THE CORE WAR NEWSLETTER

The Official Newsletter of the International Core War Society

Features

Articles

Columns

Volume 5, Number 2 **Spring 1992** **An ICWS Publication**

The Past, Present, and Future of the ICWS

by Mark A. Durham

This article began as "The Future of the ICWS", but it became increasingly clearer that in order to understand our future, we must understand our present[†] and our past as well. It also became clear that many people are not fully aware of what the International Core War Society is or how it got to be this way.

It has been over eight years since the birth of Core War. The original definition was "Core War Guidelines" by D.□G.□Jones and A.□K.□Dewdney of the Department of Computer Science at the University of Western Ontario in March of 1984. The game was first introduced to a large audience when it was described by Dewdney in his "Computer Recreations" department of *Scientific American* in May of the same year.

Graeme McRae immediately set about writing a standard for Core War, submitting it to Dewdney and Jones for commentary. Two years later, a much revised version of McRae's document would be adopted as the ICWS'86 standard by an ad hoc committee headed by Mark Clarkson, Director of the ICWS.

Clarkson volunteered for and had been picked by Dewdney to serve as Director. There was, and is, no official job description for the Director of the ICWS. Presumably, the Director coordinates all things Core War. There is no limitation of term. There is no specification on how a new Director is chosen.

Under Clarkson, ICWS tournaments were held in 1986 and 1987. At the end of 1987, the Directorship changed hands. William□Buckley, then Editor of the newly formed *The Core War Newsletter* (then a private enterprise), was named Director by a "coordination committee" consisting mainly of Dewdney, Clarkson, and Buckley. Other positions such as Secretary, Treasurer, and Standards Focal were newly formed at this time as well.

The last innovation of 1987 was the Branch Section. There is a document, called "Guidelines for Establishing a Local Core Wars Section", which defines how to form a Branch Section. The only Branch Section in 1987 was one formed in Japan.

[†] This article is a year old, and as such does not address the present situation of the ICWS or *TCWN* at all. I thought the article warranted inclusion anyway. For information about the present, see Jon□Newman's article.

1988 brought with it a new proposed standard based on the work of Standards Focal Thomas Gettys. It would later be adopted by the sharply divided vote of fewer than ten people (despite the vote being open to all ICWS members) as the ICWS'88 Core War Standard, the standard we operate under today.

Unfortunately, many people decided not to become members of the ICWS. Instead, they wrote Core War systems and Redcode programs from the "Computer Recreations" articles in *Scientific American* and *The Armchair Universe*. Their non-compatible work can be found on networks and on Bulletin Board Systems all over the world.

Which brings us to today. The ICWS consists of only three documents: ICWS'86 (voted on by an ad hoc committee), ICWS'88 (voted on by a handful of ICWS members), and the Branch Section Guidelines (established by Mark Clarkson). Leadership is without an official written basis or popular vote.

After last year's tournament, several of the ICWS members with access to electronic mail started a discussion about the present and future of the ICWS. It forms the basis for this article. They developed several questions about the ICWS and presented them to then Director, Mr. Buckley. Here are some of those questions and the answers given by Mr. Buckley (paraphrased).

Question: *What do you see as your role as Director of the ICWS?*

Buckley: *In general, the ICWS is a business, as required by contemporary tax law. The ICWS is a sole proprietorship, with myself as sole proprietor. The ICWS remains a private business.*

Question: *Why is ICWS membership (and/or TCWN and/or catalogs) so expensive?*
Buckley: *Over one half of all funds collected by the ICWS have been used for ICWS mail only. The TCWN printing fee exceeds the other half of ICWS receipts. One more point about funding. It is the only way that I can continue to help people in foreign lands who do not have the funding of Western, industrialized, and relatively wealthy people. I use the largess of occidentals to promote Core War in the lands of, shall we say, infidels. If people do not like that, they need not join.*

I have generated a rough total of all ICWS expenses since my accession to the directorship. I have taken in about $6000.00US but have spent about $9000.00US; this means that I have subsidised the ICWS to the tune of some three thousand dollars. I do not think, given my contribution, that a fee of $25.00US is too high. Quite the contrary, it appears a bit low to me.

