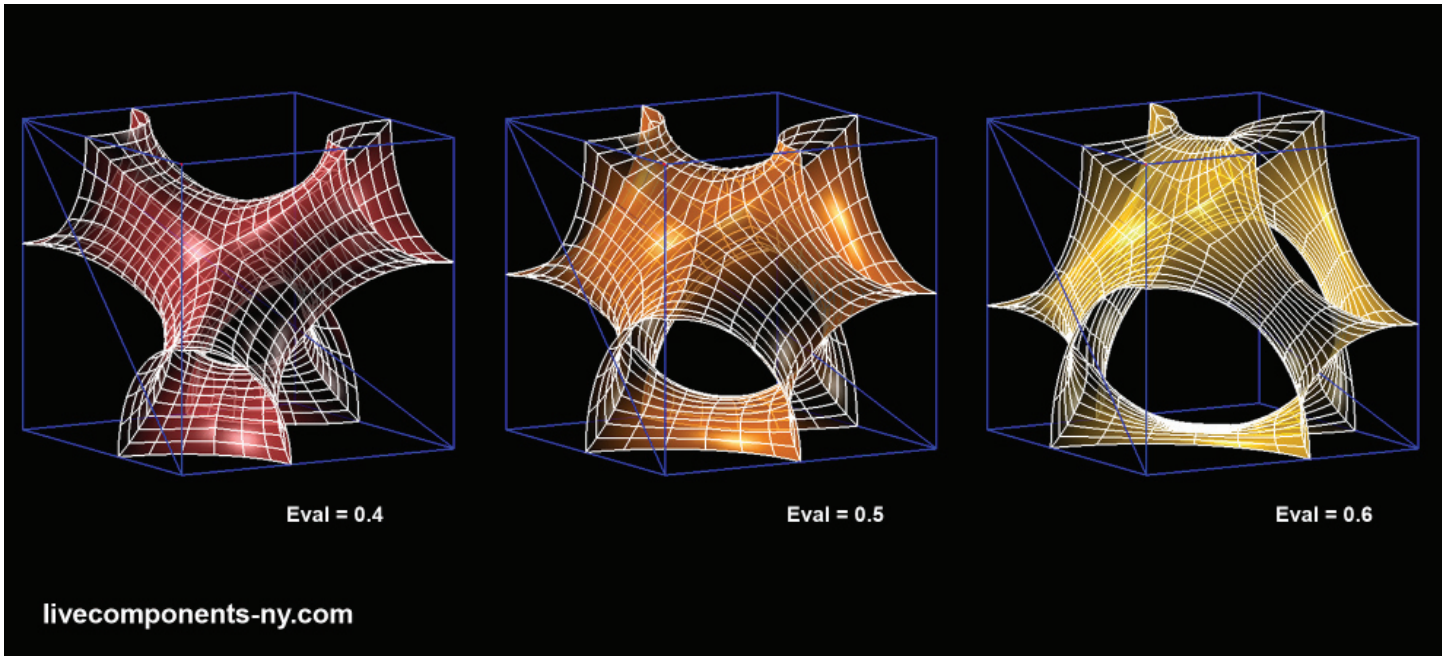
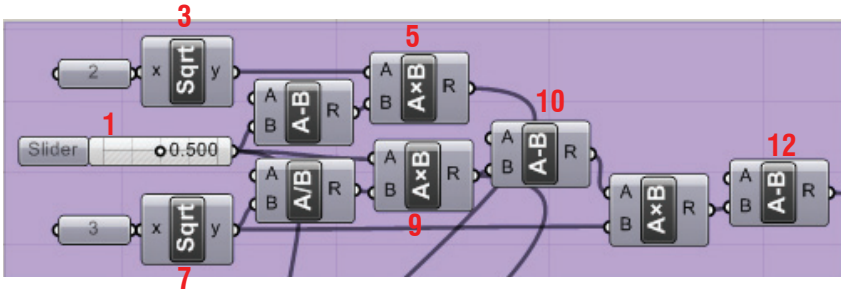
8_1 Minimal Surface Module



Step1 : Setting the Geometric Rule
- It requires the mathematical rule for parametric points on a cubic module below.

