Grasshopper Tutorial

Feb 2017

1　練習: Surface の再構築

この章では、基礎的な Component を使い、Surface の再構築の方法を学びます。また、Ghpython を用いたプログラミングで同じ機能を実現するやり方を学びます。
本章で必要なファイル:

1. Rhino モデル: turbine.3dm
2. GH ファイル: remake_surface.gh

Surface 再構築の手順:

1. Surface から Curve を抽出します (Brep Edges)。
2. 分割数を指定し、Curve 上を通る Point データを作成します (Divide Curve)。
3. その Point データを通る NURBS Curve を構築します (Interpolate)。
4. 4 つの NURBS Curve から Surface を構築します (Edge Surface)。

本章ではこの 4 つを順にやっていきます。図 1 が今回のスクリプトの全体図を示しています。このスクリプトを使えば、図 2 のように要素数を自由に変化させて surface の再構築が出来ます。一つ一つのブロックが component と呼ばれるものでそれぞれが input と output を持っています。それぞれの component の左側にあるものが input で、右側にあるものが output です。それでは、このスクリプトを作るプロセスを順に追って見ましょう。

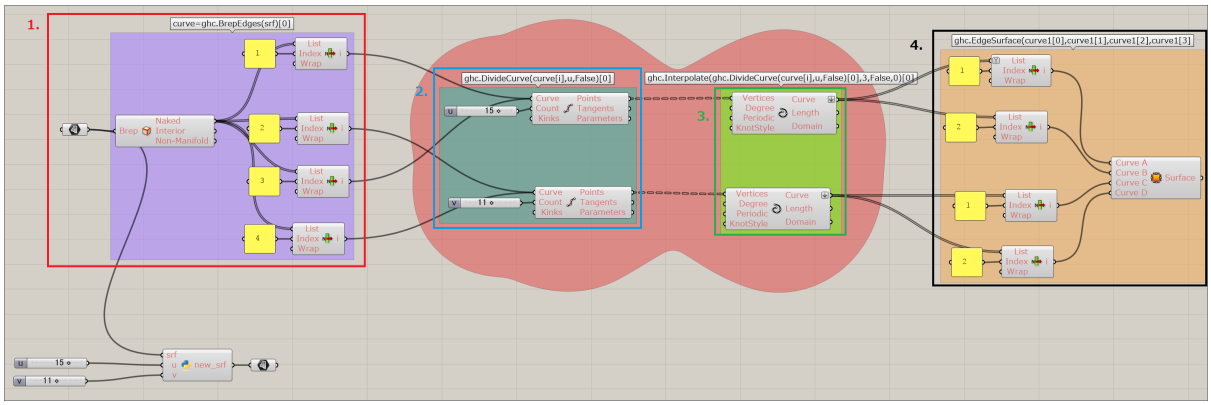

図 1: 全体像

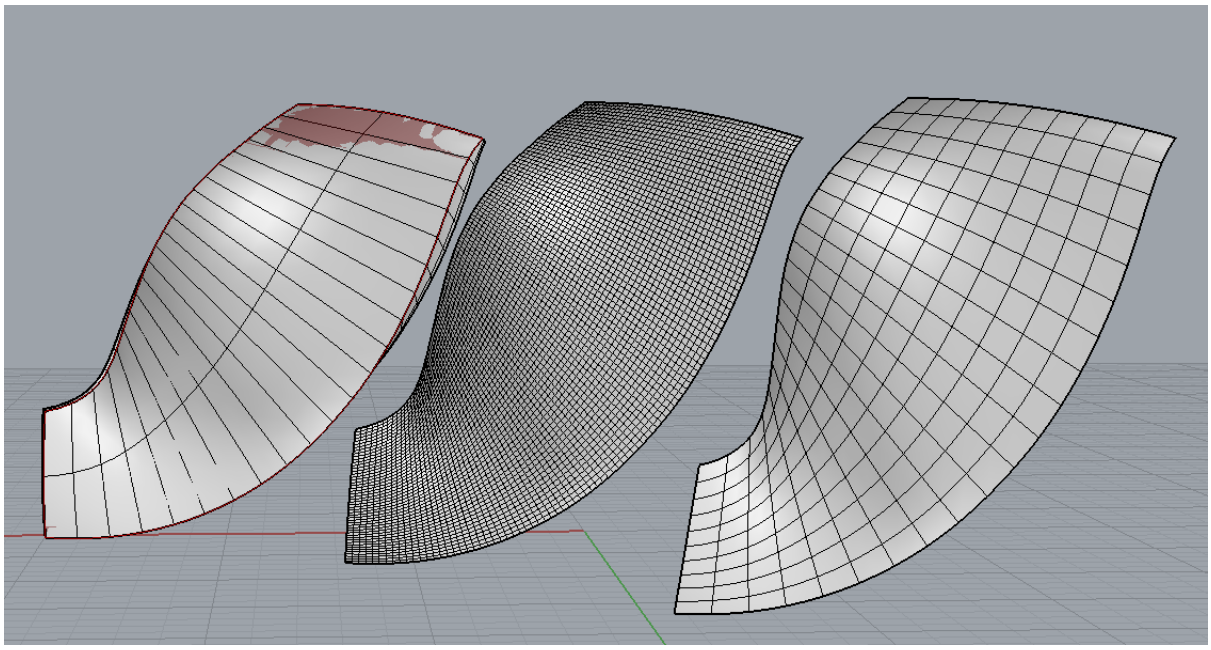

図 2: Surface の要素数を変更できる

1.1 Component を使って再構築する

1.1.1 Surface から Curve を抽出

まず編集したい Surface を Grasshopper にパラメータとして input しなければいけません。パラメータはすべて Params のページでまとめており、Surface パラメータをキャンバスに配置します。
Params の中にも Geometry、Primitive、Special の三種があります。

- Geometry はインプットデータの取り込みができ、複数のコンポーネントに同じインプットデータ (線や平面) を用いるときに便利です。
- Primitive は数値などを扱うことができます。
- Special は数値を自由に変化させる slider やデータ (点集合の xyz など) を表示するなど、いろいろなことができます。

図 3 のように配置した Surface パラメータを右クリックするとコンテキストメニューが表示されるので、'Set one Surface' を選択します。Rhinoceros 内で指定したい Surface をクリックし、Surface パラメータがオレンジから白へと変化すれば、入力に成功です。

