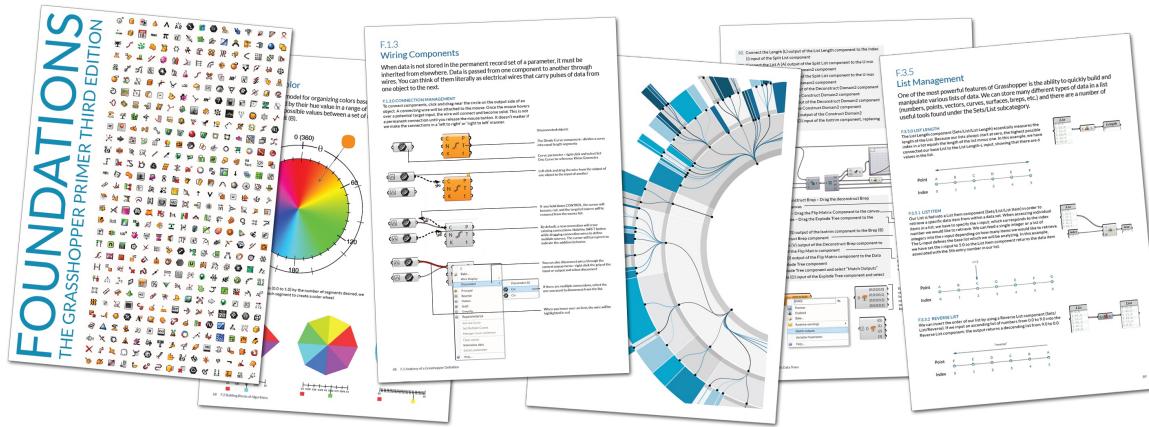


Пособие по Grasshopper (RU)

Третье издание V3.3



Grasshopper - это редактор графических алгоритмов, тесно связанный с 3-D моделирующими инструментами программы Rhino, позволяя дизайнерам создавать генераторы форм от самых простых до захватывающих дух.

ДОБРО ПОЖАЛОВАТЬ

Перед вами третье издание пособия по Grasshopper. Данное пособие было написано Эндрю О.Пэйн из Lift Architects для Rhino4 и Grasshopper версии 0.6.0007, в момент публикования оно представляло собой гигантское обновление для, и так, уже мощной платформы Grasshopper. В данный момент мы стоим перед следующим важным изменением в развитии Grasshopper и, поэтому, было подготовлено необходимое обновление существующего пособия. Мы взволнованы предстоящим добавлением данного обновленного пособия, а теперь и *интернет-пособия*, к тому невероятному вкладу, который уже внесли члены сообщества Grasshopper.

Уже имея превосходную базу, взятую за основу, наша команда в [Mode Lab](#) отправились создавать и развивать третье издание. Такая переработка имеющегося материала предоставила нам полный гид для наиболее современной версии Grasshopper 0,90076, подчеркивая, как нам кажется, обновления самых восхитительных функций. Обновленные тексты, графика и примеры работ направлены на то, чтобы обучить визуальному программированию даже самого начинающего новичка, а также провести быстрое введение в Генеративное Проектирование для опытного пользователя. Цель этого пособия в том, чтобы служить полевым гидом для новичков и давних пользователей, направленным на изучение азов использования Grasshopper в их творческой деятельности.

Это пособие расскажет вам о фундаментальных идеях и важных навыках построения работы для эффективного использования Grasshopper. Основные положения - это первая часть грядущей коллекции пособий по Grasshopper. Вот о чем мы расскажем в этом пособии:

- **Введение** - Что такое Grasshopper и как его используют?
- **Поприветствуем Grasshopper** - Создайте ваше первое определение

Table of Contents

- 0. Введение
 - 0.1. Grasshopper - Обзор
 - 0.2. Grasshopper в действии
- 1. Основные положения
 - 1.1. Поприветствуем Grasshopper!
 - 1.1.1. Установка и Запуск Grasshopper
 - 1.1.2. Grasshopper Пользовательский Интерфейс
 - 1.1.3. Взаимодействие с Rhino
 - 1.2. Структура определения Grasshopper
 - 1.2.1. Типы объектов Grasshopper
 - 1.2.2. Компоненты Grasshopper
 - 1.2.3. Типы Данных
 - 1.2.4. Компоненты Связи
 - 1.2.5. Определение Grasshopper
 - 1.3. Построение блоков алгоритмов
 - 1.3.1. Точки, плоскости и векторы
 - 1.3.2. Работая с атTRACTорами
 - 1.3.3. Математика, Выражения и Условия
 - 1.3.4. Диапазоны и Цвета
 - 1.3.5. Булевые и Логические Операторы
 - 1.4. Проектирование с использованием списков
 - 1.4.1. Геометрия кривой
 - 1.4.2. Что такое Список?
 - 1.4.3. Соединение Потока Данных
 - 1.4.4. Создание Списков
 - 1.4.5. Визуализация Списка
 - 1.4.6. Управление Списком
 - 1.4.7. Работа со списками
 - 1.5. Проектирование с использование деревьев данных
 - 1.5.1. Геометрия Поверхности
 - 1.5.2. Что такое Дерево Данных?
 - 1.5.3. Создание Дерева Данных
 - 1.5.4. Работа с Деревьями Данных
 - 1.6. Начало работы с Mesh
 - 1.6.1. Что такое Mesh?
 - 1.6.2. Понимание Топологии
 - 1.6.3. Создание Mesh
 - 1.6.4. Операции с Mesh
 - 1.6.5. Взаимодействие Mesh
 - 1.6.6. Работа с геометрией Mesh
- 2. Расширения
 - 2.1. Element*
 - 2.1.1. Введение
 - 2.1.2. Данные Half Edge
 - 2.1.3. Компоненты
 - 2.1.4. Упражнение
 - 2.1.5. Изучение применения в архитектуре

3. [Приложение](#)

- 3.1. [Индекс](#)
- 3.2. [Файлы примеров](#)
- 3.3. [Ресурсы](#)
- 3.4. [О пособии](#)

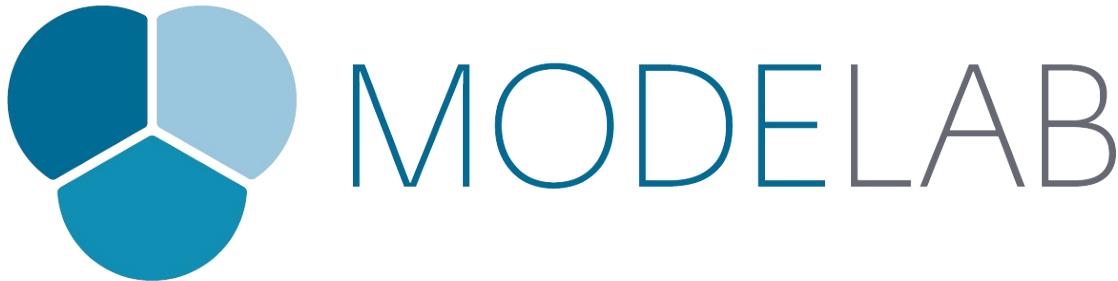
- **Структура определения Grasshopper** - Из чего состоит определение?
- **Построение блоков алгоритмов** - Начните с простого и создайте сложное
- **Проектирование с использованием списков** - Что такое список и как с ним справится?
- **Проектирование с использование деревьев данных** - Что такое структура данных и какое они имеют значение для моего проекта?
- **Приложение** - Ссылки и Рабочие файлы для продолжения изучения

Мы надеемся, что, в конце концов, это пособие вдохновит вас на изучение многочисленных возможностей программирования с Grasshopper. Мы желаем вам удачи в начале вашего путешествия.

ПРОЕКТ ПОСОБИЕ ПО GRASSHOPPER

Пособие по Grasshopper - это открытый проект, основателем которого является Боб МакНил, Скотт Дэвидсон и команда Grasshopper Development в [Robert McNeel & Associates](#).

Mode Lab стали авторами Третьего Издания пособия. <http://modelab.is>



Если вы хотите принять участие в этом проекте, зайдите на наш проект на GitHub, разде Wiki, что узнать с чего начать (<https://github.com/modelab/grasshopper-primer/wiki>).

БЛАГОДАРНОСТЬ

Особую благодарность мы выражаем Дэвиду Руттену за несокончаемое вдохновение и бесценную работу первопроходца в Grasshopper. Мы бы также хотели поблагодарить Эндрю О.Пэйна за предоставление ресурсов, с которых началась эта работа. Ну, и наконец, огромная благодарность Бобу МакНилу и всем в [Robert McNeel & Associates](#) за их щедрую поддержку все эти годы. Также благодарим Наталью Медведеву и Владимира Воронича за перевод пособия на русский язык.

НЕОБХОДИМОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Rhino5

Rhino 5.0 является лидером на рынке моделирующего ПО для промышленного проектирования. Крайне сложные формы могут быть смоделированы или получены прямо посредством 3D преобразователей. Имея мощный движок на основе NURBS (неоднородный рациональный B-сплайн), Rhino 5.0 способен создавать, редактировать, анализировать и переводить кривые, поверхности и твердые тела. Ограничений по сложности, степени или размеру просто нет.

<http://www.rhino3d.com/download/rhino/5/latest>

Grasshopper

Для проектировщиков, исследующих новые формы используя генеративные алгоритмы, Grasshopper - это редактор графических алгоритмов, тесно связанный с 3D моделирующими инструментами Rhino. В отличие от RhinoScript или Python, Grasshopper не требует знания абстрактного синтаксиса программирования, но при этом позволяет проектировщикам создавать генераторы формы от самых простых до захватывающих дух.

<http://www.grasshopper3d.com/page/download-1>

ФОРУМ

Форум Grasshopper - очень активный и предлагает прекрасный ресурс для размещения вопросов/ответов и нахождения помощи по всему, чего угодно. Форум делится на следующие категории: обсуждение общих вопросов, ошибки, примеры, популярные вопросы.

<http://www.grasshopper3d.com/forum>

Раздел общих вопросов по Grasshopper содержит ответы на многие вопросы, которые у вас могут возникнуть, а также полезные ссылки:

<http://www.grasshopper3d.com/notes/index/allNotes>

Касательно общих вопросов относительно Rhino3D, сначала проверьте на Форуме МакНила, Discourse.

<http://discourse.mcneel.com/>

ИНФОРМАЦИЯ ПО ЛИЦЕНЗИИ

Пособие по Grasshopper - лицензировано в Creative Commons Attribution - NonCommercial-ShareAlike 3.0 Unported license. Полный текст этой лицензии доступен по этому адресу:

<http://creativecommons.org/licenses/by-nc-sa/3.0/us/legalcode>

Grasshopper - Обзор

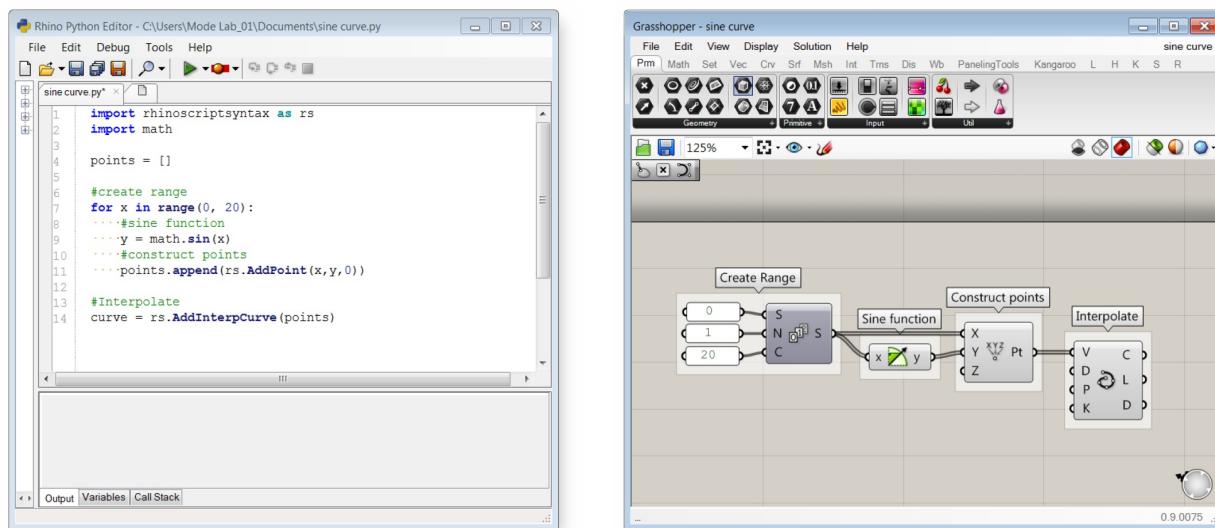
Grasshopper - это редактор визуального программирования, разработанный Дэвидом Руттеном в Robert McNeel & Associates. Будучи плагином для Rhino3D, Grasshopper интегрирован с мощной и многофункциональной моделирующей средой, которая используется творческими профессионалами в широком диапазоне областей, включая архитектуру, проектирование, дизайн продукта и др. В тандеме, Grasshopper и Rhino предлагают нам возможность установить точный параметрический контроль за моделями, способность исследовать генеративный процесс проектирования, а также платформу для развития логики программирования высокого уровня - и все это при помощи интуитивно-понятного графического интерфейса.

Grasshopper берет свое начало в функционале кнопки "Record History" в Rhino3D версия 4. Данная встроенная функция давала пользователям возможность сохранить процесс моделирования в истории по мере продвижения работы. Если вы применяли loft для 4-х кривых с включенной записью истории, а затем меняли контрольные точки одной из кривых, то поверхность геометрии обновлялась. В далеком 2008, Дэвид задался вопросом: "Что если бы у вас был более выраженный контроль за этой историей?" и тогда был создан прототип Grasshopper, Explicit History. Это дало доступ к дереву истории для подробного редактирования и позволило пользователю развивать логические последовательности помимо существующих встроенных возможностей Rhino3D.

Спустя шесть лет, Grasshopper стал мощным редактором визуального программирования, возможности которого могут быть расширены наборами дополнений, разработанных отдельно. Более того, он в корне изменил процесс работы специалистов во многих индустриях и взрастил активное глобальное сообщество пользователей.

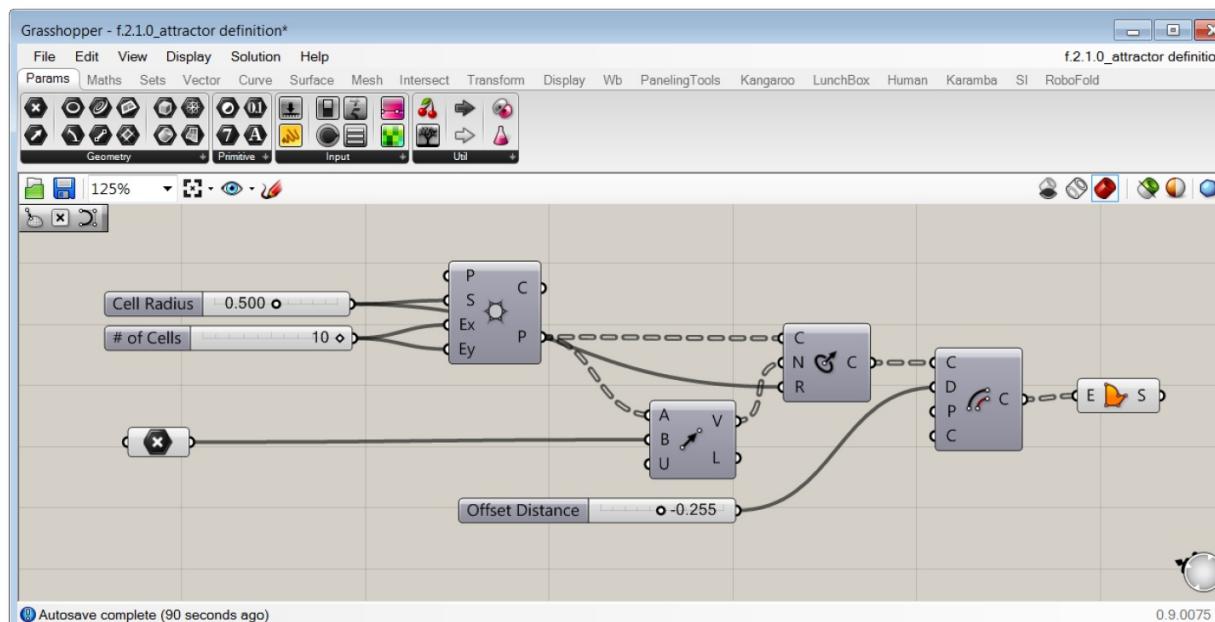
Это пособие фокусируется на Основных положениях, предлагая ключевые знания, необходимые для погружения в регулярное использование Grasshopper и нескольких ответвлений, но также достаточных для того, чтобы вы могли углублять свою практику. Перед погружением в описания, диаграммы и примеры, расположенные ниже, давайте обсудим, что есть визуальное программирование, основы интерфейса и терминологии Grasshopper, а также о "live" характеристиках - обратной связи видовых окон и взаимодействии пользователя с интерфейсом.

Визуальное программирование это парадигма компьютерного программирования внутри которой пользователь управляет логическими элементами графически, а не текстом. Некоторые из самых известных текстовых языков программирования, таких как C#, Visual Basic, Processing - и ближайшее к Rhino - Python и Rhinoscript, требуют написание кода с использованием специального языкового синтаксиса. Визуальное программирование, напротив, позволяет нам соединить функциональные блоки в последовательность действий, где единственное требование "синтаксиса" - чтобы входы блоков получали подходящие по типу данные, и в идеале, организованную в соответствии с требуемым результатом - см. разделы Соединение Потоков Данных и Проектирование с использованием Деревьев Данных. Эта черта визуального программирования позволяет избежать барьера, типично встречающегося при попытке выучить новый язык, даже разговорный, а также выдвигает на передний план интерфейс, который размещает Grasshopper в более знакомой обстановке для проектировщиков.



Это фото показывает процесс черчения синусоидальной кривой в Python и в Grasshopper.

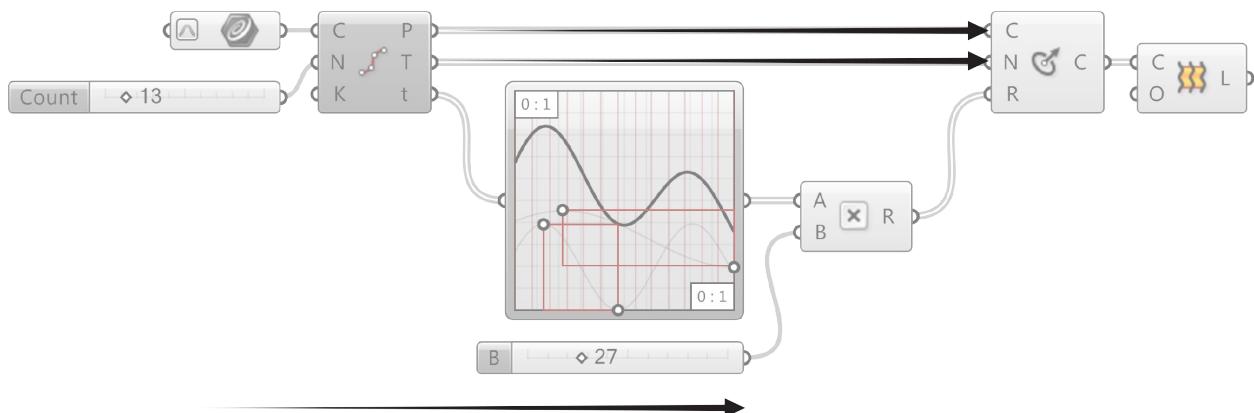
Чтобы начать работать в Grasshopper, необходимо скачать и установить программу с сайта Grasshopper3D.com. Когда программа установлена, вы можете открыть плагин, напечатав "Grasshopper" в командной строке Rhino. Когда мы сделаем это в первый раз при запуске новой сессии Rhino, нам покажут загрузочные подсказки Grasshopper, сопровождаемые окном редактирования Grasshopper. Теперь мы можем добавлять функциональные блоки, которые называются "компонентами" на "холсте", соединять их "связями" и сохранять целое "определение" в формате .ghx.



Определение Grasshopper, состоящее из компонентов, соединенных связями на холсте

Как только мы начали развивать определение Grasshopper и создали объекты, называемые "слайдер", мы можем контролировать геометрию, можем легко постигать интуитивно связи, которые мы создали между входом этого объекта и тем, что мы видим в видовом окне Rhino. Эта связь, по существу, "живая". Если мы изменим положение ползунка слайдера, мы увидим последствия этого действия, так как вводные параметры в каком-то месте нашего определения изменились и поэтому программа должна пересчитать решение и отобразить обновление. К счастью для нас, в начале использования Grasshopper, предпросмотр геометрии, которую мы видим, это упрощенное представление решения и оно обновится автоматически.

Важно сейчас принять к сведению эту связь, потому что когда ваши определения станут более сложными, правильное управление потоком данных, статусом инструмента "проверки" и отображением в окне просмотра Rhino предотвратит многие нежелательные проблемы.



Программа работает слева направо

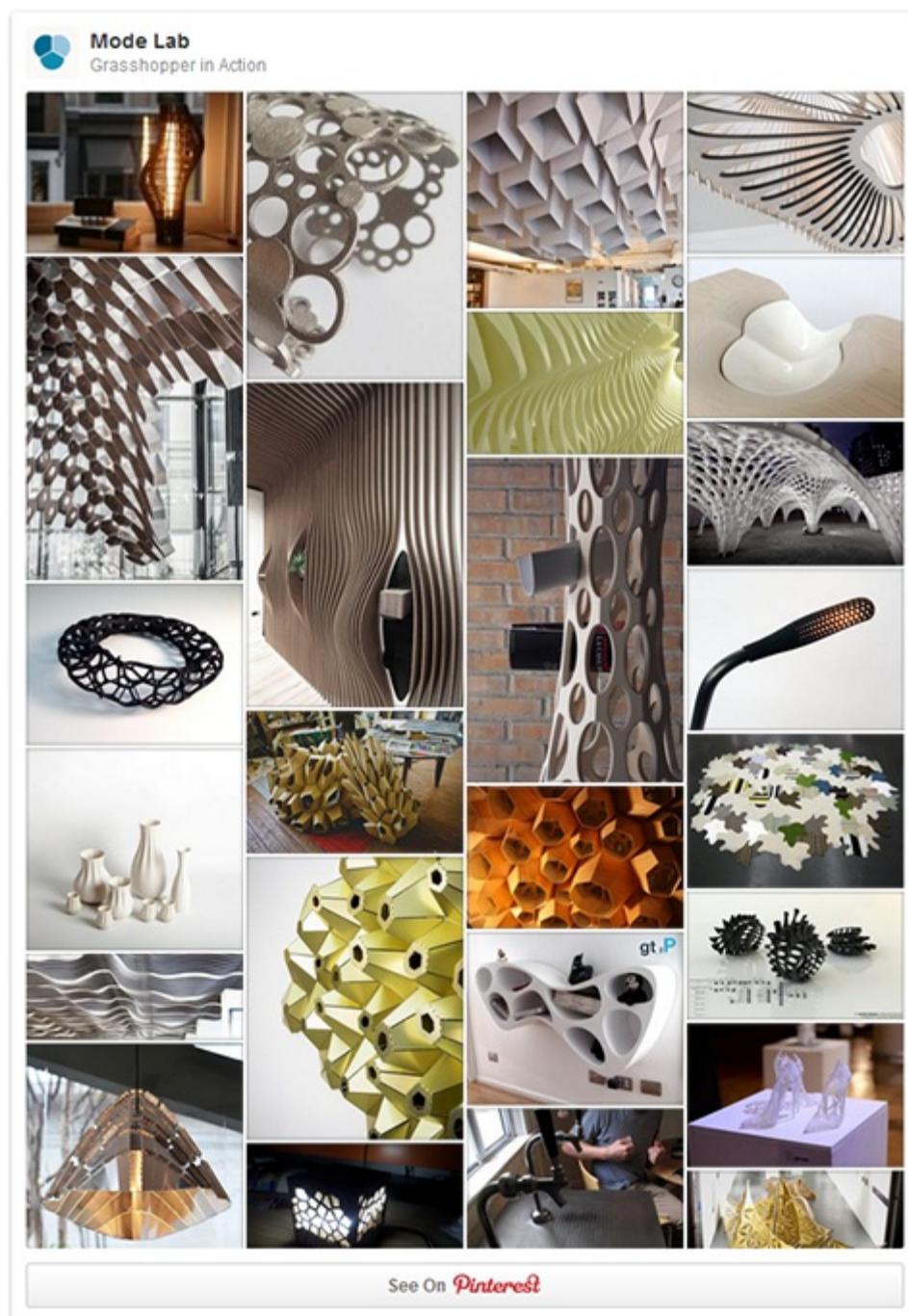
НЕОБХОДИМО ЗАПОМНИТЬ

- Grasshopper - это редактор графических алгоритмов, интегрированный с 3D моделирующими инструментами Rhino.
 - Алгоритмы - это пошаговые процедуры, спроектированные для выполнения операций.
 - Вы используете Grasshopper, чтобы спроектировать алгоритмы, которые затем автоматизируют задачи в Rhino3D.
 - Быстрый способ начать, если вы не знаете, как выполнять определенные операции в Grasshopper, это попытаться вручную и постепенно создать алгоритм, используя команды Rhino.

Как только вы впервые начали исследовать Grasshopper или развивать ваши навыки, это означает, что вы присоединились к глобальному сообществу Grasshopper, состоящему из активных людей из различных областей и с разным опытом. Форум на сайте Grasshopper3D.com - это полезный ресурс для публикования вопросов, чтобы поделиться находками и получить знания. Это сообщество, которым мы дорожим, когда писали это пособие и наблюдали за развитием Grasshopper все эти годы. Добро пожаловать!

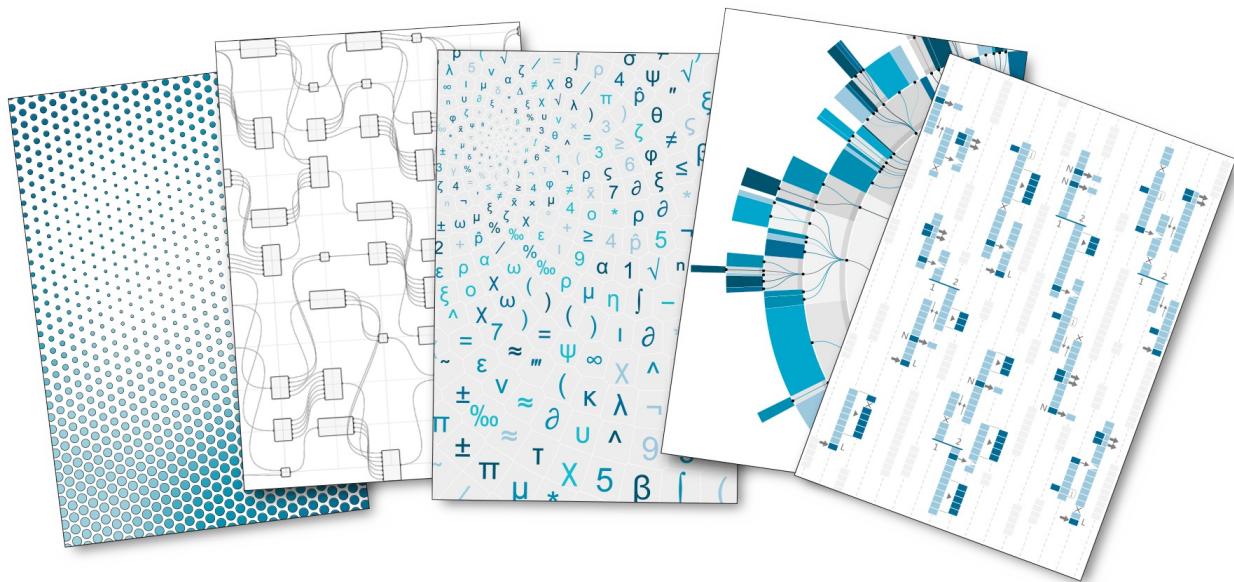
Grasshopper в Деле

Присоединяйтесь в Pinterest к доске Grasshopper in Action.



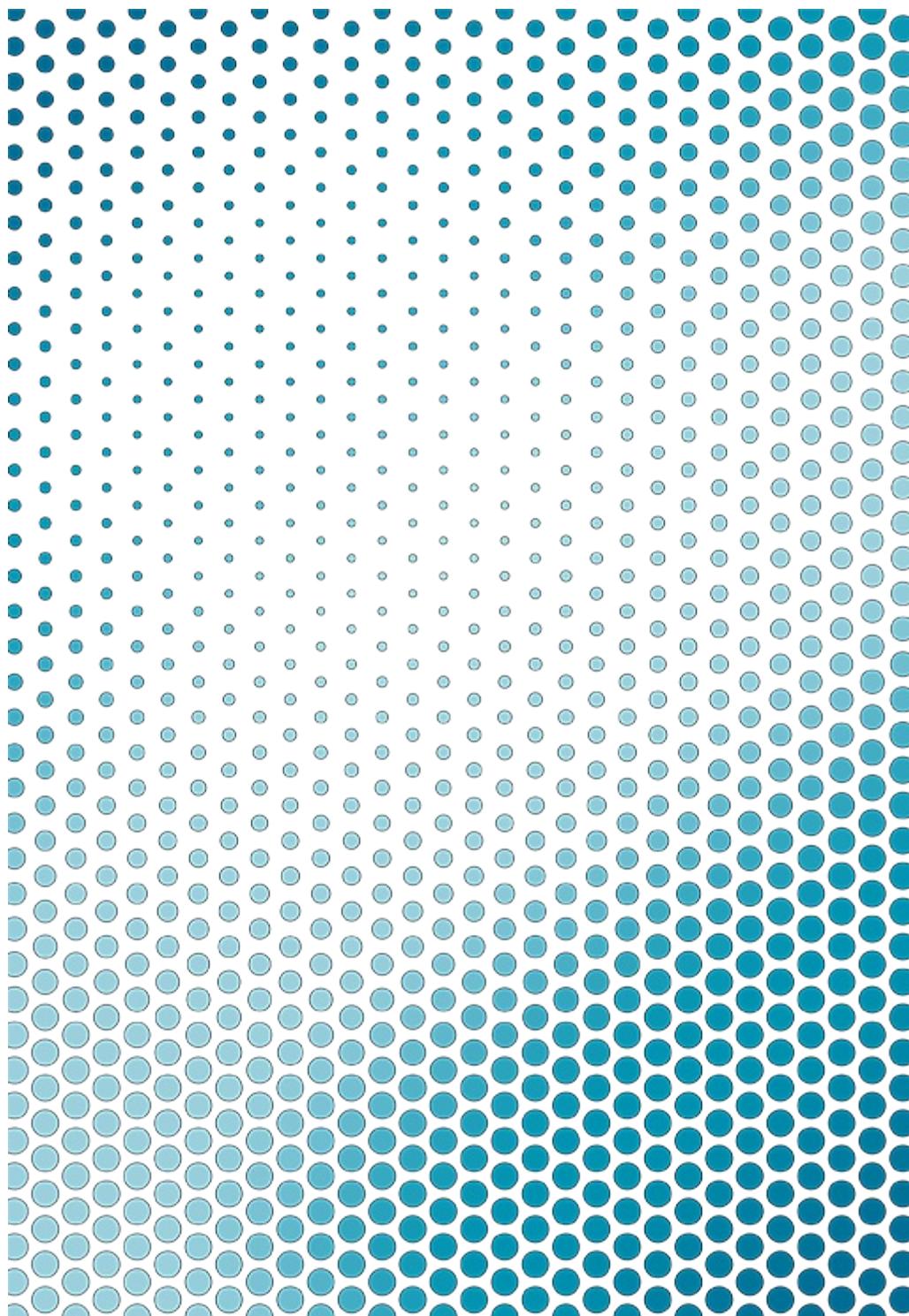
1. ОСНОВНЫЕ ПОЛОЖЕНИЯ

Прочный фундамент долго прстоит. Эта часть Пособия дает представление о ключевых концептах и чертах параметрического моделирования в Grasshopper.



1.1. ПОПРИВЕТСТВУЕМ GRASSHOPPER

Grasshopper - это редактор графических алгоритмов, интегрированный с 3D моделирующими инструментами Rhino. Grasshopper используется, чтобы спроектировать алгоритмы, которые затем автоматизируют задачи в Rhino3D.



1.1.1. УСТАНОВКА И ЗАПУСК GRASSHOPPER

Плагин Grasshopper часто обновляется, убедитесь, что у вас последняя версия.

Обратите внимание, что на данный момент версии Grasshopper для Mac не существует.

1.1.1.1. СКАЧИВАНИЕ

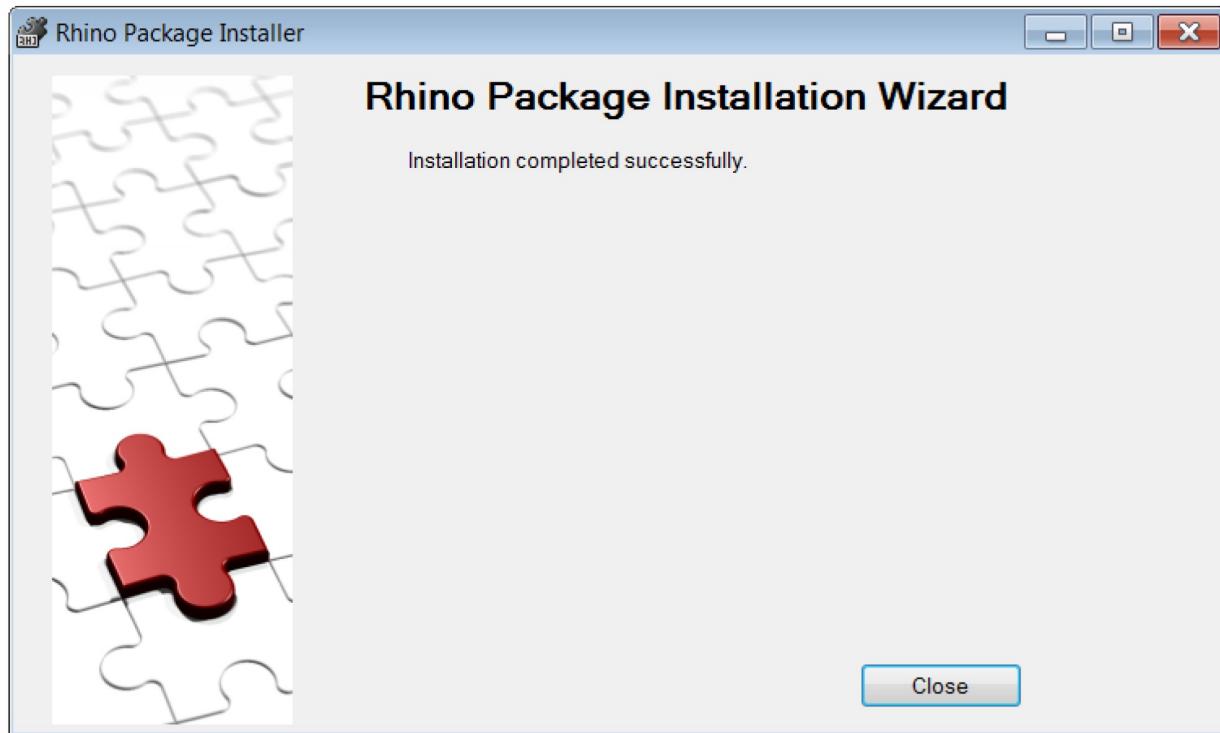
Чтобы скачать плагин Grasshopper, зайдите на сайт Grasshopper. Нажмите на вкладку Download вверху страницы, когда появится подсказка на экране, введите ваш email адрес. Затем, кликните правой клавишей по ссылке скачивания и выберите Save Target as из меню. Выберите расположение на вашем жестком диске (внимание файл не может быть загружен через сетевое подключение, поэтому файл должен быть сохранен на жесткий диск вашего компьютера) и сохраните исполняемый файл по этому адресу.

The screenshot shows the official Grasshopper website. At the top, there's a green header with the title 'Grasshopper' and the subtitle 'ALGORITHMIC MODELING FOR RHINO'. Below the header, there are several navigation links: Home, Galleries, Download (which is highlighted in red), Tutorials, Discussions, Events, and My Page. To the right of the navigation, there's a stylized illustration of a grasshopper. On the left side of the main content area, there's a large heading 'Download'. Under this heading, there are two sections: 'Latest Grasshopper for Rhino 5.0 (Windows only)' and 'Old Grasshopper for Rhino 4.0 (Windows only)'. Both sections contain a brief description and a 'Download...' link. Below these sections, it states 'There is no version of Grasshopper which works on Mac OS.' and provides sharing options for social media. On the right side of the page, there's a sidebar with various links: 'Welcome to Grasshopper', 'Sign Up' or 'Sign In', 'Translate' (with a dropdown menu for language selection), 'Search Grasshopper' (with a Google Custom Search bar), and 'Photos' (with a small thumbnail image).

Скачайте Grasshopper с сайта grasshopper3d.com.

1.1.1.2. УСТАНОВКА

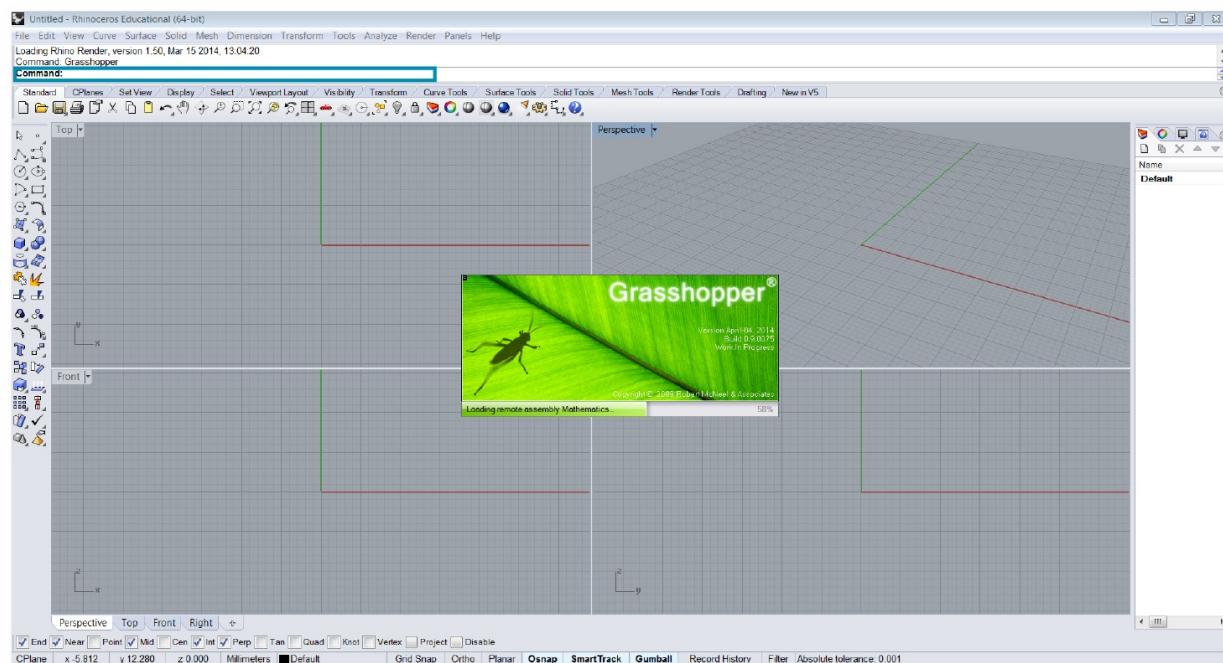
Выберите Run из диалогового окна установки и следуйте инструкциям по установке. (примечание: чтобы плагин установился корректно, у вас уже должен быть установлен Rhino 5).



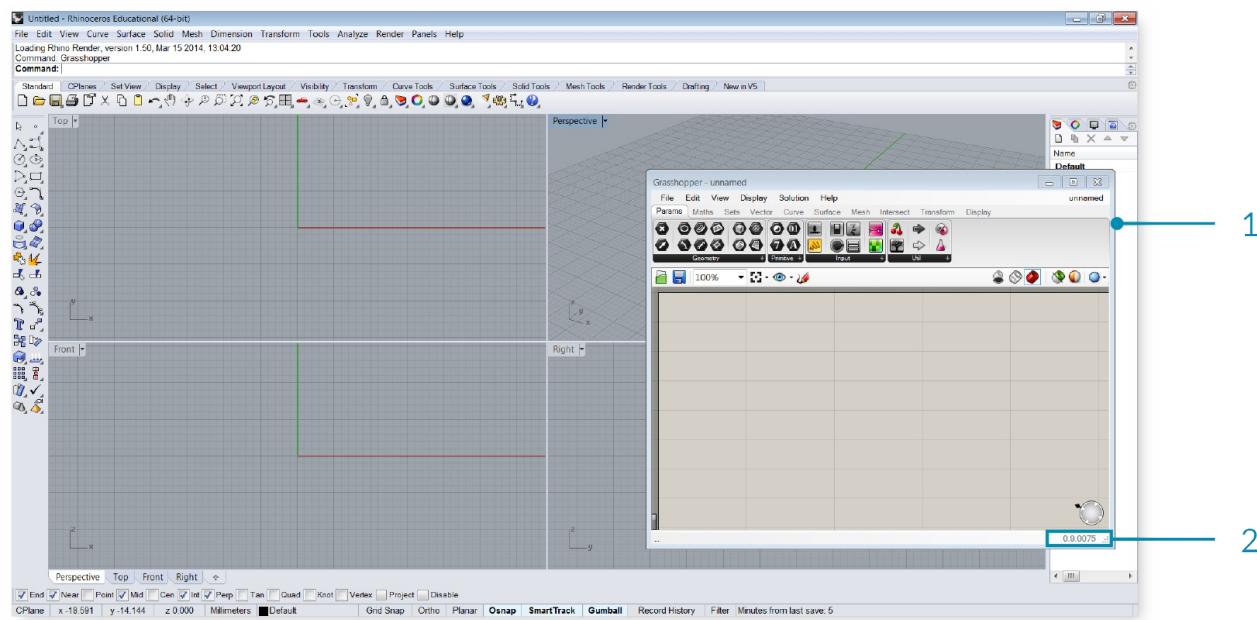
Следуйте пошагово за Мастером Установки.

1.1.1.3. ЗАПУСК

Чтобы запустить Grasshopper, наберите Grasshopper в командной строке Rhino. При запуске Grasshopper, первое, что вы увидите, это новое окно, всплывшее перед окном Rhino. Внутри этого окна вы можете создавать программы, основанные на нодах, для автоматизации различных типов функциональности в Rhino. Лучше всего, расположить окна так, чтобы они не накладывались друг на друга, чтобы окно Grasshopper не заслоняло видовое окно Rhino.

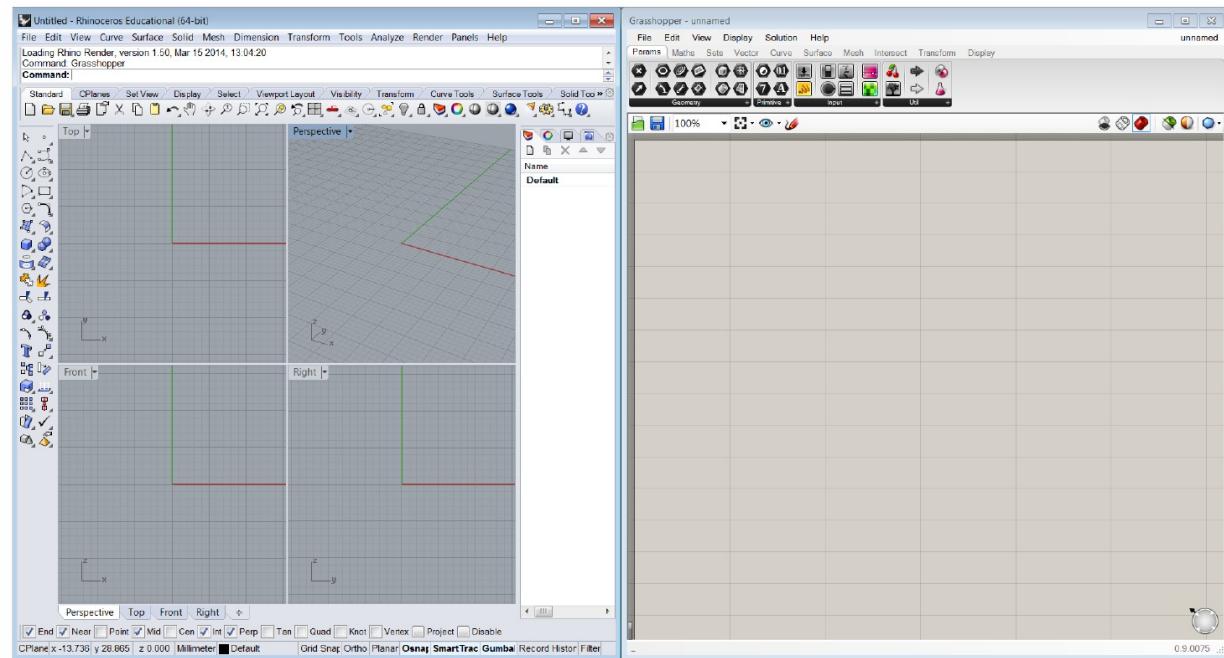


Наберите Grasshopper в командной строке Rhino для запуска плагина Grasshopper.



1. Окно Grasshopper всплывает перед видовым окном Rhino.

2. Grasshopper отображает номер версии внизу окна.



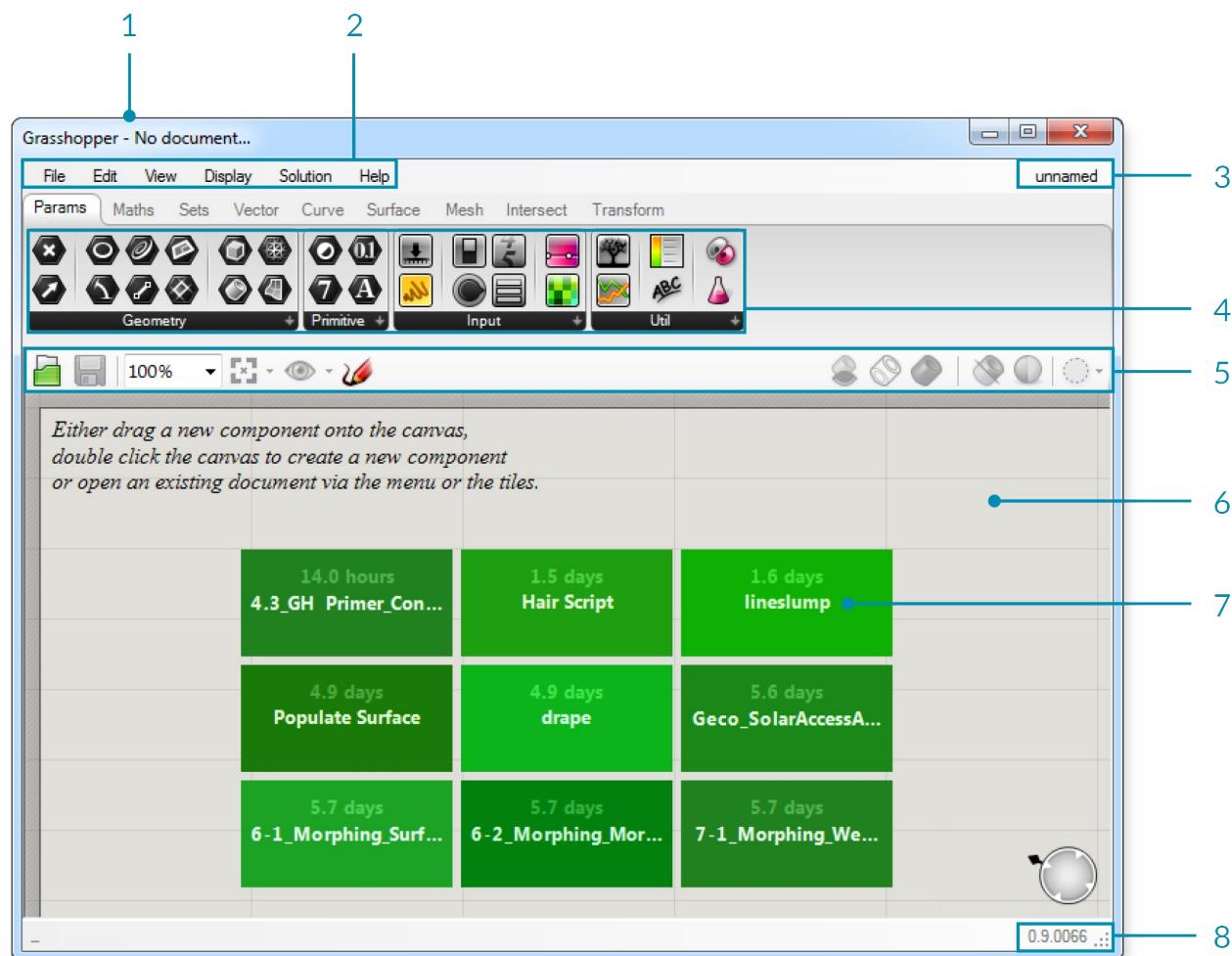
Разделите экран так, чтобы Grasshopper не перекрывал окно просмотра Rhino. Вы можете сделать это просто перетянув каждое окно в противоположные стороны экрана или зажав кнопку Windows и нажимая стрелки вправо-влево.

1.1.2. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС GRASSHOPPER

Визуальный стиль Grasshopper "plug-and-play" дает дизайнерам возможность сочетать творческое решение вопросов с инновационными системами правил посредством использования потокового графического интерфейса.

Давайте начнем знакомство с Grasshopper с пользовательского интерфейса UI. Grasshopper - это приложение визуального программирования, где вы можете создавать программы, называемые definition (определение) или documents (документы), путем расположения компонентов на главном окне редактирования (canvas - холст). Выходы этих компонентов соединяются с входами последующих компонентов - создавая график информации, который можно прочитать слева направо. Давайте начнем с этих основ.

Предположим, вы уже установили плагин Grasshopper (см. F.0.0), наберите слово "Grasshopper" в командной строке Rhino, чтобы показать редактор Grasshopper. Интерфейс Grasshopper содержит некоторое кол-во элементов, большинство которых будут хорошо знакомы пользователям Rhino. Давайте посмотрим на несколько функций интерфейса.



1. Строка заголовка окна.
2. Стока главного меню.
3. Контроль диспетчера файлов.
4. Панели компонентов.
5. Панель инструментов холста.
6. Холст.

7. Эта область, отмеченная сеткой из прямоугольных ячеек, открывает интерфейс, с помощью которого можно открыть недавно вызванные файлы. Меню 3x3 показывает файлы, которые были недавно изменены (в хронологическом порядке). Красным цветом будет отмечен файл, который не удается найти (такое может случиться если вы перемещали файл в новую папку или удалили его).
8. Стока состояния сообщает вам о том, какая версия Grasshopper установлена в текущий момент на вашем компьютере. При наличии более новой версии, выпадает меню с инструкциями, как скачать последнюю версию.

1.1.2.1. СТРОКА ЗАГОЛОВКА ОКНА

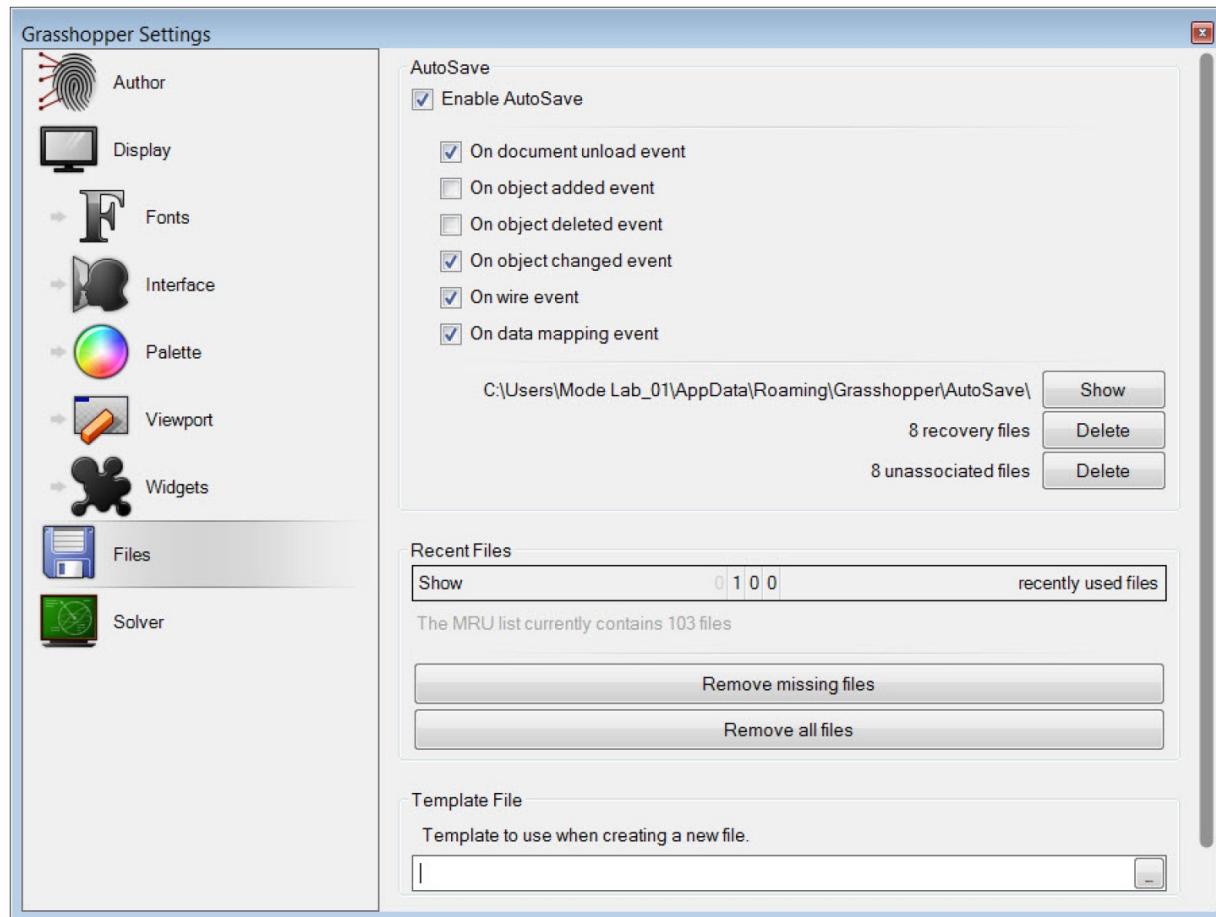
Редактор строки заголовка окна ведет себя по-разному в отличие от большинства других диалоговых окон в Microsoft Windows. Если окно находится не в уменьшенном или увеличенном размере, двойной клик по строке заголовка свернет окно в уменьшенный размер на вашем экране. Это хороший способ переключаться между плагином Grasshopper и Rhino, т.к. это окно редактора сворачивается без необходимости перемещать его к краю экрана или размещения позади других окон. Помните, что если вы закроете окно редактора, предпросмотр геометрии Grasshopper в видовом окне Rhino исчезнет, но сам файл не будет закрыт. В следующий раз когда вы запустите команду "Grasshopper" в диалоговом окне Rhino, окно вернется в том же виде с теми же загруженными файлами. Так происходит потому, что как только оно загружено по запросу команды, ваша сессия в Grasshopper остается активной до тех пор, пока Rhino не закрыт.

1.1.2.2. СТРОКА ГЛАВНОГО МЕНЮ

Эта строка похожа на обычное меню Windows, за исключением контроля диспетчера файла справа (см. следующий раздел). Меню файла предоставляет стандартные функции (например, New File (новый документ), Open (открыть), Save (сохранить) и т.д.) в добавок к нескольким инструментам, которые позволяют вам экспорттировать изображения вашего текущего документа Grasshopper (см. Export Quick Image - быстрый экспорт изображения и Export Hi-Res Image - экспорт изображения высокого разрешения). Вы можете контролировать различные аспекты пользовательского интерфейса, используя меню View (вид) и Display (дисплей/отображение). Меню Solution (решение) позволяет управлять различными параметрами того, как инструмент проверки вычисляет схемы решений.

Следует упомянуть, что многие настройки приложения можно контролировать через диалоговое окно Preferences (Настройки), располагающееся под меню File. Секция Author (Автор) позволяет настраивать персональные метаданные, которые будут сохраняться с каждым документом Grasshopper, в то время как секция Display (дисплей/отображение) предоставляет точный контроль за видом и функцией интерфейса. Секция File (файл) позволяет уточнить такие моменты, как часто и где сохранять автоматически сохраняемые файлы (в случае если приложение было непреднамеренно закрыто или дало сбой). Ну и наконец, секция Solver (инструмент проверки) позволяет управлять основными и сторонними плагинами, которые могут увеличить функциональность.

Примечание: Будьте внимательны при использовании "горячих" клавиш, так как они осуществляются активным окном, а это может быть как Rhino так холст Grasshopper или любое другое окно внутри Rhino. Довольно легко использовать "горячие" клавиши, а потом осознать, что у вас выбрано не то окно и случайно вы вызвали не ту команду.



Окно Preferences позволяет настроить большинство из настроек приложения Grasshopper.

1.1.2.3. УПРАВЛЕНИЕ ДИСПЕТЧЕРОМ ФАЙЛОВ

Диспетчер файлов позволяет быстро переключаться между различными загруженными файлами, выбирая их через выпадающий список. Получив доступ к вашим открытым файлам через выпадающий список диспетчера файлов, вы сможете быстро копировать и вставлять элементы из открытых определений. Просто кликните на имени активного файла в окне управления диспетчера, и каскадный список отобразит все открытые файлы для быстрого доступа к ним (рядом будет миниатюрное изображение каждого открытого определения). Можно также нажать комбинацию клавиш Alt+Tab, чтобы быстро переключиться между любыми открытыми документами Grasshopper.

Конечно, вы можете пойти через стандартный диалог Open File для загрузки любого документа Grasshopper, хотя вы можете также перетащить любой файл Grasshopper на холсте, чтобы загрузить определенное определение.

Grasshopper - это плагин, который работает "поверх" Rhino и поэтому имеет свои типы файлов. Тип файла по умолчанию - файл с бинарными данными, сохраняется с расширением .gh. Другой тип файлов известен как Grasshopper XML файл, имеет расширение .ghx. Тип файлов XML (расширяемый язык разметки) использует тэги для определения объектов и характеристики объектов (почти как .HTML документ), но использует настраиваемые тэги для определения объектов и данных внутри каждого объекта. Из-за того, что XML файлы отформатированы как текстовые документы, вы можете открыть любой файл Grasshopper XML в текстовом редакторе, например NotePad, чтобы посмотреть на код, который стоит за всем этим.

Grasshopper может открыть файл несколькими способами, поэтому вам нужно будет указать, какой способ вы хотите использовать.

Open File (Открыть Файл): Как и предполагает имя, этот способ открытия файла просто откроет любое определение, которое вы перетащите на холст.

Insert File (Вставьте Файл): Вы можете воспользоваться этим способом, чтобы вставить существующий файл в текущий документ как свободный компонент.

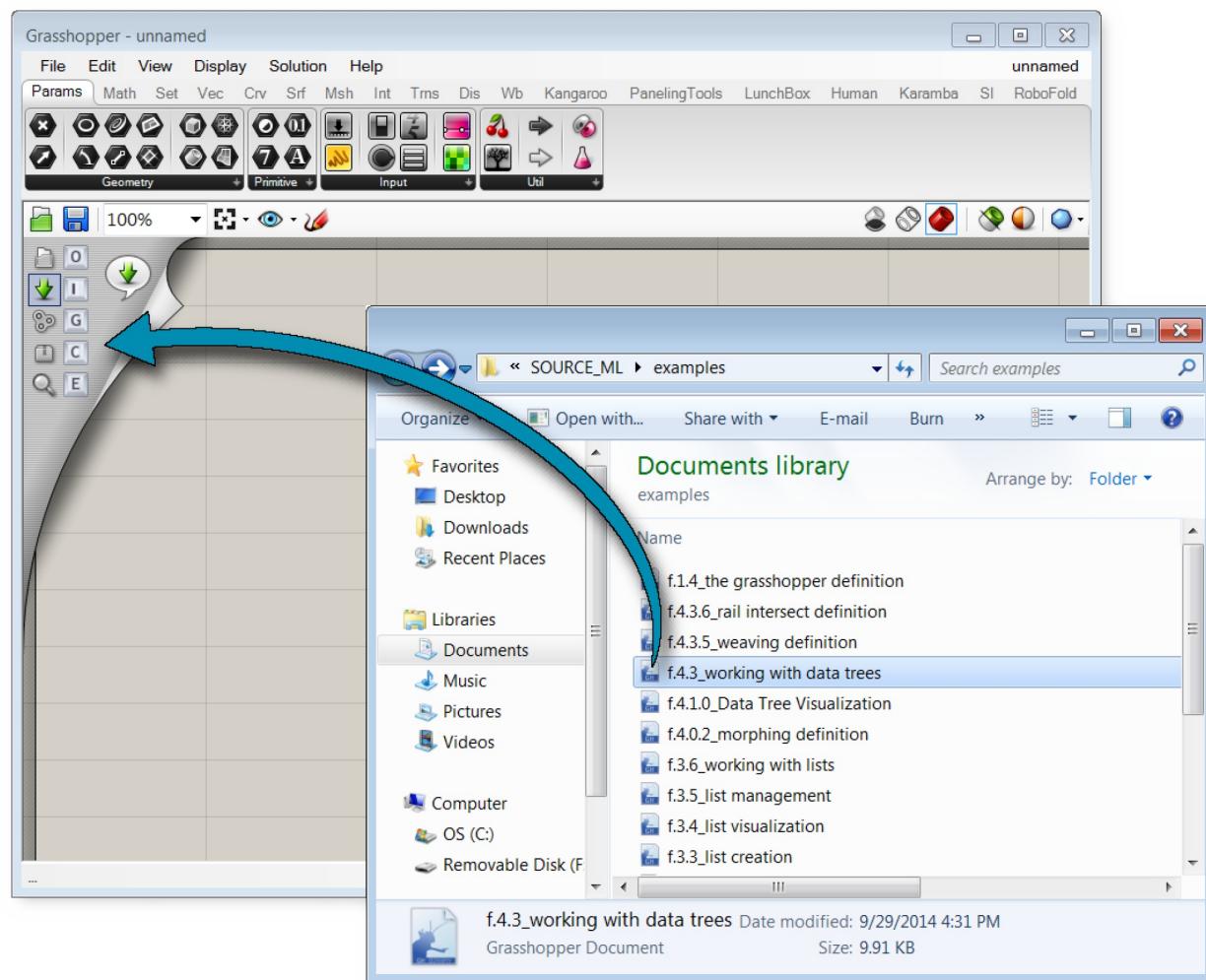
Group File (Сгруппировать Файл): Этот метод вставит файл в существующий документ, и сгруппирует все объекты вместе.

Cluster File (Кластеризовать Файл): Похоже на функцию Сгруппировать, этот метод вставит файл в существующий документ, но создаст объединенный объект (кластер) для целой группы объектов.

Examine File (Исследовать Файл): Позволяет вам открыть файл в защищенном режиме, то есть вы сможете рассмотреть определенный файл, но вы не сможете внести изменения в определение.

Grasshopper также обладает функцией Автосохранения, которая периодически будет срабатывать, основываясь на особых действиях пользователя. Список параметров Автосохранения находится под меню File на Строчке основного меню. При закрытии активной сессии Rhino, во всплывающем окне появится вопрос - хотите вы или нет сохранить какие-либо файлы Grasshopper, которые были открыты на момент закрытия Rhino.

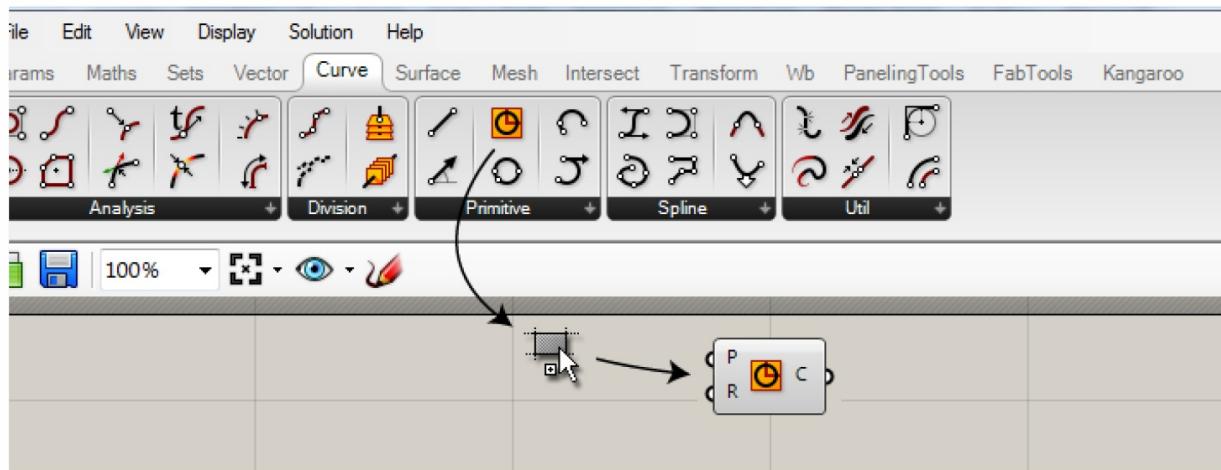
Автосохранение работает при условии, что файл уже однажды был сохранен, по крайней мере, один раз.



Файлы, которые перетащили на холст.

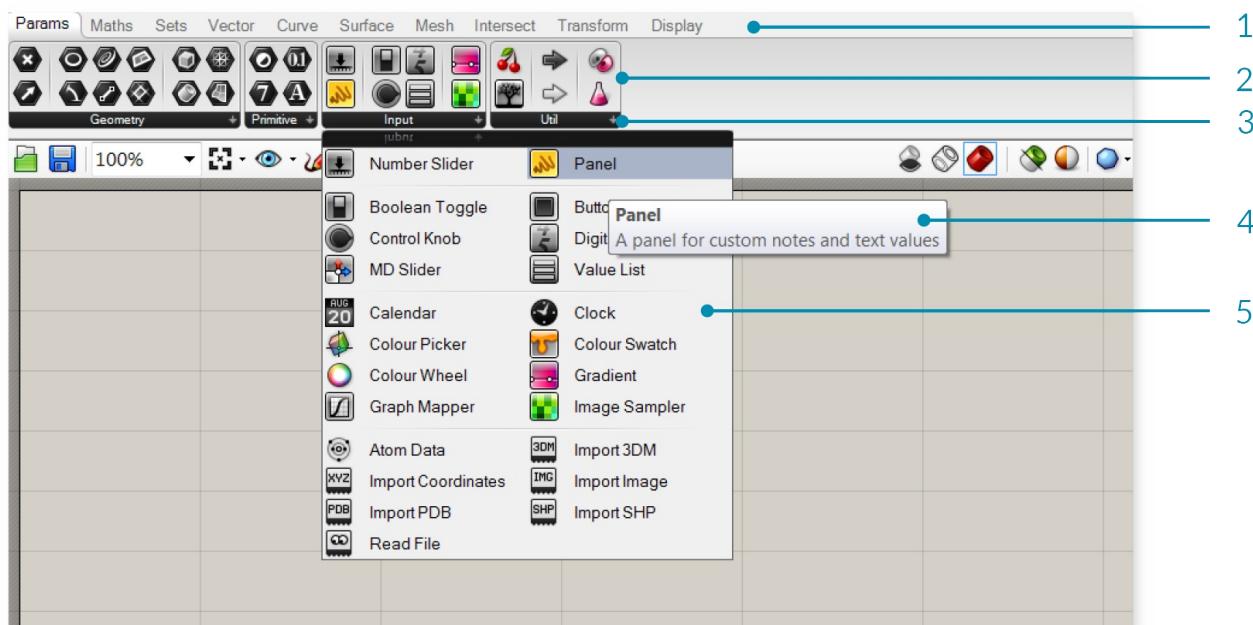
1.1.2.4. ПАНЕЛИ КОМПОНЕНТОВ

В этой области размещены компоненты, которые разделены на категории и свитки. Категории отображаются как вкладки, свитки отображаются как выпадающие панели. Все компоненты принадлежат к определенной категории. Эти категории были разбиты на классы, чтобы помочь вам найти определенный компонент (например, "Params" для всех примитивных типов данных или "Curves" для всех инструментов, связанных с кривыми). Два способа добавления компонента на холст: кликнуть на объект в выпадающем меню или перетащить компонент прямо из меню на холст.



Компонент, который перетащили с панели для добавления компонента на холст.

Так как в каждом свитке может быть гораздо больше компонентов, чем поместиться на палитру, на каждой панели отображается ограниченное число иконок. Высоту панели компонентов и ширину окна Grasshopper можно настроить так, чтобы отображалось больше или меньше компонентов в подкатегории. Чтобы посмотреть меню всех компонентов в данном свитке, просто кликните на черной строке внизу каждой панели свитка. Тогда откроется выпадающее меню, которое покажет все компоненты в данной подкатегории.



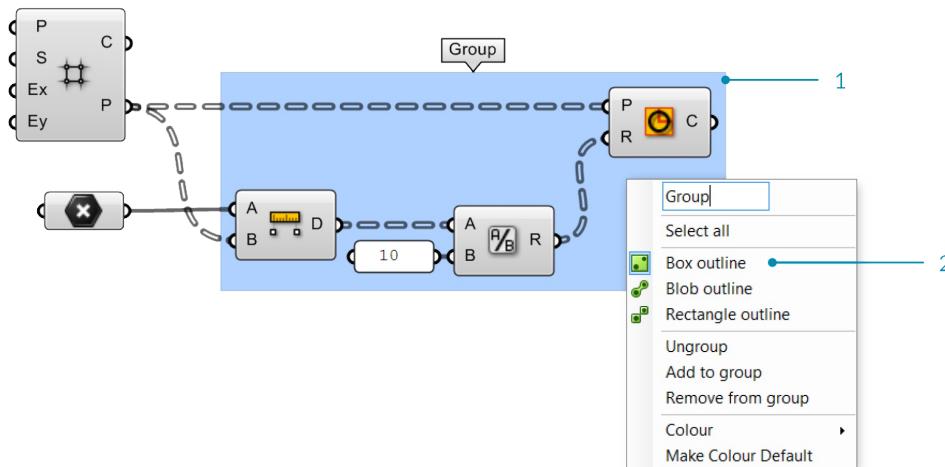
1. Вкладка категории
2. Панель свитка.
3. Кликните на черной строке, чтобы открыть панель меню свитка.
4. Наведите мышку на компонент для получения краткого описания.
5. Выпадающее меню.

1.1.2.5. ХОЛСТ

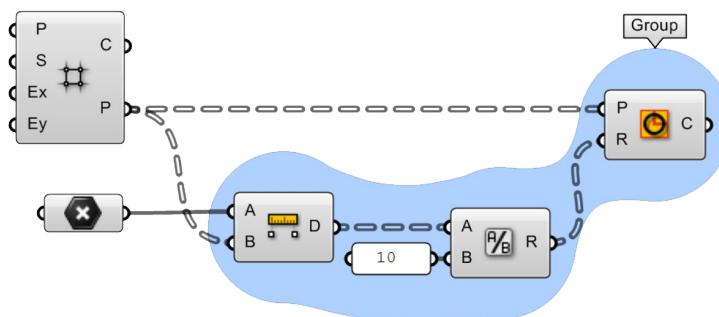
Холст - это основное рабочее пространство для создания определения Grasshopper. Здесь вы можете взаимодействовать с элементами вашего визуального программирования. Вы можете начать работать на холсте, размещая компоненты и соединяя их связями.

1.1.2.6. ГРУППИРОВАНИЕ

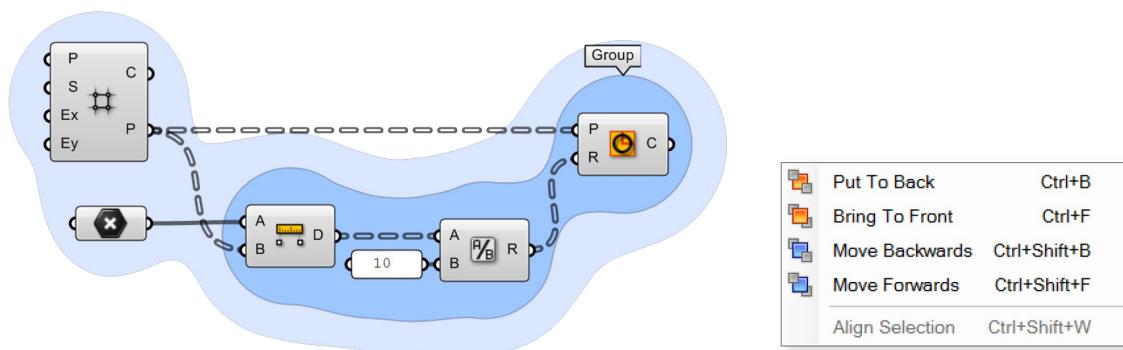
Группирование компонентов вместе на холсте может быть особенно полезно для разборчивости и доступности для восприятия. Группирование позволяет вам быстро выбирать и передвигать множество компонентов по холсту. Вы можете создать группу, набрав Ctrl+G при этом желаемый компонент должен быть выбран. Можно выбрать альтернативный метод группирования - нажав кнопку "Group Selection" (Выбор Группы) под Меню Edit на Строчке главного меню. Настройте параметры цвета группы, прозрачности, имени и типа контура кликнув правой кнопкой мыши на любом объекте группы.



1. Группа компонентов, очерченная прямоугольником.
2. Щелкните правой кнопкой мыши в любом месте группы для редактирования имени и вида группы.



Вы также можете обозначить группу, используя алгоритм метабол с использованием силуэта расплывчатого пятна.

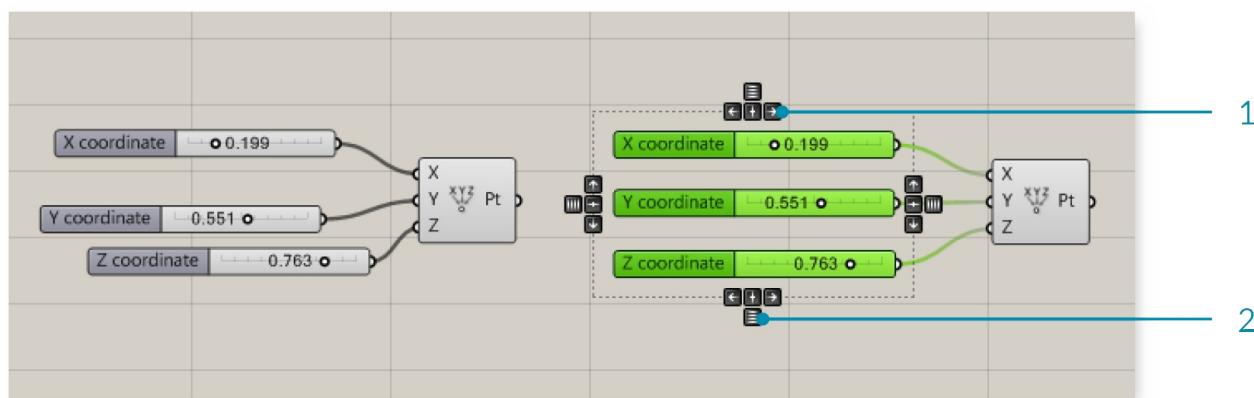


Две группы расположены одна внутри другой. Цвет (светло-голубой) был изменен у внешней группы для лучшей визуальной идентификации одной группы от другой. Группы начерчены "позади" компонентов и включают их, в таких случаях как этот, присутствует глубина цвета в соответствии и порядком двух групп. Чтобы изменить это, нажмите Edit>Arrange в строке главного меню.

1.1.1.7. ВИДЖЕТЫ

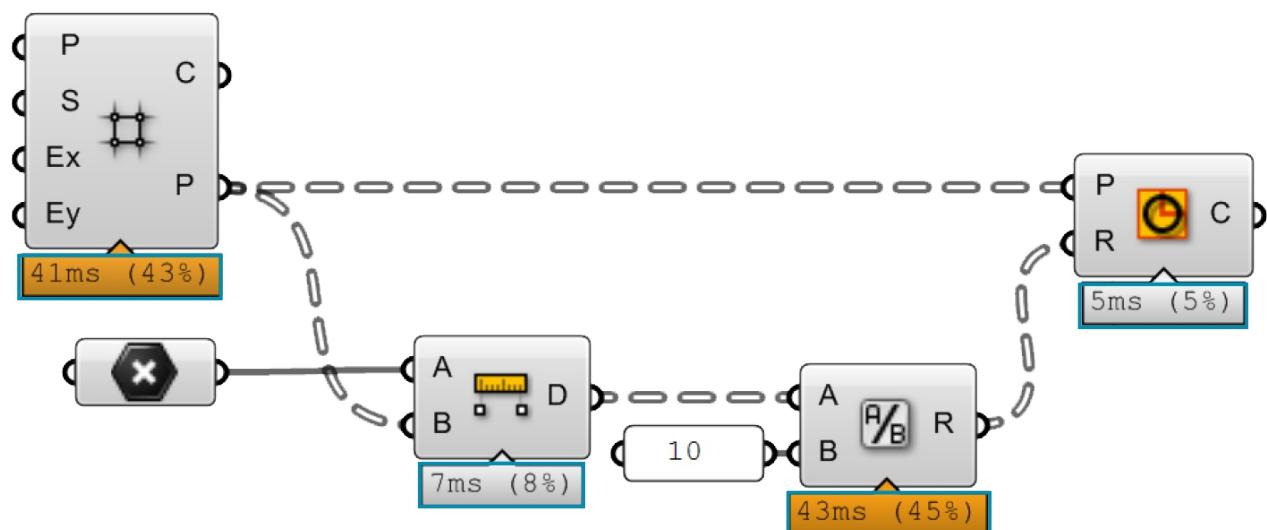
В Grasshopper доступны несколько виджетов, которые могут помочь вам выполнять полезные действия. Вы можете включить/выключить любые из этих виджетов под меню Display (дисплей/отображение) на Строчке главного меню. Ниже мы познакомимся с несколькими самыми часто используемыми виджетами.

The Align Widget Один из полезных виджетов для пользовательского интерфейса, который помогает поддерживать порядок на холсте - это Align виджет. Вы получите доступ к виджету Align, если выберите несколько компонентов одновременно и кликните на одной из опций, обнаруженных на пунктирном силуэте, которые окружают выбранные вами компоненты. Вы можете выравнивать по левой стороне, вертикально по центру, по правой стороне, по верху, горизонтально по центру, по низу или расположить компоненты равномерно через этот интерфейс. В самом начале вы можете столкнуться с тем, что инструменты мешают при работе (есть вероятность совершить ошибку при накладывании нескольких компонентов один на другой). Тем не менее, после небольшой практики эти инструменты могут стать бесценными, когда вы начнете структурировать графики, чтобы они были удобочитаемыми и понятными.



1. Выровнять по правой стороне.
2. Распределить по вертикали.

The Profiler Widget Этот виджет выдает список наихудшего времени выполнения для параметров и компонентов, позволяя отслеживать узкие места в сети и сравнивать различные компоненты в плане затрат на выполнение. Помните, что этот виджет отключен по умолчанию.

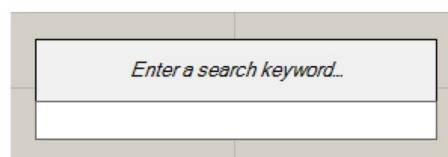


Виджет Profiler предоставляет визуальную обратную связь по тому, какие компоненты в вашем определении могут занять длительное время на выполнение операции.

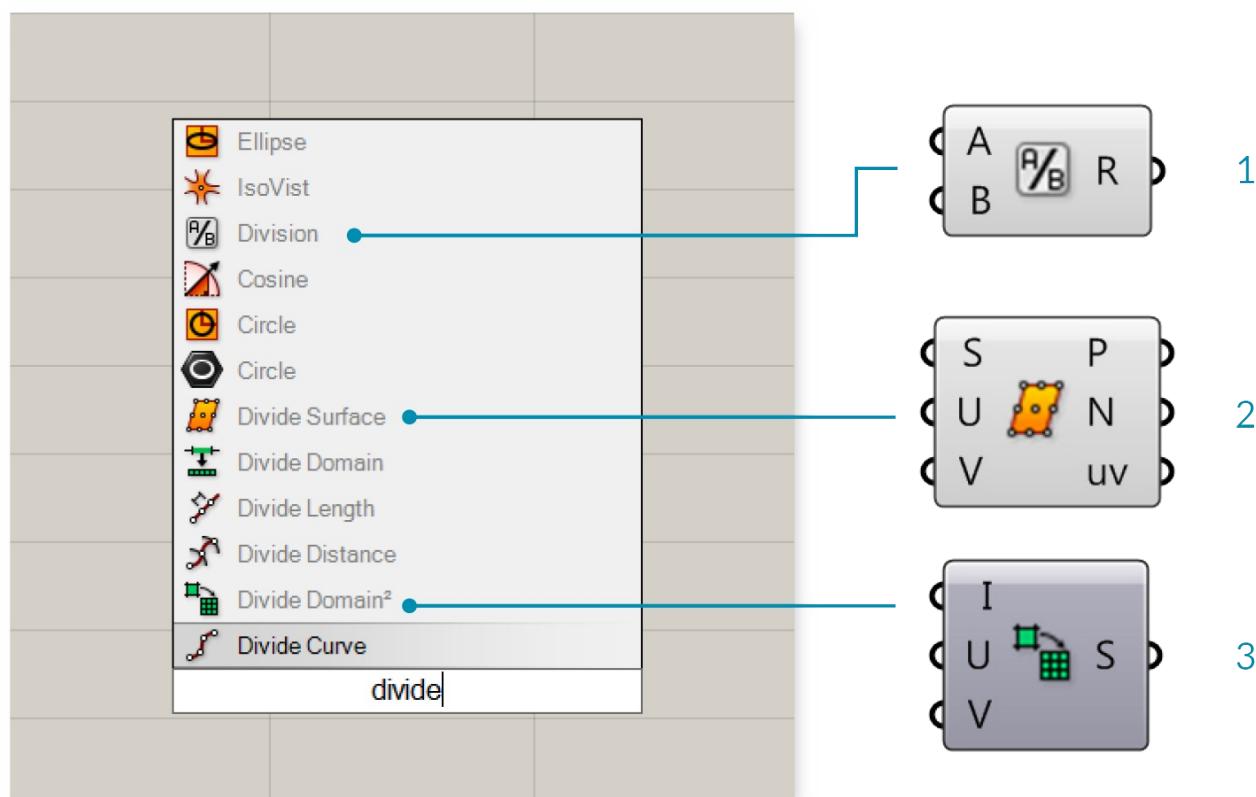
The Markov Widget Этот виджет использует цепь Маркова, чтобы предугадать какой компонент вы захотите использовать следующим, основываясь на вашем предыдущем поведении. Цепь Маркова - это процесс, который состоит из четного числа состояний (или уровней) и некоторых известных возможностей. Этому виджету потребуется некоторое время для настройки под определенного пользователя, но спустя некоторое время вы должны начать замечать, что этот виджет начинает предлагать компоненты, которые вы хотите использовать следующими. Виджет Markov может предлагать до 5 возможных компонентов в зависимости от недавней активности. Кликните правой кнопкой мыши по виджету (по умолчанию располагается в левом нижнем углу холста), чтобы разметить его в одном из других углов холста или полностью спрятать.

1.1.2.8. ИСПОЛЬЗОВАНИЕ ФУНКЦИИ ПОИСКА

Несмотря на то, что было много размышлений насчет того, как лучше расположить каждый компонент на панели, чтобы это было интуитивно легко для новых пользователей, им все равно иногда трудно найти определенный компонент, который может быть расположен в глубинах панелей категорий. К счастью, вы также можете находить компоненты по имени, дважды кликнув в любом пустом месте на холсте. Сделав это, вы вызовете окно поиска. Далее, просто наберите название компонента, появится список параметров или компонентов, подходящих под ваш поиск.



Кликните дважды в любом месте на холсте, чтобы вызвать поиск по ключевым словам для поиска определенного компонента.



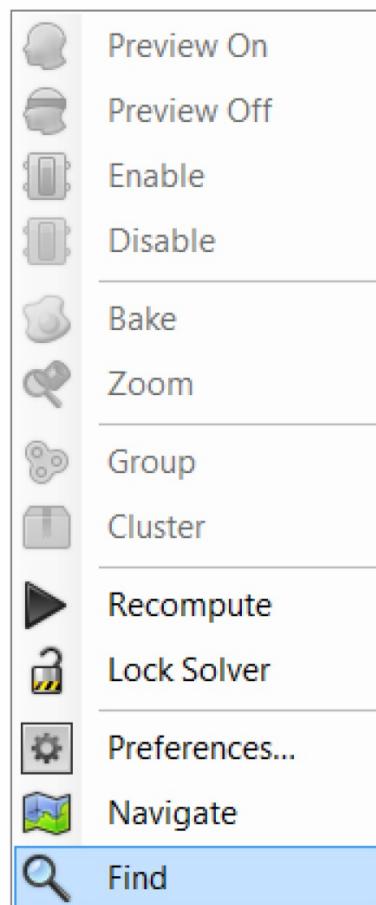
Набрав "divide", вы получите список различных компонентов.

1. Компонент Division Operator.
2. Компонент Divide Surface.
3. Компонент Divide Domain2.

1.1.2.9. ФУНКЦИЯ ПОИСКА

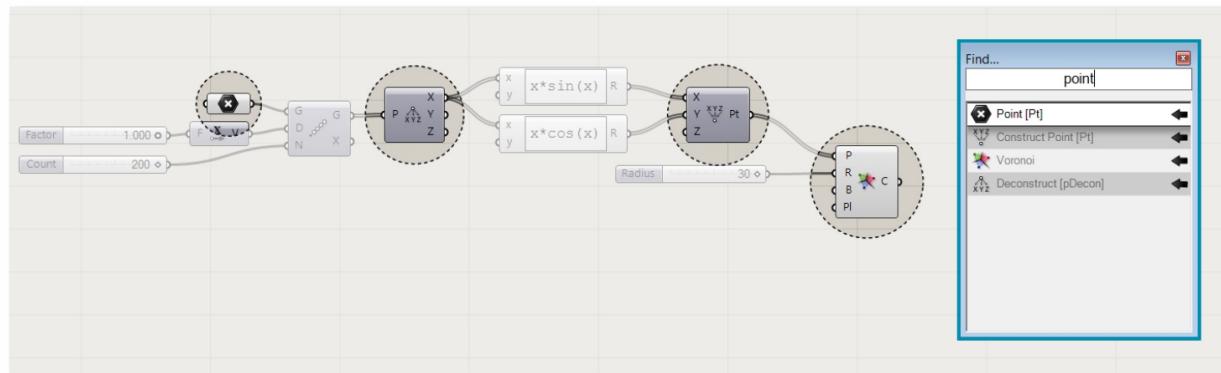
Существуют, в буквальном смысле, сотни (если не тысячи) компонентов Grasshopper, которые вы можете использовать, и вас, как новичка, может испугать поиск определенного компонента на Панели Компонентов. Быстрое решение - это дважды кликнуть в любом месте холста, чтобы запустить поиск необходимого компонента. Тем не менее, что если нам требуется найти определенный компонент, уже расположенный на нашем холсте? Беспокоиться не о чем. Кликнув правой клавишей мыши на холсте или нажав F3, вы вызовете функцию поиска. Начните печатать название компонента, который вы ищете.

Функция поиска использует некоторые сложно организованные алгоритмы, которые осуществляют поиск не только по отдельным наименованиям компонентов внутри определения (наименование компонента - это название компонента, которое находится под Панелью Компонентов и которое мы, как пользователи, не может изменить), но и также по любым уникальным признакам, которые мы присвоили определенным компонентам (известным как псевдоним или прозвище). Функция поиска также может осуществлять поиск по типу компонента на холсте или поиск по текстовой панели, наброскам и групповому контенту. Как только функция поиска нашла совпадение, она автоматически выделит серым цветом остальное определение и проведет пунктирную линию вокруг выделенного компонента. Если будут обнаружены несколько совпадений, список компонентов, отвечающих вашему поиску, будут отображены в диалоговом окне и будут находиться поверх наименования из списка, которое будет подсвечено зеленым цветом.

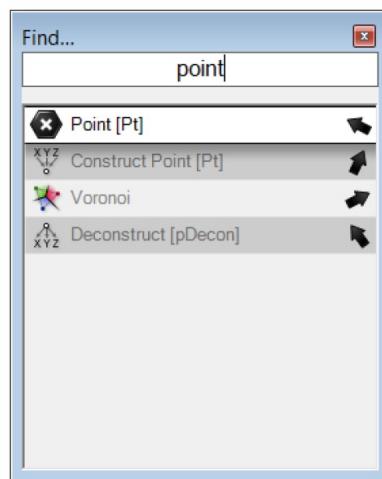


Кликнув правой клавишей мыши на холсте или нажав F3, вы вызовете функцию поиска. Начните

печатать название компонента, который вы ищете.



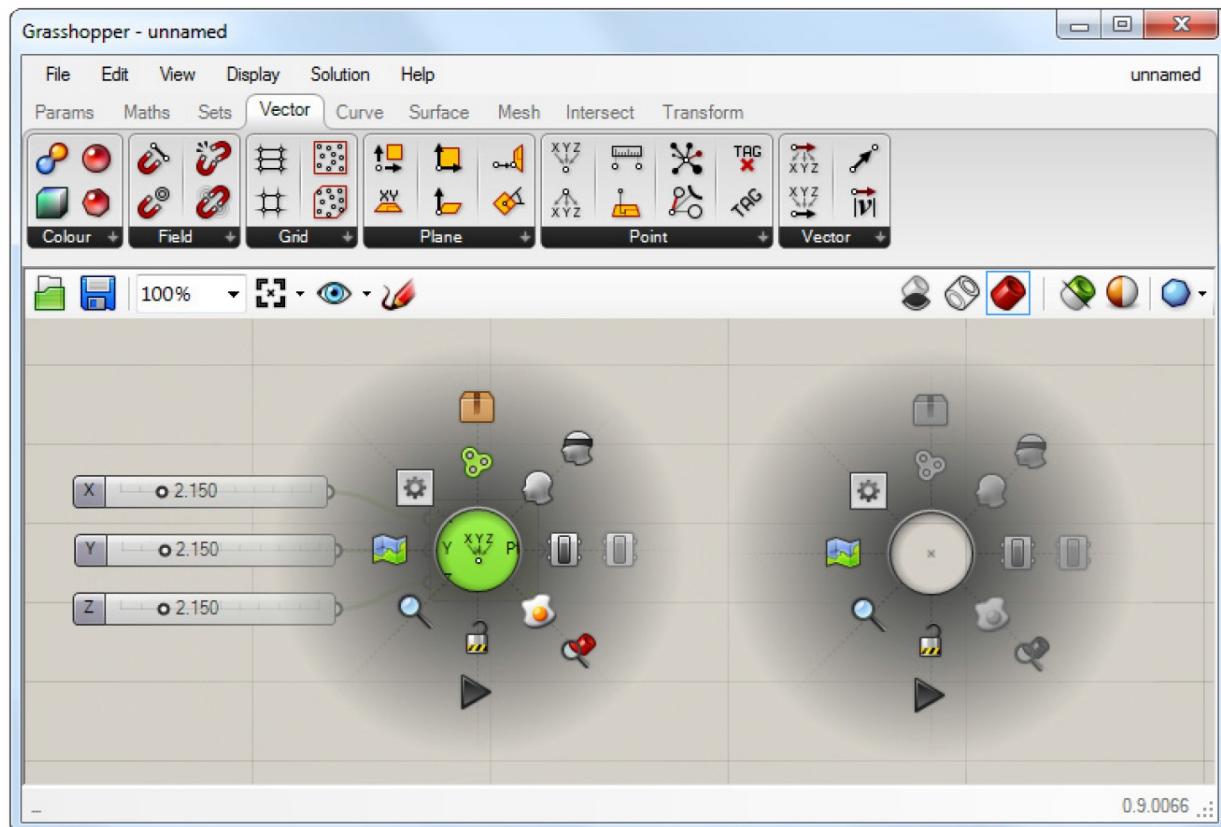
Функция поиска может быть очень полезной для нахождения определенного компонента на холсте. Кликните правой клавишей мыши по холсту, чтобы запустить окно поиска.



Маленькая стрелка также будет отображаться рядом с каждым элементом в списке, указывая на соответствующий компонент на холсте. Попробуйте покрутить поисковое диалоговое окно по холсту и посмотрите, как стрелки вращаются, чтобы отследить их компоненты. Кликнув по результатам поиска, компонент разместится (на холсте) рядом с диалоговым окном Поиска.

1.1.2.10. ИСПОЛЬЗОВАНИЕ РАДИАЛЬНОГО МЕНЮ

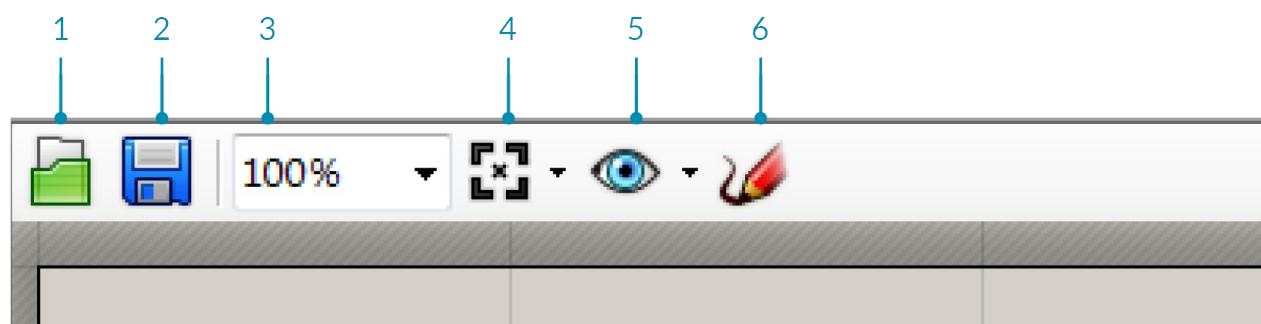
По мере того, как вы становитесь большими профессионалами в использовании интерфейса Grasshopper, вы будете находить способы ускорить вашу работу. Один из таких способов - это использование быстрых клавиш. Также существует и другой способ быстрого доступа к некоторым полезным инструментам - радиальное меню UI. Вы можете вызвать радиальное меню, нажав клавишу пробела (в то время как ваша мышка находится над холстом или компонентом) или кликнув средней клавишей мыши. Радиальное меню включает различные инструменты, в зависимости от того, вызвали ли вы меню кликнув прямо над компонентом или просто в любом месте холста. На изображении ниже, вы можете увидеть, что радиальное меню имеет больше доступных функций при клике над выбранным компонентом в отличие от клика просто в любом месте холста. Использование этого меню может значительно ускорить вашу скорость работы с документами Grasshopper.



UI радиального меню позволит быстро получить доступ к часто используемым пунктам меню.

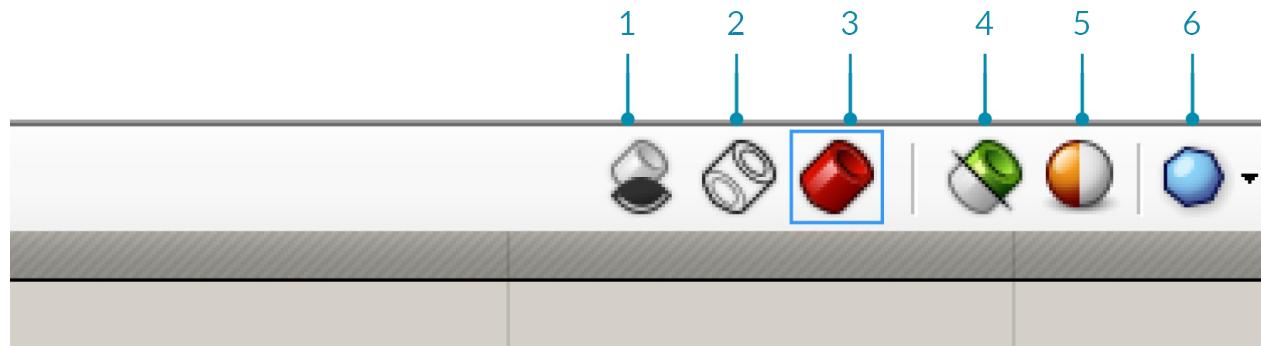
1.1.2.11. ПАНЕЛЬ ИНСТРУМЕНТОВ ХОЛСТА

Панель инструментов холста предоставляет быстрый доступ к часто используемым компонентам Grasshopper. Все инструменты доступны через меню, вы можете спрятать панель инструментов, если захотите. Панель инструментов можно редактировать во вкладке View (вид) на Строчке главного меню.

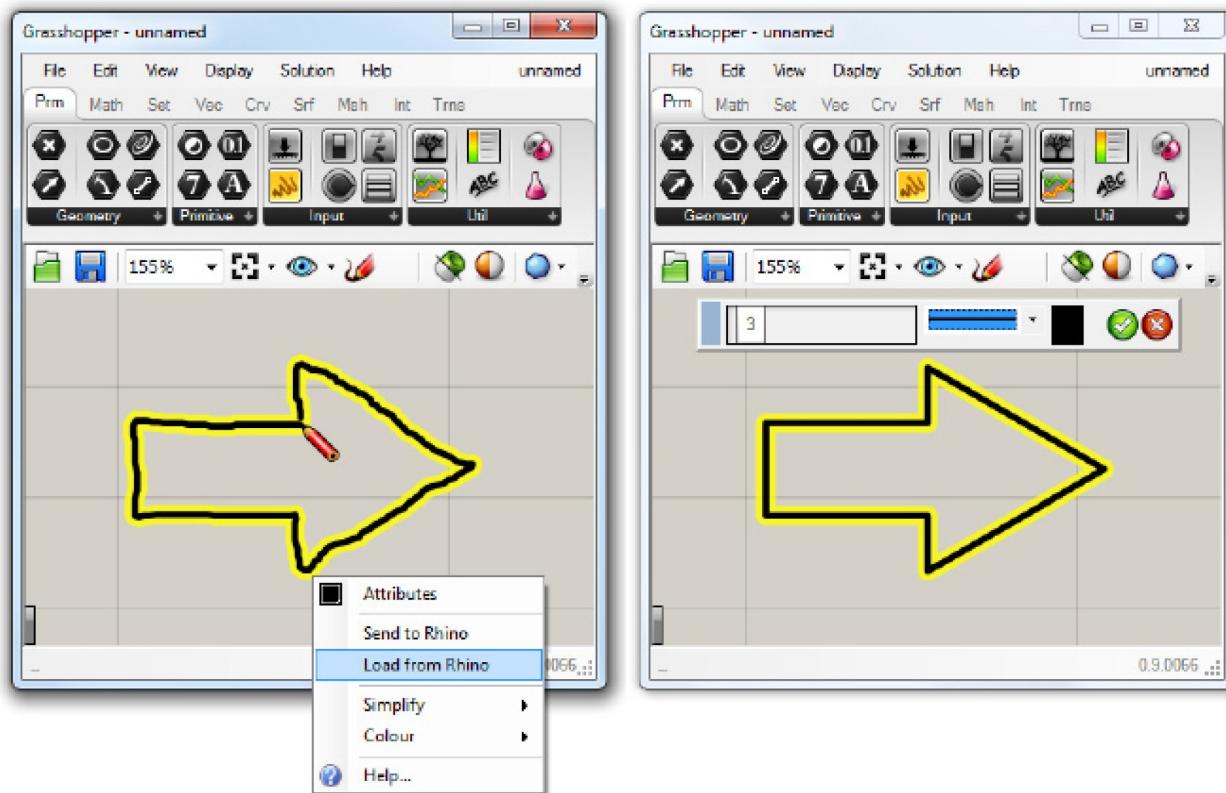


1. **Open File (Открыть Файл):** Ссылка для открытия файла Grasshopper.
2. **Save File (Сохранить файл):** Ссылка для сохранения текущего файла Grasshopper.
3. **Zoom Defaults (Масштаб по умолчанию):** Настройки масштаба по умолчанию позволяют вам увеличивать или уменьшать холст по предустановленным интервалам.
4. **Zoom Extents (Масштаб):** Масштабируйте по размеру вашего определения. Кликните по стрелке рядом с иконкой Zoom Extents, чтобы выбрать один из элементов подменю для масштабирования определенной области вашего определения.
5. **Named Views (Именнованный просмотр):** Эта функция раскрывает меню, позволяя вам разместить или переименовать любую область просмотра в вашем определении.
6. **The Sketch Tool (Инструмент для скетчей):** Этот инструмент работает также как инструмент

"карандаш" в программе Adobe Photoshop, но имеет несколько дополнительных функций.

