hypercube

Claude Heiland-Allen

2006–2017

Contents

1 AUTHORS

Claude Heiland-Allen
    main program

Nick Treleaven
    various optimizations for improved rendering speed

2 CHANGELOG

2017-06-29 FIX: 64bit compatibility
2007-04-08 NEW: patches from Nick Treleaven for improved performance
0.3.13 NEW: Pd sonification example is more aesthetic
0.3.12 NEW: OSC send of raw 2D vertex coordinates
0.3.11 NEW: OSC receive equivalents to keyboard input
0.3.10 FIX: 'pause' no longer uses 100% CPU
0.3.9 NEW: saving of consecutive frames
0.3.8 NEW: saving of images to PNG files
0.3.7 Fixes for OS X (signal.h defines stack_t)

3 COPYING

you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and

distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices
stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in
whole or in part contains or is derived from the Program or any
part thereof, to be licensed as a whole at no charge to all third
parties under the terms of this License.

c) If the modified program normally reads commands interactively
when run, you must cause it, when started running for such
interactive use in the most ordinary way, to print or display an
announcement including an appropriate copyright notice and a
notice that there is no warranty (or else, saying that you provide
a warranty) and that users may redistribute the program under
these conditions, and telling the user how to view a copy of this
License. (Exception: if the Program itself is interactive but
does not normally print such an announcement, your work based on
the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable
source code, which must be distributed under the terms of Sections
1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three
years, to give any third party, for a charge no more than your
cost of physically performing source distribution, a complete
machine-readable copy of the corresponding source code, to be
distributed under the terms of Sections 1 and 2 above on a medium
customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by

all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES

PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type 'show c' for details.

```
320
    The hypothetical commands 'show w' and 'show c' should show the appropriate
    parts of the General Public License.  Of course, the commands you use may
    be called something other than 'show w' and 'show c'; they could even be
    mouse-clicks or menu items--whatever suits your program.
325
    You should also get your employer (if you work as a programmer) or your
    school, if any, to sign a "copyright disclaimer" for the program, if
    necessary.  Here is a sample; alter the names:

330   Yoyodyne, Inc., hereby disclaims all copyright interest in the program
      'Gnomovision' (which makes passes at compilers) written by James Hacker.

      <signature of Ty Coon>, 1 April 1989
      Ty Coon, President of Vice
335
    This General Public License does not permit incorporating your program into
    proprietary programs.  If your program is a subroutine library, you may
    consider it more useful to permit linking proprietary applications with the
    library.  If this is what you want to do, use the GNU Library General
340 Public License instead of this License.
```

4 cube4d.c

```
    /*
    hypercube -- interaction in four dimensions.
    Copyright (C) 2006 Claude Heiland-Allen

 5  This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.

10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    /* cube4d object */
20
    #include "cube4d.h"
    #include "fixed.h"
    #include "geometry.h"
    #include "stack.h"
25
    /* vertices */
    int cube4d_vertices = 16;
    vector4 cube4d_vertex[16];

30  /* Euler trail */
    int cube4d_edges = 32;
    int cube4d_trailsize = 33;
    int cube4d_trail[33] = {
```

```
    0,  1,  3,  2,  6, 14, 10,  8,
    9, 11,  3,  7, 15, 14, 12, 13,
    9,  1,  5,  7,  6,  4, 12,  8,
    0,  4,  5, 13, 15, 11, 10,  2,
    0
};

/* sub-cubes */
int cube4d_cubes = 8;
int cube4d_cube[8][8];

/* comparator */
static comparator cmp_int;
static int cmp_int(stack_content a, stack_content b) {
    int ai, bi;
    ai = (int) a;
    bi = (int) b;
    return (ai - bi);
}

/* initialise */
void cube4d_init(void) {
    int i, j, v, b, h;
    stack s;
    for (i = 0; i < 16; i++) {
        for (j = 0; j < 4; j++) {
            cube4d_vertex[i][j] = (i & 1<<j) ? FIXED_ONE : f_neg(FIXED_ONE);
        }
    }
    for (i = 0; i < 8; i++) {
        s_clear(&s);
        for (v = 0; v < 16; v++) {
            b = i >> 1;
            h = i & 1;
            if (h) {
                s_push(&s, (stack_content) (v | (1<<b)));
            } else {
                s_push(&s, (stack_content) (v &~(1<<b)));
            }
        }
        s_sort(&s, cmp_int);
        s_uniq(&s, cmp_int);
        for (j = 0; j < 8; j++) {
            cube4d_cube[i][j] = (int) s_at(&s, j);
        }
    }
}

/* EOF */
```

5 cube4d.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
```

```
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.

10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    #ifndef CUBE4D_H
20  #define CUBE4D_H 1
    /* cube4d object */

    #include "geometry.h"

25  /* vertices */
    typedef vector4 c4dv[16];
    c4dv cube4d_vertex;
    int cube4d_vertices;

30  /* Euler trail */
    int cube4d_edges;
    int cube4d_trail[33];
    int cube4d_trailsize;

35  /* sub-cubes */
    int cube4d_cubes;
    int cube4d_cube[8][8];

    /* initialise vertices */
40  void cube4d_init(void);

    /* EOF */
    #endif
```

6 dynamics.c

```
    /*
    hypercube -- interaction in four dimensions.
    Copyright (C) 2006 Claude Heiland-Allen

 5  This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.

10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
```

```
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    /* movement */
20
    #include "cube4d.h"
    #include "dynamics.h"
    #include "fixed.h"
    #include "geometry.h"
25
    static matrix44 m_rot;

    void d_init(void) {
        m44_id(&m_rot);
30  }

    void d_transform(c4dv *from, c4dv *to,
        int a01, int a02, int a03, int a12, int a13, int a23
    ) {
35      static int r01 = 0;
        static int r02 = 0;
        static int r03 = 0;
        static int r12 = 0;
        static int r13 = 0;
40      static int r23 = 0;
        int i;
        r01 += a01;
        r02 += a02;
        r03 += a03;
45      r12 += a12;
        r13 += a13;
        r23 += a23;
        m44_rot6(&m_rot,
            r01,
50          r02,
            r03,
            r12,
            r13,
            r23
55      );
        for (i = 0; i < 16; i++) {
            v4m44_mul(&((*from)[i]), &m_rot, &((*to)[i]));
        }
    }
60
    /* EOF */
```

7 dynamics.h

```
    /*
    hypercube -- interaction in four dimensions.
    Copyright (C) 2006 Claude Heiland-Allen

 5  This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.
```

```
10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    #ifndef DYNAMICS_H
20  #define DYNAMICS_H
    /* movement */

    #include "cube4d.h"

25  void d_init(void);
    void d_transform(c4dv *from, c4dv *to,
        int a01, int a02, int a03, int a12, int a13, int a23
    );

30  /* EOF */
    #endif
```

8 fixed.c

```
    /*
    hypercube -- interaction in four dimensions.
    Copyright (C) 2006 Claude Heiland-Allen

 5  This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.

10  This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

15  You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    /* fixed point arithmetic */
20
    #include <math.h>

    #include "fixed.h"

25  void f_init(void) {
        int i;
        /* fill sinetable */
        for (i = 0; i < FIXED_TABSIZE; i++) {
            f_sintab[i] = FIXED_ONE * sin((double) i * 2 * M_PI / FIXED_TABSIZE);
30      }
    }
```

```
int f_cmp(fixed a, fixed b) {
    return (a - b);
}

/* EOF */
```

9 fixed.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef FIXED_H
#define FIXED_H 1
/* fixed point arithmetic */

typedef signed short fixed;

#define FIXED_ONE (128)
#define FIXED_BITS (7)
#define f_itof(a) ((fixed)((a)<<FIXED_BITS))

#define f_add(a,b) ((a)+(b))
#define f_sub(a,b) ((a)-(b))
#define f_neg(a)   (-(a))
#define f_mul(a,b) (((a)*(b))>>FIXED_BITS)
#define f_div(a,b) ((fixed)((((int)(a))<<FIXED_BITS)/(b)))
int f_cmp(fixed a, fixed b);

fixed f_sintab[512];
#define FIXED_TABSIZE (512)
#define FIXED_TABMASK (511)
#define FIXED_COSOFFSET (128)
#define f_sin(a)   (f_sintab[(a)&FIXED_TABMASK])
#define f_cos(a)   (f_sintab[((a)+FIXED_COSOFFSET)&FIXED_TABMASK])

void f_init(void);

/* EOF */
#endif
```

10 fm-osc~.pd

```
#N canvas 678 4 268 251 10;
#X obj 20 12 inlet~;
#X obj 17 219 osc~;
#X obj 17 189 +~;
#X obj 88 7 inlet;
#X obj 88 30 unpack f f;
#X obj 18 242 outlet~;
#X obj 17 124 *~;
#X obj 34 78 sig~;
#X obj 33 99 lop~ 10;
#X obj 87 76 / 640;
#X obj 88 168 mtof~;
#X obj 18 156 *~;
#X obj 89 144 scalify;
#X obj 182 12 inlet;
#X obj 87 99 * 20;
#X obj 87 124 + 10;
#X obj 17 31 hip~ 20;
#X obj 34 56 / 480;
#X connect 0 0 16 0;
#X connect 1 0 5 0;
#X connect 2 0 1 0;
#X connect 3 0 4 0;
#X connect 4 0 9 0;
#X connect 4 1 17 0;
#X connect 6 0 11 0;
#X connect 7 0 8 0;
#X connect 8 0 6 1;
#X connect 9 0 14 0;
#X connect 10 0 11 1;
#X connect 10 0 2 1;
#X connect 11 0 2 0;
#X connect 12 0 10 0;
#X connect 13 0 12 1;
#X connect 14 0 15 0;
#X connect 15 0 12 0;
#X connect 16 0 6 0;
#X connect 17 0 7 0;
```

