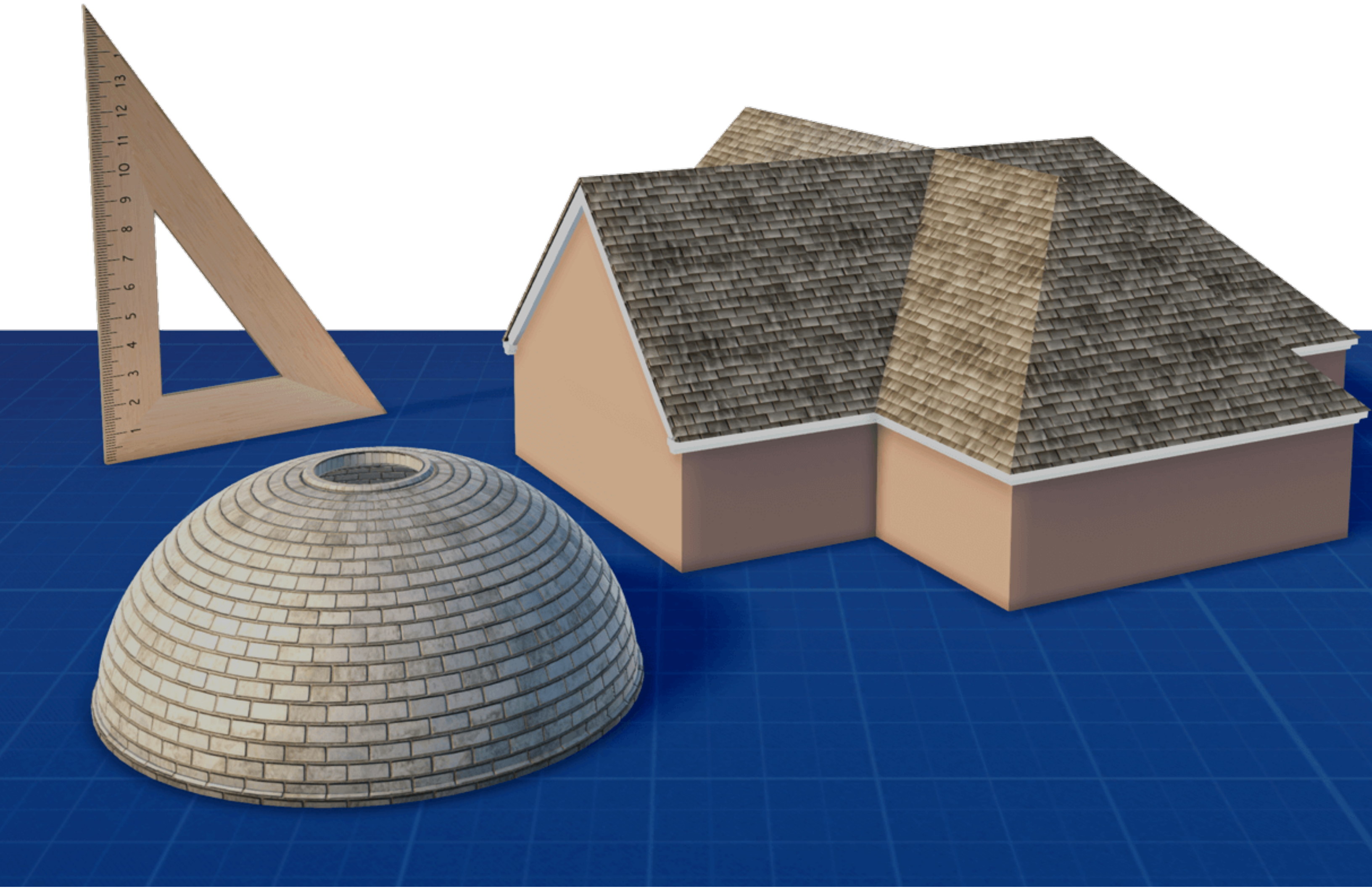
ARCHIMATIX DOME and ROOFTOP FUNDAMENTALS

by wetcircuit

Skill Level: intermediate

- 20+ mini-projects: build spline models to learn construction methods.
- Expose the controls that style each roof for parametric and runtime.
- Create *all* traditional rooftops in **Archimatix**. Fit to *any* building plan.
- Design custom roof shapes for historic and fantasy villages.
- Deconstruct complex multi-level rooftops into manageable parts.
- Get out of the “layer cake” building rut with detailed domes and roofs.



Introduction

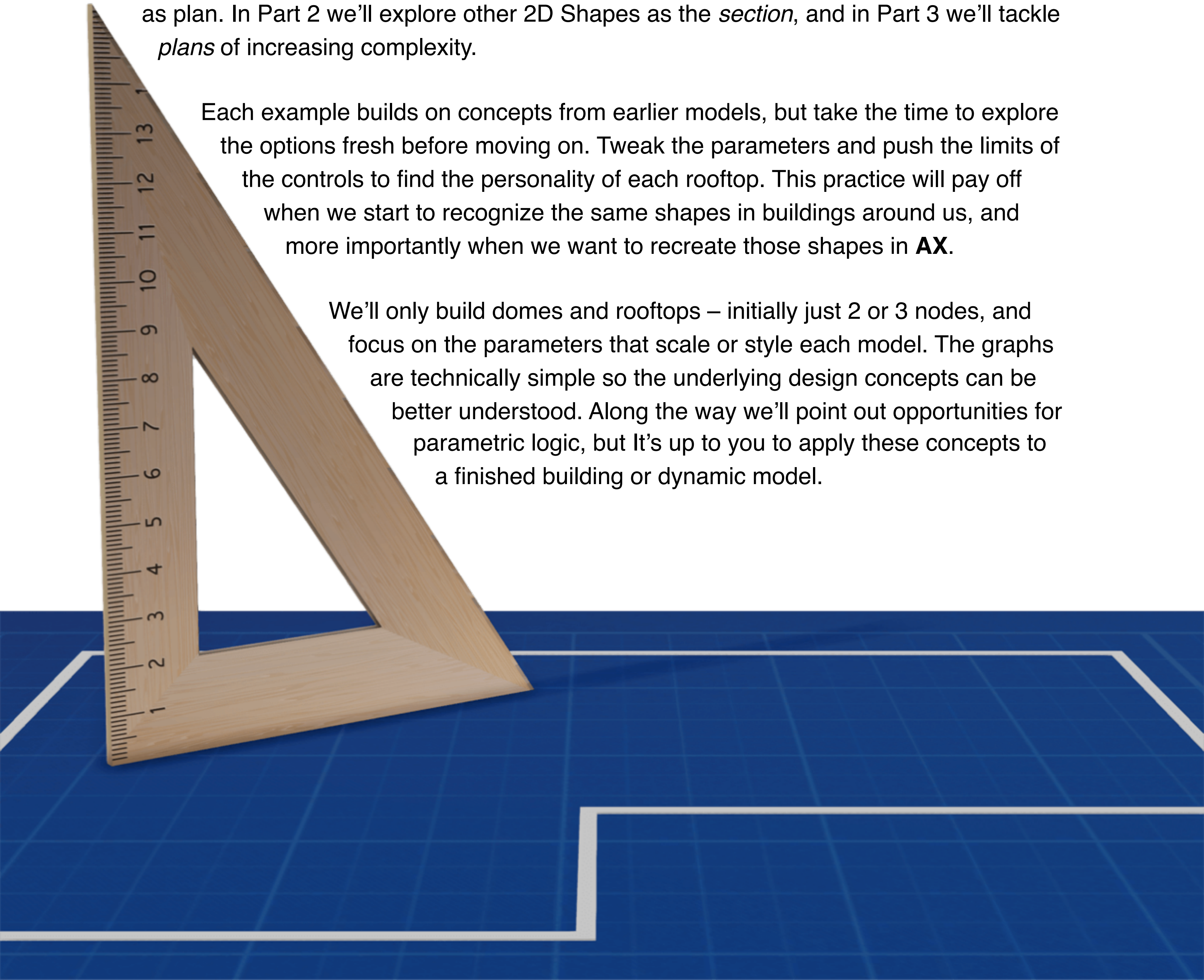
Spline modeling is not always intuitive, so let's start with some basic definitions.

The structure of every spline model is a 2D shape in the vertical plane called the *Section*, and a different 2D shape in the horizontal plane called the *Plan*. The *section* is swept from point to point along the *plan*. **Archimatix** can change a model's axis of orientation, but this default is exactly what we need: *section* will always be our vertical shape and *plan* is the horizontal.

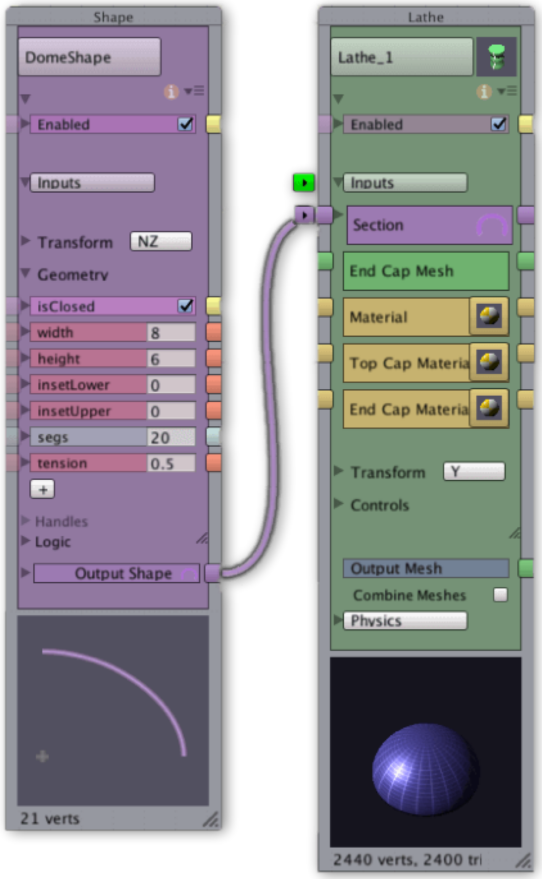
This tutorial is organized around the fundamental spline structure of *Section* and *Plan*. In Part 1 we'll learn rooftop basics with just two nodes: **DomeShape** as section, **Lathe** as plan. In Part 2 we'll explore other 2D Shapes as the *section*, and in Part 3 we'll tackle *plans* of increasing complexity.

Each example builds on concepts from earlier models, but take the time to explore the options fresh before moving on. Tweak the parameters and push the limits of the controls to find the personality of each rooftop. This practice will pay off when we start to recognize the same shapes in buildings around us, and more importantly when we want to recreate those shapes in **AX**.

We'll only build domes and rooftops – initially just 2 or 3 nodes, and focus on the parameters that scale or style each model. The graphs are technically simple so the underlying design concepts can be better understood. Along the way we'll point out opportunities for parametric logic, but It's up to you to apply these concepts to a finished building or dynamic model.

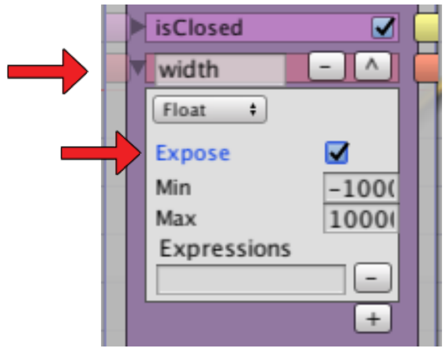


PART 1: DomeShape and Lathe

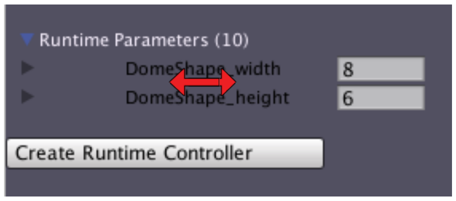


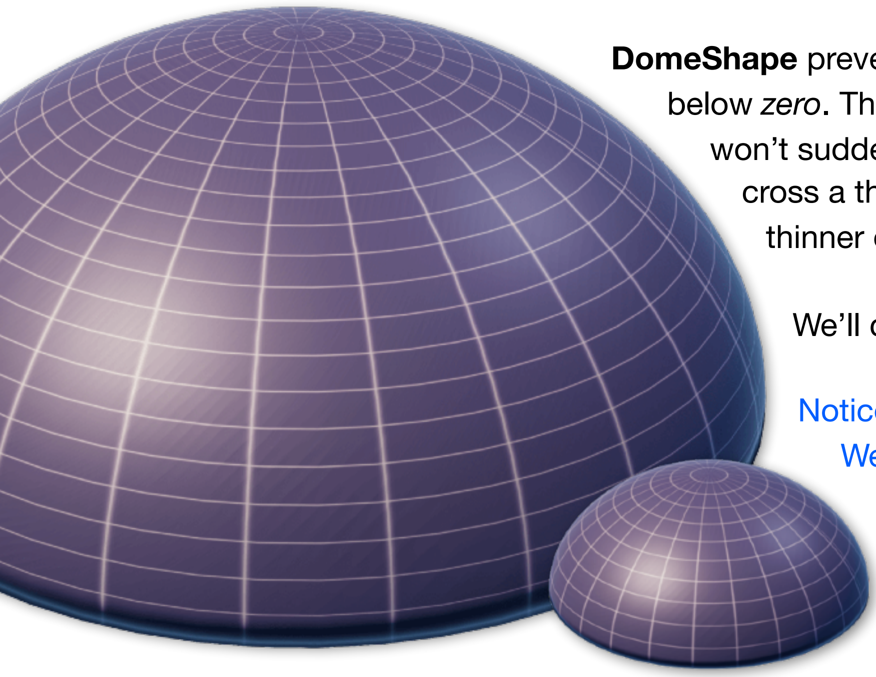
Connect a **DomeShape** node to a **Lathe** node to create a simple 2-node dome.

Expose **DomeShape Width** and **Height**.



We can make quick changes to our dome by dragging the exposed controls in the inspector under the **Runtime Parameters**.

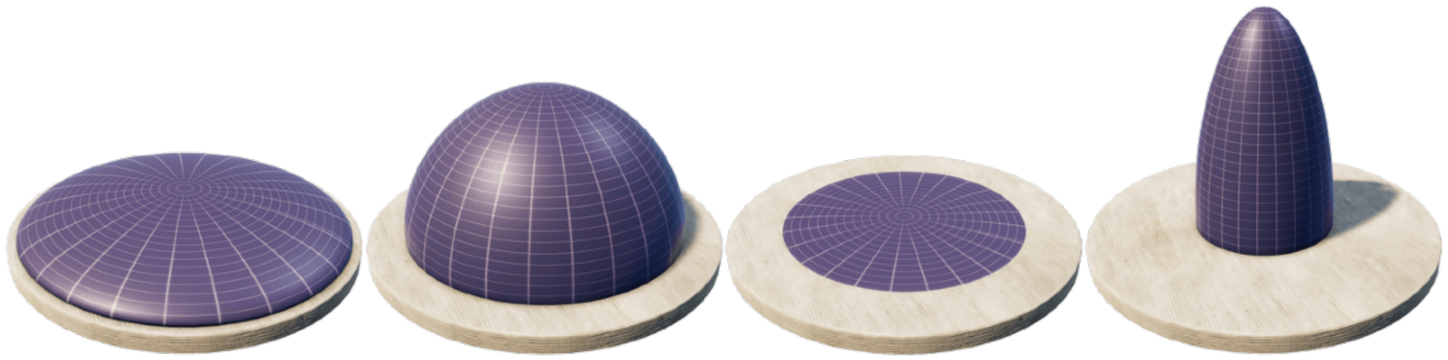




DomeShape prevents us from setting **width** and **height** below *zero*. The dome can't invert into a bowl, and it won't suddenly blink out of existence when we cross a threshold. Instead our dome just gets thinner or flatter until it can't go any further.

We'll call this *failing gracefully*.

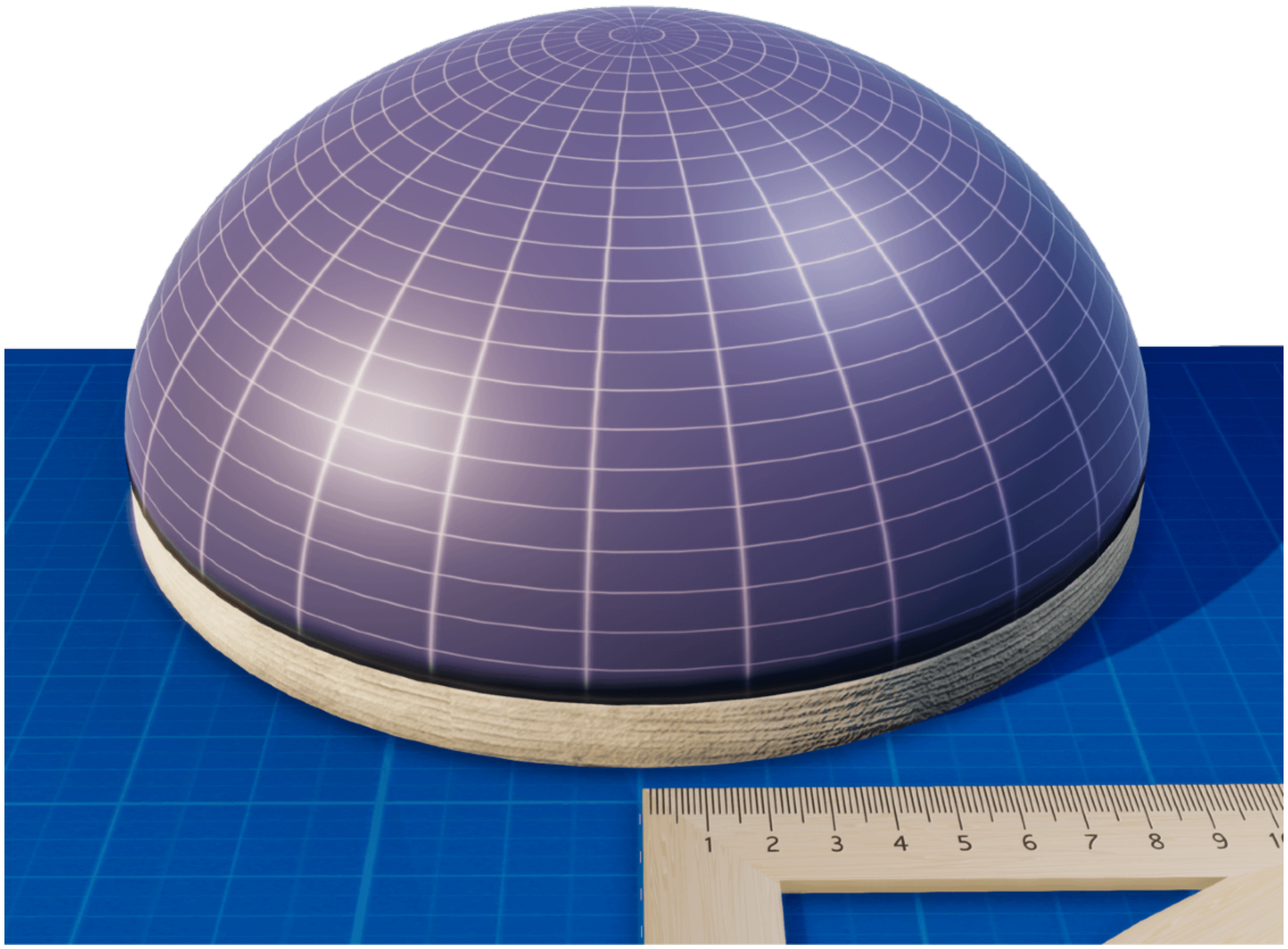
Notice the *ratio* between **width** and **height.** We can *scale* our dome by keeping the ratio. A 3:2 dome is the same curve as a 6:4 dome, and a 12:8 dome….



After playing around with our simple dome, we can establish some basic logic for parametric relations:

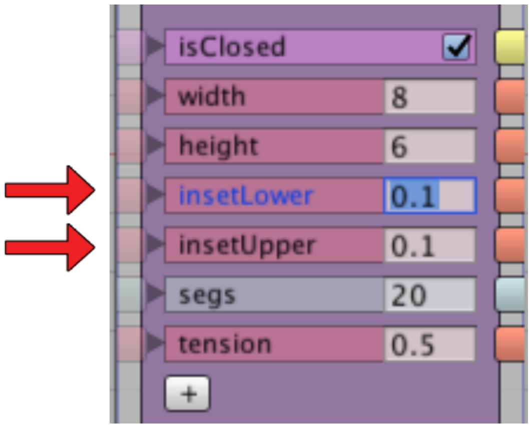
The **width** of the **DomeShape** node is *half the span* of our dome. If we know the *span* of a building, we can create a parametric dome that fits it, and vice-versa.

The **height** of the dome can be related as a *ratio* of the **width.** By maintaining a parametric *height-to-width ratio*, the dome will scale uniformly.

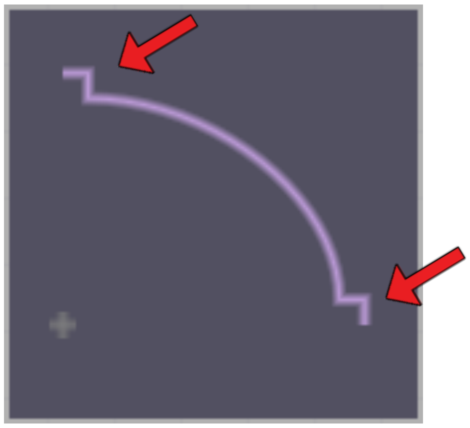


Oculus Dome

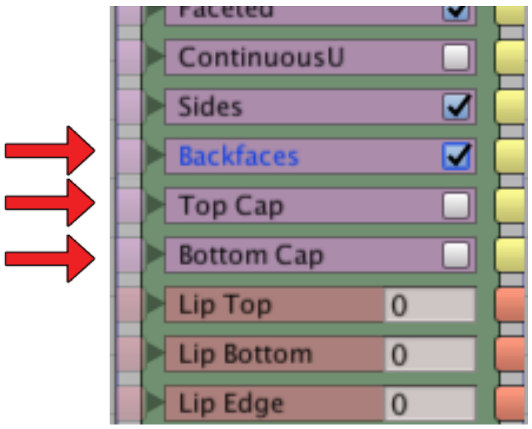
Let’s make some cosmetic changes to our dome and create our first rooftop:



isClosed	✓
width	8
height	6
insetLower	0.1
insetUpper	0.1
segs	20
tension	0.5

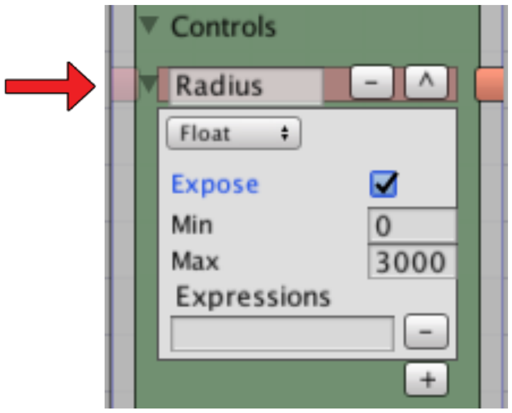


On the **DomeShape** node change **insetLower** and **insetUpper** to *0.1*. This creates architectural notches at the bottom and top of the dome.



