decided to use the word *frame* as the seed for my brainstorm. One thing led to another, and I soon had a bunch of ideas dealing with *enclosing* games within rule structures, with recursive games and *frames within frames*, like those pictures that show a person holding a picture of herself holding a picture of herself, and so on. This one seed for my mental random-number generator eventually shaped the design of my game system a great deal, and it is now called GameFrame.

It's a sure thing that my "creative DNA" determined the shape of my game, at least partially. For example, I've been fascinated with recursion and self-reference at least since I read *Gödel, Escher, Bach* when I was 14. These powerful inclinations are analogous to the rigid programming of one's PC. However, the street sign bumped me out of my rut. Before I gave my *random-number generator* a *seed* to chew on, the only idea I came up with was that my game might use drinking straws as components!

End Notes

1. Hall, Doug, and David Wecker. 1995. *Jump Start Your Brain*. Werner Books.

See Also

- Random Wikipedia page (*http://en.wikipedia.org/wiki/Special:Randompage*)
- Random page from H2G2 (*http://www.bbc.co.uk/dna/h2g2/RandomEditedEntry*)
- Other random URLs (*http://randomurl.com*)
- Wikipedia entry for "Random number generator" (*http://en.wikipedia.org/wiki/Random_number_generator*)

Force Your Connections

Use a simple process to generate many complex ideas quickly from a limited pool of simple ideas.

The process of *morphological forced connections* is fairly old; the picture books for children that allow you to combine the head of a giraffe with the body of a hippo and the tail of a fish are one example. The process was formalized by Fritz Zwicky at Caltech in the 1960s[1] and was popularized in 1972 by Don Koberg and Jim Bagnall in their book *The Universal Traveler*.[2]

Most other books that discuss the technique seem to derive their discussion of it from *The Universal Traveler* and even use the same example: creating a new design for a ballpoint pen. We'll take a somewhat different approach.

In Action

The basic process for making forced connections, as outlined by Koberg and Bagnall, is simple and sound:

1. List possible features of the object you are trying to create, one feature per column. For example, the features might include color, size, and shape.
2. In the column under each feature variable, list as many values for that variable as you can. For example, under *color* you might list all the colors of the rainbow, as well as black, white, gold, and silver.
3. Finally, randomly combine the values in your table many times, using one value from each column. To continue our example, you would use one color, one size, and one shape each time.

Technically, steps 1 and 2 are *morphological analysis*, and step 3 is the *morphological forced connections* stage.

The result will be a randomly generated list of possibilities, none of which might be just what you're looking for, but most of which will probably be interesting. Feel free to fine-tune the results. For example, you might not like the suggestion "orange, tetrahedron, a meter on a side," but "orange, tetrahedron, *half* a meter on a side" might hit the spot.

Of course, you can force connections with a pen and paper, as recommended in *The Universal Traveler*, but computers have become widespread since 1972 and you might find them to be a much more efficient tool. To that end, this hack includes a Perl script called *pyro*, which is a somewhat streamlined successor to a HyperCard stack for the Macintosh called Inspirograph that I released in the 1980s.[3]

Inspirograph could generate anything from New England place names (like Lake Nattagoonsucketpocket) to random tabloid headlines. Because the examples I included were humorous, many people who downloaded the stack thought it was only good for a laugh, but it was actually intended for serious design, as is *pyro*.