Question: *When will there be a new Director, and how will this new Director be chosen?*

Buckley: *If the Branch Sections could act as a basis for the global control of Core War gaming, a Directorship is not so important. I will hold on to my position, until such time as the collective Branch Sections can demonstrate their ability to assume the resonsibilities of the Director, and thus effectively operate the ICWS without a Director. I envision that this should take an economic foundation, a council of Branch Sections where each Branch Section is represented (much in the fashion of the Roman Senate) and a permanent staff to maintain the records.*

Question: *How many ICWS members are there?*

Buckley: *I finally have a complete database of Core War enthusiasts, basically all of the people who have shown an interest in Core War. The database contains about 2000 names, but only one-fifth of the names I have are listed therein. The database should be ten thousand records long. Only about 300 of the listed persons sent money.*

Clearly, the future of the ICWS is in the hands of the Branch Sections. How do you form a Branch Section? The following rules are taken from "Guidelines For Establishing A Local Core Wars Section".

Submit a written petition to the Director of the ICWS informing him of your desire to form a Branch Section. Then establish rules by which persons may become members. The rules may include membership fees (or lack thereof) and member rights. Send a copy of the rules to the Director.

If or when there are no conflicts between your rules and "established ICWS principles and bylaws", a Certificate of Recognition will be issued. The ICWS currently does not impose any association fees on Branch Sections.

See **Past**, Continued on Page 7

A Letter From the New Director

by Jon Newman

Greetings! My name is Jon Newman, and I am the new Director of the ICWS. On February 24, 1992, William Buckley and I concluded an agreement which transfers control of the ICWS, and makes possible the resumption of the Newsletter and the annual tournament. With your help, I hope to achieve far more than that and expand the ICWS in new directions which will interest any fan of Core War.

Before discussing my new plans and projects, I should make a few pledges:

- Although my agreement with Mr. Buckley does not include compensation for the balance of current memberships, all current ICWS memberships will be honored.

- From now on, *The Core War Newsletter* will be published four times a year. *TCWN* was always supposed to have been a quarterly newsletter, but according to my records, only seven issues have appeared in the last four years. This is not acceptable. If there are only four pages of material in any quarter, you'll get a four-page *TCWN*; but you will get it, on time.

- Similarly, the annual Tournament will be held once a year, in the winter, at a time and under conditions announced well in advance.

- The ICWS's working relationship with the Branch Sections will continue under the same guidelines as before, subject to change only by mutual agreement between the ICWS and the Branch Sections.

- Effective now, the ICWS annual membership fee for all new and renewed memberships is reduced to US$15. This includes four issues of *The Core War Newsletter*, a copy of the ICWS '88 standard (if none has yet been delivered), and a single entry in the annual tournament. During this first year, I will monitor income and costs to determine if this fee structure is sufficient to cover the organizational costs, and whether it can be further reduced. My goal is to exactly even out the income and expenditures of the ICWS.

- ICWS members who have not yet received a printed or electronic copy of the ICWS '88 Standard, will receive one.

I am pleased to announce that Mark Durham has agreed to stay on as *TCWN* Editor. In fact, his role has expanded, to include printing and mailing the quarterly newsletter. Mark's help will be invaluable in producing a quality *TCWN*, and producing it on time. Mark has done wonderful work on *TCWN*, and I look forward to working with him.

What follows is a list of future projects for the ICWS. Mark and I can't do all this by ourselves. The members of the ICWS must help out for these more ambitious goals to come together! Please, if you — yes, I mean you! — see anything here which interests you, get in touch with me.

Core War Simulator Certification

We need a Core War validation suite and certification program (by which I mean an ongoing project, rather than a computer program) to ensure that Core War simulators conform to the current ICWS Standard. Simulators which pass the test could then be advertised as "ICWS '88 compliant." Mark Durham has made an excellent start toward this goal, but more work is needed. More on this appears elsewhere in this issue of *TCWN*.

