The Painter Weaving Language

Introduction

MetaCreation's Painter 5 application provides facilities for creating images using tools that mimic natural media.

One of Painter's facilities is a "weaving" engine for an eight-shaft, eight treadle loom. On the surface, the weaving engine offers a variety of built-in weaves that can be displayed in various ways — basically a way to produce patterns. Behind the scenes and not mentioned in the user's manual, is an "advanced weaving language" [1]. The user can compose and edit expressions that describe the threading, treadling, and warp and weft color sequences. The tie-up is handled in the conventional manner. Figure 1 shows an example of the weaving dialog for a shadow weave and Figure 2 shows part of the corresponding image. See the Appendix for a threading draft.

The weaving language consists of expressions that produce sequences of characters. In the case of the threading and treadling, sequences are composed of digits 1 though 8, which identify the shafts and treadles, respectively.

For example, in threading the shafts, the sequence 1346 means the first warp thread goes through shaft 1, the second through shaft 3, the third through shaft 4 and the fourth through shaft 6. In treadling, 1346 means treadle 1 is pressed for the first weft thread, treadle 3 for the second weft thread, 4 for the third, and 6 for the fourth. Full sequences are, of course, much longer than this.

Colors are represented by letters, with the actual color values being assigned elsewhere in the application.

The power of the weaving language lies in its repertoire of expressions, which can used to describe sequence structure — patterns. There are 15 expressions in all, ranging from simple to complex in terms of their descriptive power.

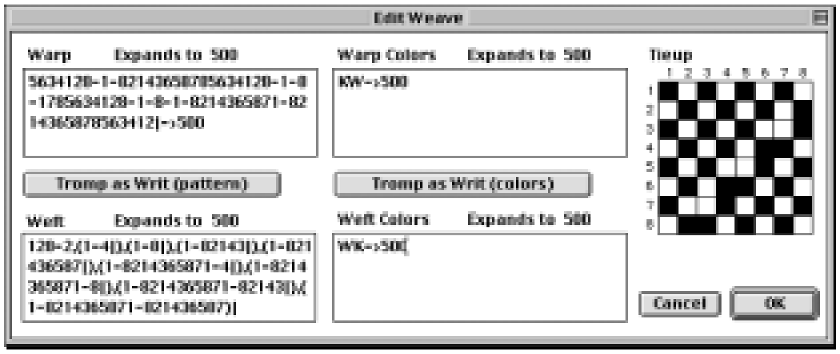
The expressions are essentially the same for colors as for the threading and treadling, but some operators do not apply to colors. In our examples, we'll use the threading and treadling expressions.

The Domain

The concept of *domain* plays an important role in the weaving language. The domain consists of the digits that label the shafts and treadles. The sequence 12345678 is called the *domain run*. Sequences "wrap around" on the domain. That is 1 follows 8 in situations involving domain runs. This is just arithmetic modulo 8 with the numbering starting at 1 instead of 0 to accommodate the convention used in weaving drafts.

Expression Syntax

Expressions are composed of *operators* and *operands* on which they operate. In ordinary arithmetic, 3 + 5 is an expression in which the operator + (addition) operates on its operands, 3 and 5 to produce 8. The operands of weaving operators are sequences and integers. The operators produce


Edit Weave

Warp Expands to 500

```
5634128-1-82143658785634128-1-8
-1785634128-1-8-1-8214365871-82
14365878563412|->500
```

Warp Colors Expands to 500

```
KW->500
```

Weft Expands to 500

```
128-2,(1-4[],(1-8[],(1-82143[],(1-821
436587[],(1-8214365871-4[],(1-8214
365871-8[],(1-8214365871-82143[],(
1-8214365871-8214365 87)|
```

Weft Colors Expands to 500

```
WK->500
```

Tieup

	1	2	3	4	5	6	7	8
1	■	□	■	□	■	□	■	□
2	□	■	□	■	□	■	□	■
3	■	□	■	□	■	□	□	■
4	□	■	□	■	□	■	■	□
5	■	□	■	□	□	■	□	■
6	□	■	□	■	■	□	■	□
7	■	□	□	■	□	■	□	■
8	□	■	■	□	■	□	■	□

Figure 1. Dialog for a Shadow Weave

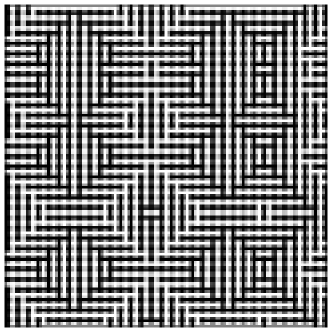


Figure 2. Image of Shadow Weave

April 10, 1999; last revised August 2, 2004 sequences. For example, in the weaving expression

```
16243 # 2
```

# is the rotation operator, its left operand is a sequence and its right operand is the number of places to rotate to the left. The result is

```
24316
```

Most operators can be written in terms of names, short names, and symbols. For example, rotate, rot, and # are equivalent. We'll use symbols where they are available.

