Chip8

Reference Manual

Peter Miller
pmiller@opensource.org.au

.

This document describes Chip8 version 1.1
and was prepared 18 September 2012.

NAME

chip8 − project change supervisor

Chip8 is distributed under the terms of the GNU General Public License. See the LICENSE section, below, for more details.

DESCRIPTION

Chip8 was an interpreter used in a number of home computers based on RCA's CDP1802 processor in the late 1970's. It implements a small machine designed specifically for simple video games. It has less than 40 instructions, including arithmetic, control flow, graphics, and sound.

This package includes an assembler for chip8, a disassembler, and an X client to run chip8 programs. This distribution includes 33 games: 15puzzle, alien, ant, blinky, blitz, brix, car, connect4, field, guess, hidden, hpiper, invaders, joust, kaleid, maze, merlin, missile, pong, puzzle, race, snake, spacefight, syzygy, tank, tetris, tictac, uboat, ufo, vbrix, vers, wipeoff, worm3.

The assembler understood by this package differs from the assembler originaly documented for CHIP-8. This is for a number of reasons... (a) the original was unpleasant to write a lexer for, (b) the opcodes were not very orthogonal in appearance, (c) the format made arbitrary expression syntax have to tiptoe all around it, (d) if I changed it I could re-use code from another of my assemblers.

ARCHIVE SITE

The latest version of *Chip8* is available by HTTP from:

URL:	http://chip8.sourceforge.net/	
File:	index.html	# the Chip8 page
File:	chip8−1.1.README	# Description, from tar file
File:	chip8−1.1.lsm	# Description, in LSM format
File:	chip8−1.1.spec	# RedHat package specification
File:	chip8−1.1.tar.gz	# the complete source

This directory also contains a few other pieces of software written by me. Please have a look if you are interested.

BUILDING

Instructions on how to build and test *Chip8* are to be found in the *BUILDING* file included in this distribution.

LICENSE

Chip8 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.

Chip8 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, see <http://www.gnu.org/licenses/>.

It should be in the *LICENSE* file included in this distribution.

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

RELEASE NOTES

For excruciating detail, and also acknowledgments of those who generously sent me feedback, please see the *aux/CHANGES.** files included in this distribution.

A number of features have been added to *Chip8* with this release. A few of them are detailed here:

NAME

chip8 − X11 Chip8 interpreter

SPACE REQUIREMENTS

You may need up to 4MB of disk space to unpack and build the *Chip8* package. (This is the worst case seen so far, most systems have binaries about 60% as big as this, 2MB is more typical.) Your mileage may vary.

SITE CONFIGURATION

The **Chip8** package is configured using the *configure* shell script included in this distribution.

The *configure* shell script attempts to guess correct values for various system-dependent variables used during compilation, and creates the *Makefile* and *common/config.h* files. It also creates a shell script *config.status* that you can run in the future to recreate the current configuration.

Running Configure

Normally, you just *cd* to the directory containing *Chip8*'s source code and type

```
% ./configure
...lots of output...
%
```

If you're using *csh* on an old version of System V, you might need to type

```
% sh configure
...lots of output...
%
```

instead to prevent *csh* from trying to execute *configure* itself.

Running *configure* takes a minute or two. While it is running, it prints some messages that tell what it is doing. If you don't want to see the messages, run *configure* with the quiet option; for example,

```
% ./configure --quiet
%
```

By default, *configure* will arrange for the *make install* command to install the **Chip8** package's files in */usr/local/bin*, */usr/local/man* and */usr/local/share/chip8*. There are a number of options which allow you to control the placement of these files.

`--prefix=`*PATH*
: This specifies the path prefix to be used in the installation. Defaults to */usr/local* unless otherwise specified.

`--exec-prefix=`*PATH*
: You can specify separate installation prefixes for architecture-specific files and architecture-independent files. Defaults to *${prefix}* unless otherwise specified.

`--bindir=`*PATH*
: This directory contains executable programs. On a network, this directory may be shared between machines with identical hardware and operating systems; it may be mounted read-only. Defaults to *${exec_prefix}/bin* unless otherwise specified.

`--datadir=`*PATH*
: This directory contains installed data, such as the documentation, reports and shell scripts distributed with Chip8. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/share/chip8* unless otherwise specified. A ''chip8'' directory will be appended if there is none in the specified path.

`--mandir=`*PATH*
: This directory contains the on-line manual entries. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/man* unless otherwise specified.

configure ignores any other arguments that you give it.

On systems that require unusual options for compilation or linking that the *Chip8* package's *configure* script does not know about, you can give *configure* initial values for variables by setting them in the environment. In Bourne-compatible shells, you can do that on the command line like this:

```
$ CC='gcc -traditional' LIBS=-lposix ./configure
...lots of output...
$
```

Here are the *make* variables that you might want to override with environment variables when running *configure*.

Variable: CC
: C compiler program. The default is *cc*.

Variable: INSTALL
: Program to use to install files. The default is *install* if you have it, *cp* otherwise.

Variable: LIBS
: Libraries to link with, in the form `-l`*foo* `-l`*bar*. The *configure* script will append to this, rather than replace it.

If you need to do unusual things to compile the package, the author encourages you to figure out how *configure* could check whether to do them, and mail diffs or instructions to the author so that they can be included in the next release.

BUILDING CHIP8

All you should need to do is use the

```
% make
...lots of output...
%
```

command and wait. When this finishes you should see a directory called *bin* containing three files: *chip8as*, *chip8dis*, and *chip8run*.

chip8as The *chip8as* program is an assembler for Chip8 programs.

chip8dis
: The *chip8dis* program is a disassembler of Chip8 programs.

aereport The *chip8run* program is used run assembled Chip8 programs.

You can remove the program binaries and object files from the source directory by using the

```
% make clean
...lots of output...
%
```

command. To remove all of the above files, and also remove the *Makefile* and *common/config.h* and *config.status* files, use the

```
% make distclean
...lots of output...
%
```

command.

The file *etc/configure.ac* is used to create *configure* by a GNU program called *autoconf*. You only need to know this if you want to regenerate *configure* using a newer version of *autoconf*.

TESTING CHIP8

The *Chip8* package comes with a test suite. To run this test suite, use the command

```
% make sure
...lots of output...
Passed All Tests
%
```

The tests take less than a minute each, but they can vary greatly depending on your CPU.

INSTALLING CHIP8

As explained in the *SITE CONFIGURATION* section, above, the *Chip8* package is installed under the */usr/local* tree by default. Use the `--prefix=`*PATH* option to *configure* if you want some other path.

All that is required to install the *Chip8* package is to use the

```
% make install
...lots of output...
%
```

command. Control of the directories used may be found in the first few lines of the *Makefile* file if you want to bypass the *configure* script. You must also edit the other files generated by *configure*; it is usually easier to re-run *configure* with the appropriate arguments.

The above procedure assumes that the *soelim*(1) command is somewhere in the command search *PATH*. The *soelim*(1) command is available as part of the *GNU Groff* package, mentioned below in the *PRINTED MANUALS* section. If you don't have it, but you do have the *cook* package, then a link from *roffpp* to *soelim* will also work.

PRINTED MANUALS

This distribution contains the sources to all of the documentation for *Chip8*, however the simplest way to get the documentation is by anonymous FTP; a PostScript file of the Reference Manual is available from the FTP sites listed in the README file. The Reference Manual contains the README and BUILDING files, as well as all of the section 1 and section 5 manual pages.

GETTING HELP

If you need assistance with *Chip8*, please do not hesitate to contact the author at

Peter Miller <pmiller@opensource.org.au>

Any and all feedback is welcome.

When reporting problems, please include the version number given by the

```
% chip8run -version
Chip8 version 1.1.D094
...
%
```

command. Please run this command to get the exact number, do not send the text of this example.

Runtime Checking

In the *common/main.h* file, the is a define of *DEBUG* in comments. If the comments are removed, extensive debugging is turned on. This causes some performance loss, but performs much run-time checking and adds the **-TRAce** command line option.

When the **-TRAce** command line option is followed by one or more file names, it turns on execution traces in those source files. It is usually best to place this on the end of the command line so that names of the files to be traced are not confused with other file names or strings on the command line.

Problem Reports

If you send email to the author, please include the following information:

1. The type of UNIX

   The author will need to know the brand and version of UNIX you are using, or if it is not UNIX but something else. The output of "uname -sr" is usually sufficient (but not all systems have it).

2. The Version Number

   In any information you send, please include the version number reported in the *common/patchlevel.h* file, or `'chip8run -vers'` if you can get it to compile.

3. The Archive Site

   When and where you obtained this version of *Chip8*.

4. Unpacking

   Did you have problems unpacking *Chip8*?

5. Building

   Did you have problems building *Chip8*? This could have been the instructions included, it could have been the configure script, it could have been the Makefile, or anything else.

6. Testing

   Did you have problems with the tests? You could have had problems running them, or some of them could have failed. If some tests fail but not others, please let me know *which* ones failed, and include the fact that *Chip8* was **not** set-uid-root at the time. The −k option to *make* can be useful if some tests fail but not others.

7. Installation

   Did you have problems installing *Chip8*? This could have been the instructions, or anything else.

At this point it would probably be a very good idea to print out the manual entries and read them carefully. You will also want to print a copy of the Reference Manual; if you don't gave GNU Groff, there should be a PostScript copy at the archive site.

8. Using Chip8

   Did you have problems using *Chip8*? This is a whole can of worms. If possible, include a shell script similar to the tests which accompany *Chip8*, which reproduces the bug. Exit code 1 on failure (bug), exit code 0 on success (for when bug is fixed).

9. The Source Code

   Did you read the code? Did you write some code? If you read the code and found problems, fixed them, or extended *Chip8*, these contributions are most welcome.

The above list is inclusive, not exclusive. Any and all feedback is greatly appreciated, as is the effort and interest required to produce it.

LICENSE

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

NAME

chip8as − assemble chip8 programs

SYNOPSIS

chip8as [*option*...] *source-file binary-file*
chip8as −Help
chip8as −VERSion

DESCRIPTION

The *chip8as* program is used to assemble chip8 programs for execution by the *chip8run* program.

OPTIONS

The following options are understood.

−Help This option may be used to provide information about how to use the *chip8as* program.

−VERSion
This option may be used to provide inforation about the version on the *chip8as* program being executed.

−Listing *listing-file*
This option may be used to specify a listing file. By default, no listing is produced. The word − may be used to indicate *stdout*.

