

# INTRALATTICE

CORE MODULES

---

## PREFACE

VERSION **0.7.5** — ALPHA

---

**INTRALATTICE** is a plugin for Grasshopper used to generate solid lattice structures within a design space, primarily for 3D printing. It was developed as an extensible, open-source alternative to current commercial solutions. As an on-going project developed at McGill's Additive Design & Manufacturing Laboratory (ADML), it has been a valuable research tool, serving as a platform for breakthroughs in multi-scale design and optimization.

By providing a modular approach to lattice design, and giving you full access to the source, we hope to collectively explore lattice design at a deeper level, and consequently, engineer better products.

**WEBSITE** — <http://intralattice.com>

**GITHUB** — <https://github.com/dnkrtz/intralattice/>

---

## THE MIT LICENSE

OPEN SOURCE

---

Copyright © 2015 ADML

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

## SUPPORT

---

If you have any issues, questions or feedback — please contact [support@intralattice.com](mailto:support@intralattice.com)

---

## LEAD DEVELOPERS

---

This project was started in the summer of 2014. Although the process was a collaborative effort, the contributions can be broadly categorized as follows:

- [Aidan Kurtz](#) — Development of CORE modules, website and documentation.
  - [Yunlong Tang](#) — Development of OPTIMIZE modules (PhD research).
  - [Prof. Fiona Zhao](#) — Project supervisor and head of the research lab.
- 

## CONTRIBUTORS

---

Many students/researchers have contributed to the project since its inception.

- Marc Wang** —
- Ken Nsiempba** —
- Huiyuan Yang** —

If you submit a pull request to the GitHub repository, and it is merged, your name and contribution will be listed. For more information on how to contribute, refer to the Developer Documentation.

---

# TABLE OF CONTENTS

---

0.0	Background . . . . .
0.1	System Requirements . . . . .
0.2	Installation . . . . .
<b>Section 1 — Core Framework</b>	
1.0	Overview . . . . .
<b>Section 2 — Cell Components</b>	
2.0	Preset Cell . . . . .
2.1	Custom Cell . . . . .
2.1.0	Create custom cell in any CAD software . . . . .
2.1.1	Create custom cell in Grasshopper . . . . .
2.1.2	Create custom cell in Python script . . . . .
<b>Section 3 — Frame Components</b>	
3.0	Basic Box . . . . .
3.1	Basic Cylinder . . . . .
3.2	Conform Surface-Surface . . . . .
3.3	Conform Surface-Axis . . . . .
3.4	Conform Surface-Point . . . . .
3.5	Uniform Trimmed . . . . .
<b>Section 4 — Mesh Components</b>	
4.0	Homogeneous . . . . .
4.1	Heterogeneous Gradient . . . . .
4.1	Heterogeneous Custom . . . . .
<b>Section 5 — Utility Components</b>	
5.0	View Report . . . . .
5.1	External Skin . . . . .
<b>Section 6 — Case Studies</b>	
6.1	Bone Graft . . . . .
6.2	Cellular Tire . . . . .

---

## 0.0 BACKGROUND

---

The freedom of form enabled by 3D printing has allowed engineers to integrate new orders of complexity into their designs. The goal of this research was to develop a **set of CAD tools for generating solid lattice structures** within a design space. The software would be used to:

- Reduce volume/weight while maintaining structural integrity.
- Increase surface area as a means of maximizing heat transfer.
- Generate porosity in bone scaffolds and implants.
- Serve as a platform for design optimization.

In doing so, it should always output a watertight mesh suited for 3D printing. The lack of flexibility of current software solutions was the motive for this project. We wanted to develop a flexible platform more conducive to research, which would allow us to explore and experiment with lattice design at a deeper level. The obvious first step was to decide in which environment we would develop our system. Rhinoceros is known to be very open ended, having its own engine for interpreting scripts (Python, C# and VB), and a powerful plugin SDK ( [RhinoCommon](#) ). Its Grasshopper addon is a visual programming tool widely used in architecture which provides an ideal interface for systematic design. In this visual interface, parameters and function components are combined sequentially to carry out the design of 3D models. By developing a set of custom components for Grasshopper, we could establish a modular workflow for generative lattice design.

That being said, if you are not familiar with Grasshopper, you are highly encouraged to have a look at the latest [Grasshopper Primer](#), to bring you up to speed.

---

## 0.1 SYSTEM REQUIREMENTS

---

<b>Operating System:</b>	Windows 7 or 8 (64-bit recommended)
<b>RAM:</b>	8GB or more
<b>Video Card:</b>	OpenGL 2.0 capable video card
<b>CPU:</b>	No more than 63 CPU cores

---

## 0.2 INSTALLATION

---

The following **required software** should be installed on your system.

