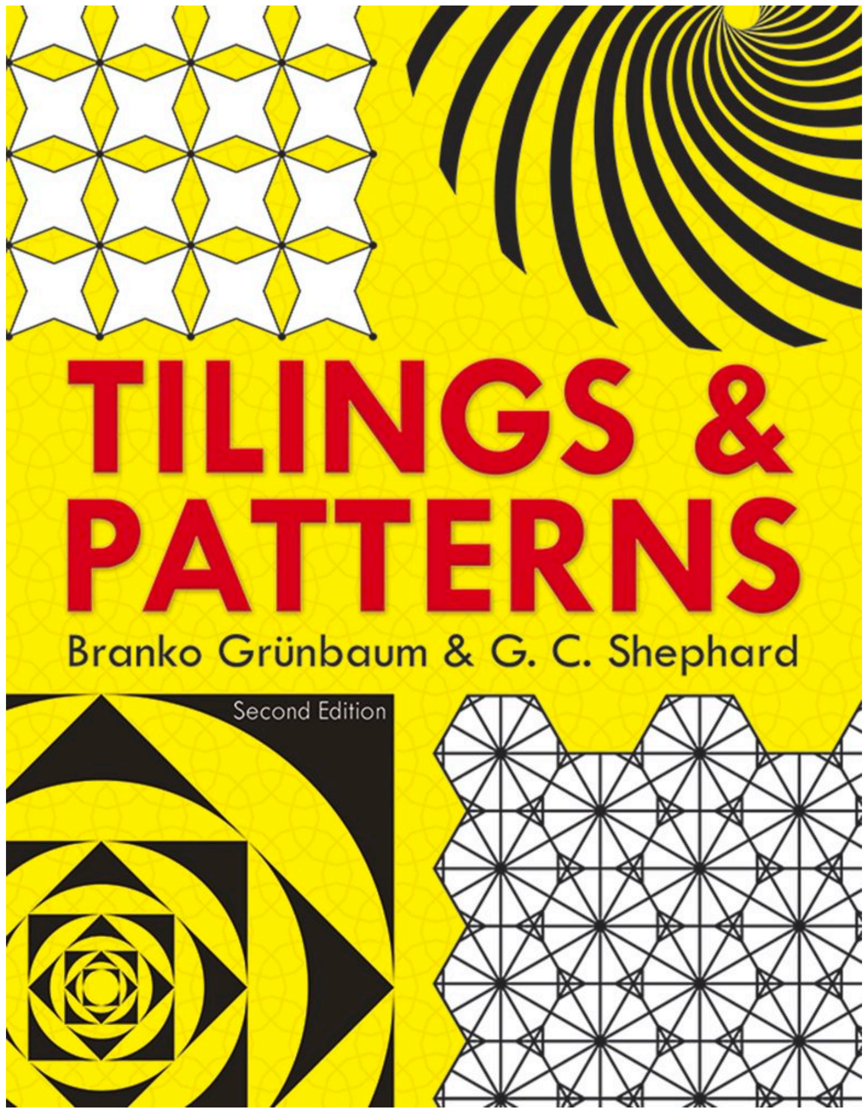
Computational Fabrication

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

Last Class: Categorizations of Tiles



Today: (Periodic) Tile Generation

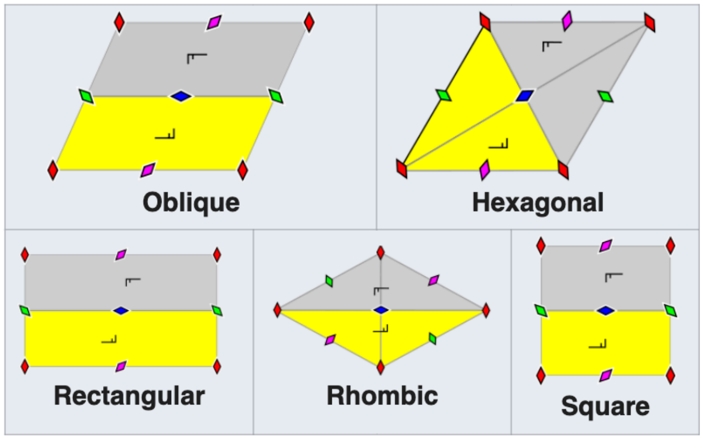
Periodic Tilings and Wallpaper Groups

- Any periodic tiling can be characterized as a "wallpaper".
- Wallpaper Groups: formal categories that describe the types of symmetries present in a tiling
- Describing symmetry = describing transformations (translation, rotation, reflection). Useful information for constructing tilings.

https://en.wikipedia.org/wiki/Wallpaper_group

Group *p*2 (2222) [edit]

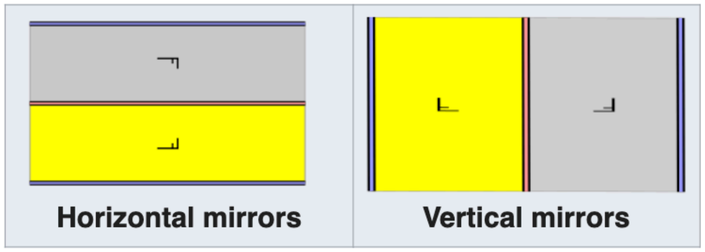
Cell structures for *p*2 by lattice type



- Orbifold signature: 2222
- Coxeter notation (rectangular): [∞,2,∞]$^+$
- Lattice: oblique
- Point group: C_2
- The group ***p*2** contains four rotation centres of order two (180°), but no reflections or glide reflections.

Group *pm* (**) [edit]

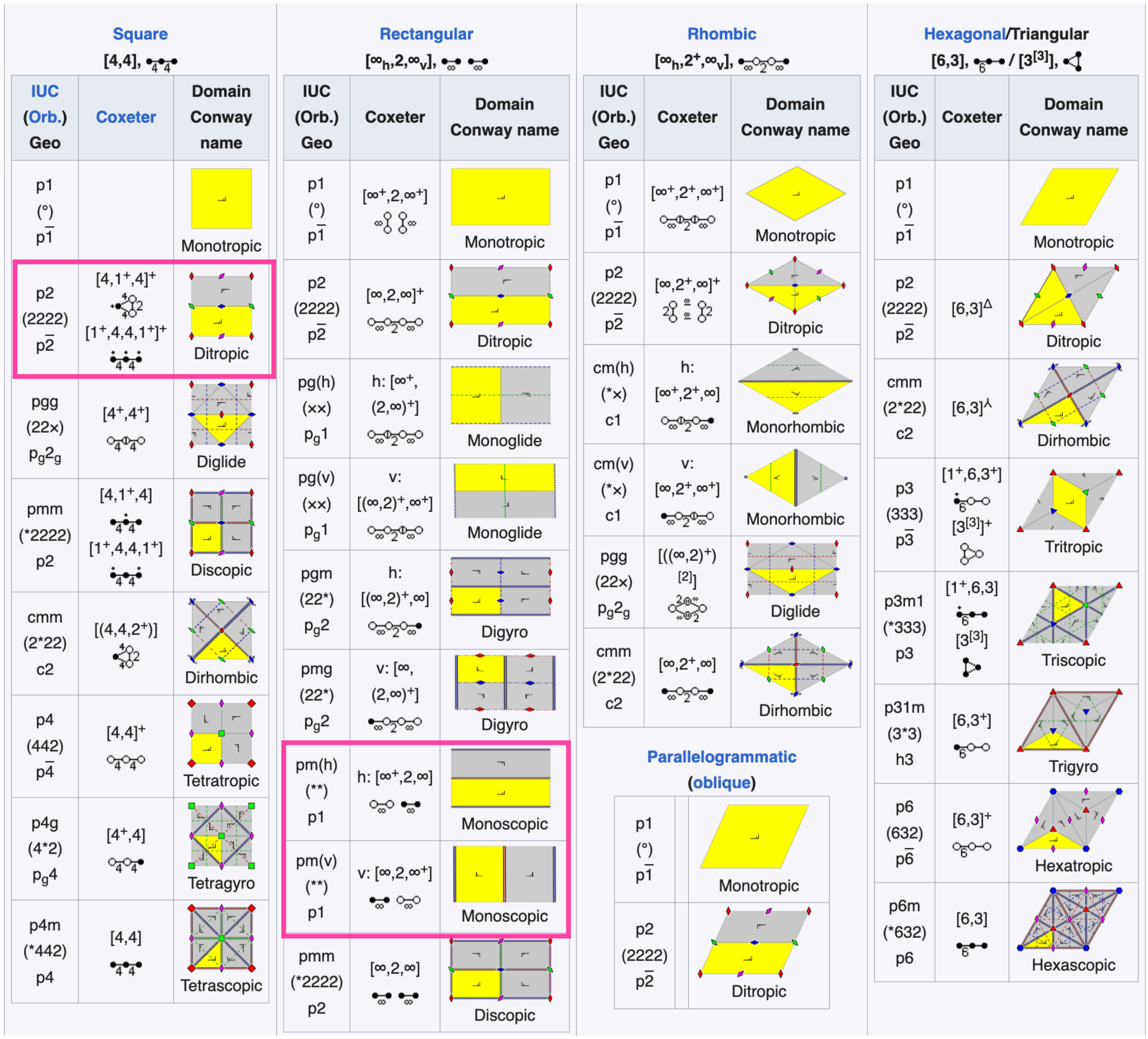
Cell structure for *pm*



- Orbifold signature: **
- Coxeter notation: [∞,2,∞$^+$] or [∞$^+$,2,∞]
- Lattice: rectangular
- Point group: D_1
- The group ***pm*** has no rotations. It has reflection axes, they are all parallel.

https://en.wikipedia.org/wiki/Wallpaper_group

17 Wallpaper Groups (2D)



Square
$[4,4]$

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$		Monotropic
p2 (2222) $p\bar{2}$	$[4,1^+,4]^+$ $[1^+,4,4,1^+]^+$	Ditropic
pgg (22×) p_g2_g	$[4^+,4^+]$	Diglide
pmm (*2222) p2	$[4,1^+,4]$ $[1^+,4,4,1^+]$	Discopic
cmm (2*22) c2	$[(4,4,2^+)]$	Dirhombic
p4 (442) $p\bar{4}$	$[4,4]^+$	Tetratropic
p4g (4*2) p_g4	$[4^+,4]$	Tetragyro
p4m (*442) p4	$[4,4]$	Tetrascopic

