
Pieのどうじんし
2
いっと☆わーくす！

Contents

Chapter 1

piet-editorを作ったので宣伝

1.1　はじめに

はじめまして、またはお久しぶりです。のな(@nonamea774)です。今回C91に当選していたのものの、10月ぐらいから体調を崩していて全然原稿が進まず、頒布を諦めようかとも結構悩んだのですが、すばらしい表紙と記事を描いて(書いて)もらったので、なんとか出すこととなりました。すごく薄い本となってしまいましたが……許して貰えると幸いです。

1.2　つくりました

周りの人がよくPietのエディタを作っているので [1] [2]、一度ぐらい作っておかないといけないかな と思って作りました。

piet-editor(https://piet-editor.github.io/)です。スマートフォンでもPietの実行ぐらいはできそうな感じですが、全ての機能を利用するにはPCから実行してください。

piet-editorは、ブラウザで描いたり実行したり共有したりできるようにと作りました。描く、実行、ステップ実行、import、exportと一応一通りできます[3]。下に画面のスクリーンショットが出ていますが、Shareという所からURIを作ることによって、ブラウザで描いたPietをURIにして共有することもできます。例えば、https://piet-editor.github.io/#code=H4sIAMg7XVgCA0tMTM9OS6nx9q8U4MT87Kr0lMTPT29gYAp0dBWRsAAAA@&width=6&height=4というURI……は長いので、短縮したものhttp://bit.ly/2inLPSZを辿ると、適当に描いた‘@’を出力するPietが表示されると思います。

アプリのURIを辿るとこんな感じの画面が出たりするはずです[4]。CSSをまだ全く書いていないので(あとデバッグ用の出力が下のほうに出ていますし)、まだUIがわかりづらいかもしれません。