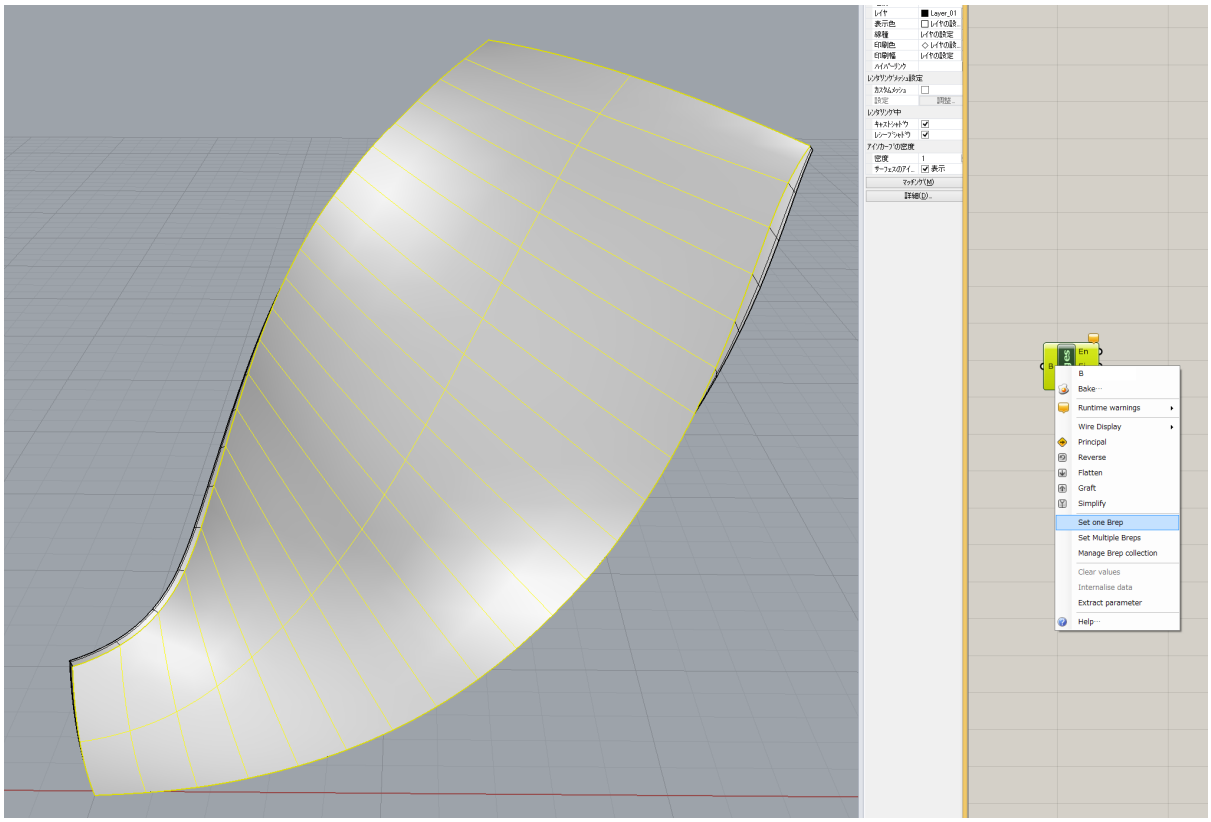

図 3: surface の選択

Tip1: Component がどこにあるのかを探したいとき:

Ctrl+Alt を押して、マウスの右上に information マークが表示されたのを確認した後、component を押すと、自動的に Component のあるページと場所を矢印で示してくれます。各自 gh ファイルを開いてこの方法でチェックしてみてください (これ以降の Component の場所は説明しません)

Tip2: Component を挿入したいとき:

キャンパスの開いているところをダブルクリックすると、component 検索ウインドウが出ます。component の名前を入れると、上に表示するのでクリックしてキャンパスに配置します。

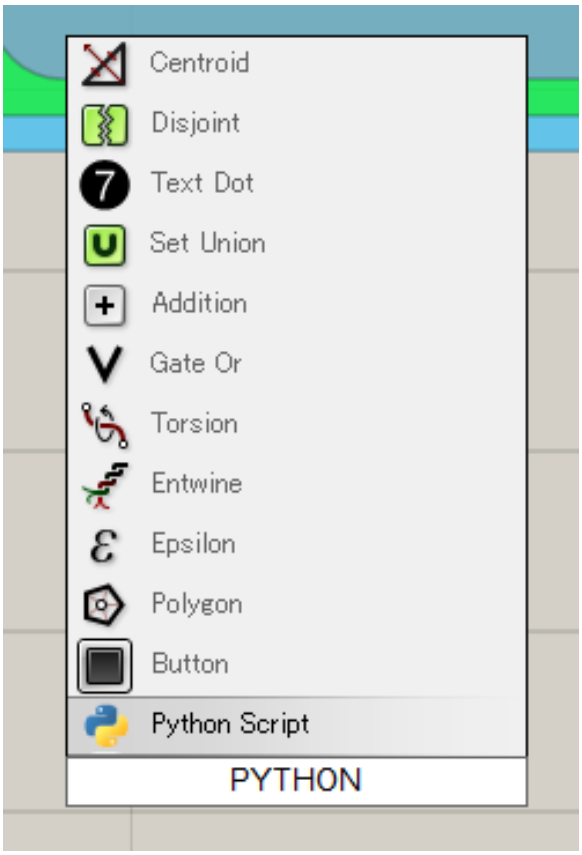

図 4: component の検索

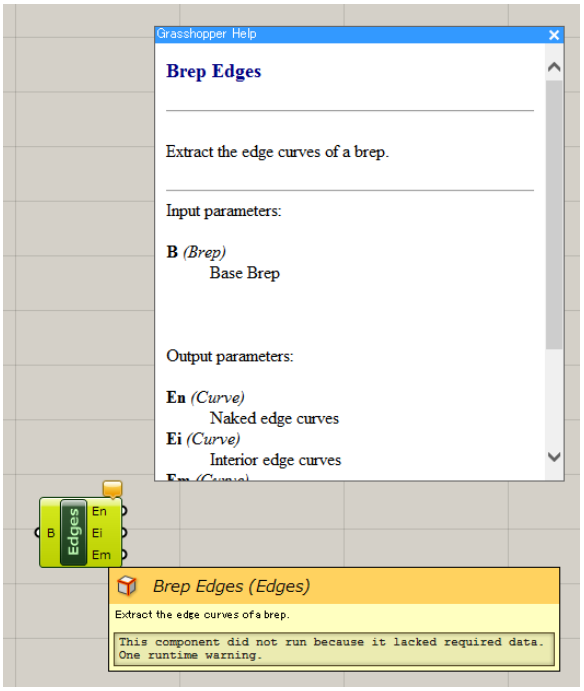

図 5: component のヘルプ

Tip3: Component がどのような使い方をするのか調べたいとき:

Component を代表している図形 (または文字) を右クリックし、Help を選択する。新しい Component を使う前に必ず確認しましょう。同様に input と output の入力形式を調べたいときは、カーソルを調べたものの上に右クリックし help を選択する。

Tip4: Input type

Component に入力がある場合、Grasshopper は入力するデータタイプの認識ができないので、手動で設定する必要があります。Component 上に接続する変数の名前のところを右クリックし、Data type から適切なものを選びます。

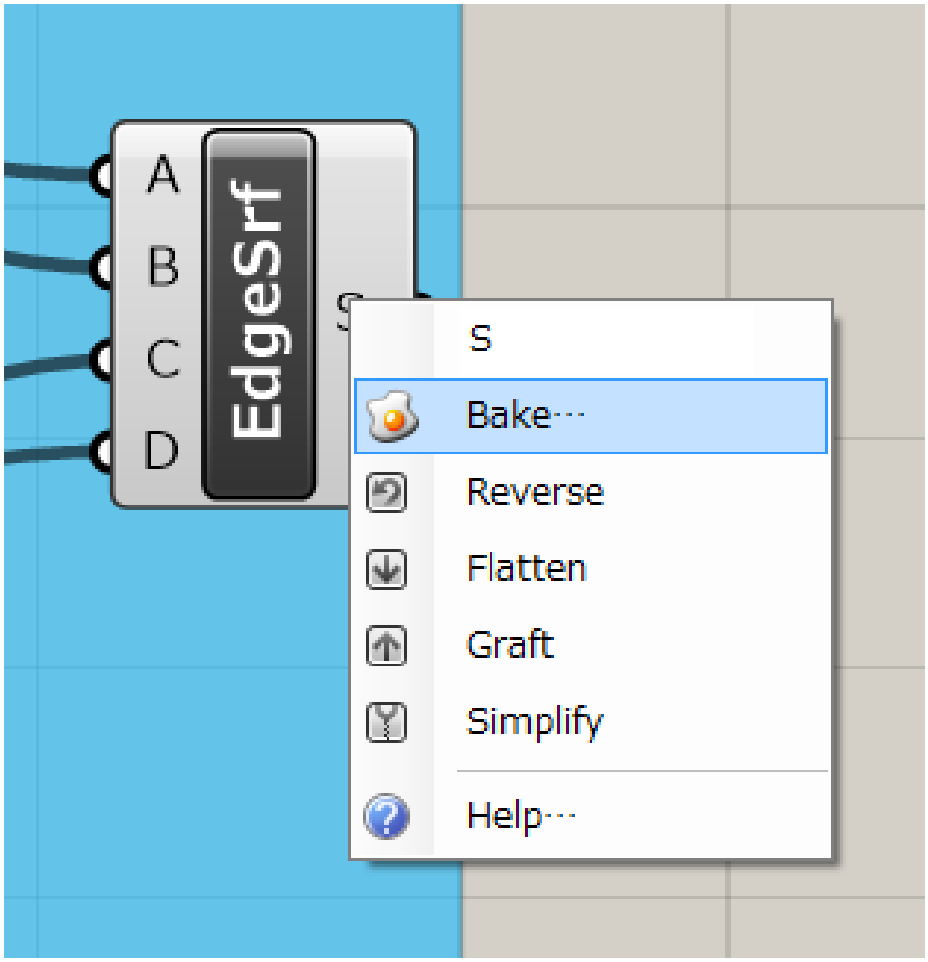

図 6: bake

図 7 のように、Brep Edges の Component を配置し、input を先ほどの Surface パラメータと接続します。これによって、Surface のエッジ (4 つの Curve) がひとつの List として出力されます。List から要素を抽出するときは、List item を使います。List を入力し、Index で要素番号を指定するとその要素が出力されます。Index の入力は、カーソルを index の文字上において右クリック、'Set an integer' で数字を入れる方法。または、何の数字を入れたのか後でわかりやすくできるように、Panel Component を使用する方法があります。