11 geometry.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
```

```
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    /* coordinate geometry */

20
    #include "geometry.h"
    #include "fixed.h"

    void v2_add(vector2 *p, vector2 *q, vector2 *to) {
25      (*to)[0] = f_add((*p)[0], (*q)[0]);
        (*to)[1] = f_add((*p)[1], (*q)[1]);
    }

    void v2_sub(vector2 *p, vector2 *q, vector2 *to) {
30      (*to)[0] = f_sub((*p)[0], (*q)[0]);
        (*to)[1] = f_sub((*p)[1], (*q)[1]);
    }

    int v2_rturn(vector2 *p, vector2 *q, vector2 *r) {
35      return (
            (((int)((*q)[0]))*((int)((*r)[1])) +
             ((int)((*p)[0]))*((int)((*q)[1])) +
             ((int)((*r)[0]))*((int)((*p)[1]))) -
            (((int)((*q)[0]))*((int)((*p)[1])) +
40           ((int)((*r)[0]))*((int)((*q)[1])) +
             ((int)((*p)[0]))*((int)((*r)[1]))) < 0
        );
    }

45  void v3_add(vector3 *p, vector3 *q, vector3 *to) {
        (*to)[0] = f_add((*p)[0], (*q)[0]);
        (*to)[1] = f_add((*p)[1], (*q)[1]);
        (*to)[2] = f_add((*p)[2], (*q)[2]);
    }
50
    void v3_sub(vector3 *p, vector3 *q, vector3 *to) {
        (*to)[0] = f_sub((*p)[0], (*q)[0]);
        (*to)[1] = f_sub((*p)[1], (*q)[1]);
        (*to)[2] = f_sub((*p)[2], (*q)[2]);
55  }

    void v3_project(vector3 *p, fixed d, vector2 *to) {
        (*to)[0] = f_div(f_mul((*p)[0], d), f_sub(d, (*p)[2]));
        (*to)[1] = f_div(f_mul((*p)[1], d), f_sub(d, (*p)[2]));
60  }

    void v4_add(vector4 *p, vector4 *q, vector4 *to) {
        (*to)[0] = f_add((*p)[0], (*q)[0]);
        (*to)[1] = f_add((*p)[1], (*q)[1]);
65      (*to)[2] = f_add((*p)[2], (*q)[2]);
        (*to)[3] = f_add((*p)[3], (*q)[3]);
    }

    void v4_sub(vector4 *p, vector4 *q, vector4 *to) {
70      (*to)[0] = f_sub((*p)[0], (*q)[0]);
        (*to)[1] = f_sub((*p)[1], (*q)[1]);
        (*to)[2] = f_sub((*p)[2], (*q)[2]);
        (*to)[3] = f_sub((*p)[3], (*q)[3]);
```

```
}

void v4_project(vector4 *p, fixed d, vector3 *to) {
    (*to)[0] = f_div(f_mul((*p)[0], d), f_sub(d, (*p)[3]));
    (*to)[1] = f_div(f_mul((*p)[1], d), f_sub(d, (*p)[3]));
    (*to)[2] = f_div(f_mul((*p)[2], d), f_sub(d, (*p)[3]));
}

void m44_mul(matrix44 *p, matrix44 *q, matrix44 *to) {
    int i, j, k;
    int s;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            s = 0;
            for (k = 0; k < 4; k++) {
                s += ((*p)[i][k] * (*q)[k][j]);
            }
            (*to)[i][j] = s >> FIXED_BITS;
        }
    }
}

void v4m44_mul(vector4 *v, matrix44 *m, vector4 *to) {
    int i, j;
    int s;
    for (i = 0; i < 4; i++) {
        s = 0;
        for (j = 0; j < 4; j++) {
            s += ((*v)[j] * (*m)[i][j]);
        }
        (*to)[i] = s >> FIXED_BITS;
    }
}

void m44_id(matrix44 *m) {
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            (*m)[i][j] = f_itof(i == j);
        }
    }
}

void m44_rot1(matrix44 *m, int d1, int d2, int a) {
    fixed c, s;
    m44_id(m);
    c = f_cos(a);
    s = f_sin(a);
    (*m)[d1][d1] = c; (*m)[d2][d1] = -s;
    (*m)[d1][d2] = s; (*m)[d2][d2] =  c;
}

void m44_rot6(matrix44 *m, int a01, int a02, int a03,
                           int a12, int a13, int a23) {
    matrix44 m01, m02, m03, m12, m13, m23;
    matrix44 m0102, m0312, m1323;
    matrix44 m01020312;
```

```
    m44_rot1(&m01, 0, 1, a01);
    m44_rot1(&m02, 0, 2, a02);
    m44_rot1(&m03, 0, 3, a03);
    m44_rot1(&m12, 1, 2, a12);
    m44_rot1(&m13, 1, 3, a13);
    m44_rot1(&m23, 2, 3, a23);
    m44_mul(&m01, &m02, &m0102);
    m44_mul(&m03, &m12, &m0312);
    m44_mul(&m13, &m23, &m1323);
    m44_mul(&m0102, &m0312, &m01020312);
    m44_mul(&m01020312, &m1323, m);
}

/* EOF */
```

12 geometry.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef GEOMETRY_H
#define GEOMETRY_H 1
/* coordinate geometry */

#include "fixed.h"

typedef fixed vector2[2];
typedef fixed vector3[3];
typedef fixed vector4[4];
typedef fixed matrix44[4][4];

void v2_add(vector2 *p, vector2 *q, vector2 *to);
void v2_sub(vector2 *p, vector2 *q, vector2 *to);
int v2_rturn(vector2 *p, vector2 *q, vector2 *r);

void v3_add(vector3 *p, vector3 *q, vector3 *to);
void v3_sub(vector3 *p, vector3 *q, vector3 *to);
void v3_project(vector3 *p, fixed d, vector2 *to);

void v4_add(vector4 *p, vector4 *q, vector4 *to);
void v4_sub(vector4 *p, vector4 *q, vector4 *to);
void v4_project(vector4 *p, fixed d, vector3 *to);
```

```
void m44_mul(matrix44 *p, matrix44 *q, matrix44 *to);
void v4m44_mul(vector4 *v, matrix44 *m, vector4 *to);
void m44_id(matrix44 *m);
void m44_rot1(matrix44 *m, int d1, int d2, int a);
void m44_rot6(matrix44 *m, int a01, int a02, int a03,
                           int a12, int a13, int a23);

/* EOF */
#endif
```

13 .gitignore

```
hypercube
*.o
```

14 hull2d.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006,2017 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
/* 2d convex hull */

#include "stack.h"
#include "geometry.h"
#include "fixed.h"

static comparator hull2d_cmp;

static int hull2d_cmp(stack_content a, stack_content b) {
    vector2 *u = (vector2 *) a;
    vector2 *v = (vector2 *) b;
    int c = f_cmp((*u)[0], (*v)[0]);
    if (c) return (c);
    return (f_cmp((*u)[1], (*v)[1]));
}

void hull2d(stack *in, stack *to) {
    int i;
    stack p, u, l;
    s_copy(in, &p);
    s_sort(&p, hull2d_cmp);
```

```
    s_uniq(&p, hull2d_cmp);
    s_clear(&u);
    s_push(&u, s_at(&p, 0));
    s_push(&u, s_at(&p, 1));
    for (i = 2; i < s_size(&p); i++) {
        s_push(&u, s_at(&p, i));
        while (s_size(&u) > 2 &&
               !v2_rturn((vector2 *) s_at(&u, -3), (vector2 *) s_at(&u, -2), (vector2 *) s_at(&u, -1))) {
            s_delete(&u, -2);
        }
    }
    s_reverse(&p);
    s_clear(&l);
    s_push(&l, s_at(&p, 0));
    s_push(&l, s_at(&p, 1));
    for (i = 2; i < s_size(&p); i++) {
        s_push(&l, s_at(&p, i));
        while (s_size(&l) > 2 &&
               !v2_rturn((vector2 *) s_at(&l, -3), (vector2 *) s_at(&l, -2), (vector2 *) s_at(&l, -1))) {
            s_delete(&l, -2);
        }
    }
    s_delete(&l, 0);
    s_delete(&l, -1);
    s_copy(&u, to);
    s_append(&l, to);
}

/* EOF */
```

15 hull2d.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
/* 2d convex hull */

#include "stack.h"

void hull2d(stack *ps, stack *to);
```

```
/* EOF */
```

16 hypercube.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006, 2007 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
/* test rendering */

#include <stdlib.h>
#include <sys/poll.h>

#ifdef HAVE_LIBLO
#include <lo/lo.h>
#endif

#include <SDL.h>

#include "cube4d.h"
#include "dynamics.h"
#include "fixed.h"
#include "geometry.h"
#include "palette.h"
#include "r2sdl.h"
#include "raster.h"
#include "render.h"
#include "sdl2png.h"
#include "stack.h"

int d1 = 512;
int d2 = 1024;
int a[4][4] = {
    { 0, 0, 0, 0 },
    { 0, 0, 0, 0 },
    { 0, 0, 0, 0 },
    { 0, 0, 0, 0 }
};

int running = 1;

