Teach Yourself KMC

独習KMC vol.3

京大マイコンクラブ　著

はじめに

本誌「独習 KMC vol.3」は京大マイコンクラブが編集する、復刊してから３＋α番目の部誌です。前回は C81、前々回は C80 とコミケに合わせて刊行してきました。今回の部誌では最近の KMC の活動をまとめた「KMC 近況」、そして Web フレームワークの特集記事を掲載しています。また、部員が何でも書きたいことを書く「部員のコラム」もあります。どうぞお楽しみください。

目次

KMC 近況

最近の KMC

jf712

はじめに

こんにちは。KMC 第 35 代会長の jf712 です。

今回、「KMC の近況を書いてくれ」と頼まれたので、前回部誌を発行して以降の、KMC の活動を簡単に月ごとにまとめ、読者の皆さんにお送りします。

1 月〜2 月

年明けを迎え、新たなる一歩を踏み出した KMC

コミケ (C81) を終え、今年も恙なく、KMC の一年は始まりました。……年賀状の数？ 数はどうでもいいのですよ。質が大事です。

ある上回生の企画で、今年の年初めの例会では書初めが行われました。部員一同それぞれが今年の志などを書き綴り、一年の始まりとして気を引き締めていました。

部員たちの書初めは集められ、どんど焼きの際に焼かれる……予定だったのですが、つい先日、部室の物置として使われている場所に、新聞紙で包まれ放置されているのを確認した気がしなくもありません。

2 月は 2 月で、12 日にボードゲーム大会が開かれました。また、15 日には前日が 14 日だったことを受けて「にぼしの日パーティー」が行われ、部員達は大量のお菓子を食べました。なかにはお菓子を作ってくる部員もいて、さすが KMC だなぁと思いました。

遊んでばかりではありません。12 日には今年最初のコーディング大海[*1]が、17 日〜19 日にはサインペン合宿と称し、ひたすら絵の修行をする合宿が開かれました。

ちなみに大学の試験期間は 2 月前半です。

3 月

春合宿、そして春の足音とともに近づいてくる新勧への準備

[*1] 誤字ではない。「大海」である。

図 1　嬉しさのあまり……。

KMC 部員の 3 月は、春合宿の準備（スライドなど）で始まる……のだと思います[*2]。KMC は 3 月のどこかで「春合宿」なる数日間の合宿を行い、部員のさまざまな分野の知識やボードゲームやお酒の知識を養っています[*3]。今年は、映画の上映会もあったようです。

図 2　今年も行方不明者等はいなかった。

春合宿から帰ってきた一回生を待つものは、そう、新勧の準備である。新入生を迎えるべく用意をしなければいけない。毎年恒例の看板立てや[*4]、部室の掃除、ビラの作成を行ったりして新勧に備えました。ビラは新勧の中心となる一回生と、二回生の一部が作ったのですが、人数が多かった

[*2] スライドの準備が出来ていない者は、合宿中皆が遊んでいる横で悲しみながらスライドを仕上げないといけない……。

[*3] お酒の知識を養うのは全員 20 歳以上である。

[*4] 毎年、東大路通りにずらりと立て看板が並ぶ。

ので、なんと 14 種類もビラが出来てしまいました。結局、新勧期を通して印刷したビラの総数は約 50000 枚にもなりました[*5]。

図 3　人気のあったビラ。

そうこうしているうちに新しい年度が始まりました。

4 月〜5 月

めくるめく新勧、新入生の数々

今年の新勧は、4/2 にいきなり 5 人も入部したことを皮切りに、慌ただしく始まりました。週 2 回で開く説明会には多くの新入生が足を運んでくれるのですが、毎回説明会用のスライドのネタを考えるのも一苦労でした[*6]。各新入生の一週間の動きが定まってくる 4 月の下旬からは、もはや KMC の名物となっている「新入生プロジェクト」も始まりました。

[*5]ちなみに、京都大学の新入生の総数は約 3000 人である。
[*6]何回もやっていると飽きる……。

「C#でゲームを作ろう 2012」「DTM 練習会 2012」「イラスト品評会 2012」「Rails で Web サービス 2012」「ICPC 勉強会 2012」などに加え、今年はプロジェクトとして「3DCG 武闘会」「ブラウザ幻想.js」など新しい方向性の勉強会も開かれました[*7]。また「経営シミュレーションゲームを作ろう」といった、数人で一つのゲームを作り上げるというプロジェクトも動いています。メンバーたちが作り上げた経営シミュレーションゲームが、この部誌とともに頒布する CD に収録される予定です。ぜひお楽しみください[*8]。

他にも、一日でゲーム作りを体験しよう、という趣旨のもと開かれた「ラピッドゲームコーディング祭り」や、上回生が作ったカレーを新入生上回生入り混じっておいしくいただくという「KMC ごはん」など、たくさんの新勧イベントが行われました。

その結果、去年の入部人数 32 人を上回る 37 人もの新入部員を我々は獲得しました。これだけの人数が入ってくれて、新勧を行った身としてはうれしい限りです。新歓コンパの人数が毎年増えていくのは未来が感じられてよいですね。

図 4　新歓コンパの様子。全体の 1/3 ぐらいしか写っていない。

新勧以外の話題では、KMC 部員のチームが ICPC（国際的な競技プログラミングの大会）の世界大会へ出場権を勝ち取りました。残念ながらメダル獲得には至りませんでしたが、24 位と健闘されたようです。素晴らしい。

6 月〜7 月

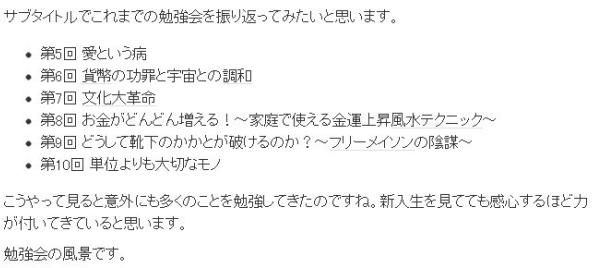
この期間で話題になるようなこと……は、特にないですね。強いて言うなら、部員たちの間で風邪が流行ってしまったことでしょうか。過去に「KMC パンデミック」と言われたサークル内での風邪の流行があったようですが、今年はその再来だったのでしょうか。かくいう私も風邪にや

[*7] こう書いてしまうと過去形だが、現在進行形で続いていますですよ！

[*8] 経営シミュレーションゲームのプロジェクトリーダーにプレッシャーを与えているわけでは、断じてない。

られてしまい、6 月最終週ずっと学校を休んで家で倒れている、という状態になってしまいました……[*9]。

各種勉強会は順調に進んでいるようです。ためしに新入生プロジェクトの一つ、「Rails で Web サービス 2012」のブログ[*10]を見てみましょう。


サブタイトルでこれまでの勉強会を振り返ってみたいと思います。

- 第5回 愛という病
- 第6回 貨幣の功罪と宇宙との調和
- 第7回 文化大革命
- 第8回 お金がどんどん増える！～家庭で使える金運上昇風水テクニック～
- 第9回 どうして靴下のかかとが破けるのか？～フリーメイソンの陰謀～
- 第10回 単位よりも大切なモノ

こうやって見ると意外にも多くのことを勉強してきたのですね。新入生を見てても感心するほど力が付いてきていると思います。

勉強会の風景です。

図 5　なにやら枠に縛られない勉強会のようである。

なんやかんやで 6 月、7 月は過ぎていきました。……まぁこの記事を執筆しているのは 7 月頭なので、これから何か起こるのかも知れませんが。

おわりに

さて、これでそろそろ読者の皆さんともお別れです。我々 KMC は、これからも切磋琢磨し合って成長していくことでしょう。そんな KMC という存在を少しでもこの部誌で知っていただけたら幸いです。

それではいつかまた逢う日まで。

KMC 第 35 代会長　jf712

[*9]実は、この記事を病み上がりで書いています。

[*10]「KMC 活動報告 blog」`http://d.hatena.ne.jp/kmc-log/` 勉強会やイベントをブログの形で簡単に報告しています。

新入生プロジェクト紹介

プロジェクト担当者一同

新入生プロジェクトとは KMC が新歓期に新入生向けとして行なっているプロジェクトのことです。新歓の雰囲気を感じ取っていただければと思います。

C#でゲームを作ろう 2012

C#でゲームを作ろう 2012 担当の lunan です。このプロジェクトは、プログラミングをやったことのない人を対象として、C#の基礎から始め、最後は一人一つ XNA を使ってゲームを作ってもらおうというものです。

制作したゲームは夏コミの CD に収録する予定です（この部誌が手に取られている時点では、無事に完成したかどうか判明してるはずですが)。このプロジェクトに参加した新入生が将来面白いゲームを作ってくれることを願っています。

lunan

イラスト品評会 2012

イラスト品評会とは、一言で言うと「叩かれて強くなること」をコンセプトとする会です。自分の描いた絵が皆にツッコまれて欠点が露わに！ つらい！ しかし、そうやって自分の苦手な所と向き合わなければ弱点は克服できないのです。

そんなストイックな会の予定でしたが、「誰でもツッコミの荒波にもまれれば自然と強くなるだろう」などと軽い気持ちでうっかり＜初心者歓迎＞を謳ってしまったため、ほとんど絵を描いたことのない人もやってきました。いわば、「これから空手を始めてみようと思って近所の道場を覗いてみたらいきなり試合をさせられた」ような状態です。わけもわからずボコボコに殴られて、理不尽な思いをしたことでしょう。僕が間違ってました。

そこで第 4 回からは方針を転換しました。品評会の代わりに講座＋実戦練習を行い、その場で基礎的な知識・技術を身につけていこうというものです。題するなら「イラスト練習会」になるのでしょうか。初心者にはもちろん、講座を準備する僕も含めてある程度絵を描いている人にも実りのあるものになりました。お互いにとって良いきっかけになったと言えるでしょう。

とはいえ、いつも課題がないのは寂しいものです。なので、たまに品評会もします。ただし、課題はテーマだけ与えて自由に描いてきてもらうのではなく、模写や塗り絵など、もう少しはっきり

としたガイドラインのあるものにしました。ツッコむ側・ツッコまれる側ともに、幾分やりやすかったように思います。

当初は自分から見ても他人から見ても僕が主催者というのが不安だったものですが、そんなこんなでイラスト品評会も第 8 回。主催者なのに遅れて来たり、講座のスライドが未完成だったりと色々失敗してばっかりですが、なんとかやっています。

ちなみに後期はもう少しコアなテーマで講座をやりたいなと思っています。作品を作ってツッコみ合う、本来のイラスト品評会もどんどんやっていきたいですね。

madaragi

DTM 練習会 2012

こんにちは。DTM 練習会のリーダーを務め、日々精進に励むべく音楽聴いてばっかりの KMC 2 回生、jf712 です。……ちょっと風邪をこじらせて苦しんでいますが。

DTM 練習会は音楽の練習場です。毎回、課題として何かテーマを決め、各自がそのテーマに沿った曲を創って来ます。あとは、部室でその曲を発表し皆で品評を行うという形式です。

勉強会の初めのうちは、DTM 初心者に合わせて MIDI 音楽編集ソフトを紹介したり、ドラムのついてない曲にドラムをつけてきたりといった、いわば「PC での作曲の仕方」について重点的にやりました。そしてちょうど次回から、各自が己の力で一つの曲を創ってくる、というフェイズに入りました。まさにこの練習会が真の姿を現すときです[*1]。

今年のメンバーには、音楽経験者も複数人参加しています。実際の楽器を弾く人たちの視点からの意見も聞けることでしょう。楽しみですね。

それではみなさん、よい Musix Life を。

jf712

Rails で Web サービス 2012

Rails で Web サービス 2012 担当の uiureo です。この勉強会は、Ruby on Rails を使って Web サービスを作るという新入生プロジェクトです。参加者が自分の力で好きな Web アプリケーションを開発できるようになるのを目標にしています。

新入生に一からプログラミングを教える新入生プロジェクトなので、Ruby でプログラミング入門することから始めました。ひと通り Ruby の基本的な使い方を習得した後、Ruby on Rails を用いた Web アプリケーション開発の勉強を始めました。現在は『Rails によるアジャイル Web アプリケーション開発 第 4 版』を読み進めるという形で勉強会を進めています。

初めてのプログラミングに戸惑う新入生も多かったようですが、実際にコードを書く練習を繰り

[*1] 発表曲に対し、違和感を感じた部分にツッコミを入れる……。いわば殴り合い……ではないですね。はい。

返すことで、だんだんとプログラムを書けるようになってきたようです。Web サービスを作れるようになるにはまだまだ道のりは長いですが、少しずつ頑張っていきたいと思います。

uiureo

ICPC 勉強会

ICPC 勉強会、主催の natsugiri です。この勉強会では競技プログラミングの問題をコンテスト形式で解き、ACM-国際大学対抗プログラミングコンテスト (ACM-ICPC) の練習をしています。競技プログラミングの面白さはコンテストの順位や問題の正解数によって、自分のプログラミングスキルを数値で知ることができ、成長を実感できる点にあると思います。

プログラミングの経験が少ない新入生が多数いる中、環境導入が済んだら早速問題に挑戦するというハードな始まりでした。それでも週一回のペースで続け、KMC から新入生を含む多くの部員が今年 7 月の ACM-ICPC 国内予選に参加しました。更なるプログラミングスキルの向上のために、これからも勉強会を続けていきます。

natsugiri

経営シミュレーションゲームを作ろう

経営シミュレーションゲームを作ろう担当の mamopm です。このプロジェクトは、新入生に実際に多人数でのゲーム製作を体験してもらうことを主な目的にしています。（もちろん、ゲームを作ること自体も目的の一つです）

基本的なプログラミングなどの勉強は他のプロジェクトで行ってもらって、基幹部分は私が作ったものをいじりながらゲーム製作をしていくという形をとっています。

執筆地点ではまだ話し合い程度しかしておりませんが、今後実際にコードを書いてもらう等の具体的な製作に移っていこうかと思います。

成果物はゲーム CD に収録される予定ですので、CD を入手された方はぜひ遊んでください。

mamopm

3DCG 武闘会

Sup! 3DCG 武闘会の講座担当、TJ です。

3DCG 武闘会とは、3D モデリングやアニメーションの技術を学び、匠の技を鍛え上げて三次元世界の神になろう！……という企画です（だった気がします）。

開始当初は全くの未経験者や趣味で少し触った程度の初心者が多かったこのプロジェクトですが、3D モデリング、テクスチャマッピング、ボーンアニメーションといった様々な分野を学んで

いくうち参加者たちの顔つきも逞しくなってきたのではないかと思います[*2]。夏休み明けには今までの知識の応用をしたり、専門的なテクニックを拾ったりしながら、3D ゲームや本格的な動画制作などに踏み込んでいきたいと思っています。