1. ***Slider*** *-> parameter A*
- Lower limit=0, Upper limit=1, Value=0.6 (varies)
2. ***Integer***
- Set integer : 2
3. ***Sqrt (Square Root)***
- x : Integer(y)
4. ***A-B (Subtraction)***
- A : Set generic data : 1
- B : Slider
5. ***AxB (Multiplication)*** *-> parameter B*
- A : Sqrt(y)
- B : A-B(R)
6. ***Integer***
- Set integer : 3
7. ***Sqrt (Square Root)***
- x : Integer(y)
8. ***A/B (Subtraction)***
- A : Set generic data : 2
- B : Sqrt(y)
9. ***AxB (Multiplication)*** *-> parameter C*
- A : Slider
- B : A/B(R)
10. ***A-B (Subtraction)***
- A : Set generic data : 1
- B : AxB(R)
11. ***AxB (Multiplication)***
- A : A-B (A)
- B : Sqrt(y)
12. ***A-B (Subtraction)*** *-> parameter D*
- A : 1
- B : AxB(R)



Step2 : Constructing a Module Piece
It starts from a basic geometry module which is flexible parametrically.

13. ***Pt (Point)*** -> these are going to be inputs for the module expansion
- "Pt Mid" - *center points of one side surface*
- "Pt A" - *one corner of surface*
- "Pt B" - *next corner of surface*
- "Pt Inside" - *center points of inside lines*

14. ***Ln (Line)***
- A : Pt B
- B : Pt A

15. ***Eval (Evaluate Length)***
- C : Ln(L)
- L : Slider <- *parameter A*

16. ***Ln (Line)***
- A : Pt A
- B : Pt Mid

17. ***Eval (Evaluate Length)***
- C : Ln(L)
- L : AxB <- *parameter B*

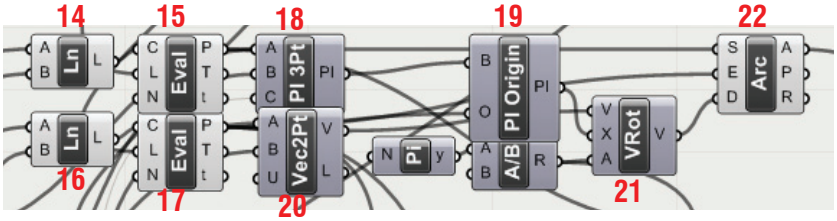
18. ***Pl 3Pt (Plane 3Pt)***
- A : Pt Mid
- B : Pt A
- C : Pt B

19. ***Pl Origin (Plane Origin)***
- B : Pl 3Pt
- O : Eval(P)

20. ***Vec2Pt (Vector 2Pt)***
- A : Pt B
- B : Pt A

21. ***VRot (Rotate)***
- V : Vec2Pt(V)
- X : Pl Origin (Pl)
- A : π / -2 (see definitions)

22. ***Arc (Arc SED)***
- S : Eval(P) from 15.
- E : Eval(P) from 17.
- D : VRot(V)



23. ***Ln (Line)***
- A : Pt B
- B : Pt Inside

24. ***Eval (Evaluate Length)***
- C : Ln(L)
- L : AxB <- *parameter C*

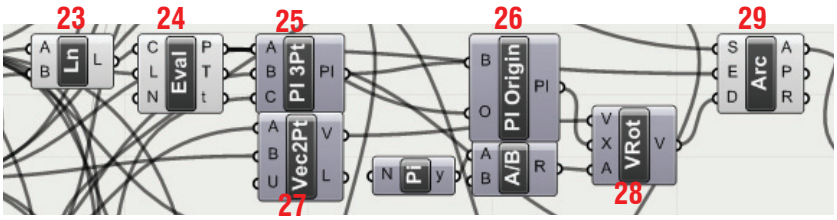
25. ***Pl 3Pt (Plane 3Pt)***
- A : Pt A
- B : Pt B
- C : Pt Inside

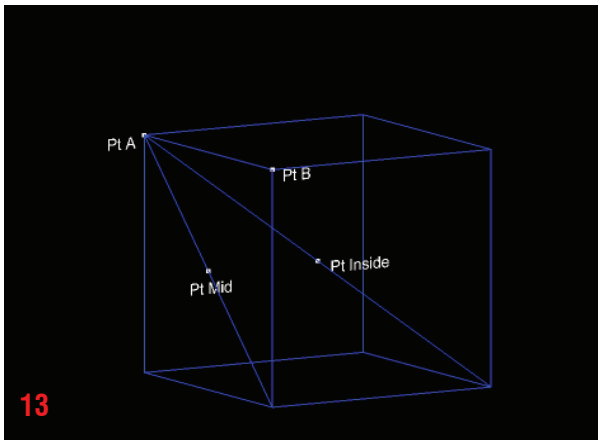
26. ***Pl Origin (Plane Origin)***
- B : Pl 3Pt
- O : Eval(P) from 24.

