Information technology researching society
of the Gunmer, by the Gunmer, for the Gunmer

Vol. 02
Comic Market 88 版

群馬大学電子計算機研究会

Lollipop

IGGG
http://www.iggg.org/

群馬大学電子計算機研究会　著

群馬大学電子計算機研究会 IGGG "Lollipop" Vol. 02 Comic Market 88 版

目次

Ctf のための ruby 入門

ひげ

概要

最近，Python と CTF を始めた．友人の勧めで “Ctf のための python 入門”[6] というスライドを読んだ．そこで生まれる，“Ruby ではだめなのだろうか？” という必然的な疑問[1]．ということで，まとめた．ちなみに，Ruby は 1 年ぐらい前から勉強している．

また，比較もかねて，“Ctf のための python 入門” の内容に準拠している．要するにパロディである．勝手にパロディしてしまい申し訳ない．自分は Python も好きです．

1 CTF と Ruby

1.1 CTF とは

CTF とは Capture the Flag の略称で，コンピュータセキュリティの技術を競うコンテストの総称である．CTF で良い成績を得るためにはコンピュータに関する，幅広い知識が必要である．

1.2 Ruby とは

Ruby は動的でかつ純粋なオブジェクト指向プログラミング言語である [5]．日本人が開発した言語ではじめて国際規格となった言語として有名である．特徴的な機能としては，オープンクラスや Proc クラス，強力なメタプログラミングなどが挙げられる．

1.3 なぜ Ruby なのか

CTF では良く以下のような処理を行う．

- 数値と文字の相互変換
- 巨大整数演算
- エンコード・デコード
- ハッシュ関数

Ruby なら，これらの処理を楽に行うことが出来る[2]．

[1] Python と Ruby は両方とも有名なスクリプト言語で，ライバルのような関係である．

[2] もちろん Python でもできる

2 数値と文字列

ここから前述した処理について Ruby ではどうやるのかを紹介する．Ruby は irb という対話型処理系があり，それで逐次実行して動作を確認できる[3]．

2.1 電卓として使う

Ruby も他の言語同様，基本的な演算を行える．

```
1 + 2 * 3 -10 / 3    #=> 4
1 << 16              #=> 65536
5 ** 1024            #=> 556268464626...12890625
0xca ^ 0xfe          #=> 52
```

上から順に，四則演算，ビットシフト，べき乗，XOR である．べき乗の結果は非常に大きな整数値となっており，C 言語での `long long int` 型ですら入り切らない．しかし，Ruby は `Bignum` クラスで長整数型を表現しており（`Bignum` クラスの上限はメモリサイズ），巨大な整数演算を簡単に行うことが出来る．

2.2 10・16・8・2 進数

CTF では暗号の解読や，バイナリデータの解析などを行うことが良くあり，その際に様々な基数の数値や文字を相互変換する必要がある．

```
# 10進数からその他
123.to_s      #=> "123"
123.to_s(16)  #=> "7b"
123.to_s(8)   #=> "173"
123.to_s(2)   #=> "1111011"

# その他から10進数
'123'.to_i         #=> 123
'7b'.to_i(16)      #=> 123
'173'.to_i(8)      #=> 123
'1111011'.to_i(2) #=> 123

# フォーマット指定
'%d' % 123                  #=> "123"
'%8d, %08d' % [123, 123] #=> " 123, 00000123"
0x7b                        #=> 123
```

フォーマットを指定する際に，d, x, o, b の順で 10, 16, 8, 2 進数を表している

[3] 実は irb は eval を使って Ruby 自身で実装されたプログラムである．組込みで対話型の処理系が実装されている Python との相違点の 1 つである [3].

2.3 文字列処理

CTF では数値と文字との相互変換と同様に文字列処理も良く使う（数値を文字列に変換して利用できる）．

```
# 数値とASCII文字
'A'.ord  #=> 65
65.chr   #=> "A"

# 文字列の分割・連結と配列（リスト）
'abc'.split('')                  #=> ["a", "b", "c"]
'abc'.split('').map{|c| c.ord} #=> [97, 98, 99]
['a', 'b', 'c'].join             #=> "abc"
['a', 'b', 'c'].join(':')        #=> "a:b:c"

# 部分文字列
'abcdefg'[0]        #=> "a"
'abcdefg'[-1]       #=> "g"
'abcdefg'[1, 3]     #=> "bcd"
'abcdefg'[3..-1]    #=> "defg"
'abcdefg'[3...-1]   #=> "def"
```

Ruby の部分文字列の取り出しはかなり特殊で，カンマ区切りで始点と長さを指定でき，ドットで範囲指定できる．ただし，ドットが 2 つのときは末尾が入り，ドットが 3 つのときは入らない．また，上記の他に文字列を直接入れたり，正規表現指定できたりする．

2.4 CTF 過去問

今迄の知識を使って一つ CTF の問題を解いてみる．以下は CSAW CTF 2011 の Crypto3 という問題である [1].

リスト 1: CSAW CTF 2011：Crypto3

```
01001100001100000101110011011101000010000001110111011001001011
00101011010110111001100100000011011010110001010110010110111101
00011010010110111001100111001000000111011101100001011100011 0
01000000110000100100000011001110111001001100101011000010111
01000010000001110011011101010110001101100011011001010111001
10111001100101110001000000101011101100101001000000111001101
10010101100101011011010010000001110100011011110010000001100
01001100101001000000110011101100101011011100110010101110010
01100001011101000110100101101110011001110001000000110000100 1
00000011011000111011101000010000001101111011001100010000
00011000100111010101111010011110100010000001100001011000100
11011110111
```

Crypto3 はこれを復号する問題である [4]．この問題，実は英文を 2 進数に直したものなのである．そのため，まずは，8 ビットずつに分割する必要がある．split だと難しいので，正規表現が利用できる scan メソッドを利用する．以下が解答である．

リスト 2: Crypto3 の解答

```
#textの中に問題の数列が文字列として入っている
text.scan(/.{1,8}/).map{|c| c.to_i(2).chr}.join
#=> "Last weeks meeting was a great success. We seem to be
     generating a lot of buzz about the movement. The key
     for next weeks meeting is resistance. If there is
     anyone else you know of that may be interested in
     joining bring them to the meeting this week. It will be
      held same time, same place."
```

scan の中の正規表現は “任意文字が 1 以上 8 以下” を意味している．要するに 8 文字ずつ切り出して，余った分はそのまま取り出すという意味になる．

[4] 自分だけかもしれないが，irb ではこのような大量の入力をコピペすることが出来ない．そういう時は，コピペしたい値を返すメソッドを別ファイルで書いて load すると良い．

3 エンコード・デコード

この章は Ruby での様々なエンコード方法やハッシュ値の生成方法を紹介する．CTF では暗号やバイナリの問題ではもちろんのこと，通信系の問題でもこれらの処理をよく利用する．

3.1 文字列変換

CTF では文字列を ASCII コードの数値に変換したり，数値を ASCII コードに従って文字列に変換したり，したいときが良くある（例えばさっきのような問題）．さっきの問題では，map, to_i, chr なんかを駆使して変換したが，一発で変換できるメソッドがある．

```
"abc".unpack('H*')      #=> ["616263"]
["616263"].pack('H*')   #=> "abc"
"abc".unpack('B*')      #=> ["011000010110001001100011"]
["011000010110001001100011"].pack('B*') #=> "abc"
"abc".unpack('C*')      #=> [97, 98, 99]
[97, 98, 99].pack('C*') #=> "abc"
```

メソッドの引数に H や B のようなオプションを指定することで，様々な変換が行える．*（アスタリスク）は何文字分変換するかを指定していて，*は全てを意味する．このほかにもオプションはある．ちなみに，pack メソッドはリ Array クラスのメンバなので，[] で囲む必要がある．

3.2 ROT13

ROT13 とは換字式暗号の 1 つである．英字を大文字小文字わけて，英字の個数のちょうど半分である 13 個ずらす．そうすることで，エンコードとデコードが同じアルゴリズムになる．もちろん，暗号としての強さはないが，パッと見の意味は分からなくなるので，クイズの答えなどに利用するらしい．

Ruby には ROT13 を行ってくれるメソッドはない．しかし，Ruby はオープンクラスのため，以下のように組み込むことが出来る [5]．

```
class String
  def rot13()
    return self.tr('A-Ma-mN-Zn-z', 'N-Zn-zA-Ma-m')
  end
end
```

また，任意長の ROT メソッドを作るには，以下のようにすればよい．

```
class String
  def rot(n)
    return self.split('')
      .map{|c| /[A-Z]/=~ c ?
        ('A'.ord + (c.ord - 'A'.ord + n) % 26).chr : c}
      .map{|c| /[a-z]/=~ c ?
        ('a'.ord + (c.ord - 'a'.ord + n) % 26).chr : c}
      .join
  end
end
```

[5] もちろん，変換部分自体は 1 行で書けるのでそのまま実行してもよい．

3.3 Base64

Base64 エンコードとは，印字可能な 64 文字（A-z, 0-9, +, /）へのエンコード方式（上記の記号に加えてパディングとして = も使われる）である．印字可能文字に変換できることから，それ以外の文字を扱うことのできない通信方式でバイナリデータなどを送受信する際に良く利用される．

Ruby には，初めから Base64 を行ってくれるメソッドが存在するのでそれを利用する．

```
['abc'].pack('m')      #=> "YWJj\n"
"YWJj\n".unpack('m')   #=> ["abc"]
"YWJj".unpack('m')     #=> ["abc"]
```

3.4 uuencode

uuencode とは，バイナリ・テキストデータ間のエンコード方式の一つで，電子メールの添付データなどでよく使われた．以下のような特徴的なフォーマットになっている

```
begin <mode> <pathname>  # modeは3桁の数字
#86)C                    # ここにエンコードした文字列を記述
                         # 実はここに空白1つある
end
```