Rectangular
$[\infty_h,2,\infty_v]$

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$	$[\infty^+,2,\infty^+]$	Monotropic
p2 (2222) $p\bar{2}$	$[\infty,2,\infty]^+$	Ditropic
pg(h) (××) p_g1	h: $[\infty^+,(2,\infty)^+]$	Monoglide
pg(v) (××) p_g1	v: $[(\infty,2)^+,\infty^+]$	Monoglide
pgm (22*) p_g2	h: $[(\infty,2)^+,\infty]$	Digyro
pmg (22*) p_g2	v: $[\infty,(2,\infty)^+]$	Digyro
pm(h) (**) p1	h: $[\infty^+,2,\infty]$	Monoscopic
pm(v) (**) p1	v: $[\infty,2,\infty^+]$	Monoscopic
pmm (*2222) p2	$[\infty,2,\infty]$	Discopic

Rhombic
$[\infty_h,2^+,\infty_v]$

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$	$[\infty^+,2^+,\infty^+]$	Monotropic
p2 (2222) $p\bar{2}$	$[\infty,2^+,\infty]^+$	Ditropic
cm(h) (*×) c1	h: $[\infty^+,2^+,\infty]$	Monorhombic
cm(v) (*×) c1	v: $[\infty,2^+,\infty^+]$	Monorhombic
pgg (22×) p_g2_g	$[((\infty,2)^+)^{[2]}]$	Diglide
cmm (2*22) c2	$[\infty,2^+,\infty]$	Dirhombic

Parallelogrammatic (oblique)

p1 (°) $p\bar{1}$		Monotropic
p2 (2222) $p\bar{2}$		Ditropic

Hexagonal/Triangular
$[6,3]$, / $[3^{[3]}]$,

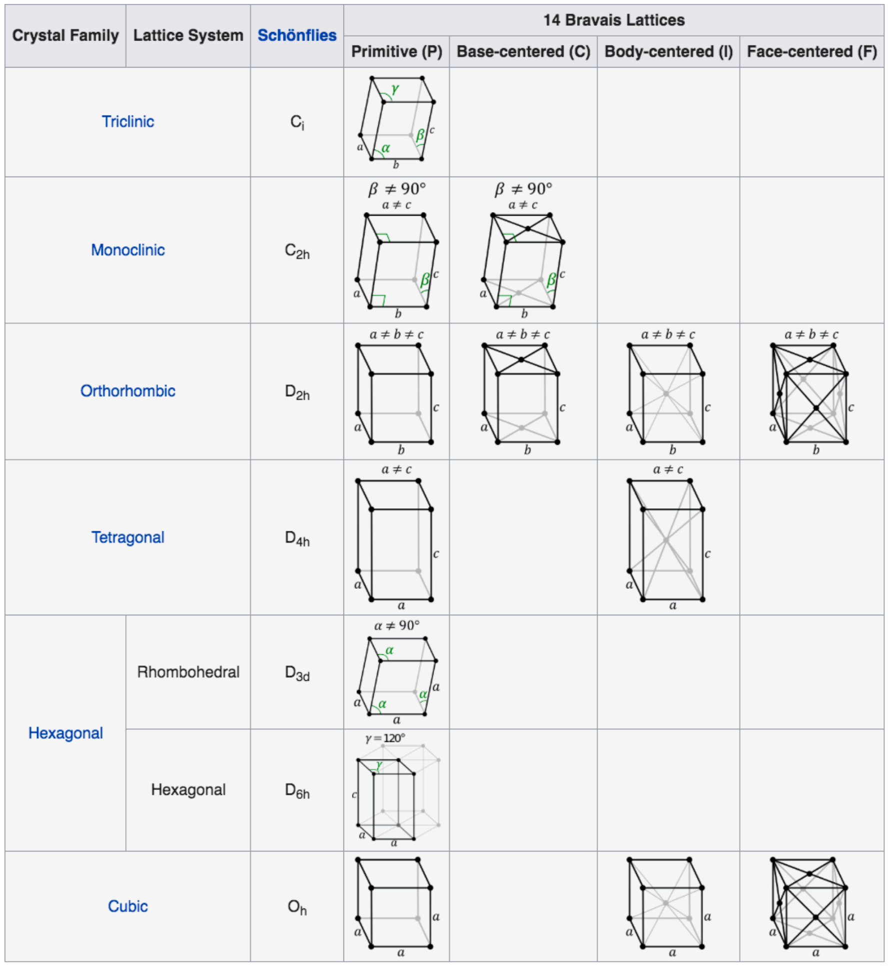
IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$		Monotropic
p2 (2222) $p\bar{2}$	$[6,3]^\Delta$	Ditropic
cmm (2*22) c2	$[6,3]^\curlywedge$	Dirhombic
p3 (333) $p\bar{3}$	$[1^+,6,3^+]$ $[3^{[3]}]^+$	Tritropic
p3m1 (*333) p3	$[1^+,6,3]$ $[3^{[3]}]$	Triscopic
p31m (3*3) h3	$[6,3^+]$	Trigyro
p6 (632) $p\bar{6}$	$[6,3]^+$	Hexatropic
p6m (*632) p6	$[6,3]$	Hexascopic

https://en.wikipedia.org/wiki/Wallpaper_group

Bravais Lattices

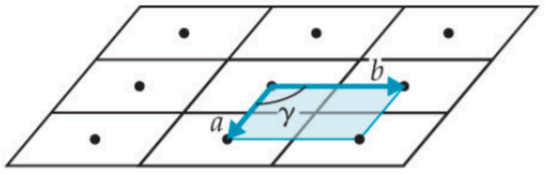
- Mathematical definition: an infinite arrangement of points in space such that the lattice looks exactly the same when viewed from any lattice point.
- In 3D, Bravais Lattices define the 14 different configurations into which atoms can be arranged in crystals.

14 3D Bravais Lattice Structures

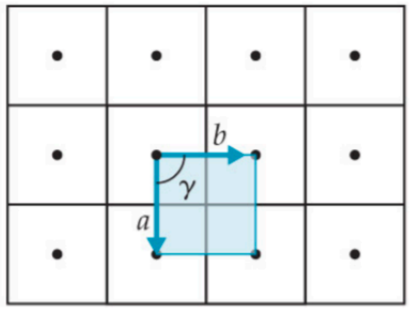


Crystal Family	Lattice System	Schönflies	14 Bravais Lattices			
			Primitive (P)	Base-centered (C)	Body-centered (I)	Face-centered (F)
Triclinic		C_i				
Monoclinic		C_{2h}	$\beta \neq 90°$ $a \neq c$	$\beta \neq 90°$ $a \neq c$		
Orthorhombic		D_{2h}	$a \neq b \neq c$	$a \neq b \neq c$	$a \neq b \neq c$	$a \neq b \neq c$
Tetragonal		D_{4h}	$a \neq c$		$a \neq c$	
Hexagonal	Rhombohedral	D_{3d}	$\alpha \neq 90°$			
	Hexagonal	D_{6h}	$\gamma = 120°$			
Cubic		O_h				

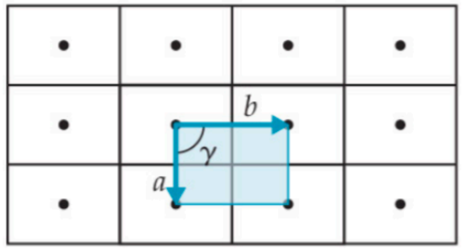
5 2D Bravais Lattice Structures



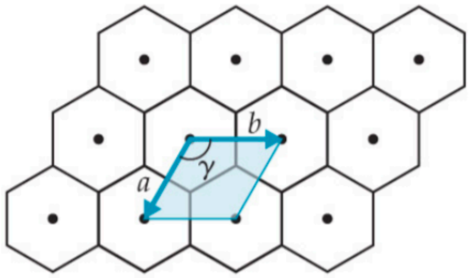
Oblique lattice ($a \neq b$, γ = arbitrary)



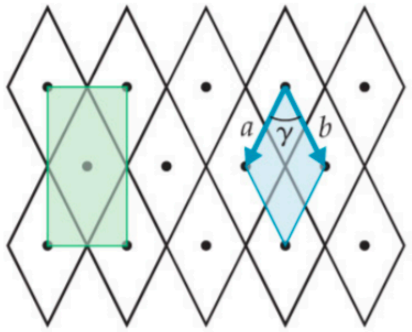
Square lattice ($a = b$, $\gamma = 90°$)



Rectangular lattice ($a \neq b$, $\gamma = 90°$)



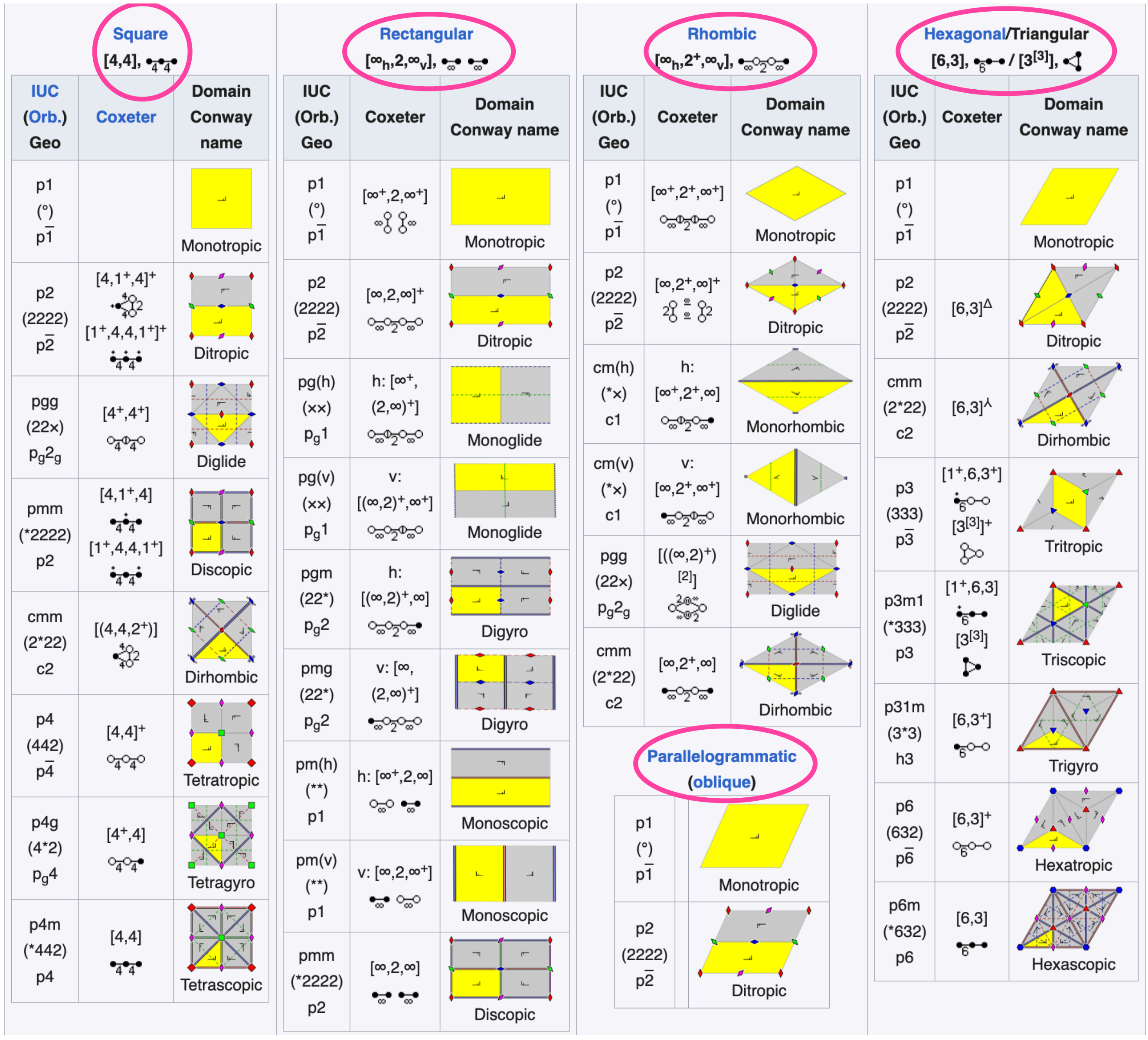
Hexagonal lattice ($a = b$, $\gamma = 120°$)



