Core!

A Redcode development system for the Macintosh

Release Version 1.1
14 September 1991
by Jon Newman jonn@microsoft.com
with help from Robert Martin and Paul DuBois

Introduction

This document describes Core!, a Core War simulator application for the Macintosh computer. It should run on any Macintosh with 512K RAM or more. Later versions may support sound, hierarchical menus, and other extensions which may require recent versions of the System or ROMs. Core! is fully MultiFinder compatible and is System 7 friendly.

Core! is a SIMULATOR; it is entirely different from the old Core War programs from the early days of the Macintosh or from mainframe hacking. It is not necessary to disconnect your hard drive to run Core!! Core! is no more (or less) dangerous than other Macintosh applications.

Distribution Information

ShareWare fees:
Just Registration: US$14;
Registration Plus Latest Update: US$17, or US$14 plus stamped self-addressed disk mailer with 800K disk;
payable in check or money order to:
Jon Newman
1809 Imperial Ridge
Las Cruces, NM 88001 USA
(PLEASE NOTE: Core! is my own personal product, and is not in any way associated with Microsoft Corp. or its legal department. This is just my best SnailMail address…)
About ShareWare: Core! is a ShareWare product. For those unfamiliar with ShareWare, this means that the product is distributed freely throughout the Macintosh community, except that commercial distribution is prohibited. After a trial period using Core!, you are honor-bound to either send in the registration fee or erase Core! from your software collection. Feel free to give Core! to your friends! but be sure to tell them that Core! is not free, and that if they like and use Core! they too must send in their personal registration fee. If you have questions, contact me at the address above, or at jonn@microsoft.com.

Using This Documentation

Core! consists of:

- This document " Core! manual" for MacWrite 5.0
- The document "MARS Standard" for MacWrite 5.0
- The application " Core!"
- An assortment of fighter programs
- READ.ME (of course!)

If you already understand Core War, read the MENUS and WINDOWS sections to learn how to use this particular simulator. If you are unfamiliar with Core War, skip ahead to GENERAL INFORMATION and come back to MENUS and WINDOWS later. If you have questions about Core War syntax, program behavior, or the Core War extensions implemented in Core!, check the MARS Standard document.

2
MENUS

The File Menu

New Fighter: Creates an empty new Edit window. This window can be used to edit, load and save MARS fighter programs. The editor is a standard multi-window TextEdit-based editor.
Open Fighter: Loads a file into a new Edit window.
Close Fighter: Closes the frontmost Edit window.
Save Fighter: Saves the foremost Edit window to a file.
Save As…: Saves the foremost Edit window to a file whose name you provide.
Save A Copy As…: Saves a copy of the foremost Edit window to a file whose name you provide.
Revert: Reverts the foremost Edit window to the last saved version.
Quit: Quits Core!.
If you are already familiar with any Macintosh text editor, your instincts here are probably right.

The Edit Menu

Undo, Cut, Copy, Paste, Clear: Standard Clipboard commands. Undo is not implemented for Core! Edit windows, and is present only for those DAs which need it.

The Font Menu

Monaco, etc.: Changes the font of the foremost Edit window. Monaco is default. Tabs will look right only with monospaced fonts.

The Size Menu

9 Point, etc.: Changes the typesize of the foremost Edit window. 9 point is default.

The War Menu

Fight: Starts the MARS simulation. The timer is automatically reset when you restart the simulation after the timer has run out.
Step: Steps through one instruction execution, and prints a notification of that instruction to the Debug window.
Step-Step: Starts the MARS simulation, and for each instruction execution, prints a notification in the Debug window.
Halt: Halts the MARS simulation. Command-'.' will also stop the simulation.
Set Timer Val: Brings up a dialog which can change the timer setting and reset the timer.
Reset Timer: Restarts the timer at its previous setting.
XXX cycles of YYY left: Displays the maximum timer setting, and the amount of time left. This menu item is informational only and is always disabled.
Load "<fighter name>": Loads a fighter from the foremost Edit window into core; it only works if three or fewer fighters are active. In the subsequent dialog, you can select the side to load, the maximum number of processes for that side, and the location in which to load it. The default process count is same as for the last fighter to be loaded, or 64; the default load location is a random location guaranteed to be at least CoreSize/8 cells away from the initial location for any other active side. (The default load locations are generally adequate.) If the load is successful, the selected side becomes active; if not, check the Debug window to find out what line(s) is/are in error. Note that Load writes over cells in core whether or not it is successful, and can even write into addresses about to be executed by other sides; you should be careful about loading fighters into an ongoing battle, or about specifying load locations other than the default location. Press Command-'.' to abort the load process.
Load From File is like Load From Window, except that it loads directly from a file rather than from the forward Edit window. It will continue prompting you for more files to load until you hit "Cancel", a load fails, or all four sides are loaded.
Clear One deactivates one side, and clears all memory cells last "colored" by that side. You should be careful about clearing single sides from an ongoing battle.