Electronic Data Transfer

Virtually all Core War *aficionados* have access to a computer. So why does the ICWS do its business solely on paper? If there is enough interest, the ICWS can provide many common documents on electronic media. If enough requests come in for a specific document on a specific medium, we can provide it. Do you want the next tournament fighter roster on DOS 720K floppy? The ICWS '88 Standard on Mac 800K floppy? It can happen if you want it.

New Core War Standards and Extensions

The ICWS '88 standards have served long and well, but we should always be thinking of improvements. The last truly mobile Tournament winner was Mice. IMHO (In My Humble Opinion), mobility and reproduction are at the heart of Core War, and the Standard should encourage fighter programs with these properties. Many individual simulators have implemented one or another extension. We should standardize the interpretation of these extensions, experiment with them, and consider instituting a new Standard with those extensions which seem to encourage the most interesting fighter programs.

Other Programming Games

Core War is not the only programmer's game. Many Core War programmers also experiment with XTANK, C–Robots, and other "program the robot" games, as well as more esoteric pursuits such as the artificial life system Tierra. The ICWS should establish ties with groups sponsoring such systems, and *TCWN* should run reviews of simulators for these related pursuits.

The best way to contact me is via electronic mail, at JONN@MICROSOFT.COM. (Microsoft sometimes has mailer trouble; if your electronic mail is not answered within a week, please resubmit your mail.) I can also be reached at:

Jon Newman
13824 NE 87th Street
Redmond, WA 98052-1959
USA

If you can send a stamped self-addressed envelope with postal mail, this will help me answer as soon as possible. Please remember that the Director of the ICWS gets quite a lot of mail; this little courtesy will be very helpful.

Finally, please note that Microsoft has no connection with the ICWS. In fact, they don't know about any of this, and they probably wouldn't approve. Microsoft takes no responsibility or liability for any activities of the ICWS. (You'll be seeing a lot of these disclaimers, you might as well get used to them.)

Jon Newman

Past continued from Page 4

Branch Sections are required to promote the use of programs consistent with standards approved by the ICWS. Branch Sections must conduct tournaments in a timely, periodic fashion. No more than ten winners may be submitted by the Branch Section to the ICWS annual tournament. Other than this, what your Branch Section is or does is entirely up to you.

In keeping with the spirit of the above and the ICWS, a new Branch Section has been formed and is seeking official recognition. It is the Electronic Branch Section (EBS). Anyone interested in joining the EBS is welcome to do so. The only requirement is that you have access to electronic mail. There are absolutely no fees directly associated with the EBS. All inquiries about the EBS should be posted to the USENET newsgroup rec.games.corewar.

The MADgic Corner

by Mark A. Durham

First of all, I wish to apologize to the readers of *The Core War Newsletter*. No, you did not lose a year's worth of issues of *TCWN*. They were not delivered to your neighbor next door who keeps your mis-directed mail in hopes of one day black-mailing you with it.

There were no Spring, Summer, or Fall 1991 issues and there was no Winter 1992 issue. Your subscriptions will be appropriately adjusted, of course.

The lack of a year's worth of issues is two-fold. The first problem is the lack of articles and news. *TCWN* NEEDS YOU! Please feel free to submit articles, letters, solutions to the Quarterly Challenge, jokes, cartoons, or whatever you feel like. Send them to

> Mark A. Durham
> P.O. Box 301173
> Houston, TX 77230-1173

or in care of the ICWS. Without you, there can be no newsletter.

The second problem has been the transfer of *TCWN* along with the ICWS. Because that transfer is now complete, *TCWN*s will be arriving on a regular schedule. If you do not submit items of interest though, the following issues may be very thin!

Important news items from this last year include the formation of the Electronic Branch Section. The EBS has established (with the assistance of administrator Jon Blow) an anonymous ftp site for archiving and distributing Core War documents, Redcode programs, and Core War systems. The site is

```
soda.berkeley.edu
(128.32.131.179)
```

and the Core War directories are found in `/pub/corewar`.

