PostScript
spielt

MASTER MIND

(schon wieder ...)

Die Regeln

- Spielziel: den vom Gegner vorgegebenen Code knacken
- Code: 4 Stifte in jeweils einer von 6 Farben und einer bestimmten Reihenfolge
- 10 Versuche

Die Regeln (2)

- Wie?
  Nach jedem Versuch
  gibt der Gegner Tips:
  - 1 schwarzen Stift
    pro Stift mit
    richtiger Farbe
    und richtiger Position
  - 1 weißen Stift
    pro Stift mit
    richtiger Farbe
    aber *falscher* Position

Stand beim letzten Mal

- Programm knackt vorgegebenen Code
- auch komplexere Varianten, d. h. $n \neq 4$ und / oder $c \neq 6$
- langsam!

- wenig idiomatisch
  - lange Idiome
  - kaum Spracheigenheiten

Optimierung

- algorithmisch
  - großer Aufwand
  - wenig Erfolg
  - nicht der Sinn der Sache
- sprachlich (Code)
  - unerwartete Ergebnisse
    - Ops auf Stack langsamer als auf Arrays
    - je besser faktorisiert desto langsamer
    - ...

Faktorisierung

- sprechende Namen für kurze Codestücke
  - `/pushdown { 1 add 1 roll } bind def`
  - `/pullup { 1 add -1 roll } bind def`
  - `/dup2nd { 1 index exch } bind def`
  - `/array0 { [ exch {0} repeat ] } bind def`
  - `/udef { currentdict exch undef } bind def`
- generische Idiome
  -
```
/aryreplace {
   2 pushdown 2 copy get
   3 pullup exec put
} bind def
```

% ary i {proc} => _

Faktorisierung (2)

- generische Idiome
  - `forall` für 2 Arrays

```
/2forall {                                % Stack: [ary1] [ary2] {proc}
   /_param 3 dict def
    _param /proc 2 pullup put
    _param /a2 2 pullup put
    _param /a1 2 pullup put
   0 1 _param /a1 get length 1 sub {
      dup
      _param /a1 get exch get
      _param /a2 get 2 pullup get
      _param /proc get exec          % Stack (in proc): a1_i a2_i
   } for
   /_param udef
} bind def
```

Spracheigenheiten

Farben – vorher:

```
% color setters - CHANGE GSTATE

/red      {1 0 0 setrgbcolor} bind def
/green    {0 1 0 setrgbcolor} bind def
/blue     {0 0 1 setrgbcolor} bind def
...
/brown    {0.75 0.5 0.25 setrgbcolor} bind
def

% colors (as defined above)
% to use for the pegs and board

/peg-colors
  [ /red /green /blue ... ]
def

/board-color /brown load def
```

Farben – nachher:

```
% a dictionary for various colors

<<
  /red      {1 0 0}
  /green    {0 1 0}
  /blue   {0 0 1}
  ...
  /brown    {0.75 0.5 0.25}
>> begin

% colors (as defined above)
% to use for the pegs and board

/peg-colors
  [ /red /green /blue ... ]
def

/board-color //brown def
```

Code – exact matches

```
% Stack: [code] [code] => exact_matches

/matchingpositions_OLD {
  [ 3 1 roll
    0 1 n 1 sub { % for each position
      3 copy exch 1 index
      get
      3 1 roll
      get
      eq {1 4 1 roll} if
      pop
    } for
    pop pop sum
} bind def
```

```
% Stack: [code] [code] => exact_matches

/matchingpositions_NEW {
  0 2 pushdown {
    eq  { 1 add } if
  } 2forall
} bind def
```

Code – color matches

```
% generates a "histogram" for a given code
% Stack: code => [h_c0 h_c1 ... h_c(c-1)]

/countcolors_OLD {
  [c {0} repeat] exch % init ary w/ 0s
  { % forall positions of the code
    2 copy {1 add} aryreplace
    pop
  } forall
} bind def

/countcolors_NEW {
  c array0 exch
  { % forall positions of the code
    2 copy {1 add} aryreplace
    pop
  } forall
} bind def

% Stack: [code] [code] => color_matches

/matchingcolors_BOTH {
    countcolors exch
    countcolors
    colminsum
} bind def
```

```
% calculates Sum(min(hist1[i],hist2[i]))
% = color matches
% Stack: hist1 hist2 => color_matches

/colminsum_OLD {
  [ 3 1 roll
    aload pop c 1 add -1 roll aload pop
    c { % repeat
      c index min
      c 2 mul 1 roll
    } repeat
    c {pop} repeat
  sum
} bind def

/colminsum_NEW {
    0 2 pushdown
    { min add } 2forall
} bind def

% Stack: [ i1 i2 ... in => sum_i

/sum_OLD {
  counttomark 0 exch {add} repeat
  exch pop
} bind def
```

Code – eligibility

```
% checks if two codes belong to the same
% class (share a certain eval)
% Stack: em cm [code] [code] => bool

/eligible-single_OLD {
  2 copy
  6 -1 roll 3 1 roll
  matchingpositions eq
    { matchingcolors2 eq }
    { pop pop pop false }
  ifelse
} bind def

% checks for consistency with the history
% Stack: [guess] => bool

/eligible-all_OLD {
  history
  { % forall
    dup null eq {pop pop true exit} if
    aload pop 3 index
    eligible-single not {pop false exit}
if
  } forall
} bind def
```

