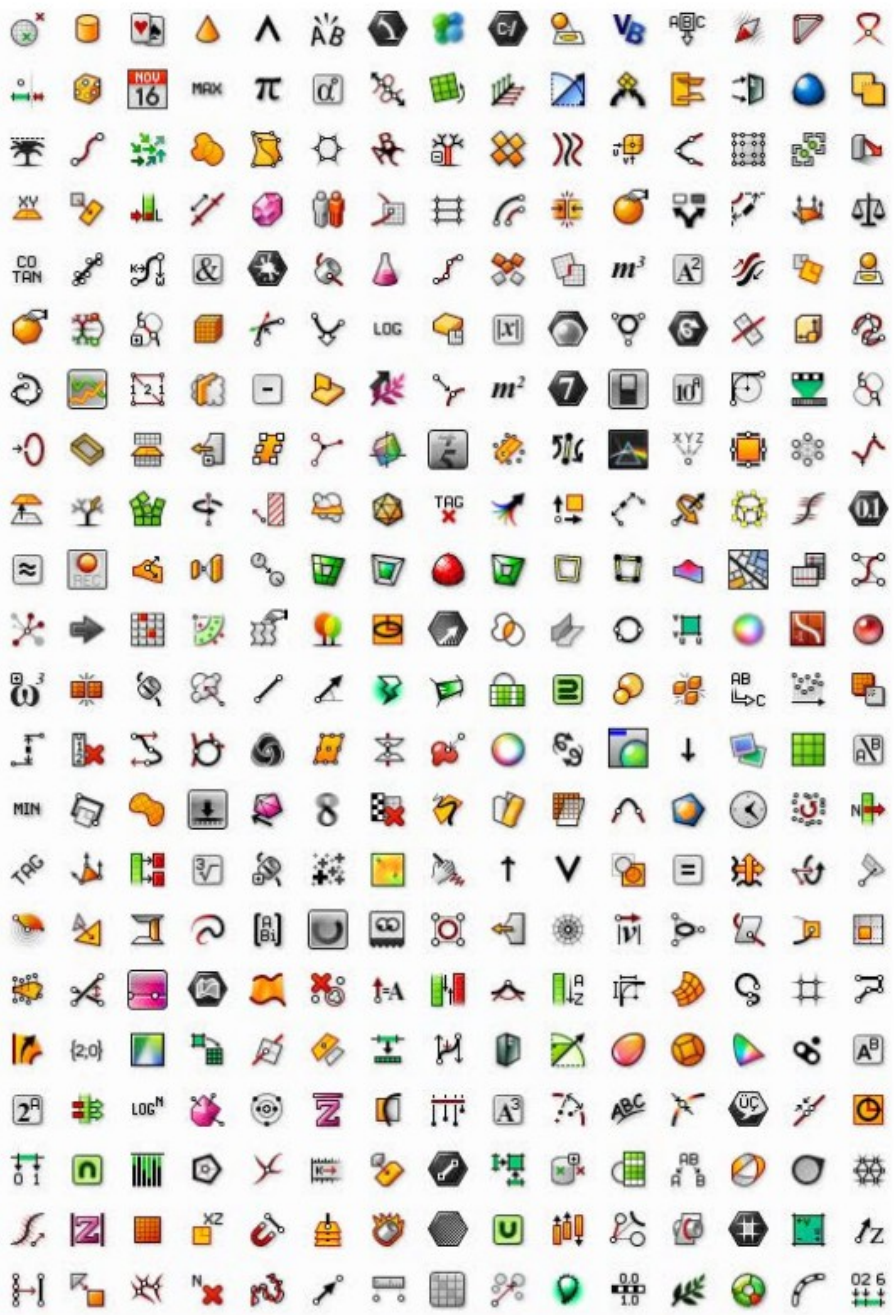
FOUNDATIONS

THE GRASSHOPPER PRIMER THIRD EDITION

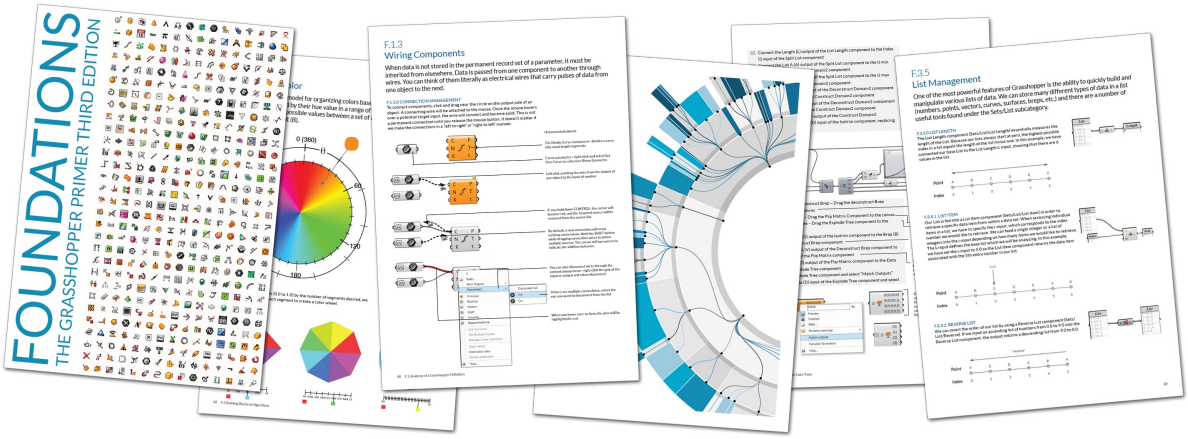


Grasshopper.
Учебник для начинающих.
Основы.
Редакция V3.2.
Автор перевода:
Дмитрий Булка

RHINO-HELP.COM

Учебник Grasshopper.

Третья Редакция (V3.2)



> Grasshopper предоставляет графический редактор алгоритмов, который тесно интегрирован с инструментами моделирования Rhino, позволяющий дизайнерам создавать дефинишины, генерирующие формы, от самых простых до внушающих благоговение.

ДОБРО ПОЖАЛОВАТЬ!

Вы только что открыли третье издание Grasshopper Primer. Этот учебник был первоначально написан Andrew O. Payne из Lift Architects для Rhino4 и Grasshopper version 0.6.0007 который уже во время его выпуска был гигантски обновлён до уже стабильной платформы Grasshopper. Теперь мы оказываемся перед лицом ещё одного важного сдвига в развитии, так что обновление данного учебника для начинающих было просто необходимо. Мы очень рады добавить описания множества этих замечательных обновлений Grasshopper, продвинутых членами сообщества любителей Grasshopper.

С этим отличным фундаментом наша команда Mode Lab принялась за развитие и дизайн обновленной третьей редакции учебника.
Эта редакция предоставляет полное руководство пользователя наиболее актуальной текущей версии Grasshopper (version 0.90076), выделяя, на наш взгляд, наиболее самые захватывающие обновления функций. Переработанный текст, графика и рабочие примеры предназначены научить визуальному программированию абсолютного новичка, а также обеспечить быстрое погружение в технологию Генеративного Дизайна для опытного ветерана. Наша цель - чтобы этот учебник служил в качестве полевого справочника для новых и существующих

пользователей, желающих ориентироваться в тонкостях использования Grasshopper в их творческой практике.

Этот учебник под названием Основы знакомит Вас с основными понятиями и даёт фундаментальные навыки эффективного использования рабочих процессов Grasshopper.
«Основы» являются первым томом в предстоящей коллекции учебников по Grasshopper. Это первый том, проводящий через введение в пользовательский интерфейс Grasshopper, анатомию дефинишинов Grasshopper и три главы, посвященных параметрической концепции и примерам применения:

- **Введение** — Что такое Grasshopper и как его можно использовать?
- **Здравствуй, Grasshopper** — Создание вашего первого дефинишина.
- **Анатомия Grasshopper-дефинишина** — Из чего состоит дефинишин?
- **Строительные Блоки Алгоритмов** — Начните с простого, чтобы создать сложное.
- **Проектирование с использованием Списков** — Что такое списки данных и как можно ими управлять?
- **Проектирование с использованием Деревьев Данных** — Какова структура данных и какое значение она имеет для моего процесса?
- **Приложение** — заканчивает этот том, предоставляя Вам ресурсы для самостоятельного продолжения исследований.

ПРОЕКТ «GRASSHOPPER PRIMER» (Учебник Grasshopper)

Grasshopper Primer – это проект с открытым исходным кодом, инициированный Bob McNeel, Scott Davidson и командой Grasshopper Development из Robert McNeel & Associates.

Mode Lab является авторами Третьего Издания этого учебника. http://modelab.is

БЛАГОДАРНОСТЬ

Отдельное спасибо David Rutten за бесконечное вдохновение и неоценимый труд пионера Grasshopper. Мы также хотели бы поблагодарить Andrew O. Payne за предоставленные примеры работ с которых этот проект была инициирован. Наконец, большое спасибо Bob McNeel и всем в Robert McNeel & Associates за их щедрую поддержку во все эти годы.

НЕОБХОДИМОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Rhino5

Rhino 5.0 является лидером рынка программ 3D-моделирования и промышленного дизайна. Очень сложные формы могут быть непосредственно смоделированы или введены через 3D-сканеры. Благодаря мощному движку, основанному на полной поддержке NURBS, в Rhino 5.0 можно создавать, редактировать, анализировать и преобразовывать кривые, поверхности и твёрдые тела. Нет никаких ограничений по сложности, степени или размеру.

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

Grasshopper

Для дизайнеров, которые ищут новые формы, используя алгоритмы генерации, Grasshopper предоставляет графический редактор алгоритмов, который тесно интегрирован с инструментами моделирования Rhino. В отличии от RhinoScript или Python, Grasshopper не требует знания абстрактного синтаксиса скриптов, но, тем не менее, позволяет дизайнерам создавать дефинишины, генерирующие формы, от самых простых до внушающих благоговение.

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

ФОРУМЫ

Форум Grasshopper очень активный и представляет собой замечательный ресурс для публикации вопросов/ответов и поиску помощм по чему угодно.
Форум имеет категории для обсуждения общих вопросов, ошибок и багов, образцов и примеров, а также раздел наиболее часто задаваемых вопросов.

http://www.grasshopper3d.com/forum

Раздел Общих вопросов содержит ответы на многие имеющиеся у Вас вопросы, а также полезные ссылки:

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

Форумы McNeel для разрешения более общих вопросов, касающихся Rhino3D стоит проверить Обсуждения на McNeel Forum.

http://discourse.mcneel.com/

ОБЩИЕ ССЫЛКИ

Essential Mathematics
Essential Mathematics (Математические Основы) используют Grasshopper, чтобы предоставить профессиональным дизайнерам основные математические концепции, знание которых необходимо для эффективного развития вычислительных методов 3D-моделирования и компьютерной графики. Автор: Rajaa Issa.
http://www.rhino3d.com/download/rhino/5.0/EssentialMathematicsThirdEdition/

Wolfram MathWorld
Мир Математики — интернет-ресурс для математиков, который собрал Eric W. Weisstein при содействии нескольких тысяч пользователей. После того, как эти данные впервые появились онлайн в 1995 году, MathWorld стала связующим звеном знаний по математике в обоих сообществах: и математических и образовательных. Его записи широко упоминаются в журналах и книгах, охватывающих все уровни образования.
http://mathworld.wolfram.com/

Мы надеемся, что этот учебник, по крайней мере, вдохновит Вас начать исследование огромных возможностей программирования в Grasshopper. Мы желаем Вам удачи, сразу же, как Вы начнёте свой полёт.

Grasshopper. Краткий обзор.

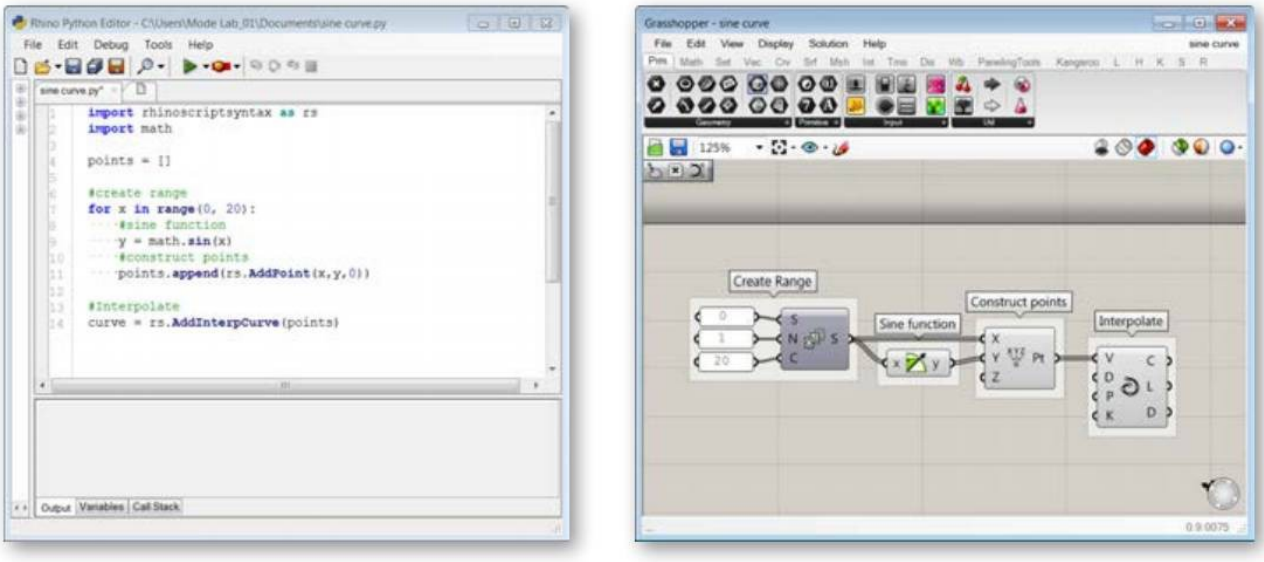
Grasshopper — это визуальный редактор программирования, разработанный David Rutten для Robert McNeel & Associates. В качестве плагина для Rhino3D Grasshopper интегрирован в надёжную и универсальную среду моделирования и используется творческими профессионалами, работающими в широком диапазоне направлений, включая архитектуру, инжиниринг, промышленный дизайн и многое другое. В тандеме Grasshopper и Rhino дают нам возможность использовать точный параметрический контроль над моделью, способность исследовать процессы генеративного дизайна, а также даёт платформу для развития высокоуровневой программной логики — и всё это в пределах интуитивного графического интерфейса.

Происхожение Grasshopper можно отследить до появления функциональной кнопки “Record History” (Запись Истории) в 4-й версии Rhino3D. Эта встроенная возможность позволила пользователям сохранять процедуры моделирования неявно, на заднем плане, прямо во время моделирования. Если бы Вы протянули лофт через четыре кривые с включенной записью истории, а затем отредактировали контрольные точки одной из этих кривых, то поверхность геометрии обновилась автоматически.
Ещё в 2008 году Дэвид поставил вопрос: «Что, если Вы могли бы иметь более явный контроль над этой Историей?» и родился предшественник Grasshopper - Explicit History (Открытая История). Это открыло возможность подробного редактирования дерева истории и дало пользователю способ разрабатывать логические последовательности, создающие построения, выходящие за рамки существующих возможностей Rhino3D. Спустя шесть лет, мы теперь имеем Grasshopper — надёжный визуальный редактор программирования, функционал которого может быть расширен с помощью внешних дополнений — аддонов сторонних производителей. Кроме того, он в корне изменил рабочие процессы профессионалов различных отраслей промышленности и способствовал активному объединению глобального сообщества пользователей.

Этот учебник фокусируется на фундаменте, предлагая базовые знания, достаточные, чтобы начать погружение в регулярное использование Grasshopper и несколько трамплинов, помогающих развить собственную творческую практику. Перед погружением в описание, схемы и примеры, приведённые далее, давайте обсудим, что из себя представляет визуальное программирование, базовые знания по интерфейсу и терминологии Grasshopper, а также характеристики «живой» обратной связи с окнами проекции (viewport) и их взаимодействие с пользователем.

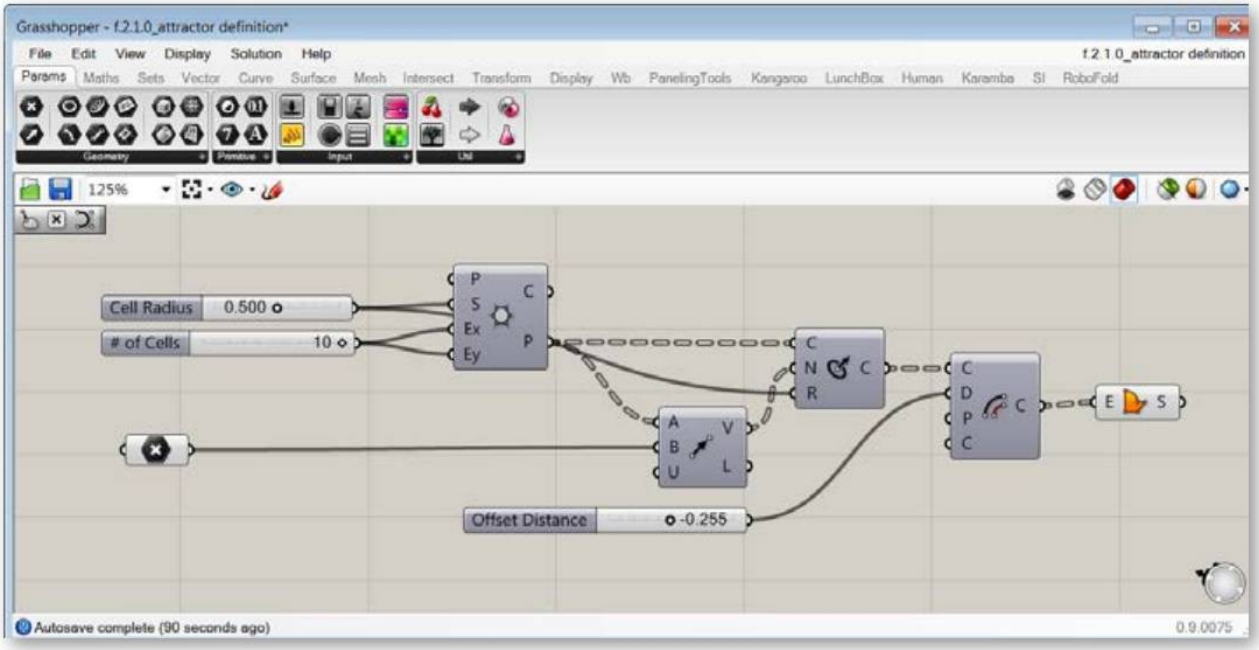
Визуальное программирование — это парадигма программирования, в рамках которой пользователь управляет графическими логическими элементами вместо правки текста. Некоторые из самых известных языков программирования, таких как C#, Visual Basic, Processing более близки к написанию исходного кода для Rhino, например, Python и Rhinoscript требуют от нас писать код, который связан особым синтаксисом конкретного языка. В отличие от этого визуальное программирование позволяет использовать подключение функциональных блоков в последовательность действий, где из «синтаксиса» необходимо только чтобы входы приёма данных у блоков соответствовали по типу, и в идеале, были организованны в зависимости от желаемого результата — см. разделы по согласованию потоков данных и

проектирование с использованием деревьев данных. Такие характеристики визуального программирования позволяет избежать барьеров, которые обычно встречаются при попытке выучить новый язык, даже разговорный. На переднем плане также стоит интерфейс, который у дизайнеров располагает Grasshopper в пределах более знакомой территории.



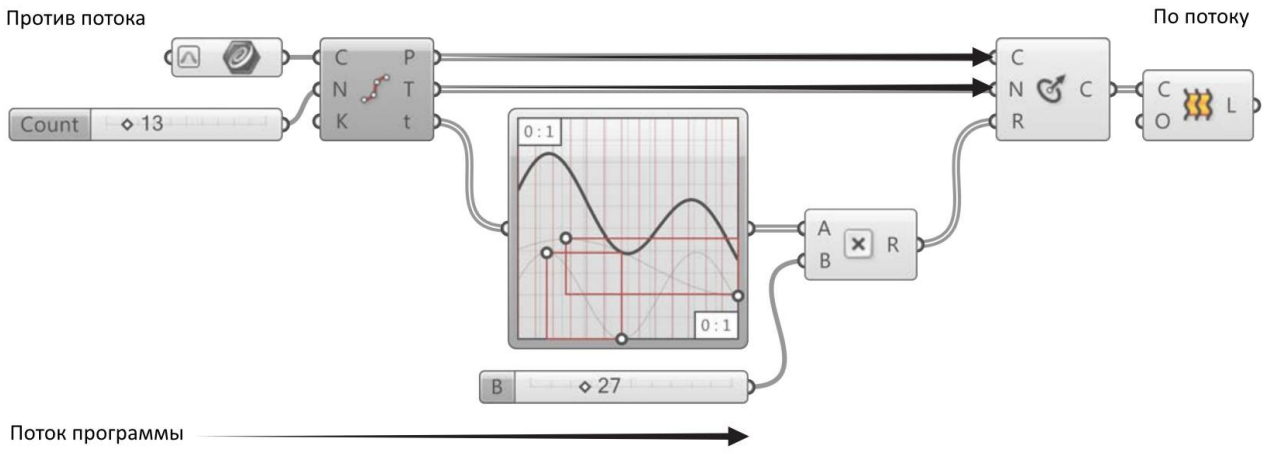
Это изображение отображает процесс построения синусоиды в Python и Grashopper.

Чтобы получить доступ к Grasshopper и его возможностям визуального программирования, нам нужно скачать и установить программу с веб-сайта Grasshopper3D.com. После установки мы можем открыть плагин, набрав “Grasshopper” в командной строке Rhino. Первый раз мы это делаем в новой сессии Rhino, где перед нами предстанет окно редактора Grasshopper. Теперь мы можем добавлять функциональные блоки, называемые “components” (компоненты) на “canvas” (холст), соединяя их с помощью “wires” (проводов), а после сохранить эти “definition” (дефинишины) в файл формата .ghx.



Дефинишин Grasshopper, состоящий из компонентов, связанных проводами на холсте.

После того, как мы начали разработку дефинишина Grasshopper и создали объект “slider” (ползунок) для контроля геометрии, мы можем видеть связь исходных объектов в окнах проекций Rhino и объектов внутри нашего холста. Эта связь, по сути, «живая»: если мы будем перемещать рукоять ползунка, то увидим последствия этого в виде изменений исходного объекта воздействия нашего дефинишина, после того, как программа пересчитает решение и обновит отображение.
К счастью, для нашего быстрого старта начала работы с использованием Grasshopper, предварительный просмотр геометрии выполняется автоматически в виде облегчённого представления.
Важно принять к сведению такую связь сейчас, по мере того, как ваши дефинишины становятся всё более сложными, чтобы научиться управлять потоками данных и состоянием “solver” (решателя), наблюдая предпросмотр в окне Rhino, что позволит предотвратить многие нежелательные головные боли.



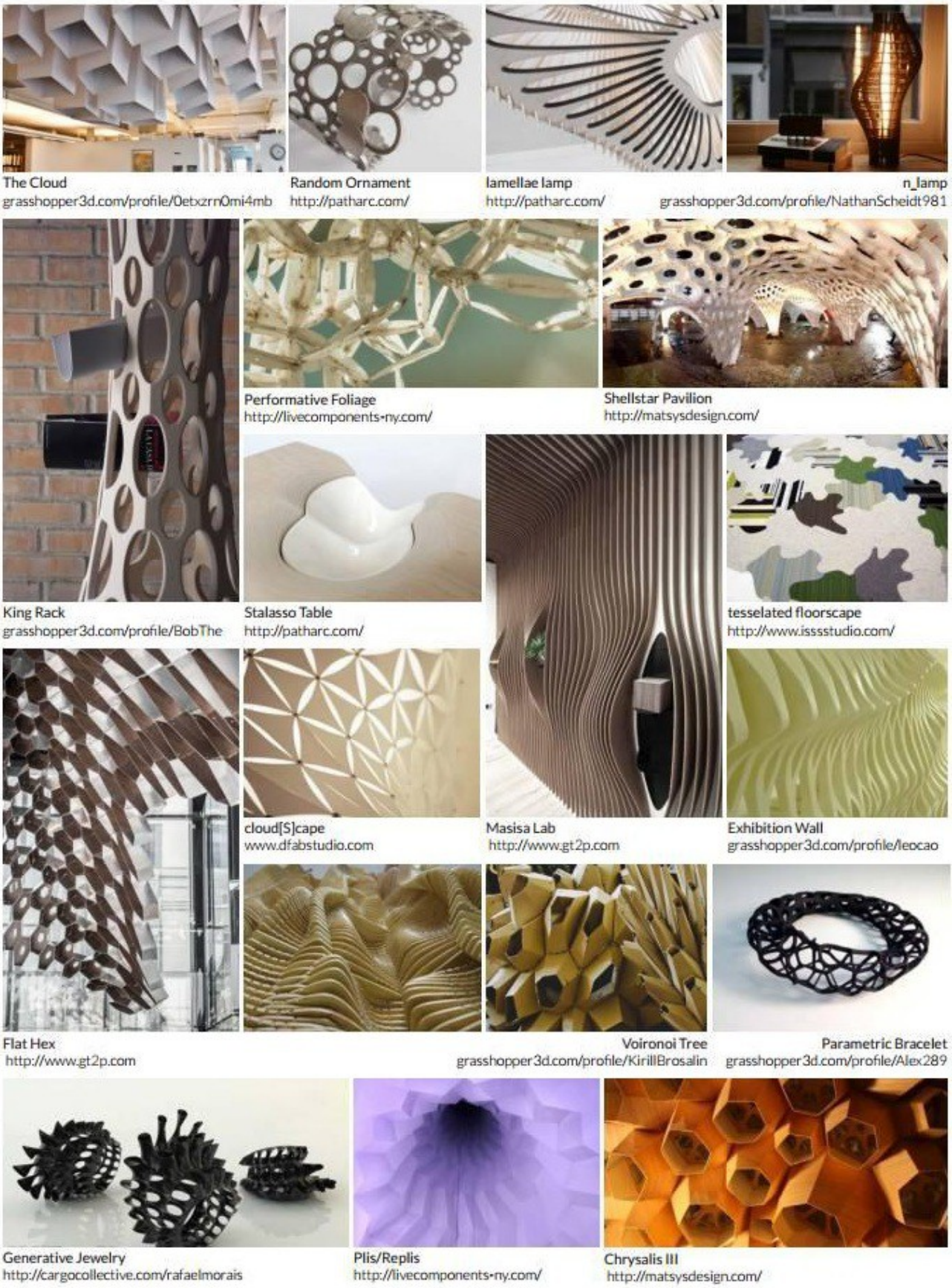
Программный поток течёт слева направо.

ЗАПОМНИ ЭТО

- Grasshopper — это графический редактор алгоритмов, который глубоко интегрирован в инструменты моделирования Rhino3D.
- Алгоритмы — это пошаговые процедуры, предназначенные для выполнения операции.
- Вы можете использовать Grasshopper для разработки алгоритмов, которые в последствии автоматизируют выполнение задач в Rhino3D.
- Если Вам не ясно, как выполнять конкретные операции в Grasshopper, самый простой способ начать — пошагово пытаться вручную использовать команды Rhino и постепенно записывать алгоритм.

Как только Вы начинаете своё первое знакомство с Grasshopper, и в дальнейшем, для развития своих навыков присоединяйтесь к глобальному сообществу Grasshopper, которое полно активными членами из разных областей, с разным уровнем опыта. Форум Grasshopper3D.com — это очень полезный ресурс для того, чтобы задать вопрос, а также делиться находками и знаниями. Пройдя дорогой через это сообщество мы написали данный учебник, глядя как Grasshopper развивается в течении нескольких лет. Добро пожаловать!

Grasshopper в действии



bend lamp
http://patharc.com

Parametric Bar
http://urdirlab.blogspot.com/

Soft Voronoi Shelf
http://www.gt2p.com/

clove lamp
http://patharc.com

Tarrugao Collection
http://www.gt2p.com/

Opening Chronometry
livecomponents-ny.com/

NU:S 3D Printed Shoes
http://www.arturotedeschi.com/

minima(maxima)
www.grasshopper3d.com/profile/JohnBrockway

ZigZag
http://www.issstudio.com/

bayou-luminescence
http://patharc.com/

Divider
http://www.arturotedeschi.com/

Jewelry Lightbox
www.grasshopper3d.com/profile/jovanoviczmilos

Catalyst Hexshell
http://matsysdesign.com/

Shhh the hope keeper
http://www.gt2p.com/

Image credit
www.creditlink.com

Bartop
grasshopper3d.com/profile/MattHutchinson

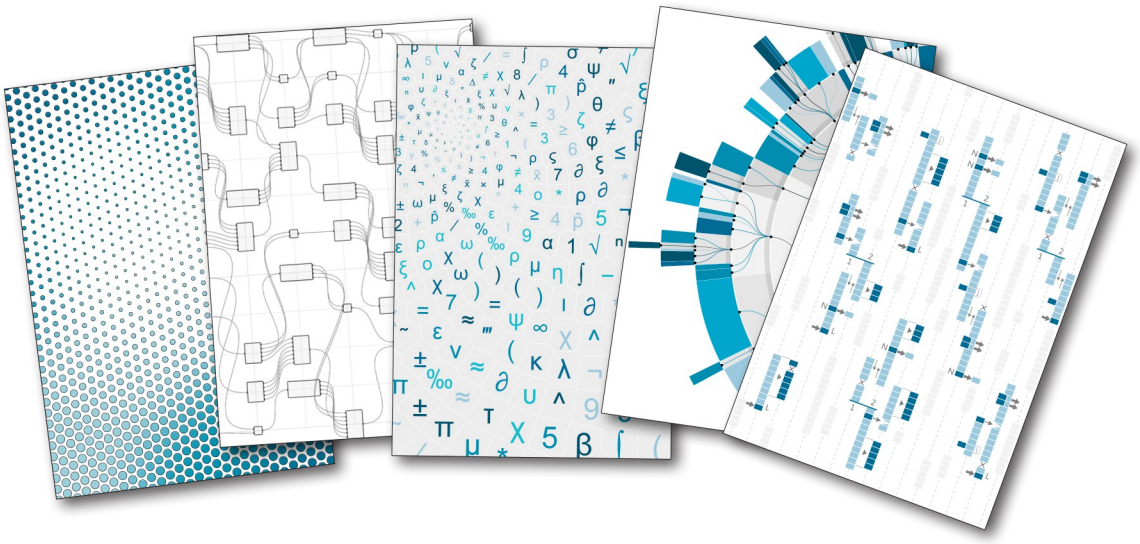
Co Ceiling
http://www.gt2p.com/

Vilu Light
http://www.gt2p.com/

11

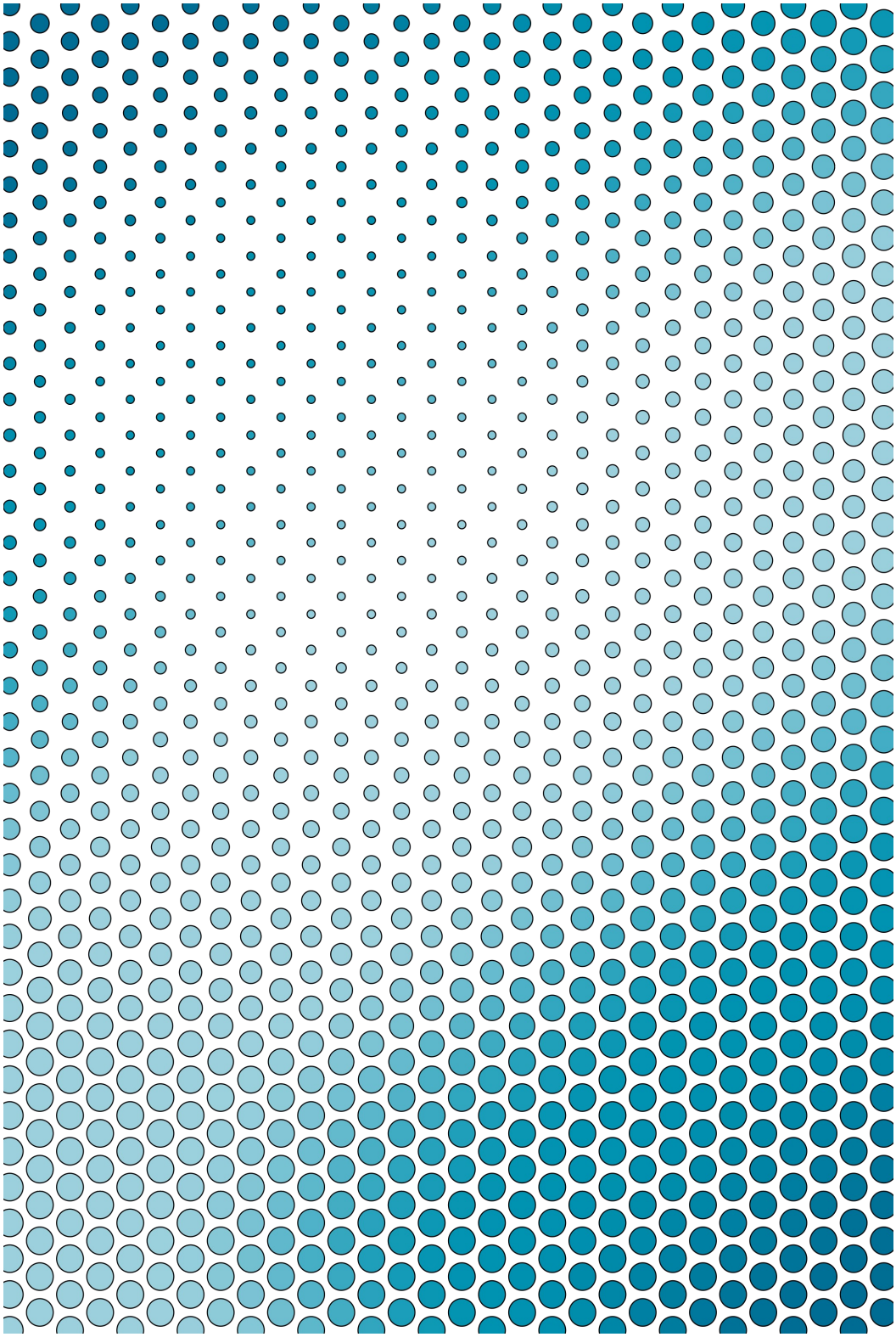
1. ОСНОВЫ

Прочный фундамент к последующим построениям. В этом томе учебника представлены основные понятия и особенности параметрического моделирования с помощью Grasshopper.



1.1. ЗДРАВСТВУЙ, GRASSHOPPER

Grasshopper – это графический редактор алгоритмов, интегрированный в инструменты моделирования Rhino3D. Используйте Grasshopper для разработки алгоритмов, автоматизирующих задачи в Rhino3D.



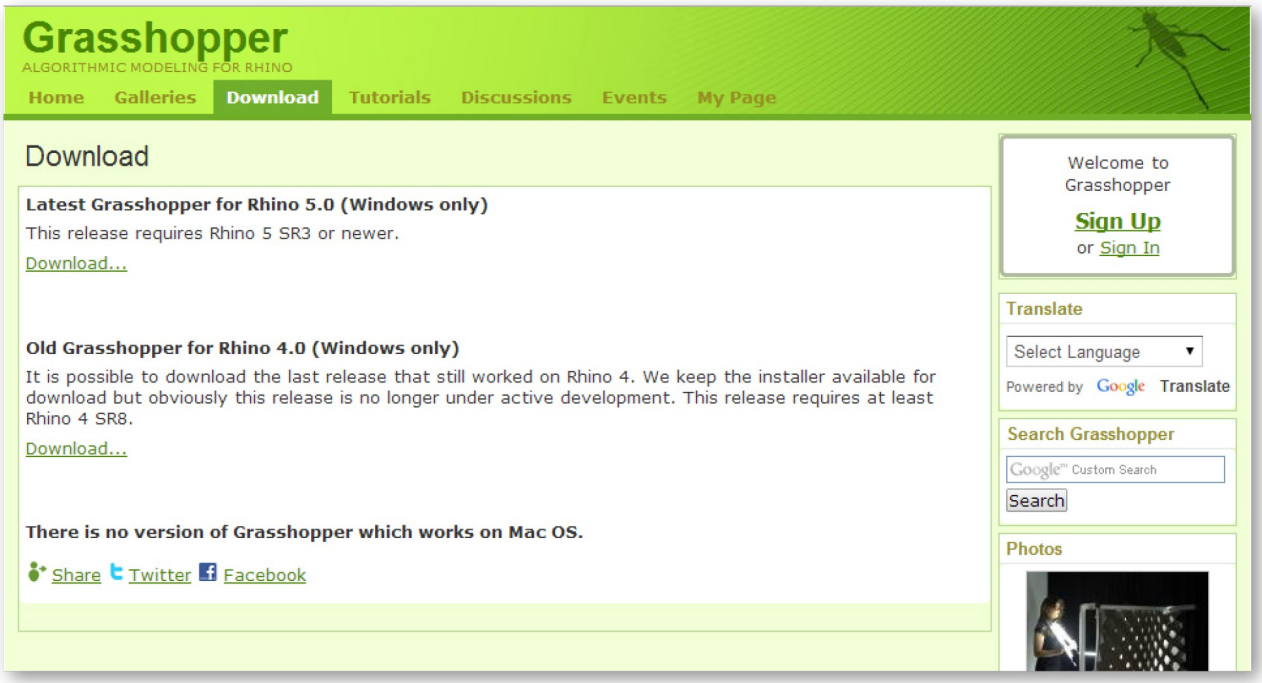
1.1.1. Установка и запуск Grasshopper

Плагин Grasshopper часто обновляется, так что не забудьте обновиться до последней сборки.

Обратите внимание, что в настоящее время нет ни одной версии Grasshopper для Mac.

1.1.1.1. ЗАГРУЗКА

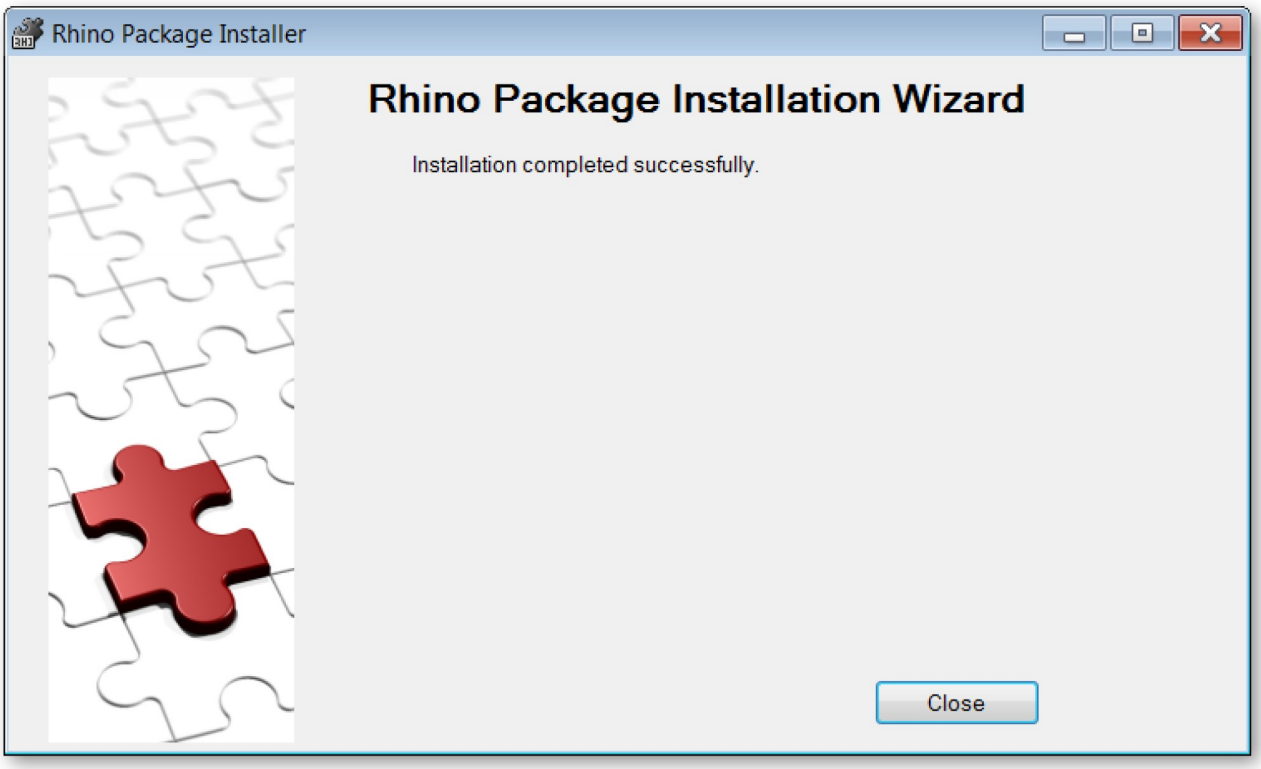
Для того, чтобы скачать плагин Grasshopper, посетите веб-сайт grasshopper3d.com. Кликните по вкладке Download (Скачать) и при появлении запроса на следующем экране введите свой адрес электронной почты. Теперь щёлкните правой кнопкой мыши на ссылке загрузки и в выпавшем контекстном меню выберите «Сохранить как». Укажите место на вашем жёстком диске (примечание: файл не может быть загружен в определённое место локальной сети, так что должен быть загружен только локально на жёсткий диск вашего компьютера) и сохраните исполняемый файл в указанную папку.



Загрузка Grasshopper с веб-сайта grasshopper3d.com.

1.1.1.2. УСТАНОВКА

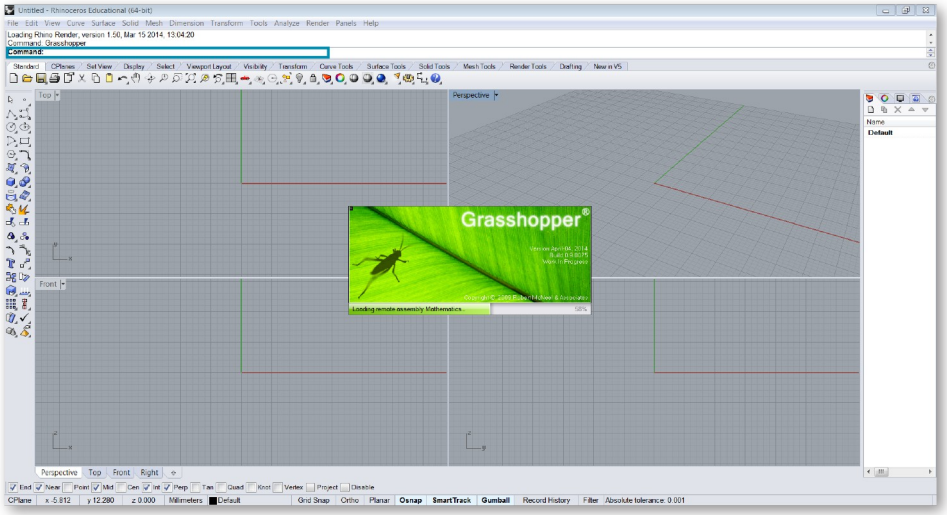
Выберите «Открыть» в диалоговом окне загрузки и следуйте инструкциям инсталлятора. (примечание: для правильной установки плагина Вы должны иметь уже установленным Rhino 5 на вашем компьютере).



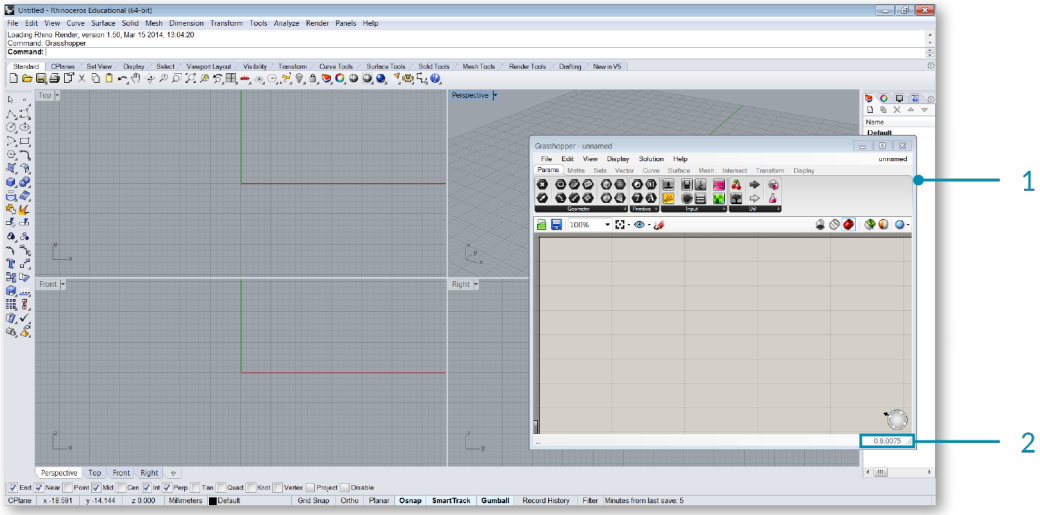
Пошагово следуйте указаниям мастера установки

1.1.1.3. ЗАПУСК

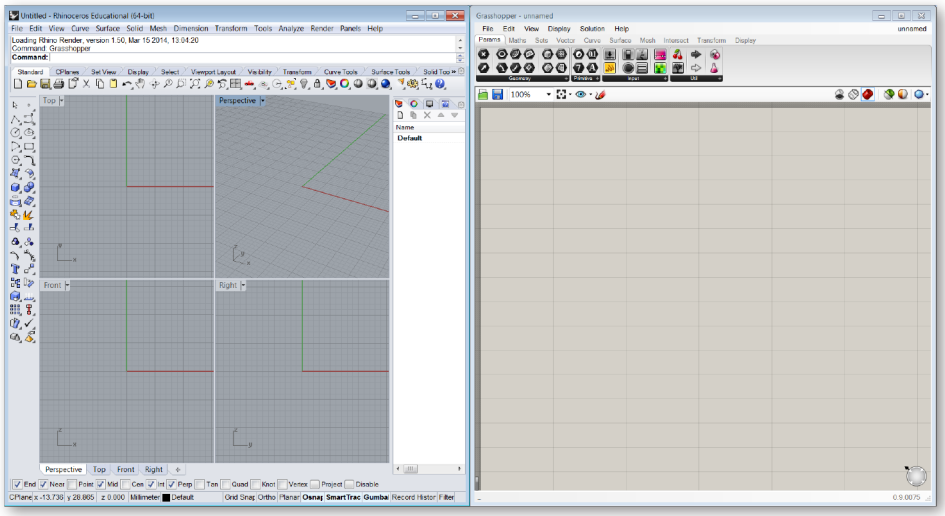
Для запуска Grasshopper введите надпись Grasshopper в Командную строку Rhino. При запуске Grasshopper первое, что Вы увидите — это новое плавающее окно поверх окна Rhino. В этом окне Вы можете создавать программы, составленные из нодов для автоматизации различных типов функций Rhino. Рекомендуем расположить окна так, чтобы окно Grasshopper не перекрывало окна проекций Rhino.



Для загрузки плагина Grasshopper введите в командной строке Rhino «Grasshopper».



1. Окно Grasshopper плавает поверхн окон проекции (вьюпортов) Rhino.
2. Grasshopper отображает номер версии внизу окна.



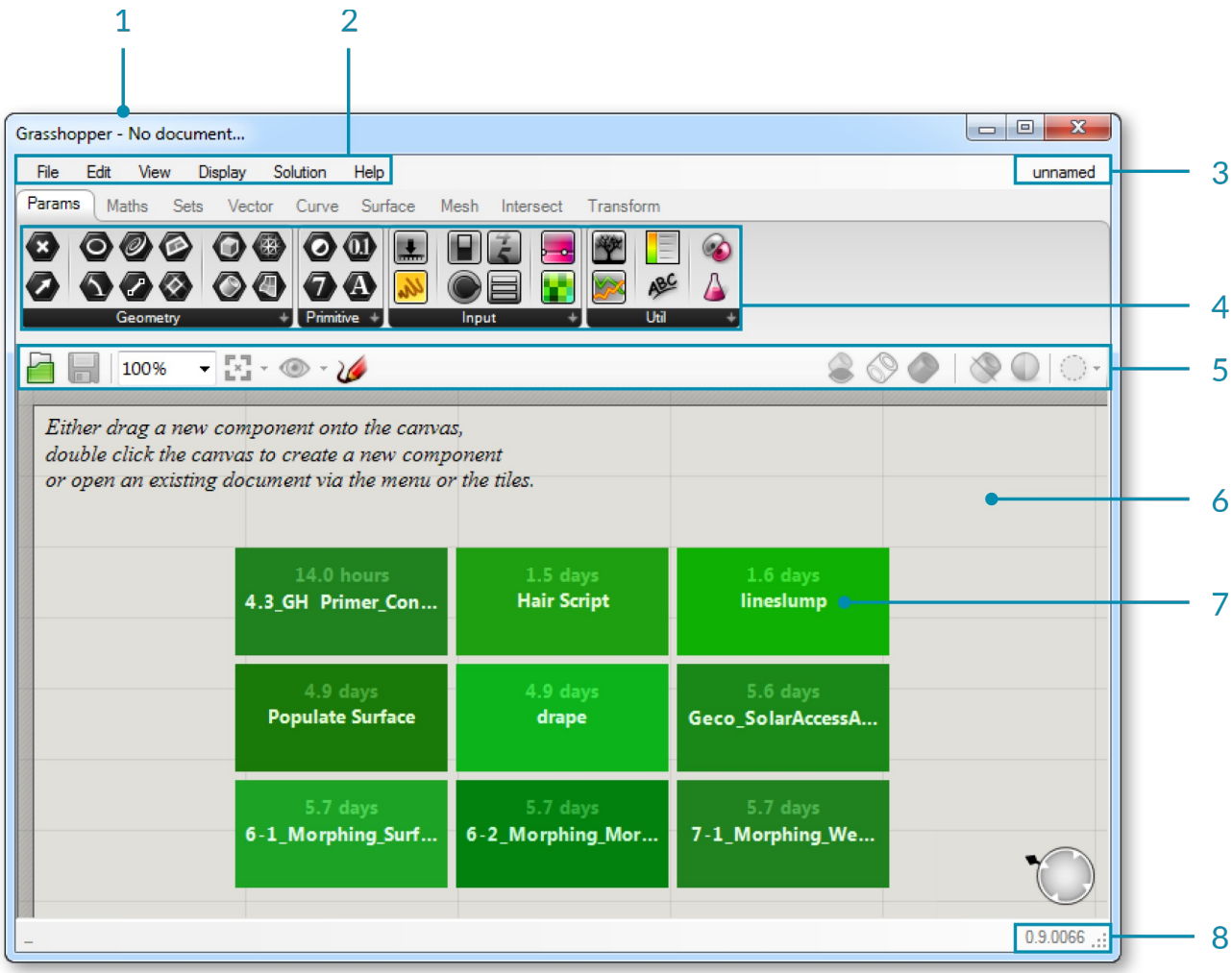
> Распределите окна таким образом, чтобы Grasshopper не перекрывал окна проекций Rhino. Вы можете сделать это быстро, просто потянув окна за заголовок к противоположным сторонам экрана, либо удерживая кнопку Windows, нажав клавиши стрелка влево и стрелка вправо.

ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ GRASSHOPPER (UI)

Визуальный стиль Grasshopper - “plug-and-play” (подключи и работай) обеспечивает дизайнерам возможность объединить творческое решение задач с новаторской системой правил, используя подвижный графический интерфейс.

Давайте начнём с изучения пользовательского интерфейса Grasshopper. Grasshopper — это приложение, визуально программирующее программы, называемые definition (дефинишин) или document (документ), с помощью перетаскивания компонентов по основному окну редактора, называемому canvas (холст). Выходы этих компонентов соединяют со входами последующих компонентов, создавая граф информации, который можно читать слева направо. Давайте начнём с самых основ.

Предполагается, что Вы уже установили плагин Grasshopper (см. F. 0.0). Наберите в командной строке Rhino слово ”Grasshopper” для того, чтобы Редактор Grasshopper отобразился. Интерфейс Грасхопера содержит ряд элементов, большинство которых окажутся весьма знакомы большинству пользователей Rhino. Давайте рассмотрим некоторые особенности интерфейса.



1. Заголовок окна Windows.
2. Строка основного меню.
3. Браузер управления файлами.
4. Палитры компонентов.
5. Панель инструментов Холста.
6. Холст.
7. Эта область в виде сетки прямоугольных ячеек обеспечивает интерфейс возможностью быстрого доступа к файлам, открытых последними. Меню 3x3 отображает (хронологически) недавно просмотренные файлы. Ячейкой красного цвета будут отображаться не найденные файлы (если, например, Вы их удалили или переместили в другую папку).
8. Строка состояния сообщает Вам текущую версию Grasshopper, установленного на Ваш компьютер. Если доступна более свежая версия, то в вашем трее появится всплывающее меню, дающее инструкции по загрузке последней версии.

1.1.2.1. ЗАГОЛОВОК ОКНА WINDOWS

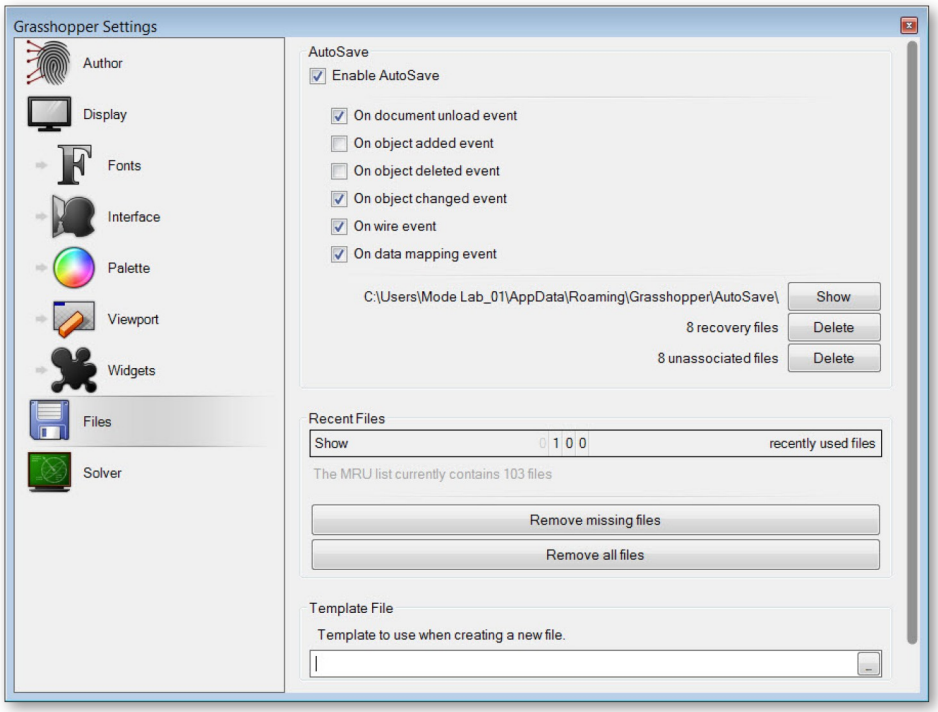
Заголовок Окна Редактора отличается поведением от большинства других диалоговых окон Microsoft Windows. Если его окно не минимизированно, либо максимизированно, то двойной щелчок по заголовку окна заставит сколлапсировать его до минимально возможно малой строки вашего экрана. Это — отличный способ переключаться между плагином и Rhino, потому что минимизирует редактор без необходимости перемещения его в низ экрана или задвигания за другие окна. Обратите внимание, что если Вы закроете Редактор, предварительный просмотр Grasshopper-геометрии в окнах проекций Rhino исчезнет, но физически файл не будет закрыт. Как только Вы снова выполните команду “Grasshopper” в диалоговом окне Rhino, окно Грасхопера вернётся в том же самом состоянии тех же загруженных файлов.
Так происходит потому, что как только Грасхопер загружается с помощью командной строки, ваша сессия Грасхопера остаётся активной, пока этот экземпляр Rhino не будет закрыт.

1.1.2.2. СТРОКА ГЛАВНОГО МЕНЮ

Строка меню похожа на типичные меню Windows, кроме браузера управления файлами справа (см. предыдущий раздел). Меню File (Файл) содержит типичные функции (например, New File (Новый Файл), Open (Открыть), Save (Сохранить), и т. д.), помимо нескольких утилит, позволяющих экспортировать изображения из вашего текущего документа Грасхопера (см. Export Quick Image (Быстрый Экспорт Изображения) и Export Hi-Res Image (Экспорт Изображения Высокого Разрешения)). Вы можете управлять различными аспектами внешнего вида пользовательского интерфейса, используя меню View (Вид) и Display (Отображение), в то время, как меню Solution (Решатель) позволяет Вам управлять различными атрибутами того, как решатель будет вычислять решение для графа.

Стоит отметить, что многими параметрами приложения можно управлять с помощью диалогового окна Preferences (Настройки), доступного через меню File (Файл). Его раздел Author (Автор) позволяет установить персональные метаданные, которые будут храниться в каждом документе Грасхопера. А раздел Display (Отображение) предоставляет Вам тонкую настройку внешнего вида интерфейса. Раздел Files (Файлы) позволяет указать, как часто и где сохранять файл автоматического сохранения (применяемый в случае случайного закрытия или сбоя приложения). И, наконец, раздел Solver (Решатель) позволяет управлять основными и сторонними плагинами, которые могут расширить функциональность.

> ***Примечание:*** Будьте осторожны при использовании клавиш сокращения, так как они будут воздействовать на активное окно, которым может в данный момент быть либо окно Rhino, либо окно Грасхопера, либо любое другое окно внутри Rhino. Использовать клавиатурные сокращения очень удобно, главное только убедиться, что в данный момент активно нужное окно, чтобы случайно не выполнить неправильную команду.



Диалоговое окно Preferences (Настройки) позволяет задать множество настроек приложения Grasshopper.

F.0.1.2 БРАУЗЕР УПРАВЛЕНИЯ ФАЙЛАМИ (FILE BROWSER CONTROL)

Браузер Файлов (File Browser) позволяет Вам быстро переключаться между загруженными в текущий момент файлами, выбирая их с помощью этого выпадающего списка. Доступ к вашим открытым файлам через выпадающий список файлового браузера позволяет быстро копировать и вставлять элементы из открытых дефинишинов. Просто кликните по имени активного файла в браузере управления файлами и отобразится каскадный список всех открытых файлов (включая небольшое изображение миниатюры открытого дефинишина) для лёгкого доступа к ним. Также Вы можете нажимать Alt+Tab, чтобы быстро переключаться между любыми открытыми документами Грасхопера.

Grasshopper — это плагин, который работает «поверх» Rhino и имеет свои собственные типы файлов. Тип файла по умолчанию — файл двоичных данных, сохраняемый с расширением .gh. Другой тип файла известен как Grasshopper XML-файл, который использует расширение .ghx. Файл типа XML (Расширяемый Язык Разметки) использует признаки, чтобы определять объект и атрибуты объекта (во многом, как документ .HTML), но использует пользовательские теги, чтобы определить объекты и данные в пределах каждого объекта. Поскольку XML-документ отформатирован в формате текстового документа, Вы можете открыть любой XML-файл Грасхопера в текстовом редакторе, например, Блокноте, чтобы заглянуть за кулисы и увидеть код.

Конечно, для загрузки любого документа Grasshopper Вы можете использовать стандартный диалог Open File (Открыть Файл), но также можно и просто перетащить файл Грасхопера на холст, чтобы загрузить любой дефинишин.

Grasshopper имеет несколько различных методов, с помощью которых можно открыть файл, и Вы можете указать, какой какую опцию Вы хотели бы использовать при использовании этого метода.

Open File (Открыть Файл): Как следует из названия, этот вариант обработки файла - будет просто открыть любой дефинишин, который вы перетащите на холст.

Insert File (Внедрить Файл): Вы можете использовать эту опцию, чтобы внедрить существующий файл в текущий документ в качестве несвязанных компонентов.

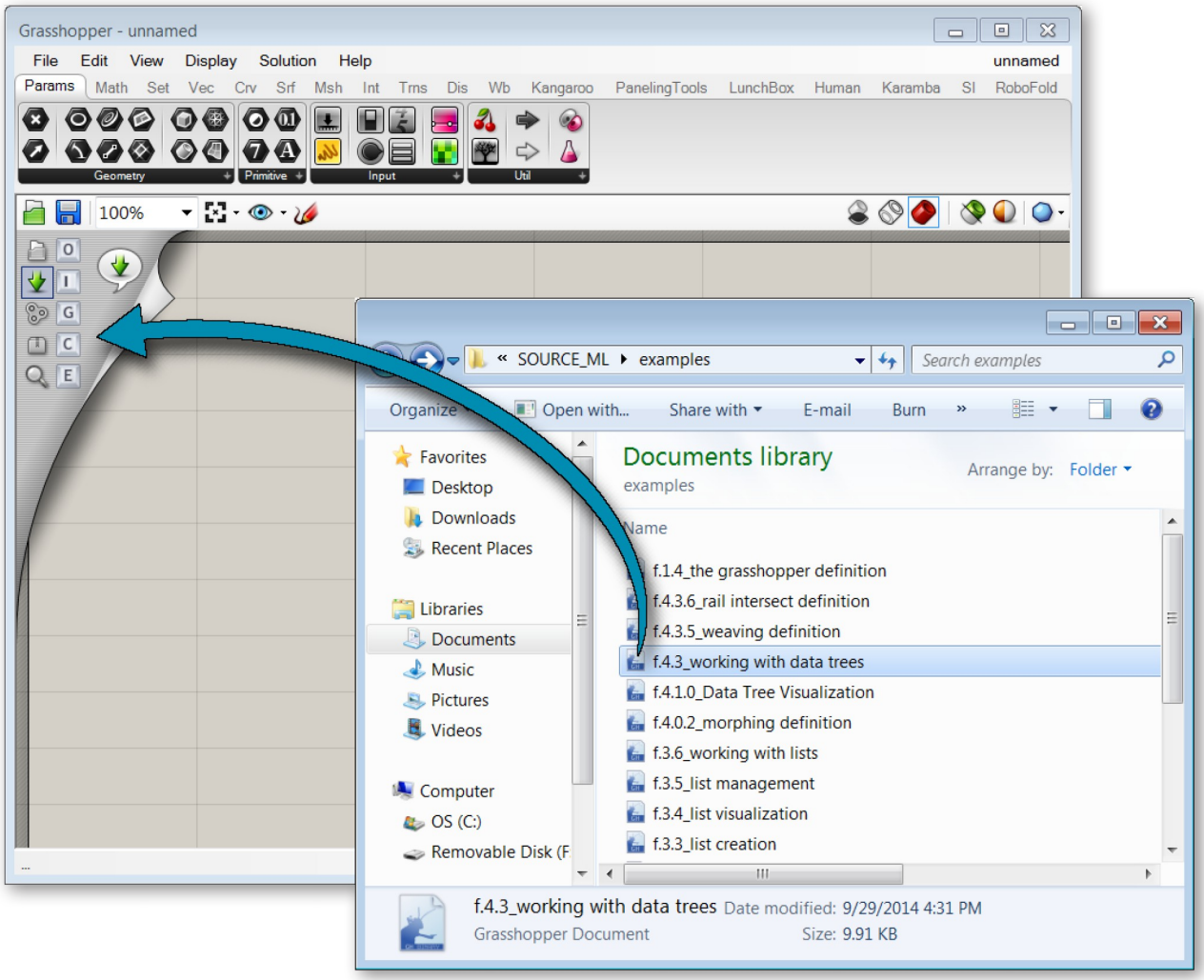
Group File (Сгруппировать Файл): Этот метод будет внедрять файл в существующий документ, но при этом все объекты будут сгруппированы вместе.

Cluster File (Кластер Файла): Подобно методу группировки этот метод внедрит файл в существующий документ, но из группы привнесённых объектов создаст Cluster (Кластер).

Examine File (Изучить Файл): Позволяет Вам открыть файл в заблокированном состоянии, то есть Вы можете изучить особенности файла, но не сможете вносить в этот дефинишин изменения.

У Grasshopper также есть особенность автосохранения, которая будет периодически проявляться в связи с определёнными действиями пользователя. Список настроек автосохранения можно найти через Главное меню (Меню File (Файл) — Preferences (Настройки) — Files (Файлы)). В случае закрытия активного экземпляра Rhino, всплывёт диалоговое окно с запросом, хотите ли вы сохранить файлы Грасхопера, открытые на момент закрытия Rhino.

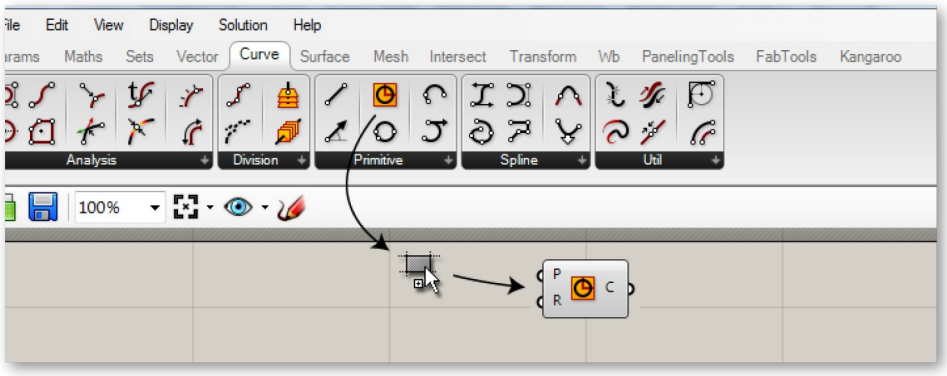
> ***Примечание:*** Автосохранение работает только при условии, что файл буже был сохранён, по крайней мере, один раз.



Перетаскивайте файлы на Холст.

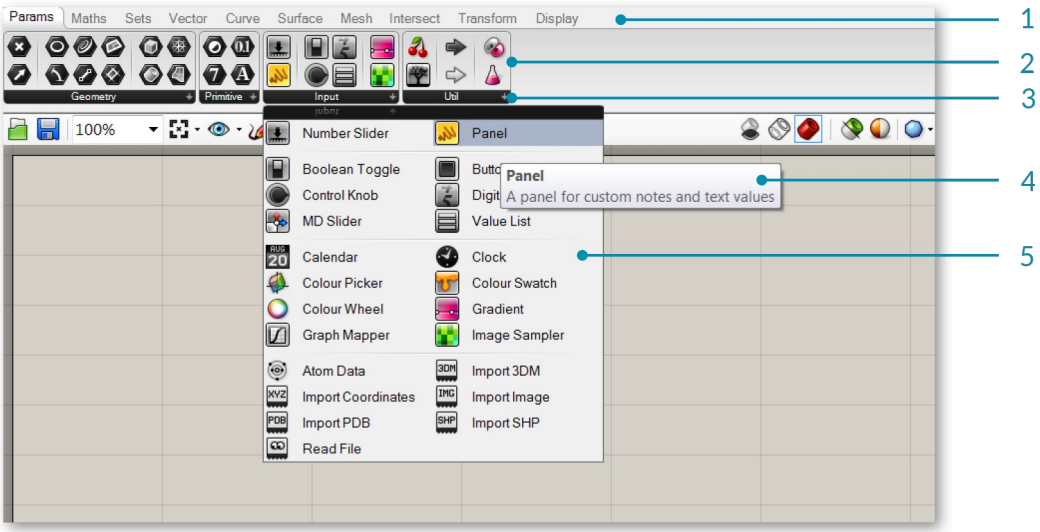
1.1.2.4. ПАЛИТРЫ КОМПОНЕНТОВ (COMPONENT PALETTES)

Эта область организует компоненты по категориям и подкатегориям. Категории отображаются в виде вкладок, и подкатегории отображаются в виде раскрывающихся панелей. Все компоненты относятся к определенной категории. Эти категории помечены, чтобы помочь Вам найти конкретный компонент, который Вы ищете (например, “Params” для примитивов всех типов данных или “Curves” для всех инструментов, связанных с кривыми). Чтобы добавить компоненты на холст, Вы можете щёлкнуть левой кнопкой мыши на объекте в выпадающем меню и перетащить компонент из меню непосредственно на холст.



Чтобы добавить компонент на холст — перетащите его из палитры.

Поскольку в каждой подкатегории может быть значительно большее количество компонентов, чем может поместиться на свёрнутой палитре, на ней отображается ограниченное количество иконок. Если изменить высоту палитры компонентов и ширину окна Grasshopper, то это повлияет на количество отображаемых компонентов в каждой подкатегории. Чтобы увидеть все компоненты, входящие в данную подкатегорию, просто щёлкните на чёрной полосе, находящейся внизу каждой подкатегории панели. При этом откроется выпадающее меню, которое обеспечивает доступ ко всем компонентам в данной подкатегории.



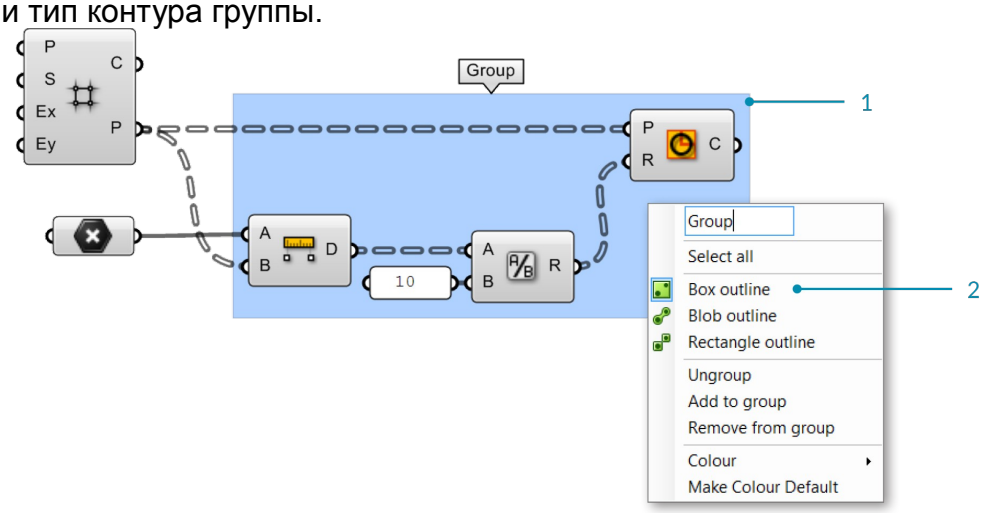
1. Корешок категории.
2. Панель подкатегории.
3. Кликните по чёрному корешку для открытия панели меню подкатегории.
4. Подержите курсор мыши над компонентом, чтобы получить короткое описание.
5. Выпадающее меню.

1.1.2.5. ХОЛСТ (CANVAS)

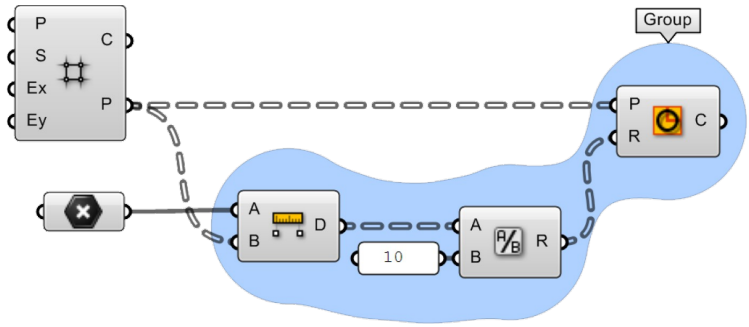
Холст — это основное рабочее пространство для создания дефинишинов (definitions) Grasshopper. Именно здесь Вы будете взаимодействовать с элементами вашей визуальной программы. Вы можете начать работу с размещения компонентов на холсте и соединения их с помощью связей.

1.1.2.5.6. ГРУППИРОВКА (GROUPING)

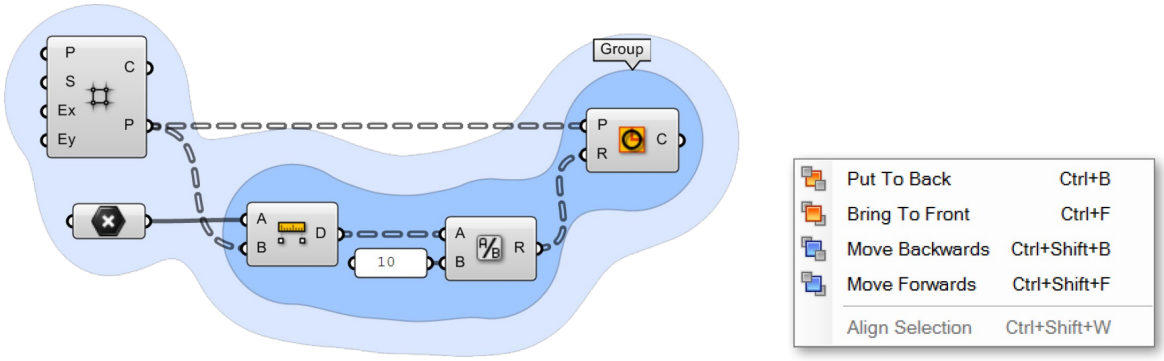
На холсте группирование объектов воедино может быть особенно полезно для читабельности и наглядности. Группировка позволяет Вам быстро выделить и переместить множество компонентов относительно холста. Создать группу можно нажатием клавиш сокращения Ctrl+G, предварительно выделив нужные компоненты. В качестве альтернативного метода можно задействовать Главное Меню — Edit (Правка) - Group (Группа). А щёлкнув правой кнопкой по выделенной цветом области группы можно настроить параметры выделения группы: цвет (Color), прозрачность, имя и тип контура группы.



1. Группа компонентов, очерченная профилем Box Outline (Прямоугольный Контур).
2. Кликните правой кнопкой мыши в любом месте группы, чтобы откорректировать и имя и внешний вид группы.



Также Вы можете определить группу, используя алгоритм meta-ball, использующий профиль Blob Outline (Контур Сгустка).

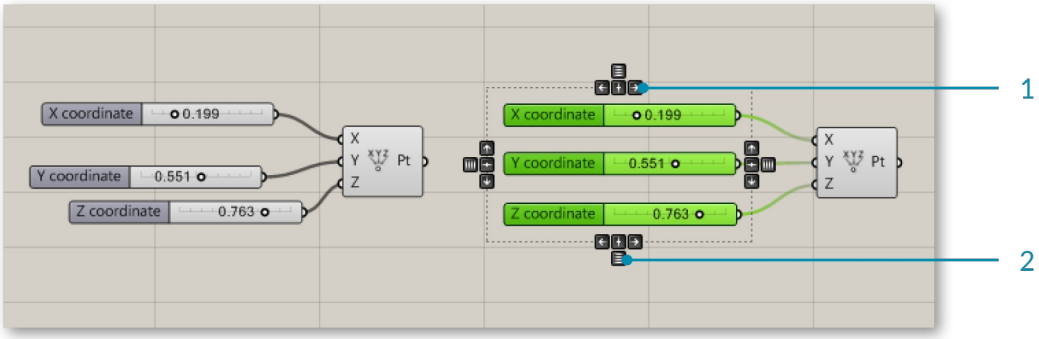


Две группы вложены друг в друге. Цвет внешней группы был изменён на голубой, чтобы визуально отделить одну группу от другой. Определения групп размещены «позади» компонентов и сохраняют своё местоположение относительно компонентов, упорядоченные по глубине в соответствии с порядком глубины групп. Чтобы изменить этот параметр, зайдите в Главное меню — Edit (Правка) — Arrange (Изменить порядок).

1.1.1.7. ВИДЖЕТЫ (WIDGETS)

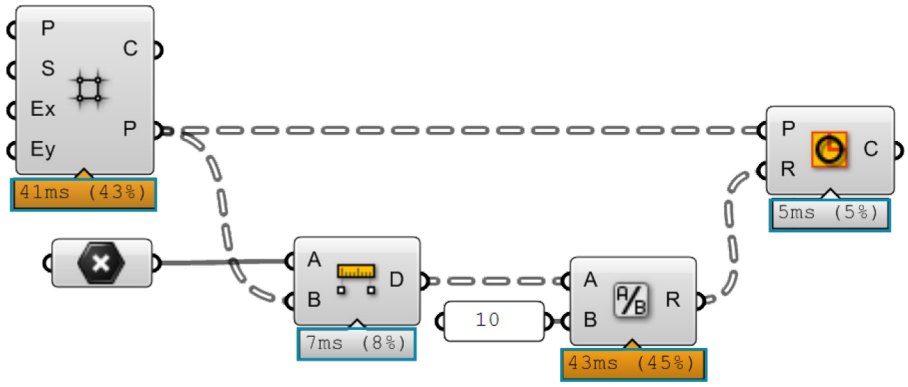
Для Grasshopper доступно несколько виджетов, которые могут существенно помочь Вам выполнять полезные действия. Вы можете переключать состояние включено/выключено (on/off) любого из них через Главное меню на вкладке Display (Отображение). Ниже мы рассмотрим несколько наиболее часто используемых виджетов.

Виджет Align (Выравнивание). Один из весьма полезных виджетов пользовательского интерфейса, который поможет сохранить порядок на вашем Холсте — это виджет Align (Выравнивание). Вы можете получить доступ к виджету выравнивания, как только выделите одновременно несколько компонентов. Кликните по одной из опций, находящейся на пунктирной линии, окружающей выделенные компоненты. Вы можете выровнять объекты по левому или правому краю, по вертикальному и/или горизонтальному центрам, по нижнему или верхнему краям, а также равномерно их распределить. Когда начнёте, Вам может показаться, что эти инструменты иногда смешивают объекты (можно ошибочно наложить объекты друг на друга). Однако, при небольшой практике эти инструменты могут иметь неоценимое значение, когда ваши графы, уже сложной структуры, станут легко читаемыми и понятными.



1. Выровнять по правому краю.
2. Распределить по вертикали.

Виджет Profiler (Профайлер). Профайлер составляет список времени выполнения параметров и компонентов и выделяет худший случай, позволяя Вам легко отыскать узкие места в сети компонентов и сравнить различные компоненты с точки зрения быстродействия. Обратите внимание, что по умолчанию данный виджет выключен.

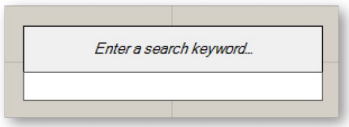


Виджет Profiler (Профайлер) даёт Вам обратную связь, показывая, какие компоненты в дефинишине могут быть причиной длительных циклов вычислений.

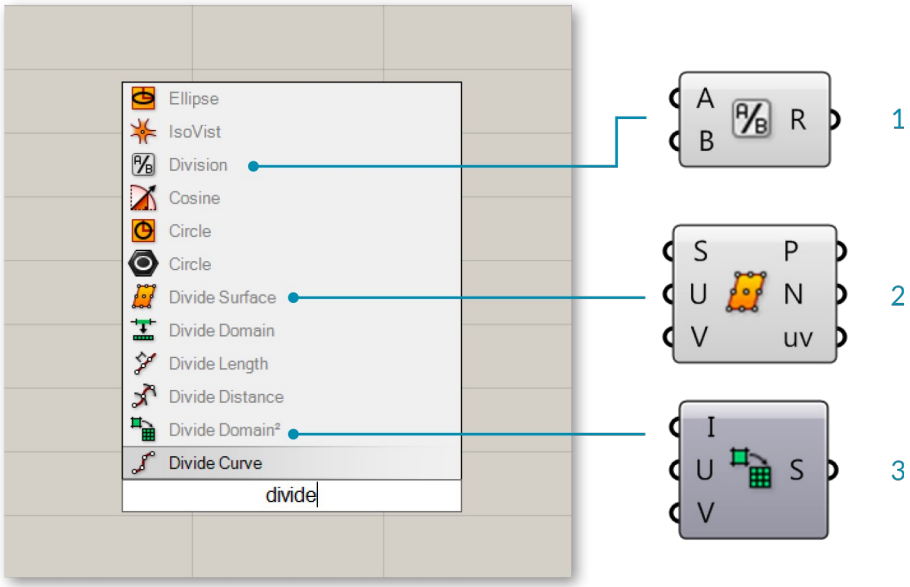
Виджет Markov (Марков). Этот виджет использует цепи Маркова, чтобы «предсказать», какой компонент Вы захотите использовать следующим, основываясь на вашем поведении в прошлом. Цепь Маркова — это процесс, который состоит из конечного числа состояний (уровней) и некоторых известных вероятностей. Чтобы привыкнуть к конкретному пользователю, виджету может понадобиться некоторое время, но со временем, Вы должны начать замечать, что этот виджет начал предлагать компоненты, которые Вы, возможно и должны были использовать следующими.
Виджет Марков может предложить до пяти возможных компонентов в зависимости от вашей недавней деятельности. Вы можете кликнуть по нему правой кнопкой мыши (обычно он располагается в левом нижнем углу холста), чтобы из его контекстного меню выбрать: в каком именно углу его прикрепить, или скрыть его полностью.

1.1.2.8. ИПОЛЬЗОВАНИЕ ФУКЦИИ ПОИСКА (SEARCH)

И хотя, много мысленных усилий было потрачено на то, чтобы размещение каждого компонента на панели компонентов сделало его применение интуитивно понятным для новых пользователей, многим иногда бывает трудно найти конкретный компонент, который оказывается спрятанным глубоко внутри одной категории панели. Но, к счастью, Вы легко можете найти компоненты по имени, просто дважды кликнув в любом пустом месте холста. Это вызовет всплывающее окно поиска. Просто начните вводить имя компонента, который вы ищете, и вы увидите список параметров или компонентов, которые соответствуют вашему запросу.



Дважды кликните в любом пустом месте холста, чтобы вызвать поиск необходимого компонента из Панелей Компонентов по ключевому слову

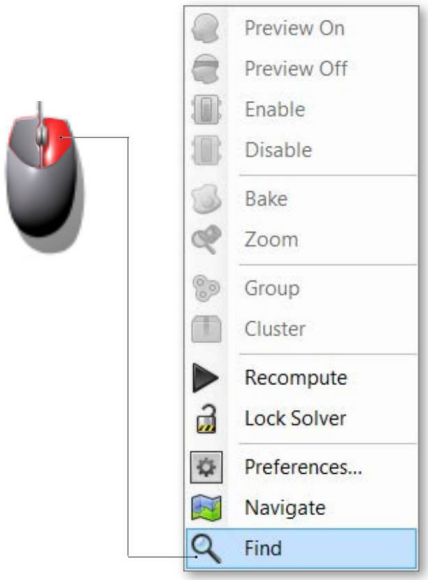


При поиске по слову «divide» перечисляются вариации компонентов:

1. Компонент оператора
2. Division (Деление);
3. Компонент Divide Surface (Делить Поверхность);
4. Компонент Divide Domain2 (Разделить Двумерный Домен).

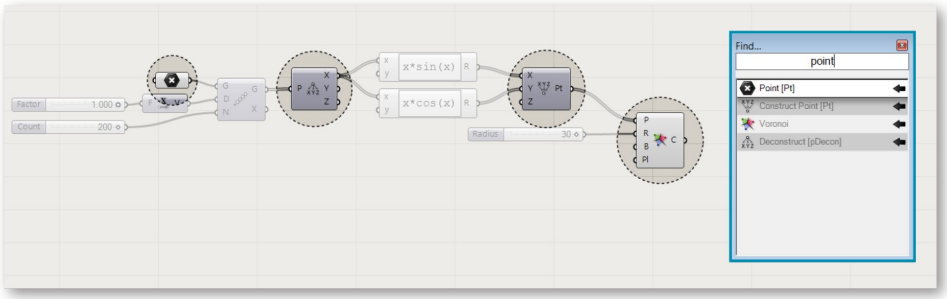
1.1.2.9. ФУНКЦИИ ПОИСКА

Существуют буквально сотни (если не тысячи) компонентов для Grasshopper, которые Вам доступны и для новичка это может сильно повлиять на возможность знать, где искать на Палитре Компонентов определённый компонент. Для быстрого решения данной проблемы дважды щёлкните в любом пустом месте холста, чтобы запустить запрос поиска искомого компонента. Однако, что делать, если нам нужно найти определенный компонент уже размещенный на нашем холсте? Нет повода для беспокойства.

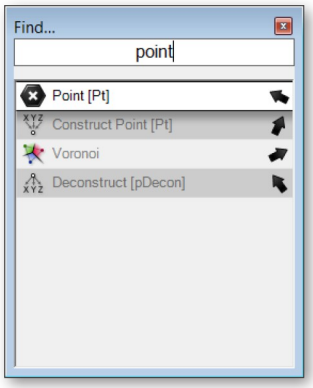


> С помощью клика правой кнопкой мыши в любом пустом месте холста или нажатия клавиши F3 Вы можете вызвать функцию поиска. Начинайте вводить имя искомого компонента.

Функция Поиска (Find) использует очень сложные алгоритмы, которые позволяют найти в дефинишине не только имя любого экземпляра компонента (имя компонента — его название, найденное в Панели Компонентов, которую мы, как пользователи не можем изменить), но также и любые уникальные надписи, которые мы, возможно, назначили для особого компонента (известного как псевдоним (nickname)). Функция поиска позволяет искать на холсте любой тип компонента или искать в текстовых панелях, в записках (scribble), а также в содержимом групп. Как только найдутся соответствия, найденный компонент автоматически выделится (всё вокруг покроется серым цветом, а найденное будет обведено пунктирной линией). Если соответствия обнаружатся в нескольких компонентах, то в окне запроса будет отображён список найденных компонентов, при клике по пункту которого указанный компонент переместится вместе с холстом ближе к списку.



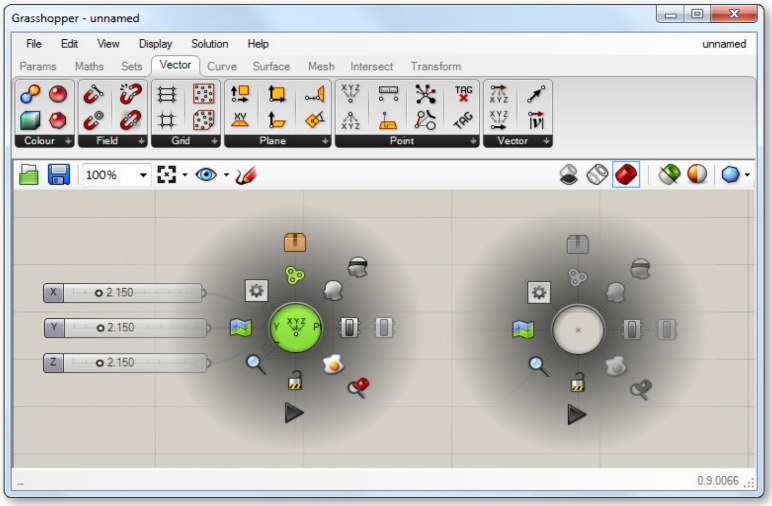
> Функция поиска может быть весьма полезна для поиска конкретного компонента на холсте. Кликните правой кнопкой мыши в любом месте холста, чтобы запустить диалоговое окно Find (Найти).



Маленькая стрелка будет отображаться рядом с каждым элементом списка, которая указывает на соответствующий компонент на холсте. Попробуйте переместить диалоговое окно по холсту и Вы увидите, что стрелки вращаются, отслеживая направление компонентов.

F.0.1.9 ИСПОЛЬЗОВАНИЕ РАДИАЛЬНОГО МЕНЮ (RADIAL MENU)

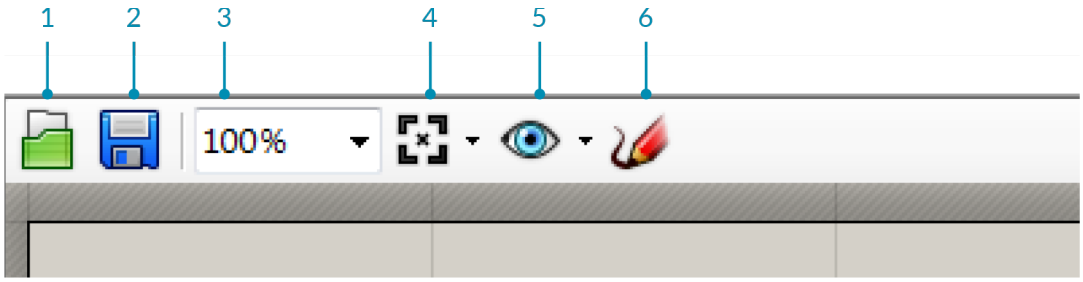
Когда Вы станете более опытным в использовании Грасхопера, Вы начнёте искать способы ускорить ваш рабочий процесс. Использование клавиш сокращения — это лишь один из способов, однако, есть и ещё другая возможность, как можно получить быстрый доступ к ряду полезных инструментов — радиальное меню пользовательского интерфейса (radial UI menu). Вы можете вызвать радиальное меню, нажав пробел (при наведении курсора мыши на холсте или компоненте) или, щелкнув средней кнопкой мыши. Радиальное меню предоставляет доступ к различным инструментам в зависимости от того, вызовите ли Вы меню, нажав непосредственно на компонент, или просто в любом пустом месте холста. На нижепредставленном рисунке Вы можете увидеть, что радиальное меню предоставляет больше возможностей, нажав на выделенный компонент, чем на пустом месте холста. Это меню позволяет резко увеличить скорость, с которой Вы создаёте документы Грасхопера.



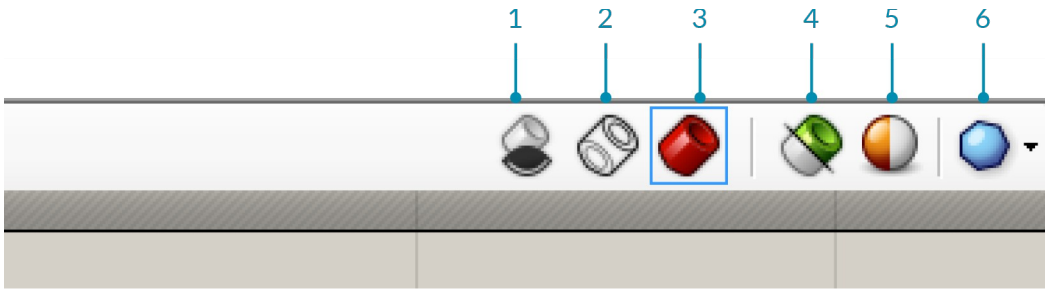
Радиальное меню пользовательского интерфейса позволяет Вам получить быстрый доступ к часто используемым пунктам меню.

1.1.2.11. ПАНЕЛЬ ИНСТРУМЕНТОВ ХОЛСТА (CANVAS TOOLBAR)

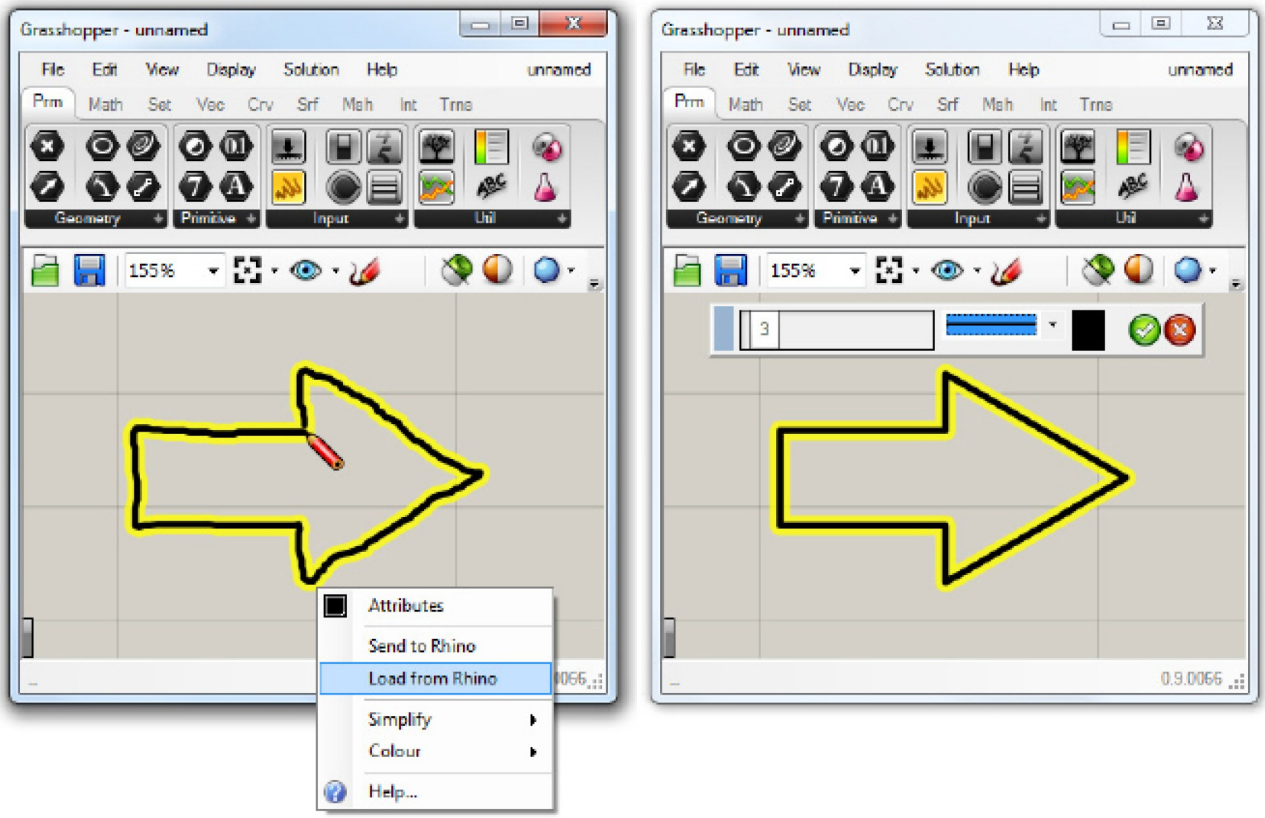
Панель инструментов холста предоставляет быстрый доступ к наиболее часто используемым функциям Grasshopper. Все эти инструменты доступны и через меню и Вы можете скрыть Панель инструментов, если она Вам не нравится. Панель инструментов может быть снова включена через главное меню — view (вид) — Canvas Toolbar (Панель Инструментов Холста).