The Code

Place the following Perl script in a file called *pyro* and make it executable. You also can download the *pyro* script and accompanying *utopia.dat* file from this book's page on O'Reilly's web site (see the Preface for details).

```
#!/usr/bin/perl -w
my $infilename = $ARGV[0];
my $basevar = "\@$ARGV[1]\@";

my $pickatrandom = 1;
if (($ARGV[2]) && ($ARGV[2] eq "all"))
{
    $pickatrandom = 0;
}

# Seed the output file
my $outfilename = "/tmp/pyro.txt";
open(OUTFILE, "> $outfilename")
    or die "Couldn't open $outfilename for writing: $!\n";
print OUTFILE "$basevar\n";
close(OUTFILE);

local $/;
undef $/;

open(INFILE, "< $infilename")
    or die "Couldn't open $infilename for reading: $!\n";
$infilecontents = <INFILE>;
close(INFILE);

open(OUTFILE, "< $outfilename")
    or die "Couldn't open $outfilename for reading: $!\n";
$outfilecontents = <OUTFILE>;
close(OUTFILE);

while ($outfilecontents =~ /\@[A-Za-z0-9]+\@/)
{
    local $/;
    undef $/;

    open(OUTFILE, "< $outfilename")
        or die "Couldn't open $outfilename for reading: $!\n";
    $outfilecontents = <OUTFILE>;
    close(OUTFILE);

    # $baseline is the first line in OUTFILE with a variable.

    if ($outfilecontents =~ /^(.*?)(\@\w+\@)(.*?)$/m)
    {
        $baseline = "$1$2$3";
        $varname=$2;
    }
    chomp $varname;

    if ($infilecontents =~ /\#$varname\n(.*?)\n\n/s)
    {
        $varblock = "$1\n";
    }
```

```
    else
    {
        die "Did not find variable $varname in $infilename.\n";
    }

    @varblockarr = split (/\n/, $varblock);
    @outlinesarr = ();

    if ($pickatrandom)
    {
        # Generate a random string from the input elements
        $randline = $varblockarr [ rand @varblockarr ];
        $curline = $baseline;
        chomp ($curline);
        chomp ($randline);
        $curline =~ s/$varname/$randline/;
        push (@outlinesarr, $curline);
    }
    else
    {
        # Generate all possible combinations of the input elements
        foreach $varline (@varblockarr)
        {
            $curline = $baseline;
            chomp ($curline);
            chomp ($varline);
            $curline =~ s/$varname/$varline/;
            push (@outlinesarr, $curline);
        }
    }

    $outlines = join ("\n", @outlinesarr);
    $outfilecontents =~ s/\Q$baseline\E/$outlines/s
        or die "baseline not found.\n";
    $outfilecontents =~ s/\n\n/\n/mg;
    open(OUTFILE, "> $outfilename")
        or die "Couldn't open $outfilename for writing: $!\n";
    print OUTFILE $outfilecontents;
    close(OUTFILE);
}

if ($pickatrandom)
{
    print $outfilecontents;
}
```

Running the Hack

Here are the first 20 lines of a datafile (called *utopia.dat*) for the *pyro* script. You can use this file to generate interesting settings for fantasy and science fiction stories. The values for the variables were culled from two reference works on speculative fiction.[4,5]

In case you don't have a computer running Perl handy, or if you'd simply rather not get into running code, I'll explain an alternate way to do the same thing with dice later.

```
#@place@
@type@ made of @material@ @location@, inhabited by @inhabitants@, @govt@,
@special@

#@type@
a cavern
a city
a country
a forest
a jungle
a mountain
a planet
a sealed habitat
a village
an island

#@material@
a superstrong material
crystal
flesh
gold
```

See the "How to Run the Programming Hacks" section of the Preface if you need general instructions on running Perl scripts.

If you have Perl installed on your system, save the *pyro* script and the *utopia.dat* file in the same directory, and then run *pyro* by typing the following command within that directory:

```
perl pyro utopia.dat place
```

If you're on a Linux or Unix system, you might also be able to use the following shortcut:

```
./pyro utopia.dat place
```

Each variable in any *pyro* datafile you create should be wrapped in two `@` signs, as in this example. To assign a set of possible values to a variable, place the variable on a line by itself preceded by a hash mark (`#`).

Each successive line should contain one possible value. Separate variable/value sections with blank lines, and leave two or more blank lines at the end of the file. You can create variables whose values contain other variables, as shown with the `place` variable in this example.

In Real Life

Table 3-1 shows the data from *utopia.dat* in tabular form. Each column of text contains the 10 possible values for one variable, whose name is at the head of that column. There are six columns of 10 items, so you can generate 10^6 or a million possible fantasy places from this data.

Table 3-1. Data to generate fantasy places

No.	Type	Material	Location	Inhabitants	Govt.	Special
0	A cavern	A super-strong material	In an unknown place	Aliens	A socialist utopia	Exceed-ingly war-like
1	A city	Crystal	In another universe	Apes	An anarchy	Where cannibal-ism is practiced
2	A country	Flesh	In space	Completely normal people	Ruled by a corporation	Where everyone is happy
3	A forest	Gold	In the desert	Fairies	Ruled by a council of many species	Where everyone is insane
4	A jungle	Paper	In the dream world	Fish	Ruled by a god	Where learning is exalted
5	A mountain	Porce-lain	In the fourth dimension	Ghosts	Ruled by a hereditary monarch	Where telepathy is common
6	A planet	Rubber	In the polar regions	Giants	Ruled by a magical elite	Which is boundless
7	A sealed habitat	Stone	In the sky	Insects	Ruled by ancient ritual	Which is micro-scopic
8	A village	Water	Under the Earth	Intelligent plants	Ruled by com-puter	Which is sacred
9	An island	Wood	Under the sea	Robots	Ruled by women	Whose location changes

If you have a 10-sided die (or, even better, six of them), or if you have some other way to generate random digits, you can create fantasy places using Table 3-1, generating digits sequentially and selecting one item from each column. For example, Utopia #895779 is "a village made of wood in the

fourth dimension, inhabited by insects, ruled by an ancient ritual, whose location changes."

