

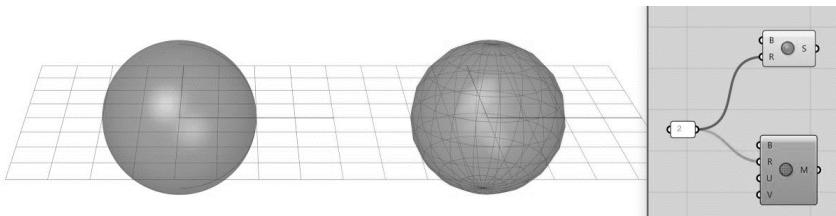
MESH

smoothness
form-finding
tensegrity
bending-active
hybrid

9. SMOOTHNESS DAN MESH

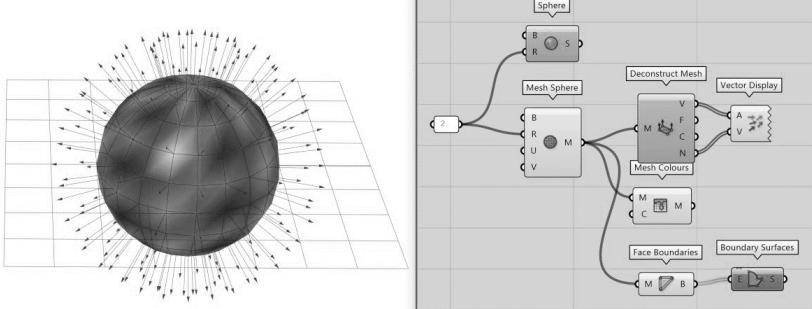
File latihan di bab ini dapat diunduh di:
<https://github.com/aswinindra/eskacangmerah>

Jika geometri berbasis NURBS yang kita pelajari sebelumnya pada dasarnya adalah representasi dari formula-formula matematika yang tidak eksis di dunia nyata (contohnya adalah *sphere*, obyek sphere sempurna tidak dapat dikonstruksi karena harus bulat sempurna, sedangkan yang dapat dikonstruksi adalah aproksimasi atau perkiraan yang paling mendekati obyek bola/sphere). Metode NURBS juga memiliki keterbatasan dalam mengolah obyek-obyek yang lebih organik atau *freeform geometry* yang memiliki tingkat kehalusan/ *smoothness* yang tinggi.



Gambar 191. Perbedaan antara Sphere (kiri) dan Mesh Sphere (kanan).

Obyek-obyek *Mesh* dikonstruksi oleh Polygon Mesh yang umumnya terdiri dari dua jenis: *Triangular Mesh* dan *Quadrangle (Quad) Mesh*. *Triangular Mesh* **pasti** merupakan obyek planar sedangkan *Quad Mesh* **belum tentu** planar. Jika kita kembali ke contoh di atas, maka obyek-obyek Mesh pada prinsipnya dikonstruksi oleh kumpulan Polygon Mesh yang mana, semakin banyak kumpulan Polygon ini, semakin halus dan lebih presisi obyek mesh bersangkutan.

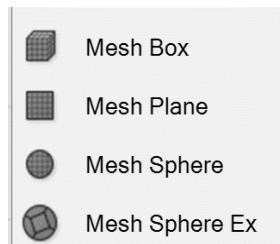


Gambar 192. Vektor Normal dari Quad Mesh

- **Mesh Sphere:** membuat obyek bola berbasis mesh.
- **Deconstruct Mesh:** mendekonstruksi obyek mesh menjadi komponen: vertices, Face, Color dan Normal
- **Face Boundary:** mengekstrak polygon dari obyek mesh dan dapat dikonversi menjadi Surface.

9.1. Mesh di Grasshopper

Ada beberapa teknik membuat obyek mesh di GH selain obyek primitif: **Mesh Box**, **Mesh Plane**, dan **Mesh Sphere**.



Gambar 193. Komponen Pembentuk Mesh di Grasshopper

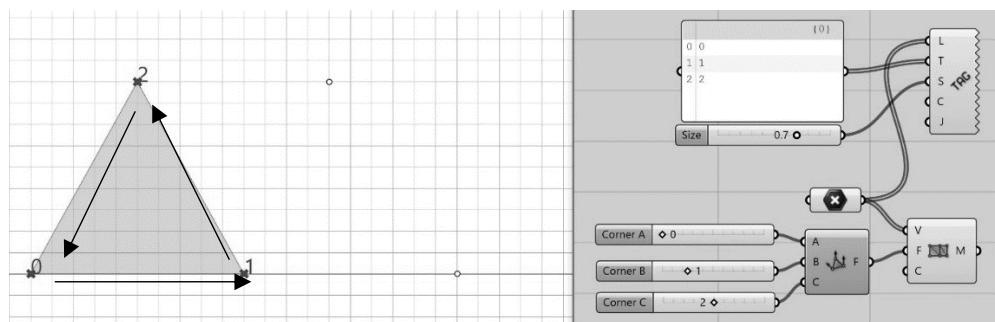
Teknik-teknik tersebut adalah:

1. *Mesh by topology*
2. *Mesh by triangulation*
3. *NURBS to mesh conversion*

9.1.1. Mesh dengan Topologi: Triangluar Mesh dan Quad Mesh

Sebuah Mesh didefinisikan sebagai obyek yang dibentuk oleh beberapa *Vertices* secara *sekuensial* (berurutan) yang menentukan topologi mesh bersangkutan. Obyek mesh paling sederhana adalah Triangular Face yang dibentuk oleh 3 vertices: 0, 1, 2 (lihat ilustrasi di bawah). Komponen yang digunakan untuk membuat mesh adalah: **Construct Mesh**. Komponen ini memerlukan input **V**= vertices: titik-titik yang akan dibuat mesh; **F**=face mesh yang dihasilkan dari titik-titik tersebut; **C**= color. Prinsip kerja dari Construct Mesh adalah face yang dihasilkan oleh sekumpulan titik-titik, harus diurutkan dengan orientasi yang sama (CCW atau CW) agar memiliki topologi yang sama.

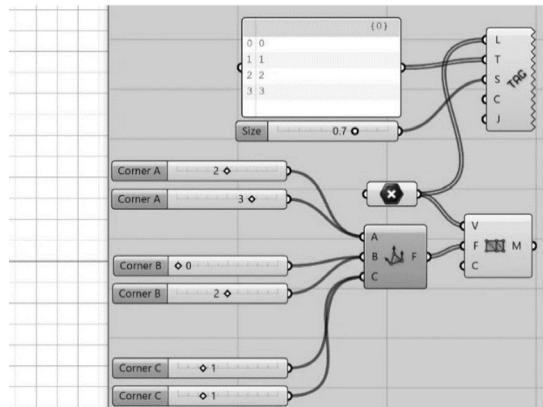
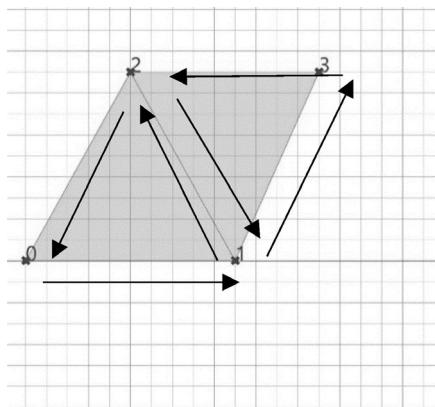
Konstruksi Satu Triangular Mesh



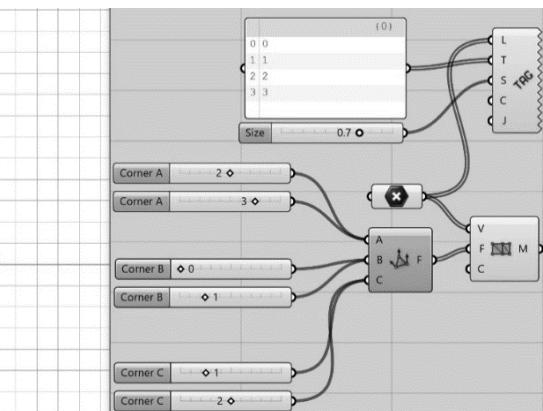
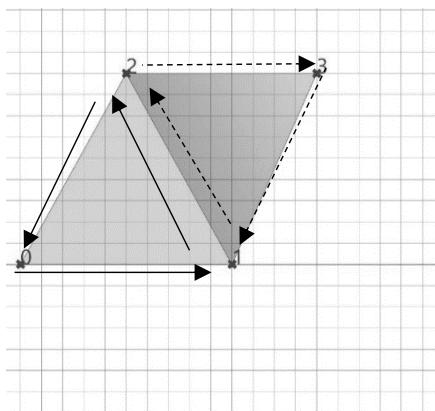
- Perlu tiga titik, buat titik-titik ini di Rhino, masukkan dalam komponen **Point** (Multiple Points).
- Masukkan komponen **Mesh Triangle**. Komponen ini pada dasarnya mengatur orientasi dan urutan susunan indeks titik-titik yang dipilih.

- Masukkan Number Slider (0-2) untuk menentukan urutan titik-titik.
- Contoh di atas, ada tiga titik, didefinisikan sebagai Triangular Mesh dengan urutan :0-1-2 atau 1-2-0, atau 2-0-1 (CCW). *Silakan dicoba beberapa kombinasi urutan ini.*
- Urutan orientasi tidak harus CCW tetapi yang jelas, pada kumpulan mesh yang berbasis triangle, harus memiliki orientasi yang sama.
- Bagaimana kalau kita menambahkan satu titik, sehingga ada 4 titik untuk dijadikan Triangular Mesh?*

Konstruksi Beberapa Triangular Mesh

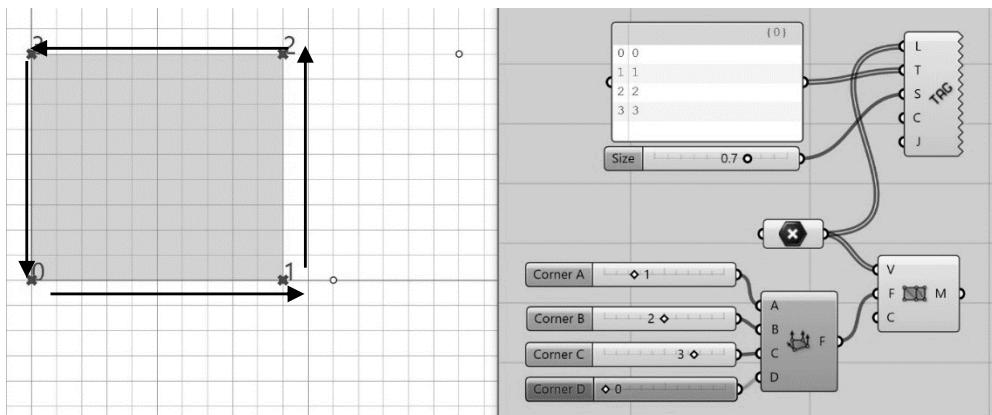


- Jika ada empat titik, maka akan ada dua Triangular Mesh, sehingga harus ditambahkan masing-masing satu Number Slider pada setiap titik segitiga.
- Perhatikan nilai pada masing-masing slider: Segitiga 1: 2-0-1; Segitiga 2: 3-2-1. Perhatikan bahwa pada masing-masing segitiga, urutan indeks verticesnya sama (CCW).
- Bagaimana kalau salah satu segitiga urutan verticesnya tidak sama (CW?)*



- Perhatikan bahwa pada Segitiga 2, urutannya adalah 3-1-2, dan ini berbeda urutan dengan Segitiga 1. Dalam kasus ini, GH akan mengeluarkan warning: *Incompatibility Warning* berupa warna gradasi. Ini berarti, warna *default* (cerah) adalah Front Face, warna gelap dan gradasi adalah Back Face.

Konstruksi Quad Mesh



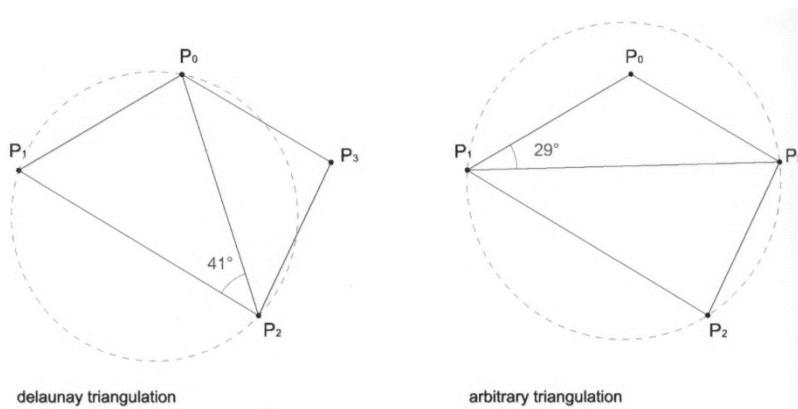
- Komponen **Quad Mesh** digunakan untuk menentukan urutan orientasi vertices atau titik-titik untuk membentuk Quad Mesh. Sama dengan Triangular Mesh, topologi obyek ini ditentukan oleh kesamaan urutan penyusunan titik-titik vertices, baik CW maupun CCW.

9.2. Mesh dengan Triangulasi: Algoritma Delaunay

Pembuatan mesh dengan topologi mendefinisikan obyek mesh baik Triangular maupun Quad dengan jalan menentukan urutan koneksi antara vertices. Teknik ini dapat digunakan jika jumlah titik-titik relatif sedikit, beraturan dan terdapat pengulangan. Teknik topologi tidak dapat digunakan pada kumpulan titik jumlah besar dan iregular.

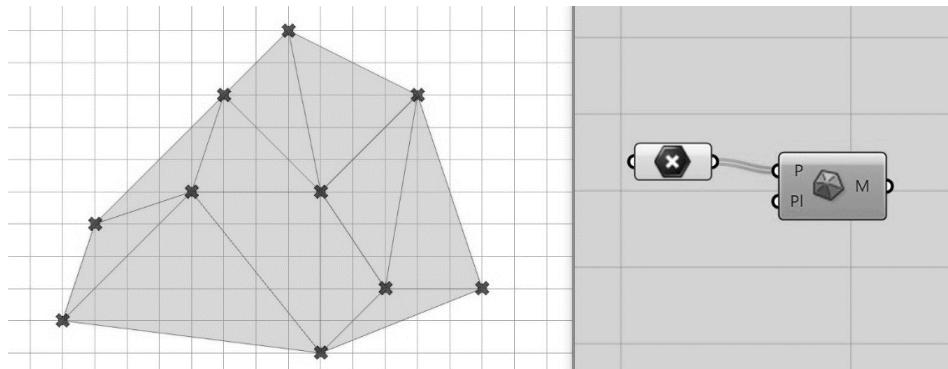
Terdapat beberapa teknik triangulasi untuk mendefinisikan Triangular Mesh pada kumpulan titik random/ arbitrer. Prinip dari teknik triangulasi adalah: meminimalisir perbedaan antara segitiga mesh dan menghindari bentuk segitiga yang sudutnya kecil agar terbentuk triangular meshes yang lebih reguler. Salah satu algoritma yang sering digunakan untuk triangulasi adalah **algoritma Delaunay** (dari matematikawan Rusia, Boris Delaunay yang menemukan metode ini di 1934).

Metode triangulasi Delaunay bekerja dengan cara menghubungkan semua titik-titik random dengan prosedur geometri yang memaksimalkan sudut-sudut pada segitiga yang terbentuk sehingga menghindari terbentuknya segitiga/ triangular mesh yang bersudut kecil. Algoritma Delaunay membuat segitiga dengan cara memilih tiga titik yang mana jika dibuat lingkaran yang memuat ketiga titik tersebut pada keliling lingkaran, tidak ada titik lain yang ada pada lingkaran tersebut. Prinsip triangulasi Delaunay adalah **sudut-sudut yang dibentuk oleh triangulasi ini tidak ada yang minimal**.