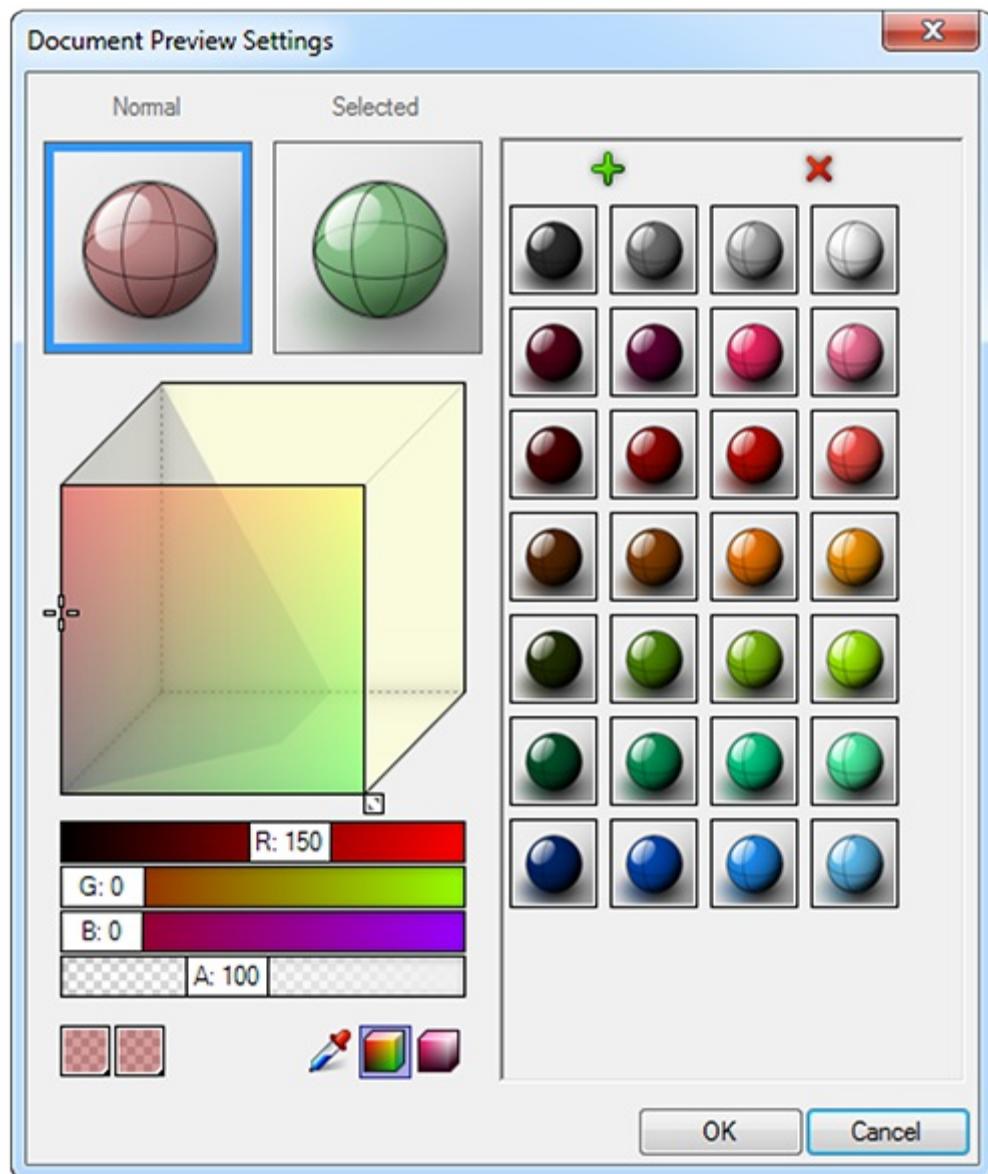


1. **Настройки просмотра:** Если компонент Grasshopper генерирует какую-либо геометрию, тогда предпросмотр этой геометрии будет происходить в видовом окне по умолчанию. Вы можете отключить предпросмотр для каждого отдельного объекта правым кликом мыши по каждому компоненту и отключить функцию предпросмотра, либо вы можете глобально изменить способы предпросмотра, используя одну из трех кнопок.
2. Режим просмотра структуры / каркаса.
3. Отключить предпросмотр.
4. Просмотр с тенью (по умолчанию).
5. **Предпросмотр выбранного объекта:** При нажатии этой кнопки, Grasshopper будет отображать геометрию, которая является частью выбранных компонентов, даже если у тех компонентов выключен режим предпросмотра.
6. **Настройки просмотра документа:** В Grasshopper по умолчанию настроена цветовая схема для выбранных (полупрозрачный зеленый) и невыбранных (полупрозрачный красный) геометрий. Эти цвета можно переназначить с использованием диалогового окна Настройки просмотра документа.
7. **Предпросмотр качества Mesh:** В целях оптимизации эти настройки позволят вам контролировать качество отображения mesh/поверхности той геометрии, которая была отрендерена в Rhino. Высокие настройки качества станут причиной долгих просчетов, в то время как низкие настройки будут отображать не аккуратную геометрию при просмотре. Следует заметить, что геометрия все равно будет сохранять высокое разрешение при запекании файла в Rhino. Эти настройки, в основном, влияют на выполнение отображения и качество.



Скетч инструмент позволяет поменять толщину линии, тип линии и цвет. Кликнув правой клавишей мыши по выбранному скетч-объекту, вы сможете выбрать упростить линию, чтобы создать эффект сглаживания. Кликните правой клавишей мыши по скетч-объекту и выберите "Load from Rhino". При появлении подсказки выберите любую 2D форму в среде Rhino. Как только вы выбрали исходную форму, нажмите Enter, и предыдущий скетч будет отредактирован по вашей исходной форме в Rhino.

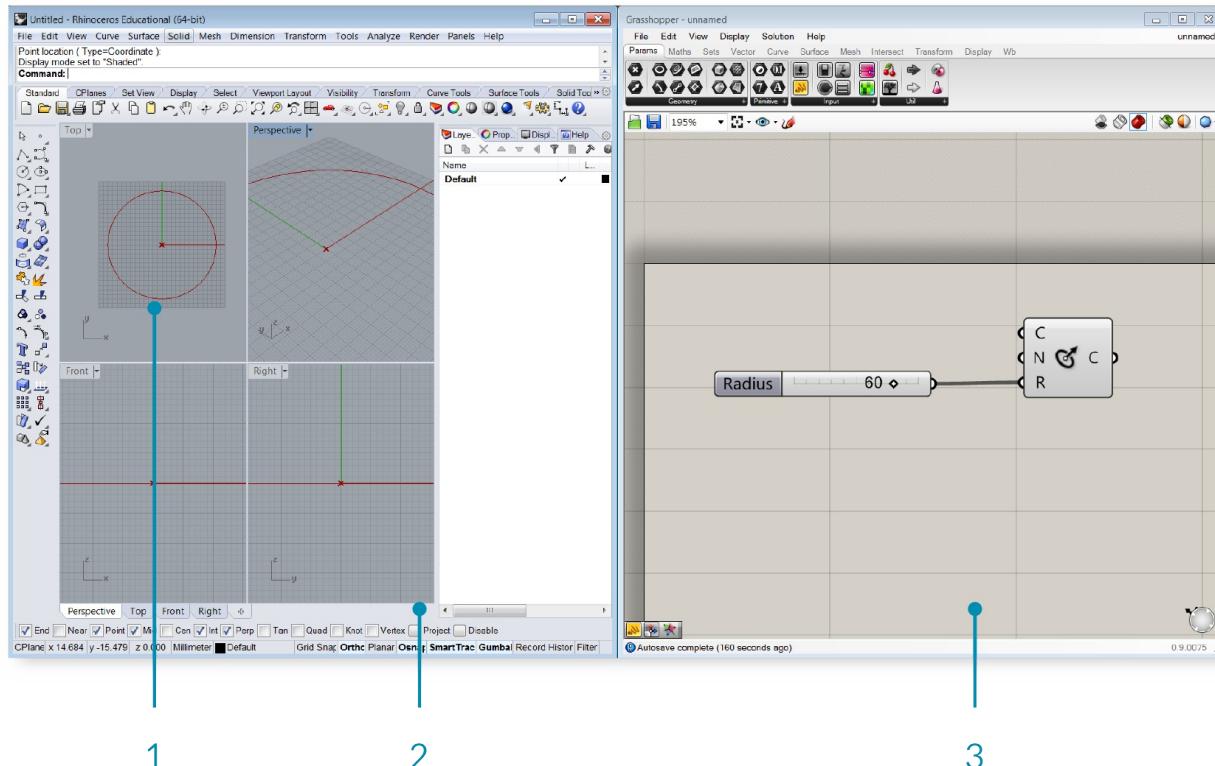
Примечание: Ваш скетч-объект может сместиться с первоначального положения, когда вы загрузите форму из Rhino. Grasshopper располагает ваш скетч-объект относительно исходной точки холста (верхний левый угол) и мировых координат в Rhino.



В Grasshopper, по умолчанию, настроена цветовая схема для выбранных (полупрозрачный зеленый) и невыбранных (полупрозрачный красный) геометрий. Эти цвета можно переназначить с использованием диалогового окна Настройки просмотра документа.

1.1.3. ВЗАИМОДЕЙСТВИЕ С RHINO

В отличие от документа Rhino, определение Grasshopper не содержит реального объекта или геометрии. Вместо этого, определение Grasshopper представляет собой набор правил и инструкций о том, как Rhino может автоматизировать задачи.

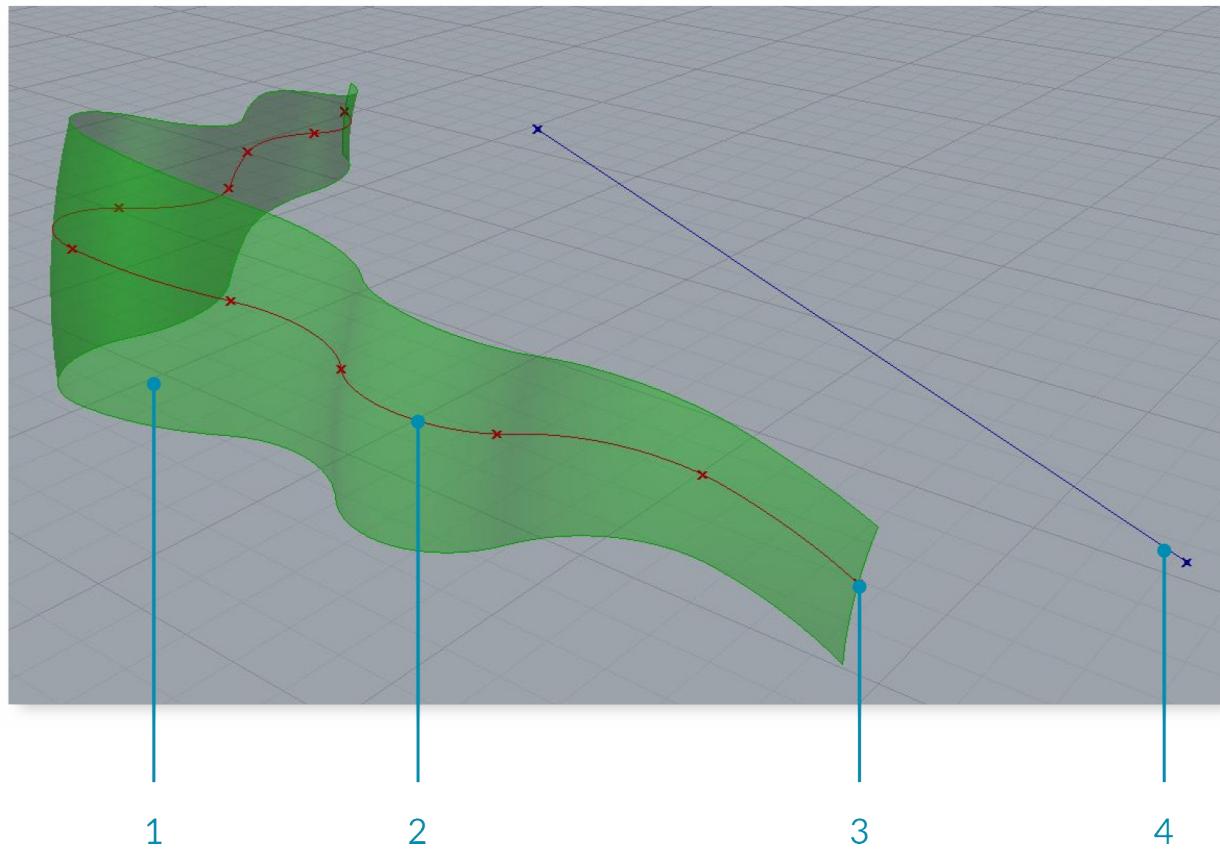


1. Предпросмотр геометрии Grasshopper.
2. Видовые окна Rhino.
3. Окно Grasshopper.

1.1.3.1. ОБРАТНАЯ СВЯЗЬ С ВИДОВЫМ ОКНОМ

Вся геометрия, которая генерируется при использовании различных компонентов Grasshopper, будет отображаться (по умолчанию) в видовом окне Rhino. Этот предпросмотр является, всего лишь, OpenGL аппроксимацией действительной геометрии и, поэтому, вы не сможете выбрать геометрию в видовом окне Rhino (сначала вам нужно запечь ее). Вы можете отключить / включить предпросмотр геометрии правым кликом мыши по компоненту и выбрав Preview toggle. Геометрия в видовом окне кодируется цветом, чтобы предоставить визуальный отклик. Изображение ниже представляет цветовую схему по умолчанию.

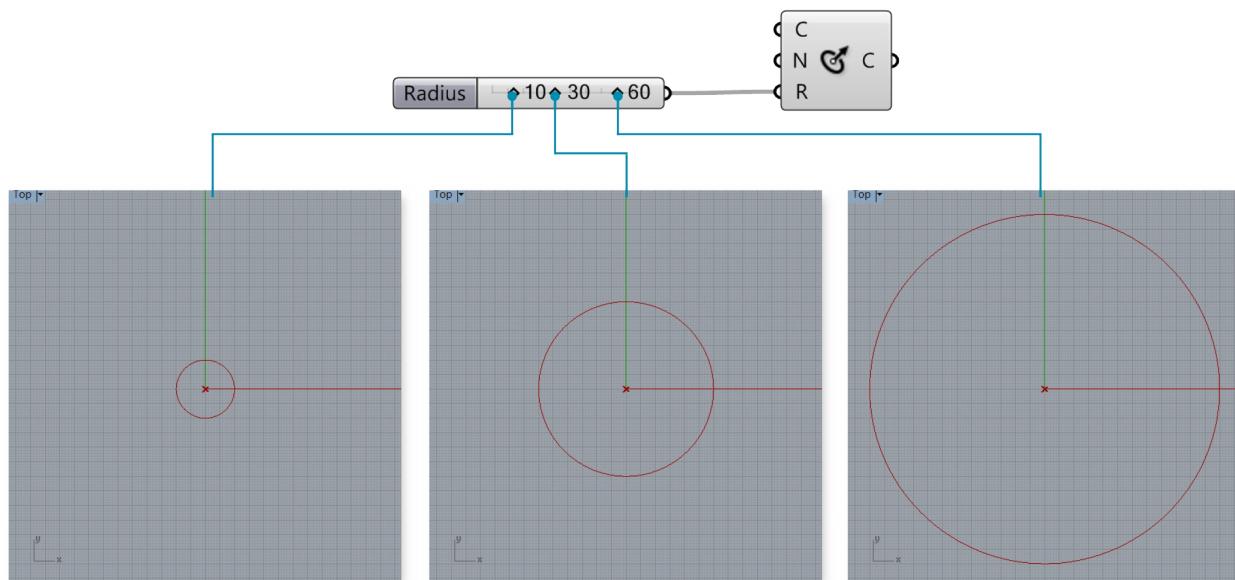
Примечание: Это цветовая схема по умолчанию, которую можно изменять, используя настройки просмотра документа на панели инструментов холста.



1. Зеленый цвет в видовом окне относится к компоненту, который выбран в текущий момент.
2. Красный цвет в видовом окне относится к компоненту, который не выбран в текущий момент.
3. Геометрия точек отображается в виде крестов, а не прямоугольников, для отличия ее от других точечных объектов Rhino.
4. Голубой цвет означает, что в данный момент вы выбираете объект в видовом окне Rhino.

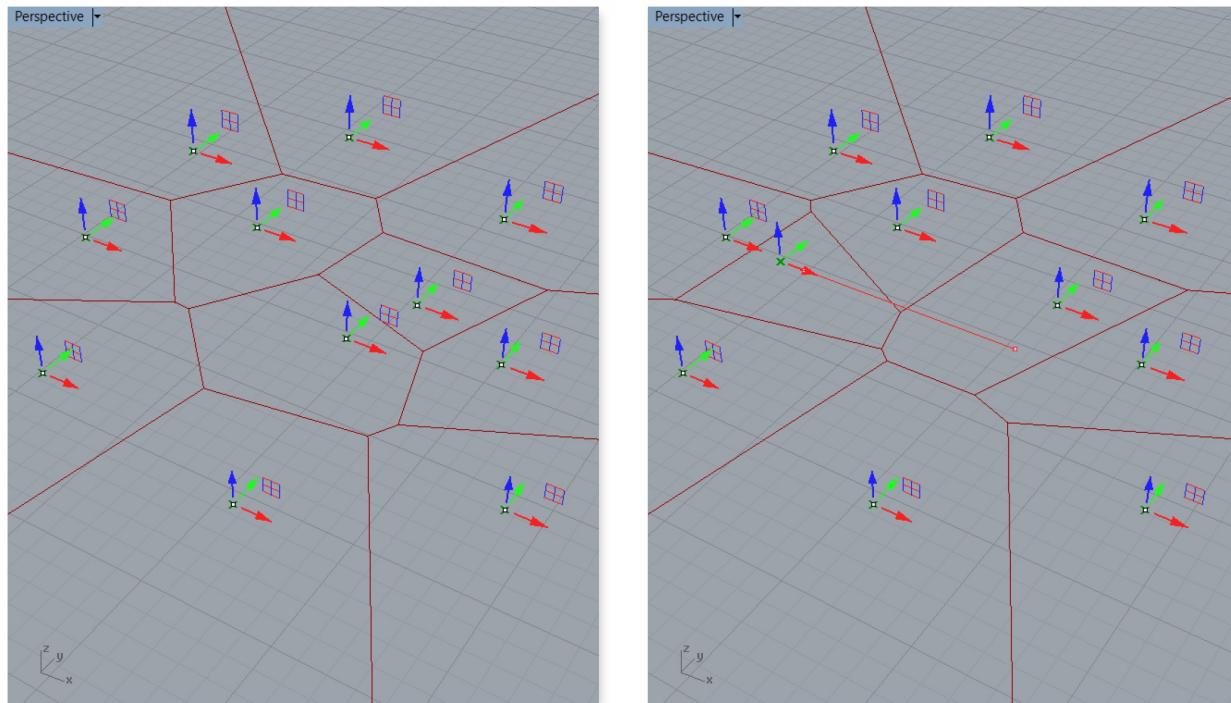
1.1.3.2. СВЯЗИ

Grasshopper - это динамическая среда. Изменения, которые происходят в реальном времени и их предпросмотр, обновляются в видовом окне Rhino.



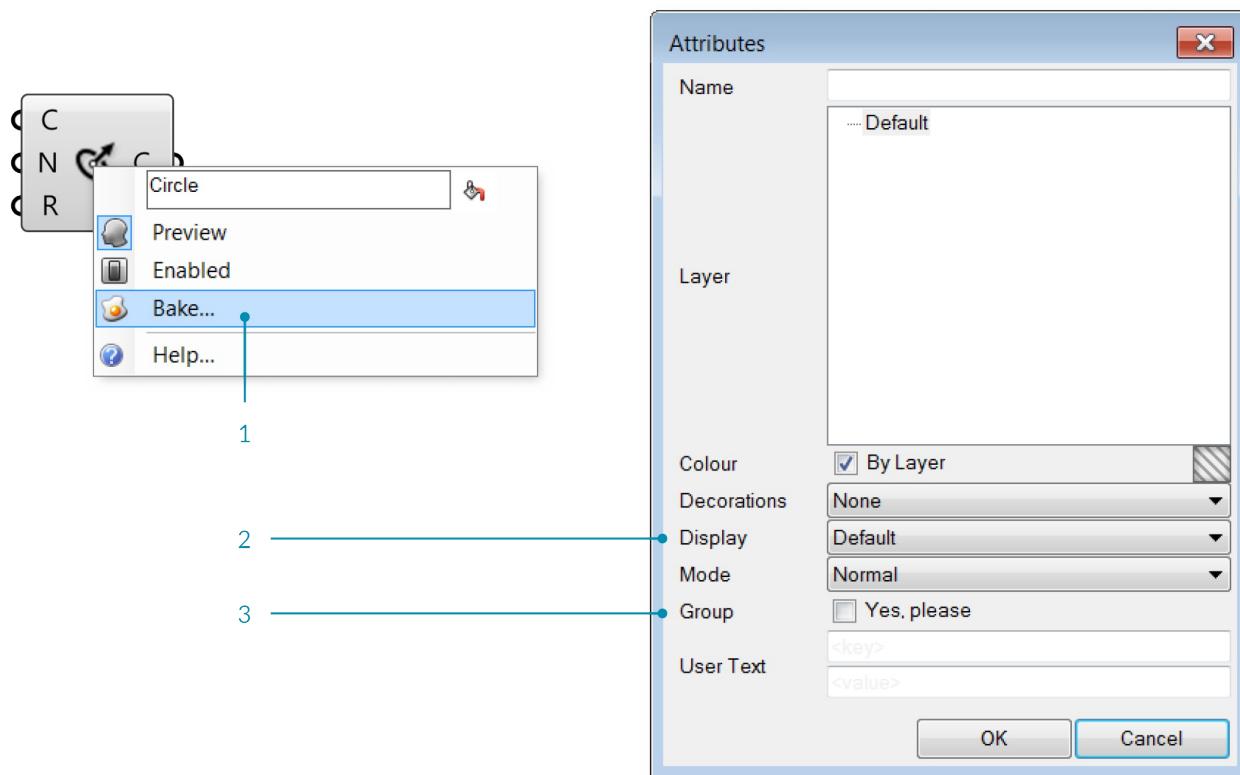
1.1.3.3. ВИДЖЕТ GUMBALL

При сохранении геометрии как внутренней в параметре Grasshopper, Gumball позволяет вам взаимодействовать с геометрией в видовом окне Rhino. При манипулировании Gumball, обновление будет происходить сразу же. Напротив, геометрия, исходящая из Rhino, будет и дальше существовать в документе Rhino и обновления из Grasshopper произойдут только после того, как произойдут какие-либо изменения.



1.1.3.4. "ЗАПЕКАНИЕ" ГЕОМЕТРИИ

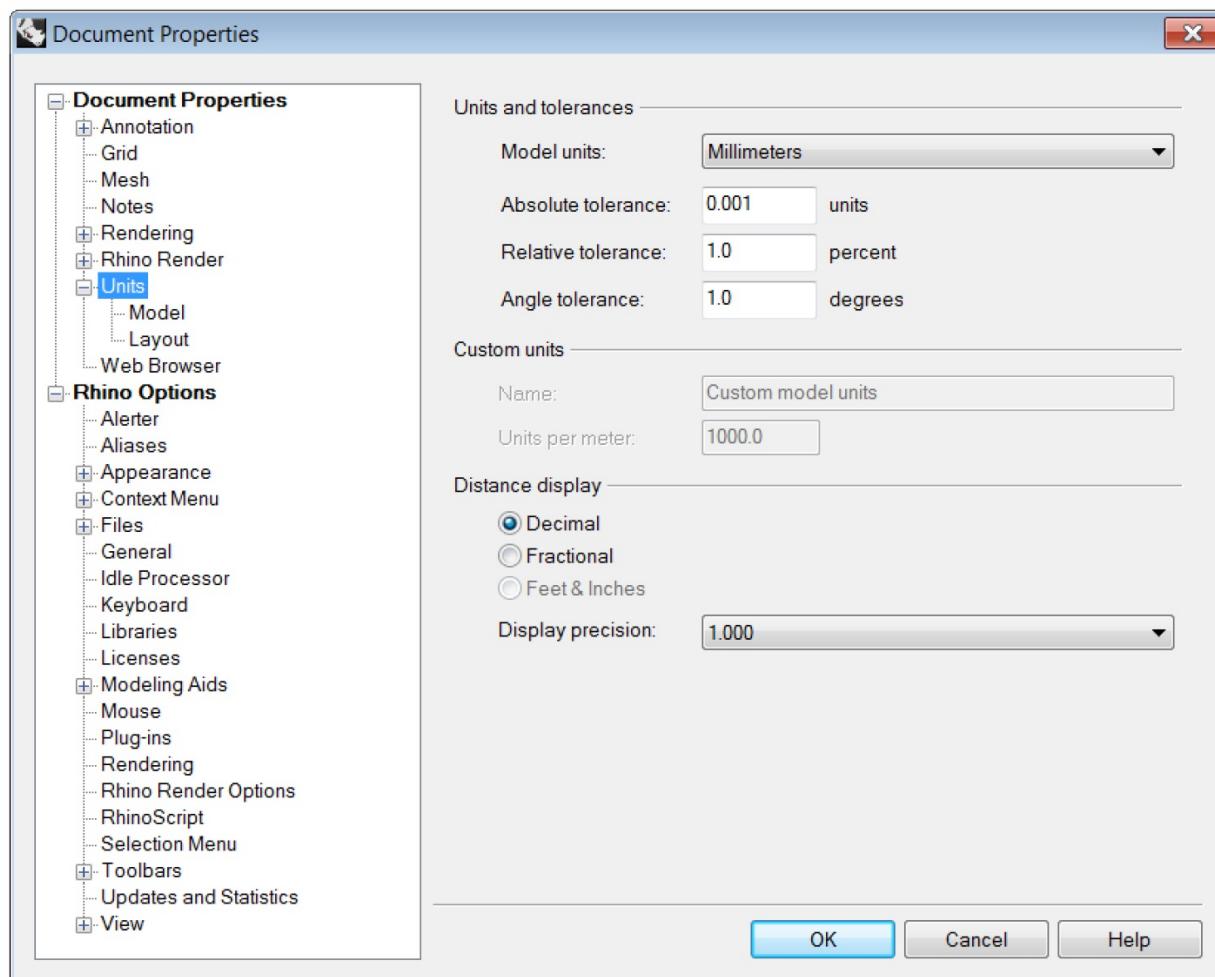
Для того, чтобы работать с геометрией в Rhino (выбрать, редактировать, изменять и т.д.), которая была создана в Grasshopper, вы должны "запечь" ее. "Запекание" создает новую геометрию в Rhino, берущей за основу текущее состояние из Grasshopper. Больше геометрия не будет реагировать на изменения в вашем определении.



- Чтобы запечь геометрию, кликните правой клавишей мыши на компоненте и выберите Bake.
- Появится диалоговое окно, в котором вы сможете выбрать, в какой слой Rhino вы хотите "запечь" геометрию.
- Группирование "запеченой" геометрии является удобным способом управления созданной в Rhino геометрией, особенно, если вы создаете много объектов в Grasshopper.

1.1.3.5. ЕДИНИЦЫ ИЗМЕРЕНИЯ И ТОЧНОСТЬ

Grasshopper наследует единицы измерения и точность от Rhino. Чтобы изменить единицы измерения, впишите в командную строку Rhino Document Properties, чтобы зайти в меню Document Properties. Выберите Units, чтобы изменить единицы измерения и точность.



Измените единицы измерения и точность в Rhino в меню Document Properties.

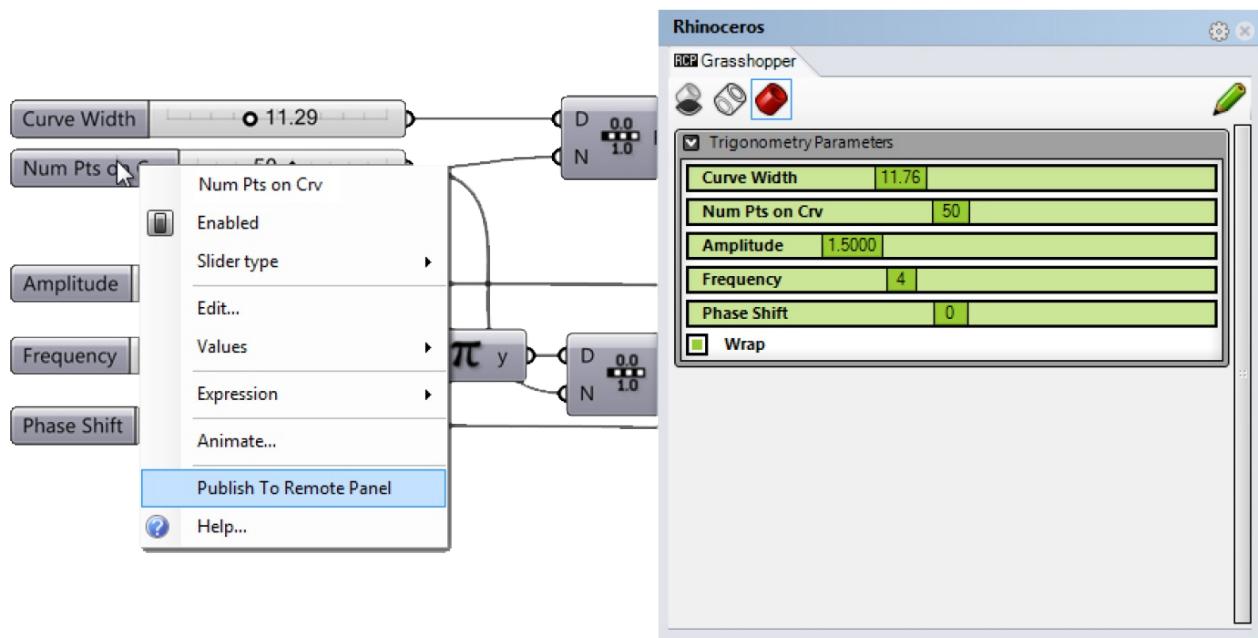
1.1.3.6. ПАНЕЛЬ ДИСТАНЦИОННОГО УПРАВЛЕНИЯ

Как только вы научитесь использовать ее, вы убедитесь, что Grasshopper - это невероятно мощный и гибкий инструмент, который позволяет исследовать итерации проектирования, используя графический интерфейс. Тем не менее, если вы работаете с одним экраном, вы уже могли заметить, что Grasshopper занимает много пространства экрана. И другого, более удобного, решения этой проблемы, только как постоянное приближение и отдаление окон, не существует. Так было, пока не выпустили панель дистанционного управления!

Панель дистанционного управления (RCP) предоставляет минимальный интерфейс для осуществления контроля над вашим определением и не занимает львиную долю вашего экрана. RCP можно создать кликнув по кнопке под меню View (вид) на панели главного меню. По умолчанию, RCP отключен, т.е. не содержит никакой информации о вашем текущем файле Grasshopper. Чтобы наполнить RCP такими элементами UI как слайдеры, переключатели и кнопки, просто кликните по элементу и кликните Publish To Remote Panel (вывести на RCP). Так вы создадите новую группу и синхронизируйте элементы UI с RCP. Изменяя значение элемента в RCP, вы также обновите значение в графике, а также изменит любую геометрию в видовом окне, которая может зависеть от этого параметра. Вы можете опубликовать многочисленные элементы и создать полный интерфейс, который можно использовать для контроля вашего файла, при этом не создавать бардак из визуальных графиков выделяющихся поверх видовых окон Rhino.

Примечание: RCP перенимает название элементов UI и использует их как ярлыки. Принято обновлять названия слайдеров и переключателей на четкие и понятные названия. Это обновление также

переместится прямо в RCP и сделает его использование проще.



Чтобы достать элемент UI (например, слайдер, переключатель, кнопки и т.д.) и отобразить его в RCP сначала необходимо опубликовать его.

UI RCP также можно настроить для себя и это позволит вам контролировать появление объектов на интерфейсе, названия и цвета различных групп. Чтобы изменить слой RCP, вам сначала надо переключиться из режима Working Mode (по умолчанию) в Edit Mode. Вы можете перейти в Edit Mode, кликнув по зеленому карандашу в правом верхнем углу RCP. Находясь в Edit Mode, вы можете создавать новые группы UI, перегруппировать элементы внутри группы, добавлять ярлыки, менять цвета и др. Чтобы удалить элемент UI, просто перетащите элемент за границы RCP. Вы не можете изменять индивидуальные значения параметров, если вы находитесь в Edit Mode. Вместо этого, вам необходимо кликнуть на иконку зеленого карандаша, чтобы переключиться обратно в стандартный режим (Working).

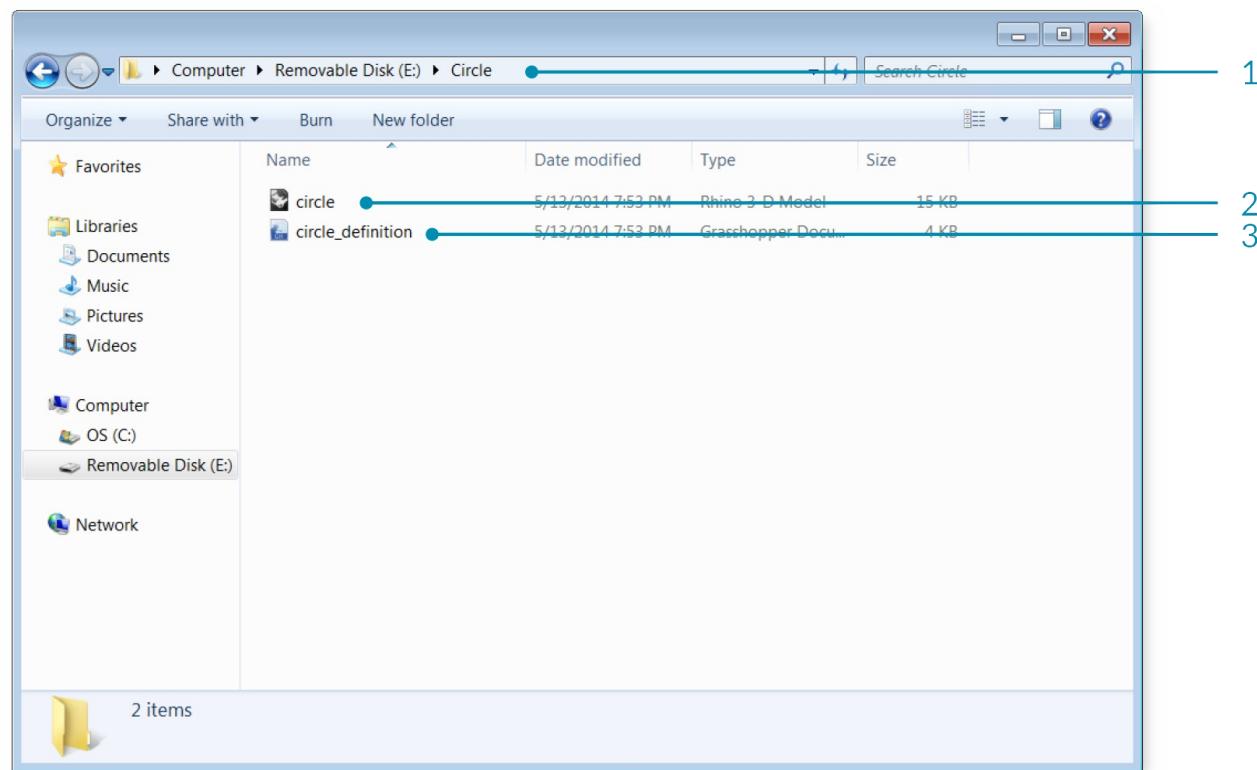
У RCP имеются два режима работы: *Edit Mode* (режим редактирования) (слева) - позволяет реорганизовать внешний вид RCP, *Working Mode* (рабочий режим) позволяет менять реальные значения элементов UI.



В Edit Mode у RCP оранжевый фон.

1.1.3.7. УПРАВЛЕНИЕ ФАЙЛАМИ

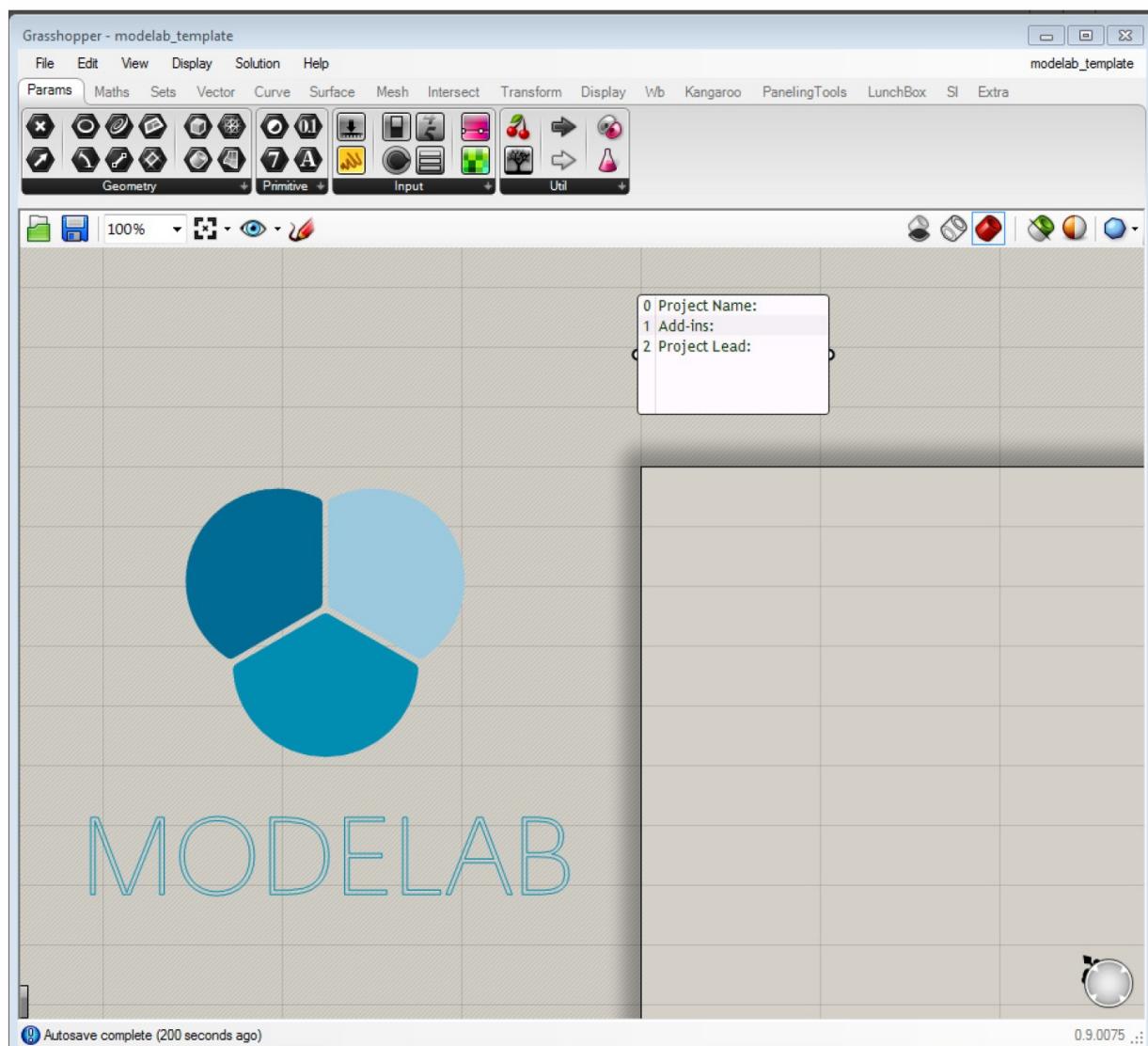
Если ваш файл Grasshopper берет начальную геометрию из Rhino, вам следует открыть тот же самый файл. Храните файлы Rhino и Grasshopper организованно в одной папке с одинаковыми именами файлов.



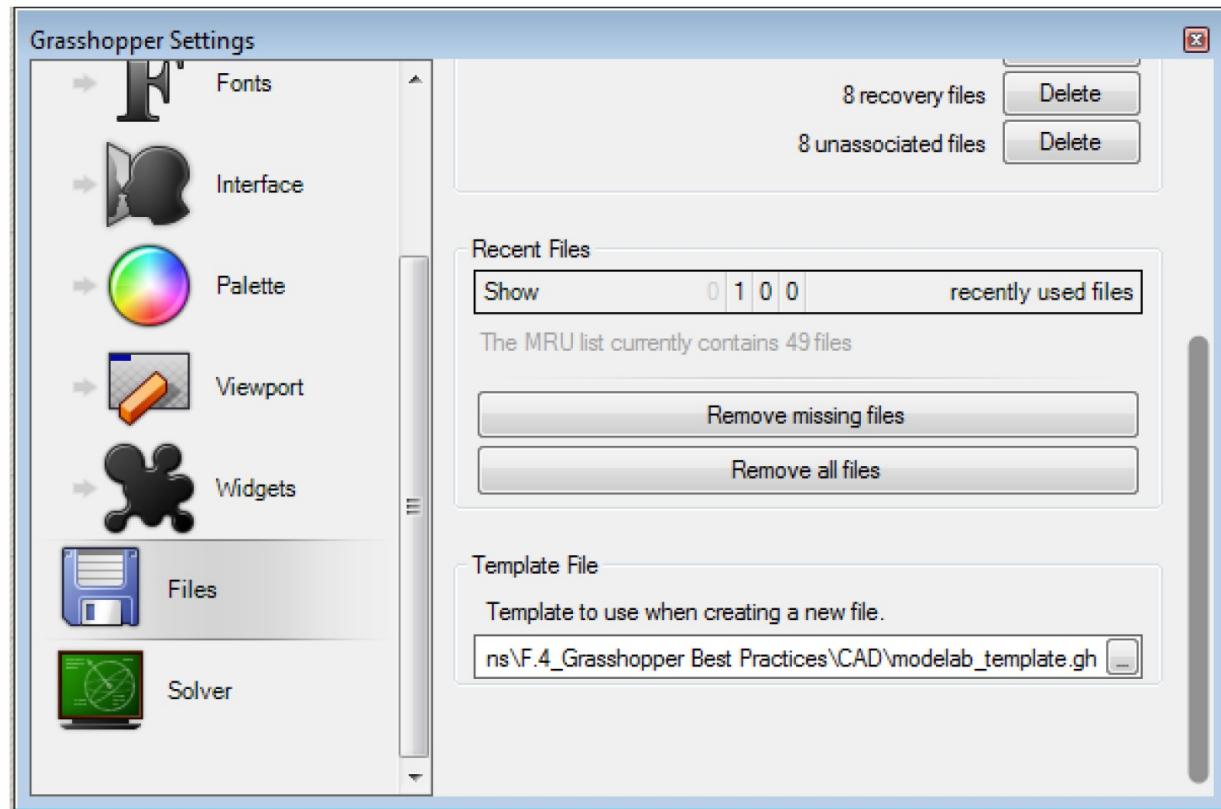
1. Папка проекта.
2. Файл Rhino.
3. Файл Grasshopper.

1.1.3.8. ШАБЛОНЫ

Создавая и устанавливая файл шаблон в настройках, Grasshopper создает удобный способ создания каждого нового определения Grasshopper. Шаблон может включать компоненты Grasshopper также как панели и скетч-объекты для обозначения.



Создайте шаблон файла и сохраните его



1. В File/Preferences, загрузите файл, который вы создали в шаблонах. Ваш шаблон будет теперь использоваться каждый раз при создании нового файла.

1.2. СТРУКТУРА ОПРЕДЕЛЕНИЯ GRASSHOPPER

Grasshopper дает возможность создавать визуальные программы, которые называются определениями. Эти определения состоят из нодов, объединенных связями. В следующей главе рассказывается об объектах Grasshopper и как с ним взаимодействовать, чтобы начать создавать определение.



1.2.1. ТИПЫ ОБЪЕКТОВ GRASSHOPPER

Grasshopper состоит из двух основных типов пользовательских объектов: параметров и компонентов. Параметры хранят данные, а компоненты выполняют действия, которые превращаются в данные. Самый основной способ понять Grasshopper - это помнить о том, что мы будем использовать данные для определения вводных параметров действий (что в итоге превратится в новые данные, которые мы сможем продолжить использовать).

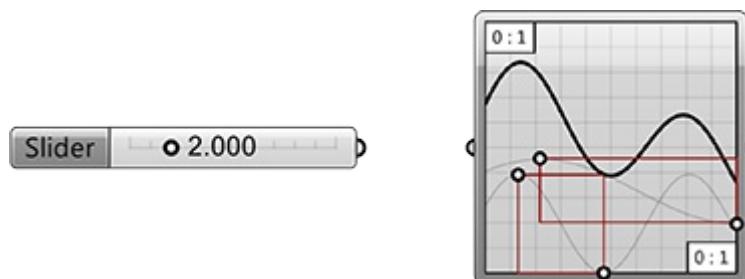
1.2.1.1. ПАРАМЕТРЫ

Параметры хранят данные - числа, цвета, геометрию и тд - которые мы передаем посредством схемы в нашем определении. Параметры - это такие контейнеры, которые обычно выглядят как небольшие прямоугольники с одним входом и одним выходом. Мы также понимаем, что это параметры, исходя из формы их иконки. У всех параметров вокруг их иконки есть шестиугольная рамка.

Геометрические параметры могут ссылаться на геометрию из Rhino или наследовать геометрию от других компонентов. Точка и кривая - это геометрические параметры.

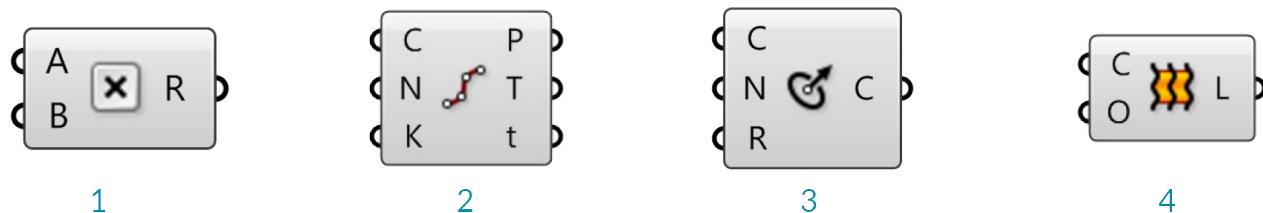


Параметры ввода - это динамические объекты интерфейса, которые позволяют вам взаимодействовать с вашим определением. Number slider (числовой слайдер) и graph mapper оба являются параметрами ввода.



1.2.1.2. КОМПОНЕНТЫ

Компоненты выполняют действия, основываясь на вводных параметрах, которые они получили. Имеется много типов компонентов для различных задач.



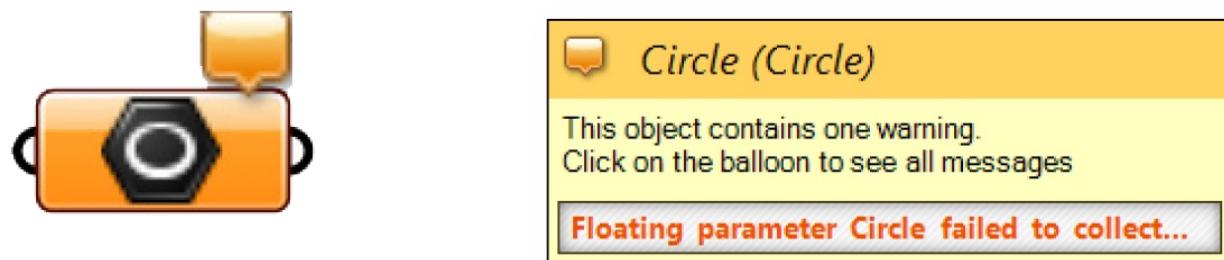
1. Компонент multiplication- это оператор, который просчитывает произведение двух чисел.
2. Компонент Divide работает с геометрией, разделяя кривую на равные сегменты.
3. Компонент Circle CNR создает геометрию круга из вводных данных: точка начала координат, вектор нормали и радиус.
4. Компонент Loft создает поверхность посредством кривых.

1.2.1.3. ЦВЕТА ОБЪЕКТОВ

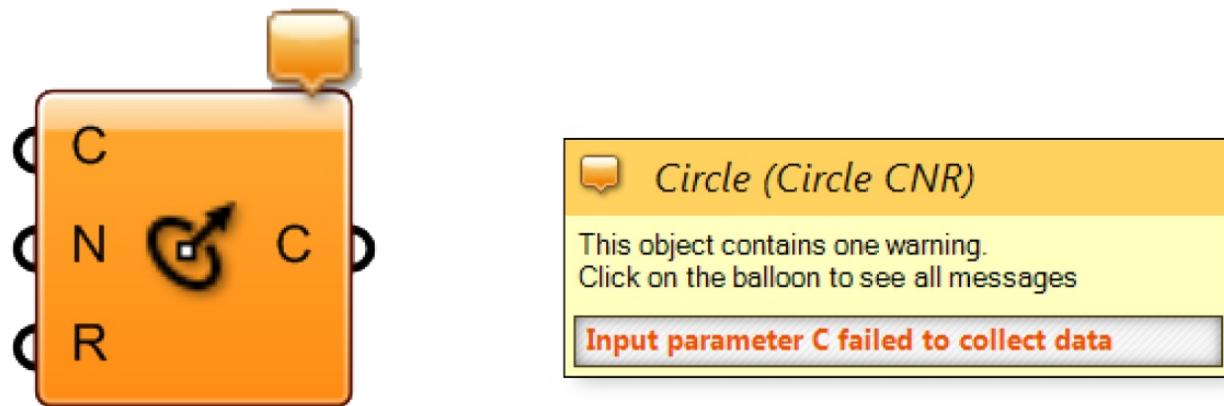
Мы можем почерпнуть информацию о состоянии каждого объекта, основываясь на их цветах. Давайте посмотрим на цветовую кодовую систему Grasshopper по умолчанию.

Параметр, который не содержит ни предупреждений ни ошибок, светло-серого цвета. Этот цвет объекта показывает, что этот параметр работает должным образом.

Параметр, который содержит предупреждение, отображается оранжевым цветом. Любой объект, который не смог собрать данные, считается подозрительным в определении Grasshopper из-за того, что он не работает на достижение решения. Таким образом, все параметры (недавно добавленные) - оранжевого цвета, чтобы отобразить, что они не содержат каких-либо данных и, поэтому, не несут функционального влияния на решение. По умолчанию, параметры и компоненты оранжевого цвета также имеют маленький шарик в верхнем правом углу объекта. Если вы наведете курсор мышки на этот шарик, он отобразит информацию о том, почему этот компонент содержит предупреждение. Как только параметр унаследует или приобретет данные, он станет серого цвета и шарик сверху исчезнет.



Компонент - это всегда более сложный объект из-за того, что нам приходится разбираться и, затем, координировать каковы его входы и выходы. Как и параметры, компонент с предупреждением отображается оранжевым цветом. Запомните, предупреждение- это не обязательно что-то плохое. Просто Grasshopper привлекает ваше внимание к потенциальной проблеме в вашем определении.



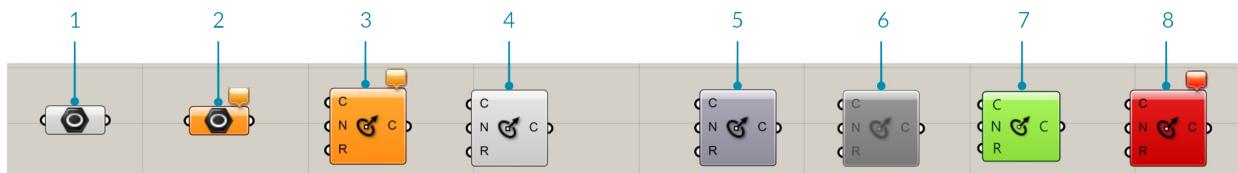
Параметр, который не содержит ни предупреждений ни ошибок, светло-серого цвета.

Компонент, просмотр которого был отключен, имеет темно-серый цвет. Существует два способа отключить просмотр компонента. Первый - просто нажмите правой клавишей мыши на компоненте и переключите кнопку просмотра. Чтобы одновременно отключить просмотр более чем одного компонента, сначала выберите требуемые компоненты и затем переключите иконку просмотра (человек с завязанными глазами) кликнув правой клавишей мыши в любом месте холста.

Компонент, который был отключен, имеет тусклый-серый цвет. Чтобы отключить компонент, нажмите правой клавишей мыши на компоненте и переключите кнопку отключения. Также вы можете выбрать требуемые компоненты, кликнув правой клавишей мыши в любом месте на холсте и выбрать Disable. Отключенные компоненты прекращают отправлять данные компонентам, следующим за ними.

Выбранный компонент будет иметь светло-зеленый цвет. Если выбранный компонент сгенерировал какую-либо геометрию с использованием среды Rhino, то он также будет зеленого цвета.

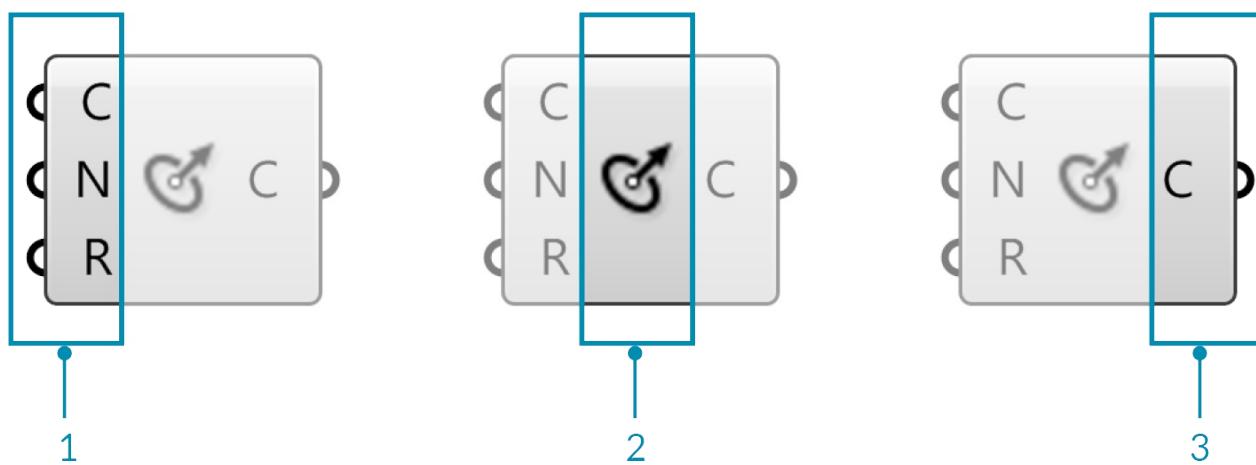
Компонент, содержащий хотя бы 1 ошибку, будет красного цвета. Ошибка может произойти из-за самого компонента или из-за одного из его входов и выходов.



1. Параметр без предупреждений или ошибок
2. Параметр с предупреждением
3. Компонент с предупреждением
4. Компонент без предупреждений или ошибок
5. Компонент с отключенным просмотром
6. Отключенный компонент
7. Выбранный компонент
8. Компонент с ошибкой

1.2.2. ЧАСТИ КОМПОНЕНТА GRASSHOPPER

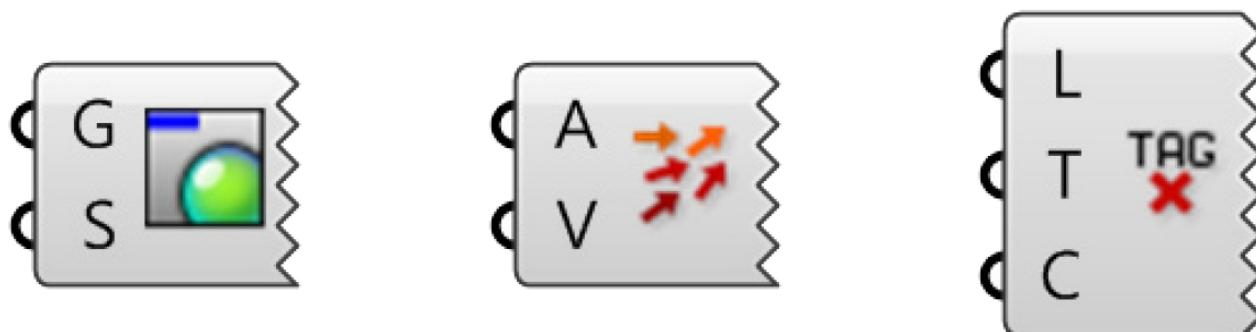
Компоненты - это объекты, которые размещают на холсте и соединяют друг с другом связями, чтобы создать визуальную программу. Компоненты могут представлять Rhino геометрию или такие операции как Math Functions. У компонентов имеются входы и выходы.



1. Три параметра ввода компонента Circle CNR.
2. Область компонента Circle CNR.
3. Параметр выхода компонента Circle CNR.

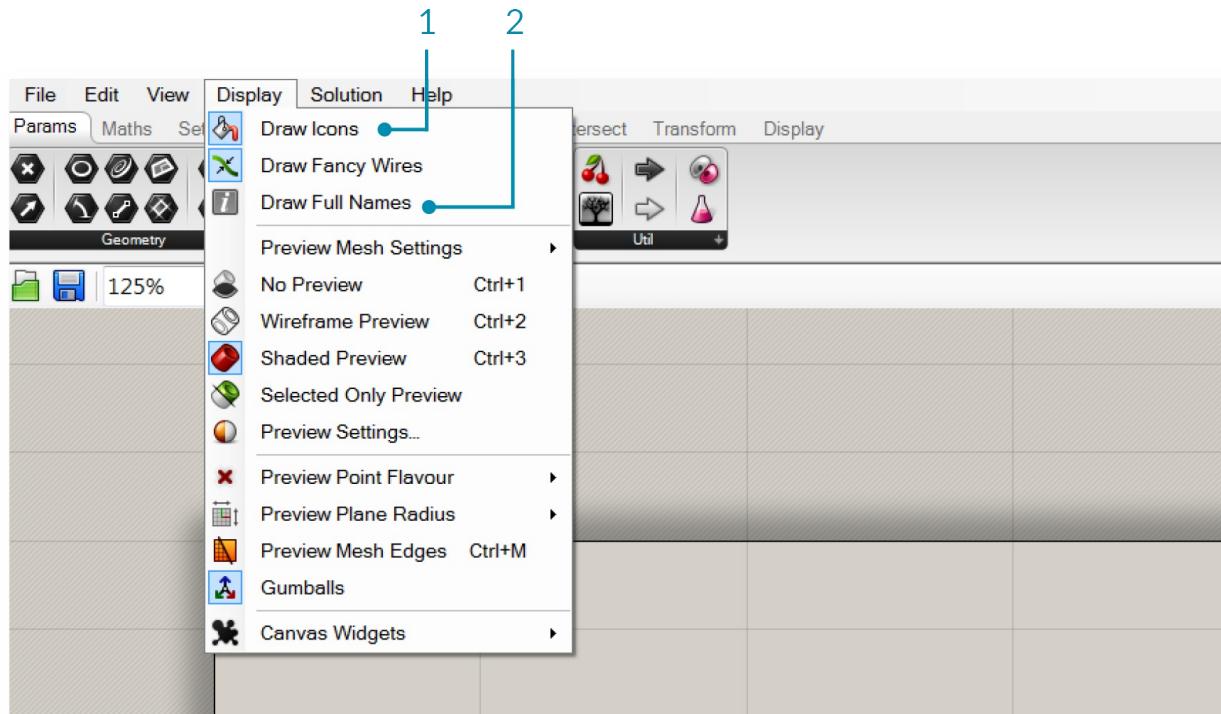
Компоненту требуются данные, чтобы выполнить действие и достигнуть результата. Поэтому, большинство компонентов имеют ряд встроенных параметров, которые обозначаются как Вход и Выход, соответственно. Входные параметры расположены слева, параметры выхода справа.

Ниже приведены несколько компонентов Grasshopper, которые имеют входы, но не имеют выходов, либо наоборот. Если у компонента нет входов или выходов, то у него будет зазубренный край.



1.2.2.1. ОТОБРАЖЕНИЕ ЛЕЙБЛОВ И ИКОНОК

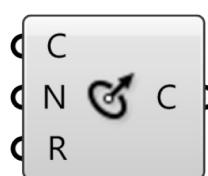
У каждого объекта Grasshopper уникальная иконка. Эти иконки располагаются в центральной области объекта и соответствуют иконкам, расположенным на Палитре Компонентов. Объекты также могут отображаться в виде текстовых ярлыков. Чтобы переключиться между отображением в виде иконки и ярлыка, выберите Draw Icons в меню Display. Вы также можете выбрать Draw Full Names, чтобы отобразить полное имя каждого объекта, а также его входы и выходы.



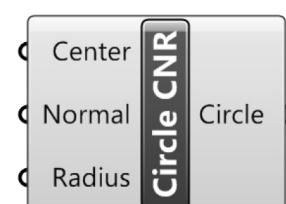
1. Переключение отображения в виде иконки и ярлыка.
2. Отобразить полное имя компонента и его входы и выходы.



1



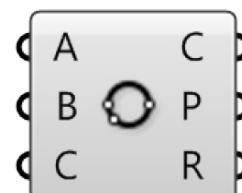
2



3

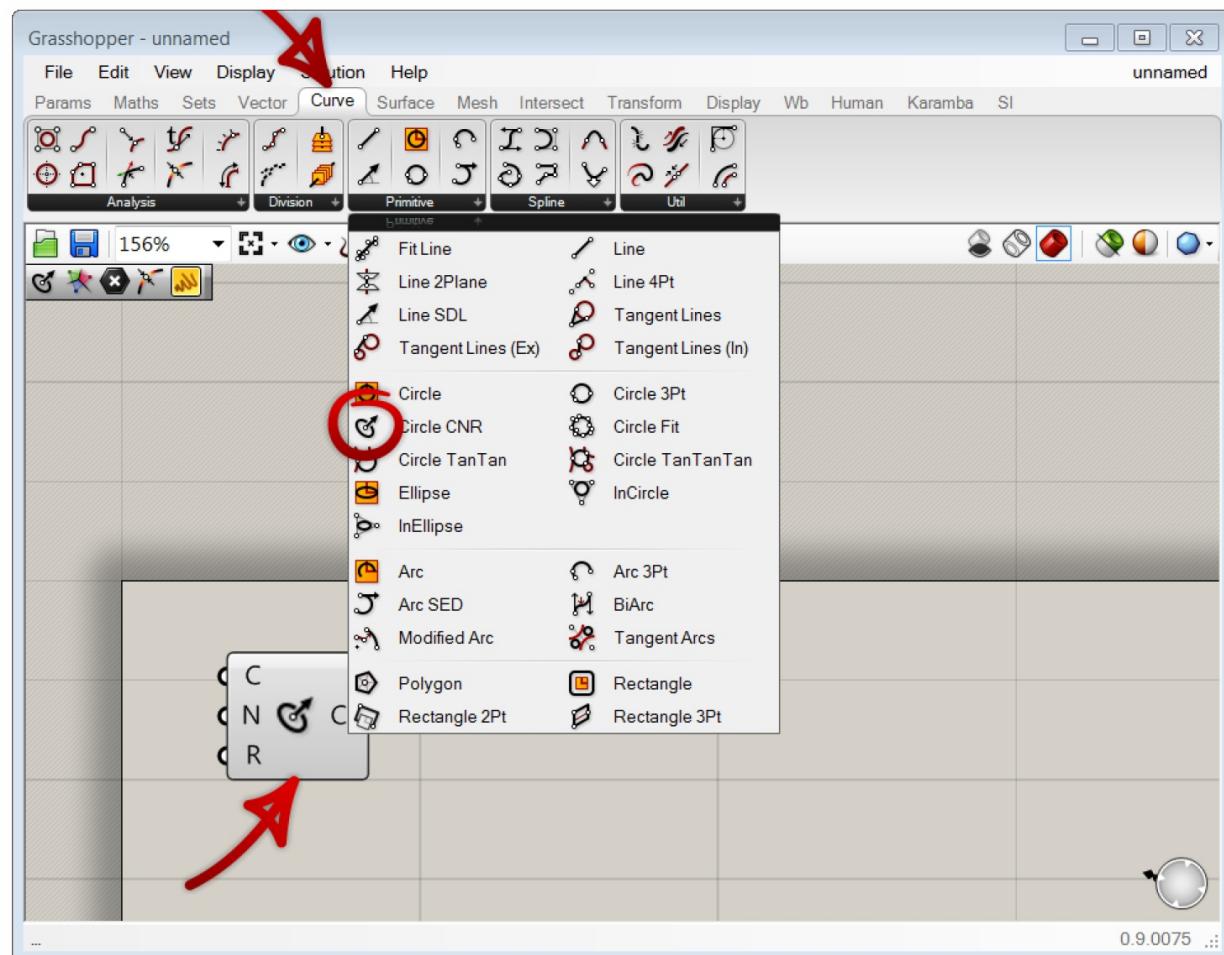
1. Компонент Circle CNR - отображение ярлыка
2. Компонент Circle CNR - отображение иконки
3. Компонент Circle CNR с отображением полных наименований

Мы рекомендуем использовать вид иконки, чтобы познакомиться с иконками компонентов, так чтобы вы могли быстро находить их на палитрах. Это также позволит вам с первого взгляда понимать определение. Текстовые ярлыки могут сбивать с толку из-за того, что различные компоненты могут иметь одинаковые ярлыки.



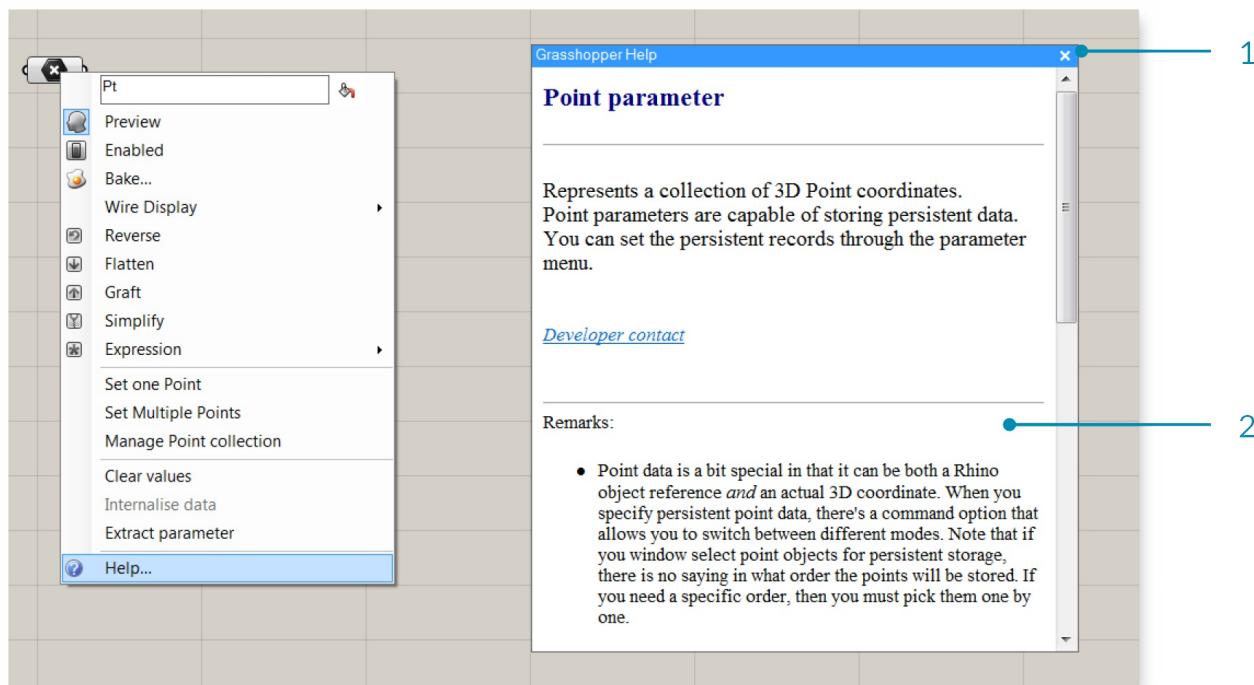
У Circle CNR и Circle 3pt одинаковые ярлыки, но различные иконки.

Чтобы обнаружить расположение компонентов на панелях зажмите **Ctrl + Alt** и кликните на существующий компонент на холсте. Так вы обнаружите компонент на палитре.



1.2.2.2. ПОМОЩЬ ПО КОМПОНЕНТАМ

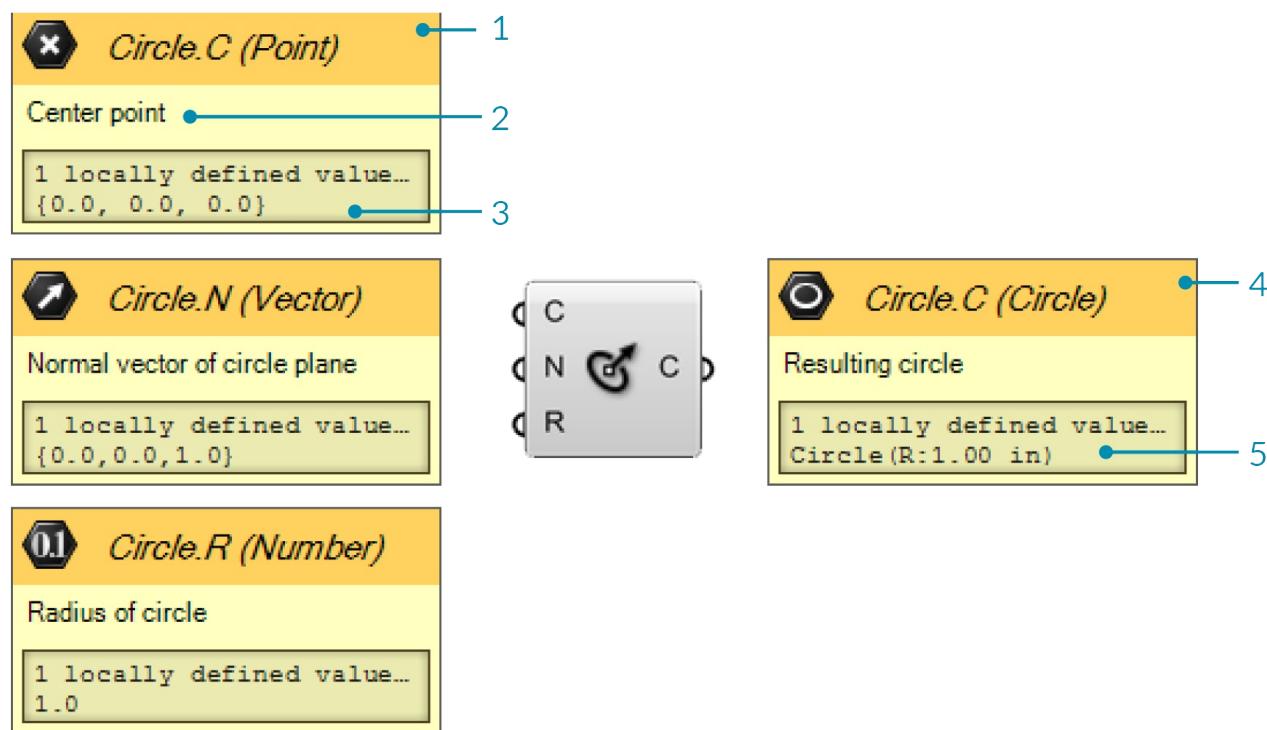
Кликните правой клавишей мыши по объекту и выберите "Help" из выпадающего меню, чтобы открыть справочное окно Grasshopper. Справочное окно содержит более подробное описание объекта, список вводов и выводов и замечания.



1. Справочное окно Grasshopper для параметра Point.
2. Замечания в справочном окне дают дополнительные сведения о параметре Point.

1.2.2.3. ПОДСКАЗКИ ПО ИНСТРУМЕНТАМ

Входы компонента ожидают получения определенного типа данных, например, компонент может показывать, что вы должны подсоединить точку или плоскость к его входу. При наведении мышки на отдельные части компонента, вы увидите разные подсказки, которые отображают определенный тип объекта, находящийся под мышкой. Подсказки довольно информативны, так как они сообщают тип и данные индивидуальных параметров.



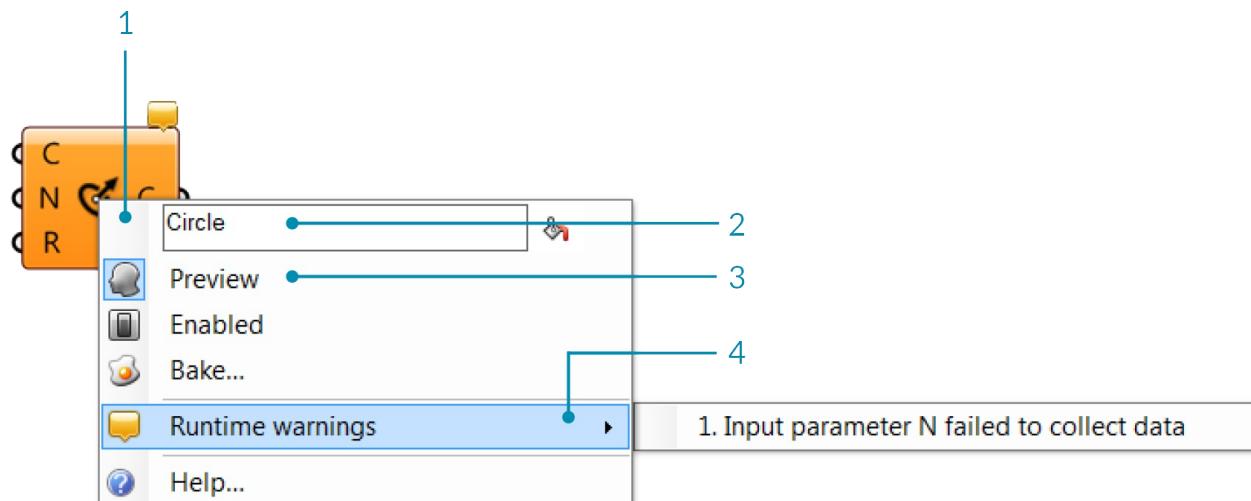
1. Заголовок подсказки показывает иконку типа входа, название компонента, ярлык для входа и тип

входа в текстовом виде.

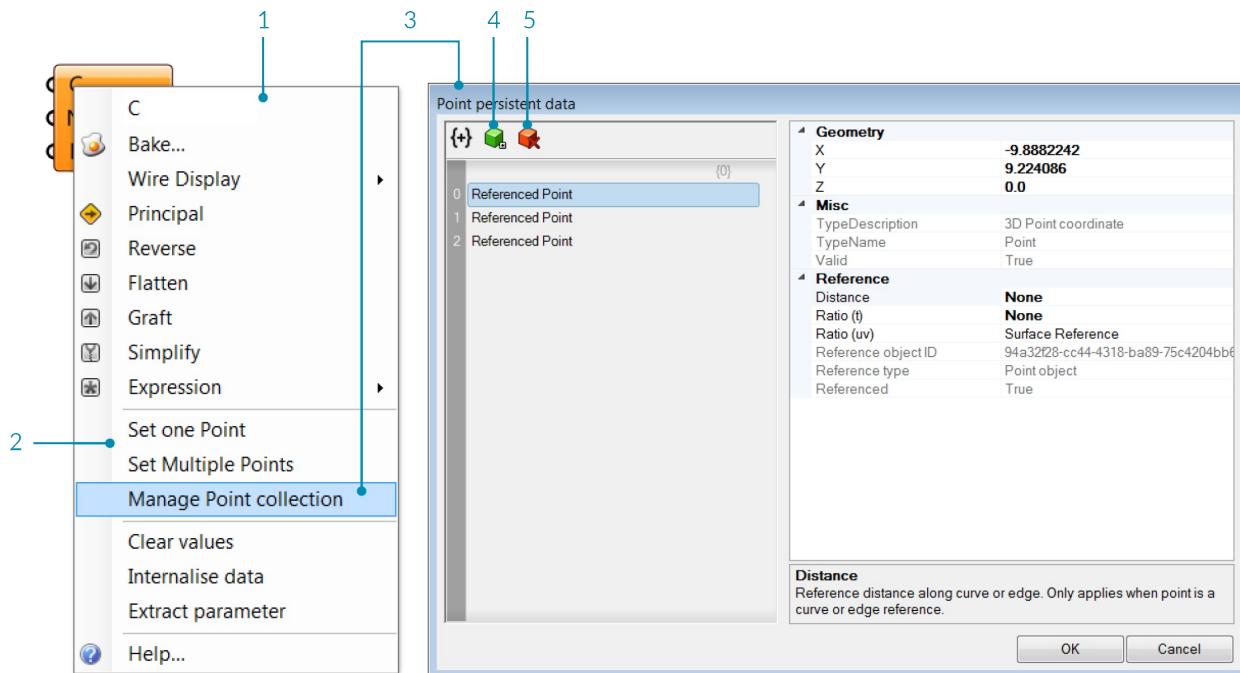
2. Описание простым языком того, какой вход для компонента.
3. Любые значения определенные для входа - либо локальные либо из подключенных связей.
4. Заголовок выхода подсказки по инструментам предоставляет те же самые данные как для входов, но для соответствующего выхода.
5. Результат действий компонента.

1.2.2.4. КОНТЕКСТНЫЕ ВЫПАДАЮЩИЕ МЕНЮ

Все объекты на холсте имеют свое контекстное меню, которое раскрывает их настройки и подробные данные. Кликните правой клавишей мыши по центру компонента, и вы вызовете контекстное меню. И вход, и выход имеют свои контекстные меню, которые можно посмотреть тем же способом.



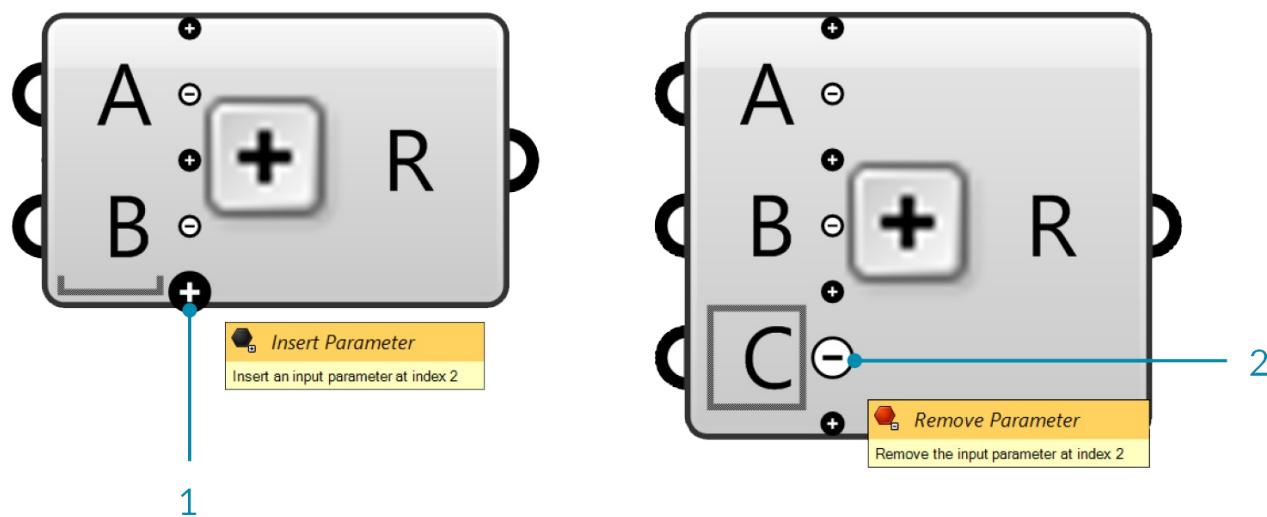
1. Контекстное меню компонента.
2. Редактируемое текстовое поле, которое отображает имя объекта.
3. Значок предпросмотра - сообщает о том, будет ли геометрия, произведенная этим объектом, видна в видовом окне Rhino. Отключение предпросмотра ускорит скорость работы видового окна Rhino и время, затраченное на решение.
4. Предупреждение о времени выполнения задания - представляет предупреждения о том, что то-то затрудняет работу компонента.



1. Контекстное меню входа С.
2. Установите одну или несколько точек - позволяет вам выбрать исходную геометрию в видовом окне Rhino.
3. Manage Point collection (управление набором точек) - открывает диалоговое окно, в котором вы можете добавлять или удалять точки из набора точек и просматривать информацию о каждой точке.
4. Добавить элемент в набор.
5. Удалить выбранное.

1.2.2.5. МАСШТАБИРУЕМЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

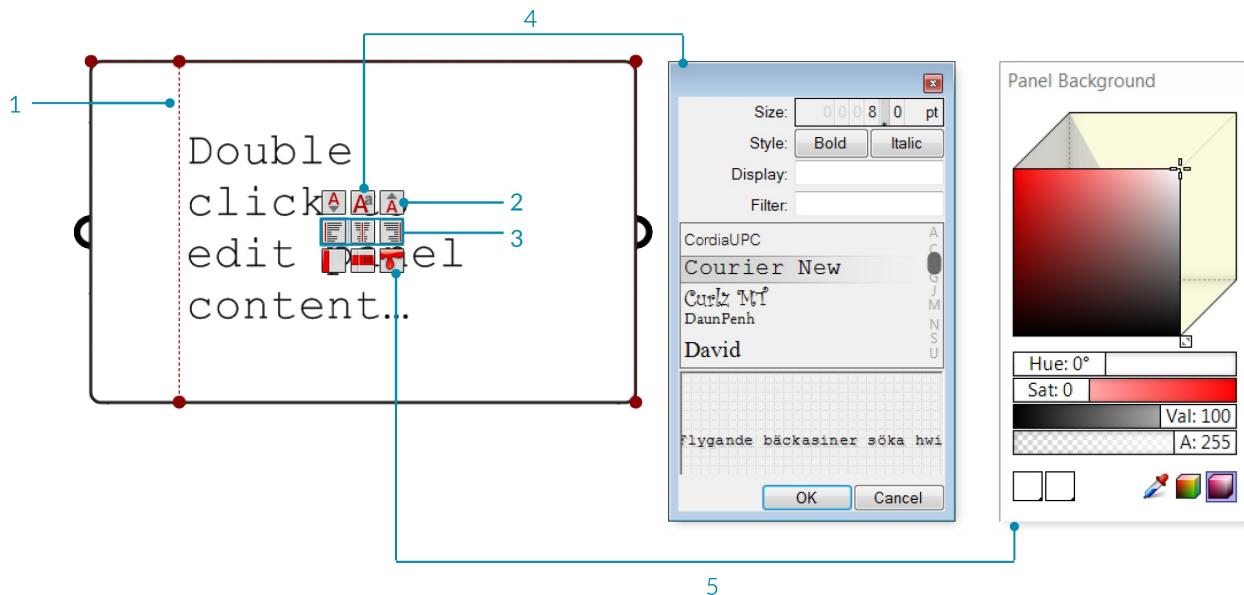
У некоторых компонентов может быть увеличено число входов или выходов посредством Масштабируемого Пользовательского Интерфейса (ZUI). Увеличивая компонент на холсте, появится дополнительный набор опций, которые позволяют вам добавить или удалить входы или выходы у данного компонента. Компонент Addition позволяет вам добавлять входы, предоставляя дополнительные элементы для дополнительных операций.



1. Кликнуть + знак, чтобы добавить Вход.

2. Кликнуть - знак, чтобы удалить Вход.

Компонент panel (панель) также имеет масштабируемый UI. Он представляет собой самоклеящийся стикер. Благодаря этому вы можете добавлять небольшие заметки или объяснения к определению. Вы можете менять текст через меню или двойным кликом на поверхности панели. Панели могут получать и отображать информацию, откуда угодно. Если вы подключите выход в панель, вы увидите содержание этого параметра в реальном времени. Все данные в Grasshopper можно просматривать таким образом. При приближении панели появится меню, в котором вы сможете поменять фон, шрифт и другие характеристики. Эти опции становятся также доступны, если кликнуть правой клавишей мыши по панели



1. Подвигайте ползунки, чтобы настроить поля панели.
2. Увеличить или уменьшить размер шрифта содержимого панели.
3. Измените выравнивание содержания панели.
4. Выберите размер содержания панели.
5. Выберите цвет фона панели. Вы можете установить новый цвет по умолчанию для ваших панелей, кликнув правой клавишей мышки и выбрав "Set Default Color".

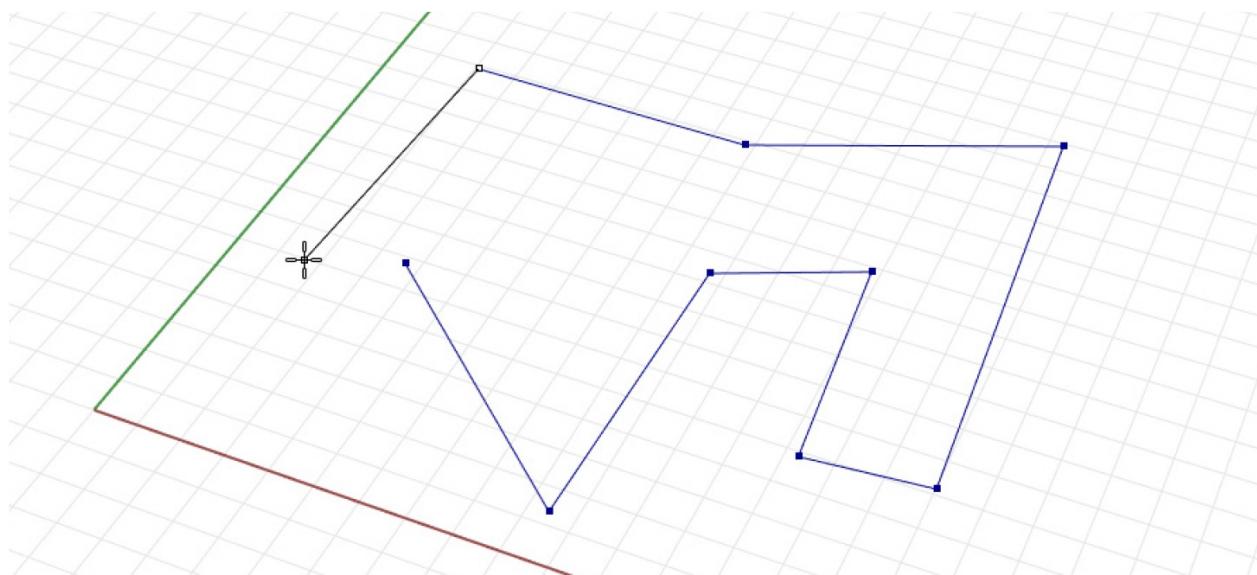
1.2.3. ТИПЫ ДАННЫХ

Большинство параметров хранят два различных типа данных: изменяемые и постоянные. Изменяемые данные наследуются из одного или более ресурсов и разрушаются (т.е. собираются заново) всякий раз, когда начинается новое решение. Постоянные данные - это данные, которые были специально введены пользователем.

1.2.3.1. ПОСТОЯННЫЕ ДАННЫЕ

Постоянныe данные доступны через меню и, в зависимости от типа параметра, имеют разных управляющих. Параметр Point, например, позволяет вам настроить одну или больше точек через меню. Но, давайте вернемся на несколько шагов назад и посмотрим, как ведет себя параметр Point.

Когда вы перетащили параметр Point из Params/Geometry Panel на холст, параметр оранжевого цвета, сигнализируя о наличии предупреждения. Это не очень страшно, предупреждение просто информирует вас о том, что параметр пустой (не содержит постоянных записей и не может собрать изменяемые данные) и, таким образом, не влияет на результат решения. Контекстное меню параметра предлагает два способа настройки постоянных данных: один и множество. Кликните правой клавиши мыши по параметру, чтобы установить Multiple Points. Когда вы кликнете по какому-либо из этих элементов меню, окно Grasshopper закроется и программа попросит вас выбрать точку в одном из видовых окон Rhino.



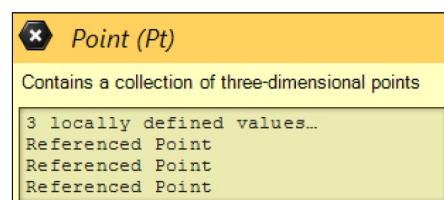
Как только вы выберете все точки, нажмите Enter и они станут частью постоянных данных параметра. Это означает, что параметр больше не пустой и он поменяет цвет с оранжевого на серый. (Заметьте, что информационный шарик в правом верхнем углу также исчезнет, так как предупреждения больше нет). В этот момент вы можете использовать точки, хранящиеся в этом параметре, для любого последующего входа в вашем значении.



1



2



3

1. Оранжевый параметр информирует о том, что параметр не содержит постоянных данных

(параметр не смог собрать изменяемые данные), таким образом, не влияет на результат решения. Кликните правой клавишей мыши по любому параметру, чтобы установить его постоянные данные.

2. Как только параметр содержит какие-либо постоянные данные, компонент из оранжевого превращается в серый.
3. Подсказка по инструменту для параметра Point отображает постоянные данные (набор исходных точек), которые он хранит.

1.2.3.2. ИЗМЕНЯЕМЫЕ ДАННЫЕ

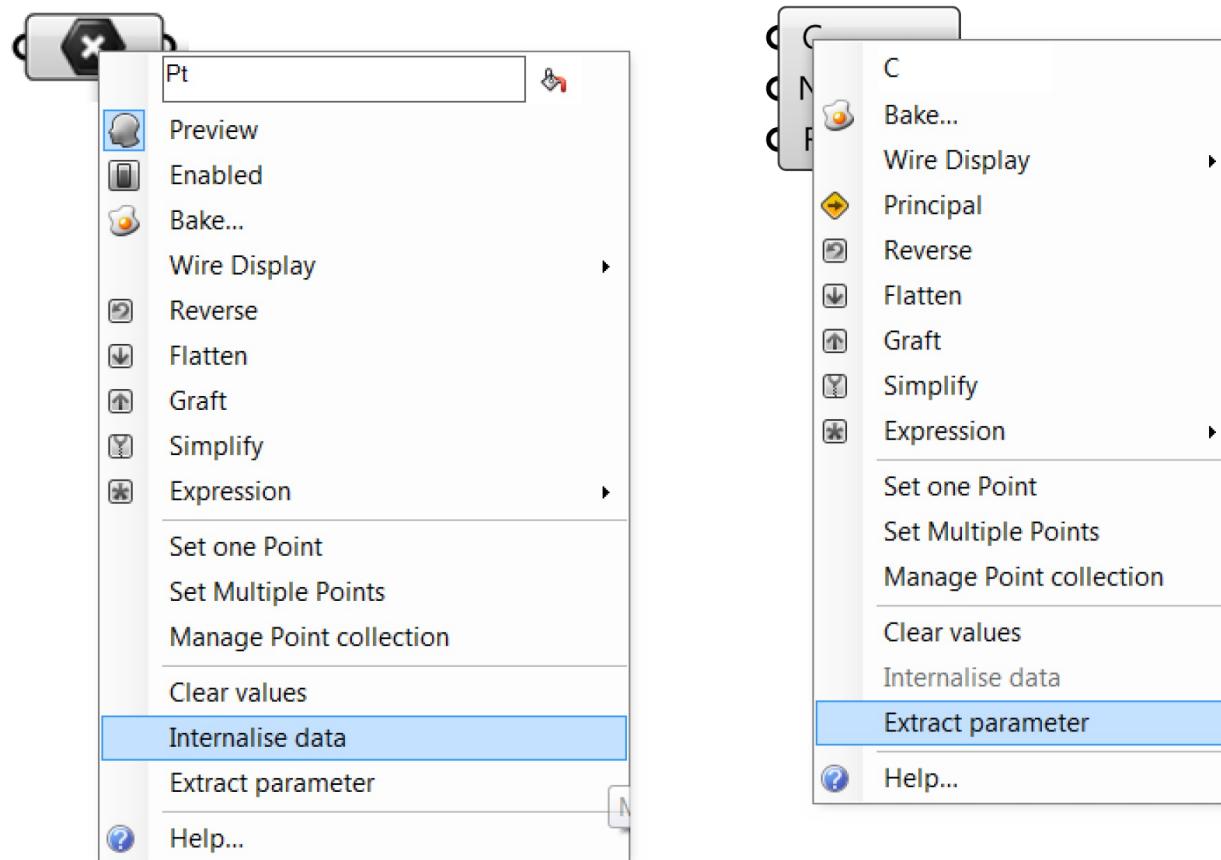
Изменяемые данные, как подразумевает название, не постоянные и будут уничтожаться каждый раз, когда срабатывает решение. Тем не менее, это будет также часто запускать событие для перестроения решения и обновления сцены. Говоря в целом, большая часть данных, генерированных "на лету", считается изменяемой.

Как упоминалось раньше, данные Grasshopper хранятся в разделе Параметры (либо в Изменяемой или Постоянной форме) и используются в различных Компонентах. Когда данные не хранятся в постоянном месте записи в Параметрах, появляется необходимость взять их откуда-то. Каждый Параметр (за исключением параметров выхода) сам определяет, откуда он получает свои данные, и большинство Параметров не очень точно определяют это. Вы можете подключить числовой Параметр (тот, который определяет, что это десятичное число) в целый источник и он возьмет на себя обязанность произвести конвертацию.

Вы можете изменить способ получения данных и способ хранения данных в контекстном меню параметра или входа компонента. Чтобы изменить хранящуюся исходную геометрию Rhino в самом определении Grasshopper, кликните правой клавишей мыши по параметру и выберите Internalise data из меню. Это полезно, если вы хотите, чтобы ваше определение Grasshopper было независимым от файла Rhino.

Вы также можете подключить Internalise data ко входу компонента. Как только вы выберете Internalise data в меню, все связи отсоединятся от этого входа. Данные изменились с изменяемых на постоянные и больше не будут обновляться.

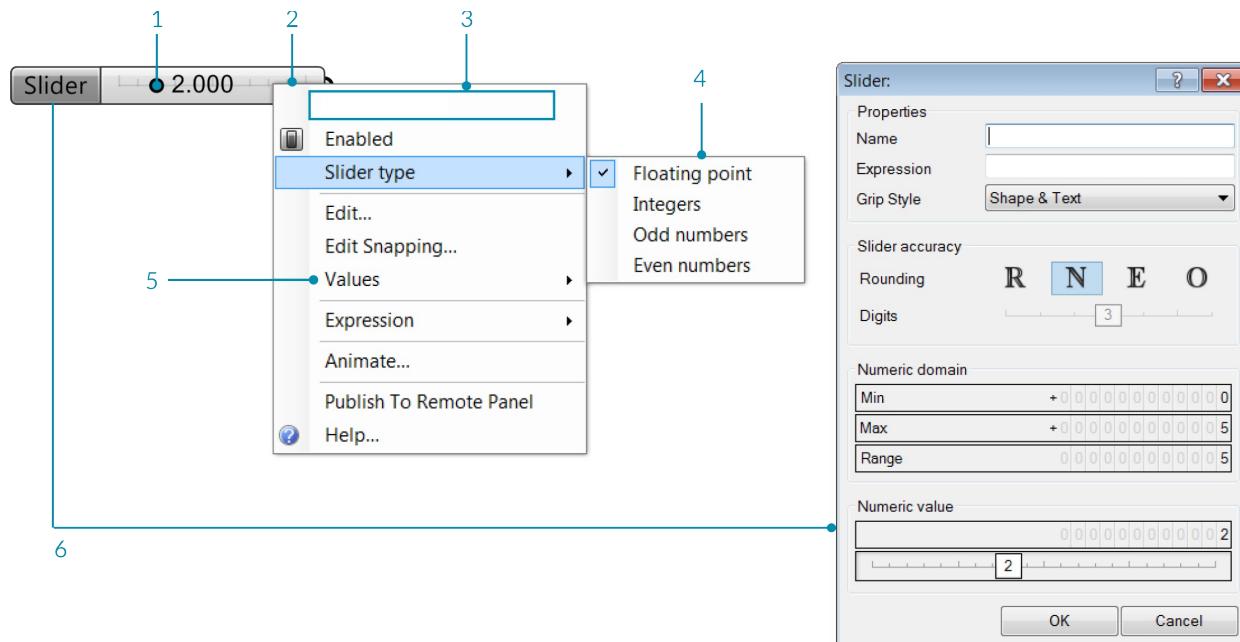
Если вы хотите, чтобы данные снова превратились в изменяемые, просто отсоедините связи от входа и значения автоматически заменятся. Вы также можете кликнуть правой клавишей мыши по входу и выбрать параметр Extract (извлекать). Grasshopper создаст параметр, подключенный связью с входом, которая содержит данные.



1.2.3.3. ВВОДНЫЕ ПАРАМЕТРЫ

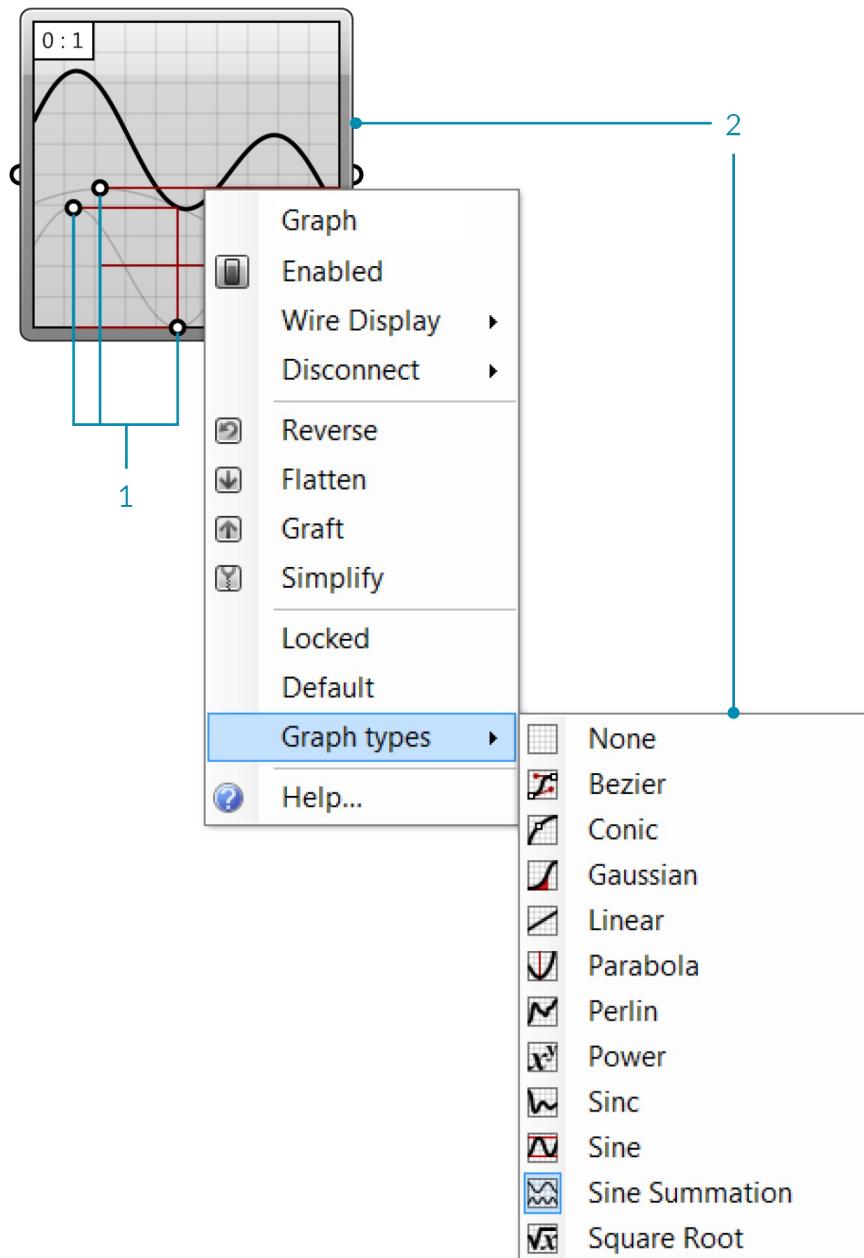
У Grasshopper есть разнообразные параметры, которые дают возможность взаимодействовать с данными, которые поставляются во входы компонента и, таким образом, контролируют изменение результата вашего определения. Из-за того, что параметры меняются с входом, они генерируют изменяемые данные.

Number Slider Цифровой слайдер - это самый важный и широко используемые параметр входа. Он позволяет нам выводить значения между двумя данными экстремумами, путем взаимодействия с его ползунками при помощи нашей мышки. Слайдеры могут использоваться, чтобы указывать значение и наблюдать за изменением нашего определения, которые происходят при перемещении ползунка, но слайдер также должен рассматриваться как средство определения успешного диапазона нашего определения.

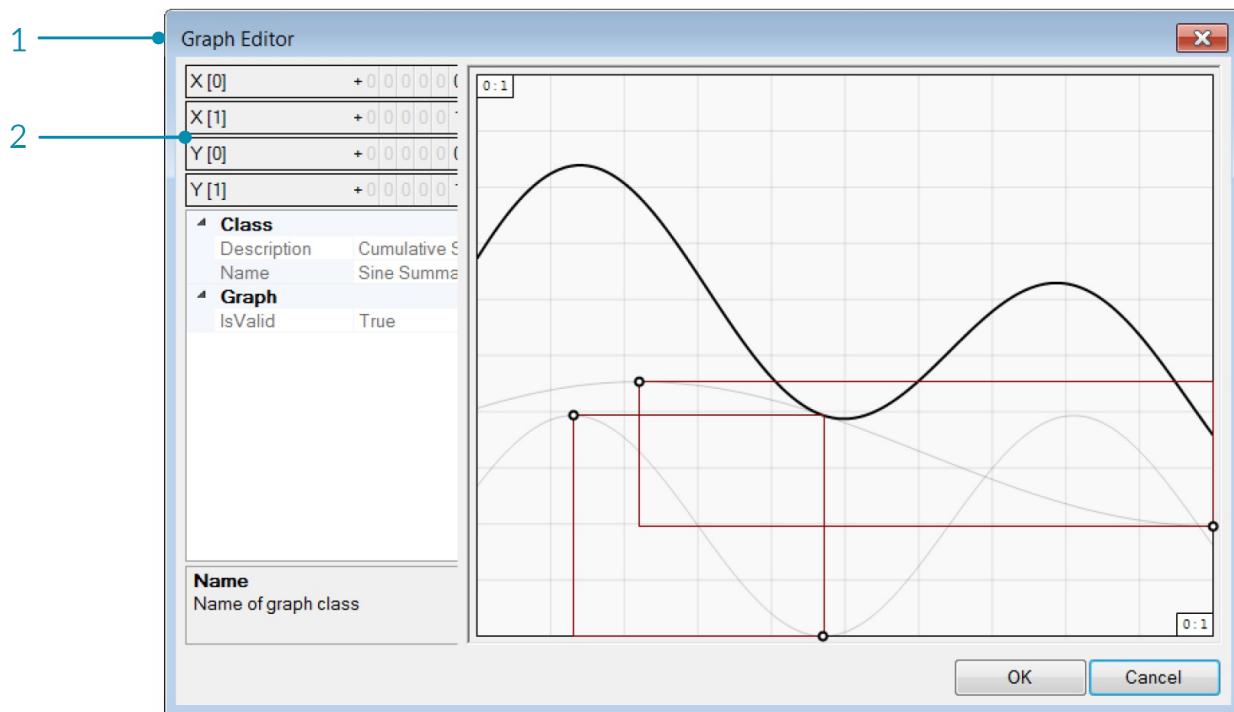


1. Перемещайте ползунок слайдера, чтобы поменять значение. При каждом перемещении ползунка, Grasshopper будет пересчитывать решение.
2. Кликните правой клавишей мыши по слайдеру, чтобы изменить имя, тип и значение.
3. Редактируемое текстовое поле, которое отображает имя слайдера.
4. Выберите тип числа для использования слайдером.
5. Изменить диапазон значений.
6. Дважды кликните по имени слайдера, чтобы открыть Slider Editor (редактор слайдера).

