I Can See The Pixels:
Designing Cross-Stitch Patterns in OCaml

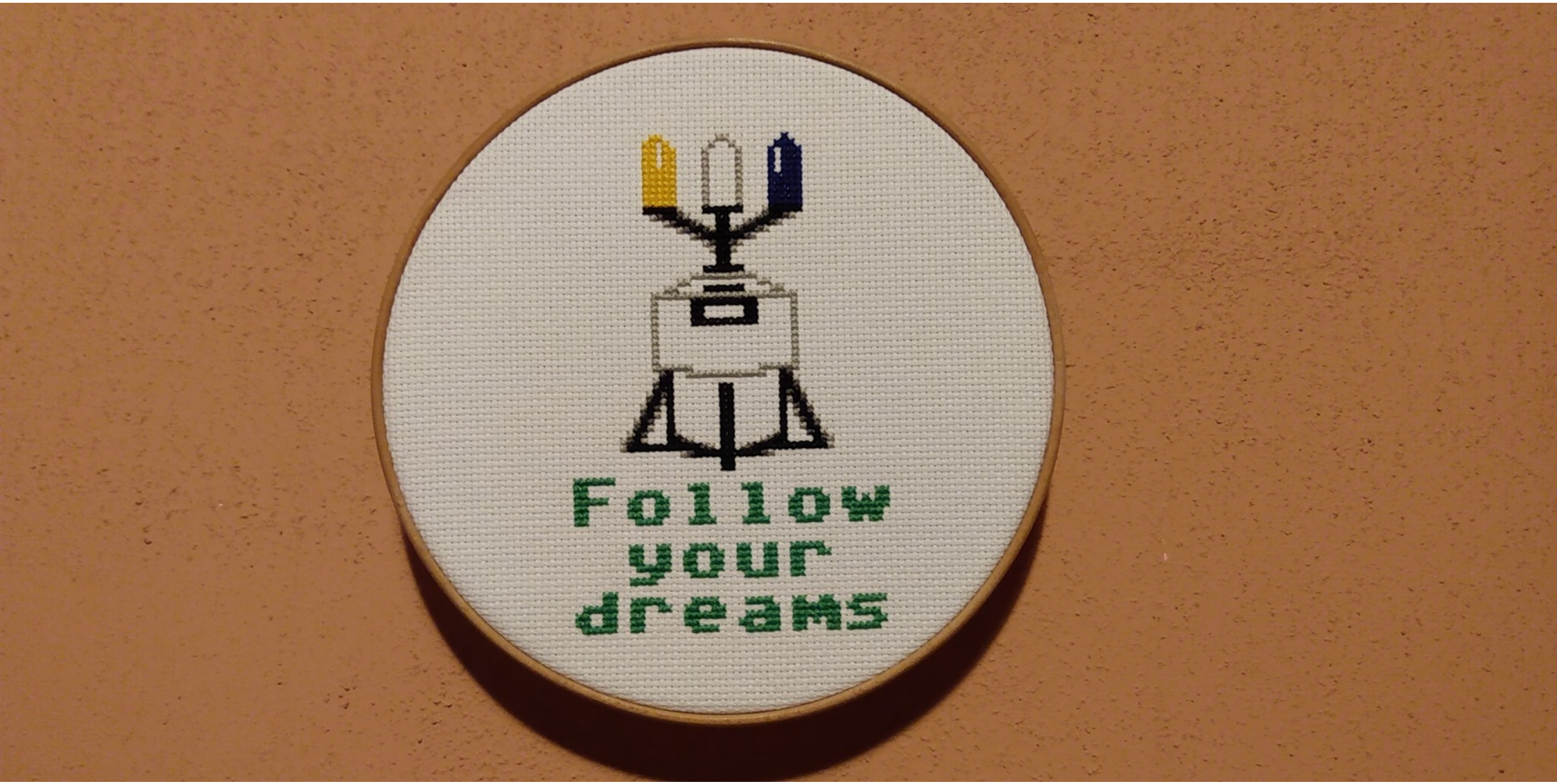
FUN OCaml 2025

Mindy Preston (`yomimono`)
http://wandering.shop/@yomimono

this presentation is about "fun"

```
type fun =
| Immediate
| Delayed

"type 1" fun and "type 2" fun
```


Follow
your
dreams

for cross-stitch, you need a substrate:

```
(* this is rather unimaginative ;) *)
type grid = | Fourteen | Sixteen | Eighteen

type substrate =
  { background : RGB.t; (* this is fancy (int * int * int) *)
    grid : grid;
    (* farthest coordinate on each axis (least is always 0) *)
    max_x : int; [@generator Crowbar.range 1023]
    max_y : int; [@generator Crowbar.range 1023]
  }
```

you need some thread:

```
type thread = DMC.Thread.t
[@@deriving eq, yojson]
```

```
module Thread : sig
  include Thread.S
  val compare : t -> t -> int
end = struct
  type t = { name : string; (* prose name (e.g. "Lavender-VY DK") *)
             identifier : string; (* floss number (except white & ecru) *)
             rgb : RGB.t;
           } [@@deriving yojson]
```

```
(* mappings from schemes/dmc.xml in kxstitch *)
let (rgb_map, id_map) =
  let rgb = RGBMap.empty and id = StringMap.empty in
  let (rgb, id) = add_thread rgb id "Blanc" "White" (252, 251, 248) in
  let (rgb, id) = add_thread rgb id "White" "White" (252, 251, 248) in
  let (rgb, id) = add_thread rgb id "B5200" "Snow White" (255, 255, 255) in
  let (rgb, id) = add_thread rgb id "Ecru" "Ecru" (240, 234, 218) in
  let (rgb, id) = add_thread rgb id "150" "Dusty Rose Ult Vy Dk" (171, 2, 73) in
  let (rgb, id) = add_thread rgb id "151" "Dusty Rose Vry Lt" (240, 206, 212) in
  let (rgb, id) = add_thread rgb id "152" "Shell Pink Med Light" (226, 160, 153) in
```

there are a lot of stitches we could make:

```
type cross_stitch =
  | Full (* X *) (* full stitch *)
    (* half stitches *)
  | Backslash (* \ *) (* upper left <-> lower right *)
  | Foreslash (* / *) (* lower left <-> upper right *)
    (* quarter stitches *)
  | Backtick (* ` (upper left quadrant) *)
  | Comma (* , (lower left quadrant) *)
  | Reverse_backtick (* mirrored ` (upper right quadrant) *)
  | Reverse_comma (* mirrored , (lower right quadrant) *)
[@@deriving eq, yojson]

type stitch = | Cross of cross_stitch
[@@deriving eq, yojson]
```