Clear All deactivates all sides, resets core to its initial state, and resets the timer.
Start Tournament… starts a Core War tournament. You can select the timer setting, maximum process count / player, number players/battle, and the scoring, as well as the roster of fighters; however, if you wish to choose an alternate Core War standard, select **Change Standard** before selecting **Start Tournament**. A record of the tournament is stored into a file of your choice. You can continue to use the Core War application while a tournament is in progress, including the editor, range windows, breakpoints, etc.; some actions will necessarily halt an ongoing tournament, and you will be warned when you attempt such an action.
N-player tournaments with two players per battle require n(n-1) battles; three or four players/battle take n(n-1)(n-2) or n(n-1)(n-2)(n-3) battles, respectively. Tournaments are fastest with all windows closed, and with the **No Maintenance** option checked in the **Windows** menu. Do not change the files containing the fighter programs in the roster while a tournament is in progress; this will render the tournament results invalid, and the tournament may or may not fail.
Finally, the War menu contains a list of the sides, the names of the files loaded as that side (if any) and the maximum number of processes for that side.

The Standards Menu

Change Standard: Brings up a dialog in which you can change the number of memory cells in the simulated computer, the compatibility mode, and the Address Range and Descendant Count variations. These are explained in the MARS Standard document. You will automatically Clear All whenever changing any of these values.
Clear All Breakpoints clears all current breakpoints.
Set Static Breakpoints -- when this item is selected, SHIFT-CLICK[-DRAG] in the core window (or SHIFT-CLICK in a range window) will set static breakpoints in the selected core location(s). Static breakpoints remain in a fixed core location until core is cleared.
Set Copy Breakpoints -- Copy breakpoints are copied to the new location when the MOV instruction copies their core location.
Set Copy-Erase Breakpoints -- Copy-erase breakpoints are copied to the new location when the MOV instruction copies their core location, and are erased when another core location is copied onto them.
Clear Breakpoints -- when this item is selected, SHIFT-CLICK[-DRAG] in the core window (or SHIFT-CLICK in a range window) will clear breakpoints in the selected core location(s).
The Standards menu also contains a summary of the current Coresize, compatibility mode etc.

The Windows Menu

Show/Hide Debug brings the Debug window forward in case it is hidden or lost behind other windows or hides it if it is already forward.
Show/Hide Core brings the Core window forward in case it is hidden or lost behind other windows, or hides it if it is already forward.
Magnify Core doubles/halves the scale of the Core window; this can help relieve unnecessary eyestrain for big screen users. This menu item is checked when the window is magnified.
Show/Hide Battle Icons brings up/closes an informational window describing the shapes and meanings of the Core window graphics.
The next four options allow the real-time fight window graphics to be reduced or eliminated in order to improve performance. They allow the Core window graphics to fall temporarily behind, but the Core window will still be drawn correctly on updates (for example, if it is closed and reopened).
Show Executed displays the "executed" icon in a core location when a process executes the instruction in that location.
Show Executing displays the "executing" icon in a core location when a process is about to execute the instruction in that location.
Show Changed displays the "changed" icon in a core location when a process alters the instruction in that location.
Flash Changed flashes the "changed" icon in a core location off and on when a process alters the instruction in that location. This is useful for fast machines where just "show changed" does not clearly show when a core location is altered.

No Maintenance disables the four preceding options, halting all Core window graphics. Furthermore, it disables the maintenance of the data behind that window, so that even if the Core window is closed and reopened, the graphic data will still be out-of-date. This option gives the best performance.
Close All Range Windows closes all Range windows.
Show All Range Windows brings all Range windows forward.
Load All Fighters loads the fighters in all open Edit windows into core.
Close All Fighters closes all Edit windows, giving you the option to save those which are dirty.
Save All Fighters saves all "dirty" Edit windows.
Selecting an item from the list of Edit windows brings that Edit window forward. A diamond appears next to the entry for "dirty" Edit windows.