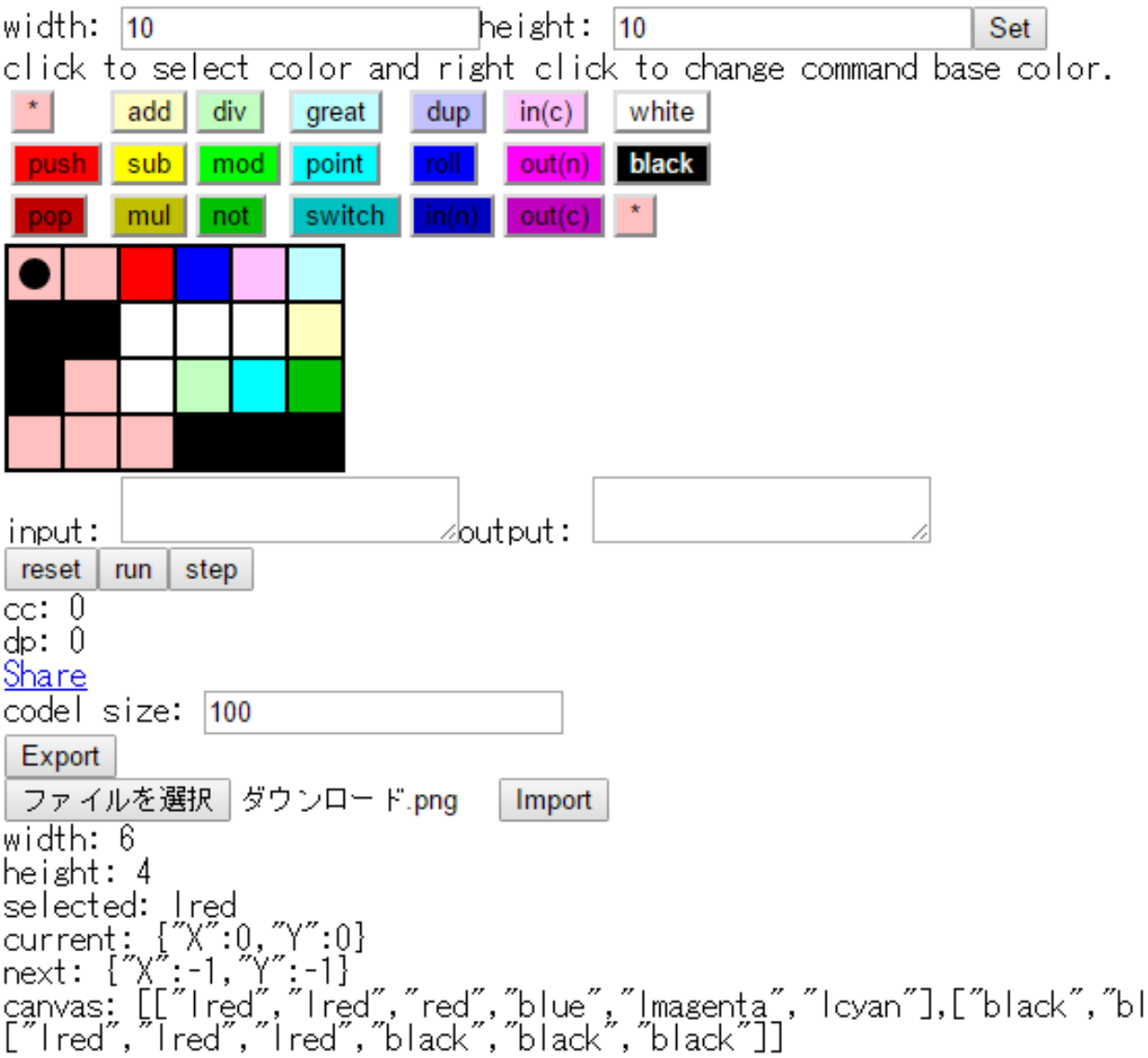
---

[1] Pietのエディタを作った話(http://www.slideshare.net/KMC_JP/piet-46068527) Pidet https://github.com/dnek/Pidet

[2] Muratam/UltraPiet https://github.com/Muratam/UltraPiet

[3] undoがまだできないので実装しないと……。

[4] 画面は開発中のものです。大きな変更が入る可能性がありますが、何も出ないとか明らかなバグみつけたら教えてください……。https://github.com/piet-editor/piet-editor/issues/new or なんかリプライででも



Pidet[5] を触っていたらなんとなくわかるような雰囲気にしたつもりですが[6]、ブラウザだけでPietに入門できてうれしい みたいな気持ちの人が現れるようにと作った点もあるので、そういう意味では今のUIはまだまだよくないですね。もうちょっとUIに説明入れないと……。

上の方の3*7のカラーパレットで色を選んで、その下のキャンバスにポチポチとドット(コーデル)を打っていくような感じです。

実装については一応Pietの本を名乗っているので詳しくは省きますが、React+es6で、piet-testutils[7]からインタプリタ部分を切り出して、piet-interpreter[8]というnpm moduleにしたりし

[5] https://github.com/dnek/Pidet

[6] Pidet触ったことが無くてもぜんぜん良いんですが、Pietそのものについては Pietのエディタを作った話(http://www.slideshare.net/KMC_JP/piet-46068527)の46ページぐらいまでの内容(Pietの言語仕様についての説明です)は前提にさせてもらいたいです……。そもそもPietの仕様が前提に無い人がこの本読んで嬉しいんだろうか……とか考え出すと、そもそも誰がこれを読む????? みたいなきもちになってくるのでよくないですね。

[7] https://github.com/nna774/piet-testutils

[8] https://github.com/nna774/piet-interpreter

て使い回したりしたので、意外とすぐにできたという感じです。

一応既にPietDev[9]のような物が存在していたりするのですが、『Pietのエディタを作った話』でも上げられていたような不満点があり、また、上記のように既存のコードを使い回せる部分が多いと思ったので作りました [10]。

アイコンであるhttp://bit.ly/2io3oCrや、さっきのhttp://bit.ly/2inLPSZを開いてポチポチしたりするなどして是非一度遊んでみてください (そして是非改善点やバグなどを見付けたら教えてください)。

[9] http://www.rapapaing.com/blog/?page_id=6

[10] 既存のコードに微妙なコーナーケースでのバグがあったりしてハマったりもしたのですが……

Chapter 2

画像のPiet化のメイキングについて

2.1　はじめに

　はじめましての人ははじめまして。そうでない人はこんにちは、murataです。のなさんのサークル「いっと☆わーくす」がコミケに当選したとのことで、今回も寄稿させてもらいます。

