GSD-6338:
INTRODUCTION TO COMPUTATIONAL DESIGN

Harvard University Graduate School of Design, Fall 2024.
Lectures & Workshops: Tuesdays and Thursdays 12:00 – 1:15 pm ET, Gund 516.
Sections: Wednesdays 6:00-7:15 pm ET, Gund 516 (TBD)

TEACHING STAFF

Instructor:

Dr. Jose Luis García del Castillo y López
jgarciadelcasti@gsd.harvard.edu

Teaching Fellows:

Sang Won Kang
sangwonkang@gsd.harvard.edu

Teaching Assistants:

Una Liu
unaliu@gsd.harvard.edu

Fatima Mamu
fatimamamu@mde.harvard.edu

Sophia Cabral
scabral@gsd.harvard.edu

Office Hours with the Teaching Staff can be booked on https://appt.link/gsd-6338

DESCRIPTION

#GSD6338 is an introductory course on Computational Design, with particular focus on architecture, landscape and urbanism.

In this course, we will understand "Computational Design" as the set of methods borrowed from fields such as computer science, mathematics and geometry, applied to solving design problems. Chances are that a significant portion of your typical design workflow is mediated by digital tools and, in particular, computer software that has been designed and created by a third party, and therefore, your creativity is partially biased by someone else's opinions. However, the real craftsman is the one who understand their tools so well that they can change, improve and adapt them to their own desires. In this course, you will learn how to think algorithmically, and how to understand and create computer software, so that you will be able to explore new creative opportunities and relate them to your personal interests.

The course will offer student the possibility of becoming familiar with the process of programming in a creative context, as the power of computational media will be revealed through examination of code and data as a medium for creative expression.

WHY THIS COURSE?

In this course, you will learn:

- ***Computer programming***: the fundamentals of writing computer code, and designing algorithms.
- ***Computational geometry***: advanced topics in complex geometry generation and manipulation.
- ***CAD to CAM***: how to translate abstract geometrical information to machine code that can be executed by digital fabrication tools such as 3D printers or industrial robots.
- ***Data manipulation***: how to generate, read, transform and store streams of data in different common formats.
- ***Fundamentals of software development***: how to write scripts that integrate into common design environments, simple programs that can be executed stand-alone, or a combination thereof.
- ***Creative technology***: how to use code as a platform for the creation of expressive media within architecture, design and the arts.
- ***Critical thinking***: within the creative fields, which problems are well suited to be addressed with computational techniques? Which ones are human brains still better at?

This class is ideal for architects, engineers, designers, artists and/or "creative technologists," students of professionals alike, who are interested in discovering the vast range of expressive potential that computation can provide for their practice. This is particularly true for those with special focus on complex geometry, digital fabrication, art installations, academic research, etc.

PREREQUISITES

There are no prerequisites to take this course.

General familiarity with 3d modeling using Rhinoceros and parametric modeling using Grasshopper is desirable. If a prospective participant has no experience with any of these, they will be required to complete a set of online tutorials during the first week of the semester. General familiarity with design processes is desirable, although this will be practiced through the homework assignments.

Previous knowledge of computer programming is NOT required; this is part of what you will learn in this course!

Course is open to cross-registration petitions. The course is not open to auditors.

STRUCTURE

The course will be structured around the following instructional elements:

- **Lectures**: high-level descriptions of the topics covered in the class, led by the instructor. Attendance is mandatory.
- **Workshops**: hands-on sessions covering the technical aspects of the class, led by the instructors. In-class laptops will be required from students. Attendance is mandatory.
- **Student Presentations**: class-wide reviews of student's assignments and/or final projects, facilitated by the Teaching Staff and led by the students. Attendance is mandatory.
- **Sections**: review sessions, to practice or extend the topics of the lectures/workshops. Led by the Teaching Fellow/s. Attendance is optional.