#ifdef HAVE_LIBLO
```

```
void osc_error_handler(int n, const char *msg, const char *path) {
    printf("liblo server error %d in path %s: %s\n", n, path, msg);
}

int osc_quit_handler(
    const char *path, const char *types,
    lo_arg **argv, int argc, void *data, void *user_data
) {
    running = 0;
    return (0);
}

int osc_rotate_inc_handler(
    const char *path, const char *types,
    lo_arg **argv, int argc, void *data, void *user_data
) {
    int dim1 = argv[0]->i;
    int dim2 = argv[1]->i;
    if (
        0 <= dim1 && dim1 < 4 &&
        0 <= dim2 && dim2 < 4 &&
        dim1 < dim2
    ) {
        a[dim1][dim2]++;
    }
    return (0);
}

int osc_rotate_dec_handler(
    const char *path, const char *types,
    lo_arg **argv, int argc, void *data, void *user_data
) {
    int dim1 = argv[0]->i;
    int dim2 = argv[1]->i;
    if (
        0 <= dim1 && dim1 < 4 &&
        0 <= dim2 && dim2 < 4 &&
        dim1 < dim2
    ) {
        a[dim1][dim2]--;
    }
    return (0);
}

int osc_rotate_zero_handler(
    const char *path, const char *types,
    lo_arg **argv, int argc, void *data, void *user_data
) {
    int dim1 = argv[0]->i;
    int dim2 = argv[1]->i;
    if (
        0 <= dim1 && dim1 < 4 &&
        0 <= dim2 && dim2 < 4 &&
        dim1 < dim2
    ) {
        a[dim1][dim2]=0;
    }
```

```
110         return (0);
    }

    void osc_send_v2(lo_address sender, c4d2dv *v2) {
        int i;
115         for (i = 0; i < 16; i++) {
            lo_send(sender, "/hypercube/vertex2d", "iff",
                i, (float) (*v2)[i][0], (float) (*v2)[i][1]
            );
        }
120 }
    #endif

    int main(int argc, char **argv) {

125         int w, h, t;
        raster *rb = NULL;
        raster *tb = NULL;
        rgb256 p;
        c4dv v4;
130         c4d2dv v2;
        int paused = 0;
        int save = 0;
        SDL_Event event;
        SDL_Surface *surface = NULL;
135
    #ifdef HAVE_LIBLO
        int oscfd;
        char *oscrport;
        lo_server oscserver;
140         struct pollfd pollfds[1];

        char *oscsport;
        lo_address sender;
    #endif
145
        char filename[64];
        char *fileformat = "hypercube-%08d.png";
        int filecount = 0;
        int savecount = 512;
150
        if (argc != 8) exit(1);

        w = atoi(argv[1]);
        h = atoi(argv[2]);
155         d1 = atoi(argv[3]);
        d2 = atoi(argv[4]);
        savecount = atoi(argv[5]);

    #ifdef HAVE_LIBLO
160         oscrport = argv[6];
        oscsport = argv[7];
    #endif

        w = w < 64 ? 64 : w;
165         h = h < 64 ? 64 : h;
        d1 = d1 < 512 ? 512 : d1;
```

```
    d2 = d2 < 512 ? 512 : d2;
    savecount = savecount < 1 ? 1 : savecount;

#ifdef HAVE_LIBLO
    oscserver = lo_server_new(oscrport, osc_error_handler);
    lo_server_add_method(oscserver, "/hypercube/rotate/inc", "ii", ↙
        ↳ osc_rotate_inc_handler, NULL);
    lo_server_add_method(oscserver, "/hypercube/rotate/dec", "ii", ↙
        ↳ osc_rotate_dec_handler, NULL);
    lo_server_add_method(oscserver, "/hypercube/rotate/zero", "ii", ↙
        ↳ osc_rotate_zero_handler, NULL);
    lo_server_add_method(oscserver, "/hypercube/quit", "", osc_quit_handler, ↙
        ↳ NULL);
    oscfd = lo_server_get_socket_fd(oscserver);
    sender = lo_address_new(NULL, oscsport);
#endif

    if ((surface = r2sdl_begin(w, h))) {
        SDL_WM_SetCaption("Hypercube", "Hypercube");
        f_init();
        cube4d_init();
        d_init();
        rb = r_alloc(w, h);
        tb = r_alloc(w, h);

        t = 0;
        while (running) {
            if (!paused) {
                r_clear(rb, 0);
                p_planar8bit(&p, t);
                d_transform(
                    &cube4d_vertex, &v4,
                    a[0][1], a[0][2], a[0][3], a[1][2], a[1][3], a[2][3]
                );
                n_project(&v4, d1, d2, &v2);
                n_scale2screen(&v2, w, h);
                n_layered(rb, tb, &v2);
                n_wireframe(rb, &v2, 0);
                r2sdl_setpalette(&p);
                r2sdl_display(rb);
                t++;
#ifdef HAVE_LIBLO
                osc_send_v2(sender, &v2);
#endif
            }
            if (save) {
                snprintf(filename, 60, fileformat, filecount++);
                sdl2png(filename, surface);
                save--;
            }
            while (SDL_PollEvent(&event)) {
                switch (event.type) {
                case SDL_QUIT:
                    running = 0;
                    break;
                case SDL_KEYDOWN:
                    switch (event.key.keysym.sym) {
```

```
                        case SDLK_ESCAPE: running = 0; break;
                        case SDLK_SPACE: paused = !paused; break;
                        case SDLK_PRINT: save = savecount; break;
                        case SDLK_q: a[0][1]++; break;
                        case SDLK_w: a[0][2]++; break;
                        case SDLK_e: a[0][3]++; break;
                        case SDLK_r: a[1][2]++; break;
                        case SDLK_t: a[1][3]++; break;
                        case SDLK_y: a[2][3]++; break;
                        case SDLK_a: a[0][1]=0; break;
                        case SDLK_s: a[0][2]=0; break;
                        case SDLK_d: a[0][3]=0; break;
                        case SDLK_f: a[1][2]=0; break;
                        case SDLK_g: a[1][3]=0; break;
                        case SDLK_h: a[2][3]=0; break;
                        case SDLK_z: a[0][1]--; break;
                        case SDLK_x: a[0][2]--; break;
                        case SDLK_c: a[0][3]--; break;
                        case SDLK_v: a[1][2]--; break;
                        case SDLK_b: a[1][3]--; break;
                        case SDLK_n: a[2][3]--; break;
                        default: break;
                        }
                        break;
                    default:
                        break;
                    }
                }
#ifdef HAVE_LIBLO
                pollfds[0].fd = oscfd;
                pollfds[0].events = POLLIN | POLLPRI;
                if (poll(pollfds, 1, 0) > 0) {
                    lo_server_recv_noblock(oscserver, 0);
                }
#endif
                SDL_Delay(40);
            }

            r_free(tb);
            r_free(rb);
            r2sdl_end();
        }

        return (0);
}

/* EOF */
```

17 hypercube-controller-help.pd

```
#N canvas 372 63 270 80 10;
#X obj 7 8 hypercube-controller;
```

18 hypercube-controller.pd

```
#N canvas 649 329 608 359 10;
#X obj 181 331 sendOSC;
```

```
#X msg 479 236 disconnect;
#X obj 477 137 select 1 0;
#X obj 11 26 bng 15 250 50 0 \$0-inc--s \$0-inc--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 11 42 bng 15 250 50 0 \$0-zero--s \$0-zero--r empty 0 -6 0 8
-260818 -258699 -1;
#X obj 11 58 bng 15 250 50 0 \$0-dec--s \$0-dec--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 27 26 bng 15 250 50 0 \$0-inc--s \$0-inc--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 27 42 bng 15 250 50 0 \$0-zero--s \$0-zero--r empty 0 -6 0 8
-260818 -258699 -1;
#X obj 27 58 bng 15 250 50 0 \$0-dec--s \$0-dec--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 43 26 bng 15 250 50 0 \$0-inc--s \$0-inc--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 43 42 bng 15 250 50 0 \$0-zero--s \$0-zero--r empty 0 -6 0 8
-260818 -258699 -1;
#X obj 43 58 bng 15 250 50 0 \$0-dec--s \$0-dec--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 59 26 bng 15 250 50 0 \$0-inc--s \$0-inc--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 59 42 bng 15 250 50 0 \$0-zero--s \$0-zero--r empty 0 -6 0 8
-260818 -258699 -1;
#X obj 59 58 bng 15 250 50 0 \$0-dec--s \$0-dec--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 75 26 bng 15 250 50 0 \$0-inc--s \$0-inc--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 75 42 bng 15 250 50 0 \$0-zero--s \$0-zero--r empty 0 -6 0 8
-260818 -258699 -1;
#X obj 75 58 bng 15 250 50 0 \$0-dec--s \$0-dec--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 91 26 bng 15 250 50 0 \$0-inc--s \$0-inc--r empty 0 -6 0 8 -260818
-258699 -1;
#X obj 91 42 bng 15 250 50 0 \$0-zero--s \$0-zero--r empty 0 -6 0 8
-260818 -258699 -1;
#X obj 91 58 bng 15 250 50 0 \$0-dec--s \$0-dec--r empty 0 -6 0 8 -260818
-258699 -1;
#X msg 423 89 set \$1;
#X floatatom 210 42 5 0 0 0 port #0-port-r #0-port-s;
#X symbolatom 116 59 21 0 0 2 host #0-host-r #0-host-s;
#X obj 250 42 tgl 15 0 \$0-connect-s \$0-connect-r connect -44 -6 0
12 -24198 -90133 -1 0 1;
#X obj 476 171 pack f s f;
#X obj 313 9 loadbang;
#X msg 313 38 symbol localhost;
#X msg 438 36 7777;
#X obj 523 111 r \$0-port-s;
#X obj 437 59 s \$0-port-r;
#X obj 313 60 s \$0-host-r;
#X obj 490 89 r \$0-host-s;
#X msg 11 139 send /hypercube/rotate/zero 0 1;
#X msg 11 119 send /hypercube/rotate/inc 0 1;
#X msg 11 159 send /hypercube/rotate/dec 0 1;
#X msg 11 189 send /hypercube/rotate/inc 0 2;
#X msg 11 209 send /hypercube/rotate/zero 0 2;
#X msg 11 229 send /hypercube/rotate/dec 0 2;
```