Rhombic lattice ($a = b$, γ = arbitrary)
Centered rectangular lattice

17 Wallpaper Groups (2D)

https://en.wikipedia.org/wiki/Wallpaper_group



Square
[4,4]

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$		Monotropic
p2 (2222) $p\bar{2}$	$[4,1^+,4]^+$ $[1^+,4,4,1^+]^+$	Ditropic
pgg (22×) p_g2_g	$[4^+,4^+]$	Diglide
pmm (*2222) p2	$[4,1^+,4]$ $[1^+,4,4,1^+]$	Discopic
cmm (2*22) c2	$[(4,4,2^+)]$	Dirhombic
p4 (442) $p\bar{4}$	$[4,4]^+$	Tetratropic
p4g (4*2) p_g4	$[4^+,4]$	Tetragyro
p4m (*442) p4	[4,4]	Tetrascopic

Rectangular
$[\infty_h,2,\infty_v]$,

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$	$[\infty^+,2,\infty^+]$	Monotropic
p2 (2222) $p\bar{2}$	$[\infty,2,\infty]^+$	Ditropic
pg(h) (××) p_g1	h: $[\infty^+,(2,\infty)^+]$	Monoglide
pg(v) (××) p_g1	v: $[(\infty,2)^+,\infty^+]$	Monoglide
pgm (22*) p_g2	h: $[(\infty,2)^+,\infty]$	Digyro
pmg (22*) p_g2	v: $[\infty,(2,\infty)^+]$	Digyro
pm(h) (**) p1	h: $[\infty^+,2,\infty]$	Monoscopic
pm(v) (**) p1	v: $[\infty,2,\infty^+]$	Monoscopic
pmm (*2222) p2	$[\infty,2,\infty]$	Discopic

Rhombic
$[\infty_h,2^+,\infty_v]$,

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$	$[\infty^+,2^+,\infty^+]$	Monotropic
p2 (2222) $p\bar{2}$	$[\infty,2^+,\infty]^+$	Ditropic
cm(h) (*×) c1	h: $[\infty^+,2^+,\infty]$	Monorhombic
cm(v) (*×) c1	v: $[\infty,2^+,\infty^+]$	Monorhombic
pgg (22×) p_g2_g	$[((\infty,2)^+[2])]$	Diglide
cmm (2*22) c2	$[\infty,2^+,\infty]$	Dirhombic

Parallelogrammatic (oblique)

p1 (°) $p\bar{1}$		Monotropic
p2 (2222) $p\bar{2}$		Ditropic

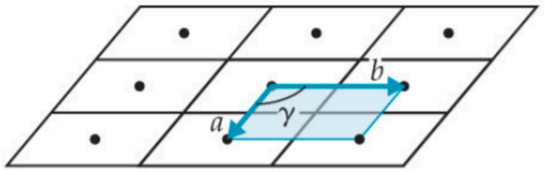
Hexagonal/Triangular
[6,3], / $[3^{[3]}]$,

IUC (Orb.) Geo	Coxeter	Domain Conway name
p1 (°) $p\bar{1}$		Monotropic
p2 (2222) $p\bar{2}$	$[6,3]^{\Delta}$	Ditropic
cmm (2*22) c2	$[6,3]^{\curlywedge}$	Dirhombic
p3 (333) $p\bar{3}$	$[1^+,6,3^+]$ $[3^{[3]}]^+$	Tritropic
p3m1 (*333) p3	$[1^+,6,3]$ $[3^{[3]}]$	Triscopic
p31m (3*3) h3	$[6,3^+]$	Trigyro
p6 (632) $p\bar{6}$	$[6,3]^+$	Hexatropic
p6m (*632) p6	[6,3]	Hexascopic

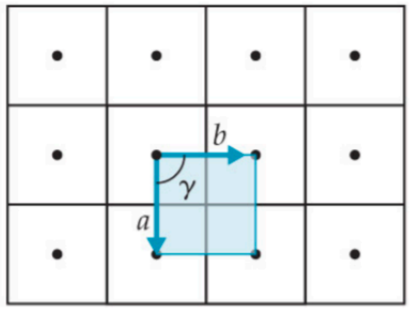
Bravais Lattice Structures

Any periodic 2D tiling maps to one of these 5 fundamental lattice structures.

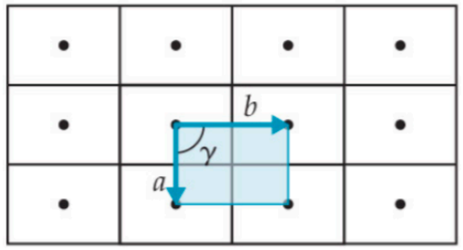
5 2D Bravais Lattice Structures



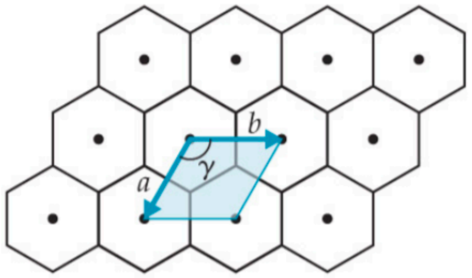
Oblique lattice ($a \neq b$, γ = arbitrary)



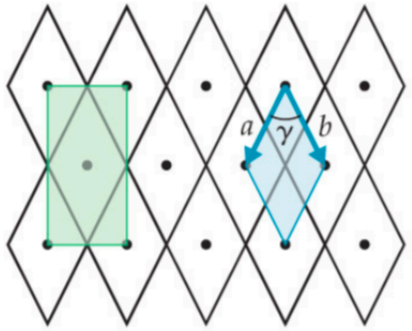
Square lattice ($a = b$, $\gamma = 90°$)



Rectangular lattice ($a \neq b$, $\gamma = 90°$)

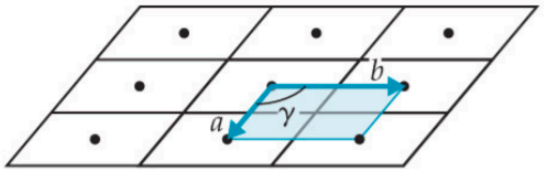


Hexagonal lattice ($a = b$, $\gamma = 120°$)

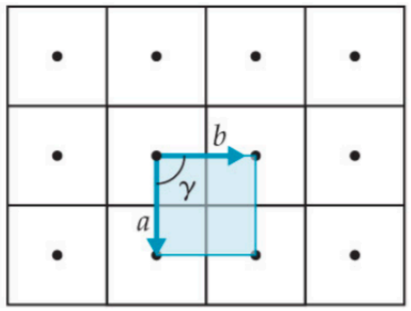


Rhombic lattice ($a = b$, γ = arbitrary)
Centered rectangular lattice

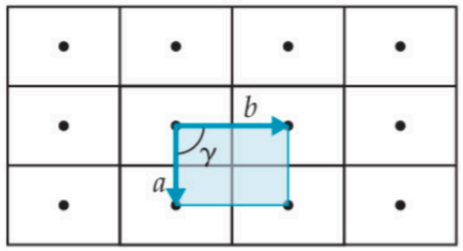
Note that they're all related



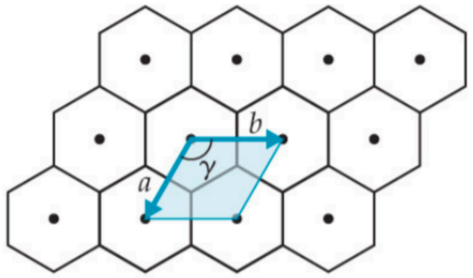
Oblique lattice ($a \neq b$, γ = arbitrary)



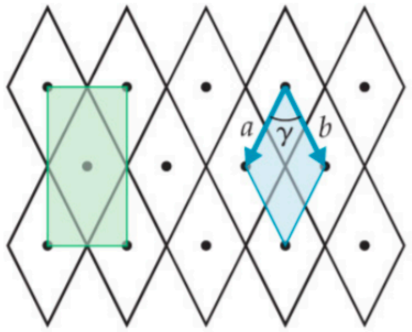
Square lattice ($a = b$, $\gamma = 90°$)



Rectangular lattice ($a \neq b$, $\gamma = 90°$)



Hexagonal lattice ($a = b$, $\gamma = 120°$)