WORK

Evaluations for this class will be entirely based on the work you develop within it, i.e. no final quiz or tests. For this matter, there will be four different types of homework:

- ***Lecture Reviews***: these will be short quizzes with questions about the content from main lectures, designed to help you review the most important topics of the lecture. These will not be evaluated; students are expected to independently complete them and use them as a learning resource.
- ***Workouts***: these will be short, concise exercises designed to train a particular topic we may have touched upon in class. Think of these as brain workouts, designed to pump up your coding skills. These will not be evaluated; students are expected to independently complete them and use them as a learning resource.
- ***Assignments***: a few assignments will be presented in class, in which you will be asked to apply your programming skills to a small design task. These assignments will have a bigger scope, and allow you some expressive freedom in a creative context. Students will conclude these assignments with small presentations in class.
- ***Final Project***: we will spend the last month of this course working on a personal final project. You will choose a topic of your own interest, and develop a complete, coherent design project worthy of the cover of your portfolio! This will be the most important part of this course, and we will work very hard on it.

Learning to program requires serious dedication and a lot of time designing, coding, testing, debugging, and revising. Programming is an incredibly iterative process. Expect to spend a considerable amount of time outside of class working on your programs. Programming is time consuming and mysterious bugs can send you on a tailspin at any moment, thus, if you begin working on a program immediately after it's assigned, you'll have enough time to seek help in the event you get stuck.

This course assumes an elevated degree of student initiative and independent work. Links to additional resources will be consistently provided throughout the semester, and class participants are expected to conduct independent research on topics of their interest, amplifying the scope of their learning and practice within this class. Considerations of merit will be strongly based on this aspect.

If you want to get a glimpse of what your Final Project may look like, feel free to check out previous student work on http://gsd6338.org.

GRADING

Grading criteria, deadlines and submission guidelines will vary depending on the task at hand, and will be specified in each particular handout. In general, you can expect the following:

- If you stick to the class contents, submit what is strictly required in the assignments, and complete a final project using mostly standard available computational tools, you will likely be awarded a *Pass.*
- If you go beyond the scope of the material taught in class, incorporate some learnings from external resources in your practice, push the boundaries of assignments with your own contributions and personal views, and submit a final project with a significant code base written specifically for this purpose, you will likely be awarded a *High Pass*.
- If you consistently complement your learning with external resources, incorporate techniques or environments not discussed in class in your assignments, and submit a final project of excellence such that contributes to the research field and/or could have an actual impact on a real-world community of users, you might be eligible for a *Distinction*.

The final course grade will be distributed 50% between Assignments and 50% for the Final Project.

TOOLS

From a technical point of view, the main frameworks used in this class will be:

- ***McNeel Grasshopper3D***: the initial weeks of the class will use Grasshopper as an introductory platform for computational geometry. This was chosen due to its powerful geometry kernel, its gentle learning curve as a UI-based visual programming language, and its capacity to host different scripting environments. However, students are expected to quickly supersede the capacities of "vanilla" Grasshopper, and use it mostly as a GUI for the development of advanced scripts.
- The ***C# programming language***: in the initial weeks, we will also learn the fundamentals of computer programming using the C# language within the .NET framework. Many are the reasons behind this choice: high-level language with high performance, familiarity of the syntax with many other popular programming languages, but mostly, its seamless integration with most of today's common CAD environments.
- ***RhinoCommon***: over the course of the semester, we will slowly bring these two together, and start writing our own scripted C# components in Grasshopper. We will rely strongly on RhinoCommon, Rhino's geometry kernel, accessible through Grasshopper's scripts components.
- Finally, the class will evolve to demonstrate possible applications of these techniques in other C#-based environments such as stand-alone code libraries (DLLs), compiled components for VPLs (such as Grasshopper), video game engines (such as Unity), etc.

SCHEDULE

This is a project-based, hybrid lecture/workshop course. Most of the learning will happen though a combination of instructor-led conceptual lectures, and hands-on workshop sessions. Class meetings will start each week with a presentation introducing the topic of the week and reviewing the programming exploration assigned the previous week. The balance of class time will consist of instructor-led hands-on exercises and/or independent work.