−Hewlett_Packard_Header
This option may be used to cause chip8as to emit a Hewlett-Packard 48 calculator header to the binary file. The default is to emit no header.

−Unix_Header
This option may be used to cause chip8as to emit a unix #! header to the binary file. he default is to emit no header.

All options are case-insensitive. Abbreviations are indicated by the upper-case letters. Options and other command line arguments may be aribtrarily mixed on the command line.

EXIT STATUS

The *chip8as* command will exit with a status of 1 on any error. The *chip8as* command will only exit with a status of 0 if there are no errors.

SEE ALSO

chip8as(5)
The opcodes, as understood by the assmbler, and how they are interpreted by the interpreter.

chip8run(1)
The interpreter, to run the assembled byte code.

chip8run(5)
The file format, as output by the assembler and as understood by the interpreter.

chip8dis(1)
A disassembler, so that you can turn assembled byte code into something more readable.

COPYRIGHT

chip8as version 1.1

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

NAME

chip8dis − disassemble chip8 programs

SYNOPSIS

chip8dis [*option*...] *binary-file assembler-file*
chip8dis −Help
chip8dis −VERSion

DESCRIPTION

The *chip8dis* program is used to disassemble chip8 programs.

OPTIONS

The following options are understood.

−Help This option may be used to provide information about how to use the *chip8dis* program.

−VERSion
This option may be used to provide inforation about the version on the *chip8dis* program being executed.

All options are case-insensitive. Abbreviations are indicated by the upper-case letters. Options and other command line arguments may be aribtrarily mixed on the command line.

EXIT STATUS

The *chip8dis* command will exit with a status of 1 on any error. The *chip8dis* command will only exit with a status of 0 if there are no errors.

SEE ALSO

chip8as(1)
An assembler, for assembling chip8 programs into the byte-code to be interpreted.

chip8as(5)
The opcodes, as understood by the assmbler, and how they are interpreted by the interpreter.

chip8run(1)
The interpreter, to run the assembled byte code.

chip8run(5)
The file format, as output by the assembler and as understood by the interpreter.

COPYRIGHT

chip8dis version 1.1

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software − to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that 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

Appendix: 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) 19yy *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, see <http://www.gnu.org/licenses/>.

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) 19yy 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.

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.

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:

> 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

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 Public License instead of this License.

NAME

chip8run − run a chip8 program

SYNOPSIS

chip8run [*option*...] *game*
chip8run [*option*...] **−File** *binary-file*
chip8run −Help
chip8run −VERSion

DESCRIPTION

The *chip8run* program is used to run chip8 programs. It is an X11 client.

The first form, where only the name of a game is given, constructs a filename of the form `/usr/local/lib/chip8/`*game*`.chp` and attempts to open it.

The second form may be used to open a specific file.

A great variety of file formats are understood, including some of the HP48 formats, and also the UNIX executable `#!` convention.

OPTIONS

The following options are understood.

−Help This option may be used to provide information about how to use the *chip8run* program.

−VERSion
This option may be used to provide inforation about the version on the *chip8run* program being executed.

−BackGround *color*
This option may be used to set the backgound of the *chip8run* window to the color specified.

−BorDer_color *color*
This option may be used to set the border color of the *chip8run* window to the color specified.

−Border_Width *number*
This option may be used to set the border width of the *chip8run* window to the width specified.

−DEbug
This option may be used to turn on debugging mode. Additional information is displayed as to the chip8 machine's internal state. Addition buttons are provided for additional control.

−Display *display-name*
This option may be used to set which X display to use. The default is the *$DISPLAY* environment variable, or **:0** if not set.

−FoNt *font-name*
This option may be used to set the font to be used in the *chip8run* window.

−ForeGround *color*
This option may be used to set the foregound of the *chip8run* window to the color specified.

−Geometry *geometry-spec*
This option may be used to specifiy the initial position and/or size of the *chip8run* window.

−IConic This option may be used to specify that the *chip8run* window should initially be iconic. You need a cooperative window managaer.

−Icon_Geometry *geometry-spec*
This option may be used to specify the position and/or size of the *chip8run* window's icon.

−Name *string*
This option may be used to set the name the *chip8run* window is referred to as.

−Test_Mode
This option may be used to test the chip8 machine. It makes the "Quit" button generate and exit code of 1, rather than 0 (it is an error if the user quits out of a test). It makes screen placement forced. Run-time errors result in exit 1, rather than turning on debugging.

−Title *string*
This option may be used to set the name the *chip8run* window is referred to as.

−XRM *string*
This option may be used to specify an X resource.

All options are case-insensitive. Abbreviations are indicated by the upper-case letters. Options and other command line arguments may be aribtrarily mixed on the command line.

EXIT STATUS

The *chip8run* command will exit with a status of 1 on any error. The *chip8run* command will only exit with a status of 0 if there are no errors.

FILES

/usr/local/lib/chip8
This directory is where games are installed. A number of games are included in this distribution.

SEE ALSO

chip8as(1)
An assembler, for assembling chip8 programs into the byte-code to be interpreted.

chip8as(5)
The opcodes, as understood by the assmbler, and how they are interpreted by the interpreter.

chip8run(5)
The file format, as output by the assembler and as understood by the interpreter.

chip8dis(1)
A disassembler, so that you can turn assembled byte code into something more readable.

COPYRIGHT

chip8run version 1.1

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

NAME

chip8as − chip8 opcodes

DESCRIPTION

This file documents the input format of the *chip8as* program. As a side-effect, it also documents the various opcodes the *chip8run* program understands. Input files are text files.

MACHINE

This section describes the chip8 machine.

Address Space

The chip8 machine has an address space from 0x000 to 0xFFF. This address space contains both code and data. Addresses 0x000 to 0x1FF are reserved. Some implementations place the interpreter here. When the chip8 machine is reset, the PC is set to 0x200, thus chip8 programs start at 0x200.

Opcodes

Opcodes are all 16 bits long. Opcodes must be aligned on even-byte boundaries (some implementations do not require this). Opcodes are stored big-endian, high byte first then low byte.

Registers

The chip8 machine has a number of registers.

v0 to **v15**
: These are general 8-bit unsigned arithmetic registers. Regsiters **v10** to **v15** may also be referred to as **vA** to **vF**. Arithmetic is done in 2s compliment.

v15
: In addition to being a general register, this register may sometimes be used as an exception indicator by some opcodes.

time
: The time register is a count-down timer. It is decrimented by 1 every 60th of a second until it reaches 0.

tone
: The time register is a count-down timer. It is decrimented by 1 every 60th of a second until it reaches 0. If the register is not 0, a beeper will be beeping. (Some implementations make the beeping optional.)

i
: This register is a pointer register. It is 16-bits long, although the address space is only 12-bits. It is an error to attempt to reference memory above 0xFFF.

.
: The dot register may be used to refer to the value of the program counter maintained by the assembler. It refers to the value of the program counter before the opcode is generated.

key
: This isn't really a register, even though the opcode treat it that way. There are 16 keys available, labeled "0" − "F". Keys may be sampled or waited for.

Display

The chip8 machine has a 64×32 display. It is 64 pixels wide and 32 pixels high. Row 0 is the top of the screen, row 31 is the bottom of the screen. Column 0 is the left of the screen, column 63 is rhe right of the screen. You may assume pixels are square. (See the *high* and *low* opcodes; there is also a high resolution 128×64 display mode.)

EMULATOR DIFFERENCES

There are a number of differences between the various emulators. No doubt this could all be resolved if anyone had the sources to the original (circa 1977) interpreter.

Add Aligned Opcodes
: Some emulators accept opcodes at odd addresses, and some don't. Defensive programming should use even-aligned opcodes. (This distribution demands even-aligned opcodes on emulation, and gives a fatal error for odd-aligned opcodes on assembly, but accepts odd-aligned

opcodes for disassembly.)

The I Register

The value of the I register after *save* and *restore* opcodes is not well defined. Some emulator leaves the I register pointing *past* the last location referenced, other emulators leave it pointing at the first location referenced. Defensive programming should assume the I register is not meaningful after these opcodes. (This distribution is consistent with Gustafsson's emulator, which moves the I register *past* the last location referenced.)

The Borrow Flag

Many emulators calculate the *borrow* after the *sub* and *dif* opcodes incorrectly. Defensive programming should assume the v15 register is not meaningful after these opcodes. (This distribution is consistent with Gustafsson's emulator, which performs the operation in more than 8 bits, and sets the v15 register to 1 if any of the high bits of the result are non-zero, otherwise sets it to 0)

LINE FORMAT

The input of the *chip8as* program is oriented around lines. Each line, in general, has the form

[*label***:**] *opcode* [*expr***,** ...]

The label definition on the start of the line is optional, opcodes take zero or more expressions. End-of-line is significant, except after commas. It is also legal to have blank lines, or lines consisting of only a label definition. Forward references of labels are legal, as it is a 2-pass assembler.

TOKENS

This section details how the text file is broken into tokens.

White Space

White space consists of tabs and spaces. White space is ignored, except where it serves to separate tokens. Comments commence with semicolon (';') and end at end-of-line ('\n'); comments are logical white space.

Identifiers

Identifiers start with an alphabetic character or and underscore ('_') or dot ('.'); followed by zero or more alphabetic, numeric, underscore or dot characters. Identifiers are case-sensitive unless they are opcode names or register names. Identifiers may be of any length, and all characters are significant.

Numbers

The default radix is decimal. An "0x" prefix will yeild hexadecimal numbers, a "0" prefix will yield octal numbers.

OPCODES

The chip8 machine opcodes are described here.

`scdown` *n*

Scroll the screen down *n* pixels. *[Super-Chip]*
This opcode delays until the start of a 60Hz clock cycle before drawing in low resolution mode. (Use the delay timer to pace your games in high resolution mode.)
Code generated: `0x00C`*n*

`clear`

Clear the display.
Code generated: `0x00E0`

`ret`

Return from subroutine. See also: the "call" opcode. It is an error if there is no subroutine to return from.
Code generated: `0x00EE`

`compatibility`
: Mangle the ‘‘save’’ and ‘‘restore’’ opcodes to leave the I register unchanged.
Warning: This opcode is not a standard Chip 8 opcode. It is provided soley to allow testing and porting of Chip 8 games which rely on this behaviour.
Code generated: `0x00FA`

`scright`
: Scroll the screen right 4 pixels. *[Super-Chip]*
This opcode delays until the start of a 60Hz clock cycle before drawing in low resolution mode. (Use the delay timer to pace your games in high resolution mode.)
Code generated: `0x00FB`

`scleft`
: Scroll the screen left 4 pixels. *[Super-Chip]*
This opcode delays until the start of a 60Hz clock cycle before drawing in low resolution mode. (Use the delay timer to pace your games in high resolution mode.)
Code generated: `0x00FC`

`low`
: Low resolution (64×32) graphics mode (this is the default). *[Super-Chip]*
Code generated: `0x00FE`

`high`
: High resolution (128×64) graphics mode. *[Super-Chip]*
Code generated: `0x00FF`

`jump` *addr*
: Jump to *addr*. The *addr* must be even. The *addr* must be in the range 0x200 to 0xFFE. The *addr* expression must be relative to some label.
Code generated: `0x0NNN`, where NNN is the low 12 bits of *addr*.

