Programming Contest in OCaml

Shinya Kawanaka a.k.a. mayah (@mayahjp)

自己紹介

- 名前: 川中 真耶 (Kawanaka, Shinya)
- id: mayah
- twitter id: @mayahjp
- OCaml との出会い
  - 大学の ML 演習で (2002 年)
  - 使っていくうちに「コンパイルが通ったらだいたい動く」のに惚れ込んでいく

今明かされるOCaml.JPの真実

- 実はまずネタでドメインを取った
  - 東京大学理学部７号館のごく一部で、言語に関するドメインを取るのが流行った
  - ocaml.jp, csharp.jp, ...
- 取ったからにはなんかやらなきゃ...

OCaml で Programming Contest?

OCaml が如何に優れているか

世界中に知らしめてやろう

ふんかポケモン バクーダ

OCaml で Programming Contest?

- しかし、C++ や Java しか使えないコンテストも多数...

OCaml で戦えるコンテスト

- 手元で実行して答えを提出するのであれば、OCaml でも戦える！
- たとえば
  - ICFP Programming Contest
  - Google Code Jam
  - コンテストではないが Project Euler など

普段のプログラミングとコンテストでは何が違う？

- 時間が短い
- 複雑なアルゴリズムが要求される
- 実行速度も重要

時間が短い

- １時間から、長くても数日間
  - バグを入れてハマると貴重な時間が浪費されて非常に不利
  - → バグを入れにくい言語が有利
  - → 書くのが楽な言語が有利
- やや長期的な戦略も重要になってくる
  - 初速だけだったら LL でも OK だが...
  - → ある程度の仕様変更に強いことが必須

複雑なアルゴリズムが要求される

- 普段の開発にはあまり出てこないようなアルゴリズムを実装する必要がある場合が多い
- データ構造もしばしば複雑になりがち
- → 複雑な問題をミスなく書かせてくれる機構が必要

実行速度も要求される

- ２秒以内に答えを出せ、など

- ある程度言語としての速度が速ければ、ちょっとぐらい計算量のオーダーが悪くても力技でなんとかなったりする場合がある

- → 速い言語の方が有利

つまり、プログラミングコンテストに重要なのは

- 書くのが楽
- 高速
- 仕様変更に強い

どの言語が向いているか？

	OCaml	C++	Java	Haskell	Ruby
書くのが楽	○	×	×	○	○
高速	○	○	○	×	×
仕様変更に強い	○	△	△	○	×

つまり、OCaml で書くのは

有利

すいません、言い過ぎました

- 素の OCaml ではライブラリが足らない
  - 標準ライブラリが貧弱すぎる、とにかく関数が足りない
    - というかほとんど OCaml を作るために必要な関数しかない
  - ruby gems ぐらい揃ってくれれば...
- ライブラリそろってしまえば、バグの心配もあまりせずに非常にサクサク書いていける

OCaml で実際にコンテスト

をどのように戦うか？

ICFP Programming Contest (ICFP-PC) を例に考える

- ICFP (International Conference on Functional Programming)の数ヶ月前に行われる 72 時間耐久プログラミングコンテスト
- もうおなじみの最も優れた言語を決める【違】コンテスト
- 優勝した言語は、the programming tool of choice for discriminating hackers を名乗ることができる
  - ここに OCaml の名前を刻むのが OCaml プログラマの宿命
- 最近日本人も結構参加している

ICFP-PC の出題内容は？

- 非常に多岐にわたる
  - 高速な実装を作るもの
  - ゲームの AI を作るもの
  - 多数のパズルをひたすら解くもの
- 関数型の分野と近い問題が出る場合もあれば、関係が非常に見いだしにくい場合もある

