Computational Fabrication

CS 491 and 591
Professor: Leah Buechley
https://handandmachine.cs.unm.edu/classes/Computational_Fabrication_Spring2021/

Artist: Iris van Herpen

https://www.irisvanherpen.com/

2011, Iris van Herpen

Iris van Herpen

Iris van Herpen

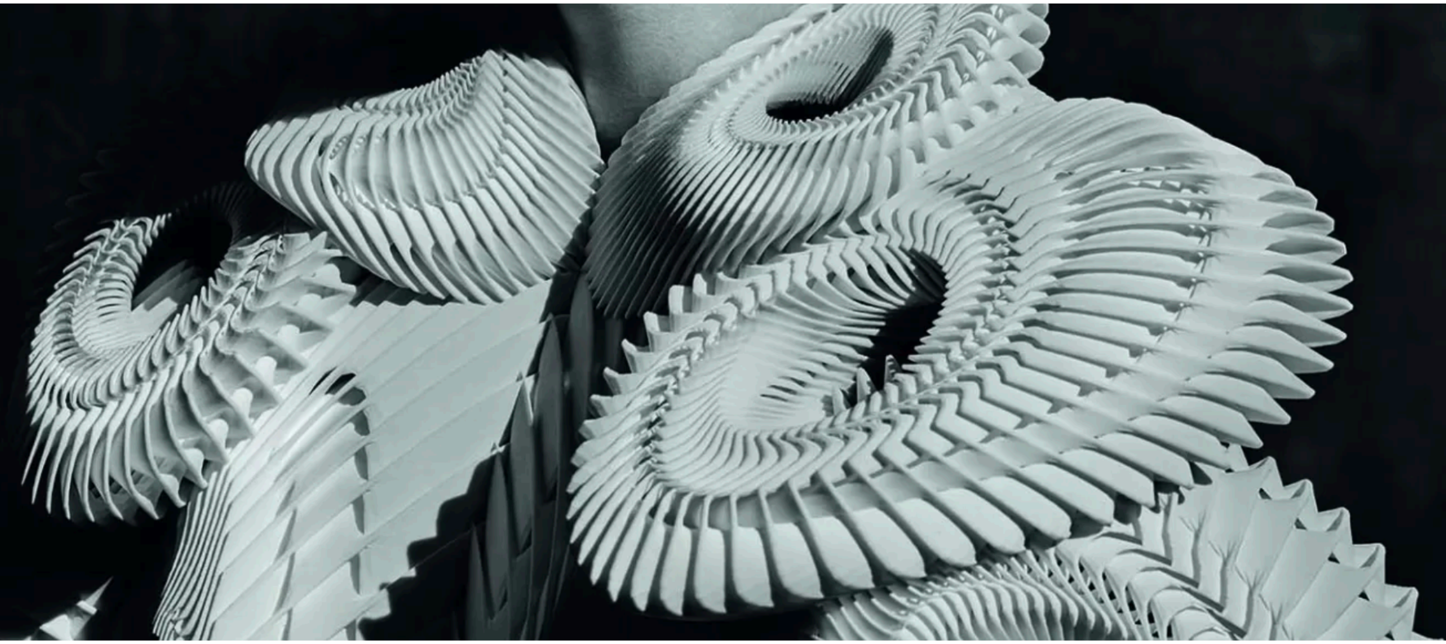
Iris van Herpen +
Rogan Brown





Iris van Herpen

mad



Les Arts Décoratifs   Musée des Arts Décoratifs   Exhibitions   Current exhibitions

IRIS VAN HERPEN. SCULPTING THE SENSES

from 29 November 2023 to 28 April 2024

https://madparis.fr/Iris-van-Herpen-Sculpting-the-Senses

Held at the Musée des Arts décoratifs, from 29 November 2023 to 28 April 2024, the exhibition *Iris van Herpen.*

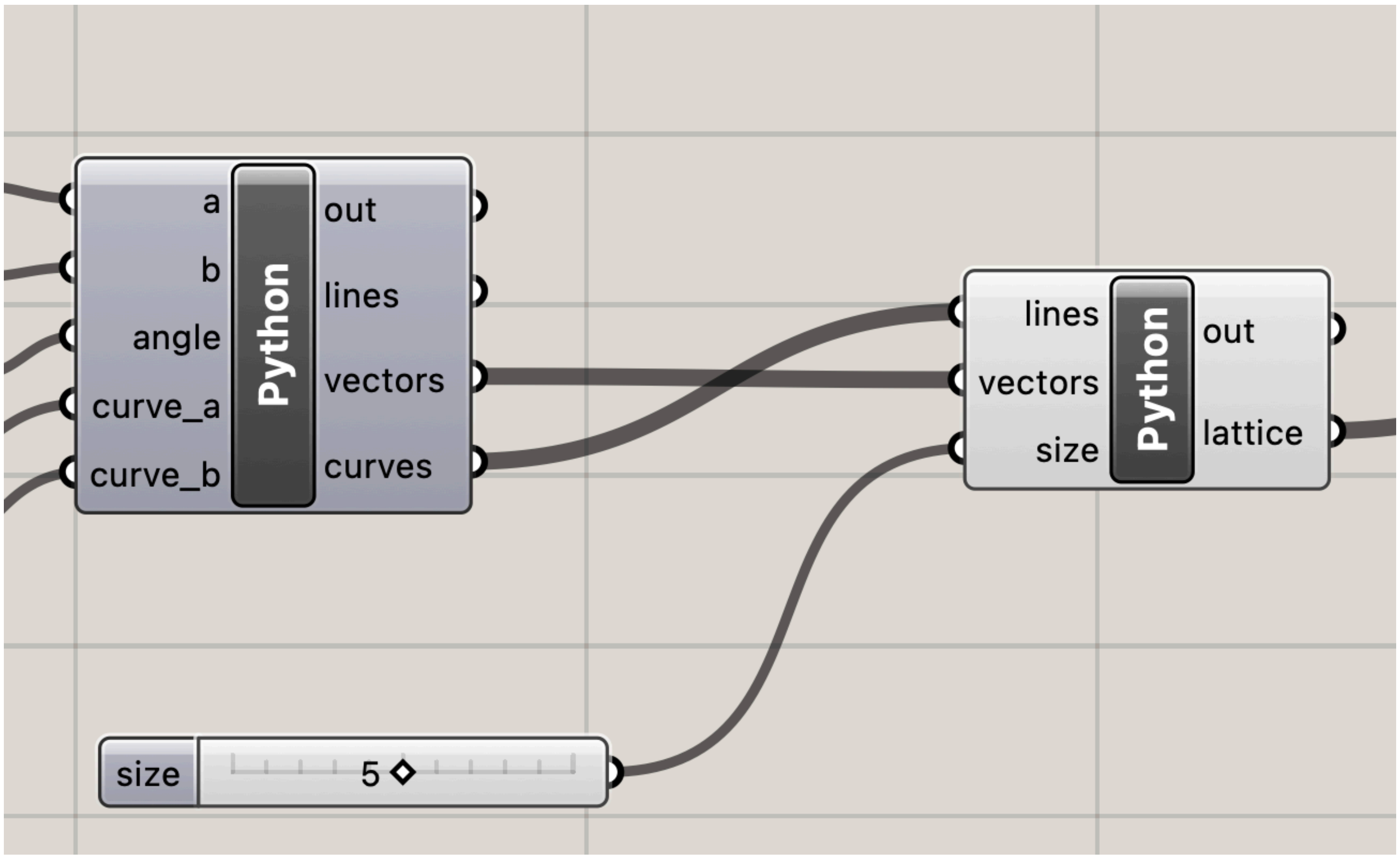
TICKETS

What we did last class

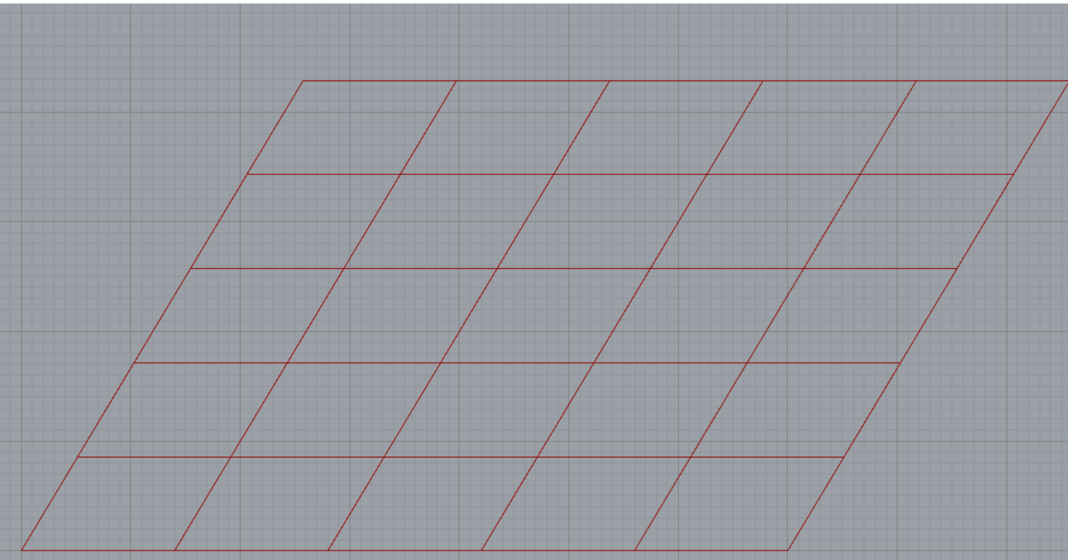
1. Write code to generate 2D lattices, illuminating some fundamental tiling geometry
2. Use our lattice generating code to generate 2D tiles and tilings

Open up your project from last class. First open the Rhino file with your curves. Then open your GH file.