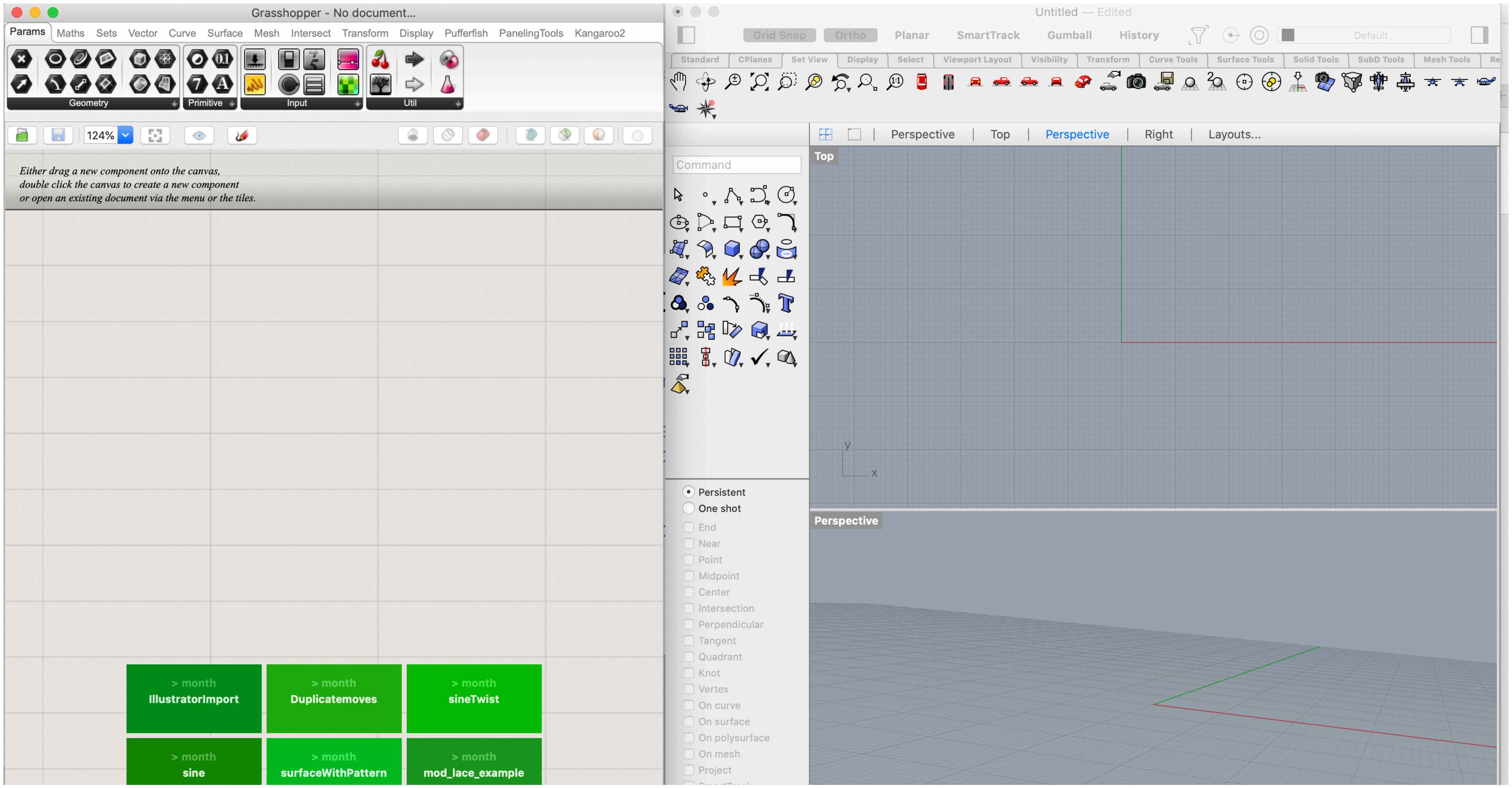


Rhombic lattice ($a = b$, γ = arbitrary)
Centered rectangular lattice

What we’ll do today

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

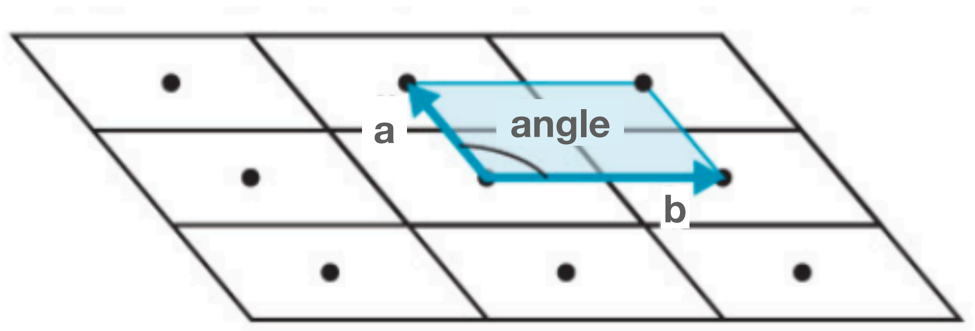
open up Rhino and Grasshopper



Either drag a new component onto the canvas, double click the canvas to create a new component or open an existing document via the menu or the tiles.

One lattice cell

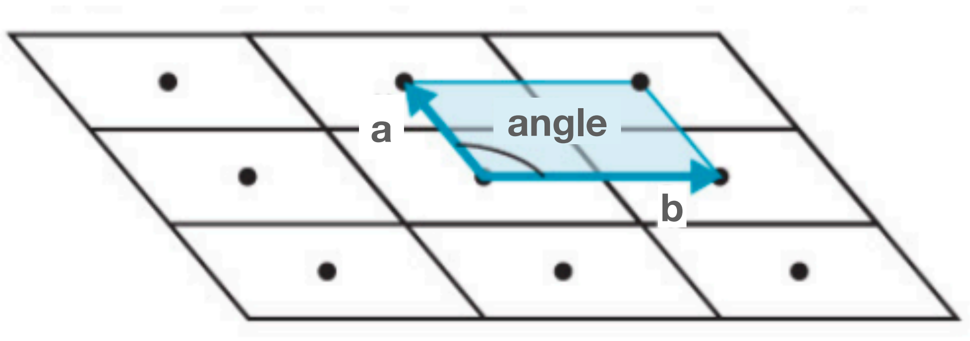
Parametric lattice: 3 simple variables



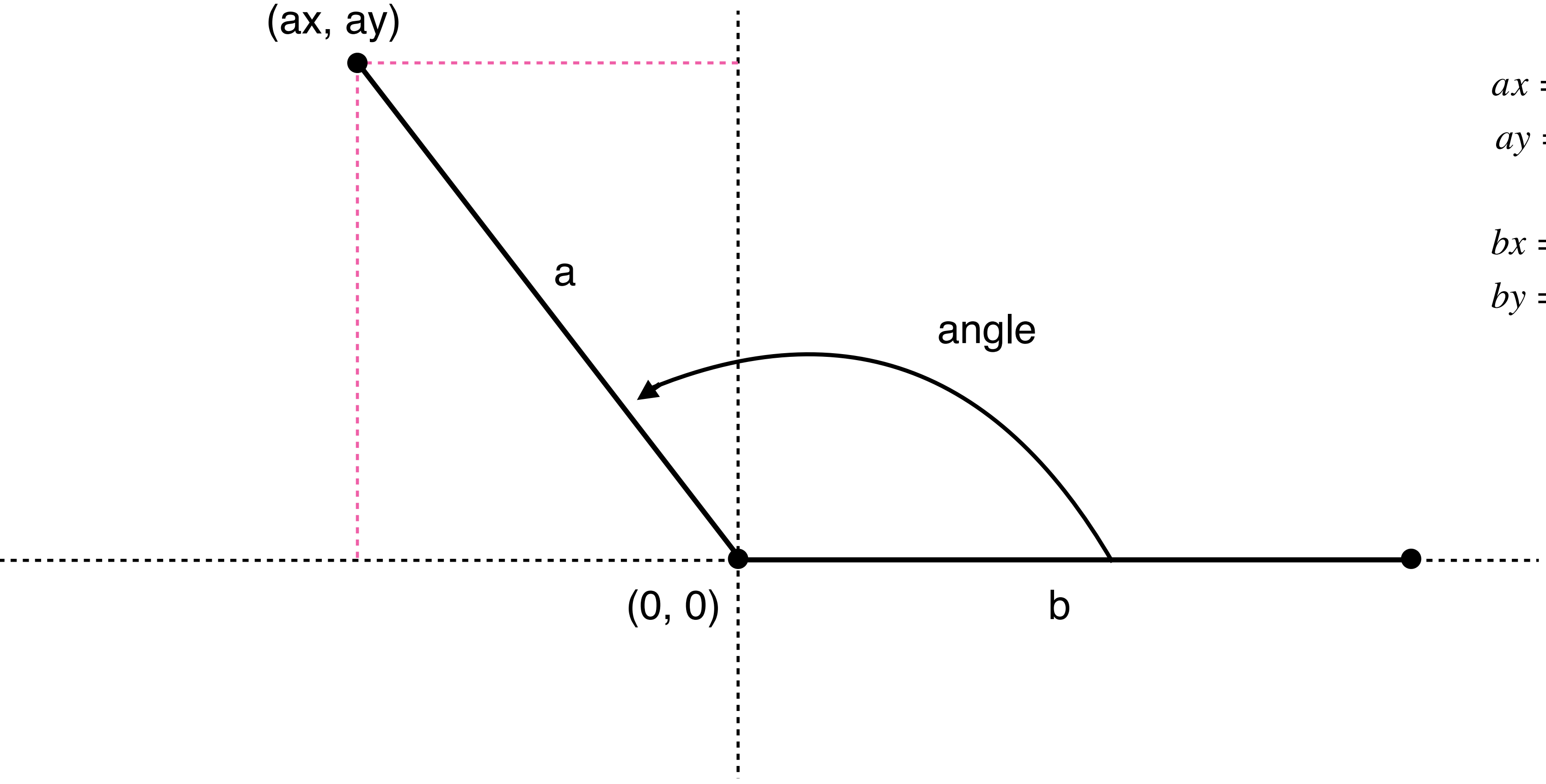
- a
- b
- γ (angle)

Grasshopper & Python

- Inputs:
  - a, b, angle
- Output:
  - lines for a and b
  - vectors for a and b