- [Rhinoceros 5](#)
- [Grasshopper](#)

Next, if you haven't yet, download the latest version of **INTRALATTICE**

- [Intralattice](#)

To install, simply drag the 'IntraLattice.gha' file into your Grasshopper viewport. A new toolbar will appear.

## Section 1

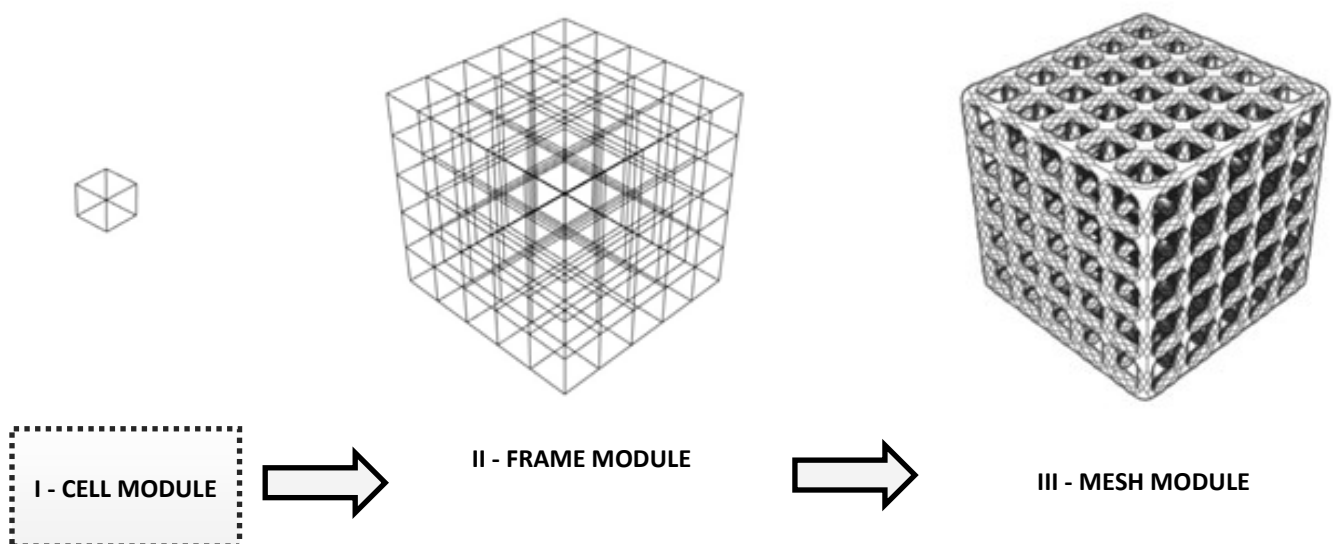
# CORE FRAMEWORK

As mentioned previously, INTRALATTICE is a set of components for Grasshopper, a visual programming tool with an intuitive interface for creating algorithms that generate 3D models. This is particularly useful for designing cellular structures; a process that is highly parametric/systematic by nature. The core algorithm is split into modules

The benefits of defining a modular workflow include:

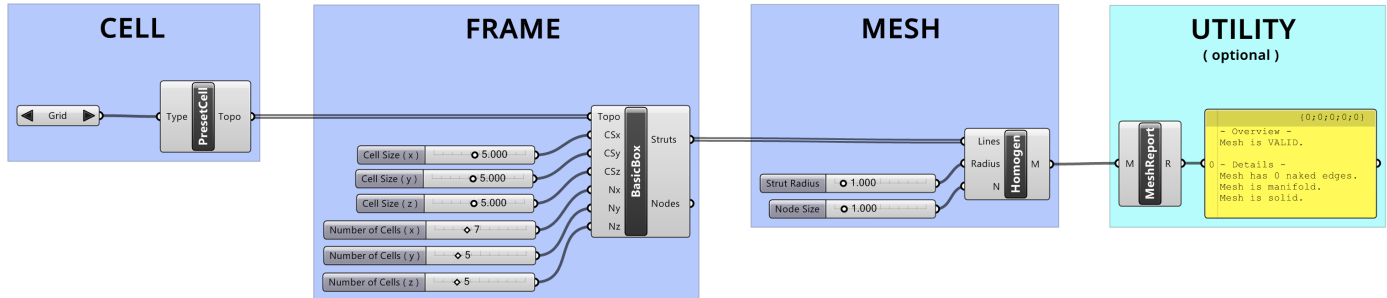
- **Flexibility** : each module can be changed independently.
- **Progressive computation** : you can preview each module before computing the next.

To illustrate the framework concept, consider the following simple example for a cubic design space.



## 1.0 CORE MODULES

The core framework is shown below.



Each module serves a specific purpose, but this purpose can be accomplished in a variety of ways. Note that the lattice structures generated are cellular structures by design, however, when pseudo-randomization is introduced, the cell boundaries dissolve (more on this later).