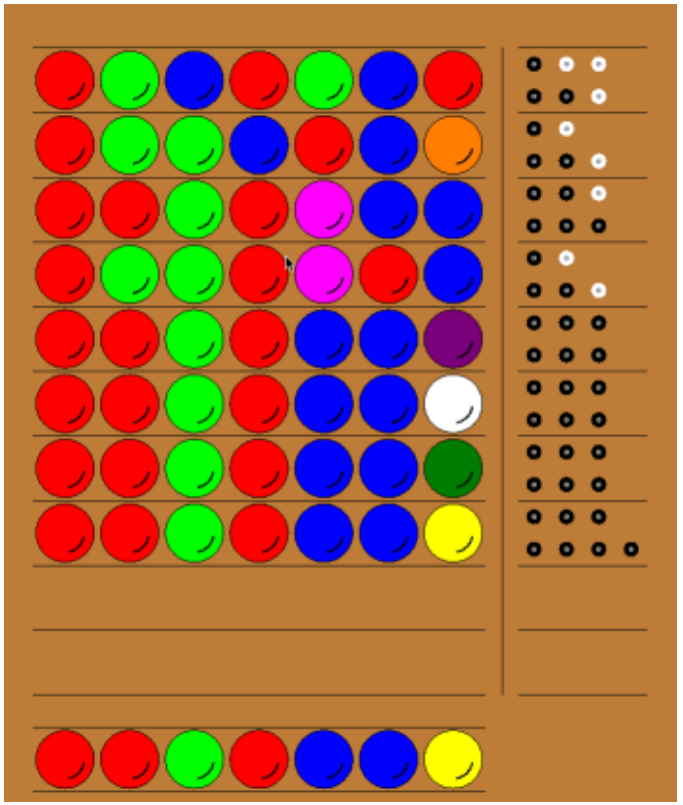
```
% checks if a code is consistent with the
% history re exact matches
% (analoguous for color matches)
% Stack: [code] => bool

/eligible-exact_NEW {
  true exch
  history {
    dup null eq { pop exit } if
    dup2nd
    aload pop
    3 pullup
    matchingpositions
    exch pop
    ne {exch not exch exit} if
  } forall
  pop
} bind def

% checks for consistency with the history
% Stack: [guess] => bool

/eligible-all_NEW {
    dup eligible-exact { eligible-color }
    { pop false } ifelse
} bind def
```

Code – eligibility

```
% checks if two codes belong to the same
% class (share a certain eval)
% Stack: em cm [code] [code] => bool

/eligible-single_OLD {
  2 copy
  6 -1 roll 3 1 roll
  matchingpositions eq
    { matchingcolors2 eq }
    { pop pop pop false }
  ifelse
} bind def

% checks for consistency with the history
% Stack: [guess] => bool

/eligible-all_OLD {
  history
  { % forall
    dup null eq {pop pop true exit} if
    aload pop 3 index
    eligible-single not {pop false exit}
if
  } forall
} bind def
```

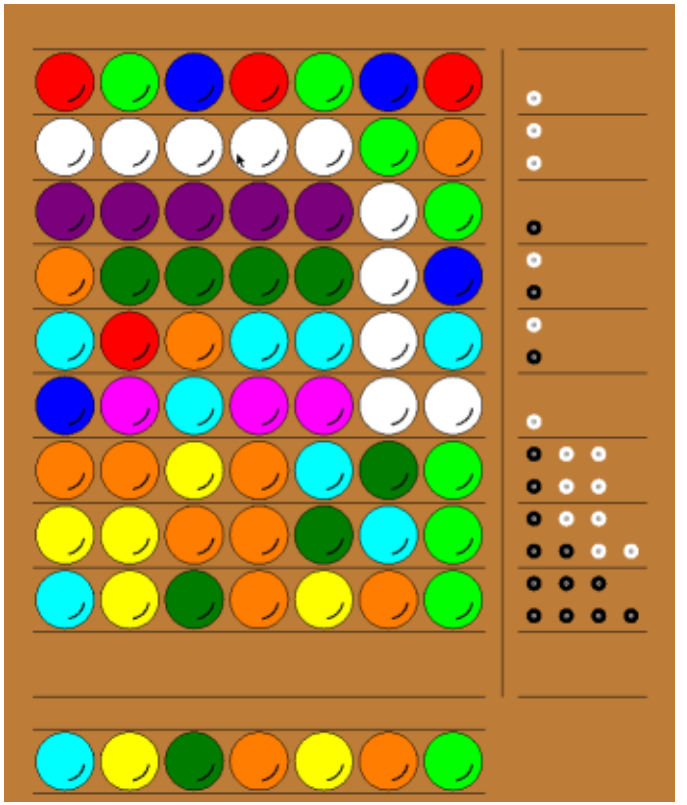
```
% checks if a code is consistent with the
% history re exact matches
% (analoguous for color matches)
% Stack: [code] => bool

/eligible-exact_NEW {
  true exch
  history {
    dup null eq { pop exit } if
    dup2nd
    aload pop
    3 pullup
    matchingpositions
    exch pop
    ne {exch not exch exit} if
  } forall
  pop
} bind def

% checks for consistency with the history
% Stack: [guess] => bool

/eligible-all_NEW {
    dup eligible-exact { eligible-color }
    { pop false } ifelse
} bind def
```

Der Lohn der Mühen ...



[6374341] 7/10

297 vs. 365



[0010223] 7/10

65 vs. 134

Sonstige Quellennachweise

- Bild 1 (MasterMind Schachtel): Images of Master Mind, http://www.tnelson.demon.co.uk/mastermind/images1.html (images/mastermind7.jpg)
- Bild 2 (SuperHirn Brett): Doppelplusspiel, http://home.pages.at/ottodix/ (superhirn02.JPG)