi ユーザーが他の周波数帯に移動することによる間接的なものしか受けられない。

また UI にはもう少し工夫がほしいと感じられる。概して Xperia などに比べるとデザインの面で劣っており、国産のスマートフォンに比べても垢抜けない印象が拭えない。操作性そのものは悪くない。処理スペックが十二分であるため、カクつきなどもなく快適な操作が可能である。見た目はともかく使いにくいというほどひどいものではない。それでもしかしやはり見た目というものは重要ではないだろうか、と思う。かっこいいものを持っているというのは、やはりそれだけで所有欲を満たされるし、使っていて気分がいいものであるからである。

今年の冬に GALAXY S III α という後継機種が登場する。こちらは新しい周波数帯にも対応しており、GALAXY S III のよいところを引き継いでいるため GALAXY S III よりも更におすすめできる端末となりそうである。これから購入を検討するのであればぜひ SIII α をおすすめしたい。韓国メーカーであるサムスンということで抵抗があるという人も多いと思うが、そのような抵抗を捨ててぜひ一度触ってみて欲しい。なかなかに完成度の高い端末である。

# MBP 環境構築記

KHe7

**はじめに**

KHe7 が MacBook Pro 15” with Retina display を購入するに至った経緯と購入後の環境構築・設定調整に関する日記。

## 1　ことの始まり

この話は 10 月 21 日、応用情報技術者試験の夜に eclipse のアップデートと考えたことから始まる。

調布祭や合宿などで度々紹介しているため知っている方も多いだろうが、私が使用していた計算機は Dell の Inspiron 700m、出荷日 2005 年 6 月 14 日のご老体である。無謀だと知りつつも、TopCoder や AtCoder 用[*1]に Pleiades 4.2 Ultimate を放り込んで使っていたのだが、その夜、事件は起こった。

Eclipse のアップデート中に、それも外付け HDD へのアクセス中に計算機の応答が喪失。応答なしを通り越して反応なし。マウスカーソルすら動かない完璧な反応喪失。

仕方がなく電源ボタン長押しで強制終了して起動すると、外付け HDD のデータが破損している様子。コンピューターの管理のディスクの管理で Bad Disk と表示される始末。パーティションの認識から壊れているのでひとまず修復系のツールを用いて全セクタスキャンをすることに。進行具合から残り時間を計算すると 16 時間と出たので放置する。

## 2　続き

22 日。朝。起きると何故か作業が中断していて驚く。再び作業を開始して大学へ。帰って様子を見るがまだ途中。

23 日。朝。起きるとチェックは終了していたが何故かその先のステップに進めず。電源を落とし外付け HDD の電源を抜いて大学に。夕方。帰宅。駄目元で接続すると何故か正しく認識、なんとか動いてくれている……と思いきや壊れたファイルを踏むと Bad Disk の表示が再発。エラーが表示されるとかいうレベルではなくパーティションの認識ごと飛ぶ。これは…。

24 日。HDD は放置して課題をするなど。地雷を踏まなければ何とか繋がっていてくれると思っていたが自動的に走りだしたバックアップソフトが踏みに行って泣ける。そういえば放置していました。常駐設定を解除する。

25 日。何故か FORTRAN のプログラムを書く。高校時代の同級生[*2]に頼まれたとはいえ、それどころではないような…。

26 日。特にこの日は気にして活動した記憶はなし。計算機を使うような課題さえなかったか。

27 日。XCOPY が CRC をスルーしてくれると聞いたので弟の計算機につないでテスト。エラー吐いても無視して先に進むことを確認したので次の HDD を用意してなんとかすることに。そして、この事態の原因となった 700m には流石にご隠居願おうということで我が家の最高権力こと父を召喚。XPS 15 でも悪くないと思いつつも DELL は有名な “USB 地雷” [*3]を含めて USB 周りに色々問題抱えている&&持ち運びに不便なため MacBook Pro 15” with Retina display と主張して通す。異様な状態の HDD からデータを退避させるためにヨドバシカメラで 2TB のものを購入。不穏かもと思いつつ BUFFALO でファンを付けて。しかし…あんなファンで 2k[*4]ってちょとない。最初から付けろ。Windows と Mac OS X の両方から使うことを考えて exFAT でフォーマットする。そして、XCOPY で片っ端からデータを複製。30 個ほど CRC で飛んだがなんとでもなるファイル達だったので生存 。

28 日。水曜までと思っていた実験が実は月曜日までであったことが発覚。700m から JED に繋いで必死にプログラムを完成させる。耐久プログラミング。明日デモで明後日レポートとかちょっと無理。

29 日。バグの残るプログラムでデモ。途中でバグが出て残念な感じになるなど。帰宅後、強引に課題を仕上げたツケが回ってきて本気で寝ていた。

30 日。大学に行って帰ってきて。夕方からバグ修正とレポート生成と提出。忙しい。近日中にという感じになったレポートは本来の〆切の 2 時間後に提出。

31 日。レポート提出直後ですが次のテーマ。JED で sudo wireshark をする権限が降ってきて楽しい。変なのが見える…とか思うと先生が ssh で入ってきているなど。ふむふむ。と、帰宅後にメールを見ると MBP はようやく出荷されたとか。少し遅い。確かに 1〜3 営業日ですが…。1〜3 営業日と書かれたら 2 営業日くらいで出して欲しいとか思うなど。

11 月 1 日。どういう環境を準備するかについて考えを巡らせ始める。BootCamp するのは必然として 7 か 8 か。Windows 7 に素人が購入するとは考えにくい例の Visual

[*1] 何か作ろうとか何度も思いながら作ってない。

[*2] 横国に行ったあの人といいこの人といい人に丸投げしても学べないし身に付かないぞ？この先生きのこれるのか？というか、2 人共横国だったのですか……知らなかった。

[*3] USB メモリを USB ポートに挿したまま計算機の電源を入れると、BIOS が起動途中で止まるという BIOS の致命的なバグ。

[*4] 直径 40mm のファンだし安っぽいし 100 円してないんじゃないの…？

Studio 2012 Ultimate[*5]と Pleiades 4.2 Ultimate を放り込んで、LaTeX 環境も再構築…とか考えると案外忙しくなりそう。と、予め準備ということで様々を Dream Spark から 0 円購入で引っ張ってきておく。

11 月 2 日。部会後にポテト L、油そば学生大盛りとやって胃の許容量限界を体感。自分の状態くらい把握しないと、と思う。

## 3　到着、そして。

そして 11 月 3 日。午前 10 時 5 分前、ついに MacBook Pro 15” with Retina display 到着。

早速の開封イベント。Galaxy Nexus[*6]片手で進みます。

図 1　本体画像

構成は CPU が 2.6GHz、メモリ 16GB、SSD256GB、そして JIS キーボードです。他のオプションは選択せず。学割で 193641 円。流石にお高いが 7 年間がありますし、その間に何台か買ったと思えば。

我が家の他の計算機は AKIA と DELL だったので Apple 製品は初。箱の完成度とその小ささに感心するなど。そして、リファレンスの類がクイックスタートガイドのみという潔さに良くも悪くも感心する。

起動して触る。初 Apple 製品というだけあって少し勝手が分からないが、それらしく触るとそれらしく反応するので面白い。この辺の “感覚” に関しては計算機・スマートフォン好きという辺りが多分に影響していそう。設定をひと通り問題ないように変更していざ BootCamp。

導入する OS は Windows 7。先に少し書いた通り電気通信大学の情報・通信工学科が契約している DreamSpark から 0 円で “購入” したもの。BOOTCAMP 用のパーティションに 80GB ほどを割り当てて導入開始。

Bootcamp ツールが USB メモリに iso イメージを展開。Windows 用のドライバツールを放り込んで再起動。Windows 7 のインストール、ドライバの導入、Microsoft Update の反復、Visual Studio、Office 2010、Microsoft Update の反復を抜けて、Pleiades を解凍してようやく終了……というところで Mac 側からデータの共有用に 3 つ目のパーティションを切ったら BootCamp が見えなくなって。どうにも直せなかったのでやり直し。順番とかあったんですかね…。

容量を半々で切るようにして再び。同じ作業で非常にやる気が削がれているが仕方がなく。Windows 7、Office 2010、Visual Studio、Microsoft Update × n 回。Pleiades を解凍してデスクトップやマイドキュメント、スタートメニュー周辺のファイルを差し戻して終了。しかし、セットアップだけで 1 日が終わってしまった。

壊れたと思っていた HDD を繋いだところ、ディスクの修復がどうのこうのと言ってきたので実行させたらエラーは回復したとか。なんと…。余分な投資が…。とはいえ、論理的な状態が不穏なことは変わらないので旧から新への RichCopy[*7]を走らせて追加分をコピーした後、旧をフォーマット。新から旧へバックアップ。700GB 近いデータのコピー。時間かかるので就寝。

11 月 4 日。起きると、何故かコピーが終了していないにもかかわらずスタンバイになっていて驚くなど。どうなってる。dropbox 辺りが Windows の権限警告に引っかかって作業が止まり、そのまま時間が経ってしまった模様。なんというか微妙。再開したコピーを止めるわけにもいかず、なにか始めるにも微妙という膠着状態。積んでいたアニメでも消化するというなんというか非生産的な活動に移行。

11 月 5 日、月曜日。コピー作業とバックアップに関する設定が終わった。本体重量が僅か 2kg なので持ち運ぶことに。

## 4　使用開始、感想

まずは起動時間について。

Mac OS X での起動時間は 30 秒。読み込み中といった感じの表示がなくなるまでの実測でこの程度なので非常に快適。スリープ運用でも電池を食わないという。これ、瞬時復帰で素晴らしい。この辺りが流石の Apple。

一方、Windows 7 での起動時間は 45 秒。Mac が立ち上がって BootCamp の判定が走ってそれから Windows の起動作業に入るようで少し時間がかかる。十分に短いが瞬時とは言い

[*5] 結局何だったんでしょうねあのネタ。
[*6] iPhone5 とか iPad とか持ってません。
[*7] XCOPY コマンドの拡張である ROBOCOPY コマンドの GUI 版

がたい感じ。ただ、シャットダウンせずに運用を行えば……と考えていましたがスタンバイは不可能な模様。休止状態に入れて復帰するのとシャットダウンしてしまって起動するのでは起動時間に有意な差がないことを考えるとかなり微妙な感じ。とはいえ、電源ボタンを押してから 10 分くらい放置しなければ行けなかった先代の 700m と比べると隔世の感。

次、電池運用持続可能時間について。

公称 7 時間。Mac OS X で使っていると講義中に適当に資料を見ている程度、時々テキストを入力する程度であれば 1 時間に 10% しか減らない。Mac OS X の電源管理素晴らしい……流石 Apple。3 コマ + 昼休みに使っていて 3 割残るという充実の電池。Mac 側の環境整備が微妙で簡単な作業以外が出来ず、常用するに至らないのが残念であるものの整えばこっちを基本にしたい。

Windows 側。普通に使っていると 5 時間がいいところ。ある程度激しく使うと 3 時間半程度。時間 20% 以上は持っていかれるので安心できない。Windows の電源管理が残念なのか、上手く設定できていないのか、ドライバの詰めが甘いのか。大容量バッテリーであっても、解像度が高くサイズの大きい液晶に結構持っていかれる様子。明るさを控えめに設定することで持続時間は改善するものの全てを Apple が作った Mac OS X 側とは流石に違う。講義中にある程度 CPU を回して使うのであれば、電源を取ることを考えておかないとバッテリーが心配になる。ただ、1 時間が限界だった 700m [*8]からすれば進歩。1 コマ 2 コマは確実に持つので安心。

ソフトウェアのインストール以外で、特に気にしたのはキーボードの設定。

Mac 側では、control と command キーを入れ替えた。コピー、ペースト、ブラウザのタブ操作などで command キーを頻繁に利用するため、この入れ替えは必須。マカー[*9]があの command キーでどう使っているのかが良く分からない。JIS108A の Ctrl でさえ小指が無理するのに。そして、Windows 側の変態設定を Mac 側にも持ち込んだ。KeyRemap4MacBook で Use EISUU as KANA/EISUU (toggle) と KANA to Return を設定。両親指で入力切り替えるの性に合わなかった。これで Windows 側と設定を共通化できたので便利に。

Windows 側では、英数キーを半角全角に、かなキーを Enter に、右 command キーを App キーに変更した。Mac OS X 側に持ち込んだと書いた 700m の時代からの変態設定。小指を伸ばして Enter を打つのがかなり面倒なので。あと、マウスに手を伸ばして右クリックというのが 700m 時代から面倒に感じていたため、App キーは必須。BootCamp が Ctrl+1 を半角全角キーに割り当てているがこれが非常に迷惑。ブラウザのタブ切り替えでよく使われる Ctrl+1 を潰しているだけではなく、キーボードを離す順番に気をつけて置かなければ 1 が入力されてしまうという結構重大な問題に思う。設定から切り捨てることができるようになれば理想だが多分 Apple は対応してくれないだろう。全角半角キーがないことへの対応だしね……

また、ディスプレイの設定もある程度弄った。

Mac 側は QuickRes で 2880x1800 に変更。フォントを 18pt 前後に引き上げて。ただし、細かい調整が効かないので今後要検討。標準 (Best for Retina)[*10]では作業空間が狭すぎて勿体無い。むしろ Retina を無駄にしている。

Windows 側では 2880x1800 でテキストサイズの設定をデフォルトの 150% から 100% に変更。150% ではこの DPI の設定を作るときに考えていないアプリケーションでウィンドウサイズを 1920x1080 に設定すると画面一杯に変に引き伸ばされるので汚い。この辺は各アプリケーションの最適化が望まれるか。100% の問題点はフォントサイズの設定。かなり工夫しない限り小さすぎて読めないということ。Windows の画面設定で IPAmj 明朝の 16pt を基本にひと通り設定。Pleiades も IPAmj 明朝と IPA 明朝を基本にひと通り差し替えた。[*11]カラープロファイルを適当に放り込んだもののあまり変わらなかったので調整済みという事なのだろうか。このあたりはそのうち再検討。


図 2　執筆作業

とりあえず、左右に並べて百萬石の記事が執筆できるのが素晴らしい。1280x800 では隣にタブレットを出してきてレポートを執筆していた。あまり面倒でもなかったが、わざわざ接続

[*8] 7 年前の公称 2 時間 45 分が 7 年経っても実測 1 時間というのは十分持つというべきか？

[*9] この呼称は正しいのか？

[*10] 何が Best だ。

[*11] フォントとか良く分からないのでとりあえず IPA で適当に選んだ。ゴシックだと汚く見えたので明朝に変えた。語れる人はおすすめとかあれば。

する手間と電池が勿体無いような気がするという点で精神的に気疲れしなくてよいので楽。

微妙な設定も一応書いておく。

Windows 側の話だが、hiberfil.sys と pagefile.sys で 37GiB を占領していた。Windows 側に 120GB 切って Bootcamp していて、BOOTCAMP パーティションの空き容量が 30GB 程度になっていて何がこんなに…と首を傾げていたらこれである。

hiberfil.sys は MBP ではスタンバイとハイバネーションができないため、一時的に閉じるとなると休止状態となってしまうため外すことができないが、pagefile.sys の方についてはメモリを 16GB に変えているので日頃は不要と判断して最小割り当ての 800MB [*12]に設定。最大値は一応現在の設定の 16GB 程度にしておく。これで空き容量が 41GB になったので平和か。

Windows と Mac OS X からアクセスできる共通のデータ保存領域だが、安い SDXC を買ってきて使うという案に落ち着きそうだ。dropbox や Box などを利用する案もあるが実用性を考えて決めたい。ネットワーク環境がない場所での運用もありますし。

そして、いくつか適当にベンチマーク。


図 3　内蔵 SSD ベンチマーク

内蔵 SSD の I/O はシーケンシャルが 400MB/s 前後、512K が 300MB/s、4K が 13MB/s、4KQD32 で 255MB/s と非常に良好。試験方法に依るだろうし状況にも依るだろうが十分かと。しかし、書き込みの方が早かったりするのはなぜだ…どういう中身になっているのか非常に気になるけど気にしない事にする。

ベンチマークとは別になるが、eclipse が 10 秒で起きてくるのは感動的。

エクスペリエンスインデックスはほぼ満点という感じの成績か。グラフィック関連は MacBook Pro 15” with Retina display とはいえ流石にノートブックであるので 7.2。だが、十分高評価という異様な端末。いや、確かにお高い計算機なのでこれくらいはという感じもあるがやはり驚くべき性能。

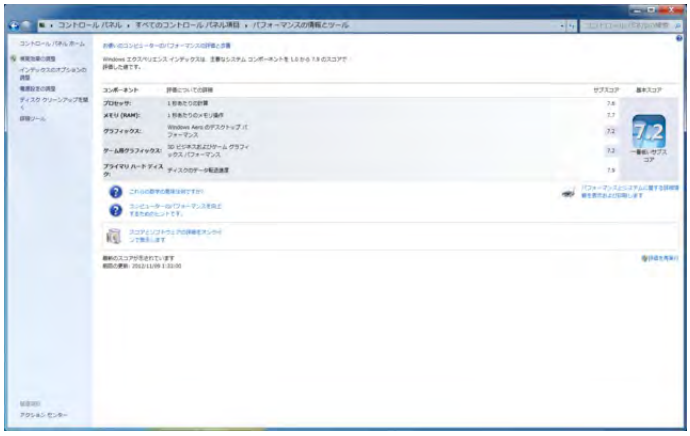


図 4　エクスペリエンスインデックス

## 5　終わりに

MBP 環境構築記と称しているものの、ここ 2 週間の私の日記から 700m と MBP に関する部分を抽出したものとなった。また、MBP の環境構築が事実上 Windows 7 の環境構築になっているのも少し勿体無い。

700m という 7 年間戦い続けてくれたご老体からの乗り換えになるのでどのようなスペックの計算機でさえ十分な環境改善になるが、この買い替えは満足という言葉では表せない何かがあったと素直に思う。

ただ、問題があるとするならば、Windows 環境からの離脱ができないがために、何かの作業をするとなると Windows 7 一択になってしまっているというのが非常に悲しい。

Mac OS X は見た目が大変素晴らしいので使いたいのは山々だが、如何せん今までの予備知識と資産のある Windows のほうが何をするにも考えなくていいというのは非常に微妙。Google で調べるだけとはいえそれさえ手間に感じるのは環境を移動するにあたって結構なハードル。とりあえず、Windows 側と Mac 側の両方に暮らしていける妥当な環境を整備するという方向で頑張ることにします。

手元の環境は十分強力になったので、適当に色々始めた辺りで次は MMA らしく独自ドメインと VPS 辺りに手を出してみようと思う。

[*12] 無効にしようとするとエラーの記憶用に 800MB は要ると Windows に警告されたため

# OpenGL(OpenTK) で星を描いたよ

alphakaz

### はじめに

さて MMA に所属しているにも関わらずプログラミングにも UNIX にもたいして通じず Firefox を入れてなお IE を愛用し,CL[*1]でも FP[*2]でもなんだかよくコマンド等が覚えきれないぞと嘆く筆者はまずこの記事に何を書くか迷った. 確かに筆者はそれなりに小さいころからパソコンを使用していた. だがそれは, プログラミングがどうとか計算機の仕組みがどうとか思案するためではなかった. 小学校低学年のころに家にあった Windows98(VAIO) では『3D あっ！とホームプランナー[*3]』(ご存知の方はいるだろうか) でひたすらビフォーアフターなクソ屋敷を建設していたことしか覚えていない. その後は父上から黒塗りのアコーディオンみたいなお下がりパソコンを与えられ, 小学校高学年から中学にかけてはもっぱら CFW がどうとかエロ同人誌がどうとか黒歴史のブログ小せ t……ゴホンゴホン. しばらくしたのち, その PC がブルースクリーンを吐き出したのでググってみると蟹のマークのマザボが Windows のバージョンアップで対応しないとかどうとかで中坊には結局解決も出来ずに大量のエロ画像が電脳の闇に葬られた. その後色々な意味で生 (性) 活に支障が出ると自作した PC は箱がお古であり, 装飾 LED は光らずスロットカバー紛失で代わりにガムテープがあちこちに貼ってある始末. サイズを調べもせずに買った『無限 2』というどでかい CPU クーラーの為にカバーに取り付けられていたファンすら外さなければならないという大惨事まで発生した. なんだかんだで今でもそのまま動いている.

だからなにが言いたいかというと別に筆者は計算機等について大して詳しくないニワカだという言い訳がしたいのである. 記事が面白くなかったり間違いだらけでも許してほしいのである. 最初はもう開き直ってセクサロイドに必要な性的人工知能について大真面目に考察するとか, いっそ自らの性癖を暴露しつつ短編電脳エロ小説を垂れ流そうとか思案したがそれでは流石に真面目にやっている方々に申し訳ない. そこで自分に問いかけた. じゃあ一体お前は何が書けるのか.