OCaml ならどのように戦えるのだろう？

- 過去の問題を OCaml でどのように戦えるかを考える

2010 年

廃車の山を作るゲーム

2004 年

アリさんを操って

エサを集めるゲーム

ICFP-PC 2010

- 問題
  - ~~多数の車を廃車にしろ！~~
  - 車に燃料を供給して利益を稼げ！

- 今年の問題なのでちょっと詳しめに。
- チーム Oh! tomaton で参加

車 = 0 以上の整数からなる正方行列の不等式たち

- $B^2 < C$
- $ABC \leq ACB$

} これで 1 つの車

- A ≤ B の意味は、任意のベクトル u をかけて Au, Bu を考えたとき、その全ての成分が Au[i] ≤ Bu[i] となる
- A < B の意味は、A ≤ B かつ、任意のベクトル u をかけて Au, Bu を考えたとき、第 1 成分は Au の方が真に小さい

燃料 = その不等式を満たすような実際の行列

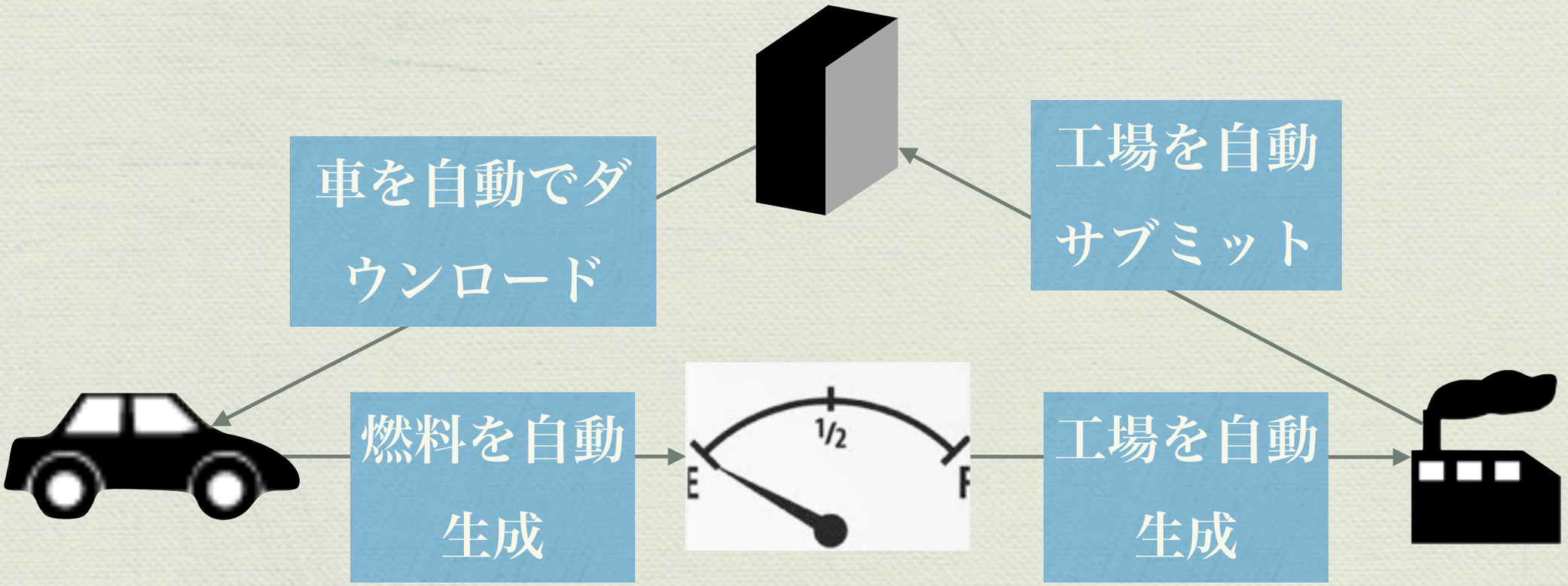
- $B^2 < C$
- $ABC \leq ACB$
- 具体的には、
- $$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

利益 = スコア

- 車に燃料を供給すると利益(=スコア)が得られる
  - N チームによって燃料が供給される場合、良い燃料順に 1/N, ..., 1/(2N-1) 点が定期的に入る
    - 良い燃料 = 燃料を作り出す「工場」が小さいほど良い
    - 工場 = ２入力２出力のゲートを組み合わせてできる回路
- 実際には燃料ではなく、工場を提出する。
- 誰でも燃料を供給できる大衆車に燃料を供給しても儲からない
- 燃料供給の難しいエリート車に燃料を供給してこそ儲かる

