

CONCEPT

WEBS

The site is bounded by highly varied volumetric wall types, providing great opportunities to attach to the surfaces. Referencing spiderwebs down a crevice, the aim is to provide a playful canopy that opens up imagination and whose web becomes a light sculpture.



NATURAL WEB CONSTRUCTION SEQUENCE

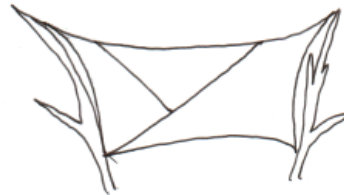
WEBS



1 bridge line made



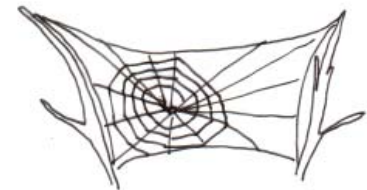
2 fork constructed



3 boundary web



4 spokes made



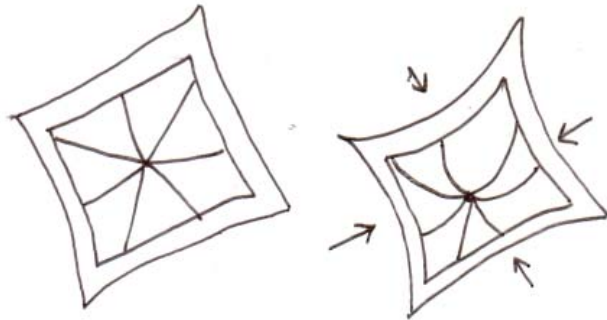