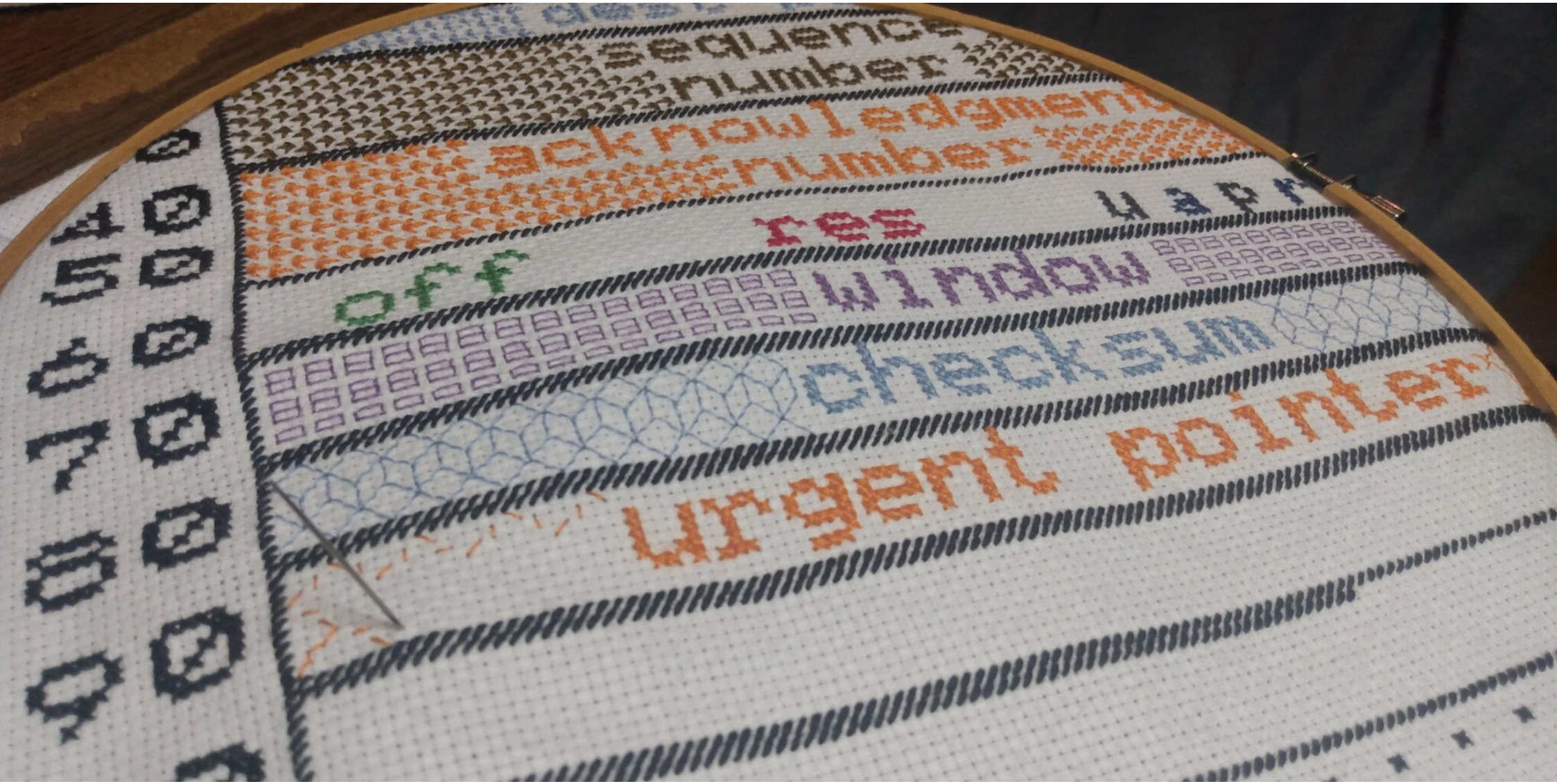
but we almost always use 'full cross stitch'

(backstitch doesn't get a fancy type)

we'll imagine the substrate
as having two grids,
for two kinds of stitches:

```
type coordinates = int * int [@@deriving yojson]
type segment = coordinates * coordinates [@@deriving yojson]
module Coordinates : Map.OrderedType with type t = coordinates
module CoordinateSet : sig
  include Set.S with type elt = coordinates
  val to_yojson : t -> Yojson.Safe.t
  val of_yojson : Yojson.Safe.t -> (t, string) result
end
module SegmentSet : sig
  include Set.S with type elt = segment
  val to_yojson : t -> Yojson.Safe.t
  val of_yojson : Yojson.Safe.t -> (t, string) result
end
```

(here's an example, since that's hard to visualize)


sequence
number
acknowledgment
number
res
off
window
checksum
urgent pointer

the pattern is an expression of where to put which stitch

```
type layer = {
  thread : thread;
  stitch : stitch;
  stitches : CoordinateSet.t;
} [@@deriving yojson]

type backstitch_layer = {
  thread : thread;
  stitches : SegmentSet.t;
} [@@deriving yojson]

type pattern = {
  substrate : substrate;
  layers : layer list; [@default []]
  backstitch_layers : backstitch_layer list; [@default []]
} [@@deriving yojson]
```

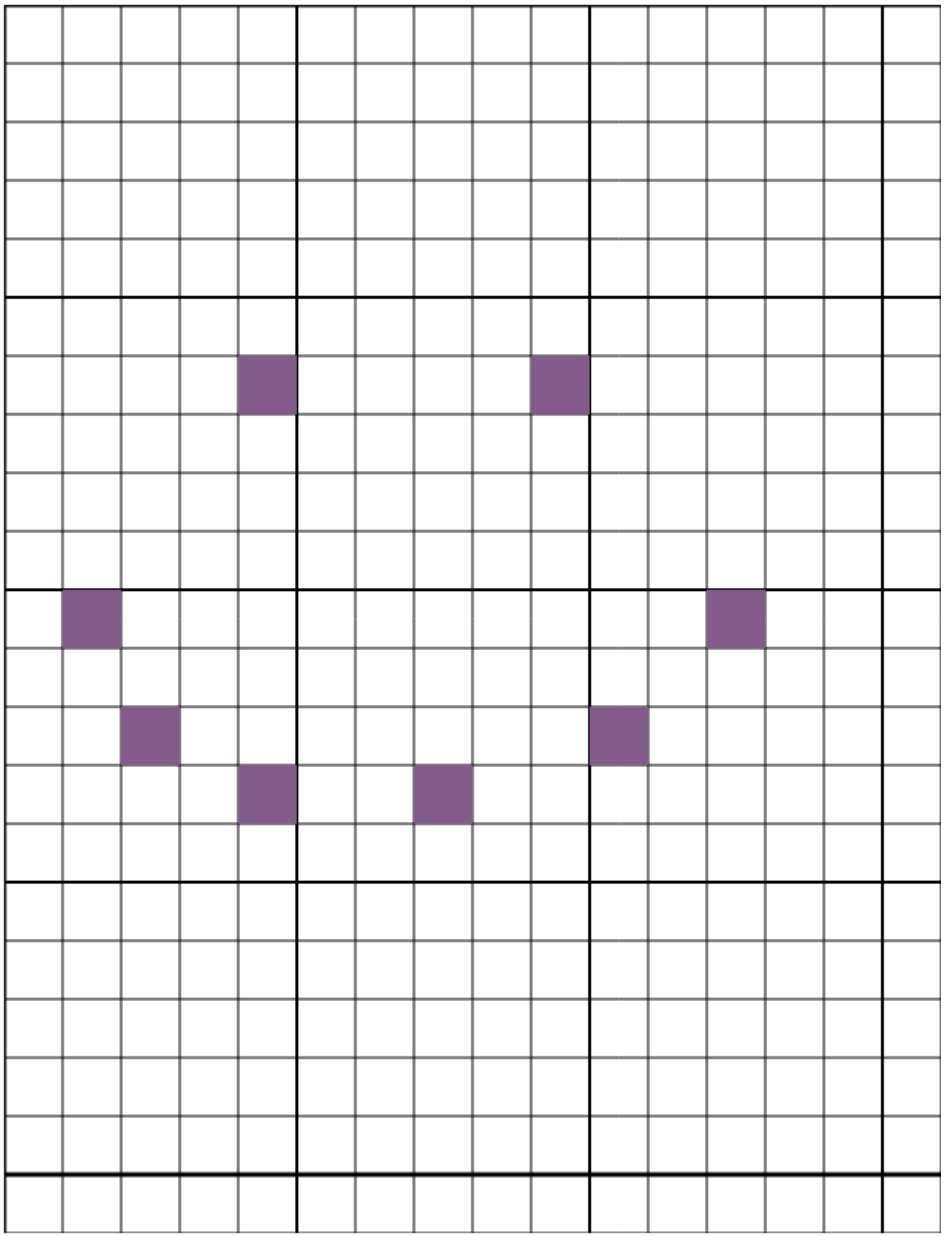
so let's make one

first 'fun': js_png_canvas
(an excuse to use js_of_ocaml)