Graph mapper Graph Mapper - это двух-пространственный интерфейс, с помощью которого вы можете изменять числовые значения, выстраивая вход вдоль оси X и выводя соответствующие значения вдоль оси Y на пересечении с осью X на графике. Это очень полезно при варьировании набора значений внутри установленного интерфейса с ползунками.

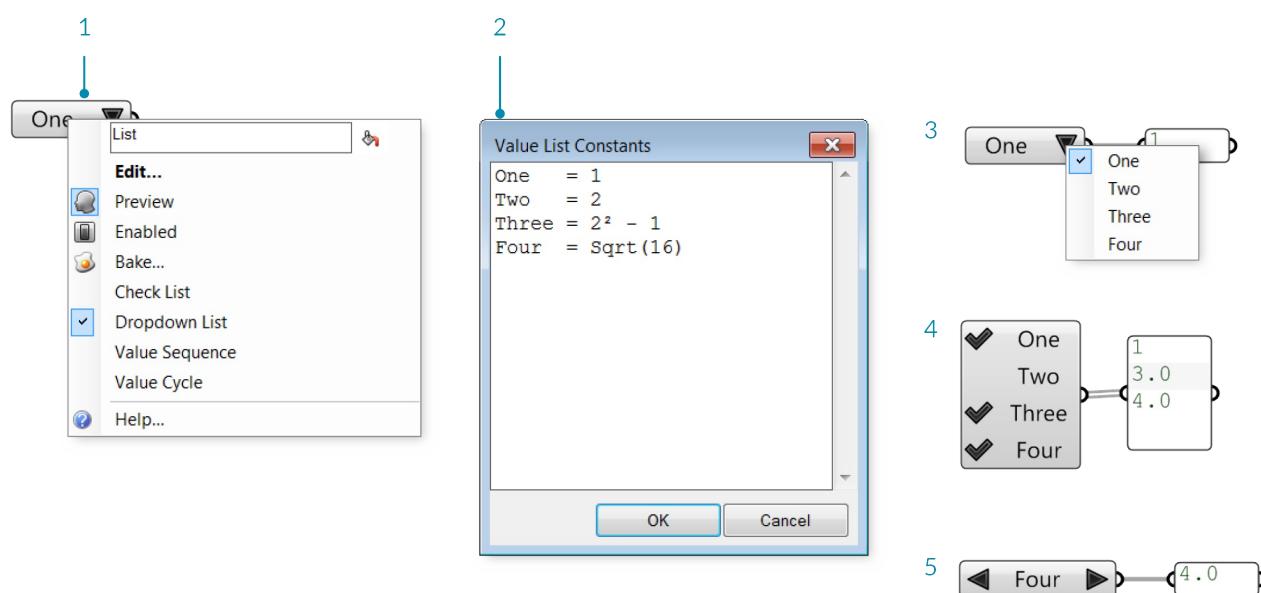


1. Перемещайте ползунки, чтобы отредактировать график. При каждом перемещении ползунка, Grasshopper будет пересчитывать решение.
2. Кликните правой клавишей мыши по компоненту graph mapper, чтобы выбрать тип графика.



1. Дважды кликните по graph mapper, чтобы открыть Graph Editor (редактор графика).
2. Измените диапазоны х и у.

Value List Value List (список значений) хранит набор значений с соответствующим списком ярлыков, связанных при помощи знака равенства. Это особенно полезно, когда вы хотите иметь несколько опций, названных по значению, которые могут предоставить специфические значения на выходе.



1. Кликните правой клавишей мыши по компоненту Value List и выберите опцию из меню.
2. Дважды кликните по компоненту Value List, чтобы открыть редактор и добавить или изменить значения.
3. В режиме Dropdown List (Выпадающий Список) кликните по стрелке, чтобы выбрать одно из значений. Решение будет вычисляться каждый раз, когда вы меняете значение.
4. В режиме Check List (Проверочный список) кликните рядом с каждым значением, чтобы проверить его. Компонент выдаст все значения, которые были проверены.
5. В режимах Value Sequence (Последовательность значений) и Value Cycle (Цикл значений),

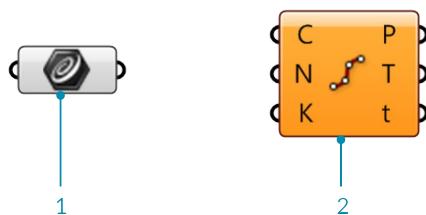
кликните по левой или правой стрелочке, чтобы пролистать значения.

1.2.4. КОМПОНЕНТЫ СВЯЗИ

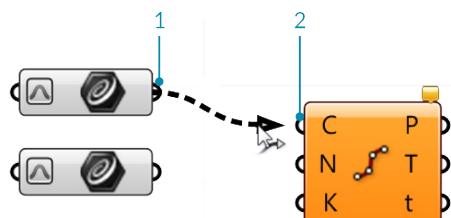
Когда данные не хранятся в постоянном месте записи в Параметрах, появляется необходимость взять их откуда-то. Данные передаются из одного компонента в другой посредством связей. Можете представить их себе, буквально, электрическими цепями, которые переносят сигналы данных из одного объекта в следующий объект.

1.2.4.1. УПРАВЛЕНИЕ СВЯЗЯМИ

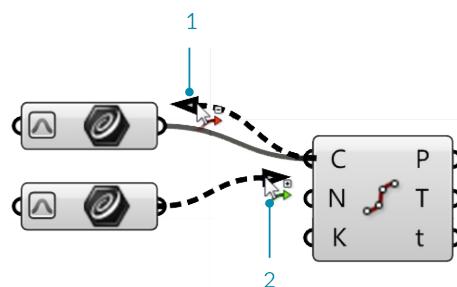
Для того, чтобы соединить компоненты, кликните и перетащите рядом с кругом на сторону выхода объекта. Соединительная связь прикрепится к мышке. Как только мышка будет наведена над потенциальным входом, связь подсоединится и станет прочной. Это не постоянная связь, пока вы не отпустите кнопку мыши. Не имеет значения, как мы осуществим подключения - слева-направо или справа-налево.



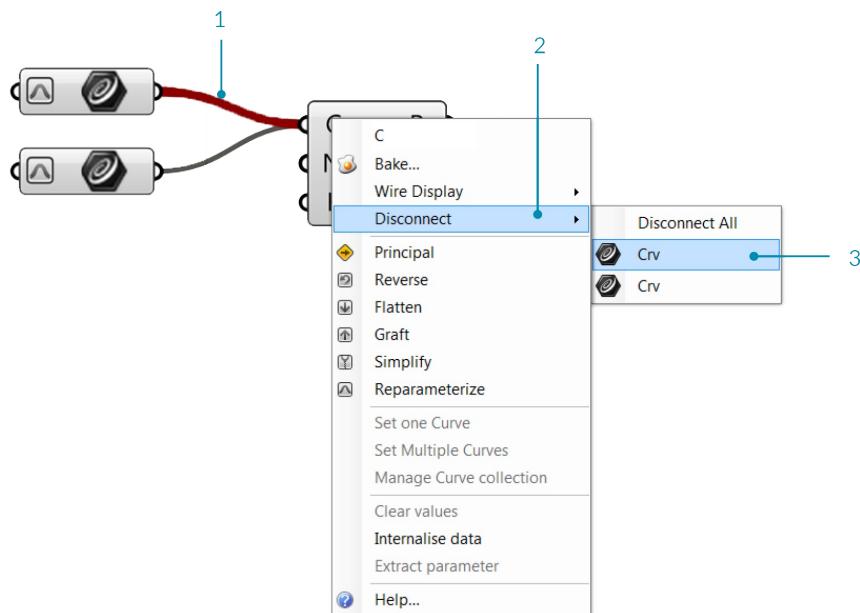
1. Компонент Divide Curve - разделяет кривую на равные по длине сегменты.
2. Параметр Curve (кривая) - кликните правой клавишей мыши и выберите Set One Curve (установить одну кривую), чтобы привязаться к геометрии Rhino.



Кликните левой клавишей мыши и перетащите связь из выхода (1) одного объекта в выход (2) другого объекта.



1. Если вы зажмете CONTROL, курсор станет красного цвета и целевой источник удалится из списка ресурсов.
2. По умолчанию, новое соединение удалит существующее соединение. Зажмите SHIFT во время перемещения соединений, чтобы выбрать множество источников. Курсор станет зеленого цвета, т.о. отображая дополнительное поведение.

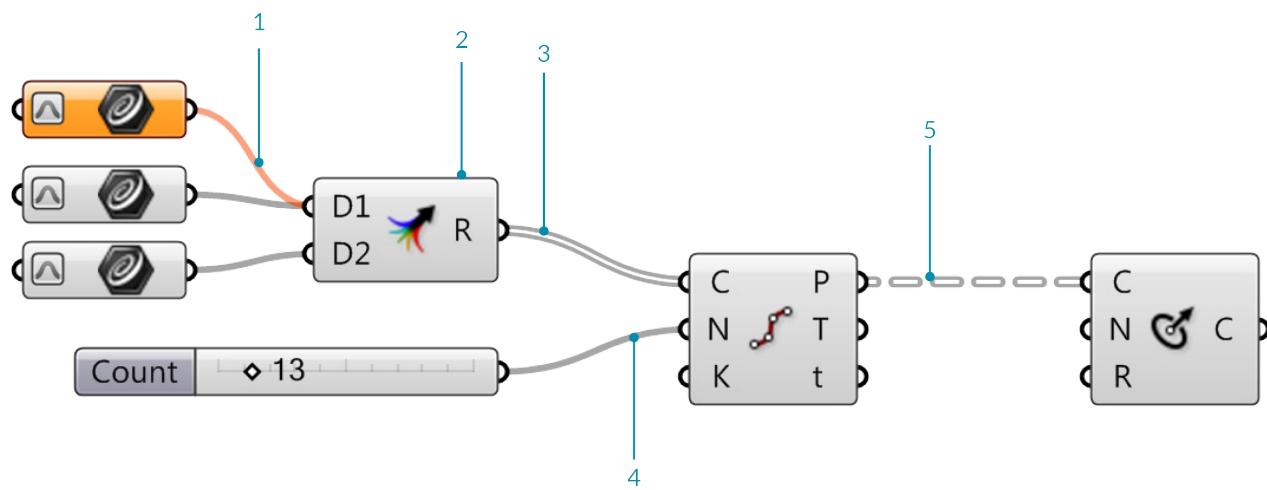


1. Вы также можете разъединить связь через контекстное выпадающее меню - правый клик по ползунку входа или выхода и выберите разъединить.
2. При наличии нескольких соединений, выберите одно, которое вы хотите отсоединить от списка.
3. При наведении мышки на элемент, связь подсветиться красным.

1.2.4.2. FANCY СВЯЗИ

Связи представляют соединения так же как поток данных внутри графика в нашем определении.

Grasshopper может также давать нам визуальные подсказки о том, что протекает через связи. Настройки по умолчанию, так называемые, "fancy wires" отключены, поэтому вам необходимо включить их до того как вы сможете просматривать различные типы линий для соединительных связей. Чтобы выполнить это, просто кликните на Display Tab в Строчке основного меню и выберите кнопку "Draw Fancy Wires." Fancy wires могут поведать много информации о том, какой тип информации протекает из одного компонента в другой.



1. Empty Item - тип связи оранжевого цвета, сигнализирует о том, что никакой информации не передается. Этот параметр генерирует предупреждающее сообщение, потому что оно не содержит данных и, поэтому, никакой информации не транслируется по связям.
2. Компонент Merge - это альтернатива подключению больше, чем одного источника к одному входу.
3. List - если информация, исходящая из компонента, содержит список информации, тип связи будет отображаться двойной серой линией.
4. Single Item - поток данных из любого параметра, который содержит один элемент, будет показан

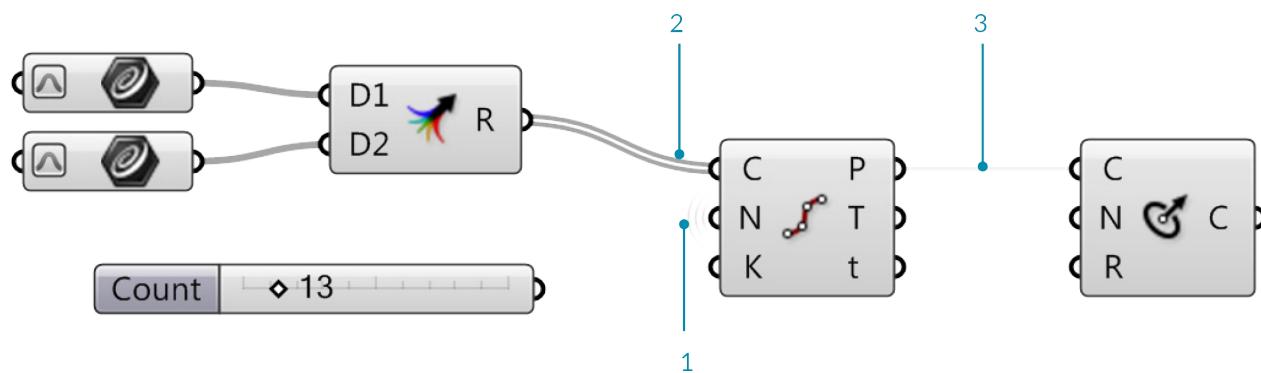
цельной серой линией.

- Tree (дерево) - информация, передающаяся между компонентами, которые содержат структуры данных, отображается связью серой двойной линией с точками.

1.2.4.3. ОТОБРАЖЕНИЕ СВЯЗЕЙ

Если вы уже провели достаточно много времени работая над одним определением Grasshopper, вы уже столкнулись с тем, что холст сильно и быстро захламляется "гнездами" связей. К счастью, у нас есть возможность управлять отображением связей для каждого входа компонента.

Существуют три вида отображения связей: Default Display (по умолчанию), Faint Display (исчезающее отображение), and Hidden Display (скрытое отображение). Чтобы изменить отображение связи, просто кликните правой клавишей мыши на любом входе компонента и выберите один из видов, доступных под Wire Display в выпадающем меню.



1. Hidden Display - когда выбран этот режим, связь будет абсолютно не видимой. Данные передаются "беспроводным" путем от источника к параметру входа. Если вы выберете исходный или конечный компонент, появится зеленая связь, которая показывает какие компоненты соединены с какими компонентами. При отмене выбора компонента, связь исчезнет.
2. Default Display (отображение по умолчанию) - отображение связи по умолчанию отобразит все соединения (при условии, что fancy wires отключены).
3. Faint Display - исчезающее отображение связей будет отображать соединение связей очень тонкими полупрозрачными линиями. Faint и Hidden отображение связей могут очень полезны, если в один вход входит несколько исходных связей.

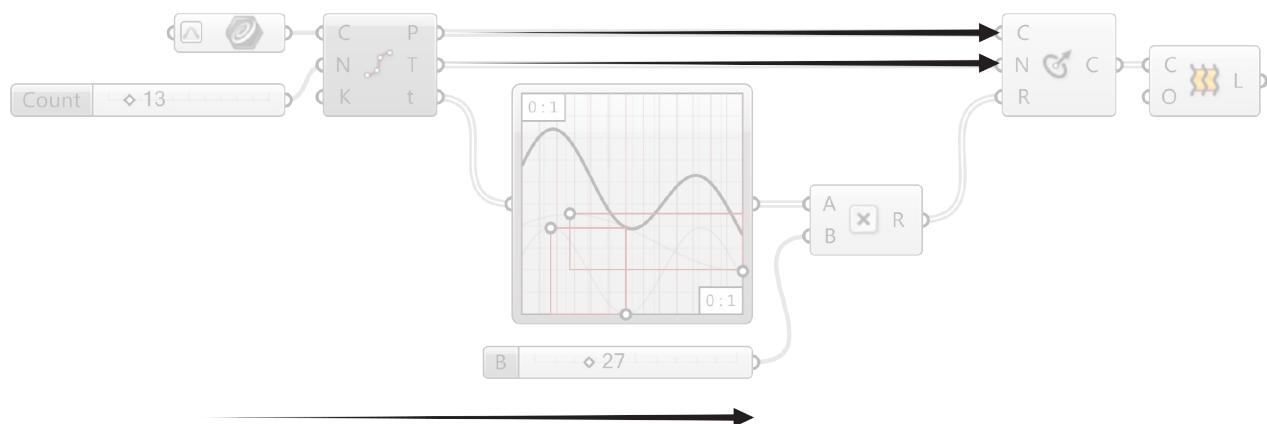
1.2.5. ОПРЕДЕЛЕНИЕ GRASSHOPPER

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

У определений Grasshopper есть процесс работы программы, который отображает, где начать выполнение программы, что делать в середине и как узнать, когда выполнение программы закончено.

1.2.5.1. ПРОЦЕСС РАБОТЫ ПРОГРАММЫ

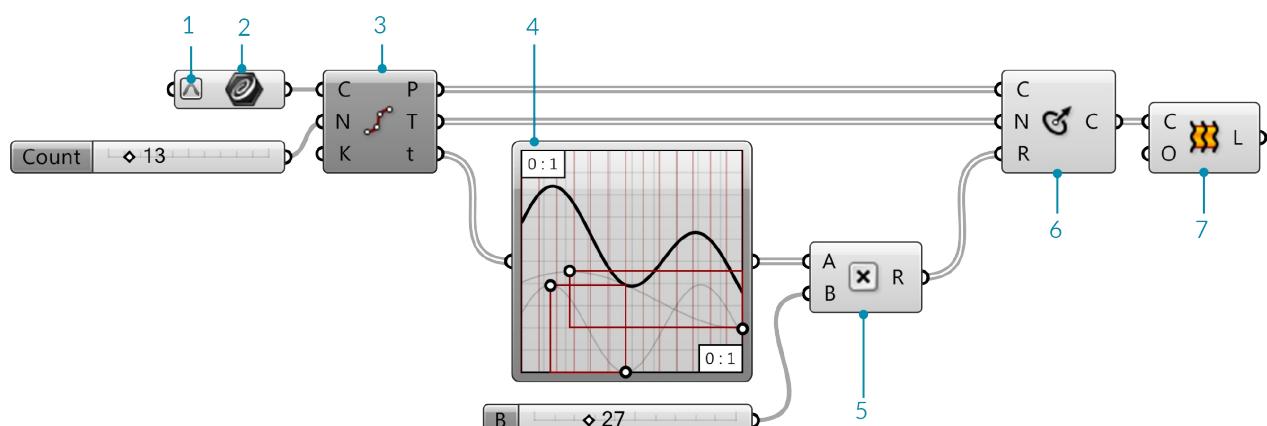
Визуальные программы Grasshopper выполняются слева направо. Чтение графика, связанного с соединениями связей из upstream с downstream, предоставляет понимание о процессе работы программы.



Направление данных слева направо.

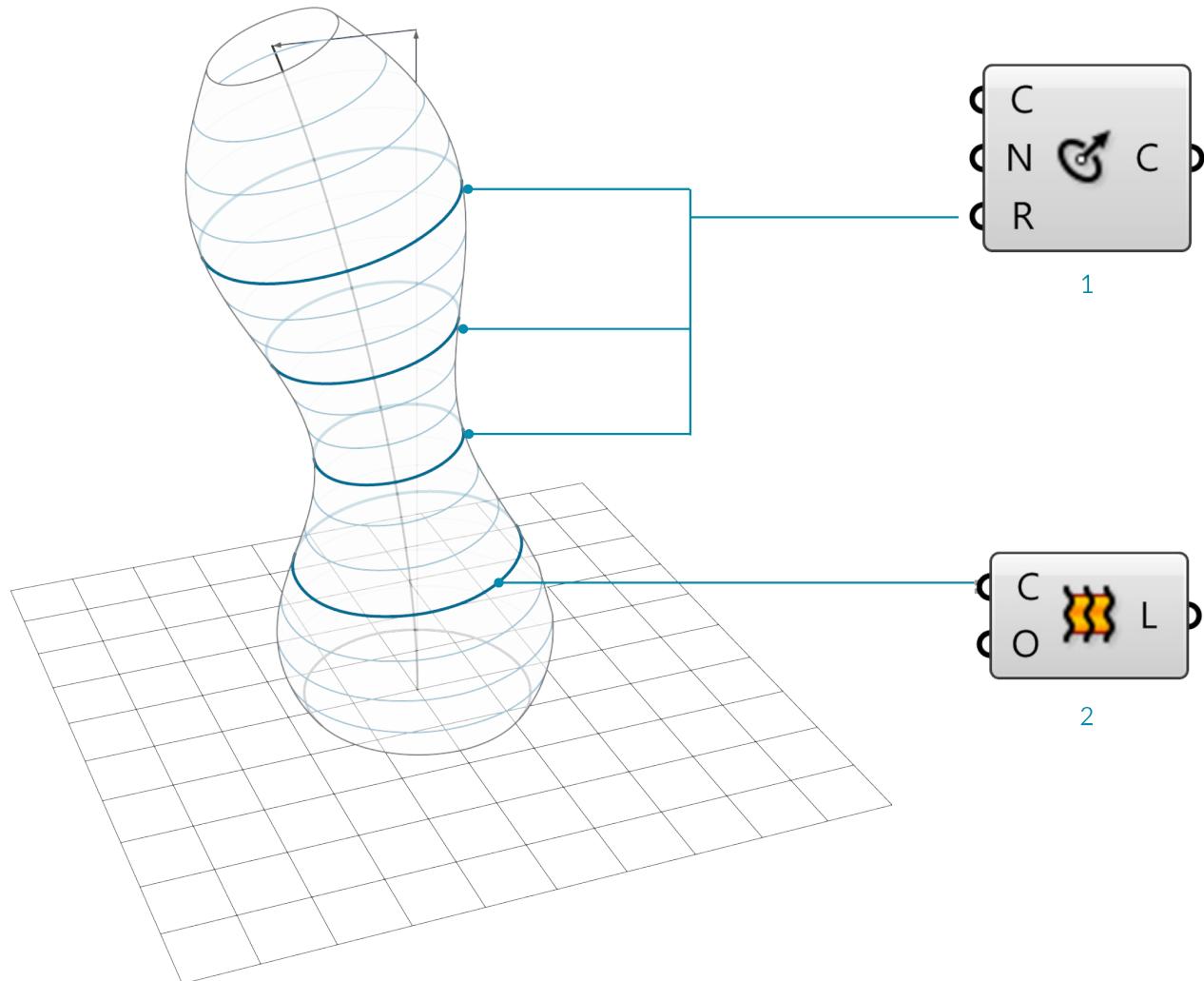
1.2.5.2. ЛОГИЧЕСКИЙ ГРАФИК

Все объекты и связи, соединяющие объекты, представляют логический график нашей программы. Этот график показывает поток данных, взаимозависимости каждого входа от подключенного выхода. Каждый раз когда график изменяется, иногда в связи с "загрязненностью", каждое соединение downstream и объект будут обновляться.



1. Перепараметризовать домейн кривой между 0.0 и 1.0.
2. Ссылка на кривую из Rhino.
3. Разделить кривую на 13 равных сегментов.
4. Запустить значение параметров на каждой точке деление кривой через график.
5. Умножьте каждое значение на 27.

6. Нарисуйте круг у каждой точки деления вдоль кривой, нормальный тангенс вектор у каждой точки с радиусом, определяемым значением параметра (t), изменяемым при помощи graph mapper и умноженным на 27.
7. Пространство между поверхностями кругов.



1. Изменяемый радиус круга.
2. Пространство между кругами

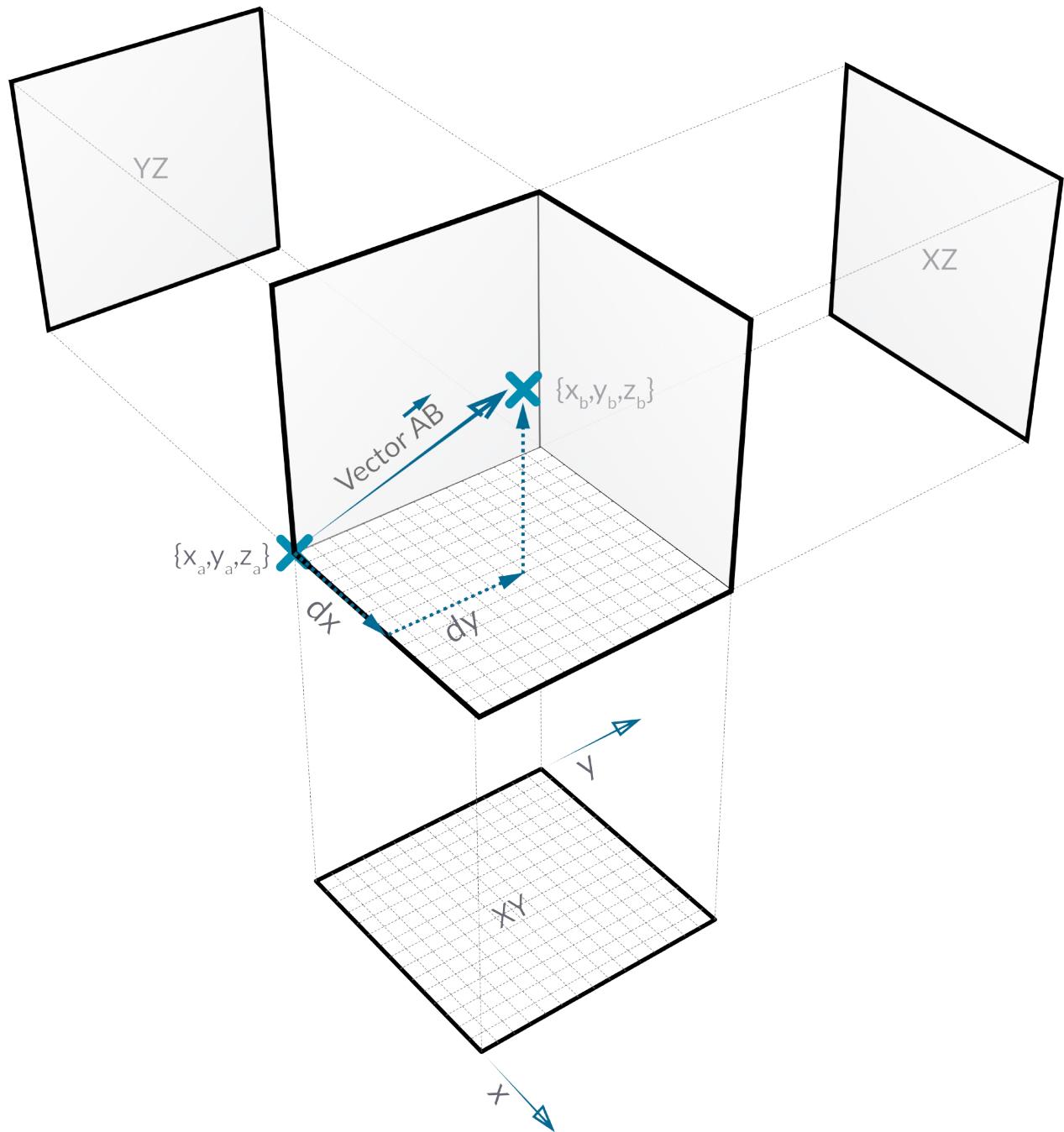
1.3. Построение блоков алгоритмов

Эта глава расскажет вам о базовой геометрии и математических концептах, и о том, как они применяются и используются в Grasshopper.



1.3.1. Точки, плоскости и векторы

Все начинается с точки. Точка - это ничего более, чем одно или несколько значений, называемых координатами. Число значений координат соответствует числу измерений пространства, в котором оно располагается. Точки, плоскости и векторы - это основа для создания и трансформации геометрии в Grasshopper.

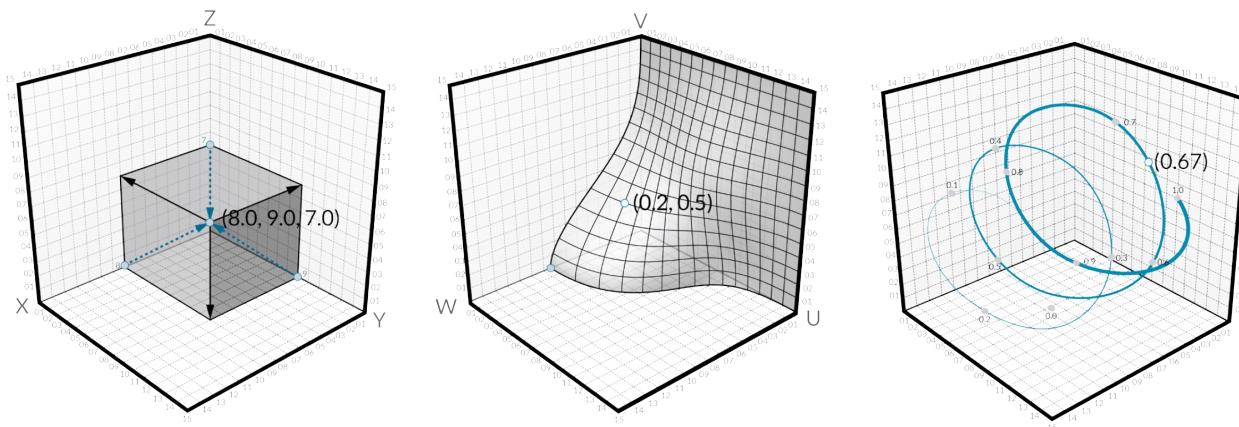


1.3.1.1 ТОЧКИ

Точки в пространстве 3D имеют три координаты - обычно их обозначают $[x,y,z]$. Точки в пространстве 2D имеет только две координаты, которые обозначают либо $[x,y]$ либо $[u,v]$, в зависимости от того, о каком двухмерном пространстве мы говорим. 2D параметрическое пространство ограничено конечной поверхностью. Тем не менее, она все-таки непрерывная, т.е. гипотетически существует бесконечное число точек на поверхности, но максимальное расстояние между этими точками очень сильно ограничено. 2D параметрические

координаты действительны, только если они не превышают определенный диапазон. На примере показано, что диапазон был установлен от 0.0 до 1.0 для обоих направлений [u] и [v], но также это может быть и любой конечный диапазон. Точка с координатами [1.5, 0.6] будет находиться где-то снаружи поверхности и, поэтому, не будет работать.

Так как поверхность, которая определяет это параметрическое пространство находится в правильном 3D пространстве, мы всегда можем переместить параметрическую координату в координату 3D пространства. Точка [0.2, 0.5] на поверхности, например, такая же как точка [1.8, 2.0, 4.1] в мировых координатах. Как только мы изменили поверхность, 3D координаты, которые связаны с [0.2, 0.5] будут меняться.

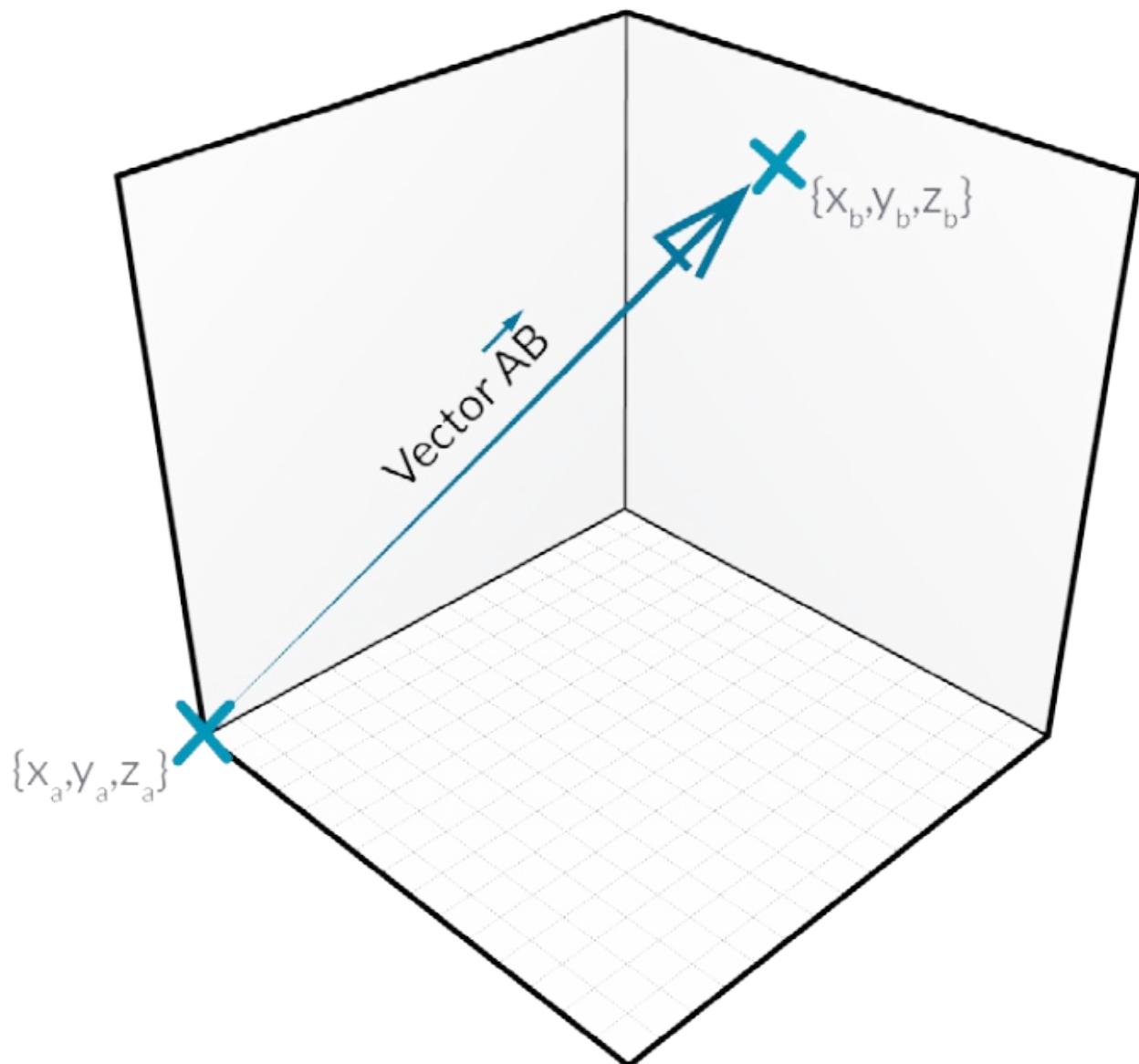


Если это сложная идея для понимания, то вам может помочь следующее - представить себя и свое положение в пространстве. Мы обычно используем местную систему координат, чтобы описать наше положение: "Я сижу на третьем месте седьмого ряда в кинотеатре", "Я на заднем сидении". Если вы находитесь в автомобиле, едущем по дороге, ваше положение в системе глобальных координат будет меняться все время, даже если вы остаетесь сидеть в том же самом кресле.

1.3.1.2. ВЕКТОРЫ

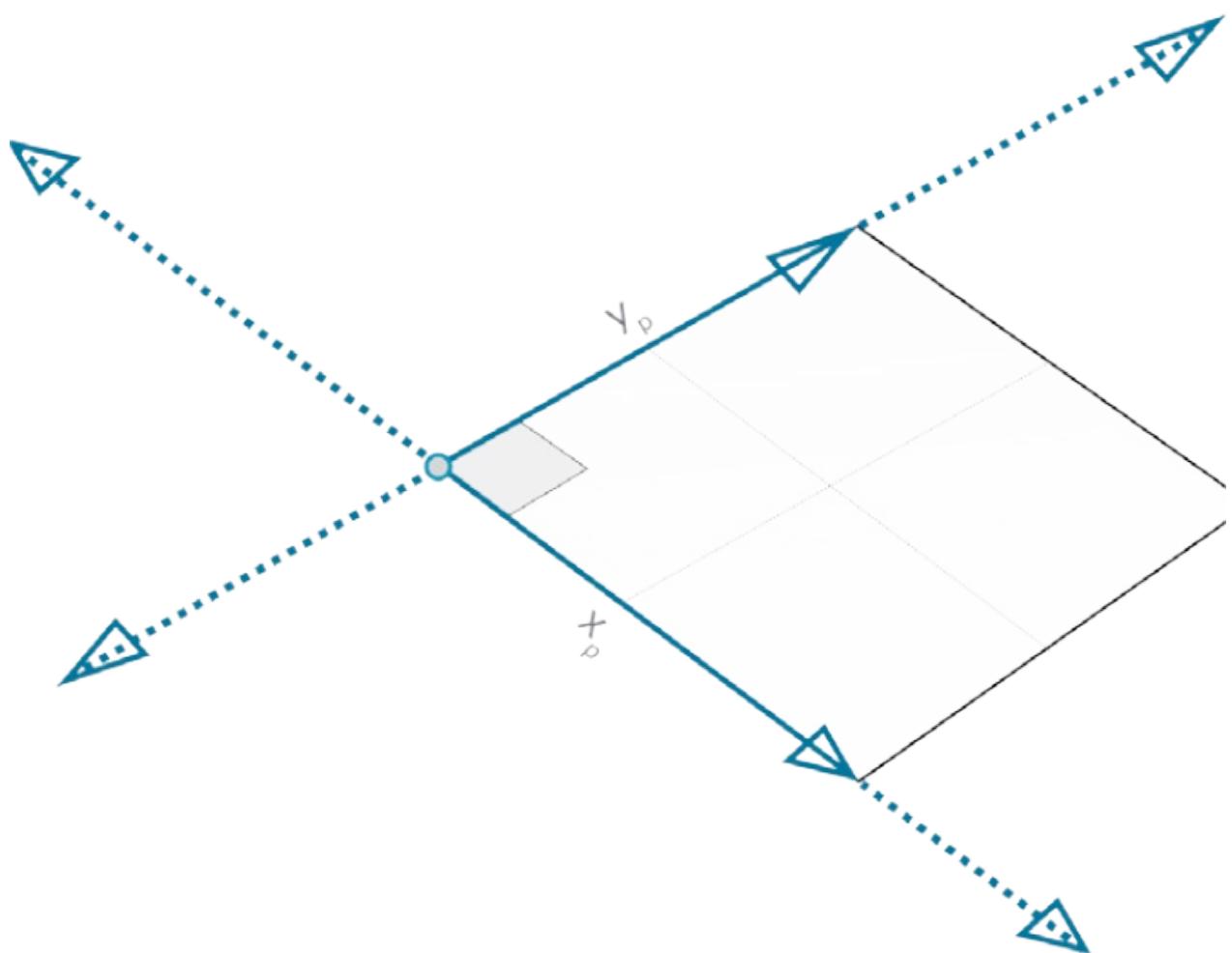
Вектор - это геометрическая величина, описывающая Направление и Амплитуду. Векторы абстрактны, т.е. они представляют величину, а не геометрический элемент.

Векторы неотделимы от точек. Так как, они оба представляют собой список из трех чисел, поэтому совершенно нет возможности определить, что представляет из себя некий список - точку или вектор. Хотя на практике разница есть; точки абсолютны, векторы относительны. Когда мы относимся к списку из трех пар как к точке - она представляет определенную координату в пространстве, когда мы относимся к нему как к вектору, он представляет собой определенное направление. Вектор - это стрелка в пространстве, которая всегда начинается с мировых координат (0.0, 0.0, 0.0) и заканчивается в указанной координате.



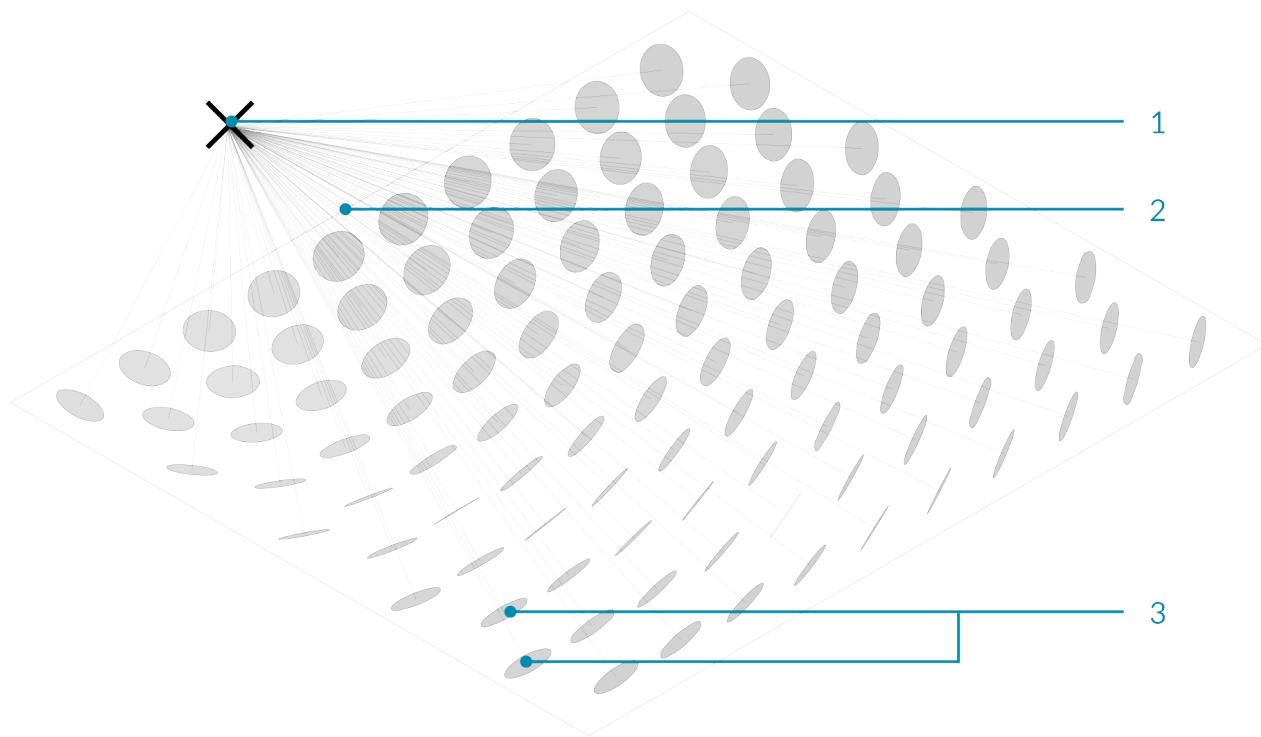
1.3.1.3. ПЛОСКОСТИ

Плоскости - "плоские" и вытягиваются бесконечно в двух направлениях, определяя локальную систему координат. Плоскости не подлинные объекты в Rhino, они используются для определения системы координат 3D пространстве. На деле, лучше думать о плоскостях как о векторах, т.к. это просто математические структуры.



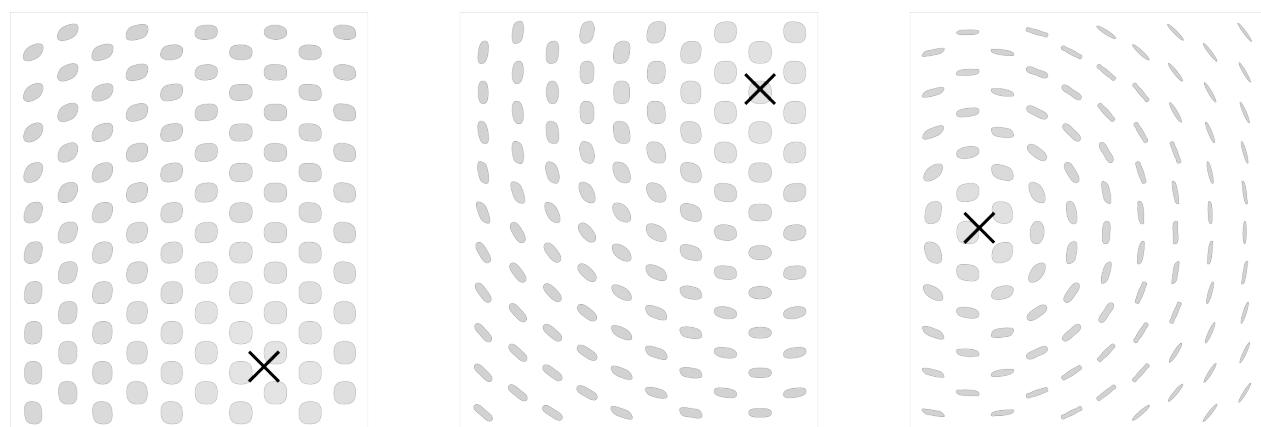
1.3.2. Работая с атTRACTорами

АтTRACTоры - это точки, которые ведут себя как виртуальные магниты - либо притягивают либо отталкивают другие объекты. В Grasshopper, любая геометрия, взятая из Rhino или созданная в Grasshopper, может быть использована как атTRACTор. АтTRACTоры могут влиять на любое число параметров окружающих объектов включая масштаб, вращение, цвет и положение. Эти параметры изменяются на основе их отношений с геометрией атTRACTора.



1. Точка атTRACTора
2. Векторы
3. Круги ориентируются по направлению к атTRACTору на основе их нормалей

На изображении выше, векторы изображены между точкой атTRACTора и точкой начала координат каждого круга. Эти векторы используются для определения ориентации кругов, таким образом, что они всегда обращены к точке атTRACTора. Тот же самый атTRACTор может быть использован для изменения других параметров кругов. Например, те круги, которые ближе всего к атTRACTору, могут иметь больший размер, используя длину каждого вектора, чтобы масштабировать радиус каждого круга.

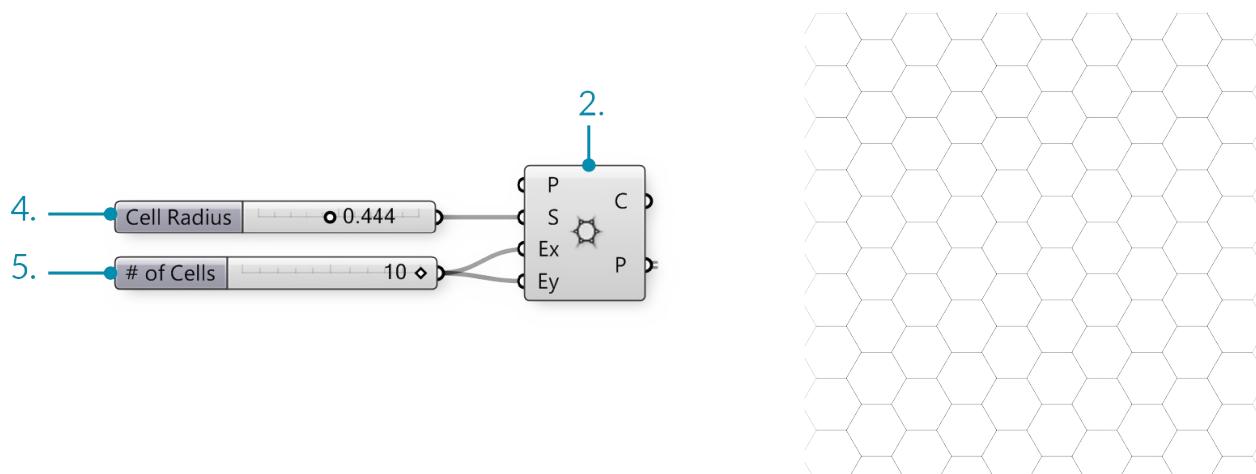


1.3.2.1. ОПРЕДЕЛЕНИЕ АТTRACTОРА

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

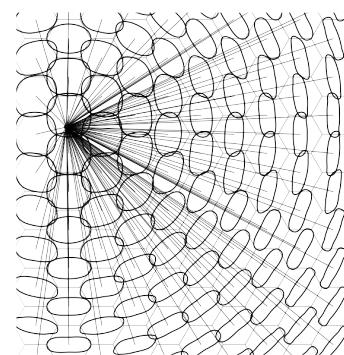
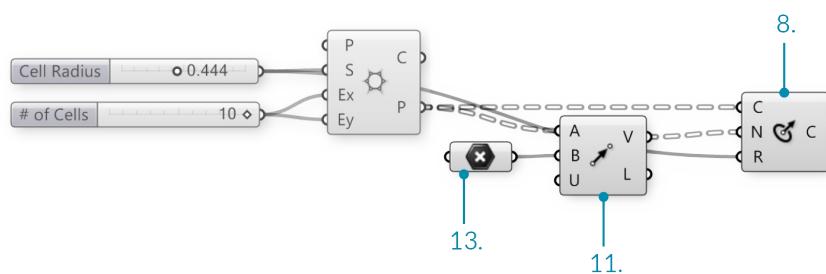
В этом примере, мы будем использовать точку атTRACTора, чтобы расположить сетку кругов, основанную на векторах между точками начала координат кругов и точками атTRACTоров. Каждый круг будет ориентироваться таким образом, что он будет перпендикулярен (обращен) к точке атTRACTора.

01.	Чтобы начать новое определение, нажмите Ctrl+N в Grasshopper	
02.	Зайдите в Vector/Grid/Hexagonal - перетащите компонент Hexagonal Grid на холст	
03.	Зайдите в Params/Input/Slider - перетащите два слайдера Numeric Sliders на холст	
04.	Дважды кликните по первому слайдеру Numeric Sliders и установите следующее: Name (Имя): Cell Radius (радиус ячейки) Rounding (Округление): Floating Point (с плавающей точкой) Lower Limit (нижняя граница): 0.000 Upper Limit (верхняя граница): 1.000 Value (значение): 0.500	
05.	Дважды кликните по второму слайдеру Numeric Sliders и установите следующее: Name (Имя): # of Cells (число ячеек) Rounding (Округление): Integers (целые числа) Lower Limit (нижняя граница): 0 Upper Limit (верхняя граница): 10 Value (значение): 10	
06.	Соедините слайдер Number Slider (Cell Radius) с входом Size (S) компонента Hexagon Grid	
07.	Соедините слайдер Number Slider (# of Cells) с входом Extent X (Ex) и входом Extent Y (Ey) компонента Hexagon Grid	

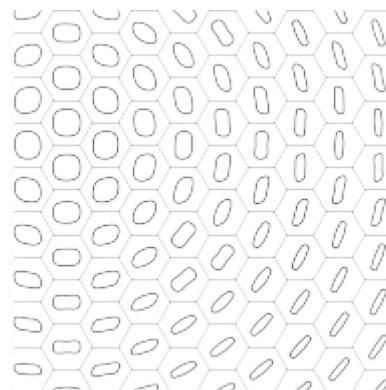
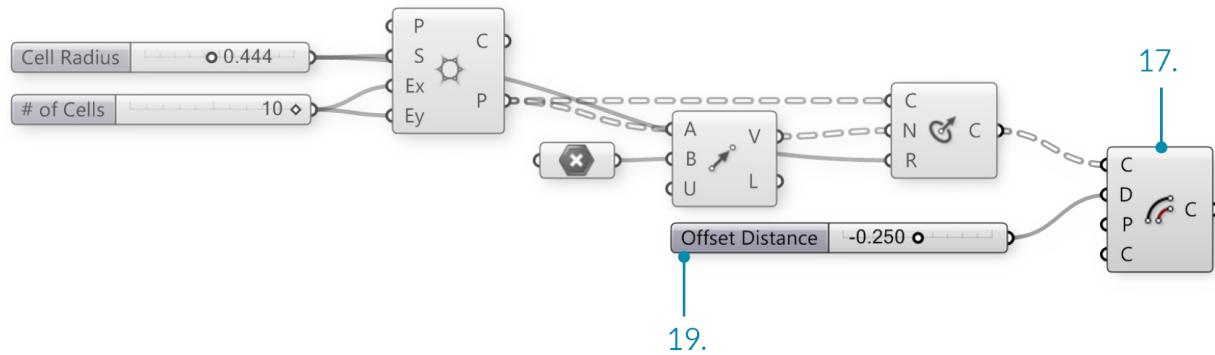


08.	Зайдите в Curve/Primitive/Circle CNR - перетащите компонент Circle CNR на холст	
-----	---	---

09.	Соедините выход Points (P) компонента Hexagon Grid с входом Center (C) компонента Circle CNR	
10.	Соедините слайдер Number Slider (Cell Radius) с входом Radius (R) компонента Circle CNR .	
11.	Зайдите в Vector/Vector/Vector 2Pt - перетащите компонент Vector 2Pt на холст	
12.	Соедините выход Points (P) компонента Hexagonal Grid с входом Base Point (A) компонента Vector 2Pt .	
13.	Зайдите в Params/Geometry/Point – перетащите компонент Point на холст	
14.	Кликните правой клавишей мыши по компоненту Point и выберите set one point. В модели пространства выберите место, где вы хотите разместить точку аттрактора	
15.	Соедините компонент Point с входом Tip Point (B) компонента Vector 2Pt	
16.	Соедините выход Vector (V) компонента Vector 2Pt с входом Normal (N) компонента Circle CNR .	

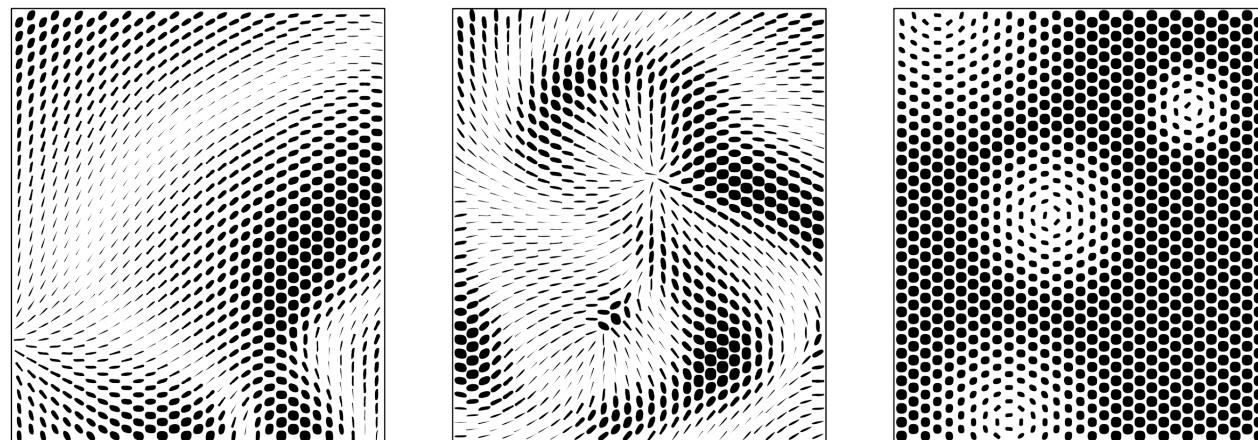
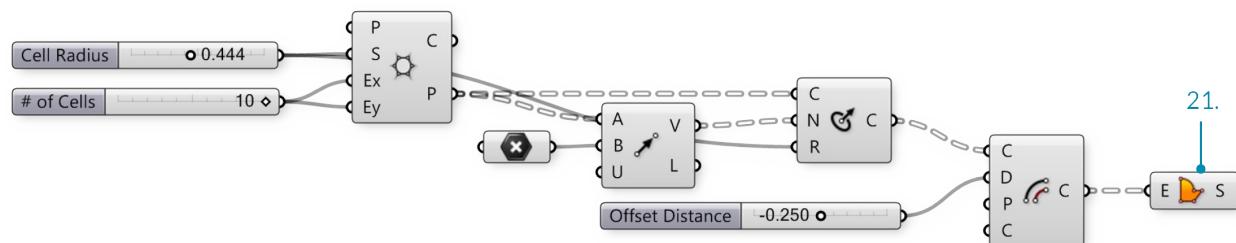


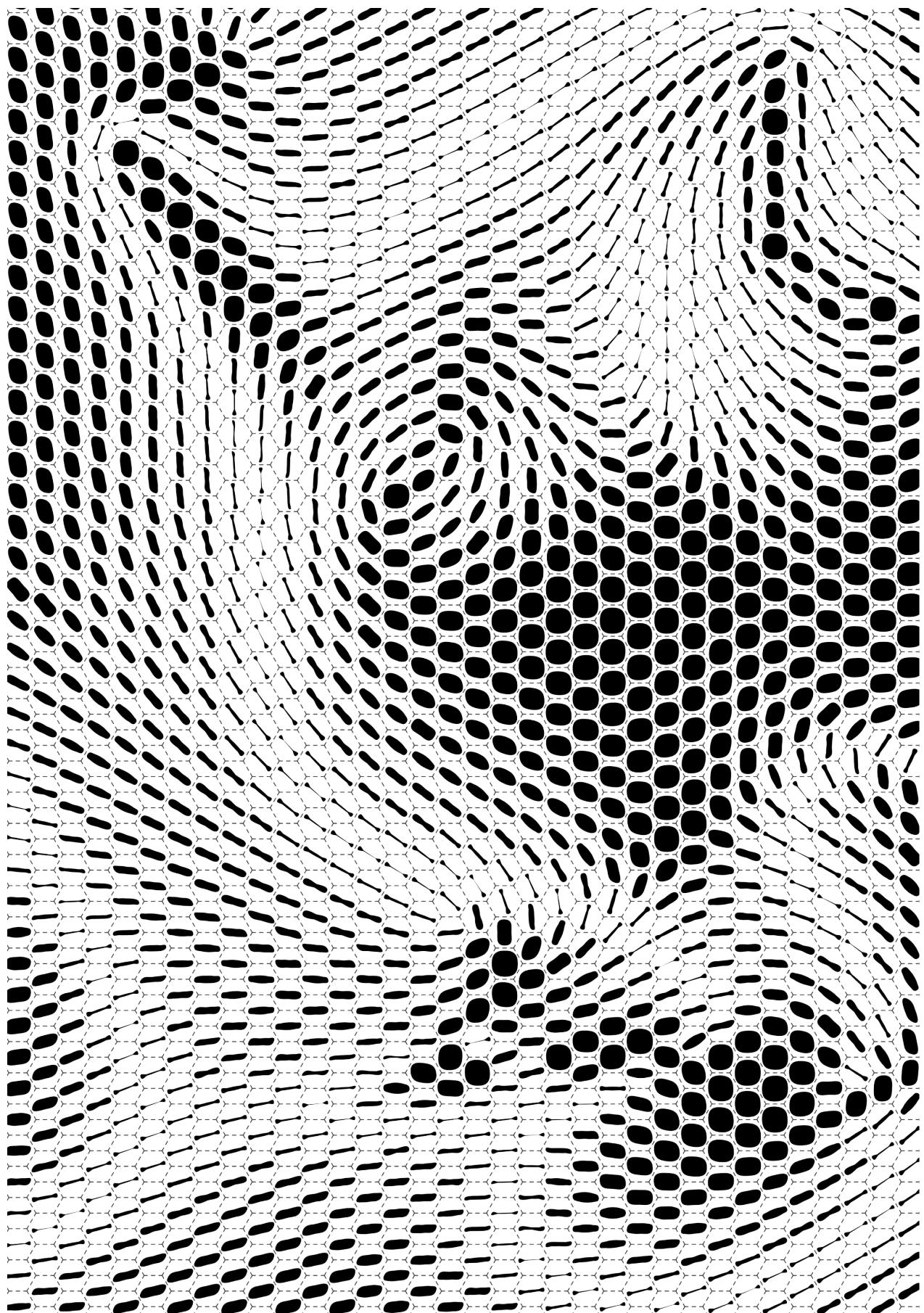
17.	Зайдите в Curve/Util/Offset – перетащите компонент Offset Component на холст.	
18.	Зайдите в Params/Input/Slider - перетащите слайдер Numeric Slider на холст	
19.	Дважды кликните по слайдеру и установите следующее: Name: Offset Distance Rounding: Floating Point Lower Limit: - 0.500 Upper Limit: 0.500 Value: -0.250	
20.	Соедините слайдер Number Slider (Offset Distance) с входом Distance (D) компонента Offset	



19.

21.	Зайдите в Surface/Freeform/Boundary Surfaces – перетащите компонент Boundary Surfaces на холст	
22.	Соедините выход Curves (C) компонента Offset с входом Edges (E) компонента Boundary Surfaces	





1.3.3. Математика, Выражения и Условия

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Знание как работать с числовыми данными - необходимый навык при изучении использования Grasshopper. Grasshopper содержит много компонентов для выполнения математических операций, определения условий и управления числовыми множествами.

В математике, числа организуются в множества и с двумя из них вы, возможно, знакомы:

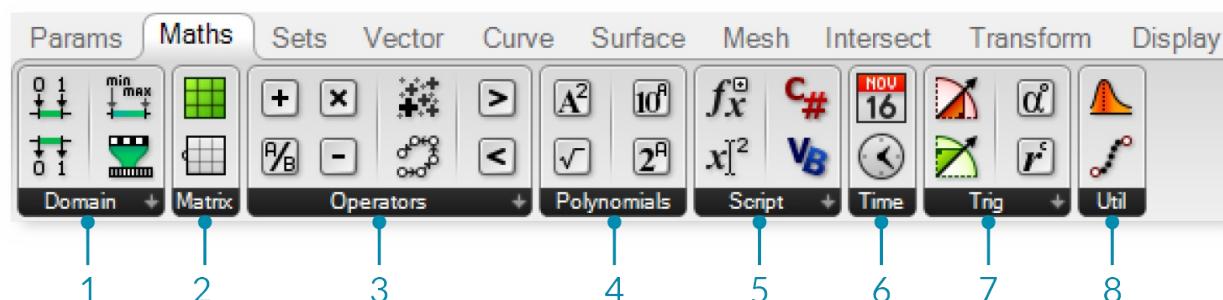
Целые числа: [..., -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, ...]

Числа с плавающей точкой: [8, ..., -4.8, -3.6, -2.4, -1.2, 0.0, 1.234, e, 3.0, 4.0, ..., 8]

Несмотря на то, что существуют и другие типы числовых множеств, эти два интересует нас больше всего, потому что они активно используются в Grasshopper. Хотя существуют и ограничения в точном представлении этих множеств в цифровой среде, мы можем аппроксимировать их с высокой степенью точности. В дополнение, следует понимать, что различие между Целыми числами и Числами с плавающей точкой соответствует различию между дискретными и непрерывными диапазонами. В этой главе, мы собираемся исследовать различные методы работы и определения различных числовых множеств.

1.3.3.1. ВКЛАДКА MATHS

Большинство компонентов, работающих с математическими операциями и функциями, располагаются во вкладке Math:



1. Диапазоны используются для определения диапазонов значений (ранее назывались интервалами) между двумя числами. Компоненты в свитке Domain позволяют вам создавать или разрушать различные типы диапазонов.
2. В математике, матрица - это массив чисел, организованных в ряды и колонки. Этот свиток содержит набор инструментов для создания и изменения матриц.
3. Operators (Операторы) используются для выполнения математических операций, таких как Сложение, Вычитание, Умножение и т.д. Условные операторы позволяют вам определить, является ли числовое множество больше чем, меньше чем или равным другому числовому множеству.
4. Polynomials (Многочлены) - одна из самых важных идей в алгебре и во всей математике и науке. Вы можете использовать компоненты из этого свитка для вычисления факториалов, логарифмов или для возведения числа в n-ую степень.
5. Свиток Script содержит одно- и многозначные выражения, а также компоненты скриптов VB.NET и C#.
6. Эти компоненты позволяют решить тригонометрические функции, такие как Синус, Косинус, Тангенс и др.

7. Свиток Time включает набор компонентов, которые позволяют создавать примеры даты и времени.
8. Свиток Utility - это 'мешок' с полезными компонентами, которые можно использовать в различных математических уравнениях. Зайдите сюда, если вы пытаетесь найти максимальные и минимальные значения двух списков чисел; или среднее в группе чисел.

1.3.3.2. ОПЕРАТОРЫ

Как было указано ранее, операторы - это наборы компонентов, которые используют алгебраические функции с двумя числовыми значениями входа, которые дают одно значение выхода.

Большую часть времени, вы будете использовать Математические операторы (Math) для совершения арифметических действий с числовыми множествами. Тем не менее, эти операторы могут также использоваться для различных типов данных, включая точки и векторы.

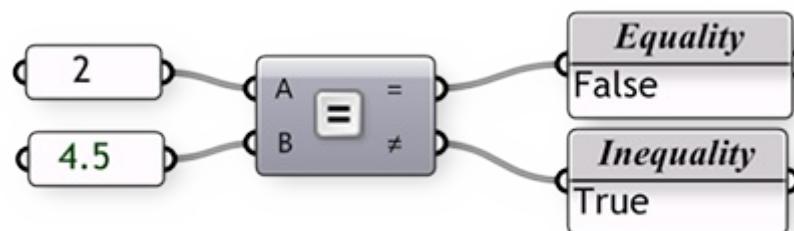


1.3.3.3. УСЛОВНЫЕ ОПЕРАТОРЫ

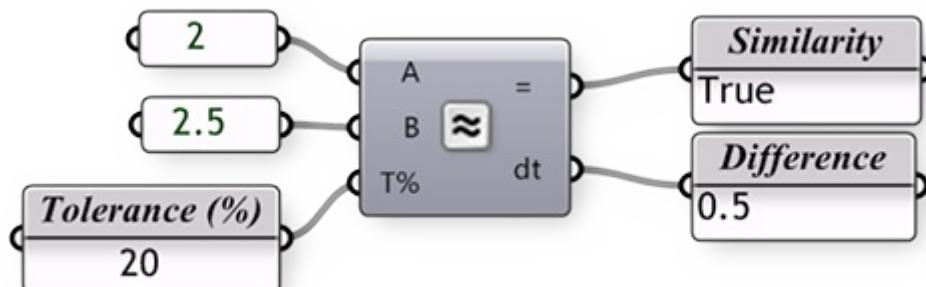
Почти каждый язык программирования имеет метод определения условных выражений. В большинстве случаев, программист создает части кода, чтобы задать простой вопрос "Что если". Что если область контура

поля превышает программные требования? Или, что если изгиб моей крыши превышает реальные данные? Эти важные вопросы представляют высокий уровень абстрактного мышления. Компьютерные программы способны анализировать вопросы "что если" и действовать в зависимости от ответа на этот вопрос. Давайте посмотрим на очень простое условное выражение, которое программа может понять: Если объект - кривая, удалить его. Часть кода сначала смотрит на объект и определяет единичные булевые значения для решения кривая это или нет. Среднего здесь не дано. Булевое значение равно True (правда), если объект - это кривая, или False (Ложь), если объект не кривая. Вторая часть утверждения выполняет действие в зависимости от результата условного утверждения; в этом случае, если объект - это кривая, то решение - удалить ее. Это условное утверждение называется If (Если) утверждение. Существуют четыре условных оператора (располагаются во вкладке Math/ свиток Operators), которые определяют условие и выдают булевое значение.

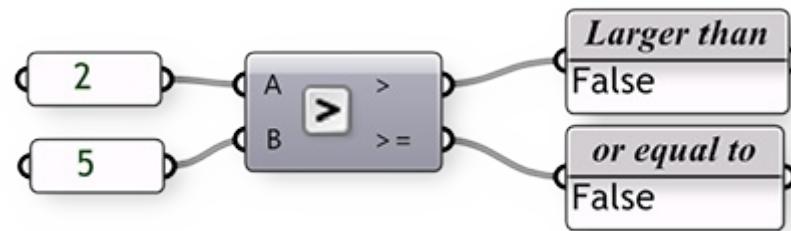
Компонент Equality (равенство) берет два списка и сравнивает первый элемент списка A с первым элементом списка B. Если два значения одинаковые, тогда создается булевое значение True, напротив, если два значения неравные, тогда создается булевое значение False. Компонент повторяет этот процесс со списками в соответствии с заданными алгоритмами совпадения (по умолчанию устанавливается самый длинный список). У этого компонента имеются два выхода. Первый возвращает список булевых значений, который показывает, какие из значений в списке были равны другим значениям. Второй выход возвращает список, который показывает, какие значения не были равны другим, либо список, который получается из первого выхода.



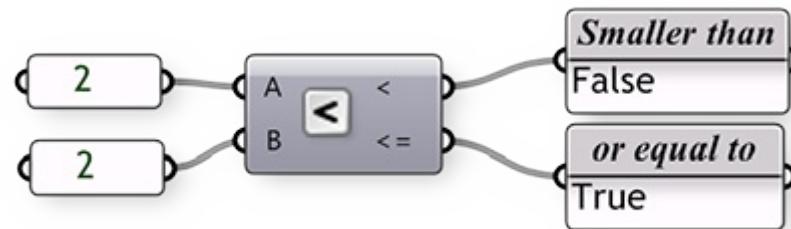
Компонент Similarity (сходство) оценивает два списка данных и тестирует сходство между двумя числами. На деле, это очень похоже на способ работы компонента Equality, который сравнивает два списка, за одним исключением: у него есть вход для процентов, который определяет степень списка A, которую список B может исказить до того как неравенство будет определено. Компонент Similarity также имеет выход, который определяет абсолютное значение расстояния между двумя входными списками.



Компонент Larger Than берет два списка данных и определяет, больше ли первый элемент списка A, чем первый элемент списка B. Два выхода позволяют вам определить, хотите ли вы определить два списка в соответствии с условием больше чем (>) или больше чем и равняется (>=).



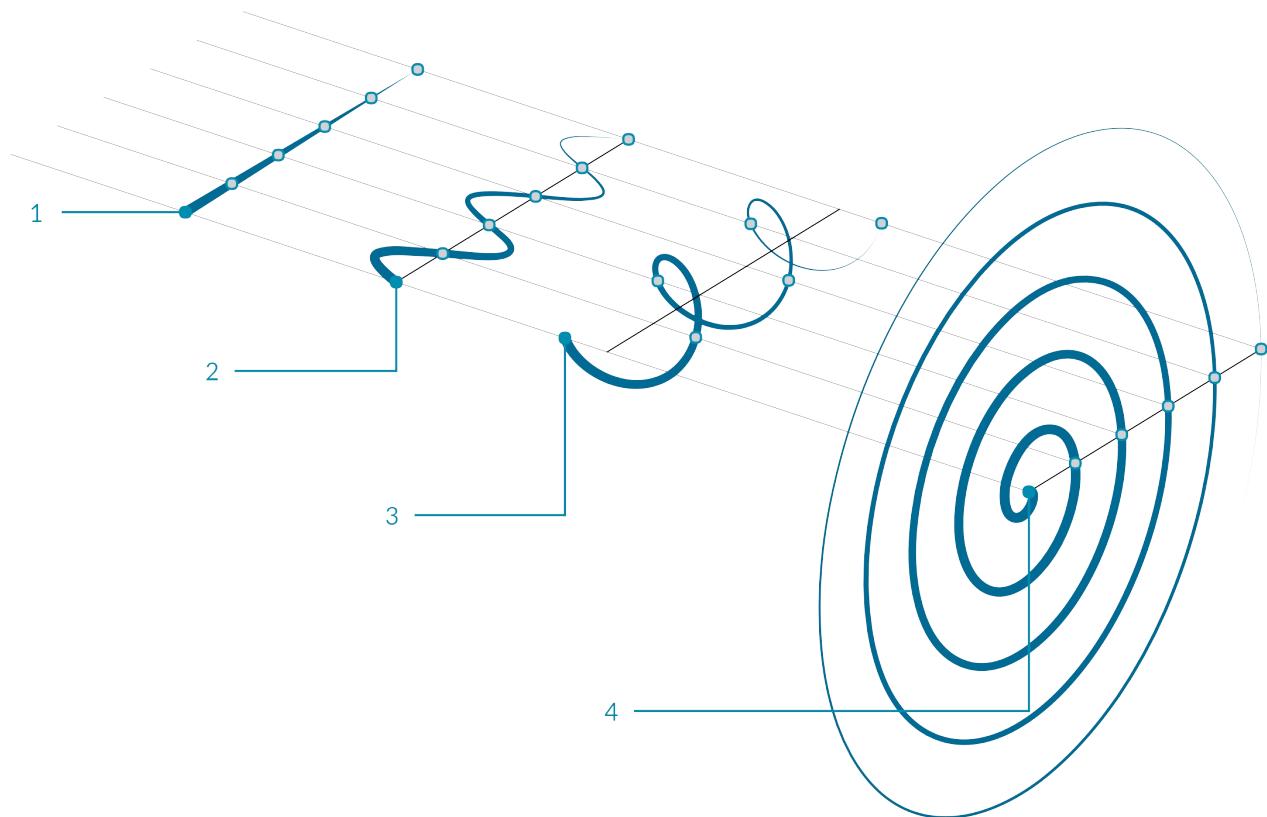
Компонент Smaller Than выполняет противоположное действие компонента Larger Than. Компонент Smaller Than определяет, если список А меньше, чем список Б и возвращает список булевых значений. Схоже, два выхода позволяют определить вам, хотели бы вы определить каждый список в соответствии с условием меньше чем (<) или меньше чем и равно (<=).



1.3.3.4. ТРИГОНОМЕТРИЧЕСКИЕ КОМПОНЕНТЫ

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Мы уже показали, что мы можем использовать компонент Expression (или Evaluate) для определения условных выражений, а также для вычисления алгебраических уравнений. Тем не менее, существуют другие способы вычисления простых выражений, используя несколько встроенных Тригонометрических функций. Мы можем использовать эти функций для определения периодического феномена как синусоидальная волновая форма, например, волны океана, звуковые волны и световые волны.



1. Линия

 $y(t) = 0$

2. Синусная кривая

 $y(t) = \sin(t)$

3. Винтовая спираль

 $x(t) = \cos(t)$ $y(t) = \sin(t)$ $z(t) = b(t)$

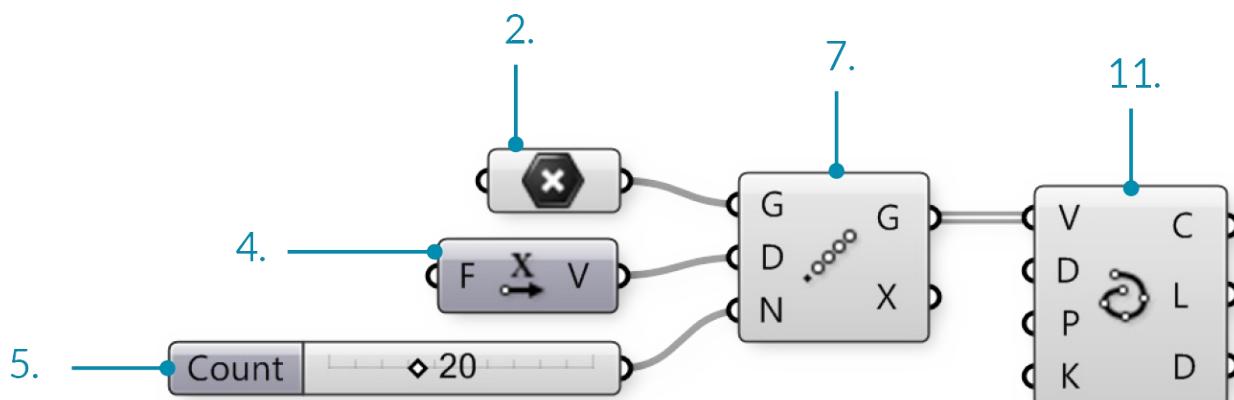
4. Спираль

 $x(t) = t * \cos(t)$ $y(t) = t * \sin(t)$

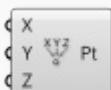
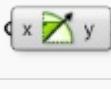
В этом примере, мы будем использовать Grasshopper для создания различных тригонометрических кривых, используя компоненты тригонометрических функций из раздела Math:

01.	Чтобы начать новое определение, нажмите Ctrl+N в Grasshopper	
02.	Зайдите в Params/Geometry/Point – перетащите параметр Point на холст	
03.	Кликните правой клавишей мыши по параметру Point и кликните по Set One Point – выберите точку в видовом окне Rhino	
04.	Зайдите в Vector/Vector/Unit X – перетащите компонент Unit X на холст	
05.	Зайдите в Params/Input/Number Slider – перетащите слайдер Number Slider на холст	
06.	Дважды кликните по слайдеру Number Slider и установите следующее: Rounding: Integer Lower Limit: 10 Upper Limit: 40	

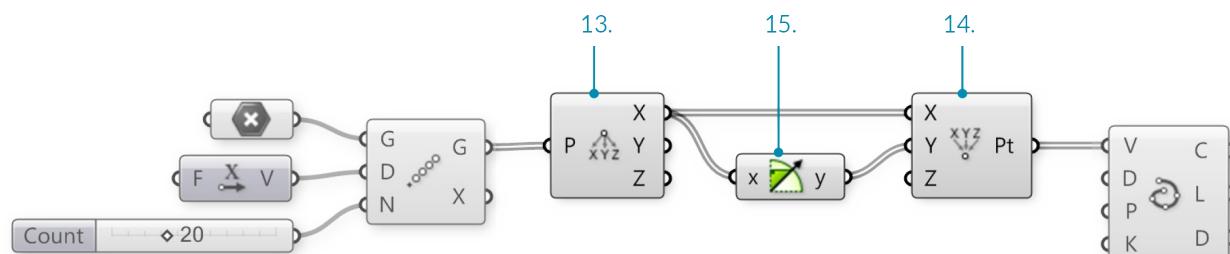
	Value: 20	
07.	Зайдите в Transform/Array/Linear Array – перетащите компонент Linear Array на холст	
08.	Соедините выход параметра Point с входом Geometry (G) компонента Linear Array	
09.	Соедините выход Unit Vector (V) компонента Unit X с входом Direction (D) компонента Linear Array Вы должны увидеть линию из 20 точек вдоль оси X в Rhino. Настройте слайдер, чтобы менять количество точек в массиве.	
10.	Соедините выход Number Slider с входом Count (N) компонента Linear Array	
11.	Зайдите в Curve/Spline/Interpolate – перетащите компонент Interpolate Curve на холст	
12.	Соедините выход Geometry (G) компонента Linear Array с входом Vertices (V) компонента Interpolate Curve	



Мы только что создали линию, соединив массив точек с кривой. Давайте попытаемся использовать некоторые из Тригонометрических компонентов Grasshopper, чтобы изменить кривую:

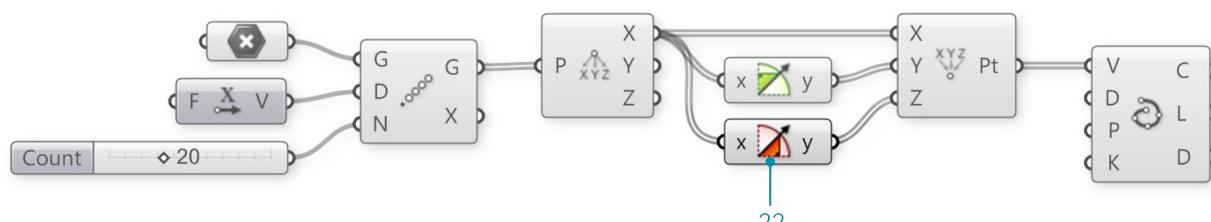
13.	Зайдите в Vector/Point/Deconstruct – перетащите компонент Deconstruct на холст	
14.	Зайдите в Vector/Point/Construct Point - перетащите компонент Construct Point на холст	
15.	Зайдите в Maths/Trig/Sine - перетащите компонент Sine на холст	
16.	Отсоедините связь от входа Vertices (V) компонента Interpolate Curve . Вы можете разъединить связи, зажав Control и перетащив связь, либо кликнув правой клавишей на входе и выбрав Disconnect	
	Соедините выход Geometry (G) компонента Linear Array с входом Point (P)	

	компонента Deconstruct	
18.	Соедините выход Point X (X) компонента Deconstruct с входом X coordinate (X) компонента Construct Point	
19.	Соедините второй связью выход Point X (X) компонента Deconstruct с входом Value (x) компонента Sine	
20.	Соедините выход Result (y) компонента Sine с входом Y coordinate (Y) компонента Construct Point Мы переделали наши точки с такими же X значениями, изменив значения Y с синусоидальной кривой.	
21.	Соедините выход Point (Pt) компонента Construct Point с входом Vertices (V) компонента Interpolate	



Сейчас вы должны видеть синусоидальную волновую кривую вдоль оси X в Rhino

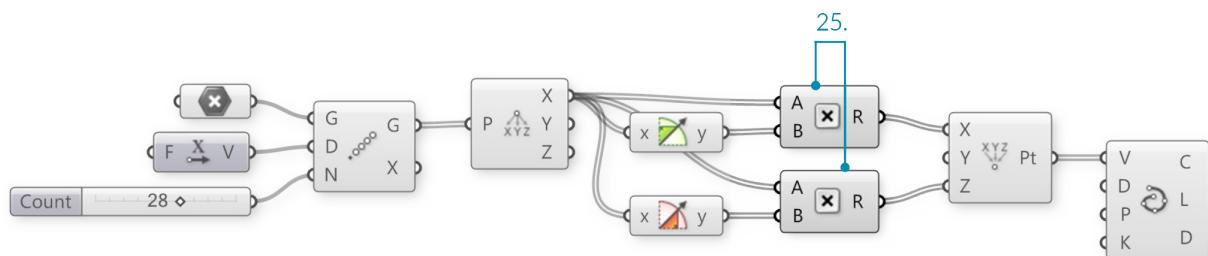
22.	Зайдите в Maths/Trig/Cosine – перетащите компонент Cosine на холст	
23.	Соедините третьей связью выход Point X (X) компонента Deconstruct с входом Value (x) компонента Cosine	
24.	Соедините выход Result (y) компонента Cosine с входом Z coordinate (Z) компонента Construct Point	



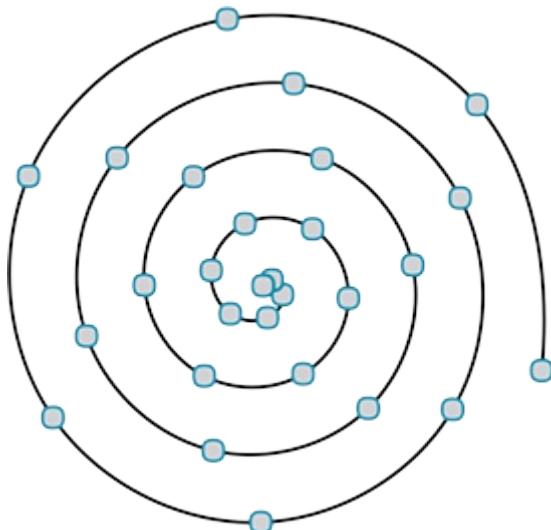
Сейчас мы создали 3D винтовую спираль

--	--	--

25.	Зайдите в Maths/Operators/Multiplication – перетащите два компонента Multiplication на холст	
26.	Соедините связями выход Point X (X) компонента Deconstruct с входом (A) каждого компонента Multiplication	
27.	Соедините выход Result (y) компонента Sine с входом (B) первого компонента Multiplication	
28.	Соедините выход Result (y) компонента Cosine с входом (B) второго компонента Multiplication	
29.	Отсоедините связь от входа Y Coordinate (Y) компонента Construct Point	
30.	Соедините выход Result (R) первого компонента Multiplication со входом X Coordinate (X) компонента Construct Point	
31.	Соедините выход Result (R) второго компонента Multiplication со входом Z Coordinate (Z) компонента Construct Point	



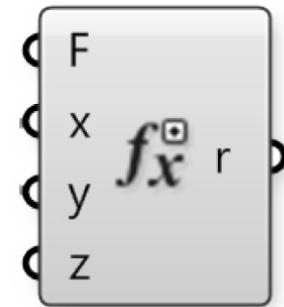
Сейчас вы должны увидеть спиральную кривую



1.3.3.5. ВЫРАЖЕНИЯ

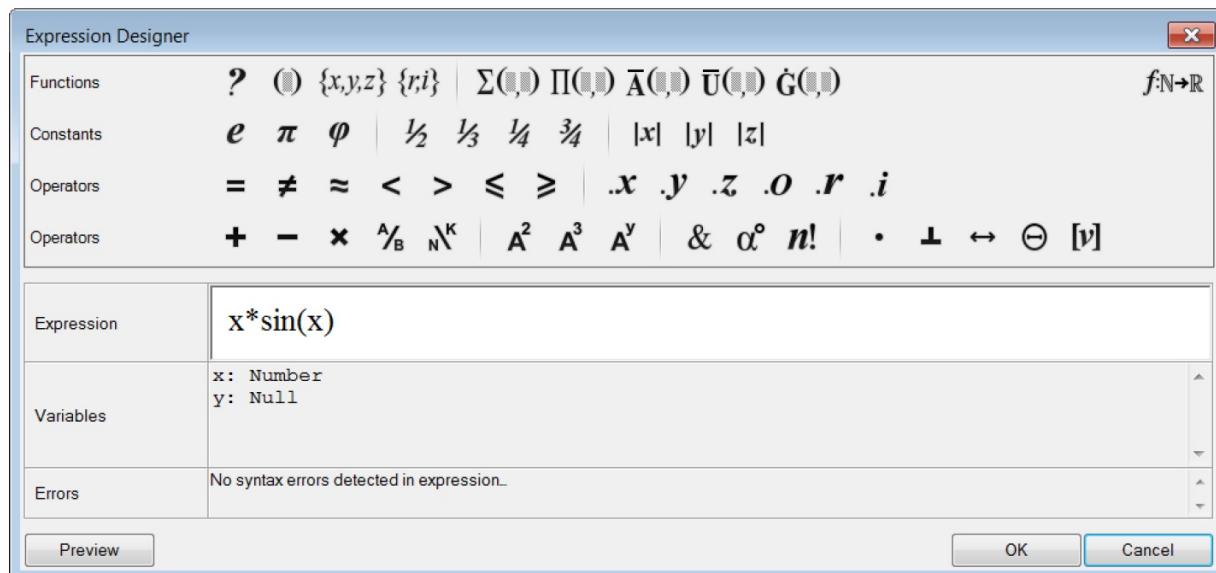
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Компонент **Expression** (и его брат компонент **Evaluate**) представляет собой очень гибкий инструмент, который можно использовать для различных приложений. Мы можем использовать компонент **Expression** (или **Evaluate**), чтобы решать математические алгоритмы и выдавать числовые данные.



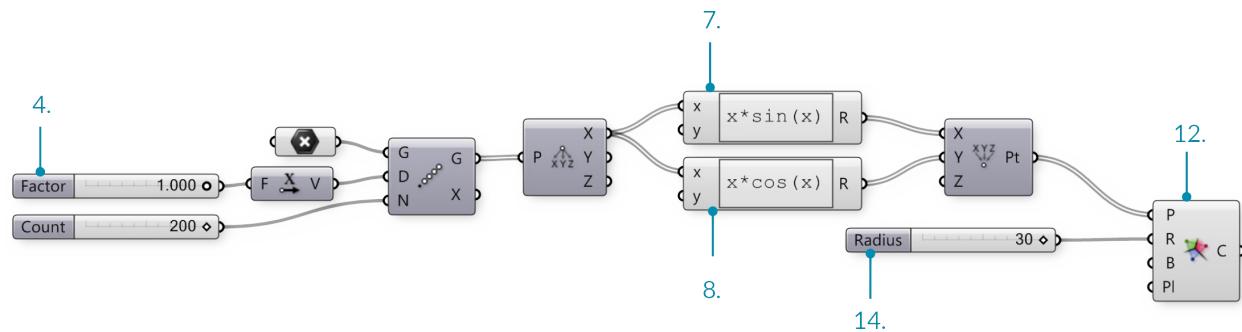
В следующем примере, мы рассмотрим математические спирали, которые можно наблюдать в природе, и как мы можем использовать несколько компонентов Functions для создания похожих паттернов в Grasshopper. Наши тригонометрические кривые мы возьмем за начальную точку.

01.	Из предыдущего определения Grasshopper откройте тригонометрические кривые	
02.	Удалите компоненты Sine , Cosine , Multiplication и Interpolate	
03.	Зайдите в Params/Input/Number Slider – перетащите слайдер Number Slider на холст	
04.	Дважды кликните по слайдеру Number Slider и установите следующее: Rounding: Float Lower Limit: 0.000 Upper Limit: 1.000 Value: 1.000	
05.	Подключите слайдер Number Slider к входу Factor (F) компонента Unit X. Этот слайдер позволит настроить расстояние между точками в массиве.	
06.	Зайдите в Maths/Script/Expression – перетащите два компонента Expression на холст	
07.	Дважды кликните на первом компоненте Expression чтобы открыть редактор Expression и изменить выражение: $x * \sin(x)$	
08.	8. Дважды кликните на втором компоненте Expression чтобы открыть редактор Expression и изменить выражение: $x * \cos(x)$	

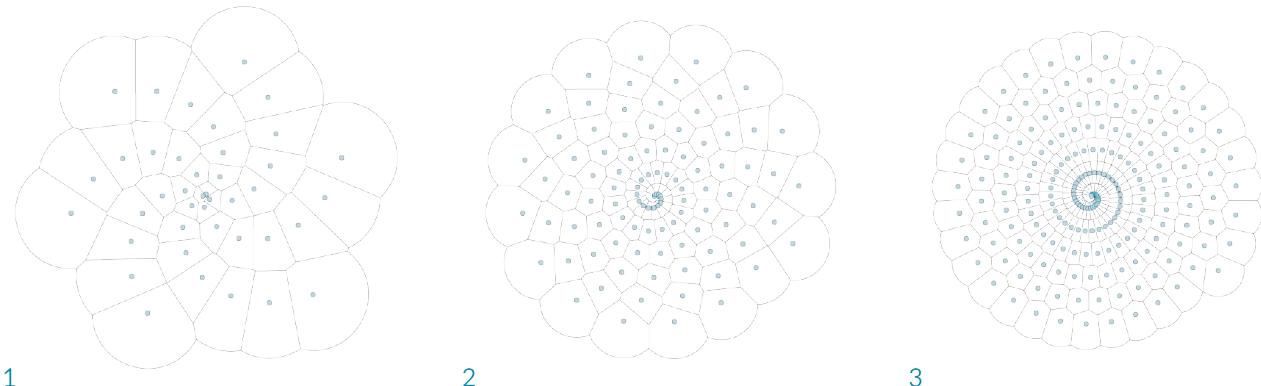


Дважды кликните на компоненте Expression, чтобы открыть редактор Grasshopper Expression

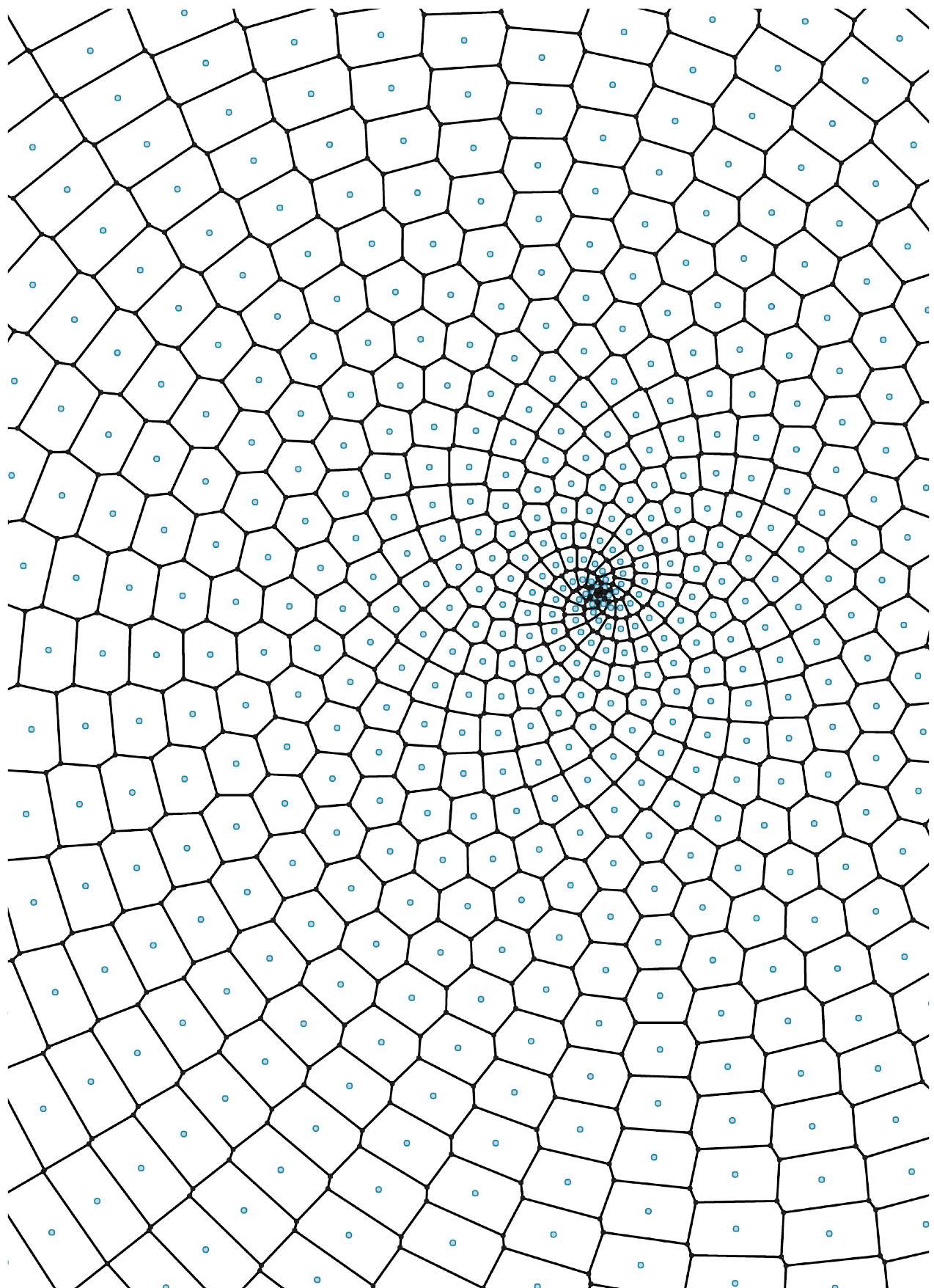
09.	Соедините две связи из выхода Point X (X) компонента Deconstruct с входом Variable x (x) каждого компонента Expression	
10.	Соедините выход Result (R) первого компонента Expression с входом X coordinate (X) компонента Construct Point	
11.	Соедините выход Result (R) второго компонента Expression с входом Y coordinate (Y) компонента Construct Point Мы заменили тригонометрические функции и операторы умножения компонентами Expression для того, чтобы определение было более эффективным.	
12.	Зайдите в Mesh/Triangulation/Voronoi – перетащите компонент Voronoi на холст	
13.	Зайдите в Params/Input/Number Slider – перетащите слайдер Number Slider на холст	
14.	Дважды кликните по слайдеру Number Slider и установите следующее: Rounding: Integer Lower Limit: 1 Upper Limit: 30 Value: 30	
15.	Подключите слайдер Number Slider к входу Radius (R) в компоненте Voronoi	
16.	Соедините выход Point (Pt) компонента Construct Point с входом Points (P) компонента Voronoi	



Вы можете создавать различные паттерны Вороного, меняя значение слайдеров Factor, Count и Radius.
Ниже приведены три примера:



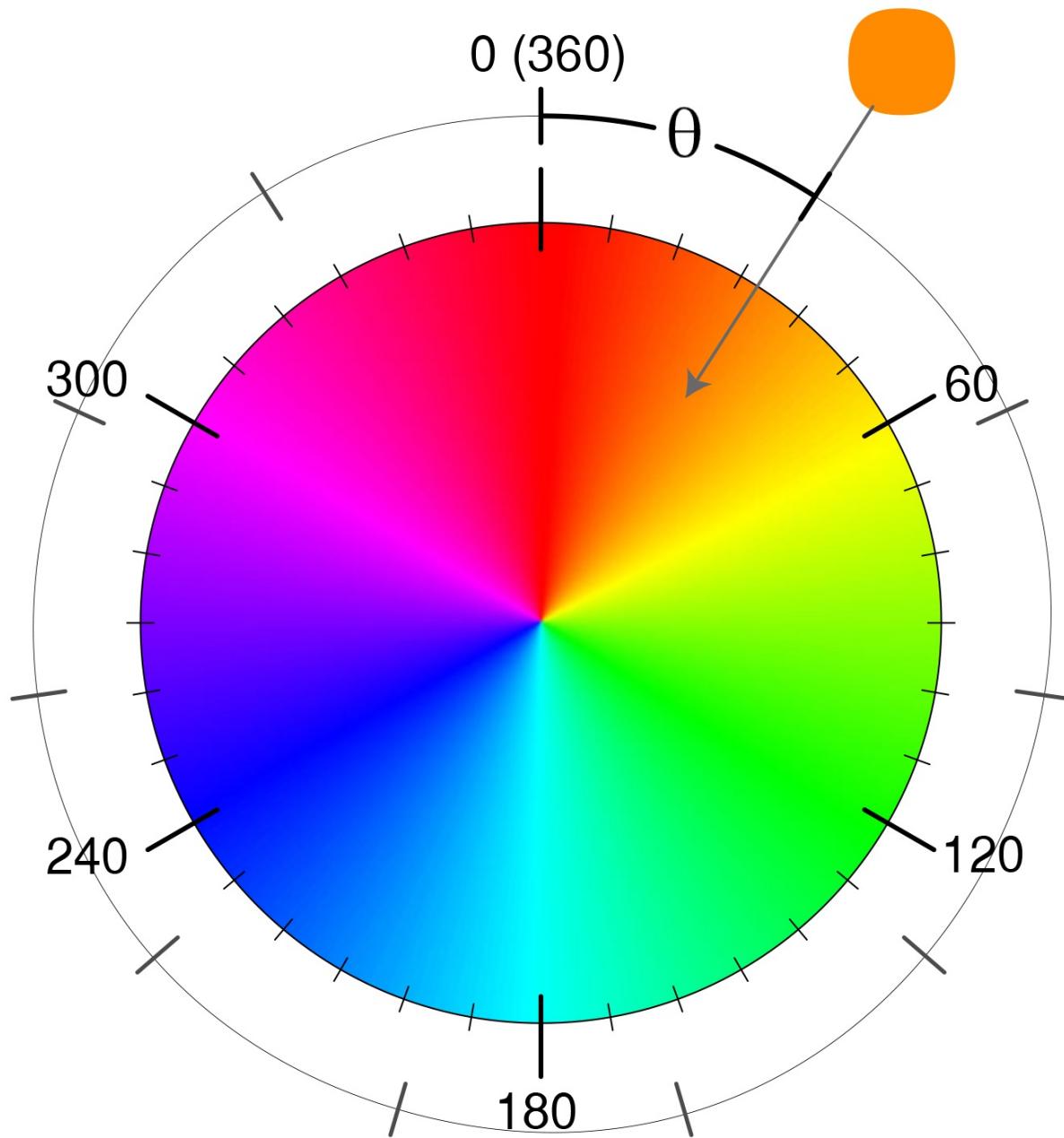
1. Factor = 1.000, Radius = 15
2. Factor = 0.400, Radius = 10
3. Factor = 0.200, Radius = 7



1.3.4. Диапазоны и Цвета

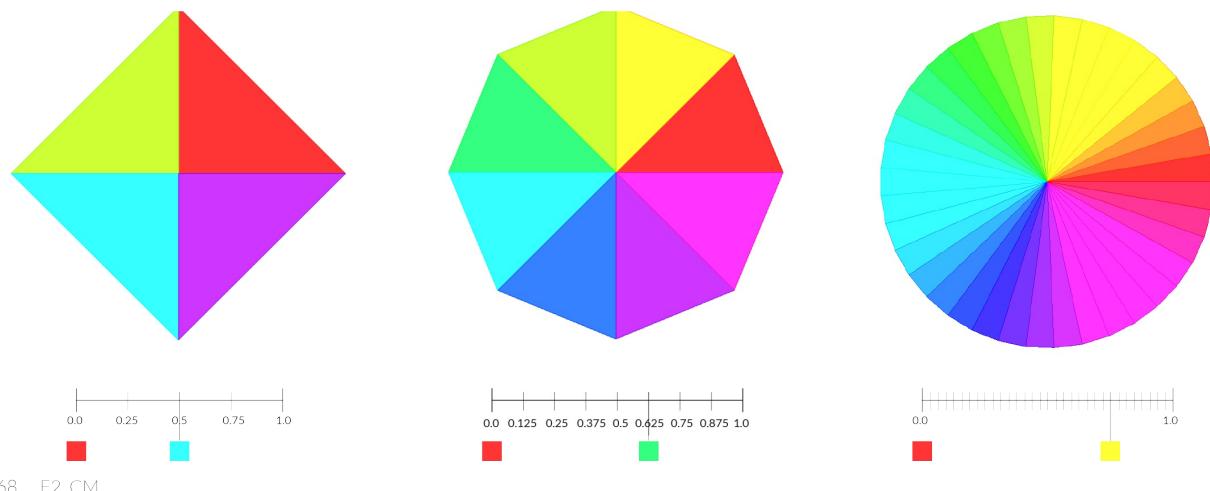
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Цветовой круг - это модель организации цветов, основанная на их тоне. В Grasshopper цвета могут определяться по значению их тона в диапазоне от 0.0 до 1.0. Диапазоны используются для определения диапазона всех возможных значений между числовым набором и нижней границей (A) и верхней границей (B).