Attendance is mandatory for all Lectures, Workshops and Student Presentations. Attendance is optional for Sections.

The tentative schedule for this semester looks as follows (please expect last minute changes and adaptations to the pace of the class):

WEEK	DAY	TIME	TITLE	TYPE
#0	Thursday, September 5, 2024	12:00-1:15pm	**COURSE PRESENTATION**	Lecture
#1	Tuesday, September 10, 2024	12:00-1:15pm	**DATUM, DATA & POINTS**	Lecture
	Wednesday, September 11, 2024	6:00-7:15pm	**SECTION: GRASSHOPPER REFRESHER**	Section
	Tuesday, September 12, 2023	12:00-1:15pm	**HELLO WORLD!**	Workshop
#2	Tuesday, September 17, 2024	12:00-1:15pm	**FLOW CONTROL**	Workshop
	Wednesday, September 18, 2024	3:00-4:15pm	**SECTION: ITERATION**	Section
	Thursday, September 19, 2024	12:00-1:15pm	**VECTORS**	Lecture
#3	Tuesday, September 24, 2024	12:00-1:15pm	**DATA STRUCTURES**	Workshop
	Wednesday, September 25, 2024	6:00-7:15pm	**SECTION: VECTOR ALGEBRA**	Section
	Thursday, September 26, 2024	12:00-1:15pm	**DATA COLLECTIONS**	Workshop
#4	Tuesday, October 1, 2024	12:00-1:15pm	**FUNCTIONS**	Workshop
	Wednesday, October 2, 2024	6:00-7:15pm	**SECTION: RECORDING MOTION**	Section
	Thursday, October 3, 2024	12:00-1:15pm	**GEOMETRY KERNELS**	Workshop
#5	Tuesday, October 8, 2024	12:00-1:15pm	**FUNCTIONAL GEOMETRY**	Lecture
	Wednesday, October 9, 2024	6:00-7:15pm	**SECTION: VISUALIZING MOTION**	Section
	Thursday, October 10, 2024	12:00-1:15pm	**CURVES**	Lecture
#6	Tuesday, October 15, 2024	12:00-1:15pm	**ASSIGNMENT[0] REVIEW**	Student presentations
	Wednesday, October 16, 2024	6:00-7:15pm	**SECTION: CURVES & SURFACES**	Section
	Thursday, October 17, 2024	12:00-1:15pm	**SURFACES**	Lecture
#7	Tuesday, October 22, 2024	12:00-1:15pm	**MESHES**	Lecture
	Wednesday, October 23, 2024	6:00-7:15pm	**SECTION: MESHES**	Section
	Thursday, October 24, 2024	12:00-1:15pm	**FABRICATION**	Lecture
#8	Tuesday, October 29, 2024	12:00-1:15pm	**ASSIGNMENT[1] REVIEW**	Student presentations
	Wednesday, October 30, 2024	6:00-7:15pm	**SECTION: 3D PRINTING**	Section
	Thursday, October 31, 2024	12:00-1:15pm	**OBJECT-ORIENTED PROGRAMMING**	Workshop
#9	Tuesday, November 5, 2024	12:00-1:15pm	**CODE LIBRARIES**	Lecture
	Wednesday, November 6, 2024	6:00-7:15pm	**SECTION: 3D SCANNING**	Section
	Thursday, November 7, 2024	12:00-1:15pm	**CONTINUOUS EXECUTION I**	Workshop
#10	Tuesday, November 12, 2024	12:00-1:15pm	**ASSIGNMENT[2] REVIEW**	Student presentations
	Wednesday, November 13, 2024	6:00-7:15pm	**SECTION: INTERACTIVE ART**	Section
	Thursday, November 14, 2024	12:00-1:15pm	**CONTINUOUS EXECUTION II**	Workshop
#11	Tuesday, November 19, 2024	12:00-1:15pm	**TBD**	TBD
	Wednesday, November 20, 2024	6:00-7:15pm	**SECTION: UNITY PROJECT**	Section
	Thursday, November 21, 2024	12:00-1:15pm	**CONTINUOUS EXECUTION III**	Workshop
#12	Tuesday, November 26, 2024	12:00-1:15pm	**TBD**	TBD
	Wednesday, November 27, 2024	6:00-7:15pm	**(Thanksgiving recess)**	(No class)
	Thursday, November 28, 2024	12:00-1:15pm	**(Thanksgiving recess)**	(No class)
#13	Tuesday, December 3, 2024	12:00-1:15pm	**FINAL PROJECT PIN-UP**	Student presentations
	Wednesday, December 4, 2024	6:00-7:15pm	**FULL CODE CLINIC**	Office Hours
	Thursday, December 5, 2024	12:00-1:15pm	**(Studio Reviews)**	(No class)
#14	Dec 13-18, 2024	TBD	**FINAL REVIEW**	Show & Tell

