kjhf

Claude Heiland-Allen

2010–2019

Contents

1 examples/helloworld.hf . . . 3
2 .gitignore . . . 3
3 prototype/bitstream.h . . . 3
4 prototype/fbitstream.c . . . 4
5 prototype/fbitstream.h . . . 5
6 prototype/fbitstream-test.c . . . 6
7 prototype/hyperspace.c . . . 6
8 prototype/hyperspace.h . . . 10
9 prototype/machine.c . . . 12
10 prototype/machine.h . . . 13
11 prototype/main.c . . . 13
12 prototype/Makefile . . . 14
13 prototype/NEWS . . . 14
14 prototype/parser.c . . . 16
15 prototype/parser.h . . . 18
16 prototype/README . . . 18
17 prototype/TODO . . . 20
18 src/10.txt.utf-8.patch.gz . . . 20
19 src/code.png . . . 21
20 src/DataTape.hs . . . 21
21 src/ding.pd . . . 23
22 src/dvd.xml . . . 24
23 src/font.png . . . 25
24 src/getdat.sh . . . 25
25 src/interleave31.c . . . 25
26 src/kjhf.cabal . . . 26
27 src/KJHF.hs . . . 27
28 src/kjhf.pd . . . 32
29 src/LICENSE.html . . . 34
30 src/Machine.hs . . . 40
31 src/Makefile . . . 42
32 src/monk.pd . . . 42
33 src/monks2.pd . . . 44
34 src/monks.pd . . . 44
35 src/png2rgba.sh . . . 45
36 src/README . . . 46
37 src/Setup.hs . . . 46
38 src/Snapshot.hs . . . 46
39 src/tile.png . . . 47
40 src/video.sh . . . 47
41 web/kjhf.svg . . . 47

1 examples/helloworld.hf

```
;;;+;+;;+;+;
+;+;+;+;;+;;+;
;;+;;+;+;;+;
;;+;;+;+;;+;
+;;;;+;+;;+;
;;+;;+;+;+;;
;;;;;+;+;;
+;;;+;+;;;+;
+;;;;+;+;;+;
;+;+;;+;;;+;
;;+;;+;+;;+;
;;+;+;;+;;+;
+;+;;;;+;+;;
;+;+;+;
```

2 .gitignore

```
*.hi
*.o
KJHF
```

3 prototype/bitstream.h

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#ifndef BITSTREAM_H
#define BITSTREAM_H 1

/***    struct ibitstream    ************************************************
**
**  an input stream of bits
**
**  USAGE
**    i = i->get(i);  // get() invalidates its argument
**
*/

struct ibitstream {
  /* public, readonly */
  int end;
  int bit;
  struct ibitstream *(*get)(struct ibitstream *);
  /* private, implementation defined */
  void *data;
};

/***    struct obitstream    ************************************************
**
**  an output stream of bits
```

```
**
**  USAGE
**    o = o->put(o, b);  // put() invalidates its argument
**
*/

struct obitstream {
  /* public, readonly */
  struct obitstream *(*put)(struct obitstream *, int);
  /* private, implementation defined */
  void *data;
};

#endif
```

4 prototype/fbitstream.c

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#include <stdlib.h>
#include "fbitstream.h"

struct fileData {
  int bit;
  int byte;
  FILE *file;
};

static struct ibitstream *fibitstreamGet(struct ibitstream *bs) {
  if (!bs->end) {
    struct fileData *data = bs->data;
    if (data->bit == 0) {
      data->byte = fgetc(data->file);
      if (data->byte == EOF) {
        bs->end = 1;
        return bs;
      } else {
        data->bit = 8;
      }
    }
    data->bit--;
    bs->bit = (data->byte & (1 << data->bit)) >> data->bit;
  }
  return bs;
}

struct ibitstream *fibitstream(FILE *f) {
  struct ibitstream *bs = malloc(sizeof(struct ibitstream));
  bs->bit = 0;
  bs->end = 0;
  bs->get = &fibitstreamGet;
  struct fileData *data = malloc(sizeof(struct fileData));
```

```
  data->bit = 0;
  data->byte = 0;
  data->file = f;
  bs->data = data;
  return bs;
}

static struct obitstream *fobitstreamPut(struct obitstream *bs, int b) {
  struct fileData *data = bs->data;
  data->bit--;
  data->byte |= (1 & b) << data->bit;
  if (data->bit == 0) {
    fputc(data->byte, data->file);
    fflush(data->file);
    data->bit = 8;
    data->byte = 0;
  }
  return bs;
}

struct obitstream *fobitstream(FILE *f) {
  struct obitstream *bs = malloc(sizeof(struct obitstream));
  bs->put = &fobitstreamPut;
  struct fileData *data = malloc(sizeof(struct fileData));
  data->bit = 8;
  data->byte = 0;
  data->file = f;
  bs->data = data;
  return bs;
}
```

5 prototype/fbitstream.h

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#ifndef FBITSTREAM_H
#define FBITSTREAM_H 1

#include <stdio.h>

#include "bitstream.h"

/***    struct ibitstream *fibitstream(FILE *)    ***************************
**
**  a stream of bits from a file (big-endian bytes)
**
*/

struct ibitstream *fibitstream(FILE *f);

/***    struct obitstream *fobitstream(FILE *)    ***************************
**
```

```
25  **  a stream of bits to a file (big-endian bytes)
    **
    */

    struct obitstream *fobitstream(FILE *f);
30
    #endif
```

6 prototype/fbitstream-test.c

```
    /*************************************************************************
    **
    **  hyperfuck -- boolfuck in the hyperbolic plane
    **  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
 5  **
    */

    #include <stdlib.h>
    #include "fbitstream.h"
10
    /* cat */
    int main(int argc, char **argv) {
      struct ibitstream *i = fibitstream(stdin);
      struct obitstream *o = fobitstream(stdout);
15    do {
        i = i->get(i);
        if (!i->end) {
          o = o->put(o, i->bit);
        }
20    } while (!i->end);
      return 0;
    }
```

7 prototype/hyperspace.c

```
    /*************************************************************************
    **
    **  hyperfuck -- boolfuck in the hyperbolic plane
    **  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
 5  **
    */

    #include <assert.h>
    #include <stdlib.h>
10  #include "hyperspace.h"

    /***    struct cell *cell(void)    ***************************************
    **
    **  create a new cell
15  **
    */

    struct cell *cell(void) {
      struct cell *c = malloc(sizeof(struct cell));
20    c->up = c->left = c->right = c->downLeft = c->downRight = 0;
      c->data = 0;
```

```
  return c;
}

/***    struct space *space(struct bitstream *)    ************************
**
**  create a new space
**
*/

struct space *space(struct ibitstream *bs) {
  struct space *s = malloc(sizeof(struct space));
  s->above = bs;
  s->top = cell();
  return s;
}

/***    struct cursor *cursor(struct space *)    ***************************
**
**  create a new cursor
**
*/

struct cursor *cursor(struct space *s) {
  struct cursor *c = malloc(sizeof(struct cursor));
  c->space = s;
  c->cell = s->top;
  return c;
}

/***    struct cursor *up(struct cursor *, int)    ************************
**
**  move up
**
**  TOP                    LEFT                    RIGHT
**    left    right          (exists)                (as LEFT)
**    +---+   +---+
**    | N |   | N |
**    +-+-+   +-+-+
**    |O|       |O|
**    +++       +++
**
*/

struct cursor *up(struct cursor *o, int wantRight) {
  if (!o->cell->up) {
    int r;
    if (o->space->above->end) {
      r = wantRight;
    } else {
      o->space->above = o->space->above->get(o->space->above);
      r = o->space->above->bit;
    }
    o->cell->up = cell();
    if (r) { o->cell->up->downRight = o->cell; }
    else   { o->cell->up->downLeft  = o->cell; }
    o->space->top = o->cell->up;
  }
```

```
  o->cell = o->cell->up;
  return o;
}

/***    struct cursor *downLeft(struct cursor *)    ************************
**
**  move down left
**
**  TOP                       LEFT                      RIGHT
**    +---+                     +---+---+                 (as LEFT)
**    | O |                     |???| O |
**    +-+-+                     +-+-+-+-+
**    |N|?|                       |?|N|?|
**    +++++                       +++++++
**
*/

struct cursor *downLeft(struct cursor *o) {
  if (!o->cell->downLeft) {
    o->cell->downLeft = cell();
    o->cell->downLeft->up = o->cell;
    if (o->cell->downRight) {
      o->cell->downRight->left = o->cell->downLeft;
      o->cell->downLeft->right = o->cell->downRight;
    }
    if (o->cell->left && o->cell->left->downRight) {
      o->cell->left->downRight->right = o->cell->downLeft;
      o->cell->downLeft->left = o->cell->left->downRight;
    }
  }
  o->cell = o->cell->downLeft;
  return o;
}

/***    struct cursor *downRight(struct cursor *)    ************************
**
**  move down right
**
**  TOP                       LEFT                      RIGHT
**    +---+                     +---+---+                 (as LEFT)
**    | O |                     | O |???|
**    +-+-+                     +-+-+-+-+
**    |?|N|                     |?|N|?|
**    +++++                     +++++++
**
*/

struct cursor *downRight(struct cursor *o) {
  if (!o->cell->downRight) {
    o->cell->downRight = cell();
    o->cell->downRight->up = o->cell;
    if (o->cell->downLeft) {
      o->cell->downLeft->right = o->cell->downRight;
      o->cell->downRight->left = o->cell->downLeft;
    }
    if (o->cell->right && o->cell->right->downLeft) {
      o->cell->right->downLeft->left = o->cell->downRight;
```

```
      o->cell->downRight->right = o->cell->right->downLeft;
    }
  }
  o->cell = o->cell->downRight;
  return o;
}

/***    struct cursor *left(struct cursor *)    ****************************
**
**  move left
**
**  TOP                        LEFT                       RIGHT
**      +-------+                +---+---+                  +---+---+
**      |   N   |                |???<###|                  |???|###|
**      +-/-+-^-+                +-+\+^+-+                  +-+-+/+^+
**    { | N | N | }              |?|N|O|                      |?|N|O|
**    { +-+\+^+-+ }              +++++++                      +++++++
**        |N|O|
**        +++++
**
*/

struct cursor *left(struct cursor *o) {
  if (!o->cell->left) {
    if        (o->cell->up && o->cell->up->downRight == o->cell) { /* RIGHT */
      o->cell = o->cell->up;
      o = downLeft(o);
      return o;
    } else if (o->cell->up && o->cell->up->downLeft  == o->cell) { /* LEFT */
      o->cell = o->cell->up;
      o = left(o);
      o = downRight(o);
      return o;
    } else if (!o->cell->up) {                                    /* TOP */
      struct cell *c;
      int n = 0;
      do {
        c = o->cell;
        o = up(o, 1);
        n++;
      } while (o->cell->downLeft == c);
      o = downLeft(o);
      while (n--) {
        o = downRight(o);
      }
      return o;
    } else {
      assert(0); /* broken */
    }
  }
  o->cell = o->cell->left;
  return o;
}

/***    struct cursor *right(struct cursor *)    ***************************
**
**  move right
```

```
**
**  TOP                          LEFT                         RIGHT
**       +-------+                  +---+---+                    +---+---+
**       |   N   |                  |###|???|                    |###>###|
**       +-^-+-\-+                  +^+\+-+-+                    +-+^+/+-+
**     { | N | N | }                |O|N|?|                        |O|N|?|
**     { +-+^+/+-+ }                +++++++                        +++++++
**         |O|N|
**         +++++
**
*/

struct cursor *right(struct cursor *o) {
  if (!o->cell->right) {
    if        (o->cell->up && o->cell->up->downRight == o->cell) { /* RIGHT */
      o->cell = o->cell->up;
      o = right(o);
      o = downLeft(o);
      return o;
    } else if (o->cell->up && o->cell->up->downLeft  == o->cell) { /* LEFT */
      o->cell = o->cell->up;
      o = downRight(o);
      return o;
    } else if (!o->cell->up) {                                    /* TOP */
      struct cell *c;
      int n = 0;
      do {
        c = o->cell;
        o = up(o, 0);
        n++;
      } while (o->cell->downRight == c);
      o = downRight(o);
      while (n--) {
        o = downLeft(o);
      }
      return o;
    } else {
      assert(0); /* broken */
    }
  }
  o->cell = o->cell->right;
  return o;
}
```

8 prototype/hyperspace.h

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#ifndef HYPERSPACE_H
#define HYPERSPACE_H 1

#include "bitstream.h"
```

```
/***    struct cell    ******************************************************
**
**  pentagonal hyperbolic cell data structure definition
**
**  TOP                        LEFT                        RIGHT
**    c->up == 0                 c->up != 0                  c->up != 0
**                               c->up->downLeft == c        c->up->downRight == c
**                                 +-------+                   +-------+
**        0                        |   ^   |                   |   ^   |
**      +---+                      +---+---+                   +---+---+
**    0 |###| 0                  ?<|###|>?                       ?<|###|>?
**      +-+-+                      +-+-+                         +-+-+
**       / \                        / \                           / \
**      ?   ?                      ?   ?                         ?   ?
**
**  INVARIANT
**    exactly one cell is TOP
**
*/

struct cell {
  struct cell *up, *left, *right, *downLeft, *downRight;
  void *data;
};

struct cell *cell(void);

/***    struct space    ****************************************************
**
**  a region of hyperbolic space defined by its top cell below a bitstream
**
**  INVARIANT
**    s->above is finite, OR,
**    every suffix of s->above contains a 0 and a 1
**
**  RATIONALE
**    an infinite stream of 0 will cause problems when moving left
**    an infinite stream of 1 will cause problems when moving right
**
*/

struct space {
  struct ibitstream *above;
  struct cell *top;
};

struct space *space(struct ibitstream *);

/***    struct cursor    ****************************************************
**
**  a point in a region of hyperbolic space
**  moving a cursor expands the region as required
**
*/

struct cursor {
```

```
  struct space *space;
  struct cell *cell;
};

struct cursor *cursor(struct space*);

struct cursor *up       (struct cursor *, int);
struct cursor *left     (struct cursor *);
struct cursor *right    (struct cursor *);
struct cursor *downLeft (struct cursor *);
struct cursor *downRight(struct cursor *);

#endif
```