1. **Open File (Открыть Файл)**: Иконка для открытия файла Grasshopper.
2. **Save File (Сохранить Файл)**: Иконка для сохранения текущего файла Grasshopper.
3. **Zoom Defaults (Масштабирование вида по-умолчанию)**: От установленного по-умолчанию масштаба холста можно увеличить или уменьшить вид на величину предустановленных интервалов.
4. **Zoom Extents (Масштабирование вида до заполнения)**: Масштабирование вида до заполнения содержимым дефинишина (определения). Кликните на стрелочке, следующей за иконкой, чтобы выбрать один из пунктов подменю масштабирования вида для увеличения по определённой области вашего дефинишина.
5. **Named Views (Именованные Виды)**: Эта функция предоставляет меню, позволяющее Вам хранить или вернуть обзор любой области в вашем дефинишине.
6. **The Sketch Tool (Инструмент Рисования)**: Инструмент рисования работает аналогично карандашу. Набор его функций подсмотрен в Adobe Photoshop с добавлением нескольких дополнительных возможностей.

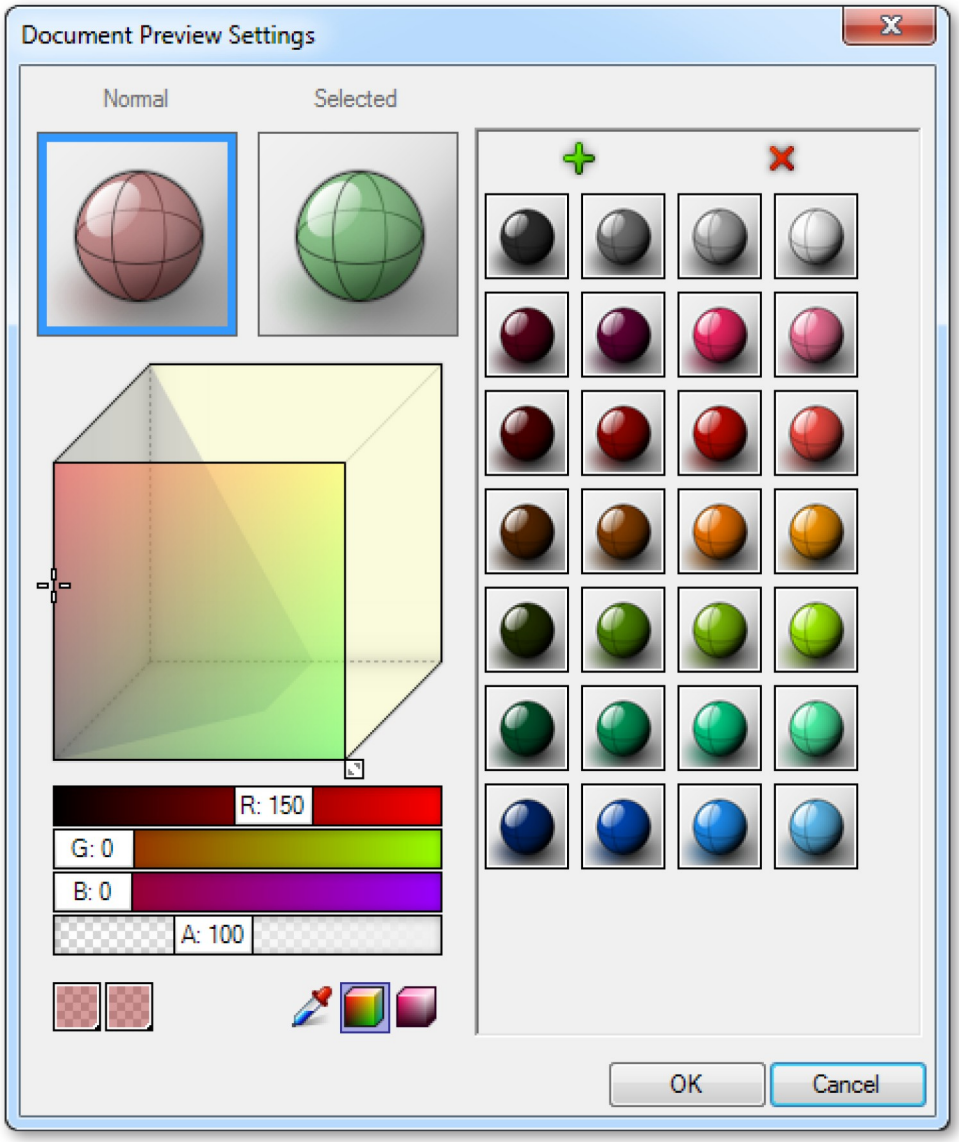


1. **Preview Settings (Настройки Предварительного просмотра)**: Если Grasshopper-компонент генерирует некую геометрическую форму, то, по-умолчанию, предварительный просмотр этой геометрии будет виден в окне проекции (viewport). Вы можете отключить предпросмотр по-объектно, кликая на каждом правой кнопкой мыши и деактивируя функцию предварительного просмотра. Либо, это можно сделать глобально, изменив состояние предпросмотра с помощью одной из этих трёх кнопок.
2. Предварительный просмотр в режиме Wire-frame (Проволочный каркас).
3. Отключить предварительный просмотр.
4. Затенённый (Shaded) предпросмотр (по-умолчанию).
5. **Preview Selected Objects (Предварительный просмотр только выделенных объектов)**: Эта кнопка переключает состояние Grasshopper: будет ли он отображать только ту геометрию, часть компонентов которой выделена, даже если они имеют состояние предпросмотра в положении «off» (отключено).
6. **Document Preview Settings (Настройки Предпросмотра Документа)**: По-умолчанию Grasshopper использует следующие цветовые схемы: для выделенных объектов полупрозрачно-зелёную, а для геометрии невыделенных объектов полупрозрачно-красную. С помощью диалогового окна Document Preview Settings (Настройки Предпросмотра Документа) можно эти настройки цвета переопределить.
7. **Preview Mesh Quality (Качество Предпросмотра Полигональной сетки)**: В целях оптимизации эти настройки позволяют управлять качеством отображения геометрии в виде полигональных сеток и поверхностей, визуализируемой в Rhino. Повышение качества влечёт за собой увеличение времени расчётов, в то время, как более скромные установки ухудшают точность предварительного просмотра геометрии. Следует отметить, что сама геометрия по-прежнему сохраняет высокую степень разрешения при «запекании» в Rhino-документе. Эти настройки влияют лишь на качество отображения предварительного просмотра геометрии.



Инструмент рисования позволяет изменить толщину линии, её тип и цвет. Кликнув правой кнопкой мыши на выделенном объекте рисования Вы можете выбрать функцию упрощения (simplify) вашей линии, для придания ей эффекта сглаженности. Кликните правой кнопкой мыши на объекте рисования и выберите пункт “Load from Rhino” (Загрузить из Rhino). По появившемуся запросу выберите любую 2D-форму из сцены Rhino. После того, как Вы выделили вашу ссылочную форму, нажмите Enter и ваш предыдущий набросок изменит свою конфигурацию по ссылочной форме из Rhino.

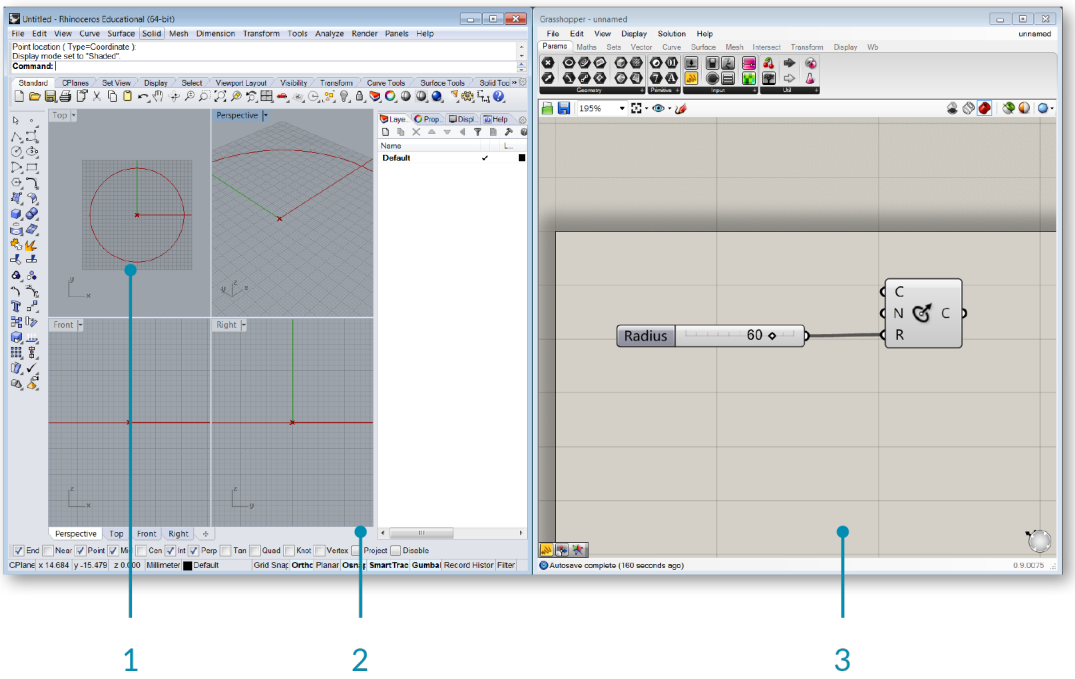
Примечание: Ваш объект рисования может быть перемещён из оригинального местоположения, как только Вы загрузили форму из Rhino. Grasshopper разместит ваш объект рисования относительно начала координат (origin) холста (левый верхний угол) и начала координат плоскости XY глобальной системы координат Rhino (world XY plane).



По-умолчанию Grasshopper использует следующие цветовые схемы: для выделенных объектов полупрозрачно-зелёную, а для геометрии невыделенных объектов полупрозрачно-красную. С помощью диалогового окна Document Preview Settings (Настройки Предпросмотра Документа) можно эти настройки цвета переопределить.

1.1.3. ПОГОВОРИМ О RHINO

В отличии от Rhino-документа, Grasshopper-определение (дефинишин) не содержит каких-либо фактически существующих объектов геометрии. Вместо этого Grasshopper-дефинишин представляет собой набор правил и инструкций о том, как Rhino может автоматизировать задачи.

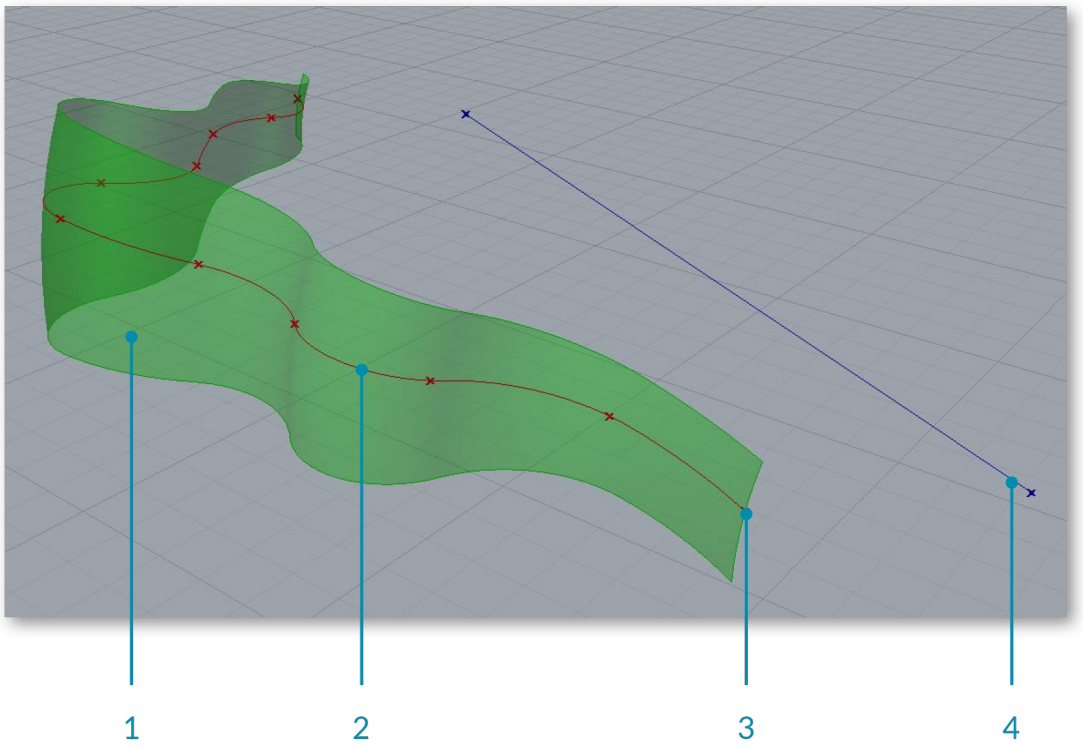


1. Предпросмотр Grasshopper-геометрии.
2. Окна проекций Rhino (viewports).
3. Окно приложения Grasshopper.

1.1.3.1. ОБРАТНАЯ СВЯЗЬ С ОКНАМИ ПРОЕКЦИЙ (ВЬЮПОРТАМИ)

Вся геометрия, генерируемая различными Grasshopper-компонентами будет отображаться (по-умолчанию) в окнах проекций Rhino. Этот предварительный просмотр является лишь аппроксимацией (приближённым отображением) с помощью Open GL фактической геометрии. Поэтому Вы не сможете выделить эту геометрию непосредственно из вьюпорта (viewport) Rhino (сначала её нужно «запечь» (bake) в сцену). Вы можете переключать состояние предварительного просмотра (on/off) геометрии через клик правой кнопкой мыши на компоненте и воздействуя на переключатель состояния предпросмотра. Чтобы обеспечить более эффективную обратную связь во вьюпорте состояние геометрии кодируется цветом. Данное ниже изображение описывает цветовую схему, принятую по-умолчанию.

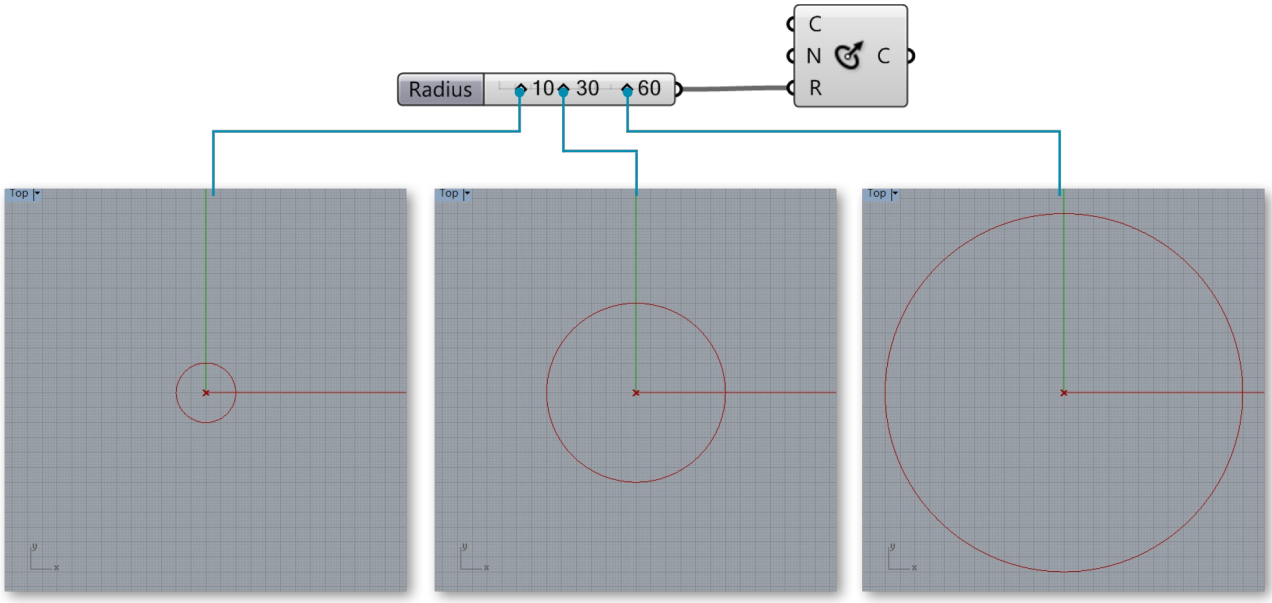
> Примечание: Цветовую схему, принятую по-умолчанию, можно изменить, используя инструменты Document Preview Settings (Настройки Предварительного просмотра Документа), находящиеся на Панели инструментов холста (canvas toolbar).



1. Зелёный цвет геометрии во вьюпорте показывает, что геометрия относится к выделенному в данный момент компоненту.
2. Геометрия, отображаемая во вьюпорте красным цветом, относится к компонентам, не выделенным в текущий момент времени.
3. Точечная геометрия (Point) отрисовывается в виде креста, а не квадратика, чтобы её было легко отличить от других точечных объектов Rhino.
4. Синяя линия обратной связи означает, что Вы прямо сейчас осуществляете выделение во вьюпорте Rhino.

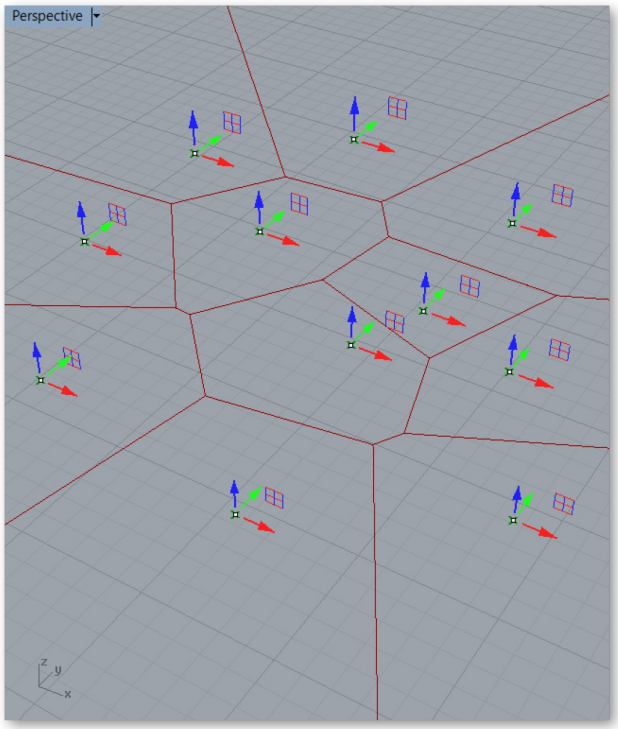
1.1.3.2. ДИНАМИЧЕСКИЕ СВЯЗИ (LIVE WIRES)

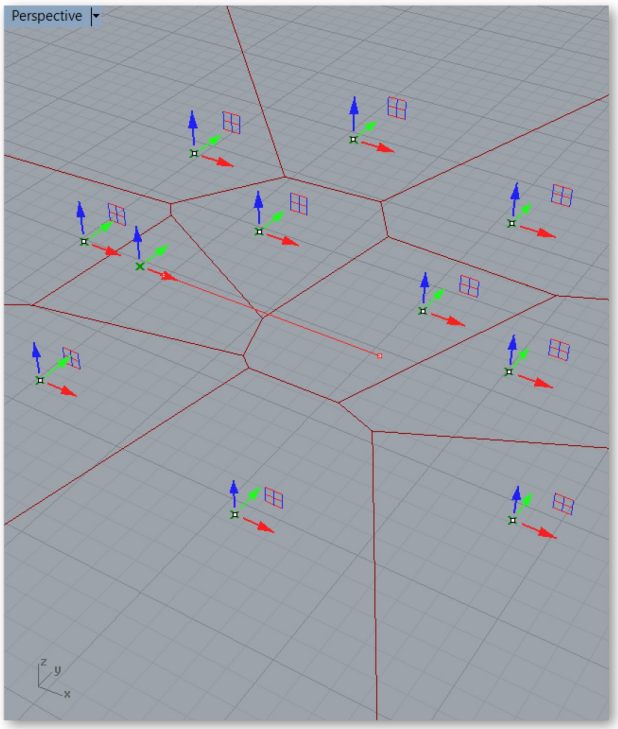
Grasshopper — это динамическая среда. Изменения, внесённые в текущий момент времени сразу же отражаются на предварительном просмотре, обновляя его во вьюпортах Rhino.



1.1.3.3. ВИДЖЕТ GUMBALL (ГАМБОЛ)

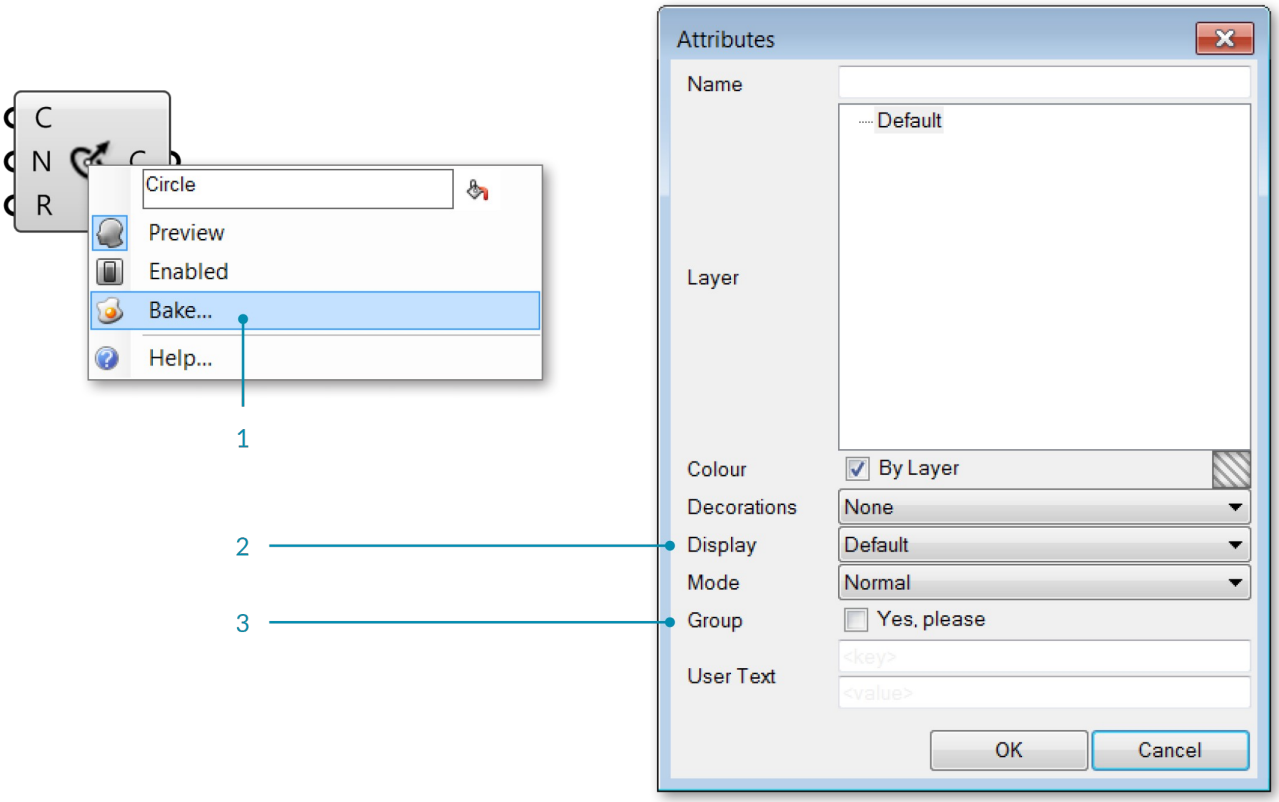
Если геометрия исходит из Grasshopper-параметра, то гамбол позволяет взаимодействовать с этой геометрией непосредственно во вьюпорте Rhino. Эти взаимодействия в режиме реального времени обновляются напрямую от манипуляций с помощью Гамбола. В отличии от этого, если ссылочная геометрия Rhino напрямую продолжает существовать непосредственно в Rhino-документе, обновление, инициируемое из Грасхопера произойдёт только **после** окончания изменений (а не **во время**).





1.1.3.4. ЗАПЕКАНИЕ ГЕОМЕТРИИ

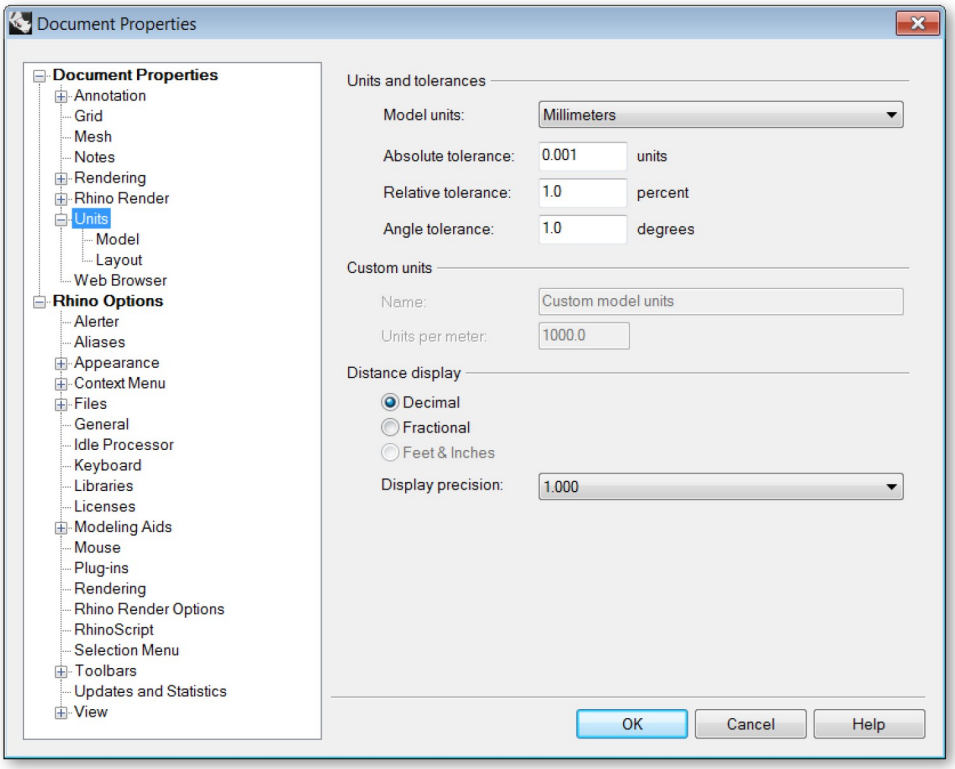
Для того, чтобы иметь возможность взаимодействовать (выделять, редактировать, трансформировать и т. п.) непосредственно в Rhino с геометрией, генерируемой Грасхопером, её необходимо «запечь» (bake). Запекание, создающее новую Rhino-геометрию, основывается на текущем состоянии Грасхопер-графа. И она больше не будет реагировать на изменения в вашем дефинишине.



1. Запекайте по клику правой кнопкой мыши на компоненте и выберите пункт «Bake» (Запечь).
2. Появится диалоговое окно, позволяющее выбрать слой Rhino на который будет запечена геометрия.
3. Группировка запечённой геометрии — удобный способ управлять вность создаваемой Rhino-геометрией, особенно при создании многочисленных объектов из Grasshopper.
4. Чтобы быстро запечь геометрию выделенных компонентов без дополнительных запросов на текущий в данный момент времени слой Rhino — просто нажмите на клавиатуре клавишу Insert.

1.1.3.5. ЕДИНИЦЫ ИЗМЕРЕНИЯ (UNITS) И ДОПУСКИ (TOLERANCES)

Grasshopper наследует единицы измерения и допуски из Rhino. Чтобы сменить единицы измерения, введите в командную строку Rhino команду Document Properties. Вам станут доступны настройки Свойств Документа. Выберите вкладку Units (Единицы измерения), чтобы изменить их величину допусков.



Смена единиц измерения (units) и допусков (tolerances) через меню Document Properties в Rhino.

1.1.3.6. ПАНЕЛЬ ДИСТАНЦИОННОГО УПРАВЛЕНИЯ (REMOTE CONTROL PANEL)

Как только Вы приобретёте навыки работы в Grasshopper, поймёте, что Grasshopper — невероятно мощный и гибкий инструмент, который позволяет Вам исследовать итерационный дизайн (design iterations), используя графический интерфейс.

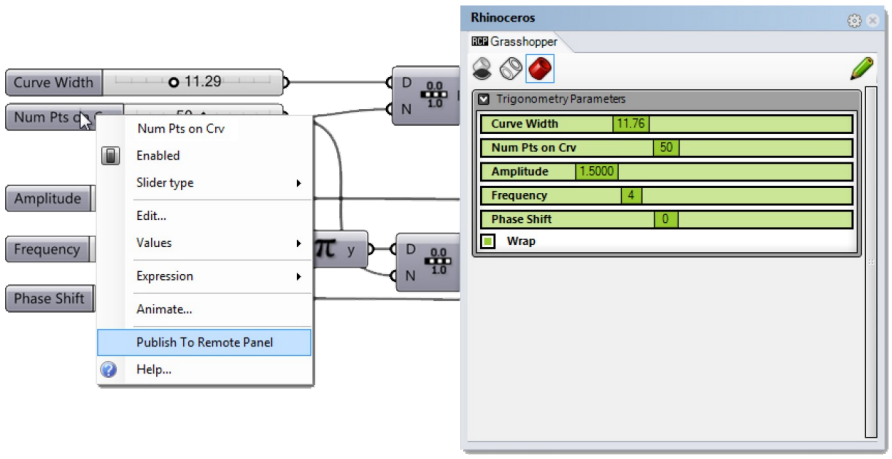
[Итерационный дизайн — это циклический процесс прототипирования, тестирования, анализирования, и улучшения изначального продукта или процесса *(Примечание переводчика)*].

Однако, если Вы работаете на одном экране, то Вы, возможно, уже заметили, что Grasshopper занимает большое пространство экрана. И до сих пор не было более элегантного решения проблемы, кроме постоянного масштабирования и перемещения окон по экрану. Пока не появилась Панель дистанционного управления!

Панель Дистанционного Управления (Remote Control Panel (RCP)) обеспечивает минимальный интерфейс для управления вашим дефинишином, не занимая

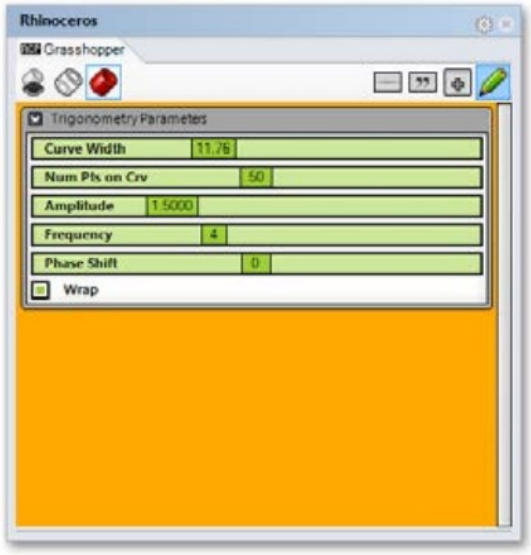
значительной части экрана. RCP может быть активизирована кликом по соответствующему переключателю во вкладке View (Вид) Главного Меню. По-умолчанию, RCP остаётся пустым, то есть не содержит какой-либо информации о вашем текущем Grasshopper-документе. Для заполнения RCP элементами интерфейса (UI), такими, например, как слайдеры, переключатели и кнопки, просто кликните правой кнопкой мыши по элементу и выберите «Publish To Remote Panel» (Вывести на Панель Дистанционного Управления). Это создаст в RCP новую группу, а в этой группе синхронизированный элемент пользовательского интерфейса. Изменение значения элемента в RCP повлечёт обновление значения в Графе, что приведёт к модификации любой связанной геометрии во вьюпорте Rhino, зависящей от от этого параметра. Вы можете вывести (публиковать) несколько элементов и полностью заполнить ими интерфейс, который может быть использован для управления файлом без путаницы в визуальном графе, отобразив RPC поверх вьюпорта Rhino.

> Примечание: RCP будет наследовать имена элементов пользовательского интерфейса и использовать их в качестве этикеток. Хорошая практика — обновлять имена ваших слайдеров и переключателей на понятные и значащие. Они будут транслироваться непосредственно в RCP, что сделает его проще в использовании.



> Для того, чтобы получить элемент пользовательского интерфейса (UI element), например, такой как слайдер, переключатель, кнопка и т.п, отображаемым в Remote Control Panel (Панели Дистанционного Управления), Вы должны сначала его опубликовать (Publish).

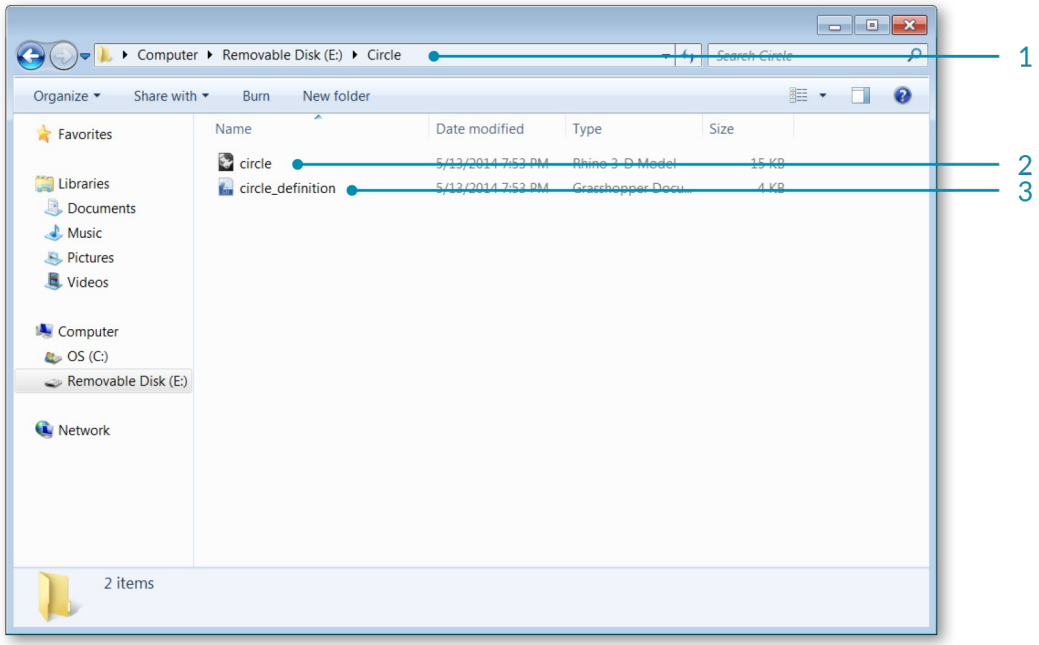
Пользовательский интерфейс (UI) RCP также может быть настроен под конкретного пользователя, что позволяет Вам управлять тем, какие элементы будут отображаться в интерфейсе, а также именовать и раскрашивать различные группы элементов. Чтобы модифицировать компоновку RCP, сначала Вы должны переключить её из рабочего режима (по-умолчанию это режим просмотра RCP) , в режим Правки. Войти в режим редактирования Вы можете, кликнув по зелёному карандашу в верхнем правом углу RCP. Теперь Вы можете создавать новые группы UI, переставлять элементы внутри группы, добавлять метки, менять цвета и многое другое. Чтобы удалить элемент UI, просто перетащите элемент вовне границ RCP. Вы не можете изменить отдельные значения параметров, находясь в режиме редактирования RPC. Для этого Вы должны кликнуть по значку зелёного карандаша, чтобы вернуться в стандартный рабочий режим.



Remote Control Panel (Панель Дистанционного Управления) имеет два режима работы: Режим Правки (слева), который позволяет реорганизовать внешний вид RCP, и Рабочий Режим, в котором Вы можете менять фактические значения элементов пользовательского интерфейса (UI). RCP в режиме правки имеет фон оранжевого цвета.

1.1.3.7. УПРАВЛЕНИЕ ФАЙЛАМИ

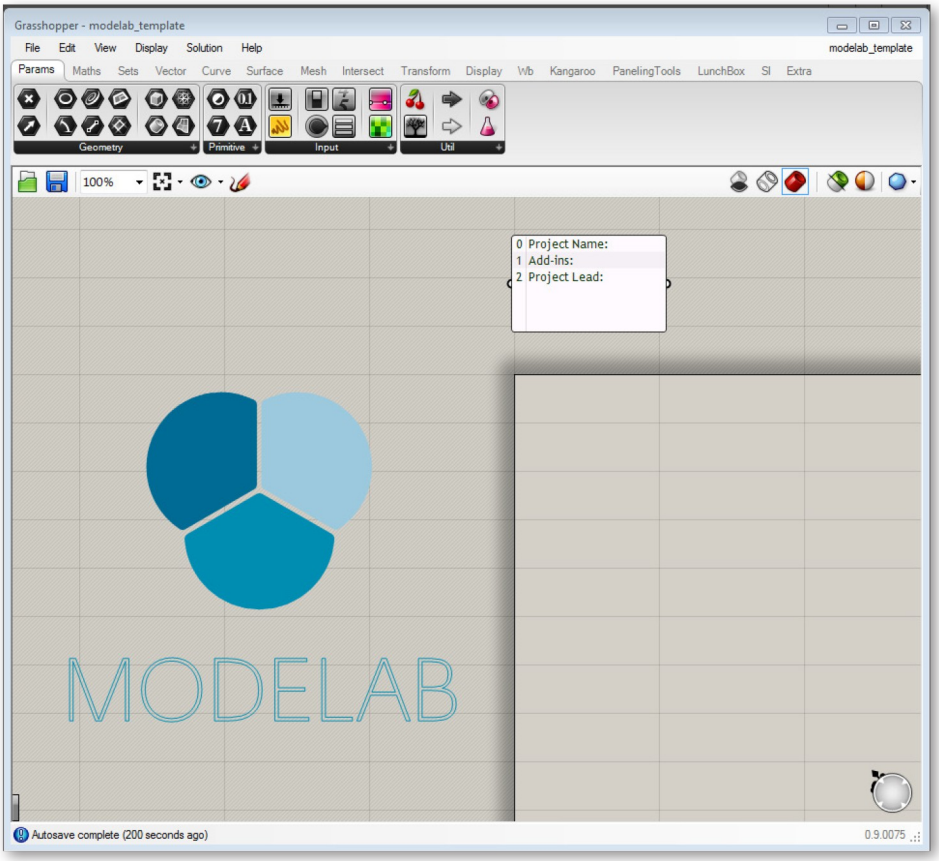
Если ваш Grasshopper-файл ссылается на Rhino-геометрию, Вы должны открыть тот же файл, чтобы дефинишн снова заработал в полной мере. Предотвратить рассинхронизацию ваших Grasshopper- и Rhino-файлов поможет их совместное хранение в одной папке и задание им связанных имён.



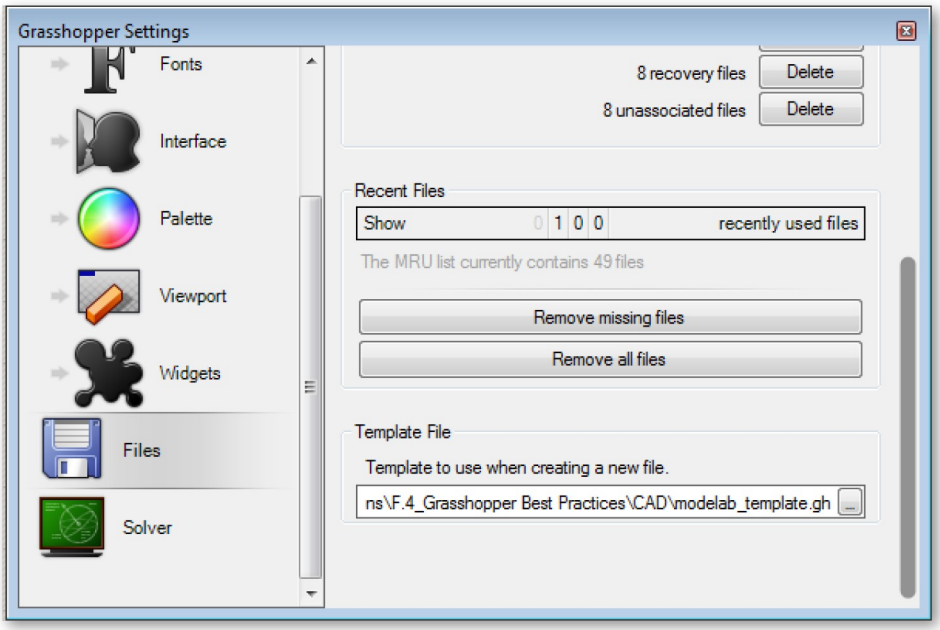
1. Папка Проекта.
2. Rhino-файл.
3. Grasshopper-файл.

1.1.3.8. ШАБЛОНЫ (TEMPLATES)

Создание и определение в качестве файла шаблона вашего Grasshopper-дефинишина — это удобный способ быстрой настройки каждого нового Grasshopper-дефинишина, который Вы создаёте. Шаблоны могут включать такие Grasshopper-компоненты, как Панели (panels) и объекты рисования, применяемые для маркировки дефинишина вашим лейблом.



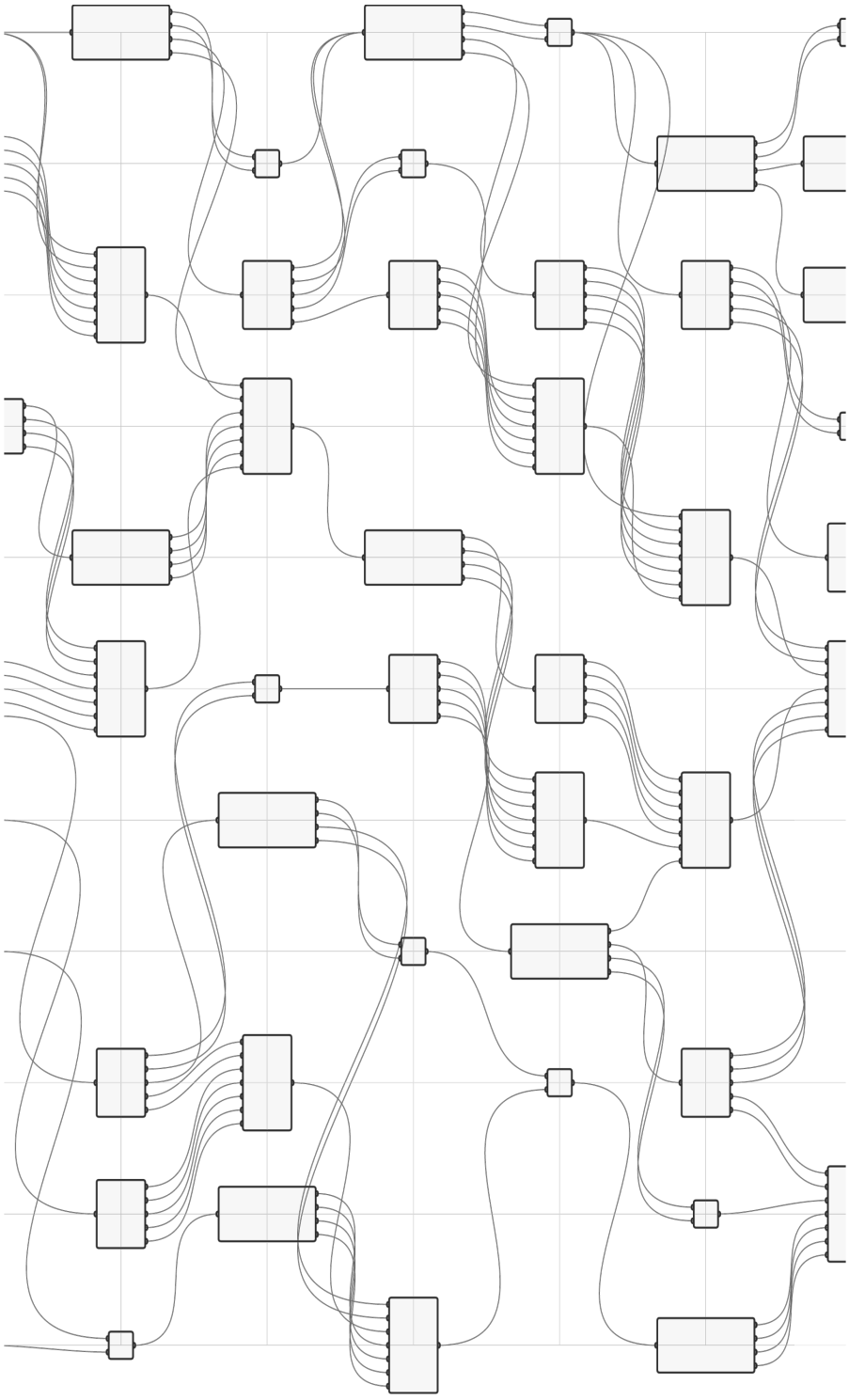
Создайте файл шаблона и сохраните его



Через Меню File/Preferences (Файл/Настройки) загружаем только что созданный файл в качестве файла шаблона (Template File). Теперь ваш шаблон будет использоваться каждый раз при создании нового файла.

1.2. АНАТОМИЯ GRASSHOPPER-ДЕФИНИШИНА

Grasshopper позволяет создавать визуальные программы, называемые дефинишинами (дословно «определениями»). Эти дефинишины составлены из нодов (узлов), соединяемых проводами (связями). Следующая глава познакомит Вас с объектами Грасхопера и способами их взаимодействия для того, чтобы уже, наконец, начать самостоятельное создание дефинишинов.



1.2.1. ТИПЫ ОБЪЕКТОВ GRASSHOPPER

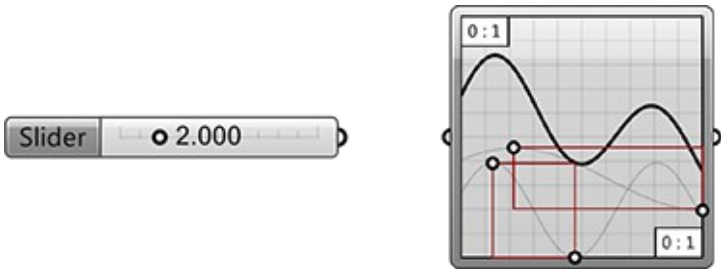
Grasshopper содержит два основных типа пользовательских объектов: параметры (parameters) и компоненты (components). Параметры хрянят данные, в то время, как компоненты выполняют действия, результатом которых являются данные. Самый лучший и основной способ понять работу Грасхопера — это запомнить, что мы будем использовать данные в качестве исходных объектов, чтобы выполнить над ними определённые действия (результатом которых будет получение новых данных, которые мы будем продолжать использовать).

1.2.1.1. ПАРАМЕТРЫ (PARAMETERS)

Параметры используются для хранения данных — чисел, цветов, геометрии и др. — тех, которых мы прогонять через граф нашего дефинишина. Параметры являются объектами-контейнерами, которые, как правило, визуально отображаются в виде небольшой прямоугольной рамки с одним входом и одним выходом. Также Вы должны знать, что все параметры отличаются формой их значка. Все объекты параметров имеют шестигранную границу вокруг иконки.

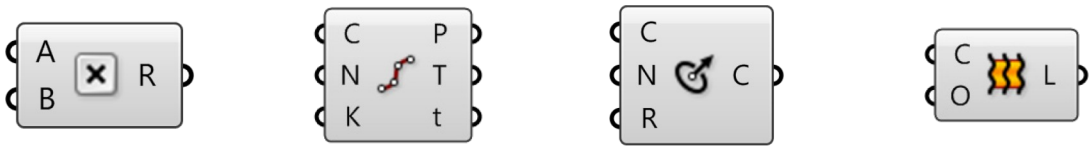
Геометрические параметры могут ссылаться на геометрию из Rhino, или наследовать геометрию из других компонентов. Объекты точек и кривых — оба являются геометрическими параметрами (geometry parameters).

Вводными параметрами (Input parameters) могут быть динамические объекты интерфейса (dynamic interface objects), которые позволяют Вам взаимодействовать с вашим дефинишином. Числовой слайдер (number slider) и переналожение по графу (graph mapper) — оба являются вводными параметрами.



1.2.1.2. КОМПОНЕНТЫ (COMPONENTS)

Компоненты выполняют действия, основанные на получаемых вводных данных. Для различных типов задач существует множество типов компонентов.



1 2 3 4

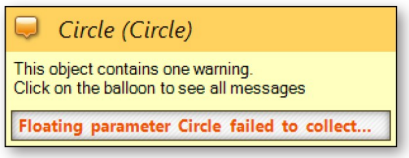
1. Компонент Multiplication (Умножение) представляет собой оператор, который вычисляет произведение двух чисел.
2. Компонент Divide (Разделить) оперирует геометрией, деля кривую (curve) на равные сегменты.
3. Компонент Circle CNR (Окружность по Центру, Нормали, Радиусу) выполняет построение геометрической окружности, основываясь на входных вводных данных: расположение точки центра, направление нормали и величину радиуса.
4. Компонент Loft (Лофтинг) выполняет построение поверхности (surface), основываясь на данных опорных кривых.

1.2.1.3. ЦВЕТА ОБЪЕКТОВ

Мы можем почерпнуть некоторые сведения о состоянии каждого объекта, исходя из его цвета. Давайте взглянем на заданную по-умолчанию схему цветового кодирования Grasshopper.

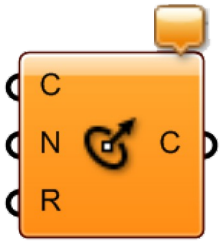
Параметр, не содержащий ни предупреждений, ни ошибок, отображается светло-серым. Этот цвет объектов указывает на то, что всё, что связано с этим параметром, работает должным образом.

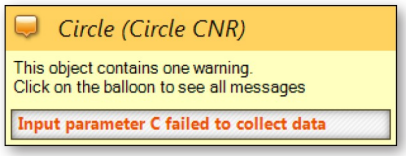
Параметр, который содержит предупреждения (warnings) отображается в виде оранжевой рамки. Любой объект, который не может собрать данные, в Grasshopper-дефинишине рассматривается как подозрительный, поскольку он не способствует решению (solution). Таким образом, все параметры (когда добавляются в качестве нового) показаны оранжевым, чтобы указать Вам, что они пока не содержат каких-либо данных и, таким образом, не имеют никакого функционального влияния на исход решения. По-умолчанию, параметры и компоненты, отображаемые оранжевыми, также сопровождаются небольшим значком подсказки в верхнем правом углу. Если навести на него курсор мыши, эта подсказка раскроет информацию о том, почему компонент выдаёт Вам предупреждение. После того, как параметр начнёт наследовать или определит данные, он становится серым и значок подсказки исчезает.



Если Вы наведёте курсор поверх значка подсказки в правом верхнем углу параметра — подсказка раскроет сообщение о предупреждении.

Компонент — это всегда более сложный объект, поэтому мы обязательно должны понимать и координировать то, что заводим на его входы и ожидаем получить на выходах. Как и компонент, параметр с предупреждением отображается оранжевым. Помните: предупреждение — это не обязательно — плохо , это обычно обозначает лишь то, что Grasshopper предупреждает Вас о потенциальной проблеме в вашем дефинишине.





Этот компонент содержит предупреждение потому, что не имеет достаточно информации для создания окружности.

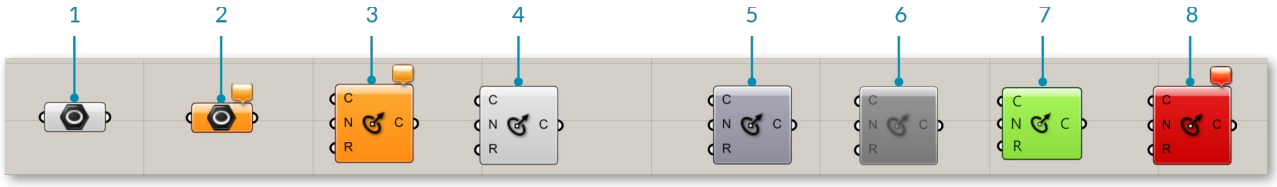
Компонент, не содержащий ни ошибок, ни предупреждений отображается светло-серым.

Компонент, предварительный просмотр которого отключен отображается немного более тёмным серым цветом. Есть два способа отключить отключить предпросмотр компонента. Первый: Просто кликните правой кнопкой мыши на компоненте и переключите кнопку предварительного просмотра. Чтобы отключить одновременно предпросмотр нескольких компонентов: сначала выделите нужные компоненты, а затем переключите режим предпросмотра в состояние «выключен» (человек со связанными глазами) через клик правой кнопкой мыши в любом пустом месте холста.

Отключенный компонент отображается тускло-серым. Чтобы отключить компонент, нужно кликнуть по нему правой кнопкой мыши и переключить кнопку в состояние «выключен» Для отключения нескольких компонентов выделите их, кликните в пустом месте холста правой кнопкой мыши и выберите «Disable» (Отключить). Отключенные компоненты прекращают передачу данных последующим компонентам.

Компонент, который был выделен, отображается светло-зелёным цветом. Если выделенный компонент сгенерировал некую геометрию внутри сцены Rhino, то эта геометрия также будет отображаться зелёной, чтобы дать Вам некую визуальную обратную связь.

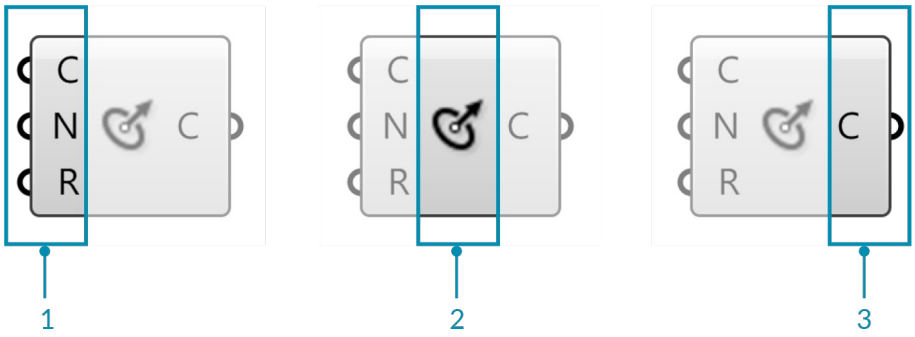
Компонент, который содержит хотя бы одну ошибку отображается красным. Ошибка может исходить как от самого компонента, так и от одного из его входов или выходов.



1. Параметр, не содержащий предупреждений (warnings) или ошибок (erros)
2. Параметр с предупреждениями
3. Компонент с предупреждениями
4. Компонент, не содержащий предупреждений или ошибок
5. Компонент, с отключенным предварительным просмотром
6. Компонент, который был отключен (disabled)
7. Выделенный компонент
8. Компонент с ошибкой

1.2.2. ЧАСТИ GRASSHOPPER-КОМПОНЕНТОВ

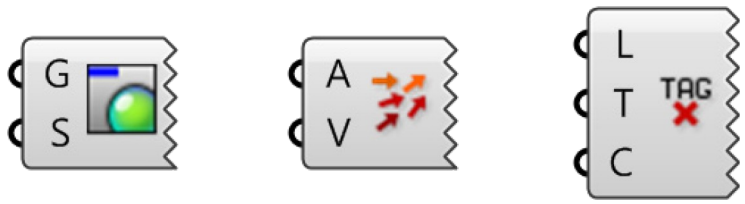
Компоненты — это объекты, которые Вы размещаете на холсте и соединяете вместе связями (wires), чтобы сформировать визуальную программу. Компоненты могут представлять Rhino-геометрию или операции, такие, например, как математические функции. Компоненты имеют входы (inputs) и выходы (outputs).



1. Это три входа (input) параметров компонента Circle CNR (Окружность по Центру, Нормали и Радиусу).
2. Область непосредственно самого компонента Circle CNR.
3. Выход (output) параметра компонента Circle CNR.

Для того, чтобы выполнить свои действия, компоненту требуются данные. И, как правило, это происходит с результатом. Именно поэтому большинство компонентов имеют набор вложенных параметров, называемых «Входы» (Inputs) и «Выходы» (Outputs), соответственно. Входные (вводные) параметры всегда располагаются вдоль левой стороны, а выходные (выводные) параметры по правой стороне.

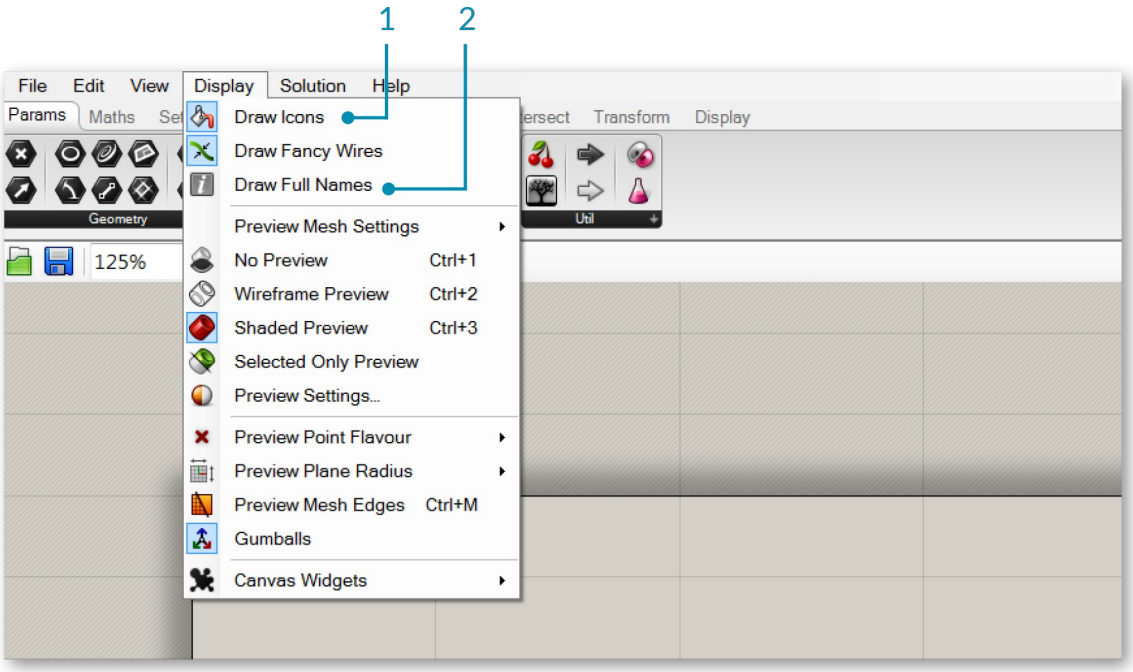
Есть несколько Grasshopper-компонентов, которые имеют входы, но не имеют выходов или наоборот. Если компонент без входа или выхода — он будет показан с соответствующим зазубренным краем.



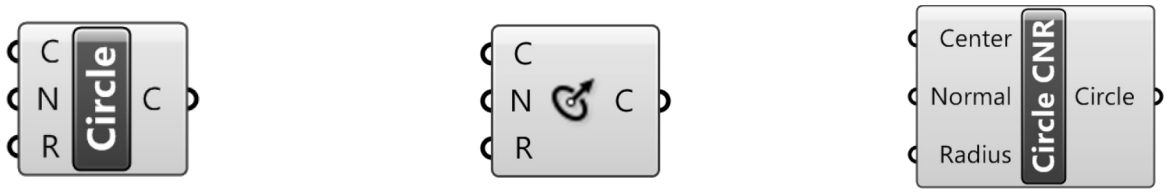
Компонент без входа или выхода имеет зазубренный край.

1.2.2.1. ОТОБРАЖЕНИЕ В ВИДЕ ИКОНКИ (ICON) ИЛИ ТЕКСТОВОЙ МЕТКИ (LABEL)

Каждый Grasshopper -объект имеет уникальный значок. Эти значки отображаются в центральной области объекта и соответствуют иконкам, отображаемым в палитрах компонентов (component palettes). Также объекты могут отображаться с текстовыми метками (text labels). Чтобы переключить режим отображения между иконками или метками выберите пункт “Draw Icons” (Отображать Иконки) в меню Display (Отображение). Вы также можете отметить пункт «Отображать Полные Имена» (Draw Full Names), чтобы имена каждого объекта, а также его входов и выходов отображались полностью.



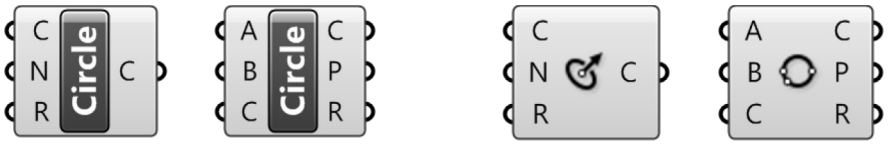
1. Переключение между отображением в виде иконок или значков.
2. Отображать полные имена компонентов, а также их входов и выходов.



1 2 3

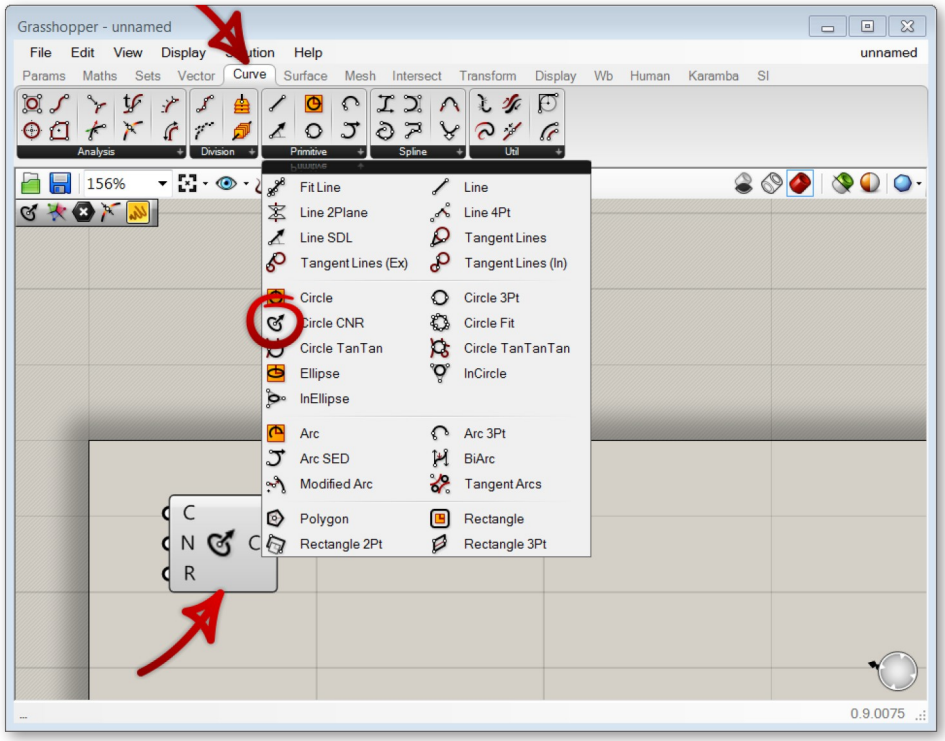
3. Компонент Circle CNR (Окружность по Центру, Нормали и Радиусу) в режиме отображения Меток
4. Компонент Circle CNR в режиме отображения Иконок
5. Компонент Circle CNR в режиме отображения полных имён

Мы рекомендуем, используя отображение в виде иконок, самостоятельно ознакомиться с остальными значками в палитре компонентов, чтобы Вы могли их быстро найти. Также это поможет быстро понять суть этой части дефинишина с первого взгляда. Текстовые метки могут ввести в заблуждение, потому что другие компоненты могут также использовать подобную метку.



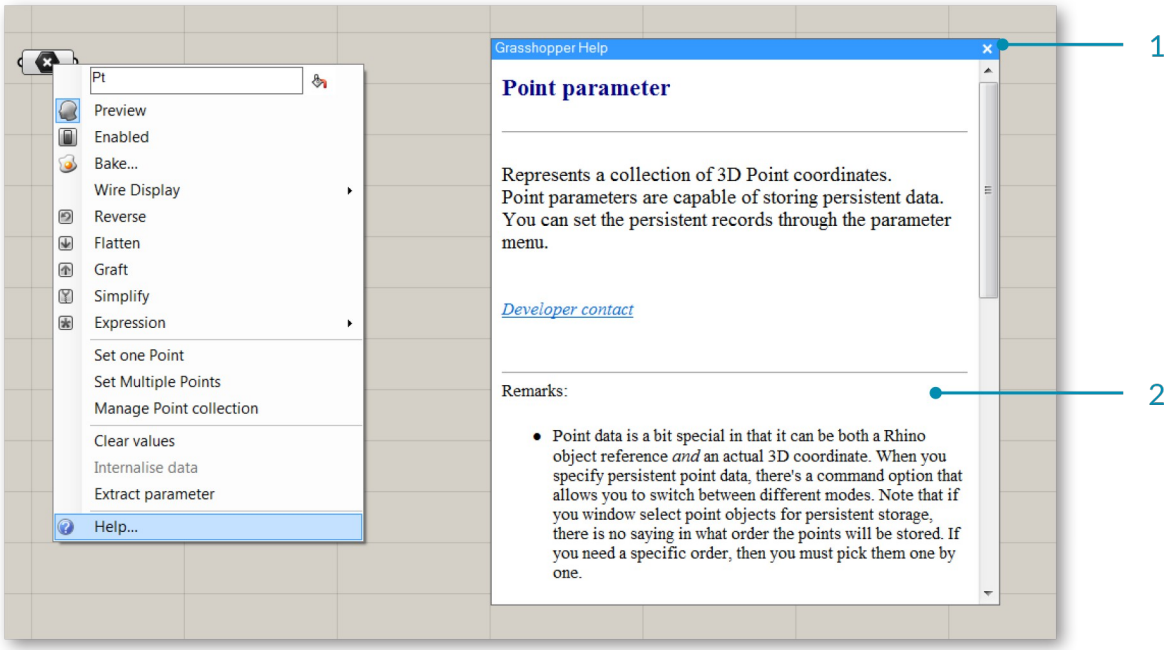
> Компоненты Circle CNR (Окружность по Центру, Нормали и Радиусу) и Circle 3pt (Окружность по Трём Точкам) имеют одинаковые текстовые метки, но разные иконки.

Ещё одна интересная функция, которая поможет Вам разобраться с расположением компонентов в палитрах: Удерживая нажатыми Ctrl + Alt, кликните по уже существующему на холсте компоненту. Это позволит выявить его расположение в палитре.



1.2.2.2. СПРАВКА ПО КОМПОНЕНТУ

Клик правой кнопкой мыши по объекту и выбор пункта “Help” (Справка) из выпадающего контекстного меню откроет окно со справочной информацией по Grasshopper-компоненту. Окно справки содержит более детальное описание объекта, список его входов и выходов, а также замечания.

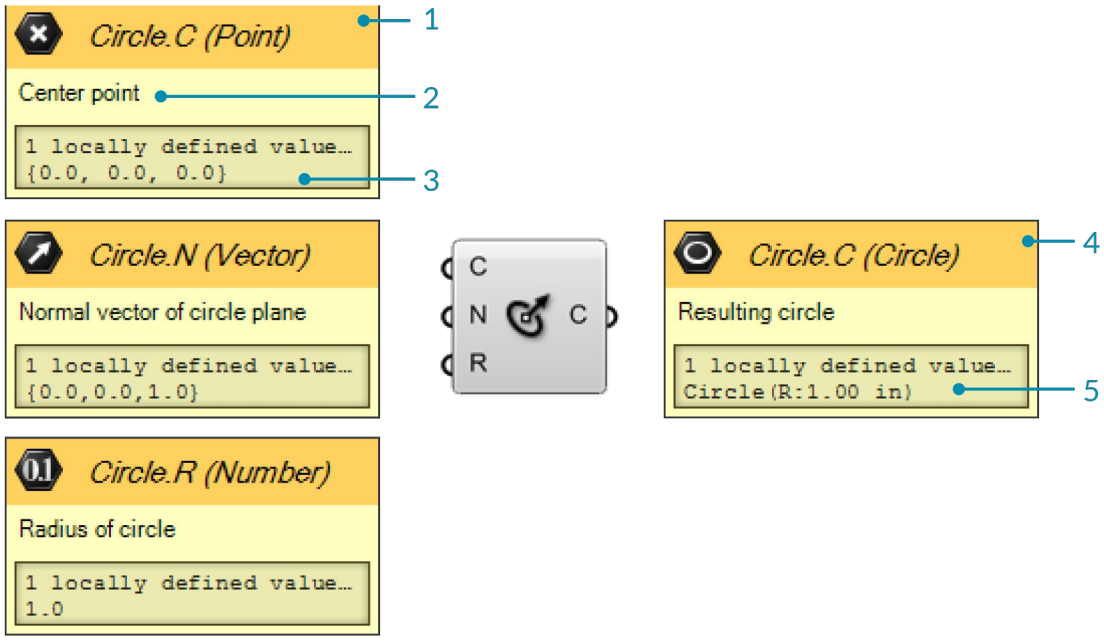


1. Окно справки Grasshopper для параметра Point (Точка)
2. Замечания (Remarks) в окне справки даёт дополнительную информацию о параметре.

[Перевод на русский язык Справки по Grasshopper Вы можете найти на моём сайте RHINO-HELP.COM (Примечание переводчика)]

1.2.2.3. ПОДСКАЗКИ ИНСТРУМЕНТОВ (TOOL TIPS)

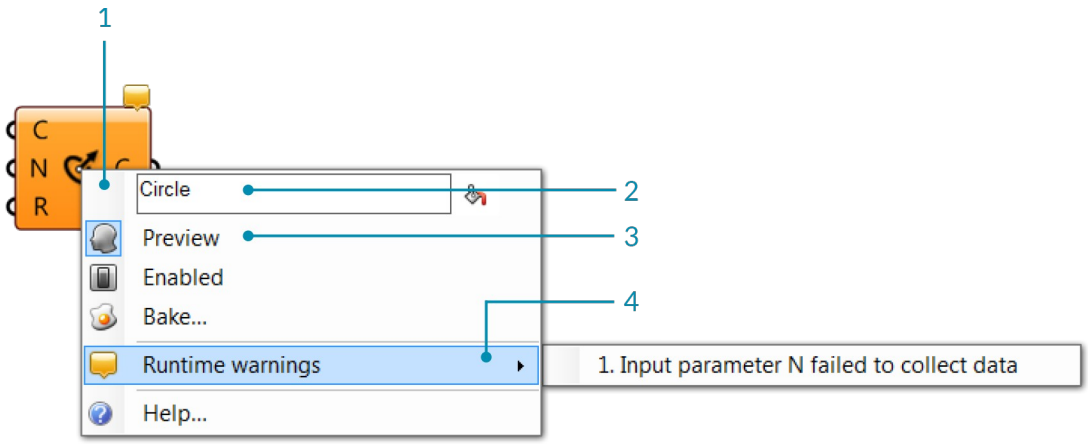
Входы компонентов ожидают получить определённые типы данных, например, компонент может требовать, чтобы Вы подключили на его вход точку (point) или плоскость (plane). При наведении курсора мыши на отдельные части компонента Вы увидите различные подсказки, которые указывают на конкретный тип под-объекта, находящегося в данный момент под курсором мыши. Всплывающие подсказки весьма информативны, поскольку показывают Вам как тип, так и сами данные отдельных параметров.



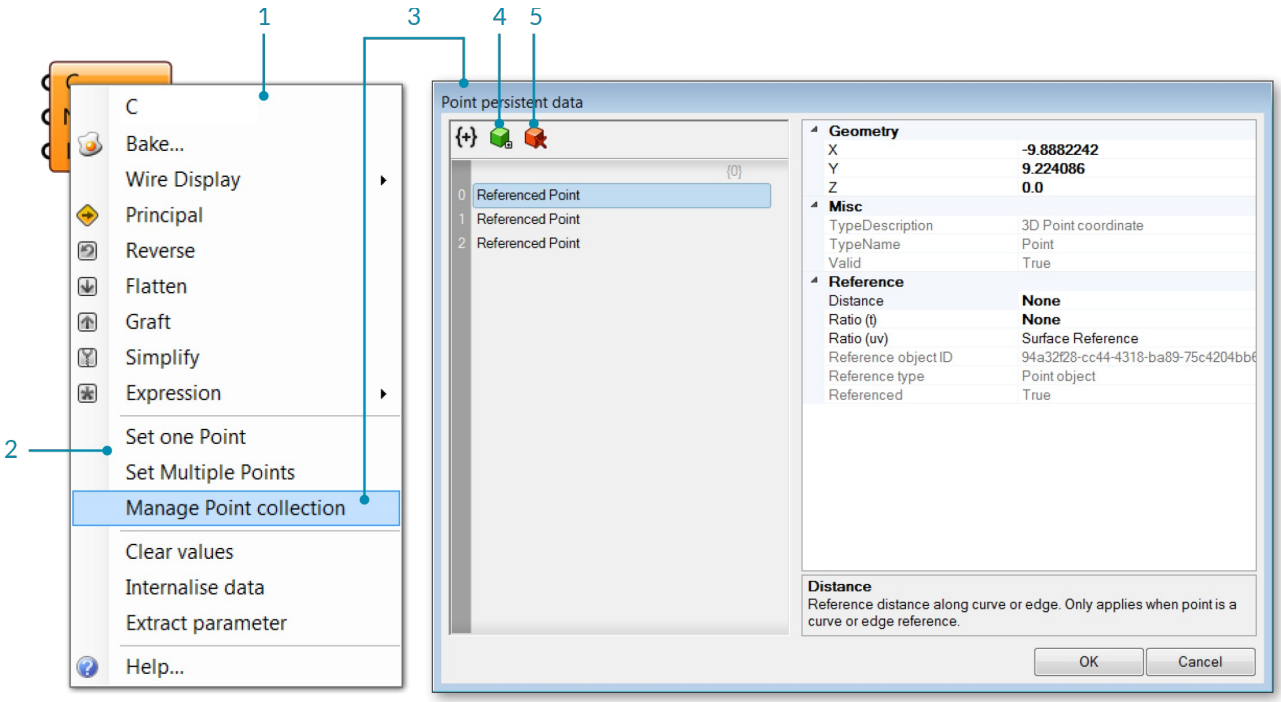
1. Заголовок подсказки отображает иконку типа требуемых входных данных, имя компонента, текстовую метку входа и требуемый тип входных данных в текстовом формате.
2. Текстовое описание входа компонента.
3. Любые значения, определённые на входе: локально или пришедшие по связующему проводу.
4. Заголовок подсказки к выходу предоставляет такую же детальную информацию, что и для входа, но соответствующую выходным данным.
5. Результат действия компонента.

1.2.2.4. ВСПЛЫВАЮЩЕЕ КОНТЕКСТНОЕ МЕНЮ

Все объекты на Холсте имеют свои контекстные меню, предоставляющие индивидуальные настройки и детали. Получить доступ к этому контекстному меню можно по клику правой кнопкой мыши на центральной части каждого компонента. Входы и выходы имеют свои собственные контекстные меню, которые также вызываются кликом правой кнопкой мыши на них.



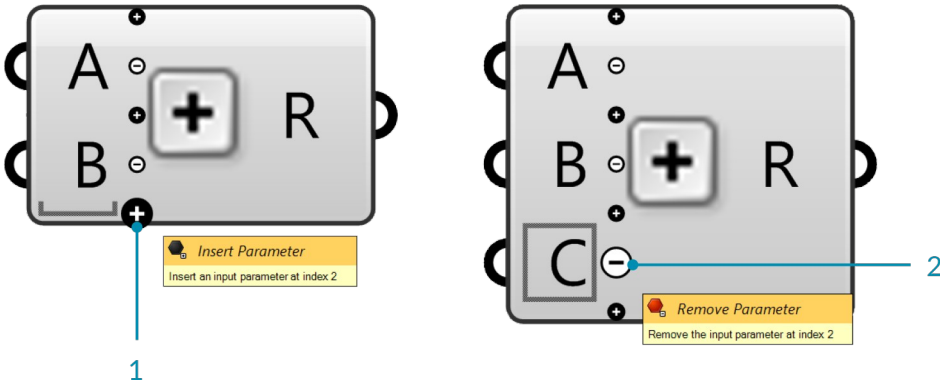
1. Контекстное меню компонента.
2. Редактируемое текстовое поле объекта, содержащее его имя.
3. Флажок предварительного просмотра — отображает и переключает состояние предварительного просмотра, отчего зависит, будет ли объект виден во вьюпортах Rhino и сократит время, необходимое для обсчёта решения (solution).
4. Предупреждения исполняемой среды (Runting warnings) — список предупреждений о препятствиях функционированию компонента.



1. Контекстное меню входа «C».
2. Задать одну или несколько точек («Set one...» или «...multiple points») - позволяет Вам выделить ссылочную геометрию во вьюпорте Rhino.
3. «Manage Point collection» (Управление набором Точек) — открывает диалоговое окно, позволяющее добавлять или исключать точки из набора, а также просматривать информацию по каждой точке.
4. Добавить элемент в набор (Add item).
5. Удалить выделенные (Delete selection).

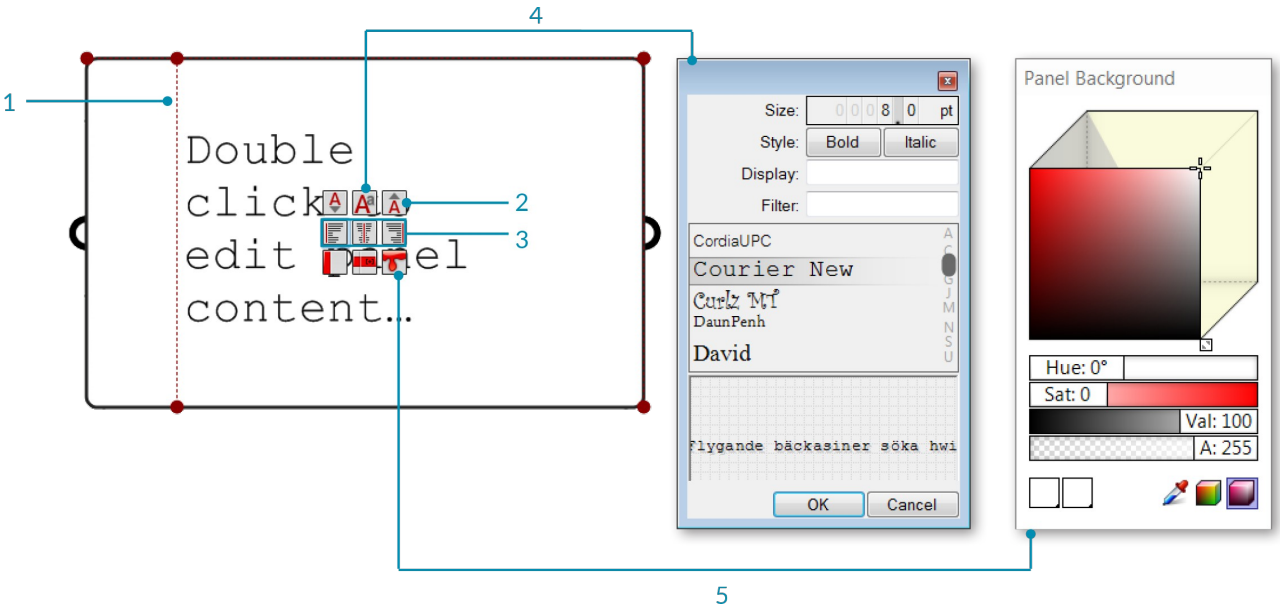
1.2.2.5. МАСШТАБИРУЕМЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС (ZOOMABLE USER INTERFACE)

Некоторые компоненты позволяют увеличить количество входов и выходов посредством ZUI (Масштабируемого Пользовательского Интерфейса). Если увеличить масштаб компонента на холсте, то на нём появится дополнительный набор опций, позволяющий добавить или удалить Входы или Выходы у этого компонента. Компонент Addition (Сложение) позволит Вам добавить Входы , предоставляя дополнительные элементы для операции добавления.



1. Кликните по знаку «+», чтобы добавить Вход (Input).
2. Кликните по знаку «-», чтобы удалить Вход.

Компонент Panel (Панель) также имеет масштабируемый пользовательский интерфейс. Панель подобна стикеру Post-It™. Это позволяет добавить небольшие замечания или пояснения в документ. Вы можете изменить текст через меню или дважды кликнув по поверхности Панели. Панели также могут получать и отображать данные из других источников. Если Вы подключите выход другого компонента или параметра ко входу Панели, то увидите содержимое этого параметра в режиме реального времени. Таким образом можно просматривать любые данные Grasshopper. При увеличении масштаба на Панели появится меню, позволяющее изменить фон, шрифт и другие атрибуты. Также эти опции доступны по клику правой кнопкой мыши на Панели.



1. Перетаскивайте узловые точки для изменения поля Панели.
2. Увеличение или уменьшение размера шрифта содержимого Панели.
3. Изменение выравнивания содержимого Панели.
4. Выбор шрифта содержимого Панели.
5. Выбор цвета фона Панели. Вы можете установить новый цвет фона, задаваемый по-умолчанию, кликнув правой кнопкой мыши на Панели и выбрав "Set Defaut Color" (Задать Цвет По-умолчанию).

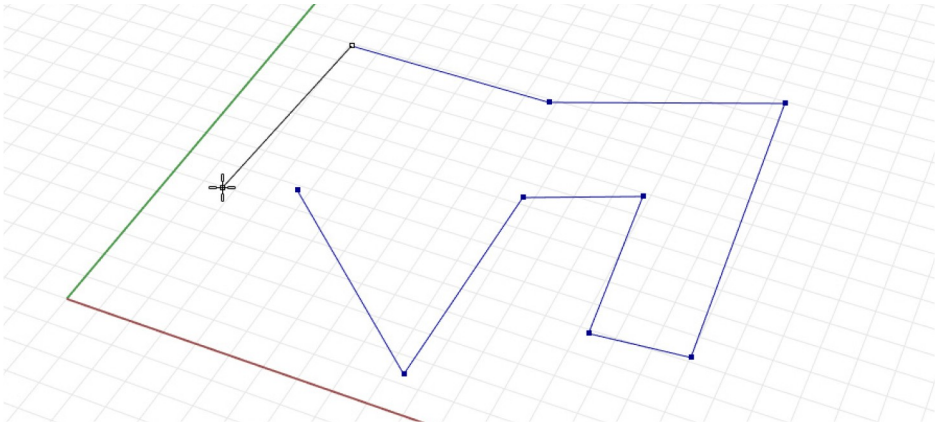
1.2.3. ТИПЫ ДАННЫХ

Большинство параметров позволяет хранить два различных вида данных: Переменные (Volatile) и Постоянные (Persistent). Переменные данные наследуются из одного или более источников и пропадают (то есть собираются заново), когда решатель (solution) запускается снова. Постоянные данные представляют собой данные, специально заданные пользователем.

1.2.3.1. ПОСТОЯННЫЕ ДАННЫЕ (PERSISTENT DATA)

Постоянные данные доступны через меню и в зависимости от параметра имеют разное управление. Параметр Point (Точка), например, позволяет через меню задать одну, либо сразу несколько точек. Но давайте вернёмся на несколько шагов назад и рассмотрим как ведёт себя параметр Point (Точка).

Во время перетаскивания параметра Point (Точка) с Панели Params/Geometry (Параметры/Геометрия) на холст, цвет параметра — оранжевый, указывающий на то, что генерируется предупреждение. Но ничего страшного: это просто предупреждает Вас о том, что параметр пуст (не содержит постоянную запись, а переменные данные собрать ещё не удалось). И таким образом, это ещё не влияет на исход решения. Контекстное меню параметра предоставляет два способа установки постоянных данных: единственное (single), либо несколько сразу (multiple). Кликните правой кнопкой мыши на Параметре, чтобы воспользоваться пунктом «Set Multiple Points» (Задать Несколько Точек) для того, чтобы выбрать несколько точек за один раз. После того, как Вы выберите любой из этих пунктов контекстного меню, окно Grasshopper исчезнет, чтобы Вам было удобнее задавать точки во вьюпортах Rhino.

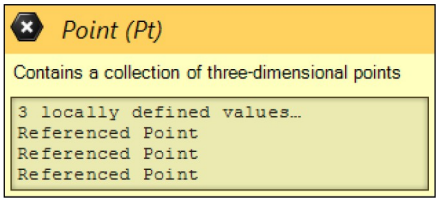


После того, как Вы задали все нужные точки, необходимо нажать клавишу Enter и они станут частью записи постоянных данных Параметра. Это означает, что Параметр больше не является пустым и сменит цвет с оранжевого на серый. (Обратите внимание, что облачко сообщения в верхнем правом углу исчезает, отмечая, что больше предупреждений нет). С этого момента Вы можете

использовать точки, сохранённые в этом Параметре для ввода в любой последующий Параметр дефинишина.

1

2



3

1. Параметр — оранжевый, указывая, что он пока не содержит постоянную запись и пока не удалось собрать переменные данные и, таким образом, этот параметр не влияет на исход решения. Задать постоянные данные любому параметру можно через клик по нему правой кнопкой мыши.
2. Как только параметр получит постоянные данные — цвет компонента сменится с оранжевого на серый.
3. Подсказка параметра Point (Точка) отображает хранимые постоянные данные (коллекцию ссылочных точек).

Исключением здесь являются Выходы, в которых нельзя ни хранить постоянные записи, ни определять набор источников. Выходы явно зависят от того компонента, частью которого являются.

1.2.3.2. ПЕРЕМЕННЫЕ ДАННЫЕ (VOLATILE DATA)

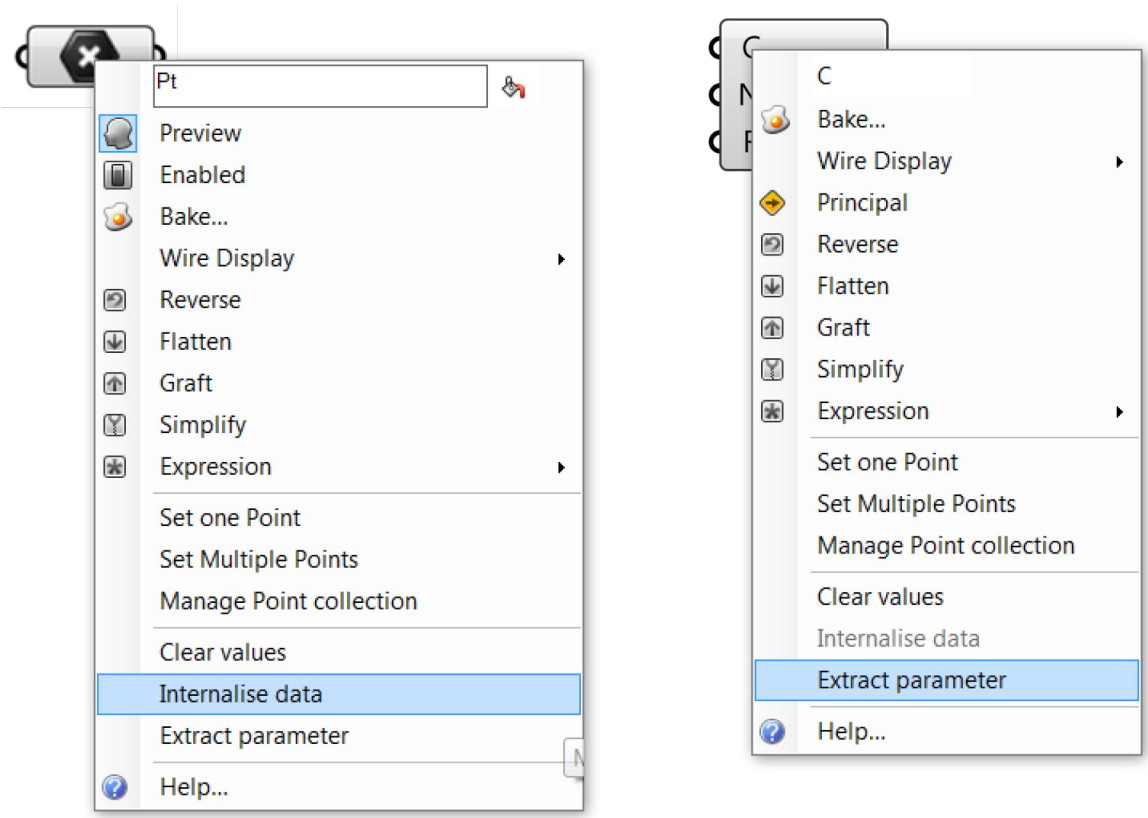
Переменные данные, как и следует из названия, не являются постоянными и будут перезаписаны каждый раз, как решатель (solution) закончит расчёты. Однако, это часто будет инициировано внешними событиями и повлечёт запуск перерасчётов и обновление сцены. В общем, можно сказать, что большинство даннх, полученных «на лету» рассматриваются как переменные.

Как отмечалось ранее, Grasshopper сохраняет данные в Параметрах (в форме переменных или постоянных данных) и использует их в различных Компонентах. Если данные не сохранены в виде постоянной записи в этом параметре, то они должны быть унаследованы от другого параметра. Каждый Параметр (за исключением выходных параметров) определяет, откуда он получает свои данные. И большинство параметров не слишком привередливы. Вы можете подключить числовой параметр (который просто обозначает, что это десятичное число) в источник, требующий целых чисел и он сам позаботится о преобразовании.

Вы можете изменить способ хранения и наследования данных в отдельном параметре или параметре, входящем в состав входа компонента через его контекстное меню. Для того, чтобы изменить способ сохранения данных ссылочной геометрии Rhino в grasshopper-дефинишине на самостоятельное хранение, не зависимое напрямую от конкретного Rhino -файл, нужно через клик правой кнопкой мыши в контекстном меню параметра выбрать «Internalise data» (Перенять данные). Это может быть полезно, когда ваш Grasshopper-дефинишин должен быть независимым от конкретного Rhino-файла. Теперь отключение проводов связи не повлечёт потерю данных. Данные были изменены с переменных на постоянные и больше не будут обновляться.

Если Вы интернализировали данные параметра (Internalise data), являющегося входом компонента, то подключение провода связи от внешнего параметра временно отключит действие интернализированных данных на этом входе. Как только Вы подключили провод связи — стали восприниматься переменные данные. И их значения будут меняться автоматически.

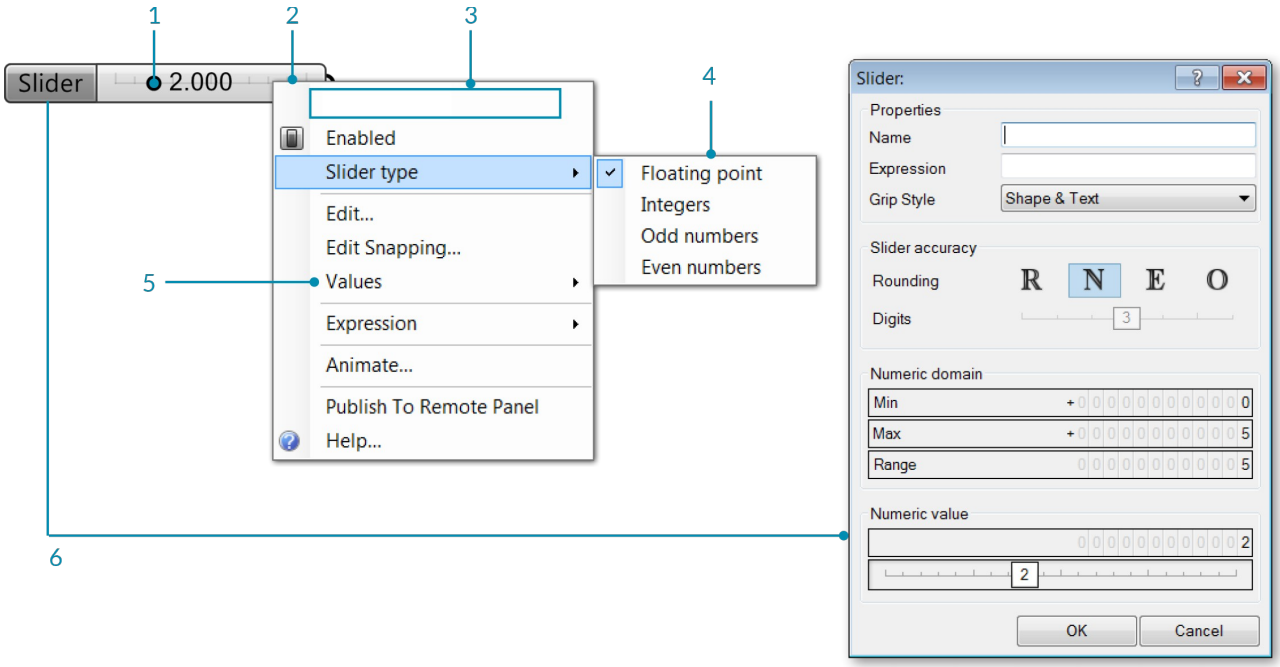
Также Вы можете по клику правой кнопкой мыши на параметре выбрать в его контекстном меню пункт «Extract parameter» (Извлечь параметр) и Grasshopper самостоятельно создаст рядом Параметр, соединённый с данным входом проводом связи и содержащий те же данные, что и извлекаемый параметр.