The directory "`/standards`" contains the original Jones and Dewdney Guidelines, the ICWS'86 standard, and the ICWS'88 standard (made available with the assistance of the ICWS). The directory "`/redcode`" contains previous tournament entries and Redcode programs of interest. The directory "`/systems`" contains public domain and shareware Core War systems, including one for X-Windows systems. Finally, there is a directory "`/incoming`" for programs you may want to make available to the Core War community.

The EBS was also the main force behind starting the new USENET newsgroup

```
rec.games.corewar
```

This newsgroup is now where the EBS does most of its business. Articles posted to r.g.cw are archived on soda for posterity and for those who may have missed them when they were first posted.

Also of interest is “King Of The Hill”. KotH is an ongoing Core War tournament available to anyone with EMail and is provided by William Shubert <wms@iwarp.intel.com>. You enter by submitting a Redcode program with the special comment

```
;redcode
```

at the beginning. More information can be had by sending EMail (without the special comment) to Mr. Shubert or by inquiring about KotH on the r.g.cw newsgroup.

In this edition of The MADgic Corner I would like to take up the topic of bombing strategies. Almost every Redcode program, from the simplest to the most complex, uses some bombing strategy. For instance, IMP “bombs” the next location every cycle. DWARF bombs every fourth location every third cycle. IMP reduces *free space* (the maximum contiguous memory untouched by IMP, initially memory size M minus one) by one each cycle, eventually reducing the free space to zero after M cycles. An opposing program either needs to move itself an average of once every M cycles (or better) or defeat IMP quickly. Although the former is not too difficult, the latter is a much easier and a more desirable solution.

DWARF reduces free space by four every third cycle. This is obviously faster than IMP. There is a trade-off though in that any program three instructions long (or fewer) can remain in place indefinitely if properly positioned (just as DWARF itself does). Free space is never reduced below three. Also, fully three-fourths of memory remains unbombed.

DWARF's speed can be improved at the expense of the *free space limit* (the largest contiguous block of memory after free space is no longer decreasing) and the total unbombed memory. A DWARF-like program could bomb every eighth location every third cycle and have a free space limit of seven. There is the additional trade-off that the total unbombed memory of DWARF(8) is seven-eighths of memory which is larger than the total unbombed memory of DWARF(4).

Taking this argument to its limit, a DWARF(M/2) program immediately cuts free space in half, but has a free space limit of M/2 and leaves almost every memory location unbombed. If we think of the two unbombed halves of memory as new memories to be subdivided, we arrive at a divide and conquer strategy which is theoretically very efficient at reducing both free space and total unbombed memory with a potential free space limit of zero. [See Problem 1 of Quarterly Challenge #5].

Many programs use a similar but less complicated approach, moving the program with the bomb and using a constant step-size which is not a divisor (or multiple of a divisor) of memory size. These programs also reduce free space quickly and efficiently, and have a free space limit of zero. The most efficient step size depends on the memory size and the average opponents' program size. So far, choosing the best step-size is a combination of good guessing and trial and error.

There are almost as many bombing strategies as there are Redcode programmers, but I will leave you with one final bombing strategy for a fixed program. [See Problem 2 of Quarterly Challenge #5]. Consider a program, DWARF(random), which bombs a different random location of memory every time it bombs. Such a program also reduces both free space and total unbombed memory efficiently. The program can also be written such that it has a free space limit of zero. Unfortunately, the difficulties of multiplication and division as well as producing random numbers in Redcode can remove the efficiencies of the various bombing strategies.

There is an aspect of bombing strategy which I have failed to address in the previous paragraphs. That aspect is what kind of bomb is used. IMP bombs memory with itself and DWARF (in its original form) uses DAT 0 bombs. A popular (and efficient) bomb used in previous ICWS tournaments has been the combination of SPL 0 followed by DAT 0 so bombed memory looks like this:

```
SPL  0
DAT  0
SPL  0
DAT  0
SPL  0
```

Under ICWS'86 rules, this was a very effective killer. The bombed program would quickly SPLit off a total of 64 SPL 0 tasks, after which they would all commit suicide on the next DAT□0.