### CELL

Generates the unit cell, as a list of lines. The **PresetCell** component provides a selection of built-in unit cells. Users may also define custom unit cells with the **CustomCell** component. This custom design can be imported from any CAD software, or defined directly in Grasshopper.

### FRAME

Generates the lattice wireframe, as a list of lines (and a data tree of the nodes). This wireframe is based on the unit cell and a design space. The **BasicBox** and **BasicCylinder** components generate lattices within a pre-defined design space; ideal for quickly testing new cell topologies, and 3D printing samples for testing. The **ConformSS**, **ConformSA** and **ConformSP** components generate lattices that conform to a surface-based design space. The **UniformDS** component generates a trimmed uniform lattice within a mesh or brep design space.

### MESH

Generates solid mesh based on the lattice wireframe. The **Homogen** component generates a homogeneous mesh, meaning the strut radius is constant throughout the entire structure. The **HeterogenGradient** component generates a heterogeneous mesh, based on a set of predefined thickness gradients, but also lets the user define custom gradients mathematically. The **HeterogenCustom** component is used when the user wants to completely customize the thickness distribution. The underlying mesh generation methods are based on [Exoskeleton by David Stasiuk](#).

### UTILITY

(optional)

Post-processing tools. The **MeshReport** component gives a comprehensive report of a mesh. *More coming soon...*



## Section 2

# **CELL MODULE**

In our framework, the unit cell of the basis of the topology. At the moment,

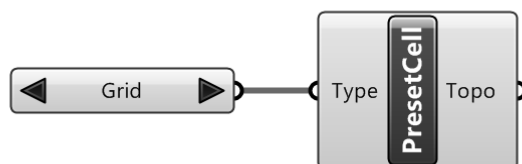


---

## 2.0 PRESET CELL

---

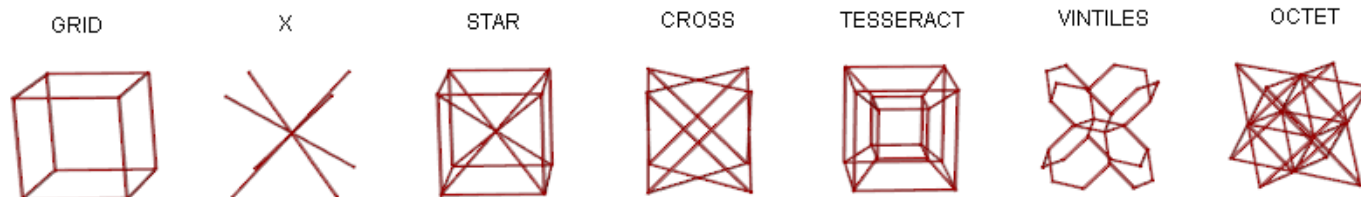
Built-in selection of unit cell topologies.



---

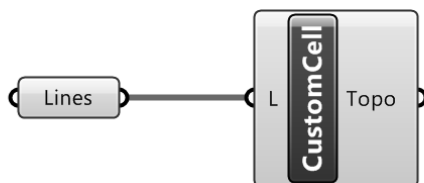
### EXAMPLE OUTPUT

---



## 2.1 CUSTOM CELL

This component can be used to pre-process a custom unit cell. It will verify that the cell is valid, and return an error if it fails any of the validity tests.



### 2.1.0 What is a valid unit cell?

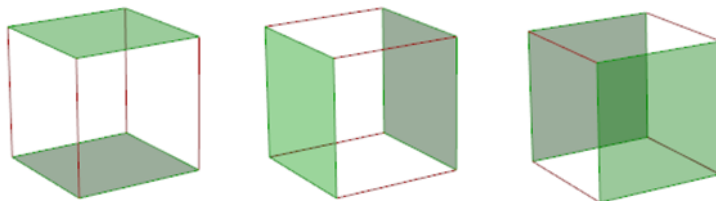
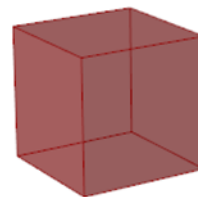
All unit cells generated by the PresetCell component are valid, and to that extent, the following information isn't useful. However, when designing custom unit cells, you need to keep in mind that the INTRALATTICE framework is built around a set of assumptions. The primordial assumption is that the cell has a cubic bounding box.

**Requirement 1** : All struts are linear

The next two requirements are concerned with continuity within the lattice.