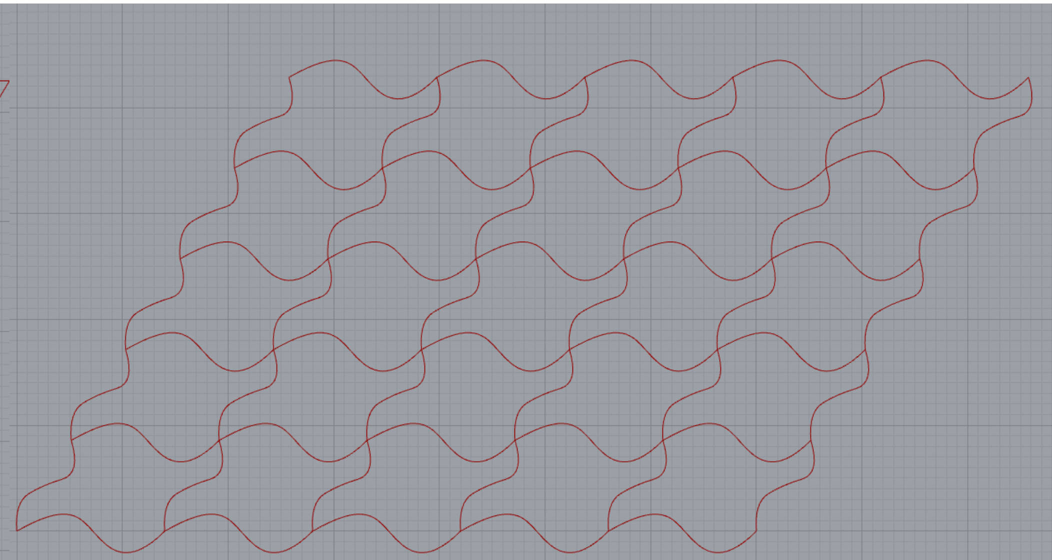
Connect Escher Curves to Tiling Code



Connect Curves to Tiling Code



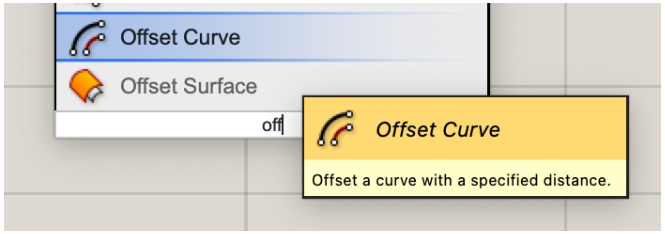
Tiling with lines

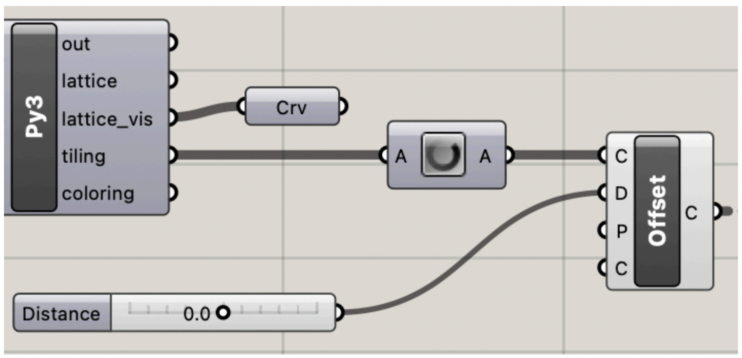


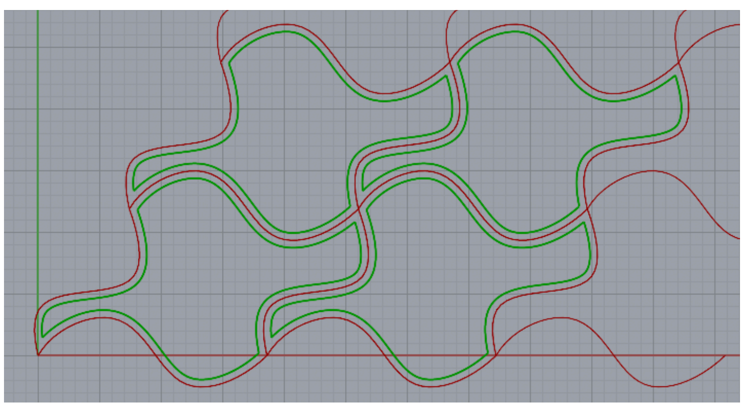
Tiling with Escher curves

Generating Printable 3D Tiles

Offset Tile Shape for Physical Tiling

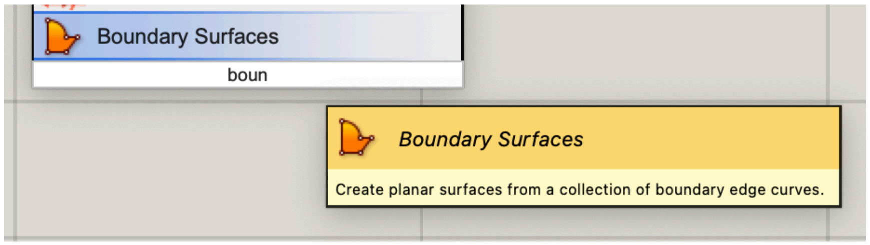


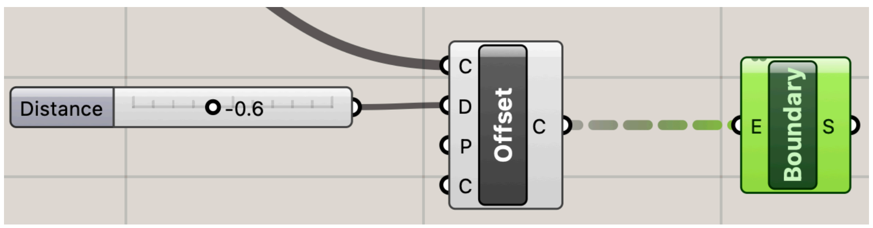


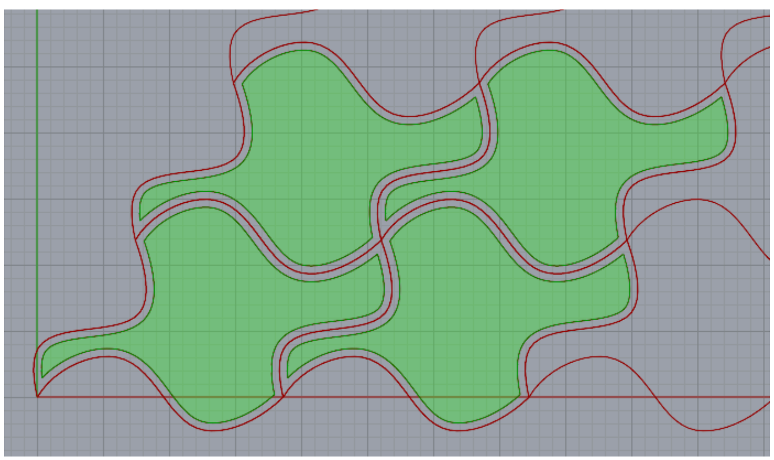


1. Create an **Offset Curve** GH block
2. Connect your tiles to the C (curve) input
3. Create a float number slider
   Range of number slider: -3.0 to 3.0
4. Connect number slider to the D (distance) input of Offset Curve block
   Negative number: offset in
   Positive number: offset out

Create Tile Surface

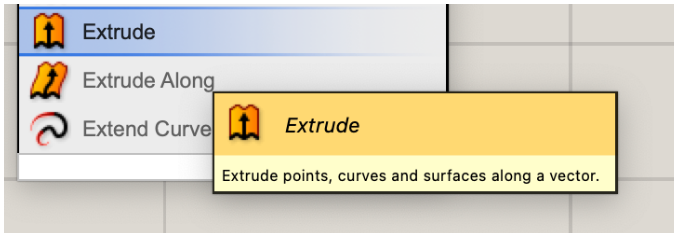


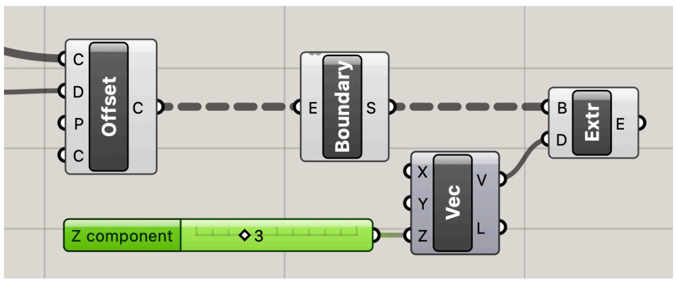


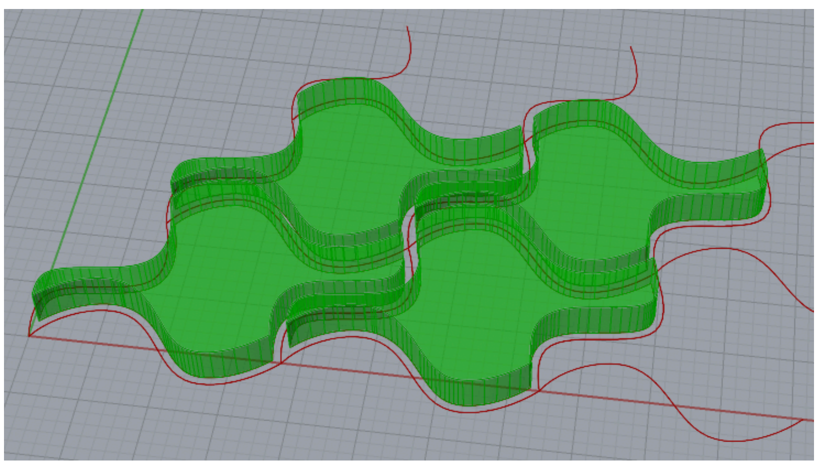


1. Create a **Boundary Surfaces** GH block
2. Connect the C output from Offset to the E input to Boundary

Extrude Surface to Generate 3D Tile



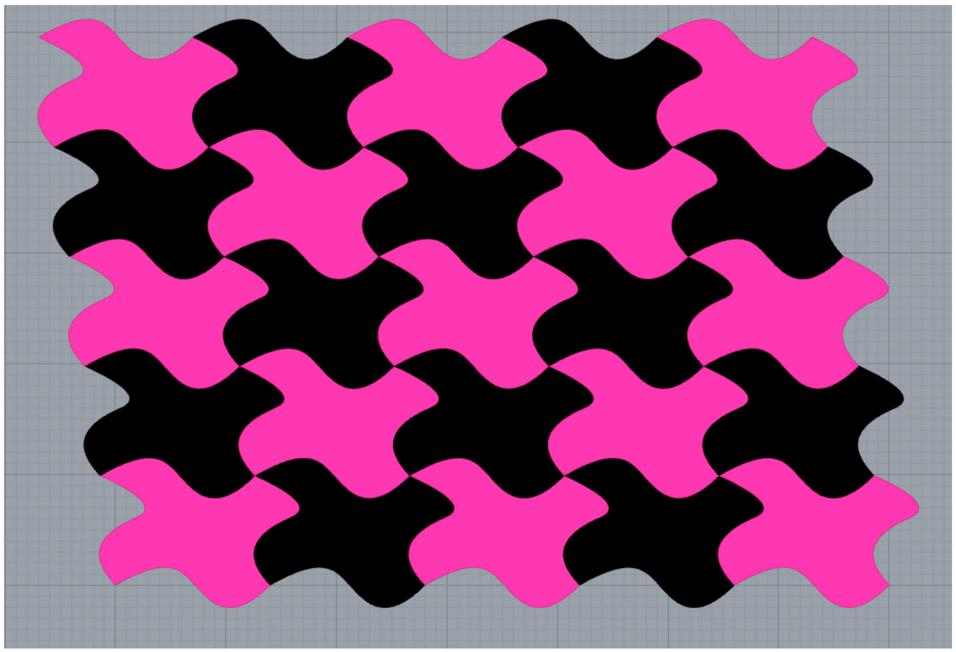




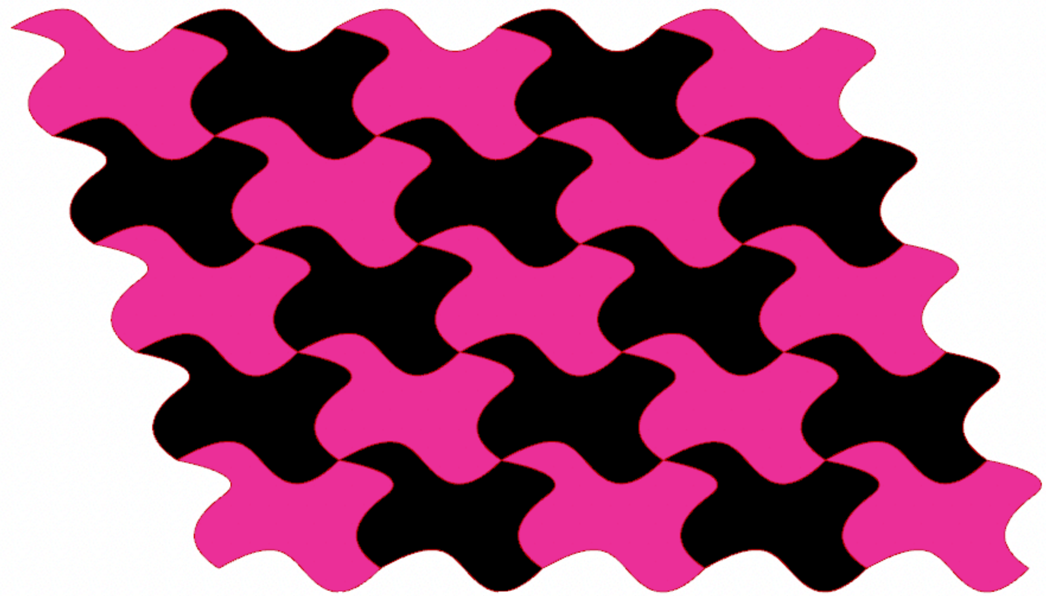
1. Create an **Extrude** GH block
2. Create a **Vector** GH block and provide a number slider input for Z.
3. Connect the S output from Boundary to B on Extrude and the V output from Vector to D

questions?