――さて, ここ調布にある電気通信大学[*4]に入学した際, 筆者は数ある有象無象のサークル群から 3 つのサークルを選んだ.MMA, 文芸・文学総合研究会, ロボメカ工房 VR 部隊である. そのうち現在, 所属しているロボメカ工房 VR 部隊に於いて, 筆者は 2012 年度調布祭関係のプログラミングをしている. おおざっぱに言ってしまえば Kinect を使ったエフェクト合成ゲームである. 一年生は C#を用いて OpenGL(OpenTK) でエフェクトを書く作業を任された. 記事にするならこれぐらいしかないと, そういう背景で今回のテーマがこうなった次第である. 以上.

## 1 環境

前述したとおりに今回は Kinect を動かすので C#を使用する. しかし OpenGL はそのままでは C#上で動作しない. そのため, ラッパークラスに OpenTK を用いて C#フォームアプリケーション上の GLControl で動かした. 開発は例の数万から数十万するともっぱらの噂の VisualStudio2010(Academic) で行った. 参照設定で, インストール済みの OpenTK,OpenTK.GLControl をぶち込んだ.

## 2 星の描画

Timer イベントハンドラとか GLControl(Load/Paint) とかそこら辺は全部設定し終わっていると仮定し, 早速星の描画について述べる.

### 2.1 目標設定

星の形は, 内部が塗りつぶされた五芒星として定義. これを正しく 2 次元平面上の中心に描画することを目標とした. ただし, 各頂点の座標を手計算でしこしこ求め, 座標を数値で直接指定するのは禁止とした. そして, 思いつく限り最短のコード数・単純なポリゴンで星を完全に描く. エレガントに行こう.

[*1] コンピュータリテラシー.
[*2] 基礎プログラミング演習.C.
[*3] 家具や窓もデザインを選んで配置できる便利な建築設計ソフトウェア. 何故か空飛ぶスーパーマンまでがオブジェクトに用意されていた謎設計. 販売は, エプソンに吸収され今は亡きエー・アイ・ソフトより.2007 年 6 月 30 日をもって製品サポートを終了している.
[*4] 国立大学法人

## 2.2 コード

とりあえず完成した star メソッドのコードを記述する.

```
private void star(double cx, double cy, double r)
{
    double x, y;
    for (int i = 0; i <= 4; i++)//三角形を, その角度を 2/5 π変えつつ 5 つ描画.
   {
       GL.PushMatrix();//現在の座標の位置を保存
       GL.Translate(cx, cy, 0.0);//原点座標をずらす
       GL.Begin(BeginMode.TriangleStrip);//描画開始
       {
           GL.Vertex2(0, 0);
           x = r * Math.Cos(2 * Math.PI * (0.05 + 0.2 * i));
           y = r * Math.Sin(2 * Math.PI * (0.05 + 0.2 * i));
           GL.Vertex2(x, y);
           x = r * Math.Cos(2 * Math.PI * (0.05 + 0.4 + 0.2 * i));
           y = r * Math.Sin(2 * Math.PI * (0.05 + 0.4 + 0.2 * i));
           GL.Vertex2(x, y);
       }
       GL.End();//描画終了
       GL.PopMatrix();//原点を保存された位置に戻す
   }
}
```

## 2.3 各メソッド解説

わざわざ言うまでもないかもしれないが, いくつかの OpenTK のメソッドについて述べる. あくまでもラッパークラス使用であり, 本家とは記法が異なるので注意.

### 2.3.1 GL.PushMatrix,GL.Translate,GL.PopMatrix

OpenGL は直交座標を用いているので,2 軸がどこにあるかはとても重要になってくる. たとえば描いたオブジェクトを動かすのには GL.Transrate(x 座標,y 座標,z 座標) を用いる. 要するに軸ごとオブジェクトを移動させてしまうのである[*5]. しかしオブジェクトをこのように動かした場合, 軸そのものが移動してしまうので, 他に描画したオブジェクトも移動後の座標に従って一緒に動かされてしまう. よって, オブジェクトごとに座標を動かして扱いやすくするため,GL.PushMatrix と GL.PopMatrix を用いる. 分かりやすく言うと, 前者は現在の座標をセーブし, 後者は格納された座標をロードする. 要するに, この二つで挟まれた内部の記述においていくら座標移動を行っても, 外部の他のオブジェクトに影響しないのである.

今回は他のオブジェクトと同時に描画するわけではないので関係ないが, このコードではメソッド star の引数に cx,cy として Transrate の描画座標を指定しているので, この二つの引数を変えてやれば, いくつかの星を別の場所に描画することも可能である.

### 2.3.2 GL.Begin,GL.Vertex2,GL.End

GL.Begin は描画の開始,End は描画の終了地点を表す. 内部にある GL.Vertex2 の引数が描画の指定座標として扱われる. BeginMode.TriangleStrip は, ひとつ前の指定座標と二つ前の指定座標, そして今回の指定座標を, 塗りつぶされた三角形として描画するためのモードである.(n,n-1,n-2) を座標とするので, 三角形の集合で表せる図形なら何でも描ける便利な野郎である.

[*5] ちなみに引数は double 型で, 座標は中心から端までを 1 とした比率を表す. 右上が (1.0,1.0), 左下が (-1.0,-1.0) といった具合である.

## 2.4 星

星の描き方にはいろいろある. しかしでこぼこの図形であるがゆえに, 単純な角形を書くより多少面倒である. 当初は, 全ての頂点の座標を計算して 10 個の 3 角形に分割しようとも考えた. しかし, もっと少ない三角形の集合で描画できないものだろうか……と考えたところ, 今回の案に行き着いた. この場合, 中心の 5 角形は 2 重になってしまうが, 描画する 3 角形は 5 枚で済むのである[*6]. そのうち最初の一枚だけ描画すると以下のようになる.

図 1　1 個めの三角形を描画した状態

これは隣り合わない外側の頂点二つと中心の 3 点を結んだ 3 角形である. これだけではわかりにくいので, 二つ目も描画してみよう.

図 2　2 個めの三角形を描画した状態

2 つ目の三角形は, 最初の三角形が 72 度――つまり,$2/5\pi$ だけ反時計回りに回転したものである. コード内では,i の係数部分が $2/5\pi(=2\times0.2\pi=0.4\pi)$ となっており,i が増加するたびに $2/5\pi$ 回転するようになっている. そしてこの回転を合計 4 回繰り返すと, 5 個の 3 角形が描画し終わり, 次のように完全な星として完成するのである.

図 3　5 つの三角形を描画した状態

## 2.5 評価

実質このソースコードは, 単純な描画に直接関係しない PushMatrix や Transrate を除けば 10 行で完結している. 星の定義を五芒星とするなら, 正確さも申し分ない. しかし, 最短かは確実ではない. 少なくとも, 内側と外側の 5 角形を用いた, 交互に頂点をなぞっていく 10 行の描画方法なども Google 検索により発見した[*7]. しかしそちらはあくまでも星の外側のラインをなぞっただけに過ぎず, 内側を塗りつぶしてはいない. 加えて, 中心から頂点までの比を正確に記述するのにはもう数行必要なため, そのソースコードよりはこちらの方が, 簡潔さと正確さの点では優れていると言えるだろう[*8].

[*6] 3 枚で済ます方法もあるが, 対称ではなく (コード的に) 綺麗じゃないので却下した.

[*7] Yahoo!知恵袋「opengl で星を作成」`http://detail.chiebukuro.yahoo.co.jp/qa/question\_detail/q1313047633`

[*8] 尤も, 形の愛らしさで言えば, カービィの乗るスターのような形をした向こうには完敗だったが.

### 2.6 終わり

…….

## 3 だからなんだよ

と言われそうである.——が, 何度も書くが, 筆者はプログラミング初心者である. プログラミングのプの半濁点をようやく書き始めた程度のひよっこである. しかし重要なのは, この星を描くコードは少なくとも, 筆者が何も参考とせずに, 課題を与えられたうえで自分だけで考えて書いたソースコードだということである.HelloWorld などは書け, とは言われても所詮答えが載っている. 筆者はクズなことに問題集の答えは躊躇せず見るタイプなので, 最初から最後まで何も見ずに自分のみで考えたコードが思い通りの形になったのは初めてで, とても嬉しい気持ちになった. 初心を知ったのである.

## 4 まとめ

これから大学生活をしていく上で, 行き詰ることも多々あるだろう. しかし全ての人には初心があった. どんなにエライ人間でも, みんな最初はひよっこだったのである. 宇宙がどれほどデカかろうとビッグバンの直後には睾丸程度の大きさしかない時があっただろう.

それを忘れたら, 行き詰った時に本当に何もできなくなってしまう. だから, 数年後でも数日後でも, 何か行き詰ったら, この冊子を開いて自分の出発点を, 初心を思い出すために, 今回の記事を書いたのである.

筆者以外の人間には甚だ時間つぶしのくだらない記事であったかもしれないが, 今回の記事が, あわよくば行き詰っている人の助けに, 導く星になればと, そう願っている.

## 5 イイハナシカナー

因みに筆者はこれと似たようなことを５年ほど言い続けている.

# JavaScript と Java の違いのわかる人になろう!

hirosun

**はじめに**

時折 JavaScript と Java を混同している人がいます。需要があるかどうかは別として「違いのわかる人になろう」という魂胆のもとにこの記事は生成されています。なお、筆者は Java を扱ったことのない人なので懇切丁寧さに欠けることはご了承ください。

## 1 そもそも JavaScript と Javaって何なの?

JavaScript と Java はともにプログラミング言語です。名前が似ていますが別の言語です。もう一度言います、別の言語です。どのくらい違うかというと ト◯ポとポ◯キーくらい違います。Java の別ヴァージョンが JavaScript だとか、その逆もないです。正直これだけわかって頂ければもう書くことはないのですがどう違うのかを示すことも必要でしょう。さて、両者共にオブジェクト指向のプログラミング言語であり、世界で広く使われています。JavaScript は主に Web クライアント側で動的なウェブサイトの構築を行う為に使用されていますが、サーバー側で使われることもあります。一方、Java は主にサーバー側で使われることが多いですが、実行環境が広く用意されているために携帯アプリや Web クライアントなど多岐に使用されています。まずそれぞれの言語の歴史を軽く振り返った後、その「違い」について解説していきます。[*1]

## 2 JavaScript と Java の歴史

### 2.1 Java の歴史

JavaScript と Java は正式な登場年は同じですがまずは Java の歴史から紐解いていくほうがいいでしょう。Java は 1990 年に米 Sun Microsystems(現在は米 Oracle 傘下) がプログラミング言語 C/C++ の代替となる言語を開発するプロジェクトを立ち上げ、開発が始まりました。なぜ Java という名称になったかは定かではないですが、開発メンバーの一部が出入りしていた珈琲ショップで命名されたとか、Java という言葉はアメリカでは coffee を示す一般名詞であるだとか珈琲に関係する憶測が多いようです。Java は 1995 年 5 月に正式に公表され、それまで HTML や画像の表示に留まっていた Web ブラウザで Web ページ内にアニメーション表示やマウスを用いたインタラクティブな操作を実現してみせました。また、米 Netscape Communications が自社のブラウザである Netscape Navigator に Java の実行環境 (Java Applet) を提供する予定であることを発表し、広く注目を集めることになりました。以降は実行環境を増やしていき、現在の形となりました。

### 2.2 JavaScript の歴史

続いて JavaScript の歴史。JavaScript は Java に少し遅れて 1995 年に登場しました。開発経緯は米 Netscape Communications が「Java Applet の搭載発表したけど、Java Applet ほど不雑さを持たない簡単なスクリプト言語が欲しい。出来れば Javaっぽいやつがいいなぁ。じゃあ後は Brendan 氏、よろしくね!!」と身勝手なことを並べ立てた上で Brendan Eich 氏に半ば Netscape から丸投げされる形で開発が始まりました。その開発は超短期間の内に行われ、先程も出てきました Netscape Navigator 2.0 で実装されました。開発当初は LiveScript という名称でしたが、先に発表された Java が初期マーケティングに成功していて名が通っていたことから、「そのまま肖 (あやか)ってしまえばきっと成功する」という若干セコい発想から Java の名を入れて JavaScript という名称にすることになりました。ようはパクったんです。はい。以降、米 Microsoft の Internet Explorer に搭載されたこと、Netscape が Ecma International という標準化団体に提出し、以降この団体に言語仕様策定が委ねられたことで ECMAScript として標準規格化されて普及が進んでいきます。JavaScript の開発は現在は Mozilla Foundation が行なっており ECMAScript とは互換関係にある形となります。ただし、ECMAScript の標準化作業では Google、Opera、Mozilla、Microsoft など各ブラウザ開発企業間での衝突があったり、企画変更後に新しいブラウザがリリースされても過去のヴァージョンのブラウザを使用し続けるユーザー[*2]または古い OS を使用していてそもそもアップデートが出来ないなどという人がいるなどの理由からブラウザ間で互換性を持たせることを強いられる大変な言語となりました。

[*1] 解説されているはずです。

[*2] 大抵はアップデートもせず「動かない」とか「表示が壊れる」とか「そもそも表示できない」などという苦情をいう人

# 3 それぞれの特徴

## 3.1 Java の特徴

Java は前述の通りその開発経緯から、C や C++ から構文を多く引き継ぎ、クラスベースオブジェクト指向プログラミングの考えに基いて設計されており、ネットワーク関連の機能やセキュリティ機構を備えたそれまでの言語の良いとこ取りをしたような言語となっています。そして、最も注目すべきはプラットフォームに依存せずに動作させることが可能であるという点にあります。これを可能とするのが Java 仮想マシン (Java Virtual Machine) であり、Java のコードをコンパイル時に中間コード化してその状態でソフトウェアを配布し、Java 仮想マシンによりプラットフォームに適した形式に変換してから実行させることで、ソフトウェアやハードウェアに依存しない実行環境の構築を実現しています。この環境を各種プラットフォームに移植することで様々な環境での Java プログラムの動作を可能としているのです。しかしながら実行時にコード変換作業を挟むという仕様上、完全な事前コンパイルを要する言語に比べて実行速度が落ちる傾向があります。ですが、現在では Java 仮想マシンは高速化が図られ、こうしたプラットフォームに依存したプログラムの実行速度と遜色ない程度の性能を実現しています。

## 3.2 JavaScript の特徴

先程の若干残念な開発経緯から当初からいろいろと不遇な思いをするはめになる JavaScript ですが、「Javaっぽい」という言葉ほど Java は参考にされてはおらず、その他 C や Scheme やらの影響を受けて開発されたプロトタイプベースオブジェクト指向プログラミング言語となっています。柔軟な言語設計はある人には簡易な言語であるというイメージを持たせ、またある人には抽象的で難解であるというイメージを持たせます。[*3]また、実行環境が手軽であるという点も特徴の 1 つです。基本的には Web ブラウザで実行されるため、ブラウザ 1 つで実行環境は整ってしまうわけです。しかしながら、JavaScript の処理系やブラウザのレンダリングエンジンはブラウザによって異なり、同じブラウザでもヴァージョンの違いもあるためにそれぞれでコードの動作や実行そのものが怪しいことがあります。このことが、JavaScript を難しいものにする原因と言えます。ECMAScript による標準化はあるものの、まだまだ環境の移植性には Java に比べると少々難がある状態です。実行方式については以前は実行時に逐次解釈していくインタプリタ方式が主流でしたが、最近では JIT(Just-In-Time) コンパイラを導入しています。表面的なインタプリタ動作をさせますが、内部でコンパイルしてからそのコードを実行するため、完全なインタプリタ方式より高速な処理を実現させています。

# 4 その差異とは

## 4.1 構文

Java と JavaScript は構文の上では似た部分を持ち合わせています。if 文や while 文、for 文など、キーワードと制御構造は似た部分があります。これは、Javaっぽさ (?) を持たされて JavaScript が開発されたためでしょう。以下は、コンソールで Hello! We are MMA!と表示するプログラムです。

ソースコード 1　JavaScript Hello.js

```
1 // Hello.js
2   console.log("Hello! We are MMA!");
```

ソースコード 2　Java Hello.java

```
1 // Hello.java
2 public class Hello {
3   public static void main(String[] args) {
4     System.out.println("Hello! We are MMA!");
5   }
6 }
```

でも、これだけ見ると見た目はそれほど類似性はないですね。

[*3] この柔軟さを害悪だとする人もいます

### 4.2 型

型については全くの正反対です。要は動的型と静的型です。Java は強い静的な型付けがなされており、変数には必ずデータ型に宣言を必要とします。そのため、変数に代入出来る値やオブジェクト参照には制約が伴います。一方 JavaScript は動的な型付けがなされる言語であり、変数に型がないために変数宣言にデータ型を指定してやる必要がありません。変数には任意の型の値の代入やオブジェクトの参照が行えます。JavaScript の柔軟さはここから来ていると言っても過言ではないですが、変数だけでなく関数の型やパラメータにも明確な型付けがないこの柔軟さが思わぬバグや想定外の動作の原因になりうることは少なくありません。それ故、データ型を確認しながらプログラミングを行なっていく必要があります。その他にも JavaScript には型変換やら変数スコープやらにそれぞれ癖があるために扱いには注意深さを伴う言語です。[*4]

ソースコード 3　Java での変数宣言

```
1 // int, char のようなデータ型の指定が必要。
2
3 int x = 0;
4 char y = 'Hoge';
```

JavaScript では変数の宣言時に var を付けるとローカル変数として宣言されたことになります。 また、var による宣言なしでも変数に値を代入することが可能です。その場合はグローバル変数として宣言されたことになります。関数内で var を用いずに宣言するともちろんグローバル変数として扱われます。

ソースコード 4　JavaScript での変数宣言

```
1 var x = 0;  // ローカル変数として宣言
2 var y = 'Hoge';  // クォーテーションマークで囲むことで文字列として代入ができます。
3
4 z = 0;  // ここで初めて'z'が現れた場合はグローバル変数として扱われます。
```

グローバル変数は汚染され易い、また偶然ライブラリと同名の変数名がグローバル変数として存在すると干渉を起こすなどの可能性があるため注意が必要です。

### 4.3 クラスベースとプロトタイプベース

Java はクラスベースオブジェクト指向言語でありのオブジェクトはユーザによるクラス定義とそれをもとにしたインスタンスという実体からなります。継承はクラスを通すことで行われ、クラスやインスタンスにはプロパティやメソッドの動的な追加はできません。JavaScript はプロトタイプベースオブジェクト指向言語であり、オブジェクトは静的なクラスはなく、単なるプロパティとメソッドの集合体であり、新しいオブジェクトは既存のオブジェクトのクローン (プロトタイプ) を利用して定義されます。継承はこのプロトタイプを通して行うことができ、またプロパティやメソッドの追加はどんなオブジェクトにも動的に行うことが可能です。ちょっとわかり難いですが、クラスベースは静的でオブジェクトはクラス定義より生成され、プロトタイプベースは動的でオブジェクトはオブジェクトから生成されます。

## 5 おわりに

さて、ここまで読んで頂いて JavaScript と Java の違いについて理解してもらえたと思います。いや、理解して頂きたい。成り立ちが異なれば、言語としての仕様も異なるのに名称は Java をパクって JavaScript という、この安直な名称のパクリがこれほどまでに世の中で誤解を生む結果となっていたのです。その誤解を解くことができたならばこの記事を書いた甲斐があるというものです。プログラミング言語としての差異はまだまだあるのですが必要 (?) 最小限に抑える形にしました。この記事では散々JavaScript をこき下ろしている感がありますが別段 JavaScript が残念な言語だということは決してありませんし、私自身 JavaScript が嫌いなわけでもなんでもありませんし、むしろ好きです。JavaScript と Java のどちらが優れているとかそういうことではなく、共通する領域[*5]で使用する場合は単純に好みの問題でしょう。さぁ、この記事を読んで JavaScript と Java の「違いのわかる人」になったアナタ、今後「違いのわからない人」に遭遇したら是非とも丁寧にその違いを説明してあげてください。

[*4] それはもうプログラミングしていく上での良いパーツと悪いパーツを解説した本が出てしまうほどに

[*5] 例えば Web クライアントサイド

# 最強爆速 IRC クライアント-telnet-

hiyakashi

**はじめに**

俺「telnetって何ですか？」iz「telnet 」師匠「telnet 」
web も見れる最強 IRC クライアントである telnet を用いて IRC の使い方を説明する。

## 1 IRC とは

IRC(Internet Relay Chat) はサーバ・クライアント方式のチャットシステム。利用には専用のクライアントを用いる。

## 2 接続する

まずは最寄りの IRC サーバを探す。サーバに接続するためにはターミナル上で以下のようにする。IRC では一般的に 6667 番が使われる。

```
telnet wasii.wawasi.club.uec.ac.jp 6667
```

