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

Instructor

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

Meeting Pattern and Location

Thursdays 1:45 pm-4:44 pm

Fisher-Bennett Hall 16

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 primitive and advanced 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.

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) primitive geometric data types (vectors, planes, transformations, curves, surfaces, etc.); (ii) advanced geometric data types (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.

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.

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 comprises 30% lectures, 45% hands-on workshops, and 25% discussions and desk critiques.

Expected number of students: 12 to 18.

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.

Assignments and Deliverables

There are five assignments for this course.

Assignment	Description	Type	Deliverables	Evaluation
	Attendance and Participation			10%
Script 1 series	Following the instructions, create a Grasshopper script that automates a design task. GhPython should not be used.	Individual	.gh file	30%
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	10%
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	0%
Proposal 2	Introduce the final project. Briefly review present tools/studies of similar tasks. Give an overview of the outcome.	Individual or in groups of 2	Word/PDF, 2-4 pages	10%
Final project	Demonstrate the final project. Discuss the obstacles and solutions. Show outcomes. Discuss the limitations.	Individual or in groups of 2	.gh files, Slides, Video	40%

Grading Criteria

- Attendance and Participation - 10%
- Assignments
  - Script 1 series - 30%
  - Script 2 - 10%
  - Proposal 1 - 0%
  - Proposal 2 - 10%
  - Final project - 40%

Academic Integrity Statement

All students are expected to adhere to the University of Pennsylvania's Code of Academic Integrity.

A.I. Use Policy

A.I. tools might be used for writing proposals and scripts. If used, the student should not copy and paste text/code directly from A.I. tools. Instead, the student should understand the logic and re-write the text/code in their own style. The student should also document the use of A.I. tools in the assignment submission and submit a PDF copy of each dialogue.

The student should be responsible for every line of text/code they submit. Any text/code that the student cannot explain during office hours or critiques will be considered a violation of the academic integrity policy.

Graphic assets (images, diagrams, etc.) generated by A.I. tools should not be used in any submission or presentation.

Detailed Weekly Schedule

	Lecture and Workshop	Assignment
	I. Primitive Geometric Data Types	
WK1 Jan 15	**Introduction, GH IO** • Introduction: Computation, computation and design • Components and I/O	• Proposal 1
WK2 Jan 22	**Vectors, Planes, DataTrees** • Points, vectors, planes • Vectors, planes • Transformations • Transformations • Data types • Lists and DataTree management	
(Jan 27)	Course selection period ends	
WK3 Jan 29	**Lists, Curves, Surfaces, Intersections** • Polylines and NURBS curves • Boolean algebra and Conditions • List and list operations • Surfaces • Proximity, intersections, and attractors • Conversion between points, vectors, curves, planes, and surfaces	• Proposal 1 DUE • Script 1 (GH)
	II. Advanced Geometric Data Types	
WK4 Feb 5	**Boxes, Breps, Convex Hulls, Python**	

	Lecture and Workshop	Assignment
	• Discussion of Proposal 1 • Boxes, bounding boxes, breps, convex hulls (2D, 3D) • Case study: Fabrication rationalization of frame structures • Python: Introduction, code structure, data types	
WK5 Feb 12	**Graph Theory, Mesh** • Graph theory • Meshes • Case study: Mesh heatmaps and deformations	• Script 1 DUE
	III. Spatial Data Structures and Algorithms	
WK6 Feb 19	**Grids and Tessellations, GhPython** • Discussion of Script 1 • Grid-based methods • Tessellations • Triangulation, Voronoi diagrams • From Python to GhPython: use GhPython libraries	• Script 1b
WK7 Feb 26	Travel week (no class)	
WK8 Mar 5	**Algorithms, Complexity, Functions** • Introduction: Algorithms • Computational complexity • GhPython: Lists and loops • GhPython: Tree helpers • GhPython: Functions	• Script 1b DUE • Script 2 (GhPython)
WK9 Mar 12	Spring Break (no class)	
WK10 Mar 19	**L Systems, Flowchart, Proposal 2** • Recursive algorithms: Euclidean algorithm • L systems and fractal geometry • Flowchart	• Script 2 DUE • Proposal 2
	IV. Final Project	
WK11 Mar 26	**Case Study, Clustering** • Case study: 3D printing with Ovenbird • Grasshopper clustering • Desk critique	• Final project
WK12 Apr 2	• Desk critique	• Proposal 2 DUE
WK13 Apr 9	• Desk critique	
WK14 Apr 16	• Desk critique	
WK15 Apr 23	• Presentation of final projects	• Final project DUE