Faceted	✓
ContinuousU	☐
Sides	✓
Backfaces	✓
Top Cap	☐
Bottom Cap	☐
Lip Top	0
Lip Bottom	0
Lip Edge	0

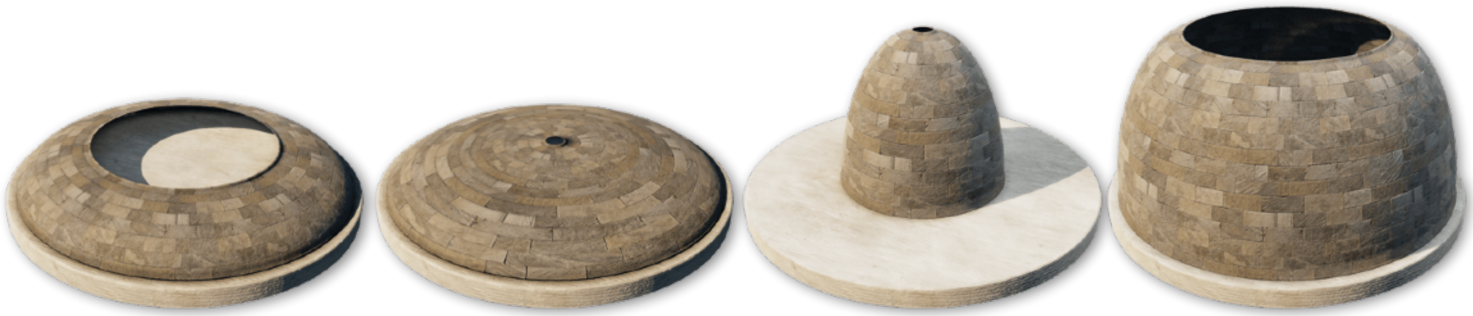
We’ll make changes to the **Lathe** node too. Toggle **Backfaces** *on*, and switch **Top Cap** and **Bottom Cap** *off*.



Radius: Float, Expose ✓, Min 0, Max 3000

To create the defining feature of the *oculus dome*, expose the **Radius** parameter of the **Lathe** node.

Lathe may seem like the puny little brother of **PlanSweep**, but it takes shortcuts when creating a spline model that make it faster to set up and easier to use in some situations. **Lathe** uses a built-in polygon shape for its *plan*, and has a few more tricks we're about to see.

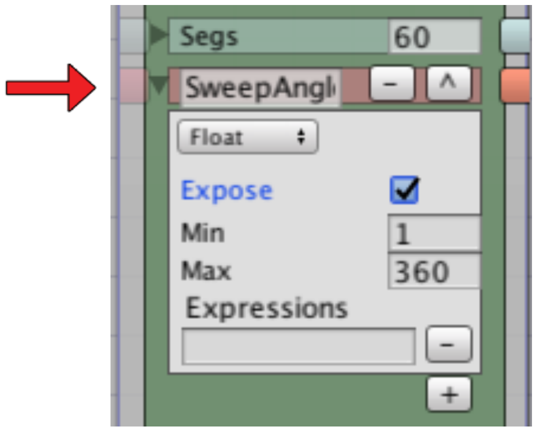


Explore the new relationship between the **Width** of the dome and the **Radius** of the oculus gap. Notice how **DomeShape**'s curve is preserved when we open the oculus. We can make our roof span wider while preserving the dome's curve profile.

Our parametric logic needs to be updated to include the oculus gap. Our new roof span is *double the sum* of the dome's **Width** and the **Lathe**'s **Radius**.

Alcove, Half Shell, Corner Arc

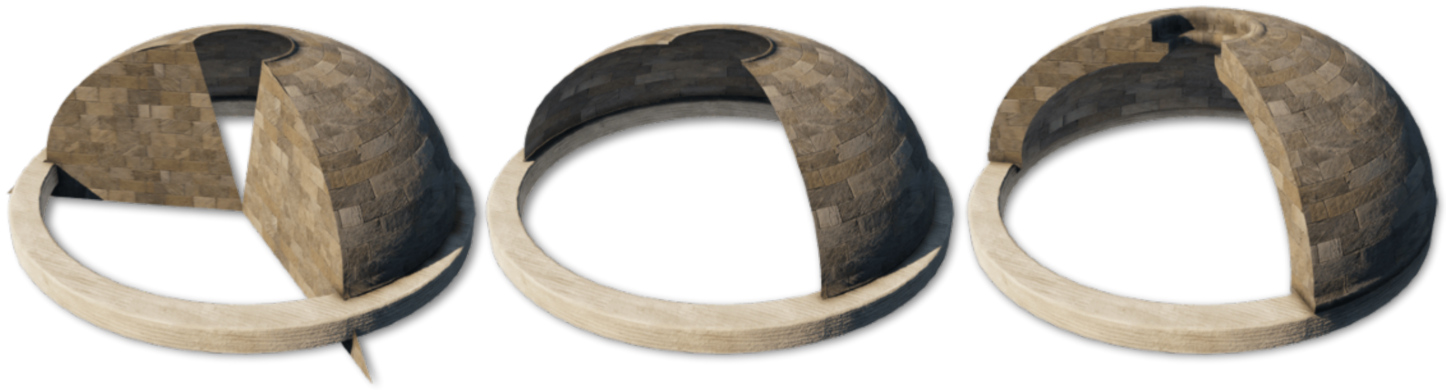
Lathe has another trick that would require a half-dozen nodes with a **PlanSweep**. We can create *alcoves* and *half domes* by setting the **SweepAngle**. 180° is a *half-dome*. 90° is a *quarter-dome*. 360° is a closed *circle*.



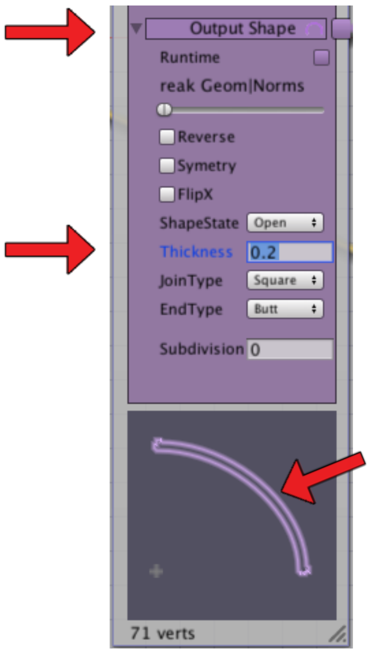
Segs 60
SweepAngl
Float
Expose ✓
Min 1
Max 360
Expressions

On the **Lathe** node, expose **SweepAngle**, and we'll also expose a complimentary parameter under Transform: **Rot_Y**. Together these settings can create a partial dome shell that rotates any direction.

If we see the ragged end caps (below left), it's because **DomeShape** is an *open* 2D shape. Switch *off* **End Cap A** and **End Cap B** on the **Lathe**, and with the **Backfaces** switched *on* we get the thin 2-sided mesh (below center):



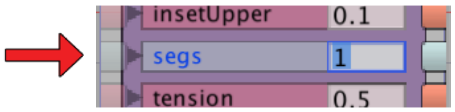
To create *thickness* on the shell like the third example, open the *Output Shape* on the **DomeShape** node and add a small amount of **Thickness**, about *0.2*. Since our dome is now a *closed* 2D shape we can turn *off* **Backfaces** on the **Lathe** node, and turn *on* the **End Caps**.



Output Shape
Runtime
reak Geom|Norms
Reverse
Symetry
FlipX
ShapeState Open
Thickness 0.2
JoinType Square
EndType Butt
Subdivision 0
71 verts

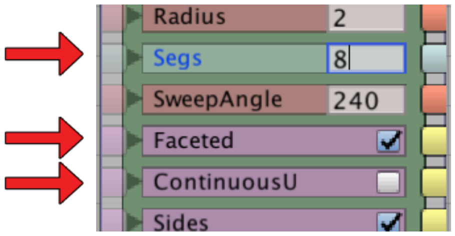
Be aware of the unseen polygons in our models. **Thickness** and **Backfaces** increase the polygon count geometrically. They should probably never be used together.

Gazebo, Pavilion, Turret



insetUpper	0.1
segs	1
tension	0.5

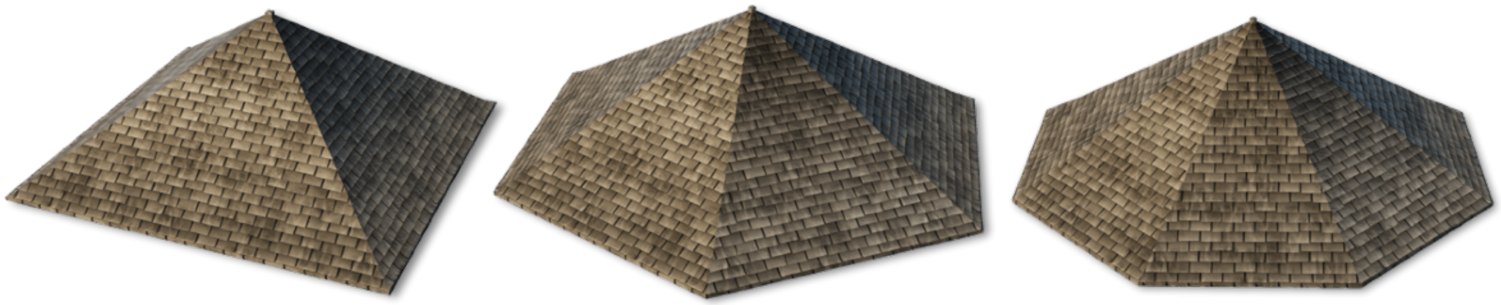
On the **DomeShape**, change **InsetLower** to *0*. Expose the **Segs** parameter. Leave **Segs** at *1* for this lesson. Ta-dah! A pitched-roof.



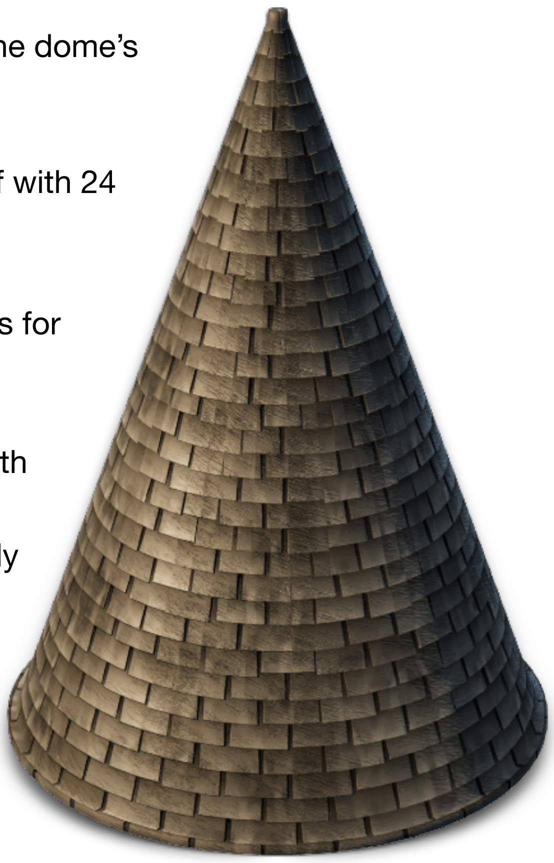
Radius	2
Segs	8
SweepAngle	240
Faceted	✓
ContinuousU	☐
Sides	✓

On the **Lathe** Node, expose the *other* **Segs**, and also two Booleans: **Faceted** and **ContinuousU**.

DomeShape Segs is the number of facets in our roof's *section*. **Lathe Segs** is the number of sides on our roof *plan*.



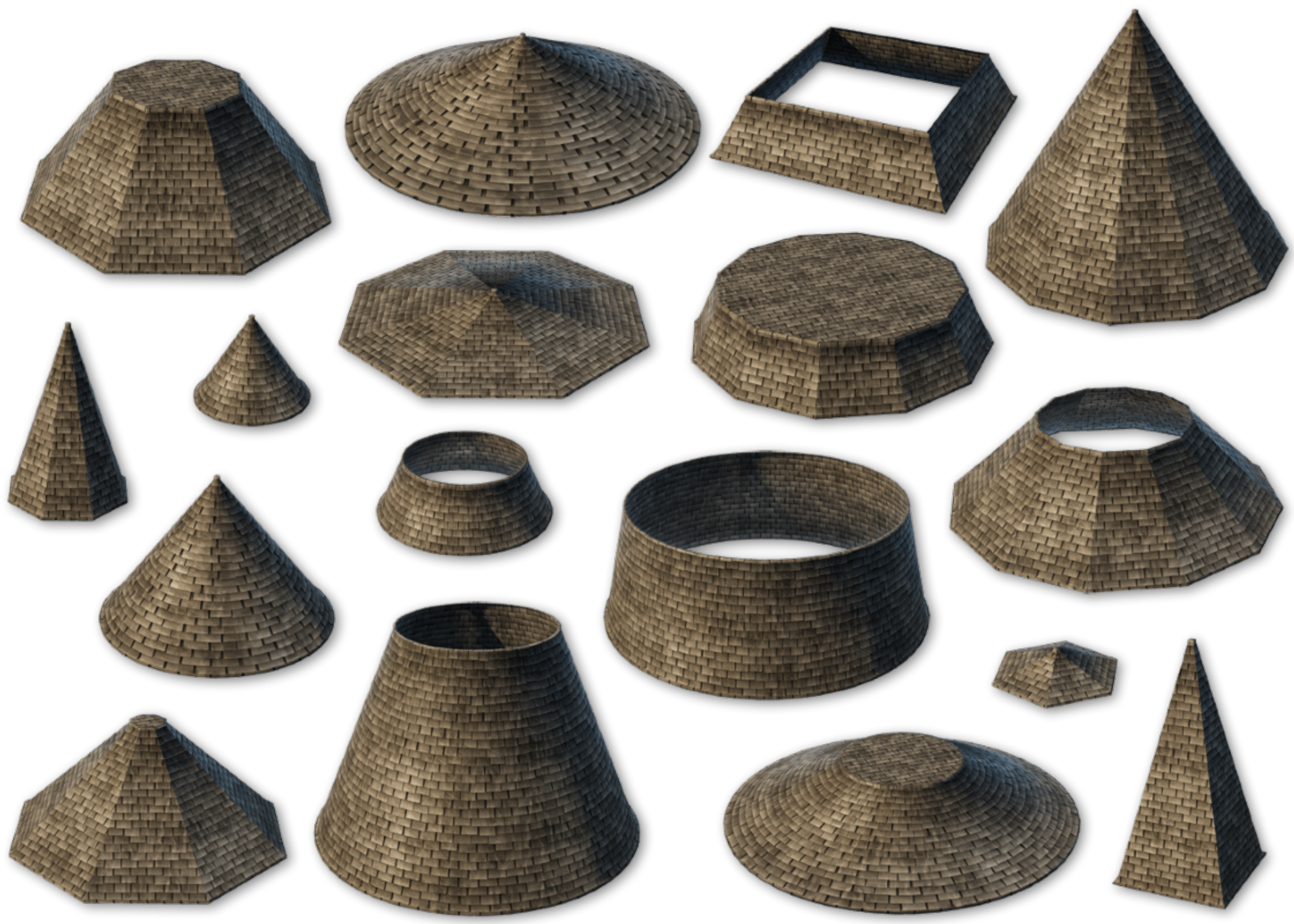
- a 4-sided roof is called a *square hip* or *pyramid*. If the dome's **Height** is very low it's called a *pavilion* roof.
- *Gazebos* have a low roof with 6, 8, or 12 sides.
- A *tower* or *turret* might have a tall cone-shaped roof with 24 **Segs** or more.



As we transition from cone to pyramid, **Lathe** has options for dealing with the UV: **Faceted** and **ContinuousU**.

- Turn **Faceted** *on* with low segment pyramids, *off* with round cones and domes.
- **ContinuousU** attempts to wrap the texture smoothly around the **Lathe**, but needs more **Segs** or the texture will distort. For pitched roofs turn **ContinuousU** *off* and textures will remain flat.

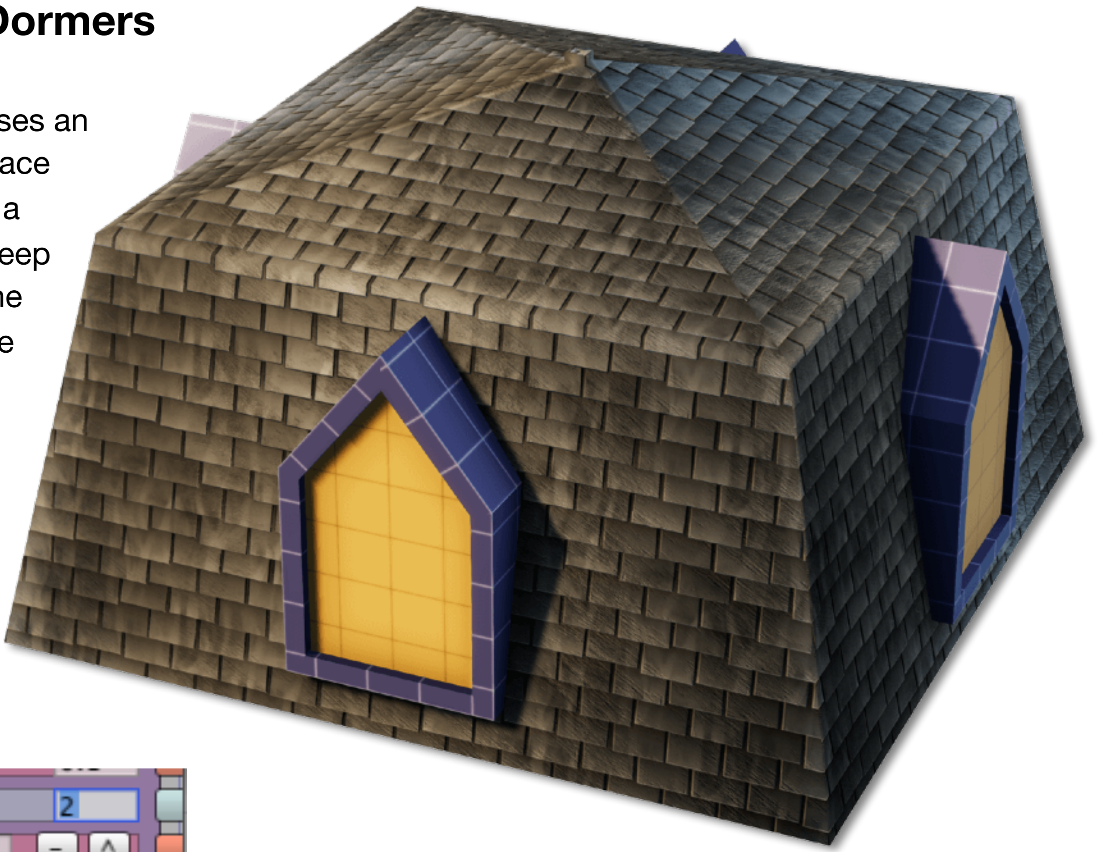
Let’s push through our *Runtime Parameters* again: *width* and *height* but also radius, with very low **Lathe** plan **Segs**: 4, 6, 8…, and very high: 24, 60… We can turn **Top Cap** *on* to cover the oculus gap, or imagine it capped with a ventilator, cupola, or skylight.



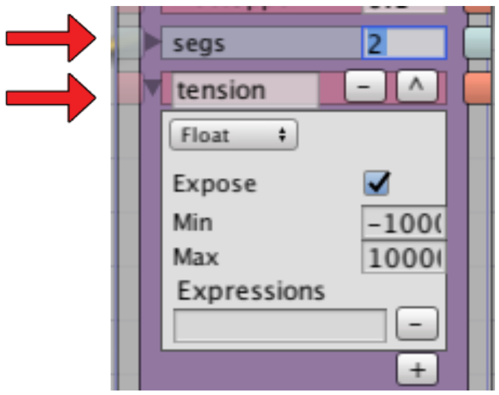
The goal isn’t to prove how many different shapes we can make, but to give some thought about the random shapes we are making. Does this roof resemble a castle? A bungalow? An information booth? What kind of people live here? What industry? What is their climate and economy? What purpose does each roof suggest?

Notice how the dome’s **height** plays a strong roll in the personality of a tiny rooftop…. A steep roof might suggest a snowy climate, while a wide roof implies a temporary structure or a welcoming portico…. Tall domes announce importance and power…. What we discover here is more than just a scaling rooftop, it’s a design lab to spur the imagination. Take the time to explore.

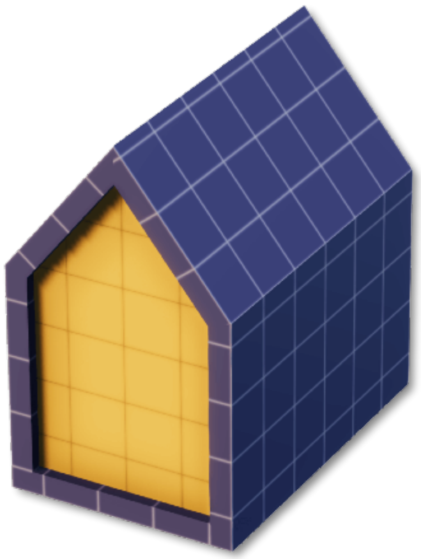
Mansard with Dormers



A *Mansard* roof encloses an extra floor of living space called a *garret,* under a *hip roof* with with a steep outside pitch. From the ground it may look like a flat roof but Mansards have a low pyramid peak. (For this roof the oculus **Radius** should be closed.)

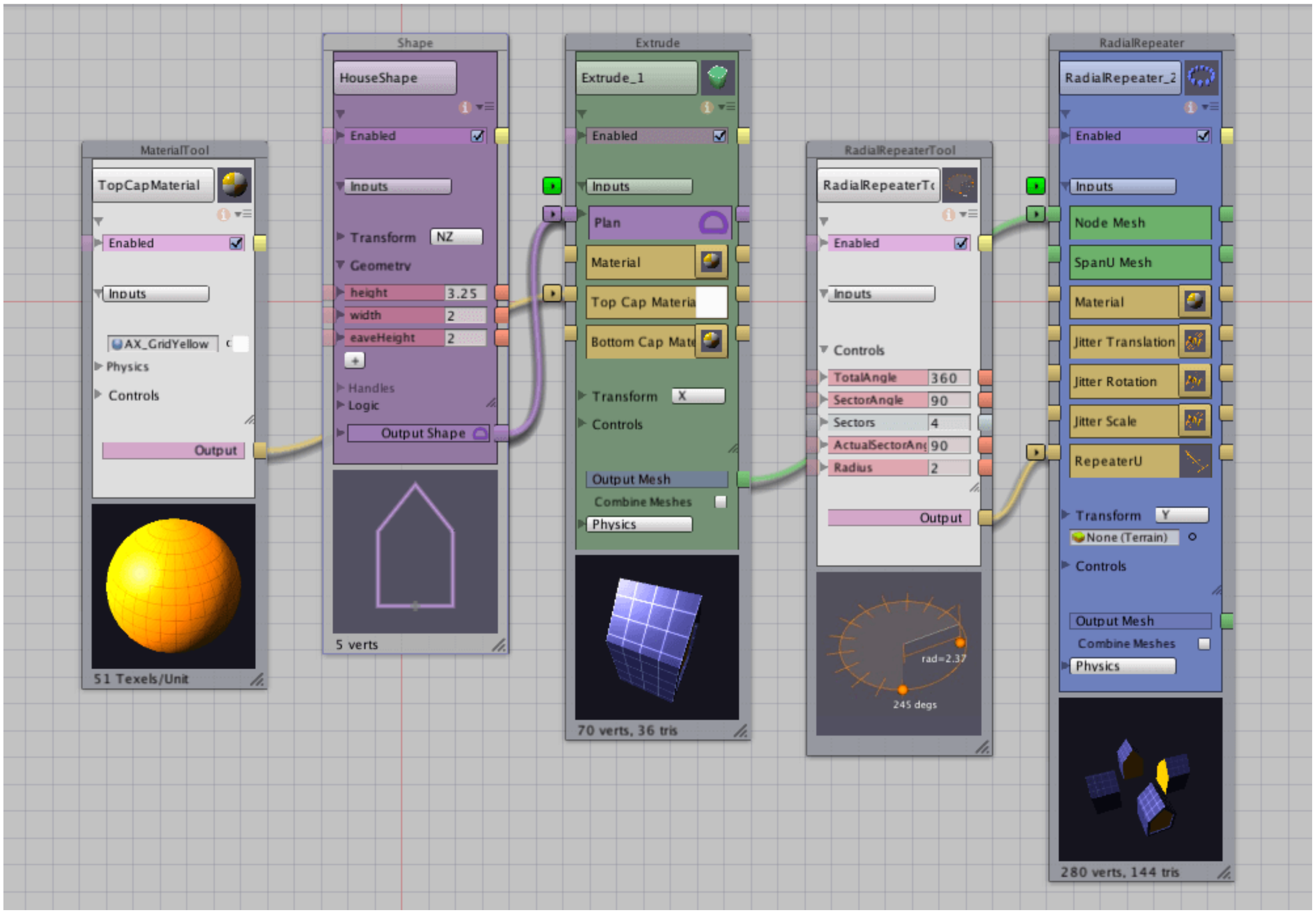


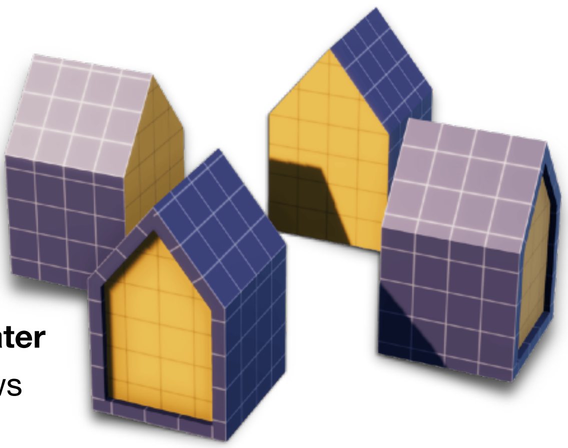
On the **DomeShape** node, set **Segs** to *2*.
Expose **Tension,** and keep it between *0.6* and *1*.