9 prototype/machine.c

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#include <stdio.h>
#include <stdlib.h>
#include "machine.h"
#include "fbitstream.h"
#include "parser.h"

static void *zero = 0;
static void *one = &one;

struct machine *machine(char *c, FILE *g, FILE *i, FILE *o) {
  struct machine *m = malloc(sizeof(struct machine));
  m->input = fibitstream(i);
  m->output = fobitstream(o);
  m->data = space(fibitstream(g));
  m->dataPtr = cursor(m->data);
  m->code = parse(c);
  m->codePtr = m->code;
  m->running = 1;
  return m;
}

struct machine *step(struct machine *m) {
  int bit = m->dataPtr->cell->data != zero;
  enum opcode op = m->codePtr->op;
  /* fputc("+^<>/\\,;|#"[op], stderr); */
  switch (op) {
  case (oHalt):      m->running = 0; break;
  case (oFlip):      m->dataPtr->cell->data = !bit ? one : zero; break;
  case (oUp):        m->dataPtr = up       (m->dataPtr, 0); break;
  case (oLeft):      m->dataPtr = left     (m->dataPtr); break;
  case (oRight):     m->dataPtr = right    (m->dataPtr); break;
  case (oDownLeft):  m->dataPtr = downLeft (m->dataPtr); break;
  case (oDownRight): m->dataPtr = downRight(m->dataPtr); break;
  case (oInput):     m->input = m->input->get(m->input);
```

```
                        m->dataPtr->cell->data = m->input->bit ? one : zero;
                        break;
    case (oOutput):     m->output = m->output->put(m->output, bit);
                        if (m->input->end) { m->dataPtr->cell->data = zero; }
                        break;
    default:            break;
    }
   m->codePtr = m->codePtr->next[bit];
   return m;
}
```

10 prototype/machine.h

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#ifndef MACHINE_H
#define MACHINE_H 1

#include <stdio.h>
#include "hyperspace.h"

struct machine {
  struct ibitstream *input;
  struct obitstream *output;
  struct space *data;
  struct cursor *dataPtr;
  struct instruction *code;
  struct instruction *codePtr;
  int running;
};

struct machine *machine(char *, FILE *, FILE *, FILE *);
struct machine *step(struct machine *);

#endif
```

11 prototype/main.c

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#include <assert.h>
#include <stdio.h>
#include "machine.h"

int main(int argc, char **argv) {
  assert(argc > 1);
```

```
  char *source = argv[1];
  FILE *geometry = fopen("kjv10.txt", "rb");
  assert(geometry);
  struct machine *m = machine(source, geometry, stdin, stdout);
  while (m->running) { m = step(m); }
  return 0;
}
```

12 prototype/Makefile

```
CC = gcc
CFLAGS = -std=c99 -Wall -pedantic -O3 -ggdb
LIBS =
OBJS = fbitstream.o hyperspace.o machine.o main.o parser.o

all: hyperfuck

clean:
	-rm -f $(OBJS)

test: fbitstream-test

.SUFFIXES:
.PHONY: all clean test

# link program
hyperfuck: $(OBJS)
	$(CC) $(CFLAGS) -o hyperfuck $(OBJS) $(LIBS)

fbitstream-test: fbitstream-test.o fbitstream.o
	$(CC) $(CFLAGS) -s -o fbitstream-test fbitstream-test.o fbitstream.o

# dependencies
fbitstream.o: fbitstream.c fbitstream.h bitstream.h
fbitstream-test.o: fbitstream-test.c fbitstream.h bitstream.h
hyperspace.o: hyperspace.c hyperspace.h bitstream.h
machine.o: machine.c machine.h bitstream.h fbitstream.h hyperspace.h parser.h
parser.o: parser.c parser.h
main.o: main.c machine.h hyperspace.h bitstream.h

# C source to object file
%.o: %.c
	$(CC) $(CFLAGS) -o $@ -c $<
```

13 prototype/NEWS

```
(For the sake of this proposal, it is better to consider the
language non-profane.)

'Brainfuck' is an esoteric programming language based on a
minimal Turing machine: it has a data tape and a code array,
the instructions in the code modify the data in cells on the
tape. Despite the presence of only a few simple instructions
in the language it is theoretically possible to compute
everything that can be computed within the Church/Turing
model of computation; further an even more minimal
```

derivative of Brainfuck, 'Boolfuck', restricts the data on the tape to consist only of single bits (either 0 or 1).

Now, consider changing the structure of the data 'tape', in particular extending it into a 2D hyperbolic plane. Hyperbolic geometry allows many parallel lines to a given line through a given point not on the line, with the consequence that the hyperbolic plane has exponentially more space than the common flat Euclidean plane.

Now, it is possible to tile the hyperbolic plane with congruent shapes, in many more ways than flat space can be tiled. For example, with one particularly shaped pentagonal tile, one can tile the hyperbolic plane in uncountably infinitely many ways. To navigate in this new arrangement of cells in this hyperbolic pentagonal tape, one adds three new instructions to the language: in addition to moving the cursor in the data tape East and West, new instructions allow moving North, South-East and South-West.

This best visualised in the Poincaré half-plane model of hyperbolic geometry, in which shapes are distorted to give a horizon at infinity to the South (compare with the Poincaré disk model in which there is a circular horizon at infinity, most popularized by M C Escher's 'Circle Limit I-IV' prints).

This 2D hyperbolic modification of 'Boolfuck' I call 'Hyperfuck', for fairly obvious reasons.

Now, the pentagonal tiling described above has a kind of tree-like memory property: if you proceed from a starting location and head North, you might enter the pentagon above from either its South-East edge or its South-West edge: this allows information to be encoded in the geometry of the initial 'Garden Of Eden' configuration of the system. By modifying data in the cells and moving suitably, comparing the results of the movements with what you expect allows you to recover the stored bitstream. Further, this information need have no bound, for example one could encode an infinite library of every book ever written and that ever will be written (compare with Borges, the Shakespeare Typing Monkeys, and the text entry software 'Dasher').

However, a suitable source of information if a programmer is to be able to write useful programs more easily needs to be widely known and predictable: thus I propose the ASCII bitstream of a certain authorized version of the Christian Bible.

This liturgical modification of Hyperfuck I call 'King James Hyperfuck', for fairly obvious reasons.

Now that the theoretical background to the project is explained, I can proceed to the concrete proposal: to implement a graphical King James Hyperfuck interpreter that will run in a web browser, using Javascript and either SVG

```
or Canvas (to be decided).  There will be a facility to
choose from a library of simple example programs, and in
addition the visitor can write their own code to execute.
The interpreter will provide a visualisation of the
hyperbolic data tape and its mutation by the running code;
the code tape will also be displayed.  The output of the
program will be displayed as it is generated.

The first example program will reconstruct the text of the
King James Bible; its source code:

    +[^/+;^+]

The web installation will also present the visitor with the
theoretical background to the project, to enlighten and
inform.

Current state of the project: I have an unpublished
interpreter implemented in the C programming language
(though its algorithms need some minor tweaks to cope with
pathological bitstreams), and I have worked out the required
geometric shape of the pentagonal tiling when embedded in
the Poincaré half-plane model of hyperbolic geometry.  I
have prior experience of Javascript/SVG/Canvas programming
and modelling hyperbolic geometry, so I am confident I can
fully implement this proposal.

References:

http://en.wikipedia.org/wiki/Brainfuck
http://web.archive.org/web/20070403131306/http://www.rpi.edu/~hughes/boof/
http://en.wikipedia.org/wiki/Uniform_tilings_in_hyperbolic_plane
http://en.wikipedia.org/wiki/Poincar%C3%A9_half-plane_model
http://en.wikipedia.org/wiki/The_Library_of_Babel
http://en.wikipedia.org/wiki/Infinite_monkey_theorem
http://en.wikipedia.org/wiki/Dasher
http://www.gutenberg.org/etext/10
```

14 prototype/parser.c

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#include <assert.h>
#include <stdlib.h>
#include "parser.h"

struct instruction *parse(char *s) {
  /* pass 1: instruction allocation */
  struct instruction *pc = malloc(sizeof(struct instruction));
  struct instruction *pc0 = pc;
  struct instruction *pc1, *pc2;
  int depth;
```

```
  do {
    switch (*s) {
    case ('\0'): pc->op = oHalt; pc->next[0] = pc->next[1] = 0; break;
#define OP(C,X) case (C): pc->op = (X); pc = pc->next[0] = pc->next[1] = malloc(↙
    ↳ sizeof(struct instruction)); break
    OP('+', oFlip);
    OP('^', oUp);
    OP('<', oLeft);
    OP('>', oRight);
    OP('/', oDownLeft);
    OP('\\',oDownRight);
    OP(',', oInput);
    OP(';', oOutput);
    OP('[', oBra);
    OP(']', oKet);
#undef OP
    }
  } while (*s++);
  /* pass 2: bracket fixating, eg:  [+[+]+]
      ---------------------------------------
     /          ------------------           |
    /          /                  |          |
    skip,-flip->skip,->flip->skip<           |
        |           |------------>flip->skip<
        |-------------------------------------->END
  */
  pc = pc0;
  while (pc) {
    pc2 = pc->next[0];
    assert(pc->op != oKet);
    if (pc->op == oBra) {
      pc1 = pc->next[0];
      depth = 1;
      while (depth > 0) {
        if      (pc1->op == oBra) {
          depth++;
          pc1 = pc1->next[0];
        } else if (pc1->op == oKet) {
          depth--;
          if (depth > 0) {
            pc1 = pc1->next[0];
          }
        } else {
          pc1 = pc1->next[0];
        }
      }
      assert(depth == 0 && pc1 && pc1->op == oKet);
      pc1->next[1] = pc->next[0];
      pc->next[0] = pc1->next[0];
      pc->op = pc1->op = oSkip;
    }
    pc = pc2;
  }
  return pc0;
}
```

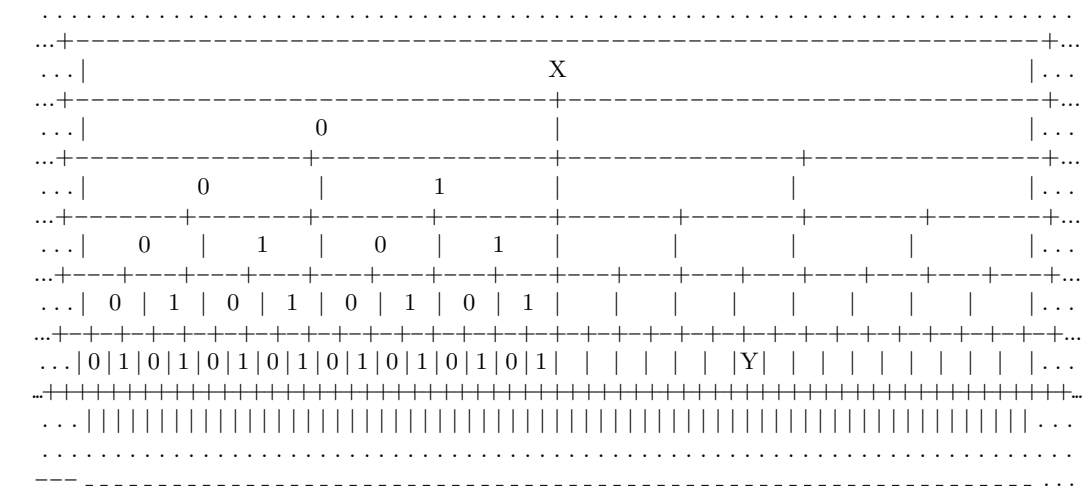
15 prototype/parser.h

```
/*****************************************************************************
**
**  hyperfuck -- boolfuck in the hyperbolic plane
**  Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>
**
*/

#ifndef PARSER_H
#define PARSER_H 1

enum opcode {
  oFlip,
  oUp, oLeft, oRight, oDownLeft, oDownRight,
  oInput, oOutput,
  oSkip,
  oHalt,
  oBra, oKet
};

struct instruction {
  enum opcode op;
  struct instruction *next[2];
};

struct instruction *parse(char *);

#endif
```

16 prototype/README

```
hyperfuck -- boolfuck in the hyperbolic plane
Copyright (C) 2010 Claude Heiland-Allen <claude@mathr.co.uk>


Weaker, less useful, more futile: Boolfuck[1].
-----------------------------------------------

Minimal Turing machine with an infinite 1D data tape containing bits (0 or 1)
and a code tape containing instructions:

    +   *p = !*p;                       toggle bit under the pointer
    <   p--;                            move pointer left  one bit
    >   p++;                            move pointer right one bit
    [   while(*p) {                     skip forward  when 0 (nesting)
    ]   }                               skip backward when 1 (nesting)
    ,   *p = getbit();                  read bit from input stream
    ;   putbit(*p); if(eof()) *p = 0;   output (and check end of input)

The machine halts at the end of the code tape.  Input and output are in
bytewise little-endian bit order.


Tiling the hyperbolic plane with identical pentagons.
------------------------------------------------------
```

The hyperbolic plane can be modelled in various ways, pick the Poincaré half-plane model for our purpose. Horizontal line at the bottom is the horizon at infinity; geodesics are vertical half-lines and semi-circles centered on the horizon. Vertical distortion in the ASCII diagram below: each row should be twice the height of the next row.

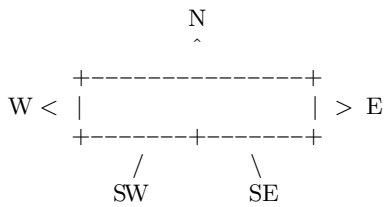


Left children are labelled 0, right children with 1.

Vertical paths between points can encode data: X-->Y == 1 0 1 1 0 == 22.

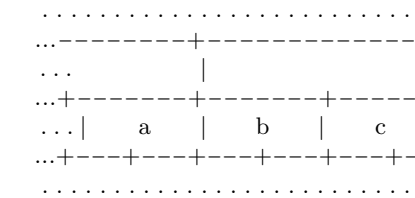
Hyperfuck is hyperbolic Boolfuck.

Extend the Boolfuck pointer manipulation instruction set from two to five commands, corresponding to the five edges of the hyperbolic pentagons::



Navigating with a five-pointed compass.