接続に成功すると以下のようなメッセージが出てくる。

```
Connected to wasii.wawasi.club.uec.ac.jp.
```

## 3 登録する

接続に成功したらすぐにユーザ登録を行わなければいけない。ユーザ登録には USER メッセージを用いる。文法は以下の通り。

```
USER ⟨username⟩ ⟨hostname⟩ ⟨servername⟩ ⟨realname⟩
```

4 つパラメータが必要だがぶっちゃけどれも使わないので適当に空白区切りで 4 つ入れるといい。ユーザ登録が終わったら続いてニックネームを登録する。ニックネームの設定には NICK メッセージを用いる。

```
NICK ⟨nickname⟩
```

同一サーバ上の誰かが既に使用しているニックネームは使用できない。ここで設定したニックネームは後で変更可能である。USER と NICK の両方のメッセージの送信が成功して始めてログインできる。これを 20 秒以内にやらないとサーバから蹴られる。ログインが成功すると画面に夥しい量の出力が見えるだろう。これはリプライといって、メッセージが成功したり失敗したりしたときにサーバ側が返してくる情報である。クライアント側はこの情報を利用することができ、telnet の場合は人間が目視で情報を処理する。

## 4 チャンネルを選ぶ

IRC にはチャンネルと呼ばれる、チャットの部屋のようなものがある。チャンネルの一覧を取得するには LIST を用いればよい。

```
LIST
```

上記のようにすればサーバ内の全チャンネル（シークレット属性が与えられているものを除く）が取得できる。先頭が#や&で始まっているのがチャンネルである。入りたいチャンネルを見つけたら JOIN を使って部屋に入る。#mma に入りたいときは

```
JOIN #mma
```

とすればよい。チャンネルが存在しなかった場合は新しくチャンネルが作られる。チャンネルに入ると自動的に NAMES メッセージが送信され、チャンネル内にいるユーザのリストを得ることができる。名前の先頭に@（なると）が付いているユーザはチャンネルオペレータといい、チャンネルの管理を行うための特権を持っている。IRC は完全カースト制となっており、なると持ちに逆らうとチャンネルから蹴り出されたり、匿名掲示板に晒されたりする。

## 5 メッセージを送る

所謂普通のメッセージを相手に送りたいときは PRIVMSG メッセージを使う。PRIVMSG はチャンネルかユーザを対象に選択できる。チャンネルに n フラグが設定されていなければチャンネルに入ってない状態でもメッセージを送ることは可能である。

```
PRIVMSG hiyakashi,mznh ファミチキください
PRIVMSG #mma むくり
```

## 6 PING について

IRC を使っていると時折、次のようなメッセージが流れてくるかもしれない。

```
PING :wasii.wawasi.club.uec.ac.jp
```

これは PING と言って、サーバ側がクライアントの接続をテストしているものである。このメッセージを受けたら速やかに PONG メッセージを返さなければ、接続が終了される。

```
PONG
```

PING は定期的に送られてくるので、その都度送り返そう。

## 7 補足

- チャンネルから出るには PART、サーバからログアウトするときは QUIT を使う。
- 念でボイスチャットもできる。
- 分からないことがあったら RFC1459,2812,2813 あたりを読んでください。
- IRC クライアント及びサーバを作ろうなんて考えてる人いないだろうけど、悪い事は言わないからやめよう。
- 別の世界線の MMA でリレーしない IRC サーバを作る計画があったらしい。

## 8 Ruby で IRC サーバのような物体を書いたお話

偶然にもチャットシステムを開発することになった俺達は何故か RFC を読んで劣化 IRC モドキを作ることなっていたの巻。開発の流れとしては RFC を読みながら仕様通りに 1 つずつメッセージを実装するという非常に地味で、創造性の欠片もないような作業。特に MODE が夢に出てくるくらい苦痛だった。あと無駄に膨大な量のリプライが規定されていて、既存の IRC サーバと挙動を比較したりしながら作っていった。そんなわけで telnet にはとてもお世話になりました。サーバの実装には Ruby の Webrick とかいうライブラリを使ったんで、ネットワークプログラミング的なことはほとんどしなかった。けどマルチスレッドとかやったことなかったしとにかく色々勉強になりました。Ruby は AOJ でも使えるし (SF 限定) まじでいい言語なんで 1 年生は覚えましょう。

# LaTeX に Ruby を埋め込む

iz

**はじめに**
すこぶる書きにくい LaTeX ソースをどうにかして書きやすくしようというもの

## 1 いきさつ

LaTeX で化学実験のレポートを書いていたのだがどうにも書きにくい．イオン反応式ひとつ出力するのに `\[\rm{Zn}(固体)+\rm{Cu}^{2+}\longrightarrow \rm{Zn}^{2+}+\rm{Cu}(固体)\]` などと書かなければいけない．ちなみにこの LaTeX ソースの出力はこんなかんじ．

$$\mathrm{Zn(固体)} + \mathrm{Cu}^{2+} \longrightarrow \mathrm{Zn}^{2+} + \mathrm{Cu(固体)}$$

2,3 回打ちこんだころに嫌気がさし, プリプロセッサを自作しようかと思ったがすでに同じものがあるのに気がついた[*1]．ERB である．

## 2 ERB

### 2.1 ERB とは

ERB はドキュメント中に Ruby スクリプトを埋めこむためのプログラムである．html に埋めこむのを意識して設計してあるようだが,LaTeX でも十分つかえる．

### 2.2 インストール

Ruby インストールしたら入ってたような気がする．見つからなかったり日本語が通らなかったりする時は次の Ruby スクリプトを作成する．だいたい同じ動作をする．

myerb.rb

```
#!/usr/bin/env ruby
#coding: utf-8
require 'erb'
puts ERB.new(ARGF.read, nil, '-').result
```

### 2.3 ERB ソースの書き方

ERB は次のタグで囲まれた部分を Ruby スクリプトとして実行する．このタグで囲まれた部分は出力されない．

- "<%"と"%>"で囲まれた部分については実行するだけ．
- "<%="と"%>"で囲まれた部分は実行し, 返り値をそこに出力する．

"<%"を出力したいときは"<%%"と入力する．"%>"は対応する開始タグが存在しないときはそのまま出力される．
たぶん例を見たほうが早い．

[*1] 岩崎先生は自作したらしい．http://www.franz.com/services/conferences_seminars/jlugm00/conference/Talk11_iwasaki.pdf 参照

ERB の使用例

```
<%
  def double(x)
    x*2
  end
%>
\begin{document}
6 の 2 倍は<%=double(6)%>
\end{document}
```

3 から 7 行目で double メソッドを定義し,9 行目で使用している．これを ERB に渡すと次のように出力される (よけいな空行は抜いてある).

出力

```
\begin{document}
6 の 2 倍は 12
\end{document}
```

詳しくはこの辺を参照.

- http://www2a.biglobe.ne.jp/seki/ruby/erb.html
- http://jp.rubyist.net/magazine/?0017-BundledLibraries

# 3 例

”基礎科学実験 B ダニエル電池の起電力測定”のレポートを作成するために書いたメソッド群.

## 3.1 化学式

化学式を返すメソッド．数式環境内で使用する．引数に String を取り, 連続するアルファベット列をローマン体に, "->"を \longrightarrow にしたものを返す.

chem

```
def chem(s)
  s.gsub(/([a-zA-Z]+)/, "\\rm{\\1}").gsub("->", " \\longrightarrow ")
end
```

使用例

```
\[<%= chem("Zn -> Zn^{2+}+2e^-") %>\]
\[<%= chem("(-) Zn|ZnSO_{4aq}||CuSO_{4aq}|Cu (+)") %>\]
```

ERB の出力

```
\[\rm{Zn}  \longrightarrow  \rm{Zn}^{2+}+2\rm{e}^-)\]
\[(-) \rm{Zn}|\rm{ZnSO}_{4\rm{aq}}||\rm{CuSO}_{4\rm{aq}}|\rm{Cu} (+)\]
```

ERB の出力を LaTeX で処理したもの

$$\mathrm{Zn} \longrightarrow \mathrm{Zn^{2+}} + 2\mathrm{e^-}$$

$$(-)\mathrm{Zn|ZnSO_{4aq}||CuSO_{4aq}|Cu}(+)$$

### 3.2 分数, 対数, 下付き文字

数式環境内で使用する. chem メソッドと併用するために作った.

frac, ln, String#subs

```
def frac(num, den)
  "\\frac{#{num}}{#{den}}"
end
def ln(s)
  "\\ln{s}"
end
class String
  def subs(s) # subscript
    self << "_{#{s}}"
  end
end
```

使用例

```
\[<%= frac(5, 3) %>\]
\[<%= ln x %>\]
\[<%= ln frac("a".subs(chem "Zn^{2+}"), "a".subs(chem "Cu^{2+}")) %>\]
```

ERB の出力

```
\[\frac{5}{3}\]
\[\ln{x}\]
\[\ln{\frac{a_{\rm{Zn}^{2+}}}{a_{\rm{Cu}^{2+}}}}\]
```

ERB の出力を LaTeX で処理したもの

$$\frac{5}{3}$$

$$\ln x$$

$$\ln \frac{a_{\mathrm{Zn^{2+}}}}{a_{\mathrm{Cu^{2+}}}}$$

### 3.3 テーブル

LaTeX のテーブルを書くのがあまりに面倒だったので書いた. 本当にシンプルなケースしか対応していない. CSV データを文字列として読み込み, 各要素を ERB で処理した後 LaTeX のテーブルに変換する. オプションとしてハッシュを取り, table のオプションと caption,label を設定できる.

table

```
require 'csv'
require 'erb'
def table(csv, option)
  table_config = option[:table_config] || "htbp"
  caption = "\\caption{#{option[:caption]}}" if option[:caption]
  label = "\\label{#{option[:label]}}" if option[:label]
  data = CSV.parse(csv).map do |r|
    r.map do |c|
      ERB.new(c).result(binding)
    end
  end
  tabular_config = '|' + Array.new(data.size-1, 'r').unshift('l').join('|') + '|'
  contents = data.map {|r| r.join('&') << "\\\\ \n"}.join
<<TEXSOURCE
  \\begin{table} [#{table_config}]
    \\begin{center}
    #{caption}
    #{label}
    \\begin{tabular} {#{tabular_config}} \\hline
      #{contents}
    \\end{tabular}
    \\end{center}
  \\end{table}
TEXSOURCE
end
```

使用例

```
<%= table(File.read('table.csv'), {:captoin=>'濃度の組合せ'}) %>
```

ERB の出力

```
  \begin{table} [htbp]
    \begin{center}
    \caption{濃度の組合せ}

    \begin{tabular} {|l|r|r|r|r|r|} \hline
      組合せ  & $\rm{ZnSO}_4$&$\rm{CuSO}_4$\\ \hline
1& 0.010 mol/L         & 0.25 mol/L \\ \hline
2& 0.10 mol/L          & 0.25 mol/L \\ \hline
3& 0.50 mol/L          & 0.25 mol/L \\ \hline
4& 0.50 mol/L          & 0.050 mol/L\\ \hline
5& 0.50 mol/L          & 0.0050 mol/L \\ \hline

    \end{tabular}
```

```
    \end{center}
  \end{table}
```

上で使った table.csv の内容

```
組合せ  ," $<%= chem(""ZnSO_4"") %>$","$<%= chem(""CuSO_4"") %>$"
1, 0.010 mol/L         , 0.25 mol/L
2, 0.10 mol/L          , 0.25 mol/L
3, 0.50 mol/L          , 0.25 mol/L
4, 0.50 mol/L          , 0.050 mol/L
5, 0.50 mol/L          , 0.0050 mol/L
```

ERB の出力を LaTeX で処理したもの

表 1　濃度の組合せ

| 組合せ | $ZnSO_4$ | $CuSO_4$ |
|---|---:|---:|
| 1 | 0.010 mol/L | 0.25 mol/L |
| 2 | 0.10 mol/L | 0.25 mol/L |
| 3 | 0.50 mol/L | 0.25 mol/L |
| 4 | 0.50 mol/L | 0.050 mol/L |
| 5 | 0.50 mol/L | 0.0050 mol/L |

## 4　何が嬉しいか

別に化学式を書くのはそのためのパッケージ[*2]を使ってもできるしテーブルに関しても CSV を LaTeX のテーブル形式に変換するソフトは多分ある.

ERB を使用することの最大の利点はソースコードを埋め込めることにある. 別に計算した値をコピーしてこなくてもそれを計算するソースコードを埋め込んでしまえばいい.

## 5　おわりに

この記事で ERB に興味を持っていただければ幸いである. その気になればもっとシンプルに`\chem(...)` とか`\chem{...}`などと書けるようなプリプロセッサを作れると思う.

[*2] mhchem

# ACM-ICPC 2012 国内予選の解説

k_operafan, iz

**はじめに**

今年度の ACM-ICPC 国内予選の問題について簡単な解説と C++(一部 Java) のソースコードを書いてみた．問題文は http://www.cs.titech.ac.jp/icpc2012/icpc2012-mondai/icpc2012/common/guest_top_en.html から見ることが出来る．また，書いたコードが正しいかは AIZU ONLINE JUDGE http://judge.u-aizu.ac.jp/onlinejudge/ で検証することが出来る．

## A 問題

問題で与えられる歴において y 年 m 月 d 日の次の日を求めるプログラムを作ることを考える．y 年 m 月において，その月が 20 日まである (大の月である) 条件は，y が 3 で割り切れるか m が奇数であることとなる．d がその月の最後の日ではない場合は，y 年 m 月 (d+1) 日が次の日となる．d がその月の最後の日の場合は，さらに m について場合わけが発生して，m が 10 より小さい時は y 年 (m+1) 月 1 日が次の日となり，m が 10 の場合は (y+1) 年 1 月 1 日が次の日となる．

次の日を求めることができたら，後は問題文で与えられる日付に対して，何回日付を進めれば 1000 年 1 月 1 日になるかを調べることで解を出すことが出来る．

```
#include <stdio.h>

int n; // データーセット数
int main(){
  int i;
  scanf("%d", &n);
  for(i = 0; i < n; i++){
    int y, m, d; // 入力で与えられる日付（y 年 m 月 d 日）
    int answer = 0; // ミレニアム記念日までの日数
    scanf("%d %d %d", &y, &m, &d);
    while(y != 1000){ // ミレニアム記念日に到達するまで次の日に進める．
      int month; // y 年 m 月は何日あるか
      if(y % 3 == 0 || m % 2 == 1){
        month = 20;
      }else{
        month = 19;
      }
      if(d < month){
        d = d + 1;
      }else if(m < 10){
        d = 1;
        m = m + 1;
      }else{
        d = 1;
        m = 1;
        y = y + 1;
      }
      answer++;
    }
    printf("%d\n", answer);
  }
  return 0;
}
```

## B 問題

ある整数の数字を並び変えて最も大きい整数を作るには数字を大きい順に並び変えるとよい．最も小さい整数についても同様である．

数字を大きい順に並び変えるのは，0〜9 の各数字が何度表われたかを数えることで，簡単に行うことが出来る．

後は $a_0$ から順番に数列 a の内容を計算していく．問題文より $a_{20}$ まで計算すれば十分なので，条件を満たす i, j を全部調べて探すことで答えが出る．

```c
#include <stdio.h>

int a[21], L;

// a_i から a_(i+1)を計算する
int calc_next(int a, int L){
  int i, j;
  int h[10] ={}; // ヒストグラム
  int maximum = 0, minimum = 0; // 最大,最小
  for(i = 0; i < L; i++){
    h[a % 10]++;
    a = a / 10;
  }
  // 大きい順に使う
  for(i = 9; i >= 0; i--){
    for(j = 0; j < h[i]; j++){
      maximum = maximum * 10 + i;
    }
  }
  // 小さい順に使う
  for(i = 0; i <= 9; i++){
    for(j = 0; j < h[i]; j++){
      minimum = minimum * 10 + i;
    }
  }

  return maximum - minimum;
}

// a[i] = a[j]なるjがあるかどうかの確認
int check(int i){
  int j;
  for(j = 0; j < i; j++){
    if(a[i] == a[j]){
      printf("%d %d %d\n",j, a[i], i - j);
      return 1;
    }
  }
  return 0;
}

int main(){
  while(1){
    int i;
    scanf("%d %d", &a[0], &L);
    if(a[0] == 0 && L == 0)break; // データーセットの終了
    i = 0;
    while(1){
      a[i + 1] = calc_next(a[i], L);
      i++;
      if(check(i)) break;
    }
  }
  return 0;
}
```

## C 問題

サイコロの面の配置は，サイコロの正面に書かれている数字と上部に書かれているの数字により一意に定まる．サイコロはライブラリ化しておくか[*1]、もしくは、ペア (正面の数字, 上部の数字) に対して，右側の数字が何になるかを実際にサイコロを転がしてみて全て求めておくというのが一つの方法である．

後は問題文にかかれている (1)-(3) をプログラムし，シミュレートをする．各マスについて一番上にかかれている数字の情報と積み上げられているサイコロの個数を持てばよい．

[*1] http://www.prefield.com/algorithm/misc/dice.html が有名

```
#include <stdio.h>
#include <string.h>

int dx[4]={-1,1,0,0};
int dy[4]={0,0,1,-1};

enum{TOP,BOTTOM,LEFT,RIGHT,FRONT,BACK};
int dice[6] = {1,6,2,5,3,4};
void l_rotate(int *a, int *b, int *c, int *d){
  int t = *a;
  *a = *b; *b = *c; *c = *d; *d = t;
}

// d=方向 にサイコロを回転させる
void rotate(int d){
  if(d == 0){
    l_rotate(&dice[LEFT], &dice[TOP], &dice[RIGHT], &dice[BOTTOM]);
  }else if(d == 1){
    rotate(0); rotate(0); rotate(0);
  }else if(d == 2){
    l_rotate(&dice[FRONT], &dice[TOP], &dice[BACK], &dice[BOTTOM]);
  }else if(d == 3){
    rotate(2); rotate(2); rotate(2);
  }else{
    l_rotate(&dice[FRONT], &dice[RIGHT], &dice[BACK], &dice[LEFT]);
  }
}

int t[200][200]; // 上部の数字
int h[200][200]; // 各マスにつまれているサイコロの数
// サイコロを落とす
void solve(int top, int front){
  int x = 100, y = 100, i, j;

  // 同じ回転状態のサイコロを探す.
  for(i = 0; i < 6; i++){
    for(j = 0; j < 4; j++){
      rotate(-1);
      if(dice[TOP] == top && dice[FRONT] == front){
        goto exit_loop; // 発見したので 2重ループから抜ける
      }
    }
    if(i & 1){
      rotate(0);
    }else{
      rotate(2);
    }
  }
exit_loop:
  h[x][y]++;
  while(1){
    for(i = 6; i >= 3; i--) { // 転がす面
      if(i == 3){
        t[x][y] = dice[TOP];
        return;
      }
      for(j = 0; j < 4; j++){
        if(dice[2 + j] == i && h[x + dx[j]][y + dy[j]] < h[x][y] - 1){
          h[x][y]--;
          rotate(j);
          x += dx[j]; y += dy[j];
          h[x][y]++;
          i = 2; break;
        }
      }
    }
  }
}


int main() {
```

```
  int n, i, j;
  while(scanf("%d", &n) != EOF){
    int count[7] = {};
    if(n == 0) break;

    memset(h, 0, sizeof(h));
    memset(t, 0, sizeof(t));
    for(i = 0; i < n; i++){
      int top, front;
      scanf("%d %d", &top, &front);
      solve(top, front);
    }
    for(i = 0; i < 200; i++){
      for(j = 0; j < 200; j++){
        count[t[i][j]]++;
      }
    }
    for(i = 1; i <= 6; i++){
      printf(i == 1 ? "%d" : " %d", count[i]);
    }
    puts("");
  }
  return 0;
}
```

## D 問題

同じ鉄道会社を利用する場合，問題の条件より，出来るだけ長い区間を連続して使った方が特である．そのため，各鉄道会社毎に全点間最短経路をワーシャルフロイド法[*2]を用いて計算する．そして，鉄道会社毎に距離と運賃の関係を求め，同じ鉄道会社を用いて移動する時の，全点間の最安料金を計算しておく．

その後乗り換えを考慮するために，全ての鉄道会社について一つにまとめたグラフを構築する．これは，全点間について料金が最も安い鉄道会社を用いることにすれば良い．出来たグラフについてワーシャルフロイド法を用いることで最安料金を計算出来る．

```
#include <algorithm>
#include <vector>
#include <iostream>
using namespace std;

typedef long long int lli;

#define REP(i,x) for(int i=0;i<(int)(x);i++)

typedef vector<lli> Array;
typedef vector<Array> Matrix;
const lli INF=1LL<<50;

void warshall_floyd(Matrix& G){
    int V=G.size();
    REP(k,V)REP(i,V)REP(j,V)
        G[i][j]=min(G[i][j],G[i][k]+G[k][j]);
}
int n,m,c,s,g;
int p[20];
vector<lli> q[20],r[20];
Matrix G[20]; // 鉄道会社毎の最短経路を記録する
bool input(){
    cin>>n>>m>>c>>s>>g;
    if(c==0)return false;
    g--;s--;
    REP(i,c){
        G[i]=Matrix(n,vector<lli>(n,INF));
        REP(j,n)G[i][j][j]=0;
```