1.2.3.3. ВВОДНЫЕ ПАРАМЕТРЫ (INPUT PARAMETERS)

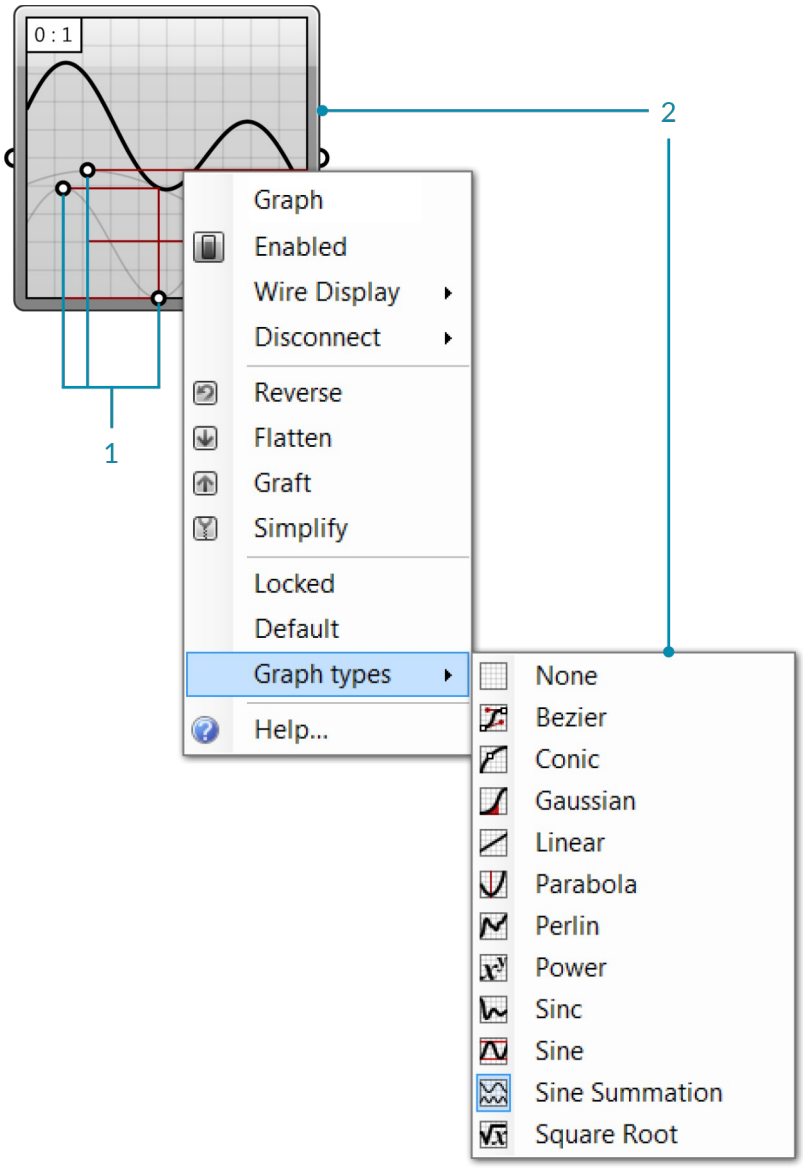
Grasshopper имеет различные Параметры (Parameters), предоставляющие возможность взаимодействовать с данными, поставляя их на входы компонентов (Component inputs) и, таким образом, влияя на результаты работы дефинишина. Поскольку эти параметры изменяют данные, поступающие на вход компонента, то, следовательно, они генерируют Переменные данные (Volatile Data).

Number Slider (Числовой Слайдер). Числовой слайдер является наиболее важным и широко используемым вводным параметром. Он позволяет Вам выводить определённое значение между двумя крайними (экстремумами) с помощью мыши. Слайдеры могут использоваться, чтобы визуально подобрать нужное значение или удачный диапазон значений, глядя на изменения, генерируемые дефинишином.

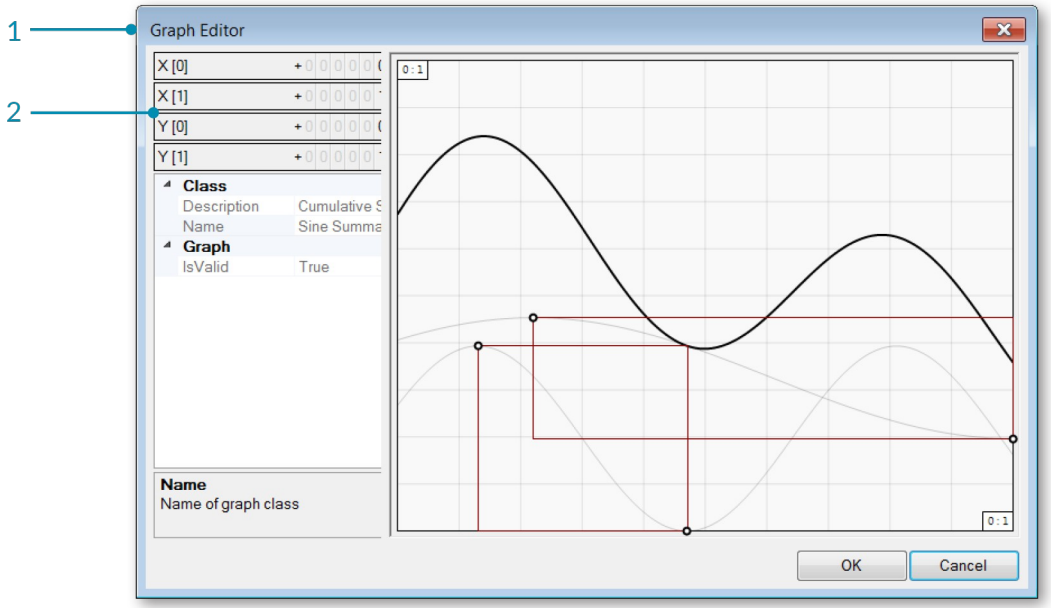


1. Перетаскивание рычага слайдера изменяет значение. Каждый раз, когда Вы это делаете, Grasshopper пересчитывает решение (solution).
2. Кликните правой кнопкой мыши на слайдере, чтобы изменить его имя, тип и значения.
3. Редактируемое текстовое поле имени слайдера.
4. Выбор типа используемых в слайдере чисел.
5. Правка диапазона значений.
6. Двойной клик по части слайдера, содержащей его имя, открывает Редактор Слайдера (Slider Editor).

Graph mapper (Переналожение по Графу). Graph mapper — это двумерный интерфейс, позволяющий изменять числовые значения путём построения графика. Входные значения откладываются вдоль его X-оси, а значения Y-оси там, где X-значения пересекаются с Графом. Это чрезвычайно полезно для модуляции набора значений, ограниченных рычагами интерфейса.

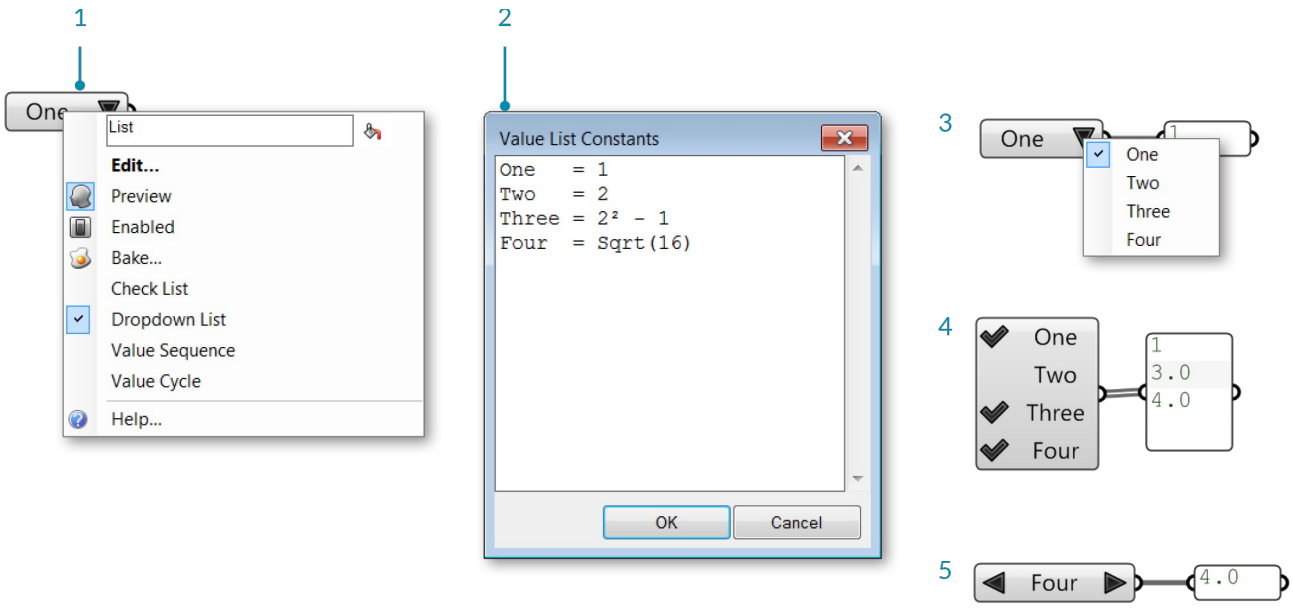


1. Перемещайте рычаги, чтобы редактировать Граф. Каждый раз, когда Вы делаете это, Grasshopper будет пересчитывать решение (solution).
2. Кликните на компоненте Graph mapper (Переналожение по Графу) правой кнопкой мыши и выберите тип Графа в подпункте контекстного меню (Graph type).



1. Двойной клик на Графопостроителе откроет Редактор Графа (Graph Editor).
2. Измените диапазоны (домены) по X и Y.

Value List (Список Значений) Список Значений сохраняет набор значений в виде списка ассоциаций с текстовыми метками, связанных посредством знака равенства. Это особенно полезно, если Вы хотите иметь список из нескольких вариантов значений, обозначенных надписями со смыслом, в то же время выдающих конкретные выходные значения.



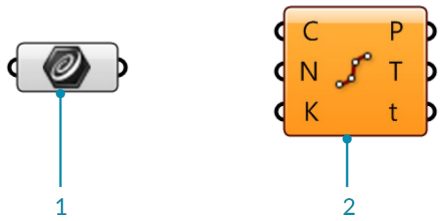
1. Кликните правой кнопкой мыши по компоненту Value List (Список Значений) и выберите нужную опцию из контекстного меню.
2. Дважды кликните по компоненту Список Значений, чтобы открылся редактор и можно было добавить и изменить значения.
3. В режиме Выпадающего Списка (Dropdown List) нужно кликнуть по стрелке вниз, чтобы выбрать одно из значений. Решение будет пересчитываться каждый раз, как только Вы смените значение.
4. В режиме Контрольного Списка (Check List) нужно кликать по каждому следующему значению, чтобы отметить его. На выход компонент будет выдавать только отмеченные значения.
5. В режимах Последовательность Значений (Value Sequence) и Цикл Значений (Value Cycle) клик по стрелке влево или вправо будет перебирать значения из списка по кругу.

1.2.4. СВЯЗЬ МЕЖДУ КОМПОНЕНТАМИ

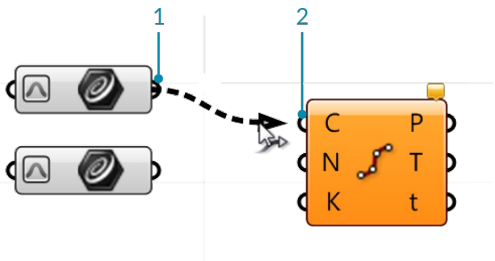
Если данные не сохранены напрямую в параметре в виде постоянной записи, то они должны быть унаследованы из другого параметра. От одного параметра к другому данные передаются по проводам (wires). Вы можете воспринимать их буквально, как электрические провода, передающие импульсы данных от одного объекта к следующему.

1.2.4.1. УПРАВЛЕНИЕ СОЕДИНЕНИЯМИ

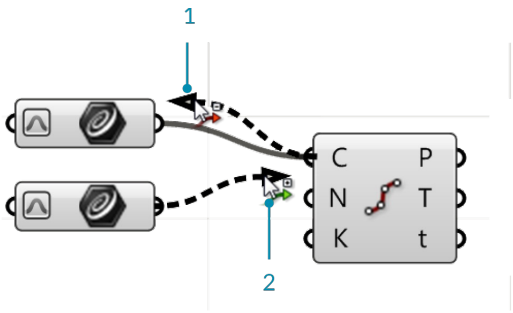
Чтобы соединить компоненты, кликните вблизи выступа на выходе параметра или компонента, и, не отпуская, перетащите мышью на вход следующего. За указателем мыши потянется соединительный провод. Как только указатель мыши окажется над потенциально целевым входом — провод будет подсоединяться и «твердеть». Пока Вы не отпустили кнопку мыши — это ещё не постоянное соединение. Не имеет принципиального значения, в какую сторону протягивать соединение: слева-направо или справа-налево.



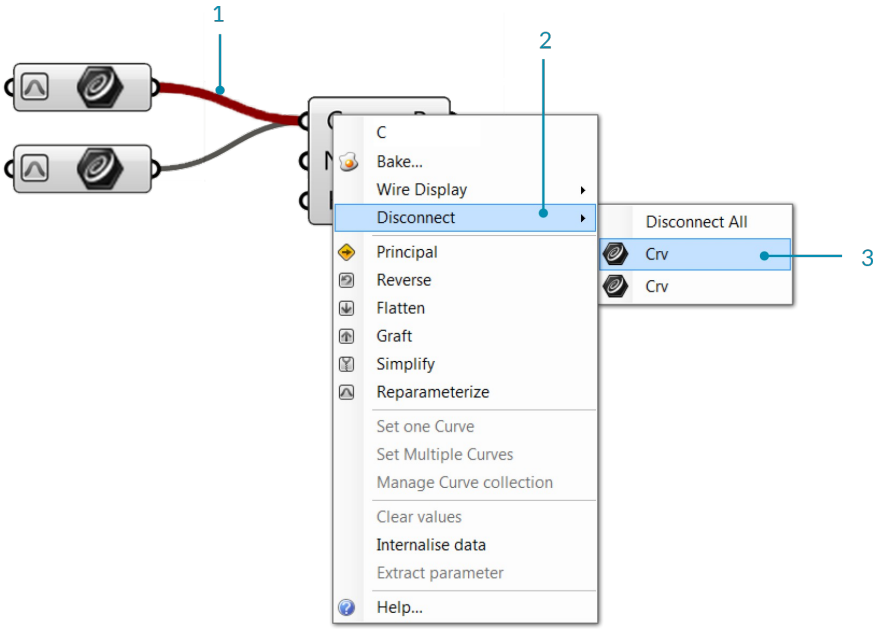
1. Компонент Divide Curve (Разделить Кривую) делит кривую на сегменты равной длины.
2. Параметр Curve (Кривая) позволяет выбрать ссылочную Rhino-геометрию через клик правой кнопкой мыши и выбор пункта «Set One Curve» (Выбрать Одну Кривую).



Кликните и перетащите левой кнопкой мыши, чтобы протянуть связь от выхода одного объекта (1.) ко входу другого (2.).



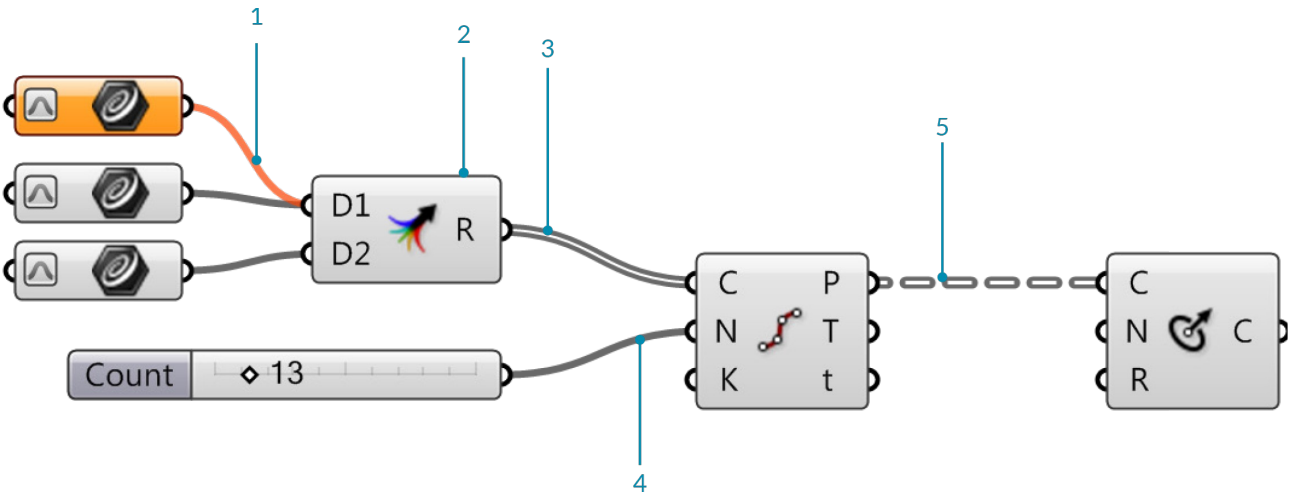
1. Если во время перетаскивания удерживать нажатой клавишу CONTROL, то курсор сменит цвет на красный и целевой источник будет исключен из списков источников.
2. По-умолчанию, новое соединение будет убирать предыдущее. Удерживайте нажатой клавишу SHIFT при перетаскивании соединительного провода, чтобы задать несколько источников одному входу. Курсор сменит цвет на зелёный, показывая, что осуществляется добавление.



1. Также Вы можете отключить провод через всплывающее контекстное меню с помощью клика правой кнопкой мыши на рычаге входа или выхода и выбора пункта «Disconnect» (Отсоединить).
2. Если имеется несколько подключений, то выберите из списка то, которое хотите отключить.
3. Когда курсор мыши будет находиться над элементом в списке, его провод будет подсвечиваться красным.

1.2.4.2. FANCY WIRES (ПОНЯТНЫЕ ПРОВОДА)

Провода представляют собой соединения, по которым течёт поток данных вдоль всей протяжённости графа нашего дефинишина. Grasshopper также может дать нам подсказки на то, что же именно течёт по этим проводам. По-умолчанию, эти «понятные провода» могут быть выключены, так что, возможно, предварительно их придётся включить, чтобы просматривать различные типы линий для разных проводов. Чтобы сделать это — просто переключите в нажатое состояние кнопку “Draw Fancy Wires” (Отрисовывать Понятные Провода) на вкладке View (Вид) [или Display (Отображение)] в строке главного меню. «Понятные провода» поведают Вам много информации о том, какого типа информация течёт по проводу от одного компонента к другому.

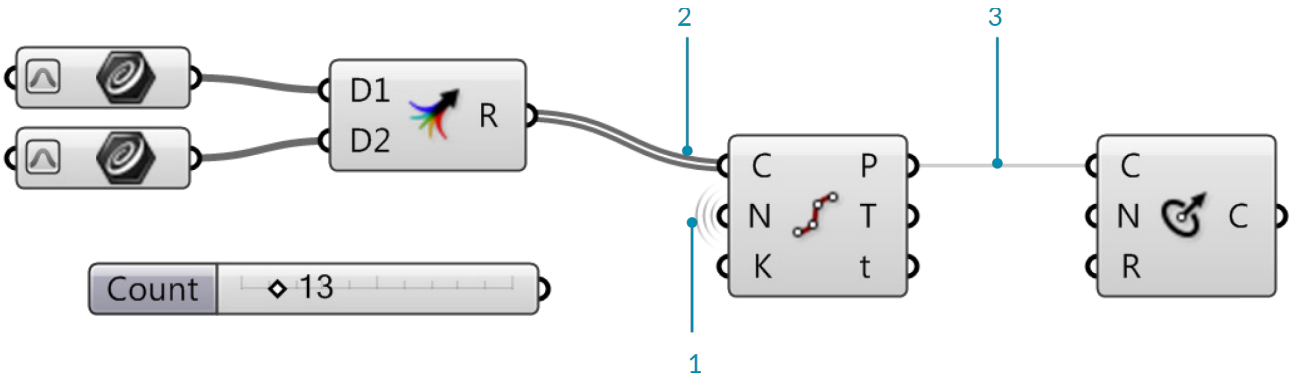


1. Пустой Элемент – Оранжевый тип провода указывает на то, что информация не была передана. Этот параметр генерирует предупреждение, поскольку не содержит никаких данных, и, таким образом, нет информации, передаваемой по каналу связи.
2. Компонент Merge (Слияние) — альтернатива подсоединения более чем одного ресурса к единственному входу.
3. Список (List) – В случае, если информация, текущая из компонента содержит перечень сведений, то такой тип провода будет отображаться двойной серой линией.
4. Единственный Элемент (Single Item) – Если из любого параметра исходят данные в виде одного-единственного элемента, то такой провод будет показан сплошной серой линией.
5. Дерево (Tree) – Информация передаётся между компонентами в виде древовидной структуры и отображается в виде двойной пунктирной линии.

1.2.4.3. ОТОБРАЖЕНИЕ СВЯЗЕЙ (WIRE DISPLAY)

Если Вы уже провели какое-то время в работе над Grasshopper-дефинишином, то, скорее всего, уже поняли, что на холсте довольно быстро образуется нагромождение проводов. К счастью, у нас есть средство для управления отображением каждого входа на компоненте.

Есть три способа отображения проводов: Default Display (Отображение По-умолчанию), Faint Display (Тусклое Отображение) и Hidden Display (Скрытое Отображение). Чтобы сменить способ отображения, просто кликните правой кнопкой мыши на нужном входе компонента и выберите один из доступных способов отображения из контекстного меню.



1. Hidden Display (Скрытое Отображение) – Когда выбрано скрытое отображение, провод будет полностью «невидимым». Данные от источника ко входу параметра передаются по «беспроводной сети». Если Вы выделите ресурсный или целевой компонент, появится зелёный провод, чтобы показать Вам, какие компоненты соединены друг с другом. После того, как выделение с компонента будет снято, провода снова скроются.
2. Default Display (Отображение По-умолчанию) – По-умолчанию отображаются все соединения. (В более старых версиях, если включена опция fancy wires (понятные провода)).
3. Faint Display (Тусклое Отображение) – Будет показывать провода связей очень тонкими и полупрозрачными линиями. Тусклое (Faint) и Скрытое (Hidden) отображение проводов может быть очень полезно, когда у Вас есть множество проводов из источников, поступающих на единственный вход.

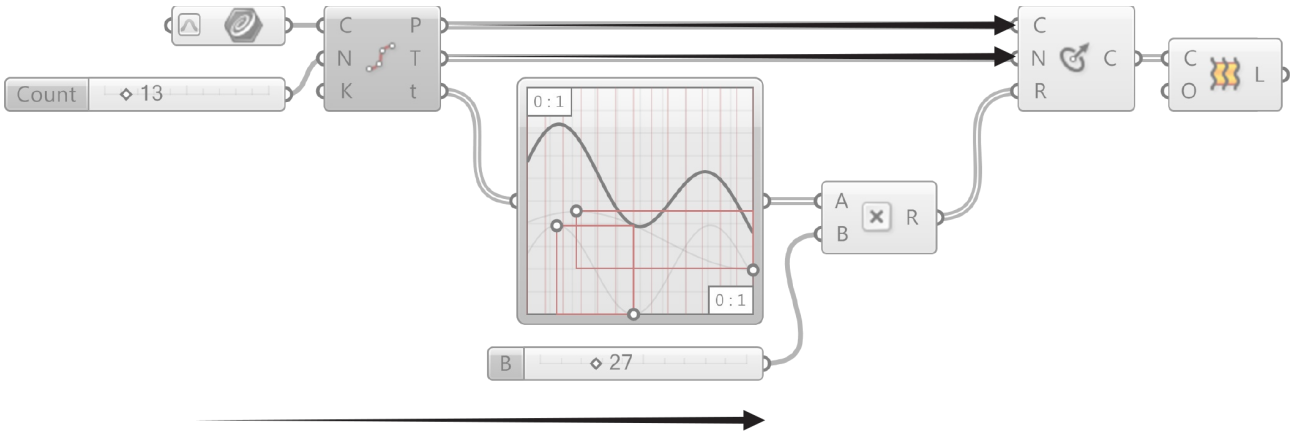
1.2.5. GRASSHOPPER-ДЕФИНИШИН

Загрузить файл примера, сопровождающий данный раздел.

Grasshopper-Дефинишин представляет собой Программный Поток (Program Flow), определеляющий, с чего начать выполнение программы, что делать в середине и как узнать, что выполнение программы завершено.

1.2.5.1. ПРОГРАММНЫЙ ПОТОК

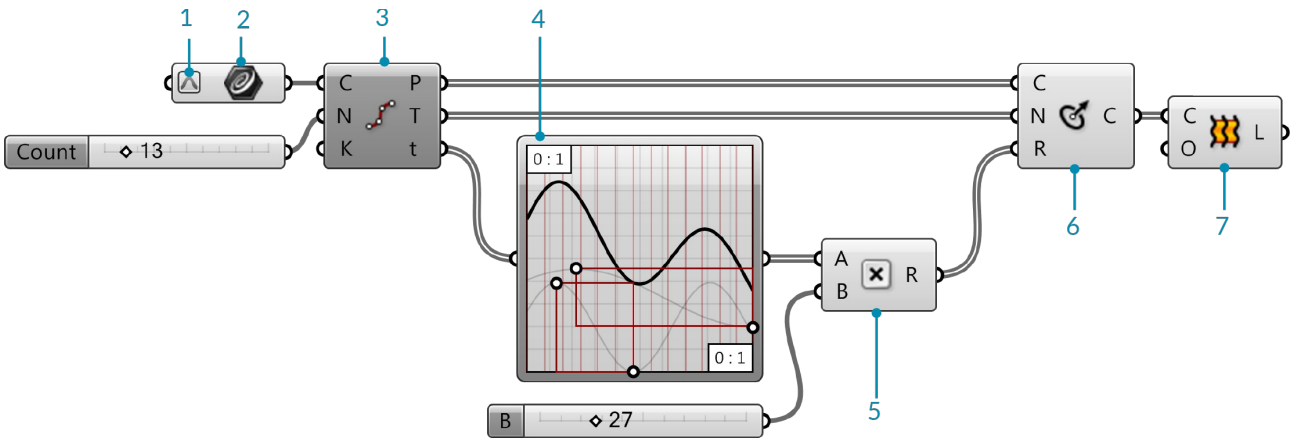
Визуально Grasshopper-команда выполняется слева-направо. Чтение графа, ориентируясь по проводам связей от истока до устья обеспечивает понимание Программного Потока.



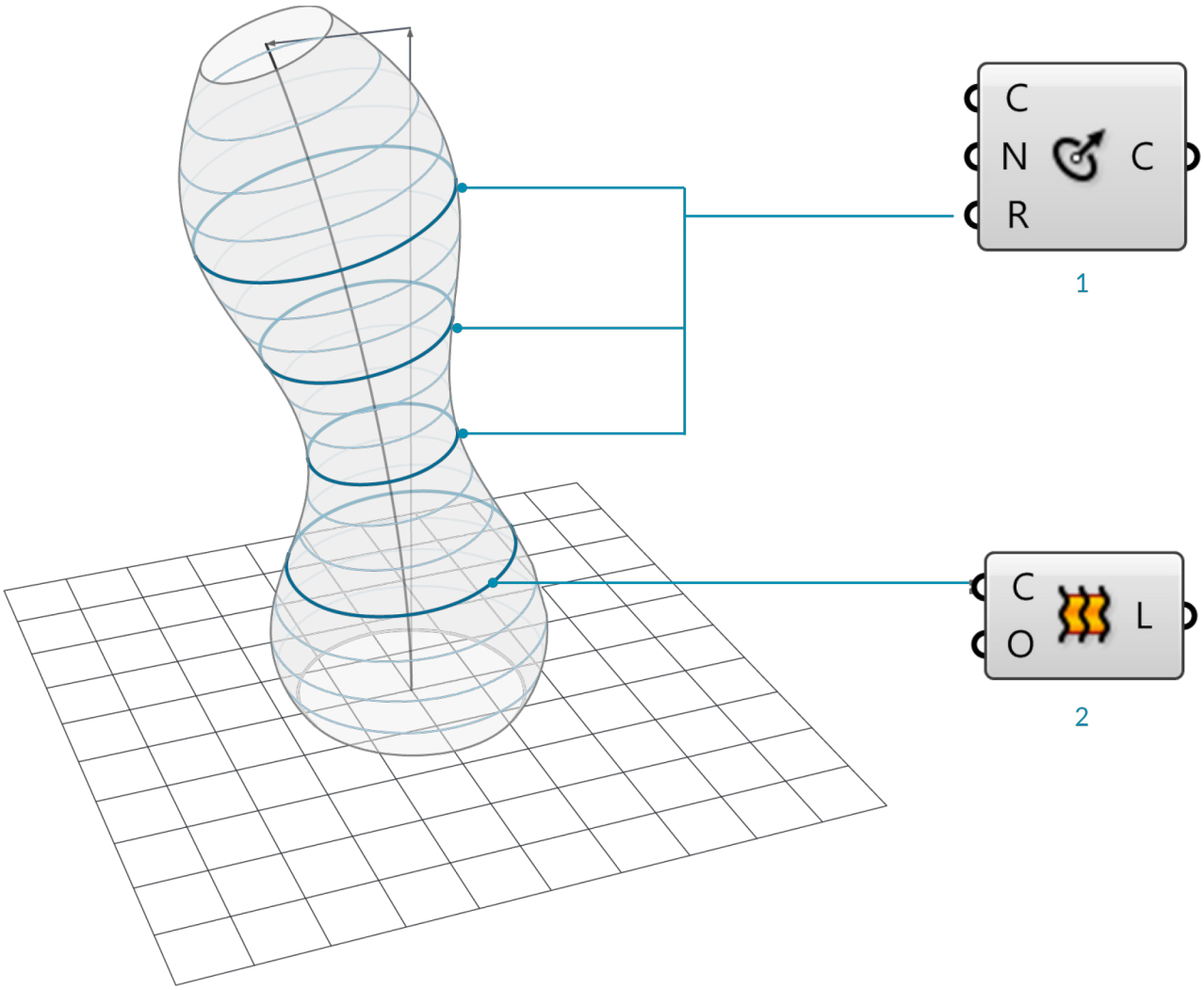
Направленность течения данных происходит слева-направо.

1.2.5.2. ЛОГИЧЕСКИЙ ПУТЬ (LOGICAL PATH)

Все объекты и провода, соединяющие эти объекты представляют собой логический график (граф) вашей программы. Этот график отображает поток данных, отдающихся в подчинение какому-то входу от присоединённого к нему связью выхода. Каждое изменение графа повлечёт за собой обновление всего последующего потока и всех связанных с ним объектов, даже, если будут привнесены некорректные данные.



1. Репараметризация домена кривой на диапазон между 0.0 и 1.0.
2. Ссылочная кривая из Rhino.
3. Деление кривой на 13 равных сегментов.
4. Запуск значений параметра каждой точки деления кривой через граф.
5. Умножение каждого значения на 27.
6. Построение окружности на каждой точке деления вдоль кривой по нормали к вектору касательной в каждой точке радиусом, определённым значением параметра (t), модифицированного графопостроителем (graph mapper) и умноженного на 27.
7. Поверхность, построенная лофтингом окружностей.



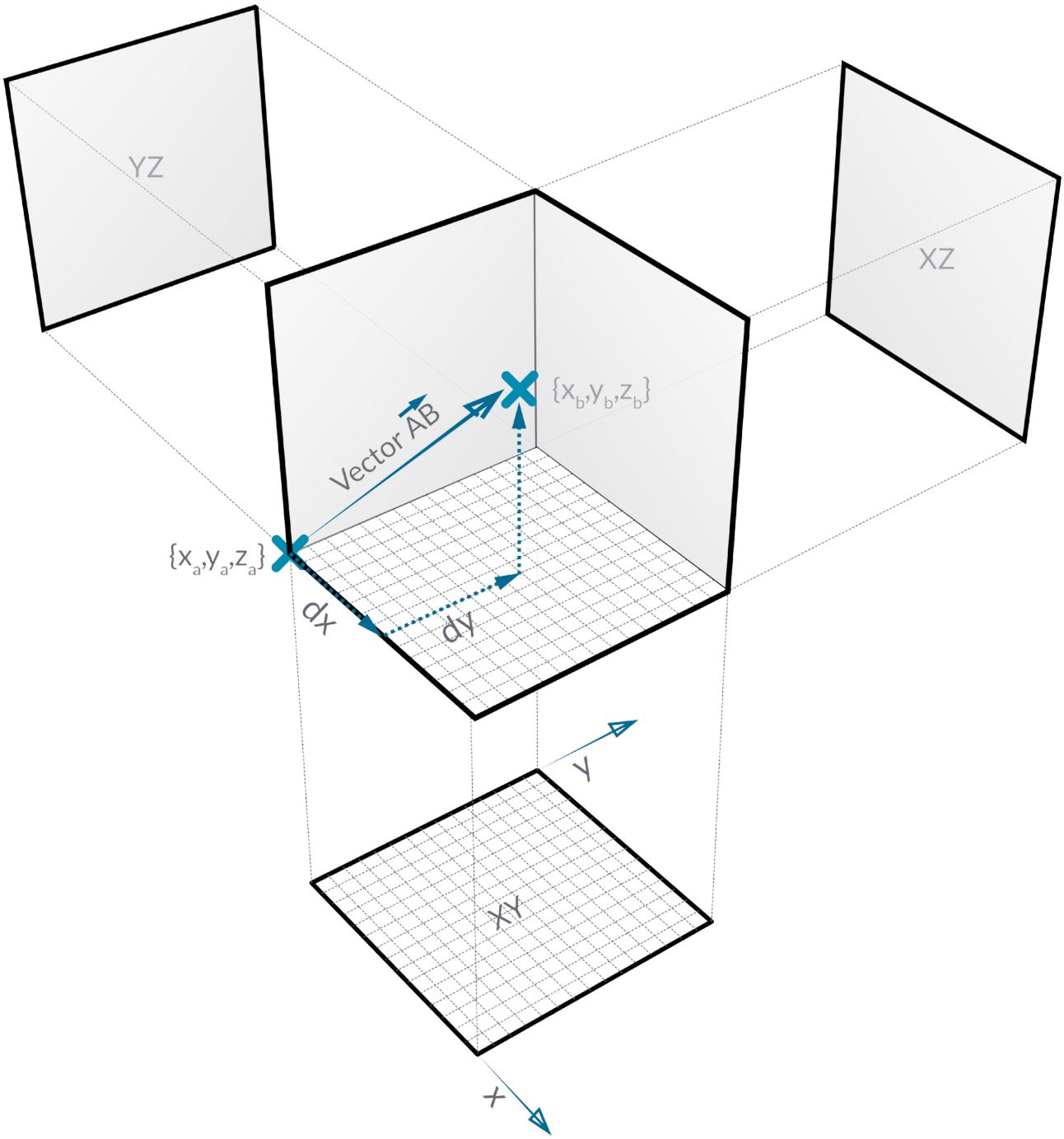
1. Вариабельные радиусы окружностей.
2. Лофтинг между окружностями.

1.3. СТРОИТЕЛЬНЫЕ БЛОКИ АЛГОРИТМОВ

Эта глава познакомит Вас с основными геометрическими и математическими понятиями, а также с тем, как они реализованы в Grasshopper и как он позволяет ими манипулировать.

1.3.1. Точки (Points), Плоскости (Planes) и Векторы (Vectors)

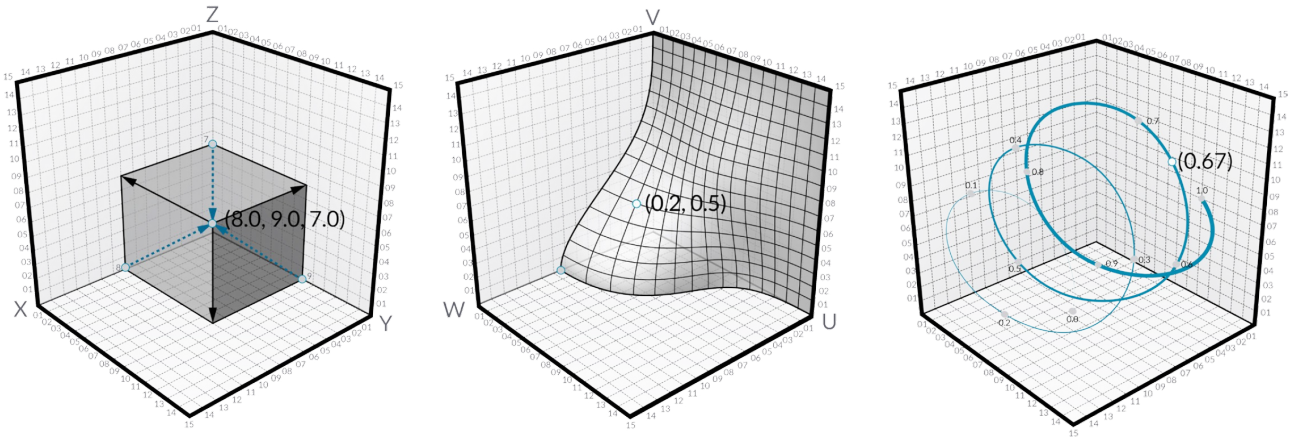
Всё начинается с точек. А точка — это ничего более, чем одно или несколько значений, называемых координатами. Количество значений координат соответствует числу измерений пространства в котором они находятся. Точки, плоскости и векторы — это база для создания и трансформации геометрии в Grasshopper.



1.3.1.1 ТОЧКИ (POINTS)

Точки в 3D-пространстве имеют координаты, обычно отображаемые в формате [x,y,z]. Точки в 2D-пространстве имеют лишь 2 координаты, которые называют либо [x,y], либо [u,v], в зависимости от того, о каком двумерном пространстве мы говорим. 2D-параметр пространства ограничен конечностью поверхности. Тем не менее, она непрерывна, то есть, гипотетически, существует бесконечное количество точек на поверхности. Но максимальное расстояние между любыми из этих точек очень ограничено. Параметр 2D-координаты работает только если числа координаты не превышают определённый диапазон. В приведённом в качестве примера диапазон задан между 0.0 и 1.0 в обоих [u] и [v] направлениях, но это могла быть любая конечная область. Точка с координатами [1.5, 0.6] была бы где-нибудь вне поверхности и, таким образом, была бы некорректной.

Поскольку поверхность, определённая этим конкретным параметром 2D-пространства находится в постоянном глобальном 3D-пространстве (world space), мы всегда можем перевести параметрические координаты в глобальные 3D-координаты. Например, точка [0.2, 0.5] на поверхности примера то же самое, что и точка [1.8, 2.0, 4.1] в глобальных координатах. Как только мы трансформируем или деформируем поверхность, 3D-координаты, которые соответствуют [0.2, 0.5] изменятся.

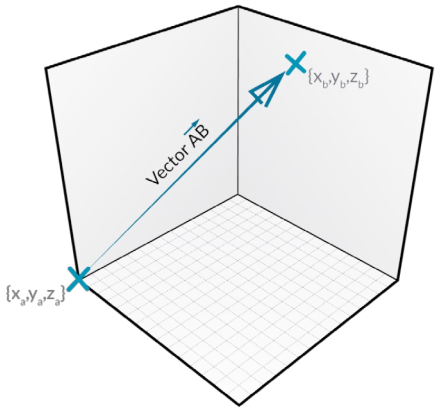


Если будет трудно понять эту концепцию, то может помочь думать о себе и о своём положении в пространстве. Мы склонны использовать локальные системы координат для описания нашего местонахождения «я сижу в кинотеатре на 3-м месте 7-го ряда», «я на заднем сиденье». Если Вы на дороге в автомобиле, то ваша позиция в глобальных координатах всё время меняется, даже если Вы всё время находитесь «на заднем сиденье» в локальных координатах.

1.3.1.2. ВЕКТОРЫ (VECTORS)

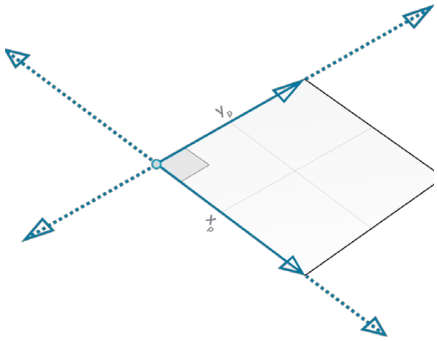
Вектор — это геометрическая величина, описывающая Направление (Direction) и Величину (Magnitude). Векторы являются абстрактными, то есть они представляют собой количество, величину, а не геометрический элемент.

Векторы неотличимы от точек. То есть, оба они состоят из списка трёх чисел, так что нет абсолютно никакой возможности определить, представляет ли определённый перечень собой точку или вектор. Существует лишь параметрическая разница: точки являются абсолютными, а векторы — относительны. Когда мы рассматриваем список трёх сдвоенных чисел как точку, то эти числа представляют определённые координаты в пространстве, а когда мы рассматриваем их в качестве вектора, то они представляют собой направление. Вектор — это стрелка в пространстве, которая начинается всегда в начале глобальных координат (0.0, 0.0, 0.0) и заканчивается в указанной координате.



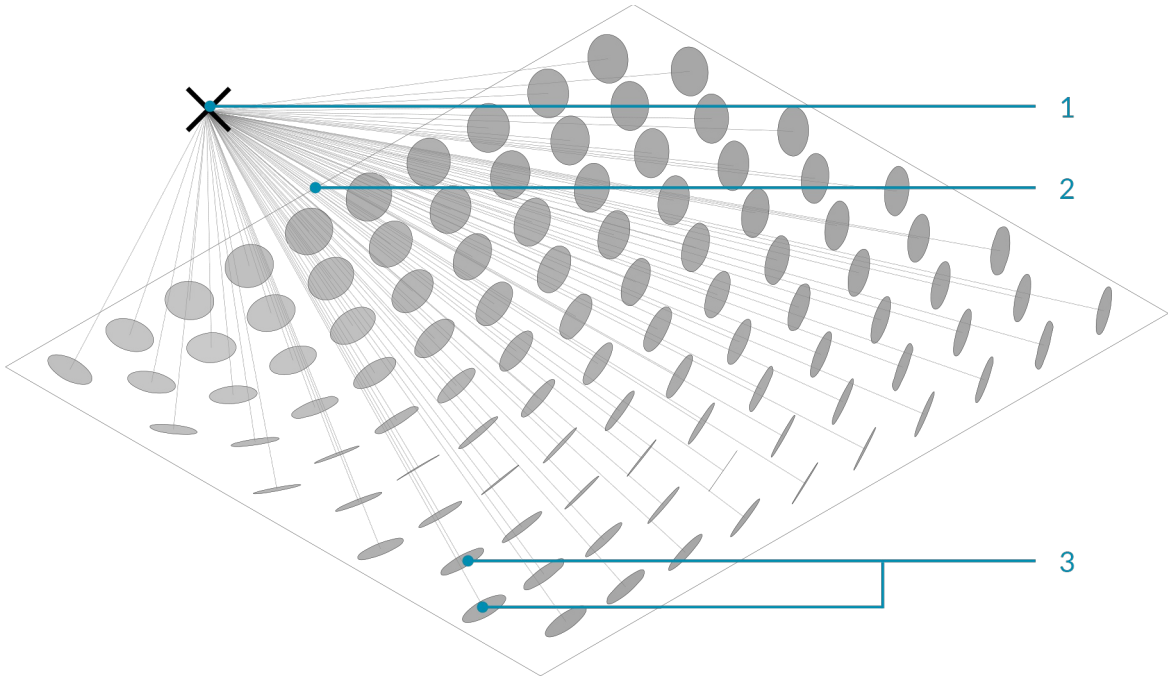
1.3.1.3. ПЛОСКОСТИ (PLANES)

Плоскости «плоские» и простираются бесконечно в двух направлениях, определяющих локальную систему координат. Плоскости не являются истинными объектами Rhino, они используются лишь для определения системы координат в глобальном 3D-пространстве. На самом деле лучше думать о плоскостях в виде векторов. Они являются лишь математическими конструкциями.



1.3.2. Работа с аттракторами (Attractors)

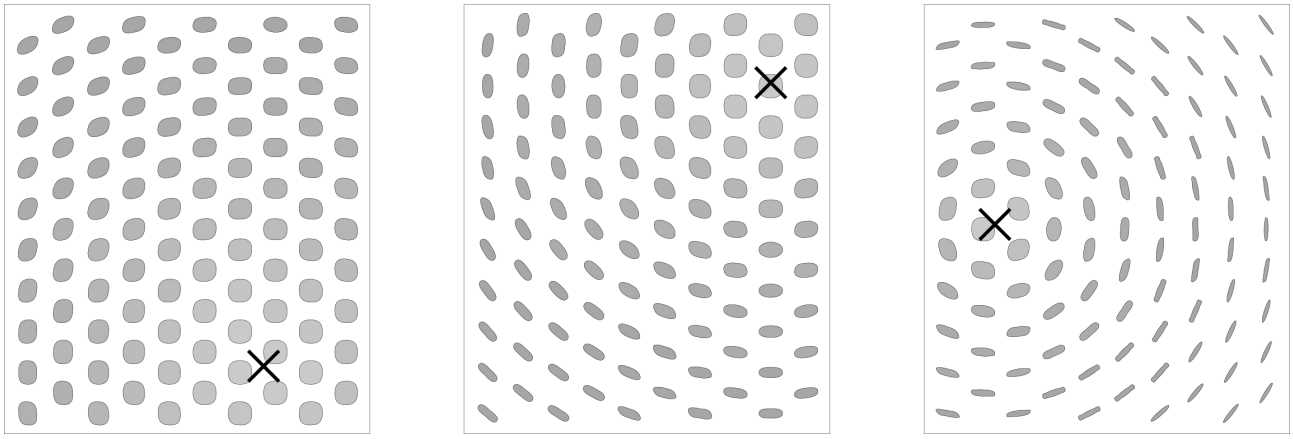
Аттракторы — это точки, которые действуют как виртуальные магниты: либо притягивают, либо отталкивают другие объекты. В Grasshopper любая геометрия, будь она ссылочной из Rhino, либо сгенерированной в Grasshopper, может быть использована в качестве аттрактора. Аттракторы могут влиять на любое количество параметров окружающих объектов, включая масштаб, угол поворота, цвет и местоположение. Эти параметры изменяются, основываясь на их взаимосвязях с геометрией аттрактора.



1. Точка-аттрактор
2. Векторы
3. Окружности, ориентированные в направлении, основанном на их нормали

На изображении выше векторы построены между точкой-аттрактором и точкой центра каждой окружности. Эти векторы используются для определения ориентации окружностей, поэтому они всегда обращены к точке-аттрактору.

Этот же аттрактор может быть использован для изменения других параметров окружности. Например, окружности, находящиеся ближе всего к аттрактору, могут быть масштабированы сильнее, использую длину каждого вектора для масштабирования радиуса каждой окружности.

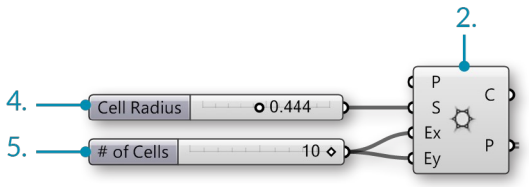


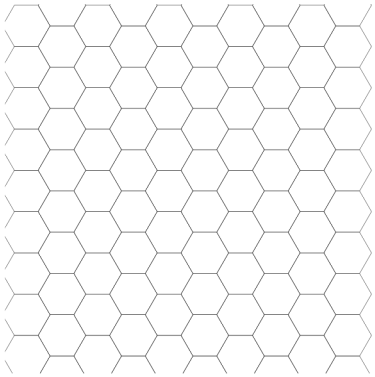
1.3.2.1. ДЕФИНИШИН С АТТРАКТОРОМ

В этом примере мы будем использовать точку-аттрактор для ориентации сети окружностей, базируясь на векторах между точками центра и точкой-аттрактором. Каждая окружность будет ориентирована таким образом, чтобы её нормаль была обращена к точке-аттрактору.

1.	Введите Ctrl+N в Grasshopper, чтобы начать новый дефинишин	
2.	**Vector/Grid/Hexagonal (Вектор/Сетка/Шестигранная Сетка)** — Перетащите компонент **Hexagonal Grid (Шестигранная Сетка)** на холст	
3.	**Params/Input/Slider (Параметры/Вводные/Слайдер)** — Перетащите на холст два **Числовых Слайдера (Numeric Slider)**	
4.	Дважды кликните на первом **Числовом Слайдере** и задайте следующее: Name (Имя): Cell Radius (Радиус Ячейки) Rounding (Округление): Floating Point (С плавающей запятой) Lower Limit (Нижний Лимит): 0.000 Upper Limit (Верхний Лимит): 1.000 Value (Значение): 0.500	
5.	Дважды кликните на втором **Числовом Слайдере** и задайте следующее:	

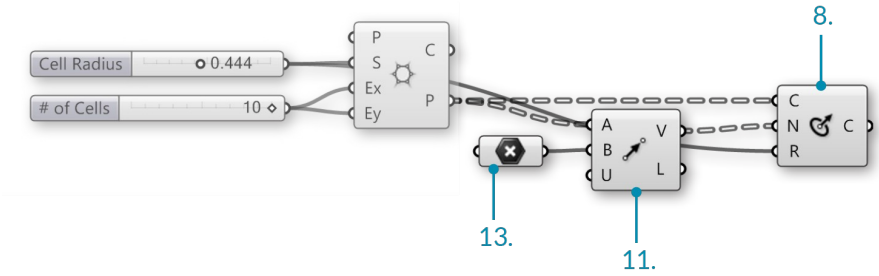
	Name (Имя): # of Cell (Номер Ячейки) Rounding (Округление): Integers (Целые числа) Lower Limit (Нижний Лимит): 0 Upper Limit (Верхний Лимит): 10 Value (Значение): 10	
6.	Подсоедините **Числовой Слайдер (Радиус Ячейки)** ко входу Size (S) (Размер) компонента Hexagon Grid (Шестигранная Сетка)	
7.	Подсоедините Числовой Слайдер (Номер Ячейки) ко входу Extent X (Ex) (Протяжённость по X) и входу Extent Y (Ey) (Протяжённость по Y) компонента Hexagon Grid (Шестигранная Сетка)	

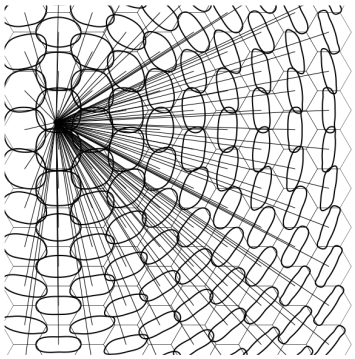




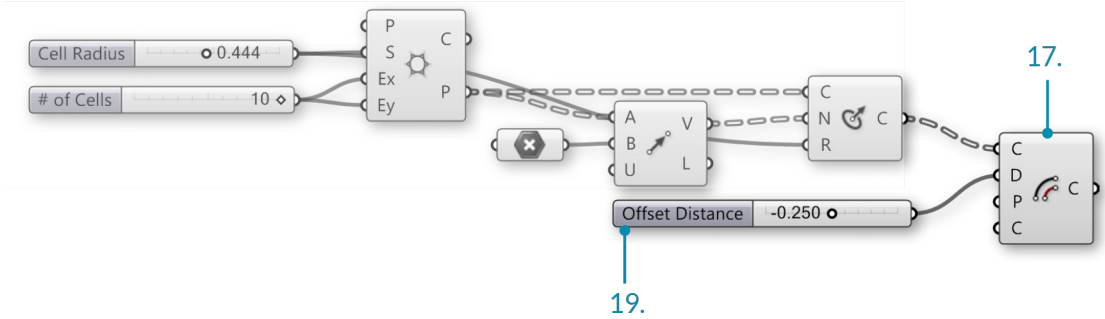
8.	**Curve/Primitive/Circle CNR (Кривая/Примитивы/Окружность по Центру, Нормали и Радиусу)** — Перетащите компонент **Circle CNR** на холст	
9.	Подсоедините выход **Points (P) (Точки)** компонента **Hexagon Grid (Шестигранная Сетка)** ко входу Center (C) (Центр) компонента **Circle CNR (Окружность по Центру, Нормали и Радиусу)**	
10.	Подсоедините **Числовой Слайдер** (Радиус Ячейки) ко входу Radius (R) (Радиус) компонента **Circle CNR**.	
11.	**Vector/Vector/Vector 2Pt (Вектор/Вектор/Вектор по 2м Точкам)** — Перетащите компонент **Vector 2Pt** на холст	
12.	Подсоедините выход **Points (P) (Точки)** компонента **Hexagonal Grid (Шестигранная Сетка)** ко входу Base Point (A) (Базовая Точка) компонента **Vector 2Pt (Вектор по 2м Точкам)**.	

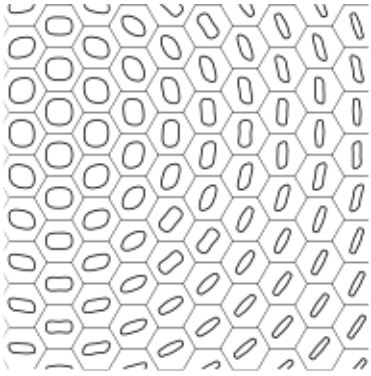
13.	**Params/Geometry/Point (Параметры/Геометрия/Точка)** – Перетащите компонент **Point** на холст	
14.	Кликните правой кнопкой мыши по компоненту **Point (Точка)** и выберите одну точку. В пространстве модели выберите где бы Вы хотели установить точку-аттрактор.	
15.	Подсоедините компонент **Point (Точка)** ко входу Tip Point (B) (Точка Верхушки) компонента **Vector 2Pt (Вектрор по 2-м Точкам)**	
16.	Подсоедините выход Vector (V) (Вектор) компонента **Vector 2Pt (Вектор по 2-м Точкам)** ко входу Normal (N) (Нормаль) компонента **Circle CNR (Окружность по Центру, Нормали и Радиусу)**.	



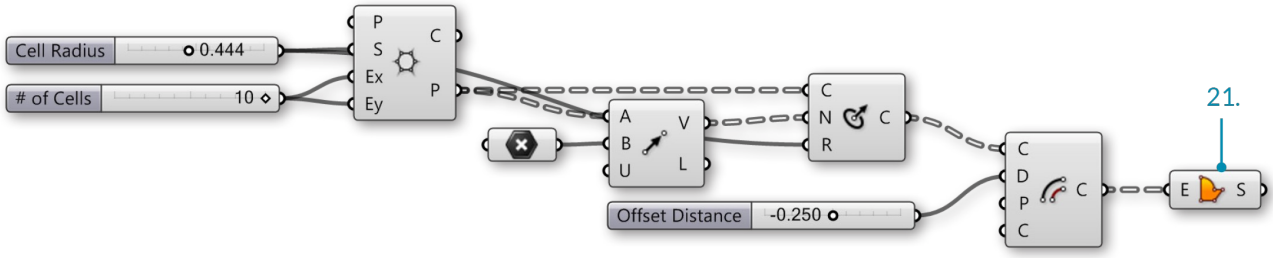


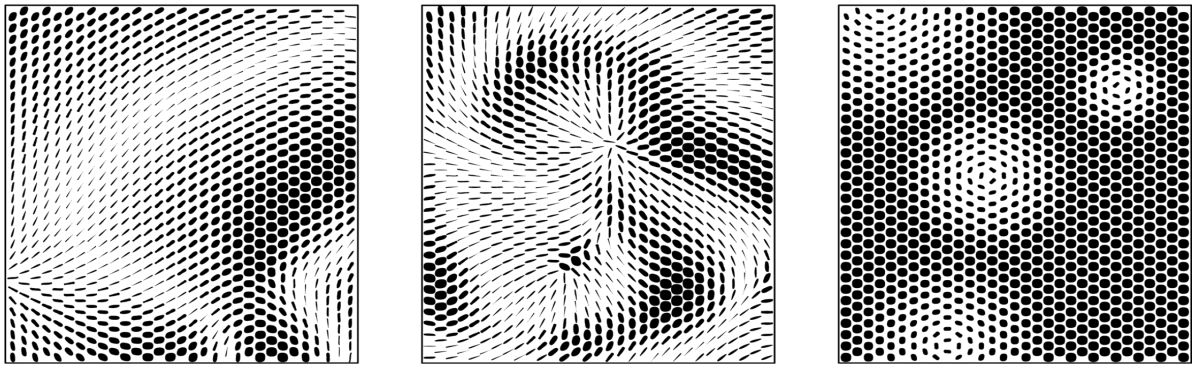
17.	**Curve/Util/Offset (Кривая/Утилиты/Сдвиг)** – Перетащите на холст компонент **Offset (Сдвиг)**.	
18.	Дважды кликните на Числовом Слайдере и задайте следующее: Name (Имя): Offset Distance (Расстояние Сдвига) Rounding (Округление): Floating Point (С Плавающей Запятой) Lower Limit (Нижний Предел): - 0.500 Upper Limit (Верхний Предел): 0.500 Value (Значение): -0.250	
19.	Подсоедините Числовой Слайдер (Расстояние Сдвига) ко входу Distance (D) (Расстояние) компонента Offset (Сдвиг)	

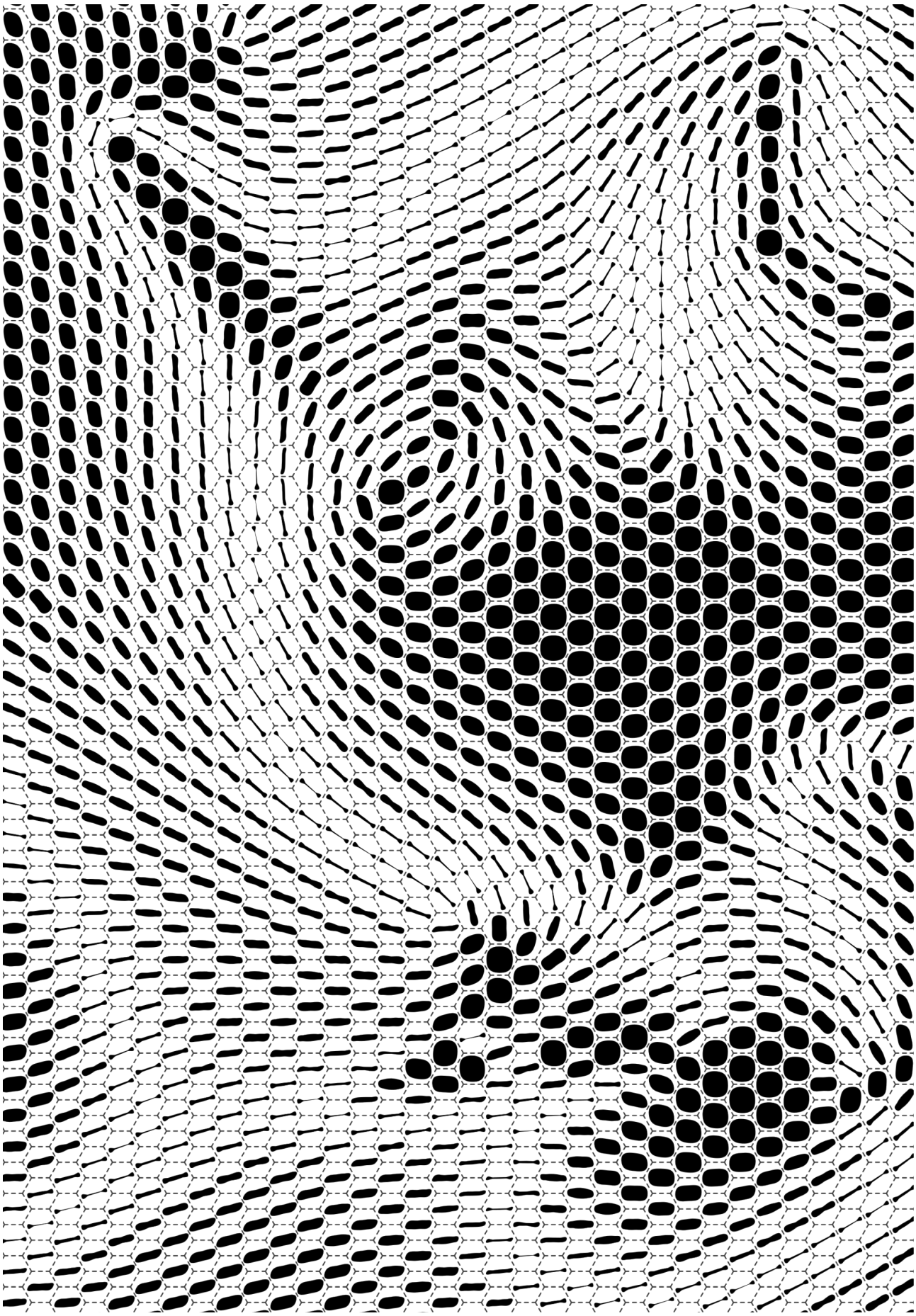




20.	**Surface/Freeform/Boundary Surfaces (Поверхность/Произвольной Формы/Граничная Поверхность)** – Перетащите на холст компонент **Boundary Surfaces**	
21.	Подсоедините выход Curves (C) (Кривые) компонента Offset (Сдвиг) ко входу Edges (E) (Края) компонента Boundary Surfaces (Граничная Поверхность)	







1.3.3. Mathematics (Математика), Expressions (Выражения) и Conditionals (Операторы Условий)

Загрузить файл примера, сопровождающего данный раздел.

Знание о том, как работать с числовой информацией является важным навыком для мастерского освоения Grasshopper. Grasshopper содержит много компонентов для выполнения математических операций, оценки условий и манипуляции набором чисел.

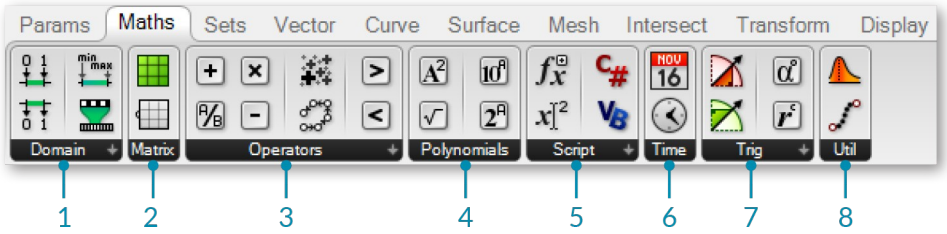
В математике цифры организованы в наборы множеств и, вероятно, с этими двумя Вы знакомы:

Целые Числа: […, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, …]
Действительные Числа: [8, …, -4.8, -3.6, -2.4, -1.2, 0.0, 1.234, e, 3.0, 4.0, …, 8]

И хотя существуют наборы и других типов, эти два интересуют нас больше всего, потому что Grasshopper их широко использует. Несмотря на то, что есть ограничения представления этих наборов именно в цифровой среде, мы можем их аппроксимировать с высокой степенью точности. Кроме того, следует понимать, что различие между целочисленными типами (целые числа) и типами с плавающей запятой (действительные числа) соответствуют различию между дискретными и непрерывными доменами. В этой главе мы собираемся исследовать различные методы обработки и оценки различных наборов чисел.

1.3.3.1. ВКЛАДКА MATH (МАТЕМАТИКА)

Большинство компонентов, имеющих дело с математическими операциями и функциями могут быть найдены в следующих подкатегориях на вкладке Math (Математика):

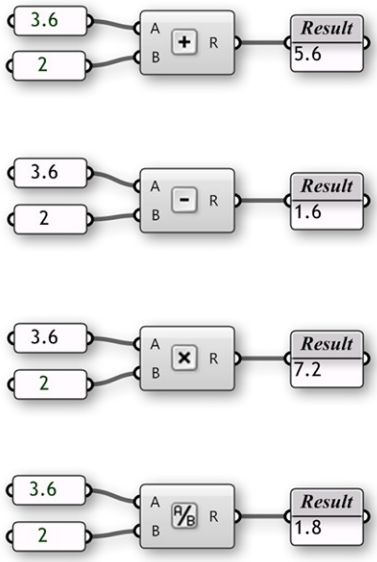


1. Domain (Домены) используются для определения диапазона значений (ранее известных как интервалы) между двумя числами. Компоненты на вкладке Domain позволяют создать или разложить различные типы доменов.
2. В математике матрица (matrix) — это массив чисел, организованных в виде строк и столбцов. Данная подкатегория содержит ряд полезных инструментов для конструирования и модификации матриц.
3. Операторы (Operators) используются для выполнения математических операций, таких как сложение, вычитание, умножение и т. д. Операторы условий позволяют Вам определить набор чисел, который больше, меньше, либо равен другому набору чисел.
4. Полиномы (Polynomials (Многочлены)) являются одним из самых важных понятий в алгебре, а также всей математике и науке. Вы можете использовать компоненты, расположенные в данной категории, чтобы вычислить факториалы, логарифмы и возвести число в n-ную степень.
5. Подкатегория Скрипты (Script (Сценарий)) содержит одно- и много-переменные выражения, а также компоненты, для подключения скриптов, написанных на VB.NET и C# .
6. Подкатегория Time (Время) имеет ряд компонентов, позволяющих конструировать структуры, учитывающие дату и время.
7. Эти компоненты позволяют решать тригонометрические функции. Такие, как синус, косинус, тангенс и т.д.
8. Подкатегория Utility (Утилиты) собрала в себя полезные компоненты, которые можно использовать в различных математических уравнениях. Проверьте здесь, если Вы пытаетесь найти максимальное и минимальное значение между двумя списками чисел, или вычислить среднее группы чисел.

1.3.3.2. OPERATORS (ОПЕРАТОРЫ)

Как уже упоминалось ранее, операторы представляют собой набор компонентов, которые используют алгебраические функции с двумя входящими значениями и одним исходящим — результирующим значением.

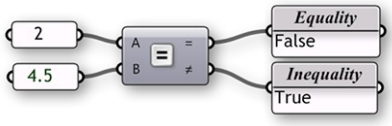
Большую часть времени Вы будете использовать математические операторы для выполнения арифметических действий над наборами чисел. Тем не менее, эти операторы могут быть также использованы на различных типах данных, включая точки и векторы.



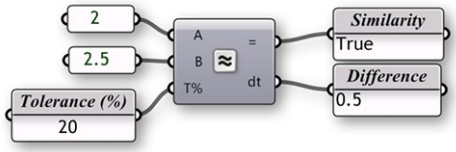
1.3.3.3. ОПЕРАТОРЫ УСЛОВИЙ (CONDITIONAL OPERATORS)

Почти каждый язык программирования имеет метод для оценки операторов условий. В большинстве случаев программист создаёт кусок кода, чтобы задать простой вопрос: «а что, если?» Что делать, если площадь контура крыши превышает программные требования? Или, что делать, если кривизна моей крыши реалистичную величину? Это — важные вопросы, которые представляют более высокий уровень абстрактного мышления. Компьютерные программы имеют способность анализировать эти «что, если?» и предпринимать действия, в зависимости от ответа на данный вопрос. Давайте взглянем на очень простой оператор условия, который программа может интерпретировать: Если объект является кривой — удалить его. Кусок кода сначала смотрит на объект и определяет единственное логическое значение: является объект кривой или нет. Логическое значение «True» (ИСТИНА), если объект является кривой, либо «False» (ЛОЖЬ), если не является. Вторая часть оператора выполнит действие в зависимости от результата условного выражения. В том случае, если объект является кривой — удалит его. Этот условный оператор называется оператор «If» (ЕСЛИ). Есть четыре оператора условия (ищите в подкатегории Math/ Operators (Математика/Операторы), оценивающие состояние и возвращающих логическое значение.

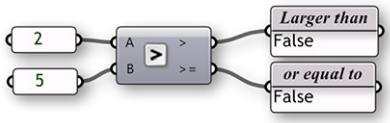
Компонент Equality (Равенство) берёт два списка и сравнивает первый элемент Списка A с первым элементом Списка B. Если эти два значения совпадают, то создаётся логическое значение «True» (ИСТИНА) и наоборот: если два значения не равны, то создаётся логическое значение «False» (ЛОЖЬ). Компонент просматривает списки согласно алгоритму набора данных (по-умолчанию, он установлен в состояние «Longest List» (Длинный Список)). У этого компонента есть два выхода. Первый возвращает список булевых (логических) значений, который показывает, какие из значений в списке были равны друг другу. Второй выход возвращает список, который показывает, какие значения не равны друг другу, или, другими словами — список, инвертированный от первого выхода.



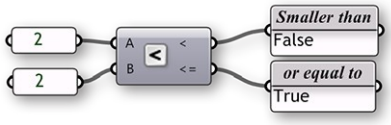
Компонент Similarity (Подобие) оценивает два списка данных и тестирует на сходство между двумя числами. Это, практически, идентично тому, как сравнивает два списка компонент Equality (Равенство), за одним исключением: он учитывает процентное соотношение допустимого отклонения значений списка A от значений списка B. Компонент Similarity (Подобие) также имеет выход, который определяет абсолютное значение дистанции между двумя исходными списками.



Компонент Larger Than (Больше Чем) берёт два списка данных и определяет, является ли первый элемент списка A больше первого элемента списка B. Два выхода компонента позволяют определить: хотели ли Вы сравнить два списка по схеме «Больше, чем» (>) или «Больше или равно» (>=).



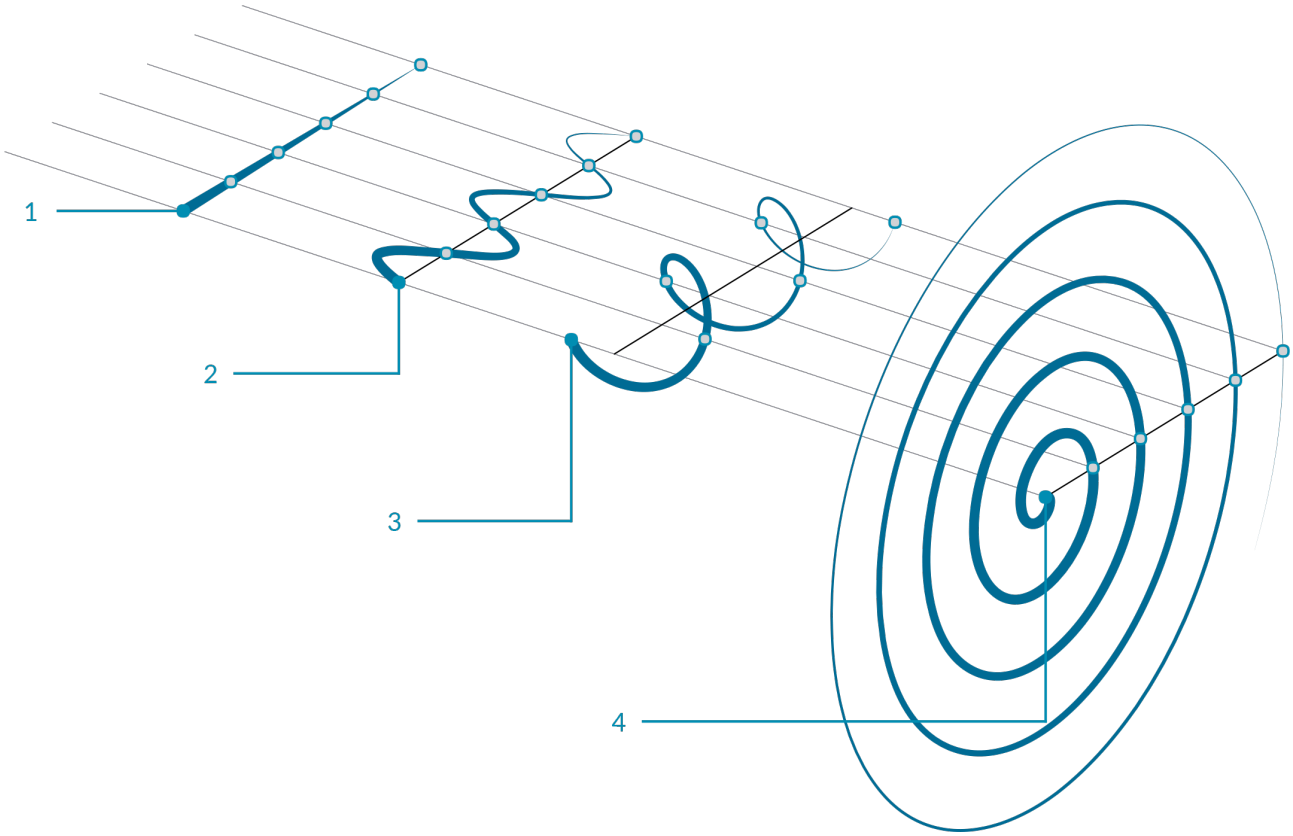
Компонент Smaller Than (Меньше Чем) выполняет действие, противоположное действию компонента Larger Than (Больше Чем). Компонент Smaller Than (Меньше Чем) определяет в списке A меньшие значения, чем в списке B и возвращает список логических значений. Аналогичным образом два выхода позволяют определить, хотели ли Вы оценивать каждый список по схеме «Меньше Чем» (<) или «Меньше или равно» (<=).



1.3.3.4. ТРИГОНОМЕТРИЧЕСКИЕ КОМПОНЕНТЫ

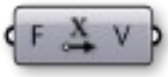
Загрузить файл примера, относящийся к данному разделу.

Мы уже показали, что можем использовать компоненты Expression (Выражение) (или Evaluate (Сравнение)) для вычисления алгебраических уравнений или сравнения. Однако, существуют и другие способы вычисления простых выражений с помощью нескольких встроенных тригонометрических функций. Мы можем использовать эти функции для определения таких периодических явлений, как волны синусоидальной формы типа океанских, звуковых или световых волн.

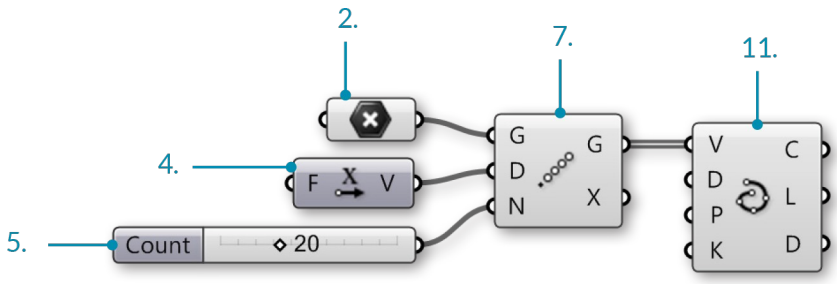


1. Прямая
   y(t) = 0
2. Синусоида
   y(t) = sin(t)
3. Пружина
   x(t) = cos(t)
   y(t) = sin(t)
   z(t) = b(t)
4. Спираль
   x(t) = t*cos(t)
   y(t) = t*cos(t)

В этом примере мы будем использовать Grasshopper для конструирования различных тригонометрических кривых, используя тригонометрические функции компонентов, которые Вы можете найти на вкладке Math (Математика):

1.	Введите Ctrl+N (в Grasshopper), чтобы начать новый дефинишин	
2.	**Params/Geometry/Point (Параметры/Геометрия/Точка)** – Перетащите на холст параметр **Point (Точка)**	
3.	Кликните правой кнопкой мыши на параметре **Point (Точка)** и выберите «Set One Point» (Выбрать Одну Точку) – выберите точку во вьюпорте (viewport) Rhino	
4.	**Vector/Vector/Unit X (Вектор/Вектор/Унифицированный по X)** – Перетащите на холст компонент **Unit X (Унифицированный по X)**	
5.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст **Числовой Слайдер**	
6.	Дважды кликните по **Числовому Слайдеру** и задайте следующее: Rounding (Округление): Integer (Целые числа) Lower Limit (Нижний Предел): 10 Upper Limit (Верхний Предел): 40 Value (Значение): 20	
7.	**Transform/Array/Linear Array (Трансформирование/Массив/Линейный Массив)** – Перетащите на холст компонент **Linear Array (Линейный Массив)**	
8.	Подсоедините выход параметра **Point (Точка)** ко входу Geometry (G) (Геометрия) компонента **Linear Array (Линейный Массив)**	
9.	Подсоедините выход Unit Vector (V) (Унифицированный Вектор) компонента **Unit X (Унифицированный по X)** ко входу Direction (D) (Направление) компонента **Linear Array (Линейный Массив)** Вы должны увидеть строку из 20 точек вдоль X-оси в Rhino. Откорректируйте слайдер, чтобы изменить количество точек в массиве.	
10.	Подсоедините выход **Числового Слайдера** ко входу Count (N) (Счёт) компонента **Linear Array (Линейный Массив)**	

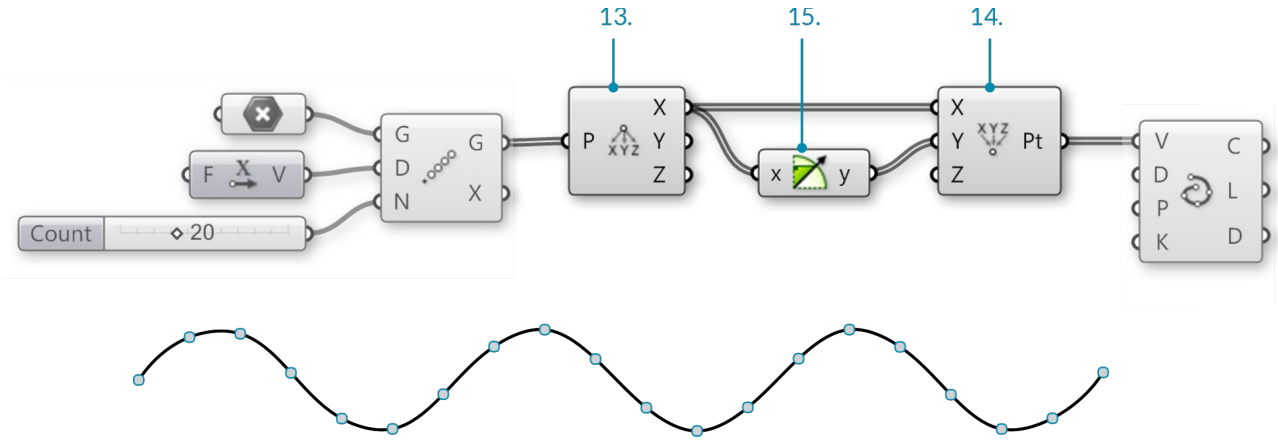
11.	**Curve/Spline/Interpolate (Кривая/Сплайн/Интерполированная Кривая)** – Перетащите на холст компонент **Interpolate Curve (Интерполированная Кривая)**	
12.	Подсоедините выход Geometry (G) (Геометрия) компонента **Linear Array (Линейный Массив)** ко входу Vertices (V) (Вершины) компонента **Interpolate Curve (Интерполированная Кривая)**	



Мы только что подключили массив точек к кривой. Давайте попробуем использовать некоторые из компонентов Тригонометрии Grasshopper, чтобы изменить эту кривую:

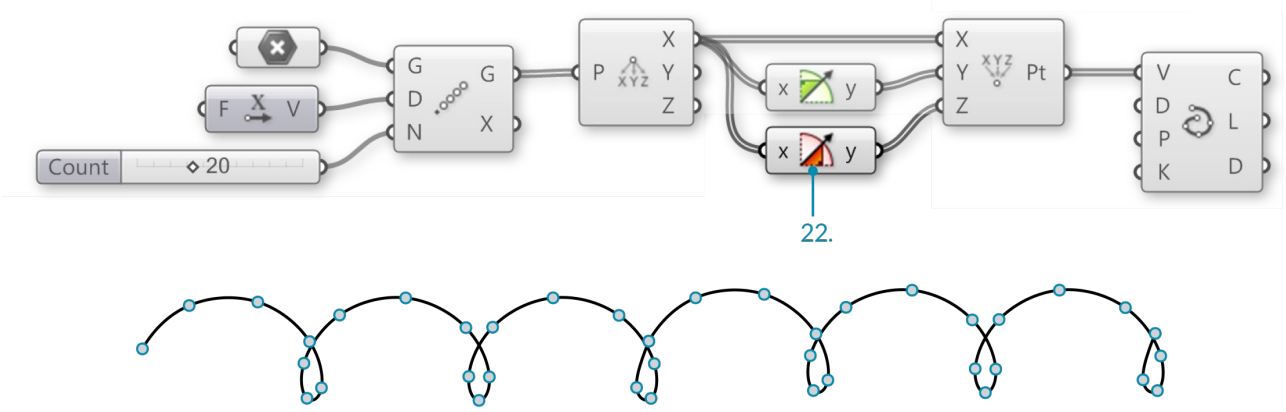
13.	**Vector/Point/Deconstruct (Вектор/Точка/Разобрать)** – Перетащите на холст компонент **Deconstruct (Разобрать Точку)**	
14.	**Vector/Point/Construct Point (Вектор/Точка/Сконструировать)** — Перетащите на холст компонент **Construct Point (Сконструировать Точку)**	
15.	**Maths/Trig/Sine (Математика/Тригонометрия/Синус)** — Перетащите компонент **Sine (Синус)** на холст	
16.	Отсоедините провод от входа Vertices (V) (Вершины) компонента **Interpolate Curve (Интерполированная Кривая)**. Вы можете отсоединить провода, удерживая нажатой клавишу Ctrl и перетаскивая, либо кликом правой кнопкой мыши по входу и выбором Disconnect (Отсоединить)	
17.	Подсоедините выход Geometry (G) (Геометрия) компонента **Linear Array (Линейный Массив)** ко входу Point (P) (Точка) компонента **Deconstruct (Разобрать Точку)**	

18.	Подсоедините выход Point X (X) (X-координата Точки) компонента **Deconstruct (Разобрать Точку)** ко входу X coordinate (X) (X-координата) компонента **Construct Point (Сконструировать Точку)**	
19.	Подсоедините вторым проводом выход Point X (X) (X-координата Точки) компонента **Deconstruct (Разобрать Точку)** ко входу Value (x) (Значение X) компонента **Sine (Синус)**	
20.	Подсоедините выход Result (y) (Результат) компонента **Sine (Синус)** ко входу Y coordinate (Y) (Координата по Y) компонента **Construct Point (Сконструировать Точку)** Мы только что реконструировали наши точки с теми же значениями координат по X, но с модифицированными с помощью синусоиды значениями координат по Y.	
21.	Содеините выход Point (Pt) (Точка) компонента **Construct Point (Сконструировать Точку)** ко входу Vertices (V) (Вершины) компонента **Interpolate (Интерполированная Кривая)**	



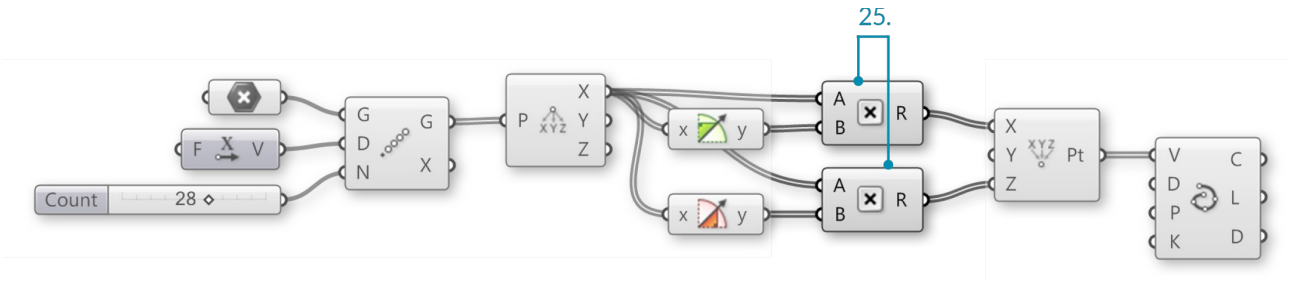
Сейчас Вы должны увидеть кривую в виде синусоидальной волны вдоль X-оси в Rhino.

22.	**Maths/Trig/Cosine (Математика/Тригонометрия/Косинус)** – Перетащите на холст компонент **Cosine (Косинус)**	
23.	Подсоедините третьим проводом выход Point X (X) (X-координата Точки) компонента **Deconstruct (Разобрать Точку)** ко входу Value (x) (Значение X) компонента **Cosine (Косинус)**	
24.	Подсоедините выход Result (y) (Результат) компонента **Cosine (Косинус)** ко входу Z coordinate (Z) (Z-координата) компонента **Construct Point (Сконструировать Точку)**	

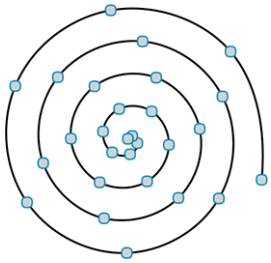


Теперь мы получили 3D-пружину

25.	**Maths/Operators/Multiplication (Математика/Операторы/Умножение)** – Перетащите на холст два компонента **Multiplication (Умножение)**	
26.	Подсоедините проводом выход Point X (X) (X-координата Точки) компонента **Deconstruct (Разобрать Точку)** ко входу (A) каждого компонента **Multiplication (Умножение)**	
27.	Подсоедините выход Result (y) (Результат) компонента **Sine (Синус)** ко входу (B) первого компонента **Multiplication (Умножение)**	
28.	Подсоедините выход Result (y) (Результат) компонента **Cosine (Косинус)** ко входу (B) второго компонента **Multiplication (Умножение)**	
29.	Отсоедините провод от входа Y Coordinate (Y) (Y-координата) компонента **Construct Point (Сконструировать Точку)**	
30.	Подсоедините выход Result (R) (Результат) первого компонента **Multiplication (Умножение)** ко входу X Coordinate (X) (X-координата) компонента **Construct Point (Сконструировать Точку)**	
31.	Подсоедините выход Result (R) (Результат) второго компонента **Multiplication (Умножение)** ко входу Z Coordinate (Z) (Z-координата) компонента **Construct Point (Сконструировать Точку)**	



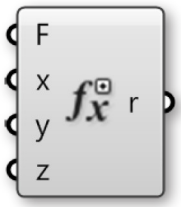
Теперь Вы должны увидеть спиральную кривую



1.3.3.5. EXPRESSIONS (ВЫРАЖЕНИЯ)

Загрузить файл примера, относящегося к данному разделу.

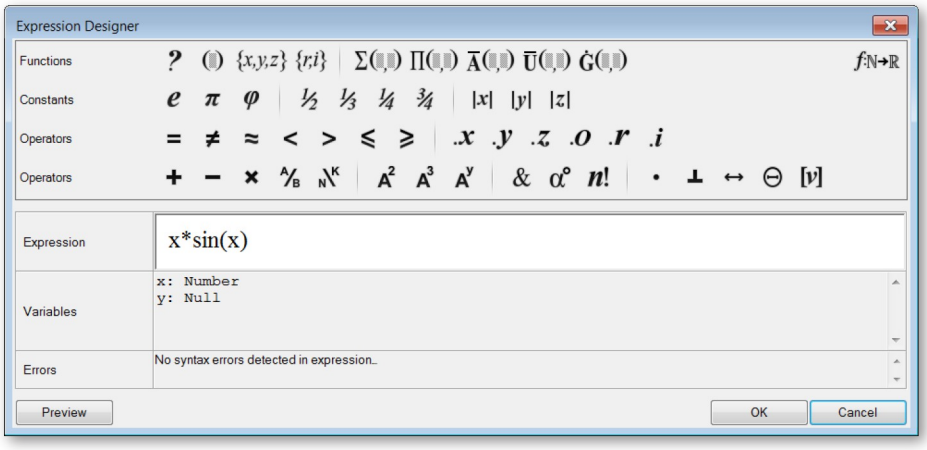
Компонент Expression (Выражение) (и его брат — компонент Evaluate (Оценка)) — очень гибкие инструменты, что означает, что они могут быть использованы в различных приложениях. Вы можете использовать Выражение (или Оценку) для решения математических алгоритмов и возвращения числовых данных на выходы.



В следующем примере мы рассмотрим математические спирали, встречающиеся в природе и то, что как мы сможем использовать математические функции Грасхопера для создания аналогичных структур. В качестве отправной точки мы будем опираться на наши тригонометрические кривые.