```
#X msg 11 259 send /hypercube/rotate/inc 1 2;
#X msg 11 279 send /hypercube/rotate/zero 1 2;
#X msg 11 299 send /hypercube/rotate/dec 1 2;
#X msg 247 119 send /hypercube/rotate/inc 0 3;
#X msg 247 139 send /hypercube/rotate/zero 0 3;
#X msg 247 159 send /hypercube/rotate/dec 0 3;
#X msg 247 189 send /hypercube/rotate/inc 1 3;
#X msg 247 209 send /hypercube/rotate/zero 1 3;
#X msg 248 231 send /hypercube/rotate/dec 1 3;
#X msg 247 259 send /hypercube/rotate/inc 2 3;
#X msg 247 279 send /hypercube/rotate/zero 2 3;
#X msg 247 299 send /hypercube/rotate/dec 2 3;
#X obj 255 11 bng 10 250 50 0 \$0-quit-s \$0-quit-r quit -38 9 0 12
-258699 -250685 -1;
#X msg 477 209 connect \$2 \$3;
#X msg 246 92 send /hypercube/quit;
#X connect 0 0 21 0;
#X connect 1 0 0 0;
#X connect 2 0 25 0;
#X connect 2 1 1 0;
#X connect 3 0 34 0;
#X connect 4 0 33 0;
#X connect 5 0 35 0;
#X connect 6 0 38 0;
#X connect 7 0 37 0;
#X connect 8 0 36 0;
#X connect 9 0 39 0;
#X connect 10 0 40 0;
#X connect 11 0 41 0;
#X connect 12 0 42 0;
#X connect 13 0 43 0;
#X connect 14 0 44 0;
#X connect 15 0 45 0;
#X connect 16 0 46 0;
#X connect 17 0 47 0;
#X connect 18 0 48 0;
#X connect 19 0 49 0;
#X connect 20 0 50 0;
#X connect 21 0 24 0;
#X connect 24 0 2 0;
#X connect 25 0 52 0;
#X connect 26 0 27 0;
#X connect 26 0 28 0;
#X connect 27 0 31 0;
#X connect 28 0 30 0;
#X connect 29 0 25 2;
#X connect 32 0 25 1;
#X connect 33 0 0 0;
#X connect 34 0 0 0;
#X connect 35 0 0 0;
#X connect 36 0 0 0;
#X connect 37 0 0 0;
#X connect 38 0 0 0;
#X connect 39 0 0 0;
#X connect 40 0 0 0;
#X connect 41 0 0 0;
#X connect 42 0 0 0;
```

```
#X connect 43 0 0 0;
#X connect 44 0 0 0;
#X connect 45 0 0 0;
#X connect 46 0 0 0;
#X connect 47 0 0 0;
#X connect 48 0 0 0;
#X connect 49 0 0 0;
#X connect 50 0 0 0;
#X connect 51 0 53 0;
#X connect 52 0 0 0;
#X connect 53 0 0 0;
#X coords 0 -1 1 1 256 64 1 10 10;
```

19 hypercube-receiver~.pd

```
#N canvas 879 2 362 443 10;
#X obj 53 7 dumpOSC 8888;
#X obj 53 28 OSCroute /hypercube;
#X obj 53 54 OSCroute /vertex2d;
#X obj 53 77 route 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
#X obj 142 264 dac~;
#X obj 110 221 expr~ $v1/2.1 \; $v2/2.1;
#X obj 9 123 fm-osc~;
#X obj 120 377 writesf~ 2;
#X obj 20 284 select 1 0;
#X msg 53 311 stop;
#X msg 22 343 open hypercube.wav \, start;
#X obj 18 261 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X msg 270 54 4 \$1-scale;
#X obj 270 32 float \$0;
#X obj 268 8 loadbang;
#X obj 245 340 s \$0-scale;
#X obj 243 373 table \$0-scale 4;
#X obj 240 9 bng 15 250 50 0 empty empty empty 0 -6 0 8 -262144 -1
-1;
#X msg 245 305 0 0 4 7 10;
#X obj 109 123 fm-osc~;
#X obj 8 146 fm-osc~;
#X obj 8 168 fm-osc~;
#X obj 108 146 fm-osc~;
#X obj 108 166 fm-osc~;
#X obj 8 188 fm-osc~;
#X obj 108 188 fm-osc~;
#X obj 199 123 fm-osc~;
#X obj 198 146 fm-osc~;
#X obj 198 168 fm-osc~;
#X obj 299 123 fm-osc~;
#X obj 298 146 fm-osc~;
#X obj 298 166 fm-osc~;
#X obj 198 188 fm-osc~;
#X obj 298 188 fm-osc~;
#X obj 245 276 loadbang;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 6 1;
```

```
#X connect 3 1 20 1;
#X connect 3 2 21 1;
#X connect 3 3 24 1;
#X connect 3 4 19 1;
#X connect 3 5 22 1;
#X connect 3 6 23 1;
#X connect 3 7 25 1;
#X connect 3 8 26 1;
#X connect 3 9 27 1;
#X connect 3 10 28 1;
#X connect 3 11 32 1;
#X connect 3 12 29 1;
#X connect 3 13 30 1;
#X connect 3 14 31 1;
#X connect 3 15 33 1;
#X connect 5 0 4 0;
#X connect 5 0 7 0;
#X connect 5 1 4 1;
#X connect 5 1 7 1;
#X connect 6 0 20 0;
#X connect 8 0 10 0;
#X connect 8 1 9 0;
#X connect 9 0 7 0;
#X connect 10 0 7 0;
#X connect 11 0 8 0;
#X connect 12 0 6 2;
#X connect 12 0 19 2;
#X connect 12 0 20 2;
#X connect 12 0 21 2;
#X connect 12 0 22 2;
#X connect 12 0 23 2;
#X connect 12 0 24 2;
#X connect 12 0 25 2;
#X connect 12 0 26 2;
#X connect 12 0 27 2;
#X connect 12 0 28 2;
#X connect 12 0 29 2;
#X connect 12 0 30 2;
#X connect 12 0 31 2;
#X connect 12 0 32 2;
#X connect 12 0 33 2;
#X connect 13 0 12 0;
#X connect 14 0 13 0;
#X connect 17 0 13 0;
#X connect 18 0 15 0;
#X connect 19 0 22 0;
#X connect 20 0 21 0;
#X connect 21 0 24 0;
#X connect 22 0 23 0;
#X connect 23 0 25 0;
#X connect 24 0 5 0;
#X connect 25 0 5 0;
#X connect 26 0 27 0;
#X connect 27 0 28 0;
#X connect 28 0 32 0;
#X connect 29 0 30 0;
#X connect 30 0 31 0;
```

```
#X connect 31 0 33 0;
#X connect 32 0 5 1;
#X connect 33 0 5 1;
#X connect 34 0 18 0;
```

20 Makefile

```
# hypercube -- interaction in four dimensions.
# Copyright (C) 2006, 2007 Claude Heiland-Allen
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
#
# Makefile

OPTFLAGS = -O3
FEATURES = -DHAVE_LIBLO
# FEATURES =

CC = gcc
CFLAGS = $(OPTFLAGS) $(FEATURES) -Wall `sdl-config --cflags` `libpng-config --cflags` `pkg-config liblo --cflags`
LIBS = `sdl-config --libs` `libpng-config --libs` `pkg-config liblo --libs` -lm
RM = rm -f

OFILES = cube4d.o dynamics.o fixed.o geometry.o hull2d.o palette.o \
r2sdl.o raster.o render.o sdl2png.o stack.o

#

all: hypercube

clean:
	$(RM) $(OFILES)

dist-clean:
	$(RM) hypercube

#

cube4d.o: cube4d.c cube4d.h fixed.h geometry.h stack.h
	$(CC) $(CFLAGS) -o cube4d.o -c cube4d.c

dynamics.o: dynamics.c dynamics.h cube4d.h fixed.h geometry.h
	$(CC) $(CFLAGS) -o dynamics.o -c dynamics.c
```

