**Module Description CDF 1**

**Module level:**  
Master

**Language:**  
English

**Module duration:**  
one semester

**Occurrence:**  
Winter Semester

**Credits\*:**  
6

**Total number  
of hours:**  
180

**Self-study  
hours:**  
120

**Contact  
hours:**  
60

\* The number of credits can vary depending on the corresponding SPO version. The valid number is always indicated on the Transcript of Records or the Performance Record.

**Description of achievement and assessment methods:**  
This module is structured around a sequence of frontal lectures and tutorials that delve into various aspects of computational design with a special focus on structural design, shape optimization, and digital fabrication. Each session focuses on specific topics, accompanied by student assignments in the form of programming exercises designed to reinforce learning objectives. These assignments are pivotal, as students are required to complete and submit them for evaluation. The module's grading criteria primarily assess the quality of these individual submissions, thereby emphasizing hands-on application and understanding.

**Possibility of re-taking:**  
In the next semester: No  
At the end of the semester: No

**(Recommended) requirements:**  
Basic knowledge of programming (e.g., in Python) and/or Rhino Grasshopper (software) is recommended but not necessary.

**Contents:**  
The course bridges fundamental principles of geometric computing and design and leverages these insights to develop new algorithms and tools for 3D shape generation, simulation, structural design, and digital fabrication in design and engineering. Through a blend of frontal lectures and immersive tutorials, students embark on a journey into innovative computational design solutions, emphasizing advanced fabrication and construction techniques across diverse scales. This interdisciplinary exploration draws upon the symbiotic relationship between various scientific disciplines, including mathematics, computer science, structural engineering, and architecture. The course curriculum is structured to provide a comprehensive understanding of key topics:

* Introduction to computational geometry, fostering algorithmic thinking and programming proficiency.
* Exploration of geometry processing, data structures, and interface intricacies crucial for effective computational design.
* Introduction to foundational concepts of structural- and fabrication-aware design, laying the groundwork for design solutions that integrate seamlessly into real-world applications.

**Study goals:**  
After attending the course, students will be able to:

* Learn and apply the basic principles of parametric and algorithmic design
* Implement basic versions of selected algorithms related to architectural geometry, structural design, and digital fabrication
* Use common CAD tools as interfaces to self-implemented solutions
* Gain basic knowledge through practical exercises in computational design
* Understand the scope and relevance of computational methods for architectural research and practice.

By seamlessly integrating theory with practical application and exposing students to diverse design tools, this course equips them with the necessary expertise to tackle contemporary design challenges effectively.

**Teaching and learning methods:**  
This module comprises a comprehensive series of lectures and tutorials with associated computer programming exercises. Emphasizing crucial aspects such as structural form-finding, shape optimization, and the utilization of digital fabrication technologies, the course provides a solid foundation for students to navigate the complexities of contemporary design processes. These foundational principles are further reinforced through hands-on assignments, ensuring practical application and deeper understanding.

The course structure unfolds as follows:

* Theoretical Foundations: Through frontal lectures, students delve into the theoretical underpinnings of Computational Design, gaining insights into fundamental concepts and methodologies.
* Practical Application: Students translate theoretical knowledge into practice through programming assignments, where they engage directly with computational tools to implement and experiment with various design techniques.
* Toolchain Exploration: A curated sequence of computational design tools is introduced and demonstrated using real-world examples. Students leverage these tools to complete assignments, gaining proficiency in software utilization while improving their design skills.

**Media formats:**  
Media-supported presentations (PowerPoint, videos, etc.) for the frontal lectures and a programming software platform for the tutorials.

**Literature:**  
References will be provided at the beginning of the semester

**Responsible for the module:**  
Dörfler K

**Lecturer:**  
D'Acunto P, Dörfler K, Hojjat M, Goldbach A