Ruby には Base64 同様すでに用意されているので，それを利用する．

```
['abc'].pack('u')      #=> "#86)C\n"
'#86)C\n'.unpack('u')  #=> ["abc"]
```

3.5 ハッシュ値

ハッシュ値とはハッシュ関数によって出力された値のことで，ハッシュ関数とは様々な入力に対し，定数時間で対応する数値を出力する関数のことである [6]．定数時間で終わることから，データの探索や比較の高速化に用いられる．

代表的なハッシュ値には MD5 や SHA1 等があり，Ruby はそれらのハッシュ関数が初めから定義されている．

```
require 'digest'

Digest::MD5.hexdigest('abc')
  #=> "900150983cd24fb0d6963f7d28e17f72"
Digest::SHA1.hexdigest('abc')
  #=> "a9993e364706816aba3e25717850c26c9cd0d89d"
Digest::SHA256.hexdigest('abc')
  #=> "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410
      ff61f20015ad"
Digest::SHA512.hexdigest('abc')
  #=> "ddaf35a193617abacc417349ae20413112e6fa4e89a97ea20a9e
      eee64b55d39a2192992a274fc1a836ba3c23a3feebbd454d44236
      43ce80e2a9ac94fa54ca49f"
```

[6] 一般に入力が出力への一方向性を持っていたり，複数の入力が同じ出力になったりする．

3.6 CTF 過去問

前回同様に今迄の知識を利用して CTF の問題を解いてみる．今回は Web 上でいつでも解くことが出来る，KSNCTF から Onion という問題をとってきた [2]．Onion は以下のような文字列が記述されている（極めて長いので冒頭だけ）．

リスト 3: CSAW CTF 2011：Crypto3

```
EVm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSV01WbDNXa1JTV0ZKdGVGWl
ZNakExVmpBeFYySkVUbGhoTWsweFZtcEdZV015U2tWVQpiR2hvVFZWd1ZWW
nRjRWRUTWxKSVZtdFdVZ3BpVlZwWVZtMTRjMDB4V25GUmJVWlVUV3hLU1Za
dGRHdFhRWEJwVW01Q1VGZFhNSGhpCk1WWlhWMjVHVW1KVldtRldha0Y0VGx
aVmVXUkdaRmRWV0VKd1ZXcEtiMlJzV2tkWGJHUnJDazFXY0ZoV01qRZV3
hLV0ZWdFJs...
```

全て英数字で構成されているのがわかるはずだ．そのため，とりあえず，Base64 デコードをかけてみる．

```
# textに問題の文字列が入っている
text = text.unpack('m')
#=> ["Vm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSWFJteFZVMjA1VjAx
     V2JETlhhMk0xVmpGYWMySkVU\nbGhoTVVwVVZtcEdTMlJIVmtWUgp
     iVVpYVm14c00xWnFRbUZUTWxKSVZtdGtXQXBpUm5CUFdXMHhi\nMV
     ZXV25GUmJVWmFWakF4TlZVeWRGZFdVWEJwVWpKb2RsWkdXbGRrCk1
     WcFhWMjVTYWxKWFVtRldh\n..."]
```

似たような文字列が出てきた．良く見ると，一定の周期で改行文字（\n）が出てきているのがわかる．なので，これも Base64 デコードをしてみる．

結果は載せないが，また同じような文字列が出てくる．違いは徐々に短くなっていること．なので，可能な限り繰り返してみる．

16 回ほど繰り返したところで以下のような文字列が出てくる．

```
text = text[0].unpack('m')
#=> ["begin 666 <data>\n51DQ!1U]&94QG4#-3:4%797I74$AU\n \n
     end\n"]
```

明らかに uuencode である．あとはこれを uudecode してやると，フラグを得ることが出来る．

4 ファイル操作と通信

最後にバイナリファイルの読み書きや Socket, HTTP 通信の方法について紹介する．前章までに比べて，この章は簡単にする [7]．

4.1 バイナリファイルの入出力

CTF では，謎のバイナリファイルが渡されて，そのバイナリファイルをいろいろと変換することでフラグを得るような場合もある．その際に，16 進数表示で取得出来たり，16 進数を書き込めたり出来ると都合がよい．Ruby には，それらを容易に行えるメソッドが用意されている．

```
# 読み込み
binary = File.binread('Reverseit').unpack(‘H*’)
# 書き込み
File.binwrite('aaa', binary.pack('H*'))
```

[7] 自分はネットワーク系が苦手なため．

4.2 外部コマンドの実行

外部コマンドなんかが実行できたら便利だろう[8]．Ruby には外部コマンドを実行する方法が何種類か用意されている．

```
# 統一的で便利
require 'open3'
out, err, status = Open3.capture3('ls -l')

# 出力が戻り値で受け取れる
`ls -l`

# 戻り値が真偽値
system('ls ^^e2^^80^^93l')
```

他にもあるが，過去との互換性をとったり，C 言語のプログラムを移植したりするために使うそうなので，Open3 を使えばよいらしい．

4.3 Socket 通信

Ruby には Socket 通信用に socket ライブラリがある．詳しくは説明しないが，それを利用することで，TCP 通信などが容易に行える．

```
require 'socket'

solve = 'hoge'
s = TCPSocket.open('123.45.67.89', 1234)
s.puts(hoge)
s.close
```

4.4 HTTP 通信

Ruby には HTTP 通信用に HTTP クラスがある．詳しくは説明しないが，それを利用することで，様々な HTTP リクエストを簡単に送信することが出来る．

```
require 'net/http'

http = Net::HTTP.new('www.yahoo.co.jp')
response = http.head('/')
p response
```

4.5 CTF 過去問

最後に，この章でも CTF の問題を解いてみる．この章の問題は SECCON 2014 で出題された Reverse it という問題である [4]．問題では以下のようなバイナリデータが与えられた（とても大きいので一部分だけである）．

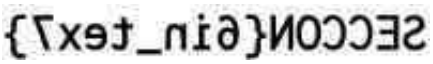
リスト 4: CSAW CTF 2011：Crypto3

```
9dff700db6dafc937263283222bdd218b425d47b8c00a1ab609ee707bb0
431567b5ad684bbd4b97e00a0ab6d36cd31c9a4d1012d5872a9b53acfc1
041b514700dd700814e9659377b431377696f81049a511cd308fd81c89c
8c46b449415f88bbc00a1ab26bc992614022d81e50b0eb453d69e930827
b6e670a0915ed7898b502bc3dcaad5850781047b5347c1af5d51aec2597
a7739b22fec2e21308e6...400101010006494 64a401000eff8dff
```

問題の名前やバイナリのヘッダ（末尾の FFD8FF は JPEG のヘッダ）から，16 進数で反転していることがわかる．なので，これを反転してバイナリファイルに出力してやればよい．

```
binary = File.binread('Reverseit').unpack('H*')
solve = binary[0].reverse
File.binwrite('aaa', [solve].pack('H*'))
```

出力した aaa ファイルの拡張子を jpg に変えて，見てみると以下のようになる．



後はこれを imagemagick 等で反転してあげればよい（しなくても頑張れば読めそうだが）．

5 おわりに

自分は Ruby も CTF も初めて日が浅いです．なので，雑な点や間違いなどがあるかもしれません．それでも，これをきっかけに Ruby や CTF（や Python）に興味を持っていただければ幸いです．最後まで読んでいただきありがとうございました．

参考文献

[1] CSAW CTF 2011. https://ctf.isis.poly.edu/static/archives/2011/challenges.php.html. (2015.7.17 閲覧).

[2] ksnctf - 5 Onion. http://ksnctf.sweetduet.info/problem/5. (2015.7.17 閲覧).

[3] Rubyist Magazine - Rubyist のための他言語探訪 【第 1 回】 Python. http://magazine.rubyist.net/?0008-Legwork. (2015.7.17 閲覧).

[4] write-ups-2014/seccon-ctf-2014/reverse-it at master ctfs/write-ups-2014 GitHub. https://github.com/ctfs/write-ups-2014/tree/master/seccon-ctf-2014/reverse-it. (2015.7.17 閲覧).

[5] オブジェクト指向スクリプト言語 Ruby. https://www.ruby-lang.org/ja/. (2015.7.17 閲覧).

[6] しらかみゅ@shiracamus. Ctf のための python 入門 - SlideShare. http://www.slideshare.net/shiracamus/ctfpython. (2015.7.17 閲覧).

[8] ちなみに，自分は CTF を始めたばかりのためか，必要になった経験はない．

コンピュータビジョンのためのSVMを用いた絶対領域検出

ism67ch

背景・概要

コンピュータビジョンとは，カメラで撮影した画像から，被写体となった対象世界がどうなっているのかを明らかにする問題を取り扱う学問・研究領域である．要約してしまうなら，ロボットに人間の眼と同様の機能を持たせる学問である．人間の眼であれば，見た空間にどのようなものがあり，その空間がどのような場所かまで認識することができる．ロボットという“未来人”が登場しつつある現代において，この研究は盛んに行われており，その活躍も期待されている．

ロボットを“未来人”と表現したことには，私なりの理由がある．ロボットの中でアンドロイドと呼ばれるものは，人間に寄せて作られている．日本テレビの人気番組『マツコとマツコ』の中で，アンドロイド研究の第一人者は「最初こそ抵抗あるが，人間よりもアンドロイドの方が接し易い場合もある」と語った．「アンドロイドはもはや人間より人間なのかもしれない」そのように感じたことこそが，“未来人”と表現する所以である．

同時に，私個人としては「再現するなら徹底的に」という気持ちも芽生えた．つまり，人間として作る限り，人間を徹底的に再現して欲しいのである．将来，コンピュータビジョンを用いて“未来人”が空間を認識するのであれば，人間の眼の癖も徹底的に再現して欲しい．空を飛ぶドローンがあれば，それに目がいくであろうし，好きな物があればそれに目がいく．それが人間の眼であり，“未来人”の眼もそうであって欲しいと思う．

ここに興味深い（超ローカルな）調査がある．被験者10人に対し，以下のような調査を行った．

図 1: ことりちゃんマジ天使な画像

Q. 図1を見て下さい．
貴方はこの画像の何処に目がいきますか？

A.