[*2] 最短経路を $O(V^3)$ で求めるアルゴリズム．実装がとても楽．

```
        r[i].clear();
        q[i].clear();
    }
    REP(i,m){
        int x,y,d,c;
        cin>>x>>y>>d>>c;
        x--;y--;c--;
        G[c][x][y]=G[c][y][x]=min<lli>(G[c][x][y],(lli)d);
    }
    REP(i,c){
        warshall_floyd(G[i]);
    }
    REP(i,c){
        cin>>p[i];
    }
    REP(i,c){
        q[i].push_back(0);
        REP(j,p[i]-1){
            lli t;
            cin>>t;
            q[i].push_back(t);
        }
        q[i].push_back(INF);
        REP(j,p[i]){
            lli t;
            cin>>t;
            r[i].push_back(t);
        }
    }
    return true;
}
lli calc_cost(int c,lli dist){
    lli ret=0;
    REP(i,p[c]){
        if(q[c][i+1]<=dist){
            ret+=(q[c][i+1]-q[c][i])*r[c][i];
        }else{
            ret+=(dist-q[c][i])*r[c][i];
            break;
        }
    }
    return ret;
}
int main(){
    while(input()){
        REP(i,c){
            REP(j,n)REP(k,n){
                G[i][j][k]=calc_cost(i,G[i][j][k]);
            }
        }
        Matrix M(n,vector<lli>(n,INF));
        REP(j,n)REP(k,n){
            REP(i,c){
                M[j][k]=min(M[j][k],G[i][j][k]);
            }
        }
        warshall_floyd(M);
        if(M[s][g]==INF){
            M[s][g]=-1;
        }
        cout<<M[s][g]<<endl;
    }
}
```

## E 問題

全ての円の交点と始点と終点をグラフの頂点とし，円の外を通らずに 2 点間移動出来るのならばグラフの辺を追加していく．後はワーシャルフロイド法を用いて最短経路を解くと良い．

```
import java.io.*;
import java.util.*;
import java.awt.geom.*;

class Circle {
    public Point2D.Double o;
    public double r;
    Circle() {
        this.o = new Point2D.Double(0, 0);
        this.r = 0;
    }
    Circle(Point2D.Double o, double r) {
        this.o = o;
        this.r = r;
    }
    Point2D.Double[] intersectionPoint(Circle c) {
        Point2D.Double[] ip = new Point2D.Double[2];
        double l = o.distance(c.o);
        double a = Math.atan2(c.o.y - o.y, c.o.x - o.x);
        double t = Math.acos((r*r - c.r*c.r + l*l) / (2 * l * r));
        ip[0] = new Point2D.Double(r*Math.cos(a+t) + o.x, r*Math.sin(a+t) + o.y);
        ip[1] = new Point2D.Double(r*Math.cos(a-t) + o.x, r*Math.sin(a-t) + o.y);
        return ip;
    }
}

public class Main {
    int n;
    Circle[] cs;
    Point2D.Double[] p;
    Scanner sc;

    Main() {
        sc = new Scanner(System.in);
    }

    boolean init() {
        n = sc.nextInt();
        if (n == 0) return false;
        cs = new Circle[n];
        for (int i = 0; i < n; i++) {
            double x = sc.nextDouble();
            double y = sc.nextDouble();
            double r = sc.nextDouble();
            cs[i] = new Circle(new Point2D.Double(x, y), r);
        }

        p = new Point2D.Double[2*n];
        p[0] = cs[0].o; p[2*n-1] = cs[n-1].o;
        for (int i = 0; i < n-1; i++) {
            Point2D.Double[] _p = cs[i].intersectionPoint(cs[i+1]);
            p[i*2+1] = _p[0];
            p[i*2+2] = _p[1];
        }

        return true;
    }

    void run() {
        while (init()) {
            double[][] graph = new double[2*n][2*n];
            for (int i = 0; i < 2*n; i++)
                for (int j = 0; j < 2*n; j++)
                    graph[i][j] = 1e18;

            for (int i = 0; i < 2*n; i++) {
                for (int j = 0; j < 2*n; j++) {
                    Line2D.Double line = new Line2D.Double(p[i], p[j]);
                    if (0 <= j - i && j - i <= 2) {
                        graph[i][j] = p[i].distance(p[j]);
                        graph[j][i] = graph[i][j];
```

```
                } else {
                    for (int k = (i+1)/2+1; k < (j+1)/2; k++) {
                        if (!line.intersectsLine(new Line2D.Double(p[k*2-1], p[k*2]))) {
                            graph[i][j] = 1e18;
                            break;
                        }
                        graph[i][j] = p[i].distance(p[j]);
                        graph[j][i] = graph[i][j];
                    }
                }
            }
        }

        for (int k = 0; k < 2*n; k++)
            for (int i = 0; i < 2*n; i++)
                for (int j = 0; j < 2*n; j++)
                    graph[i][j] = Math.min(graph[i][j], graph[i][k]+graph[k][j]);

        System.out.println(graph[0][2*n-1]);
    }
}

public static void main(String[] args) {
    new Main().run();
}
}
```

凸包を取るような感じで貪欲に解くことも出来る.

```
#include <cstdio>
#include <cmath>
using namespace std;

#define REP(i, x) for(int i = 0; i < x; i++)

int n;
double cx[100], cy[100], cr[100]; // 入力であたえられる円
double rx[200], ry[200], lx[200], ly[200];

// 2円の交点を求める。結果は副作用で (rx1,ry1), (rx2,ry2)に代入される。
void intersectsCC(double x1, double y1, double r1, double x2, double y2, double r2,
                  double& rx1, double &ry1, double &rx2, double &ry2){
    double L = hypot(x1 - x2, y1 - y2);
    double s = atan2(y2 - y1, x2 - x1);
    double aa = acos((L * L + r1 * r1 - r2 * r2) / (2 * L * r1));
    rx1 = x1 + r1 * cos(s + aa); ry1 = y1 + r1 * sin(s + aa);
    rx2 = x1 + r1 * cos(s - aa); ry2 = y1 + r1 * sin(s - aa);
}

// P1-P2-P3 が反時計回りの関係になっているかを判定する
bool ccw(double x1, double y1, double x2, double y2, double x3, double y3){
    return (x2 - x1) * (y3 - y1) - (x3 - x1) * (y2 - y1) > 0;
}

bool input(){
    scanf("%d", &n);
    if(n == 0) return false;
    REP(i, n){
        int x,y,r;
        scanf("%d%d%d", &x, &y, &r);
        cx[i] = x; cy[i] = y; cr[i] = r;
    }
    return true;
}

// greedy 解。おそらく平均 O(N)
void solve(){
    rx[0] = lx[0] = cx[0];
    ry[0] = ly[0] = cy[0];
    rx[n] = lx[n] = cx[n - 1];
    ry[n] = ly[n] = cy[n - 1];
    REP(i, n - 1){ // 交点を全て求める
```

```
        intersectsCC(cx[i], cy[i], cr[i], cx[i + 1], cy[i + 1], cr[i + 1],
                lx[i + 1], ly[i + 1], rx[i + 1], ry[i + 1]);
    }
    double now_x = lx[0], now_y = ry[0] ,res = 0, next_x, next_y;
    int l, r;
    l = r = 1;
    for(int i = 2; i <= n; i++){
        int next = -1;
        if(!ccw(now_x, now_y, lx[l], ly[l], lx[i], ly[i])){
            l = i;
            if(ccw(now_x, now_y, lx[l], ly[l], rx[r], ry[r])){
                next_x = rx[r]; next_y = ry[r];
                next = r;
            }
        }
        if(ccw(now_x , now_y, rx[r], ry[r], rx[i], ry[i])){
            r = i;
            if(next == -1 && !ccw(now_x, now_y, rx[r], ry[r], lx[l], ly[l])){
                next_x = lx[l]; next_y = ly[l];
                next = l;
            }
        }
        if(next >= 0){
            res += hypot(now_x - next_x, now_y - next_y);
            now_x = next_x; now_y = next_y;
            l = r = i = next + 1;
        }
    }
    printf("%.9f\n", res + hypot(lx[n] - now_x, ly[n] - now_y));
}

int main() {
    while(input())
        solve();
}
```

## F 問題

カードを昇順に並べた時に隣合う二枚の関係を「同じ」「差が 1」「差が 2 以上」の 3 つに分類する．この関係が全部同じカードはマッチの結果も同じになる．例えば,「1 1 2 3 5」と「2 2 3 4 10」はどちらも「同じ 差が 1 差が 1 差が 2 以上」となるので同じものとみなす．

隣り合う 2 枚の関係を全て生成してみて，問題で与えられるパターンとマッチするかを調べる．マッチするならば，隣り合うカードの関係から何通りのカードのパターンが作れるかを DP によって調べる．全てのカードのパターンの選ばれる確率は一様なので，これによって解を求めることが出来る．

```
#include <iostream>
#include <iomanip>
#include <string>
#include <map>
#include <vector>
#include <queue>
#include <cstring>
#include <cstdio>
#include <complex>
#include <algorithm>
using namespace std;

#define REP(i,x)for(int i=0;i<(int)x;i++)
int N,M,L;
struct Pat{
    int val,plus;
};
bool operator<(const Pat &a,const Pat &b){
    if(a.val!=b.val){
        return a.val<b.val;
    }
```

```
22      return a.plus<b.plus;
23  }
24  Pat pat[7];
25
26  bool input(){
27      cin>>N>>M>>L;
28      if(N==0)return false;
29
30      REP(i,L){
31          string in;
32          cin>>in;
33          Pat p;
34          p.val=0;p.plus=0;
35          if(in=="*"){
36              p.val=-1;
37          }else{
38              p.val=in[0]-'a';
39              p.plus=in.size()-1;
40          }
41          pat[i]=p;
42      }
43      sort(pat,pat+L);
44      return true;
45  }
46  struct Card{
47      int x,plus;
48  };
49  ostream &operator<<(ostream &os,const Card &c){
50      return os<<"("<<c.x<<","<<c.plus<<")";
51  }
52  bool operator==(const Card &a,const Card &b){
53      return a.x==b.x&&a.plus==b.plus;
54  }
55  Card cards[7];
56  int val[10];
57  double dp[8][61][61];
58  bool ac[8][61][61];
59  vector<int> c;
60  bool mdfs(int now,int used,int val){
61      //cout<<val<<endl;
62      if(now==(int)c.size()){
63          return true;
64      }
65      if(pat[now].val==-1){
66          REP(i,L){
67              if(used>>i&1)continue;
68              if(mdfs(now+1,used|(1<<i),val))return true;
69          }
70          return false;
71      }else if(now>0&&pat[now].val==pat[now-1].val){
72          REP(i,L){
73              if(used>>i&1)continue;
74              if(c[i]!=val+pat[now].plus)continue;
75              if(mdfs(now+1,used|(1<<i),val))return true;
76          }
77      }else{
78          // 新しい
79          REP(i,L){
80              if(used>>i&1)continue;
81              if(mdfs(now+1,used|(1<<i),c[i]))return true;
82          }
83      }
84      return false;
85  }
86  bool match(){
87      c=vector<int>();
88      REP(i,L){
89          c.push_back((cards[i].x+1)*100+cards[i].plus);
90      }
91
92      return mdfs(0,0,0);
```

```
}

double ans=0;

double solve(int k, int back,int count){
    double ans=0;
    if(k==L)return 1;
    if(ac[k][back][count])return dp[k][back][count];

    if(k==0){
        for(int i=1;i<=M;i++){
            ans+=solve(1,i,1)*1.0/M;
        }
    }else{
        if(cards[k]==cards[k-1]){
            ans+=solve(k+1,back,count+1)*(N-count)/(M*N-k);
        }else{
            if(cards[k].x==cards[k-1].x){
                if(back+1<=M){
                    ans+=solve(k+1,back+1,1)*(N)/(M*N-k);
                }
            }else{
                for(int i=back+2;i<=M;i++){
                    ans+=solve(k+1,i,1)*(N)/(M*N-k);
                }
            }
        }
    }
    ac[k][back][count]=true;
    return dp[k][back][count]=ans;
}

double _fact[8]={1,1,2,6,24,120,720,5040};
double solve(){
    double tmp=_fact[L];
    int cnt=1;
    for(int i=1;i<L;i++){
        if(cards[i]==cards[i-1]){
            cnt++;
        }else{
            tmp/=_fact[cnt];
            cnt=1;
        }
    }
    memset(ac,0,sizeof(ac));
    tmp/=_fact[cnt];
    return solve(0,0,0)*tmp;
}
void dfs(int k,int used,int plus){
    if(k==L){
        if(match()){
            ans+=solve();
        }
    }else{
        if(k==0){
            cards[0]=(Card){0,0};
            dfs(1,0,0);
            return;
        }else{
            cards[k]=(Card){used,plus};
            dfs(k+1,used,plus);
            // plus を増やす
            cards[k]=(Card){used,plus+1};
            dfs(k+1,used,plus+1);

            cards[k]=(Card){used+1,0};
            dfs(k+1,used+1,0);
        }
    }
}
int main() {
```

```
    while(input()){
        ans=0;
        int ok=true;
        int cnt[3]={};
        REP(i,L){
            if(pat[i].plus)ok=false;
            if(pat[i].val>=0){
                if(cnt[pat[i].val]++)ok=false;
            }
        }
        if(ok){
            ans=1;
        }else{
            dfs(0,0,0);
        }
        cout<<setprecision(9)<<setiosflags(ios::fixed);
        cout<<ans<<endl;
    }

    return 0;
}
```

## G 問題

多角形の凹点 2 点間を結ぶ線分を列挙する．交差しないように出来るだけ多くを選ぶことで解を求めることが出来る．これは二部グラフの最大マッチングに帰着可能である．

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
using namespace std;
int H,W;
#define REP(i,x)for(int i=0;i<(int)x;i++)
string MAP[102];
typedef pair<int,int> P;
typedef pair<P,P> L;
int dx[4]={1,0,-1,0},dy[4]={0,1,0,-1};
int px[4]={1,0,0,1},py[4]={1 ,1 , 0, 0};
typedef int Weight;
const Weight INF=99999999;
struct Edge{
    int src,dst;
    Weight weight;
    Edge(int f, int t, Weight c):src(f),dst(t),weight(c){}
};
struct Node:public vector<Edge>{ // 頂点の情報を記録する場合
};
bool operator<(const Edge &a,const Edge &b){
    return a.weight<b.weight;
}
bool operator>(const Edge &a,const Edge &b){return b<a;}

typedef vector<Node> Graph;

typedef vector<vector<Weight> > Matrix;
bool dfs(const Graph &G, int v, vector<int> &match, vector<bool> &visit){
        visit[v]=true;
        REP(i, G[v].size()){
                int dst=G[v][i].dst;
                int w = match[dst];
                if(w == -1 || (!visit[w] && dfs(G, w ,match ,visit))){
                        match[v] = dst;
                        match[dst] = v;
                        return true;
                }
        }
        return false;
```

```
}
int bipartite_matching(const Graph &G){
        int res = 0;
        vector<int> match(G.size(),-1);
        REP(v, G.size()) if(match[v] == -1){
                vector<bool> visit(G.size());
                if(dfs(G, v, match, visit)) res++;
        }
        return res;
}
int comp(const P&a,const P &b){
    return P(a.second,a.first)<P(b.second,b.first);
}
int main() {
    while(cin>>H>>W,H){
        MAP[0]=string(W+2,'.');
        REP(y,H){
            cin>>MAP[y+1];
            MAP[y+1]="."+MAP[y+1]+".";
        }
        MAP[H+1]=string(W+2,'.');
        // 凹点を探す
        vector<P> pit;
        H+=2;W+=2;
        int n=0;
        REP(y,H)REP(x,W){
            if(MAP[y][x]=='#'){
                // 凸点判定
                REP(d,4){
                    if(MAP[y+dy[d]][x+dx[d]]=='.' &&
                            MAP[y+dy[(d+1)%4]][x+dx[(d+1)%4]]=='.' ){
                        n++;
                    }
                }
            }
            if(MAP[y][x]=='#'){
                REP(d,4){
                    if(MAP[y+dy[d]][x+dx[d]]=='#' &&
                            MAP[y+dy[(d+1)%4]][x+dx[(d+1)%4]]=='#' &&
                            MAP[y+dy[(d+1)%4]+dy[d]][x+dx[(d+1)%4]+dx[d]]=='.' ){
                        pit.push_back(P(x+px[d],y+py[d]));
                    }
                }
            }
        }
        sort(pit.begin(),pit.end());
        pit.erase(unique(pit.begin(),pit.end()),pit.end());
        n+=pit.size();
        vector<L> cut; // 分割線
        int xc=0;
        REP(i,pit.size()-1){
            if(pit[i].first==pit[i+1].first){
                // x が一致
                bool ok=true;
                int miny=min(pit[i].second,pit[i+1].second),maxy=max(pit[i].second,pit[i+1].second);
                int x=pit[i].first;
                for(int y=miny;y<maxy;y++){
                    if(MAP[y][x]=='#'&&MAP[y][x-1]=='#'){

                    }else{
                        ok=false;
                        break;
                    }
                }
                if(ok){
                    cut.push_back(L(pit[i],pit[i+1]));
                }
            }
        }
        xc=cut.size();
        sort(pit.begin(),pit.end(),comp);
```

```
        REP(i,pit.size()-1){
            if(pit[i].second==pit[i+1].second){
                bool ok=true;
                int minx=min(pit[i].first,pit[i+1].first),maxx=max(pit[i].first,pit[i+1].first);
                int y=pit[i].second;
                for(int x=minx;x<maxx;x++){
                    if(MAP[y][x]=='#'&&MAP[y-1][x]=='#'){