Create a simple *dormer window* prop:

1. Add a 2D **HouseShape** node to the graph.
2. Connect it to an **extrude** node, and expose the **Enabled** parameter so we can toggle the *window dormers* in the inspector.
3. Set the **Extrude** node's Transform *Axis* to *X*. Add a *Top Cap Material* for the window, and set the **Lip Top** and **Lip Edge** to add some depth.





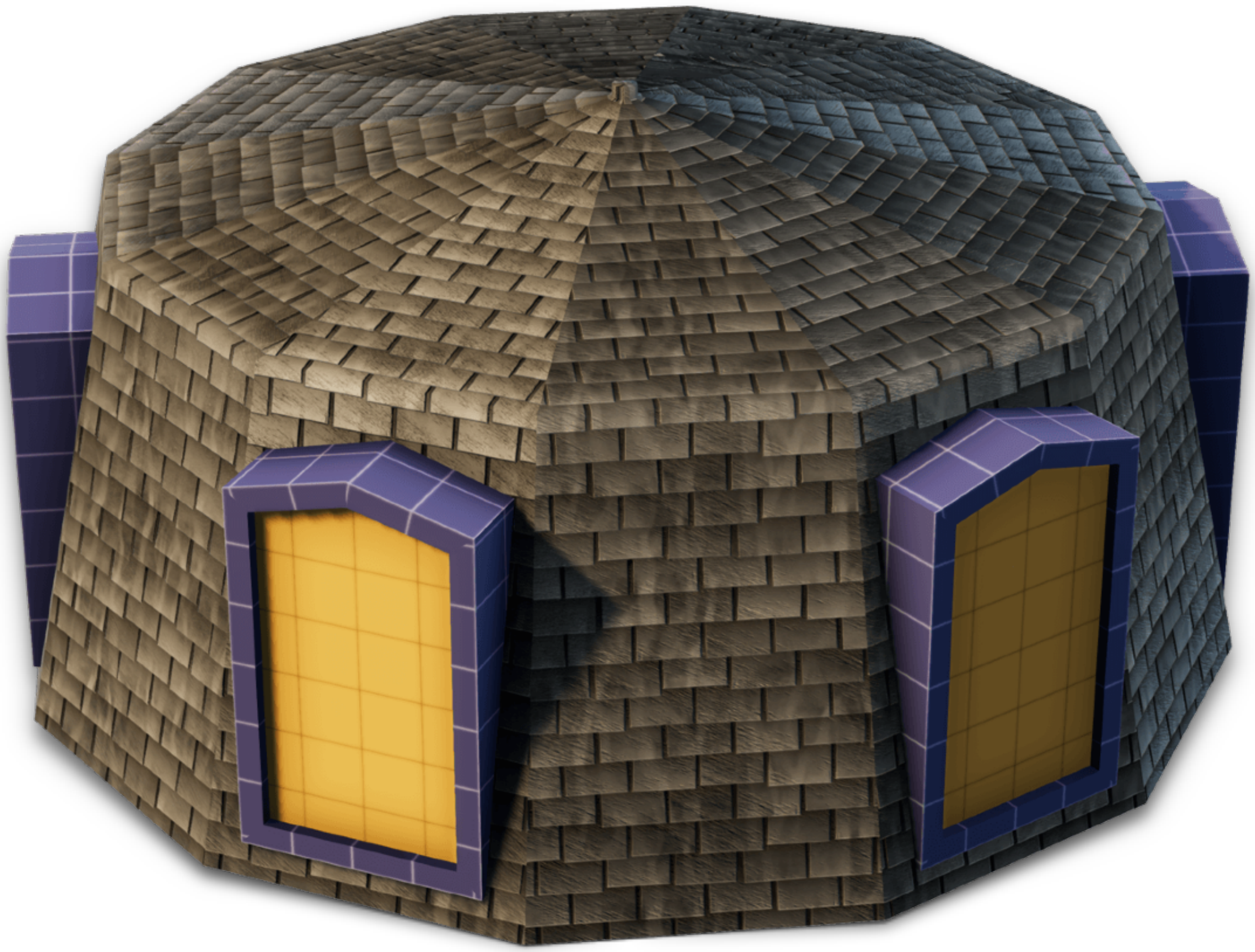
4. Connect the *Output Mesh* to a **RadialRepeater**.
5. Click on the nub next to **RepeaterU** to open the **RadialRepeaterTool.** Expose the *Repeatertool*'s **Radius** and **Sectors.**
6. The **Radius** of our *dormer windows* can be parametrically related to the **Radius** of our *oculus*. **Sectors** are related to the **Lathe Segs.**
7. Open the Transform settings on the **RadialRepeater** node, and expose **Trans_Y** and **Rot_Y.** This allows us to adjust the hight and rotation of the *dormer* group.
8. Lastly, let's expose the control parameters on our **HouseShape** node: **Height**, **Width** and **Eavesheight.**

A *garret* tower is essentially just a dressed-up roof dome, but the window props add personality and style as well as a sense of scale.

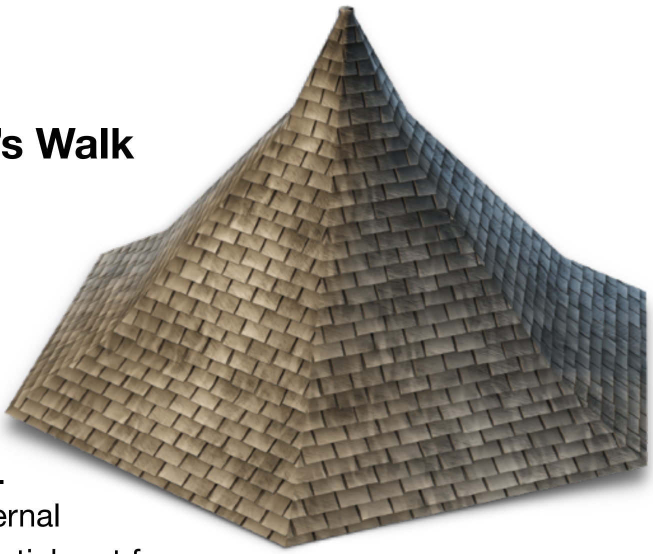
The gaps between the windows add just as much character as the windows themselves. Try a 2:1 or 3:1 ratio between **Lathe Segs** and **RadialRepeater Sectors** to get a tower with half or one-third as many windows.

Adjust the *dormer window* **HouseShape** parameters for different window personalities. Notice where the dormer's peak hits the Mansard break-line.

Play with the **Tension** parameter to adjust the pitch angles of the roof. The Mansard's profile is a combination of **Tension** and the roof's **Height** and **Width**.



Tent Roof, Witch Tower, Widow’s Walk



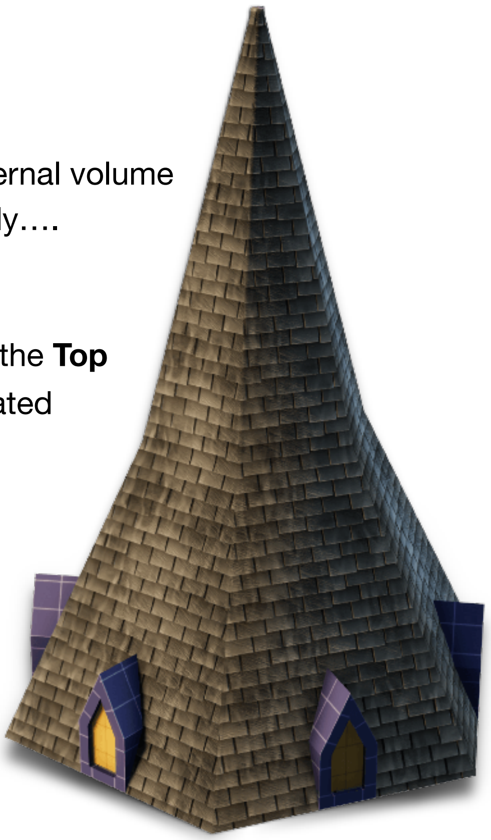
DomeShape Tension can also be set to *negative* values. Try *-0.2* for a *tent-style roof*. Use it with just 2 or 3 dome **Segs** – too many and the dome will invert.

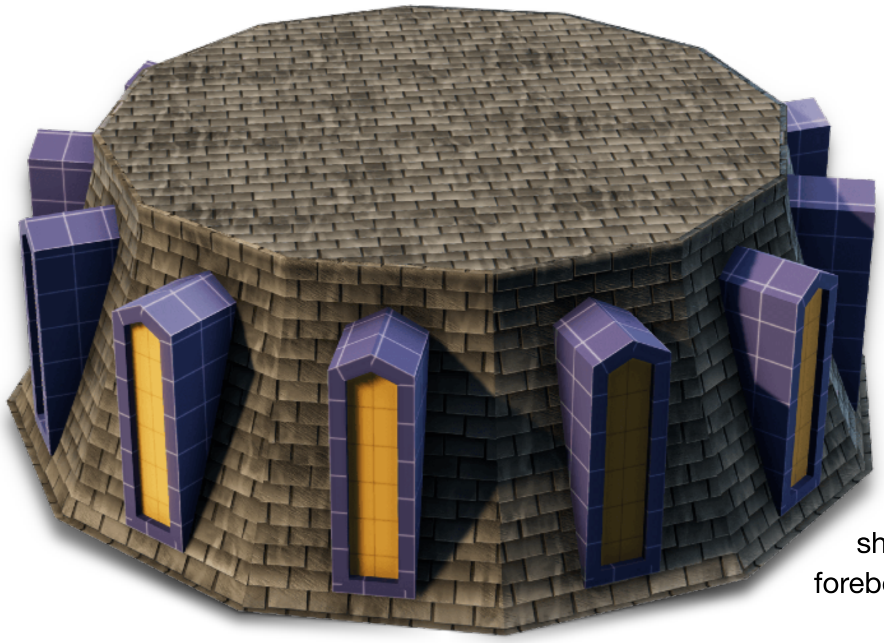
Notice there is a design issue with tent roofs. Unlike the *Mansard*, a tent roof *looses* its internal volume. If we add our dormer windows they stick out from the shrunken rooftop.

There are a few ways to regain the *garret* space.

Medieval *witch towers* use exaggerated **height.** The internal volume is maintained because the roof doesn’t reduce as quickly…. Besides, who doesn’t love an imposing tower?

Another way to regain internal roof volume is to turn on the **Top Cap** and add plenty of **Radius**. Flat roofs can be decorated with a parapet and rail to make a *widow’s walk*.





Witch tower…. Widow’s walk….

Is there a psychological reason that tent roofs seem sinister? If bulbous domes signify wealth and abundance, shrunken roofs might say the opposite: foreboding, miserly, or just lousy weather.

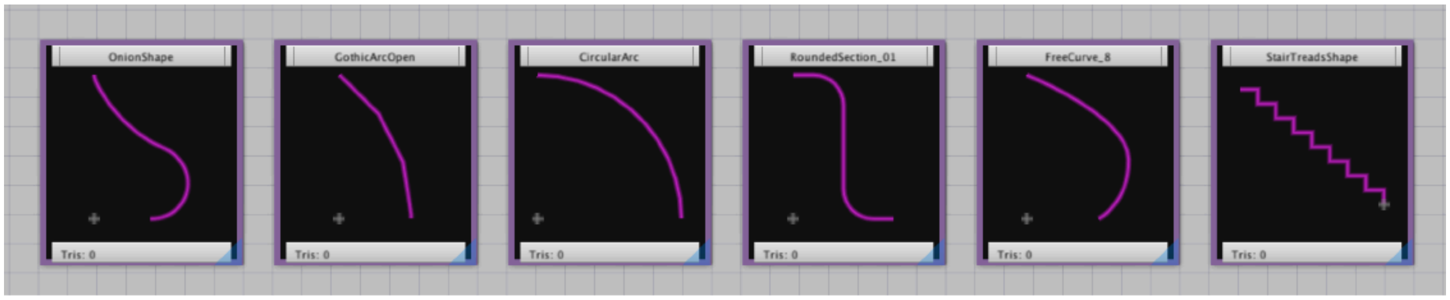
PART 2: Section and 2D Shapes

We've seen the versatility of **DomeShape** and **Lathe**. Now we'll explore the properties of other 2D Shapes to drive the s*ection* of our rooftops. Alternate 2D Shapes offer more than just a cosmetically restyled dome. In some cases they completely change the properties that can be controlled.

We won't cover every 2D shape, just nodes that are relevant to domes and rooftops. We'll also learn a few tricks to create more complex dome and spire profiles.

In this tutorial:

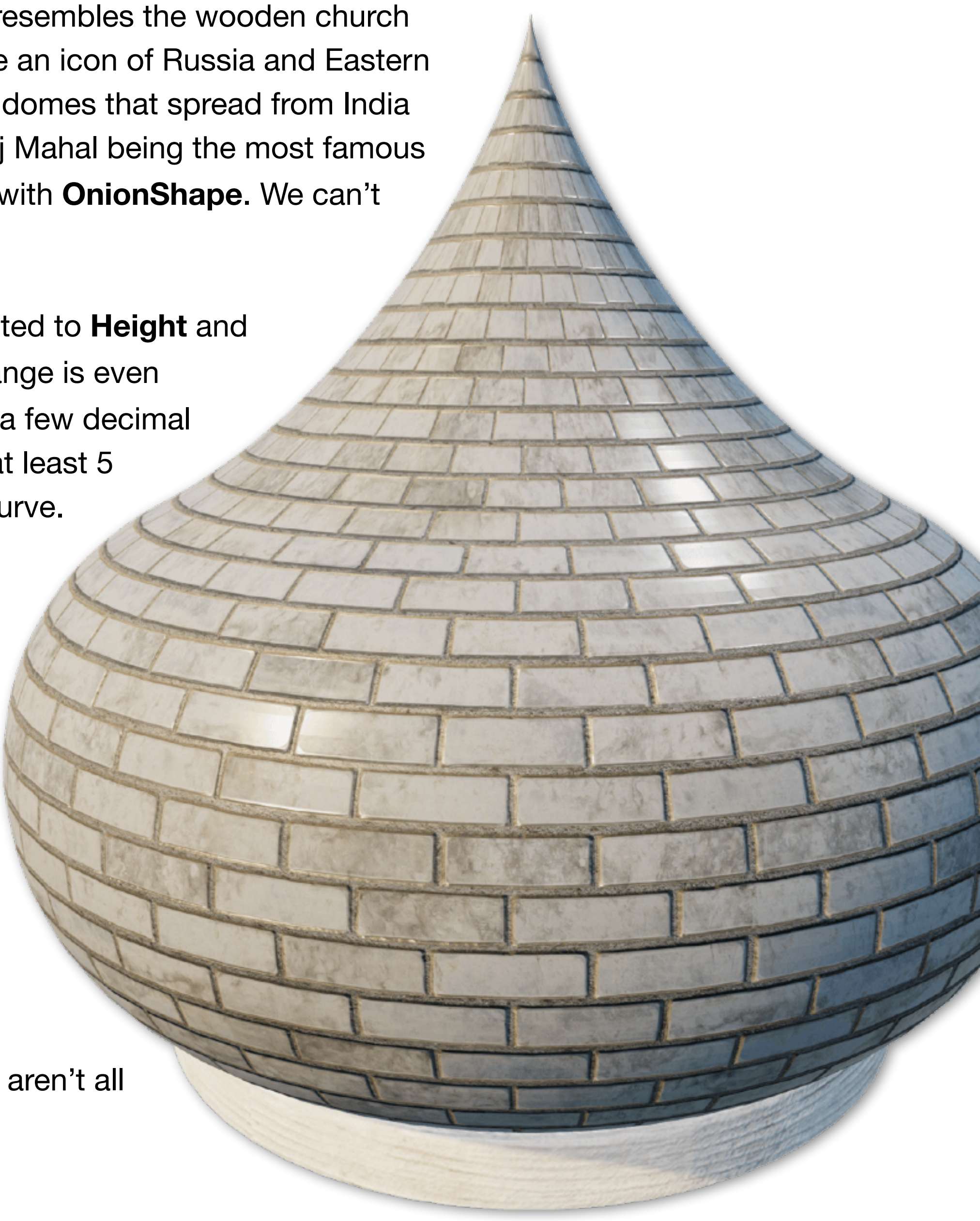
- **OnionDome**: Russia, Eastern Europe
- **GothicArcOpen**: Pointed Dome, Spike
- **CircularArc**: Radius and Angle
- **RoundedSection**: Cyma Recta and Ogee
- **FreeCurve** and **Bezier**: Bulbous Dome
- **Stacked Lathes**: Spire, Pagoda
- **Repeaters**: The Phantom Dome
- **Stepped Shapes**: aligning 2D sections



OnionShape

Connect **OnionShape** to our **Lathe**.

OnionShape resembles the wooden church domes that are an icon of Russia and Eastern Europe, but the fanciful onion domes that spread from India under the Mughal Empire – Taj Mahal being the most famous example – can't be recreated with **OnionShape**. We can't edit its base curve.



OnionShape is effectively limited to **Height** and **Width**. It's useable **Tension** range is even narrower than **DomeShape** – a few decimal points at best. We also need at least 5 **Segs** to even see the Onion curve. There's not much we can change before the shape distorts.

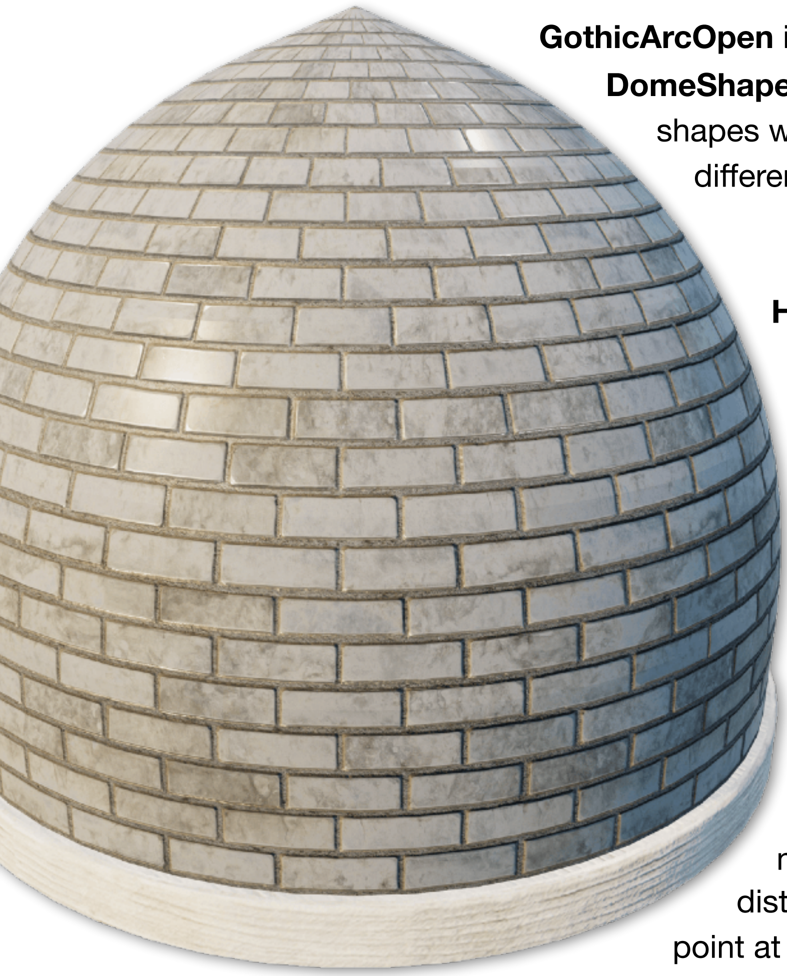
After transforming the humble **DomeShape** into an endless array of rooftop styles, the exotic **OnionShape** is a let-down.

We have the same controls as **DomeShape**, but none of the versatility…. Don't worry, they aren't all like this.

GothicArcOpen: Pointed Dome, Spike

Connect **GothicArcOpen** to our **Lathe**.

Pointed domes are everywhere in Islamic architecture, originating from the earliest Mesopotamian burial tombs – along with round domes and cones. Look for pointed domes in gateway arches and prayer *alcoves* called *mihrab*.



GothicArcOpen is also not as versatile as **DomeShape**, but **GothicArcOpen** can create shapes with a few interesting design differences.

Height and **Width** are familiar, but **StraightTop** changes the top of the dome into a sharp point like a pencil tip or missile, while **Spring** extends the legs of the dome vertically.

GothicArcOpen can create a cone with a long shaft and only a narrow curving area – useful for *spikes* and *lances* that need to maintain their radius over a distance and then reduce to a sharp point at the tip.

CircularArc: Radius and Angle

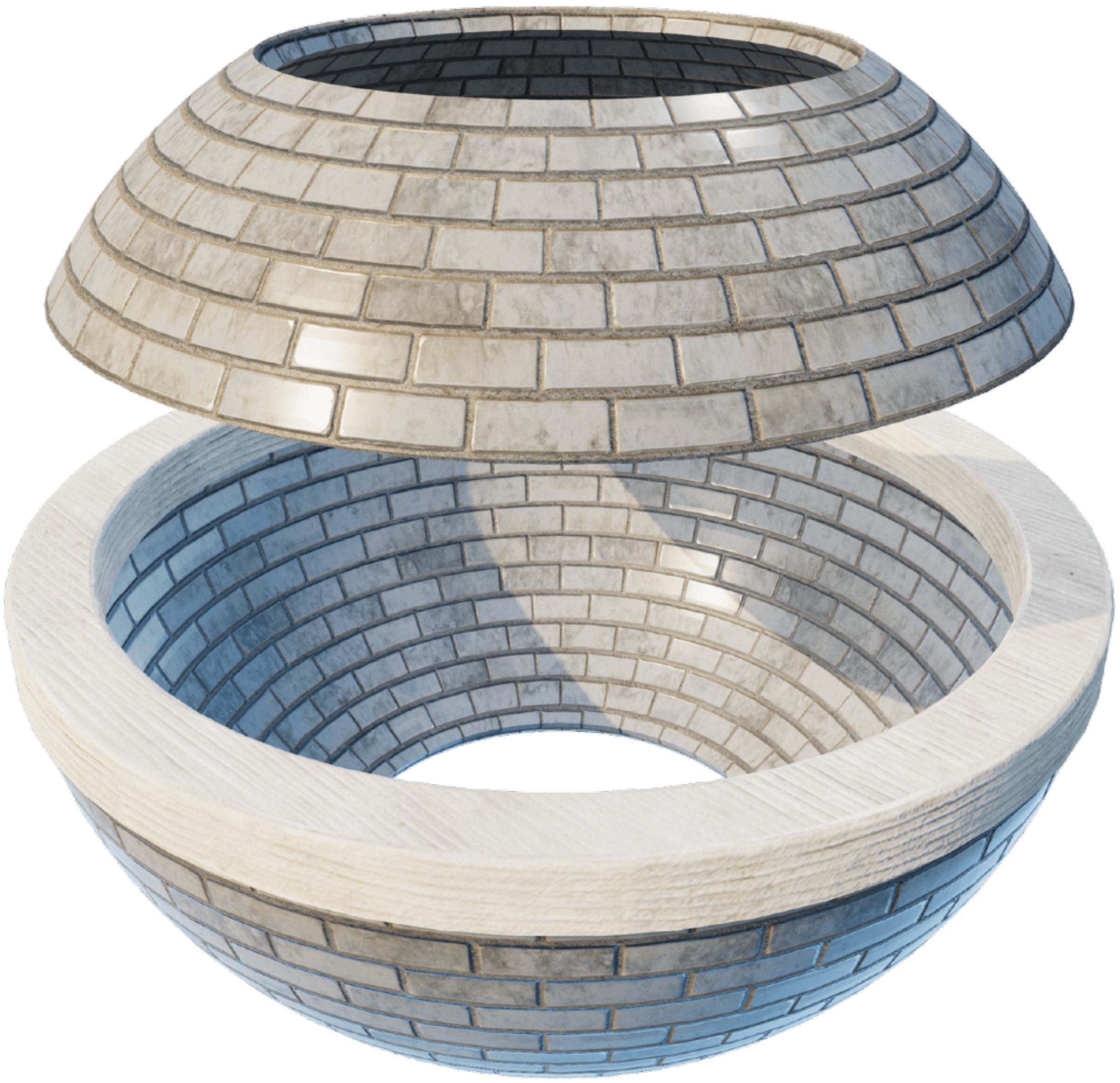
CircularArc is another 2D shape with interesting properties.