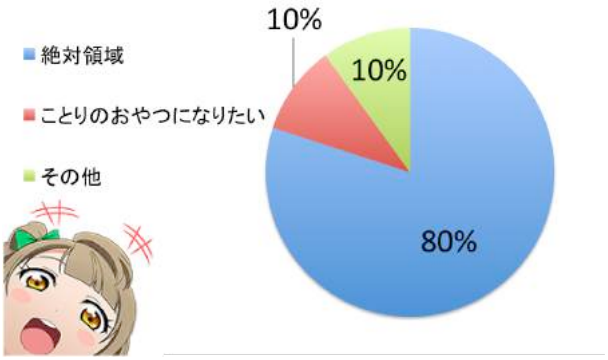


図 2: 調査結果

図2を見る限り，被験者からのことりマジ天使具合が伝わってくる．ちなみに，その他と答えた人は「見えた！」と語っていた．さて，中でも着目したいのが，被験者の80％がことりちゃんの絶対領域に目がいっていることだ．絶対領域とは，スカートとニーソックスの間から目見える太ももものことであり，今回の調査によれば多くの人がそこに目がいったことになる (もちろん，筆者も)．私はこの事実を真摯に受け止め，“未来人”の目もそこにいくべきだと考えた．このパッションを胸に，本研究の目的は「絶対領域を絶対に検出する」ことである．少し長文になってしまったが，最後まで読んでいただければ著者は幸いである．

1　手法の説明

今回の実験では，特徴抽出としてLBP(Local Binary Pattern) 特徴を用い，識別器には線形識別器 SVM(Support Vector Machine) を用いている．昨年度執筆した『HOG特徴を用いた乳首修正パッチシステム』において，HOG特徴を用いたため，今年は違う特徴量をということでLBP特徴を採用した．識別器についても軽く言及しておくと，線形識別器とは, 与えられたデータに対し，それが正例か負例かを判定するものである．今回の実験では，正例が絶対領域，

負例が絶対領域でないとして考えていただきたい．SVM は線形識別器の一つであり，割とポピュラーな識別器である．

今回の実験における，手法の流れを図 3 に示す．

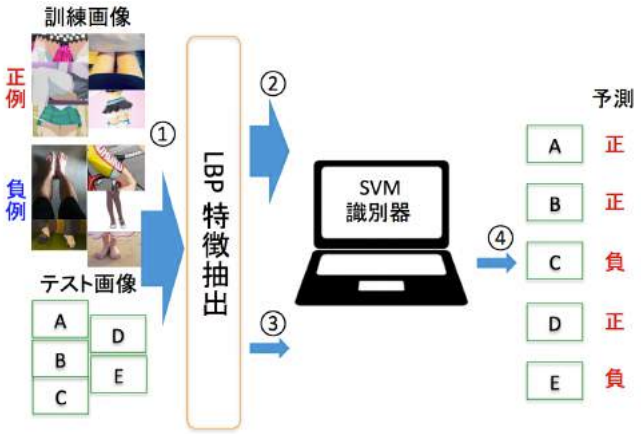


図 3: 手法の流れ

それぞれのステップの内訳は，以下のようになっている．

1. 訓練画像 (正・負が分かっている画像)，テスト画像 (正・負が分かっていない画像) の LBP 特徴を取得する．
2. 訓練画像の LBP 特徴とラベル情報 (正か負かの情報) を用いて識別器を学習する．
3. 学習した識別器に対し，テスト画像の LBP 特徴を入力する．
4. 入力されたテスト画像に対し，正か負かを予測する．(ここで，実際にはテスト画像の真のラベルは分かっているので，予測との比較で正答率が出せる)

LBP 特徴量や SVM についての理論的説明も載せたいところであるが，紙面の関係上省略する (数式ゴジャゴジャだし)．また，正例か負例かの情報をラベルと呼び，訓練画像の各画像が正例か負例かという情報を “真のラベル” と呼ぶことに留意されたい．

2 実験

今回行った実験について示す．

【実験条件】

訓練画像：

“絶対領域” でヒットした画像 50 枚の画像 (正例)

適当に集めた絶対領域でない 50 枚の画像 (負例)

テスト画像：正・負混ざった 20 枚の画像

画像サイズ：250 × 250 ピクセルに統一

各種パラメータ：ベストエフォートなパラメーター

【実験結果】

今回行った実験では，正答率 83 %をマークした．

3 考察

本研究においては，83 %という中々な正答率を得ることが出来た．しかし，“絶対” ではないため，間違ってしまったテスト画像について考察していきたいと思う．

3.1 事案 1(ガチムチ問題)

図 4: 事案 1

図 4 は不正解となった画像の一枚である．この画像に対しては，真のラベルでは負例 (絶対領域でない) とされていた．しかし，予測では正例 (絶対領域) と判断されてしまった．これは，スカートと太ももが，絶対領域の特徴と酷似していたことに起因すると考えられる．ガチムチに対しての処理が必要である．

3.2 事案 2(パンチラ問題)

図 5: 事案 2

図 5 も不正解となった画像の一枚である．この画像に対しては，真のラベルでは負例 (絶対領域でない)，予測では正例 (絶対領域) と判断されてしまった．これは一見，真のラベル付けが間違っているように思われるかもしれないが，パンチラの場合，もはやパンツの方に目がいってしまうため，今回の目的においては真のラベルを負例 (絶対領域でない) として考えている．人間の眼を再現したいので，“人間の目が自然にいく” 場所という意味を含めた負例である．“未来人” の眼を作る上では，パンチラに対しての処理が必要である．

4 終わりに

今回の実験においては，ガチムチ問題とパンチラ問題が残った．特にパンチラ問題の方は，私が勝手にパンツの方が絶対領域よりもエントロピーが高いとして先ほど論じてしまったが，今後一般的な調査が必要である (著者はパンツ強い派)．最後に，今回ことりちゃんをプッシュしていたが，他派閥のラブライバー様に喧嘩を売る目的ではないこと，そして本論文は原作を否定するものでなく，純粋に “絶対領域” を考察するためのものであることを強調しておく．

出典

南ことり@ラブライブ！
その他たくさんの画像@ Web

初心者が必死で伝えるプログラミング言語「なでしこ」の魅力

Zakky

概要

OSC Tokyo 2015 Spring に参加したとき日本語プログラミング言語「なでしこ」の存在を知り，面白そうだったので初めてみた.「なでしこを使う者[1]」になってまだ 5ヶ月の自分が「どうしてなでしこがいいのか」をたどたどしく，必死に解説します.

1　日本語プログラミング言語「なでしこ」とは

なでしこは，クジラ飛行机氏が開発したインタプリタ方式のスクリプト型プログラミング言語.「日本語プログラミング言語」とあるように文法が英語ではなく，日本語となっている．Excel/Word と連携したりなど日常で使いやすいものを簡単に作ることができる.

以下のページから無料でダウンロードする事ができる (Windows のみ)

`http://nadesi.com/`

2　「なでしこ」のよさを紹介していく

2.1　なんといっても日本語

先述したとおりなでしこは，日本語でプログラミングを行う．変数名や，命令[2] 名を英語ではなく日本語で記述可能な他,「+, −, ×, ÷」などの演算子も日本語で表現できる[3].

図 1 のように,「りんご**は** 150 円」と，演算子の「=」を日本語で表現でき，合計を求める部分も「A **と** B の**合計**」という表記で他の言語での「sum(A, B)」のようなものを表現できる．もちろん通常の演算子も使用できるので，下の方にあるような英語プログラムライクの文法も使用できる．なお，日本語入力を考慮してか，記号は全角・半角どちらでも問題ない.

[1]Python を使う人のことを Pythonista，Ruby を使う人を Rubyist などと呼ぶが，なでしこにはそのような呼び名が見つからなかったので勝手に命名．日本語の言語だしどうせなら呼び名も日本語でってことで.

[2]他の言語での関数などの意味

[3]あんまりやらないほうがいいかもしれない．(理由は後述)

```
1  りんごは150円
2  みかんは100円
3  金額はりんごとみかんの合計
4
5  「{金額}円」を表示
6
7  strawberry = 200
8  peach = 300
9  sum = strawberry + peach
10
11 表示(「¥{sum}」)
12 [EOF]
```

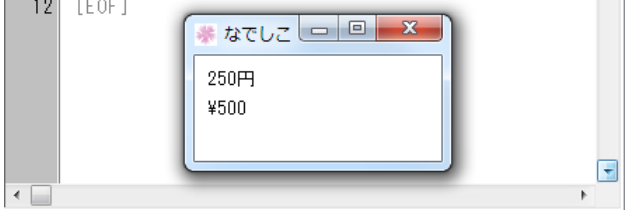


図 1: それぞれの値段の合計を表示する

2.2　初心者にやさしい開発環境

なでしこにはなでしこエディタと呼ばれる開発エディタが用意されている．エディタでは，なでしこ言語での命令の強調表示，プログラムの実行のほか，exe ファイルへの出力もすることができる.

また，命令を検索するフォームもついているため,「この動きどう書くんだっけ？」といったようなものを探すことができる．なでしこでは命令に引数を「から」や「と」など助詞で表現するのだが，そこにはその関数の引数の取り方も説明されているので，簡易マニュアルみたいに使用することもできる．もっと詳細が知りたければそこから詳細な説明をしている web ページに飛ぶこともできる.