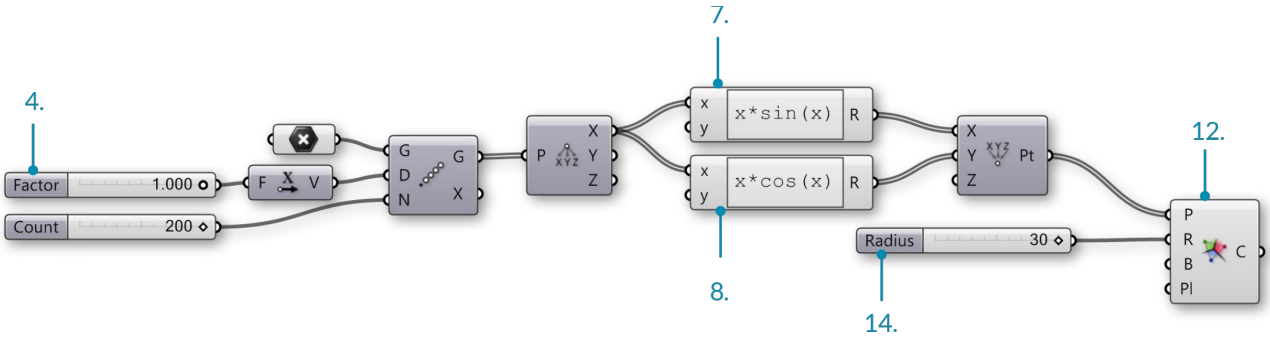
1.	Откройте ваш Grasshopper -дефинишин с Тригонометрическими кривыми из предыдущего примера	
2.	Удалите компоненты **Sine (Синус)**, **Cosine (Косинус)**, **Multiplication (Умножение)** и **Interpolate (Интерполированная Кривая)**	

3.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст **Числовой Слайдер**	
4.	Дважды кликните по **Числовому Слайдеру** и задайте следующее: Rounding (Округление): Float (С плавающей запятой) Lower Limit (Нижний Предел): 0.000 Upper Limit (Верхний Предел): 1.000 Value (Значение): 1.000	
5.	Подсоедините **Числовой Слайдер** ко входу Factor (F) (Коэффициент) компонента Unit X (Унифицированный по X). > Этот слайдер позволяет откорректировать расстояние между точками в массиве.	
6.	**Maths/Script/Expression (Математика/Скрипты/Выражение)** – Перетащите на холст два компонента **Expression (Выражение)**	
7.	Дважды кликните по первому компоненту **Expression (Выражение)**, чтобы открыть Expression Editor (Редактор Выражения) и измените выражение на следующее: x*sin(x)	
8.	Дважды кликните по второму компоненту **Expression (Выражение)**, чтобы открыть Expression Editor (Редактор Выражения) и измените выражение на следующее: x*cos(x)	

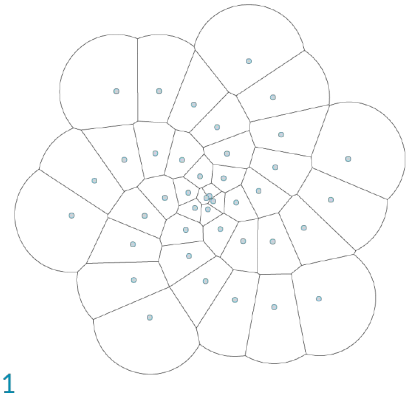


> Двойной клик по компоненту Expression (Выражение) откроет Редактор Выражений Грасхопера (Expression Editor)

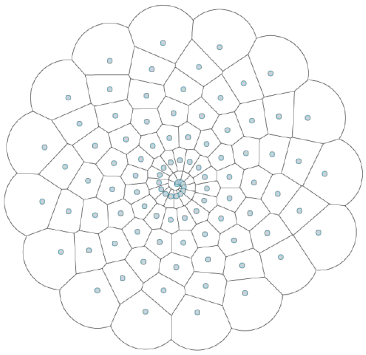
9.	Протяните два провода связи от выхода Point X (X) (X-координата Точки) компонента **Deconstruct (Разобрать Точку)** ко входу Variable x (x) (Переменная X) каждого компонента **Expression (Выражение)**	
10.	Подсоедините выход Result (R) (Результат) первого компонента **Expression (Выражение)** ко входу X coordinate (X) (X-координата) компонента **Construct Point (Сконструировать Точку)**	
11.	Подсоедините выход Result (R) (Результат) второго компонента Expression (Выражение) ко входу Y coordinate (Y) (Y-коодината) компонента Construct Point (Сконструировать Точку) > Мы заменили Тригонометрические функции и операторы умножения на компонент «Выражение» для повышения эффективности дефинишина.	
12.	**Mesh/Triangulation/Voronoi (Полигональная сетка/Триангуляция/Вороного)** – Перетащите на холст компонент **Voronoi (Вороной)**	
13.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст **Числовой Слайдер**	
14.	Дважды кликните на **Числовом Слайдере** из задайте следующиее: Rounding (Округление): Integer (Целые Числа) Lower Limit (Нижний Предел): 1 Upper Limit (Верхний Предел): 30 Value (Значение): 30	
15.	Подсоедините **Числовой Слайдер** ко входу Radius (R) (Радиус) компонента **Voronoi (Вороной)**	
16.	Подсоедините выход Point (Pt) (Точка) компонента **Construct Point (Сконструировать Точку)** ко входу Points (P) (Точки) компонента **Voronoi (Вороной)**	



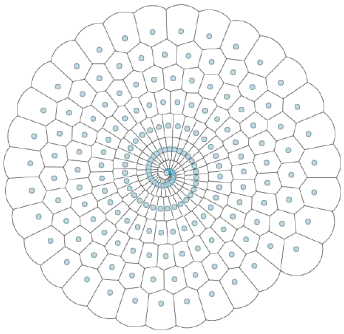
Вы можете создать другие паттерны Вороного, манипулируя слайдерами Factor (Коэффициент), Count (Количество) и Radius (Радиус). Ниже приведены три примера:



1

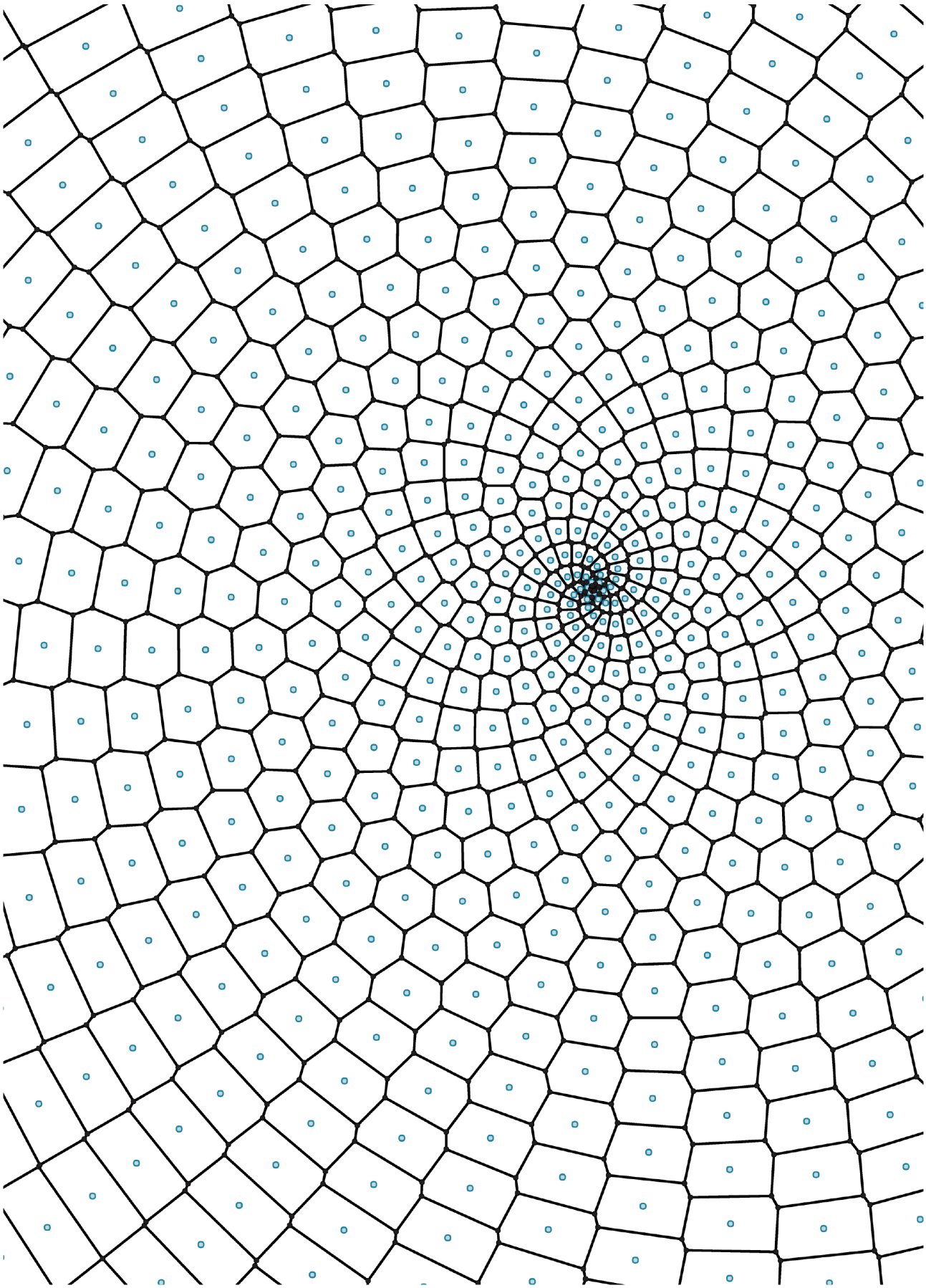


2



3

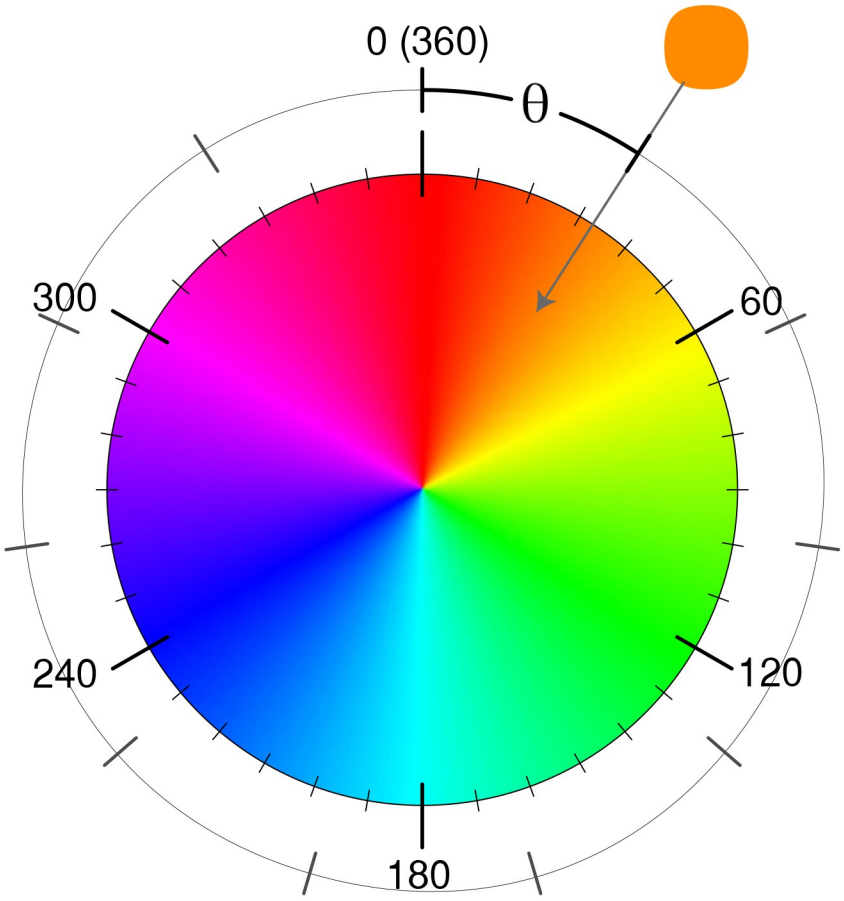
1. Factor = 1.000, Radius = 15
2. Factor = 0.400, Radius = 10
3. Factor = 0.200, Radius = 7



1.3.4. Домены (Domains) и Цвет (Color)

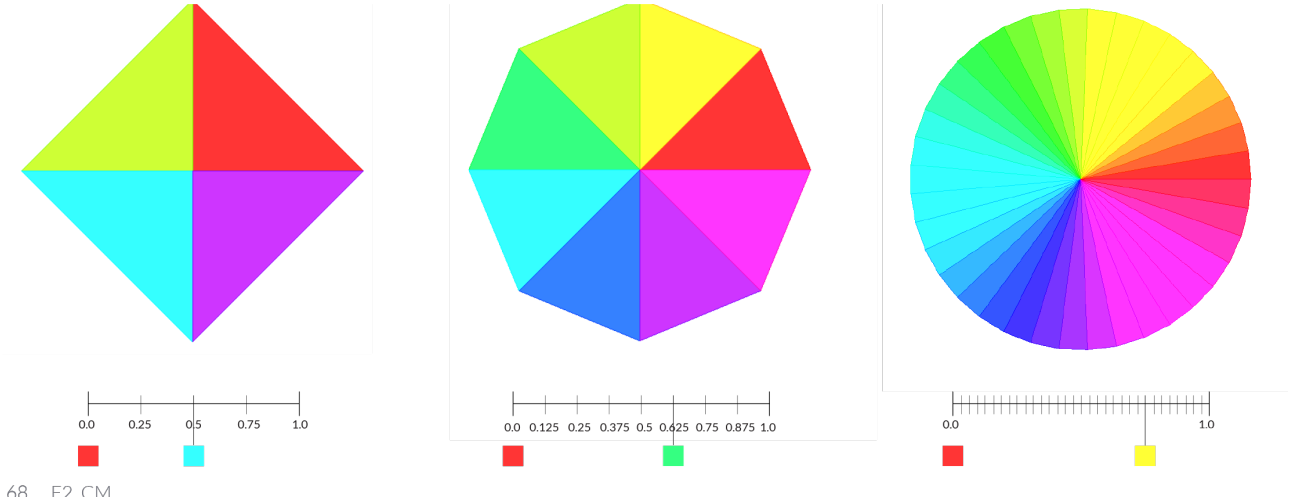
Загрузить файл примера, относящегося к данному разделу.

Цветовой круг — это модель организации цвета, базируясь на их оттенке. В Grasshopper цвета могут быть определены по значению их оттенка в диапазоне от 0.0 до 1.0. Домены используются для определения диапазона всех возможных значений набора чисел между нижним пределом (A) и верхним пределом (B).



В цветовом круге оттенок соотносится с углом. Grasshopper взял этот диапазон от 0 до 360 и перераспределил его между нулём и единицей.

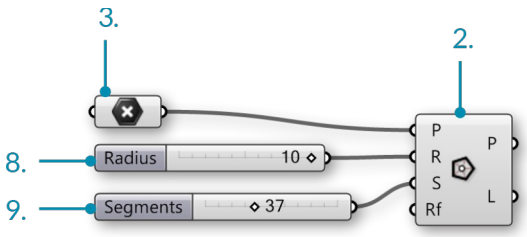
Разделив домен Hue (Оттенок) (0.0 до 1.0) по количеству нужных сегментов мы можем присвоить значение оттенка для каждого сегмента, создав таким образом свой цветовой круг.



В этом примере мы будем использовать Grasshopper-домен и цветовые компоненты для создания цветового круга с переменным количеством сегментов.

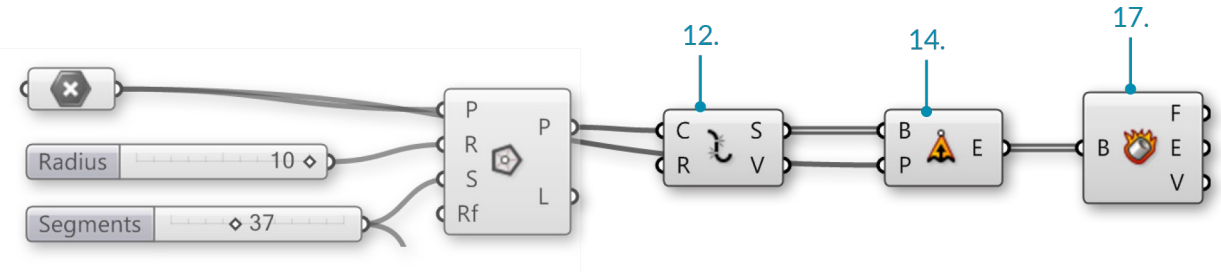
1.	Введите Ctrl+N (в Grasshopper), чтобы начать новый дефинишин	
2.	**Curve/Primitive/Polygon (Кривая/Примитивы/Многогранник)** – Перетащите на холст компонент **Polygon (Многогранник)**	
3.	**Params/Geometry/Point (Параметры/Геометрия/Точка)** – Перетащите на холст параметр **Point (Точка)**	
4.	Кликните правой кнопкой мыши по компоненту Point и выбретите «Set One Point» (Задать Одну Точку)	
5.	Задайте одну точку в пространстве модели.	
6.	Подсоедините Параметр **Point (Точка (Базовая Точка))** ко входу Plane (Плоскость)(P) компонента **Polygon (Многогранник)**	
7.	**Params/Input/Number Sliders (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст два **Числовых Слайдера**	
8.	Дважды кликните на первом **Числовом Слайдере** из	

	задайте следующиее: Rounding (Округление): Integer (Целые Числа) Lower Limit (Нижний Предел): 1 Upper Limit (Верхний Предел): 10 Value (Значение): 10	
9.	Дважды кликните на втором **Числовом Слайдере** из задайте следующиее: Rounding (Округление): Integer (Целые Числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 100 Value (Значение): 37	
10.	Подсоедините Числовой Слайдер (Радиус) ко входу Radius (R) компонента Polygon (Многоугольник) > Когда Вы подсоедините числовой слайдер к компоненту, то его имя автоматически заменится именем того входа, к которому его присоединили.	
11.	Подсоедините Числовой Слайдер (Количество Сегментов) ко входу Segments (S) (Сегменты) компонента Polygon (Многоугольник)	



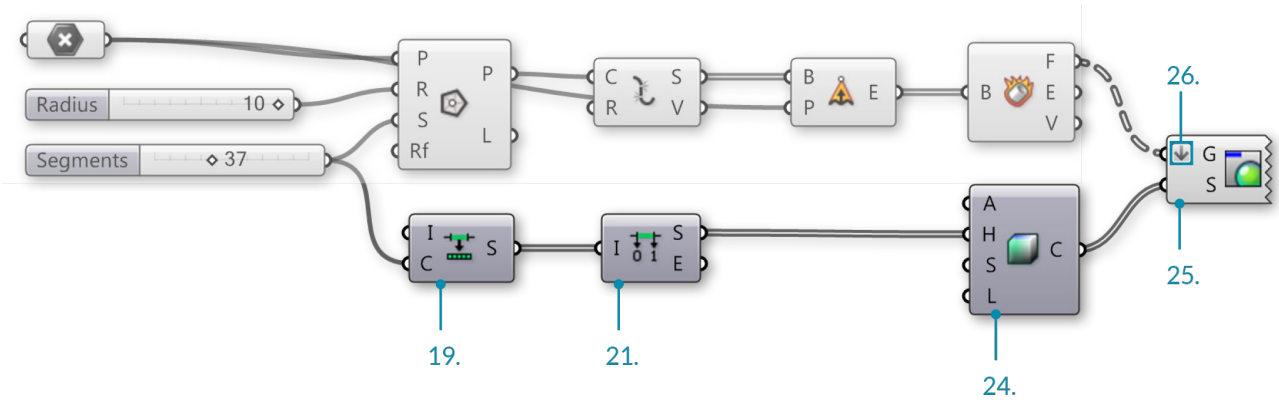
12.	**Curve/Util/Explode (Кривая/Утилиты/Разбить)** – Перетащите на холст компонент **Explode (Разбить)**.	
13.	Подсоедините выход Polygon (P) (Многоугольник) компонента **Polygon (Многоугольник)** ко входу компонента **Curve (C) (Кривая)**	
14.	**Surface/Freeform/Extrude Point (Поверхность/Произвольной формы/Выдавить в Точку)** – Перетащите на холст компонент Extrude Point **(Выдавить (экструдировать) в Точку)**	
15.	Подсоедините выход Segments (S) (Сегменты) компонента	

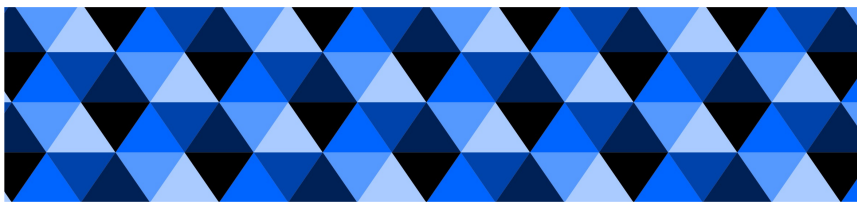
	Explode (Разбить) ко входу Base (B) (База) компонента **Extrude Point (Выдавить в Точку)**	
16.	Подсоедините Параметр **Point (Точка (Базовая Точка))** ко входу Extrusion Tip (P) (Вершина Выдавливания) компонента **Extrude Point (Выдавить в Точку)**	
17.	**Surface/Analysis/Deconstruct Brep (Поверхность/Анализ/Разобрать Brep)** – Перетащите на холст компонент **Deconstruct Brep (Разобрать Brep)** *[Brep — геометрия, представленная в виде внешних поверхностей(Прим.переводчика)]*	
18.	Подсоедините выход Extrusion (E) (Выдавливание (Экструзия)) компонента **Extrude Point (Выдавить в Точку)** ко входу компонента **Deconstruct Brep (B) (Разобрать Brep)**	



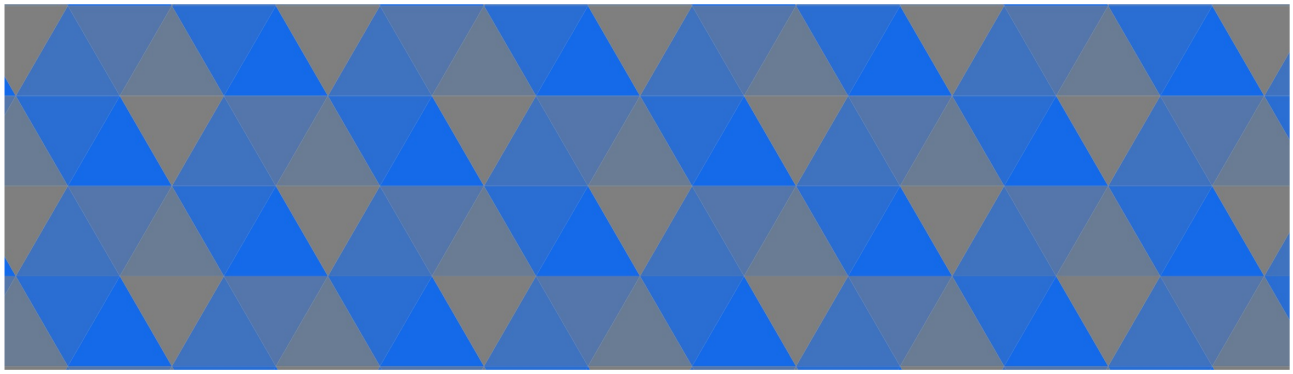
19.	**Maths/Domain/Divide Domain (Математика/Домен/Разбить Домен)** – Перетащите на холст компонент **Divide Domain (Разбить Домен)** Базовый Домен (Base Domain (I)) автоматически установится между 0.0-1.0 и это как раз то, что нам нужно для данного упражнения	
20.	Подсоедините **Числовой Слайдер** (Segments (Количество Сегментов)) ко входу Count (C) (Число) компонента **Divide Domain (Разбить Домен)**	
21.	**Math/Domain/Deconstruct Domain (Математика/Домен/Разобрать Домен)** – Перетащите на холст компонент **Deconstruct Domain (Разобрать Домен)**	
22.	Подсоедините выход Segments (S) (Сегменты) компонента **Divide Domain (Разбить Домен)** ко входу Domain (I) (Домен) компонента **Deconstruct Domain (Разорбрать Домен)**	

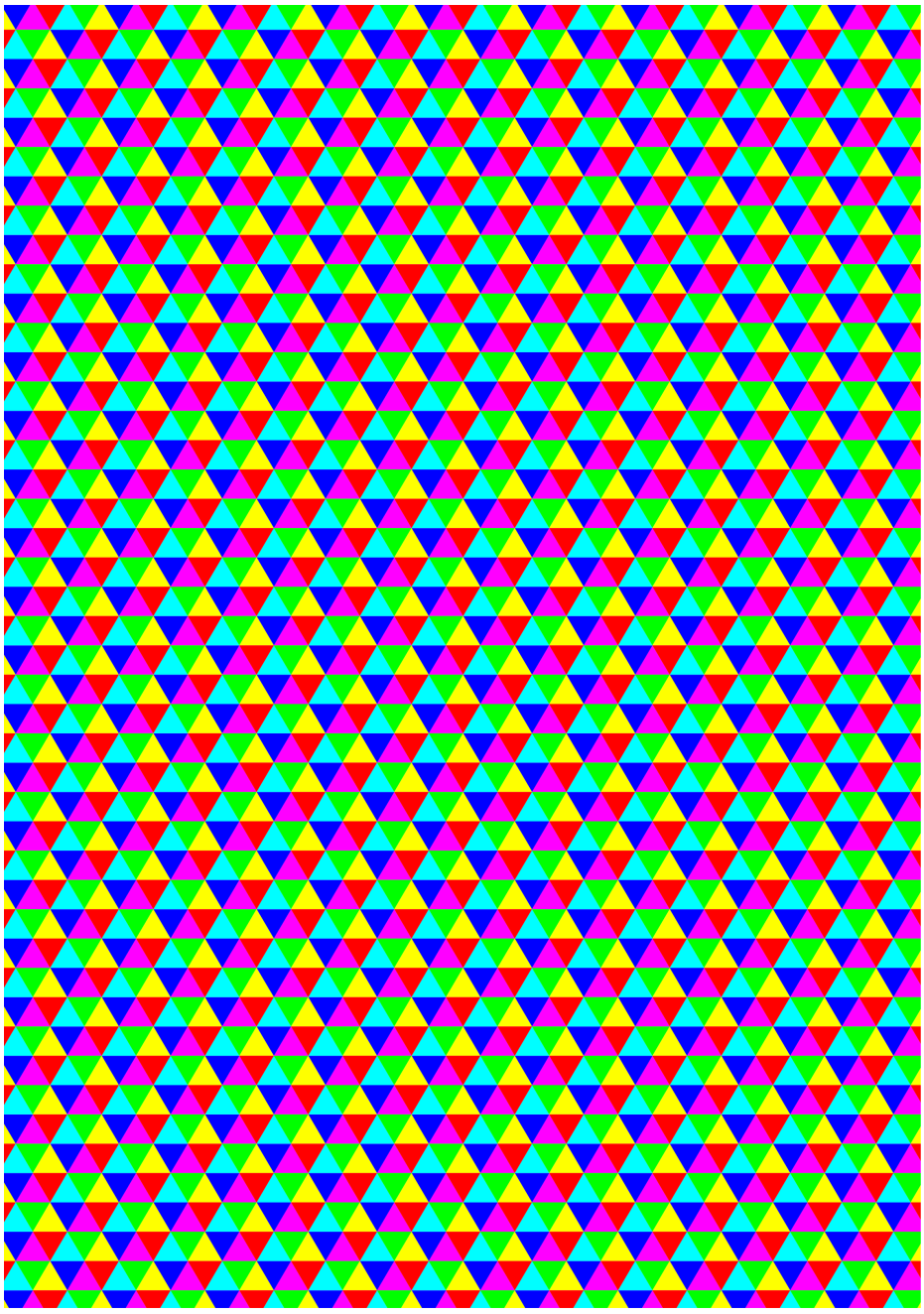
23.	**Display/Colour/Colour HSL (Отображение/Цвет/Цвета HSL** *[Тон, Насыщенность, Светлота]***)** – Перетащите на холст компонент **Colour HSL (Цвета HSL)**	
24.	Подсоедините выход Start (S) (Начало) компонента **Deconstruct Domain (Разобрать Домен)** ко входу Hue (H) (Тон) компонента **Colour HSL (Цвета HSL)**	
25.	**Display/Preview/Custom Preview (Отображение/Предпромотр/Пользовальский Предварительный просмотр)** – Перетащите на холст компонент **Custom Preview (Пользовательский Предпросмотр)**	
26.	Кликните правой кнопкой мыши по входу Geometry (G) (Геометрия) компонента **Custom Preview (Пользовательский Предпросмотр)** и выберите Flatten (Обрубить Дерево Данных) Смотрите пп. 1-4 «Разработка с применением Деревьев Данных), чтобы узнать детали о уплощении данных (flattening)	
27.	Подсоедините выход Faces (F) (Грани) компонента **Deconstruct Brep (Разобрать Brep)** ко входу Geometry (G) (Геометрия) компонента **Custom Preview (Пользовательский Предпросмотр)**	
28.	Подсоедините выход Colour (C) (Цвет) компонента Colour HSL (Цвета HSL) ко входу Shade (S) (Шейдер) компонента Custom Preview (Пользовательский Предпросмотр)	





Для задания другого цветового эффекта, попробуйте подсоединить компонент Deconstruct Domain (Разобрать Домен) ко входам Saturation (S) (Насыщенность) или Luminance (L) (Светлота) компонента Colour HSL (Цвета HSL).





1.3.5. Булевские переменные (Booleans) и Логические Операторы (Logical Operators)

Загрузить файл примера, относящегося к данному разделу.

1.3.5.1. БУЛЕВСКИЕ ПЕРЕМЕННЫЕ (BOOLEANS)

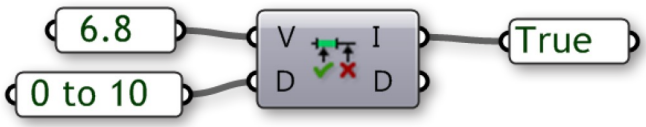
Числовые переменные могут содержать целый ряд различных чисел. Логические (булевы) переменные могут хранить только два значения, называемые «Yes» (Да) или «No» (Нет), «True» (ИСТИНА) или «False» (ЛОЖЬ), «1» или «0». Очевидно, что мы никогда не используем булевы переменные для выполнения вычислений из-за их ограниченного диапазона. Мы используем булевы переменные, чтобы оценить условия.

Boolean Parameter

В Grasshopper булевы операторы могут быть использованы несколькими способами. Параметр Boolean (Булевская Переменная) служит контейнером для одного или нескольких булевских значений, в то время, как Boolean Toggle (Логический Переключатель) позволяет быстро переключаться между единственным значением на выходе: истина или ложь.

Двойной клик по Boolean Toggle (Логическому Переключателю) позволяет быстро переключить значение на выходе между «True» (Истина) и «False» (Ложь)

Grasshopper также содержит объекты, способные проверять условия и выводить булевские (логические) значения. Например, компонент Includes (Содержит?) позволяет проверить числовое значение, чтобы увидеть, включено ли оно в домен (числовой диапазон) или нет.



Компонент Includes (Содержит?) проверяет, входит ли число 6.8 в числовой диапазон от 0 до 10 и возвращает логическое значение «True» (Истина).

1.3.5.2. ЛОГИЧЕСКИЕ ОПЕРАТОРЫ (LOGICAL OPERATORS)

Логические операторы, в основном, работают на булевских переменных, что действительно очень логично. Как Вы помните, булевские переменные могут иметь только два значения. Булева математика была разработана Джоном Булем (George Boole (1815-1864)) и сегодня она является основой всей цифровой индустрии. Булева алгебра даёт нам инструменты для анализа, сравнения и описания набора данных. Хотя Булем изначально было определено шесть логических операторов, мы будем обсуждать только три из них:

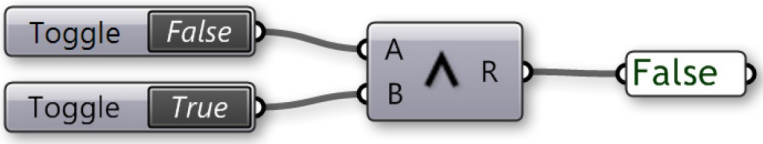
1. Not (НЕ)
2. And (И)
3. Or (ИЛИ)

Оператор Not (НЕ) является несколько необычным среди других операторов, поскольку не требует непременно двух значений. Вместо этого он инвертирует входящий. Представьте себе, что у нас есть скрипт, который проверяет наличие группы определений блока (Block definitions) в Rhino. Если определений блока нет, то мы хотим прервать сценарий и информировать об этом пользователя.

Grasshopper-оператор Not (Не) (логический элемент)

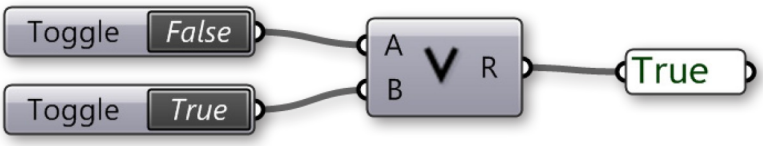
And (И) и Or (ИЛИ) — оба используют с одной стороны два оператора. Оператор And (И) требует, чтобы оба входящих были в состоянии True (ИСТИНА), чтобы результирующий оценился как True (ИСТИНА). Оператор Or (ИЛИ) выдаёт True (ИСТИНА), если хотя бы на одном из выходов будет True (ИСТИНА).

Как Вы можете видеть проблема с логическими операторами — не теория, это то, что происходит, когда Вам нужно их много, чтобы оценить что-то. Набирать их в последовательность быстро приводит к запутанности кода, не говоря уже о проблеме приоритета операторов.



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

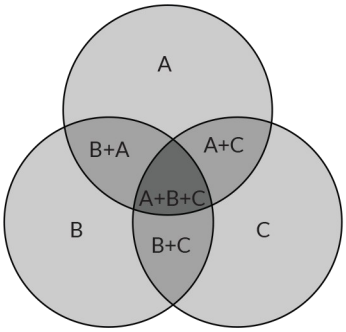
Grasshopper-оператор And (И) (логический элемент)

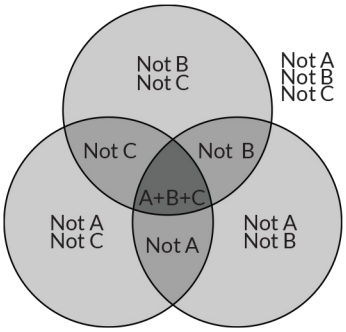


A	B	Result
True	True	True
True	False	True
False	True	True
False	False	False

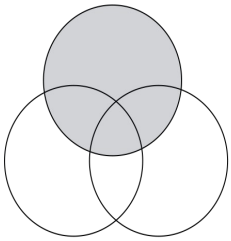
Grasshopper-оператор Or (ИЛИ) (логический элемент)

Хороший способ решить собственную задачу по булевой логике — использовать диаграмму Венна. Диаграмма Венна представляет собой графическое представление булевых наборов, где каждая область содержит (суб)набор значений, имеющих общее свойство. Наиболее известной из которых является трёхкруговая диаграмма:

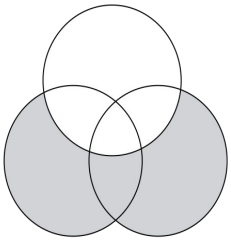




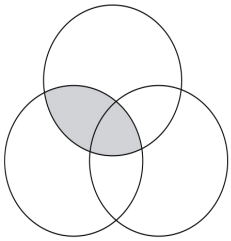
Каждая область круга содержит все значения, которые входят в набор. Верхний круг, например, помечен, как набор {A}. Каждое значение внутри этого круга оценивается как True (ИСТИНА) для {A} и каждое значение вне этого круга оценивается, как False (ЛОЖЬ) для {A}. Окрашивая эти области, мы можем имитировать логические оценки в программном коде:



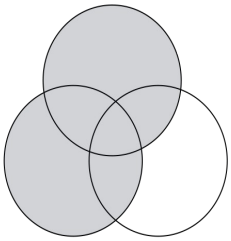
A



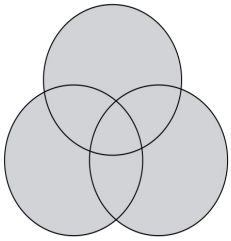
Not A



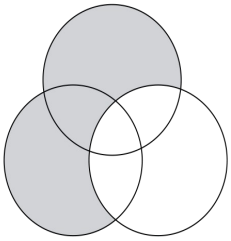
A And B



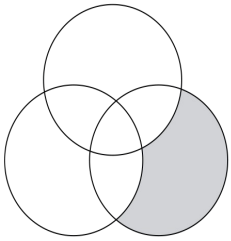
A Or B



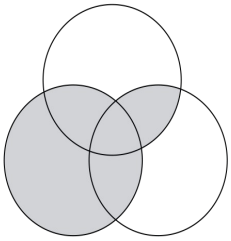
A Or B Or C



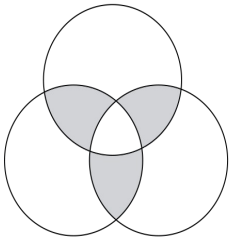
(A Or B) and Not C



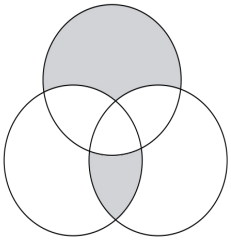
C And Not A and Not B



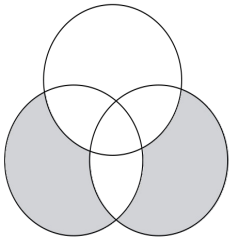
B Or (C And A)



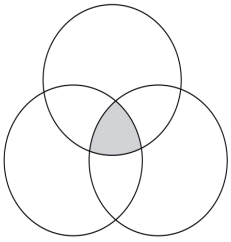
(A And B) Or
(B And C) Or (A And C)



((B And C) and Not A) Or
(A and Not B and Not C)



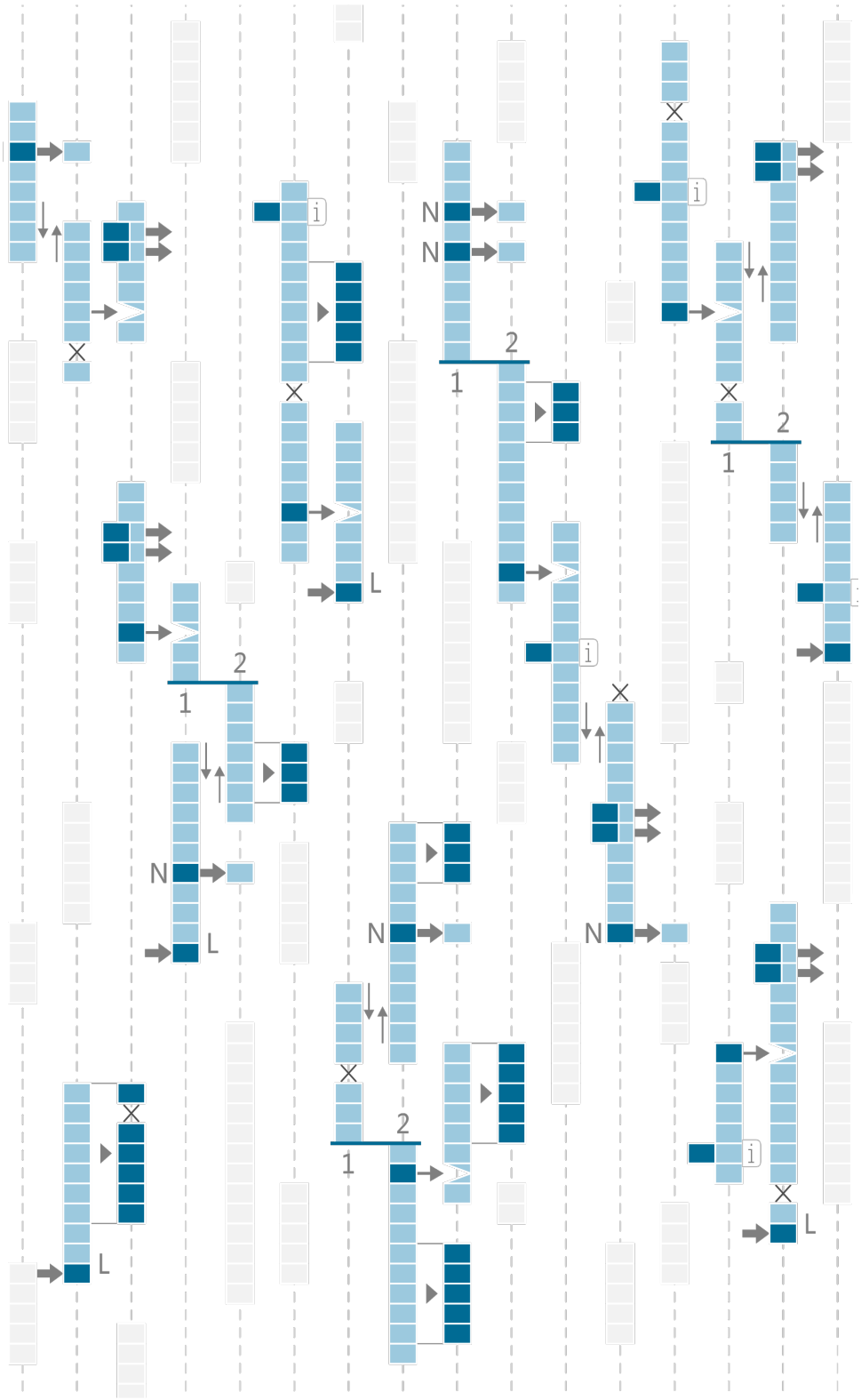
(B and Not C) Or
(C and Not B)



A And B And C

1.4. Конструирование с применением Списков (List)

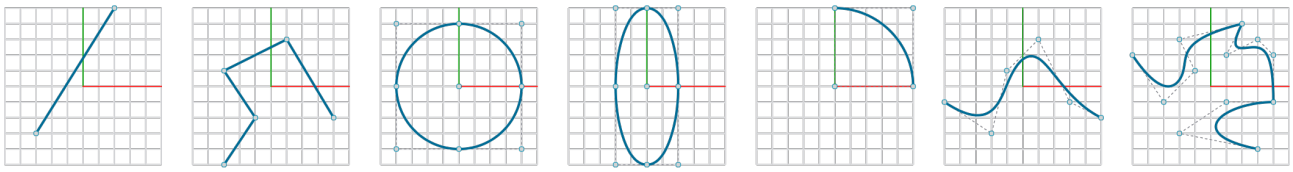
Одна из самых мощных функций Grasshopper — способность быстро создавать списки данных и манипулировать ими. В этой главе будет рассказано о том, как создавать и управлять списками данных. А также, как их визуализировать.



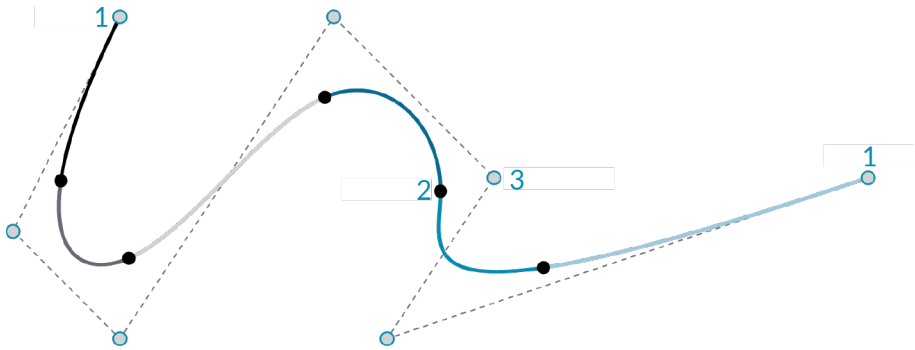
1.4.1. ГЕОМЕТРИЯ (GEOMETRY). КРИВЫЕ (CURVE).

NURBS (non-uniform rational B-splines *[не-однородные рациональные B-сплайны]*) являются математичекими представлением, которое может точно моделировать любую форму от простой 2D-геометрии (окружности, дуги и т.п.) и простых 3D-форм (параллелепипед, шар и т. п.) до сложных произвольных 3D-форм, таких, как органические поверхности или твёрдые тела. Благодаря своей гибкости и точности, NURBS-модели могут быть использованы в любом процессе от иллюстрации и анимации до промышленного производства.

Кривые (curves), как геометрические объекты, обладают рядом свойств и характеристик, которые могут быть использованы для их описания и анализа. Например, каждая кривая имеет координату начала и координату конца. Когда расстояние между этими координатами равно нулю, это значит, что эта кривая — замкнутая. Кроме того, каждая кривая имеет определённое количество контрольных точек (control points). Если эти точки расположены в одной плоскости, то и вся кривая в целом является плоской. Некоторые свойства применимы к кривой в целом, в то время, как другие применимы только к конкретным точкам на кривой. Например, является ли кривая плоской — это глобальное свойство, в то время, как векторы касательных — это локальное свойство. Кроме того, некоторые свойства распостраняются только на некоторые типы кривых. До сих пор мы рассуждали о Примитивах Кривых (Primitive Curve) Grasshopper. Это такие компоненты как: прямые (lines), окружности (circles), эллипсы (ellipses) и дуги (arcs).



1. Line (Прямая)
2. Polyline (Ломаная)
3. Circle (Окружность)
4. Ellipse (Эллипс)
5. Arc (Дуга)
6. NURBS Curve (NURBS-Кривая)
7. Polycurve (Составная кривая)



1. End Point (Точка Конца)
2. Edit Point (Редактирующая Точка)
3. Control Point (Контрольная Точка)

1.4.1.1. NURBS-КРИВЫЕ (NURBS CURVES)

Степень (Degree): Степень — это целое положительное число. Это число обычно равно 1, 2, 3 или 5, хотя это может быть любое положительное целое число. Степень кривой определяет область влияния контрольных точек на кривую: чем выше степень, тем больше область влияния. У NURBS-прямой или ломаной обычно степень равна 1, NURBS-окружность имеет степень 2, а большинство кривых произвольной формы имеют степень 3 или 5.

Контрольные Точки (Control Points): Минимальное число контрольных точек у кривой может быть не меньше, чем число степени + 1. Один из самых простых способов изменить форму NURBS-кривой — это переместить её контрольные точки.

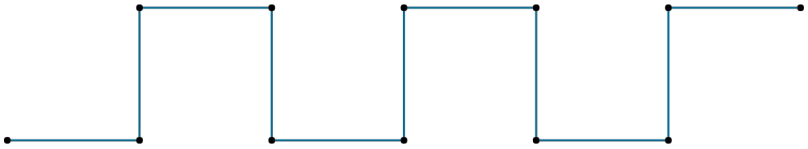
Вес (Weight): Контрольные точки имеют связанную характеристику, называемую «weight» (вес). Вес обычно является положительным числом. Когда контрольные точки кривой имеют одинаковый вес (обычно 1), то кривая называется «non-rational» (нерациональная), в иных случаях мы имеем дело с рациональными кривыми (rational curves). Большинство NURBS-кривых - нерациональные кривые. Но есть такие кривые, которые являются рациональными всегда: например, окружности и эллипсы

Узлы (Knots): Узлы — это список чисел (степень+N-1), где N — это количество контрольных точек.

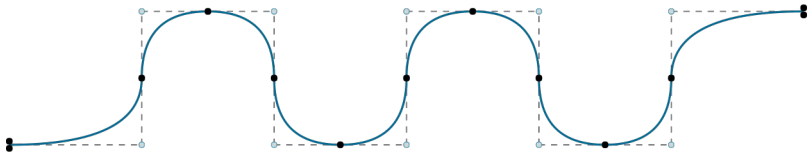
Редактирующие Точки (Edit Points): Точки на кривой вычисляются путём усреднения узлов. Редактирующие точки подобны контрольным точкам, за исключением того, что они всегда находятся на кривой и перемещение одной контрольной точки, как правило, ведёт к изменению формы всей кривой (перемещение же одной контрольной точки ведёт к локальному изменению формы

кривой). Редактирующие точки могут оказаться полезными, когда Вам нужно, чтобы определённая точка прошла через необходимое местоположение.

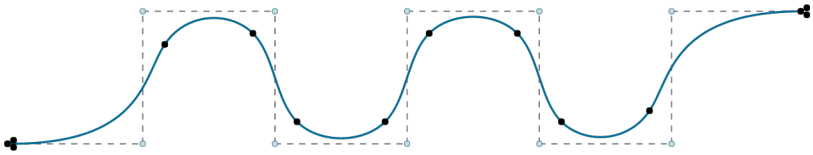
Результаты влияния смены числа степени NURBS-кривой на узлы:



> Стень = 1. NURBS-кривая ведёт себя также, как ломаная. Степень «1» кривой размещает узлы в местоположениях каждой контрольной точки.



> Степень = 2. Обычно используется при аппроксимации дуг и окружностей. Контрольный многоугольник (control polygon) пересекает сплайн посередине каждого сегмента NURBS-кривой.



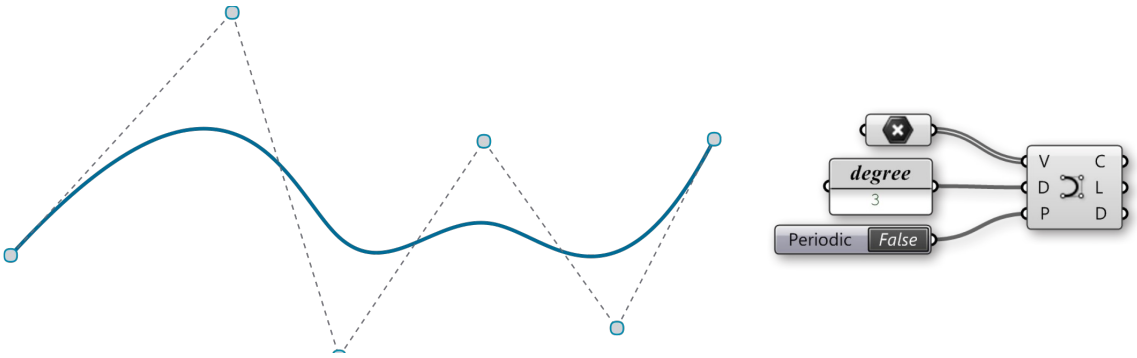
> Степень = 3. Это — наиболее распространённый тип NURBS-кривой и используется в Rhino по-умолчанию. Вы, вероятно, хорошо знакомы с визуализацией прогрессии в виде сплайна, хотя и кажется, что узлы находятся в странных местонахождениях.

1.4.1.2. КОМПОНЕНТЫ GRASSHOPPER-СПЛАЙНА

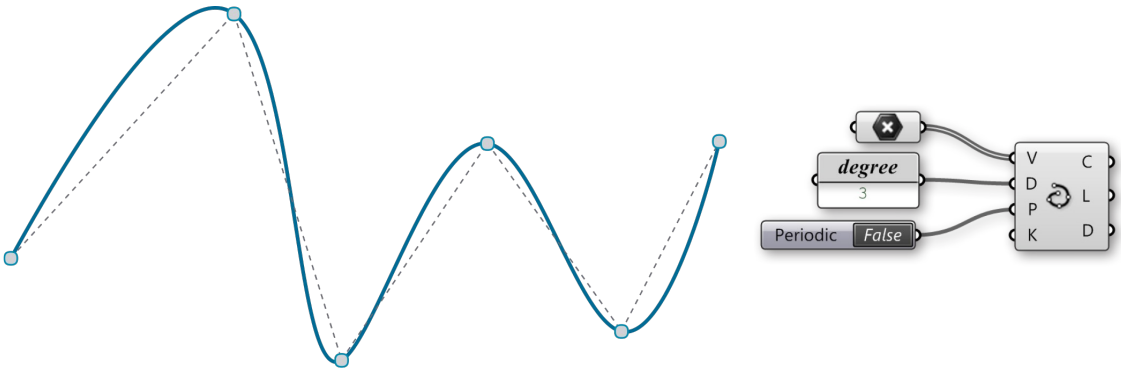
> Загрузить файл примера, относящегося к данному разделу.

Grasshopper содержит набор инструментов, способных выразить более продвинутые типы кривых Rhino, таких как NURBS-кривые и составные кривые (nurbs-curves и poly curves). Найти эти инструменты можно на вкладке Curve/Splines (Кривая/Сплайны).

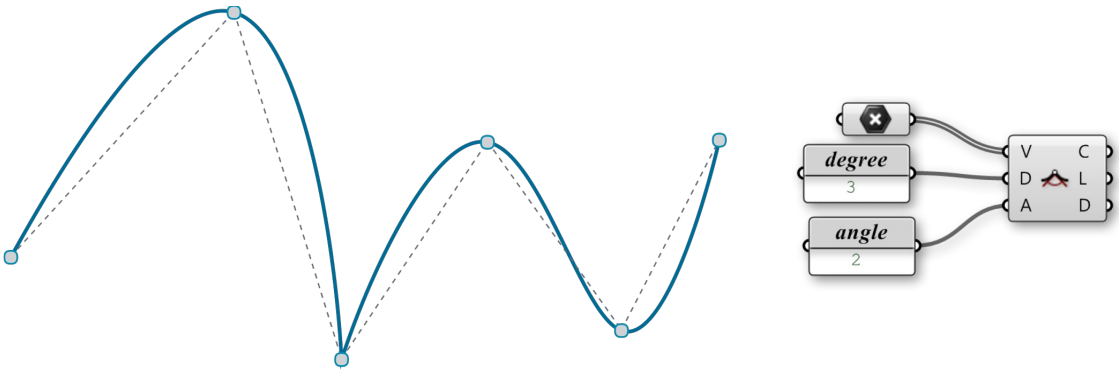
NURBS-Кривая (Nurbs Curve) (Curve/Spline/Nurbs curve (Кривая/Сплайн/NURBS-кривая)): Компонент Nurbs Curve конструирует NURBS-кривую из введённых контрольных точек (control points). Вход V определяет эти точки, которые могут быть описаны явным указанием точек в сцене Rhino или унаследованы в виде переменных данных от других компонентов. Вход D компонента Nurbs Curve задаёт степень кривой.



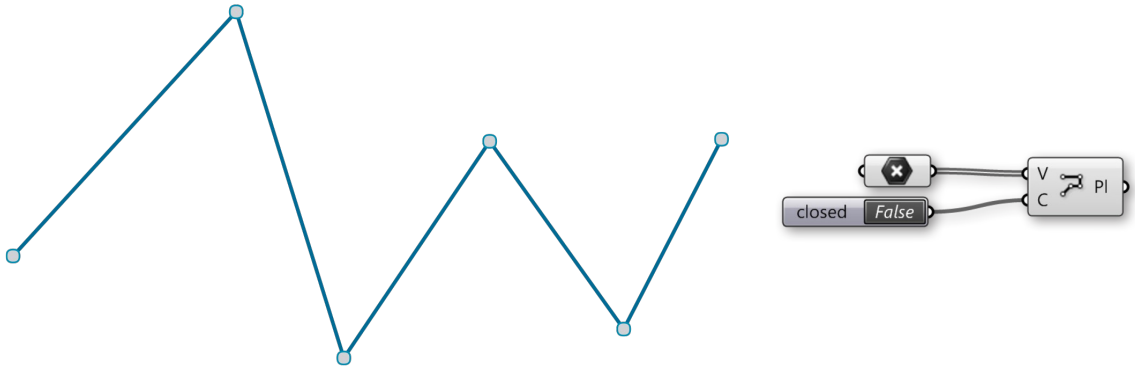
Интерполированная Кривая (Interpolate Curve) (Curve/Spline/Interpolate (Кривая/Сплайн/Интерполированная)): Интерполированные кривые ведут себя несколько иначе, чем NURBS-кривые. Вход V компонента подобен такому-же у компонента NURBS-curve, который тоже запрашивает набор точек для создания кривой. Однако, при использовании метода Интерполяции, результирующая кривая будет проходить непосредственно через эти точки, независимо от степени кривой. В компоненте NURBS curve мы могли этого достигнуть лишь убавлением числа степени кривой до единицы. Также, как и в компоненте NURBS curve, вход D определяет степень результирующей кривой. Однако, при таком способе построения, вход D принимает только нечётные числа. Кроме того, вход P определяет, будет ли кривая периодической (Periodic). Вы будете видеть некоторую шаблонность выходов у многих компонентов для создания кривых, в которых выходы C, L и D обычно определяют результирующую кривую, длину и домен кривой соответственно.



Кривая с Загибом (Kinky Curve) (Curve/Spline/Kinky Curve (Кривая/Сплайн/Кривая с Загибом)): Компонент Kinky Curve даёт возможность задать порог величины угла (вход A), при превышении которого состояние интерполированной кривой переходит из сглаженности в состояние кривой с резким загибом. Следует отметить, что вход «A» требует, чтобы угол был задан в радианах (radians).



Ломаная (Polyline) (Curve/Spline/Polyline (Кривая/Сплайн/Ломаная)): Ломаная представляет из себя набор прямых сегментов, соединяющих две и более точек. Результирующая линия всегда будет проходить через её контрольные точки, подобно интерполированной кривой. Как и у типов кривых, упомянутых выше, V-вход компонента Polyline (Ломаная) задаёт набор точек, определяющих границы каждого прямого сегмента ломаной. C-вход компонента определяет, будет ли ломаная замкнутой. Если местоположение первой точки не совпадает с местоположением последней, то, в таком случае, будет создан дополнительный сегмент, замыкающий петлю. Выход компонента Polyline (Ломаная) отличается от тех, что были у компонентов кривых в предыдущих примерах тем, что результатом является лишь сама кривая.

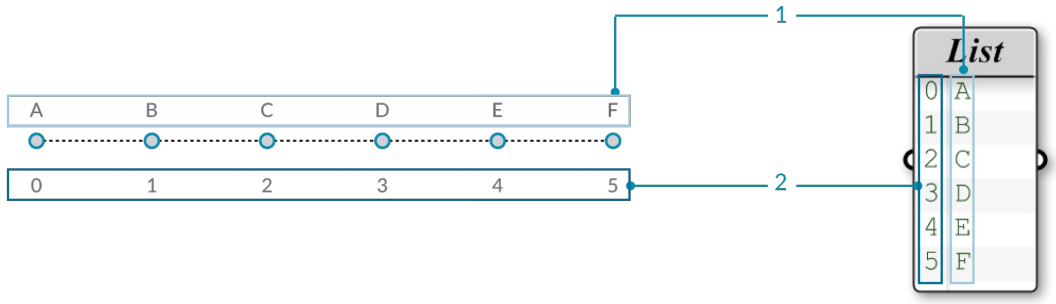


1.4.2. Что такое Список (List)?

О Grasshopper полезно думать как о потоке, так как графический интерфейс разработан в виде потока данных от одних компонентов к другим. Тем не менее, эти данные определяют информационный поток на входах и выходах компонентов. Понимание того, как управлять списками данных, имеет решающее значение для понимания работы плагина Grasshopper.

Как правило, Grasshopper оперирует двумя типами данных: постоянные и переменные (persistent и volatile). И хотя эти типы данных имеют разные характеристики, обычно Grasshopper хранит эти данные в массиве в виде списка переменных.

При хранении данных в виде списка полезно знать позицию каждого элемента в этом списке, чтобы быстро получить к ним доступ и возможность манипулировать необходимыми в данный момент элементами. Позиция элемента в списке называется index number (порядковый номер) *[или просто Индекс]*.

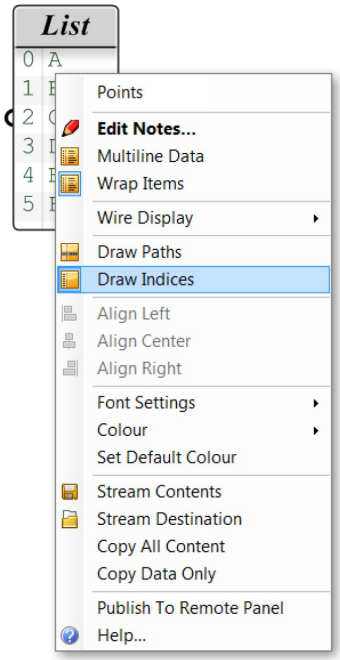


1. List Item (Элемент Списка)
2. Index (Индекс)

Единственное, что может показаться странным на первый взгляд это то, что первый порядковый номер в списке не единица, а ноль. Поэтому, когда мы говорим о первом элементе списка, то на самом деле, имеем ввиду элемент с порядковым номером (индексом) 0.

Например, если мы должны были сосчитать количество пальцев у нас на правой руке, скорее всего, мы бы насчитали от 1 до 5. Однако, если бы мы этот же список занесли в массив, тогда наш список бы уже считался от 0 до 4. Обратите внимание, что у нас по-прежнему 5 элементов в списке, просто массив использует систему подсчёта, основанную на ноле. Элементы, хранящиеся в списке совсем необязательно должны быть числами. Это могут быть данные любого типа, поддерживаемые Грасхопером, например, точками, кривыми, поверхностями, полигональными сетками и т.д.

Зачастую, самый простой способ взглянуть на тип данных, хранящихся в списке — это подсоединить к его выходу на конкретном компоненте Текстовую Панель (Text Panel (Params/Input/Panel (Параметры/Вводные/Панель)). По-умолчанию, Текстовая Панель автоматически отображает все индексы в левой части Панели, а элементы данных в правой части. Числовой индекс станет важным элементом, когда мы начнём работать с нашими списками. Вы можете включить и выключить отображение индексов кликом правой кнопкой мыши на текстовой панели в выбором пункта контекстного меню “Draw Indices” (Отображать Индексы). А в настоящее время давайте оставим включенным отображение индексов на наших текстовых панелях.



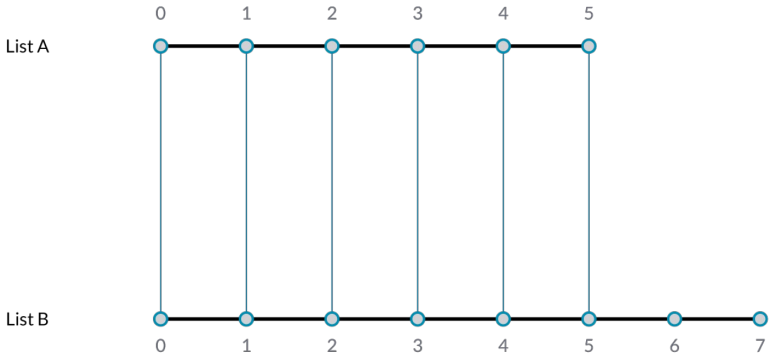
1.4.3. Сопоставление Потоков Данных (Data Stream Matching)

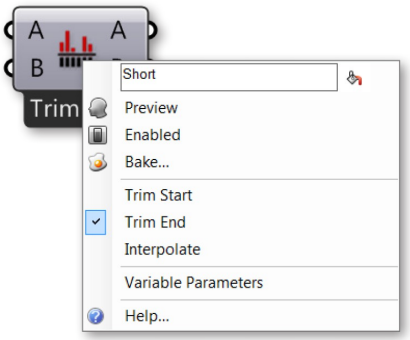
Сопоставление данных является проблемой, не имеющей чистого решения. Она возникает, когда компонент получает доступ ко входным данным разного размера списка. Изменение алгоритма сопоставления данных может привести к существенно различным результатам.

Представьте себе компонент, который создаёт прямые сегменты между точками. Он будет иметь два входный параметра, оба поставляющих координаты точек (Список A (List A) и Список B (List B)):

Как Вы можете видеть, существуют различные способы, с помощью которых мы можем выполнять построение прямых между этими двумя наборами точек. Новыми в Grasshopper 0.9 появились три компонента для сопоставления потоков данных, которые можно найти на панели Sets/List (Наборы/Список): Shortest List (Кратчайший Список), Longest List (Длиннейший Список) и Cross Reference (Перекрёстная Ссылка). Эти новые компоненты обеспечивают повышенную гибкость в пределах трёх основных алгоритмов сопоставления. Клик правой кнопкой мыши по каждому компоненту позволяет выбрать опцию сопоставления данных из контекстного меню.

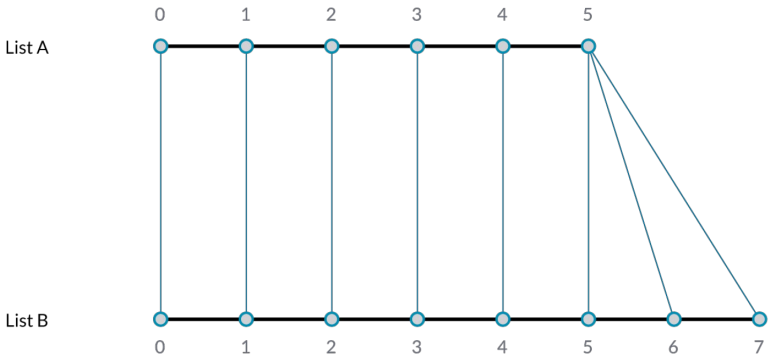
Самый простой способ — это подключить выходы один-к-одному, пока один из потоков не иссякнет. Такой алгоритм называется “Shortest List” (Кратчайший Список):



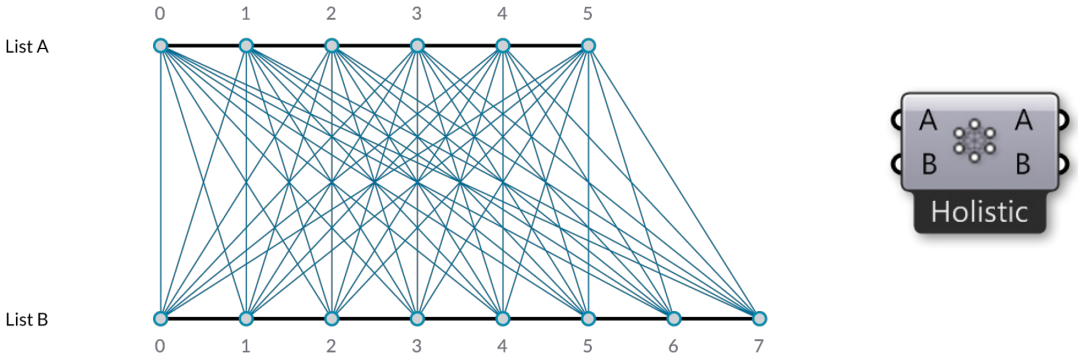


> Выбор опции алгоритма сопоставления через контекстное меню по клику правой кнопкой мыши на компоненте.

Алгоритм **“Longest List” (Длиннейший Список)** продолжает подключать входы, пока все потоки не иссякнут. Это поведение компонента по-умолчанию:

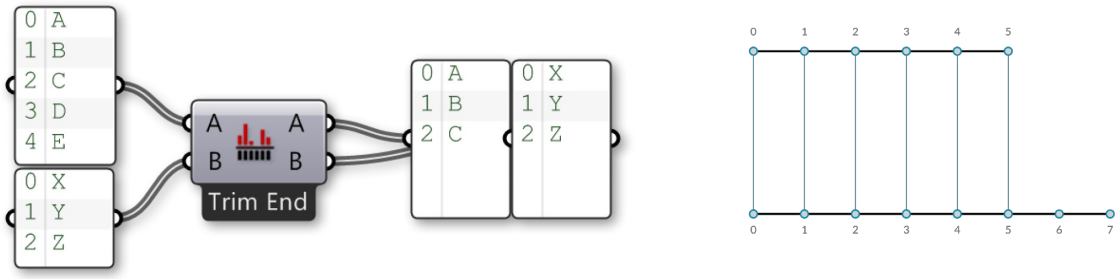


И, наконец, метод **“Cross Reference” (Перекрёстная Ссылка)** осуществляет все возможные подключения:

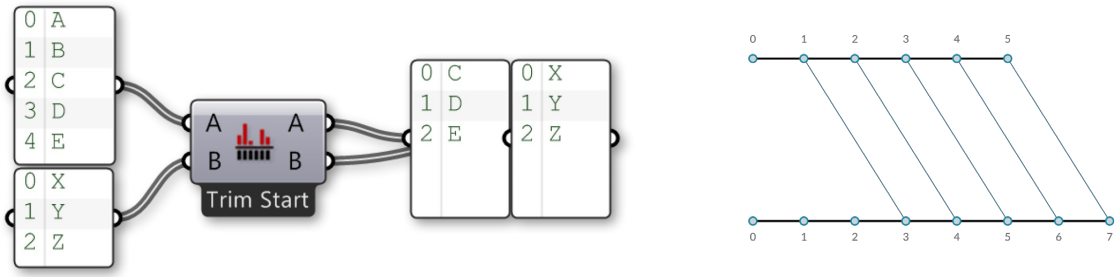


Это потенциально опасно, поскольку количество значений на выходе может быть огромным. Проблема ещё более усложняется, когда кроме вводных параметров участвуют ещё и наследуемые переменные данные. Если логическую схему оставить прежней, их количество начинает перемножаться.

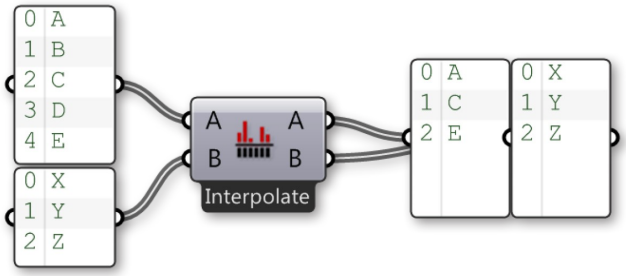
Давайте рассмотрим внимательнее самый короткий вариант использования компонента Shortest List (Кратчайший Список):

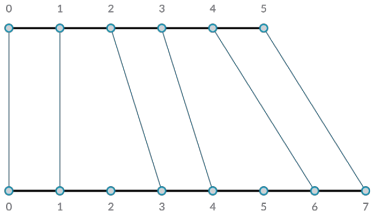


Здесь мы имеем два входных списка: {A,B,C,D,E} и {X,Y,Z}. При использовании опции Trim End (Обрезать Конец), последние два элемента из первого списка игнорируются и, таким образом, списки становятся как бы одинаковой длины.

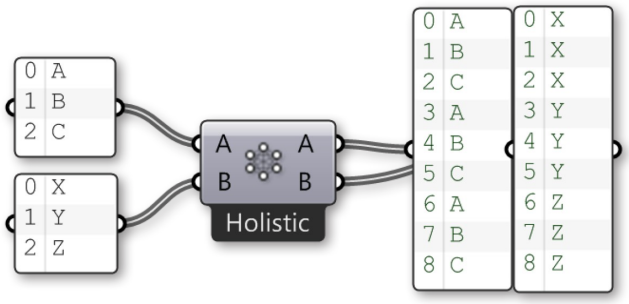


При использовании опции Trim Start (Обрезать Начало) первые два элемента первого списка будут игнорироваться, что уравняет длину списков.





Опция Interpolate (Интерполировать) пропустит второй и четвёртый элемент в первом списке. А теперь взглянем на компонент Cross Reference (Перекрёстная Ссылка):

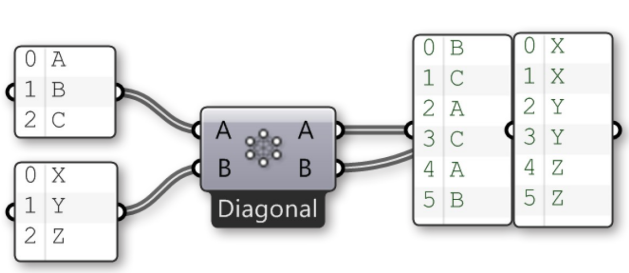


Здесь мы имеем два входных списка: {A,B,C} и {X,Y,Z}. Обычно, Grasshopper бы стал перебирать эти списки, рассматривая только их комбинации в формате {A,X}, {B,Y} и {C,Z}. Однако, существует ещё шесть комбинаций, которые обычно не просчитываются: {A,Y}, {A,Z}, {B,X}, {B,Z}, {C,X} и {C,Y}. Как Вы видите, на выходе из компонента Cross Reference (Перекрёстная Ссылка) мы имеем девять обязательных элементов перестановки.

Мы можем отобразить поведение Перекрёстной Ссылки, используя таблицу. Строки представляют первый список элементов, а столбцы — второй. Если мы воссоздадим все возможные варианты перестановок, то таблица будет иметь точку в каждой клетке, так как каждая клетка представляет собой уникальное сочетание двух источников индексов списков.

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

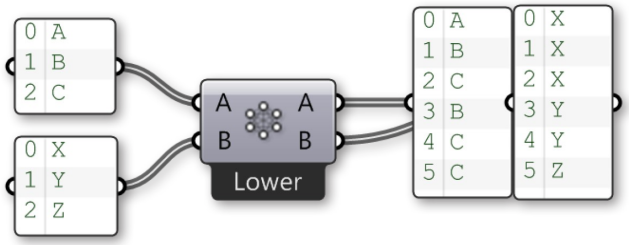
Однако, иногда, Вам не нужны все возможные перестановки. Иногда Вы хотите исключить определённые области, потому что они приведут к бессмысленным или некорректным вычислениям. Общий принцип исключения состоит в том, чтобы игнорировать все ячейки, находящиеся на диагонали таблицы. Изображение выше показывает «целостное» сопоставление данных, в то время как опция «Diagonal» (Диагональ)(доступная из контекстного меню компонента Cross Reference) имеет пробелы для {0,0}, {1,1}, {2,2} и {3,3}. Если мы применим это к нашим наборам {A,B,C} и {X,Y,Z}, что были в примере, то мы должны ожидать сокрытие комбинаций для {A,X}, {B,Y} и {C,Z}:



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

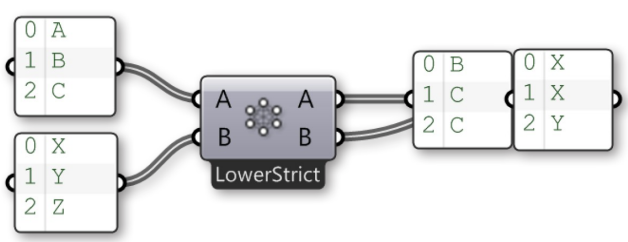
Правило, которое применяется к «диагональному» сопоставлению: «Игнорировать все перестановки, где элементы имеют одинаковые индексы списка». «Coincident» (Совпадающий) сопоставляет данные двух входных списков таким же образом, как «Диагональ», но правило несколько отличается: «Игнорировать все перестановки, где любые два элемента списка имеют одинаковый индекс».

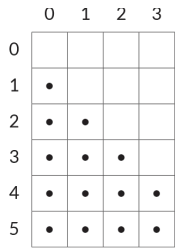
Четыре оставшиеся алгоритма являются вариациями на ту же тему. «Lower triangle» (Нижний Треугольник) применяет правило соответственно: «Игнорировать все перестановки, где индекс элемента меньше, чем индекс элемента в следующем списке» . Результатом мы имеем пустой треугольник, но с элементами на диагонали.



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

«Lower triangle (strict)» (Нижний Треугольник (строго)) идёт на шаг раньше и исключает элементы уже и с главной диагонали:





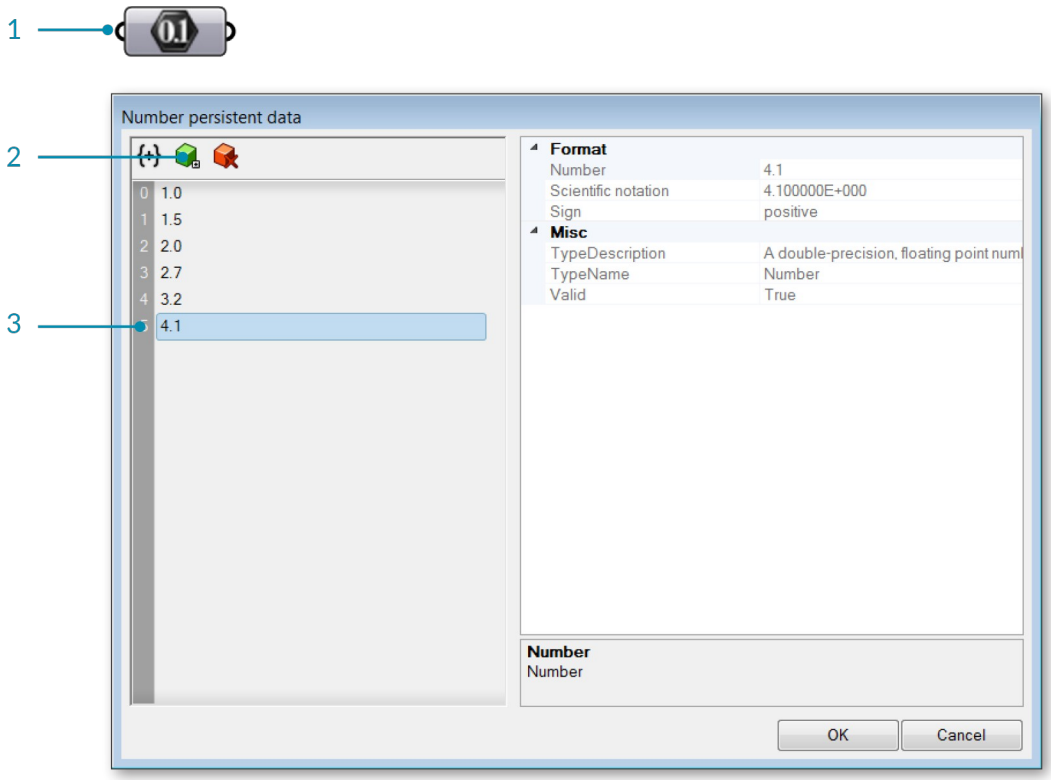
«Upper Triangle» (Верхний Треугольник) и «Upper Triangle (strict)» (Верхний Треугольник (строго)) являются зеркальным отображением двух предыдущих алгоритмов, в результате чего пустые треугольники оказываются по другую сторону от линии диагонали.

1.4.4. Создание Списков (Lists)

В Grasshopper есть много различных способов создания списков. Ниже мы рассмотрим несколько разных методов, а затем взглянем на то, как данные могут использоваться, чтобы передать информацию в окно проекции (viewport) посредством визуализации.

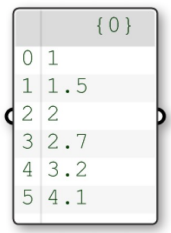
1.4.4.1. РУЧНОЕ СОЗДАНИЕ СПИСКА

Возможно, самый простой способ создания списка (и один из самых недооценённых) — это ручной ввод списка значений в параметр. Использование этого метода накладывает на пользователя дополнительную ответственность, так как опирается на введённые им данные напрямую. В таком случае для создания списка используется постоянные данные (persistent data). Чтобы изменить значения в списке пользователь должен сделать это с каждым значением вручную. Это может быть затруднено, если элементов в списке много. Есть несколько способов создания списка вручную. Одним из способов является использование числового параметра (Number paramter). Щёлкните правой кнопкой мыши по числовому параметру и выберите “Manage Number Collection” (Управление Набором Чисел).



1. Кликните правой кнопкой мыши по компоненту Число (Number), чтобы открыть Number collection Manager (Управление Набором Чисел).
2. Чтобы добавить число в список кликните по иконке Add Item (Добавить Элемент).
3. Чтобы изменить значение элемента — дважды кликните по нему.

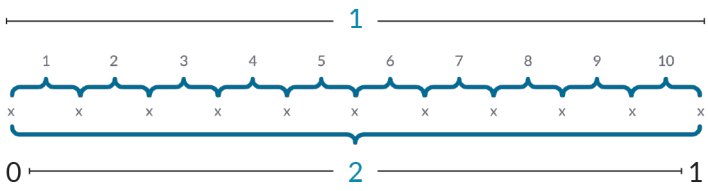
Другой метод ручного создания списка элементов — ввод в Panel (Текстовую Панель). Убедитесь, что кнопка “Multiline Data” (Многострочные Данные) отжата.



1.4.4.2. RANGE (ДИАПАЗОН)

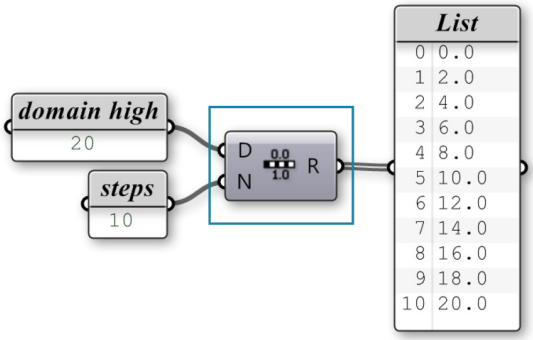
Компонент Range (Диапазон) можно найти на вкладке Sets/Sequence/Range (Наборы/Последовательность/Диапазон). Он создаёт список чисел, значения которых равномерно распределены, между заданными наименьшим и наибольшим значениями, называемый диапазоном. Диапазон также иногда упоминается как домен или интервал — при упоминании этих понятий речь всё время идёт о всех возможных числах между двумя числовыми экстремумами (пределами).

Компонент Range (Диапазон) разделяет числовой домен на равные сегменты и возвращает список значений.



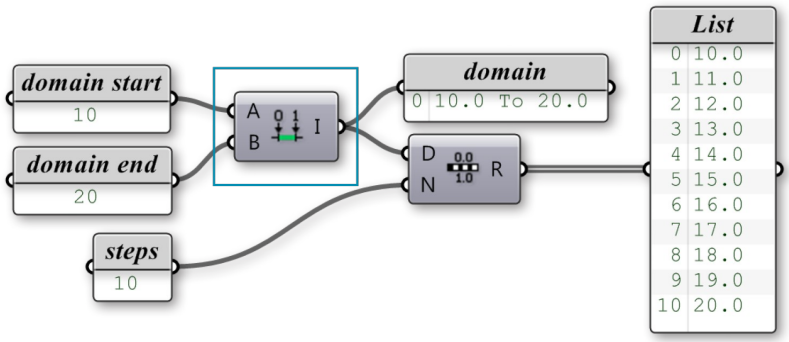
1. Количесво Шагов (Steps) = 10
2. Домен простирается от 0 до 1
3. Общее количество точек = 11

В приведённом ниже примере числовой домен был определён как каждое возможное число между 0 и 20. Компонент Range (Диапазон) берёт этот домен и делит его на заданное число шагов (в данном случае 10). Таким образом мы получили 10 отдельных сегментов. Компонент Range (Диапазон) возвращает список значений. Так как он удерживает первое и последнее значение в списке, то на выходе из компонента Range (Диапазон) мы всегда имеем на 1 элемент больше, чем задавали количество шагов (Steps). В приведённом выше примере мы задали 10 шагов, поэтому Range (Диапазон) вернул 11 значений.



Создание списка с использованием компонента Range (Диапазон) с указанием Домена (Domain) и количеством шагов.

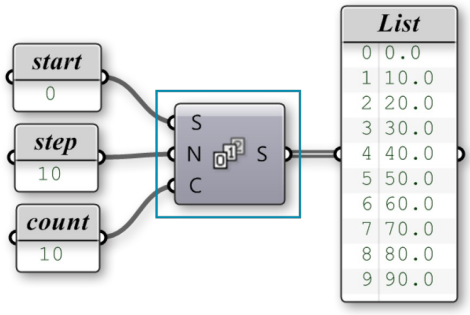
Возможно, Вы обратили внимание на некую причудливость сделанной нами конструкции. Мы знаем, что домен всегда определяется двумя значениями (верхним и нижним). Тем не менее, в нашем дефинишине мы просто подсоединили единичное значение ко входу домена. Для того, чтобы избежать ошибок, Grasshopper делает предположение, что Вы пытаетесь определить домен между нулём и каким-то вашим числом (значение с вашего слайдера). Для того, чтобы создать домен, не начинающийся с нуля, мы должны использовать компонент Construct Domain (Сконструировать Домен).



Для создания Диапазона, начинающегося не с нуля, используйте компонент Construct Domain (Сконструировать Домен).

1.4.4.3. SERIES (СЕРИЯ)

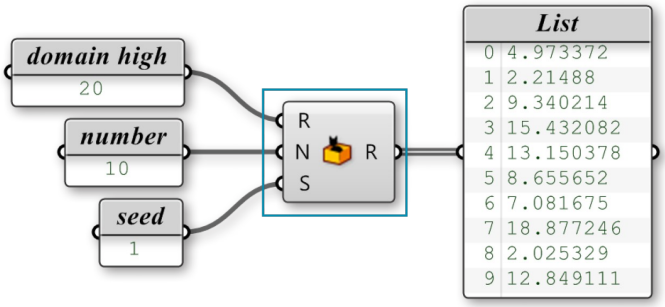
Компонент Series (Серия) подобен компоненту Range (Диапазон) тем, что тоже создаёт список чисел. Однако, отличен тем, что создаёт набор дискретных чисел., базирующийся на начальном значении, величине шага и количестве значений в серии.



Компонент Series (Серия) создаёт список, основанный на начальном значении (start), величине шага (step) и количестве значений в списке (count).

1.4.4.4. RANDOM (СЛУЧАЙНОЕ)

Компонент Random (Sets/Sequence/Random (Наборы/Последовательность/Случайная)) может быть использован для генерирования списка псевдослучайных чисел. Они называются «псевдо» случайными потому, что числа последовательности являются уникальными, но стабильно зависимыми от начального значения (seed). Таким образом, Вы можете сгенерировать совершенно новый набор случайных чисел, путём изменения начального значения (S-вход). Домен, как и в предыдущем примере, представляет собой определённый интервал между двумя числовыми крайностями.

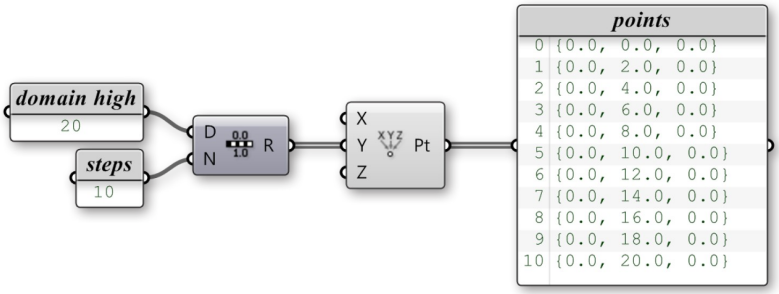


1.4.5. Визуализация Списка

Загрузить файл примера, относящегося к данному разделу.

Понимание работы со списком в Grasshopper может оказаться сложным без возможности воспринять визуально данные, перетекающие из одного компонента в другой. Существует несколько способов отображения данных списка, которые помогут их лучше понять и эффективнее ими управлять.

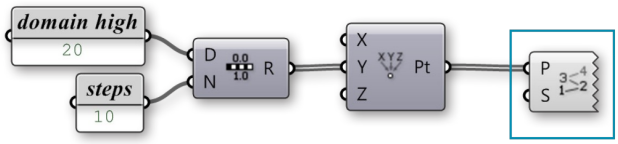
Есть много различных способов визуализации списка данных. Наиболее распространённым является создание некоторой геометрии, связанной с данными списка. Соединив выход R компонента Range (Диапазон) со входом Y компонента the Construct Point (Сконструировать Точку), Вы сможете увидеть массив точек в Y-направлении.



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

1.4.5.1. КОМПОНЕНТ POINT LIST (СПИСОК ТОЧЕК)

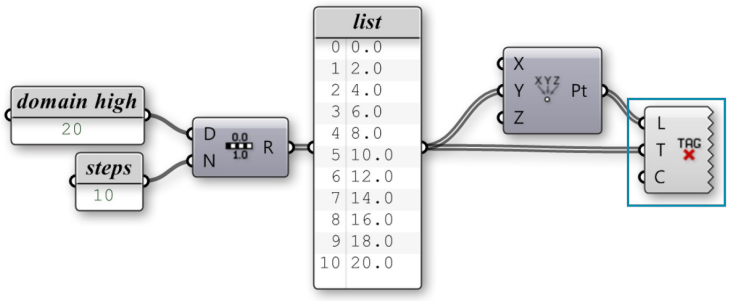
Компонент Point List (Список Точек) является чрезвычайно полезным инструментом для визуализации порядка набора точек в список. По существу, компонент Point List (Список Точек) размещает индекс числового элемента рядом с геометрической точкой во вьюпорте (номер индекса пункта рядом с точечной геометрией в окне просмотра). Вы также можете задать будут или нет отображаться теги чисел, линии связей, а также скорректировать размер надписей тегов.



Вы можете визуализировать порядок точек в наборе, используя компонент Point List (Список Точек)

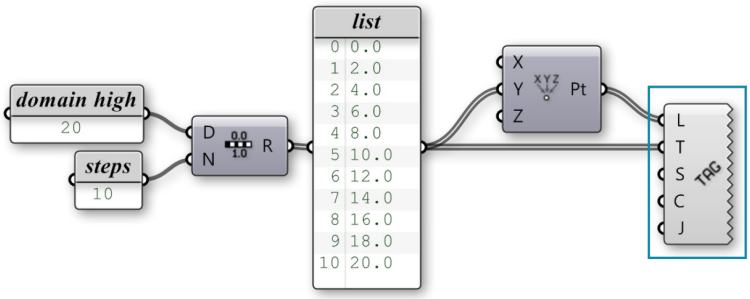
1.4.5.2. TEXT TAGS (ТЕКСТОВЫЕ ТЕГИ)

Компонент Text Tag (Текстовые Теги) позволяет отрисовывать небольшие строки (строки, состоящие из набора символов ASCII) во вьюпорте в качестве обратной связи с элементами. Текст и местоположение могут быть заданы в качестве входных параметров. Если текстовые теги запечь (bake) в сцену Rhino, то они превратятся в Text Dots (Точки Аннотации). Другая интересная особенность текстовых тегов это то, что они независимы от изменений, происходящих во вьюпорте, то есть они всегда направлены в сторону камеры (включая перспективные виды), также они остаются постоянного размера вне зависимости от настроек масштабирования вьюпорта.



Используя компонент Text Tag (Текстовые Теги) Вы можете визуализировать любую строку информации во вьюпорте Rhino. В данном случае мы решили отобразить значение каждой точки поверх её местоположения. Мы могли назначить для отображения любой другой текст.

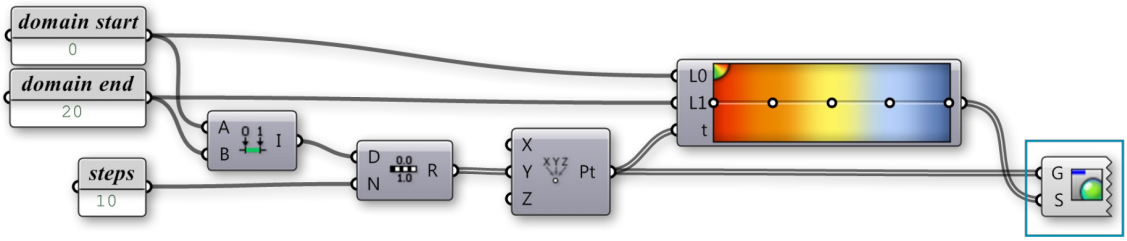
Компонент Text Tag 3d (Текстовый Тег 3D) работает очень похожим образом, как и компонент Text Tag (Текстовой Тег). Они отличаются тем, что при запекании объектов (bake), генерируемых компонентом Text Tag 3d (Текстовый Тег 3D)в сцену Rhino они становятся текстовыми объектами (Text objects). Масштаб шрифта в Text Tag 3d (Текстовый Тег 3D) может управляться через вход Size (S) (Размер), (который недоступен у компонента Text Tag (Текстовый Тег)).

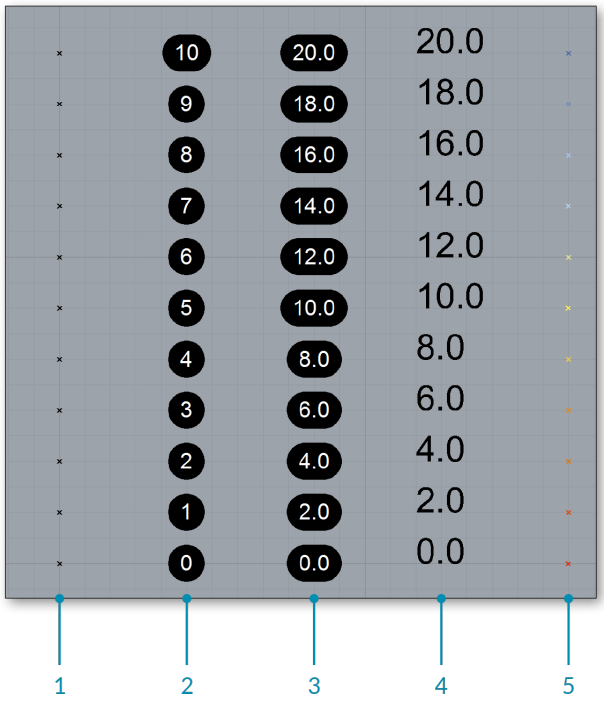


> Вы можете использовать компонент Text Tag 3d (Текстовой Тег 3D), чтобы визуализировать информацию подобно Текстовым объектам (Text object) в Rhino.

1.4.5.3. ЦВЕТ (COLOR)

Одна из других возможностей визуализации списка данных — это назначение геометрии цвета. Grasshopper имеет ограниченные возможности «рендеринга», но мы можем управлять простыми параметрами Open GL, такими, как цвет (color), рефлекс (specular color), прозрачность (transparency) и т.д. Значение L0 представляет нижний предел (левая сторона) градиента, в то время, как L1 представляет верхний предел (правая сторона). Эти значения соответствуют началу и концу нашего домена. Т-значенмя элементов списка будут сопоставляться в диапазоне от L0 до L1. На выход градиента поступает список значений цвета в цветовом пространстве RGB, соотносящихся с каждой точкой из списка. По клику правой кнопкой мыши на градиенте можно выбрать один из предустановленных видов градиента или определить свой собственный, используя захваты настроек цвета.





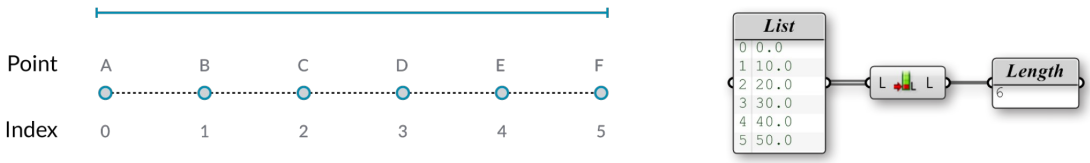
1. Points (Точки)
2. Point list (Список Точек)
3. Text Tag (Текстовые Теги)
4. Text Tag 3D (Текстовые Теги 3D)
5. Custom color preview (Цвет предпросмотра задан пользователем)

1.4.6. List Management (Управление Списком)

Одна из самых мощных возможностей Grasshopper — это способность быстро создавать различные списки данных и манипулировать ими. Мы можем хранить в списке множество различных типов данных (числа, точки, векторы, кривые, поверхности, брепы и т. д.). И в подкатегории Sets/List (Наборы/Список) Вы можете найти ряд полезных инструментов.

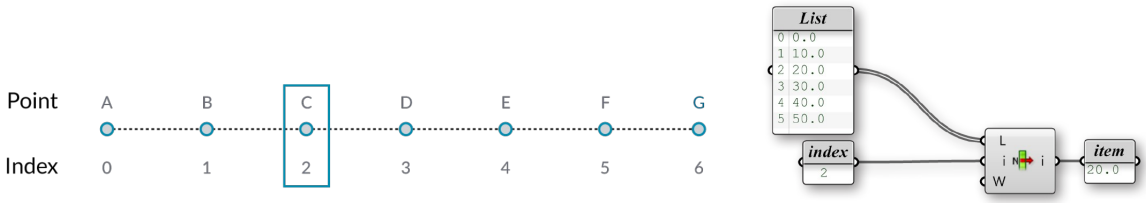
1.4.6.1. LIST LENGTH (ДЛИНА СПИСКА)

Компонент List Length (Длина Списка) (Sets/List/List Length (Наборы/Список/Длина Списка)) по существу измеряет длину списка. Так как наши списки всегда начинаются с нуля, максимально возможный индекс в списке будет равен длине списка минус один. В этом примере мы подключили наш базовый список ко входу L компонента List Length (Длина Списка), показывающему, что в списке 6 значений.



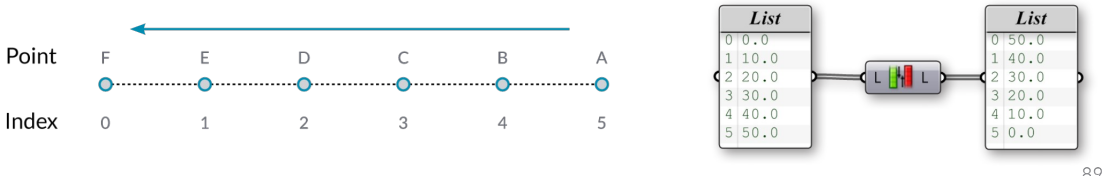
1.4.6.2. LIST ITEM (ЭЛЕМЕНТ СПИСКА)

Наш список поступает в компонент List Item (Элемент Списка) (Sets/List/List Item (Наборы/Список/Элемент Списка)) в целях извлечения конкретного элемента данных из набора данных. Для доступа к отдельным элементам списка мы должны подать на i-вход соответствующие числа индексов тех элементов, которых бы хотели получить. Мы можем скормить ему целое число или список целых чисел на i-вход, в зависимости от того, сколько элементов мы бы хотели получить. L-вход определяет базовый список, который мы будем анализировать. В этом примере мы имеем на входе набор до 5.0 элементов и компонент List Item (Элемент Списка) возвращает элемент данных, ассоциированный с пятым номером в списке.



