Harvey, B. (2020). Commentary: Thoughts on "Exploring Language: The Journey from Logo to Snap!" *Contemporary Issues in Technology and Teacher Education, 20*(3), 583-585.

Commentary: Thoughts on "Exploring Language: The Journey from Logo to Snap!"

Brian Harvey
University of California, Berkeley

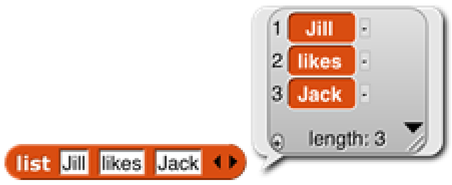
Read related papers:
Introductory Article: Integrating Language Arts and Computational Thinking: A Reflection on the Importance of Gossip
Commentaries:
Gossip and Other Life Sentences: A Simple Grammar
Thinking: A Reflection on the Importance of Gossip
Reflections on "Exploring Language With Logo"
Exploring Language: The Journey from Logo to Snap!

Editor's Note: Brian Harvey was a member of the design teams for microcomputer versions of Logo for the Apple II, the Atari 800, and the Apple Macintosh and served as the lead developer of Berkeley Logo, a free implementation of Logo available from the University of California, Berkeley. He is the codeveloper, with Jens Mönig, of Snap*!*, a block programming language.

I am delighted that Snap*!* is viewed as an improvement, but in various ways this article does not do justice to Logo.

Lists of Words

The Snap*!* LIST block is compared with the Logo square bracket notation. But one advantage of the Logo notation is that it's the same for input and output. The Snap*!* representation of a list on output is meant for complicated lists of various kinds of things, including lists of lists, and it is a bit heavy if what you want is a sentence. A Snap*!* list doesn't look anything like a kid's idea of a sentence, which is simply a bunch of words next to each other.



Also, Logo has two different input notations for lists: the square brackets for a *constant* list, and a LIST reporter, just like Snap*!*'s, that allows for arbitrary expressions to compute the items:

```
(LIST "Sandy "Dale "Dana "Chris)
```

Snap*!* lacks the simpler input notation for a constant list; the only way to input a list is with the LIST reporter.

Grouped Sequence of Words

Snap*!* doesn't have words and sentences as primitive data types; instead it has the lower-level idea of text strings. Thus the sequence of verbs

```
[cheats [loves to walk] yells]
```

doesn't require the inner brackets when given as three text strings in Snap*!*. But there is a cost to that: you lose the information that the second verb is made of three words. Here is why that matters: Those verbs are all in the third person singular form, with an "s" at the end of ... of the verb? Not if you consider "loves to walk" a verb. The "s" comes at the end of the first word of the verb.

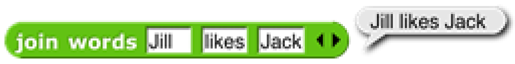
Now suppose you want to write a procedure to convert these verbs to third person plural form. To simplify the discussion, let's just convert one verb, and then map that function over the list of verbs. So:

```
to pluralize :verb
  if wordp :verb [output butlast :verb]
  output sentence (butlast first :verb) (butlast :verb)
end
```

Doing the same thing in plain Snap*!* (without the Word-Sentence library) is a nightmare of searching for space characters, focusing attention on computers instead of on language.

Combining Words to Form Sentences

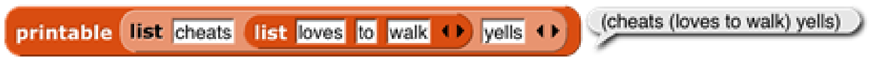
Speaking of focusing attention on computers, it is actually kind of annoying to have to put those explicit spaces between words, so you need 2n-1 inputs to JOIN for n words. That's why the Word-Sentence library has a JOIN WORDS block that takes care of that for you.



The "Words and Sentences" Library

In Logo, the representation of sentences as lists of words was designed so that both the programming (in terms of words, not characters) and the visual representation (looking just like text) fit with kids' habitual experience of words and sentences. In Snap*!*, we inherited from Scratch a low-level notion of character strings. So we have to choose between strings, which look like sentences but are a pain to program with, and lists, which allow Logo word-sentence programming techniques but do not look anything like sentences.

Among the Snap*!* libraries is a "Words and Sentences" one that allows you to use the normal-looking representation (character strings) but apply Logo-style operations (e.g., all but first word) to them. And, on the other hand, it has a SENTENCE reporter that takes any combination of strings and lists, and combines them into one flat list of words, and an inverse operation PRINTABLE that takes a list and turns it into a text string, with sublists indicated, as in Logo (but with parentheses instead of square brackets).



One advantage that Snap*!* does have over Logo is that the user does not have to type procedure names.

Instead of Logo's BUTFIRST function that works differently on words and sentences, we can have **"all but first letter of ___"** for words and **"all but first word of ___"** for sentences.