Stitchcraft Cloud Experience for Makers 2022 Platinum Edition (TM)(R) — Mozilla Firefox

Stitchcraft Cloud Experience fo ×

http://localhost:8080/edit



eraser display json

{"substrate":{"background":[255,255,255],"grid":["Fourteen"],"max_x":20,"max_y":20},"la

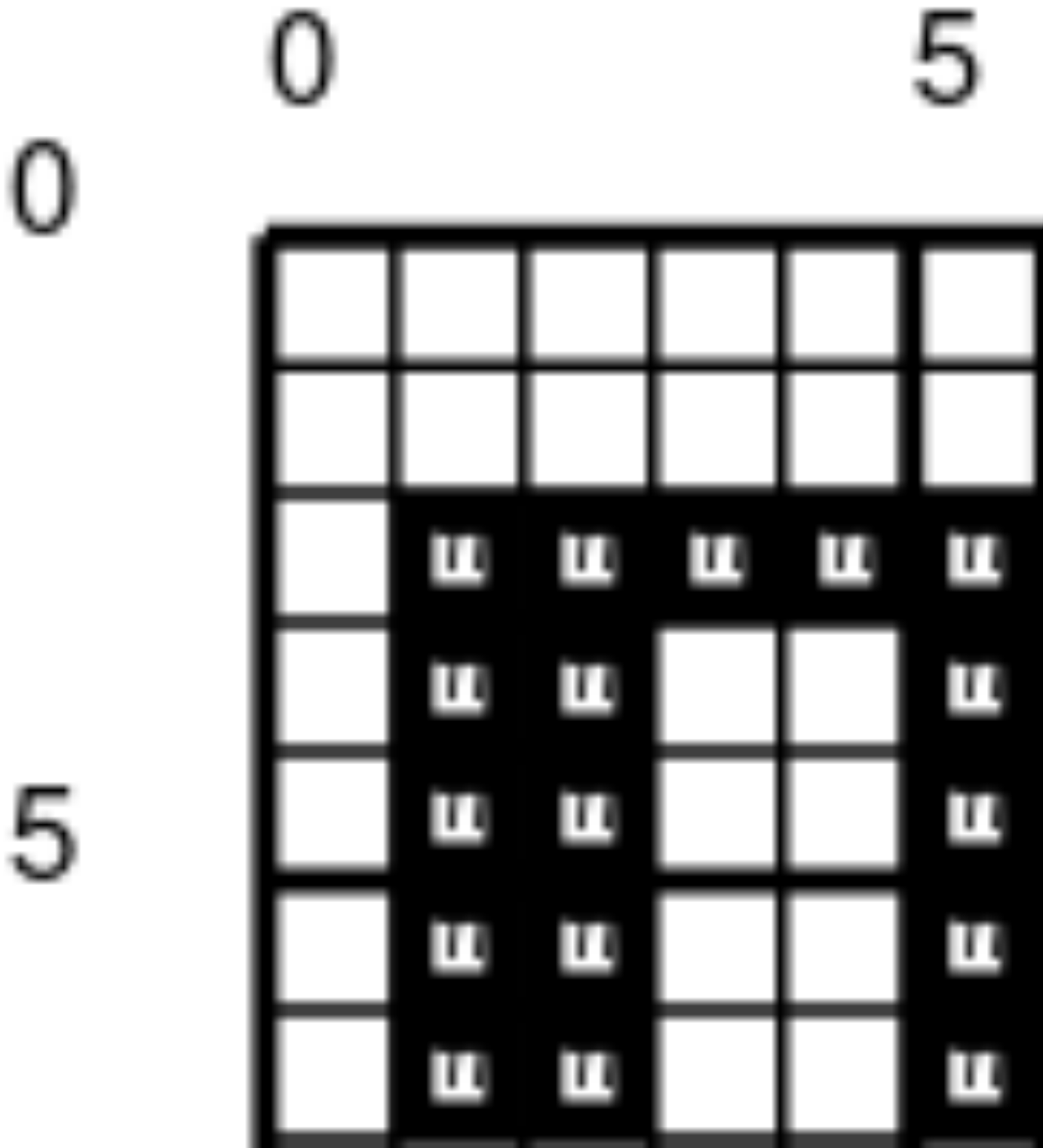
fun level insufficient, abandoned for years

second 'fun': pdf output
(an excuse to use camlpdf)

...pdfs are complicated

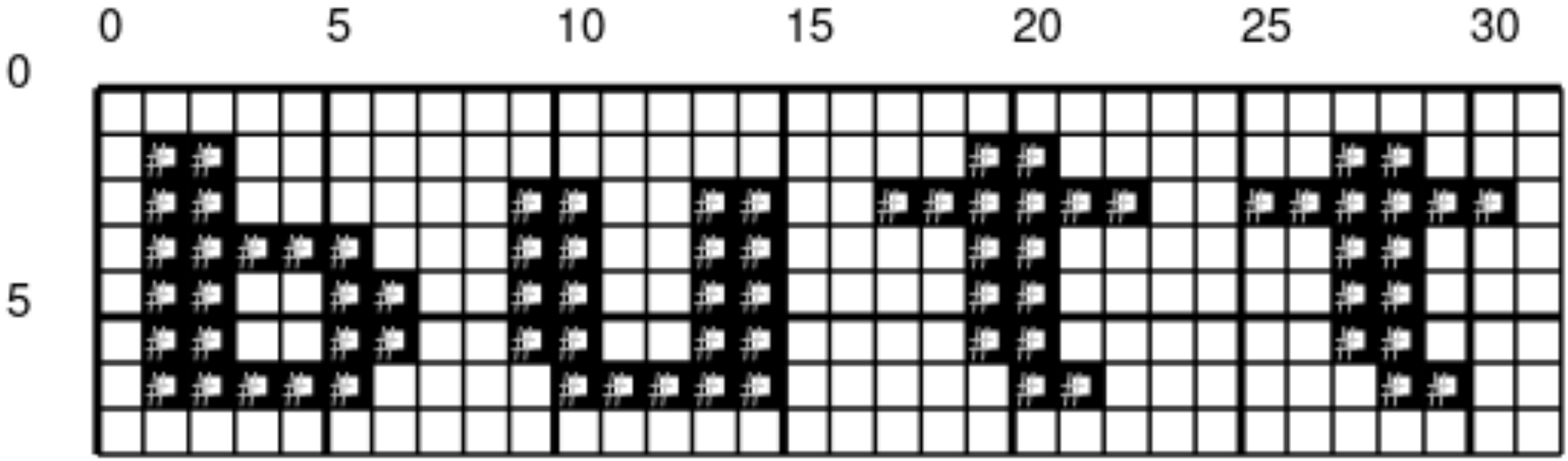
```
let paint_pixel ~font_size ~pixel_size ~x_pos ~y_pos r g b symbol =
  let stroke_width = 3. in (* TODO this should be relative to the thickness of fat lines *)
  let (font_key, symbol) = Font.key_and_symbol symbol in
  let font_stroke, font_paint =
    let r, g, b = Colors.ensure_contrast_on_white (r, g, b) in
      Pdfops.Op_RG (r, g, b), Pdfops.Op_rg (r, g, b)
  in
  let font_location =
    (* y_transform gives us the offset to draw our character in a vertically centered location *)
    let y_transform = Pdfstandard14.baseline_adjustment Pdftext.ZapfDingbats |> float_of_int |> (/.) 1000. in
    (* we can get the text width in millipoints directly *)
    let symbol_width = (Pdfstandard14.textwidth false Pdftext.ImplicitInFontFile Pdftext.ZapfDingbats symbol)
|>
                        float_of_int |> (/.) 2000. in
    Pdftransform.Translate
      ((x_pos +. ((pixel_size *. 0.5) -. symbol_width)),
       (y_pos -. pixel_size *. 0.5 -. y_transform))
  in
  Pdfops.([
      Op_q;
      Op_w stroke_width;
      Op_s;
      Op_cm
        (Pdftransform.matrix_of_transform [font_location]);
      font_stroke;
      font_paint;
      Op_Tf (font_key, (float_of_int font_size));
      Op_BT;
      Op_Tj symbol;
      Op_ET;
      Op_Q;
    ])
```

but rewarding!