TJ

ブラウザ幻想.js

ブラウザ幻想.js 担当の kirita です。3 月の終わり頃、担当者が『共同幻想論』を読んでいたのが名前の由来です。この勉強会では、ウェブブラウザの拡張機能を作ることを目標として、JavaScript や jQuery についての解説をしています。最終的には参加者全員が自分で何か 1 つ拡張機能を自作してみることまでができればいいなと思っています。

ブラウザの拡張機能というものに新入生が興味をもってくれるのか、不安なところもありましたが、毎回新入生から 3 回生まで、あわせて 5 人くらいが来てくれています。また、サークルの JavaScript に詳しい人も時々監督的な立場で参加してくださっていて、担当者の足りない知識にツッコミを入れていただいています。この記事を書いている時点では知識の解説は大体終わって、自分の欲しい拡張機能を作る段階に入っています。それらが完成したら、どこかで公開できたらいいなあ、とも考えています。

kirita

[*2]「まったく」「簡」「単だ！」

Web フレームワーク特集

Python/Flask で学ぶ Web アプリ入門

@ichyo

初めに

この記事では、Flask を使って簡単な Web アプリを作る方法について説明しています。Web アプリについてあまり知らない人のために基本的なことを中心に書きました。言語は Python を使っていますが、 Python を知らなくてもある程度理解できるような内容になっていると思います。

簡単なアプリ

さて、ここからは、実際にアプリケーションを作っていきましょう。現在の日時と日付を計算して表示するプログラムを作ります。まず、Flask をインストールしましょう。端末で次のコマンドを実行してください。

```
$ sudo pip install Flask
```

pip がインストールされていないときは、次のコマンドを実行しましょう。

```
$ sudo easy_install install pip
```

インストールは終わりましたか？ ここからはコードを書いていきます。

```
from flask import Flask
from datetime import datetime
app = Flask(__name__)
```

1,2 行目では必要なクラスをインポートし、3 行目でそのクラスのインスタンスを作っています。ここはあまり重要ではないので、おまじないのようなものだと思って先に進みましょう。

つぎに日付を表示する関数を作ります。

```
@app.route('/date')
def print_date():
    now = datetime.now()
    return "%d/%02d/%02d %02d:%02d:%02d" % (now.year, now.month,
                      now.day, now.hour, now.minute, now.second)
```

関数の前に`@app.route()` を置いて、その引数に、関数を呼び出す URL を指定します。このプログラムでは、“`/date`” にアクセスした時に `print_date()` を呼び出すように記述しています。関数の中では、先ほどインポートした `datetime` の `now` 関数を利用して時刻を取得し、その値を文字列に埋め込んで返しています。

最後に `run()` 関数でローカルサーバーを走らせます。

```
if __name__ == '__main__':
    app.run()
```

`__name__`は、直接 python インタプリタから実行された時だけ`'__main__'` が入ります。 import されたときには `app.run()` は実行されません。

ここまで書けたら、一度保存してスクリプトを実行してみましょう。ファイル名は “flask.py” 以外ならなんでも OK です。

```
$ python date.py
* Running on http://127.0.0.1:5000/
```

上のようなメッセージが表示されたら正しく動いています。手元のブラウザで `http://127.0.0.1:5000/date` にアクセスしてみましょう。現在の日時と時刻が表示されたら無事成功です！

入力を受け取る

私たちがインターネットを利用するとき、ブラウザは主に次の 2 つのことを行います。

- 「○○がほしい」といったリクエストをサーバーに送る
- サーバーからのレスポンスを受け取って表示する

実際にサーバーに送られるメッセージを覗いてみましょう。`http://en.wikipedia.org/` で KMC を検索するとサーバーにこのようなメッセージが送られます。

```
GET /w/api.php?format=json&action=opensearch&search=KMC HTTP/1.1
Host: en.wikipedia.org
Connection: keep-alive
...
...
```

1 行目にはメソッド名、URL、HTTP のバージョンが書かれます。2 行目以降にはホスト名や、接続状況などが書かれています。1 行目を見ると GET メソッドが使われていることがわかります。GET はページなどを取得する際に使われるメソッドで、最も使用される頻度が多いです。次に URL を見てみましょう。なにやら`?`や`&`などの見慣れない記号が使われています。実は、`?`以降の文字列はクエリと呼ばれるもので、サーバー上のプログラムに値を渡すために使われます。クエリは python の dictionary に似たデータ構造で、$key = value$ を`&`で繋げた形式をしています。もう一つ、POST メソッドの例を見てみましょう。

```
POST /w/api.php HTTP/1.1
...
...
format=json
action=opensearch
search=KMC
```

GET メソッドでは、クエリは URL に繋げて書かれますが、URL には文字数の制限があるので、クエリの文字数も制限されます。一方、POST メソッドでは、サーバーに渡す値を、URL とは別の部分に書かれるので、長い文章を送信する時などに使われます。

ここまでリクエストの例を二つ見てきました。さて、はじめのプログラムは、クライアントからの入力を受け取らないものでした。次は、リクエストの内容によって動的にレスポンスが変わるようなプログラムを作ってみましょう。プログラムの内容は、日付を受け取って、その曜日を表示するものとします。

まずは日付を受け取って、曜日を返す関数を作りましょう。

```
def get_weekday(y, m, d):
    weekday_id = datetime(year=y, month=m, day=d).weekday()
    weekday_name = ["Monday", "Tuesday", "Wednesday", "Thirsday",
                    "Friday", "Saturday", "Sunday"]
    weekday = weekday_name[weekday_id]
    return "%04d/%02d/%02d is %s" % (y, m, d, weekday)
```

2 行目では datetime クラスの weekday() 関数を使って曜日の id を取得します。返り値は、月曜日が 0、火曜日が 1、というように続くので、それに合わせて曜日の名前のリストを作り、それらを組み合わせて、文字列を生成して返します。

さて、この関数を使って、日付のリクエストから曜日のレスポンスを返す関数を書いてみましょう。実装の仕方はいくつかあります。

1 つ目は、/weekday1/2011/3/21 のような形式の URL から値を受け取る方法です。

```
@app.route('/weekday1/<int:year>/<int:month>/<int:day>')
def print_weekday1(year, month, day):
    return get_weekday(year, month, day)
```

@app.route() で、<*variable*>を使うと、その部分の URL の文字列を関数に渡すことができます。<int:*variable*>, <float:*variable*>のようにすると、それぞれの型に変換して引き渡されます。2 つ目は URL 上のクエリを使った方法です。クエリにアクセスするには request クラスをインポートする必要があります。

```
from flask import request
@app.route('/weekday2')
def print_weekday2():
    year = request.args.get('year', '1')
    month = request.args.get('month', '1')
    day = request.args.get('day', '1')
    return get_weekday(int(year), int(month), int(day))
```

request.args.get('key', 'default') はクエリから'key' の値を探して返します。'key' が存在しないときは'default' が返されます。 3 つ目は POST メソッドによるリクエストを受け取

る方法です。

```
@app.route('/weekday3', methods=['GET', 'POST'])
def print_weekday3():
    year = request.form.get('year', '')
    month = request.form.get('month', '')
    day = request.form.get('day', '')
    return get_weekday(int(year), int(month), int(day))
```

ここまで完成したら、保存してプログラムを実行してみましょう。http://127.0.0.1:5000/weekday1/2012/07/01 と http://127.0.0.1:5000/weekday2?year=2012&month=07&day=01 に正常にアクセスできたら成功です。さて、3 番目の関数を呼び出すにはどうすればよいのでしょうか。そのためには HTML でフォームを作る必要があります。

テンプレートエンジン

今までは HTML ではない生のテキストを出力していました。ここからは HTML を出力する方法を紹介します。まずは print_weekday3() 関数に入力を渡すためのフォームを HTML で書きます。

```
<html><head>
<meta http-equiv="content-type" content="text/html;charset=utf-8"></head>
<body>
<form action="/weekday3" method="post">
<p>year: <input type="text" name="year" size="10" value=""></p>
<p>month: <input type="text" name="month" size="10" value=""></p>
<p>day: <input type="text" name="day" size="10" value=""></p>
<p><input type="submit" value="submit"></p>
</form></body></html>
```

ここにテンプレートエンジンという機能を使って、プログラムの値を埋め込んでいきます。

```
...
...
```

```
<p>year: <input type="text" name="year" size="10" value={{ year }}></p>
<p>month: <input type="text" name="month" size="10" value={{ month }}></p>
<p>day: <input type="text" name="day" size="10" value={{ day }}></p>
<p><input type="submit" value="submit"></p>
{{ message }}
</form></body></html>
```

これを templates/weekday.html に保存しましょう。最終的に関数はこのような形になります。

```
@app.route('/weekday3', methods=['GET', 'POST'])
def print_weekday3():
    year = request.form.get('year', '')
    month = request.form.get('month', '')
    day = request.form.get('day', '')
    message = ''
    if year and month and day:
      message = get_weekday(int(year), int(month), int(day))
    return render_template('weekday.html',
        year=year, month=month, day=day, message=message)
```

`render_template()` 関数がテンプレートエンジンの役割を果たしています、第一引数に置換する html ファイルの名前、第二引数以降は、名前付き引数で変数の値を渡しています。テンプレートエンジンでは、{{ *variable* }} で *variable* の値を埋め込むほかにも、for 文や if 文で条件分岐や繰り返しをすることも可能です。

`http://127.0.0.1:5000/weekday3` にアクセスして、フォームに値を入れて送信してみましょう。曜日が表示できたら無事成功です。

まとめ・参考資料など

Flask は最小限の機能で構成されたフレームワークなので、学習コストが少なく、ドキュメントも充実しているので小規模のアプリを作るときにはおすすめです。Flask について書かれたサイトには以下のようなものがあります。

- Welcome — Flask (A Python Microframework) [*1]

[*1] `http://flask.pocoo.org/`

- Flask へ ようこそ - Flask v0.5.1 documentation [*2]
- Flask を始めるときの参考サイト - RENAISSANCE [*3]

今回はページ数の都合で、O/R マッパーなど Web アプリケーションフレームワークの基本的な要素を説明できませんでした。それらについては、次の本が実装レベルから詳しく説明しています。筆者のご厚意により、期間限定で PDF が無料公開されています（2012/07/01 現在）。

- 『みんなで Python Web アプリ編 [*4]』

[*2] http://a2c.bitbucket.org/flask/
[*3] http://d.hatena.ne.jp/Johan511/20110326/1301099330
[*4] https://lindoc.jp/docs/1163

Clack による疎結合な Web アプリケーションの作り方

kirita

Clack[*1] とは Web アプリケーションのための統一的な API を提供する Common Lisp のライブラリです。この記事では Clack の仕組みを解説しながら疎結合な開発がどういうものであるかということについて述べていきたいと思います。

環境構築

Clack に限らず Common Lisp のライブラリは Quicklisp を使ってインストールすると便利です。Quicklisp を導入するためには、まず `http://beta.quicklisp.org/quicklisp.lisp` をダウンロードし、それを読み込みます。ここでは Clozure CL を処理系の例としますが、SBCL などでも問題なく動作します。

```
$ wget http://beta.quicklisp.org/quicklisp.lisp
$ ccl --load quicklisp.lisp
```

このようにコマンドを打つと ccl のプロンプトに入るのでそこで

```
? (quicklisp-quickstart:install)
```

と評価すると Quicklisp のインストールが始まります。最後に、処理系を起動するとき Quicklisp を自動的にロードするために

[*1] `http://clacklisp.org/`

```
? (ql:add-to-init-file)
```

とすれば終わりです。

Quicklisp の使い方

Quicklisp は他の言語で言えば Ruby の RubyGems とか Perl の CPAN といったものに相当します。RubyGems でライブラリをインストールするときは端末からコマンドを叩きますが、Quicklisp はインストールなども S 式として評価します。基本的には `ql:system-apropos` と `ql:quickload` の 2 つの関数を覚えればいいでしょう。`ql:system-apropos` は名前からライブラリを検索する関数です。

```
? (ql:system-apropos "ppcre")
#<SYSTEM arnesi.cl-ppcre-extras / arnesi-20101006-darcs / quickli...
#<SYSTEM cl-ppcre / cl-ppcre-2.0.3 / quicklisp 2012-05-20>
#<SYSTEM cl-ppcre-template / cl-unification-20120208-cvs / quickl...
#<SYSTEM cl-ppcre-test / cl-ppcre-2.0.3 / quicklisp 2012-05-20>
#<SYSTEM cl-ppcre-unicode / cl-ppcre-2.0.3 / quicklisp 2012-05-20...
#<SYSTEM parser-combinators-cl-ppcre / cl-parser-combinators-2012...
```

このようにその名前を含むライブラリの一覧が表示されます。ライブラリをインストールするときは `ql:quickload` を使います。

```
? (ql:quickload :cl-ppcre)
```

筆者の環境では `~/quicklisp/dists/quicklisp/software/` 以下にインストールしたライブラリのファイルが置かれています。

SLIME の導入

Common Lisp で開発をするとき、SLIME という Emacs 上の開発環境を使うと非常に便利です[*2]。インストールにはまず公式サイト[*3]から最新版をダウンロードしてください。それを解凍したら Emacs の設定ファイルに次のコードを追加します。

```
(setq inferior-lisp-program "ccl")  ;; 使用する lisp の処理系によって変更する
(add-to-list 'load-path "/path/to/slime/")
(require 'slime)
(setq slime-net-coding-system 'utf-8-unix)
(slime-setup '(slime-repl slime-fancy slime-banner))
```

4 行目はそれぞれの環境によって euc-jp-unix などに変えてください。設定ができたら、`M-x slime` で SLIME が起動します。ここで

```
CL-USER> (ql:quickload :clack)
```

を評価することで Clack をインストールすることができます。

Clack による最小の Web アプリケーション

Clack をロードしたら、次のサンプルコードを評価してみましょう。

```
(defun app (env)
  '(200
    (:content-type "text/plain")
    ("Hello, World")))

(clack:clackup #'app)
```

[*2] Emacs のバージョンが 24 以降であれば Emacs Lisp のパッケージ管理システムが標準で使えるため、そこから SLIME をインストールすると楽ですが、最新版でない可能性があるので注意してください。

[*3] `http://common-lisp.net/project/slime/`

２つ目の S 式を評価すると Web サーバが起動します。次のように clack から clackup をインポートしてもいいでしょう。

```
(import '(clack:clackup))
(clackup #'app)
```

そしてブラウザから http://localhost:5000/ にアクセスします。ここで Hello World と表示されれば成功です。

基本的には、Web アプリケーションはリクエストを受け取りレスポンスを返すものです。レスポンスはステータス、ヘッダ、及び本文の３つから構成されます。Clack ではレスポンスはこの３つのリストとして表現します。app の引数 env がリクエストを表すものですが、もちろん文字列がそのまま入ってるわけではありません。app の定義を変更して env の中身を見てみましょう。

```
(defun app (env)
  (print env)
  '(200
    (:content-type "text/plain")
    ("Hello, World")))
```