解答は
ウェブインタフェースで提出

- 出題者のサーバーに DOS 攻撃して提出
- 最終的には、次の４つを自動的にぶん回せるようにしていく。



最近の ICFP-PC には Qualification Round がある

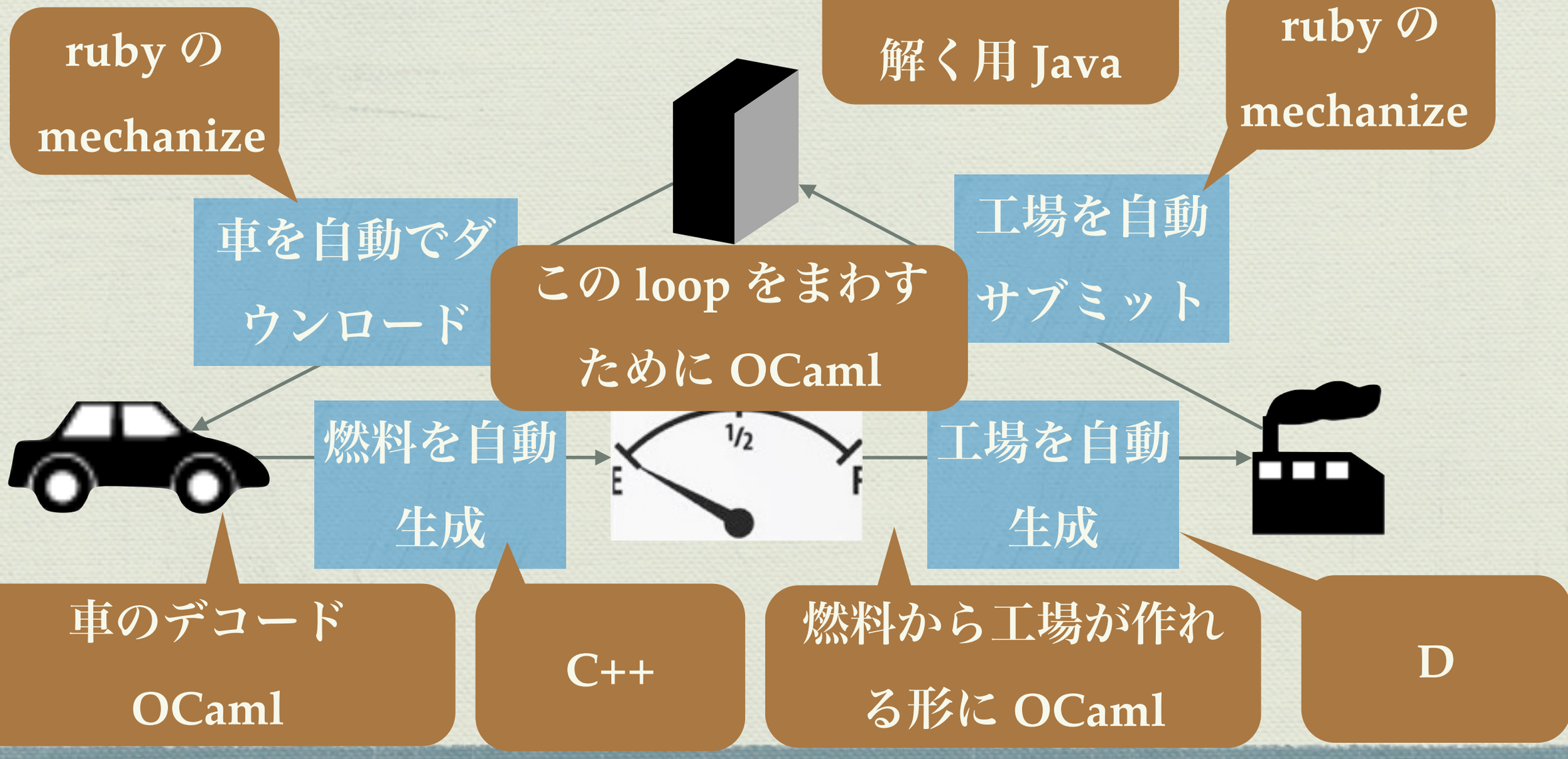
- 本質的な問題を適当に難読化して、スキルの低いひとを遠ざけてしまう措置
  - (ほんとうにやめてほしい)
- 車も燃料も実は難読化されている
  - 車は３進数の列で書かれており、仕様は推測する必要がある
  - 燃料は得点を稼ぐには工場をゴルフする必要がある