　今回の記事は、みんなが気になるであろうPietのメイキングについて書こうと思います。作成するPietは、「涼しい顔で締め切りを破る「すずしい」と出力する輿水幸子」です。更に難易度が上がりますが、Pietでは本来18色+白黒を使えるところを、色数を制限して5色+白で描こうと思います。この画像を描こうと思った経緯は、僕のはてなブログ (chy72.hatenablog.com)に 書いてありますので、合わせてお読み下さい。http://chy72.hatenablog.com/entry/2016/12/24/175722

2.2 元絵を描く

まずは適当に絵を描きます。この構図は、キルミーベイベーのやすなの有名なセリフ「もし感動しなかったら木の下に埋めて貰っても構わないよ」のパロディです。

2.3 Pietで扱いやすいように変換する

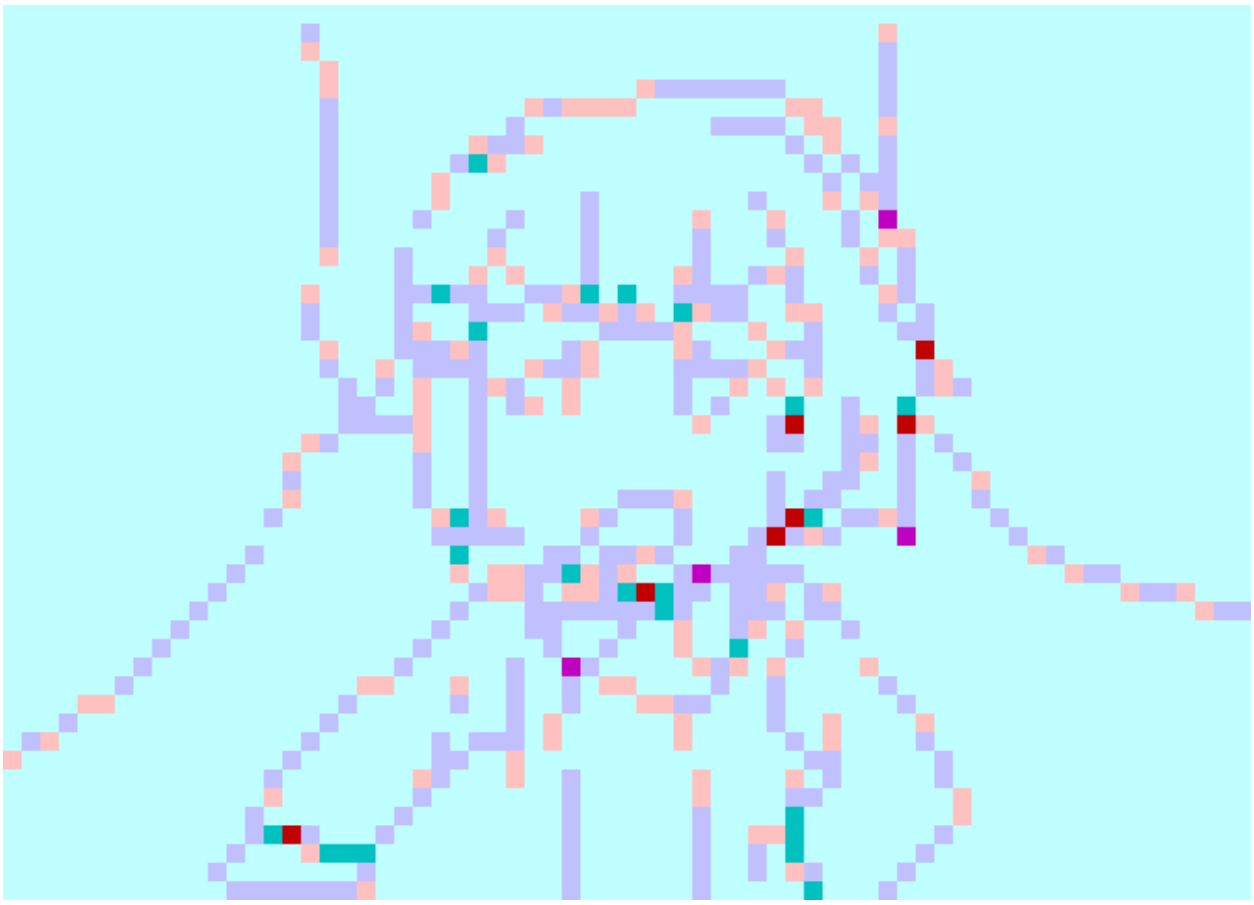
目的のPietに近いサイズに圧縮します。FireAlpacaなどのお絵かきソフトがあれば簡単に出来るはずです。