5
Windows

The Core Window

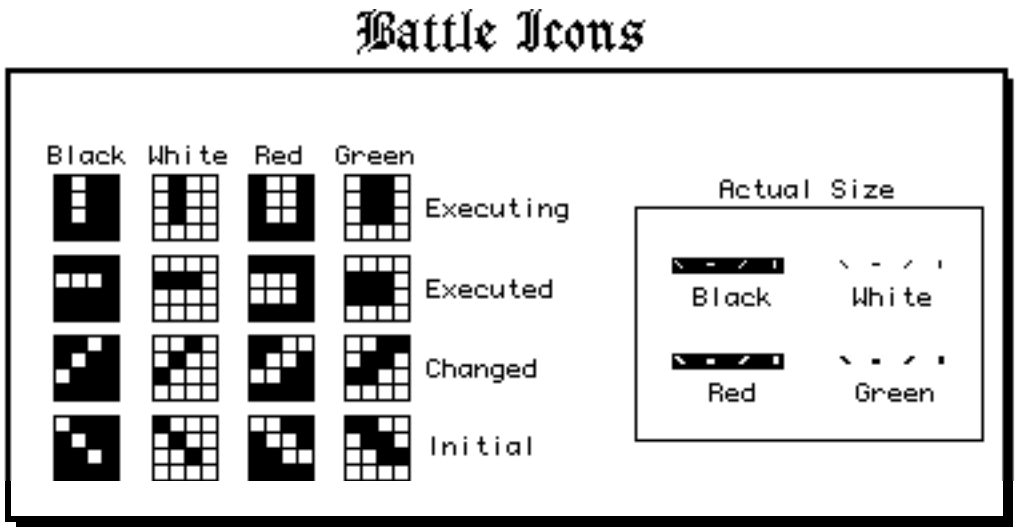
The first thing you see is the CORE window. If you are a small-screen user it is about all you will see, since it covers almost the entire standard Macintosh screen. This window graphically displays the entire core array, where each cell initially takes up a 4 X 4 array of pixels. There are initially 8192 cells, 128 cells wide by 64 high where cell 0 is in the upper-left-hand corner, cell 128 is in the upper-right-hand corner, and cell 8191 is in the lower-right-hand corner. Initially the window is near-white, with one dot in every cell; all memory locations contain DAT 0 0, and no fighter programs have been loaded (this is explained in the MARS Standard document). At the bottom of the Core window is the timer; this timer counts down with each simulated instruction execution, and the MARS simulation halts when it runs out. The title of the Core window shows whether and in what mode the simulation is running, and lists the active sides and (as space permits) the names of the loaded fighters.

You can do the following with the Core window:
-- change the total number of cells using the "Change Standard…" menu item;
-- change the graphic size of each cell (from 4 X 4 pixels to/from 8 X 8 pixels) using the "Magnify Core" menu item;
-- change the shape of the Core window using the grow box (this grow box is somewhat nonstandard);
-- enter a new timer setting by clicking on the timer box (or by choosing the "Set Timer Val…" menu item);
-- hide the Core window by clicking on the GoAway button, or by choosing the "Hide Core" menu item;
-- redisplay the Core window by choosing the "Show Core" menu item;
-- print the contents of up to 32 cells in a single row to the Debug window by clicking on one and dragging horizontally (see the section on the Debug window below);
-- print the contents of an unlimited range of cells to the Debug window by option-clicking on one end and dragging to the other (use command-'.' to break off this series); or
-- open a new Range window displaying a specified range of memory cells by shift-clicking or shift-option-clicking.

Note that clicking in the Core window contents will NOT bring the Core window forward; this is a violation of the Macintosh standard interface, but the violation is necessary since the Core window is so big and we don't want it to automatically cover the Debug window.

If there is any space left in the last row of cells, that space will be boxed off.

Memory cells look like this:



(this figure courtesy of Robert Martin) Cells are displayed as Executing when they are about to be executed, Executed when they were just executed, Changed when their contents were altered and Initial when they were just loaded or when a process just died in this location. Use Magnify Core from the Windows menu to double the dimensions of these cells (if your monitor is big enough to show all of them). On a color monitor, RED cells are red and GREEN cells are green; this will further help you distinguish between different players. You can view the above graphic in the program using the Show/Hide Battle Icons menu item.

Edit Windows

You can edit fighter programs using Edit windows, which are created using commands in the File menu. The editor is a standard TextEdit-based editor which supports multiple windows. If you wish, you can use any outside editor which supports TEXT files; however, the fighter programs may not be more than 32K characters. The editor supports tabs; it has a fixed tab spacing of 10 characters, and inserts tabs as multiple spaces which must be deleted individually.