This time, **Height** and **Width** are replaced with a single control: **Radius**. The arc is always circular, but we get a **Start Angle** and **End Angle** for the dome.

This is our first dome shape that wraps *below* the roofline and *beyond zero on the x-axis*. This allows us to create bubbles, torus, cups, curbs, transitions, and partial shells.

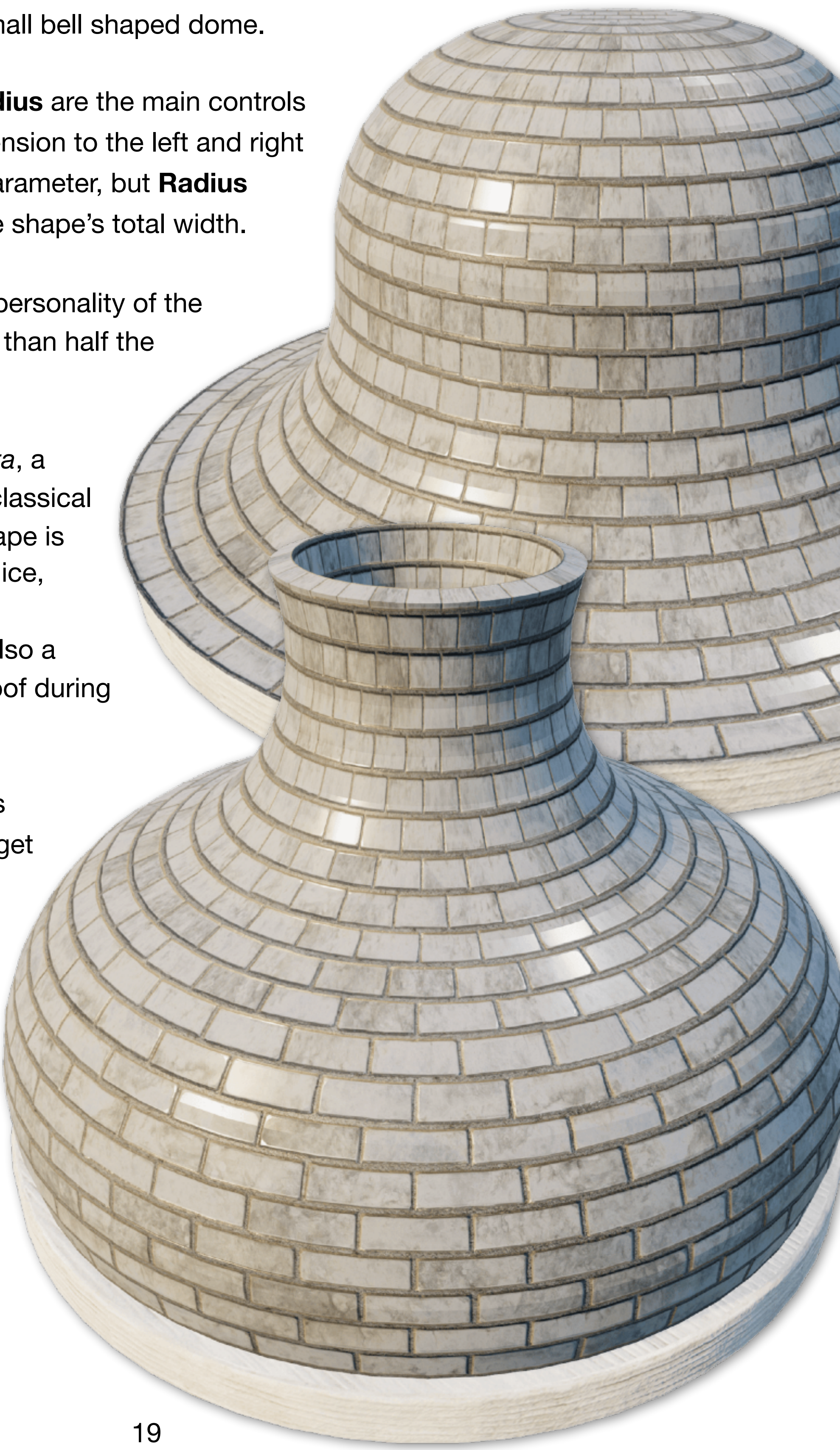
We can create runtime spheres that raise and lower like a curtain. Awnings that “roll” down.

Combined with **Lathe SweepAngle** we can create a latitude and longitude slice of a sphere using degree and radius.



RoundedSection: Cyma Recta and *Ogee*

Connect **RoundedSection** to our **Lathe** and we should see a small bell shaped dome.



Height and **Radius** are the main controls here. **Lip** adds a horizontal extension to the left and right sides. We don't have a width parameter, but **Radius** added to **Lip** is equal to half the shape's total width.

Radius has a big effect on the personality of the shape, but should not be larger than half the **Height** or the shape fails badly.

RoundedSection is *Cyma Recta*, a traditional molding curve from classical architecture. As an arch this shape is called Ogee and is found in Venice, India, and flamboyant gothic architecture. The s-curve was also a fancier profile of the *Mansard* roof during the French *Second Empire*.

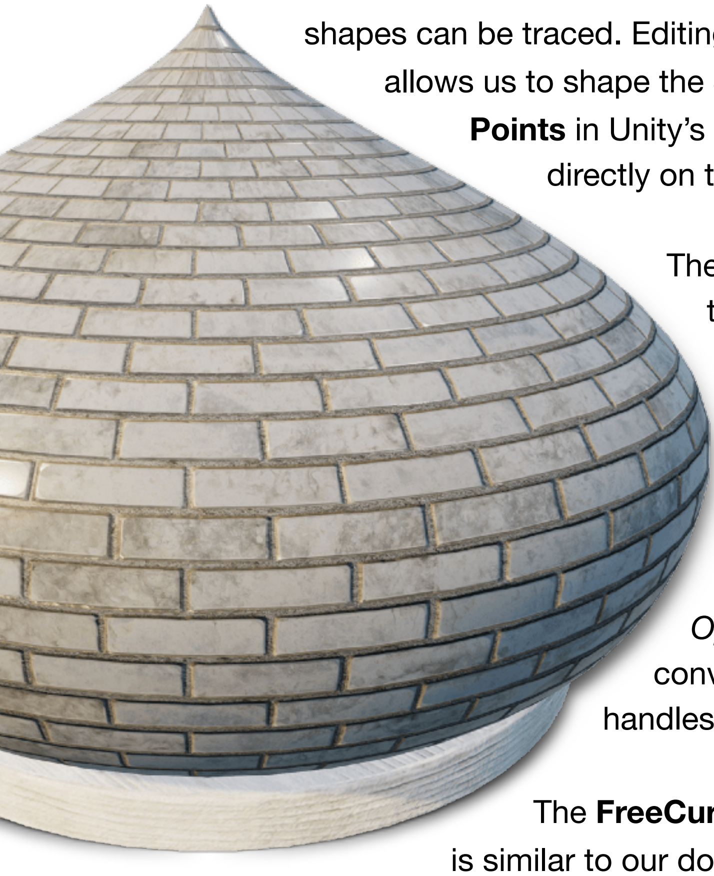
Adjust **RoundedSection** node's **Rot_Z** to -90° and set **FlipX** to get a pointed *ogee* dome. **Lip** now extends from the top and bottom of the shape.

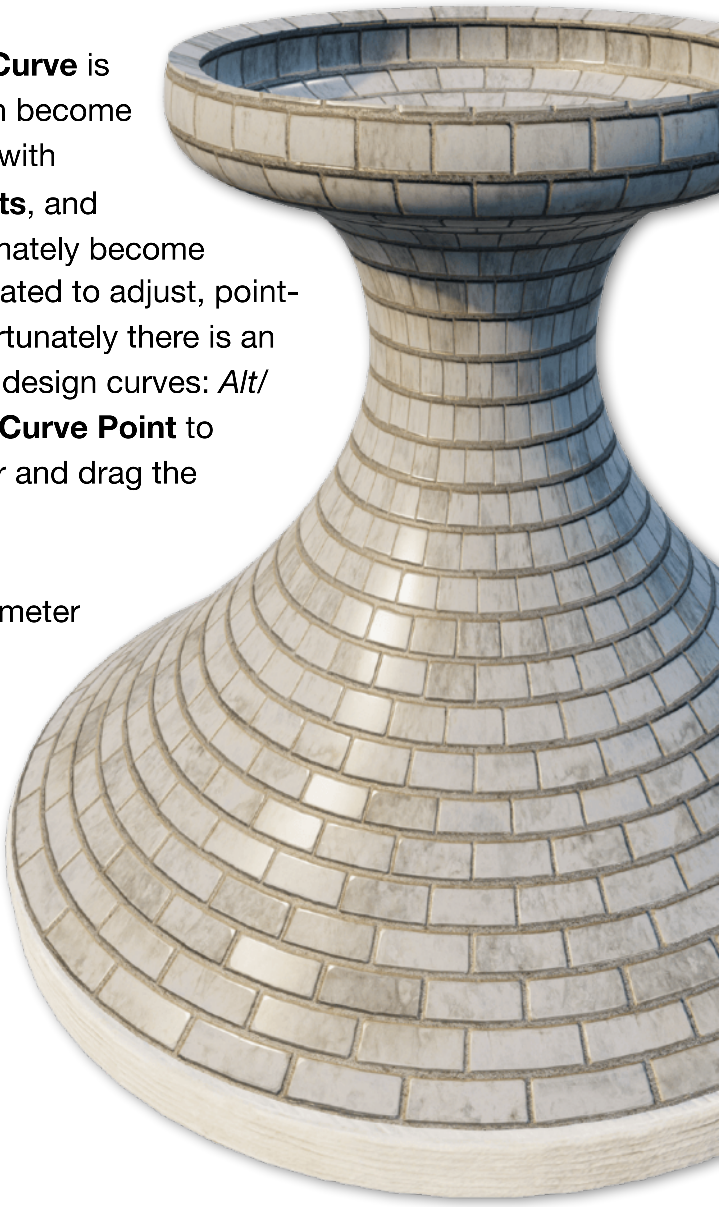
Other interesting domes are possible by tweaking the node's **Rot_Z** and adjusting the **Trans_X** and **Trans_Y**. Organic mud huts or a kiln shape with a central chimney.

FreeCurve and Bezier: Bulbous Dome

At some point we'll want to use **FreeCurve** to create our own dome and spire profiles. Connect **FreeCurve** to **Lathe**, and mark the dome's *profile* with **Curve Points**. After working with our other dome shapes this should be almost intuitive.

Each **Curve Point** becomes a vertex on our 3D mesh, so very precise profile shapes can be traced. Editing the **FreeCurve** section while attached to Lathe allows us to shape the 3D dome in place, either by dragging **Curve Points** in Unity's Scene view, or by editing the **X** and **Y** parameters directly on the node.



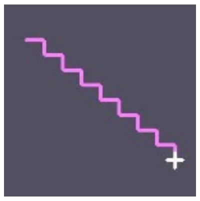


The limit of **FreeCurve** is that nodes can become over-loaded with **Curve Points**, and shapes ultimately become too complicated to adjust, point-by-point. Fortunately there is an easier way to design curves: *Alt/Option+Click* a **Curve Point** to convert it to bezier and drag the handles.

The **FreeCurve Segs** parameter is similar to our dome **Segs**, controlling the smoothness of each bezier curve.

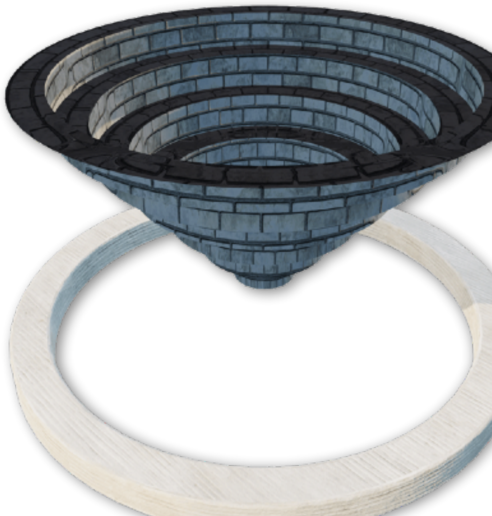
Curve Points and bezier handles cannot be controlled through parameters (yet).

While a complicated **FreeCurve** shape is not always the most flexible solution for roof domes, a lean **FreeCurve** node with two or three bezier **Curve Points** can create custom onion or bulbous dome shapes.



Stepped Shapes: aligning 2D sections

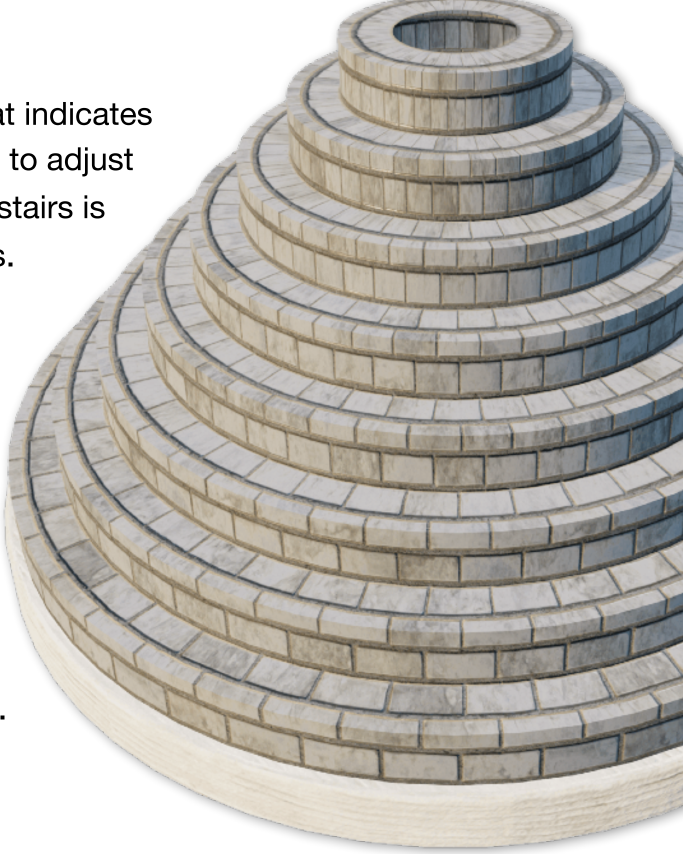
Our 2D shapes have all started at top left and sloped down to the right. As long as we use positive numbers as *section* **Height** and **Width**, we get a dome, cone or pyramid. This has made our rooftops work without needing any transform adjustments.



But let's take a look at the various stairs and treads shapes – some are *closed* and some are *open* but this doesn't matter. They all appear to start top left and slope down to the right, however this is deceptive as we'll see when we connect a stair shape to the **Lathe**.

Notice our 2D shape is not just flipped around, instead the dome's *Section* is on the wrong side of the lathe, and if we use a 2D shape we might see the mesh inverted, shadow where it should be bright, even if we have **Backfaces** *on*.

If we adjust the **Lathe Radius** it reveals the problem: we have a stepped dome but our AX mesh is not "*failing gracefully*". The 2D stair shape sits below *zero* on its **Trans_X**.



If we look at the node's icon we see a white cross that indicates where *the section* will be swept by the *plan*. We need to adjust the 2D stair node's **Trans_X** until the left edge of the stairs is at the center of our **Lathe**, marked by the white cross.

In parametric terms, we have learned another important relationship. As long as we know the width of our *Section*, we can shift its **Trans_X** in either direction – the full width or a ratio. This gives us control over where the *Section* profile sits in relation to the *Plan* shape.

With **Trans_X**, a rooftop can be shifted to the *inside* or *outside* of a wall plan, or straddle it to make eaves. Remember this when we work with *plans* in Part 3.

Stacked Lathes: Finial, Spire, Pagoda

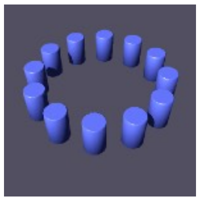
A dome tower can be created with stacked and overlapping **Lathe** nodes, and as many individual 2D *sections* as necessary.

Each **Lathe** node's **Radius** and **Trans_Y** can be parametrically related to the next dome's **width** and **height**, or their parameters can be related as a *ratio* of the whole tower.

Stacked Lathes can change **Segs** with each tier, transitioning from round to square or more complex faceted geometry.

A **FloorRepeater** adds vertical repetition to exaggerate the spire's height and provoke interest. Each tier might represent a visual mnemonic – a parable told through shape and numeric repetition. The vertical transitions might describe a heroic journey of spiritual ascension, represent the elements, or add a stylish point of focus to an otherwise utilitarian rooftop.

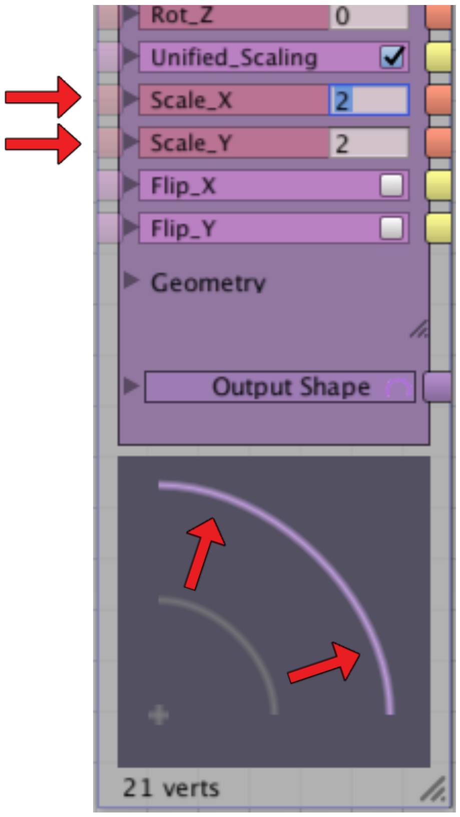
Repetition, tiers, and symbolism give our *spire* structure and purpose.



The Phantom Dome

Our final dome is more an abstract concept than an actual dome. It's a visual trick where a central dome is surrounded by identical smaller domes set at lower elevation – a third tier of still smaller domes usually rings the perimeter. The effect is dazzling, and adds energy to the main dome with vertical repetition.

But it also creates an optical illusion. The matching domes creates the unifying impression of one big dome that contains the volume of all the other domes. The effect is amplified when the *phantom dome* echoes the height and width ratio of the dome family.



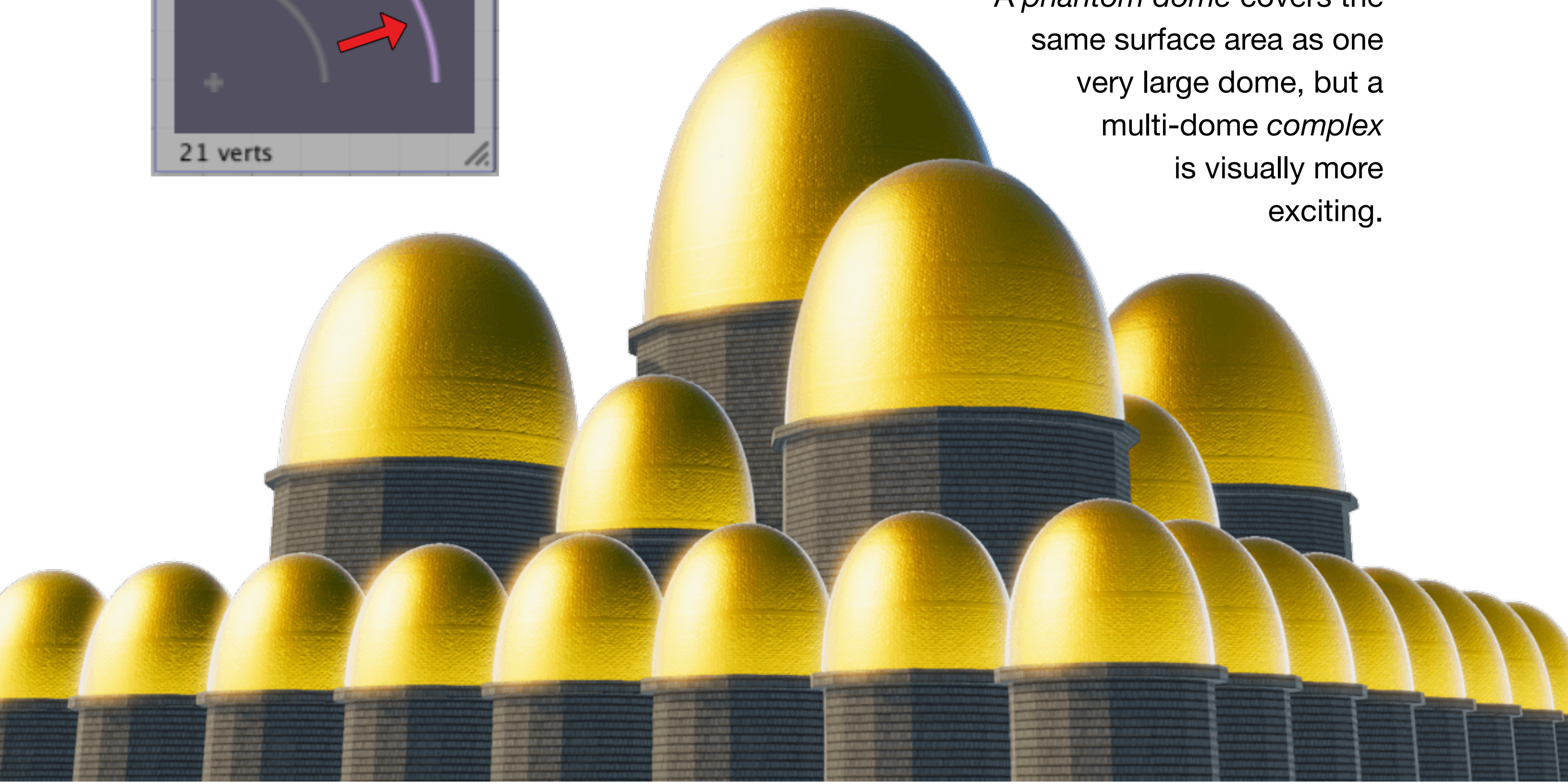
Rot_Z 0
Unified_Scaling ✓
Scale_X 2
Scale_Y 2
Flip_X
Flip_Y
Geometry
Output Shape
21 verts

Split the **DomeShape** through one or more **Offsetter** nodes. Scale with Transform **Scale_X**, and connect each **Offsetter** to a new **Lathe**. Unite the **Lathes** with a shared **Material** node and replicate them with **RadialRepeater**, **LineRepeater**, **GridRepeater**, or **PlanRepeater** to form each tier.