燃料の作成

- そんな簡単に車に適合する燃料（行列）が求まるものではない
  - 問題のクラス的には NP-complete
- どうしようもないので適当に焼きなまし法で探索
  - 不等式 A > B があると、エネルギーを B - A の各要素で正であるものの和として適当に探索するなど (これはあんまり頭よくない)

チーム Oh! tomaton では

- 各人が勝手に好きな言語を使っていたので、OCaml, ruby, C++, D, Java などが使われていた



実際のコードから (1)

- なぜか突然 C++ コードを吐く

```
let solve () =
  let str = read_line () in (* read a car *)
  let out_chan = open_out "calc_tmp2.cc" in
  decode str out_chan;
  close_out out_chan;
  ..
```

decode の一部

```
if main_chamber then begin
  Printf.fprintf out_chan "ans += Q(%s, %s);\n" upper lower
end else begin
  Printf.fprintf out_chan "ans += P(%s, %s);\n" upper lower
end;
```

実際のコードから (2)

- g++ を呼び出し始める
- ./annealing で fuel.ml という OCaml ソースが吐かれる

```
let solve () = ... (* さっきの続き *)

(* call g++ *)
let compiled =
  Sys.command("g++ -O3 -Wall annealing.cc -o annealing")
in
if compiled != 0 then
  failwith "compilation failure";

for i = 1 to trynum do
  (* run *)
  let solved = Sys.command("./annealing") in
  ..
```

実際のコードから (3)

- 問題が解けるということは「例外的」な状況らしい
- #use で g++ が吐いたコードを読み込むという似非 eval

```
let _ =
  try
    solve ();
    prerr_endline "COULD NOT BE SOLVED.";
    exit (-1)
  with
    | Exit -> (* seems OK *) ()
;;

#use "tencode.ml"
#use "fuel.ml"
let _ = print_fuel fuel;;
```

この年の問題の裏の意図

- 実は項書換系の停止性問題だったッ！！
  - 車 = 問題インスタンス
  - 燃料 = インスタンスに対する証明
- 項書換系
  - ABC → ACB のような書き換えで計算を行う
- これが止まるかどうか？　を判定する問題

停止性の判定

- 停止することを証明するためには、項が書き換えで常に「小さく」なっていくことを示すのが一つの方法
  - 無限下降列が作れないことを示せばよい
- 車「ABC > ACB」は行列が「小さく」なるということを示している
  - これは書換ルール ABC → ACB が「小さく」なることを示す
- 燃料はそのような行列が「具体的にある」ことを示している