See **MADgic**, Continued on Page 14

Multi-Tasking Core Wars

by Vinay Gupta

I would like to suggest an alternative to the third of Jon Newman's Proposals for Future Core Wars Standards (the Descendant Count Modification).

When a procedure SPLits off to a new location, it should specify what proportion of processing time will be devoted to the child routine, with the remainder remaining with the parent. For example (assuming that the B-operand represents the percentage devoted to child process)

```
SPL IMPSTOMP 50
```

as the first instruction in a program would ensure that the IMPSTOMP routine will always execute on every second cycle, regardless of how many other processes the main program spawns. This allows a fair approximation of multi-tasking, opening up many new possibilities to battle programmers.

It should be noted that this means

```
SPL ANYWHERE 100
```

is now equivalent to `JMP`.

The problem with this suggested modification is implementing it in a MARS. Scheduling the processes in a deterministic, non-random way is a tricky problem, and probably best avoided (although if done correctly, it could incorporate the Descendant Count Modification). A stochastic scheduling system is much easier to implement, but may slow up MARS operation unacceptably (as well as adding an element of randomness to Core Wars, perhaps also unacceptable).

Implementing The System

Assuming that random scheduling is OK, I would suggest a variant of Jon Newman's tree structure, with each instruction cycle “trickling” in from the top of the tree, and randomly deciding which branch to follow on the basis of the SPL percentage.

This is tricky to explain, so I will try an example. Each time this multitasking warrior (let us call it **Juggle**) gets to carry out an instruction, it starts at **Root**, the main body of the program.

Process ID	Child ID	%Split	TPT% Left
Root	**ImpStomp**	50%	50% Total Processing Time
	OtherStuff	20%	40% (10% tpt to **OtherStuff**)
		total of	40% tpt executing **Root** code
ImpStomp		total of	10% tpt executing **ImpStomp**
OtherStuff	**Attack**	50%	5% (5% tpt to **Attack**)
	Copyist	60%	2% (3% tpt to **Copyist**)
	Defend	100%	0% (last 2% tpt to **Defend**)
		total of	0% tpt executing **OtherStuff**
Attack		total of	5% tpt executing **Attack**
Copyist		total of	3% tpt executing **Copyist**
Defend		total of	2% tpt executing **Defend**

Root first SPLits off **ImpStomp** at 50%, so half the time it jumps to **ImpStomp**'s entry (**ImpStomp** has no child routines, so on average it gets half of all available processing time). **Juggle** next splits off **OtherStuff**, at 20%. This means that if **ImpStomp** was not executed, there is a 20% chance **OtherStuff** will be executed (10% of total processing time on average). If **OtherStuff** was not executed, the next instruction in the body of the program will be carried out - otherwise the scheduling moves down to **OtherStuff**'s entry, and the whole process begins again.

If the code at **Root** is killed, what happens? I would suggest the most consistent option is to reassign the 40% of absolute processing time between **ImpStomp** and **OtherStuff**, at the existing ratio. So the new **Root** entry becomes:

Process ID	Child ID	%Split	TPT% Left
Root	**ImpStomp**	83%	83% Total Processing Time
	OtherStuff	100%	17% tpt

This form of scheduling imparts much resilience to programs.

For example, even if **ImpStomp** is subverted with nasty `SPL 0 30` instructions, the rest of **Juggle** will continue to function at its normal speed (the `SPL 0 0` bomb is still better than a `DAT 0` bomb because killing **ImpStomp** redistributes its resources among the other parts of **Juggle**).

A rather cunning variation would be to reassign any instructions which would execute killed code to the other warrior, giving a scheduling tree like this:

Process ID	Child ID	%Split	TPT% Left
Root	**ImpStomp**	50%	50% Total Processing Time
	OtherStuff	20%	40% tpt (10% tpt to **OtherStuff**)
	Enemy	100%	40% tpt executing **Enemy**

These instruction cycles could later be stolen back by terminating the code which used them. This option is quite 'realistic', and opens up all kinds of fascinating possibilities. For instance, if a routine is killed by the enemy, the only way to "disinfect" your program (and steal back the cycles) would be to slay the parent process of the subverted code.