0
5
0
5

cross-stitch is often text

`c64stitch "butt" | stitchpattern`


0
5
10
15
20
25
30
0
5

butt

this needs a TUI
(an excuse to use notty)

```
          012345678901234567890 12                    # DMC 310: Black
 0123456789111111111122222222222333
0
1 ##                  ##      ##
2 ##      ##  ##  ######  ######
3 #####   ##  ##    ##      ##
4 ##  ##  ##  ##    ##      ##
5 ##  ##  ##  ##    ##      ##
6 #####    #####     ##      ##
7
8
Quit Solid view
```

much better :)

programming with `notty`:
nice concatenation, I want some

```
(alias
 (name be_gay_do_crimes)
 (action
  (progn
   (run c64stitch -o red.json -t red "2")
   (run c64stitch -o orange.json -t orange "3")
   (run c64stitch -o yellow.json -t yellow "5")
   (run c64stitch -o green.json -t green "7")
   (run c64stitch -o hblue.json -t blue "11")
   (run c64stitch -o hpurple.json -t purple "13")
   (run c64stitch -o black.json "\\")

   (run c64stitch -o vblueone.json -t blue "1")
   (run hcat -o vblueeleven.json vblueone.json vblueone.json)
   (run c64stitch -o vpurpleone.json -t purple "1")
   (run c64stitch -o vpurplethree.json -t purple "3")
   (run hcat -o vpurplethirteen.json vpurpleone.json vpurpleth
ree.json)

   (run vcat -o vro.json red.json orange.json)
   (run vcat -o vroy.json vro.json yellow.json)
   (run vcat -o vroyg.json vroy.json green.json)
   (run vcat -o vroygb.json vroyg.json hblue.json)
   (run vcat -o vroygbv.json vroygb.json hpurple.json)

   (run hcat -o hro.json red.json orange.json)
   (run hcat -o hroy.json hro.json yellow.json)
   (run hcat -o hroyg.json hroy.json green.json)
   (run hcat -o hroygb.json hroyg.json vblueeleven.json)
   (run hcat -o hroygbv.json hroygb.json vpurplethirteen.json)

   (run c64stitch -o be_gay.json "be gay")
   (run c64stitch -o find_primes.json "find primes")
   (run hcat -o saying.json be_gay.json find_primes.json)
   (run embellish_stitch --top=vroygbv.json --side=hroygbv.jso
n --corner=black.json --center=saying.json -o be_gay_find_prim
es.json)
   (run stitchpattern -i be_gay_find_primes.json -o be_gay_fin
d_primes.pdf)
   (run evince be_gay_find_primes.pdf)
        )))
```

hideous! but effective:


2357 1113
2357 1113
be
gay
find
primes

suddenly it's useful!

(but only if you really like the Commodore 64 font)

```
type glyph = {
  stitches : CoordinateSet.t;
  backstitches : SegmentSet.t;
  height : int;
  width : int;
} [@@deriving yojson]

module UcharMap : sig
  include Map.S with type key = Uchar.t
end

type font = glyph UcharMap.t [@@deriving yojson]
```

...text is complicated

```
    List.fold_left (fun (x_off, y_off, missing_chars, stitches, backst
itches, max_x, max_y) uchar ->
        match Uucp.Gc.general_category uchar with
        | `Zl | `Cc when Uchar.to_char uchar = '\n' ->
          let _, height = get_dims lookup default_char in
          let height = max min_height height in
          let y_increase = height + interline in
          (0, y_off + y_increase, missing_chars,
           stitches, backstitches, max_x, max_y + y_increase)
        | `Ll | `Lm | `Lo | `Lt | `Lu
        (* for the moment, we ignore all combining marks *)
        (* there are many fonts for which we could do the right thing
here -- TODO *)
        | `Nd | `Nl | `No
        | `Pc | `Pd | `Pe | `Pf | `Pi | `Po | `Ps
        | `Sc | `Sk | `Sm | `So
        | `Zs ->
          (* TODO: we should probably center or something when given a
 min_dimension
           * larger than the one we looked up? *)
          let width, _ = get_dims lookup uchar in
          let width = max min_width width in
          let new_max_x = max (x_off + width) max_x in
          let missing_chars, stitches, backstitches = add_stitches_for
_glyph ~x_off ~y_off uchar missing_chars stitches backstitches in
          ((x_off + width), y_off, missing_chars, stitches, backstitch
es, new_max_x, max_y)
        | _ -> (* not a lot of chance we know what to do with this; ig
nore it *)
          (x_off, y_off, missing_chars, stitches, backstitches, max_x,
 max_y)
      ) (0, 0, [], empty_layer, empty_bs_layer, starting_x, starting_y
```

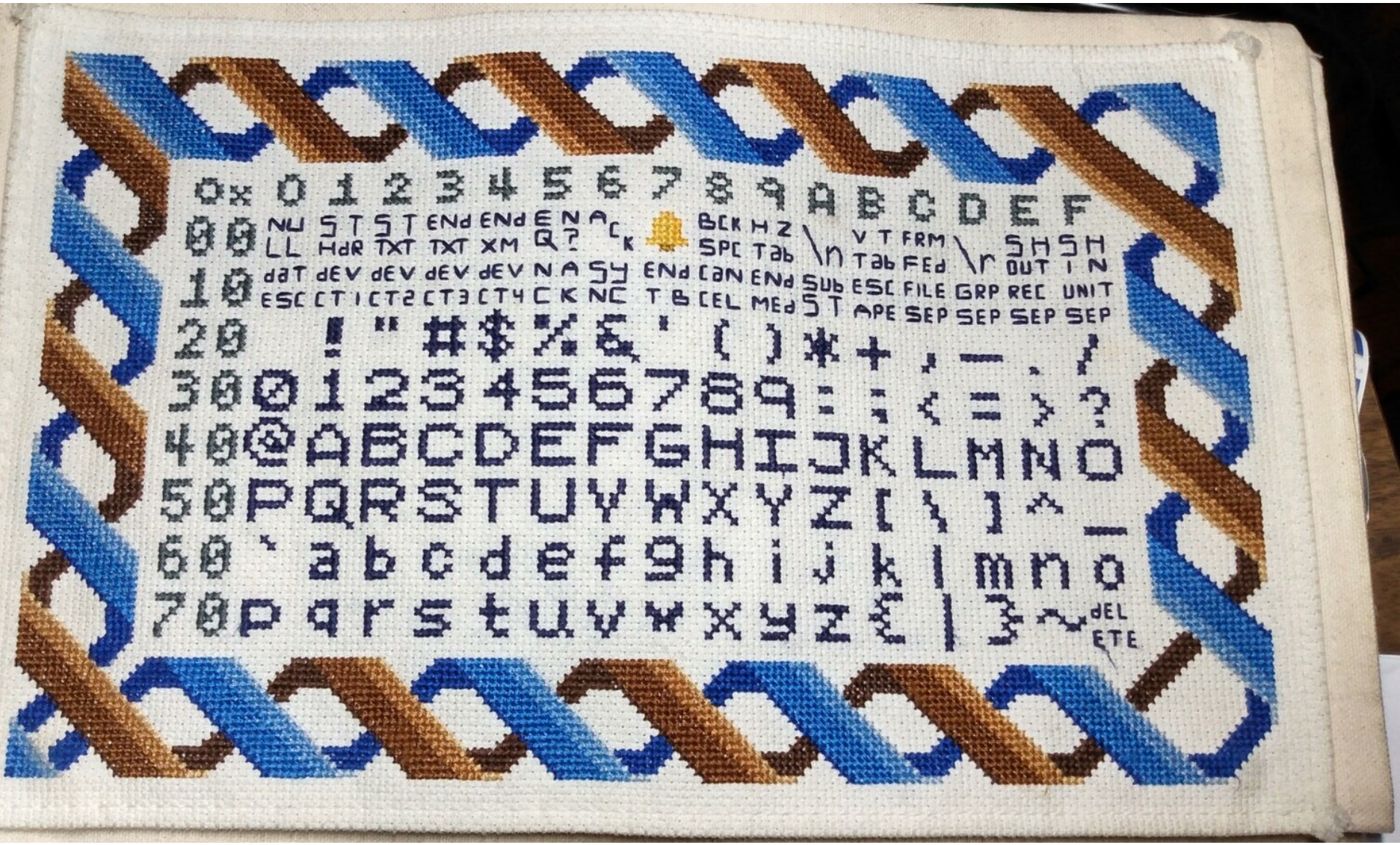
but rewarding!


