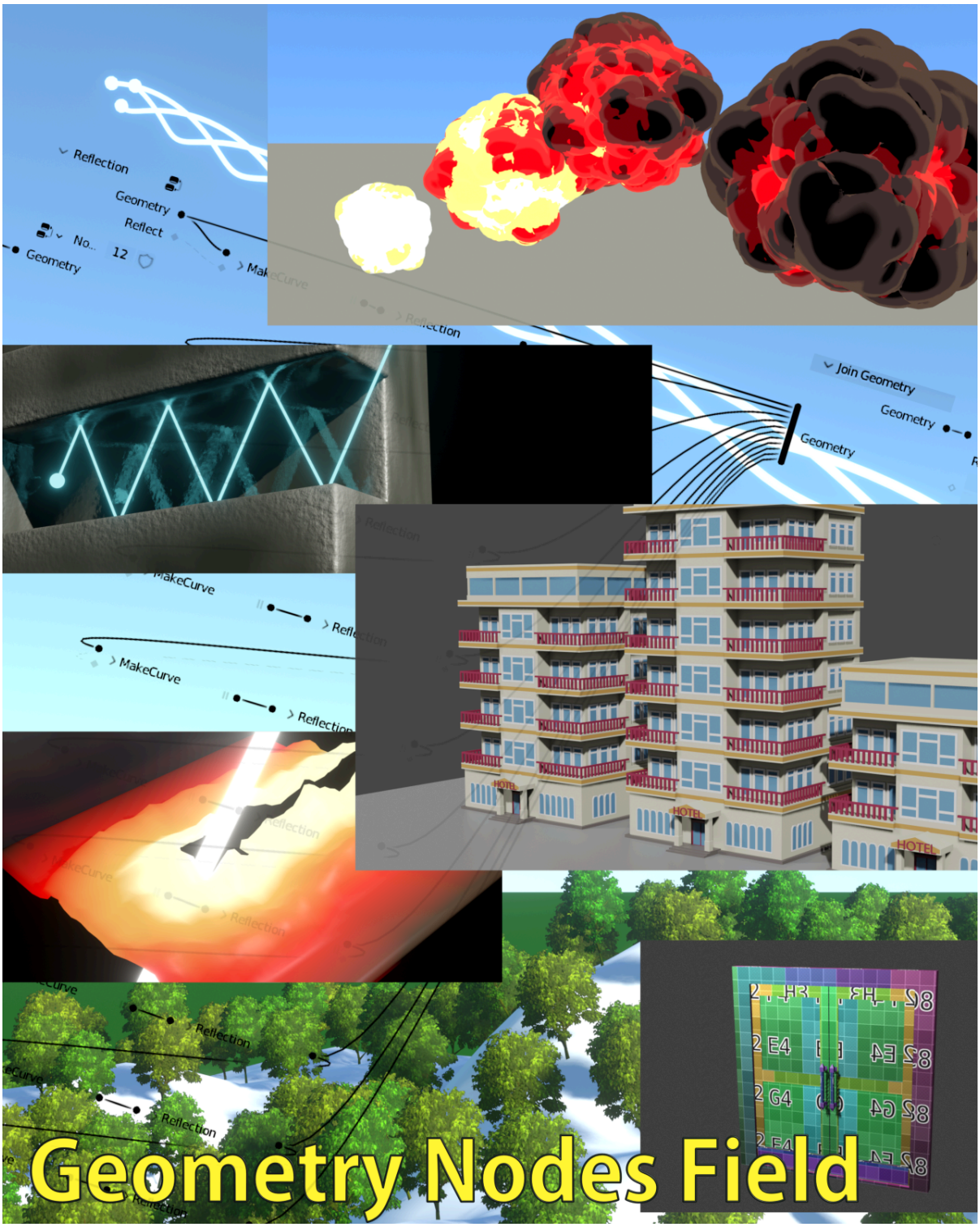
Blender ジオメトリノード 解説＆作例集 Vol.1

Geometry Nodes (for Blender 3.2 or later)

Version 2.0

Q@スタジオぽぷり
@popqjp

作者 Twitter アカウント
Q@スタジオぽぷり



目次

Blender ジオメトリノード 解説＆作例集 Vol.1

Geometry Nodes (for Blender 3.2 or later)

作者 Twitter アカウント
Q@スタジオぽぷり

2021.12.04 ver 1.0	2022.10.14 ver 1.5 Blender 3.3 への言及追加。サンプルを幾つか更新。
2021.12.24 ver 1.1 (UV関係の説明を更新しました)	Vol.2 の公開に合わせて Vol.1 と改題
2022.01.29 ver 1.2 (サブセクションを3つ追加しました)	2022.12.18 ver 1.6 Blender 3.4 への言及など追加。サンプルを幾つか更新。
2022.03.11 ver 1.3 (Blender 3.1 で増えたノード用に追記)	2023.04.06 ver 1.7 Blender 3.5 への言及など追加。サンプルを幾つか更新。
2022.03.14 ver 1.31 (一様回転に関するノードの誤りを修正)	2023.07.09 ver 1.8 Blender 3.6 への言及など追加。サンプルを１つ追加。
2022.05.15 ver 1.32 (Extrude Mesh 記述修正。3.2に関して軽い追記)	2023.11.19 ver 1.9 Blender 4.0 への言及など追加。サンプルを幾つか更新。
2022.06.14 ver 1.4 Blender 3.2 の仕様変更部分に関して一部修正	2024.03.28 ver 2.0 Blender 4.1 への言及など追加。サンプルを１つ追加。

初めに

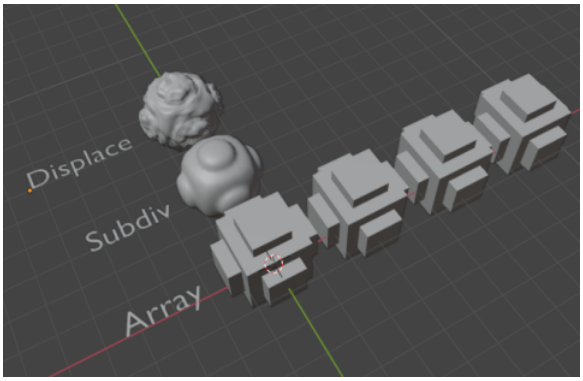
ジオメトリノードは、Blender 2.92 で導入された機能で、ノードベースでオブジェクトを操作することができます。
基本的にメッシュの形状を操ることが主要な機能で、積極的に開発が進められていて、Blender のバージョンが上がるごとに様々な機能が追加されていています。

ここでは Blender 3.1 までの範囲を中心にしてジオメトリノードでできることを、様々な作例とともに解説していこうと思います。
また、3.2以降で仕様の変更された部分に関しては、更新によってできるだけ新しいものに即して記述したいと思います。
ver 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0 でそれぞれ、Blender 3.2, 3.3, 3.4, 3.5, 3.6, 4.0, 4.1 についての言及を少量追加をしました。
ver 1.6以降のサンプルは Blender 3.4以降で作成したため、Blender 3.3以下ではノードを差し替えないと機能しないものがあります。
ver 1.9以降のサンプルは Blender 4.0以降で作成したため、Blender 3.6 では開けますが、それ以前の Blender では開けません。
※)Blender 4.0以降のポリシーは、前のメジャーバージョンの最後のLTSで開けるような互換性を維持するというものです。
3.6 で開いて保存したファイルは、さらに遡って3.5以前で開ける事になります(正常に動作するかは保証できませんが…)

読者には、ある程度 Blender の操作に慣れている人を想定しています。
簡単な注意点などは途中で説明を挟んでいきますが、基本的な操作法などは既に理解しているものとして説明していく点は注意してください。

モディファイア

ジオメトリノードの基本は、モディファイアとしての利用です。
モディファイアは、元になるメッシュの形状を保ったまま（非破壊で）変形させるための機能です。



例えば、もっとも良く使われるモディファイアである

Array(配列)、
Subdivision Surface(サブディビジョンサーフェス)、
Displace(ディスプレイス)

を単純な形に対して追加すると、このように（元の形状のデータはそのまま保ったままで）変形ができます。

これらのモディファイアを後から取り除けば、オブジェクトは元の形状に戻るので「非破壊」というわけです。

普通のモディファイアは、大抵１つの機能を実行するようになっています。
ジオメトリノードは、ノードベースで様々な機能を作ることができる、巨大で詳細なカスタマイズができるモディファイアと考えることができます。

フィールド

ジオメトリノードの基本操作は、2.93から3.0でかなり変更されています。
昔の方式でのやり方は3.2で完全に取り除かれています。
以前のジオメトリノードを使っていた人は少し気を付ける必要があります。
3.2 では代わりに旧ジオメトリノードの仕組みに「似た」運用をするためのノード(名前付き属性)が追加されています。

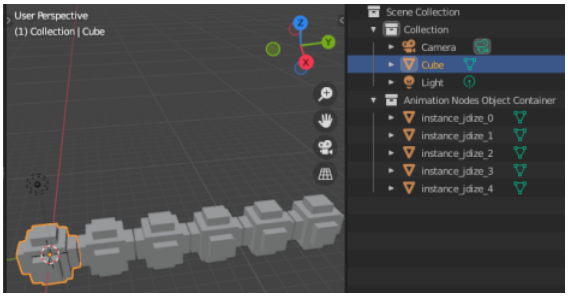
新しい方式は、シェーダーのノードと似たノード構成になるように設定されています。
古いやり方と区別するためにフィールドという用語が使われています。
過渡期のジオメトリノードの記事などで、Geometry Node Fields というような記述になっていれば、新しいほうの仕組みのジオメトリノードの記事、という事が分かります。
といっても、作例無しに説明を読んでもなんのこっちゃというところです。
フィールド方式になったノードの使い方は、本文でゆっくりと見てみましょう。

アニメーションノード

Blender でノードベースで様々な効果を実現する Add-on として有名なものに Animation Nodes があります。
名前も、概念も似ているので、「ジオメトリノードと何が違うのか」分かりづらいところなので、軽く説明しておきます。

一部で重なっている機能がありますが、Animation Nodes の方はその名の通り、時間による変化などのアニメーションを意識した作りになっています。
また、実行できる操作の対象が広く、その分複雑な作りになっています。

例えば、物を沢山配置するような効果を作る場合、ジオメトリノードはモディファイアなので、１つのオブジェクトとして複数の形状を作る形になります。
アニメーションノードでは（少し頑張れば同じようなことをすることも可能ですが）、オブジェクトを複数配置するといったような実現方法になります。



配列モディファイアのように、メッシュが並んで１つのオブジェクトを作るのではなくて、オブジェクトの複製が自動で作られています。
（これをノードで制御して動かして、アニメーションにする、といったことができるわけです）

Blender は今後、どんどんノードで制御ができる範囲を増やしていく方向性のようです。
Geometry Nodes もその流れの中で開発されて、Blender 3.x の間に多くの新規ノードが追加されました。
アニメーションノードの持つ機能も、いずれ Blender 本体に統合されていくという方針のようです。

今の時点では アニメーションノードは、ジオメトリノードよりも広い範囲を対象とした（似ているけど）別物と考えてよいと思います。
ジオメトリノードは、アニメーションノードほど複雑ではないですがその分扱いやすく、 Blender のネイティブな機能として実装されていますから、より本体と密接に結び付いていて高速です。