It's possible to find out whether your current position is a left child or a right child (assuming you can overwrite cell contents):



```
To find if you are at a right child, check if:

    ^/ == <

For example:

    c^/  ==  b  ==  c<          therefore c is a right child
    b^/  ==  b  !=  a  == b<    therefore b is a not a right child

More concrete code to perform the check:

    <[+]+    set west cell to 1
    >[+]     set this cell to 0
    ^/[+>    this is a right child
    ]

By symmetry, you can check you are at a left child when:

    ^\ == >


King James Hyperfuck
--------------------

As Hyperfuck, but with the initial position such that this program reveals
a certain text[2]:

    +[^/+;^+]


References
----------

[1] http://web.archive.org/web/20070403131306/http://www.rpi.edu/~hughes/boof/
[2] http://www.gutenberg.org/dirs/etext90/kjv10.txt
```

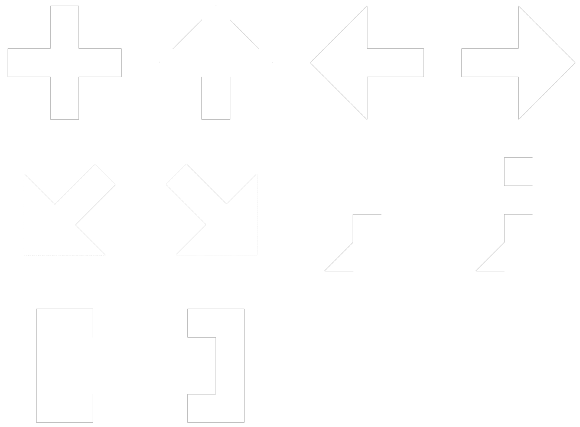
17 prototype/TODO

```
visual version (web browser js svg xmlhttprequest etc)
```

18 src/10.txt.utf-8.patch.gz

(application/gzip; charset=binary)

19 src/code.png



20 src/DataTape.hs

```
module DataTape(DataTape(), dataTape, get, modify, up, left, right, downLeft, ↙
    ↳ downRight, isRightChild, getPointer, pretty) where

import Data.List(intersperse)
import Data.Map(Map)
import qualified Data.Map as M

default (Int)

data Cell = Cell{ cUp, cLeft, cRight, cDownLeft, cDownRight :: Maybe Int, cData ↙
    ↳ :: Bool }
  deriving (Read, Show)

cell :: Cell
cell = Cell Nothing Nothing Nothing Nothing Nothing False

data DataTape = DataTape{ dtCells :: Map Int Cell, dtPtr :: Int, _dtPosition :: ↙
    ↳ [Bool] }

dataTape :: [Bool] -> DataTape
dataTape ps = DataTape (M.singleton 0 cell) 0 ps

get :: DataTape -> Bool
get (DataTape m p _) = cData (m M.! p)

modify :: (Bool -> Bool) -> DataTape -> DataTape
modify f dt@(DataTape m p _) = dt{ dtCells = M.update (\c -> Just c{ cData = f (↙
    ↳ cData c) }) p m }

merge :: Int -> Int -> DataTape -> DataTape
```

```
merge l r dt = dt{ dtCells = M.update (\c -> Just c{ cLeft  = Just l }) r
                           . M.update (\c -> Just c{ cRight = Just r }) l
                           $ dtCells dt }

isRightChild :: DataTape -> Bool
isRightChild dt = let dt' = downRight . up $ dt
                  in dtPtr dt == dtPtr dt'

getPointer :: DataTape -> Int
getPointer dt = dtPtr dt

up :: DataTape -> DataTape
up dt@(DataTape m p (b:bs)) = case cUp (m M.! p) of
  Just q  -> dt{ dtPtr = q }
  Nothing -> let q = M.size m
                 c = if b then cell{ cDownRight = Just p } else cell{ cDownLeft = Just p }
                 n = M.update (\c0 -> Just c0{ cUp = Just q }) p m
             in DataTape (M.insert q c n) q bs
up    (DataTape _ _ []    ) = error "DataTape.up: no more position data"

left :: DataTape -> DataTape
left dt@(DataTape m p _) = case cLeft (m M.! p) of
  Just q  -> dt{ dtPtr = q }
  Nothing -> let dt'@(DataTape m' u _) = up dt
             in if cDownRight (m' M.! u) == Just p then
                 downLeft $ dt'
                else
                if cDownLeft  (m' M.! u) == Just p then (\dt'' -> merge (dtPtr dt'') p dt'') . downRight . left $ dt' else
                error "DataTape.left: internal error: went up but no child"

right :: DataTape -> DataTape
right dt@(DataTape m p _) = case cRight (m M.! p) of
  Just q  -> dt{ dtPtr = q }
  Nothing -> let dt'@(DataTape m' u _) = up dt
             in if cDownLeft  (m' M.! u) == Just p then
                 downRight $ dt'
                else
                if cDownRight (m' M.! u) == Just p then (\dt'' -> merge p (dtPtr dt'') dt'') . downLeft . right $ dt' else
                error "DataTape.right: internal error: went up but no child"

downLeft :: DataTape -> DataTape
downLeft dt@(DataTape m p _) = case cDownLeft (m M.! p) of
  Just q  -> dt{ dtPtr = q }
  Nothing -> let q = M.size m
                 c = cell{ cUp = Just p }
                 n = M.update (\c0 -> Just c0{ cDownLeft = Just q }) p m
             in maybe id (\r -> merge q r) (cDownRight (m M.! p)) dt{ dtCells = M.insert q c n, dtPtr = q }

downRight :: DataTape -> DataTape
downRight dt@(DataTape m p _) = case cDownRight (m M.! p) of
  Just q  -> dt{ dtPtr = q }
  Nothing -> let q = M.size m
                 c = cell{ cUp = Just p }
```

```
                n = M.update (\c0 -> Just c0{ cDownRight = Just q }) p m
            in maybe id (\l -> merge l q) (cDownLeft (m M.! p)) dt{ dtCells = M↙
                ↳ .insert q c n, dtPtr = q }

pretty :: DataTape -> String
pretty dt = pretty' 3 (5 * 2^(8 - 3 :: Int) + 2) (dtPtr dt) (left . left . left ↙
    ↳ . left . left $ dt)

pretty' :: Int -> Int -> Int -> DataTape -> String
pretty' n dx p0 dt | n >  0 = let dt' = up dt
                                  isR = cDownRight (dtCells dt' M.! dtPtr dt') ↙
                                      ↳ == Just (dtPtr dt)
                                  dx' = dx + (if isR then id else negate) (2 ^ ↙
                                      ↳ (7 - n)) - 1
                              in pretty' (n - 1) dx' p0 dt'
                   | otherwise = let w = 2^(8 :: Int) + 1
                                 in unlines . (++ [replicate w '+']) . concat . ↙
                                     ↳ zipWith inflate [5,4 ..] . map (take w) .↙
                                     ↳  map (drop dx) . drop 1 . take 8 . pretty↙
                                     ↳ '' 7 p0 $ dt

inflate :: Int -> String -> [String]
inflate n l | n >= -1 = let h = map (\c -> if c == '|' then '+' else '-') l
                            i = map (\c -> if c == '|' then '|' else ' ') l
                            is = drop 1 (replicate (2^(n+1) `div` 4) i)
                        in [h] ++ is ++ [l] ++ is
            | otherwise = error "DataTape.inflate: n < -1"

pretty'' :: Int -> Int -> DataTape -> [String]
pretty'' n p0 dt = (concat . intersperse "|" . map (pretty''' n p0) . iterate ↙
    ↳ right $ dt) : pretty'' (n - 1) p0 (downLeft dt)

pretty''' :: Int -> Int -> DataTape -> String
pretty''' n p0 dt | dtPtr dt == p0 = replicate (2^n-2) ' ' ++ "[" ++ (if get dt ↙
    ↳ then "1" else "0") ++ "]" ++ replicate (2^n-2) ' '
                  | otherwise      = replicate (2^n-1) ' ' ++       (if get dt ↙
                      ↳ then "1" else "0")       ++ replicate (2^n-1) ' '
```

21 src/ding.pd

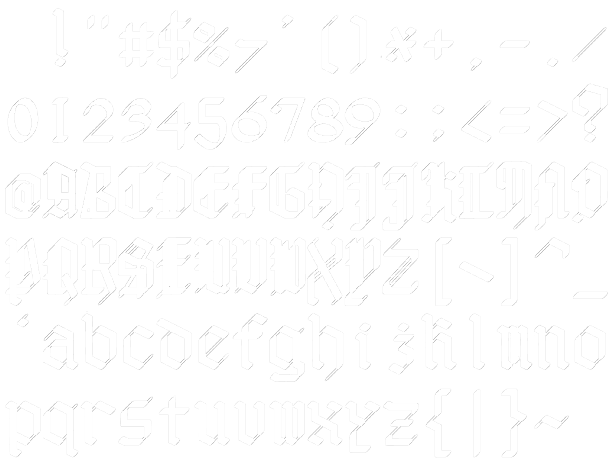
```
#N canvas 0 0 450 300 10;
#X obj 102 114 osc~;
#X obj 103 53 vline~;
#X obj 103 80 *~ 4;
#X obj 142 114 osc~;
#X obj 182 114 osc~;
#X obj 143 80 *~ 5;
#X obj 183 80 *~ 6;
#X obj 141 143 *~;
#X obj 102 143 *~;
#X obj 101 165 *~;
#X obj 101 193 outlet~;
#X obj 227 75 vline~;
#X obj 227 7 inlet;
#X msg 227 51 1 5 \, 0 1000 5;
#X obj 227 107 *~;
#X obj 227 127 *~;
```

```
#X obj 102 33 mtof;
#X connect 0 0 8 0;
#X connect 1 0 2 0;
#X connect 1 0 5 0;
#X connect 1 0 6 0;
#X connect 2 0 0 0;
#X connect 3 0 7 0;
#X connect 4 0 7 1;
#X connect 5 0 3 0;
#X connect 6 0 4 0;
#X connect 7 0 8 1;
#X connect 8 0 9 0;
#X connect 9 0 10 0;
#X connect 11 0 14 0;
#X connect 11 0 14 1;
#X connect 12 0 13 0;
#X connect 12 0 16 0;
#X connect 13 0 11 0;
#X connect 14 0 15 0;
#X connect 14 0 15 1;
#X connect 15 0 9 1;
#X connect 16 0 1 0;
```

22 src/dvd.xml

```
<dvdauthor jumppad="no">
  <vmgm />
  <titleset>
    <menus/>
    <titles>
      <pgc>
        <vob file="kjhf.mpeg" />
      </pgc>
    </titles>
  </titleset>
</dvdauthor>
```

23 src/font.png



24 src/getdat.sh

```
#!/bin/bash
touch kjhf.dat &&
mv -fv kjhf.dat kjhf.dat.bak.$$ &&
wget -N -c "http://www.gutenberg.org/ebooks/10.txt.utf-8" &&
cat 10.txt.utf-8.patch.gz | gunzip |
patch -p1 -o - 10.txt.utf-8 - |
sed "s|\r||g" > kjhf.dat
```

25 src/interleave31.c

```
#include <stdio.h>

int main(int argc, char **argv) {
  if (argc != 3) {
    return 1;
  }
  FILE *in3 = fopen(argv[1], "rb");
  if (in3) {
    FILE *in1 = fopen(argv[2], "rb");
    if (in1) {
      for (int i = 0; i < 1024*1024; ++i) {
        putchar(getc(in3));
        putchar(getc(in3));
        putchar(getc(in3));
        putchar(getc(in1));
      }
      fclose(in1);
    } else {
      return 1;
```

```
    }
    fclose(in3);
  } else {
    return 1;
  }
  return 0;
}
```

26 src/kjhf.cabal

```
name:                kjhf
version:             0
synopsis:            liturgical boolfuck in hyperbolic space
description:
  King James Hyperfuck
  .
  Copyleft: This is a free work, you can copy, distribute, and modify it under
  the terms of the Free Art License <http://artlibre.org/licence/lal/en/>

license:             OtherLicense
license-file:        LICENSE.html
homepage:            http://code.mathr.co.uk/kjhf
author:              Claude Heiland-Allen
maintainer:          claude@mathr.co.uk
copyright:           2010,2015 Claude Heiland-Allen
category:            Art
build-type:          Simple
extra-source-files:
  README
  10.txt.utf-8.patch.gz
  code.png
  font.png
  tile.png
  ding.pd
  kjhf.pd
  monk.pd
  monks.pd
  monks2.pd
  Makefile
  interleave31.c
  getdat.sh
  png2rgba.sh
  video.sh
  dvd.xml
cabal-version:       >=1.10

executable kjhf
  main-is:             KJHF.hs
  other-modules:       DataTape Machine Snapshot
  build-depends:       base >=4.8 && <4.9
                     , bytestring >=0.10 && <0.11
                     , containers >=0.5 && <0.6
                     , GLUT >=2.7 && <2.8
                     , hosc >=0.15 && <0.16
                     , network >=2.6 && <2.7
  default-language:    Haskell2010
```