The Debug Window

The Debug window displays information about the ongoing MARS simulation, including
-- program loading, program load error, program death, victory, and other battle status messages;
-- if you are Stepping or Step-Stepping through the battle, every instruction will be noted here as it is executed (see Range Windows for details on these informational lines);
-- If you click, shift-click-drag or shift-option-click-drag on the cells in the Core window, the contents of the selected cell(s) will be displayed in the Debug Window. (Press command-'.' to break off this series.)

The contents of the Debug window cannot be edited or erased, although the oldest lines will automatically be removed when the contents become excessively long. Display this window using the Show Debug menu item; hide it using the GoAway button or the Hide Debug menu item. The Debug information is maintained even when the window is hidden.

Range Windows

Range windows echo the information in the Core window in text format. Each line of a Range window give the address and the complete contents of the memory cell. In addition, if any processes are waiting to execute the instruction, they are displayed by player -- a lower-case letter indicates one process, and an upper-case letter indicates two or more processes waiting to execute this instruction. Thus, "b R " indicates one black process and two or more red processes.

The Battle Icons Window

This window contains the Battle Icons graphic shown above. It is for informational purposes only -- it doesn't do anything.

7
GENERAL INFORMATION

What Is Core War?

Core War is a "game" in which fighter programs vie for control of the MARS ("**M**emory **A**rray **R**edcode **S**imulation") simulated computer. The machine language supported by the Mars computer is called REDCODE, and both the machine language and associated assembly language are similar to those on real computers. Rival programs execute simultaneously in a shared address space under a tightly-defined model of multiprocessing, and attempt to defend themselves while seeking out and destroying other programs; the last program with a surviving process is declared the winner. The Core! application simulates the MARS computer and associated loader, editor and debugger.

How to use Core!

Before you start to write your own fighter programs, you will want to experiment with the example programs provided. Load them into Edit windows using **Open Fighter** in the **File** menu, and examine them. Then load them into Core using **Load "<fighter name>"** in the **War** menu, and step through the battle using **Step** from the **War** menu (or its command-key equivalent!) Show the Debug window with **Show Debug** from the **Windows** menu for more detail on each individual step, or open a Range window over a program to watch it execute in maximum detail. You will understand the MARS Standard document more easily once you have seen several programs in action.

History of Core War

Core War is derived from a game called Darwin, invented by Doug McIlroy and others around 1962. It was first featured in Software -- Practice and Experience in 1972, and adopted by A.K. Dewdney in his May '84 and May '85 "Computer Recreations" columns in Scientific American. (This according to Doug McIlroy in a posting to comp.risks.) Today, the International Core War Society (ICWS) has held three annual contests (they generally occur in the winter), publishes a quarterly newsletter, and has hundreds of members in many countries, with some of the strongest entries coming from Japan and the Soviet Union. Dewdney's REDCODE specification has been superseded by ICWS '86 for these contests. The version of REDCODE supported by Core! is called "Core Wars '88 -- a proposed standard," and was published in the Summer 1988 Core War Newsletter; it was designed by the ICWS Standards Focal headed by Thomas Gettys. This version has been adopted as the official standard, and will probably be used in future contests; it is at any rate largely compatible with earlier REDCODE standards. Core! supports both the '86 and '88 standards, along with some extensions; these standards and extensions are all described in the MARS Standard document.

For information about the ICWS, write
ICWS, Office of the Secretary
5712 Kern Drive
Huntington Beach CA 92649-4535

About Contests 1 and 2:

The '86 contest was dominated by MICE, a program which does nothing but self-replicate at an alarming rate. Some pundits thought the MICE could never be beat, but in the '87 contest, just about everyone beat MICE. The top contestants in 1987 were all cut from the same cookie cutter: typically they sprayed memory with SPL 0,0 bombs, then cleaned up the disabled opponents with DAT bombs. Hardly any of these programs did any self-replication, preferring to stay small and avoid being hit. (One such program was my own HITHARD2, which reached the top fifteen.) Any of them could defeat any other with a lucky core placement.
The '88 contest was won by COWBOY, an entry from the Soviet Union.
The '89 contest was held at the Artificial Life conference in February, in Santa Fe, NM. Results should become available in May-June 1990.
The '87 and subsequent contests are held in round-robin style, with wins worth 3 points, ties 1 and losses 0.

What was Core War?

In the early days of Macintosh Hacking (and earlier on some other systems), Core War was very different than it is today. These old versions of Core War single-stepped real processes in the real computer memory. This was, of course, very dangerous, since fighter programs could step on device drivers, PRAMs, and just about anything else; it was necessary to eject disks and disconnect hard drives before starting this sort of Core War battle. DO NOT CONFUSE OLD-STYLE CORE WAR WITH Core!! Core! is strictly a simulation of this sort of battle, and is no more (or less) dangerous than other Macintosh applications.

