Sequential Logic Grammars

# Abstract

Existing solutions for procedural content generation primarily fall into one of two categories: systems which operate as dynamic agents and systems which generate content with a static single pass. In the middle is a growing field of hybrid generation systems, and in this paper we present the technique of sequential logic grammars, a grammar-based technique which makes use of sequential logical operations in order to allow

# Overview

A sequential logic grammar consists of structured rules for the creation of an object, applied in sequential fashion, with each layer depending on the previous layers. Unlike standard grammars, they are not inherently recursive; like standard grammars, they involve multiple sequential steps and have a tendency to produce semi-fractal structures. In effect, a sequential logic grammar consists of a *series* of grammars in which the steps are temporally dependent in a linear fashion.

Each stage in the sequential grammars we have implemented follows the same procedure. First, a genotype is generated from the rules of the step’s grammar; this genotype is then combined with a series of restrictions and modifications which affect its final form. For instance, a grammar intended to produce stairs might have a possibility of producing a set of them in up to four locations per level; in a four-level structure with roof access, it might place three on the first level, two on the second level, none on the third level, and two on the fourth level. Following the determination of these stair locations, restrictions may be applied; for instance, if the stairs run along the outer walls, it might be inappropriate to place them such that they create a “floating” door. Because doors are dynamically generated by a prior layer (the process is reversed for a posterior layer), the constraint of “do not spawn stairs beneath doors” cannot itself be coded into the grammer; instead, it must be applied dynamically at runtime. For this purpose, a simple Boolean check will suffice. If a door has already spawned on the east wall on Floor 3, the stairs on the east wall on Floor 2 should be removed from the genotype. In practice this is done by logical combination of two vectors of Booleans using a mapping of the logical AND operation – a staircase may only spawn if generated by the grammar AND allowed by the constraints.

It is relatively easy to specify that a floor must have between one and four sets of stairs; indeed, one can do it by simple random population of four Booleans. It is also relatively easy to specify that stairs may not spawn immediately below a door; one may do this by using the logical formula *stairs[floor][wall] && !door[floor + 1][wall]*, where *stairs* is the