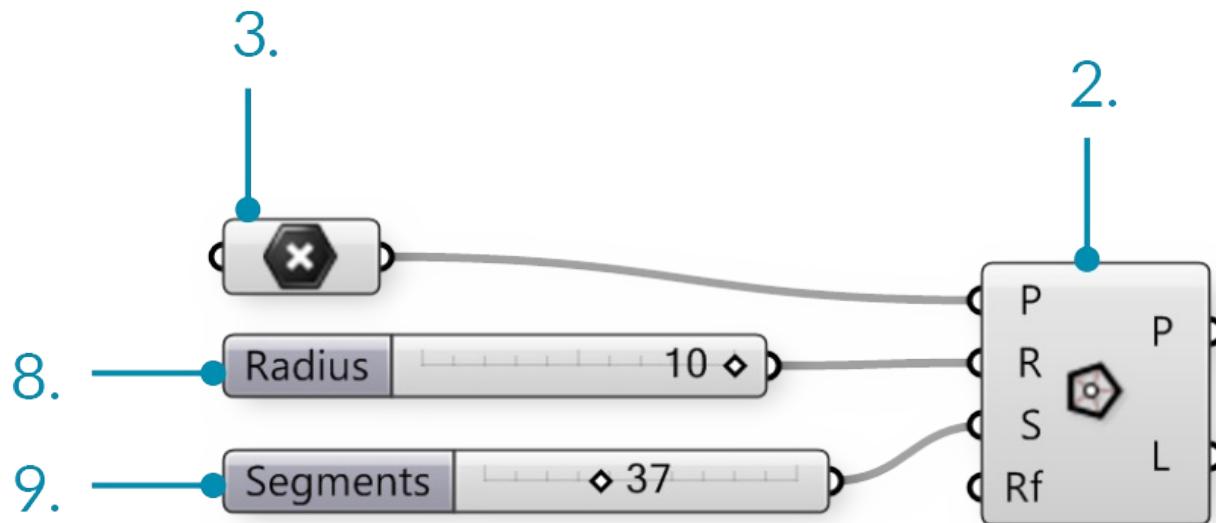
В цветовом круге, тон соответствует углу. Grasshopper взял эти 0-360 диапазоны и перенес их на значения от нуля до единицы.

Разделив диапазон Тона (0.0 - 1.0) на желаемое число сегментов, мы можем назначить значение тона для каждого сегмента, чтобы создать цветовой круг.

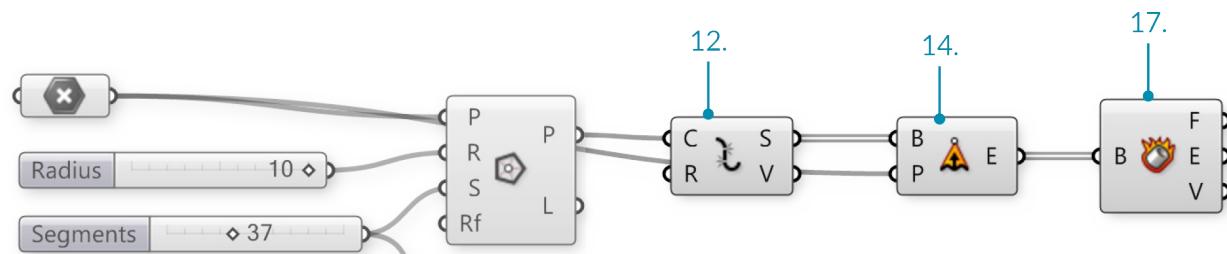


В этом примере, вы будете использовать домены Grasshopper и цветовые компоненты, чтобы создать цветовой круг с различным числом сегментов.

01.	Чтобы начать новое определение, нажмите Ctrl+N в Grasshopper	
02.	Зайдите в Curve/Primitive/Polygon – перетащите компонент Polygon на холст	
03.	Зайдите в Params/Geometry/Point – перетащите параметр Point на холст	
04.	Кликните правой клавишей мыши по компоненту Point и выберите set one point	
05.	Назначьте точку в пространстве модели.	
06.	Соедините параметр Point (Base Point) с входом Plane (P) компонента Polygon	
07.	Зайдите в Params/Input/Number Sliders – перетащите два слайдера Number Sliders на холст	
08.	Дважды кликните по первому слайдеру Number Sliders и установите следующее: Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10	
09.	Дважды кликните по второму слайдеру Number Sliders и установите следующее: Rounding: Integers Lower Limit: 0 Upper Limit: 100 Value: 37	
10.	Соедините Number Slider (Radius) с входом Radius (R) компонента Polygon При подсоединении слайдера к компоненту, автоматически изменится его имя на имя входа, к которому он подсоединен.	
11.	Соедините Number Slider (Segments) с входом Segments (S) компонента Polygon	

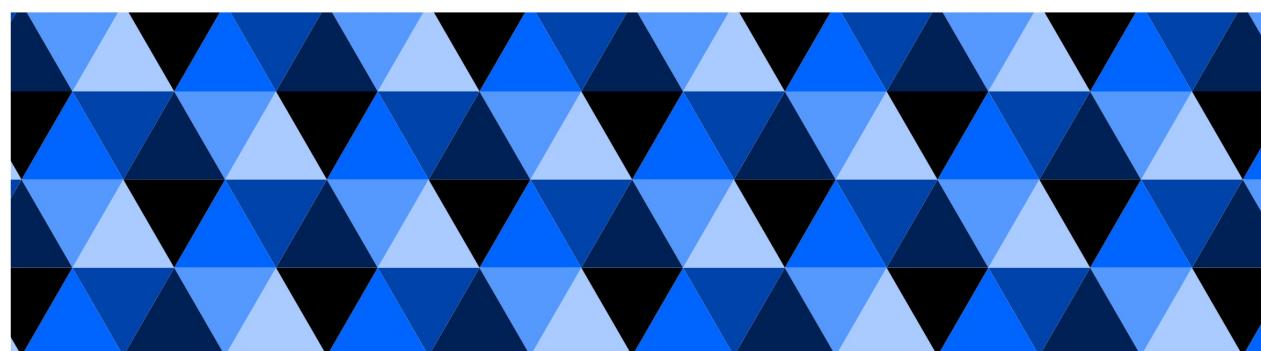
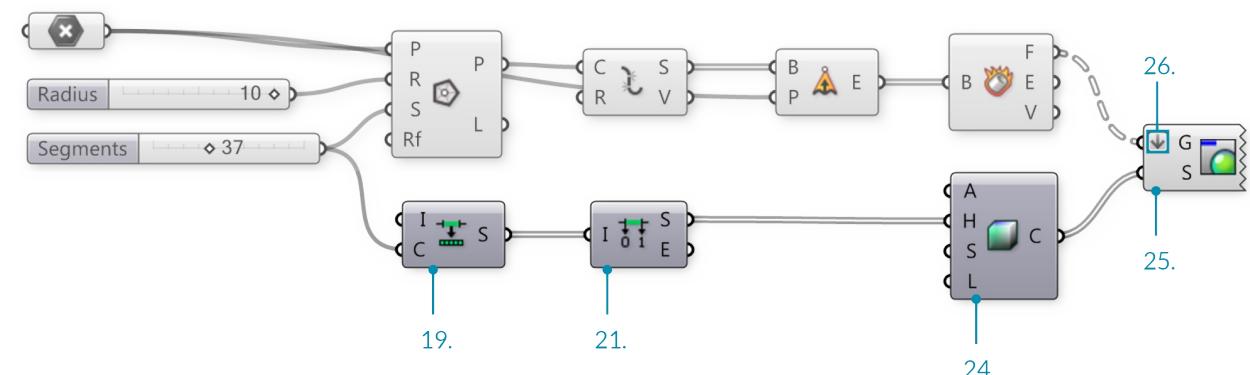


12.	Зайдите в Curve/Util/Explode – перетащите компонент Explode на холст.	
13.	Соедините выход Polygon (P) компонента Polygon с входом Curve (C) компонента Explode	
14.	Зайдите в Surface/Freeform/Extrude Point – перетащите компонент Extrude Point на холст	
15.	Соедините выход Segments (S) компонента Explode с входом Base (B) компонента Extrude Point	
16.	Соедините параметр Point (Base Point) со входом Extrusion Tip (P) компонента Extrude Point	
17.	Зайдите в Surface/Analysis/Deconstruct Brep – перетащите компонент Deconstruct Brep на холст	
18.	Соедините выход Extrusion (E) компонента Extrude Point с компонентом Deconstruct Brep (B)	

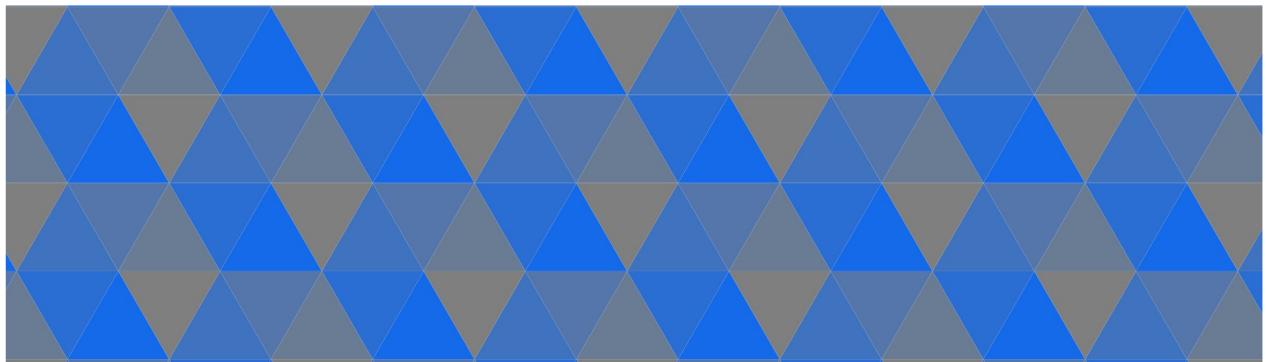


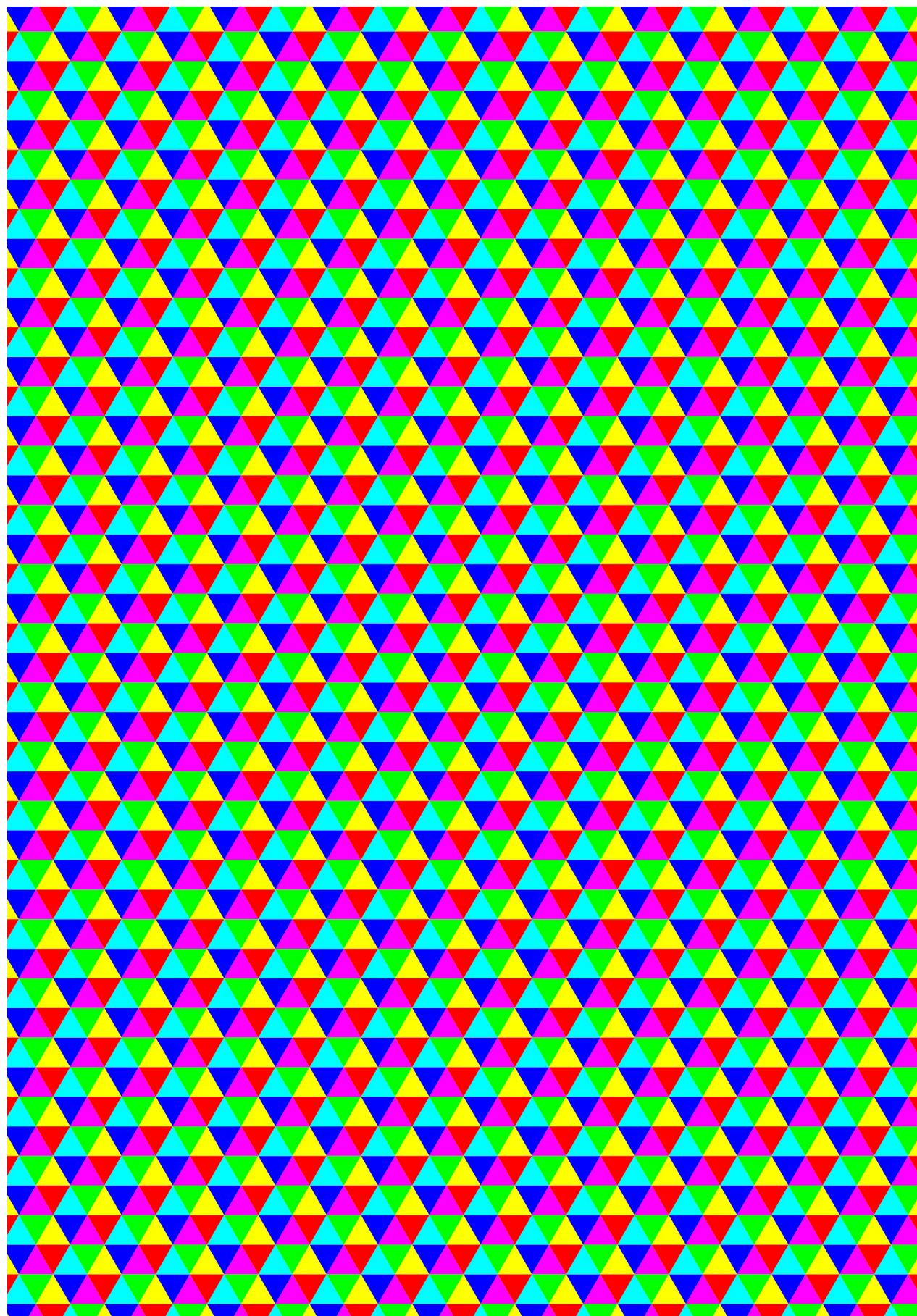
19.	Зайдите в Maths/Domain/Divide Domain – перетащите компонент Divide Domain Base Domain (I) автоматически установится от 0.0 до 1.0, как раз то, что нам нужно для этого упражнения	
20.	Соедините Number Slider (Segments) с входом Count (C) компонента Divide Domain	

21.	Зайдите в Math/Domain/Deconstruct Domain – перетащите компонент Deconstruct Domain	
22.	Соедините выход Segments (S) компонента Divide Domain с входом Domain (I) компонента Deconstruct Domain	
23.	Зайдите в Display/Colour/Colour HSL – перетащите компонент Colour HSL	
24.	Соедините выход Start (S) компонента Deconstruct Domain с входом Hue (H) компонента Colour HSL	
25.	Зайдите в Display/Preview/Custom Preview – перетащите компонент Custom Preview	
26.	Кликните правой клавишей мыши на входе Geometry (G) компонента Custom Preview и выберите Flatten Более подробно о Flatten см. 1-4 Проектирование с использованием Деревьев Данных	
27.	Соедините выход Faces (F) компонента Deconstruct Brep с входом Geometry (G) компонента Custom Preview	
28.	Соедините выход Colour (C) компонента Colour HSL с входом Shade (S) компонента Custom Preview	



Чтобы получить различные цветовые эффекты, попробуйте соединить компонент **Deconstruct Domain** с входами **saturation (S)** или **Luminance (L)** компонента **Colour HSL**.





1.3.5. Булевые и Логические Операторы

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

1.3.5.1. БУЛЕВЫЕ ЗНАЧЕНИЯ

Числовые переменные могут хранить целый ряд различных чисел. Булевые переменные могут хранить только два значения, обозначаемые как Да или Нет, Правда или Ложь, 1 или 0. Очевидно, мы никогда не сможем использовать булевые значения для выполнения расчетов из-за их ограниченного диапазона. Мы можем использовать булевые значения для определения условий.



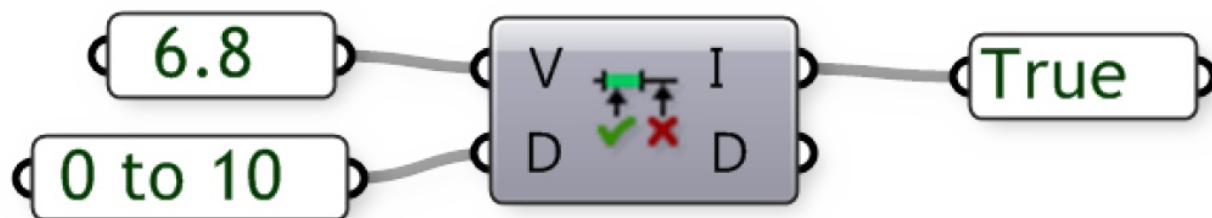
Параметр Boolean

В Grasshopper, Boolean могут использоваться несколькими способами. Параметр Boolean - это контейнер для одного или нескольких булевых значений, в то время как Boolean Toggle позволяет быстро изменить значение правды или лжи как входов.



Boolean Toggle - дважды кликните по переключателю булевых значений для изменения значения правда или ложь

Grasshopper также содержит объекты, которые проверяют условие и результат булевых значений. Например, компонент Includes позволяет проверить числовое значение на то, включено ли оно в диапазон.



Компонент Includes проверяет включено ли число 6.8 в диапазон 0 - 10. Возвращается булевое значение Правда.

1.3.5.2. ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Логические операторы, в своем большинстве, работают с булевыми значениями и очень логичны. Как вы помните, булевые значения могут иметь только два значения. Булевая математика была разработана Джорджем Булем (1815-1864) и сегодня она является ядром всей цифровой индустрии. Булевая алгебра дает нам инструменты анализа, сравнения и описания наборов данных. Хотя Буле изначально определил шесть операторов булевых значений, мы обсудим только три из них:

1. Not (Нет)
2. And (И)
3. Or (Или)

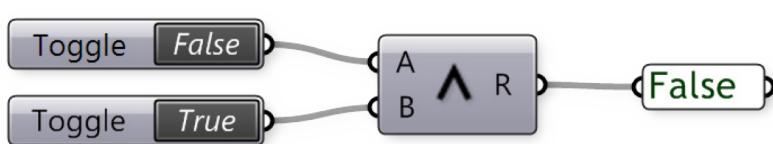
Оператор Nor немного выделяется среди других операторов, потому что ему не требуются два значения. Вместо этого, он просто преобразует одно значение в правильное. Представьте, у вас есть скрипт, который проверяет существование набора Блок определений в Rhino. Если Блок определение не существует, мы хотим проинформировать пользователя и прервать скрипт.



Оператор Grasshopper Nor

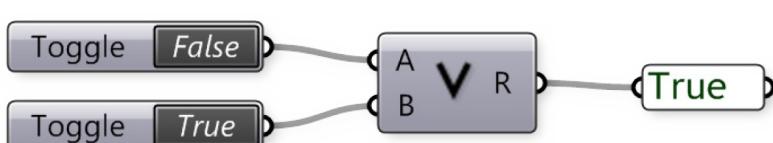
And и Or требуют наличие двух аргументов с обеих сторон. Оператору And требуется два аргумента True, чтобы определить значение как True. Оператору Or достаточно одного значения True.

Как видите, сложность в работе с логическими операторами не в теории, а в том, что происходит, когда вам требуется их большое количество для оценки чего-либо. Использование их вместе быстро приводит к запутанности, не говоря уже о проблемах с приоритетом операторов.



A	B	Result
True	True	True
True	False	False
False	True	False
False	False	False

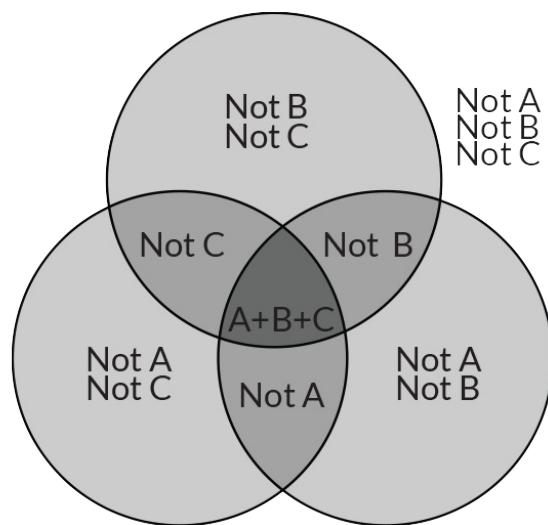
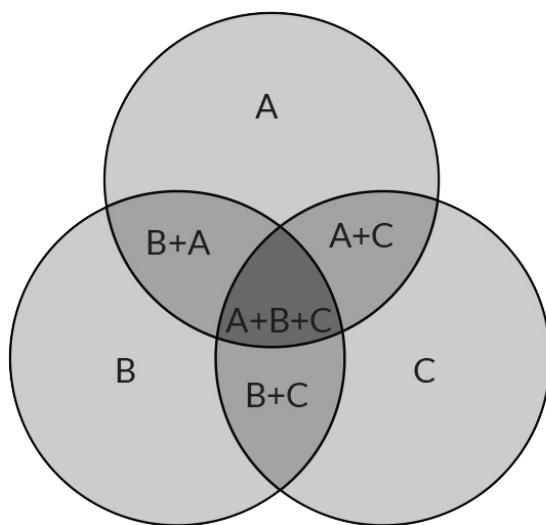
Оператор Grasshopper And



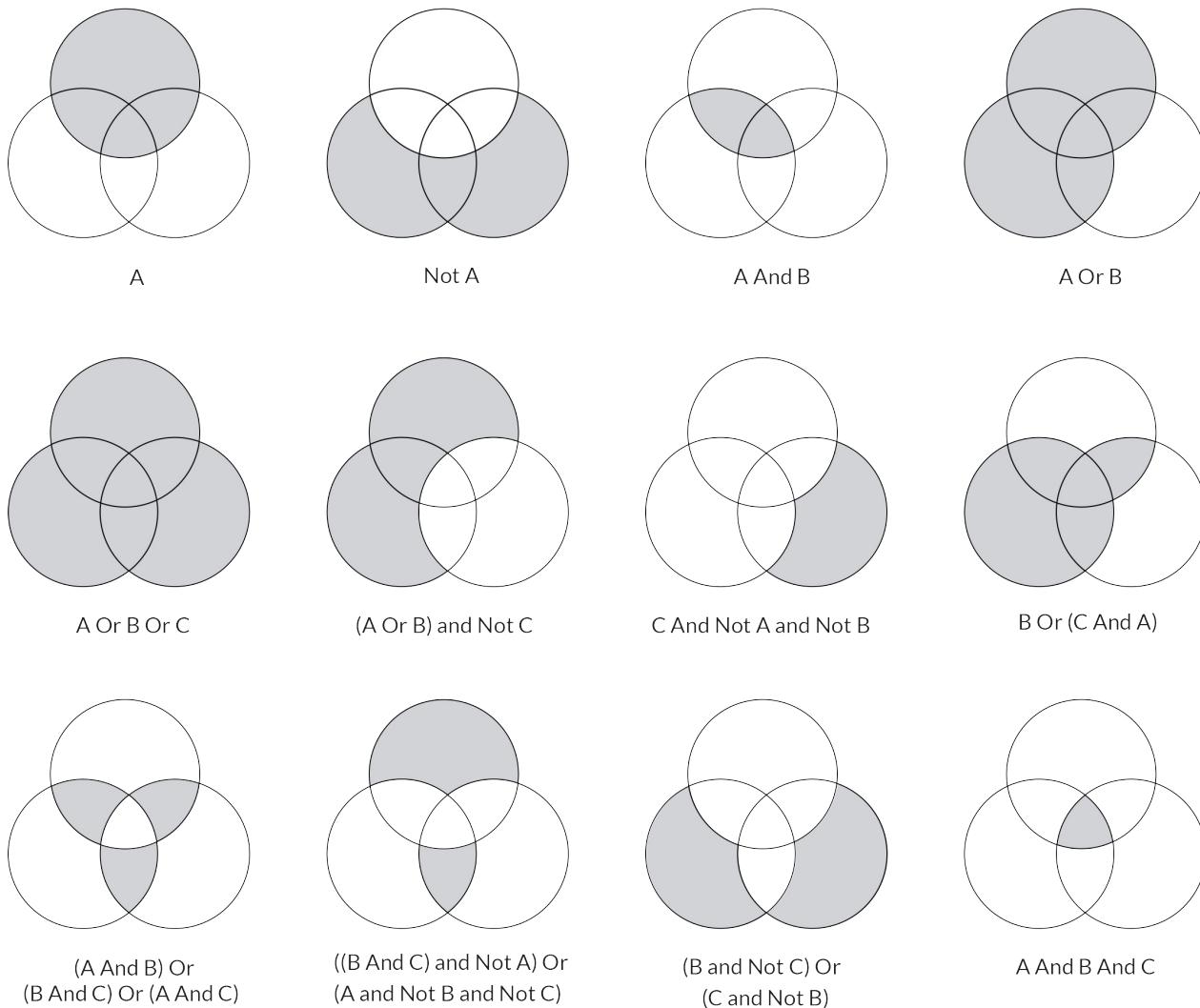
A	B	Result
True	True	True
True	False	True
False	True	True
False	False	False

Оператор Grasshopper Or

Хороший способ поработать над вашей булевой логикой - это использовать диаграммы Venn. Диаграмма Venn - это графическое представление наборов булевых значений, где каждый диапазон содержит (под)набор значений, которые обладают общими характеристиками. Самая известная из них - это диаграмма из трех кругов:

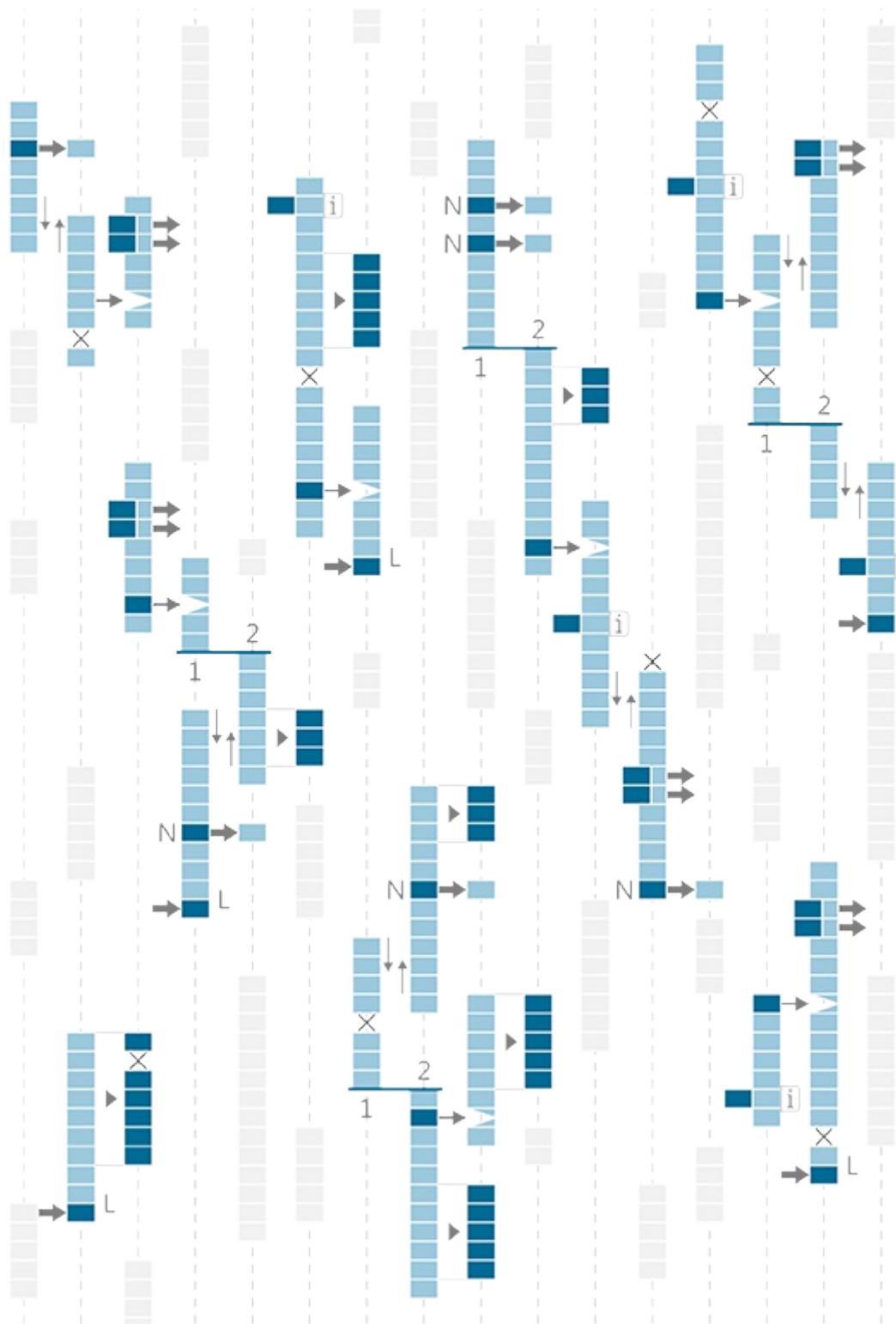


Каждый круговой диапазон содержит все значения, которые принадлежат этому набору; верхний круг, например, отмечает набор $\{A\}$. Каждое значение внутри этого круга определяет True для $\{A\}$ и каждое значение вне этого круга оценивает False для $\{A\}$. Цветовая кодировка диапазонов помогает нам изобразить булевое определение в программном коде:



1.4. Проектирование с использованием списков

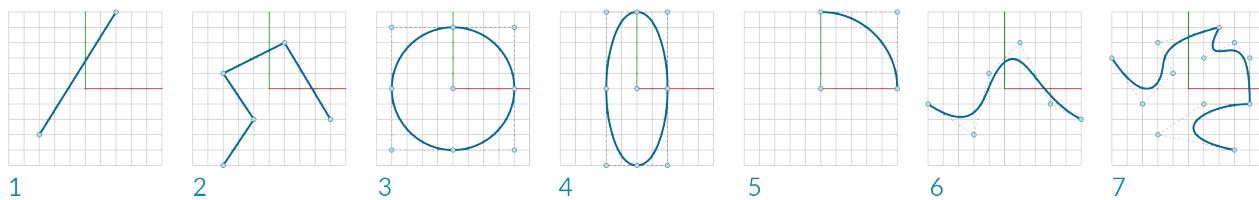
Одна из самых мощных характеристик Grasshopper - это способность быстро строить и изменять списки данных. Эта глава объяснит, как создавать, изменять и визуализировать список данных.



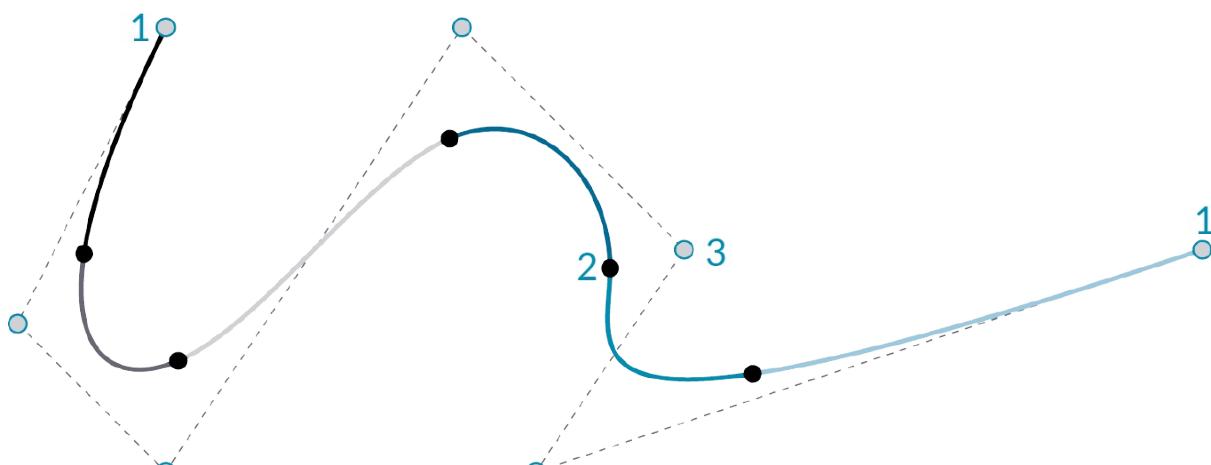
1.4.1. ГЕОМЕТРИЯ КРИВОЙ

NURBS (неоднородные рациональные Б-сплайны) - математическое представление, которое может точно смоделировать любую форму из простой 2D линии, круга, арки или коробки в самую сложную 3D произвольной формы органическую поверхность или тело. Благодаря своей гибкости и точности, NURBS модели могут использоваться в любом процессе от иллюстрации и анимации до производства.

Так как кривая является геометрическим объектом, она обладает некоторым числом характеристик, которые можно использовать для ее описания или анализа. Например, каждая кривая имеет начальную координату, и каждая кривая имеет конечную координату. Когда расстояние между этими двумя координатами равно нулю, кривая замкнута. Также, у каждой кривой есть некое число контрольных точек, если все эти точки расположены на одной плоскости, кривая, в целом, будет планарной. Некоторые характеристики применимы ко всей кривой, в целом, в то время как другие применимы к особым точкам на кривой. Например, планарность - это универсальная характеристика, в то время как тангенс векторы - это узкая характеристика. Также, некоторые характеристики применимы только к некоторым типам кривых. На данный момент мы обсудили некоторые компоненты Grasshopper из раздела Primitive Curve, такие как линии, круги, эллипсы и арки.



- 1. Линия
- 2. Полилиния
- 3. Круг
- 4. Эллипс
- 5. Арка
- 6. NURBS кривая
- 7. Поликривая



- 1. Конечная точка
- 2. Точка редактирования
- 3. Контрольная точка

1.4.1.1. КРИВЫЕ NURBS

Порядок: Порядок это положительное целое число. Обычно это число 1, 2, 3 или 5, но это может быть любое положительное целое число. Порядок кривой определяет степень влияния контрольных точек на кривую, где чем выше порядок, тем больше степень влияния. NURBS линии и полилинии обычно имеют порядок 1, NURBS круги - порядок 2, большая часть кривых произвольной формы порядок 3 или 5.

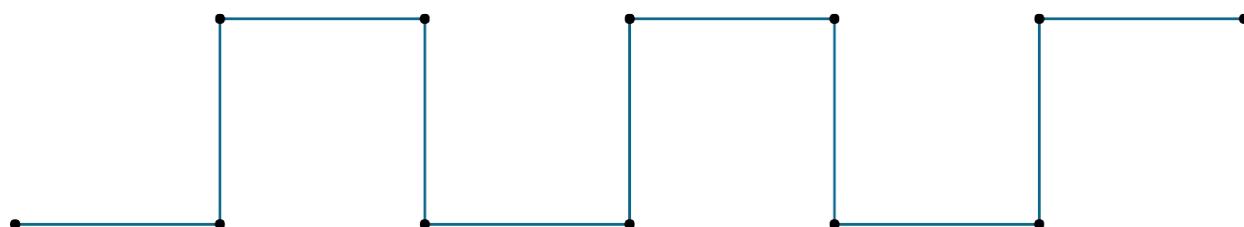
Контрольные точки: Контрольные точки - это списки точек, у которых порядок+1, по меньшей мере. Один из самых легких способов изменить форму NURBS кривой - переместить контрольные точки.

Вес: Контрольные точки имеют соответствующие им числа, называемые весом. Вес обычно положительное число. Когда контрольные точки кривой имеют одинаковый вес (обычно 1), кривая называется нерациональной, в противном случае, кривая называется рациональной. Большинство NURBS кривых являются нерациональными. Несколько NURBS кривых, таких как круги и эллипсы, всегда являются рациональными.

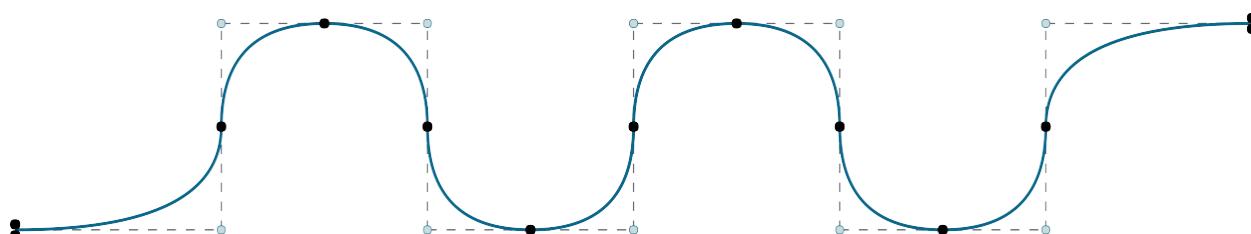
Узлы: Узлы - это список чисел (порядок+N-1), где N - число контрольных точек.

Точки редактирования: Точки на кривой определяются по средней величине узла. Точки редактирования похожи на контрольные точки, за исключением того, что они всегда располагаются на кривой и при перемещении одной точки редактирования, будет меняться форма всей кривой (перемещая одну контрольную точку, форма кривой будет меняться локально). Точки редактирования полезны, когда вам нужна точка на кривой, чтобы она проходила точно через определенный участок.

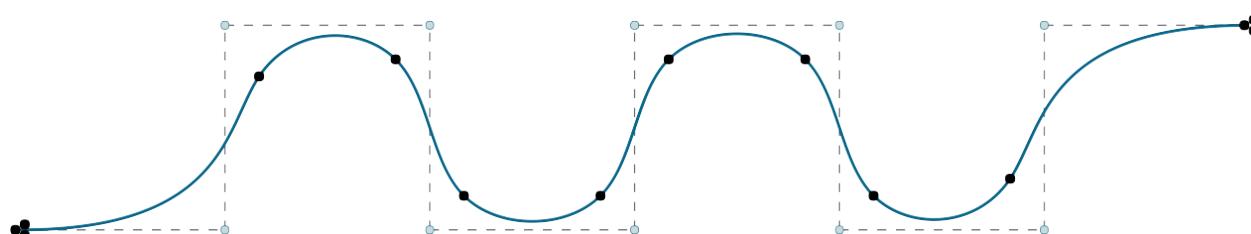
NNURBS кривые образуют узел как результат различного порядка:



AD¹ кривая ведет себя также как полилиния. AD¹ кривой есть узел для каждой контрольной точки.



D² NURBS кривые обычно используются только для аппроксимации арок и кругов. Сплайн пересекается с контрольным полигоном наполовину каждого сегмента.



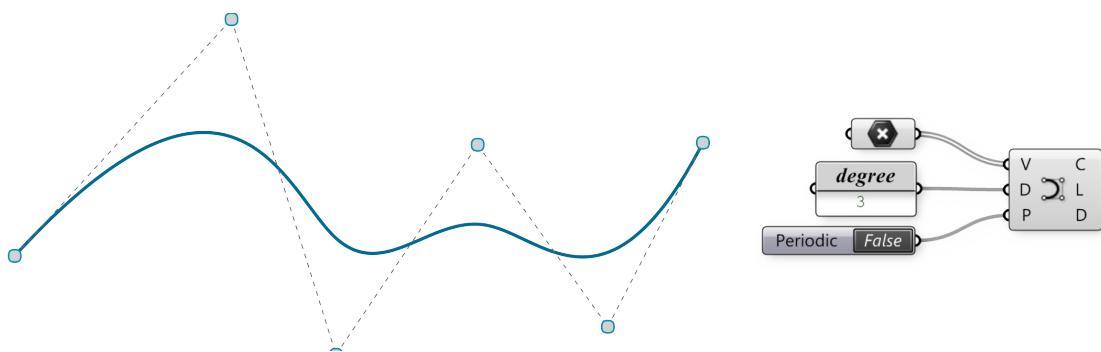
D³ это самый распространенный тип NURBS кривой и он включен по умолчанию в Rhino. Возможно вы знакомы с визуальной прогрессией сплайна, даже хотя узлы и появляются в странных местах.

1.4.1.2. КОМПОНЕНТЫ СПЛАЙН GRASSHOPPER

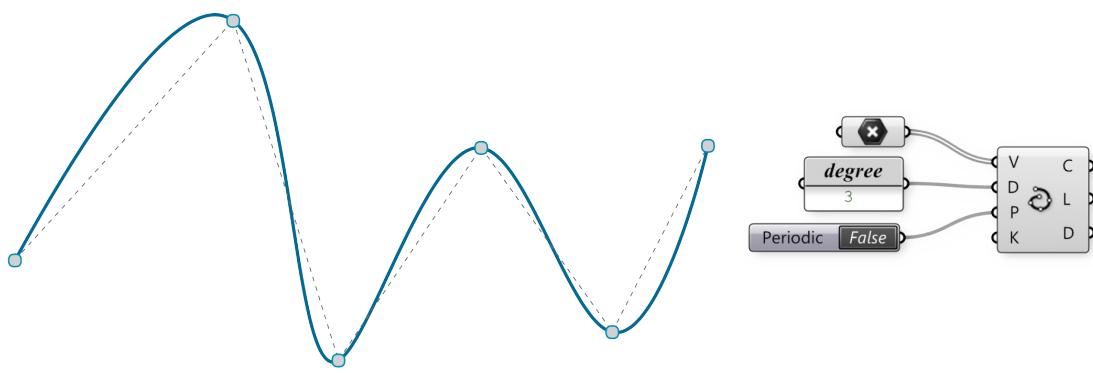
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Grasshopper обладает набором инструментов для передачи более продвинутых типов кривых Rhino, таких как NURBS кривые и поликривые. Эти инструменты можно найти во вкладке Curve/Splines.

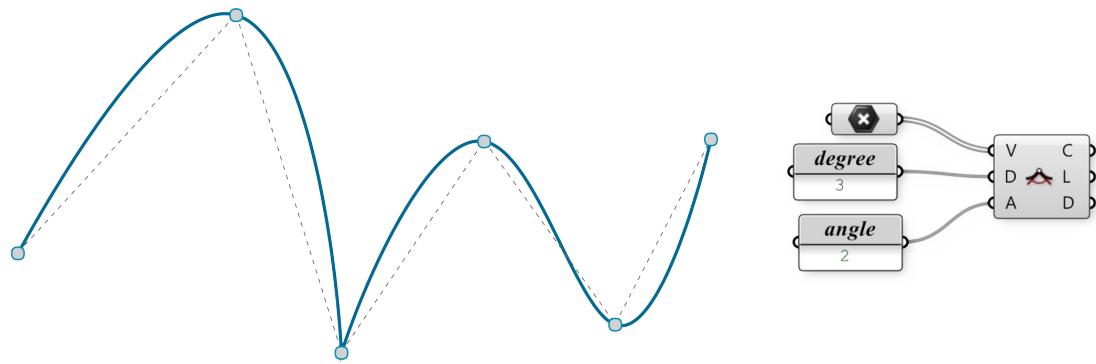
Nurbs Curve (Curve/Spline/Nurbs curve): Компонент Nurbs Curve создает NURBS кривую из контрольных точек. Вход V определяет эти точки, которые могут быть описаны косвенно путем выбора точек из Rhino или путем наследования изменяемых данных из других компонентов. Вход D компонента Nurbs Curve устанавливает порядок кривой.



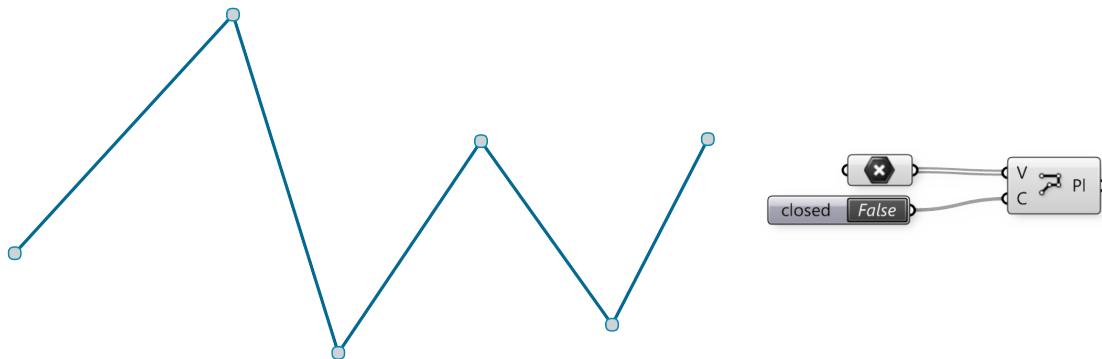
Interpolate Curve (Curve/Spline/Interpolate): Интерполированные кривые ведут себя немного по-другому, чем NURBS кривые. Вход V для компонента, похожего на компонент NURBS в том, что он требует особый набор точек для создания кривой. Тем не менее, при применении метода интерполированной кривой, полученная кривая будет проходить через эти точки, вне зависимости от порядка кривой. В компоненте NURBS кривой, мы можем добиться этого, только когда порядок кривой был установлен на 1. Также, как и компонент NURBS кривая, вход D определяет порядок итоговой кривой. Тем не менее, при применении этого метода, используются только нечетные числовые значения для входа порядка. Снова, вход P определяет, является ли кривая периодической (Periodic). Вы начнете видеть некий паттерн в выходах для многих компонентов кривой, в том плане, что выходы C, L и D указывают, в основном, итоговую кривую, длину и диапазон кривой соответственно.



Kinky Curve (Curve/Spline/Kinky Curve): Компонент kinky curve позволяет вам контролировать особые предельные значения угла, A, где кривая осуществит переход от узловой линии к гладкой, интерполированной кривой. Следует отметить, что вход A требует ввода в радианах.



Polyline (Curve/Spline/Polyline): Полилиния - это набор сегментов линии, соединяющих две или более точек, результирующая линия будет всегда проходить через ее контрольные точки, что схоже с Interpolated Curve. Как вышеуказанные типы кривых, вход V компонента Polyline указывает набор точек, которые будут определять границы каждого сегмента линии, составляющей полилинию. Вход C компонента определяет является ли полилиния открытой или закрытой кривой. Если расположение первой точки не совпадает с расположением последней точки, будет создаваться сегмент линии для закрытия loop. Выход компонента Polyline отличается от предыдущих примеров в том, что в результате получается только сама кривая.

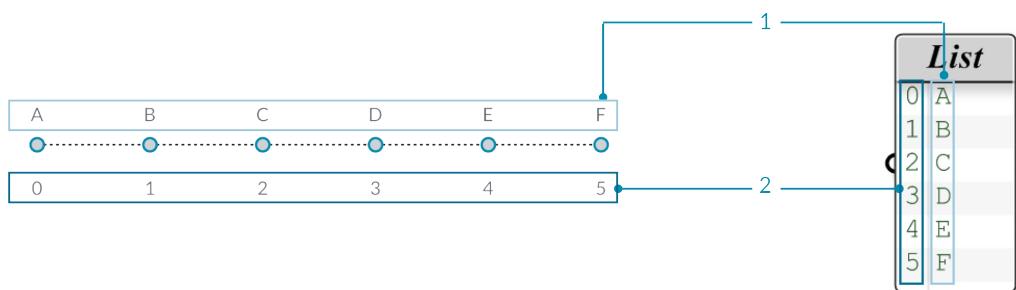


1.4.2. Что такое Список?

Полезно думать о Grasshopper в рамках потока, так как графический интерфейс спроектирован таким образом, чтобы данные входили в и выходили из разных типов компонентов. Тем не менее, именно данные определяют, какая информация будет входить и выходить из компонентов. Понимание того, как работать со списком данных критично для понимания плагина Grasshopper.

В целом, Grasshopper имеет два типа данных: постоянные и изменяемые. Даже при том, что типы данных имеют разные характеристики, обычно Grasshopper хранит эти данные в массиве, в списке переменных.

При сохранении данных в список, полезно знать расположение каждого элемента в этом списке, чтобы иметь возможность получить доступ или работать с определенными элементами. Положение элемента в списке называется индексом.



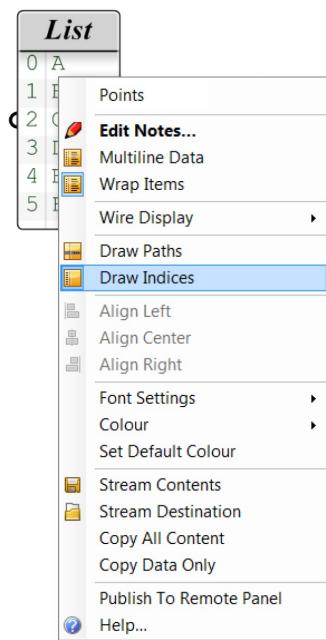
1. Элемент списка
2. Индекс

Единственное, что может показаться странным сначала, что первый индекс списка всегда 0, а не 1. Поэтому, когда мы говорим о первом элементе списка, мы, на самом деле, подразумеваем элемент, который соответствует индексу 0.

Например, если бы нам нужно было посчитать число пальцев на нашей правой руке, вы бы посчитали от 1 до 5. Тем не менее, если бы этот список сохранялся в массиве, тогда наш список считался бы от 0 до 4.

Заметьте, что у нас все равно 5 элементов в списке; дело только в том, что в массиве начинает счет с нуля. Элементы, хранящиеся в списке, не обязательно должны быть только числами. Это может быть любой тип данных, которые поддерживает Grasshopper, такой как точки, кривые, поверхности, mesh и т.д.

Часто, самый легкий способ, чтобы посмотреть какой тип данных хранится в списке, это подключить Text Panel (Params/Input/Panel) к выходу определенного компонента. По умолчанию, Text Panel автоматически показывает все индексы слева от панели и отображает элементы данных справа от панели. Индекс становится важным элементом, когда мы начинаем работать со списками. Вы можете включить и выключить индекс правым кликом по Text Panel (текстовой панели) и кликнув по элементу "Draw Indices" в свитке. Ну а теперь, давайте оставим вводные числа включенными на всех наших текстовых панелях.

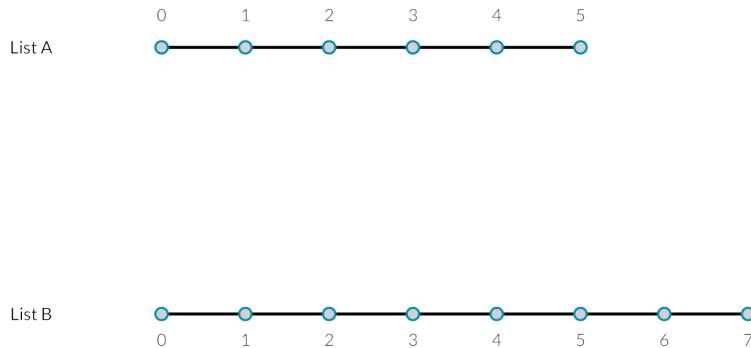


1.4.3. Соединение Потока Данных

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

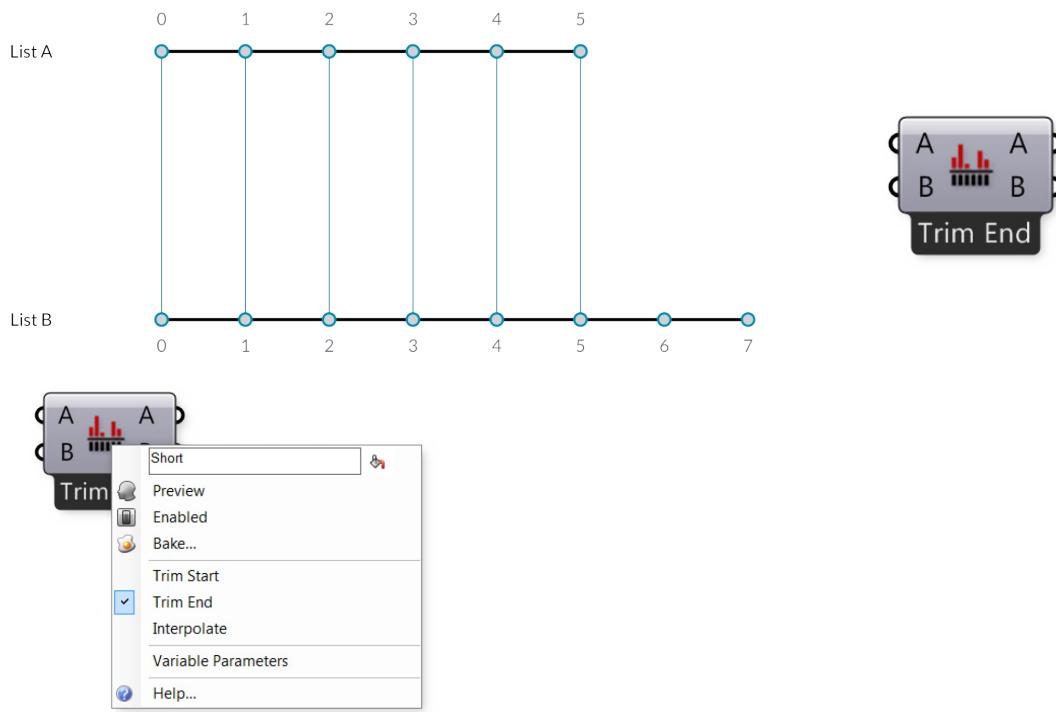
Соединение данных - это проблема, для которой нет одного решения. Это происходит, когда компонент имеет доступ к входам различного размера. Изменение алгоритма соединения данных может привести к значительно отличающимся результатам.

Представьте компонент, который создает сегменты линии между точками. Он будет иметь два параметра ввода, которые оба будут предоставлять координаты точек (Список А и Список Б):



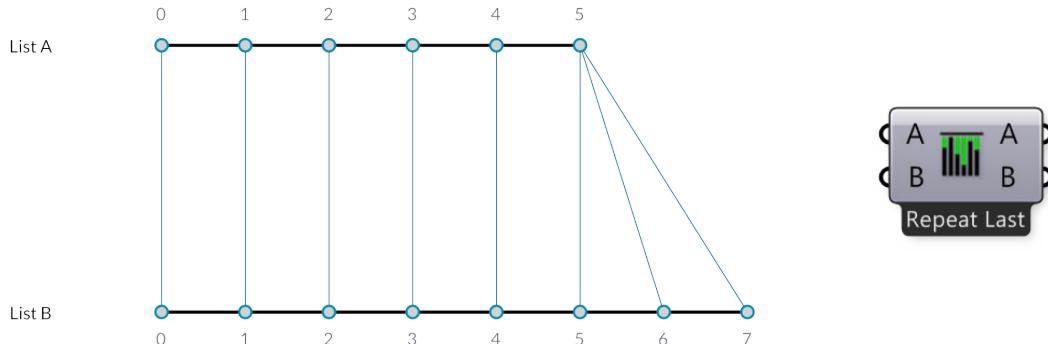
Как вы можете видеть, существуют разные способы рисования линий между этими наборами точек. В Grasshopper 0.9 есть три новых компонента для соединения данных, они находятся в разделе Sets/List panel: Shortest List (самый короткий список), Longest List (самый длинный список) и Cross Reference (перекрестные ссылки). Эти новые компоненты позволяют добиться большей гибкости внутри трех базовых алгоритмов соединения данных. Кликните правой клавишей мыши по каждому компоненту и вы сможете выбрать опцию соединения данных из меню.

Самый простой способ - соединить входы один к одному пока один из потоков не исчерпает себя. Этот алгоритм называется “**Shortest List**” (самый короткий список):

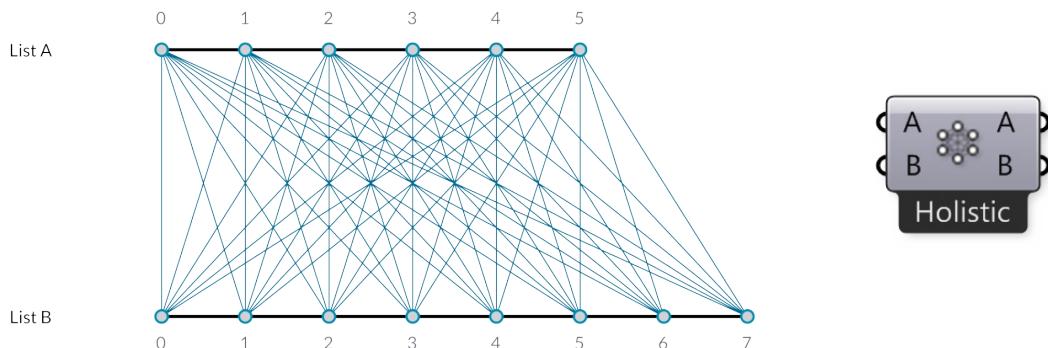


Выберите опцию алгоритма соединения из меню компонента правым кликом мыши по компоненту.

Алгоритм “**Longest List**” соединяет входы пока все потоки не иссякнут. Это поведение включено по умолчанию у всех компонентов:

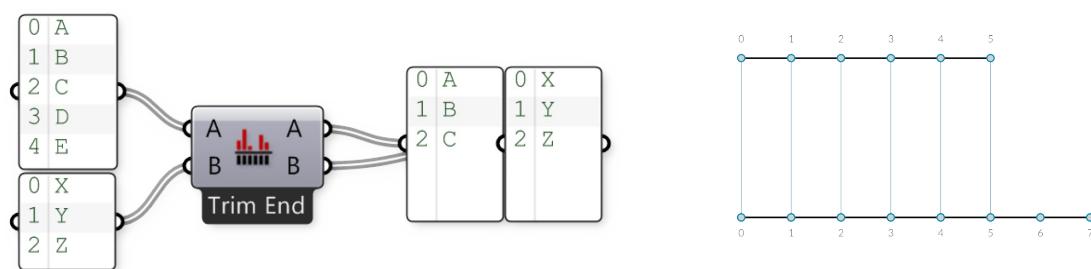


И последнее, метод “**Cross Reference**” создает все возможные соединения:

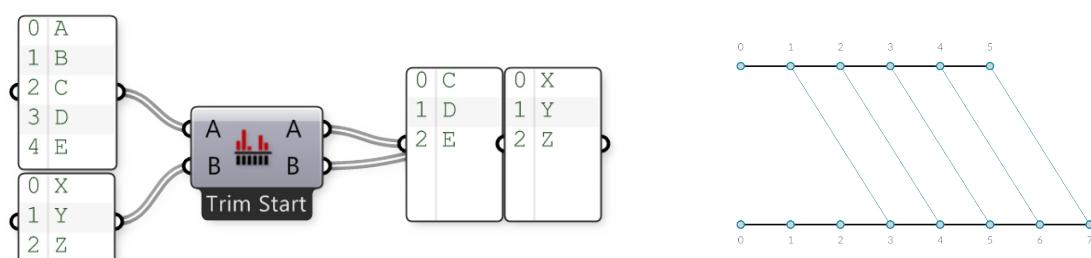


Это может быть потенциально опасно, так как создается невероятно огромное количество выходов. Проблема усложняется еще тем, что вовлекается больше параметров ввода и наследование изменяемых данных начинает увеличивать данные, при этом логика остается той же.

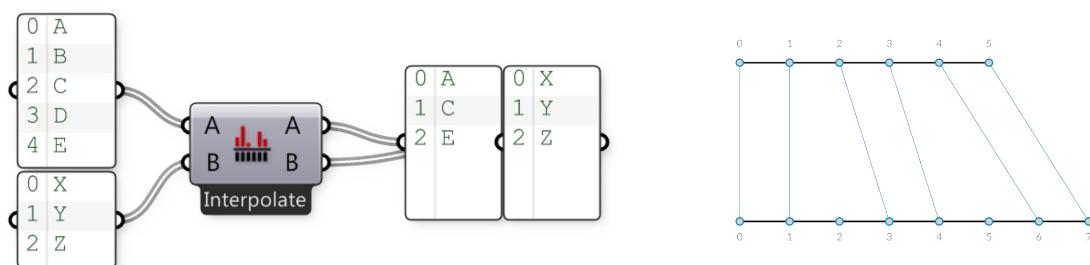
Давайте более подробно разберем компонент **Shortest List**:



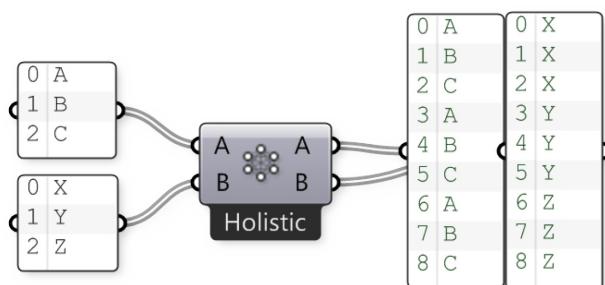
Тут у нас имеются два входных списка {A,B,C,D,E} и {X,Y,Z}. Используя опцию Trim End, два последних элемента в первом списке не учитываются, поэтому списки становятся одинаковой длины.



Используя опцию Trim Start, два первых элемента в первом списке не учитываются, поэтому списки становятся одинаковой длины.



Опция Interpolate пропускает второй и четвертый элементы в первом списке. Теперь давайте посмотрим на компонент Cross Reference:

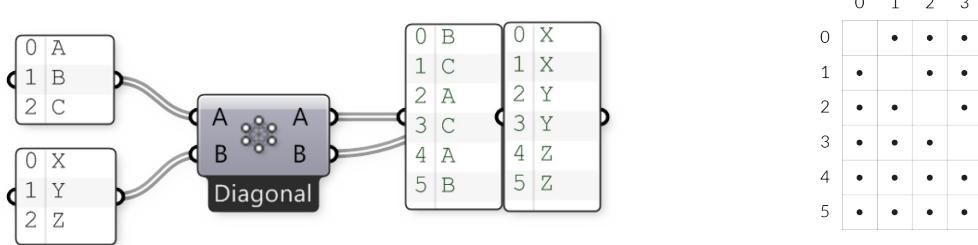


Тут у нас имеются два входных списка {A,B,C} и {X,Y,Z}. Обычно Grasshopper провел бы итерацию этих списков и рассмотрел бы только комбинации {A,X}, {B,Y} и {C,Z}. Тем не менее, тут имеются еще шесть комбинаций, которые обычно не рассматриваются, а именно: {A,Y}, {A,Z}, {B,X}, {B,Z}, {C,X} и {C,Y}. Как вы можете видеть, выход компонента Cross Reference таков, что присутствуют все девять сочетаний.

Мы можем отметить поведение данных перекрестных ссылок используя таблицу. Ряды представляют первый список элементов, колонки второй. Если мы создадим все возможные сочетания, таблица будет иметь точку в каждой отдельной ячейке, так как каждая ячейка представляет собой уникальную комбинацию двух индексов исходных списков.

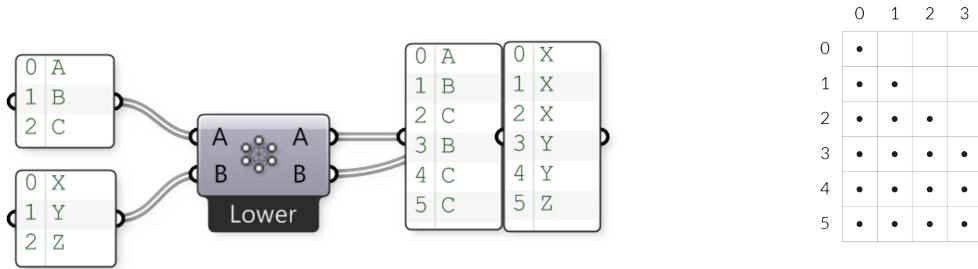
	0	1	2	3
0	•	•	•	•
1	•	•	•	•
2	•	•	•	•
3	•	•	•	•
4	•	•	•	•
5	•	•	•	•

Иногда, тем не менее, вы не хотите, чтобы происходили все возможные сочетания. Иногда вы хотите исключить некоторые области потому, что они приведут к бессмысленным или неправильным вычислениям. Общий принцип исключения - это игнорировать все ячейки, расположющиеся по диагонали таблицы. Изображение выше представляет "холистическое" соединение, где "диагональная" опция (доступна из меню компонента Cross Reference) имеет пропуски в {0,0}, {1,1}, {2,2} и {3,3}. Если мы применим это к нашим примерам {A,B,C}, {X,Y,Z}, нам следует ожидать, что мы не увидим комбинации для {A,X}, {B,Y} и {C,Z}:

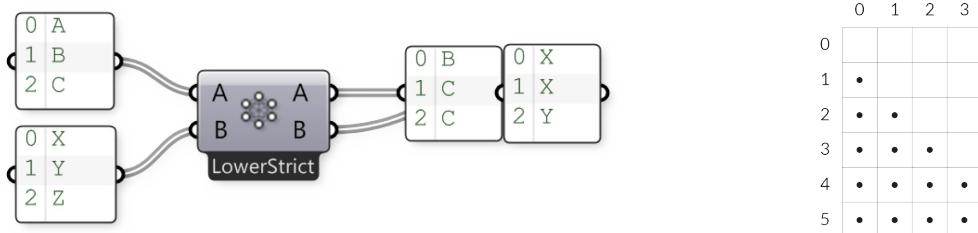


Правило, которое применяется к "диагональному" соединению, следующее: "Пропустить все подстановки, где все элементы имеют такой же индекс списка". "Случайное" соединение - это то же самое, что "диагональное" соединение в случае двух вводных списков, но правило несколько отличается: "Пропустить все подстановки, где любые два элемента имеют такой же индекс списка".

Четыре оставшихся алгоритма соединения - это все вариации на ту же тему. Соединение "нижний треугольник" применяет правило: "Пропускать все подстановки, где индекс элемента меньше, чем индекс элемента в следующем списке", приводя в пустому треугольнику, но с элементами по диагонали.



Соединение "нижний треугольник" (строгий) идет на один шаг вперед и также устраняет элементы по диагонали:



"Верхний Треугольник" и "Верхний Треугольник (строгий)" - это зеркальное изображение двух предыдущих алгоритмов, результат которых пустой треугольник на другой стороне диагональной линии.

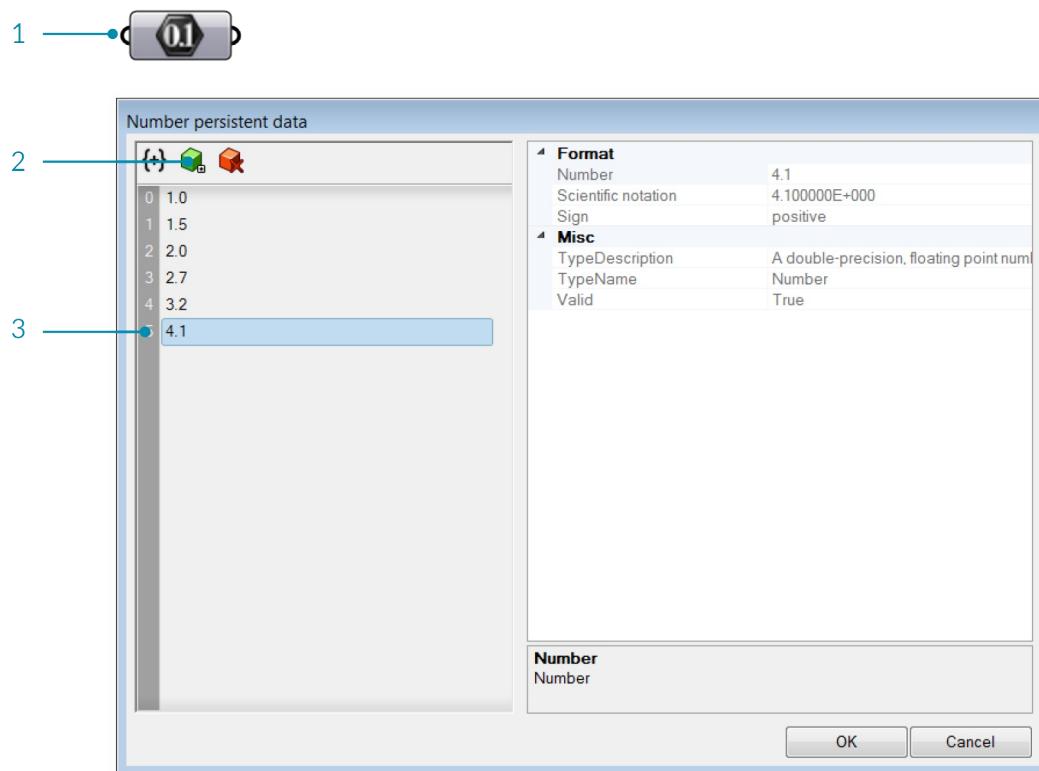
1.4.4. Создание Списков

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Существуют много различных способов генерировать списки в Grasshopper. Ниже, мы рассмотрим несколько разных способов генерирования списков и затем посмотрим на то, как данные могут использоваться для передачи информации в видовое окно посредством визуализации.

1.4.4.1. СОЗДАНИЕ СПИСКА ВРУЧНУЮ

Возможно, самый простой способ создать список (и один из самых просматриваемых методов) - это вручную прописать список значений в параметр. Использование этого метода накладывает ответственность на пользователя, потому что этот метод основывается на прямом вводе данных пользователем (т.е. постоянных данных) для создания списка. Чтобы изменить значение списка, пользователь должен вручную прописать это в каждом отдельном значении, что может представлять некоторую сложность, если в списке много записей. Существует несколько способов создать список вручную. Первый способ - использовать параметр Number. Кликните правой клавишей мыши по параметру Number и выберите "Manage Number Collection".



1. Кликните правой клавишей мыши по компоненту Number, чтобы открыть Number Collection Manager.
2. Кликните по иконке Add Item, чтобы добавить число к списку.
3. Дважды кликните по числу, чтобы изменить его значение.

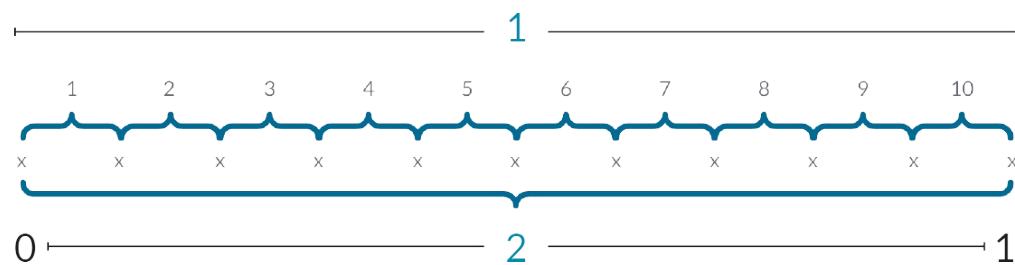
Другой способ - вручную ввести элементы списка на панель. Убедитесь, что снята отметка "Multiline Data".

{ 0 }	
0	1
1	1.5
2	2
3	2.7
4	3.2
5	4.1

1.4.4.2. КОМПОНЕНТ RANGE

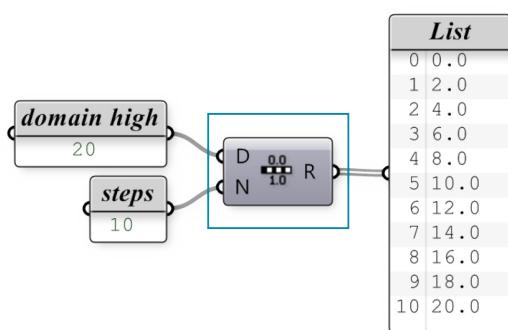
Компонент Range можно найти в разделе Sets/Sequence/Range, он создает список равнотстоящих чисел между меньшими и большими значениями, называемыми Domain (диапазон). Диапазон (также иногда называемый интервал) - это каждое возможное число между двумя числовыми экстремумами.

Компонент Range разделяет числовой диапазон на равные сегменты и выдает список значений.



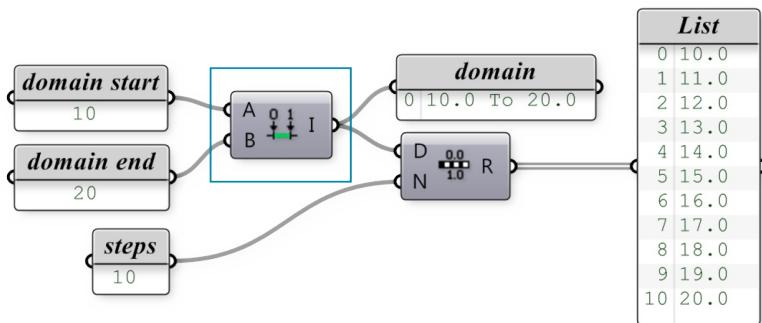
1. Число шагов = 10
2. Диапазон от 0 до 1
3. Общее число точек = 11

В примере ниже, числовой диапазон определялся как каждое возможное число между 0 и 20. Компонент Range берет этот диапазон и разделяет на число шагов (в данном случае 10). Итак, мы получим 10 равнотстоящих сегментов. Компонент Range выдает список значений. Из-за того, что он сохраняет первое и последнее значение в списке, выход компонента Range всегда на один больше, чем число шагов. В примере выше, мы создали 10 шагов, поэтому компонент Range выдает 11 значений.



Создайте список, используя компонент Range, указав диапазон и число шагов.

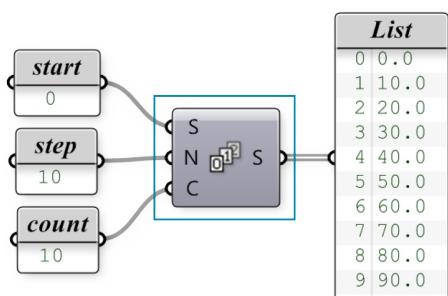
Вы могли заметить кое-что странное в только что созданной настройке. Мы знаем, что диапазон всегда определяется двумя значениями (наибольшим и наименьшим). Тем не менее, в нашем определении мы просто подключили одно значение ко входу диапазона. Чтобы избежать ошибок, Grasshopper делает предположение, что вы пытаетесь определить диапазон между нулем и каким-либо другим числом (значение нашего слайдера). Чтобы создать диапазон между двумя числами, который начинается не с нуля, мы должны использовать компонент Construct Domain, чтобы указать диапазон.



Чтобы создать Range из диапазона, который не начинается с нуля, используйте компонент Construct Domain.

1.4.4.3. КОМПОНЕНТ SERIES

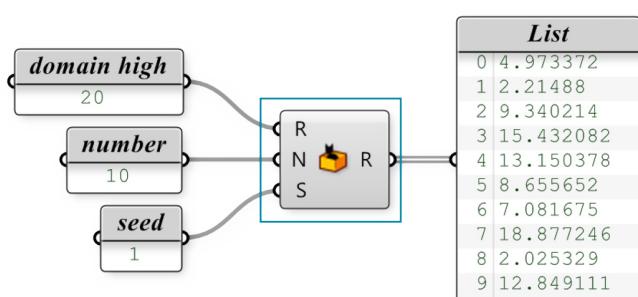
Компонент Series похож на компонент Range тем, что тоже создает списки чисел. Тем не менее, компонент Series отличается, потому что создает набор дискретных чисел, основывающихся на начальном значении, размере шага и числе значений в последовательности.



Компонент Series создает список, основанный на начальном значении, значении шага и количестве значений в списке.

1.4.4.4. КОМПОНЕНТ RANDOM

Компонент Random (Sets/Sequence/Random) может быть использован для генерации списка псевдо случайных чисел. Их называют "псевдо" случайными, потому что числовая последовательность уникальна, но стабильна для каждого значения seed. Таким образом, вы можете сгенерировать полностью новый набор случайных чисел изменения значение seed (S вход). Диапазон, как в предыдущем примере, это определяемый интервал между двумя числовыми экстремумами.

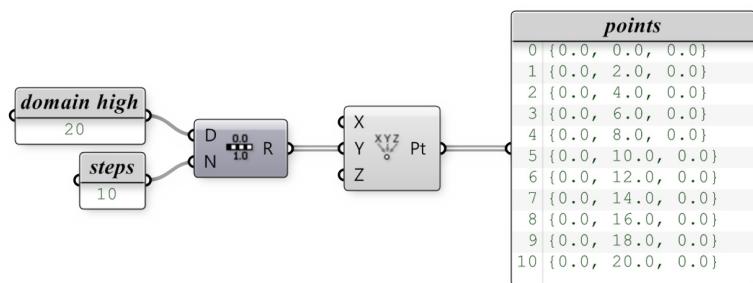


1.4.5. Визуализация Списка

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Понимание работы со списками в Grasshopper может быть трудно без возможности видеть как данные перетекают из одного компонента в другой. Существует несколько способов визуализации списков, которые могут помочь понять и работать с данными.

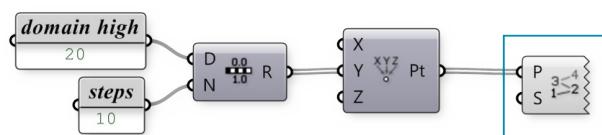
Существует много разных способов визуализации списка данных. Самый распространенный способ - это создать некую геометрию со списком данных. Затем соединяя выход R компонента Range со входом Y компонента Construct Point, мы сможем увидеть массив точек в направлении Y.



Давайте посмотрим на некоторые компоненты, которые могут помочь нам понять данные.

1.4.5.1. КОМПОНЕНТ POINT LIST

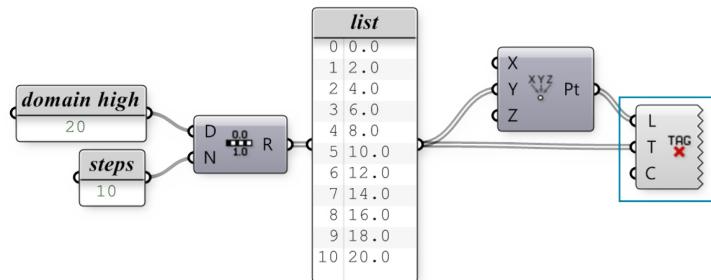
Компонент Point List - это невероятно полезный инструмент для визуализации порядка в наборе точек в списке. Главным образом, компонент Point List размещает число индекса элемента рядом с точкой геометрии в видовом окне. Вы также можете указать хотите ли вы или нет указывать числовые тэги, соединительные линии или размер текстовых тэгов.



Вы можете визуализировать порядок набора точек, используя компонент Point List.

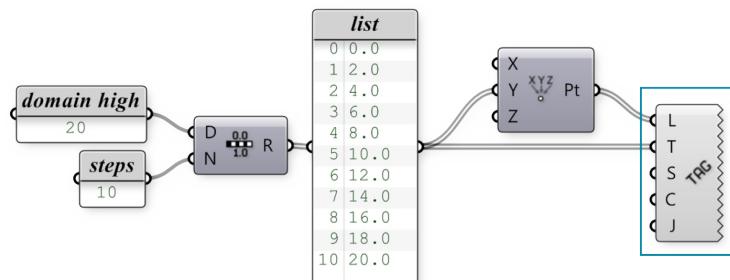
1.4.5.2. КОМПОНЕНТ TEXT TAGS

Компонент text tag позволяет отобразить небольшие текстовые типы данных (набор знаков ASCII) в видовом окне как элементы обратной связи. Текст и расположение определяются как параметры ввода. Когда текстовые тэги запекаются в сцене, они превращаются в Text Dots (текстовые точки). Другой занимательный момент, касающийся Текстовых Тэгов, они независимы от видового окна - это значит, что тэги всегда обращены к камере (включая вид перспективы) и они всегда остаются того же размера на экране, независимо от ваших настроек масштаба.



Вы можете визуализировать любую информацию текстовых типов данных в видовом окне, используя компонент Text Tag. В этом наборе мы решили отобразить значение каждой точки поверх расположения каждой точки. Мы могли бы назначить любой текст для отображения.

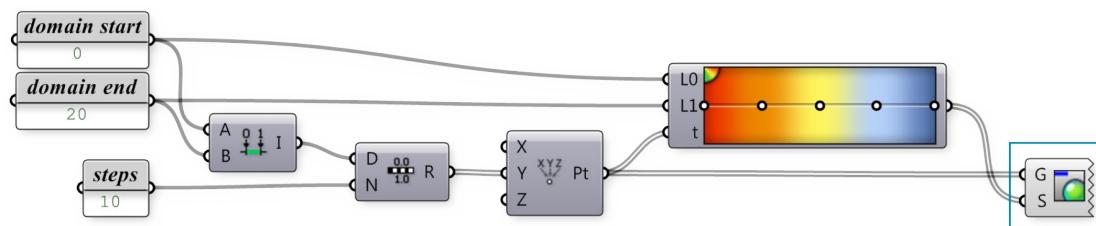
Компонент Text Tag 3d работает так же как компонент Text Tag. Они отличаются тем, что когда объекты Text Tag 3d запекаются в сцене, они становятся Текстовыми объектами в Rhino. Масштаб шрифта Text Tag 3d также можно контролировать через вход (который недоступен в компоненте Text Tag).



Вы также можете использовать компонент Text Tag 3d для визуализации информации подобно Текстовому объекту в Rhino.

1.4.5.3. COLOR

Еще одна из вещей, которые мы можем сделать для визуализации списка данных, - это присвоить геометрии цвет. У Grasshopper ограниченные возможности рендера, но мы можем контролировать простые OpenGL настройки, такие как цвет, отраженный цвет, прозрачность и т.д. Значение L0 представляет нижнюю границу (левая сторона) градиента, в то время как значение L1 представляет верхнюю границу (правая сторона). Эти значения совпадают с началом и концом нашего диапазона. Т-значения - это элементы списка, который будет перенесен где-то внутри диапазона L0 и L1. Выход градиента - это список цветовых значений RGB, которые соответствуют каждой точке в нашем списке. Кликните правой клавишей мыши по Gradient, чтобы настроить одну из преднастроек градиента или определить ваш собственный, используя цветовые точки.





1. Точки
2. Список точек
3. Текстовые тэги
4. Text Tag 3D
5. Предпросмотр настроенного цвета

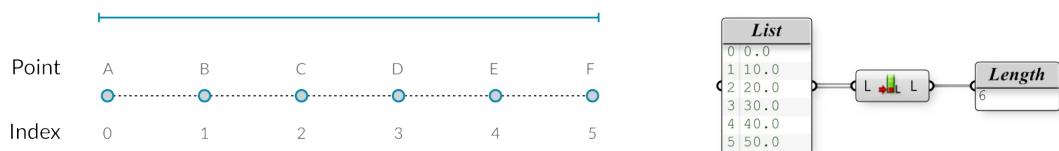
1.4.6. Управление Списком

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Одна из самых мощных характеристик Grasshopper - это способность быстро строить и изменять различные списки данных. Мы можем хранить много различных типов данных в списке (числа, точки, векторы, кривые, поверхности, брэпы и т.д.), существуют полезные для этого инструменты в разделе Sets/List.

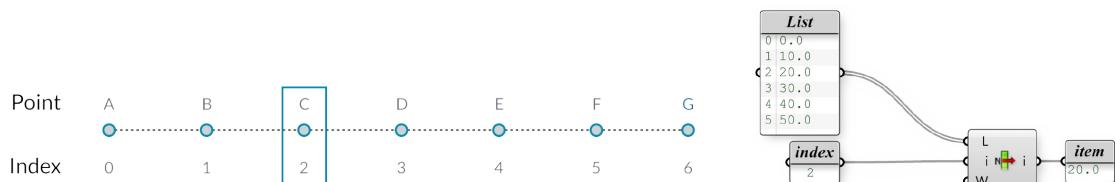
1.4.6.1. ДЛИНА СПИСКА

Компонент List Length (Sets/List/List Length) измеряет длину списка. Из-за того, что наши списки всегда начинаются с нуля, самый большой возможный индекс в списке равняется длине списка минус один. В этом примере, мы соединили наш базовый Список с входом L компонента List Length, показывая 6 значений в списке.



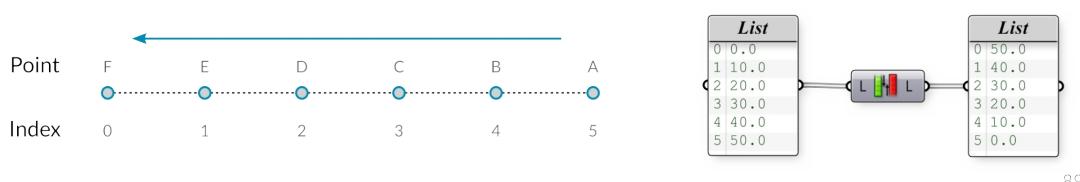
1.4.6.2. КОМПОНЕНТ LIST ITEM

Наш список вводится в компонент List Item (Sets/List/List Item), чтобы выносить особые элементы данных изнутри массива данных. При получении индивидуальных элементов списка, мы должны указать i вход, который соответствует индекс числу, которое мы хотели бы извлечь. Мы можем внести одно целое или список целых чисел в i вход в зависимости от того, как много элементов мы хотели бы извлечь. Вход L определяет базовых список, который мы будем анализировать. В этом примере, мы установили i вход на 2, так что компонент List Item выдает элементы данных, связанных с 3-м вводным числом в нашем списке.



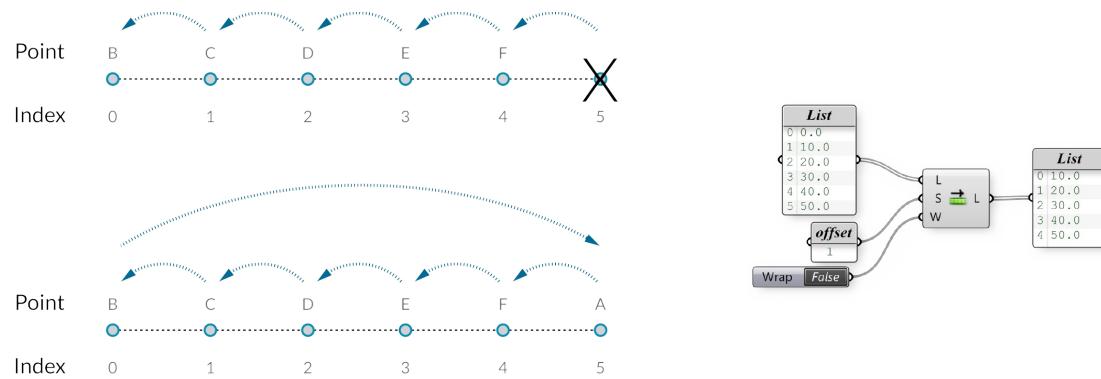
1.4.6.3. КОМПОНЕНТ REVERSE LIST

Мы можем инвертировать порядок в нашем списке, используя компонент Reverse List (Sets/ List/Reverse). Если мы введем список возрастающих чисел от 0.0 до 50.0 в компонент Reverse List; выход выдаст убывающий список от 50.0 до 0.0.



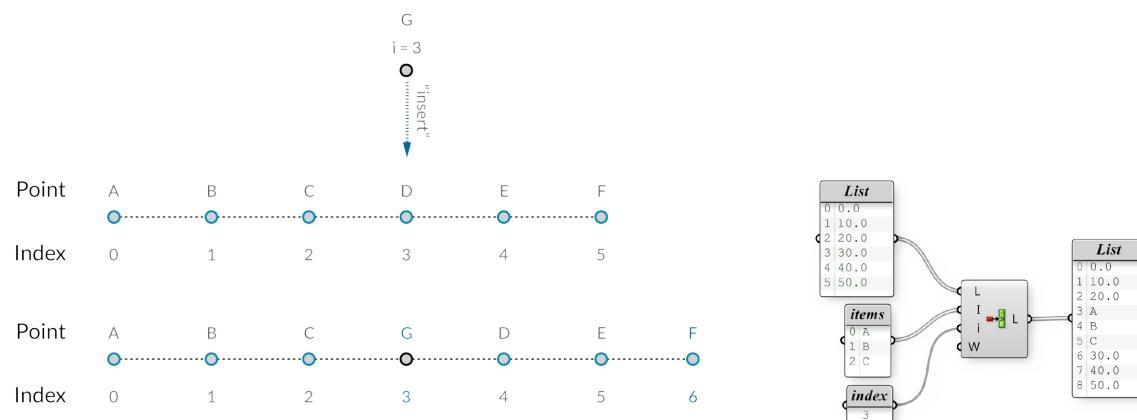
1.4.6.4. КОМПОНЕНТ SHIFT LIST

Компонент Shift List (Sets/Sequence/Shift List) будет либо перемещать элементы в списке вверх или вниз на какой-то шаг, в зависимости от значения смещения. Мы подключили выход List ко входу Shift-L, в тоже время подключив число ко входу Shift-S. Если мы установим смещение на -1, все значения списка будут смещаться вниз на одно число. Точно также, если мы изменим смещение на +1, все значения списка будут смещаться вверх на одно число. Если вход Wrap равняется True (правда), то элементы, которые не попадают в границы, будут крепиться к началу или концу списка. В этом примере, мы поставили значение смещения на +1, так что наш список смещается на одно значение вверх. Теперь, нам предстоит решить, как нам поступить с первым значением. Если мы установим значение Wrap на False (ложь), то первое значение будет смещаться вверх и выходить из списка, главным образом, удаляя это значение из набора данных (так что длина списка на один меньше, чем была до этого). Тем не менее, если мы установим значение Wrap на True, первое значение переместиться к концу списка



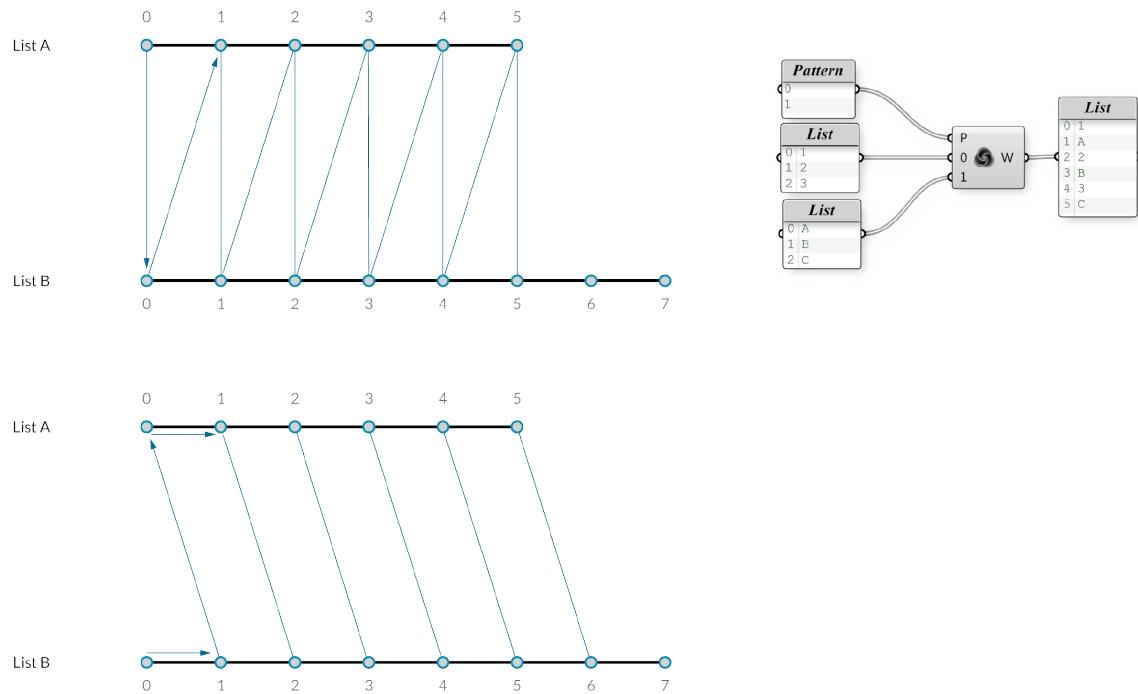
1.4.6.5. КОМПОНЕНТ INSERT ITEMS

Компонент Insert Items (Sets/Lists/Insert Items) позволяет вставлять набор элементов в список. Для того чтобы это все работало как надо, вам необходимо знать элементы, которые вы хотите вставить и индекс положение для каждого нового элемента. В примере ниже, мы вставим буквы A, B и C на индекс положение три.



1.4.6.6. КОМПОНЕНТ WEAVE

Компонент Weave (Sets/Lists/Weave) смешивает два или более списка вместе, на основе указанного паттерна плетения (Р вход). Когда паттерн и направление (поток) не совпадают идеально, этот компонент может либо вставить неизвестные значения в выходы потоков или может игнорировать потоки, которые уже были исчерпаны.



1.4.6.7. КОМПОНЕНТ CULL PATTERN

Компонент Cull (Sets/Sequence/Cull Pattern) удаляет элементы в списке, используя повторяющуюся битовую маску. Битовая маска определяется как список булевых значений. Битовая маска повторяется до тех пор, пока все элементы из списка данных не будут определены.

Point	A	B	C	D	E	F
Index	0	1	2	3	4	5
Pattern	True	False	(Repeat)		

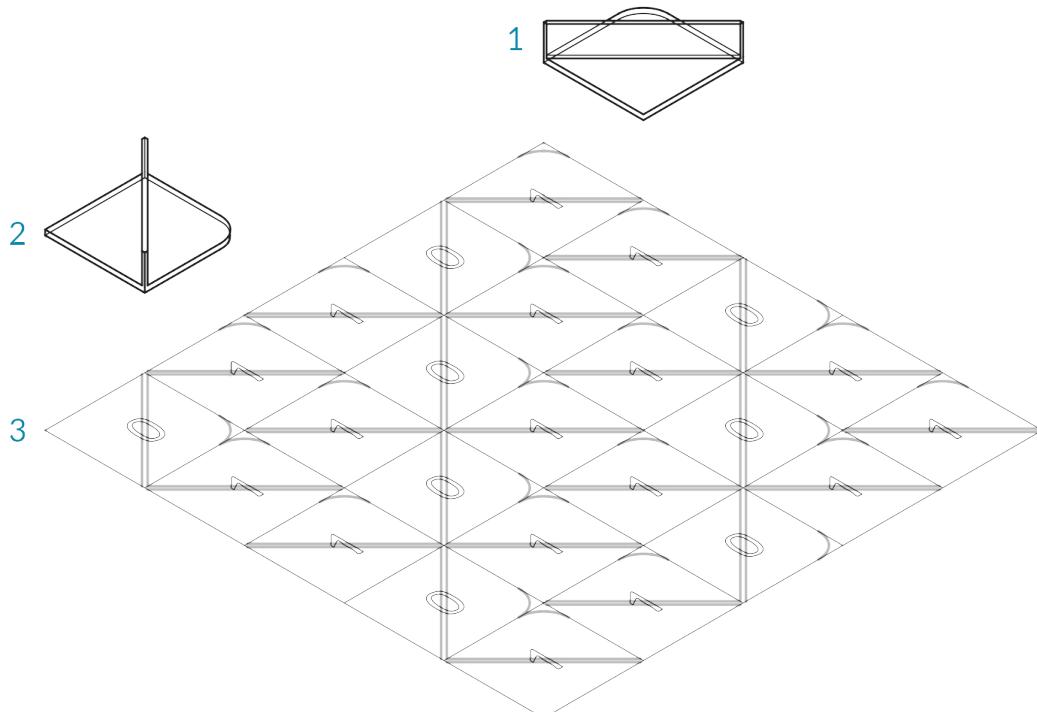
Point	A	B	C	D	E	F
Index	0	1	2	3	4	5
Pattern	True	False	True	False	True	False

Point	A		C		F	
Index	0		1		2	
Pattern	True	False	True	False	True	False

1.4.7. РАБОТА СО СПИСКАМИ

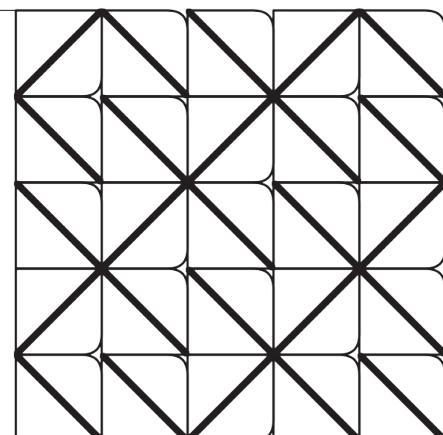
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

Давайте посмотрим на пример, используя компоненты из предыдущего раздела. В этом примере мы создаем рисунок под плитку применяя перенос геометрии к прямоугольной сетке. Этот паттерн создается путем использования компонента List Item для извлечения желаемой плитки из списка геометрии.



1. Геометрия соответствует индексу 1
2. Геометрия соответствует индексу 0
3. Прямоугольная сетка

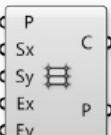
0	1	1	0	1
1	1	0	1	1
1	0	1	1	0
0	1	1	0	1
1	1	0	1	1

**1****2**

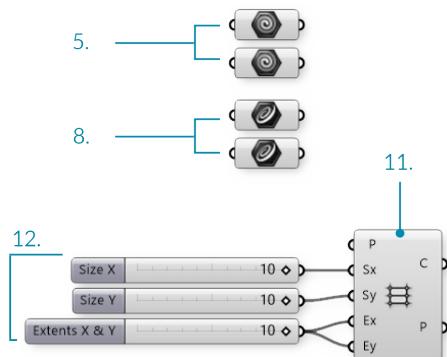
1. Перенесенный паттерн
2. Перенесенная геометрия

01.

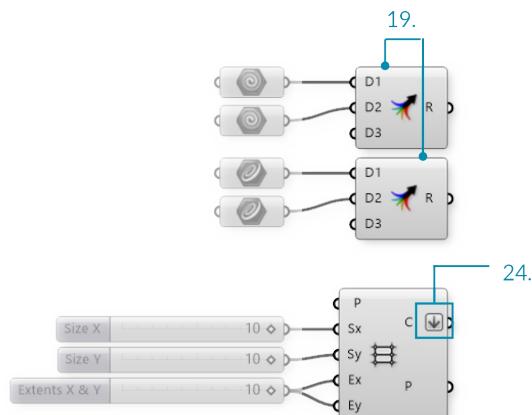
Запустите файл Rhinoceros.

02.	Создайте два равных по размеру квадрата.	
03.	<p>Создайте различную геометрию в каждом квадрате.</p> <p>В примере выше, мы создали простую поверхность с вкладкой скруглена, чтобы продемонстрировать ориентацию, и основа скруглена, чтобы различить две геометрии.</p>	
04.	Запустите новое определение набрав Ctrl+N (в Grasshopper).	
05.	Зайдите в Params/Geometry/Geometry – перетащите два параметра Geometry на холст. 	
06.	Кликните правой клавишей мыши по первому параметру Geometry и выберите set one Geometry . Укажите первую геометрию, к которой вы делаете привязку.	
07.	<p>Кликните правой клавишей мыши по второму параметру Geometry и выберите set one Geometry. Укажите вторую геометрию, к которой вы делаете привязку.</p> <p>Можно сделать привязку к нескольким геометриям в одном параметре, но для простоты мы использовали два отдельных параметра.</p>	
08.	Зайдите в Params/Geometry/Curve – вытащите два параметра Curve на холст. 	
09.	Кликните правой клавишей мыши по первому параметру Curve и выберите set one Curve . Укажите первый квадрат, к которому вы делаете привязку.	
10.	<p>Кликните правой клавишей мыши по второму параметру Curve и выберите set one Curve. Укажите второй квадрат, к которому вы делаете привязку.</p> <p>Убедитесь, что геометрия и квадрат, на который вы ссылаетесь, находятся в соответствии.</p>	
11.	Зайдите в Vector/Grid/Rectangular – перетащите компонент Rectangular Grid на холст. 	
12.	Зайдите в Params/Input/Slider - перетащите три слайдера Number Sliders на холст.	
13.	<p>Дважды кликните по первому слайдеру Number Slider и установите следующее:</p> <p>Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10</p>	
14.	<p>Дважды кликните по второму слайдеру Number Slider и установите следующее:</p> <p>Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10</p>	
15.	<p>Дважды кликните по третьему слайдеру Number Slider и установите следующее:</p> <p>Name: Extents X & Y Rounding: Integers Lower Limit: 0 Upper Limit: 10 Value: 10</p>	

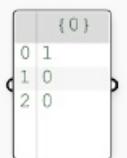
16.	Соедините первый Number Slider с входом Size X (Sx) компонента Rectangular Grid .
17.	Соедините второй Number Slider с входом Size Y (Sy) компонента Rectangular Grid .
18.	Соедините третий Number Slider с входом Extent X (Ex) и входом Extent Y (Ey) компонента Rectangular Grid .

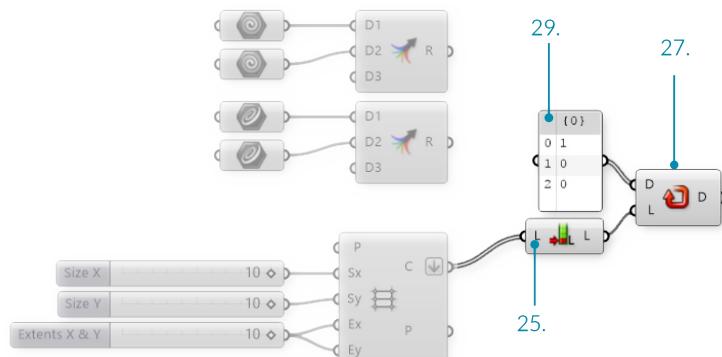


19.	Зайдите в Sets/Tree/Merge – вытащите два компонента Merge на холст.	
20.	Соедините первый параметр Geometry с входом Data Stream 1 (D1) первого компонента Merge .	
21.	Соедините второй параметр Geometry с входом Data Stream 2 (D2) первого компонента Merge .	
22.	Соедините первый параметр Curve с входом Data Stream 1 (D1) второго компонента Merge .	
23.	Соедините второй параметр Curve с входом Data Stream 1 (D2) второго компонента Merge .	
24.	Кликните правой клавишей мыши на выходе Cells (C) компонента Rectangular Grid и выберите Flatten.	

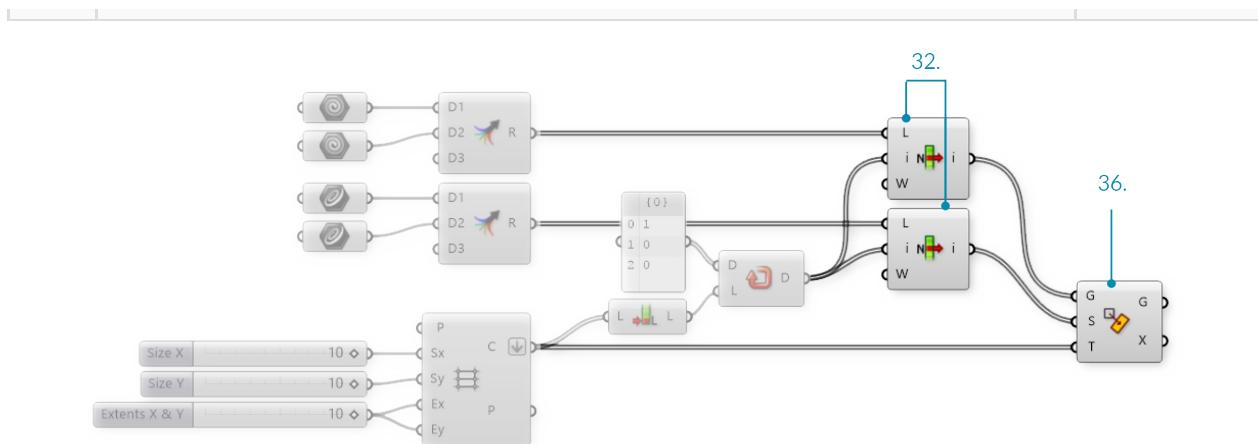


25.	Зайдите в Sets/List/List Length – вытащите компонент List Length на холст.	
26.	Соедините выход Cells (C) компонента Rectangular Grid со входом List (L) компонента List Length .	

