Program Visualization and Weave Design

Computer programmers face a daunting task in finding errors and inefficiencies in large, complex programs. One technique that has been used effectively is program visualization [1-3], in which the activities of a running program are presented as evolving visual images. This allows the powerful human vision system to detect patterns and note anomalies that would be lost in equivalent textual information.

Various methods have been used in program visualization. Color typically is used to identify types of program activity, while size and shape are used to represent magnitudes. Some visualization techniques are adaptations of conventional graphic presentations, like pie charts and bar graphs. Other techniques use metaphors, like exploding stars and storms, to trigger recognition. Others are abstract and appeal to the right side of the brain.

The third page of this article shows some snapshots from program visualization.

What does all this have to do with weave design? The connection comes from the fact that program activity consists of a sequence of events. These events often can be characterizes by a type and an associated value. For example, reading and writing strings are different events in which the lengths (not the specific characters) may be of interest. Sequences of events can be separated into two parallel sequences: type sequences and a value sequences.

For drafting, these sequences can be interpreted as threading or treadling sequences with associated color sequences. A useful interpretation needs to take into account the ranges of types and values. There usually are comparatively few different types, while values vary greatly in magnitude. For this reason, it works best for types to be associated with shafts and treadles and values with color — rather the opposite of program visualization.

There are several problems in using event sequences for drafting.

One is that typical event sequences are very long — many times the number of ends or picks for a practical draft. A portion of the event sequence therefore needs to be used. It is best not to use the initial part of an event sequence, because it comes from program initialization and may not be typical. The same consideration applies to the end of an event sequence, where program termination also may not be typical of program activity.

Another problem is that event sequences often contain runs of the same event. These would translate into adjacent duplicates in threading and treadling sequences. The simplest way to solve this problem is to delete events with successive identical types. Adjacent duplicates in colors do not, of course, present a structural problem.

Finally, there may be a problem with the range of values, which may far exceed the numbers of different colors that are possible, much less desirable. One solution is to categorize values by range. For example, if values range from 1 to 1,000 and five colors are to be used, values from 1 to 200 can be interpreted as the first color, values from 201 to 400, the second color, and so on. There is, incidentally, no perception in the human visual system of a relationship been color and magnitude, so there is no point in trying to convey this in a weave. An example of a draft based on an event sequence appears on the last page of this article.

This approach to drafting is somewhat akin to code drafting to commemorate a word or phrase. With program visualization, what is commemorated is a segment of computational activity in a running program.

This can be taken as an example of how far-fetched drafting techniques can be. And there is the problem that getting event sequences is a job for a programming specialist.

References

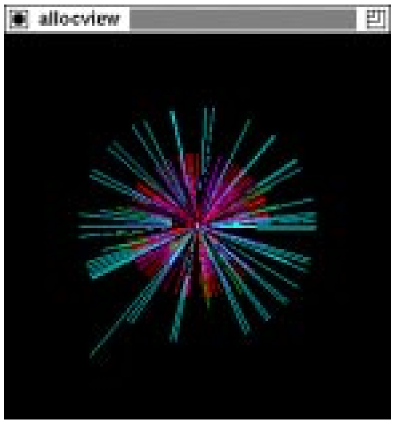
1. *Program Visualization Research at IBM:* http://www.research.ibm.com/pvres/

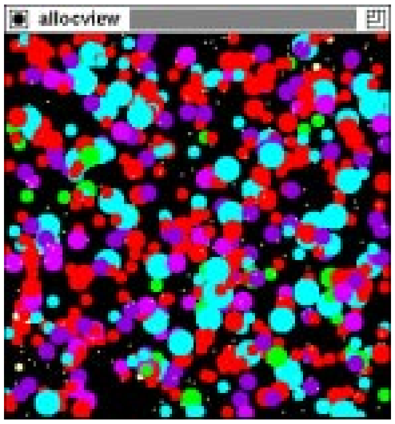
2. *Program Visualization -- Overview:* http://www.cs.arizona.edu/icon/progvis/lectures/intro.htm

