

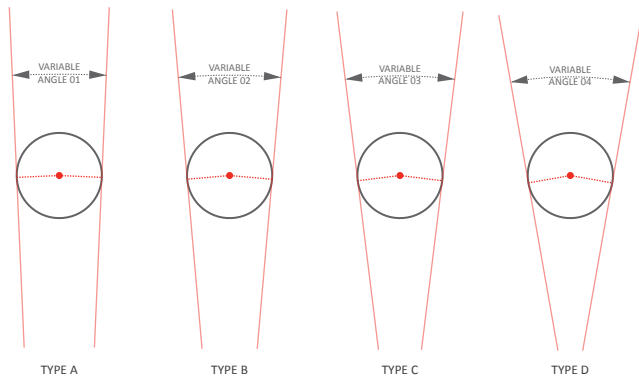
GRASSHOPPER WORKEXAMPLE
VB SCRIPTING + GALAPAGOS GH VER. 0.8.0010

WOOJAE SUNG
WOOJAE.SUNG@YAHOO.COM
[HTTP://WOOJSUNG.COM](http://woosung.com)

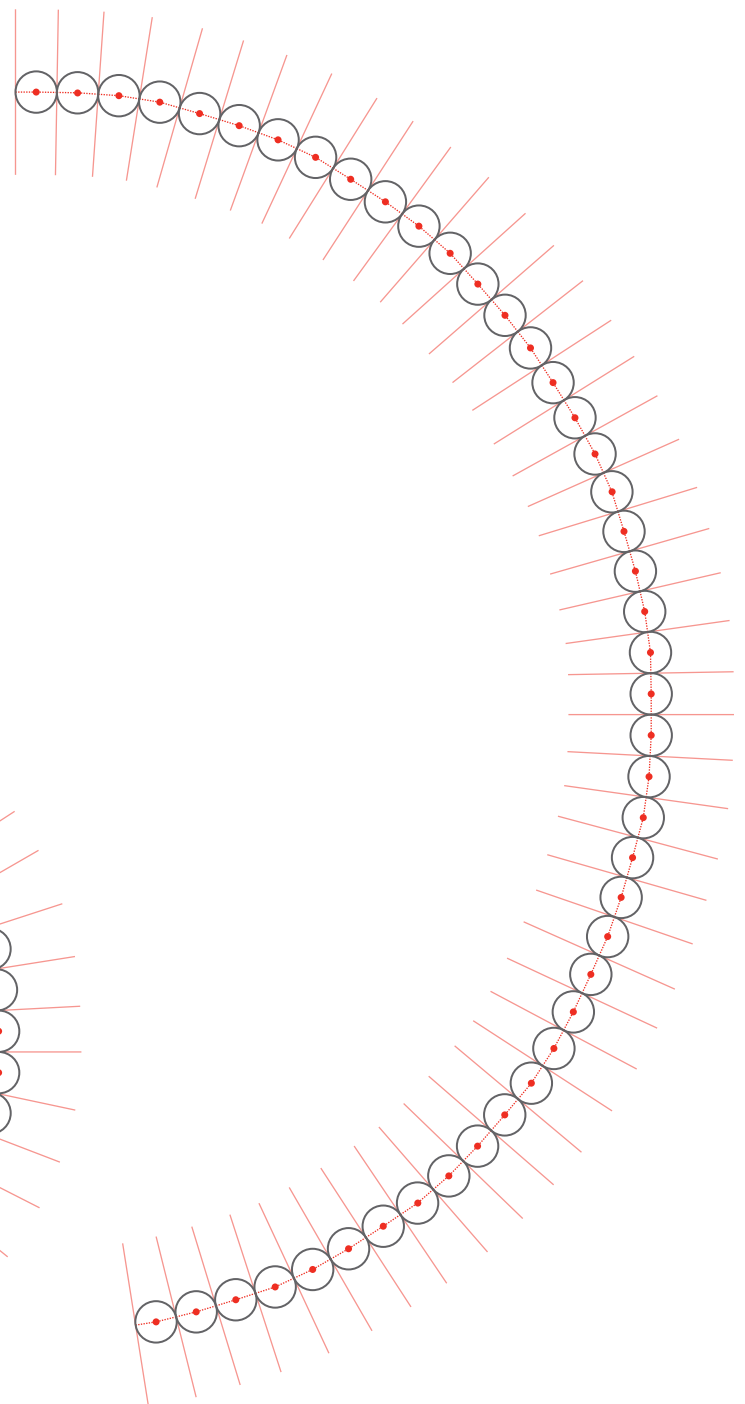
IDEA

기본적인 개념은 서로 독립적인 변수에 의해 정의되는 네개의 컴포넌트를 만든후 이를 이웃하여 배열하는 것으로 부터 출발합니다. 아래 그림의 step 01 에서 볼수 있듯이, 각각의 컴포넌트는 각기 다른 범위의 사잇각에 의해 정의됩니다. 예를 들어 type A의 경우는 사잇각이 5도에서 10도 사이의 어떠한 각이든 가질 수 있으며 type B는 7도에서 12도 사이의 어떤 각도 가질수 있습니다. 이렇게 서로 동일한 기하학적 배열을 가지고 있으면서도 서로 다른 사잇각을 가지는 네가지의 컴포넌트들은, 서로 변을 공유하며 일정한 순서-순서는 차후 조정하게 됩니다-에 의하여 선형배열을 하게 됩니다. 다른 type들 간의 서로 조금씩 다른 사잇각들로 인해 step 02와 같은 초기 배열을 형태를 가지게 됩니다. 우리의 목표는 독립적인 네개의 사잇각 변수들을 조금씩 조정하는 과정을 통해 step 02 에서 볼수 있는 선형배열을 step 03에서 처럼 원형 배열의 형태로 변환하는 것입니다. 즉 step 03의 그림에서 볼수 있듯이, 시작하는 선과 끝나는 선의 시작점과 끝점에서의 각각의 거리를 D1과 D2라고 할때, 두 거리값의 합이 최소가 되도록 각각의 사잇각을 조금씩 조절하여 나가는 과정에서 최적의 배열, 즉 최소한의 거리값을 가지는 사잇각들의 조합을 찾아내는 것입니다. 이러한 과정은 일일이 넘버 슬라이더를 조절하여서는 쉽지 않으리라고 생각됩니다. 이러한 무수한 시행-착오의 과정을 거쳐야 하는 방정식의 해는 Grasshopper의 Galapagos라는 방적식 풀이 모듈을 통해 쉽게 찾아낼수 있습니다.