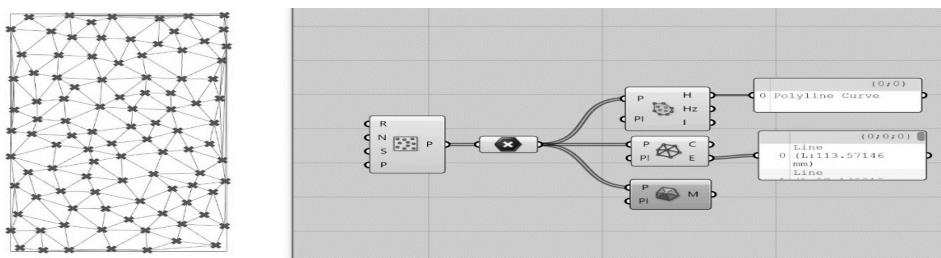
Gambar 194. Triangulasi Delaunay (kiri) dan Triangulasi Konvensional (kanan). (Sumber: Tedeschi, 2014)

Komponen **Delaunay Mesh** berfungsi untuk menghasilkan mesh berdasarkan triangulasi Delaunay atas sekumpulan titik-titik.



Gambar 195. Mesh Delaunay

Jika Delaunay Mesh menghasilkan sekumpulan mesh atas beberapa titik-titik, maka komponen **Convex Hull** akan membuat boundary atau polyline yang melingkupi semua titik-titik dan komponen **Delaunay Edge** akan mengekstrak Edge dari Delaunay mesh yang dihasilkan.



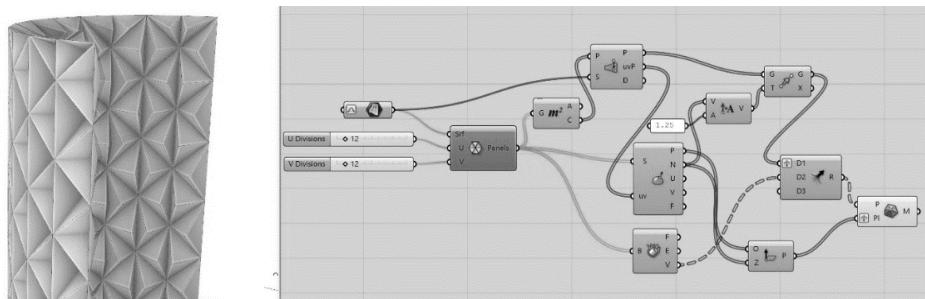
Gambar 196. Delaunay Mesh, Convex Hull dan Delaunay Edge

Contoh aplikasi Delaunay Mesh adalah sebagai berikut. Untuk dapat menggunakan contoh berikut anda harus menginstal Lunchbox (<https://www.food4rhino.com/app/>

lunchbox), salah satu add-ins untuk Grasshopper yang menyediakan beberapa fitur untuk panel, struktur dan bentuk-bentuk matematis lain. Setelah anda menginstalasi Lunchbox, maka tab Lunchbox akan muncul di Grasshopper:



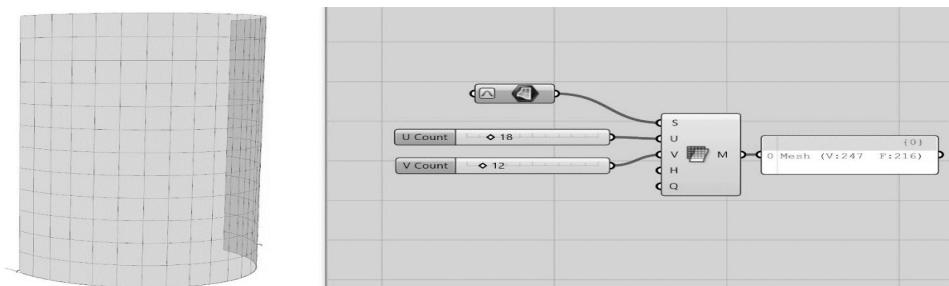
Salah satu komponen pada Lunchbox yang akan digunakan adalah **Triangle Pattern B**.



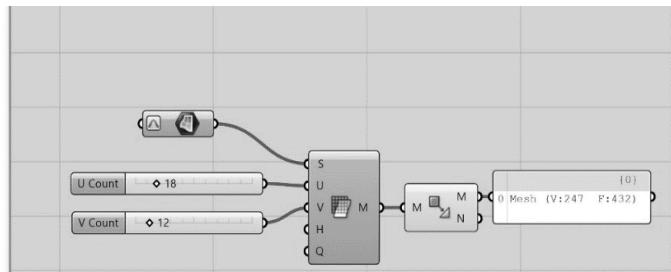
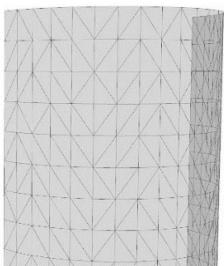
- Tentukan Surface dan reparameterize Surface
- Buat panel segitiga pada Surface dengan Triangle Panels B (komponen LunchBox), gunakan Number Slider untuk membuat parameter u dan v.
- Gunakan Area untuk menentukan titik pusat setiap panel segitiga dan tentukan titik terdekat dengan pusat tersebut berikut plane uv menggunakan Surface Closest Point.
- Gunakan Evaluate Surface untuk menentukan vector normal dari setiap titik tengah panel segitiga. Pindahkan titik terdekat dengan titik pusat menggunakan Move dengan besar jarak ditentukan menggunakan Amplitude dan arah pada vector normal.
- Gunakan komponen Deconstruct Brep untuk mengekstrak setiap vertices dari Triangle Panels B dan gabungkan dengan titik pusat yang sudah dipindahkan.
- Gunakan Delaunay Mesh untuk titik-titik vertices dan titik pusat yang dipindahkan. Gunakan komponen Plane Normal untuk menentukan arah perpindahan.

9.2.1. Mesh dengan Konversi

Mesh dapat juga diproduksi dari obyek berbasis NURBS. Komponen **Mesh Surface** berfungsi untuk mengkonversi Surface menjadi Mesh Surface berupa Quad Mesh.



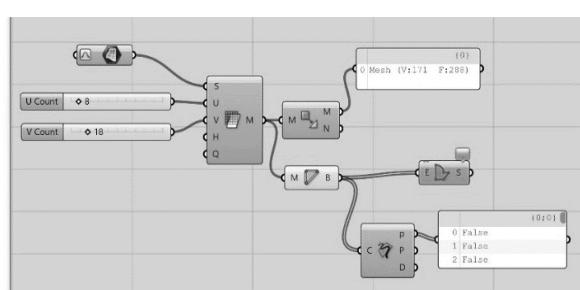
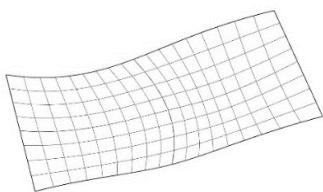
Quad Mesh dapat diubah menjadi Triangular Mesh menggunakan komponen **Triangulate**.



Hal yang perlu dicatat adalah:

- **Triangular Mesh** selalu planar, sedangkan **Quadrangular Mesh** belum tentu planar.

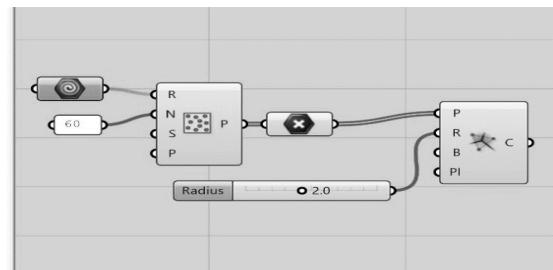
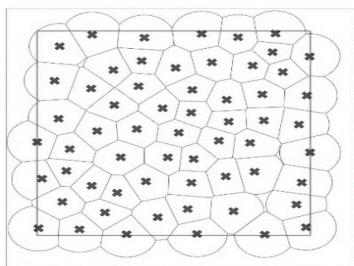
Anda dapat melakukan test untuk melihat apakah Mesh tersebut planar atau tidak dengan cara berikut.



- Mesh Surface akan menghasilkan Quad Mesh dari suatu Surface dan Triangulate akan menghasilkan Triangular Mesh.
- Gunakan komponen Face Boundary untuk menghasilkan polyline dari setiap mesh.
- Gunakan Planar untuk menghasilkan nilai T/F (T=planar, F=non planar) atau anda dapat gunakan Boundary Surface pada Face Boundary. Jika komponen ini menghasilkan surface, maka polyline tersebut planar.
- Contoh di atas memperlihatkan bahwa Quad Mesh pada surface contoh tidak ada yang menghasilkan permukaan planar. Anda bisa coba untuk Triangular Mesh.

9.2.2. Diagram Voronoi

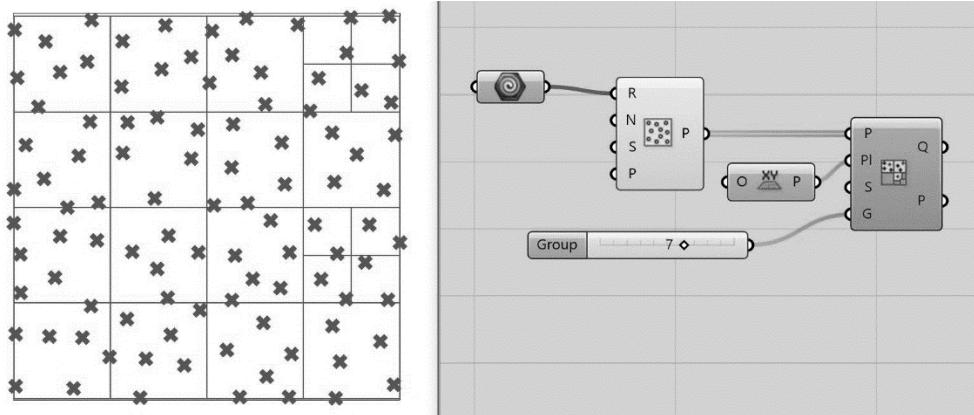
Diagram Voronoi dibentuk dari area maksimal yang dapat dihasilkan dari sebuah titik pusat. Jika ada kumpulan titik random, maka diagram Voronoi akan memaksimalkan area dari masing-masing titik tersebut.



- Komponen **Populate 2D** akan menghasilkan sekumpulan titik random di dalam suatu batas geometri tertentu.
- Komponen **Voronoi 2D** akan menghasilkan diagram Voronoi atas titik-titik tersebut yang dapat dibatasi oleh parameter B (Boundary) dan juga R (Radius).

9.2.3. Diagram Quad Tree

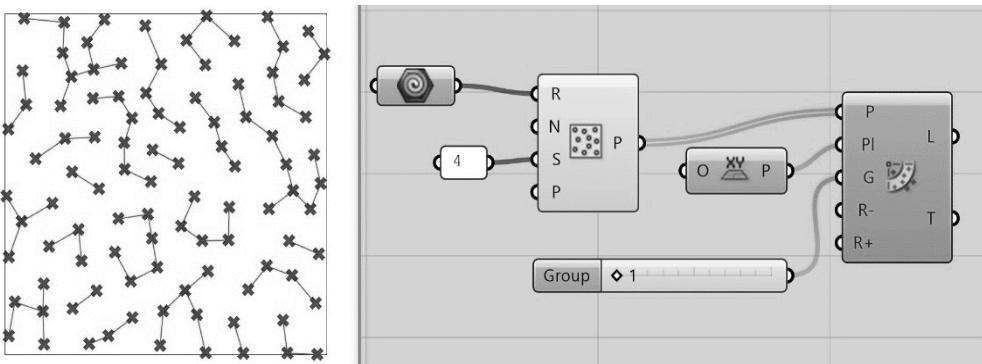
Diagram **Quad Tree** membuat batas-batas berupa area segi empat dari beberapa titik random berdasarkan berapa banyak titik yang dikelompokkan.



- Hasil dari komponen Quad Tree adalah Polyline dan banyaknya titik dari setiap Quad area.

9.2.4. Diagram Proximity

Diagram **Proximity** menghubungkan titik-titik random dalam jarak paling dekat satu dengan lainnya.

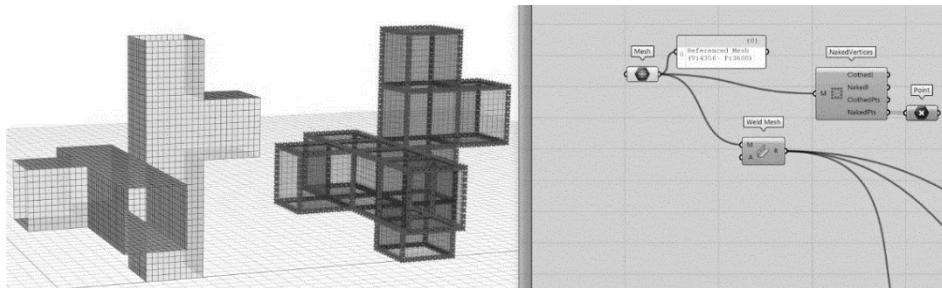


9.3. Smoothness

Membuat atau mengkonstruksi, memodifikasi obyek-obyek berbasis mesh untuk meningkatkan kehalusan, membuat obyek-obyek lebih organik dapat dilakukan dengan beberapa metode dan memerlukan beberapa add-ins.

Beberapa free add-ins yang dapat diinstal terkait dengan pemodelan dan manipulasi obyek berbasis mesh adalah:

1. **MesheditTools2:** <https://www.food4rhino.com/app/meshedit>, adalah add-ins GH yang memiliki fitur-fitur untuk bekerja dengan mesh. Setelah instalasi, fitur-fitur MeshEditTools akan ada pada tab Mesh.
2. **Weaverbird:** <http://www.giuliopiacentino.com/weaverbird/>, adalah Topological Mesh Editor add-ins GH yang memiliki beberapa fitur untuk pemodelan subdivisi dan transformasi obyek.
3. **Kangaroo Physics:** <https://www.food4rhino.com/app/kangaroo-physics>, adalah add-ins GH yang memiliki mesin fisik (Physics Engine) untuk simulasi-simulasi interaktif berdasarkan hukum-hukum fisika yang melibatkan beban, gaya, memiliki fitur-fitur untuk form-finding berdasarkan beban dan gaya tersebut.

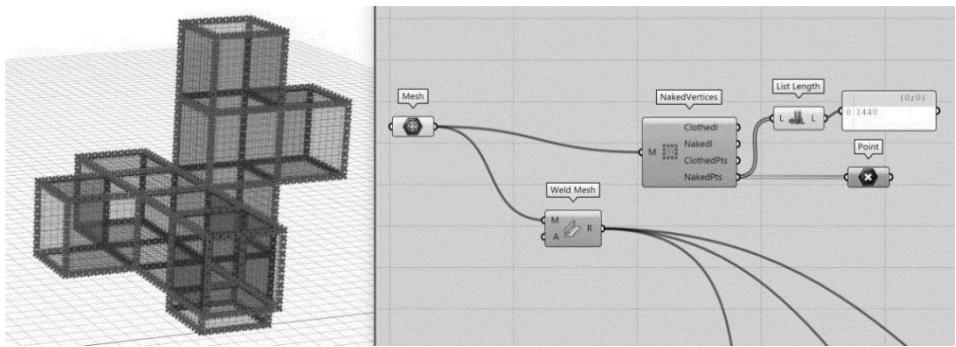


Misalkan kita memiliki obyek berbasis mesh berikut. Obyek ini dibuat di Rhinoceros menggunakan Mesh Plane berupa kotak dengan empat sisi (dua sisi lainnya adalah lubang). Tidak ada Mesh Plane yang berhimpitan, artinya tidak ada obyek mesh yang terduplicasi. Join obyek-obyek tersebut sehingga menjadi satu kesatuan dan assign sebagai obyek Mesh di GH.

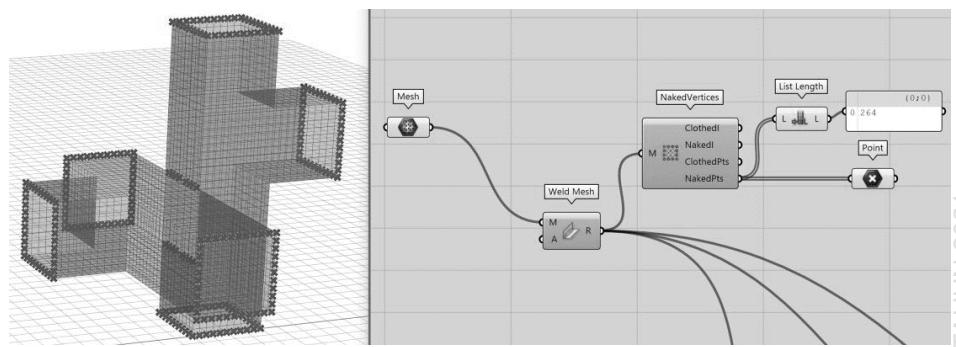
Beberapa hal yang perlu diketahui dalam membuat atau memodifikasi obyek berbasis mesh untuk metode smoothness adalah:

1. Obyek berupa kumpulan bidang-bidang/ surface, bukan obyek solid.
2. Tidak ada bidang yang berhimpitan.
3. Pada umumnya, naked vertices atau vertices yang tidak memiliki face/ mesh face akan disatukan/ weld sehingga tidak terjadi duplikasi naked vertices.