                    }else{
                        ok=false;
                        break;
                    }
                }
                if(ok){
                    cut.push_back(L(pit[i],pit[i+1]));
                }
            }
        }
        Graph G(cut.size());
        for(int i=0;i<xc;i++){
            for(int j=xc;j<(int)G.size();j++){
                if(cut[j].first.first<=cut[i].first.first&&cut[i].first.first<=cut[j].second.first){
                    if(cut[i].first.second<=cut[j].first.second&&cut[j].first.second<=cut[i].second.second){
                        G[i].push_back(Edge(i,j,0));
                        G[j].push_back(Edge(j,i,0));
                    }
                }
            }
        }
        int match=bipartite_matching(G);
        int d=G.size()-match;
        cout<<n/2-1-d<<endl;
    }
    return 0;
}
```

# プログラミングを人に聞く前にすべきソリューション集

kakakaya

**はじめに**

プログラミングをする授業でとりあえず人に全部聞く勢に言いたいと思ってたけどコミュ障なので書きました。内容はググれば出てくる程度未満ですのでガチ勢は期待しないでください。また、授業で使っているのは C 言語なのでそれに合わせて書いています。

## 1 コンパイルが通らない

とりあえず適当に書くととりあえず起こる。まだ焦るような時間じゃない。まずエラーメッセージを読もう。
コンパイルが通らない、というのはコンパイラが誤りを検出している、ということであり、何が悪いかはエラーメッセージが教えてくれています。隣の席に座っている人に聞くのも良いですが、エラーメッセージに書いてあることを読めばだいたい解決します。英語ですが、行番号と書いてある単語を読めば何が言いたいかはわかるでしょう。
ここで書いているのはよく見る奴メインなので、書いていないのが起きたり、ひと通り見たい人は「gcc エラー一覧」で検索。

### 1.1 incompatible implicit declaration of built-in function と出る

おまじない部分を確認しよう。ヘッダの include が不足したりしているなど。なお、この書いてある通りのはエラーでなく警告なのでコンパイルは成功します。

### 1.2 '...' undeclared と出る

変数宣言忘れ。

### 1.3 stray '...' in program

ASCII 外（日本語など）が挟まっているとこういうのが出ます。

### 1.4 expected ')' や';' が出る

括弧が閉じられていない、関数の後に; を入れていない、変な文字が挟まっている、など。

### 1.5 too few—many arguments to function '...'

引数が多い・少ない。

### 1.6 expected '...' but argument is type of '...'

引数の型が求められているのと違う。ポインタ関連でやらかしやすい？

## 2 コンパイルが通ったけど上手く動かない

ここからが本当の地獄だ……

### 2.1 変な数字が出る・セグメンテーション違反が起きる

配列の範囲外にアクセス、アドレスの受け渡しミスなどが予想される。

### 2.2 表示が止まらない・実行が終わらない

終了条件設定ミス。

### 2.3 何も出ない・その他

本格的にデバッグしよう！

## 3 原因不明の恐怖　〜デバッグ〜

コンパイルが通らないとか、明らかに実行結果が何かおかしいならば、それを見て対策を講じることが出来る。だが、もしそうでないならば……

### 3.1 とりあえず変数を見てみる

所々に printf を挟んで終了条件や問題となっている変数の様子を確認する。たいていはこれでなんとかなる。

### 3.2 ブレークポイントを設定したり、ステップ実行をする

幾つかの変数を見るだけでは済まされないような状況の場合、本格的にデバッグが必要となる。この時、gdb を Emacs 上

で動かすと非常に楽なので一度やってみると良いと思う。筆者は http://techno-st.net/2008/08/28/gdb-emacs.html を参考にやってみたが、実際快適であった。

# 4 そもそも修正が何度も必要なプログラムを書かないために

コードの乱れは心の乱れ、心の乱れは家庭の乱れ、家庭の乱れは社会の乱れ、社会の乱れは国の乱れ、国の乱れは宇宙の乱れ。綺麗なコードを心掛けましょう。

## 4.1 Emacs を使う

Emacs は良いエディタであり、OS です。Emacs でファイルを開く→記述する→ Emacs を終了する→コンパイルする→ Emacs を再度立ち上げる、という人をよく見ますが、Emacs 内で M-x shell を実行するとシェルが立ち上がるのでその中で実行する方が何かと手間がかからなくてよいでしょう。また、Emacs からメールを送ったり、w3m を開いたり、電卓開いたり、五目並べしたり、その他色々行えます。残念なことに、Emacs で Emacs を開くことはできませんが……

## 4.2 インデントを揃える

時々スペースでわざわざインデントしている人を見ますが、普通のプログラミング用のエディタなら Tab を押せば正しい位置にインデントされます。Emacs は当然として Vim でもできます。Vim でさえも、です。

## 4.3 分かりやすい変数名を心掛ける

数値や文字関係なく、必要になるたびに a からどんどん変数を宣言していく酷いコードを見たことがありますが、外から見ても何がしたいのかさっぱりわかりませんでした。読みやすいコードのために、何の変数か分かる変数名をつけましょう。また、命名規則を揃えるのも重要です。個人的にはハンガリアン記法が好きです。

## 4.4 コメントを挟む

コメントは重要です。ちなみに、使っている C 言語の規格が C99 なら C++ の一行コメント[*1]が使えるので積極的に使っていきましょう。

[*1] a = test;//hoge みたいな奴

## 4.5 Emacs を使う

とても大事なことです。Emacs はエディタであり、環境であり、全てです。あと、C-x 2 とか C-x 3 すると楽しくなるのでやりましょう。切り替えは C-x o です。削除は C-x 0 です。

# 5 何とかなったが、正しい答えが出ない

それはどうにもならないです。アルゴリズムなんかは経験や知識、直感が大きい要素なので、人にある程度意見を求めるのも已む無しだと思います。……が、出来るだけ自分で考えるようにしましょう。

# 6 終わりに

この記事は書き始めも書き終わりも締め切りの 13 日後に（ｒｙ
皆さんプログラミングしましょう。楽しいですよ。個人的にはピタゴラスイッチやインクレディブル・マシーンみたいなものだと思ってやってます。授業では C だの Java だのばかり[*2]やってますが、もっと別の言語[*3]で自分の興味が有るものを作ってみたりプログラミングコンテストに参加してみるのはきっと楽しいですよ。
あとこの記事のソースの TEX ファイルも 128 行になって素敵だと思いました。

[*2] I 科では今現在 C 言語しかやってない
[*3] インデントが綺麗で読みやすくて素敵な Python が個人的には好きです。

# for(;;) と while(1) の熱き戦い

kzm，mznh

**はじめに**

この世には様々な争いがある．1 つは Windows と Unix,1 つは Vi と Emacs,
そして 1 つは for と while の無限ループについてである．

## 1 概要

電気通信大学では 1 年生は後学期に基礎プログラミング及び演習という授業で C 言語を学習する．C 言語には直接無限ループを表す文が無い，そこで我々は while(1) や for(;;) といったものを使うのだが，この 2 つどちらのほうが優れているのだろうか？

## 2 2 つの違い

for(;;) と while(1) この 2 つにはどのような違いがあるのだろうか，思いつく点をあげてみる．

- for(;;) の利点
  - while と比べタイプ数が少ない．
  - 最後の;) がかわいい
- while(1) の利点
  - while はループを行う構文なので意識的に無限ループだとわかる．
  - while('o')って書くとかわいい

ちなみに執筆者の mznh は while 派で kzm は for 派である．

## 3 コンパイル結果を比較してみる

客観的な比較をするために以下のソースコードを GCC の -S オプション用いてアセンブラに出力させてみた．

for.c

```
main(){
  for(;;);
}
```

while.c

```
main(){
  while(1);
}
```

なおコンパイラの最適化を排除するために-O0 オプションも追加してコンパイルした．また GCC は Windows MinGW version 4.6.2 を用いた．

## 4 結果

出力されたものが以下である．

for.c

```
    .file  "for.c"
    .def  ___main;    .scl  2;   .type  32;   .endef
    .text
    .globl _main
    .def  _main; .scl  2;   .type  32;   .endef
_main:
LFB0:
    .cfi_startproc
    pushl  %ebp
    .cfi_def_cfa_offset 8
    .cfi_offset 5, -8
    movl  %esp, %ebp
    .cfi_def_cfa_register 5
    andl  $-16, %esp
    call  ___main
L2:
    jmp   L2
    .cfi_endproc
LFE0:
```

while.c