	27. Зайдите в Sets/Sequence/Repeat Data – перетащите компонент Repeat Data на холст.	
28.	Соедините выход Length (L) компонента List Length со входом Length (L) компонента Repeat Data .	
29.	Зайдите в Params/Input/Panel – перетащите Panel на холст.	
30.	Дважды кликните по Panel . Снимите выделение с multiline data, wrap items и special codes. Введите следующее: 1 0 0	
31.	Это паттерн, в котором распространяются геометрии. 0 вызывает первую исходную Geometry, 1 вызывает вторую исходную Geometry. Изменяя числовую последовательность будет меняться паттерн, также как будет меняться протяженность сетки.	
31.	Соедините Panel с входом Data (D) компонента Repeat Data .	

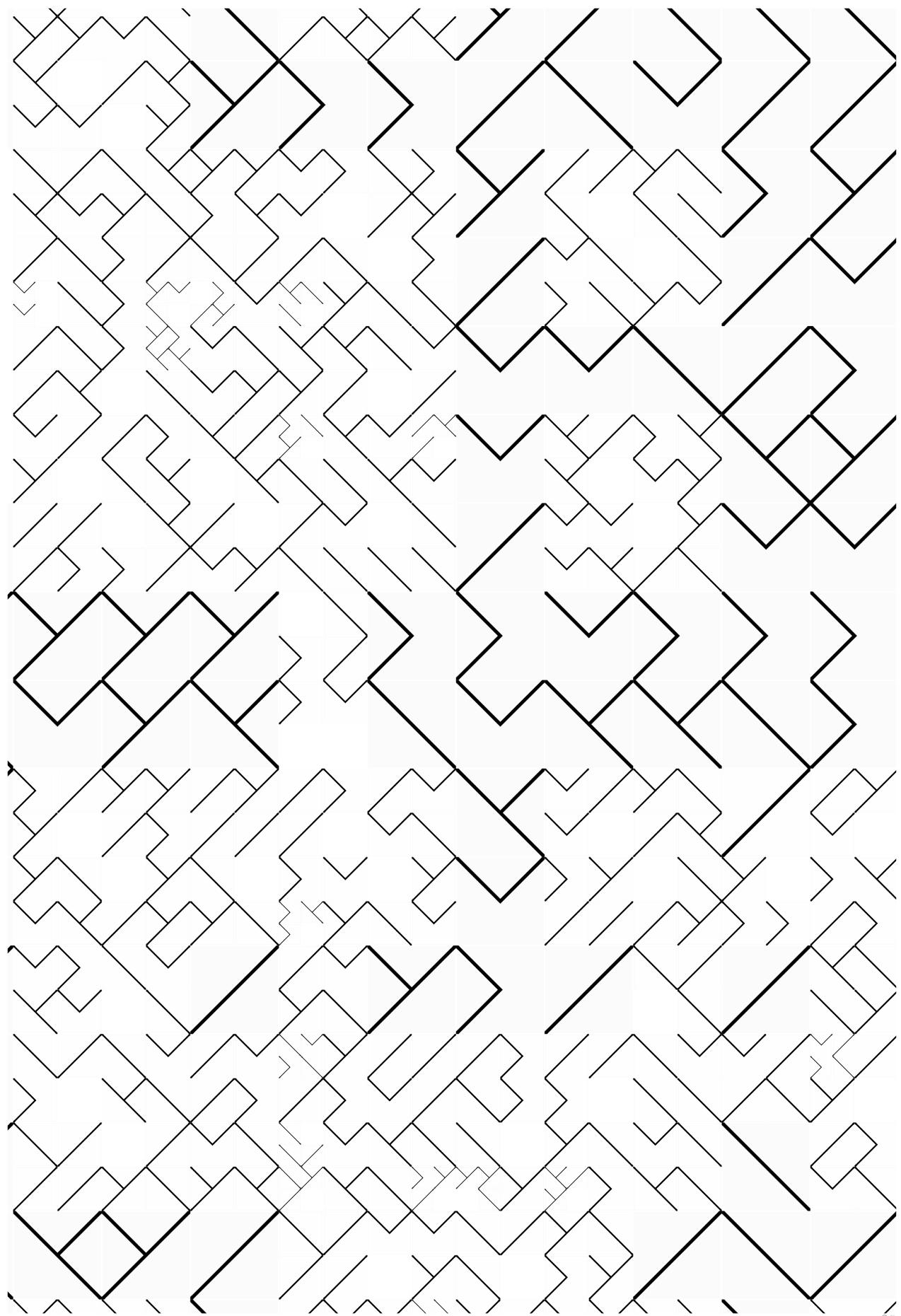


	32. Зайдите в Sets/List/List Item – перетащите два компонента List Item .	
33.	Соедините выход Result (R) первого компонента Merge с входом List (L) первого компонента List Item .	
34.	Соедините выход Result (R) второго компонента Merge с входом List (L) второго компонента List Item .	
35.	Соедините выход Data (D) второго компонента Repeat Data с входом Index (i) первого и второго компонентов List Item .	
36.	Зайдите в Transform/Affine/Rectangle Mapping – перетащите Rectangle Mapping компонент на холст.	
37.	Соедините выход Cells (C) компонента Rectangular Grid со входом Target (T) компонента Rectangle Mapping .	
38.	Соедините выход items (I) первого компонента List Item с входом Geometry (G) компонента Rectangular Grid .	
39.	Соедините выход items (I) второго компонента List Item с входом Source (S) компонента Rectangular Grid .	



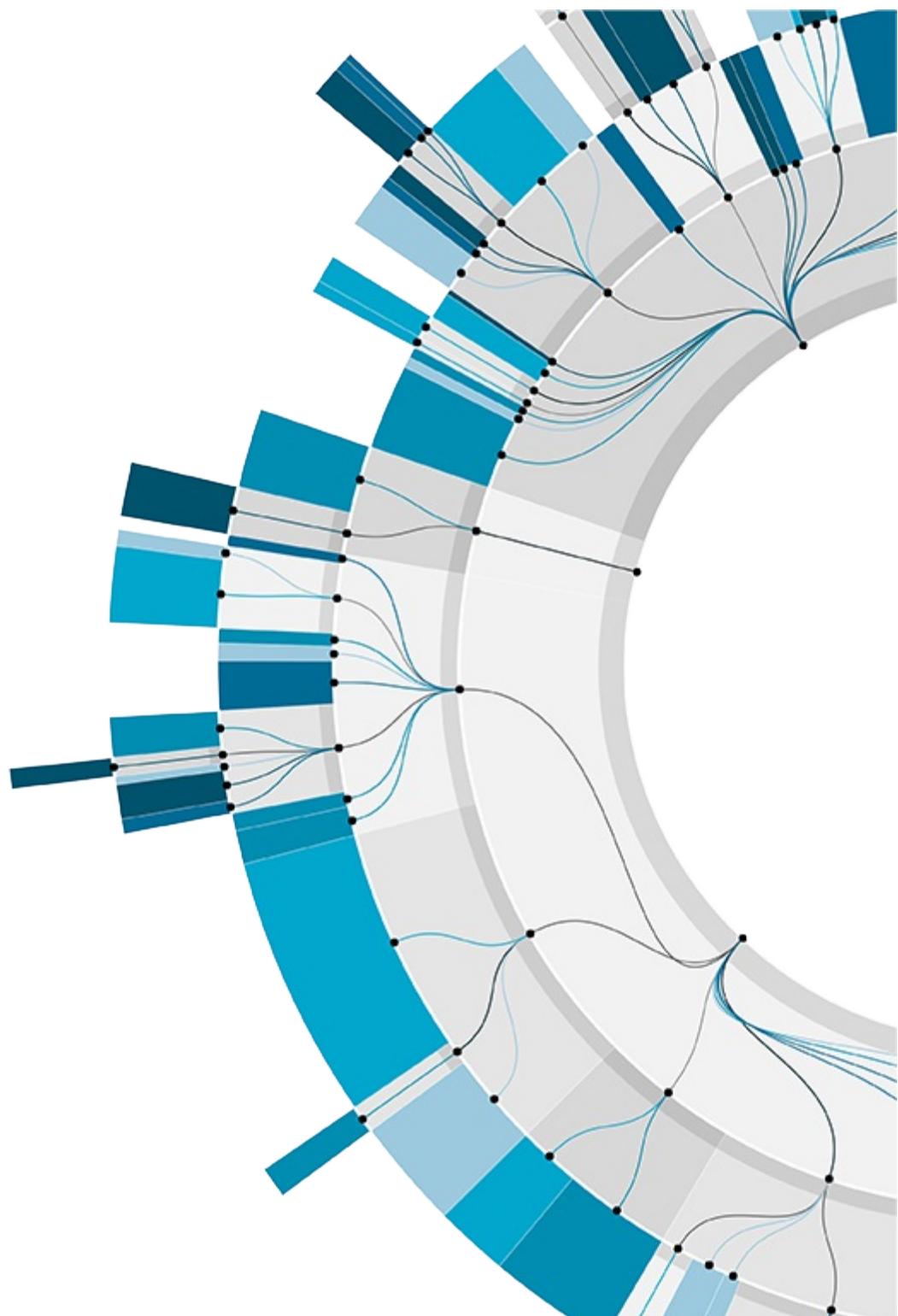
При изменении вводной геометрии и паттерна будет меняться итоговый паттерн плитки.

	#	A1,A2	B1,B2	C1,C2	HYBRID																										
	0 1	☒ ☒	☒ ☒	☒ ☞	B1,C2																										
1,1,0...		<table border="1"> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	0	1	1	0	1	1	1	0	1	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1	1				
0	1	1	0	1																											
1	1	0	1	1																											
1	0	1	1	0																											
0	1	1	0	1																											
1	1	0	1	1																											
1,0,0...		<table border="1"> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	1	0				
0	1	0	0	1																											
1	0	0	1	0																											
0	0	1	0	0																											
0	1	0	0	1																											
1	0	0	1	0																											



1.5. ПРОЕКТИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДЕРЕВЬЕВ ДАННЫХ

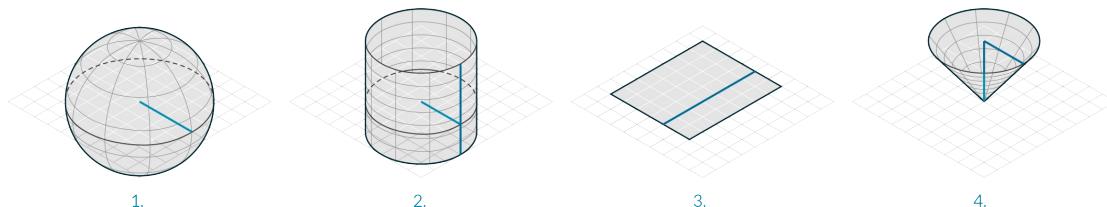
При увеличении сложности ваших определений, объем данных также увеличивается. Чтобы эффективно использовать Grasshopper, важно понимать, как хранятся большие объемы данных, как к ним можно получить доступ и как с ними обращаться.



1.5.1. Геометрия Поверхности

NURBS (неоднородные рациональные Б-сплайны) - математическое представление, которое может точно смоделировать любую форму из простой 2D линии, круга, арки или коробки в самую сложную 3D произвольной формы органическую поверхность или тело. Благодаря своей гибкости и точности, NURBS модели могут использоваться в любом процессе от иллюстрации и анимации до производства.

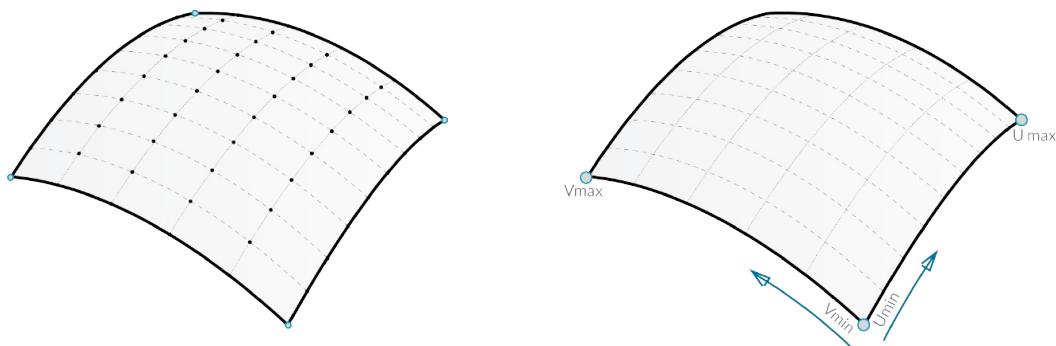
В отличие от нескольких примитивных типов поверхности, таких как сфера, конусы, плоскости и цилиндры, Rhino поддерживает три типа поверхностей свободных форм, самая полезная из которых NURBS поверхность. Как и кривые, все возможные формы поверхности могут быть представлены NURBS поверхностью, которая является запасной по умолчанию в Rhino. Она также, на данный момент, является самой полезной поверхностью определения и именно на ней мы и сосредоточимся.



1. Сфера примитив [плоскость, радиус]
2. Цилиндр примитив [плоскость, радиус, высота]
3. Плоскость примитив [плоскость, ширина, высота]
4. Конус примитив [плоскость, радиус, высота]

1.5.1.1. NURBS ПОВЕРХНОСТИ

NURBS поверхности очень похожи на NURBS кривые. Такие же алгоритмы используются для вычисления формы, нормалей, тангенсов, кривизны и других характеристик, но существуют и явные отличия. Например, кривые имеют тангенс векторы и нормальные плоскости, в то время как поверхности имеют нормальные векторы и тангенс плоскости. Это означает, что у кривых отсутствует ориентация, в то время как у поверхности отсутствует направление. В случае NURBS поверхностей, фактически два направления заложены в геометрию, потому что NURBS поверхности - это прямоугольные сетки кривых $\{u\}$ и $\{v\}$. И даже хотя эти направления часто произвольны, мы все-равно их используем, потому что они облегчают нам жизнь.

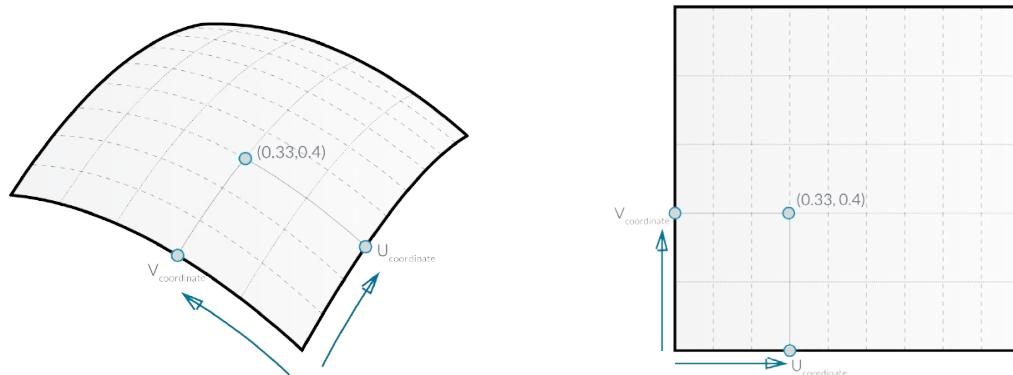


Вы можете рассматривать NURBS поверхности как сетку NURBS кривых, которые идут в двух направлениях. Форма NURBS поверхности определяется числом контрольных точек и порядком этой поверхности в направлениях "u" и "v". NURBS поверхности достаточно для хранения и представления поверхностей свободной формы с высокой степенью точности.

Surface Domain Поверхность диапазона определяется порядком параметров (u, v), которые определяют в 3-D точке на этой поверхности. Диапазон в каждом пространстве (u или v) обычно описывается как два

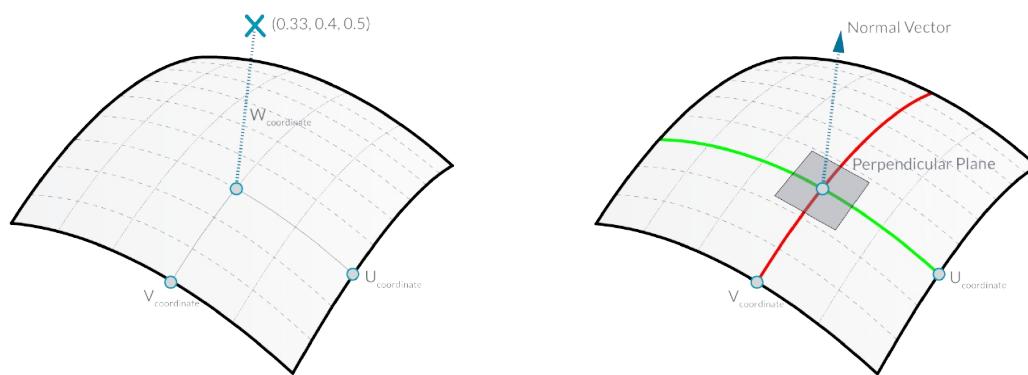
действительных числа (u_{min} к u_{max}) и (v_{min} к v_{max}). Изменение диапазона поверхности считается репараметризацией поверхности.

В Grasshopper часто бывает полезно репараметризовать NURBS поверхности так, чтобы диапазоны u и v были оба от 0 до 1. Это позволяет нам легко определить и производить операции на поверхности.



Определение параметров на равных интервалах в 2-D параметрическом прямоугольнике не всегда переносится на равные интервалы в 3-D пространстве.

Surface evaluation Определение поверхности по параметру, который является частью диапазона поверхности, приводит к точке на поверхности. Имейте ввиду, что середина диапазона ($mid-u$, $mid-v$) не обязательно может определять среднюю точку 3D поверхности. Также, при определении u и v значений, которые не входят в диапазон поверхности, мы не получим какого-либо полезного результата.



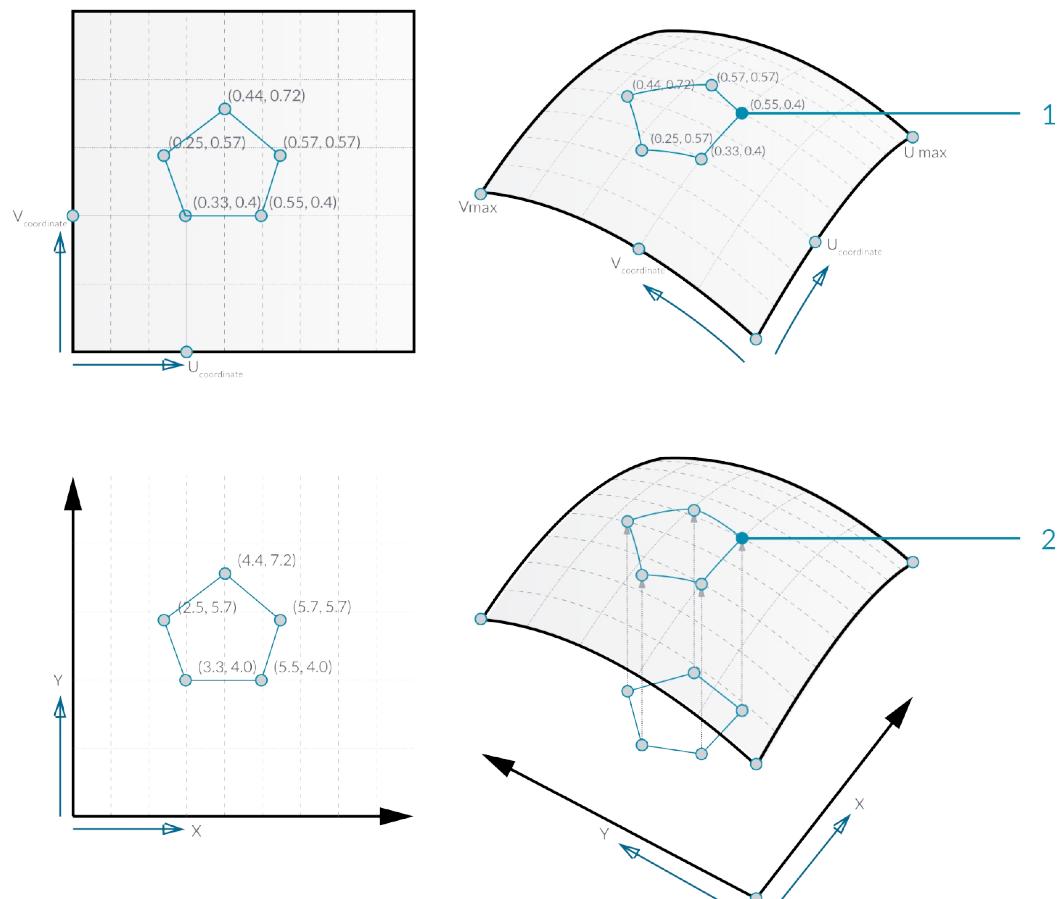
Normal Vectors and Tangent Planes Тангенс плоскость к поверхности в указанной точке - это плоскость, которая касается поверхности в этой точке. Z направление тангенс плоскости представляет нормальное направление поверхности в этой точке.

Grasshopper занимается NURBS поверхностями схожим образом как Rhino, потому что он построен на том же самом ядре операций, которые необходимы для генерации поверхности. Тем не менее, из-за того, что Grasshopper отображает поверхность поверх видового окна Rhino (именно поэтому, вы не можете выбрать геометрию, созданную через Grasshopper, в видовом окне, до тех пор, пока вы не запечатаете результаты в этой сцене) некоторые из настроек mesh слегка ниже, чтобы держать скорость вычислений Grasshopper достаточно высоко. Вы могли заметить некоторую граненность поверхности mesh, но это ожидаемо и является результатом настроек прорисовки Grasshopper. Любая запечатенная геометрия будет все равно использовать самые высокие настройки mesh.

1.5.1.2. ПРОЕЦИРОВАНИЕ ПОВЕРХНОСТЕЙ

В предыдущем разделе мы объяснили, что NURBS поверхности содержат их собственные координаты

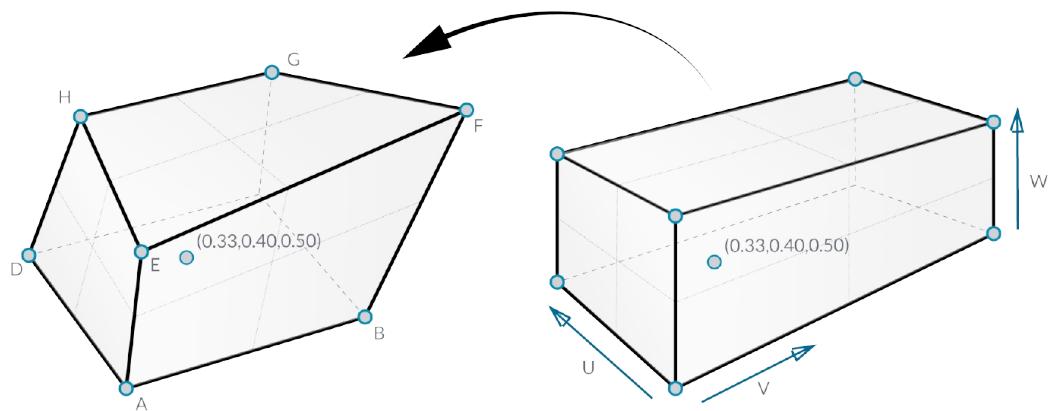
пространства, установленные диапазонами и и в. Это означает, что двух-пространственная геометрия, которая определяется координатами х и у, может быть перенесена на пространство uv. Геометрия будет растягиваться и изменяться в ответ на кривизну поверхности. В этом отличие от простого проецирования 2d геометрии на поверхность, где векторы прорисовываются из 2d геометрии в указанном направлении пока они не пересекутся с поверхностью.



Вы можете думать о проецировании как о геометрии, накладывающей тень на поверхность, и о переносе как о геометрии, которую растянули по поверхности.

1. Перенесенная геометрия, определяемая координатами uv
2. Проецируемая геометрия на поверхность

Как 2d геометрия может быть спроектирована на пространство uv поверхности, так и 3d геометрия, находящаяся в коробке, может быть перенесена на соответствующую скрученную коробку на участке поверхности. Эта операция называется *box morphing* и используется для заполнения кривых поверхностей трех-пространственными геометрическими компонентами.

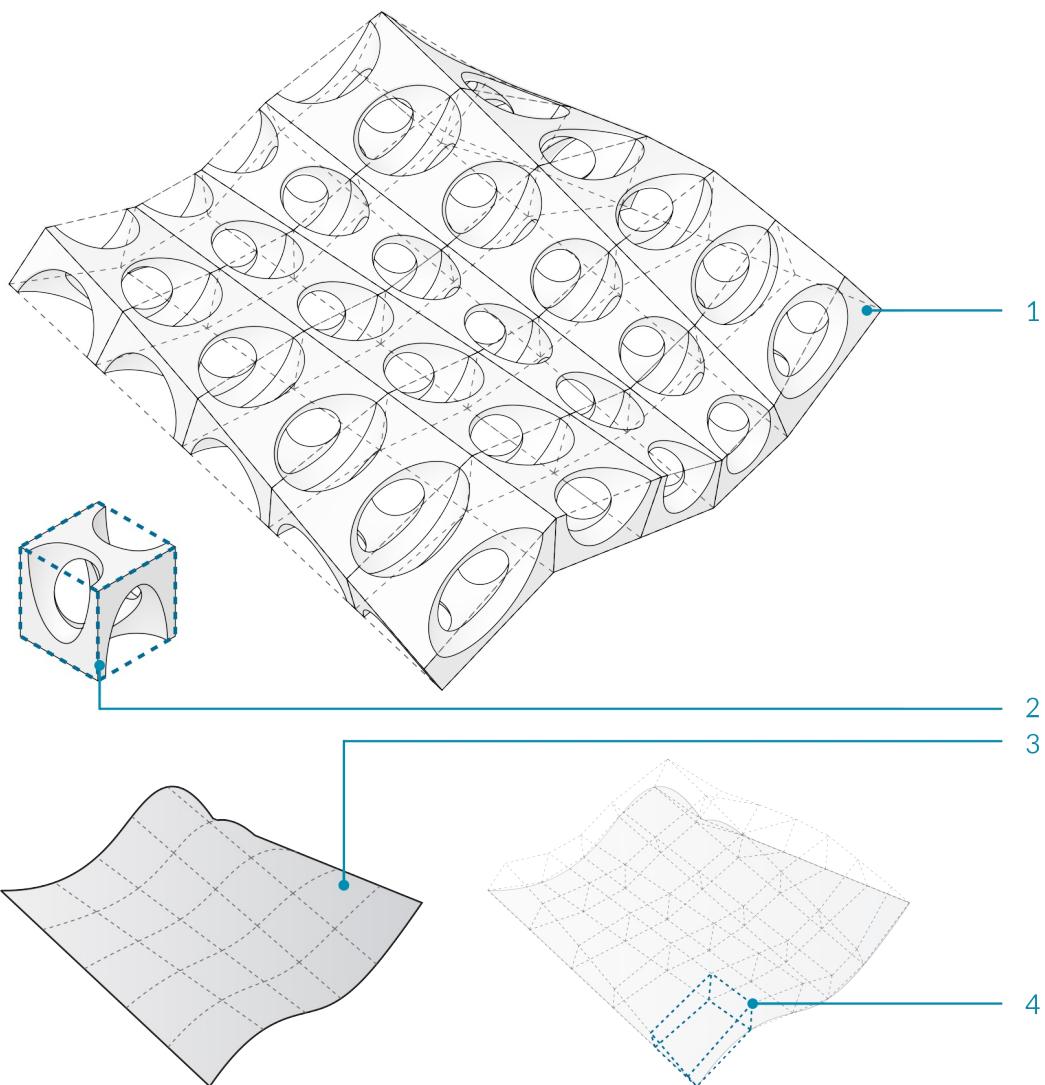


Чтобы разместить скрученную коробку на поверхности, диапазон поверхности должен быть разделен, чтобы создать сетку на участке поверхности. Скрученная коробка создается прорисовкой векторов нормали в углах каждого участка до желаемой высоты и созданием коробки, определяемой конечными точками тех векторов и угловых точек участка.

1.5.1.3. ОПРЕДЕЛЕНИЕ ПРЕОБРАЗОВАНИЯ

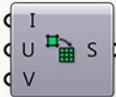
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

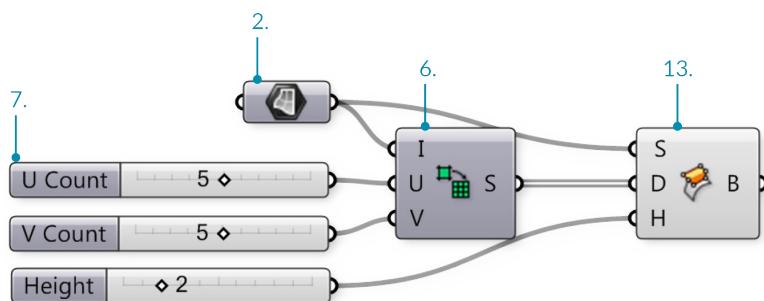
В этом примере, мы будем использовать компонент `box morph` для заполнения NURBS поверхности геометрическими компонентами.



1. NURBS поверхность, заполненная компонентом.
2. Исходный компонент в исходной коробке.
3. Поверхность, разделенная на участки.
4. Скрученные коробки, размещенные на поверхности.

01.	Запустите новое определение набрав Ctrl+N (в Grasshopper)	
02.	Зайдите в Params/Geometry/Surface – перетащите параметр Surface на холст Это поверхность, которая будет заполнена геометрическими компонентами.	
03.	Зайдите в Params/Geometry/Geometry – перетащите параметр Geometry на холст Этот компонент будет размещен поверх поверхности.	
04.	Кликните правой клавишей мыши по параметру Surface и выберите “Set One Surface” – выберите поверхность для привязки в видовом окне Rhino	
05.	Кликните правой клавишей мыши по параметру Geometry и выберите “Set One Geometry” – выберите вашу геометрию Rhino	

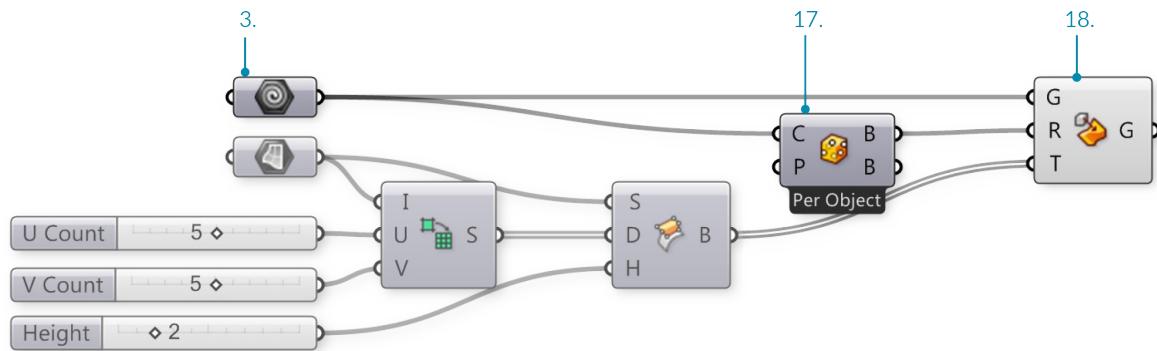
06.	Domain2 на холст	
07.	Зайдите в Params/Input/Number Slider – перетащите три слайдера Number Sliders на холст	
08.	Дважды кликните по первому Number Slider и установите следующее: Rounding: Integer Lower Limit: 0 Upper Limit: 10 Value: 5	
09.	Установите такие же значения на втором и на третьем Number Sliders	
10.	Соедините выход параметра Surface с входом Domain (I) компонента Divide Domain2	
11.	Подключите первый Number Slider к входу U Count (U) в компоненте Divide Domain2	
12.	Подключите второй Number Slider к входу V Count (V) в компоненте Divide Domain2	
13.	Зайдите в Transform/Morph/Surface Box – вытащите компонент Surface Box на холст	
14.	Соедините выход параметра Surface с входом Surface (S) компонента Surface Box	
15.	Соедините выход Segments (S) компонента Divide Domain2 к входу Domain (D) компонента Surface Box	



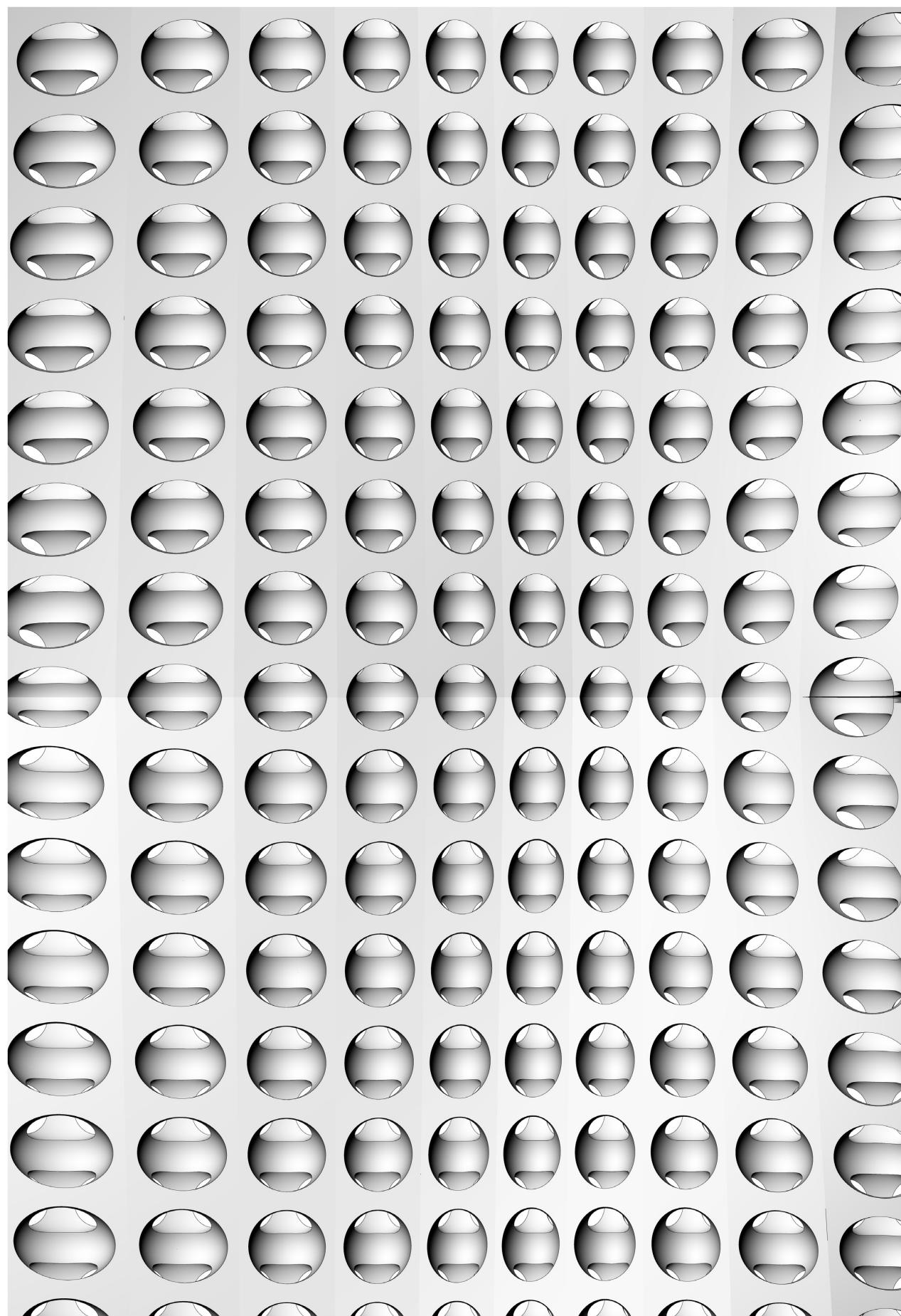
Вы должны увидеть сетку скрученных коробок, заполняющих вашу исходную поверхность. Поменяйте слайдеры U и V, чтобы изменить количество коробок и используйте слайдер высоты, чтобы настроить их высоту.

16.	Подключите третий Number Slider к входу Height (H) в компоненте Surface Box	
17.	Зайдите в Surface/Primitive/Bounding Box – перетащите компонент Bounding Box на холст	
18.	Зайдите в Transform/Morph/Box Morph – перетащите компонент Box Morph на холст	
19.	Соедините выход параметра Geometry с входом Content (C) компонента	

19.	Соедините выход параметра Geometry с входом Content (C) компонента Bounding Box	
20.	Соедините выход параметра Geometry с входом Geometry (G) компонента Box Morph	
21.	Соедините выход Box (B) компонента Bounding Box с входом Reference (R) компонента Box Morph	
22.	Соедините выход Twisted Box (B) компонента Surface Box с входом Target (T) компонента Box Morph	



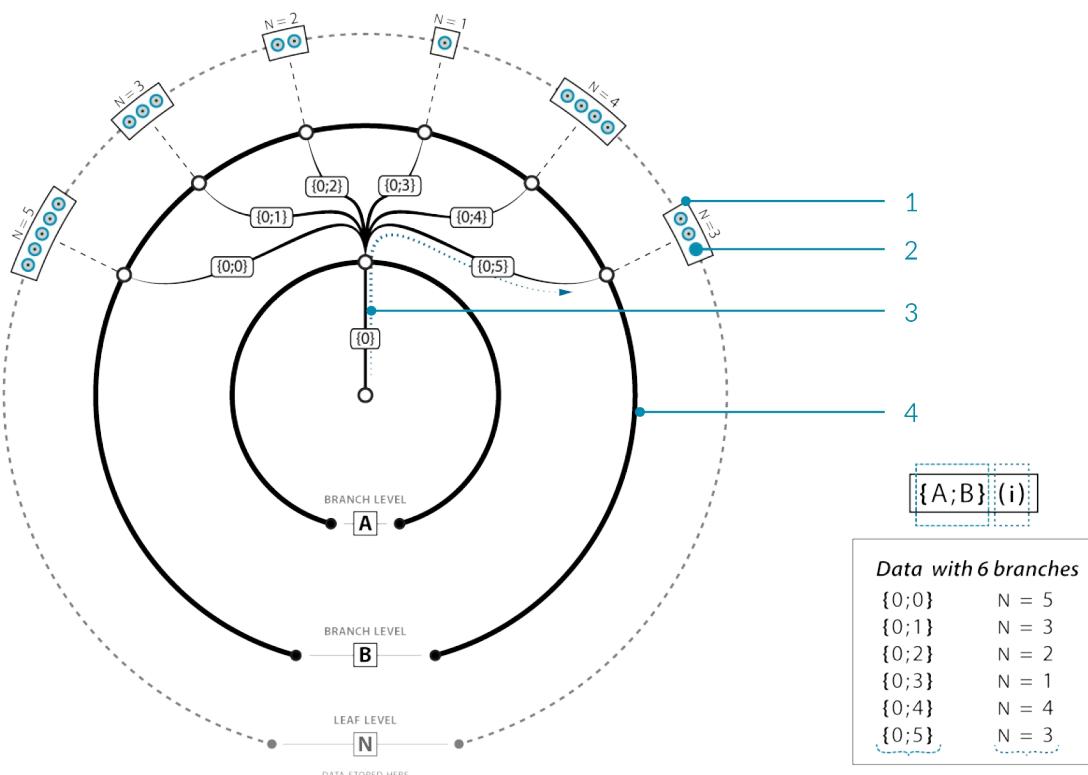
Вы должны увидеть теперь, как ваша геометрия наполняет вашу поверхность.



1.5.2. Что такое Дерево Данных?

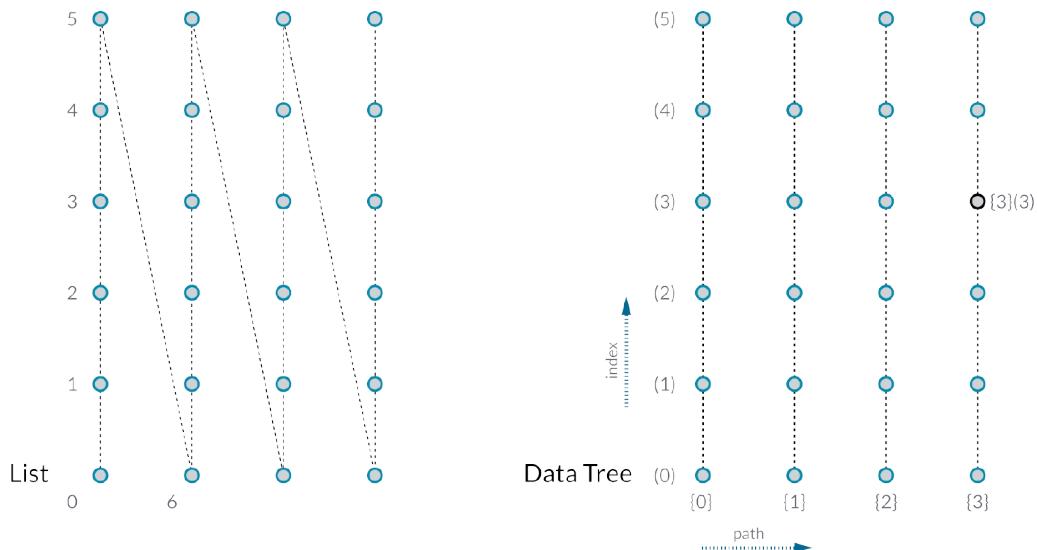
Дерево данных - это иерархическая структура для хранения данных, состоящая из списков. Деревья данных создаются, когда компонент Grasshopper структурируется, чтобы принять набор данных и вывести множественные наборы данных. Grasshopper работает с этими новыми данными, размещая их в под-списки. Эти под-списки работают таким же образом, как и структуры папок на вашем компьютере, а именно, чтобы получить доступ к индексированным элементам требуется пройти по путям, которые проинформированы своим поколением родительских списков и их собственными под-индексами.

Это становится возможным иметь множественные списки данных внутри одного параметра. Как только становятся доступны множественные списки, возникает необходимость в способе распознавания каждого отдельного списка. Дерево данных, по сути, список списков или, иногда, список списков списков (и т.д.).



В изображении выше, вы видите одну главную ветку (можете называть ее стволов, но так как тут возможно наличие множества главных веток, то этот термин может быть неточным) пути {0}. Этот путь не содержит данных, но имеет 6 под-веток. Каждая из этих под-веток наследует индекс родительской ветки {0} и добавляет свой собственный под-индекс (0, 1, 2, 3, 4 и 5, соответственно). Будет неправильно называть это число "индекс", потому что это подразумевает только одно число. Возможно, лучше называть это "путь", так как он похож на структуру папок на диске. У каждой из этих под-веток мы встречаем какие-либо данные. Каждый элемент данных, таким образом, часть одной (и только одной) ветки в дереве, каждый элемент имеет индекс, который указывает его расположение внутри ветки. У каждой ветки есть свой путь, который указывает ее расположение внутри дерева.

Изображение ниже иллюстрирует разницу между списком и деревом данных. Слева, массив из четырех колонок по 6 точек каждая - это все, что содержится в одном списке. Первая колонка пронумерована 0-5, вторая 6-11 и т.д. Справа расположен тот же самый массив точек, содержащихся в дереве данных. Дерево данных - это список из четырех колонок, где каждая колонка - это список из шести точек. Индекс каждой точки это (номер колонки, номер ряда). Это намного более полезный способ организации этих данных, потому что вы можете легко получить доступ и работать со всеми точками в данном ряду или колонке, удалить каждый второй ряд точек, соединить чередующиеся точки и прочее.

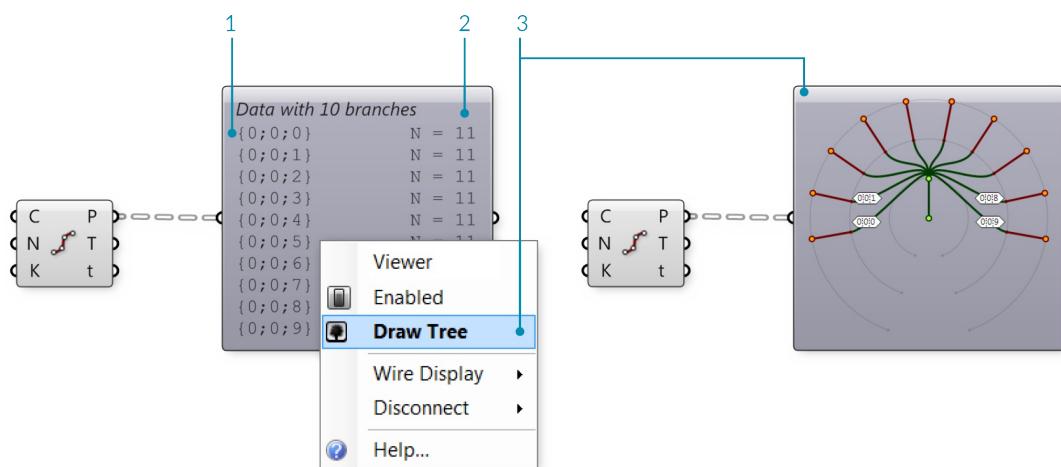


1.5.2.1. ВИЗУАЛИЗАЦИЯ ДЕРЕВА ДАННЫХ

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

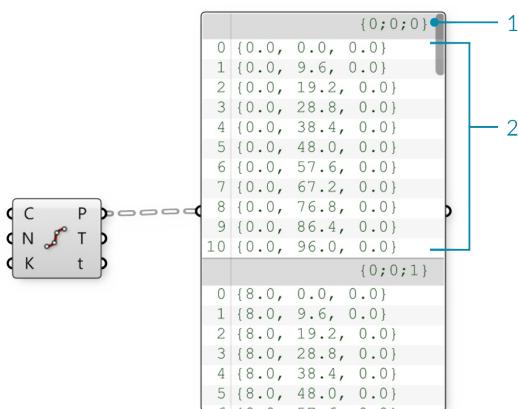
Из-за их комплексности понять, как работают деревья данных, может быть затруднительно. У Grasshopper есть несколько инструментов для визуализации и понимания данных, хранящихся в дереве.

The Param Viewer Param Viewer (Params/Util/Param Viewer) позволяет визуализировать данные в текстовом формате и в виде дерева. Соедините любой выход, содержащий данные, с входом Param Viewer. Чтобы данные отобразились в виде дерева, кликните правой клавишей мыши по Param Viewer и выберите "draw tree". В этом примере, Param Viewer соединен с выходом Points (P) компонента Divide Curve, который разделил 10 кривых на 10 сегментов каждую кривую. Десять веток соответствуют десяти кривым, каждая содержит список из 11 точек, которые являются точками деления кривой.



1. Путь каждого списка
2. Число элементов в каждом списке
3. Выберите "Draw Tree" для отображения дерева данных

Если подсоединить панель к тому же самому выходу, то она отобразит десять списков из 11 элементов каждый. Вы также можете заметить, что каждый элемент является точкой, определяемой тремя координатами. Путь отображается поверх каждого списка и соответствует путям, перечисленным в Param Viewer

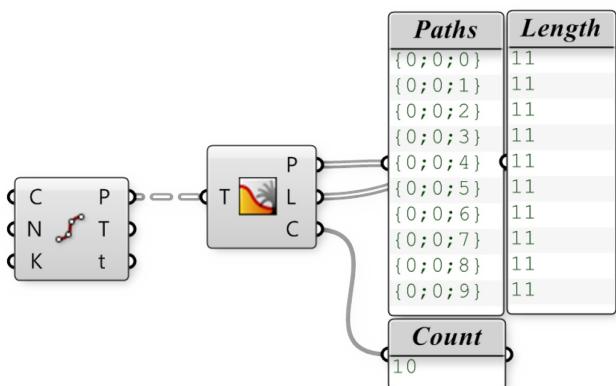


1. Путь
2. Список из 11 элементов

Tree Statistics Компонент Tree Statistics (Sets/Tree/Tree Statistics) выдает обратно некоторую статистику Дерева Данных, включая:

- P - все пути дерева
- L - длина каждой ветки дерева
- C - число путей и веток дерева

Если соединить выход Points того же самого компонента Divide Curve, мы сможем показать все пути, длины и число панелей. Этот компонент полезен тем, что он разделяет статистику на три выхода, позволяя просматривать только один, которые необходим.



Оба компонента Param Viewer и Tree Statistics полезны для визуализации изменений в структуре Дерева данных. В следующем разделе, мы рассмотрим некоторые операции, которые могут быть выполнены, чтобы изменить эту структуру.

1.5.3. Создание Дерева Данных

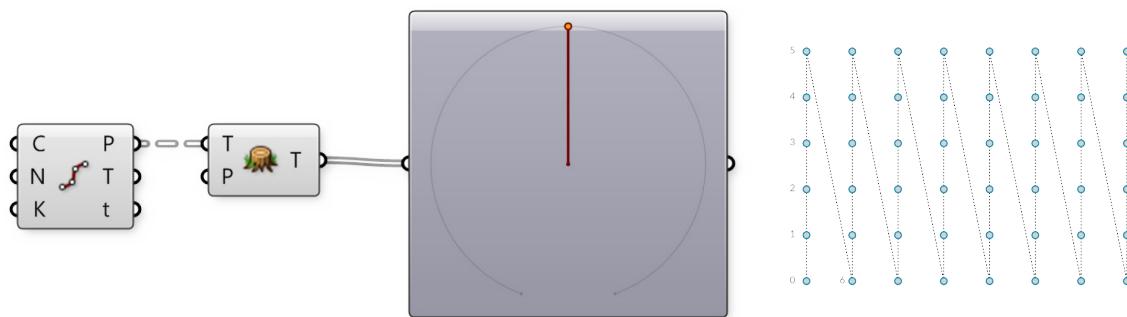
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

В Grasshopper есть инструменты для изменения структуры дерева данных. Эти инструменты могут помочь вам получить определенную информацию, касающуюся этого дерева, и изменить способ хранения, расположения и идентификации.

Давайте посмотрим на некоторые манипуляции с деревом данных и визуализируем то, как они влияют на дерево.

1.5.3.1. FLATTEN

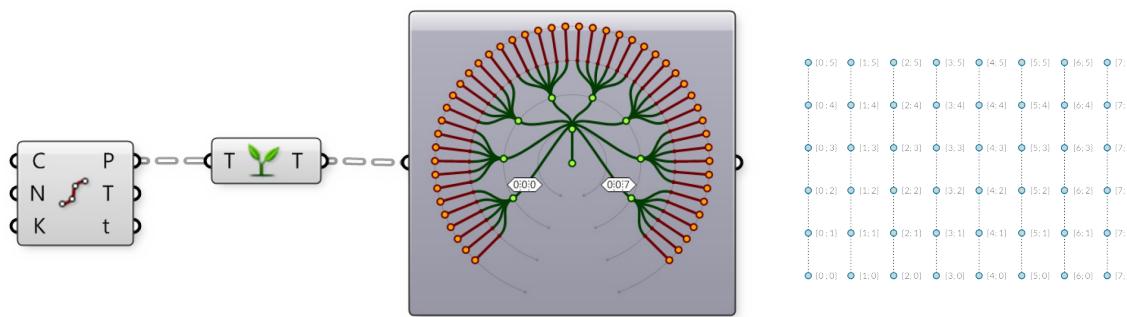
Flatten удаляет все уровни Дерева Данных и оставляет только один Список. Применяя компонент Flatten (Sets/Tree/Flatten) к выходу Р компонента Divide Curve, мы можем использовать Param Viewer для визуализации новой структуры данных.



В Param Viewer мы можем увидеть, что сейчас у нас только 1 ветка, содержащая список из 48 точек.

1.5.3.2. GRAFT TREE

Grafting создает новую Ветку для каждого Элемента Данных. Если мы запустим данные через компонент Graft Tree (Sets/Tree/Graft Tree), каждая точка деления сейчас имеет свою отдельную ветку, а не делит ветку с другими точками делениями на той же самой кривой.

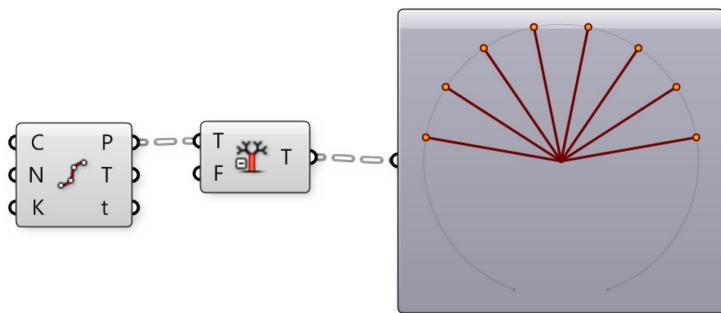


В Param Viewer мы можем увидеть, какими были данные с 8 ветками из 6 элементов каждая и что сейчас мы имеем 8 веток с 6 под-ветками, содержащими по одному элементу каждая.

1.5.3.3. SIMPLIFY TREE

Simplify удаляет накладывающиеся Ветки в Дереве данных. Если мы запустим данные через компонент

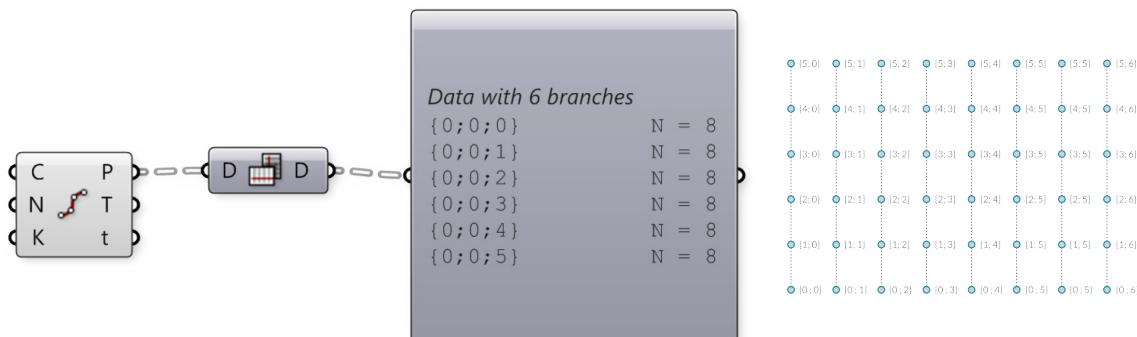
Simplify Tree (Sets/Tree/Simplify Tree), то первая ветка, не содержащая данных, будет удалена.



В Param Viewer у нас все равно имеется 8 веток из 6 элементов каждого, но первая ветка была удалена.

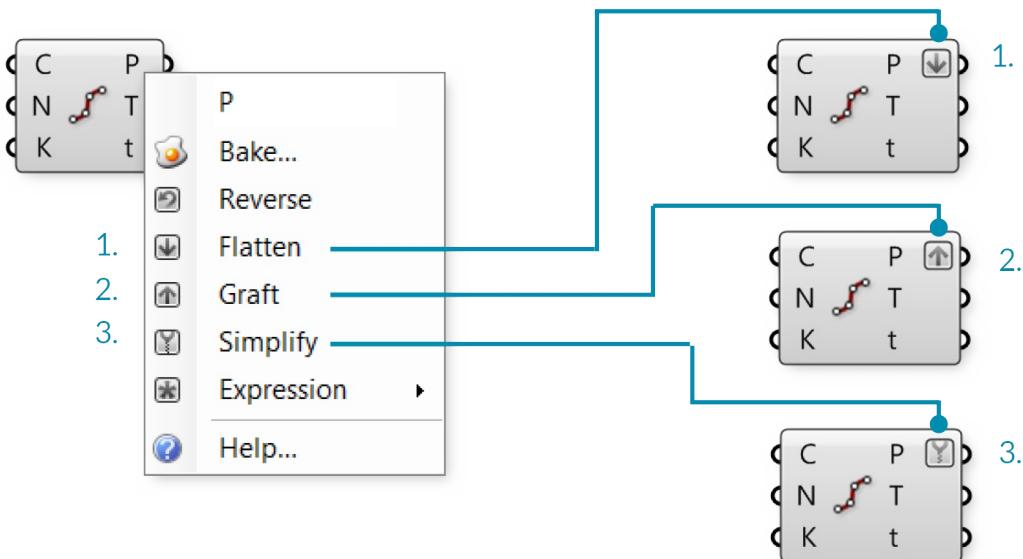
1.5.3.4. FLIP MATRIX

Компонент Flip Matrix (Sets/Tree/Flip Matrix) меняет местами "Ряды" и "Колонки" Дерева Данных с двумя индексами путями.



В Param Viewer мы можем увидеть, какими были данные с 8 ветками из 6 элементов каждого и что сейчас мы имеем 6 веток с 8 элементами каждого.

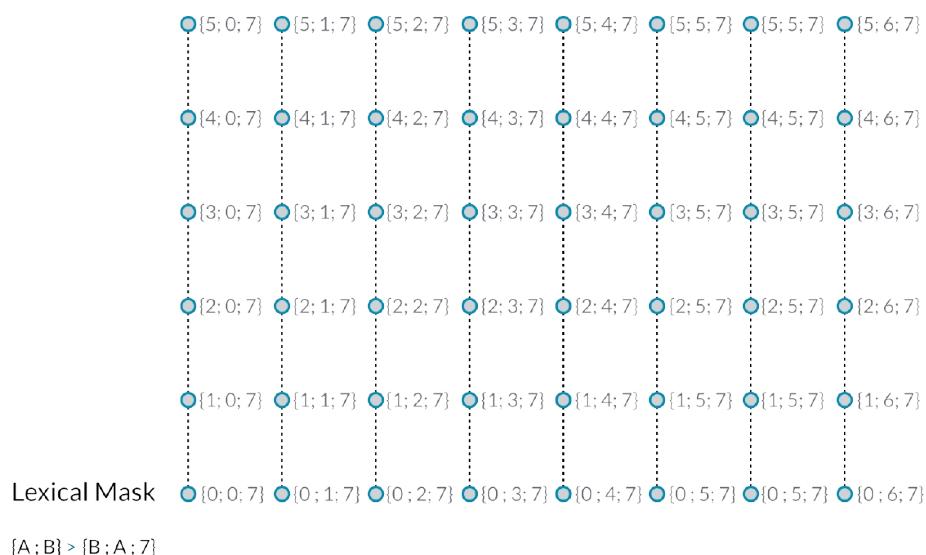
Действия Flatten, Graft и Simplify можно применить к входу или выходу компонента, вместо того, чтобы пропускать данные через отдельный компонент. Просто кликните правой клавишей по нужному входу или выходу и выберите Flatten, Graft или Simplify из меню. Компонент отобразит иконку, говорящую о том, что дерево было изменено. Помните о процессе работы Grasshopper. Если вы подключите Flatten к входу компонента, данные подвернутся его воздействию до того, как действие компонента будет выполнено. Если вы подключите Flatten к выходу компонента, данные подвернутся его воздействию после того, как компонент выполнит свое действие.

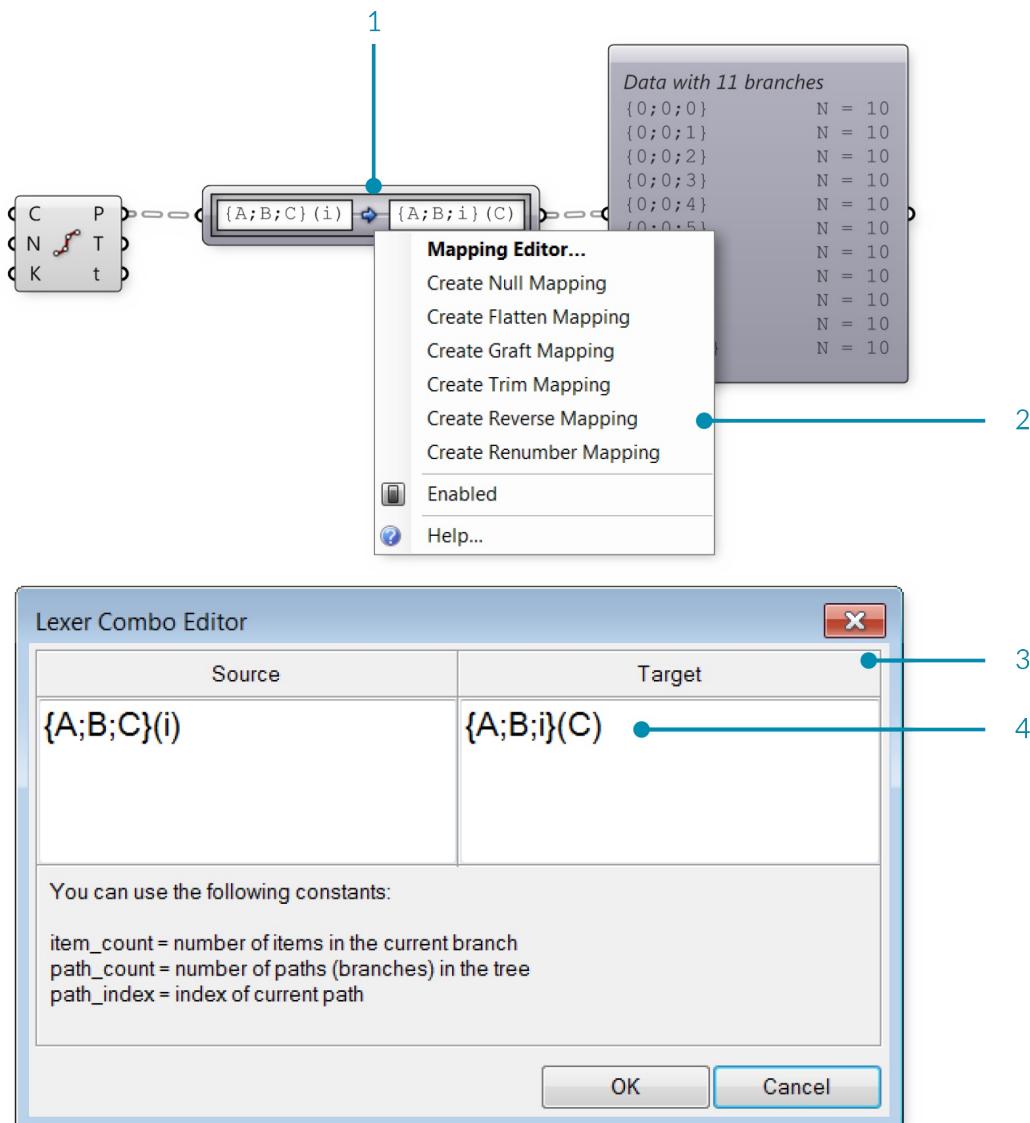


1. Flattened выход P
2. Grafted выход P
3. Simplified выход P

1.5.3.5. THE PATH MAPPER

Компонент Path Mapper (Sets/Tree/Path Mapper) позволяет вам выполнять лексические действия с деревьями данных. Лексические действия – это логические переносы между путями данных и индексы, которые определяются текстовыми (лексическими) масками и паттернами.



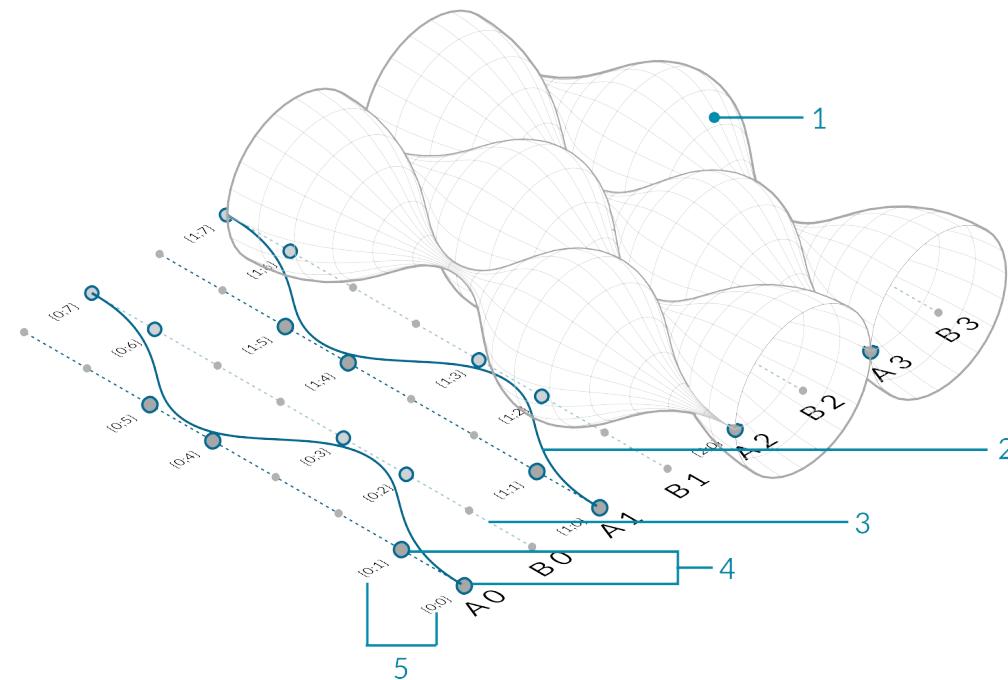


1. Компонент Path Mapper
2. Кликните правой клавишей по компоненту Path Mapper и выбрать заранее заданную опцию переноса из меню или откройте Mapping редактор.
3. Mapping редактор
4. Вы можете изменить дерево данных путем еще одного переноса индекса пути и нужной ветки.

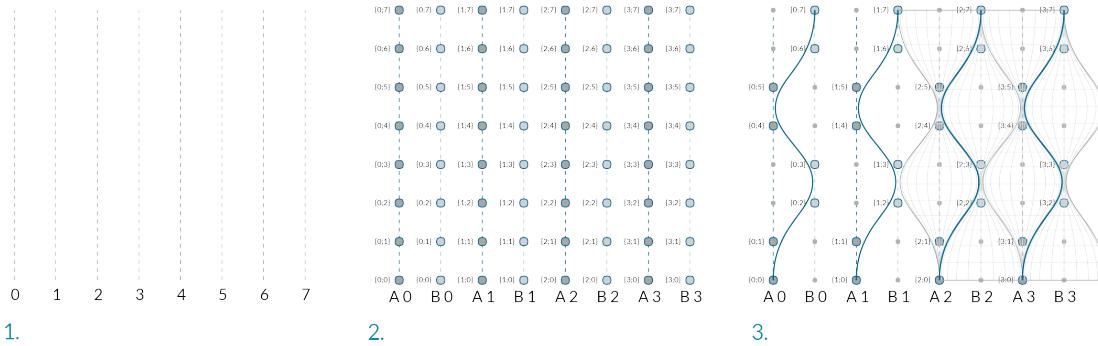
1.5.3.6. WEAVING DEFINITION

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

В этом примере мы будем работать со списками и деревьями данных, чтобы соединить списки точек, определить паттерн и создать поверхность геометрии.



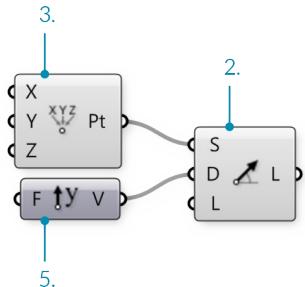
1. Вращающаяся NURBS поверхность
2. NURBS кривая
3. Массив кривой
4. Точки деления
5. Пути (индексы) точек



1. Массив кривых
2. Разделение кривых на списки А и В, разделение кривых
3. Cull points, Weave и Revolve

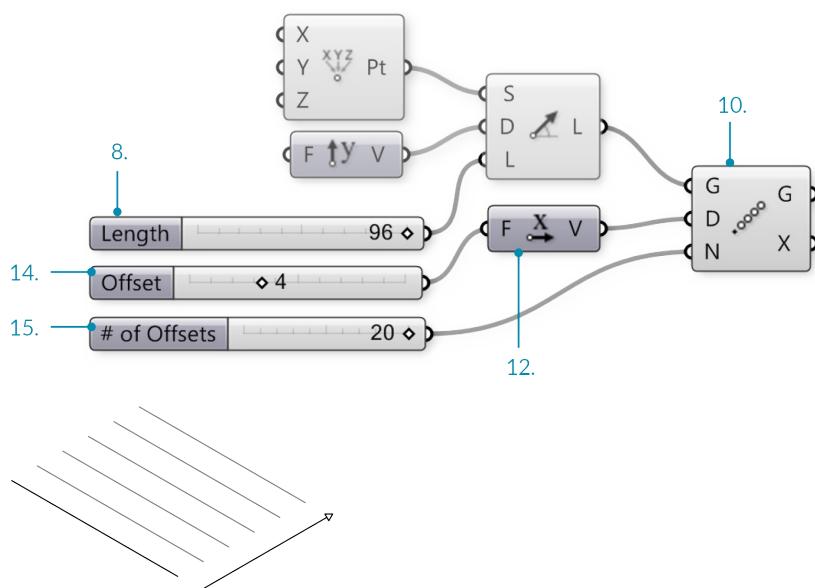
01.	Запустите новое определение, набрав Ctrl+N (в Grasshopper)	
02.	Зайдите в Curve/Primitive/Line SDL – перетащите компонент Line SDL на холст	
03.	Зайдите в Vector/Point/Construct Point – перетащите компонент Construct Point на холст	
04.	Соедините выход Point (Pt) компонента Construct Point с входом Start (S) компонента Line SDL	
05.	Зайдите в Vector/Vector/Unit Y – перетащите компонент vector Unit Y на холст Коэффициент компонентов Unit Vector, по умолчанию, 1.0.	

05.	Коэффициент компонентов Unit Vector, по умолчанию, 1.0.
06.	Соедините компонент Unit Y с входом Direction (D) компонента Line SDL



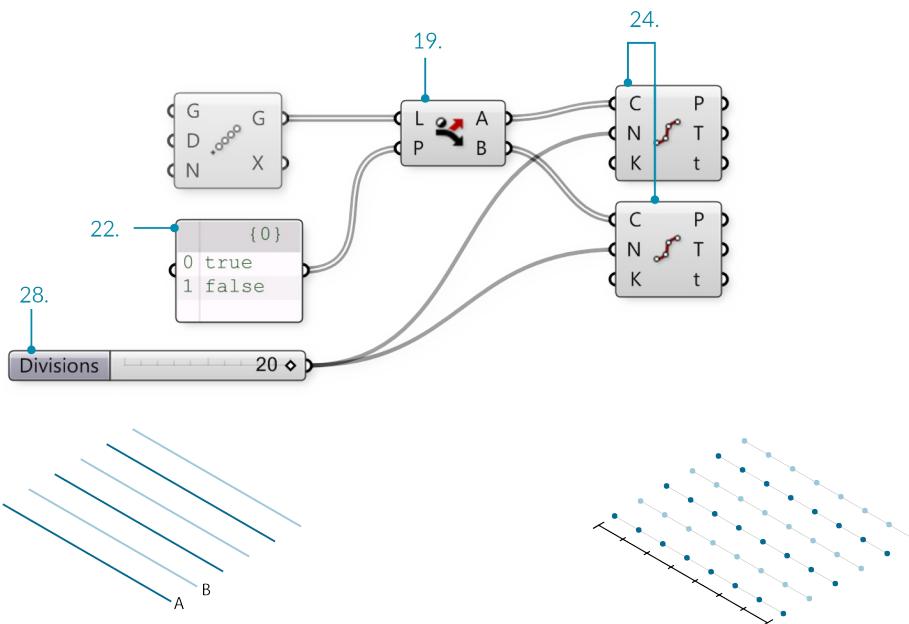
07.	Зайдите в Params/Input/Number Slider – перетащите компонент Number Slider на холст
08.	Дважды кликните по Number Slider и установите следующее: Name: Length Rounding: Integer Lower Limit: 0 Upper Limit: 96 Value: 96
09.	Подключите Number Slider к входу Length (L) компонента Line SDL
10.	Зайдите в Transform/Array/Linear Array – перетащите компонент Linear Array на холст
11.	Соедините выход Line (L) компонента Line SDL с входом Geometry (G) компонента Linear Array
12.	Зайдите в Vector/Vector/Unit X – перетащите компонент vector Unit X на холст
13.	Зайдите в Params/Input/Number Slider – перетащите два компонента Number Slider на холст
14.	Дважды кликните по первому Number Slider установите следующее: Name: Offset Distance (Дистанция смещения) Rounding: Integer Lower Limit: 1 Upper Limit: 10 Value: 4
15.	Дважды кликните по второму Number Slider и установите следующее: Name: # of Offsets Rounding: Even Lower Limit: 2 Upper Limit: 20 Value: 20
16.	Подключите Number Slider (Offset Distance) к входу Factor (F) компонента Unit X
17.	Соедините выход Vector (V) компонента Unit X с входом Direction (D) компонента Linear Array
	Соедините выход Number Slider (# of Offsets) с входом Count (N) компонента

18. Соедините выход **Number Slider (# of Offsets)** с входом Count (N) компонента **Linear Array**



Сейчас вы должны увидеть массив линий в видовом окне Rhino. Три слайдера позволят вам изменить длину этих линий, их дистанцию друг от друга и число линий в массиве.

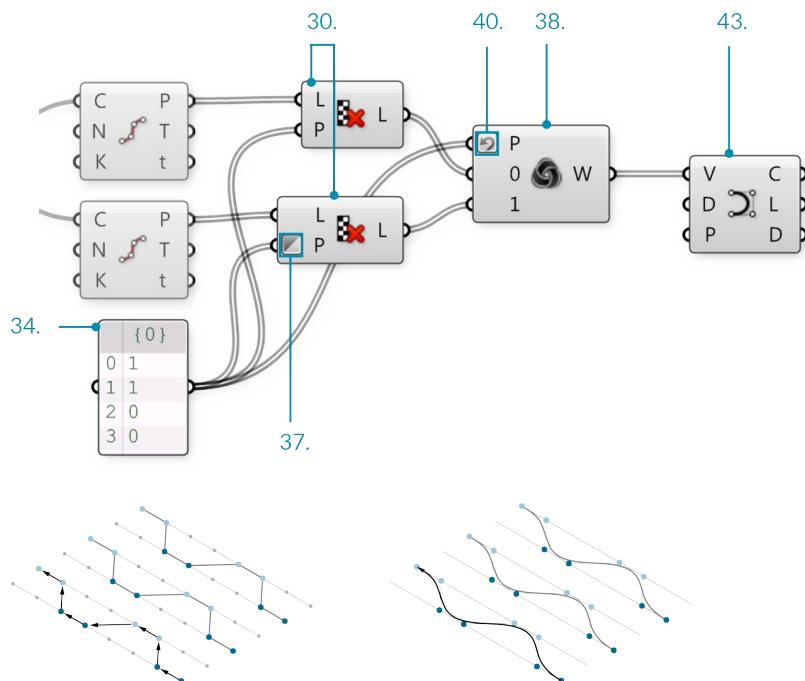
19.	Зайдите в Sets/Lists/Dispatch – перетащите компонент Dispatch component onto the canvasha холст	
20.	Соедините выход Geometry (G) компонента Linear Array с входом List (L) компонента Dispatch	
21.	Зайдите в Params/Input/Panel – перетащите компонент Panel на холст	
22.	Дважды кликните по Panel , снимите выделение с Multiline Data, Wrap Items и Special Codes, и введите следующее: true false	
23.	Соедините Panel с входом Pattern (P) компонента Dispatch	
24.	Зайдите в Curve/Division/Divide Curve – перетащите два компонента Divide Curve на холст	
25.	Соедините выход List A(A) компонента Dispatch с входом Curve (C) первого компонента Divide Curve	
26.	Соедините выход List B (B) компонента Dispatch с входом Curve (C) второго компонента Divide Curve	
27.	Зайдите в Params/Input/Number Slider – перетащите компонент Number Slider на холст	
28.	Дважды кликните по Number Slider и установите следующее: Name: Divisions Rounding: Integer Lower Limit: 0 Upper Limit: 20 Value: 20	



1. Компонент Dispatch отправляет каждую вторую кривую в массив на отдельный список.
2. Компонент Divide Curve разделяет кривые на число сегментов, определяемых слайдером.
Настройте слайдер, чтобы менять количество точек.

30.	Зайдите в Sets/Sequence/Cull Pattern – перетащите два компонента Cull Pattern на холст	
31.	Соедините выход Points (P) первого компонента Divide Curve с входом List (L) первого компонента Cull Pattern	
32.	Соедините выход Points (P) второго компонента Divide Curve с входом List (L) второго компонента Cull Pattern*	
33.	Зайдите в Params/Input/Panel – перетащите второй компонент Panel на холст	
34.	Дважды кликните на вторую Panel и снимите выделение с: Multiline Data, Wrap Items и Special Codes. Затем введите следующее: 1 1 0 0	
	Мы используем 1 и 0 вместо true и false. Grasshopper принимает два синтакса для булевых значений.	
35.	Соедините вторую Panel с входом Pattern (P) первого компонента Cull Pattern	
36.	Соедините вторую Panel с входом Pattern (P) второго компонента Cull Pattern	
37.	Кликните правой клавишей мыши по входу Pattern (P) второго компонента Cull Pattern и выберите Invert Это поможет инвертировать **Cull Pattern**, полезный трюк, чтобы ваше определение было компактным.	
38.	Зайдите в Sets/List/Weave – перетащите компонент Weave на холст	

38.	Зайдите в Sets/List/Weave – перетащите компонент Weave на холст	
39.	Соедините вторую Panel с входом Pattern (P) компонента Weave	
40.	Кликните правой клавишей по входу Pattern (P) компонента Weave и выберите reverse	
41.	Соедините выход List (L) первого компонента Cull Pattern с входом Stream 0 (0) компонента Weave	
42.	Соедините выход List (L) второго компонента Cull Pattern с входом Stream 0 (0) компонента Weave	
43.	Зайдите в Curve/Spline/Nurbs Curve – перетащите компонент Nurbs Curve на холст	
44.	Соедините выход Weave (W) компонента Weave с входом Vertices (V) компонента Nurbs Curve .	



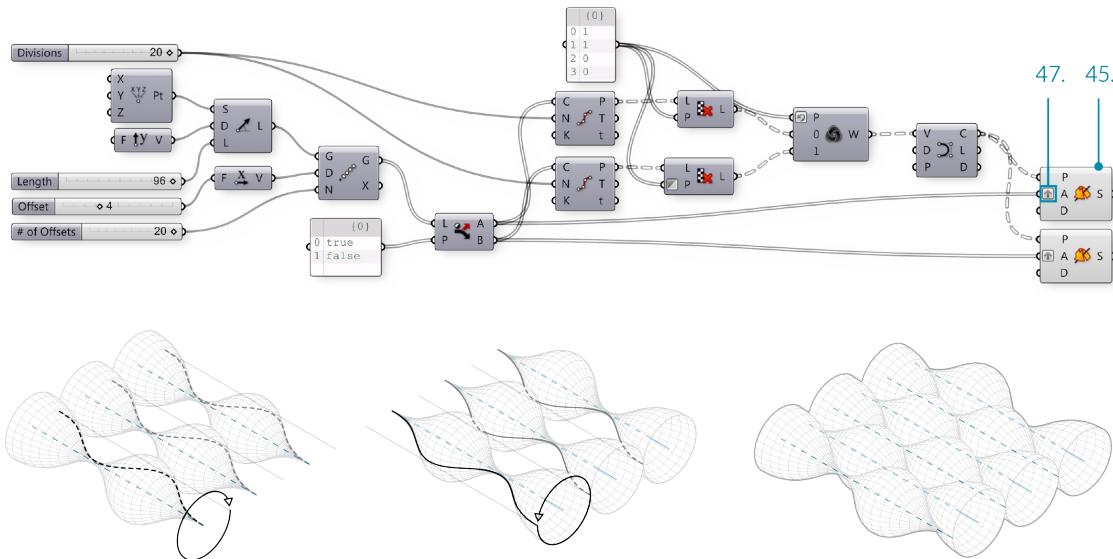
1. Cull patterns удаляют чередующиеся точки из каждого списка.
2. Компонент Weave собирает данные из списков точек в соответствии с настроенным паттерном. Эти данные передаются в компонент Interpolate для создания кривых.

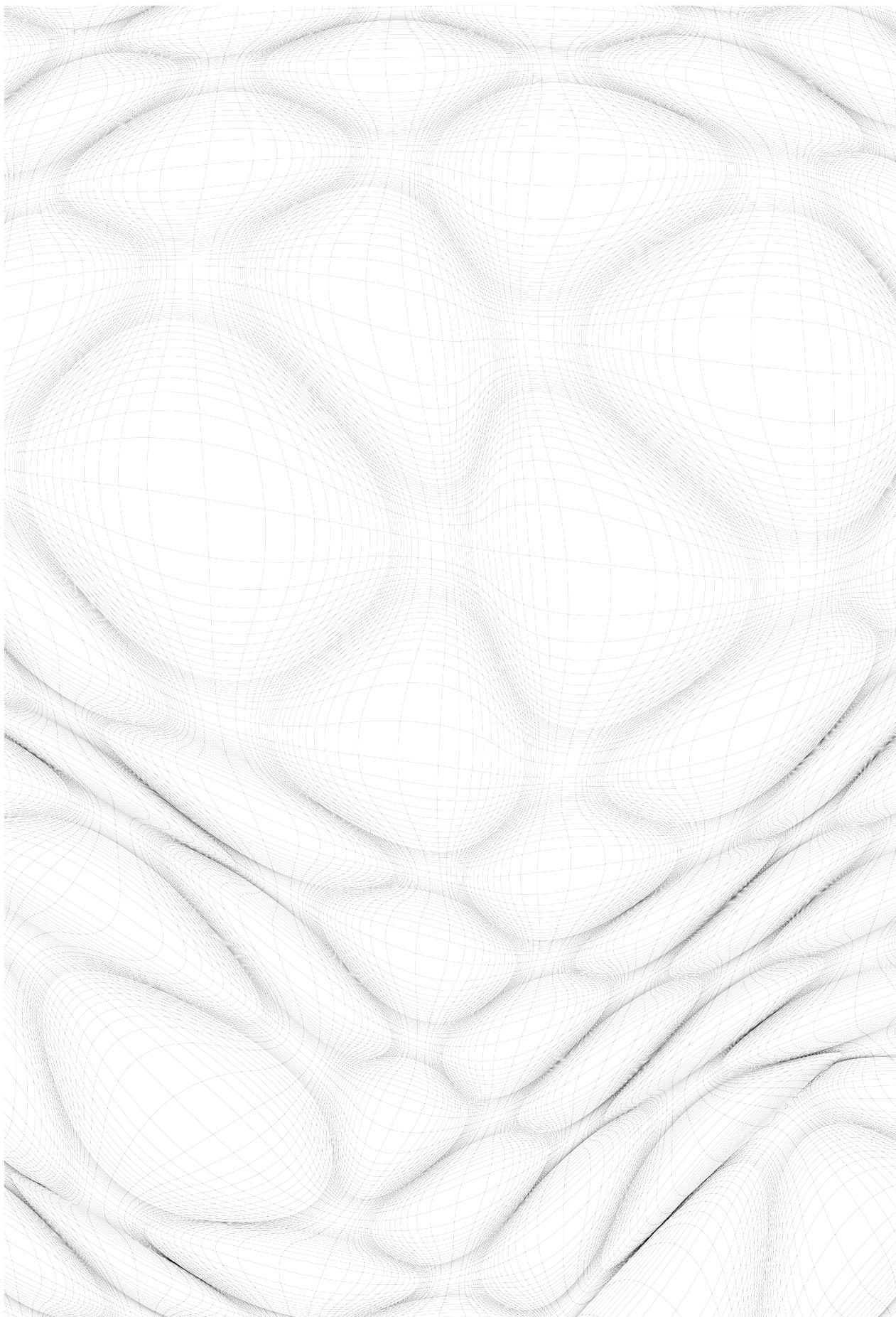
45.	Зайдите в Surface/Freeform/Revolution – перетащите два компонента Revolution на холст	
46.	Соедините выход Curve компонента Nurbs Curve с входом Profile Curve (P) обоих компонентов Revolution .	
47.	Кликните правой клавишей мыши по входу Axis (A) обоих компонентов Revolution и выберите Graft.	
48.	Соедините выход List A(A) компонента Dispatch с входом Axis (A) первого компонента Revolution	

Соедините выход List B (B) компонента **Dispatch** с входом Axis (A) второго компонента **Revolution**

49.

Выберите все компоненты, за исключением двух компонентов Revolution, и отключите предпросмотр - это помогает при создании определения для того, чтобы сфокусироваться на самой новой геометрии

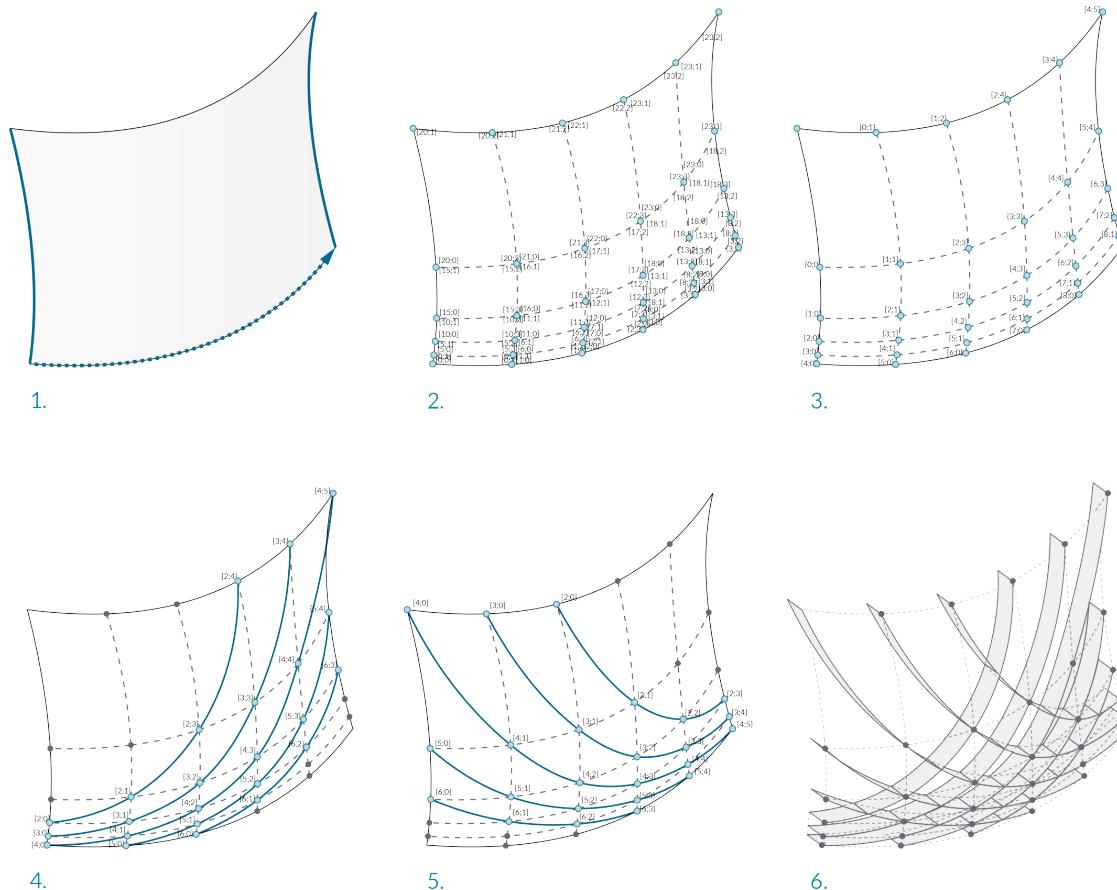




1.5.4. Работа с Деревьями Данных

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

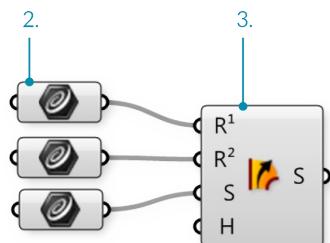
В этом примере, мы будем использовать некоторые инструменты Grasshopper для работы с деревьями данных для извлечения, реорганизации и интерполяции нужных точек, содержащихся в дереве данных и создания решетки пересекающихся ребер.



1. Проведите две перекладины для создания NURBS поверхности.
2. Разделите поверхность на сегменты, переменного размера, исключите вершины. Данные, собранные в один список с четырьмя элементами для каждого сегмента.
3. Сделайте Flip матрицы, чтобы изменить структуру данных. Данные, собранные из четырех списков, каждый содержит одну угловую точку каждого сегмента.
4. Разбейте дерево, чтобы соединить угловые точки и прорисовать диагональные линий через каждый сегмент.
5. Сократите дерево до cull веток, содержащих неполные точки, чтобы создать NURBS кривую третьего порядка и интерполировать точки.
6. Вытянуть кривые, чтобы создать пересекающиеся окончания.

01.	Запустите новое определение, набрав Ctrl+N (в Grasshopper)	
02.	Зайдите в Params/Geometry/Curve – перетащите три параметра curve на холст	

03.	Зайдите в Surface/Freeform/Sweep2 – перетащите компонент Sweep2 на холст	
04.	Кликните правой клавишей мыши по первому параметру Curve и выберите “Set one curve.” Выберите первую rail кривую в видовом окне Rhino	
05.	Кликните правой клавишей мыши по второму параметру Curve и выберите “Set one curve.” Выберите вторую rail кривую в видовом окне Rhino	
06.	Кликните правой клавишей мыши по третьему параметру Curve и выберите “Set one curve.” Выберите кривую в видовом окне Rhino	
07.	Соедините выходы параметров Curve с входами Rail 1 (R1), Rail 2 (R2) и Sections (S) компонента Sweep2 соответственно	



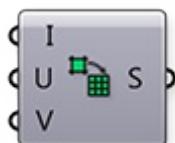
Мы только что создали NURBS поверхность

08.	Зайдите в Params/Geometry/Surface – вытащите параметр Surface на холст	
09.	Соедините выход Brep (S) компонента Sweep2 с входом параметра Surface	
10.	Кликните правой клавишей мыши по параметру Surface и выберите “Reparameterize”. На этом шаге мы заново перенесли диапазоны и и в поверхности между 0 и	

1. Это позволит совершить дальнейшие операции.</blockquote>|



11. Зайдите в Maths/Domain/Divide Domain2 – перетащите компонент Divide Domain2 на холст



|12.| Зайдите в **Params/Input/Number Slider** – вытащите два слайдера **Number Sliders** на холст|13.|

Дважды кликните по первому **Number Sliders** и установите следующее:

Rounding: Integer

Lower Limit: 1

Upper Limit: 40

Value: 20

|| 14. Установите такие же значения на втором **Number Sliders** || 15. Соедините выход

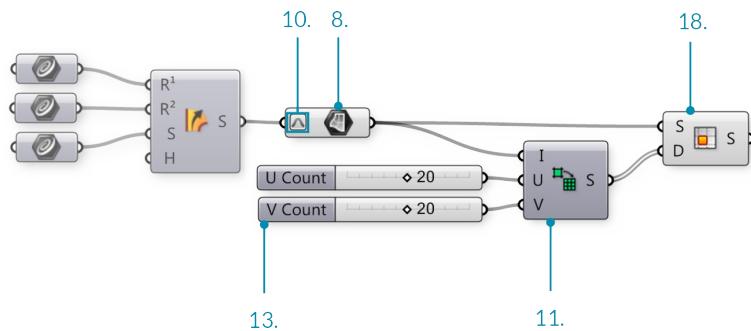
репараметризованого параметра **Surface** с входом **Domain (I)** компонента **Divide Domain2** || [16.]

Подключите первый **Number Sliders** к входу U Count (U) в компоненте **Divide Domain2** | [17.] Подключите

второй **Number Sliders** к входу V Count (V) в компоненте **Divide Domain2** || |18.| Зайдите в **Surface/Util/Isotrim** – вытащите компонент **Isotrim** на холст



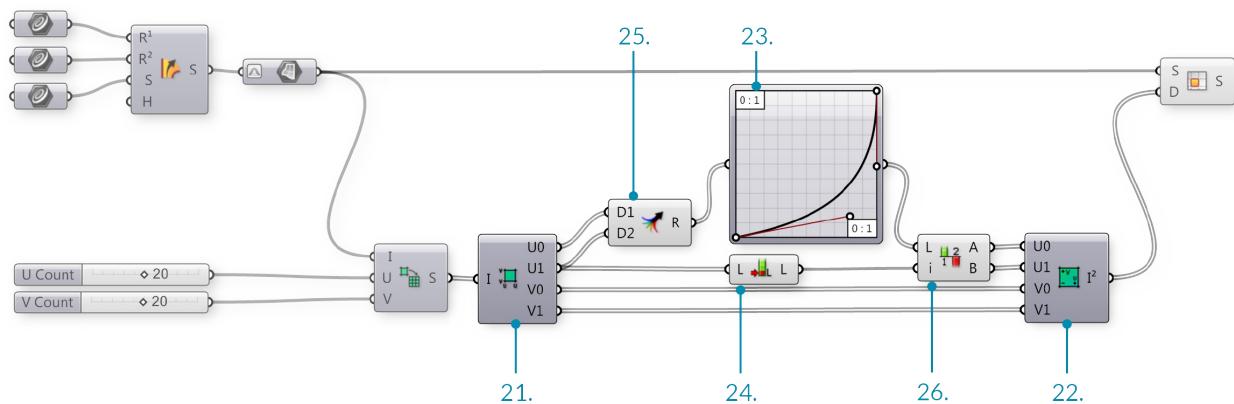
|| |19.| Соедините выход Segments (S) компонента **Divide Domain2** с входом Domain (D) компонента **Isotrim**|| |20.| Соедините выход параметра **Surface** с входом Surface (S) компонента **Isotrim**|||



Сейчас мы разделили поверхность на небольшие, легко измеримые, поверхности. Настройте слайдеры U и V Count для изменения числа подразделений. Давайте добавим Graph Mapper, чтобы размеры сегментов стали переменными.

21.	Зайдите Maths/Domain/Deconstruct Domain2 – вытащите компонент Deconstruct Domain2 на холст	
22.	Зайдите Maths/Domain/Construct Domain2 – вытащите компонент Construct Domain2 на холст	
23.	Зайдите в Params/Input/Graph Mapper – вытащите Graph Mapper на холст	
24.	Зайдите в Sets/List/List Length – вытащите компонент List Length на холст	
25.	Зайдите в Sets/Tree/Merge – вытащите компонент Merge на холст	
26.	Зайдите в Sets/List/Split List – вытащите компонент Split List на холст Компоненты Merge и Split используются, чтобы тот же самый Graph Mapper мог использоваться для обоих значений U min и U max.	
27.	Соедините выходы U min (U0) и U max (U1) компонента Deconstruct Domain2 с входами Data 1 (D1) и Data 2 (D2) компонента Merge	
28.	Соедините выход Result (R) компонента Merge с входом Graph Mapper	
29.	Кликните правой клавишей мыши по Graph Mapper и выберите “Bezier” в “Graph Types”	
30.	Соедините второй связью из выхода U max (U1) компонента Deconstruct Domain2 с входом List (L) компонента List Length	

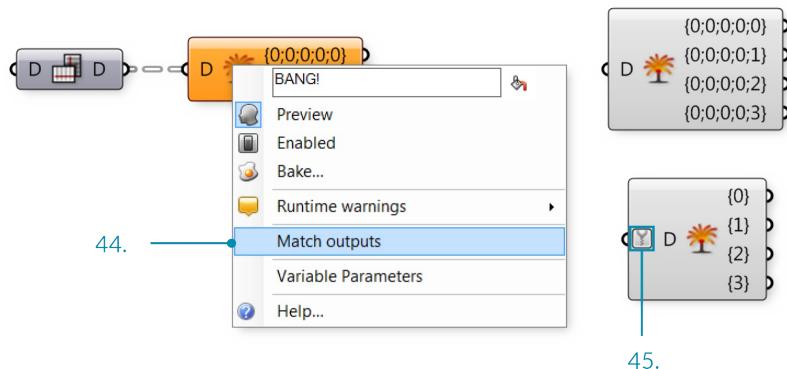
31. Соедините выход **Graph Mapper** с входом List (L) компонента **Split List**
32. Соедините выход Length (L) компонента **List Length** со входом Index (i) компонента **Split List**
33. Соедините выход List A(A) компонента **Split List** с входом U min (U0) компонента **Construct Domain2**
34. Соедините выход List B (B) компонента **Split List** с входом U max (U1) компонента **Construct Domain2**
35. Соедините выход V min (V0) компонента **Deconstruct Domain2** с входом V min (V1) компонента **Construct Domain2**
36. Соедините выход V max (V1) компонента **Deconstruct Domain2** с входом V max (V1) компонента **Construct Domain2**
37. Соедините выход 2D Domain (l2) компонента **Construct Domain2** с входом Domain (D) компонента **Isotrim**, заменяя существующую связь



Мы только что поменяли строение диапазонов каждого сегмента поверхности, заново перенесли значения U, используя Graph Mapper, и реконструировали диапазоны. Настройте ползунки Graph Mapper чтобы изменить распределение сегментов поверхности. Давайте используем Деревья Данных для работы с подразделениями поверхности.

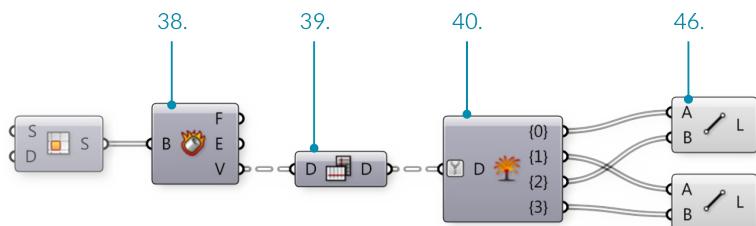
38.	Зайдите в Surface/Analysis/Deconstruct Brep – вытащите компонент Deconstruct Brep вытащите компонент	
39.	Зайдите в Sets/Tree/Flip Matrix – вытащите компонент Flip Matrix на холст	
40.	Зайдите в Sets/Tree/Explode Tree – вытащите компонент Explode Tree на холст	
41.	Соедините выход Surface (S) компонента Isotrim с входом Brep (B) компонента Deconstruct Brep	
	Компонент Deconstruct Brep деконструирует Brep в Полигоны, Ребра и Вершины. Это полезно, если вы хотите работать со специфически сложенной поверхностью.	
42.	Соедините выход Vertices (V) компонента Deconstruct Brep с входом Data (D) компонента Flip Matrix	
	Мы изменили структуру Дерева Данных из одного списка из четырех вершин, которые определяют каждую поверхность, в четыре списка, каждый содержит одну вершину каждой поверхности.	

43.	Соедините выход Data (D) компонента Flip Matrix с входом Data (D) компонента Explode Tree	
44.	Кликните правой клавишей мыши по компоненту Explode Tree и выберите "Match Outputs"	
45.	Кликните правой клавишей мыши по входу Data (D) компонента Explode Tree и выберите simplify	



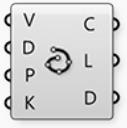
Каждый выход компонента **Explode Tree** содержит список из одной вершины каждой поверхности. Другими словами, один список со всеми верхними углами справа, один список со всеми нижними углами справа, одни списки с верхними левыми углами и один список с нижними левыми углами.

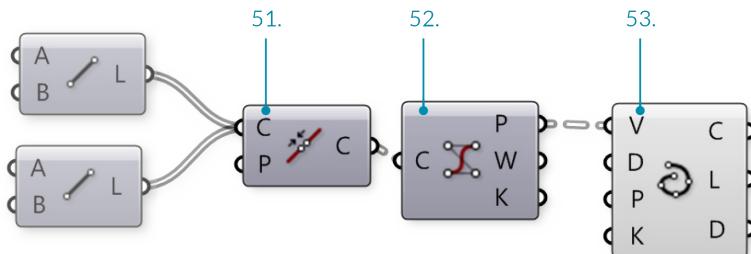
46.	Зайдите в Curve/Primitive/Line – вытащите два компонента Line на холст	
47.	Соедините выход Branch 0 {0} компонента Explode Tree с входом Start Point (A) первого компонента Line	
48.	Соедините выход Branch 1 {1} компонента Explode Tree с входом Start Point (A) второго компонента Line	
49.	Соедините выход Branch 2 {2} компонента Explode Tree с входом End Point (B) первого компонента Line	
50.	Соедините выход Branch 3 {3} компонента Explode Tree с входом End Point (B) второго компонента Line	



Сейчас мы соединили угловые точки каждой поверхности диагонально с линиями.