Pada contoh di atas, Ketika kita memasukkan komponen **Naked Vertices** (salah satu komponen Kangaroo → Mesh), kita akan melihat list titik-titik yang tidak memiliki face. Untuk menggabungkan naked vertices yang merupakan titik-titik yang terduplicasi (terdapat pada bagian obyek, bukan di pinggi obyek), kita menggunakan komponen **Weld Mesh** (salah satu komponen MesEditTools2).



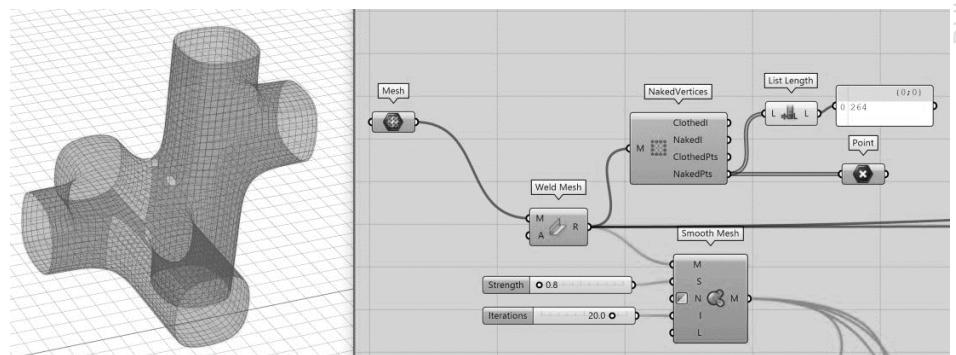
Naked Vertices tanpa Weld Mesh



Naked Vertices dengan Weld Mesh

Obyek dimana naked vertices sudah di-weld adalah obyek yang siap dikonstruksi atau dimodifikasi menggunakan beberapa metode penghalusan/ smoothness.

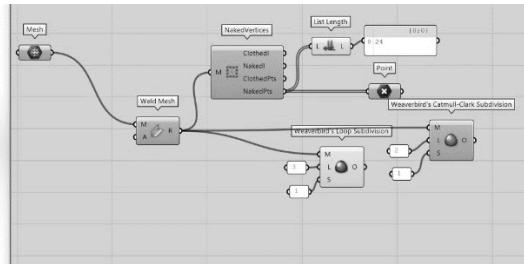
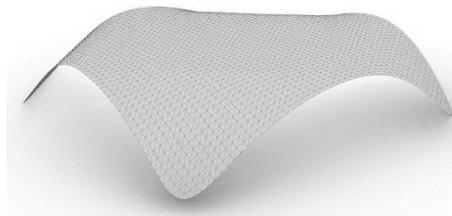
9.3.1. Smooth Mesh



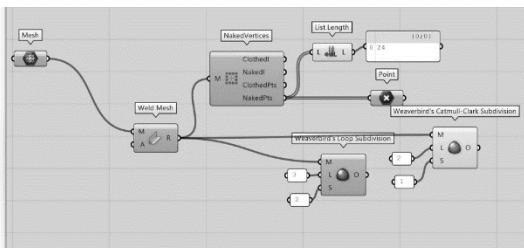
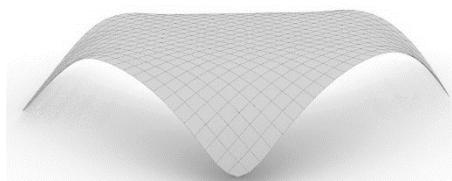
- Komponen Smooth Mesh (MeshEditTools) memerlukan input: M:mesh (welded naked vertices), S: Strength, N:Skip Naked Vertices, I: iteration.
- Parameter Skip Naked Vertices bila bernilai False maka naked vertices yang ada di pinggir obyek akan ikut diperhalus.

9.3.2. Weaverbird Subdivision: Loop Subdivision (Triangular Mesh) dan Catmull-Clark Subdivision (Quad Mesh)

Weaverbird memiliki dua metode untuk subdivision atau memperbanyak polygon dalam setiap polygon mesh eksisting.



- Loop Subdivision: membagi setiap polygon dari mesh eksisting ke dalam triangular mesh.
- Input L: loop menentukan seberapa dalam level subdivision.
- Input S: Smooth Naked Edge, adalah parameter untuk titik-titik yang terletak pada pinggir obyek (Naked Edge).



- Catmull-Clark Subdivision: membagi setiap polygon mesh eksisting ke dalam quad mesh. Parameter L dan S pada input memiliki fungsi yang sama dengan komponen Loop Subdivision.

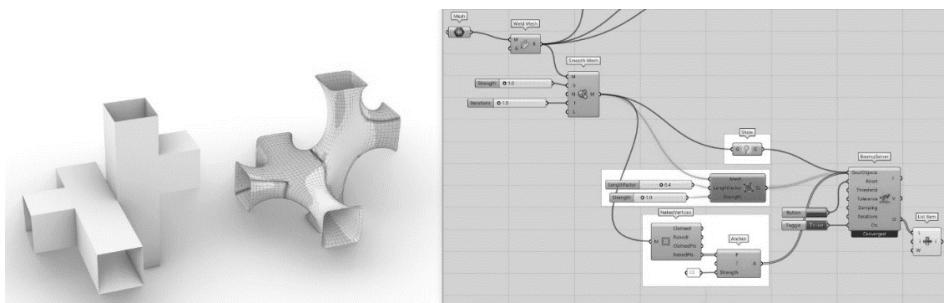
9.4. Interactive Form-finding: Kangaroo Physics

Kangaroo adalah mesin fisik (Physics Engine) yang dapat digunakan untuk melakukan berbagai simulasi yang melibatkan beban dan gaya (tarik, tekan) seperti simulasi elastisitas, simulasi *rigid body* pada obyek titik, kurva, bidang (mesh). Program ini dikembangkan oleh Daniel Piker, sekarang bekerja untuk Foster & Partners, London. Salah satu fitur yang akan digunakan adalah mengkonstruksi obyek organic dengan input berupa obyek berbasis mesh. Cara kerja Kangaroo Physics secara ringkas sebagai berikut:

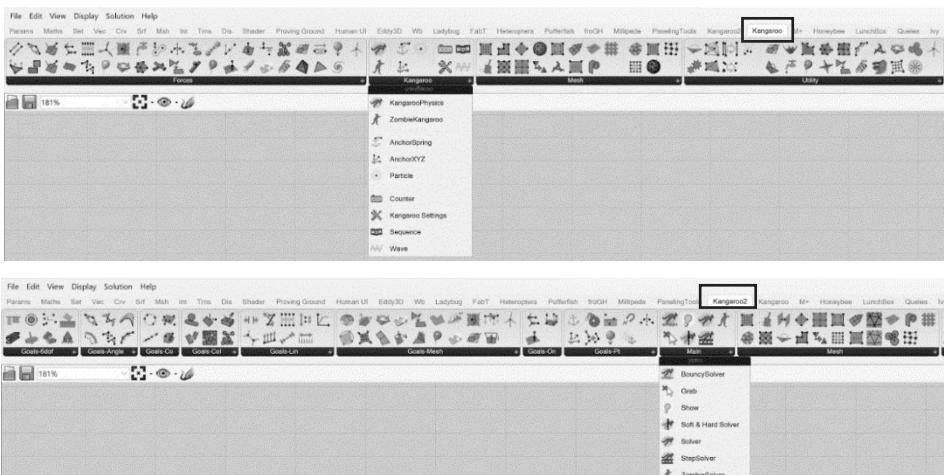
1. **Solver;** ini adalah komponen simulasi. Karena simulasi adalah simulasi dinamis, maka hasil simulasi (bentuk final) ditentukan setelah komponen ini berstatus *Converged*, yang artinya, bentuk final sudah stabil. Komponen ini memerlukan beberapa input pada bagian **Goal Objects**, **Reset** (memerlukan komponen Button), **On** (memerlukan komponen Boolean Toggle untuk Start (True) simulasi). Hasil komponen ini yang penting adalah O: Object.

2. Agar komponen Solver dapat bekerja, yang paling utama adalah komponen-komponen yang menjadi Goal Objects atau Forces (v.1), komponen-komponen ini inputnya adalah mesh hasil Weld Mesh dan Smooth Mesh (komponen dari MeshEditTools):

- Show: komponen menampilkan proses simulasi terhadap obyek.
- Edge Length: komponen ini menentukan Length Factor dari setiap edge pada obyek mesh. Jika Length Factor <1 maka Panjang setiap edge akan berkurang dari panjang awal. Strength adalah parameter kekuatan Length Factor.
- Anchor: komponen ini akan menetapkan angkur berupa titik-titik/ vertices yang ditentukan melalui komponen Naked Vertices. Strength adalah kekuatan parameter Anchor.



Anda sebaiknya mendownload dua versi Kangaroo: Kangaroo Physics dan Kangaroo 2 yang jika keduanya sudah diinstalasi, akan menampilkan dua tab yang berbeda.



Kangaroo 2 merupakan pengembangan dari Kangaroo (v.1) dimana ada empat kelompok fitur yakni: Goals, Main, Mesh dan Utility. Sementara di Kangaroo v.1 (Kangaroo Physics) terdapat fitur: Forces, Kangaroo, Mesh dan Utility. Komponen-komponen di Kangaroo v.1 **hanya bisa bekerja** dengan **Solver** di Kangaroo v.1 dan juga demikian untuk Kangaroo v.2. Hanya beberapa komponen yang kompatibel. Berikut adalah beberapa contoh definisi untuk mengetahui dan memahami fungsi dan algoritma *form finding* menggunakan Kangaroo Physics dan fitur-fitur pada v.1.

Kangaroo Physics (v.1):

1. **Forces:** koleksi dari komponen-komponen yang menyebabkan suatu obyek berperilaku tertentu, terdeformasi, bergerak dan sebagainya.
 2. **Kangaroo:** komponen solver, menghitung dan mensimulasikan semua forces dan menghasilkan geometri final.
 3. **Mesh:**
 4. **Utility:** komponen-komponen static untuk memodifikasi atau mengedit obyek yang tidak langsung berhubungan dengan simulasi.

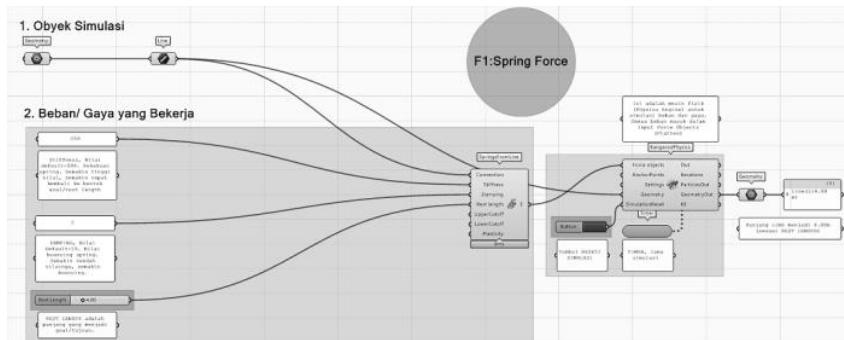
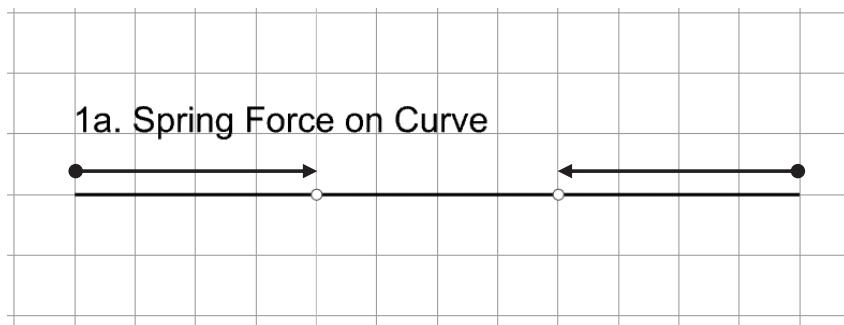
Kangaroo 2:

1. Goals: komponen-komponen serupa dengan Forces yang menyebabkan objek berperilaku tertentu.
 2. Main: komponen Solver untuk melakukan simulasi.
 3. Mesh:
 4. Utility: komponen-komponen statik yang fungsinya kurang lebih sama dengan Utility di Kangaroo v.1

9.4.1. Gaya (Forces) dan Hukum Hooke

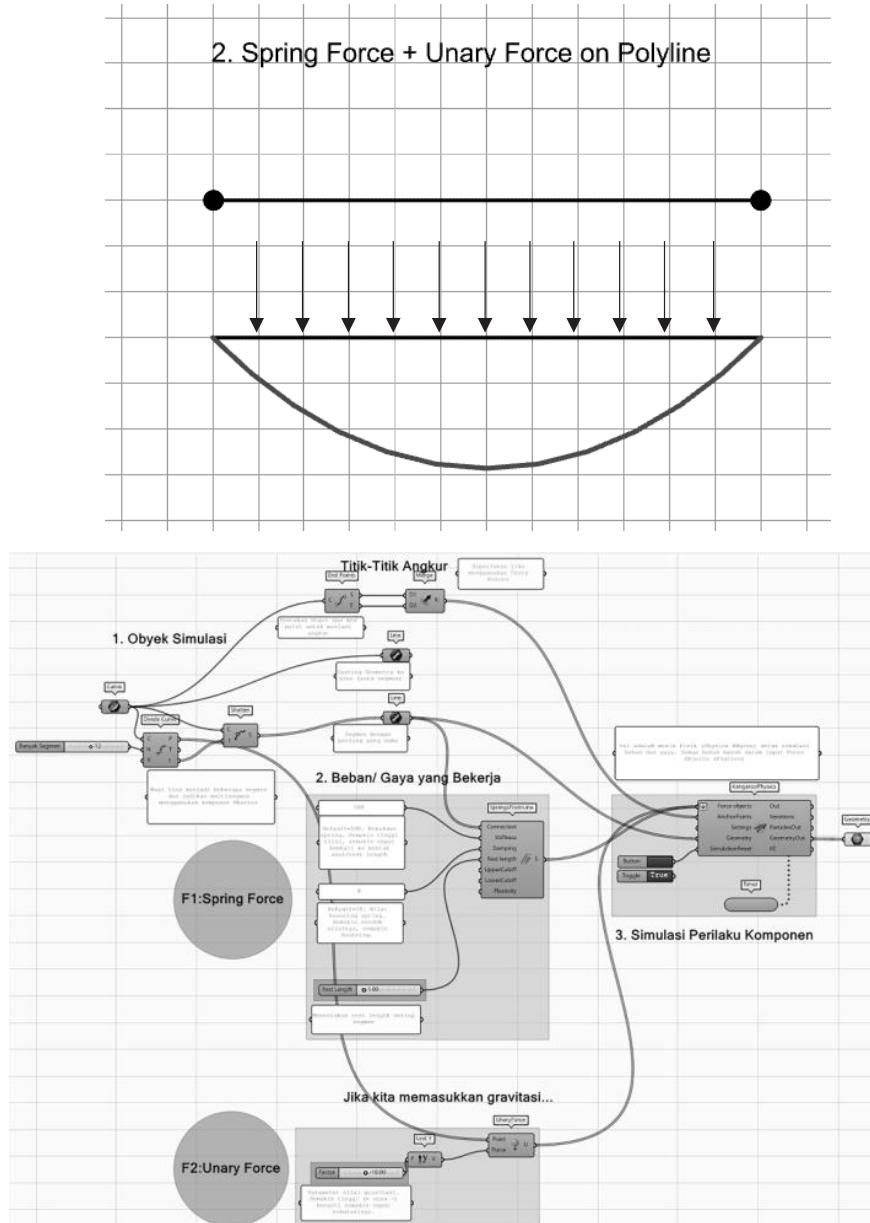
Berikut adalah beberapa contoh dasar untuk memahami bagaimana Kangaroo Physics bekerja.