本文中のキャプチャ画像や動画

この本の中のノード名や操作名は、長すぎたりしない場合はできるだけ英語(日本語)という形で記述します。
キャプチャ画像については、基本英語を中心にしますが、幾つか日本語でキャプチャ画像も混ざる形になります。

本書に埋め込んである画像の一部は GIF アニメーションになっています。
.pdfとして書き出したファイルは、残念ながら静止画として最初のフレームが使われているだけなのですが、html版はブラウザーで見れば動いて見えるはずです。

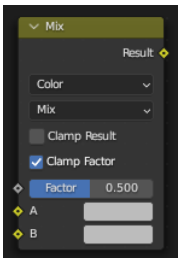
Blender 3.2 以降での大きな仕様の変更

Blender 3.2 以降も新しいノードはどんどん追加されていますが、その他従来のノードや機能が廃止されて、新しいものに置き換わるような更新もいくつか行われています。
そうした変更点は注意しなければいけないので、本書では出来るだけアップデートで対応をしようと考えています。

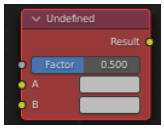
ジオメトリノードでは、各頂点について位置や色のパラメーターを弄ります。
それらのパラメーターを Attribute (アトリビュート) と呼んでいます。
昔からある Vertex Color (頂点カラー) や UV情報などのデータは、Attribute とは別の似て非なるデータになっていたのですが Blender 3.2 以降汎用の Attribute の一種として順次統合されています。
それに伴って、UV情報や旧頂点カラー関連の挙動が少し変化したところがあります。
また、Blender 3.2 での名前付き属性の導入も使い勝手の向上に大きく影響しています。

ノードがいくつか変更

Blender 3.4 では、Tansfer Attribute(属性転送)が廃止になり、新しいノード(Sample Nearest Surface(最近接面サンプル)などがその役目を引き継いでいます。



新機能ではないのですが、Mix Color が拡張されて、色以外にもベクトルや数値も扱えるようになっています。

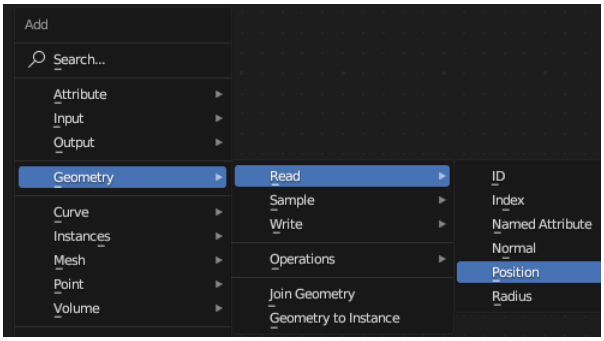


ところが、その代償というか、別ノードになってしまったためにBlender3.4で保存されたファイルは、
Blener3.3で開くとノードがNot Foundで無効になってしまいます。

バージョンを遡って.blendファイルを読み込むのは元から推奨されていませんが、この変更は影響範囲が大きいので特に要注意です。

メニューと名前の更新

Blender 3.5 では、機能(ノードの種類)の増加に伴って、ノードの追加メニューが再編成されています。



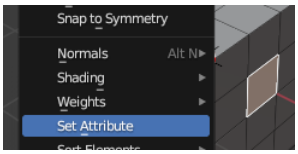
Geometry や Mesh, Curve といった大カテゴリーから、Read, Write 等分類、そして個々の項目と
3階層が基本のメニュー構成になっています。

また、一部ノードの名前やソケット名が変更になっています。
Transform > Transform Geometry
Field at Index > Evaluate at Index
Interpolate Domain > Evaluate on Domain
また、Group Index という名のソケットは、Group ID という名前に変更されています。

本文中のメニューアイテムへの言及は、できるだけ3.5以降の最新に合わせるように修正をしていますが、旧来の表記が残っている箇所もあるかもしれないのでご了承ください。

アトリビュート編集

また、Blender 3.5での機能追加として、専用のGUIなどが伴わない簡易的なアトリビュートの編集機能が付きました。

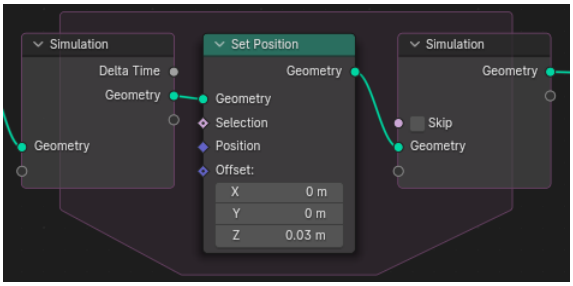


メッシュ編集時のメニューから、Mesh - Set Attribute(属性を設定)にあります。
現在アクティブなアトリビュートに対して、数値やそのほか色などの設定をすることができます。

これによって、メッシュの頂点や面にあらかじめ手でパラメーターを与えて、ジオメトリノードでそれを利用することが容易になっています。
これ以上のGUIを用いた編集手段の充実は、将来の目標になっているようです。

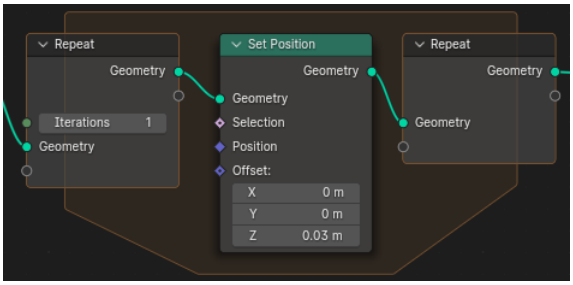
シミュレーションノード と Repeat Zone(リピートゾーン)

Blender 3.6 と 4.0 ではそれぞれ、Simulation nodes(シミュレーションノード) と Repeat Zone(リピートゾーン)機能が追加されました。
これにより、時間的に進化していくような効果や、繰り返しをともなう処理の作成機能が大幅に強化されています。
Simulation Input と Simulation Output に挟まれた部分の処理が、シミュレーションで実行される処理です。



ジオメトリに何らかの処理をして Simulation Ouput まで行くと、
その結果を次の時間フレームの Simulation Input に使う、という理屈で時間と共に進化していくシミュレーションが実現できます。
このシンプルなノード組みは、毎フレームごとに0.03だけZ軸方向に移動する、非常に単純なシミュレーションの例です。

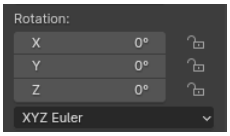
シミュレーション機能にはこのシリーズでは詳しくは踏み込みませんが、非常に強力な機能なので、ジオメトリノードに慣れたら是非シミュレーションにも手を出してみて欲しいと思います。



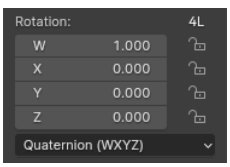
Repeat Zone も似たような仕組みで、Repeat Input と Repeat Output に挟まれた処理を繰り返します。
時間が進んだら処理をする代わりに、Iteration で設定された回数だけ処理を繰り返します。
左の図は　0.03 だけ移動するという手順を10回くりかえすノード組みです。
見た目はシミュレーションノードと似ていますが、色が微妙に異なっているのと繰り返しの回数(Iterations)の指定がついていることが分かります。

「0.03ずつ10回移動するなら0.3移動すれば良いではないか」と思いますが、
もちろんもっと複雑なことをするのに役に立ちます。

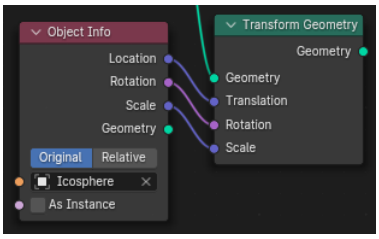
回転の表現



Blender 3.6 までは、ジオメトリノードの回転の表現にはオイラー回転の３成分のベクトル表記のみが使われていました。
XYZ軸回りの回転の程度を表す、デフォルトでオブジェクトの回転状態を表しているこの３成分のことです。



Blender 4.0 以降、Quaternion 表記に相当する回転の表現が追加で導入されています。
データのタイプとして Rotation (回転) となります。
Quaternion 表記では成分は４成分になり（人間目線だと）やや理解が難しいのですが、数学的には式がシンプルになり計算がしやすくなります。

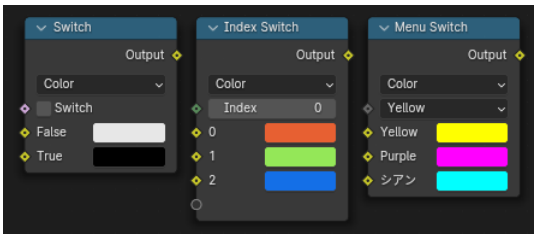


新たに Rotation 用に紫色のソケットが導入されて、Quaternion による回転状態をやり取りできます。
これに伴い、回転に関係するソケットが、ベクトル（青）から回転（紫）に変更されたノードが複数あります。

その分、Blender 4.0 ではどちらの表現での回転なのかの区別を厳密に使い分ける必要が生じたのですが、
Blender 4.1 以降では、ベクトル（オイラー回転）と回転のソケットをつなぐと自動で変換が行われるようになり、区別が曖昧でも使いやすくはなりました。
※ ただ、混乱のもとなので、使い分けの意識は持っていた方が良いと思います。

スィッチの強化

Blender 4.1 ではスイッチ機能が強化されて、Index Switch(インデックススイッチ) と Menu Switch(メニュースィッチ)が追加されました。



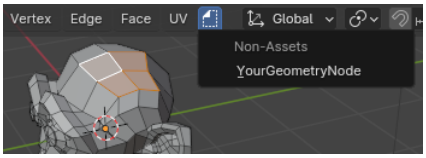
今まで、Switch ノードによって、True か False かの 2 択で値を使い分けるということが出来ました。
False なら黒、True なら白にする、というような使い方です。
Index Switch は従来の Swithc を機能強化したもので、任意の数の選択肢から選択するような使い方が可能になっています。

複数の選択をするような場面では今まで複数の Switch を組み合わせる必要があったのですが、
Index Switch 1 つで済むようなっています。

また、Menu Swith(メニュースイッチ) によって、番号ではなく名前で選択することが可能になり、ユーザーの使いやすさが向上しています。

ツールとしての利用

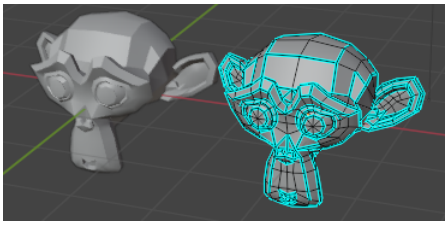
Blender 3.6 までは、ジオメトリノードはモディファイアとして使う以外の利用法はありませんでした。
Blender 4.0 で、オブジェクトやメッシュに即影響に使える機能としてジオメトリノードを使うことができます。
（モディファイアを追加した後適用して、すぐにメッシュの形状を変化させるようなイメージです）



適切に設定をすると、メニューの中から自分の作成したジオメトリノードを直接呼び出して編集作業に使うことが出来ます。
この Tool タイプのジオメトリノードに関しては、Vol.4 の本で詳しく解説をします。

Auto Smooth(自動スムーズ)廃止

Blender 4.1 での大きな変更として、旧来の Auto Smooth(自動スムーズ)機能が取り除かれて、ジオメトリノードによる管理に移行しています。
従来の Auto Smoothのような自動処理をしたい場合には面と辺に対しての Smooth/Sharp 設定で管理を行います。