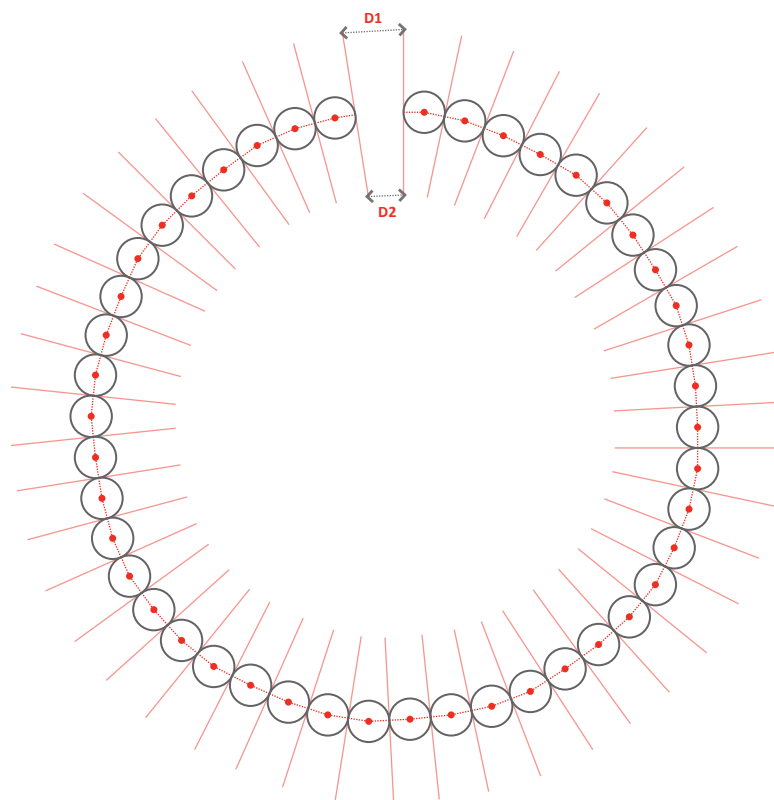
STEP01 COMPONENTS



STEP02 INITIAL ARRAY

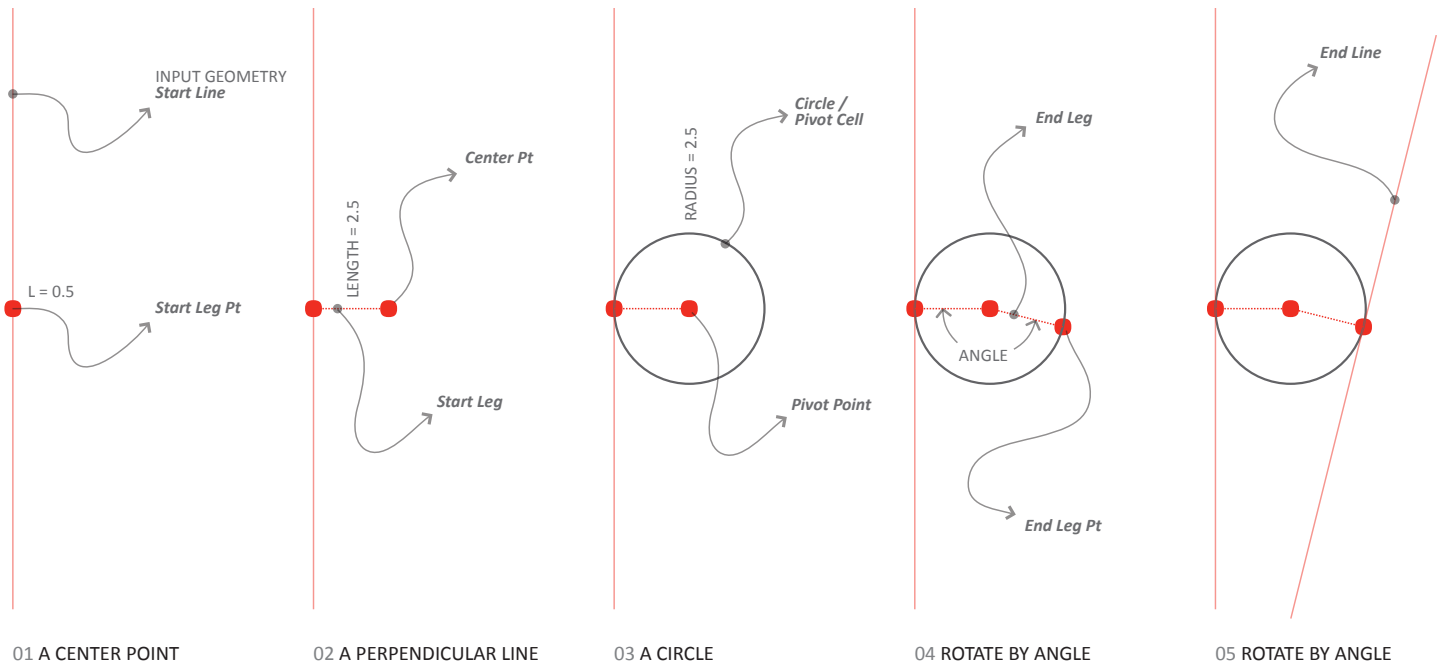


STEP03 PACKED



DEFINING A COMPONENT

첫번째 단계로, 우리는 Grasshopper의 기본 오브젝트만을 통해 컴포넌트를 정의해 볼것입니다. 첫번째로 해야 할 것은, 라이노에서 길이 20이 되는 선을 그리고(Start Line) 이를 Grasshopper의 Curve 오브젝트에 연결합니다. 그후 그 선의 중점(Start Leg Pt)을 찾고 이 점에서 길이 2.5의 법선(Start Leg)을 그립니다. 이 선의 끝나는 점을 찾고(Center Pt), 이를 중점으로 하며 Start Line에 접하는 원(Circle / Pivot Cell)을 그립니다. 그리고, 이 원의 중심(Pivot Point)을 회전축으로 삼아 Start Leg, Start Leg Pt, Start Line을 특정각도(Angle)로 회전시켜 End Leg, End Leg Pt, End Line을 작성하게 됩니다.