1. Spring Force on Curve



- Simulasi yang menggambarkan elastisitas sebuah kurva sehingga menghasilkan panjang final.
- Sebuah Curve dengan Panjang 12 unit (contoh di atas).
- Komponen Spring From Line (v.1) dan Kangaroo Physics (v.1).
- Paramater Rest Length adalah nilai panjang final Curve yang diinginkan setelah simulasi Spring dijalankan.
- Timer: mengatur berapa lama durasi simulasi (ms).

2. Spring Force + Unary Force on Curve



- Simulasi menggambarkan elastisitas sebuah kurva jika mendapatkan gaya tidak hanya *Spring* (elastis) melainkan juga *Unary Force* (gaya searah pada suatu sumbu, misalnya simulasi beban gravitasi).
- Sebuah *Curve* dibagi menjadi titik-titik segmen dengan **Divide Curve**. Setiap segmennya didefinisikan sebagai segmen garis (Line) menggunakan *Shatter* yang mana parameter *C* adalah *Curve* dan parameter *t* adalah hasil *t (list)* dari *Divide Curve*. Masukkan sebagai input *Geometry* pada komponen **Kangaroo Physics**. Ini adalah parameter yang berfungsi untuk menampilkan geometri Ketika simulasi berlangsung.
- Tentukan dua titik angkur yakni *Start Point* dan *End Point* dari *Curve*, masukkan sebagai input *Anchor Points* pada komponen **Kangaroo Physics**.
- Komponen **Spring** digunakan untuk menampilkan simulasi elastisitas. Hasil dari *Shatter* dimasukkan dalam parameter *Connection*. Parameter *Rest Length* untuk menentukan berapa besar rasio kontraksi panjang setiap segmen yang dimasukkan dalam *Connection*.
- Komponen **Unary Force** memerlukan input *Force* berupa direction vector dimana bisa gunakan **Unit Y** dan masukkan parameter besar gaya.

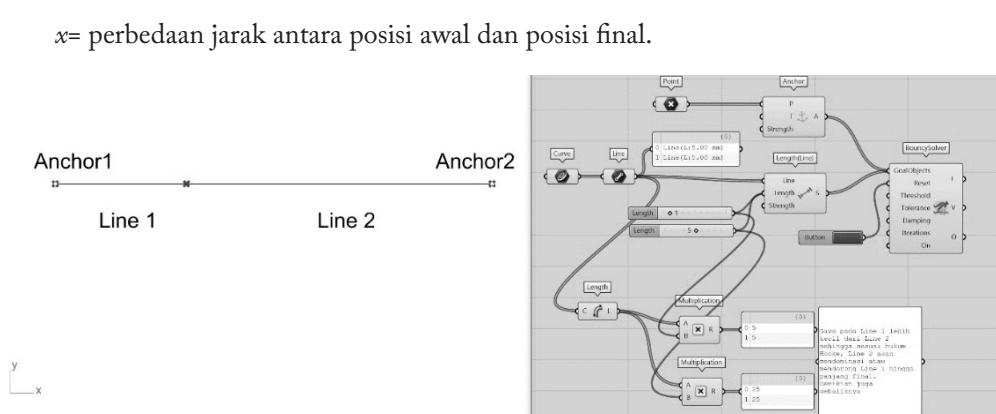
Pertanyaan: Mengapa kita masukkan Unit Y di sini? Mengapa tidak Unit Z?

- Semua hasil dari komponen beban dimasukkan dalam parameter *Force Objects* di **Kangaroo Physics (Flatten)**.

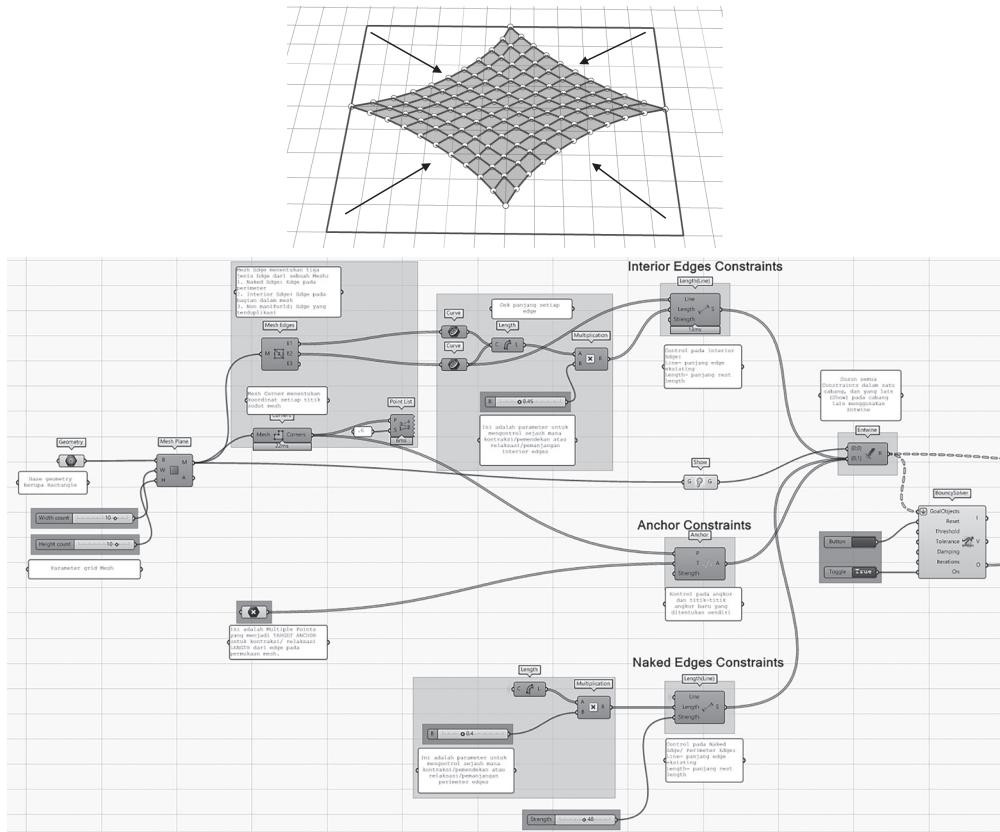
Hukum Hooke

Algoritma Kangaroo pada intinya menggunakan prinsip dari Hukum Hooke yang menyatakan besarnya gaya adalah proporsional dan berbanding lurus dengan jarak pergerakan pegas/obyek elastis dari posisi awal, atau: $F=-x \cdot k$

k = konstanta, atau jika di Kangaroo adalah parameter *Strength* pada komponen *Length*.
 x = perbedaan jarak antara posisi awal dan posisi final.



9.4.2. Pemodelan dan Simulasi Membran



1. Membran dapat dibuat melalui bentuk dasar **Rectangle** yang dikonversi menjadi **Mesh Plane** dan kemudian ditentukan komponen-komponennya menggunakan beberapa komponen **Goals: Line, Anchor**.
2. Komponen-komponen untuk membuat simulasi membran elastis adalah:
 - A. Mesh Edge:** yang akan menghasilkan: 1) *Naked Edge*, edge yang berada di pinggir mesh; 2) *Interior Edge*, edge yang berada di dalam obyek mesh. Keduanya akan menjadi obyek atau input komponen **Length** yang akan menjadi komponen simulasi untuk elastisitas (kontraksi atau ekspansi). Ingat, komponen **Length** hanya akan melakukan simulasi pada obyek **Line**.
 - B. Anchor:** yang akan mengunci titik-titik angkur dari obyek **Mesh Plane** atau titik-titik yang kita tentukan sendiri sebagai titik-titik angkur. Jika menggunakan obyek **Mesh Plane**, maka titik-titik angkur dapat dihasilkan melalui komponen **Mesh Corner**. Titik-titik angkur ini dapat kita definisikan sendiri untuk menghasilkan bentuk membran yang kita inginkan, jika kita menentukan titik-titik **Target** (parameter pada **Anchor**).
 - C. (Opsional) Edge Length:** adalah komponen yang menentukan elastisitas sebuah *mesh* melalui input *Length Factor* yang secara otomatis akan mendeteksi *Naked Edge* dan *Interior Edge* dari suatu *mesh*. Namun dengan komponen ini, anda tidak memiliki kontrol parametrik terhadap *Naked Edge* atau *Interior Edge*.

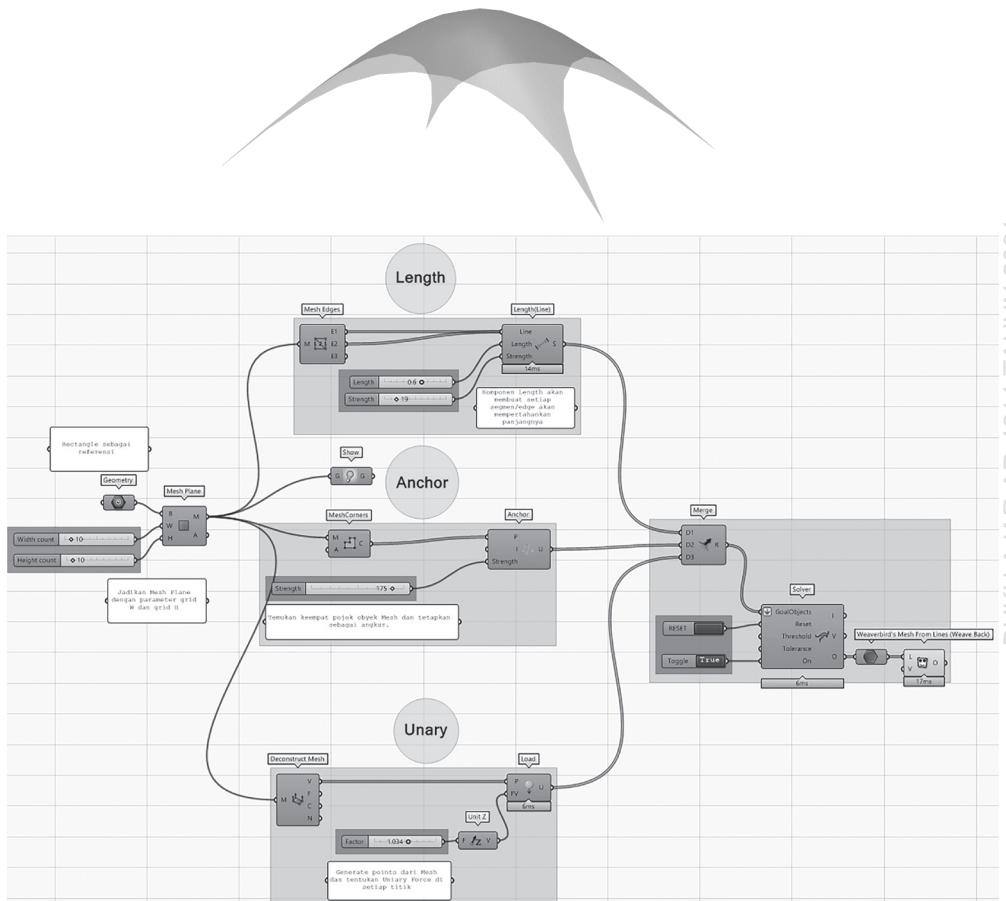
D. **Bouncy Solver:** mesin simulasi.

E. **Show:** komponen yang menampilkan proses simulasi pada obyek yang ditentukan.

1. Bentuk Catenary

Bentuk Catenary adalah kurva yang dihasilkan oleh obyek elastis atau rangkaian obyek-obyek akibat bebananya sendiri dan hanya ditopang oleh angkur-angkur pada sisi-sisinya.

Kita dapat membuat bentuk Catenary berupa bidang yang mana, jika nilai beban Unary Force adalah (-) maka bidang itu akan melendut sesuai beban gravitasi, sebaliknya, jika Unary Force bernilai (+) maka bidang tersebut mengembung.

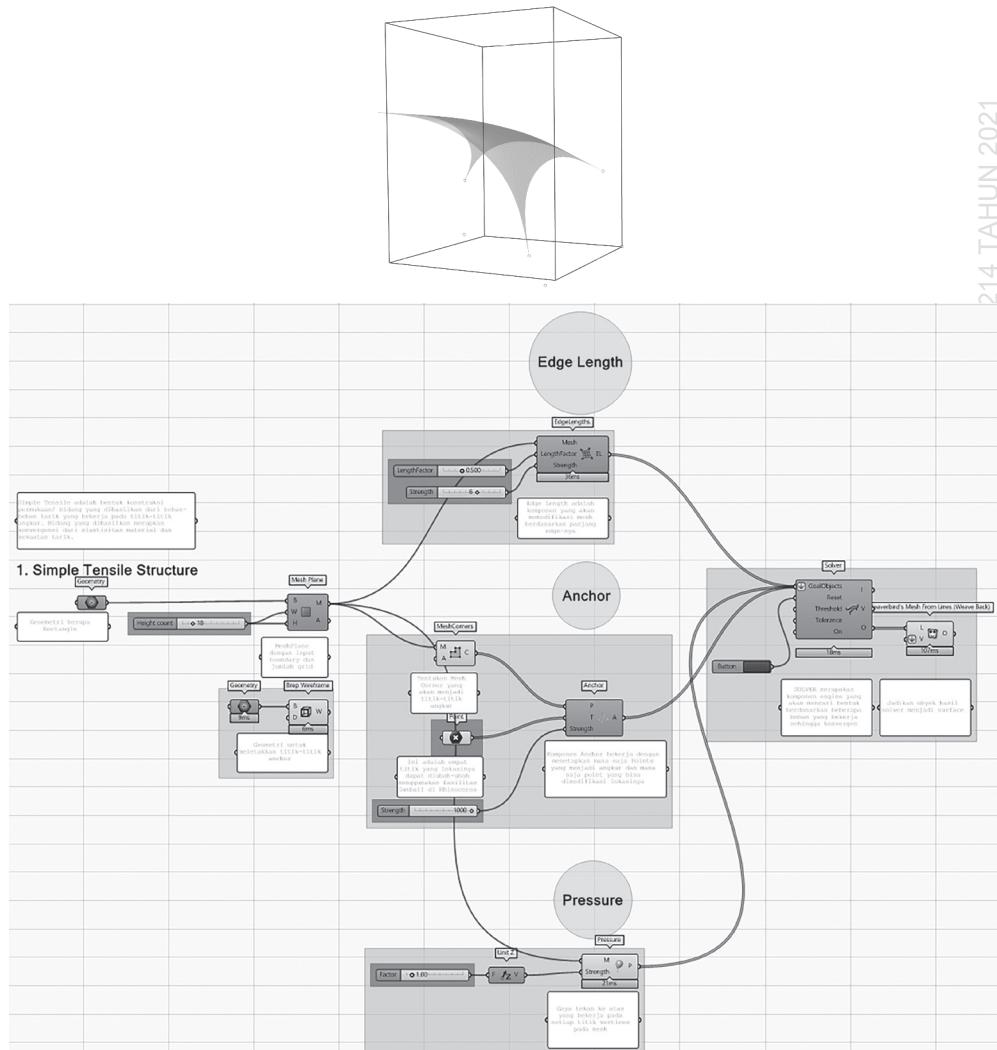


- Tentukan Geometri berupa Rectangle, konversi ke Mesh Plane dan tentukan parameter grid u dan v .
- **Length Elasticity:** gunakan komponen Length untuk melakukan simulasi elastisitas (kontraksi panjang atau ekspansi panjang) dari *edge* pada obyek Mesh Plane. Input dari komponen ini adalah Line yang didapat dari ekstraksi setiap *edge* obyek Mesh Plane menggunakan komponen Mesh Edge (yang diperlukan adalah Naked Edge dan Interior Edge). Hasil dari komponen ini adalah salah satu input Goal Objects pada Solver (v.2).