27 src/KJHF.hs

```
module Main(main) where

import Graphics.UI.GLUT hiding (get, cursor)
import Sound.OSC.Time(time)

import Data.IORef
import Data.Char(ord, chr)
import Data.Bits(testBit, shiftR)
import Control.Monad(when)
import Network(connectTo, PortID(PortNumber))
import System.IO(stdout, openFile, withBinaryFile, IOMode(ReadMode, WriteMode), ↙
    ↳ hGetBuf, hSetBuffering, BufferMode(LineBuffering), hPutStrLn, Handle)
import System.Exit(exitFailure)
import Foreign.Marshal.Alloc(allocaBytes)

import Machine   hiding (pretty)
import DataTape  hiding (pretty)
import Snapshot

rt :: Bool
rt = True

data World = World
  { wMachine, wNextMachine :: Machine
  , wNow :: Double
  , wCorners :: (GLdouble, GLdouble, GLdouble, GLdouble)
  , wScale :: GLdouble
  , wTranslate :: GLdouble
  , wDelta :: GLdouble
  , wOutput :: [Bool]
  , wOutputText :: [Char]
  , fontTex :: TextureObject
  , tileTex :: TextureObject
  , codeTex :: TextureObject
  , wPd :: Handle
  , wFrames :: Int
  }

reshape :: IORef World -> Size -> IO ()
reshape worldRef vp@(Size w h) = do
  let s = 8 * 12 / 2
      cs@(x0,x1,y0,y1) = let v = s * fromIntegral w / fromIntegral h
                         in (-v,v,0,2*s)
  writeIORef worldRef . (\ww-> ww{ wCorners = cs }) =<< readIORef worldRef
  viewport $= (Position 0 0, vp)
  matrixMode $= Projection
  loadIdentity
  ortho x0 x1 y0 y1 (-1) 1
  matrixMode $= Modelview 0
  loadIdentity
  postRedisplay Nothing

clock :: Double
clock = 0.24
```

```
55  simulate :: Int -> IORef World -> IO ()
    simulate mspf worldRef = do
      when (not rt) $
        writeIORef worldRef . (\ww -> ww{ wFrames = wFrames ww + 1 }) =<< readIORef ↙
            ↳ worldRef
      world <- readIORef worldRef
60    tnow <- if rt then time else return (0.04 * fromIntegral (wFrames world) + ↙
          ↳ wNow world)
      let t  = (tnow - wNow world) / clock
      when (t >= 1) $ do
        case step (wNextMachine world) of
          (Just (mach, op), out) -> do
65          let ds = case op of
                       OpUp        -> 0.5
                       OpDownLeft  -> 2.0
                       OpDownRight -> 2.0
                       _           -> 1.0
70              dx = case op of
                       OpUp        -> if isRightChild (getData (wMachine world))
                                      then  6.0
                                      else -6.0
                       OpLeft      ->  12.0
75                     OpRight     -> -12.0
                       OpDownLeft  ->   3.0
                       OpDownRight ->  -3.0
                       _           ->   0.0
                os = case out of
80                     Just b  -> b : wOutput world
                       Nothing ->     wOutput world
                (c, os') = if length os >= 8
                           then (Just . littleChar . take 8 $ os, drop 8 os)
                           else (Nothing, os)
85          hPutStrLn (wPd world) ("op " ++ show (fromEnum op) ++ ";")
            case c of
              Just c' ->  writeIORef worldRef world
                { wMachine = wNextMachine world
                , wNextMachine = mach
90              , wNow = tnow
                , wScale = ds
                , wTranslate = (dx / 6)
                , wOutput = os'
                , wOutputText = c' : take 31 (wOutputText world)
95              , wFrames = 0
                }
              Nothing ->  writeIORef worldRef world
                { wMachine = wNextMachine world
                , wNextMachine = mach
100             , wNow = tnow
                , wScale = ds
                , wTranslate = (dx / 6)
                , wOutput = os'
                , wFrames = 0
105             }
          _ -> writeIORef worldRef world{ wScale = 1.0, wTranslate = 0.0 }
      world' <- readIORef worldRef
      writeIORef worldRef world'
        { wDelta = realToFrac ((tnow - wNow world') / clock) }
```

```
110     postRedisplay Nothing
        addTimerCallback mspf $ simulate mspf worldRef

    display :: IORef World -> IO ()
    display worldRef = do
115   w <- readIORef worldRef
        let dt = getData . wMachine $ w
            (x0,x1,y0,y1) = wCorners w
            ox0 = (x0   +   x1) / 2
            oy0 = (y0*3 + 1*y1) / 4
120         d = wDelta w
            ds = wScale w ** d
            dx = wTranslate w * (d ** (3/2))
            ox = ds * ox0 + dx * oy
            oy = ds * oy0
125         gw = 10
            gh = 20
            textCoords = drop 1 $ [x1, x1 - gw ..] `zip` repeat y1
        -- setup
        clear [ColorBuffer]
130     blend $= Enabled
        blendFunc $= (SrcAlpha, OneMinusSrcAlpha)
        texture Texture2D $= Enabled
        -- draw tiles
        textureBinding Texture2D $= Just (tileTex w)
135     renderPrimitive Quads $
          mapM_ drawTile . concat . take 10 .
          map (takeWhile (not . farRight x1) . dropWhile (farLeft x0)) .
          dropWhile (farAbove y1 . head) . toTileRows (getPointer dt) .
          toTopLeft (x0, y1) (ox, oy) $ dt
140     -- draw text
        textureBinding Texture2D $= Just (fontTex w)
        renderPrimitive Quads $
          mapM_ (drawGlyph gw gh) . zip textCoords . take 32 $ wOutputText w
        -- draw code
145     let cw = 20
            ch = 20
            cy = y1 - gh
            codeLeftCoords  = drop 2 $ [cw/2, cw/2 - cw ..] `zip` repeat cy
            codeRightCoords =          [cw/2 - cw, cw/2 ..] `zip` repeat cy
150         (codeLeft, codeRight) = getCode (wMachine w)
        textureBinding Texture2D $= Just (codeTex w)
        renderPrimitive Quads $ do
          mapM_ (drawCode cw ch) . zip codeLeftCoords  $ codeLeft
          mapM_ (drawCode cw ch) . zip codeRightCoords $ codeRight
155     -- cleanup
        textureBinding Texture2D $= Nothing
        texture Texture2D $= Disabled
        swapBuffers
        when (not rt) $ hSnapshot stdout (Position 0 0) (Size 880 480)
160
    drawGlyph :: GLdouble -> GLdouble -> ((GLdouble,GLdouble), Char) -> IO ()
    drawGlyph w h ((x,y), c) = do
        let n = ord c
            tx0 :: GLdouble
165         tx0 = fromIntegral (n `mod` 16    ) *  64 / 1024
            ty0 = fromIntegral (n `div` 16    ) * 128 / 1024
```

```
        tx1 = fromIntegral (n `mod` 16 + 1) *  64 / 1024
        ty1 = fromIntegral (n `div` 16 + 1) * 128 / 1024
    color $ Color3 1 1 (1::GLdouble)
    texCoord $ TexCoord2 tx0 ty0
    vertex $ Vertex2 x y
    texCoord $ TexCoord2 tx1 ty0
    vertex $ Vertex2 (x+w) y
    texCoord $ TexCoord2 tx1 ty1
    vertex $ Vertex2 (x+w) (y-h)
    texCoord $ TexCoord2 tx0 ty1
    vertex $ Vertex2 x (y-h)

drawCode :: GLdouble -> GLdouble -> ((GLdouble,GLdouble), Opcode) -> IO ()
drawCode w h ((x,y), c) = do
  let n = fromEnum c
      tx0 :: GLdouble
      tx0 = fromIntegral (n `mod` 4     ) * 256 / 1024
      ty0 = fromIntegral (n `div` 4     ) * 256 / 1024
      tx1 = fromIntegral (n `mod` 4 + 1) * 256 / 1024
      ty1 = fromIntegral (n `div` 4 + 1) * 256 / 1024
  color $ Color3 1 1 (1::GLdouble)
  texCoord $ TexCoord2 tx0 ty0
  vertex $ Vertex2 x y
  texCoord $ TexCoord2 tx1 ty0
  vertex $ Vertex2 (x+w) y
  texCoord $ TexCoord2 tx1 ty1
  vertex $ Vertex2 (x+w) (y-h)
  texCoord $ TexCoord2 tx0 ty1
  vertex $ Vertex2 x (y-h)

data Tile = Tile{ tcx, tcy :: GLdouble, tdata :: Bool, tcursor :: Bool }

drawTile :: Tile -> IO ()
drawTile t = do
  let cx = tcx t
      cy = tcy t
      y0 = cy *  4 / 6
      y1 = cy * 10 / 6
      x0t = cx - cy
      x1t = cx + cy
      x0b = cx - cy
      x1b = cx + cy
      c = case (tdata t, tcursor t) of
        (False, False) -> Color3 (176/255) (128/255) ( 80/255)
        (False, True ) -> Color3 (255/255) (186/255) (116/255)
        (True,  False) -> Color3 (168/255) ( 40/255) ( 16/255)
        (True,  True ) -> Color3 (255/255) ( 64/255) ( 24/255 :: GLdouble)
  color c
  texCoord $ TexCoord2 0 (1::GLdouble)
  vertex $ Vertex2 x0b y0
  texCoord $ TexCoord2 0 (0::GLdouble)
  vertex $ Vertex2 x0t y1
  texCoord $ TexCoord2 1 (0::GLdouble)
  vertex $ Vertex2 x1t y1
  texCoord $ TexCoord2 1 (1::GLdouble)
  vertex $ Vertex2 x1b y0
```

```
toTileRows :: Int -> (DataTape, (GLdouble, GLdouble)) -> [[Tile]]
toTileRows cursor dtcs = map (toTileRow cursor) . iterate downLeft' $ dtcs
  where downLeft' (dt, (cx, cy)) = (downLeft dt, (cx - cy / 2, cy / 2))

toTileRow :: Int -> (DataTape, (GLdouble, GLdouble)) -> [Tile]
toTileRow cursor dtcs = map (toTile cursor) . iterate right' $ dtcs
  where right' (dt, (cx, cy)) = (right dt, (cx + cy * 2, cy))

toTile :: Int -> (DataTape, (GLdouble, GLdouble)) -> Tile
toTile cursor (dt, (cx, cy)) = Tile
  { tcx = cx, tcy = cy, tdata = get dt, tcursor = cursor == getPointer dt }

toTopLeft :: (GLdouble, GLdouble) -> (GLdouble, GLdouble) -> DataTape
          -> (DataTape, (GLdouble, GLdouble))
toTopLeft xy@(x, y) cxy@(cx, cy) dt
  | cy * 2 / 3 < y = toTopLeft xy (if isRightChild dt
                                   then cx - cy
                                   else cx + cy, cy * 2) (up dt)
  | cx + cy    > x = toTopLeft xy (cx - 2 * cy, cy) (left dt)
  | otherwise      = (dt, cxy)

farAbove, farRight, farLeft :: GLdouble -> Tile -> Bool
farAbove y t = y < tcy t * 2 / 3
farRight x t = x < tcx t - tcy t
farLeft  x t = x > tcx t + tcy t


main :: IO ()
main = do
  let w = 880
      h = 480
  initialWindowSize $= Size w h
  initialDisplayMode $= [RGBAMode, DoubleBuffered]
  (_, code:_) <- getArgsAndInitialize
  _ <- createWindow "KJHF"
  tnow <- time
  dpos <- readBits "kjhf.dat"
  inpu <- readBits "kjhf.in"
  fontT <- loadTexture "font.rgba"
  tileT <- loadTexture "tile.rgba"
  codeT <- loadTexture "code.rgba"
  pd <- if rt then connectTo "localhost" (PortNumber 6666) else openFile "kjhf.↙
    ↳ score" WriteMode
  hSetBuffering pd LineBuffering
  let Just mach = machine code
                  (dpos ++ strBits (cycle " Forever and ever, amen.")) inpu
      world = World
              { wMachine = mach
              , wNextMachine = mach
              , wNow = tnow
              , wDelta = 0
              , wScale = 1
              , wTranslate = 0
              , wOutput = []
              , wOutputText = []
              , wCorners = (0, fromIntegral w, 0, fromIntegral h)
              , fontTex = fontT
```

```
                  , tileTex = tileT
                  , codeTex = codeT
                  , wPd = pd
                  , wFrames = 0
                  }
  worldRef <- newIORef world
  displayCallback $= display worldRef
  reshapeCallback $= Just (reshape worldRef)
  let mspf = floor (1000 / 25 :: Double)
  addTimerCallback mspf $ simulate mspf worldRef
  mainLoop

loadTexture :: FilePath -> IO TextureObject
loadTexture f = do
  withBinaryFile f ReadMode $ \h -> do
    let count = 1024 * 1024 * 4
    allocaBytes count $ \pixels -> do
      count' <- hGetBuf h pixels count
      when (count' /= count) exitFailure
      [tex] <- genObjectNames 1
      texture Texture2D $= Enabled
      textureBinding Texture2D $= Just tex
      build2DMipmaps Texture2D RGBA' 1024 1024
        (PixelData RGBA UnsignedByte pixels)
      textureBinding Texture2D $= Nothing
      texture Texture2D $= Disabled
      return tex

strBits :: String -> [Bool]
strBits = concatMap (take 8 . littleBits . ord)

readBits :: FilePath -> IO [Bool]
readBits f = return . strBits =<< readFile f

littleBits :: Int -> [Bool]
littleBits = map (`testBit` 0) . iterate (`shiftR` 1)

littleChar :: [Bool] -> Char
littleChar = chr . foldl (\i b -> i * 2 + if b then 1 else 0) 0
```

28 src/kjhf.pd

```
#N canvas 45 32 371 425 10;
#X obj 103 158 loadbang;
#X msg 103 179 0;
#X msg 141 132 -1;
#X msg 83 134 1;
#X obj 37 40 netreceive 6666;
#X obj 37 70 route op;
#X obj 244 204 ding;
#X obj 284 204 ding;
#X msg 244 180 60;
#X msg 284 180 72;
#X obj 37 100 select 0 1 2 3 4 5 6 7 8 9;
#X obj 29 201 ding;
#X msg 29 177 67;
#X obj 202 205 ding;
```

```
#X msg 202 181 55;
#X obj 172 205 ding;
#X msg 172 181 48;
#X obj 134 351 dac~;
#X obj 99 307 expr~ $v1 * 0.25 \; $v2 * 0.25;
#X msg 52 131 1.5;
#X msg 112 134 -1.5;
#X obj 183 43 textfile;
#X msg 183 16 read kjhf.score \, rewind;
#X obj 78 15 metro 240;
#X obj 46 12 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 1 1
;
#X obj 94 209 monks2;
#X obj 114 396 writesf~ 2;
#X msg 16 371 open -bytes 4 kjhf.32.wav \, start;
#X msg 231 378 stop;
#X obj 21 14 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X connect 0 0 1 0;
#X connect 1 0 25 0;
#X connect 1 0 25 1;
#X connect 2 0 25 1;
#X connect 3 0 25 1;
#X connect 4 0 5 0;
#X connect 5 0 10 0;
#X connect 6 0 18 0;
#X connect 6 0 18 1;
#X connect 7 0 18 0;
#X connect 7 0 18 1;
#X connect 8 0 6 0;
#X connect 9 0 7 0;
#X connect 10 0 12 0;
#X connect 10 1 19 0;
#X connect 10 2 3 0;
#X connect 10 3 2 0;
#X connect 10 4 20 0;
#X connect 10 4 3 0;
#X connect 10 5 20 0;
#X connect 10 5 2 0;
#X connect 10 6 16 0;
#X connect 10 7 14 0;
#X connect 10 8 8 0;
#X connect 10 9 9 0;
#X connect 11 0 18 0;
#X connect 11 0 18 1;
#X connect 12 0 11 0;
#X connect 13 0 18 0;
#X connect 13 0 18 1;
#X connect 14 0 13 0;
#X connect 15 0 18 0;
#X connect 15 0 18 1;
#X connect 16 0 15 0;
#X connect 18 0 17 0;
#X connect 18 0 26 0;
#X connect 18 1 17 1;
#X connect 18 1 26 1;
#X connect 19 0 25 0;
```

```
#X connect 20 0 25 0;
#X connect 21 0 5 0;
#X connect 21 1 28 0;
#X connect 22 0 21 0;
#X connect 23 0 21 0;
#X connect 24 0 23 0;
#X connect 25 0 18 0;
#X connect 25 1 18 1;
#X connect 27 0 26 0;
#X connect 28 0 26 0;
#X connect 29 0 24 0;
#X connect 29 0 27 0;
```