次に、とりあえずPietが扱える20色に変換します。今回は以下のpythonのコードを用います。

```
from PIL import Image
import sys
from colormath.color_objects import sRGBColor, LabColor
from colormath.color_conversions import convert_color
from colormath.color_diff import delta_e_cie2000

piet_colors = [
    "211", "221", "121", "122", "112", "212",
    "200", "220", "020", "022", "002", "202",
    "100", "110", "010", "011", "001", "101"
]

def piet(p):
    return [0, 192, 255][int(p)]

def calc_diff(r, g, b, rp, gp, bp):
    c1 = sRGBColor(r / 255.0, g / 255.0, b / 255.0)
    c2 = sRGBColor(piet(rp) / 255.0,
                   piet(gp) / 255.0,
                   piet(bp) / 255.0)
    c1 = convert_color(c1, LabColor)
```

```
    c2 = convert_color(c2, LabColor)
    delta = delta_e_cie2000(c1, c2)
    return delta

def to_piet_color(x, y, img):
    r, g, b = img.getpixel((x, y))
    pre_diff = 100000000
    decided_color = 0
    for (rp, gp, bp) in piet_colors:
        diff = calc_diff(r, g, b, rp, gp, bp)
        if diff < pre_diff:
            decided_color = (piet(rp), piet(gp), piet(bp))
            pre_diff = diff
    img.putpixel((x, y), decided_color)

if __name__ == "__main__":
    imgname = sys.argv[1]
    img = Image.open(imgname, 'r')
    for x in range(img.width):
        for y in range(img.height):
            to_piet_color(x, y, img)
    img.save("out.png")
```

colormathライブラリのdelta_e_cie2000 を使うと、人間の視覚的な色の差を計算することが出来るので、それを利用します。このコードを実行すると下の画像が得られます。



ここまでで原型ができたので、この画像を適当なPietのエディタを使用して見やすい形に 手動で変えます。

大体の原型がここまでで完成します。この画像が一番綺麗な画像となります。

2.4 幸子を5色で表す

今回生成する目的画像は、「すずしい」と出力する「有彩5色+白」のPiet画像です。理論上最低5色があればPietで実行できる命令は全て実行できるのですが、どの5色を使うのかを決める必要があります。

今回はC++でコードを書いてどのようになるかを計算しました。そのコード自体は前述の僕のはてなブログに書いてありますので省略します。

```
---- パターン1 ----
o o o x x x
o x x o x x
x x x x x x
---- パターン2 ----
o o x o x x
o x x x x x
x x o x x x
---- パターン3 ----
o o x x o x
```

```
o x x x x x
x x x o x x
---- パターン4 ----
o x x x x x
o o x o x x
x x o x x x
```

Pietでは18色(明度3段階(行)*色相6段階(列))を使うのですが、そのうちの5色を組み合わせて17命令を実行できるパターンを 探索したところ、この4パターンが条件に妥当すると判明しました。(実際には、この4パターンの遷移、上下左右反転を含みます） 今回は、幸子の肌を表す「薄い黄」、髪を表す「薄い紫」、服を表す「薄い赤」、輪郭を表す「濃い色」を使いたかったので、パターン1、つまり「薄い黄」「黄」「薄い赤」「薄い紫」「青」を用いることにしました。まずは先程描いた幸子をこの5色で描いてみます。(まだPiet的意味は持っていません)

2.5 Piet的意味を持たせる

これ以降、画像にPiet的意味を持たせるフェイズに入ります。Piet的意味を強引に入れていくので、これ以降生成される画像はPiet的意味を持つに従って劣化していきます。Piet的情報量は画像の綺麗さとトレードオフの関係なのです。

さて、どのように「すずしい」というPiet的意味を持たせましょうか。「すずしい」の各文字はUTF16の10進数表記で「12377,12378,12375,12356」となります。12377のような大きな数字をスタックに積む簡単な方法は 一番近い二乗の数字を利用することです。例えば、12356であれば、111 * 111 + 35 で表現できるので、111 をスタックに積み、コピーして掛け算して35を積んで足すだけで表現できます。111をスタックに積むには、111のサイズのカラーブロッ