- **Anchor:** menentukan titik-titik angkur yakni keempat titik sudut obyek Mesh Plane. Cara menemukan titik-titik sudut menggunakan komponen **Mesh Corner**.
- **Unary Force:** simulasi gravitasi (Unary Force pada sumbu Z) menggunakan input *Points*. Points ini didapatkan dari mengekstraksi semua titik-titik *vertices* pada obyek Mesh Plane menggunakan komponen **Deconstruct Mesh**.
- Kombinasi dari parameter strength (Length dan Anchor), besarnya parameter *Factor* pada Unit Z akan mempengaruhi bentuk final.
- *Pertanyaan: Jika base geometry bukan berupa Rectangle, bagaimana membuat mesh-nya?*

2. Tensile-base Form-finding #1

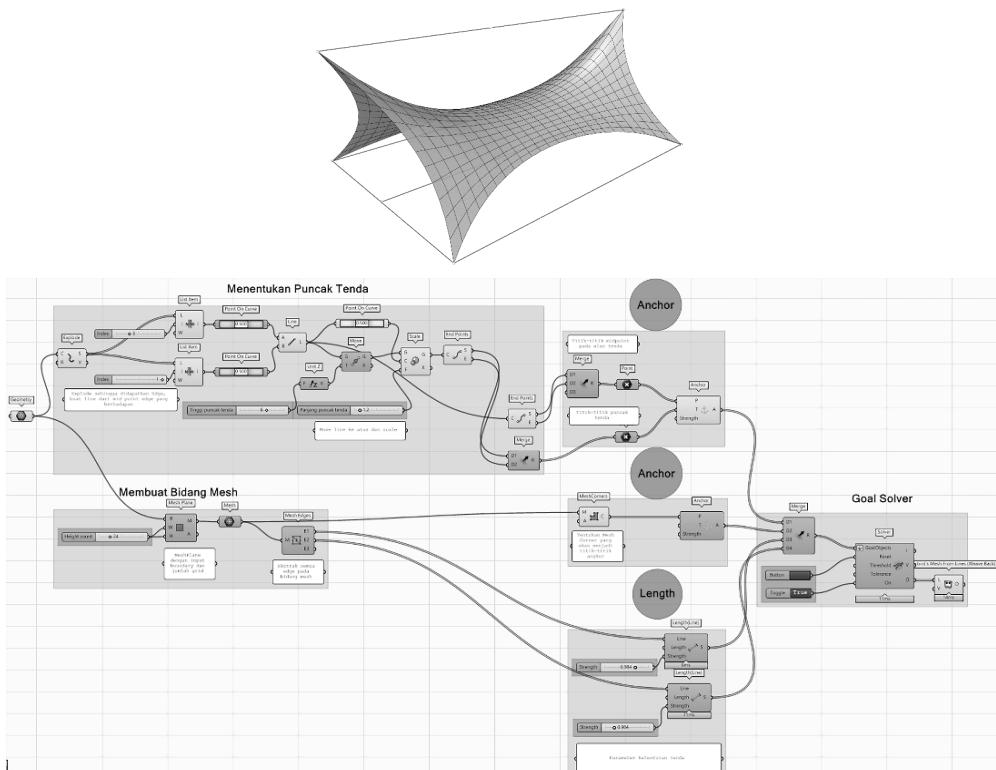
Selain tensile menerapkan prinsip Catenary, bentuk-bentuk berbasis tensile yang lain dapat juga dicari' menggunakan komponen-komponen dalam Kangaroo (baik v.1 maupun v.2) yang menerapkan gaya-gaya, deformasi/transformasi pada edge dan angkur.



- Bentuk tenda yang didapatkan dari bidang mesh yang di-stretch mengikuti posisi empat titik angkur.
- Komponen Edge Length (v.2) digunakan untuk kontraksi/ekspansi Panjang edge.
- Komponen Anchor, parameter P: titik-titik pojok pada Mesh Plane; T: Target, titik-titik target yang merupakan titik-titik angkur baru.
- Pertanyaan: Bagaimana kalau kita menambahkan Unary Force? Apakah komponen itu bisa digunakan di sini? Apa komponen penggantinya untuk membuat simulasi gaya-gaya searah?

3. Tensile-base Form-finding #2

Contoh berikut adalah pengembangan definisi *form-finding* menggunakan *Target Points* pada komponen **Anchor**. Perlu diingat, parameter *P* pada komponen **Anchor** harus terletak pada plane yang sama dengan obyek *mesh*. Sedangkan parameter *T* atau *Target* merupakan titik-titik yang terletak di luar bidang/plane obyek *mesh*.

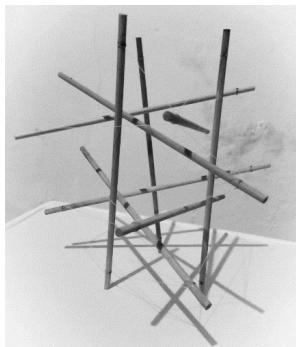


- Obyek dasar adalah **Rectangle**. Di sini bentuk tenda yang dihasilkan bergantung pada titik-titik angkur dan titik-titik deformasinya (*Target*). Cari *Midpoint* dari dua *edge* yang berhadapan dari obyek **Rectangle**. Kedua titik ini yang terletak pada bidang/plane **Rectangle** tersebut merupakan input untuk parameter *P* dari komponen **Anchor**.
- Buat **Line** dari dua titik tadi, pindahkan ke atas (sumbu *Z*) dan skalakan (ubah Panjang/pendek **Line**), tentukan titik-titik *Start Point* dan *End Point* dari garis ini. Ini akan menjadi titik-titik input untuk parameter *T* (*target*) dari komponen **Anchor**.

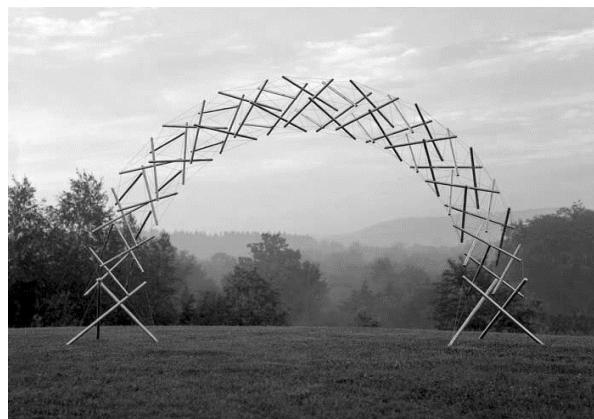
- Buat Mesh Plane dari obyek Rectangle, tentukan Mesh Corners dan gunakan ini sebagai input untuk parameter *P* dari komponen Anchor kedua.
- Tentukan *Naked Edges* dan *Interior Edges* dari obyek Mesh Plane menggunakan komponen Mesh Edge. Masing-masing *Edge* ini merupakan input untuk komponen Length. Parameter *Strength* dari masing-masing komponen Length menentukan besarnya kontraksi dari *Edge*.
- Gabungkan semua input dari dua komponen Anchor dan dua komponen Length dan jalankan simulasi.

9.5. Pemodelan dan Simulasi Tensegrity

Tensegrity adalah sebuah sistem konstruksi rangka dimana suatu rangkaian komponen tekan yang diskontinyu bertemu atau berinteraksi dengan rangkaian komponen tarik yang kontinyu sedemikian rupa sehingga keduanya membentuk sebuah konstruksi volumetrik ruang yang stabil.



Gambar 197. Model Tensegrity. (Sumber Foto: Dr.-Ing Andry Widyowijatnoko)

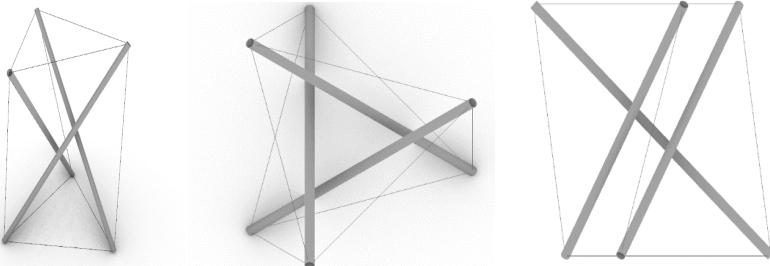


Gambar 198. Rainbow Arch (Sumber: Kenneth Snelson, 2001)

Kangaroo dapat digunakan sebagai perangkat untuk mendesain dan melakukan simulasi Tensegrity.

Yang perlu diperhatikan dalam pemodelan sistem Tensegrity adalah penentuan segmen/Line yang menjadi komponen *Tension* (Tarik), yang merupakan elemen kabel dan segmen/Line yang menjadi komponen *Compression* (Tekan), yang merupakan elemen batang.

9.5.1. Pemodelan Modul Tensegrity Sederhana



Gambar 199. Modul Tensegrity Prisma Segitiga (kika: 3D; Tampak Atas; Tampak Samping)

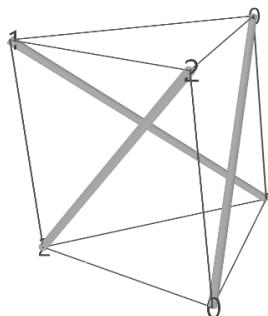
Modul tensegrity sederhana dapat dibuat dengan prisma tiga arah dengan basis segitiga selain beberapa modul unit lain¹¹.

Pada pemodelan Tensegrity ini beberapa prinsip yang diperhatikan adalah:

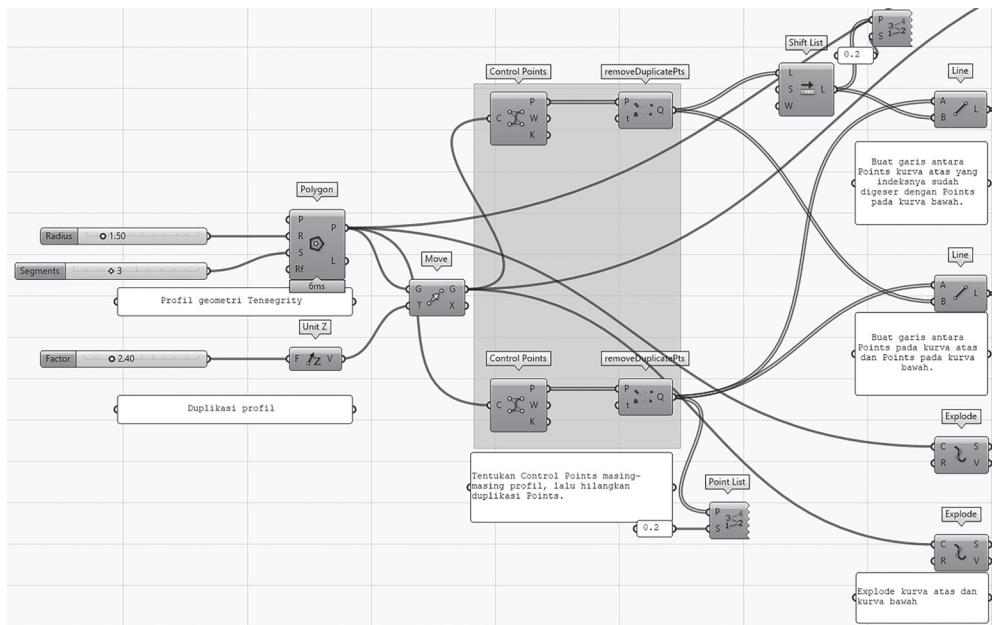
1. Semua vertices (titik sudut) tidak ada yang terduplikasi.
2. Semua kurva yang akan menjadi komponen tarik (kabel) harus kontinyu.
3. Semua kurva yang akan menjadi komponen tekan (batang) harus diskontinyu.

Kita dapat mendesain modul tensegrity beserta parameter kontrol dan simulasi fisik di Grasshopper, atau membuat konfigurasi batang tekan dan kabel tarik di Rhinoceros, lalu kemudian membuat simulasi fisik dengan Kangaroo di Grasshopper.

Langkah 1: Persiapan geometri, penentuan batang tekan dan kabel tarik

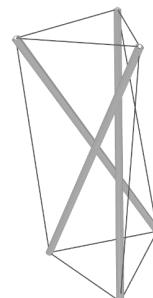


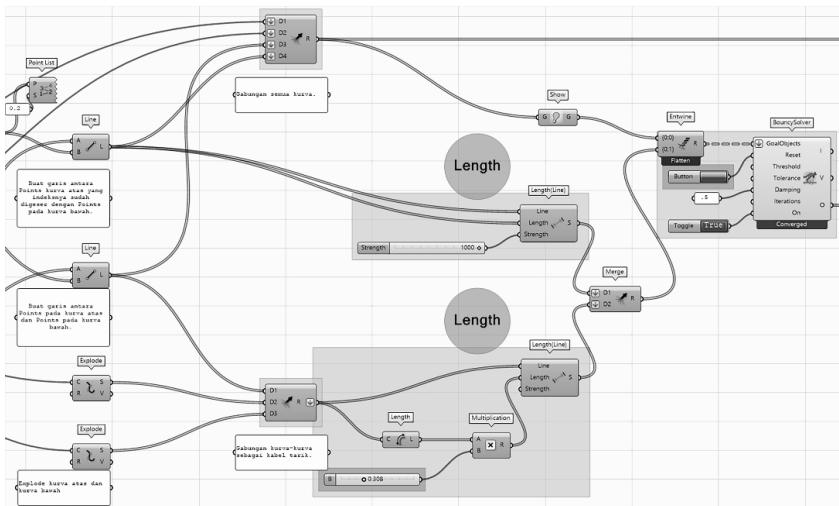
¹¹ Kennethsnelson.net



- Buat **Polygon** dengan tiga sisi, lalu duplikasi ke sumbu Z dengan parameter ketinggian.
- Tentukan **Control Points** masing-masing **Polygon**, hilangkan duplikasi *control points* dengan komponen **removeDuplicatesPts** (Kangaroo) atau **Cull Duplicates**.
- Dari masing-masing **Polygon** yang diketahui index verticesnya, kita membuat dua kelompok **Line**:
 - Lines** yang merupakan batang tekan, menghubungkan antar vertices dengan salah satu list vertices dikenai **Shift List** agar nilai indeks bergeser (lihat garis hijau).
 - Lines** yang termasuk kabel tarik, menghubungkan antara vertices pada **Polygon** atas dan vertices pada **Polygon** bawah (garis-garis merah vertikal).
- Sampai proses ini kita sudah menentukan aspek penting dari model tensegrity: penentuan batang-batang tekan dan kabel-kabel tarik. Seperti terlihat di gambar, batang-batang tekan adalah yang berwarna hijau, kabel-kabel tarik adalah semua garis yang berwarna merah.

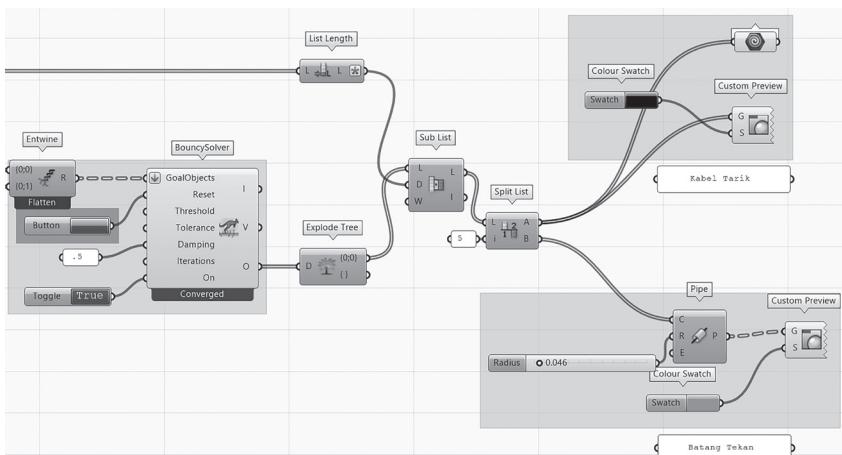
Langkah 2: Penentuan beban/gaya dan simulasi





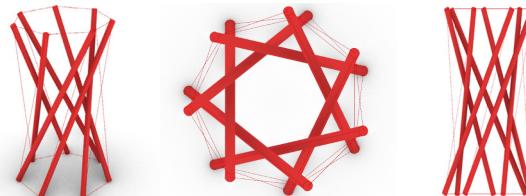
- Setelah semua batang tekan dan semua kabel tarik ditentukan, maka kita akan menggunakan jenis *Goals* atau *Forces* dari Kangaroo yakni: **Length**. Prinsip dari *Length Goal* adalah melakukan kontraksi (memendekkan) atau relaksasi (memanjangkan) obyek Lines dari panjang asalnya.
- Komponen **Length** pertama kita gunakan pada semua batang tekan, dimana parameter input Length yang merupakan parameter *target length* diisi dari semua batang tekan. Logika di sini adalah, batang-batang tekan ditentukan untuk tidak mengalami kontraksi maupun relaksasi, artinya panjangnya harus tetap.
- Komponen **Length** kedua kita gunakan untuk semua garis yang menjadi kabel tarik. Kita dapat membuat parameter pengali untuk menentukan sependek apa *target length* yang kita tentukan. Misalnya kita bisa menentukan bahwa semua panjang kabel tarik, *target length* adalah 0.5 kali panjang asalnya (seperti pada contoh di atas). Logika di sini adalah, masing-masing kabel tarik diset untuk berkontraksi 0.5 kali panjang asalnya.