`jump` *addr*`, v0`
: Jump to *addr* + v0. The *addr* must be even. The *addr* must be in the range 0x200 to 0xFFE. The *addr* expression must be relative to some label. It is a run-time error if the value of register v0 is odd. It is a run-time error if *addr*+v0 is not in the range 0x2000 to 0xFFE.
Code generated: `0xBNNN`, where NNN is the low 12 bits of *addr*.

`call` *addr*
: Call subroutine at *addr*. The *addr* must be even. The *addr* must be in the range 0x200 to 0xFFE. The *addr* expression must be relative to a label. It is a run-time error if there are too many subroutine calls, although you may safely assume at least 32 levels.
Code generated: `0x1NNN`, where NNN is the low 12 bits of *addr*.

`skip.eq` v*X*, *value*
: Skip the next instruction if the value of register v*X* is equal to *value*. The *value* must be in the range −128 to 255.
Code generated: `0x3XYY`, where YY is the low 8 bits of *value*.

`skip.eq` v*X*, v*Y*
: Skip the next instruction if the value of register v*X* is equal to the value of register v*Y*.
Code generated: `0x3XY0`

`skip.eq` v*X*, `key`
: Skip the next instruction if they key with the same number as the low 4 bits of the value of register v*X* is currently being pressed.
Code Generated: `0xEX9E`

`skip.ne` v*X*, *value*
: Skip the next instruction if the value of register v*X* is not equal to *value*. The *value* must be in the range −128 to 255.
Code generated: `0x4XKK`, where KK is the low 8 bits of *value*.

`skip.ne` v*X*, v*Y*
: Skip the next instruction if the value of register v*X* is not equal to the value of register v*Y*.
Code generated: `0x9XY0`

`skip.ne` v*X*, `key`
: Skip the next instruction if the key with the same number as the the low 4 bits of the value of register v*X* currently not being pressed.
Code generated: `0xEXA1`

`load` v*X*, *value*
: Load register v*X* with the *value*. The *value* must be in the range −128 to 255.
Code generated: `0x6XKK`, where KK is the low 8 bits of *value*.

`load` v*X*, `key`
: If no key is currently being pressed, block until one is. Load register v*X* with lowest number of all keys currently being pressed. The beeper will sound while any key is being pressed. Block until the key is released (implementations need not block if they guarantee that the key will not be "seen" by any of the key opcodes until it is released and pressed again).
Code generated: `0xFX0A`

`load` v*X*, v*Y*
: Load register v*X* with the value of register v*Y*. Some implementations may alter the value of register v15 (to what?).
Code generated: `0x8XY0`

`load` v*X*, `time`
: Load register v*X* with the value of the time register.
Code generated: `0xFX07`

`load time`, v*X*
: Load the time register with the value of register v*X*.
Code generated: `0xFX15`

`load tone`, v*X*
: Load the tone register with the value of register v*X*.
Code generated: `0xFX18`

`load i`, *addr*
: Load register i with the *addr*. The *addr* must be in the range 0x200 to 0xFFF.
Code generated: `0xANNN`, where NNN is the low 12 bits of *addr*.

`add` v*X*, *value*
: Add *value* to register v*X*. The *value* must be in the rabge −128 to 255.
Code generated: `0x7XKK`, where KK is the low 8 bits of *value*.

`add` v*X*, v*Y*
: Add the value of register v*Y* to register v*X*. The register v15 is set to 1 if the result overflows, otherwise 0.
Code generated: `0x8XY4`

`add i`, v*X*
: Add the value of register v*X* to register i.
Code generated: `0xFX1E`

`or i`, v*X*
: Bitwise OR the value of register v*Y* into register v*X*. Some implementations may change the value of register v15 (to what?).
Code generated: `0x8XY1`

`and` v*X*, v
: Bitwise AND the value of register v*Y* into register v*X*. Some implementations may change the value of register v15 (to what?).

Code generated: `0x8XY2`

`xor` v*X*, v*Y*
: Bitwise XOR the value of register v*Y* into register v*X*. Some implementations may change the value of register v15 (to what?).
Code generated: `0x8XY3`

`sub` v*X*, v*Y*
: Subtract the value of register v*Y* from register v*X*. Register v15 is set to 1 is the result would be less than zero, 0 otherwise.
Code generated: `0x8XY5`

`shr` v*X*
: Shift the value of register v*X* right one bit. Register v15 is set to 1 if v*X* was odd before the operation, 0 otherwise.
Code generated: `0x8X06`

`dif` v*X*, v*Y*
: Set register v*X* to the value of register v*Y* minus the value of register v*X*. Register v15 is set to 1 is the result would be less than zero, 0 otherwise.
Code generated: `0x8XY7`

`shl` v*X*
: Shift the value of register v*X* left one bit. Register v15 is set to 1 if the high bit of register v*X* was set before the operation, 0 otherwise.
Code generated: `0x8X0E`

`rnd` v*X*, *value*
: Register v*X* is set to the bitwise AND of a pseudo-random number and the *value*. The *value* must be in the range 0 to 255.
Code generated: `0xCXKK`, where KK is the low 8 bits of *value*.

`draw` v*X*, v*Y*, *rows*
: This opcode is used to draw an image on the screen. The image will be 8 pixels wide and *rows* pixels long. The image will be displayed at (x,y) coordinates, where x is the value of register v*X* bitwise-AND 0x3F, and y is the value of register v*Y* bitwise-AND 0x1F. If any of the image would be drawn outside the screen area, it is clipped (it does not wrap around).
The origin (0,0) is the top-left corner of the screen. The image to be drawn is pointed to by the i register. The most-significant bit is on the left.
Drawing is done by using XOR. If this causes one or more pixels to be erased, v15 is set to 1, otherwise v15 is set to 0.
The *rows* must be in the range 1 to 15. It is a run-time error if the value of register i causes non-existent memory to be accessed.
This opcode delays until the start of a 60Hz clock cycle before drawing in low resolution mode. (Use the delay timer to pace your games in high resolution mode.)
Code generated: `0xDXYN`, where N is the low 4 bits of *rows*.

`xdraw` v*X*, v*Y*, *rows*
: As above, however the image is always 16×16 pixels. *[Super-Chip]*
Code generated: `0xDXY0`

`hex` v*X*
: Point I to an image of a hex character for the low 4 bits of the value of register v*X*. The image is 4 pixels wide and 5 pixels high.
Code generated: `0xFX29`

`bcd` v*X*
: Store a BCD representation of the value of register v*X* into the three bytes pointer to be register i, most significant digit first. It is a run-time error if the value of register i causes non-existent memory to be accessed.

Code generated: `0xFX33`

`save` v*X*
: Store the values of registers v0 to v*X* into the bytes pointed to by register i, incrimenting register i past them. It is a run-time error if the value of register i causes non-existent memory to be accessed.
Code generated: `0xFX55`

`restore` v*X*
: Read the values of registers v0 to v*X* from the bytes pointed to by register i, incrimenting register i past them. It is a run-time error if the value of register i causes non-existent memory to be accessed.
Code generated: `0xFX65`

`flags.save` v*X*
: Store the values of registers v0 to v*X* into the ‘‘flags’’ registers (this means something in the HP48 implementation). (*X* < 8) *[Super-Chip]*
Code generated: `0xFX75`

`flags.restore` v*X*
: Read the values of registers v0 to v*X* from the ‘‘flags’’ registers (this means something in the HP48 implementation). (*X* < 8) *[Super-Chip]*
Code generated: `0xFX85`

`exit`
: This opcode is used to terminate the *chip8run* program. It causes the *chip8run* program to exit with a successful exit status. *[Super-Chip]*
Code generated: `0x00FD`.

`exit` *value*
: This opcode is used for testing the *chip8run* program, and are not normal chip8 opcodes. They cause the *chip8run* program to exit with the given exit status. This is primarily of use in performing regression tests of the chip8 machine implementation. The *value* must be in the range 0 to 1.
Exit status 0 indicates a successful result, exit status 1 indicates an error result.
Code generated: `0x001X`, where X is the low 4 bits of *value*.

PSEUDO-OPS

The following are pseudo-ops of the *chip8as* program.

`.ascii` *expr*, ...
: Each expression is emitted into the code stream as the bytes composing the given strings. This is usually used to embed copyright notices into programs. There is no built-in ascii font available for drawing.

`.byte` *expr*, ...
: Each expression is emitted into the code stream as a byte. The *expr*s must be in the range −128 to 255.

`.word` *expr*, ...
: Each expression is emitted into the code stream as a word, high byte then low byte. It is an error if `.` is odd. The *expr*s must be in the range −32768 to 32767.

`.align` [*expr*]
: This pseudo-op may be used to align `.` with various boundaries. The default is a word boundary (2 byte). Alignment boundaries must be powers of 2. Zero or more zero-valued bytes are emitted into the code stream to achieve alignment.

`.ds` *expr* [, *expr*]
: This pseudo-op may be used to define storage space in the code stream. The first argument is the number of bytes to use, the second argument is the value to store there (defaults to zero).

label `.equ` *expr*
This pseudo-op may be used to set a label to a calculated expression.

`.pic` *expr*, ...
This pseudo-op may be used to construct a picture for future drawing by a series of `draw` opcodes. It breaks the image up into a series of vertical slices 8 pixels wide.
All of the expressions must be strings, blank characters will be 0s in the output, all other characters will be 1s. It does not understand tabs. All the strings must be of the same length.
See the *snake* example program distributed with chip8 for a number of examples of this pseudo-op.

`.title` *word*, *word*
This pseudo-op may be used to set the page title on the listing. The title has two lines. The first argument is the first line of the title, and the second argument is the second line.