GUI 実装も簡単に行える．なでしこには積み木デザイナエディタと呼ばれる GUI エディタの機能があり，なでしこエディタのデザインタブを選ぶと使うことができる．好きな場所にオブジェクトを配置し，そのオブジェクトを選択するとそれらの動きが設定できる，設定したものは自動でソースコードに記述されるので，GUI 実装をしたことがない人でも簡単に作成することができる.

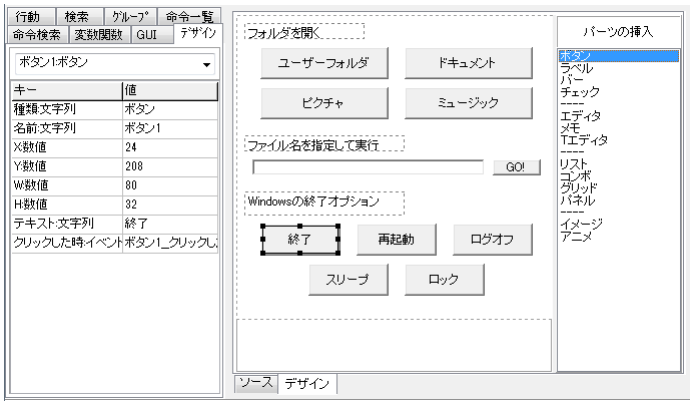


図 2: GUI 実装画面 (簡易スタートメニューを作成中)

2.3 Windows，Office との連携機能

Windows の操作を行う命令，Word・Excel の操作を行う命令がなでしこには多く搭載されているので簡単なマクロを組みたいと思ったときに便利．

List1 にある,「エクセル〜」といった命令はエクセルを制御する命令である．このプログラムでエクセルの起動・終了，ファイル選択・保存，セルの読み取り・書き込みを行っている．

また,「Windows〜」といった命令を使うと，Windows の操作も行うことができ，起動，終了等の操作も行うことができる．

List 1: 転置行列をエクセルで行うプログラム

```
1  母艦の可視はオフ
2
3  Nは4
4
5  エクセル起動
6  「.xls,.xlsx,.xlsm」でファイル選択
7  ファイル名はそれ
8  それをエクセル開く
9  Sheet1にエクセルシート注目
10
11 元はCELL(1,1)からCELL(N,N)までエクセル一括取得
12 元を表示
13
14 エクセル新規シート
15
16 iで0からN-1まで繰り返す
17   jで0からN-1まで繰り返す
18     CELL(i+1,j+1)に元【j,i】をエクセルセル設定
19
20 ファイル名へエクセル保存
21
22 エクセル終了
23
24 終了
```

3 なでしこのデメリット

ここまで読んでくるとなんだか「なでしこすげぇじゃん!もう全部これでいいじゃん!」って感じになってくるがもちろんデメリットもある．

「なでしこ」では引数として助詞を用いているが，その助詞は自然な日本語になるように設定されている [4]．つまり引数の取り方が統一されていない．

図 3 のように四則演算だけでも命令の引数がバラバラである．全部日本語でやろうとすると国語の授業並みにきれいな日本語を強制させられてしまうので演算などは記号を用いたほうがいいだろう．

```
1  aは10
2  bは2
3
4  aとbを足す
5  それを表示
6
7  aからbを引く
8  それを表示
9
10 aにbを掛ける
11 それを表示
12
13 aをbで割る
14 それを表示
15 [EOF]
```

図 3: 四則演算

また，助詞は予約語となっているため変数に使うことができないのでここも注意．

さらに何度も説明はしているが，動作環境は Windows のみであるので完成したプログラムを Mac や Linux に持って行っても起動しないので気を付けなければいけない．現在なでしこ 2.0 として C# で書かれており，他の OS でも使えるものを開発している [5]．

4 まとめ

初心者がここまで必死に説明してきましたが「なでしこ」の魅力に気づいてもらえたでしょうか．この記事全体においてデメリットの項を除いて,「できる」しか言ってない気がしますが実際「できる」んだから仕方ないと思います．他の言語を扱っている人からすれば,「日本語プログラミング言語なんか遊びの範囲でしか使わないだろ」と思うかもしれませんが (実際自分も思ってました)，ちょっとしたことのプログラムを書くには「なでしこ」と言う言語はすごく適していると思うのでぜひ触って実感してほしいです．

「なでしこ」はプログラミング初心者も上級者も使って楽しい言語だと思います．Wiki なども作られているのでいろいろ調べながらぜひチャレンジしてみてください．

あなたも「なでしこを使う者」になろう!!!

[4] 先述したように “足す (a, b)“という書き方もできるが推奨はされていない．

[5] http://nadesi.com/dev/wiki/index.php からダウンロードできるが現段階ではあまり実用的ではない．

電動パワーステアリングシステムのためのモデリング

トム

はじめに

今回とりあげるのは、「電動パワーステアリング (EPS)」についてです。これは車の自動運転技術や省エネに関係した技術であり、皆さんが乗っている車の多くにすでに実装されています。この記事では EPS についての概略と簡単なシミュレーションの方法についてお話したいと思います。また、最後にこのモデルと従来制御の課題について取り上げます。

1　電動パワーステアリングとは

パワーステアリングとは、人の手に合わせて車自身がハンドルを回す力を加えることで、重さが 1000kg 近くある車の重いハンドルを思い通りに回せるように力を補助するシステムです。現在ではほぼすべての車に実装されており、最近は従来の油圧式ではなくモータを利用した電動式が普及しています。EPS は電気でアシストするため燃費が良く、各センサから得られる情報から車の次の状態を予測し、モータを動かすことで運転補助および自動運転を実現することができます。今回は比較的単純な機構を持つラック＆ピニオン式を例として取り扱います。

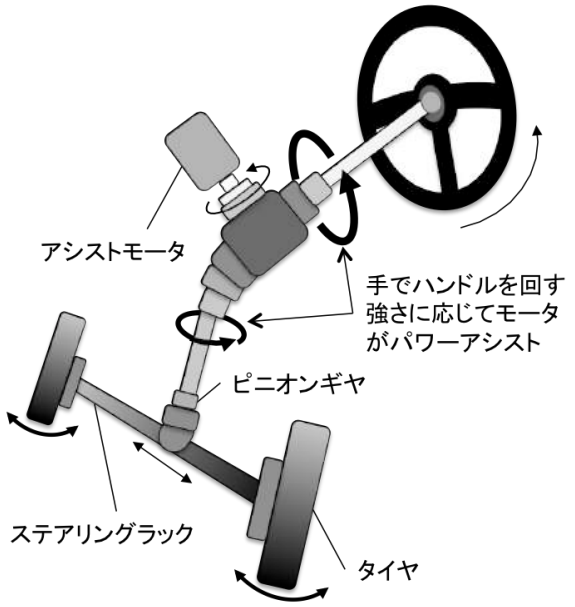


図 1: ラック＆ピニオン式電動パワーステアリングシステム

2　ステアリング機構のモデリング

まずは主となるステアリング機構のモデリングです。$\theta_h, \theta_m, \theta_T$ はハンドル･モータ･タイヤ回転角度を表し、T_{SAT} はセルフアライニングトルクといい、ハンドルを回したときに元の中心に戻ろうとする力のことです。これらを剛体の回転運動とみなすと、式 1, 式 2, 式 3 の微分方程式で表せます。

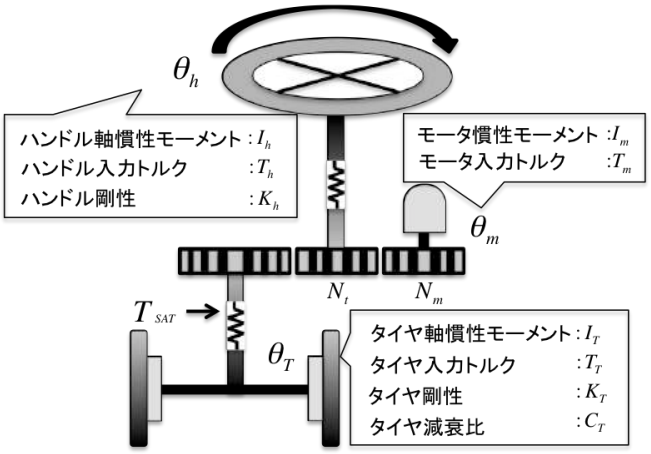


図 2: ステアリング機構のモデル

$$I_h\ddot{\theta_h} + K_h(\theta_h - \theta_T) = T_h \tag{1}$$

$$I_m\ddot{\theta_m} + K_h(\theta_m - \theta_T) + T_T = T_m \tag{2}$$

$$I_T\ddot{\theta_T} C_T\dot{\theta_T} + K_T\theta_T = T_T + T_{SAT} \tag{3}$$

ステアリングギア比 N_t、モータギア比 N_m より、その力をすべてタイヤ軸周りに変換します。添字の小文字は各軸での運動や係数であることを示し、添字の大文字はタイヤ軸周りへの変換値とします。$\theta_M = \theta_T, I_M = (N_tN_m)^2I_m + I_T$ とし、そして制御に用いやすい形式である状態方程式に変形します。また T_M を式 5 で定義します。

$$\begin{bmatrix} \dot{\theta_H} \\ \ddot{\theta_H} \\ \dot{\theta_T} \\ \ddot{\theta_T} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K_H}{I_H} & 0 & \frac{K_H}{I_H} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{K_H}{I_M} & 0 & -\frac{K_T+K_H}{I_M} & -\frac{C_T}{I_M} \end{bmatrix} \begin{bmatrix} \theta_H \\ \dot{\theta_H} \\ \theta_T \\ \dot{\theta_T} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{T_H}{I_H} \\ 0 \\ \frac{T_M+T_{SAT}}{I_M} \end{bmatrix} \tag{4}$$

$$T_M = K_1K_H(\theta_H - \theta_M) - K_2C_T\dot{\theta_M} \tag{5}$$

3　車両のモデリング

次は、車両のモデルを見ていきます。下記は車体の理論解析のなかで最も単純化されたモデルです。自動車は四輪ですが左右には同じ力が働くと考え、等価二輪モデルとして単純化しています。座標は車両固定で、進行方向を x 軸、垂直方向を y 軸とすると状態方程式は式 6 のとおりとなります。