```
    .file  "while.c"
    .def  ___main;    .scl  2;   .type  32;   .endef
    .text
    .globl _main
    .def  _main; .scl  2;   .type  32;   .endef
_main:
LFB0:
    .cfi_startproc
    pushl  %ebp
    .cfi_def_cfa_offset 8
    .cfi_offset 5, -8
    movl  %esp, %ebp
    .cfi_def_cfa_register 5
    andl  $-16, %esp
    call  ___main
L2:
    jmp   L2
    .cfi_endproc
LFE0:
```

完全に一緒であった．完

## 5　結論

コンパイル結果が全く同じになったので好きな方を使えば良いのではないだろうか．もしくは `#define loop ~~for(;;)~~ ~~while(1)~~ for(;;)` とか最初に定義しておくとかどうでしょう．

## 6　最後に

この記事を執筆している途中に昼ごはんを食べにふらっと深大寺に行った．そこではゆっくりとした時間が流れていて，争いなどすべて心から消えていき，記事を書く気が失せました．なので完成度があれですがお許しください．

# ウィンドウマネージャを作ってみた

clear

**はじめに**

毎日使うウィンドウマネージャを自分で作れたらきっと楽しいだろう、ということで作ってみた。

## 1 X Window System とウィンドウマネージャ

Unix 系 OS では GUI を実現するために X Window System(X) が広く用いられています。X はクライアントサーバモデルを採用しており、テキストエディタや Web ブラウザといった各アプリケーションは「X クライアント」です。「X サーバ」はユーザが直接操作しているコンピュータの上で動作し[*1]、クライアントからやってくる要求 (例えば画面の描画要求) を処理するとともに、キーボードやマウスからの入力をクライアントに伝える役割を担います。

ウィンドウマネージャ(WM) はクライアントの一種ですが、他のクライアントが作成したウィンドウの配置や外観を制御するという特殊な性質を持っています。どの WM を使うかによってデスクトップの見た目や操作感は大きく変化します。WM にはデスクトップ環境[*2]の一部として開発されているものもありますし、また単体で開発されているものもあります。ウィンドウの扱い方も様々で、以下のようなものが存在します。

**スタック型**　ウィンドウを重ねて表示する、よくあるタイプ
**タイル型**　ウィンドウを重ねず、タイルのように敷き詰めて表示するタイプ[*3]
**コンポジット型**　各ウィンドウの描画を独立に行っておき、画面上でそれを合成するタイプ[*4]

## 2 必要なもの

### 2.1 開発環境

X 上で動作するプログラムを開発する場合、GTK+ や Qt といった GUI ツールキットを使うのが一般的ですが、今回はより低いレイヤーに位置するライブラリである Xlib を使いました。これを使うことにした主な理由は、普段使っている WM(dwm[*5]) が Xlib を使って書かれていたため、という単純なものです。

Xlib は X クライアントを作成するための C 言語向けのライブラリです。古いライブラリなので現在は XCB という後継のライブラリを使うことが推奨されていますが、比較的小さな WM の実装などには未だに用いられているようです。Xlib や XCB では GUI を構成するための各種部品 (ウィジェット) は定義されておらず、ウィジェットの提供は上位の GUI ツールキットに任されています。

### 2.2 知識

実際に WM を実装するためには、WM が行うべきことについて知る必要があります。私の場合は完全に「ソースコードが手本」で、普段使っている dwm のソースコードを読んでみた[*6]ことが WM についての知識を得るきっかけとなりました。Xlib のドキュメントを開きながら dwm のソースコードを追ってゆくことで大まかな知識は得られたと思います。少し分かってきた所で他の WM の実装も見てみようと思い、ミニマリスト向けの WM として知られている evilwm[*7]が Xlib を使って書かれていたので一緒に読むことにしました。

[*1] サーバとクライアントの位置関係は一般的なクライアントサーバモデルと逆になっており、サーバが常にユーザの手元で動作する
[*2] メジャーな例としては GNOME と KDE が挙げられる
[*3] awesome など。なお、スタック型として動作させることも可能なものが多い
[*4] Windows Vista 以降の Desktop Window Manager や Mac OS X の Quartz Compositor はこれにあたる。他には Compiz など
[*5] `http://dwm.suckless.org/`
[*6] dwm は比較的小さい WM でとっつきやすかったのが大きい
[*7] `http://www.6809.org.uk/evilwm/`

# 3　何をどうやって実現するのか

## 3.1　WM がすべきこと

WM の基本的な機能をまとめてみます。

**ウィンドウ配置の制御**

ウィンドウ配置の制御は WM の最も基本的な機能です。これ無くしては WM と呼べません。X において、WM が動作していない状態でユーザがウィンドウを移動したりサイズを変更することは基本的にできないので[*8]非常に重要です。

**ウィンドウの装飾**

Windows や Mac OS でもおなじみですが、通常ウィンドウの周囲にはタイトルバーやボタンなどの装飾が施されます。X においては、ウィンドウの装飾は WM に一任されています。これは Windows や Mac OS のウィンドウシステムと大きく異なる点です。どのような装飾を施すかは WM によって様々で、これが見た目の違いにつながります。画面スペースを有効活用することを目的としてほとんど装飾を施さない WM も存在します。[*9]

**入力フォーカスの管理**

フォーカスとは「どのウィンドウが入力を受け取るか」を表す情報です。例えばキーボードから何か文字を入力すると、その情報は現在フォーカスを得ているウィンドウに送られます。フォーカスの割り当て方にはいくつかの流儀があり、圧倒的に多く見られるのはクリックされたウィンドウがフォーカスを得る方式 (click to focus) ですが、他にマウスポインタの下にあるウィンドウがフォーカスを得る方式 (focus follows mouse) などもあります。

## 3.2　イベント駆動と WM

一般的に、GUI 上で動作するプログラムはイベント駆動型となります。WM もイベント駆動であることに変わりはないのですが、他のプログラムが作成したウィンドウを操作するという性質から一般的なプログラムに比べて監視対象とするイベントの幅が広く、時には他のプログラムに届くはずのイベントを横取りして自分の処理を割りこませることもします。

ウィンドウを管理するという目的を果たすために WM は様々な種類のイベントを処理しますが、紙面の都合上ここでは 2 つだけ例を示します。

**例 1. ウィンドウの表示要求**

X では新たにウィンドウを作成しただけでは表示されず、そのウィンドウを表示するよう X サーバに要求する必要があります。何もないと (WM が存在しないと) その要求は直ちに実行されますが、WM はこの表示要求を監視して処理を割りこませることができます。例えば、ウィンドウにタイトルバーなどの装飾を施す WM はこのタイミングで装飾を行います。

**例 2. ユーザ入力全般の処理**

先にも挙げた通り、ユーザの操作に応じてウィンドウを移動したり、ウィンドウのサイズを変えるのは WM の基本的な機能です。マウスのボタンが押されたり、あるいはキーボードのキーが押されたりするとそれぞれ対応するイベントが発行されるので、WM はこれを監視することによってユーザの入力を検知し、必要な処理を行います。

# 4　作ってみる

とにかく実装して動かしてみないことには始まらないので、最初に大まかな仕様を決め、細かい挙動に関しては実装しながら詰めてゆきました。最低限の実用性を確保しつつ可能な限り実装量を減らす方向で考えた結果、次のようなものに落ち着きました。

**種類**　スタック型 WM

**機能**　基本的なウィンドウ操作 (移動、サイズ変更、クローズ)、ウィンドウの巡回、コンパイル時に指定したプログラムの起動

[*8] WM は 1 つのクライアントに過ぎないので X を動作させる上で必須の存在ではないが、無いと修羅の道を歩むことになる

[*9] タイル型 WM には目立った装飾を施さないものが多い

**装飾**　なし (ウィンドウの枠だけ)
**その他**　マルチディスプレイに対応[*10]

WM の内部で用いるデータ構造や各イベントに対する処理をどのように行えばよいかといった実装の詳細に関しては、dwm や evilwm を参考にしました。紙面の都合上ここでは作成した WM のソースコードを示すことができません[*11]が、後日どこか[*12]で公開するかもしれません。

## 5　It works.

少し機能を追加してはバグを見つけてそれを直すということをしばらく続けた結果、曲がりなりにも動くものができました。作った WM が実際に動いている画面のスクリーンショットを図 1 に示します。[*13]

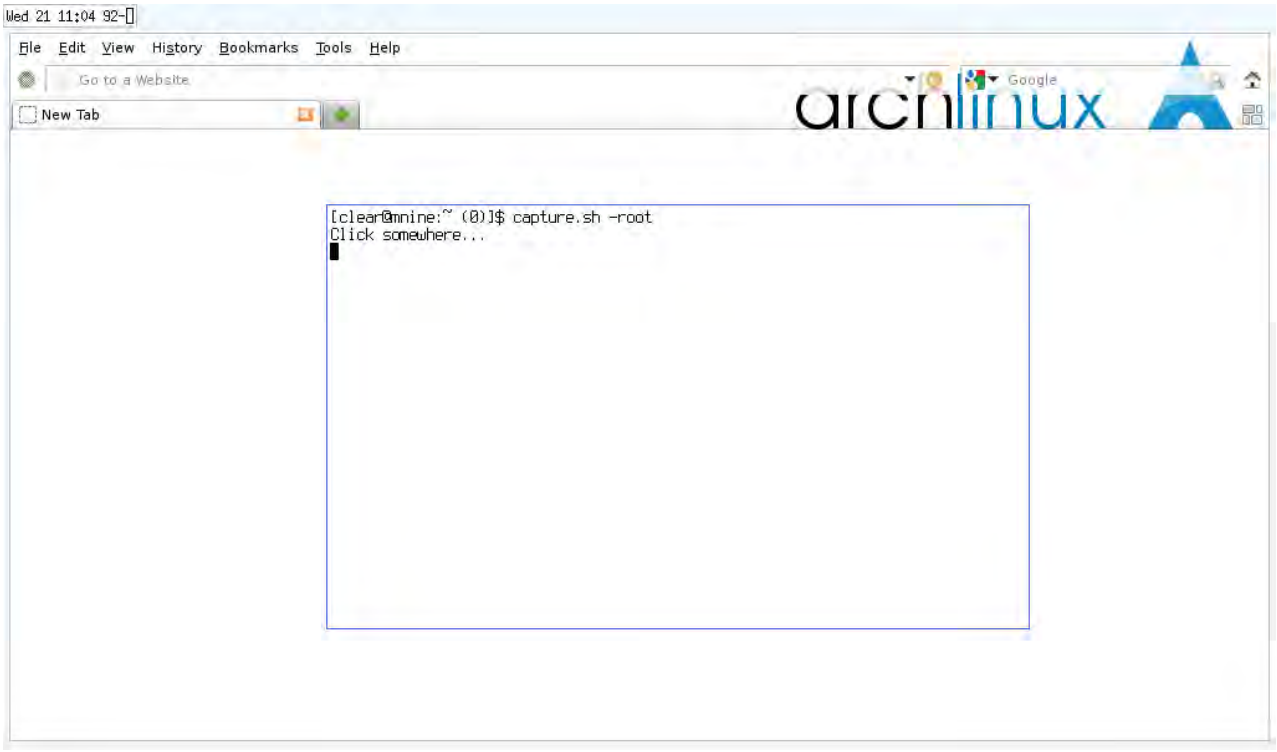



図 1　WM が動作している様子

今回作った WM は機能こそ少ないですがなんとか実用には耐えるので、早速色々な環境でコンパイルして使っています。そのうち不満に突き動かされるか必要に駆られるかして新たな機能を実装することになりそうです。あれこれ試行錯誤しながら書き足していった結果として非常に見通しの悪いコードとなってしまっているので、まずはそれを何とかする所からでしょうか。

## 6　おわりに

実はこれまでに数回 WM を作ろうとしては途中で闇に葬り去ることを繰り返してきたのですが、ようやくそれなりに使えるものを作れたので良かったです。自分で使うためのものを自分で作る分には、失敗してもやり直しが効くので気軽に取り組めます。WM に限らず、面白そうな題材が見つかったら作ってみると楽しいかもしれません。

[*10] ノート PC にプロジェクタを接続したときにも使えるようにする必要があった
[*11] 作ってみたと主張しつつ一行もコードを示さないという暴挙
[*12] おそらく `http://www.mma.club.uec.ac.jp/~clear/` あたり
[*13] 困ったことに、画像だけでは本当に動いているのかどうか分からない

# 気がついたら花火師になっていた件

mernao

**はじめに**

MMA では毎年夏の合宿で花火を揚げる。これだけ聞くと普通であるが、実はこの花火が隅田川の花火大会で打ち揚げられるような 4 号球の花火なのである。もちろん資格を持っていなければ無理なのであるが、気が付けば私は花火を打ち揚げる「花火師」と化し、花火を打ち揚げていた。

## 1　MMA と花火の関係

MMA では新歓期に「コンピュータのサークルでーす」とか言って宣伝して回っているが、突然「実は花火を打ち揚げるサークルなんだよ！」とか意味不明なことを言い出す。が、あながち間違いではない。MMA は夏の合宿で隅田川で打ち揚げるのと同じ 4 号球の花火を打ち揚げるのだ。これはどうやら MMA の伝統らしく、代々受け継がれているようである。私も入部した時はよく分かっていなかった為、1 年の頃に部室を漁っていた (別にやましいことをしていたわけではない) 時に変な鉄製の筒 (打ち揚げ用の筒) を見つけた時には一体何なのか理解が出来なかった。こいつの役割は夏合宿で初めて知ることとなる。とにかく MMA は花火サークルなので今年 3 年になった私も遂に花火を打ちあげる事となったのだ。

## 2　煙火消費保安手帳取得

花火を打ち揚げるためにはまず打ち揚げの資格を得なければならない。具体的には講習に参加して煙火消費保安手帳を交付されなければならない。ちなみにこれは「花火を打ち揚げて良い免許」ではない。あくまで講習に参加したという証である。毎年、都内の某花火商店[*1]が開催している煙火消費保安手帳講習会に MMA は参加している。1 万円を払ってここで 2 時間程度の講習を受ければ講習は終わりである。内容は主に過去の事故例である。非常にコワーイ事故例を紹介されるので正直花火なんてやりたくなくなる。特に打ち揚がらずに筒の中で開発してしまう「筒ばね」は爆発の威力で筒の破片が吹き飛んできて非常に恐ろしいので死んでもこんなことにはなりたくない[*2]と思った。しかし講師の方の愚痴のようなものも聴けて面白かったりもする。1ヶ月ほど後に煙火消費保安手帳が郵送されてくる。

## 3　下見

図 1　下見時の打ち揚げ場所

[*1] 別に隠す必要はないと思うが。しかし MMA は特殊な事情があって参加できているようなので他の人が参加できるかどうかは微妙である。

[*2] というか死ぬ

無事に煙火消費保安手帳が交付されたら次は打ち揚げ会場の下見に行かなければいけない。色々移動が必要なため毎回[*3]車で行くが、その道程は長く丸一日は普通にかかる。MMA は近年、伊豆の修善寺の狩野川の川岸で打ち揚げを行なっている。まずは 60m の保安距離[*4]の取れる場所、つまりは周りに何もない場所を探す。草が生えていたりしてもいけない。幸い去年の打ち揚げ場所がそのまま使えそうだった為すんなり済んだ。経験上、あまり早い時期に下見に行ってしまうと夏であるから本番時には草だらけなんてこともありえるので、1ヶ月前ぐらいにしておいた方がいい。今回は 8 月のお盆過ぎに下見に行った。

## 4　書類の提出

下見が済んだらその足で河川管理事務所と消防署に赴く。多分どっちに先に行っても良いだろうが、最悪「先にあちらの許可を取ってください」と双方に言われてデッドロックになることがあるらしい[*5]が、今回はそんな事は無かった。ちなみに今回は先に消防署に行った。田舎の消防署だがかなり綺麗で立派な消防署で、調布の消防署より綺麗なのではという感じだった。行った時は訓練中で消火ホースが駐車場に展開されていたのでわざわざ片付けてもらって侵入するという VIP 待遇 (?) だった。ここで「煙火打上げ・煙火仕掛け届出書」を提出し諸注意を受けて打ち揚げの許可を貰う。提出するものとして、打ち揚げ場所周辺の地図に保安距離を書き込んだ地図が必要になる。当たり前だが煙火消費保安手帳は必須である。

消防署に必要書類を提出したので河川管理事務所に向かう。消防署では打ち揚げの許可を貰うが、河川の使用許可は別途河川管理事務所で貰う必要がある。ここで河川一時使用届出書を提出する。その際に消防署で貰った許可証の写しが必要となったので、やはり先に消防署に行ったのは正解だったのかもしれない。

これで各機関への届け出は完了である。消防署と河川管理事務所に伺う際は、事前に電話などでアポを取っておくことが望ましい。朝の 8 時頃に家を出て調布に向かい、途中大渋滞に遭ったので家に帰ったのは夜の 9 時頃であった。今年で運転免許を取って 2 年目であるが、それなりに疲れる旅である。昨年運転した時はまだ初心者であったために非常にヒヤヒヤモノであった。

## 5　花火の購入

申請をするのが山場であるため、ここまで来ると準備は万全の様に思えてしまうが、肝心の打ち揚げる花火玉を忘れてはいけない。[*6]花火は煙火消費保安手帳講習会を受けたのと同じ花火屋さんで注文した。今年は 4 号球の牡丹を 5 発、色はおまかせで注文した。この時に一緒に電気導火線を注文しないといけない。忘れると筒に直接火を投げ入れる方式を取るしかなくなり危険なので中止ということになってしまう。気をつけよう。この時、万が一に備えて電気導火線は必要数以上注文するのがいいだろう。送付先は合宿で宿泊する貸し別荘の方が受け取って下さるので毎回直接宅配便で店から送ってもらっている。さすがに部室に置きっぱなしにしたり、車で調布から運搬するのは御免こうむりたい。

## 6　打ち揚げ当日

打ち揚げ当日の朝は早い。というのは嘘だが、打ち揚げ当日の午後は忙しい。

### 6.1　穴を掘る

通常の花火大会では丈夫な木枠に筒を固定して打ち揚げを行うが、MMA では筒を地中に埋めている。これは万が一筒ばねを起こした場合に被害を最小限にするためである。筒の長さは 80cm 程度だがこれを埋めるのに一苦労する。というのも打ち揚げ場所が河原であるために、地面は石だらけなのである。しかも巨大な石が邪魔になるのでそいつを掘り起こすので大変になる。私も 1 年生の時にやったが、まさかコンピュータを扱うサークルの合宿でこんな作業をすることになるとは……という感じであった。幸い昨年と今年は自動車による人員輸送というミッションがあったため、私は穴掘りを逃れることができた。

[*3] 実は昨年も私は運転手として行っている。今年も運転したが。
[*4] 保安距離は打ち揚げる玉の種類や地域によって異なる
[*5] そんな時は「向こうにも同じように言われたのですが」と言えば回避できるとかできないとか
[*6] 昨年は忘れかけていて危なかった

図 2　過酷な肉体労働

## 6.2　筒を埋める

穴が掘れれば後は筒を埋めるだけだ。筒を固定できる杭を打ち込んで、筒と杭を縄でしっかりと固定する。この時に筒の溶接面を打ち揚げの際に人間の居ない方向に向けて筒を設置する。これもまた筒ばね対策で、筒ばねした際には強度の一番弱い溶接面から破裂するからだ。しっかり固定できたら周りを土で固めて筒の設置は完了である。

## 6.3　割り入れ

花火玉は点火するための親道というのがあり、打ち揚げる時にこの親道に割り入れと言われる作業をして点火ができるような状態にする。割り入れの方法については詳しく解説しないが、カッターナイフなどで切り開いていく。この時切り過ぎには注意する。割を入れたら親道に黒色火薬をかぶせるようにして、テープでしっかりと固定する。黒色火薬は最初に筒から弾を打ち揚げるための火薬である。この黒色火薬からは速火線が伸びていて、速火線の終端に電気導火線を取り付ける。電気導火線の取り付けは筒にセットしてから行うのでここではまだしない。この状態になるといつ火がついてもおかしくない状態になるので慎重に取り扱う。

図 3　割り入れ作業中

図 4　黒色火薬固定完了

### 6.4　筒への装填

いよいよ花火玉を筒に装填する。筒の中に入れる時は速火線を持ってそっと中に入れる。セットしたらいつ打ち揚がってもおかしくないので、絶対に筒の中を覗いてはいけない。速火線を筒から外に出して、終端に電気導火線を取り付ける。電気導火線に電流を流すために更に導線を取り付けて十分離れた場所まで引き伸ばす。このときに導線の抵抗値のチェックなどをしないといけないが、ここでは割愛する。

### 6.5　打ち揚げ

いよいよ打ち揚げを行う。と、ここで雨が少し降り出した。昼間は晴れていたのになんということだ。急いで筒などの火薬類が絶対に雨に濡れないようにした。幸い雨はすぐに止む様であったため、少しの間待機した。無事に雨は止み、速火線などへの吸水も確認されなかったので打ち揚げ可能と判断し、打ち揚げ作業に戻った。点火は伸ばしてきた導線に 9V の乾電池で電流を流すことで行った。合計 5 発、今年も無事に打ち揚げることができた。残念ながら写真はない。

### 6.6　後処理

打ち揚げが終わった後は後片付けをした。まず、念の為に周囲に火の気がないかをしっかりと確認した。次に筒の中に水を入れて完全に消火し、筒などを引きぬいて穴を埋めた。道具類はその日のうちに全て片付けた。翌日明るくなってから周囲にゴミなどが落ちていないかをもう一度確認してすべての作業は終了した。

## 7　感想

今までは傍から見ている役だったので気楽だったが、いざ自分がやるとなると非常に恐ろしかった。事故れば死者が出るのだから当たり前だ。今年も無事終了できたので本当に良かった。今後もずっと無事故で行きたいと心から願っている。打ち揚げた時は「あぁ綺麗だ」というより「あぁ無事に揚がった」という感じで、ほっとした。来年は是非、牡丹だけでなくて柳などの別の種類の玉を揚げてみたいと思う。

## 8　謝辞

今回の花火打ち揚げが無事終了したのも OB の方々の指導のお陰です。OB の方々には色々な場面で助けていただきました。この場を借りてお礼を申し上げます。本当にどうもありがとうございました。

## 9　免責事項

この記事を読んで花火を打ち揚げて何らかの被害を被った場合でも、私は責任を負いかねます。また、打ち揚げに必要なすべての事項が記載されているわけではありません。

# 痛グラス、痛ガラス制作

saharakei

**はじめに**

グラスリッツェンというものが世の中にはあります。それは専用の工具を用いてガラスに傷をつけ絵を書く、というものです。自分は過去にその技術で作られた工芸品を見たことがあるのですが、それは実に美しく魅了されました。ただその製品はとても高く自分に手の出せるものではありませんでしたが。 しかしその後調べてみると電動工具ルーターを用いて似たような事をするものがあるのを知りました。これには初期投資も 5000 円程度とあまり多くなく自分にもできそうだったので手を出してみることにしました。

## 1　制作に至るまで

いきなりルーターとガラスを渡されて「作ってみろ」と言われても作れる訳がありません。そこでネットでつくり方などを調べるわけですが、面白いものを見つけました。それは「痛グラス」と呼ばれるものであり、ガラスのコップにアニメ等のキャラクターの絵を彫るものでした。 絵が書けない自分がどのように模様を付けるかが問題だったのですが、これなら自分で絵を書かなくてもネットから拾ってくればいい、ということでこの「痛グラス」を作ることに決めました。

## 2　痛グラス、痛ガラス

痛グラス、痛ガラスというのは要するにガラスコップや板ガラスにアニメキャラなどが彫ってある「痛い」グラス、ガラスなわけです。そしてこの痛グラスには大きく分けると 3 つの種類があります。一つは外郭だけを彫り線画を作るもの、一つは彫るところと彫らないところで二色のモノクロ絵を作るもの、最後は濃淡を利用して絵を彫るものです。自分はだいたい二番目を作っています。三番目も作ってみたいのですが技術がたらず未だに作ることができません。

## 3　使用器具

さて、作るには道具が必要です。必要な道具は

- ルーター
- ビット
- ガラスコップ or 板ガラス（アクリル板）
- マスク
- 防護メガネ

です。ビットというのはリューターに取り付ける鑢みたいなものであり、細いのと太いの二つ用意するといいでしょう。また、マスクは削って飛び散った破片を吸い込まないために、防護マスクはガラスの粉が目に入らないようにするために必須です。ガラスコップは失敗してもいいように安い 100 均の物を買えばいいでしょう。板ガラスの場合は写真立てを買うといいです。

## 4　作成手順

まず初めに彫るもとになる絵を選びます。この絵はコントラストがはっきりしていた方がいいです。

次にその絵を白黒の二色絵にします。Windows を使っている場合は絵をペイントで編集し、**名前をつけて保存**から**モノクロビットマップ**（*.bmp,*.div）を選択し保存します。これでカラーの絵がモノクロになります。

絵を印刷し、グラスの内側やガラスの裏側に貼り付け上から油性マジックペンで下書きします。コップの内側に絵を貼り付ける時は絵を入れた後に布等を詰めてぴったりと貼り付ける必要があります。下書きはしてもしなくても構いませんがした方がやりやすいと思います。

なぞったら下の絵を外します。下書きしない場合はそのままで大丈夫です。

これで下準備は完了です。これから実際にガラスを彫っていきます。また、彫っていくにあたり注意点がいくつかあります。

- ガラスの粉末が目に入ると危険なため安全ゴーグルを付ける
- ガラスの粉末を吸い込むと危険なためマスクをする
- ルータは長時間使用すると発熱し壊れるため適度に休ませながら使う

これらに注意しながら彫っていきましょう。

まずは細いビットを使って輪郭を彫っていきます。ここでやめると図 1 のような線画の痛グラスになります。

彫り終わったら次に面を太いビットで彫っていきます。ミスしないように広い部分から彫っていくといいです。

基本的にこれで終わりで後は自分の好きなように彫っていけばいいでしょう。気を付けるべき事といえばマジックで書いた下書きが手汗や手油で消えないようにすること、ガラスに彫るのでやり直しが聞かないこと、です。前者は消えにくいペンを使う等すればいいのですが後者はただ気を付けるしかありません。

図 1　線画