29 src/LICENSE.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head profile="http://gmpg.org/xfn/11">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Artlibre.org: Free Art License 1.3</title>

<meta name="generator" content="WordPress 2.8.1" /> <!-- leave this for stats -->
<link rel="icon" type="image/png" href="http://artlibre.org/favicon.png"/>
<link rel="stylesheet" href="http://artlibre.org/wp-content/themes/artlibre/style.php" type="text/css" media="screen" />
<link href="http://artlibre.org/wp-content/themes/artlibre/print.css" rel="stylesheet" type="text/css" media="print"/>
<link rel="alternate" type="application/rss+xml" title="RSS 2.0" href="http://artlibre.org/feed" />
<link rel="alternate" type="text/xml" title="RSS .92" href="http://artlibre.org/feed/rss" />
<link rel="alternate" type="application/atom+xml" title="Atom 0.3" href="http://artlibre.org/feed/atom" />
<link rel="pingback" href="http://artlibre.org/xmlrpc.php" />
<link rel="alternate" type="application/rss+xml" title="Copyleft Attitude &raquo; Free Art License 1.3 Flux des commentaires" href="http://artlibre.org/licence/lal/en/feed" />
<link rel="EditURI" type="application/rsd+xml" title="RSD" href="http://artlibre.org/xmlrpc.php?rsd" />
<link rel="wlwmanifest" type="application/wlwmanifest+xml" href="http://artlibre.org/wp-includes/wlwmanifest.xml" />
<link rel='index' title='Copyleft Attitude' href='http://artlibre.org' />
<link rel='up' title='Licence Art Libre' href='http://artlibre.org/licence/lal' />
<meta name="generator" content="WordPress 2.8.1" />
</head>
<body>
<div class="container">
<div class="header_left"><a href="http://artlibre.org"><img alt="Logo" width="50" height="50" src="http://artlibre.org/logo.php?1"/></a></div>
<div class="header_center"><a href="http://artlibre.org">copyleft_attitude</a></div>
```

```
</div>

<div class="container">
        <div id="sidebar">
<menu>
<!-========== MENU GENERE AUTOMATIQUEMENT ==========->

<li><a href="http://artlibre.org/licence" title=""></a> <ul> <li><a href="http:/
    ↳ ://artlibre.org/licence/faq" title="FAQ">FAQ</a></li> <li><a href="http://
    ↳ artlibre.org/licence/lal" title="Licence Art Libre">Licence Art Libre</a>
    ↳ <ul> <li><a href="http://artlibre.org/licence/lal/en" title="Free Art
    ↳ License 1.3">Free Art License 1.3</a></li> <li><a href="http://artlibre.
    ↳ org/licence/lal/de" title="Lizenz Freie Kunst">Lizenz Freie Kunst</a></li>
    ↳  <li><a href="http://artlibre.org/licence/lal/es" title="Licencia Arte
    ↳ Libre">Licencia Arte Libre</a></li> <li><a href="http://artlibre.org/
    ↳ licence/lal/pt" title="Licen&ccedil;a da Arte Livre 1.3">Licen&ccedil;a da
    ↳  Arte Livre 1.3</a></li> <li><a href="http://artlibre.org/licence/lal/it"
    ↳ title="Licenza Arte Libera">Licenza Arte Libera</a></li> <li><a href="http
    ↳ ://artlibre.org/licence/lal/pl" title="Licencja Wolnej Sztuki 1.3">
    ↳ Licencja Wolnej Sztuki 1.3</a></li> <li><a href="http://artlibre.org/
    ↳ licence/lal/licence-art-libre-12" title="Licence Art Libre 1.2">Licence
    ↳ Art Libre 1.2</a></li> <li><a href="http://artlibre.org/licence/lal/
    ↳ mirrors" title="Les sites miroirs">Les sites miroirs</a></li> </ul> </li>
    ↳ </ul> </li> <li><a href="http://artlibre.org/copyleft" title="Copyleft
    ↳ Attitude">Copyleft Attitude</a> <ul> <li><a href="http://artlibre.org/
    ↳ copyleft/listes" title="Liste de diffusion">Liste de diffusion</a></li> <
    ↳ li><a href="http://artlibre.org/copyleft/events" title="Chronique des &
    ↳ eacute;v&eacute;nements">Chronique des &eacute;v&eacute;nements</a></li> <
    ↳ li><a href="http://artlibre.org/copyleft/echos" title="Echos critiques">
    ↳ Echos critiques</a></li> <li><a href="http://artlibre.org/copyleft/logos"
    ↳ title="Logos copyleft">Logos copyleft</a> <ul> <li><a href="http://
    ↳ artlibre.org/copyleft/logos/expo" title="Expo des LogoLefts">Expo des
    ↳ LogoLefts</a></li> </ul> </li> </ul> </li> <li><a href="http://artlibre.
    ↳ org/ressources" title="Les ressources">Les ressources</a> <ul> <li><a href
    ↳ ="http://artlibre.org/ressources/culture" title="Sur l&#8217;art, la
    ↳ culture, le partage et l&#8217;auteur">Sur l&#8217;art, la culture, le
    ↳ partage et l&#8217;auteur</a></li> <li><a href="http://artlibre.org/
    ↳ ressources/wikimedia-commons-art-libre" title="Wikimedia Commons Art Libre
    ↳ ">Wikimedia Commons Art Libre</a></li> <li><a href="http://artlibre.org/
    ↳ ressources/libres" title="Sur le libre et le logiciel libre">Sur le libre
    ↳ et le logiciel libre</a></li> <li><a href="http://artlibre.org/ressources/
    ↳ juridiques" title="Sur les questions juridiques">Sur les questions
    ↳ juridiques</a></li> <li><a href="http://artlibre.org/ressources/textes"
    ↳ title="Textes">Textes</a></li> </ul> </li>
<!-========== MENU DES OEUVRES ==========->

<li><a href="http://oeuvres.artlibre.org/">Les oeuvres</a>
<ul>
<li><a href="http://artlibre.org/archives/oeuvres/263">Cr&eacute;er une oeuvre
    ↳ libre</a></li>
<li><a href="http://oeuvres.artlibre.org/new.html">R&eacute;f&eacute;rencer son
    ↳ oeuvre</a></li>
<li><a href="http://users.artlibre.org/">Annuaire des auteurs</a></li>
<li><a href="http://oeuvres.artlibre.org/recherche/">Rechercher une oeuvre</a></
    ↳ li>
</ul></li>
<li><a href="http://artlibre.org/news">Actualit&eacute;s</a>
```

```
    </li>

    <!--==========MENU DES AUTEURS==========-->
50
    <li><a href="http://users.artlibre.org/">Auteurs</a>
        <ul>
            <li><a href='http://users.artlibre.org/identification.html'>S'identifier ↙
                ↳ </a></li>
    </ul>
55  </li>
    <li><a href="http://artlibre.org/about">A propos</a><ul>
    <li><a href="http://artlibre.org/about/rss-2">Fils Rss</a></li>
    </ul></li></menu>
    <br/>
60  <form method="get" id="searchform" action="http://artlibre.org">
    <div><input type="text" value="" name="s" id="s" />
    <input type="submit" id="searchsubmit" value="Rechercher" />
    </div>
    </form></div>
65  <div id="content">
    <div class="hbar"><i>   <a href="http://artlibre.org">Accueil</a> > <a href↙
        ↳ ='http://artlibre.org/licence/lal'>Licence Art Libre</a>  > Free Art ↙
        ↳ License 1.3  </i><div class='printdiv'><form action='' method='get'><↙
        ↳ input class='printable' type='submit' name='print' value='Imprimer'/></↙
        ↳ form></div></div><h2>Free Art License 1.3</h2>
    <div class="entrytext">
    <p><span id="more-"></span><br />
    [ Copyleft Attitude ]</p>
70  <p>Free Art License 1.3 (FAL 1.3) </p>
    <p><strong>Preamble</strong> </p>
    <p>The Free Art License grants the right to freely copy, distribute, and ↙
        ↳ transform creative works without infringing the author's rights. </p>
    <p>The Free Art License recognizes and protects these rights. Their ↙
        ↳ implementation has been reformulated in order to allow everyone to use ↙
        ↳ creations of the human mind in a creative manner, regardless of their ↙
        ↳ types and ways of expression. </p>
    <p>While the public's access to creations of the human mind usually is ↙
        ↳ restricted by the implementation of copyright law, it is favoured by the ↙
        ↳ Free Art License. This license intends  to allow the use of a work's ↙
        ↳ resources; to establish new conditions for creating in order to increase ↙
        ↳ creation opportunities. The Free Art License grants the right to use a ↙
        ↳ work, and acknowledges the right holder's and the user's rights and ↙
        ↳ responsibility.</p>
75  <p>The invention and development of digital technologies, Internet and Free ↙
        ↳ Software have changed creation methods: creations of the human mind can ↙
        ↳ obviously be distributed, exchanged, and transformed. They allow to ↙
        ↳ produce common works to which everyone can contribute to the benefit of ↙
        ↳ all. </p>
    <p>The main rationale for this Free Art License is to promote and protect these ↙
        ↳ creations of the human mind according to the principles of copyleft: ↙
        ↳ freedom to use, copy, distribute, transform, and prohibition of exclusive ↙
        ↳ appropriation. </p>
    <p><strong>Definitions</strong> </p>
    <p><em>"work"</em> either means the initial work, the subsequent works or the ↙
        ↳ common work as defined hereafter: </p>
    <p><em>"common work"</em> means a work composed of the initial work and all ↙
        ↳ subsequent contributions to it (originals and copies). The initial author ↙
```

```
    ↳ is the one who, by choosing this license, defines the conditions under ↙
    ↳ which contributions are made. </p>
<p><em>"Initial work" </em>means the work created by the initiator of the common↙
    ↳  work (as defined above), the copies of which can be modified by whoever ↙
    ↳ wants to  </p>
<p><em>"Subsequent works"</em> means the contributions made by authors who ↙
    ↳ participate in the evolution of the common work by exercising the rights ↙
    ↳ to reproduce, distribute, and modify that are granted by the license. </p>
<p><em>"Originals" </em>(sources or resources of the work) means all copies of ↙
    ↳ either the initial work or any subsequent work mentioning a date and used ↙
    ↳ by their author(s) as references for any subsequent updates, ↙
    ↳ interpretations, copies or reproductions. </p>
<p><em>"Copy"</em> means any reproduction of an original as defined by this ↙
    ↳ license. </p>
<p><strong>1. OBJECT </strong><br />
The aim of this license is to define the conditions under which one can use this↙
    ↳  work freely. </p>
<p><strong>2. SCOPE </strong><br />
This work is subject to copyright law. Through this license its author specifies↙
    ↳  the extent to which you can copy, distribute, and modify it. </p>
<p><strong>2.1 FREEDOM TO COPY (OR TO MAKE REPRODUCTIONS) </strong><br />
You have the right to copy this work for yourself, your friends or any other ↙
    ↳ person, whatever the technique used. </p>
<p><strong>2.2 FREEDOM TO DISTRIBUTE, TO PERFORM IN PUBLIC </strong><br />
You have the right to distribute copies of this work; whether modified or not, ↙
    ↳ whatever the medium and the place, with or without any charge, provided ↙
    ↳ that you:<br />
attach this license without any modification to the copies of this work or ↙
    ↳ indicate precisely where the license can be found,<br />
specify to the recipient the names of the author(s) of the originals, including ↙
    ↳ yours if you have modified the work,<br />
specify to the recipient where to access the originals (either initial or ↙
    ↳ subsequent).<br />
The authors of the originals may, if they wish to, give you the right to ↙
    ↳ distribute the originals under the same conditions as the copies. </p>
<p><strong>2.3 FREEDOM TO MODIFY </strong><br />
You have the right to modify copies of the originals (whether initial or ↙
    ↳ subsequent) provided you comply with the following conditions:<br />
all conditions in article 2.2 above, if you distribute modified copies;<br />
indicate that the work has been modified and, if it is possible, what kind of ↙
    ↳ modifications have been made;<br />
distribute the subsequent work under the same license or any compatible license↙
    ↳ .<br />
The author(s) of the original work may give you the right to modify it under the↙
    ↳  same conditions as the copies. </p>
<p><strong>3. RELATED RIGHTS </strong><br />
Activities giving rise to author's rights and related rights shall not challenge↙
    ↳  the rights granted by this license.<br />
For example, this is the reason why performances must be subject to the same ↙
    ↳ license or a compatible license. Similarly, integrating the work in a ↙
    ↳ database, a compilation or an anthology shall not prevent anyone from ↙
    ↳ using the work under the same conditions as those defined in this license.↙
    ↳  </p>
<p><strong>4. INCORPORATION OF THE WORK </strong><br />
Incorporating this work into a larger work that is not subject to the Free Art ↙
    ↳ License shall not challenge the rights granted by this license.<br />
If the work can no longer be accessed apart from the larger work in which it is ↙
```