例えばスザンヌに Auto Smooth をすると、今までは角度の浅い角が自動で滑らかに（急な角はシャープに）なったのですが、
Blender 4.1では、Smooth/Sharp の設定を角度に応じて設定するようになります。
今までの Auto Smooth のような自動処理をしたい場合には、ジオメトリノードで管理することも可能になっています。

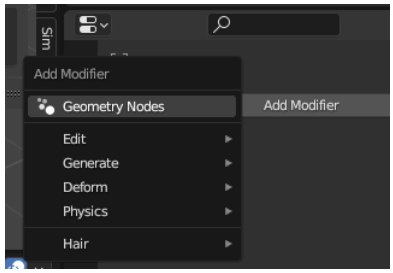
若干管理が煩雑になったとも言えますが、その分きめ細かく制御をすることも可能です。

ジオメトリノードの基本

まずは実際にジオメトリノードの使用例を見てみましょう。

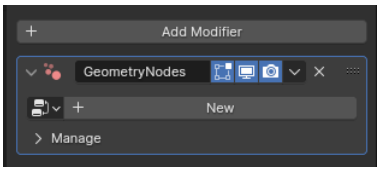
モディファイアの作成

デフォルトキューブをジオメトリノードで変形させてみます。
モディファイアの追加メニューにGeometry Nodes(ジオメトリノード)の項目があるので選択します。



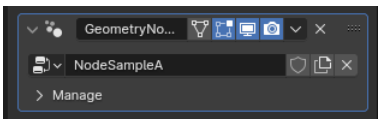
モディファイアのメニュー表示は Blender 3.6 から 4.0 で更新されています。
ジオメトリノードは選択しやすいメニュートップに移動しました。

モディファイアとして追加されました。



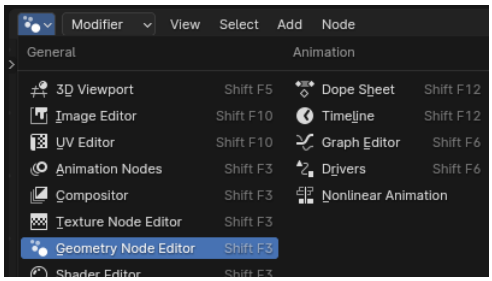
赤く警告が出ているのは、まだノードそのものは作成されておらず、空になっているからです。
モディファイア追加時には自動でノードは作成されず、New(新規)ボタンを押す必要があります。

これは「既にあるノードを使いまわしたい場合」に、「自動でノードが作成されると無駄なノードが増えてしまう」問題を回避するための仕様です。
「New(新規)ボタン」を押して新規に「Geometry Nodes という名前のノード」を作成します。

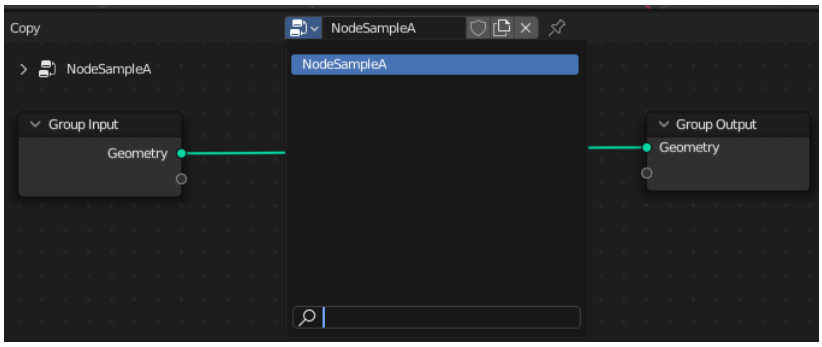


デフォルトのモディファイア名とノード名が似ているので、混乱しやすい点に注意です。
ノードの名前は好きに変更できるので、今回はNodeSampleA、とでもしておきます。

編集は、Geometry Node Editor で行えます。

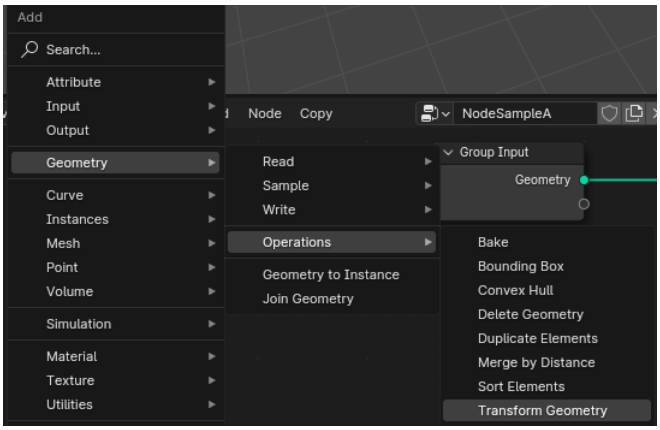


先ほど名前を変えた NodeSampleA が普通に選べるので、そのまま選択して編集に入ります。



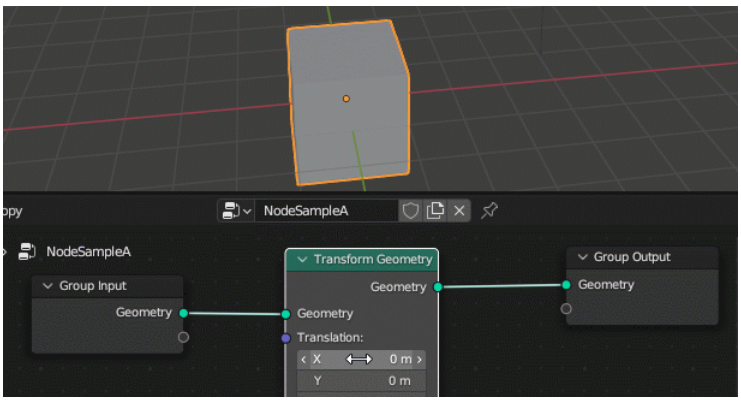
最初のノード編集

デフォルト状態では、「Group Input（グループ入力）」「Group Output（グループ出力）」という2つのノードのソケット同士がつながっています。
ここから、まずは非常に単純なノード構成を作ってみましょう。
Add (追加、Shift+A) - Geometry(ジオメトリ) - Operations(処理リスト) - Transform Geometry(ジオメトリをトランスフォーム)ノードを作成します。



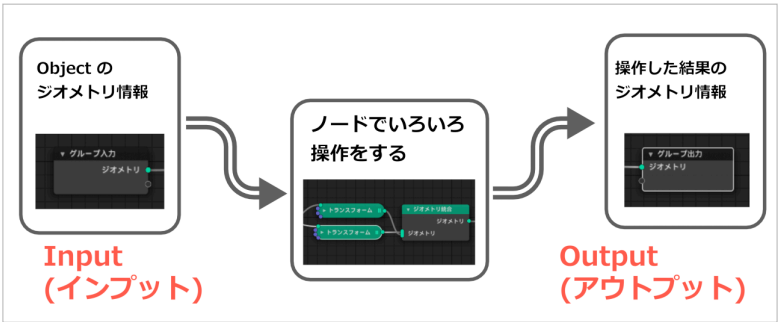
*)追加メニューの中身は、Blender 3.5 で整理整頓がなされて大幅に変更になりました。
3.4 以前の Blender はメニューの並びが違うので注意です

Transform Geometry ノードを間に挟み、パラメータを変化させると、それに応じてデフォルトキューブが移動することが分かります。

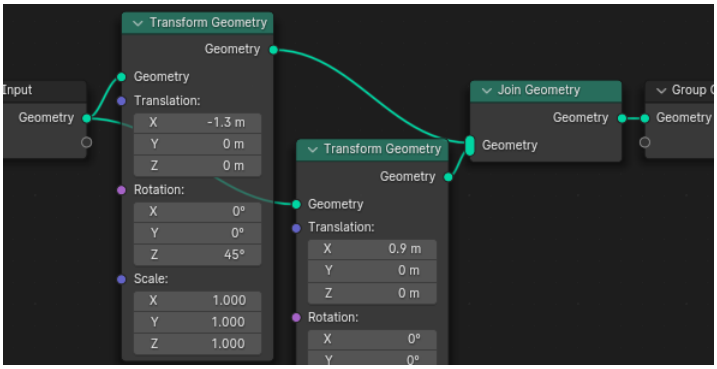


オブジェクトが移動しているように見えますが、よく見るとオブジェクトの原点位置は動いていないので、
実際にはオブジェクトは動いておらず、メッシュの形が変形していることが分かります。
動画)Array01.gif(pdfでは先頭のコマのみ表示されています)

ジオメトリノードの基本形は、Group Input から Group Output の間に、様々な操作を挟み込む形になります。
Group Input (グループ入力) の Geometry (ジオメトリ) ソケットは、元々のメッシュ形状の情報になっています。
それをそのまま Group Output (グループ出力)につなぐと、何もせずに形状は元のままです。
その間に、いろいろな操作を挟むことで、様々な編集が行えます。

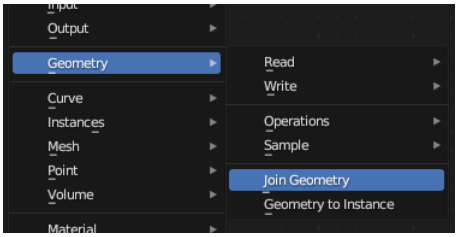


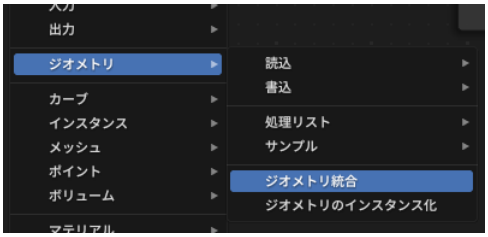
もう少し複雑なノード編集

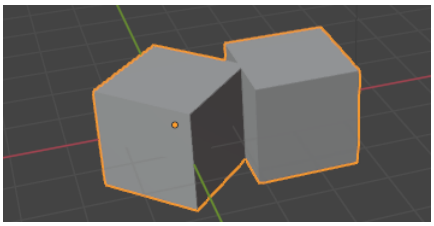


もう少し複雑な編集にしてみます。
Group Input からのつながりを分岐させて、2つの Transform Geometry (ジオメトリのトランスフォーム)につなぎ、
それぞれで別の位置や回転の変形させてみます。

この２つの流れを Add - Geometry(ジオメトリ) - Join Geometry (ジオメトリ結合)で１つにつないで、Group Output(グループ出力)につなぎます。