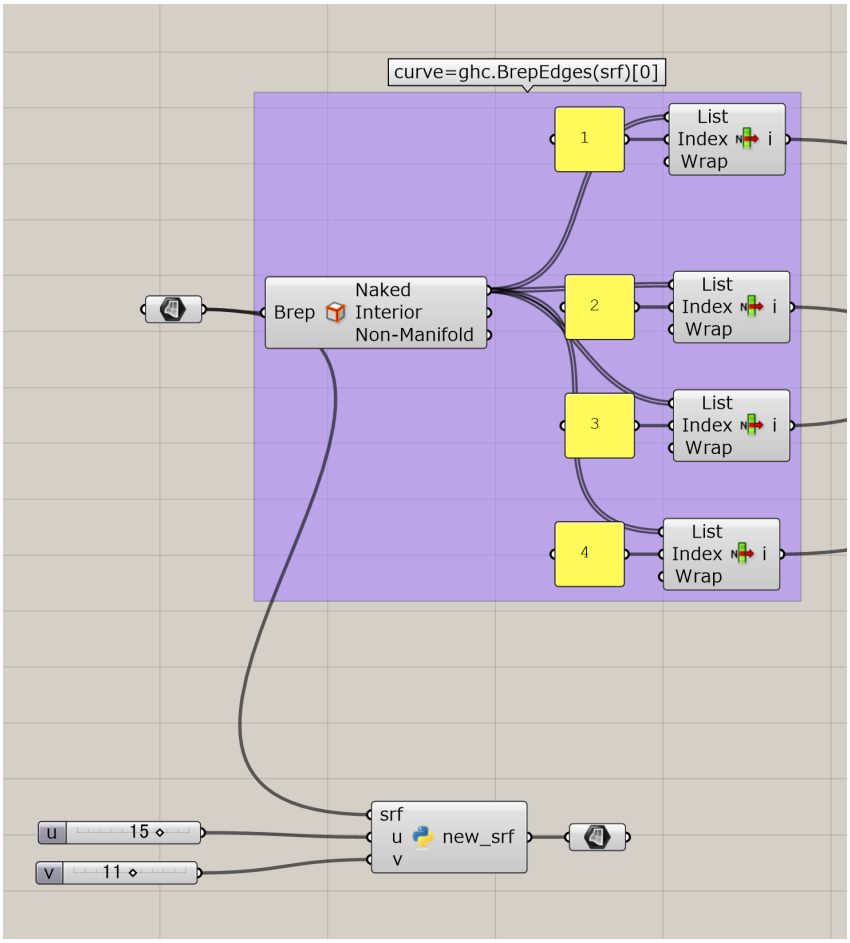

図 7: 1.1.1 節の components

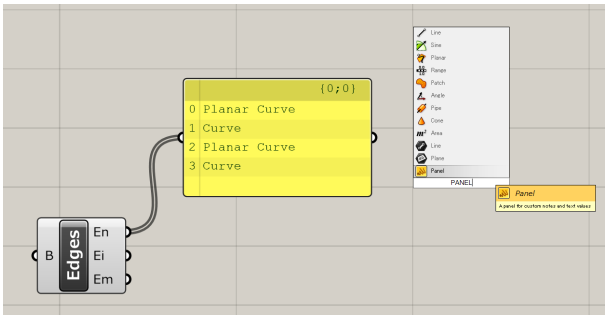

図 8: Panel の挿入

Tip5:

データが一つの要素のとき、component 同士をつなぐ線が実線になります。List である場合は二重線、Tree Data (複数の List) である場合は点線となります。

1.1.2　Curve 上を通る点群を作成

　ここで使う Component: Divede Curve
Divide Curve で分割数を指定し、分割点を出力してくれます。u,v の二方向があるので、二つ Divide Curve Component を配置します。
図 9 のように Brep Edges で出力される Curve は時計回りに配置しているので、奇数番目の Curve を u 方向、偶数番目の Curve を v 方向にまとめて入力します。複数のデータを一つの場所に線を繋ぐときは shift 押しながら線を繋ぎましょう。
Divide Curve の Count 入力に Slider をつなぎます。

Tip6: Slider

Slider は型 (int、double) の指定、範囲 (Range) の指定をし、つまんで動かすことで、数値の変更に合わせたジオメトリの変化をリアルタイムで見ることができます。型、範囲の指定は slider を右クリック、Edit でできます。