5 spiral web weaved
until edge reached

ARCHITECTURAL TRANSLATION OF IDEA

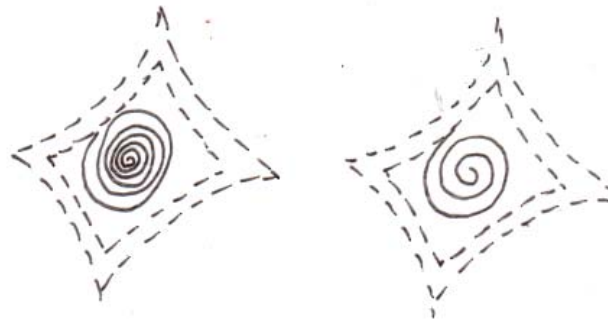
WEBS

1 component based on 4 points

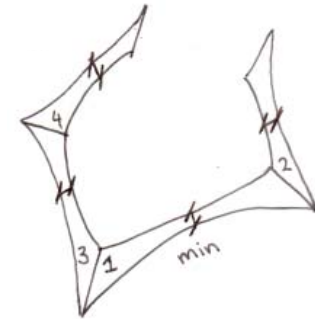
variables -



1 amount of structural relaxation



2 density of spiral weave

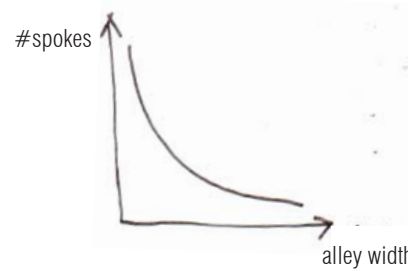


2 min thickness of web for mdf output

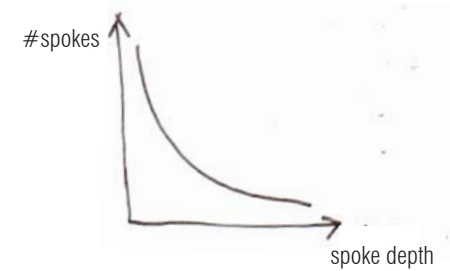
site reactive elements -



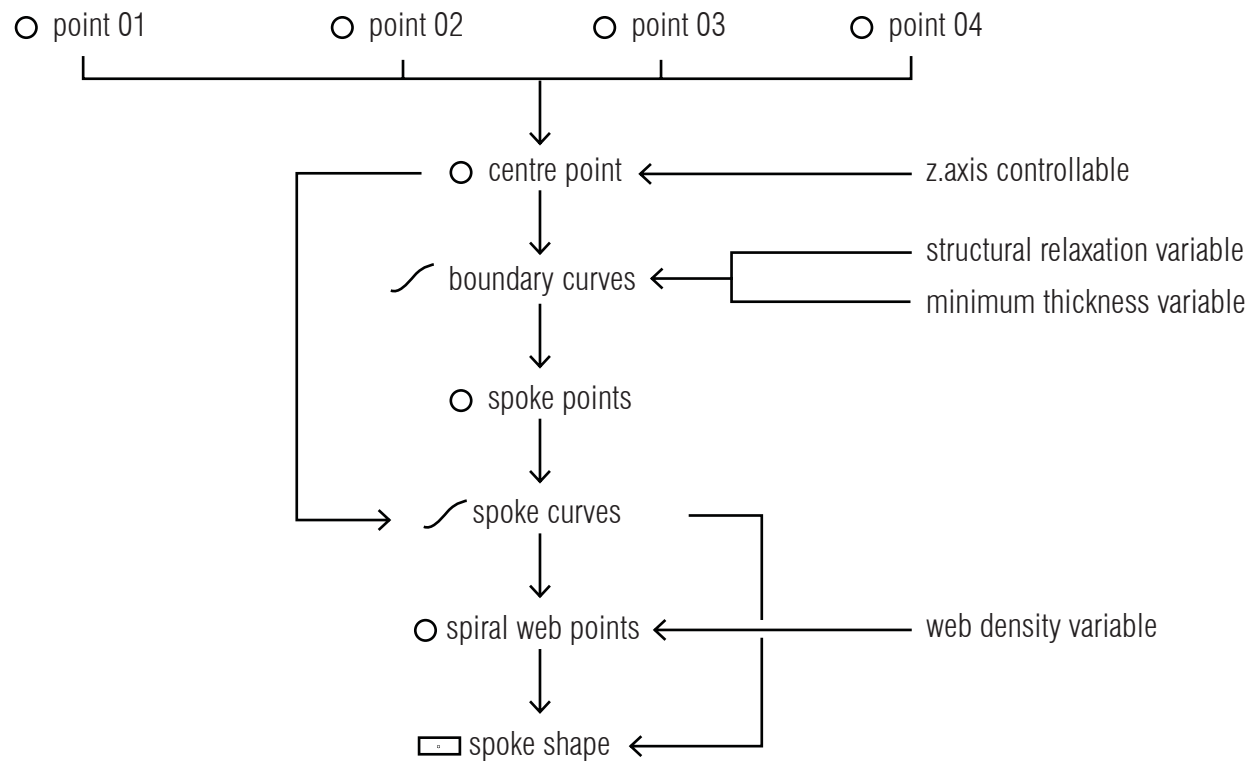
1 width at corner points \propto longest distance from centre