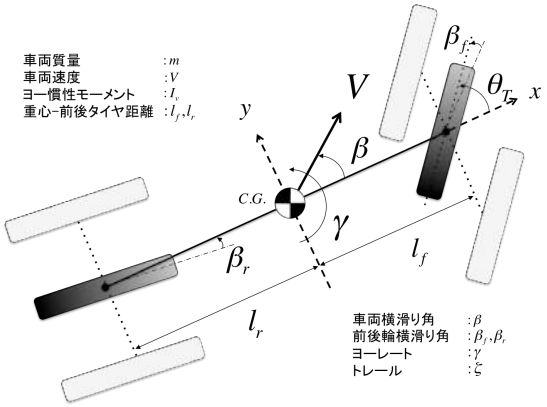


図 3: 車両のモデル

$$\begin{bmatrix}\dot{\beta}\\ \dot{\gamma}\end{bmatrix}=\begin{bmatrix}\frac{-2(K_f+K_r)}{mV} & \frac{-2(l_fK_f-l_rK_r)}{mV^2}-1\\ \frac{-2(l_fK_f-l_rK_r)}{I_v} & \frac{-2(l_f^2K_f-l_r^2K_r)}{VI_v}\end{bmatrix}\begin{bmatrix}\beta\\ \gamma\end{bmatrix}+\begin{bmatrix}\frac{2K_f}{mV}\\ \frac{2l_fK_f}{I_v}\end{bmatrix}\theta_T \tag{6}$$

2 章のステアリング機構で解説したセルフアライニングトルク T_{SAT} はこの状態方程式の解を用い、以下のように表します。この値を先ほどの式にフィードバックすることでステアリング-車両のシミュレーションができます。

$$T_{SAT}=-2K_f\zeta(\theta_T-\frac{l_f}{V}\gamma-\beta) \tag{7}$$

4　シミュレーション

今回は Matlab/simulink という制御用ソフトウェアでブロック線図を図 4 のように組み上げ、シミュレーションを行いました。

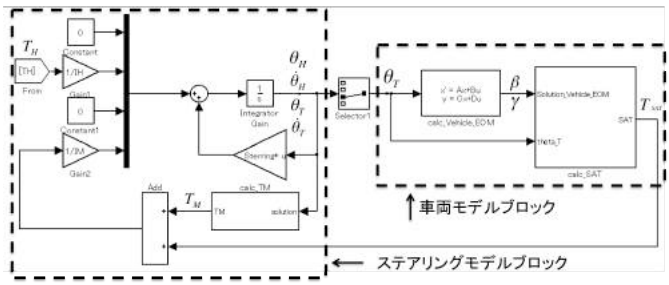


図 4: ステアリング-車両統合ブロック線図

図 5 はモータのアシストの強さである K_2 を変化させた時の特性の変化を表したものです。入力トルクに対して、(a) はヨーレートのインパルス応答 (ハンドルを一瞬だけ回した時の車の運動)、(b) はハンドル角のリサージュ曲線です。K_2 は操舵力特性と車両の収斂性に関係しており、これらを両立させるために様々なパラメータを繰り返し変更しながら最適値を設定します。

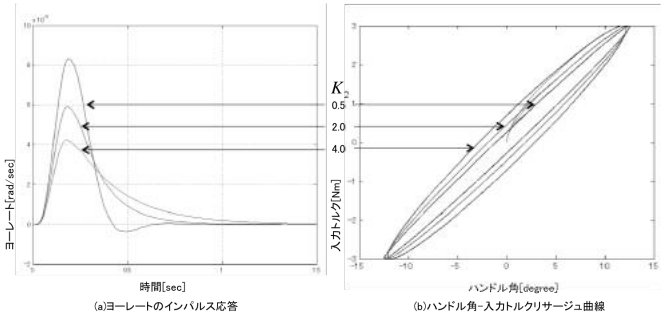


図 5: K_2による特性変化

5　従来制御の課題

従来制御で課題となるのは操舵力特性と車両収斂性のトレードオフ関係です。例えば、操舵力特性を優先させると車両の収斂性が不十分になり、また逆のケースも起こります。

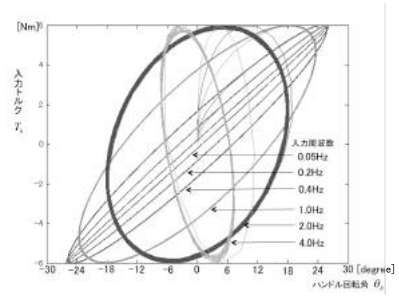


図 6: 入力周波数と操舵特性変化

図 6 を見ると入力周波数 (ハンドルを回す早さ) を変えるごとにリサージュ曲線の形が変化してゆく様子がはっきりと確認できます。ほかにも車体の質量 (乗客・荷物の重さ) や速度、道路の凹凸などが曲線の外形に大きな影響を与えます。これらが実際にハンドルを回すときの感触に深く関係しており、このままではあらゆるパラメータの変化がそのままハンドルを回す感覚に影響を及ぼしてしまいます。このような変化、変動に対して車両の操舵特性や運動特性の変動は当初の目標値から大きく変動しないことが望まれます。これらを成立させるためにあらゆる制御ロジックが提案され、今日に至っています。

参考文献

[1] 竹原 伸, 吉岡 透, “外乱オブザーバを用いた電動パワーステアリングによる操舵/車両応答特性の向上” 日本機械学会論文集 (C 編)70 巻 698 号 (2004-10).

AviUtlWorkshop 〜AviUtl を使った動画編集講座の記録〜

風土

概要

「AviUtl」とは、「KEN くん」氏が個人で開発しているフリーの動画編集ソフトである。

筆者は以前から、ET ロボコン 2014 北関東大会のオープニング動画を担当したり、IGGGmeetup などで AviUtl を用いて編集した動画を公開したり、AviUtl による動画編集について発表などを行ったりしていた。そこから「AviUtl や動画編集に興味を持った」というメンバーの声があり、AviUtl を用いた基本的な動画編集を勉強する会、名づけて「AviUtlWorkshop」を開催する運びとなった。

本記事はあくまで「AviUtlWorkshop の報告書」として作成したものであり、具体的な動画編集については省略する箇所もあるということをご理解頂きたい。

1　動画編集とは

動画編集は、主に２つの手法に分けることができる。

1. 動画を素材として編集し、新たな動画を作る手法。
2. 静止画を素材として編集し、新たな動画を作る手法。

前者は、ビデオカメラなどで撮影した動画の不要な部分を切り取ったり、効果音をつけたりすることが主たる目的である. これを非常に極端に行ったものが「MAD 動画」と呼ばれるものになる。後者は、静止画に動きと BGM をつけ、動画として再構成するものである。これによって作られたもの動画は「静止画 MAD 動画」と呼ばれる。静止画 MAD 自体も「静止画 MAD として作られた動画を素材として並べて、最終的な動画として出力させた MAD 動画」という捉え方もできるため、「静止画 MAD 動画を作るためには、普通の MAD 動画を作るための技術も必要である」とも言える。

2　AviUtl とは

WindowsPC に標準搭載されている動画編集ソフト「Windows ムービーメーカー」は、シンプルな MAD 動画を作るだけなら非常に有効である。しかし、細かなカット切り替えを必要とする動画編集には向いておらず、静止画 MAD の編集に関してはほぼ不可能である。一方 AviUtl は、Windows ムービーメーカーに比べて扱いが難しいという欠点を持つものの、編集の自由度ははるかに上である。最小で 0.03 秒ごとにカット割りが可能であり、最大 100 枚のレイヤーを同時に用いることで、複雑かつ多彩な編集を行うことができる。静止画自体にも XYZ 軸の動きをつけることが可能であり、「カメラ」の使用によって、2D の素材を擬似的に 3D 動画にすることもできる。

3　AviUtlWorkshop 開催にあたって

AviUtlWorkshop の参加者を募ったところ、6 名の参加表明があった。

各参加者のスケジュールの都合上、2 回以上にわたって講義を開催することにした。第 1 回を 6 月 27 日 (土)、第 1.5 回 (第 1 回に参加できなかった人向けの補講。内容は第 1 回とほぼ同様) を 7 月 9 日 (木) に行った。講義の会場は IGGG 部室とした。

4　第 1 回 AviUtlWorkshop 活動内容

6 月 27 日 (土) 開催。4 名が参加。うち 3 名が IGGG メンバー、残り 1 人は外部からの参加である。

11 時より講義を開始し、途中で昼食休憩をはさみつつ、17 時に終了した。

AviUtl と拡張編集プラグイン、フリーのエンコーダツールである「つんでれんこ」の導入から、簡単な動画の出力までを行った。

4.1　AviUtl とつんでれんこのインストール

まずは、AviUtl などの編集環境の導入からレクチャーした。

AviUtl の開発者 HP「AviUtl のお部屋」から、AviUtl 本体及び拡張編集プラグインが同梱された zip ファイルをダウンロード。このままでは拡張子が avi のファイルしか読みこむことができないので、L-SMASH Works と呼ばれるプラグインの導入が必要である。しかし、配布元のサイトが利用不可（サーバーの使用期限が切れた？）なため、筆者