このようにして Web サーバを起動するとリクエストがあるたびに env が出力されます。たとえば http://localhost:5000/blog/archive?id=10 にアクセスすると

```
(:REQUEST-METHOD :GET
 :SCRIPT-NAME ""
 :PATH-INFO "/blog/archive"
 :SERVER-NAME "localhost"
 :SERVER-PORT 5000
 :REQUEST-URI "/blog/archive?id=10"
 :QUERY-STRING "id=10"
 :REMOTE-ADDR "127.0.0.1"
 :REMOTE-PORT 51329
 :HTTP-HOST "localhost:5000"
 :HTTP-USER-AGENT "Mozilla/5.0 (X11; Linux i686) ..."
```

```
 ...)
```

SLIME 上にこのように出力されます。見てわかる通り、env は属性リストです。属性リストの要素を取得するためには getf を使います。app を次のように定義すればアクセスしてきたユーザーエージェントがブラウザ上に表示されます。

```
(defun app (env)
  `(200
    (content-type "text/plain")
    (,(getf env :http-user-agent))))
```

アプリケーション

clackup には今まで関数を与えていましたが、別のオブジェクトを与えることもできます。次に例を挙げます。

```
(import '(clack:<component> clack:call clack:clackup))

(defclass <echo-app> (<component>) ())

(defmethod call ((this <echo-app>) env)
  `(200
    (:content-type "text/plain")
    (,(getf env :request-uri))))

(clackup (make-instance '<echo-app>))
```

これらの S 式を評価して Web サーバを起動し、ブラウザから http://localhost:5000/hello/hello にアクセスすると、/hello/hello が表示されます。

コードを見てみましょう。1 行目は<component>を継承した<echo-app>というクラスを定義しています。次に<echo-app>に対して call というメソッドを定義しています。最後に<echo-app>のインスタンスを作りそれを clackup に与えています。clackup には関数だけで

なく<component>を継承したクラスのインスタンスも与えることができるのです。

これくらいでは関数を使ったほうが簡単ですが、次の例を考えてみます。

```
(clackup
 (make-instance 'clack.app.directory:<clack-app-directory>
    :root #p"/home/kirita/public_html/"))
```

Clack.App.Directory は Clack に標準で付属するアプリケーションで、:root で指定したパス以下のファイルをブラウザから見ることができます。このようにインスタンスを作るとき何らかの引数を取るような場合にはクラスを使ったほうが便利です。また、クラスを使うことによって、継承など Common Lisp のオブジェクト指向システムの恩恵を受けることができます。

ミドルウェア

関数もしくは<component>を継承したクラスを作成し clackup に与えることで大抵の Web アプリケーションは作れます。しかし、何も考えずにコードを書き始めても、産み出されるのは様々な機能が入り混じった再利用性の低いゴミの山でしょう。見通しよく開発するための何らかの機構が必要です。

Clack の**ミドルウェア**はそういった要求を解決するための強力で柔軟なシステムです。まず、アプリケーションはリクエストを受け取ってレスポンスを返すものでした。ミドルウェアはアプリケーションをラップし、リクエストとレスポンスを加工したり、その他の挙動に変更を加えたりします。

次のサンプルはアプリケーションがリクエストを処理し、レスポンスを返すまでの時間を計測し出力するものです[*4]。

```
(import '(clack:<middleware>
          clack:call
          clack:call-next))

(import '(local-time:now
          local-time:timestamp-difference))

(defclass <time-mw> (<middleware>) ())
```

[*4] ここで使われている local-time は日付を扱うライブラリで、これも QuickLisp でインストールすることができます。

```
(defmethod call ((this <sample-mw>) env)
  (let* ((before (now))
         (response (call-next this env)))
    (format t "[~A]~%" (timestamp-difference (now) before))
    response))

(defun app (env)
  (declare (ignore env))
  '(200
    (:content-type "text/plain")
    ("Hello, Clack!")))

(clackup
 (wrap (make-instance '<time-mw>) #'app))
```

ミドルウェアは<middleware>を継承したクラスです。call はアプリケーションと同じですが、call-next に注目してください。これはこのミドルウェアにラップされているアプリケーション[*5]を呼び出すものです。<time-mw>は call-next する前の時刻と返ってきてからの時刻を単純に引き算し、出力するミドルウェアです。

ミドルウェアは何層にも重ねて使うことができます。つまり、ミドルウェアでミドルウェアをラップするのです。

```
(clackup (wrap mw3 (wrap mw2 (wrap mw1 #'app))))
```

しかし、このような書き方はあまり見やすくないですね。Clack.Builder を使いましょう。

```
(import 'clack.builder:builder)
(clackup (builder mw3 mw2 mw1 #'app))
```

アプリケーションと各ミドルウェアは相互に独立しているため、他の場所で使い回すことができます。Clack には標準でいくつかの有用なミドルウェアが提供されています。

[*5] この例では app です。

Clack.Middleware.Session

Clack.Middleware.Session はセッションを扱うためのミドルウェアです。

```
(use-package '(:clack
               :clack.builder
               :clack.middleware.session
               :clack.session.state.cookie))
(clackup
 (builder
  (<clack-middleware-session>
   :state (make-instance '<clack-session-state-cookie>))
  (lambda (env)
    (unless (gethash :counter (getf env :clack.session))
      (setf (gethash :counter (getf env :clack.session)) 0))
    `(200
      (:content-type "text/plain")
      (,(format nil "Hello, you've been here for ~Ath times!"
                (incf (gethash :counter (getf env :clack.session))))))))
```

このサンプルコードではページにアクセスした回数が表示されます。env の中で:clack.session というキーに対応する要素がセッションを表すハッシュテーブルです。

Clack.Middleware.Static

Clack.Middleware.Static は指定したパス以下にアクセスしたとき静的なファイルをレスポンスとして返すミドルウェアです。

```
(<clack-middleware-static>
 :path "/public/"
 :root #p"/static/")
```

このように書けば、http://localhost:5000/public/foo.jpg にアクセスしたとき /static/foo.jpg が表示されます。

Caveman

以上で Clack の解説を終わります。Clack でも楽に Web アプリケーションを作ることができますが、Web フレームワークを使うことによってより簡単に開発をすることができます。Caveman[*6]は Clack を用いて作られた Common Lisp の Web フレームワークで、Sinatra や Flask、Amon2 などの影響を受けているそうです。MVC モデルを採用しており、デフォルトのテンプレートライブラリは cl-emb です。もちろん cl-who など別のライブラリを使うこともできます。また、cl-annot[*7]というアノテーションライブラリを使うことによって、直感的なルーティングを実現しています。

```
@url GET "/hello/:name"
(defun hello (params)
  (format nil "Hello, ~A" (getf params :name)))
```

このコードを controller.lisp に記述して、`http://localhost:5000/hello/kirita` にアクセスすると `Hello, kirita` が表示されます。

おわりに

いかがだったでしょうか。Lisp というとかなり歴史の古い言語であり、また括弧を多用する特徴的な文法からネタ的に扱われがちな面もありますが、この **Clack** を使うことによって、モダンでスマートな Web アプリケーションの開発が **Common Lisp で**できるかもしれないと思っていただけたら幸いです。

[*6] `http://fukamachi.github.com/caveman/`
[*7] `https://github.com/arielnetworks/cl-annot`

部員のコラム

ファジー理論 AI キホンのキ

kmc-id: fuka

はじめに

砂山のパラドックスというものをご存知でしょうか。砂の山があり、そこから砂を少しずつ取り去っていったとき、どの時点から砂山でなくなるのか？ もちろん砂を一つも取り去っていない状態では砂山は砂山のままですが、砂が一粒しか残らなくなるまで砂を取り去れば、もうそれは砂山と呼べる代物ではなくなってしまします。どの時点から砂山でなくなってしまうのか？ 我々の直感が間違っていて、実は砂一粒でも砂山であるのか？ このパラドックスに一つの答えを出せるのがファジー理論です。

ファジー理論は「あいまいさ」を数学的に扱うことができるようにするための道具で、コンピュータとの相性がいい分野です。ファジー理論を使うことで、環境変数が変化するときに閾値を超えた瞬間出力が不連続的に変化するようなものではなく、連続的で自然な出力をさせることができます。主な応用例として、テストや実績といったものの評価や、状況に対応した制御、変数を分析して分類するといったものがあります。本来は奥が深い複雑な理論ですが、感覚的な理解を優先するために厳密な表現は避け、ゲームの AI 等を例に簡単に紹介していきます[*1]。

ファジー集合

ファジー集合はファジー理論の根幹をなすものです。通常、集合というと、ある要素はある集合に含まれるか、含まれないかの 2 通りしか存在しません[*2]。しかし、ファジー集合ではある要素はこの含まれるか、含まれないかの間を連続的に変化することができます。この「ある要素がどの集合にどれだけ含まれるか」を表した関数を**メンバーシップ関数**といいます。

メンバーシップ関数

メンバーシップ関数としての関数は無数に考えられますが、単純な関数の組み合わせで十分実用的な働きをします。例えば、従来型の集合に含まれるか、含まれないかを表すメンバーシップ関数

[*1] 多分に筆者の力不足という理由もありますが。
[*2] この従来型の集合を**クリスプ集合**といいます。

は次のような階段型の関数で表わすことができます。

$$f(x) = \begin{cases} 1 & (x > a) \\ 0 & (x \leq a) \end{cases}$$

また、よく使われるメンバーシップ関数に三角型、傾斜型があります。

$$f(x) = \begin{cases} 0 & (x \leq x_0) \\ \dfrac{x}{x_1 - x_0} - \dfrac{x_0}{x_1 - x_0} & (x_0 < x \leq x_1) \\ \dfrac{-x}{x_2 - x_1} + \dfrac{x_2}{x_2 - x_1} & (x_1 < x < x_2) \\ 0 & (x \geq x_2) \end{cases}$$

$$f(x) = \begin{cases} 0 & (x \leq x_0) \\ \dfrac{x}{x_1 - x_0} - \dfrac{x_0}{x_1 - x_0} & (x_0 < x < x_1) \\ 1 & (x \geq x_1) \end{cases}$$

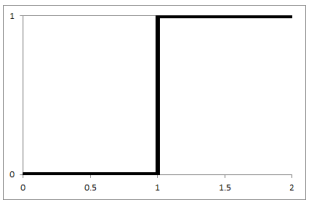


図 1　階段型関数

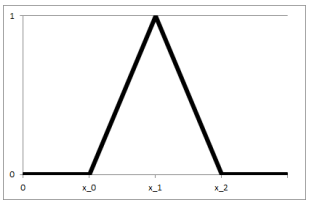


図 2　三角型関数

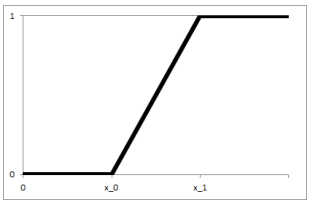


図 3　傾斜型関数

この関数にパラメータを入れたときの値が、そのパラメータのその集合への参加度合を表すことになります。例えば、階段型関数ではどのパラメータも 1（その集合に含まれている）か、0（その集合には含まれていない）の 2 通りしかありませんが、三角型関数などでは 0~1 までの実数値を取ることができることになります。

砂山の例に戻ると、図 4 を見れば 550 粒[*3]の砂粒の集まりは 0.5 の度合いだけ砂山だということが分かります。

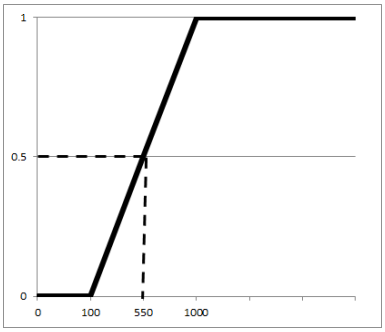


図 4　砂山のメンバーシップ関数

ファジー集合演算

普通の集合と同様に、ファジー集合にも和集合、積集合、補集合が存在します。ある集合のファジー集合を

$$F_{\text{ある集合}}(x)$$

[*3]数値はでたらめです。このような境界の決め方は一意に定まるものではなく、結局は文脈に依存することになります。

と書くことにすると、集合 A と B の和集合 A∪B は、

$$F_{A\cup B}(x) = max\{F_A(x), F_B(x)\}$$

集合 A と B の積集合 A∩B は、

$$F_{A\cap B}(x) = min\{F_A(x), F_B(x)\}$$

集合 A の補集合 $\bar{a}$ は、

$$F_{\overline{A}}(x) = 1 - F_A(x)$$

で表されます。

ファジールール

プログラムを組む時に、if 文は無くてはならない存在です。この部誌を読んでいる人には次のようなコードはかなりなじみ深い存在だと思います。

```
if 空腹度 > 80 then 松屋行く()
```

ファジー理論を用いても同じように if 文を使ったコードを書くことができます。

```
if 相手が弱い then たたかう()
if 相手がふつう or MPが残っている then まほう()
if 相手が強い and MPがない then にげる()
```

だいぶ人間の使う言葉に近いロジックが使われることになります。この時、通常の if 文とは違い、if 文の結果は実行されるか、されないかではなく、実行の度合いになります。

AI の設計

例として、RPG のキャラクターの敵に出会った場合の AI を設計します。ここでは、自分の HP と相手のレベルを見て、たたかうべきかそうでないかを決めるルールを設計することにします。

たたかうべきかの設計

とりあえず「たたかうべきか」のファジールールを設計しなくてはなりません。この AI の最終的な出力はたたかうべきかの 0~100 の値となります。この値に対して、たたかうべきでない、たたかうべき、とてもたたかうべきの 3 つのメンバーシップ関数を作ります。図にたたかうべきかのメンバーシップ関数を示しました。

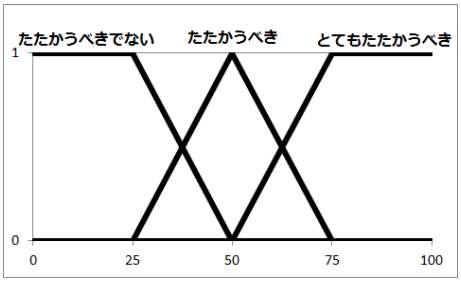


図 5　たたかうべきかの設計

相手のレベルの設計

簡単のために自分のレベルは 20 で固定で、相手のレベル 1~100 までについて考えることにしましょう。図にメンバーシップ関数を示しました。

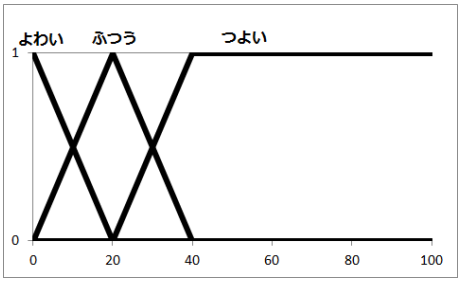


図 6　相手のレベルの設計

自分の HP の設計

図に自分の HP の設計を示しました。もうお分かりかと思いますが、これらの設計は決して一意に決まるものではありません。そのゲームのバランスやキャラクターの個性によっていくらでも変わりうるもので、開発者はうまくバランスをとる必要があります。

