Isometric Generative Grammars

Leon Rische

[2019-08-08 Thu 12:38]

Contents

1 Context

I've started reading "Paradigms in Artificial Intelligence Programming" and in chapter 2, a program for generating random sentences using a subset of english grammar is presented.

One of the exercises is to extend this program to generate programs in a different language.

language=Lisp,label= ,caption= ,captionpos=b,numbers=none (defun one-of (set) "Pick one element of set, and make a list of it." (list (random-elt set)))

(defun random-elt (l) "Choose an element from a list at random." (elt l (random (length l))))

(defun sentence () (append (noun-phrase) (verb-phrase))) (defun noun-phrase () (append (Article) (Adj*) (Noun) (PP*))) (defun verb-phrase () (append (Verb) (noun-phrase))) (defun Article () (one-of '(the a))) (defun Noun () (one-of '(image lisp program sentence grammar programmer))) (defun Verb () (one-of '(expanded generated processed programmed))) (defun PP () (append (Prep) (noun-phrase))) (defun Adj () (one-of '(big beautiful isometric complex intricate convoluted meaningless))) (defun Prep () (one-of '(to in by with on)))

(defun Adj* () (if (= (random 2) 0) nil (append (Adj) (Adj*))))

```
(defun PP* () (if (= (random 2) 0) nil (append (PP) (PP*))))
(defun print-sentence (s) (format t "  ( a )
(print-sentence (sentence))
```

the program expanded the grammar in a complex grammar

a sentence programmed a complex convoluted grammar

a lisp generated a image

2 A Grammar for Isometric Objects

Working with isometric objects, the "Nouns" (objects) are cuboids and groups, lists of either cuboids or other groups.

Groups and cuboids can be modified using transformations:

- `swap-xy` swaps the x and y coordinate of the object / all objects in the group
- `swap-xz`
- `swap-yz`
- `translate-repeat` Generates a group by translating an object multiple times
- `mirror-x` mirrors the object along the x axis
- `mirror-y`
- `mirror-z`

```
object -> (random-cuboid)
modified-group -> (modification modified-group) group
group ->
  modified-group
 (group modified-object group)
  random-cuboid
```

3 Two Paths

When implementing these kinds of sub-languages in Lisp, there are two options:

- Write them in plain lisp
- Write a interpreter for the language

In the English grammar example, the second approach makes more sense since adding a new set of rules takes less effort.

In the second example, I'm currently generating a tree structured program of transformations, groups & cuboids, so implementing it in plain Lisp makes more sense.

TODO, Is it viable to always use groups as main element? Then I would only need to distinguish between operations producing groups and operations on groups.

(functions of zero or one argument)

The only operations producing groups are "combinations" and the initial "random-cuboid" which could be rewritten as a "add-random-cuboid" operation on a group.