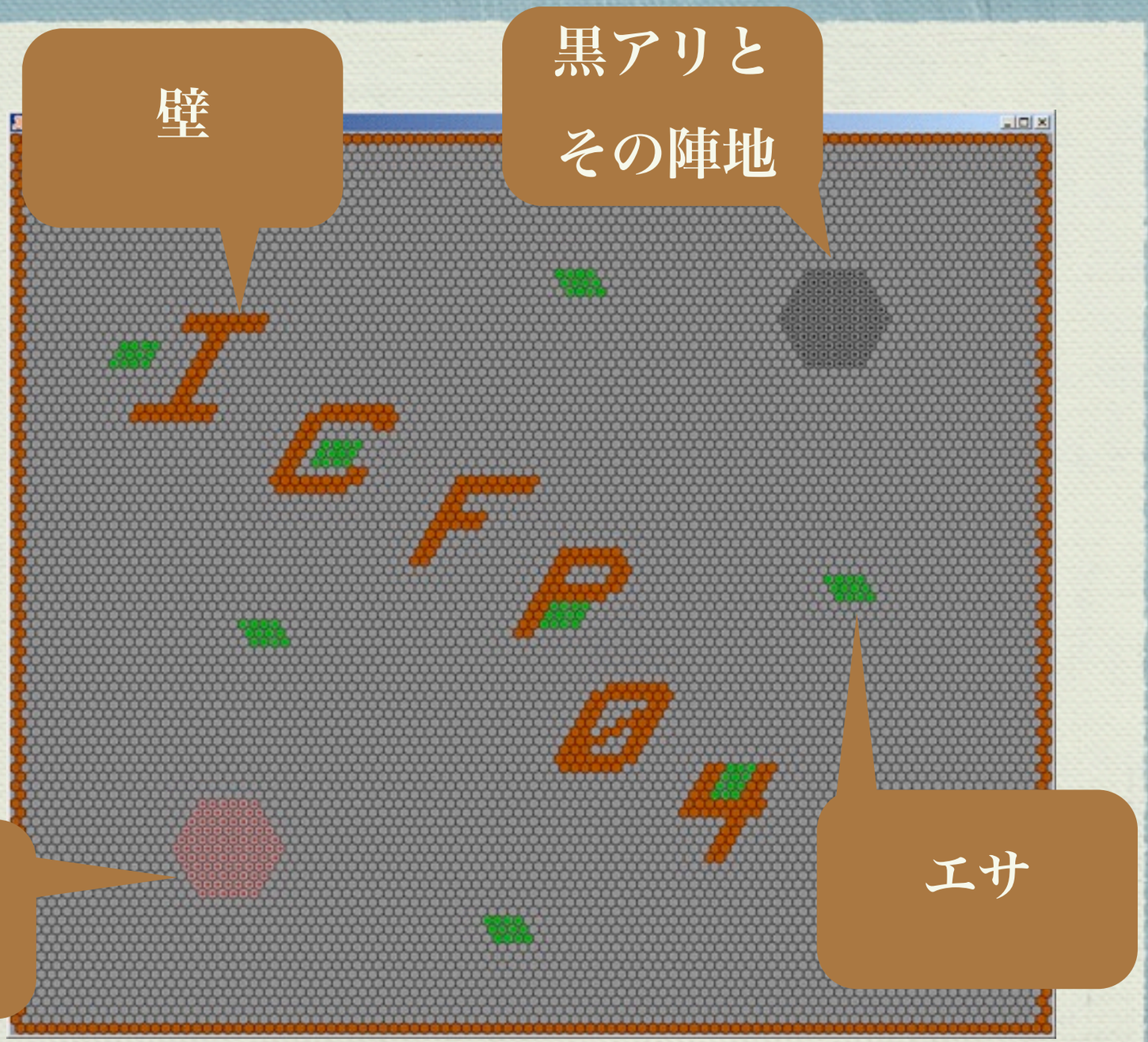
ICFP-PC 2004

- 問題
  - オートマトンでアリを記述して、相手よりも多くえさを集めろ！

- 一人で poppomayaya というチームで参加

マップと対戦形式

- 赤アリと黒アリの対戦形式
- エサを自分の陣地に多く運べば勝ち



アリができること

- Move　　前へ進む
- Turn Right, Turn Left　　右へ/左へ向きを変える
- PickUp/Drop　　エサを取る/落とす
- Marker　　フェロモンを残す(６種類)
- Sense　　周りを調べて条件判断
  - 味方/敵/陣地/壁/フェロモン
- Flip　　乱数を発生