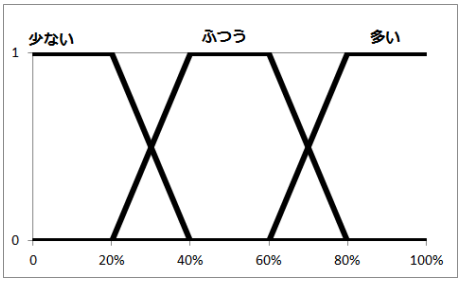


図 7　自分の HP の設計

ルールセット設計

これで下準備が完了しました。文脈を設定したので、いよいよ if 文を書くことができます。if 文は、考えるべき要素の組み合わせすべての場合について書くべきなので、ここでは 9 つの if 文を作成することになります。例として以下のような if 文のセットを考えました。

ルール 1 if 相手がよわい and HP が少ない then たたかうべきでない
ルール 2 if 相手がよわい and HP がふつう then たたかうべき
ルール 3 if 相手がよわい and HP が多い then たたかうべき
ルール 4 if 相手がふつう and HP が少ない then たたかうべきでない
ルール 5 if 相手がふつう and HP がふつう then たたかうべき
ルール 6 if 相手がふつう and HP が多い then とてもたたかうべき
ルール 7 if 相手がつよい and HP が少ない then たたかうべきでない
ルール 8 if 相手がつよい and HP がふつう then たたかうべきでない
ルール 9 if 相手がつよい and HP が多い then たたかうべきでない

ファジー推論

実際に推論の手続きを追ってみましょう。推論は以下の順で進みます。

(i) 各 if 文の条件に対して入力値を入力して値を得、結論の値を計算する。
(ii) 推論された結論を組み合わせる。
(iii) 組み合わされた結論を非ファジー化する。

では、相手のレベルが 25、自分の HP が 65% 残っているとして計算してみます。

ルール 5 if 相手がふつう and HP が少ない then たたかうべきでない

レベル 25 の「ふつう」集合に対するメンバーシップ度は 0.5、HP65% の「少ない」集合に対するメンバーシップ度は 0 なので、これらを AND でつなぐ (最小値を取る) と結論「たたかうべきでない」の値は 0 となります。

ルール 6 if 相手がふつう and HP がふつう then たたかうべき

レベル 25 の「ふつう」集合に対するメンバーシップ度は 0.5、HP65% の「ふつう」集合に対するメンバーシップ度は 0.75 なので、これらを AND でつなぐと結論「たたかうべき」の値は 0.5 となります。

ルール 7 if 相手がふつう and HP が多い then とてもたたかうべき

レベル 25 の「ふつう」集合に対するメンバーシップ度は 0.5、HP65% の「ふつう」集合に対するメンバーシップ度は 0.75 なので、これらを AND でつなぐと結論「とてもたたかうべき」の値は 0.5 となります。

以上のように計算をしてすべての結論の値を出します。表 1 に計算結果をまとめました。

さて、結果を見ると、たたかうべき、とてもたたかうべきの結論の値は決まりましたが、たたかうべきでないの結論の値は 2 通りあります。このような場合に結論の処理をする方法はいくつかの方法があります。よく使われるものでは、有界和と最大値がありますが、どちらでも最終的な結果には大きな違いは生まれません。ここでは最大値を使うことにします。これで 3 つの集合に割り当てられる値が決まりました。グラフにして表現すると、図 8 のようになります。これらを合わせたものが図 9 です。

非ファジー化

結論はでましたが、このままでは扱いにくいので数値に落とし込みます。この操作を**非ファジー化**といいます。非ファジー化にも種々の方法があります。グラフの重心を調べる方法もありますが、今回は最大値の平均を使って求めます。

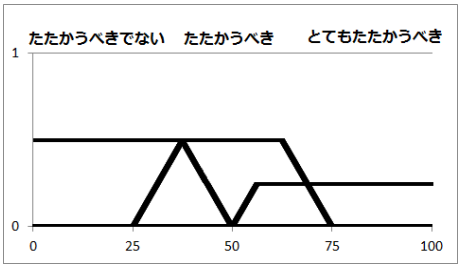


図 9　推論の結果を合わせたもの

表 1　結論の値

	HP が低い	HP がふつう	HP が高い
レベルがひくい	たたかうべきでない 0	たたかうべき 0	たたかうべき 0
レベルがふつう	たたかうべきでない 0	たたかうべき 0.5	とてもたたかうべき 0.25
レベルがたかい	たたかうべきでない 0	たたかうべきでない 0.5	たたかうべきでない 0.25

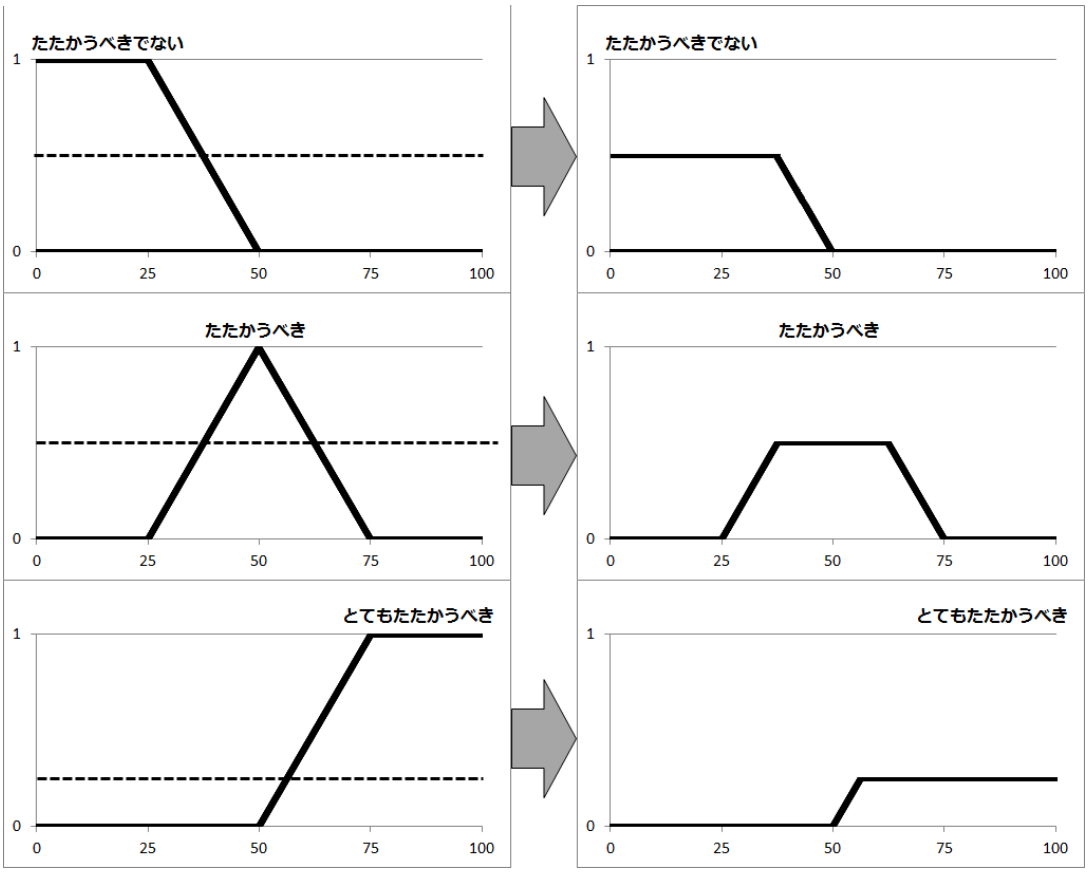


図 8　推論の結果のグラフ

最大値の平均の方法を使うために各グラフの代表値を調べる必要があります。代表値とは、グラフの最大値となる値の平均です。三角型の関数なら頂点の値ですし、傾斜型なら平坦な部分の左端と右端の和の平均となります。これらをそれぞれの信頼値、グラフの高さで重みつけをした平均が求める数値となります。それぞれの値を表にまとめました。

$$\text{求める値} = \frac{\Sigma(\text{代表値} \times \text{信頼値})}{\Sigma\text{信頼値}}$$

表 2　各集合の代表値と信頼値

集合	代表値	信頼値
たたかうべきでない	13.75	0.5
たたかうべき	50	0.5
とてもたたかうべき	78.125	0.25

計算すると、

$$(\text{たたかうべきかの値}) = \frac{13.75 \times 0.5 + 50 \times 0.5 + 78.125 \times 0.25}{0.5 + 0.5 + 0.25} = 40.965$$

となり、これで、レベル 25 の敵が出てきて、HP が残り 65% の時にたたかうべきかの値が出ました。

この他に、同様の手順でまほうだったり、にげるといったコマンドを使うべきかの値を出すことができれば、そのキャラクターがどのように動くべきかを計算することができることになります。

参考文献など

いかがでしたか。少しだけファジー理論のことを分かった気になることができたでしょうか。ところどころでかけ足の説明になってしまったので、分かりにくい説明がありましたらすいません。以下参考文献です。

- 『実例で学ぶゲーム AI プログラミング』、Mat Buckland、松田　晃一訳、オライリー・ジャパン (2007)、ISBN:978-4-87311-339-5
- 『ゲーム開発者のための AI 入門』、David M. Bourg, Glenn Seemann、株式会社クイープ訳、オライリー・ジャパン (2005)、ISBN:4-87311-216-8
- 『ファジィ理論　基礎と応用』、山下　元監、瀧澤　武信編、共立出版 (2010)、ISBN:978-4-320-01936-2

XNA で初めてのゲーム作り

lunan

はじめに

この記事は C#を少し使える[*1]が、ゲームを作ったことは無い人を対象として、XNA を使ってゲームを作る方法について紹介します。環境は Windows 上で Visual Studio Professional（Express も可能です）を想定しています[*2]。

XNA の導入

まず XNA を導入しましょう。`http://create.msdn.com/ja-JP/resources/downloads` の「XNA Game Studio 4.0 Standalone のダウンロード」からダウンロードしてください。[*3]インストールは「Yes」とか「I Agree」とかを適当に押していけばできるはずです。インストールが出来たら、Visual Studio を起動して新しいプロジェクトを作る画面を開くと、テンプレートに Windows Game が増えているはずです。さっそく適当に名前をつけて[*4]プロジェクトを作りましょう。

構造

新しいプロジェクトを作ったら何もせずに「F5」を押しましょう。青い画面が出てくるか、長いエラーが出てくると思います。長いエラーが出てきたらソリューションエクスプローラー（特に弄っていなかったら右側にあるやつです）の Properties を開いて XNA Game Studio の項の「Use HiDef to access the complete API (including features unavailable for Windows Phone).」を「Use Reach to access a limited API set supporeted by Windows Phone, Xbox 360, and Windows.」に変えてください。改めて「F5」を押すと右上に＿□×のついた青い画面が出てくるはずです。Console Application を作ると、最初に Program.cs が開かれていますが、Windows Game では Game1.cs が開かれています。Windows Game では Program.cs はゲームを実行するクラスを呼んでいるだけで、普通弄ることはありません。Game1.cs には幾つかの関数があります。

[*1] クラスを理解してると良い感じ。
[*2] Mono では Mono.XNA を利用すると XNA を使えるようですが、本稿では触れません。
[*3] 日本語 Language Pack はお好みで。
[*4] 適当に名前をつけていたら Console Application28 まで行きました。

ここに処理を書いていく事でゲームが作れます。

Initialize

ゲームを実行する前の初期化をここでします。画像以外のコンテンツ（音など）もここでロードします。

LoadContent

ここで画像をロードします。

UnloadContent

コンテンツをアンロードするようですが、余り気にしなくてもいいと思います。

Update

毎フレームの更新処理を書きます。

Draw

毎フレームの描画処理を書きます。ただし、処理が重くて描画が間に合わない場合は飛ばされることがあります。

基本的には起動した時に `Initialize` と `LoadContent` が呼ばれ、その後に `Update` と `Draw` が交互に呼ばれます。ただし `Update` や `Draw` に時間がかかってそのままではフレームレートを維持できない場合は `Draw` が飛ばされることがあります。なので、`Draw` にキャラクターの移動などの更新処理を書いてはいけません。

画像の処理

内部でどんな処理をしてもそれを表示できなかったら意味が無いので、まず画像の表示をしてみましょう。ますソリューションエクスプローラーの下の方にある（プロジェクト名）Content を右クリックし、追加→既存の項目で適当な画像[*5]を取り込みます。次に、クラス `Game1` に `Texture2D` 型[*6]の変数を作ってください。（以後 `texture` という変数を作ったことにします）そして `LoadContent` の中で

[*5] bmp, dds, dib, hdr, jpg, pfm, png, ppm, tga が使えます。透過も有効です。

[*6] このクラスは `Microsoft.Xna.Framework.Graphics` に入っているので、新しくクラスファイルを追加した時に using を忘れると出て来ません。

```
texture=Content.Load<Texture2D>("ファイル名（拡張子不要）");
```

と書いてください[*7]。これで texture の中に画像がロードされます。画像を変数にロードしたら、Draw 関数の中で、

```
spriteBatch.Begin();
spriteBatch.Draw(texture,new Vector2(),Color.White);
spriteBatch.End();
```

と書いてください。これで実行すると左上にロードした画像が表示されているはずです。spriteBatch.Begin はいまから画像を描画しますよーという関数です[*8]。spriteBatch.Draw は Begin と End の間に書いてください。Draw の引数ですが、1 つ目が Texture2D 型の画像、2 つ目が Vector2 型の位置、3 つめが Color 型の色付けです。そのまま表示したい場合は Color.White を指定します。大きさも指定したい場合は 2 つ目の引数を Rectangle 型にしてください。

文字の表示

画像だけでなく文字も表示してみましょう[*9]。画像と同じように、ソリューションエクスプローラーの下の方にある (プロジェクト名)Content を右クリックし、今回は「追加」→「新しい項目」→「Sprite Font」を選んでください。そうすると SpriteFont1.spritefont ができるのでそれを開きましょう。中にフォントやサイズなどが書かれているので好きに弄ります。次に、SpriteFont 型の変数をクラス Game1 に作ります。（以後 font という変数を作ったことにします。）そして LoadContent の中で

```
font=Content.Load<SpriteFont>("ファイル名（拡張子不要）");
```

と書いてください。これで font の中にフォントがロードされます。この後 spriteBatch.Begin と End の間に

[*7]絶対パスと相対パスどちらでも可能です。相対パスの場合.exe ファイルが.sln があるフォルダから（プロジェクト名）/（プロジェクト名）/bin/x86/Debug/以下にできることに気をつけてください。Release コンパイルの場合は Debug のではなく Release にできます。