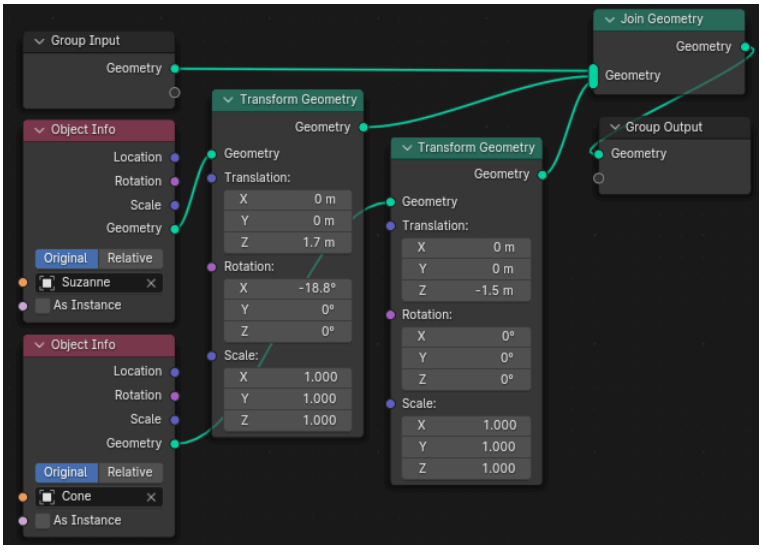




変形したキューブが重なった形状になりました。
Join Geometry(ジオメトリ統合) は その名の通り 複数のメッシュ形状などを統合して１つにします。

別オブジェクトの形状を利用する

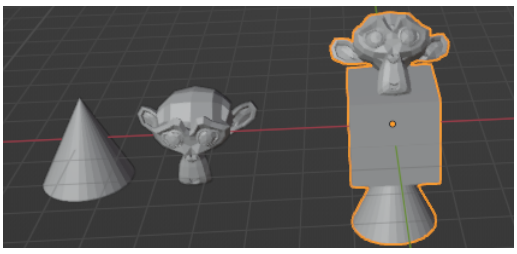
いままでの例では、デフォルトのキューブを作ってそのキューブの形状を動かしたり複製したりしました。
「自分自身の元の形状」だけではなくて、他のオブジェクトの形状を利用することができます。



キューブの他に、円錐やスザンヌなどのオブジェクトを作成し、邪魔にならない場所に置いておきます。

Add - Input(入力) - Scene(シーン) - Object Info (オブジェクト情報)を使って、
そうした他のオブジェクトのメッシュ情報を使ってノードを構築します。

Original(オリジナル) と Relative(相対)の２つのモードがあります。
オリジナルの場合は、元データの形状をそのまま使い、
相対を選ぶと、キューブから見た相対的な位置のずれを踏まえた形状になります。



キューブの他に、スザンヌの頭や円錐を重ねたような形状ができました。

Original モードであれば、まずスザンヌや円筒は Cube と重なるように Cube の原点に配置され、
Transform Geometry で指定した位置や回転状態にずらすことで、このような配置になります。

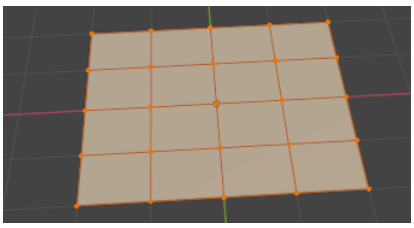
インスタンス

インスタンスは沢山のオブジェクトの複製を表示するような際に使われる技術上のテクニックです。
例えば100個複製を表示したいときに、メッシュの情報を丸ごと100個分複製するのでは無駄が多いので、位置や回転などの必要最低限の情報だけをコピーする方法です。
（正確にはそうしてコピーされた物のことを意味する言葉です）

そのため大量の複製を配置する処理（地面に小石をまき散らすなど）が、かなりの高速で行えるという仕組みです。

よく見ると、先ほど使った Object Inof (オブジェクト情報)のノードにも、As Instance というチェックボックスがあります。
１つ２つコピーをする、のであれば特に必要はないのですが、もし数百ものオブジェクトをコピーしたいのであれば、これをチェックしておけば高速で処理ができるというわけです。
（その代わり、位置情報だけを使うわけですから、後から変形などはできないということですね）

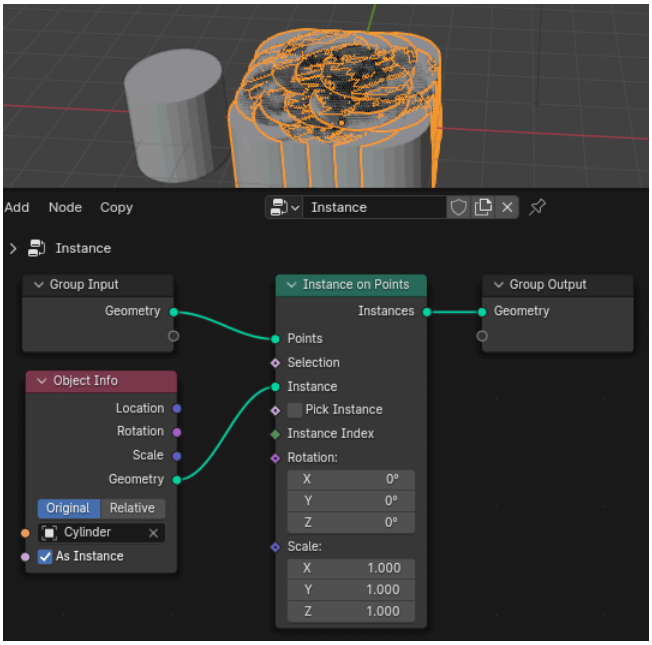
ジオメトリノードには、インスタンスとして物を大量に配置する仕組みが備わっているので、さっそくそれを駆使してみましょう。



平面を細分化して、4x4ぐらいの頂点を持つオブジェクトを作成しました。

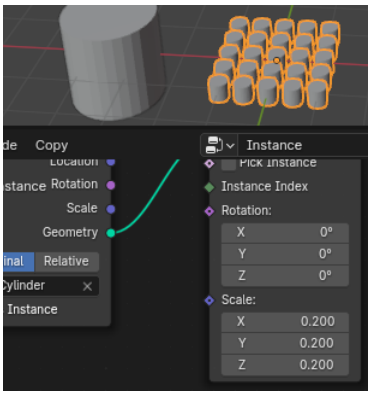
そして、コピーされる側として別のオブジェクトを置きます。ここでは円柱形にしました。

Add - Instance(インスタンス) - Instance on Points(ポイントにインスタンス作成)　を配置して、インスタンスされる側（円柱）ジオメトリ情報もつなぎます。
頂点の位置に Cylinder のインスタンスが配置されます。

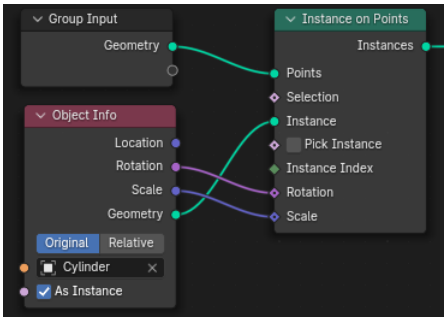


円柱が重なりまくって表示がちょっと乱れていますね。
円柱のサイズを小さくしましょう。
ただこの場合は円柱のメッシュの情報だけを利用しているので、元の円柱をオブジェクトとして縮小してもインスタンスの表示は変わりません。
インスタンスのサイズは Scale のソケットで拡大縮小をします。

とはいえ、エディットモードで元の円柱のメッシュの形状自体を縮小すれば、もちろんインスタンスも小さくなります。
しかし、元の形状をエディットモードで弄るよりも、インスタンスで複製する時にパラメーターとして向きや大きさを変えたほうがより汎用的です。

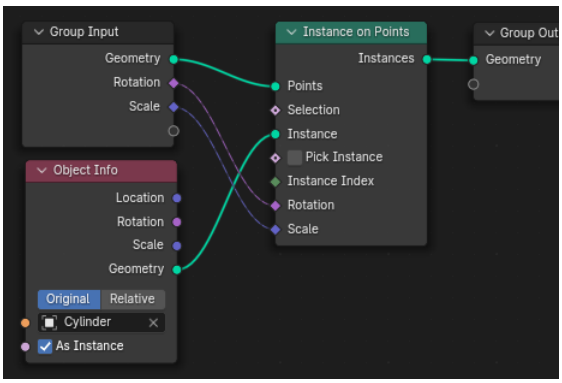


スケールを数値で指定する他に、Object Info から回転やスケールの情報をつなぐことで、
オブジェクトの操作でインスタンスの操作をすることもできます。

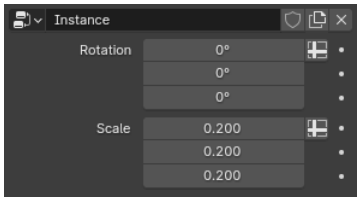


パラメータの入力

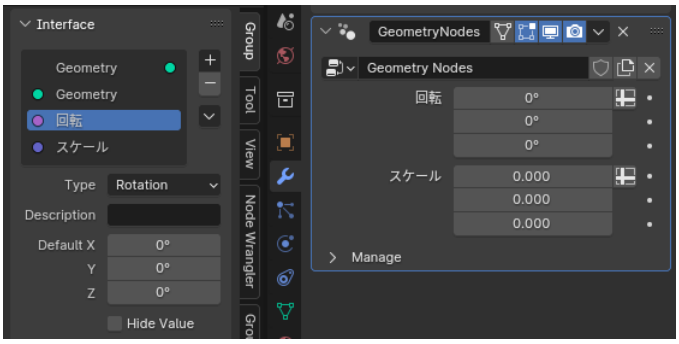
こうしたパラメータは、直接ノードで数値を入力するだけではなく、Group Input(グループ入力)のソケットに接続することができます。



接続したパラメータは、モディファイアのプロパティに表示され、
ノード内で入力するのではなくモディファイアのパラメータとして扱うことができます。



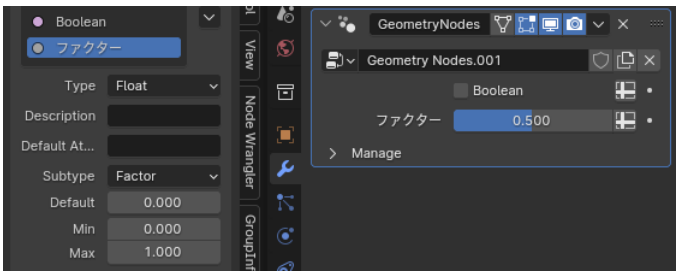
パラメーターの入力や出力は、ノードエディタの右側のパネルで設定ができます。
Blender 3.6 までは Group タブ内の Inputs/Outputs の2つのタブで設定をしたのですが、Blender 4.0 で Interface パネル１つに統合されています。



Interface タブの「+」「-」ボタンでソケットの追加や削除ができます。
左図は回転やスケールのパラメータがモディファイア側から入力できるようになっています。
こうすれば、パラメータ違いのバリエーションも含め何度も別オブジェクトで再利用が可能になりますから、非常に便利です。

図のようにソケットの名前を自由に（日本語も含めて）変えることもできます。
（選択部分をダブルクリックで名前変更ができます）

一部のデータタイプは、Subtype(サブタイプ)の設定を行えます。
例えば回転情報は、Rotation タイプの他にも、3成分のベクトルとしても表現ができます。(オイラー角表現)
ベクトルを位置や速度ではなく角度用の情報として使うのであれば、オイラー角を設定すれば(ラジアンではなく)度を単位として入力できるなど、操作性が上がっています。

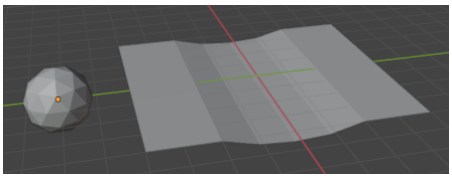


Float タイプの入力を作成し Subtype を Factor(係数)にすれば、スライダーとして数値の変更が行えますし、
Type が Boolean(ブーリアン)の場合には True か False かの選択をチェックボックスで行えます。