01 A CENTER POINT

02 A PERPENDICULAR LINE

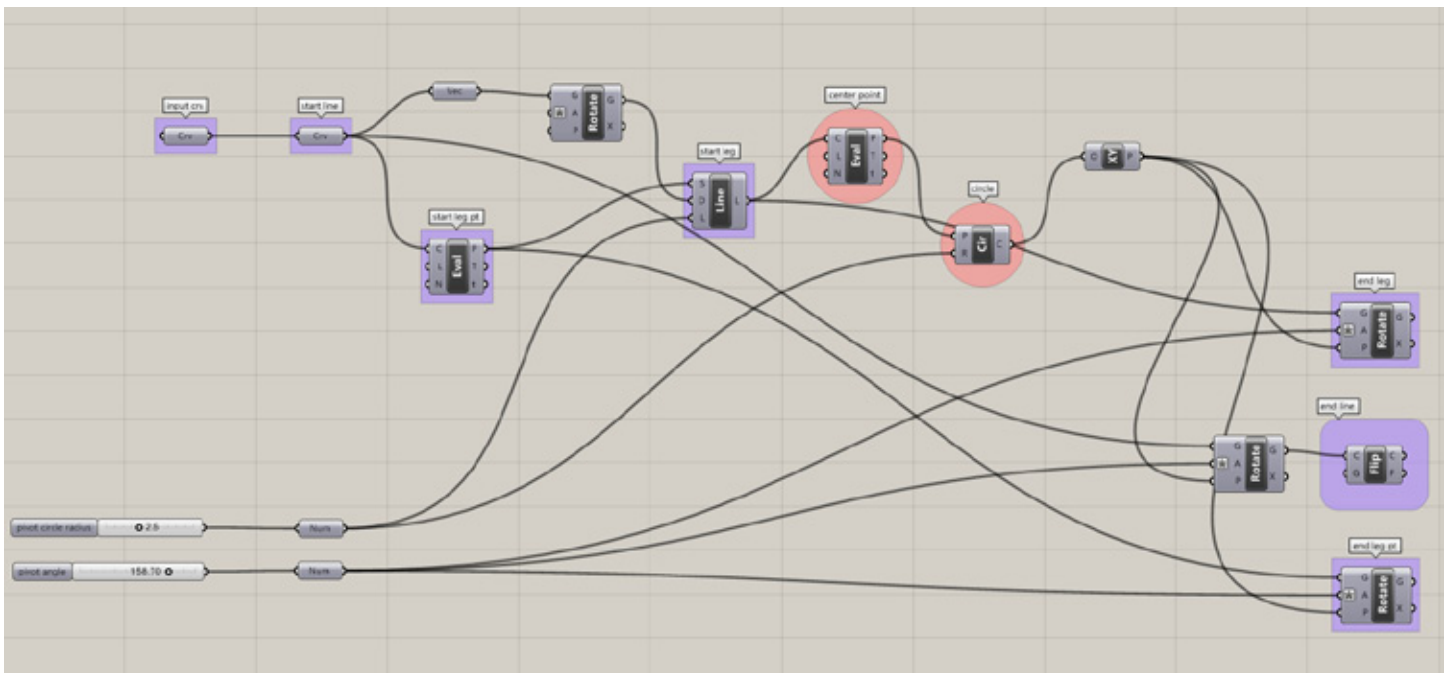
03 A CIRCLE

04 ROTATE BY ANGLE

05 ROTATE BY ANGLE

GH DEFINITION

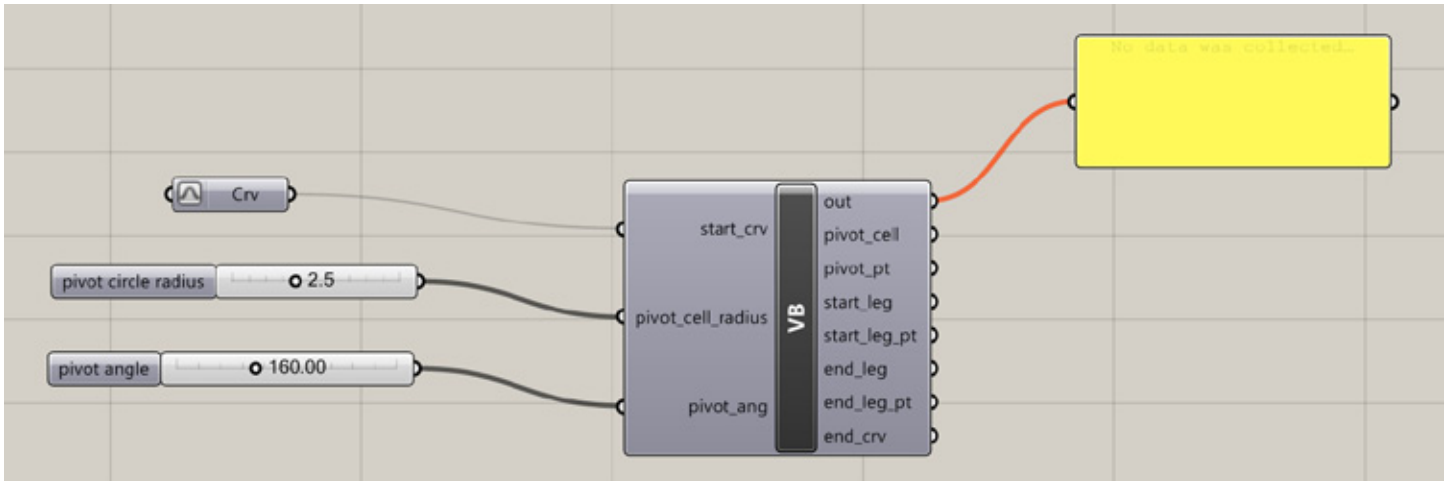
아래 스크린 샷은 완성된 Grasshopper Definition의 모습입니다. (001 component GH objects.ghx 파일을 참조하세요).