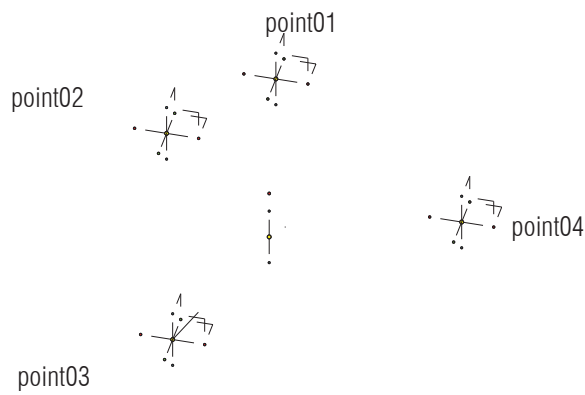


2 number of spokes $\propto \frac{1}{\text{alley width}}$

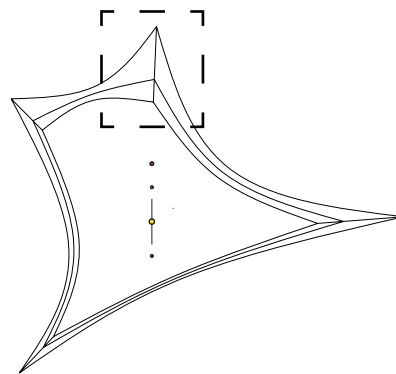
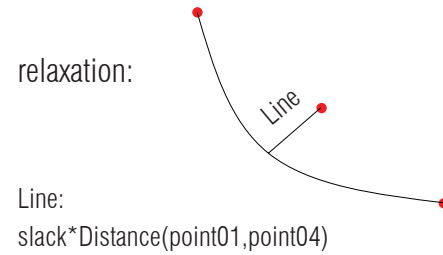


3 number of spokes $\propto \frac{1}{\text{spoke depth}}$



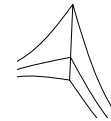


1 insert points

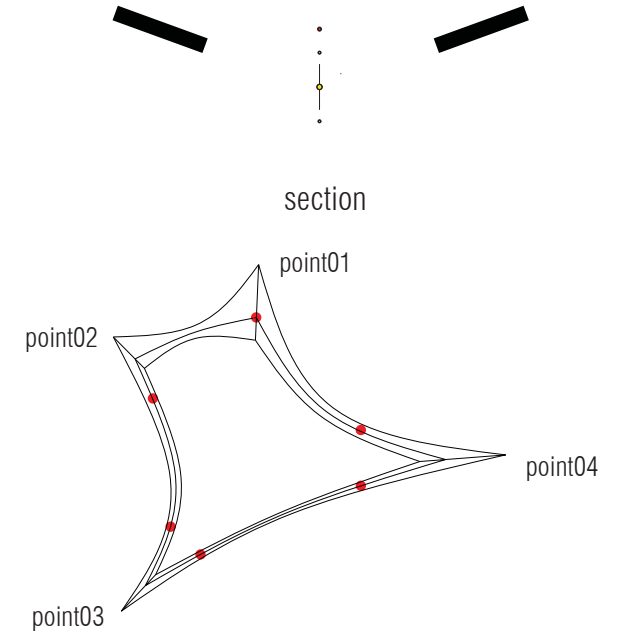


2 boundary curves angled towards centre point

corner length:



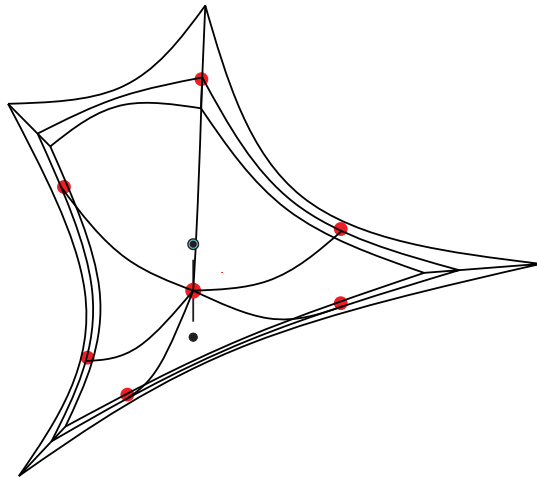
Line. Length =
 $.2 * NPmaxInList(\{$
 $Distance(point01, point02),$
 $Distance(point02, point03),$
 $Distance(point03, point04),$
 $Distance(point04, point01)\}$
 $, false)$



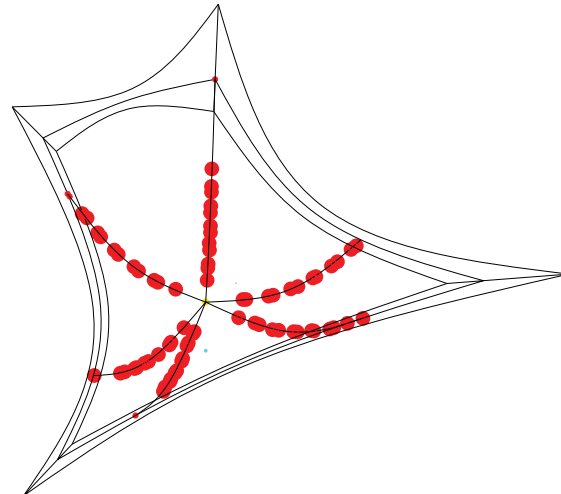
3 make composite curve and add spoke points by number along curve