Add Some Color



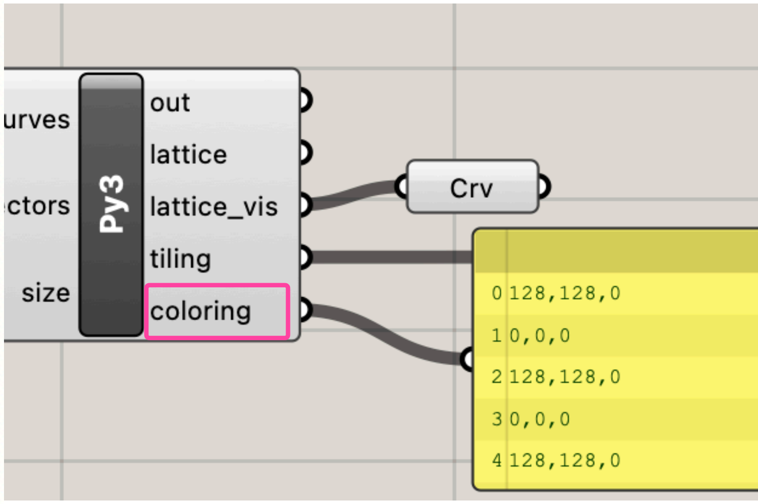
Wireframe view in Rhino



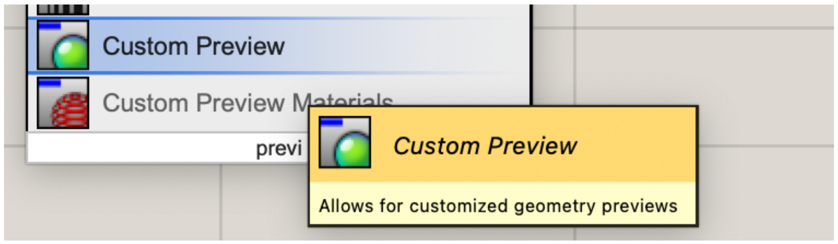
Rendered view in Rhino

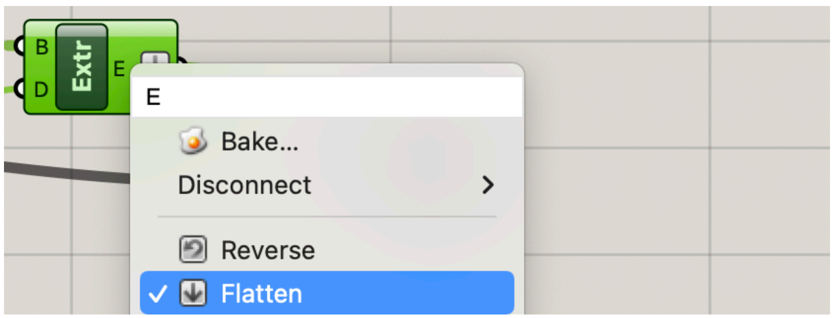
2nd Python Block

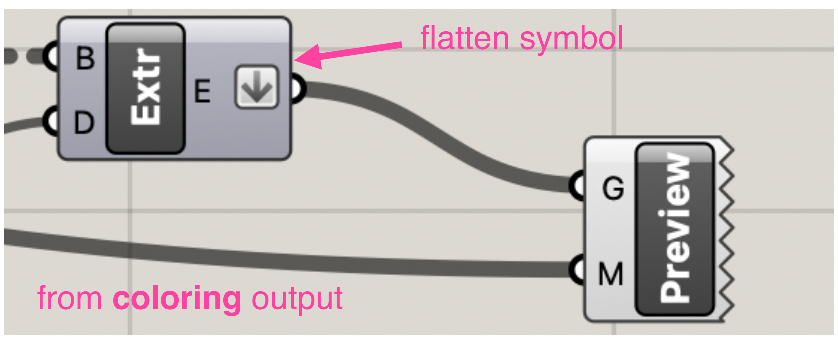
```
tiling = []
coloring = []
for i in range (len(lattice)-1):
    for j in range (len(lattice)-1):
        bottom_left = lattice[i][j]
        top = rs.ExplodeCurves(lattice[i+1][j])[0]
        right = rs.ExplodeCurves(lattice[i][j+1])[1]
        tile = rs.JoinCurves([bottom_left,top,right])
        if (rs.CloseCurve(tile)):
            tile = rs.CloseCurve(tile)
        else:
            print("can't make a closed tile")
        tiling = tiling+tile
        if (i%2==0 and j%2==0):
            coloring.append("128,128,0")
        elif (i%2==1 and j%2==1):
            coloring.append("128,128,0")
        else:
            coloring.append("0,0,0")
```



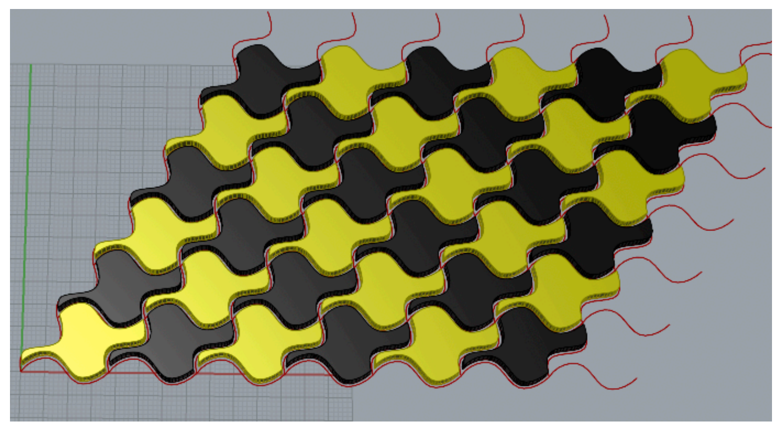
Add Some Color



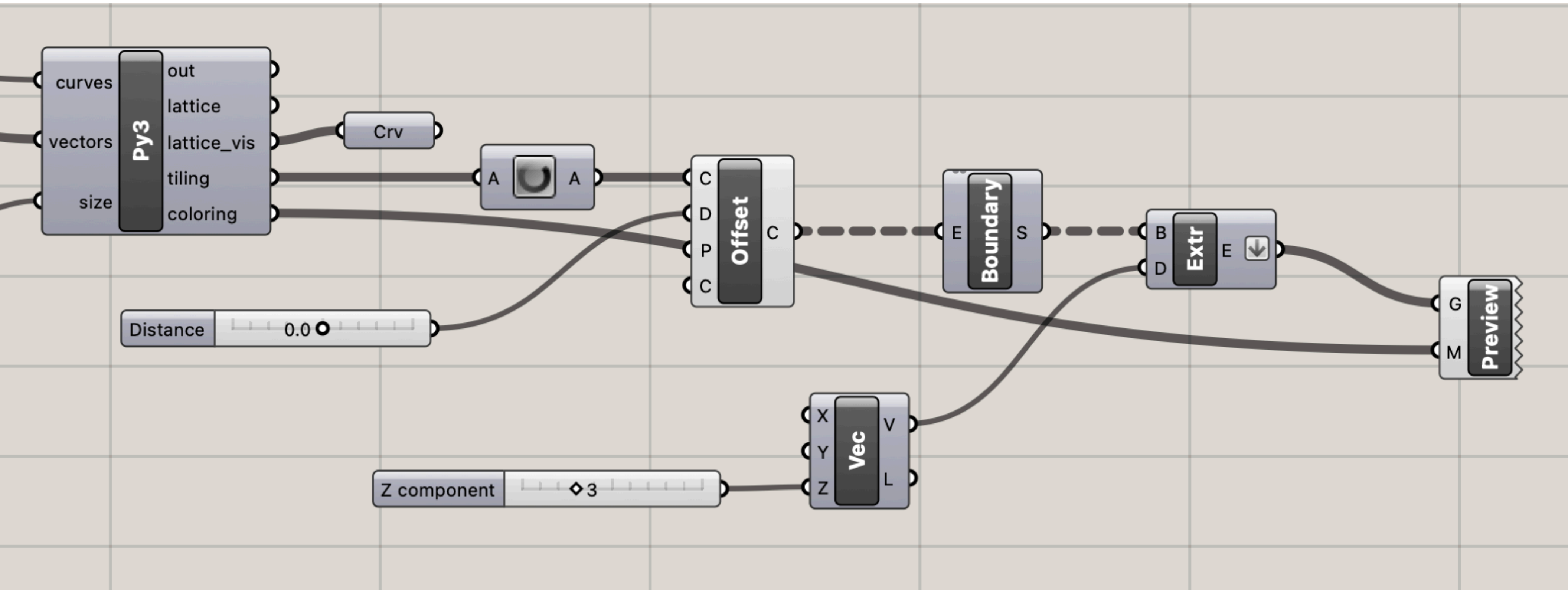




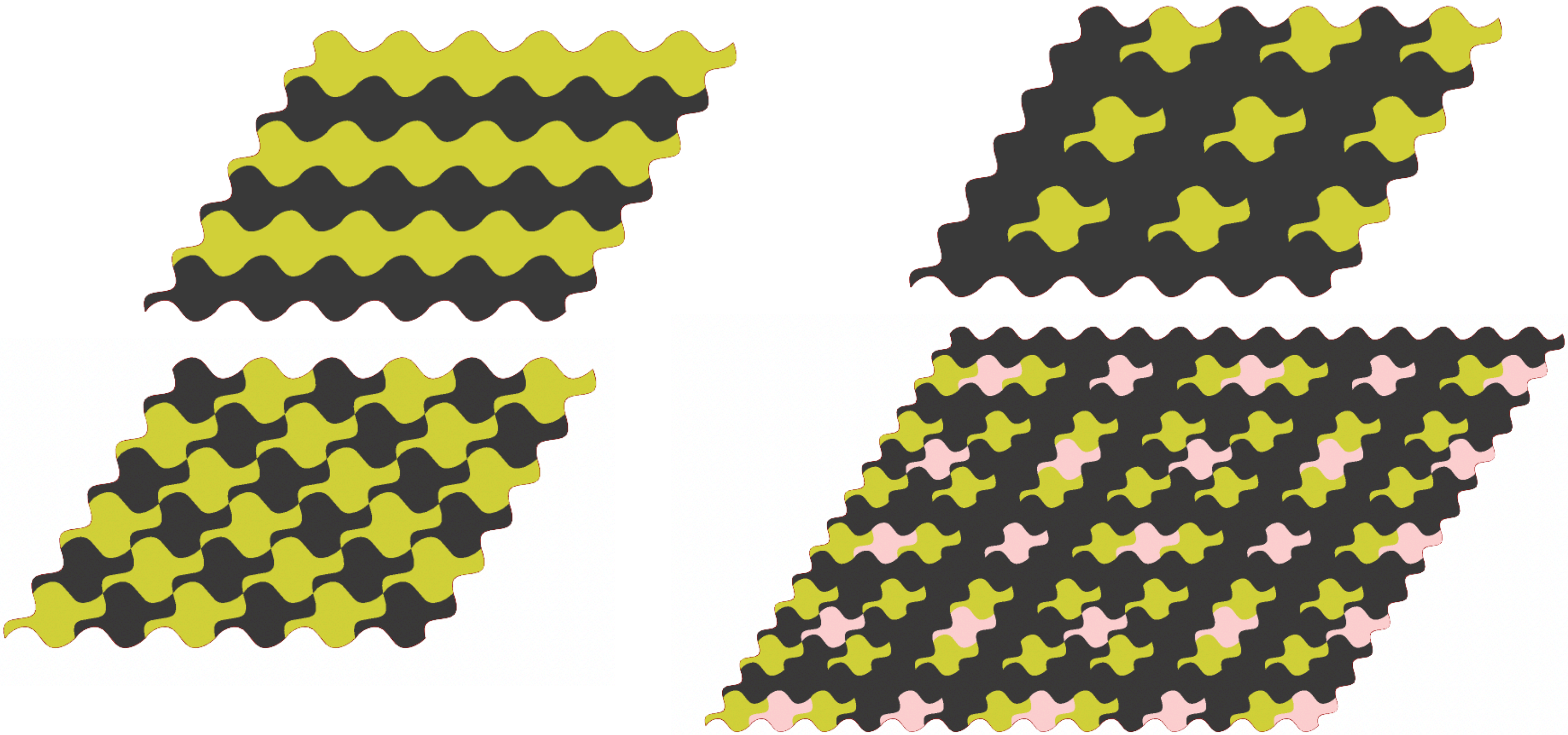
1. Create an **Custom Preview** GH block
2. Connect the output of the Extrude block to G (geometry) input. **Flatten** the output from Extrude.
3. Connect the coloring output to the M (materials) input.

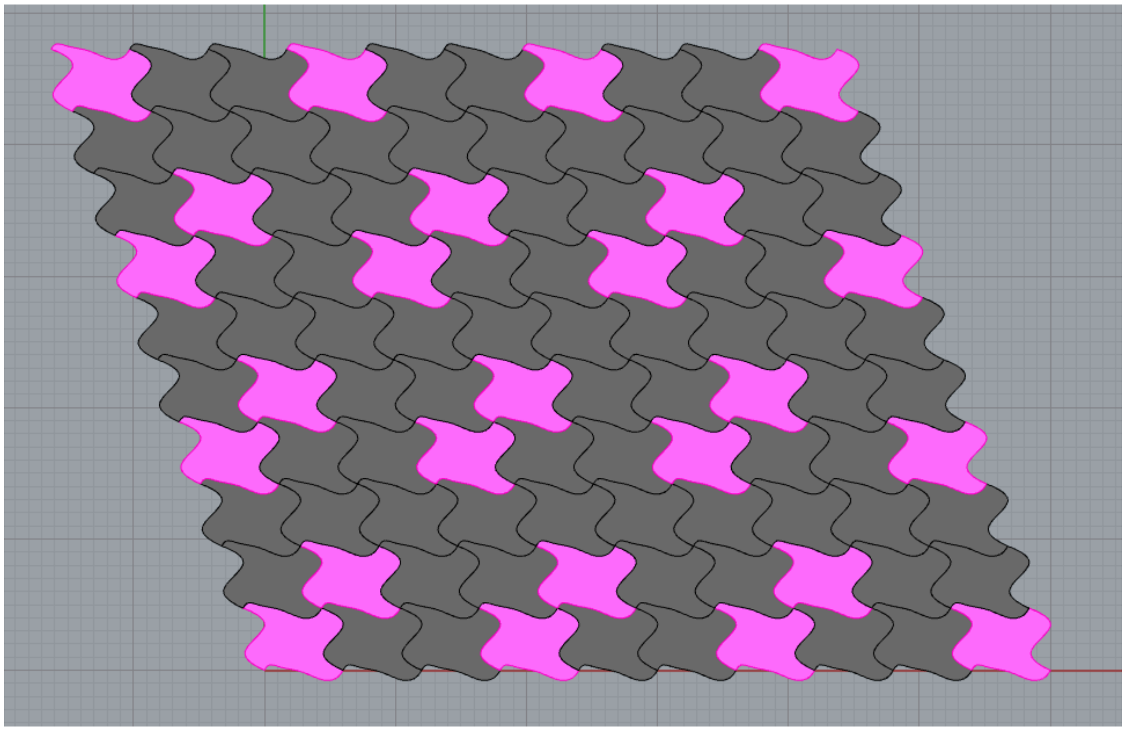
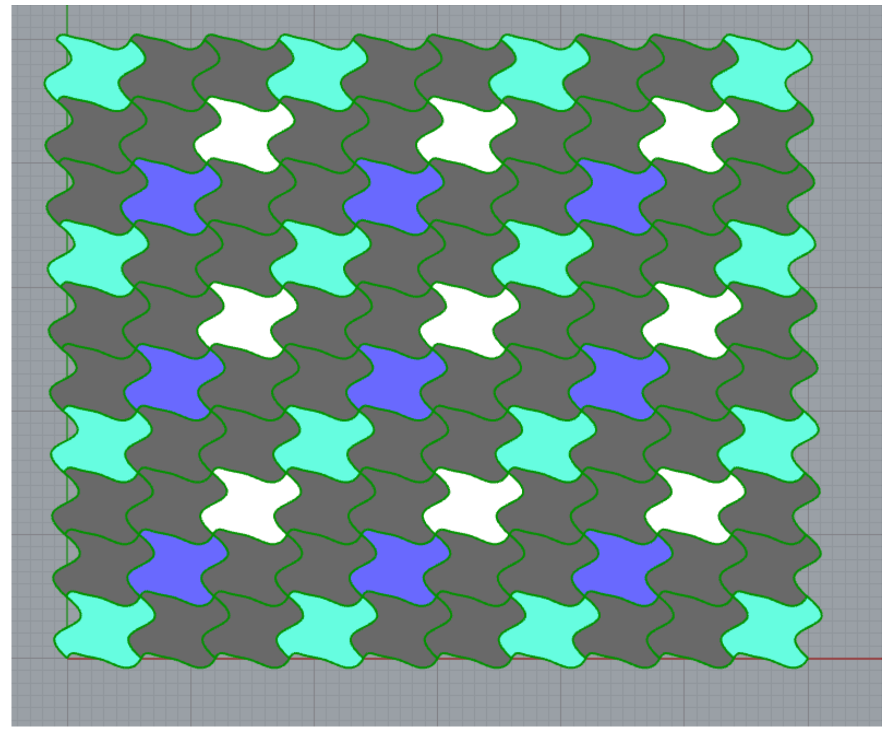