The Competition to Core!

Core! is based on an earlier program written by Robert Martin. Most of the interface and some of the actual source code has been blatantly stolen from him. I purchased a copy of the source code from him and use it here with his permission; a portion of your ShareWare fees is forwarded to him. Robert Martin is also the author of Pharaoh, an excellent pyramid-building game which is sort of a cross between Santa Paravia and Imhotep, for fellow fans of those oldies. Contact Robert Martin at 12310 Lee Ave., Waukegan IL 60085 for further information on these Macintosh products.

The Douglas McDaniels Core War simulator is an excellent Core War simulator. This PC program is advertised as clocking 2000 cycles per second on a mere 8MHz AT (compared to about 40 cycles per second for Core!, on an SE and running two fighters); I have never timed it, but I can tell you, this program books. As McDaniels puts it, "the kernel code is written is assembly language for the 8086 and furthermore … makes highly eccentric use of the segmented architecture." I'll bet; I wonder if it runs on a PS/2? Like Core!, it has a tournament mode to set up round-robin tournaments to execute automatically. One serious limitation is that the McDaniels simulator only allows 64 processes per side, which can change the behavior of some programs. Also, the graphics don't match up to Core!, nor do the debugging facilities -- I wouldn't want to write fighter programs using the McDaniels simulator -- but it's certainly worth having around to test your new creations against earlier contest winners, provided you have access to a PC with Hercules- or CGA-compatible graphics adapter. All this for ten bucks! Contact Douglas McDaniels, 2627 Arlington Dr. #101, Alexandria VA 22306.

The PC Core War Colosseum is the "standard" simulator since it is the tournament simulator for ICWS tournaments; this is unsurprising, since its author is the director of the ICWS. It requires CGA or software simulating CGA; I have never used it. Cost is $24.95. Contact AMRAN-Software, 5712 Kern Drive, Huntington Beach CA 92649-4535.

A mainframe simulator (for UNIX?) is available from: Na Choon Piaw, P.O. Box 4067, Berkeley, CA 94704-0067, piaw@cory.Berkeley.edu.

I have heard of other simulators, including the AMIGA MADgic CORE (The MADgic CORE, 8450 Cambridge #1172, Houston TX 77054-3120), and the mainframe Mass Compare Program which determines the results of every possible contest between two fighters for every possible initial separation between them (Kenneth W. Clapp and Robert R. Reed, no address sorry). I have only used the Robert Martin and Douglas McDaniels simulators myself.

Acknowledgements

Core! is based on the Core War simulator by Robert Martin. Most of the interface and some of the actual source code has been blatantly stolen from him. I have taken this program ShareWare with his approval, and he receives a $4 share for every copy "sold". Does anyone out there have a current address (of any kind) for Robert Martin?

Core! is also based on **TransSkel**, a public-domain program shell. TransSkel handles the mundane tasks of routing events to the appropriate window, calling update handlers, and so on. Core! also contains the public-domain TransSkel modules **TransEdit** and **TransDisplay**, which handle the editor and debug window and associated menus. It would have been far more difficult to write Core! without the use of these modules, and it probably would have been far buggier than it is; I recommend their use as an "extended Toolbox" to anyone writing any halfway-normal Macintosh program. Versions of these modules exist for Pascal as well as Lightspeed C™. If you are interested in using **TransSkel**, **TransEdit** or **TransDisplay**, contact

Paul DuBois
Wisconsin Regional Primate Research Center
1220 Capitol Court
Madison, WI 53715-1299 USA

or send electronic mail to
UUCP: {harvard,rutgers,ucbvax}!uwvax! rhesus!dubois
Internet: dubois@primate.wisc.edu

Alternately, contact me and I can send you a copy of the C modules and documentation.

Thanks also to Dan Bornstein for the "Core!" name and to Hjalamar Dijkstra for the Core! icons.

The version of Core! described in this document was created using LightspeedC version 5.0. LightspeedC is a trademark of:

THINK Technologies, Inc.
420 Bedford Street Suite 350
Lexington, MA 02173 USA

Disclaimer: My employers at Microsoft don't know anything about this project and don't want to. Let's keep it our little secret, shall we?
Backup Disclaimer: If you think Microsoft product support will help you with Core! questions, please contact your local mental health authorities. Contact me via Email, or, if necessary, at the SnailMail address below.

Jon Newman
1809 Imperial Ridge
Las Cruces, NM 88001
jonn@microsoft.com
Fnord