1.4.6.3. REVERSE LIST (ОБРАТИТЬ СПИСОК)

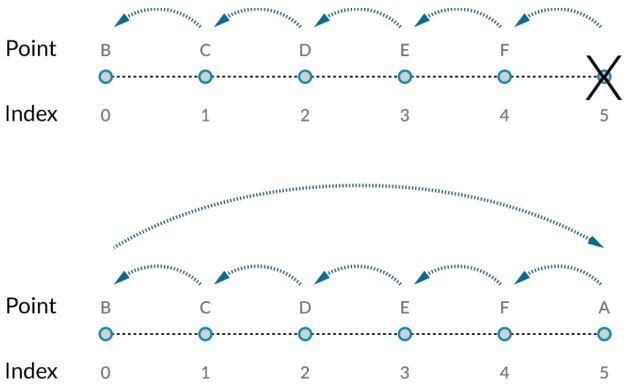
Мы можем инвертировать (обратить) порядок в нашем списке, используя компонент Reverse List (Обратить Список) (Sets/List/Reverse (Наборы/Список/Обратить Список)). Если мы введём список возрастающих чисел от 0.0 до 9.0 в компонент Reverse List (Обратить Список), то на выходе из него получим список нисходящих чисел от 9.0 до 0.0.

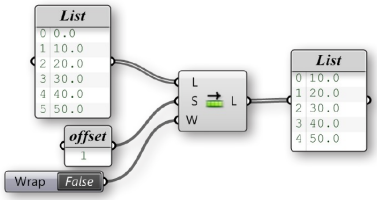


1.4.6.4. SHIFT LIST (СДВИГ СПИСКА)

Компонент Shift List (Сдвиг Списка) (Sets/Sequence/Shift List (Наборы/Последовательность/Сдвиг Списка)) будет смещать элементы по списку вверх или вниз на величину значения шага смещения. Мы подключили выход List

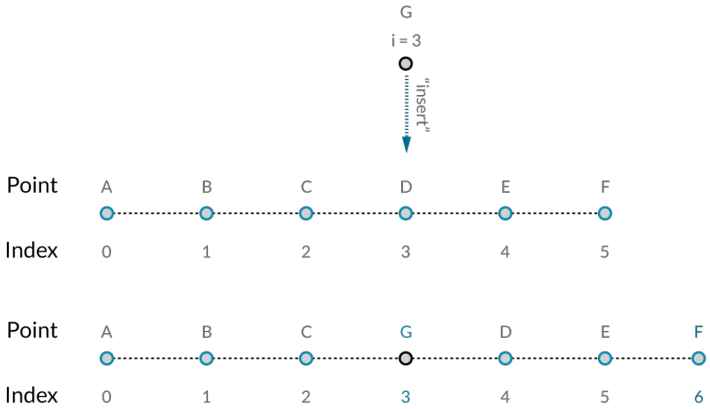
(Список) к L-входу компонента Shift-List (Сдвиг Списка). Если весь список будет сдвигаться на величину -1, то все значения в списке переместятся вниз на один номер индекса от того, что был на входе. Точно также, если мы изменим величину сдвига (offset) на +1, то все значения в списке сместятся вверх по списку индексов на одно значение. Если на вход Wrap (Завернуть) входит логический параметр со значением True (ИСТИНА), то элементы, которым не хватило индексов в конце списка и они должны были как бы отпасть (игнорироваться), будут завёрнуты из конца в начало списка. Так же и наоборот: если список смещается вверх, то выпавшие параметры из верхней части будут перемещены в нижнюю часть списка. Если значение на входе Wrap (Заворачивать) будет False (ЛОЖЬ), то, как в нашем примере, список просто укоротится вместе с «выпавшими» значениями.

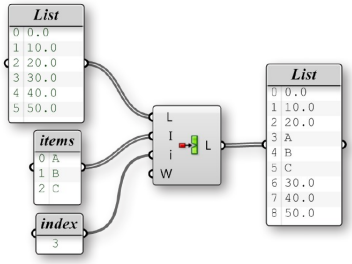




1.4.6.5. INSERT ITEMS (ВНЕДРЕНИЕ ЭЛЕМЕНТОВ)

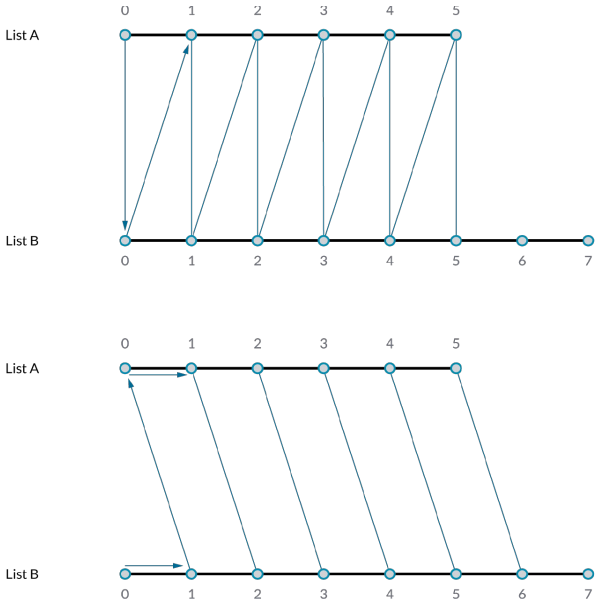
Компонент Insert Items (Внедрить Элементы) (Sets/Lists/Insert Items (Наборы/Список/Внедрить Элементы)) даёт возможность добавить к вашему списку ещё один набор элементов, вставив их в указанные места списка. Для того, чтобы это работало должным образом, Вы должны знать, какие именно элементы Вы хотите вставить и индекс позиции для каждого нового элемента. На примере ниже иы будем внедрять буквы A, B, C в позицию с индексом три.

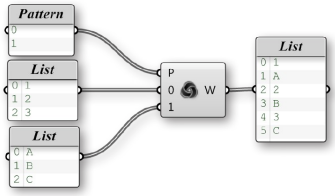




1.4.6.6. WEAVE (СПЛЕСТИ)

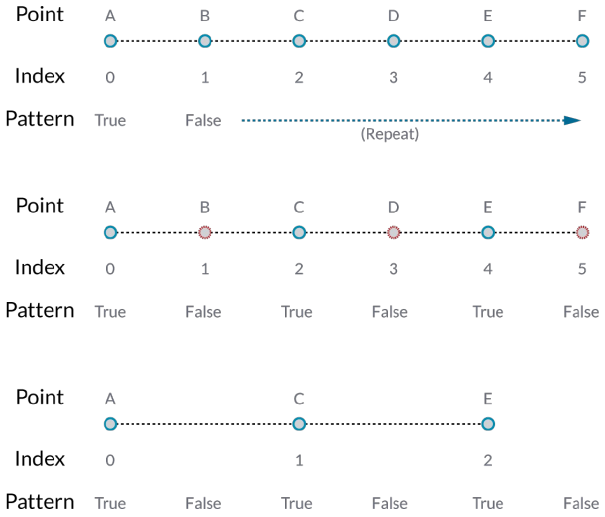
Компонент Weave (Сплести) (Sets/Lists/Weave (Наборы/Списки/Сплести) обеспечивает слияние двух или нескольких списков в один, основываясь на указанном шаблоне сплетения (P-вход). Если шаблон (pattern) и потоки (streams) не совпадают идеально, этот компонент может добавлять нули в исходящие потоки, либо может игнорировать уже истощившиеся потоки.





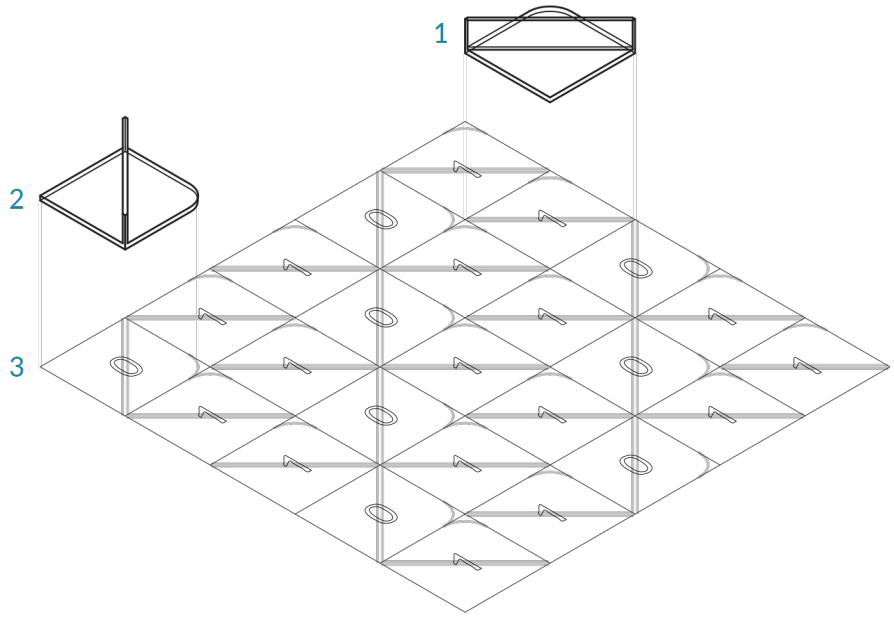
1.4.6.7. CULL PATTERN (ШАБЛОН ОТБРАСЫВАНИЯ)

Компонент Cull (Отбрасывание) (Sets/Sequence/Cull Pattern (Наборы/Последовательность/Шаблон Отбрасывания)) удаляет элементы из списка, используя повторяющуюся битовую маску. Битовая маска задаётся в виде списка логических значений (True (ИСТИНА) или False (ЛОЖЬ)). Битовая маска используется снова и снова, пока все элементы в списке не будут оценены.



1.4.7. РАБОТА СО СПИСКАМИ

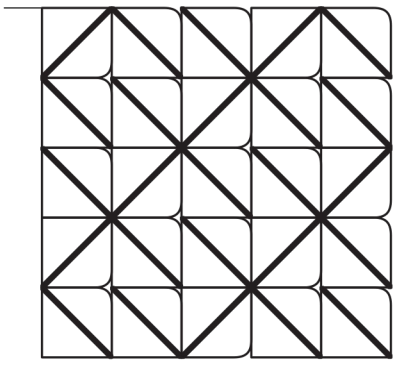
Давайте рассмотрим пример, используя компоненты из предыдущего раздела. В этом примере мы создадим плиточный шаблон, накладывающий геометрию на прямоугольную сетку. Шаблон будет создаваться, используя компонент List Item (Элемент Списка) для получения желаемого геометрического рисунка.



1. Геометрия, сопоставляемая с индексом 1
2. Геометрия, сопоставляемая с индексом 0
3. Прямоугольная сетка

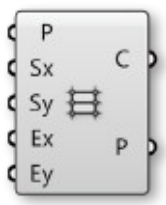
0	1	1	0	1
1	1	0	1	1
1	0	1	1	0
0	1	1	0	1
1	1	0	1	1

1

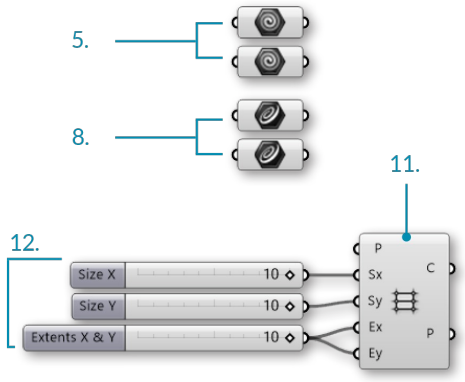


2

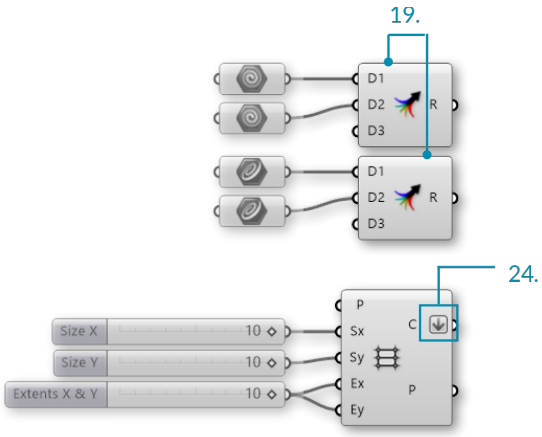
1. Шаблон наложения
2. Накладываемая геометрия

1.	Начните новый файл Rhinoceros.	
2.	Создайте два квадрата равного размера.	
3.	Создайте в каждом квадрате различные геометрические формы. В примере, показанном выше, мы создали простую поверхность со срезанным уголком. Уголки срезаны, чтобы продемонстрировать ориентацию двух исходных поверхностей и изменение ориентации распределённых и сориентированных результирующих поверхностей.	
4.	Начните новый дефинишин, введя Ctrl+N (в Grasshopper)	
5.	**Params/Geometry/Geometry (Параметры/Геометрия/Геометрия)** – Перетащите на холст два параметра **Geometry (Геометрия)**.	
6.	Кликните правой кнопкой мыши на первом параметре Curve (Кривая) и выберите Set One Curve (Выбрать Одну Кривую). Выберите первый квадрат, используемый в качестве ссылочного.	
7.	Кликните правой кнопкой мыши на втором параметре **Geometry (Геометрия)** и выберите Set One Geometry (Выбрать Один объект Геометрии). Укажите второй геометрический объект в качестве ссылочного. Можно использовать несколько (multiple) геометрических объектов в качестве ссылочных в одном-единственном параметре, но для простоты понимания мы использовали два раздельных параметра.	
8.	**Params/Geometry/Curve (Параметры/Геометрия/Кривая)** – Перетащите на холст два параметра **Curve (Кривая)**.	
9.	Кликните правой кнопкой мыши на первом параметре **Curve (Кривая)** и выберите Set One Curve (Выбрать Одну Кривую). Укажите первый квадрат, используемый в качестве ссылочного.	
10.	Кликните правой кнопкой мыши на втором параметре **Curve (Кривая)** и выберите Set One Curve (Выбрать Одну Кривую). Укажите первый квадрат, используемый в качестве ссылочного. Убедитесь, что габариты геометрии и ссылочный квадрат соответствуют.	
11.	**Vector/Grid/Rectangular (Векторы/Сетка/Прямоугольная)** – Перетащите на холст компонент **Rectangular Grid (Прямоугольная Сетка)**.	

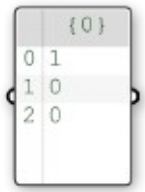
12.	**Params/Input/Slider (Параметры/Вводные/Слайдер)** — Перетащите три **Числовых Слайдера** на холст.	
13.	Дважды кликните на первом **Числовом Слайдере** и задайте следущее: Rounding (Округление): Integers (Целые числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 10 Value (Значение): 10	
14.	Дважды кликните на втором **Числовом Слайдере** и задайте следущее: Rounding (Округление): Integers (Целые числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 10 Value (Значение): 10	
15.	Дважды кликните на третьем **Числовом Слайдере** и задайте следущее: Name (Имя): Extents X & Y (Протяжённость по X и Y) Rounding (Округление): Integers (Целые числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 10 Value (Значение): 10	
16.	Подсоедините первый **Числовой Слайдер** ко входу Size X (Sx) (Размер по X) компонента **Rectangular Grid (Прямоугольная Сетка)**.	
17.	Подсоедините второй **Числовой Слайдер** ко входу Size Y (Sy) (Размер по Y) компонента **Rectangular Grid (Прямоугольная Сетка)**.	
18.	Подсоедините третий **Числовой Слайдер** ко входам Extent X (Ex) (Протяжённость по X) и Extent Y (Ey) (Протяжённость по Y) компонента **Rectangular Grid (Прямоугольная Сетка)**.	

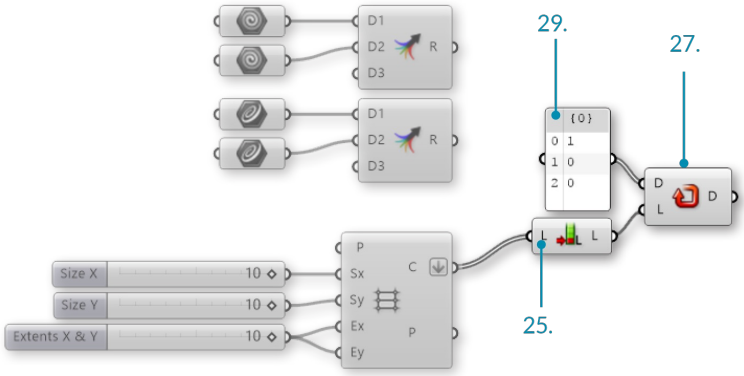


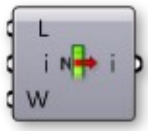
19.	**Sets/Tree/Merge (Наборы/Дерево Данных/Слияние)** – Перетащите на холст два компонента **Merge (Слияние)**.	
20.	Подсоедините первый параметр **Geometry (Геометрия)** ко входу Data Stream 1 (D1) (Поток Данных 1) первого компонента **Merge (Слияние)**.	
21.	Подсоедините второй параметр **Geometry (Геометрия)** ко входу Data Stream 2 (D2) (Поток Данных 2) первого компонента **Merge (Слияние)**.	
22.	Подсоедините первый параметр **Curve (Кривая)** ко входу Data Stream 1 (D1) (Поток Данных 1) второго компонента **Merge (Слияние)**.	
23.	Подсоедините второй параметр **Curve (Кривая)** ко входу Data Stream 2 (D2) (Поток Данных 2) второго компонента **Merge (Слияние)**.	
24.	Кликните правой кнопкой мыши по выходу Cells (C) (Ячейки) компонента **Rectangular Grid (Прямоугольная Сетка)** и выберите Flatten (Обрубить Дерево).	



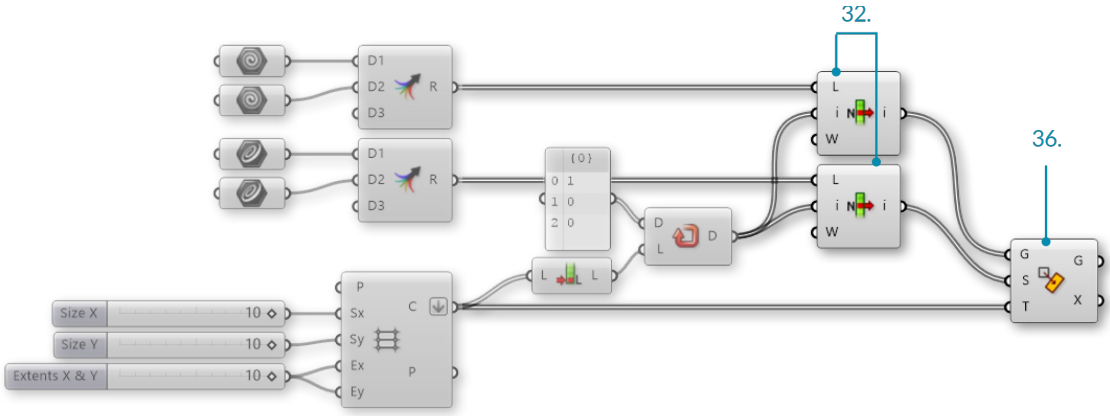
25.	**Sets/List/List Length (Наборы/Список/Длина Списка)** – Перетащите на холст компонент **List Length (Длина Списка)**.	
26.	Подсоедините выход Cells (C) (Ячейки) компонента **Rectangular Grid (Прямоугольная Сетка)** ко входу List (L) (Список) компонента **List Length (Длина Списка)**.	
27.	**Sets/Sequence/Repeat Data (Наборы/Последовательность/Повторение Данных)** – Перетащите на холст компонент **Repeat Data (Повторение Данных)**.	
28.	Подсоедините выход Length (L) (Длина) компонента **List Length (Длина Списка)** ко входу Length (L) (Длина)	

	компонента **Repeat Data (Повтор Данных)**.	
29.	**Params/Input/Panel (Параметры/Вводные/Текстовая Панель)** – Перетащите на холст **Панель**.	
30.	Дважды кликните по Панели, Развыделите Multiline data (Многострочные данные), Wrap items (Заворачивать элементы) и Special codes (Специальные коды). Введите следующее: 1 0 0 Это шаблон распределения геометрии. 0 вызывает первую ссылочную геометрию (Geometry), а 1 вызывает вторую ссылочную геометрию (Geometry). Изменение последовательности номеров будет изменять шаблон. Так же будут действовать и изменения границ сетки.	
31.	Подсоедините **Панель** ко входу Data (D) (Данные) компонента **Repeat Data (Повторение Данных)**.	

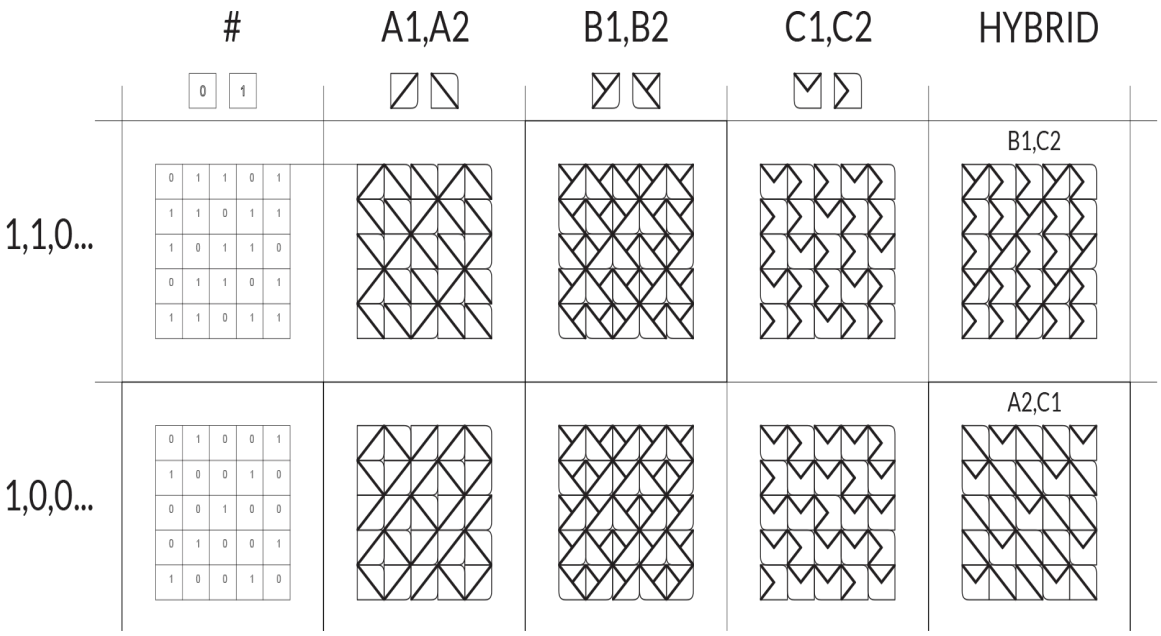


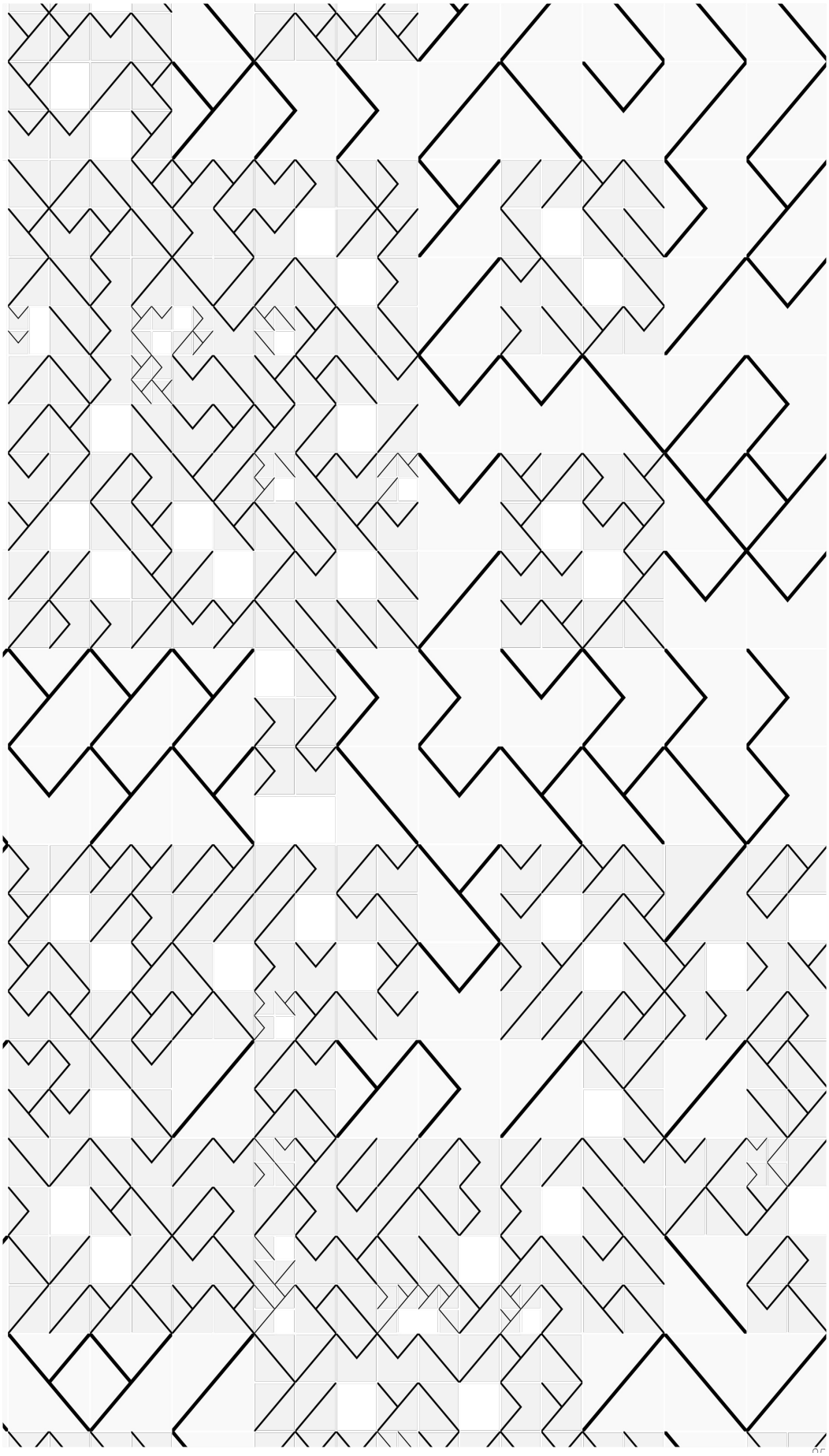
32.	**Sets/List/List Item (Наборы/Список/Элемент Списка)** – Перетащите на холст два компонента **List Item (Элемент Списка)**.	
33.	Подсоедините выход Result (R) (Результат) первого компонента **Merge (Слияние)** ко входу List (L) (Список) первого компонента **List Item (Элемент Списка)**.	
34.	Соедините выход Result (R) (Результат) второго компонента **Merge (Слияние)** ко входу List (L) (Список) второго компонента **List Item (Элемент Списка)**.	
35.	Подсоедините выход Data (D) (Данные) компонента **Repeat Data (Повторение Данных)** ко входам Index (i) (Индекс) первого и второго компонентов **List Item (Элемент Списка)**.	

36.	**Transform/Affine/Rectangle Mapping (Трансформация/Афинная/Прямоугольное Наложение)** – Перетащите на холст компонент **Rectangle Mapping (Прямоугольное Наложение)**.	
37.	Подсоедините выход Cells (C) (Ячейки) компонента **Rectangular Grid (Прямоугольная Сетка)** ко входу Target (T) (Цель) компонента **Rectangular Mapping (Прямоугольное Наложение)**.	
38.	Подсоедините выход Items (I) (Элементы) первого компонента **List Item (Элемент Списка)** ко входу Geometry (G) (Геометрия) компонента **Rectangular Mapping (Прямоугольное Наложение)**.	
39.	Подсоедините выход Connect the Items (I) (Элементы) второго компонента **Rectangular Grid (Прямоугольная Сетка)** ко входу Source (S) (Источник) компонента **Rectangular Mapping (Прямоугольное Наложение)**.	



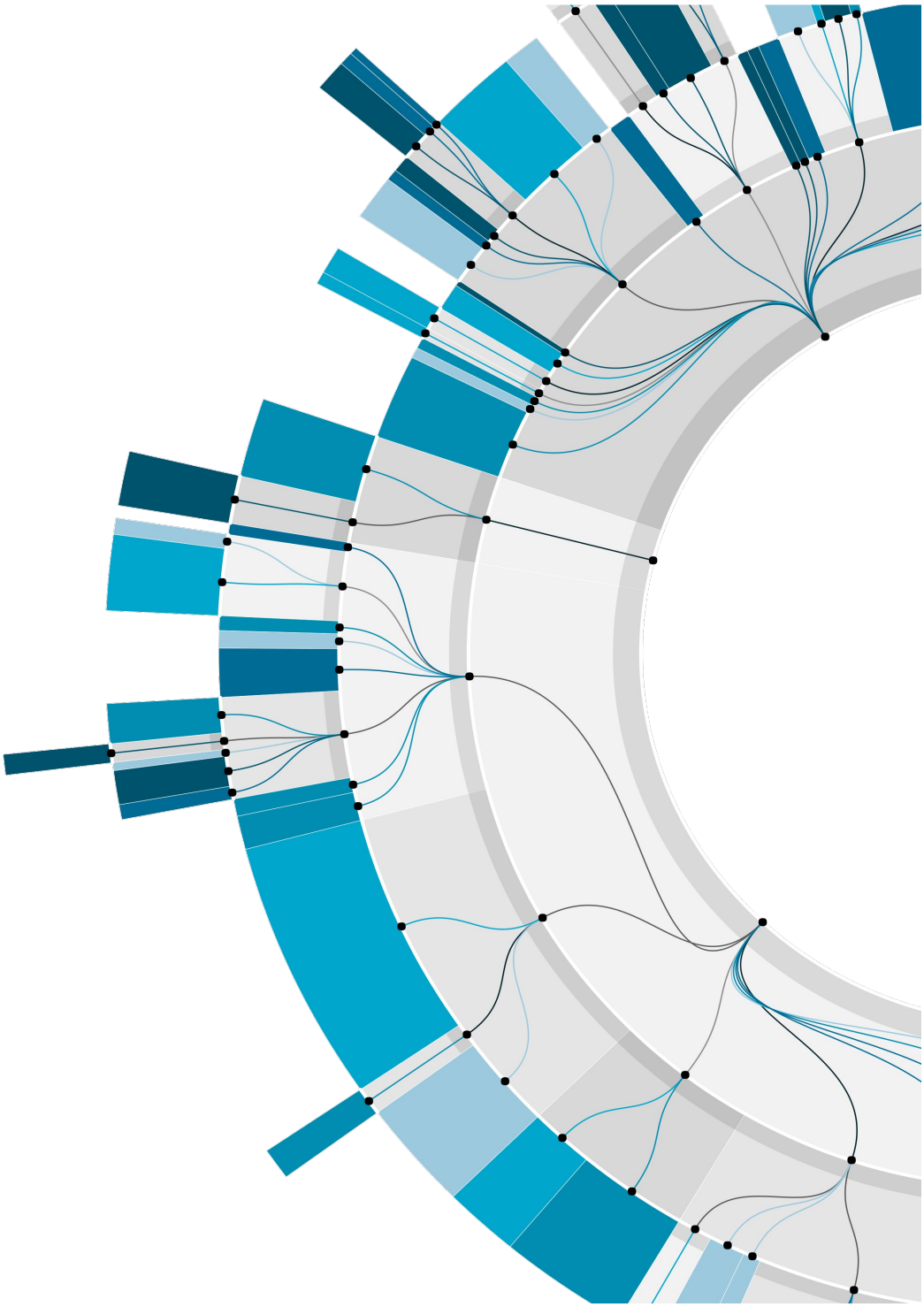
Изменение исходной геометрии и шаблона вызовет изменения и в финальном рисунке.





1.5. ПРОЕКТИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДЕРЕВЬЕВ ДАННЫХ (DATA TREES)

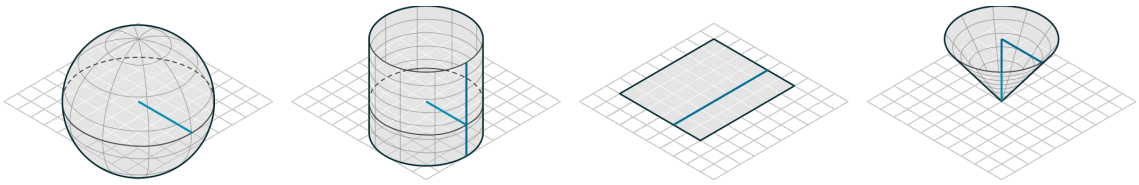
По мере усложнения ваших дефинишинов также возрастает количество данных, протекающих через них. Для того, чтобы эффективно использовать Grasshopper важно понять каким образом большие объёмы данных хранятся, как к ним организовать доступ, а также, как ими можно манипулировать.



1.5.1. Surface Geometry. (Геометрия. Поверхность.)

NURBS (non-uniform rational B-splines (неоднородные рациональные B-сплайны)) — это математические представления, которые могут точно смоделировать любую форму от простых 2D: линий, окружностей, дуг; простых 3D-форм, например, параллелепипедов, сфер и т. п., до сложных органических поверхностей произвольной формы, а также твёрдых тел. Благодаря их гибкости и точности, NURBS-модели могут быть использованы в любом процессе от иллюстраций и анимации до промышленного производства.

Помимо нескольких типов примитивов поверхности, таких как сфера, конус, плоскость и цилиндр, Rhino поддерживает три типа поверхностей произвольной формы, наиболее эффективным из которых является NURBS-поверхность. Подобно кривым, все возможные формы поверхностей можно представлять в виде NURBS-поверхности и это встроено в Rhino по-умолчанию. Это такая, безусловно, наиболее полезная форма определения поверхности и мы будем концентрироваться на ней.

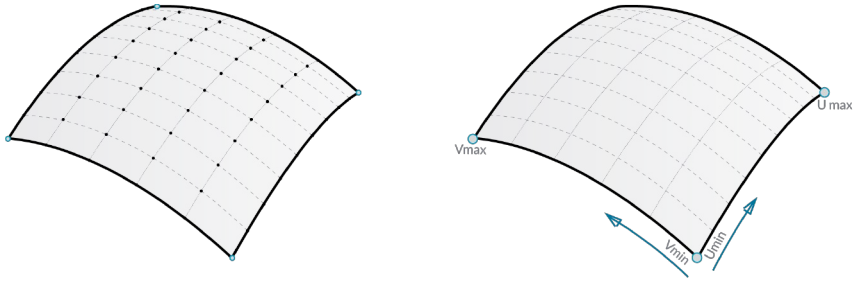


1. 2. 3. 4.

1. Примитив Sphere (Сфера) [plane (плоскость), radius (радиус)]
2. Примитив Cylinder (Цилиндр) [plane (плоскость), radius (радиус), height (высота)]
3. Примитив Plane (Плоскость) [plane (плоскость, width (ширина), height (высота)]
4. Примитив Cone (Конус) [plane (плоскость), radius (радиус), height (высота)]

1.5.1.1. NURBS-ПОВЕРХНОСТИ

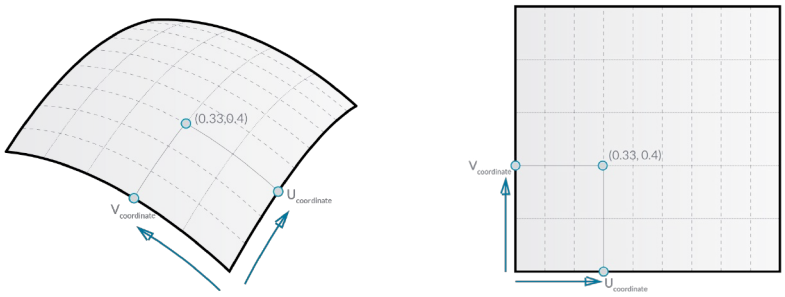
NURBS-поверхности очень схожи с NURBS-кривыми. Те же алгоритмы используются для вычисления формы, нормалей, касательных, кривизны и других свойств, но есть и некоторые явные различия. Например, кривые имеют векторы касательных и нормали плоскостей, в то время, как у поверхностей есть векторы нормалей и касательные плоскостей. Это означает, что кривые не имеют ориентации, а поверхности не имеют направления. В случае NURBS-поверхностей, на самом деле, есть два направления, обуславливающие геометрию, потому что NURBS-поверхности — это прямоугольные сетки из {U} и {V} (Горизонтальных и Вертикальных) кривых. И хотя эти направления часто произвольны, мы в конечном итоге их всё равно используем в любом случае, так как это здорово облегчает нам жизнь.



> Вы можете думать о NURBS-поверхностях, как о сетке NURBS-кривых, идущих в двух направлениях. Форма NURBS-поверхности определяется количеством контрольных точек и величиной степени поверхности в U и V направлениях. NURBS-поверхности эффективны для хранения и представления поверхностей произвольной формы с высокой степенью точности.

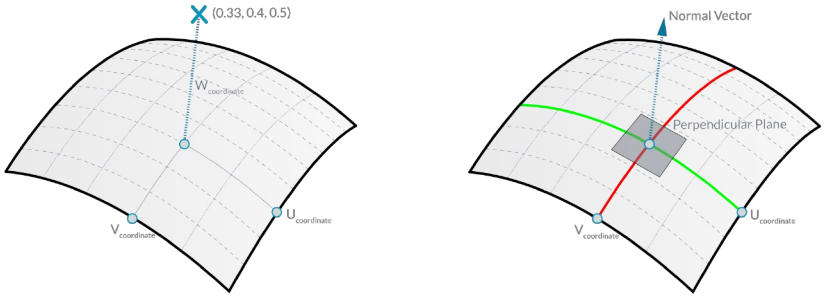
Surface Domain (Домен Поверхности). Домен поверхности определяется как диапазон (U,V) параметров, оцениваемых в 3D-точке на поверхности. Домен в каждом измерении (U или V), как правило, описывается как два действительных числа (u_min to u_max) и (v_min to v_max). Изменение домена поверхности называется репараметризация поверхности.

В Grasshopper часто бывает полезно репараметрировать NURBS-поверхность так, чтобы диапазон обоих доменов: и U и V стал от 0 до 1. Это облегчает оценку и работу с этими поверхностями.



> Оценка параметров на равных интервалах в 2D-параметре необязательно ведёт к равенству этих интервалов в 3D-пространстве.

Surface evaluation (Оценка Поверхности). Есть особенности оценки поверхности в параметре, являющегося результирующим от доменов поверхности в точке, находящейся на этой поверхности. Имейте ввиду, что середина домена (mid-u, mid-v) необязательно будет оценена, как точка середины 3D-поверхности. Кроме того, оценка U- и V-значений, которые находятся за пределами домена поверхности не дадут полезных результатов.

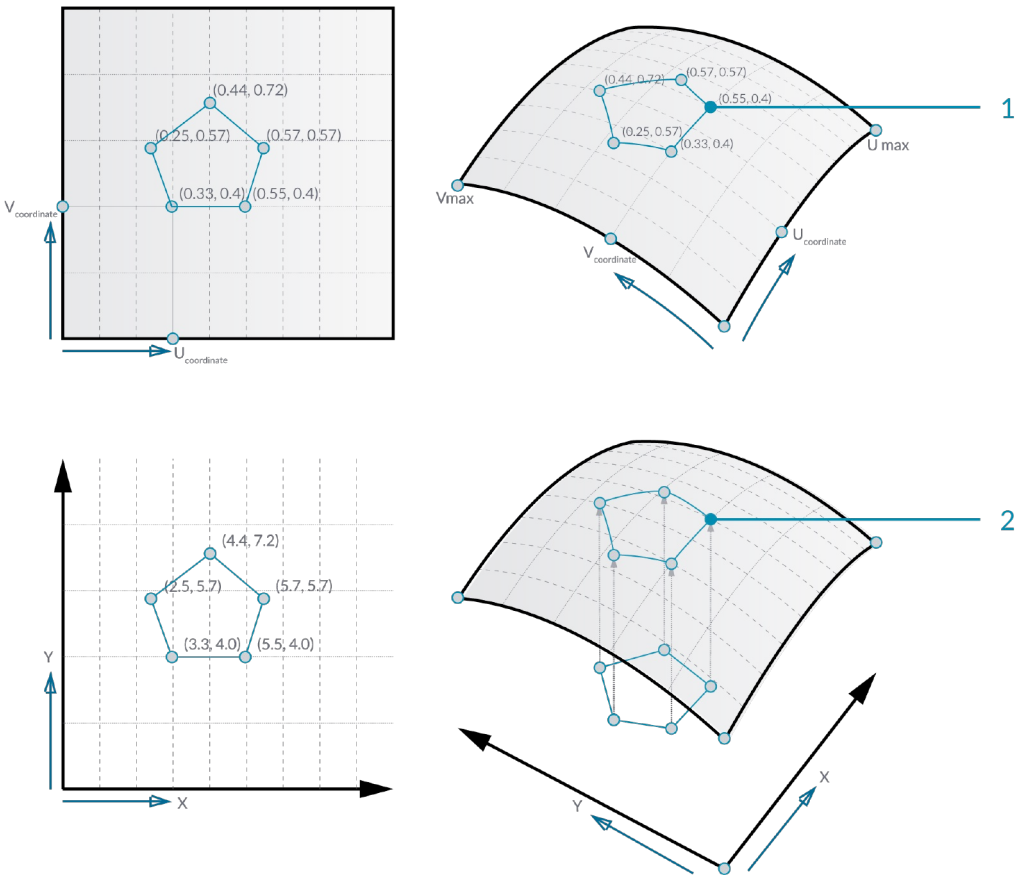


Normal Vectors (Векторы Нормалей) и Tangent Planes (Касательные Плоскости). Касательная плоскость к поверхности в данной точке представляет собой плоскость, которая касается поверхности в данной точке. Z-направление касательной плоскости соответствует нормали поверхности в этой точке.

Grasshopper обрабатывает NURBS-поверхности аналогично тому, как это делает Rhino, потому что он построен на ядре операций, которые необходимы для создания поверхностей. Однако, из-за того, что Grasshopper отображает поверхность во вьюпортах Rhino раздельно от Rhino-геометрии (что, собственно, является причиной невозможности выделения Grasshopper-геометрии во вьюпортах Rhino до её запекания), некоторые параметры полигональной сетки несколько снижены, чтобы поддерживать скорость отображения результатов Grasshopper довольно высокой. Вы можете заметить несколько грубоватые грани на визуализационных полигональных сетках ваших поверхностей, но это является лишь результатом настройки отрисовки. Любая запечённая геометрия будет использовать более высокие параметры полигональной сетки.

1.5.1.2. ПРОЕЦИРОВАНИЕ НА ПОВЕРХНОСТЬ (PROJECTING SURFACES)

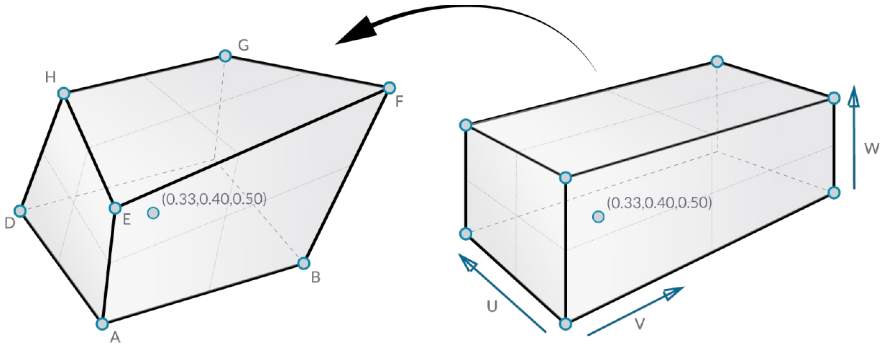
В предыдущем разделе мы объяснили, что NURBS-поверхность содержит собственное координатное пространство, определяемое по U и V доменам. Это означает, что двумерная геометрия, определяемая по X и Y координатам может быть наложена на UV-пространство поверхности. Геометрия эта будет растягиваться и изменяться в зависимости от кривизны поверхности. Это отличается от простого проецирования 2D-геометрии на поверхность, где векторы отрисовываются от 2D-геометрии в указанном направлении до их пересечения с поверхностью.



Вы можете думать о проекции как об отбрасывании геометрией тени на поверхность. Накладываемая геометрия также будет растягиваться по поверхности.

1. Накладываемая геометрия определяется по uv-координатам
2. Проецирование геометрии на поверхность

Также, как 2D-геометрия может проецироваться на UV-пространство поверхности, 3D-геометрия, содержащаяся в параллелограмме габаритного контейнера может быть наложена на соответствующую часть поверхности в виде содержания уже искажённого соответствующим образом контейнера. Эта операция называется Box Morphing (Трансформация Контейнером) и полезна для заполнения изогнутыми поверхностями при помощи компонентов, генерирующих 3D-геометрию.

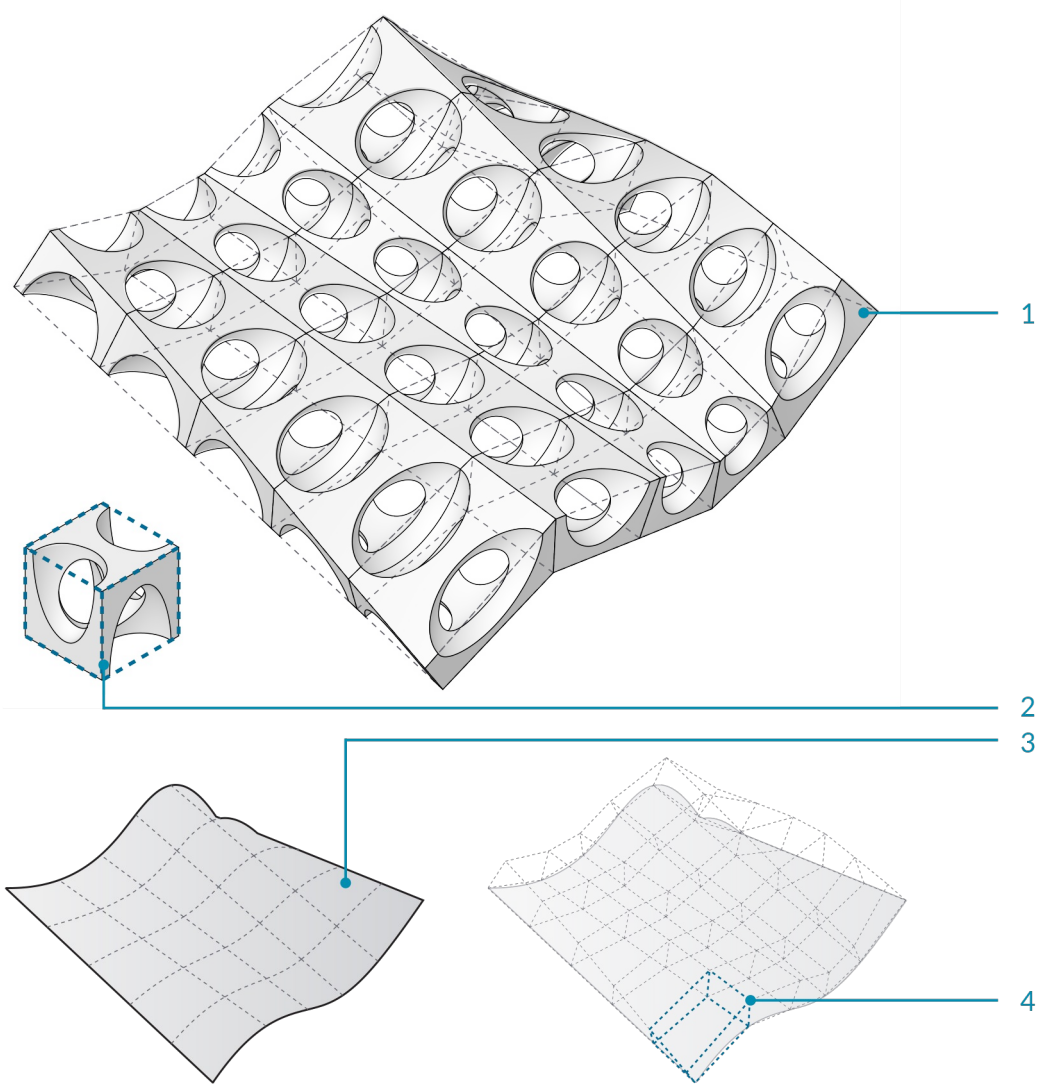


Чтобы создать массив трансформированных контейнеров (array twisted boxes) на поверхности, домен поверхности должен быть разделён для создания сетки из лоскутов поверхности. Эти трансформированные контейнеры будут по нормалям векторов , отрисованных в углах каждого лоскута, соблюдая необходимую высоту и создавая контейнеры, определяемые по конечным точкам этих векторов и угловых точек лоскутов поверхностей.

1.5.1.3. ДЕФИНИШИН ДЛЯ МОРФИНГА (MORPHING DEFINITION)

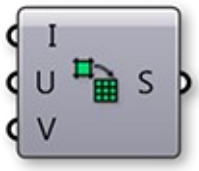
Загрузить файл примера, относящегося к данному разделу.

В этом примере мы будем использовать компонент для морфинга с помощью контейнера с целью заполнить NURBS-поверхность геометрическим компонентом.

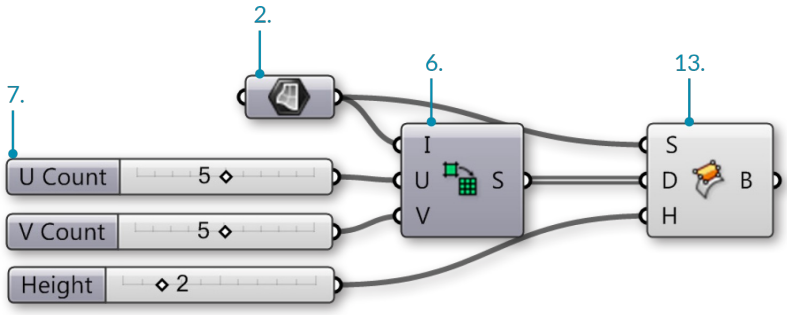


1. NURBS-поверхность, заполненная геометрическим компонентом.
2. Исходный компонент в ссылочном контейнере.
3. Поверхность, разделённая на лоскуты.
4. Искривлённые контейнеры, собранные в массив на поверхности.

1.	Начните новый дефинишин, введя Ctrl+N (в Grasshopper)	
2.	**Params/Geometry/Surface (Параметры/Геометрия/Поверхность)** – Перетащите на холст параметр **Surface (Поверхность)**.	

	Это поверхность, которая будет заполнена геометрическими компонентами.	
3.	**Params/Geometry/ Geometry (Параметры/Геометрия/Геометрия)** – Перетащите на холст параметр **Geometry (Геометрия)**. Это компонент, который будет собран в массив поверх поверхности.	
4.	Кликните правой кнопкой мыши по Параметру **Surface (Поверхность)** и выберите “Set One Surface” (Выбрать Одну Поверхность) – для того, чтобы задать вашу ссылочную Rhino-геометрию из вьюпорта Rhino.	
5.	Кликните правой кнопкой мыши по Параметру **Geometry (Геометрия)** и выберите “Set One Geometry”, чтобы задать вашу ссылочную Rhino-геометрию.	
6.	**Maths/Domain/Divide Domain2 (Математика/Домен/Разделить Двумерный Домен)** – Перетащите на холст компонент **Divide Domain2 (Разделить Двумерный Домен)**	
7.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст три **Числовых Слайдера**	
8.	Дважды кликните по первому **Числовому Слайдеру** и задайте следующее: Rounding (Округление): Integer (Целые Числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 10 Value (Значение): 5	
9.	Задайте те же самые значения для второго и третьего **слайдеров**	
10.	Подсоедините выход параметра **Surface (Поверхность)** ко входу Domain (I) компонента **Divide Domain2 (Разделить Двумерный Домен)**	
11.	Подсоедините первый **Числовой Слайдер** ко входу U Count (U) (Число Сегментов в Горизонтальном направлении) компонента **Divide Domain2 (Разделить Двумерный Домен)**	
12.	Подсоедините второй **Числовой Слайдер** ко входу V Count (V) (Число Сегментов в Вертикальном направлении) компонента **Divide Domain2 (Разделить Двумерный Домен)**	

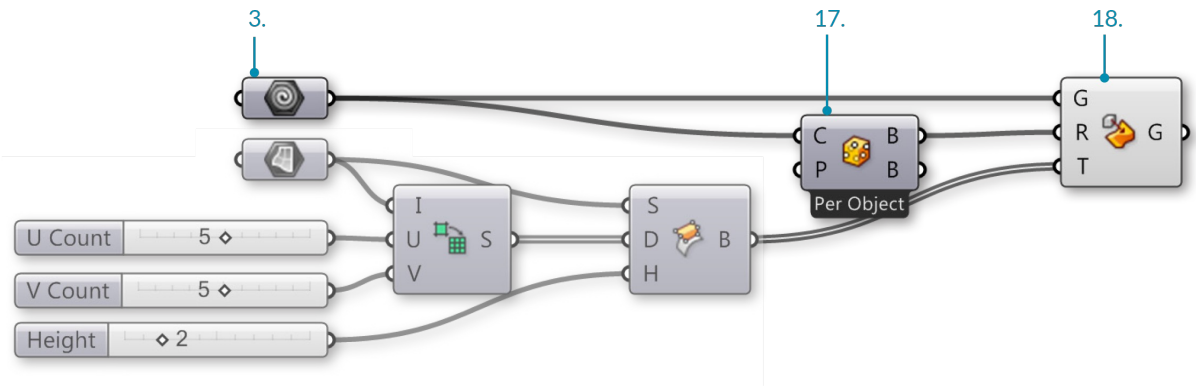
13.	**Transform/Morph/Surface Box (Трансформация/Морфинг/Габаритным Контейнером Поверхности)** – перетащите на холст компонент **Surface Box (Контейнер на Поверхности)**	
14.	Подсоедините выход параметра **Surface (Поверхность)** ко входу Surface (S) (Поверхность) компонента **Surface Box (Контейнер на Поверхности)**	
15.	Подсоедините выход Segements (S) (Сегменты) компонента **Divide Domain2 (Разделить Двумерный Домен)** ко входу Domain (D) компонента **Surface Box (Контейнер на Поверхности)**	



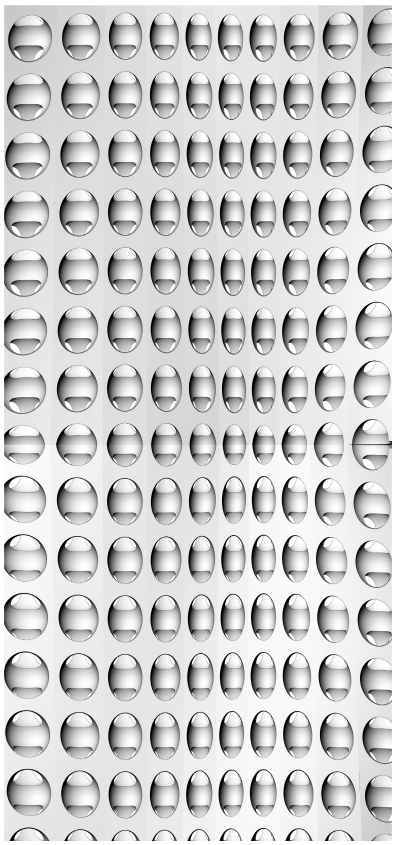
Вы должны увидеть сетку, подготовленную для искривления по ней контейнера, размноженного по вашей ссылочной поверхности. Измените значение Count (Счёт) на слайдерах U (Горизонталь) и V (Вертикаль), чтобы изменить количество контейнеров, а также подвигайте рычаг слайдера высоты для коррекции контейнеров по высоте.

16.	Подсоедините третий **Числовой Слайдер** ко входу Height (H) (Высота) компонента **Surface Box (Контейнер на Поверхности)**	
17.	**Surface/Primitive/Bounding Box (Поверхнсть/Примитивы/Габаритный Контейнер)** – Перетащите на холст компонент **Bounding Box (Габаритный Контейнер)**	
18.	Transform/Morph/Box Morph (Трансформаця/Морфинг/Трансформация Контейнером)– Перетащите не холст компонент Box Morph (Трансформация Контейнером)	
19.	Подсоедините выход параметра **Geometry (Геометрия)** ко входу Content (C) (Содержимое) компонента **Bounding Box (Габаритный Контейнер)**	

20.	Подсоедините выход параметра **Geometry (Геометрия)** ко входу Geometry (G) (Геометрия) компонента **Box Morph (Трансформация Контейнером)**	
21.	Подсоедините выход Box (B) (Параллелепипед) компонента **Bounding Box (Габаритный Контейнер)** ко входу Reference (R) (Источника) компонента **Box Morph (Трансформация Контейнером)**	
22.	Подсоедините выход Twisted Box (B) (Искривлённый Параллелепипед) компонента Surface Box (Контейнер на Поверхности) ко входу Target (T) (Цель) компонента Box Morph (Трансформация Контейнером)	



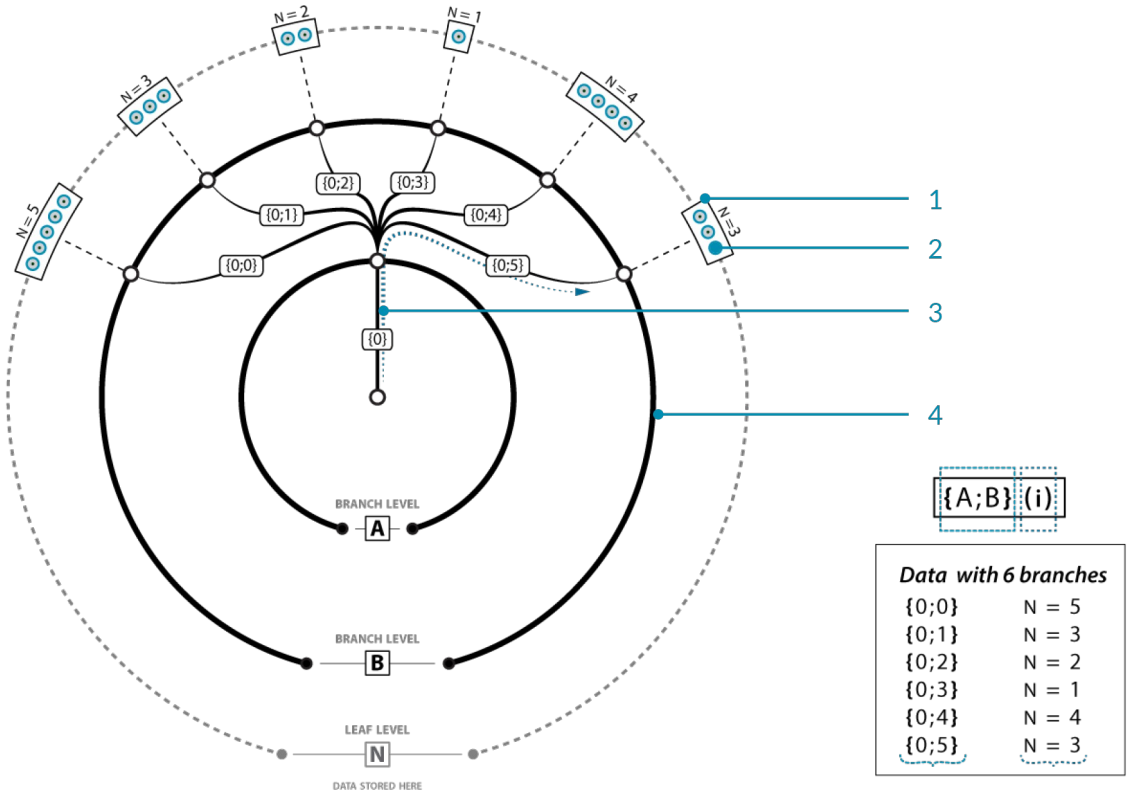
Теперь Вы должны увидеть вашу геометрию, размноженную по поверхности.



1.5.2. Что такое Дерево Данных (Data Tree)?

Деревья Данных (Data Tree) — это иерархическая структура для хранения данных во вложенных списках. Деревья данных создаются тогда, когда Grasshopper -компоненты структурированы на то, чтобы взять набор данных и вывести множество наборов данных. Grasshopper обрабатывает эти новые данные путём их вложения в виде подчинённый списков (sub-lists). Эти вложенные подчинённые списки работают также, как структура папок на вашем компьютере в том, что доступ к индексированным элементам требует перемещения по путям, сообщающим о том, от каких родительских списков они сгенерированы и об их собственном сую-индексе (sub-index).

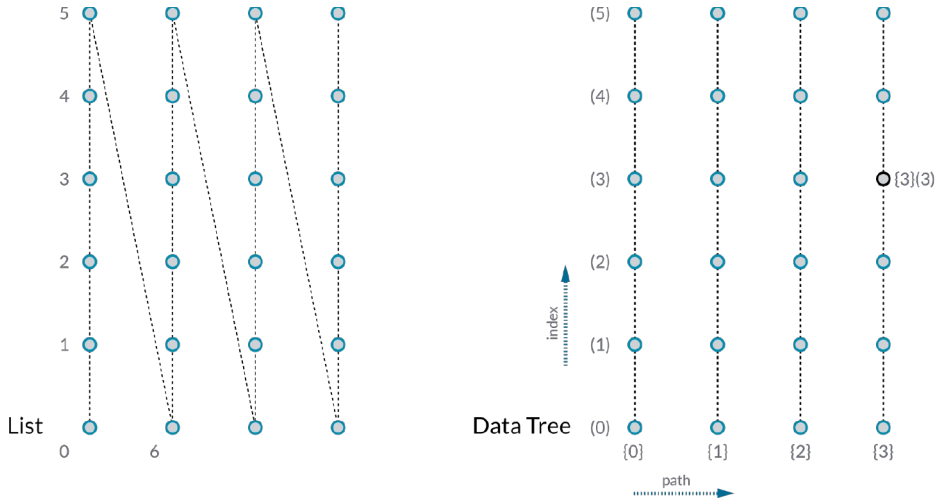
Это позволяет иметь несколько списков данных внутри одного параметра. Поскольку списков несколько — должен быть способ идентификации каждого индивидуального списка. Дерево данных представляет собой список списков, или, иногда даже список списка списков и так далее.



На изображении выше есть одна главная ветвь (Вы могли бы назвать её «ствол», но так как возможно иметь несколько главных ветвей, то, пожалуй, это может быть некорректным названием) на пути {0}. Эта ветвь не содержит никаких данных, но имеет 6 подчинённых ветвей. Каждая из этих подчинённых ветвей наследует индекс {0} от родительской и добавляет к нему собственный суб-индекс (0, 1, 2, 3, 4 и 5 соответственно). Было бы неправильно назвать это “индекс” (index), так как такое название подразумевает только одно число. Наверное, лучше обращаться к нему, как «путь» (path), так как это напоминает структуру папки на диске. На каждой из этих

суб-ветвей мы находим некоторые данные. Каждый элемент данных является таким образом частью одной (и только одной) ветви дерева и каждый элемент имеет индекс, указывающий его местонахождение в пределах ветви. А каждая ветвь содержит путь, определяющий её местоположение на дереве.

Изображение ниже демонстрирует разницу между списком и деревом данных. Слева массив из четырёх столбцов и шести точек на каждом. И всё это содержится в одном списке. Те, что в первом столбце нумерованы 0-5, во втором 6-11 и так далее. Справа тот же самый массив точек, но на сей раз данные содержатся в виде дерева. Данные в виде дерева — это список из четырёх столбцов, а каждый столбец представляет собой список из шести точек. Индекс каждой точки представлен в формате: номер столбца, номер строки. Это гораздо более полезный способ организации данных, поскольку с его помощью Вы можете легко получить доступ и оперировать как всеми точками сразу, так и, например, удалить каждый второй ряд точек, подключить альтернативные точки и т. д.

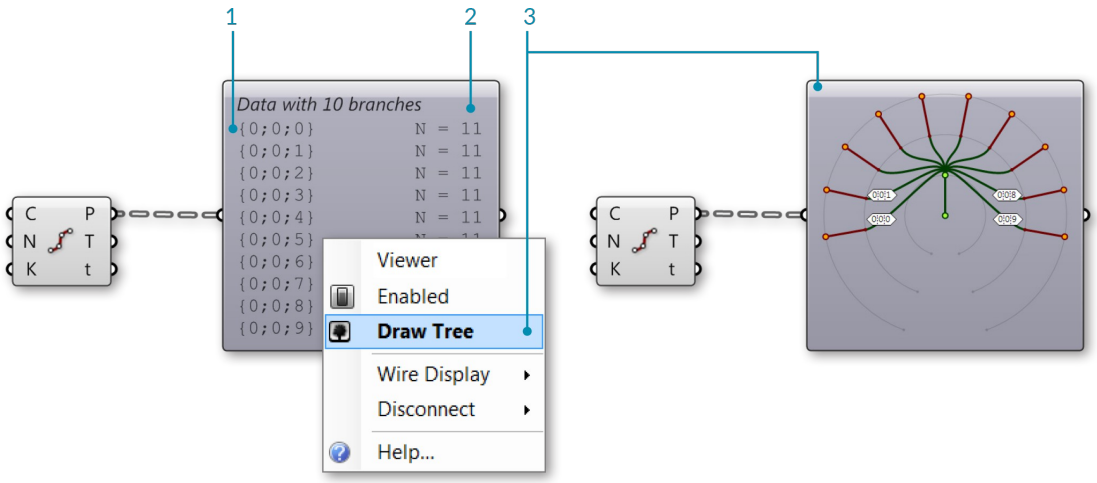


1.5.2.1. ВИЗУАЛИЗАЦИЯ ДЕРЕВЬЕВ ДАННЫХ

> Загрузить файл примера, относящегося к данному разделу.

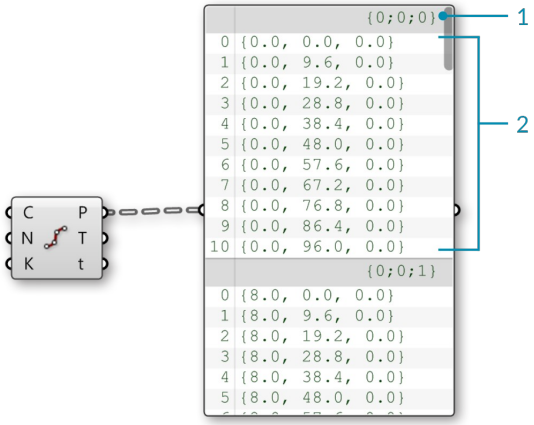
Из-за их сложности, бывает трудно разобраться с деревом данных. Grasshopper имеет ряд инструментов, помогающих визуализировать и понять структуру данных, организованных в деревья.

Param Viewer (Просмотрщик Параметра) (Params/Util/Param Viewer (Параметры/Утилиты/Просмотрщик Параметра)) позволяет визуализировать структуру данных в текстовом виде, либо в виде дерева. Подсоедините любой выход, содержащий данные ко входу Просмотрщика Параметров. Для того, чтобы структура данных отображалась в виде дерева, кликните правой кнопкой мыши на Промотрщике Параметра и выберите «Draw Tree» (Отрисовывать Деревом). В данном примере Просмотрщик Параметра подключен к выходу Points (P) (Точки) компонента Divide Curve (Разделить Кривую), который делит 10 кривых на 10 сегментов каждую. Десять ветвей соответствует десяти кривым, каждая ветвь содержит список из 11 точек, которые являются результатом деления кривых на 10 частей.



1. Путь (Path) каждого списка (list)
2. Количество элементов (items) в каждом списке
3. Выберите "Draw Tree" (Отрисовывать Деревом), чтобы увидеть дерево данных (data tree).

Если мы подключим Panel (Текстовую Панель) к тому же выходу, она отобразит 10 списков с одиннадцатью элементами в каждом. Вы можете видеть, что каждый элемент — это точка, определяемая по трём координатам. Путь отображается в верхней части каждого списка и соответствует путям, перечисленным в Param Viewer (Просмотрщике Параметра).

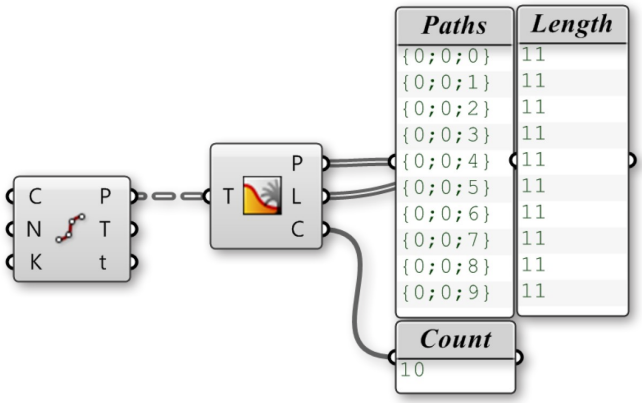


1. Путь (Path).
2. Список (List) из 11 элементов (items).

Tree Statistics (Статистика Дерева Данных) Компонент Tree Statistics (Статистика Дерева Данных) (Sets/Tree/Tree Statistics (Наборы/Дерево Данных/Статистика Дерева Данных)) возвращает некоторую статистику, относящуюся к Дереву Данных, включая:

- P — Все пути (paths) дерева данных
- L — Длину (length) каждой ветви (branch) дерева данных
- C — Число (count) путей и ветвей в дереве данных

Если присоединить выход Points (Точки) того же самого компонента Divide Curve (Разделить Кривую), Вы можете увидеть пути, длины и количества в Текстовой Панели. Этот компонент полезен, поскольку разделяет статистику на три выхода, позволяя просматривать только то, что Вам необходимо.



Оба компонента: и Param Viewer (Просмотрщик Параметра), и Tree Statistics (Статистика Дерева Данных) 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 Деревьев Данных (Data Trees)

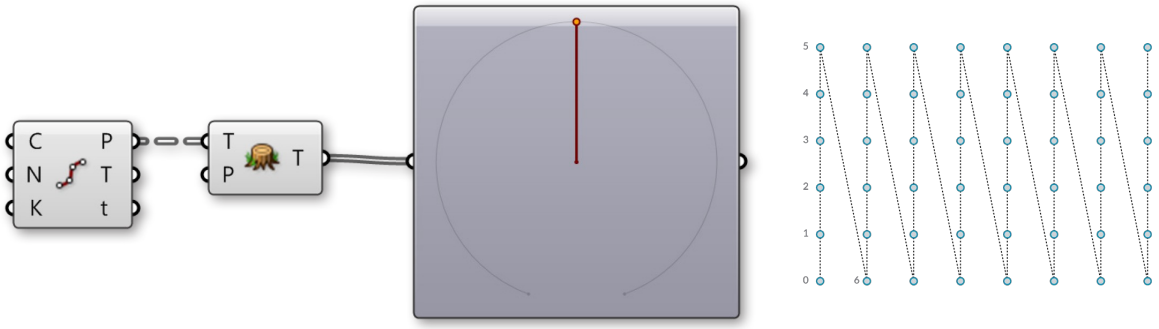
> **Загрузить** файл примера, относящегося к данному разделу.

Grasshopper содержит инструменты, изменяющие структуру дерева данных. Эти инструменты могут помочь Вам получить доступ к конкретным данным в дереве, изменить способ, которым они сохранены, упорядочены и идентифицированы.

Давайте взглянем на некоторые манипуляции над деревом данных и визуализируем эффект от их воздействия на дерево.

1.5.3.1. FLATTEN (ОБРУБИТЬ ДЕРЕВО)

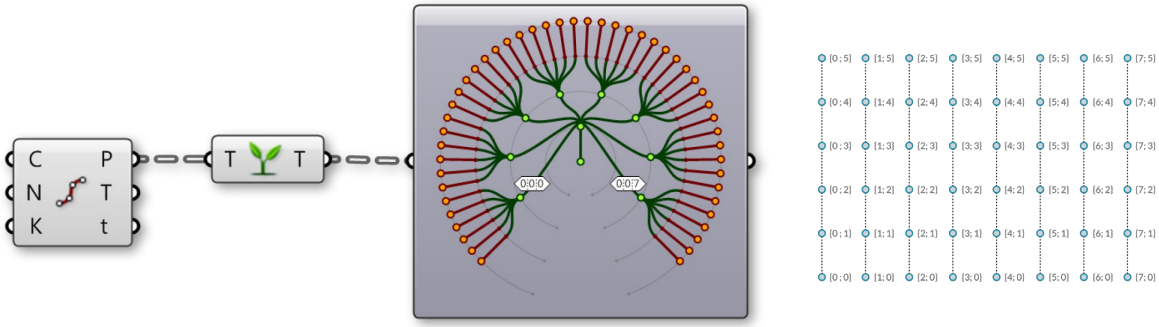
Обрубка (Flattening) убирает разветвлённость списков, удаляя все уровни Дерева Данных. В результате мы получаем единственный Список (List). Используя компонент Flatten (Обрубить Дерево) (Sets/Tree/Flatten (Наборы/Дерево Данных/Обрубить)) на пути от выхода «P» нашего компонента Divide Curve (Разделить Кривую) мы сможем увидеть в Param Viewer (Просмотрщике Параметра) новую структуру данных.



> В Просмотрщике Параметра (Param Viewer) Вы можете увидеть, что теперь мы имеем лишь одну ветвь, содержащую список из 48 точек.

1.5.3.2. GRAFT TREE (ПРИВИТЬ ДЕРЕВО ДАННЫХ)

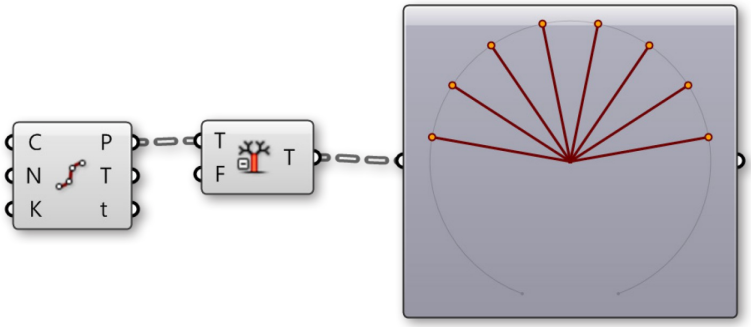
Прививание (Grafting) создаёт новую Ветвь (Branch) для каждого Элемента Данных (Data Item). Если мы запустим данные через компонент Graft Tree (Привить Дерево Данных) (Sets/Tree/Graft Tree (Наборы/Дерево Данных/Привить)), каждая точка деления теперь будет иметь свою индивидуальную ветвь, а не делить общую ветвь с другими точками деления одной кривой.



> В Просмотрщике Параметра (Param Viewer) мы можем увидеть, что раньше данные имели следующую структуру: 8 ветвей, на каждой по 6 элементов. Теперь мы имеем 8 ветвей, на каждой из которых 6 подчинённых ветвей, содержащих каждая по 1 элементу.

1.5.3.3. SIMPLIFY TREE (УПРОСТИТЬ ДЕРЕВО ДАННЫХ)

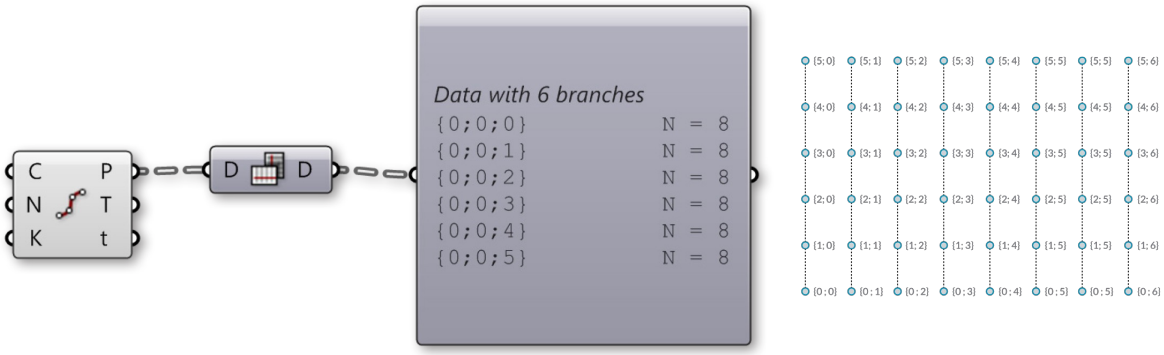
Упрощение (Simplify) убирает из Дерева Данных (Data Tree) информацию о перекрывающих ветвях (overlapping Branches). Если мы упростим дерево через компонент Simplify Tree (Упростить Дерево Данных) (Sets/Tree/Simplify Tree (Наборы/Дерево Данных/Упростить)), то информация о первой ветви, не содержащей данных, будет удалена.



> В Просмотрщике Параметра (Param Viewer) мы видим, что у нас по-прежнему 8 ветвей с 6-ю элементами на каждой, но первая ветвь была удалена.

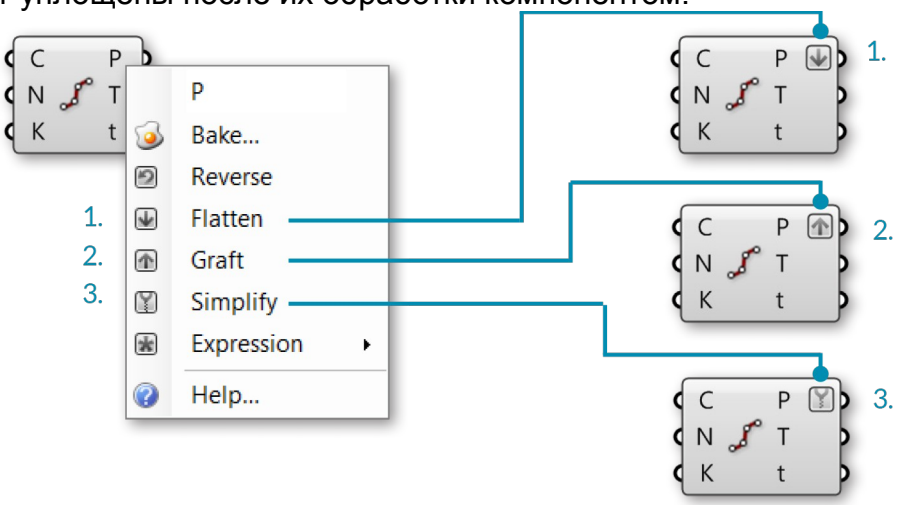
1.5.3.4. FLIP MATRIX (ПЕРЕВЕРНУТЬ МАТРИЦУ)

Компонент Flip Matrix (Перевернуть Матрицу) (Sets/Tree/Flip Matrix (Наборы/Дерево Данных/Перевернуть Матрицу)) меняет местами «Столбцы» и «Строки» Деревьев Данных, имеющих два Пути Индексов (Path Indices).



> В нашем Просмотрщике Параметра (Param Viewer) мы можем видеть, что структура данных в 8 ветвей с 6-ю элементами на каждой сменилась на другую: 6 ветвей с 8-ю элементами на каждой.

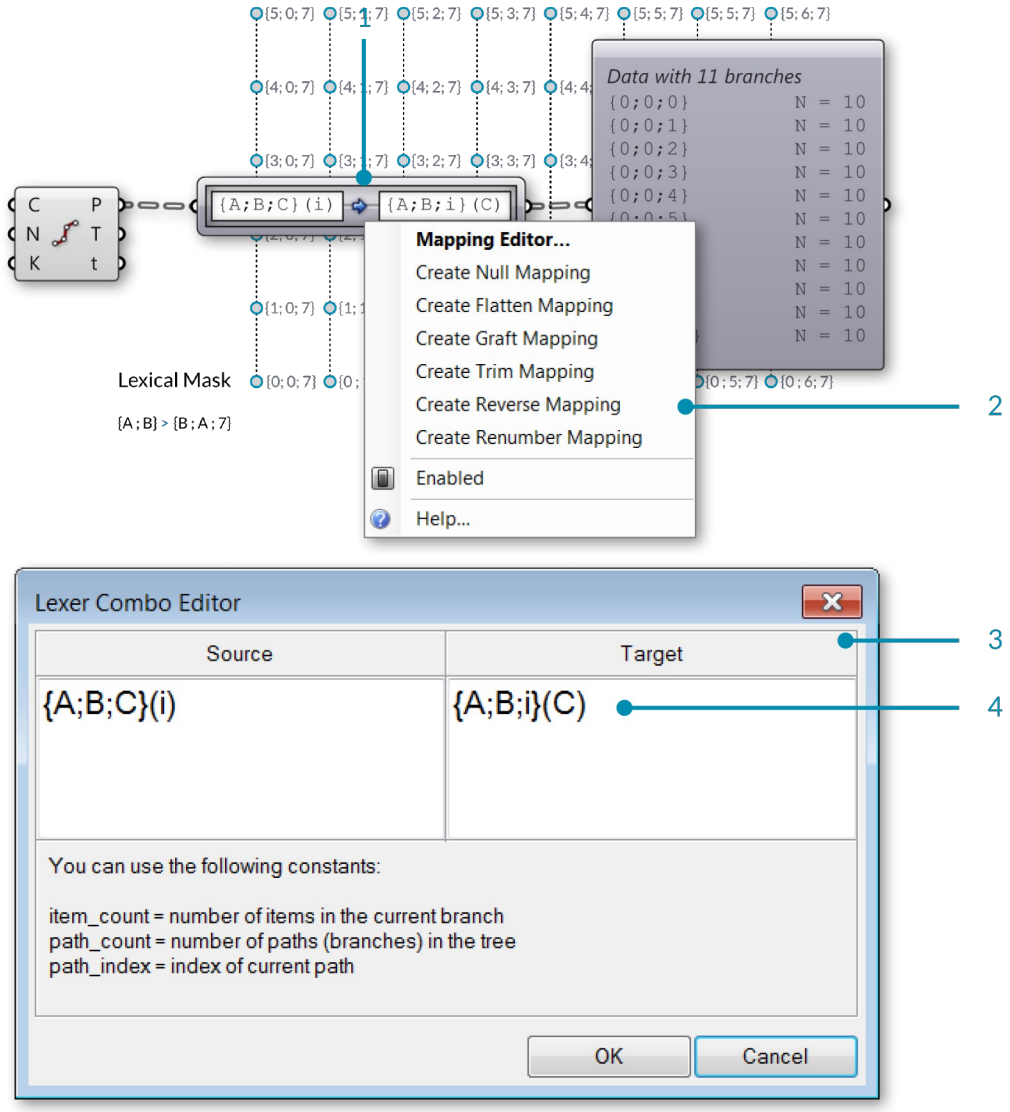
Операции Flatten (Обрубить), Graft (Привить) и Simplify (Упростить) могут быть применены непосредственно на входе или выходе компонентов, не задействуя отдельные специализированные компоненты. Просто кликните по нужному входу или выходу правой кнопкой мыши и выберите Flatten, Graft или Simplify из контекстного меню. После этого компонент будет отображать значок, указывающий, что Дерево Данных модифицировано. Не забывайте про течение остальной части программного потока Grasshopper. Если Вы обрубите Дерево Данных на входе компонента, то и далее будут течь уплощённые (flatten) данные и именно в таком виде они будут обработаны компонентом. Если Вы примените flatten на выходе компонента, то данные будут уплощены после их обработки компонентом.



1. Данные Уплощены (Flattened) на выходе P
2. Данные Привиты (Grafted) на выходе P
3. Упрощение данных (Simplified) на выходе P

1.5.3.5. PATH MAPPER (СОПОСТАВИТЕЛЬ ПУТЕЙ)

Компонент Path Mapper (Сопоставитель Путей) (Sets/Tree/Path Mapper (Наборы/Дерево Данных/Сопоставитель Путей)) позволяет выполнить лексические операции над деревьями данных. Лексические операции — это операции логического сопоставления между путями данных (data paths) и индексами (indices), определяемые текстовыми (лексуальными (lexical)) масками и шаблонами (паттернами (patterns)).

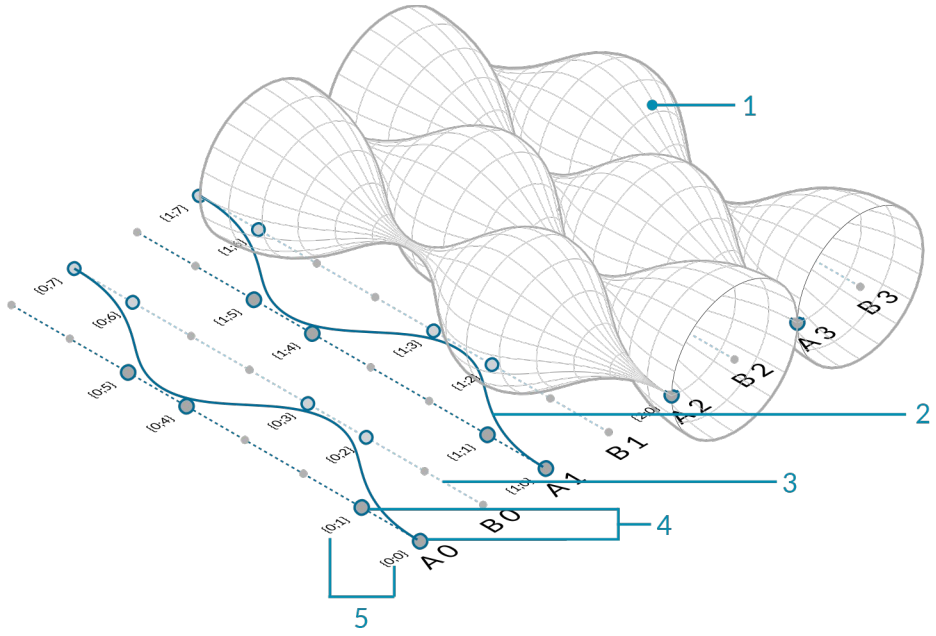


1. Компонент Path Mapper (Сопоставитель Путей)
2. Кликните правой кнопкой мыши по компоненту Path Mapper (Сопоставитель Путей) и выберите предопределённую опцию из контекстного меню или откройте Mapping Editor (Редактор Наложения)
3. Mapping Editor (Редактор Наложения)
4. Вы можете модифицировать дерево данных, переналожив пути индексов и нужных ветвей

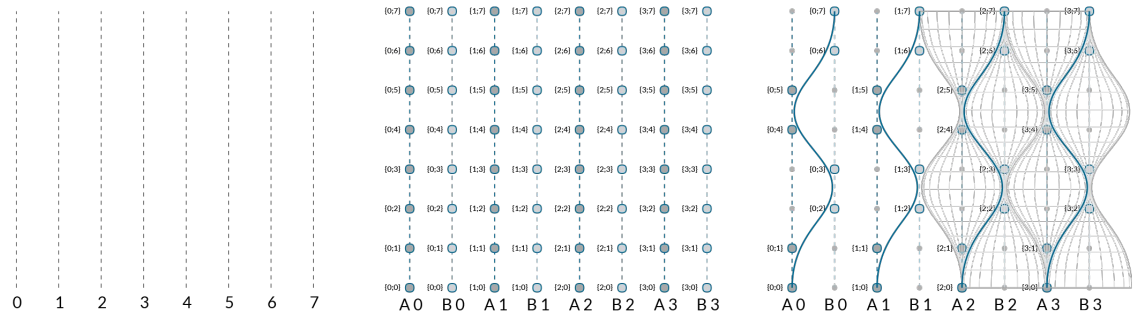
1.5.3.6. ДЕФИНИШИН ПЕРЕПЛЕТЕНИЯ (WEAVING)

Загрузить файл примера, относящегося к данному разделу.

В этом примере мы будем манипулировать списками (lists) и деревьями данных (data trees) для того, чтобы сплести (weave) воедино списки точек (points), определить паттерн (pattern) и содать геометрию поверхности (surface geometry).

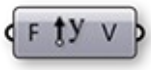


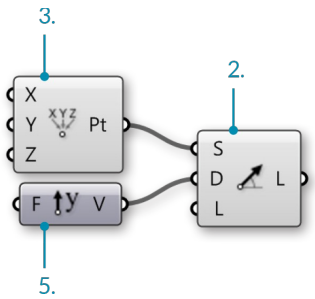
1. NURBS-поверхность вращения
2. NURBS-кривая
3. Массив кривых
4. Точки деления
5. Пути (индексы) точек



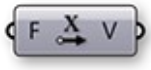
1. 2. 3.

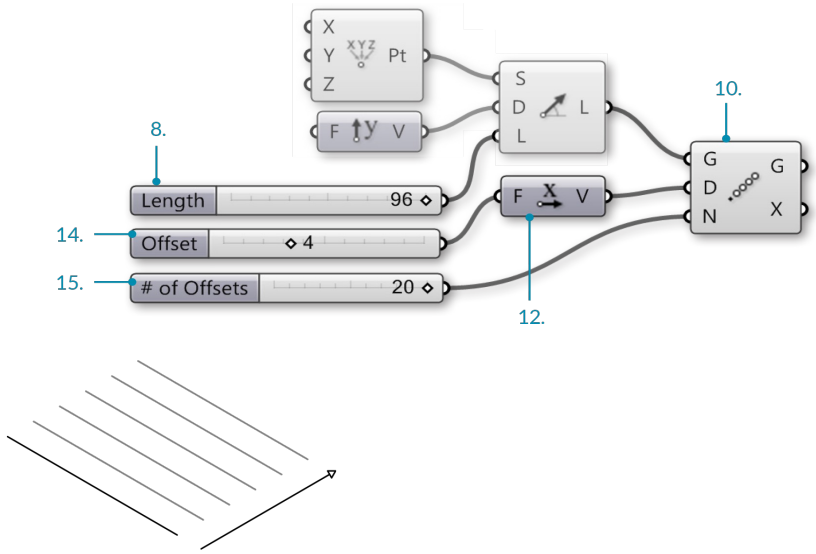
1. Массив кривых
2. Перенаправление кривых в списки A и B, разбиение кривых
3. Отбрасывание точек, сплетение и вращение

1.	Начните новый дефинишин, введя Ctrl+N (в Grasshopper)	
2.	**Curve/Primitive/Line SDL (Кривая/Примитивы/Прямая SDL (Начало, Направление, Длина))** – Перетащите на холст компонент **Line SDL (Прямая SDL (Начало, Направление, Длина))**	
3.	**Vector/Point/Construct Point (Вектор/Точка/Сконструировать Точку)** – Перетащите на холст компонент **Construct Point (Сконструировать Точку)**	
4.	Подсоедините выход Point (Pt) (Точка) компонента **Construct Point (Сконструировать Точку)** ко входу Start (S) (Начало) компонента **Line SDL (Прямая SDL (Начало, Направление, Длина))**	
5.	**Vector/Vector/Unit Y (Вектор/Вектор/Унифицированный по Y)** – Перетащите на холст компонент **Unit Y (Унифицированный по Y)**. По-умолчанию коэффициен (factor) компонентов Unit Vector (Унифицированный Вектор) равен 1.0	
6.	Подсоедините компонент **Unit Y (Унифицированный по Y)** ко входу компонента **Line SDL (Прямая SDL (Начало, Направление, Длина))**	

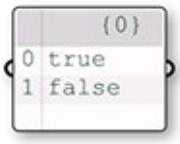


7.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст компонент **Числовой Слайдер**	
8.	Дважды кликните на **Числовом Слайдере** и задайте следущее: Name (Имя): Length (Длина) Rounding (Округление): Integer (Целые числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 96 Value (Значение): 96	
9.	Подсоедините **Числовой Слайдер** ко входу Length (L) (Длина) компонента **Line SDL (Прямая SDL (Начало, Направление, Длина))**	

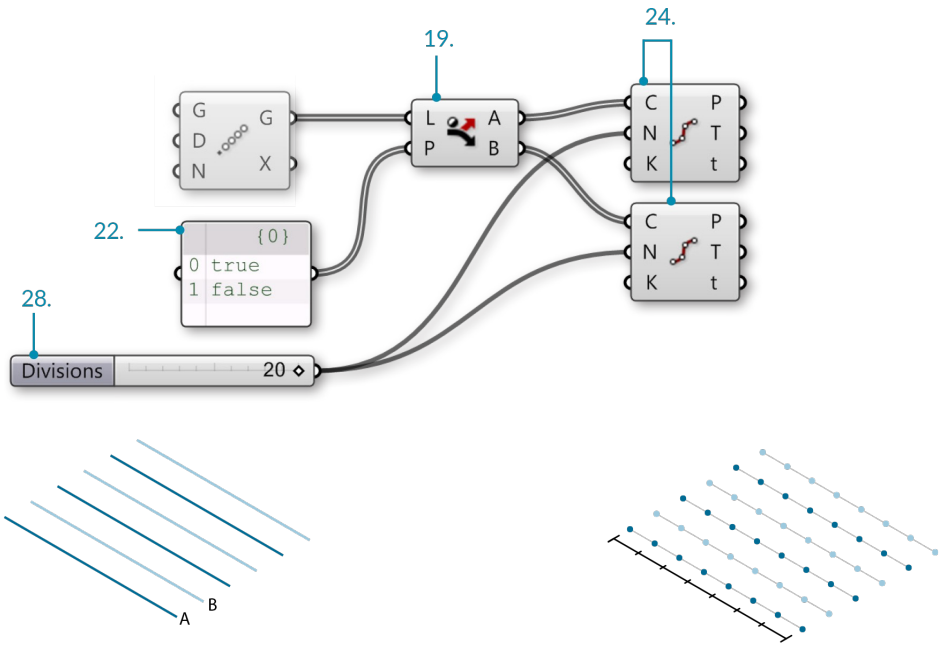
10.	**Transform/Array/Linear Array (Трансформация/Массив/Линейный Массив)** – Перетащите на холст компонент **Linear Array (Линейный Массив)**	
11.	Подсоедините выход Line (L) (Прямая) компонента **Line SDL (Прямая SDL (Начало, Направление, Длина))** ко входу Geometry (G) (Геометрия) компонента **Linear Array (Линейный Массив)**	
12.	**Vector/Vector/Unit X (Вектор/Вектор/Унифицированный по X)** – Перетащите на холст компонент вектора **Unit X (Унифицированный по X)**	
13.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст два **Числовых Слайдера**	
14.	Дважды кликните на первом **Числовом Слайдере** и задайте следущее: Name (Имя): Offset Distance (Расстояние Сдвига) Rounding (Округление): Integer (Целые числа) Lower Limit (Нижний Предел): 1 Upper Limit (Верхний Предел): 10 Value (Значение): 4	
15.	Дважды кликните на втором **Числовом Слайдере** и задайте следущее: Name (Имя): # of Offsets (Количество Сдвигов) Rounding (Округление): Even (Чётные числа) Lower Limit (Нижний Предел): 2 Upper Limit (Верхний Предел): 20 Value (Значение): 20	
16.	Подсоедините **Числовой Слайдер** (Offset Distance (Расстояние Сдвига)) ко входу Factor (F) (Коэффициент) компонента **Unit X (Унифицированный Вектор по X)**	
17.	Подсоедините выход Vector (V) (Вектор) компонента **Unit X (Унифицированный Вектор по X)** ко входу Direction (D) (Направление) компонента **Linear Array (Линейный Массив)**	
18.	Подсоедините **Числовой Слайдер** (# of Offsets (Количество Сдвигов)) ко входу Count (N) (Количество) компонента **Linear Array (Линейный Массив)**	



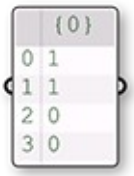
Теперь Вы можете наблюдать во вьюпорте Rhino массив из прямых. Три слайдера позволяют Вам изменять длину этих линий, расстояние между ними и количество линий в массиве.