Add Some Color

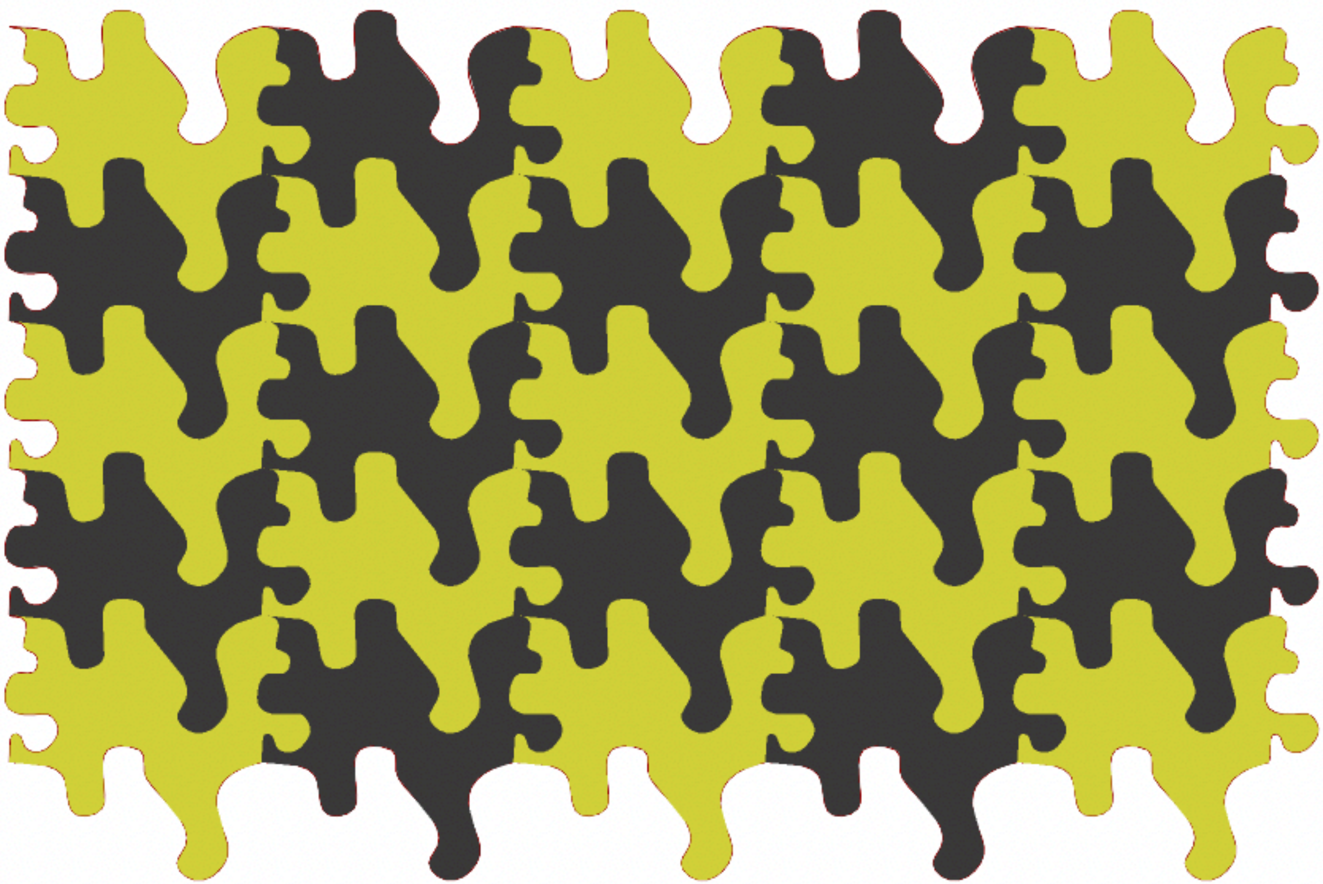


Play with Coding & Color Patterns



Play with Different Input Curves



questions?

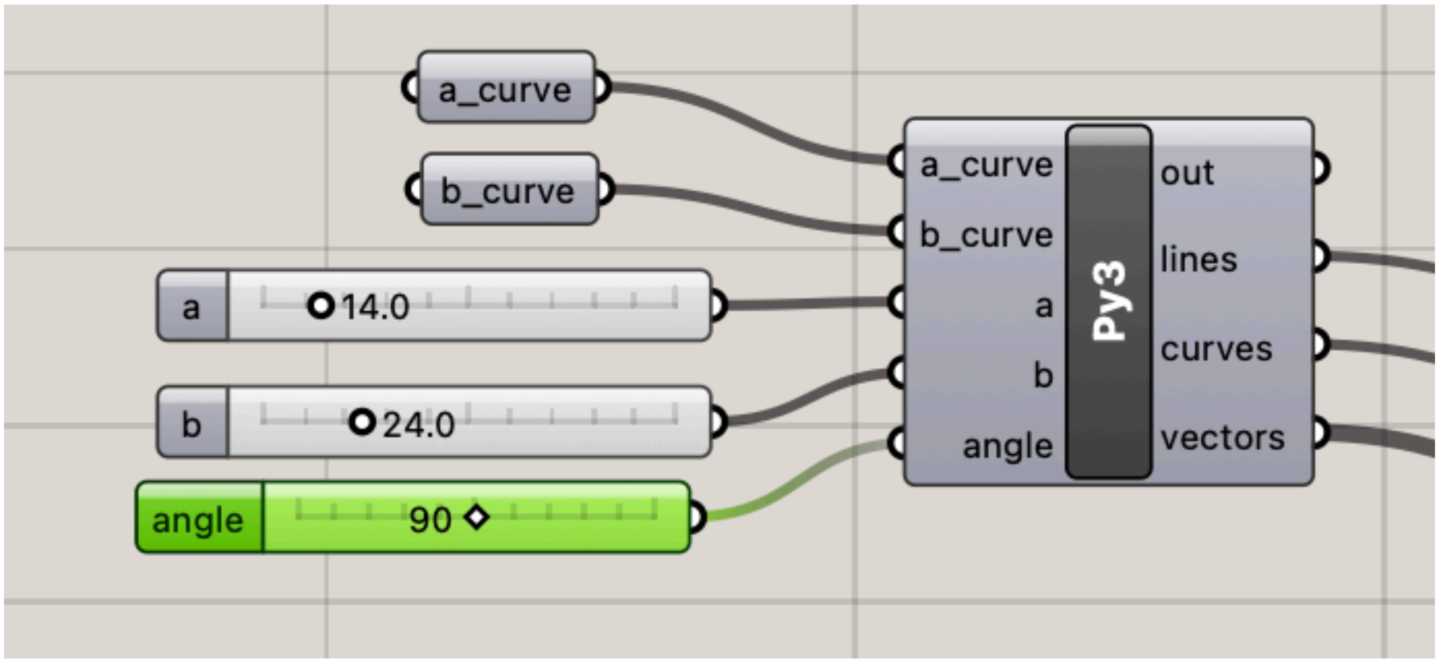
Save your GH file

Tiling Complex Surfaces

Now, we’ll morph our tiling across a surface

Note: can morph most simple **3D geometry** across most simple **surfaces**

Set your lattice angle to 90°

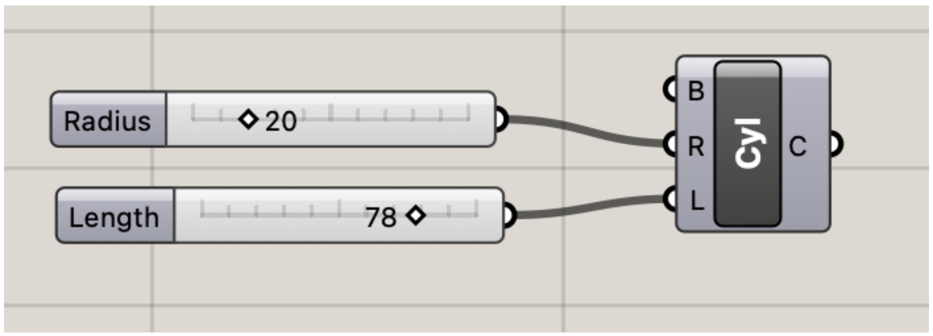


This will insure that the tiling fits perfectly across your surface.

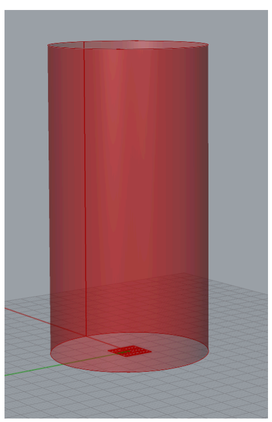
If you want to preserve this angle in your morph,
think about what you need to change in the code &/or talk to Leah.

Create A Surface to morph onto

We'll start with a cylinder

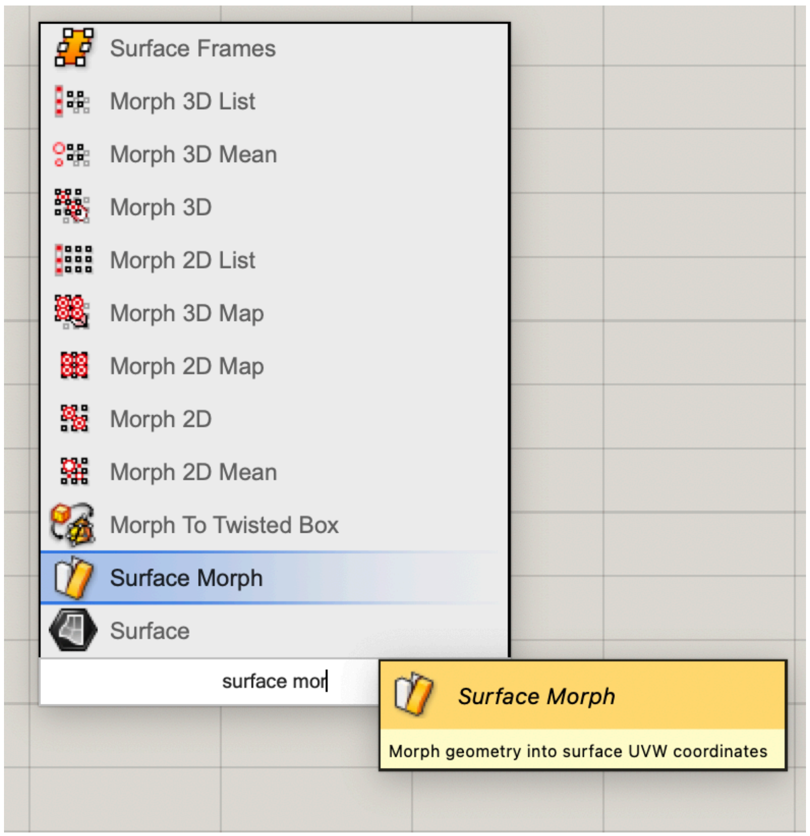


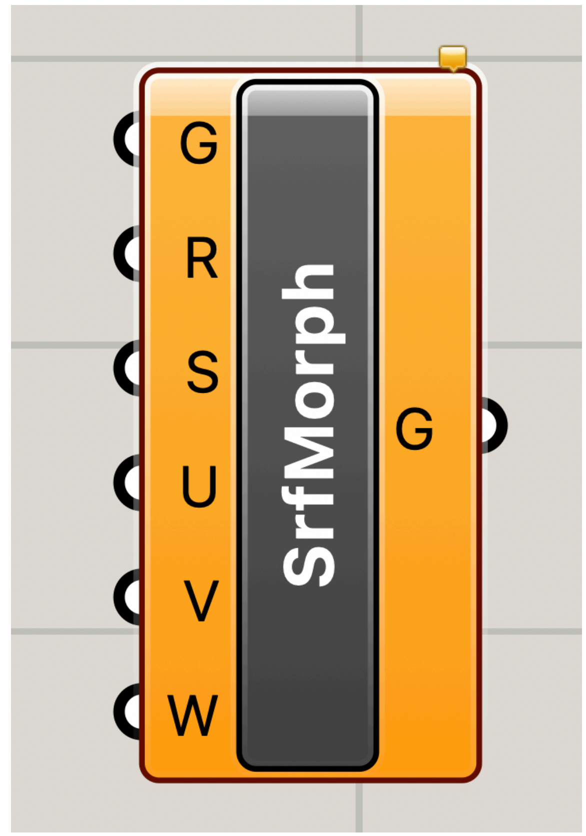
1. Create a **Cylinder** GH block
2. Create two number sliders.
   One for R (radius) +
   One for L (length/height)



questions?