Langkah 3: Pengorganisasian data keluaran simulasi



- Keluaran data hasil simulasi Kangaroo dapat dikelompokkan menjadi kurva tarik (kabel) dan kurva tekan (batang).
- Keluaran simulasi diseleksi menggunakan **Sub List** dengan domain sesuai panjang list (List Length) dari komponen **Merge** yang merupakan data semua komponen baik tekan maupun tarik.
- Penggunaan **Split List** untuk memisahkan kurva yang merupakan kabel tarik dan kurva yang merupakan batang tekan.

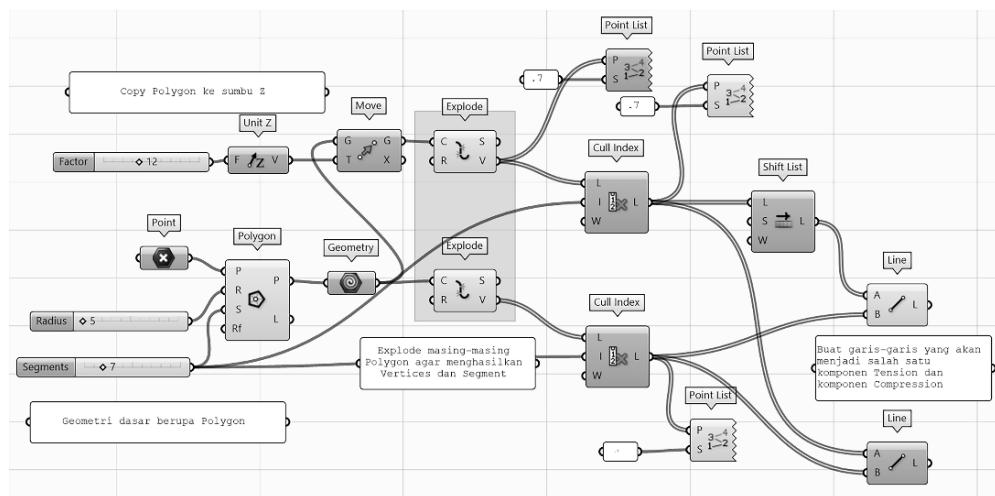
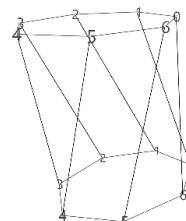
9.5.2. Pemodelan Unit Tensegrity Berbasis Heksagon



Gambar 200. Modul Tensegrity Berbasis Heksagon (kika: 3D; Tampak Atas; Tampak Samping)

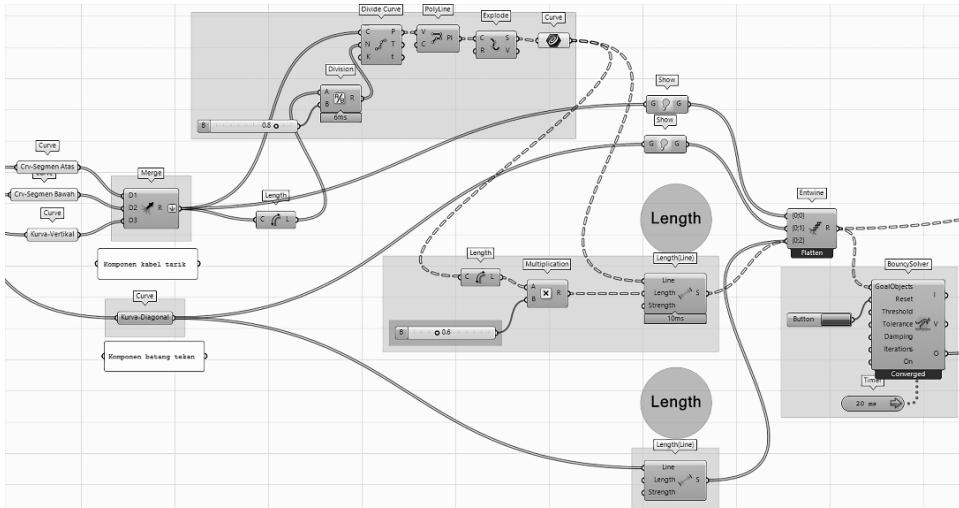
Selain dengan basis segitiga, model tensegrity lainnya adalah berbasis kurva heksagon dengan prinsip dan algoritma pembuatan yang serupa dengan sebelumnya.

Langkah 1: Persiapan geometri, penentuan batang-batang tekan dan kabel-kabel tarik



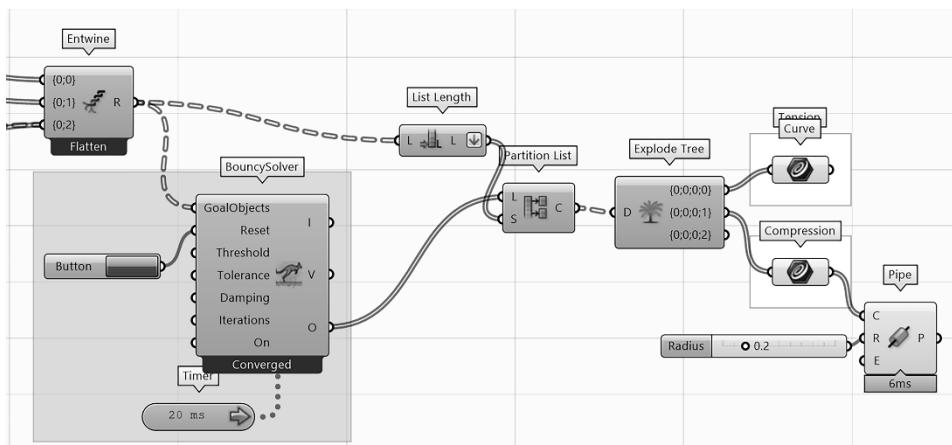
- Prinsip penentuan batang-batang tekan dan kabel-kabel tarik sama dengan proses pembuatan modul tensegrity sebelumnya, yakni dengan menggunakan beberapa komponen: **Explode**, **Cull Index** dan **Shift List**.
- Komponen **Line** digunakan untuk membuat garis-garis yang akan ditentukan sebagai batang-batang tekan maupun kabel-kabel tarik.

Langkah 2: Penentuan beban-beban (Goals/Forces) pada masing-masing kelompok garis



- Pada elemen *tension* (kabel), pastikan kita memiliki input Length pada komponen Length yang merupakan Length Ratio/ nilai kontraksi dari kabel-kabel *tension*.
- Pada elemen *compression* (batang), parameter input Length pada komponen Length tidak memiliki nilai *Length Ratio* (dibiarkan tanpa input), karena batang-batang akan kita buat tidak mengalami kontraksi maupun relaksasi.

Langkah 3: Pengorganisasian keluaran simulasi



- Geometri hasil simulasi dipisahkan agar kita dapat memodifikasi kurva batang (menggunakan Pipe untuk menambah ketebalan). Komponen yang digunakan adalah: **Explode Tree**.

Langkah 4: Meningkatkan kualitas simulasi- *Discretizing/ Subdividing Elements*

Sampai pada Langkah 3, kita sudah dapat membuat simulasi tensegrity dimana elemen-elemen tarik/*tension* mengalami kontraksi hingga mencapai kondisi ekuilibrium dengan elemen-elemen tekan/*compression*. Namun pada Langkah ini, semua elemen tarik berupa kabel-kabel, disimulasikan dengan menggunakan satu satuan panjang pada setiap elemen kabel tersebut.

Agar simulasi lebih realistik, kita perlu membagi setiap segmen pada elemen-elemen tarik menjadi beberapa segmen.

- Gunakan **Divide Curve** untuk membagi segmen elemen tarik.
- Gunakan komponen **Polyline** untuk menggabungkan titik-titik **Divide Curve** menjadi **Polyline** dan gunakan **Explode** untuk mengubahnya menjadi **Curve/segmen**.

9.6. Pemodelan dan Simulasi Bending Active

Struktur Bending-Active adalah struktur yang dirangkai dari batang-batang (*rod*) yang memiliki karakteristik *bending* atau melengkung sesuai dengan gaya yang ditimpakan kepadanya. Bentuk akhir yang didapatkan adalah hasil dari lengkung yang merupakan resultan dari sudut-sudut yang dihasilkan dari setiap segmen dari batang tersebut. Disebut *active* karena bentuk akhir merupakan transformasi akibat gaya yang bekerja pada batang tersebut.

Contoh batang yang memiliki karakteristik *bending* adalah bamboo. Contoh aplikasi dari struktur *bending-active* adalah: gridshell, tenda, hybrid: bending active + tensile.



Gambar 201. Hybrid Tower oleh CITA (Center of Information Technology and Architecture), The Royal Danish Academy of Fine Arts, School of Architecture, Design and Conservation.

Dalam pemodelan *bending active*, yang menjadi obyek adalah **Line** dengan dua jenis **Goals:**

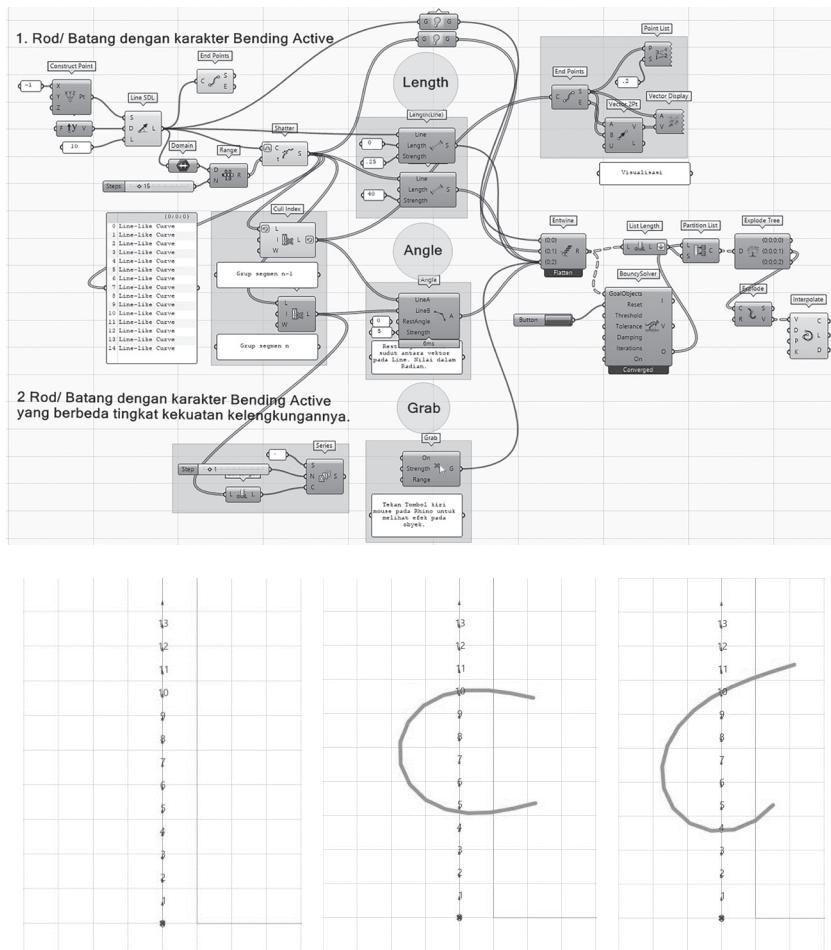
1. **Length:** untuk membuat garis terkontraksi (memendek) atau terrelaksasi (memanjang) terhadap panjang asal.

2. Angle: untuk membuat sudut (dalam Radian) antara dua buah garis.

Agar sebuah batang dapat disimulasikan untuk berperilaku bending ketika dikenai suatu gaya, maka yang perlu diperhatikan jika simulasi menggunakan Length dan Angle adalah:

1. **Goals Length** yang bekerja pada Line (1 unit panjang) memiliki parameter Length=0 karena objek Line akan berkontraksi maksimal untuk mencapai nilai panjang final=0.
2. **Goal Length** yang bekerja pada segmen dari Line (n unit panjang) memiliki parameter Length= panjang tiap segmen karena objek setiap segmen tidak berkontraksi melainkan akan bending.
3. **Goal Angle** bekerja pada segmen dari Line dimana besar sudut bending ditentukan pada pertemuan antara dua segmen.

9.6.1. Batang/rod Berkarakter Bending-active



Ki-ka: Inisiasi Batang/Rod yang terdiri dari beberapa segmen, Hasil simulasi bending dengan *Strength Angle* menggunakan konstanta (setiap segmen memiliki nilai

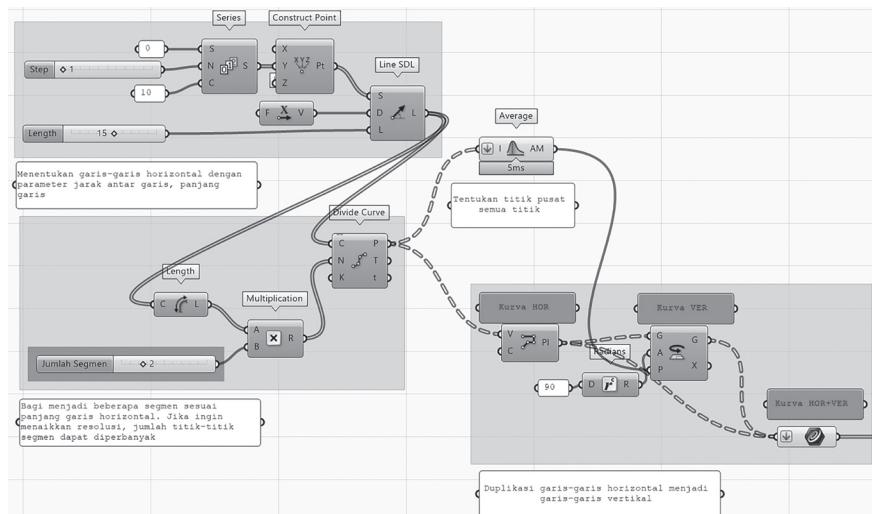
kelengkungan yang sama), Hasil simulasi bending dengan nilai *Strength Angle* berbeda berdasarkan komponen Series. Di sini kelengkungan masing-masing segmen berbeda sesuai dengan hasil dari komponen Series.

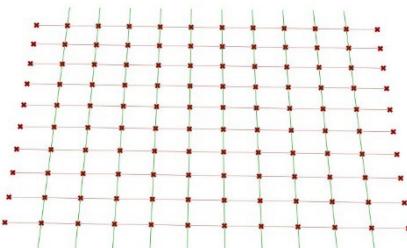
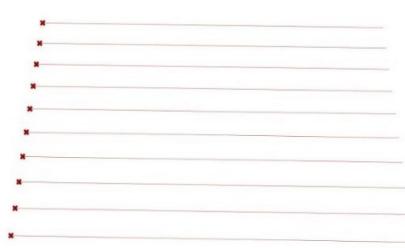
- Line dibagi menjadi beberapa segmen menggunakan komponen Shatter. Kita dapat membuat parameter kontrol banyaknya segmen. Panjang setiap segmen di sini adalah sama.
- Komponen goal Length yang pertama untuk obyek Line berfungsi agar garis ini berkontraksi maksimal (parameter Length=0).
- Komponen goal Length kedua untuk segmen hasil Shatter berfungsi agar segmen-semen ini panjangnya tetap dengan panjang asal, sehingga nilai parameter Length sama dengan obyek sagmen (atau bisa juga parameter ini tidak diisi).
- Agar komponen goal Angle berfungsi, maka segmen-semen hasil Shatter ini harus diduplikasi, dimana hasil duplikasi dibalik arah vektornya, dan pada masing-masing list segmen, segmen ujung (indeks=0) tidak menjadi obyek untuk goal Angle.
- Komponen Grab digunakan untuk berinteraksi dengan obyek simulasi. Kita dapat melakukan click-drag pada obyek di Rhinoceros untuk melihat simulasi *bending-active* dari batang tersebut.
- Jika sudut *bending* tidak sama, maka kelengkungan pada segmen-semen juga tidak sama. Cara menentukan ini adalah menggunakan Series dengan faktor pengali parametrik pada parameter angle di komponen goals Angle.