[*8]実際には Draw を呼んでもすぐに描画されるわけではなく、End が呼ばれた時に一気に描画するようです。

[*9]この方法だと日本語が表示できません。紙面の都合上日本語を表示する方法は略します。詳しくはひにけに XNA さんの http://blogs.msdn.com/b/ito/archive/2012/02/19/wpf-font-processor.aspx を見てください。

```
spriteBatch.DrawString(font,"hogehoge",new Vector2(),Color.White);
```

と書いてください。これで実行すると左上に hogehoge と表示されているはずです。DrawString の引数ですが、1 つ目が SpriteBatch 型のフォント、2 つ目が表示する String 型の文字列、3 つ目が Vector2 型の位置、4 つめが Color 型の色です。

マウス

先ほど表示した画像をマウスカーソルに追随するようにしてみましょう。画像の位置を保存する変数としてクラス内に Vector2 型の変数を作り、Draw 関数の spriteBatch.Draw の引数の new Vector2() と置き換えてください。（以後 pos[*10]という変数を作ったことにします。）マウスの状態を取得するには、Mouse.GetState を使います。その返り値のメンバ[*11]に X 座標や Y 座標が入っているので、Update 内で

```
MouseState mouseState=Mouse.GetState();
pos=new Vector2(mouseState.X,mouseState.Y);
```

と書きましょう。これで毎フレーム pos がマウスの位置と同期され、画像がマウスを追随してくれます。

Worker クラス

今までのことを使ってゲームらしきものを作ってみましょう。そこら辺に湧いてくる敵にマウスの画像が当たると消えて得点が入るゲームを作ってみます。まず Worker というクラスを作ります。今回は自機を表す Player クラスと敵を表す Enemy クラスをまとめたものになりますが、もっと大きいゲームでは弾やボタンやエフェクトなどゲーム中に出てくる何かも含めたクラスにするといいでしょう。必要なメンバは以下のようになるでしょう。

[*10] Visual Studio の支援が受けられる言語では、変数名や関数名は略すべきでは無いでしょう。しかし、Position を pos と略すのはよく使われる気がします

[*11] ボタンの状態も取得出来ますが、押されているか押されていないかの 2 種類だけで、今押された、今離されたという状態がありません。これらの状態をとるには前のフレームの状態をとっておく必要があります。それらを含めたマウスの状態を取得する静的クラスを作っておくと楽でしょう。

```
public Vector2 Pos {get; protected set;}    //位置
public bool Dead;                           //死んだら true
protected Texture2D texture;                 //画像

public Worker(Vector2 Pos,Texture2D texture)
{
    this.Pos=Pos;
    this.texture=texture;
}
```

また、毎フレームのこの Worker の更新と描画を書くために抽象メソッドで Update と Draw を作りましょう。Worker 自体も抽象クラスとなるので abstract をつけましょう。

```
public abstract void Update();
public abstract void Draw(SpriteBatch spriteBatch);
```

Player と Enemy クラス

先ほど作った Worker クラスを継承した Player クラスを作りましょう。Player が Update でやることは自分の位置をマウスの位置に合わせることです。また、Draw では自分の位置に自分の画像を描画すればいいでしょう。

```
class Player:Worker
{
  //コンストラクタを適当に作っておく
  public Player(Vector2 Pos,Texture2D texture):base(Pos,texture){}
  public override void Update()
  {
    MouseState mouseState=Mouse.GetState();
    Pos=new Vector2(mouseState.X,mouseState.Y);
  }
  public override void Draw(SpriteBatch spriteBatch)
```

```
  {
    spriteBatch.Draw(texture,Pos,Color.White);
  }
}
```

続いて Enemy クラスを作りましょう。Update で右へずんずん進むことにしましょう。Draw は同じです。

```
class Enemy:Worker
{
  //こっちにもコンストラクタを適当に作っておく
  public Enemy(Vector2 Pos,Texture2D texture):base(Pos,texture){}
  public override void Update()
  {
    Pos=new Vector2(Pos.X+2,Pos.Y);
  }
  public override void Draw(SpriteBatch spriteBatch)
  {
    spriteBatch.Draw(texture,Pos,Color.White);
  }
}
```

Game1 クラスに戻って、メインの処理を書きましょう。やることは、

- 自機を作る（1 フレーム目のみ）
- 敵を作る
- 敵と自機を更新する
- 自機と敵が当たっていたら敵を消して点を入れる

ですね。敵はリストで管理すればいいでしょう。当たり判定は円がおすすめです[*12]

```
List<Enemy> enemies;          //敵のリスト
Player player;                //自機
```

[*12] 昔楕円の扇形の当たり判定を深夜のテンションで提案したら通ってしまいました。絶対にやめましょう。

```
bool setup;                         //自機をもう作ったかどうか
int time;                           //始まってから何フレームたったか
Random random;                      //乱数をつくるため
int point;                          //点数
protected override void Initialize()
{
  time=0;
  setup=false;
  random=new Random();
  enemies=new List<Enemy>();
  point=0;
  base.Initialize();
}

protected override void Update(GameTime gameTime)
{
  //はじめから書かれているコードは略 base.Update(gameTime) より上に書いてください
  if(!setup)player=new Player(new Vector2(),texture);//最初だけ自機を作る
  //5 フレームに 1 回敵を作る
  if(time%5==0)
    enemies.Add(new Enemy(
          new Vector2(random.Next(640),random.Next(480)),texture));
  //自機と敵を更新
  player.Update();
  foreach(Enemy enemy in enemies)enemy.Update();
  foreach(Enemy enemy in enemies)
  {
    //当たり判定 両方の距離が 60 以下ならあたって死んだことにする
    //Distance は内部で平方根を使っていて重いので普通は平方根されていないこちらで
比較する
    if(Vector2.DistanceSquared(player.Pos,enemy.Pos)<3600)
    {
      enemy.Dead=true;
      point+=10;
    }
  }
```

```
  //死んだら消す
  enemies.RemoveAll(e=>e.Dead);
  //時間を増やす
  time++;
}
```

そして Draw で Player と Enemy と点数をを描画しましょう。

```
protected override void Draw(GameTime gameTime)
{
  spriteBatch.Begin();
  enemies.ForEach(e => e.Draw(spriteBatch));
  player.Draw(spriteBatch);
  spriteBatch.DrawString(font,point.ToString(),new Vector2(),Color.White);
  spriteBatch.End();
}
```

これで実行すると、自機が敵にあたったときに敵が消えて点数が入るはずです。

まとめ

とりあえずそれっぽいものを作ってみましたが、ゲーム性はほとんどなくてつまらないです。面白くするにはシステムを追加したりしてもっと巨大にする必要があるでしょう。大規模なゲームになっても Update で更新して Draw で描画するのは変わりません。今作っている霊安京[*13]というゲームも 1.5 万行以上のコードになっていますが[*14]、ここでのサンプルのように Worker クラスを使って更新と描画をしています[*15]。大規模になっても基本は特に変わりません。読者の皆様が面白いゲームを作られることを願っています。

[*13] C82 の KMC ゲーム CD に 2 面までが収録されています（されるはず）。
[*14] 単にコードが汚いから巨大なだけとも。主に私が汚している。
[*15] 実際には Worker を管理する WorkerManager というクラスを作ったりしています。

ノンパラメトリックベイズ入門

qwerty

はじめに

機械学習の世界で近年ノンパラメトリックベイズという手法が注目されています。このコラムではノンパラメトリックベイズについて紹介し、ノンパラメトリックベイズモデリングでよく利用される中華料理店過程 (Chinese restaurant process: CRP) とインディアンビュッフェ過程 (Indian buffet process: IBP) について解説します。

潜在変数モデルとノンパラメトリックベイズ

この章では潜在変数を利用した生成モデル、モデル選択問題とノンパラメトリックベイズについて述べます。「御託はいいからさっさと説明しろ」という方はこの章を読み飛ばしていただいて構いません。

「観測されたデータを統計的に解析して何らかの知識を手に入れる」ということは様々な分野で用いられています。例えば、音声信号を解析して発話内容を出力する音声認識システムや、新聞の記事を入力してその記事のトピックを推定するシステムなどが開発されています。

観測データを統計的に解析する際、生成モデルというものを利用する事が多々あります。これは「観測されたデータの生成過程をモデル化してそのモデルを学習する」というものです。その時、実際には観測されない変数「潜在変数」を導入し、モデルの自由度を高めて推論を行います。

生成モデルに基づく推論を行う時によく用いられるのがベイズ推論です。ベイズ推論では、モデルに含まれる変数についての事前知識 (事前分布) を導入し、その事前分布と観測データのもっともらしさ（尤度）を利用して観測データを踏まえた変数の分布（事後分布）を計算します。

従来のベイズ推論の問題の一つに「モデル選択問題」があります。つまり、従来のベイズ推論では潜在変数の数をあらかじめ指定して推論を行う必要がありました。ノンパラメトリックベイズでは、その「潜在変数の数」も同時に推定することでこのモデル選択問題を回避します。

ノンパラメトリックベイズで事前分布としてよく用いられるのが CRP と IBP です。このコラムでは具体例を用いて CRP を説明し、IBP についても少し触れて、ノンパラメトリックベイズの雰囲気をお伝えしようと思います。

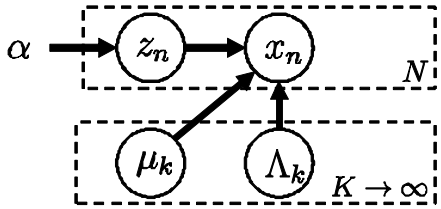


図 1　この問題のグラフィカルモデル

中華料理店過程（CRP）

問題：データのクラスタリング

まず初めに以下のような問題を考えます。

> N 個の観測データ $x_1, x_2, x_3, \cdots, x_N$ が与えられる。それぞれのサンプルをいくつかの正規分布から生成されたものと考えるとき、どのサンプルがどのような正規分布から生成されたかを推定せよ。ただし、元になった正規分布の数は未知とする。

この問題で推定するのは「各正規分布の平均・分散」と「各データ点がどの正規分布から生成されたか」となります。しかし問題文にある通り、元になった正規分布の数がわかっていないので、「モデル選択問題」について考えなければなりません。

生成モデル作成

まずはこの問題を解くための生成過程の潜在変数モデルを作ります。初めに変数を定義しておきます。$z_1, z_2, \cdots, z_N$ は各観測データが生成された正規分布のインデックスを表し、$\mu_1, \mu_2, \cdots$ と $\Lambda_1, \Lambda_2, \cdots$ はそれぞれ各正規分布の平均と精度行列を表すとします。

図 1 がこの問題のグラフィカルモデルです。また、この問題における生成モデルは以下のようになります。

$$\mu_k \sim \mathcal{N}(\mu_0, \Lambda_0), \tag{1}$$
$$\Lambda_k \sim \mathcal{W}(W_0, n_0), \tag{2}$$
$$z_i \sim \mathrm{CRP}(\alpha), \tag{3}$$
$$x_i \sim \mathcal{N}(\mu_{z_i}, \Lambda_{z_i}) \tag{4}$$

ここで、$\mathcal{N}(\mu, \Lambda)$ は平均 μ、精度行列 Λ の正規分布を、$\mathcal{W}(W_0, n_0)$ は自由度 n_0 のウィシャート分布を表します。$\mathrm{CRP}(\alpha)$ は後述する中華料理店過程を表します。また、$x \sim \mathcal{N}(\mu, \Lambda)$ は x が $\mathcal{N}(\mu, \Lambda)$ からサンプリングされていることを表します。

CRP の概要

生成モデルにおいて、潜在変数 z_i の事前分布として用いられているのが中華料理店過程 (Chinese restaurant process: CRP) です。CRP では潜在的に無限個のクラスを扱うことができ、クラス数

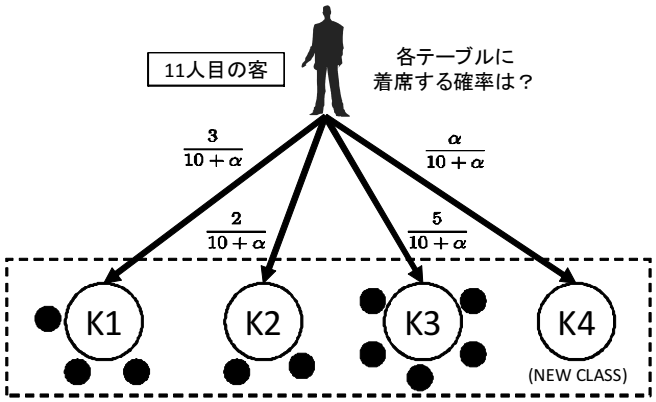


図 2　CRP のイメージ

も含めた推定が可能になります。この CRP はディリクレ過程 (Dirichlet process: DP) というもののアナロジーになっています。詳しくは参考文献をご覧ください。

中華料理店に客が入ってくる様子を思い浮かべてみましょう。はじめ、店の中には円卓がたくさんありますが、客は一人もいません。そこへ客が一人ずつ入っていくことを考えます。客はそれぞれ以下のようなルールに従って着席していきます。

1. 1 人目の客はテーブル 1 に着席する
2. i 人目の客は以下の確率にしたがって着席する。
   (a) m_k 人の客がすでに座っているテーブル k に確率 $\frac{m_k}{i-1+\alpha}$ で着席する。
   (b) 新しいテーブルに確率 $\frac{\alpha}{i-1+\alpha}$ で着席する。

CRP のイメージを図 2 に示しました。数式で表すと i 番目の客がテーブル k に着席する確率は以下のようになります。

$$P(z_i = k|z_1, \cdots, z_{i-1}) = \begin{cases} \frac{m_k}{i-1+\alpha} & (\text{if } m_k > 0) \\ \frac{\alpha}{i-1+\alpha} & (\text{otherwise}) \end{cases} \tag{5}$$

m_k はすでにテーブル k に着席している人数を表します。今回の場合、各テーブルはクラスに、客は観測データに対応しています。つまり、「i 番目の客がテーブル k に着席している」状態は「i 番目のデータ x_i は k 個目の正規分布から生成された」ということになります。この CRP を事前分布として z_i を生成することで、あらかじめ正規分布の個数が分かっていない場合にも対応できるようになります。

CRP には非常に重要な性質があります。それは「交換可能性」です。これは $z_1, \cdots, z_N$ の同時分布 $P(z_1, \cdots, z_N)$ が客が来る順番によらないという性質です。これは以下の計算結果が来客の順番とは関係なく、各クラスの合計人数のみに依存している事からも分かります。

$$\begin{aligned} P(z_1, \cdots, z_N) &= P(z_1)P(z_2|z_1)\cdots P(z_N|z_1, \cdots, z_{N-1}) \\ &= \frac{\alpha^K \sum_{k=1}^{K}(m_k - 1)!}{\alpha(\alpha+1)\cdots(\alpha+N-1)} \end{aligned} \tag{6}$$

この性質は各変数を一つ一つ更新していく Gibbs サンプリングなどと相性がよいため、CRP を用いた場合はモデルの学習には Gibbs サンプリングが用いられることがよくあります。