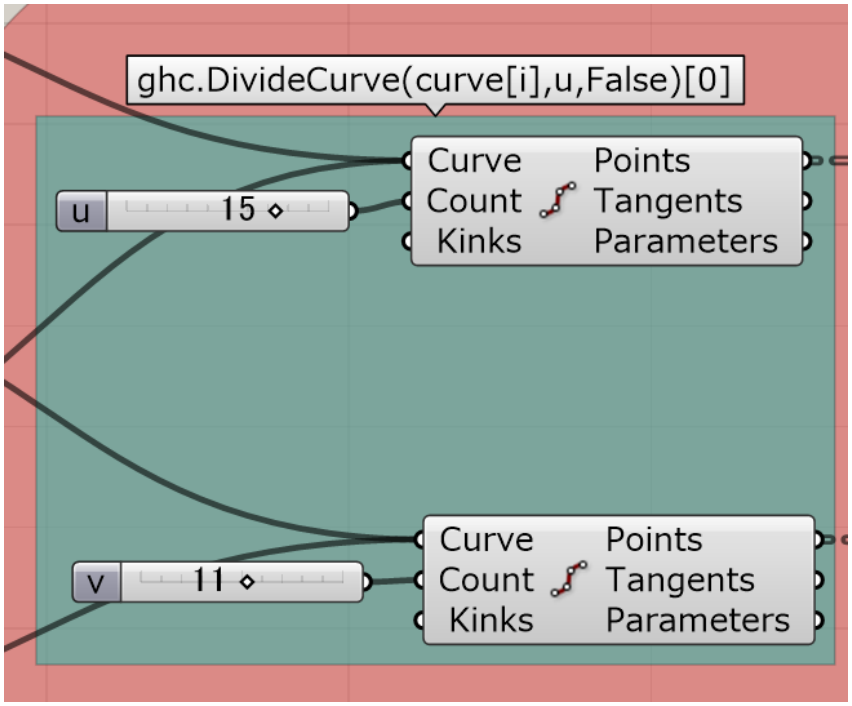

図 9: 1.1.2 節の components

1.1.3　点群を通る NURBS Curve の構築

　ここで使う Component: Interpolate
Point データを Interpolate Component に繋ぎます。Degree,Periodic,Knotstyle は入力がないとデフォルト値で入力されます。Interpolate は入力された Point データで Curve を作成してくれるものです。二つの List で Point データを入力したので、出力も二つの List となります。出力されるすべての要素を一つの List に入れたいときは、出力される場所にカーソルを合わせ、右クリックして、Flatten を選択しましょう。