```
    ↪ incorporated, then incorporation shall only be allowed under the condition ↙
    ↪  that the larger work is subject either to the Free Art License or a ↙
    ↪ compatible license. </p>
<p><strong>5. COMPATIBILITY </strong><br />
A license is compatible with the Free Art License provided:<br />
it gives the right to copy, distribute, and modify copies of the work including ↙
    ↪ for commercial purposes and without any other restrictions than those ↙
    ↪ required by the respect of the other compatibility criteria;<br />
it ensures proper attribution of the work to its authors and access to previous ↙
    ↪ versions of the work when possible;<br />
it recognizes the Free Art License as compatible (reciprocity);<br />
it requires that changes made to the work be subject to the same license or to a↙
    ↪  license which also meets these compatibility criteria. </p>
<p><strong>6. YOUR INTELLECTUAL RIGHTS </strong><br />
This license does not aim at denying your author's rights in your contribution ↙
    ↪ or any related right. By choosing to contribute to the development of this↙
    ↪  common work, you only agree to grant others the same rights with regard ↙
    ↪ to your contribution as those you were granted by this license. Conferring↙
    ↪  these rights does not mean you have to give up your intellectual rights. ↙
    ↪ </p>
<p><strong>7. YOUR RESPONSIBILITIES </strong><br />
The freedom to use the work as defined by the Free Art License (right to copy, ↙
    ↪ distribute, modify) implies that everyone is responsible for their own ↙
    ↪ actions. </p>
<p><strong>8. DURATION OF THE LICENSE </strong><br />
This license takes effect as of your acceptance of its terms. The act of copying↙
    ↪ , distributing, or modifying the work constitutes a tacit agreement. This ↙
    ↪ license will remain in effect for as long as the copyright which is ↙
    ↪ attached to the work. If you do not respect the terms of this license, you↙
    ↪  automatically lose the rights that it confers.<br />
If the legal status or legislation to which you are subject makes it impossible ↙
    ↪ for you to respect the terms of this license, you may not make use of the ↙
    ↪ rights which it confers. </p>
<p><strong>9. VARIOUS VERSIONS OF THE LICENSE </strong><br />
This license may undergo periodic modifications to incorporate improvements by ↙
    ↪ its authors (instigators of the "Copyleft Attitude" movement) by way of ↙
    ↪ new, numbered versions.<br />
You will always have the choice of accepting the terms contained in the version ↙
    ↪ under which the copy of the work was distributed to you, or alternatively,↙
    ↪  to use the provisions of one of the subsequent versions. </p>
<p><strong>10. SUB-LICENSING </strong><br />
Sub-licenses are not authorized by this license. Any person wishing to make use ↙
    ↪ of the rights that it confers will be directly bound to the authors of the↙
    ↪  common work. </p>
<p><strong>11. LEGAL FRAMEWORK</strong><br />
This license is written with respect to both French law and the Berne Convention↙
    ↪  for the Protection of Literary and Artistic Works. </p>
<p><strong>USER GUIDE </strong></p>
<p><em>- How to use the Free Art License? </em><br />
To benefit from the Free Art License, you only need to mention the following ↙
    ↪ elements on your work:<br />
[Name of the author, title, date of the work. When applicable, names of authors ↙
    ↪ of the common work and, if possible, where to find the originals].<br />
Copyleft: This is a free work, you can copy, distribute, and modify it under the↙
    ↪  terms of the Free Art License http://artlibre.org/licence/lal/en/ </p>
<p><em>- Why to use the Free Art License? </em><br />
1.To give the greatest number of people access to your work.<br />
```

```
2.To allow it to be distributed freely.<br />
3.To allow it to evolve by allowing its copy, distribution, and transformation by others.<br />
4.So that you benefit from the resources of a work when it is under the Free Art License: to be able to copy, distribute or transform it freely.<br />
5.But also, because the Free Art License offers a legal framework to disallow any misappropriation. It is forbidden to take hold of your work and bypass the creative process for one's exclusive possession.</p>
<p><em>- When to use the Free Art License? </em><br />
Any time you want to benefit and make others benefit from the right to copy, distribute and transform creative works without any exclusive appropriation, you should use the Free Art License. You can for example use it for scientific, artistic or educational projects. </p>
<p><em>- What kinds of works can be subject to the Free Art License? </em><br />
The Free Art License can be applied to digital as well as physical works.<br />
You can choose to apply the Free Art License on any text, picture, sound, gesture, or whatever sort of stuff on which you have sufficient author's rights.</p>
<p><em>- Historical background of this license: </em><br />
It is the result of observing, using and creating digital technologies, free software, the Internet and art. It arose from the "Copyleft Attitude" meetings which took place in Paris in 2000. For the first time, these meetings brought together members of the Free Software community, artists, and members of the art world. The goal was to adapt the principles of Copyleft and free software to all sorts of creations. http://www.artlibre.org </p>
<p>Copyleft Attitude, 2007.<br />
You can make reproductions and distribute this license verbatim (without any changes). </p>
<ul>
<code>Translation : Jonathan Clarke, Benjamin Jean, Griselda Jung, Fanny Mourguet, Antoine Pitrou.<br />
Thanks to <a href="http://www.framalang.org">framalang.org</a></code></ul>

</div>
<div class="post_separation">
<small>
Cette page a été éditée le 20/07/2005</small>
</div>
</div>
</div></div>
<div class="footer">
<br/><br/>
|
<a href="http://artlibre.org/licence/lal">Licence Art Libre</a> |
<a href="http://artlibre.org/licence/faq">FAQ</a> |
<a href="http://oeuvres.artlibre.org/">Les Oeuvres</a> |
<a href="http://artlibre.org/news">Actualit&eacute;s</a> |
<a href="http://artlibre.org/copyleft/listes/">Listes de Diffusion</a> |
</div>

<div class="footer_print">
Document publié sur <a href="http://artlibre.org">artlibre.org</a><br/>
Vous pouvez le consulter en ligne depuis <a href="#">http://artlibre.org/licence/lal/en</a>
</div>
</body>
```

</html>

30 src/Machine.hs

```
module Machine(Machine(), Opcode(..), machine, step, pretty, run, getCode, ↙
    ↳ getData) where

import Data.Maybe(catMaybes, listToMaybe)

import qualified DataTape as D

swap :: (a, b) -> (b, a)
swap p = (snd p, fst p)

data Opcode = OpToggle | OpUp | OpLeft | OpRight | OpDownLeft | OpDownRight | ↙
    ↳ OpInput | OpOutput | OpLoopStart | OpLoopEnd
  deriving (Read, Show, Eq, Enum)

opcodes :: [(Char, Opcode)]
opcodes = "+^<>/\\,;[]" `zip` [OpToggle, OpUp, OpLeft, OpRight, OpDownLeft, ↙
    ↳ OpDownRight, OpInput, OpOutput, OpLoopStart, OpLoopEnd]

opchars :: [(Opcode, Char)]
opchars = map swap opcodes

parse :: String -> [Opcode]
parse = catMaybes . map (`lookup` opcodes)

prettyOp :: [Opcode] -> String
prettyOp = catMaybes . map (`lookup` opchars)

data Instruction = InOp Opcode | InLoop [Instruction]
  deriving (Read, Show, Eq)

compile :: [Opcode] -> Maybe [Instruction]
compile = fixup . foldr compile' (Just [[]])
  where
    compile' OpLoopEnd   (Just (     ls)) = Just $             []  : ls
    compile' OpLoopStart (Just (l:l':ls)) = Just $ (InLoop l : l') : ls
    compile' op          (Just (l   :ls)) = Just $ (InOp  op : l ) : ls
    compile' _           _                = Nothing
    fixup (Just [is]) = Just is
    fixup _           = Nothing

prettyIn :: [Instruction] -> String
prettyIn [] = []
prettyIn (InOp  op : is) = prettyOp [op] ++ prettyIn is
prettyIn (InLoop l : is) = "[" ++ prettyIn l ++ "]" ++ prettyIn is

data CodeTape = CodeTape{ ctParent :: Maybe CodeTape, ctLeft :: [Instruction], ↙
    ↳ ctRight :: [Instruction] }
  deriving (Read, Show)

codeTape :: [Instruction] -> CodeTape
codeTape is = CodeTape Nothing [] is

getCode :: Machine -> ([Opcode], [Opcode])
```

```
50  getCode m = let (ls, rs) = getCode' ([], []) (mCode m) in (reverse ls, rs)

    getCode' :: ([Opcode], [Opcode]) -> CodeTape -> ([Opcode], [Opcode])
    getCode' (ls, rs) ct = case ctParent ct of
      Nothing ->          (                 ops (reverse $ ctLeft ct) ++ ls, rs ++ ↙
          ↳ ops (ctRight ct))
55    Just pt -> getCode' ([OpLoopStart] ++ ops (reverse $ ctLeft ct) ++ ls, rs ++ ↙
          ↳ ops (ctRight ct) ++ [OpLoopEnd]) pt{ ctRight = drop 1 (ctRight pt) }

    ops :: [Instruction] -> [Opcode]
    ops = concatMap op1

60  op1 :: Instruction -> [Opcode]
    op1 (InOp   o) = [o]
    op1 (InLoop l) = [OpLoopStart] ++ ops l ++ [OpLoopEnd]

    prettyCT :: CodeTape -> String
65  prettyCT ct = prettyCT' "#" ct

    prettyCT' :: String -> CodeTape -> String
    prettyCT' s ct = case ctParent ct of
      Nothing -> prettyIn (reverse (ctLeft ct)) ++ s ++ prettyIn (ctRight ct)
70    Just pt -> prettyCT' ("{[" ++ prettyCT' s ct{ ctParent = Nothing } ++ "]}") pt↙
          ↳ { ctRight = tail (ctRight pt) }

    data Machine = Machine{ mCode :: CodeTape, mData :: D.DataTape, mInput :: [Bool]↙
        ↳ }

    machine :: String -> [Bool] -> [Bool] -> Maybe Machine
75  machine code dpos input = case compile . parse $ code of
      Just is -> Just Machine{ mCode = codeTape is, mData = D.dataTape dpos, mInput ↙
          ↳ = input }
      Nothing -> Nothing

    step :: Machine -> (Maybe (Machine, Opcode), Maybe Bool)
80  step m = case nextIn m of
      Just (InOp o) -> let md f = (Just (m{ mData = f (mData m) }, o), Nothing)
                       in advance' $ case o of
        OpToggle     -> md (D.modify not)
        OpUp         -> md D.up
85      OpLeft       -> md D.left
        OpRight      -> md D.right
        OpDownLeft   -> md D.downLeft
        OpDownRight  -> md D.downRight
        OpInput      -> case (mInput m) of
90        []     -> (Just (m, o), Nothing)
          (i:is) -> (Just (m{ mData = D.modify (const i) (mData m), mInput = is }, o↙
              ↳ ), Nothing)
        OpOutput     -> let b = D.get (mData m)
                            f = if null (mInput m) then D.modify (const False) else ↙
                                ↳ id
                        in (fst (md f), Just b)
95      _            -> error "Machine.step: internal error: unexpected opcode"
      Just (InLoop is) -> let b = D.get (mData m)
                          in if b then (Just (m{ mCode = (codeTape is){ ctParent = ↙
                              ↳ Just (mCode m) } }, OpLoopStart), Nothing)
                                  else advance' (Just (m, OpLoopStart), Nothing)
```

```
    Nothing -> case ctParent (mCode m) of
      Nothing -> (Nothing, Nothing)
      Just cp -> (Just (m{ mCode = cp }), OpLoopEnd), Nothing)
    where
      advance' :: (Maybe (Machine, Opcode), Maybe Bool) -> (Maybe (Machine, Opcode↙
          ↳ ), Maybe Bool)
      advance' (Just (m', o'), out) = (Just (m'{ mCode = advance (mCode m') }, o')↙
          ↳ , out)
      advance' (Nothing,       out) = (Nothing, out)

nextIn :: Machine -> Maybe Instruction
nextIn m = listToMaybe (ctRight (mCode m))

advance :: CodeTape -> CodeTape
advance (CodeTape p ls (r:rs)) = CodeTape p (r:ls) rs
advance (CodeTape _ _  [])     = error "Machine.advance: internal error: no more↙
    ↳  code"

pretty :: Machine -> String
pretty m = D.pretty (mData m) ++ prettyCT (mCode m) ++ map (\b -> if b then '1' ↙
    ↳ else '0') (take 72 (mInput m))

run :: Machine -> ([Machine], [Bool])
run m = case step m of
  (Nothing, _) -> ([m], [])
  (Just (m', _), o) -> let (ms, os) = run m' in (m : ms, maybe os (:os) o)

getData :: Machine -> D.DataTape
getData m = mData m
```

31 src/Makefile

```
CC = gcc
CFLAGS = -std=c99 -Wall -pedantic -O3

all: KJHF font.rgba tile.rgba code.rgba kjhf.dat

clean:
	-rm -f KJHF KJHF.hi KJHF.o Machine.hi Machine.o DataTape.hi DataTape.o ↙
	    ↳ Snapshot.hi Snapshot.o interleave31 font.rgba tile.rgba code.rgba