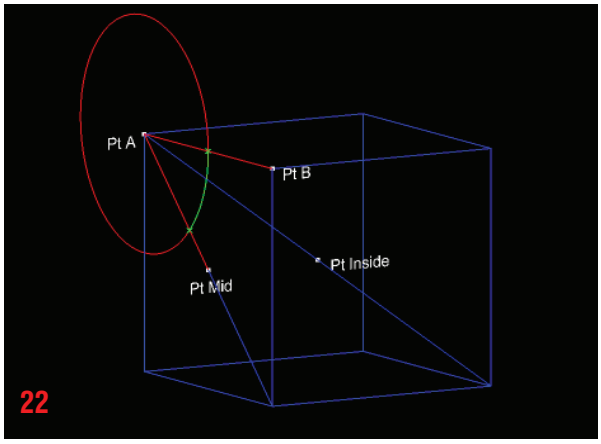
27. ***Vec2Pt (Vector 2Pt)***
- A : Pt B
- B : Pt A

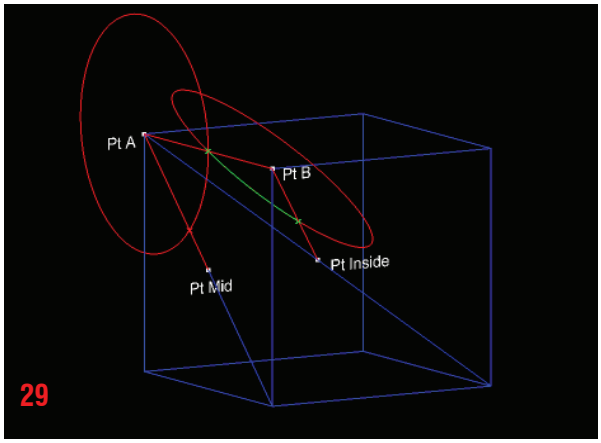
28. ***VRot (Rotate)***
- V : Vec2Pt(V)
- X : Pl Origin (Pl)
- A : π / -2 (see definitions)

29. ***Arc (Arc SED)***
- S : Eval(P) from 15.
- E : Eval(P) from 24.
- D : VRot(V)









30. ***Ln (Line)***
- A : Pt Mid
- B : Pt Inside

31. ***Eval (Evaluate Length)***
- C : Ln(L)
- L : A-B <- *parameter D*

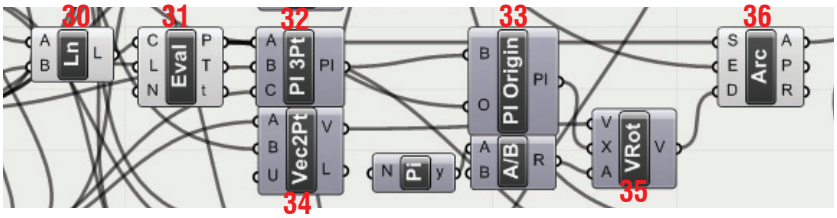
32. ***Pl 3Pt (Plane 3Pt)***
- A : Pt Mid
- B : Pt B
- C : Pt Inside

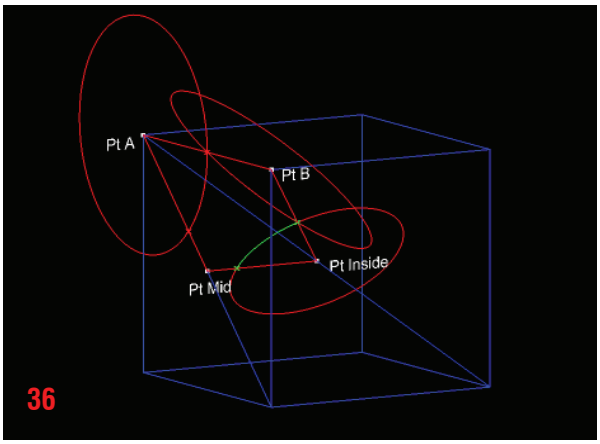
33. ***Pl Origin (Plane Origin)***
- B : Pl 3Pt
- O : Eval(P) from 31.

34. ***Vec2Pt (Vector 2Pt)***
- A : Pt Inside
- B : Pt Mid

35. ***VRot (Rotate)***
- V : Vec2Pt(V)
- X : Pl Origin (Pl)
- A : π / +2 (see definitions)

36. ***Arc (Arc SED)***
- S : Eval(P) from 31.
- E : Eval(P) from 24.
- D : VRot(V)





37. ***Pl 3Pt (Plane 3Pt)***
- A : Pt Mid
- B : Pt A
- C : Pt Inside

38. ***Pl Origin (Plane Origin)***
- B : Pl 3Pt
- O : Eval(P) from 17.