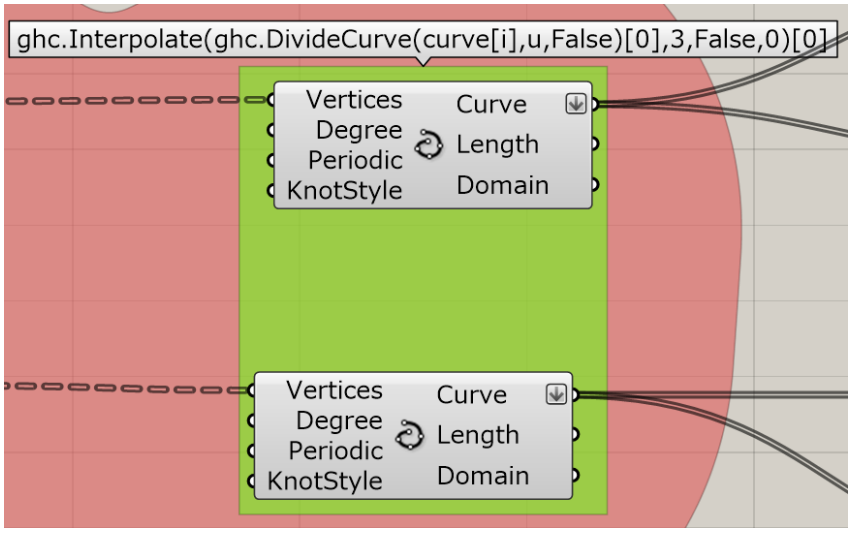

図 10: 1.1.3 節の components

1.1.4　4 つの NURBS Curve から Surface の構築

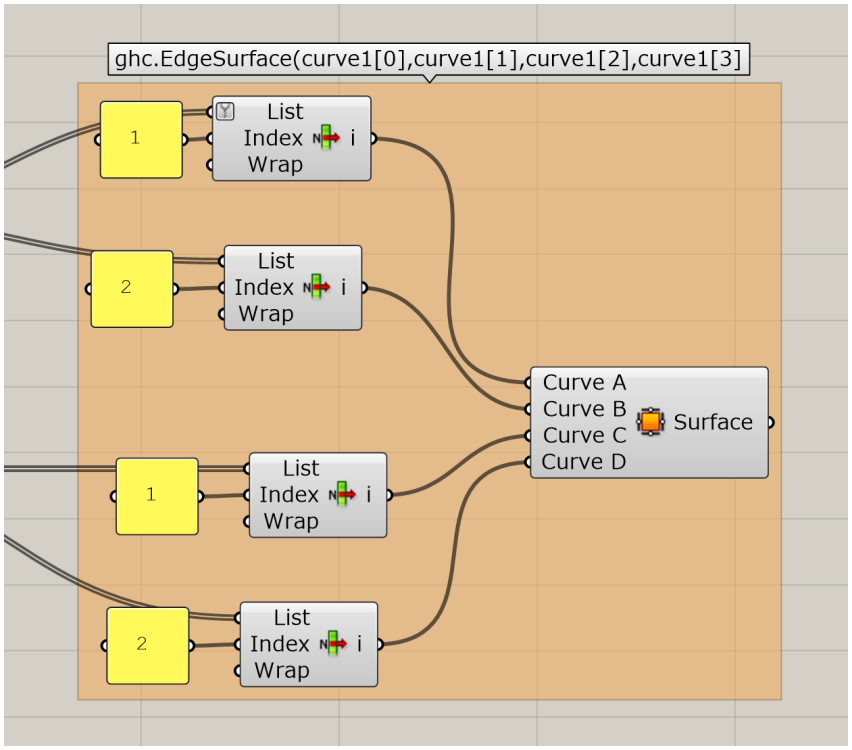

図 11: 1.1.4 節の components

　ここで使う Component: Edge Surface
Edge Surface はエッジである Curve を入力すると、Surface を構築してくれる Component です。先ほどの Interpolate の出力は一つにつき、二つの Curve が入っているので、List item でそれぞれ出します。Edge Surface の出力を右クリック、Bake を選択すれば完成です。

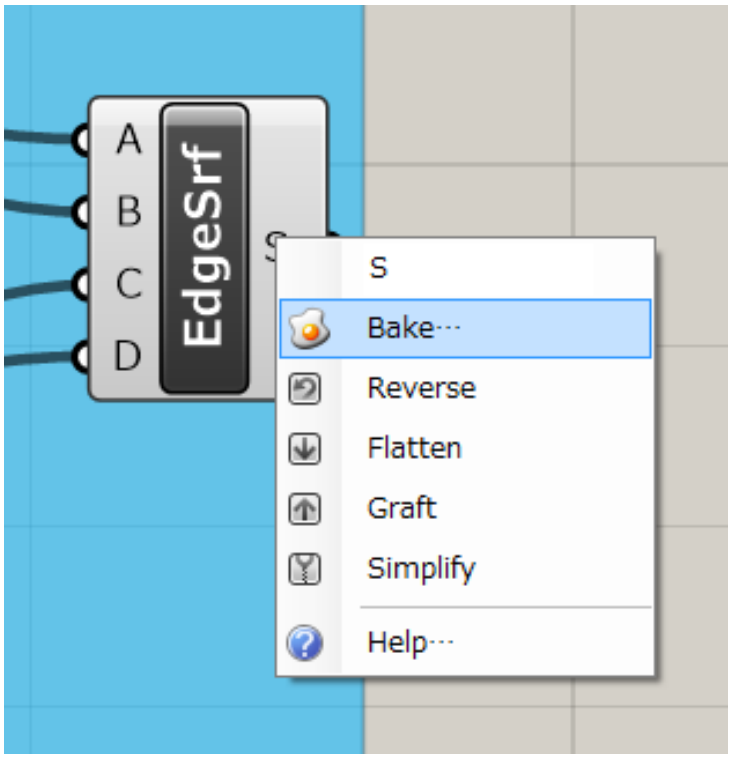

図 12: Bake のやり方

1.2 Ghpython による作り方

Ghpython は、Grasshopper 内で Python によるプログラミングで図形の編集を可能としたツールです。先ほどのプロセスと同じもので、プログラミングしたのが以下のコードとなります。

```
import ghpythonlib.components as ghc
curve=ghc.BrepEdges(srf)[0]
curve1=[]
for i in range(0,4):
  if i\%2==0:
    curve1.append(ghc.Interpolate(ghc.DivideCurve(curve[i],u,False)[0],3,False,0)[0])
  else:
    curve1.append(ghc.Interpolate(ghc.DivideCurve(curve[i],v,False)[0],3,False,0)[0])
new_srf=ghc.EdgeSurface(curve1[0],curve1[1],curve1[2],curve1[3])
```