KJHF: KJHF.hs Machine.hs DataTape.hs Snapshot.hs
	ghc -O2 -Wall --make KJHF.hs

%.rgba: %.png interleave31 png2rgba.sh
	./png2rgba.sh $*

interleave31: interleave31.c
	$(CC) $(CFLAGS) -s -o interleave31 interleave31.c

kjhf.dat:
	echo "inspect getdat.sh"
```

32 src/monk.pd

```
#N canvas 0 0 450 483 10;
```

```
#X obj 96 423 outlet~;
#X obj 203 377 *~;
#X obj 193 244 mtof~;
#X obj 270 220 -~ 80;
#X obj 270 245 /~ 60;
#X obj 267 270 *~;
#X obj 254 292 sig~ 1;
#X obj 253 317 -~;
#X obj 254 339 max~ 0;
#X obj 198 191 vline~;
#X obj 318 14 inlet;
#X obj 143 67 f \$1;
#X obj 143 88 +;
#X obj 197 109 moses 152;
#X obj 143 110 moses 8;
#X obj 198 163 f;
#X obj 144 128 + 144;
#X obj 248 130 - 144;
#X obj 96 18 inlet~;
#X obj 318 429 outlet;
#X obj 143 41 t b f;
#X obj 193 266 *~ 0.125;
#X obj 193 221 lop~ 2;
#X obj 175 288 noise~;
#X obj 175 314 *~ 16;
#X obj 175 340 vcf~ 16;
#X connect 1 0 0 0;
#X connect 2 0 21 0;
#X connect 3 0 4 0;
#X connect 4 0 5 0;
#X connect 4 0 5 1;
#X connect 5 0 7 1;
#X connect 6 0 7 0;
#X connect 7 0 8 0;
#X connect 8 0 1 1;
#X connect 9 0 3 0;
#X connect 9 0 22 0;
#X connect 10 0 19 0;
#X connect 10 0 20 0;
#X connect 11 0 12 0;
#X connect 12 0 14 0;
#X connect 13 0 15 0;
#X connect 13 1 17 0;
#X connect 14 0 16 0;
#X connect 14 1 13 0;
#X connect 15 0 11 1;
#X connect 15 0 9 0;
#X connect 16 0 15 0;
#X connect 17 0 15 0;
#X connect 18 0 0 0;
#X connect 20 0 11 0;
#X connect 20 1 12 1;
#X connect 21 0 25 1;
#X connect 22 0 2 0;
#X connect 23 0 24 0;
#X connect 24 0 25 0;
#X connect 25 0 1 0;
```

33 src/monks2.pd

```
#N canvas 0 0 450 300 10;
#X obj 105 218 *~;
#X obj 151 217 *~;
#X obj 167 139 clip~ 0 0.25;
#X obj 120 183 cos~;
#X obj 166 162 -~ 0.25;
#X obj 166 183 cos~;
#X obj 94 21 inlet;
#X obj 166 66 vline~;
#X obj 167 119 *~ 0.125;
#X obj 167 100 +~ 1;
#X obj 166 15 inlet;
#X msg 166 39 \$1 125 \, 0 125 125;
#X obj 104 266 outlet~;
#X obj 156 266 outlet~;
#X obj 111 135 monks;
#X obj 52 134 monks;
#X connect 0 0 12 0;
#X connect 1 0 13 0;
#X connect 2 0 3 0;
#X connect 2 0 4 0;
#X connect 3 0 0 1;
#X connect 4 0 5 0;
#X connect 5 0 1 1;
#X connect 6 0 14 0;
#X connect 6 0 15 0;
#X connect 7 0 9 0;
#X connect 8 0 2 0;
#X connect 9 0 8 0;
#X connect 10 0 11 0;
#X connect 11 0 7 0;
#X connect 14 0 1 0;
#X connect 15 0 0 0;
```

34 src/monks.pd

```
#N canvas 0 0 360 437 10;
#X obj 124 397 outlet~;
#X obj 214 16 inlet;
#X obj 217 256 vline~;
#X obj 217 288 *~;
#X obj 217 308 *~;
#X obj 124 341 *~;
#X obj 60 369 bp~ 500 3;
#X obj 124 60 monk 12;
#X obj 124 80 monk 24;
#X obj 124 100 monk 36;
#X obj 124 120 monk 48;
#X obj 124 140 monk 60;
#X obj 124 160 monk 72;
#X obj 124 181 monk 84;
#X obj 124 202 monk 96;
#X obj 124 223 monk 108;
#X obj 124 244 monk 120;
```

```
#X obj 124 265 monk 132;
#X obj 124 286 monk 144;
#X obj 123 369 bp~ 1000 3;
#X obj 193 369 bp~ 2000 3;
#X msg 217 232 0.8 5 \, 0.5 1000 5;
#X connect 1 0 21 0;
#X connect 1 0 7 1;
#X connect 2 0 3 0;
#X connect 2 0 3 1;
#X connect 3 0 4 0;
#X connect 3 0 4 1;
#X connect 4 0 5 1;
#X connect 5 0 6 0;
#X connect 5 0 19 0;
#X connect 5 0 20 0;
#X connect 6 0 0 0;
#X connect 7 0 8 0;
#X connect 7 1 8 1;
#X connect 8 0 9 0;
#X connect 8 1 9 1;
#X connect 9 0 10 0;
#X connect 9 1 10 1;
#X connect 10 0 11 0;
#X connect 10 1 11 1;
#X connect 11 0 12 0;
#X connect 11 1 12 1;
#X connect 12 0 13 0;
#X connect 12 1 13 1;
#X connect 13 0 14 0;
#X connect 13 1 14 1;
#X connect 14 0 15 0;
#X connect 14 1 15 1;
#X connect 15 0 16 0;
#X connect 15 1 16 1;
#X connect 16 0 17 0;
#X connect 16 1 17 1;
#X connect 17 0 18 0;
#X connect 17 1 18 1;
#X connect 18 0 5 0;
#X connect 19 0 0 0;
#X connect 20 0 0 0;
#X connect 21 0 2 0;
```

35 src/png2rgba.sh

```
#!/bin/bash
PNGFILE="${1}.png"
PPMFILE="${1}.ppm"
PGMFILE="${1}.pgm"
RGBFILE="${1}.rgb"
AFILE="${1}.a"
RGBAFILE="${1}.rgba"
pngtopnm        "${PNGFILE}" > "${PPMFILE}"
pngtopnm -alpha "${PNGFILE}" > "${PGMFILE}"
tail -c 3145728 "${PPMFILE}" > "${RGBFILE}"
tail -c 1048576 "${PGMFILE}" > "${AFILE}"
./interleave31 "${RGBFILE}" "${AFILE}" > "${RGBAFILE}"
```

```
rm -f "${PPMFILE}" "${PGMFILE}" "${RGBFILE}" "${AFILE}"
```

36 src/README

```
for concept documentation see files in ../prototype/

example usage:

make &&
./KJHF '+[^+\;^]'

still early days, more info to follow...
```

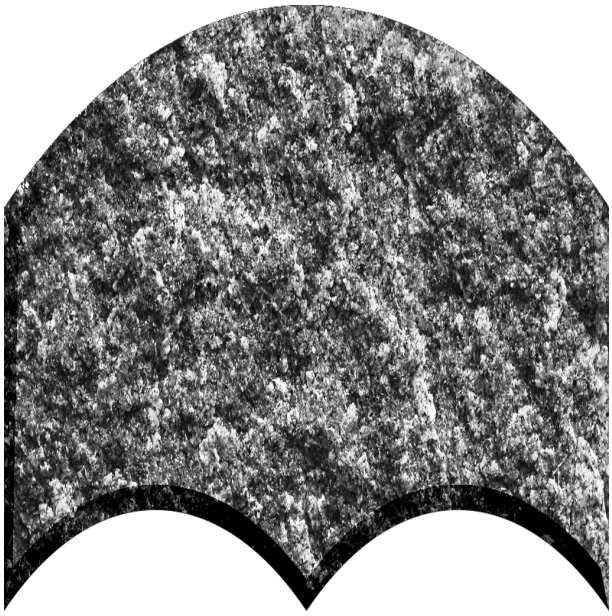
37 src/Setup.hs

```
import Distribution.Simple
main = defaultMain
```

38 src/Snapshot.hs

```
module Snapshot (hSnapshot, writeSnapshot, snapshotWith) where

import Control.Monad(forM_)
import System.IO(Handle())
import Graphics.UI.GLUT(
  readPixels,
  Position,
  Size(Size),
  PixelData(PixelData),
  PixelFormat(RGB),
  DataType(UnsignedByte))
import Foreign.Marshal.Alloc(allocaBytes)
import Foreign.Ptr(plusPtr)
import qualified Data.ByteString.Internal as BSI
import qualified Data.ByteString as BS

-- save a screenshot to a handle as binary PPM
snapshotWith :: (BS.ByteString -> IO b) -> Position -> Size -> IO b
snapshotWith f p0 vp@(Size vw vh) = do
  let fi q = fromIntegral q
      p6 = "P6\n" ++ show vw ++ " " ++ show vh ++ " 255\n"
  allocaBytes (fi (vw*vh*3)) $ \ptr -> do
    readPixels p0 vp $ PixelData RGB UnsignedByte ptr
    px <- BSI.create (fi $ vw * vh * 3) $ \d -> forM_ [0..vh-1] $ \y ->
      BSI.memcpy
        (d`plusPtr`fi(y*vw*3))
        (ptr`plusPtr`fi ((vh-1-y)*vw*3))
        (fi(vw*3))
    f $ BS.pack (map (toEnum . fromEnum) p6) `BS.append` px

hSnapshot :: Handle -> Position -> Size -> IO ()
hSnapshot h = snapshotWith (BS.hPutStr h)

writeSnapshot :: FilePath -> Position -> Size -> IO ()
writeSnapshot f = snapshotWith (BS.writeFile f)
```

39 src/tile.png



40 src/video.sh