`.xpic` *expr*, ...
This pseudo-op may be used to construct a picture for drawing by a series of `xdraw` opcodes. It breaks the image up into a series of vertical slices 16 pixels wide. Because the `xdraw` opcode requires a 16 line high image, the given picture will be padded to a multiple of 16 lines, if necessary.

`.xref on`
This pseudo-op may be used to cause a cross reference to be added to the listing.
All of the expressions must be strings, blank characters will be 0s in the output, all other characters will be 1s. It does not understand tabs. All the strings must be of the same length.

CONDITIONAL ASSEMBLY

It is possible to conditionally assemble different portions of code. Conditionals may be nested arbitrarily deeply. This is controlled by the following directives:

`define` *name*
This directive is used to define a conditionl name. The conditions are based on the existence or absence of these lines.

`ifdef` *name*
Begin a conditionl portion of code, including the bracketed lines if the *name* was previously defined by a `define` directive.

`ifndef` *name*
Begin a conditionl portion of code, including the bracketed lins if the *name* was not previously defined by a `define` directive.

`else` Reverse the sense of a conditional compilation. (This is optional.)

`endif` End the bracketing of source lines performed by a conditional.

SEE ALSO

chip8as(1)
An assembler, for assembling chip8 programs into the byte-code to be interpreted.

chip8as(5)
The opcodes, as understood by the assmbler, and how they are interpreted by the interpreter.

chip8run(1)
The interpreter, to run the assembled byte code.

chip8run(5)
The file format, as output by the assembler and as understood by the interpreter.

chip8dis(1)
A disassembler, so that you can turn assembled byte code into something more readable.

COPYRIGHT

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

NAME

chip8 file formats

DESCRIPTION

The *chip8run* program understands 3 file formats: raw, HP and #!. The only difference is the headers. All 3 formats have a header, followed by data. The data will be at most 3584 bytes long.

RAW FORMAT

The raw format has no header. There is only the data section in the file.

HP FORMAT

the HP format has a 13 byte header in the format

8-bytes	magic number "HPHP48-A" 48,50,48,50,34,38 (hex)
2.5-bytes	type 0x02C2A for "string"
2.5-bytes	length in nybbles, including length.

The 2.5 byte fields are encodes as big-endian nybbles.

UNIX FORMAT

The #! format discards all bytes upto the first newline ('\n') character, this is useful to make chip8 programs self-executing, by making the first line

```
#! /usr/local/bin/chip8run −
```

Which tells unix to invoke the *chip8run*(1) program and feed it the executing file on its standard input.

SEE ALSO

chip8as(1)
: An assembler, for assembling chip8 programs into the byte-code to be interpreted.

chip8as(5)
: The opcodes, as understood by the assmbler, and how they are interpreted by the interpreter.

chip8run(1)
: The interpreter, to run the assembled byte code.

chip8dis(1)
: A disassembler, so that you can turn assembled byte code into something more readable.

COPYRIGHT

chip8 version 1.1

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

NAME

15puzzle − X11 Chip8 game

SYNOPSIS

chip8run 15puzzle

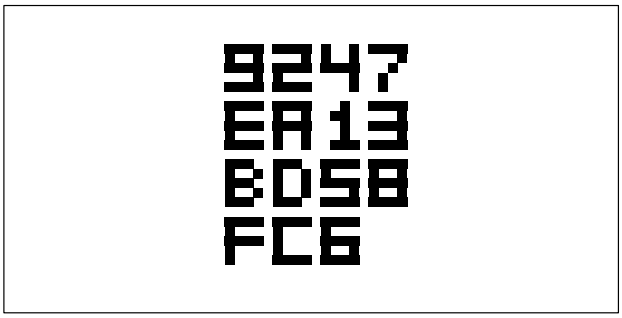
DESCRIPTION

Here is the infamous 15 puzzle for Chip8. The puzzle, as you doubtless recall, has 15 squares numbered 1 through 15 (in this case, 1 through F) and one hole. You can move the hole about and must put the pieces in order.

When you first run puzzle, it comes up solved. Thereafter, pressing IRestart causes the puzzle to have 32 random moves made, effectively randomizing the puzzle. If 32's not enough for you, just press IRestart again.

The program does not check to see if you've solved the puzzle, and therefore nothing special happens when you do except for the warm, fuzzy feeling that you have beaten it.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

You enter moves by using the keys. The key's position in that 4×4 matrix corresponds to the square in that position of the puzzle matrix. Pushing a key causes the hole to migrate to that position. The migration is performed in the order up, down, left, right; it is not necessary to limit your moves to those rows and columns containing the hole; you can request that the hole move to any position.

COPYRIGHT

15 Puzzle version 1.0
Author Unknown

AUTHOR

Author Unknown

NAME

Alien − Shoot the alien invaders

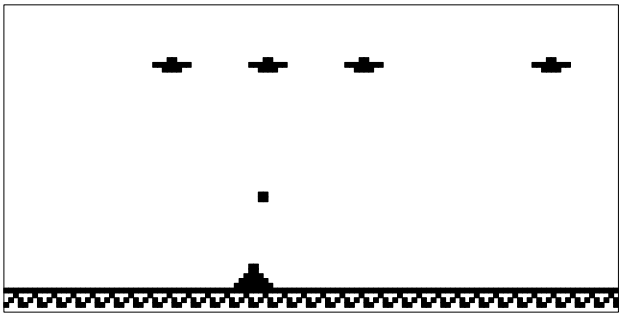
SYNOPSIS

chip8run alien

DESCRIPTION

None.

The Display



They Keys

1	2	3	C	3: Left C: Right
4	5	6	D	A: Fire
7	8	9	E	
A	0	B	F	

Copyright

Jonas Lindstedt

Author

Jonas Lindstedt

NAME

ant − In search of Coke

SYNOPSIS

chip8run ant

DESCRIPTION

Rumours through the grape vine of the local ant community indicate that there is a partially empty Coke can in Zoom's room. So far many brave ants have gone in search of the mystic can, yet none have returned. They are feared to be dead. It is your mission as Bink to find a safe path to the Coke can so that others may follow. Beware, however, for Zoom has set many treacherous obsticles to block you and booby traps to zap you. Bink has only his cool nerves and jumping ability (the only jumping ant in the world) to help him on this journey.

Pressing the jump key and the right key simultaneously will help clearing large obsticles. Actually, press the jump key slightly ahead of the right key.

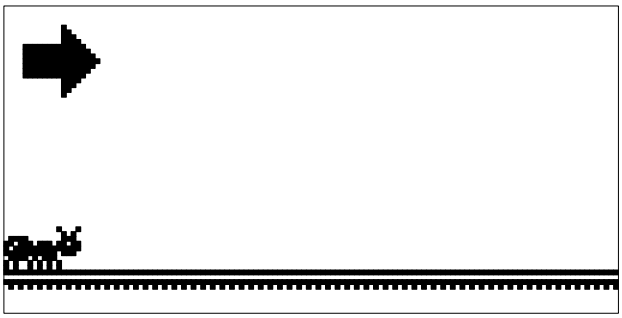
Being of little mass, Bink can stop on a dime and change course in mid-air at will.

This game has no randomization. Learn the terrain.

I take full credit for the design and development of Ant. The game theme is original as far as I know.

There are many tricks to getting around Ant. Be aware of spoilers when posting to the net.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

3: Left
C: Right

A: Jump

COPYRIGHT

Ant version 1.0
Copyright (C) 1991 Erin S Catto

I take no blame for worn calculator keys.

AUTHOR

Erin S Catto <catto@ecn.purdue.edu>

NAME

Blinky − PacMan clone

SYNOPSIS

chip8run blinky

DESCRIPTION

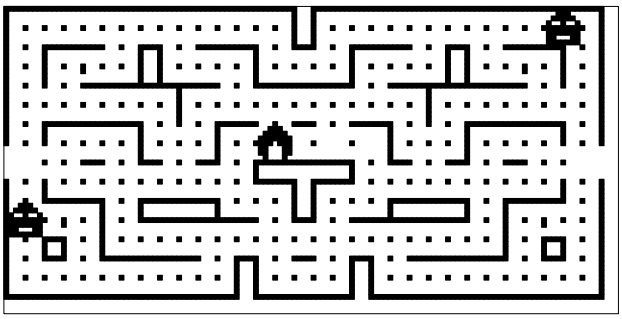
Blinky is a PacMan game for the Chip8.

As Blinky, you are chased around in an office environment by two bosses, Packlett and Heward. Packlett does management by walking around, but Heward believes in the American dream, and possesses quite a killer instinct. Anyway, don't let them get to you, unless you are feeling very aspirant. This, of course, requires the recent fulfilment of one of four major contracts, found near the corners of the building. Otherwise, the office is filled with small tasks, just waiting for your attention. If you manage to take care of them all, your intray will overflow, just over weekend. This is the curse of any responsible and hard working employee. However, neither boss know of the emergency exit which leads from one part of the office to the other, so this may be one way to avoid them, if everything else fails. As in most decent companies, you are given a chance to clear up the mess of your first blunder. The second time on the rug... off you go. There is a small comfort in that the recommendation reflects how well you did.

Recommendation points are awarded as follows:

0 points	Make use of the emergency exit
1 point	Take care of a minor task
4 points	Negotiate a major contract
25 points	Show Heward a recently fulfiled contract
50 points	Show Packlett a recently fulfiled contract
100 points	Clear the office environment from tasks

The Display



The Keys

Blinky control keys are:

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

1: reset
3: up
6: down
7: left
8: right

COPYRIGHT

AUTHOR

Christian Egeberg <egeberg@solan.unit.no>

NAME

Blitz − bomb the bad guys

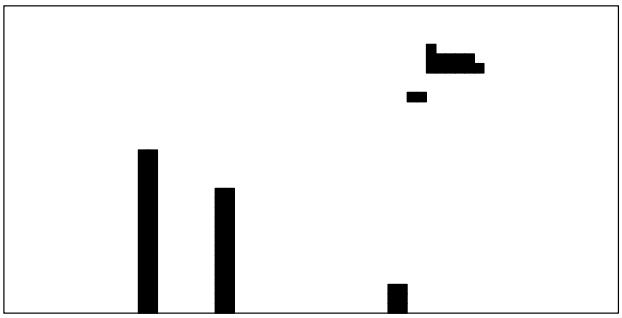
SYNOPSIS

chip8run blitz

DESCRIPTION

This game is a BOMBER clone. You are in a plane, and you must destroy the towers of a town. Your plane is flying left to right, and goes down. The game ends when you crash yourself on a tower...

This Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

5: Drop bomb

COPYRIGHT

Blitz version 1.0
Copyright (C) David Winter

AUTHOR

David Winter <winter@worldnet.net>

NAME

Brix − Knock out the bricks

SYNOPSIS

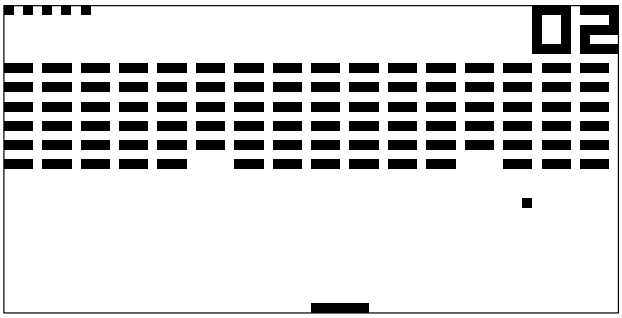
chi8run brix

DESCRIPTION

This is a version of the classic game of knock out the bricks. Not particularly exciting, but I had to include something to show how the interpreter is used.

Once you get all of the bricks, the game will freeze. Use the restart button to start the game again.

The Display



The dots at the top-left indicate the number of balls you have remaining.

The number at the top-right is the number of bricks you have hit so far. You are aiming for a total of 96.

The short line at the bollom of the screen is the paddle. The ball will bounce off the top, the sideas and the paddle. If it touches the bottom of the screen, you lose a ball.

The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: move paddle left
6: move paddle right

COPYRIGHT

AUTHOR

Andreas Gustafsson <gson@niksula.hut.fi>

NAME

Car − driving game

SYNOPSIS

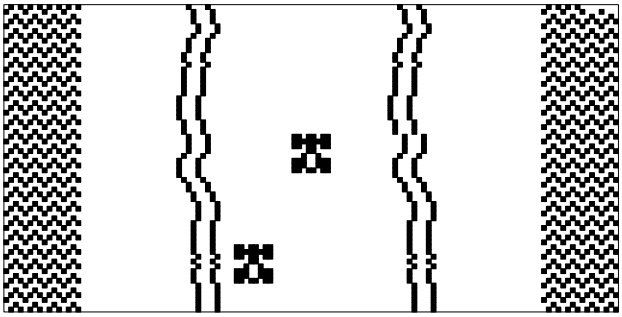
chip8run car

DESCRIPTION

This game simulates a racing car.

The other cars are a big dumb, though. They seem to have all stalled.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

7: left
8: right

COPYRIGHT

Car version

AUTHOR

K. v. Sengbusch

NAME

Connect4 − two player join the dots

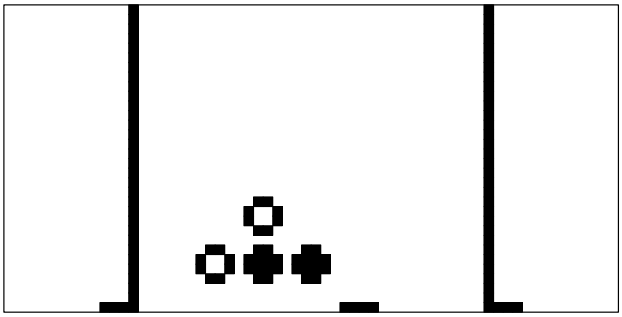
SYNOPSIS

chip8run connect4

DESCRIPTION

This game is for two players. The goal is to align 4 coins in the game area. Each player's coins are colored. When you drop a coin, it is paced on the latest dropped coin in the same column, or at the bottom if the column is empty. Once the column is full, you cannot place any more coins in it. To select a column, use 4 and 6. To drop a coin, use 5. There is no winner detection yet. This will be soon avalaible (Hey! I don't spend my life on CHIP8 !).

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: left
5: place piece
6: right

Placement of pieces alternate.

Points

1 point	per egg touched
8 points	for clearing bonus round
1 player	for clearing bonus round

COPYRIGHT

AUTHOR

David Winter <winter@worldnet.net>

NAME

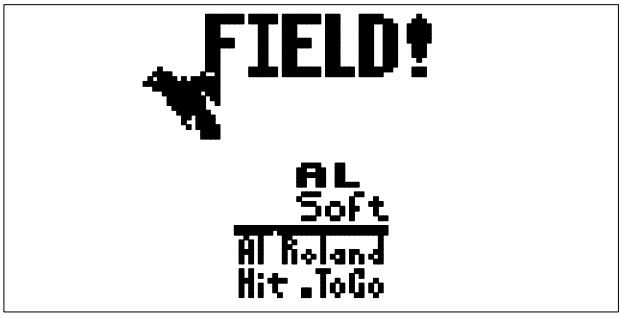
Field − mavigate the asteroid field

SYNOPSIS

chip8run field

DESCRIPTION

A title screen will appear proclaiming the game name (and my name) with quite a bit of animation. The amount of title screen animation explains for its rather large size for a S-CHIP game of 700 some odd bytes. However, it is still amazing to me that a game as simple as this can be written in so few bytes!



To play press the ‘‘0’’ key during the title sequence, you may have to hold it down for a second or to (since I don’t check keypresses between every single frame of animation.)

The screen will then appear like:

```
 _____________________________
|#                           #|
|                             |  .−−−−ASTEROIDS
|#                            |  |
|                            #<−−−’
|                             |
|#___________________________#|
```

The keys are now defined as:

1 − Apply thrust to the left (so you move right!) 2 − Apply thrust to the right (so you move left!) / − Move up * − Move Down 7 − Abandon Game and return to the Title Screen

Now once the game is going.(you must apply thrust to actually start.) The screen appears like this.

```
 _____________________________ |#       #          #| |    +−        #     | |#   ^        #      | | # |  #
#| |    |        #      | |#____|____________________#|
    |
   You
```

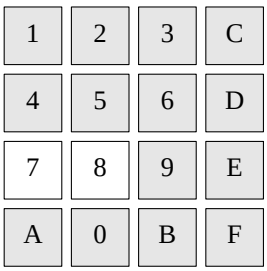
Now your on your own... Just don’t hit anything.

Your score is based on how long you last, actually it is more of a rating than a score, and it is revealed to you by the OWL on the Title screen. The score is 0−F (in *hex* because that was easier to do, and since this is my *first* chip program, I think you can be happy with that, anyway it will probably only be by an act of God that you make it very far.)

Hope you enjoy!

Let me know if you like it, or if there are any bugs, or you want more. I would greatly appreciate hearing from any of you HP48sx whizes. This horizontal scrolling looks tremendously like DEFENDER and perhaps that is where this program will go!.

The Display
The Keys



7: left
8: right

COPYRIGHT

Field version 1.0

Oh yeah... I release this as FREEWARE, play it, and enjoy it. But don’t modify, mutilate, or spindle this program unless I know about (and approve) it.

AUTHOR

Al Roland <droland@eng.auburn.edu>

711 Cary Drive
Auburn, Al 36830

I, Al Roland, being of sound body and mind, and poor grammatical and spelling ability, make no warranties expressed or implied to anyone no matter how unsound their body and mind are regarding the included software, and the damages that might result to those unsound bodies and minds, when they misuse this software on their calculator. I also will not be held responsible for a grade drop when you play this game in class and forget to take notes. And, blah blah blah....

NAME

Guess − guess the number

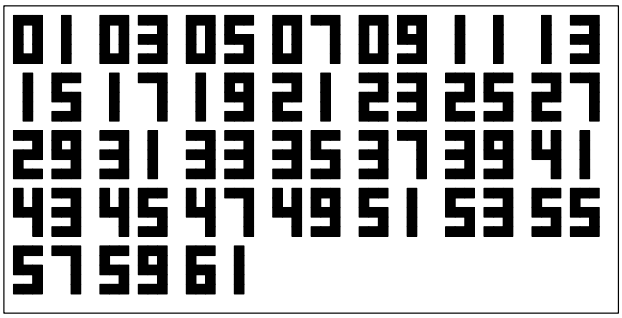
SYNOPSIS

chip8run guess

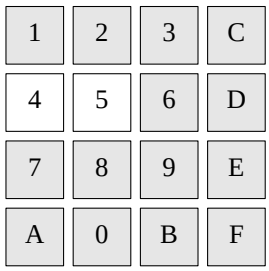
DESCRIPTION

Think to a number between 1 and 63. CHIP8 shows you several boards and you have to tell if you see your number in them. Press 5 if so, or another key if not. CHIP8 gives you the number...

The Display



The Keys



5: Yes
4: No

COPYRIGHT

Guess version 1.0
Copyright (C) 1991 David Winter

AUTHOR

David Winter <winter@worldnet.net>

NAME

Hidden − find pairs of cards

SYNOPSIS

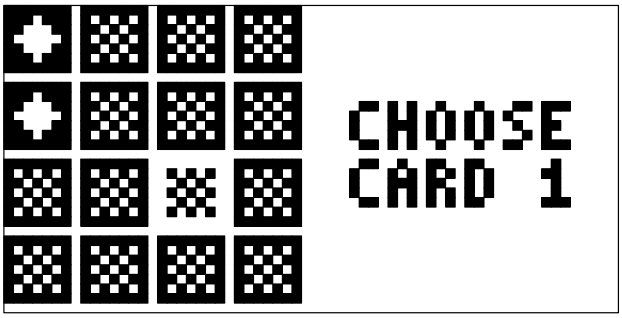
chip8run hidden

DESCRIPTION

This is a version of the classic memory game. You are asked to find pairs of cards.

The Display

At the openning splash screen, press any key. You will then see the cards.



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

2: up
4: left
5: pick
6: right
8: down

COPYRIGHT

Hidden version 1.0
Copyright (C) 1991 David Winter

AUTHOR

David Winter <winter@worldnet.net>

NAME

hpiper − plumbing game

SYNOPSIS

chip8run hpiper

DESCRIPTION

This probably not the super game you were hoping for, Erik, but it has only been three days. The only SCHIP feature this game takes advantage of is the higher screen resolution.

OBJECT

You are presented with a 6 row, 10 column grid and a preview panel of 5 pipes. A starting pipe will be placed on the grid and will in a few seconds start leaking. The object is to lay pipes on the grid to contain the leak as long as you can. If the water flows onto a empty grid site, onto the side of another pipe or border, that plumbing job is finished. If you did well enough, you may get another job. Otherwise the game is over.