が自身の PC に取り込んだプラグインのコピーを渡すことで解決した。

同様に、つんでれんこの開発者 HP「つんでれんこのお部屋」からつんでれんこ本体をダウンロード。以上で編集環境の導入が完了した。

4.2 震える名前

まずは、「動画を出力する」という、動画編集に最低限必要な動作を練習する。

拡張編集のタイムラインにテキストで「各自の名前」を入力し、それを「アニメーション効果」の「震える」を用いて振動させる動画を作成、avi 出力によって動画として出力をする。avi ファイルは非常に容量が大きいため（10 秒間の動画で 334MB ほど）、PC のスペックによっては処理落ちが発生する。そこで、エンコーダによって圧縮する必要がある。つんでれんこを用いてエンコードすることで、容量を 1000 分の 1 以下にすることができる。

4.3 回転する立方体

カメラ制御による擬似 3D 動画の作成の練習。

カメラ制御と拡張編集機能を用いて、平面である正方形を 6 枚組み合わせ、立方体を組み立てる。立方体をグループ制御によって X 軸および Y 軸で回転させる。正方形は「静止画」であるため、これも「静止画 MAD 動画」である。立方体を構成する正方形が同一色であると回転がわかりにくいため、それぞれ異なる色で彩色した。

4.4 動画で構成された回転する立方体

先程述べた「回転する立方体」を構成する正方形を、同じサイズの動画に置き換える。

動画ファイル自体に「カメラ制御」を適用することで、動画自体も XYZ 軸で動かすことができる。

4.5 スーパーボール

中間点を用いて、様々な方向に移動するオブジェクトを作成する。

オブジェクトが表示されている間、中間点を「区切り」として、物体の移動方向やエフェクトの付与時間などを自由に調節できる。これはオブジェクトに限らず、カメラの座標などにも適用できる。これによって、XYZ 軸空間を視聴者が飛びまわっているような動画を作ることが可能である。

4.6 時計

中間点による編集の応用。

中間点を用いて針状のオブジェクトを一定間隔で回転させ、60 秒で 1 周する時計を作る。

0.5 秒で 6° 回転し、0.5 秒間停止する運動を 60 回繰り返すことで実現可能である。ここでの作業で 1 時間半近く時間を取ってしまったが、これは筆者自身の知識不足による時間浪費によるものである。

5 第 1.5 回 AviUtlWorkshop 活動内容

7 月 9 日 (木) 開催。2 名が参加、2 人とも IGGG メンバーである。

第 1 回の補講として行ったものであり、内容を一部省略して開催した。

5.1 AviUtl とつんでれんこのインストール

事前に開発環境導入のための資料を渡しておいたので、講義開始時には環境設定がほぼ完了していた。

5.2 動画編集演習

作成した練習動画は「第 1 回 AviUtlWorkshop」に準ずるため省略する。

6 まとめと反省

AviUtl による動画編集の自由度は非常に高いため、動画製作者の数だけ、様々な編集方法を用いて「自由な動画編集」を行うことができる。AviUtl は、あっちこっち適当に触って、遊んで、見つけ出すツールだと考えている。自分が意図せずに作り上げた動画というものが、意外と良い味を出したりすることは少なくない。それは筆者の経験則でもある。今回の AviUtlWorkshop において、動画編集の楽しさというものを少しでも知ってもらえたら嬉しいと思う。

今回の反省点としては、筆者自身の知識不足についてである。1 年近く AviUtl による動画編集を行っているが、自分一人だけではどうしても気づかない点を、講義を受ける側の学生から指摘されたことが度々あった。そういう意味では、筆者自身も多くのことを学べた講義であったと考える。

今回は「無声動画」のみを作成したため、第 2 回は「音楽を用いた動画」及び「カメラワーク」をメインとした講義を行う予定である。学生が長期休業期間である 9 月中の開催を予定している。

ジャンク修理歴5年のプロが教えるジャンク修理

桐生川 (kuriuzu)

ドーモ、皆＝サン。クリウズです。ニンジャ殺すべし、慈悲はない。今期アニメはニンジャスレイヤーフロムアニメイシヨンを見てます。Flash の衝撃がヤバかった…。
今回は、5 年前からジャンク品を修理しまくってきた経験をもとにジャンク修理についてのレポートを書いてみました。ジャンク修理は楽しいのでまぁ暇な人とかは挑戦してみるといいんじゃないでしょうか。

1　はじめに

ジャンク修理は電子工学とか機械工学の知識が必要です。(逆に言うとジャンク修理でそれらの知識が身につく。）あとハンダ付けとかするので工具も必要です。電子工作ができるセット（はんだごては温調式を推奨）があればよいでしょう。テスターも必要です。アナログでもデジタルでも自分の好きな方で。導通チェック機能（抵抗が 0 オームだとブザーがなる）は必須です。私はテスターはアナログもデジタルも両方持ってるけどデジタルの方をメインに使ってます。

2　ジャンクの入手

2.1　知り合いとの取引

知り合い、友人、家族から壊れた家電製品をもらったりする方法です。タダで入手できます。他にも、大学の一斉廃品回収日（ジャンク祭り、ゴミケと呼ばれる）に大量にゴミ集積所に集められたものから漁ったりします。我々IGGG は 6 月下旬に開かれるジャンク祭りでディスプレイとか PC を拾ってきて修理するのが恒例行事となっています。私はそれで Tektronix の新品価格は 100 万円は越えるであろうデジタルオシロスコープを拾ってきて直しました。

2.2　ハードオフのジャンクコーナー

私は、ジャンク品はハードオフのジャンクコーナーで入手することが多いです。月 2 回くらいジャンクコーナーを見にハードオフに行ってます。ハードオフのジャンクは値段高めですがたまに安い掘り出し物もあります。ハードオフのジャンクコーナーで入手しやすいのは PC、デジタルカメラ、ディスプレイなどです。

図 1: 我が大学のジャンク祭りの様子

2.3　ヤフーオークション

品物はとても多いです。見つからないものはないと言ってもいいでしょう。値段は一番安いのではないでしょうか。でも送料が高い。修理失敗品とか商品説明と状態が異なる品物を送りつけられたりといったリスクが高いのであんまりおすすめしないです。

2.4　秋葉原

秋葉原でも最近はジャンク品を扱う店はかなり減ってしまったようです。しかし、最近「PCNET　秋葉原ジャンク通り店」が開店しました。今後に期待です。
「PCNET　秋葉原中央口店」は、デスクトップ PC が多いです。
「インバース」という店はノート PC、のジャンクを多く扱っています。
「じゃんぱら」は中古の自作 PC パーツとかディスプレイとか。ジャンク以外のものも多いです。ラジオデパート地下のジャンク屋「秋葉原エレクトリックパーツ」は業務用の PC 周辺機器とか多いです。珍しいものが売ってたりします。

3　ジャンク品の見分けかた

「動作未確認」という商品は、対応する AC アダプタがないなどの理由で動作確認をしていない物です。こういうのは 7 割くらいは正常品です。(ただしヤフオクを除く) あとで互換の付属品を買えば修理することなく普通に使用できます。Amazon で探せば出てきます。
「通電確認」というのは、「コンセントを入れたときに電源ランプが点灯した (動作するとは言ってない)」という意味です。動作しません。修理が必要です。俺はこれにダマされて YLOD の PS3 を買うハメになった…。

4　ジャンク品の故障原因と修理

故障原因で一番多いのは「電解コンデンサの寿命」です。電解コンデンサの異常は見た目で分かります。異常な電解コンデンサは頭が膨らんでたりします。電解コンデンサをはんだごてを使って交換する必要があります。
二番目に多いのは「はんだクラック」です。はんだクラックとは、衝撃や温度変化によってはんだに物理的応力がかかり、剥離し接触不良となったものです。PS3 の YLOD、xbox360 の RROD が有名です。ほとんどは目視で見つかりますが、細かくて目視で見つからないものもあります。はんだクラックの修理ははんだごてでハンダ付けをやり直したりします。PS3 の YLOD、xbox360 の RROD、グラフィックボードといった BGA パッケージのはんだクラックは直すのは困難です。あきらめてください。フラックスを注入しヒートガンで 200 ℃に 3 分ほど加熱したりオーブンで基板ごと焼いたりすれば 50%くらいの確率で直ることがあります。

5　ジャンク品の種類と傾向

5.1　デスクトップ PC

修理難易度　★
値段　2 万円〜
HDD や電源を新品に交換するだけで動くことが多い。小型 PC は電源・マザーボードが ATX 規格ではないため交換部品が入手できない恐れがある。2003 年ごろは不良電解コンデンサーによる故障が多かったが、最近は品質が良くなったのか電解コンデンサーの不良による故障はあまり見られない。値段が千円〜数万円と幅広いが、安いやつはスペック的に利用価値がない。正直なところ、ジャンクの PC は高価なのでジャンク品を買うくらいなら少し金を足して保証付の中古 PC 買ったほうがいいと思う。

5.2　ノート PC

修理難易度　★★
値段　1 万円〜
秋葉原のジャンク屋でたくさん売ってる。HDD を交換すれば動くことが多い。分解が難しかったり部品入手がやりずらいので修理難易度は高い。ThinkPad シリーズは分解しやすく部品入手が簡単、インターネットで修理方法が検索できるのでジャンク初心者にもオススメ。

5.3　ゲーム機

修理難易度　★★★
値段　5000 円〜
修理は難しいが、インターネットで分解/修理方法が検索できる。部品入手は難しい。PS3 の YLOD・xbox360 の RROD といった BGA の不具合品は修理しても再発するのでやめとけ。

5.4　ディスプレイ

修理難易度　★★★★
値段 1000 円〜
修理するにはハンダ付け作業が必要。電源基板の電解コンデンサーかバックライト蛍光管の寿命で故障することが多いのでこれらを交換すればたぶん直る。

5.5　デジタルカメラ

修理難易度　★★★★★
値段　2000 円〜
部品がかなり細かく修理は困難。衝撃によるレンズのズーム機構のギアの故障が多い。しかし「充電器がないため動作確認なしジャンク」という商品は狙い目。リチウムイオン充電器は互換品が Amazon マーケットプレイスで 1000 円程度で買えるから安くデジタルカメラを手に入れられる。

5.6　スマートフォン

修理難易度　★★★★★
値段　5000 円〜
部品がかなり細かく修理は困難。iPhone の液晶割れジャンクなら修理できる可能性はある。腕に自信がある人はどうぞ。

5.7　グラフィックボード

Twitter で「壊れたグラフィックボードをオーブンで焼けば直る」って誰かが言ってた。[GPU オーブン][検索]

闇鍋と言えば...イエローだよー!
もう何も怖くない黄色鍋でティロフィナーレ!!