REFERENCES

There are several online resources which will be helpful for this class, but of special importance will be ***ParametricCamp***, the YouTube channel created and maintained by the head instructor:
http://youtube.parametric.camp.

The channel contains multitude of live streams and tutorial videos related to computational design, organized in series such as:

- **Introduction to Parametric Modeling**: a beginner series on parametric modeling using Rhinoceros & Grasshopper:
  http://introtoparametricmodeling.parametric.camp
- **Learning C#: Introduction to Computer Programming for Designers**: an introductory series to computer programming using the C# programming language:
  http://learningcsharp.parametric.camp
- **Geometry Gems**: a collection of short tutorials covering the algorithms underlying basic geometrical operations and transformations:
  http://geometrygems.parametric.camp
- **Algorithmic Modeling Challenges:** a collection of tutorials breaking down how to perform complex modeling projects, both in vanilla Grasshopper and C# scripting:
  http://algorithmicmodelingchallenges.parametric.camp
- Advanced Development in Grasshopper: a full playlist on advanced programming techniques in GH, code your own plug-ins!
  https://www.youtube.com/playlist?list=PLx3k0RGeXZ_yZgg-f2k7fO3WxBQ0zLCeU
- **Live Streams:** all the live streams so far!
  http://livestreams.parametric.camp

If you are geeky enough, you should definitely follow Daniel Shiffman's YouTube channel ***The Coding Train***: https://www.youtube.com/thecodingtrain. It is full of amazing videos covering a lot of topics of basic programming, Processing, p5.js, and fantastic examples of creative things you can do with code.

There is no textbook required for this class. Readings will be assigned from resources available online or handouts provided through the course website. The following resources will be used for reference throughout the semester:

- Ko, Joy and Kyle Steinfeld. **Geometric Computation: Foundations for Design.** Routledge, 2018. https://doi.org/10.4324/9781315765983. *Available in Loeb Library.*
- Issa, Rajaa. **Essential Mathematics for Computational Design.** McNeel, 2019 (4th Edition). *This book can be downloaded for free from https://www.rhino3d.com/download/rhino/6/essentialmathematics*
- Woodbury, Robert. **Elements of parametric design.** Routledge, 2010. *Available in Loeb Library.*
- Pottmann, Helmut, et al. **Architectural geometry**. Vol. 724. Exton, PA: Bentley Institute Press, 2007.