Lines and vectors? Simple math

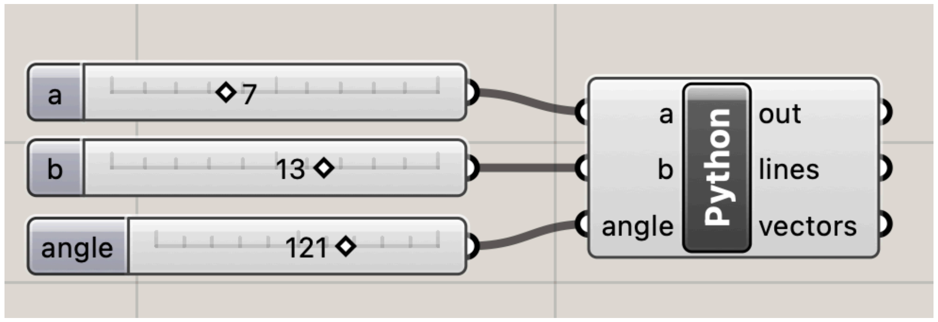


$$ax = a * cos(angle)$$
$$ay = a * sin(angle)$$

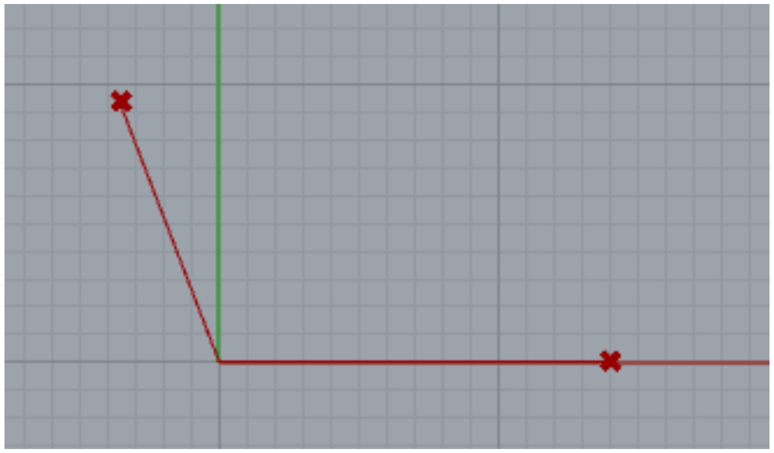
$$bx = b$$
$$by = 0$$

Grasshopper & Python Code

- Inputs:
  - a, b, angle
- Output:
  - lines for a and b
  - vectors for a and b



input: **Float** Type hints

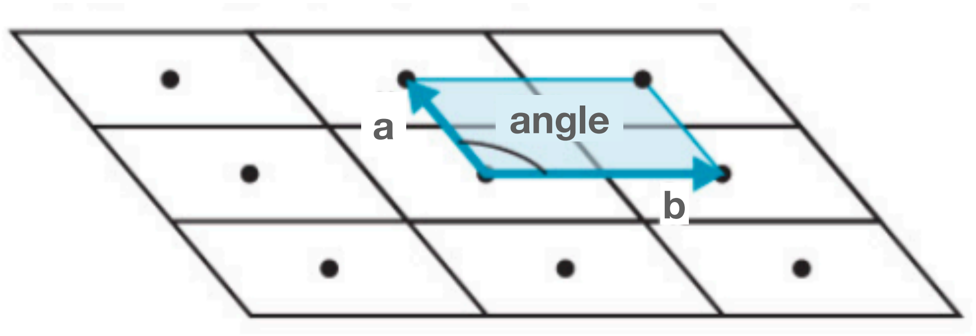


questions?

Generating the Lattice

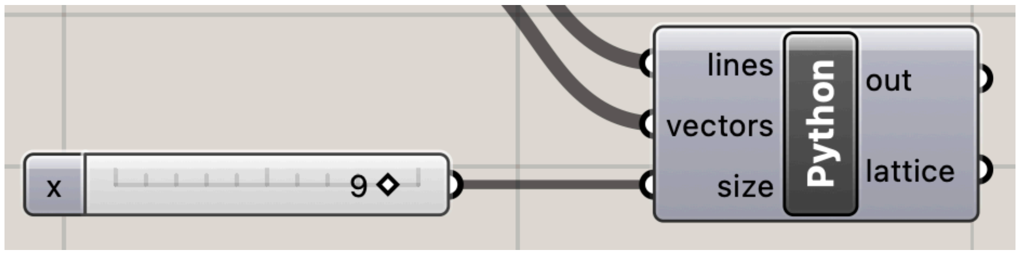
Copy and translate cell using vectors

- Inputs:
  - lines
  - vectors
  - size of lattice
- Output:
  - 2D lattice
    as list of tiles
    tile = closed curve

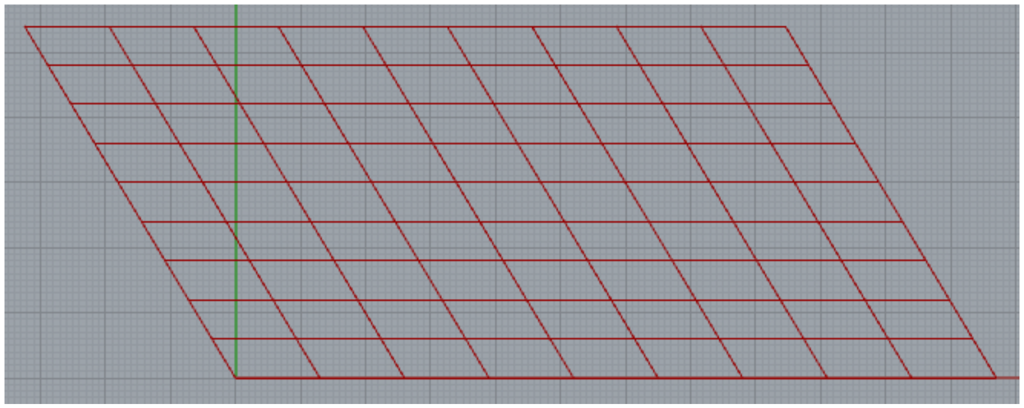


Grasshopper & Python

- Inputs:
  - lines
  - vectors
  - size of lattice
- Output:
  - 2D lattice
    as list of tiles
    tile = closed curve

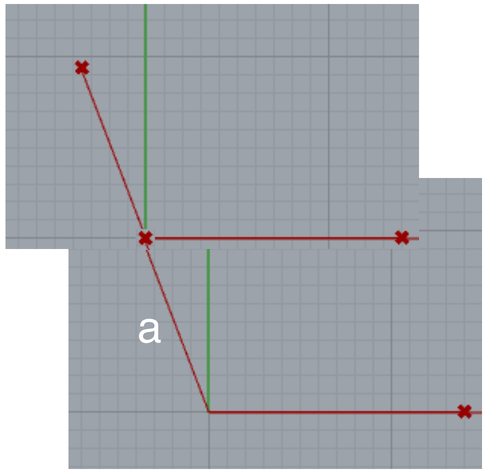


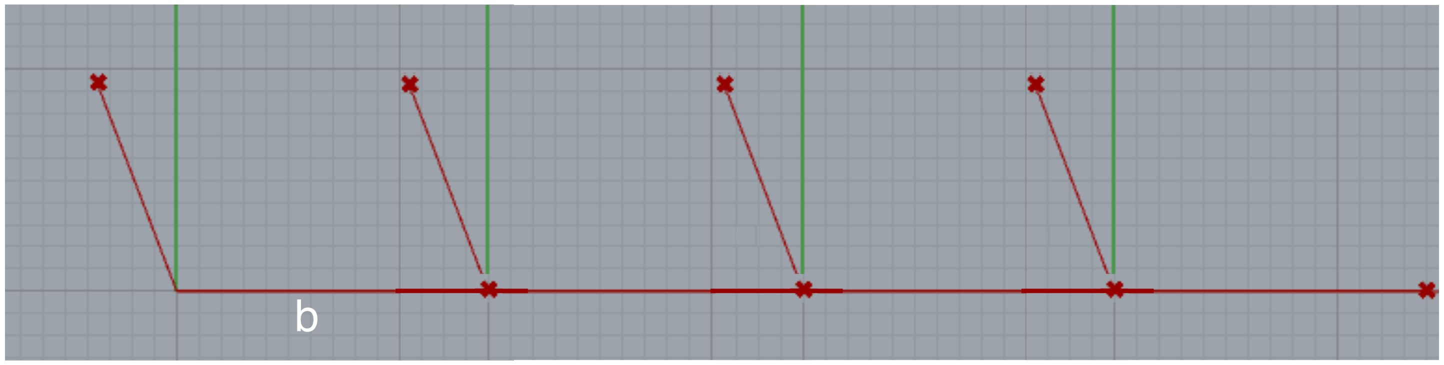
lines: **Curve** type hint
vectors: **Point** type hint
size: **int** type hing



Approach: 2D Lattice

- Copy input curves and translate along **a** and **b** vectors
- Use **rs.MoveObject()** to translate





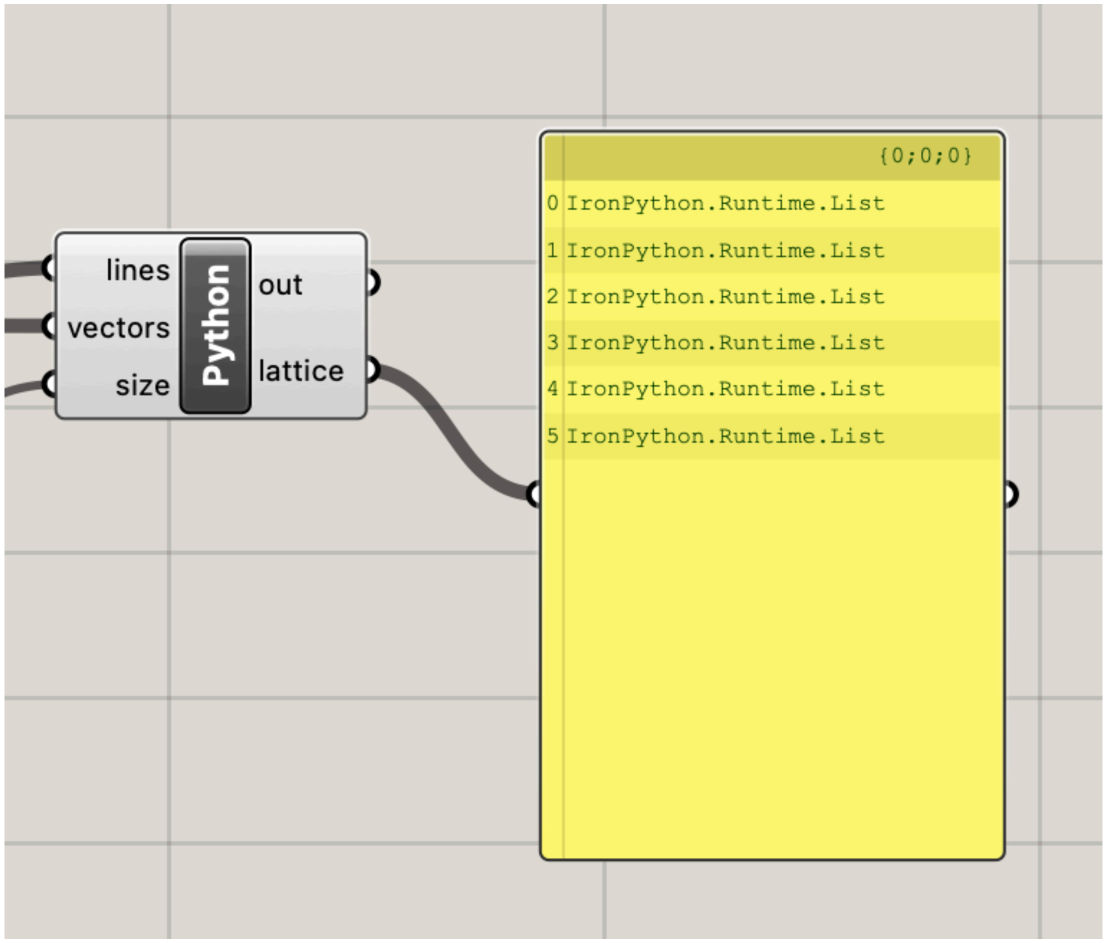
Approach: 2D Lattice

```python
import rhinoscriptsyntax as rs
import math
import copy

lattice = []
for i in range (0,size+1):
    row = []
    for j in range(0,size+1):
        new_lines = copy.deepcopy(lines)
        rs.MoveObject(new_lines,vectors[0]*i) #translate cells along the a vector
        rs.MoveObject(new_lines,vectors[1]*j) #translate cells along the b vector
        row.append(new_lines)
    lattice.append(row)
```

questions?