All this does mean that the MARS may spend a long time just working out which side gets the next instruction rather than executing code. Clearly an option worth experimenting with however.

The only other obvious 'tweak' to add to all this is an instruction for sharing the time given to a process equally among its parts. I would suggest that when the SHaRe (`SHR`) instruction is executed, the MARS juggles the %SPLit figures to ensure that all children of the executing process each receive an equal share of processor time (leaving no time left over, `SHR` would be an alternative to SPL□CHILD□100).

Process ID	Child ID	%Split	TPT% Left
Root	**First**	50%	50% Total Processing Time
	Second	50%	25%
	Third	50%	12.5%

would after a SHR become

Process ID	Child ID	%Split	TPT% Left
Root	**First**	33%	33% (67% tpt remains)
	Second	50%	33% (half of 67%)
	Third	100%	33% (the remainder).

Comments

I certainly support Jon Newman's suggestion about the Address Space Modification. It sounds like an excellent way of encouraging a more interesting combat style, though being able to access 25% of the core from any single location seems rather much to me. I would favor a more finely divided core (considering CORESIZE=32000 is not all that unusual).

I should make it very clear that these suggestions are, for the most part, untested.

Mr. Gupta can be reached at 3 Moncrieff Way, Livingston, West Lothian, EH54 8LL, Scotland. In addition to playing Core War, Mr. Gupta runs "Tek", a cyberpunk RPG (Role Playing Game) and Science Fiction magazine. Write to him at the above address for details.

MADgic, Continued from Page 10

Unfortunately (or fortunately, depending on your point of view), this stratagem no longer works under ICWS'88 rules. The recently bombed program executes a SPL 0, spawning a new task, but the new task does not SPLit another new task until the previous task has committed suicide. Thus, the number of tasks executing for the bombed-out program does not increase to the limit, nor does it ever decrease to zero.

One "fix" for this change is to bomb with a combination of SPL 0 and JMP -1 as below:

```
SPL  0
JMP -1
SPL  0
JMP -1
SPL  0
```

This causes the number of tasks to increase to the limit. Sometime afterward, it is necessary to bomb every other location with DAT 0 to finish the program off.

Quarterly Challenge #5

The challenge for this season deals with two different bombing strategies discussed in The MADgic Corner.

Problem 1. Write a Redcode program which recursively SPLits itself halfway into free space. The program should copy itself to M/2 and then SPLit that task. Afterward, it should copy itself to M/4 (the copy should copy to 3M/4) and then SPLit that task. The process should continue until all tasks have been allocated or free space is zero.

Problem 2. Write a fixed Redcode program which bombs memory randomly.

You may restrict yourself to a memory size of 8192 if you wish. The best solutions will be printed in an upcoming issue of *TCWN*. In the case of Problem 2, best will be determined by speed, "randomness", and the free space limit.

The solutions to QCs #3 and #4 are the growth rates for Programs A, B, C, and D under ICWS'88 and ICWS'86:

Program A: ICWS'88 - One task is SPLit on cycle one and one task terminates on cycle two. This two-cycle cycle repeats indefinitely.

ICWS'86 - The program grows by one task each cycle (remember, growth rates are unrestricted — no task limit).

Program B: ICWS'88 - This is really a very difficult growth process. There is a growth cycle and a marking time cycle which alternate. There are four tasks after the first growth cycle, and the number of tasks is doubled at the end of each subsequent growth cycle. There is no net growth at the end of each marking cycle. The number of machine cycles for each (growth, marking cycle) is:

```
(4,2), (6,6), (10,14),
 (18,30), (34,62), ...
 (2*[g-1]-2,2*[m-1]+2)
```

where `[g-1]` is the number of machine cycles for the previous growth cycle and `[m-1]` is the number of machine cycles for the previous marking cycle.

ICWS'86 - The program grows by one each task cycle.