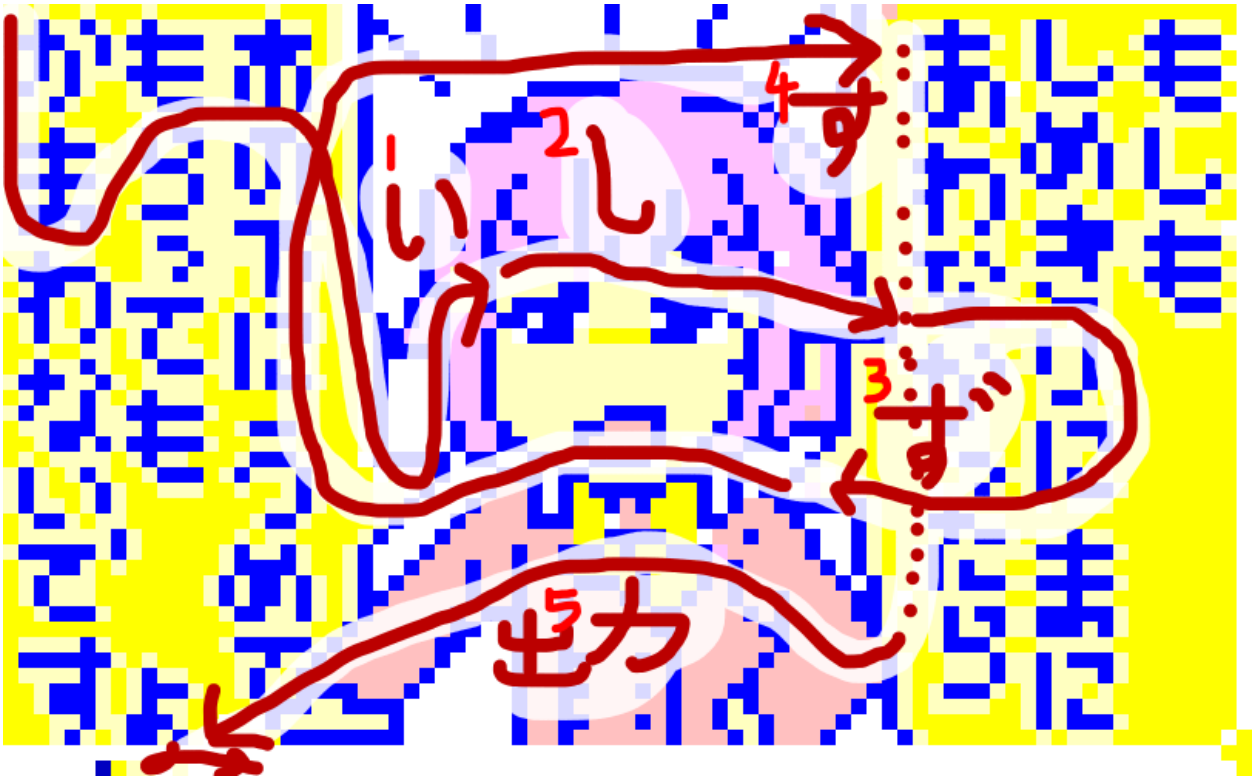
クが必要なので、今回の絵を鑑みると、セリフの部分で頑張ればいいということがなんとなく分かります。よって、方針としては、左のセリフ部分で 12356を作成し、コピーして差の分を加える…ということを繰り返すということが見えてきます。12356 を作成すれば、コピーして21を積んで足すだけで12375を作成することができます。

今回は黒を使えないということで方向転換がだいぶ難しいので、いっそポインターの赴くままに臨機応変にPietにしていきました。文字を出力するためには「黄」→「薄い赤」の遷移が必要で、それを不自然無く満たす場所は幸子のリボンであるので、そこを最後に 通るようにということを考えながら描きました。黒を使わずに終了するには、一面白の行と列が必要であり、それを不自然無くおけるところは画面の端なので、最後には画面の端に行けるようにということも考慮しながら描きました。コツとしては、白を上手く使って実行する命令を制御すること、不自然な色を使う必要が出てきた場合は、なるべく端っこで実行することで 絵全体としての品質を損なわないように意識することが挙げられます。