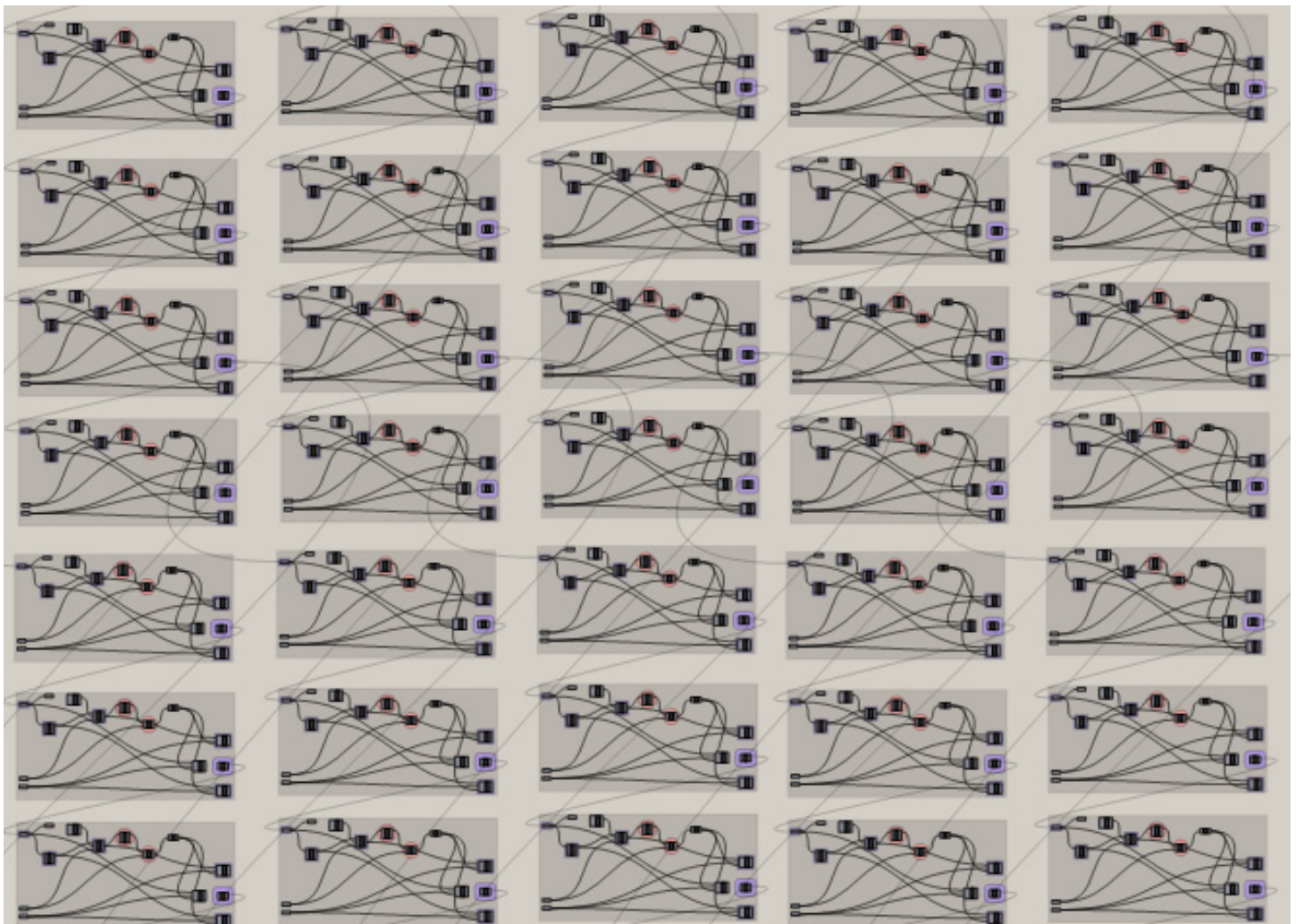
CUSTOM VB COMPONENT

이번에는 위에서 정의한 Grasshopper 객체들을 VB 스크립트를 통하여 단일 오브젝트로 변환해보겠습니다. 아래 스크린샷에서 보시는 input / output 변수의 이름들은 앞장의 컴포넌트 정의 다이어그램의 변수명들과 동일합니다. 즉, input line, 반지름, 그리고 사잇각의 각도를 input 파라미터로 가지고 결과물로서 pivot_cell, pivot_pt 등등을 만들어 냅니다. 기본적으로 동일한 작업을 하게 되지만 단 하나의 오브젝트로 이루어져 있습니다.

(002 component VB.ghx 파일을 참조하세요)

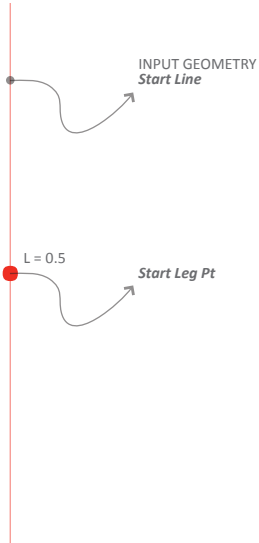


그렇다면, 굳이 이러한 VB Scripting을 써서 작업을 해야 하는 이유가 뭘까요? 많은 다른 이유들이 있을수 있겠지만, 본 작업의 경우는 아래와 같이 컨트롤 하기 어려운 상황을 만들지 않기 위해서 입니다.



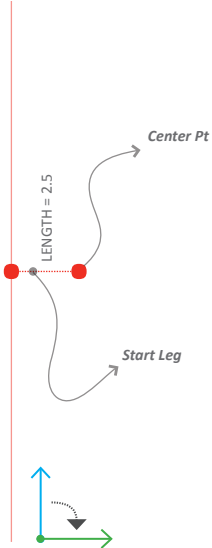
CODE REVIEW

이번에는 VB 오브젝트의 코드를 한줄씩 살펴보도록 하겠습니다. VB 오브젝트를 더블클릭하면 스크립트를 수정할수 있는 창이 나옵니다.



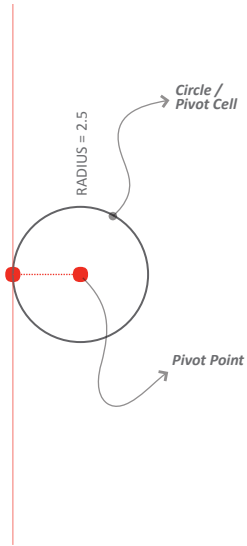
```
'defines start_leg_pt ////////////////////////////////////////
Dim s_pt As New point3d(start_crv.PointAt(0.5))
start_leg_pt = s_pt
```