Lattice output in Grasshopper



```
{0;0;0}
0 IronPython.Runtime.List
1 IronPython.Runtime.List
2 IronPython.Runtime.List
3 IronPython.Runtime.List
4 IronPython.Runtime.List
5 IronPython.Runtime.List
```

Grasshopper & Python Data Structures

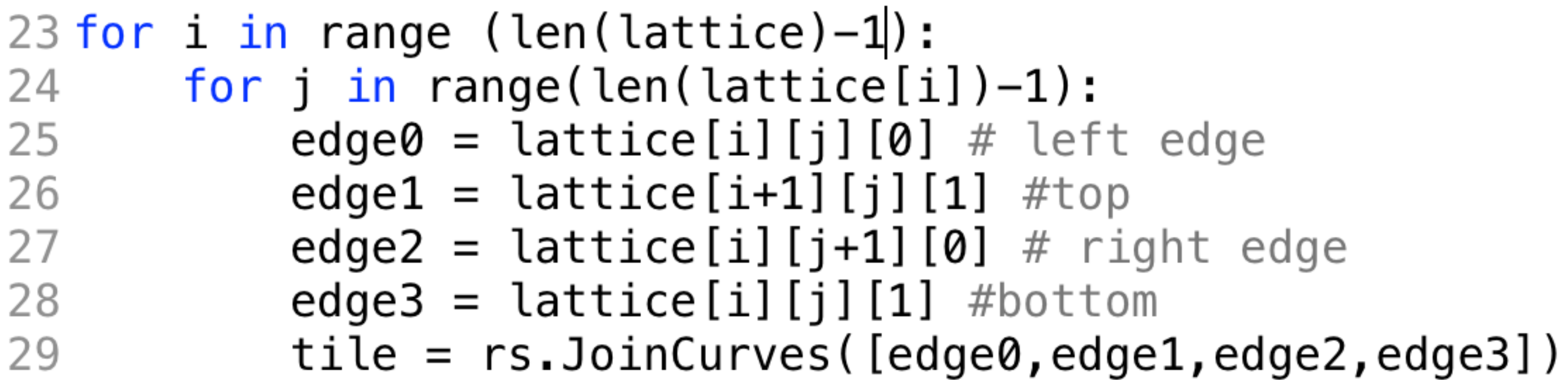
- Python: lists, arrays
- Grasshopper: 1D lists and trees only
- Grasshopper can't handle arrays :'(
  - Can't manipulate data from arrays
  - Can't render/visualize data from arrays

Lattice —> Tiles

2D Array of Lines —> 1D List of Closed Curves

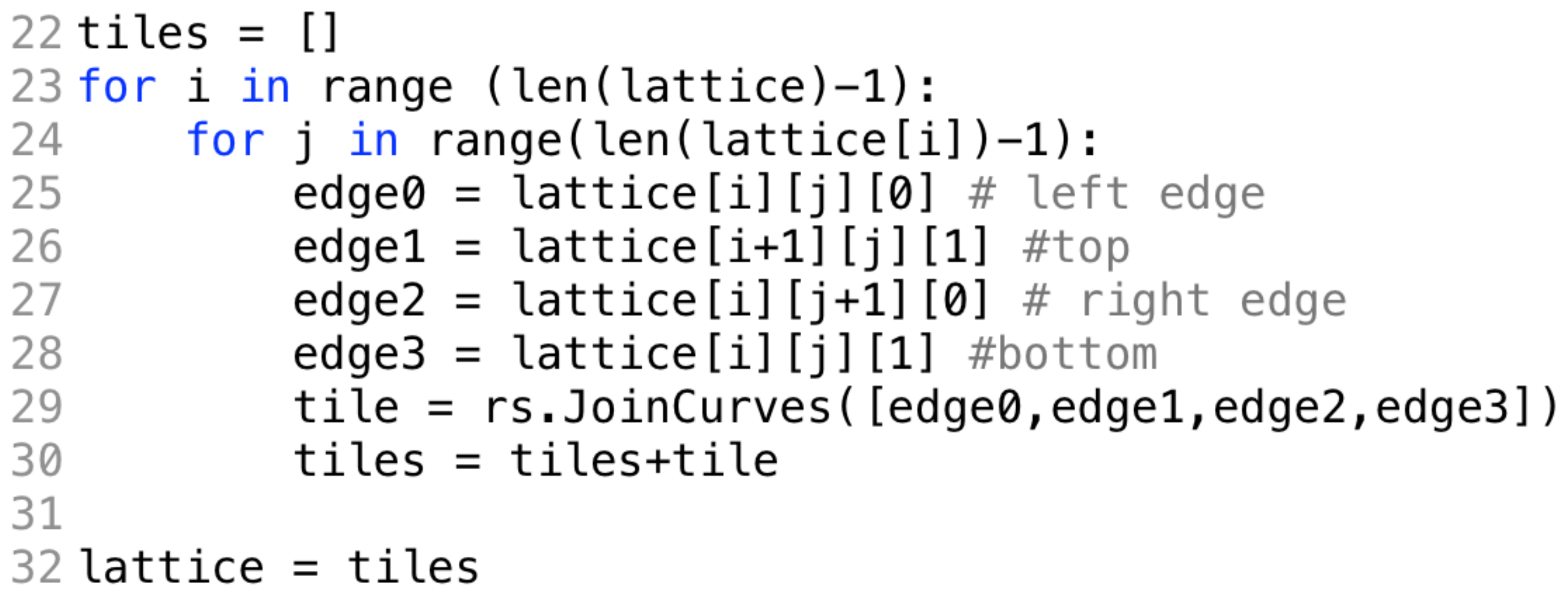
- Two tasks:
  1. Generate Tiles (Closed Curves) from lines
  2. Generate 1D List of Tiles as output

Find Tile Edges & Generate Tile



```python
for i in range (len(lattice)-1):
    for j in range(len(lattice[i])-1):
        edge0 = lattice[i][j][0] # left edge
        edge1 = lattice[i+1][j][1] #top
        edge2 = lattice[i][j+1][0] # right edge
        edge3 = lattice[i][j][1] #bottom
        tile = rs.JoinCurves([edge0,edge1,edge2,edge3])
```

Add each tile to **tiles** list

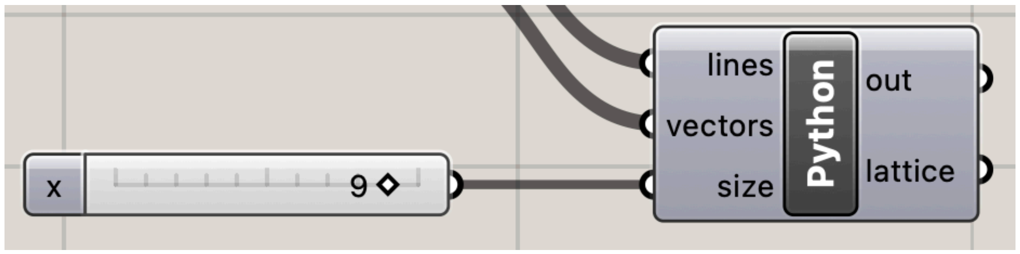


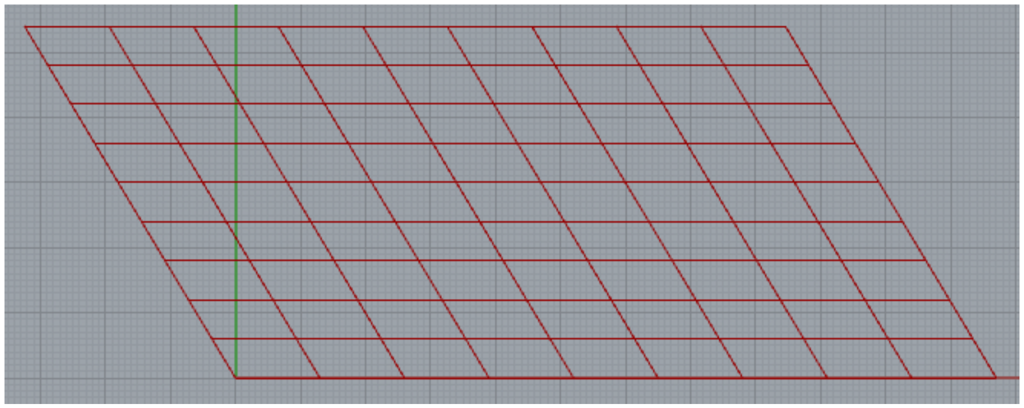
```python
tiles = []
for i in range (len(lattice)-1):
    for j in range(len(lattice[i])-1):
        edge0 = lattice[i][j][0] # left edge
        edge1 = lattice[i+1][j][1] #top
        edge2 = lattice[i][j+1][0] # right edge
        edge3 = lattice[i][j][1] #bottom
        tile = rs.JoinCurves([edge0,edge1,edge2,edge3])
        tiles = tiles+tile

lattice = tiles
```

Grasshopper & Python

- Inputs:
  - lines
  - vectors
  - size of lattice
- Output:
  - 2D lattice
    as list of tiles
    tile = closed curve





questions?

What we'll do today

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

What we'll do today

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

What we’ll do today

2. Use our lattice generating code to generate 2D tiles and tilings.

   Adding some Escher-like tile manipulation

Suggestions for an approach?