```
fixed.o: fixed.c fixed.h
	$(CC) $(CFLAGS) -o fixed.o -c fixed.c

geometry.o: geometry.c geometry.h fixed.h
	$(CC) $(CFLAGS) -o geometry.o -c geometry.c

hull2d.o: hull2d.c hull2d.h fixed.h geometry.h stack.h
	$(CC) $(CFLAGS) -o hull2d.o -c hull2d.c

palette.o: palette.c palette.h fixed.h
	$(CC) $(CFLAGS) -o palette.o -c palette.c

r2sdl.o: r2sdl.c r2sdl.h palette.h raster.h
	$(CC) $(CFLAGS) -o r2sdl.o -c r2sdl.c

raster.o: raster.c raster.h stack.h
	$(CC) $(CFLAGS) -o raster.o -c raster.c

render.o: render.c render.h cube4d.h fixed.h geometry.h hull2d.h \
raster.h stack.h
	$(CC) $(CFLAGS) -o render.o -c render.c

sdl2png.o: sdl2png.c sdl2png.h
	$(CC) $(CFLAGS) -o sdl2png.o -c sdl2png.c

stack.o: stack.c stack.h
	$(CC) $(CFLAGS) -o stack.o -c stack.c

#

hypercube: hypercube.c cube4d.h dynamics.h fixed.h geometry.h \
palette.h r2sdl.h raster.h render.h stack.h cube4d.o dynamics.o fixed.o \
geometry.o hull2d.o palette.o r2sdl.o raster.o render.o sdl2png.o stack.o
	$(CC) $(CFLAGS) -o hypercube hypercube.c cube4d.o dynamics.o \
fixed.o geometry.o hull2d.o palette.o r2sdl.o raster.o render.o sdl2png.o \
stack.o $(LIBS)

# EOF
```

21 palette.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
```

```
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    /* funky colour palettes */

    #include "fixed.h"
    #include "palette.h"

    void rgb_zero(rgb *to) {
        (*to)[0] = f_itof(0);
        (*to)[1] = f_itof(0);
        (*to)[2] = f_itof(0);
    }

    void rgb_one(rgb *to) {
        (*to)[0] = f_itof(1);
        (*to)[1] = f_itof(1);
        (*to)[2] = f_itof(1);
    }

    void rgb_copy(rgb *to, rgb *from) {
        (*to)[0] = (*from)[0];
        (*to)[1] = (*from)[1];
        (*to)[2] = (*from)[2];
    }

    void rgb_add(rgb *to, rgb *from) {
        (*to)[0] += (*from)[0];
        (*to)[1] += (*from)[1];
        (*to)[2] += (*from)[2];
    }

    void rgb_mul(rgb *to, rgb *from) {
        (*to)[0] = f_mul((*to)[0], (*from)[0]);
        (*to)[1] = f_mul((*to)[1], (*from)[1]);
        (*to)[2] = f_mul((*to)[2], (*from)[2]);
    }

    void rgb_sadd(rgb *to, fixed f) {
        (*to)[0] = f_add((*to)[0], f);
        (*to)[1] = f_add((*to)[1], f);
        (*to)[2] = f_add((*to)[2], f);
    }

    void rgb_smul(rgb *to, fixed f) {
        (*to)[0] = f_mul((*to)[0], f);
        (*to)[1] = f_mul((*to)[1], f);
        (*to)[2] = f_mul((*to)[2], f);
    }

    void rgb_sdiv(rgb *to, fixed f) {
        (*to)[0] = f_div((*to)[0], f);
        (*to)[1] = f_div((*to)[1], f);
        (*to)[2] = f_div((*to)[2], f);
    }

    void rgb_normalize(rgb *to) {
```

```
    fixed max;
    int i;
    max = 0;
    for (i = 0; i < 3; i++) {
        if ((*to)[i] > max) {
            max = (*to)[i];
        }
    }
    if (max) {
        rgb_sdiv(to, max + 1);
    }
}

void p_planar8bit(rgb256 *p, int a) {
    int c, b, n;
    rgb core[8];
    rgb colour;
    for (c = 0; c < 8; c++) {
        core[c][0] = (128 + f_sin((c << 6) + a +   0));
        core[c][1] = (128 + f_sin((c << 6) + a + 171));
        core[c][2] = (128 + f_sin((c << 6) + a + 341));
    }
    for (c = 0; c < 256; c++) {
        rgb_zero(&colour);
        n = 0;
        for (b = 0; b < 8; b++) {
            if (c & (1<<b)) {
                rgb_add(&colour, &(core[b]));
                n++;
            }
        }
        if (n) {
            rgb_sdiv(&colour, f_itof(n));
        }
        rgb_copy(&((*p)[c]), &colour);
    }
}

/* EOF */
```

22 palette.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
```

```
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef PALETTE_H
#define PALETTE_H 1
/* funky colour palettes */

#include "fixed.h"

typedef fixed rgb[3];
typedef rgb rgb256[256];

#define rgb_fto64(x) (((x) >> 1) & 63)

void rgb_zero(rgb *to);
void rgb_copy(rgb *to, rgb *from);
void rgb_add(rgb *to, rgb *from);
void rgb_mul(rgb *to, rgb *from);
void rgb_smul(rgb *to, fixed f);
void rgb_sdiv(rgb *to, fixed f);
void rgb_normalize(rgb *to);

void p_planar8bit(rgb256 *p, int a);
void p_set(rgb256 *p);

/* EOF */
#endif
```

23 patches/have-liblo.patch

```
Index: Makefile
===================================================================
--- Makefile    (revision 194)
+++ Makefile    (working copy)
@@ -17,7 +17,7 @@
 #
 # Makefile

-OPTFLAGS = -O3
+OPTFLAGS = -O3 -DHAVE_LIBLO

 CC = gcc
 CFLAGS = $(OPTFLAGS) -Wall `sdl-config --cflags` `libpng-config --cflags` `pkg-
    ↪ config liblo --cflags`
Index: hypercube.c
===================================================================
--- hypercube.c (revision 194)
+++ hypercube.c (working copy)
@@ -21,7 +21,9 @@ Foundation, Inc., 51 Franklin Street, Fi
 #include <stdlib.h>
 #include <sys/poll.h>

+#ifdef HAVE_LIBLO
 #include <lo/lo.h>
+#endif

 #include <SDL.h>
```

```
@@ -48,6 +50,7 @@ int a[4][4] = {
 int running = 1;

+#ifdef HAVE_LIBLO
 void osc_error_handler(int n, const char *msg, const char *path) {
     printf("liblo server error %d in path %s: %s\n", n, path, msg);
 }
@@ -117,6 +120,7 @@ void osc_send_v2(lo_address sender, c4d2
         );
     }
 }
+#endif

 int main(int argc, char **argv) {

@@ -131,6 +135,7 @@ int main(int argc, char **argv) {
     SDL_Event event;
     SDL_Surface *surface = NULL;

+#ifdef HAVE_LIBLO
     int oscfd;
     char *oscrport;
     lo_server oscserver;
@@ -138,6 +143,7 @@ int main(int argc, char **argv) {

     char *oscsport;
     lo_address sender;
+#endif

     char filename[64];
     char *fileformat = "hypercube-%08d.png";
@@ -151,9 +157,11 @@ int main(int argc, char **argv) {
     d1 = atoi(argv[3]);
     d2 = atoi(argv[4]);
     savecount = atoi(argv[5]);
-    oscrport = argv[6];

+#ifdef HAVE_LIBLO
+    oscrport = argv[6];
     oscsport = argv[7];
+#endif

     w = w < 64 ? 64 : w;
     h = h < 64 ? 64 : h;
@@ -161,6 +169,7 @@ int main(int argc, char **argv) {
     d2 = d2 < 512 ? 512 : d2;
     savecount = savecount < 1 ? 1 : savecount;

+#ifdef HAVE_LIBLO
     oscserver = lo_server_new(oscrport, osc_error_handler);
     lo_server_add_method(oscserver, "/hypercube/rotate/inc", "ii", ↙
         ↳ osc_rotate_inc_handler, NULL);
     lo_server_add_method(oscserver, "/hypercube/rotate/dec", "ii", ↙
         ↳ osc_rotate_dec_handler, NULL);
@@ -169,6 +178,7 @@ int main(int argc, char **argv) {
```

```
      oscfd = lo_server_get_socket_fd(oscserver);

      sender = lo_address_new(NULL, oscsport);
+#endif

      if ((surface = r2sdl_begin(w, h))) {
          SDL_WM_SetCaption("Hypercube", "Hypercube");
@@ -194,7 +204,9 @@ int main(int argc, char **argv) {
                  r2sdl_setpalette(&p);
                  r2sdl_display(rb);
                  t++;
-                 osc_send_v2(sender, &v2);
+#ifdef HAVE_LIBLO
+                                 osc_send_v2(sender, &v2);
+#endif
              }
              if (save) {
                  snprintf(filename, 60, fileformat, filecount++);
@@ -236,11 +248,13 @@ int main(int argc, char **argv) {
                      break;
                  }
              }
+#ifdef HAVE_LIBLO
              pollfds[0].fd = oscfd;
              pollfds[0].events = POLLIN | POLLPRI;
              if (poll(pollfds, 1, 0) > 0) {
                  lo_server_recv_noblock(oscserver, 0);
              }
+#endif
              SDL_Delay(40);
          }
```

24 patches/r_bitor_bounded.patch

```
Index: raster.c
===================================================================
--- raster.c    (revision 194)
+++ raster.c    (working copy)
@@ -147,6 +176,24 @@ void r_bitor(raster *to, raster *from) {
      }
  }

+void r_bitor_bounded(raster *to, const raster *from, const fixed *bounds) {
+    int xstart;
+    int last;
+
+    xstart = bounds[0] + bounds[1] * to->w;
+    last = bounds[0] + bounds[3] * to->w;
+
+    for (; xstart <= last; xstart += to->w)
+    {
+        pixel *tx = to->data + xstart;
+        pixel *fx = from->data + xstart;
+        pixel *lastx = tx + (bounds[2] - bounds[0]);
+
+        for (; tx <= lastx;)
+                        *tx++ |= *fx++;
```