- ' 로 시작하는 줄은 주석입니다. 즉 어떠한 일도 하지 않는 줄입니다.
- 두번째 줄에서는 s_pt라는 새로운 점을 생성하고 이 점에 start_crv의 L=0.5 즉 중점에 해당하는 점을 할당합니다.
- 그 다음, s_pt를 start_leg_pt라는 output변수에 할당하여 줍니다. 즉 라이노 뷰포트에서 이점을 볼수 있게 해줍니다.
- 만약 세번째 줄의 명령이 없다면 VB는 점을 생성하여 놓고도 그것을 우리에게 보여주지 않습니다.
- 변수의 정의와 점으로 구분되는 명령어들에 대한 좀 더 자세한 설명을 찾으시는 분은 <http://www.rhino3d.com/5/rhinocommon/> 을 방문하시면 됩니다.



```
'defines start_leg ////////////////////////////////////////
Dim start_crv_start As New Point3d(start_crv.PointAt(0.0))
Dim start_crv_end As New Point3d(start_crv.pointat(1.0))
Dim original_vector As New Vector3d(start_crv_start - start_crv_end)
Dim rotation_ang_start_leg As Double = math.PI / 2
Dim rotation_axis_start_leg As New Vector3d(0, 0, 1)
original_vector.Rotate(rotation_ang_start_leg, rotation_axis_start_leg)
Dim line_start As New line(start_leg_pt, original_vector, pivot_cell_radius)
start_leg = line_start
```

- 이번에는 전단계에서 생성한 중점, 즉 start_leg_pt를 출발점으로 하는 법선을 그릴 차례입니다.
- 먼저, 기준이 되는 선, 즉 start_crv의 방향벡터를 구하고 이 벡터를 시계방향으로 90도 돌려줌으로써 법선 벡터를 생성하게 됩니다.
- 기준이 되는 선, start_crv의 양끝점을 정의하여 줍니다.
- 그다음 두 점의 차를 통해 original_vector를 정의해 줍니다. 이는 왼편그림에 파란색 화살표로 표시되어져 있습니다.
- 이제는 이 파란색 벡터를 회전시켜 초록색 벡터로 만들어 주어야 합니다. 그러기 위해서는 rotation이란 methods를 사용해야 합니다. rotation method는 회전각과 회전축이라는 두가지 변수를 통해 작동합니다.
- 회전각, rotation_ang_start_leg를 90도로 정의 합니다. (파이/2 = 90도)
- 회전축은 Z 방향벡터로 잡습니다.
- 그후 original_vector에 rotate method를 점 연결자로 연결하고 회전각과 회전축을 입력하여 original_vector를 90도 돌립니다. 회전된 벡터는 이제 초록색 화살표와 동일합니다.
- 이제 새로운 선을 긁습니다. 시작점(start_leg_pt)와 방향(이미 회전된 original_vector)와 길이(pivot_cell_radius)를 입력하여 line_start라는 선을 생성합니다.
- 이제 그 선을 start_leg라는 output 파라미터에 연결하여 뷰포트상에 출력될수 있도록 합니다.



```
'defines pivot_pt ////////////////////////////////////////

Dim p_pt As New point3d(line_start.PointAt(1.0))

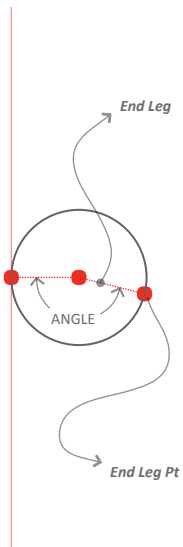
pivot_pt = p_pt

'defines pivot_cell ////////////////////////////////////////

Dim pivot_circle As New Circle(line_start.PointAt(1.0), pivot_cell_radius)

pivot_cell = pivot_circle
```

- 전단계에서 만든 법선, line_start를 이용하여 원을 그립니다.
- 우선 line_start의 끝점에 점을 생성하여 이를 p_pt라는 점에 할당하고 이를 다시 pivot_pt라는 output 파라미터에 할당합니다.
- p_pt라는 점을 중심으로 하고 pivot_cell_radius만큼의 반지름을 가지는 원을 생성하여 pivot_circle이라는 원에 할당합니다. 이는 다시 pivot_cell이라는 output 파라미터에 할당됩니다.



```
'defines end_leg ////////////////////////////////////////

Dim rot As transform = transform.Rotation(pivot_ang * math.PI / 180, vector3d.ZAxis, line_start.PointAt(1.0))

Dim rotated_leg As line = line_start

rotated_leg.Transform(rot)

end_leg = rotated_leg

'defines end_leg_pt ////////////////////////////////////////

s_pt.Transform(rot)

end_leg_pt = s_pt

'defines end_crv ////////////////////////////////////////

Dim st_line As line = start_crv

st_line.Transform(rot)

Dim end_line_st_pt As New Point3d(st_line.PointAt(1.0))

Dim end_line_end_pt As New Point3d(st_line.PointAt(0.0))

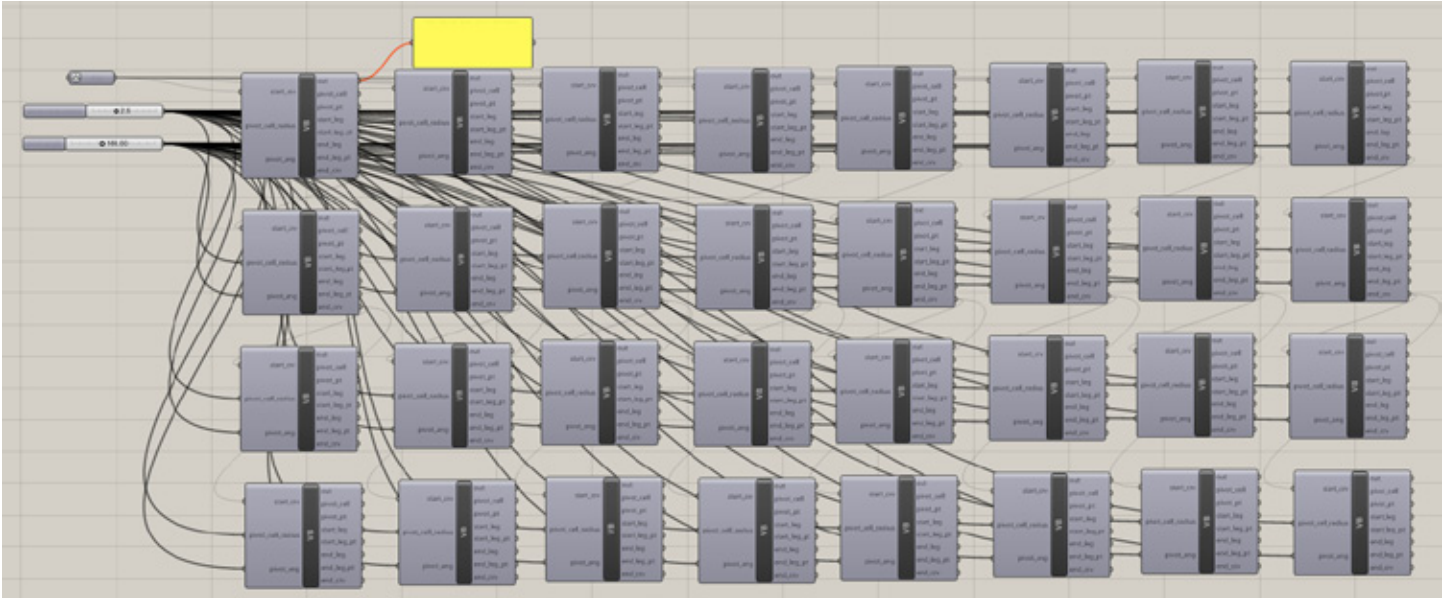
Dim end_line As New line(end_line_st_pt, end_line_end_pt)

end_crv = end_line
```