51.	Зайдите в Curve/Util/Join Curves – вытащите компонент Join Curves на холст	
52.	Зайдите в Curve/Analysis/Control Points – вытащите компонент Control Points на холст	

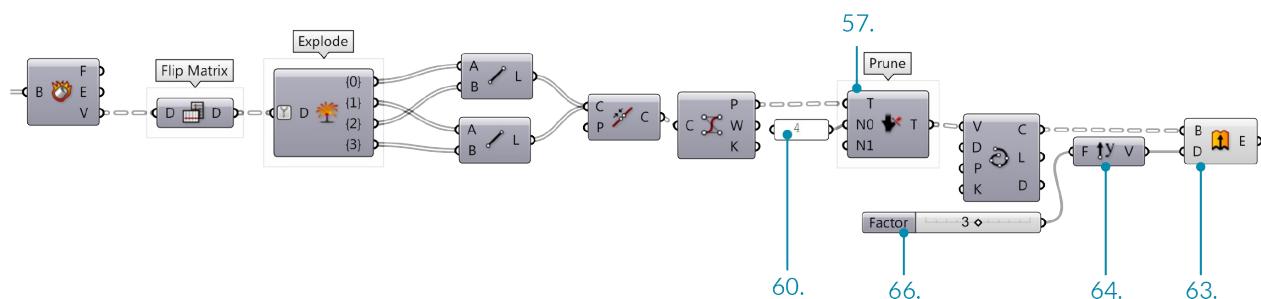
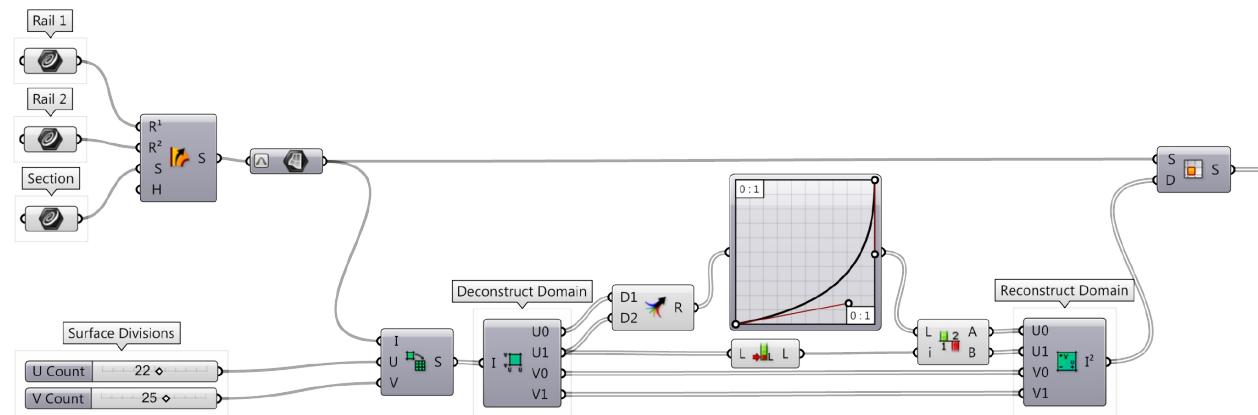
	53. Зайдите в Curve/Spline/Interpolate – вытащите компонент Interpolate на холст	
54.	Соедините выходы Line (L) каждого компонента Line с входом Curves (C) компонента Join Curves Зажмите клавишу Shift, чтобы подключить множественные связи к одному входу	
55.	Соедините выход Curves (C) компонента Join Curves с входом Curve (C) компонента Control Points	
56.	Соедините выход Points (P) компонента Control Points с входом Vertices (V) компонента Interpolate	



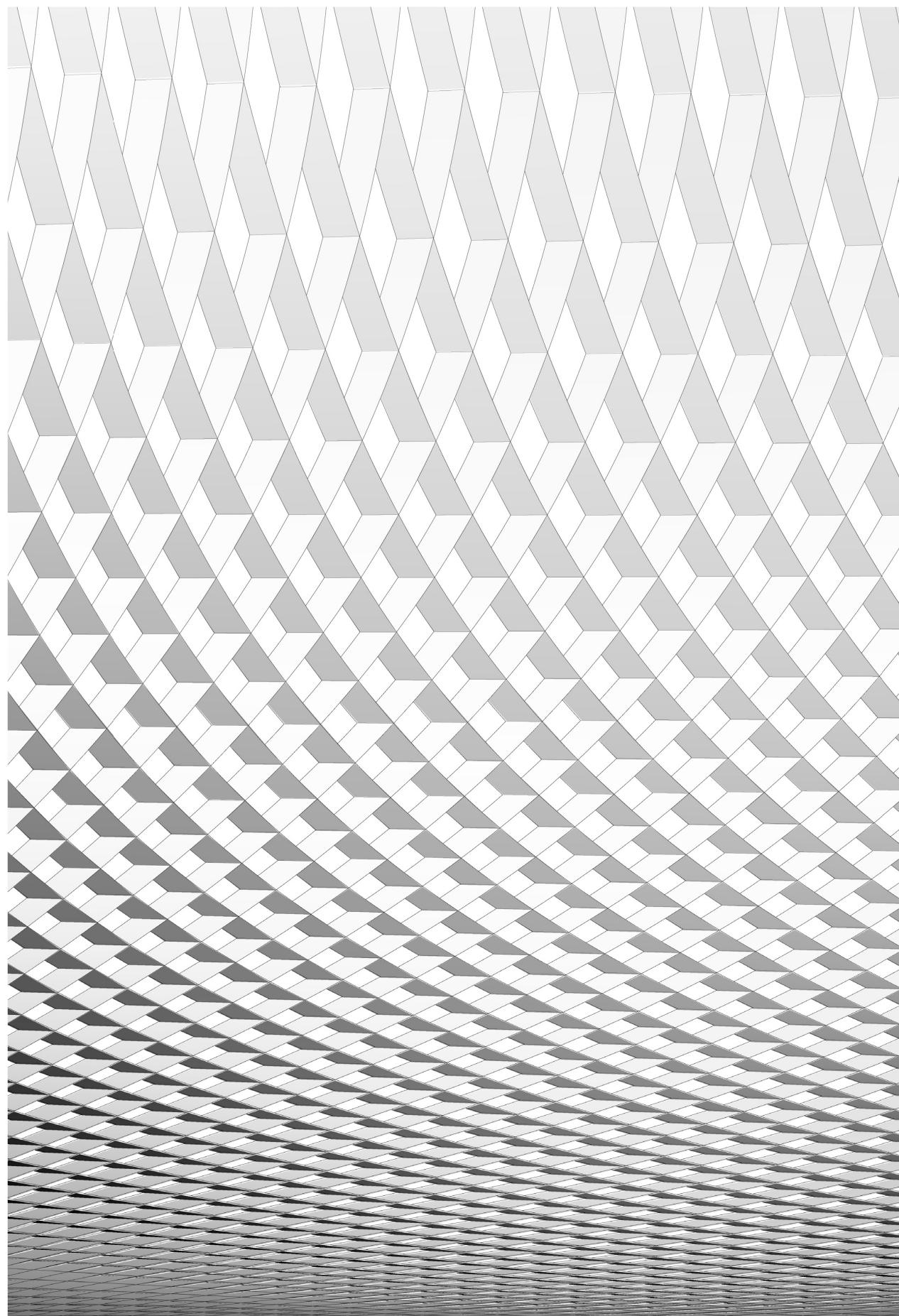
Сейчас мы соединили наши линии в полилинию и реконструировали их как NURBS кривые путем интерполяции их контрольных точек. В видовом окне Rhino, вы могли заметить, что более короткие кривые - это прямые линии. Это потому, что вы не можете создать NURBS кривую третьего порядка с менее, чем четырьмя контрольными точками. Давайте поработаем с деревом данных, чтобы устраниТЬ списки контрольных точек с менее, чем четырьмя элементами.

	57. Зайдите в Sets/Tree/Prune Tree – вытащите компонент Prune Tree на холст	
58.	Зайдите в Params/Input/Panel – вытащите Panel на холст	
59.	Соедините выход Points (P) компонента Control Points с входом Tree (T) компонента Prune Tree Если вы подключите один Param Viewer к входу Points (P) компонента Control Points и другой вход Tree (T) компонента Prune Tree, вы увидите, что число веток сократилось.	
60.	Дважды кликните по Panel и введите значение 4.	
61.	Соедините выход Panel с входом Minimum (N0) компонента Prune Tree	
62.	Соедините выход Tree (T) компонента Prune Tree с входом Vertices (V) компонента Interpolate	
63.	Зайдите в Surface/Freeform/Extrude – вытащите компонент Extrude на холст	
	Зайдите в Vector/Vector/Unit Y – вытащите компонент Unit Y на холст	

64.	Вам может понадобиться использовать вектор Unit X , в зависимости от ориентации вашей исходной геометрии в Rhino	
65.	Зайдите в Params/Input/Number Slider – вытащите слайдер Number Slider на холст	
66.	Дважды кликните по Number Slider и установите следующее: Rounding: Integer Lower Limit: 1 Upper Limit: 5 Value: 3	
67.	Соедините выход Curve (C) компонента Interpolate с входом Base (B) компонента Extrude	
68.	Подключите выход Number Slider к входу Factor (F) компонента Unit Y	
69.	Соедините выход Unit Vector (V) компонента Unit Y с входом Direction (D) компонента Extrude	

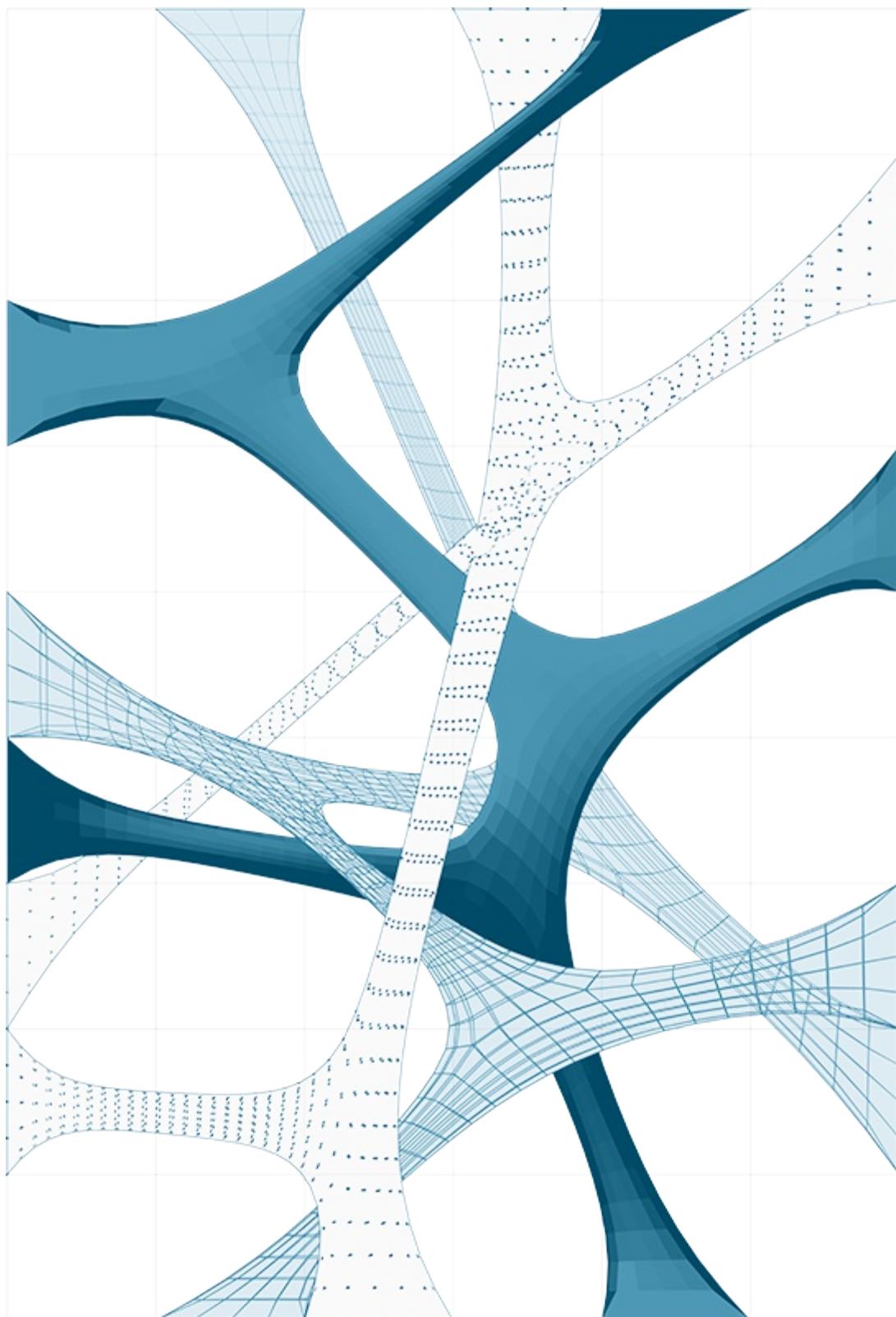


Сейчас вы должны увидеть диагональную сетку полосок или окончаний в видовом окне Rhino.
Настройте слайдер Factor, чтобы изменить глубину окончаний



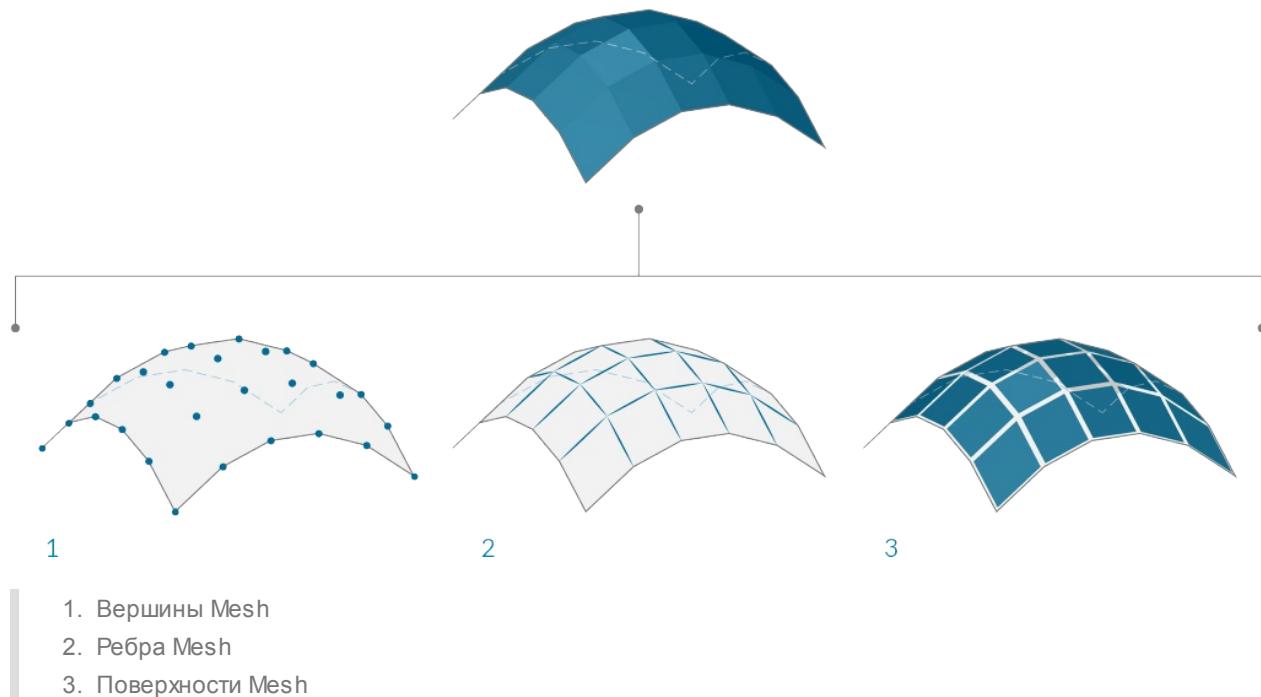
1.6. Начало работы с Mesh

В области вычислительного моделирования, mesh - одна из самых распространенных форм представления 3D геометрии. Геометрия Mesh может быть легковесной и гибкой альтернативой работе с NURBS и используется везде, начиная от рендера и визуализации и заканчивая цифровым производством и 3D печатью. Эта глава представляет собой введение в то, как работать с геометрией Mesh в Grasshopper.



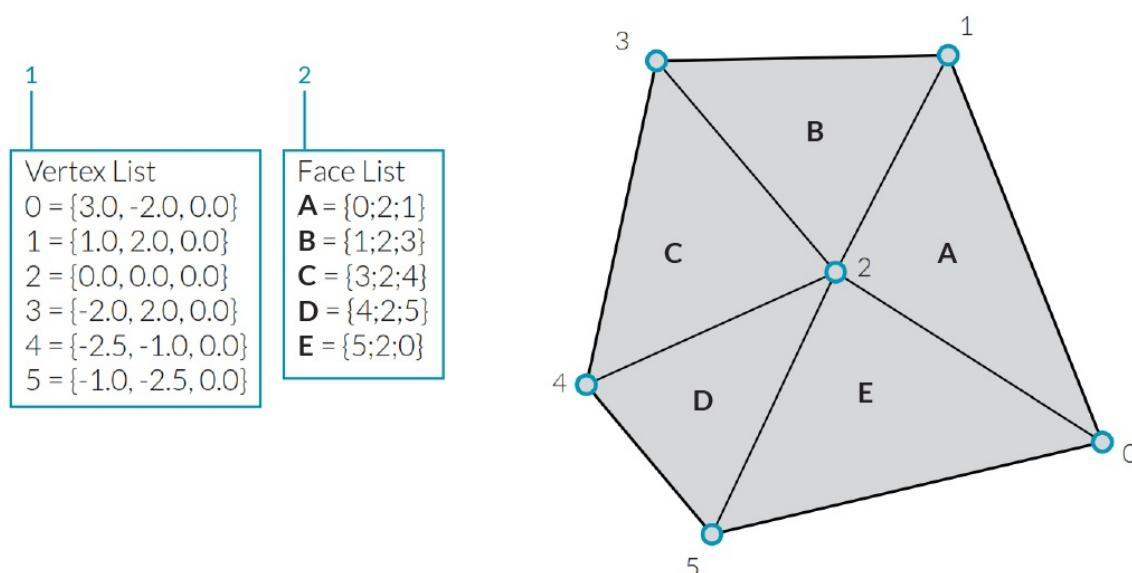
1.6.1 Что такое Mesh?

Mesh - это набор четырехугольников и треугольников, который образует поверхность или твердое тело. В этой секции обсуждается структура объекта **Mesh**, которая включает вершины, ребра и поверхности, а также дополнительные характеристики mesh, такие как цвет и нормали.



1.6.1.1 Основное строение Mesh

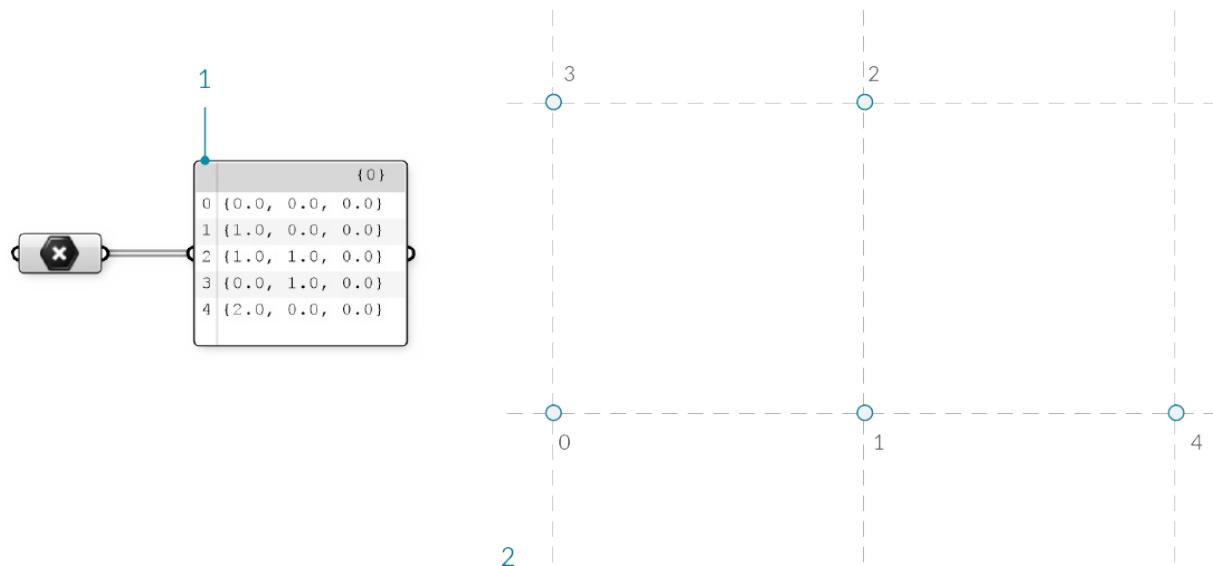
Grasshopper определяет mesh, используя структуру данных Face-Vertex. На самом примитивном уровне, это структура представляет собой набор точек, которые группируются в полигоны. Точки mesh называются *vertices* (вершины), а полигоны - *faces*. Чтобы создать mesh, нам требуется список вершин и система группирования этих вершин в полигоны.



1. Список вершин.
2. Полигоны с группами вершин

Вершины

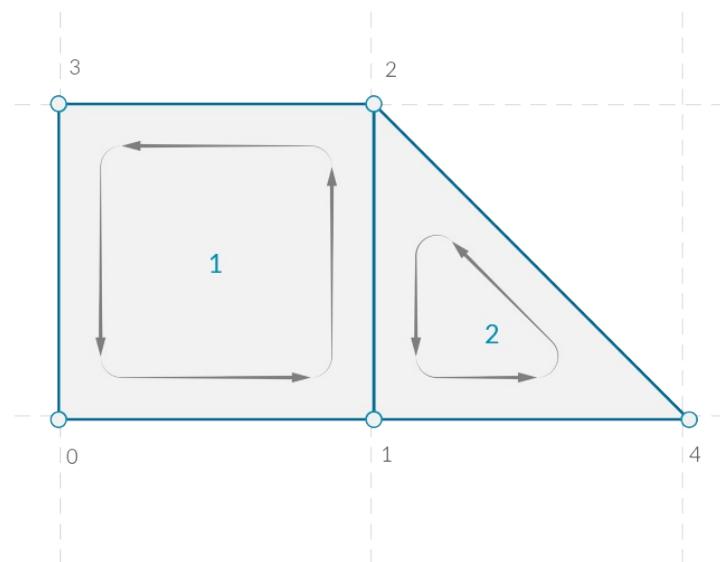
Вершины mesh - это просто список точек. Напомним, что список Grasshopper - это набор объектов. Каждый объект в списке, имеет индекс, который описывает положение объектов в списке. Индекс вершины имеет большую важность при создании mesh, при получении информации о структуре mesh.



1. Список точек. Все списки в Grasshopper начинаются с индекса ноль
2. Набор точек со своими индексами

Полигоны

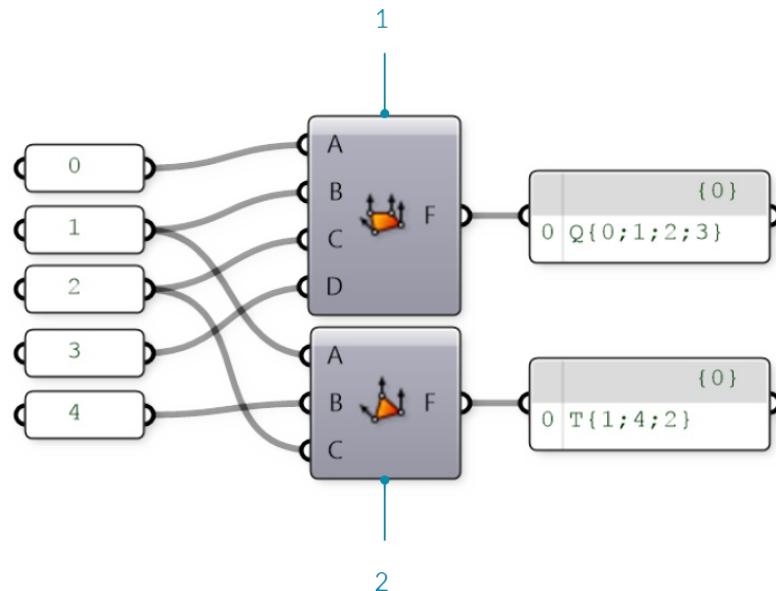
Face - это структурированный список из трех-четырех вершин. "Поверхностное" представление поверхности mesh означает в соответствии с положением вершины и ее индексом. У нас уже есть список вершин, которые составляют mesh, поэтому, вместо предоставления отдельных точек для определения face, мы просто используем индекс вершин. Это также позволяет нам использовать одну и ту же вершину в более чем одном полигоне.



1. Четырехугольный полигон состоит из индексов 0, 1, 2 и 3
2. Треугольный полигон состоит из индексов 1, 4 и 2

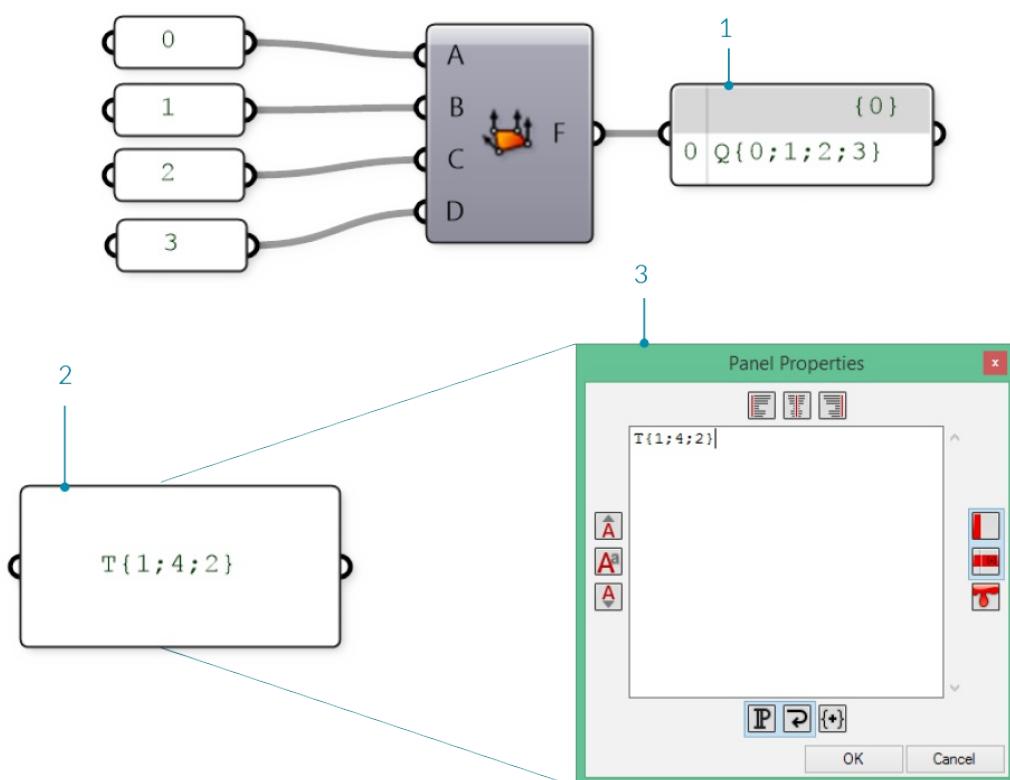
В Grasshopper face могут создаваться компонентами **Mesh Triangle** и **Mesh Quad**. Вводные параметры для

этих компонентов - целые числа, которые соответствуют индексу вершин, которые мы хотим использовать для создания полигона. Соединяя **Panel** с выходом этих компонентов, мы можем увидеть, что треугольный полигон представлен как $T\{A;B;C\}$ и четырехугольный полигон как $Q\{A;B;C;D\}$. Полигон с более чем 4 сторонами не допускаются. Чтобы создать 5-сторонний элемент mesh, mesh должна быть разбита на два или более полигонов.



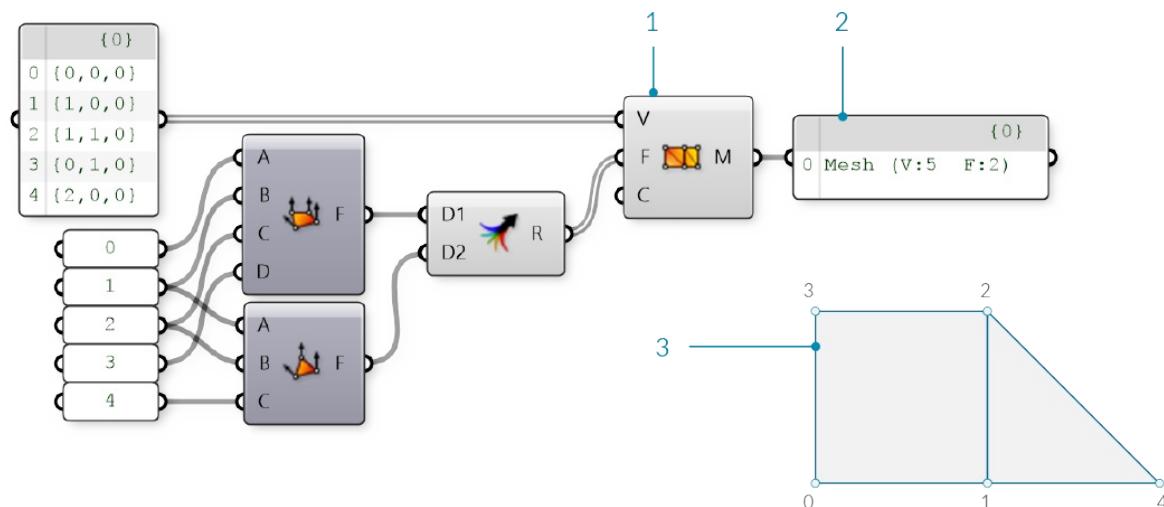
1. Компонент **Mesh Quad** с индексами 0, 1, 2 и 3
2. Компонент **Mesh Triangle** с индексами 1, 4 и 2

Важно помнить, что эти компоненты не создают геометрию mesh, также как и выход - это список индексов, которые определяют, как mesh должна создаваться. Уделяя внимание формату этого списка, мы также можем создать полигон вручную путем редактирования компонента **Panel** вводя подходящий формат для треугольных либо четырехугольных полигонов.



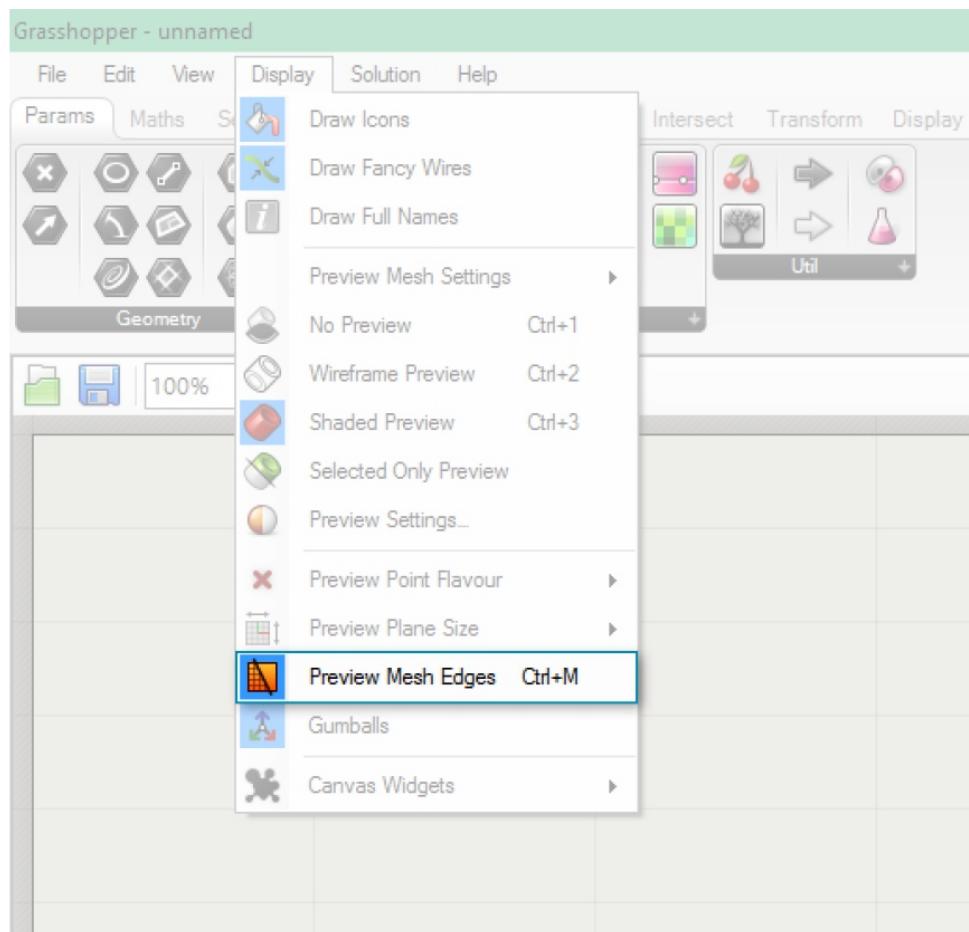
1. Полигон, созданный с использованием компонента **Mesh Quad**
2. Полигон, созданный с использованием **Panel**
3. Окно характеристик Panel автоматически открывается при двойном клике по панели при масштабировании или при правом клике по панели и выборе "Edit Notes..."

На данный момент у нас есть список вершин и набор файлов полигонов, но мы все еще не создали mesh. Чтобы создать mesh, нам нужно соединить полигон и вершины используя компонент **Construct Mesh**. Мы подключаем наш список вершин к входу V и смешанный список полигонов к входу F. (У компонента также есть дополнительный вход для цвета, мы обсудим это позже.) Если мы подключим panel к выходу **Construct Mesh** мы сможем увидеть информацию о числе полигонов и числе индексов.

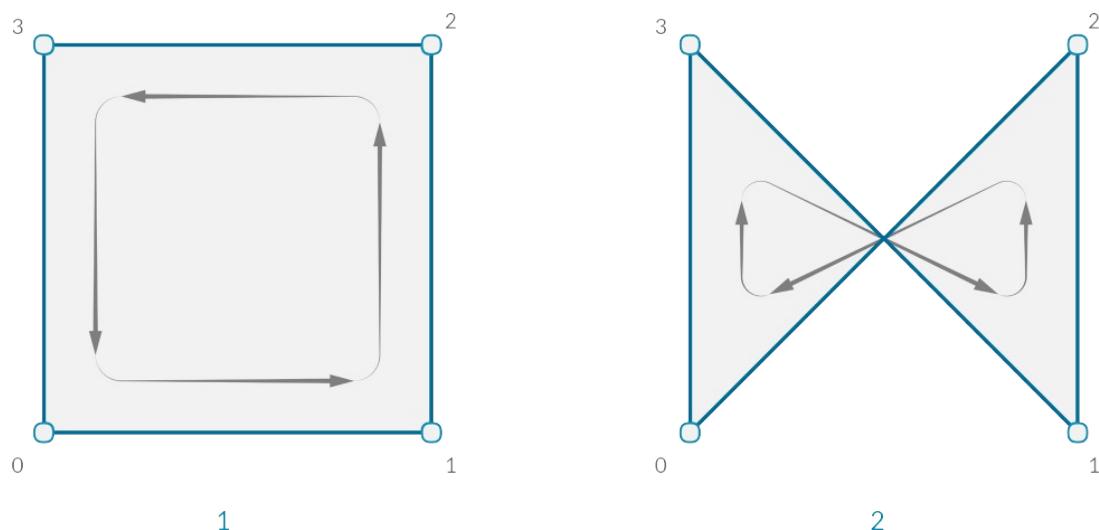


1. Компонент **Construct Mesh** берет список вершин и список полигонов как вводные данные. Вход Color опциональный, оставляется пустым пока что
2. Panel показывает, что мы создали mesh с 5 вершинами и 2 полигонами
3. Итоговая mesh (вершины были пронумерованы по их индексам)

По умолчанию, Grasshopper не показывает ребра геометрии mesh. Чтобы предварительно посмотреть на ребра и на поверхности, вы можете включить предпросмотр ребер mesh, используя комбинацию Ctrl-M или пройдя в меню Display и выбрав "Preview Mesh Edges".



Очень важно обратить внимание на порядок индексов при создании полигона `mesh`. Полигон будет создаваться при соединении пронумерованных вершин по порядку, так что четырехугольные полигоны $Q\{0,1,2,3\}$ и $Q\{1,0,2,3\}$ очень отличаются, несмотря на то, что используют одни и те же четыре вершины. Неправильное упорядочивание вершин может вести к проблемам, таким как щели (отверстия), неоднородная геометрия `mesh` либо неориентируемая поверхность. Такая геометрия `mesh` обычно неправильно рендерится и ее невозможно отправить на 3D печать. Эти вопросы обсуждаются более подробно в разделе **Understanding Topology**.



1. Четырехугольный полигон с индексами 0,1,2,3
2. Четырехугольный полигон с индексами 0,3,1,2

1.6.1.2 Скрытые данные Mesh

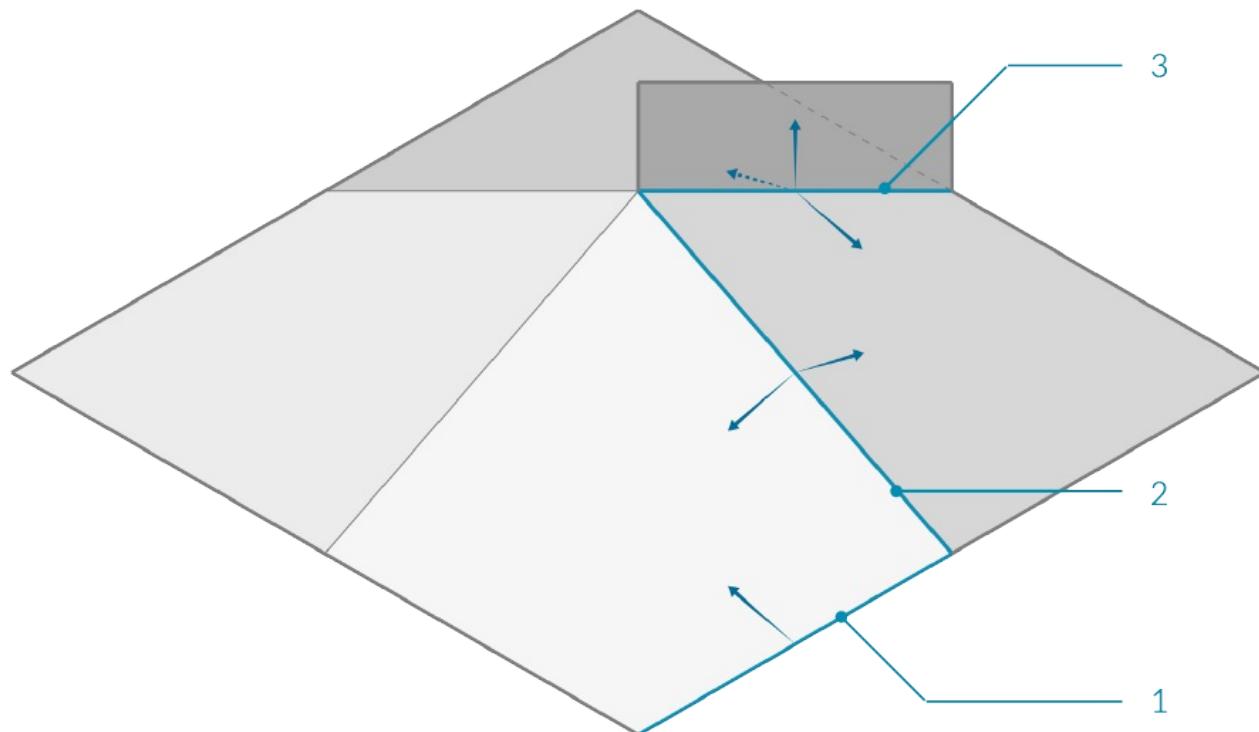
В дополнение к полигонам и вершинам, есть и другая информация о mesh, которую мы хотим использовать. В mesh, основанной на Face-Vertex, такие данные как *edges* (ребра) и *normals* (нормали) рассчитываются непрямо, на основе данных полигона и вершин. Этот раздел описывает способы получения этой информации.

Ребра

Ребра это линии, соединяющие любые две последовательные вершины в полигоне. Заметьте, что некоторые ребра принадлежат к нескольким полигонам, в то время как другие ребра принадлежат только одному полигону. Число полигонов, к которым ребра принадлежат, называется *valence* (валентность) этого ребра.

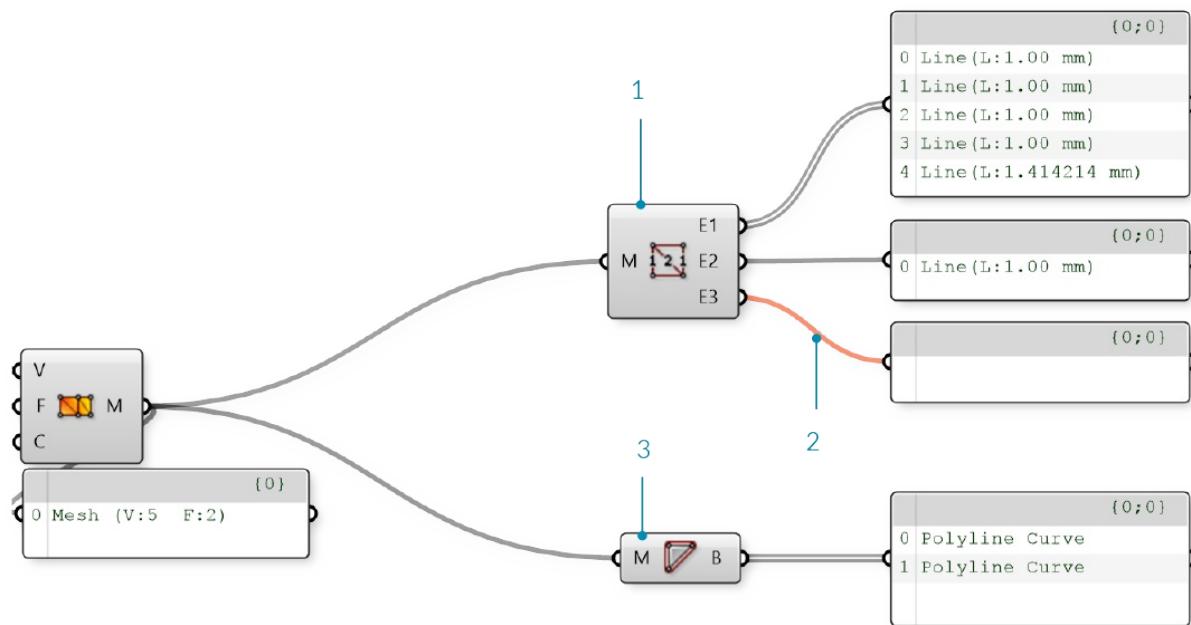
Grasshopper группирует ребра в три категории, на основе валентности:

1. E1 - 'Naked Edges' имеют валентность 1. Они состоят из внешней границы mesh.
2. E2 - 'Interior Edges' имеют валентность 2.
3. E3 - 'Non-Manifold Edges' имеют валентность 3 и выше. Mesh, которые содержат такие структуры, называются "Non-Manifold" (неоднородные) и обсуждаются в следующем разделе.



- 1. Naked edge имеют валентность 1
- 2. Interior edge имеют валентность 2
- 3. Non-manifold edge имеют валентность 3

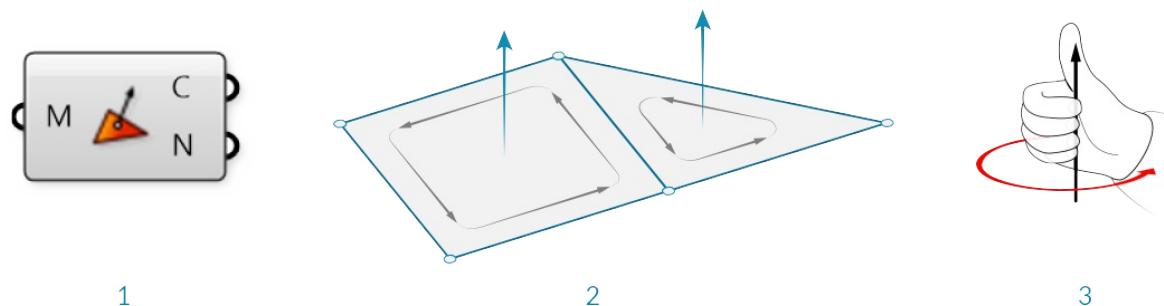
Мы можем использовать компонент **Mesh Edges** чтобы получить ребра mesh в соответствии с их валентностью. Это позволяет вам расположить ребра вдоль границы mesh или определить неоднородные ребра. Иногда, тем не менее, бывает более полезно иметь полное представление о граничных плоскостях каждого полигона. Для этого, мы можем использовать компонент **Face Boundaries**. Этот компонент извлекает полилинию для каждого полигона.



1. Компонент **Mesh Edges** выдает три набора ребер. Этот mesh имеет 5 naked, 1 interior и ноль non-manifold ребер
2. Выход E3 пустой, потому что у mesh нет non-manifold ребер, результатом чего становится оранжевая связь.
3. Компонент **Face Boundaries** выдает одну полилинию для каждого face

Нормали Поверхности

Normal vector (нормальный вектор) это вектор с амплитудой, которая перпендикулярна поверхности. В случае треугольных полигонов, мы знаем, что любые три точки должны быть плоскими, тогда нормаль будет перпендикулярна к этой плоскости, но как нам узнать в каком направлении (вверх или вниз) будет указывать нормаль? И снова порядок индексов является решающим. Полигоны mesh в Grasshopper определяется против часовой стрелки, поэтому полигон с индексами {0,1,2} будет "перевернут" по сравнению с индексами {1,0,2}. Другой способ визуализации - использовать *Right-Hand-Rule* (Правило правой руки).



1. Компонент **Face Normals** извлекает список точек начала координат и нормальных векторов для каждой face
2. Нормали полигона в соответствии с последовательностью вершин
3. "Правило правой руки" для определения направления нормали

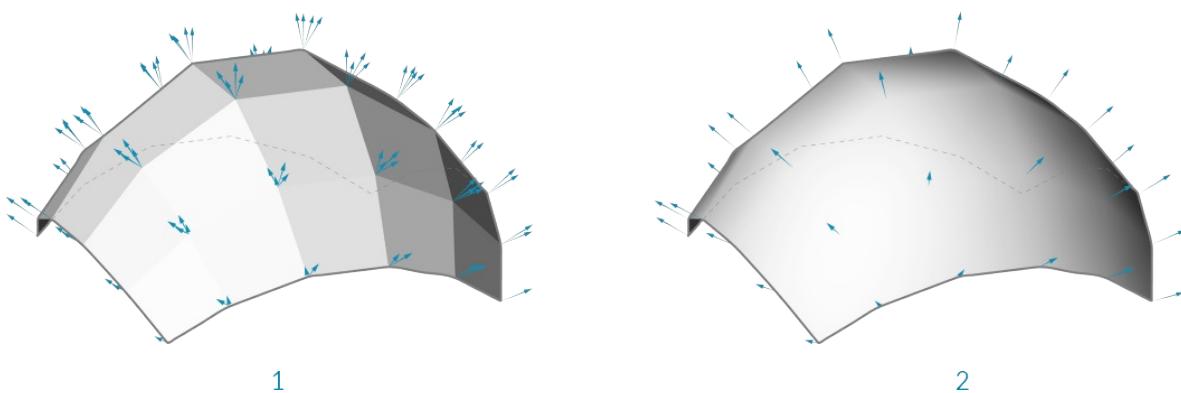
Grasshopper также допускает четырехугольные полигоны, в которых 4 точки не всегда будут плоскими. Для таких полигонов, точка начала координат будет просто средним координат 4-х вершин (в случае не плоских четырехугольников, заметьте, что эта точка не обязательно находится на mesh). Чтобы рассчитать нормаль четырехугольного полигона, нам необходимо сначала триангулировать четырехугольник разбив его на два плоских треугольника. Нормаль полного полигона, тогда становится, средним значение двух нормалей,

взвешенных в соответствии с областью двух треугольников.

Нормали Вершины

В дополнение к нормалям полигонов, также возможно высчитать нормали каждой вершины mesh. Для вершины, которая используется только в одном полигоне, нормаль вершины будет указывать в том же направлении, что и нормаль полигона. Если у вершины имеются многочисленные смежные полигоны, то нормаль вершины рассчитывается через среднее всех полигонов.

Хотя этот расчет менее интуитивный, чем у нормали полигонов, нормали вершины важны для сглаженной визуализации mesh. Вы могли заметить, что даже когда mesh состоит из плоских полигонов, такая mesh может все равно выглядеть сглаженной и округлой при затенении в Rhino. Использование нормали вершины позволяет получить такую сглаженную визуализацию.



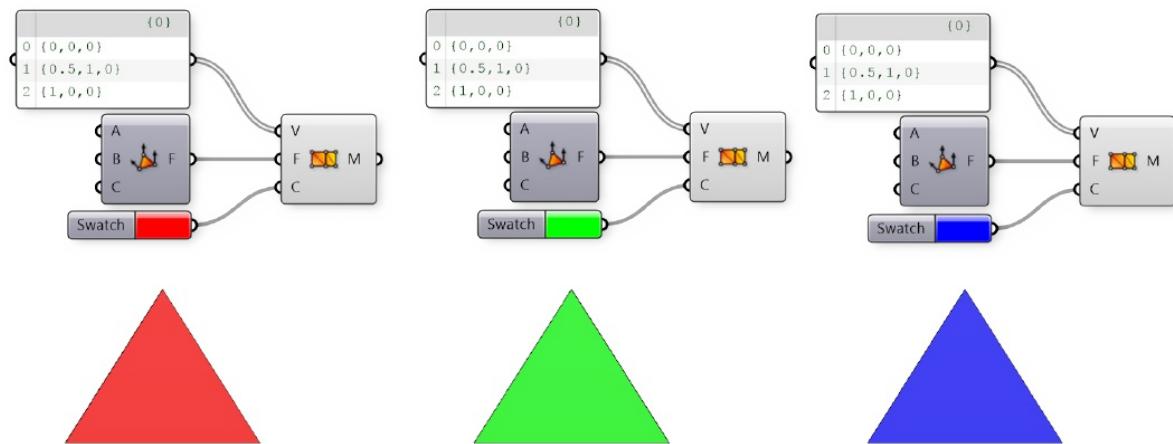
1. Нормали, установленные в соответствии с нормалями полигонов, дают в итоге в дискретное полигональное затенение
2. Смежные нормали полигонов усреднены вместе для создания нормали вершин, что приводит к сглаженному затенению полигона

1.6.1.3 Свойства Mesh

Mesh могут быть также приписаны дополнительные свойства либо вершин либо полигонов. Самое простое из этого - это цвет вершины, который описывается ниже, но существуют и другие свойства, такие как текстура UV координат. (Некоторые программы даже позволяют нормалям вершин быть назначеными как атрибуты вместо того, чтобы быть извлеченными из полигона и вершин, которые могут предоставить даже большее адаптивности во внешнем виде отрендеренной поверхности.)

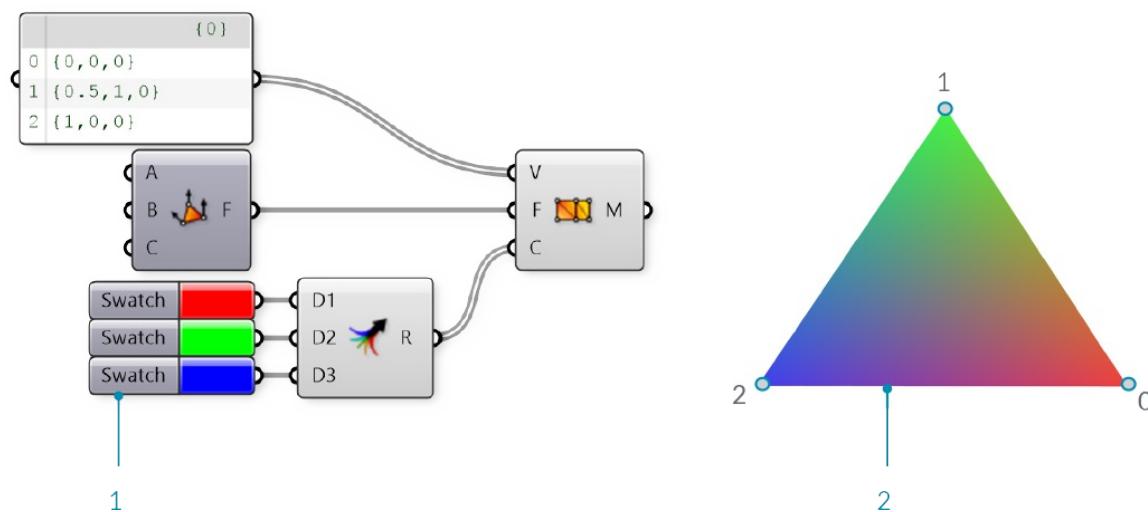
Цвет

При использовании компонента **Construct Mesh** существует дополнительный вход для цвета вершины. Цвета также могут быть приписаны существующей mesh используя компонент **Mesh Color**. Используя один цвет для mesh, мы можем закрасить mesh полностью.



Треугольные объекты mesh раскрашенные красным, зеленым или голубым

В то время как вышеуказанные примеры закрашивали полностью mesh, цветовые данные обычно приписываются к каждой вершине. Используя список из трех цветов, мы можем закрасить каждую вершину в треугольнике по отдельности. Эти цвета используются для визуализации, каждый полигон рендерится как интерполяция цветов вершин. Например, изображение ниже показывает треугольный полигон с цветами вершин красный, зеленый и голубой.

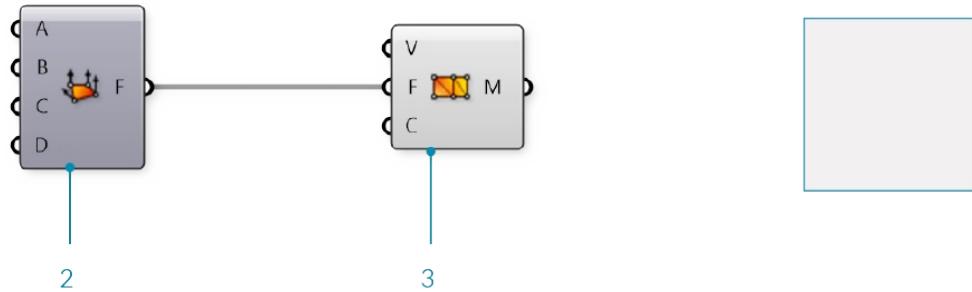


1. Красный, зеленый и голубой цвета приписаны к трем вершинам mesh
2. Итоговая mesh интерполирует цвета вершин

1.6.1.4 Упражнение

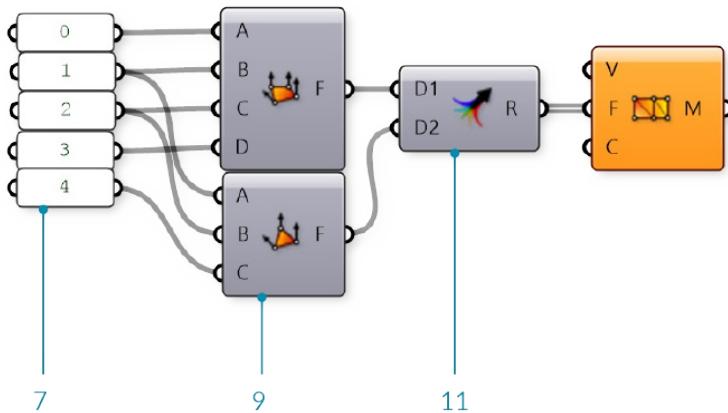
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

01.	Запустите новый файл набрав Ctrl-N (в Grasshopper)	
02.	Зайдите в Mesh/Primitive/Mesh Quad - перетащите компонент Mesh Quad на холст	
03.	Зайдите в Mesh/Primitive/Construct Mesh - перетащите компонент Construct Mesh на холст	
04.	Соедините выход Face (F) компонента Mesh Quad с входом Faces (F) компонента Construct Mesh	



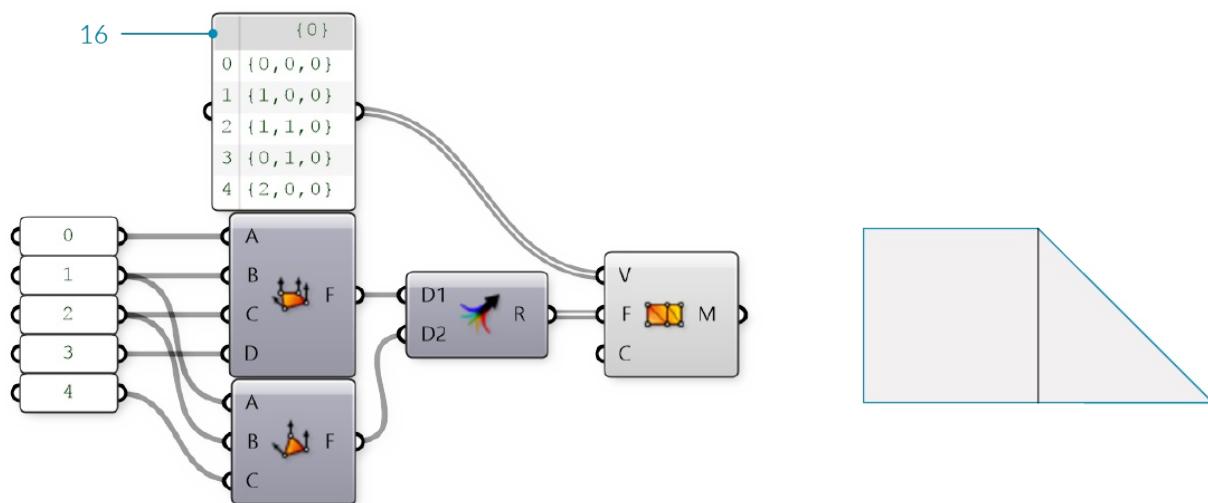
У **Mesh Quad** и **Construct Mesh** есть значения по умолчанию, которые создают один полигон у mesh. Затем, мы заменим значения по умолчанию нашими собственными значениями наших вершин и полигонов.

05.	Зайдите в Params/Input/Panel - перетащите компонент Panel на холст	
06.	Дважды кликните по компоненту Panel и установите значение "0"	
07.	Зайдите в Params/Input/Panel - перетащите на холст еще четыре компонента Panel и установите их значения на 1,2,3, и 4 Вы также можете скопировать первую **Panel** кликните и зажмите клавишу, перетащите, затем нажмите на клавишу Alt перед тем как отпустить клавишу	
08.	Соедините Panels и с входами компонента Mesh Quad в следующем порядке: 0 - A 1 - B 2 - C 3 - D	
09.	Зайдите в Mesh/Primitive/Mesh Triangle - перетащите компонент Mesh Triangle на холст	
10.	Соедините Panels с входами Mesh Triangle в следующем порядке: 1 - A 2 - B 4 - C	
11.	Зайдите в Sets/Tree/Merge - вытащите компонент Merge на холст	
12.	Соедините выход Face (F) компонента Mesh Quad с входом Data1 (D1) компонента Merge и выход Face (F) компонента Mesh Triangle с входом Data2 (D2) компонента Merge	
13.	Соедините выход Result (R) компонента Merge с входом Faces (F) компонента Construct Mesh	



По умолчанию список Vertices (V) компонента **Construct Mesh** имеет только 4 точки, но наш компонент **Mesh Triangle** использует индекс 4, который будет соответствовать пятой точке в списке. Так как вершин недостаточно, компонент **Construct Mesh** выдает ошибку. Чтобы ее исправить, мы предоставим наш список точек.

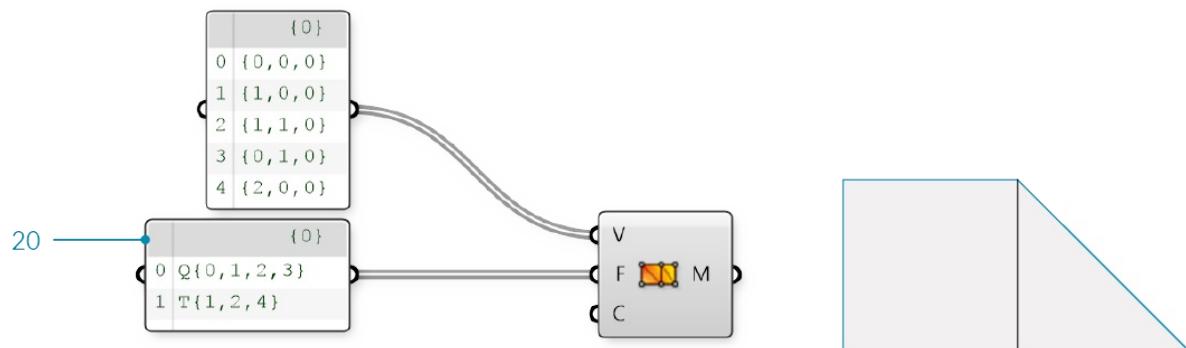
14.	Zайдите в Params/Input/Panel - вытащите компонент Panel на холст	
	Кликните правой клавишей мыши по компоненту Panel и снимите выделение с опции 'Multiline Data'	
15.	По умолчанию, у панели опция 'Multiline Data' включена. Отключая ее, каждая строчка с панели будет прочитываться как отдельный элемент внутри списка.	
	Дважды кликните по компоненту Panel для его редактирования, введите следующие точки: {0,0,0} {1,0,0} {1,1,0} {0,1,0} {2,0,0}	
16.	Убедитесь, что вы используете правильную систему чисел. Чтобы определить точку на **Panel**, вы должны использовать фигурные скобки: '{' и '}' с запятыми между значениями x, y и z	
	Соедините компонент Panel с входом Vertices (V) компонента Construct Mesh	



У нас сейчас есть mesh с двумя полигонами и 5 вершинами.

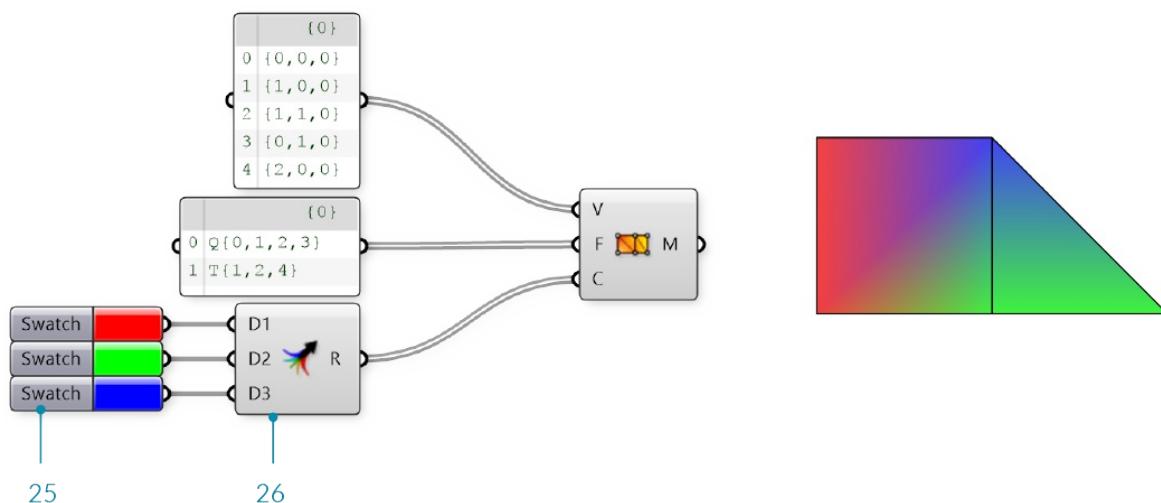
На выбор, мы можем заменить компоненты **Mesh Quad** и **Mesh Triangle** Панелью, указав индексы полигонов.

18.	Зайдите в Params/Input/Panel - вытащите компонент Panel на холст	
19.	Кликните правой клавишей мыши по компоненту Panel и снимите выделение с опции 'Multiline Data'	
20.	Либо скопируйте существующую **Panel** которую мы использовали для точек, у которых уже снято выделение с опции 'Multiline Data'	
21.	Дважды кликните по компоненту Panel для его редактирования, введите следующее: Q{0,1,2,3} T{1,2,4}	
21.	Соедините компонент Panel с входом Faces (F) компонента Construct Mesh	



22.	Зайдите в Params/Input/Colour Swatch - вытащите компонент Colour Swatch на холст	<input type="button" value="Swatch"/>
-----	--	---------------------------------------

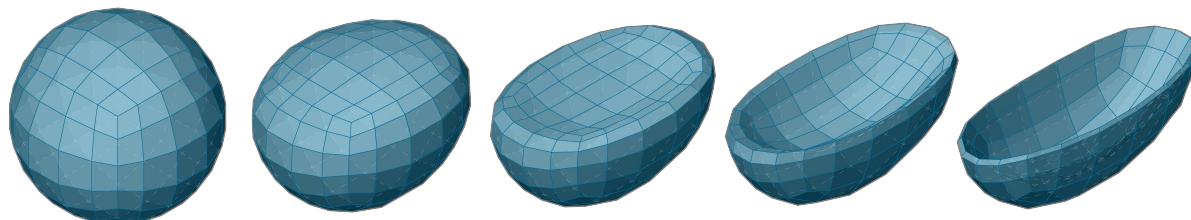
23.	Кликните по цветной секции компонента (по умолчанию указан белый цвет), чтобы открыть панель выбора цвета
24.	Используйте слайдеры, чтобы установить значение G и B на ноль. Палитра должна стать красной
25.	Зайдите в Params/Input/Colour Swatch - перетащите на холст еще два компонента Colour Swatch и настройте их цвета на Blue (голубой) и Green (зеленый)
26.	Зайдите в Sets/Tree/Merge - вытащите компонент Merge на холст
27.	Соедините три компонента Color Swatch с входами D1, D2, и D3 компонента Merge .
28.	Соедините выход Result (R) компонента Merge с входом Colours (C) компонента Construct Mesh



У нас есть 5 вершин и только 3 цвета. Grasshopper назначит цвета в соответствии с повторяющимся паттерном, так что в этом случае вершины 0 и 3 будут красного цвета, вершины 1 и 4 будут зеленого цвета, а вершина 2 будет голубого цвета.

1.6.2 Понимание Топологии

В то время как вершины mesh содержат информацию о расположении, связи между вершинами дают геометрии mesh ее уникальную структуру и гибкость.

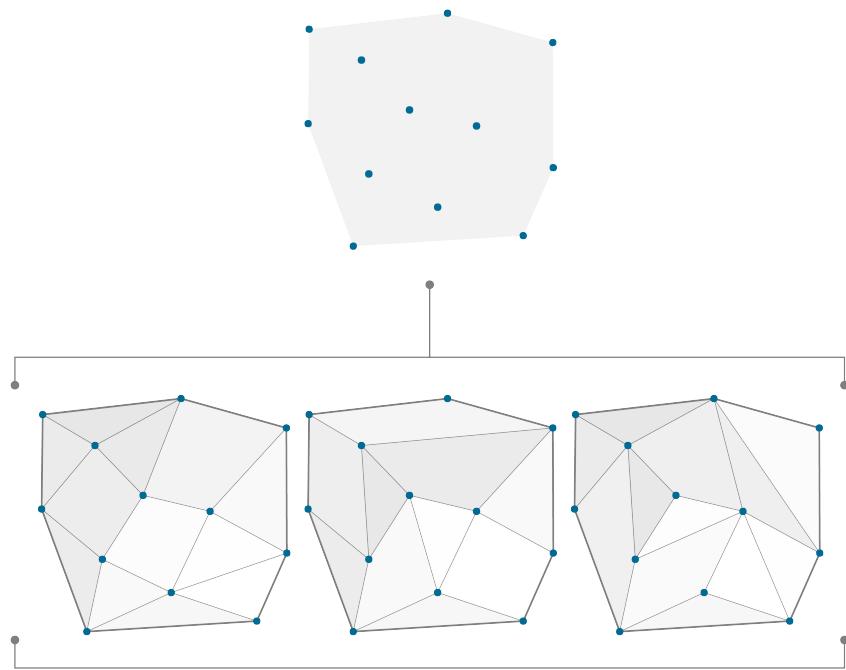


1.6.2.1 Что такое Топология?

Любое введение в геометрию mesh будет неполным без базовых знаний о топологии. Все потому, что топология занимается взаимосвязями и характеристиками набора "вещей", а не самими "вещами", топология привлекается для огромной сферы применения одновременно материального и нематериального характера. В этом пособии нам интересно базовое применение касательно параметрического процесса работы, которые предоставляет нам возможность создавать и контролировать геометрию mesh.

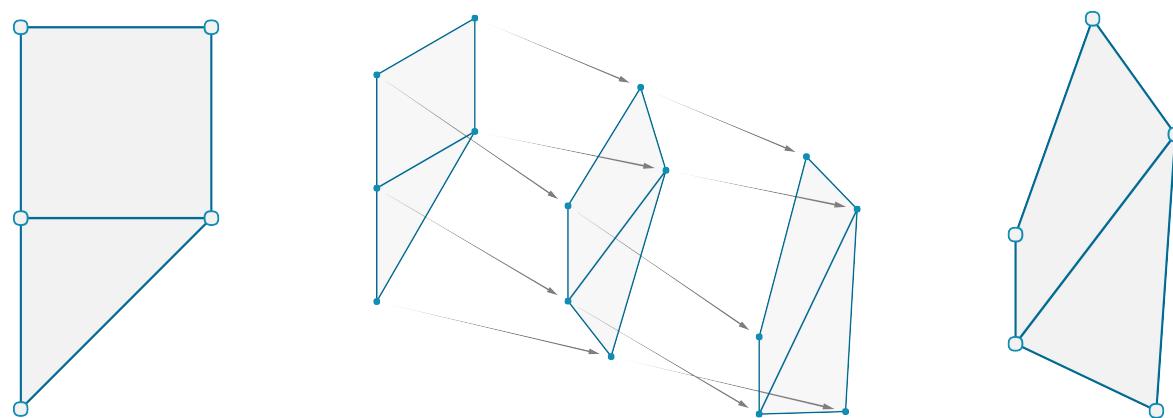
В Grasshopper требуются два основных типа информации для определения mesh - это геометрия и наличие связей; другими словами, набор точек в rhino-пространстве (вершины) и набор соответствующих точек-соединений (полигоны).

Без информации о наличии связей mesh неструктурированная и, потому, неопределенная. Введение в набор полигонов - это шаг (или прыжок), который, в конечном итоге, реализовывает mesh и создает ее характер с учетом непрерывности, конвергентности и связанности; эта структурная сеть называется *топологическим пространством*.



Такой же набор вершин может иметь разную связанность и, как следствие, разную топологию.

Гомеоморфность

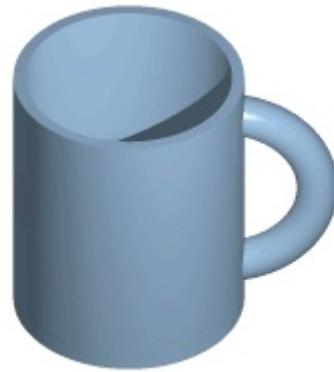


Точки mesh могут быть перемещены без изменения информации о наличии связей. Новая mesh имеет такую же топологию, как и исходная.

Это возможно для двух различных форм mesh быть топологически идентичными. Все это означает, что они созданы из одинакового набора точек и что точки структурированы одним и тем же набором полигонов. Ранее, мы установили, что полигон mesh интересуется только индексами набора точек, но не их фактическим положением в rhino-пространстве. Таким образом, если единственная разница между двумя различными mesh формами в особом 3-х пространственном положении точек, которые используются для ее определения, тогда две mesh считаются "гомеоморфными" (или "топологически эквивалентными") и, поэтому, имеют одни и те же топологические характеристики.

{A, R} {B} {C, G, I, J, L, M, N, S, U, V, W, Z}
 {D, O} {E, F, T, Y} {H, K} {P, Q} {X}

Пример гомеоморфности среди букв (заметьте, что некоторые из вышеуказанных гомеоморфных групп могут отличаться в зависимости от того, какой шрифт рассматривается)

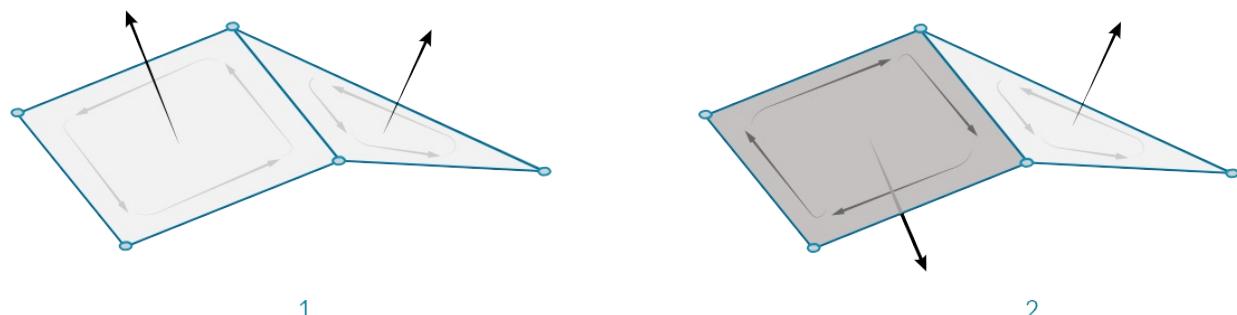


Топологически эквивалентные кружка и пончик

1.6.2.2 Характеристики Mesh

Ориентируемый

Mesh считается *ориентируемой* если имеются хорошо определяемые стороны mesh. Простой пример неориентируемой mesh, когда смежные полигоны имеют нормали, указывающие в противоположных направлениях. Такие "перевернутые полигоны" могут вызвать проблемы в визуализации и рендеринге, а также при производстве или 3D-печати.



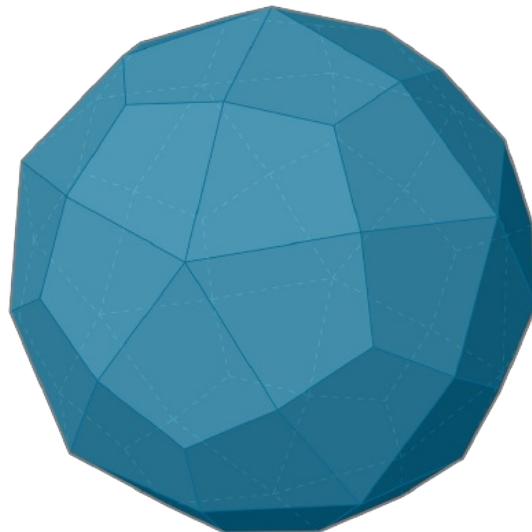
1. Ориентируемая поверхность с нормалями полигона, указывающими в одном направлении.
2. Неориентируемая поверхность имеет смежные нормали, указывающие в разных направлениях.

Открытая - Закрытая

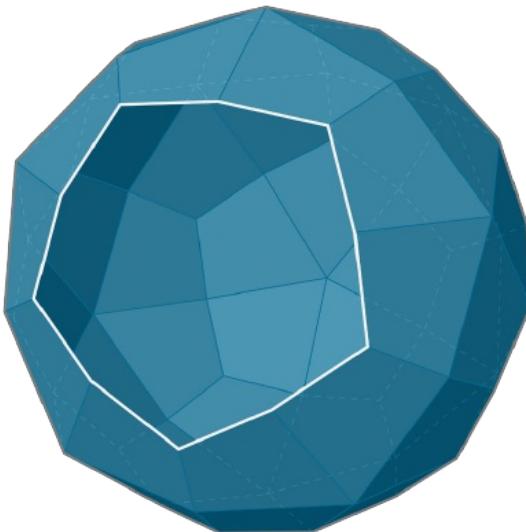
Часто необходимо знать является ли mesh *закрытой* объемное твердое тело, или открытой, 2-х пространственная поверхность. Разница может быть обязательной, особенно для возможностей производства. Вы не можете сделать 3D печать одиночной поверхности, у которой нет толщины, но вместо этого утолщает mesh, так что она становится твердым телом. Mesh геометрия твердого тела также требуется для операций с булевыми значениями (подробно в следующем разделе).

Компонент **Mesh Edges** может использоваться для определения этого фактора. Если ни одно из ребер mesh не имеет валентность 1 (если выход E1 - *отсутствующее значение*), тогда мы знаем, что все ребра являются *Interior* ребрами и у mesh нет внешнего граничного ребра и, таким образом, mesh закрытая.

С другой стороны, если имеется *Naked* ребра, тогда такие ребра должны располагаться на границе mesh и тогда mesh не закрытая.



1

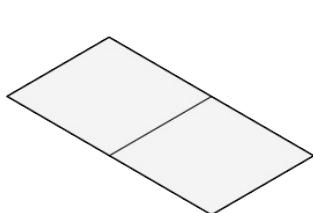


2

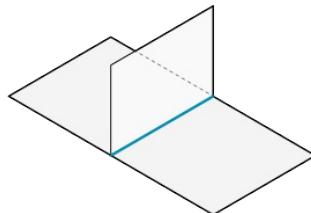
1. Закрытая mesh. Все ребра являются смежными с двумя полигонами.
2. Открытая mesh. Белые ребра являются смежными только для одного полигона.

Однородная - Неоднородная

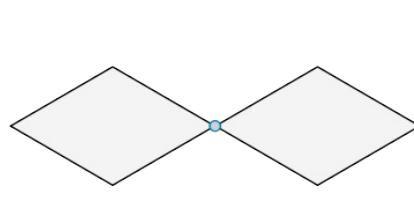
Неоднородная геометрия, не что иное как геометрия, которая не может существовать в "реальном мире". Это неизбежно делает ее "плохой геометрией", но об этом нужно знать из-за сложностей, которые она может представлять для инструментов и операций (например, рендер с эффектом преломления, симуляция флюидных потоков, булевые операции, 3d печать, и т.д.). Общие условия, которые приводят к неоднородной mesh, включают: самопересечение, naked ребра (из отверстий или внутренних полигонов), не объединенная топология и совпадающие/дублирующиеся полигоны. Mesh может также считаться *неоднородной* если она включает любые вершины, которые находятся на полигонах, которые не имеют общих ребер или имеют любые ребра с валентностью больше 2-х, создавая узел из, по крайней мере, 3-х полигонов



1



2



3

1. Простая однородная mesh
2. Три полигона встречаются на одном ребре - неоднородная mesh, также известная как T-узел
3. Два полигона встречаются только на одной вершине, но не имеют общего ребра - это неоднородная mesh

1.6.2.3 Meshes и NURBS

Как геометрия mesh отличается от геометрии NURBS? В каких случаях вы захотите использовать одну вместо другой?

Параметризация

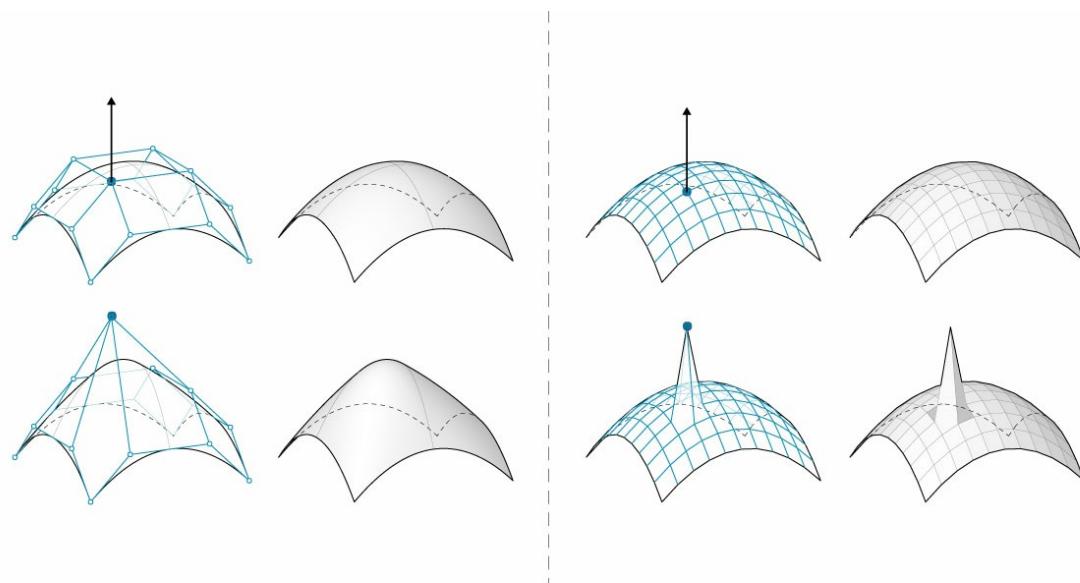
В предыдущей главе, мы видели, что поверхности NURBS определяются как последовательность кривых NURBS, направленных в двух направлениях. Эти направления отмечаются как U и V и позволяют параметризовать NURBS поверхность в соответствии с двух-пространственным диапазоном поверхности. Кривые, сами по себе, хранятся как уравнения в компьютере, позволяя рассчитать итоговую поверхность до какой угодно малой степени точности. Тем не менее, может быть сложно комбинировать множество NURBS поверхностей вместе. Объединяя две NURBS поверхности, мы получим полиповерхность, где различные секции геометрии будут иметь различные UV параметры и файлы кривых.

Mesh, с другой стороны, состоят из дискретного числа точно определенных вершин и полигонов. Связь вершин, обычно, не может быть определена простыми UV координатами и из-за того, что полигоны дискретны, степень точности встроена в mesh и может быть изменена только исправлением mesh и добавлением больше полигонов. Отсутствие UV координат, тем не менее, дает mesh гибкость, чтобы справиться с более сложной геометрией при помощи одной mesh, вместо использования полиповерхности, как при работе с NURBS.

Примечание - хотя у mesh нет скрытой UV параметризации, иногда полезно определить такую параметризацию, чтобы перенести текстуру или файл изображения на геометрию mesh для рендера. Некоторое моделирующее ПО, в таких случаях, принимает UV координаты вершины mesh за характеристику (как цвет вершины), которой можно управлять и изменять. Они, обычно, определены и не полностью определены самой mesh.

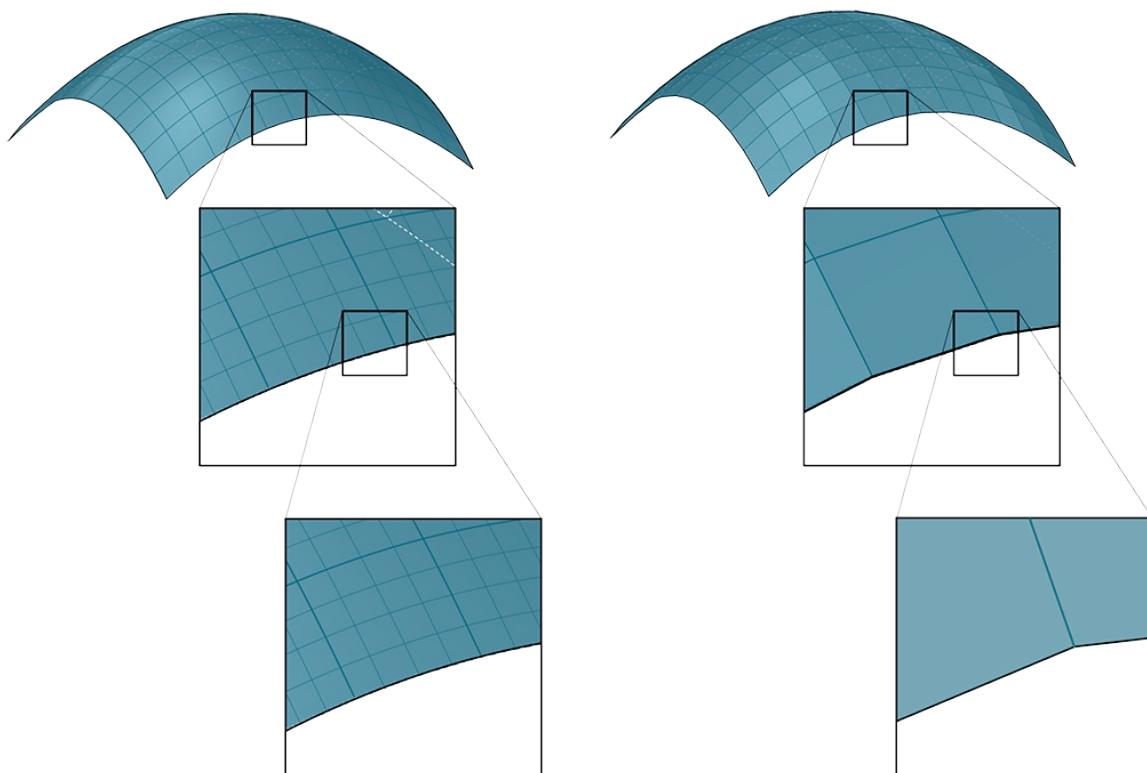
Локальное и Глобальное влияние

Еще одна важная разница - степень того, как локальное изменение в mesh или NURBS геометрии влияет на всю форму. Mesh геометрия полностью локальна. Перемещая одну вершину, мы повлияем только на тот полигон, который смежен с этой вершиной. В NURBS поверхностях степень влияния более сложная и зависит от степени поверхности, также как и веса и узлов контрольных точек. В общем, тем не менее, перемещение одной контрольной точки в NURBS поверхности создает более глобальные изменения в геометрии.



1. NURBS поверхность - перемещение контрольной точки имеет глобальное влияние
2. Геометрия Mesh - перемещение вершины имеет локальное влияние

Полезной может быть одна аналогия - сравнить векторное изображения (состоящее из линий и кривых) с растровым изображением (состоящим из индивидуальных пикселей). Если вы приблизите векторное изображение, кривые останутся закрученными и чистыми, в то время как приближение растрового изображения показывает отдельные пиксели. В этом примере, NURBS поверхности можно сравнить с векторным изображением, а mesh ведет себя как растровое изображение.



Приближение NURBS поверхности оставляет гладкую кривую, в то время как элементы mesh имеет фиксированное разрешение

Интересно отметить, что хоть NURBS поверхности хранятся как математические уравнения, их фактическая визуализация этих поверхностей требует использование mesh. Для компьютера это является невозможным отобразить непрерывное уравнение. Вместо этого, он должен разбить это уравнение на небольшие части, в результате чего, все процессы рендера или отображения должны конвертировать NURBS в mesh. В качестве аналога, представьте, что даже несмотря на то, что мы можем хранить уравнение в виде строчки на компьютере, чтобы отобразить эту линию, компьютер должен в какой-то момент конвертировать линию в последовательность дискретных пикселей на экране для отображения.

1.6.2.4 За и Против Mesh

Когда мы задаем вопрос "Какие есть за и против моделирования с помощью mesh?", мы действительно спрашиваем "Какие есть за и против моделирования с помощью формы, которые определяются только набором вершин и соответствующей топологической структуры?" Посредством этого метода выработки вопроса, становится легче увидеть, что "упрощенная" природа mesh является критичным аспектом, который сделает mesh либо подходящей либо неподходящей для моделирования в зависимости от контекста ее применения.

Mesh может подходить в ситуациях, где:

- Должен быть динамически обновляемый рендер формы, который меняется в самой форме, а не в связности полигона
- Было бы достаточно дискретной аппроксимации кривой геометрии
- Геометрия низкого разрешения должна быть систематически слажена (или артикулирована) с использованием вычислительных методов для получения модели высокого разрешения.
- Модель низкого разрешения должна одновременно поддерживать детали локальные, высокого разрешения

Mesh могут быть неподходящими в ситуациях, где:

- Кривизна и слаженность должны быть представлены на высоком уровне точности
- True derivatives должны быть определены
- Геометрия должна быть конвертирована в изготавливаемое твердое тело
- Финальная форма должна быть легко изменяемой вручную

1.6.3 Создание Mesh

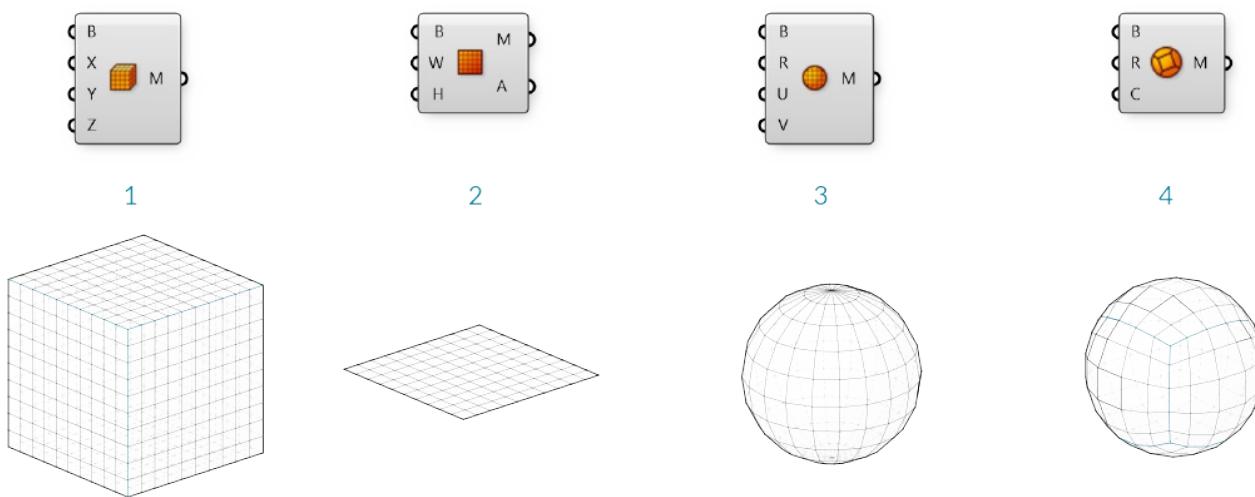
В последнем разделе мы рассмотрели основные структуры mesh. В этом разделе мы представим краткое введение в различные способы создания геометрии mesh.

Существует три основных способа создания геометрии mesh в Grasshopper:

1. Начать с mesh примитивы
2. Вручную создать mesh из полигонов и вершин
3. Конвертировать NURBS геометрию в mesh

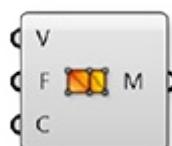
1.6.3.1 Примитивы

В Grasshopper есть несколько простых компонентов mesh примитивы:



1. **Mesh Box** - этот примитив нуждается в объекте Box (коробка) как вход, который предоставляет размер и положение, а также значения X, Y и Z, которые определяют на сколько полигонов разделить коробку. Шесть сторон Mesh Box **неспаенные**, что создает возможности для складок. (В следующем разделе мы расскажем подробнее о спаенных mesh)
2. **Mesh Plane** - этот примитив нуждается в входе Прямоугольник для определения размера и положения плоскости, а также значений W и H для определения числа полигонов.
3. **Mesh Sphere** - этот примитив нуждается в базовой плоскости для определения центра и ориентации сферы, радиус для размера и значения U и V для определения числа полигонов.
4. **Mesh Sphere Ex** - также известный как Квадрошар, этот примитив создает сферу, состоящую из шести участков, которые разделены в соответствии с входом С. Квадрошар - это топологический эквивалент куба, хотя он и геометрически сферический.

1.6.3.2 Создание Mesh



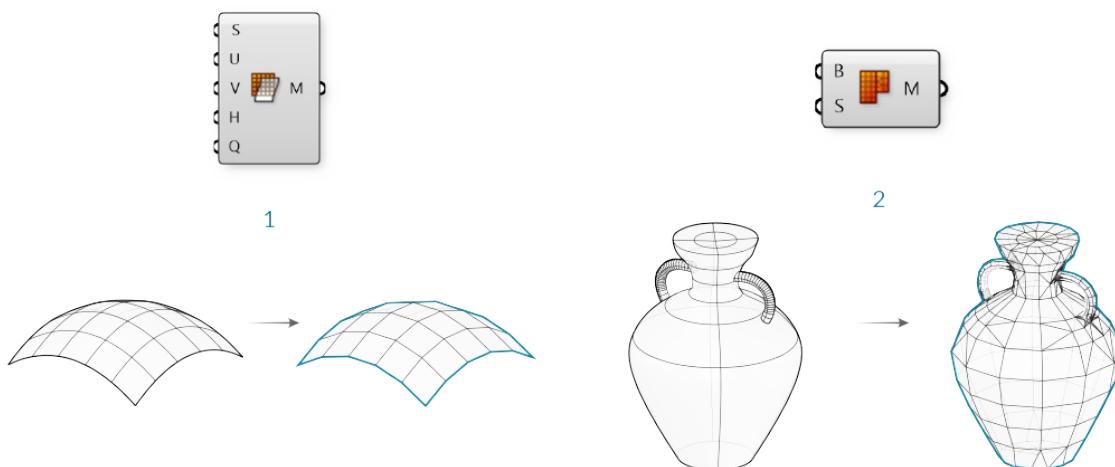
Как мы видели в предыдущем разделе, компонент **Construct Mesh** может быть использован, чтобы непосредственно создать mesh из списка вершин и из списка полигонов (и из дополнительного списка цветов вершин). Создание целой mesh вручную может быть невероятно утомительным, поэтому этот

компонент чаще используется с существующим списком полигонов и вершин, которые были извлечены используя компонент **Deconstruct Mesh** на существующую mesh.

1.6.3.3 NURBS в Mesh

Возможно, самый распространенный способ создания сложной mesh - ее генерирование на основе геометрии NURBS. Индивидуальные поверхности NURBS могут быть конвертированы в mesh используя компонент **Mesh Surface**, он просто разделяет поверхность вдоль ее UV координат и создает четырехугольные полигоны. Этот компонент позволяет вводить число U и V подразделений для получаемой mesh.

Более комплексные полиповерхности могут быть конвертированы в одиночную mesh с помощью компонента **Mesh Brep**. Этот компонент также имеет опциональный вход **Settings** (настройки), который может быть настроен используя один из встроенных компонентов **Settings - Speed** (скорость), **Quality** (качество), или **Custom** (пользовательский) также можно кликнуть правой клавишей мыши по входу **S** и выбрать "Set Mesh Options". Для эффективного использования mesh, часто необходимо исправить эту mesh используя различные стратегии, такие как перестроение, сглаживание или подразделение. Некоторые из этих техник будут обсуждаться дальше в этом Пособии.



1. **Mesh Surface** конвертирует NURBS поверхность в mesh
2. **Mesh Brep** может конвертировать полиповерхности и более сложные геометрии в одиночные mesh. Регулируя настройки можно создать больше или меньше полигонов, низкое или высокое разрешение mesh.

ПРИМЕЧАНИЕ: в целом, гораздо легче конвертировать NURBS геометрию в объект mesh, чем наоборот. И если UV координаты NURBS поверхности - несложные для конвертации в четырехугольные полигоны mesh, то обратное не всегда верно, так как mesh может содержать комбинацию треугольников и четырехугольников таким образом, что извлечь систему координат UV не просто.

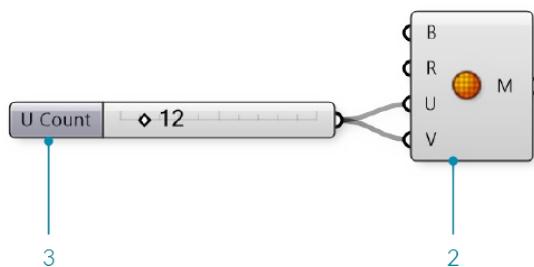
1.6.3.4 Упражнение

В этом упражнении мы используем основные Mesh примитивы, выполним изменение вершин и, затем, присвоим цвета, основываясь на нормали векторов для аппроксимации процесса рендера.

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

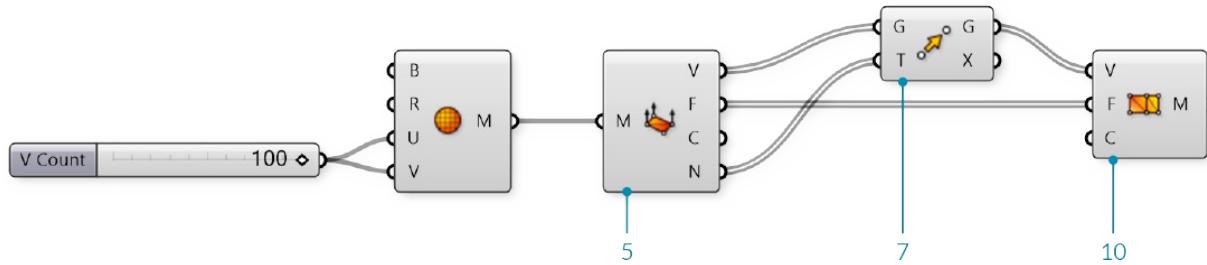
01.	Запустите новое определение, набрав Ctrl+N (в Grasshopper)
-----	--

02.	Зайдите в Mesh/Primitive/Mesh Sphere - перетащите компонент Mesh Sphere на холст	
03.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Rounding: Integer Lower Limit: 0 Upper Limit: 100 Value: 10	
04.	Подключите слайдер Number Slider к входам U Count (U) и V Count (V) компонента Mesh Sphere	



Переместите слайдер и посмотрите как изменится разрешение сферы в видовом окне Rhino. Высокие значения ведут к более сглаженной сфере, но также производят большие массивы данных, которые могут требовать большее время обработки.

05.	Зайдите в Mesh/Analysis/Deconstruct Mesh - перетащите компонент Deconstruct Mesh на холст	
06.	Соедините выход Mesh (M) компонента Mesh Sphere с входом Mesh (M) компонента Deconstruct Mesh	
07.	Зайдите в Transform/Euclidean/Move - перетащите компонент Move на холст	
08.	Соедините выход Vertices (V) компонента Deconstruct Mesh с входом Geometry (G) компонента Move	
09.	Соедините выход Normals (N) компонента Deconstruct Mesh с входом Motion (T) компонента Move	
10.	Зайдите в Mesh/Analysis/Construct Mesh - перетащите компонент Construct Mesh на холст	
11.	Соедините выход Geometry (G) компонента Move с входом Vertices (V) компонента Construct Mesh	
12.	Соедините выход Faces (F) компонента Deconstruct Mesh с входом Faces (F) компонента Construct Mesh	

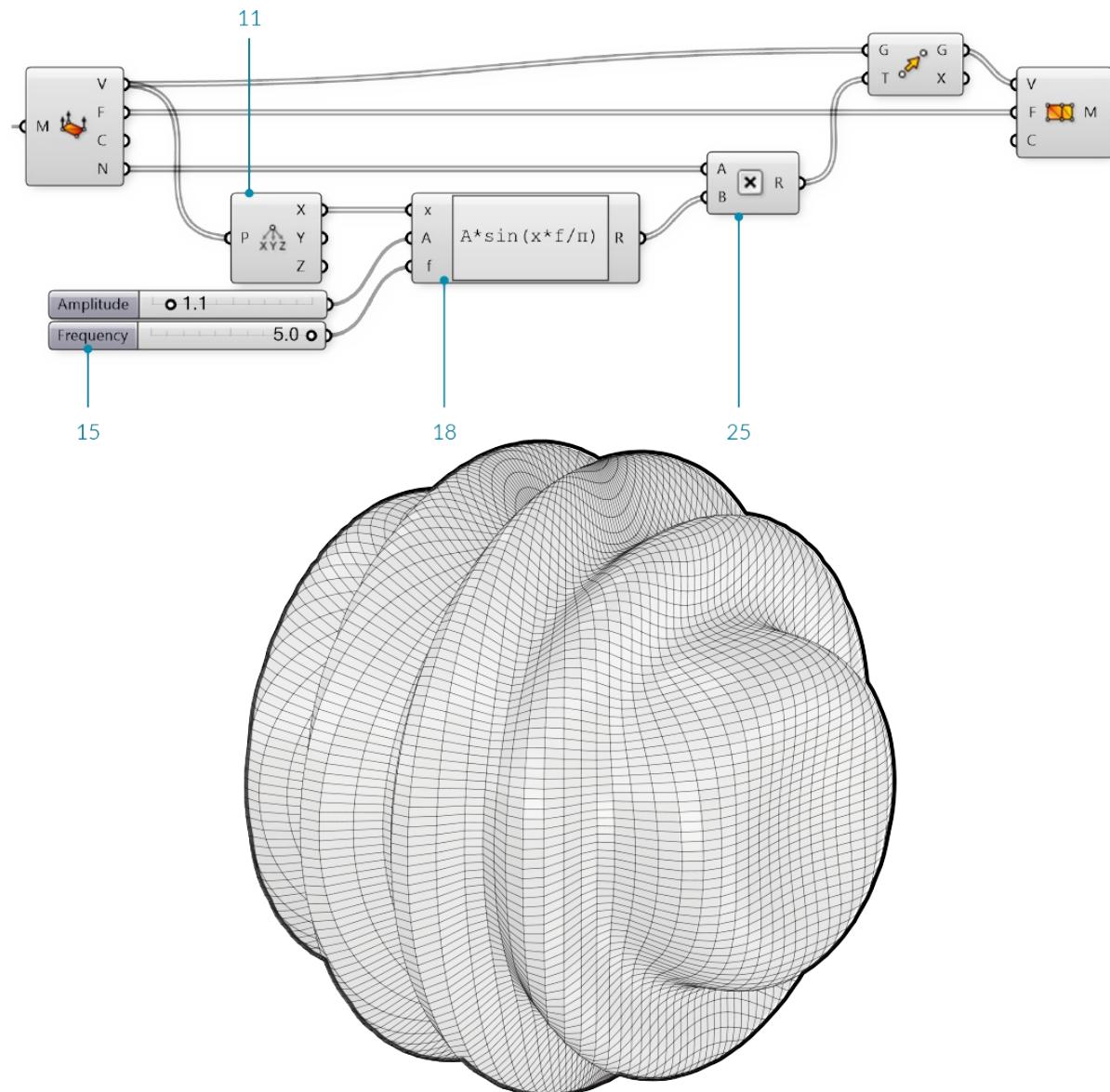


Мы разобрали mesh на вершины, полигоны и нормали. Затем мы просто передвигаем каждую вершину в соответствии с ее нормаль вектором. Из-за того, что мы совсем не изменили топологию сферы, мы заново использовали список полигонов, чтобы реконструировать новую mesh. Нормаль векторы всегда имеют длину один, поэтому это привело к реконструированию новой сферы mesh с радиусом больше на один больше, чем исходная сфера.

Далее, мы будем использовать функцию синуса для работы со сферой более сложным способом.