**Requirement 2** : Each face of the bounding box of the cell has at least 1 node lying on it.

**Requirement 3** : Opposing faces of the bounding box have the same nodes



---

## 2.1.0 Import a unit cell from any CAD software

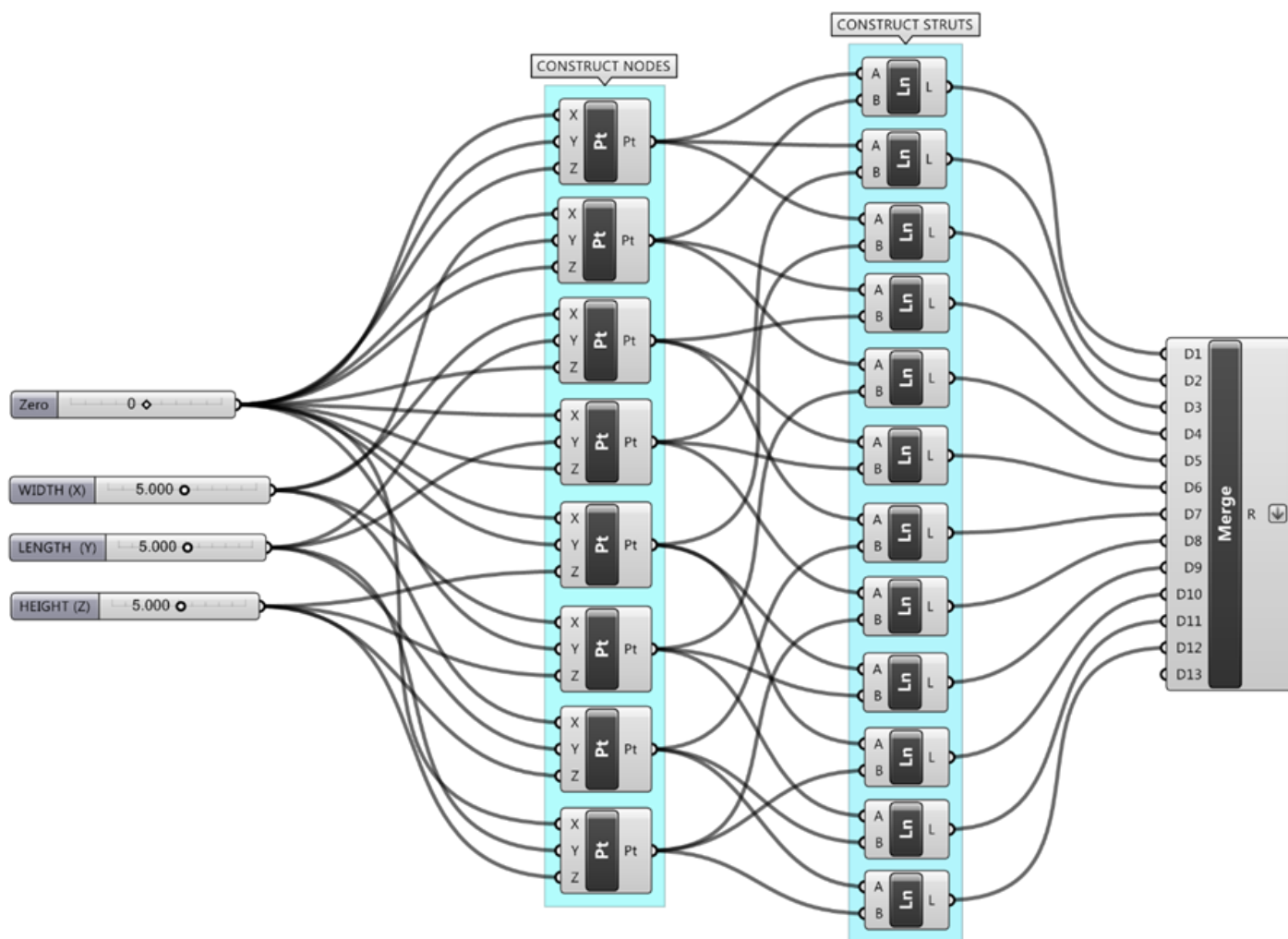
---

Once you've created a unit cell as a set of lines, save it in a universal format (ex: \_\_\_\_\_).

- 1) **Open the file** in Rhinoceros.
  - A) In Rhinoceros, go to "File -> Open".
  - B) Select your file.
  
- 2) **Set the curves** in Grasshopper.
  - A) In Rhinoceros, select all lines that make up your unit cell.
  - B) In Grasshopper, right-click on the 'L' input, then choose "Set multiple curves".
  
- 3) **Internalise the data** on the CustomCell component.
  - A) In Grasshopper, right-click on the 'L' input, then choose "Internalise data".  
The unit cell geometry is now being stored inside Grasshopper.
  
- 4) **Remove curves** from Rhinoceros.
  - A) In Rhino3D, select all the lines and click Delete.