インスタンスを利用した配置

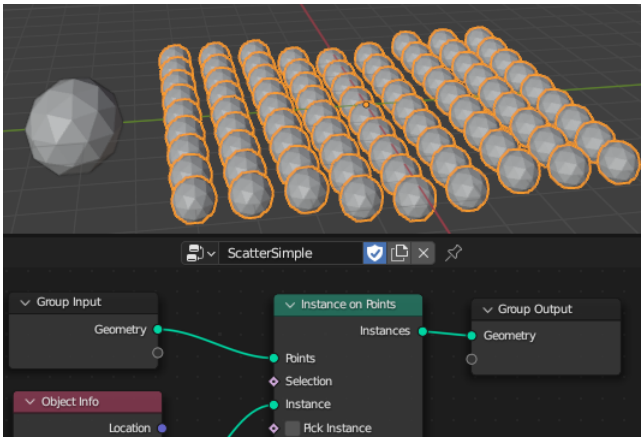
ポイントにインスタンス作成

Instance on Points (ポイントにインスタンス作成)の最初の例では、メッシュの頂点にオブジェクトを配置しました。
それはそれで、オブジェクトの配置を細かく制御するには便利ですが、もっと適当に大量の物をバーッと配置する、というような時には少し手間がかかりすぎます。
頂点の位置ではなく、メッシュの表面にランダムに物を配置することができます。



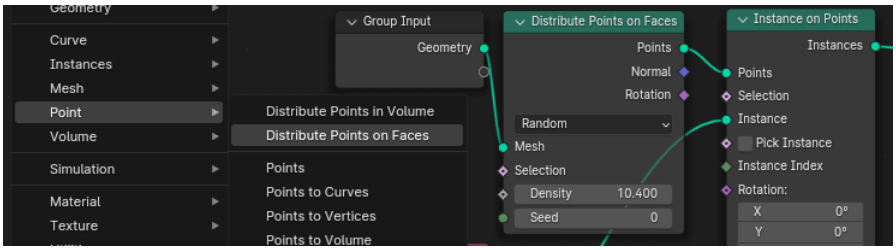
配置の元になるような平面と、配置するための球を作成しておきます。
地形に小石を置く、というような利用を考えて少し凹凸をつけておきます。

先ほどと同様に、Instance on Points を設定すれば、頂点の位置に球が配置されます。

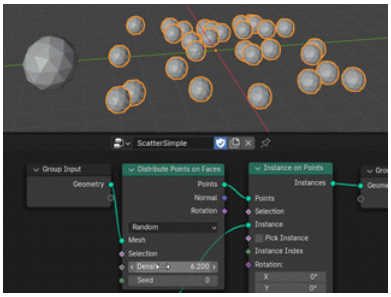


左に配置した Icosphere の複製を配置しました。
ただし、このサイズのままだと重なりまくってしまうはずです。
インスタンスのスケールなどを適切に設定しておきます。

平面の頂点以外の場所にインスタンスを置いてみましょう。
Add - Point(ポイント) - Distribute Points on Faces(面にポイント配置)を挟み込みます。



これにより Points の情報は頂点の配置ではなく、
メッシュの面上にランダムに分布されたポイントクラウド（点群）に変換されます。



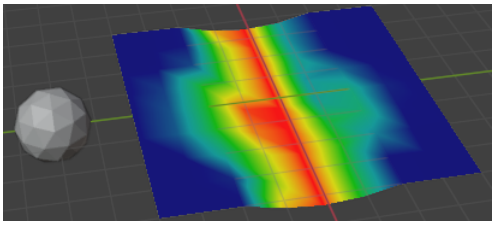
Density(密度) パラメータを変更することで、密度が変化していることが分かります。
動画)Scatter01.gif(pdfでは先頭のコマのみ表示されています)

さて、小石をちりばめるような場合には、完全にランダムな配置ではなくて、濃い薄いの指定がしたいところです。
密度を指定するために、元のメッシュにパラメーターを設定します。
そうした用途に使えるパラメーターには、Vertex Weight(頂点ウェイト)と Attribute (属性) の２つが存在しています。
頂点ウェイトは、以前よりずっと存在する機能で、メッシュの各頂点に 0-1 の値を持たせることができます。
モディファイアの効き目の調整や、ボーンを使ったアニメーションを設定するときにおなじみの機能ですね。

Attribute は、頂点ウェイトに似ていますがジオメトリノードのために追加された、より新しく多機能なパラメーターの収納法です。
頂点ウェイトは Attribute の一種で（昔から存在するために専用の編集モードなどを持った）特別なものと考えることもできます。
どちらも似た感覚で使えるのですが、まず頂点ウェイトを使って密度の設定をしてみましょう。

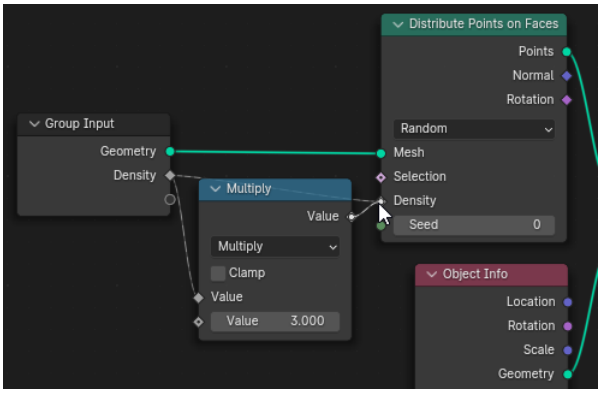
Group Input(グループ入力)によるアトリビュートの利用

頂点ウェイト編集で、元になるメッシュの凹んだ部分に頂点ウェイトを塗ってみます。



Vertex Group (頂点グループ)の名前は、デフォルトの Group のままにしました。
（実際は分かりやすい名前に変更するのが良いです）

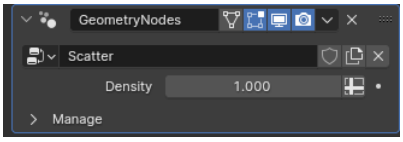
こうした頂点ウェイト（やAttribute）を利用する方法は２つあります。
一つは、Group Input(グループ入力) ノードを設定する方法です。



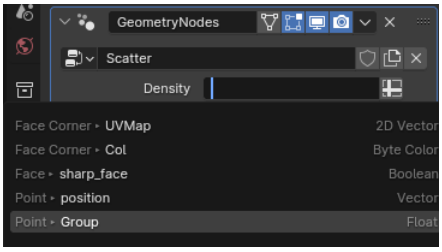
タブからの操作ではなく、入力用のソケット(例えば Density)のソケットを
Group Input (グループ入力)の空いているソケットにつなぐ操作でも、ソケットの追加が行えます。
動画)Scatter02.gif(pdfでは先頭のコマのみ表示されています)

ただし、実際には頂点ウエィトの範囲[0-1]では大きさが足りない場合も多いので、
調整ができるように間に Multiply(乗算)ノードなどを挟んで定数倍をかけた方が良いでしょう。

四則演算などの基本的な数式ノードは、
Utilities(ユーティリティ) - Math(数式) - Math(数式) から追加します。



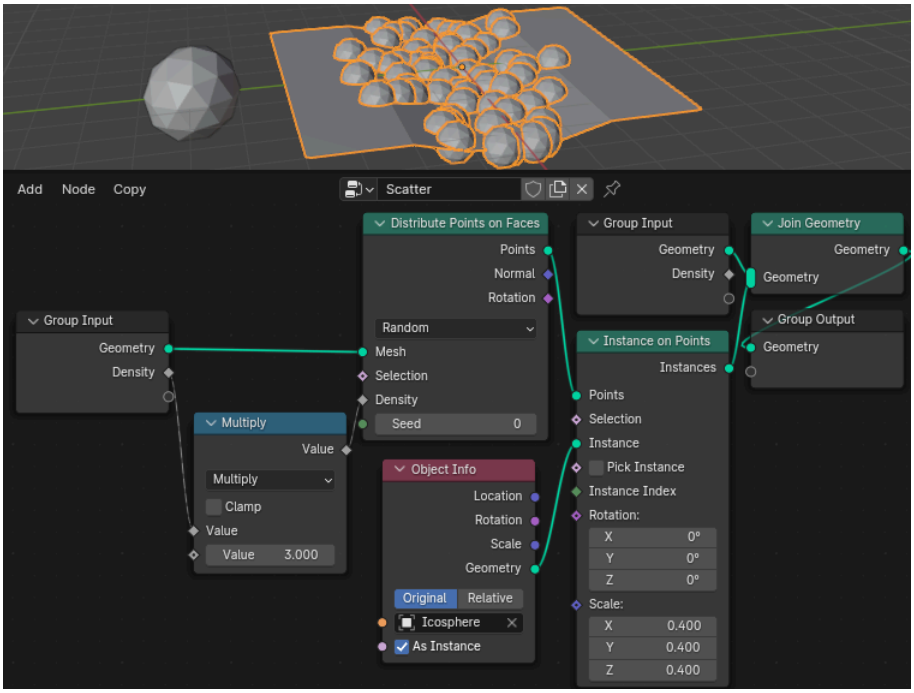
モディファイアの設定から Density のパラメーターを入力できます。
ここまでは最初の作例と一緒です。
ここで、数値の脇にある十字のマーク（スプレッドシートのマーク）をクリックします。



すると「一定の数値」ではなく頂点ウェイトを含むアトリビュートを選択して使えるようになります。
今回、Group という名前の頂点ウェイト情報があるので選択肢から選べます。

文字で直接指定することもできます。

密度を Group に設定することで、頂点グループの値に従って分布が変化しました。
最後に、Join Geometry を使って、元の形状とインスタンスを両方表示するようにします。
サンプルはあまりに単純な形状ですが、形状をきちんと作れば窪地に小石がたまったような表現ができます。



ところで Density を繋いでいるソケットの形が、丸ではなくて菱形なのに気が付いたでしょうか？
この菱形のソケットは、各頂点ごとに値を持つ情報であることを意味しています。

例えば、ランダムに配置するための シード値(Seed)は、
乱数を設定するための数値一つのパラメーターなので、丸いソケットがつながっています。
Density は各頂点ごとに濃淡を設定できるという事で、菱形のソケットになっているわけです。

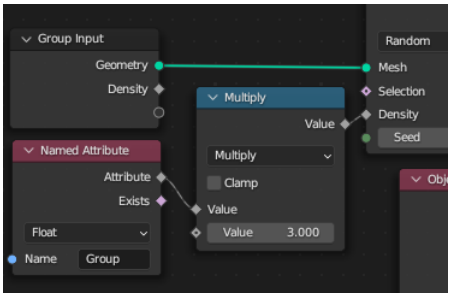
菱形に小さな点が打ってあるソケットは、菱形と丸の間のような状態のソケットです。
菱形のソケットなので各頂点ごとにバラバラの値を持つことができるのですが、１つ値をパラメーターとして使っていることを表します。
密度一様な分布などは、数値一つで指定しても良いわけですね。

Instance on Points の Rotation や Scale のソケットには、各頂点でバラバラの値を指定ができるのですが、
今は回転が 0 スケールが 0.3 の定数が指定されているので、黒ポッチの付いた菱形になっているというわけです。
もう少し詳しい説明は後で見てみましょう。

Named Attribute(名前付き属性)