問題を考えてみる

CRP を組み込んだ生成モデルを使って先程の問題について考えてみましょう。ここで、$x_1, \cdots, x_N$ をまとめたものを $\mathbf{X}$、$z_1, \cdots, z_N$ をまとめたものを $\mathbf{Z}$、$\mu_1, \mu_2, \cdots$ をまとめたものを $\boldsymbol{\mu}$、$\Lambda_1, \Lambda_2, \cdots$ をまとめたものを $\boldsymbol{\Lambda}$、と書くことにします。

まずは尤度は以下のようになります。

$$P(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N} \mathcal{N}(x_n; \mu_{z_n}, \Lambda_{z_n}) \tag{7}$$

次に事前分布と尤度からベイズの法則を用いて各変数の事後分布を計算します。その事後分布からサンプリングを行うことで各変数を更新していきます。以後、$\mathbf{Z}_{-n}$ と書いたときは、$\mathbf{Z}$ の z_n 以外の要素すべてをまとめたものを表すことにします。まずはクラス割り当て z_n の事後分布について見ていきましょう。z_n の事後分布は以下のようになります。

$$P(z_n = k|\mathbf{Z}_{-n}, \mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \propto P(x_n|z_n = k, \boldsymbol{\mu}, \boldsymbol{\Lambda}) P(z_n = k|\mathbf{Z}_{-n}) \tag{8}$$

ここで、第 1 項目は尤度を表していて、

$$P(x_n|z_n = k, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \mathcal{N}(x_n; \mu_k, \Lambda_k) \tag{9}$$

となります。第 2 項目は CRP による事前分布です。この計算は交換可能性を利用して、「n 番目の客が最後に来店した」と考えて計算します。

$$P(z_n = k|\mathbf{Z}_{-n}) = \begin{cases} \frac{m_{k,-n}}{N-1+\alpha} & (k : \text{すでにあるクラス}) \\ \frac{\alpha}{N-1+\alpha} & (\text{otherwise}) \end{cases} \tag{10}$$

ここで、$m_{k,-n}$ は z_n 以外で $z_i = k$ に割り当てられているものの個数を表します。

事後分布を用いた各データ点のクラス割り当てが終わったら、平均と精度行列の更新を行います。平均と分散が更新されるとクラス割り当てで用いた尤度の値が変わるため、再び各データ点のクラス割り当てを行います。これを繰り返していくと、いずれ収束していきます。適当なところで打ち切って推定を終わらせましょう。

インディアンビュッフェ過程（IBP）

IBP の必要性

先程の問題では各データごとに一つのクラスが割り当てられるというモデルでしたが、一般的にはそうとは限りません。各データは複数のクラスに割り当てられることがあり得ます。例えば、本などのタグ付けなどを考えた時、一つの本に対して複数のタグが付けられる場合も十分に考えられます。この場合、各データを一つのクラスに割り当てるという CRP の考え方では解くことができません。そこで複数クラスへの割り当てを考慮したモデルを考える必要があります。ここで登場するのがインディアンビュッフェ過程です。

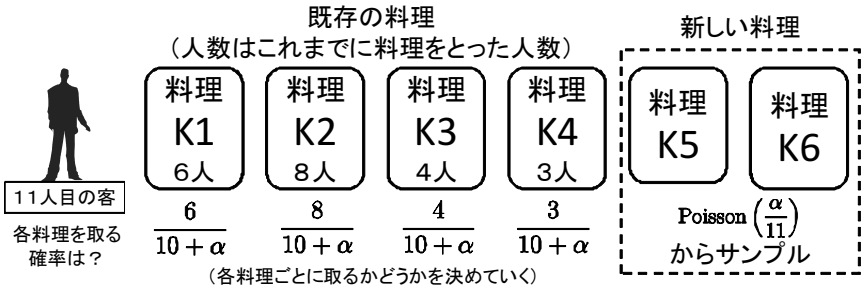


図 3　IBP のイメージ

IBP の概要

インディアンビュッフェ過程 (Indian buffet process: IBP) は CRP と同様に潜在的に無限個のクラスを扱うことが可能で、クラスの数も含めて推定します。CRP と異なる点は、各データ点の複数のクラスへの割り当てを考慮しているということです。CRP が DP のアナロジーであったのに対して、IBP はベータ過程 (Beta process: BP) のアナロジーであることが知られています。これについても詳しくは参考文献をご覧ください。

さて、IBP とはどのようなものでしょうか。先程は中華料理店を思い浮かべましたが、今度はビュッフェ形式のインド料理店を思い浮かべてください。インド料理店には料理が何種類も並んでいて、入ってくる客は以下の要領で順番に料理を取るか取らないかを決めていきます。

1. 1 人目の客は $\kappa_1 \sim \mathrm{Poisson}(\alpha)$ 個の料理を取る。
2. i 人目の客は以下にしたがって料理を取っていく。
   (a) これまでの客に一度でも取られた料理については、その料理が取られた人数を m_k 人とした時、確率 $\frac{m_k}{i-1+\alpha}$ でその料理を取る。
   (b) i 番目の客は $\kappa_i \sim \mathrm{Poisson}(\frac{\alpha}{i})$ 個の新しい料理を取る。

IBP のイメージを図 3 に示しました。客がデータ点に対応している点は CRP と同じですが、IBP の場合は料理一つ一つが各クラスに対応しています。

CRP と IBP の違いは、CRP の時は客一人に対してテーブルが一つ割り当てられたのに対して、IBP では客一人がそれまでの料理すべてについてその料理を取るか取らないかを決めるため客一人が複数の料理を取ることも考えられます。

IBP も CRP と同様に「データの交換可能性」が成り立っていて、Gibbs サンプリングとの相性が良いのでしばしば利用されます。

まとめ

このコラムでは、ノンパラメトリックベイズの紹介を行い、データのクラスタリングを題材にして CRP を説明し、IBP についても少し紹介しました。現在では CRP や IBP をベースとした様々

なモデルが考えられています。

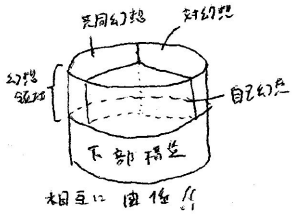
これらの手法を応用すれば、以下のようなこともできる事が知られています。

- 本のタイトルや本文を用いた自動トピック分類や自動タグ付け
- 教師データなしで行う単語分割や形態素解析
- データを適切な次元に圧縮

このコラムでは詳しいところまでは説明できておらず、また分かりにくい部分も多くあるため、これだけでは理解できない点が多々あると思います。さらに、いま挙げたもの以外にも数多くの実用例が発表されています。興味を少しでも持っていただけたなら、以下に挙げる参考文献とともにそちらの方もぜひご一読ください。

参考文献

1. 上田修功, 山田武士, ノンパラメトリックベイズモデル. 応用数理, pp.196–214, 2007.
2. 持橋大地. 最近のベイズ理論の進展と応用 (III) ノンパラメトリックベイズ. 電子情報通信学会誌, 93(1), pp.73–79, 2010.
3. T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Advances in Neural Information Processing Systems, 18:475–482. 2006.



共同幻想論

kmc-id: hidesys

マルクスの上部・下部構造論は、社会分析としては高度であるものの、人間が本質的に持っている性的要素を軽視している。一方、フロイトは人間の性衝動を鋭く分析しているが、社会領域にまでリビドー論を無前提に拡張しすぎている。『共同幻想論』の著者である吉本隆明は、両者がカテゴリー錯誤を犯さないように整理し、補完、融合させ、自分の共同幻想分析に用いようとした。マルクスの上部構造を共同幻想、フロイトのリビドーを対幻想と言い換えれば、吉本はマルクスとフロイトの補完と融合という世界思想史に残る大仕事を『共同幻想論』でやってのけたことになる。

KMC における共同幻想論

春合宿 2012 において『幻想領域と下部構造の関係、主に宗教＜共同幻想＞や家族構造＜対幻想＞が生産や消費の様式＜経済＞に及ぼす影響について』と題する 90 分講座を hidesys が行った。なお、この講座を行った翌日に著者の吉本隆明が満 87 歳で逝去された。また、逝去日は僕の誕生日でもあった。偶然とはいえ、めぐり合わせを感じずにはいられない。その後、hidesys はこの講座に触発された後輩など（kirita 他）とともに、2012 年度から共同幻想論勉強会を毎週木曜日に行う。6 月には折り返し地点ということもあり、中間までの総まとめを兼ねた他サークル合同勉強会を実施し、5 つのサークルメンバーが参加した。

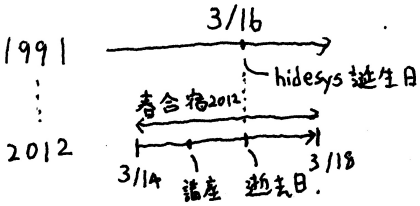


図 1　時系列

共同幻想論について

共同幻想論とは、幻想としての国家の成立を描いた国家論である。共同幻想論登場時の国家論では、国家とはルール体系であり機能性を重視したシステムだと考えられていた。集団生活を成立させる機能として国家を作ったという社会契約説や、国家とはブルジョアジーが自分の既得権益を守るために作った応力装置であるというレーニン的な国家論などがそれである。しかし吉本によると、国家は共同の幻想である。人間は社会の中に社会をつくりながら、実際の生活をしており、国家は共同の幻想としてこの社会の上にそびえている。共同幻想論では、個人の抱くタブーから村落の共同幻想、そして原始国家の共同幻想へとより高度になっていく過程が描かれた。

この寄稿では、まず共同幻想論要件を述べ、『共同幻想論』各章のさわりを説明した後に、共同

幻想論を応用した現実生活への分析例をいくつか示す。

共同幻想論要件

世界を上部構造と下部構造に分割し初めて社会分析に用いたのはマルクスであった。ここでいう下部構造とは、例えば資源の賦存状況や金銭的やり取りや科学のような物質的なものである。それに対して、上部構造とは人間が頭の中で思い描くことである。吉本は上部構造という言葉には手垢がついているとして、そのまま幻想領域と言い換えた。

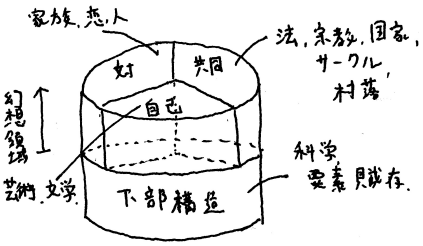


図 2　幻想領域と下部構造

下部構造――例えば科学――というものは退行することがない。問題なのは、下部構造は本当は部分的なものであるのに、それが世界の全てである誤解してしまうことである。こういった誤解は、人間の生活はより豊かになっていくしかないだろうといったような楽観論を生む。しかし、世界は下部構造だけではなく幻想領域によっても構成されている。下部構造は後退することがないものの、幻想領域は下部構造に対し、先に行き過ぎたり立ち止まってみたり逆立ちしてみたりする。こういった幻想領域を解き明かす鍵は何か。

幻想領域――人間がこの世界で取りうる態度――には 3 つの軸がある。一つは自己幻想、もう一つは対幻想、そして最後は共同幻想である。自己幻想とは、表現された表現や芸術や文学など。対幻想は、男女の性的な関係やそれを基盤にした家族の関係。共同幻想は、人間共同の仕組みやシステムをつくってそれが守られたり流布されたり慣行となったりしているところにある、国家や法やサークルなどの、観念の形態である。人間はしばしば自分の存在が圧殺されるために、また圧殺されることを知りながら様々な負担を作り出すことのできる存在である。共同幻想もこの種の負担の一つである。よって、共同幻想は自己幻想に逆立する構造を持っている。

つまり、この 3 つの軸を導入し、軸の内部構造と 3 つの軸の相互関係がどうなっているかを解明することによって、全幻想領域の問題は解くことができる。この本が提示したのは、特に人間の生み出す共同幻想の様々な態様を、総合的な視野のうちに包括するための新たな方法であった。

禁制論

フロイトの『トーテムとタブー』によると、トーテム信仰と近親相姦はエディプス・コンプレックスから導かれるらしい。というのも、父に恐れを抱いていた兄弟たちが結託して父を殺し、彼らの性的な願望の対象であった母を父から奪い取るのであるが、結局兄弟間での争いを招くことになってしまうが故に父を倒した果実であるところの母を諦め、その父を神格化しトーテムとして崇め、そして一方近親相姦をタブー[*1]（禁制）とするということである。しかし、この禁制への解釈

[*1] フロイトによると、ある事象に対して心を迂回させて触れたがらないときには、必ずそれは願望の対象でありながら恐れの対象でもあるという両価性がある。

は間違っている。なぜなら、個人の性的な幻想がいつの間にか禁制という、本来ならば自己幻想に逆立しているはずの共同幻想にそのまま無作為に広がってしまっているからである。

ここで、山人が山へ入ったら恐ろしいモノに遭ったという『遠野物語』の山人譚[*2]を題材にして、恐怖の共同性を取り出す。そして、禁制――ここでは山へ入ってはいけないということ――が生み出されるには、体験者が入眠状態に陥ることと生活圏が閉じられて弱小であることが必要であると指摘する。

憑人論

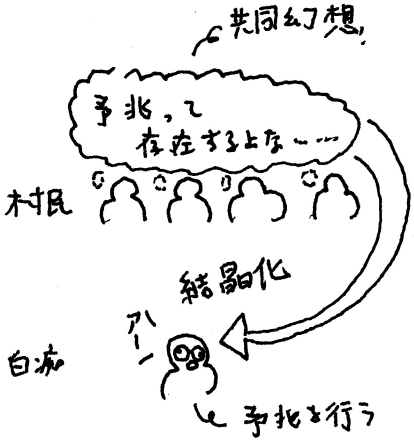


図３　予兆譚

入眠状態を、始原的なものへ向かう志向、他なるものへ向かう志向、自同的なるものへ向かう志向の３つに分類する。それぞれは、子どもの時期に神かくしに遭いそうになった話での入眠状態、エスパーで千里眼が使えるがそれによって自己の感覚を喪失している者の入眠状態、神との一体化を体験した者の宗教者としての入眠状態である。これらは、共同幻想が個体の幻想へ凝集し逆立していく過程を表している。ここで、村の白痴が村において予兆を行う予兆譚を題材に、自己幻想と共同幻想は逆立しているが故に「予兆は存在する」という共同幻想を背負わされたのは狂った白痴であったことを吉本は示した。

巫覡論

行きずりの行者からいづな（狐）を授けられた村人がそのいづなを使って術を行うと不思議なほど良く当たった。といういづな使いの民話を題材にする。術が上手く当たったのはどうしてか。それは、「狐は霊性のある動物である」という共同幻想がある共同体で、いづな使いはその村落の共同幻想をいづなに凝集させ媒介とし、自己幻想を逆立する共同幻想へと入眠状態の中で接続することで、自己幻想と共同幻想の時間性を同調させて共同