Additionally, there are some other resources that can be useful to expand your horizon and extend your knowledge:

- Petzold, Charles. **Code: The hidden language of computer hardware and software**. Microsoft Press, 2000.
- Reas, Casey, and Ben Fry. **Processing: a programming handbook for visual designers and artists**. Vol. 6812. MIT Press, 2007. 2nd edition is also good for this class. *Available in Loeb Library.*

- Bohnacker, Gross, Laub, Lazzeroni. **Generative Design: Visualize, Program and Create with Processing.** Princeton Architectural Press, 2012.
- Shiffman, Daniel, Shannon Fry, and Zannah Marsh. **The nature of code**. D. Shiffman, 2012. *This book can be downloaded for free from http://natureofcode.com*

Elizabeth Patitsas maintains a collective reading list on *Equity in Computing*, with the hope to advance the understanding of "the social context of computing education, with particular attention to how and why various groups are marginalized in computing (women, racial minorities, low-SES, Indigenous peoples, people with disabilities, etc)." You can read more and find additional resources on https://tinyurl.com/criticalCS

ORIGINAL WORK, USE OF 3RD PARTY SOURCES, AI-ASSISTED CONTENT, CREDITING & PLAGIARISM

For your code exploration assignments and project work you are expected to write original code. You must be able to explain how your program functions whenever asked. It is acceptable, and encouraged, to seek design, coding, and debugging assistance from other students, staff, or faculty in order to complete your work. Likewise, you are encouraged to help other students whenever possible. You are encouraged to study and learn from code examples. You may use third-party libraries to simplify coding, however, grading is based on the functionality of your original code along with the clarity of your comments and documentation. Whenever you incorporate third-party open-source materials in your work you must properly attribute the works.

When completing assignments that involve the use of AI-generated or assisted content—including, but not limited to, text, images, video, sound, point clouds, 3d models or code—students must ensure that proper credit is given to any sources used in their work. This includes both the original data used to train the AI model (where possible) and any specific AI algorithms, pre-trained models or tools utilized to generate content. In order to credit these sources appropriately, students should provide citations or references to any relevant papers, datasets, APIs or software libraries used. Additionally, if the student has made modifications to the output of an AI model or algorithm, it is important to describe these modifications and acknowledge the original source of the content. Proper attribution is critical to upholding academic integrity and avoiding plagiarism. Any questions regarding proper citation should be directed to the course instructor or teaching assistant.[1]

Any student submitting work that can be reasonably believed to not be their own will be reported to the Office of Student Services at the GSD for academic misconduct. Please review the Academic Integrity and Standards of Conduct section of the Student Handbook: https://www.gsd.harvard.edu/student-affairs/student-handbook/

Please check the GSD's guidance on the use of Generative AI: https://www.gsd.harvard.edu/resources/ai/

[1] Paragraph writing assisted by OpenAI's GPT-3 model via https://chat.openai.com/chat. Text prompt: "Write a paragraph for a course syllabus on how students should credit content in their homework that has been created assisted by an ai," queried on January 10th 2023. Text modified to include additional content types and sources.

LAPTOP, HARDWARE & SOFTWARE REQUIREMENTS

As per the GSD's requirements, "all GSD students are expected to bring to school a laptop capable of running Windows software." If you don't have a laptop, purchasing one now presents a hardship, or if your laptop is broken and out for repairs and you need one to use in class, please contact Help Desk for assistance: https://www.gsd.harvard.edu/information-technologies/

Having a computer setup with two screens is highly recommendable, to be able to better follow hands-on workshops.

The importance of backing up files cannot be stressed enough. Automated backup and cloud storage services should be strongly considered. Hardware failures and lost data will not constitute excuses to late or skipped dues.

ACCESIBILITY & COMMUNITY

The Harvard GSD is committed to providing an accessible academic community. If you will be in need of special accommodations in this class, please let the instructor and/or Student Services know as early as possible. More information on https://www.gsd.harvard.edu/admissions/admitted-students/students-with-disabilities/

This course strives to provide an inclusive, equal, respectful and conscious environment for each individual to feel welcome for who they are, and how they choose to express themselves; it is our shared responsibility, class leaders and participants alike, to maintain it this way. If you ever feel concerned about a disruption in this spirit, or have ever felt distressed yourself in any way, please reach out immediately to the class instructor and/or Student Services. Read more on https://www.gsd.harvard.edu/community-values-rights-and-responsibilities/

COPYRIGHT