Number:
 $Round(k / (Distance(point01, point03) + Distance(point02, point04)) + 4)$

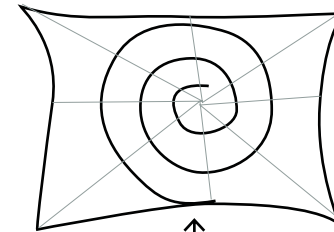
where k is a constant
 (+ 4 means the minimum is always 4)



4 Spoke Curves by Points where 3rd point calculated as in relaxation



5 Spiral Points added



↑ spiral stops when hits edge ↑



```
function (Curve myCurve,double tightness)
```

```
{
  double incVal = 0;
  double lengthCurve = 0;
  Point webPt = {};
  int i = 13;
```

// so spiral of points dont begin at Curve.T = 0

```
do
```

```
{
  webPt[i] = new Point();
  incVal = tightness*(i ^ 1.9) ;
  webPt[i].ByDistanceAlongCurve(myCurve[i%myCurve.Count],incVal);
```

// new points
// exponential growth so the higher the i the more it expands
// i%myCurve.Count allows point creation revolving repeatedly around myCurve

informs this →

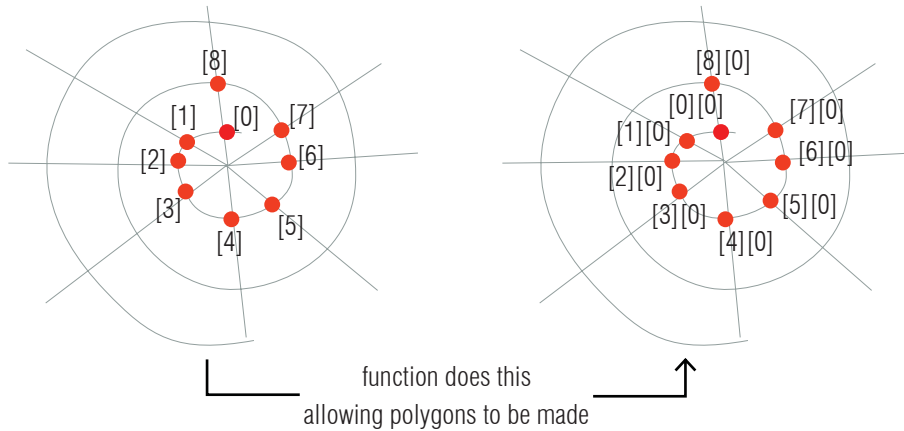
```
lengthCurve = myCurve[i%myCurve.Count].Length;
  i++;
}
```

// remembers the curve's length that the last point falls on
// i+1

```
while (incVal < lengthCurve);
```

//stop command so it stops creating points when the incremental distance is gets to the length of the curve

```
return webPt;
}
```



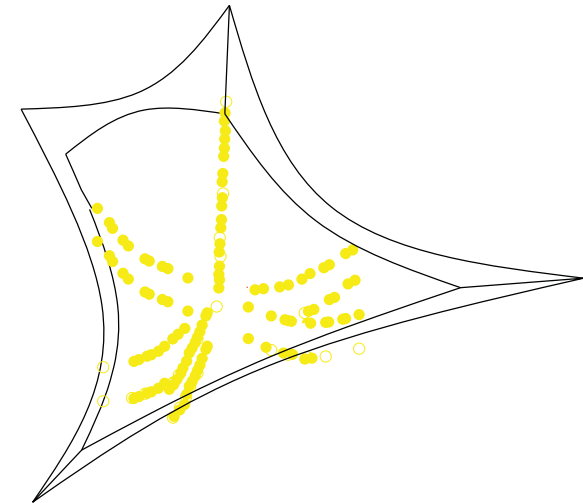
```
function (Point myPoint, Point myPoint2)
{
    Point SortPt = {};

    for (int i = 0; i < myPoint2.Count; i++)
    {
        SortPt[i] = {};
        int index = i%(myPoint2.Count); //returns each revolution second indice
        SortPt[index] = Add(SortPt[index], myPoint2[i]); //adds first indice from curve number
    }
}
```