Daughter domes might be arranged radially, in a cross, or at the corners. Granddaughters can radiate from the daughters, or extend from the main dome, or trace the outline of a larger perimeter shape.

Tiny domes often accent a building as *chhatris*, *cupolas*, and *minarets*, or create diminishing tiers like a *Chinese pagoda* – in many *Hindu* temples a smaller copy of the entire dome tower merges into the faces of the main dome like a scaling fractal.

A *phantom dome* covers the same surface area as one very large dome, but a multi-dome *complex* is visually more exciting.



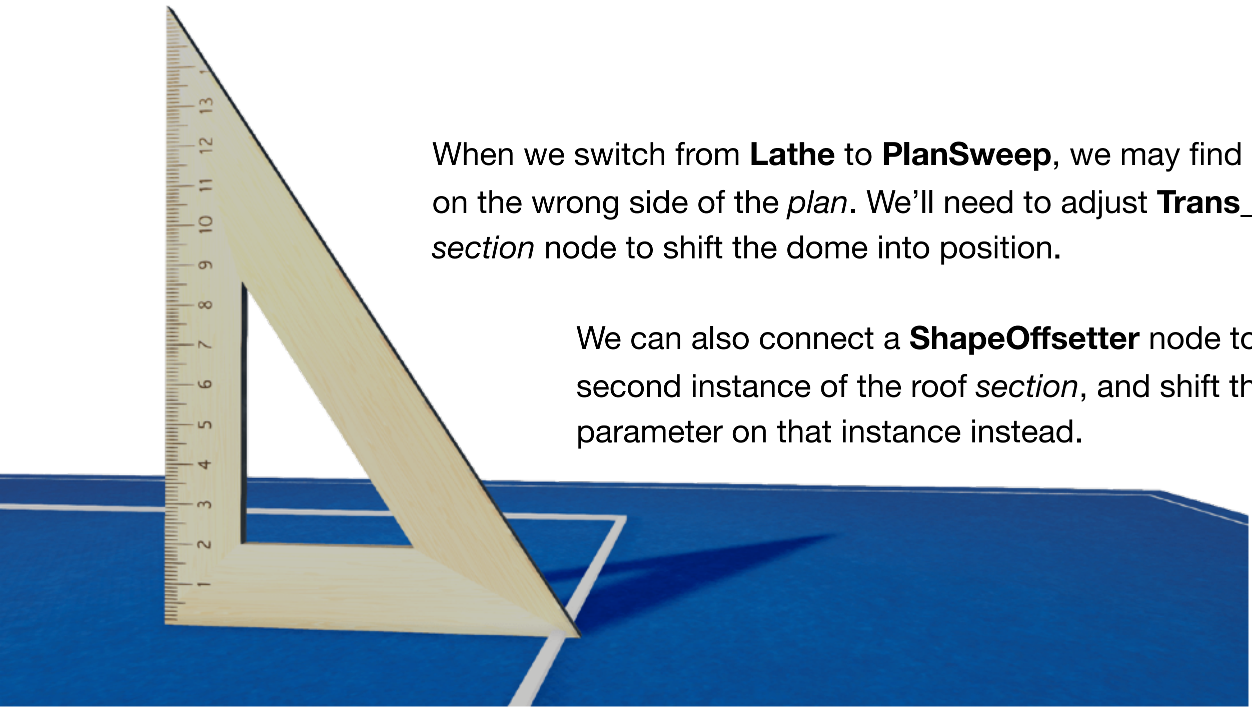
PART 3: Plan and Rooftop Construction

Now that we can style our roof *section*, we'll swap **Lathe** with a **PlanSweep** and focus on creating rooftop *plans*.

We'll build the 3 traditional rooftops: *hip and valley*, *skillion*, and *gable*.
We'll tackle irregular *plans* and combination roofs using stacked **PlanSweep** nodes.
Finally, we'll create multi-plans for *cross-joint gables*, *hip and valley* corners, and *groin vaults*.

In this tutorial:

- **Hip and Valley**
- **Skillion**, **Mono-Pitch, Clerestory, Truss Frame**
- **Gable**, **Gambrel**, **Barrel**, **Butterfly**, **Sawtooth**, **Split-Skillion**
- **2D Gable**, **FreeCurve Barrel**
- **Oculus Gable**
- **Combination**, **Plantation**, **Dutch Gable**, **Jerkinhead**, **Saltbox**
- **Cross-Joint Gable**, **Hip and Valley Corners**
- **Fascia and Gable Trim**
- **Groin Vault, Ribbed Vault**

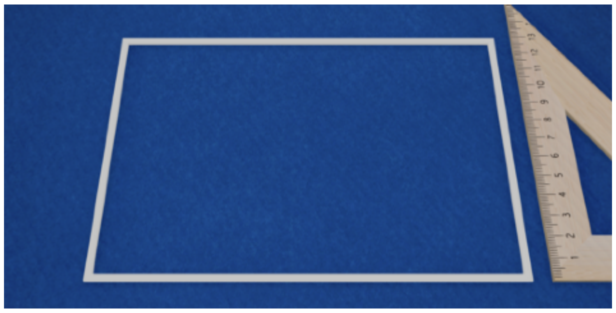


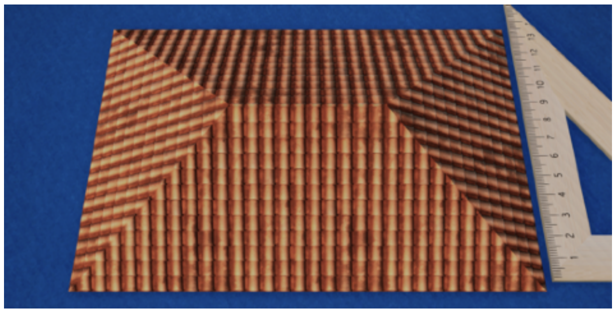
When we switch from **Lathe** to **PlanSweep**, we may find our roof is on the wrong side of the *plan*. We'll need to adjust **Trans_X** on the *section* node to shift the dome into position.

We can also connect a **ShapeOffsetter** node to create a second instance of the roof *section*, and shift the **Trans_X** parameter on that instance instead.

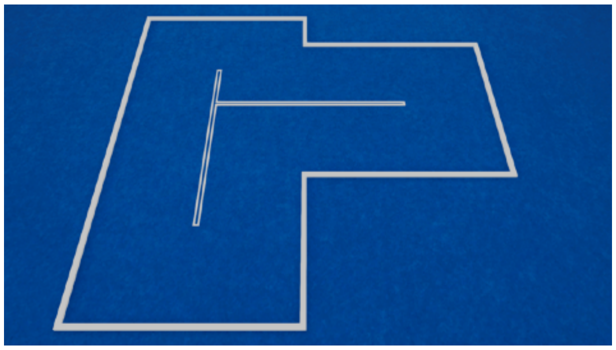
Hip and Valley

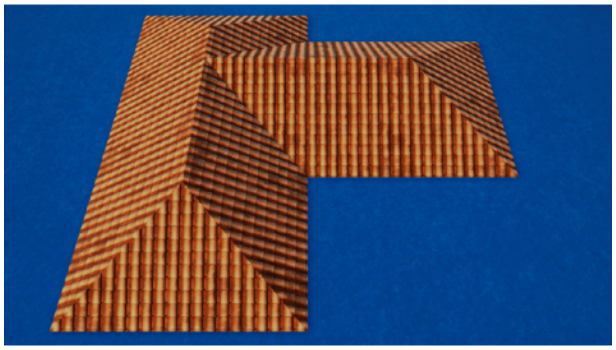
A rectangular *hip* roof needs a dome **width** that is *half the length of the shortest span*.





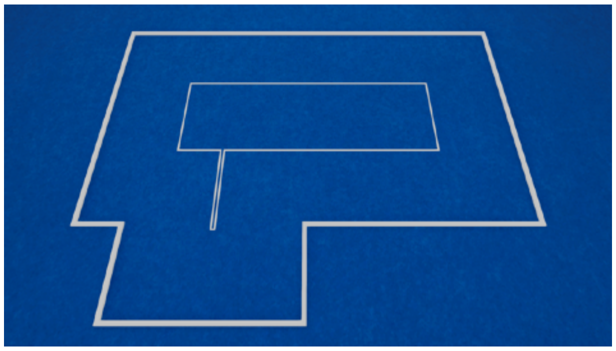
Measure the shortest span on the rectangle, divide by 2, and that becomes the dome width for our roof. We'll call this the ***shortest span rule***.

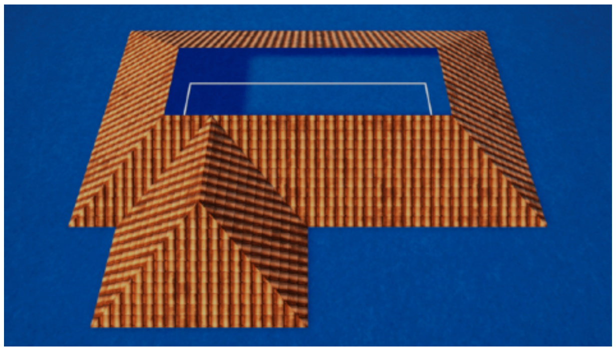




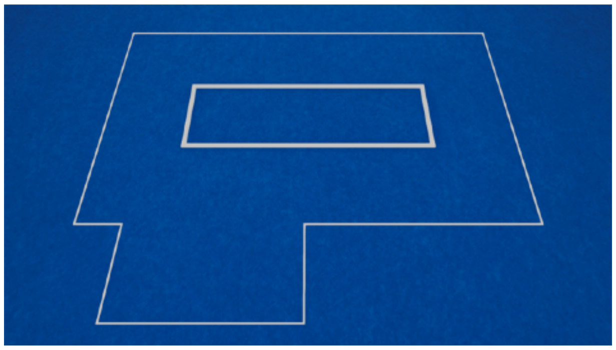
A regular *hip and valley* uses one roof span throughout. As long as the walls are a consistent distance from each other, the entire *plan* can be covered in one roof.

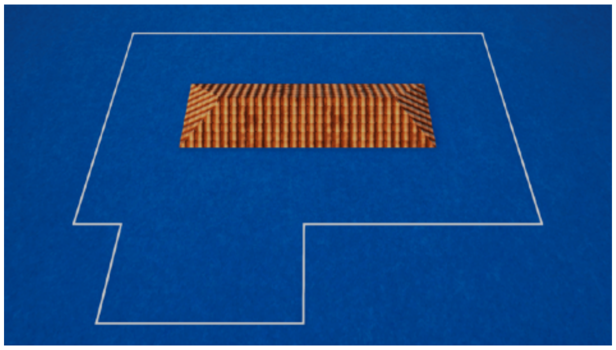
But as *hip and valley plans* become complex and irregular, we can't rely on a single **PlanSweep** to cover everything. it will be necessary to build the rooftop as two or more stacked **PlanSweep** shapes.





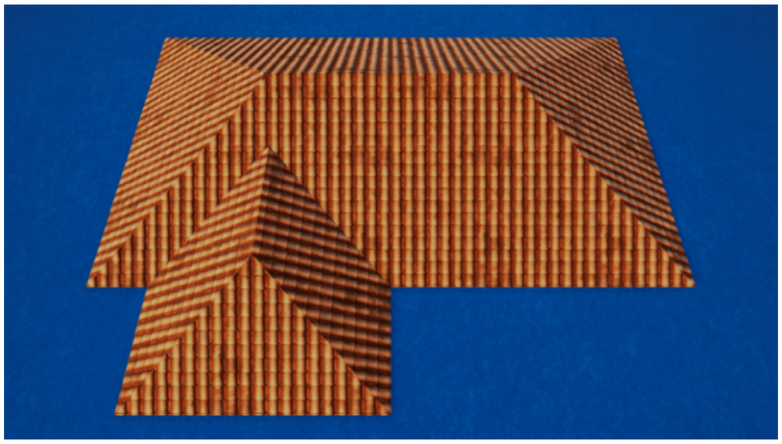
Our first roof will use the *shortest span rule* to find its dome width. This roof will cover the narrowest span but leave a gap over larger areas.





We'll create a second roof *plan* using a **ShapeOffsetter** of the original *plan* which subtracts the first roof's *width*. *Offset*, and the first dome's *width* can be related.

We'll use the *shortest span rule* again to find the second roof's *width*, and make a new 2D *section* to fit. When scaling a smaller roof *section*, remember to preserve the *height to width ratio* of the original dome.

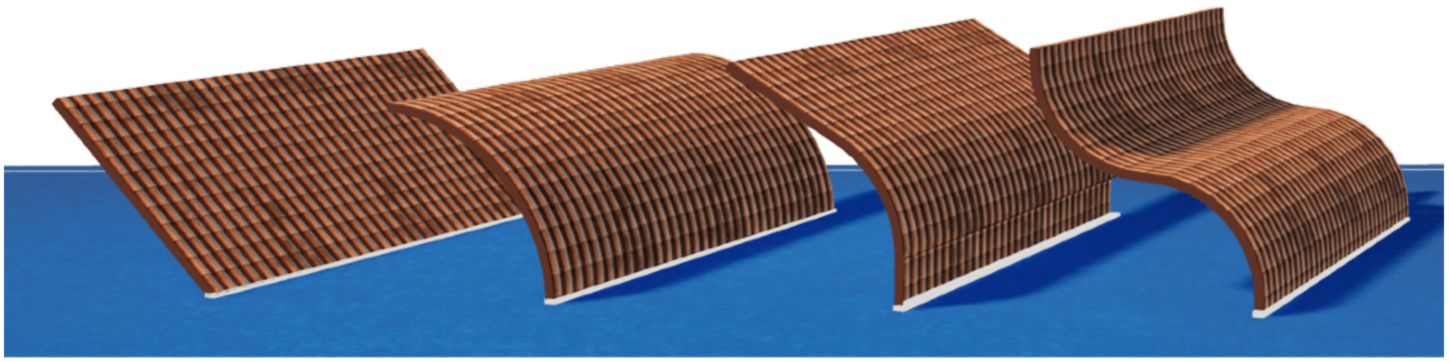


Set the second roof's **Trans_Y** to the *height* of the first roof.

We can repeat the process until the entire roof is filled in with *hip and valley* layers.

Skillion, Mono-Pitch

A *mono-pitch* roof or *skillion* – sometimes called a *shed* roof – is actually easier to visualize than to build in **Archimatix**.

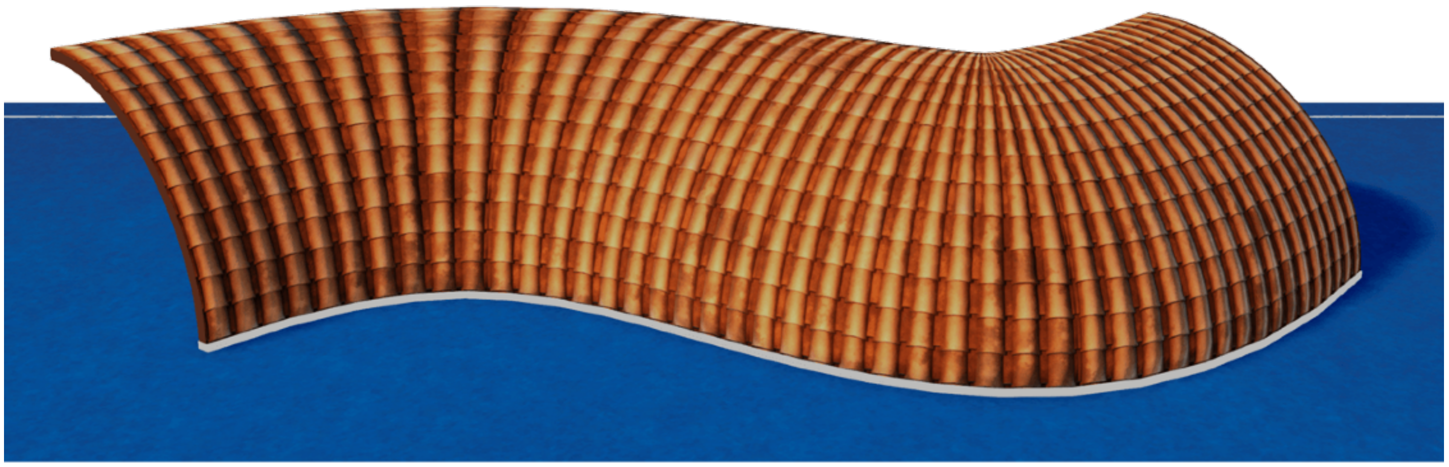


It ought to be simple, we just need a *plan* shape with a straight line. Correction – we need a *plan* shape with two points: *start* and *end*. A spline model *section* follows its *plan* by sweeping from point to point in a specific order. That may seem like an overwrought description but this is how **Archimatix** understands our simple *skillion*. Moving forward we should consider all rooftop *plans* as a series of connected points, rather than closed geometric shapes.

Now that we understand how **Archimatix** defines a straight line, we have options for how to create one for our *skillion plan*.

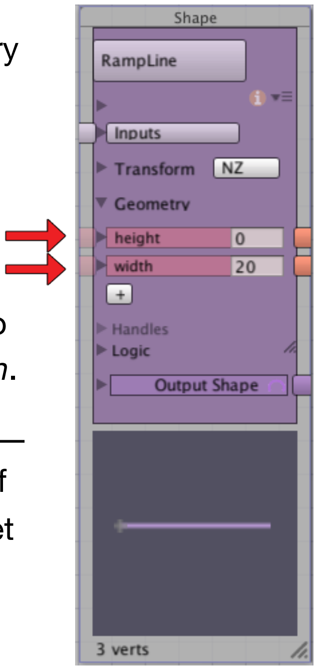
FreeCurve allows us to make the *start* and *end* points directly in the X-Z plane where we want the roof to appear. The advantage is that our *plan* is very straightforward. It's easy to "fit" the *skillion* to a wall simply by dragging the **Curve Points** in the *scene* view.

We can also use the **FreeCurve** bezier tools to create a *skillion* dome that curves.



One downside to **FreeCurve**, we can't expose **Curve Points** or relate them to another parameter so it's difficult to manipulate a **FreeCurve** line as a parametric model. Transform *scale* and *Flip_X* are also ignored by **FreeCurve**.

For these reasons, **RampLine** is usually a better node for parametric control, but it also has some quirks. The **Trans_X** and **Trans_Y** of **RampLine** set the node's *start* point. The **height** and **width** parameters position the *end* point. **RampLine** is the only 2-point shape that comes with **Archimatix**, and by its name we can guess it's intended for use as a *section*, not a *plan* – we could have used **RampLine** to build our pitched roof but **DomeShape** is more versatile.

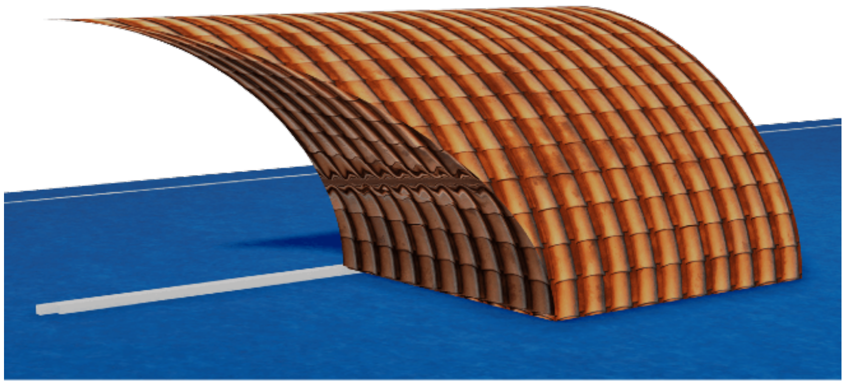


height 0
width 20

With **RampLine** as our *plan* we have more controls than necessary so let's simplify. The only parameter we need from this node is **Width**. Set **RampLine**'s *height* to *zero* and ignore it to avoid problems. *

Expose **Rot_Z** and **Trans_Y** on the **PlanSweep**. We now have a *skillion* roof that can parametrically extend its length, and rotate to any angle. The skillion's "dome" shape is controlled by the *section*.

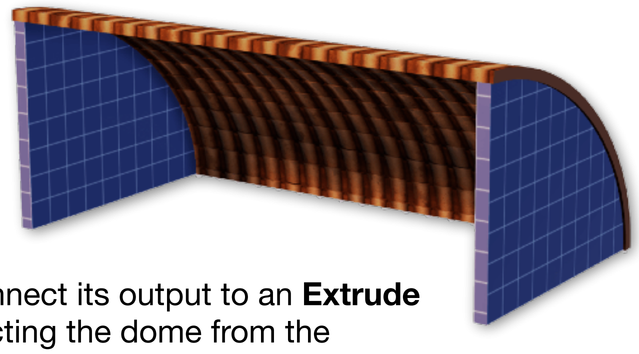
* There is a quirk in **RampLine** when it's used in the *plan* shape. If we *zero* the **width** parameter, and expose **height** instead, we get a *skillion* with one amazing difference, the dome's *section* is added or subtracted at the *end point* as the shape attempts to *close* and return to *zero* on the X-axis. We get a *skillion* that is open at one end, with a quarter-end cap over the *other*.



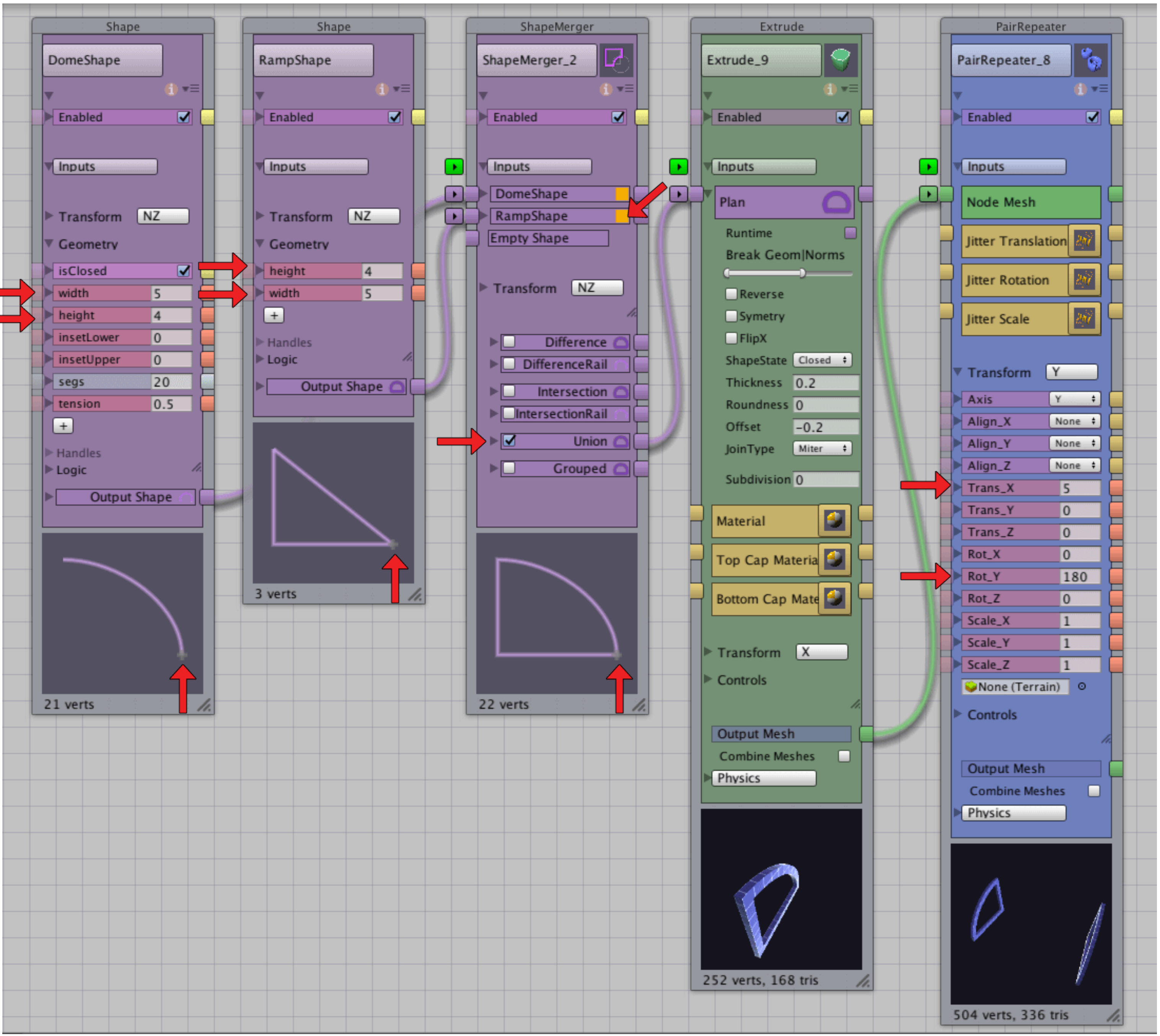
Try a *negative* value in the *height* parameter and toggle *Flip_X* to see the variations. Also shift Trans_X on the dome. The subtracted end cap is unique, but expect UV problems since we're exploiting a bug in **RampLine**.