## 5　完成作品

今まで作成手順をほとんど図なしで書いてきたので完成作品を次ページに並べました。作品の説明を元ネタと共に簡単に説明したいと思います。

**線画**　魔法少女まどか☆マギカの佐倉杏子と美樹さやか、表と裏に線で彫ってあります。
**左上、中上**　デスノートのリュークと夜神月、表と裏に彫ってあります。
**右上**　めぞん一刻の管理人さん。個人的にこの作品が一番うまくできたと思っています。
**左下**　魔法少女まどか☆マギカの暁美ほむら 白黒化に少し失敗してしまいました
**中下**　化物語の八九寺真宵、DVD ジャケットから絵を拝借しました。
**右中**　ネットで見つけた桜の切り絵を題材にしました。切り絵と痛ガラスの表現方法は同じなので参考になります。
**右下**　劇場版魔法少女リリカルなのはのミッドチルダ式魔法陣、細かいところが多くて疲れました。

これらはだいたい早いものは 1 時間ほど、遅いものは 10 時間程かかりました。一番早かったのは線画で一番遅かったのは桜と魔法陣です。この二つは特に細かく、神経を使う作業だったのでゆっくりとしかできませんでした。ですが、この二つを除いた作品の作業時間は大体 2〜4 時間なので結構気楽にできると思います。

## 6　まとめ

自分がこの痛グラス制作を始めたときに用意したものは 4000 円のビット付きリューター、100 均の安全メガネとマスク、そしてガラスコップだったので実際 5000 円以内で始めることができました。また、ダイソーならばリューターが 600 円で売っていることもあり、1000 円あれば作れます。ただ、安物だとリューターの連続使用時間が短かったりパワーが弱かったりする為ホームセンター等でしっかりしたものを買うといいでしょう。

自分はまだ初めて半年もたっていませんが痛グラスを作るのに特別なスキルが必要無いのでこれくらいの作品を作れるようになりました。制作時間も複雑なものを作らなければ短くてすみ、手の出しやすい値段で始めることができ、ちょっとした小物も作れるので気が向いたらやってみてはいかがでしょうか？

図 2　作成品

## 7　参考試料

彫る下絵にしたものの出典を示します。

- 佐倉杏クリーナークロス
- 美樹さやかクリーナークロス
- 漫画 DEATHNOTE 夜神月
- 漫画 DEATHNOTE リューク
- めぞん一刻　管理人さん
- DVD 化物語 第二巻 / まよいマイマイ
- 魔法少女まどかマギカ TPS FEATURING 暁美ほむら http://aniapp.animate.tv/madokamagica-tps
- 桜の切り絵 http://www.pixiv.net/member_illust.php?mode=medium&illust_id=27103343
- リリカルなのは魔法陣 映画版 http://kyubishuppan.com/gallery2/magicgate_movie.html

# 本気、出してみませんか

zico

**はじめに**

「日本の学生は勉強しない」と言われて久しいが、ここではその原因を企業が学生に求める能力の基準に見出した。そして活発なコミュニケーションが奨励される真の意味を知り、勉強不足な筆者自身を啓発しようと思う。

## 1　勉強しない学生

日本の大学生の勉強不足を懸念して、文部科学省は大学入学時と卒業時で 2 度受験する「共通テスト」の開発を検討している。在学中の伸びを調べ、大学教育の改善を促すというものだ。日本の大学生の 1 日の勉強時間は平均 4.6 時間[*1]だという。1 日 8 時間程度とされる欧米と大きな差がある[*2]。日本の学生はなぜ勉強しないのだろうか。

今日、世界的な向きとして、学校や企業ではグループワークが奨励され、プレゼンなどでの表現力も重視される。これに対して日本では「コミュニケーション能力 (コミュ力)」が重視され、我々学生はコミュ力向上に励んでいる (はずの) ことは周知の事実だ。しかし、このコミュ力という言葉は非常に曖昧である。

雑誌「大学ランキング」の中村正史編集長は次のように語る。

> 『コミュ力』の中には、読解力やら基礎教養、協調性など多くの要素が本来含まれているはず。それをまとめて呼び、就職には必須だと言われれば誰にも否定できない。変に流行しすぎている言葉だと思う。
>
> 朝日新聞『(be report)「コミュ力」が流行する時代』, 2012 年 7 月 21 日

コミュ力というと、主にプレゼンやグループでの会議などで自分の伝えたいことを人にわかりやすく効果的に伝える能力のように思われがちだ。これはテレビタレントやお笑い芸人に当てはまるような「トーク力」に似ているが、実際に企業が求めているのは単に社会人としての総合力であろう。しかし、今まで私が知る同世代の学生の多くは、就職についてリーダーシップや社交性ばかりを関心事としており、単位が取れる最低限の勉強をして、専ら部活・サークルでの活動や自分の趣味に注力しているようである。

また、コミュ力を総合力としても、大学での専門的な勉強が就職に繋がりにくいことが言える。日本経済研究センター (JCER) 研究顧問の深尾光洋氏によると、「日本企業は新卒、とりわけ文系の学生を採用するときに、大学時代の成績を問わないことが多い」[*3]という。これでは学生が就職のために勉強しないのは当然である。しかし実際は業種によってさまざまな専門知識が必要になる。それを入社後の社員教育ですべて習得させるのは難しいし、十分な教育はできないと思われる。かといって誰にでもできる仕事をしていては、グローバル社会で海外からの人材と張り合うことができない。企業が、仕事に関わる専門知識などの十分な学力を採用基準として明確に示せば、学生は就職のために勉学に励むはずであり、世界からみた日本の人的資源の質も向上するだろう。

## 2　豊かなコミュニケーション

現状の日本社会では、就職のために勉強するメリットがあまりないように思えるが、社会全体からしてみれば、我々学生がすべきことはやはり勉強だという事を言いたい。

松岡正剛氏は「21 世紀は、主題ではなく、方法の時代である」と語る。今日、日本でも国際社会でもあらゆる分野において、テーマやコンセプトはほとんど出尽くしている。これからは、山積みになった主題をどう分析し、動かしていくか、といった方法に注目するのがよいという[*4]。有効な方法を見つけ出すために今までよりも重要となるのが、知識やアイデアの交換、共有である。1 つのプロジェクトに対し、専門分野の異なる多様な人々が意見を出しあい、その豊かな議論から新しい方法が生まれる。

---

[*1] 朝日新聞, 2012 年 2 月 16 日

[*2] 同上

[*3] JCER『深尾光洋の金融経済を読み解く - 個々の企業でできる経済活性化』(`http://www.jcer.or.jp/column/fukao/index343.html`)

[*4] 松岡正剛事務所『方法と編集』(`http://www.eel.co.jp/pdf/edit-method.pdf`)

そしてグループワークの奨励が世界的に行われていることは先に述べた。ただし、単に他人との意見交換を積極的に行えばいいということではない。

作家の Susan Cain 氏は次のように語る。

> グループというのはその場の支配的ないしはカリスマ的な人の意見に従うものです。優れた話し手であることとアイデアが優れていることの間に相関なんて全くないにもかかわらず —
>
> Susan Cain “The power of introverts”(訳:Yasushi Aoki)[*5]

各個人の深い思索なしに、ただ話しあうだけでは良いアイデアなど生まれるはずがない。個人がそれまでに蓄えてきた知識や様々な経験、あるいはそれをもとに自分でじっくり考えぬいた「方法」があってこそ、それを交換することで生産的なコミュニケーションが実現する。我々学生について言えば、豊かな学業経験なしに豊かなコミュニケーションはありえないのだ。

今こそ、「勉強しない学生」の汚名を返上するときである。「明日から本気だす」ともったいぶらずに今日からでも、高尚な学問の世界に浸ってみよう。

[*5] http://www.ted.com/talks/lang/en/susan_cain_the_power_of_introverts.html

# もし最近話題の森が〇〇になったら

kakakaya

**はじめに**

最近話題になっている某森についてのネタ記事。締め切り前に書く予定だった記事は神の言語[*1]を扱いきれなくて消えました。
まだ新しい森やり込んでないので色々と突っ込まないでください。サメ釣り楽しいです。

## 1 もし某森が近代風の Web サービスになったら

- 村のデザインが無駄にスマートになる
- 事ある毎にイイネ！される
- 自分の村に人が来たらメールで通知される
- Facebook や Gmail の連絡先から友達のインポートが出来る
- GPS で自分の現在位置と連携、地域ごとのアイテムを獲得
- 各種ニュースサイトに「森に投稿」みたいなマークが付く
- フィッシングサイトが横行、パスワードを抜かれたユーザーがバグだ！と叫ぶ
- スマートフォン対応
- 芸能人はこぞって登録する
- 時々炎上する
- API が公開され、外部から自分の村や他人の村をどうにか出来るようになる
- おさ＊＊たーとかおさ＊＊　＊＊＊＊＊＊　＊＊ぶみたいなの使ってる人が色々な村を荒らしに来る。結果村人の口癖がキモい顔文字になる
- 時々非常にスタイリッシュな開発オフィスの様子が公開される[*2]
- どうやって収益化しているかが不明瞭になる

## 2 もし某森が RPG ツクールゲーになったら

もし El*na 風になったら、で書きたかったがさすがにカオスすぎて書けない。

- 鳥系の村人に捕まるとどこかへ連れて行かれる
- 牛系の村人は宇宙人に連れて行かれて改造される
- 額縁から手が生えて追ってくる
- 咲いているバラを全部散らすとゲームオーバー
- 一人でモーターボートで逃げ出すとバッドエンド
- 蛇っぽい上級者向け村人が終盤に正体を表す
- アイテムコンプリート後にベランダに行くとエンディング

有名なので揃えてあるので元ネタは適当な人に尋ねてください。

## 3 もし某森がリアルになったら

### 3.1 普通にリアルだったら

そこは檻のないサファリパーク、草食動物と同じ空間に入れられた肉食動物は瞬く間にその野生をむき出しにし、住民の数を半分以下にしてしまった……
人間でありながらここに連れてこられた私も当然被食者。サバイバルライフが今始まる。

### 3.2 中途半端にリアルになったら

- 自宅が買えるまで数十年単位でかかる
- 住民にハエがたかってたりする
- 一部の住民が冬眠する
- 住民に餌をやるとなつき、芸を仕込んだり出来る
- 時々災害が起きたりする
- 持てるアイテムの量が種類の量ではなく重量で換算される
- 部屋の模様替えに一々人を呼ばなくてはならない
- 昼間に人の家を訪ねていってもいない
- 昼間からぶらぶらしていると悪い噂が流れる
- 夜にぶらぶらしていると職務質問される
- 斧で村人を切ると病院送りに出来る
- スコップで石を叩いて[*3]いると不審者扱いされる
- 川とかに落ちることが出来、そのまま海を泳いでどこかへ行ける
- 釣りは餌の選択が重要
- 時々プロ市民みたいなのが村に来る。新聞や宗教の勧誘も来る
- 家に湧くゴキブリがめっちゃリアルになって 3D で飛び

[*2] 見学者には海鮮丼？

[*3] やる前に後ろに穴を掘っておくと反発の影響がなくなる。

出す。しかも踏むと中身が飛び散り、消えない
- さらに、気づかずに卵とか入った家具をアイテム欄に入れてしまうとアイテム欄まで侵食される
- そのまま放置しておくと 3DS 本体ホーム画面にまで湧き、当然 3D でリアルに飛び出す
- ( ’ω ’) うわあああああああああああ

## 4　もし某森が村長ではなく市長をするゲームになったら

2k と earth が好きです。わたし、新作の 5 が気になります！

- 村の形はどんどん変えていける
- 効率厨はピラミッド状の水力発電・水道用ポンプが大量に埋まった山を作って村が狭くなる
- 初期状態で村全体が低密度住宅地区となっているので、半分ほど高密度工業地区、2 割程度商業地区に変えるのが鉄板
- 公害が酷いと猫[*4]の住民がアリクイになる
- 「動物園を建築しろ」という要望が沸く
- そのうち飽きられて災害を起こされる

## 5　もし某森が東側の国にあったら

ソビエトロシアでは村の住民から 3DS が飛び出す！

- 人から呼ばれるときは常に「同志」
- 店は並ばないと入れないし、入っても売り物がない
- アメリカでは癒しのために動物園を作った。一方、プレイヤーは動物の群れの中に入った。
- （新築のプレイヤーの住居にて）「なぜ、私の家は注文と違う形なのかね？」「ペレストロイカ（建て直し）！」「それに、作りかけじゃないか？」「ウスカレーニエ（加速化）！」「誰かが住んだような形跡があるぞ？」「ガスプリヨームカ（国家品質承認）！」「君はなぜそんなことまでわざわざ教えてくれるのかね？」「グラスノスチ（情報公開）！」
- 「あれ、あそこに住んでた人どうしたの？」「村長の政策を批判してシベリア行きさ。愚かなことさ。君もそう思うだろう？」「そうだね、全くだ。あの村長にはうんざりさせられるよ。」「そうか、おめでとう！君は彼に会いにいけることになった！ちなみに私が愚かだと批判したのは彼のことだ――それと、君もだが。」

[*4] イグアナだっけ？

- ～開発陣の会議～
  開発責任者「村に登場させるのは何人必要だ？」アメリカ人「10 人もいれば十分だろう。」ドイツ人「脱走を防ぐため、倍の 20 人で相互監視をさせよう。」日本人「2 人で十分だ。村に来るプレイヤーと、廃村になったことを伝える者。」

## 6　もし某森がクトゥルフ神話に登場したら

(」・ω・)」うー！(／・ω・)／にゃー！

- 交通手段はバスのみ
- 住民は皆インスマス面
- 冒涜的な家具を家に設置したり、名状し難き服装[*5]を身にまとったり醜く歪んだ虫や魚を捕ったりすると SAN 値が減り、回復はしない
- SAN 値が 0 になると発狂し、セーブデータが消え、村人の仲間入りしてしまう

> 村に遊びに来たが、偶然バスが故障し村に泊まらないといけなくなった。そして深夜、停電が起こり、部屋の外から何か粘性を持ったものがドアに体当たりをするような音が聞こてくる。
> だが、まさかこの家が壊れるなんて無いはずだろう。
> ああ！そんな、まさか！あの手はなんだ！窓に！窓に！

――棚の下で見つけた奇妙な傷の付いた手記より。

## 7　もし某森がソーシャルゲーになったら

基本プレイ無料。今すぐ検索！※通信料と一部項目が有料になります。

- 上画面の表示
- 下画面のタッチすると飛ばされる広告の撤去
- 二人目以降のプレイヤー
- レアな魚・虫が捕れる釣竿・虫取り網（確率で壊れる）
- コラボ村人、声優のボイス付き村人、レアな村人は DLC
- 一日に動ける量が決まっており、スタミナドリンク的なものを大量に消費しないと村に来ない[*6]

[*5] 電通生の着ている奴
[*6] シンデレラなんとかは廃人になるのが怖いのでやってないですが輿水幸子ちゃんが誘い受けツンデレで可愛いと思いました

- 化石は最小単位が骨一本ずつになる。化石を一セット揃えると次の段階の化石の部位に変化する
- UI はしょぼい Flash
- 友達を紹介すると手に入るレアアイテムを目当てに、「紹介されてくれればスタドリ送ります」が横行する
- ゲーム間バトルが出来、ポケモ○と自分の所の村民を戦わせたり出来る[*7]

## 8　もし某森が先進的な感じになったら

- 村民との会話は主人公との会話から生成された文章になる
- 「道」「椅子」などを表す AR マーカーを設置し、カメラを通して住民が現実世界で遊ぶ様子が見れる
- 村人の行う行動は非常に多彩になり、勝手に木を全部切ったり主人公の家の家具を持って行ったり花を全部散らしたり店の物を買い占めたりする。
- 結果、制作に時間がかかり発売が遅れる
- 住民同士で子供を作ったりする。当然、主人公とも…

## 9　もし某森が FPS ゲーになったら

核戦争後の世界、数少ない安全な地下シェルターから犯罪者の汚名を着せられ逃げ出してきた主人公が汚染された地上で見たのは動物の顔をしたミュータントだった……

- 武器はパチンコと斧
- 敵の周りに穴を掘って行動を阻害し、味方に攻撃してもらうのが基本戦術。落とし穴を塹壕にするのもあり
- 村人の中にも派閥が存在し、敵対する派閥同士を戦わせたり出来る
- 体力をスタドリ的なもので回復出来る（青いストロンチウム入りの物は AP が回復する）
- 住民の MOD を入れることが出来、何のゲームかわからなくなったりする
- 最大 256 人までひとつの村に入れるようになったりする
- でも、頭がデカイので HS が楽

[*7] 村の地図厳選とか言う文化があるのを最近初めて知りました。

## 10　もし某森がプログラミング言語になったら

マインクラフトもチューリング完全だしなんとかなるのでは。適当に考えたので手作業でインタプリタごっこしてください。

### 10.1　記述方法

マップにアイテムを置いて記述。コメントはアイテムを埋める。実行は Befunge みたいに上下左右動く感じ。というかコンセプトが Befunge。
アイテムは種類ごとに命令の種類を決める。ベルを整数、小数は Float 宣言的なアイテム置いた後に桁数指定のベル + 実際の数値のベル、文字列は String 宣言的なアイテム +ASCII コード数値分のベル + ヌル文字的なアイテム。画像をデザイン機能で簡単に扱える…？
変数はアイテム欄。つまり使用可能なメモリ領域は 30Byte 前後に換算？マップに置いたりする命令も作ることでより難解になる……かもしれない。
標準入出力？なにそれおいしいの？

### 10.2　よくある事故

- いつの間にかハニワや化石が埋まってコメントがややこしいことに
- ゴミ捨て場のところにうっかりおいてしまい消え、村人が勝手に持って行ってしまい消える
- 作業のために木を伐採、花を散らしておいたら物寂しい村になる
- 1 億 5000 万台の 3DS で走る村人。

## 11　もし某森がニンジャが出て殺す小説になったら

最近書籍化も二冊目行ったことだし、そろそろアニメ化の話が出ても良いのでは。

- 週末は「アンタイセイ！アンタイセイ！」と叫ぶ白い犬[*8]がいる
- 釣れる魚の例：オーガニック・マグロ、合成ブリ粉末、タラバーガニ
- 実際安いたぬきの店

[*8] 序盤にヤシの実を投げてくる方が好き。

- ◆◆◆◆◆◆◆◆◆
- 「アニマル・ニンジャ・ビレッジ」 #1
- 雑草が生い茂り、サツバツとしたノボリが立てられ、マッポーめいたアトモスフィアをかもす村をトレンチコートとハンチング帽を身につけた男が歩いて行く。彼の目的地は村の外れにある家だ。だが、依頼人からもらった地図通りの家はヤクザベンツに囲まれている。 1
- 理由は明白、住人は悪質に家賃を滞納しているのだ。彼の家のドアは囲まれて棒で叩かれている！「そろそろ来るかな…」つぶやく悪質家賃滞納プレイヤーの家のドアを破りアサルトヤクザが突入！ 2
- 「スッゾ！」「スッゾ！」「スッゾ！」「スッゾ！」「「「「スッゾコラー！」」」」」ＣＲＡＡＡＳＨ!! 続けて窓の割れる音とともにメンポをかぶった男が侵入！マシーンめいてスリケン連続投擲！「「「「アバー！」」」」」殲滅！ワザマエ！ 3
- 「アイエエエ……何者だ……」家賃滞納プレイヤーは失禁しながら震え声で尋ねた。いつもどおりに窓を開けて逃げようとしていたら、この怪しい男が入って来たのだ。 4
- 「ドーモ、プレイヤー＝サン。ニンジャスレイヤーです。」ブッダ！聞いたことがある…忍者を殺す忍者だ！だがここにニンジャはいない！一体どのような目的が？ 5
- 「別の村の男からお前を助けるように依頼されてきた。さあ行くぞ」「ちょ、ちょっと待ってくれよ…家賃こそ滞納しているが、脅されるだけで命まで狙われたことはないし、村の人は優しいし…」 6
- 「そんなことはない、奴らは」ＣＲＡＡＡＳＨ!! 再び破壊音！ 反対側の窓が割れ村長突入！そのままアンブッシュ攻撃！「イヤーッ！」ニンジャスレイヤーはこれを側転回避！ 7
- 「ドーモ、ニンジャスレイヤー＝サン。村長です。いらんことを教えおって……だが、その男を逃がすわけには行かぬのじゃ。」そういうと彼は背負っていた甲羅を脱ぎ捨てた…甲羅は床に穴を開けた。 8
- 驚くべきことに、彼はその重量を自身のカラテで支えていたのだ！その力実際およそ常人の三倍！「ドーモ、村長＝サン。わざわざ出向いてくるとはご苦労なことだ。だが従うわけにはいかない。」 9
- 村長だった男とコートの男が突然、驚くべき速度で戦闘を開始した！ニンジャ動体視力を持たないプレイヤーも凄まじい光景に再失禁！その時、足元の床がタタミごと崩落！ 10
- 「こっちだ！付いて来い」穴の底では黄色いヘルメットをかぶった作業員めいた男が立っており、その背のトンネルを指さしていた。（従うべきか…いや、こいつも村長の仲間なのでは？それに、コートの男が信頼出来るかも分かっちゃいない。）11
- （そもそも、どうしてこんなことになっちまったんだ？あのコートの男が来てからじゃないのか？）何にせよ、背後のイクサに巻き込まれてネギトロにされてはたまらない。立ち上がると、ひげの長い作業員に従ってトンネルを走ることに決めた。 12
- （「アニマル・ニンジャ・ビレッジ」 #1 終わり。 #2 へ続く。） 10

クローンヤクザが良い感じに雑魚で好きです。でもヤモト＝サンのほうがもーっとフィーヒヒヒ！

## 12　あとがき

- はい。これ書いたのは例のが Most 入りしたからです。その通りです。
- 例のは単純な改変だったので楽に書けたが、実際自分で最初から書くとなるとなかなか大変であった。
- 固有名詞を使っていないのは著作権とかが怖いので。商業目的ではないので平気と思うが念のため…
- 当初書きたかった内容は Web+ ソーシャル + 割愛された「Apple の製品だったら」の混ざったような内容だったが、文量が確保出来ず、急遽このような鉄拳風の内容になってしまった。
- 再認識したが箇条書きメソッドは実際 Twitter ぽくて楽だなと。長文は前後関係が必要だし面倒なので。
- 一応、文章的には「通して読むと全体で少しずつ内容が繋がっているような、繋がっていないような…？」という感じを目指して書いたが、書いた順番は相当乱れ、後からどんどん追記していく感じで書きました。
- ちなみにこれ書く前の記事の予定は「「最短期間で覚える○○」系で実際に極小期間で覚えて色々な問題を解いてみよう！」みたいなもので、初っ端から Lisp を扱いきれなくて消滅、黙祷…。
- 書いてて一番楽しかったのは NJSLYR 編。#2 が出るかは不明。でも一応最後までの流れの案はある。
- 書き始めたのが締め切り 6 日後、書き上げてるのが締め切り 7 日後の午前 7 時。実際眠い。教訓：早めに記事を書こう。
- どうでも良い事ですが、原稿の TEX ファイルが 256 行でとっても素敵な感じになりました。

# 私の愛したらーめんまぜそばがっつん

alstamber

**はじめに**

読者の方々は「らーめんまぜそばがっつん」(以下がっつんと呼ぶ) というラーメン屋をご存知であろうか。調布駅東口近辺にあり今年の 10 月 30 日をもって惜しまれつつも閉店したラーメン屋である。がっつんは多くの電気通信大学生の胃袋を満たし、多くの電気通信大学生の体重を増加させた。私はそんながっつんを愛したと自負している人間の一人である。がっつんがこの地球上から失われてしまった今、がっつんを愛した者としてがっつんに対する思いを綴らねばならぬと思い、筆を執る次第である。

## 1　私とがっつんの出逢い

私ががっつんに出会ったのは MMA に入部した直後のことであった。つまり昨年の春のこととなる。もはや 1 年半も前のこととなるので詳細は覚えていないが、aru に連れて行かれたことは記憶している。調布駅東口の近くにある調布百店街に入り、キャバクラの並び立つ細い路地をくぐると、そこにそれは存在していた。その建物は異様な獣臭い空気を放出していた。当時の私がおよそラーメン屋と呼ばれる施設において体験したことのない臭気であった。しかし店の前には列が形成されており席も満席であったことから、このラーメン屋と称する施設が人気を誇っていることは容易に察することができた。

とりあえず列に並び待った。まだ肌寒い季節であったように思う。しばらく並んでいるとあれほど強烈だと感じた獣の香りには馴れてしまった。漸く食券販売機の前にたどり着き、食券を購入する番となった。販売機には「らーめん」というおおよそどのようなものが供されるのか想像のつくボタンもあれば、「まぜそば」という当時の私にとっては未知の食物を示すボタンも存在していた。とりあえず初見の場においては冒険はするまいと決めていた私は、やや逡巡した上で小銭を突っ込み「らーめん」のボタンを押した。