Spaces between operators and operands are optional in most situations and can be used to improve readability. Parentheses can be used to group operators and their operands and also to improve readability.

The Operators

Concatenation

Concatenation (concat or ,) appends one sequence to another to produce a longer sequence. For example,

```
(16243#2),432
```

appends 432 to the result of rotating 16242 by two places and produces

```
24316432
```

Repetition

Repetition (repeat, rep, or ∗) repeats its left-operand sequence the number of times given by its right operand. For example,

```
1346∗7
```

produces

```
1346134613461346134613461346
```

Blocks

In creating blocks (block or [ ]), the left operand is a pattern and the right operand is a sequence of integers. Each character in the left operand is repeated individually by the corresponding integer in the right operand. For example,

```
1346[ ]9231
```

expands to

```
111111111334441
```

Integers greater than 9 can be specified by enclosing them in braces. For example,

```
1345[ ]12{10}3
```

produces

```
1334444444444666
```

There also is an interleaved form. For example,

```
[1 1 3 2 4 10 5 3]
```

produces the same result as the example above.

Extension

The concept of *extension* permeates the weaving language, as it does weaving itself. Extension (extend, ext, or –>) replicates its left operand to produce a result whose length is given by its right operand. For example,

```
1346 –> 16
```

produces

```
1346134613461346
```

If the replication does not come out even, it is truncated on the right. For example,

```
1346 –> 15
```

produces

```
134613461346134
```

Reversal

Reversal (reverse, rev, or `) reverses the order of a sequence. In this case, there is only one operand and the operator follows it in suffix position. For example,

```
12365238`
```

produces

```
83256321
```

Rotation

Rotation was described earlier but is included here for completeness. The left operand of rotation (rotate, rot, or #) is a sequence and its right operand an integer, which may be negative. It moves the specified number of characters from the beginning of the sequence and places them at the end. If the number is negative, the characters are moved from the end to the beginning. For example,

```
16243#–2
```

produces

43162

Palindromes

Palindromes are common in weaving and other design contexts. The palindrome operator (palindrome, pal, |), like rotation, is in suffix position. For example

1346|

produces

134643

Note that this is not a true palindrome — it does not read the same forward and backward. It is what is called a pattern palindrome

The reason pattern palindromes are not true palindromes has to do with their use in repeats. Consider, for example, the (true) palindrome 1346431, derived from 1346. If this is repeated, the result is

1346431134643113464311346431 …

Note the duplication of 1s at the boundaries of the repeats. This produces unavoidable and generally undesirable artifacts. On the other hand, if the pattern palindrome 134643 is repeated, there is no such artifact:

134643134643134643134643 …

When creating a true palindrome from a sequence, the last character of the sequence could be repeated, as in 13466431. This also produces artifacts:

13466134664314311346643113466431…

This also is not done in pattern palindromes.

Interleaving

Interleaving (interleave, int, ~) interleaves the characters of two sequences. For example,

1234 ~ 5678

produces

15263748

If one operand is shorter than the other, it is extended to the length of the other.

Domain Runs

There are four operators related to domain runs.

“Upto” (upto, <, or –) concatenates its left and right operands but inserts between them the portion of the domain between the last character of the left operand the first character of the right operand. For example,

1346 < 8

produces

134678

If, however, the last character in the left operand is greater than the first character in the right operand, the intervening portion “cycles” through the domain, so that

1346 < 3

produces

134678123

“Tick marks”, indicated by a single quotes, can be placed in front of the right operand. When this is done, the number of tick marks specifies the number of complete domain runs to be added in. For example,

1346 <" 8

adds two domain runs and produces

1346781234567812345678

The operator – can be used in place of < if the last character of the left operand is less than the first character of the right operand, as in

123 – 765

There is a corresponding “downto” operator (downto, >, or –). For example,

8 > 3

produces

876543

The operator – can be used in place of > if the last character of the left operand is greater than the first character of the right operand.