## 2.1.1 Create custom cell in Grasshopper

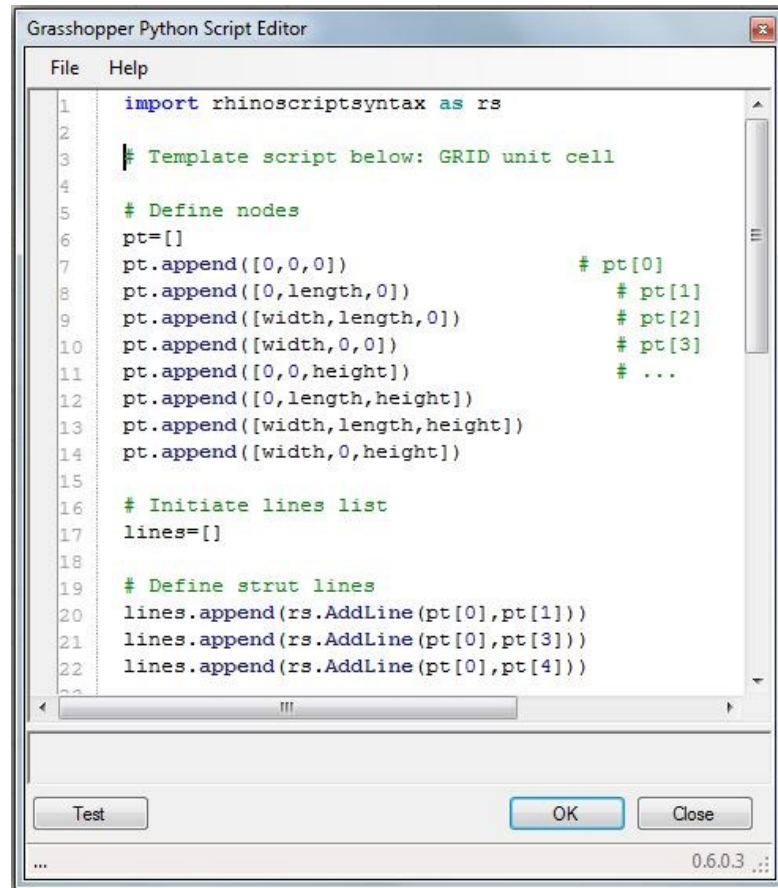
Another efficient way of designing unit cells is directly in Grasshopper. The figure below shows a simple example of how this could be done for the simple grid topology.



## 2.1.2 Create custom cell in C#/VB/Python script

Grasshopper has scripting capabilities for C# and VB.

Let's consider an example with Python, for it's relative ease-of-use.





## Section 3

# **FRAME MODULE**

The frame module generates a wireframe of the lattice structure, based on the unit cell generated by cell module, and a design space.

---

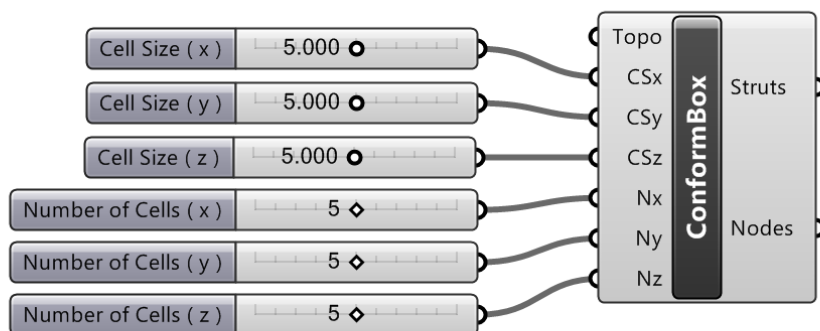
## 3.0 HOW TO DEFINE YOUR DESIGN SPACE

---

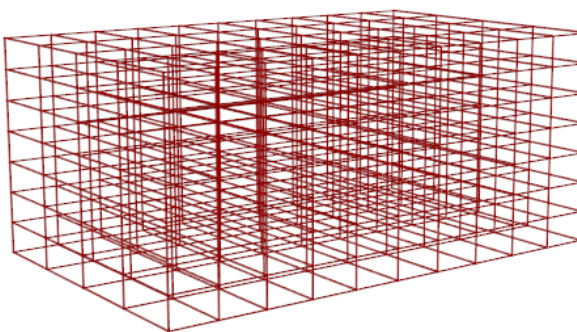
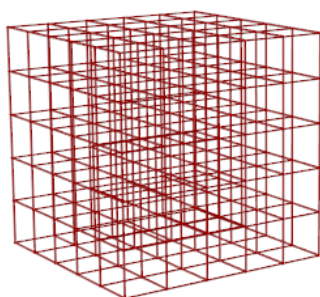
Important considerations: The frame will be thickened into a solid at the next step. When this happens, struts at the boundary will be thickened beyond the boundary. That being said, you may want to offset, or shrink your design space, to account for this. This is quite easy to do in Grasshopper, using any of the 'Offset' or 'Scale' components.