Clerestory

Let's cap our *skillion* ends with **Extrude** nodes that conform to the dome to create a *clerestory*.



1. Add a **RampShape** node to the graph, and align it with the dome section.
2. Match the dome's *width* and *height* to **RampShape** node's *width* and *height.*
3. Connect both nodes to a **ShapeMerger** and connect its output to an **Extrude** node. (If our dome is a *tent* roof, we'll be subtracting the dome from the **RampShape**, and our **ShapeMerger** settings will be different.)
4. Output the **Extrude** node to a 3D **PairRepeater**, and adjust *Trans_X* and *Rot_Y*.
5. Lastly, adjust the *Separation* to match the width of the skillion.



Shape: DomeShape
Enabled ✓
Inputs
Transform NZ
Geometry
isClosed ✓
width 5
height 4
insetLower 0
insetUpper 0
segs 20
tension 0.5
Handles
Logic
Output Shape
21 verts

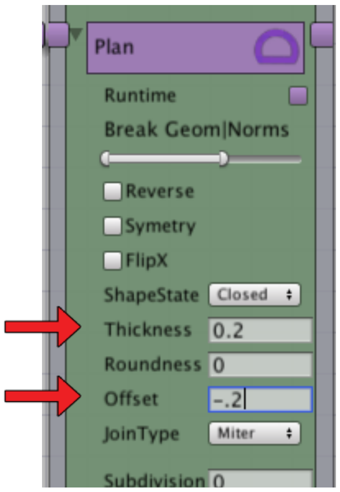
Shape: RampShape
Enabled ✓
Inputs
Transform NZ
Geometry
height 4
width 5
Handles
Logic
Output Shape
3 verts

ShapeMerger: ShapeMerger_2
Enabled ✓
Inputs
DomeShape
RampShape
Empty Shape
Transform NZ
☐ Difference
☐ DifferenceRail
☐ Intersection
☐ IntersectionRail
✓ Union
☐ Grouped
22 verts

Extrude: Extrude_9
Enabled ✓
Inputs
Plan
Runtime
Break Geom|Norms
☐ Reverse
☐ Symetry
☐ FlipX
ShapeState Closed
Thickness 0.2
Roundness 0
Offset -0.2
JoinType Miter
Subdivision 0
Material
Top Cap Materia
Bottom Cap Mate
Transform X
Controls
Output Mesh
Combine Meshes ☐
Physics
252 verts, 168 tris

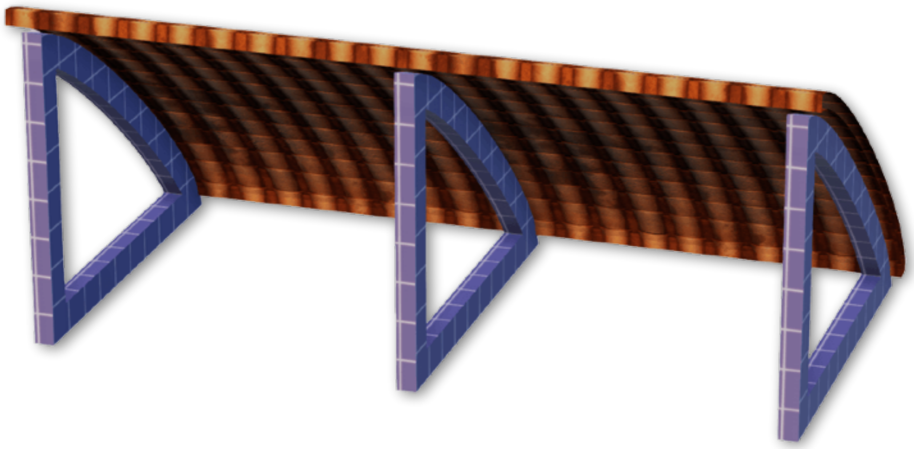
PairRepeater: PairRepeater_8
Enabled ✓
Inputs
Node Mesh
Jitter Translation
Jitter Rotation
Jitter Scale
Transform Y
Axis Y
Align_X None
Align_Y None
Align_Z None
Trans_X 5
Trans_Y 0
Trans_Z 0
Rot_X 0
Rot_Y 180
Rot_Z 0
Scale_X 1
Scale_Y 1
Scale_Z 1
None (Terrain)
Controls
Output Mesh
Combine Meshes ☐
Physics
504 verts, 336 tris

Truss Frame

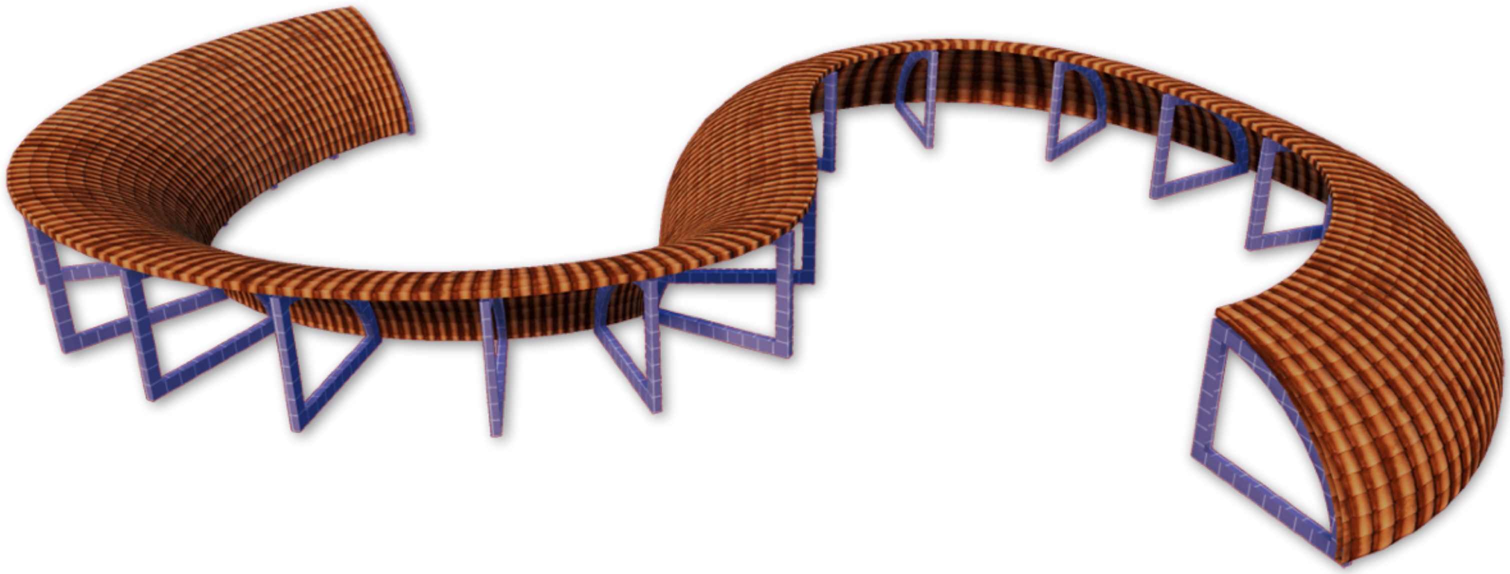


ShapeState: Closed
Thickness: 0.2
Roundness: 0
Offset: -.2
JoinType: Miter
Subdivision: 0

Add *Thickness* and subtract *Offset* from the *Plan Input* on the **Extrude** node to create a simple *truss* frame that always fits under the dome.



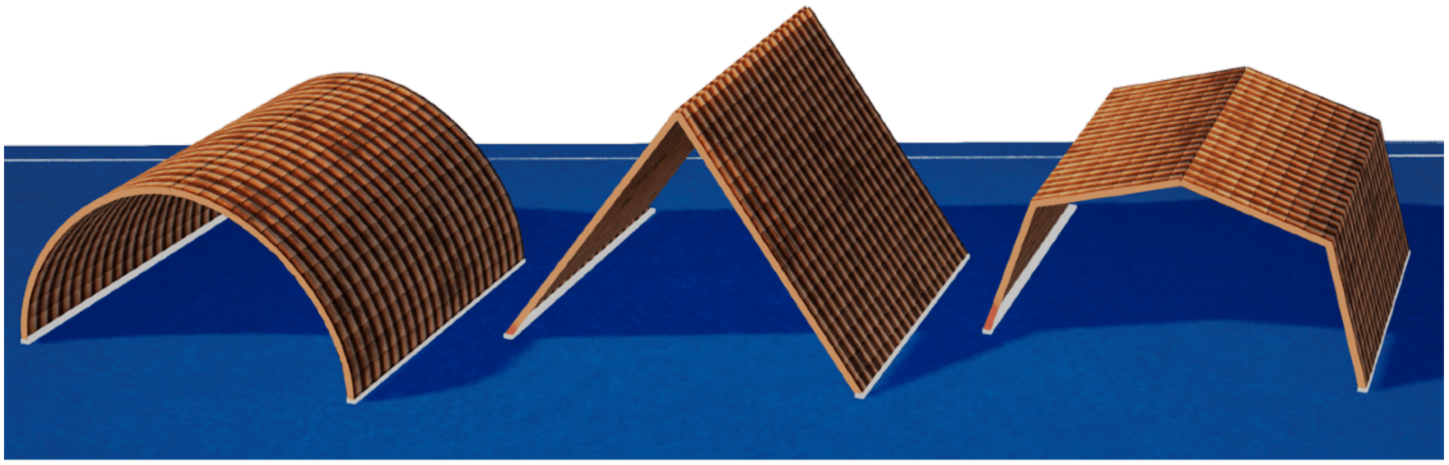
The *truss* can be replicated along the *plan* with a 3D **PlanRepeater.**



We may need to shift the *section* node's **Trans_X** depending on which side of the *plan* we use to align the *skillion* and truss.

Gable, Gambrel, Barrel

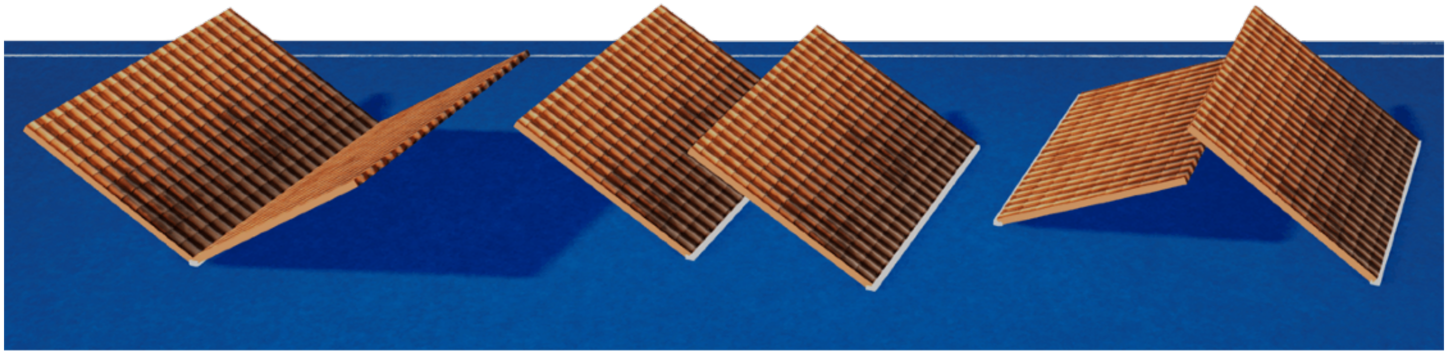
Now that we have a working *skillion*, we'll explore 3 methods to make *barrel*, *gable*, and gambrel roofs like the ones seen below. Each method has limits and advantages.



The easiest method is to treat a *gable* as two separate *skillion* roofs. The sides are not connected, but they touch at the peak just like our *hip roof*. *Gables* and *hips* can share dome *sections*, it's just a matter of how their *plans* are arranged.

Butterfly, Sawtooth, Split-Skillion

With two *skillions* we can turn our *gable* into a *butterfly*, *sawtooth*, or a *split-skillion* that uses two different sections. We can also match our *oculus* dome by simply leaving a gap between the two roofs.

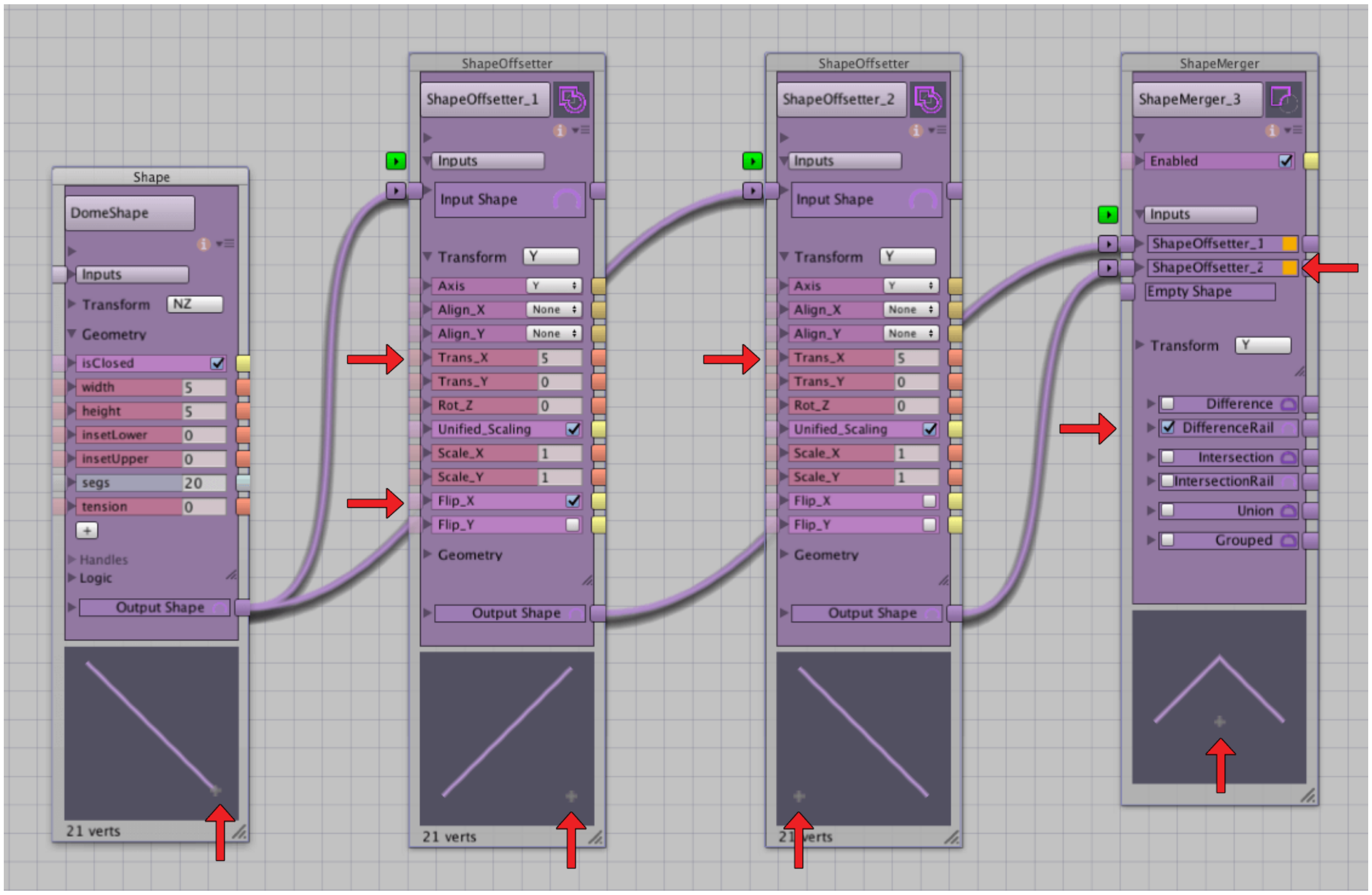


It's natural to want to work with a *gable* as a single object. The two *skillions* can be parametrically related, or combined in a **Grouper**, but consider it's easier to simply rotate and position two *skillions* than to set up complicated parametric spaghetti.

Separate roofs offer more creative options.

2D Gable, FreeCurve Barrel

The second method creates a 2D *gable section*. It uses **ShapeOffsetter** to duplicate and flip our dome node, and **ShapeMerger** to combine them into one *section*. We still make adjustments to our original node, which is mirrored to both sides of the *gable*.



Shape
DomeShape
Inputs
Transform NZ
Geometry
isClosed ✓
width 5
height 5
insetLower 0
insetUpper 0
segs 20
tension 0
Handles
Logic
Output Shape
21 verts

ShapeOffsetter
ShapeOffsetter_1
Inputs
Input Shape
Transform Y
Axis Y
Align_X None
Align_Y None
Trans_X 5
Trans_Y 0
Rot_Z 0
Unified_Scaling ✓
Scale_X 1
Scale_Y 1
Flip_X ✓
Flip_Y
Geometry
Output Shape
21 verts

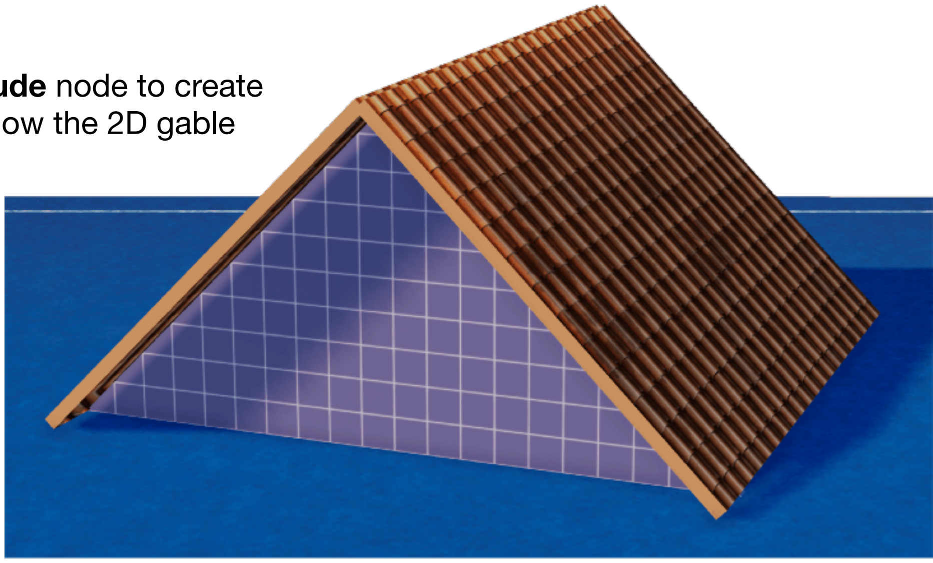
ShapeOffsetter
ShapeOffsetter_2
Inputs
Input Shape
Transform Y
Axis Y
Align_X None
Align_Y None
Trans_X 5
Trans_Y 0
Rot_Z 0
Unified_Scaling ✓
Scale_X 1
Scale_Y 1
Flip_X
Flip_Y
Geometry
Output Shape
21 verts

ShapeMerger
ShapeMerger_3
Enabled ✓
Inputs
ShapeOffsetter_1
ShapeOffsetter_2
Empty Shape
Transform Y
Difference
DifferenceRail ✓
Intersection
IntersectionRail
Union
Grouped

Following the graph from left to right:

1. Our **DomeShape** connects to a **ShapeOffsetter.** Under Translate, switch **Flip_X** *on* to create the mirror of our dome. Adjust **Trans_X** to shift the white cross to the center of the gable.
2. Connect **DomeShape** to a second **ShapeOffsetter**, and again adjust **Trans_X** to shift the white cross to the center of the gable
3. Connect both **ShapeOffsetter** nodes to a **ShapeMerger.** Their order is important. Make sure both *inputs* on the **ShapeMerger** are set to *solid*.
4. If any of the shapes have closed, check all *inputs* and *outputs* in the node chains until all shapes are set to *open*.
5. The **ShapeMerger** *DifferenceRail* will output a contiguous 2D *gable* shape.

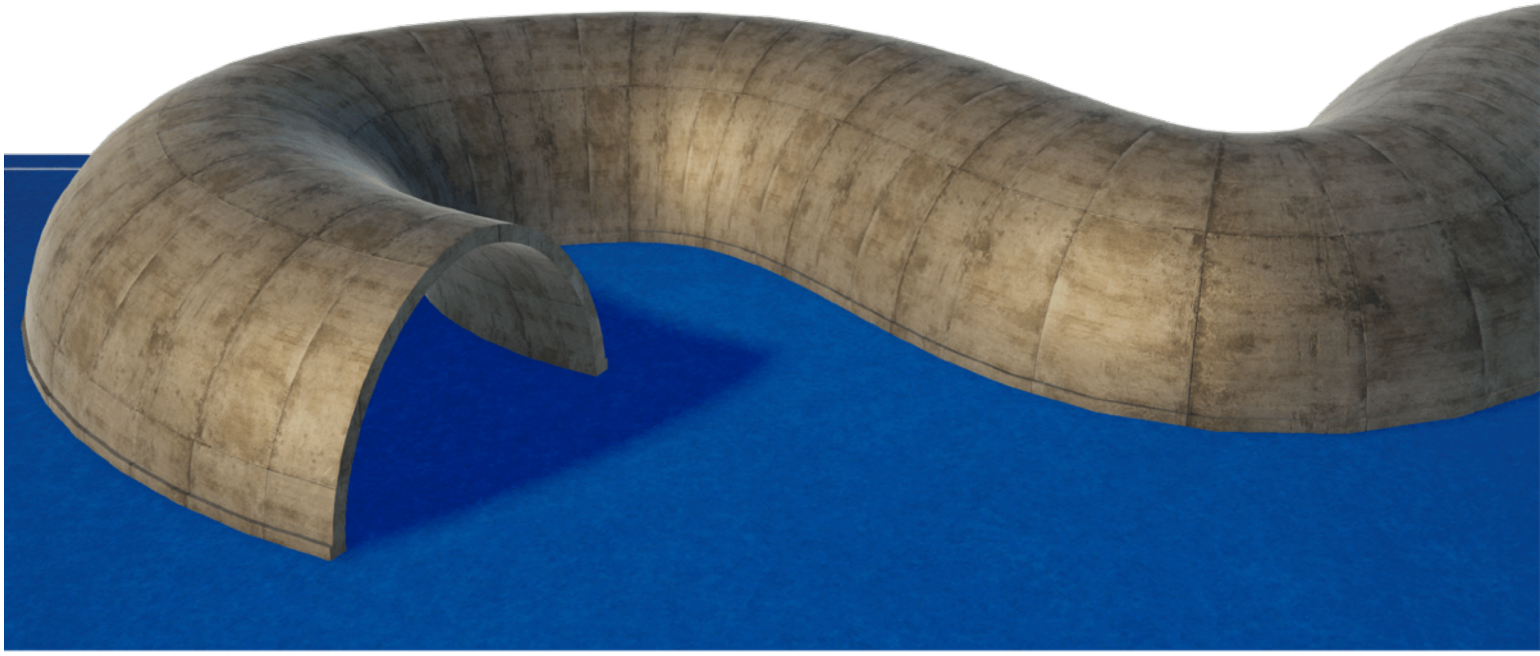
We'll use our 2D *gable* and an **Extrude** node to create the end wall under the roof. If we allow the 2D gable to close on the **Extrude** node's *Plan Input*, we have the perfect cap shape that updates with our roof domes.