Grasshopper の Component を使いたいときは、ghpythonlib.components を導入する必要があります。

```
import ghpythonlib.components as ghc
```

Component を呼び出すときは、ghc. を打ち、その後に component の名前をいれます。

```
curve=ghc.BrepEdges(srf)[0]
```

BrepEdges は、三つの出力があります。Naked, Interior, Non-Manifold. 今回は Naked の出力が欲しいので、後ろに [0] をつけます。
Component の正式な名前は、help を調べたときに表示しています。Ghpython 内の各 Component の詳細な使い方は、ghc.component 名を打ち終わったときに、code 入力ウィンドウの下に自動的に表示します。

偶数番目のカーブを u とし、奇数番目のカーブを v とします。

```
ghc.DivideCurve(curve[i],u,False)[0]
```

がカーブを分割してポイントをアウトプットしています。その結果を Interpolate の入力として使います。

```
ghc.Interpolate(ghc.DivideCurve(curve[i],u,False)[0],3,False,0)[0]
```

そして Inerpolate して出来上がったカーブを curve1 という list に append します。

```
curve1.append(ghc.Interpolate(ghc.DivideCurve(curve[i],u,False)[0],3,False,0)[0])
```

四つのカーブが for で循環し終えた後、EdgeSurface を使って新しい面を作ります。

```
new_srf=ghc.EdgeSurface(curve1[0],curve1[1],curve1[2],curve1[3])
```

new_surf を Bake すれば完成です。

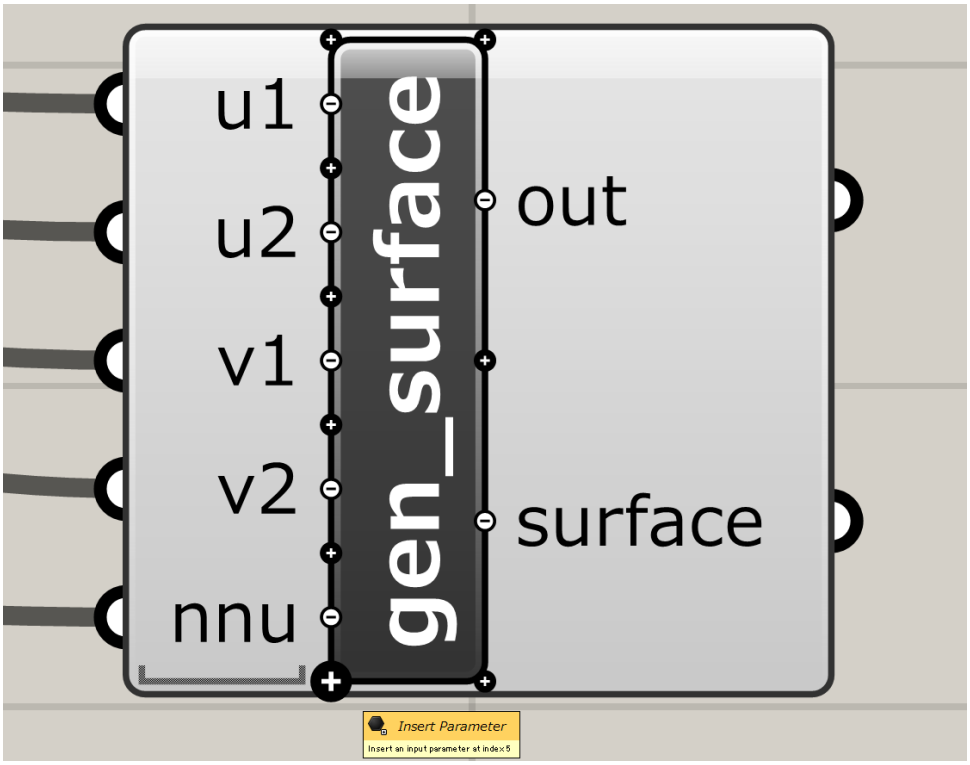

図 13: Bake のやり方