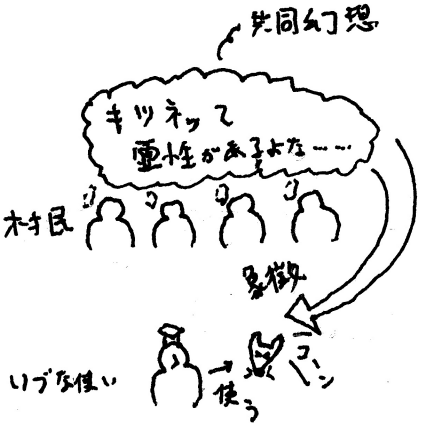


図４　いづな使い

[*2]「イノシシに突撃されて怪我をした」などではなく、「山奥に一人で行くともうろう状態になり、綺麗な女の人に遭った。とうてい女などいるところではなかったので撃ち殺し、証拠にと髪を少し切って懐に入れた。しかし、帰りがけになんともいえない眠気に襲われ、わけのわからぬまま、あらわれた背の丈の高い山人に髪を持ち去られてしまった。」のような、説話として残るような物語である。

体の下部構造を知ることができたからだと吉本は示した。この話は、共同幻想がいづなという象徴に凝縮している点で、予兆譚より高度である。

巫女論

巫女は単なる巫覡的な女性ではなく、共同幻想を自らの対幻想の相手（性的な対象）とすることが出来る人物であると述べる。巫女が儀式を行うときは、性的な光悦状態に入ることで自分自身が性になるのだ。

図 5　巫女譚

他界論

さて、死のイメージは体験した者は語り得ず自らも前もって体験できないという意味では、共同幻想（語られる死）に侵食された自己幻想である。ここで日本において、農耕民を主とする村落共同体の共同幻想では、他界の観念は時間的にと空間的に二重化されるほかはなかったことを指摘する。埋め墓は空間的な他界の表象であり、詣で墓は時間的な他界の表象である。

祭儀論

死では過程であらわれるに過ぎなかった対幻想（家族）が、生誕では本質的な意味で登場してくる。というのも、彼の自己幻想の生成に第一次的にあずかっているのは、その両親の対幻想だからである。なお、生誕において、自己幻想は徐々に周囲の共同幻想を跳ね除けることで成立する。よって、自己幻想が成立したあかつきには共同幻想から疎外されているだけではなく、逆立さえしているのだ。ここで、女の神が殺されて死体から穀物の種が生えてくる日本の神話を題材にし、共同幻想の象徴（神）である＜女性＞が＜死＞ぬことで、共同利害の表象である穀物の生育と結び付けられていることを指摘する。また、豊作祭において女性ではなく夫婦神（対幻想）の表徴を用いるケースを指摘し、これは生誕と対幻想が直で結び付けられている点でより高度だと指摘する。

母制論

母系制社会の発生には、村落内の一組の対幻想（男女の関係）が村落の共同幻想と同致することが必要であると述べる。そして、空間的に村落レベルまで押し広げられることに耐えられる対幻想は、兄弟姉妹間の対幻想しかないと結論する。彼らは自然的な性行為に基づかないからこそ緩くはあるがかえって永続する対幻想を担っているのだ。よって、妹が神権を継ぎ、兄が政治実権を執る社会が想定される。なお、母系社会の宗教的規範を現実的規範——法——として認めたとき、社会

は母権制へと転化する。

対幻想論

対幻想を共同幻想と同致出来る人物を疎外したときはじめて、家族が生まれた。男または女としての人間という範疇は、自由な個人としての人間という範疇とも共同社会の成員としての人間という範疇とも矛盾している。ここで、両者、自己幻想と共同幻想との間に対幻想という考えを導かずには居られない。なお、自然的な性関係に基づきながら決して自己還帰をしないで、一方の意識が他方の意識の内に自分を直接認める幻想関係を対幻想と呼ぶ。時間が導入された対幻想は親子関係である。穀物の栽培と収穫の時間性と女性が子どもを妊娠し夫とともに育てて成人させるという幻想性同士の時間性が違うことに気付いたとき、人間は部族の共同幻想と家族の対幻想の違いを意識し差異を獲得していった。ここで近親相姦もタブー化された。そして、穀物の栽培と収穫と男女の性行為の結びつきは農耕祭儀として疎外された。

図 6　『お兄ちゃんのことなんかぜんぜん好きじゃないんだからねっ!!』

罪責論

神話はその種族の共同幻想の構成を語る。日本では、大和朝廷勢力に支配され母権的な農耕社会を肯定したとき、人々は前農耕的段階の暮らしを否定し良心の疼きに当面した。そこで彼らは様々な農耕祭儀を生み出し、そしてこの倫理意識を補償することになった。

規範論

半ば宗教であり半ば法であるような中間状態を今「規範」と呼ぶことにすれば、どのような規範が宗教に分裂するのか、そして、どのような規範が法に転化して国家と法に分裂するのか。法的な共同規範が取り出されるのは、共同幻想が血縁的な社会集団の水準を少しでも離脱した時である。日本では、農耕法的な共同規範を法として取り出し、前段階にある共同幻想を宗教に蹴落とした。この転移は共同幻想それ自体の疎外である。

起源論

家族における兄弟姉妹婚の禁止→農耕権の専有と土地の私有の発生→村落における血縁共同性の崩壊→部族的な共同体の成立　というクニの発生までの道筋を描く。そして最後に、『魏志倭人伝』などの資料を頼りに、邪馬台国が高度な国家の段階にあったことを示す。

現実生活へ応用

サークルもまた共同幻想の一つである。むろん、カルト的集団婚サークルでない限り対幻想と共同幻想が完全に同致することはありえず、よって共同幻想の禁制から国家の成立までの段階の中では非常にプリミティブな立ち位置になる。サークル運営における様々な諸問題を幻想的問題として取り出すことによって、問題の起源や構造そして本質は、自己幻想・対幻想・共同幻想の内部構造と相互関係として考察をすることができる。例えば、サークルという共同幻想において対幻想に対する接し方はどのようなものか、所謂肉食系テニサーのように対幻想が共同幻想を覆うような状態を示しているとか、または共同幻想としては対幻想の要素を無視して各個人にしか構成要素として見えていないとか。例えば、結成願いといった外的要求に応じて規範から法を取り出す際に気を付けなければいけないことは何かとか。

では、どのように問題を分析すればよいだろうか。手順は次の通りである。 1.) 世界を下部構造と幻想領域に分ける。 2.) 下部構造と幻想領域のズレに目を向ける。 3.) 幻想領域を、自己幻想・対幻想・共同幻想として取り出す。 4.) 各幻想領域や下部構造の内部を把握する。 5.) 各幻想領域や下部構造同士の接着域を分析する。 物事を下部構造と幻想領域に分ける際には、なにが物質的事象でなにが人間の頭の中で起こっていることかということを考えれば良い。そして、彼らの想いの実行を妨げている物質的事柄を見つける。また、ある特定の個人に着目して、彼が個人的に想うことと実際に共同体が持っている思想の逆立関係も細部に書き出す。こうして現実の分析ができたのなら、最後に理想とする状態を明確にそこまでたどり着くにはどうしたらよいのかを考える。

結語

下部構造、科学を世界の全体性だとするような誤解を捨て、幻想領域自体と幻想領域とのズレに対して目を向けるべきだ。人間は経済原理だけに従って動くのではない。一時の感情や慣習や宗教によっても動くのである。共同幻想論分析が世界を見る時の一つの視点になれば幸いである。

参考文献

- 吉本隆明 (1982)『共同幻想論』角川書店（角川ソフィア文庫）
- ジークムント・フロイト (1977)『精神分析入門』新潮社（新潮文庫）
- ジークムント・フロイト (2009)『トーテムとタブー』（フロイト全集 {12}）岩波書店
- カール・マルクス (1981)『賃労働と資本』岩波書店（岩波文庫）
- カール・マルクス、エンゲルス (1971)『共産党宣言』岩波書店（岩波文庫）
- 柳田国男 (1976)『遠野物語』岩波書店（岩波文庫）
- 舎人親王 (養老 42 年 [720])『日本書紀』
- 草野紅壱 (2009)『お兄ちゃんのことなんかぜんぜん好きじゃないんだからねっ!!』双葉社（アクションコミックス）

きょう “も” アスミッション

Moko

この記事は、2011 年暮れ〜2012 年早春にかけて実行された、芸大生と KMC 部員によるコラボ・プロジェクト「きょうのアスミッション」の記録として書かれたものです。

出会い

「はじめましてこんにちは。京都造形芸術大学 4 回生の西本 明日美と申します」

ある冬の日、KMC に 1 通のメールが届きました。日々様々なメールを頂く KMC ですが、あまり目にしない類の差出人のせいか、すぐにわたしたちの話題にのぼりました。差出人は京大の近所の京都造形芸術大学の 4 回生、西本明日美さん（以下、明日美さん）。卒業制作をするにあたり、スマートフォンで動作するアプリケーションを作りたいが、ご自分ではプログラミングができない。そこで部員の力を借りたい、ということが少し切羽詰まった文章で書かれていました。

なんとなく作業時間が短そうということが気になったのと、アプリ制作に明るい部員が少ないこともあり、「これはノーチャン[*1]ではないか」という意見が多数挙がりました。しかし、せっかくの芸大生との交流機会です。その場にいなかった tobiuo 君に、私はすぐに電話をかけました。

「tobiuo、芸大の困っている女の子を助けてあげようよ」
「やりましょう！」

まさに二つ返事でした。tobiuo 君は Web プログラミングができる部員のうちの一人で、個性派揃いの KMC 部員の中でも、群を抜いてユニークなキャラクターのため部内でたいへん愛されていました。部員たちのなま温かい支援のもとに tobiuo 君はメールを書きました。そして大学近くのカフェ “Cafe collection” で明日美さんと打ち合わせすることになったのです。

補佐役の抜擢

プログラミングについての打ち合わせをする時には、自分の力量と、その実行にかかる時間を把握していることが必要です。そうでなければ割に合わない仕事を引き受けてしまうからです。心優しい tobiuo 君のこと、そんな状況に陥るかもしれない。補佐役が必要ですが、相手は女性です。

[*1] やめたほうがええんとちゃうか、の意。反対に、ワンチャンはいけるいける！ を意味する。当時の部内での流行スラングであり、執筆時現在でも部内では普通に使用されている。

男性部員 2 人で打ち合わせに向かえば、圧迫感があるかもしれません。こんなサークルですから、口下手な部員は少なくありません。またメールから推察される限り、作業自体はそれほど専門性が必要ではなさそうでした。

そんなわけでプログラミングはほとんどできないものの、当時唯一の現役女性部員[*2]であった私が tobiuo 君の補佐役となりました。

初めての打ち合わせ

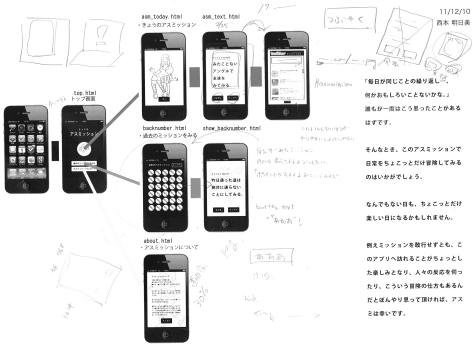


図 1　企画書

Cafe collection は、味にうるさい部員も舌鼓を打つおいしいコーヒー、そして素朴なケーキ、スープたっぷりのパスタが魅力の京大近所の喫茶店です。雰囲気が落ち着いていることもあって、部員も当時よく利用していました。店で tobiuo 君と 2 人で待っていると、約束の時間の少し前に明日美さんが入ってきました。彼女の姿をみて、もともと緊張気味だった tobiuo 君が、コチコチになっています。大丈夫か。明日美さんが持ってきてくださった企画書 (図 1) をもとに詳しい企画内容を聞いたところ、そのアプリケーションは以下のような内容でした。

アプリケーションを起動すると、最初に「ゴー」ボタンが表示され、それを押すと、明日美さんの描いた日替わりのイラストが現れる。次に進むと、絵の内容を表した文章が表示される。たとえば「みたことないアングルで友達を見てみる」のような「日常をちょこっと冒険するミッション」である。最後に、ミッションについて思ったことや実行した感想などを `Twitter` につぶやける。また、過去につぶやかれたツイートの一覧を見ることができる。最初の画面からは過去のミッションが一覧できる。

マルチプラットフォームで動作して欲しいとのことでしたが、JavaScript を仕込んだ Web ペー

[*2] 今年度ついに女性部員がひとり入部しました。実に 2008 年度以来の出来事です。

ジで作ればいろんなスマートフォンで利用できること、思ったよりも簡単なものなのですぐに作業に取り掛かれることを伝えました。プロジェクトの最終目標は 2 月末から 3 月初頭にかけて開催される卒業制作展に出展することでしたが、途中何度か審査があるため、意外と時間が無いのだということでした。なんにせよ、あまりゆっくりはしていられません。

言語の壁・あるある

Web ページを作成するにあたり、まずは明日美さんにそのための画像を作成してもらう必要があります。しかし、普段 KMC で話すのとはまったく違う感覚がそこにありました。

tobiuo 「Web ページを作るために、画像素材を切り出してもらえますか」
明日美さん 「切り出す？ っていうのは何のためですか？」
tobiuo 「ボタン要素にその画像を背景として表示させて、それに『次のページにいく』『送信』とかの挙動を仕込むんです」
明日美さん 「……？？？」
Moko 「……？？？（言いたいことはわかるが、お前は何を言っているんだ）」

tobiuo 君は完全にプログラマ視点で画像の切り出しが必要な理由を説明しており、明日美さんにそれが通じなかったのでした。やむなく私は彼の言葉を通訳しました。

Moko 「Web ページはいくつかの画像の部品を組み合わせてできていて、1 枚の全体の画像があるだけじゃできないんです。例えばこの画像は真ん中に置いて、ボタンはその下に置く、という風に組み立てるんです。明日美さんには、その部品を作っていただきたいんです。具体的に言えば、この『ゴー』ボタン画像や、『トップへ』ボタン画像です[*3]」
明日美さん 「なるほど！」

私の役割は「補佐役」から「日日通訳担当」になったようでした。

できる限りはやく画像素材を作っていただき、それを使って次回ミーティングまでにアプリケーションを形にすること、来週同じ場所・時間にもう一度打ち合わせをすることを約束してこの日のミーティングは終了しました。

何故かギリギリ

「Moko が CSS を組んでから tobiuo が JavaScript ＋ HTML を作る」という本来とは逆の作業手順が正しいと tobiuo 君が思い込んでいることが発覚したのは、第 1 回ミーティングから実に 5 日後でした。彼の作業が終わるのを待っていた私は愕然としました。発覚の夜はちょっとした修羅場を迎え、私は tobiuo 君の尻を叩きながら必死になって CSS を書き、なんとか見た目だけは取り繕ったデモ版ができあがりました。Twitter への投稿機能はまだ実装されていませんでし