その後少し時間が経過して、席に案内される段となった。やや腰掛けるには高い椅子に座り、注文したものを待つ。私は未知の食物との邂逅を目の前にして胸をこれまでにないほどに高鳴らせていた。あれほど人気なのである、きっと素晴らしい出会いに違いないだろう、と私は期待していたのである。

やがて店員の女性が私に声をかけた。「ニンニク入れますか？」……ニンニクと来たか。ここは噂に聞くラーメン二郎の系列の店であったのか。少し考えて最初は入れない味を楽しむべきだろうと判断し、申し出を断った。その少し後、突如目の前に巨大な塊が出現した。それは私の想像を絶した食物である。いや、これは食物なのだろうか。私の知っている食べ物という概念を大幅にはみ出した物体がそこに屹立していた。まず眼に入るのはひたすらに高いモヤシとキャベツのタワーであった。スープや麺というラーメンにあるはずのファクターはひと目では確認できない。ヤサイタワーの麓には丼との僅かな隙間に巨大な肉塊が押し込まれていた。チャーシューと呼ぶには不自然なものである。とにかく眼を見張るほど大きく、分厚く、ホモサピエンスの中に眠る食物に対する根源的欲求を刺激する存在であった。

目の前のこのあまりに大きなモンスターを今から攻略せねばならぬのだ。ほんの少し前までは予想だにしていなかった壁が今立ちはだかっている。まずこのヤサイによって形成されたバベルの塔を攻略しようと決意し、ヤサイを箸で崩しながら、口へと運ぶ。しかし、口に運べど運べどヤサイは減らないのである。まったくもって恐ろしい奴と遭遇してしまった。漸くヤサイの減少を目で感じることができるようになった段階で、麺とスープの存在を確認することができた。ああ、これはちゃんとラーメンだった。麺とスープが存在する料理であった。その麺も非常に太くその量と相まって出現した瞬間に圧倒的な存在感を示していた。麺を口に運ぶ。強烈な歯ごたえだ。私の貧弱な顎では噛み切るのにやや苦労する。大きく口を開けて麺を口に押し込み、もぐもぐと噛む。洪水のように押し寄せる小麦感。そして麺がスープの味をしっかり捉えて離していないことにも気づく。スープの存在感も一級品であった。鮮烈という言葉が相応しいといえるその味、多量の脂と飽和しきった旨み成分がこのスープがジャンクフードのファクターであることをこれでもかというほどに証明している。舌がグルタミン酸ナトリウムの存在を頻りに知らせてくる。嗚呼、これはグルタミン酸ナトリウムの海だ。今私はグルタミン酸ナトリウムの海に溺れているのだ。陶酔感のようなものを感じながら、私は麺を食べ進めた。

しかしあと少しというところで、突如感覚への変化が訪れた。快楽の海に突如現れたのは苦痛であった。少食の私の胃は一度にこれだけの食物を受け入れた経験が殆ど無かったのである。胃はびっくりして食物の受け入れを拒否し始めたのだ。なんとか水を使って麺を体の中に入れていく。なんとか寸前のところで私の精神は胃に勝利した。完食したのである。私は這々の体で店

から出た。aru はすでに完食していたようで、外で待っていた。家路についたが、胃の猛烈な膨満感との闘いであった。帰宅してからもしばらくはまともに動けなかったことは今でも色濃い記憶として残っている。

このような具合に、私のがっつんとの出逢いは概ね最高のものではなかった。私の中のがっつんは「とにかく量が多くて辛いもの」となってしまったのである。

## 2　変革する意識

そんな出逢いであったため、私はそのあと暫くがっつんに行く事はなかった。aru はその後も足繁くがっつんに通っていたようだが、私は「よくあんなところに通うなあ」と思っていた。しかし、これはどのようなきっかけであったのか全く覚えていないのだが、再びがっつんに行く機会があったのである。ちょうど初めてがっつんに行った 3ヶ月後ほどのことであったと思う。何を食べたかもあまり覚えてはいないが、やはり同じ「らーめん」であったのではないかと思う。

一つ確かに覚えているのは、がっつんを「食べる」という行為に対して意識を振り分けられるようになったということだろう。初めてがっつんを食べたときは、兎角ヤサイや麺を体に押し込んでいるという状態だった。2 度めのがっつんは、やはり胃が苦痛をアピールしていることに変わりはないものの、ヤサイ、麺、豚、スープの味についてもう少し意識できるようになっていた。それとともにがっつんが持つ他のラーメンにはない鮮烈な味に対する興味が頭を擡げてきたのである。

私の中にこの短い人生の中であってもラーメンという料理に対するイメージは一応形成されていた。がっつんはそのイメージをことごとく蹂躙していったのである。「お前の知っているラーメンなど、見識が狭い！」と言わんばかりであった。がっつんの味は決して上品なものではない。どちらかと言えば粗雑であるし、綺麗にまとまっているとも言えない。しかしグルタミン酸ナトリウムと動物性脂肪が集中砲火のように舌を刺激してくるこの感覚は私の中に眠っていたホモサピエンスの原始的欲求をかつてないほどに呼び覚ましてくるのだ。動物としての本能がこの食べ物を旨いと叫んでいた。かつて誰かが言っていた「体に悪いものほど旨い」という言葉に私はすこしだけ同意しかけていた。たしかにがっつんは脂と塩分にまみれた、寿命を伸ばすとは言いがたい料理である。しかしこの口の中に満ち溢れている刺激は快楽以外の何物でもなかった。私は徐々にがっつんというものに体を預けるようになっていったのである。

## 3　がっつんのある生活

それ以来月に一度はがっつんに行く生活が始まった。最初は拒否反応を示していた胃もやがてがっつんに飼い慣らされていった。暫くがっつんでは「らーめん」しか頼まなかったのであるが、あるとき出来心で「まぜそば」なるものを注文することもあった。これは汁なしラーメン (俗にいう油そばのようなもの) の類で、汁がない分タレの味が濃く、かつ具だくさんというものである。「らーめん」の味にすら鮮烈な印象を覚えた私に対して、「まぜそば」はあまりに苛烈だった。一口目が舌に触れた瞬間、体中に電撃のようなものが走った感覚を覚えた。強烈な味の濃さである。がっつんのスープをそのまま濃縮した味。これまでにない脂とグルタミン酸ナトリウムの濃度に体中が吃驚した。あまりに強い旨みであったため、食べ進めるうちに舌が悲鳴を上げ始めた。人間の体というのはあまりに強い快楽を受けると、逆に苦痛になる。私はそれを食物でもって体感することとなったのである。まぜそばは汁が少ない分、麺や具といった固形物の量が多く、まぜそばを食べきるのは至難の業であった。多量の麺、強烈なタレ、そしてそれをかろうじてまとめあげているヤサイ。ジャンクフードの極みである。決して体には良くない。決して高級感ある食品であるわけでもない。しかし、強烈な快楽に支配された私にとって、それは至上の料理だったのである。

夏か秋ごろだっただろうか、新しいメニューが登場した。「中華そばクラシック」というものである。この料理は私にがっつんの新境地を見せてくれた。がっつんの特徴である強烈な旨味成分はそのままに、魚介の風味を生かしたものとなっていた。がっつんのスープといえば、とにかく動物性の脂が主体で、陸上生物の香りに満ち溢れていたが、ここにきて魚のテイストを取り入れたことで、がっつんの新たな可能性を示すことになった。量もそれほど多くなかったため、お腹がそれほど空いてない時にも食べられる嬉しいメニューだったことを記憶している。

## 4　さようなら、がっつん

それからも一時は毎週行く事もあるぐらいに私はがっつんの魅力に取り憑かれ、がっつんを愛するようになった。生活にがっつんがあることが当然になっていたし、また幸せを感じていた。今年度になって忙しくなったことと金銭的な都合で行く頻度は減ったが、それでもがっつんに行く事はやめなかった。

しかし出逢いと別れは何時の時代も突然である。がっつんが 10 月 30 日をもって閉店することが決定したのである。最初その話を聞いたのは Twitter であった。私はにわかにそれを信じられず、Twitter にありがちなデマだろうと思っていた。しかしがっつんから正式に告知があり、それがデマでもなんでもないことを知るのはそれからすぐのことだった。私の心に灰色の悲しみが発露した。あまりに突然のことであったので、私は不自然なほどに冷静であった。嗚呼、もうしばらくすればあの味は食べられなくなるのか。確かに辛い。残念極まりない。しかし私の心は穏やかであった。あまりに突然だったからだろうか。どんな店であれいつかはこうなると思っていたからだろうか。自分の精神でありながらはっきりとしたことを言えないのが恥ずかしいが、兎角私はがっつんの閉店の報せをかようにして受け入れたのである。

それからはがっつんに機会があれば行くようになった。私以外にもやはり同じ事を考えている人はいたようで、がっつんの前にできる行列は日に日に長くなっていった。今まで食べることがなかった「魚まぜそば」なども試した。すべてのメニューを食べきらなければ、多分後悔が残るだろうと思ったのである。閉店が近づいてもがっつんはがっつんだった。らーめんもまぜそばも中華そばクラシックも私にがっつんのラーメンを食べているという充実感を十二分に与えてくれた。スープの味が変わったり、豚が変わったこともあったが、がっつんの根本的な特徴は失われることがなかった。

私は閉店の日にがっつんに行く事を考えていたが、その日にどうしても外すことのできない予定が入ってしまい、私の希望は脆くも砕かれてしまった。閉店前にがっつんにいける最後の日は 10 月の 26 日となってしまった。私は「まぜそば」を注文した。「らーめん」にすることも考えたのだが、やはり一番がっつんらしさがあるメニューにしようと思ったのである。最後の「まぜそば」もやはり「まぜそば」だった。強烈な旨みが私の舌を刺激してくる。これがまぜそばだ。鮮烈で容赦がない。これが私の愛した味なのである。私は最後の一口を少し名残惜しく、しかし感謝の気持ちを湛えて口に運んだ。丼を上げて「ごちそうさま」の一言。店を出る。もうここに来ることはないのだ、と思うとここに来て漸く寂しさがこみ上げてきた。感謝と悲しさと寂しさと、形容しがたい感情がない混ぜになったまま私は店を後にし、家路についた。

私ががっつんという場所を知って通っていたのは高々1 年半ほどのことで、その程度の人間が色々と勿体ぶった語りをしても良いのだろうか、と少々思っていた。しかし MMA 部員の多くから「がっつんのことを書けるのは今しかない」と言われ、筆を執った次第である。私ががっつんというものを愛していたのは事実だったと自負しているし、今でもなにか大切な物を失ったような、丁度恋人と別れた後のような心境のままである。今後新しいラーメン屋が調布の地にできることもあろう。それでも私はがっつんというラーメン屋がこの地にあったことを忘れないであろう。一度愛した人のことは死ぬまで忘れられないものなのである。

ありがとう、がっつん。さようなら、がっつん。

# アンケート

**はじめに**

世の中には様々な宗教の対立があります。それは Windows 教 vs UNIX 教であったり、Emacs 教 vs Vi 教であったりします。MMA 部員を対象にアンケートをとってみました。皆さんはどのような環境を使っているのでしょうか。

## 1 好きなもの・嫌いなもの・強いられているもの

### 好きな OS

- Windows 7: 7 票
- MacOS: 5 票
- Ubuntu: 4 票
- Debian: 3 票
- Windows XP: 3 票
- FreeBSD: 3 票
- Android(4.0 以降): 3 票
- Arch Linux: 2 票
- Fedora: 2 票
- GNU/Linux: 2 票
- iOS: 2 票
- Gentoo Linux
- Windows 2000
- Vine Linux
- CentOS
- Windows Phone 7
- Windows(〜7)
- Windows
- Emacs
- LinuxMint
- MS-DOS

### 嫌いな OS

- Windows 8: 3 票
- Windows Vista: 2 票
- Solaris: 2 票
- Android(4.0 より前): 2 票
- Windows Me
- Mac OS 9 以前
- お金のかかるヤツ

### 強いられている OS

- Solaris: 2 票
- Windows: 2 票
- Windows XP
- Vine Linux
- MacOS X
- Ubuntu
- Fedora

### 嫌いな CPU

- ヒートスプレッダ付きの Socket 式 CPU
- Atom
- Bulldozer
- Pentium 4
- x64 じゃないやつ
- 消費電力が高い

### 強いられている CPU

- x64
- 最近の Intel 製
- Pentium 4

### 好きな CPU

- ARM
- Socket 370
- Socket A
- 最新のゲームがしっかり動けば
- intelCPU
- K10 世代までの AMDCPU
- Intel Pentium4 HT 630
- 最近の Pentium
- Intel Core Processor Family
- intel i486DX (特に意味は無い)
- 消費電力が低い

### 好きなエディタ

- vim: 14 票
- Emacs: 9 票
- Eclipse: 4 票
- terapad: 4 票
- notepad: 2 票
- gedit: 2 票
- Nano: 2 票
- Notepad++: 2 票
- Kwrite
- edit
- edlin
- mifes
- 秀丸
- ed
- xyzzy
- Editra
- Visual Studio
- CotEditor
- Coda
- Sakura Editor

### 強いられている言語

- C: 7 票
- 英語: 5 票
- ドイツ語: 3 票
- 日本語: 2 票
- Java: 2 票
- C++: 2 票
- PHP: 2 票
- 第二外国語

### 強いられているエディタ

- Emacs
- Eclipse
- Gmail
- AOJ

### 好きな言語

- 日本語: 10 票
- C#: 3 票
- Python: 3 票
- Ruby: 3 票
- Java: 3 票
- C: 2 票
- Bourne Shell: 2 票
- JavaScript: 2 票
- VisualBasic: 2 票
- 英語 2 票
- ドイツ語
- Perl
- C++
- Whitespace
- ヒュムノス
- bc
- 中国語
- LaTeX
- LISP
- 忍殺語
- 各種ネットスラング
- Haskell
- Brainfuck
- R

### 嫌いなエディタ

- メモ帳: 3 票
- Wordpad: 3 票
- Emacs
- Dreamweaver
- vim
- Emacs キーバインド風の別の何か（例：C-h でヘルプが表示される開発環境）
- Emacs

### 嫌いな言語

- Ruby: 2 票
- PHP: 2 票
- Visual Basic
- Python
- 英語
- Java
- AHK
- 変数宣言や標準出力が面倒な言語
- 電通病患者の誤字脱字漢字間違いや日本語の誤用溢れる謎言語
- C
- JavaScript
- フランス語

## 好きなソフトウェア

- Eclipse: 2 票
- Visual Studio: 2 票
- Opera: 2 票
- rogue
- simutrans
- elona
- ゲーム (特に FPS、ADV)
- タイル型ウィンドウマネージャ
- BSD-style init
- ACDSee
- 駅すぱあと
- Chrome
- Total Terminal
- エロゲはソフトウェアですか
- iTerm2
- Sparrow
- Cygwin
- Photoshop
- Illustrator
- マウスのない環境を想定されているもの全般
- SourceEngine
- Minecraft
- x264
- gnome-terminal

## 嫌いなソフトウェア

- 調布祭アプリ: 3 票
- Internet Explorer: 3 票
- ウォークマンに転送する奴
- Dreamspark のダウンローダー
- マウスの移動量について考えられていないもの
- やたら点滅するもの
- ”無料で高機能のツールバーをインストールする”がスクロールしないと見えないところにあるインストーラ
- Games for Windows Live

## 強いられているソフトウェア

- X アプリ: 2 票
- マップ (iOS6)
- cmd.exe
- Firefox
- Chrome
- systemd
- iTunes
- Acrobat X
- Adobe 系
- ATOK
- Origin

## 嫌いな Twitter クライアント

- 公式: 2 票
- TheWorld
- Tweet this
- Tweet button
- Web
- Twitter for(Android|iPhone)
- [0-9]+favs
- twittbot その他定型句投稿 bot

## 強いられている Twitter クライアント

- twicca: 3 票
- PSPったー
- Web ブラウザ
- Tweetbot
- yubitter

## 好きな Twitter クライアント

- ShootingStar: 7 票
- Krile2: 4 票
- twicca: 4 票
- mikutter: 3 票
- Janetter: 3 票
- Tween: 3 票
- Saezuri: 2 票
- TweetDeck: 2 票
- Web: 2 票
- 公式: 2 票
- 自作
- twil2
- TheWorld
- TweetATOK
- OpenTween
- Tweetie
- Azurea
- Janetter
- ラーメン大陸
- 夜フクロウ
- Tweetbot
- TweetATOK
- 夜狐四重奏
- turpial

## 好きなハードウェア

- SSD: 5 票
- Magic Trackpad: 2 票
- US 配列キーボード
- Let'snote
- Happy Hacking Keyboard(JIS 配列)
- 人差し指・中指トラックボール
- ER-4S
- 入力が軽いキーボード
- 水冷クーラ
- 大容量の HDD
- walkman
- MacBookPro
- Apple Wireless Keyboard
- RTX810
- iPod(初代) or iPod nano(初代 or 6th gen)
- 10k[mAh] で 2k[円] の価格帯のモバブ各種
- Victor 製イヤホン
- GeForce
- メカニカルキーボード (JIS 配列)
- (ゲーミング) マウス
- corei 第二世代
- トラックポイント
- 70-80 年代の製品 (主にオーディオ)
- デジタルガジェット
- QWERTY 搭載スマホ
- 無線機器
- Think Pad
- 液晶タブレット
- TabletPC

## 強いられているハードウェア

- ソフトウェアキーボード: 2 票
- JED のキーボード
- JED のマウス
- JED 端末
- JED にあるもの全部
- イーサネットコンバータ
- 計算機室のパソコン
- 小さいキーボード
- Arduino

## 嫌いなハードウェア

- 情基端末
- MacBook Pro のキーボード
- BUFFALO 製品
- Kobo
- JED のキーボード
- 縦長のキーがあるキーボード
- 左下の配列が変なキーボード
- スペースキーが中途半端な長さのキーボード（C と V の間-N の中間がベスト）
- 4Core 化できない AMD 製 cpu
- タブレット

## 2　持っているものについて

### 計算機の数

- 8 台: 1 票
- 7 台: 2 票
- 5 台: 4 票
- 4 台: 1 票
- 3 台: 3 票
- 2 台: 8 票
- 1 台: 2 票

### 携帯端末の数

- 30 台以上: 1 票
- 16 台: 1 票
- 10 台: 1 票
- 8 台: 1 票
- 7 台: 2 票
- 4 台: 2 票
- 3 台: 2 票
- 2 台: 5 票
- 1 台: 5 票
- 0 台: 1 票

### 自宅サーバあるいは VPS の有無

- VPS 10 つ
- 自宅サーバ 7 つ
- なし: 12 票

### メインの端末・回線

- iPhone4S: 3 票
- iPhone5: 3 票
- GALAXY S2 LTE: 2 票
- EVO3D: 2 票
- GALAXY S III
- EVO
- S005
- iPhone4
- SC-04D
- IS12S
- SC-01D
- Galaxy Note
- T-01C
- SH-11C
- SH007
- HTC J
- Nexus7 16GB
- DIGNO
- SH906i

### 使っている回線

- docomo 7
- au 13
- Softbank 3
- モバイルルネッサンス 1

### 自宅環境の画面解像度の合計

- 1920x1080 + 1024x768: 3 票
- 1280x1024: 2 票
- 1920x1070: 1 票
- 1920x1080 + 1440x900: 1 票
- 1920x1080 + 1920x1080 + 1280x1024: 1 票
- 1920x1080 + 1920x1080 + 1680x1050: 1 票
- 1920x1080 + 1024x600: 1 票
- 2880x1800: 1 票
- 1,440x900 + 1,366x768: 1 票
- 1920x1080 + 1440x900 + 1440x900 + 1024x768: 1 票
- 1440x900: 1 票
- 1920x1080 + 1680x1050 + 1280x800: 1 票
- 1980x1024 + 1980x1024 + 1024x800: 1 票
- 1920x1080 + 1920x1080: 1 票
- 1366x768 + 1024x600: 1 票

## 3　きのこ or たけのこ or ブラックなんとか

- 中立: 2 票
- たけのこ: 3 票
- ブラックなんとか

## 4　一言

- ニャル子かわいい
- VPS ほしい
- Opera は素晴らしいブラウザです.
- PSP を携帯端末代わりに使うのにはちょっと無理が出てきました
- 家にあるルータの数: 8 台
- もう一度 AMD を信じてもいいかなって
- PDF を快適に読める端末がほしい今日このごろ
- 回線運用、キャッシュフロー管理が辛い。Google 万歳！
- MacBook Air とタブレット端末が欲しいです
- MMA 部員らしからぬ姿勢反省しない
- 新 iMac とラックサーバと 19 インチラックと PS Vita と Corei7 機が欲しいです。
- あ
- 敵: Radeon
- 零えぼフリーズ修正アップデートはよ
- どうぶつの森と 3DS ほしい
- たけのこ派, ブラックサンダー販売業者です
- メイン機のストレージサイズが 10.5TB になりました
- 最近コールサインとりました JH1LVO
- ラーメンとりんごジュースが好きです。
- 今年は作曲家クロード・ドビュッシーの生誕から 150 年のアニバーサリーイヤーです。

# 百萬石 2012 年 秋号 ©電気通信大学 MMA

2012 年 11 月 23 日　初版第一刷発行　【本書の無断転載を禁ず】

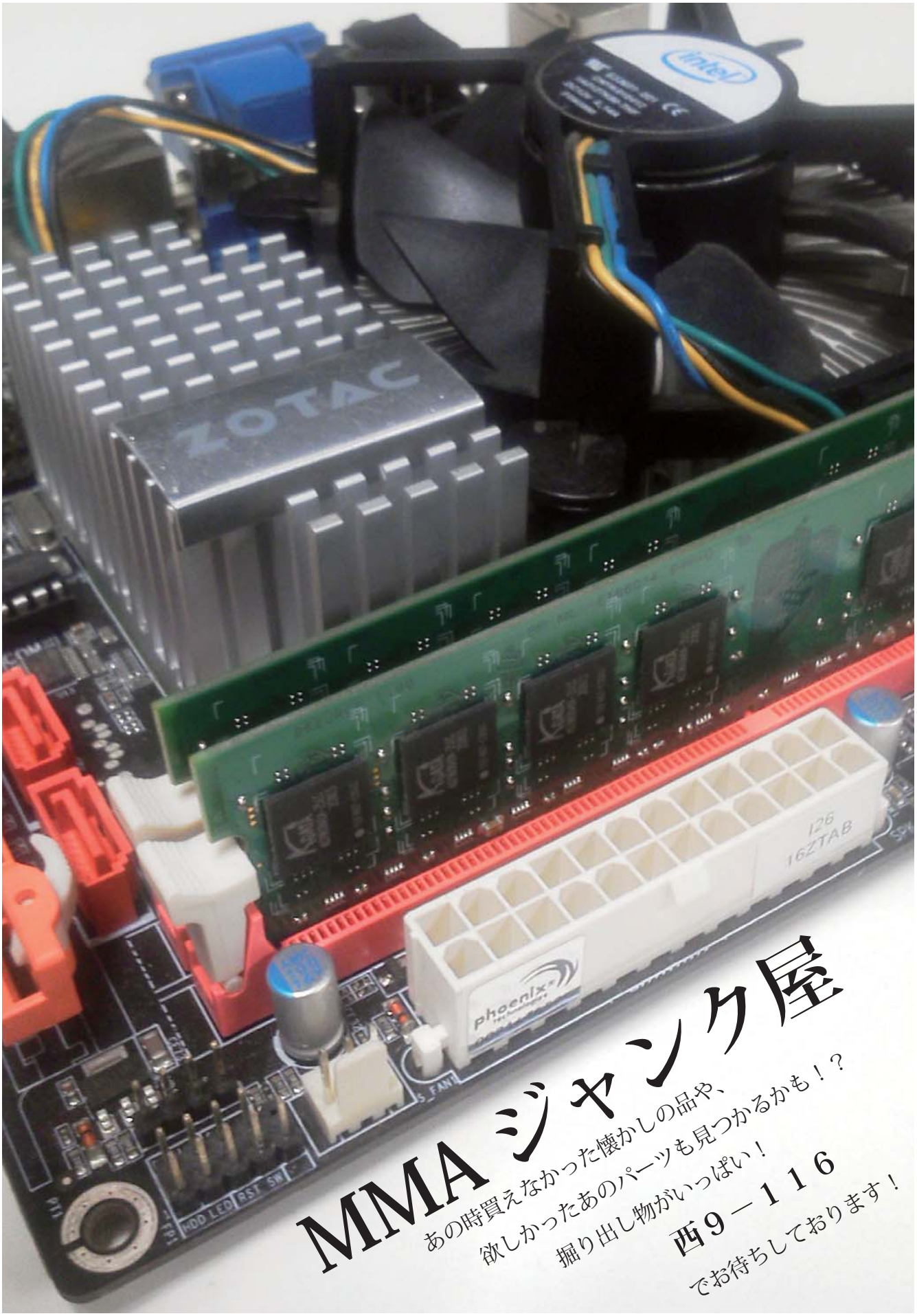
著　者　alphakaz, alstamber, clear, hirosun, hiyakashi, iz, kakakaya, kzm, KHe7, k_operafan, mernao, mznh, saharakei, zico
編集者　clear
発行者　電気通信大学 MMA
発行所　電気通信大学 MMA 部室
〒182−8585 東京都調布市調布ヶ丘 1−5−1 サークル会館 208 号室
http://www.mma.club.uec.ac.jp/
印刷所　電気通信大学 MMA 部室
製本所　電気通信大学 西 9 号館 116 教室
表　紙　電気通信大学 MMA

# 季刊百萬石

## 2012秋号

平成24年11月23日発行
(毎年春・秋発行)通巻38号
発行人：MMA
編集人：clear
表紙：hiro1357
表紙写真：alstamber, mernao

発行：電気通信大学 MMA
〒182-8585
東京都調布市調布ヶ丘1-5-1
サークル会館2階




# MMAジャンク屋

あの時買えなかった懐かしの品や、
欲しかったあのパーツも見つかるかも！？
掘り出し物がいっぱい！

**西9－116**

でお待ちしております！

http://wiki.mma.club.uec.ac.jp/

Printed in Japan　MMA 印刷


特別定価 **0円**

MMA