先ほど説明したのは、Group Input を使って アトリビュートの名前をモディファイアのパネルで指定して利用するという方法です。
これは、利用者がノードの中身を知らなくても「パネルで名前を指定すれば使える」というメリットはあるのですが、手数が多くて面倒です。

パネルを使わずに、ジオメトリノード内で（名前を決め打ちした）アトリビュートを利用をすることができるノードが、Blender 3.2 で追加されています。
Geometry - Read - Named Attribute(名前付き属性) です。
これによって、パネルと Group Input のノードを介さなくても、アトリビュート情報を利用することが出来ます。



Group Input を使う代わりに、Named Attribute(名前付き属性) を使ってみます。

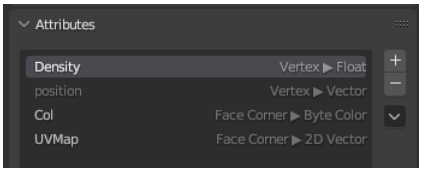
Group Input を経由せず、ノードの構成の中で（ユーザーがパラメーターを指定せずに）
Group という名前を決め打ちして頂点ウェイト「Group」の情報を使うことができます。

※Name(名前)のソケットを使い、ユーザーに名前を決めさせることも可能です

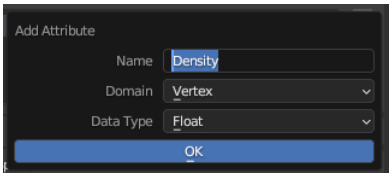
アトリビュートの名前を「ユーザーが変更する必要が無い」「決め打ちの名前で充分」という時は、パネルを介したやり方よりも手数が減って楽になります。
また、アトリビュート情報を利用するだけではなく、アトリビュートへの書き込みを行うStore Named Attribute(名前付き属性収納)というノードも用意されています。
もっと詳しい内容については、Vol.2 でより詳細に説明しているので、そちらを確認してください。

Attribute(アトリビュート)の編集

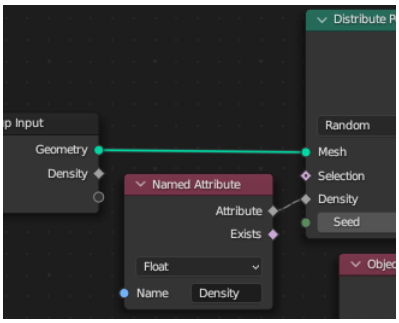
先ほどは 頂点ウェイトを利用してメッシュに密度のための情報を持たせました。
この他に、汎用的な Attribute を利用する方法があります。



メッシュのプロパティの中にある、Attribute(アトリビュート)のタブで確認ができます。
既に頂点の位置情報(編集不可)や UVMap の情報が Attribute として登録されていることが分かります。
「+」ボタンから密度用にアトリビュート(Density)を追加します。



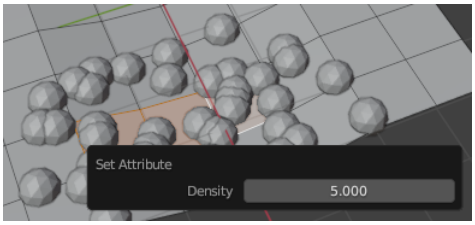
名前や、データの種類、そして Domain を指定します。
頂点ウェイトは「頂点ごとに」「値(Float)」を持つようなデータでした。
Attribute では同様の設定もできますし、頂点ではなく「辺ごと」や「面ごと」に「値」や「色」を持つデータなど、多彩な設定が可能です。
そうした「何がデータを持っているのか」という設定には Domain(ドメイン)という用語が使われます。



頂点ウェイトの時と同様に Named Attribute ノードで名前を指定すれば、その名前のアトリビュート情報を利用できます。

頂点ウェイトは[0-1]と値の範囲が決まっていましたが、アトリビュートの範囲は決まっていません。
大きい値を利用するために Multiply を挟まなくても良いことになります。
（挟んだ方が、全体的に密度をx倍する…というような操作が簡単になるので便利ですが）

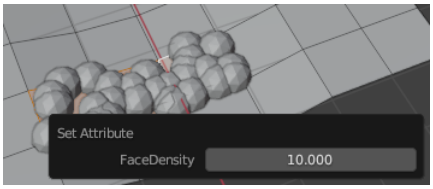
アトリビュートの編集機能は、blender 3.5 で簡易的なものが実装されています。
メッシュ編集時に、設定したい頂点(/辺/面)を選択して、Mesh メニューの Set Attribute (属性を設定)を実行します。



この時、編集したいアトリビュートをパネルから選択をしてアクティブにしておく必要があります。
（そうしないと、どのアトリビュートを編集するのか分からないですからね…）

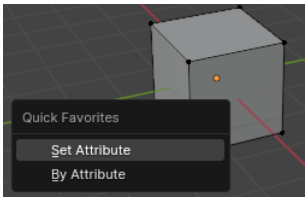
数値を入力すれば、それが Attribute に反映されます。
今回は頂点ウェイトと似た設定として「頂点ごと」に値を設定しました。
選択している面の外側もにも球の配置がはみ出しています。

というのも、密度5を設定した頂点とその他の密度0の頂点の間にある面では、補間がされて密度が0の扱いにはならないからです。
隣接する面の範囲まで分布は広がります。



ドメインとして「Face(面)」を指定しておけば、面ごとにパラメーターを持つので、
このように、くっきりとした境界を持つ密度変化を表現することもできます。
頂点ウェイトよりも、アトリビュートの方が柔軟な操作ができて高機能ですね。

この他、ブーリアンのアトリビュートに対しては、メニューの Select - By Attribute によって、True である頂点/辺/面 の選択をする機能があります。
※ブーリアンに対してのみ使えるので、少し使い勝手は悪いものです。

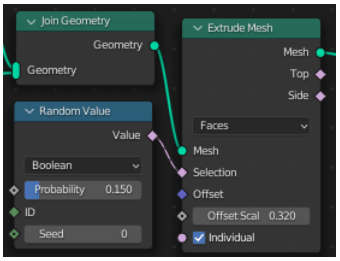


ジオメトリノードを多用する場合には、
これらのメニュー項目は、Quick Favorites(お気に入りツール)に登録しておくと便利ではないかと思います。

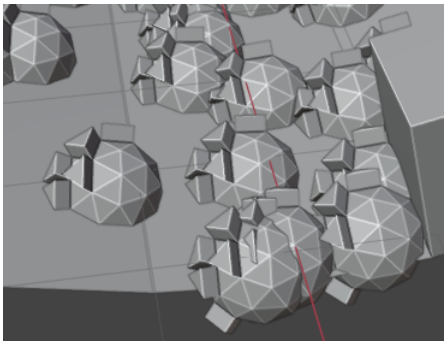
しかし、アトリビュートの編集機能は Blender 4.1 の時点では簡易的なものです。
メニューから選んで値（タイプ次第でベクトルや色など種類に応じたデータ）を指定するしかできません。
頂点ウェイトのようにGUIを使ってブラシで塗るような編集操作などは未実装です。
（比較的近い未来にそうしたGUIを使った編集を実装するのが目標のようです）

インスタンスとメッシュの変形、実体化

インスタンスは、既に存在しているメッシュの複製です。
そのため、変形を個別のインスタンスにかけることはできません。
変形に関しては次の章で説明するので、少し先走った内容になりますが複製した球に変形をかけてみます。

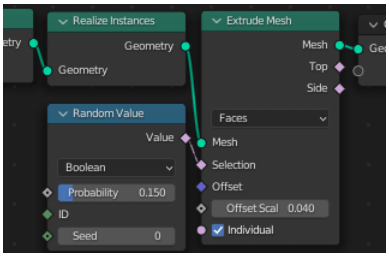


作成されたインスタンスを含む形状全体に、Extrude Mesh(メッシュ押し出し)をランダムにかける効果を加えます。
メッシュの変形に関しての詳細は、次の章で説明します。

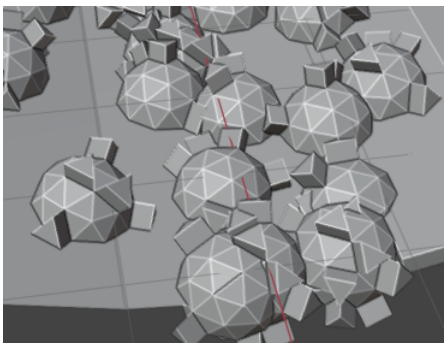


面の一部が押し出されます。
この時、インスタンスである球に対しても変形が適用されていますが…
よく見るとすべて同じように変形をしていて、（全体としては）ランダムな変形ではないことが分かります。

これは、「インスタンスを複製する元になるメッシュ情報」に変形が適用されていているからです。
あくまで複製を配置する機能なので、配置後の形状に対しての変形にはならないのですね。
そのため、全部が同じ変形になるわけです。



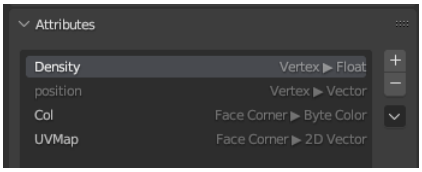
Instance - Realize Instance(インスタンス実体化)のノードを使うと、
インスタンスだった形状がメッシュに変換されます。
（複製だった球が、全て頂点データを持った普通のメッシュになるのですから、その分データとしては重くなるわけです）



この状態であれば、全ての球が別々の形状が取れます。
ランダムを使った変形なので、すべてが違う形状になりました。

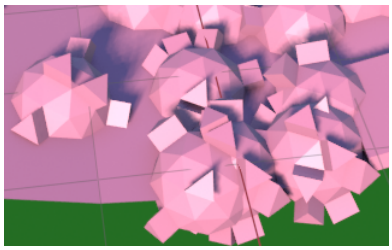
（逆に、もし、全部同じ形状になるような変形で良いのであれば、
速度やデータの重さという点ではインスタンス状態のままの方が有利なわけですね）

アトリビュートとUV



アトリビュートのパネル内に UVMap の情報が見えることから分かるように、
Blender 3.5 以降では UVMap 情報はアトリビュートの一種とみなすことができます。
Face Corne(面コーナー)が持つ 2D Vector という、特別な場合のアトリビュートというわけです。

UVマップの情報は昔から実装されてきた古い機能で、アトリビュートはジオメトリノードの開発にともなって開発された新しい機能です。
Blender 3.4 まではこの2つは「似ているけれども違うデータ」として扱われていました。
そのため、Blender 3.4 までの時点で Realize Instance など一部機能を使う時に、UV情報が取れてしまうという問題がありました。
例えば Eevee 使用時に UV を使ったテクスチャを貼っていると、特定の操作やノードの利用で下の左図のようになってしまったのです。

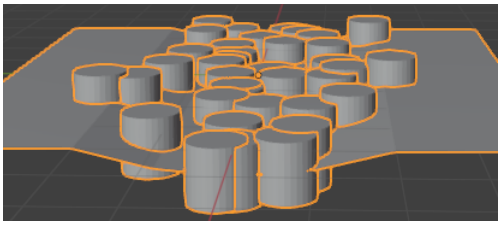


これらの現象は UV 情報が Attribute に変換されて保存されることが原因です。
Blender 3.5 以降では、UV もアトリビュートの一種としての実装になったため、
オブジェクトの変換などの処理をしても、UVの情報が失われるという問題は恐らく解消されたようです。