大宮・テイラー

概要

３月某日, 愉快な仲間たちを集めて鍋をやろうと考えていた. しかし, 数多の鍋料理を作ってきた我々[2] には, ただの闇鍋では物足りず, 今回の鍋の方向性が決まらない. そこで, とある後輩ちゃんに聞いてみた.
「何鍋がいい？」
「えー……」
「やっぱいいや何色が好き？」
「黄色！」
こんな理由から闇鍋ならぬ黄色鍋を作ることに決定した.

1　はじめに

読者諸君らに問いかけたい. 君たちは今までに極寒の寒空の下を凍えながら自宅をめざし, 家に帰ると暖かい鍋と帰りを待つ愛すべき恋人や家族, 親しい友人が出迎えてくれる経験がないだろうか. 研究室や仲間内でのギスギスとした現代社会の荒波に揉まれ冷え切った心と体, あの頃の輝きを失った瞳を真まで包み込み癒してくれる起床後に離れてくれないオフトゥンの様な存在が「鍋料理」である（図 1, 図 2）.

図 1: 鍋を囲む子供たちをたしなめる保坂

図 2: 囲まれる新鮮なタラをふんだんに使った鍋

今回, 鍋料理に新たな発想もたらすべく, 各員が持ち寄った好きな食材ではなく色によって入れる食材を選択した. 料理において彩り, つまりビジュアルは最重要項目といっても過言ではない. 赤・黄・緑の彩りは食欲を増大させ食卓をシアワセの色にする. 特に黄色は「集中力を発揮させる」「判断力が UP する」「注意をうながす」などの心理効果がある [1]（図 3, 図 4）.

図 3: 黄色は判断力 UP の色

図 4: 凛ちゃんと言えば? イエローだよ!

もちろん, 今回の黄色鍋も闇鍋と同じく, 食材を一つの鍋の中に入れ煮込んだ汗と涙の結晶であり, バランス栄養食である. 大切なことは鍋をやる愉快な仲間の確保とどんな自由な発想で鍋を調理しても良いということである（図 5）. ここに記すのは鍋料理に変革と革命を起こすべく終結した開拓者たちの記録である.

図 5: 自由な発想で鍋の具材を入れるスクールアイドル

2　方法

今回の黄色鍋は一般的な鍋料理や闇鍋と違い, 鍋に入れる食材を黄色っぽい食材で調理した（図 6）.

図 6: 黄色い食材たち

今回用いた黄色い食材たちは以下の通り.

前半: 長崎ちゃんぽん鍋スープ, 白菜, モヤシ, 豆乳プリン, 油揚げ, ウインナー, かぼちゃ, 鶏肉

後半: からしマヨネーズ, 黄色いパプリカ, たくあん, バナナ, 卵豆腐, カステラ, ドライフルーツ, レモン, とろけるチーズ, パインアメ

以降より今回の鍋の食材の特徴を示す. なお解説しやすいよう前後半に分け記した.

3 結果・考察

3.1 前半パート

前半に加えた食材は闇鍋ならぬ一般的な食材が多く, ひとときの平穏な闇鍋生活を満喫できた. 鍋の根元としての白菜, 鶏肉と万全の布陣である. また, 黄色枠として油揚げ, かぼちゃ, ウインナー（チーズ in）, もやし, そして**豆乳を投入**した. もやしは黄色い屋根のラーメン屋さんでたくさん盛られているので増し増しで加えた. 今回は黄色の食材という前提より、プリン味の**豆乳を投入**した（図 7）.

3.2 後半パート

後半に加えた食材はこれまで通り期待を裏切らないケイオスな食材の数々を加えた. ここから先は黄色い食材のオンパレードである. からしマヨネーズから始まり, イエローオブイエロー:黄色の野菜王パプリカや白米の随伴艦として名高いたくあん, フルーティー枠でバナナ, レモンを加えた. そしてダシが取れそうなドライフルーツとパインアメ, 最後に貴重な鍋の水分を吸い取るカステラと部屋全体をチーズの世界に導くとろけるチーズを鍋に入れ闇鍋ならぬ黄色鍋を完成させた（図 8）.

図 7: 平和だったころの鍋

図 8: 黄色に染まった鍋

3.3 試食

今回の黄色鍋は食べ進めるにつれて汁の味が変わっていくものだった. 最初はとろけるチーズの味が強かったが, 次第にレモンの酸味が増していった. ベースとしてあったはずの長崎ちゃんぽん鍋スープの原型はどこへ行ったのか…. 終盤では油のような黄色いスープ層とそうでない層に分離した. 上側の黄色いスープの層は味がない味がして衝撃を受けた. おそらくからしマヨネーズにより形成されたと思われる.

普段の鍋になかなか投入しない食材も個性的な味を見せた. 調味料としてではなく具材として入れたレモンは, 食べると酸味と苦さを兼ね揃えた暖かい何かだった. まるでドライマンゴーのような見た目で鍋を彩ったたくあんは序盤こそ暖かすぎるたくあんといった感じの味だったが, 後半になるとたくあんの食感はするが味はたくあんでない何かになっていた. たくあん本来のエキスが黄色鍋の黄金のスープに溶け出したものと思われる. カステラは鍋の水分を吸ってしまい, カステラ本来の甘さよりも黄色鍋のスープの味が強かった. その他, バナナやパプリカなどは普段温めて食べないものを温めたらこんな感じだろうといった感じだった.

4 まとめ

完成した闇鍋ならぬ黄色鍋は食べるたびに変化するスープとレモンの酸味と苦味と苦味の効いたものだった. 普段なかなか買わない食材を, 普段なかなか行われない調理方法で調理（鍋に投入）するのは新しいものを取り入れ日々進化をとげる電気街のようであった.

今回の黄色鍋では換気をしっかりしたので闇鍋特有の甘ったるいニオイが残らなかった点と無心状態で映画「アバター」を見る異様な光景は広がることはなかったので比較的平和に終えることができた（換気の慢心, ダメ, 絶対）. また今回好きな色が「黄色」であったから良かったが赤なら辛そう, ターコイスなら再現が難しそう, そして闇色と答えられたら世紀末な闇鍋ができていたかもしれない. 次回, 色で鍋を決める機会があればちょっと挑戦してみたいと考える.

最後に, もし, これを読んで鍋の味を決める際に, 色で挑戦しようと言う勇者が現れた時, どんな状態でも最後まであきらめずルールを守って楽しく闇鍋を行うことができることを切に願う. 非常にオリジナリティーのある世界で一番素敵な闇鍋ができるだろう. ぜひ鍋料理を君たちの色で染め上げてくれ.

「ファイトだよ！」

参考文献

[1] 色カラー（黄色のイメージ黄色が与える色の効果）. http://iro-color.com/episode/about-color/yellow.html. (2015.8.2 閲覧).

[2] 大宮・テイラー. 正しい闇鍋のプロトコール. http://ftp.iggg.org/taira/yaminabe.pdf. (2015.8.2 閲覧).

たのしいThinkPad　初心者編

Buu[†]

概要

みんな大好きThinkPad。その拡張性はまだまだ健在である。そこで本稿ではその拡張性に着目し、人柱として「5ボタントラックパッド」を搭載したThinkPad T440pのトラックパッド部を「3ボタンクリックパッド」に換装する。また同時に、JPキーボードをバックライト付きUSキーボードに差し替え、快適なThinkPadを目指す。さらに、ブート時に表示されるロゴを変更し、古き良きThinkPadを再現してみる。

1　ThinkPad is 何？

ThinkPadとは、かつてIBMという会社によって開発・設計、販売されていたラップトップPCのシリーズである。ハイエンドからモバイルまで幅広いラインナップ、ホームポジションからあまり手を動かさずにマウスカーソルが操作できるトラックポイントや工夫が凝らされたキーボード配列、本体の高い拡張性から多くの人々に愛され、他社に売却された現在でも様々な場面で使われているのを見かける。洗練された黒い筐体、キーボードの真ん中で大きな存在感を放つ赤いトラックポイント、明滅する緑のインジケータLED、そして青いエンターキー…そんな存在に心を惹かれたら、明日からあなたもThinkPadユーザーかもしれない。

2　トラックパッド換装

おっと、このままだと魅力だけを語り続けてしまいそうである。紙面も限られているので、早速換装を行ってみよう。用意したものは、

- ThinkPad T440p
- ThinkPad T440pでも使える3ボタンクリックパッド
- ドライバー各種（工具のほう）

である。3ボタンクリックパッドは、ebayなどのECサイトで45USドル程度で容易に入手可能（執筆時点）。筆者のところには香港から送られてきた。詳細はebayで「thinkpad 3 buttons trackpad」のようなキーワードを入れて検索してみよう。

オーダーから数週間後、3ボタンクリックパッドが到着。換装作業は、LenovoのWebサイトで提供されている「ハードウェア保守マニュアル ThinkPad T440p」に従った。[1] このマニュアルの76ページに「タッチパッド」の項目があるので、これを参考に以下の順番で本体をばらし、換装作業を行う。電気製品をばらすのはやっぱり楽しい。