13.	Зайдите в Vector/Point/Deconstruct - перетащите компонент Deconstruct на холст	
14.	Соедините выход Vertices (V) компонента Deconstruct Mesh с входом Point (P) компонента Deconstruct	
15.	Зайдите в Params/Input/Number Slider - вытащите два слайдера Number Slider на холст	
16.	Установите значения на первом слайдере Number Slider : Name: Amplitude (амплитуда) Rounding: Float Lower Limit: 0 Upper Limit: 10	
17.	Установите значения на втором слайдере Number Slider : Name: Frequency Rounding: Float Lower Limit: 0 Upper Limit: 5	
18.	Зайдите в Maths/Script/Expression - перетащите компонент Expression на холст	
19.	Приблизьте компонент Expression , пока не увидите опцию для добавления или удаления вводных переменных и кликните на '+', чтобы добавить переменную 'Z'	
20.	Кликните правой клавишей мыши по входу 'Y' компонента Expression и измените текст на 'A'	
21.	Кликните правой клавишей мыши по входу 'Z' компонента Expression и измените текст на 'f'	
22.	Дважды кликните по компоненту Expression для редактирования выражения и введите следующее: $A * \sin(X * f / \pi)$	
23.	Соедините выход X компонента Deconstruct с входом 'X' компонента Expression	
24.	Соедините слайдер Amplitude Number Slider с входом A и слайдер Frequency Number Slider с входом f компонента Expression	
25.	Зайдите в Maths/Operators/Multiplication - перетащите компонент Multiplication на холст	

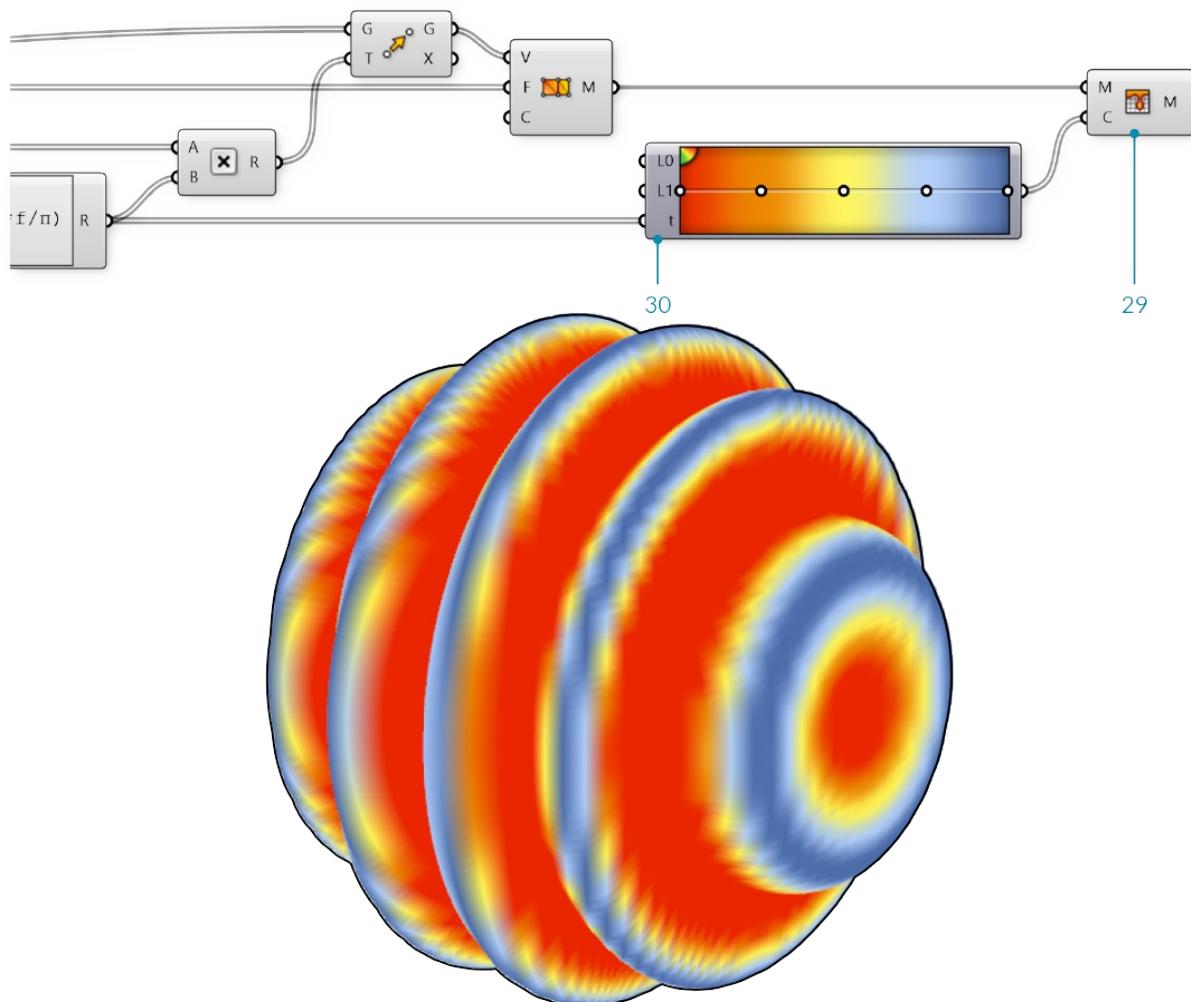
26.	Соедините выход Normals (N) компонента Deconstruct Mesh с входом A компонента Multiplication
27.	Соедините выход Result (R) компонента Expression с входом B компонента Multiplication
28.	Соедините выход Result (R) компонента Multiplication с входом Motion (T) компонента Move



Настройте слайдеры Amplitude и Frequency, чтобы посмотреть, как вновь созданные mesh изменятся.

29.	Зайдите в Mesh/Primitive/Mesh Colours - перетащите компонент Mesh Colours на холст
30.	Зайдите в Params/Input/Gradient - вытащите компонент Gradient на холст Вы можете кликнуть правой клавишей мыши по компоненту Gradient и выберите "Presets", чтобы изменить цвет градиента. В этом примере мы использовали Красно-Желтый-Синий градиент
31.	Соедините выход Result (R) компонента Expression с входом Parameter (t) компонента Gradient

32.	Соедините выход компонента Gradient с входом Colours (C) компонента Mesh Colours	
33.	Соедините выход Mesh (M) компонента Construct Mesh с входом Mesh (M) компонента Mesh Colours	
33.	В этом шаге, мы могли достичнуть такого же результата, если бы соединили Gradient прямо с входом Colours (C) компонента *Construct Mesh*	

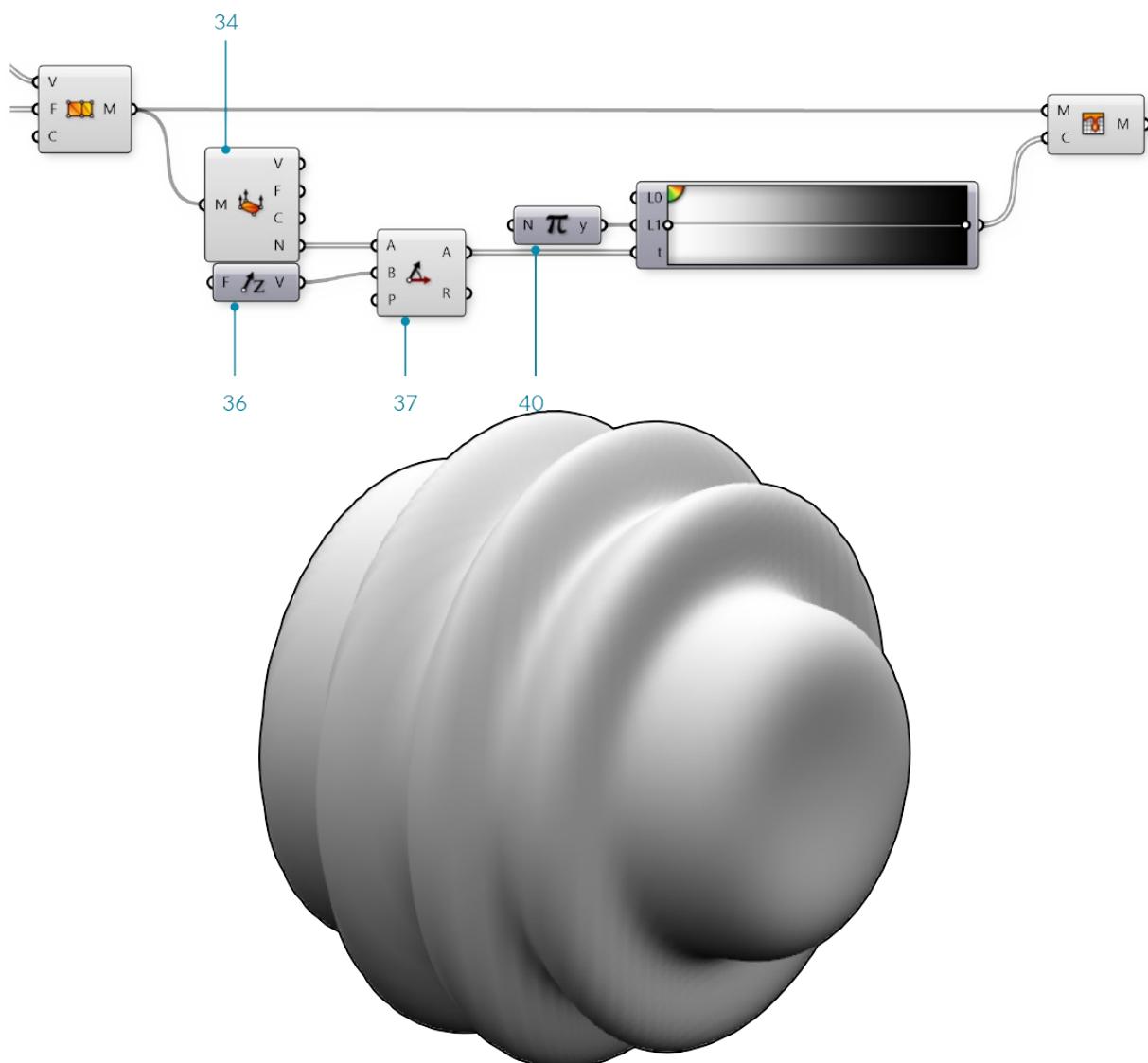


Мы использовали результаты Expression, чтобы перемещать движения и вершин и цвет mesh, так что цветовой градиент, в этом случае, соответствует амплитуде движения вершин.

В конце этого упражнения, мы будем использовать направление нормалей относительно вектора источника света, чтобы симулировать основной процесс рендеринга mesh.

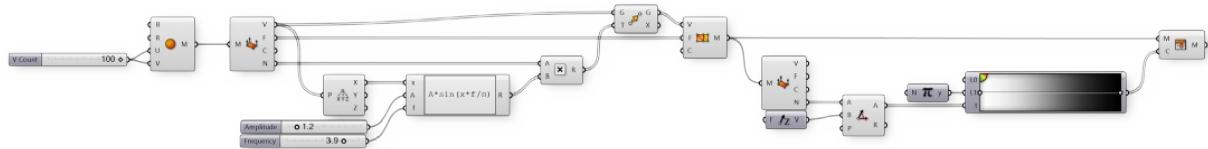
34.	Mesh/Analysis/Deconstruct Mesh - перетащите компонент Deconstruct Mesh на холст	
35.	Соедините выход Mesh (M) компонента Construct Mesh с входом Mesh (M) компонента Deconstruct Mesh	
35.	В то время как топология исходной mesh изменилась, нормаль векторы будут отличаться, поэтому нам необходимо использовать новый компонент Deconstruct Mesh , чтобы найти новые нормали.	
	Зайдите в Vector/Vector/Unit Z - перетащите компонент Unit X на холст	

36.	Мы будем использовать это как направление источника света. Вы можете использовать другие векторы, или ссылааться на линию из Rhino, чтобы сделать ее более динамичной	
37.	Зайдите в Vector/Vector/Angle - перетащите компонент Angle на холст	
38.	Соедините выход Normals (N) компонента Deconstruct Mesh с входом А компонента Angle	
39.	Соедините выход компонента Unit Z с входом В компонента Angle	
40.	Зайдите в Maths/Util/Pi - перетащите компонент Pi на холст	
41.	Соедините компонент Pi с входом Upper Limit (L1) компонента Gradient	
42.	Соедините выход Angle (A) компонента Angle с входом Parameter (t) компонента Gradient	



Мы использовали черно-белую настройку для этого градиента. Это устанавливает цвет mesh в соответствии с углом между нормалью и источником света, с нормальми, которые направлены прямо на источник черного света, и с нормальми, которые направлены от источника белого цвета (чтобы быть более точными, вы можете развернуть градиент, настраивая ползунки). Реальный процесс рендеринга mesh намного более сложный, чем этот, конечно же, но это базовый процесс создания света

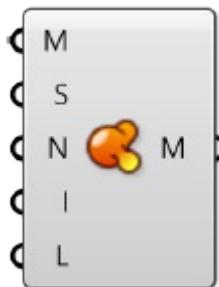
и тени на отрендеренном объекте.



1.6.4 Операции с Mesh

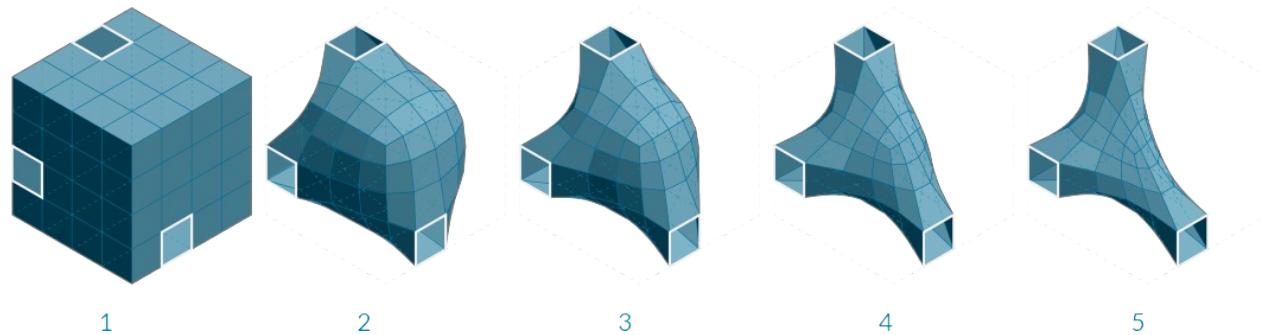
В последнем разделе мы рассмотрели основные структуры mesh. В этом разделе мы рассмотрим способы работы с геометрией mesh.

1.6.4.1 Smooth



Сглаженные mesh иногда могут быть получены простым увеличением числа полигонов в процессе, который называется *subdivision* (дробление). Это часто может вести к невероятно огромным массивам данных, вычисление которых занимает длительное время. Также Grasshopper требуется установить дополнительные не-встроенные аддоны для этого. В этих ситуациях компонент **Smooth** может использоваться как альтернатива, чтобы сделать mesh менее шероховатой или граненой без увеличения числа вершин и полигонов или изменения топологии. *Прочность*, *число итераций*, и *ограничение смещения* можно использовать для настройки того, как происходит сглаживание.

При прикреплении булевого значения к входу N предоставляет возможность пропустить naked вершины. Вершина является naked, если она подключена к naked ребру, что означает, что вершина находится на границе открытой mesh. Переключая эту опцию вы поддерживаете внешнюю границу mesh в то время как вы гладите внешние ребра.

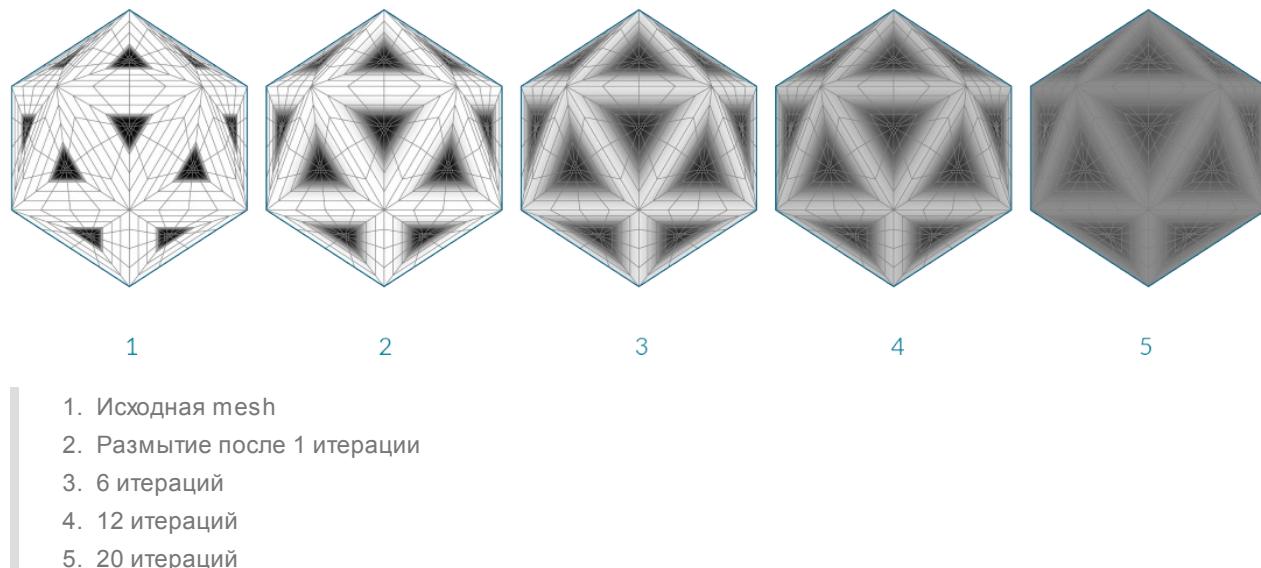


1. Исходная форма mesh, 3 полигона удалены
2. Сглаживание после 2 итераций
3. 6 итераций
4. 25 итераций
5. 50 итераций

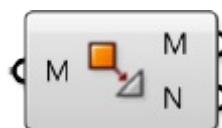
1.6.4.2 Blur



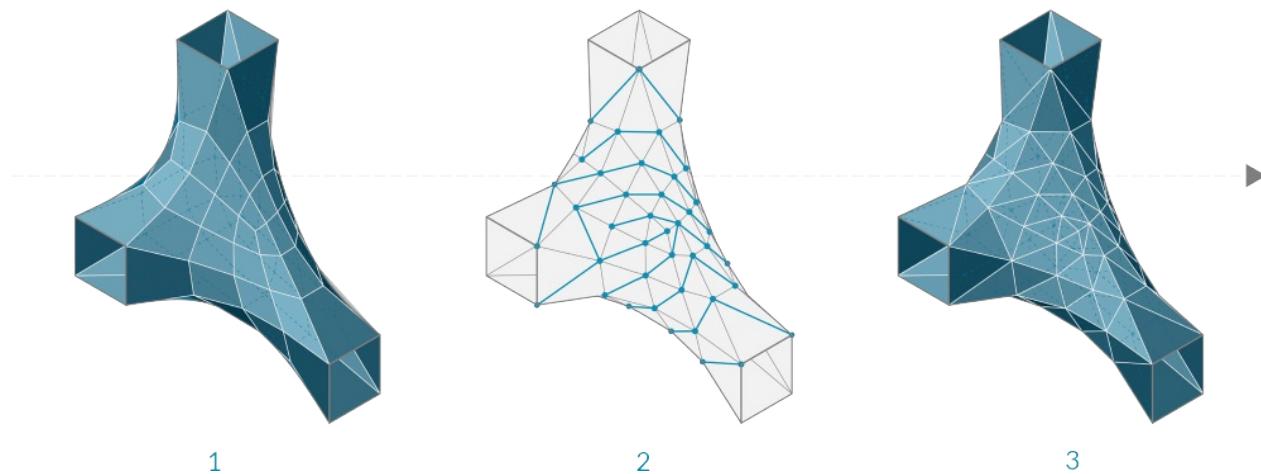
Компонент **Blur** работает почти так же как компонент **smooth**, за исключением того, что он воздействует только на цвета вершин. Он также может использоваться для сокращения шероховатости цветной mesh, хотя и в меньшем объеме, т.к. он не меняет никакой геометрии.



1.6.4.3 Triangulate



Чтобы убедиться в том, что каждый полигон планарный или чтобы экспортить mesh в другое ПО, которое не допускает четырехугольные полигоны, иногда необходимо триангулировать mesh. При использовании компонента **Triangulate** каждый четырехугольный полигон заменяется двумя треугольными полигонами. Grasshopper всегда использует самую короткую диагональ полигона, чтобы создать новое ребро.

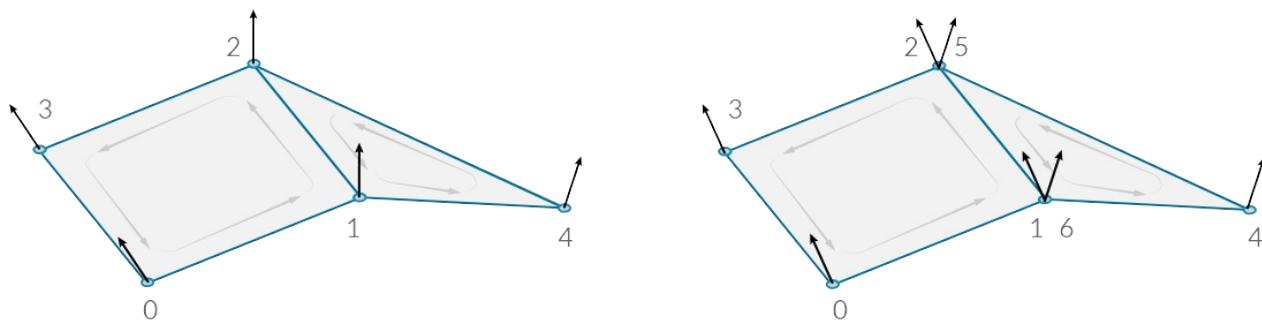


1. Исходная четырехугольная mesh
2. Добавленные ребра в соответствии с самым коротким расстоянием через четырехугольники
3.
 - i. Триангулированная итоговая mesh

1.6.4.4 Weld



В последнем разделе мы заметили, что одиночная вершина может разделяться смежными полигонами и нормаль для этой вершины рассчитывается как среднее смежных полигонов, выдавая сглаженную визуализацию. Тем не менее, иногда желательно иметь резкий сгиб или шов, где один полигон не переходил гладко в следующий путем вершины нормали. Для этой ситуации необходимо, чтобы каждый полигон имел свою собственную вершину со своей собственной нормалью. Список вершин будет содержать по меньшей мере две точки, которые имеют одинаковые координаты, но разные индексы.



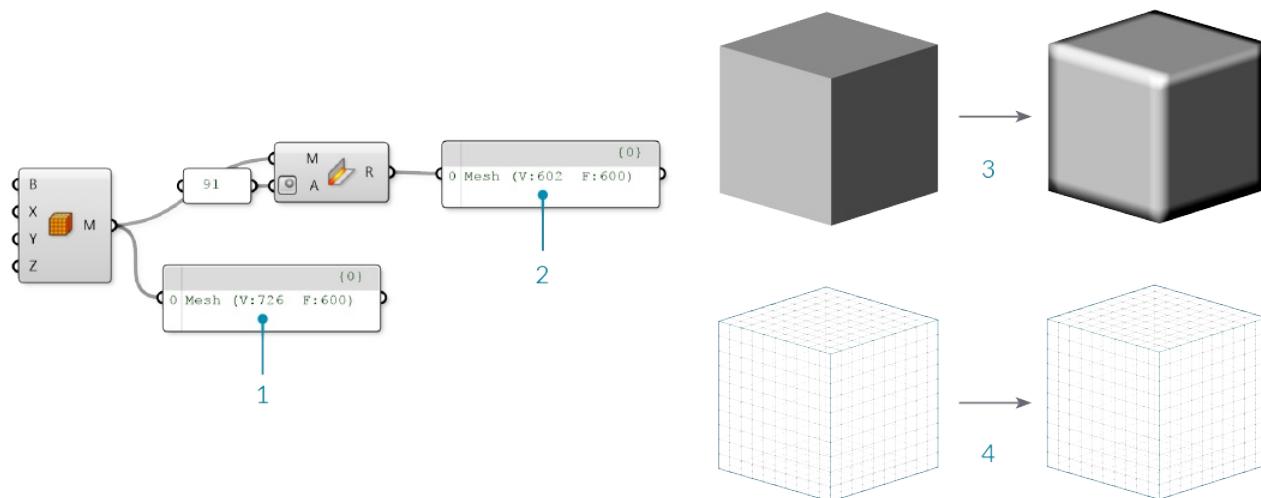
1	Vertex List	Face List
0 = {0.0, 0.0, 0.0}	Q {0, 1, 2, 3}	
1 = {1.0, 0.0, 1.0}	T {1, 4, 2}	
2 = {1.0, 1.0, 1.0}		
3 = {0.0, 1.0, 0.0}		
4 = {2.0, 0.0, -1.0}		

2	Vertex List	Face List
0 = {0.0, 0.0, 0.0}	Q {0, 1, 2, 3}	
1 = {1.0, 0.0, 1.0}	T {4, 5, 6}	
2 = {1.0, 1.0, 1.0}		
3 = {0.0, 1.0, 0.0}		
4 = {2.0, 0.0, -1.0}		
5 = {1.0, 1.0, 1.0}		
6 = {1.0, 0.0, 1.0}		

1. Сплененные полигоны - оба полигона коллективно используют вершины 1 и 2. Нормали вершин в этих вершинах - среднее вершин полигонов.
2.
 - i. Несплаенные Полигоны - Дублирующиеся вершины добавлены к списку. Полигоны не используют коллективно ни одну из вершин. Вершины 1 и 6 и вершины 2 и 5 имеют одинаковые координаты, но являются разными вершинами. Они каждая имеют свою собственную нормаль вершины

Процесс, когда берутся две вершины, находящиеся в одном и том же месте, и объединяются в одну вершину, называется **спайка**, в то время как **несплаивание** берет одну вершину и разделяет ее на множество вершин.

Компонент **Weld** использует предельные значения угла как ввод. Любые два смежных полигона с углом меньше, чем предельное значение угла будут спаиваться вместе, приводя к общим вершинам с нормалью, которая является средним смежных полигонов. Компонент **Unweld** работает противоположным образом, где смежные полигоны с углом больше, чем данное предельное значение, будут расспаиваться и их совместные вершины будут дублироваться.



1. По умолчанию Box Mesh имеет 726 вершин. Mesh сгибаются в углах коробки, где вершины сдавиваются.
2. Если mesh спаивается с углом больше, чем 90 градусов, итоговые полигоны mesh спаиваются вместе, число вершин уменьшилось до 602, в то время как число полигонов осталось таким же.
3. Если посмотреть на геометрию в предпросмотре, вы можете заметить, что отрендеренная сплайненная mesh имеет сглаженные углы.
4. В отличие от компонента Smooth, который меняет геометрию mesh, эта mesh выглядит более сглаженно, только из-за роли нормалей вершин в рендеринге и затенении. Реальное положение вершин остается без изменений.

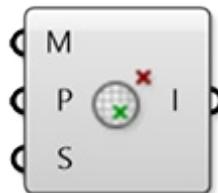
В изображении ниже, мы использовали угол 91 градус, потому что мы знаем, что стороны квадрата будут 90 градусов. Чтобы полностью спасти целую mesh, вам следует использовать угол 180 градусов.

1.6.5 Взаимодействие Mesh

Этот раздел рассматривает способы взаимодействия Объектов Mesh с другими объектами, такими как определение ближайших точек или объединение множества mesh вместе.

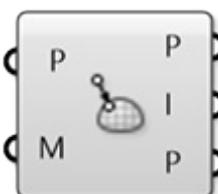
1.6.5.1 Геометрия Mesh и Точки

Inclusion

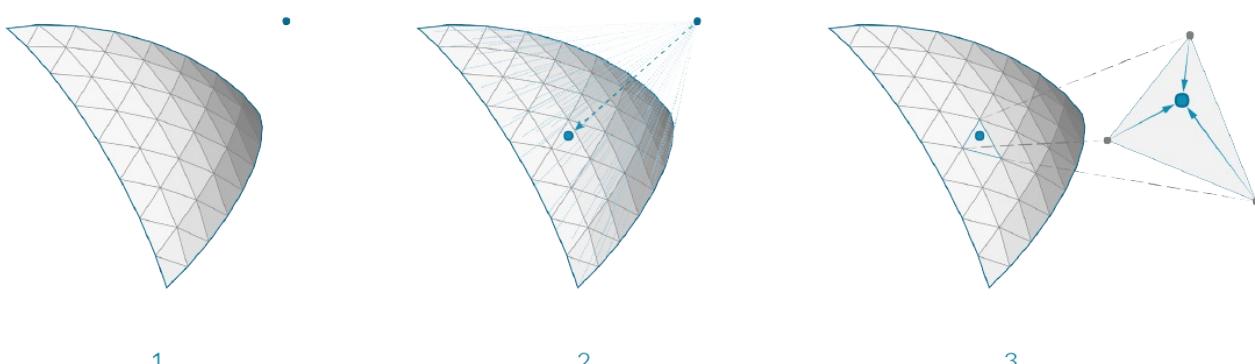


Этот компонент проверяет и определяет, находится ли данная точка внутри твердого тела mesh или нет. Это работает только с закрытыми mesh.

Mesh Closest Point



Этот компонент рассчитывает положение на mesh, ближайшее к данной точке. Этот компонент выдает три части данных: координаты рассчитанной точки на mesh, индекс полигона, которые содержит эту точку и параметр mesh. Этот параметр невероятно полезен в связке с компонентом **Mesh Eval**, который обсуждается ниже.

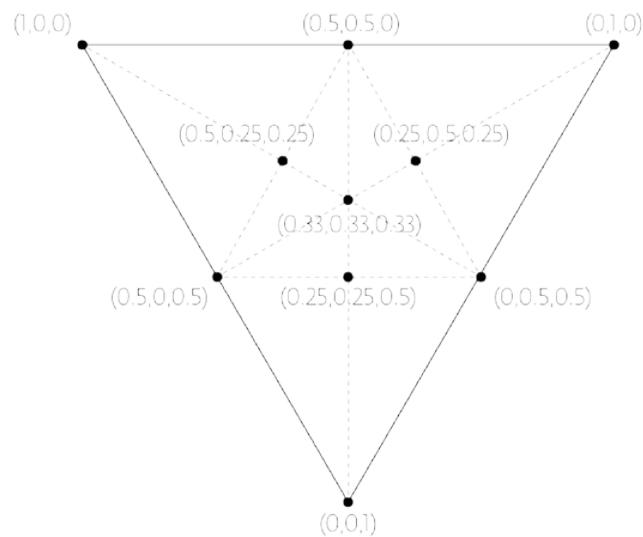


1. Данная точка в пространстве, Мы хотим найти ближайшую ближайшую точку на mesh
2. Полигон, который содержит ближайшую точку, определен
3. Рассчитаны параметры ближайшей точки на полигоне

Для тех пользователей, которые заинтересованы в больших подробностях о том, как параметризуется mesh, мы можем рассмотреть подробнее, как структурированы параметры mesh. Вы можете увидеть эту структуру путем прикрепления panel к выходу параметра компонента **Mesh Closest Point**. Параметр mesh имеет форму: N[A,B,C,D]. Первое число, N, - это индекс полигона, который содержит расчетную точку.

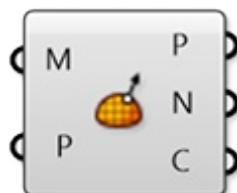
Следующие четыре числа определяют *барицентрические* координаты точки внутри этого полигона.

Координаты исходной точки можно найти путем умножения каждой вершины полигона на эти числа по порядку и затем складывания результатов вместе. (Конечно, это уже сделано за нас и дается в выходе Point). Также заметьте, что барицентрические координаты уникальны только для треугольных полигонов, что означает, что на четырехугольном полигоне та же самая точка может иметь множество различных определений значений параметров. Grasshopper избегает этой проблемы путем внутренней триангуляции четырехугольного полигона при расчете параметра, результат которого и есть одно из тех четырех чисел в параметре mesh, по меньшей мере, одно из них всегда будет ноль.



Барицентрические координаты

Mesh Eval



Компонент **Mesh Eval** использует параметр mesh как вход и выдает исходную точку, а также нормаль и цвет в этой точке. Цвет и нормаль рассчитываются как интерполяции цветов вершин и нормалей вершин, используя те же самые барицентрические координаты как параметр mesh.

1.6.5.2 Объединение Геометрии Mesh

Mesh Join



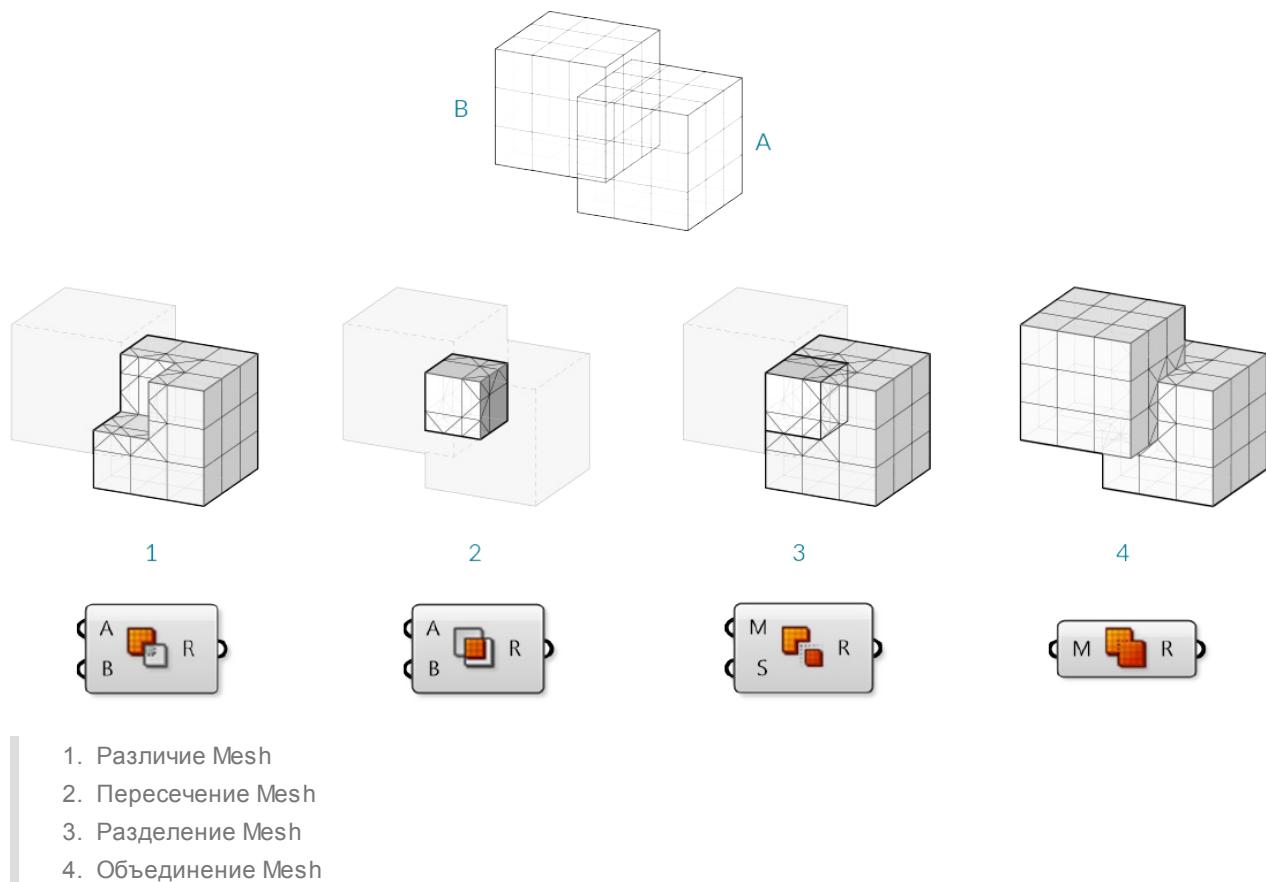
В отличие от объединения кривых или NURBS поверхностей, которые требуют смежности, любые mesh могут быть объединены в одну mesh - даже mesh, которые не соприкасаются. Вспомните, что mesh - это просто список вершин или список полигонов. Нет требования о том, что те полигоны должны быть соединены (Хотя

в большинстве приложений, такая mesh будет не желательной!!).

Этот компонент не спаивает вершины mesh вместе, поэтому часто полезно использовать его в комбинации с компонентом **Weld**.

Mesh Boolean

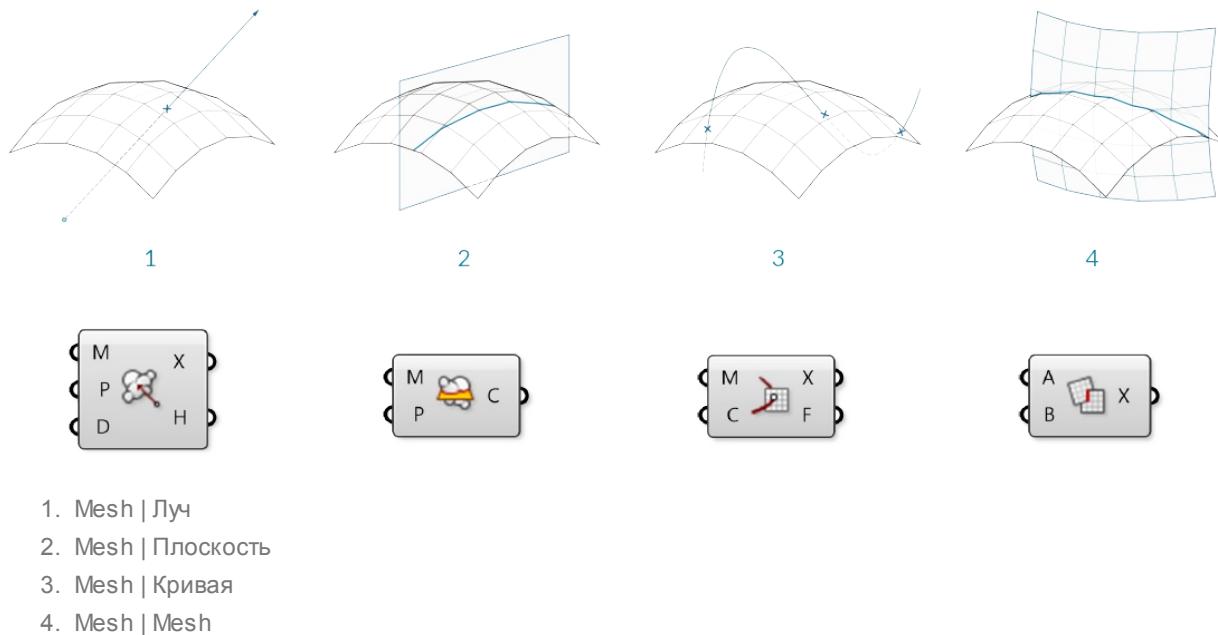
Mesh в Grasshopper имеет набор булевых операций похожих на булевые операции для твердых тел NURBS. Булевые операции зависят от порядка, это означает, что переключение порядка входа mesh A и B приведет к разным выходам.



1.6.5.3 Intersections (Пересечения) и Occlusions (Преграды)

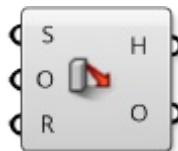
Intersect

Пересечения могут быть рассчитаны между mesh и другими объектами: лучи, плоскости, кривые и другие mesh.



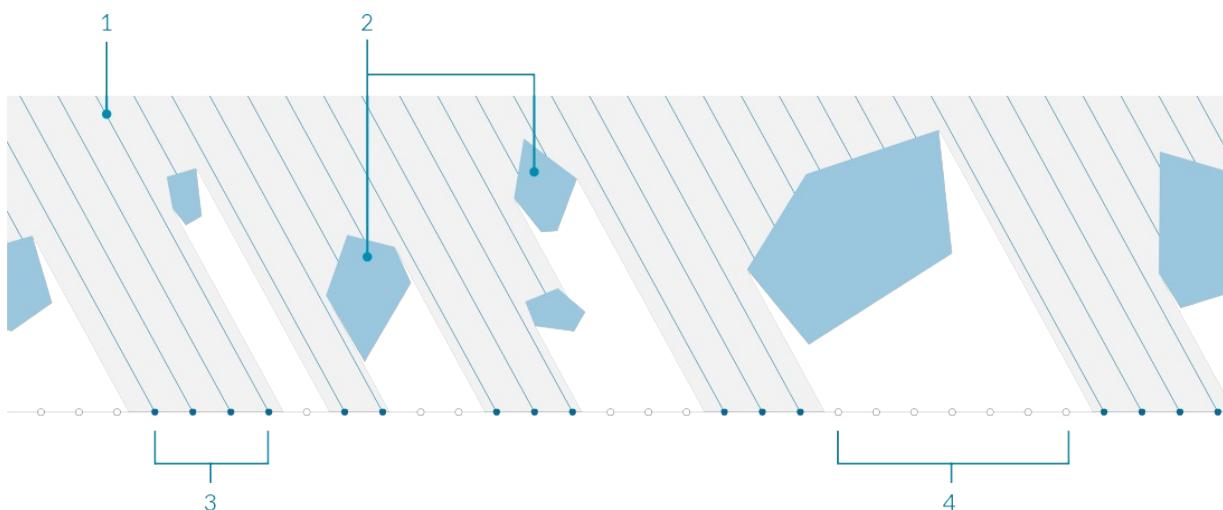
1. Mesh | Луч
2. Mesh | Плоскость
3. Mesh | Кривая
4. Mesh | Mesh

Occlusion



Как мы уже обсуждали, одно из (множества) использований mesh геометрии - для визуализации и создания теневых рендеров на основе нормалей полигона. При рендерах, также необходимо знать, когда объект в тени находится позади другого объекта. Компонент **Occlusion** в Grasshopper позволяет нам вводить точки выборки, совместно с преграждающей mesh геометрией, которая будет "отбрасывать тень", и *вид Ray*, или, вектор, чтобы отобразить направление исходящего "света".

Такой процесс может быть использован для создания теней в рендере или для того, чтобы определить спрятан ли объект от определенного вида видеокамеры.



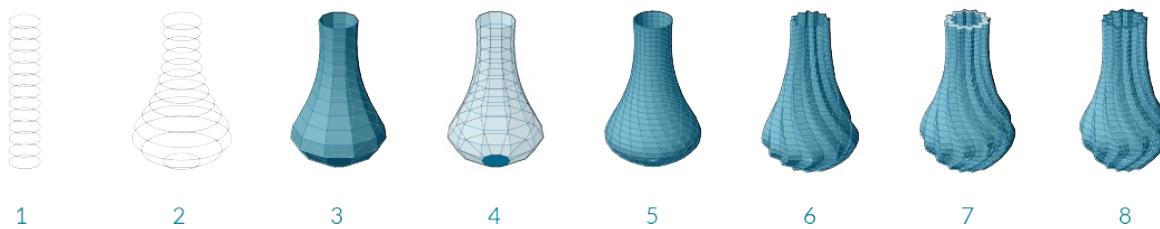
1. Вид Ray для проверки преграждения
2. Преграждающая геометрия mesh
3. 'Hit' точки выборки
4. 'Преграждающие' точки выборки

1.6.6 Работа с геометрией Mesh

В этом разделе мы будем работать над производством твердого тела mesh. К концу этого упражнения у нас будет динамическое определение для производства индивидуальных ваз, которые можно напечатать на 3D принтере.

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

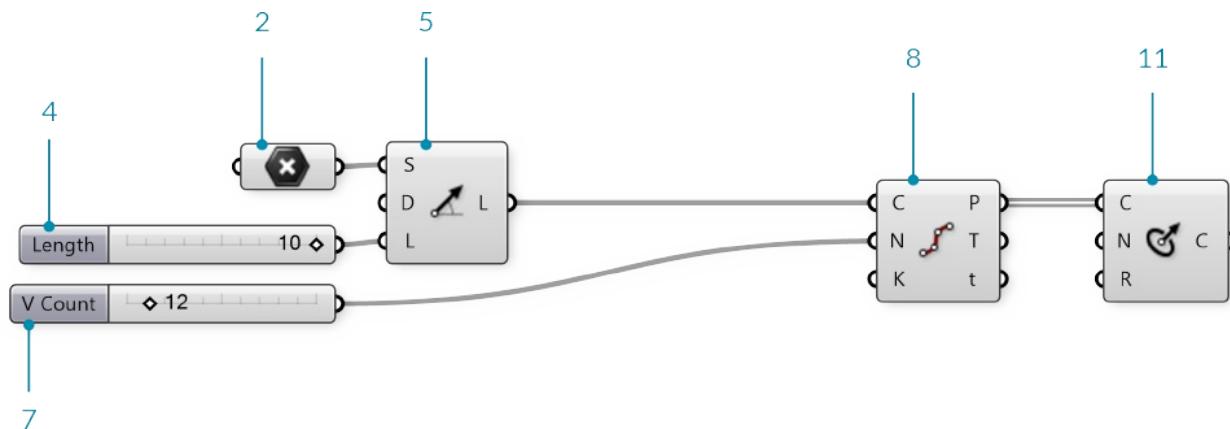
Так как это определение несколько больше, чем предыдущие упражнения в этом пособии, мы сначала разберем базовые этапы, через которые мы пройдем:



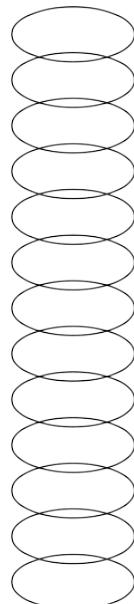
1. Создадим последовательность кругов в качестве базового цилиндра
2. Используем компонент Graph Mapper, чтобы определить контур нашей вазы
3. Создадим топологию полигонов mesh, чтобы произвести единую поверхность mesh
4. Закроем низ mesh
5. Введем закручивание по вертикали для создания более динамической формы
6. Добавим складчатые ребра для текстуры
7. Сместим поверхность mesh, чтобы придать вазе толщину
8. Закроем верхний зазор между двумя поверхностями, чтобы создать закрытое твердое тело

01.	Запустите новое определение, набрав Ctrl+N (в Grasshopper)	
02.	Зайдите в Params/Geometry/Point - вытащите контейнер Point на холст	
03.	Кликните правой клавишей мыши в Rhino по компоненту Point и выберите Set one Point, чтобы привязать точку. Она будет служить исходной точкой нашей вазы.	
04.	Вы можете создать точку вручную в Grasshopper, дважды кликнув по холсту, чтобы вызвать окно поиска, затем ввести координаты точки, разделенные запятыми, вот так: '0,0,0' (без кавычек)	
05.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Name: Length Lower Limit: 1 Upper Limit: 10	
06.	Зайдите в Curve/Primitive/Line SDL - вытащите компонент Line SDL на холст	
	Соедините компонент Point с входом Start (S) компонента Line SDL и соедините слайдер Number Slider с входом Length (L).	
	Значение Direction (D) компонента **Line SDL**, по умолчанию, Unit Z вектор, который мы будем использовать в этом упражнении.	

	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Name: V Count Rounding: Integer Lower Limit: 1 Upper Limit: 100	
07.		
08.	Зайдите в Curve/Division/Divide Curve - вытащите компонент Divide Curve на холст	
09.	Соедините выход Line (L) компонента Line SDL с входом Curve (C) компонента Divide Curve	
10.	Подключите слайдер V Count к входу Count (N) компонента Divide Curve	
11.	Зайдите в Curve/Primitive/Circle CNR - вытащите компонент Circle CNR на холст	
12.	Соедините выход Points (P) компонента Divide Curve с входом Center (C) компонента Circle CNR	

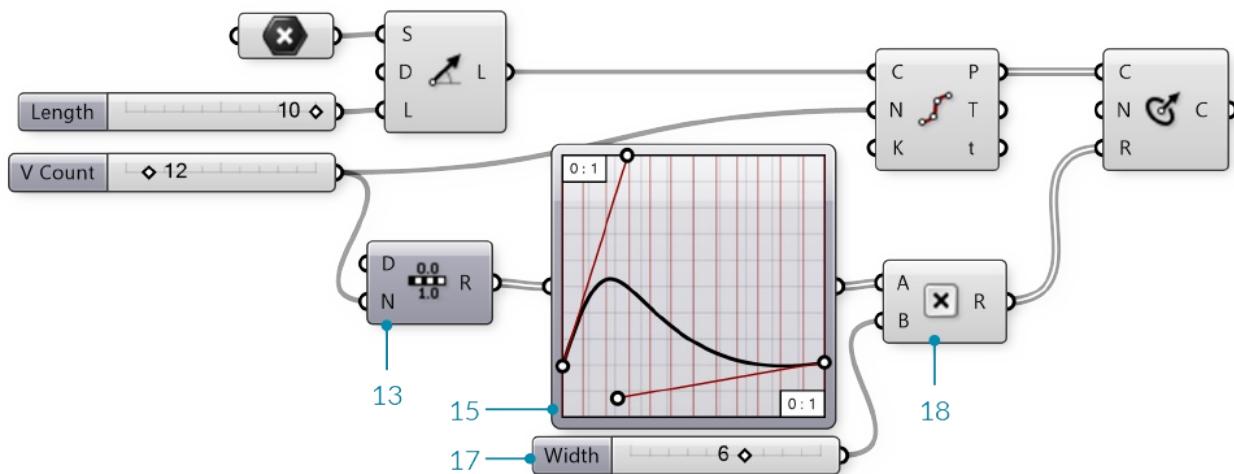


Теперь у нас есть последовательность вертикально расположенных кругов. Мы будем далее использовать их для создания контура нашей вазы.



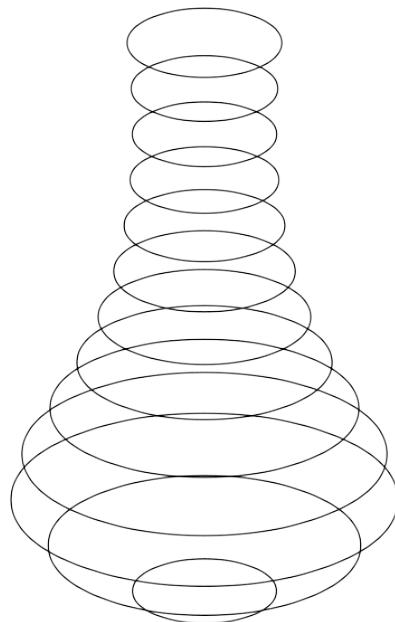
Далее, мы будем использовать Graph Mapper, чтобы контролировать радиус кругов.

13.	Зайдите в Sets/Sequence/Range - вытащите компонент Range на холст	
14.	Подключите слайдер V Count к входу Steps (N) компонента Range	
15.	Зайдите в Params/Input/Graph Mapper - вытащите компонент Graph Mapper на холст	
16.	Кликните правой клавишей мыши по Graph Mapper , кликните по 'Graph Types' в меню и выберите 'Bezier'	
17.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Name: Width Lower Limit: 0 Upper Limit: 10	
18.	Зайдите в Maths/Operators/Multiplication - перетащите компонент Multiplication на холст	
19.	Соедините Graph Mapper и слайдер Width с входами A и B компонента Multiplication	
20.	Соедините выход Result (R) компонента Multiplication с входом Radius (R) компонента Circle CNR	



Используйте ползунки на **Graph Mapper** для настройки контура кругов.

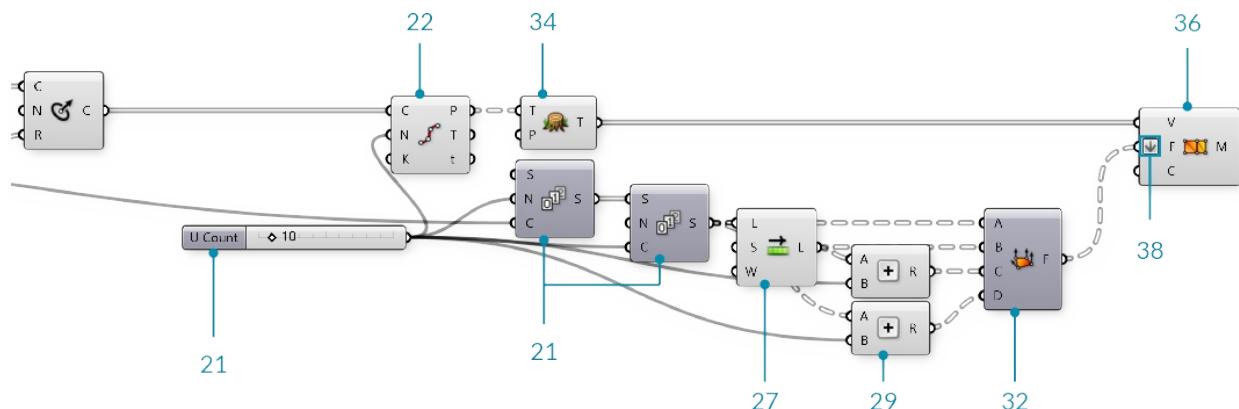
ПРИМЕЧАНИЕ: Важно убедиться, что начальная точка кривой Безье (Bezier curve) на **Graph Mapper**не равна нулю. Для этого мы поднимаем начальную точку до числа больше нуля и, таким образом, создаем плоскую основу для нашей вазы.



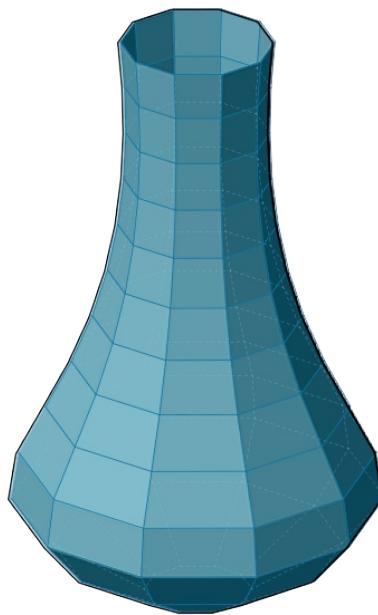
Теперь у нас есть контур для нашей вазы. Далее, мы построим поверхность mesh. Это потребует создания вершин mesh и определения полигонов mesh в соответствии с индексами тех вершин.

	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Name: U Count Rounding: Even Lower Limit: 2 Upper Limit: 100	
21.	Зайдите в Curve/Division/Divide Curve - вытащите компонент Divide Curve на холст	
22.	Соедините выход Circle (C) компонента Circle CNR с входом Curve (C) компонента Divide Curve , соедините слайдер U Count с входом Count (N)	
23.	Выход Points(P) этого компонента - вершины, которые мы будем использовать для нашей mesh	
24.	Зайдите в Sets/Sequence/Series - вытащите два компонента Series на холст	
25.	Соедините слайдер U Count с входом Step (N) первого компонента Series , и соедините слайдер V Count с входом Count (C) того же самого компонента Series	
26.	Соедините выход Series (S) первого компонента Series с входом Start (S) второго компонента Series , и соедините слайдер U Count с входом Count (C)	
27.	Зайдите в Sets/List/Shift List - вытащите компонент Shift List на холст	
28.	Соедините выход второго компонента Series с входом List (L) компонента Shift List	
29.	Зайдите в Maths/Operators/Addition - перетащите два компонента Addition на холст	
30.	Соедините выход второго компонента Series и слайдер U Count с входами A и B первого компонента Addition	
31.	Соедините выход компонента Shift List и слайдер U Count с входами A и B второго компонента Addition	

32.	Mesh/Primitive/Mesh Quad - перетащите компонент Mesh Quad на холст	
33.	Соедините следующее с входами компонента Mesh Quad : A- Второй компонент **Series** B - **Shift List** C - Первый компонент **Addition** D - Второй компонент **Addition**	
34.		
35.	Мы только что создали начальную топологию нашей mesh. Эти полигоны будут комбинироваться с вершинами. Порядок этих соединений критичен, поэтому далее дважды проверяйте все соединения в этой точке!	
36.	Зайдите в Sets/Tree/Flatten - вытащите компонент Flatten Tree на холст	
37.	Соедините выход Points (P) компонента Divide Curve с входом Tree (T) компонента Flatten Tree	
38.	Mesh/Primitive/Construct Mesh - перетащите компонент Construct Mesh на холст	
	Соедините выход Tree (T) компонента Flatten Tree с входом Vertices (V) компонента Construct Mesh	
	Соедините выход Face (F) компонента Mesh Quad с входом Faces (F) компонента Construct Mesh . Кликните правой клавишей мыши по входу F (Faces) и выберите 'Flatten'	

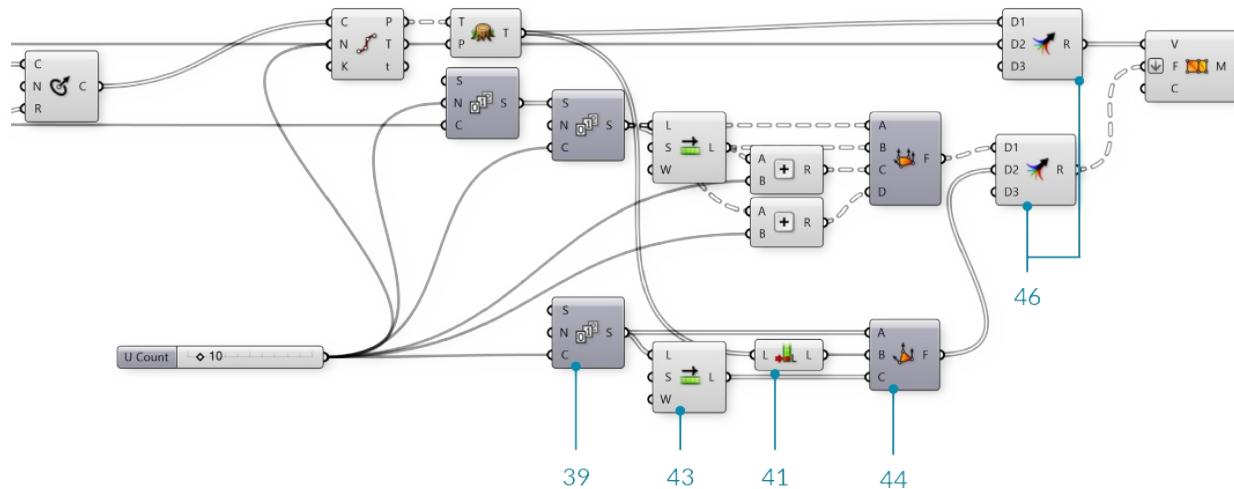


Теперь у нас есть поверхность mesh для нашей вазы.

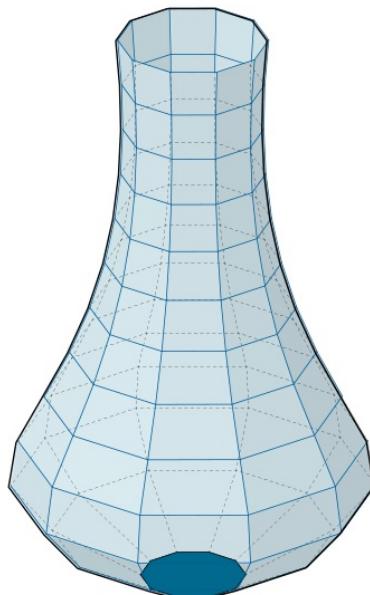


Далее мы закроем низ вазы. Чтобы сделать это, мы добавим исходную точку начала координат в наш список вершин и затем создадим треугольные полигоны mesh от нижнего ребра до этой точки.

39.	Зайдите в Sets/Sequence/Series - вытащите компонент Series на холст	
40.	Соедините слайдер U Count с входом Count (C) компонента Series	
41.	Зайдите в Sets/List/List Length - вытащите компонент List Length на холст	
42.	Соедините выход Tree (T) компонента Flatten Tree с входом List (L) компонента List Length	
42.	Это будет индекс точки начала координат после того, как мы добавим ее к существующему списку вершин.	
43.	Зайдите в Sets/List/Shift List - вытащите компонент Shift List на холст	
44.	Зайдите в Mesh/Primitive/Mesh Triangle - перетащите компонент Mesh Triangle на холст	
45.	Соедините следующее с входами компонента Mesh Triangle : А- новейший компонент **Series** Б - **List Length** С - **Shift List**	
46.	Зайдите в Sets/Tree/Merge - вытащите два компонента Merge на холст	
47.	Соедините выход Tree (T) компонента Flatten Tree с входом D1 и соедините исходный компонент Point с входом D2 первого компонента Merge	
48.	Соедините выход Faces (F) компонента Mesh Quad с входом D1 соедините выход Mesh Triangle с входом D2 второго компонента Merge	
49.	Соедините первый компонент Merge с входом Vertices (V) компонента Construct Mesh соедините второй компонент Merge с входом Faces (F) компонента Construct Mesh .	



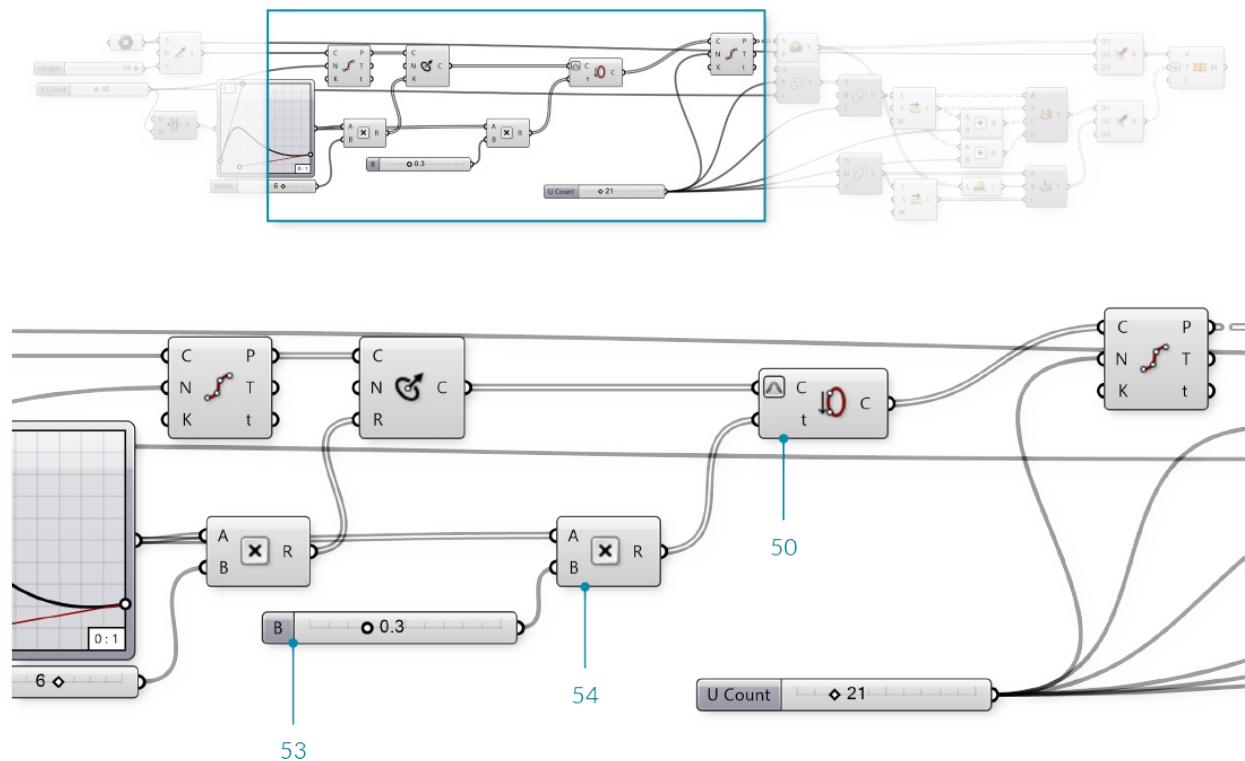
Мы закрыли низ базы треугольными полигонами mesh.



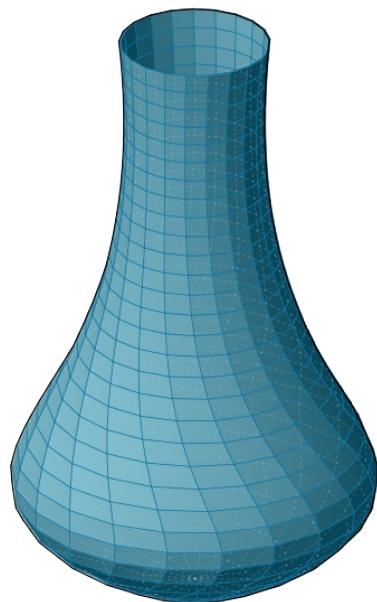
Теперь мы добавим некоторые детали. Начнем с добавления кривых в вертикальном направлении, настраивая край исходных кругов

50.	Зайдите в Curve/Util/Seam - перетащите компонент Seam на холст	
51.	Соедините выход Circle (C) Соедините выход Circle CNR с входом Curve (C) компонента Seam	
52.	Кликните правой клавишей по входу Curve (C) компонента Seam и выберите 'Reparameterize'	
53.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст. Мы будем использовать настройки по умолчанию этого слайдера	
54.	Зайдите в Maths/Operator/Multiplication - перетащите компонент Multiplication на холст.	
55.	Соедините выход из Graph Mapper с входом A, последний слайдер Number Slider с входом B компонента Multiplication	
56.	Соедините Result (R) компонента Multiplication с входом Parameter (t) компонента	

56. Seam



Кривизна достигается путем изменения положения шва начальных кругов и использует такой же Graph Mapper как контур вазы.



Далее мы добавим вертикальные ребра.

57.

Зайдите в **Sets/List/Dispatch** - перетащите компонент **Dispatch** на холст

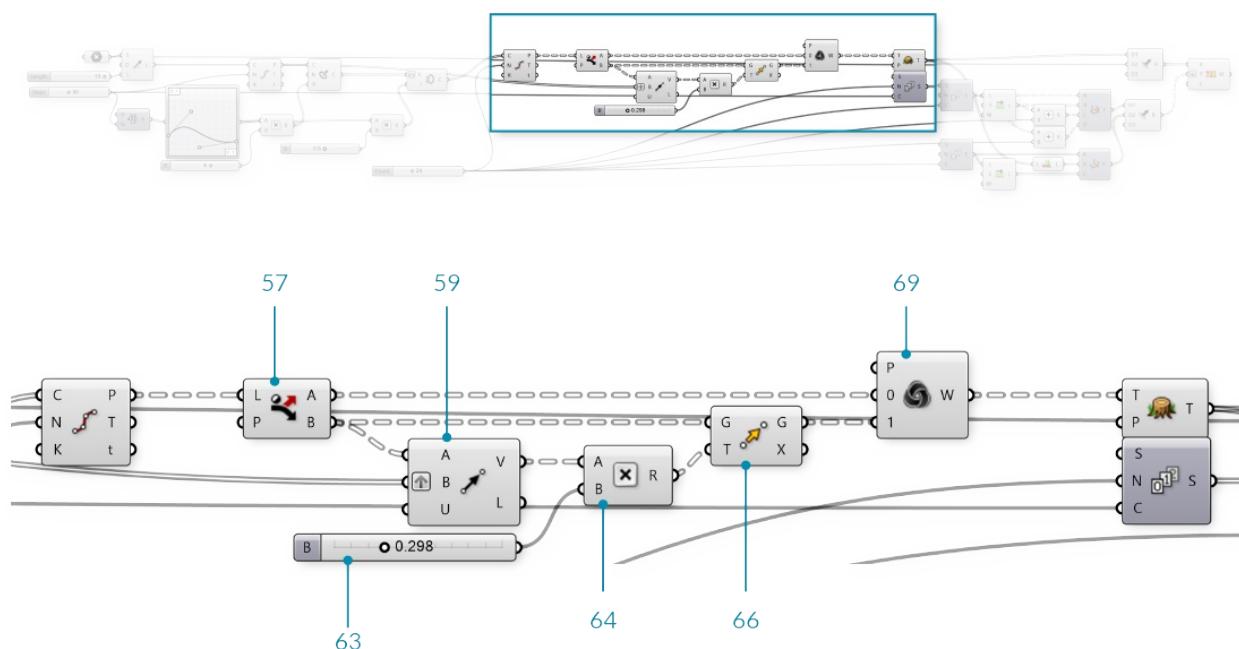


Соедините выход Point (P) второго компонента **Divide Curve** с входом List (L) компонента **Dispatch**

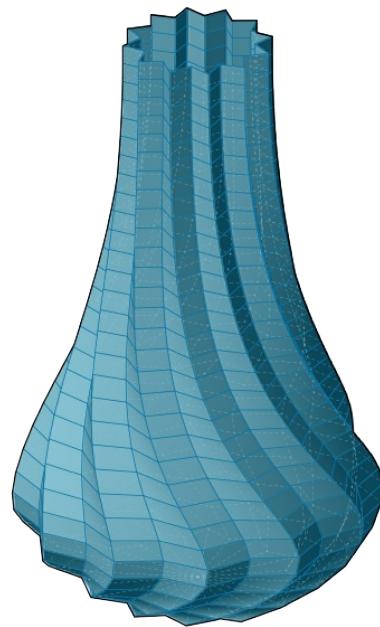
58.

Мы используем по умолчанию вход Pattern (P) компонента **Dispatch**, чтобы разделить точки на два списка с переменными точками

59.	Зайдите в Vector/Vector/Vector 2Pt - перетащите компонент Vector 2Pt на холст	
60.	Соедините выход B компонента Dispatch с входом A компонента Vector 2Pt	
61.	Соедините Points (P) первого компонента Divide Curve с входом B компонента Vector 2Pt	
62.	Кликните правой клавишей мыши по входу B компонента Vector 2Pt и выберите 'Graft', кликните правой клавишей по входу Unitize (U) , перейдите в 'Set Boolean' и выберите 'True' Это создаст вектор unit для каждой точки, которая указывает по направлению к центру круга	
63.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст. Мы будем использовать настройки по умолчанию	
64.	Зайдите в Maths/Operator/Multiplication - перетащите компонент Multiplication на холст	
65.	Соедините выход Vector (V) компонента Vector 2Pt с входом A , соедините слайдер Number Slider с входом B компонента Multiplication	
66.	Зайдите в Transform/Euclidean/Move - перетащите компонент Move на холст	
67.	Соедините выход B компонента Dispatch с входом Geometry (G) компонента Move	
68.	Соедините выход Result (R) компонента Multiplication с входом Motion (T) компонента Move	
69.	Зайдите в Sets/List/Weave - перетащите компонент Weave на холст	
70.	Соедините выход A компонента Dispatch с входом 0 компонента Weave	
71.	Соедините выход Geometry (G) компонента Move с входом 1 компонента Weave	
72.	Соедините выход Weave (W) компонента Weave с входом Tree (T) компонента Flatten Tree	

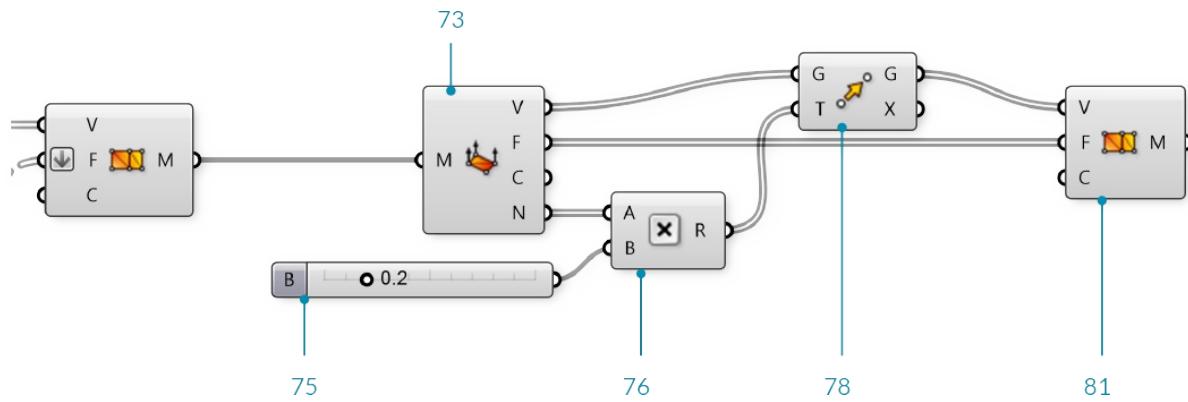
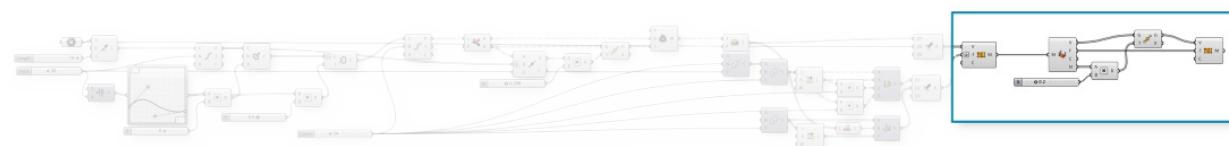


Помните о том, что нужно вернуться назад и настроить слайдеры и graph mapper, чтобы видеть как меняется модель и быть уверенным, что все работает правильно. Это называется "зондирование" модели и должно производиться часто для проверки на ошибки в определении.

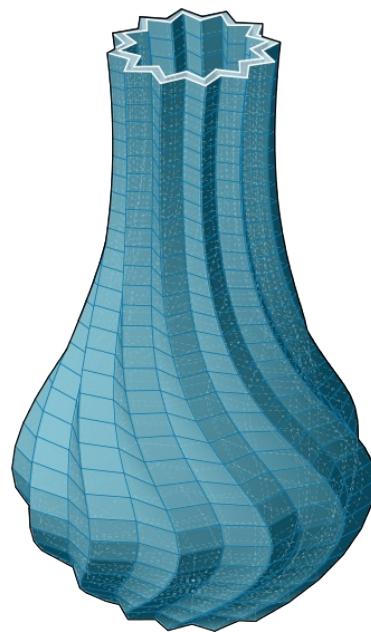


Теперь у нас есть единая поверхность для нашей вазы. Если мы захотим напечатать эту вазу на 3D принтере, нам потребуется, чтобы она была закрытым твердым телом. Мы создадим твердое тело путем смещения текущей mesh, затем объединим исходную mesh и mesh со смещением.

73.	Зайдите в Mesh/Analysis/Deconstruct Mesh - перетащите компонент Deconstruct Mesh на холст	
74.	Соедините выход Mesh (M) компонента Construct Mesh с входом Mesh (M) компонента Deconstruct Mesh	
75.	Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст. Мы будем использовать настройки по умолчанию	
76.	Зайдите в Maths/Operator/Multiplication - перетащите компонент Multiplication на холст	
77.	Соедините выход Normals (N) компонента Deconstruct Mesh с входом A, соедините слайдер Number Slider с входом B компонента Multiplication	
78.	Зайдите в Transform/Euclidean/Move - перетащите компонент Move на холст	
79.	Соедините выход Vertices (V) компонента Deconstruct Mesh с входом Geometry (G) компонента Move	
80.	Соедините выход Result (R) компонента Multiplication с входом Motion (T) компонента Move	
81.	Зайдите в Mesh/Primitive/Construct Mesh перетащите компонент Construct Mesh на холст	
82.	Соедините выход Geometry (G) компонента Move с входом Vertices (V) компонента Construct Mesh	
83.	Соедините выход Faces (F) компонента Deconstruct Mesh с входом Face (F) компонента Construct Mesh	



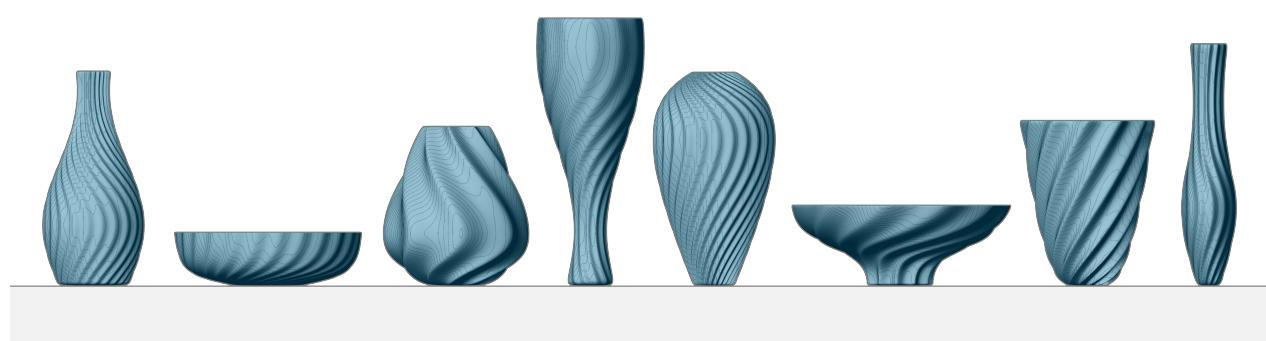
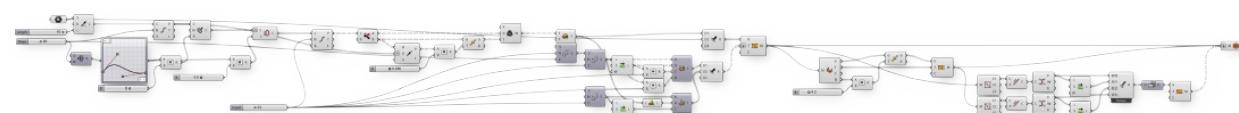
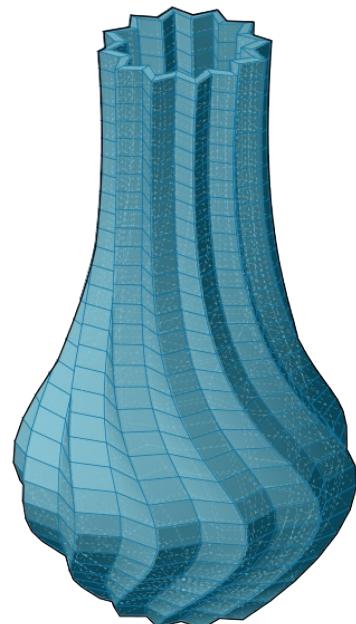
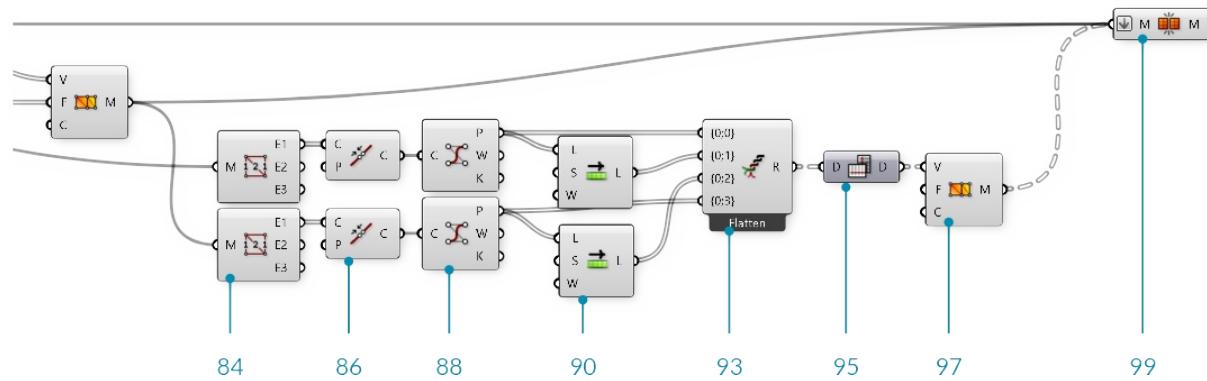
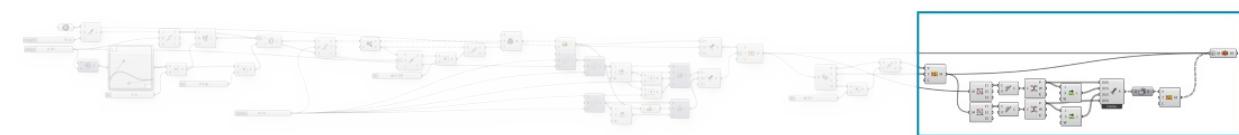
Сместив mesh в соответствии с нормалями вершин, мы теперь имеем "внутреннюю сторону" и "внешнюю сторону" mesh, но наверху между двумя геометриями mesh есть зазор



Последний шаг будет в создании закрытой mesh путем создания новой геометрии mesh, чтобы закрыть зазор, и затем соединим две mesh вместе.

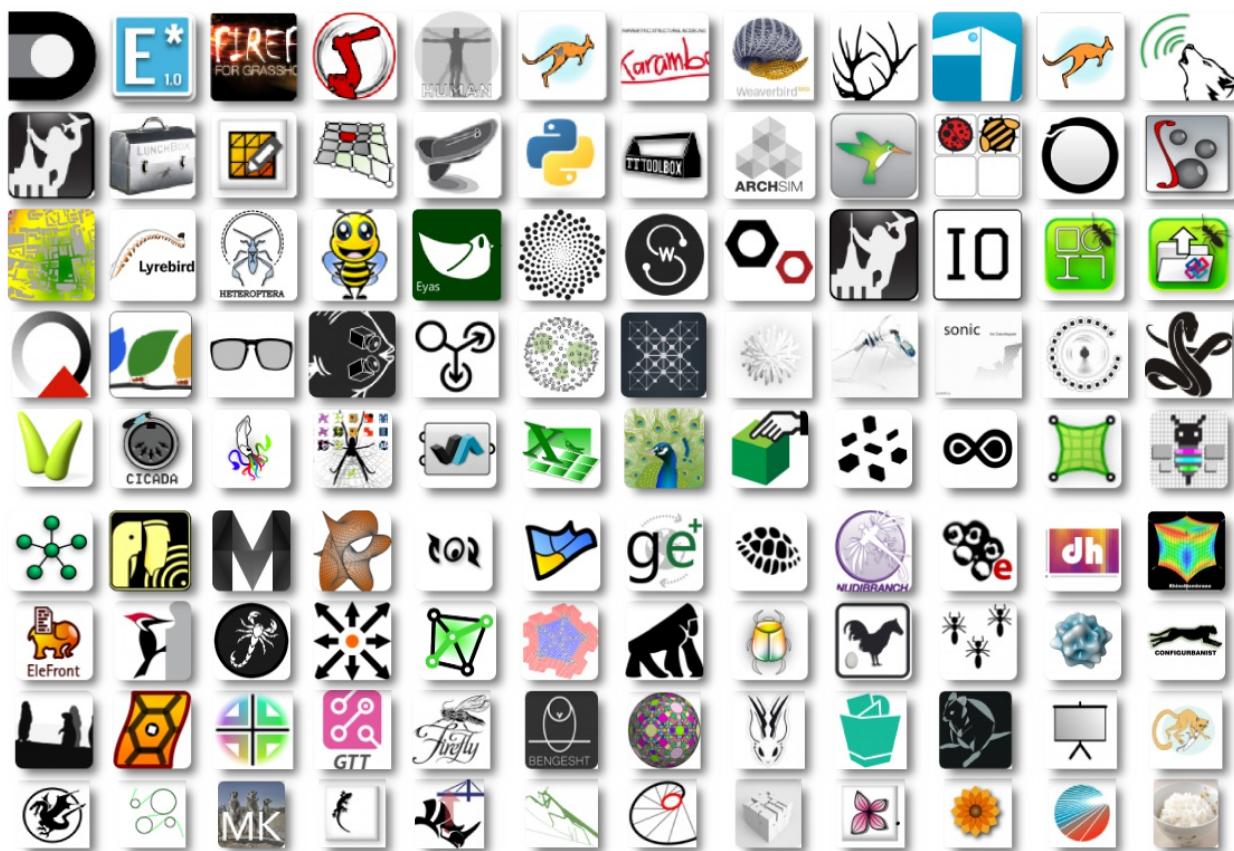
84.	Зайдите в Mesh/Analysis/Mesh Edges - перетащите компонент Mesh Edges на холст	
85.	Соедините выход Mesh (M) первого компонента Construct Mesh с входом Mesh (M) компонента Mesh Edges	
86.	Зайдите в Curve/Util/Join Curves - вытащите компонент Join Curves на холст	
87.	Соедините выход Naked Edges (E1) компонента Mesh Edges с входом Curves (C) компонента Join Curves	
88.	Зайдите в Curve/Analysis/Control Points - вытащите компонент Control Points на холст	

	Соедините выход Curves (C) компонента Join Curves с входом Curve (C) компонента Control Points	
89.	Путем объединения кривых и затем извлечения контрольных точек, мы убедились, что порядок точек последователен вдоль края вазы, что важно для того, чтобы итоговая mesh стала ориентированной и однородной	
90.	Зайдите в Sets/List/Shift List - вытащите компонент Shift List на холст	
91.	Соедините выход Points (P) компонента Control Points с входом List (L) компонента Shift List	
92.	Повторите шаги с 84 по 91 для второго компонента Construct Mesh	
93.	Зайдите в Sets/Tree/Entwine - вытащите компонент Entwine на холст	
94.	Приблизьте компонент Entwine пока не увидите опцию для добавления экстра входа. Нам потребуются четыре входа. Соедините их в следующем порядке: {0;0} - Points (P) из первого компонента **Control Points** {0;1} - выход из первого **Shift List** {0;2} - выход из второго **Shift List** {0;3} - Points (P) из второго компонента **Control Points**	
95.	Зайдите в Sets/Tree/Flip Matrix - вытащите компонент Flip Matrix на холст	
96.	Соедините выход Result (R) компонента Entwine с входом Data (D) компонента Flip Matrix	
97.	Зайдите в Mesh/Primitive/Construct Mesh - перетащите компонент Construct Mesh на холст	
98.	Соедините выход Data (D) компонента Flip Matrix с входом Vertices (V) компонента Construct Mesh	
99.	Зайдите в Mesh/Util/Mesh Join - перетащите компонент Mesh Join на холст	
100.	Соедините все три компонента Construct Mesh с компонентом Mesh Join зажав клавишу Shift и соединяя в это время связи (или использовать компонент Merge). Кликните правой клавишей мыши по входу Mesh (M) компонента Mesh Join и выберите 'Flatten'	



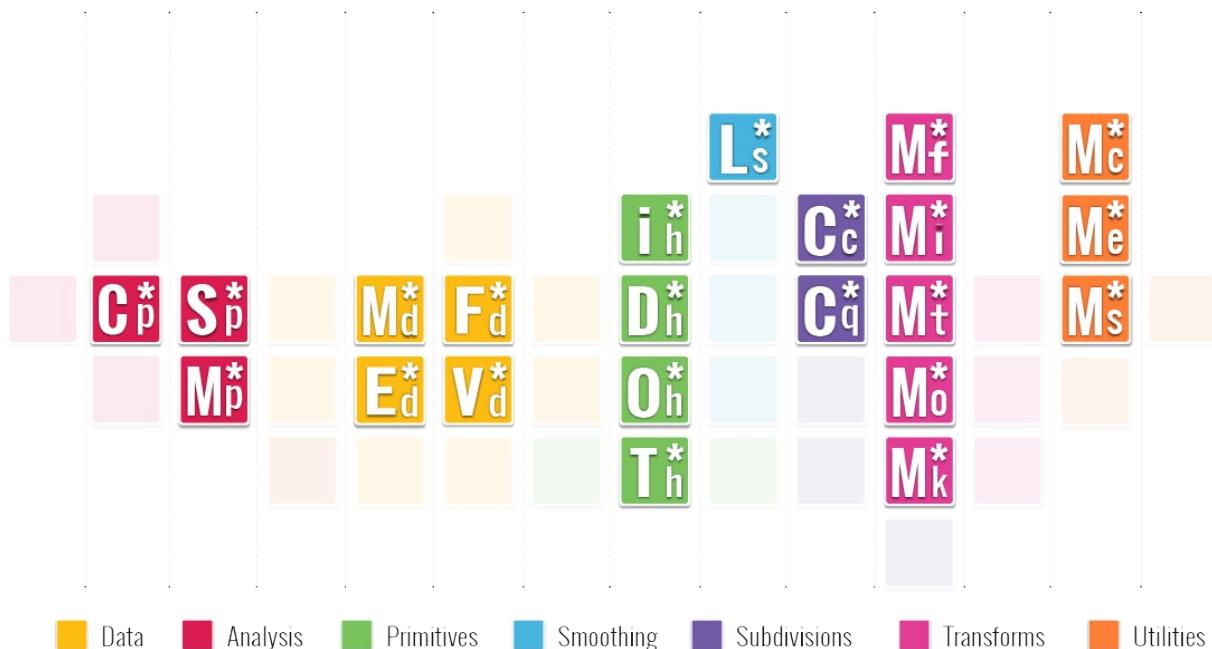
2. Расширения

Основные положения должны иметь поддержку. Эта часть представляет массив ключевых плагинов для Grasshopper, которые расширяют его функциональность и ваши возможности в проектировании.



2.1. Element*

Element* это плагин mesh геометрии для Grasshopper, позволяющий создавать mesh, анализировать, изменять, подразделять и сглаживать. **Element*** предоставляет доступ к топологическим данным mesh, используя структуру данных half-edge Plankton для полигонов mesh.



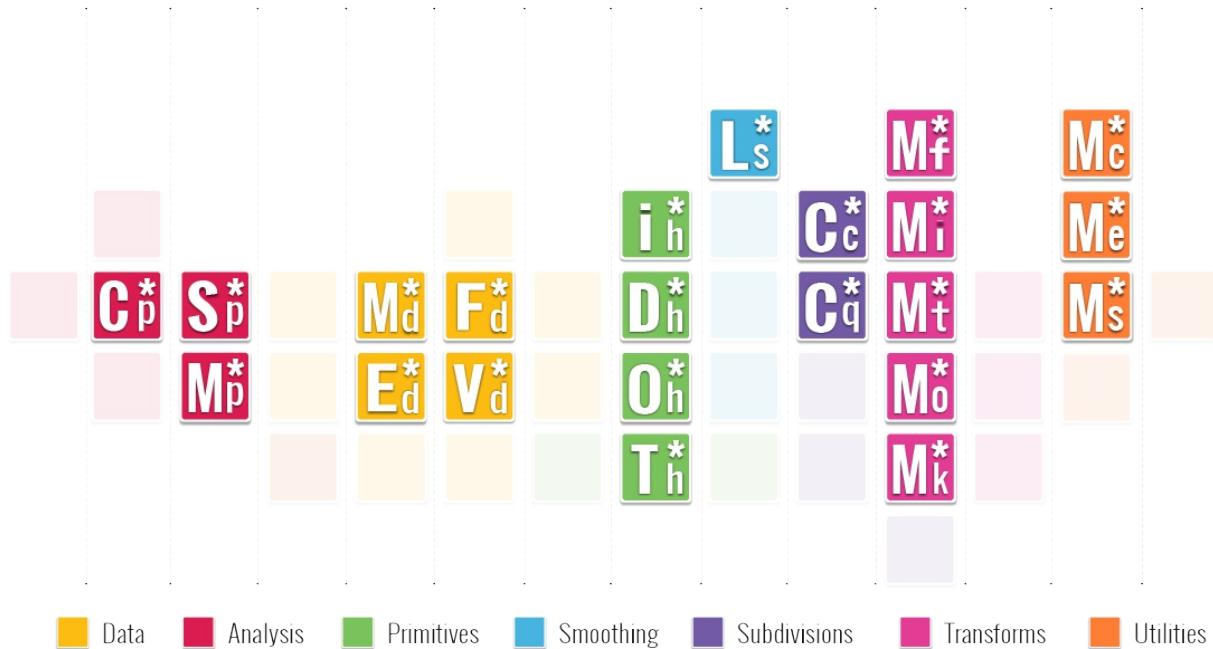
2.1.1. Element*

Интегрируя Mesh в вашу работу, вам предоставляется широкий набор возможностей создания форм, от граненных до сглаженных. Element* позволяет идти еще дальше, благодаря интуитивному подходу к анализированию топологии mesh и сглаживанию рутинной работы. Эта глава, по сути, руководство по использованию плагина Element* версия 1.1, которая введет вас в курс дела.

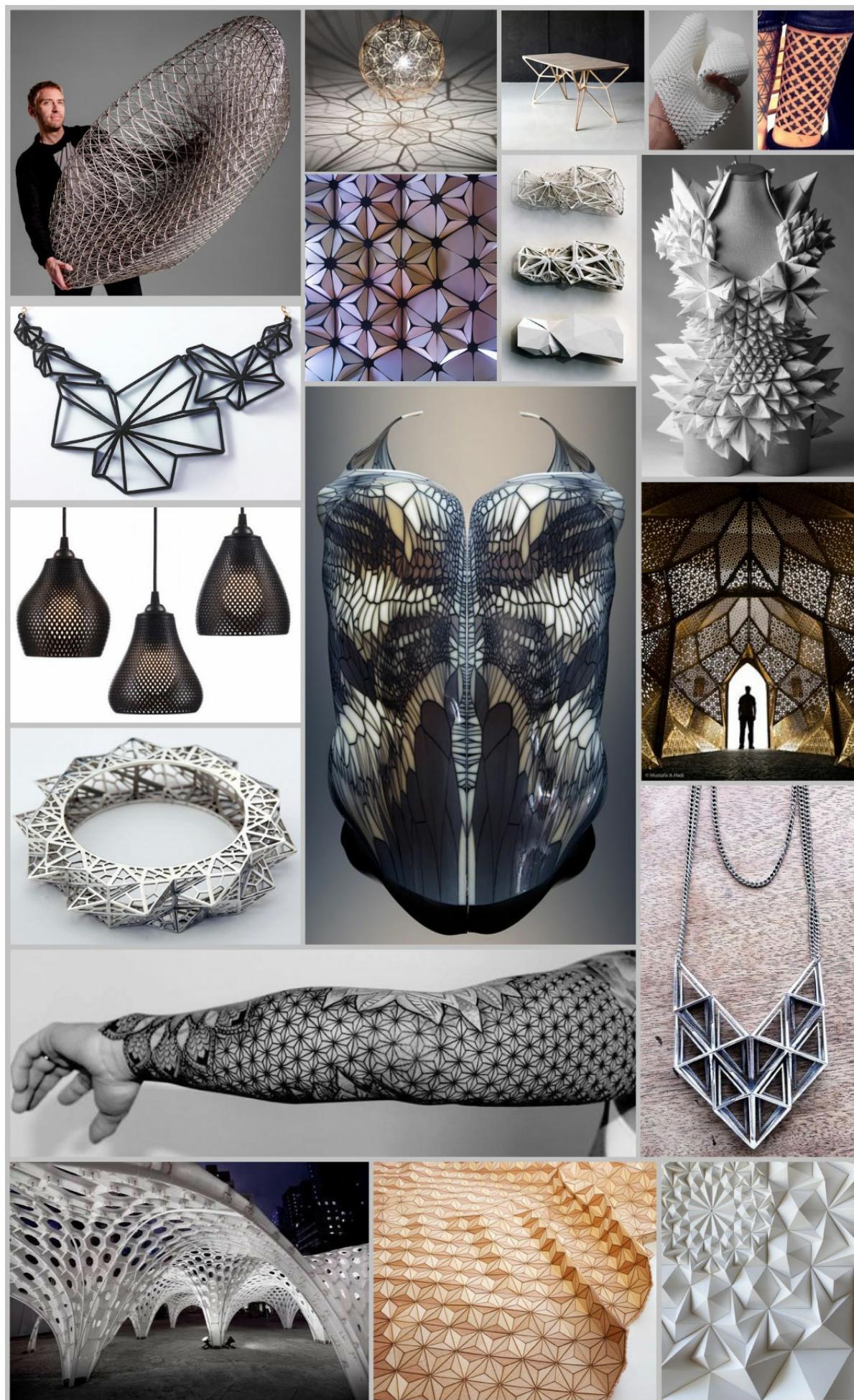
Скачайте плагин Element* , чтобы начать



Компоненты Element* разбиты на категории на основе их действий. Также как Периодическая таблица химических элементов представляет собой структуру для анализа химических процессов, так и Element представляет структуру для анализа и исследования геометрии, основываясь на данных и операциях mesh. Мы думаем, что новые компоненты будут созданы основываясь на анализе взаимоотношений между компонентами в каждой категории.



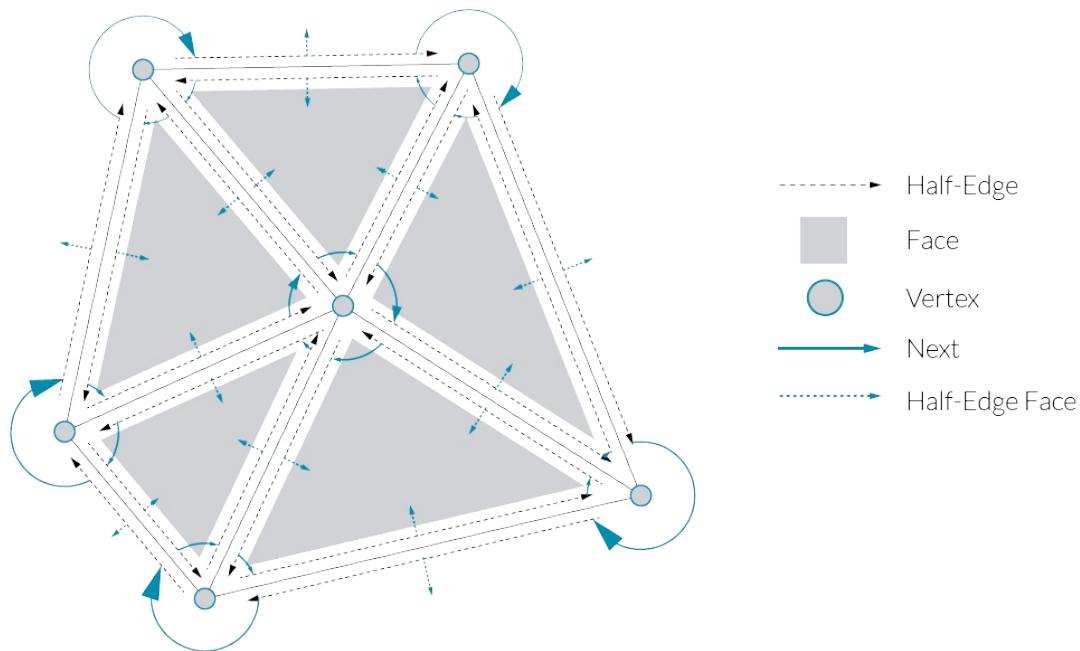
Ниже представлены несколько вдохновляющих изображений продуктов, которые могли бы быть созданы при помощи Element*.



2.1.2. Данные Half Edge

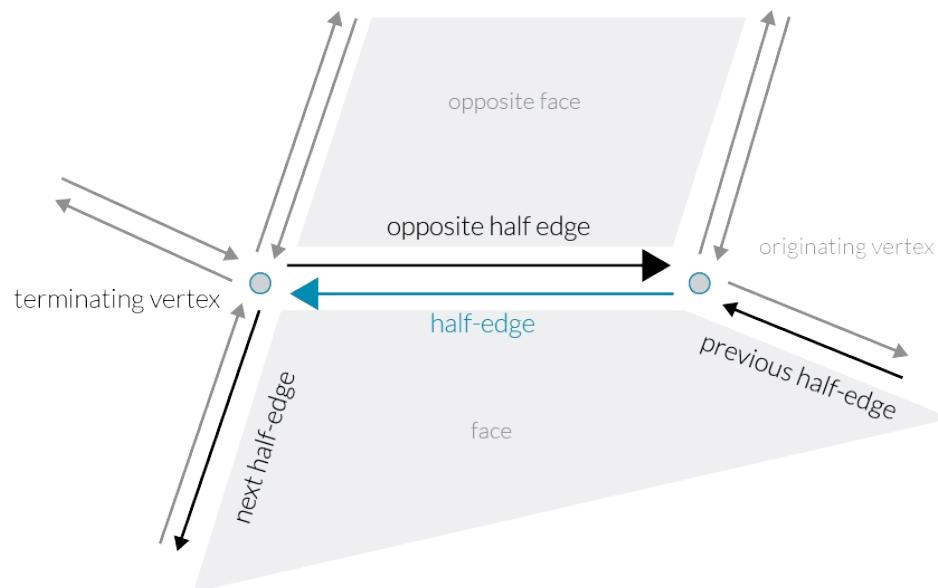
В пособии Grasshopper мы рассматривали как Grasshopper определяет mesh, используя структуру данных Полигон-Вершина. Это, относительно, простая структура данных, она широко используется в приложениях, которые используют mesh, но может быть неэффективной для вычисления более сложных алгоритмов. Аддон Element* реструктурирует mesh, используя данные Half-Edge: структура данных, ориентированная на ребра, которая выдает удобные запросы о смежных вершинах, полигонах и ребрах, что может улучшить скорость и производительность алгоритма. Эта структура способна сохранять информацию о вершинах, ребрах и полигонах. Этот метод способствует созданию новых паттернов и геометрий, основанных на топологических отношениях базовой геометрии.

Структура данных half-edge является представлением mesh, в котором каждое ребро делится на два полу-ребра с противоположными направлениями. Это предоставляет нам открытый и косвенный доступ к данным как одного элемента mesh, так и к смежным элементам.



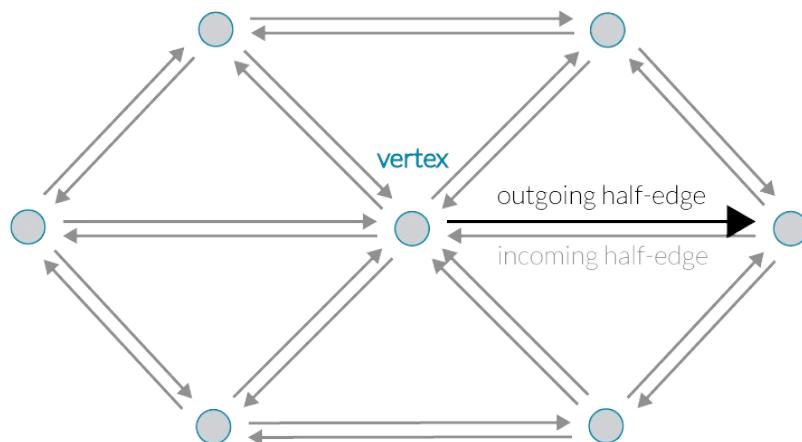
2.1.2.1 Связанность Half-Edge

Выделенные голубым цветом полу-ребра открыто хранят индексы их конечных точек, смежных полу-ребер и полигонов, к которым они принадлежат. К другой информации (серый цвет) можно получить косвенный доступ.

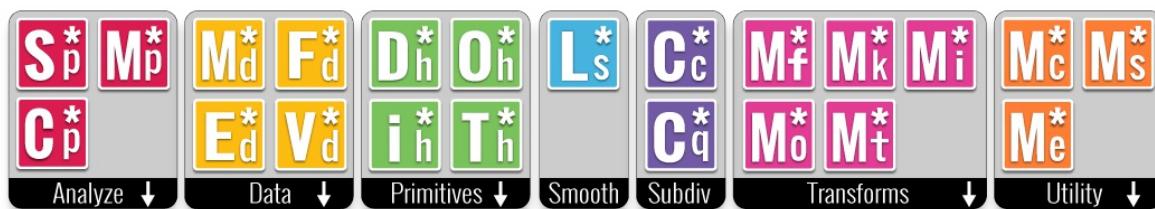


2.1.2.2 Связанность Вершин

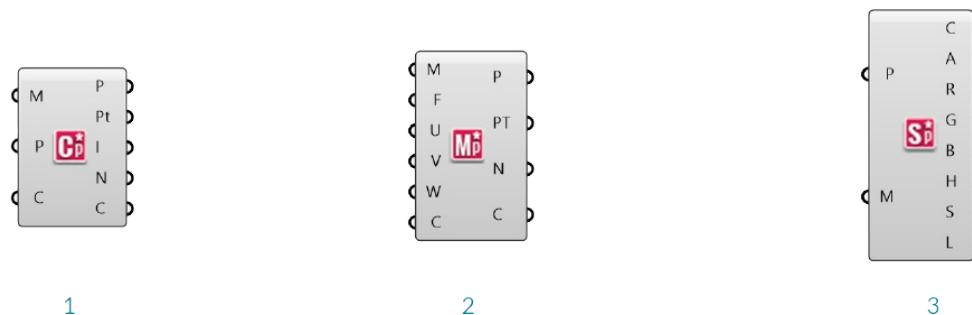
Выделенная голубым цветом вершина открыто хранит индекс одной из ее исходящих полу-ребер. К другой информации (серый цвет) можно получить косвенный доступ.