Program C: ICWS'88 - This is similar to Program B, but even more complicated. The program really spends very little time losing tasks for N > 2, therefore a growth rate of one task per cycle is very close to the actual growth rate.

ICWS'86 - The program grows by one task each cycle (and never gets beyond statement ONE).

Program D: ICWS'86 - This is the program closest to CHANG1, the original impetus for this line of inquiry. For reference, it is reprinted here.

```
; Program D
;
START SPL  IMP           ; S
      JMP  START         ; J
IMP   MOV  IMP,   IMP+1  ; I
;            (I', I", etc.)
      END  START
```

There are four cycles with two tasks, S-I-J-I, six cycles with three tasks, S-I'-I-J-I'-I, eight cycles with four tasks, S-I"-I'-I-J-I"-I'-I, etc. Therefore, the number of cycles with T tasks is just 2T. The total number of cycles to get there is the twice the sum of the numbers two to T.

To find the sum of numbers from one to T, use this trick. Write the sum both forwards and backwards, then add the two sums.

```
  1 +  2  +  3  + . . . +  T
 +T +(T-1)+(T-2)+ . . . +  1
------------------------------
(T+1)+(T+1)+(T+1)+ . . . +(T+1)

          = T(T+1)
```

The sum from one to T is then T(T+1)/2. The sum from two to T is one less and the number of cycles is twice the sum from two to T, thus the total number of cycles C can be written as

$$C = (T+2)(T-1)$$

To answer our question ("what is the growth rate?"), we need to express T as a function of C. Thanks to the quadratic equation, we can write T as

$$T = \frac{-1 + \sqrt{4C + 9}}{2}$$

Be sure to round the answer down. You can easily see that after four cycles there are two tasks, after ten cycles there are three tasks, after eighteen cycles there are four tasks, etc.

So how many tasks are there after thirty-thousand cycles? There are 172 tasks after thirty-thousand cycles. The claim that CHANG1 spawns more than seven hundred thirty tasks after thirty-thousand cycles (Summer 1988 *TCWN*) is clearly false. It would take far more than five hundred thousand cycles for Program D to spawn that many tasks, and CHANG1 is even less efficient.

How to Get Your Name in Print

The Core War Newsletter contains mostly submissions from you, the readers! Without your help, it would be much slimmer and less interesting. Our problem is not too many submissions; we don't have enough! This means that, if you send us an article, or just a short “Letter To The Editor”, chances are it will appear in the next newsletter, or maybe the one after that. Here are some suggestions on “How to Get Your Name in Print”:

- Send in the code for your favorite Core War fighter program, and a description of how it works. We hope to run a Featured Fighter article in every TCWN issue. Your fighter doesn't have to win against the top tournament contenders, as long as it displays an interesting design idea.
- Write a review of a Core War simulator (either someone else's or your own!).
- Write a review of a “program the robot” game, or any other development which might be of interest to Core War fans.
- Describe the latest developments in Artificial Life.
- Write a “Letter To The Editor”. Express your opinion on anything reasonably related to Core War!

Core War Simulator Certification Proposal

Proposed: The ICWS should establish an official procedure to confirm that individual Core War simulators comply with the Core War’88 standard.

The current Core War Standard is the ICWS’88 Standard, defined by the ICWS’88 document produced by Thomas Gettys. Despite the standard, Core War fighter programmers have no way to verify that the simulator they are using behaves the same as other simulators. In particular, they cannot be sure that their fighter program will behave the same in a tournament as it does at home. This means that they must use the tournament simulator to be sure of fair treatment in the tournament.

The ICWS should take the lead

in confirming that simulators comply with the standard, and that they comply with the current understanding of the standard where the standard is unclear. Simulators which pass the test are assured to behave exactly the same in tournaments, allowing Core War programmers to work with the compliant simulator of their choice.

Simulators which pass the test shall be declared “ICWS’88 compliant”, and allowed to use that copyrighted phrase in advertisement. This should create strong incentive for simulator writers to submit their simulators for certification, and to fix compliance problems where they are discovered.