in this house
we use
monospaced
fonts

bitmap fonts are for weirdos (complimentary)
many of them like defining their own file formats

```
[yomimono@halftop stitchcraft] $ find fontreader/li
b -name '*2stitchfont.ml' -exec  wc -l {} \;
105 fontreader/lib/psf2stitchfont.ml
82 fontreader/lib/otf2stitchfont.ml
82 fontreader/lib/js2stitchfont.ml
253 fontreader/lib/yaff2stitchfont.ml
```

counting is terrible


0x 0 1 2 3 4 5 6 7 8 9 A B C D E F
00 NULL SOH STX ETX EOT ENQ ACK BEL BS HT LF VT FF CR SO SI
10 DLE DC1 DC2 DC3 DC4 NAK SYN ETB CAN EM SUB ESC FS GS RS US
20 ! " # $ % & ' ( ) * + , - . /
30 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
40 @ A B C D E F G H I J K L M N O
50 P Q R S T U V W X Y Z [\ ] ^ _
60 ` a b c d e f g h i j k l m n o
70 p q r s t u v w x y z { | } ~ DEL

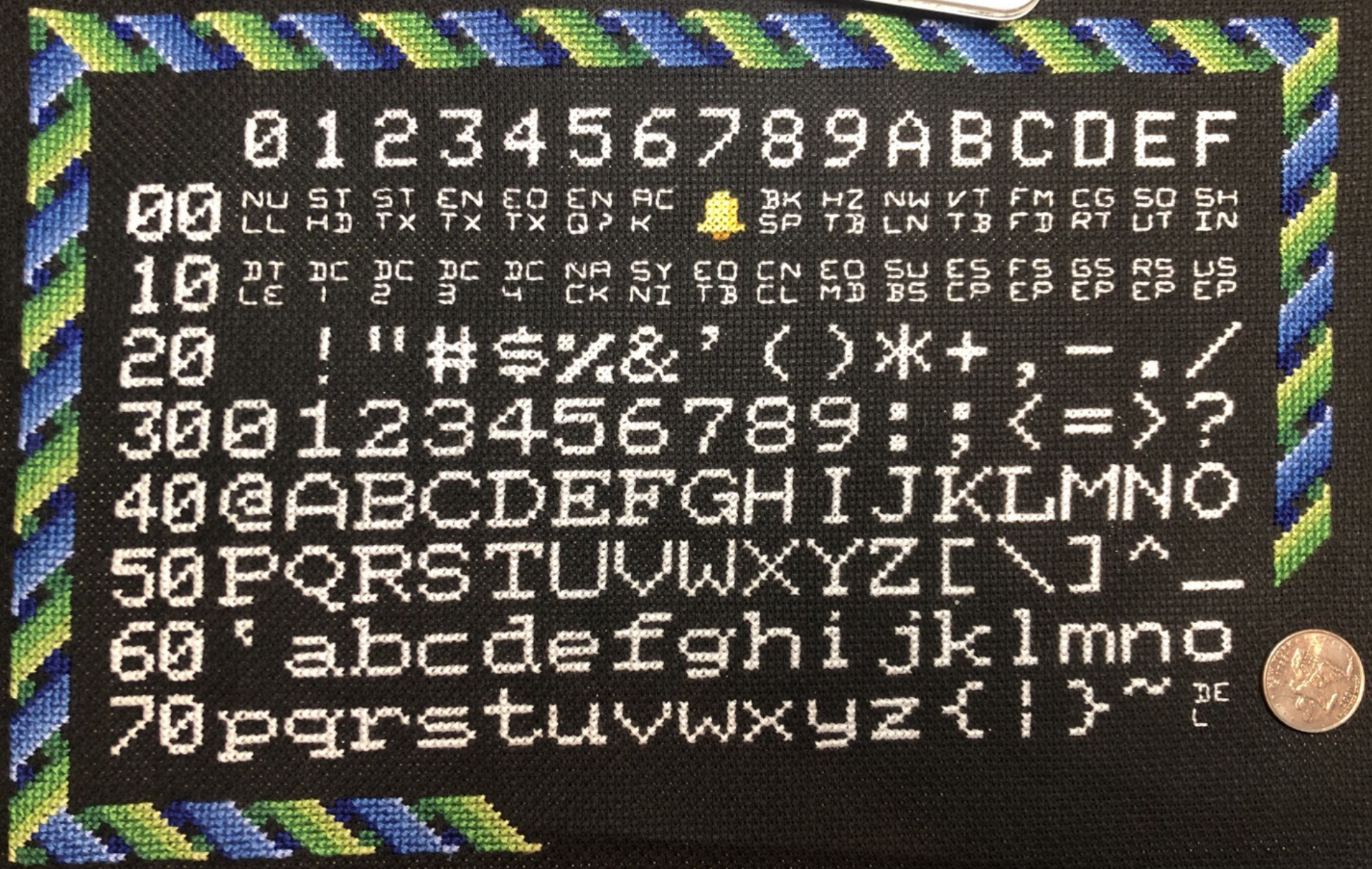
let's make the computer do it

what is a border?

```
type transformation = | Turn | Flip | Nothing [@@deriving yojson]
type transformable_pattern = {
  transformation : transformation; [@default Nothing]
  pattern : pattern;
} [@@deriving yojson]

type border = {
  corner : transformable_pattern;
  side : transformable_pattern option;
  fencepost : transformable_pattern option;
} [@@deriving yojson]
```


(@ 0123456789 ?)
! a------ghi j-lm [
n-pq---t-uwxyz]
f u c k
borders
worked by yoninono
2021 CE, aged 36
no gods no masters


0123456789ABCDEF
00 NU ST ST EN EO EN AC BK HZ NW VT FM CG SO SH
LL HD TX TX TX QK SP TB LN TB FD RT UT IN
10 DT DC DC DC DC NA SY EO CN EO SU ES FS GS RS US
LE 1 2 3 4 CK NI TB CL MD BS CP EP EP EP EP EP
20 !"#$%&'()*+,-./
3001234567 89:;<=>?
40@ABCDEFGHIJKLMNO
50PQRSTUVWXYZ[\]^_
60`abcdefghijklmno
70pqrstuvwxyz{|}~ DEL

dangerous non-fun: "you should sell these on etsy"

someone did buy the "butt" PDF for USD 0.25

`c64stitch "butt" | pdfstitch | upload_to_etsy`

oauth2 is complicated

and not all that rewarding

what is this?