HOW TO PLAY

On the bottom and right borders of the grid are grid pointers. The bottom pointer can be moved left with the (7) key and right with the (8) key. The side pointer can be moved up with the (3) key and down with the (6) key. (Note: this is identical to the movement keys for Syzygy by Roy Trevino)

Using the pointers, chose a grid site to place a pipe. The pipe to be placed is the one at the bottom of the preview column. Once a site is chosen, press the (1) key to place the pipe there. You can replace a pipe already at the grid site with a penalty of one point. You cannot replace a pipe that the water has already flown through. Trying to do this will make you lose the pipe you were trying to place and also penalize you one point.

When you have placed all the pipe you wish to for a particular job, you can press the (F) key to make the water flow *fast*. You can still lay pipe if you realize you made a mistake, but hurry!

There are ten levels with increasing water flow speed. Level 10’s speed is the same as that for *fast*. If you manage to get through level 10, it repeats at that level till you don’t.

SCORING

You receive 3 points for every pipe the water flows through. For the crossed pipes, flowing through them in both directions gives you 6 points. (Note: some adventuresome combinatorist might want to figure out what the maximum possible score is. If it is over 255, there could be a problem, but I doubt it.) You are penalized 1 point for replacing or trying to replace a pipe already on the grid.

The current job score is shown in upper right corner up to 99. If the score goes over 99, the displayed score rolls over. However, up to 255, the full score is kept in memory.

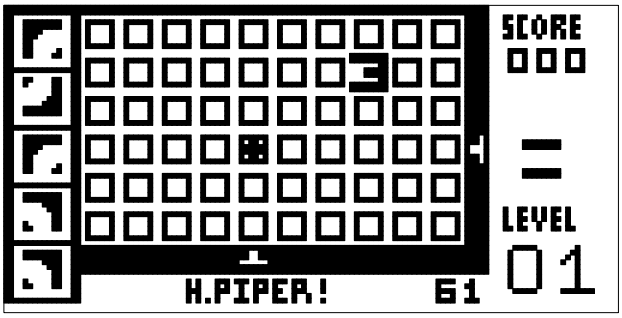
Once a job is over, the score earned is added to the grand total and displayed on the screen shown after (SPACE) is pressed. This total has two parts. The first part before the two periods is the total up to 255. The second part is the number of times the total rolled over. Therefore to get the real total, multiply the second part by 256 and add the first part. (Note: Sorry about this method, but I haven’t worked out the preferred way yet.)

If the game is over, the word OVER will be displayed above the score and the (DEL) key must be pressed to exit. Else, press (SPACE) to go on to the next job. The points needed to go to the next job are obtained by the following formula:

Points needed = 100 − (41 − 4 * level)

Happy plumbing!

The Display



The Keys

The keys used are as follows:

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

1: place
3: up
4: uplace
6: down
7: left
8: right
F: fast

COPYRIGHT

Since CHIP makes use of undocumented features of the HP48SX, anything happen: loss of data, meltdown, *etc*. Therefore, I take no responsibility for any damage whatsoever that occurs.

AUTHOR

Paul Raines <vapsppr@prism.gatech.edu>

NAME

Invaders − a space invaders clone

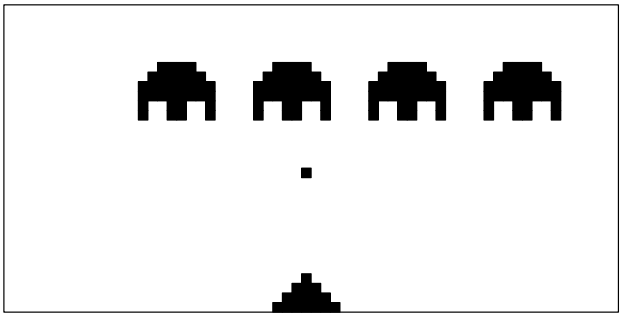
SYNOPSIS

chip8run invaders

DESCRIPTION

The well known game. Destroy the invaders with your ship.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: left
5: fire
6: right

COPYRIGHT

Invaders version 0.9
Copyright (C) 1991 David Winter

AUTHOR

David Winter <winter@worldnet.net>

NAME

Joust − arcade game

SYNOPSIS

chip8run joust

DESCRIPTION

Hoorah! Hoorah! Here it is, the latest CHIP48 game; my rendition of the arcade game Joust.

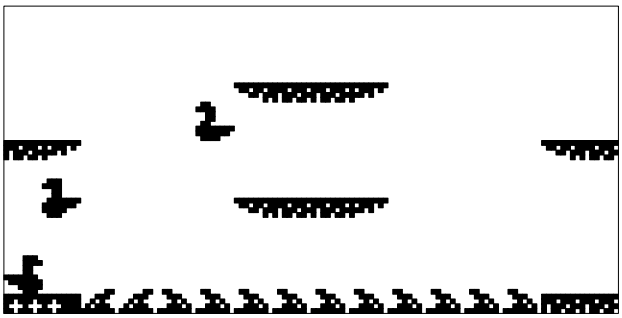
The blank dots in the bottom left corner indicate the number of players left, not including the current player. Sorry, no bonus guys. Maybe in the future.

At the score screen press A to replay.

If you feel the game is slow, then wait until you get to level 7! The levels roll over at 15, I think. I haven't gotten that far! Therefore there maybe unseen bugs.

The game is novel. I'm not sure of its lasting qualities.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

3: left
C: right
A: up

A: At the score screen press A to replay.

Points

1 point	per egg touched
8 points	for clearing bonus round
1 player	for clearing bonus round

COPYRIGHT

AUTHOR

Erin S Catto <catto@ecn.perdue.edu>

NAME

kaleid − draw pretty patterns

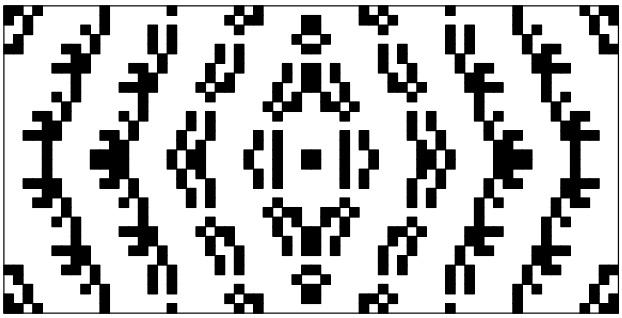
SYNOPSIS

chip8run kaleid

DESCRIPTION

A little program (not a game) to make funny graphics. Move around the screen with 2 4 6 8. To finish and make CHIP8 repeat your moves, press 0.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

0: finish
2: up
4: left
6: right
8: down

COPYRIGHT

kaleid version 1.0

AUTHOR

David Winter <winter@worldnet.net>

NAME

maze − draws random mazes

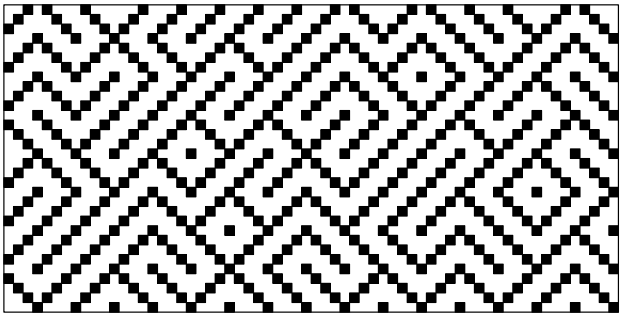
SYNOPSIS

chip8run maze

DESCRIPTION

This little program draws random mazes.

The Display



The Keys

Press any key to cause a different maze to be drawn.

COPYRIGHT

AUTHOR

David Winter <winter@worldnet.net>

NAME

merlin − remember the order

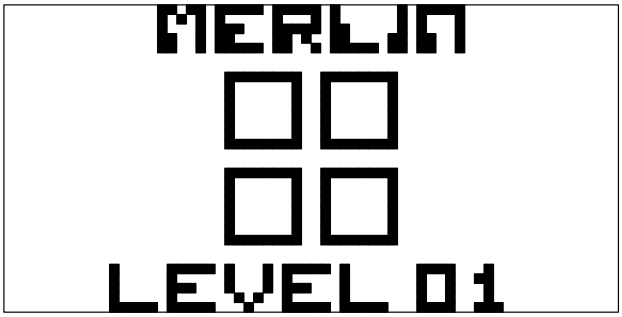
SYNOPSIS

chip8run merlin

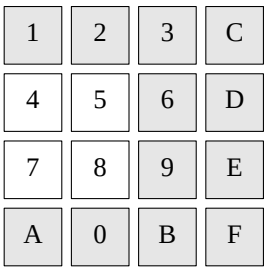
DESCRIPTION

This is the *Simon* game. The goal is to remember in which order the squares are lighted. The game begins by lighting 4 random squares, and then asks you to light the squares in the correct order. You win a level when you give the exact order, and each increasing level shows a additionnal square. The game ends when you light an incorrect square. Keys are 4 and 5 for the two upper squares, then 7 and 8 for the two lower ones.

The Display



The Keys



4: left
6: right

COPYRIGHT

merlin version 1.0

AUTHOR

Unknown <unknown@example.com>

NAME

missile − shoot the targets

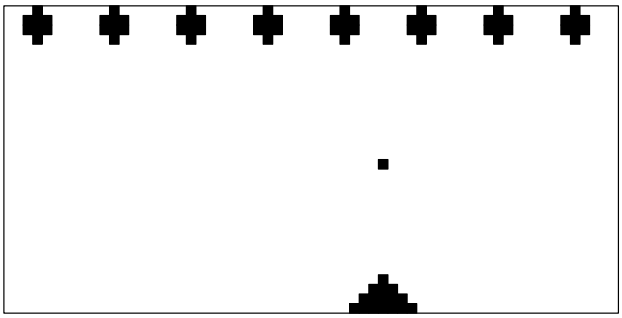
SYNOPSIS

chip8run missile

DESCRIPTION

You must shoot the 8 targets on the screen. Your shooter moves a little bit faster each time you shoot. You have 12 missiles to shoot all the targets, and you win 5 points per target shot.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

8: fire

COPYRIGHT

missile version 1.0

AUTHOR

David Winter <winter@worldnet.net>

NAME

pong − the original video game

SYNOPSIS

chip8run pong

DESCRIPTION

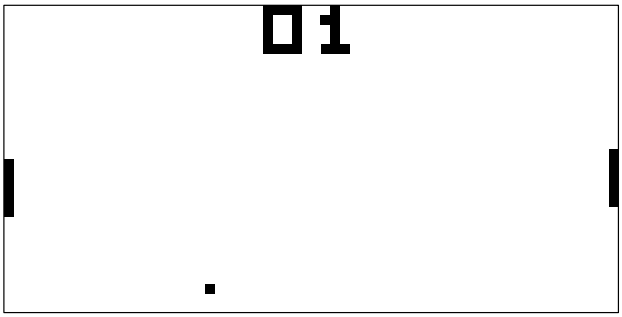
OK folks here it is! Pong for the Chip8.