アリ = オートマトン

- このオートマトンを記述する言語は非常に貧弱
- 各行が状態を表し、その遷移先を行番号で指定する

```
Sense Ahead 1 3 Food   ; state 0:
Move 2 0               ; state 1:
PickUp 8 0             ; state 2:
Flip 3 4 5             ; state 3:
Turn Left 0            ; state 4:
Flip 2 6 7             ; state 5:
Turn Right 0           ; state 6:
Move 0 3               ; state 7:
...
```

アリはあまりにも貧弱なので当然高級言語を作る一択

- シミュレータ５、６時間ぐらいで作った後、
  - 自分のソースで 820 行
- OCaml みたいな言語を５、６時間ぐらいで作る
  - 自分のソースで 570 行

作った言語

```
let turnr2 = begin Turn Right; Turn Right end in
let turnl2 = begin Turn Left; Turn Left end in
let turnr3 = begin Turn Right; Turn Right; Turn Right end in

(* Move exactly one step *)
let move_exactly_one = while ! Move do () end in

(* Never Move. *)
let stay = loop Drop end in

let try_walk =
  if ! Move then begin
    if Flip 2 then Turn Left else Turn Right
  end
in
...
```

動かしてみる

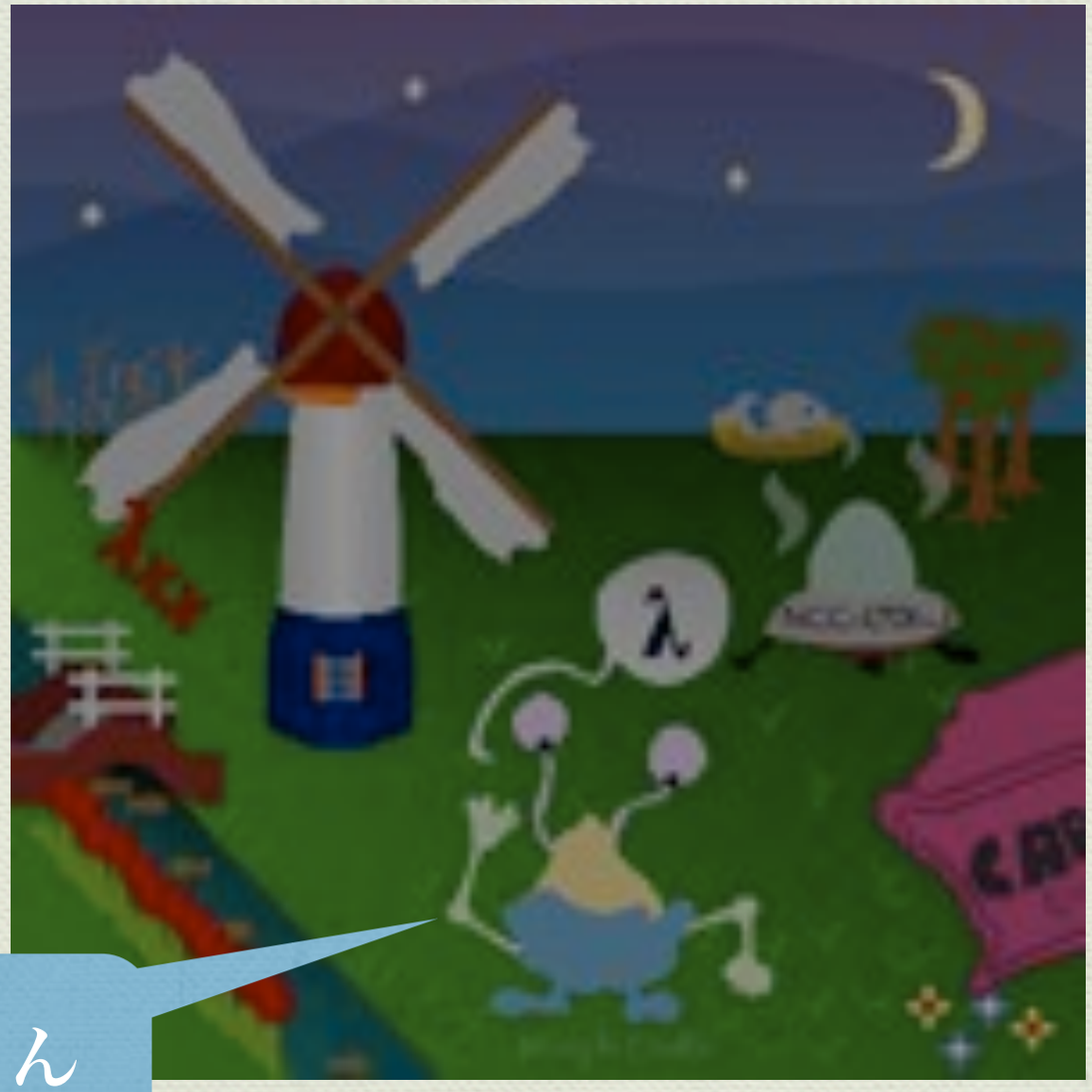
- ソースが全部残っているので当時作ったもののデモを

もう１つぐらい過去の問題を

- 2007 年

ICFP-PC 2007

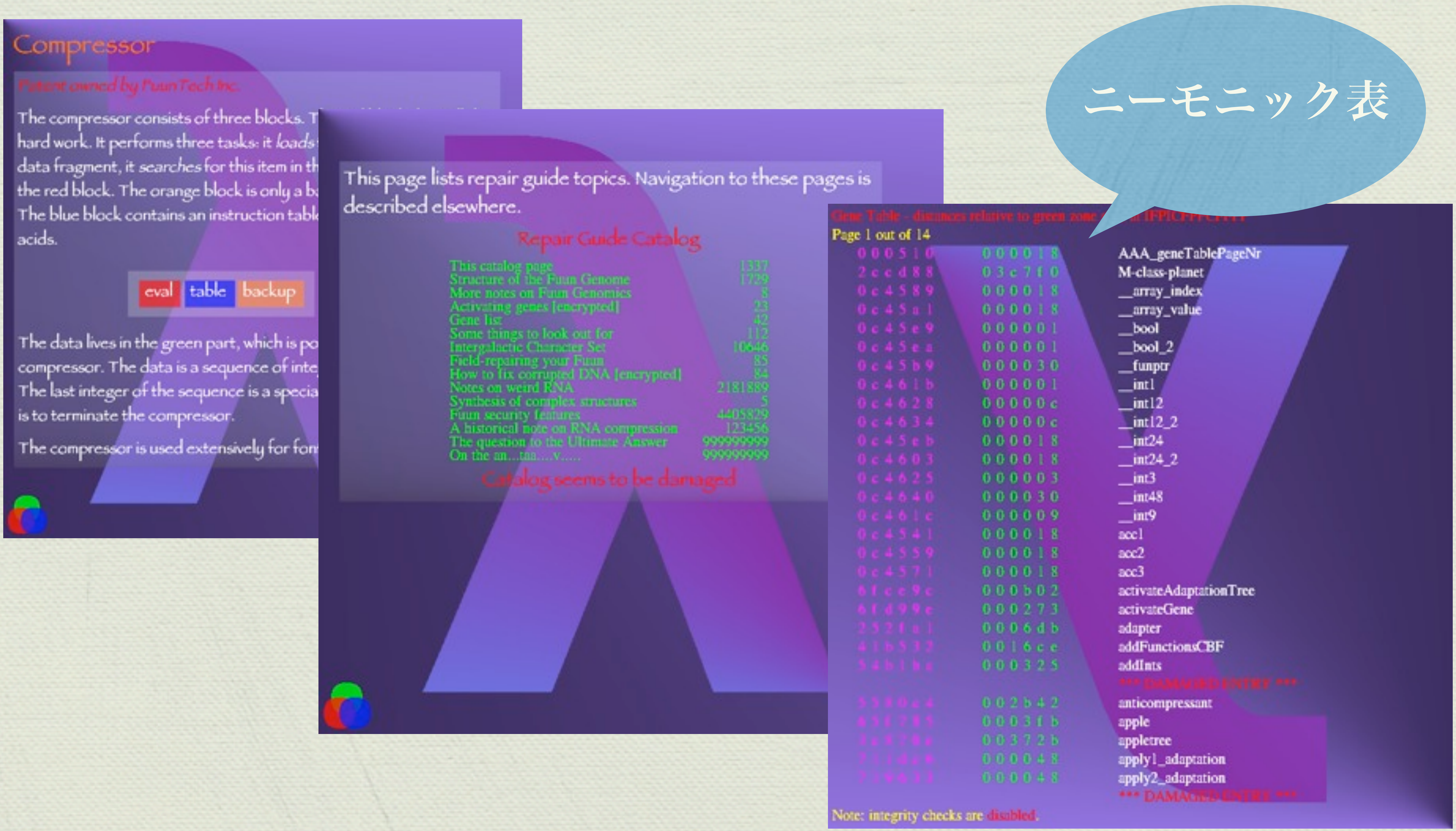
- 問題
  - I, C, F, P からなる DNA を環境に適するように書き換えて、遠藤さんを助けろ！