- 이번단계에서는 전단계에서 만들었던 start line, start leg, start leg point들을 중점인 pivot point를 중심으로 pivot_ang만큼 회전시켜서 end line, end leg, end leg point등을 생성하도록 합니다.
- 한개 이상의 회전 작업을 해야 하므로 회전 matrix를 생성후 이를 재차 사용하는 것이 편리합니다.
- 회전 matrix의 input으로는 회전각(pivot_ang), 회전축(z axis), 그리고 회전의 중심이 되는 점이 필요합니다.
- rotated_leg라는 새로운 라인을 생성후 이에 기존의 선인 line_start를 할당합니다.
- rotated_leg를 transform matrix인 rot을 통해 회전합니다. 그리고 이 회전된 선을 end_leg라는 output 파라미터에 할당합니다.
- let_pt에도 동일한 작업을 해 줍니다.
- end_crv에도 동일한 작업을 해 줍니다. 하지만 선은 방향을 가지고 있으므로 start_crv를 회전시 방향이 바뀌게 됩니다. (초록색 화살표로 표시). 그래서 이 선의 방향을 반대로 돌려야 합니다.
- 일단 선을 회전한후, (st_line.Transform(rot)), 이 선의 양끝점을 찾아 반대 방향으로 연결하여 end_crv의 방향을 바로 잡습니다.

USING A SUBROUTINE (FUNCTION) IN VB COMPONENT

이제 VB custom 오브젝트가 잘 작동합니다. 이번에는 컴포넌트의 타입에 의거해서 이 컴포넌트들을 배열하려고 합니다. 그러나 아래 그림에서 보듯 아직도 복사와 붙임이라는 과정을 거쳐야 하므로 역시나 번거롭고 더딘 작업이 될것 같습니다. (003 subroutine component VB.ghx 파일을 참조하세요)



하지만 우리가 VB scripting의 서브루틴의 개념을 이해 한다면 이렇게 긴 코드를 좀더 효율적으로 줄일수도 있을겁니다. 서브루틴이란 일종의 사용자 함수와 같은 개념인데, 일단 정의를 해 놓으면, 서브루틴은 우리가 VB scripting에서 line이라는 공용 함수를 쓰는것과 동일하게 사용할 수 있습니다.

```
line(start point, direction vector, distance)
```

예를 들어, FunctionA라고 불리는 서브루틴을 만들고 싶다고 합시다. 이 서브루틴은 두개의 input과 하나의 output으로 이뤄져 있고 어떤일인가를 하게 됩니다. 이 함수가 작동하게 하기 위해서 우리의 VB 코드내의 어딘가엔가는 이 함수가 무엇을 하는 함수인지에대해 정의를 해줘야 하며 또 다른 어떤 부분인가에서는 이 함수를 실제로 사용하기 위해 call 하는 부분도 있어야 할것 입니다. 함수의 정의는 보통 Custom additional code 를 위한 별도의 공간에 위치하게 되며 함수를 사용하기 위해 call하는 부분은 우리가 흔히 코드를 입력하는 곳에 위치하여야 합니다. 함수의 정의시 눈여겨 봐야 할 부분은 ByVal과 ByRef인데요, ByVal은 이 변수가 input변수라는 것을 말하고 ByRef는 이 변수가 output 파라메터라는 것을 암시하여 줍니다.

```
FunctionA (ByVal input1, ByVal input2, ByRef output1)
```

```
<Custom additional code>  
Sub FunctionA (ByVal input1, ByVal input2, ByRef output1)  
  
.....  
  
End Sub  
</Custom additional code>
```

스크립트 에디터를 열고 “Private Sub ~” 에서 “End Sub” 까지를 복사합니다.

```
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53 Utility Functions
54
55
56
57
58
59
60
61
62
63
64
65 Private Sub RunScript(ByVal start_crv As Line, ByVal pivot_cell radius As Double, ByVal pivot_ang As Double, ByRef pivot_cell As Object, ByRef
66
67 'defines start leg pt //////////////////////////////////////
68
69 Dim s pt As New point3d(start_crv.PointAt(0.5))
70
71 start leg pt = s pt
72
73 'defines start leg //////////////////////////////////////
74
75 Dim start_crv_start As New Point3d(start_crv.PointAt(0.0))
76 Dim start_crv_end As New Point3d(start_crv.pointat(1.0))
77
78 Dim original_vector As New Vector3d(start_crv_start - start_crv_end)
79
80 Dim rotation_ang_start_leg As Double = math.PI / 2
81 Dim rotation_axis_start_leg As New Vector3d(0, 0, 1)
82
83 original_vector.Rotate(rotation_ang_start_leg, rotation_axis_start_leg)
84
85 Dim line_start As New line(start leg pt, original_vector, pivot_cell radius)
86
87 start leg = line start
88
89 'defines pivot pt //////////////////////////////////////
90
91 Dim p pt As New point3d(line_start.PointAt(1.0))
92
93 pivot pt = p pt
94
95 'defines pivot cell //////////////////////////////////////
96
97 Dim pivot_circle As New Circle(line_start.PointAt(1.0), pivot_cell radius)
98
99 pivot_cell = pivot_circle
100
101 'defines end leg //////////////////////////////////////
102
103 Dim rot As transform = transform.Rotation(pivot_ang * math.PI / 180, vector3d.ZAxis, line_start.PointAt(1.0))
104
105 Dim rotated_leg As line = line start
106
107 rotated_leg.Transform(rot)
108
109 end leg = rotated leg
110
111 'defines end leg pt //////////////////////////////////////
112
113 s pt.Transform(rot)
114
115 end leg pt = s pt
116
117 'defines end crv //////////////////////////////////////
118
119 Dim st_line As line = start_crv
120
121 st_line.Transform(rot)
122
123 Dim end_line_st_pt As New Point3d(st_line.PointAt(1.0))
124
125 Dim end_line_end_pt As New Point3d(st_line.PointAt(0.0))
126
127 Dim end_line As New line(end_line_st_pt, end_line_end_pt)
128
129 end crv = end line
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152 End Sub
```