```
+    }
+}
+
 void r_bitand(raster *to, raster *from) {
     int i;
     int n = to->w * to->h;
Index: raster.h
===================================================================
--- raster.h    (revision 194)
+++ raster.h    (working copy)
@@ -39,9 +39,11 @@ pixel r_get(raster *rb, int x, int y);
 void r_hline(raster *rb, int x0, int y, int x1, pixel pen);
 void r_line(raster *rb, int x0, int y0, int x1, int y1, pixel pen);
 void r_poly(raster *rb, stack *p, pixel pen);
+void r_bounds(raster *rb, stack *p, fixed *bounds, int clear_bounds);
 void r_fillpoly(raster *rb, stack *p, pixel pen);
 void r_copy(raster *to, raster *from);
 void r_bitor(raster *to, raster *from);
+void r_bitor_bounded(raster *to, const raster *from, const fixed *bounds);
 void r_bitand(raster *to, raster *from);
 void r_bitxor(raster *to, raster *from);

Index: render.c
===================================================================
--- render.c    (revision 194)
+++ render.c    (working copy)
@@ -80,6 +80,8 @@ void n_wireframe(raster *rb, c4d2dv *v2,
 void n_layered(raster *rb, raster *tb, c4d2dv *v2) {
     int i, j;
     stack s, t;
+    fixed bounds[4];
+
     for (i = 0; i < 8; i++) {
         r_clear(tb, 0);
         s_clear(&s);
@@ -88,7 +90,8 @@ void n_layered(raster *rb, raster *tb, c
         }
         hull2d(&s, &t);
         r_fillpoly(tb, &t, 1 << i);
-        r_bitor(rb, tb);
+               r_bounds(rb, &t, bounds, 1);
+        r_bitor_bounded(rb, tb, bounds);
     }
 }
```

25 patches/r_bounds2.patch

```
Index: raster.c
===================================================================
--- raster.c    (revision 194)
+++ raster.c    (working copy)
@@ -106,20 +97,53 @@ void r_poly(raster *rb, stack *p, pixel
     }
 }

+#define MAX(a, b) \
+       ((a) > (b) ? (a) : (b))
```

```
+#define MIN(a, b) \
+       ((a) < (b) ? (a) : (b))
+
+// bounds should be a fixed[4]
+void r_bounds(raster *rb, stack *p, fixed *bounds, int clear_bounds) {
+    int i;
+    vector2 *tmp;
+
+    if (clear_bounds)
+    {
+       bounds[0] = rb->w - 1;
+       bounds[1] = rb->h - 1;
+       bounds[2] = 0;
+       bounds[3] = 0;
+    }
+    for (i = 0; i < s_size(p); i++) {
+        tmp = (vector2 *) s_at(p, i);
+        bounds[0] = MIN(bounds[0], (*tmp)[0]);
+        bounds[1] = MIN(bounds[1], (*tmp)[1]);
+        bounds[2] = MAX(bounds[2], (*tmp)[0]);
+        bounds[3] = MAX(bounds[3], (*tmp)[1]);
+    }
+       // ensure bounds are within rb limits
+       bounds[0] = MAX(bounds[0], 0);
+       bounds[1] = MAX(bounds[1], 0);
+       bounds[2] = MIN(bounds[2], rb->w - 1);
+       bounds[3] = MIN(bounds[3], rb->h - 1);
+}
+
 void r_fillpoly(raster *rb, stack *p, pixel pen) {
     int x, y, x0, x1;
+    fixed bounds[4];
     r_clear(rb, 0);
     r_poly(rb, p, 1);
-    for (y = 0; y < rb->h; y++) {
+    r_bounds(rb, p, bounds, 1);
+    for (y = bounds[1]; y <= bounds[3]; y++) {
         x0 = rb->w+1;
         x1 = -2;
-        for (x = 0; x < rb->w; x++) {
+        for (x = bounds[0]; x <= bounds[2]; x++) {
             if (r_get(rb, x, y)) {
                 x0 = x;
                 break;
             }
         }
-        for (x = rb->w - 1; x >= 0; x--) {
+        for (x = bounds[2]; x >= bounds[0]; x--) {
             if (r_get(rb, x, y)) {
                 x1 = x;
                 break;
```

26 patches/r_bounds.patch

```
Index: raster.c
===================================================================
--- raster.c    (revision 194)
```

```
+++ raster.c    (working copy)
@@ -106,20 +97,45 @@ void r_poly(raster *rb, stack *p, pixel
     }
 }

+#define MAX(a, b) \
+       ((a) > (b) ? (a) : (b))
+#define MIN(a, b) \
+       ((a) < (b) ? (a) : (b))
+
+// bounds should be a fixed[4]
+static void r_bounds(raster *rb, stack *p, fixed *bounds) {
+    int i;
+    vector2 *tmp;
+    memset(bounds, 0, 4);
+    for (i = 0; i < s_size(p); i++) {
+        tmp = (vector2 *) s_at(p, i);
+        bounds[0] = MIN(bounds[0], (*tmp)[0]);
+        bounds[1] = MIN(bounds[1], (*tmp)[1]);
+        bounds[2] = MAX(bounds[2], (*tmp)[0]);
+        bounds[3] = MAX(bounds[3], (*tmp)[1]);
+    }
+       bounds[0] = MAX(bounds[0], 0);
+       bounds[1] = MAX(bounds[1], 0);
+       bounds[2] = MIN(bounds[2], rb->w - 1);
+       bounds[3] = MIN(bounds[3], rb->h - 1);
+}
+
 void r_fillpoly(raster *rb, stack *p, pixel pen) {
     int x, y, x0, x1;
+    fixed bounds[4];
     r_clear(rb, 0);
     r_poly(rb, p, 1);
-    for (y = 0; y < rb->h; y++) {
+    r_bounds(rb, p, bounds);
+    for (y = bounds[1]; y <= bounds[3]; y++) {
         x0 = rb->w+1;
         x1 = -2;
-        for (x = 0; x < rb->w; x++) {
+        for (x = bounds[0]; x <= bounds[2]; x++) {
             if (r_get(rb, x, y)) {
                 x0 = x;
                 break;
             }
         }
-        for (x = rb->w - 1; x >= 0; x--) {
+        for (x = bounds[2]; x >= bounds[0]; x--) {
             if (r_get(rb, x, y)) {
                 x1 = x;
                 break;
```

27 patches/r_clear-memset.patch

```
Index: raster.c
===================================================================
--- raster.c    (revision 194)
+++ raster.c    (working copy)
```

```
@@ -19,6 +19,7 @@ Foundation, Inc., 51 Franklin Street, Fi
 /* raster drawing algorithms */

 #include <stdlib.h>
+#include <string.h>    // memset

 #include "geometry.h"
 #include "raster.h"
@@ -42,11 +43,8 @@ void r_free(raster *rb) {
 }

 void r_clear(raster *rb, pixel pen) {
-    int i;
     int n = rb->w * rb-> h;
-    for (i = 0; i < n; i++) {
-        rb->data[i] = pen;
-    }
+    memset(rb->data, (int)pen, n);
 }

 void r_plot(raster *rb, int x, int y, pixel pen) {
@@ -131,6 +129,7 @@ void r_fillpoly(raster *rb, stack *p, pi
     }
 }

+// Note: rewrite with memcpy for speed
 void r_copy(raster *to, raster *from) {
     int i;
     int n = to->w * to->h;
```

28 patches/README

Some patches from Nick Treleaven:

have-liblo.patch
    Enable building a version without OSC.

r_clear-memset.patch
    r_clear profiled about 30% total time with default arguments (from README) on my machine. Using memset that time is ~0, so about 30% faster.

r_bounds.patch
    Bounds checking patch for r_fillpoly makes it about 20% faster on my machine.

r_bounds2.patch
    There was a bug in the bounds checking patch that only minimized the lower coordinate. This update will be a further 14% faster.

r_bitor_bounded.patch
    Patch does a bounded bitor, is about 40% faster than previous patched hypercube. About 22% is due to bounding, rest due to using *ptr++.

29 r2sdl.c

```
/*
```

```
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#include <SDL.h>
#include "r2sdl.h"

static SDL_Surface *surface = NULL;

SDL_Surface *r2sdl_begin(int w, int h) {
    SDL_Init(SDL_INIT_VIDEO);
    surface = SDL_SetVideoMode(w, h, 8, SDL_SWSURFACE);
    return (surface);
}

void r2sdl_end(void) {
    SDL_Quit();
}

void r2sdl_setpalette(rgb256 *p) {
    int c;
    SDL_Color colors[256];
    for (c = 0; c < 256; c++) {
        colors[c].r = (*p)[c][0];
        colors[c].g = (*p)[c][1];
        colors[c].b = (*p)[c][2];
    }
    SDL_SetColors(surface, colors, 0, 256);
}

void r2sdl_display(raster *rb) {
    int x, y;
    Uint8 *ps = (Uint8 *) surface->pixels;
    Uint8 *psx;
    Uint8 *pr = rb->data;
    SDL_LockSurface(surface);
    for (y = 0; y < rb->h; y++) {
        psx = ps;
        for (x = 0; x < rb->w; x++) {
            *psx++ = *pr++;
        }
        ps += surface->pitch;
    }
    SDL_UnlockSurface(surface);
```

```
}

/* EOF */
```

30 r2sdl.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef R2SDL_H
#define R2SDL_H 1
/* interface to SDL display */

#include <SDL.h>

#include "palette.h"
#include "raster.h"

SDL_Surface *r2sdl_begin(int w, int h);
void r2sdl_end(void);
void r2sdl_setpalette(rgb256 *p);
void r2sdl_display(raster *rb);

/* EOF */
#endif
```