2.1.3. Компоненты Element*



2.1.3.1 Analyse (Анализ)



1. Mesh Closest Point
2. Mesh Evaluate
3. Mesh Sample Plus

Element* Mesh Closest Point (Ближайшая точка Mesh)

В отличие от компонента Grasshopper **Mesh Closest Point** этот компонент также вычисляет нормаль и цвет выведенной точки, устранив необходимость в компоненте **Mesh Eval** и упрощая пространство холста.

Element* Mesh Evaluate (Определение Mesh)

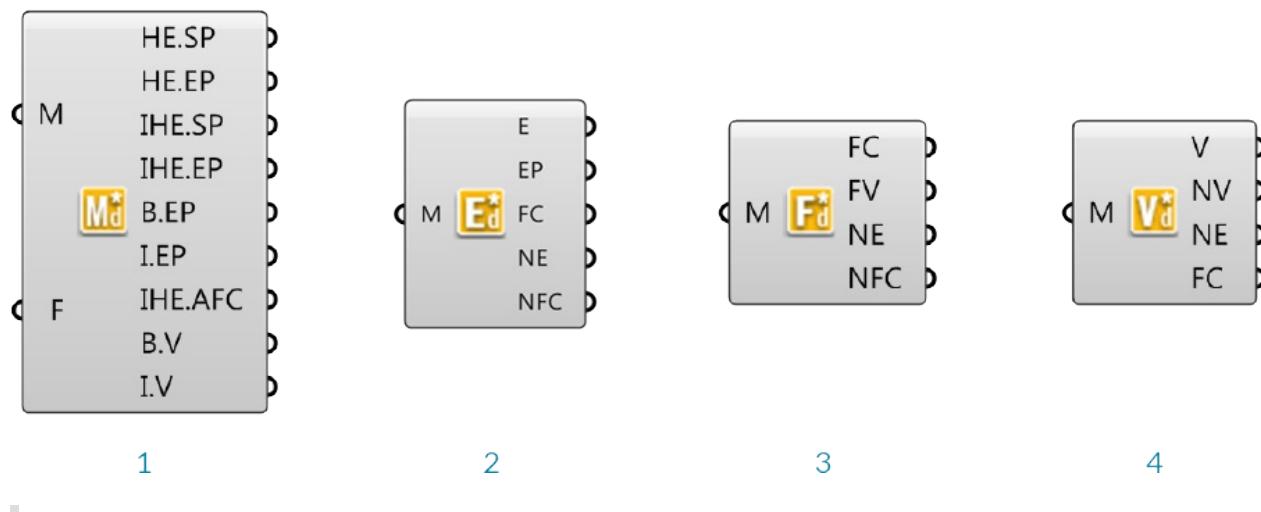
Встроенный компонент Grasshopper **Mesh Eval** требует параметр `mesh` в качестве ввода, который можно извлечь из компонента **Mesh Closest Point*** но который трудно создать вручную. Компонент Element Closest Point предоставляет прямой ввод индекса полигона `mesh` и барицентрические координаты.

Примечание - барицентрические координаты определяются таким образом, что они всегда прибавляют 1. Если значения ввода U, V и W не прибавляют 1, этот компонент будет поддерживать коэффициент трех значений, в то же время нормализуя их. Например, если у вас имелись значения ввода 2, 2 и 4, параметр `mesh` был бы рассчитан как {0.25;0.25;0.5}

Element* Mesh Sample Plus

Этот компонент используется, чтобы быстро извлечь информацию о цвете из `mesh`. Он выдает значения Альфа, Красный, Зеленый, Голубой, Тон, Насыщенность и Яркость введенных точек. Если данные точки находятся не на `mesh`, этот компонент будет брать для расчета ближайшую точку. Этот компонент использует параллельное вычисление для быстроты.

2.1.3.2 Data (Данные)



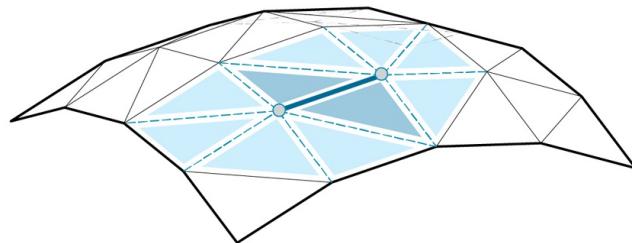
1. Data Visualizer
2. Edge Neighbors
3. Face Neighbors
4. Vertex Neighbors

Element* Data Visualizer (Визуализация Данных)

Этот компонент используется для визуализации данных half-edge полигонов вводной mesh.

Element* Edge Neighbors (Соседние Ребра)

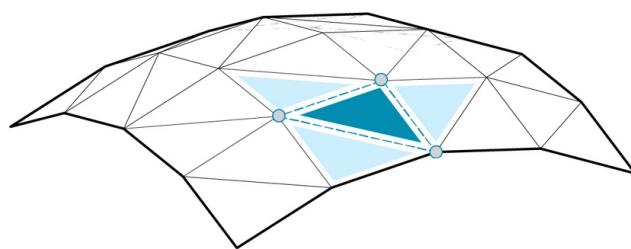
Этот компонент предоставляет доступ к смежным данным, которые структурируются в соответствии с ребрами вводной mesh. Данные на выходе представляются как дерево с одной веткой для каждого ребра mesh. Он выдает ребра mesh, конечные точки ребра, центральные точки на полигонах смежные с каждым ребром (дуальные), соседние ребра как линейные объекты (упорядочены по часовой стрелке), и центры соседних полигонов (центральные точки полигонов смежные с начальной и конечной точками ребра).



Edge Neighbors - ребра, конечные вершины, центры смежных полигонов, соседние ребра и центры соседних полигонов

Element* Face Neighbors (Соседние Полигоны)

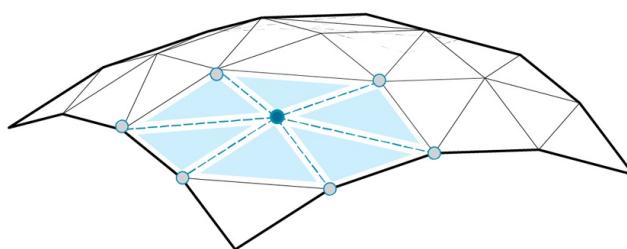
Этот компонент похож на другие из этого раздела, но данные организуются в виде дерева, в соответствии с полигонами mesh, с одной веткой на полигон. Выходы - это центры полигона, вершины каждого полигона (упорядочены против движения часовой стрелки), соседние ребра (упорядочены против движения часовой стрелки) и центры соседних полигонов (упорядочены против движения часовой стрелки).



Face Neighbors - центры полигона, вершины полигона, соседние ребра, центры соседнего полигона

Element* Vertex Neighbors (Соседние Вершины)

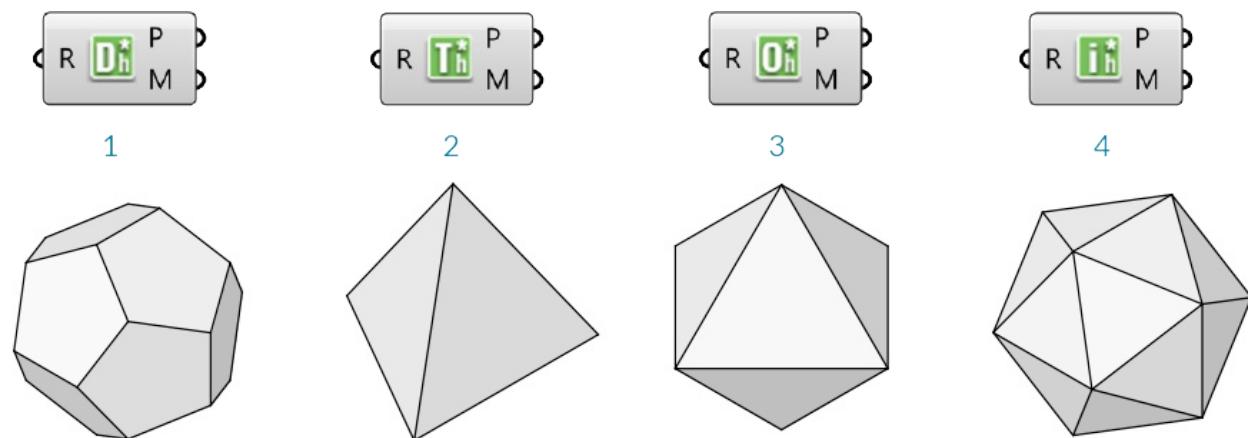
Этот компонент выдает вершины mesh, соседние вершины (упорядочены по часовой стрелке), соседние ребра (упорядочены по часовой стрелке) и центры соседнего полигона (упорядочены по часовой стрелке), в виде дерева в соответствии с вершинами mesh.



Vertex Neighbors - Вершины, соседние вершины, соседние ребра, центры соседнего полигона

2.1.3.3 Primitives (Примитивы)

Element* предоставляет четыре дополнительных mesh примитива: додекаэдр, тетраэдр, октаэдр и икосаэдр. Эти компоненты берут одно число как вход для радиуса и производят несколько mesh, с центром в начале системы координат и состоящих из одного полигона на одной стороне. Добавив Куб, который является уже встроенным примитивом в Grasshopper, они представляют пять Платоновых тел.

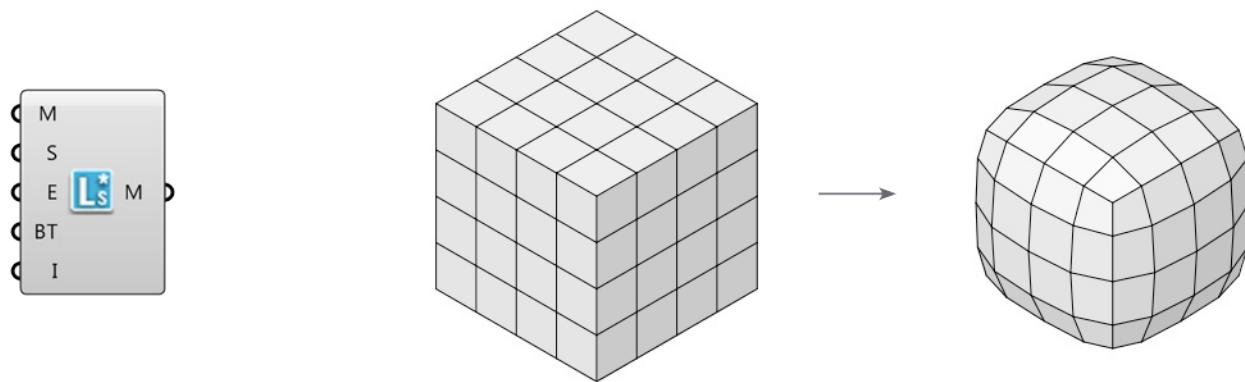


1. Додекаэдр

2. Тетраэдр
3. Октаэдр
4. Икосаэдр

2.1.3.4 Smooth (Сглаживание)

Element* Smooth предоставляет оптимизированный сглаженный алгоритм, который более продуктивен, чем компонент Grasshopper **Smooth Mesh** для огромных массивов данных. Он использует алгоритм Lapacian Smoothing для Half-Edge структурированных mesh. Он не меняет топологию или количество вершин спаенных mesh, но он будет объединять идентичные вершины, если имеются любые дубликаты, причиной которых является неспаянная mesh. Мы можем указать силу сглаживания, граничные условия, граничную точность, а также число итераций.



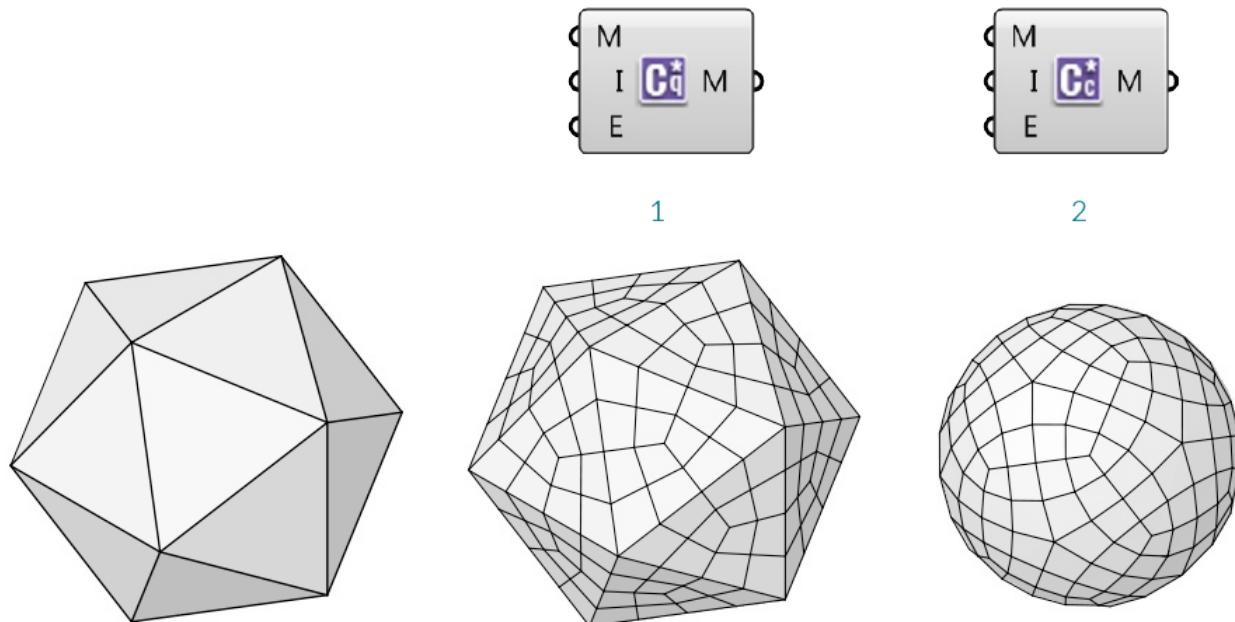
2.1.3.5 Subdivide (Подразделение)

Element* Catmull Clark Subdivision

Это рекурсивное подразделение, определяемое алгоритмом Catmull Clark. Мы можем указать число итераций, а также то, как поступать с naked ребрами.

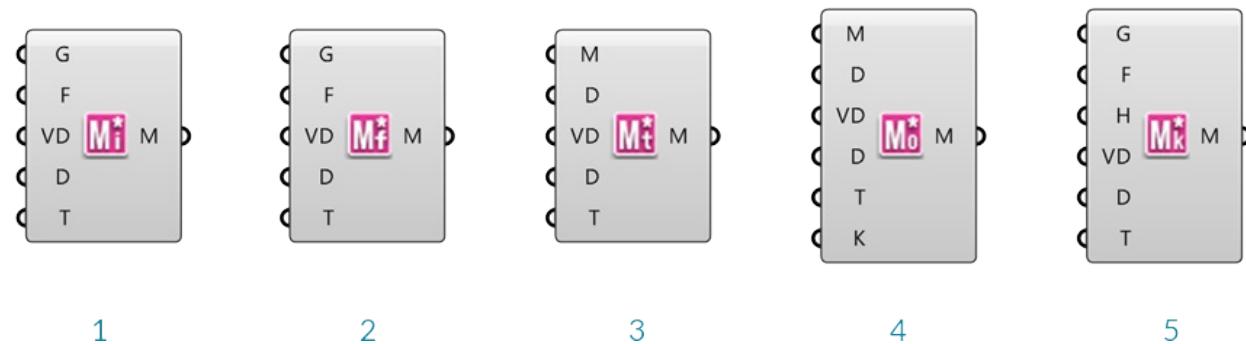
Element* Constant Quad

Этот компонент подразделения создает все четырехугольные mesh путем добавления полигона для каждого ребра mesh.



1. Constant Quad подразделение
2. Catmull Clark подразделение

2.1.3.6 Transform (Трансформация)



1. Mesh Window
2. Mesh Frame
3. Mesh Thicken
4. Mesh Offset
5. Mesh Poke Face

Эти компоненты предоставляют различные трансформации, которые описываются ниже. Каждый компонент имеет дополнительную способность принятия данных о расстоянии reg-vertex, что позволяет вариации трансформации амплитуды через mesh.

Element* Mesh Window

Реконструирует новую mesh внутри полигона, основываясь на значении смещения. Этот компонент принимает либо mesh либо список закрытых полилиний как вход.

Element* Mesh Frame

Выдает рамку вокруг полигонов mesh. У каждого итогового полигона будет новое отверстие в центре. Этот компонент принимает либо mesh либо список закрытых полилиний как ввод.

Element* Mesh Thicken

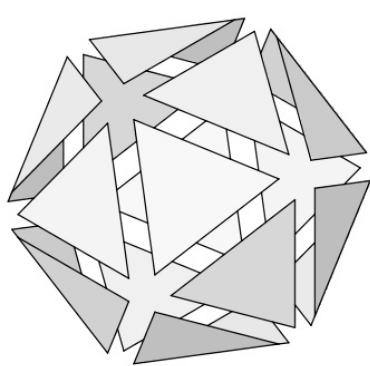
Этот компонент будет утолщать вход mesh вдоль нормали вершин, а также в соответствии с предоставленными значениями расстояния.

Element* Mesh Offset

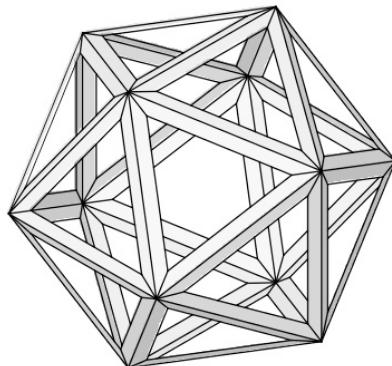
Этот компонент создает смещение вводной mesh на основе вершин нормали.

Element* Mesh Poke Face

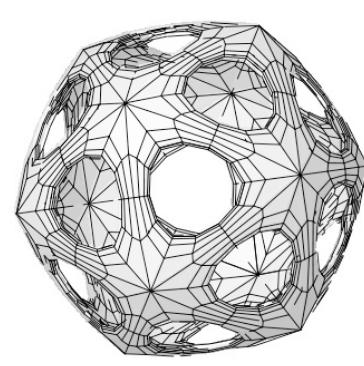
Сначала, полигон mesh проходит через операцию фрейминга, затем внутренний полигон разделяется на выбранные полигоны и позволяет пользователю указывать силу притягивания или отталкивания от центра исходного полигона. Например, четырехсторонний полигон (четырехугольник) разделяется на 4 трехсторонних полигона с одной общей вершиной в середине. Ввод высоты позволяет трансформировать эту вершину.



1

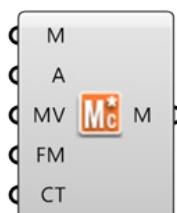


2



3

1. Mesh Window
2. Mesh Frame
3. Икосаэдр, измененный вместе с mesh фреймингом, затем применено утолщение и подразделение

2.1.3.7 Utility

1



2



3

1. Mesh Combine & Clean
2. Mesh Edges
3. Mesh Status

The **Element* Mesh Combine and Clean**

Этот компонент объединяет множество mesh и обладает опциями для спаивания mesh на основе вводного угла или для объединения одинаковых вершин. Мы также можем перевернуть ориентацию mesh. Этот компонент также определяет возможные вопросы по топологии и выдает Замечания и Предупреждения с подробными объяснениями. В случае, если объединение одинаковых вершин создает плохую топологию, компонент вернет вводный список mesh вместо объединенных и смешанных mesh. Пользователь также может выбрать объединение mesh без смешивания каких-либо из ее вершин.

The **Element* Mesh Edges**

Этот компонент выдает naked ребра mesh, ребра mesh, полилинии полигона и, если mesh неспаенная, он выдаст неспаенные ребра mesh.

The **Element* Mesh Status**

Этот компонент выдает информацию о mesh, на основе топологии. Имеются два режима, в которых мы можем просматривать информацию, первый - Mesh Info, который выдает данные о геометрии, такие как правильность mesh, количество вершин, количество полигонов и количество нормалей. Другой режим выдает Статус Mesh, который рассказывает о состоянии mesh, есть ли у нее неоднородные ребра, количество неправильных полигонов, количество naked ребер и количество необъединенных mesh. Этот компонент не работает с mesh, он просто выдает данные пользователю. Также есть опция для объединения одинаковых вершин, таким образом, что пользователь может видеть его влияние на mesh.

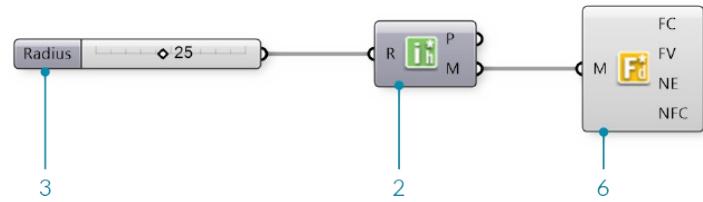
2.1.4. Упражнение

В этом разделе мы проработаем простое упражнение, взяв примитивы Element* за основу. Мы внедрим half-edge структуру данных, а также используем обе характеристики компонентов трансформации (uniform и per vertex)



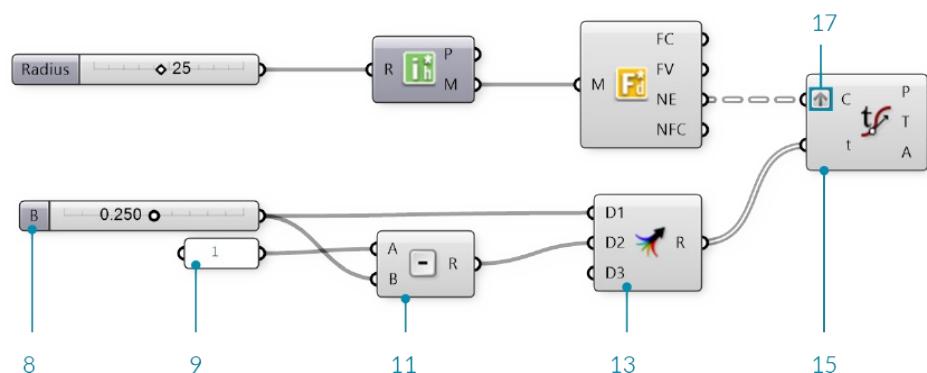
Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

01.	Запустите новое определение, набрав Ctrl+N (в Grasshopper)	
02.	Зайдите в Element*/Primitive/Icosohedron - перетащите компонент Icosohedron на холст	
03.	Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст	
04.	Подключите Number Slider к входу Radius (R) в компоненте Icosohedron	
05.	Дважды кликните по Number Slider и установите подходящие значения. В этом примере мы использовали: Name: Radius (Радиус) Rounding: Integer Lower Limit: 5 Upper Limit: 50 Value: 25	
06.	Зайдите в Element*/Data/Face Neighbors - перетащите компонент Face Neighbors на холст	
07.	Соедините выход Mesh (M) компонента Icosohedron с входом Mesh (M) компонента Face Neighbors .	



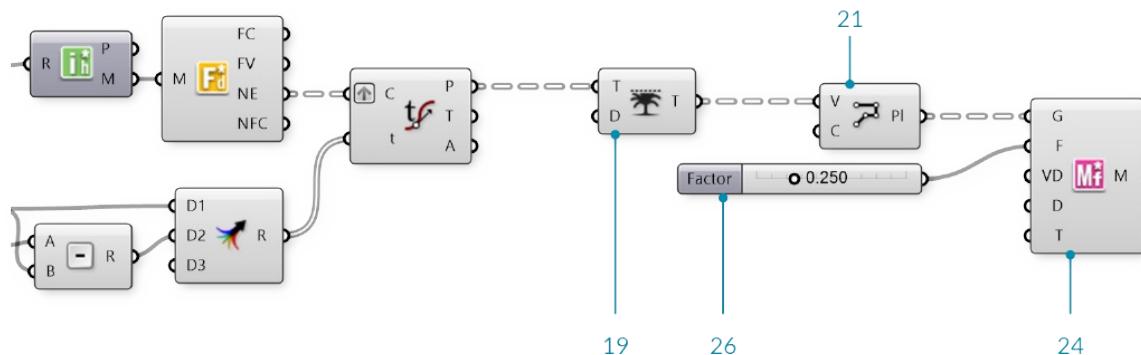
Посмотрев на данные выхода Neighboring Face Edges (NE), мы видим дерево и 20 ветками, где каждая ветка содержит три линии. 20 веток каждая представляет полигон икосаэдра, у которого 20 сторон, в то время как три линии - это ребра каждого триангулированного полигона.

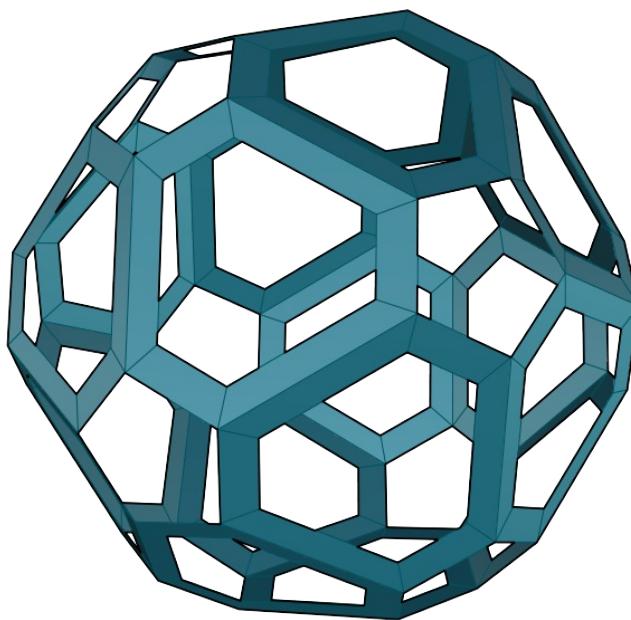
	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Rounding: Float Lower Limit:0 Upper Limit: 0.5	
08.		
09.	Зайдите Params/Input/Panel - вытащите компонент Panel на холст	
10.	Дважды кликните по Panel и введите "1" в текстовое поле	
11.	Зайдите в Math/Operators/Subtraction - перетащите компонент Subtraction на холст	
12.	Соедините Panel с значением "1" с входом A, соедините слайдер с входом B компонента Subtraction	
13.	Зайдите в Sets/Tree/Merge - вытащите компонент Merge на холст	
14.	Соедините Number Slider с входом D1 компонента Merge , соедините выход R компонента Subtraction с входом D2 компонента Merge	
15.	Зайдите Curve/Analysis/Evaluate Curve - вытащите компонент Evaluate Curve на холст	
16.	Соедините выход Face Edges (NE) компонента Face Neighbors с входом Curve (C) компонента Evaluate Curve	
17.	Кликните правой клавишей по входу Curve (C) компонента Evaluate Curve и выберите Graft. Это создаст новую ветку для каждого ребра.	
18.	Соедините выход Result (R) компонента Merge с входом Parameter (t) компонента Evaluate Curve Из-за того, что мы сделали graft с входом Curve, каждое ребро определяется по обоим параметрам компонента Merge	



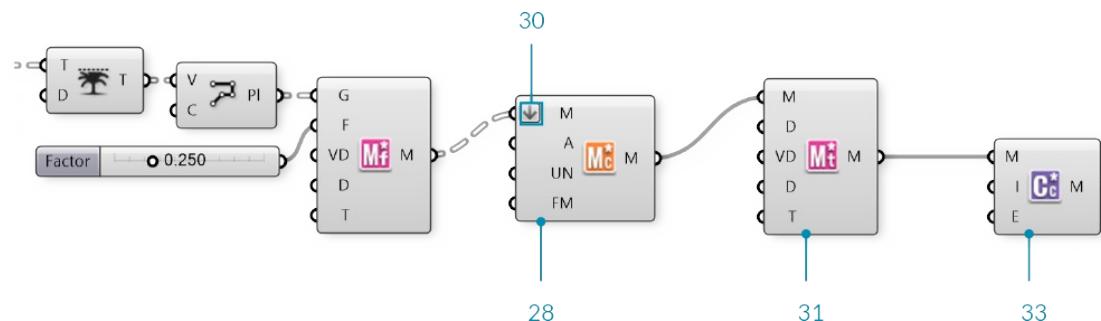
19.	Зайдите в Sets/Tree/Trim Tree - вытащите компонент Trim Tree на холст	
	Соедините выход Points (P) компонента Evaluate Curve с входом Tree (T) компонента Trim Tree	

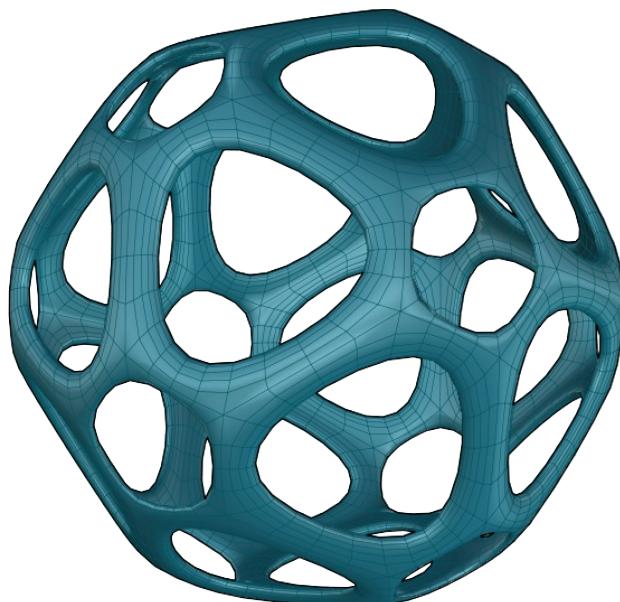
	Trim Tree. Значение входа Depth (D) по умолчанию для **Trim Tree** равно 1. Это сокращает глубину нашего дерева данных на один уровень путем смешения внешних веток. В результате у нас получается 20 веток, каждая с шестью точками.	
21.	Зайдите в Curve/Spline/Polyline - перетащите компонент Polyline на холст	
22.	Соедините выход Tree (T) компонента Trim Tree с входом Vertices (V) компонента Polyline	
23.	Кликните правой клавишей мыши по входу Closed (C) компонента Polyline , кликните по "Set Boolean" и выберите значение True Так мы создали закрытую полилинию из шести сторон для каждого исходного полигона mesh.	
24.	Зайдите в Element*/Transform/Mesh Frame - перетащите компонент Mesh Frame на холст	
25.	Соедините выход Polyline (Pl) компонента Polyline с входом Geometry (G) компонента Mesh Frame Заметьте, что компонент **Mesh Frame** может принимать в качестве входа как mesh так и список закрытых полилиний кривых	
26.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст. Мы будем использовать по умолчанию диапазон от 0 до 1 для этого слайдера	
27.	Подключите Number Slider к входу Factor (F) компонента Mesh Frame	





28.	Зайдите в Element*/Utility/Mesh Combine and Clean - перетащите компонент Mesh Combine and Clean на холст	
29.	Соедините выход Mesh (M) компонента Mesh Frames с входом Mesh (M) компонента Mesh Combine and Clean	
30.	Кликните правой клавишей мыши по входу Mesh (M) компонента Mesh Combine and Clean и выберите Flatten Применяя Flatten для дерева mesh, **Combine and Clean** будет смешивать все 20 полигонов mesh в одну mesh	
31.	Зайдите в Element*/Transform/Mesh Thicken - перетащите компонент Mesh Thicken на холст	
32.	Соедините выход Mesh (M) компонента Combine and Clean с входом Mesh (M) компонента Mesh Thicken	
33.	Зайдите в Element*/Subdivide/Catmull Clark Subdivision - перетащите компонент Catmull Clark Subdivision на холст	
34.	Соедините выход Mesh (M) компонента Mesh Thicken с входом Mesh (M) компонента Catmull Clark Subdivision	

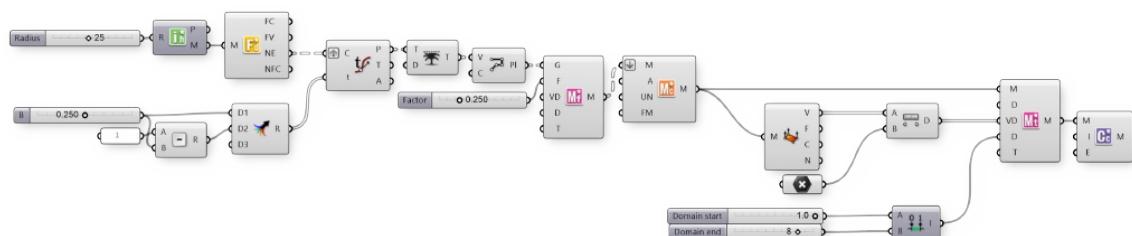
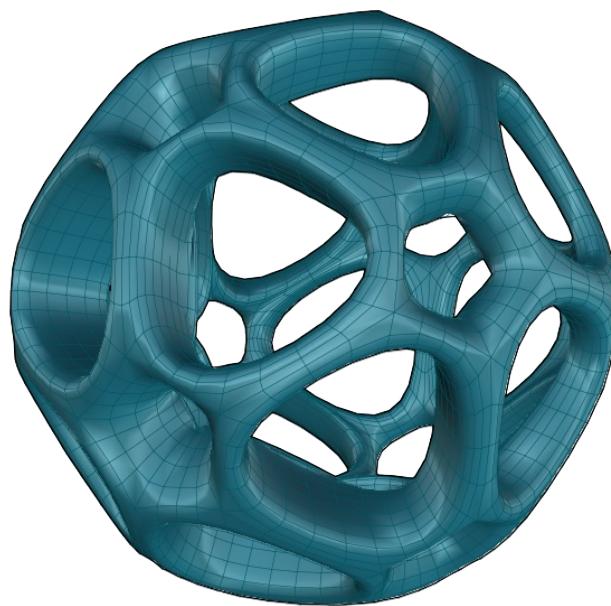
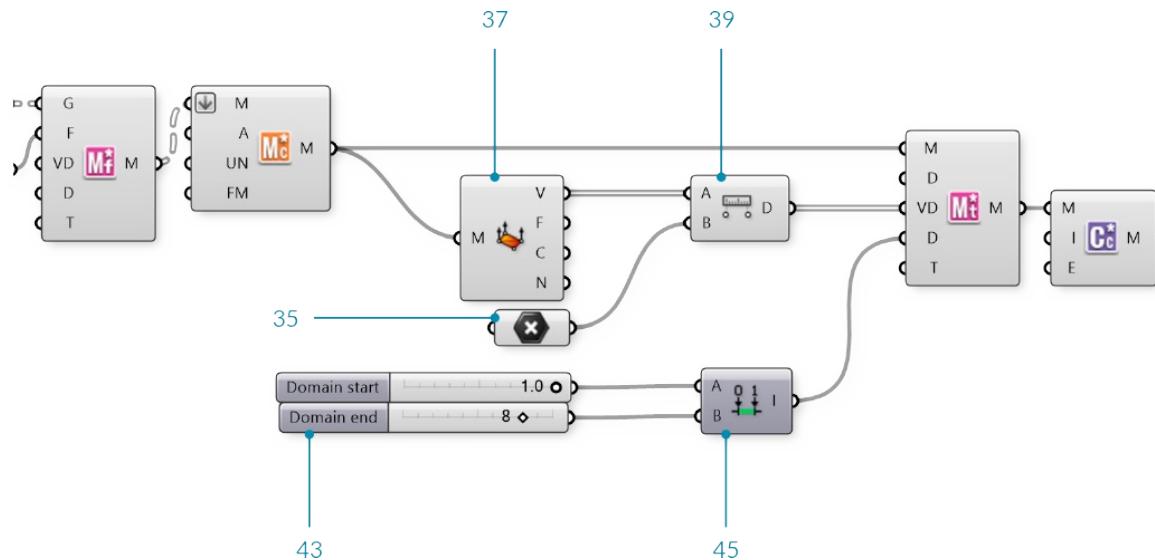




Мы срезали верхушку тетраэдрических полигонов исходной mesh, тем самым также создали кольца вокруг каждой исходной вершины. Мы также создали рамку для каждого полигона, затем утолщили mesh и детализировали ее с помощью подразделения. Далее мы используем преимущество Per Vertex для трансформации компонентов, используя атTRACTор.

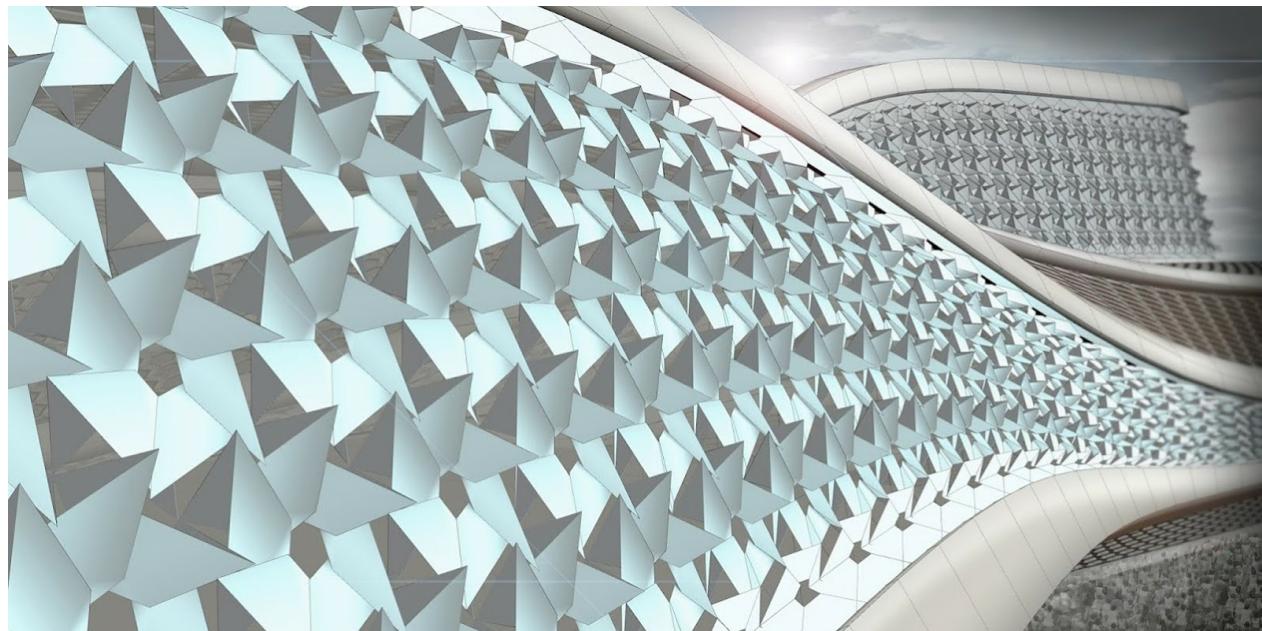
35.	Зайдите в Params/Geometry/Point - перетащите параметр Point на холст	
36.	Кликните правой клавишей мыши по параметру Point и кликните по "Set on point" выберите точку в видовом окне Rhino Подсказка - вы также можете создать точку прямо в Grasshopper дважды кликнув по холсту, чтобы появилось окно Поиска, затем указать координаты точки, такие как "10,10,0" (без кавычек)	
37.	Зайдите в Mesh/Analysis/Deconstruct Mesh - перетащите компонент Deconstruct Mesh на холст	
38.	Соедините выход Mesh (M) компонента Combine and Clean с входом Mesh (M) компонента Deconstruct Mesh . Это для того, чтобы извлечь вершины нашей объединенной mesh, а затем применить атTRACTор к этим вершинам	
39.	Зайдите в Vector/Point/Distance - перетащите компонент Distance на холст	
40.	Соедините выход Vertices (V) компонента Deconstruct Mesh с входом A компонента Distance	
41.	Соедините компонент Point с входом B компонента Distance	
42.	Соедините выход Distance (D) компонента Distance с входом PerVertex Data (VD) компонента Thicken	
43.	Зайдите в Params/Input/Number Slider - вытащите два слайдера Number Slider на холст. Мы будем использовать их для установки нижней и верхней границ для компонента Mesh Thicken	
44.	Дважды кликните по Number Sliders слайдеру и установите значения. В этом примере, мы оставим у первого слайдера значения по умолчанию, а у второго слайдера установим верхнюю границу на 5.0	

45.	на холст	
46.	Соедините два слайдера с входами A и B компонента Construct Domain	
47.	Соедините выход Domain (I) компонента Construct Domain с входом Min and Max Values (D) компонента Mesh Thicken .	
48.	Кликните правой клавишей мыши по входу Type (T) компонента Thicken , выберите "Set Integer" и введите значение 1 Вы также можете включить PerVertex данные, используя компонент Boolean Toggle установленный на True.	



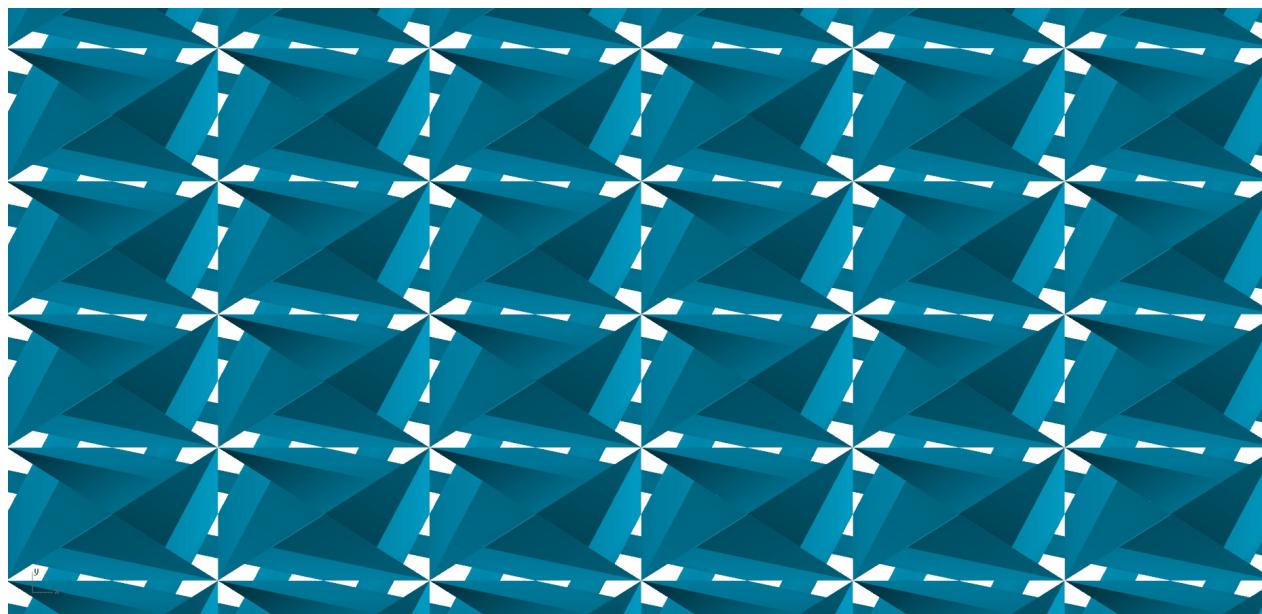
2.1.5. Element* Изучение применения в архитектуре

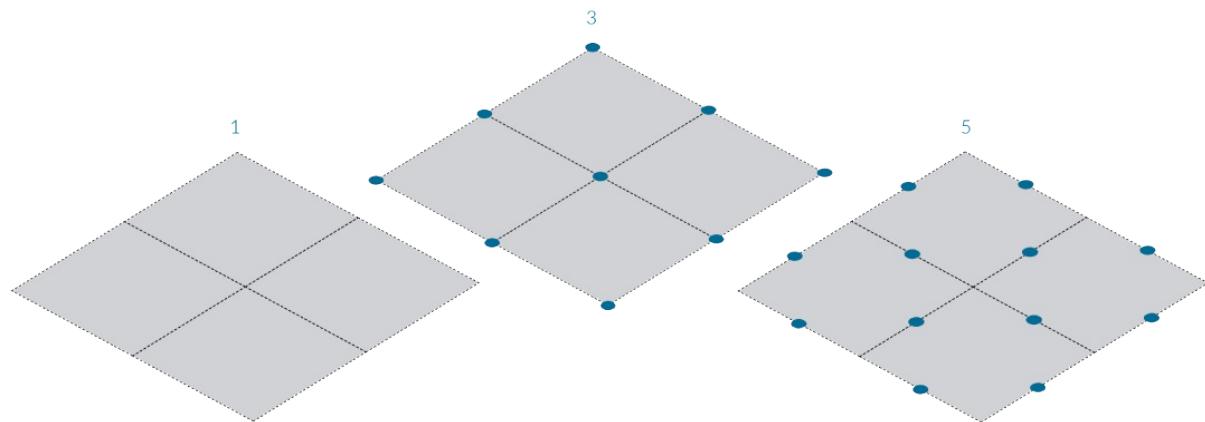
В этом разделе мы проработаем простой пример, в качестве введения в работу с инструментами Element. Мы изучим некоторые паттерны и обработку фасадов в области архитектуры, где мы подключим структуру данных Half Edge вместе с базовыми компонентами Element без использования характеристик per vertex.



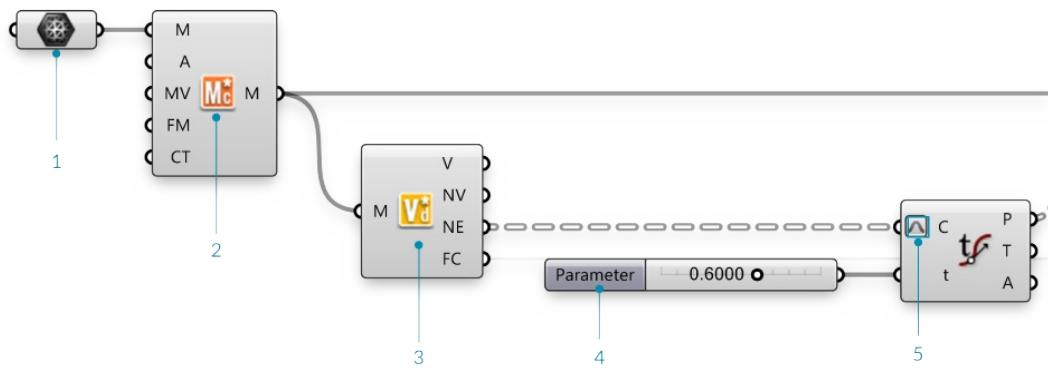
2.1.5.1 Пример 1

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

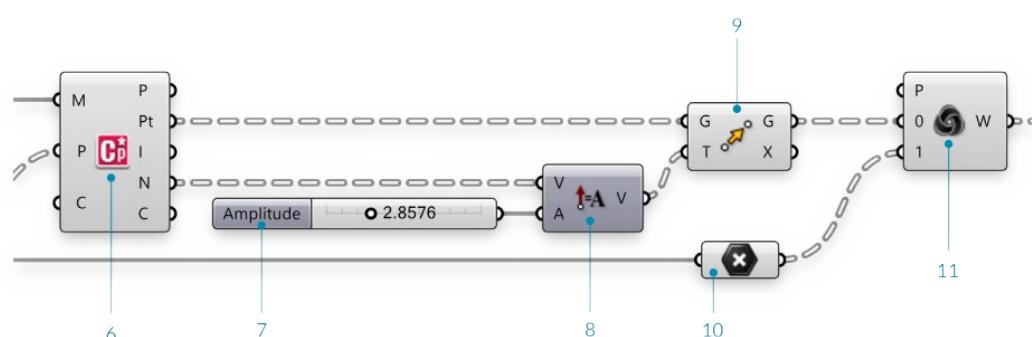


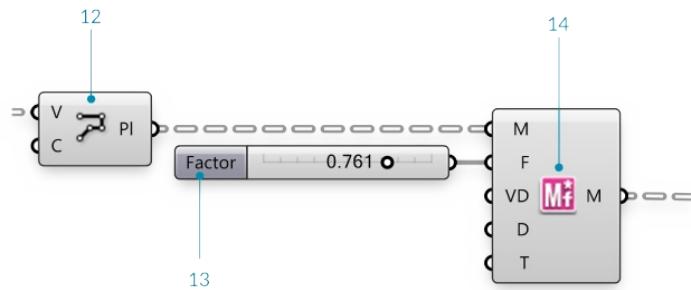


00.	Создайте плоскость mesh в Rhino с XFaces = 2 и YFaces = 2 и запустите новое определение, набрав Ctrl-N (в Grasshopper)	
01.	Зайдите в Params/Geometry/Mesh - вытащите контейнер Mesh на холст	
01b.	Чтобы привязать mesh в Rhino, кликните правой клавишей мыши по компоненту Mesh и выберите "Set one Mesh". Мы будем использовать простую плоскость mesh в ходе работы с этим определением, но вы можете заменить эту mesh вашей собственной	
02.	Зайдите в Element*/Utility/Mesh Combine and Clean - перетащите компонент Element* Mesh Combine and Clean на холст	
03.	Зайдите в Element*/Data/Vertex Neighbors - перетащите компонент Element* Vertex Neighbors на холст	
04.	Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст и установите следующие значения: Lower Limit: 0.0000 Upper Limit: 1.0000	
05.	Зайдите в Curve/Analysis/Evaluate Curve - перетащите компонент Evaluate Curve на холст	
05b.	Соедините выход Neighbouring Edges (NE) компонента Element* Vertex Neighbors с входом Curve (C) компонента Evaluate Curve	
05c.	Соедините слайдер Number Slider с входом Float (t) компонента Evaluate Curve и установите значение на 0.5000	
05d.	Кликните правой клавишей по входу Curve (C) компонента Evaluate Curve и выберите Reparameterize	

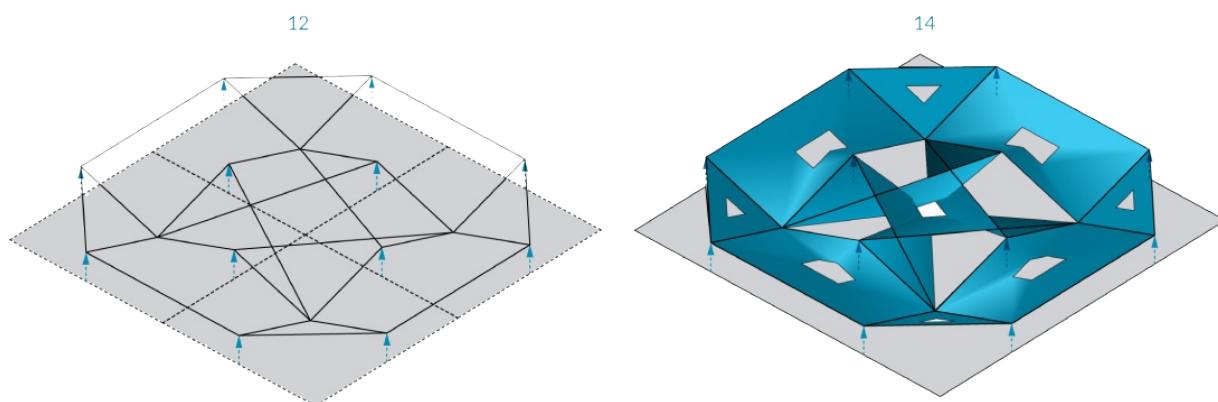


06.	Зайдите в Element*/Analyse/Mesh Closest Point - перетащите компонент Element* Mesh Closest Point на холст	
06a.	Соедините выход Mesh (M) компонента Element*/Utility/Mesh Combine and Clean с входом Mesh (M) компонента Element* Mesh Closest Point	
06b.	Соедините выход Points (P) компонента Curve/Analysis/Evaluate Curve с входом Point (P) компонента Element* Mesh Closest Point	
07.	Зайдите в Params/Input/Number Slider - вытащите компонент Number Slider на холст и установите следующие значения: Rounding: Float Lower Limit:0 Upper Limit: 10.000	
08.	Зайдите в Vector/Vector/Amplitude - перетащите компонент Amplitude на холст	
09.	Зайдите в Transform/Euclidean/Move - перетащите компонент Move на холст	
10.	Зайдите в Params/Geometry/Point - перетащите компонент Point на холст	
10b.	Соедините выход Face Centers (FC) компонента Element* Vertex Neighbors с компонентом Point	
11.	Зайдите в Sets/List/Weave - перетащите компонент Weave на холст	



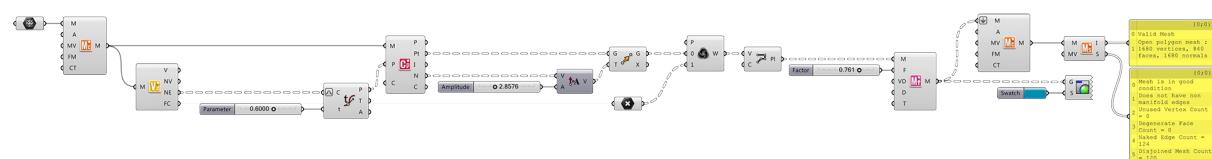
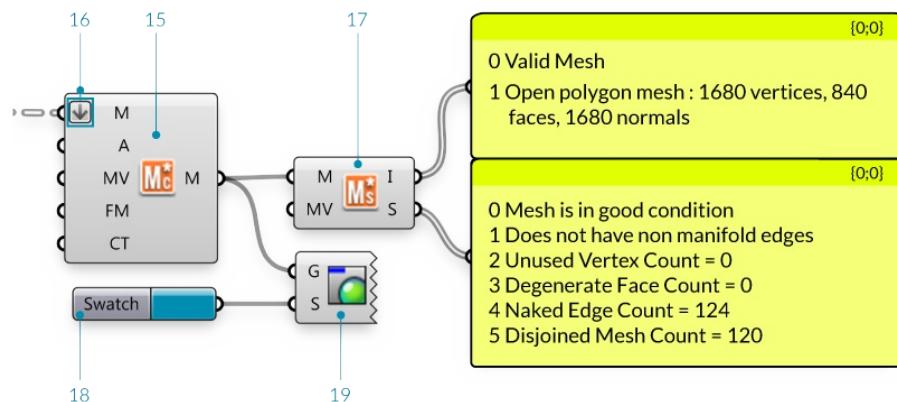


12.	Зайдите в Curve/Primitive/Polyline - перетащите компонент Polyline на холст	
12b.	Соедините выход Weave (W) компонента Weave с входом Vertices (V) компонента Polyline	
12c.	Кликните правой клавишей мыши по входу Closed (C) компонента Polyline , кликните по "Set Boolean" и выберите значение True Так мы создали закрытую полилинию.	
13.	Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст. Мы будем использовать по умолчанию диапазон от 0 до 1 для этого слайдера.	
14.	Зайдите в Element*/Transform/Mesh Frame - перетащите компонент Element* Mesh Frame на холст.	
14b.	Соедините выход Polyline (PI) компонента Polyline с входом Geometry (G) компонента Mesh Frame Заметьте, что компонент **Mesh Frame** может принимать в качестве входа как mesh, так и список закрытых полилиний кривых	
14c.	Подключите слайдер Number Slider к входу Factor (F) компонента Mesh Frame	



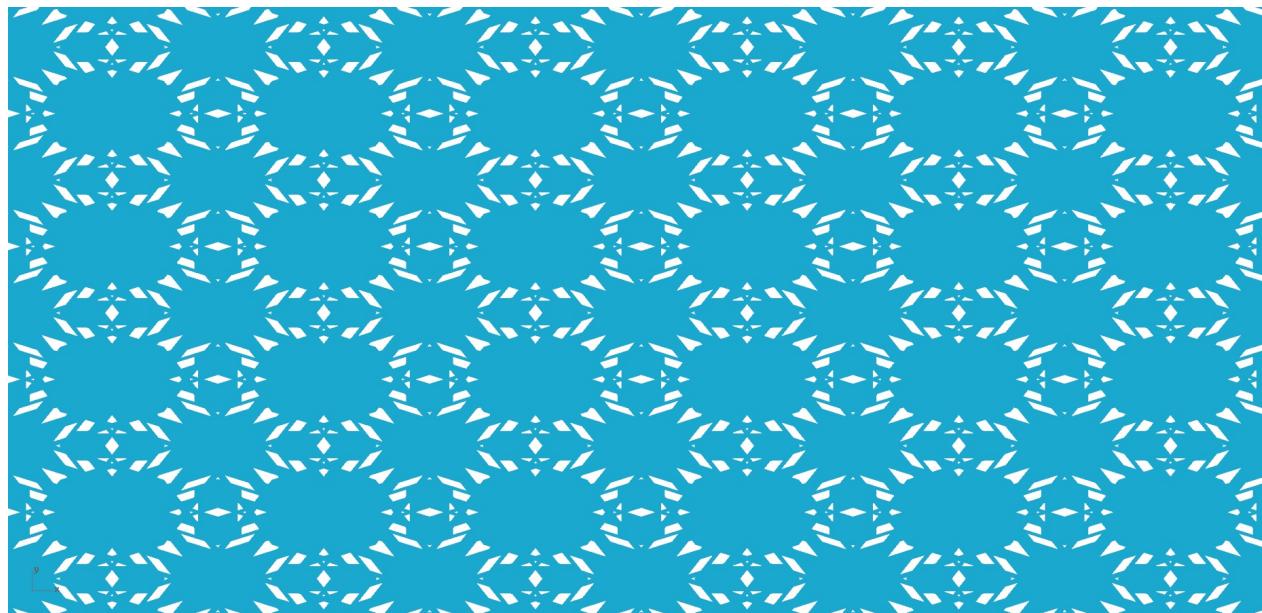
15.	Зайдите в Element*/Utility/Mesh Combine and Clean - перетащите компонент Element* Mesh Combine and Clean на холст	
	Кликните правой клавишей мыши по входу Combine Type (CT) компонента Element* Mesh Combine and Clean , кликните по "Set Integer" и установите значение 1.	

15b.	Вход Combine Type имеет две опции (0, которая объединяет и очищает mesh) и (1, которая соединяет mesh в список и не смешивает вершины). В этом примере мы хотим соединить mesh
16.	Кликните правой клавишей мыши по входу Mesh (M) компонента Element* Mesh Combine and Clean и выберите "Flatten". После этого мы сможем соединить список mesh.
17.	Зайдите в Element*/Utility/Mesh Status - перетащите компонент Element* Mesh Status на холст
17b	Соедините выходы Info (I) и Status (S) компонента Element* Mesh Status с компонентом Params/Input/Panel Выход mesh **Info** содержит информацию о правильности mesh, закрытом или открытом типе и о количестве компонентов mesh (вершины, полигоны, нормали). **Status** mesh информирует пользователя о том, в "хорошем" ли состоянии mesh, а также передает данные о неоднородных ребрах, количестве неиспользованных вершин, количестве неправильных полигонов, количестве naked ребер и количестве необъединенных mesh.
18.	Зайдите в Params/Input/Colour Swatch - перетащите компонент Colour Swatch на холст
19.	Зайдите в Display/Preview/Custom Preview - перетащите компонент Custom Preview на холст

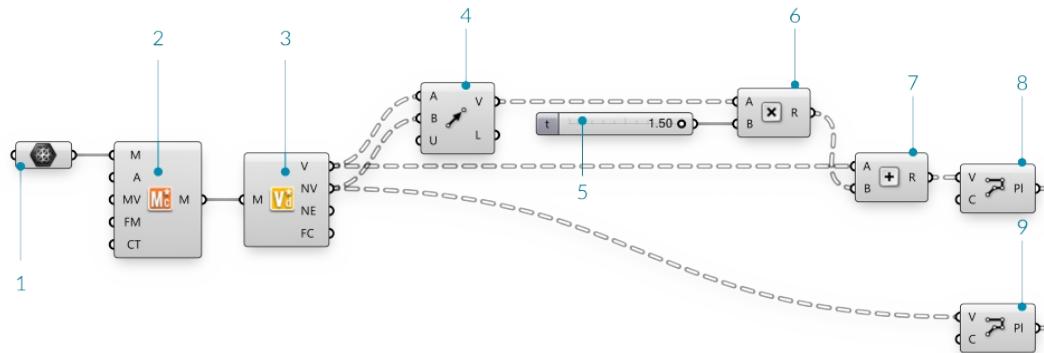


2.1.5.2 Пример 2

Файлы упражнения, которые сопровождают этот раздел: http://grasshopperprimer.com/appendix/A-2/1_gh-files.html

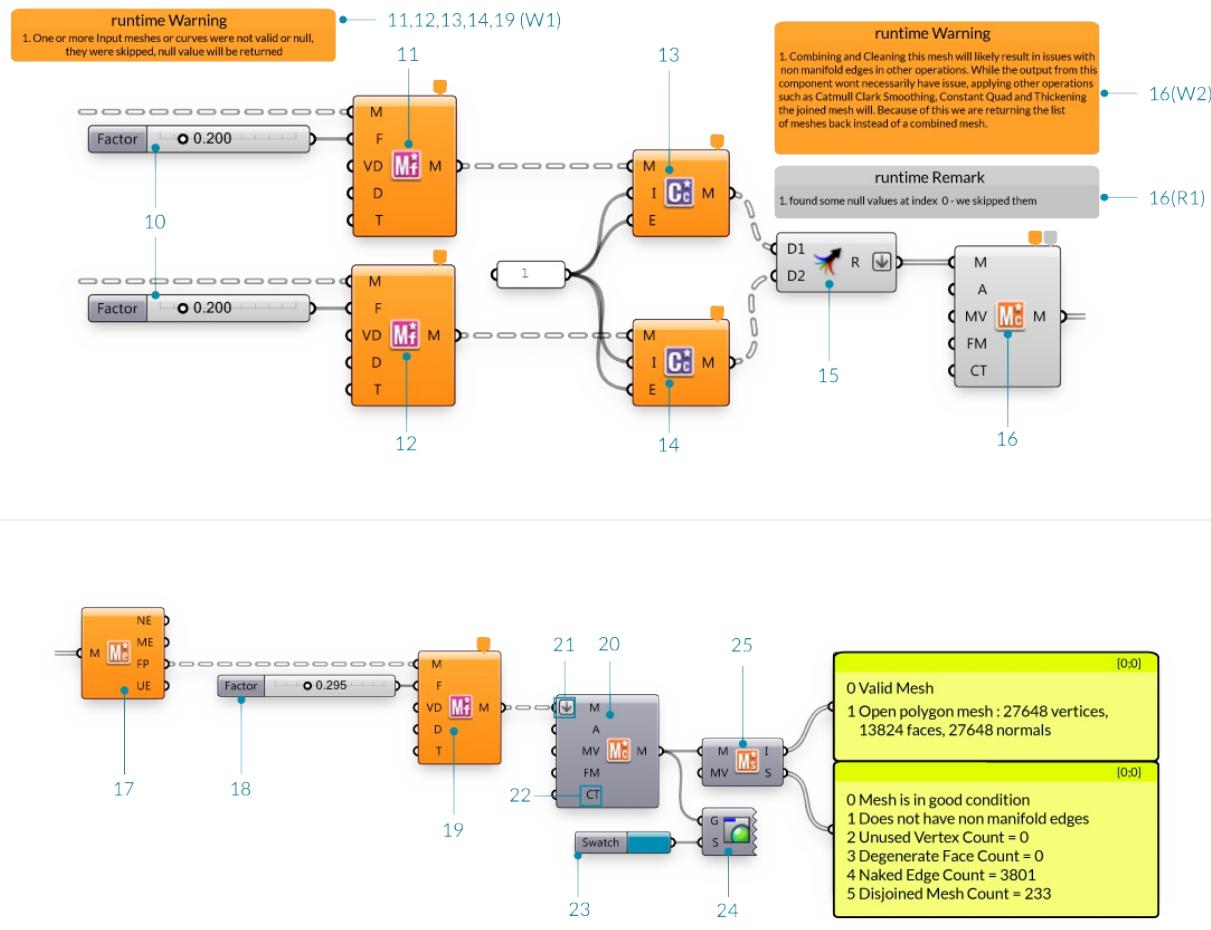


00.	Создайте плоскость mesh в Rhino с XFaces = 2 и YFaces = 2 и запустите новое определение, набрав Ctrl-N (в Grasshopper)	
01.	Зайдите в Params/Geometry/Mesh - перетащите компонент Mesh на холст	
01b.	Чтобы привязать mesh в Rhino, кликните правой клавишей мыши по компоненту Mesh и выберите "Set one Mesh". Мы будем использовать простую плоскость mesh в ходе работы с этим определением, но вы можете заменить эту mesh вашей собственной	
02.	Зайдите в Element*/Utility/Mesh Combine and Clean - перетащите компонент Element* Mesh Combine and Clean на холст	
03.	Зайдите в Element*/Data/Vertex Neighbors - перетащите компонент Element* Vertex Neighbors на холст	
04.	Зайдите в Vector/Vector/Vector2Pt - перетащите компонент Vector2Pt на холст	
05.	Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст и установите следующие значения: Rounding: Float Lower Limit:0 Upper Limit: 2.000	
06.	Зайдите в Maths/Operator/Multiplication - перетащите компонент Multiplication на холст	
07.	Зайдите в Maths/Operators/Addition - перетащите два компонента Addition на холст	
08.	Зайдите в Curve/Primitive/Polyline - перетащите компонент Polyline на холст	
09.	Зайдите в Curve/Primitive/Polyline - перетащите компонент Polyline на холст	



	<p>Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст и установите следующие значения:</p> <p>10. Rounding: Float Lower Limit:0 Upper Limit: 1.000</p>	
11,12.	<p>Зайдите в Element*/Transform/Mesh Frame - перетащите компонент Element* Mesh Frame на холст.</p>	
11b,12b.	<p>Соедините выход Polyline (Pl) компонента Polyline с входом Geometry (G) компонента Mesh Frame</p> <p>Заметьте, что компонент Mesh Frame может принимать в качестве входа как mesh, так и список закрытых полилиний кривых</p>	
11c,12c.	<p>Подключите Number Slider (10) к входу Factor (F) компонента Mesh Frame</p>	
13,14.	<p>Зайдите в Element*/Subdivide/Catmull Clark Subdivision - перетащите компонент Catmull Clark Subdivision на холст</p> <p>Мы установим значение входа Iterations (I) на 1, также как и значение входа Edge Condition (E) на 1. Опции входа edge condition следующие 0 = Общая фиксация, 1 == Сглаживание, 2 == Фиксация Углов.</p>	
15.	<p>Зайдите в Sets/Tree/Merge - вытащите два компонента Merge на холст</p>	
15b.	<p>Right click the Result (R) output of the Merge component and click "Flatten".</p>	
16.	<p>Зайдите в Element*/Utility/Mesh Combine and Clean - перетащите компонент Element* Mesh Combine and Clean на холст</p>	

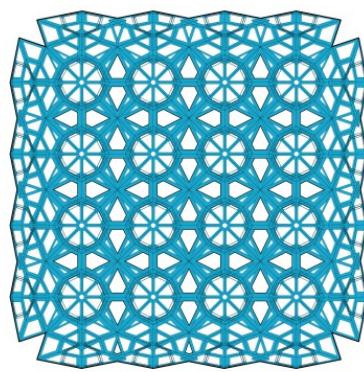
У компонентов имеются подробные замечания и предупреждения, информирующие пользователя о текущих или возможных проблемах, которые могут возникнуть после итерации с другими компонентами. В некоторых случаях вы можете использовать компонент **Element Combine and Clean** чтобы объединить одинаковые вершины на mesh, что может привести к неоднородным ребрам, если в дальнейшем mesh будет утолщена. Компонент **Element Combine and Clean** проинформирует вас об этом моменте и о том, что он вернет список обратно. У вас есть возможность установить **Combine Type** на значение 1, которое объединит mesh в список, но не объединит идентичные вершины.



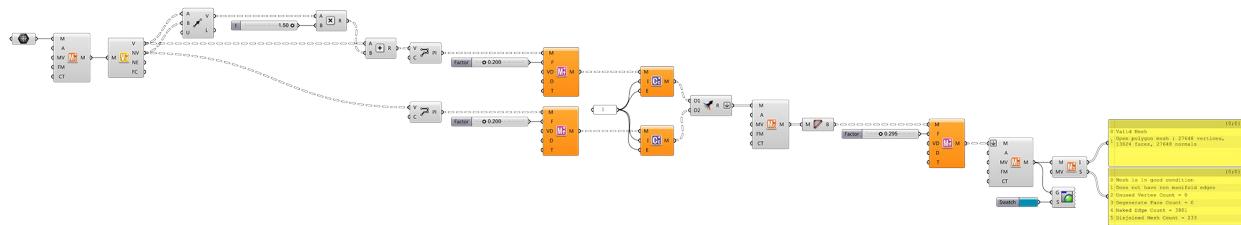
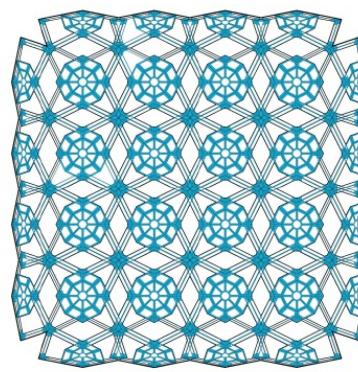
17.	Зайдите в Element*/Utility/Mesh Edges - перетащите компонент Element* Mesh Edges на холст	
17b	Соедините выход Mesh (M) компонента Element* Mesh Combine and Clean (16) с входом Mesh (M) компонента Element* Mesh Edges	
18.	Зайдите в Params/Input/Number Slider - перетащите компонент Number Slider на холст и установите следующие значения: Rounding: Float Lower Limit:0 Upper Limit: 1.000	
19.	Зайдите в Element*/Transform/Mesh Frame - перетащите компонент Element* Mesh Frame на холст.	
19b	Соедините выход Face Polylines (FP) компонента Element* Mesh Edges с входом Mesh (M) компонента Element* Mesh Frame	
19c	Подключите Number Slider к входу Float (f) компонента Element* Mesh Frame	
20.	Зайдите в Element*/Utility/Mesh Combine and Clean - перетащите компонент Element* Mesh Combine and Clean на холст	
21.	Кликните правой клавишей мыши по входу Mesh (M) компонента Element* Mesh Combine and Clean и выберите "Flatten".	
	Кликните правой клавишей мыши по входу Combine Type (CT) компонента Element*	

	Mesh Combine and Clean , кликните по "Set Integer" и установите значение 1.	
22.	Вход Combine Type имеет две опции (0, которая объединяет и очищает mesh) и (1, которая соединяет mesh в список, но не совмещает вершины). В этом примере мы хотим соединить несколько mesh	
23.	Зайдите в Params/Input/Colour Swatch - перетащите компонент Colour Swatch на холст	
24.	Зайдите в Display/Preview/Custom Preview - перетащите компонент Custom Preview на холст	
25.	Зайдите в Element*/Utility/Mesh Status - перетащите компонент Element* Mesh Status на холст	
25b	Соедините выходы Info (I) и Status (S) компонента Element* Mesh Status с компонентом Params/Input/Panel Выход mesh **Info** содержит информацию о правильности mesh, закрытом или открытом типе и о количестве компонентов mesh (вершины, полигоны, нормали). **Status** mesh информирует пользователя о том, в "хорошем" ли состоянии Mesh, а также передает данные о неоднородных ребрах, количестве неиспользованных вершин, количестве неправильных полигонов, количестве naked ребер и количестве необъединенных mesh.	

20

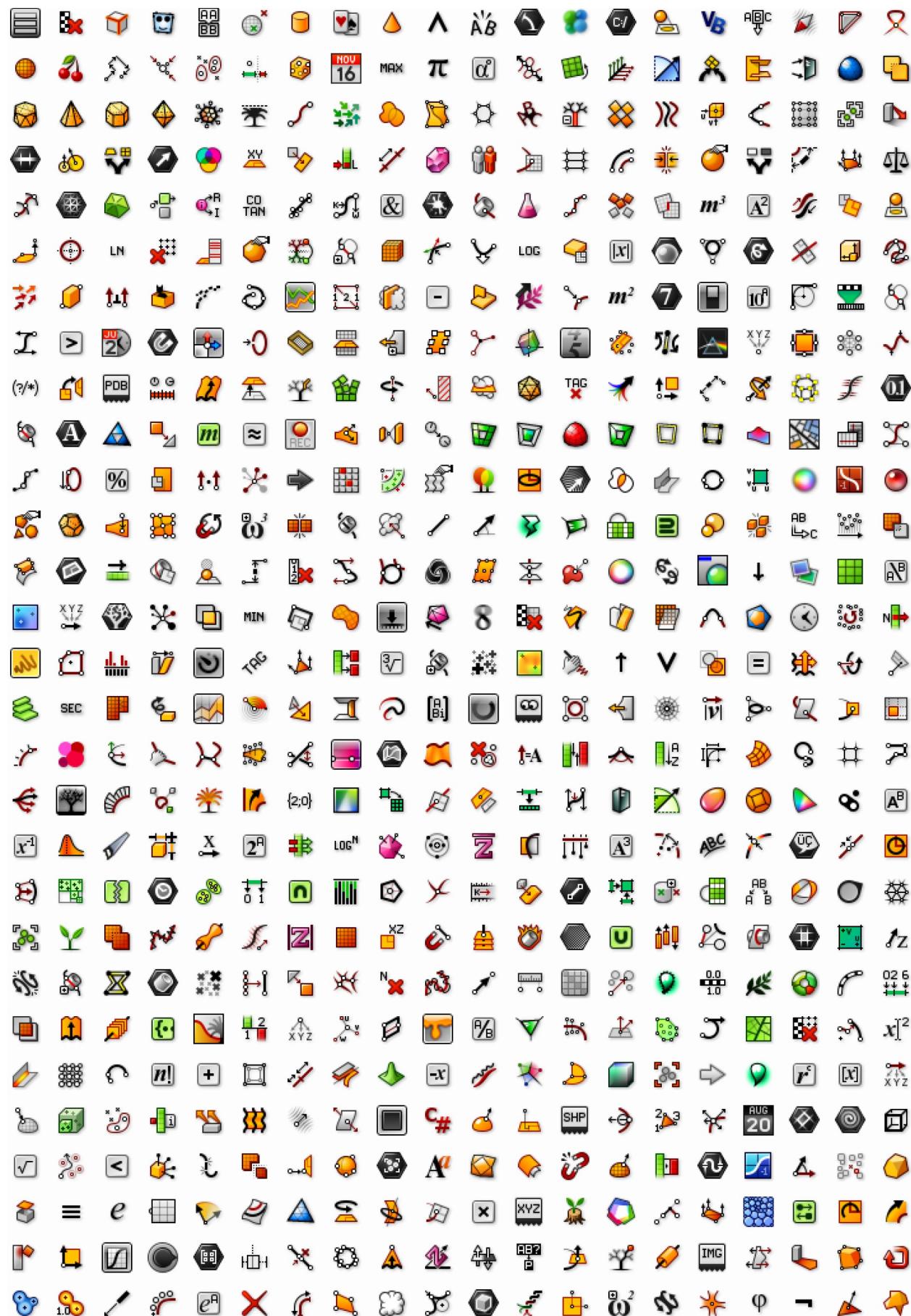


20 (varied values)



Приложение

Следующий раздел содержит полезные ссылки, включая список всех компонентов, использованных в этом пособии, а также дополнительные ресурсы для изучения Grasshopper.



2.1. Индекс

Этот индекс содержит дополнительную информацию по всем компонентам, использованным в этом пособии, а также другие компоненты, которые могут оказаться вам полезными. Это небольшое введение в более чем 500 компонентов в плагине Grasshopper.

Параметры

Geometry

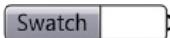
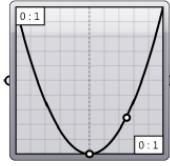
P.G.Crv	Curve Parameter Представляет набор геометрии Curve (кривая). Геометрия Curve - это общий знаменатель всех типов кривых в Grasshopper.	
P.G.Circle	Circle Parameter Представляет набор примитивов Circle (круг).	
P.G.Geo	Geometry Parameter Представляет набор 3D геометрии.	
P.G.Pipeline	Geometry Pipeline Определяет геометрию pipeline из Rhino в Grasshopper.	
P.G.Pt	Point Parameter Параметр Point (точка) способен хранить постоянные данные. Вы можете настроить запись постоянных данных через меню Parameter.	
P.G.Srf	Surface Parameter Представляет набор геометрии Surface (поверхность). Surface геометрия - это общий знаменатель всех типов поверхности в Grasshopper.	

Primitive

P.P.Bool	Boolean Parameter Представляет набор Булевых значений (Правда/Ложь).	
P.P.D	Domain Parameter Представляет набор одно-пространственных Диапазонов. Диапазоны обычно используются для представления фрагментов кривых и длительных числовых диапазонов. Диапазон состоит из двух чисел, которые показывают ограничения диапазона, все, что между этими числами является частью диапазона.	
P.P.D2	Domain2 Parameter Содержит набор двух-пространственных диапазонов. 2D диапазоны обычно используются для представления фрагментов поверхности. Двух-пространственный диапазон состоит из двух одно-пространственных диапазонов.	
P.P.ID	Guid Parameter Представляет набор Globally Unique Identifiers (глобально уникальных идентификаторов). Параметры Guid способны	

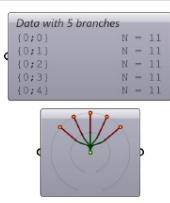
	хранить постоянные данные. Вы можете настроить запись постоянных данных через меню Parameter.	
P.P.Int	Integer Parameter Представляет набор Integer (целых) численных значений. Параметры Integers способны хранить постоянные данные. Вы можете настроить запись постоянных данных через меню Parameter.	
P.P.Num	Number Parameter Представляет набор численных значений с плавающей точкой. Параметры Number способны хранить постоянные данные. Вы можете настроить запись постоянных данных через меню Parameter.	
P.P.Path	File Path Содержит набор путей файла.	

Input

P.I.Toggle	Boolean Toggle Переключатель булевых значений (правда/ложь).	
P.I.Button	Button Кнопка с двумя значениями. При нажатии, кнопка возвращает значение true (правда) и затем сбрасывает значение на false (ложь).	
P.I.Swatch	Color Swatch Swatch (образец) - это особый объект интерфейса, который позволяет быстро настроить индивидуальные цветовые значения. Вы можете поменять цвет образца через контекстное меню.	
P.I.Grad	Gradient Control Gradient control (контроль градиента) позволяет определить градиент цвета внутри числового диапазона. По умолчанию используется диапазон (от 0.0 до 1.0), но это можно настроить через параметры ввода L0 и L1. Также вы можете добавить цветовые ползунки к градиенту перетаскивая из верхнего левого угла цветовое колесико и настраивая цветовые ползунки кликая правой клавишей мыши.	
P.I.Graph	Graph Mapper Graph Mapper позволяет вам перенести набор чисел. По умолчанию диапазоны {x} и {y} графической функции - это диапазон секции (0.0 ~ 1.0), но это также можно настроить через Graph Editor. Graph Mapper может содержать одиночную функцию переноса, которую можно выбрать через контекстное меню. Графы обычно имеют ползунки (небольшие круги), которые можно использовать для изменения переменных, которые определяют графическое уравнение. По умолчанию, объекты graph mapper не содержат графиков и выполняют перенос значений 1:1.	
P.I.Slider	Number Slider Slider (слайдер) - это особый объект интерфейса, который позволяет быстро настроить индивидуальные числовые значения. Вы можете менять значения и характеристики через меню или двойным кликом на поверхности слайдера. Слайдеры можно удлинить или укоротить перетаскивая самое крайнее правое ребро влево или вправо. Заметьте, что слайдеры имеют только вход (т.е. не имеют выхода).	

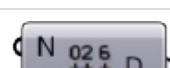
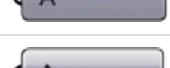
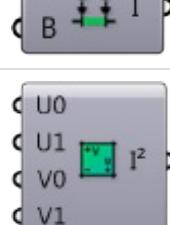
	<p>Panel (панель) для разных заметок и текстовых значений. Обычно, это неактивный объект, который позволяет вам добавлять комментарии или объяснения к Документу. Панели могут получать информацию откуда угодно. Если вы подключите параметр выхода в панель, вы увидите содержание этого параметра в реальном времени. Все данные в Grasshopper можно просматривать таким образом. Панели также могут передавать их содержание в текстовый файл.</p>	 <p>Double click to edit panel content...</p>
P.List	<p>Value List Предоставляет список преднастроенных значений из которых можно выбрать.</p>	 <p>One ▼</p>

Utilities

P.U.Cin	Cluster Input Представляет параметр Cluster Input.	
P.U.COut	Cluster Output Представляет параметр Cluster Output.	
P.U.Dam	Data Dam Задерживает данные на их пути по всему документу.	
P.U.Jump	Jump Прыжок между различными местами.	
P.U.Viewer	Param Viewer Просмотрщик структуры данных.	
P.U.Scribble	Scribble Быстрая заметка.	Doubleclick Me!

Maths

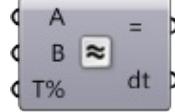
Domain

M.D.Bnd	Bounds Создает числовой диапазон, который содержит список чисел.	
M.D.Consec	Consecutive Domains Создает последовательные диапазоны из списка чисел.	
M.D.Dom	Construct Domain Создает числовой диапазон из двух числовых экстремум.	
M.D.Dom2Num	Construct Domain ² Создает двухпространственный диапазон из четырех чисел.	

M.D.DeDomain	Deconstruct Domain Разрушает числовой диапазон на его компоненты.	
M.D.DeDom2Num	Deconstruct Domain ² Разрушает двух-пространственный диапазон на четыре числа.	
M.D.Divide	Divide Domain ² Разделяет двух-пространственный диапазон на равные сегменты.	
M.D.Inc	Includes Проверяет числовое значение, включено ли оно в диапазон.	
M.D.ReMap	Remap Numbers Переносит числа в новый числовой диапазон.	

Operators

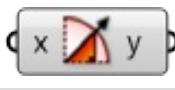
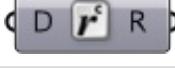
M.O.Add	Addition Математическое добавление.	
M.O.Div	Division Математическое разделение.	
M.O.Equals	Equality Проверяет на (не)равенство двух чисел.	
M.O.And	Gate And Выполняет соединение булевых значений (AND gate). Оба входа должны быть True, для того чтобы результат был True.	
M.O.Not	Gate Not Выполняет отрицание булевых значений (NOT gate).	
M.O.Or	Gate Or Выполняет разъединение булевых значений (OR gate). Только один вход должен быть True, для того чтобы результат был True.	
M.O.Larger	Larger Than Больше чем (или равно).	
M.O.Multiply	Multiplication	

M.O.Similar	Similarity Проверяет на сходство двух чисел.	
M.O.Sub	Subtraction Математическое извлечение.	

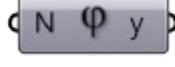
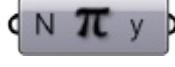
Script

M.S.Eval	Evaluate Определяет выражение с гибкими числами переменных.	
M.S.Expression	Expression Определяет выражение.	

Trig

M.T.Cos	Cosine Вычисляет косинус значения.	
M.T.Deg	Degrees Конвертирует угол, указанный в радианах, в градусы.	
M.T.Rad	Radians Конвертирует угол, указанный в градусах, в радианы.	
M.T.Sin	Sine Вычисляет синус значения.	

Utilities

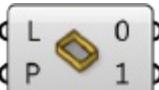
M.U.Avg	Average Проверяет среднее арифметическое для набора элементов.	
M.U.Phi	Golden Ratio Выдает множество золотого сечения (Phi).	
M.U.Pi	Pi Выдает множество Pi.	

Sets

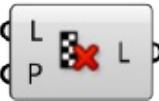
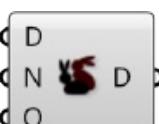
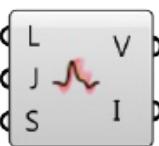
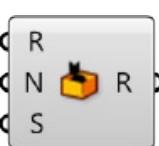
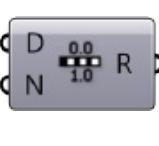
List

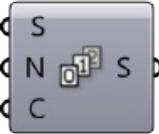
S.L.Combine	Combine Data Сочетает неизвестные значения из нескольких входов.	
S.L.CrossRef	Cross Reference	

S.L.Combine	Combine Data Сочетает не-известные значения из нескольких входов.	
S.L.CrossRef	Cross Reference Производит кросс-референс данных из множества списков.	
S.L.Dispatch	Dispatch Распаковывает элементы списка в два целевых списка. List Dispatching очень похоже на компонент [Cull Pattern], за исключением того, что оба списка предстаиваются как выходы.	
S.L.Ins	Insert Items Вставляет коллекцию элементов в список.	
S.L.Item	List Item Извлекает особые элементы из списка.	
S.L.Lng	List Length Измеряет длину списка. Элементы списка идентифицируются по их индексу. Первый элемент сохраняется под индексом ноль, второй элемент под индексом один и так далее. Самый большой возможный индекс в списке равняется длине списка минус один.	
S.L.Long	Longest List Выращивает набор списков по самому длинному среди них.	
S.L.Split	Split List Разделяет лист на отдельные части.	
S.L.Replace	Replace Items Заменяет определенные элементы в списке.	
S.L.Rev	Reverse List Разворачивает порядок списка. Новый индекс каждого элемента будет $N-i$, где N - наивысший индекс в списке и i - старый индекс элемента.	
S.L.Shift	Shift List Смещает все элементы в списке. Элементы в списке смещаются (перемещаются) к концу списка, если shift смещение имеет положительное значение. Если Wrap равняется True, тогда те элементы, которые выпадают за края, присоединяются заново.	
S.L.Short	Shortest List Сжимает коллекцию списков по длине самого короткого из	

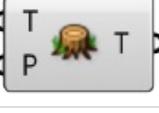
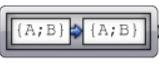
	Sift элементы в списке используют повторяющийся паттерн индексов.	
S.L.Sort	Sort List Сортирует список числовых ключей. Чтобы что-то отсортировать, это должно быть сначала сопоставлено. Большинство типов данных не может быть сопоставлена, Numbers и Strings - единственное исключение. Если вы хотите отсортировать другие типы данных, такие как кривые, сначала вам потребуется создать список ключей.	
S.L.Weave	Weave Соединяет набор вводных данных используя определенный шаблон. Шаблон определяется как список индексов значений (целых), которые определяют порядок, в котором вводные данные собираются.	

Sets

S.S.Culli	Cull Index Cull (удаляет) проиндексированные элементы в списке.	
S.S.Cull	Cull Pattern Cull (удаляет) элементы в списке, используя повторяющуюся битовую маску. Битовая маска определяется как список булевых значений. Битовая маска повторяется, пока все элементы из списка данных не будут определены.	
S.S.Dup	Duplicate Data Дублировать данные предопределенное количество раз. Данные могут дублироваться двумя способами, либо копии списка прикрепляются в конце до тех пор, пока не будет достигнуто некое количество копий, либо каждый элемент дублируется некоторое число раз, прежде чем перейти к следующему элементу.	
S.S.Jitter	Jitter Случайным образом перемешивает список значений. Вводный список реорганизуется на основе флюктуационного шума. Jittering (создание быстрых флюктуаций) - это хороший способ получения случайного набора с хорошим распределением. Параметры jitter (создания быстрых флюктуаций) настраивают радиус флюктуационного шума. Если jitter равняется 0.5, тогда каждому элементу разрешено переместиться случайным образом в пределах половины ширины всего набора.	
S.S.Random	Random Генерируется список псевдо-случайных чисел. Числовая последовательность уникальна, но стабильная для каждого значения seed. Если вам не нравится случайное распределение, попробуйте другие значения seed.	
S.S.Range	Range Создает диапазон чисел. Числа распределяются равномерно внутри числового диапазона. Используйте этот компонент, если вам необходимо создать числа между экстремумами. Если вы хотите контролировать интервалы между последовательными числами, вам следует использовать компонент [Series].	

	<p>внутри числового диапазона. Используйте этот компонент, если вам необходимо создать числа между экстремумами. Если вы хотите контролировать интервалы между последовательными числами, вам следует использовать компонент [Series].</p>	
S.S.Repeat	<p>Repeat Data Повторяет паттерн до тех пор, пока не достигнет определенной длины.</p>	
S.S.Series	<p>Series Создает последовательность чисел. Числа распределены в соответствии со значением {Step}. Если вам необходимо распределить числа внутри фиксированного числового порядка, то используйте компонент [Range].</p>	

Tree

S.T.Explode	<p>Explode Tree Извлекает все ветки из дерева.</p>	
S.T.Flatten	<p>Flatten Tree Сглаживает дерево данных путем удаления всей информации о ветках.</p>	
S.T.Flip	<p>Flip Matrix Переворот дерева данных типа матрицы меняя местами ряды и колонны.</p>	
S.T.Graft	<p>Graft Tree Обычно, элементы данных хранятся в ветках в определенных индекс значениях (0 - первый элемент, 1 - второй элемент и т.д.) и ветки хранятся в деревьях в определенных путях веток, например: {0;1}, который отображает вторую под-ветку первой основной ветки. Grafting создает новую ветку для каждого отдельного элемента данных.</p>	
S.T.Merge	<p>Merge Смешивает множество потоков данных.</p>	
S.T.Path	<p>Path Mapper Выполняет лексические операции с деревьями данных. Лексические операции - это логические переносы между путями данных и индексов, которые определяются текстовыми (лексическими) масками и паттернами.</p>	
S.T.Prune	<p>Prune Tree Удаляет все ветки у дерева, которое содержит определенное число элементов данных. Вы можете установить обе границы, нижнюю и верхнюю, при сокращении веток.</p>	
S.T.Simplify	<p>Simplify Tree Упрощает дерево, удаляя накладки среди всех веток.</p>	
S.T.TStat	<p>Tree Statistics Можно получить некоторую статистику касательно дерева данных.</p>	

Vector

Grid

V.G.HexGrid	Hexagonal 2D сетка с шестиугольными ячейками.	
V.G.RecGrid	Rectangular 2D сетка с прямоугольными ячейками.	
V.G.SqGrid	Square 2D сетка с квадратными ячейками	

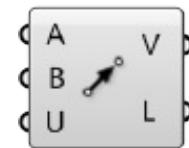
Point

V.P.Pt	Construct Point Построение точки из координат {xyz}.	
V.P.pDecon	Deconstruct Разбирает точку на ее компоненты.	
V.P.Dist	Distance Вычисляет Евклидово расстояние между координатами двух точек.	

Vector

V.V.X	Unit X Unit вектор параллелен мировой оси {x}.	
V.V.Y	Unit Y Unit вектор параллелен мировой оси {y}.	
V.V.Vec2Pt	Vector 2Pt Создает вектор между двумя точками.	

Curve



Curve

Analysis

CACP	Control Points Извлекает контрольные точки nurbs и узлы кривой.	
------	--	--

Division

CD.Divide	Divide Curve Разделяет кривую на равные по длине сегменты.	
-----------	---	--

Primitive

CPCir	Circle Создает круг, определяемый основной плоскостью и радиусом.	
CPCir3Pt	Circle 3Pt Создает круг, определяемый тремя точками.	
CPCirCNR	Circle CNR Создает круг, определяемый центром, нормалью и радиусом.	
CPLine	Line SDL Создает линейный сегмент, определяемый начальной точкой, тангенсом и длиной.	
CPPolygon	Polygon Создает полигон с дополнительными круглыми ребрами.	

Spline

C.S.IntCrv	Interpolate	
------------	-------------	--

C.S.Nurbs	Nurbs Curve Строит кривую nurbs из контрольных точек.	
C.S.PLine	PolyLine Создает полилинию, соединяющую некоторое число точек.	

Util

C.U.Explode	Explode Разбивает кривую на маленькие сегменты.	
<C.U.Join	Join Curves Соединяет так много кривых как возможно.	
C.U.Offset	Offset Смещает кривую на определенную дистанцию.	

Surface

Analysis

S.A.DeBrep	Deconstruct Brep Разбирает brep на ее составляющие.	
------------	--	--

Freeform

S.F.Boundary	Boundary Surfaces Создает плоскую поверхность из коллекции граничных ребер кривых.	
S.F.Extr	Extrude Вытесняет кривые и поверхности вдоль вектора.	
S.F.ExtrPt	Extrude Point Вытесняет кривые и поверхности к точке.	
S.F.Loft	Loft Создает выпуклую поверхность через набор секций кривых.	

S.F.ExtrPt	Extrude Point Вытесняет кривые и поверхности к точке.	
S.F.Loft	Loft Создает выпуклую поверхность через набор секций кривых.	
S.F.RevSrf	Revolution Создает поверхность вращения.	
S.F.Swp2	Sweep2 Создает загнутую поверхность с двумя rail кривыми.	

Primitive

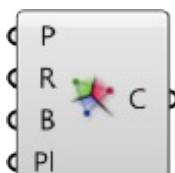
S.P.BBox	Bounding Box Solve ориентированная геометрия граничных прямоугольников.	 Per Object
----------	--	--

Util

S.U.SDivide	Divide Surface Генерирует сетку точек {uv} на поверхности.	
S.U.SubSrf	Isotrim Извлекает изопараметрическое подмножество поверхности.	

Mesh

Triangulation

M.T.Voronoi	Voronoi Плоскостная диаграмма Вороного для коллекции точек.	
-------------	--	---

Transform

Affine



Array

T.A.Array	Linear Array Создает линейный массив геометрии.	
-----------	--	--

Morph

T.M.Morph	Box Morph Превращение объекта в скрученную коробку.	
T.M.SBox	Surface Box Создает скрученную коробку на участке поверхности.	

Display

Color

D.C.HSL	Colour HSL Создает цвет из плавающих точек {HSL} каналов.	
---------	--	--

Dimensions

D.D.Tag	Text tags Компонент текстовый тэг позволяет рисовать небольшие Strings в видовом окне как элементы обратной связи. Текст и расположение определяются как параметры ввода. Когда текстовые тэги запекаются, они превращаются в Text Dots (текстовые точки).	
D.D.Tag3D	Text Tag 3D Представляет список 3D текстовых тэгов в видовом окне Rhino	



Preview

D.P.Preview	Custom Preview Позволяет настраивать предпросмотр геометрии.	
-------------	---	--

Vector

D.V.Points	Point List Отображает подробные данные о списках точек.	
------------	--	--

2.2. Файлы с примерами Grasshopper

Эти файлы с примерами сопровождают Пособие по Grasshopper и расположены в соответствии с разделами.

1.2.	
	1.2.5_the grasshopper definition.gh

1.3.	
	1.3.2.1_attractor definition.gh
	1.3.3_operators and conditionals.gh
	1.3.3.4_trigonometry components.gh
	1.3.3.5_expressions.gh
	1.3.4_domains and color.gh
	1.3.5_booleans and logical operators.gh

1.4.	
	1.4.1.2_grasshopper spline components.gh
	1.4.3_data matching.gh
	1.4.4_list creation.gh
	1.4.5_list visualization.gh
	1.4.6_list management.gh
	1.4.7_working with lists.gh

1.5.	
	1.5.1.3_morphing definition.gh
	1.5.2.1_Data Tree Visualization.gh
	1.5.3_working with data trees.gh
	1.5.3.6_weaving definition.gh
	1.5.4_rail intersect definition.gh

1.6.	
	1.6.1_what is a mesh.gh
	1.6.3_creating meshes.gh
	1.6.6_working with meshes.gh

2.3. Ресурсы

Существует множество ресурсов доступных для дальнейшего изучения Grasshopper и концептов параметрического проектирования. Также существует больше сотни плагинов и аддонов, которые расширяют функциональность Grasshopper. Ниже некоторые наши любимые ресурсы.

Сообщества по плагинам



food4Rhino (WIP)) - это новое сообщество с плагинами, созданное McNeel. Как пользователь, вы найдете новейшие плагины для Rhino, аддоны для Grasshopper, Textures и Backgrounds. Добавляйте ваши комментарии, обсуждайте новые инструменты, связывайтесь с разработчиками этих приложений, делитесь своими скриптами. <http://www.food4rhino.com/>



Страница аддонов Grasshopper <http://www.grasshopper3d.com/page/addons-forgrasshopper>

Аддоны, которые мы любим



DIVA-for-Rhino позволяет пользователям выполнять оценку экологическое воздействие индивидуальных строений и городского ландшафта. <http://diva4rhino.com/>



Element это плагин mesh геометрии для Grasshopper, позволяющий создавать mesh, анализировать, изменять, подразделять и сглаживать. <http://www.food4rhino.com/project/element>



Firefly предоставляет всеобъемлющий комплект ПО инструментов, направленный на создание связи между Grasshopper и микро-контроллером Arduino. <http://fireflyexperiments.com>



GhPython - это интерпретирующий компонент Python для Grasshopper, который позволяет выполнять динамические скрипты любого типа. В отличие от других скриптовых компонентов, GhPython позволяет использовать rhinoscriptsyntax, чтобы начать писать скрипт без необходимости быть программистом. <http://www.food4rhino.com/project/ghpython>



HAL это плагин Grasshopper для программирования промышленных роботов, поддерживающих ABB, KUKA и Universal Robots. <http://hal.thibaultschwartz.com/>



Расширяет возможности Grasshopper для создания и привязывания к геометрии, включая свет, блоки и текстовые объекты. Также обеспечивает доступ к информации об активном документе Rhino, касающейся материалов, слоев, типов линий и других настроек. <http://www.food4rhino.com/project/human>



Karamba это интерактивная программа, параметрических конечных элементов. Она позволяет проанализировать реакцию 3-пространственной балки и оболочки под произвольными нагрузками.
<http://www.karamba3d.com/>



Kangaroo это движок Live Physics для интерактивной симуляции, оптимизации и нахождения формы прямо из Grasshopper. <http://www.food4rhino.com/project/kangaroo>



Fold panels используют изогнутое сгибание и распространение контрольной панели на поверхности с порядком системы атTRACTоров. <http://www.food4rhino.com/project/robofoldkingkong>



LunchBox плагин для Grasshopper для исследования математических форм, панелизации, структуры и организации рабочего процесса. <http://www.food4rhino.com/project/lunchbox>



Meshedit это набор компонентов, которые расширяют возможности Grasshopper в работе с mesh.
<http://www.food4rhino.com/project/meshedittools>



Параметрические инструменты для создания и использования прямоугольных сеток, атTRACTоров и поддержки креативного преобразования параметрических шаблонов. <http://www.food4rhino.com/project/pt-gh>



Platypus позволяет авторам Grasshopper передавать геометрию в веб пространство в реальном времени. Он работает как чат-комната для параметрической геометрии и позволяет немедленно делать мэшап 3D модели в браузере. <http://www.food4rhino.com/project/platypus>



TT Toolbox представляет набор различных инструментов, которые мы используем на регулярной основе из Core Studio у Thornton Tomasetti. Надеемся, что вы оцените это приложение.

<http://www.food4rhino.com/project/tttoolbox>



Weaverbirdтопологический моделер, который содержит много из известных подразделений и изменяемых операторов, с легкостью используется дизайнерами. Этот плагин реконструирует форму, разделяет любую mesh, состоящую даже из полилиний, и помогающую подготовить ее к производству.

<http://www.giuliopiacentino.com/weaverbird/>

Дополнительные пособия

The Firefly Primer Эта книга направлена на обучение основам электроники (использование Ардуино), а также различным цифровым/ физическим техникам прототипирования для специалистов новых в этой сфере. Это не всеобъемлющая книга по электронике (так как уже написано много великолепных книг посвященных это теме). Вместо этого, эта книга сфокусирована на ускорении процесса прототипирования. Автор - Andrew Payne. <http://fireflyexperiments.com/resources/>

Essential Mathematics Essential Mathematics использует Grasshopper для объяснения проектировщикам основных положений математических идей, необходимых для эффективного развития вычислительных методов 3D моделирования и компьютерной графики. Автор - Rajaa Issa.

<http://www.rhino3d.com/download/rhino/5.0/EssentialMathematicsThirdEdition/>

Generative Algorithms Серия книг, которые нацелены на развитие различных концептов в области Генеративных Алгоритмов и Параметрического Проектирования. Автор - Zubin Khabazi.
<http://www.morphogenesism.com/media.html>

Rhino Python Primer Это пособие направлено на обучение программированию абсолютных новичков, людей, которые немного пробовали себя в программировании или экспертов-программистов, ищущих быстрое введение в методологию Rhino. Автор - Skylar Tibbits.

<http://www.rhino3d.com/download/IronPython/5.0/RhinoPython101>

Общие ссылки

Wolfram MathWorld математический онлайн ресурс, организованный Эриком В. Вайсштайном при помощи нескольких тысяч участников. С тех пор, как первый контент появился онлайн в 1995 MathWorld превратился в связующее звено математической информации между математическим и образовательным сообществами. Данные этого ресурса широко используются в журналах и книгах на всех уровнях

образования. <http://mathworld.wolfram.com/>

Дальнейшее чтение

- Burry, Jane, and Mark Burry. *The New Mathematics of Architecture*. London: Thames & Hudson, 2010.
- Burry, Mark. *Scripting Cultures: Architectural Design and Programming*. Chichester, UK: Wiley, 2011.
- Hensel, Michael, Achim Menges, and Michael Weinstock. *Emergent Technologies and Design: Towards a Biological Paradigm for Architecture*. Oxon: Routledge, 2010.
- Jabi, Wassim. *Parametric Design for Architecture*. Laurence King, 2013.
- Menges, Achim, and Sean Ahlquist. *Computational Design Thinking*. Chichester, UK: John Wiley & Sons, 2011.
- Menges, Achim. *Material Computation: Higher Integration in Morphogenetic Design*. Hoboken, NJ: Wiley, 2012.
- Peters, Brady, and Xavier De Kestelier. *Computation Works: The Building of Algorithmic Thought*. Wiley, 2013.
- Peters, Brady. *Inside Smartgeometry: Expanding the Architectural Possibilities of Computational Design*. Chichester: Wiley, 2013.
- Pottmann, Helmut, and Darij Bentley. *Architectural Geometry*. Exton, PA: Bentley Institute, 2007.
- Sakamoto, Tomoko, and Albert Ferré. *From Control to Design: Parametric/algorithmic Architecture*. Barcelona: Actar-D, 2008.
- Woodbury, Robert. *Elements of Parametric Design*. London: Routledge, 2010.