小石をちりばめる

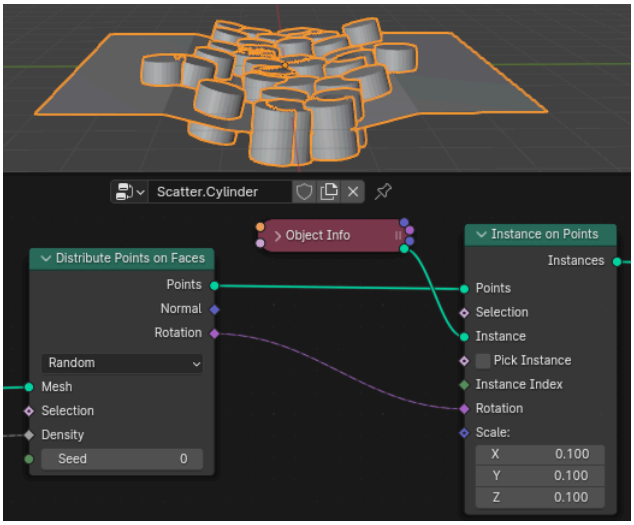
それでは、折角なのでもう少し凝って、地面に小石をちりばめてみましょう。

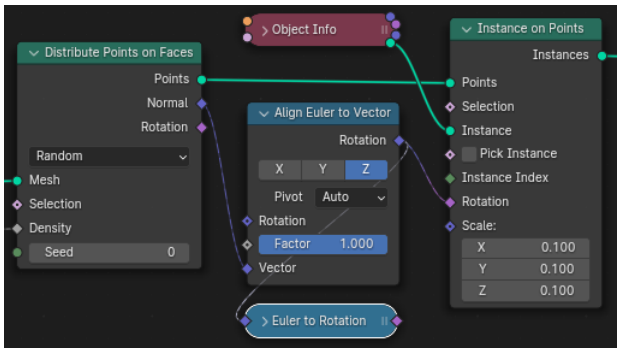
小石が全部同じ向きを向いていると不自然なので、まずは向きをランダムにしてみます。
最初に、そもそものインスタンスの向きを確認します。
しかし、先のサンプルで使った球だと向きが分からないので、球ではなくて Cylinder をインスタンス化することで、向きを確認してみます。



発生元は単純な平面ではなく、
少し傾きなどをいれた平面にしてみましょう。

平面の向きにかかわらず、特に回転方向の変化は無いことが分かります。
（古い2.9x時代の対応するノードは、法線向きにインスタンスを発生させていたので、挙動が逆な点に注意します）

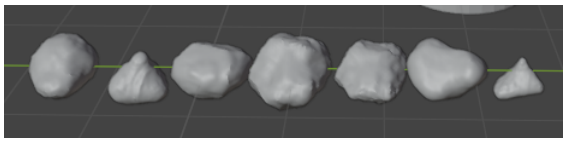




法線方向を向かせるためには、Rotation のソケット同士を接続します。
Distribut Points on Faces の紫の Rotation のソケットからは、法線方向を向ける回転情報が出力されています。

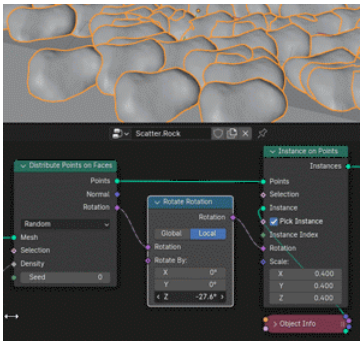
もしくは 法線ベクトルから対応する回転成分を得る、Utilities(ユーティリティ) - Rotation - Align Euler to Vector(オイラーをベクトルに整列) を使う手もあります。
この例だと単にノードが増えただけですが、自力でベクトルを計算して向きを揃えたりする際によく使うノードです。
厳密に考えると、Align Euler to Vector はオイラー回転での回転表現なので Euler to Rotation ノードを使ってデータのタイプをそろえるべきなのですが、
Blender 4.1 以降はタイプの違う回転データも自動で変換されるので、直接紫のRotationソケットにつなぐことが可能です。
※Align Eulter to Vector で行う計算は法線方向に回転を向かせる計算で、ねじり方向には自由度があり Rotation を直接つないだ場合とはねじり方向に異なる結果になる場合もあります

ただの球や円柱だとありがたみが薄いので、
例えば Rock Generator (標準アドオン Add Mesh: Extra Objects) などでそれらしい形を作成して砂利をちりばめてみます。



形状を複数用意しました。しかし、まずはこのうちの1つを使って配置をして向きをランダムにしてみます。

紫ソケットの回転情報を制御するのは Add - Utilitiest(ユーティリティ) - Rotation(回転) - Rotate Rotation(回転を回転)です。

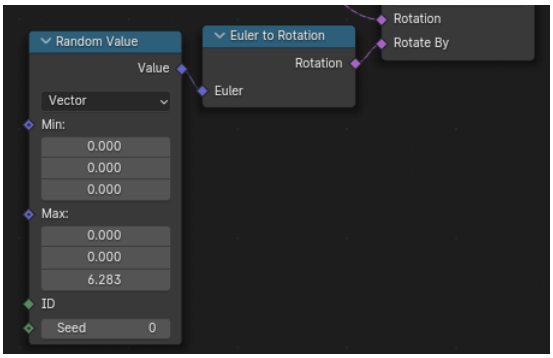


回転のパラメーターを変えると、配置している小石の向きが変わります。
動画)Scatter09.gif(pdfでは先頭のコマのみ表示されています)

Global設定をするとグローバル座標を基準に、Localの設定すると今それぞれの点の向き(つまり平面の法線)を基準にしてオイラー回転を行います。
(x軸回りに何度、y軸回りに何度…と回転を設定します)

そのまま数値入力をするだけでは、全部一斉に同じように回転してしまうので、バラバラにランダムな回転させる方法を探ります。

Add - Utilities(ユーティリティ) - Random Value(ランダム値)を使って、ランダムな値（もしくはベクトル）を作成できます。

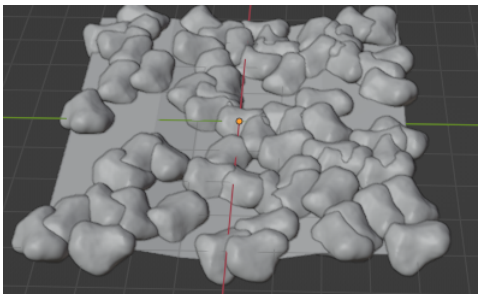


範囲を指定するソケット(Min, Max)が菱形なので、各点毎にバラバラのランダムな値を設定することもできます。
しかし、今回は一定の値で良いので、MinやMaxに定数を直接入力します。

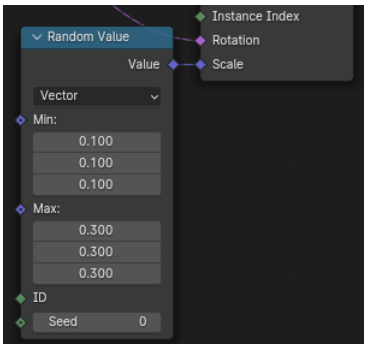
今回は、z軸に沿った回転を与えてみましょう。
ベクトルのz要素だけ値を持てばよいので、
(0,0,0) から (0,0,6.283) の範囲を指定します。

回転の単位はGUIの表示上では「度」の表記になっていますが、内部的には実はラジアンで扱っているので、一回転は360ではなくて2π、約 6.283 になります。
(入力欄に pi*2 と入力しても良いのを知っていると便利です)

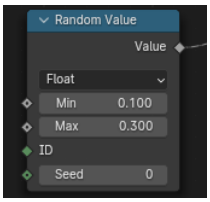
このZ方向を向いたランダムな大きさのベクトル(0, 0, ???)は、Z軸回りの回転をオイラー回転で表現したものです。
紫の回転情報のソケットにつなぐには、本当は Utilities - Rotation - Euler to Rotaion(オイラーの回転化)を使って回転情報に変換をしないといけません。
Blender 4.1 以降は、青ソケットのオイラー回転情報を紫ソケットの回転情報のソケットにつなぐと、自動で変換が行われるので省略可能です。



ランダムに回転した配置が得られました。
サイズのランダム化もほぼ同様に Random Value ノードを利用して行えます。



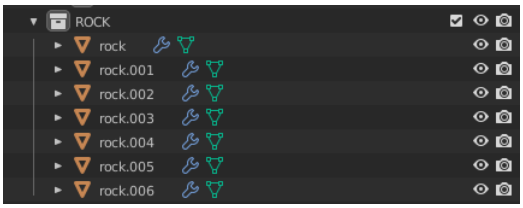
Scale のソケットに Random Value を接続することでサイズをランダム化できます。
使うのはベクトルなので、通常はモードを Vector にしておきます。



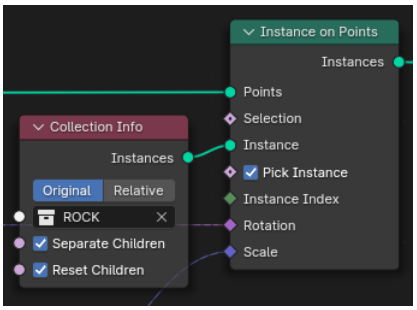
ただし、XYZ各成分ごとにばらばらのランダムなので、細長く引き延ばされたりもします。
比率を変えずに、成分が一緒のランダムの方が良い場合は、値(Float)のランダムをベクトルのソケットにつなぐことができます。

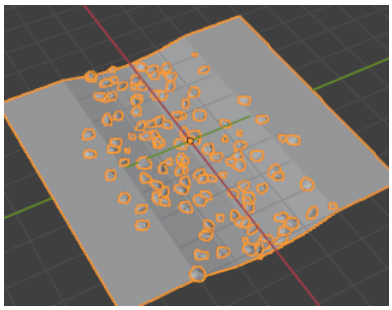
コレクションの複製

さて、1種類の石だけを分布させたのですが、折角何種類かの石を作成したので、さらに複数の種類の石をちりばめてみましょう。
オブジェクトではなく、コレクションをインスタンスとして配置することができます。
その前準備として、作成した石のセットを、一つのコレクションにまとめます。

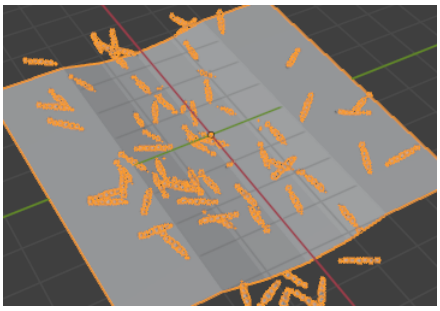


Instance on Points につなぐノードを Object Info から Collection Info にすることで、コレクション内のオブジェクトを配置します。

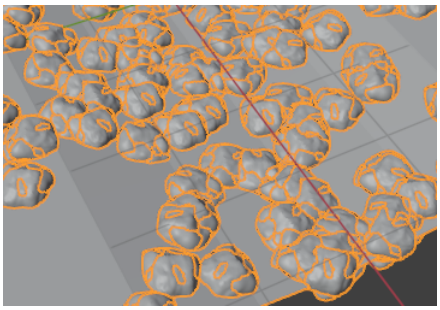




幾つかチェックボックスの設定がありますが、今回のような使い方の場合はすべて有効しておきます。



Separate Children(子を分離) - コレクション内のオブジェクトをひとまとめに扱わずに、バラバラに扱います。
例えば、もともとの岩のセットを一列に並べていた場合に、これが無効のままだと
ひとまとめに扱われるので、このように並んだ状態で複製されてしまいます。