The game never ends. It keeps score, but only up to 9 for each player, then it will roll over to 0. Sorry, it's the only way I could think of to do it. So, you have to play ''whoever gets to a number first, wins.''

It is kind of slow, but then there are two paddles and ball moving around all at once.

The player who got the last point gets the serve...

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

1: move left paddle up
4: move left paddle down

C: move right paddle up
D: move right paddle down

COPYRIGHT

Pong version 1.0
Copyright (C) 1990 Paul Vervalin

AUTHOR

Paul Vervalin <vervalin@austin.lockheed.com>

NAME

puzzle − X11 Chip8 game

SYNOPSIS

chip8run puzzle

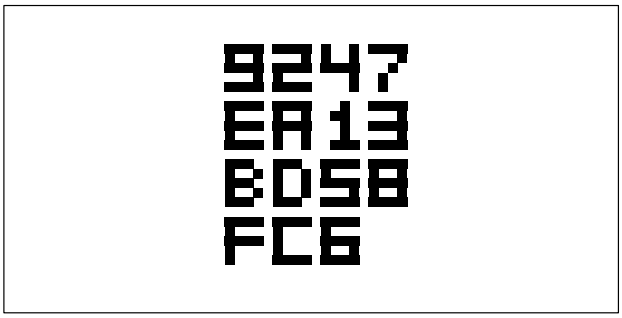
DESCRIPTION

Here is the infamous 15 puzzle for Chip8. The puzzle, as you doubtless recall, has 15 squares numbered 1 through 15 (in this case, 1 through F) and one hole. You can move the hole about and must put the pieces in order.

When you first run puzzle, it comes up solved. Thereafter, pressing IRestart causes the puzzle to have 32 random moves made, effectively randomizing the puzzle. If 32's not enough for you, just press IRestart again.

The program does not check to see if you've solved the puzzle, and therefore nothing special happens when you do except for the warm, fuzzy feeling that you have beaten it.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

You enter moves by using the keys. The key's position in that 4×4 matrix corresponds to the square in that position of the puzzle matrix. Pushing a key causes the hole to migrate to that position. The migration is performed in the order up, down, left, right; it is not necessary to limit your moves to those rows and columns containing the hole; you can request that the hole move to any position.

COPYRIGHT

AUTHOR

Roger Ivie <slsw2@cc.usu.edu>

NAME

race − driving game

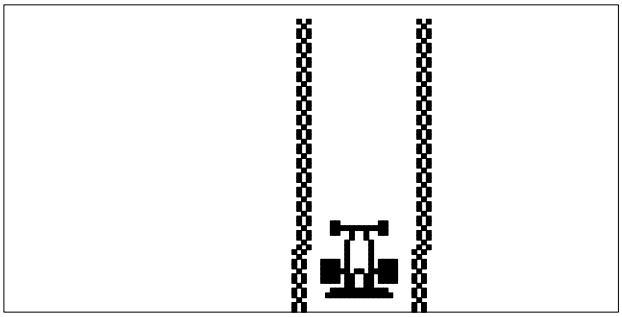
SYNOPSIS

chip8run race

DESCRIPTION

This game simulates a racing car.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

7: left
8: right

COPYRIGHT

race version 1.0
Copyright (C) 1991 David Winter

AUTHOR

David Winter <winter@worldnet.net>

NAME

Snake − navigate the snake through the maze

SYNOPSIS

chip8run snake

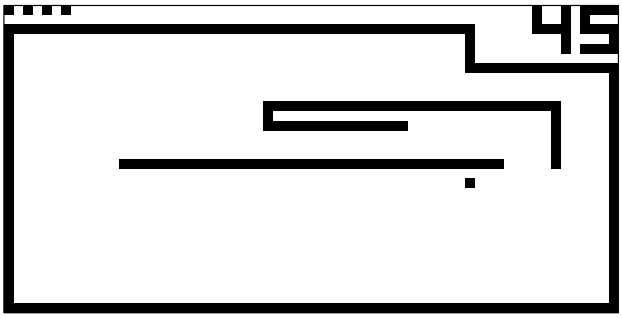
DESCRIPTION

In this game, you a guiding the snake to eat apples as they appear. The snake gets longer as it eats the apples. Each level has 5 apples, and then a small exit will appear.

If you bite the wall, or bite yourself, you lose a life.

The number of lives you have left is displayed in the top left corner.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: turn left
6: turn right

COPYRIGHT

chip8 version 1.1
Copyright (C) 1990, 1991, 1998, 1999, 2012 Peter Miller

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.

AUTHOR

Peter Miller E-Mail: pmiller@opensource.org.au
/\/* WWW: http://miller.emu.id.au/pmiller/

NAME

space fight − space invaders clone

SYNOPSIS

chip8run spacefight

DESCRIPTION

Ok, here’s a little Super-Chip game for all you mad S-Chip-freaks out there!

PURPOSE

The purpose of the game is.... *to win* (say what?)

In order to complete this quite difficult task your body (yes only your body and not your mind (this remark shows that I believe that the body merely serves as a tool for the brain)) have been warped through 87 different time-warps due to some foolish scientists who didn’t know what they were doing. This resulted in a rather big waste of resources but that’s of course another story. When you finally arrive you can’t recognize anything and two weird-looking dudes tell you that this is the year 2091!!! Frightening eh?

You are of course not very happy about your situation partly because you only have 25 dollars in your wallet and the prizes have gone up about 30000% but mainly because IBM and the fucking PC clones have completely taken over all of the computer-market resulting in CHIP8’s death!

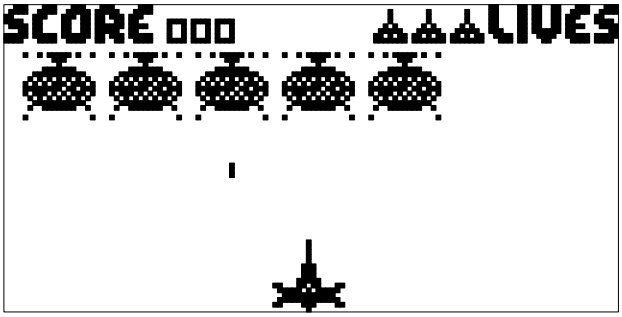
In a Holo-BiDirectional-FullDuplex-TV-store you see a newsflash concerning some evil aliens about to invade the good ol’ EARTH. Scientists think that they are being drawn towards the Earth due to the massive pollution the PC’s cause. However if people had bought CHIP8’s instead (which don’t pollute) they would have been no aliens now!

This is your call! You sneak into a Warp-Transporter without a ticket and you end up at NASA where you get to talk to the boss and you tell him: "OK you listen to me you scumbag! I wanna get into a spacecraft, get up there and KICK SOME ALIEN BUTT!". "But why?" the boss says. "Personal motives" you answer. If you can save the CHIP8’s existence you’re ready to do anything!

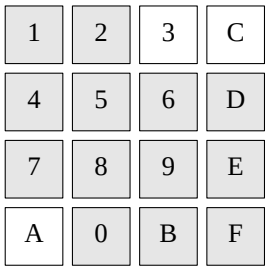
You get the newest spacecraft (CHIP8) and you take off!

Battle you way through the 6 (increasingly harder) levels!

The Display



The Keys



3: left
A: fire
C: right

COPYRIGHT

AUTHOR

Carsten Soerensen <u920659@daimi.aau.dk>

Carsten Soerensen
Kronhjortevej 4
DK-8270 Hoejbjerg
Denmark

NAME

syzygy − navigate the little snake

SYNOPSIS

chip8run syzygy

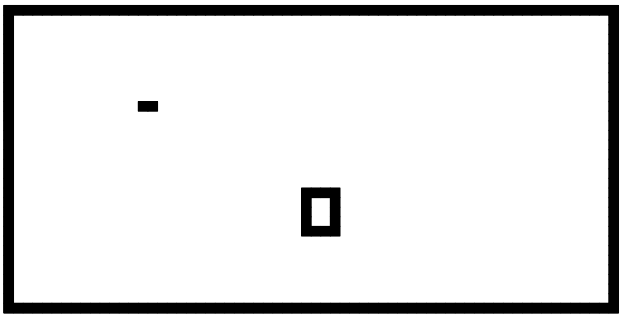
DESCRIPTION

One of the first games I remember playing on a computer was called "syzygy" on a now ancient TRSH80 Model 1. It has since appeared on other computers under various names. Why it was called syzygy, I have no idea (consult Websters). However, since the HP48SX has approximately the same memory, graphics and cpu power as my TRSH80 did (something like 16kB, 128x64, and a 1.2Mhz Z80), I thought it would be amusing to play it again. Now, approximately to my recollection, and with many apologies to the original author, here is a CHIP48 version of SYZYGY for the HP48. Enough drivel.

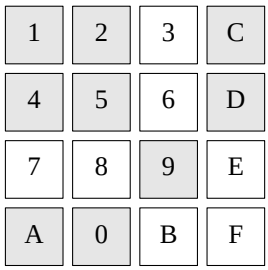
The object of the game is to seek out "targets". You do this with your syzygy. Initially small, the syzygy will grow by some amount each time a target is hit. Eventually, your syzygy will be so long as to make tougher and tougher to get any points (and easier and easier to get killed). Confused? Just try it.

Anyways, the syzygy is not allowed to run into anything except targets. It cannot run into the screen border (if present), or itself (this includes backing into itself). Fast and immediate death will result. Don’t worry if you die quickly a few times. The keys take a few minutes to get used to.

The Display



The Keys



3: up
6: down
7: left
8: right
B: Show Score
E: No Border
F: Border

COPYRIGHT

WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

AUTHOR

Roy Trevino <rtrevino@sedona.intel.com>

NAME

Tank − tank shoots bad guys

SYNOPSIS

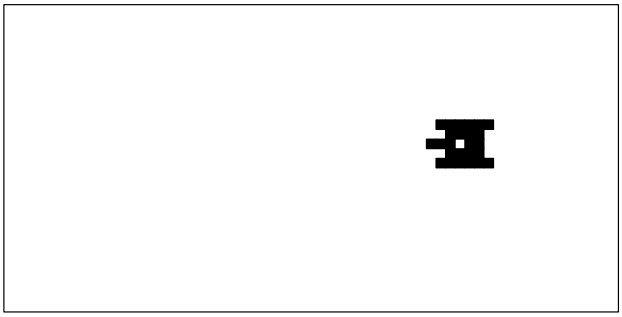
chip8run tank

DESCRIPTION

You have 20 missiles to shoot ad the bad guys.