Drag out a Surface Morph block

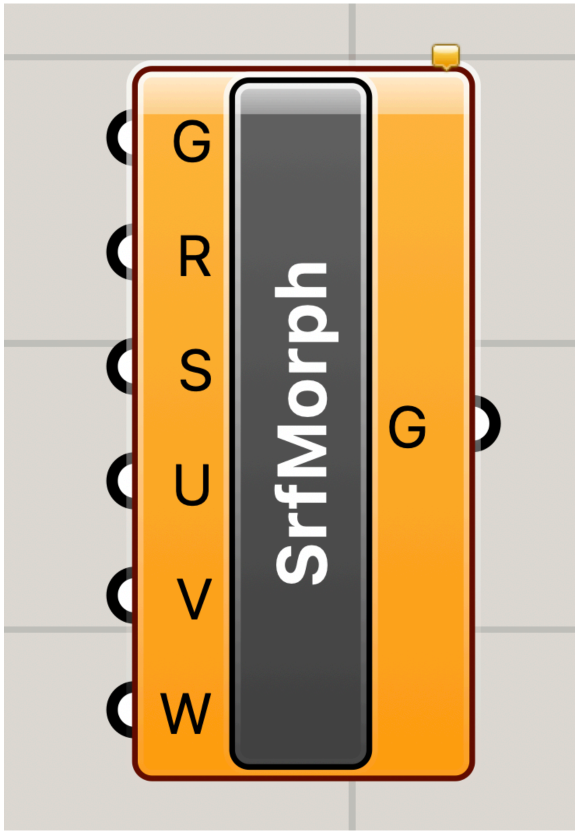




Surface Morph

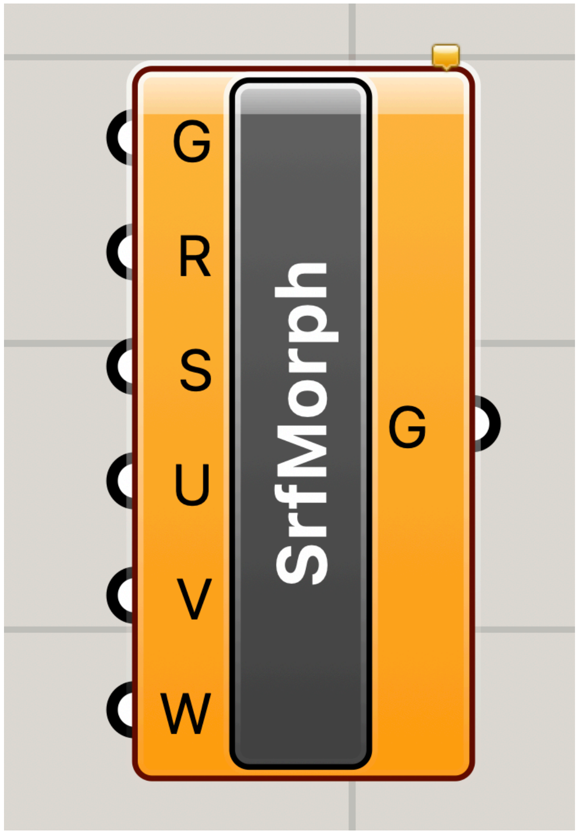
The surface morph block takes as input:

- A 3D geometry (G), we'll use our complete 3D tiling
- A size reference for the input (R), we will use the size of one basic lattice cell.
- A surface (S) to morph onto
- U,V,W = a size reference that determines how the geometry is stretched across the surface in the x(U), y(V), and z(W) dimensions
  We will express this size in terms of the dimension of one lattice cell.



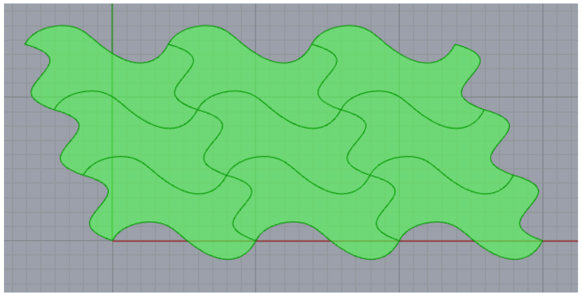
Surface Morph

- Outputs the input 3D geometry (G) stretched across the input surface (S)

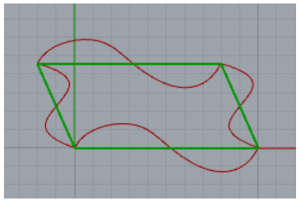


Surface Morph

- A geometry (G), we'll use our complete 3D tiling
- A size reference for the input (R), we will use the size of one basic lattice cell
- U,V,W = a size reference that determines how the geometry is stretched across the surface in the x(u), y(v), and z(w) dimensions



G



R = lattice cell

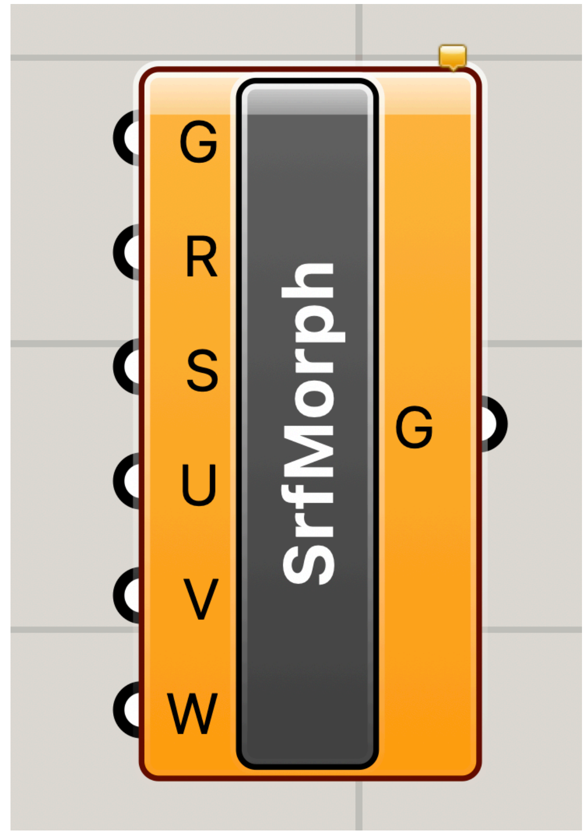
U,V = 1/N   N = size of array
one tile takes up 1/N
of the surface area

W = thickness of output

Surface Morph: warnings

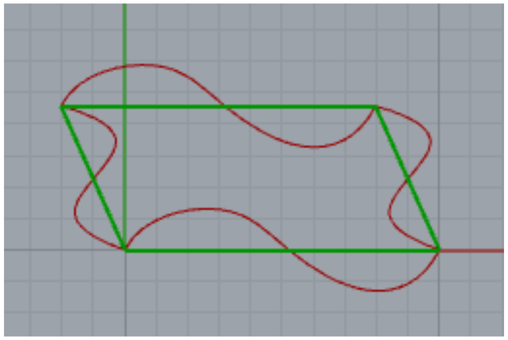
- Very **computationally expensive** and slow. More expensive for complex input geometries (tilings) and complex surfaces. Expect some spinning balls.

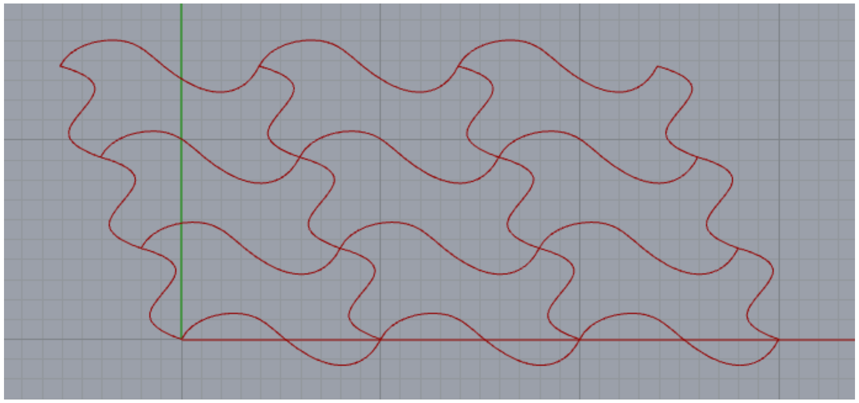
- **Size parameters** are critical. If you get any of them wrong (R, U, or V) you will crash Rhino and/or your results will be bizarre.



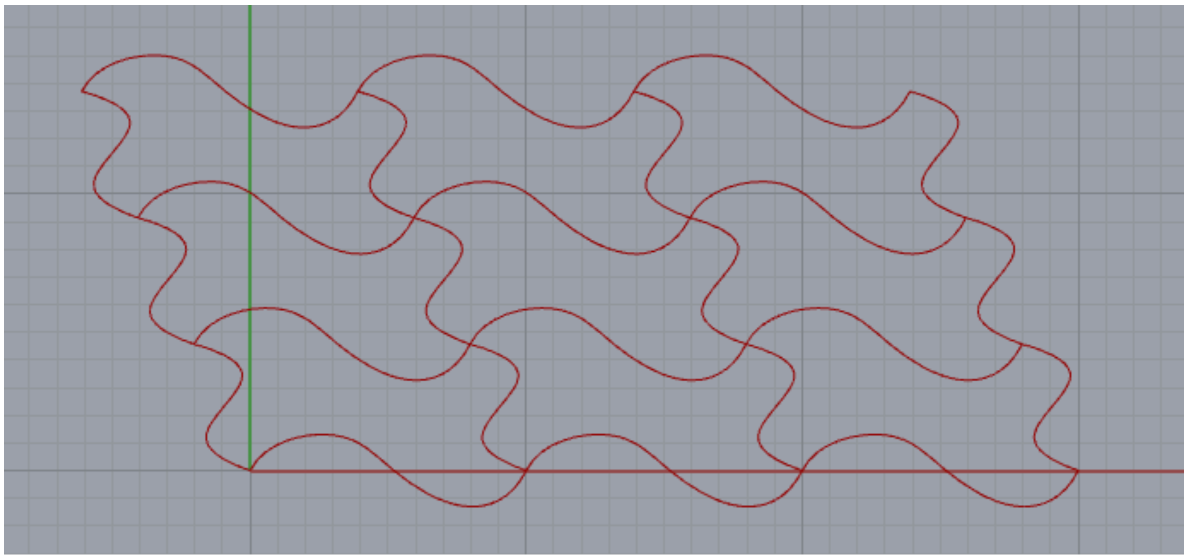
Some Observations About Sizes

Tiling is generated across the **lattice**.
Lattice cell determines tile translations.
Lattice cell edges: **a** and **b**
We need to use lattice dimensions
for our reference sizes.