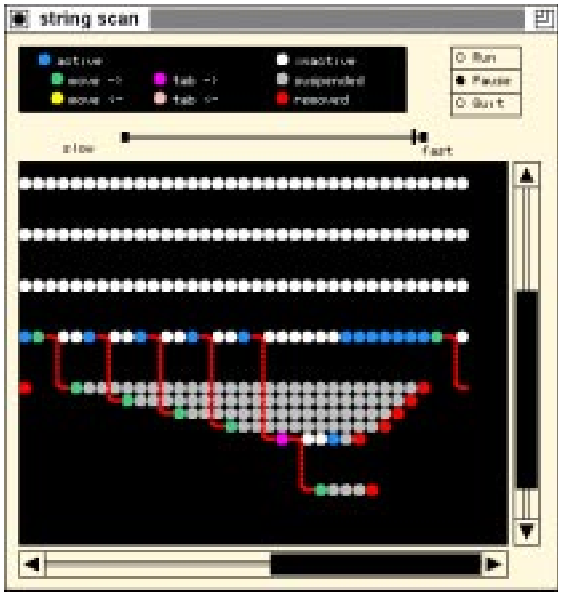
3. *An Overview of Program Visualization Tools and Systems:* http://portal.acm.org/citation.cfm?id=275358&dl=ACM&coll=portal