그다음 이를 ‘<Custom additional code> 과 ‘</Custom additional code> 사이의 공간에 붙여 넣습니다.

```
153
154 '<Custom additional code>
155
156 '</Custom additional code>
157
158 End Class
```

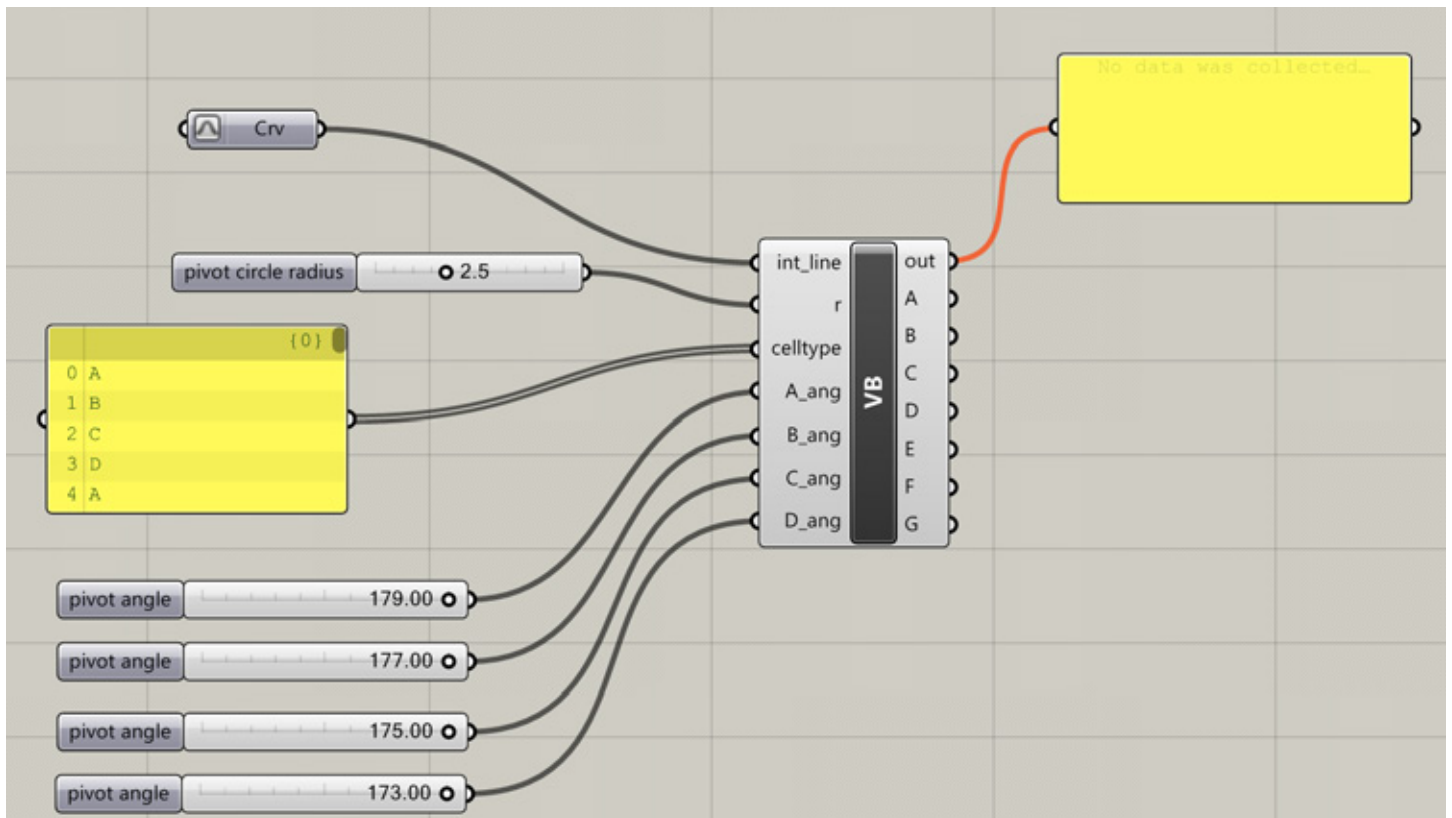
그리고 다음 그림과 같이 함수의 이름을 정의 해 줍니다. 이번 경우는 함수의 이름이 component입니다.

```

182 End Sub
183
184 /--/
185
186 Private Sub RunScript(ByVal start_crv As Line, ByVal pivot_cell_radius As Double, ByVal pivot_ang As Double, ByRef pivot_cell As Object, ByRef pivot
187
188 'defines start_leg_pt ////////////////////////////////////////////////////
189
190 Dim s_pt As New point3d(start_crv.PointAt(0.5))
191
192
193
194 /--/
195
196 Sub component(ByVal start_crv As Line, ByVal pivot_cell_radius As Double, ByVal pivot_ang As Double, ByRef pivot_cell As Object, ByRef pivot
197
198 'defines start_leg_pt ////////////////////////////////////////////////////
199
200 Dim s_pt As New point3d(start_crv.PointAt(0.5))
201

```

이제 VB오브젝트의 실질적인 input과 output탭을 다음 그림과 같이 바꾸어 줍니다. input은 celltype이라는 string 변수가 추가되고 세개의 추가적인 사잇각 변수가 추가된것을 제외하고는 거의 이전과 동일합니다. 여기서 중요한 것은 celltype이라는 문자열 변수입니다. 이 변수는 얼마나 많은 컴포넌트들이 어떠한 배열로 연결되어 있는지를 정의하는 변수입니다.



