48-747 Shape Grammars

A Course on Shape, Computation and Languages of Design

Spring Semester • **9-12 units** • Pittsburgh: TR 3:00-4:20pm (TBD) **Doha: UT 3:00-4:20pm (CMB 2147)**

Computational Design graduate students must register for 12 units (when offered in Pittsburgh)

Instructor: Ramesh Krishnamurti • ramesh@cmu.edu

Course website: http://www.andrew.cmu.edu/~ramesh/teaching/course/48-747

Syllabus

Motivation

The purpose of this course is to introduce spatial grammars and their applications, primarily to design and composition. The emphases, for the most part, will be on the formal and informal aspects of grammars, evolution of grammatical ideas, their relevance, application and use in the analyses of 'styles', synthesis of 'form', and incorporation of 'function', and not least, in how to teach grammars to a computer.

Spatial grammars have their origin in formal grammars, which are rule-based systems that can be used for composition. Grammatical approaches to design offer an alternative to traditional approaches. The goal of grammars is not merely to produce a single design as the final outcome, but, rather, to provide an understanding of the underlying spatial relations that come into play in an eventual design. For nearly three decades, grammars have been used extensively to understand styles of architecture, landscape design, fine art and ornament. More recently, there has been an increasing application of grammatical ideas to many other disciplines.

Course Objective

A particular kind of grammatical formalism, called *shape grammars*, will be examined in detail. Shape grammars encapsulate spatial compositions of designs within a certain style as a *language*.

Course Description

This course is essentially divided into two parts. The first part explores the grammar paradigm and its application to architecture, urban design, and design. The second part looks at more advanced topics in grammar work. The course is divided into a *series* of lectures, each based on a theme (or topic) and will essentially, taking the format of a discussion that I will lead and expect you to participate in. The lecture topics will cover some of the following aspects of grammars. The choice of the topics, particularly in the second half of the course, will be determined by the strengths of the students in the class.

- Fundamental notions of grammars and rewriting systems, languages of grammars
- Introduction to shape grammars and their properties
- Introduction to geometric transformations, patterns and symmetries in the plane
- Transformations of grammars
- Colour grammars
- Kinds of spatial grammars
  e.g., set, structure, solid grammars, graph and other types of grammars
- Application of grammars to aspects of architecture and urban design

- Application of grammars to other disciplines
- Weights, *sorts*, and augmented spatial grammars
- Implementing shape grammars [and other spatial grammars]
- Recent trends in grammar research
- Other topics suggested by students

Although, typically, the topics above constitute most of the lecture material, I have a policy of keeping the last third of the second half of the course open to the particular interests in the subject matter of the students registered for the course.

Readings and Reference

Shape by George Stiny, MIT Press.

There is no textbook for this course. I will prepare a reader of seminal and important articles, which will include an extensive bibliography of articles on grammars some of which I will use as the source material for the lectures. Slides, if any, used in the lectures can be obtained from me or from the course web site (under construction)

Course Credit – 9 units

When offered in Pittsburgh, for Computational Design students, this is a 12-unit course

The course is divided into a lecture course (6 units), which supplies the basis of understanding grammatical ideas and construction and a project course (3 units), which is regarded as the starting point for pursuing individual research into advanced generative processes and systems.

Computational Design students have to complete **an additional 3 unit** programming assignment.

Course Requirements

Irrespective of how students are initially registered for this course – students will be evaluated on the following requirements.

- Assignments (5 units)
- In class participation (1 unit)
- Project (3 units)

- Programming Assignment (3 units)

The programming assignment is **only required of all Computational Design students**.

Other students may opt to take the programming assignment instead of doing a project, or additionally for 3 extra units.

Grading

Ideally, no student should take a elective course for a grade, but *out of interest* or *curiosity*.

If you show a real affinity and flair for the subject, you should expect an A. If you show a basic level of competence and interest, you should expect a B —otherwise, this is not a course for you. Having said this, the following grading scheme seems to fit this objective. Passing grades are awarded according to the following scale (A 90+ B 80+ C 70+) with the following caveat:

- A (excellent) means that you have scored at least B or better in each assignment.
- B (good) means that you have at least successfully completed (passed) each assignment.
- C means you do not fail the course, and you either you performed uniformly badly, or failed in one of the assignments.

My marking will be objectively fair as humanly possible. You are at liberty to question my marking, but only once. I have a tendency to mark *hard assignments leniently* and *easy assignments hard*.

- *Written* parts for any assignments must be *grammatical*.
- *Drawn* parts for any assignment must be *neat* (if you can't freehand neatly, use a computer drawing program).
- *Programmed* parts or algorithms for any assignment must have *a clear logic*.
- *Presentations* must be *visual*

Prerequisites

Computational Design students are expected to have programming experience.

All other students with a programming background are eligible to take the 12 units version of the course **when offered**, or substitute the project requirement with the programming assignment.

Schedule

Will be developed on a week-by-week basis