// this bit is not working correctly. it attempts to add spoke point to the end of the second indice. Currently adding spoke point to as [i][0]

```
//for (int i = 17; i < myPoint.Count; i++)
//{
//    int index = i%(myPoint2.Count);
//    SortPt[index] = Add(SortPt[index], myPoint[i]);
//}
```

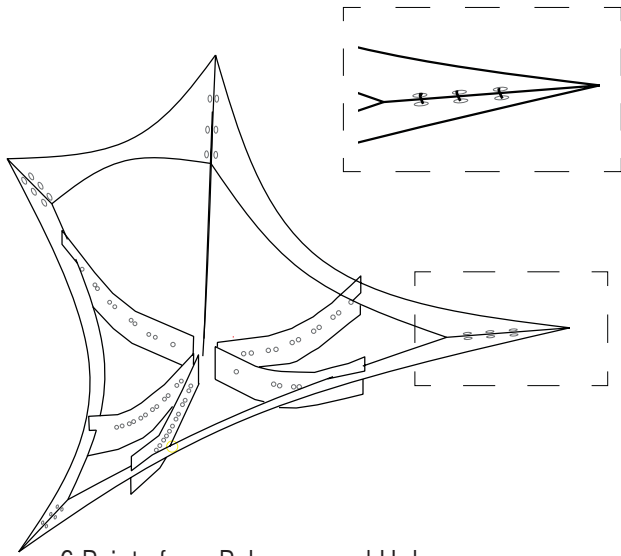
```
return SortPt;
}
```



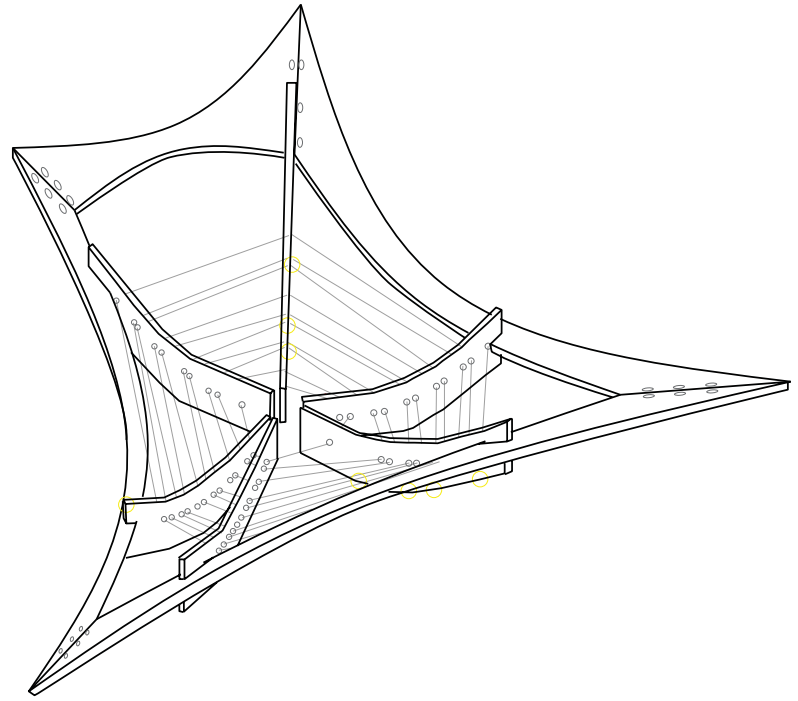
5 Points by Cartesian Coordinates to add depth