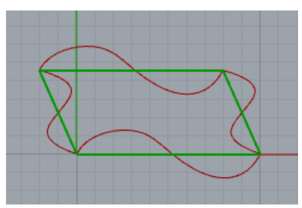




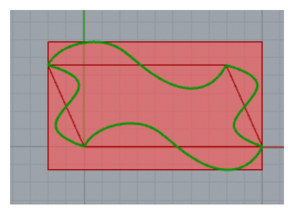
Some Observations About Sizes



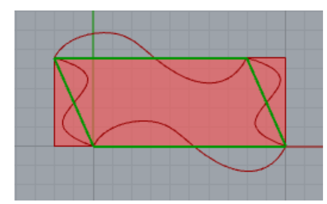
tile



+ lattice cell



+ tile
bounding box

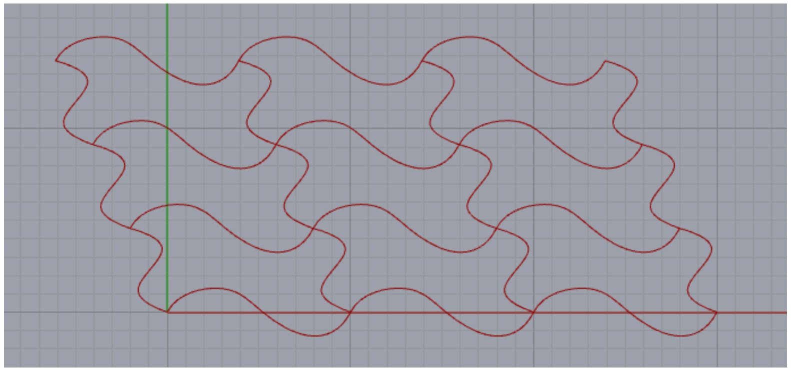


+ lattice cell
bounding box

Some Observations About Sizes

- We will use the entire tiling as G input, a single lattice cell as R (reference size) and 1/size as both U and V.

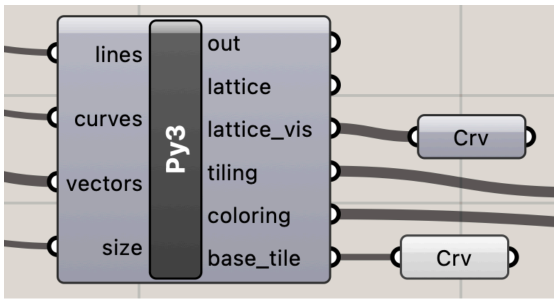
- Alternatively, we could use the entire tiling as G input and the size of the entire lattice (not the size of the tiling) as R and 1 as U and V



- Key insight: R, U and V have to align and are based on the size of the lattice, not the tiling

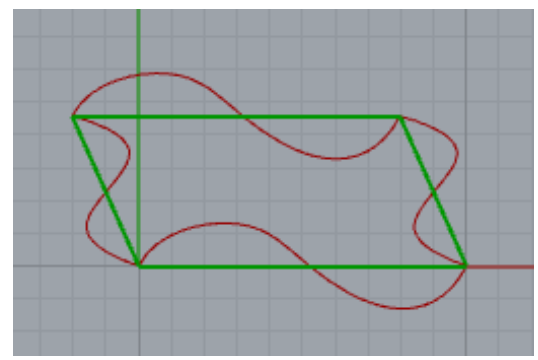
questions?

Create a Base Tile for Size Reference

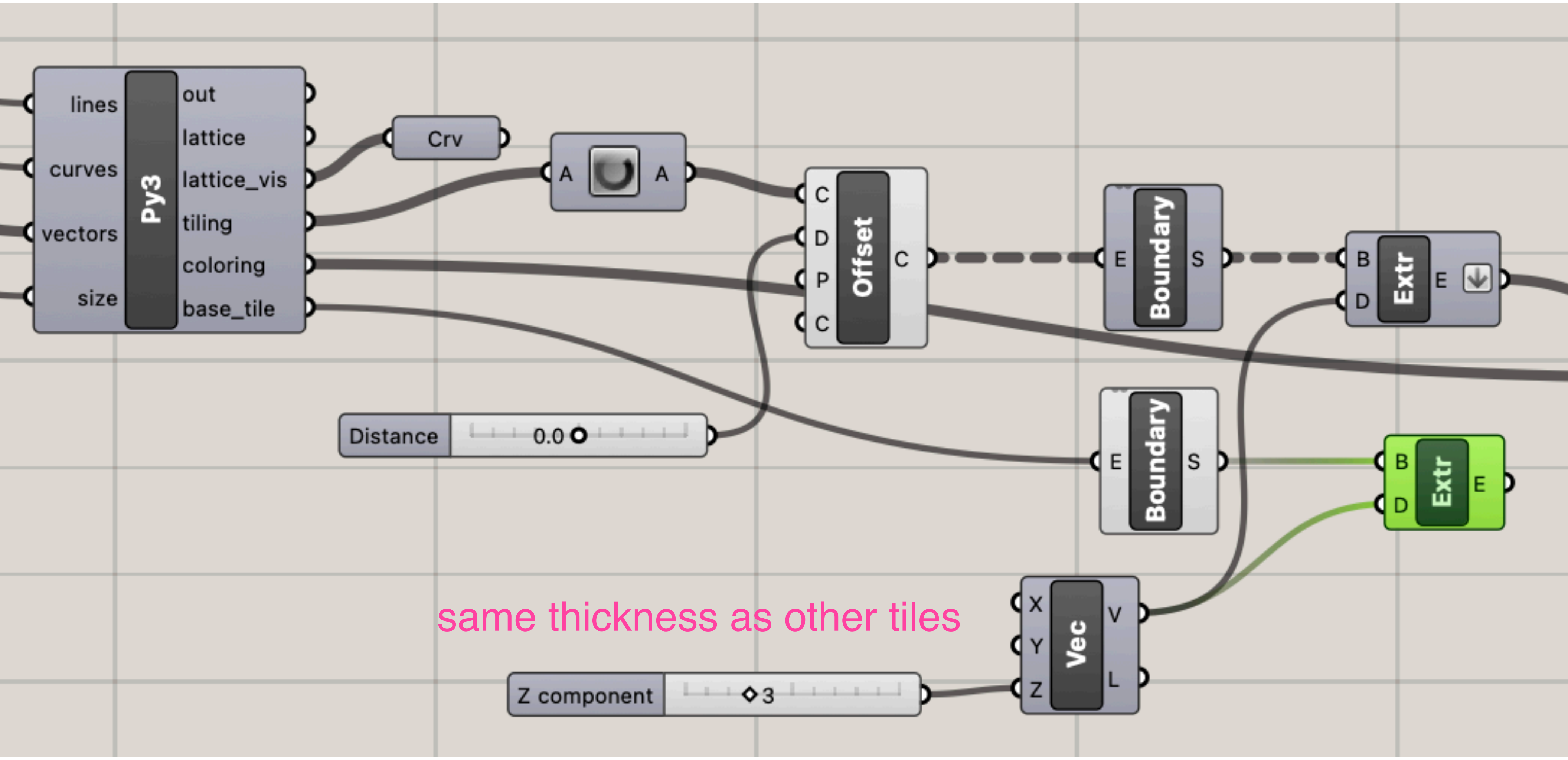


1. Add a lines input to your 2nd python block
2. Connect the lines input from the 1st python block
3. Edit code to create a base tile
4. Add a base_tile output

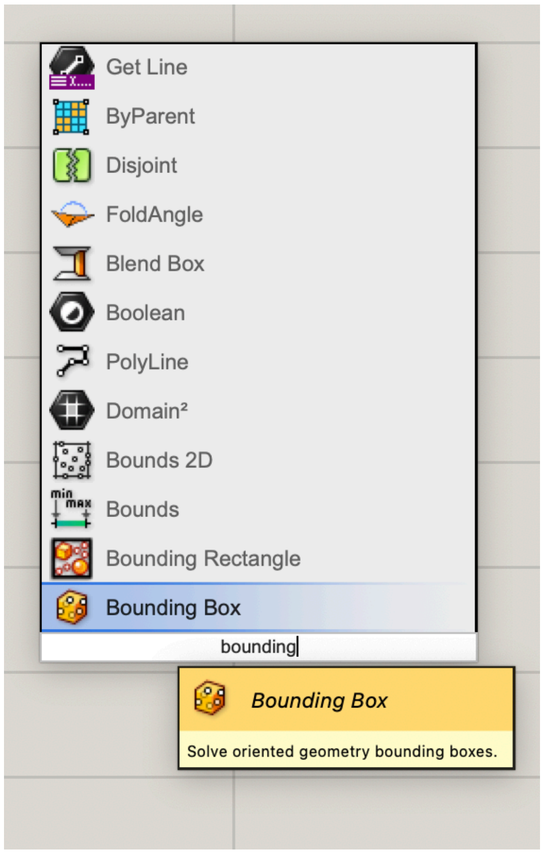
```
#generate base tile
bottom_left = lines
top = rs.ExplodeCurves(lines)[0]
rs.MoveObject(top,vectors[0])
right = rs.ExplodeCurves(lines)[1]
rs.MoveObject(right,vectors[1])
base_tile = rs.JoinCurves([bottom_left,top,right])
base_tile = rs.CloseCurve(base_tile)
```

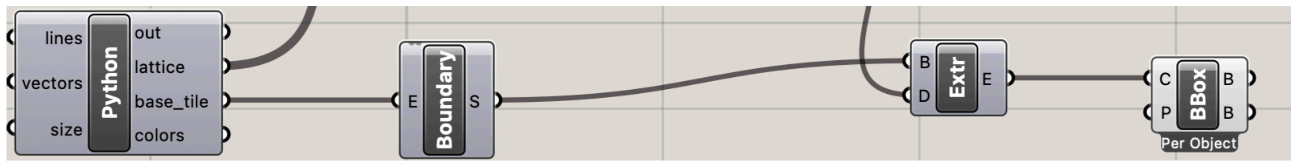


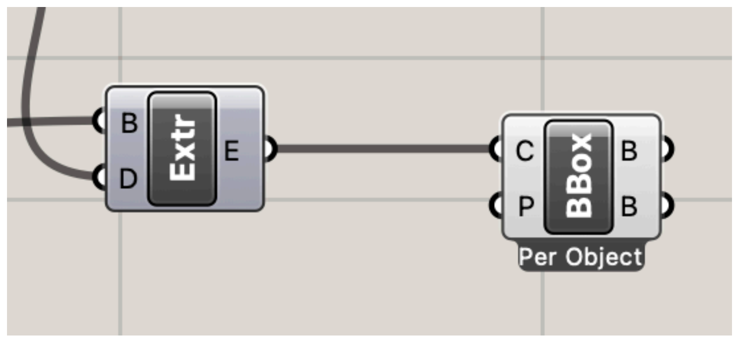
Make 3D Base Tile

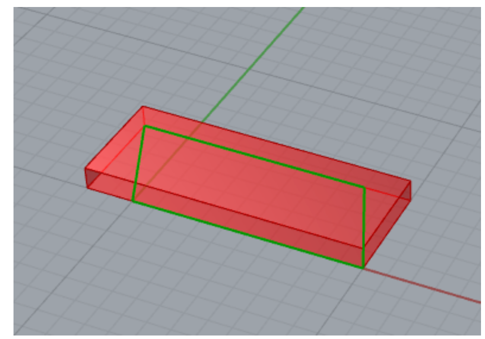


Get Size of 3D Base Tile



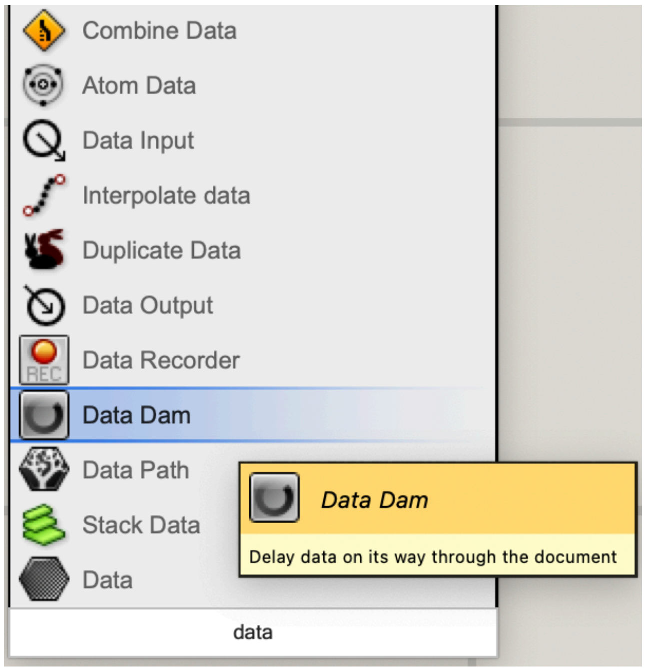




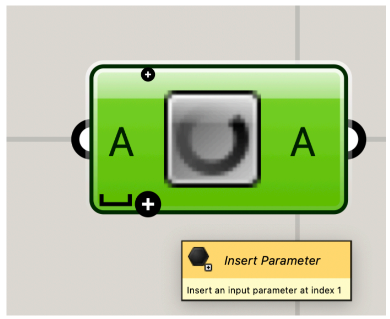


Connect Bounding Box to 3D Base Tile

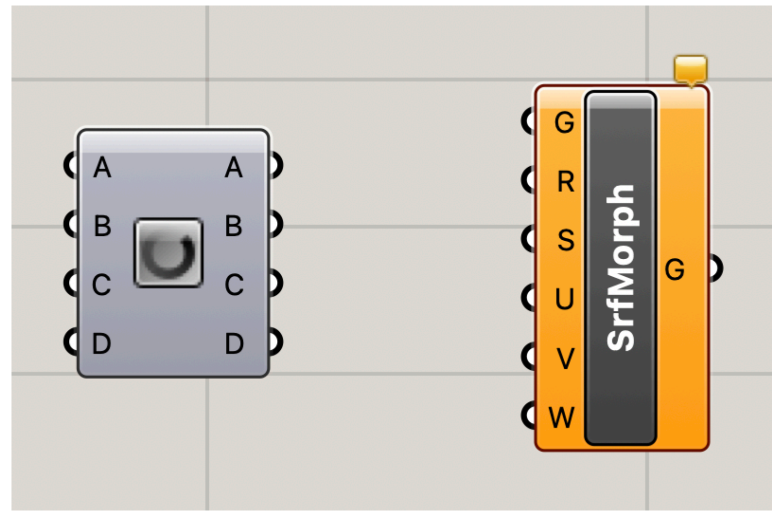
Create a data dam to avoid triggering the computationally expensive Surface Morph