```
#!/bin/bash
./KJHF '+[^+\;^]' |
ppmtoy4m -S 444 -F 25:1 |
y4mscaler -I sar=1/1 -O preset=dvd_wide -O yscale=1/1 |
mpeg2enc -f 8 -q 3 -b 8000 -B 768 -D 9 -g 9 -G 15 -P -R 2 -o kjhf.m2v
# TODO: render kjhf.score to kjhf.wav using Pd batch mode
# twolame -b 224 kjhf.wav &&
# mplex -f 8 -V -o kjhf.mpeg kjhf.m2v kjhf.mp2 &&
# rm kjhf.wav kjhf.mp2 kjhf.m2v kjhf.score &&
# ffmpeg2theora -p preview kjhf.mpeg
```

41 web/kjhf.svg

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/
    ↪ 1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/
    ↪ xlink"
 width="1280" height="640" onload="init(evt)" viewbox="0 0 1280 640"
><title>King James Hyperfuck</title
><desc>...</desc
><script type="text/ecmascript"><![CDATA[

// svg variables
var ns = "http://www.w3.org/2000/svg";
var width = 1280;
var height = 640;
var clock = 16;
var svgdoc;
```

```
15  var svgroot;

    // the machine
    var mach;

20  // initialization callback
    function init(evt) {
      svgdoc = evt.target.ownerDocument;
      svgroot = svgdoc.documentElement;
      var bg = svgdoc.createElementNS(ns, "rect");
25    bg.setAttributeNS(null, "x", 0);
      bg.setAttributeNS(null, "y", 0);
      bg.setAttributeNS(null, "width", width);
      bg.setAttributeNS(null, "height", height);
      bg.setAttributeNS(null, "stroke", "none");
30    bg.setAttributeNS(null, "fill", "white");
      svgroot.appendChild(bg);
      var t0 = svgdoc.createElementNS(ns, "g");
      t0.transform.baseVal.initialize(svgroot.createSVGTransformFromMatrix(svgroot.↙
          ↪ createSVGMatrix().translate(width/2.0,height).scale(height/40.0).flipY()↙
          ↪ ));
      var ts = svgdoc.createElementNS(ns, "g");
35    svgroot.tiles = ts;
      t0.appendChild(ts);
      svgroot.appendChild(t0);
      var c0 = svgdoc.createElementNS(ns, "g");
      addGlyph(c0, 0, "M 0 -1 L 0 -2 L 5 -2 L 5 -1 L 0 -1 Z M 0 6 L 5 6 L 5 7 L 0 7 ↙
          ↪ L 0 6 Z"); // cursor
40    c0.transform.baseVal.initialize(svgroot.createSVGTransformFromMatrix(svgroot.↙
          ↪ createSVGMatrix().translate(width/2.0,height/5.0).scale(height/40.0)));
      var cs = svgdoc.createElementNS(ns, "g");
      cs.transform.baseVal.initialize(svgroot.createSVGTransformFromMatrix(svgroot.↙
          ↪ createSVGMatrix().translate(width/2.0,height/5.0).scale(height/40.0)));
      svgroot.code = cs;
      svgroot.appendChild(c0);
45    svgroot.appendChild(cs);
      var args = parseLocation('' + evt.target.ownerDocument.location);
      codeGlyphs(cs, args.code);
      mach = kjhf(args.code);
      window.setTimeout("mainLoop()", 30);
50  }

    // parse location into associative array
    function parseLocation(q) {
      var data = new Object();
55    q = q.substring(q.indexOf('?') + 1);
      if (q.length >= 1) {
        var kv = new Object();
        var n = 1;
        while (q.indexOf('&') > -1) {
60        kv[n] = q.substring(0, q.indexOf('&'));
          q = q.substring(q.indexOf('&') + 1 );
          n = n + 1;
        }
        kv[n] = q;
65      for (var i = 0; i <= n; i = i + 1) {
          var k = (''+kv[i]).substring(0, (''+kv[i]).indexOf('='));
```

```
        var v = (''+kv[i]).substring((''+kv[i]).indexOf('=') + 1);
        data[decodeURIComponent(k)] = decodeURIComponent(v);
      }
    }
    return data;
}

// generate code display
function codeGlyphs(s, code) {
  debug(code);
  var i;
  for (i = 0; i < code.length; i = i + 1) {
    var c = code.substring(i, i + 1);
    switch (c) {
    case '+': addGlyph(s, i, "M 0 2 L 2 2 L 2 0 L 3 0 L 3 2 L 5 2 L 5 3 L 3 3 L↙
        ↳ 3 5 L 2 5 L 2 3 L 0 3 L 0 2 Z"); break;
    case '^': addGlyph(s, i, "M 0 2.5 L 2.5 0 L 5 2.5 L 3 2.5 L 3 5 L 2 5 L 2 ↙
        ↳ 2.5 L 0 2.5 Z"); break;
    case '<': addGlyph(s, i, "M 0 2.5 L 2.5 0 L 2.5 2 L 5 2 L 5 3 L 2.5 3 L 2.5↙
        ↳ 5 L 0 2.5 Z"); break;
    case '>': addGlyph(s, i, "M 0 2 L 2.5 2 L 2.5 0 L 5 2.5 L 2.5 5 L 2.5 3 L 0↙
        ↳ 3 L 0 2 Z"); break;
    case '/': addGlyph(s, i, "M 0 1 L 1.5 2.5 L 4 0 L 5 1 L 2.5 3.5 L 4 5 L 0 5↙
        ↳ L 0 1 Z"); break;
    case '\\': addGlyph(s, i, "M 0 1 L 1 0 L 3.5 2.5 L 5 1 L 5 5 L 1 5 L 2.5 3.5↙
        ↳ L 0 1 Z"); break;
    case ',': addGlyph(s, i, "M 1 4 L 2 4 L 2 3 L 3 3 L 3 5 L 1 5 L 1 4 Z"); ↙
        ↳ break;
    case ';': addGlyph(s, i, "M 1 4 L 2 4 L 2 3 L 3 3 L 3 5 L 1 5 L 1 4 Z M 2 1↙
        ↳ L 3 1 L 3 2 L 2 2 L 2 1 Z"); break;
    case '[': addGlyph(s, i, "M 1 0 L 4 0 L 4 1 L 2 1 L 2 4 L 4 4 L 4 5 L 1 5 L↙
        ↳ 1 0 Z"); break;
    case ']': addGlyph(s, i, "M 1 0 L 4 0 L 4 5 L 1 5 L 1 4 L 3 4 L 3 1 L 1 1 L↙
        ↳ 1 0 Z"); break;
    default: break;
    }
  }
}

// add a code glyph to a container
function addGlyph(s, i, d) {
  var x0 = (i - 0.5) * 6.0 + 0.5;
  var y0 = 1.0;
  var back = svgdoc.createElementNS(ns, "path");
  back.setAttributeNS(null, "d", d);
  back.setAttributeNS(null, "stroke", "yellow");
  back.setAttributeNS(null, "stroke-width", 0.5);
  back.setAttributeNS(null, "fill", "none");
  back.setAttributeNS(null, "transform", "translate(" + x0 + "," + y0 + ")");
  var front = svgdoc.createElementNS(ns, "path");
  front.setAttributeNS(null, "d", d);
  front.setAttributeNS(null, "stroke", "black");
  front.setAttributeNS(null, "stroke-width", 0.2);
  front.setAttributeNS(null, "fill", "yellow");
  front.setAttributeNS(null, "fill-opacity", 0.25);
  front.setAttributeNS(null, "transform", "translate(" + x0 + "," + y0 + ")");
  s.appendChild(back);
```

```
  s.appendChild(front);
}

// main loop callback
function mainLoop() {
  run();
  window.setTimeout("mainLoop()", 30);
}

// main loop
var frame = 0;
function run() {
  if (mach.running) {
    if (frame == 0) {
      mach.dataPtr.space.matrix0 = mach.dataPtr.space.matrix;
      mach.dataPtr.space.ds = 1.0;
      mach.dataPtr.space.dx = 0.0;
      mach = step(mach);
    }
    var m = svgroot.createSVGMatrix().translate(mach.dataPtr.space.dx * (frame +↙
        ↳ 1) * 1.0 / clock, 0.0).scale(Math.pow(mach.dataPtr.space.ds, (frame +↙
        ↳ 1) * 1.0 / clock));
    mach.dataPtr.space.matrix = m.multiply(mach.dataPtr.space.matrix0);
    svgroot.tiles.transform.baseVal.initialize(svgroot.↙
        ↳ createSVGTransformFromMatrix(mach.dataPtr.space.matrix));
    frame = (frame + 1) % clock;
  }
}

function cell(matrix) {
  var c = { up: null, left: null, right: null, downLeft: null, downRight: null, ↙
      ↳ data: 0 };
  var s = svgdoc.createElementNS(ns, "path");
  s.cell = c;
  c.svg = s;
  s.transform.baseVal.initialize(svgroot.createSVGTransformFromMatrix(matrix));
  s.setAttributeNS(null, "d", "M 0,4 L 0,8 A 10,10 0 0 0 12,8 L 12,4 A 5,5 0 0 1↙
      ↳ 6,4 A 5,5 0 0 1 0,4 Z");
  s.setAttributeNS(null, "stroke", "black");
  s.setAttributeNS(null, "stroke-width", 0.1);
  s.setAttributeNS(null, "fill", "silver");
  svgroot.tiles.appendChild(s);
  return c;
}

function space(ibits) {
  return {
    above: ibits,
    top: cell(svgroot.createSVGMatrix().translate(-6.0,0.0)),
    matrix: svgroot.createSVGMatrix(),
    ds: 1.0,
    dx: 0.0
  };
}

function cursor(spc) {
  return {
```

```
    space: spc,
    cell: spc.top,
    matrix: svgroot.createSVGMatrix().translate(-6.0,0.0)
  };
}

function up(o, wantRight) {
  debug("up");
  var m = svgroot.createSVGMatrix();
  var dx;
  m = m.scale(2.0);
  if (! o.cell.up) {
    var r;
    if (o.space.above.end) {
      r = wantRight;
    } else {
      o.space.above = o.space.above.get(o.space.above);
      r = o.space.above.bit;
    }
    if (r) {
      m = m.translate(-6.0, 0.0);
      dx =  3.0;
    } else {
      dx = -3.0;
    }
    o.matrix = o.matrix.multiply(m);
    o.cell.up = cell(o.matrix);
    if (r) { o.cell.up.downRight = o.cell; }
    else   { o.cell.up.downLeft  = o.cell; }
    o.space.top = o.cell.up;
  } else {
    if (o.cell != o.cell.up.downLeft) {
      m = m.translate(-6.0, 0.0);
      dx =  3.0;
    } else {
      dx = -3.0;
    }
    o.matrix = o.matrix.multiply(m);
  }
  o.cell = o.cell.up;
  o.space.ds = 0.5;
  o.space.dx = dx;
  return o;
}

function downLeft(o) {
  debug("downLeft");
  var m = svgroot.createSVGMatrix();
  m = m.scale(0.5);
  m = m.translate( 0.0, 0.0);
  o.matrix = o.matrix.multiply(m);
  if (!o.cell.downLeft) {
    o.cell.downLeft = cell(o.matrix);
    o.cell.downLeft.up = o.cell;
    if (o.cell.downRight) {
      o.cell.downRight.left = o.cell.downLeft;
      o.cell.downLeft.right = o.cell.downRight;
```

```
    }
    if (o.cell.left && o.cell.left.downRight) {
      o.cell.left.downRight.right = o.cell.downLeft;
      o.cell.downLeft.left = o.cell.left.downRight;
    }
  }
  o.cell = o.cell.downLeft;
  o.space.ds = 2.0;
  o.space.dx =  6.0;
  return o;
}

function downRight(o) {
  debug("downRight");
  var m = svgroot.createSVGMatrix();
  m = m.scale(0.5);
  m = m.translate(12.0, 0.0);
  o.matrix = o.matrix.multiply(m);
  if (!o.cell.downRight) {
    o.cell.downRight = cell(o.matrix);
    o.cell.downRight.up = o.cell;
    if (o.cell.downLeft) {
      o.cell.downLeft.right = o.cell.downRight;
      o.cell.downRight.left = o.cell.downLeft;
    }
    if (o.cell.right && o.cell.right.downLeft) {
      o.cell.right.downLeft.left = o.cell.downRight;
      o.cell.downRight.right = o.cell.right.downLeft;
    }
  }
  o.cell = o.cell.downRight;
  o.space.ds = 2.0;
  o.space.dx = -6.0;
  return o;
}

function left(o) {
  debug("{ left");
  if (!o.cell.left) {
    if      (o.cell.up && o.cell.up.downRight == o.cell) { /* RIGHT */
      o = up(o, true);
      o = downLeft(o);
      debug("} left");
      o.space.ds = 1.0;
      o.space.dx =  12.0;
      return o;
    } else if (o.cell.up && o.cell.up.downLeft  == o.cell) { /* LEFT */
      o = up(o, true);
      o = left(o);
      o = downRight(o);
      debug("} left");
      o.space.ds = 1.0;
      o.space.dx =  12.0;
      return o;
    } else if (!o.cell.up) {                                 /* TOP */
      var c;
      var n = 0;
```

```
280         var z = [ ];
            do {
              z[n] = o.cell;
              c = o.cell;
              o = up(o, true);
285           n = n + 1;
            } while (o.cell.downLeft == c);
            o = downLeft(o);
            while (1 < n) {
              n = n - 1;
290           z[n].left = o.cell;
              o.cell.right = z[n];
              o = downRight(o);
            }
            debug("} left");
295         o.space.ds = 1.0;
            o.space.dx =  12.0;
            return o;
          } else {
            alert("BROKEN: left");
300       }
        } else {
          o.cell = o.cell.left;
          var m = svgroot.createSVGMatrix();
          m = m.translate(-12.0, 0.0);
305       o.matrix = o.matrix.multiply(m);
          debug("} left");
          o.space.ds = 1.0;
          o.space.dx =  12.0;
          return o;
310     }
      }

      function right(o) {
        debug("{ right");
315     if (!o.cell.right) {
          if        (o.cell.up && o.cell.up.downRight == o.cell) { /* RIGHT */
            o = up(o, false);
            o = right(o);
            o = downLeft(o);
320         debug("} right");
            o.space.ds = 1.0;
            o.space.dx = -12.0;
            return o;
          } else if (o.cell.up && o.cell.up.downLeft  == o.cell) { /* LEFT */
325         o = up(o, false);
            o = downRight(o);
            debug("} right");
            o.space.ds = 1.0;
            o.space.dx = -12.0;
330         return o;
          } else if (!o.cell.up) {                                 /* TOP */
            var c;
            var n = 0;
            var z = [ ];
335         do {
              z[n] = o.cell;
```

```
        c = o.cell;
        o = up(o, false);
        n = n + 1;
      } while (o.cell.downRight == c);
      o = downRight(o);
      while (1 < n) {
        n = n - 1;
        z[n].right = o.cell;
        o.cell.left = z[n];
        o = downLeft(o);
      }
      debug("} right");
      o.space.ds = 1.0;
      o.space.dx = -12.0;
      return o;
    } else {
      alert("BROKEN: right");
    }
  } else {
    o.cell = o.cell.right;
    var m = svgroot.createSVGMatrix();
    m = m.translate( 12.0, 0.0);
    o.matrix = o.matrix.multiply(m);
    debug("} right");
    o.space.ds = 1.0;
    o.space.dx = -12.0;
    return o;
  }
}

function step(m) {
  var bit = m.dataPtr.cell.data != 0;
  var op = m.codePtr.op;
  switch (op) {
  case '#': m.running = false; break;
  case '+': m.dataPtr.cell.data = !bit ? 1 : 0;
            m.dataPtr.cell.svg.setAttributeNS(null, "fill", !bit ? "yellow" : "↙
                ↳ silver");
            break;
  case '^': m.dataPtr = up       (m.dataPtr, false); break;
  case '<': m.dataPtr = left     (m.dataPtr); break;
  case '>': m.dataPtr = right    (m.dataPtr); break;
  case '/': m.dataPtr = downLeft (m.dataPtr); break;
  case '\\': m.dataPtr = downRight(m.dataPtr); break;
  case ',': m.input = m.input.get(m.input);
            m.dataPtr.cell.data = m.input.bit ? 1 : 0;
            break;
  case ';': m.output = m.output.put(m.output, bit);
            if (m.input.end) { m.dataPtr.cell.data = 0; }
            break;
  default:  break;
  }
  svgroot.code.transform.baseVal.initialize(svgroot.createSVGTransformFromMatrix↙
      ↳ (svgroot.createSVGMatrix().translate(width/2.0,height/5.0).scale(height↙
      ↳ /40.0).translate(-6.0 * m.codePtr.index, 0)));
  m.codePtr = m.codePtr.next[bit ? 1 : 0];
  return m;
```

```
}

function parse(s) {
  /* pass 1: instruction allocation */
  var pc = { op: '#', index: -1, next: [ null, null ] };
  var pc0 = pc;
  var pc1, pc2;
  var depth;
  var i;
  for (i = 0; i < s.length; ++i) {
    var c = s[i];
    switch (c) {
    case '#':
    case '\0':
      break;
    case '+':
    case '^':
    case '<':
    case '>':
    case '/':
    case '\\':
    case ',':
    case ';':
    case '[':
    case ']':
      pc.op = c;
      pc.index = i;
      pc.next[0] = { op: '#', index: i + 1, next: [ null, null ] };
      pc.next[1] = pc.next[0];
      pc = pc.next[0];
      break;
    default:
    }
  }
  /* pass 2: bracket fixating, eg:  [+[+]+]
      ----------------------------------------
     /          ------------------            |
    /          /                  |           |
    skip,-flip->skip,->flip->skip<            |
        |           |------------>flip->skip<
        |-------------------------------------->END
  */
  pc = pc0;
  while (pc.op != '#') {
    pc2 = pc.next[0];
    if (pc.op == ']') { alert("BROKEN: ket"); }
    if (pc.op == '[') {
      pc1 = pc.next[0];
      depth = 1;
      while (depth > 0) {
        if       (pc1.op == '[') {
          depth++;
          pc1 = pc1.next[0];
        } else if (pc1.op == ']') {
          depth--;
          if (depth > 0) {
            pc1 = pc1.next[0];
```

```
          }
        } else {
          pc1 = pc1.next[0];
        }
      }
      if (! (depth == 0 && pc1 != null && pc1.op == ']')) { alert("BROKEN: bra")↙
          ↳ ; }
      pc1.next[1] = pc.next[0];
      pc.next[0] = pc1.next[0];
      pc.op = pc1.op = ' ';
    }
    pc = pc2;
  }
  return pc0;
}

function machine(c, d, i, o) {
  var s = space(d);
  var p = cursor(s);
  var pc = parse(c);
  return { input: i, output: o, data: s, dataPtr: p, code: pc, codePtr: pc, ↙
      ↳ running: true };
}

function kjhf(c) {
  return machine(
//     "+[>>>>>>>>>+[+/][<+]+[^]+\\[<+]+]",
//     "+[^/+;^+]#",
//     "+[+^+]#",
//     "+[[^\\>+^</+]+[^/<+^>\\+]+^+]#",
    c,
    { get: function(i) { i.bit = Math.random() > 0.5; return i; }, bit: false, ↙
        ↳ end: false },
    { get: function(i) { i.bit = Math.random() > 0.5; return i; }, bit: false, ↙
        ↳ end: false },
    { put: function(o, b) { /* alert("out: " + b); */ return o; } }
  )
}

function debug2(str){
//  throw new Error("debug: " + str)
}

function debug(str){
//  setTimeout("debug2('"+str+"')", 1)
}

// ]]></script></svg>
```