An OAuth2 authorization server for communicating with etsy.com implemented in MirageOS. Etsy's authentication flow may be similar enough to other OAuth2 resource servers to make this server useful for them as well.

what do I need to run this?

- an Etsy developer key
- a publicly-registered FQDN corresponding to a public IP where you can run a unikernel

and in order to do it I had to store stuff

filesystems are complicated

[censored]

and kind of rewarding?

Chamelon: MVP persistent block storage for MirageOS

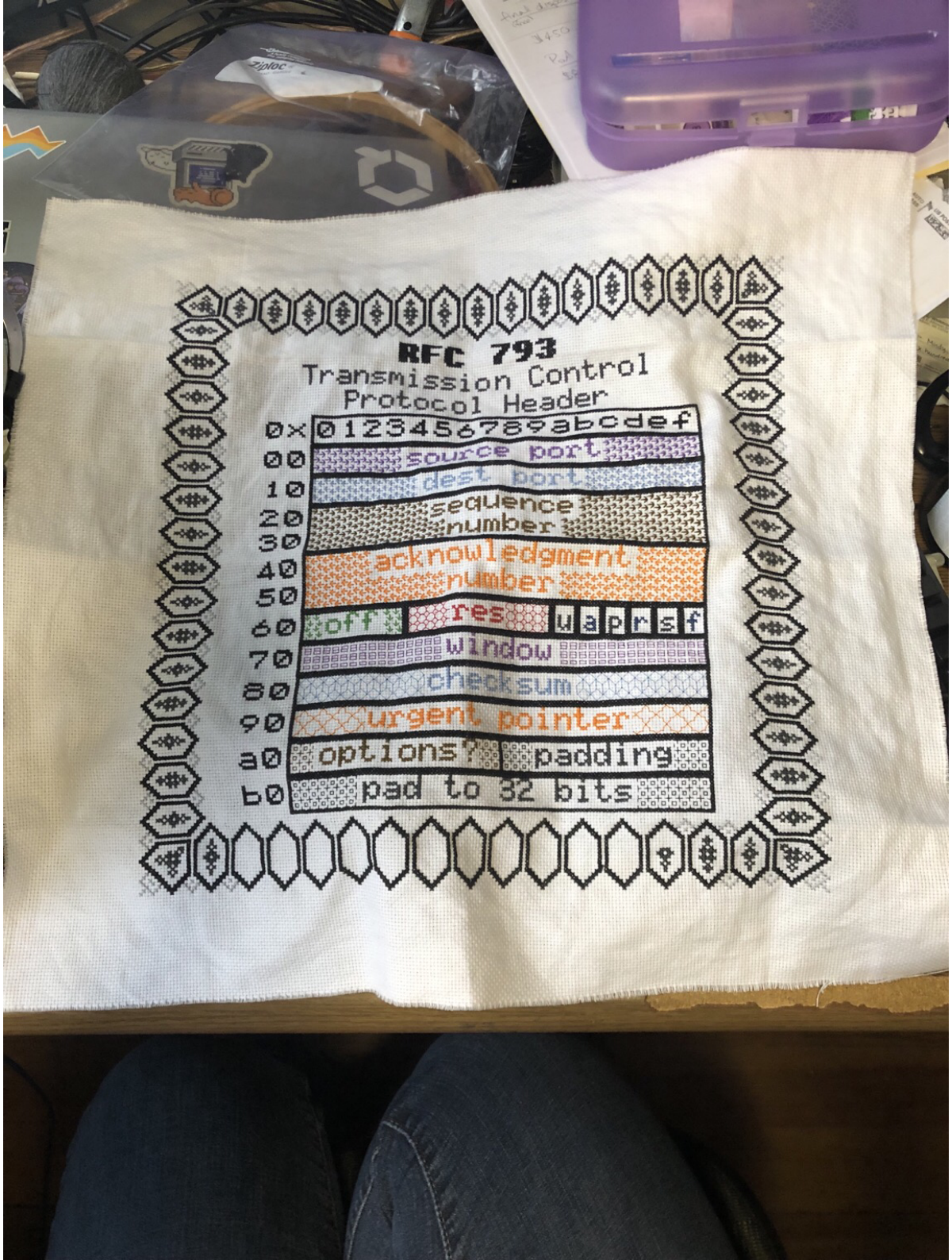
TL;DR: I wrote a **key-value store** for **MirageOS** backed by **block storage**. It's called **chamelon**, it's based off **LittleFS**, and if you're brave, you can use it to store data. Examples are available: **a URL shortener** and an **OAuth2 authorization server**.

In English: I couldn't save or load files before, and now I could. Wowzers!

Read more...

but I never deployed any of this stuff

instead I cross-stitched a big TCP header


RFC 793
Transmission Control
Protocol Header
0x 0123456789abcdef
00 source port
10 dest port
20 sequence
30 number
40 acknowledgment
50 number
60 off res u a p r s f
70 window
80 checksum
90 urgent pointer
a0 options? padding
b0 pad to 32 bits

nice border, where'd you steal it from?

Alexandre	13	87	88	98	102	109	111	113	114	115	116	117	118	119	120	122*	132
	133	138	143	145	159	160	173*	179*	182	232	234	235	238*	249			
L.V.	8	56	100	206	209	210	212	213									
Rouyer	10*	14	21*	26	29	31	37	44	99	145	231	241	248	254*	257	258	260*
	263	264	265														
Sajou	1	2	3	4	5	6	7	8		31*	32*		51	52	53	54	55
	56		62c*	76*	77*	78*	79*		91*		101	102	103	104	105	106	107
	108	109	110		113		131*	132*	134*	135*	136*	150	151	152		157s	159s
	163s		160	161	163	170	171	172	173	174		181	182	184	185	186	
	201	202	203	203	204	205	206		231*	233*	235*	236*		251	252	253	254
	256	290	301c*	302c*	303c*	304c*	305c*	306c*	307c*	309c*	310c*	321	322	323	324	325	326
	341*	342*	343*	344*	345*	346*			361	362	363	364	366	440	451	452	454
	455	456	457	481*	484*	486*	502	504	505	601	602	603	604	605	606	615*	620*
	622*	651	652	653	654	655	656	657	658		661*	663*	664*	665*	666*		