結果として、以下のようなPietを作成することができました。

https://nna774.net/piet/images/c91/10out4.png

移動の概略は、以下のような感じです。



2.6 最後に

元の絵を描いて、それを徐々にPietにしていくという方法を とると、ある程度簡単にきれいなPietを作成することができます。最低限のPietの知識は必要ですが、ぜひ皆さんも 画像をPiet化して裏の意味を持たせる遊びをしてみると楽しいと思います。この記事を読んでくださって、ありがとうございました。

Chapter 3

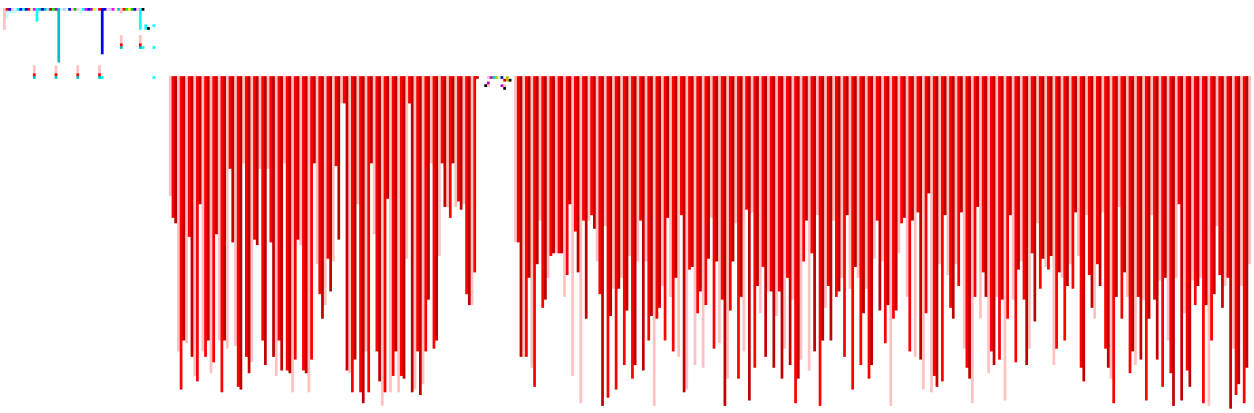
KMCのエイプリルpiet

KMCは今年のエイプリルフールでpietでwebサーバを書いて(描いて?)いました。4/1にはhttp://april-2016.kmc.gr.jp[1]でpietで書かれたHTTP serverが動いていて、それがnginxの下にぶらさがっていたようです[2]。

今回はどのようなコードが動いていたか、簡単に見て行きたいと思います。

使われていたサーバは以下のようなソースコードとなっていました。……とここにソースコードの画像をほどんど線のように見えるだろうなと予想しながらも貼ろうとしましたが、57286 x 153 という大きさのためか、TEXの制限? で貼ることができませんでした。https://raw.githubusercontent.com/kmc-jp/2016-aplil-fool/master/htmlserver.png[3] で見ることができます。

巨大すぎるので、最初のほうを切り出してきますと、このような感じです。



これだけでpietの自動生成に詳しい人は、だいたいどのように書かれたのか予想がつくかもしれません。pietの公式ページ? のサンプル[4]にあるAssembled Piet Codeや、http://www.toothycat.net/wiki/wiki.pl?MoonShadow/Pietなどが生成するpietに似ていますね。コードの生成を実際行なってるのはこのコードです。https://github.com/kmc-jp/2016-aplil-fool/blob/master/bin2piet.py

[1] 今はこのURIにアクセスすると、解説ブログへとリダイレクトされます。是非一度ご覧ください。

[2] ようです、というのは、これには私はかかわっていないからです(ここで気付きましたが、この記事はのなの二つ目の記事です。はじめにが無いので判り辛いですね)。

[3] https://github.com/kmc-jp/2016-aplil-fool/にレポジトリがあり、その中にこのコードも含まれています。

[4] http://www.dangermouse.net/esoteric/piet/samples.html

レポジトリには簡単な解説が含まれているので、その画像を下に引用しておきます。

補足しておくと、この解説では「pietは仕様でunicodeしか出力できないのでバイナリを出力するのは難しい」とありますが、これはout characterでunicodeで出力するという仕様が定まっている訳ではないので、処理系の問題でしかないように思います。