Create a Data Dam

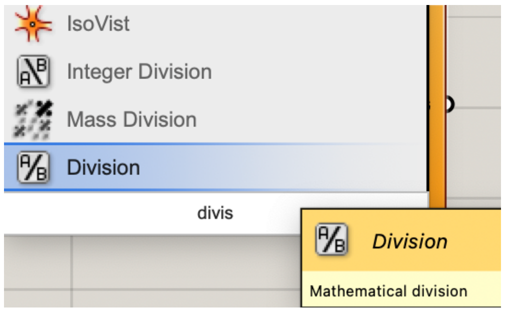


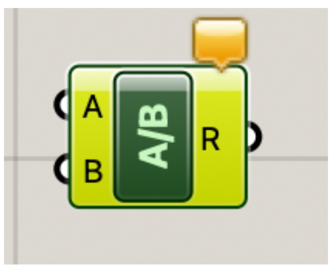
Zoom into the block to add parameters, just like we do with Python blocks



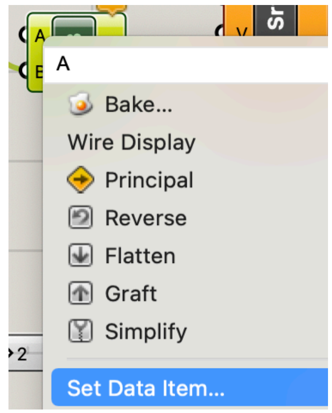
Add 5 parameters to the Data Dam block

Generate U,V information for Surface Morph



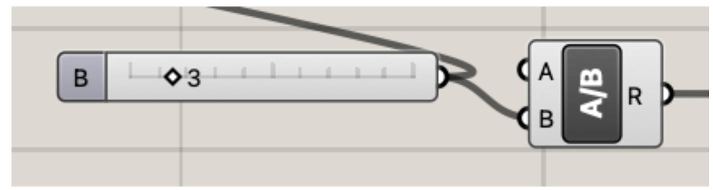


Create a Division block



Right click on the A parameter on the Division block

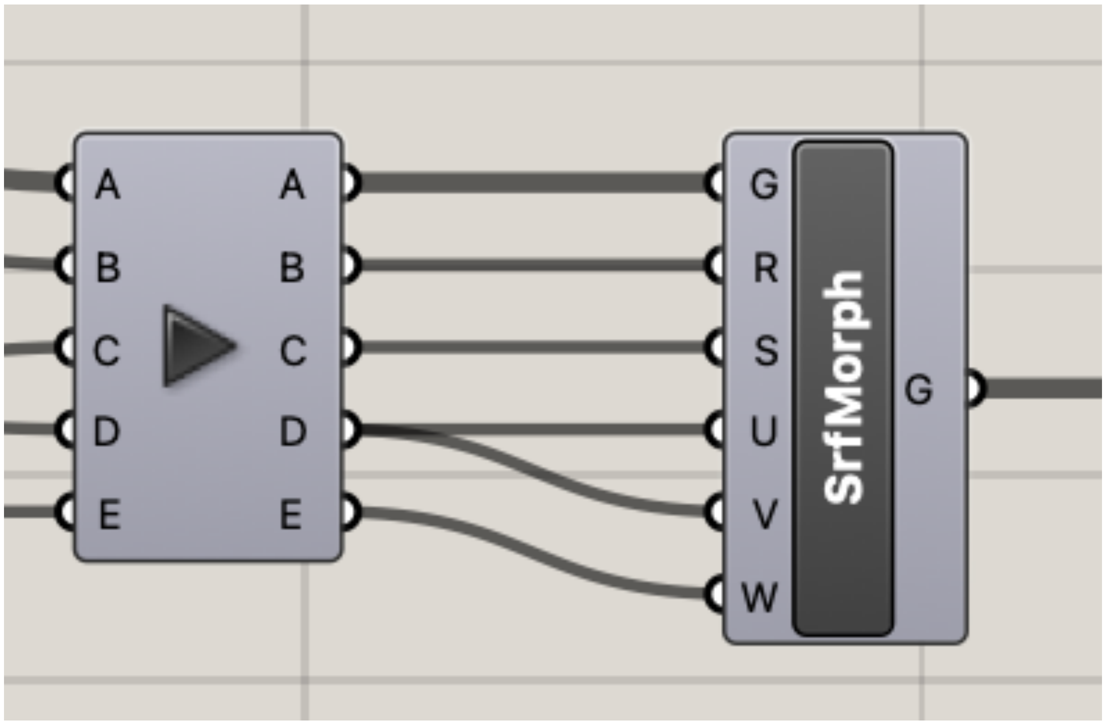
Enter 1 as a Data Item



Connect the size slider to B on the Division Block

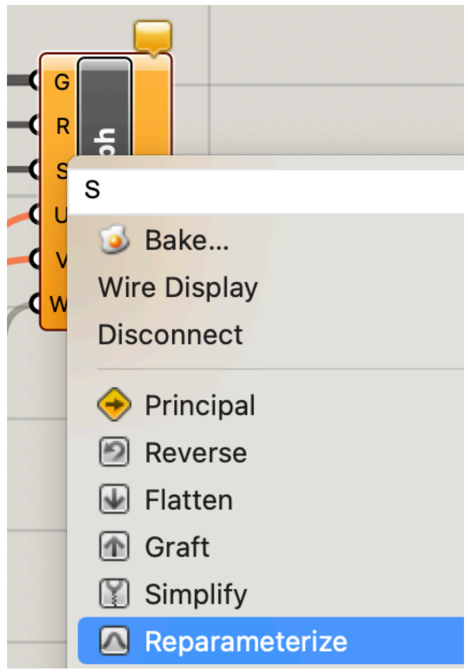
R = 1/number of tiles

Connect inputs through the Data Dam



- tiling goes to G
- Bounding Box goes to R
- Surface goes to S
- 1/size goes to U and V
- number slider goes to W

Reparameterize Surface Morph



Right click on the S input to the Surface Morph block and click Reparameterize.

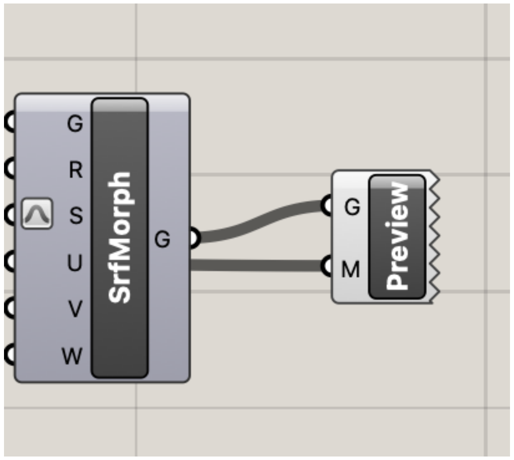
This will tell the block that U, V, and Z are percentages instead of absolute values.

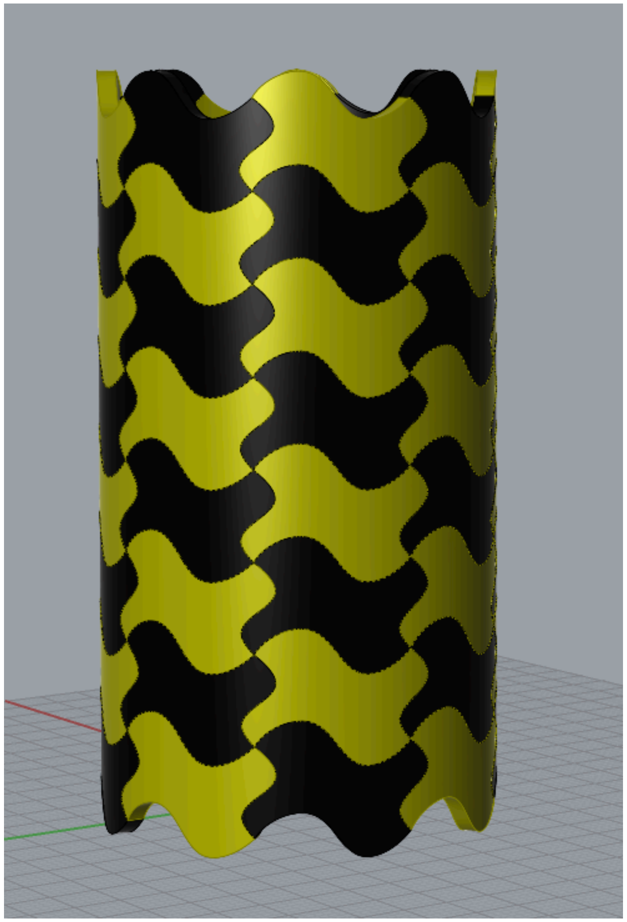
Save your Grasshopper file
in case the next step crashes Rhino.

Click the Play button on your Data Dam to trigger Surface Morph

- Note that the output is a size x size tiling morphed across the input surface (a cylinder)
- Verify that tiles connect smoothly all the way around form
- If you get parameters wrong, Rhino is likely to crash :'(

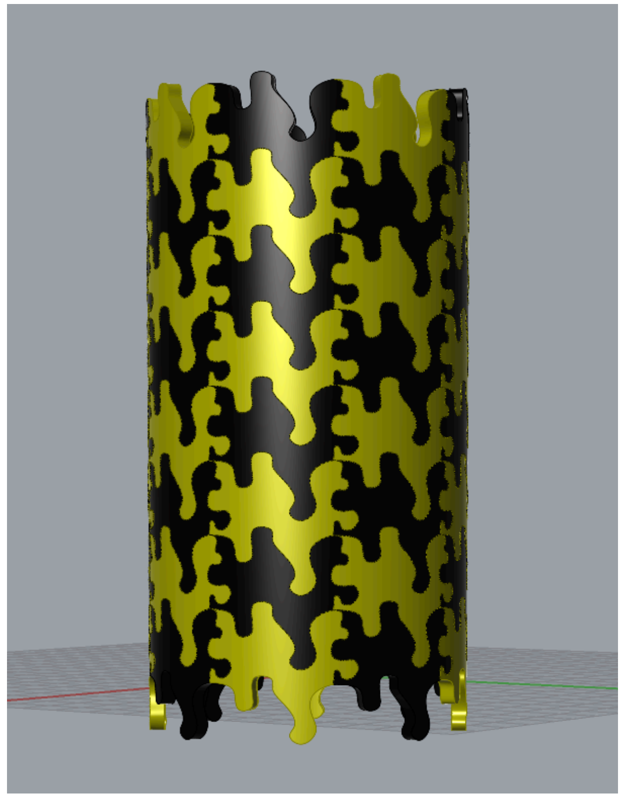
Add Some Color





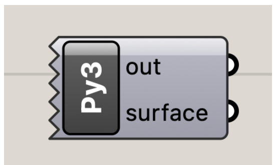
- M is color list we generated for tiling
- Note that the output is a size x size tiling morphed across the input surface (a cylinder)
- Verify that tiles connect all the way around form. Note: may be a slight seem for very complex tiles