The *pyro* script will do the same thing faster. It takes two mandatory arguments and a third optional one:

- The first argument is the name of the datafile to use, such as *utopia.dat*.
- The second argument is the name of the variable from the datafile that you want to expand, such as `place`. (Do not wrap the variable name in `@` signs to match the datafile; *pyro* will do that for you.)
- The third argument, which is optional, is the word `all`. If this option is used, *pyro* will generate all possible combinations of the elements in the datafile, instead of one random element.

Generating all possible combinations with the `all` argument might take some time.

Thus, the following command will generate all one million possible fantasy places and leave the results in the file */tmp/pyro.txt*:

```
./pyro utopia.dat place all
```

But this command will simply generate one fantasy place:

```
./pyro utopia.dat place
```

Here's a slightly edited console log of running the previous command:

```
$ ./pyro utopia.dat place
a mountain made of rubber in the sky, inhabited by fish, ruled by a
hereditary monarch, where learning is exalted
$ ./pyro utopia.dat place
an island made of crystal in the fourth dimension, inhabited by aliens, an
anarchy, which is microscopic
$ ./pyro utopia.dat place
a sealed habitat made of paper in the dream world, inhabited by giants, a
socialist utopia, where learning is exalted
$ ./pyro utopia.dat place
an island made of flesh in the fourth dimension, inhabited by fairies, an
anarchy, which is boundless
$ ./pyro utopia.dat place
a mountain made of a superstrong material in the dream world, inhabited by
apes, an anarchy, where cannibalism is practiced
$ ./pyro utopia.dat place
a cavern made of porcelain in the polar regions, inhabited by intelligent
plants, ruled by a council of many species, which is microscopic
$ ./pyro utopia.dat place
a sealed habitat made of stone in an unknown place, inhabited by giants,
ruled by a magical elite, where everyone is insane
```

```
$ ./pyro utopia.dat place
a sealed habitat made of flesh in the desert, inhabited by giants, ruled by
a corporation, where cannibalism is practiced
$ ./pyro utopia.dat place
an island made of paper under the sea, inhabited by fairies, an anarchy,
which is boundless
$ ./pyro utopia.dat place
a forest made of flesh in an unknown place, inhabited by fairies, ruled by
computer, exceedingly warlike
```

While *pyro* is useful as it stands, it's certainly possible to improve it. More robust error checking could be added (such as checks for circular variable definitions, and lack of a newline at the end of the file or between entries), and it might be nice if the number of random strings to be generated were an alternate value for the third command-line option. It's open source, so hack away!

End Notes

1. Ritchey, Tom. 2002. "General Morphological Analysis: A general method for non-quantified modeling." *http://www.swemorph.com/ma.html*.
2. Koberg, Don, and Jim Bagnall. 1976. *The Universal Traveler: A Soft-Systems Guide to Creativity, Problem-Solving, & the Process of Reaching Goals*. William Kaufmann, Inc.
3. The original Inspirograph. *http://ron.ludism.org/cloudbusters/inspirograph-10.hqx*.
4. Manguel, Alberto, and Gianni Guadalupi. 1980. *The Dictionary of Imaginary Places*. Macmillan Publishing Co., Inc.
5. Stableford, Brian. 1999. *The Dictionary of Science Fiction Places*. The Wonderland Press.

See Also

- Raymond Queneau of the French literary group, the Oulipo **[Hack #24]**, used morphological forced connections in his 1960 book *Cent Mille Milliards de Poèmes*, or *One Hundred Thousand Billion Poems*, a flip book with 10 possible strips for each of the 14 lines of a sonnet, hence 10^{14} or 100,000,000,000,000 sonnets. As usual, computers make the whole thing easier. Here's a decent web version in English: *http://www.bevrowe.info/Poems/QueneauRandom.htm*.
- The Oulipo member Harry Mathews also devised a forced-connections procedure dubbed Mathews's Algorithm, which is neither random nor exhaustive. You can read about it and experiment with it online: *http://bumppo.hartwick.edu/Oulipo/Mathews.php*.