The ”updown” operator, (updown or <>) produces alternating ascending and descending domain runs. The first, ascending, run starts at the first character of the left operand and goes to the first character of the second operand. The second, descending, run starts from there and goes to the second character of the right operand, and so on, alternating between ascending and descending runs. For example,

```
1234<>5678
```

produces

```
1234543234565434567654567 8
```

As in "upto", tick marks can be used to indicate domain cycles between runs.

The "downup" operator, (downup or ><), is like "updown" except that the order is descending, ascending, … .

Permutation

The permutation operator (permute or perm) applies a permutation vector (right operand) to a pattern (left operand). The pattern is permuted in sections whose lengths are the lengths of the permutation vector. The permutation vector specifies the positions of the elements in a section. For example, 4123 puts the fourth character of the section first, the first second, the second third and the third fourth. Thus,

```
1346 perm 4123
```

produces 6134.

In the case that the pattern is not the same length as the permutation vector, the pattern is extended to an *integer multiple* of the length of the vector.

Pattern Boxes

The pattern box operator (pbox), like perm, has a left operand pattern and a right operand permutation vector. The permutation vector is extended to the length of the pattern and the permutation is applied. For example,

```
123456787654321 pbox 21436587
```

produces

```
214365878563412
```

Templates

The template operator (template, temp, or :) provides for "sub-articulation" of a pattern (left operand) by a "texture pattern" (right operand). The first character (digit) in the template pattern is taken as the root. The remaining digits in the template pattern are taken with respect from their distance from the root. For example, in the template pattern 342 the root, *r*, is 3 and the template is *r*, *r* + 1, *r* – 1. The template is applied to each character in the pattern with the character replacing the root. For example,

```
12345678:121
```

has the template *r*, *r* + 1, *r* and produces

```
121232343565676787818
```

Note that values wrap around on the domain, so that for the last character of the pattern, 8, *r* + 1 produces 1.

Comments

It is interesting to note that the weaving language, as rich as it is, does not have operators for some patterns that occur frequently in weaving. Two missing ones are true palindromes and the interleaving of more than two sequences.

Another problem relates to domains. Although some of Painter's built-in weaves use only four shafts and four treadles, and only the labels 1, 2, 3, and 4, there is no way to restrict domain runs to this subset. For example, 4<2 produces 4567812, not 412 as it would if the domain could be restricted. Of course, 5678 does nothing.

The restriction to 8 shafts and 8 treadles is more fundamental, since it limits the kinds of things that can be woven. More shafts and treadles could be handled by extending the domain to include more labeling characters. If the number of shafts and treadles is not the same, the situation becomes more complicated, especially for domain operators.

The weaving language is difficult to use in the context of Painter. Expressions are limited to 256 characters. The fields for entering expressions are small and there is no way to find out what sequence an expression produces except by trying to puzzle it out in a resulting image.

It also is much harder to produce desired results in an expression-based language than it is to do so with a graphic representation.

The Painter weaving language is, nonetheless, very powerful. It allows complicated patterns to be expressed concisely and built up from simpler ones. It could well serve as a basis for a more powerful language that has variables, data structures, conditionals, control structures, and recursion. Such a language might not be suitable for designing weaves directly, but it could serve as a foundation for a powerful weaving program with graphical capabilities.

Reference

1. *Advanced Weaving*, PDF on Painter 5 CD-ROM, 1998.

Appendix — The Threading

The complete sequence for the threading (and treadling) is 183 characters long:

```
123456787654323456782876543456782128765
456782141287656782143412876782143634128
782143656341282143656341287821436341287
678214341287656782141287654567821287653
45678287654323456787654321
```

Figure 3 shows a conventional draft for the first half of the threading. The remainder is the reversal of the first half. It was obtained by a program that converts Painter weaving specifications to WIFs. The WIF then was opened in SwiftWeave.

Ralph E. Griswold

Department of Computer Science
The University of Arizona
Tucson, Arizona

© 1999, 2002 , 2004 Ralph E. Griswold

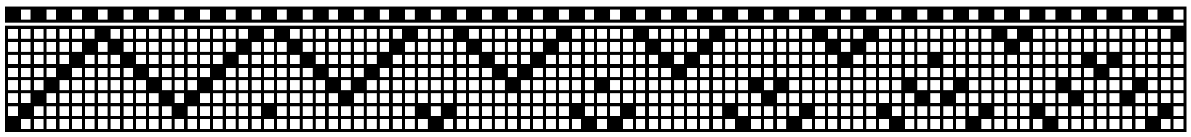


Figure 3. The Threading