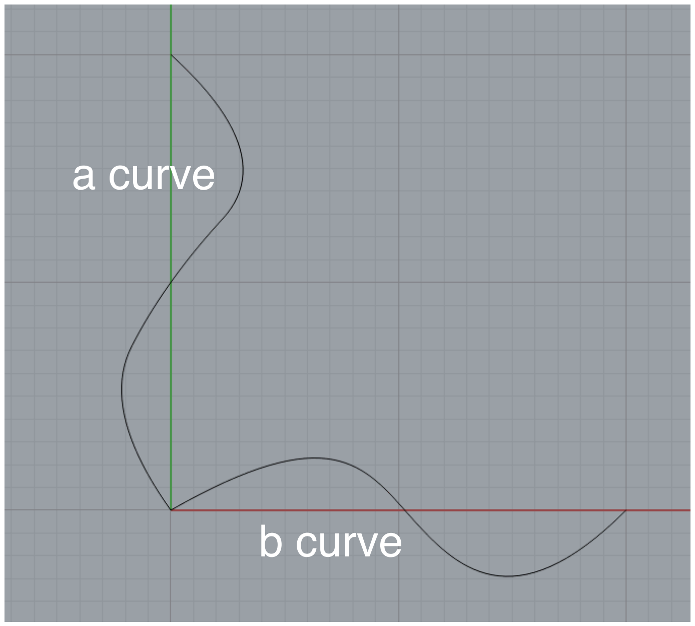
Approach

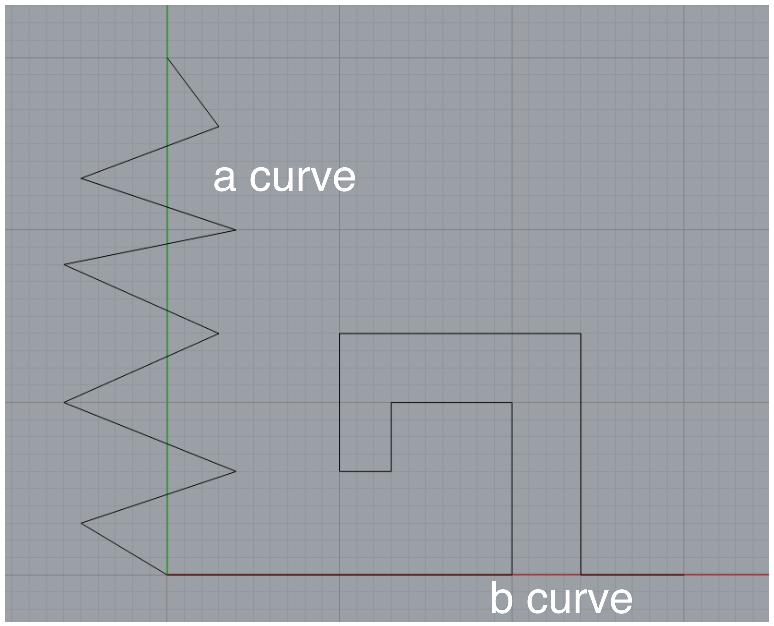
1. Allow Escher input curves as **a** and **b** curves of lattice.
2. Input curve requirements:
   - **a** curve: begins at origin and ends at point on y axis
   - **b** curve: begins at origin and ends at point on x axis
3. Edit first Python block
   - Accept Escher curves as input
   - Output appropriately scaled and rotated Escher curves.

questions?

Draw Curves in Rhino

- **a** curve: begins at origin and ends at point on y axis
- **b** curve: begins at origin and ends at point on x axis



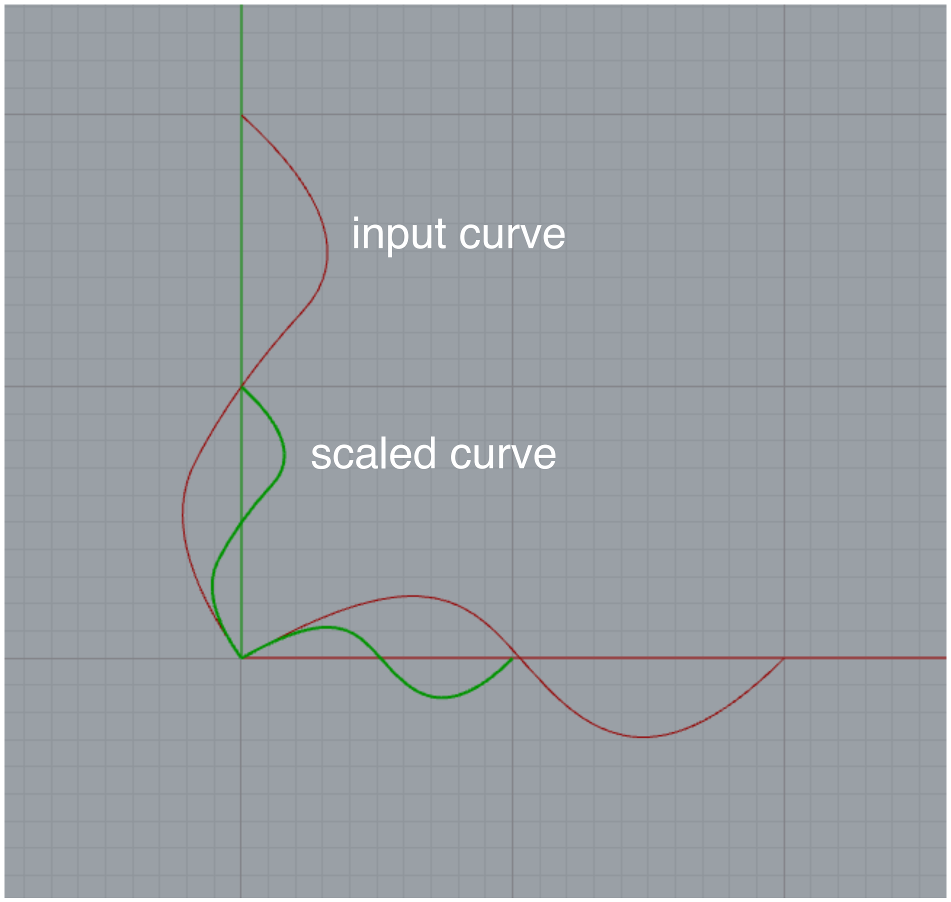


Scale Curves to fit Lattice

1. Use **rs.CurveEndPoint()** to find end points of curves.
2. What does the end point tell us about the length of curve **a**?
3. Use **rs.ScaleObject()** to scale each curve
4. What is the scale factor for curve **a**?

```python
#scale curves to match magnitude inputs
curve_a_length=rs.CurveEndPoint(curve_a).Y
a_scale = a_length/curve_a_length
rs.ScaleObject(curve_a, point, rs.CreatePoint(a_scale,a_scale,1))
```