X:0 X:0
 Y:0 Y:0
 Z: -(k/spokePt.Count) Z: k/spokePt.Count

where k is a constant



6 Points form Polygons and Holes



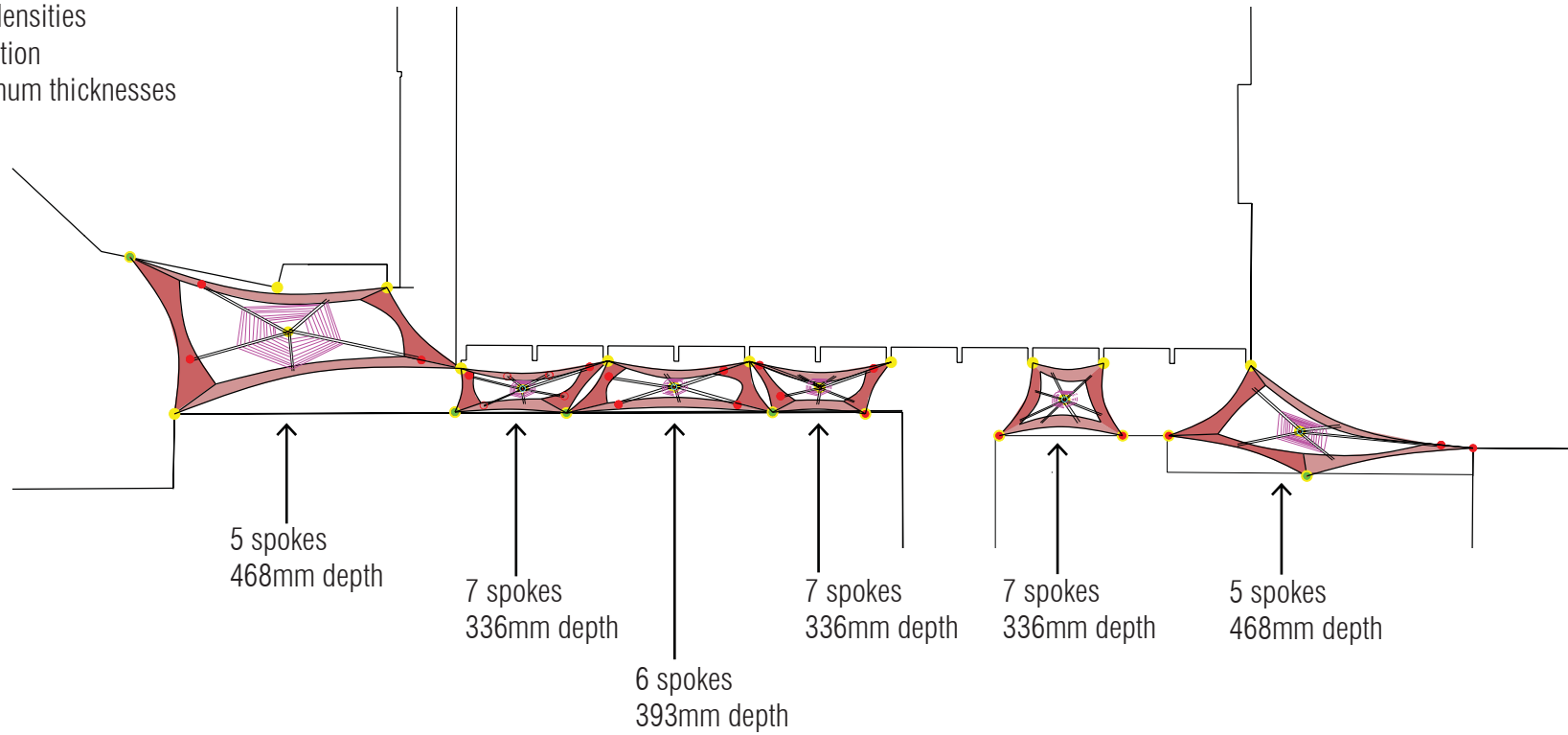
7. Holes looped with fishing line and mdf notched to both surfaces

1 component based on 4 points

- deformed shapes
- different spoke counts
- differing spoke thicknesses

inputting different variables for

- web densities
- relaxation
- minimum thicknesses



FABRICATION PLANNING

WEBS