31 raster.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006, 2007 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
    */
    /* raster drawing algorithms */

    #include <stdlib.h>
    #include <string.h>

    #include "geometry.h"
    #include "raster.h"
    #include "stack.h"

    raster *r_alloc(int w, int h) {
        raster *rb;
        rb = (raster *) malloc(sizeof(raster) + w*h);
        if (rb) {
            rb->w = w;
            rb->h = h;
            rb->data = (pixel *) (((char *) rb) + sizeof(raster));
        }
        return (rb);
    }

    void r_free(raster *rb) {
        if (rb) {
            free (rb);
        }
    }

    void r_clear(raster *rb, pixel pen) {
        int n = rb->w * rb-> h;
        memset(rb->data, (int) pen, n);
    }

    void r_plot(raster *rb, int x, int y, pixel pen) {
        rb->data[y*rb->w+x] = pen;
    }

    pixel r_get(raster *rb, int x, int y) {
        return (rb->data[y*rb->w+x]);
    }

    void r_hline(raster *rb, int x0, int y, int x1, pixel pen) {
        int x;
        pixel *p = &(rb->data[y*rb->w+x0]);
        for (x = x0; x <= x1; x++) {
            *p++ = pen;
        }
    }

    void r_line(raster *rb, int x0, int y0, int x1, int y1, pixel pen) {
        short steep, t, dx, dy, e, de, x, y, ystep;
        steep = abs(y1 - y0) > abs(x1 - x0);
        if (steep) {
            t = x0; x0 = y0; y0 = t;
            t = x1; x1 = y1; y1 = t;
```

```
    }
    if (x0 > x1) {
        t = x0; x0 = x1; x1 = t;
        t = y0; y0 = y1; y1 = t;
    }
    dx = x1 - x0;
    dy = abs(y1 - y0);
    e = 0;
    de = dy;
    y = y0;
    ystep = (y0 < y1) ? 1 : -1;
    for (x = x0; x <= x1; x++) {
        if (steep) {
            r_plot(rb, y, x, pen);
        } else {
            r_plot(rb, x, y, pen);
        }
        e += de;
        if ((e<<1) >= dx) {
            y += ystep;
            e -= dx;
        }
    }
}

void r_poly(raster *rb, stack *p, pixel pen) {
    int i;
    vector2 *p0, *p1;
    for (i = 0; i < s_size(p); i++) {
        p0 = (vector2 *) s_at(p, i - 1);
        p1 = (vector2 *) s_at(p, i);
        r_line(rb, (*p0)[0], (*p0)[1], (*p1)[0], (*p1)[1], pen);
    }
}

#define MAX(a, b) ((a) > (b) ? (a) : (b))
#define MIN(a, b) ((a) < (b) ? (a) : (b))

/* bounds should be a fixed[4] */
void r_bounds(raster *rb, stack *p, fixed *bounds, int clear_bounds) {
    int i;
    vector2 *tmp;
    if (clear_bounds)
    {
        bounds[0] = rb->w - 1;
        bounds[1] = rb->h - 1;
        bounds[2] = 0;
        bounds[3] = 0;
    }
    for (i = 0; i < s_size(p); i++) {
        tmp = (vector2 *) s_at(p, i);
        bounds[0] = MIN(bounds[0], (*tmp)[0]);
        bounds[1] = MIN(bounds[1], (*tmp)[1]);
        bounds[2] = MAX(bounds[2], (*tmp)[0]);
        bounds[3] = MAX(bounds[3], (*tmp)[1]);
    }
    /* ensure bounds are within rb limits */
```

```
    bounds[0] = MAX(bounds[0], 0);
    bounds[1] = MAX(bounds[1], 0);
    bounds[2] = MIN(bounds[2], rb->w - 1);
    bounds[3] = MIN(bounds[3], rb->h - 1);
}

void r_fillpoly(raster *rb, stack *p, pixel pen) {
    int x, y, x0, x1;
    fixed bounds[4];
    r_clear(rb, 0);
    r_poly(rb, p, 1);
    r_bounds(rb, p, bounds, 1);
    for (y = bounds[1]; y <= bounds[3]; y++) {
        x0 = rb->w+1;
        x1 = -2;
        for (x = bounds[0]; x <= bounds[2]; x++) {
            if (r_get(rb, x, y)) {
                x0 = x;
                break;
            }
        }
        for (x = bounds[2]; x >= bounds[0]; x--) {
            if (r_get(rb, x, y)) {
                x1 = x;
                break;
            }
        }
        if (x1 >= x0) {
            r_hline(rb, x0, y, x1, pen);
        }
    }
}

void r_copy(raster *to, raster *from) {
    int n = to->w * to->h;
    memcpy(to->data, from->data, n);
}

void r_bitor(raster *to, raster *from) {
    int i;
    int n = to->w * to->h;
    for (i = 0; i < n; i++) {
        to->data[i] |= from->data[i];
    }
}

void r_bitor_bounded(raster *to, const raster *from, const fixed *bounds) {
    int xstart;
    int last;
    xstart = bounds[0] + bounds[1] * to->w;
    last = bounds[0] + bounds[3] * to->w;
    for (; xstart <= last; xstart += to->w) {
        pixel *tx = to->data + xstart;
        pixel *fx = from->data + xstart;
        pixel *lastx = tx + (bounds[2] - bounds[0]);
        for (; tx <= lastx;) {
            *tx++ |= *fx++;
```

```
        }
    }
}

void r_bitand(raster *to, raster *from) {
    int i;
    int n = to->w * to->h;
    for (i = 0; i < n; i++) {
        to->data[i] &= from->data[i];
    }
}

void r_bitxor(raster *to, raster *from) {
    int i;
    int n = to->w * to->h;
    for (i = 0; i < n; i++) {
        to->data[i] ^= from->data[i];
    }
}

/* EOF */
```

32 raster.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006, 2007 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef RASTER_H
#define RASTER_H 1
/* raster drawing algorithms */

#include "stack.h"

#define RASTER_BPP (1)

typedef unsigned char pixel;
typedef struct {
    int w;
    int h;
    pixel *data;
} raster;
```

```
raster *r_alloc(int w, int h);
void r_free(raster *rb);
void r_clear(raster *rb, pixel pen);
void r_plot(raster *rb, int x, int y, pixel pen);
pixel r_get(raster *rb, int x, int y);
void r_hline(raster *rb, int x0, int y, int x1, pixel pen);
void r_line(raster *rb, int x0, int y0, int x1, int y1, pixel pen);
void r_poly(raster *rb, stack *p, pixel pen);
void r_bounds(raster *rb, stack *p, fixed *bounds, int clear_bounds);
void r_fillpoly(raster *rb, stack *p, pixel pen);
void r_copy(raster *to, raster *from);
void r_bitor(raster *to, raster *from);
void r_bitor_bounded(raster *to, const raster *from, const fixed *bounds);
void r_bitand(raster *to, raster *from);
void r_bitxor(raster *to, raster *from);

/* EOF */
#endif
```

33 README

```
hypercube -- interaction in four dimensions.
Copyright (C) 2006,2017 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.

version:   0.3.13
requires:  libsdl, libpng, liblo
compile:   make
usage:     ./hypercube <width> <height> <depth1> <depth2> <savecount> <↙
    ↳ listenport> <sendport>
example:   ./hypercube 640 480 512 1024 1 7777 8888
keys:
 ESC          -- exit (or use window close button)
 SPACE        -- pause/unpause
 PRINTSCREEN -- save frame to numbered PNG file(*)
 q,w,e,r,t,y -- increase rotation speeds
 a,s,d,f,g,h -- zero rotation speeds
 z,x,c,v,b,n -- decrease rotation speeds
open sound control:
 /hypercube/quit -- quit
 /hypercube/rotate/inc  dim1:f dim2:f -- increment rotation speeds
 /hypercube/rotate/zero dim1:f dim2:f -- stop rotations
 /hypercube/rotate/dec  dim1:f dim2:f -- decrement rotation speeds
 /hypercube/vertex2d n:i x:f y:f -- raw vertex coordinates
```

```
 (see included Pd files for examples of control and sonification)

(*) each time you press PRINTSCREEN savecount images will be
saved in the current directory in PNG format, with filenames
like hypercube-00000000.png, hypercube-00000001.png, etc.
Note that the counter restarts from 0000000 each time you
restart hypercube, so you might want to move the previously
saved images out of the way to avoid overwriting them.
These frames can be converted to animated gif using:
find *.png -exec convert {} {}.gif \;
gifsicle --colors 256 --delay 4 --optimize --loopcount forever \
*.png.gif > hypercube.gif
```

34 render.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006, 2007 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
/* render a hypercube in different ways */

#include <limits.h>

#include "cube4d.h"
#include "fixed.h"
#include "geometry.h"
#include "hull2d.h"
#include "raster.h"
#include "render.h"
#include "stack.h"

/* project 4D to 2D */
void n_project(c4dv *from, fixed d43, fixed d32, c4d2dv *to) {
    int i;
    vector3 v3;
    for (i = 0; i < 16; i++) {
        v4_project(&((*from)[i]), d43, &v3);
        v3_project(&v3, d32, &((*to)[i]));
    }
}