19.	**Sets/Lists/Dispatch (Наборы/Списки/Диспетчер)** – Перетащите на холст компонент **Dispatch (Диспетчер)**	
20.	Подсоедините выход Geometry (G) (Геометрия) компонента **Linear Array (Линейный Массив)** ко входу List (L) (Список) компонента **Dispatch (Диспетчер)**	
21.	**Params/Input/Panel (Параметры/Вводные/Текстовая Панель)** – Перетащите на холст компонент **Panel (Текстовая Панель)**	
22.	Дважды кликните по **Текстовой Панели**, развыделите Multiline Data (Многострочные Данные), Wrap Items (Заворачивать Элементы) и Special Codes (Специальные Коды), а затем введите следующее: true false	
23.	Подсоедините **Текстовую Панель** ко входу Pattern (P) (Шаблон) компонента **Dispatch (Диспетчер)**	
24.	**Curve/Division/Divide Curve (Кривая/Разделение/Разделить Кривую)** – Перетащите на холст два компонента **Divide Curve (Разделить Кривую)**	
25.	Подсоедините выход List A (A) (Список A) компонента **Dispatch (Диспетчер)** ко входу Curve (C) (Кривая) первого компонента **Divide Curve (Разделить Кривую)**	

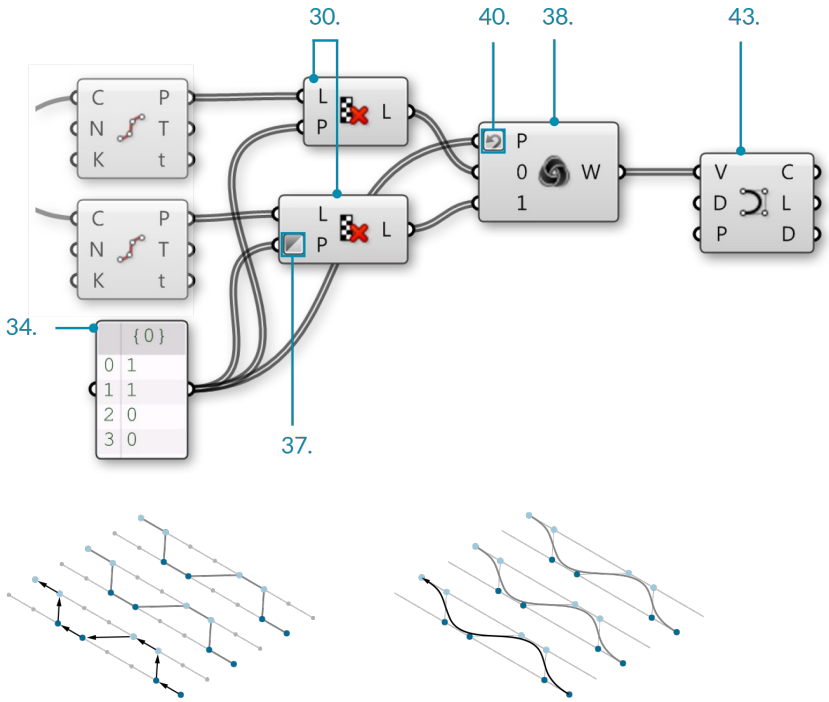
26.	Подсоедините выход List B (B) (Список B) компонента **Dispatch (Диспетчер)** ко входу Curve (C) (Кривая) второго компонента **Divide Curve (Разделить Кривую)**	
27.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст **Числовой Слайдер**	
28.	Дважды кликните на втором **Числовом Слайдере** и задайте следущее: Name (Имя): Divisions (Деления) Rounding (Округление): Integer (Целые числа) Lower Limit (Нижний Предел): 0 Upper Limit (Верхний Предел): 20 Value (Значение): 20	
29.	Подсоедините **Числовой Слайдер** (Divisions (Деления)) ко входу Count (N) (Счёт) обоих компонентов **Divide Curve (Разделить Кривую)**.	



1. Компонент Dispatch (Диспетчер) отправляет каждой второй кривой в массиве команду на разделение списка.
2. Компонент Divide Curve (Разделить Кривую) разделяет кривую на количество сегментов, заданное слайдером. Откорректируйте слайдер, чтобы изменить количество точек.

30.	**Sets/Sequence/Cull Pattern (Наборы/Последовательность/Шаблон Отбрасывания)** – Перетащите на холст два компонента **Cull Pattern (Шаблон Отбрасывания)**	
31.	Подсоедините выход Points (P) (Точки) первого компонента **Divide Curve (Разделить Кривую)** ко входу List (L) (Список) первого компонента **Cull Pattern (Шаблон Отбрасывания)**	
32.	Подсоедините выход Points (P) (Точки) второго компонента **Divide Curve (Разделить Кривую)** ко входу List (L) (Список) второго компонента **Cull Pattern (Шаблон Отбрасывания)**	
33.	**Params/Input/Panel (Параметры/Вводные/Текстовая Панель)** – Перетащите на холст второй компонент **Panel (Текстовая Панель)**	
34.	Дважды кликните по второй **Текстовой Панели**, развыделите Multiline Data (Многострочные Данные), Wrap Items (Заворачивать Элементы) и Special Codes (Специальные Коды), а затем введите следующее: 1 1 0 0 В данном случае мы используем 1 и 0 вмето true (ИСТИНА) и false (ЛОЖЬ). Grasshopper допускает эти две подмены в качестве определения булевых (логических) значений.	
35.	Подсоедините вторую **Текстовую Панель** ко входу Pattern (P) (Шаблон) первого компонента **Cull Pattern (Шаблон Отбрасывания)**	
36.	Подсоедините вторую **Текстовую Панель** ко входу Pattern (P) (Шаблон) второго компонента **Cull Pattern (Шаблон Отбрасывания)**	
37.	Кликните правой кнопкой мыши по входу Pattern (P) (Шаблон) второго компонента **Cull Pattern (Шаблон Отбрасывания)** и выберите Invert (Инвертировать) Это инвертирует (обратит) Шаблон Отбрасывания. Полезный трюк, чтобы сохранять дефинишин более коротким.	
38.	**Sets/List/Weave (Наборы/Списки/Сплести)** – Перетащите на холст компонент **Weave (Сплести)**	
39.	Подсоедините вторую **Текстовую Панель** ко входу Pattern (P) (Шаблон) компонента **Weave (Сплести)**	

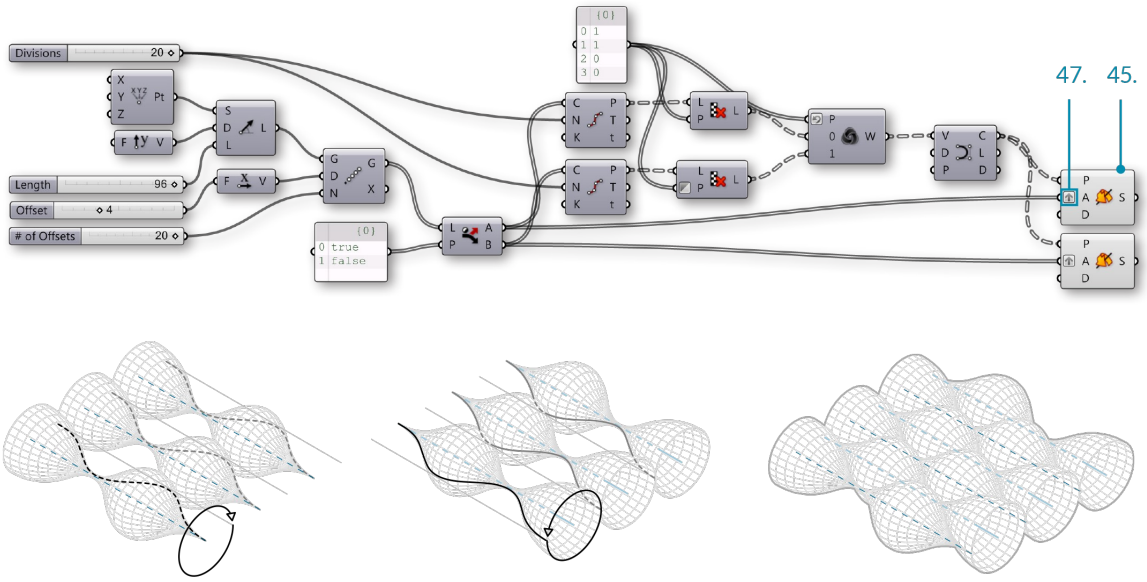
40.	Кликните правой кнопкой мыши по входу Pattern (P) (Шаблон) компонента **Weave (Сплести)** и выберите Reverse (Обратить)	
41.	Подсоедините выход List (L) (Список) первого компонента **Cull Pattern (Шаблон Отбрасывания)** ко входу Stream 0 (0) (Поток 0) компонента **Weave (Сплести)**	
42.	Подсоедините выход List (L) (Список) второго компонента **Cull Pattern (Шаблон Отбрасывания)** ко входу Stream 1 (1) (Поток 1) компонента **Weave (Сплести)**	
43.	**Curve/Spline/Nurbs Curve (Кривая/Сплайн/ Nurbs-Кривая)** – Перетащите на холст компонент **Nurbs Curve (Nurbs-Кривая)**	
44.	Подсоедините выход Weave (W) (Сплетение) компонента **Weave (Сплести)** ко входу Vertices (V) (Вершины) компонента **Nurbs Curve (Nurbs-Кривая)**.	

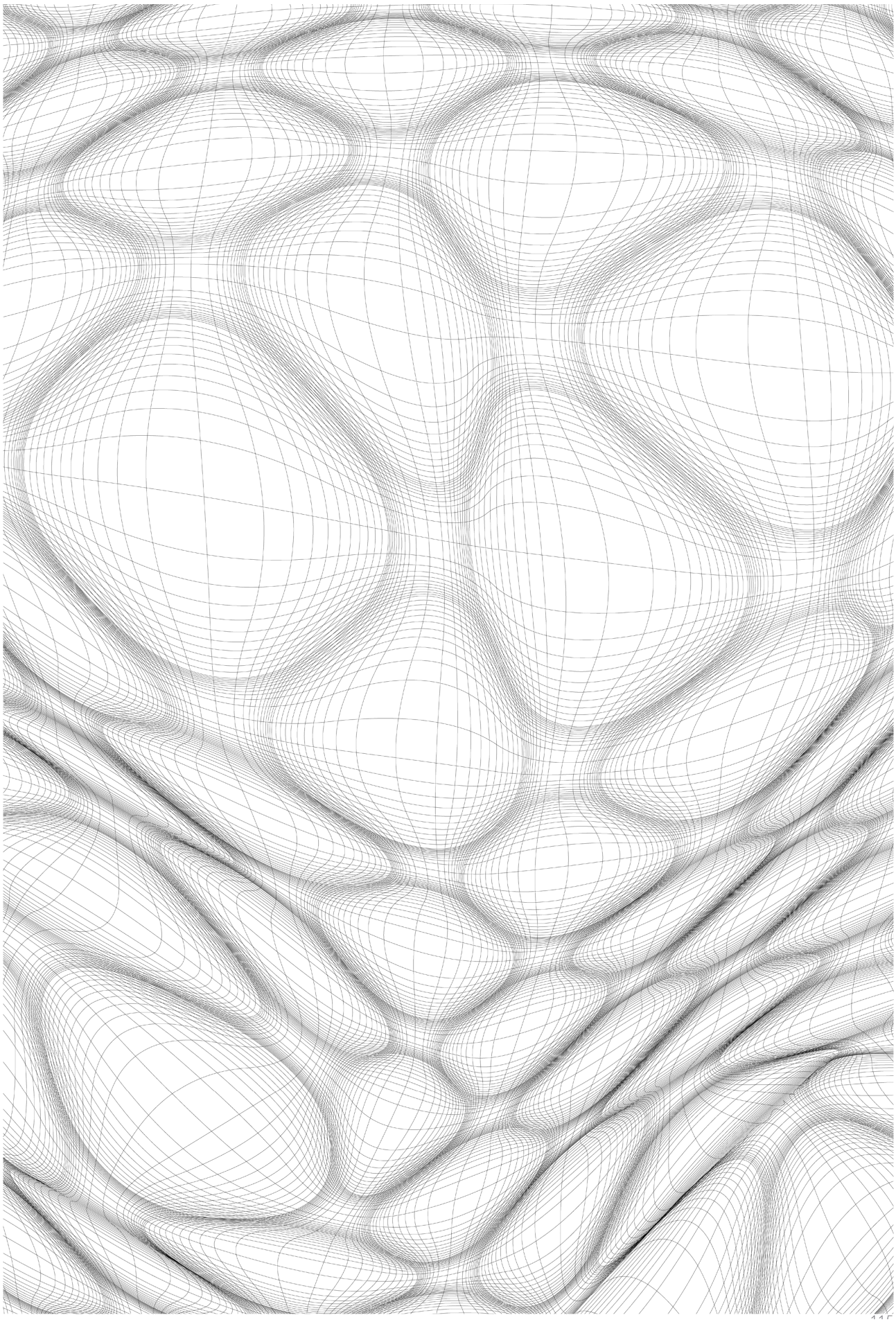


1. Шаблон отбрасывания (cull pattern) удаляет чередующиеся точки из каждого списка.
2. Компонент Weave (Сплести) собирает данные из списков точек, руководствуясь пользовательским шаблоном. Эти данные поступают на компонент Interpolate (Интерполировать) для создания кривых.

45.	**Surface/Freeform/Revolution (Поверхность/Произвольной формы/Вращением)** – Перетащите на холст компонент **Revolution (Поверхность Вращения)**	

46.	Подсоедините выход Curve (Кривая) компонента **Nurbs Curve (NURBS-кривая)** ко входу Profile Curve (P) (Кривая Профиля) обоих компонентов **Revolution (Поверхность Вращения)**.	
47.	Кликните правой кнопкой мыши на входе Axis (A) (Ось Вращения) обоих компонентов **Revolution (Поверхность Вращения)** и выберите Graft (Привить).	
48.	Подсоедините выход List A (A) (Список A) компонента **Dispatch (Диспетчер)** ко входу Axis (A) (Ось A) первого компонента **Revolution (Поверхность Вращения)**	
49.	Соедините выход List B (B) (Список B) компонента **Dispatch (Диспетчер)** ко входу Axis (A) (Ось Вращения) второго компонента **Revolution (Поверхность Вращения)**. Выделите все компоненты, кроме двух компонентов Revolution (Поверхность Вращения) и отключите предварительный просмотр. Это полезно: включать предпросмотр, пока Вы составляете дефинишин, а потом отключать предпросмотр всего, кроме итоговой геометрии.	

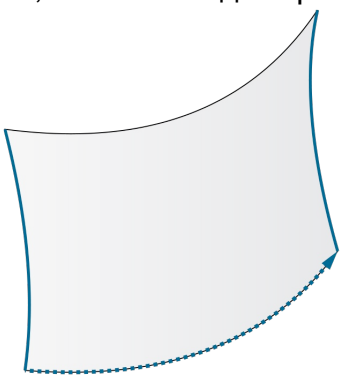




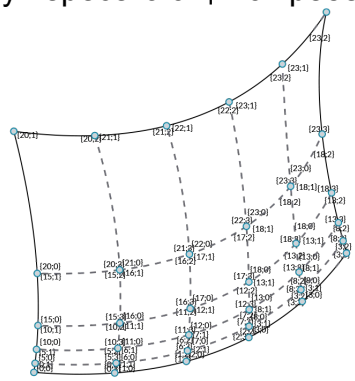
1.5.4. Работа с Деревьями Данных (Data Trees)

Загрузить файл примера, относящегося к данному разделу.

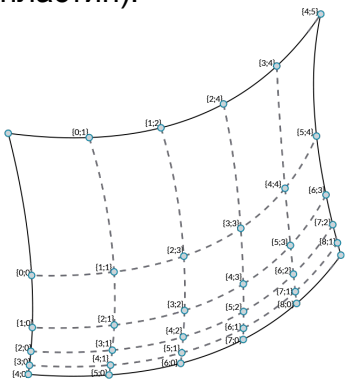
В этом примере мы будем использовать некоторые инструменты Grasshopper для манипулирования деревьями данных, чтобы нужные данные извлечь, реорганизовать и интерполировать необходимые точки, содержащиеся в дереве данных, а также создать решётку пересекающихся рёбер (пластин).



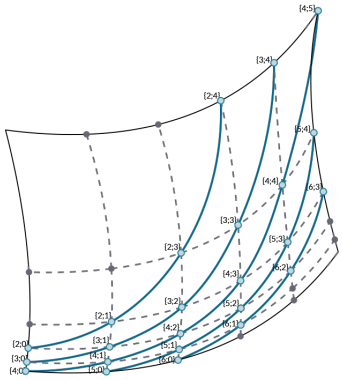
1.



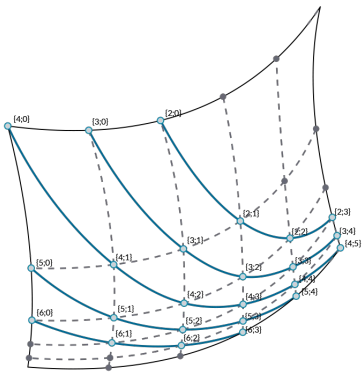
2.



3.



4.

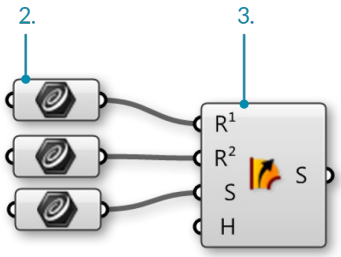


5.

6.

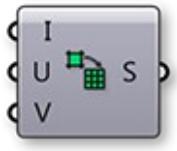
1. Создание NURBS-поверхности с помощью протяжки по двум рельсам.
2. Деление поверхности на сегменты разной длины, извлечение вершин. Данные имеют структуру одного списка, где каждый сегмент определяется четырьмя точками.
3. Переворачивание матрицы для изменения структуры данных. Теперь данные имеют структуру, в которой каждый сегмент определён четырьмя списками, содержащими информацию о единственной угловой точке для каждого сегмента.
4. Разрыв дерева для того, чтобы соединить угловые точки, и для построения диагональных линий, пересекающих каждый сегмент.
5. Сокращение дерева для отбраковки ветвей, не содержащих достаточное количество точек для конструирования NURBS-кривой третьей степени, а также интерполяции точек.
6. Экструзия (выдавливание) пластин для создания пересекающихся пластин.

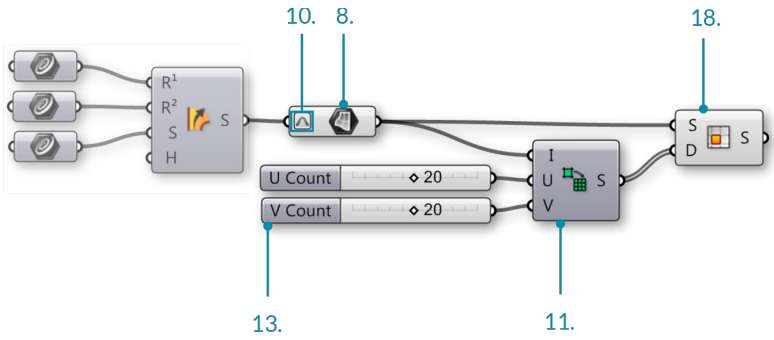
1.	Начните новый дефинишин, введя Ctrl+N (в Grasshopper)	
2.	**Params/Geometry/Curve (Параметры/Геометрия/Кривая)** – Перетащите на холст три параметра **Curve (Кривая)**	
3.	**Surface/Freeform/Sweep2 (Поверхность/Произвольной формы/Протяжкой по 2м рельсам)** – Перетащите на холст компонент **Sweep2 (Протяжка по 2м рельсам)**	
4.	Кликните правой кнопкой мыши по первому параметру **Curve (Кривая)** и выберите “Set one curve” (Выбрать Одну Кривую). Укажите во вьюпорте Rhino первую кривую рельсы (направляющей).	
5.	Кликните правой кнопкой мыши по второму параметру **Curve (Кривая)** и выберите “Set one curve” (Выбрать Одну Кривую). Укажите во вьюпорте Rhino вторую кривую рельсы.	
6.	Кликните правой кнопкой мыши по третьему параметру **Curve (Кривая)** и выберите “Set one curve” (Выбрать Одну Кривую). Укажите во вьюпорте Rhino кривую сечения.	
7.	Подсоедините выходы параметров **Curve (Кривая)** ко входам Rail 1 (R1) (Рельса 1), Rail 2 (R2) (Рельса 2) и Sections (S) (Секущие) компонента **Sweep2 (Протяжка по 2м Рельсам)** соответственно.	



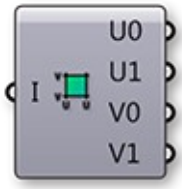
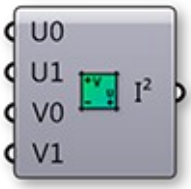
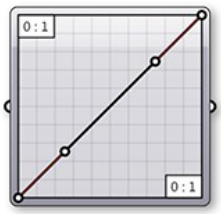
Мы только что создали NURBS-поверхность

8.	**Params/Geometry/Surface (Параметры/Геометрия/Поверхность)** – Перетащите на холст параметр **Surface (Поверхность)**	
9.	Подсоедините выход Brep (S) (Геометрия, представленная в виде внешних поверхностей) компонента **Sweep2 (Протяжка по 2м рельсам)** ко входу параметра **Surface (Поверхность)**	
10.	Кликните правой кнопкой мыши по параметру **Surface (Поверхность)** и выберите “Reparameterize”	

	(Репараметризировать). На этом шаге мы выполнили переналожение U и V доменов поверхности между 0 и 1. Это сделает возможными последующие операции.	
11.	**Maths/Domain/Divide Domain2 (Математика/Домен/Разделить Двумерный Домен)** – Перетащите на холст компонент **Divide Domain2 (Разделить Двумерный Домен)**	
12.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст два **Числовых Слайдера**	
13.	Дважды кликните по первому **Числовому Слайдеру** и задайте следующее: Rounding (Округление): Integer (Целые числа) Lower Limit (Нижний Предел): 1 Upper Limit (Верхний Предел): 40 Value (Значение): 20	
14.	Задайте те же значения второму **Числовому Слайдеру**	
15.	Подсоедините выход параметра репараметризированной поверхности (**Surface**) ко входу Domain (I) (Домен) компонента **Divide Domain2 (Разделить Двумерный Домен)**	
16.	Подсоедините первый **Числовой Слайдер** ко входу U Count (U) (Число по Горизонтали) компонента **Divide Domain2 (Разделить Двумерный Домен)**	
17.	Подсоедините второй **Числовой Слайдер** ко входу V Count (V) (Число по Вертикали) компонента **Divide Domain2 (Разделить Двумерный Домен)**	
18.	**Surface/Util/Isotrim (Поверхность/Утилиты/Обрезка по Изопарме)** – Перетащите на холст компонент **Isotrim (Обрезка по Изопарме)**	
19.	Подсоедините выход Segments (S) (Сегменты) **компонента Divide Domain2 (Разделить Двумерный Домен)** ко входу the Domain (D) (Домен) компонента **Isotrim (Обрезка по Изопарме)**	
20.	Подсоедините выход параметра **Surface (Поверхность)** ко входу Surface (S) (Поверхность) компонента **Isotrim (Обрезка по Изопарме)**	

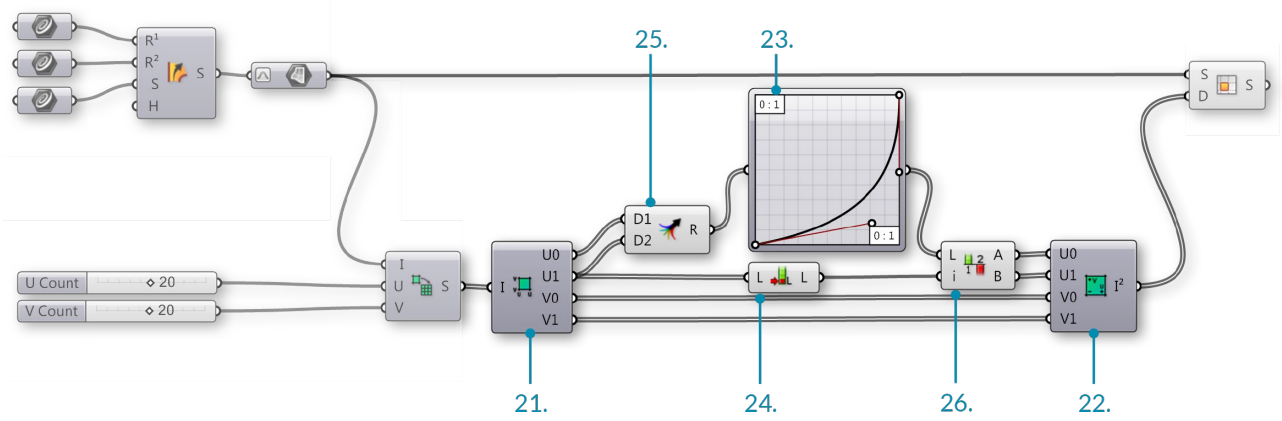


Мы только что разделили единую поверхность на меньшие, равного размера поверхности. Корректируя слайдеры U и V Count (Число по Горизонтали и Вертикали) Вы можете изменить количество разбиений. Давайте добавим Graph Mapper (Переналожение по Графу), чтобы задать сегментам переменный размер.

21.	**Maths/Domain/Deconstruct Domain2 (Математика/Домен/Разобрать Двумерный Домен)** – Перетащите на холст компонент **Deconstruct Domain2 (Разобрать Двумерный Домен)**	
22.	**Maths/Domain/Construct Domain2 (Математика/Домен/Сконструировать Двумерный Домен)** – Перетащите на холст компонент **Construct Domain2 (Разобрать Двумерный Домен)**	
23.	**Params/Input/Graph Mapper (Параметры/Вводные/Переналожение по Графу)** – Перетащите на холст компонент **Graph Mapper (Переналожение по Графу)**	
24.	**Sets/List/List Length (Наборы/Список/Длина Списка)** – Перетащите на холст компонент **List Length (Длина Списка)**	
25.	**Sets/Tree/Merge (Наборы/Дерево Данных/Слияние)** – Перетащите на холст компонент **Merge (Слияние)**	
26.	**Sets/List/Split List (Наборы/Список/Разбить Список)** – Перетащите на холст компонент **Split List (Разбить Список)** Компоненты Слияние (Merge) и Разбиение (Split)	

	использованы для подачи в Graph Mapper (Переналожение по Графу) обоих значений: и U min, и U max.	
27.	Подсоедините выходы U min (U0) и U max (U1) (Минимальное и Максимальное значения по Горизонтали) компонента **Deconstruct Domain2 (Разобрать Двумерный Домен)** ко входам Data 1 (D1) и Data 2 (D2) (Данные для Слияния) компонента **Merge (Слияние)**	
28.	Подсоедините выход Result (R) (Результат) компонента **Merge (Слияние)** ко входу **Graph Mapper (Переналожение по Графу)**	
29.	Кликните по компоненту **Graph Mapper (Переналожение по Графу)** правой кнопкой мыши и выберите «Bezier» (Безье) из «Graph Types» (Типы Графов)	
30.	Подсоедините вторым проводом выход U max (U1) (Максимум по Горизонтали) компонента **Deconstruct Domain2 (Разобрать Двумерный Домен)** ко входу List (L) (Список) компонента **List Length (Длина Списка)**	
31.	Соедините выход компонента **Graph Mapper (Переналожение по Графу)** ко входу List (L) (Список) компонента **Split List (Разбить Список)**	
32.	Подсоедините выход Length (L) (Длина) компонента **List Length (Длина Списка)** ко входу Index (i) (Индекс) компонента **Split List (Разбить Список)**	
33.	Подсоедините выход List A (A) (Список A) компонента **Split List (Разбить Список)** ко входу U min (U0) (Минимум по Горизонтали) комопнента **Construct Domain2 (Сконструировать Двумерный Домен)**	
34.	Подсоедините выход List B (B) (Список B) компонента **Split List (Разбить Список)** ко входу U max (U1) (Максимум по Горизонтали) комопнента **Construct Domain2 (Сконструировать Двумерный Домен)**	
35.	Подсоедините выход V min (V0) (Минимум по Вертикали) компонента **Deconstruct Domain2 (Разобрать Двумерный Домен)** ко входу V min (V1) (Минимум по Вертикали) комопнента **Construct Domain2 (Сконструировать Двумерный Домен)**	
36.	Подсоедините выход V max (V1) (Максимум по Вертикали) компонента **Deconstruct Domain2 (Разобрать Двумерный Домен)** ко входу V max (V1) (Максимум по Вертикали) комопнента **Construct Domain2 (Сконструировать Двумерный Домен)**	
37.	Подсоедините выход 2D Domain (I2) (Двумерный Домен) компонента **Construct Domain2 (Сконструировать Двумерный Домен)** ко входу Domain (D) (Домен) компонента **Isotrim (Обрезка по Изопарме)**, заменив	

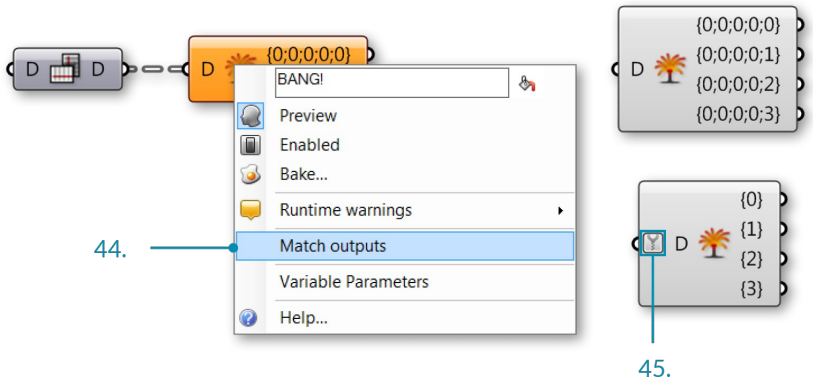
	существующее соединение	



Мы только что разобрали домен каждого сегмента поверхности, переназначили значения U (по Горизонтали), используя Graph Mapper (Переналожение по Графу), чтобы изменить распределение участков поверхности. Теперь давайте попробуем использовать деревья данных для манипуляции разделением поверхностей.

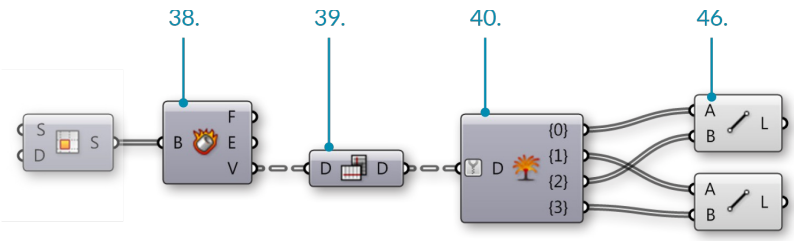
38.	**Surface/Analysis/Deconstruct Brep (Поверхность/Анализ/Разобрать Brep)** – Перетащите на холст компонент **Deconstruct Brep (Разобрать Brep)**	
39.	**Sets/Tree/Flip Matrix (Наборы/Дерево Данных/Перевернуть Матрицу)** – Перетащите на холст компонент **Flip Matrix (Перевернуть Матрицу)**	
40.	**Sets/Tree/Explode Tree (Наборы/Дерево данных/Разорвать Дерево)** – Перетащите на холст компонент **Explode Tree (Разорвать Дерево Данных)**	
41.	Подсоедините выход Surface (S) (Поверхность) компонента **Isotrim (Обрезка по Изопарме)** ко входу Brep (B) компонента **Deconstruct Brep (Разобрать Brep)** Компонент Deconstruct Brep (Разобрать Brep) разбирает Brep (объект, представленный в виде внешних поверхностей) на Грани (Faces), Края (Edges) и Вершины (Vertices). Это полезно, когда Вы хотите оперировать отдельной составляющей поверхности.	
42.	Подсоедините выход Vertices (V) (Вершины) компонента **Deconstruct Brep (Разобрать Brep)** ко входу Data (D) (Данные) компонента **Flip Matrix (Перевернуть Матрицу)** Мы просто изменим структуру дерева данных так, чтобы из одного списка, включающего 4 вершины, определяющих	

	каждую поверхность сделать 4 списка, каждый из которых содержит одну вершину каждой поверхности.	
43.	Подсоедините выход Data (D) (Данные) компонента **Flip Matrix (Перевернуть Матрицу)** ко входу Data (D) (Данные) компонента **Explode Tree (Разорвать Дерево Данных)**	
44.	Кликните правой кнопкой мыши по компоненту **Explode Tree (Разорвать Дерево Данных)** и выберите «Match Outputs» (Привести в соответствие данные на выходах)	
45.	Кликните правой кнопкой мыши по входу Data (D) (Данные) комопнента **Explode Tree (Разорвать Дерево Данных)** и выберите «Simplify» (Упростить)	



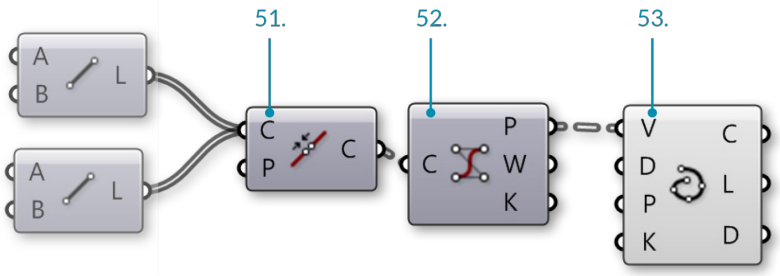
Каждый выход комопнента Explode Tree (Разорвать Дерево Данных) содержит список с одной вершиной каждой поверхности. Иными словами, один список со всеми верхними правыми углами, один список со всеми нижними правыми углами, один список со всеми верхними левыми углами и один список со всеми нижними левыми углами.

46.	**Curve/Primitive/Line (Кривая/Примитивы/Прямая)** – Перетащите на холст два компонента **Line (Прямая)**	
47.	Подсоедините выход Branch 0 {0} (Ветвь 0) компонента **Explode Tree (Разорвать Дерево Данных)** ко входу Start Point (A) (Точка Начала) первого компонента **Line (Прямая)**	
48.	Подсоедините выход Branch 1 {1} (Ветвь 1) компонента **Explode Tree (Разорвать Дерево Данных)** ко входу Start Point (A) (Точка Начала) второго компонента **Line (Прямая)**	
49.	Подсоедините выход Branch 2 {2} (Ветвь 2) компонента **Explode Tree (Разорвать Дерево Данных)** ко входу End Point (A) (Точка Конца) первого компонента **Line (Прямая)**	
50.	Подсоедините выход Branch 3 {3} (Ветвь 3) компонента **Explode Tree (Разорвать Дерево Данных)** ко входу End Point (A) (Точка Конца) второго компонента **Line (Прямая)**	

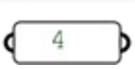


Только что мы соединили угловые точки каждой поверхности по её диагонали отрезками прямых.

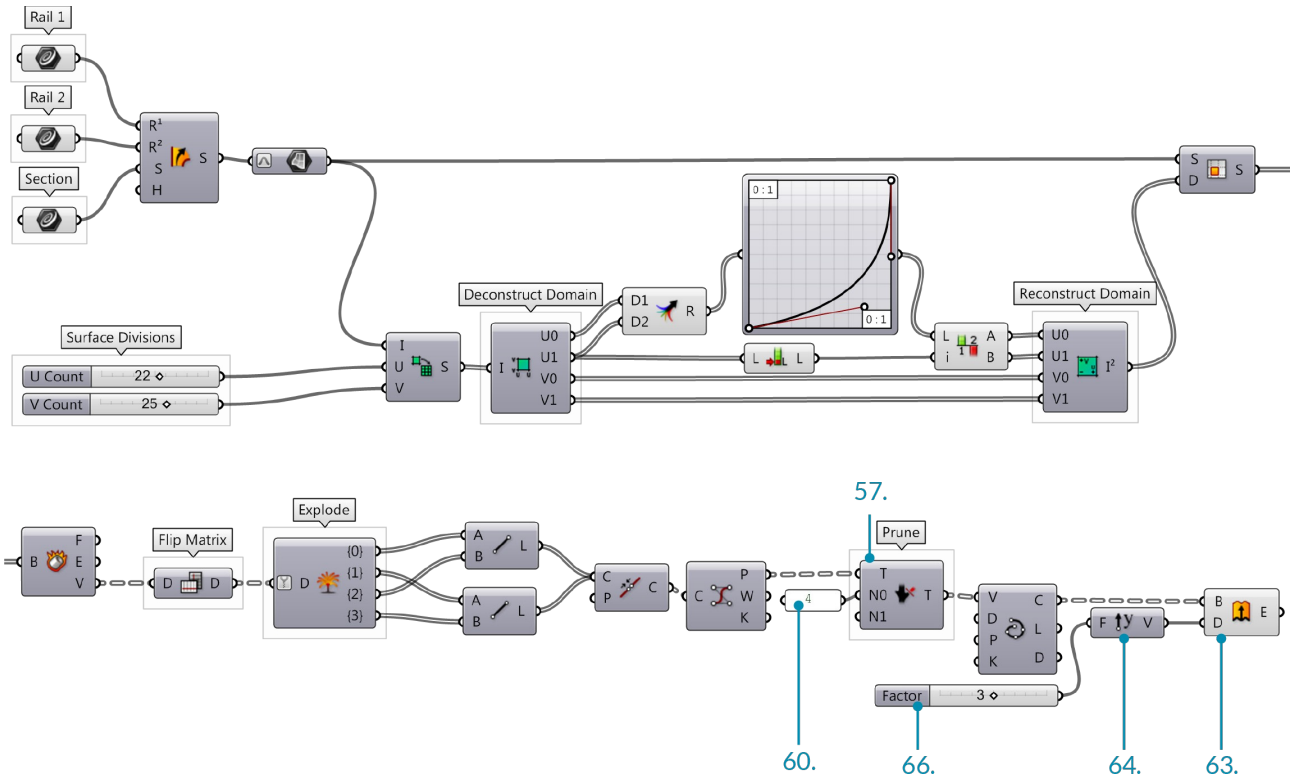
51.	**Curve/Util/Join Curves (Кривая/Утилиты/Объединить Кривые)** – Перетащите на холст компонент **Join Curves (Объединить Кривые)**	
52.	**Curve/Analysis/Control Points (Кривая/Анализ/Контрольные Точки)** – Перетащите на холст компонент **Control Points (Контрольные Точки)**	
53.	**Curve/Spline/Interpolate (Кривая/Сплайн/Интерполировать)** – Перетащите на холст компонент **Interpolate (Интерполировать)**	
54.	Подсоедините выходы Line (L) (Прямая) каждого компонента **Line (Прямая)** ко входу Curves (C) (Кривые) компонента **Join Curves (Объединить Кривые)** Удерживайте нажатой клавишу Shift, чтобы подсоединить несколько проводов к одному входу.	
55.	Подсоедините выход Curves (C) (Кривые) компонента **Join Curves (Объединить Кривые)** ко входу Curve (C) (Кривая) компонента **Control Points (Контрольные Точки)**	
56.	Подсоедините выход Points (P) (Точки) компонента **Control Points (Контрольные Точки)** ко входу Vertices (V) (Вершины) компонента **Interpolate (Интерполировать)**	



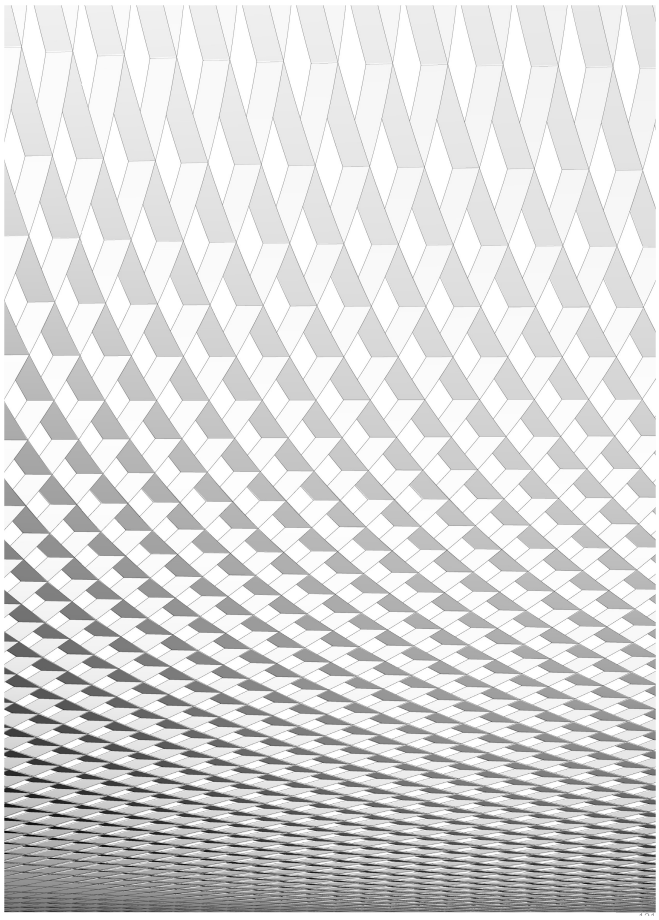
Сейчас мы объединили наши прямые в ломаные и реконструировали их в NURBS-кривые, интерполируя их через контрольные точки. Во вьюпорте Rhino Вы могли заметить, что наиболее короткие кривые всё ещё вполне прямые. Это произошло оттого, что NURBS-кривая со степенью 3 не может быть создана по менее, чем четырём контрольным точкам. Давайте произведём манипуляцию с деревом данных, чтобы устранить из списков контрольных точек те, в которых содержится менее четырёх элементов.

57.	**Sets/Tree/Prune Tree (Наборы/Дерево Данных/Подрезать Дерево)** – Перетащите на холст компонент **Prune Tree (Подрезать Дерево Данных)**	
58.	**Params/Input/Panel (Параметры/Вводные/Текстовая Панель)** – Перетащите **Текстовую Панель** на холст	
59.	Подсоедините выход Points (P) (Точки) компонента **Control Points (Контрольные Точки)** ко входу Tree (T) (Дерево Данных) компонента **Prune Tree (Подрезать Дерево)** Если Вы подсоедините один Param Viewer (Просмотрщик Параметра) к выходу Points (P) (Точки) компонента Control Points (Контрольные Точки), а второй — к выходу Tree (T) (Дерево Данных) компонента Prune Tree (Подрезать Дерево), то Вы сможете увидеть, что количество ветвей сократилось.	
60.	Дважды кликните по **Текстовой Панели** и введите цифру 4.	
61.	Подсоедините выход **Текстовой Панели** ко входу Minimum (N0) (Минимум) компонента **Prune Tree (Подрезать Дерево Данных)**	
62.	Подсоедините выход Tree (T) (Дерево Данных) компонента **Prune Tree (Подрезать Дерево Данных)** ко входу Vertices (V) (Вершины) компонента **Interpolate (Интерполировать)**	

63.	**Surface/Freeform/Extrude (Поверхность/Произвольной формы/Выдавливание)** – Перетащите на холст компонент **Extrude (Выдавливание (Экструзия))**	
64.	**Vector/Vector/Unit Y (Вектор/Вектор/Унифицированный по Y)** – Перетащите на холст компонент **Unit Y (Унифицированный по Y)** Возможно, Вам понадобится компонент вектора другого направления, например, Unit X (Унифицированный по X). Это зависит от ориентации вашей ссылочной Rhino-геометрии.	
65.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** – Перетащите на холст **Числовой Слайдер**	
66.	Дважды кликните на **Числовом Слайдере** и задайте следующее: Rounding (Округление): Integer (Целые числа) Lower Limit (Нижний Предел): 1 Upper Limit (Верхний Предел): 5 Value (Значение): 3	
67.	Подсоедините выход Curve (C) (Кривая) компонента **Interpolate (Интерполировать)** ко входу Base (B) (Основа) компонента **Extrude (Выдавливание)**	
68.	Подсоедините Числовой Слайдер ко входу **Factor (F) (Коэффициент)** компонента **Unit Y (Унифицированный по Y)**	
69.	Подсоедините выход Unit Vector (V) (Унифицированный Вектор) компонента **Unit Y (Унифицированный по Y)** ко входу Direction (D) (Направление) компонента **Extrude (Выдавливание)**	

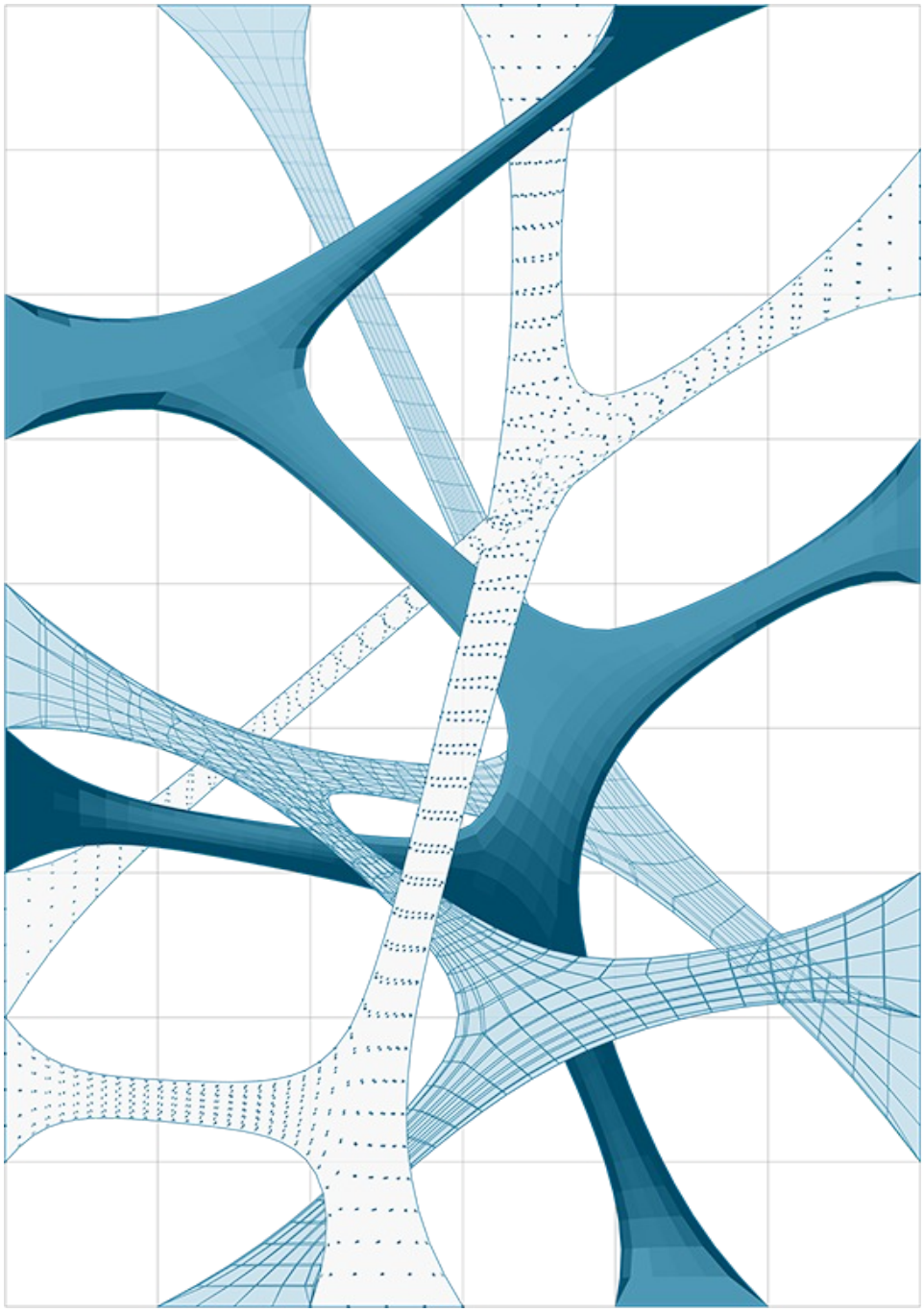


Теперь Вы должны увидеть диагональную сетку полос из пластин во вьюпорте Rhino. Корректируйте слайдер Factor (Коэффициент), чтобы менять высоту пластин.



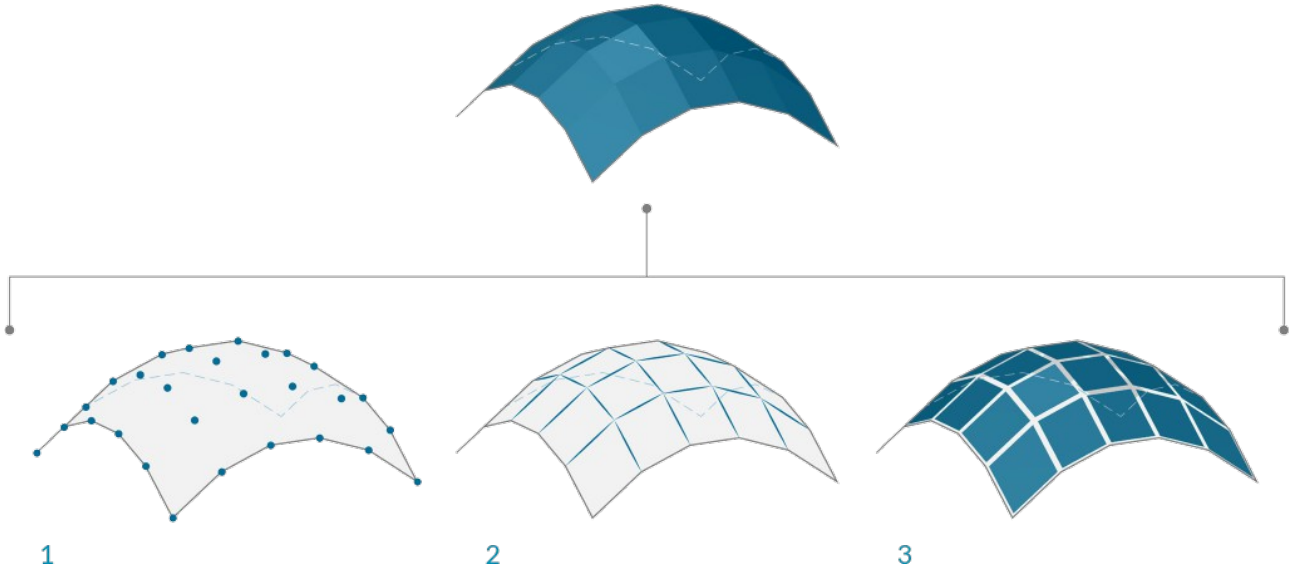
1.6. Полигональные сетки (Meshes). Быстрый Старт.

В области компьютерного моделирования, mesh (полигональные сетки) являются одним из самых распространенных методов передачи формы, представляющей 3D геометрию. Mesh-геометрия может предоставить лёгкую и гибкую альтернативу при работе с NURBS и используется повсюду от рендеринга и визуализации, до цифрового производства и 3D-печати. Эта глава предоставит введение в то, каким образом Grasshopper манипулирует полигнальной геометрией (mesh geometry).



1.6.1 Что такое Mesh?

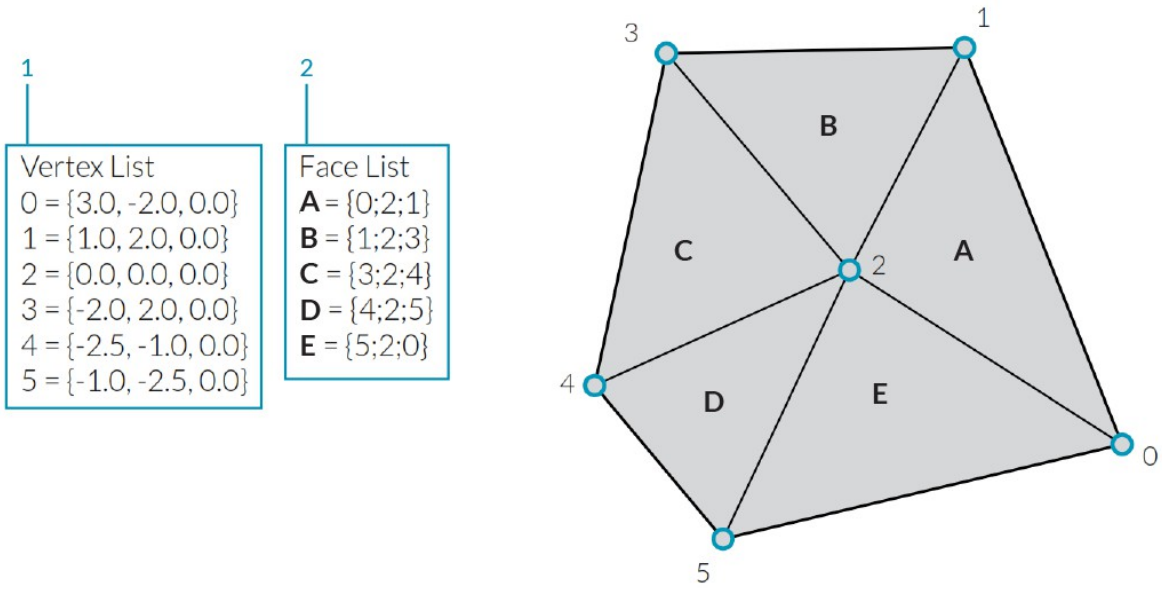
Полигональная сетка (Mesh) это набор четырёхугольников и треугольников, представляющих в совокупности поверхности (surface) или твердотельную (solid) геометрию. В этом разделе обсуждается структура полигонального объекта, включающего в себя вершины (vertices), края (edges) и грани (faces), а также дополнительные свойства, такие как цвета (colors) и нормали (normals).



> Mesh vertices — вершины полигональной сетки
>
> Mesh edges — края полигональной сетки
>
> Mesh faces — грани полигональной сетки

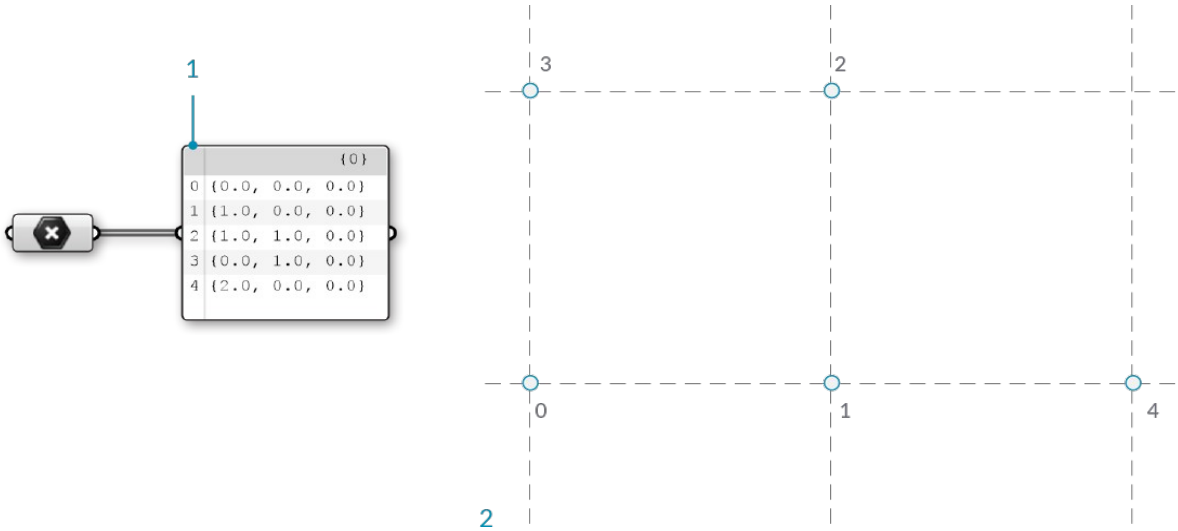
1.6.1.1 Базовая Анатомия Полигональной сетки

Grasshopper определяет полигональные сетки, используя структуру данных типа Face-Vertex (Грань-Вершина). По своей сути, эта структура — просто набор точек, сгруппированных в многоугольники. Точки полигональной сетки называются вершинами, а многоугольники называются гранями. Для создания сетки необходим список вершин и система группирования этих вершин в грани.



1. Список вершин.
2. Грани, сгруппированные из вершин

Вершины (Vertices)

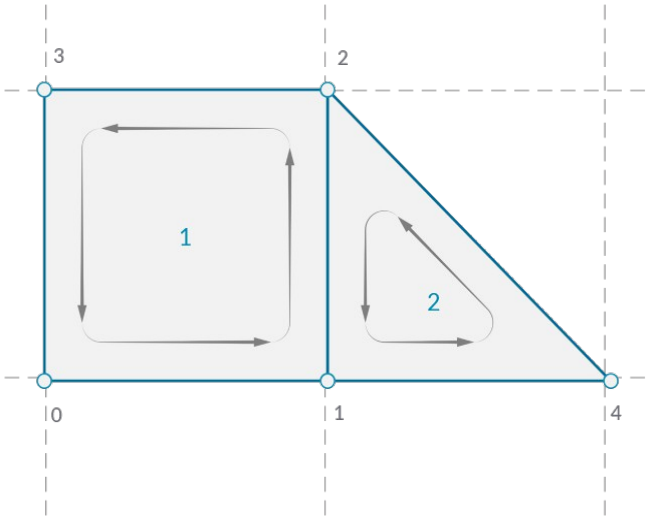


Вершины полигональной сетки — это просто список точек (points). Напомним, что списком (list) в Grasshopper называется набор объектов. Каждый объект в списке имеет индекс (index), описывающий его позицию в списке. Индекс вершин очень важен при создании полигональной сетки, или для того, чтобы получить информацию о структуре сетки.

1. Список точек. Все списки в Grasshopper начинаются с индекса ноль
2. Набор точек, маркированный по их индексам

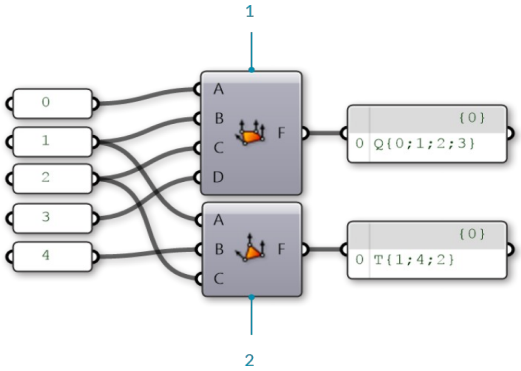
Грани (Faces)

Грань является упорядоченным списком трёх или четырёх вершин. “Поверхность” представлена гранями сетки, следовательно, вершины индексируются в соответствии с их местоположением. У нас уже есть список вершин, образующих сетку, таким образом, чтобы определить грань, вместо представления отдельных точек мы используем индексы вершин. Также это позволяет нам использовать одну и ту же вершину более чем в одной грани.



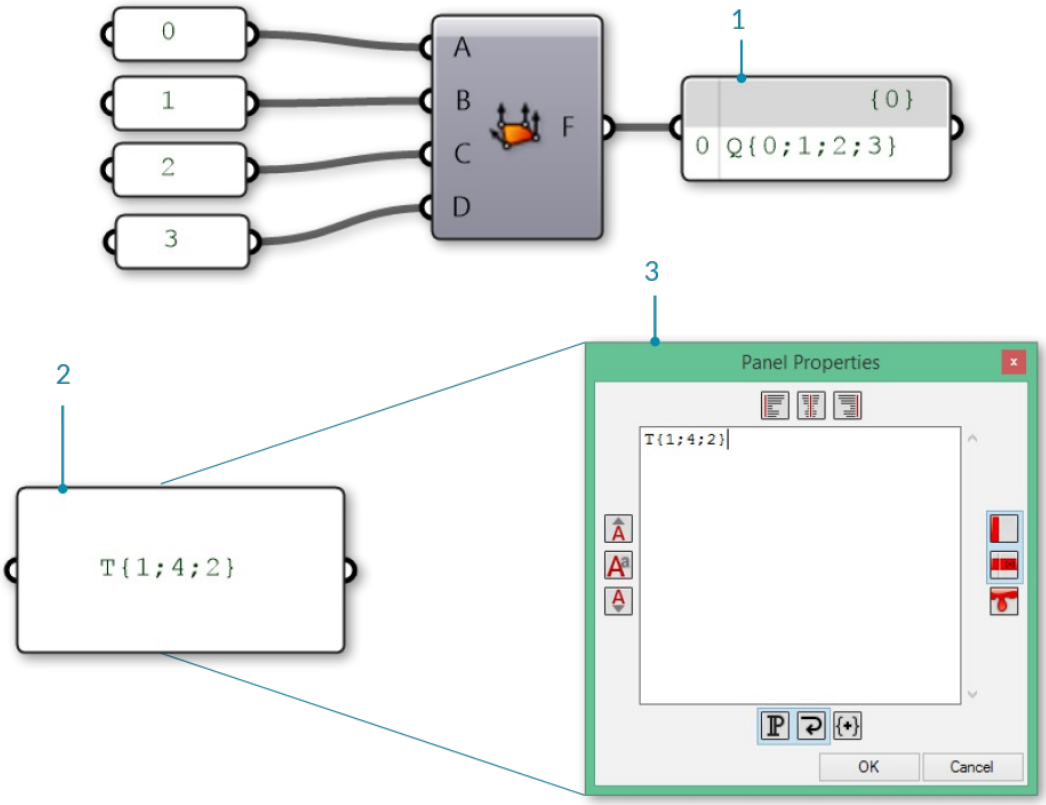
1. Четырёхугольная грань создана по индексам 0, 1, 2 и 3
2. Треугольная грань созданная по индексам 1, 4 и 2

В Grasshopper грани могут быть созданы с помощью компонентов Mesh Triangle (Сетка с треугольными гранями) и Mesh Quad (Сетка с четырёхугольными гранями). Входам у этих компонентам требуются целые числа, которые соответствуют индексам вершин, используемым для грани. Подключив Panel (Текстовую Панель) к выходам этих компонентов, то мы сможем увидеть, что треугольная грань представлена в формате T{A;B;C}, а четырёхугольная грань как Q{A;B;C;D}. Грани с более чем четырьмя сторонами не допускаются. Чтобы создать 5-сторонний полигональный элемент, сетка должна быть разбита на две и более граней.



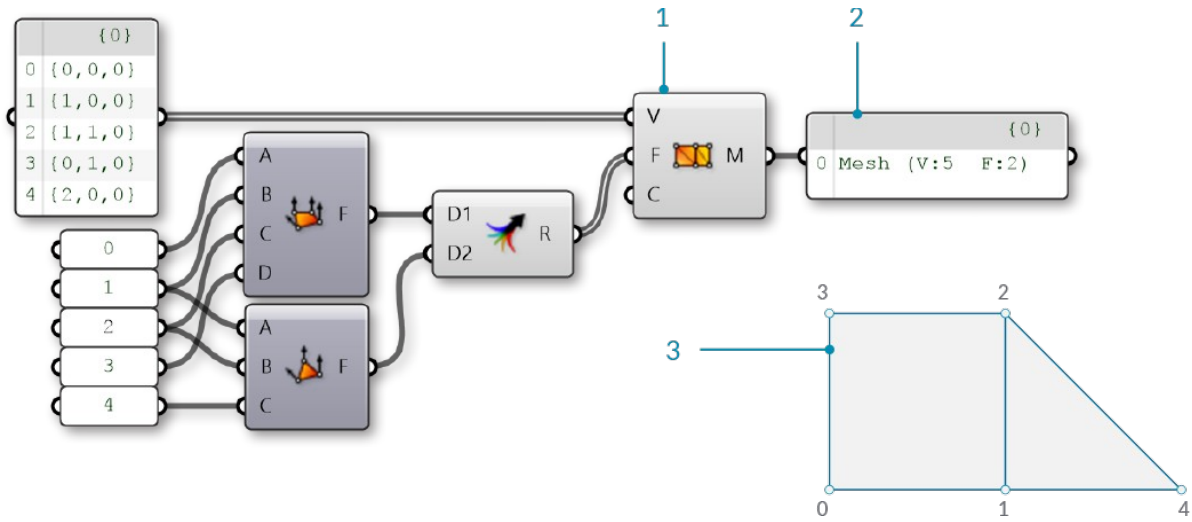
1. Компонент Mesh Quad (Сетка с четырёхугольными гранями) с индексами 0, 1, 2 и 3
2. Mesh Triangle (Сетка с трёхугольными гранями) с индексами 1, 4 и 2

Важно помнить, что данные компоненты в качестве результата не создают полигональную сетку, а выдают данные о том, каким образом сетка должна быть сконструирована. Обращая внимание на формат данного списка, мы можем также создать грань вручную, редактируя его в компоненте Panel (Текстовая Панель), чтобы вводить данные в соответствующем формате для треугольных и четырёхугольных граней.



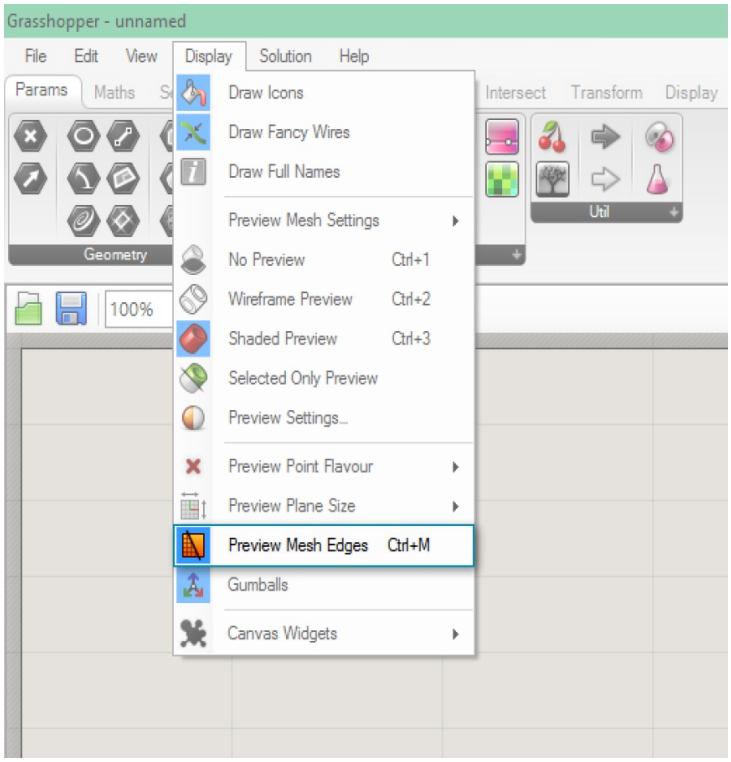
1. Грань, созданная с использованием компонента Mesh Quad (Сетка с четырёхугольными гранями)
2. Грань, созданная с использованием компонента Panel (Текстовая Панель)
3. При двойном клике по Панели автоматически открывается окно Panel Properties (Свойства Панели), также его можно открыть, щёлкнув правой кнопкой мыши на панели и выбрав "Edit Notes..." (Редактировать Примечания...)

Пока у нас есть только список вершин и набор определений граней, но полигональная сетка из них ещё не создана. Для того, чтобы создать сетку мы должны соединить грани и вершины воедино, используя компонент Construct Mesh (Сконструировать Полигональную сетку). Мы подсоединим наш список вершин ко входу V, а список объединения в грани ко входу F. (Этот компонент также имеет место для опционального подключения значения цвета (Color), который обсуждается ниже.) Если подсоединить Панель к выходу Construct Mesh (Сконструировать Полигональную сетку), мы сможем увидеть информацию о количестве граней и числе индексов.

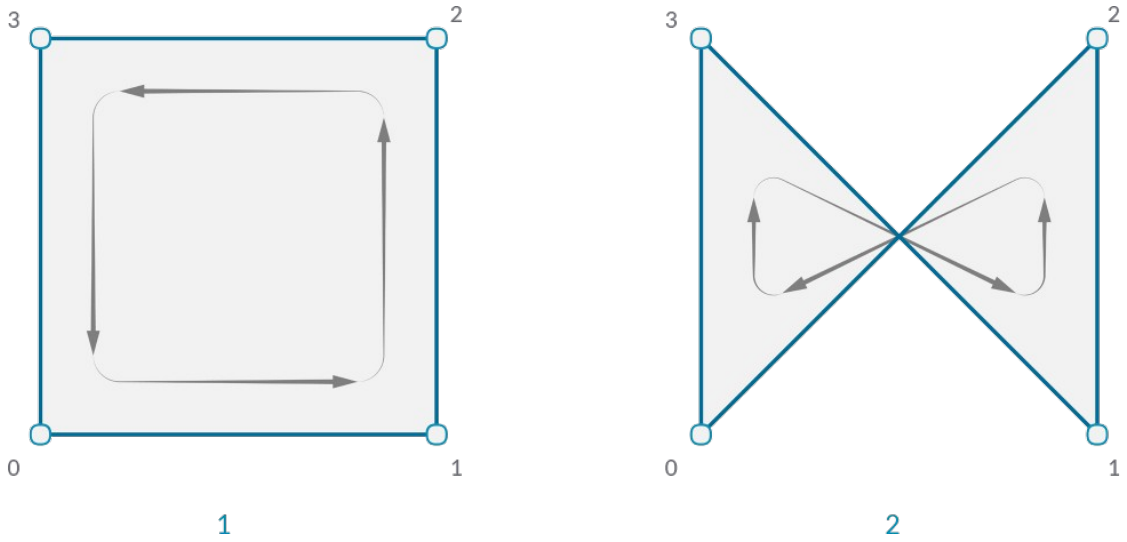


1. Компонент Construct Mesh (Сконструировать Полигональную сетку) принимает на своих соответствующих входах список вершин и список граней. Вход Color (Цвет) является опциональным и пока оставлен пустым.
2. Панель показывает, что мы создали сетку с пятью вершинами и двумя гранями
3. Результирующая полигональная сетка (вершины были помечены их индексами)

По-умолчанию, Grasshopper не отображает края полигональной геометрии. Для предпросмотра краёв сеток, также, как у поверхностей, Вы должны включить предварительный просмотр краёв полигональных сеток, используя клавиши сокращения Ctrl-M, или, перейдя в меню Display (Отображение) и выбрав 'Preview Mesh Edges' (Предварительный просмотр Краёв Полигональной сетки).



Чрезвычайно важно обратить внимание на порядок индексов при конструировании грани полигональной сетки. Грань будет построена соединением вершин, перечисленных по порядку. Четырёхугольная грани Q{0,1,2,3} и Q{1,0,2,3} будут очнь разными, несмотря на использование одних и тех же четырёх вершин. Некорректый порядок вершин может привести к таким проблемам, как отверстия (holes), геометрия с путаницей краёв (non-manifold) или неориентированным поверхностям (non-orientable surfaces). Геометрия такой сетки, как правило, не отображается правильно и не может быть напечатана в 3D. Эти вопросы подробнее осуждаются в разделе Понятие о Топологии.



1. Четырёхугольная грань с индексами 0,1,2,3
2. Четырёхугольная грань с индексами 0,3,1,2

1.6.1.2 Неявные Данные Полигональной сетки (Implicit Mesh Data)

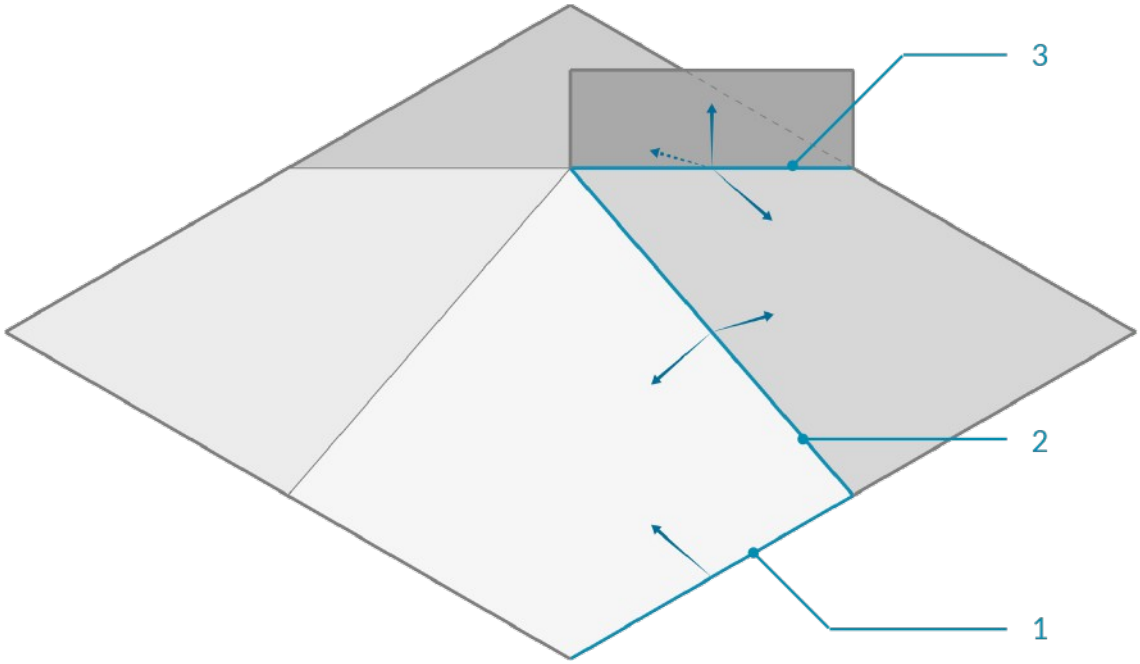
В дополнение к граням и вершинам, может быть иная информация, которую мы можем использовать. В полигональной сетке, основанной на структуре Face-Vertex (Грань-Вершина) есть такие данные, как края (edges) и нормали (normals), которые вычисляются, в отличие от явно заданных граней и вершин. Этот раздел описывает способы добыть данную информацию.

Края (Edges)

Краями полигональной сетки являются линии, объединяющие любые две последовательные вершины в грань. Обратите внимание, что некоторые края используются совместно несколькими гранями, в то время, как другие края принадлежат лишь единственной грани. Количество граней, использующих один край называется валентностью этого края (valence of edge).

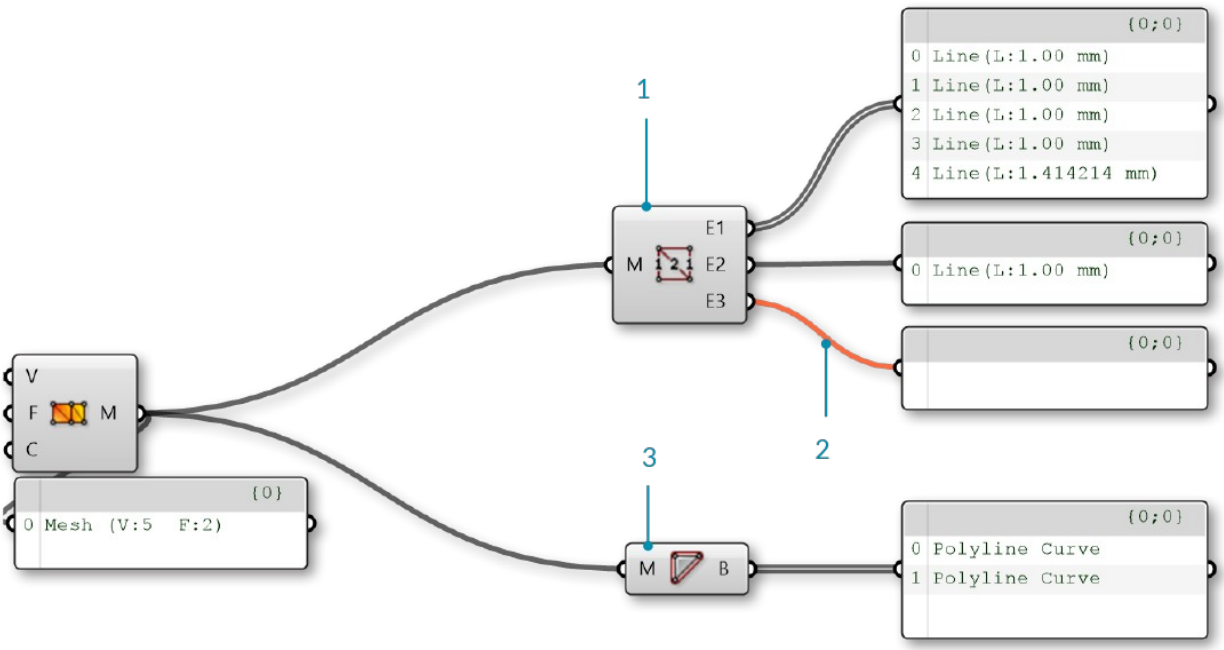
Grasshopper группирует края по трём категориям, основанным на валентности:

1. E1 - 'Naked Edges' (Открытые Края) имеют валентность 1. Они составляют внешнюю границу полигональной сетки.
2. E2 - 'Interior Edges' (Внутренние края) имеют валентность 2.
3. E3 - 'Non-Manifold Edges' (Многосложные (спутанные) Края) имеют валентность 3 или выше. Полигональные сетки, содержащие такие структуры, называются "Non-Manifold" и обсуждаются в следующем разделе.



4. Naked edge (Открытые края) с валентностью 1
5. Interior edge (Внутренние края) с валентностью 2
6. Non-manifold edge (Многосложные края) с валентностью 3

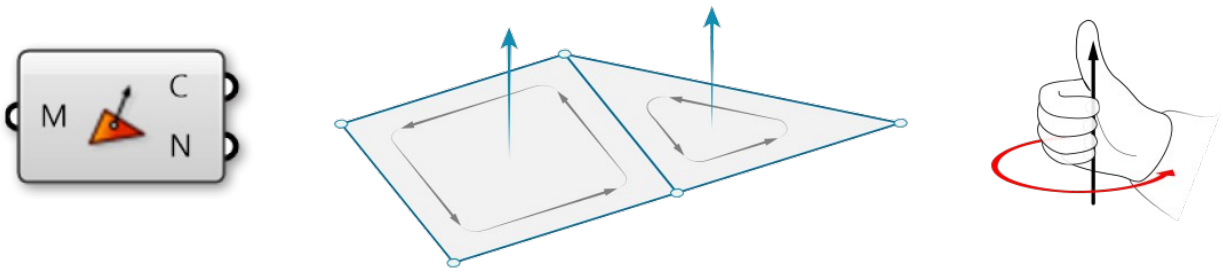
Чтобы извлечь края полигональной сетки в соответствии с их валентностью, Вы можете использовать компонент Mesh Edges (Края Полигональной сетки). Это позволяет найти края, лежащие вдоль внешней границы сетки, а также определить, есть ли многосложные (non-manifold) края. Однако, иногда полезнее иметь информацию о полной границе каждой грани. Для этого мы можем использовать компонент Face Boundaries (Границы Граней). Он извлекает ломаную линию границы каждой грани.



1. Компонент Mesh Edges (Края Полигональной сетки) выводит три набора краёв. Эта сетка содержит 5 открытых (naked) краёв, 1 внутренний (interior) край и не содержит многосложных (non-manifold) краёв
2. Выход E3 пуст, поскольку эта сетка не содержит многосложных (non-manifold) краёв, поэтому и результирующий провод окрашен оранжевым цветом.
3. Компонент Face Boundaries (Границы Граней) выводит по одной ломаной линии (polyline) для каждой грани

Нормали Граней (Face Normals)

Вектор *нормали (normal)* — это вектор с магнитудой один, под перпендикуляром к поверхности. В случае с треугольными гранями, мы знаем, что любые три точки должны быть в одной плоскости, так что нормаль будет перпендикулярна этой плоскости, но откуда мы знаем, в каком направлении («вверх» или «вниз») эта нормаль будет указывать? Ещё раз отметим, что порядок индексов в данном случае критически важен. Грани полигональной сетки в Grasshopper определяются против часовой стрелки, так что грани с индексами {0,1,2} будут «перевёрнутыми» относительно граней с индексами {1,0,2}. Другой способ это визуализировать — это использовать *Правило-Правой-Руки*.



1 2 3

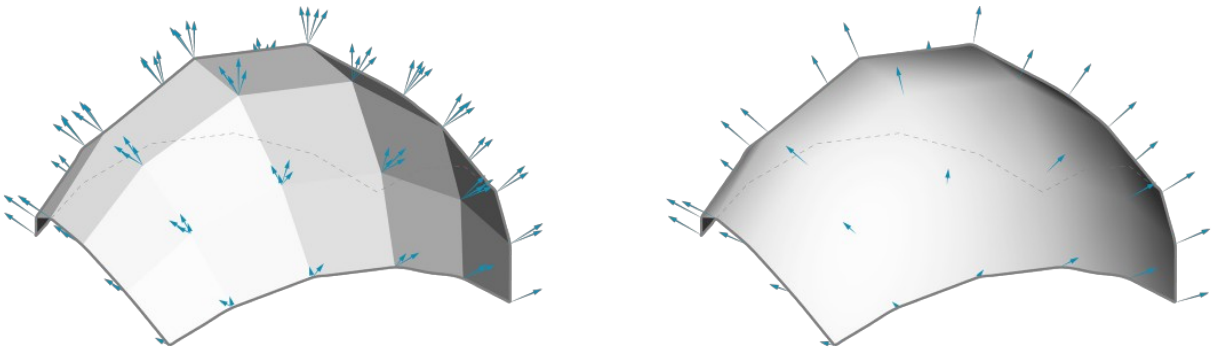
1. Компонент Face Normals (Нормали Граней) возвращает список точек центра и векторов нормалей каждой грани
2. Нормали граней определяются по последовательности вершин
3. "Правило-Правой-Руки", определяющее направление нормалей

Grasshopper также допускает использование четырёхугольных граней, но, в этом случае, все 4 точки не всегда будут оказываться в одной плоскости. Для этих граней точка центра просто будет посередине между координатами 4 вершин (в случае неплоской четырёхугольной грани обратите внимание, что эта точка не обязательно будет находиться на полигональной сетке). Чтобы вычислить нормаль четырёхугольной грани, мы вынуждены сначала триангулировать четырёхугольник, разделив его на два плоских треугольника. Нормалью будет среднее направление от двух нормалей, взвешенных (weighted) в соответствии с площадью двух треугольников.

Номали Вершин (Vertex Normals)

В дополнение к нормалям граней, можно также рассчитать нормали для каждой вершины полигональной сетки. Для вершины, использующейся только одной гранью, нормаль вершины будет указывать в том же направлении, что и нормаль грани. Если вершина принадлежит нескольким смежным граням, нормаль вершины будет вычисляться усреднением нормалей всех этих граней.

И, хотя, понятие нормалей граней может показаться менее ясным, чем нормали граней, нормали вершин являются важными элементами для сглаженной визуализации полигональных сеток. Вы можете заметить, что даже когда сетка состоит из плоских граней, такая сетка может отображаться гладкой и округлой при затенении в Rhino. Использование именно нормалей вершин позволяет добиться этой гладкой визуализации.



1 2

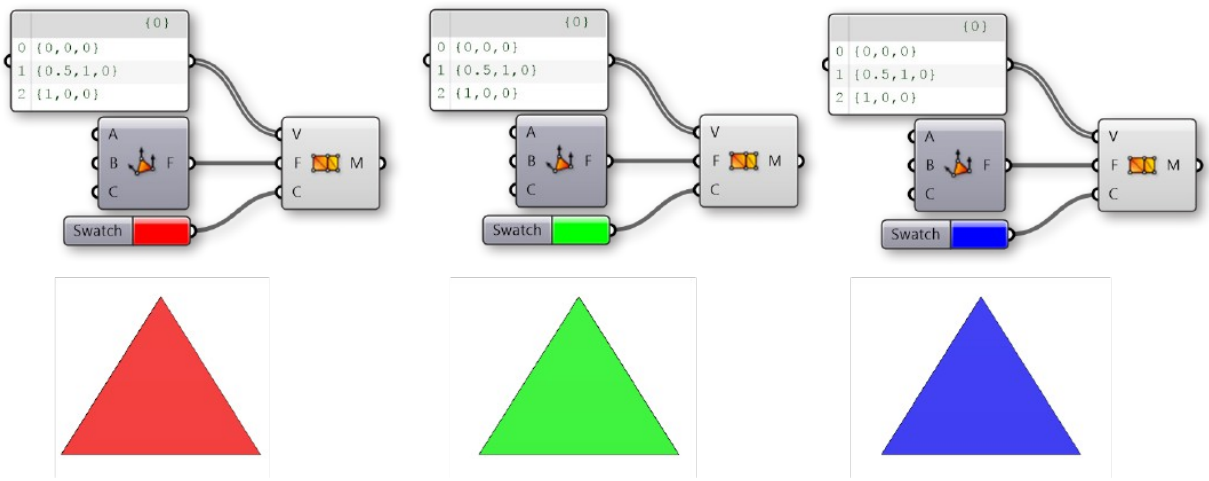
1. Нормали полигонального затенения установлены в соответствии с нормалями граней. В результате мы получаем дискретное полигональное затенение.
2. Номали вершин создаются усреднением нормалей граней. Результатом является сглаженной затенение поверх граней.

1.6.1.3 Атрибуты Полигональной сетки (Mesh Attributes)

Полигональным сеткам могут быть назначены дополнительные атрибуты: либо их вершинам, либо их граням. Простейшим из них является цвет вершины (vertex color), который описан ниже, но существуют и другие атрибуты, такие, например, как горизонтально-вертикальные координаты текстуры (texture UV coordinates). (Некоторые программы позволяют нормалям вершин быть назначенными в качестве атрибутов, вместо того, чтобы быть производными от граней и вершин, что позволяет достичь ещё большей гибкости при визуализации внешнего вида поверхности.)

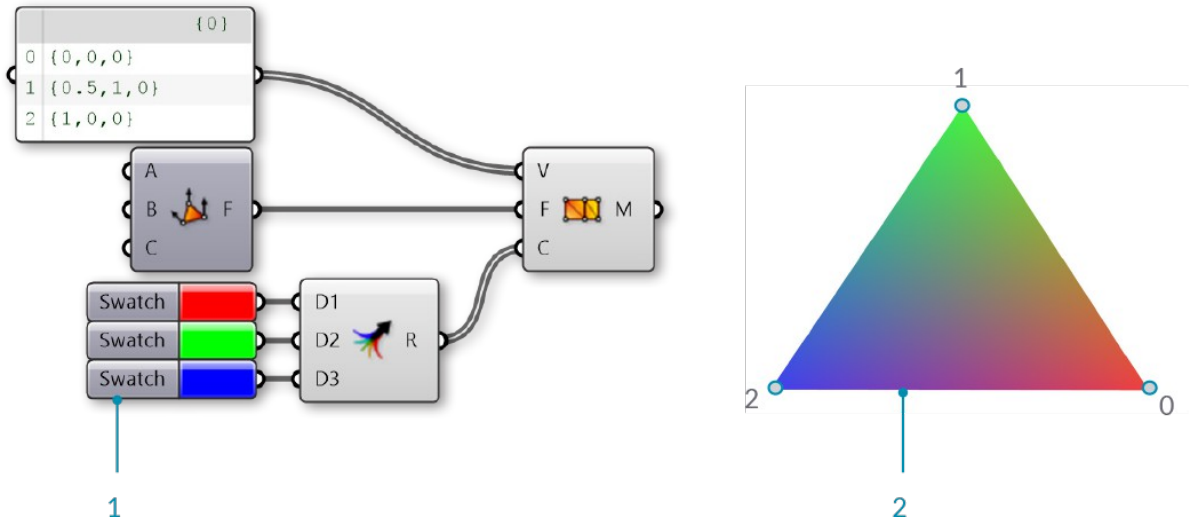
Цвет (Color)

При использовании компонента Construct Mesh (Скорнструировать Полигональную сетку) имеется вариант задействования специального входа для задания вершинам параметра цвета. Цвета также могут быть назначены существующей сетке, используя компонент Mesh Colours (Цвета Полигональной сетки). При использовании единственного параметра цвета для всей сетки, можно им окрасить всю сетку полностью.



Треугольный объект из полигональной сетки окрашен красным, зелёным либо синим

И, хотя выше приведён пример закрашивания всей сетки целиком, данные цвета, на самом деле, могут быть назначены индивидуально каждой вершине. Используя список из трёх цветов мы зададим отдельный цвет каждой вершине. Эти цвета будут использованы для визуализации, итерполируясь между цветами вершин. Например, на рисунке ниже показана треугольная грань с вершинами Красного (Red), Зелёного (Green) и Синего (Blue) цвета.

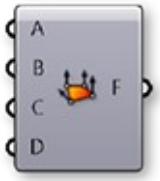


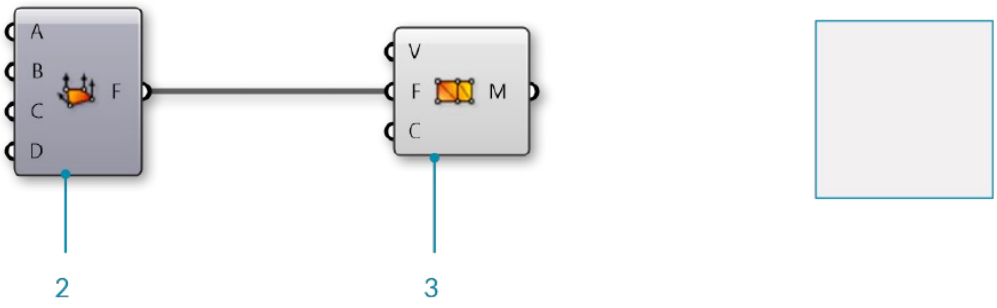
Красный, зелёный и синий назначенные трём вершинам полигональной сетки

Результирующая сеть интерполирует цвета вершин

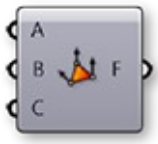
1.6.1.4 Упражнение

Скачать файлы примера, сопровождающие данный раздел.