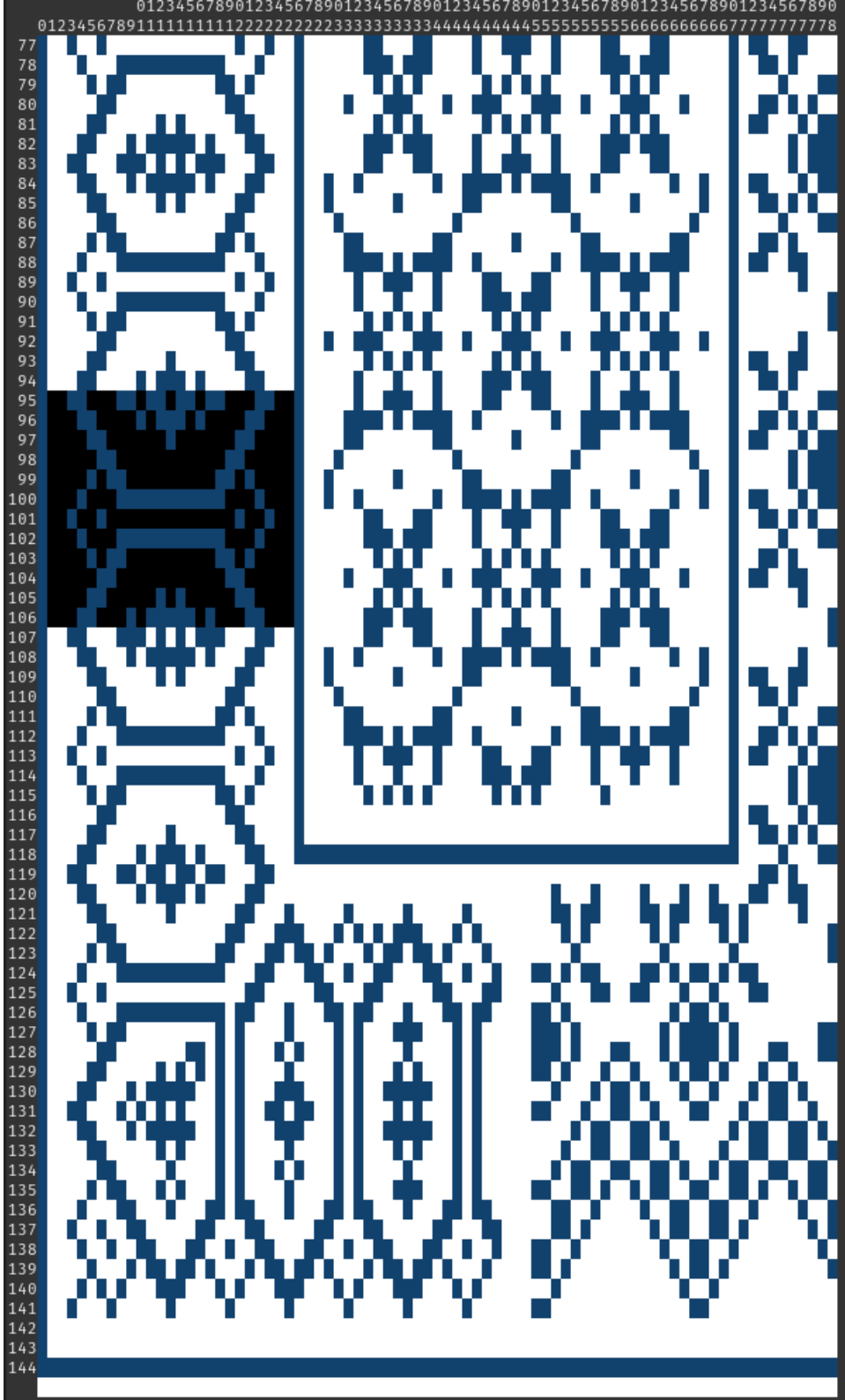
(check them out yourself at
http://patternmakercharts.blogspot.com)

the proprietary file format is not too complicated

and extremely rewarding

maybe too rewarding

this needs a browser
(an excuse to emulate midnight commander)
(and an excuse to use notty again)



Sajou_657_3.pat.pattern
Sajou_657_4.pat.pattern
Sajou_657_6.pat.pattern
Sajou_658_page_2.pat.pattern
Sajou_6_2.pat.pattern
Sajou_8_03.pat.pattern
Sajou_8_05.pat.pattern
Sajou_8_07.pat.pattern
Sajou_No_007_-_3.pat.pattern
Sajou_No_007_-_4.pat.pattern
Sajou_No_007_-_5.pat.pattern
Sajou_No_007_-_6.pat.pattern
Sajou_No_172_-_7.pat.pattern
Sajou_No_201_-_02.pat.pattern
Sajou_No_201_-_03.pat.pattern
Sajou_No_321_2.pat.pattern
Sajou_No_321_6.pat.pattern
Sajou_No_363_2.pat.pattern
Sajou_No_504_-_1.pat.pattern
Sajou_No_504_-_3.pat.pattern
Sajou_No_504_-_4.pat.pattern
Sajou_No_504_-_8.pat.pattern
Sajou__604_03.pat.pattern
Sajou__604_09.pat.pattern
Sajou_n°_002_01.pat.pattern
Sajou_n°_002_04.pat.pattern
Sajou_n°_002_05.pat.pattern
Sajou_n°_003_01.pat.pattern
Sajou_n°_003_03.pat.pattern
Sajou_n°_003_06.pat.pattern
Sajou_n°_005_02.pat.pattern
Stickmuster-Buck_010.pat.pattern
Stickmuster-Buck_02.pat.pattern
Stickmuster-Buck_07.pat.pattern
Vorlagen_zum_Wäschezeichnen_03.pat.pattern
Vorlagen_zum_Wäschezeichnen_07.pat.pattern
Vorlagen_zum_Wäschezeichnen_10.pat.pattern
Vorlagen_zum_Wäschezeichnen_12.pat.pattern
sajou656_pg_12.pat.pattern
sajou_186_05.pat.pattern
sajou_186_07.pat.pattern
sajou_206_1.pat.pattern
sajou_206_2.pat.pattern
sajou_206_3.pat.pattern
sajou_206_4.pat.pattern
sajou_325_1.pat.pattern
sajou_325_2.pat.pattern
sajou_325_3.pat.pattern
sajou_325_8.pat.pattern
sajou_362_1.pat.pattern
sajou_362_3.pat.pattern
sajou_602_1.pat.pattern
sajou_602_13.pat.pattern
sajou_602_14.pat.pattern
sajou_602_15.pat.pattern

this needs a database
(an excuse to use caqti)

this needs a web app
(an excuse to use dream^H vif)

```
let routes =
  let open Vif.Uri in
  let open Vif.Route in
  [ post (Vif.Type.multipart_form) (rel / "pattern" / "new" /?? nil) --> create
  ; post (Vif.Type.m search_form) (rel / "search" /?? nil) --> post_search
  ; get (rel / "pattern" / "new" /?? nil) --> upload
  ; get (rel / "pattern" /% int /?? any) --> show
  ; get (rel / "search" /?? nil) --> get_search
  ; get (rel / "edit" /?? nil) --> edit
  ; get (rel /?? nil) --> upload
  ]
```

```
small_font := "BmPlus_HP_100LX_6x8"
big_font := "BmPlus_IBM_VGA_9x16"

fonts :
        stitchcraft import font -i {{small_font}}.otb -o {{small_font}}.json
        stitchcraft import font -i {{big_font}}.otb -o {{big_font}}.json

injection :
        stitchcraft gen text -o injection.pattern -t 310 --font {{big_font}}.json -- \
        "border');" "DROP TABLE patterns; --"
border :
        stitchcraft gen text -t 535 --font {{small_font}}.json \
        "x\`;#'#\0" -o border_text.pattern
        stitchcraft import emborder --ct=Turn -o border.json border_text.pattern

fancy :
        just injection
        just border
        stitchcraft manip surround --border border.json --center injection.pattern -o fanc
y.pattern

pdf :
        just fancy
        stitchcraft export pdf -i fancy.pattern -o fancy.pdf -w "'); INSERT INTO bookmarks
 best VALUES http://stitch.website"

upload :
        curl -F tags="sql,injection,little bobby tables,omg hackers" \
        --form-string name="border'); DROP TABLE patterns; --" \
        -F "pattern=@fancy.pattern" http://localhost:8080/pattern/new
```