ShapeMerger can combine the 2D *gable* with the lower wall shape and window voids.

Our 2D gable can also be connected to a **PlanSweep** and used as a full-dome rooftop. The mirrored-dome *section* is fundamentally different from our single-dome *section*. It's not compatible with our *hip and valley* because it uses a different *plan* strategy.

One advantage of connecting a *2D gable* to **PlanSweep** is it can be swept with a **FreeCurve** *plan*. For obvious reasons we'll call this a *FreeCurve barrel*.

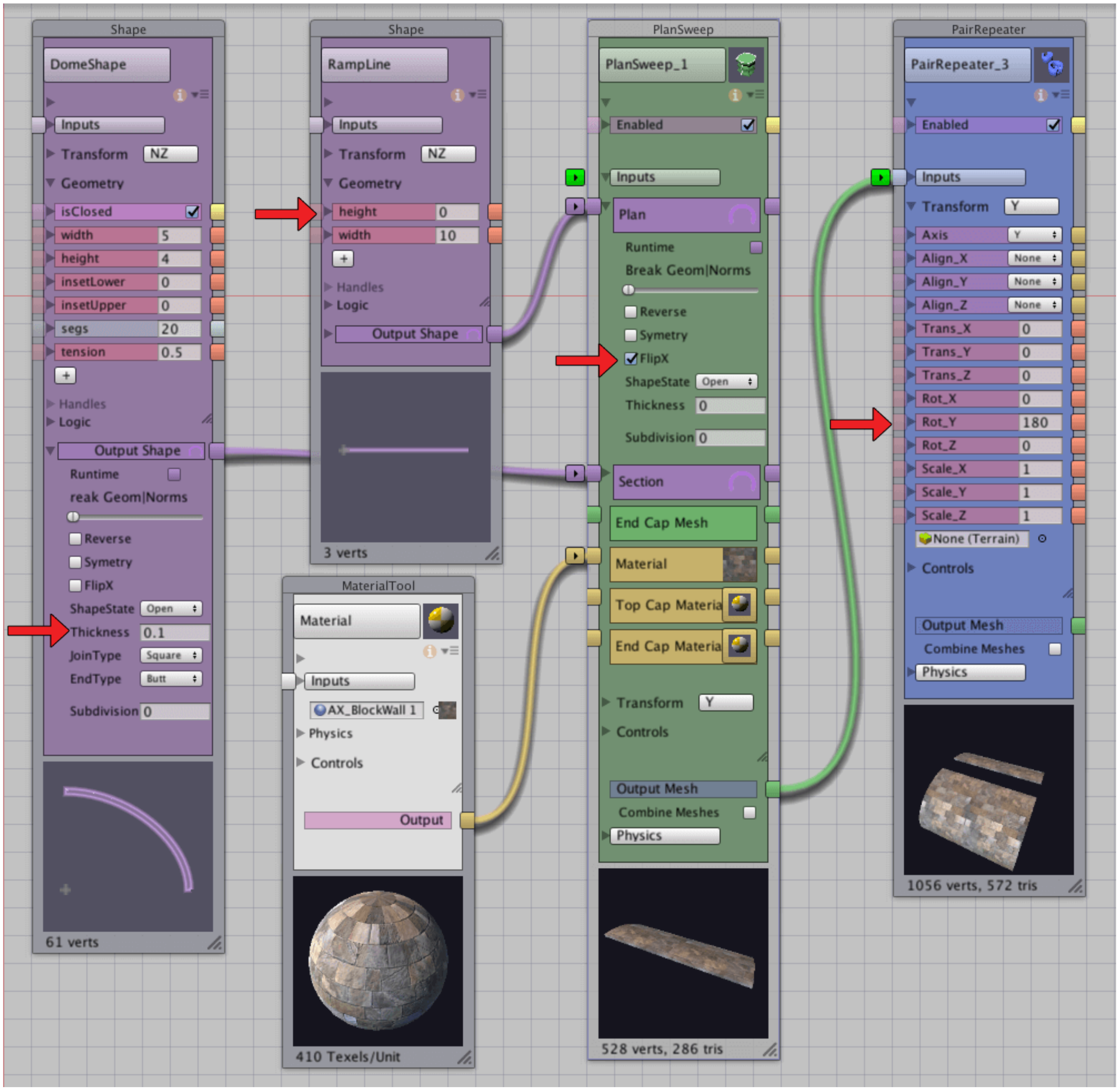


A *FreeCurve barrel* works well with smooth bezier, but creates ugly overlapping geometry on sharp corners.

Oculus Gable

Our third *gable* technique, and the most complicated, connects the *skillion*'s **PlanSweep** to a 3D **PairRepeater**. We'll use the **Separation** parameter to duplicate the *radius* gap we had with our *oculus dome*.

Again it seems simple, but replicating our *skillion* through the 3D **PairRepeater** can be a headache. Depending on how we created our 2-point *plan*, we might have two *skillions* pointing in random directions. For **PairRepeater** to work we need to "straighten" our line so it is *horizontal* (or *vertical)* in the 2D Shape preview.


Shape
DomeShape
Inputs
Transform NZ
Geometry
isClosed ✓
width 5
height 4
insetLower 0
insetUpper 0
segs 20
tension 0.5
Handles
Logic
Output Shape
Runtime
reak Geom|Norms
Reverse
Symetry
FlipX
ShapeState Open
Thickness 0.1
JoinType Square
EndType Butt
Subdivision 0
61 verts

Shape
RampLine
Inputs
Transform NZ
Geometry
height 0
width 10
Handles
Logic
Output Shape
3 verts

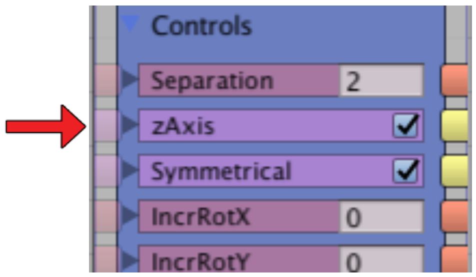
MaterialTool
Material
Inputs
AX_BlockWall 1
Physics
Controls
Output
410 Texels/Unit

PlanSweep
PlanSweep_1
Enabled ✓
Inputs
Plan
Runtime
Break Geom|Norms
Reverse
Symetry
FlipX ✓
ShapeState Open
Thickness 0
Subdivision 0
Section
End Cap Mesh
Material
Top Cap Materia
End Cap Materia
Transform Y
Controls
Output Mesh
Combine Meshes
Physics
528 verts, 286 tris

PairRepeater
PairRepeater_3
Enabled ✓
Inputs
Transform Y
Axis Y
Align_X None
Align_Y None
Align_Z None
Trans_X 0
Trans_Y 0
Trans_Z 0
Rot_X 0
Rot_Y 180
Rot_Z 0
Scale_X 1
Scale_Y 1
Scale_Z 1
None (Terrain)
Controls
Output Mesh
Combine Meshes
Physics
1056 verts, 572 tris

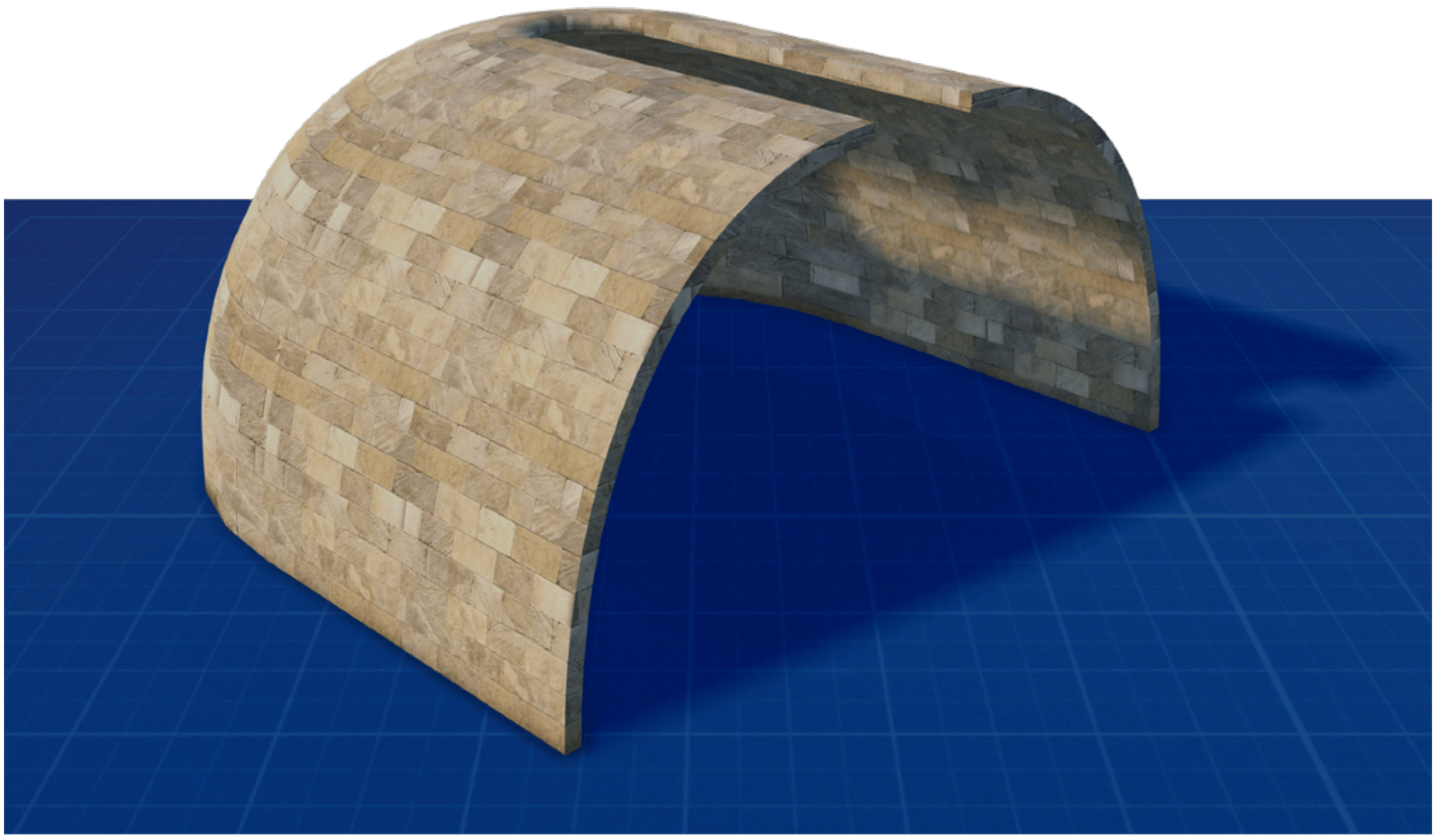
On the **PairRepeater**, turn **Symmetrical** on so our skillions mirror each other, and toggle **zAxis** until the insides of the gables face each other – or at least face in opposite directions. The line shape node's **Trans_X** and **Trans_Y** may need adjustment to line up the two roofs.

Next check the **Separation** control. If the domes touch when **Separation** is set to *zero*, but pass through each other when set higher, we'll probably need to toggle **Flip_X** somewhere in the *plan*. We can use the **Flip_X** setting on the **PlanSweep**'s *Plan Input*.



Separation 2
zAxis ✓
Symmetrical ✓
IncrRotX 0
IncrRotY 0

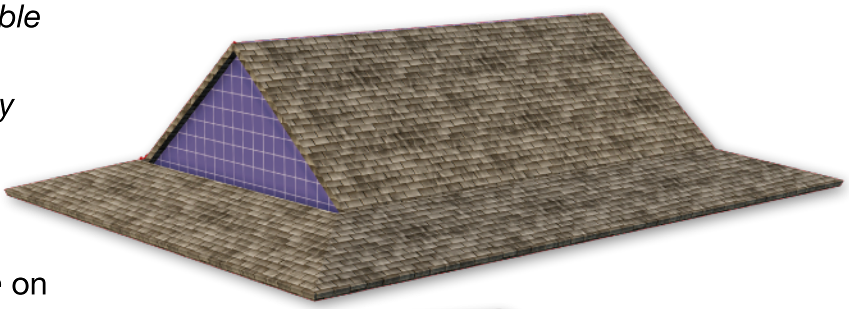
Once we have the *gable* aligned, we can expose the **Rot_Z** and **Trans_Y** of the **PlanRepeater** node for parametric control. Our *oculus gable* can share it's dome *section* with a **Lathe** node, and can be capped with complimentary *half shells* and *corner-rounds*. **Lathe**'s *radius* is half the **PairRepeater**'s *Separation*.



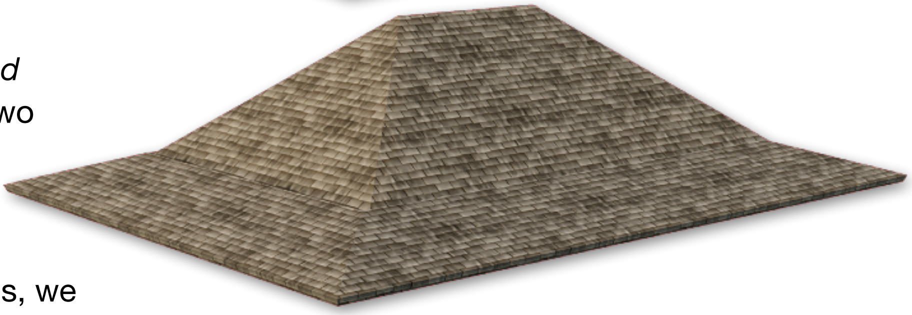
Combination, Plantation, Dutch Gable, Jerkinhead, Saltbox

A surprising number of traditional roof styles can be created with a combination of *gable* and *hip* roofs, stacked in the same way we created our irregular *hip and valley* roof.

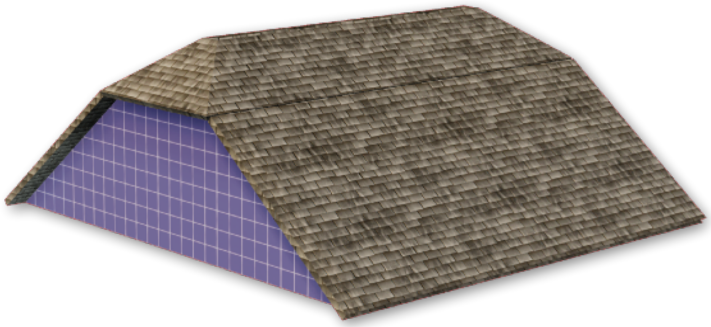
A typical *combination* roof is a large *gable* on top of a low *hip and valley roof*. *Plantation* roofs use a low *hip and valley* outside, and a steep *Hip and Valley* inside.



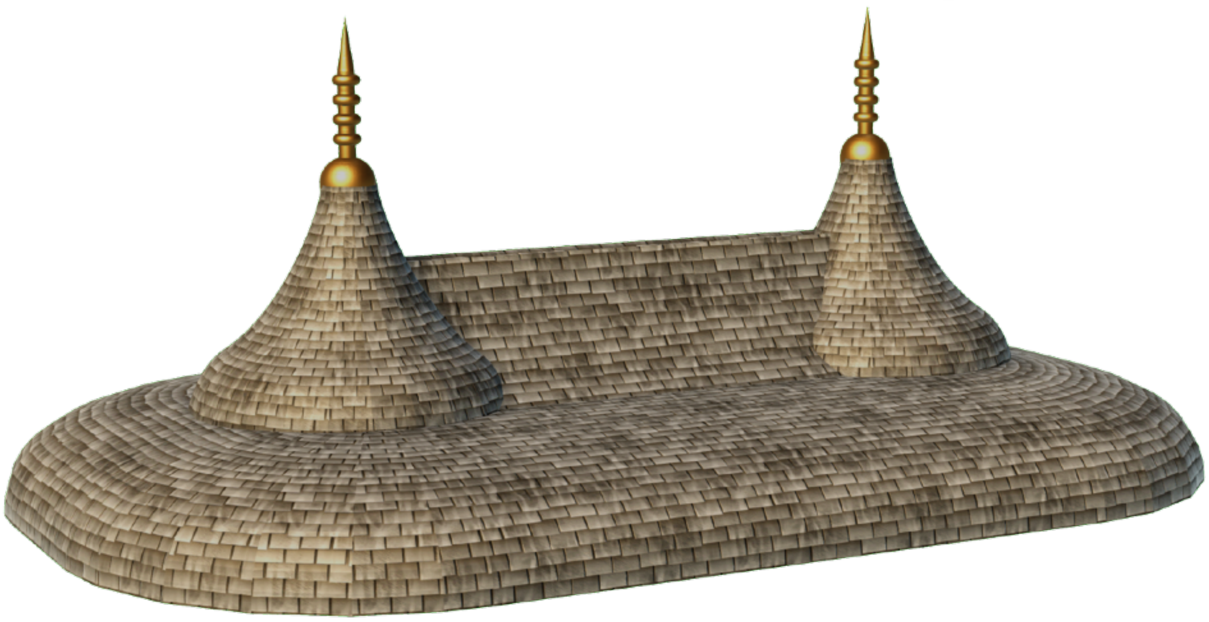
Dutch gable is essentially a small *gable* on top of a large *hip and valley*, while a *jerkinhead* is reversed: a small *hip and valley* on top of an *oculus gable* (or two *skillions*). A *Saltbox* is a single *skillion* next to a *gable*….



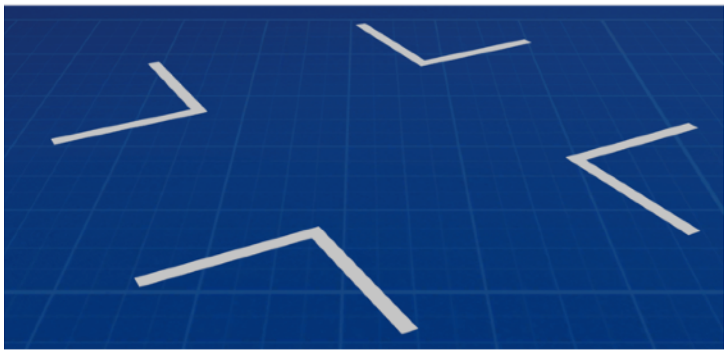
Adapting this idea to our own designs, we can create custom combination roofs by stacking dome, hip, and gable *sections* each connected to offset *plans*.

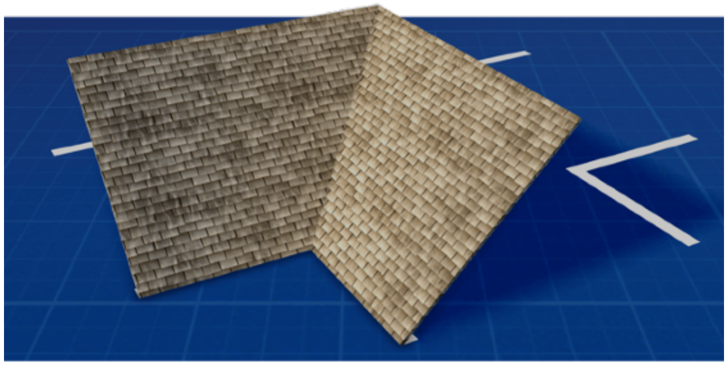


The top section might be a cosmetic ridge, or a **PlanRepeater** with prefab ornaments and gargoyle corners.



Cross-Joint Gable, Hip and Valley Corners

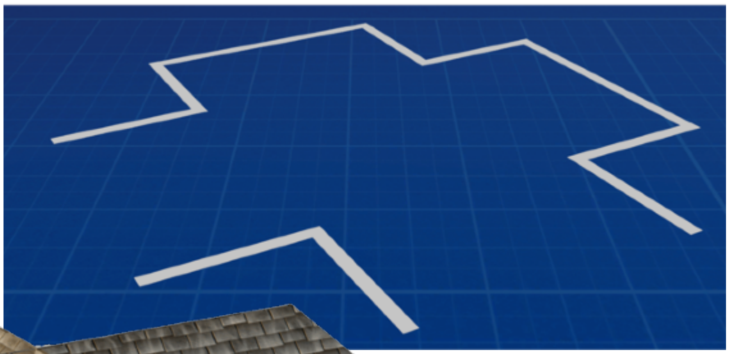




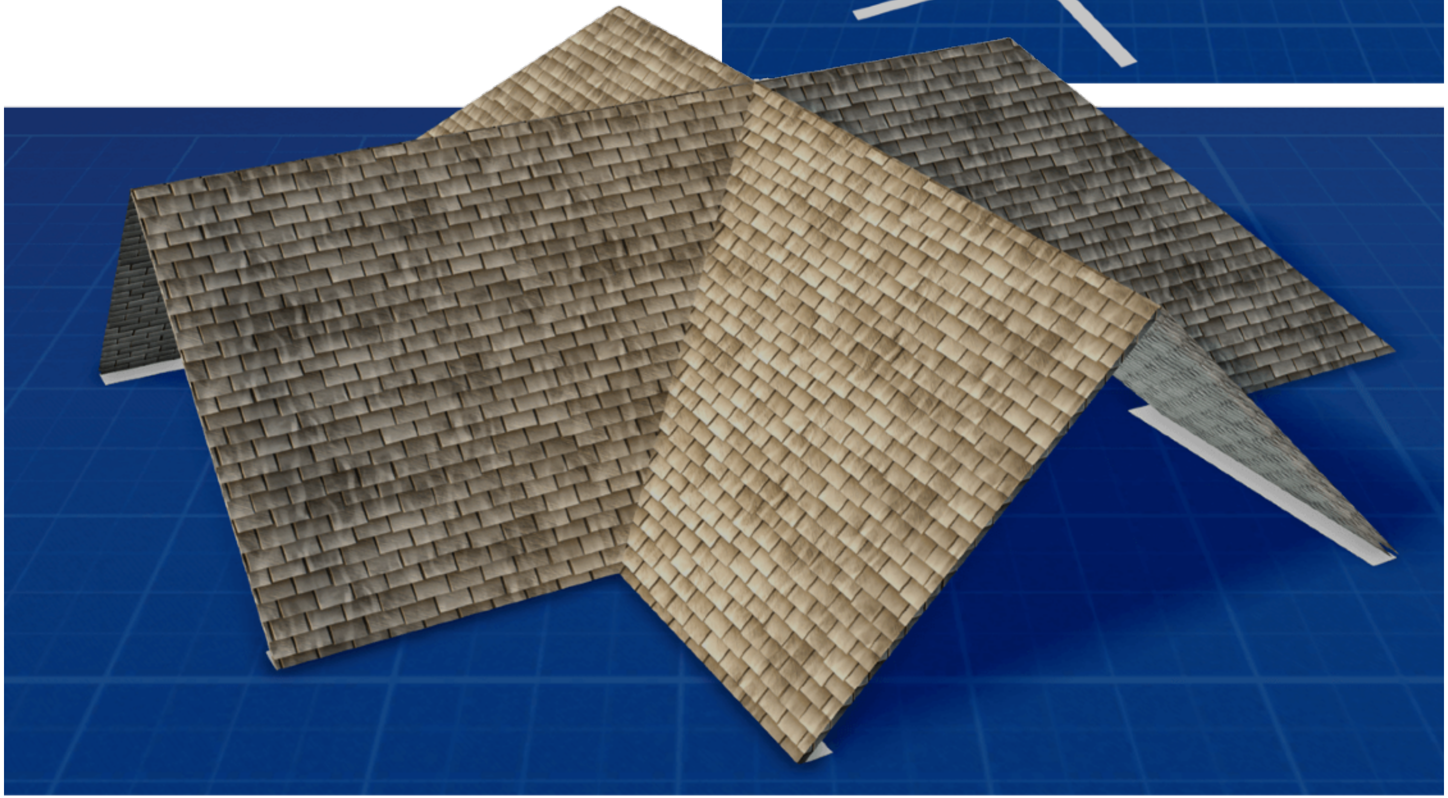
Our next rooftop is a *cross-joint gable* made with *hip and valley* corners that are created in-place using a 3-point **FreeCurve** in the *plan*.