More complex tiling

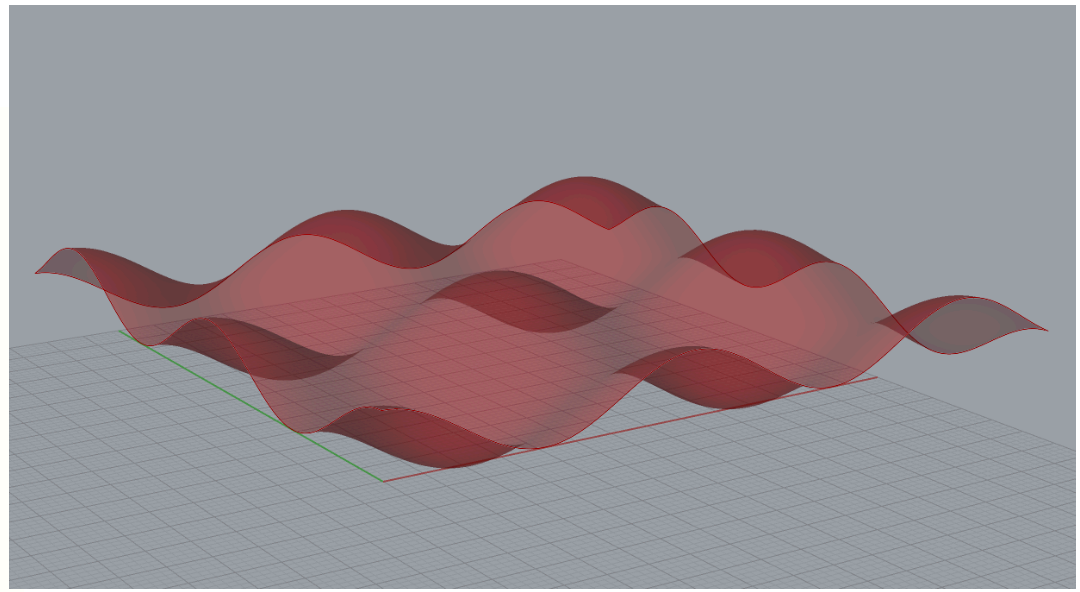


Morph onto a more interesting surface

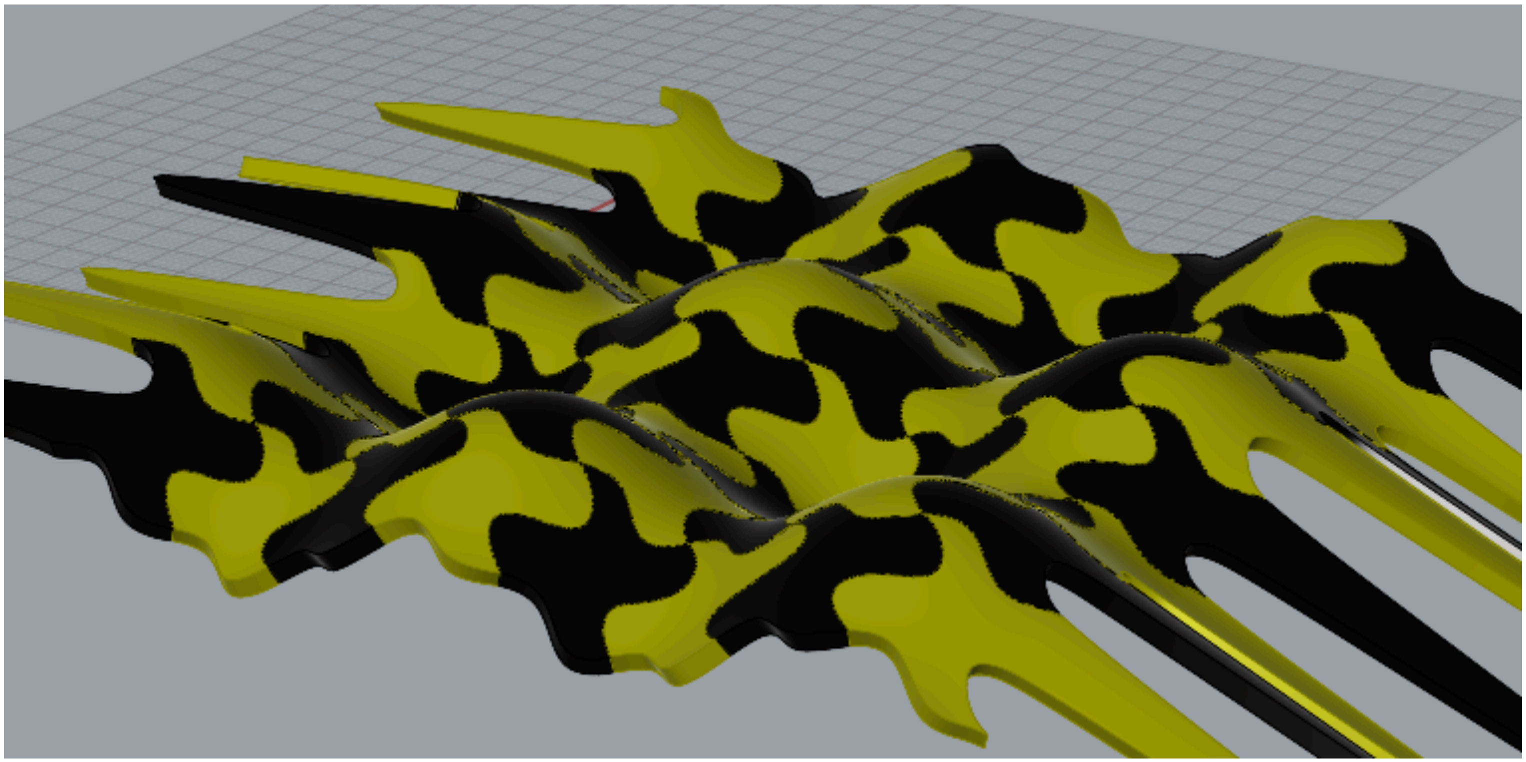
Create a 2D surface




```
import rhinoscriptsyntax as rs
import math

lines = []
for i in range (0,100):
    points = []
    for j in range (0,100):
        z = math.cos(i*.2)+ math.cos(j*.2)
        points.append(rs.CreatePoint(i,j,3*z))
    line = rs.AddCurve(points)
    lines.append(line)


surface = rs.AddLoftSrf(lines)
```

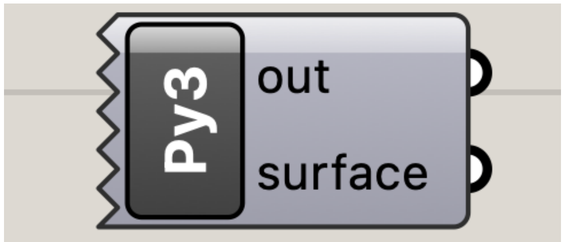


Morph onto this surface

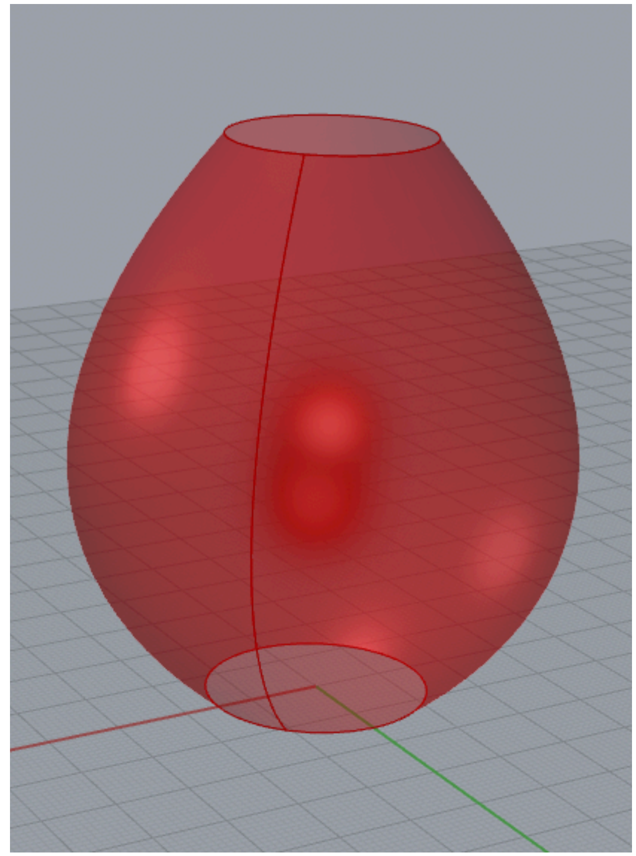


Edges get weird

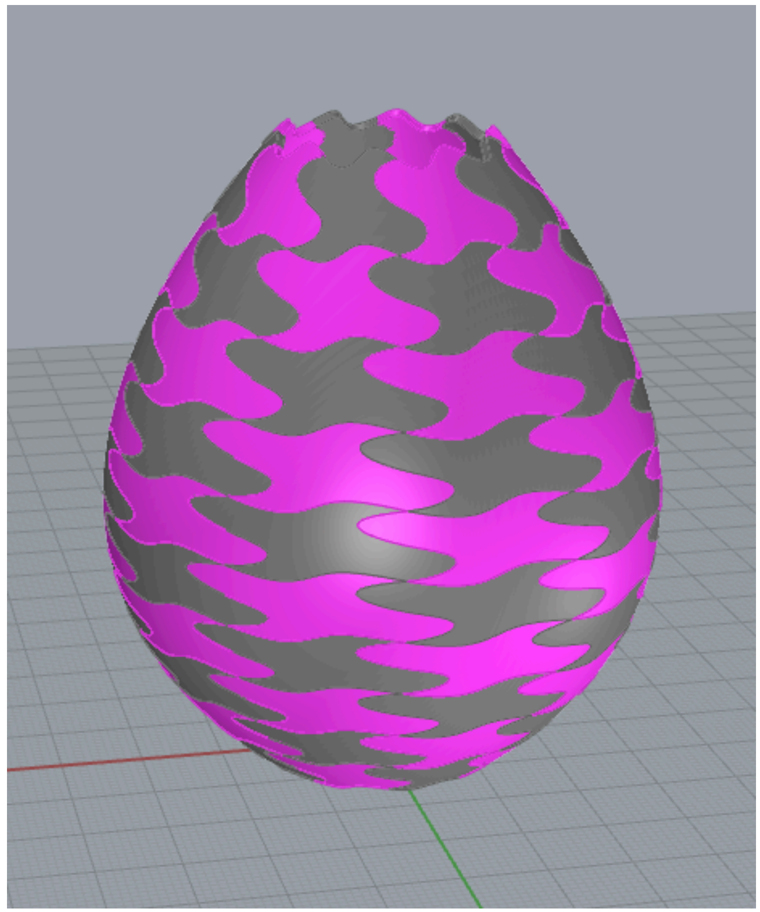
Create a simple vase surface




```python
import rhinoscriptsyntax as rs

r1 = 20
r2 = 50
r3 = 15

circle1 = rs.AddCircle([0,0,0],r1)
circle2 = rs.AddCircle([0,0,50],r2)
circle3 = rs.AddCircle([0,0,100],r3)

surface = rs.AddLoftSrf([circle1, circle2, circle3])
```

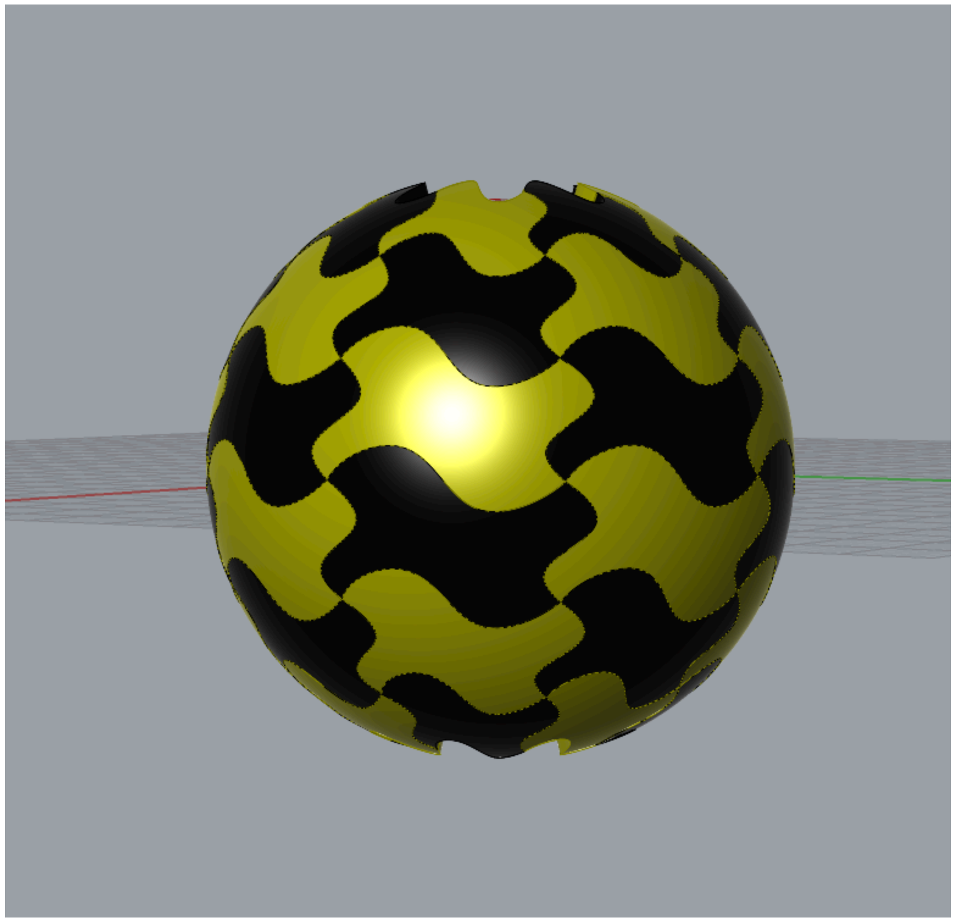


Morph onto this surface



Complex tile shape may create a slight seam

Morph onto a sphere...



Can morph tiles onto any surface
you’ve made so far.
ie: vases, topo maps, etc.

questions?

Thank you!

CS 491 and 591
Professor: Leah Buechley
https://handandmachine.cs.unm.edu/classes/Computational_Fabrication_Spring2021/