2.1　キーボード・ベゼルを外す

まず、バッテリーパックと裏蓋を外す。そしてマニュアル75ページ、ステップ1のネジを外していく。ネジ穴を壊さないように慎重に…。さらに光学ドライブを外すとその下に1つネジが出てくるので、これも外す (マニュアル75ページ、ステップ2)。具体的な位置を図1に示す。

図 1: ベゼルを外すときに外すネジ

上記のすべての箇所のネジを外し終わったら、次にキーボードを外していく。キーボードはネジで本体に固定されており、そのネジはなんとキーボード・フレームの下に隠れている。図2に2箇所ある○で囲んである部分をよく見てみると小さな凹みがあり、ここにマイナスドライバー等を当ててフレームを上へスライドさせると、今までフレームで隠れていた部分に6個の小さなネジが現れる。このネジ6個をゆるめ、ゆっくりキーボードを持ち上げてみると、本体のシステムボードとの間にフレキシブルケーブルが2本接続されているのがわかるので、システムボード側のツメをカチッというまで持ち上げてケーブルのロックを解除したら慎重に引き抜いていこう。これでキーボードの取り外しは完了。

図 2: キーボードを外すときに外すネジ

キーボードが取り外せたら、システムボード側にあと1本フレキシブルケーブルが接続されているのが分かる。これが、パッドに繋がるケーブルである。このケーブルもキーボードの時と同じ要領で引き抜いておく。ここまできたらベゼルを取り外して、この手順は完了！(結構力が要るので、割らないように注意＞＜）全手順の中で、ベゼルの取り外しが一番大変かも。

[†]Twitter: @buu0528

[1] https://download.lenovo.com/jp/mobiles_pdf/sp40a25467_02_ja.pdf

2.2 パッドの換装

さて、念願の人柱アイテム、3 ボタンクリックパッドの導入である。図 3 に矢印で示した 4 箇所のネジを外すだけで簡単にパッドを取り外すことができる。GND として周囲にアルミテープが貼ってあるので、破れてしまわないように気をつけて…！

取り外したら、旧パッドのフレキシブルケーブルを新パッドに取り付け、あとは今までと逆の手順で組み立てていけば作業終了！

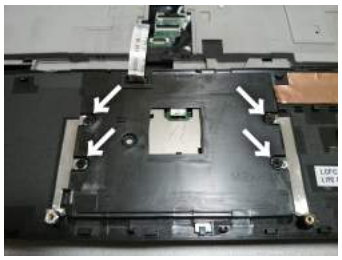


図 3: パッドの取り外し

2.3 ドライバの導入と動作確認

図 4 となり、いい感じの見た目となった。3 ボタンの押し心地も良好である。さっそく電源を入れてみた…が、ドライバが適切でないと正しく動作しない。適切なドライバのインストールが必要だが、まずは既存のドライバをアンインストールしておこう。コントロールパネルから「プログラムのアンインストール」へ行き、既存の「Synaptics Pointing Device Driver」を削除する。

図 4: 換装後

ドライバは Lenovo の Web サイトからダウンロードすることができる。新しく導入したパッド裏側の金属面がザラザラしている（細かいドットが打ってあるような質感）ものであれば、Synaptics のドライバ [2]、つるつるしている（金属板そのままのような質感）ものであれば、Alps のドライバ [3] を導入する。

これらのドライバは少し古いバージョンのもの。ThinkPad Club の「ThinkPad T440s の独立クリックボタン化改造」というフォーラム [4] によれば、最新のドライバでは ThinkPad の機種を判定し、異なる機種用のパッドが動作しないようになっているとのことで、古いバージョンを使用する必要があるようだ。

ドライバを導入して動くようになったら… Enjoy!

[2] https://download.lenovo.com/mobilesjp/n10gx25j.exe

[3] http://download.lenovo.com/ibmdl/pub/pc/pccbbs/mobiles/j5gh07ww.exe

[4] http://www.thinkpad-club.net/modules/d3forum/index.php?topic_id=6199

3 バックライト付き US キーボードへの換装

紙面に余裕がなくなってきたので手短に orz こちらも ebay で「backlit t440p us keyboard」のように検索すると 27US ドル程度で入手可能である。裏蓋を開けずに、2 章と同じ手順でキーボードだけを換装すれば問題ない。換装が終わったらデバイスマネージャーからドライバの更新を行い、「標準 PS/2 101/102 キーボード」というドライバに変更する。

4 ブートロゴの変更

とにかく紙面が足りない。2 ページなんて言わなきゃよかった。ThinkPad シリーズはシステム起動時に表示されるロゴを変更することができる。Lenovo の Web サイトから自分のモデルにあった BIOS アップデート用ファイルをダウンロードして展開したら、展開先にある「BIOS_LOGO.TXT」を読んでみよう。

言いたいことは図 5 に集約されている。このための GIF 画像を自作したので、自己責任でぜひともご使用いただきたい→ http://buu0528.com/think_boot.zip

図 5: 古き良き ThinkPad is 最高

こんなこともできる。楽しい。

図 6: ラブライブ！最高

5 謝辞

最後までお読みいただきありがとうございました。興味があれば、ぜひ挑戦してみてください。ファイトだよっ！

あとがき

ひげ

「IGGG に Ruby を浸透させよう委員会」会長のひげです(会員は約 1 名). IGGG は Python のシェアが高いので日々奮闘していますいつか IGGG から Python を追い出して Ruby 一色にするのが夢です. うそです. Python も大好きです. 共存の道を歩きましょう.

ism67ch(@ism67ch)

先に言っておくと, IGGG は変態ばかりではありません！←これ重要(OB はド変態の集まりですが)←これもっと重要　得た技術や知識を変な方向に持っていくのが好きで, 今回も前回同様ぶっ飛んでます(/∇＼*)　普段は画像処理・人工知能中心に研究してる M1 です(･ε･)　C88 には行けませんが, 我が校の文化祭で会えると思います！是非来て下さい m(_ _)m

Zakky

前回 Raspberry Pi の話だったのでそっち系の話するかと思ったら全く違う話をした Zakky です. 今度こそ売り子します. 今の悩みは部員に「なでしこ」薦めていますが誰も興味をしめしてくれないことです. みんなもやろうよ「なでしこ」. 楽しいってまぢで. 次号やるときはまた「なでしこ」でなんかやるかもしれないし, Raspberry Pi やるかもしんないし, 全く違うのに目移りしてるかもしんないです.

トム

どうも. 初投稿トムと申します. ジェリーはいません. 名前こんなんだけど純日本産です. 近頃制御とステアリングの勉強を始めたので, 今回の内容は文献をなぞるだけになってしまいましたが次回は制御ロジックを組み込んでシミュレーションを行って検証してみたいと思います. 知能化制御とかもなんか流行ってるし取り組んでみたいと思います.

風土

二度目まして, 「風土」です. 『群馬大学に AviUtl を広める会』の会長をしております（6 人が仮入会）. 前回の C87 では売り子もしてました. さて, かれこれ 1 年近く AviUtl による動画編集をしていますが, まだまだ知らないことばかりです（自分が教える側のはずなのに, 色々教わることが多いような気がしました. ）. 具体的な動画編集の仕方について書くと余裕で 30 ページくらい書きそうなので今回は自重しました（ネタの温存だなんて絶対に言えない）.

桐生川(kuriuzu)

「部誌作ろう」と言い出した張本人です. C87 に続き C88 でも部誌を作れることになりました. C87 の部誌は予想の 2 倍以上の 90 部近く売れたので, 今回もそれくらい売れてくれないかな～と期待してます. 今回の部誌の編集作業はひげさんに全部任せちゃいました. ひげさんマジでありがとうございます. 今回の部誌も楽しんでいただけると嬉しいです.

大宮

執筆するのは 2 回目、あとがきを書くのは 1 回目の大宮デース。IGGG では突撃担当と闇鍋担当です。前回の C87 では魚屋さんの叩き売りのように？売り子をやってました。今回、色によって制限を設けることによって鍋料理の新たな可能性を感じました。ただ黄色鍋の唯一の反省点はチーズケーキを入れなかったことである。闇鍋は非常に安全な食べ物です。ルールを守って、ぜひ挑戦してみてください！

buu

目玉焼きはひっくり返さない派. カレーは箸で食べる派. 好きなラブライブ！の登場人物は穂乃果ちゃん. 見た目はしゃちく, 頭脳はようじょ, Buu です. なんだか最近 IGGG の活動拠点に肖像画が貼られているみたいですね. あの写真にタイトルをつけるなら「イェイ☆」なんてどうでしょう… さて, 前回はちょっと気合をいれて http://sv.buu0528.com/d/c87vgb.pdf を執筆したんですが, 今回は紆余曲折あってすごく駄文になってしまい…申し訳ありませんでした. 次回はがんばれたらがんばります. ファイトだよっ！

発行日	2015/8/16
発行元	群馬大学電子計算機研究会
連絡先	contact@iggg.org
印刷	西岡総合印刷株式会社

IGGG Journal "Lollipop" Vol.02, Comic Market 88 Edition

群馬大学電子計算機研究会 "Lollipop" Vol. 02 C88 版

2014 年 5 月に突如立ち上がったサークル,「群馬大学電子計算機研究会 (IGGG)」のメンバーが, 前回 C87 でうまくいき過ぎてしまったがために「次もいけるっしょ！」という軽いノリで執筆した部誌第 2 号です. ウソです. 真剣です. IGGG は Information technology researching society of the Gunmer, by the Gunmer, for the Gunmer の略称の模様. グンマーの、グンマーによる、グンマーのための IT 研究会, ということでプログラミング言語や機械学習といったソフトな分野からパソコンや制御理論といったハードな分野まで, 幅広く集めた一冊となっております. ぜひお楽しみください。

IGGG

発行／群馬大学電子計算機研究会
@IGGGorg http://www.iggg.org/