Stitchcraft Cloud Experience for Makers 2025 Platinum Edition (TM)(R) — Mozilla Firefox

yomimono, still on earth × Stitchcraft Cloud Experience ×

http://localhost:8080/search

Stitchcraft Cloud Experience for Makers 2025 Platinum Edition (TM)(R)

search upload

Tags (comma-separated, e.g. pies,cakes,cheese and crackers):

sql

Search

Stitchcraft Cloud Experience for Makers 2025 Platinum Edition (TM)(R) — Mozilla Firefox

yomimono, still on earth × | Stitchcraft Cloud Experience ×

http://localhost:8080/search

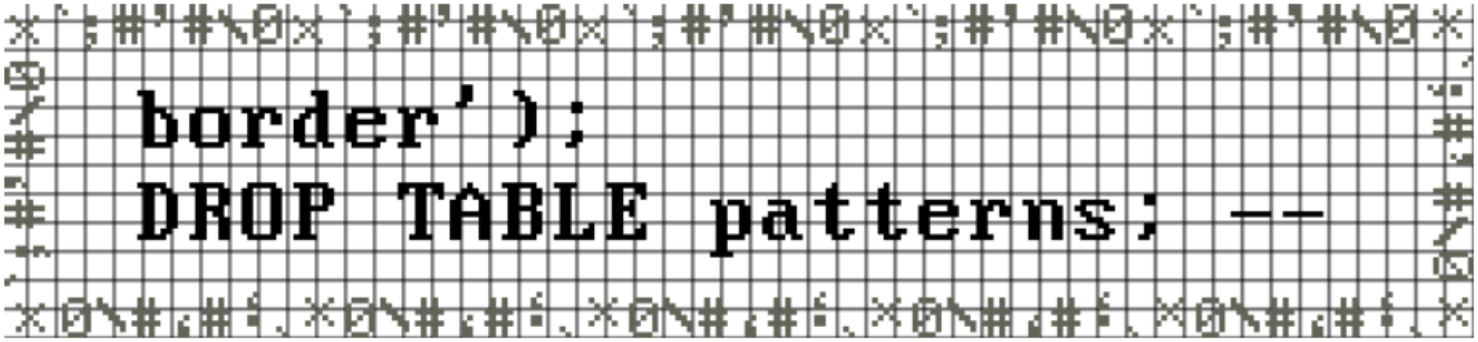
Stitchcraft Cloud Experience for Makers 2025 Platinum Edition (TM)(R)
search upload

Results for your search sql

1. border'); DROP TABLE patterns; --: matching 1 tag(s) (sql)

Stitchcraft Cloud Experience for Makers 2025 Platinum Edition (TM)(R)
search upload

border'); DROP TABLE patterns; --



Materials List

Summary

Estimated total: 2.43 USD, 324 minutes

Thread

- **DMC 310: Black**

  817 stitches (280.11 linear inches, 1 standard skein(s))

- **DMC 535: Ash Gray Vy Lt**

  1128 stitches (386.74 linear inches, 1 standard skein(s))

Fabric

a 21.00 in. x 7.00 in. piece of 14-count Aida cloth (including 1.00 in. margin on every side left blank for mounting)

this editor needs work

Stitchcraft Cloud Experience for Makers 2025 Platinum Edition (TM)(R) — Mozilla Firefox

yomimono, still on earth × | Stitchcraft Cloud Experience ×

http://localhost:8080/edit 120% Search

FUN
OCaml
2025!

eraser | display json | save to server

I think it's an excuse to use some software
but I don't know which software yet

please leave suggestions :)


ocamlc says:
"Thank you for playing
OCaml Programming.
Next time... be more careful!"
Restore
Restart
Quit

thank you!

http://github.com/yomimono/stitchcraft

I made some other stuff too

unicode explorer:
which code points can this terminal font render?

yomimono@osborne-2:~/unicode_explorer

0x2520

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2530

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2540

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2550

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2560

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2570

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2580

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x2590

0 1 2 3 4 5 6 7 8 9 a b c d e f

0x25a0

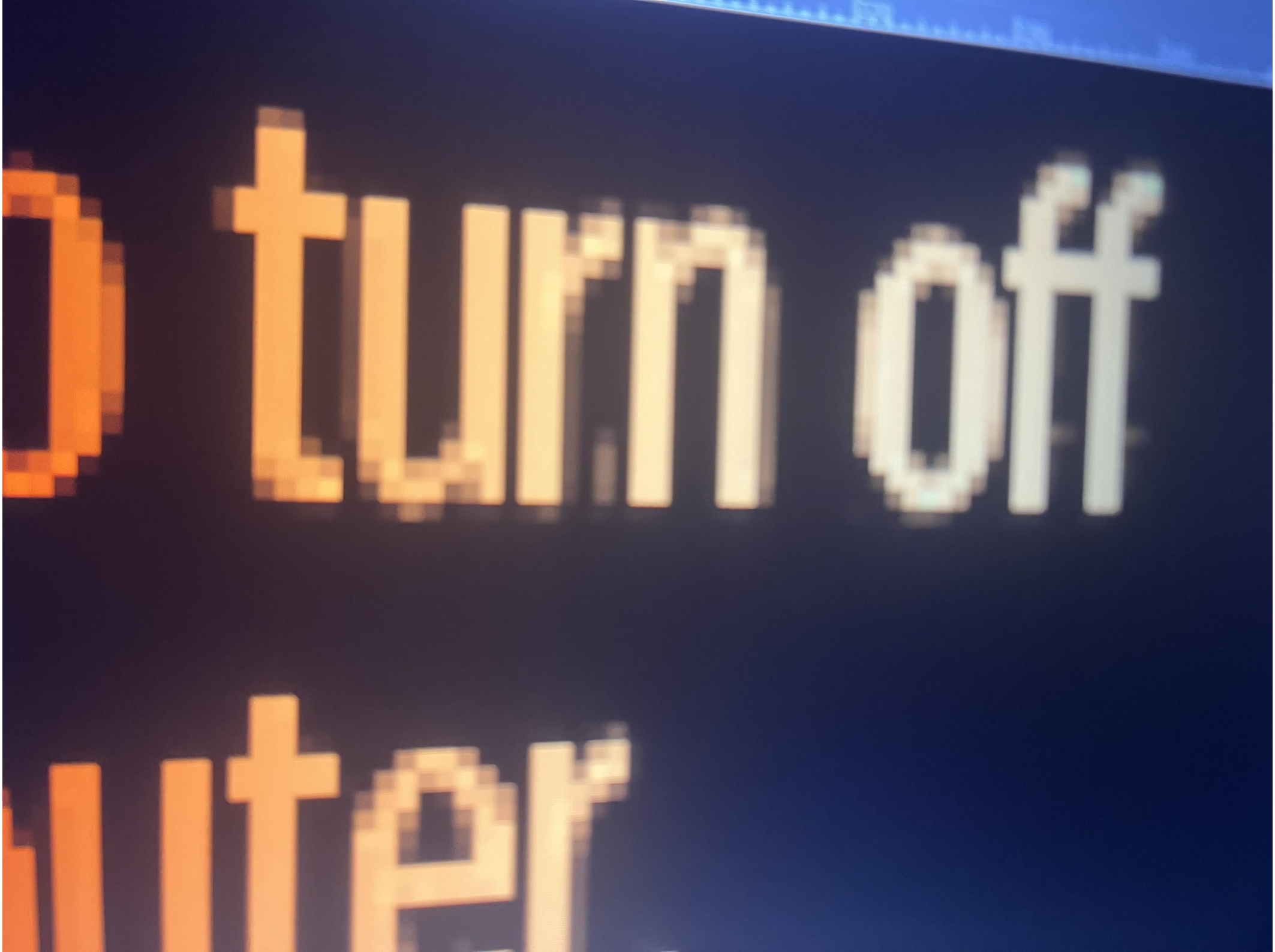
0 1 2 3 4 5 6 7 8 9 a b c d e f

colorseer:

put a color sensor in a box with floss
ask it what color the embroidery floss is
(this is "in OCaml" in the sense that
there's a desktop OCaml application talking to
an Arduino over a serial port)

and some other things for stitchcraft

like (after seven years)
image import with color matching


turn off


It's now safe to turn off
your computer.