## 3.0 BASIC BOX

**Description** — Generates a lattice box.



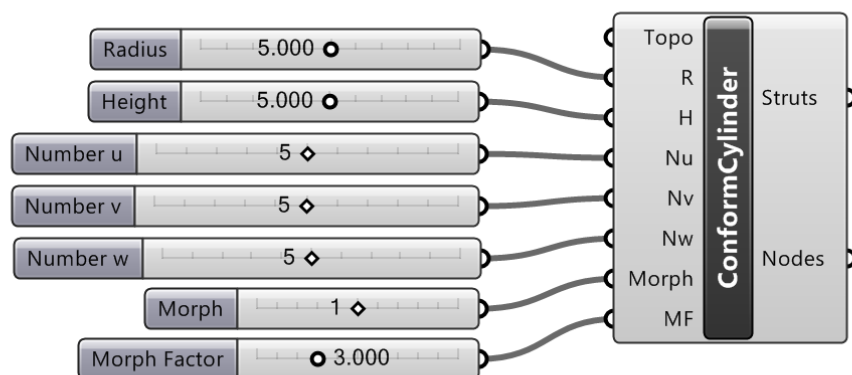
## EXAMPLE OUTPUT





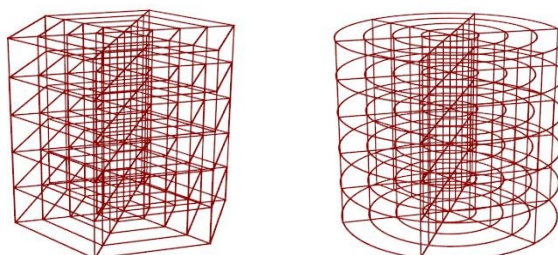
## 3.1 BASIC CYLINDER

**Description** — Generates a lattice cylinder.



### EXAMPLE OUTPUT

### GRID TOPOLOGY

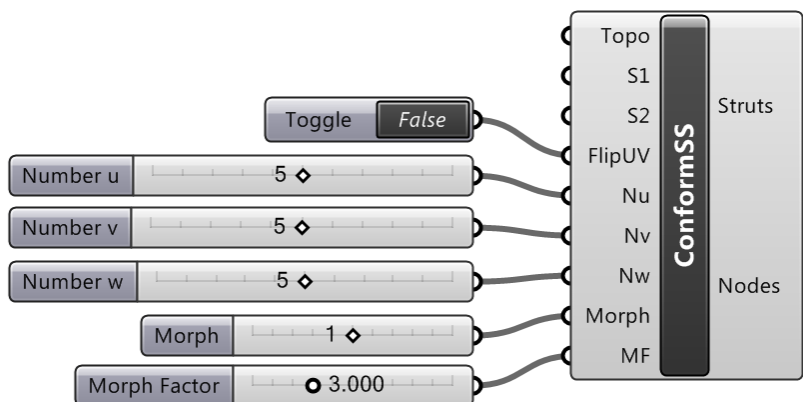


---

## 3.2 CONFORM SURFACE-SURFACE

---

**Description** — Generates a lattice cylinder.



---

### SOME EXAMPLES

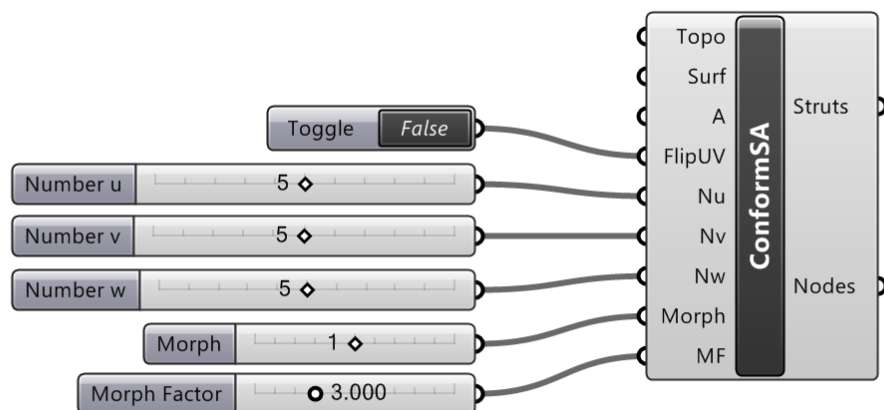
---

---

## 3.3 CONFORM SURFACE-AXIS

---

**Description** — Generates a lattice cylinder.



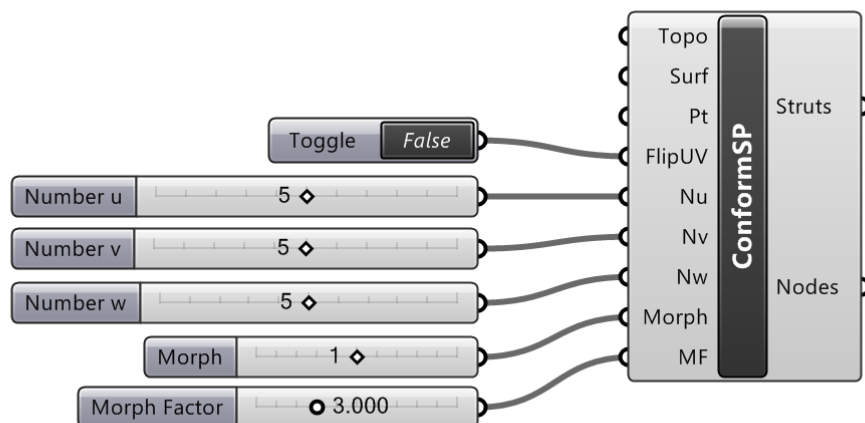
---

## SOME EXAMPLES

---

## 3.4 CONFORM SURFACE-POINT

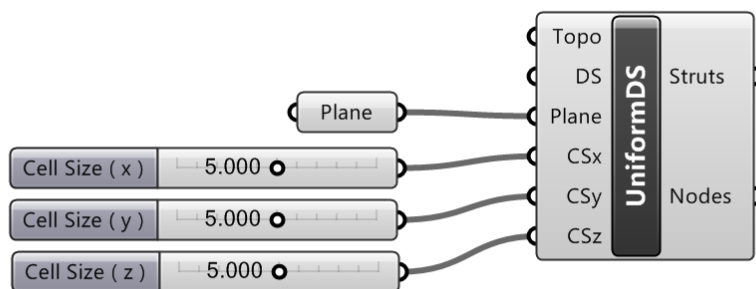
**Description** — Generates a lattice cylinder.