9.6.2. Pemodelan Gridshell

Pemodelan Gridshell di sini menggunakan basis *Rectangular* dimana titik-titik *Anchor* berada pada keempat sisinya. Karakter bending-active pada setiap segmen/edge ditentukan menggunakan Goals Rod, dan kelengkungan Gridshell pada area pusat ditentukan menggunakan Goals Load.

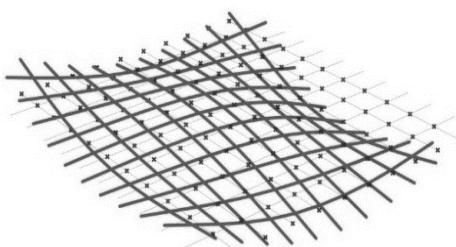
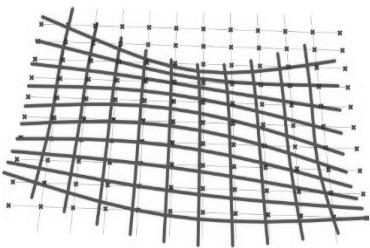
Langkah 1: Membuat grid dari batang-batang aktif





Ki-Ka: Inisiasi garis-garis horizontal dari points dan parameter panjang, duplikasi garis-garis vertikal dan menentukan titik-titik potong (*intersection*) garis-garis vertical dan horizontal.

- Buat rangkaian garis menggunakan **Series** dan **LineSDL**. Bagi menjadi beberapa segmen dengan **Divide Curve** menggunakan parameter kontrol banyaknya segmen. Buat **Polyline** baru hasil dari **Divide Curve**.
- **Rotate 90° Polyline** untuk menghasilkan garis-garis vertikal dengan posisi titik-titik segmen yang sama dengan garis-garis horizontal.
- Gunakan **Goals Rod**



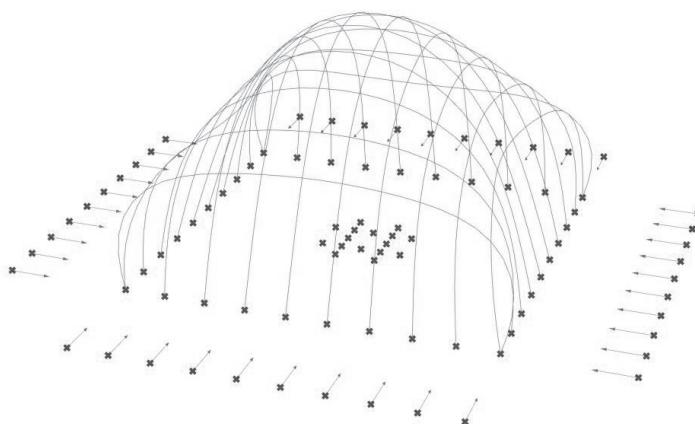
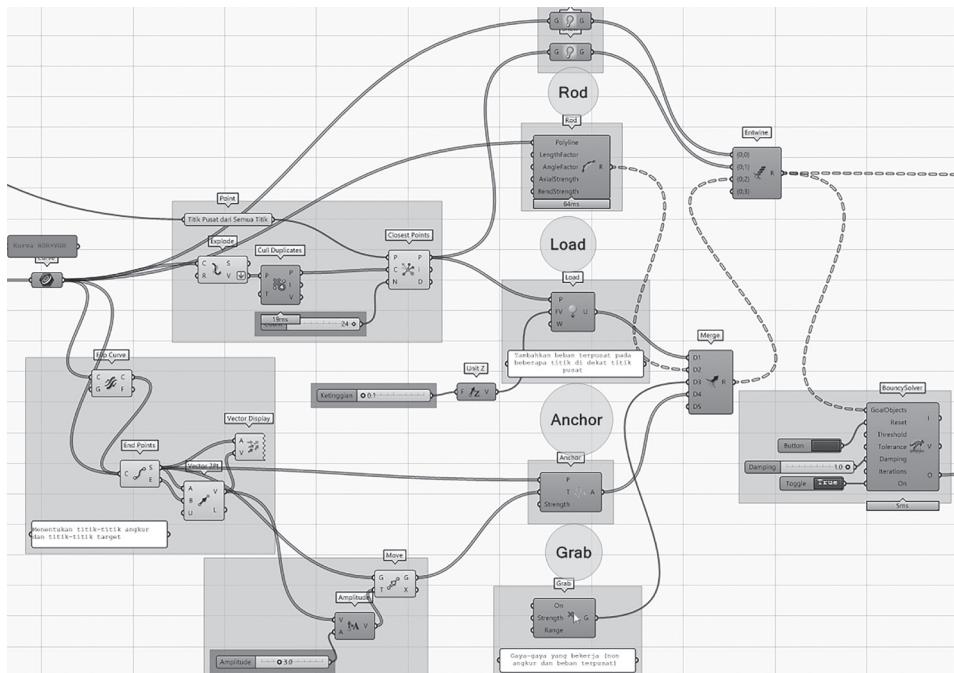
Hasil simulasi batang-batang yang membentuk gridshell

Sampai di langkah ini kita sudah membuat model karakteristik dan perilaku batang *bending-active*, perilaku Ketika batang-batang tersebut dirangkai menjadi sebuah gridshell. Selanjutnya, pada kontruksi gridshell, bentuk final yang dihasilkan didapatkan dari memindahkan titik-titik angkur pada sisi-sisi gridshell ke dalam (*inward*) sehingga rangkaian batang-batang akan mengembang ke atas.



Langkah 3: Menambahkan dan Mendefinisikan Angkur dan Beban Terpusat

Langkah kedua sebelumnya adalah memodelkan grid rods yang memiliki karakter *bending-active*. Selanjutnya kita mendefiniskan titik-titik angkur yang ‘mendesak’ batang-batang tersebut ke dalam dan juga memodelkan beban terpusat (gravitasi terbalik).



- Sampai tahap ini, anda dapat bereksperimen dengan: *bagaimana kalau yang dipindahkan tidak semua titik pada keempat sisi, tapi hanya pada dua sisi? Bagaimana kalau beban terpusat, kekuatannya (strength) gradual? Menggunakan komponen Series? dan pada titik-titik yang lebih banyak (penambahan titik-titik segmen)?*
- Di sini anda dapat melihat proses *form-finding* yang dihasilkan oleh simulasi fisik.
- Untuk melakukan PAUSE pada komponen simulasi **Bouncy Solver**, gunakan

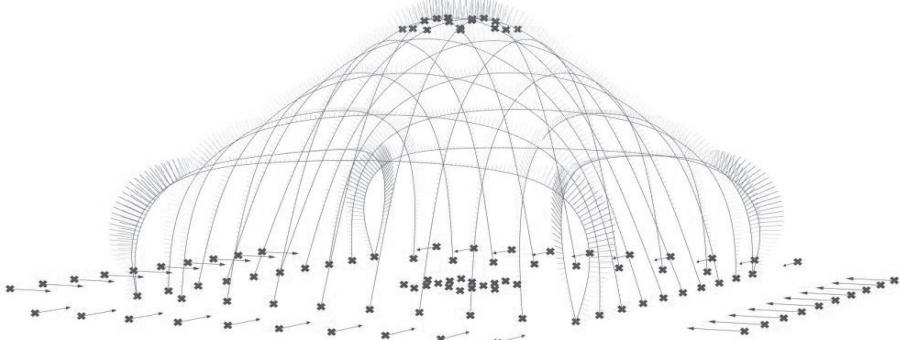
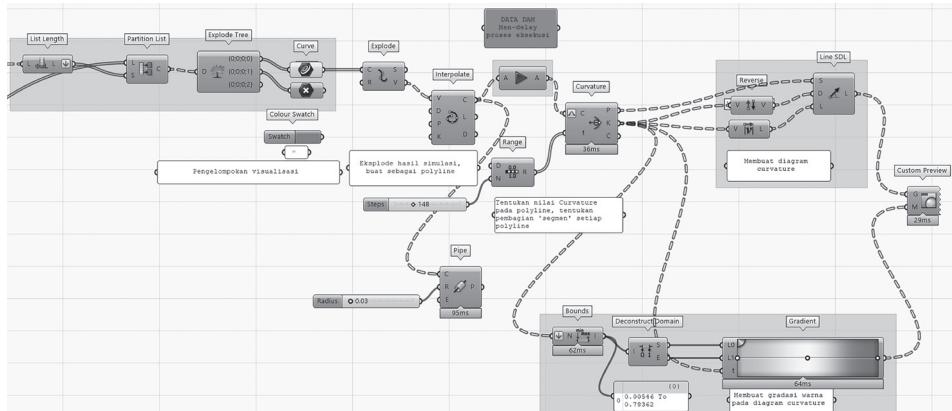
komponen Boolean Toggle pada parameter input : On.

Langkah 4: Analisis Kelengkungan Geometri

Sampai pada Langkah 3, kita melakukan pemodelan dan simulasi konstruksi system gridshell berdasarkan karakteristik material batang. Pada kenyataannya, material-material batang ini memiliki karakteristik *bending* yang berbeda-beda dan memiliki property kelengkungan yang berbeda. Properti kelengkungan ini dapat divisualisasikan pada setiap segmen pada suatu batang sehingga dapat digunakan untuk analisis selanjutnya.

Pada prinsipnya, nilai kelengkungan didefinisikan dalam:

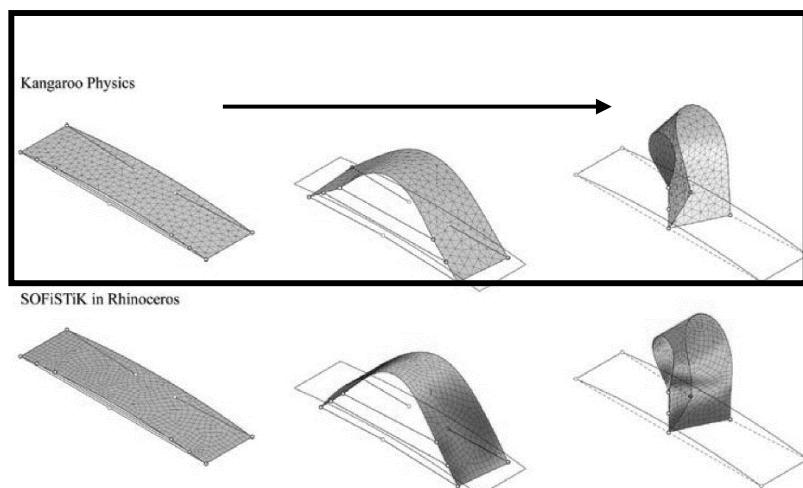
$k = \frac{1}{r}$ dimana, k = nilai kelengkungan; r =jari-jari lingkaran yang memiliki titik tangensial pada segmen yang dicari kelengkungannya.



- Pada Langkah 4 ini, akan ada dua proses utama yang akan menguras kemampuan perhitungan komputer: simulasi fisik yang menghasilkan bentuk final dan menghitung kelengkungan dari setiap segmen. Untuk mempercepat proses simulasi, kita bisa gunakan komponen **Data Dam** yang ditempatkan sebelum proses perhitungan dan pembuatan curvature. **Data Dam** ini fungsinya adalah men-*delay* arus eksekusi program.

9.6.3. Bending Active Plate

Pemodelan *bending-active* dimulai dari mendefinisikan karakteristik *bending* dari batang-batang (*rod*), kemudian menyusun batang-batang menjadi sebuah grid yang memiliki sifat *bending active* sehingga dapat digunakan untuk merancang sebuah system *gridshell*. Pengembangan selanjutnya adalah membuat dan memodelkan bidang (*mesh*) yang memiliki karakteristik *bending-active*. Pendekatan ini berangkat dari lembar planar (*flat sheet*) yang diberikan gaya tertentu sehingga berkontraksi elastis melalui elemen kabel atau elemen penyusun lembar tersebut yang memiliki karakter *bending-active*, sehingga menghasilkan bentuk final.

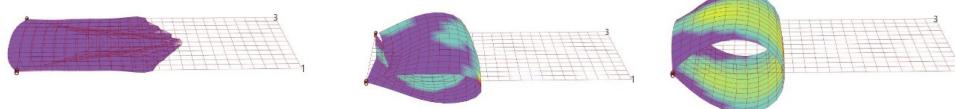
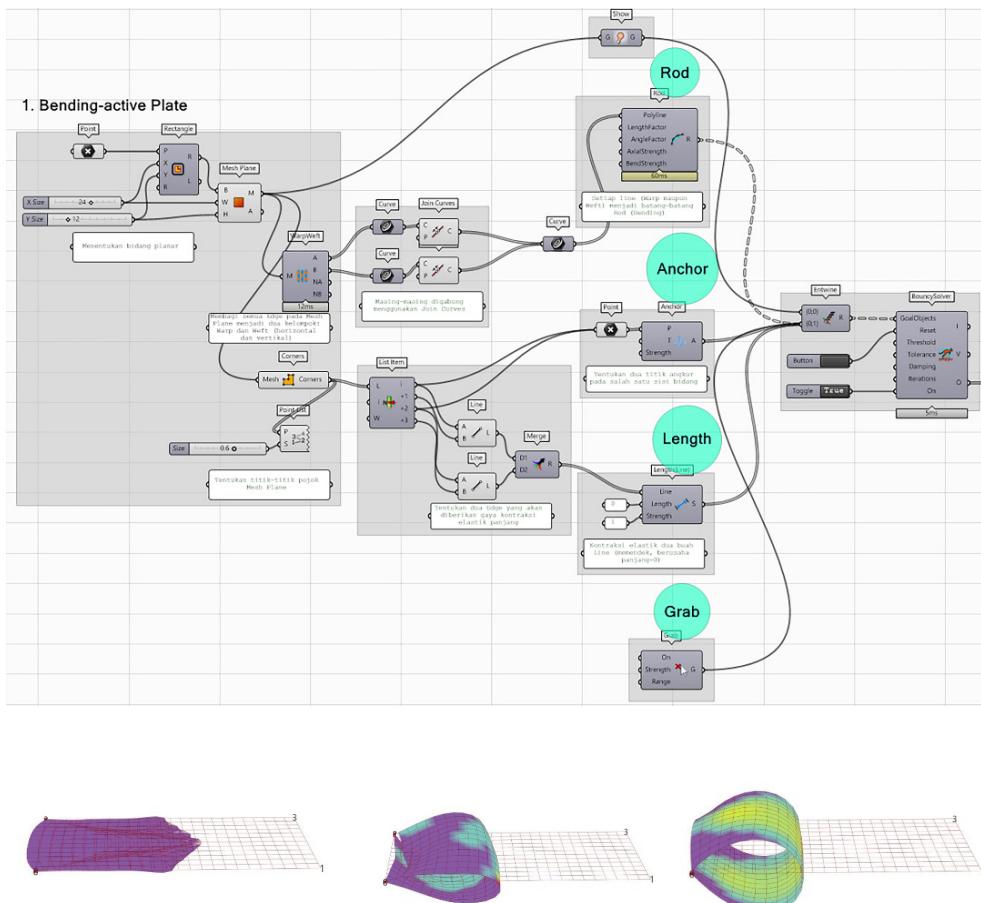


Gambar 202. *Bending-active Plate*. (Sumber: Simon Schleicher & Riccardo la Magna, ACADIA Proceeding 2016)



Gambar 203. *Bending-active Plate* ICD/ITKE Research Pavilion

Langkah 1: Membuat Bidang Mesh Berkarakter Bending-Active



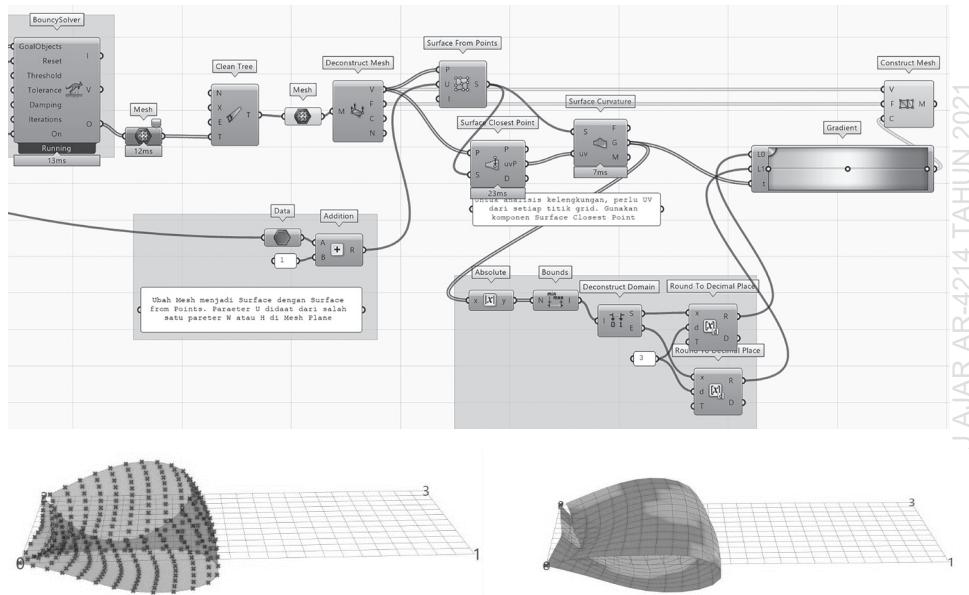
Ada tiga gaya yang dikenakan pada obyek: **Rod** : membuat semua edge pada obyek memiliki karakteristik kontraksi bending, **Anchor**: dua titik pada salah satu sisi bidang menjadi angkur, **Length**: dua garis/line pada bidang akan memendek atau berkontraksi.