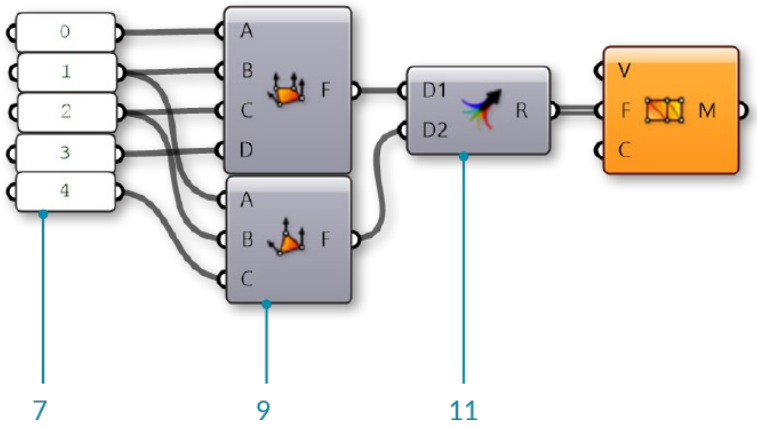
1.	Начните новый дефинишн, введя Ctrl-N (в Grasshopper)	
2.	**Mesh/Primitive/Mesh Quad (Полигональная сетка/Примитивы/Сетка с четырёхугольными гранями)** — Перетащите на холст компонент **Mesh Quad (Сетка с четырёхугольными гранями)**	
3.	**Mesh/Primitive/Construct Mesh (Полигональная сетка/Примитивы/Сконструировать Полигональную сетку)** — Перетащите на холст компонент **Construct Mesh (Сконструировать Полигональную сетку)**	
4.	Подсоедините выход Face (F) (Грань) компонента **Mesh Quad (Сетка с четырёхугольными гранями)** ко входу Faces (F) (Грани) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	



> Mesh Quad (Сетка с четырёхугольными гранями) и Construct Mesh (Сконструировать Полигональную сетку) по-умолчанию имеют настройку создания единственной грани полигональной сетки. Далее мы заменим значения, заданные по-умолчанию нашими собственными вершинами и гранями.

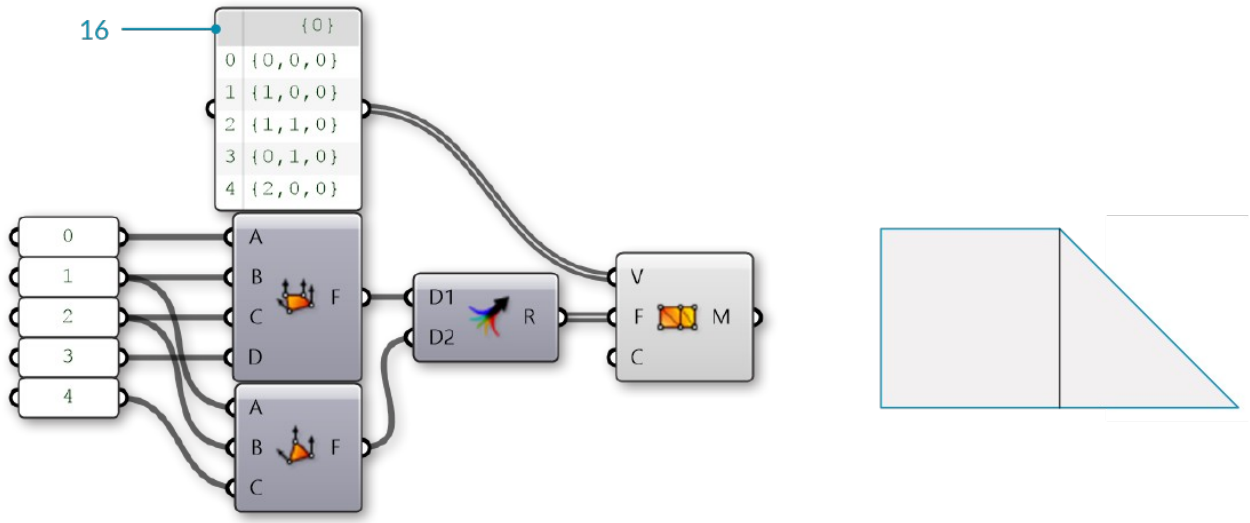
5.	**Params/Input/Panel (Параметры/Вводные/Текстовая панель)** — Перетащите на холст компонент **Panel (Текстовая панель)**	
6.	Дважды кликните на компоненте **Panel (Текстовая панель)** и задайте значение '0'	
7.	**Params/Input/Panel (Параметры/Вводные/Текстовая панель)** — Перетащите на холст ещё четыре компонента **Panel (Текстовая панель)** и задайте в них значения 1,2,3 и 4 > Вы также можете скопировать исходную Панель, кликнув по ней и при перетаскивании, перед тем, как отпустить, нажав клавишу Alt	
8.	Подсоедините **Панели** ко входам **Mesh Quad (Полигональная сетка с Четырёхугольными гранями)** в следующем порядке: 0 - A 1 - B 2 - C 3 - D	
9.	**Mesh/Primitive/Mesh Triangle (Полигональная сетка/Примитивы/Полигональная сетка с Треугольными гранями)** — Перетащите на холст компонент **Mesh Triangle (Полигональная сетка с Треугольными гранями)**	
10.	Подсоедините **Панели** ко входам **Mesh Triangle (Полигональная сетка с Трёхугольными гранями)** в следующем порядке: 1 - A 2 - B 4 - C	
11.	**Sets/Tree/Merge (Наборы/Дерево Данных/Слияние)** — Перетащите на холст компонент **Merge (Слияние)**	
12.	Подсоедините выход Face (F) (Грань) компонента **Mesh Quad (Полигональная сетка с Четырёхугольными гранями)** ко входу Data1 (D1) (Данные1) компонента **Merge (Слияние)** и выход Face (F) (Грань) компонента **Mesh Triangle (Полигональная сетка с Треугольными гранями)** ко входу Data2 (D2) (Данные2) компонента **Merge**	

	(Слияние)	
13.	Подсоедините выход Result (R) (Результат) компонента **Merge (Слияние)** ко входу Faces (F) (Грани) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	



По-умолчанию, список Vertices (V) (Вершины) содержит лишь 4 точки, но наш Mesh Triangle (Полигональная сетка с Треугольными гранями) использует индекс 4, который соответствовал бы пятой точке в списке. Так, как не хватает одной вершины, компонент Construct Mesh (Сконструировать Полигональную сетку) выдаёт ошибку. Чтобы исправить это, мы предоставим наш собственный список точек.

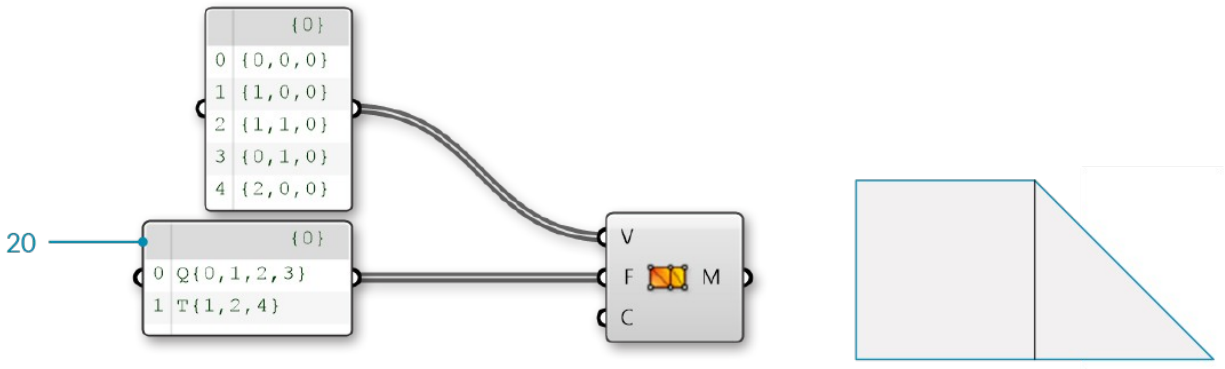
14.	**Params/Input/Panel (Параметры/Вводные/Текстовая панель)** — Перераща́ите на холст компонент **Panel (Текстовая панель)**	
15.	Кликните правой кнопкой по **Панели** и снимите выделение с опции 'Multiline Data' (Многострочные Данные) По-умолчанию, Панель имеет 'Multiline Data' (Многострочные Данные) включенной. Отключив эту опцию, каждая строка в панели будет читаться как отдельный элемент списка.	
16.	Дважды кликните по компоненту **Panel (Текстовая Панель)**, чтобы отредактировать его и введите следующие точки: {0,0,0} {1,0,0} {1,1,0} {0,1,0} {2,0,0} Убедитесь, что Вы используете правильные обозначения. Для того, чтобы определить точку в Панели, Вы должны использовать фигурные скобки: '{' и '}', разделяя запятыми значения x, y, и z.	
17.	Подсоедините компонент **Panel** ко входу Vertices (V) (Вершины) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	



Теперь мы получили полигональную сетку с двумя гранями и пятью вершинами

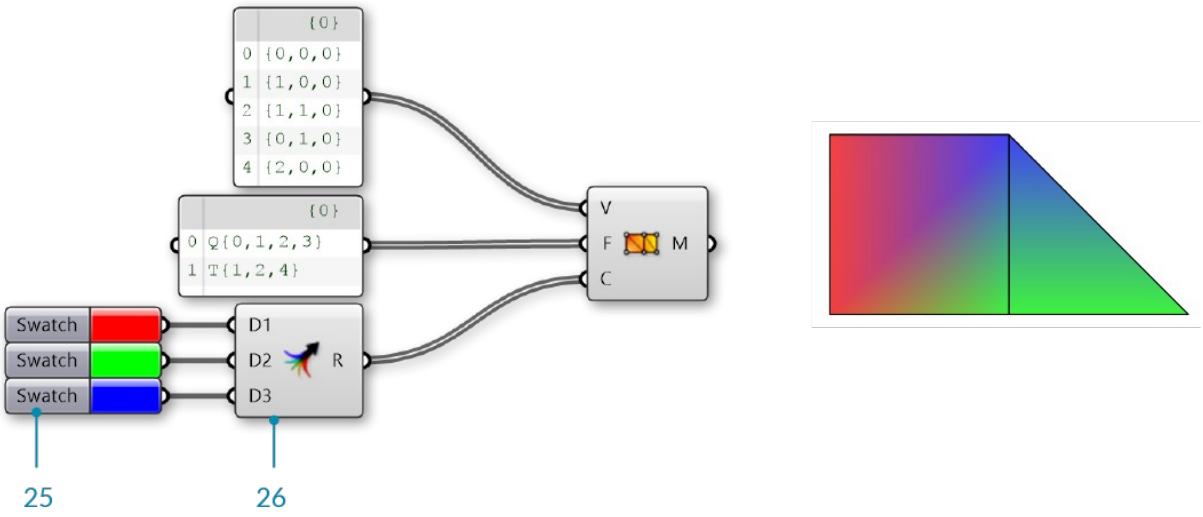
Опционально, мы можем заменить компоненты **Mesh Quad (Полигональная сетка с Треугольными гранями)** и **Mesh Triangle (Полигональная сетка с Треугольными гранями)** на компонент **Panel (Текстовая панель)**, определяющий индексы граней.

18.	**Params/Input/Panel (Параметры/Вводные/Текстовая панель)** — Перетащите на холст компонент **Panel (Текстовая панель)**	
19.	Кликните правой кнопкой по **Панели** и снимите выделение с опции 'Multiline Data' (Многострочные Данные) Кроме того, скопируйте описание точек из соответствующей, ранее используемой, **Панели**, той самой, в которой мы уже отключали опцию 'Multiline Data' (Многострочные данные)	
20.	Дважды кликните по компоненту **Panel (Текстовая панель)** для его редактирования и введите следующее: Q{0,1,2,3} T{1,2,4}	
21.	Подсоедините **Панель** ко входу Faces (F) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	



22.	**Params/Input/Colour Swatch (Образец Цвета)** — Перетащите на холст компонент **Colour Swatch (Образец Цвета)**	
23.	Кликните по цветному участку компонента (по-умолчанию, Белый (White)), чтобы открыть панель выбора цвета	

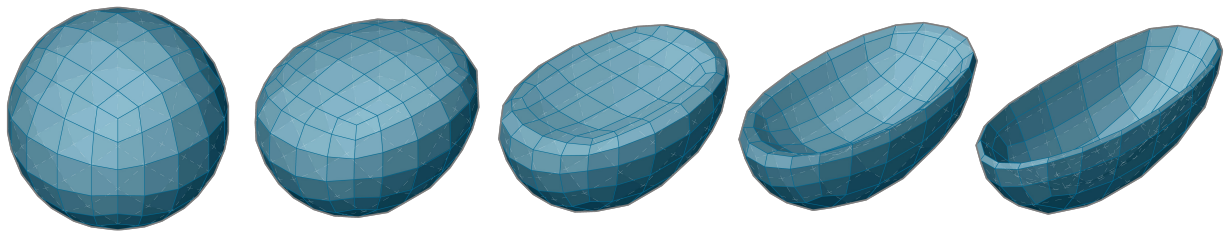
24.	Используйте слайдеры, чтобы установить значения G и B на ноль. Теперь образец должен стать Красным (Red)	
25.	**Params/Input/Colour Swatch (Образец Цвета)** — Перетащите на холст ещё два компонента **Colour Swatch (Образец Цвета)** и установите их цвета на Синий (Blue) и Зелёный (Green)	
26.	**Sets/Tree/Merge (Наборы/Дерево Данных/Слияние)** — Переращите на холст компонент **Merge (Слияние)**	
27.	Подсоедините три компонента **Color Swatch (Образец Цвета)** ко входам D1, D2 и D3 компонента **Merge (Слияние).**	
28.	Подсоедините выход Result (R) (Результат) компнента **Merge (Слияние)** ко входу Colours (C) (Цвета) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	



Мы имеем 5 вершин, но только 3 цвета. Grasshopper будет назначать цвета повторяющимся узором (паттерном). Так, в данном случае, вершины 0 и 3 будут красными, вершины 1 и 4 — зелёными, а финальная вершина 2 будет синей.

1.6.2 Понятие «Топология» (Topology)

Кроме того, что вершины полигональной сетки содержат информацию о своём местоположении, они помнят топологию, то есть конфигурацию связей между вершинами, которая и придаёт геометрии её уникальную структуру и гибкость.

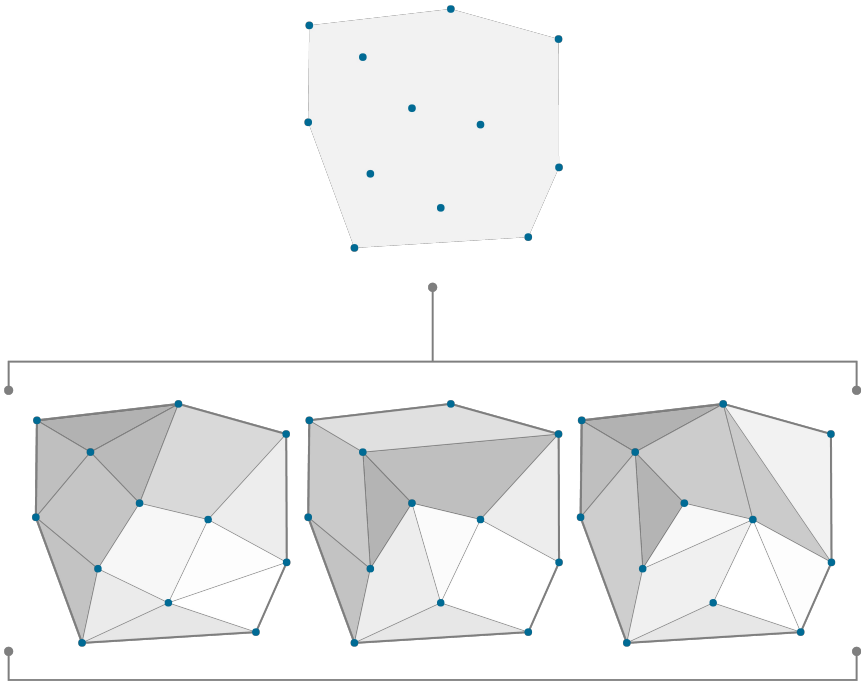


1.6.2.1 Что такое топология?

Любое введение в геометрию полигональной сетки было бы неполным без введения в основы топологии. Поскольку топология имеет дело с взаимосвязями и свойствами набора «вещей», а не «вещи» сами по себе. Это придаёт гибкость огромному диапазону как материальных, так и абстрактных применений. В этом учебнике, наш интерес лежит в сфере основного применения параметрических рабочих процессов, дающих возможность создавать и контролировать геометрию полигональной сетки.

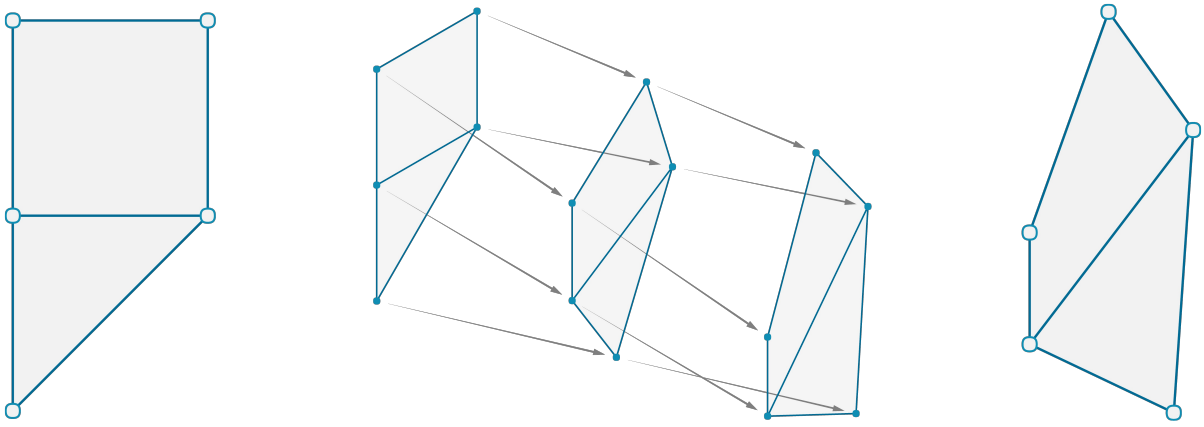
В Grasshopper есть два основных типа информации, необходимой для определения полигональной сетки — геометрия и соединения, другими словами, это набор точек в rhino-пространстве (вершины) и набор соответствующих связей-между-точками (грани).

Без информации о связях, полигональная сетка является неструктурированной, а потому и неопределённой. Введение набора граней — это шаг (или даже прыжок), который, в конечном счёте, актуализирует сетку и задаёт её характер с точки зрения непрерывности, сходимости и связности. Эта сетевая структура и называется *топологией* пространства.



Один и тот же набор вершин может иметь различные варианты соединения, что приводит к различной топологии.

Топологическое отображение или гомеоморфизм (Homeomorphism)



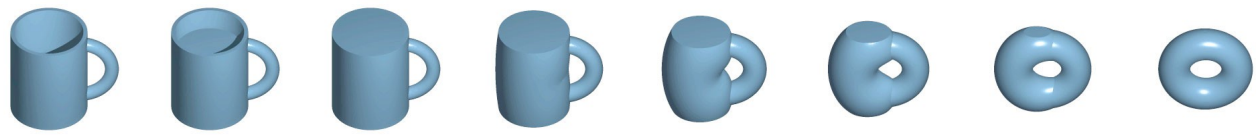
Точки полигональной сетки могут быть перемещены без изменения информации о соединениях. Новая полигональная сетка имеет ту же самую топологию, что и оригинал.

Да, это возможно. Одинаковая топология для двух полигональных сеток, различных по форме. Однако, это лишь означает, что они сконструированы из одинакового числа точек и структурированы одним и тем же набором граней. Ранее мы установили, что грань полигональной сетки имеет отношения только с индексами

набора точек и не имеет никакого интереса, насчёт её фактического местоположения в rhino-пространстве. Поэтому, если единственное различие между двумя полигональными сетками в их форме, определяемое местоположением их точек в 3D-пространстве, то обе эти полигональные сетки считаются «гомеоморфными» (или топологически эквивалентными), и, следовательно, имеющими одни и те же топологические свойства.

{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 гомеоморфизма среди букв (обратите внимание, что некоторые из вышеупомянутых групп могут считаться различными в зависимости от используемого шрифта)

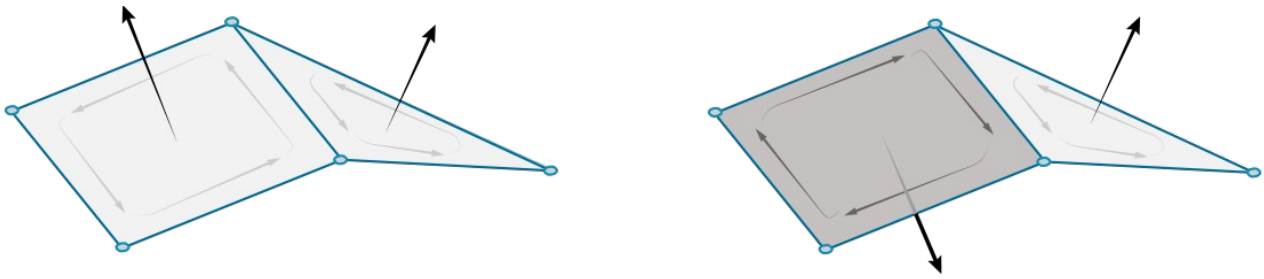


Топологически эквивалентные кружка и пончик

1.6.2.2 Характеристика Полигональной сетки (Mesh Characteristics)

Ориентированность (Orientable)

Полигональная сетка считается ориентированной, если у неё есть чётко сориентированные стороны. Простым примером неориентированной сетки могут послужить соседние грани, направленные в противоположные стороны. Эти «перевёрнутые» грани могут вызвать проблемы при визуализации и производстве 3D-печати.



1 2

Ориентированная поверхность, с нормалями граней, указывающих в одном направлении.

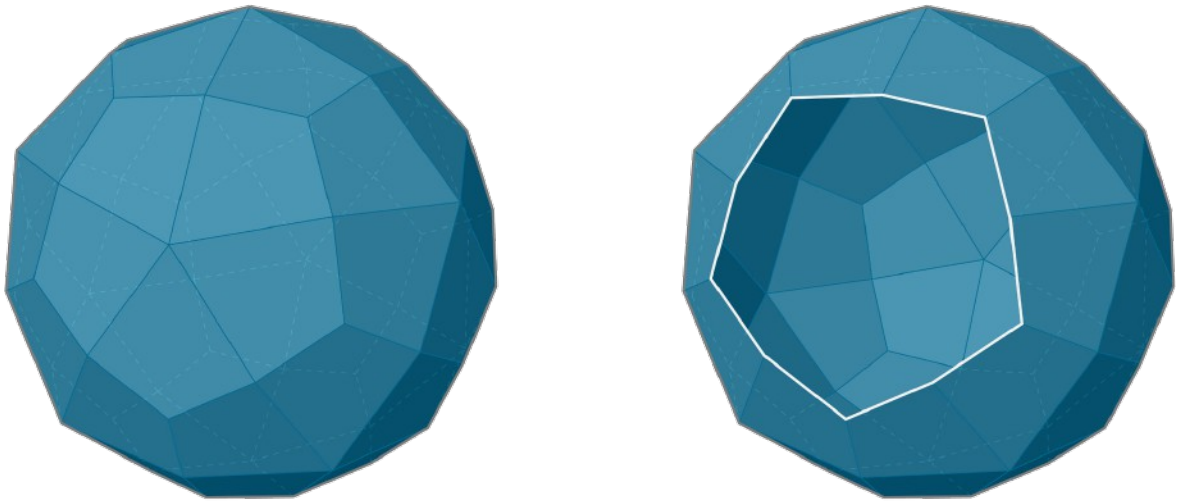
Неориентированная поверхность имеет соседние грани с нормалями, направленными в разные стороны.

Отрытые против Замкнутых (Open vs Closed)

Часто бывает необходимо знать, является ли полигональная сетка замкнутой, представляющая собой замкнутый объём (твёрдое тело (solid)), или эта полигональная сетка — открытая (open), представляющая 2-мерную поверхность. Разница может оказывать существенное влияние на технологичность. Вы не можете осуществить 3D-печать единой поверхности, не имеющей толщины, для этого Вы обязательно должны добавить толщины стенкам, и геометрия при этом станет твёрдотельной (solid). Твердотельная полигональная геометрия (Solid mesh geometry) также требуется для осуществления Булевых (Boolean) операций (рассматриваемых в следующем разделе).

В помощь определения замкнутости может быть использован компонент Mesh Edges (Края Полигональной сетки). Если ни один из краёв не имеет валентность1 (если результат на выходе E1 равен нулю), то мы знаем, что все края - 'Interior Edges' (Внутренние Края) и полигональная сетка не имеет внешней границы края, и, следовательно, является замкнутой.

С другой стороны, если присутствуют 'Naked Edges' (Открытые Края), то эти края должны быть на границе полигональной сетки и сетка незамкнута.



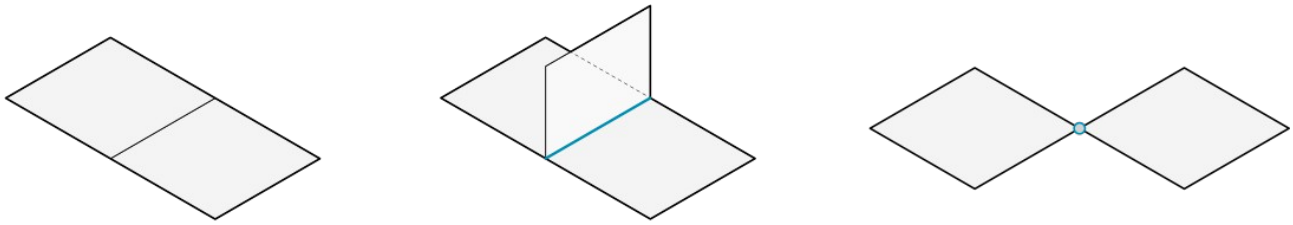
1 2

1. Замкнутая полигональная сетка. Ко всем краям примыкают строго две грани.
2. Открытая полигональная сетка. К белым краям примыкает лишь по одной грани.

Немногосложные против Многосложных (Manifold vs Non-Manifold)

Многосложная (Non-manifold) геометрия — это геометрия, которая, по существу, не может существовать в «реальном мире». Это не обязательно делает её «плохой геометрией», но понять данное состояние геометрии важно из-за возможных осложнений, которые она может представлять для инструментов и операций

(например, рендеринг эффектов рефракции, имитация жидкостей, булевы операции, 3D-печать и т.д.). Общие условия, которые приводят к появлению многосложных сеток включают: самопересечения (self intersection), открытые края (naked edges) (отверстий, либо внутренних граней), расчленённая топология (disjoint topology) и перекрытие/дублирование граней (overlapping/duplicate faces). Полигональная сетка также может считаться Многосложной (Non-Manifold), если индексы любых её вершин, являющихся общими для краёв, или любые края с валентностью большей, чем 2 создают соединение по крайней мере трёх граней.



1 2 3

1. Простая немногосложная (manifold) полигональная сетка
2. Три грани, соединяющиеся на одном краю не являются многосложными (non-manifold). Такой метод называется также T-образным стыком (T-Junction)
3. Две грани соединяются только одной вершиной, но не используют совместно края. Это — многосложная (non-manifold) геометрия

1.6.2.3 Полигональные сетки против НУРБС (Meshes Vs NURBS)

Как полигональная (mesh) геометрия отличается от сплайновой (NURBS) геометрии? В каких случаях Вы могли бы использовать одну вместо другой?

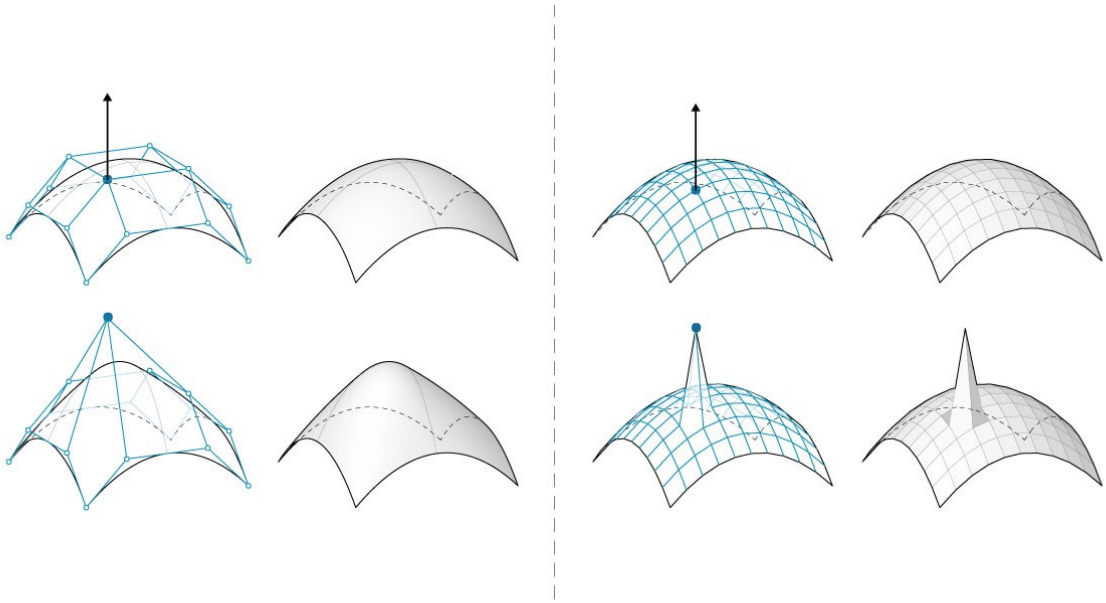
В предыдущей главе мы увидели, что NURBS-поверхности определяются по серии NURBS-кривых, взаимодействующих в двух направлениях. Эти направления помечаются, как U (Горизонталь) и V (Вертикаль) и позволяют NURBs-поверхности быть параметризованной в соответствии с двумерным диапазоном поверхности (surface domain). Сами кривые хранятся в компьютере в виде уравнений, что позволяет результирующую поверхность рассчитывать с высокой степенью точности. Однако, это может усложнить комбинирование нескольких NURBS-поверхностей воедино. Объединение (Join) двух NURBS-поверхностей создаст составную поверхность (polysurface), у которой различные секции геометрии будут иметь различные UV-параметры и определения кривых (curve definitions).

С другой стороны, Полигональные сетки (Mesh) состоят из дискретного числа точно определённых вершин и граней. Сеть вершин, как правило, не может быть определена с помощью простых UV-координат, и, поскольку грани определяются дискретными величинами точности, встроенными в полигональную сетку, то могут быть изменены только путём уточнения полигональной сетки и добавлением дополнительного количества вершин. Однако, отсутствие UV-координат, даёт полигональной сетке большую гибкость в обработке более сложной геометрии в составе единой сети, не прибегая к созданию составных поверхностей, в отличие от NURBS.

Примечание — Хотя полигональные сетки и не имеют явной изначальной UV-параметризации, иногда очень полезно назначить им такую параметризацию, чтобы было возможно наложить текстурную карту или файл изображения на полигональную геометрию для визуализации. Некоторое программное обеспечение для моделинга обрабатывает UV-координаты вершин полигональной сетки в качестве атрибута (подобно цвету вершин), который может быть изменён с помощью различных манипуляций. Обычно они назначаются и не определяются полностью самой полигональной сеткой.

Влияние. Локальное против Глобального. (Local vs Global Influence)

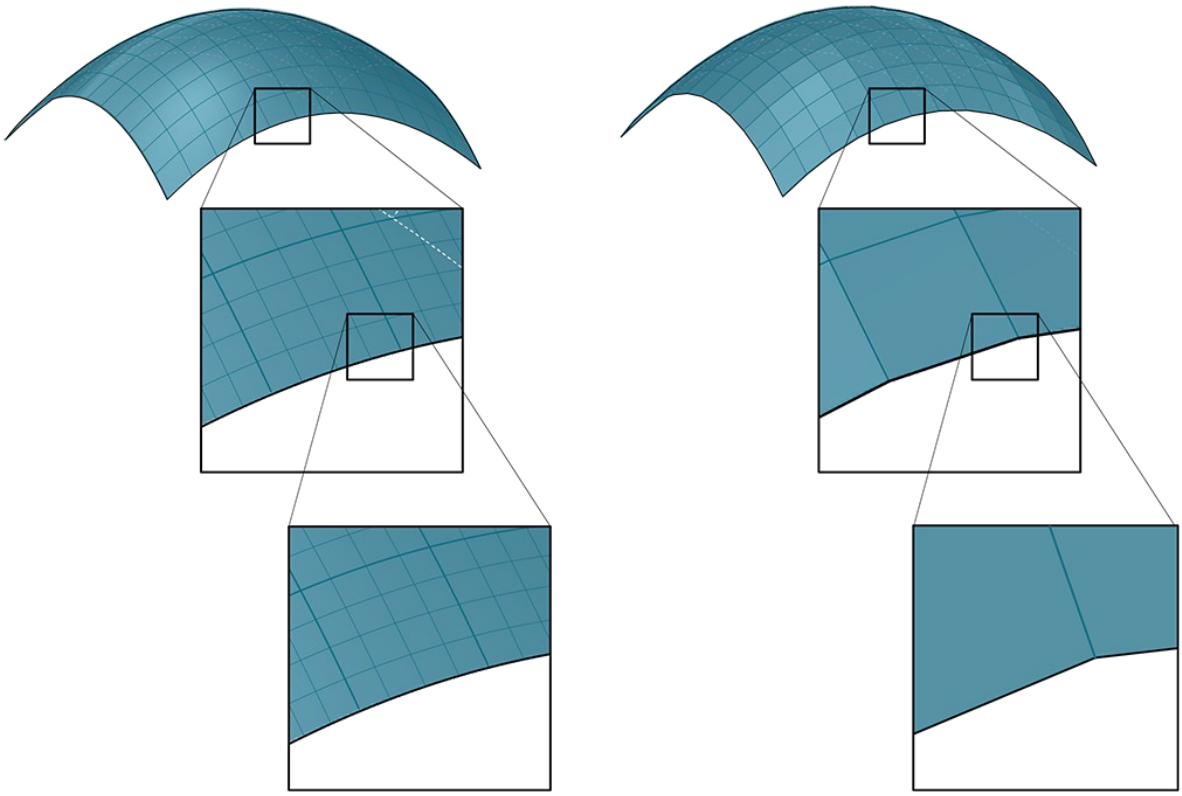
Другим важным отличием является степень, в которой локальное изменение полигональной сетки или NURBS-геометрии влияет на всю форму. Полигональная геометрия является полностью локальной. Перемещение одной вершины влияет только на грани, смежные с данной вершиной. В NURBS-поверхностях взаимодействия влияния более сложные и зависят от степени (degree) поверхности (surface), а также от веса (weight) узлов (knots) и контрольных точек (control points). Однако, в целом, перемещение одной контрольной точки в NURBS-поверхности влечёт за собой более глобальные изменения в геометрии.



1. NURBS-поверхность — перемещение контрольной точки имеет более глобальное влияние
2. Полигональная геометрия — перемещение вершины имеет локальное влияние

Одна из аналогий, которая может быть полезна, это сравнение векторного изображения (состоящего из прямых и кривых линий) и растрового изображения (состоящего из отдельных пикселей). Если Вы увеличите векторное изображение, кривые останутся такими же чёткими и ясными, в то время, как результатом

увеличения растрового изображения станет лишь наблюдение отдельных пикселей. В этой аналогии NURBS-поверхности будут сравнимы с векторным изображением, а полигональная сетка будет вести себя подобно растровому изображению.



Увеличение NURBS-поверхности сохранит гладкость, подобно кривой, в то время, как полигональная сетка имеет фиксированное разрешение

Интересно отметить, что, хотя NURBs-поверхности и хранятся в виде математических уравнений, в действительности, для их визуализации тоже требуются полигональные сетки. Пока для компьютера невозможная задача — отображать непрерывное уравнение. Вместо этого, требуется разорвать график уравнения на части, результат которых можно будет рендерить, либо отображать сконвертированными из NURBS в полигональные сетки. Это аналогично тому, что, даже при том, что мы можем хранить на компьютере линию в виде уравнения функции, при отображении её, компьютер должен в некий момент времени преобразовать эту линию в ряд дискретных пикселей на экране.

1.6.2.4 Плюсы и Минусы Полигональных сеток

Когда мы спрашиваем: «Каковы За и Против моделирования полигональными сетками?», на самом деле мы спрашиваем: «Каковы За и Против моделирования формами, определёнными исключительно набором вершин и соответствующей структурой топологии?» С помощью такого метода преобразования вопроса легче

увидеть, что «упрощённая» природа полигональной сетки — критический аспект, который сделал бы полигональную сетку подходящей или нет, в зависимости от контекста её применения.
Полигональные сетки могут быть оптимальными для применения в следующих случаях:

- Там, где необходимо динамическое обновление визуализации формы, постоянно меняющей свою конфигурацию, но не в плане соединения граней
- Там, где было бы достаточно дискретного приближенного отображения геометрии кривой;
- Там, где низко-полигональная геометрия должна систематически сглаживаться, или сочетаться с использованием вычислительных методов повышения разрешения модели;
- Там где модель с низким разрешением должна использоваться для совместной локальной поддержки детали высокого разрешения.

Полигональные сетки будут неподходящими в ситуациях когда:

- Кривизна и сглаженность должны быть представлены с высокой точностью
- Должна быть оценена истинная производная функции
- Геометрия должна быть конвертирована в твёрдое тело для производства
- Конечная форма должна быть способна легко редактироваться вручную

1.6.3 Создание Полигональных сеток

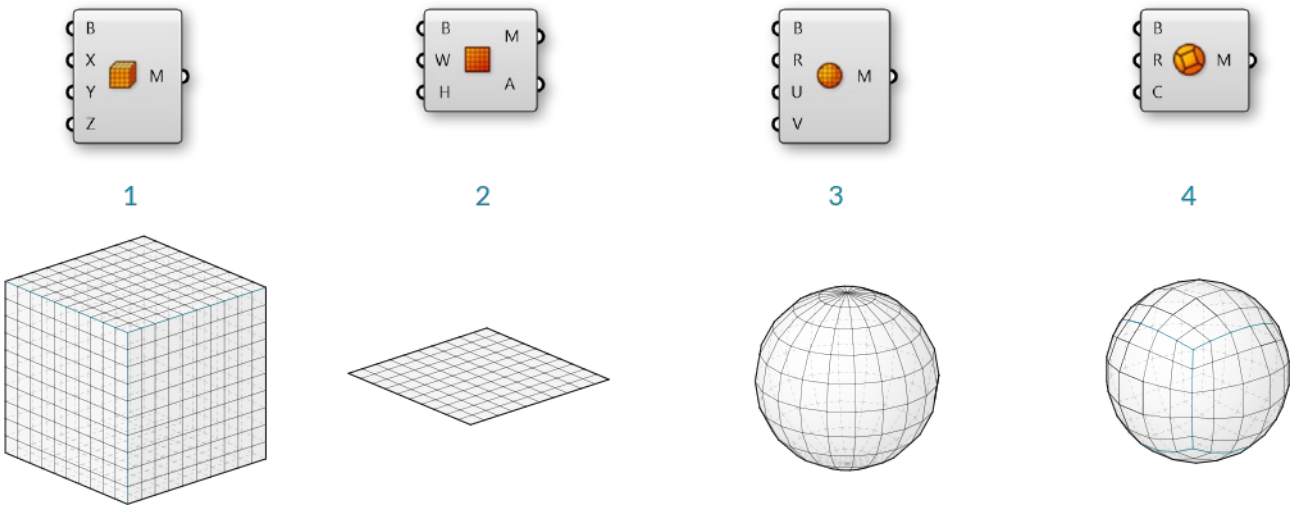
В последнем разделе мы рассмотрели базовые структуры полигональных сеток. В этом разделе, мы предоставим краткое введение в различные способы генерации полигональной геометрии.

В Grasshopper есть три основных способа создания полигональной геометрии:

1. Начинать с полигонально-сеточных примитивов
2. Построение полигональных сеток по полигонам и вершинам вручную
3. Конвертация NURBS-геометрии в полигональные сетки

1.6.3.1 Примитивы (Primitive)

Grasshopper поставляется с несколькими компонентами для создания простых примитивов полигональной сетки:



Mesh Box (Параллелепипед из Полигональной сетки) — Этот примитив требует на входе объект Box (Параллелепипед), который задаёт размер и местоположение, а сам модуль предоставляет возможность задать количество граней, на которые будет разбит конечный параллелепипед по осям X,Y и Z . Шесть сторон Параллелепипеда из Полигональной сетки не свариваются вместе (unwelded) для того, чтобы обеспечить чёткие грани (складки (creases)). (Для получения более детальной информации о сваривании полигональных сеток (welded meshes) смотрите следующий раздел).

Mesh Plane (Плоскость из Полигональной сетки) — Этот примитив требует ввода Прямоугольника (Rectangle) для определения размера и местоположения плоскости (plane), а также значений (W и H), чтобы определить количество граней.

Mesh Sphere (Сфера из Полигональной сетки) — Этот примитив требует базовой плоскости, чтобы определить центр и ориентацию сферы, радиус, чтобы задать размер, а также значения U и V, задающие количество граней.

Mesh Sphere Ex (Сфера из Полигональной сетки Бесполюсная) — Также известный, как Quadball (Квадбол), этот примитив создаёт сферу, скомбинированную из шести лоскутов (patches), которые подразделяются в соответствии с количеством, заведённым через вход C. Топологически квадбол эквивалентен кубу, хотя геометрически он сферический.

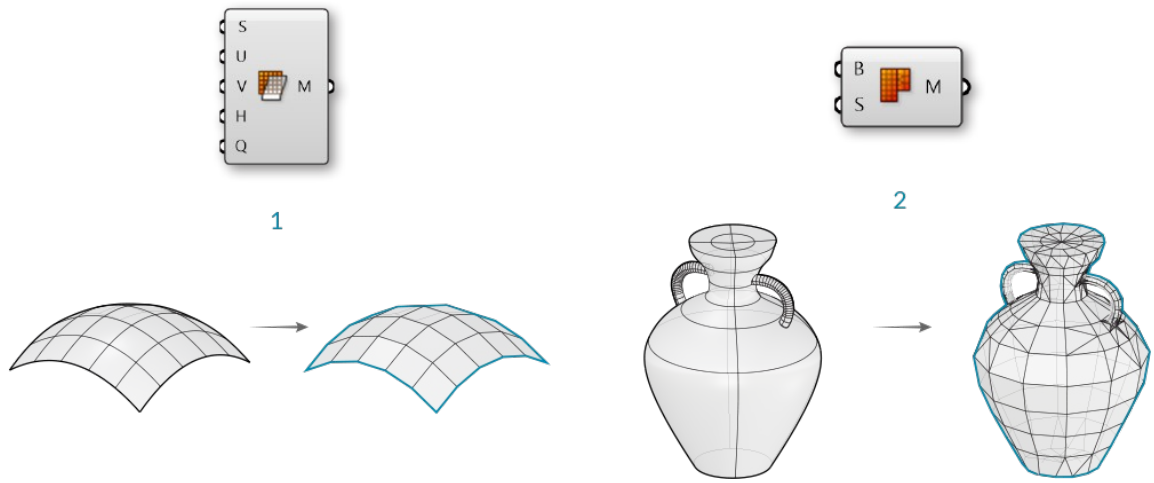
1.6.3.2 Construct Mesh (Сконструировать Полигональную сетку)

Как мы увидели в предыдущем разделе, компонент **Construct Mesh (Сконструировать Полигональную Сетку)** может быть использован для создания полигональной сетки непосредственно из списка вершин и списка граней (и опционально списка цвета вершин). Построение всей сетки вручную может оказаться чрезвычайно утомительным, так что этот компонент чаще используется совместно с уже существующим списком граней и вершин, которые предварительно были извлечены из уже существующей сетки с помощью компонента **Deconstruct Mesh (Разобрать Полигональную сетку)**.

1.6.3.3 Из NURBS в Mesh (Полигональную сетку)

Пожалуй, самым распространённым для создания сложной полигональной сетки является метод генерирования её из базовой NURBS-геометрии. Отдельные NURBS-поверхности могут быть преобразованы в полигональную сетку с использованием компонента **Mesh Surface (Полигональная сетка из NURBS-Поверхности)**, который просто делит поверхность вдоль её UV-координат и создаёт из этого четырёхугольные грани. Этот компонент позволяет указать требуемое количество подразделений по U (Горизонтали) и V (Вертикали) в результирующей полигональной сетке.

Более сложные составные поверхности (polysurface) могут быть преобразованы в единую полигональную сетку с помощью компонента **Mesh Brep (Brep Полигональной Сетки)**. Этот компонент также имеет опциональный вход Settings (Настройки), через который можно задать, используя встроенные средства настройки необходимые установки, в виде таких компонентов как *Speed (Скорость), Quality (Качество),* или *Custom Settings (Пользовательские настройки)*, либо, кликнув правой кнопкой мыши по входу S и выбрав "Set Mesh Options" (Задать Опции Полигональной сетки). Для эффективного использования полигональной сетки часто бывает необходимо усовершенствовать эту полигональную сетку, используя различные стратегии, такие как rebuilding (реконструкция), smoothing (сглаживание) или subdividing (подразделение). Некоторые из этих техник будут обсуждаться позже в этом Учебнике.



1. **Mesh Surface (Полигональная сетка из NURBS-Поверхности)** конвертирует NURBS-поверхность в полигональную сетку
2. **Mesh Brep (Brep Полигональной Сетки)** может конвертировать составные поверхности и более сложную геометрию в единую полигональную сетку. Настройки позволяют задать большее или меньшее количество граней и мелкую или грубую полигональную сетку.

ПРИМЕЧАНИЕ: как правило, гораздо легче преобразовать NURBS-геометрию в полигональную сетку, но не наоборот. В то время, как UV (Горизонтально-Вертикальные) координаты NURBS-поверхности просты для преобразования в четырёхугольные грани полигональной сетки, обратное не всегда верно, так как сетка может может содержать комбинацию треугольников и четырёхугольников в одном наборе, из которого совсем непросто извлечь систему UV-координат.

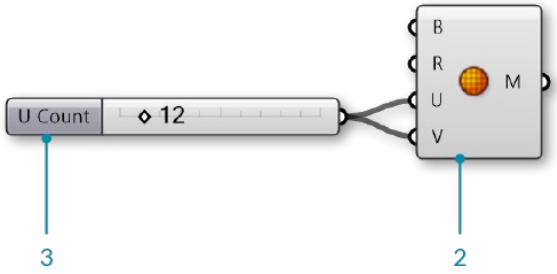
1.6.3.4 Упражнение

В этом упражнении мы будем использовать базовые примитивы Полигональной сетки (basic Mesh primitive), выполним трансформацию вершин, а затем назначим цвет, основываясь на нормалях векторов, приблизительно, как это происходит в процессе рендеринга.

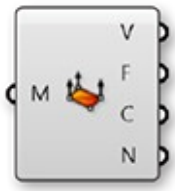
Скачать файлы примера, сопровождающие данный раздел

1.	Начните новый дефинишин, введя Ctrl-N (в Grasshopper)	
2.	**Mesh/Primitive/Mesh Sphere (Полигональная сетка/Примитивы/Сфера из Полигональной сетки)** — Перетащите на холст компонент **Mesh Sphere (Сфера из Полигональной сетки)**	
3.	**Params/Input/Number Slider (Параметры/Вводные/Числовой слайдер) —** Переращите	

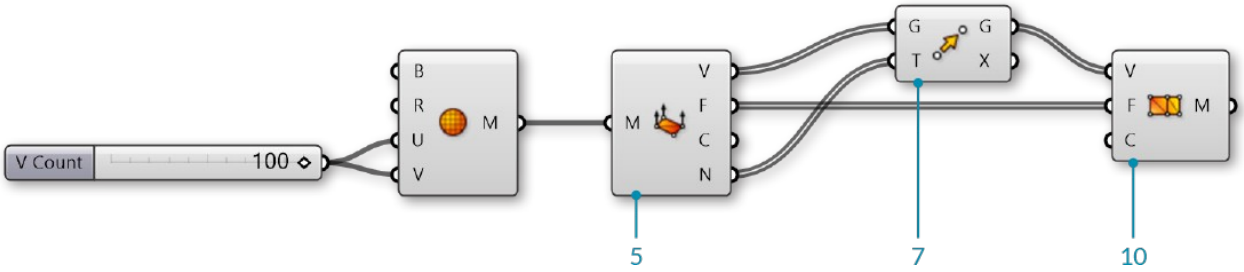
	на холст компонент **Number Slider (Числовой слайдер)** и задайте следующие значения: Rounding: Integer (Округление: Целые числа) Lower Limit:0 (Нижний Предел) Upper Limit: 100 (Верхний Предел) Value: 10 (Значение)	
4.	Подсоедините **Number Slider (Числовой Слайдер)** ко входам U Count (U) (Счёт по Горизонтали) и V Count (V) (Счёт по Вертикали) компонента **Mesh Sphere (Сфера из Полигональной сетки)**	



Откорректируйте слайдер и обратите внимание, как будет изменяться разрешение сферы во вьюпорте Rhino. Более высокие значения приводят к повышению гладкости сферы, но также и увеличению объёмов данных, которые могут потребовать больше времени для обработки.

5.	**Mesh/Analysis/Deconstruct Mesh (Полигональная сетка/Анализ/Разобрать Полигональную сетку)** — Перетащите на холст компонент **Deconstruct Mesh (Разобрать Полигональную сетку)**	
6.	Подсоедините выход Mesh (M) (Полигональная сетка) компонента **Mesh Sphere (Сфера из Полигональной сетки)** ко входу Mesh (M) (Полигональная сетка) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)**	
7.	**Transform/Euclidean/Move (Трансформация/ Евклидова/Переместить)** — Перетащите на холст компонент **Move (Перемещение)**	
8.	Подсоедините выход Vertices (V) (Вершины) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу Geometry (G) (Геометрия) компонента **Move (Перемещение)**	
9.	Подсоедините выход Normals (N) (Нормали) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу Motion (T) (Движение) компонента **Move**	

	(Перемещение)	
10.	**Mesh/Analysis/Construct Mesh (Полигональная сетка/Анализ/Сконструировать Полигональную сетку)** — Перетащите на холст компонент **Construct Mesh (Сконструировать Полигональную сетку)**	
11.	Подсоедините выход Geometry (G) (Геометрия) компонента **Move (Перемещение)** ко входу Vertices (V) (Вершины) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	
12.	Подсоедините выход Faces (F) (Грани) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу Faces (F) (Грани) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	



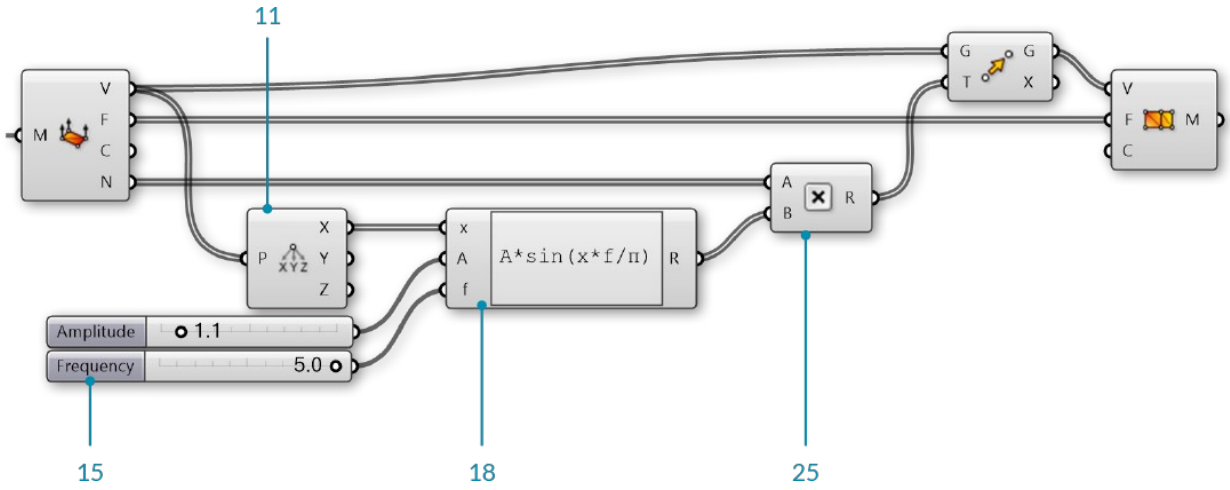
Мы разобрали полигональную сетку, чтобы извлечь списки её вершин, граней и нормалей. Затем мы просто перемещаем каждую вершину в соответствии с её вектором нормали. Поскольку мы не изменяли общую топологию сферы, мы можем использовать снова список граней для повторного построения новой полигональной сетки. Нормали векторов всегда имеют длину единица, поэтому реконструкция новой полигональной сетки заканчивается с радиусом на единицу больше, чем у исходной сферы.

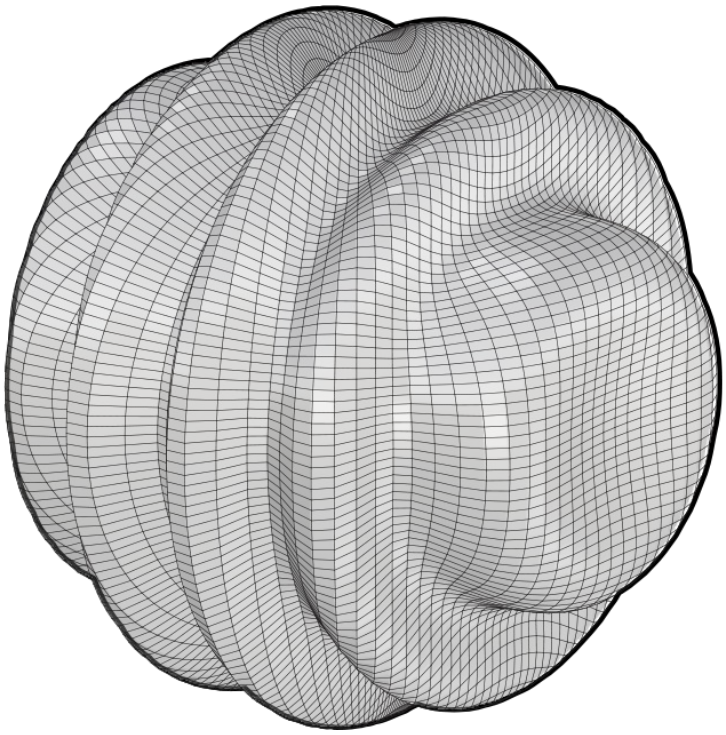
Далее мы будем использовать функцию синуса, чтобы манипулировать сферой более сложным образом.

13.	**Vector/Point/Deconstruct (Вектор/Точка/Разобрать)** — Перетащите на холст компонент **Deconstruct (Разобрать)**	
14.	Подсоедините выход Vertices (V) (Вершины) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу Point (P) (Точка) компонента **Deconstruct (Разобрать)**	
15.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст два компонента **Number Slider (Числовой Слайдер)**	

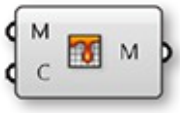
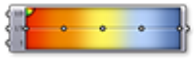
16.	Задайте на первом **Числовом Слайдере** следущие значения: Name: Amplitude (Имя: Амплитуда) Rounding: Float (Округление: Число с плавающей запятой) Lower Limit: 0 (Нижний Предел) Upper Limit: 10 (Верхний Предел)	
17.	Задайте на втором **Числовом Слайдере** следущие значения: Name: Frequency (Имя: Частота) Rounding: Float (Округление: Число с плавающей запятой) Lower Limit: 0 (Нижний Предел) Upper Limit: 5 (Верхний Предел)	
18.	**Maths/Script/Expression (Математика/Скрипт/Выражение)** — Перетащите на холст компонент **Expression (Выражение)**	
19.	Приблизьте компонент **Expression (Выражение)**, чтобы увидеть опции для добавления или удаления входов переменных и кликните по значку «+», чтобы добавить переменную «z»	
20.	Кликните правой кнопкой мыши по входу «Y» компонента **Expression (Выражение)** и измените текст на «A»	
21.	Кликните правой кнопкой мыши по входу «Z» компонента **Expression (Выражение)** и измените текст на «f»	
22.	Дважды кликните по компонету **Expression (Выражение)**, чтобы отредактировать его и введите следущее: A*sin(x*f/π)	
23.	Подсоедините выход X компонента **Deconstruct (Разобрать)** ко входу «X» компонента **Expression (Выражение)**	
24.	Подсоедините **Числовой Слайдер** Amplitude (Амлитуда) ко входу A и **Числовой Слайдер** Frequency (Частота) ко входу «f» компонента **Expression (Выражение)**	
25.	**Maths/Operators/Multiplication (Математика/Операторы/Умножение)** — Перетащите на холст компонент **Multiplication (Умножение)**	
26.	Подсоедините выход Normals (N) (Нормали) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу A компонента **Multiplication (Умножение)**	
27.	Подсоедините выход Result (R) (Результат) компонента **Expression (Выражение)** ко входу B компонента **Multiplication (Умножение)**	
28.	Подсоедините выход Result (R) (Результат) компонента	

	Multiplication (Умножение) ко входу Motion (T) (Движение) компонента **Move (Перемещение)**	

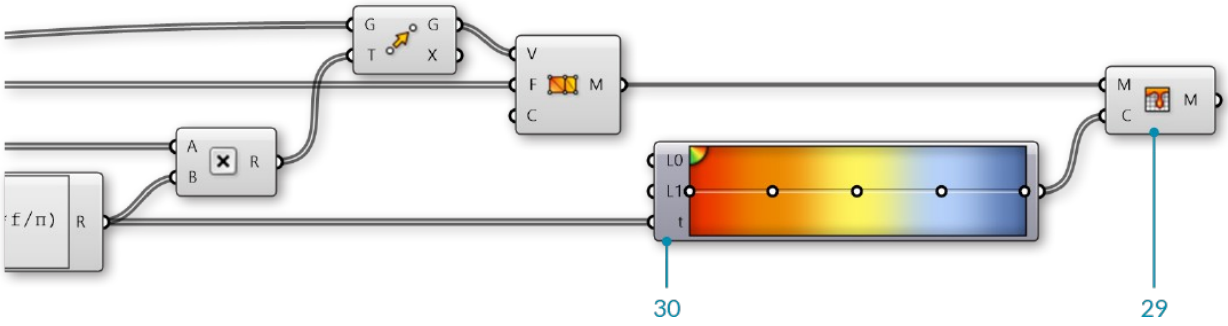


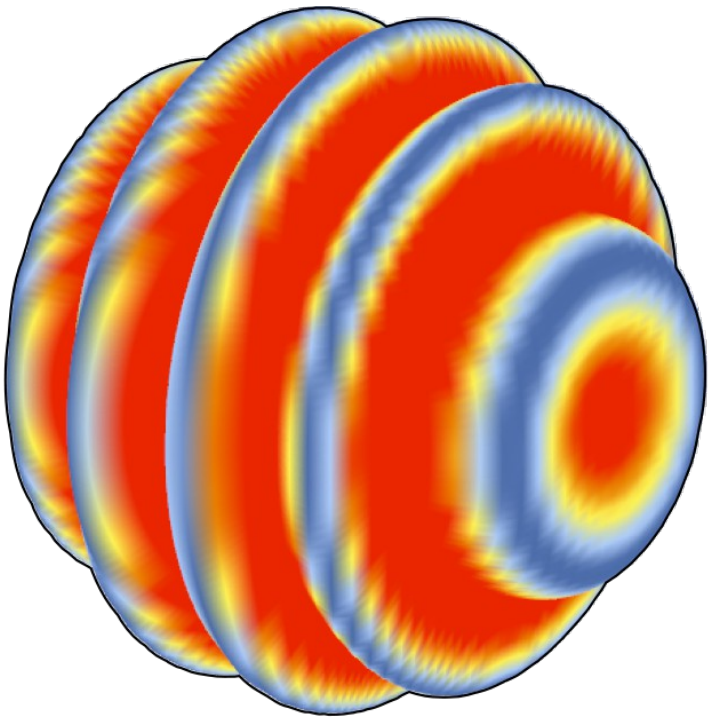


Откорректируйте числовые слайдеры Amplitude (Амплитуда) и Frequency (Частота), чтобы увидеть, как изменится вновь построенная полигональная сетка.

29.	**Mesh/Primitive/Mesh Colours (Полигональная сетка/Примитивы/Цвета Полигональной сетки)** — Перетащите на холст компонент **Mesh Colours (Цвета Полигональной сетки)**	
30.	**Params/Input/Gradient (Параметры/Вводные/Градиент)** — Перетащите на холст компонент **Gradient (Градиент)**	

	Вы можете кликнуть правой кнопкой мыши по компоненту Градиент и выбрать "Presets" (Предустановленные), чтобы изменить цвета градиента. В данном примере мы использовали градиент Red-Yellow-Blue (Красно-Жёлто-Синий)	
31.	Подсоедините выход Result (R) (Результат) компонента **Expression (Выражение)** ко входу Parameter (t) (Параметр) компонента **Gradient (Градиент)**	
32.	Подсоедините выход компонента **Gradient (Градиент)** ко входу Colours (C) (Цвета) компонента **Mesh Colours (Цвета Полигональной сетки)**	
33.	Подсоедините выход Mesh (M) (Полигональная сетка) компонента **Construct Mesh (Сконструировать Полигональную сетку)** ко входу Mesh (M) (Полигональная сетка) компонента **Mesh Colours (Цвета Полигональной сетки)** На этом этапе, того же результата можно было бы достичь, подключив градиент напрямую ко входу Colours (C) (Цвета) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	

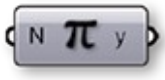


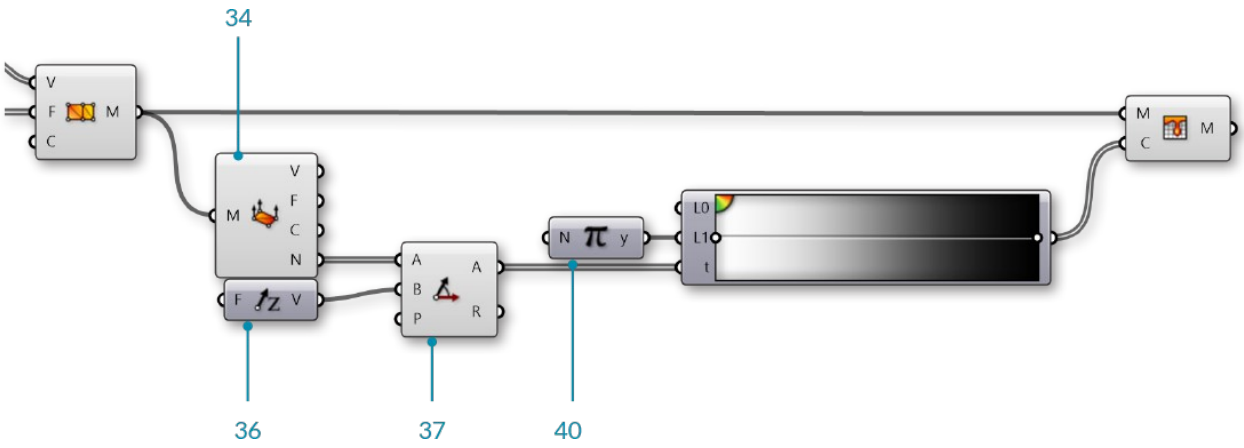


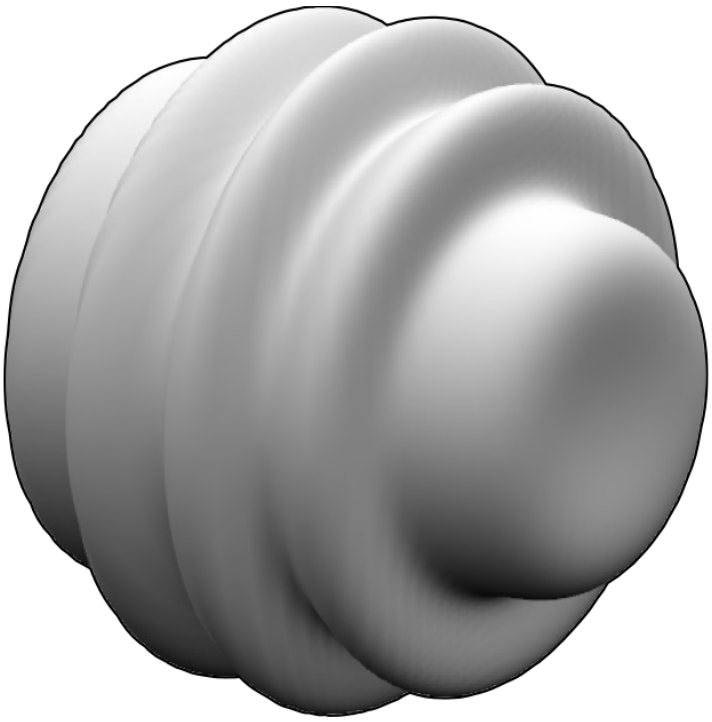
> Мы использовали результаты Expression (Выражения), чтобы осуществить два действия: перемещение вершин и окрашивание полигональной сетки. Таким образом, оттенки цветового градиента соответствуют величинам перемещения вершин.

В финальной части упражнения мы будем использовать соотношение направлений векторов нормалей относительно вектора «источника света», чтобы сымитировать базовые процессы рендеринга полигональной сетки.

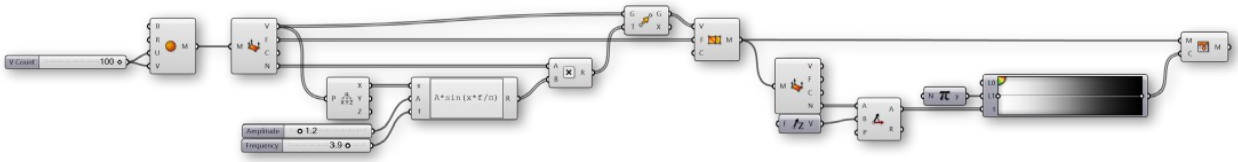
34.	**Mesh/Analysis/Deconstruct Mesh (Полигональная сетка/Анализ/Разобрать Полигональную сетку)** — Перетащите на холст компонент **Deconstruct Mesh (Разобрать Полигональную сетку)**	
35.	Подсоедините выход Mesh (M) (Полигональная сетка) компонента **Construct Mesh (Сконструировать Полигональную сетку)** ко входу Mesh (M) (Полигональная сетка) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** > Несмотря на то, что топология исходной полигональной сетки не изменилась, нормали векторов будут отличаться. Поэтому мы должны использовать новый Deconstruct Mesh (Разобрать Полигональную сетку), чтобы определить новые нормали.	
36.	**Vector/Vector/Unit Z (Вектор/Вектор/Унифицированный по Z)** — Перетащите на холст компонент **Unit Z (Вектор, Унифицированный по Z)**	

	> Мы будем его использовать в качестве направленного источника света. Вы можете использовать и другие векторы. Если использовать линию из Rhino в качестве ссылочной, то получите динамически изменяемое направление света.	
37.	**Vector/Vector/Angle (Вектор/Вектор/Угол)** — Перетащите на холст комопнент **Angle (Угол)**	
38.	Подсоедините выход Normals (N) (Нормали) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу A компонента **Angle (Угол)**	
39.	Подсоедините выход компонента U**nit Z (Вектор, Унифицированный по Z)** ко входу B компонента **Angle (Угол)**	
40.	**Maths/Util/Pi (Математика/Утилиты/Число Пи)** — Перетащите на холст компонент **Pi (Число Пи)**	
41.	Подсоедините компонент **Pi (Число Пи)** ко входу Upper Limit (L1) (Верхний Предел) компонента **Gradient (Градиент)**	
42.	Подсоедините выход Angle (A) (Угол) компонента **Angle (Угол)** ко входу Parameter (t) (Параметр) компонента **Gradient (Градиент)**	





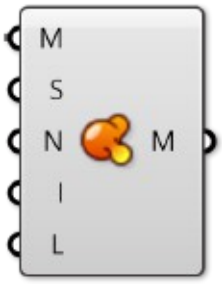
Мы использовали предустановленный градиент white-to-black (от белого к чёрному). Этот набор цветов полигональной сетки соотносится с углами между нормалями и источником света. В случае, когда нормаль грани направлена непосредственно на источник света, её цвет будет чёрным, а когда её направление противоположно источнику света — белым. (Чтобы быть педантично точными, Вы можете обратить градиент, откорректировав его рукояти). Настоящий процесс рендеринга полигональной сетки гораздо более сложен, чем мы представили сейчас, но это — основа процесса создания света и тени визуализируемых объектов.



1.6.4 Операции над Полигональными сетками

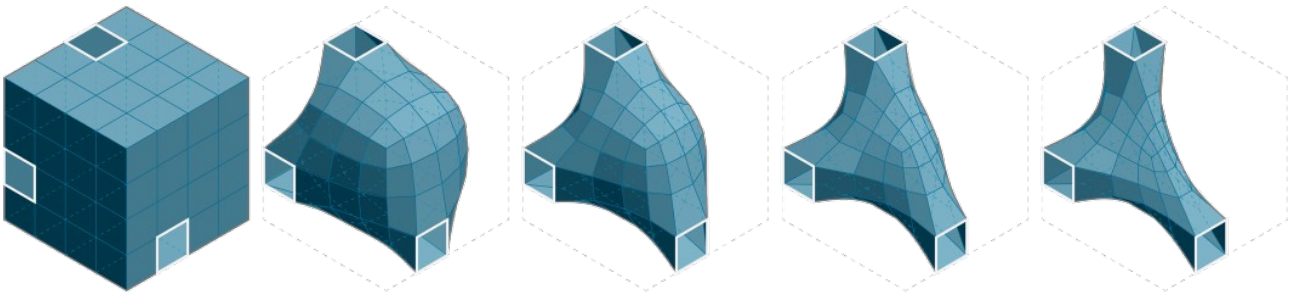
В последнем разделе мы рассмотрели основы структуры полигональной сетки. В этом разделе мы рассмотрим способы манипулирования геометрией полигональной сетки.

1.6.4.1 Smooth (Сглаживание)



Большая сглаженность полигональной сетки иногда может быть достигнута простым увеличением числа граней в процессе, называемом *subdivision (подразделение)*. Часто это приводит к экстремально большим наборам данных, которые отнимают большое количество времени вычислений, а также требуют дополнительных аддонов (надстроек) к Grasshopper, не включенных в данный учебник. В таких случаях, комопнент **Smooth (Сгаживание)** может быть использован в качестве альтернативы, чтобы сделать сетку менее зазубренной или гранёной без увеличения количества вершин и граней или изменения топологии. Для контролем над процессом сглаживания могут быть использованы: *интенсивность сглаживания (strength), количество последовательных шагов сглаживания (number of iterations)* и *предел сдвига (limit)*.

Прикрепление логического значения ко входу N предоставляет опциональное указание, пропускать ли открытые вершины (naked vertices). Вершина является открытой (naked), если она является частью открытого края (naked edge), то есть частью границы (boundary) открытой полигональной сетки (open mesh). Переключая эту опцию, Вы определяете, затронет ли сглаживание, в том числе, и внешние границы, при сглаживании внутренних краёв полигональной сетки.

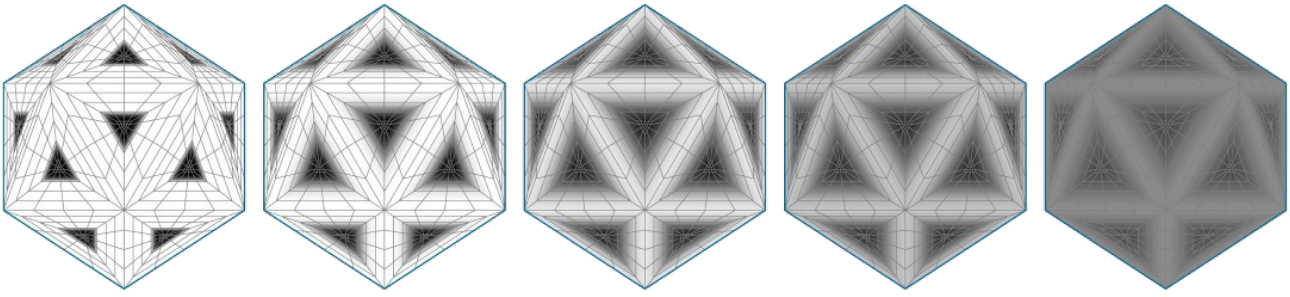


1 2 3 4 5

1. Исходный полигональный параллелепипед с тремя удалёнными гранями
2. Сглаживание после двух итераций (последовательных прохождений через операцию)
3. 6 итераций
4. 25 итераций
5. 50 итераций

1.6.4.2 Blur (Размытие)

Компонент **Blur (Размытие)** действует аналогичным образом, как и Smooth (Сглаживание), за исключением того, что оказывает влияние лишь на цвета вершин. Он также может быть использован для уменьшения вида зазубренности расцвеченных полигональных сеток, хотя и в меньшей степени, так как не изменяет геометрию.

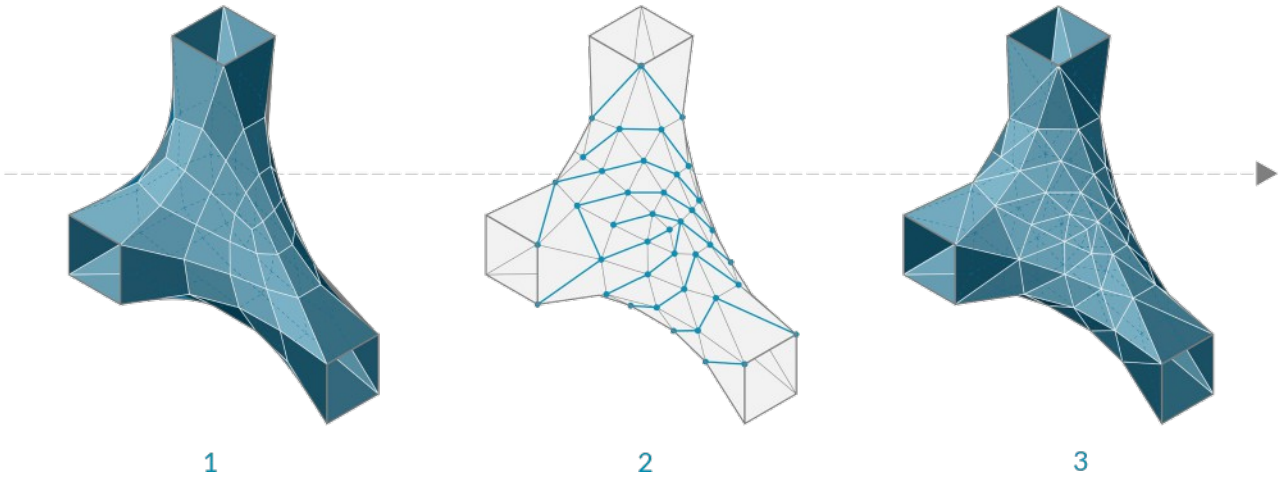


1 2 3 4 5

1. Исходная полигональная сетка
2. Размытие после 1 итерации
3. 6 итераций
4. 12 итераций
5. 20 итераций

1.6.4.3 Triangulate (Триангуляция)

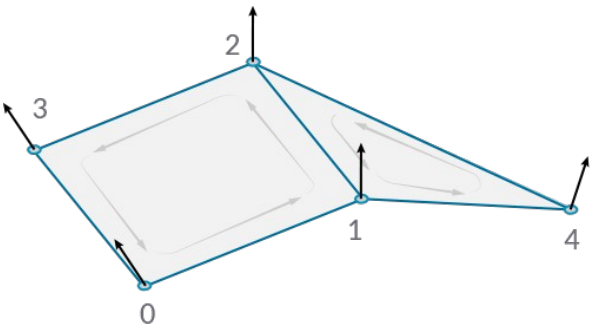
Для того, чтобы убедиться, что каждая грань является плоской, либо, если полигональная сетка будет экспортироваться в другую программу, которая не допускает обработку четырёхугольных граней, иногда требуется триангуляция полигональной сетки. При использовании компонента **Triangulate (Триангуляция)** каждая четырёхугольная грань заменяется двумя треугольными гранями. Для создания нового края Grasshopper всегда использует более короткую диагональ исходной грани.



1. Исходная полигональная сетка с четырёхугольными гранями
2. Новые края добавляются, пересекая исходные четырёхугольные грани по наикратчайшему расстоянию
3. Результирующая триангулированная полигональная сетка

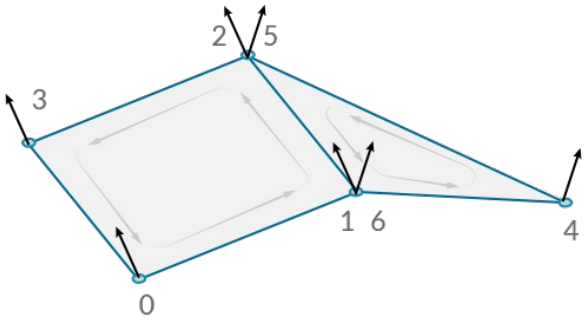
1.6.4.4 Weld (Сварка)

В последнем разделе мы отмечали, что одна вершина может совместно использоваться смежными гранями и нормаль этой вершины вычисляется, как среднее от нормалей этих смежных граней, что позволяет сглаживать визуализацию. Тем не менее, иногда необходимо показать острую складку или шов, где одна грань негладко переходит к следующей, что делается при помощи нужного направления нормалей вершин. Для такой ситуации необходимо, чтобы каждая грань имела свою собственную вершину с её собственной нормалью. В таком случае, список вершин будет содержать, по крайней мере, две точки, которые имеют одни и те же координаты, но разные индексы.



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

Face List
Q {0, 1, 2, 3}
T {1, 4, 2}



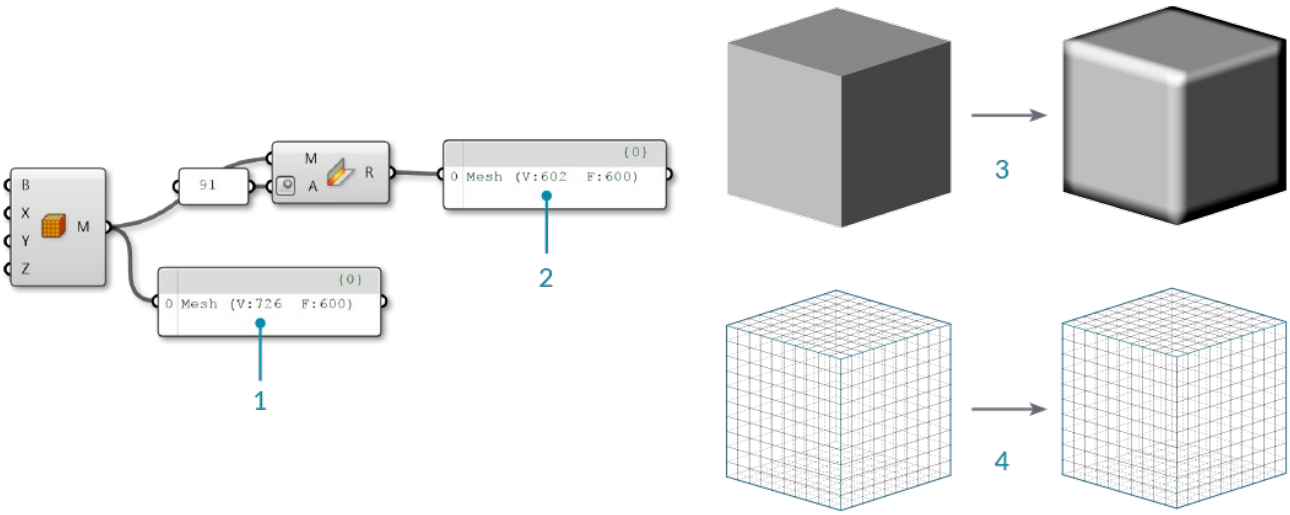
2 Vertex List
0 = {0.0, 0.0, 0.0}
1 = {1.0, 0.0, 1.0}
2 = {1.0, 1.0, 1.0}
3 = {0.0, 1.0, 0.0}
4 = {2.0, 0.0, -1.0}
5 = {1.0, 1.0, 1.0}
6 = {1.0, 0.0, 1.0}

Face List
Q {0, 1, 2, 3}
T {4, 5, 6}

1. Сваренные Грани (Welded Faces) — Обе грани совместно используют вершины 1 и 2. Нормали этих вершин усредняются от нормалей вершин смежных граней.
2. Разделённые Грани (Unwelded Faces) — Дублируют вершины, добавляя их к списку. В таком случае, грани не будут совместно использовать какие-либо индексы вершин. Вершины 1 и 6, а также вершины 2 и 5 имеют идентичные координаты, но это — разные вершины. Каждая из них имеет собственные нормали вершин.

Процесс объединения двух вершин, находящихся в одном и том же местоположении в одну вершину называется *сваркой (welding)*, и наоборот, если одна вершина разделяется на несколько вершин, то происходит *распайка вершин (unwelding)*.

В качестве вводных данных компонент Weld (Сварка) использует предельный угол (threshold angle). Любые две смежные грани со значением угла менее порогового будут сварены вместе. Unweld (Разделение) работает противоположным образом: там, где смежные грани соприкасаются под углом больше порогового, они будут разъединены, а их совместно используемые вершины будут продублированы.



1. Заданный по-умолчанию Box Mesh (Параллелепипед из Полигональной сетки) имеет 726 вершин. Сетка уплотняется в углах параллелепипеда, где количество вершин удваивается.
2. Если полигональная сетка сваривается под углами более 90 градусов, то результирующие грани свариваются и число вершин уменьшается до 602, а количество граней остаётся прежним.
3. Предварительный просмотр геометрии показывает, что сварка полигональной сетки визуально сгладило углы.
4. В отличие от компонента Smooth (Сгладить), который изменяет геометрию сетки, эта полигональная сетка отображается гладкой только благодаря роли нормалей вершин в процессах рендеринга и шейдинга (затенения). Фактическое местоположение вершин остаётся неизменным.

В изображении, представленном выше, мы использовали угол в 91 градус, потому что известно, что углы квадрата должны быть 90 градусов. Чтобы применить сварку ко всей полигональной сетке, Вы должны использовать угол 180 градусов.

1.6.5 Взаимодействие с Полигональными сетками

В этом разделе описываются способы, которыми Объекты из Полигональной сетки могут взаимодействовать с другими объектами, например, для оценки ближайших точек к сетке, или комбинирования нескольких полигональных сеток воедино.

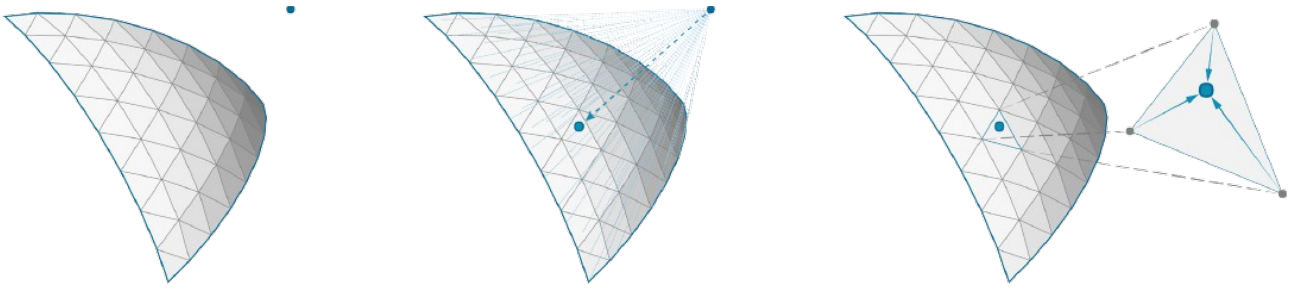
1.6.5.1 Полигональная Геометрия (Mesh Geometry) и Точки (Points)

Inclusion (Включение)

Этот компонент тестирует, находится ли указанная точка внутри твердотельной полигональной сетки или нет. Он работает только на замкнутых полигональных сетках.

Mesh Closest Point (Ближайшая Точка Полигональной сетки)

Этот компонент вычисляет местоположение на полигональной сетке, которое является ближайшим к заданной точке. Этот компонент выдаёт три порции данных: координаты вычисленной точки на полигональной сетке, индекс грани, содержащей эту точку и параметр (parameter) на полигональной сетке. Параметр является чрезвычайно полезными данными, особенно в сочетании с компонентом Mesh Eval (Оценка Полигональной сетки) и будет обсуждаться ниже.

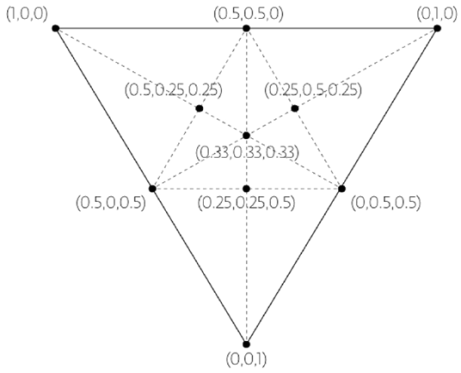


1 2 3

1. Задаём точку в пространстве, Мы хотим найти на полигональной сетке точку, являющуюся ближайшей к заданной.
2. Грань, содержащая ближайшую точку идентифицирована.
3. Параметр ближайшей точки на грани вычислен.

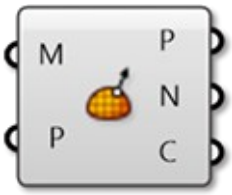
Для пользователей, которым интересно узнать больше деталей о параметризации полигональной сетки, мы можем дать более узкое понятие о структуре параметра полигональной сетки. Вы можете увидеть данную структуру, присоединив Текстовую Панель к выходу Parameter (Параметр) компонента **Mesh Closest Point (Ближайшая Точка Полигональной сетки)**. Параметр полигональной сетки складывается из: N[A,B,C,D]. Первый номер — N, это индекс грани, содержащей вычисленную точку.

Следующие четыре цифры определяют *барицентрические* координаты (barycentric coordinates) точки на этой грани. Координаты ссылочной точки можно найти перемножением координат каждой вершины этой грани по порядку вышеуказанных их номеров и сложением полученных результатов вместе. (Конечно, это сделано для нас и на выходе даёт Point (Точку)). Также обратите внимание и на то, что барицентрические координаты также уникальны и для треугольных граней, означая то, что одна и та же точка может иметь несколько различных параметризаций. Grasshopper избегает данной проблемы путём внутренней триангуляции четырёхугольной грани при расчёте, результатом которого является четыре числа параметра полигональной сетки, из которых, по крайней мере, один всегда будет равен нулю.



Бариметрические Координаты

Mesh Eval (Оценка Полигональной сетки)



Компонент **Mesh Eval (Оценка Полигональной сетки)** использует параметр полигональной сетки в качестве вводного и возвращает ссылочную точку, а также данные о нормали и цвете в данной точке. Цвет и нормаль вычисляется как интерполяция цвета вершин и нормалей вершин, используя те же самые барицентрические координаты параметра полигональной сетки.

1.6.5.2 Combining Mesh Geometry

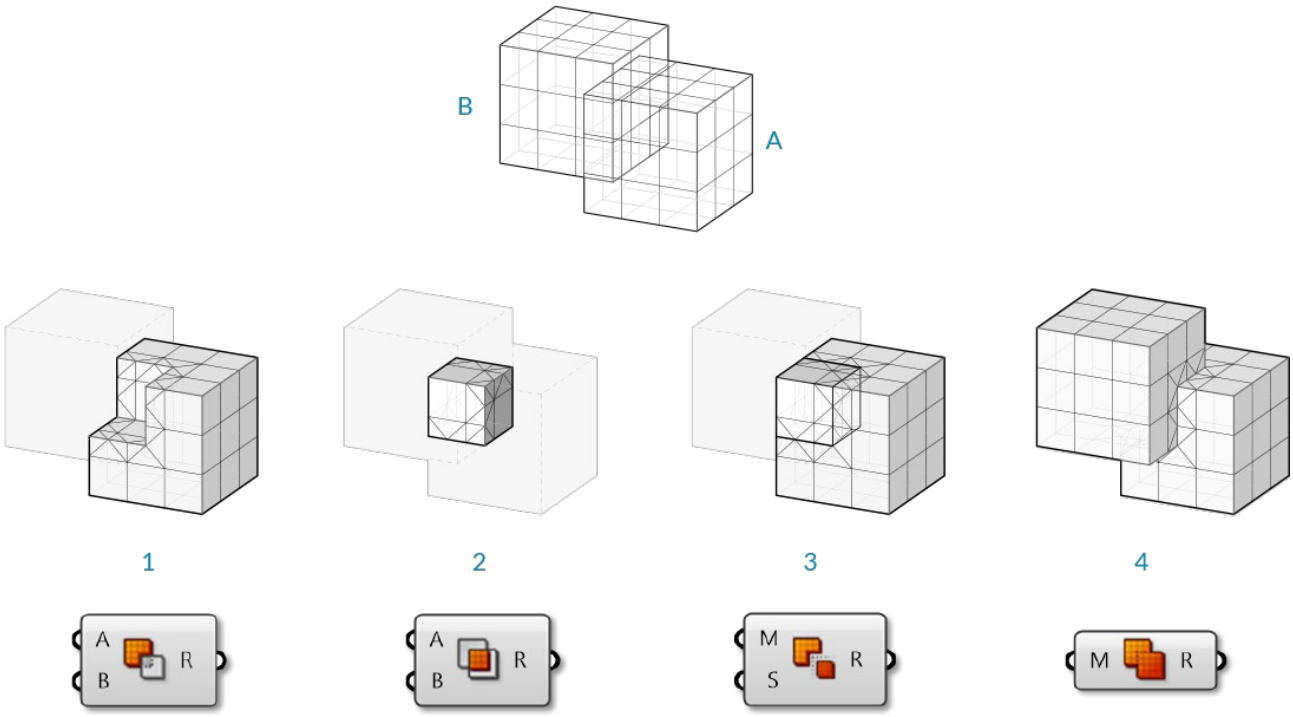
Mesh Join (Объединение Полигональных сеток)

В отличие от объединения кривых или NURBS-поверхностей, которые для этого требуют смежности, любые полигональные сетки могут быть объединены в единую полигональную сетку даже, если исходные сетки нигде не соприкасаются. Напомним, что полигональной сеткой является просто список вершин и список граней. И в них требования фактического соединения этих граней (хотя в большинстве приложений такое состояние полигональной сетки будет не слишком желательно!!).

Этот компонент не сваривает вершины полигональных сеток вместе, так что часто бывает полезно использовать его совместно с компонентом Weld (Сварка).

Булевы операции с Полигональными сетками

Полигональные сетки в Grasshopper имеют набор булевских операций, подобных булевским операциям с твёрдыми телами для NURBS. Булевские операции имеют специфику порядка подключения ко входам A и B, от которого зависит результат на выходе.

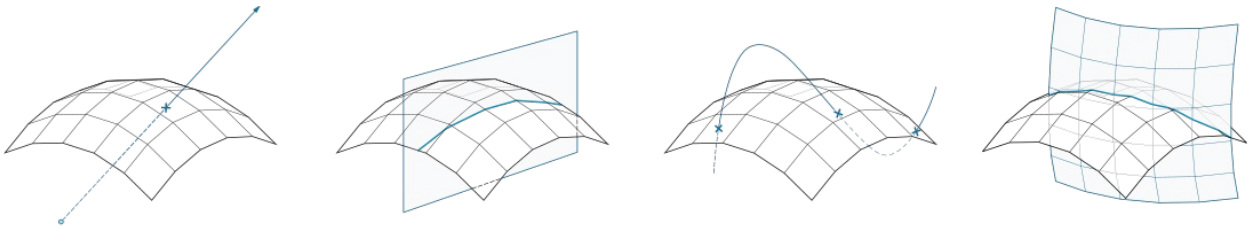


1. Вычитание Полигональных сеток (Mesh Difference)
2. Пересечение Полигональных сеток (Mesh Intersection)
3. Разбиение Полигональных сеток (Mesh Split)
4. Слияние Полигональных сеток (Mesh Union)

1.6.5.3 Пересечения и Преграды

Intersect (Пересечение)

Можно вычислить пересечения как между полигональными сетками, так и другими пересекающимися с ними объектами: лучами (rays), плоскостями (planes), кривыми (curves).