たが……。

tobiuo 君がディレクトリに “index.html” ではなく “top.html” を置き、あまつさえ

> **「いやぁ、最近 `index.html` って見ないなぁ、と思いまして」**

と発言したのは、今となっては笑い話です。まったく。

第 2 回ミーティングではなんとかデモ版を明日美さんにお見せできました。「カタチになったものが見え非常に嬉しかった」という感想をいただきました。

王子様になれるチャンスとは

第 2 回ミーティングの 4 日後、明日美さんは大学のゼミで「アスミッション」のデモを見せることになっていました。ミーティング当時のデモ版「アスミッション」は一応動作していたものの、スマートフォンやタブレットで見た時にフルスクリーンでページの全体表示ができていなかったので、そうなるように修正しておきました。

ところがゼミ当日、発表前に明日美さんからメールがありました。「iPad で見ると全体表示されない」というのです。「アスミッション」は git でバージョン管理しており、昨夜最新版を pull したのですからそんなはずはありません。しかし pull をした先は tobiuo 君が便宜的に貸している Web スペースで、私が管理しているわけでもありません。何か手違いがあったのかもしれません。私はちょうど実験中でろくに動けなかったので tobiuo 君にメールを送りまくりましたが、残念ながら連絡はつかずじまいでした。

原因は、以前のバージョンを表示した時のキャッシュが Mobile Safari に残っていて、最新版にアクセスした時にそれが表示されてしまったことでした。あとでお話を聞いたところ、なんとかゼミではうまくプレゼンテーションできたとのことで、ホッとしました。

それにしてももしあの時、tobiuo 君が颯爽と明日美さんに電話をかけ「更新ボタンを連打してください (キリッ) 」と伝えていれば、彼は一躍王子様になれたでしょうに、なんとも惜しいことをしたものです。

ゼミ後、明日美さんは帰省のためしばらく動けないということで、一旦アスミッションプロジェクトはお休みになりました。

雲行きが怪しくなってきた

1 月半ばからプロジェクトを再開しました。「アスミッション」は最終的に 2 月末からの卒業制作展に出展される予定ですが、tobiuo 君は定期試験が 2 月頭まで、私は卒論 2 月の半ばまであり、大きな変更をするならはやめに動いておかないと余裕が無くなってしまいます。しかし明日美さんは各箇所のデザインに悩み、1 月末まで試行錯誤を重ねました。変更されたデザインを元に、CSS

を書き換えるなどの作業がしばらく続きました。

2 月の頭に次のミーティングを開きました。

「トルコに行きます」

「あと、定期試験寝倒しちゃって、**トルコ出発直前までに専門の教科書の和訳 50 ページ**やらないと…。」

ミーティングでこんな言葉が tobiuo 君の口から飛び出して、目玉が飛び出る思いでした。まったく何を考えているのでしょうか。トルコ出発までには日数があるからよいものの、それを潰すように圧倒的に時間を食う和訳 50 ページを課されるとは！ 当時の状況は 図 2 のようになっていました。頭痛がするのを抑えつつ、tobiuo 君の出発前日までには Twitter への投稿周りの実装まですべて済ませることを決定し、この日のミーティングは終了しました。

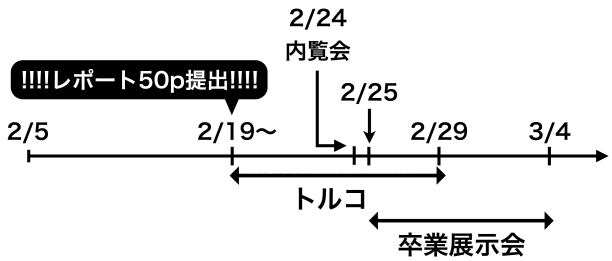


図 2　逼迫するスケジュール

2 月中旬に全体のデザインが決定しました。あとは Twitter の投稿周りを実装すればプロジェクトが終わります。

当時の Twitter への投稿仕様は、id:asumission としてツイートするとツイートにハッシュタグ #asumissonYYMMDD（年月日）が付加されて投稿され、その後、id:asumission のツイートをユーザ検索結果一覧から引いてきて表示するというものでした。トルコ出発前夜まで私達 3 人は Skype で進捗を報告し合いながら開発を進めました。

tobiuo 「実装が終わりました」
明日美さん 「さっき試しにつぶやいてみたのですが、まだ一覧に私のツイートが出ません」
tobiuo 「それも謎なんですよねえ。Twitter 側の仕様なんでしょう」

どう考えてもおかしい会話なのですが、私は追加で降ってきた CSS の調整に四苦八苦しており、そのおかしさに気づけなかったのです。

炎上

「あの、やっぱりつぶやけてないみたいなんですけど…」

明日美さんからそんなメールが届いたのは、内覧会前日の 2/23 でした。なんということでしょう……、 しかし tobiuo 君はすでにトルコ、私はプログラミングはさっぱりです。意を決して部内 IRC で状況を説明しました。するとわかったのです、全てはプログラムのせいであったことが。水が出ると思っていた蛇口はハリボテで、うしろには水道管すら通っていなかったような気分でした。Twitter に投稿できなかったのは Twitter の仕様のせいでもなんでも無かったのです。

悲しみに暮れながら、私は明日美さんにメールを書きました。

> Moko 「スクリプトに根本的な問題があることがわかったので、部員一同で解決にあたっています。もう少しお待ちください」
> 明日美さん 「なんて心強いのでしょう！」

困っていたところに、人呼んでファイア・ファイター nojima さんが反応しました。nojima さんは私達のリポジトリを pull して状況を把握し、あっという間に機能を直し、さらに新しい機能を 3 つ付けてしまいました。なんて心強いのでしょう……。しかし事態はまだ収束していなかったのです。

芸大突撃

内覧会当日になり、様子はどうかなと思っていたら、明日美さんから悲鳴のような電話がかかってきました。

> 「投稿文がダブります。ボタンの位置がずれてます。つぶやき一覧をスクロールして下にずらせません！」

卒論を出し終えたばかりの私は大学をサボタージュし、芸大に突撃しました。そこで「アスミッション」を触ってみて原因がわかりました。芸大の iPad は、私の手元の iOS シミュレータのバージョンより、また部員所有の iPad より、古い iPad だったのです。iOS4.0 以前では 1 本指スクロールはサポートされていませんでしたし、CSS のレンダリング挙動も微妙に異なるようでした。

調べたところ投稿文がダブる問題は結構有名で、日本語入力で「確定」ボタンを押さずに投稿すると書き込んだ文章が二重になってしまうというのです。部員の iPad でも何度も試してもらっていたのですが、どうやら必ず「確定」を押していたらしくこれには気付けませんでした。部員律儀すぎでは。

状況は現物を見なければわからないということで、ついにファイア・ファイター nojima さんも芸大に現れました。投稿文のダブり対策には「0.2 秒ごとに入力内容を保存しておき、送信ボタンが押された時に入力内容ではなく最新の保存内容を送信する」という力技が取られました。1 本指スクロールをいくつか実現する方法はありましたが、どんな環境でもうまくいく保証が得られなかったので「2 本指でスクロールしてください」と注意書きを入れることで対処しました。

どうやら、これで実装周りはおおかた片付いた模様です。あとは見た目の細かい整形をすればいいでしょう。そこで 3 人で部室に向かい、最後の調整をすることにしました。冬の陽は短く、大学

を出るとすでに暗くなっていました。

部室でのエピソード

私達の部室は完全に「オタクの部屋」です。得体のしれないコンピュータが林立し、コタツが並び、本棚には怪しげな技術書が乱雑につめ込まれ、押入れには謎のケーブルやアダプタがひしめき、壁には勉強会用兼国民的アニメ観賞用のスクリーンが掛けられています。でも、明日美さんは楽しそうでした。私もこんな部屋は好きですが、これを楽しみそうな女性は他にあまり見たことがなく、「さすが芸大や……」と謎の感慨を覚えました。

私がちまちま見た目を整形をしている間、コタツの向こうでは心あたたまる会話が聞こえてきます。

nojima 「Mac はフォントレンダリングがこういう仕組みでこうこうなので見た目が云々かんぬん」
明日美さん 「えっ」
Moko 「明日美さん、Mac だと字が綺麗ですよね」
明日美さん 「はい！」

nojima 「うちの部員のアカウントがみんな小文字なのは、メールアドレスに大文字小文字が混ざってるとメール送信した時にこういうことが起きてどうのこうの」
明日美さん 「えっ」
Moko 「混ざっていると都合が悪いんですよ」
明日美さん 「なるほど！」

明日美さん 「nojima さん、休日は何をしてらっしゃるんですか」
nojima 「インターネットです」
明日美さん 「も、もっとくわしく！」
nojima 「Tumblr したり ssh したりしています」
Moko 「(´ω`)……」

nojima 「プー野球っていうゲームがあるんですけど、難しくて（画面を見せる）」
明日美さん 「かわいいじゃないですか！ こんなのやってるなんて意外ですね。nojima さん、イーヨーに似てますね！」
Moko 「(´ω`)…… そういや確かに似てるな、イーヨー」

明日美さん 「通訳してもらえれば、部員の人ともしゃべれますね！」
Moko 「(´ω`) そっかぁ……」

ほっこりしているうちに「アスミッション」は完成を迎えました。作業自体は 1、2 時間で済んだのですが、こんな感じでずっと他愛もない話をしていました。

その後

卒業制作展はトラブルもなく無事に終了したそうです。 時々@asumisson のツイートを見ていたら、お客さんが結構色々なことをつぶやいていて、なかなか興味深いものでした。

その後打ち上げをしたのですが、非常に残念なことに nojima さんは来れず、明日美さん・tobiuo 君・私の 3 人で丸太町の「エル・ラティーノ」でお酒を飲みました。明日美さんの地元福井では、小学校にイノシシが出るそうです。恐ろしいですね（なんのこっちゃ）。

反省

編集長の kirita 君からは「4 ページに収めてください。無理なら最大 10 ページです」との指令を頂いていますので、なんとかその範囲で収まるよう頑張りました。

今回の件でもっとも反省すべきなのは、iOS のバージョンを最初に確認して現物を横に並べて開発をしなかったこと、tobiuo 君が変なことを言ってる時点で気付けなかったことでしょうか。しかし芸大の備品の iPad を持ちだして開発というのも難しく、ギリギリでないと結局できなかったのかもしれないとも思います。tobiuo 君・私とも、試験や卒論で忙しく、そういった細かい点を意識しながら継続して取り組めなかった点も反省すべきと考えています。

tobiuo 君と明日美さん、nojima さんと明日美さんの会話から「畑違いの人間どうしの会話というのは、本当に噛み合わない」ということがよくわかりました。今回は一貫して「日日通訳」をやっていたような気がします。

文中でも散々書きましたように、tobiuo 君は色々ミスをしたのですが、彼の最大の功績は「やりましょう！」と言ったことです。外部との連携は KMC があまり得意としないところです。KMC 独特のノリや会話が通じないから面倒なのかもしれません。ですから tobiuo 君がああ言わなければ、こんなプロジェクトはそもそも始まらなかったでしょう。

今回のプロジェクトは、先程も述べたように名乗りを挙げた tobiuo 君、開発のために iPad を貸してくれた possum 君、ファイア・ファイター nojima さん、そして数多くの部員の心強い協力によって無事終了しました。そして何より企画担当の明日美さん、お疲れ様でした。独特なイラストを見るのが楽しかったです。

なお記事のタイトルは、プロジェクト中ハプニング続きだったことから、毎日がアスミッションだったという意味でつけました。

あとがき

編集後記

kirita

編集長の kirita です。この度は独習 KMC vol.3 を読んでいただきありがとうございました。今回の部誌では、技術的な内容にしよう、ということを 3 月の春合宿でなんとなく決めたのですが、蓋を開けてみたらこのような部誌が出来上がっていました。個人的には良いものができたと思っているのですが、いかがでしたでしょうか。ご意見、ご感想などアンケートで募集しているので、よろしければご協力ください。下記の短縮 URL からアンケートページにアクセスできます。

独習 KMC vol.3 読者アンケート —— `http://goo.gl/cmxvi`

ところで、本というものは、その性質上ページ数が 4 の倍数でなければなりません。記事が集まり校正が終わった時点でページ数を数えてみると 74 ページでした。74 は 4 の倍数ではない、ということで表紙の話をします。

過去 2 回の部誌をご存知の方ならば、今回の部誌の表紙が少し違った雰囲気であることに気づかれるかと思います。KMC にはひたすら絵を描きまくっているイラスト勢という人たちがいて、当然のことながら今まではその人たちが表紙を担当していました。しかし今回はイラスト勢ではない、しかもあろうことかあるまいことか編集長の私が表紙を担当しました。表向きの理由としては、部誌に新しい風を吹き込む、というようなことを考えていますが、6 月末になっても表紙の仕事を振っていなくてやばいと思った編集長が、半ば冗談のつもりで自分でつくるわ〜と言ったところ、数名の賛同を得てしまい作り始めてしまった、というのが本当のところだったりするかもしれません。半ば冗談ということは、もう半分は本気だったのでしょうか。

ぼんやりとしたアイデアはあったので、それに従っていくつか試してみました。最終的に、自分のリュックと靴を写真で撮って 2 値化してみるといい感じだったので、とりあえずこれでどうかと部内で意見を募集しました。

すると Moko さんから「トーンで影をつけてはどうか」というコメントがありました。ですが、トーンで影などと言われても非イラスト勢の自分にはどうすればいいのかさっぱり分かりません。「では Moko さん影つけてもらえますか」と頼んだところ、10 分くらいで今の表紙が返ってきました。

速いな、と思いつつ 2 つの絵を見比べたのですが、どちらを採用すべきかでかなり迷ってしまいました。贔屓目ということも多分

にあるでしょうが、自分が全部作ったオリジナルの方が雰囲気が出てて良い、などと思ったり、そうはいっても影が付いているほうがちゃんと表紙絵としてまとまっている、と考えたり。

結局影付きを選んだのは、そっちのほうが絵として綺麗かな、と思ったからです。今だったら迷わずに影付きの方を選ぶでしょう。影の付いてないオリジナルの方も、ここで読者の皆さんにお見せすることができてよかったです。

これで表紙の話は終わりです。最後に、本誌の制作にあたり、部員のみなさんには記事の執筆を始めとして文章の校正など、その他もろもろ様々な協力をしていただきました。この場でお礼を申しあげます。

独習 KMC vol.3

2012 年 8 月 11 日　初版発行

著作・発行　京大マイコンクラブ
表紙デザイン　kirita, Moko
裏表紙デザイン　crys

メールアドレス　info@kmc.gr.jp
Web　http://www.kmc.gr.jp/

独習KMC vol.3

■利用分類

部誌

■製品分類

KMC

■レベル

初級　中級　上級

Kyoto University Micro Computer Club