/* scale 2D to fit screen */
void n_scale2screen(c4d2dv *v2, int w, int h) {
```

```
    int i;
    static int min[2] = { INT_MAX, INT_MAX };
    static int max[2] = { INT_MIN, INT_MIN };
    for (i = 0; i < 16; i++) {
        min[0] = (*v2)[i][0] < min[0] ? (*v2)[i][0] : min[0];
        min[1] = (*v2)[i][1] < min[1] ? (*v2)[i][1] : min[1];
        max[0] = (*v2)[i][0] > max[0] ? (*v2)[i][0] : max[0];
        max[1] = (*v2)[i][1] > max[1] ? (*v2)[i][1] : max[1];
    }
    for (i = 0; i < 16; i++) {
        (*v2)[i][0] -= (max[1] + min[0]) >> 1;
        (*v2)[i][1] -= (max[1] + min[1]) >> 1;
        (*v2)[i][0] *= (w>>4);
        (*v2)[i][1] *= (h>>4);
        (*v2)[i][0] /= ((max[0] - min[0])>>4)+16;
        (*v2)[i][1] /= ((max[1] - min[1])>>4)+16;
        (*v2)[i][0] += (w>>1);
        (*v2)[i][1] += (h>>1);
    }
}

/* render edges */
void n_wireframe(raster *rb, c4d2dv *v2, int pen) {
    int i;
    for (i = 0; i < 32; i = i + 1) {
        r_line(
            rb,
            (*v2)[cube4d_trail[i  ]][0],
            (*v2)[cube4d_trail[i  ]][1],
            (*v2)[cube4d_trail[i+1]][0],
            (*v2)[cube4d_trail[i+1]][1],
            pen
        );
    }
}

/* render sub-cubes as layered polygons */
void n_layered(raster *rb, raster *tb, c4d2dv *v2) {
    int i, j;
    stack s, t;
    fixed bounds[4];
    for (i = 0; i < 8; i++) {
        r_clear(tb, 0);
        s_clear(&s);
        for (j = 0; j < 8; j++) {
            s_push(&s, &((*v2)[cube4d_cube[i][j]]));
        }
        hull2d(&s, &t);
        r_fillpoly(tb, &t, 1 << i);
        r_bounds(rb, &t, bounds, 1);
        r_bitor_bounded(rb, tb, bounds);
    }
}

/* EOF */
```

35 render.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef RENDER_H
#define RENDER_H 1
/* render a hypercube in different ways */

#include "cube4d.h"
#include "fixed.h"
#include "geometry.h"
#include "raster.h"

typedef vector2 c4d2dv[16];

/* project 4d to 2d */
void n_project(c4dv *from, fixed d43, fixed d32, c4d2dv *to);

/* scale 2d to fit screen */
void n_scale2screen(c4d2dv *v2, int w, int h);

/* render edges */
void n_wireframe(raster *rb, c4d2dv *v2, int pen);

/* render sub-cubes as layered polygons */
void n_layered(raster *rb, raster *tb, c4d2dv *v2);

/* EOF */
#endif
```

36 scalify.pd

```
#N canvas 490 378 226 258 10;
#X obj 17 12 inlet;
#X obj 105 12 inlet;
#X obj 17 158 tabread;
#X obj 105 42 unpack f s;
#X msg 170 68 set \$1;
#X obj 17 89 mod;
#X obj 47 89 div;
#X obj 17 61 t f f;
```

```
#X obj 47 111 * 12;
#X obj 17 186 +;
#X obj 17 222 outlet;
#X obj 17 35 int;
#X connect 0 0 11 0;
#X connect 1 0 3 0;
#X connect 2 0 9 0;
#X connect 3 0 5 1;
#X connect 3 0 6 1;
#X connect 3 1 4 0;
#X connect 4 0 2 0;
#X connect 5 0 2 0;
#X connect 6 0 8 0;
#X connect 7 0 5 0;
#X connect 7 1 6 0;
#X connect 8 0 9 1;
#X connect 9 0 10 0;
#X connect 11 0 7 0;
```

37 sdl2png.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006,2017 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#include <SDL.h>
#include <png.h>

#include "sdl2png.h"

/* WARNING: this code is not robust, it assumes indexed palette
mode with 256 entries, if this is not the case it will crash */

int sdl2png(char *filename, SDL_Surface *surface) {
    FILE *fp = NULL;
    png_struct *png_ptr = NULL;
    png_info *info_ptr = NULL;
    png_color palette[256];
    int c, pass, passes, y;

    if ((fp = fopen(filename, "wb")) == NULL) {
        return (1);
    }
```

```
    png_ptr = png_create_write_struct(
        PNG_LIBPNG_VER_STRING,
        (void *) NULL, NULL, NULL
    );
    if (png_ptr == NULL) {
        return (2);
    }

    info_ptr = png_create_info_struct(png_ptr);
    if (info_ptr == NULL) {
        png_destroy_write_struct(&png_ptr, (png_infopp) NULL);
        return(3);
    }

    if (setjmp(png_jmpbuf(png_ptr))) {
        png_destroy_write_struct(&png_ptr, &info_ptr);
        return (4);
    }

    png_init_io(png_ptr, fp);
    png_set_compression_level(png_ptr, 9);
    png_set_IHDR(
        png_ptr,
        info_ptr,
        surface->w,
        surface->h,
        8, /* bit depth */
        PNG_COLOR_TYPE_PALETTE,
        PNG_INTERLACE_ADAM7,
        PNG_COMPRESSION_TYPE_DEFAULT,
        PNG_FILTER_TYPE_DEFAULT
    );
    for (c = 0; c < 256; c++) {
        palette[c].red   = surface->format->palette->colors[c].r;
        palette[c].green = surface->format->palette->colors[c].g;
        palette[c].blue  = surface->format->palette->colors[c].b;
    }
    png_set_PLTE(png_ptr, info_ptr, palette, 256);
    png_write_info(png_ptr, info_ptr);
    passes = png_set_interlace_handling(png_ptr);
    SDL_LockSurface(surface);
    for (pass = 0; pass < passes; pass++) {
        for (y = 0; y < surface->h; y++) {
            png_write_row(
                png_ptr,
                ((png_bytep) (surface->pixels)) + surface->pitch * y
            );
        }
    }
    SDL_UnlockSurface(surface);
    png_write_end(png_ptr, info_ptr);
    png_destroy_write_struct(&png_ptr, &info_ptr);
    fclose(fp);
    return (0);
}

/* EOF */
```

38 sdl2png.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef SDL2PNG_H
#define SDL2PNG_H 1
/* save SDL surfaces to PNG files */

#include <SDL.h>

int sdl2png(char *filename, SDL_Surface *surface);

/* EOF */
#endif
```

39 stack.c

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
/* stack */

#include "stack.h"

stack_content s_at(stack *s, int n) {
    if (n < 0) {
```

```
25          n = s->head + 1 + n;
        }
        return (s->body[n]);
    }

30  void s_delete(stack *s, int n) {
        int i;
        if (n < 0) {
            n = s->head + 1 + n;
        }
35      for (i = n + 1; i < s->head + 1; i++) {
            s->body[i-1] = s->body[i];
        }
        (s->head)--;
    }
40
    void s_copy(stack *s, stack *d) {
        int i;
        d->head = s->head;
        for (i = 0; i < s->head + 1; i++) {
45          d->body[i] = s->body[i];
        }
    }

    void s_sort(stack *s, comparator c) {
50      stack_content v;
        int i;
        int j;
        for (i = 1; i <= s->head; i++) {
            v = s->body[i];
55          for (j = i-1; j >= 0 && c(s->body[j], v) > 0; j--) {
                s->body[j+1] = s->body[j];
            }
            s->body[j+1] = v;
        }
60  }

    void s_uniq(stack *s, comparator c) {
        int r, w;
        w = 0;
65      for (r = 1; r < s->head + 1; r++) {
            if (s->body[w] != s->body[r]) {
                w++;
                s->body[w] = s->body[r];
            }
70      }
        s->head = w;
    }

    void s_reverse(stack *s) {
75      int i,j;
        stack_content v;
        for (i = 0, j = s->head; i < j; i++, j--) {
            v = s->body[i];
            s->body[i] = s->body[j];
80          s->body[j] = v;
        }
```

```
}

void s_append(stack *s, stack *d) {
    int i;
    for (i = 0; i < s->head + 1; i++) {
        d->body[d->head + 1 + i] = s->body[i];
    }
    d->head += s->head + 1;
}

/* EOF */
```

40 stack.h

```
/*
hypercube -- interaction in four dimensions.
Copyright (C) 2006,2017 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/
#ifndef STACK_H
#define STACK_H 1
/* stack */

#include <stdint.h>

#define S_MAXSIZE (64)

typedef intptr_t stack_content;

typedef int (comparator)(stack_content, stack_content);

typedef struct {
    short head;
    stack_content body[S_MAXSIZE];
} stack;

#define s_clear(s) ((s)->head = -1)
#define s_size(s) ((s)->head + 1)
#define s_push(s,x) ((s)->body[++((s)->head)] = (stack_content)(x))
#define s_pop(s) ((s)->body[((s)->head)--])
#define s_squash(s) (((s)->body[(s)->head-1] = \
    (s)->body[(s->head)]) ,((s)->head--))
stack_content s_at(stack *s, int n);
void s_delete(stack *s, int n);
```

```
    void s_copy(stack *s, stack *d);
45  void s_sort(stack *s, comparator c);
    void s_uniq(stack *s, comparator c);
    void s_reverse(stack *s);
    void s_append(stack *s, stack *d);

50  /* EOF */
    #endif
```