Other Languages: COW | **Whirl** | 3code | Taxi

Whirl

Important news!

[06/22/05] Kang Seonghoon sent in his **shocking** 15,556 instruction "99 Bottles of Beer" implementation!
[07/04/05] Victor sent in **updated versions** of the flash virtual machine and Java version of the interpreter!
[07/06/05] Victor Ortiz submitted a nice **whirl documentation tool** written in Java!
[09/22/05] Kang Seonghoon wrote a version of the interpreter in **Ruby**!
[11/04/05] Jakob Westhoff mailed a very nice looking **PHP5** version of Whirl!
[11/16/05] Douglas Michael Auclair contributed an implementation of Whirl in **Prolog**!
[11/28/05] Douglas Michael Auclair updated his Prolog Whirl implementation!
[12/15/05] Douglas Michael Auclair dropped off a link to an **AOP in Prolog** article using Whirl as an example!
[03/03/06] Aviad Ben Dov sent a short version of Hello, World as well as a Java-based **code generator**!
[05/26/07] Thanks to Temas for the maple seed logo suggestion!
[09/20/08] Keymaker built a **114,030 instruction quine**!
[09/30/08] BeeTwitLimited announces a **Brainfuck to Whirl translator**! (announcement)
[05/12/09] I wrote an **LLVM-based Whirl compiler**!
[04/02/10] Cameron Behar sent a short Hello World weighing in at only **858 instructions**!

Check this stuff out in the Downloads & Links section below.

Whirl was designed with the following state-of-the-art features in mind:

- Inheritence
- Simplicity
- Ease of use
- XML
- Maintainability
- Polymorphism
- Flexibility
- Power
- Subtraction
- Grilled Chicken

Unfortunately, none of those things made it into the final product.

The Whirl programming language has only two instructions: 0 and 1. While the Whirl Team was tempted to build a compiler that converted text source files of 0s and 1s into very small binary files with a sequence of 0/1 bits, the idea was thwarted at the last minute by a sudden attack of laziness. Maybe it'll happen later on when Whirl becomes adopted by the enterprise market to replace the aging Java platform.

In the world of Whirl, there are two instructions and two rings of 12 things to do on each. The instructions manipulate the rings and activate commands on the rings. The rings can also store some data within them. The rings always maintain their position and state, so nothing really resets the rings back to their default position. A Whirl programmer must always keep track of where the rings are positioned so as to not get a headache. Unfortunately, trying to track the rings' position after about 10 instructions often results in a sudden stoppage of programming due to the programmer's brain getting so dizzy it forgets how to breathe. **WARNING: Programming Whirl may be hazardous to your sanity!**

The Whirl environment generously gives the programmer the following tools:

- An "infinite" supply of data memory
- An ability to change the program's instruction position
- One ring of 12 Control/Logic/IO operations
- One data storage slot for the operations ring
- One ring of 12 Math functions
- One data storage slot for the math ring
- Two easy-to-remember instructions

When a Whirl program begins, the operations ring is set as active and it is in the Noop position. The memory is initialized to 0. The operations ring's data storage slot is set to 0. The math ring is at the Noop position. The math ring's data storage slot is set to 0. The direction setting of both the operations ring and the math ring are set to clockwise.

Whirl Instructions

1 Rotate the current ring in the current direction.

0 Reverse the direction of rotation of the active ring. If previous instruction was a 0 *and it did not trigger an execution*, then the currently selected command on the currently active ring is executed and the active ring is switched to the other ring. To get two executions in a row (one on the current command of each ring) requires 0000 and not 000.
(Thanks coljac for pointing out a need for clarification here!)

Note... In the following documentation, **value** refers to the ring's data value and **memval** refers to the value of the currently selected block of memory.
All commands are listed in clockwise order starting with the default command (the

command the ring is set to when a program begins).

Operations Ring Commands

Noop	Has no effect.
Exit	Terminates the program.
One	Sets **value** to 1.
Zero	Sets **value** to 0.
Load	Sets **value** to **memval**.
Store	Sets **memval** to **value**.
PAdd	Adds **value** to the current program position pointer (a jump).
DAdd	Adds **value** to the current memory position pointer (change memory location).
Logic	If **memval** is 0, then **value** is set to 0. Otherwise, **value** is set to **value** AND 1. This is a logical AND.
If	If **memval** is not 0, then add **value** to program position pointer (see PAdd).
IntIO	If **value** is 0, set **memval** to integer number read from stdin. Otherwise print **memval** to stdout as an integer.
AscIO	If **value** is 0, set **memval** to ASCII character read from stdin. Otherwise print **memval** to stdout as an ASCII character.