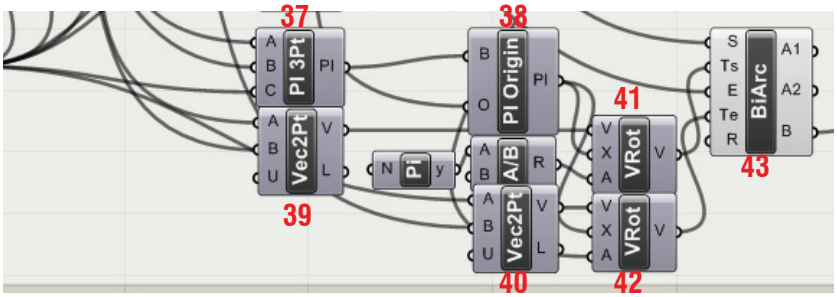
39. ***Vec2Pt (Vector 2Pt)***
- A : Pt Mid
- B : Pt A

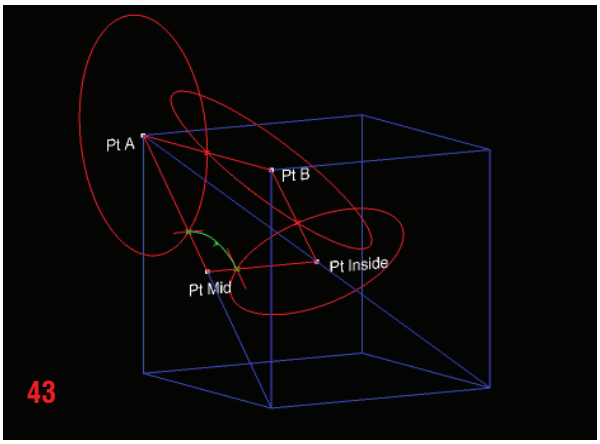
40. ***Vec2Pt (Vector 2Pt)***
- A : Pt Inside
- B : Pt Mid

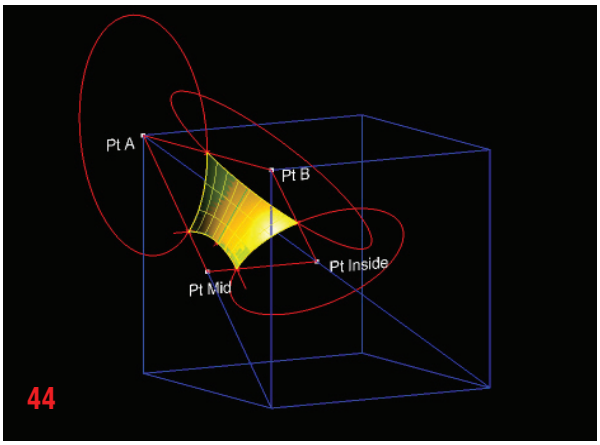
41. ***VRot (Rotate)***
- V : Vec2Pt(V) from 39.
- X : Pl Origin (Pl)
- A : π / +2 (see definitions)

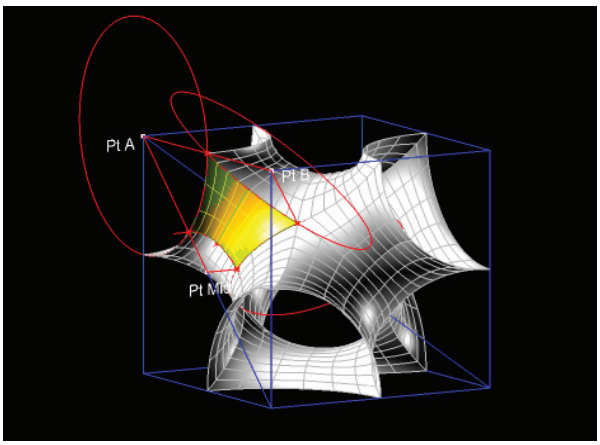
42. ***VRot (Rotate)***
- V : Vec2Pt(V) from 40.
- X : Pl Origin (Pl)
- A : π / -2 (see definitions)

43. ***BiArc (BiArc)***
- S : Eval(P) from 17.
- Ts : VRot(V) from 41.
- E : Eval(P) from 31.
- Te : VRot(V) from 42.
- R : default (0.5)









44. ***EdgeSrf***
- A : Arc(A) from 22.
- B : Arc(A) from 29.
- C : Arc(A) from 36.
- D : BiArc(B) from 43.

* Note
- The module can be mirrored diagonal axis to get the whole surfaces
- Or you can get the whole surface by shifting series of points (see 8_2 tutorial)

Appendix

- Definition map