This is one of the original Chip 8 games.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

2: up
4: left
5: fire
6: right
8: down

COPYRIGHT

Unknown 1977

AUTHOR

Unknown

NAME

tetris − drop the block

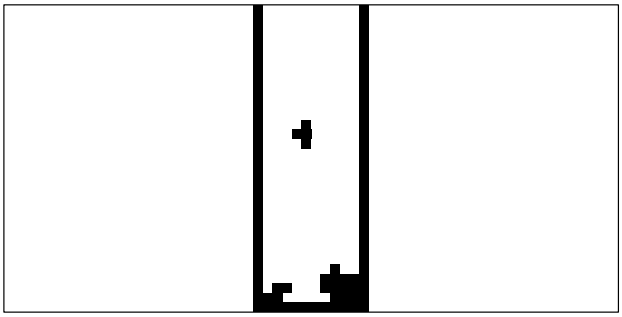
SYNOPSIS

chip8run tetris

DESCRIPTION

Guess what this game is... I’m sure you don’t need the rules. If you do, please ask your friends.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: rotate
5: left
6: right
7: drop piece

COPYRIGHT

Unknown

AUTHOR

Unknown

NAME

tictac − a tic-tac-toe game

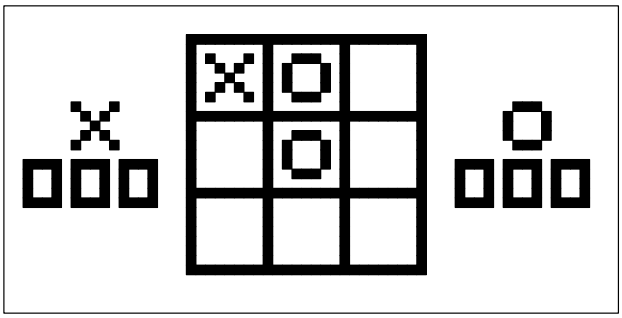
SYNOPSIS

chip8run tictac

DESCRIPTION

A tic-tac-toe game. Play with [1] to [9] keys. Each key corresponds to a square in the grid. The game never ends, so at any time, the winner is the one who has the best score.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

1-9: select corresponding square

COPYRIGHT

tictac version 1.0
Copyright (C) 1991 David Winter

AUTHOR

David Winter <winter@worldnet.net>

NAME

UBoat − Sink the submarines

SYNOPSIS

chip8run uboat

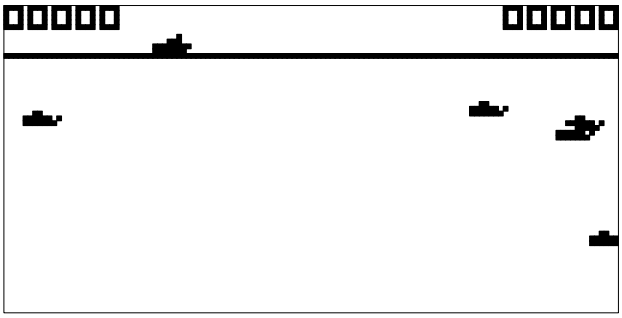
DESCRIPTION

This game is based on a game that I used to play back in the early 80’s on an Apple][. I believe the name of the program was "Depth Charge." Unfortunately I don’t know who was responsible for programming it.

The premise of the game is fairly simple. You are the Captain of a sub hunting destroyer, and your objective is to sink as many enemy U-Boats as you can. Your only weapon against the U-Boats is your ship’s supply of depth charges. You have an unlimited supply of charges but, due to the amount of time it takes your second rate crew to reload, you may only have four charges in the water at any given time.

The subs that you are hunting appear at random depths and have one of three random velocities (stopped, ahead half, or ahead full). There are six enemy subs visible at any one time. When one of these subs is destroyed, another will appear at a different location.

The Display



The Keys

The keys used are as follows:

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

7: ship’s speed full stop
8: ship’s speed ahead half
9: ship’s speed ahead full
E: drop depth charge
C: abort game

Scoring for the game is based on the depth of the sub, multiplied by one plus it’s velocity. For example: If a sub is at a depth of 50, and he is stopped then he is worth 50 points. A sub at the same depth of 50, but moving at 1/2 speed is worth 100 points. A sub at a depth of 50, but moving at full speed is worth 150 points.

The time limit for the game is about 2.5 minutes. At about 1:45 into the game a warning buzzer goes off and an almost empty hourglass is displayed by your score to let you know that the game is about over.

Once the game is over your score, the high score, the number of charges dropped, the number of subs hit, and your hit percentage are displayed.

A good game is about 2800 or so, and a really good game is around 3500. I haven’t had a whole lot of time to play it, but my high is 3551.

COPYRIGHT

(Sorry. I lightly edited it to adapt it to my assembler, and cope with the fact that X11 doesn't re-map the keys. *Peter Miller*)

AUTHOR

Michael D. Kemper <mikek@access.mountain.net>

NAME

UFO − space invaders clone

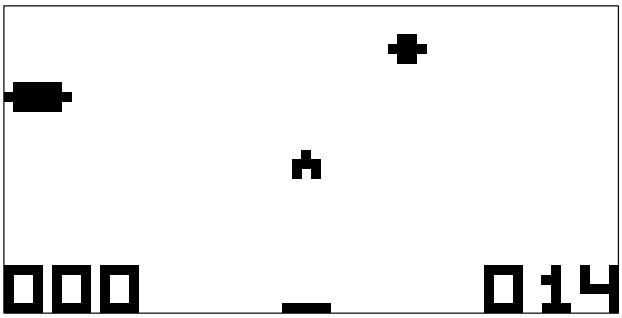
SYNOPSIS

chip8run ufo

DESCRIPTION

You have 15 missiles to shoot on the two types of invaders. The big one moves on the left and gives you 5 points. The small one moves on the right at variant speeds. The game ends after having shot the 15 missiles.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: shoot left
5: shoot up
6: shoot right

COPYRIGHT

Unknown 1977

AUTHOR

Unknown

NAME

VBrix − Knock out the bricks

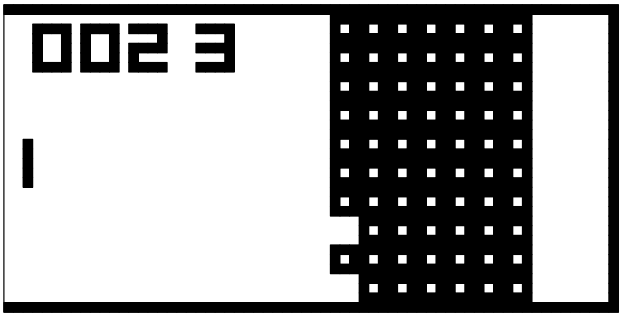
SYNOPSIS

chi8run vbrix

DESCRIPTION

This is a version of the classic game of knock out the bricks.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

1: move paddle up
4: move paddle down
7: start game

COPYRIGHT

AUTHOR

Paul Robson

NAME

vers − two tenuous worms

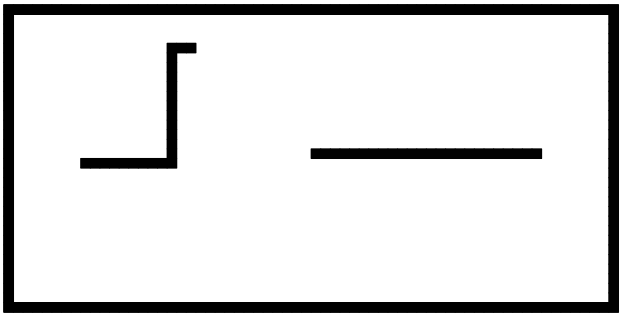
SYNOPSIS

chip8run vers

DESCRIPTION

There are two players, each controls a worm. The idea is to have the longest worm. First to 8 points, wins.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

1: Player 1 Left
1: Player 1 Right
7: Player 1 Up
A: Player 1 Down
B: Player 2 Left
C: Player 2 Up
D: Player 2 Down
F: Player 2 Right

COPYRIGHT

AUTHOR

JMN

NAME

Wipe Off − wipe off the bricks

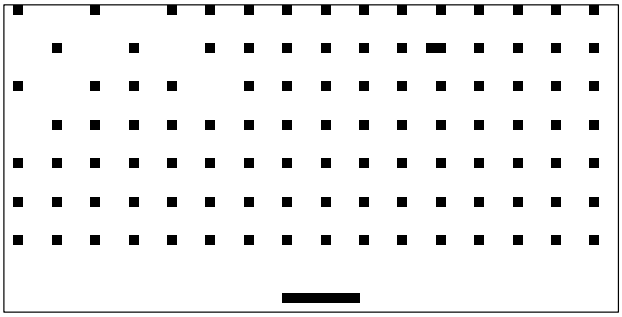
SYNOPSIS

chip8run wipeoff

DESCRIPTION

Another BRIX variant, but quite hard to play. Your score is shown when you lose all your lives.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

4: move paddle left
6: move paddle right

COPYRIGHT

Unknown

AUTHOR

Unknown

NAME

worm3 − navigate the worm to the apples

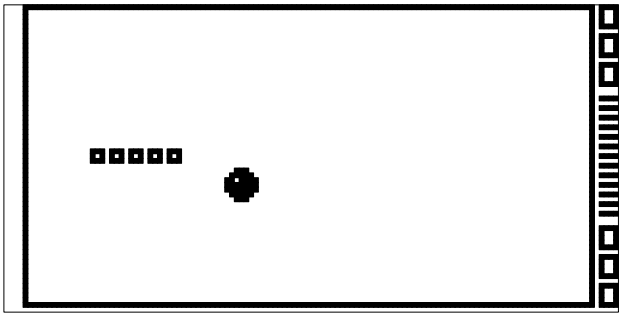
SYNOPSIS

chip8run worm3

DESCRIPTION

You control the worm. You need to guide it to each of the apples.

The Display



The Keys

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

8: left
9: right

COPYRIGHT

worm3 version 1.0
Copyright (C) 1991 Unknown

AUTHOR

Unknown