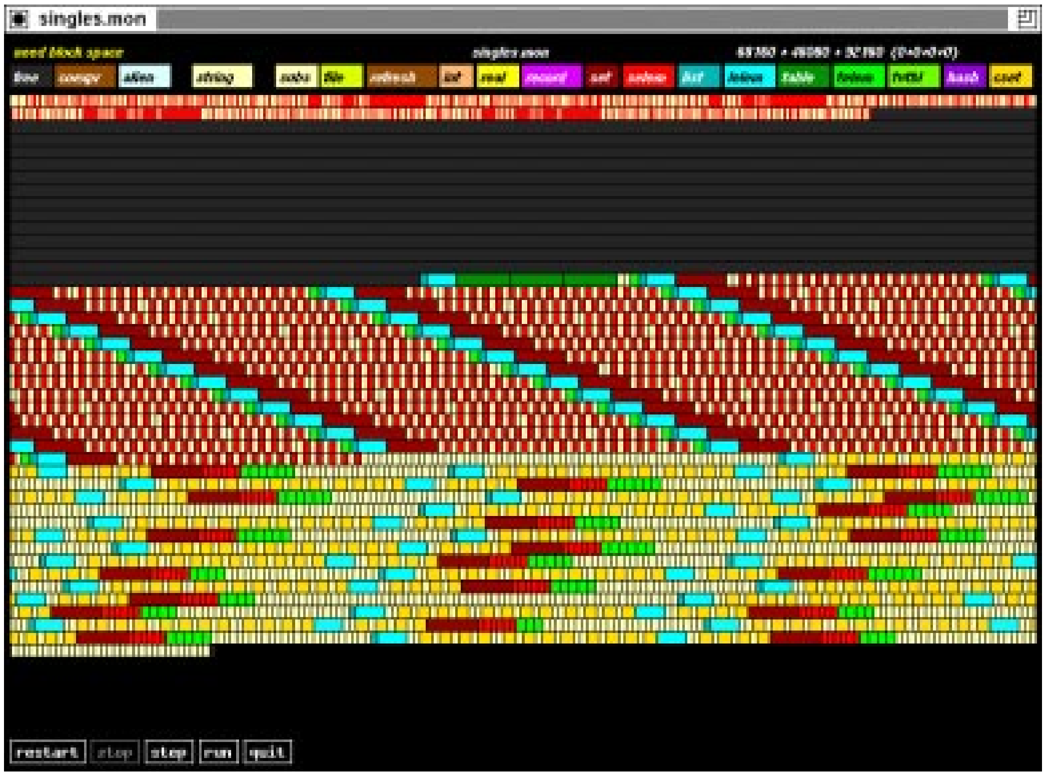
Ralph E. Griswold

Department of Computer Science
The University of Arizona
Tucson, Arizona

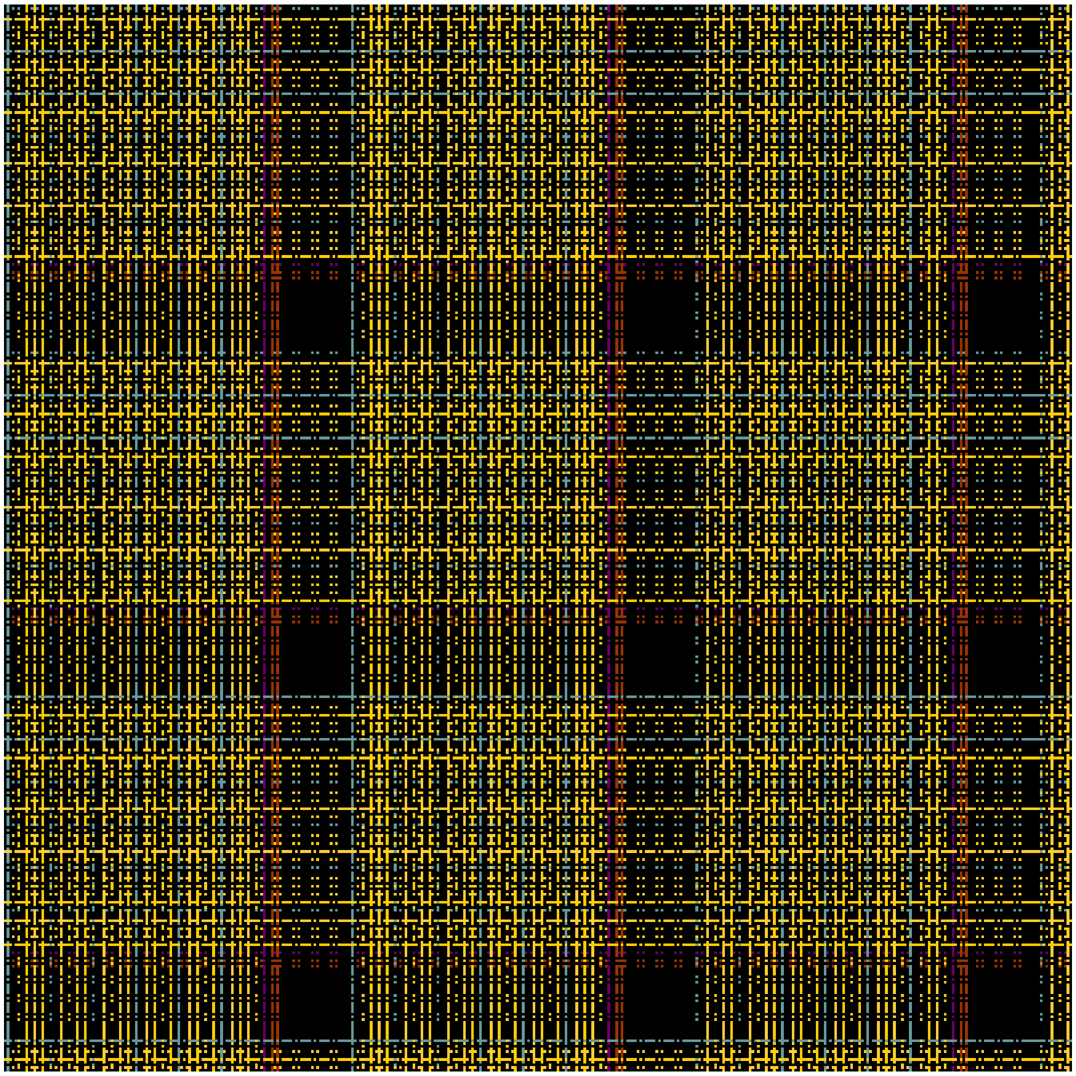








Snapshots of Program Visualization



A Chess-Playing Program in Action