CODE REVIEW

이번에는 VB 오브젝트의 코드를 한줄씩 살펴보도록 하겠습니다. VB 오브젝트를 더블클릭하면 스크립트를 수정할수 있는 창이 나옵니다.

```
'defines output parameters //////////////////////////////////////
```

```
Dim pivot_cell_list As New List(Of circle)
Dim pivot_pt_list As New List(Of point3d)
Dim start_leg_list As New List(Of Line)
Dim start_leg_pt_list As New List(Of point3d)
Dim end_leg_list As New List(Of Line)
Dim end_leg_pt_list As New List(Of point3d)
Dim end_crv_list As New List(Of Line)
```

- 이 부분에서는 VB컴포넌트의 output 값들을 정의 합니다. 우리는 복수의 컴포넌트를 가지고 있으며 이들의 모든 요소들, 즉 start line, start leg, pivot cell, pivot point, end leg, end line등등이 모두 뷰포트상에 출력되기를 바랍니다. 이를 위해서 단일 값보다는 리스트의 형태로 출력을 하는것이 더 바람직합니다.

```
'defines output parameters of "component" subroutine //////////////////////////////////////
```

```
Dim pivot_cell As circle
Dim pivot_pt As point3d
Dim start_leg As line
Dim start_leg_pt As point3d
Dim end_leg As line
Dim end_leg_pt As point3d
Dim end_crv As line
```

- 이 부분은 서브루틴의 output 파라미터들을 정의 합니다. 만약 우리가 서브루틴을 call하기 전에 이 변수들을 설정해놓지 않는다면 VB오브젝트는 에러메세지를 나타내게 될 것입니다. 그와는 반대로 서브루틴의 input 변수들은 미리 정의할 필요가 없습니다. 왜냐하면 이 변수들은 서브루틴 내에서 정의가 되거나 혹은 VB오브젝트 자체의 input 변수이기 때문입니다.

```
'defines rotation_ang //////////////////////////////////////
```

```
Dim rotation_ang As Double
```

- 이부분은 사잇각을 정의 하는데요, 네개나 사잇각을 입력하면서 왜 또 다른 변수가 필요한걸까요. 이는 네가지 다른 경우에 각각 다른 사잇각을 입력하게 되므로 아직은 어떤 값이 사잇각으로 들어갈지 모르기 때문입니다. 그래서 미리 가상의 값을 정해 놓고 경우에 따라 이 가상의 값에 다른 값을 바꿔치기 함으로써 복수의 경우의 수에 유연하게 대처할수 있습니다.

'checks component types and computes output based on the types //'

```
For i As Integer = 0 To celltype.Count - 1
```

```
  If i = 0 Then
```

```
    If celltype(i) = "A" Then
```

```
      rotation_ang = A_ang
```

```
    Else If celltype(i) = "B" Then
```

```
      rotation_ang = B_ang
```

```
    Else If celltype(i) = "C" Then
```

```
      rotation_ang = C_ang
```

```
    Else If celltype(i) = "D" Then
```

```
      rotation_ang = D_ang
```

```
    End If
```

```
    component(int_line, r, rotation_ang, pivot_cell, pivot_pt, start_leg, start_leg_pt, end_leg, end_leg_pt, end_crv)
```

- 이제 컴포넌트 타입을 나타내는 문자열에 의해 컴포넌트를 증식시키도록 합니다.
- 문자열에 얼마나 많은 문자가 있는가에 따라 for 구문을 돌립니다.
- 만약 이번 회전은 첫번째라면(if i = 0 then), "int_line", 즉 라이노에서 만든 초기 지오메트리가 서브루틴 component의 초기 입력값이 됩니다. 그렇지 않은 경우는 바로 이전의 연산에서 결과물로 나온 "end_crv"를 서브루틴의 입력값으로 사용하게 됩니다.
- 그다음 줄에서는 어떠한 타입 문자 (A/B/C/D)가 나오는지에 따라 각기 다른 사잇각 "rotation_ang"을 적용하는 지에 대한 내용입니다.
- 그 다음에서는 드디어 서브루틴 component를 호출하게 됩니다. 입력은 "int_line"(input geometry), "r"(radius of a circle) 과 "rotation_ang" 이며 이를 통해 우리는 "pivot_cell", "start_leg", "start_leg_pt", "end_leg", "end_leg_pt", 과 "end_crv" 를 얻습니다.

```
Else
```

```
  If celltype(i) = "A" Then
```

```
    rotation_ang = A_ang
```

```
  Else If celltype(i) = "B" Then
```

```
    rotation_ang = B_ang
```

```
  Else If celltype(i) = "C" Then
```

```
    rotation_ang = C_ang
```

```
  Else If celltype(i) = "D" Then
```

```
    rotation_ang = D_ang
```

```
  End If
```

```
  int_line = end_crv
```

```
  component(int_line, r, rotation_ang, pivot_cell, pivot_pt, start_leg, start_leg_pt, end_leg, end_leg_pt, end_crv)
```

```
End If
```

- 서브루틴을 호출하기 전 "end_crv" 를 다음 iteration 을 위한 초기값인 "int_line" 에 할당합니다.

```
pivot_cell_list.Add(pivot_cell)
pivot_pt_list.Add(pivot_pt)
start_leg_list.Add(start_leg)
start_leg_pt_list.Add(start_leg_pt)
end_leg_list.Add(end_leg)
end_leg_pt_list.Add(end_leg_pt)
end_crv_list.Add(end_crv)
```

Next

- 각각의 iteration에서 발생한 output값을 리스트의 형태로 저장합니다.

```
'output //////////////////////////////////////
```

```
A = pivot_cell_list
B = pivot_pt_list
C = start_leg_list
D = start_leg_pt_list
E = end_leg_list
F = end_leg_pt_list
G = end_crv_list
```

- 이 변수들을 라이노 뷰포트에 뿌려주기 위해 A,B,C,D,E,F,G라는 output 탭에 연결 시킵니다.

OPTIMUM SOLUTION BY GALAPAGOS

주지하다시피, 컴포넌트들을 아주 긴밀하게 연결시키기 위해 우측 다이어그램의 두 거리값의 합은 되도록 작아야 합니다. 하지만 Galapagos는 최소값이 아닌 최대값을 찾도록 설계되어져 있습니다. 이를 위해 우리는 두 값을 합한후 이의 역수를 구해서 이 값이 최대가 되는 사잇각의 조합을 Galapagos에게 찾도록 합니다.

(004 galapagos.ghx 파일을 참조하세요)

- VB 에서 생성된 end line중 제일 마지막 라인과 start line중 가장 첫번째 라인을 찾는다.
- 이들이 양끝점을 찾아 시작점끼리의 거리와 끝점끼리의 거리를 구해 D1, D2라 정의 한다.
- 두 값을 합한후 역수를 취한다.
- Galapagos의 Genome탭에 사잇각 슬라이더 들을 연결해 준다.
- Galapagos의 Fitness 탭을 거리값의 역수와 연결한다.
- Galapagos를 더블클릭하여 solver 탭으로 이동후 start solver버튼을 누른후 연산이 완료될때 까지 기다린다.