プログラム解析系の問題

- I, C, F, P からなる DNA は、実はプログラムになっている。
  - まず DNA を実行する VM を作るところから始まる。
- 実行すると絵が得られるので、元絵を書き換えてターゲット絵にする
- プログラムを解析することでいろんなヒントが得られ、必要な絵の一部が手に入ったりする

途中でこんな絵が...


Compressor
Patent owned by FuunTech Inc.
The compressor consists of three blocks.
eval
table
backup
The data lives in the green part
This page lists repair guide topics. Navigation to these pages is described elsewhere.
Repair Guide Catalog
This catalog page 1337
Structure of the Fuun Genome 1729
More notes on Fuun Genomics 8
Activating genes [encrypted] 23
Gene list 42
Some things to look out for 112
Intergalactic Character Set 10646
Field-repairing your Fuun 85
How to fix corrupted DNA [encrypted] 84
Notes on weird RNA 2181889
Synthesis of complex structures 5
Fuun security features 4405829
A historical note on RNA compression 123456
The question to the Ultimate Answer 999999999
On the an...tan....v..... 999999999
Catalog seems to be damaged
ニーモニック表
Page 1 out of 14
AAA_geneTablePageNr
M-class-planet
__array_index
__array_value
__bool
__bool_2
__funptr
__int1
__int12
__int12_2
__int24
__int24_2
__int3
__int48
__int9
acc1
acc2
acc3
activateAdaptationTree
activateGene
adapter
addFunctionsCBF
addInts
anticompressant
apple
appletree
apply1_adaptation
apply2_adaptation
Note: integrity checks are disabled.

OCaml で困った事例集

OCaml で困った事例集 (1)
2008 年 TCP_NODELAY

- TCP 通信が要求され、Nagle のアルゴリズムを切る必要があった
  - OCaml で TCP_NODELAY が使えるのは 3.11 から。

しょうがないのでCをよんでた

cinterface.mli

```
external set_tcp_nodelay :
 Unix.file_descr -> bool -> unit = "stub_set_tcp_nodelay"
```

cinterface.c

```
CAMLprim value stub_set_tcp_nodelay(value socket, value status)
{
  CAMLparam2(socket, status);
  int x = TCP_NODELAY;
  CAMLreturn(setsockopt_int(&x, socket, IPPROTO_TCP, 0,
status));
}
```

OCaml で困った事例集 (2)
絵が...

- 手軽に絵が出せない
- LablTK とか LablGTK とか LablGL とかめんどい
- OCamlGraphics とか...
  - クリックしたらとまったりする...

まとめ

- ライブラリが足らんのです

- ライブラリさえそろえば OCaml はかなり有利
  - 静的型、匿名関数、実行速度
  - さらに速度が必要なら命令型ぽく書けばいいじゃない

- コンテスト楽しいのでみんな出るといいですよ