Math Ring Commands

Noop	Has no effect.
Load	Sets **value** to **memval**.
Store	Sets **memval** to **value**.
Add	Sets **value** to **value** plus **memval**.
Mult	Sets **value** to **value** multiplied by **memval**.
Div	Sets **value** to **value** divided by **memval**.
Zero	Sets **value** to 0
<	If **value** is less than **memval**, then sets **value** to 1. Otherwise sets **value** to 0.
>	If **value** is greater than **memval**, then sets **value** to 1. Otherwise sets **value** to 0.
=	If **value** is equal to **memval**, then sets **value** to 1. Otherwise sets **value** to 0.
Not	If **value** is not 0, then sets **value** to 0. Otherwise sets **value** to 1.

Neg Sets **value** to **value** multiplied by -1 (inverse).

A Whirl program file consists of a series of 0 and 1s. Anything that isn't a 0 or a 1 is ignored. So commenting is pretty easy -- just beware of numbers that might have a 1 or 0 in them!

Examples

Do 1+1 and print the results:

```
00 - run ops.noop, switch to math ring
0 - math::ccw
11 - rotate to math.not
00 - run math.not, switch to ops ring
00 - run ops.noop, switch to math ring
0 - math::cw
1111 - rotate to math.store
00 - run math.store, switch to ops ring
00 - run ops.noop, switch to math ring
1 - rotate to math.add
00 - run math.add, switch to ops ring
00 - run ops.noop, switch to math ring
0 - math::ccw
1 - rotate to math.store
00 - run math.store, switch to ops ring
11 - rotate to ops.one
00 - run ops.one, switch to math ring
00 - run math.store, switch to ops ring
0 - ops::ccw
1111 - rotate to ops.IntIO
00 - run ops.IntIO, switch to math ring
```

This sample reads two numbers from stdin, adds them together, and prints the result. There is also a commented version. *Thanks coljac!*

```
01100000111100001111100000111100001111100000111000
01111100000110010000011001111100011100011110011001
11000000000111110001000111110011001111100010001100
```

Meta

Whirl has been termed a "**Turning Tarpit**" language which means: A highly minimalist despotic language in which the only way to select an operation is by moving along a list of stateful encodings. This was coined as the result of a Freudian slip of the term Turing tarpit during a discussion about Whirl on the #esoteric IRC channel.
(I found this out by accident thanks to Google. Way cool!)

Downloads & Links

The C++ source of the interpreter is available. Feel free to play with it. Thanks to The Slarty, it has been reasonably well-tested and a few fixes have been made.

Coljac has graciously provided the world with a Java version of Whirl! Victor Ortiz has

provided an updated Java version as well.

The Slarty has done one heck of an amazing job generating a Hello, World program for Whirl using an intermediate language of his own design! This sucker weighs in at about 28k. I wasn't sure it could be done... and yet...

Kang Seonghoon has contributed a slightly obfuscated C version of Whirl (updated to version 3). You can also download it from here.

Here's a much shorter version of Hello, World written by Kang Seonghoon. It can also be found here. Wow!

Writing Whirl is fun, but watching it is even better! Check out this **flash-based virtual Whirl machine** (updated to version 1.01) sent in by Victor Ortiz. He also sent in some code for computing the Fibonacci sequence. Excellent stuff!

The Hello, World by Kang Seonghoon (above) was impressive for its compactness and now Brennan Schweitzer has reverse engineered it and provided a detailed walk-through of the code. Great work!

Kang Seonghoon strikes again, this time with the classic "99 Bottles of Beer" implemented in a shocking 15,556 instructions! **Beyond cool!** The code is also available here.

Need help debugging your Whirl program? You could try out the Whirl documentation generation tool written in Java by Victor Ortiz.

Kang Seonghoon wrote a Whirl interepreter in Ruby (also here) for the universe to enjoy. There can never be too many Whirl interpreters!

I woke up one morning to a gift in my inbox: a wonderful OO implementation of Whirl written in PHP5 by Jakob Westhoff! The more the merrier! Thanks Jakob!

Douglas Michael Auclair sent me his (updated to **0.08**) Prolog version of Whirl.

The prolific Douglas Michael Auclair wrote to point to the way to his latest work which discusses Aspect Oriented Programming (AOP) in Prolog which uses his Prolog Whirl interpreter as an example application with a resulting performance improvement of nearly 150 times!

Aviad Ben Dov mailed a short version of Hello, World which he created with his Java Whirl Code Generator which has support for strings and basic math (add,sub,mul,div). The source code is also available. Awesome stuff, Aviad!