## SOME EXAMPLES

## 3.5 UNIFORM DESIGN SPACE

**Description** — Generates a lattice cylinder.





Section 4

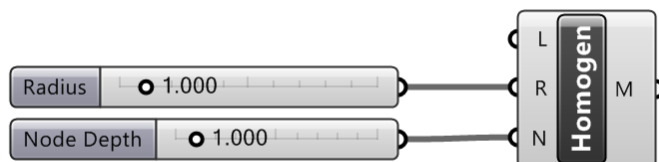
# **MESH MODULE**

---

## 4.0 HOMOGENEOUS

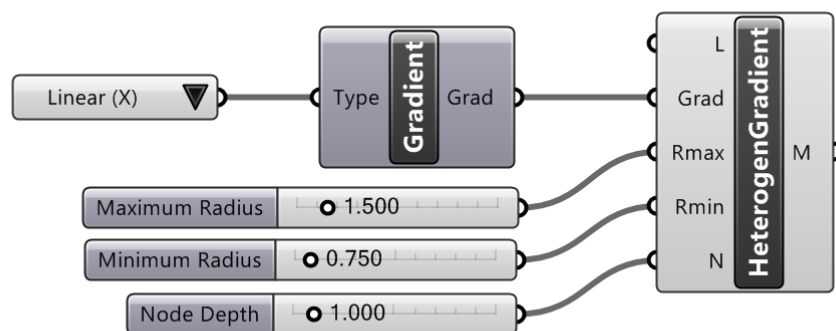
---

**Description** — Generates a lattice cylinder.



## 4.1 HETEROGENEOUS GRADIENT

**Description** — Generates a lattice cylinder.



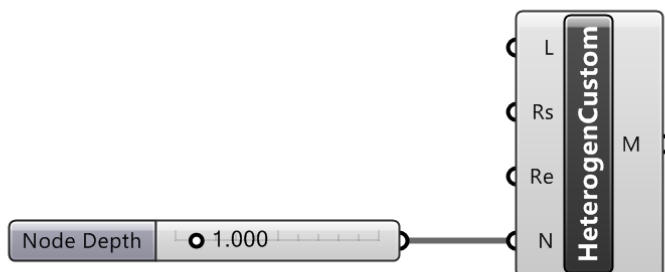


---

## 4.2 HETEROGENEOUS CUSTOM

---

**Description** — Generates a lattice cylinder.





Section 5

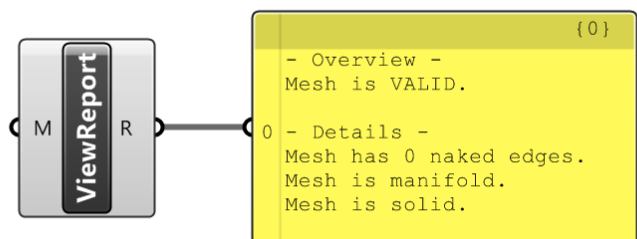
# **UTILITY MODULE**

---

## 5.0 VIEW REPORT

---

**Description** — Generates a lattice cylinder.





Section 6

# **CASE STUDIES**

## 6.0 BONE GRAFTS

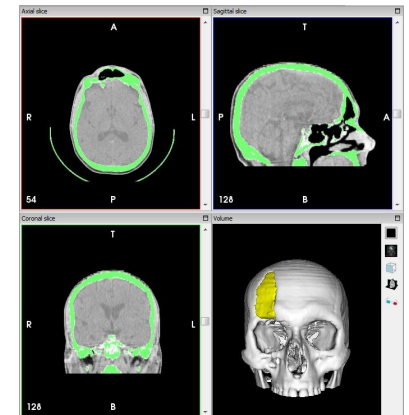
Bone tissue, unlike most other tissues, has the ability to regenerate when provided a space in which to grow. Bone grafting is a surgical procedure which aims to implant scaffold materials for reparative bone growth. The success of a bone graft relies on three biologic mechanisms: osteoconduction, osteoinduction and osteogenesis. Osteoconduction occurs when the graft serves as a scaffold upon which bone cells spread and form new bone. Naturally, the structural properties of this scaffold play an important role in this process.

Through the use of lattice structures, **INTRALATTICE** allows us to manipulate the porosity and strength of the implant at various levels. First and foremost, defining the design space requires some form of medical imaging of the patient's fracture. Below, we will walkthrough the design process of a craniofacial bone graft implant.

### I. CT Segmentation (using [InVesalius](#) software)

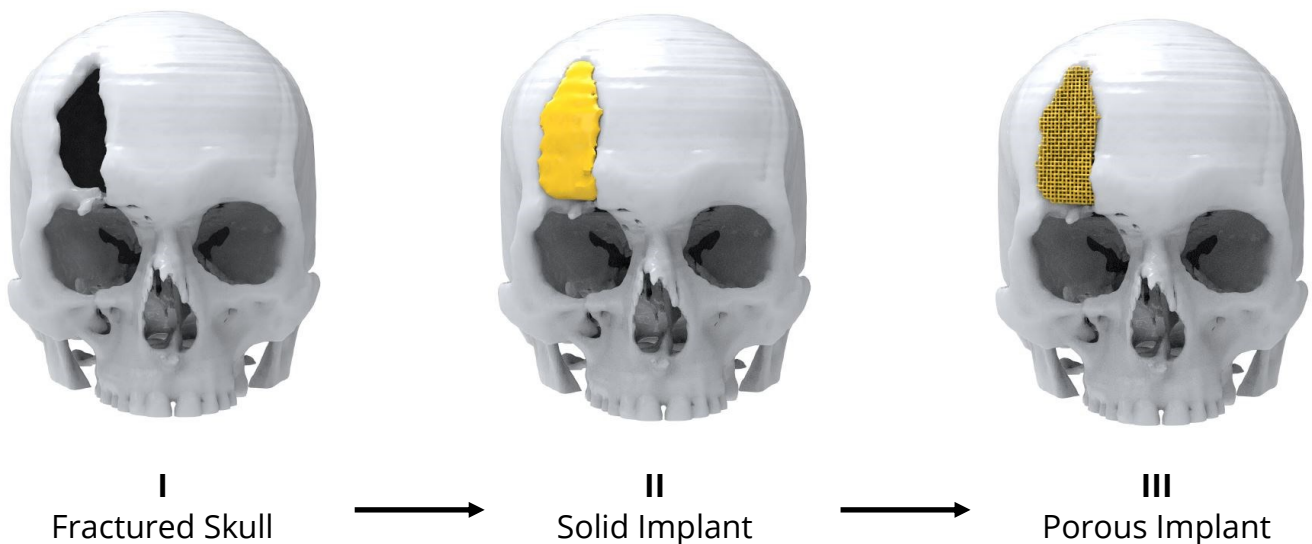
