ARCH 7327 Developing Computational Solutions for Design Problems (Spring 2026) - Syllabus

Instructor:

Yefan Zhi, Ph.D. Candidate, yefanzhi@design.upenn.edu

Background

Students at Weitzman can benefit from a rigorous course in computational design methods. Previous such courses at Weitzman have been topic-based, meaning students targeted specific design results first, and only absorbed skills they found relevant. The results might have been fascinating, but it is challenging for students to internalize and apply such skills in professional practice. As a PhD candidate and a computational design expert, I propose this fundamental course to prepare students for general computational design and design computation tasks. It emphasizes both the underlying mathematics and the hands-on practices, providing a clear, comprehensive, and stimulating structure for computational design methods.

A course with a similar focus was taught by Mostafa Akbari and Yao Lu (both PhD candidates) in fall 2023. I have discussed with them and collected feedback from some students who took the course. Some key changes in the proposed course are:

- The new course, taught by only one instructor, will have a more consistent structure from small to large. It will investigate primary geometry types, their processing methods, and finally, spatial data structures and algorithms.
- The course will introduce some fundamental algorithms and the concept of computational complexity to provide a high-level view of the potentials and limitations of computational design. Such a view is crucial to the era of artificial intelligence and mass customization.
- The final project will not require physical models, so that students can focus more on developing the computational solutions. Fabrication-aware design methods will still be discussed and valued.

1. Course Description

Developing Computational Solutions for Design Problems is a seminar for architecture students who are motivated to develop and utilize advanced computational tools to address design problems. The course examines the essential mathematics, data structures, and algorithms of the interdisciplinary practice of computational design. With hands-on Grasshopper and GhPython workshops, it equips students with foundational skills to solve design problems by developing *agile and versatile* computational tools. At the end of the course, students will develop standalone tools for design applications with the instructor's assistance.

The course is organized in four components: (i) geometric data types (vectors, planes, transformations, curves, surfaces, etc.); (ii) geometric processing (graph theory, polysurfaces, meshes, etc.); (iii) spatial data structures and algorithms (tessellation, L systems, introduction to algorithms and complexity, etc.); and (iv) the final project.

The instructor will also demonstrate Ovenbird, the 3D printing slicing software he developed, as a case study of a comprehensive tool from concept to prototype and product.

2. Course Objectives

- Examine essential knowledge of computational geometry for architectural practices.
- Introduce a wide range of topics in computational design.
- Introduce visual programming in the Rhino/Grasshopper environment, with the assistance of GhPython scripts.
- Connect the computational design paradigm with the design practices and develop computational thinking.
- Investigate ways of developing and utilizing integrated computational tools to enhance the design practice.

3. Course Methods

The course will be conducted as a mix of lectures, hands-on workshops, and discussions. The lectures will not be specific to any software or programming language. The hands-on workshops and assignments will utilize Grasshopper and GhPython. At the beginning of each class, an ungraded mini-quiz is presented to review the content from the last week.

Students will work individually on their script assignments and in groups of one or two for the final project.

The course would comprise 30% lectures, 45% hands-on workshops, and 25% discussions and desk critiques.

Expected number of students: 12 to 18.

4. Readings and Bibliography

[1] R. Issa, "Essential Mathematics for Computational Design." [Online]. Available: https://developer.rhino3d.com/guides/general/essential-mathematics/

[2] H. Pottmann, A. Asperl, and A. Kililan, *Architectural Geometry*. Philadelphia, PA: Bentley Institute Press, 2007.

[3] A. Tedeschi, *AAD Algorithms-Aided Design*. Brienza, Italy: Le Penseur, 2014.

Detailed reading materials will be provided per lecture.

5. Student Deliverables and Requirements

There are five assignments for this course.

Assignment	Description	Type	Deliverables	Time	Evaluation
Script 1	Following the instructions, create a Grasshopper script that automates a design task. GhPython should not be used.	Individual	.gh file	WK3-WK4	20%
Script 2	Following the instructions, create a Grasshopper script that automates a design task. The core function should be performed by GhPython scripts.	Individual	.gh file	WK7-WK10	20%
Proposal 1	Discuss a design problem that you have encountered in your courses/practices that can be automated by computational tools.	Individual	Word/PDF, 1-2 pages	WK1-WK3	0%
Proposal 2	Introduce the final project. Briefly review present tools/studies of similar tasks. Give an overview of the outcome.	In groups	Word/PDF, 2-4 pages	WK10-WK11	10%
Final project	Demonstrate the final project. Discuss the obstacles and solutions. Show outcomes. Discuss the limitations.	In groups	.gh files, Slides, Video	WK10-WK14	40%

On WK4 the instructor will group students based on their interests and backgrounds. Students can also voluntarily form groups.

6. Grading/Evaluation

- Attendance and Participation - 10%
- Assignments
  - Script 1 - 20%
  - Script 2 - 20%
  - Proposal 1 - 0%
  - Proposal 2 - 10%
  - Final project - 40%

7. Detailed Weekly Schedule

	Lecture	Workshop	Assignment
	I. Geometric Data Types		
WK1 Jan 15	• Introduction: Computational design in practice	• GH: Components and I/O	• Proposal 1
WK2 Jan 22	• Points, vectors, planes • Transformations • Degrees of freedom • Discrete design space	• GH: DataTree management	
(Jan 27)	Course selection period ends		
WK3 Jan 29	• Curves and surfaces • Hierarchies of geometric data • Boolean algebra	• GH: Parameterization of curves and surfaces • GH: Construction and deconstruction • GH: Conditions	• Proposal 1 DUE • Script 1 (GH)
	II. Geometric processing		
WK4 Feb 5	• Discussion of Proposal 1 • Graph theory 1 • Boxes, Polysurfaces, Convex hulls	• GH: Polysurface operations • Case study: Fabrication rationalization of frame structures	• Script 1 DUE
WK5 Feb 12	• Graph theory 2 • Meshes • SubDivisions	• Case study: Mesh heatmaps • Case study: Mesh deformations • Discussion of Script 1	
	III. Spatial data structures and algorithms		
WK6 Feb 19	• Tessellation • Triangulation, Voronoi diagrams • Grid-based methods	• GhPython: Introduction • GhPython: Code structure • GhPython: Data types	
WK7 Feb 26	• Introduction: Algorithms • Computational complexity	• GhPython: Lists and loops • GhPython: Functions	• Script 2 (GhPython)
WK8 Mar 5	• L systems and fractal geometry	• GhPython: Recursive algorithms	
WK9 Mar 12	Spring Break (no class)		
	IV. Final Project		
WK10 Mar 19	• Case study: 3D printing with Ovenbird	• GH: Clustering	• Script 2 DUE • Proposal 2 • Final project
WK10 Mar 26	• Topics related to final projects	• Desk critique	• Proposal 2 DUE
WK11 Apr 2	• Topics related to final projects	• Desk critique	
WK12 Apr 9	• Topics related to final projects	• Desk critique	
WK13 Apr 16	• Topics related to final projects	• Desk critique	
WK14 Apr 23	• Presentation of final projects		• Final project DUE