The trick, like our *2-skillion gable*, is that our roofs will each be separate pieces that touch at their peak.

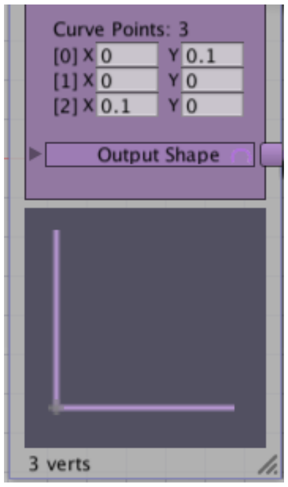
We'll need to add **FreeCurve** nodes to the graph, one for each gable. We're going to trace the low outside edges of our roof with **FreeCurve** points, and leave the *gable* ends open.



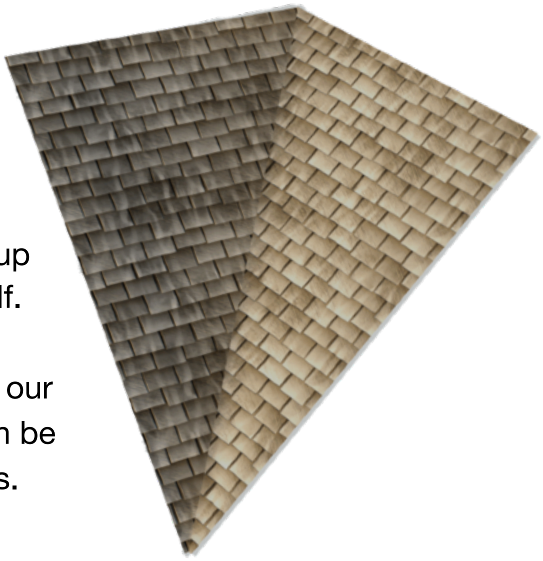
Each **FreeCurve** shape represents one unbroken roof *plan* running from *gable* to *gable* in a counter-clockwise direction.



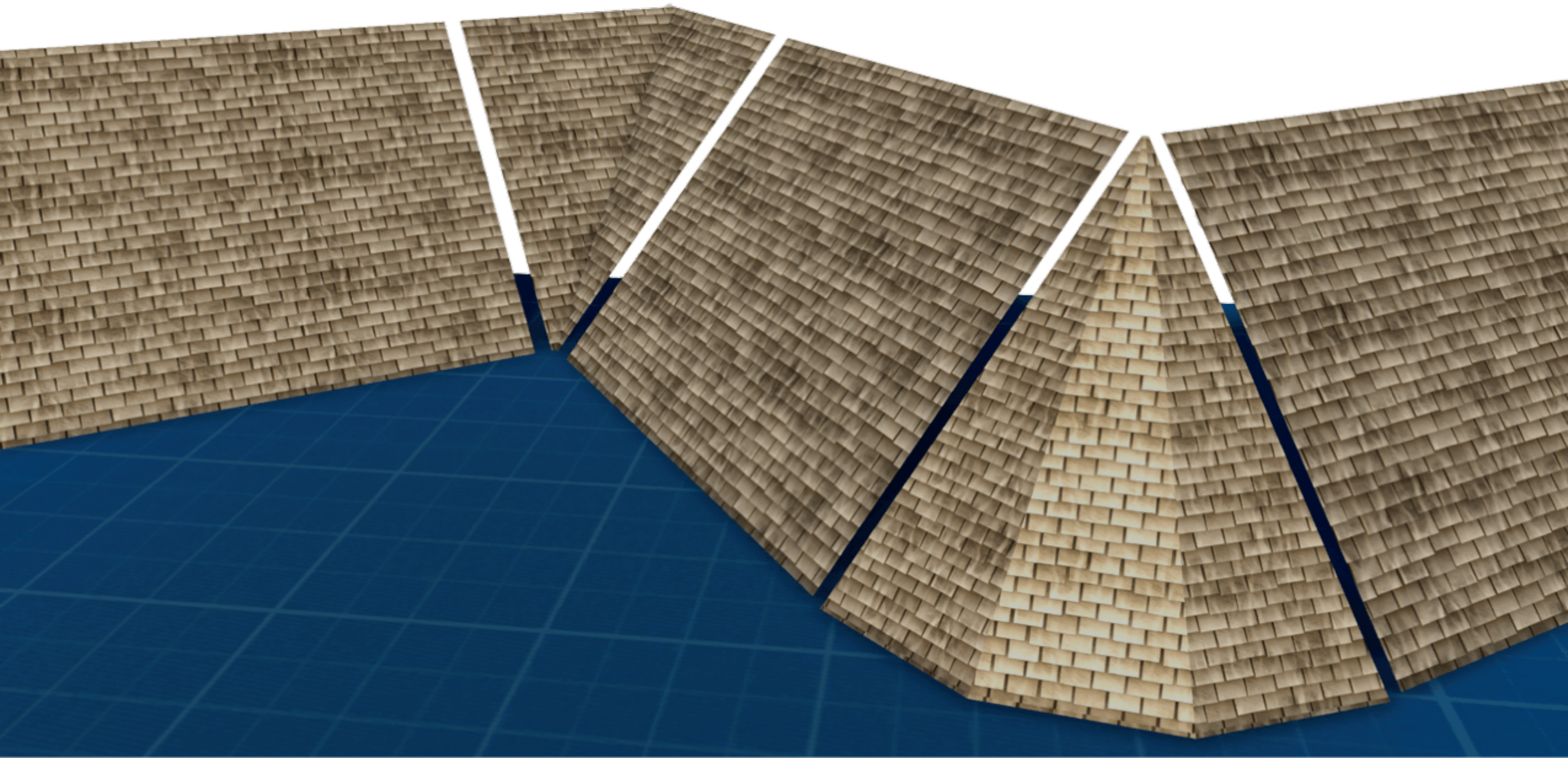
While it's tempting to combine multiple *plan* shapes into a single node using a **ShapeMerger**, there are so many things that can go wrong it just isn't worth it. Earlier when we created a *2D gable section* we got a small taste of how complicated it is to merge *open* shapes – we had to edit each node's *input* and *output* to re-*open* the shape at every connection. That's bad enough but as we learned with our 2-point *plan* for the *skillion*, **Archimatix** *plans* are actually a series of points that occur in a specific order. In a merged node the direction will flip, the order reverses, and the merged shapes attempt to close at every discontiguous connection. It's far simpler to leave the plans separate, and connect each *plan* node to its own **PlanSweep**.


Curve Points: 3
[0] X 0 Y 0.1
[1] X 0 Y 0
[2] X 0.1 Y 0

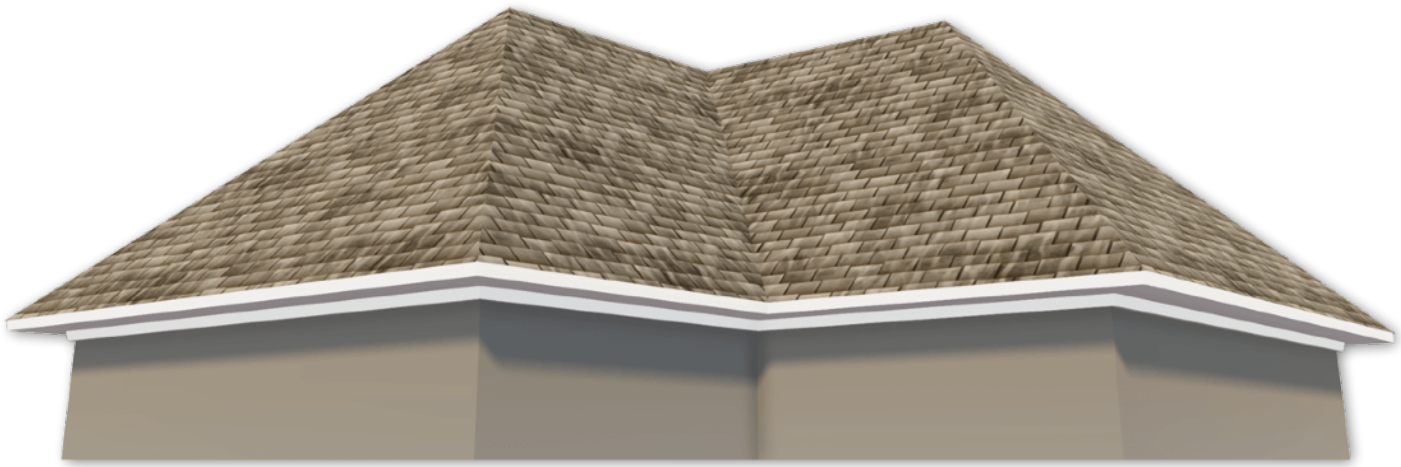
Before we move on, let's take a closer look at that **FreeCurve** corner, way closer. In fact let's edit our 3 points so the legs of our *plan* are extremely short, like *1*… or *0.1*… or *0.01*… We can build a roof from a tiny corner of a *plan* – it takes up no more space than the dome's *width* itself.



Compare this stand-alone valley corner to our **Lathe** *corner-rounds*. Rooftop facades can be assembled using only skillions and corners.

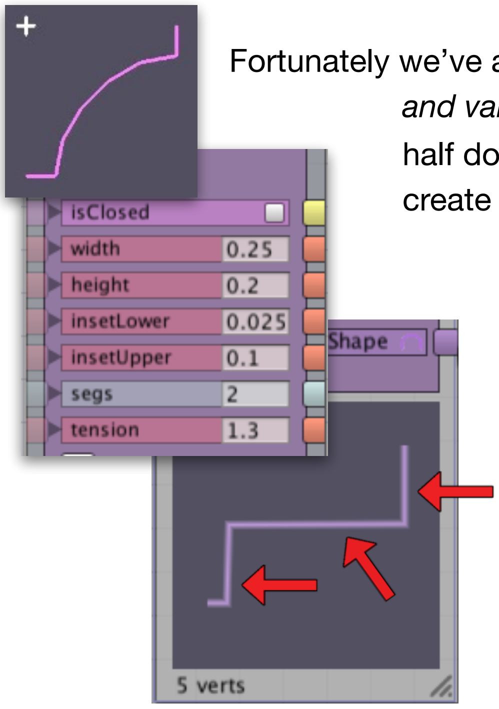


Fascia and Gable Trim



If our level-of-detail is low and the viewing angle prevents anyone seeing under the eaves, we simply can “paint” the *gable* end caps by connecting a second **Material** node to the **PlanSweep**’s *End Caps Material*. But real life *gables* are not solid slabs of marble or cement. Our *end caps* represent *fascia* trim, a functional border that follows the outside edges of the roof, adding rigidity under the eaves.

Fascia trim gives our roof “presence” like a drop shadow or underlined border. It’s the kind of detail we unconsciously expect to see, and when it isn’t there something just feels wrong. *Fascia* connects the roof to the wall, and provides a necessary level of detail to understanding our rooftop’s shape and scale.

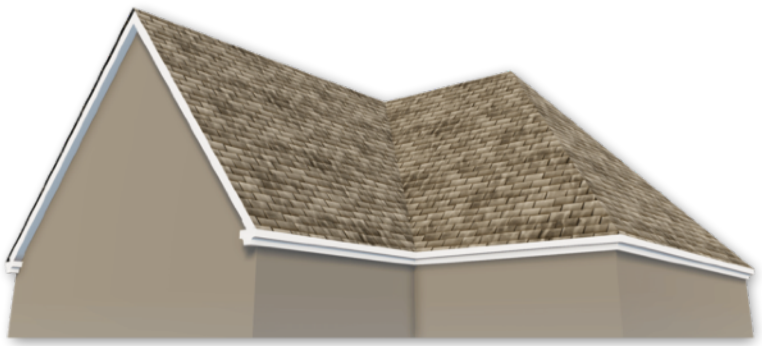
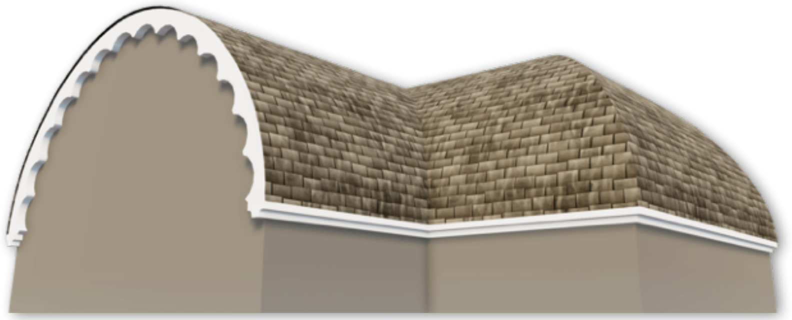


isClosed	
width	0.25
height	0.2
insetLower	0.025
insetUpper	0.1
segs	2
tension	1.3

Fortunately we’ve already made the *plan* shapes for *fascia* with our *hip and valley* and the lower sides of the *gables*, so our work is half done. We just need to add new **PlanSweeps**, and create a *fascia* trim *section* that fits under the roof.

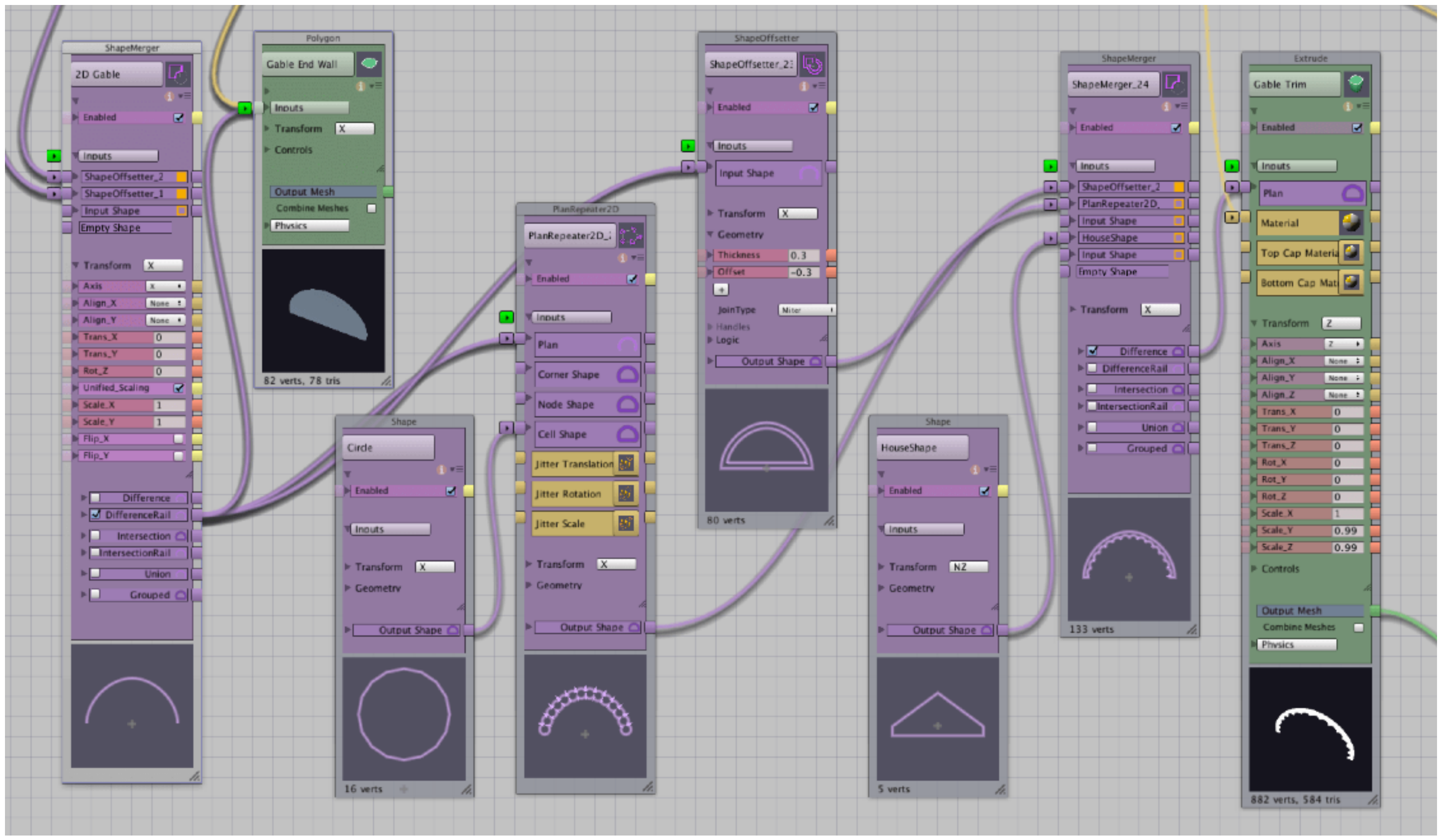
Try the **CoveMolding** shape for *fascia* trim – it looks like **DomeShape** upside-down, and the controls are similar. With some tweaking we can get a simplified trim *section* that includes a *fascia*, *soffit*, and *frieze board*.

Align the *section*’s **Trans_X** with the *plan*, using the white cross as a reference.

Rather than trying to **PlanSweep** around our *gable*, we'll use **Extrude** and make a new 3D mesh. *Gable* trim can be plain or ornate, since it is created separately from our *fascia* trim we have many choices.

We've already created a *2D gable* to cap the end wall. We can use the same 2D shape as the basis of our *gable* trim. The trick is to branch the *2D gable* shape to an **Offsetter** node where we'll add *Thickness* and *Offset*, just like our *truss frame*.



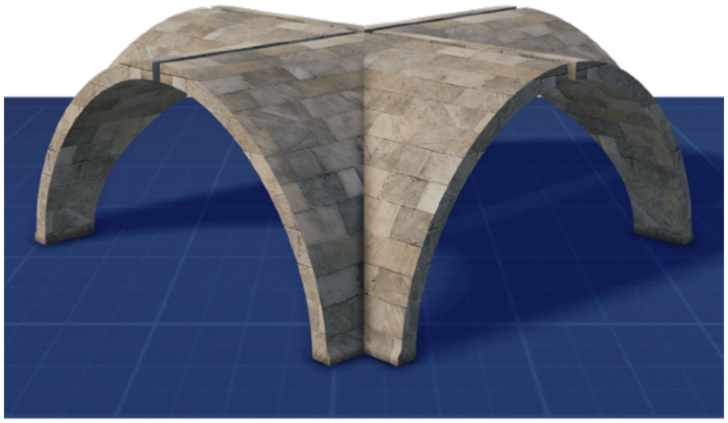
In the above graph, I've branched the *2D gable* to a **PlanRepeater** and subtracted a scalloped pattern of knock-out circles. Everything is brought together in a **ShapeMerger** node, along with another cut-out to remove the bottom of the *closed* shape, before connecting to an **Extrude** node.

Elaborate gable brackets and gingerbread decorations can be created the same way.

Groin Vault, Ribbed Vault

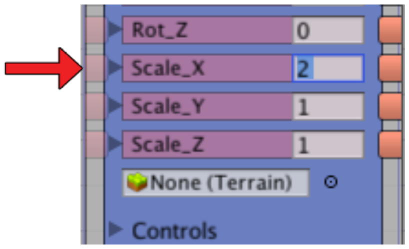
Our bonus example isn't really a roof, but it uses a *cross-joint* corner and repeaters to create vaulted ceilings. This time we are under the domes so we'll need to alter our perspective. Turn on **Backfaces** or **Thickness** if necessary.

We'll patch our single *vault* corner into a **RadialRepeater**. Set **Sectors** to *4* and **Radius** to *zero*.



Adjust **Trans_X** and **Trans_Z** on the **PlanSweep**, and **Rot_Y**, until all four corners align in the **RadialRepeater** into a *Groin Vault*, a cross-joint for *barrel* tunnels, subways, and sewers.

Swap the **DomeShape** node for **GothicArcOpen** to get a *ribbed vault*.

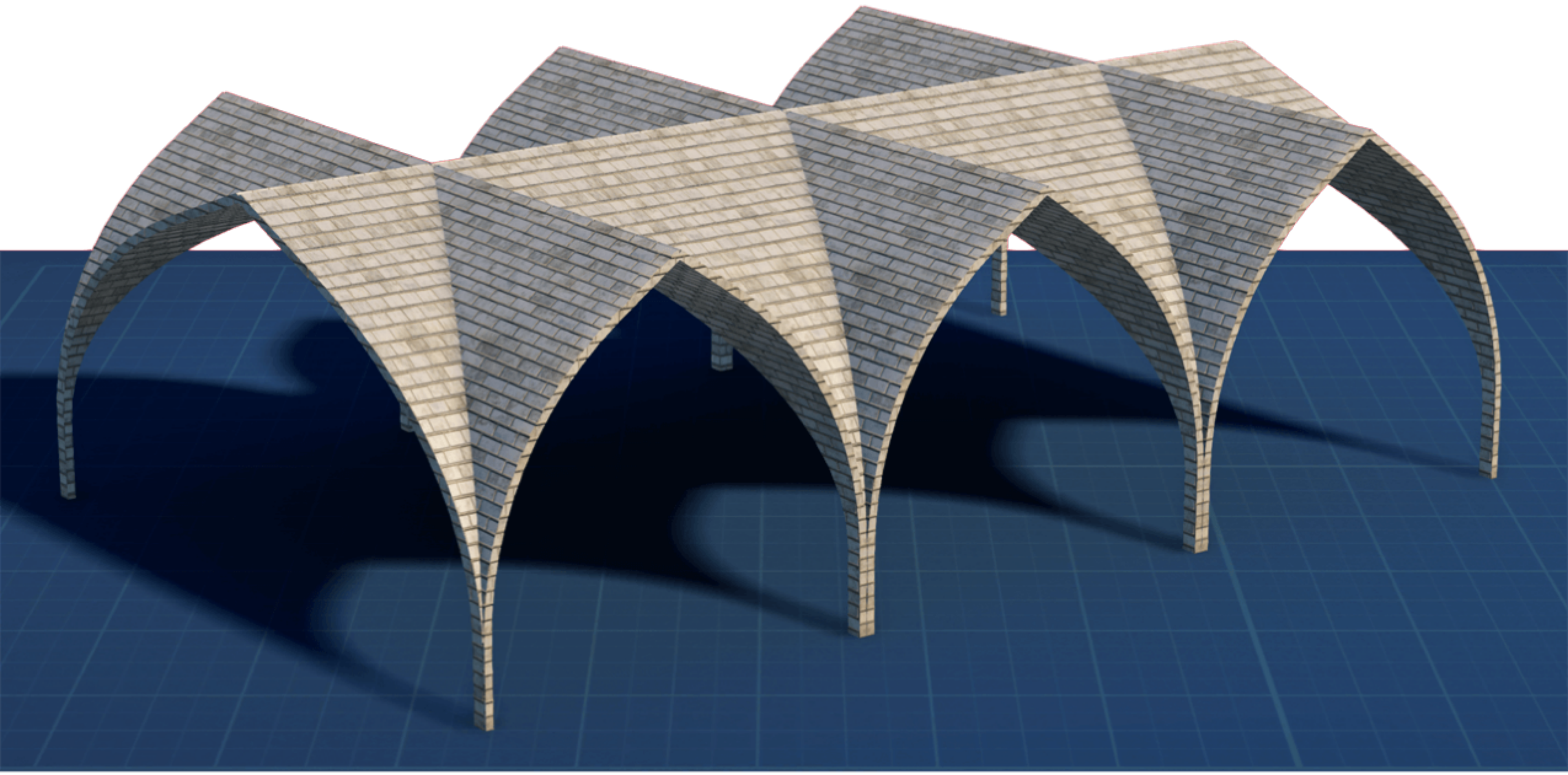


Rot_Z	0
Scale_X	2
Scale_Y	1
Scale_Z	1

None (Terrain)

Controls

On the **RadialRepeater** node we can *scale* the combined 4-corner *vault* along one axis, this creates a *vault* corner with a longer span on one side. Connect to a **LineRepeater** and we have the basis for a medieval cathedral.



Archimatix Dome and Rooftop Fundamentals

compatible with **Archimatix 1.10**
published January 23, 2018