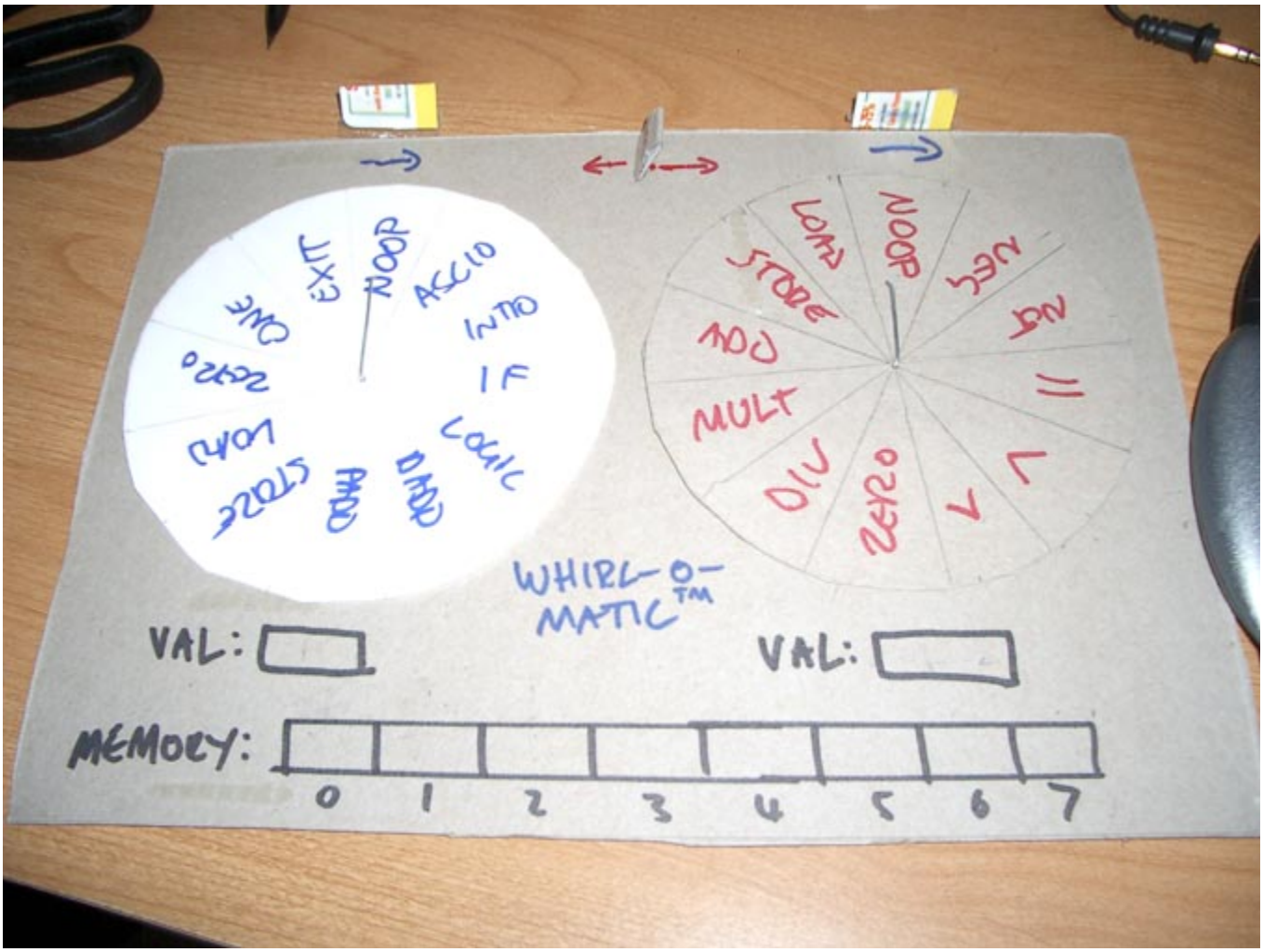
Keymaker sent me a stunning 114,030 instruction quine! (A quine is a program that, when run, prints it's own source code!) Pretty incredible!

Nate Thern (aka BeeTwitLimited) sent a Brainfuck to Whirl translator written in Perl. Be sure to read his awesome announcement, too! Thanks Nate!

Check out my LLVM-based Whirl compiler here: whirl-llvm.cpp and whirl-runtime.c (See blog for some extra notes.)

Cameron Behar sent a 858 instruction hello world - the smallest to date!

Hardware



Coljac sent in this image of a prototype for a physical Whirl machine!

History

Whirl was first published on July 9, 2004. It was based on an idea I had spinning around in my head since September 9, 2003 (or so the timestamp on the file tells me).

The Esoteric Programming Languages Ring
<< Prev Ring Hub Next >>

check out my other stuff