Reset Children(子をリセット) - バラバラにしたコレクションのオブジェクトの位置情報を使わないようにします。
今回のような場合は、石オブジェクトを並べた位置の情報を反映しては困るので、有効にします。

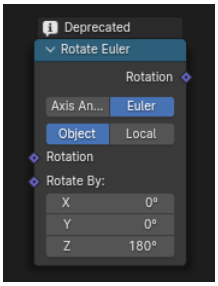
Pick Instance(インスタンスのみ) - コレクション内のオブジェクトを1つだけ使います。
これを有効にしておかないと、用意したオブジェクトを全部同じ位置に置くことになってしまい、図のように重なりまくってしまいます。

必要な設定が少し多いですが、これで様々な小石を地面に沿って配置することができました。
より自然な感じにするには、サイズや密度、頂点グループなどを調整します。

テクスチャ等を加えた .blend ファイルを、01_SCATTER/Scatter.blend として同封してあります。

このテクスチャは、Blender Kit で配布されている石や地面のテクスチャを使用したもので、UVを使用しないプロシージャルなものではなく、UVとテクスチャ画像を利用したものになっています。

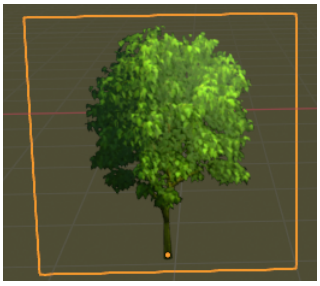
この小石は、「平面上の点」に配置されているので、半分地面の下にめり込んでいます。
平面の「上に乗っている」ようにするならば、例えば小石のメッシュの位置をずらしたり、小石のサイズの分だけインスタンスを上に移動させたりなど、もう少し調整が必要になるでしょう。



Blender 4.1 以降は、回転の表現は紫のソケットを使うことが基本になっています。
従来のオイラー回転の表現（青のベクトルのソケット）は縮小されています。
回転の計算のために使われていた Rotate Euler(オイラー回転)ノードは、Deprecated(廃止予定)に分類されました。
このノードで行っていた回転の操作は、Rotate Rotation を使って行うことが出来ます。

ビルボードの配置(カメラの方向を向ける)

沢山の物を配置する、という場合に使われがちなテクニックとして、ビルボード表示があります。
配置させたいもの自体が複雑で重い形状、例えば木をリアルに枝葉を表現したようなオブジェクト、の場合は、たとえインスタンスとして表示してもかなりの重さになります。
そうした際に、あらかじめ1枚の（もしくは複数枚の）テクスチャを使った板１枚として表示するような方法です。
カメラが近づけば粗が出てしまいますが、そうでない場合には大きな効果を発揮します。



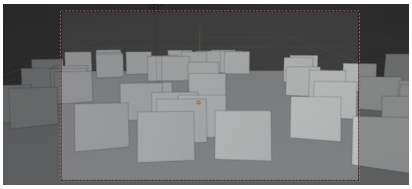
あらかじめ板に張り付ける画像を用意しておきます。
単純な一枚絵でも良いですし、少し凝ったりする場合には光源を反映させるための法線マップなどを利用すればより良い結果が得られます

木の場合は地面から生える根元を原点にしておくと制御がしやすくなります。

透明を有効にするマテリアルの設定などはここでは説明しませんので、
うまく用意してください。

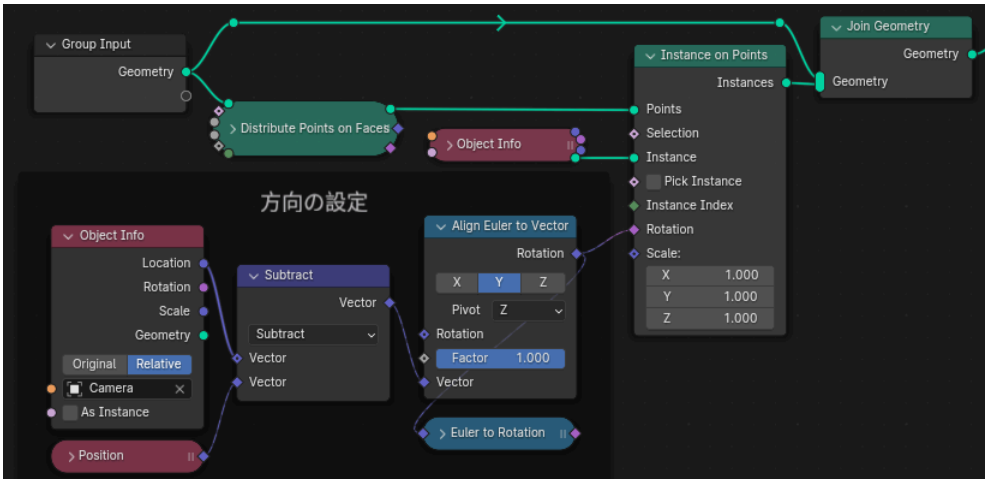
この板をインスタンスとして配置します。
もちろん、横から見た図しかないのですから、カメラが見下ろすような構図には対応しません。
カメラがほぼ横から見ているものとして、カメラの方を向くようにします。

このあたりのノードの設定は上の小石を配置する例とほぼ一緒です。
違いは、板の向きをどのようにカメラの方向を向くように設定するかです。



板がカメラの方向を向くように回転の設定をします。
大量に板を並べて森を表現したいところですが…まずは見やすく少なめの配置です。

必要な情報は、Instance on Points によって作成された、それぞれの点の位置から見て、カメラがどちらの方向にあるかです。
そこで、Object Info(オブジェクト情報) の Location (位置)を用いてカメラの位置情報を取得します。
Object Infoノードでどのオブジェクトの情報を使うかは手動で指定をしますが、Blender 4.1 以降では、Input - Scene - Active Camera(アクティブカメラ) ノードを利用しすることができます。
このとき、Relative (相対)を用いて、処理をしているオブジェクトの座標でのカメラの位置情報を得るようにします。
各点の位置 Position とこのカメラの位置の差が、各点から見た相対的なカメラ位置への差分ベクトルになります。

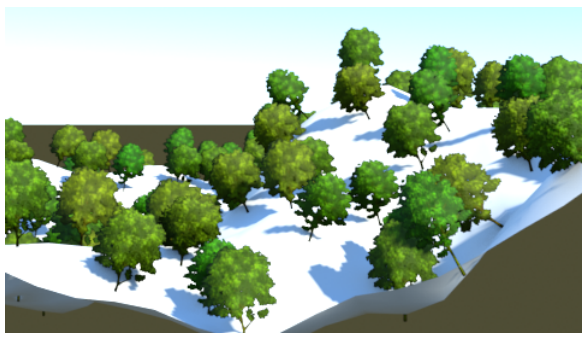


Align Euler to Vector を使って ベクトルから回転成分に変換して、Rotation のソケットにつなぎます。
板のどの方向(XYZ)を向けるか、どの軸を回すことにするのか(Pivot)などの設定はミスることが多いので気を付けます。

本来はデータのタイプを合わせるために Euler to Rotation を間に挟んで
オイラー角表現からRotation 型に変換をするべきなのですが、
Blender 4.1 以降では、自動での変換が行われるのでその手間は省略できます。

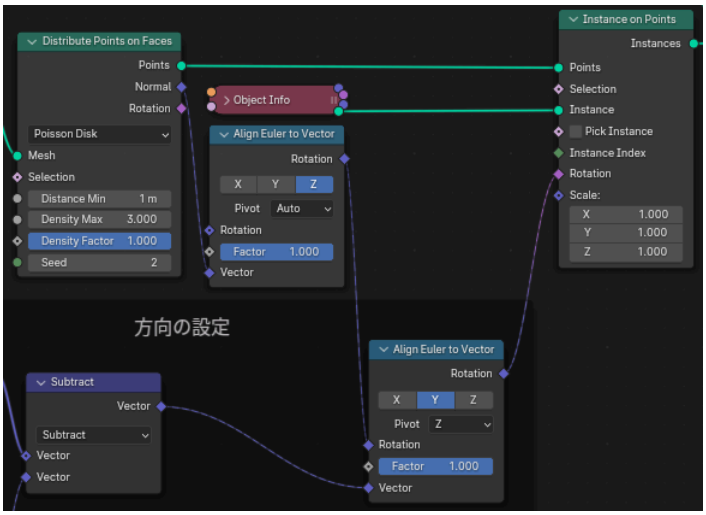
Autoだとカメラとの位置関係によっては上手くいかないことがあります。
木のような場合は、基本的に z 軸方向にだけ回転すればよいので、Pivot を Z軸に合わせるという手もあります。
また、板を向ける方向の指定はXYZ方向だけで 逆向きの設定が無いので、板の裏表は気を付けて…

木をZ軸方向に回してカメラの方向を向けると、斜面の木もやはり上を向きます。



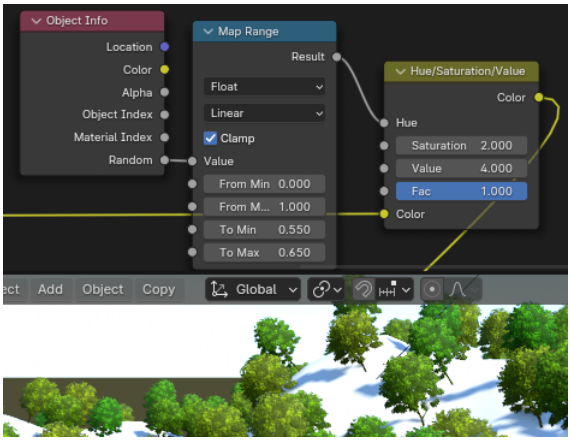
斜面に生えた杉の木(基本的に上に伸びます)のような場合はこれでも良いのですが、木の種類によっては斜面の方向に斜めに生えるような場合もあります。

斜面へ追随する成分も入れてみましょう。



Distribute Points on Faces(面にポイントを配置)のNormal と Align Euler to Vectorを使って、まずは斜面法線方向を向かせます。
(直接Rotationのソケットを使っても良いのですが、ノードを使えば、Factor を使って傾きの調整ができます)

木が斜面の法線方向を向いたので、それの後で、カメラ方向をさらに向くように、２つ目の Align Euler to Vector につなぐ形になります。
木の大きさ(scale)も調整できるので、多少ランダムを使えば同じ木が並んでいる感が薄れて良い感じなるでしょう。



インスタンスは、レンダリング時には別オブジェクト扱いになります。
なので、例えばシェーダー内でオブジェクトごとのランダムなどの情報を使うことができます。
例えば色合いに変化を持たせるというようなことは、シェーダーで処理ができます。

木の形状が完全に違うバリエーションにするならば、バリエーション違いのオブジェクトを収めたコレクションを用意して、
コレクションのインスタンスにすることで対応できます。

もしくはランダムを利用して使うテクスチャを変えるようにマテリアルのシェーダー側に組み込む事も可能です。
サンプルに収めた、02_BILLBOARD/Billboard.blend では少し複雑な例として、
シェーダー側で３種類のテクスチャパターンを切り替えるようにしてみました。
複数のテクスチャは、UDIMの番号を使って管理するようにしました。