The tests will fall into two categories:

- The validation suite will consist of short programs which test specific problem areas. Mark Durham has already written over two dozen of these tests.

- The “arena testing” will consist of a list of precisely defined battles, where the warriors involved, their relative initial spacing in core, and which starts first will be precisely stated. These battles shall be chosen such that one warrior always wins. In order to pass, not only must the expected fighters win in each case, but the length of the battle must match the expected value. (This implies that it must be possible to determine the exact length of the battle using the tested simulator.)

Simulator authors who want their simulators tested would be required to send/license one copy of their simulators to the ICWS. The ICWS would promise to only use them in the process of validating them, and not to distribute them further.

Tournament Compliance

“ICWS’88 compliance” is only the beginning. The certification described above should make it possible to reliably recreate individual battles given the warrior code and initial placement. We can extend this to recreating entire tournaments, given a description of each battle to be fought. Core War programmers will have the greatest confidence in tournament results when they can rerun the entire tournament at home.

I propose a simple tournament description language, which might look something like this:

```
CORESIZE   8192

# Maximum number of
# processes per side

MAXPROCS   1024

# Lose-tie-win scoring

SCORING    0 1 3

# Cycles before tie

CYCLES     80000

# Warrior number, then
# filename

WARRIOR    0    mice.fgt
WARRIOR    1    leech.fgt
WARRIOR    2    cowboy.fgt

# Warrior number, then
placement in core

BATTLE     0 1217   1 5522
BATTLE     0 0178   2 1150
BATTLE     1 8052   0 3026
BATTLE     1 1232   2 1902
BATTLE     2 5215   0 7623
BATTLE     2 1254   1 5903
```

Tournament description files consist of three parts: the description of the tournament rules, the listing of the contestants, and the list of the individual battles. Compliant simulators can process these files, producing compliant output such as

```
# Warrior number, then
# cycle of death (-1 for
# survivor)

BATTLE   0 5523   1 -1
BATTLE   0 -1     2 27773
BATTLE   1 -1     0 -1
BATTLE   1 66223  2 -1
BATTLE   2 -1     0 2323
BATTLE   2 -1     1 33216

# Warrior number,
# filename, and final
# score

WARRIOR  0   mice.fgt     4
WARRIOR  1   leech.fgt    4
WARRIOR  2   cowboy.fgt   9
```

"ICWS'88 tournament-compliant" simulators are those which are "ICWS'88 compliant," and which also can read and process tournament description files. They are not required to be able to generate new tournament description files, although many may implement that as well. With listings of the tournament competitors, the tournament description file, and a tournament-compliant simulator, anyone can rerun an entire tournament, or examine the individual battles to determine why an individual competitor won or lost.

Also note that, with a tournament-compliant simulator, you can restart a tournament which cannot be run to completion. Suppose your machine crashed, or else you had to briefly use the machine to get some real work done. As long as you generated the entire tournament description file beforehand, you can halt the tournament anytime, and come back to it later! Just trim out the BATTLE lines which have already been fought, and when you're finished, splice the results together. We may be able to define the tournament description language to help in this task.

This plan for compliance-testing is ambitious, especially as extended to tournament-compliance. Mark and I will need your help in defining the rules for compliance, in coming up with test cases, and in testing the dozens of simulators on many different platforms.

Please, if you think of a particularly devious test case; if you have or know of a simulator which should be tested; or if you are willing to test simulators on your exotic platform; write us! There is plenty of work to be done here.

The Core War Newsletter

Mark A. Durham
Editor

AMRAN
Founding Publisher

William R. Buckley
Founding Editor

The Core War Newsletter, the official newsletter of the International Core War Society since 1987, is published quarterly by the International Core War Society for Core War enthusiasts everywhere.

International Core War Society

Jon Newman
Director

Send all correspondence (including renewals and/or changes of address) to:

ICWS/TCWN
c/o Jon Newman
13824 NE 87th Street
Redmond, WA 98052-1959
USA

Back issues of *TCWN* (prior to 1992) are available from AMRAN, 5712 Kern Drive, Huntington Beach, CA 92649-4535.