Scale Curves to fit Lattice

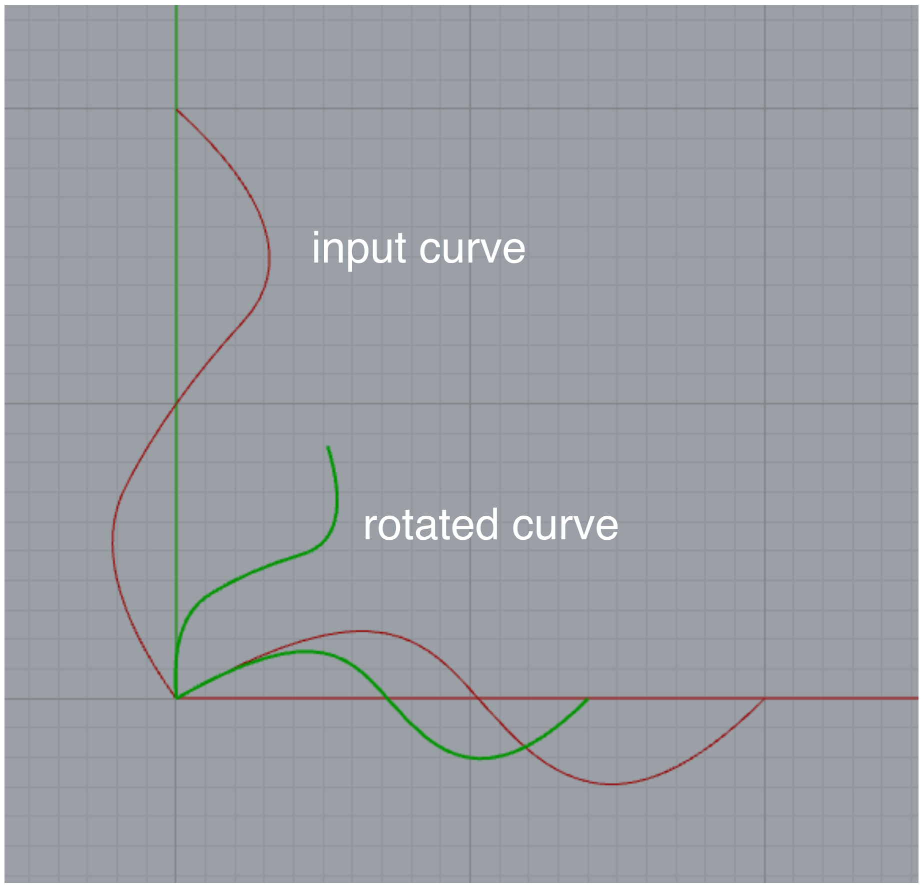


Rotate Curves to fit Lattice

1. Which curves do we have to rotate?
2. What is the rotation angle in terms of the input angle?

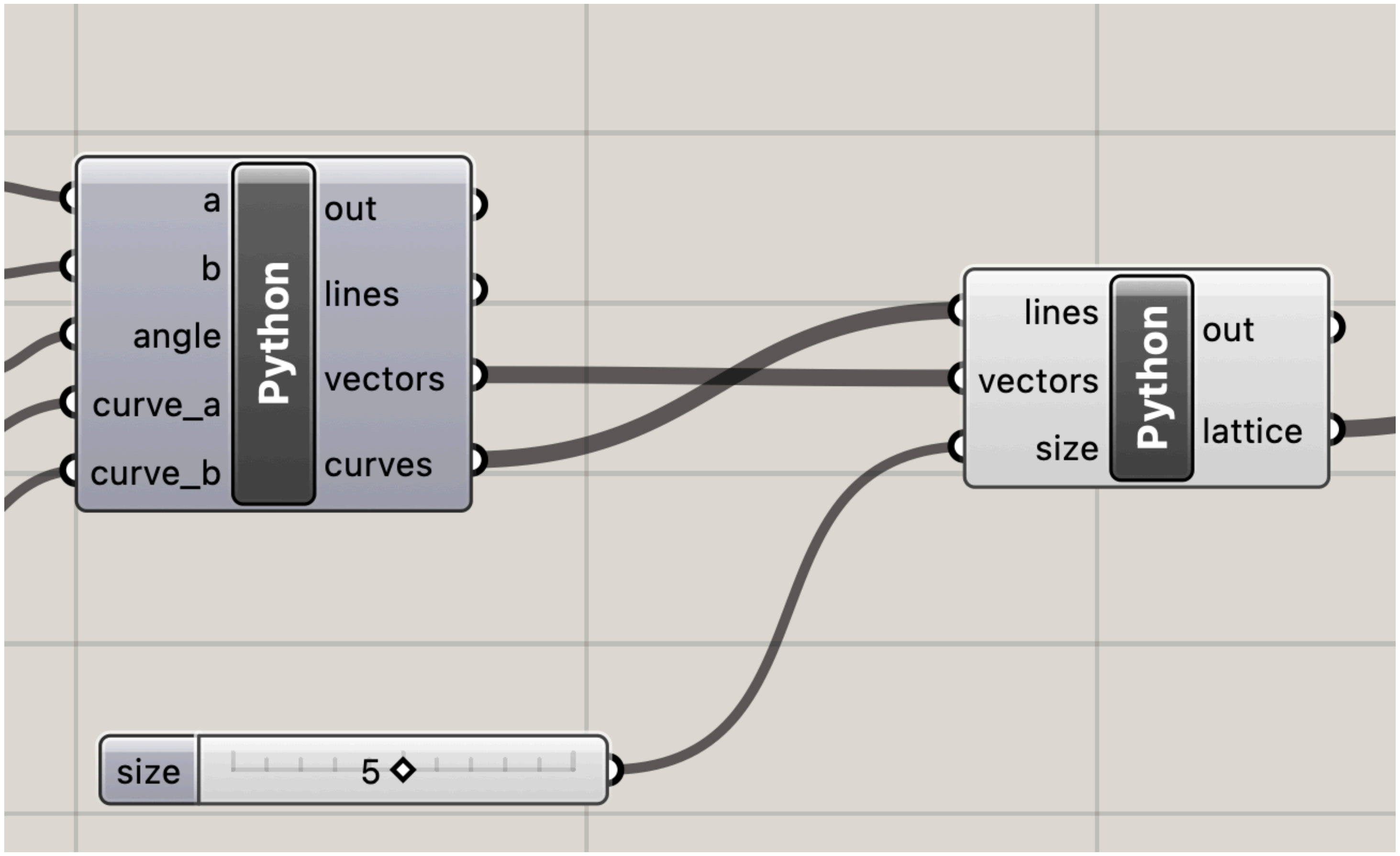
```
rs.RotateObject(curve_a,point,angle-90)
```

Rotate Curves to fit Lattice

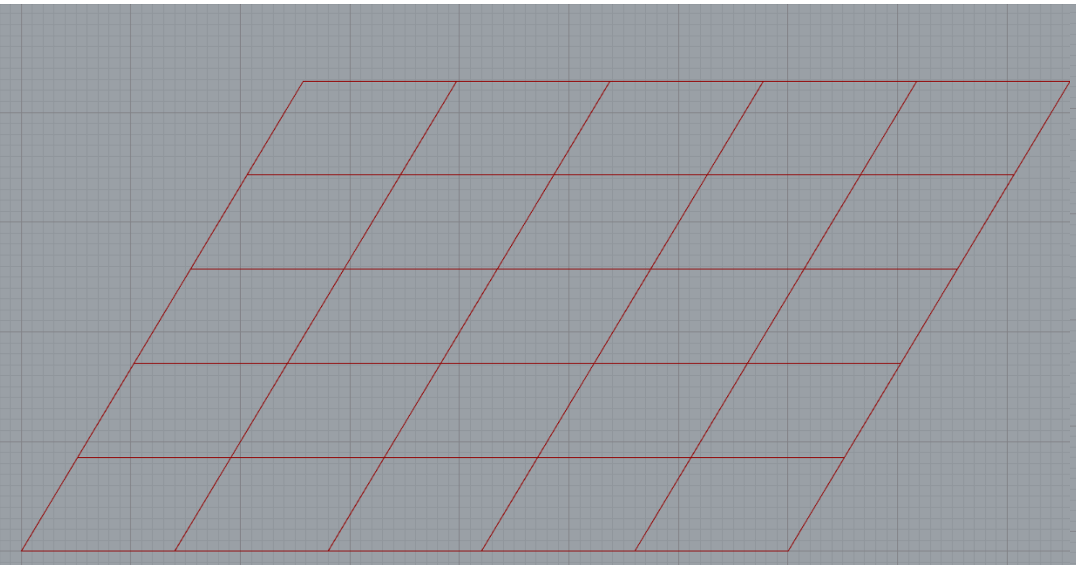


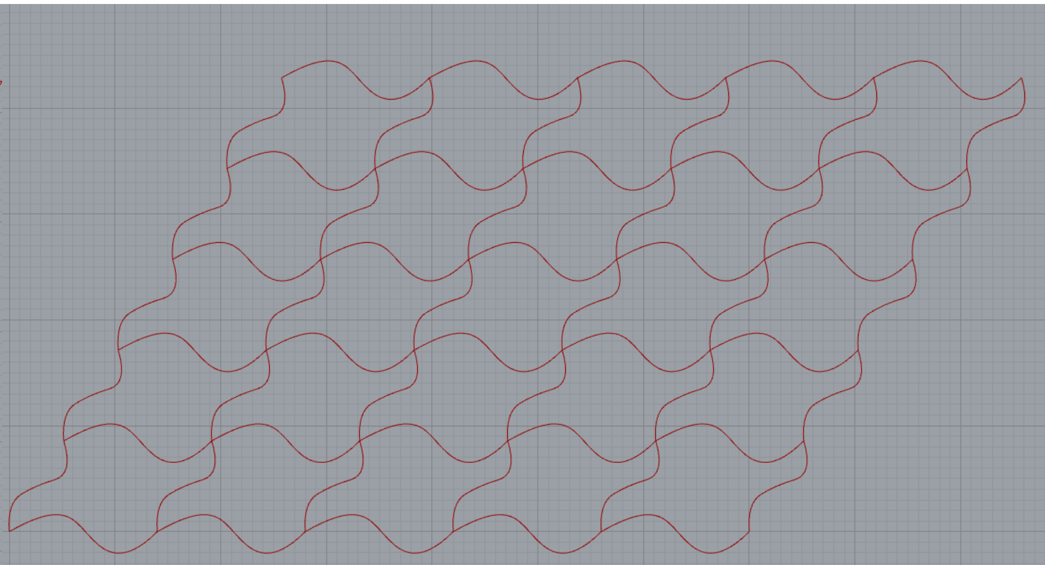
questions?

Connect Curves to Lattice Code

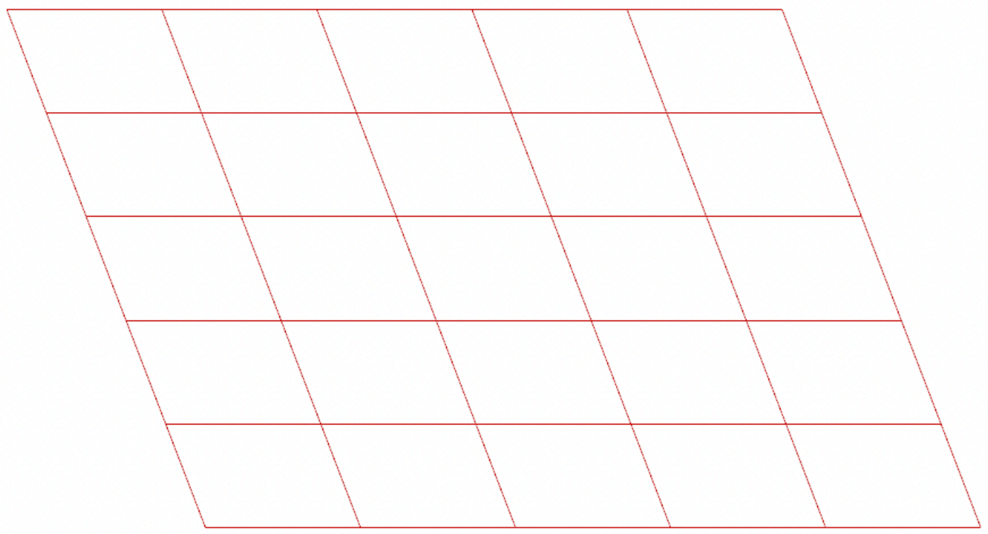


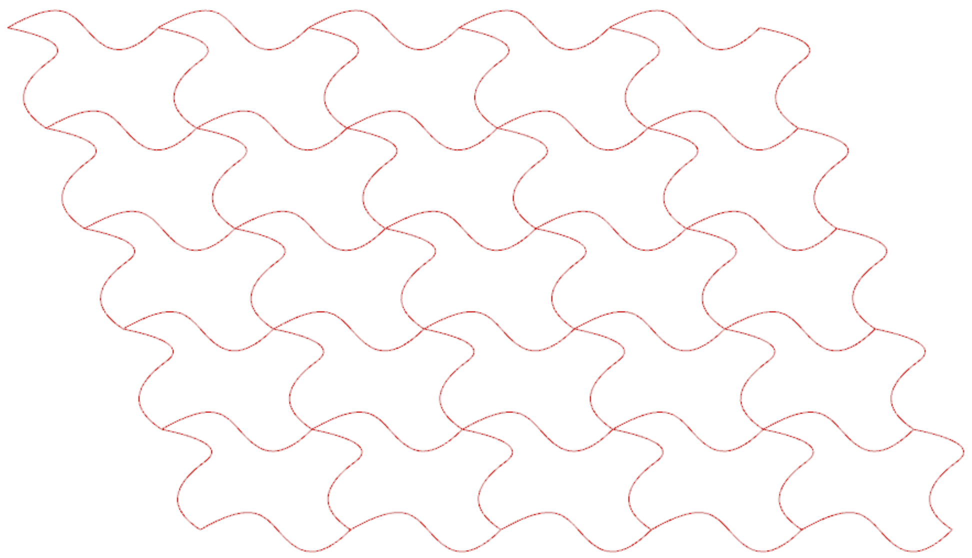
Connect Curves to Lattice Code


Lattice with lines


Lattice with Escher curves

Connect Curves to Lattice Code





Rendered view in Rhino

Thank you!

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