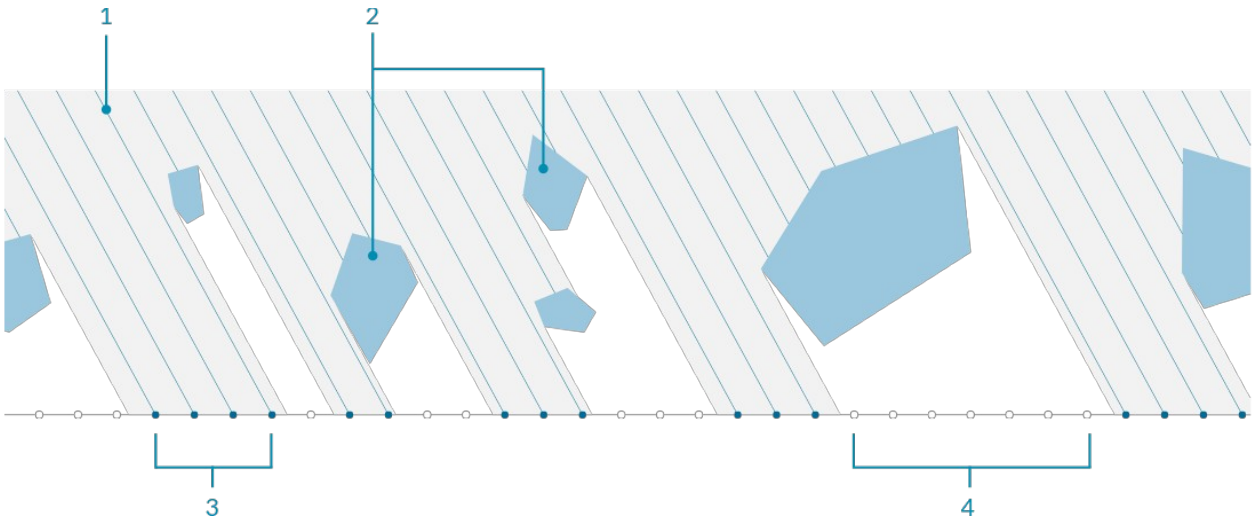
1 2 3 4

1. Полигональная сетка и Луч (Mesh | Ray)
2. Полигональная сетка и Плоскость (Mesh | Plane)
3. Полигональная сетка и Кривая (Mesh | Curve)
4. Полигональная сетка и Полигональная сетка (Mesh | Mesh)

Occlusion (Преграда)

Как мы уже говорили, одно (из многих) использований полигональной сетки — использование в качестве геометрии для визуализации и создания теней, основанное на нормалях граней. При рендеринге также необходимо знать, когда объект находится в тени, отбрасываемой другим объектом. Компонент Occlusion (Преграда) позволяет ввести в Grasshopper набор точек выборки (sample points), из которых полигональная геометрия будет «отбрасывать тени», а также луча видимости, то есть вектор, указывающий направление исходящего «света».

Такой процесс может быть использован для создания теней в визуализации или определения того, не скрыты ли объекты из поля зрения камеры, установленной в определённой точке.



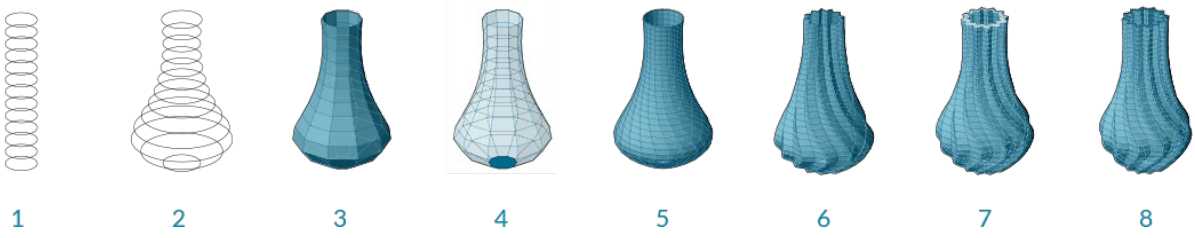
1. Луч Видимости (View Ray) для проверки на наличие преград
2. Прегражданющая полигональная геометрия
3. «Попадающие» точки выборки
4. «Прегражднные» точки выборки

1.6.6 Работа с Полигональной Геометрией

В этом разделе мы будем работать с файлом примера для получения полигонального твёрдого тела (mesh solid). К завершению данного упражнения мы будем иметь динамический дифинишин для производства вазы настраиваемой формы, которая впоследствии может быть напечатана в 3D.

Скачать файлы, сопровождающие данный раздел.

Поскольку данный дефинишин несколько больше предыдущих примеров этого учебника, мы сначала пройдём через основные шаги, которые будем предпринимать:

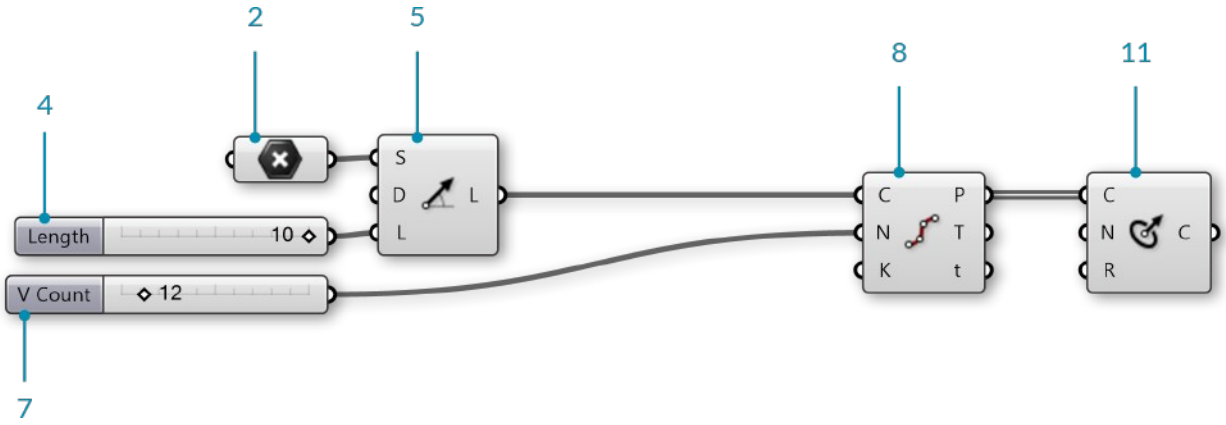


1. Создадим серию окружностей в качестве базового цилиндра
2. Используем компонент Graph Mapper (Переналожение по Графу) для определения профиля нашей вазы
3. Сконструируем топологию граней полигональной сетки, которая задаст единую форму поверхности полигональной сетки
4. Закроем нижнее отверстие полигональной сетки

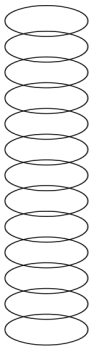
5. Для придания форме большей динамичности введём элемент скручивания в вертикальной ориентации
6. Добавим гофрированные гребни, задающие вазе текстуру
7. Создадим поверхность по отступу от полигональной сетки, которая придаст стенкам вазы толщину
8. Закроем верхнее отверстие межу двумя поверхностями стенки вазы, создав замкнутое твёрдое тело

1.	Начните новый дефинишин, нажав Ctrl-N (в Grasshopper)	
2.	**Params/Geometry/Point (Параметры/Геометрия/Точка)** — Перетащите контейнер Point на холст	
3.	Чтобы задать в окне Rhino ссылочную точку, нужно в Grasshopper кликнуть по компоненту **Point (Точка)** правой кнопкой мыши и выбрать "Set one point" (Задать одну точку). Эта точка послужит базовой точкой нашей вазы. Вы можете создать точку вручную прямо в Grasshopper, совершив двойной клик по холсту, и в открывшимся окне поиска ввести координаты точки через запятую, например: «0,0,0» (без кавычек)	
4.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Number Slider (Числовой Слайдер)** и задайте в нём следующие значения: Name: Length (Имя: Длина) Lower Limit: 1 (Нижний Предел) Upper Limit: 10 (Верхний Предел)	
5.	Curve/Primitive/Line SDL (Кривая/Примитивы/ (Прямая SDL (Начало, Касательная, Длина)) — Перетащите на холст компонент Line SDL (Прямая SDL)	
6.	Подсоедините компонент **Point (Точка)** ко входу Start (S) (Начало) компонента **Line SDL (Праямя SDL)**, а **Числовой Слайдер** ко входу Length (L) (Длина). По-умолчанию значение входа Direction (D) (Направление) компонента Line SDL (Прямая SDL) настроено на значение Unit Z vector (Вектор, Унифицированный по Z), который и будет использоваться в данном примере	
7.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Number Slider (Числовой**	

	Слайдер) и задайте в нём следующие значения: Name: V Count (Имя: Количество по Вертикали) Rounding: Integer (Округление: Целые числа) Lower Limit: 1 (Нижний Предел) Upper Limit: 100 (Верхний Предел)	
8.	**Curve/Division/Divide Curve (Кривая/Разделение/Разделить Кривую)** — Перетащите на холст компонент **Divide Curve (Разделить Кривую)**	
9.	Подсоедините выход Line (L) (Прямая) компонента **Line SDL (Прямая SDL)** ко входу Curve (C) (Кривая) комопнента **Divide Curve (Разделить Кривую)**	
10.	Подсоедините **Числовой слайдер V Count (Количество по Вертикали)** ко входу Count (N) (Количество) компонента **Divide Curve (Разделить Кривую)**	
11.	**Curve/Primitive/Circle CNR (Кривая/Примитивы/Окружность CNR (Центр, Нормаль, Радиус))** — Перетащите на холст компонент **Circle CNR (Окружность CNR)**	
12.	Подсоедините выход Points (P) (Точки) компонента **Divide Curve (Разделить Кривую)** ко входу Center (C) (Центр) копонента **Circle CNR (Окружность CNR)**	

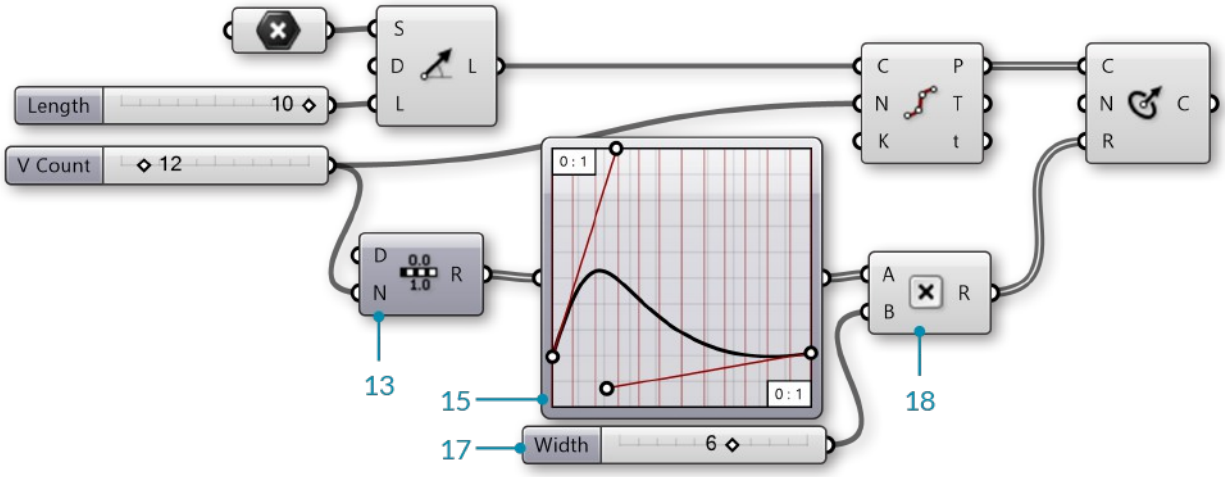


Мы получили серию окружностей, сложенных в вертикальную стопку. Их мы будем использовать в качестве профилей для нашей вазы.



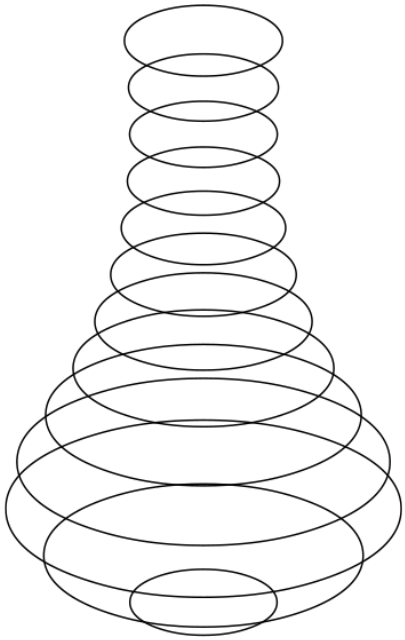
Далее мы задействуем Graph Mapper (Переналожение по Графу), с помощью которого мы будем задавать новые радиусы этим окружностям.

13.	**Sets/Sequence/Range (Наборы/Последовательность/Диапазон)** — Перетащите на холст компонент **Range (Диапазон)**	
14.	Подсоедините **Числовой слайдер V Count (Количество по Вертикали)** ко входу Steps (N) (Шаги) компонента **Range (Диапазон)**	
15.	**Params/Input/Graph Mapper (Параметры/Вводные/Переналожение по Графу)** — Перетащите на холст компонент **Graph Mapper (Переналожение по Графу)**	
16.	Кликните правой кнопкой мыши по Graph Mapper, и выберите в выпавшем контекстном меню из списка 'Graph Types' (Типы Графов) 'Bezier' (Безье (Рычажный Граф))	
17.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Number Slider (Числовой Слайдер)** и задайте в нём следующие значения: Name: Width (Имя: Ширина) Lower Limit: 0 (Нижний Предел) Upper Limit: 10 (Верхний Предел)	
18.	**Maths/Operators/Multiplication (Математика/Операторы/Умножение)** — Перетащите на холст компонент **Multiplication (Умножение)**	
19.	Подсоедините **Graph Mapper (Переналожение по Графу)** и **Числовой слайдер Width (Ширина)** ко входам A и B компонента **Multiplication (Умножение)**	
20.	Подсоедините выход Result (R) (Результат) компонента **Multiplication (Умножение)** ко входу Radius (R) (Радиус) компонента **Circle CNR (Окружность CNR (Центр, Нормаль, Радиус))**	



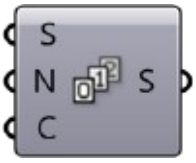
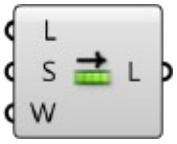
Используйте рычаги в **Graph Mapper (Переналожение по Графу)**, чтобы откорректировать профильные окружности.

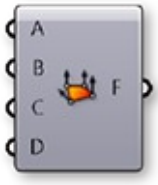
ПРИМЕЧАНИЕ: Важно убедиться, что начальная точка кривой Безье в **Graph Mapper (Переналожение по Графу)** находится не на нуле. Приподнимая начальную точку выше нуля, мы создаём плоское основание нашей вазы.

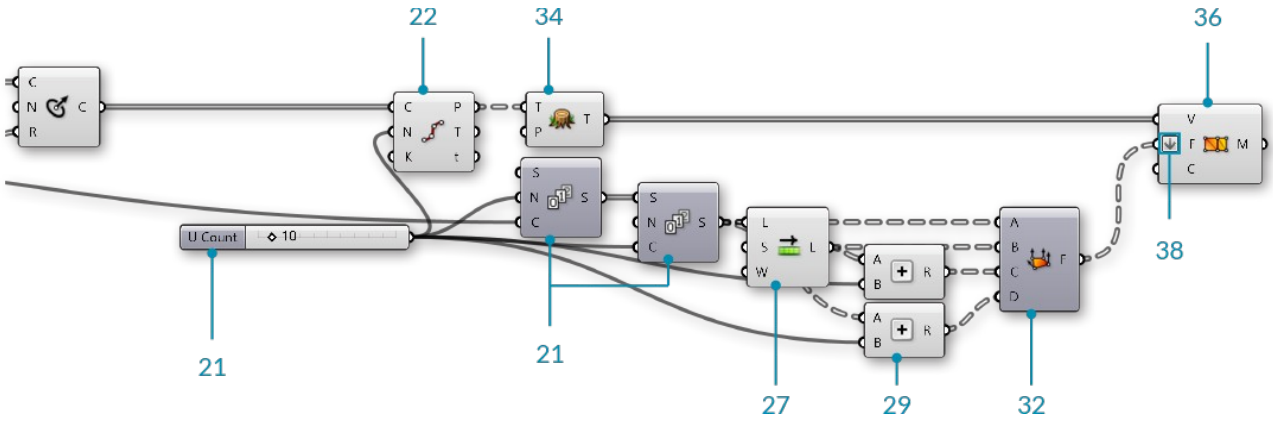


Теперь у нашей вазы есть заданный профиль. Далее мы построим поверхность из полигональной сетки. Для создания полигональной сетки потребуется создать вершины и определить грани в соответствии с индексами этих вершин.

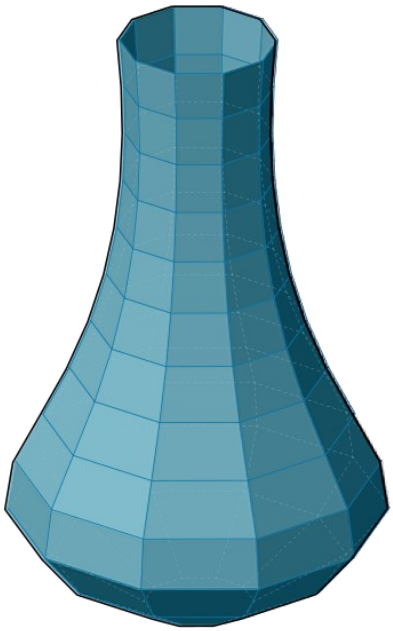
21.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Number Slider (Числовой Слайдер)** и задайте в нём следующие значения: Name: U Count (Имя: Количество по Горизонтали)	

	Rounding: Even (Округление: Чётный) Lower Limit: 2 (Нижний Предел) Upper Limit: 100 (Верхний Предел)	
22.	**Curve/Division/Divide Curve (Кривая/Разделение/Разделить Кривую)** — Перетащите на холст компонент **Divide Curve (Разделить Кривую)**	
23.	Подсоедините выход Circle (C) (Окружность) компонента **Circle CNR (Окружность CNR)** ко входу Curve (C) (Кривая) компонента **Divide Curve (Разделить Кривую)** и подсоедините выход **Числового слайдера U Count (Количество по Горизонтали)** ко входу Count (N) (Количество) Выход Points(P) (Точки) этого компонента, в нашем случае, выдаёт вершины нашей будущей полигональной сетки	
24.	**Sets/Sequence/Series (Наборы/Последовательность/Серия)** — Перетащите на холст два компонента **Series (Серия)**	
25.	Подсоедините **Числовой слайдер U Count (Количество по Горизонтали)** ко входу Step (N) (Шаг) первого компонента **Series (Серия)** и подсоедините **Числовой слайдер V Count (Количество по Вертикали)** ко входу Count (C) (Количество) того же компонента **Series (Серия)**	
26.	Подсоедините выход Series (S) (Серия) первого компонента **Series (Серия)** ко входу Start (S) (Начало) второго компонента **Series (Серия)** и подсоедините **Числовой слайдер U Count (Количество по Горизонтали)** ко входу Count (C) (Количество)	
27.	**Sets/List/Shift List (Наборы/Список/Сдвиг Списка)** — Перетащите на холст компонент **Shift List (Сдвиг Списка)**	
28.	Подсоедините от второго компонента **Series (Серия)** выход ко входу List (L) (Список) компонента **Shift List (Сдвиг Списка)**	
29.	**Maths/Operators/Addition (Математика/Операторы/Сложение)** — Перетащите на холст два компонента **Addition (Сложение)**	
30.	Подсоедините выход второго компонента **Series (Серия)** и Числовой слайдер **U Count (Количество по Горизонтали)** ко входам A и B первого компонента **Addition (Сложение)**	
31.	Подсоедините выход компонента **Shift List (Сдвиг Списка)**	

	и Числовой слайдер **U Count (Количество по Горизонтали)** ко входам A и B второго компонента **Addition (Сложение)**	
32.	**Mesh/Primitive/Mesh Quad (Полигональная сетка/Примитивы/Полигональная сетка с Четырёхугольными гранями)** — Перетащите на холст компонент **Mesh Quad (Полигональная сетка с Четырёхугольными гранями)**	
33.	Подключите следующие входы компонента **Mesh Quad (Полигональная сетка с Четырёхугольными гранями)**: A — Второй компонент Series (Серия) B — Shift List (Сдвиг Списка) C — Первый компонент Addition (Сложение) D — Второй компонент Addition (Сложение) Мы получили простое создание первоначальной топологии для нашей полигональной сетки. Эти грани будут скомбинированы с вершинами. Порядок этих подключений имеет критическое значение, так что вернитесь и дважды проверьте все имеющиеся на данный момент подключения!	
34.	**Sets/Tree/Flatten (Наборы/Дерево Данных/Обрубить)** — Перетащите на холст компонент **Flatten Tree (Обрубить Дерево Данных)**	
35.	Подсоедините выход Points (P) (Точки) компонента **Divide Curve (Разделить Кривую)** ко входу Tree (T) (Дерево Данных) компонента **Flatten Tree (Обрубить Дерево Данных)**	
36.	**Mesh/Primitive/Construct Mesh (Полигональная сетка/Примитивы/Сконструировать Полигональную сетку)** — Перетащите на холст компонент **Construct Mesh (Сконструировать Полигональную сетку)**	
37.	Подсоедините выход Tree (T) (Дерево Данных) компонента **Flatten Tree (Обрубить Дерево Данных)** ко входу Vertices (V) (Вершины) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	
38.	Подсоедините выход Face (F) (Грань) компонента **Mesh Quad (Полигональная сетка с Четырёхугольными гранями)** ко входу Faces (F) (Грани) компонента **Construct Mesh (Сконструировать Полигональную сетку)**. Кликните правой кнопкой мыши по входу F (Faces) (Грани) и выберите 'Flatten' (Обрубить)	

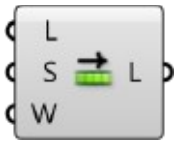
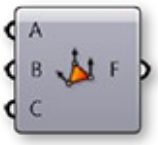


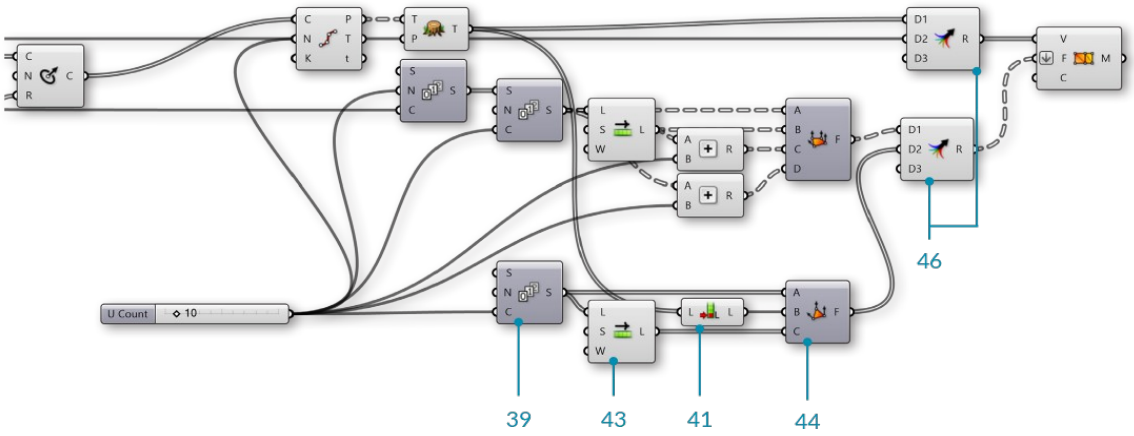
Теперь у нашей вазы есть полигональная поверхность.



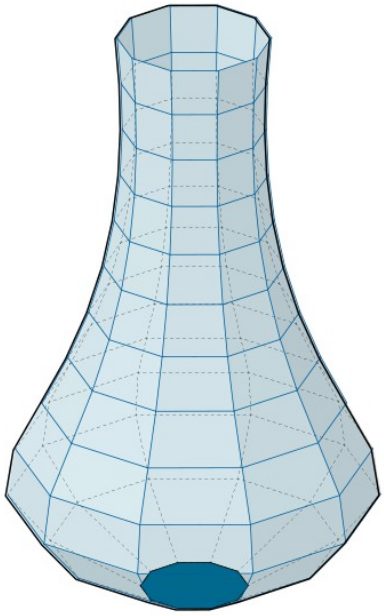
Далее мы закроем дно вазы. Чтобы сделать это, мы добавим первую начальную точку к списку вершин, а затем сконструируем треугольные грани полигональной сетки от нижних краёв до данной точки.

39.	**Sets/Sequence/Series (Наборы/Последовательность/Серия)** — Перетащите на холст компонент **Series (Серия)**	
40.	Подсоедините **Числовой слайдер U Count (Количество по Горизонтали)** ко входу Count (C) (Количество) компонента **Series (Серия)**	
41.	**Sets/List/List Length (Наборы/Список/Длина Списка)** — Перетащите на холст компонент **List Length (Длина Списка)**	
42.	Подсоедините выход Tree (T) (Дерево Данных) компонента **Flatten Tree (Обрубить Дерево)** ко входу List (L) (Список)	

	компонента **List Length (Длина Списка)** This will be the index of the origin point after we add it to the existing list of vertices.	
43.	**Sets/List/Shift List (Наборы/Список/Сдвиг Списка)** — Перетащите на холст компонент **Shift List (Сдвиг Списка)**	
44.	**Mesh/Primitive/Mesh Triangle (Полигональная сетка/Примитивы/Полигональная сетка с Треугольными гранями)** — Перетащите на холст компонент **Mesh Triangle (Полигональная сетка с Треугольными гранями)**	
45.	Подключите следующие входы компонента **Mesh Triangle (Полигональная сетка/Примитивы/Полигональная сетка с Треугольными гранями)**: A — Только что созданный компонент **Series (Серия)** B — **List Length (Длина Списка)** C — **Shift List (Сдвиг Списка)**	
46.	**Sets/Tree/Merge (Наборы/Дерево Данных/Слияние)** — Перетащите на холст два компонента **Merge (Слияние)**	
47.	Подсоедините выход Tree (T) (Дерево Данных) компонента **Flatten Tree (Обрубить Дерево)** ко входу D1 (Данные 1) и подсоедините начальный компонент **Point (Точка)** ко входу D2 (Данные 2) первого компонента **Merge (Слияние)**	
48.	Подсоедините выход Faces (F) (Грани) компонента **Mesh Quad (Полигональная сетка с Четырёхугольными гранями)** ко входу D1 (Данные 1) и подсоедините выход компонента **Mesh Triangle (Полигональная сетка с Треугольными гранями)** ко входу D2 второго компонента **Merge (Слияние)**	
49.	Подсоедините первый компонент **Merge (Слияние)** ео входу Vertices (V) (Вершины) компонента **Construct Mesh (Сконструировать Полигональную сетку)** и подсоедините второй компонент **Merge (Слияние)** ко входу Faces (F) (Грани) компонента **Construct Mesh (Сконструировать Полигональную сетку)**.	



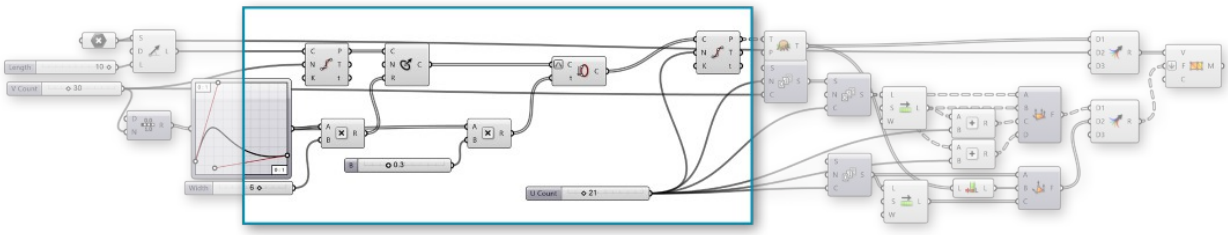
Мы получили закрыли низ вазы треугольными полигонами.

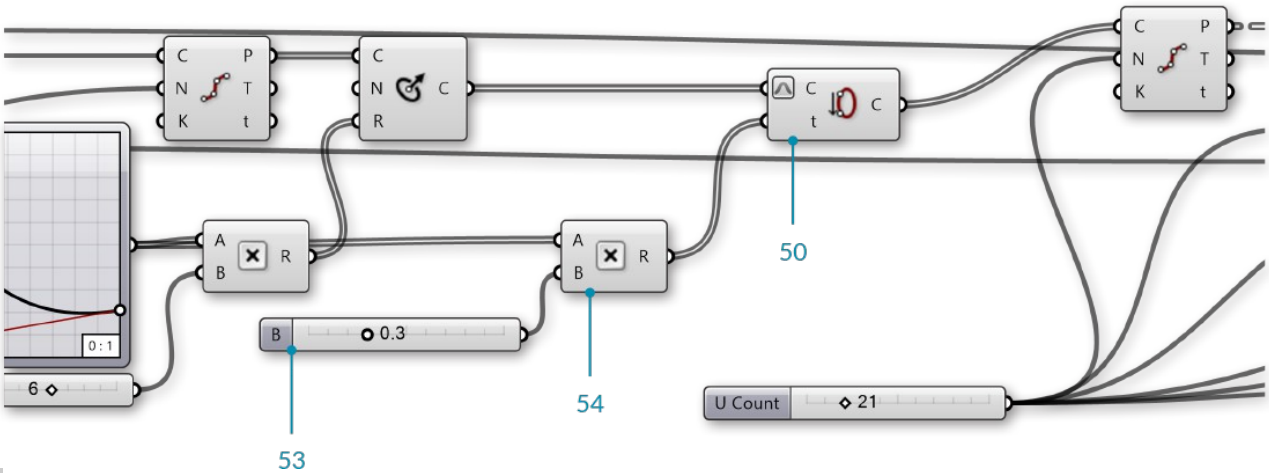


Теперь мы добавим вазе некоторую детализацию. Начнём с добавления искривления в вертикальном направлении, корректируя швы исходных окружностей.

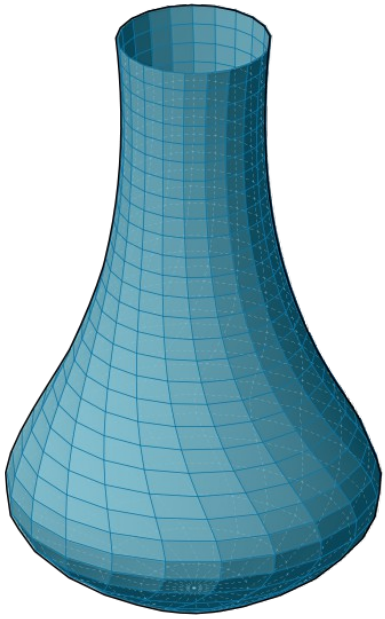
50.	Curve/Util/Seam (Кривая/Утилиты/Шов) — Перетащите на холст компонент Seam (Шов)	
51.	Подсоедините выход Circle (C) (Окружность) компонента **Circle CNR (Окружность CNR (Центр, Нормаль, Радиус))** ко входу Curve (C) (Кривая) компонента **Seam (Шов)**	
52.	Кликните правой кнопкой мыши по входу Curve (C) (Кривая) компонента **Seam (Шов)** и выберите 'Reparameterize' (Репараметризировать)	

53.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Number Slider (Числовой Слайдер)**. Можете просто использовать настройки по-умолчанию.	
54.	**Maths/Operator/Multiplication (Математика/Операторы/Умножение)** — Перетащите на холст компонент **Multiplication (Умножение)**.	
55.	Подсоедините выход компонента **Graph Mapper (Переналожение по Графу)** ко входу A и недавно созданный **Числовой Слайдер** ко входу B компонента **Multiplication (Умножение)**.	
56.	Подсоедините выход Result (R) (Результат) компонента **Multiplication (Умножение)** ко входу Parameter (t) (Параметр) компонента **Seam (Шов)**.	

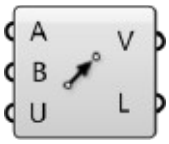




Кривизна достигается за счёт изменения положения шва исходных окружностей, используя при этом тот же Graph Mapper (Переналожение по Графу), что и для задания формы профиля вазы.

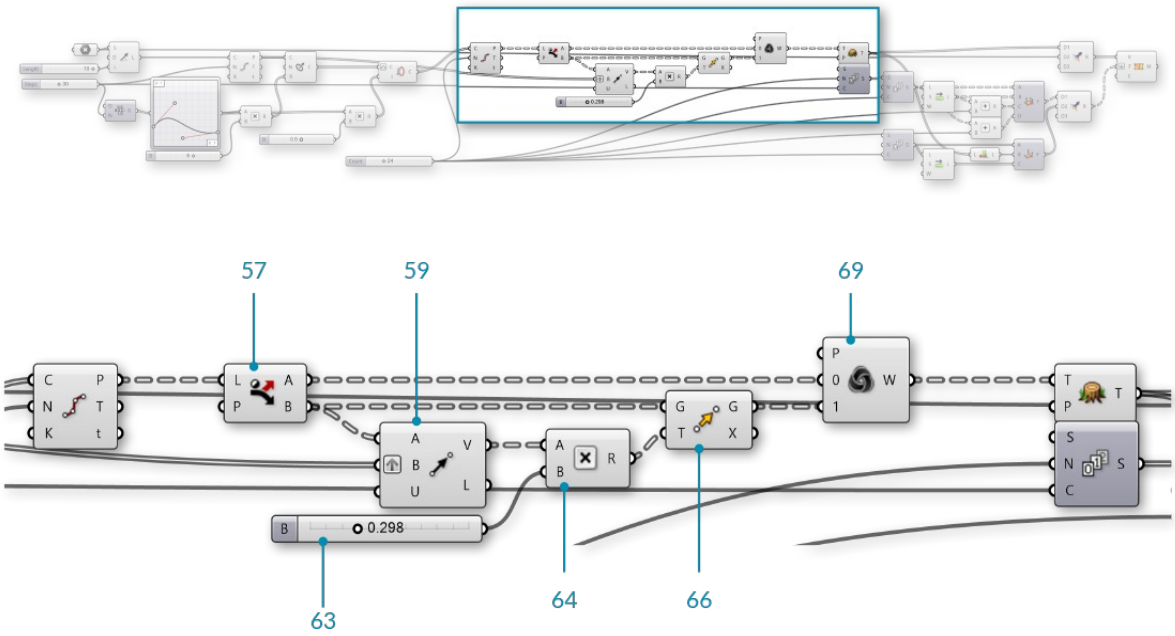


Далее мы добавим вазе небольшие вертикальные гребни.

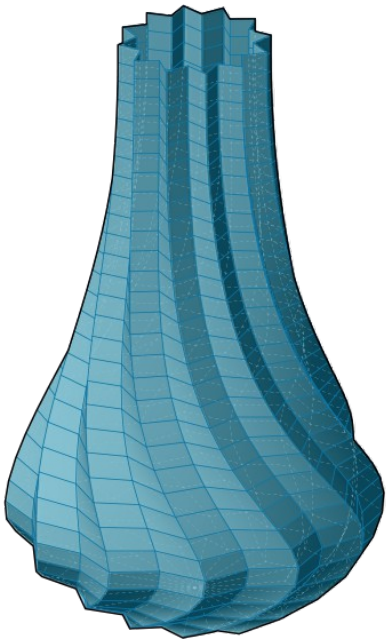
57.	**Sets/List/Dispatch (Наборы/Список/Диспетчер)** — Перетащите на холст компонент **Dispatch (Диспетчер)**	
58.	Подсоедините выход Point (P) (Точка) второго компонента **Divide Curve (Разделить Кривую)** ко входу List (L) (Список) компонента **Dispatch (Диспетчер)** Мы будем использовать шаблон-по-умолчанию входа Pattern (P) (Шаблон) компонента **Dispatch (Диспетчер)** для разделения точек на два списка с чередующимися точками.	
59.	**Vector/Vector/Vector 2Pt (Вектор/Вектор/Вектор по 2м Точкам)** — Перетащите на холст копонент **Vector 2Pt (Вектор по 2м Точкам)**	
60.	Подсоедините выход B компонента **Dispatch (Диспетчер)** ко входу A компонента **Vector 2Pt (Вектор по 2м Точкам)**	
61.	Подсоедините выход Points (P) (Точки) первого компонента **Divide Curve (Разделить Кривую)** ко входу B компонента **Vector 2Pt (Вектор по 2м Точкам)**	
62.	Кликните правой кнопкой мыши по входу B компонента **Vector 2Pt (Вектор по 2м Точкам)** и выберите 'Graft' (Привить Дерево Данных). Кликните правой кнопкой мыши и по входу Unitize (U) (Унифицировать результат) и в выпадающем списке 'Set Boolean' (Задать Логическое значение) выберите 'True' (Истина)	

	Это создает единичный вектор для каждой точки, которая указывает в направлении центра окружности	

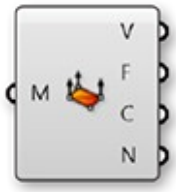
63.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Числовой Слайдер**. Мы будем использовать его настройки-по-умолчанию	
64.	**Maths/Operator/Multiplication (Математика/Оператор/Умножение)** — Перетащите на холст компонент **Multiplication (Умножение)**	
65.	Подсоедините выход Vector (V) (Вектор) компонента **Vector 2Pt (Вектор по 2м Точкам)** ко входу A и подсоедините **Числовой Слайдер** ко входу B компонента **Multiplication (Умножение)**	
66.	**Transform/Euclidean/Move (Трансформация/Евклидова/Перемещение)** — Перетащите на холст компонент **Move (Перемещение)**	
67.	Подсоедините выход B компонента **Dispatch (Диспетчер)** ко входу Geometry (G) (Геометрия) компонента **Move (Перемещение)**	
68.	Подсоедините выход Result (R) (Результат) компонента **Multiplication (Умножение)** ко входу Motion (T) (Движение) компонента **Move (Перемещение)**	
69.	**Sets/List/Weave (Наборы/Списки/Сплести)** — Перетащите на холст компонент **Weave (Сплести)**	
70.	Подсоедините выход A компонента **Dispatch (Диспетчер)** ко входу 0 компонента **Weave (Сплести)**	
71.	Подсоедините выход Geometry (G) (Геометрия) компонента **Move (Перемещение)** ко входу 1 компонента **Weave (Сплести)**	
72.	Подсоедините выход Weave (W) (Результат Сплетения) компонента Weave **(Сплести)** к входу Tree (T) (Дерево Данных) компонента **Flatten Tree (Обрубить Дерево)**	



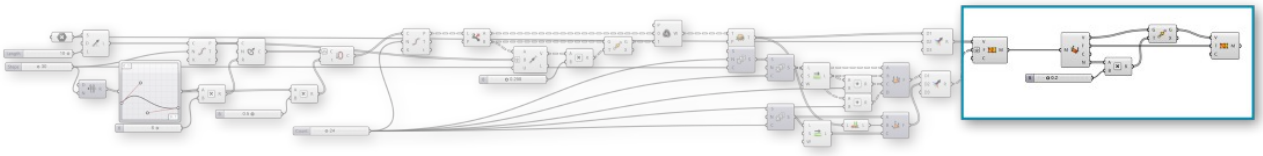
> Вернитесь назад и подкорректируйте слайдеры и граф маппер, чтобы увидеть, как изменяется модель, а также убедиться, что всё по-прежнему работает. Это действие известно как 'flexing' «проверка модели на прочность» и должно производиться почаще, чтобы убедиться в отсутствии ошибок в дефинишине.

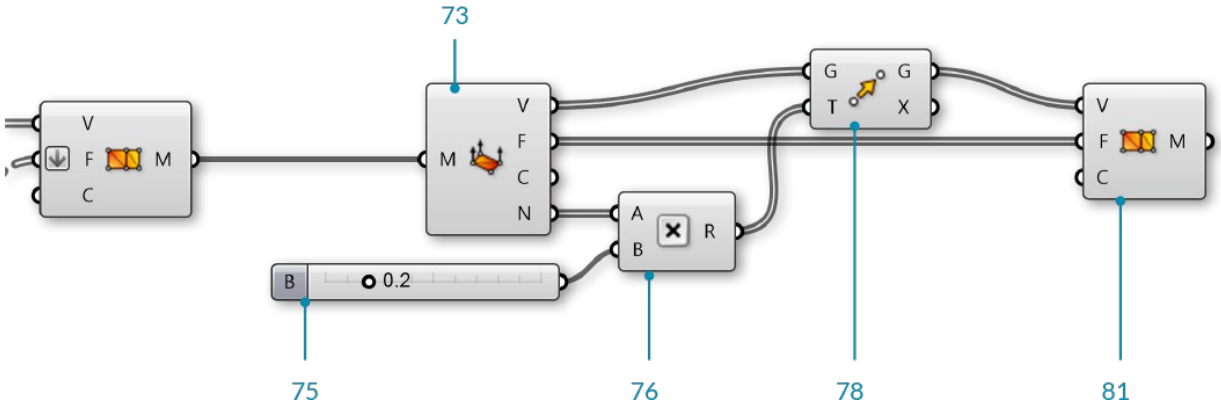


И вот уже у нашей вазы есть единая поверхность. Но, если бы мы хотели распечатать данную вазу на 3D-принтере, то пока не смогли бы сделать этого, так как для этого требуется иметь замкнутое твёрдое тело. И мы создадим из имеющегося объекта твёрдое тело путём смещения текущей полигональной сетки, а затем скомбинируем начальную и смещённую полигональные сетки вместе.

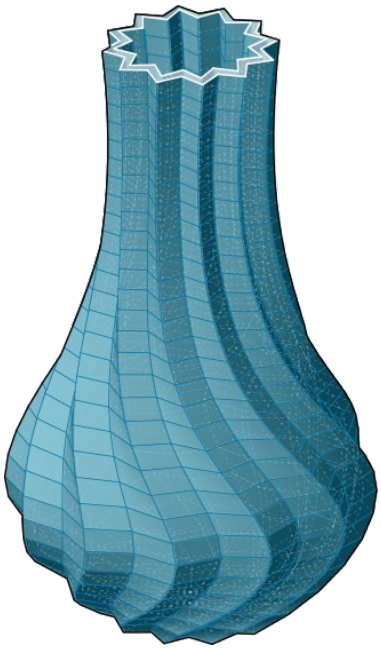
73.	**Mesh/Analysis/Deconstruct Mesh (Полигональная сетка/Анализ/Разобрать Полигональную сетку)** — Перетащите на холст компонент **Deconstruct Mesh (Разобрать Полигональную сетку)**	
74.	Подсоедините выход Mesh (M) (Полигональная сетка) компонента **Construct Mesh (Сконструировать Полигональную сетку)** ко входу Mesh (M) (Полигональная сетка) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)**	
75.	**Params/Input/Number Slider (Параметры/Вводные/Числовой Слайдер)** — Перетащите на холст компонент **Number Slider (Числовой Слайдер)**. Мы будем использовать его настройки-по-умолчанию	
76.	**Maths/Operator/Multiplication (Математика/Операторы/Умножение)** — Перетащите на холст компонент **Multiplication (Умножение)**	
77.	Подсоедините выход Normals (N) (Нормали) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу A и подсоедините **Числовой Слайдер** ко входу B компонента **Multiplication (Умножение)**	
78.	**Transform/Euclidean/Move (Трансформация/Евклидова/Перемещение)** — Перетащите на холст компонент **Move (Перемещение)**	
79.	Подсоедините выход Vertices (V) (Вершины) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу Geometry (G) (Геометрия) компонента **Move (Перемещение)**	
80.	Подсоедините выход Result (R) (Результат) комопнента **Multiplication (Умножение)** ко входу Motion (T) (Движение) компонента **Move (Перемещение)**	
81.	**Mesh/Primitive/Construct Mesh (Полигональная сетка/Примитивы/Сконструировать Полигональную сетку)** — Перетащите на холст компонент **Construct Mesh (Сконструировать Полигональную сетку)**	
82.	Подсоедините выход Geometry (G) (Геометрия) компонета **Move (Перемещение)** ко входу Vertices (V) (Вершины) комопнента **Construct Mesh (Сконструировать Полигональную сетку)**	

83.	Подсоедините выход Faces (F) (Грани) компонента **Deconstruct Mesh (Разобрать Полигональную сетку)** ко входу Face (F) (Грани) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	

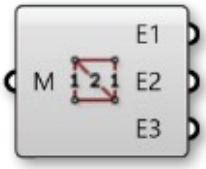
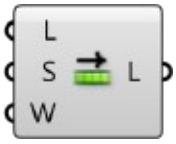


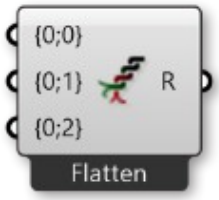


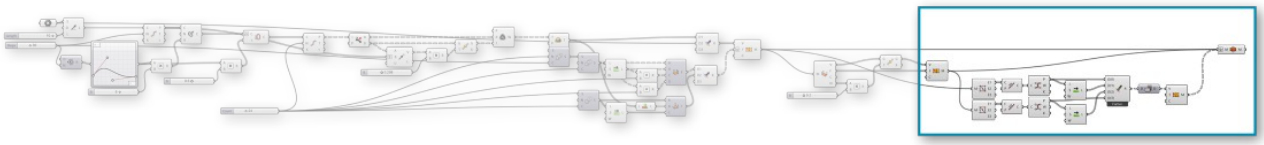
Создав сетку по отступу в соответствии с нормалями вершин, мы заимели «внутреннюю» и «наружную» полигональные сетки, но между ними всё ещё имеется брешь в верхней части требуемой полигональной геометрии.

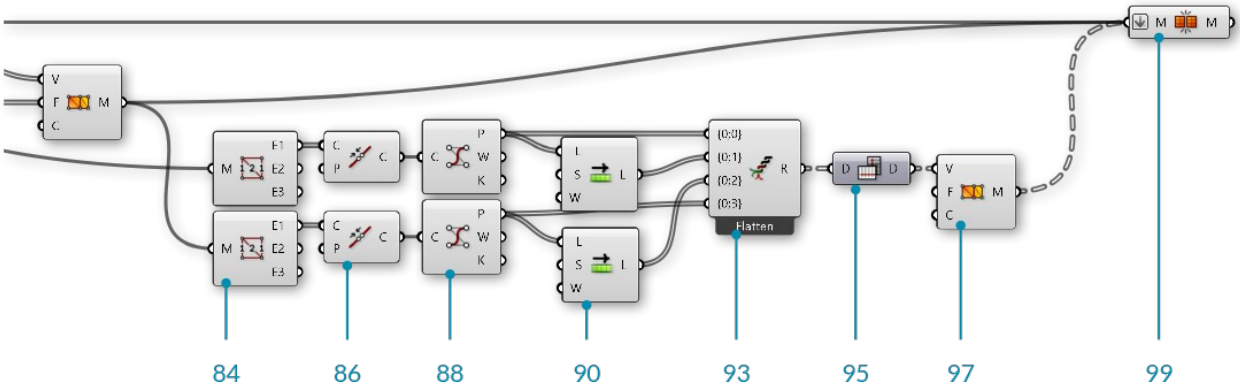


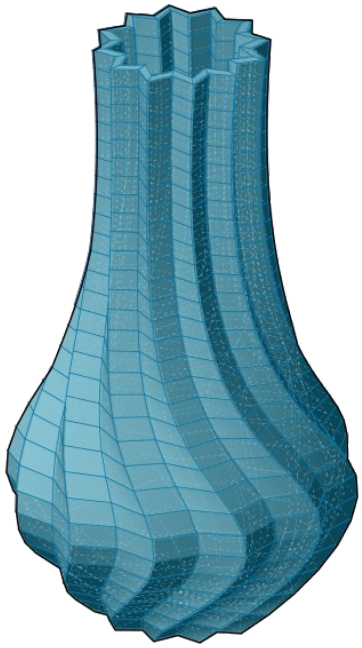
На заключительном этапе мы создадим замкнутую полигональную сетку, создав новую полигональную геометрию, закрывающую брешь, и объединив все эти полигональные сетки вместе.

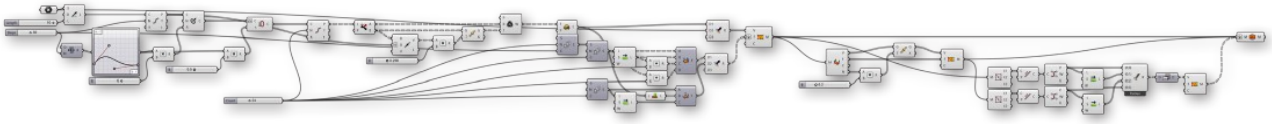
84.	**Mesh/Analysis/Mesh Edges (Полигональная сетка/Анализ/Края Полигональной сетки)** — Перетащите на холст компонент Mesh Edges **(Края Полигональной сетки)**	
85.	Подсоедините выход Mesh (M) (Полигональная сетка) первого компонента **Construct Mesh (Сконструировать Полигональную сетку)** ко входу Mesh (M) (Полигональная сетка) компонента **Mesh Edges (Края Полигональной сетки)**	
86.	**Curve/Util/Join Curves (Кривая/Утилиты/Объединить Кривые)** — Перетащите на холст компонент **Join Curves (Объединить Кривые)**	
87.	Подсоедините выход Naked Edges (E1) (Открытые Края) компонента **Mesh Edges (Края Полигональной сетки)** ко входу Curves (C) (Кривые) компонента **Join Curves (Объединить Кривые)**	
88.	**Curve/Analysis/Control Points (Кривая/Анализ/Контрольные Точки)** — Перетащите на холст компонент **Control Points (Контрольные Точки)**	
89.	Подсоедините выход Curves (C) (Кривые) компонента **Join Curves (Объединить Кривые)** ко входу Curve (C) (Кривая) компонента **Control Points (Контрольные Точки)** Объединив кривые, а затем, извлекая контрольные точки, мы гарантируем, что порядок точек соответствует вдоль обоих краёв вазы, что важно для создания результирующей полигональной сетки, правильно ориентированной и немногосложной (manifold)	
90.	**Sets/List/Shift List (Наборы/Список/Сдвиг Списка)** — Перетащите на холст компонент **Shift List (Сдвиг Списка)**	
91.	Подсоедините выход Points (P) (Точки) компонента **Control Points (Контрольные Точки)** ко входу List (L) (Список) компонента **Shift List (Сдвиг Списка)**	
92.	Повторите шаги с 84 по 91 для второго компонента **Construct Mesh (Сконструировать Полигональную сетку)**	

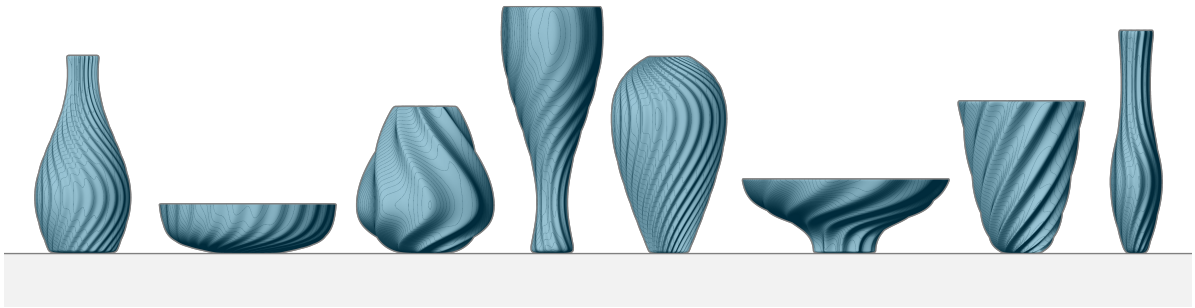
93.	**Sets/Tree/Entwine (Наборы/Деревья Данных/Сплести)** — Перетащите на холст компонент **Entwine (Сплести Набор Потоков Данных)**	
94.	Приблизьте компонент **Entwine (Сплести Набор Потоков Данных)**, чтобы отобразились опции добавления дополнительных входов. Нам понадобятся четыре входа. Подключите эти входы следующим образом: {0;0} — Points (P) (Точки) от первого компонента Control Points (Контрольные Точки) {0;1} — выход от первого Shift List (Сдвиг Списка) {0;2} — выход от второго Shift List (Сдвиг Списка) {0;3} — Points (P) (Точки) от второго компонента Control Points (Контрольные Точки)	
95.	**Sets/Tree/Flip Matrix (Наборы/Дерево Данных/Перевернуть Матрицу)** — Перетащите компонент **Flip Matrix (Перевернуть Матрицу)**	
96.	Подсоедините выход Result (R) (Результат) компонента **Entwine (Сплести Набор Потоков Данных)** ко входу Data (D) (Данные) компонента **Flip Matrix (Перевернуть Матрицу)**	
97.	**Mesh/Primitive/Construct Mesh (Полигональная сетка/Примитивы/Сконструировать Полигональную сетку)** — Перетащите на холст компонент **Construct Mesh (Сконструировать Полигональную сетку)**	
98.	Подсоедините выход Data (D) (Данные) компонента **Flip Matrix (Перевернуть Матрицу)** ко входу Vertices (V) (Вершины) компонента **Construct Mesh (Сконструировать Полигональную сетку)**	
99.	**Mesh/Util/Mesh Join (Полигональная сетка/Утилиты/Объединение Полигональных сеток)** — Перетащите на холст компонент **Mesh Join (Объединение Полигональных сеток)**	
100.	Подсоедините все три компонента Construct Mesh (Сконструировать Полигональную сетку), удерживая нажатой клавишу Shift при подключении проводов (или используйте компонент Merge (Слияние)). Кликните правой кнопкой мыши по входу Mesh (M) (Полигональная сетка) компонента Mesh Join (Объединение Полигональных сеток) и выберите 'Flatten' (Обрубить Дерево Данных)	




Flatten
84
86
88
90
93
95
97
99

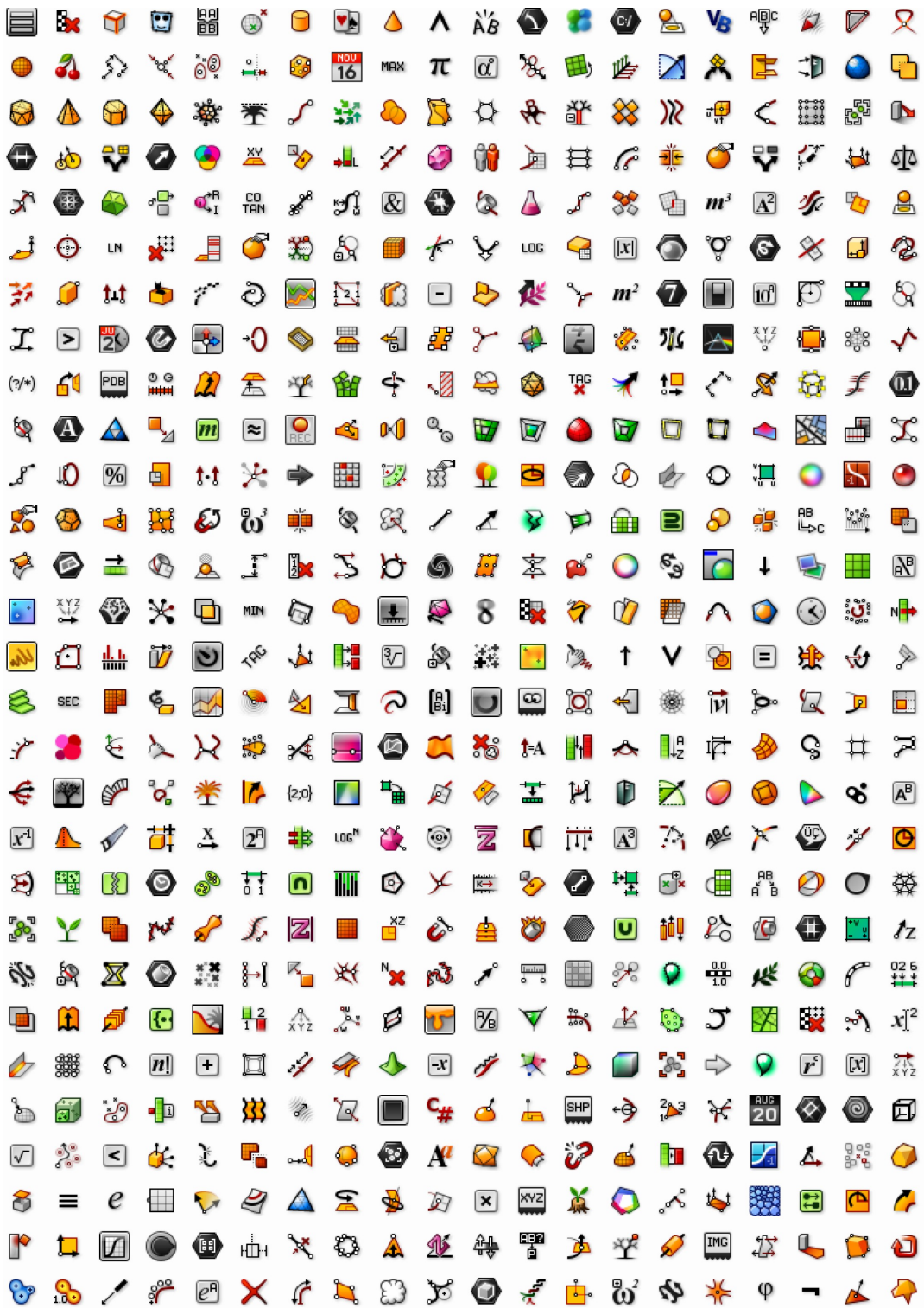






ПРИЛОЖЕНИЕ

Следующий раздел содержит полезные ссылки, в том чиле, индекс всех компонентов, используемых в этом учебнике, а также дополнительные ресурсы в помощь обучению Grasshopper.

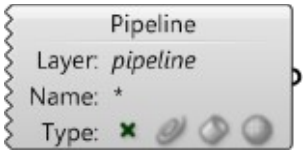


2.1. Индекс

Этот индекс предоставляет дополнительную информацию обо всех компонентах, используемых в данном учебнике, а также и о других компонентах, которые могли бы быть Вам полезны. Но это лишь только введение, ведь в плагине Grasshopper содержится более 500 компонентов.

Параметры (Parameters)

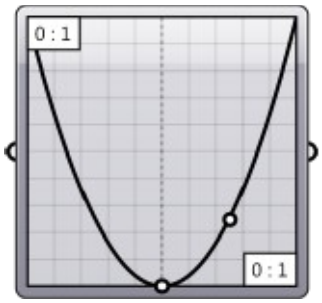
Geometry (Геометрия)

P.G.Crv	Параметр Curve (Кривая). Представляет набор геометрии Curve (Кривая). Геометрия Curve (Кривая) - это общий знаменатель всех типов кривых, используемых в Grasshopper.	
P.G.Circle	Параметр Circle (Окружность). Представляет набор примитивов Circle (Окружность).	
P.G.Geo	Параметр Geometry (Геометрия). Представляет набор 3D-Геометрии.	
P.G.Pipeline	Geometry Pipeline (Конвейер Геометрии). Позволяет геометрию, созданную в Rhino, автоматически передавать в Grasshopper.	
P.G.Pt	Параметр Point (Точка). Способен хранить постоянные данные. Вы можете записать постоянные данные, воспользовавшись контекстным меню параметра.	
P.G.Srf	Параметр Surface (Поверхность). Представляет набор геометрии Surface (Поверхность). Геометрия Surface (Поврехность) — общий знаменатель всех типов поверхностей в Grasshopper.	

Primitive (Примитивы)

P.P.Bool	Параметр Boolean (Логическое значение). Представляет набор Булевых (логических значений) (True/False (ИСТИНА/ЛОЖЬ).	
P.P.D	Параметр Domain (Домен). Представляет набор одномерных Доменов. Домены, как правило, используются для представления фрагментов кривой и непрерывных числовых диапазонов. Домен состоит из двух цифр, которые указывают пределы области, а все находящиеся между ними значения являются частью этого домена.	
P.P.D2	Параметр Domain^2 (Двумерный Домен). Содержит набор двумерных доменов. Обычно двумерные домены используются для представления фрагментов поверхности. Двумерный домен состоит из двух одномерных доменов.	
P.P.ID	Параметр Guid (Глобально-Уникальный Идентификатор). Represents a collection of Globally Unique Identifiers. Guid parameters are capable of storing persistent data. You can set the persistent records through the parameter menu.	
P.P.Int	Параметр Integer (Целое число). Представляет набор Целочисленных значений (Integer numeric values). Параметр Integer (Целые числа) способен сохранять постоянные данные. Вы можете установить постоянную запись через меню параметра.	
P.P.Num	Параметр Number (Число). Представляет набор значений с плавающей точкой. Параметр Number (Число) способен сохранять постоянные данные. Вы можете установить постоянную запись через меню параметра.	
P.P.Path	Параметр File Path (Путь Файла). Представляет набор путей файлов.	

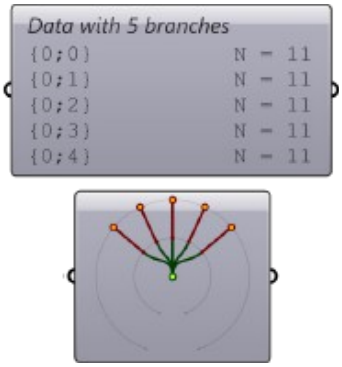
Input (Вводные)

P.I.Toggle	Boolean Toggle (Логический Переключатель). Переключатель логического значения (true/false (ИСТИНА/ЛОЖЬ))	
P.I.Button	Button (Кнопка). Объект Button (Кнопка) с двумя значениями. При клике по нему левой кнопкой мыши возвращает логическое значение True (ИСТИНА), а затем восстанавливается до состояния, возвращающее значение False (ЛОЖЬ)	
P.I.Swatch	Colour swatch (Образец Цвета). Образец (swatch) является особым объектом интерфейса, который позволяет быстро настроить отдельные цветовые значения. Вы можете изменить цвет образца через контекстное меню.	
P.I.Grad	Gradient Control (Регулятор Градиента). Регулятор градиента позволяет Вам определить цветовой градиент в цифровом диапазоне. По умолчанию используется диапазон значений (unit domain) (0.0 ~ 1.0), но он может быть настроен с помощью входных параметров L0 и L1. Вы можете добавить новый захват, кликая и перетаскивая из иконки спектра в левом верхнем углу и изменить цвет захвата, щёлкнув правой кнопкой мыши по нему.	
P.I.Graph	Graph Mapper (Переналожение по Графу). Объект Graph mapper (Переналожение по Графу) позволяет Вам переназначить набор чисел. По умолчанию домены {x} и {y} функции графа имеют единицы измерения домена (0.0 ~ 1.0), но это может быть настроено в Graph Editor (Редактор Графика). Graph mappers (Переналожение по Графу) может содержать единственную функцию построения, которая доступна через контекстное меню. Graphs (Графики) обычно имеют рукоятки (grips (маленькие кружки)), которые могут быть использованы для изменения переменных, определяющих график уравнения. По умолчанию, объекты graph mapper (Переналожение по Графу) не содержат графика и отображают значение перестроения 1:1.	

P.I.Slider	Number slider (Числовой слайдер). Слайдер – это специальный объект интерфейса, который позволяет быстро настроить индивидуальные числовые значения. Вы можете изменить значения и свойства через меню или по двойному клику по объекту слайдера. Слайдер можно сделать длиннее или короче, перетаскивая его правый край слева-направо. Отметьте, что слайдеры имеют только выходные захваты (output grips).	
P.I.Panel	Panel (Текстовая Панель). Панель для пользовательских заметок и текстовых значений. Это, как правило, неактивный объект, который позволяет добавить небольшие замечания или пояснения в документ. Панели могут также получать информацию из других источников. Если Вы подключите какой-нибудь выходной параметр к Панели, то Вы сможете видеть содержимое этого параметра в реальном времени. Этим способом в Grasshopper могут быть просмотрены любые данные. Панель может также записывать своё содержимое в текстовой файл.	
P.I.List	Value List (Список Значений). Предоставляет список заданных значений на выбор.	

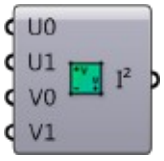
Utilities (Утилиты)

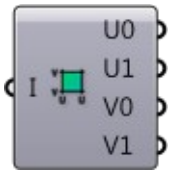
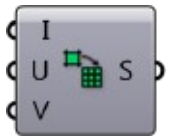
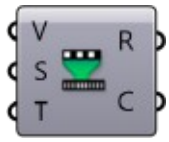
P.U.Cin	Cluster Input (Групповой Ввод). Представляет кластерный (совместный, групповой) ввод параметров.	
P.U.COut	Cluster Output (Групповой Вывод). Представляет кластерный (совместный, групповой) вывод параметров	
P.U.Dam	Data Dam (Задержка Данных). Задержка данных на их пути по документу.	
P.U.Jump	Jump (Прыжок). Быстрый переход между различными местоположениями.	

P.U.Viewer	Param Viewer (Просмотрщик Параметра). Средство просмотра структуры данных.	
P.U.Scribble	Scribble (Записка). Небольшое (быстрое) примечание.	Doubleclick Me!

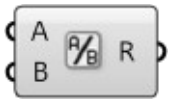
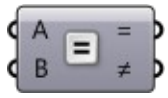
Maths (Математика)

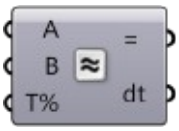
Domain (Домен)

M.D.Bnd	Bounds (Границы). Создаёт числовой домен, включающий в себя список чисел.	
M.D.Consec	Consecutive Domains (Последовательность Доменов). Создаёт из списка чисел последовательность доменов.	
M.D.Dom	Construct Domain (Сконструировать Домен). Создаёт числовой домен из двух числовых экстремумов.	
M.D.Dom2Num	Construct Domain² (Сконструировать Двумерный Домен). Создаёт из четырёх чисел двумерный домен.	
M.D.DeDomain	Deconstruct Domain (Разобрать Домен). Разбирает числовой домен на его составные части.	

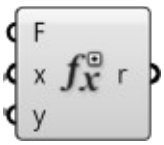
M.D.DeDom2Num	Deconstruct Domain² (Разобрать Двумерный Домен). Разбирает двумерный домен на четыре числа.	
M.D.Divide	Divide Domain² (Разделить Двумерный Домен). Делит двумерный домен на равные сегменты.	
M.D.Inc	Includes (Содержит?) Проверяет числовое значение, чтобы увидеть, включено ли оно в домен.	
M.D.ReMap	Remap Numbers (Переназначить Числа). Переназначает числа в новый числовой домен.	

Operators (Операторы)

M.O.Add	Addition (Сложение) Математическое сложение.	
M.O.Div	Division (Деление). Математическое деление.	
M.O.Equals	Equality (Равенство). Проверка двух чисел на (не)равенство.	
M.O.And	Gate And (Логический вентиль И). Выполняет логическое умножение (коньюкцию) (с логическим элементом AND (И)). Чтобы результат был True (ИСТИНА), оба вводных значения должны быть True (ИСТИНА).	
M.O.Not	Gate Not (Логический Вентиль НЕТ). Выполняет логическое отрицание (boolean negation) (с логический элементом NOT (НЕТ)).	

M.O.Or	Gate Or (Логический вентиль ИЛИ) Выполняет логическое сложение (дизъюнкцию) (с логическим элементом OR (ИЛИ)). Чтобы результат был True (ИСТИНА), оба вводных значения должны быть True (ИСТИНА).	
M.O.Larger	Larger Than (Больше Чем). Больше чем (или равно).	
M.O.Multiply	Multiplication (Умножение). Математическое умножение.	
M.O.Smaller	Smaller Than (Меньше Чем). Меньше чем (или равно).	
M.O.Similar	Similarity (Подобие). Проверка на подобие двух чисел.	
M.O.Sub	Subtraction (Вычитание). Математическое вычитание.	

Script (Скрипт)

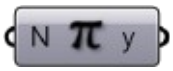
M.S.Eval	Evaluate (Вычисление). Вычисляет выражение с настраиваемым количеством переменных.	
M.S.Expression	Expression (Выражение). Вычисляет выражение.	

Trig (Тригонометрия)

M.T.Cos	Cosine (Косинус). Вычисляет косинус значения.	
M.T.Deg	Degrees (Градусы). Конвертирует значение угла, указанное в радианах (radians), в значение в градусах (degrees).	

M.T.Rad	Radians (Радианы). Конвертирует значение угла, указанное в градусах (degrees), в значение в радианах (radians).	
M.T.Sin	Sine (Синус). Вычисляет синус значения.	

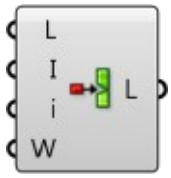
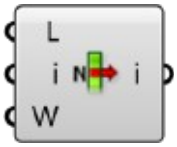
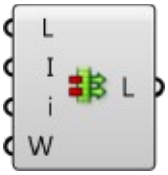
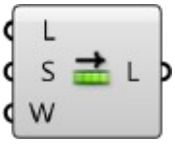
Utilities

M.U.Avr	Average (Среднее). Вычисляет среднее арифметическое набора значений.	
M.U.Phi	Golden Ratio (Золотое Сечение). Возвращает множитель золотого сечения (Phi).	
M.U.Pi	Pi (Пи). Возвращает множитель Пи (Pi).	

Sets (Наборы)

List (Список)

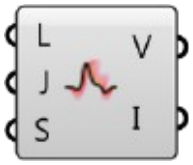
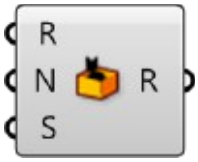
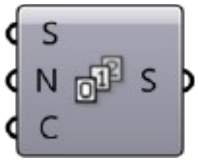
S.L.Combine	Combine Data (Комбинировать Данные). Комбинирует не-нулевые элементы, собираемые с нескольких входов.	
S.L.CrossRef	Cross Reference (Перекрёстная Ссылка). Делает Перекрёстную Ссылку данных из нескольких списков.	
S.L.Dispatch	Dispatch (Диспетчер). Перенаправляет элементы списка в два целевых списка. Диспетчеризация списка очень схожа с действием компонента [Cull Pattern] (Шаблон Отбрасывания), за исключением того, что оба списка предоставляются в качестве результирующих.	

S.L.Ins	Insert Items (Внедрить Элементы). Внедряет в список набор элементов.	
S.L.Item	List Item (Элемент Списка). Позволяет получить определённый элемент из списка.	
S.L.Lng	List Length (Длина Списка). Измеряет длину списка. Элементы в списке идентифицируются по их индексу. Первый элемент сохраняется с индексом ноль, второй элемент сохраняется с индексом один и так далее. Максимально большим индексом будет значение длины списка минус один.	
S.L.Long	Longest List (Длиннейший Список). Удлиняет набор списков по самому длинному.	
S.L.Split	Split List (Разбить Список) Разбивает список на раздельные части.	
S.L.Replace	Replace Items (Замена Элементов) Replace certain items in a list.	
S.L.Rev	Reverse List (Обратить Список). Переворачивает порядок в списке. Новый индекс каждого элемента будет N-i где N является самым верхним индексом в списке и i - это старый индекс элемента.	
S.L.Shift	Shift List (Сдвиг Списка). Смещает все элементы в списке. Элементы в списке будут смещены (перемещены) к концу списка, если смещение сдвига - положительное. Если параметр Оболочка (Wrap) находится в состоянии True (ИСТИНА), то элементы смещённые за пределы конца списка, будут применены повторно.	

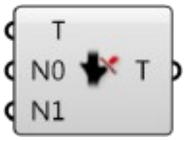
S.L.Short	Shortest List (Кратчайший Список) Обрезает набор списков по длине кратчайшего.	
S.L.Sift	Sift Pattern (Шаблон Отсеивания) Отсеивает элементы списка, используя повторяющийся шаблон индексов.	
S.L.Sort	Sort List (Сортировать Список). Сортирует список числовых ключей. Для того, чтобы что-либо могло быть отсортировано, это сначала должно стать сопоставимым. Большинство типов данных не сопоставимо, Числа (Numbers) и Строки (Strings) являются еднственным, пожалуй, исключением. Если Вы хотите отсортировать другие типы данных, таких как кривые, Вам, в первую очередь, необходимо составить список ключей.	
S.L.Weave	Weave (Сплести). Сплетает набор входящих потоков, используя пользовательский шаблон. Шаблон определяется как список индексов значений (целых чисел), который задаёт порядок, в котором данные будут собраны.	

Sets (Наборы)

S.S.Culli	Cull Index (Отбросить Индекс). Отбрасывает (удаляет) индексированные элементы из списка.	
S.S.Cull	Cull Pattern (Шаблон Отбрасывания). Отбрасывает (удаляет) элементы списка, используя повторяющуюся битовую маску (bit mask). Битовая маска определяется как список логических значений (Boolean values). Битовая маска будет повторно использоваться, пока все элементы в списке данных не будут оценены.	
S.S.Dup	Duplicate Data (Дублировать Данные). Дублирует данные определённое число раз. Данные могут быть дублированы двумя способами: либо копии списка добавляются в конце, пока не будет	

	достигнуто необходимое количество копий, либо каждый элемент будет продублирован несколько раз, прежде, чем перейти к следующему элементу.	
S.S.Jitter	Jitter (Дрожание). Перемешивает значения списка в случайном порядке. Исходный список переупорядочивается на основе случайного шума. Дрожание — хороший способ получить случайный набор с хорошим распределением. Параметр jitter (дрожание) устанавливает радиус случайного шума. Если jitter (дрожание) равен 0.5, то каждому элементу позволена самостоятельная перестановка в случайном порядке в пределах половины разброса всего набора.	
S.S.Random	Random (Случайный). Генерирует список псевдослучайных чисел. Последовательность чисел уникальна, но стабильна для каждого начального значения (seed value). Если вам не понравилось случайное распределение, попробуйте различные значения seed (начальной точки).	
S.S.Range	Range (Диапазон). Создаёт ряд чисел. Числа располагаются равномерно внутри числового домена. Используйте этот компонент, когда Вам необходимо создать числа между экстремумами. Если Вам необходим контроль над интервалами между последовательности чисел, используйте компонет [Series] (Серия).	
S.S.Repeat	Repeat Data (Повторение Данных). Повторяет шаблон, до достижения определённой длины.	
S.S.Series	Series (Серия). Создаёт серию чисел. Числа располагаются в соответствии со значением {Step} (Шаг). Если Вам необходимо распределить числа внутри фиксированного числового диапазона, рассмотрите возможность использования вместо этого компонента [Range] (Диапазон).	

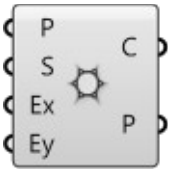
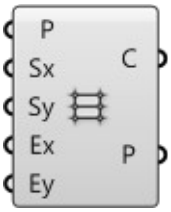
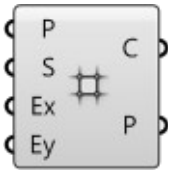
Tree (Дерево Данных)

S.T.Explode	Explode Tree (Разорвать Дерево). Извлекает из дерева данных все ветви.	
S.T.Flatten	Flatten Tree (Обрубить Дерево). Удаляет из дерева данных информацию о всех ветвях.	
S.T.Flip	Flip Matrix (Перевернуть Матрицу). Переворачивает матрицу данных, обменивая ряды и колонки.	
S.T.Graft	Graft Tree (Привить Дерево). Как правило, элементы данных сохранены в ветвях, как определённые значения индексов (0 для первого элемента, 1 для второго и так далее), а ветви сохраняются в дереве как определённые пути ветвления, например: {0;1}, которая указывает вторую подветвь первой основной ветви. Прививание создаёт новую ветвь для каждого элемента данных.	
S.T.Merge	Merge (Слияние). Слияние потоков данных в пучок.	
S.T.Path	Path Mapper (Сопоставитель Путей). Выполняет лексические операции над деревьями данных. Лексические операции — это операции логического сопоставления между путями данных и индексами, определяемого текстовыми (лексуальными) масками и шаблонами.	
S.T.Prune	Prune Tree (Подрезать Дерево). Удаляет все ветви из Дерева, имеющие особое число Элементов данных. Вы можете установить как верхний, так и нижний пределы обрезки ветвей.	
S.T.Simplify	Simplify Tree (Упростить Дерево). Упрощает дерево, удаляя перекрытие (overlap), общее для всех ветвей.	

S.T.TStat	Tree Statistics (Статистика Дерева) Выдаёт некоторые статистические данные о дереве данных.	
S.T.Unflatten	Unflatten Tree (Восстановление Дерева). Убирает последствия уплощения (Flatten Tree (Обрубить Дерево)) дерева данных, перемещая (возвращая) элементы на ветви.	

Vector (Вектор)

Grid (Сетка)

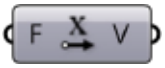
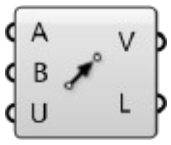
V.G.HexGrid	Hexagonal (Шестигранная сетка). Двумерная сетка с шестигранными ячейками.	
V.G.RecGrid	Rectangular (Прямоугольная сетка). Двумерная сетка с прямоугольными ячейками.	
V.G.SqGrid	Square (Сетка квадратов). Двумерная сетка с квадратными ячейками.	

Point (Точка)

V.P.Pt	Construct Point (Сконструировать Точку). Конструирует точку по {xyz}-координатам .	

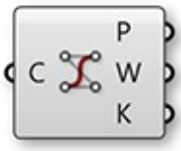
V.P.pDecon	Deconstruct (Разобрать Точку). Разбирает точку на составные части.	
V.P.Dist	Distance (Расстояние). Вычисляет Евклидово расстояние между двумя координатами точек.	

Vector (Вектор)

V.V.X	Unit X (Унифицированный X-вектор). Унифицированный вектор, параллельный оси {x} в глобальной системе координат (world).	
V.V.Y	Unit Y (Унифицированный Y-вектор). Унифицированный вектор, параллельный оси {y} в глобальной системе координат (world).	
V.V.Vec2Pt	Vector 2Pt (Вектор по 2м Точкам). Создаёт вектор между двумя точками.	

Curve (Кривая)

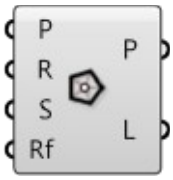
Analysis (Анализ)

C.A.CP	Control Points (Контрольные Точки). Извлекает контрольные точки и узлы (knots) NURBS-кривых.	

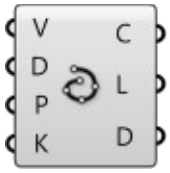
Division (Деление Кривой)

C.D.Divide	Divide Curve (Разделить Кривую). Делит кривую на сегменты равной длины.	

Primitive (Примитивы)

C.P.Cir	Circle (Окружность). Создаёт окружность, определяемую базовой плоскостью и радиусом.	
C.P.Cir3Pt	Circle 3Pt (Окружность по 3-м Точкам). Создаёт окружность, определяемую тремя точками.	
C.P.CirCNR	Circle CNR (Окружность CNR (Центр, Нормаль, Радиус)). Создаёт окружность, определяемую по центру, нормали и радиусу.	
C.P.Line	Line SDLLine SDL (Прямая SDL (Начало, Касательная (направление), Длина)) Создаёт отрезок прямой по точке начала (start point), касательной (tangent) и длине (length).	
C.P.Polygon	Polygon (Многоугольник). Создаёт многоугольник (опционально: с закруглёнными углами).	

Spline (Сплайны)

C.S.IntCrv	Interpolate (Интерполировать). Создаёт кривую, интерполированную через набор точек.	
C.S.KinkCrv	Kinky Curve (Кривая с Загибом). Создаёт кривую, интерполированную через набор точек с пороговым углом излома.	

C.S.Nurbs	Nurbs Curve (NURBS-Кривая). Создаёт из контрольных точек NURBS-кривую.	
C.S.PLine	PolyLine (Ломаная). Создаёт ломаную линию, соединяющую точки в наборе.	

Util (Утилиты)

C.U.Explode	Explode (Разорвать) Разрывает кривую на меньшие сегменты.	
C.U.Join	Join Curves (Объединить Кривые) Объединяет множество кривых в наименьшее возможное количество кривых	
C.U.Offset	Offset (Кривая по Отступу) Создаёт новую кривую, отступая от базовой на указанную дистанцию.	

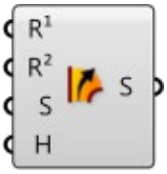
Surface (Поверхность)

Analisis (Анализ)

S.A.DeBrep	Deconstruct Brep (Разобрать Brep) Разбирает brep (геометрию, представленную ограничивающими поверхностями) на составные части.	

Freeform (Произвольные формы)

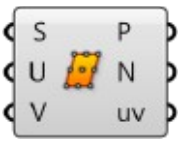
S.F.Boundary	Boundary Surfaces (Поверхность по Границам) Создаёт плоскую поверхность из набора	

	кривых — краёв границ.	
S.F.Extr	Extrude (Экструзия) Экструдирует (выдавливает) поверхности из кривых и поверхностей вдоль вектора.	
S.F.ExtrPt	Extrude Point (Экструзия в Точку) Экструдирует поверхности из кривых и поверхностей в точку.	
S.F.Loft	Loft (Лофтинг) Создаёт поверхность лофтингом через набор секущих кривых.	
S.F.RevSrf	Revolution (Поверхность Вращения) Создаёт поверхность вращения.	
S.F.Swp2	Sweep2 (Протягивая по 2м Направляющим) Создаёт поверхность, протягивая кривую профиля по двум кривым направляющих.	

Primitive (Примитивы)

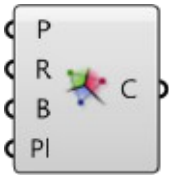
S.P.BBox	Bounding Box (Габаритный Контейнер) Вычисляет ориентированный параллелепипед, ограничивающий геометрию.	

Util (Утилиты)

S.U.SDivide	Divide Surface (Делить Поверхность). Генерирует сеть {uv} точек на поверхности.	
S.U.SubSrf	Isotrim (Обрезка по Изопарме). Извлекает изопараметрический субнабор (isoparametric subset) из поверхности.	

Mesh (Полигональная сетка)

Triangulation (Триангуляция)

M.T.Voronoi	Voronoi (Диаграмма Вороного). Создаёт из набора точек плоскую диаграмму Вороного.	

Transform (Трансформация)

Affine (Аффинная)

T.A.RecMap	Rectangle Mapping (Прямоугольное Наложение). Трансформирует геометрию, ориентируясь на изменение формы ссылочных прямоугольников: от одного к другому.	

Array (Массив)

T.A.RecMap	Linear Array (Линейный Массив). Создаёт линейный массив геометрии.	

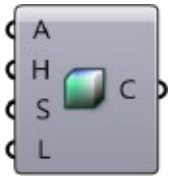
Morph (Морфинг)

T.A.RecMap	Linear Array (Линейный Массив). Создаёт линейный массив геометрии.	

T.M.SBox	Surface Box (Контейнер на Поверхности). Создаёт искривлённый параллелепипед по участку поверхности.	

Display (Отображение)

Color (Цвет)

D.C.HSL	Colour HSL (Цвет HSL). Создаёт цвет из плавающей точки каналов {HSL}.	

Dimensions (Размерности)

D.D.Tag	Text tags (Текстовые теги). Компонент text tag (текстовая тег) позволяет Вам отрисовывать небольшие Строки (Strings) во вьюпорте (viewport) в качестве обратной связи элементов (feedback items). Текст и местоположение определяются вводными параметрами. Когда текстовые теги запекаются (baked), то превращаются в Точку Аннотации (Text Dots).	
D.D.Tag3D	Text Tag 3D (Текстовой Тег 3D). Представляет список 3D текстовых тегов во вьюпорте Rhino.	

Preview (Предварительный Просмотр)

D.P.Preview	Custom Preview (Пользовательский Предварительный Просмотр). Позволяет осуществлять настраиваемый предварительный просмотр геометрии.	

Vector (Вектор)

D.V.Points	Point List (Список Точек) Отображает подробности о списке точек.	

2.2. Grasshopper-файлы Примеров.

Эти файлы примеров сопутствуют данному учебнику (Grasshopper Primer) и организованы в соответствии с его 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 для того, чтобы узнать больше о Grasshopper и параметрической концепции дизайна. Также разработано более ста плагинов и аддонов для расширения функционала Grasshopper. Ниже приведены некоторые из наших фаворитов.

Сообщество вокруг плагинов

food4Rhino (WIP (разработка в процессе работы)) — это новое сервис-сообщество МакНила. Там пользователь может найти новейшие плагины для Rhino, аддоны (дополнения) для Grasshopper, текстуры, фоны, добавить свои комментарии, подискутировать о новых инструментах, пообщаться с разработчиками этих приложений, поделиться своими скриптами. http://www.food4rhino.com/

Страница дополнений к Grasshopper:
http://www.grasshopper3d.com/page/addons-forgrasshopper

Наши любимые аддоны

DIVA. (для Rhino). Позволяет пользователю осуществлять ряд оценок показателей, относящихся к окружающей среде как для отдельных зданий, так и городских пейзажей в целом. http://diva4rhino.com/

Element — это плагин к Grasshopper для работы с mesh-геометрией, включающий создание полигональных сеток (mesh) их анализ, трансформации, разделение и сглаживание.
http://www.food4rhino.com/project/element

Firefly предлагает набор комплексных программных средств для совместной работы Grasshopper с микроконтроллером Arduino.
http://fireflyexperiments.com

GhPython — это компонент-интерпритатор Python для Grasshopper, который позволяет динамически выполнять скрипты любого типа. В отличие от других скриптовых компонентов, GhPython позволяет использовать синтаксис rhino-скриптов, чтобы начать написание скриптов без необходимости быть программистом.
http://www.food4rhino.com/project/ghpython

HAL — это плагин к Grasshopper для программирования промышленных роботов, поддерживающих ABB, KUKA, а также для машин Universal Robots.
http://hal.thibaultschwartz.com/

HUMAN расширяет возможности Grasshopper для создания ссылочной геометрии, включая источники света, блоки и текстовые объекты. Также обеспечивает доступ к материалам, слоям, типам линий и других настроек активного Rhino-документа.
http://www.food4rhino.com/project/human

Karamba — это программа для интерактивного анализа параметрических конечных элементов. Позволяет анализировать реакцию трёхмерных балочных и оболочных конструкций на различные нагрузки.
http://www.karamba3d.com/

Kangaroo — это движок Реальной Физики для интерактивной симуляции, оптимизации и поиска формы непосредственно в Grasshopper.
http://www.food4rhino.com/project/kangaroo

KingKong. Сгибает панели, используя складывание по кривым и

распределение контрольных панелей по поверхности с применением диапазона систем аттракторов. http://www.food4rhino.com/project/robofoldkingkong

LunchBox — это плагин к Grasshopper исследования математических форм, их панелирования, структурирования и автоматизации рабочего процесса.
http://www.food4rhino.com/project/lunchbox

Meshedit — это набор компонентов, расширяющих возможности Grasshopper для работы с полигональными сетками.
http://www.food4rhino.com/project/meshedittools

PanelingTools. Параметрические инструменты для создания и манипулирования прямоугольными сетками, аттракторами. Также поддерживает креативный морфинг параметрических паттернов.
http://www.food4rhino.com/project/pt-gh

Platypus позволяет авторам Grasshopper-геометрии настроить поток геометрии непосредственно в интернет в режиме реального времени. Это работает как комната чата для параметрической геометрии и позволяет «на лету» использовать 3D-модель в гибридных веб-приложениях непосредственно в веб-браузере.
http://www.food4rhino.com/project/platypus

TT Toolbox предоставляет диапазон различных инструментов, которые мы используем на регулярной основе в Core Studio, находящейся в Thornton Tomasetti и мы думаем, что некоторые из Вас могли бы высоко оценить их.
http://www.food4rhino.com/project/tttoolbox

Weaverbird — топологическое средство моделирования, содержащее множество из известных операций подразбиения и трансформации, охотно используемых дизайнерами. Этот плагин реконструирует форму и подразбиение любых полигональных сеток, даже созданных по полилиниям и помогает подготовить изделие к производству.
http://www.giuliopiacentino.com/weaverbird/

Дополнительные учебные материалы

Firefly Primer. Эта книга предназначена для обучения основам электроники (используя Arduino), а также различным цифровые/физические методы прототипирования, применяемым в данной области. Это не всеобъемлющая книга по электронике (поскольку уже имеется многочисленный ряд крупных ресурсов, посвящённых этой теме). Вместо этого данная книга сосредотачивается на ускорении процесса прототипирования. Автор: Andrew Payne.
http://fireflyexperiments.com/resources/

Essential Mathematics. Essential Mathematics (Математические Основы) используют Grasshopper, чтобы предоставить профессиональным дизайнерам основные математические концепции, знание которых необходимо 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, направленных на развитие различных концепций в области генеративных алгоритмов и параметрического проектирования. Автор: Zubin Khabazi.
http://www.morphogenesism.com/media.html

Rhino Python Primer. Этот учебник предназначен для обучения программированию абсолютных новичков, для людей, которые блуждали в тематике программирования совсем немного или опытных программистов, ищущих быстрое введение в методы Rhino. Автор: Skylar Tibbits.
http://www.rhino3d.com/download/IronPython/5.0/RhinoPython101

Основные ссылки

Wolfram MathWorld. Мир Математики — интернет-ресурс для математиков, который собрал Eric W. Weisstein при содействии нескольких тысяч пользователей. После того, как эти данные впервые появились онлайн в 1995 году, MathWorld стала связующим звеном знаний по математике в обоих сообществах: и математических и образовательных. Его записи широко упоминаются в журналах и книгах, охватывающих все уровни образования.
http://mathworld.wolfram.com/

Дополнительная литература

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 Daril 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.

Об этом учебнике

Авторы

Gil Akos, Mode Lab

Gil Akos является одним из основателей, а также техническим директором Mode Lab — многопрофильной проектной консалтинговой компании, специализирующейся в области инновационных технологий управления процессами. Он привносит разносторонний профессиональный опыт, технические экспертные знания цифровых платформ и страсть к генеративному дизайну для сервиса моделирования в студии. Его личные интересы охватывают взаимоотношения между моделированием и путями материализации смоделированного, с помощью которых эта связь может быть сделана физически ощутимой.
http://modelab.is
http://modelab.is/education

Ronnie Parsons, Mode Lab

Ronnie Parsons является одним из основателей и директором по вопросам образования в Mode Lab, специализирующейся в области инновационных технологий управления процессами. В Mode Lab, Ronnie определяет новые способы сконфигурировать и объединить рабочие процессы клиента, стратегически выравнивая видение продукта с технологическими платформами, сосредоточенными на накапливании пользовательского опыта. Компетенция Ronnie концентрируется в областях передового вычислительного моделирования, обучения проектированию, а также исследований и разработки.
http://modelab.is
http://modelab.is/education

MODELAB

Команда Mode Lab:

Sharon Jamison
Andrew Reitz
Armon Jahanshahi
Luis Quinones
Erick Katzenstein
Kimberly Parsons
Roberto Godinez
Christopher Morse

Соавтор

Andrew Payne. Руководитель Lift Architects.

Andrew Payne — квалифицированный архитектор, основавший LIFT Architects в 2007 году. Andrew исследует встраиваемые вычислительные системы, умные здания и генеративный дизайн, а также он опубликовал ряд статей и провёл обучающие семинары по всей Северной Америке и Европе. В 2010 году, Andrew и Jason K. Johnson опубликовал Firefly — набор комплексных программных средств для совместной работы Grasshopper с микроконтроллером Arduino, интернетом, аудио/визуальными инструментами и многим другим.
http://www.liftarchitects.com/

Этот учебник для начинающих предоставляет подробное руководство по самой последней на данный момент сборке Grasshopper (версии 0.90076), выделяя то, что нам кажется, самыми из захватывающих особенностей данного обновления. И наша цель — сделать этот учебник полевым справочником для новых и опытных пользователей, желающих ориентироваться в тонкостях использования Grasshopper в их творческой практике.

MODELAB

Mode Lab - многопрофильная проектная консалтинговая компания, специализирующаяся в области инновационных технологий управления процессами. С момента своего создания Mode Lab была местом для экспериментов с методами и технологиями, используемыми для проектирования и создания окружающего нас мира. Мы добиваемся лучшего понимания и улучшения процесса материализации идей — этот путь мы проходим в сотрудничестве с нашими клиентами.
http://modelab.is

MODELAB
EDUCATION

Наш бренд **Mode Lab Education** предоставляет смешанные обучающие решения для потребителей и компаний, стремящихся к продвижению. Мы проектируем и разрабатываем целевые обучающие методы повышения опыта, которые удерживают обучаемого в центре каждой техники проектирования и метода разработки, а затем делимся этим с нашим сообществом.
http://modelab.is/education

McNeel является компанией по разработке программного обеспечения с продажей его по всему миру, а также осуществляющая техподдержку и профессиональную подготовку. Основанная в 1980 году, McNeel — частная

компания, находящаяся в собственности сотрудников, имеющая офисы продажи и поддержки, а также филиалы в Сиэтле, Бостоне, Майами, Буэнос-Айресе, Барселоне, Риме, Токио, Тайбэ, Сеуле, Куала-Лумпуре и Шанхае с более чем 700-ми торговыми посредниками, дистрибьюторами, производителями оригинального оборудования и учебными центрами по всему миру.
http://www.en.na.mcneel.com/

Grasshopper — это графический редактор алгоритмов, тесно интегрированный с инструментами моделирования Rhino 3D для дизайнеров, ищущих новые формы, используя генеративные алгоритмы. В отличие от RhinoScript, Grasshopper не требует никаких знаний программирования или написания сценариев, но, тем не менее, позволяет дизайнерам создавать генерируемые формы, от простейших, до вызывающих благоговейный трепет.
http://www.grasshopper3d.com/

ИНФОРМАЦИЯ О ЛИЦЕНЗИРОВАНИИ