Based on a CT scan of the patients fractured skull, InVesalius reconstructed the set of medical images into a 3D representation. Then, by manually selecting pixels where cranial bone was missing, an implant space was defined (seen in yellow) and exported as a mesh (.stl).

For the record, this process was quite tedious and a more efficient approach is needed.



### II. INTRALATTICE Design Phase

The next step was to open the implant mesh in Rhino3D and select it as the design space in our Grasshopper algorithm. From here, experimenting with various lattice topologies and cell sizes allows us to manipulate the porosity and strength of the implant. In the example below, a grid-uniform-homogeneous lattice is generated:



## 6.1 CELLULAR TIRE

Airless tires are of particular interest in offroad, military and extra-terrestrial vehicles. In general, the quality of such tires is largely based on resilience. However, since this property is hard to evaluate through FEA, extensive physical testing is required to validate the quality of a design. In this case study, we simply show prototypes which would be tested, the quality of these designs is unknown.

As usual, the first step was to define a design space, which in this case, was the space between the tire thread and the wheel rim.

Next, we set the various design parameters, such as unit cell type, conformal iteration values and strut radius, and the lattice is generated.

By changing the design parameters, we easily obtain very diverse structures. In the figure shown here, you can see, in order of appearance:

1. Bare design space
2. Grid conformal lattice
3. Octet conformal lattice
4. Vintiles conformal lattice