- Obyek dibuat menggunakan **Mesh Plane** dengan parameter jumlah grid pada *W* dan *H*.
- Ekstrak semua edge dari **Mesh Plane** menggunakan **WarpWeft** (komponen Kangaroo 2). *Warp* dan *Weft* adalah istilah dalam *Weaving* (anyaman) dimana yang dimaksud dengan *Warp* adalah komponen yang diam sedangkan *Weft* adalah komponen yang bergerak naik-turun diantara komponen *Warp*. Pada konteks komponen **Warp Weft**, komponen ini akan membagi *edge* menjadi dua: Horisontal dan Vertikal. Masing-masing hasil dari komponen harus digabung dengan **Join Curves** agar menjadi garis.
- Garis-garis hasil komponen **Warp Weft** menjadi input parameter komponen **Rod**. Sampai tahap ini, **Mesh Plane** memiliki karakter *bending-active* melalui semua *edge*-nya.

Langkah 2: Menentukan Kelengkungan Bidang

- Agar bidang dapat melengkung pada salah satu sisi, yang harus diperhatikan adalah: angkur dan kontraksi panjang.
- Titik-titik angkur ditetapkan pada salah satu sisi, yakni dua titik pojok pada sisi pendek tersebut. Gunakan komponen **Mesh Corner** untuk ekstraksi titik-titik pojok dan **List Item** untuk menentukan dua titik pojok pada salah satu sisi bidang. Masukkan dua titik ini ke komponen **Anchor**.
- Dua sisi Panjang dari bidang, harus berkontraksi memendek ($Length=0$). Gunakan komponen **Length** dimana parameter input **Line** adalah dua **Line** dari sisi Panjang bidang **Mesh Plane**. Parameter **Strength** dari komponen **Length** sekitar 0.75-1.
- Komponen **Show** dan **Grab** digunakan untuk simulasi dan visualisasi proses simulasi.

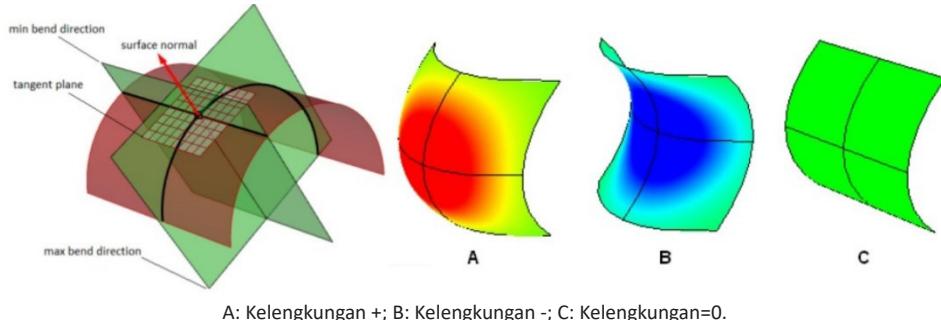
Langkah 3: Analisis Kelengkungan Bidang



- Analisis kelengkungan (*Curvature*) digunakan untuk mengetahui area yang mengalami beban dan gaya yang paling besar sehingga nilai kelengkungannya tinggi. Analisis ini juga dilakukan untuk mengetahui area mana yang berupa planar dan mana yang tidak.
- Komponen untuk analisis kelengkungan adalah **Surface Curvature**. Karena memerlukan input **Surface**, maka hasil **Mesh** harus dikonversi menjadi **Surface**.
- Konversi Mesh ke Surface dan menghasilkan nilai *uv* dari setiap titik pada **Surface** dilakukan dengan: **Mesh** → **Deconstruct Mesh** → **Surface from Points**, dan **Surface Closest Point**.
- Parameter *U* pada komponen **Surface from Points** didapatkan dari salah satu parameter *W/H* pada **Mesh Plane**. Modifikasi nilai *U* dilakukan dengan jalan menambahkan indeks 1 pada nilai *W/H*.

- Komponen **Surface Curvature** memerlukan input berupa S : *Surface* dan uv yang dihasilkan dari komponen **Surface Closest Point**.
- Komponen **Construct Mesh** digunakan untuk me-rekonstruksi Mesh yang parameter C : *color* didapatkan menggunakan **Gradient** yang input t (*parameter*) didapatkan dari keluaran G (*Gaussian*) dari *Surface Curvature*. Seperti halnya kelengkungan pada segmen, derajat kelengkungan pada permukaan ditentukan oleh dua nilai kelengkungan pada setiap titiknya:

$$K=k_{min}.k_{max}$$



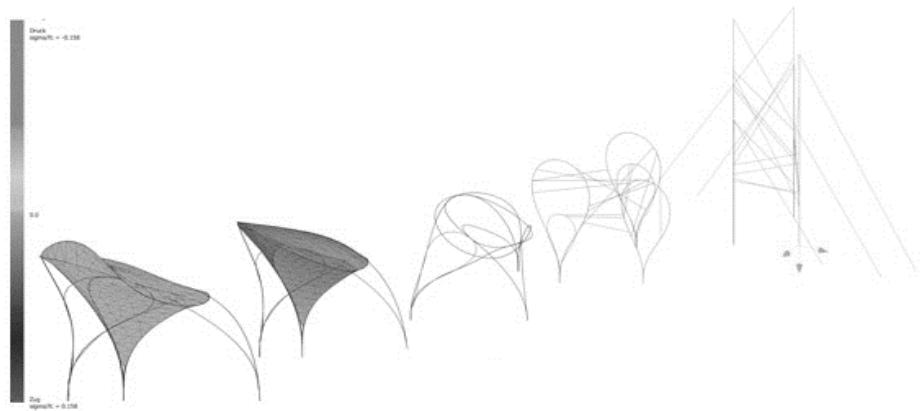
Gambar 204. Jenis Kelengkungan. (Sumber: Essential Mathematics for Computational Design Raja Isaa, 2020)

9.7. Struktur Hibrida (*Hybrid Structure*)

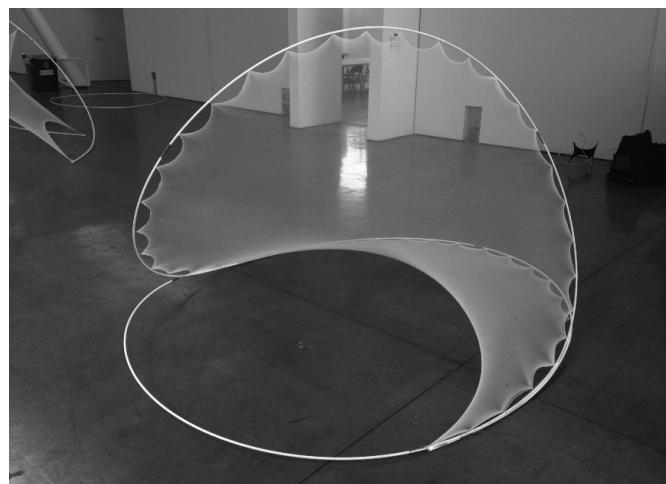
Pengembangan lain dari *form-finding* menggunakan pemodelan dan simulasi Kangaroo adalah model hibrida antara rangka batang yang memiliki kekuatan tekan (*compression*) dan menjadi rangka pengikat (*closed frame*) dengan material membran atau tekstil fleksibel (*fabric*) yang memiliki kekuatan Tarik (*tension*).



Gambar 205. Struktur hibrida. (Sumber: Jan Knippers, Institut for Computational Design (ICD), Stuttgart).

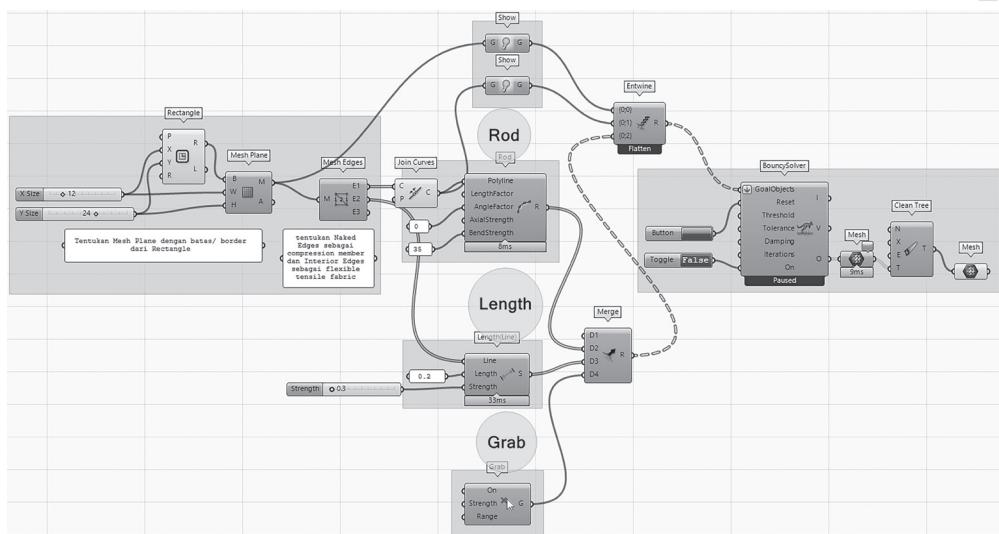


Gambar 206. Diagram Struktur Hibrida. (Sumber: Jan Knippers, Institut for Computational Design (ICD), Stuttgart).

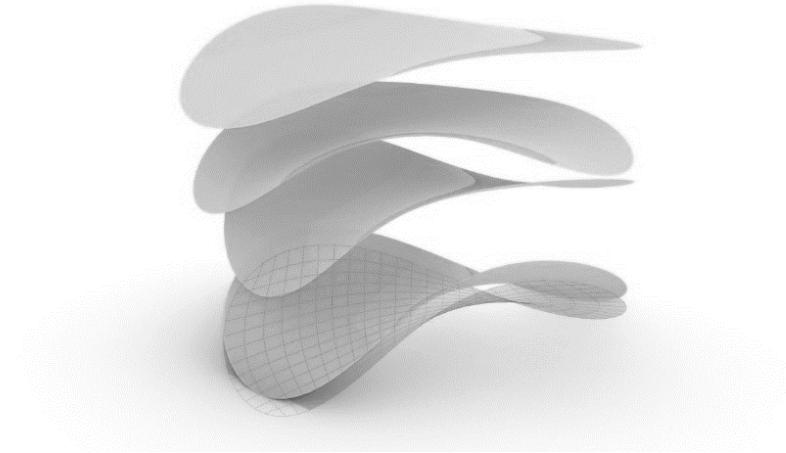


Gambar 207. Contoh Struktur Hibrida. (Sumber: Studio-LD)

BUKU AJAR AR-4214 TAHUN 2021

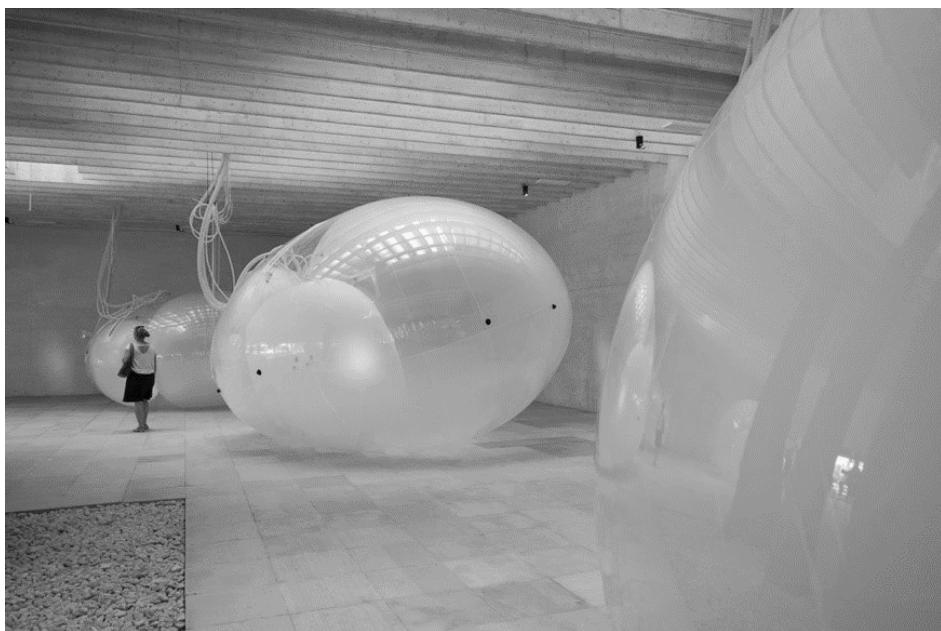


- Semua *Naked Edges* dari Mesh Plane merupakan *Rod* yang memiliki karakter *Bending-Active* sedangkan semua *Interior Edges* dari Mesh Plane memiliki karakter *tension* dari komponen **Length**. Parameter *Length* di-set=0 agar setiap *edge* berusaha memendekkan panjangnya.

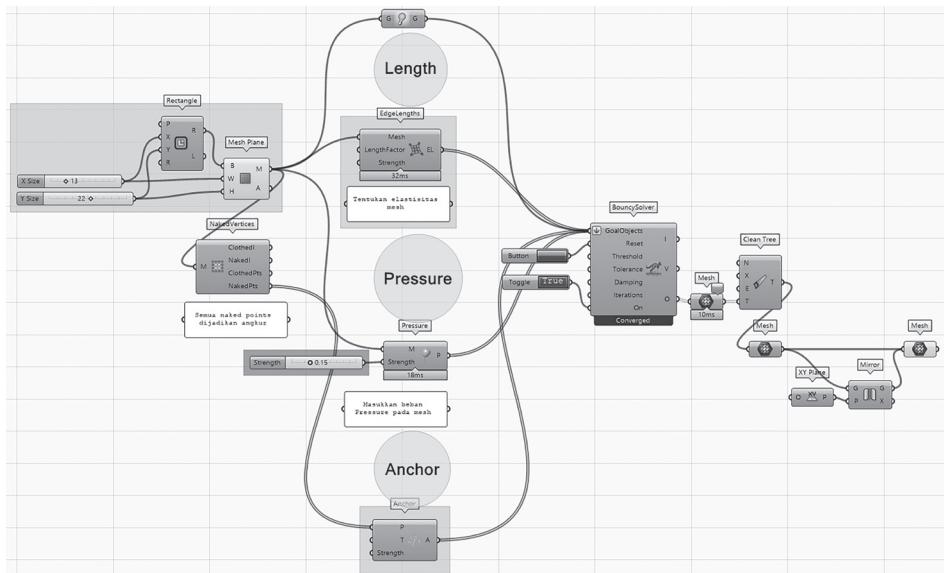


9.8. Struktur Pneumatik atau Mengembang (*Inflatable Structure*)

Struktur pneumatik menghasilkan bentuk dari tekanan positif yang dimasukkan ke dalam sebuah obyek 3D yang permukaannya memiliki karakter elastik.



Gambar 208. Struktur Pneumatik pada Paviliun Nordik pada Venice Architecture Biennale 2018.
(Sumber: inexhibit.com)



- Beban **Pressure** dikenakan pada *mesh*. Dimana semua *edge* dari *mesh* harus berkarakter elastis sehingga komponen **EdgeLength** digunakan untuk ini.
- Tentukan semua *Naked Points* pada *mesh* menggunakan komponen **Naked Vertices** dan jadikan semua titik ini sebagai angkur menggunakan komponen **Anchor**.