コードを見ると、最初にいくつかの制御部分があって、その後巨大なstringをpushしてそれを出す のような基本方針で描かれているようです。このpietの動作説明コードと思わしきrubyで書かれたコードもあります。https://github.com/kmc-jp/2016-aplil-fool/blob/master/main.rb

最初の制御部分は、branch.png[5]がそのまま埋めこまれているようで、以下のようなコードとなっています。

1. 標準入力の先頭4文字が‘GET ’でなければ400 bad requiestを返す部分に繋がる方向へジャンプ

2. 先頭4文字が‘GET ’であった時、続く2文字が‘/ ’であれば200 OKを返す部分へとジャンプ

3. そうではなかった場合、451 Unavailable For Legal Reasonsを返す方向へジャンプ

httpのリクエストが

```
GET / HTTP/1.1
Host: hogehoge
```

[5]https://github.com/kmc-jp/2016-aplil-fool/blob/master/branch.png

のような形であることを上手く利用して‘GET /’だけに応答する最小のコードが書かれているなぁ という感想です。あえて文句をつけるなら、GETされてるのが‘/’ではなかった時に451を返すことにはおそらくLegal Reasonsは無いであろうので、素直に404を返すべき という事ぐらいでしょうか。

続く赤の縞々の部分では、light red、red、dark redの三色で望む大きさの長さ下に延ばし、そこを横切ることでpushを繰り返しています。ここにbase64 encodedされた200.png[6]、400.png[7]、451.png[8]などや、これらをsrcとするimgタグの含まれたhtml、レスポンスコードなどが含まれています。

3.1 さいごに

と、ざっと軽くどのような動作をしているか書いてきましたが、いかがでしょうか。このように、このプログラムにおいてもpietの生成が行われています。pietの生成は意外と簡単なので、是非手で描くだけではなく一度挑戦してみてください。また、私の作った生成機、piet-automata[9]もよろしくお願いします。これについては、C88で出したペーパー[10]、C89で出した本などで解説してたりもするので、そちらもよろしくお願いします。

またもやpietの基礎を前提とした文章を書いてしまってすみません。この本の中でも何度も紹介していますが、「Pietのエディタを作った話」http://www.slideshare.net/KMC_JP/piet-46068527 には、pietの基礎についてもまとめられていますので、まだ読んだことがないという場合は、一度目を通して頂けると幸いです。

ここまで読んで頂いてありがとうございました。次回があればまたお会いしましょう。何かあれば奥付の連絡先にお気軽にご連絡下さい。

[6] https://github.com/kmc-jp/2016-aplil-fool/blob/master/200.png
[7] https://github.com/kmc-jp/2016-aplil-fool/blob/master/400.png
[8] https://github.com/kmc-jp/2016-aplil-fool/blob/master/451.png
[9] https://github.com/nna774/piet-automata
[10] https://nna774.net/piet/c88paper.pdf

あとがきの国

NoNameA 774　夏休み終了ぐらいから体調を崩していてギリギリまで出すか悩みましたが、なんとか形になりました。読んでくださってありがとうございます……。
　今回murataくんの記事とかは色についての言及が含まれているのに、カラーで刷れなかったので残念です。是非PDFで見てみてください。

murata　Pietの基礎知識を前提として書いてしまった感があるのですが頑張って解読してくれたら幸いです。

奥付

2016/12/29	初版発行
hash:	36fdbd1550f6ccdc7b9233c115dfecdec1cb0c54(の次)
著作・発行	NoNameA 774 (nonamea774@nnn77)
サークル	いっと☆わーくす!
メールアドレス	nonamea774@gmail.com
Web	https://nna774.net/
Twitter	@nonamea774
GPG Key	0x0C3E3AB2
fingerprint	674A 287A 21D2 2431 AD8F D328 AEF3 C3C7 0C3E 3AB2
Keybase.io	https://keybase.io/nona

今回表紙はMokoさん(@moko_oxygen)に描いて頂きました! ありがとうございました。

Pietのどうじんし 2

いっと☆わーくす